See all versions of this document

Vivado Design Suite 7 Series FPGA and Zynq-7000 SoC Libraries Guide

UG953 (v2020.1) June 3, 2020

XILINX

XILINX

Chapter 1

Introduction

Overview

This HDL guide is part of the Vivado® Design Suite documentation collection.

This guide contains the following:

- Introduction
- Descriptions of each available macro
- A list of design elements supported in this architecture, organized by functional categories
- Descriptions of each available primitive

About Design Elements

This version of the Libraries Guide describes the valid design elements for 7 series architectures including Zynq®, and includes examples of instantiation code for each element. Instantiation templates are also supplied in a separate ZIP file, which you can find on www.xilinx.com linked to this file or within the Language Templates in the Vivado® Design Suite.

Design elements are divided into the following main categories:

- **Macros :** These elements are in the UniMacro library and the Xilinx Parameterized Macro library in the tool, and are used to instantiate elements that are complex to instantiate by just using the primitives. The synthesis tools will automatically expand these macros to their underlying primitives.
- **Primitives:** Xilinx components that are native to the architecture you are targeting.

Design Entry Methods

For each design element in this guide, Xilinx evaluates the options for using the design element, and recommends what we believe is the best solution for you. The options are:

- **Instantiation:** This component can be instantiated directly into the design. This method is useful if you want to control the exact use, implementation, or placement of the individual blocks.

- **Inference:** This component can be inferred by most supported synthesis tools. You should use this method if you want to have complete flexibility and portability of the code to multiple architectures. Inference also gives the tools the ability to optimize for performance, area, or power, as specified by the user to the synthesis tool.
- **IP Catalog:** This component can be instantiated from the IP Catalog. The IP Catalog maintains a library of IP Cores assembled from multiple primitives to form more complex functions, as well as interfaces to help in instantiation of the more complex primitives. References here to the IP Catalog generally refer to the latter, where you use the IP catalogt o assist in the use and integration of certain primitives into your design.
- **Macro Support:** This component has a UniMacro that can be used. These components are in the UniMacro library in the Xilinx tool, and are used to instantiate primitives that are too complex to instantiate by just using the primitives. The synthesis tools will automatically expand UniMacros to their underlying primitives.

XILINX

Chapter 2

Xilinx Parameterized Macros

About Xilinx Parameterized Macros

This section describes Xilinx® Parameterized Macros that can be used with 7 series FPGAs and Zynq®-7000 All Programmable SoC devices. The macros are organized alphabetically.

The following information is provided for each macro, where applicable:

- Name and description
- Schematic symbol
- Introduction
- Logic diagram (if any)
- Port descriptions
- Design Entry Method
- Available attributes
- Example instantiation templates
- Links to additional information

Enabling Xilinx Parameterized Macros

The following instructions describe how to prepare Vivado to use the XPM libraries.

1. Ensure Vivado can identify the XPMs.
   - When using the IDE and/or the project flow, the tools will parse the files added to the project and setup Vivado to recognize the XPMs.
   - When using the non-project flow, you must issue the auto_detect_xpm command.
2. Select the XPM template that you wish to use from below.
3. Copy the contents of the template and paste into your own source file.
4. Set parameters/generics, and wire ports according to the documentation provided as code comments.

***Note*:** Be sure to read and comply with all code comments to properly use the XPMs.

Testbench

A testbench for XPM CDC macros is available in the XPM CDC Testbench File.

A testbench for XPM FIFO macros is available in the XPM FIFO Testbench File.

Instantiation Templates

Instantiation templates for Xilinx Parameterized Macros are also available in Vivado, as well as in a downloadable ZIP file. Because PDF includes headers and footers if you copy text that spans pages, you should copy templates from Vivado or the downloaded ZIP file whenever possible.

Instantiation templates can be found on the Web in the Instantiation Templates for Xilinx Parameterizable Macros file.

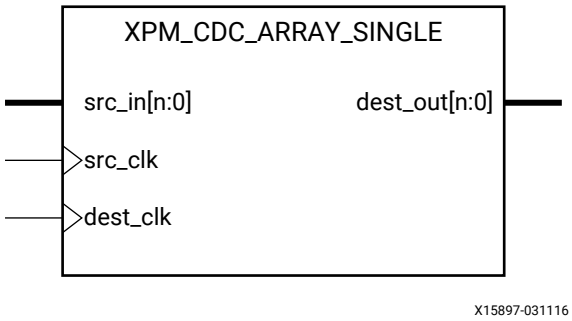
List of Xilinx Parameterized Macros

Design Element	Description
XPM_CDC_ARRAY_SINGLE	Parameterized Macro: Single-bit Array Synchronizer
XPM_CDC_ASYNC_RST	Parameterized Macro: Asynchronous Reset Synchronizer
XPM_CDC_GRAY	Parameterized Macro: Synchronizer via Gray Encoding
XPM_CDC_HANDSHAKE	Parameterized Macro: Bus Synchronizer with Full Handshake
XPM_CDC_PULSE	Parameterized Macro: Pulse Transfer
XPM_CDC_SINGLE	Parameterized Macro: Single-bit Synchronizer
XPM_CDC_SYNC_RST	Parameterized Macro: Synchronous Reset Synchronizer
XPM_FIFO_ASYNC	Parameterized Macro: Asynchronous FIFO
XPM_FIFO_AXIF	Parameterized Macro: AXI-Full FIFO
XPM_FIFO_AXIL	Parameterized Macro: AXI-Lite FIFO
XPM_FIFO_AXIS	Parameterized Macro: AXI Stream FIFO
XPM_FIFO_SYNC	Parameterized Macro: Synchronous FIFO
XPM_MEMORY_DPDISTRAM	Parameterized Macro: Dual Port Distributed RAM
XPM_MEMORY_DPROM	Parameterized Macro: Dual Port ROM
XPM_MEMORY_SDPRAM	Parameterized Macro: Simple Dual Port RAM
XPM_MEMORY_SPRAM	Parameterized Macro: Single Port RAM
XPM_MEMORY_SPROM	Parameterized Macro: Single Port ROM
XPM_MEMORY_TDPRAM	Parameterized Macro: True Dual Port RAM

XPM_CDC_ARRAY_SINGLE

Parameterized Macro: Single-bit Array Synchronizer

MACRO_GROUP: XPM
MACRO_SUBGROUP: XPM_CDC



Introduction

This macro synthesizes an array of single-bit signals from the source clock domain to the destination clock domain.

For proper operation, the input data must be sampled two or more times by the destination clock. You can define the number of register stages used in the synchronizers. An optional input register can be used to register the input in the source clock domain prior to it being synchronized. You can also enable a simulation feature to generate messages to report any potential misuse of the macro.

***Note*:** This macro expects that the each bit of the source array is independent, and does not have a defined relationship that needs to be preserved. If each bit of the array has a relationship that needs to be preserved, use the XPM_CDC_HANDSHAKE or XPM_CDC_GRAY macros.

Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
dest_clk	Input	1	NA	EDGE _RISING	Active	Clock signal for the destination clock domain.
dest_out	Output	WIDTH	dest_clk	NA	Active	src_in synchronized to the destination clock domain. This output is registered.
src_clk	Input	1	NA	EDGE _RISING	0	Unused when SRC_INPUT_REG = 0. Input clock signal for src_in if SRC_INPUT_REG = 1.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
src_in	Input	WIDTH	src_clk	NA	Active	Input single-bit array to be synchronized to destination clock domain. It is assumed that each bit of the array is unrelated to the others. This is reflected in the constraints applied to this macro. To transfer a binary value losslessly across the two clock domains, use the XPM_CDC_GRAY macro instead.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DEST_SYNC_FF	DECIMAL	2 to 10	4	Number of register stages used to synchronize signal in the destination clock domain.
INIT_SYNC_FF	DECIMAL	0, 1	0	0- Disable behavioral simulation initialization value(s) on synchronization registers. 1- Enable behavioral simulation initialization value(s) on synchronization registers.
SIM_ASSERT_CHK	DECIMAL	0, 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.
SRC_INPUT_REG	DECIMAL	1, 0	1	0- Do not register input (src_in) 1- Register input (src_in) once using src_clk
WIDTH	DECIMAL	1 to 1024	2	Width of single-bit array (src_in) that will be synchronized to destination clock domain.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_cdc_array_single: Single-bit Array Synchronizer
-- Xilinx Parameterized Macro, version 2020.1

xpm_cdc_array_single_inst : xpm_cdc_array_single
generic map (
   DEST_SYNC_FF => 4,   -- DECIMAL; range: 2-10
   INIT_SYNC_FF => 0,   -- DECIMAL; 0=disable simulation init values, 1=enable simulation init values
```

```
   SIM_ASSERT_CHK => 0, -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   SRC_INPUT_REG => 1,  -- DECIMAL; 0=do not register input, 1=register input
   WIDTH => 2           -- DECIMAL; range: 1-1024
)
port map (
   dest_out => dest_out, -- WIDTH-bit output: src_in synchronized to the destination clock domain. This
                         -- output is registered.

   dest_clk => dest_clk, -- 1-bit input: Clock signal for the destination clock domain.
   src_clk => src_clk,   -- 1-bit input: optional; required when SRC_INPUT_REG = 1
   src_in => src_in      -- WIDTH-bit input: Input single-bit array to be synchronized to destination clock
                         -- domain. It is assumed that each bit of the array is unrelated to the others.
                         -- This is reflected in the constraints applied to this macro. To transfer a binary
                         -- value losslessly across the two clock domains, use the XPM_CDC_GRAY macro
                         -- instead.

);

-- End of xpm_cdc_array_single_inst instantiation
```

Verilog Instantiation Template

```
// xpm_cdc_array_single: Single-bit Array Synchronizer
// Xilinx Parameterized Macro, version 2020.1

xpm_cdc_array_single #(
   .DEST_SYNC_FF(4),   // DECIMAL; range: 2-10
   .INIT_SYNC_FF(0),   // DECIMAL; 0=disable simulation init values, 1=enable simulation init values
   .SIM_ASSERT_CHK(0), // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .SRC_INPUT_REG(1),  // DECIMAL; 0=do not register input, 1=register input
   .WIDTH(2)           // DECIMAL; range: 1-1024
)
xpm_cdc_array_single_inst (
   .dest_out(dest_out), // WIDTH-bit output: src_in synchronized to the destination clock domain. This
                        // output is registered.

   .dest_clk(dest_clk), // 1-bit input: Clock signal for the destination clock domain.
   .src_clk(src_clk),   // 1-bit input: optional; required when SRC_INPUT_REG = 1
   .src_in(src_in)      // WIDTH-bit input: Input single-bit array to be synchronized to destination clock
                        // domain. It is assumed that each bit of the array is unrelated to the others. This
                        // is reflected in the constraints applied to this macro. To transfer a binary value
                        // losslessly across the two clock domains, use the XPM_CDC_GRAY macro instead.

);

// End of xpm_cdc_array_single_inst instantiation
```

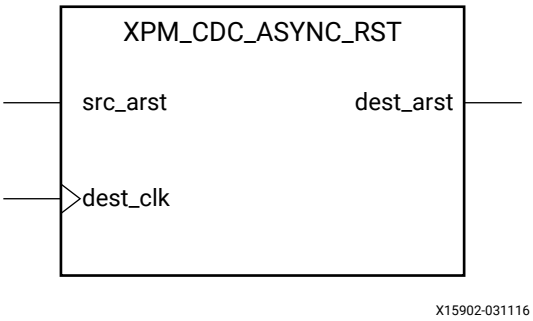
For More Information

- XPM CDC Testbench File

XPM_CDC_ASYNC_RST

Parameterized Macro: Asynchronous Reset Synchronizer

MACRO_GROUP: XPM
MACRO_SUBGROUP: XPM_CDC



Introduction

This macro synchronizes an asynchronous reset signal to the destination clock domain. The resulting reset output will be guaranteed to assert asynchronously in relation to the input but the deassertion of the output will always be synchronous to the destination clock domain.

You can define the polarity of the reset signal and the minimal output pulse width of the macro when asserted. The latter is controlled by defining the number of register stages used in the synchronizers.

***Note*:** The minimum input pulse assertion is dependent on the setup and hold requirement of the reset or set pin of the registers. See the respective DC and AC switching characteristics data sheets for the targeted architecture.

Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
dest_arst	Output	1	dest_clk	NA	Active	src_arst asynchronous reset signal synchronized to destination clock domain. This output is registered. NOTE: Signal asserts asynchronously but deasserts synchronously to dest_clk. Width of the reset signal is at least (DEST_SYNC_FF*dest_clk) period.
dest_clk	Input	1	NA	EDGE _RISING	Active	Destination clock.
src_arst	Input	1	NA	NA	Active	Source asynchronous reset signal.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DEST_SYNC_FF	DECIMAL	2 to 10	4	Number of register stages used to synchronize signal in the destination clock domain. This parameter also determines the minimum width of the asserted reset signal.
INIT_SYNC_FF	DECIMAL	0, 1	0	0- Disable behavioral simulation initialization value(s) on synchronization registers. 1- Enable behavioral simulation initialization value(s) on synchronization registers.
RST_ACTIVE_HIGH	DECIMAL	0, 1	0	Defines the polarity of the asynchronous reset signal. • 0- Active low asynchronous reset signal • 1- Active high asynchronous reset signal

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_cdc_async_rst: Asynchronous Reset Synchronizer
-- Xilinx Parameterized Macro, version 2020.1

xpm_cdc_async_rst_inst : xpm_cdc_async_rst
generic map (
   DEST_SYNC_FF => 4,     -- DECIMAL; range: 2-10
   INIT_SYNC_FF => 0,     -- DECIMAL; 0=disable simulation init values, 1=enable simulation init values
   RST_ACTIVE_HIGH => 0   -- DECIMAL; 0=active low reset, 1=active high reset
)
port map (
   dest_arst => dest_arst, -- 1-bit output: src_arst asynchronous reset signal synchronized to destination
                           -- clock domain. This output is registered. NOTE: Signal asserts asynchronously
                           -- but deasserts synchronously to dest_clk. Width of the reset signal is at least
                               -- (DEST_SYNC_FF*dest_clk) period.

   dest_clk => dest_clk,   -- 1-bit input: Destination clock.
   src_arst => src_arst    -- 1-bit input: Source asynchronous reset signal.
);

-- End of xpm_cdc_async_rst_inst instantiation
```

Verilog Instantiation Template

```
// xpm_cdc_async_rst: Asynchronous Reset Synchronizer
// Xilinx Parameterized Macro, version 2020.1

xpm_cdc_async_rst #(
   .DEST_SYNC_FF(4),     // DECIMAL; range: 2-10
   .INIT_SYNC_FF(0),     // DECIMAL; 0=disable simulation init values, 1=enable simulation init values
   .RST_ACTIVE_HIGH(0)   // DECIMAL; 0=active low reset, 1=active high reset
)
xpm_cdc_async_rst_inst (
   .dest_arst(dest_arst), // 1-bit output: src_arst asynchronous reset signal synchronized to destination
                          // clock domain. This output is registered. NOTE: Signal asserts asynchronously
                          // but deasserts synchronously to dest_clk. Width of the reset signal is at least
                             // (DEST_SYNC_FF*dest_clk) period.

   .dest_clk(dest_clk),   // 1-bit input: Destination clock.
   .src_arst(src_arst)    // 1-bit input: Source asynchronous reset signal.
);

// End of xpm_cdc_async_rst_inst instantiation
```

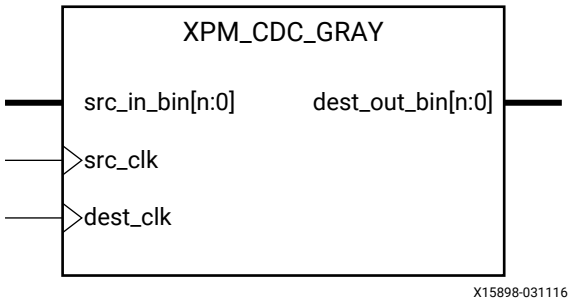
For More Information

- XPM CDC Testbench File

XPM_CDC_GRAY

Parameterized Macro: Synchronizer via Gray Encoding

MACRO_GROUP: XPM
MACRO_SUBGROUP: XPM_CDC



Introduction

This macro synchronizes a binary input from the source clock domain to the destination clock domain using gray code. For proper operation, the input data must be sampled two or more times by the destination clock.

This module takes the input binary signal, translates it into Gray code and registers it, synchronizes it to the destination clock domain, and then translates it back to a binary signal. You can define the number of register stages used in the synchronizers. You can also enable a simulation feature to generate messages to report any potential misuse of the macro.

Because this macro uses Gray encoding, the binary value provided to the macro must only increment or decrement by one to ensure that the signal being synchronized has two successive values that only differ by one bit. This will ensure lossless synchronization of a Gray coded bus. If the behavior of the binary value is not compatible to Gray encoding, use the XPM_CDC_HANDSHAKE macro or an alternate method of synchronizing the data to the destination clock domain.

An additional option (SIM_LOSSLESS_GRAY_CHK) is provided to report an error message when any binary input values are found to violate the Gray coding rule where two successive values must only increment or decrement by one.

Note: When the XPM_CDC_GRAY module is used in a design and `report_cdc` is run, the synchronizer in this module is reported as a warning of type CDC-6, Multi-bit synchronized with ASYNC_REG property. This warning is safe to ignore because the bus that is synchronized is gray-coded. Starting in 2018.3, this warning has been suppressed by adding a CDC-6 waiver to the Tcl constraint file.

You should run `report_cdc` to make sure the CDC structure is identified and that no critical warnings are generated, and also verify that `dest_clk` can sample `src_in_bin[n:0]` two or more times.

Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
dest_clk	Input	1	NA	EDGE _RISING	Active	Destination clock.
dest_out_bin	Output	WIDTH	dest_clk	NA	Active	Binary input bus (src_in_bin) synchronized to destination clock domain. This output is combinatorial unless REG_OUTPUT is set to 1.
src_clk	Input	1	NA	EDGE _RISING	Active	Source clock.
src_in_bin	Input	WIDTH	src_clk	NA	Active	Binary input bus that will be synchronized to the destination clock domain.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DEST_SYNC_FF	DECIMAL	2 to 10	4	Number of register stages used to synchronize signal in the destination clock domain.
INIT_SYNC_FF	DECIMAL	0, 1	0	0- Disable behavioral simulation initialization value(s) on synchronization registers. 1- Enable behavioral simulation initialization value(s) on synchronization registers.
REG_OUTPUT	DECIMAL	0, 1	0	0- Disable registered output 1- Enable registered output
SIM_ASSERT_CHK	DECIMAL	0, 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.
SIM_LOSSLESS _GRAY_CHK	DECIMAL	0, 1	0	0- Disable simulation message that reports whether src_in_bin is incrementing or decrementing by one, guaranteeing lossless synchronization of a gray coded bus. 1- Enable simulation message that reports whether src_in_bin is incrementing or decrementing by one, guaranteeing lossless synchronization of a gray coded bus.

Attribute	Type	Allowed Values	Default	Description
WIDTH	DECIMAL	2 to 32	2	Width of binary input bus that will be synchronized to destination clock domain.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_cdc_gray: Synchronizer via Gray Encoding
-- Xilinx Parameterized Macro, version 2020.1

xpm_cdc_gray_inst : xpm_cdc_gray
generic map (
   DEST_SYNC_FF => 4,           -- DECIMAL; range: 2-10
   INIT_SYNC_FF => 0,         -- DECIMAL; 0=disable simulation init values, 1=enable simulation init values
   REG_OUTPUT => 0,             -- DECIMAL; 0=disable registered output, 1=enable registered output
   SIM_ASSERT_CHK => 0,       -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   SIM_LOSSLESS_GRAY_CHK => 0, -- DECIMAL; 0=disable lossless check, 1=enable lossless check
   WIDTH => 2                   -- DECIMAL; range: 2-32
)
port map (
   dest_out_bin => dest_out_bin, -- WIDTH-bit output: Binary input bus (src_in_bin) synchronized to
                                 -- destination clock domain. This output is combinatorial unless REG_OUTPUT
                                 -- is set to 1.

   dest_clk => dest_clk,         -- 1-bit input: Destination clock.
   src_clk => src_clk,           -- 1-bit input: Source clock.
   src_in_bin => src_in_bin      -- WIDTH-bit input: Binary input bus that will be synchronized to the
                                 -- destination clock domain.

);

-- End of xpm_cdc_gray_inst instantiation
```

Verilog Instantiation Template

```
// xpm_cdc_gray: Synchronizer via Gray Encoding
// Xilinx Parameterized Macro, version 2020.1

xpm_cdc_gray #(
   .DEST_SYNC_FF(4),           // DECIMAL; range: 2-10
   .INIT_SYNC_FF(0),         // DECIMAL; 0=disable simulation init values, 1=enable simulation init values
   .REG_OUTPUT(0),             // DECIMAL; 0=disable registered output, 1=enable registered output
   .SIM_ASSERT_CHK(0),       // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .SIM_LOSSLESS_GRAY_CHK(0), // DECIMAL; 0=disable lossless check, 1=enable lossless check
   .WIDTH(2)                   // DECIMAL; range: 2-32
)
xpm_cdc_gray_inst (
   .dest_out_bin(dest_out_bin), // WIDTH-bit output: Binary input bus (src_in_bin) synchronized to
                                // destination clock domain. This output is combinatorial unless REG_OUTPUT
                                // is set to 1.

   .dest_clk(dest_clk),         // 1-bit input: Destination clock.
   .src_clk(src_clk),           // 1-bit input: Source clock.
   .src_in_bin(src_in_bin)      // WIDTH-bit input: Binary input bus that will be synchronized to the
                                // destination clock domain.

);

// End of xpm_cdc_gray_inst instantiation
```

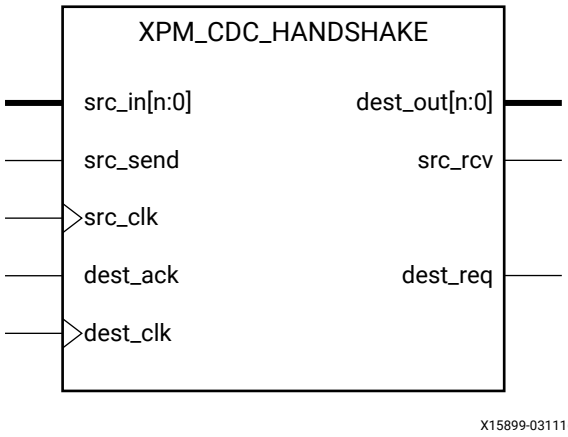
For More Information

- XPM CDC Testbench File

XPM_CDC_HANDSHAKE

Parameterized Macro: Bus Synchronizer with Full Handshake

MACRO_GROUP: XPM
MACRO_SUBGROUP: XPM_CDC



Introduction

This macro uses a handshake signaling to transfer an input bus from the source clock domain to the destination clock domain. One example of when this macro should be used is when the data being transferred is not compatible with the XPM_CDC_GRAY macro that uses Gray encoding.

For this macro to function correctly, a full handshake—an acknowledgement that the data transfer was received and a resetting of the handshake signals—must be completed before another data transfer is initiated.

You can define the number of register stages used in the synchronizers to transfer the handshake signals between the clock domains individually. You can also include internal handshake logic to acknowledge the receipt of data on the destination clock domain. When this feature is enabled, the output (dest_out) must be consumed immediately when the data valid (dest_req) is asserted.

You can also enable a simulation feature to generate messages to report any potential misuse of the macro. These messages will generate errors when the signaling provided to the macro violates the usage guidance above.

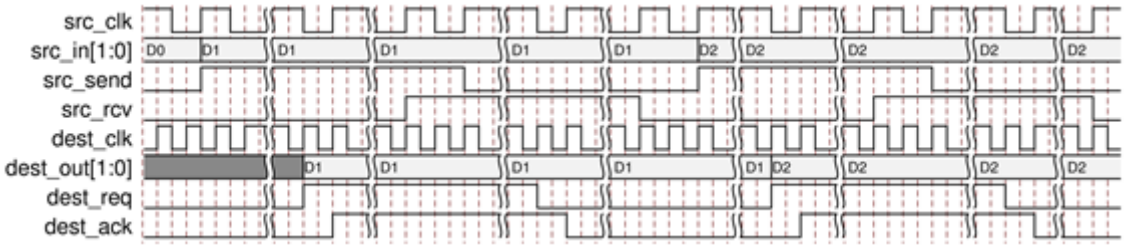
Note: When the XPM_CDC_HANDSHAKE module is used in a design and report_cdc is run, the data bus that is synchronized in this module is reported as a warning of type CDC-15, Clock Enable Controlled CDC. This warning is safe to ignore. Starting in 2018.3, this warning has been suppressed by adding a CDC-15 waiver to the Tcl constraint file.

You should run report_cdc to make sure the CDC structure is identified and that no critical warnings are generated, and also verify that dest_clk can sample src_in[n:0] two or more times.

External Handshake

The following waveform shows how back-to-back data is sent when the external handshake option is used.

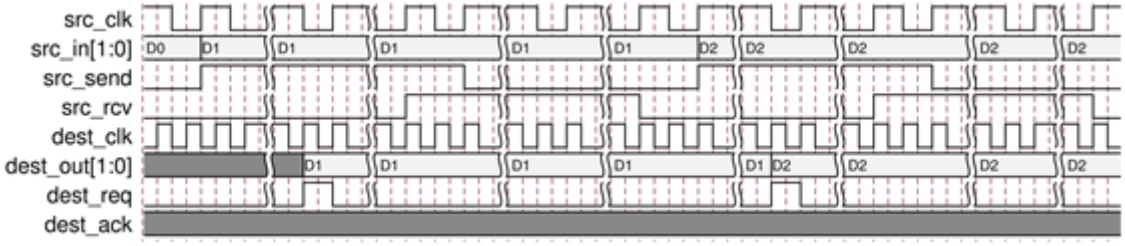
Figure 1: **External Handshake Timing Diagram**



Internal Handshake

The following waveform shows how back-to-back data is sent when the internal handshake option is enabled.

Figure 2: **Internal Handshake Timing Diagram**



Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
dest_ack	Input	1	dest_clk	LEVEL _HIGH	0	Destination logic acknowledgement if DEST_EXT_HSK = 1. Unused when DEST_EXT_HSK = 0. Asserting this signal indicates that data on dest_out has been captured by the destination logic. This signal should be deasserted once dest_req is deasserted, completing the handshake on the destination clock domain and indicating that the destination logic is ready for a new data transfer.
dest_clk	Input	1	NA	EDGE _RISING	Active	Destination clock.
dest_out	Output	WIDTH	dest_clk	NA	Active	Input bus (src_in) synchronized to destination clock domain. This output is registered.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
dest_req	Output	1	dest_clk	LEVEL _HIGH	Active	Assertion of this signal indicates that new dest_out data has been received and is ready to be used or captured by the destination logic. • When DEST_EXT_HSK = 1, this signal will deassert once the source handshake acknowledges that the destination clock domain has received the transferred data. • When DEST_EXT_HSK = 0, this signal asserts for one clock period when dest_out bus is valid. This output is registered.
src_clk	Input	1	NA	EDGE _RISING	Active	Source clock.
src_in	Input	WIDTH	src_clk	NA	Active	Input bus that will be synchronized to the destination clock domain.
src_rcv	Output	1	src_clk	LEVEL _HIGH	Active	Acknowledgement from destination logic that src_in has been received. This signal will be deasserted once destination handshake has fully completed, thus completing a full data transfer. This output is registered.
src_send	Input	1	src_clk	LEVEL _HIGH	Active	Assertion of this signal allows the src_in bus to be synchronized to the destination clock domain. • This signal should only be asserted when src_rcv is deasserted, indicating that the previous data transfer is complete. • This signal should only be deasserted once src_rcv is asserted, acknowledging that the src_in has been received by the destination logic.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DEST_EXT_HSK	DECIMAL	1, 0	1	0- An internal handshake will be implemented in the macro to acknowledge receipt of data on the destination clock domain. When using this option, the valid dest_out output must be consumed immediately to avoid any data loss. 1- External handshake logic must be implemented by the user to acknowledge receipt of data on the destination clock domain.
DEST_SYNC_FF	DECIMAL	2 to 10	4	Number of register stages used to synchronize signal in the destination clock domain.
INIT_SYNC_FF	DECIMAL	0, 1	0	0- Disable behavioral simulation initialization value(s) on synchronization registers. 1- Enable behavioral simulation initialization value(s) on synchronization registers.
SIM_ASSERT_CHK	DECIMAL	0, 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.
SRC_SYNC_FF	DECIMAL	2 to 10	4	Number of register stages used to synchronize signal in the source clock domain.
WIDTH	DECIMAL	1 to 1024	1	Width of bus that will be synchronized to destination clock domain.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_cdc_handshake: Bus Synchronizer with Full Handshake
-- Xilinx Parameterized Macro, version 2020.1

xpm_cdc_handshake_inst : xpm_cdc_handshake
generic map (
   DEST_EXT_HSK => 1,   -- DECIMAL; 0=internal handshake, 1=external handshake
   DEST_SYNC_FF => 4,   -- DECIMAL; range: 2-10
   INIT_SYNC_FF => 0,   -- DECIMAL; 0=disable simulation init values, 1=enable simulation init values
   SIM_ASSERT_CHK => 0, -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   SRC_SYNC_FF => 4,    -- DECIMAL; range: 2-10
   WIDTH => 1           -- DECIMAL; range: 1-1024
)
port map (
   dest_out => dest_out, -- WIDTH-bit output: Input bus (src_in) synchronized to destination clock domain.
                         -- This output is registered.

   dest_req => dest_req, -- 1-bit output: Assertion of this signal indicates that new dest_out data has been
                         -- received and is ready to be used or captured by the destination logic. When
                         -- DEST_EXT_HSK = 1, this signal will deassert once the source handshake
                         -- acknowledges that the destination clock domain has received the transferred
                         -- data. When DEST_EXT_HSK = 0, this signal asserts for one clock period when
                         -- dest_out bus is valid. This output is registered.

   src_rcv => src_rcv,   -- 1-bit output: Acknowledgement from destination logic that src_in has been
```

```
                        -- received. This signal will be deasserted once destination handshake has fully
                        -- completed, thus completing a full data transfer. This output is registered.

   dest_ack => dest_ack, -- 1-bit input: optional; required when DEST_EXT_HSK = 1
   dest_clk => dest_clk, -- 1-bit input: Destination clock.
   src_clk => src_clk,   -- 1-bit input: Source clock.
   src_in => src_in,     -- WIDTH-bit input: Input bus that will be synchronized to the destination clock
                            -- domain.

   src_send => src_send  -- 1-bit input: Assertion of this signal allows the src_in bus to be synchronized
                        -- to the destination clock domain. This signal should only be asserted when
                        -- src_rcv is deasserted, indicating that the previous data transfer is complete.
                        -- This signal should only be deasserted once src_rcv is asserted, acknowledging
                            -- that the src_in has been received by the destination logic.

);

-- End of xpm_cdc_handshake_inst instantiation
```

Verilog Instantiation Template

```
// xpm_cdc_handshake: Bus Synchronizer with Full Handshake
// Xilinx Parameterized Macro, version 2020.1

xpm_cdc_handshake #(
   .DEST_EXT_HSK(1),   // DECIMAL; 0=internal handshake, 1=external handshake
   .DEST_SYNC_FF(4),   // DECIMAL; range: 2-10
   .INIT_SYNC_FF(0),   // DECIMAL; 0=disable simulation init values, 1=enable simulation init values
   .SIM_ASSERT_CHK(0), // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .SRC_SYNC_FF(4),    // DECIMAL; range: 2-10
   .WIDTH(1)           // DECIMAL; range: 1-1024
)
xpm_cdc_handshake_inst (
   .dest_out(dest_out), // WIDTH-bit output: Input bus (src_in) synchronized to destination clock domain.
                           // This output is registered.

   .dest_req(dest_req), // 1-bit output: Assertion of this signal indicates that new dest_out data has been
                        // received and is ready to be used or captured by the destination logic. When
                           // DEST_EXT_HSK = 1, this signal will deassert once the source handshake
                        // acknowledges that the destination clock domain has received the transferred data.
                        // When DEST_EXT_HSK = 0, this signal asserts for one clock period when dest_out bus
                           // is valid. This output is registered.

   .src_rcv(src_rcv),   // 1-bit output: Acknowledgement from destination logic that src_in has been
                        // received. This signal will be deasserted once destination handshake has fully
                        // completed, thus completing a full data transfer. This output is registered.

   .dest_ack(dest_ack), // 1-bit input: optional; required when DEST_EXT_HSK = 1
   .dest_clk(dest_clk), // 1-bit input: Destination clock.
   .src_clk(src_clk),   // 1-bit input: Source clock.
   .src_in(src_in),     // WIDTH-bit input: Input bus that will be synchronized to the destination clock
                           // domain.

   .src_send(src_send)  // 1-bit input: Assertion of this signal allows the src_in bus to be synchronized to
                        // the destination clock domain. This signal should only be asserted when src_rcv is
                        // deasserted, indicating that the previous data transfer is complete. This signal
                        // should only be deasserted once src_rcv is asserted, acknowledging that the src_in
                           // has been received by the destination logic.

);

// End of xpm_cdc_handshake_inst instantiation
```

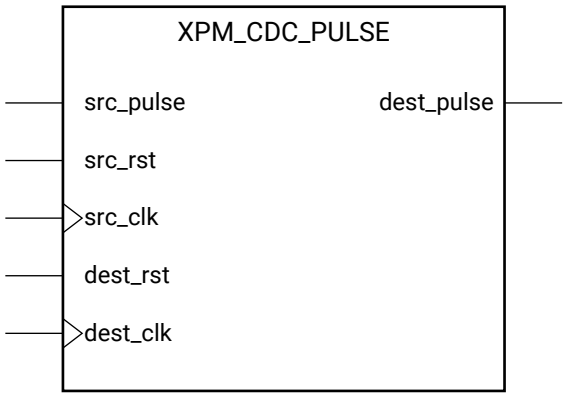
For More Information

- XPM CDC Testbench File

XPM_CDC_PULSE

Parameterized Macro: Pulse Transfer

MACRO_GROUP: XPM
MACRO_SUBGROUP: XPM_CDC



X15900-031116

Introduction

This macro synchronizes a pulse in the source clock domain to the destination clock domain. A pulse of any size in the source clock domain, if initiated correctly, will generate a pulse the size of a single destination clock period.

For proper operation, the input data must be sampled two or more times by the destination clock. You can define the number of register stages used in the synchronizers. An optional source and destination reset may be used to reset the pulse transfer logic. You can also enable a simulation feature to generate messages which report any potential misuse of the macro.

The implementation of this macro requires some feedback logic. When simulating the macro without the optional reset signals, the input pulse signal (src_pulse) must always be defined because there is no reset logic to recover from an undefined or 'x' propagating through the macro.

This macro also requires the following minimum gap between subsequent pulse inputs:

```
2*(larger(src_clk period, dest_clk period))
```

The minimum gap is measured between the falling edge of a src_pulse to the rising edge of the next src_pulse. This minimum gap will guarantee that each rising edge of src_pulse will generate a pulse the size of one dest_clk period in the destination clock domain.

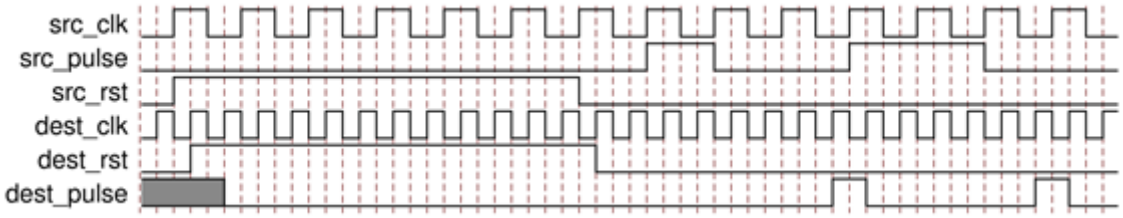
When using the optional reset signals, src_rst and dest_rst must be asserted simultaneously for at least the following duration to fully reset all the logic in the macro:

```
((DEST_SYNC_FF+2)*dest_clk_period) + (2*src_clk_period)
```

When reset is asserted, the input pulse signal should not toggle and the output pulse signal is not valid and should be ignored.

The following waveform demonstrates how to reset the macro and transfer back-to-back pulses while abiding the minimum gap between each pulse.

Figure 3: **Timing for Macro and Transfer Back-to-Back Pulses**



Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
dest_clk	Input	1	NA	EDGE _RISING	Active	Destination clock.
dest_pulse	Output	1	dest_clk	LEVEL _HIGH	Active	Outputs a pulse the size of one dest_clk period when a pulse transfer is correctly initiated on src_pulse input. This output is combinatorial unless REG_OUTPUT is set to 1.
dest_rst	Input	1	dest_clk	LEVEL _HIGH	0	Unused when RST_USED = 0. Destination reset signal if RST_USED = 1. Resets all logic in destination clock domain. To fully reset the macro, src_rst and dest_rst must be asserted simultaneously for at least ((DEST_SYNC_FF +2)*dest_clk_period) + (2*src_clk_period).
src_clk	Input	1	NA	EDGE _RISING	Active	Source clock.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
src_pulse	Input	1	src_clk	EDGE _RISING	Active	Rising edge of this signal initiates a pulse transfer to the destination clock domain. The minimum gap between each pulse transfer must be at the minimum 2*(larger(src_clk period, dest_clk period)). This is measured between the falling edge of a src_pulse to the rising edge of the next src_pulse. This minimum gap will guarantee that each rising edge of src_pulse will generate a pulse the size of one dest_clk period in the destination clock domain. When RST_USED = 1, pulse transfers will not be guaranteed while src_rst and/or dest_rst are asserted.
src_rst	Input	1	src_clk	LEVEL _HIGH	0	Unused when RST_USED = 0. Source reset signal if RST_USED = 1. Resets all logic in source clock domain. To fully reset the macro, src_rst and dest_rst must be asserted simultaneously for at least ((DEST_SYNC_FF +2)*dest_clk_period) + (2*src_clk_period).

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DEST_SYNC_FF	DECIMAL	2 to 10	4	Number of register stages used to synchronize signal in the destination clock domain.
INIT_SYNC_FF	DECIMAL	0, 1	0	0- Disable behavioral simulation initialization value(s) on synchronization registers. 1- Enable behavioral simulation initialization value(s) on synchronization registers.
REG_OUTPUT	DECIMAL	0, 1	0	0- Disable registered output 1- Enable registered output
RST_USED	DECIMAL	1, 0	1	0 - No resets implemented. 1 - Resets implemented. When RST_USED = 0, src_pulse input must always be defined during simulation since there is no reset logic to recover from an x-propagating through the macro.

Attribute	Type	Allowed Values	Default	Description
SIM_ASSERT_CHK	DECIMAL	0, 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_cdc_pulse: Pulse Transfer
-- Xilinx Parameterized Macro, version 2020.1

xpm_cdc_pulse_inst : xpm_cdc_pulse
generic map (
   DEST_SYNC_FF => 4,    -- DECIMAL; range: 2-10
   INIT_SYNC_FF => 0,    -- DECIMAL; 0=disable simulation init values, 1=enable simulation init values
   REG_OUTPUT => 0,      -- DECIMAL; 0=disable registered output, 1=enable registered output
   RST_USED => 1,        -- DECIMAL; 0=no reset, 1=implement reset
   SIM_ASSERT_CHK => 0   -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
)
port map (
   dest_pulse => dest_pulse, -- 1-bit output: Outputs a pulse the size of one dest_clk period when a pulse
                             -- transfer is correctly initiated on src_pulse input. This output is
                             -- combinatorial unless REG_OUTPUT is set to 1.

   dest_clk => dest_clk,     -- 1-bit input: Destination clock.
   dest_rst => dest_rst,     -- 1-bit input: optional; required when RST_USED = 1
   src_clk => src_clk,       -- 1-bit input: Source clock.
   src_pulse => src_pulse,   -- 1-bit input: Rising edge of this signal initiates a pulse transfer to the
                             -- destination clock domain. The minimum gap between each pulse transfer must
                             -- be at the minimum 2*(larger(src_clk period, dest_clk period)). This is
                             -- measured between the falling edge of a src_pulse to the rising edge of the
                             -- next src_pulse. This minimum gap will guarantee that each rising edge of
                             -- src_pulse will generate a pulse the size of one dest_clk period in the
                             -- destination clock domain. When RST_USED = 1, pulse transfers will not be
                             -- guaranteed while src_rst and/or dest_rst are asserted.

   src_rst => src_rst        -- 1-bit input: optional; required when RST_USED = 1
);

-- End of xpm_cdc_pulse_inst instantiation
```

Verilog Instantiation Template

```
// xpm_cdc_pulse: Pulse Transfer
// Xilinx Parameterized Macro, version 2020.1

xpm_cdc_pulse #(
   .DEST_SYNC_FF(4),   // DECIMAL; range: 2-10
   .INIT_SYNC_FF(0),   // DECIMAL; 0=disable simulation init values, 1=enable simulation init values
   .REG_OUTPUT(0),     // DECIMAL; 0=disable registered output, 1=enable registered output
   .RST_USED(1),       // DECIMAL; 0=no reset, 1=implement reset
   .SIM_ASSERT_CHK(0)  // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
)
xpm_cdc_pulse_inst (
   .dest_pulse(dest_pulse), // 1-bit output: Outputs a pulse the size of one dest_clk period when a pulse
                            // transfer is correctly initiated on src_pulse input. This output is
                            // combinatorial unless REG_OUTPUT is set to 1.
```

```
   .dest_clk(dest_clk),      // 1-bit input: Destination clock.
   .dest_rst(dest_rst),      // 1-bit input: optional; required when RST_USED = 1
   .src_clk(src_clk),        // 1-bit input: Source clock.
   .src_pulse(src_pulse),   // 1-bit input: Rising edge of this signal initiates a pulse transfer to the
                            // destination clock domain. The minimum gap between each pulse transfer must be
                            // at the minimum 2*(larger(src_clk period, dest_clk period)). This is measured
                            // between the falling edge of a src_pulse to the rising edge of the next
                            // src_pulse. This minimum gap will guarantee that each rising edge of src_pulse
                            // will generate a pulse the size of one dest_clk period in the destination
                            // clock domain. When RST_USED = 1, pulse transfers will not be guaranteed while
                               // src_rst and/or dest_rst are asserted.

   .src_rst(src_rst)         // 1-bit input: optional; required when RST_USED = 1
);

// End of xpm_cdc_pulse_inst instantiation
```

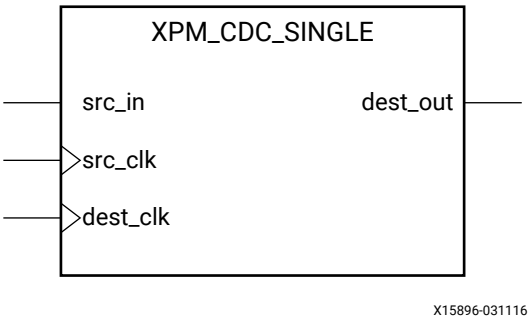
For More Information

- XPM CDC Testbench File

XPM_CDC_SINGLE

Parameterized Macro: Single-bit Synchronizer

MACRO_GROUP: XPM
MACRO_SUBGROUP: XPM_CDC



Introduction

This macro synchronizes a one bit signal from the source clock domain to the destination clock domain.

For proper operation, the input data must be sampled two or more times by the destination clock. You can define the number of register stages used in the synchronizers. An optional input register may be used to register the input in the source clock domain prior to it being synchronized. You can also enable a simulation feature to generate messages to report any potential misuse of the macro.

Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
dest_clk	Input	1	NA	EDGE _RISING	Active	Clock signal for the destination clock domain.
dest_out	Output	1	dest_clk	NA	Active	src_in synchronized to the destination clock domain. This output is registered.
src_clk	Input	1	NA	EDGE _RISING	0	Input clock signal for src_in if SRC_INPUT_REG = 1. Unused when SRC_INPUT_REG = 0.
src_in	Input	1	src_clk	NA	Active	Input signal to be synchronized to dest_clk domain.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DEST_SYNC_FF	DECIMAL	2 to 10	4	Number of register stages used to synchronize signal in the destination clock domain.
INIT_SYNC_FF	DECIMAL	0, 1	0	0- Disable behavioral simulation initialization value(s) on synchronization registers. 1- Enable behavioral simulation initialization value(s) on synchronization registers.
SIM_ASSERT_CHK	DECIMAL	0, 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.
SRC_INPUT_REG	DECIMAL	1, 0	1	0- Do not register input (src_in) 1- Register input (src_in) once using src_clk

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_cdc_single: Single-bit Synchronizer
-- Xilinx Parameterized Macro, version 2020.1

xpm_cdc_single_inst : xpm_cdc_single
generic map (
   DEST_SYNC_FF => 4,    -- DECIMAL; range: 2-10
   INIT_SYNC_FF => 0,    -- DECIMAL; 0=disable simulation init values, 1=enable simulation init values
   SIM_ASSERT_CHK => 0, -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   SRC_INPUT_REG => 1    -- DECIMAL; 0=do not register input, 1=register input
)
port map (
   dest_out => dest_out, -- 1-bit output: src_in synchronized to the destination clock domain. This output
                         -- is registered.

   dest_clk => dest_clk, -- 1-bit input: Clock signal for the destination clock domain.
   src_clk => src_clk,   -- 1-bit input: optional; required when SRC_INPUT_REG = 1
   src_in => src_in      -- 1-bit input: Input signal to be synchronized to dest_clk domain.
);

-- End of xpm_cdc_single_inst instantiation
```

Verilog Instantiation Template

```
// xpm_cdc_single: Single-bit Synchronizer
// Xilinx Parameterized Macro, version 2020.1

xpm_cdc_single #(
   .DEST_SYNC_FF(4),   // DECIMAL; range: 2-10
   .INIT_SYNC_FF(0),   // DECIMAL; 0=disable simulation init values, 1=enable simulation init values
   .SIM_ASSERT_CHK(0), // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .SRC_INPUT_REG(1)   // DECIMAL; 0=do not register input, 1=register input
)
xpm_cdc_single_inst (
   .dest_out(dest_out), // 1-bit output: src_in synchronized to the destination clock domain. This output is
                        // registered.

   .dest_clk(dest_clk), // 1-bit input: Clock signal for the destination clock domain.
   .src_clk(src_clk),   // 1-bit input: optional; required when SRC_INPUT_REG = 1
   .src_in(src_in)      // 1-bit input: Input signal to be synchronized to dest_clk domain.
);

// End of xpm_cdc_single_inst instantiation
```

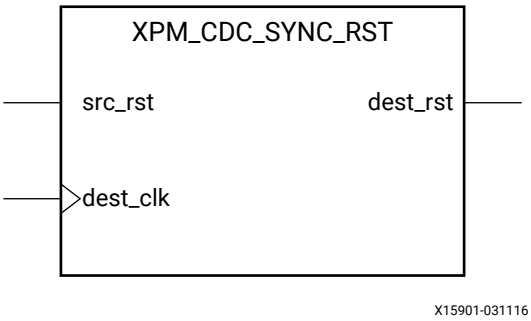
For More Information

- XPM CDC Testbench File

XPM_CDC_SYNC_RST

Parameterized Macro: Synchronous Reset Synchronizer

MACRO_GROUP: XPM
MACRO_SUBGROUP: XPM_CDC



Introduction

This macro synchronizes a reset singal to the destination clock domain. Unlike the XPM_CDC_ASYNC_RST macro, the generated output will both assert and deassert synchronously to the destination clock domain.

For proper operation, the input data must be sampled two or more times by the destination clock. You can define the number of register stages used in the synchronizers and the initial value of these registers after configuration. An optional input register may be used to register the input in the source clock domain prior to it being synchronized. You can also enable a simulation feature to generate messages which report any potential misuse of the macro.

Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
dest_clk	Input	1	NA	EDGE _RISING	Active	Destination clock.
dest_rst	Output	1	dest_clk	NA	Active	src_rst synchronized to the destination clock domain. This output is registered.
src_rst	Input	1	NA	NA	Active	Source reset signal.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DEST_SYNC_FF	DECIMAL	2 to 10	4	Number of register stages used to synchronize signal in the destination clock domain.
INIT	DECIMAL	1, 0	1	0- Initializes synchronization registers to 0 1- Initializes synchronization registers to 1 The option to initialize the synchronization registers means that there is no complete x-propagation behavior modeled in this macro. For complete x-propagation modelling, use the xpm_cdc_single macro.
INIT_SYNC_FF	DECIMAL	0, 1	0	0- Disable behavioral simulation initialization value(s) on synchronization registers. 1- Enable behavioral simulation initialization value(s) on synchronization registers.
SIM_ASSERT_CHK	DECIMAL	0, 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_cdc_sync_rst: Synchronous Reset Synchronizer
-- Xilinx Parameterized Macro, version 2020.1

xpm_cdc_sync_rst_inst : xpm_cdc_sync_rst
generic map (
   DEST_SYNC_FF => 4,   -- DECIMAL; range: 2-10
   INIT => 1,           -- DECIMAL; 0=initialize synchronization registers to 0, 1=initialize
                        -- synchronization registers to 1
   INIT_SYNC_FF => 0,   -- DECIMAL; 0=disable simulation init values, 1=enable simulation init values
   SIM_ASSERT_CHK => 0  -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
)
port map (
   dest_rst => dest_rst, -- 1-bit output: src_rst synchronized to the destination clock domain. This output
                         -- is registered.

   dest_clk => dest_clk, -- 1-bit input: Destination clock.
   src_rst => src_rst    -- 1-bit input: Source reset signal.
);

-- End of xpm_cdc_sync_rst_inst instantiation
```

Verilog Instantiation Template

```
// xpm_cdc_sync_rst: Synchronous Reset Synchronizer
// Xilinx Parameterized Macro, version 2020.1

xpm_cdc_sync_rst #(
   .DEST_SYNC_FF(4),   // DECIMAL; range: 2-10
   .INIT(1),           // DECIMAL; 0=initialize synchronization registers to 0, 1=initialize synchronization
                       // registers to 1
   .INIT_SYNC_FF(0),   // DECIMAL; 0=disable simulation init values, 1=enable simulation init values
   .SIM_ASSERT_CHK(0)  // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
)
xpm_cdc_sync_rst_inst (
   .dest_rst(dest_rst), // 1-bit output: src_rst synchronized to the destination clock domain. This output
                        // is registered.

   .dest_clk(dest_clk), // 1-bit input: Destination clock.
   .src_rst(src_rst)    // 1-bit input: Source reset signal.
);

// End of xpm_cdc_sync_rst_inst instantiation
```

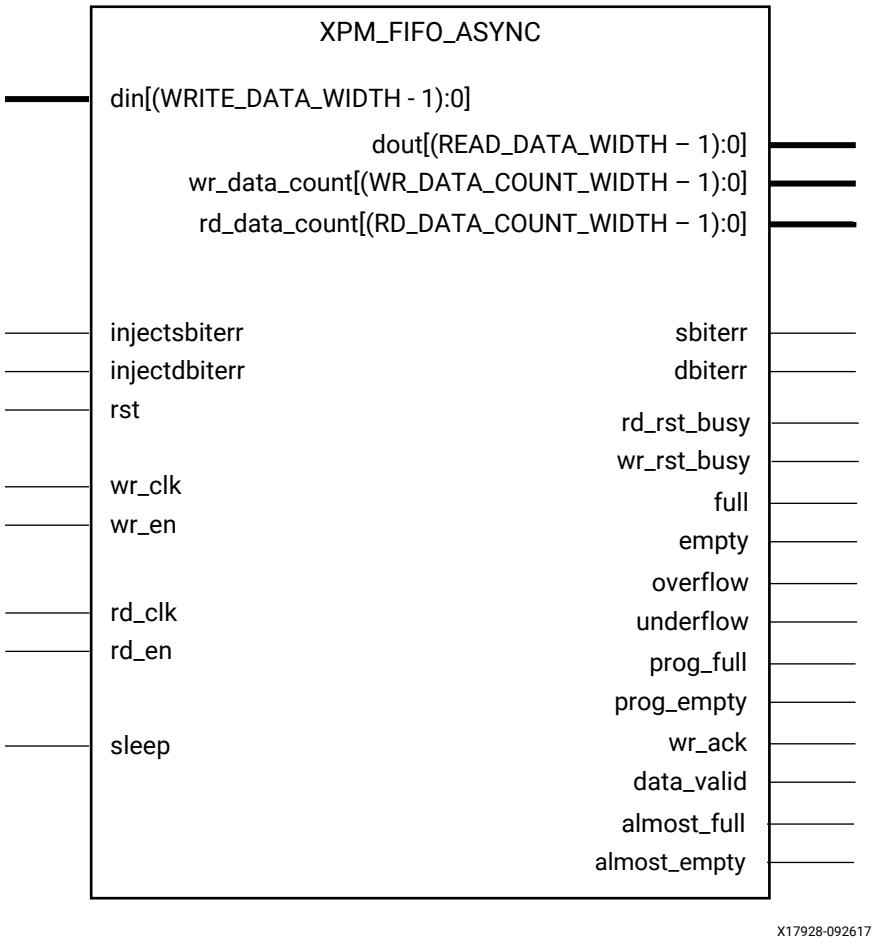
For More Information

- XPM CDC Testbench File

XPM_FIFO_ASYNC

Parameterized Macro: Asynchronous FIFO

MACRO_GROUP: XPM
MACRO_SUBGROUP: XPM_FIFO



Introduction

This macro is used to instantiate an asynchronous FIFO.

The following describes the basic write and read operation of an XPM_FIFO instance. It does not distinguish between FIFO types, clock domain or read mode.

- After a user issues a reset, the user should wait until the busy signals go low before issuing another reset.

- All synchronous signals are sensitive to the rising edge of wr_clk/rd_clk, which is assumed to be a buffered and toggling clock signal behaving according to target device and FIFO/memory primitive requirements.
- A write operation is performed when the FIFO is not full and wr_en is asserted on each wr_clk cycle.
- A read operation is performed when the FIFO is not empty and rd_en is asserted on each rd_clk cycle.
- The number of clock cycles required for XPM FIFO to react to dout, full, and empty changes depends on the CLOCK_DOMAIN, READ_MODE, and FIFO_READ_LATENCY settings.
  - It can take more than one rd_clk cycle to deassert empty due to write operation (wr_en = 1).
  - It can take more than one rd_clk cycle to present the read data on dout port upon assertion of rd_en.
  - It may take more than one wr_clk cycle to deassert full due to read operation (rd_en = 1).
- All write operations are gated by the value of wr_en and full on the initiating wr_clk cycle.
- All read operations are gated by the value of rd_en and empty on the initiating rd_clk cycle.
- The wr_en input has no effect when full is asserted on the coincident wr_clk cycle.
- The rd_en input has no effect when empty is asserted on the coincident rd_clk cycle.
- Undriven or unknown values provided on module inputs will produce undefined output port behavior.
- wr_en/rd_en should not be toggled when reset (rst) or wr_rst_busy or rd_rst_busy is asserted.
- Assertion/deassertion of prog_full happens only when full is deasserted.
- Assertion/deassertion of prog_empty happens only when empty is deasserted.

***Note*:** In an asynchronous FIFO (that is, two independent clocks), the RELATED_CLOCKS attribute should be set to TRUE only if both the wr_clk and rd_clk are generated from the same source.

Timing Diagrams

Figure 4: **Reset Behavior**

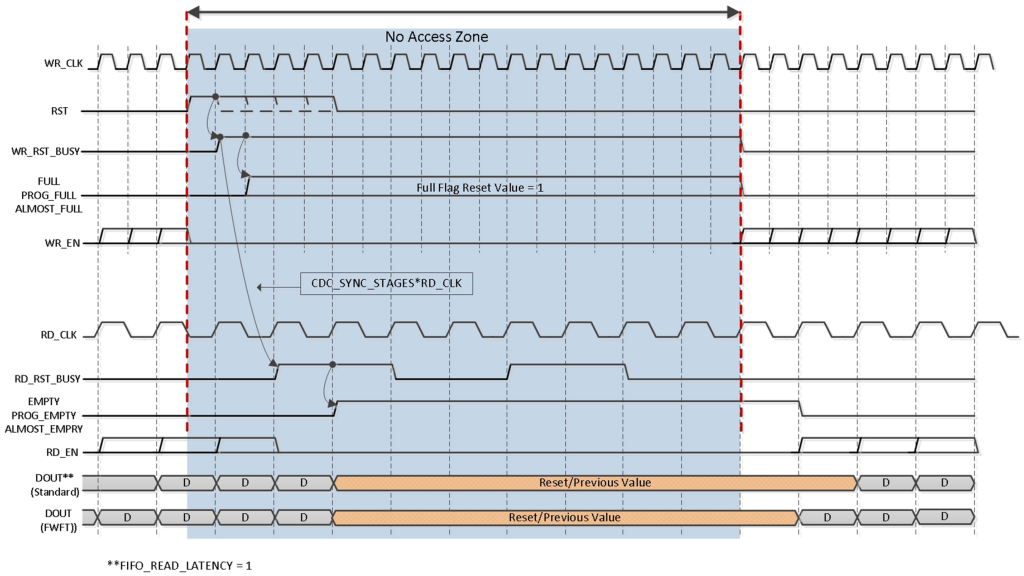


Figure 5: **Standard Write Operation**

FIFO_WRITE_DEPTH=16, PROG_FULL_THRESH=6

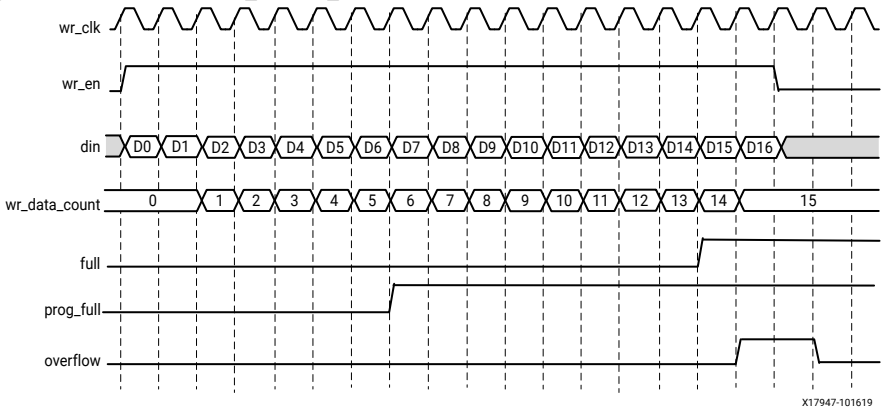


Figure 6: **Standard Read Operation**

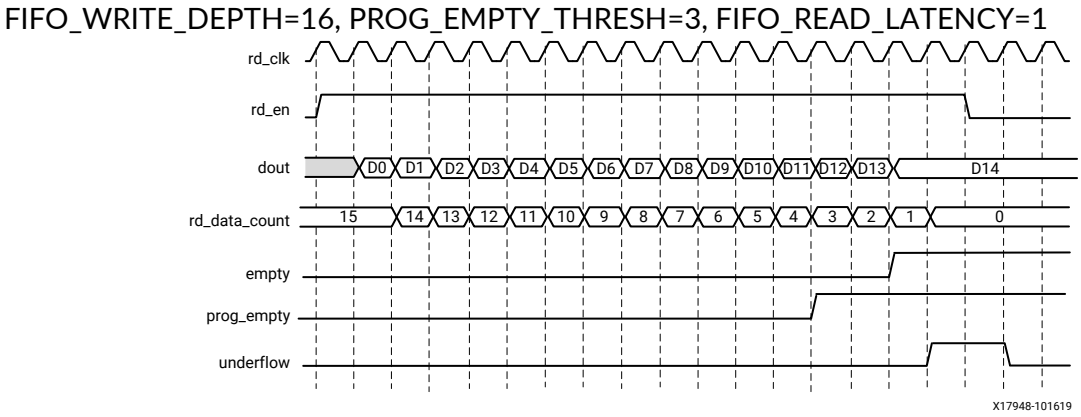


Figure 7: **Standard Read Operation**

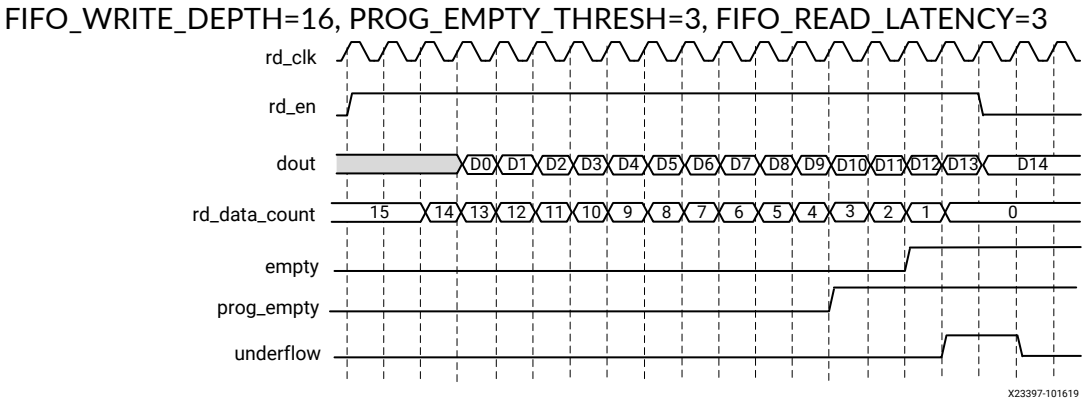


Figure 8: **Write Operation**

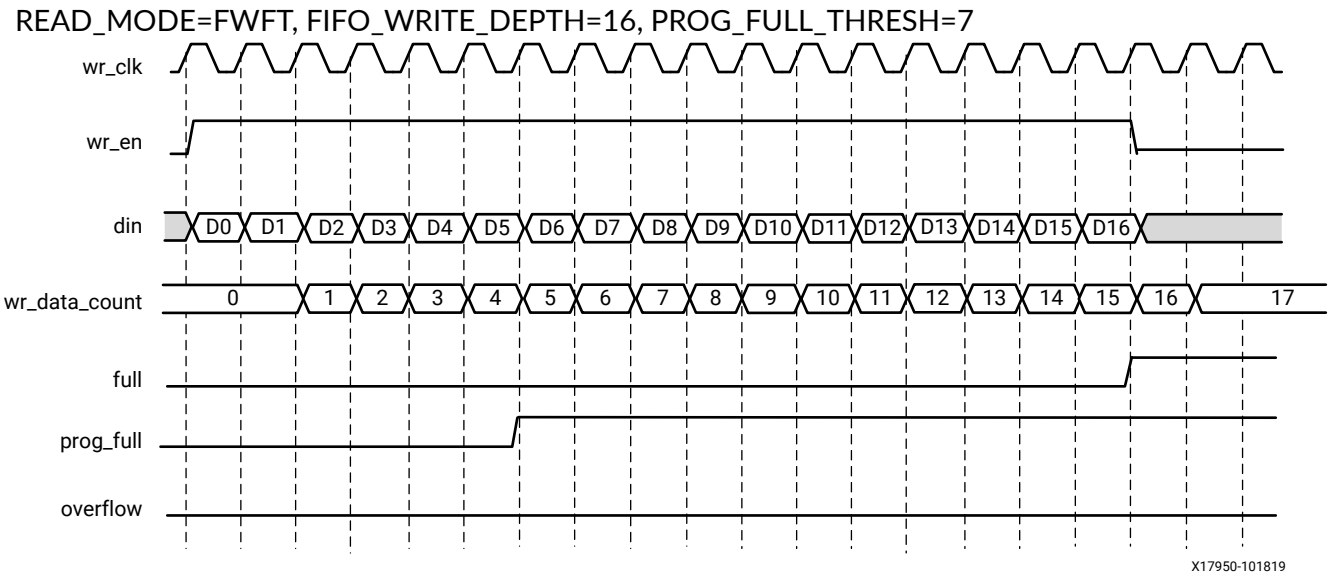


Figure 9: **Read Operation**

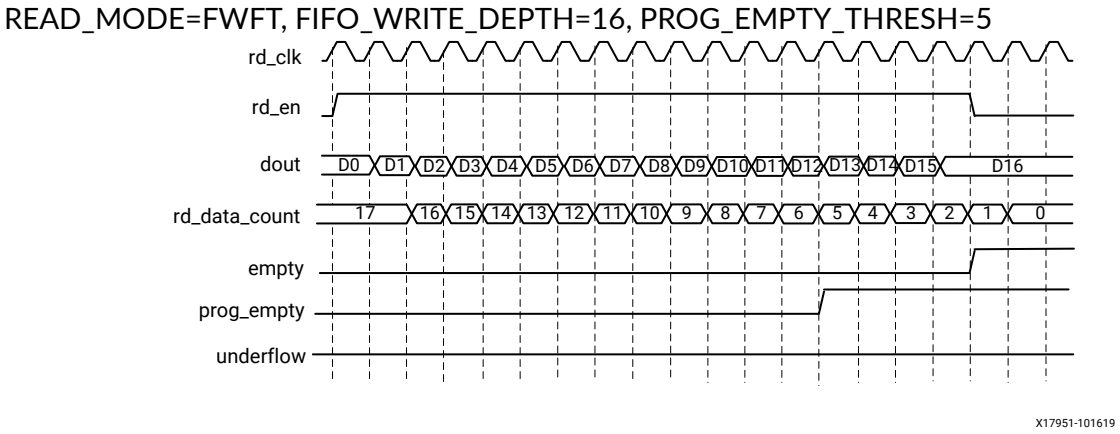


Figure 10: **Standard Write Operation with Empty Deassertion**

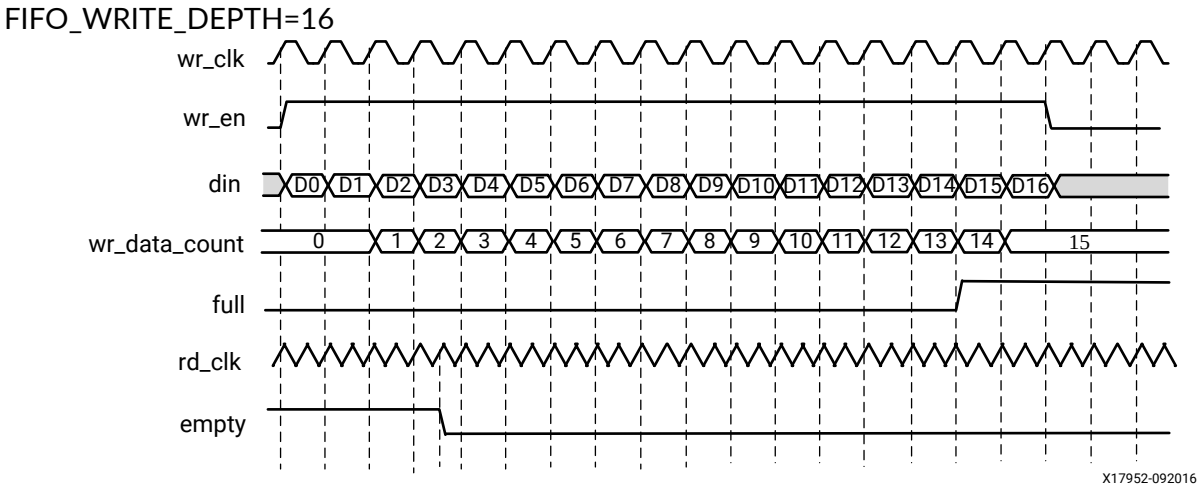
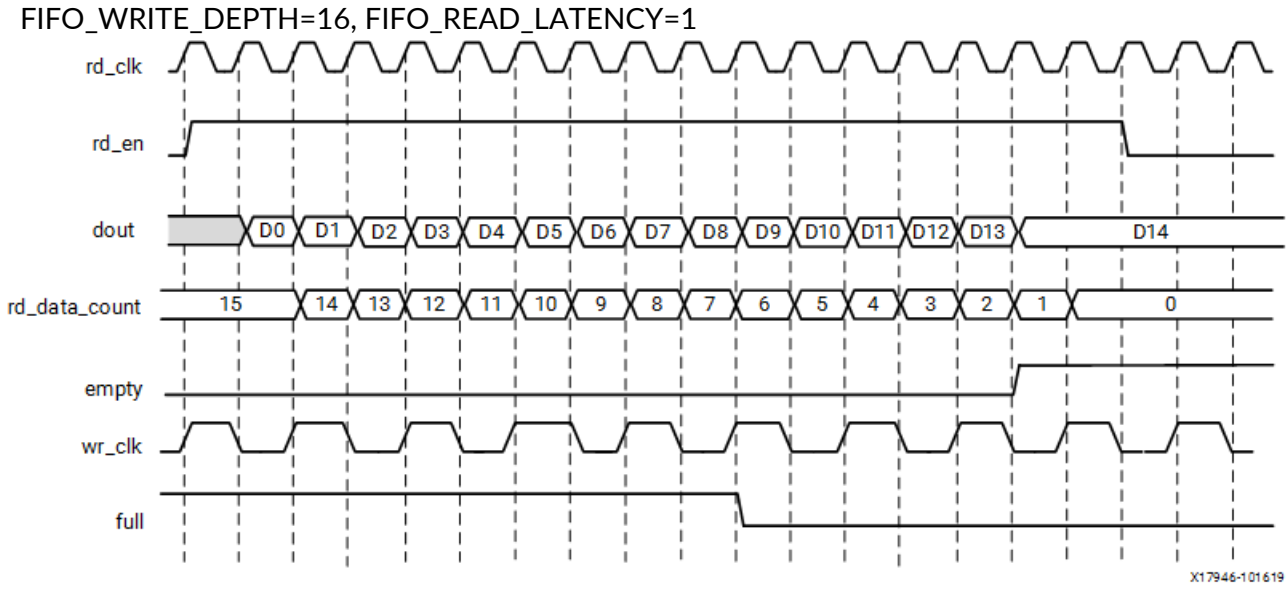


Figure 11: **Standard Read Operation with Full Deassertion**



Latency

This section defines the latency in which different output signals of the FIFO are updated in response to read or write operations for standard read mode and FWFT read mode implementations.

The following table defines the write port flags update latency due to a write operation.

Table 1: **Standard Read Mode — Write Port Flags Due to Write Operation**

Signal	Latency (wr_clk)
full	0
almost_full	0
prog_full	2
wr_ack	1
overflow	0
wr_data_count	2

The following table defines the read port flags update latency due to a read operation.

Table 2: **Standard Read Mode — Read Port Flags Due to Read Operation**

Signal	Latency (rd_clk)
empty	0
almost_empty	0
prog_empty	1

Table 2: **Standard Read Mode — Read Port Flags Due to Read Operation** *(cont'd)*

Signal	Latency (rd_clk)
data_valid	FIFO_READ_LATENCY
underflow	0
rd_data_count	2

The following table defines the write port flags update latency due to a read operation. N is the number of synchronization stages.

Table 3: **Standard Read Mode — Write Port Flags Due to Read Operations**

Signal	Latency
full	1 rd_clk + (N+2) wr_clk
almost_full	1 rd_clk + (N+3) wr_clk
prog_full	1 rd_clk + (N+2) wr_clk
wr_ack	N/A
overflow	N/A
wr_data_count	1 rd_clk + (N+2) wr_clk

The following table defines the read port flags update latency due to a write operation. N is the number of synchronization stages. In this example, N is 2.

Table 4: **Standard Read Mode — Read Port Flags Due to Write Operation**

Signal	Latency
empty	1 wr_clk + (N+2) rd_clk
almost_empty	1 wr_clk + (N+3) rd_clk
prog_empty	1 wr_clk + (N+3) rd_clk
data_valid	N/A
underflow	N/A
rd_data_count	1 wr_clk + (N+2) rd_clk

The following table defines the write port flags update latency due to a write operation.

Table 5: **FWFT Read Mode — Write Port Flags Due to Write Operation**

Signal	Latency
full	2
almost_full	1
prog_full	0
wr_ack	1
overflow	2

Table 5: **FWFT Read Mode — Write Port Flags Due to Write Operation** *(cont'd)*

Signal	Latency
wr_data_count	2

The following table defines the read port flags update latency due to a read operation.

Table 6: **FWFT Read Mode — Read Port Flags Due to Read Operation**

Signal	Latency
empty	2
almost_empty	2
prog_empty	3
data_valid	0
underflow	2
rd_data_count	2

The following table defines the write port flags update latency due to a read operation. N is the number of synchronization stages.

Table 7: **FWFT Read Mode — Write Port Flags Due to Read Operation**

Signal	Latency
full	1 rd_clk + (N+3) wr_clk
almost_full	1 rd_clk + (N+4) wr_clk
prog_full	1 rd_clk + (N+5) wr_clk
wr_ack	N/A
overflow	N/A
wr_data_count	1 rd_clk + (N+3) wr_clk

The following table defines the read port flags update latency due to a write operation. N is the number of synchronization stages. In this example, N is 2.

Table 8: **FWFT Read Mode — Read Port Flags Due to Write Operation**

Signal	Latency
empty	1 wr_clk + (N+4) rd_clk
almost_empty	1 wr_clk + (N+4) rd_clk
prog_empty	1 wr_clk + (N+3) rd_clk
data_valid	1 wr_clk + (N+4) rd_clk
underflow	N/A
rd_data_count	1 wr_clk + (N+4) rd_clk

Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
almost_empty	Output	1	rd_clk	LEVEL _HIGH	DoNotCare	Almost Empty : When asserted, this signal indicates that only one more read can be performed before the FIFO goes to empty.
almost_full	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Almost Full: When asserted, this signal indicates that only one more write can be performed before the FIFO is full.
data_valid	Output	1	rd_clk	LEVEL _HIGH	DoNotCare	Read Data Valid: When asserted, this signal indicates that valid data is available on the output bus (dout).
dbiterr	Output	1	rd_clk	LEVEL _HIGH	DoNotCare	Double Bit Error: Indicates that the ECC decoder detected a double-bit error and data in the FIFO core is corrupted.
din	Input	WRITE _DATA _WIDTH	wr_clk	NA	Active	Write Data: The input data bus used when writing the FIFO.
dout	Output	READ _DATA _WIDTH	rd_clk	NA	Active	Read Data: The output data bus is driven when reading the FIFO.
empty	Output	1	rd_clk	LEVEL _HIGH	Active	Empty Flag: When asserted, this signal indicates that the FIFO is empty. Read requests are ignored when the FIFO is empty, initiating a read while empty is not destructive to the FIFO.
full	Output	1	wr_clk	LEVEL _HIGH	Active	Full Flag: When asserted, this signal indicates that the FIFO is full. Write requests are ignored when the FIFO is full, initiating a write when the FIFO is full is not destructive to the contents of the FIFO.
injectdbiterr	Input	1	wr_clk	LEVEL _HIGH	0	Double Bit Error Injection: Injects a double bit error if the ECC feature is used on block RAMs or UltraRAM macros.
injectsbiterr	Input	1	wr_clk	LEVEL _HIGH	0	Single Bit Error Injection: Injects a single bit error if the ECC feature is used on block RAMs or UltraRAM macros.
overflow	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Overflow: This signal indicates that a write request (wren) during the prior clock cycle was rejected, because the FIFO is full. Overflowing the FIFO is not destructive to the contents of the FIFO.
prog_empty	Output	1	rd_clk	LEVEL _HIGH	DoNotCare	Programmable Empty: This signal is asserted when the number of words in the FIFO is less than or equal to the programmable empty threshold value. It is de-asserted when the number of words in the FIFO exceeds the programmable empty threshold value.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
prog_full	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Programmable Full: This signal is asserted when the number of words in the FIFO is greater than or equal to the programmable full threshold value. It is de-asserted when the number of words in the FIFO is less than the programmable full threshold value.
rd_clk	Input	1	NA	EDGE _RISING	Active	Read clock: Used for read operation. rd_clk must be a free running clock.
rd_data_count	Output	RD _DATA _COUNT _WIDTH	rd_clk	NA	DoNotCare	Read Data Count: This bus indicates the number of words read from the FIFO.
rd_en	Input	1	rd_clk	LEVEL _HIGH	Active	Read Enable: If the FIFO is not empty, asserting this signal causes data (on dout) to be read from the FIFO. • Must be held active-low when rd_rst_busy is active high.
rd_rst_busy	Output	1	rd_clk	LEVEL _HIGH	Active	Read Reset Busy: Active-High indicator that the FIFO read domain is currently in a reset state.
rst	Input	1	wr_clk	LEVEL _HIGH	Active	Reset: Must be synchronous to wr_clk. The clock(s) can be unstable at the time of applying reset, but reset must be released only after the clock(s) is/are stable.
sbiterr	Output	1	rd_clk	LEVEL _HIGH	DoNotCare	Single Bit Error: Indicates that the ECC decoder detected and fixed a single-bit error.
sleep	Input	1	NA	LEVEL _HIGH	0	Dynamic power saving: If sleep is High, the memory/fifo block is in power saving mode.
underflow	Output	1	rd_clk	LEVEL _HIGH	DoNotCare	Underflow: Indicates that the read request (rd_en) during the previous clock cycle was rejected because the FIFO is empty. Under flowing the FIFO is not destructive to the FIFO.
wr_ack	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Write Acknowledge: This signal indicates that a write request (wr_en) during the prior clock cycle is succeeded.
wr_clk	Input	1	NA	EDGE _RISING	Active	Write clock: Used for write operation. wr_clk must be a free running clock.
wr_data_count	Output	WR _DATA _COUNT _WIDTH	wr_clk	NA	DoNotCare	Write Data Count: This bus indicates the number of words written into the FIFO.
wr_en	Input	1	wr_clk	LEVEL _HIGH	Active	Write Enable: If the FIFO is not full, asserting this signal causes data (on din) to be written to the FIFO. • Must be held active-low when rst or wr_rst_busy is active high.
wr_rst_busy	Output	1	wr_clk	LEVEL _HIGH	Active	Write Reset Busy: Active-High indicator that the FIFO write domain is currently in a reset state.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
CDC_SYNC_STAGES	DECIMAL	2 to 8	2	Specifies the number of synchronization stages on the CDC path • Must be < 5 if FIFO_WRITE_DEPTH = 16
DOUT_RESET_VALUE	STRING	String	"0"	Reset value of read data path.
ECC_MODE	STRING	"no_ecc", "en_ecc"	"no_ecc"	• "no_ecc" - Disables ECC • "en_ecc" - Enables both ECC Encoder and Decoder NOTE: ECC_MODE should be "no_ecc" if FIFO_MEMORY_TYPE is set to "auto". Violating this may result incorrect behavior.
FIFO_MEMORY_TYPE	STRING	"auto", "block", "distributed"	"auto"	Designate the fifo memory primitive (resource type) to use. • "auto"- Allow Vivado Synthesis to choose • "block"- Block RAM FIFO • "distributed"- Distributed RAM FIFO NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with FIFO_MEMORY_TYPE set to "auto".
FIFO_READ _LATENCY	DECIMAL	0 to 10	1	Number of output register stages in the read data path. • If READ_MODE = "fwft", then the only applicable value is 0.
FIFO_WRITE_DEPTH	DECIMAL	16 to 4194304	2048	Defines the FIFO Write Depth, must be power of two. • In standard READ_MODE, the effective depth = FIFO_WRITE_DEPTH-1 • In First-Word-Fall-Through READ_MODE, the effective depth = FIFO_WRITE_DEPTH+1 NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
FULL_RESET_VALUE	DECIMAL	0 to 1	0	Sets full, almost_full and prog_full to FULL_RESET_VALUE during reset

Attribute	Type	Allowed Values	Default	Description
PROG_EMPTY _THRESH	DECIMAL	3 to 4194301	10	Specifies the minimum number of read words in the FIFO at or below which prog_empty is asserted. • Min_Value = 3 + (READ_MODE_VAL*2) • Max_Value = (FIFO_WRITE_DEPTH-3) - (READ_MODE_VAL*2) If READ_MODE = "std", then READ_MODE_VAL = 0; Otherwise READ_MODE_VAL = 1. NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.
PROG_FULL_THRESH	DECIMAL	5 to 4194301	10	Specifies the maximum number of write words in the FIFO at or above which prog_full is asserted. • Min_Value = 3 + (READ_MODE_VAL*2*(FIFO_WRITE_DEPTH/ FIFO_READ_DEPTH))+CDC_SYNC_STAGES • Max_Value = (FIFO_WRITE_DEPTH-3) - (READ_MODE_VAL*2*(FIFO_WRITE_DEPTH/ FIFO_READ_DEPTH)) If READ_MODE = "std", then READ_MODE_VAL = 0; Otherwise READ_MODE_VAL = 1. NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.
RD_DATA_COUNT _WIDTH	DECIMAL	1 to 23	1	Specifies the width of rd_data_count. To reflect the correct value, the width should be log2(FIFO_READ_DEPTH)+1. • FIFO_READ_DEPTH = FIFO_WRITE_DEPTH*WRITE_DATA_WIDTH/ READ_DATA_WIDTH
READ_DATA_WIDTH	DECIMAL	1 to 4096	32	Defines the width of the read data port, dout • Write and read width aspect ratio must be 1:1, 1:2, 1:4, 1:8, 8:1, 4:1 and 2:1 • For example, if WRITE_DATA_WIDTH is 32, then the READ_DATA_WIDTH must be 32, 64,128, 256, 16, 8, 4. NOTE: • READ_DATA_WIDTH should be equal to WRITE_DATA_WIDTH if FIFO_MEMORY_TYPE is set to "auto". Violating this may result incorrect behavior. • The maximum FIFO size (width x depth) is limited to 150-Megabits.

Attribute	Type	Allowed Values	Default	Description
READ_MODE	STRING	"std", "fwft"	"std"	• "std"- standard read mode • "fwft"- First-Word-Fall-Through read mode
RELATED_CLOCKS	DECIMAL	0 to 1	0	Specifies if the wr_clk and rd_clk are related having the same source but different clock ratios
SIM_ASSERT_CHK	DECIMAL	0 to 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.
USE_ADV_FEATURES	STRING	String	"0707"	Enables data_valid, almost_empty, rd_data_count, prog_empty, underflow, wr_ack, almost_full, wr_data_count, prog_full, overflow features. • Setting USE_ADV_FEATURES[0] to 1 enables overflow flag; Default value of this bit is 1 • Setting USE_ADV_FEATURES[1] to 1 enables prog_full flag; Default value of this bit is 1 • Setting USE_ADV_FEATURES[2] to 1 enables wr_data_count; Default value of this bit is 1 • Setting USE_ADV_FEATURES[3] to 1 enables almost_full flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES[4] to 1 enables wr_ack flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES[8] to 1 enables underflow flag; Default value of this bit is 1 • Setting USE_ADV_FEATURES[9] to 1 enables prog_empty flag; Default value of this bit is 1 • Setting USE_ADV_FEATURES[10] to 1 enables rd_data_count; Default value of this bit is 1 • Setting USE_ADV_FEATURES[11] to 1 enables almost_empty flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES[12] to 1 enables data_valid flag; Default value of this bit is 0
WAKEUP_TIME	DECIMAL	0 to 2	0	• 0 - Disable sleep • 2 - Use Sleep Pin NOTE: WAKEUP_TIME should be 0 if FIFO_MEMORY_TYPE is set to "auto". Violating this may result incorrect behavior.
WR_DATA_COUNT _WIDTH	DECIMAL	1 to 23	1	Specifies the width of wr_data_count. To reflect the correct value, the width should be log2(FIFO_WRITE_DEPTH)+1.

Attribute	Type	Allowed Values	Default	Description
WRITE_DATA_WIDTH	DECIMAL	1 to 4096	32	Defines the width of the write data port, din • Write and read width aspect ratio must be 1:1, 1:2, 1:4, 1:8, 8:1, 4:1 and 2:1 • For example, if WRITE_DATA_WIDTH is 32, then the READ_DATA_WIDTH must be 32, 64,128, 256, 16, 8, 4. NOTE: • WRITE_DATA_WIDTH should be equal to READ_DATA_WIDTH if FIFO_MEMORY_TYPE is set to "auto". Violating this may result incorrect behavior. • The maximum FIFO size (width x depth) is limited to 150-Megabits.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_fifo_async: Asynchronous FIFO
-- Xilinx Parameterized Macro, version 2020.1

xpm_fifo_async_inst : xpm_fifo_async
generic map (
   CDC_SYNC_STAGES => 2,       -- DECIMAL
   DOUT_RESET_VALUE => "0",    -- String
   ECC_MODE => "no_ecc",       -- String
   FIFO_MEMORY_TYPE => "auto", -- String
   FIFO_READ_LATENCY => 1,     -- DECIMAL
   FIFO_WRITE_DEPTH => 2048,   -- DECIMAL
   FULL_RESET_VALUE => 0,      -- DECIMAL
   PROG_EMPTY_THRESH => 10,    -- DECIMAL
   PROG_FULL_THRESH => 10,     -- DECIMAL
   RD_DATA_COUNT_WIDTH => 1,   -- DECIMAL
   READ_DATA_WIDTH => 32,      -- DECIMAL
   READ_MODE => "std",         -- String
   RELATED_CLOCKS => 0,        -- DECIMAL
   SIM_ASSERT_CHK => 0,        -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   USE_ADV_FEATURES => "0707", -- String
   WAKEUP_TIME => 0,           -- DECIMAL
   WRITE_DATA_WIDTH => 32,     -- DECIMAL
   WR_DATA_COUNT_WIDTH => 1    -- DECIMAL
)
port map (
   almost_empty => almost_empty,   -- 1-bit output: Almost Empty : When asserted, this signal indicates that
                                   -- only one more read can be performed before the FIFO goes to empty.

   almost_full => almost_full,     -- 1-bit output: Almost Full: When asserted, this signal indicates that
                                        -- only one more write can be performed before the FIFO is full.

   data_valid => data_valid,       -- 1-bit output: Read Data Valid: When asserted, this signal indicates
                                        -- that valid data is available on the output bus (dout).

   dbiterr => dbiterr,                  -- 1-bit output: Double Bit Error: Indicates that the ECC decoder
                                   -- detected a double-bit error and data in the FIFO core is corrupted.
```

```
dout => dout,                     -- READ_DATA_WIDTH-bit output: Read Data: The output data bus is driven
                                     -- when reading the FIFO.

empty => empty,                   -- 1-bit output: Empty Flag: When asserted, this signal indicates that
                                  -- the FIFO is empty. Read requests are ignored when the FIFO is empty,
                                     -- initiating a read while empty is not destructive to the FIFO.

full => full,                     -- 1-bit output: Full Flag: When asserted, this signal indicates that the
                                  -- FIFO is full. Write requests are ignored when the FIFO is full,
                                  -- initiating a write when the FIFO is full is not destructive to the
                                     -- contents of the FIFO.

overflow => overflow,             -- 1-bit output: Overflow: This signal indicates that a write request
                                  -- (wren) during the prior clock cycle was rejected, because the FIFO is
                                  -- full. Overflowing the FIFO is not destructive to the contents of the
                                     -- FIFO.

prog_empty => prog_empty,         -- 1-bit output: Programmable Empty: This signal is asserted when the
                                  -- number of words in the FIFO is less than or equal to the programmable
                                  -- empty threshold value. It is de-asserted when the number of words in
                                     -- the FIFO exceeds the programmable empty threshold value.

prog_full => prog_full,           -- 1-bit output: Programmable Full: This signal is asserted when the
                                     -- number of words in the FIFO is greater than or equal to the
                                  -- programmable full threshold value. It is de-asserted when the number
                                  -- of words in the FIFO is less than the programmable full threshold
                                     -- value.

rd_data_count => rd_data_count,   -- RD_DATA_COUNT_WIDTH-bit output: Read Data Count: This bus indicates
                                     -- the number of words read from the FIFO.

rd_rst_busy => rd_rst_busy,       -- 1-bit output: Read Reset Busy: Active-High indicator that the FIFO
                                     -- read domain is currently in a reset state.

 sbiterr => sbiterr,                 -- 1-bit output: Single Bit Error: Indicates that the ECC decoder
                                     -- detected and fixed a single-bit error.

underflow => underflow,           -- 1-bit output: Underflow: Indicates that the read request (rd_en)
                                  -- during the previous clock cycle was rejected because the FIFO is
                                     -- empty. Under flowing the FIFO is not destructive to the FIFO.

wr_ack => wr_ack,                 -- 1-bit output: Write Acknowledge: This signal indicates that a write
                                     -- request (wr_en) during the prior clock cycle is succeeded.

wr_data_count => wr_data_count,   -- WR_DATA_COUNT_WIDTH-bit output: Write Data Count: This bus indicates
                                     -- the number of words written into the FIFO.

wr_rst_busy => wr_rst_busy,       -- 1-bit output: Write Reset Busy: Active-High indicator that the FIFO
                                     -- write domain is currently in a reset state.

din => din,                       -- WRITE_DATA_WIDTH-bit input: Write Data: The input data bus used when
                                     -- writing the FIFO.

injectdbiterr => injectdbiterr,   -- 1-bit input: Double Bit Error Injection: Injects a double bit error if
                                     -- the ECC feature is used on block RAMs or UltraRAM macros.

injectsbiterr => injectsbiterr,   -- 1-bit input: Single Bit Error Injection: Injects a single bit error if
                                     -- the ECC feature is used on block RAMs or UltraRAM macros.

rd_clk => rd_clk,                 -- 1-bit input: Read clock: Used for read operation. rd_clk must be a
                                     -- free running clock.

rd_en => rd_en,                   -- 1-bit input: Read Enable: If the FIFO is not empty, asserting this
                                  -- signal causes data (on dout) to be read from the FIFO. Must be held
                                     -- active-low when rd_rst_busy is active high.

rst => rst,                       -- 1-bit input: Reset: Must be synchronous to wr_clk. The clock(s) can be
                                  -- unstable at the time of applying reset, but reset must be released
                                     -- only after the clock(s) is/are stable.

sleep => sleep,                   -- 1-bit input: Dynamic power saving: If sleep is High, the memory/fifo
                                     -- block is in power saving mode.

wr_clk => wr_clk,                 -- 1-bit input: Write clock: Used for write operation. wr_clk must be a
                                     -- free running clock.

wr_en => wr_en                    -- 1-bit input: Write Enable: If the FIFO is not full, asserting this
                                  -- signal causes data (on din) to be written to the FIFO. Must be held
```

```
                                    -- active-low when rst or wr_rst_busy is active high.

);

-- End of xpm_fifo_async_inst instantiation
```

Verilog Instantiation Template

```
// xpm_fifo_async: Asynchronous FIFO
// Xilinx Parameterized Macro, version 2020.1

xpm_fifo_async #(
   .CDC_SYNC_STAGES(2),        // DECIMAL
   .DOUT_RESET_VALUE("0"),     // String
   .ECC_MODE("no_ecc"),        // String
   .FIFO_MEMORY_TYPE("auto"), // String
   .FIFO_READ_LATENCY(1),      // DECIMAL
   .FIFO_WRITE_DEPTH(2048),    // DECIMAL
   .FULL_RESET_VALUE(0),       // DECIMAL
   .PROG_EMPTY_THRESH(10),     // DECIMAL
   .PROG_FULL_THRESH(10),      // DECIMAL
   .RD_DATA_COUNT_WIDTH(1),    // DECIMAL
   .READ_DATA_WIDTH(32),       // DECIMAL
   .READ_MODE("std"),          // String
   .RELATED_CLOCKS(0),         // DECIMAL
   .SIM_ASSERT_CHK(0),        // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .USE_ADV_FEATURES("0707"), // String
   .WAKEUP_TIME(0),            // DECIMAL
   .WRITE_DATA_WIDTH(32),      // DECIMAL
   .WR_DATA_COUNT_WIDTH(1)     // DECIMAL
)
xpm_fifo_async_inst (
   .almost_empty(almost_empty),   // 1-bit output: Almost Empty : When asserted, this signal indicates that
                                  // only one more read can be performed before the FIFO goes to empty.

   .almost_full(almost_full),     // 1-bit output: Almost Full: When asserted, this signal indicates that
                                     // only one more write can be performed before the FIFO is full.

   .data_valid(data_valid),       // 1-bit output: Read Data Valid: When asserted, this signal indicates
                                     // that valid data is available on the output bus (dout).

   .dbiterr(dbiterr),             // 1-bit output: Double Bit Error: Indicates that the ECC decoder detected
                                     // a double-bit error and data in the FIFO core is corrupted.

   .dout(dout),                   // READ_DATA_WIDTH-bit output: Read Data: The output data bus is driven
                                     // when reading the FIFO.

   .empty(empty),                 // 1-bit output: Empty Flag: When asserted, this signal indicates that the
                                  // FIFO is empty. Read requests are ignored when the FIFO is empty,
                                     // initiating a read while empty is not destructive to the FIFO.

   .full(full),                   // 1-bit output: Full Flag: When asserted, this signal indicates that the
                                     // FIFO is full. Write requests are ignored when the FIFO is full,
                                  // initiating a write when the FIFO is full is not destructive to the
                                     // contents of the FIFO.

   .overflow(overflow),           // 1-bit output: Overflow: This signal indicates that a write request
                                  // (wren) during the prior clock cycle was rejected, because the FIFO is
                                  // full. Overflowing the FIFO is not destructive to the contents of the
                                     // FIFO.

   .prog_empty(prog_empty),       // 1-bit output: Programmable Empty: This signal is asserted when the
                                  // number of words in the FIFO is less than or equal to the programmable
                                  // empty threshold value. It is de-asserted when the number of words in
                                     // the FIFO exceeds the programmable empty threshold value.

   .prog_full(prog_full),         // 1-bit output: Programmable Full: This signal is asserted when the
                                     // number of words in the FIFO is greater than or equal to the
                                  // programmable full threshold value. It is de-asserted when the number of
                                  // words in the FIFO is less than the programmable full threshold value.

   .rd_data_count(rd_data_count), // RD_DATA_COUNT_WIDTH-bit output: Read Data Count: This bus indicates the
                                     // number of words read from the FIFO.

   .rd_rst_busy(rd_rst_busy),     // 1-bit output: Read Reset Busy: Active-High indicator that the FIFO read
                                     // domain is currently in a reset state.
```

```
   .sbiterr(sbiterr),             // 1-bit output: Single Bit Error: Indicates that the ECC decoder detected
                                     // and fixed a single-bit error.

   .underflow(underflow),         // 1-bit output: Underflow: Indicates that the read request (rd_en) during
                                  // the previous clock cycle was rejected because the FIFO is empty. Under
                                     // flowing the FIFO is not destructive to the FIFO.

   .wr_ack(wr_ack),               // 1-bit output: Write Acknowledge: This signal indicates that a write
                                     // request (wr_en) during the prior clock cycle is succeeded.

   .wr_data_count(wr_data_count), // WR_DATA_COUNT_WIDTH-bit output: Write Data Count: This bus indicates
                                     // the number of words written into the FIFO.

   .wr_rst_busy(wr_rst_busy),     // 1-bit output: Write Reset Busy: Active-High indicator that the FIFO
                                     // write domain is currently in a reset state.

   .din(din),                     // WRITE_DATA_WIDTH-bit input: Write Data: The input data bus used when
                                     // writing the FIFO.

   .injectdbiterr(injectdbiterr), // 1-bit input: Double Bit Error Injection: Injects a double bit error if
                                     // the ECC feature is used on block RAMs or UltraRAM macros.

   .injectsbiterr(injectsbiterr), // 1-bit input: Single Bit Error Injection: Injects a single bit error if
                                     // the ECC feature is used on block RAMs or UltraRAM macros.

   .rd_clk(rd_clk),               // 1-bit input: Read clock: Used for read operation. rd_clk must be a free
                                     // running clock.

   .rd_en(rd_en),                 // 1-bit input: Read Enable: If the FIFO is not empty, asserting this
                                  // signal causes data (on dout) to be read from the FIFO. Must be held
                                     // active-low when rd_rst_busy is active high.

   .rst(rst),                     // 1-bit input: Reset: Must be synchronous to wr_clk. The clock(s) can be
                                  // unstable at the time of applying reset, but reset must be released only
                                     // after the clock(s) is/are stable.

   .sleep(sleep),                 // 1-bit input: Dynamic power saving: If sleep is High, the memory/fifo
                                     // block is in power saving mode.

   .wr_clk(wr_clk),               // 1-bit input: Write clock: Used for write operation. wr_clk must be a
                                     // free running clock.

   .wr_en(wr_en)                  // 1-bit input: Write Enable: If the FIFO is not full, asserting this
                                  // signal causes data (on din) to be written to the FIFO. Must be held
                                     // active-low when rst or wr_rst_busy is active high.

);

// End of xpm_fifo_async_inst instantiation
```

For More Information

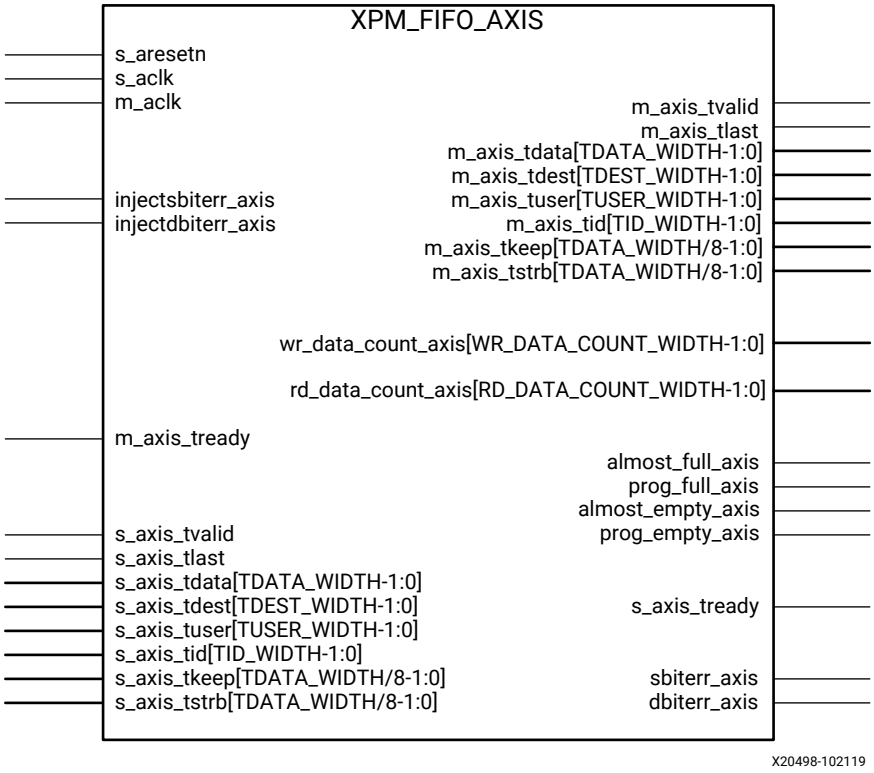
- XPM FIFO Testbench File

XPM_FIFO_AXIS

Parameterized Macro: AXI Stream FIFO

MACRO_GROUP: XPM

MACRO_SUBGROUP: XPM_FIFO



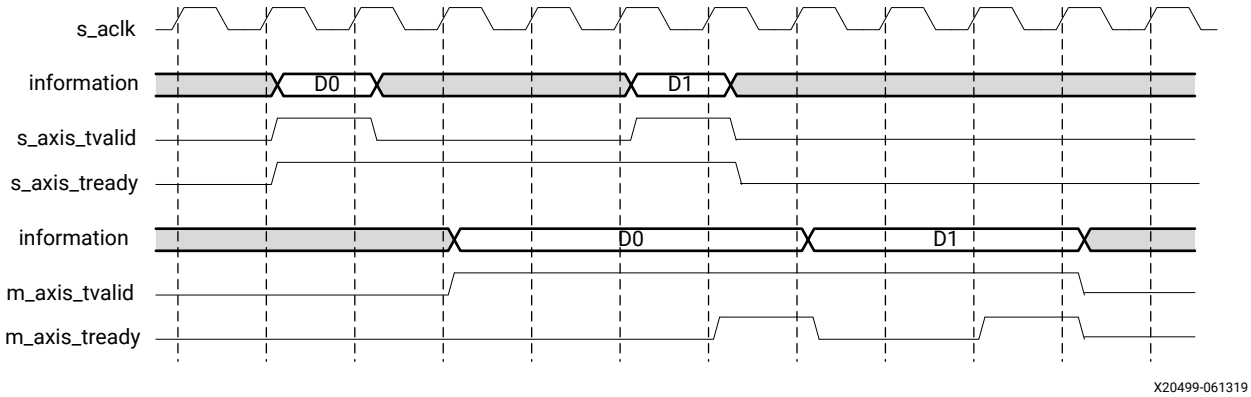
Introduction

This macro is used to instantiate AXI Stream FIFO.

AXI Stream FIFO is derived from the XPM_FIFO_SYNC and XPM_FIFO_ASYNC. The AXI Stream protocol uses a two-way valid and ready handshake mechanism. The information source uses the valid signal to show when valid data or control information is available on the channel. The information destination uses the ready signal to show when it can accept the data.

Timing Diagrams

Figure 12: **Timing for Read and Write Operations to the AXI Stream FIFO**



In the timing diagram above, the information source generates a valid signal to indicate when data is available. The destination generates a ready signal to indicate that it can accept data, and transfer occurs only when both the valid and ready signals are High.

Because the AXI Stream FIFO is derived from XPM_FIFO_SYNC and XPM_FIFO_ASYNC, much of the behavior is common between them. The ready signal is generated based on availability of space in the FIFO and is held high to allow writes to the FIFO. The ready signal is pulled Low only when there is no space in the FIFO left to perform additional writes. The valid signal is generated based on availability of data in the FIFO and is held High to allow reads to be performed from the FIFO. The valid signal is pulled Low only when there is no data available to be read from the FIFO. The information signals are mapped to the din and dout bus of Native interface FIFOs. The width of the AXI FIFO is determined by concatenating all of the information signals of the AXI interface. The information signals include all AXI signals except for the valid and ready handshake signals.

The AXI Stream FIFO operates only in First-Word Fall-Through mode. The First-Word Fall-Through (FWFT) feature provides the ability to look ahead to the next word available from the FIFO without issuing a read operation. When data is available in the FIFO, the first word falls through the FIFO and appears automatically on the output data bus.

Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
almost_empty_axis	Output	1	m_aclk	LEVEL_HIGH	DoNotCare	Almost Empty : When asserted, this signal indicates that only one more read can be performed before the FIFO goes to empty.
almost_full_axis	Output	1	s_aclk	LEVEL_HIGH	DoNotCare	Almost Full: When asserted, this signal indicates that only one more write can be performed before the FIFO is full.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
dbiterr_axis	Output	1	m_aclk	LEVEL _HIGH	DoNotCare	Double Bit Error- Indicates that the ECC decoder detected a double-bit error and data in the FIFO core is corrupted.
injectdbiterr _axis	Input	1	s_aclk	LEVEL _HIGH	0	Double Bit Error Injection- Injects a double bit error if the ECC feature is used.
injectsbiterr _axis	Input	1	s_aclk	LEVEL _HIGH	0	Single Bit Error Injection- Injects a single bit error if the ECC feature is used.
m_aclk	Input	1	NA	EDGE _RISING	Active	Master Interface Clock: All signals on master interface are sampled on the rising edge of this clock.
m_axis_tdata	Output	TDATA _WIDTH	m_aclk	NA	Active	TDATA: The primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
m_axis_tdest	Output	TDEST _WIDTH	m_aclk	NA	Active	TDEST: Provides routing information for the data stream.
m_axis_tid	Output	TID _WIDTH	m_aclk	NA	Active	TID: The data stream identifier that indicates different streams of data.
m_axis_tkeep	Output	TDATA _WIDTH / 8	m_aclk	NA	Active	TKEEP: The byte qualifier that indicates whether the content of the associated byte of TDATA is processed as part of the data stream. Associated bytes that have the TKEEP byte qualifier deasserted are null bytes and can be removed from the data stream. For a 64-bit DATA, bit 0 corresponds to the least significant byte on DATA, and bit 7 corresponds to the most significant byte. For example: • KEEP[0] = 1b, DATA[7:0] is not a NULL byte • KEEP[7] = 0b, DATA[63:56] is a NULL byte
m_axis_tlast	Output	1	m_aclk	LEVEL _HIGH	Active	TLAST: Indicates the boundary of a packet.
m_axis_tready	Input	1	m_aclk	LEVEL _HIGH	Active	TREADY: Indicates that the slave can accept a transfer in the current cycle.
m_axis_tstrb	Output	TDATA _WIDTH / 8	m_aclk	NA	Active	TSTRB: The byte qualifier that indicates whether the content of the associated byte of TDATA is processed as a data byte or a position byte. For a 64-bit DATA, bit 0 corresponds to the least significant byte on DATA, and bit 0 corresponds to the least significant byte on DATA, and bit 7 corresponds to the most significant byte. For example: • STROBE[0] = 1b, DATA[7:0] is valid • STROBE[7] = 0b, DATA[63:56] is not valid
m_axis_tuser	Output	TUSER _WIDTH	m_aclk	NA	Active	TUSER: The user-defined sideband information that can be transmitted alongside the data stream.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
m_axis_tvalid	Output	1	m_aclk	LEVEL _HIGH	Active	TVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both TVALID and TREADY are asserted
prog_empty_axis	Output	1	m_aclk	LEVEL _HIGH	DoNotCare	Programmable Empty- This signal is asserted when the number of words in the FIFO is less than or equal to the programmable empty threshold value. It is de-asserted when the number of words in the FIFO exceeds the programmable empty threshold value.
prog_full_axis	Output	1	s_aclk	LEVEL _HIGH	DoNotCare	Programmable Full: This signal is asserted when the number of words in the FIFO is greater than or equal to the programmable full threshold value. It is de-asserted when the number of words in the FIFO is less than the programmable full threshold value.
rd_data_count _axis	Output	RD _DATA _COUNT _WIDTH	m_aclk	NA	DoNotCare	Read Data Count- This bus indicates the number of words available for reading in the FIFO.
s_aclk	Input	1	NA	EDGE _RISING	Active	Slave Interface Clock: All signals on slave interface are sampled on the rising edge of this clock.
s_aresetn	Input	1	NA	LEVEL _LOW	Active	Active low asynchronous reset.
s_axis_tdata	Input	TDATA _WIDTH	s_aclk	NA	Active	TDATA: The primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
s_axis_tdest	Input	TDEST _WIDTH	s_aclk	NA	Active	TDEST: Provides routing information for the data stream.
s_axis_tid	Input	TID _WIDTH	s_aclk	NA	Active	TID: The data stream identifier that indicates different streams of data.
s_axis_tkeep	Input	TDATA _WIDTH / 8	s_aclk	NA	Active	TKEEP: The byte qualifier that indicates whether the content of the associated byte of TDATA is processed as part of the data stream. Associated bytes that have the TKEEP byte qualifier deasserted are null bytes and can be removed from the data stream. For a 64-bit DATA, bit 0 corresponds to the least significant byte on DATA, and bit 7 corresponds to the most significant byte. For example: • KEEP[0] = 1b, DATA[7:0] is not a NULL byte • KEEP[7] = 0b, DATA[63:56] is a NULL byte
s_axis_tlast	Input	1	s_aclk	LEVEL _HIGH	Active	TLAST: Indicates the boundary of a packet.
s_axis_tready	Output	1	s_aclk	LEVEL _HIGH	Active	TREADY: Indicates that the slave can accept a transfer in the current cycle.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
s_axis_tstrb	Input	TDATA _WIDTH / 8	s_aclk	NA	Active	TSTRB: The byte qualifier that indicates whether the content of the associated byte of TDATA is processed as a data byte or a position byte. For a 64-bit DATA, bit 0 corresponds to the least significant byte on DATA, and bit 0 corresponds to the least significant byte on DATA, and bit 7 corresponds to the most significant byte. For example: • STROBE[0] = 1b, DATA[7:0] is valid • STROBE[7] = 0b, DATA[63:56] is not valid
s_axis_tuser	Input	TUSER _WIDTH	s_aclk	NA	Active	TUSER: The user-defined sideband information that can be transmitted alongside the data stream.
s_axis_tvalid	Input	1	s_aclk	LEVEL _HIGH	Active	TVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both TVALID and TREADY are asserted
sbiterr_axis	Output	1	m_aclk	LEVEL _HIGH	DoNotCare	Single Bit Error- Indicates that the ECC decoder detected and fixed a single-bit error.
wr_data_count _axis	Output	WR _DATA _COUNT _WIDTH	s_aclk	NA	DoNotCare	Write Data Count: This bus indicates the number of words written into the FIFO.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
CDC_SYNC_STAGES	DECIMAL	2 to 8	2	Specifies the number of synchronization stages on the CDC path. Applicable only if CLOCKING_MODE = "independent_clock"

Attribute	Type	Allowed Values	Default	Description
CLOCKING_MODE	STRING	"common _clock", "independent _clock"	"common _clock"	Designate whether AXI Stream FIFO is clocked with a common clock or with independent clocks- • "common_clock"- Common clocking; clock both write and read domain s_aclk • "independent_clock"- Independent clocking; clock write domain with s_aclk and read domain with m_aclk
ECC_MODE	STRING	"no_ecc", "en_ecc"	"no_ecc"	• "no_ecc" - Disables ECC • "en_ecc" - Enables both ECC Encoder and Decoder NOTE: ECC_MODE should be "no_ecc" if FIFO_MEMORY_TYPE is set to "auto". Violating this may result incorrect behavior.
FIFO_DEPTH	DECIMAL	16 to 4194304	2048	Defines the AXI Stream FIFO Write Depth, must be power of two NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
FIFO_MEMORY_TYPE	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the fifo memory primitive (resource type) to use- • "auto"- Allow Vivado Synthesis to choose • "block"- Block RAM FIFO • "distributed"- Distributed RAM FIFO • "ultra"- URAM FIFO NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with FIFO_MEMORY_TYPE set to "auto".
PACKET_FIFO	STRING	"false", "true"	"false"	• "true"- Enables Packet FIFO mode • "false"- Disables Packet FIFO mode
PROG_EMPTY _THRESH	DECIMAL	5 to 4194301	10	Specifies the minimum number of read words in the FIFO at or below which prog_empty is asserted. • Min_Value = 5 • Max_Value = FIFO_WRITE_DEPTH - 5 NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.

Attribute	Type	Allowed Values	Default	Description
PROG_FULL_THRESH	DECIMAL	5 to 4194301	10	Specifies the maximum number of write words in the FIFO at or above which prog_full is asserted. • Min_Value = 5 + CDC_SYNC_STAGES • Max_Value = FIFO_WRITE_DEPTH - 5 NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.
RD_DATA_COUNT _WIDTH	DECIMAL	1 to 23	1	Specifies the width of rd_data_count_axis. To reflect the correct value, the width should be log2(FIFO_DEPTH)+1.
RELATED_CLOCKS	DECIMAL	0 to 1	0	Specifies if the s_aclk and m_aclk are related having the same source but different clock ratios. Applicable only if CLOCKING_MODE = "independent_clock"
SIM_ASSERT_CHK	DECIMAL	0 to 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.
TDATA_WIDTH	DECIMAL	8 to 2048	32	Defines the width of the TDATA port, s_axis_tdata and m_axis_tdata NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
TDEST_WIDTH	DECIMAL	1 to 32	1	Defines the width of the TDEST port, s_axis_tdest and m_axis_tdest
TID_WIDTH	DECIMAL	1 to 32	1	Defines the width of the ID port, s_axis_tid and m_axis_tid
TUSER_WIDTH	DECIMAL	1 to 4086	1	Defines the width of the TUSER port, s_axis_tuser and m_axis_tuser
USE_ADV_FEATURES	STRING	String	"1000"	Enables almost_empty_axis, rd_data_count_axis, prog_empty_axis, almost_full_axis, wr_data_count_axis, prog_full_axis sideband signals. • Setting USE_ADV_FEATURES[1] to 1 enables prog_full flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES[2] to 1 enables wr_data_count; Default value of this bit is 0 • Setting USE_ADV_FEATURES[3] to 1 enables almost_full flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES[9] to 1 enables prog_empty flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES[10] to 1 enables rd_data_count; Default value of this bit is 0 • Setting USE_ADV_FEATURES[11] to 1 enables almost_empty flag; Default value of this bit is 0

Attribute	Type	Allowed Values	Default	Description
WR_DATA_COUNT _WIDTH	DECIMAL	1 to 23	1	Specifies the width of wr_data_count_axis. To reflect the correct value, the width should be log2(FIFO_DEPTH)+1.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_fifo_axis: AXI Stream FIFO
-- Xilinx Parameterized Macro, version 2020.1

xpm_fifo_axis_inst : xpm_fifo_axis
generic map (
   CDC_SYNC_STAGES => 2,             -- DECIMAL
   CLOCKING_MODE => "common_clock", -- String
   ECC_MODE => "no_ecc",             -- String
   FIFO_DEPTH => 2048,               -- DECIMAL
   FIFO_MEMORY_TYPE => "auto",       -- String
   PACKET_FIFO => "false",           -- String
   PROG_EMPTY_THRESH => 10,          -- DECIMAL
   PROG_FULL_THRESH => 10,           -- DECIMAL
   RD_DATA_COUNT_WIDTH => 1,         -- DECIMAL
   RELATED_CLOCKS => 0,              -- DECIMAL
   SIM_ASSERT_CHK => 0,            -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   TDATA_WIDTH => 32,                -- DECIMAL
   TDEST_WIDTH => 1,                 -- DECIMAL
   TID_WIDTH => 1,                   -- DECIMAL
   TUSER_WIDTH => 1,                 -- DECIMAL
   USE_ADV_FEATURES => "1000",       -- String
   WR_DATA_COUNT_WIDTH => 1          -- DECIMAL
)
port map (
   almost_empty_axis => almost_empty_axis,    -- 1-bit output: Almost Empty : When asserted, this signal
                                              -- indicates that only one more read can be performed before
                                                    -- the FIFO goes to empty.

   almost_full_axis => almost_full_axis,      -- 1-bit output: Almost Full: When asserted, this signal
                                              -- indicates that only one more write can be performed before
                                                    -- the FIFO is full.

   dbiterr_axis => dbiterr_axis,              -- 1-bit output: Double Bit Error- Indicates that the ECC
                                              -- decoder detected a double-bit error and data in the FIFO
                                                    -- core is corrupted.

   m_axis_tdata => m_axis_tdata,              -- TDATA_WIDTH-bit output: TDATA: The primary payload that is
                                                    -- used to provide the data that is passing across the
                                              -- interface. The width of the data payload is an integer
                                                    -- number of bytes.

   m_axis_tdest => m_axis_tdest,              -- TDEST_WIDTH-bit output: TDEST: Provides routing information
                                                    -- for the data stream.

   m_axis_tid => m_axis_tid,                  -- TID_WIDTH-bit output: TID: The data stream identifier that
                                                    -- indicates different streams of data.

   m_axis_tkeep => m_axis_tkeep,              -- TDATA_WIDTH/8-bit output: TKEEP: The byte qualifier that
                                              -- indicates whether the content of the associated byte of
                                              -- TDATA is processed as part of the data stream. Associated
                                              -- bytes that have the TKEEP byte qualifier deasserted are null
                                              -- bytes and can be removed from the data stream. For a 64-bit
                                              -- DATA, bit 0 corresponds to the least significant byte on
                                              -- DATA, and bit 7 corresponds to the most significant byte.
                                              -- For example: KEEP[0] = 1b, DATA[7:0] is not a NULL byte
                                                    -- KEEP[7] = 0b, DATA[63:56] is a NULL byte
```

```
m_axis_tlast => m_axis_tlast,             -- 1-bit output: TLAST: Indicates the boundary of a packet.
m_axis_tstrb => m_axis_tstrb,             -- TDATA_WIDTH/8-bit output: TSTRB: The byte qualifier that
                                          -- indicates whether the content of the associated byte of
                                          -- TDATA is processed as a data byte or a position byte. For a
                                          -- 64-bit DATA, bit 0 corresponds to the least significant byte
                                          -- on DATA, and bit 0 corresponds to the least significant byte
                                          -- on DATA, and bit 7 corresponds to the most significant byte.
                                          -- For example: STROBE[0] = 1b, DATA[7:0] is valid STROBE[7] =
                                              -- 0b, DATA[63:56] is not valid

m_axis_tuser => m_axis_tuser,             -- TUSER_WIDTH-bit output: TUSER: The user-defined sideband
                                          -- information that can be transmitted alongside the data
                                              -- stream.

m_axis_tvalid => m_axis_tvalid,           -- 1-bit output: TVALID: Indicates that the master is driving a
                                          -- valid transfer. A transfer takes place when both TVALID and
                                              -- TREADY are asserted

prog_empty_axis => prog_empty_axis,       -- 1-bit output: Programmable Empty- This signal is asserted
                                          -- when the number of words in the FIFO is less than or equal
                                          -- to the programmable empty threshold value. It is de-asserted
                                              -- when the number of words in the FIFO exceeds the
                                              -- programmable empty threshold value.

prog_full_axis => prog_full_axis,         -- 1-bit output: Programmable Full: This signal is asserted
                                          -- when the number of words in the FIFO is greater than or
                                          -- equal to the programmable full threshold value. It is
                                          -- de-asserted when the number of words in the FIFO is less
                                              -- than the programmable full threshold value.

rd_data_count_axis => rd_data_count_axis, -- RD_DATA_COUNT_WIDTH-bit output: Read Data Count- This bus
                                          -- indicates the number of words available for reading in the
                                              -- FIFO.

s_axis_tready => s_axis_tready,           -- 1-bit output: TREADY: Indicates that the slave can accept a
                                              -- transfer in the current cycle.

sbiterr_axis => sbiterr_axis,             -- 1-bit output: Single Bit Error- Indicates that the ECC
                                              -- decoder detected and fixed a single-bit error.

wr_data_count_axis => wr_data_count_axis, -- WR_DATA_COUNT_WIDTH-bit output: Write Data Count: This bus
                                              -- indicates the number of words written into the FIFO.

injectdbiterr_axis => injectdbiterr_axis, -- 1-bit input: Double Bit Error Injection- Injects a double
                                              -- bit error if the ECC feature is used.

injectsbiterr_axis => injectsbiterr_axis, -- 1-bit input: Single Bit Error Injection- Injects a single
                                              -- bit error if the ECC feature is used.

m_aclk => m_aclk,                         -- 1-bit input: Master Interface Clock: All signals on master
                                          -- interface are sampled on the rising edge of this clock.

m_axis_tready => m_axis_tready,           -- 1-bit input: TREADY: Indicates that the slave can accept a
                                              -- transfer in the current cycle.

s_aclk => s_aclk,                         -- 1-bit input: Slave Interface Clock: All signals on slave
                                          -- interface are sampled on the rising edge of this clock.

 s_aresetn => s_aresetn,                      -- 1-bit input: Active low asynchronous reset.
s_axis_tdata => s_axis_tdata,             -- TDATA_WIDTH-bit input: TDATA: The primary payload that is
                                              -- used to provide the data that is passing across the
                                          -- interface. The width of the data payload is an integer
                                              -- number of bytes.

s_axis_tdest => s_axis_tdest,             -- TDEST_WIDTH-bit input: TDEST: Provides routing information
                                              -- for the data stream.

s_axis_tid => s_axis_tid,                 -- TID_WIDTH-bit input: TID: The data stream identifier that
                                              -- indicates different streams of data.

s_axis_tkeep => s_axis_tkeep,             -- TDATA_WIDTH/8-bit input: TKEEP: The byte qualifier that
                                          -- indicates whether the content of the associated byte of
                                          -- TDATA is processed as part of the data stream. Associated
                                          -- bytes that have the TKEEP byte qualifier deasserted are null
                                          -- bytes and can be removed from the data stream. For a 64-bit
                                          -- DATA, bit 0 corresponds to the least significant byte on
                                          -- DATA, and bit 7 corresponds to the most significant byte.
                                          -- For example: KEEP[0] = 1b, DATA[7:0] is not a NULL byte
```

```
                                          -- KEEP[7] = 0b, DATA[63:56] is a NULL byte

   s_axis_tlast => s_axis_tlast,          -- 1-bit input: TLAST: Indicates the boundary of a packet.
   s_axis_tstrb => s_axis_tstrb,          -- TDATA_WIDTH/8-bit input: TSTRB: The byte qualifier that
                                          -- indicates whether the content of the associated byte of
                                          -- TDATA is processed as a data byte or a position byte. For a
                                          -- 64-bit DATA, bit 0 corresponds to the least significant byte
                                          -- on DATA, and bit 0 corresponds to the least significant byte
                                          -- on DATA, and bit 7 corresponds to the most significant byte.
                                          -- For example: STROBE[0] = 1b, DATA[7:0] is valid STROBE[7] =
                                                -- 0b, DATA[63:56] is not valid

   s_axis_tuser => s_axis_tuser,          -- TUSER_WIDTH-bit input: TUSER: The user-defined sideband
                                          -- information that can be transmitted alongside the data
                                                -- stream.

   s_axis_tvalid => s_axis_tvalid         -- 1-bit input: TVALID: Indicates that the master is driving a
                                          -- valid transfer. A transfer takes place when both TVALID and
                                                -- TREADY are asserted

);

-- End of xpm_fifo_axis_inst instantiation
```

Verilog Instantiation Template

```
// xpm_fifo_axis: AXI Stream FIFO
// Xilinx Parameterized Macro, version 2020.1

xpm_fifo_axis #(
   .CDC_SYNC_STAGES(2),             // DECIMAL
   .CLOCKING_MODE("common_clock"), // String
   .ECC_MODE("no_ecc"),             // String
   .FIFO_DEPTH(2048),               // DECIMAL
   .FIFO_MEMORY_TYPE("auto"),       // String
   .PACKET_FIFO("false"),           // String
   .PROG_EMPTY_THRESH(10),          // DECIMAL
   .PROG_FULL_THRESH(10),           // DECIMAL
   .RD_DATA_COUNT_WIDTH(1),         // DECIMAL
   .RELATED_CLOCKS(0),              // DECIMAL
   .SIM_ASSERT_CHK(0),            // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .TDATA_WIDTH(32),                // DECIMAL
   .TDEST_WIDTH(1),                 // DECIMAL
   .TID_WIDTH(1),                   // DECIMAL
   .TUSER_WIDTH(1),                 // DECIMAL
   .USE_ADV_FEATURES("1000"),       // String
   .WR_DATA_COUNT_WIDTH(1)          // DECIMAL
)
xpm_fifo_axis_inst (
   .almost_empty_axis(almost_empty_axis),   // 1-bit output: Almost Empty : When asserted, this signal
                                            // indicates that only one more read can be performed before the
                                                  // FIFO goes to empty.

   .almost_full_axis(almost_full_axis),      // 1-bit output: Almost Full: When asserted, this signal
                                            // indicates that only one more write can be performed before
                                                  // the FIFO is full.

   .dbiterr_axis(dbiterr_axis),             // 1-bit output: Double Bit Error- Indicates that the ECC
                                            // decoder detected a double-bit error and data in the FIFO core
                                                  // is corrupted.

   .m_axis_tdata(m_axis_tdata),             // TDATA_WIDTH-bit output: TDATA: The primary payload that is
                                                  // used to provide the data that is passing across the
                                            // interface. The width of the data payload is an integer number
                                                  // of bytes.

   .m_axis_tdest(m_axis_tdest),             // TDEST_WIDTH-bit output: TDEST: Provides routing information
                                                  // for the data stream.

   .m_axis_tid(m_axis_tid),                 // TID_WIDTH-bit output: TID: The data stream identifier that
                                                  // indicates different streams of data.

   .m_axis_tkeep(m_axis_tkeep),             // TDATA_WIDTH/8-bit output: TKEEP: The byte qualifier that
                                            // indicates whether the content of the associated byte of TDATA
                                            // is processed as part of the data stream. Associated bytes
                                            // that have the TKEEP byte qualifier deasserted are null bytes
```

```
                                    // and can be removed from the data stream. For a 64-bit DATA,
                                    // bit 0 corresponds to the least significant byte on DATA, and
                                    // bit 7 corresponds to the most significant byte. For example:
                                    // KEEP[0] = 1b, DATA[7:0] is not a NULL byte KEEP[7] = 0b,
                                        // DATA[63:56] is a NULL byte

.m_axis_tlast(m_axis_tlast),        // 1-bit output: TLAST: Indicates the boundary of a packet.
.m_axis_tstrb(m_axis_tstrb),        // TDATA_WIDTH/8-bit output: TSTRB: The byte qualifier that
                                    // indicates whether the content of the associated byte of TDATA
                                    // is processed as a data byte or a position byte. For a 64-bit
                                    // DATA, bit 0 corresponds to the least significant byte on
                                    // DATA, and bit 0 corresponds to the least significant byte on
                                    // DATA, and bit 7 corresponds to the most significant byte. For
                                    // example: STROBE[0] = 1b, DATA[7:0] is valid STROBE[7] = 0b,
                                        // DATA[63:56] is not valid

.m_axis_tuser(m_axis_tuser),        // TUSER_WIDTH-bit output: TUSER: The user-defined sideband
                                    // information that can be transmitted alongside the data
                                        // stream.

.m_axis_tvalid(m_axis_tvalid),      // 1-bit output: TVALID: Indicates that the master is driving a
                                    // valid transfer. A transfer takes place when both TVALID and
                                        // TREADY are asserted

.prog_empty_axis(prog_empty_axis),  // 1-bit output: Programmable Empty- This signal is asserted
                                    // when the number of words in the FIFO is less than or equal to
                                    // the programmable empty threshold value. It is de-asserted
                                    // when the number of words in the FIFO exceeds the programmable
                                        // empty threshold value.

.prog_full_axis(prog_full_axis),    // 1-bit output: Programmable Full: This signal is asserted when
                                    // the number of words in the FIFO is greater than or equal to
                                    // the programmable full threshold value. It is de-asserted when
                                    // the number of words in the FIFO is less than the programmable
                                        // full threshold value.

.rd_data_count_axis(rd_data_count_axis), // RD_DATA_COUNT_WIDTH-bit output: Read Data Count- This bus
                                    // indicates the number of words available for reading in the
                                        // FIFO.

.s_axis_tready(s_axis_tready),      // 1-bit output: TREADY: Indicates that the slave can accept a
                                        // transfer in the current cycle.

.sbiterr_axis(sbiterr_axis),        // 1-bit output: Single Bit Error- Indicates that the ECC
                                        // decoder detected and fixed a single-bit error.

.wr_data_count_axis(wr_data_count_axis), // WR_DATA_COUNT_WIDTH-bit output: Write Data Count: This bus
                                        // indicates the number of words written into the FIFO.

.injectdbiterr_axis(injectdbiterr_axis), // 1-bit input: Double Bit Error Injection- Injects a double bit
                                        // error if the ECC feature is used.

.injectsbiterr_axis(injectsbiterr_axis), // 1-bit input: Single Bit Error Injection- Injects a single bit
                                        // error if the ECC feature is used.

.m_aclk(m_aclk),                    // 1-bit input: Master Interface Clock: All signals on master
                                    // interface are sampled on the rising edge of this clock.

.m_axis_tready(m_axis_tready),      // 1-bit input: TREADY: Indicates that the slave can accept a
                                        // transfer in the current cycle.

.s_aclk(s_aclk),                    // 1-bit input: Slave Interface Clock: All signals on slave
                                    // interface are sampled on the rising edge of this clock.

.s_aresetn(s_aresetn),                  // 1-bit input: Active low asynchronous reset.
.s_axis_tdata(s_axis_tdata),        // TDATA_WIDTH-bit input: TDATA: The primary payload that is
                                        // used to provide the data that is passing across the
                                    // interface. The width of the data payload is an integer number
                                        // of bytes.

.s_axis_tdest(s_axis_tdest),        // TDEST_WIDTH-bit input: TDEST: Provides routing information
                                        // for the data stream.

.s_axis_tid(s_axis_tid),            // TID_WIDTH-bit input: TID: The data stream identifier that
                                        // indicates different streams of data.

.s_axis_tkeep(s_axis_tkeep),        // TDATA_WIDTH/8-bit input: TKEEP: The byte qualifier that
                                    // indicates whether the content of the associated byte of TDATA
                                    // is processed as part of the data stream. Associated bytes
```

```
                                        // that have the TKEEP byte qualifier deasserted are null bytes
                                        // and can be removed from the data stream. For a 64-bit DATA,
                                        // bit 0 corresponds to the least significant byte on DATA, and
                                        // bit 7 corresponds to the most significant byte. For example:
                                        // KEEP[0] = 1b, DATA[7:0] is not a NULL byte KEEP[7] = 0b,
                                              // DATA[63:56] is a NULL byte

   .s_axis_tlast(s_axis_tlast),         // 1-bit input: TLAST: Indicates the boundary of a packet.
   .s_axis_tstrb(s_axis_tstrb),         // TDATA_WIDTH/8-bit input: TSTRB: The byte qualifier that
                                        // indicates whether the content of the associated byte of TDATA
                                        // is processed as a data byte or a position byte. For a 64-bit
                                        // DATA, bit 0 corresponds to the least significant byte on
                                        // DATA, and bit 0 corresponds to the least significant byte on
                                        // DATA, and bit 7 corresponds to the most significant byte. For
                                        // example: STROBE[0] = 1b, DATA[7:0] is valid STROBE[7] = 0b,
                                              // DATA[63:56] is not valid

   .s_axis_tuser(s_axis_tuser),         // TUSER_WIDTH-bit input: TUSER: The user-defined sideband
                                        // information that can be transmitted alongside the data
                                              // stream.

   .s_axis_tvalid(s_axis_tvalid)        // 1-bit input: TVALID: Indicates that the master is driving a
                                        // valid transfer. A transfer takes place when both TVALID and
                                              // TREADY are asserted

);

// End of xpm_fifo_axis_inst instantiation
```

For More Information

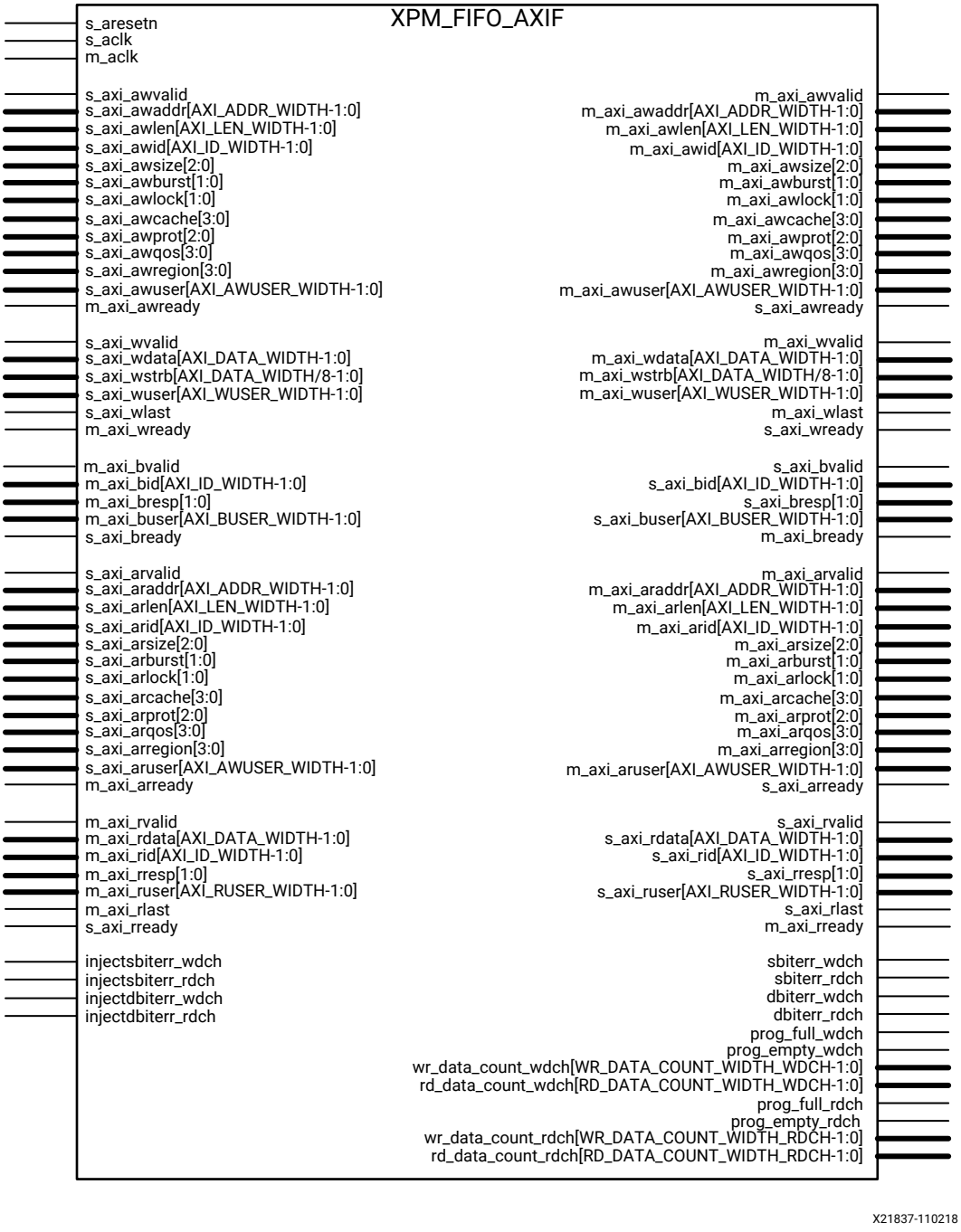
- XPM FIFO Testbench File

XPM_FIFO_AXIF

Parameterized Macro: AXI Memory Mapped (AXI Full) FIFO

MACRO_GROUP: XPM

MACRO_SUBGROUP: XPM_FIFO



Introduction

This macro is used to instantiate AXI Memory Mapped (AXI Full) FIFO.

AXI4 FIFO is derived from the XPM_FIFO_SYNC and XPM_FIFO_ASYNC. The AXI interface protocol uses a two-way valid and ready handshake mechanism. The information source uses the valid signal to show when valid data or control information is available on the channel. The information destination uses the ready signal to show when it can accept the data.

Timing Diagrams

Figure 13: **Timing for Read Operations to the AXI4 FIFO**

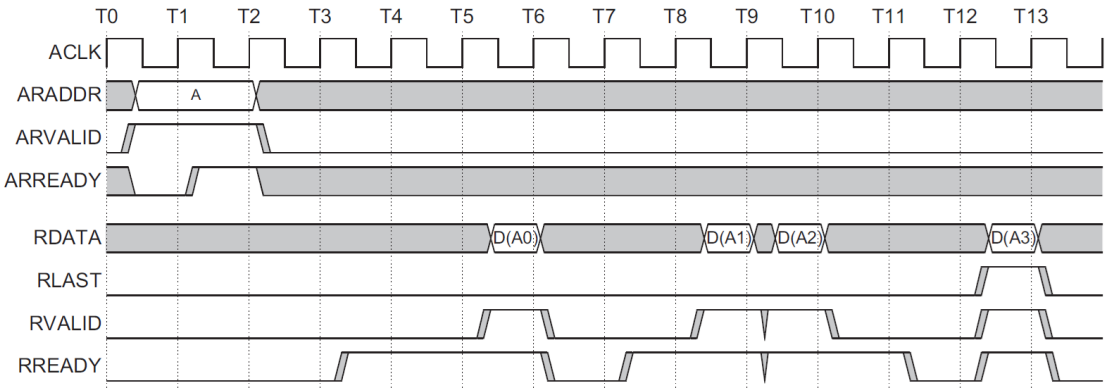
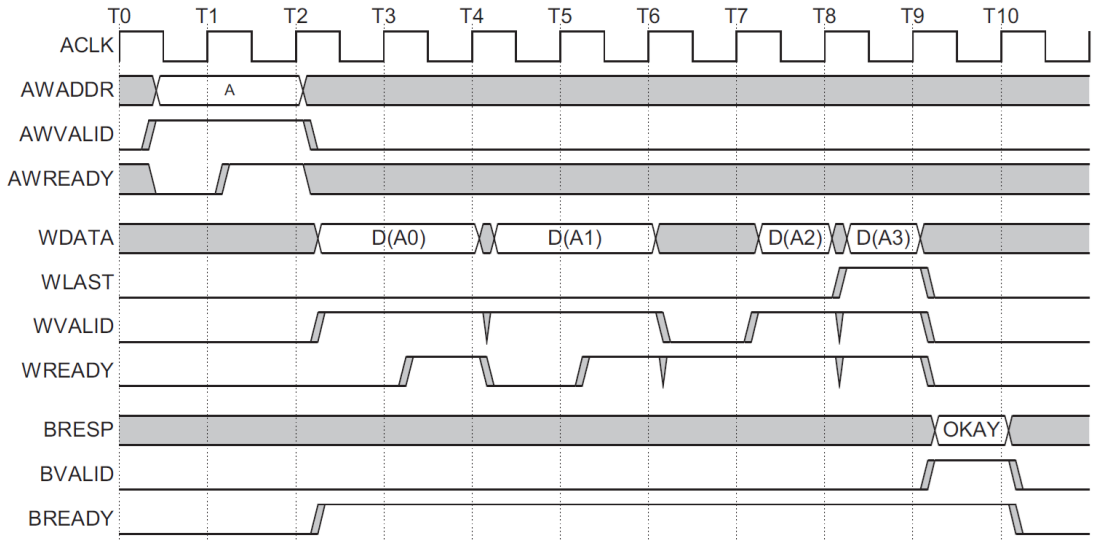


Figure 14: **Timing Write Operations to the AXI4 FIFO**



In the timing diagrams above, the information source generates the valid signal to indicate when the data is available. The destination generates the ready signal to indicate that it can accept the data, and transfer occurs only when both the valid and ready signals are High.

Because AXI4 FIFO is derived from XPM_FIFO_SYNC and XPM_FIFO_ASYNC, much of the behavior is common between them. The ready signal is generated based on availability of space in the FIFO and is held high to allow writes to the FIFO. The ready signal is pulled Low only when there is no space in the FIFO left to perform additional writes. The valid signal is generated based on availability of data in the FIFO and is held High to allow reads to be performed from the FIFO. The valid signal is pulled Low only when there is no data available to be read from the FIFO. The information signals are mapped to the din and dout bus of XPM_FIFO_SYNC and XPM_FIFO_ASYNC. The width of the AXI4 FIFO is determined by concatenating all of the information signals of the AXI interface. The information signals include all AXI signals except for the valid and ready handshake signals.

AXI4 FIFO operates only in First-Word Fall-Through mode. The First-Word Fall-Through (FWFT) feature provides the ability to look ahead to the next word available from the FIFO without issuing a read operation. When data is available in the FIFO, the first word falls through the FIFO and appears automatically on the output data bus.

Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
dbiterr_rdch	Output	1	m_aclk	LEVEL _HIGH	DoNotCare	Double Bit Error- Indicates that the ECC decoder detected a double-bit error and data in the FIFO core is corrupted.
dbiterr_wdch	Output	1	m_aclk	LEVEL _HIGH	DoNotCare	Double Bit Error- Indicates that the ECC decoder detected a double-bit error and data in the FIFO core is corrupted.
injectdbiterr _rdch	Input	1	s_aclk	LEVEL _HIGH	0	Double Bit Error Injection- Injects a double bit error if the ECC feature is used.
injectdbiterr _wdch	Input	1	s_aclk	LEVEL _HIGH	0	Double Bit Error Injection- Injects a double bit error if the ECC feature is used.
injectsbiterr _rdch	Input	1	s_aclk	LEVEL _HIGH	0	Single Bit Error Injection- Injects a single bit error if the ECC feature is used.
injectsbiterr _wdch	Input	1	s_aclk	LEVEL _HIGH	0	Single Bit Error Injection- Injects a single bit error if the ECC feature is used.
m_aclk	Input	1	NA	EDGE _RISING	Active	Master Interface Clock: All signals on master interface are sampled on the rising edge of this clock.
m_axi_araddr	Output	AXI _ADDR _WIDTH	m_aclk	NA	Active	ARADDR: The read address bus gives the initial address of a read burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
m_axi_arburst	Output	1	m_aclk	NA	Active	ARBURST: The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
m_axi_arcache	Output	1	m_aclk	NA	Active	ARCACHE: Indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction.
m_axi_arid	Output	AXI _ID _WIDTH	m_aclk	NA	Active	ARID: The data stream identifier that indicates different streams of data.
m_axi_arlen	Output	AXI _LEN _WIDTH	m_aclk	NA	Active	ARLEN: The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
m_axi_arlock	Output	1	m_aclk	NA	Active	ARLOCK: This signal provides additional information about the atomic characteristics of the transfer.
m_axi_arprot	Output	1	m_aclk	NA	Active	ARPROT: Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access.
m_axi_arqos	Output	1	m_aclk	NA	Active	ARQOS: Quality of Service (QoS) sent on the write address channel for each write transaction.
m_axi_arready	Input	1	m_aclk	LEVEL _HIGH	Active	ARREADY: Indicates that the master can accept a transfer in the current cycle.
m_axi_arregion	Output	1	m_aclk	NA	Active	ARREGION: Region Identifier sent on the write address channel for each write transaction.
m_axi_arsize	Output	1	m_aclk	NA	Active	ARSIZE: Indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update.
m_axi_aruser	Output	AXI _ARUSER _WIDTH	m_aclk	NA	Active	ARUSER: The user-defined sideband information that can be transmitted alongside the data stream.
m_axi_arvalid	Output	1	m_aclk	LEVEL _HIGH	Active	ARVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both ARVALID and ARREADY are asserted
m_axi_awaddr	Output	AXI _ADDR _WIDTH	m_aclk	NA	Active	AWADDR: The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst.
m_axi_awburst	Output	1	m_aclk	NA	Active	AWSIZE: The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
m_axi_awcache	Output	1	m_aclk	NA	Active	AWCACHE: Indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction.
m_axi_awid	Output	AXI _ID _WIDTH	m_aclk	NA	Active	AWID: Identification tag for the write address group of signals.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
m_axi_awlen	Output	AXI _LEN _WIDTH	m_aclk	NA	Active	AWLEN: The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
m_axi_awlock	Output	1	m_aclk	NA	Active	AWLOCK: This signal provides additional information about the atomic characteristics of the transfer.
m_axi_awprot	Output	1	m_aclk	NA	Active	AWPROT: Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access.
m_axi_awqos	Output	1	m_aclk	NA	Active	AWQOS: Quality of Service (QoS) sent on the write address channel for each write transaction.
m_axi_awready	Input	1	m_aclk	LEVEL _HIGH	Active	AWREADY: Indicates that the master can accept a transfer in the current cycle.
m_axi_awregion	Output	1	m_aclk	NA	Active	AWREGION: Region Identifier sent on the write address channel for each write transaction.
m_axi_awsize	Output	1	m_aclk	NA	Active	AWSIZE: Indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update.
m_axi_awuser	Output	AXI _AWUSER _WIDTH	m_aclk	NA	Active	AWUSER: The user-defined sideband information that can be transmitted alongside the data stream.
m_axi_awvalid	Output	1	m_aclk	LEVEL _HIGH	Active	AWVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both AWVALID and AWREADY are asserted
m_axi_bid	Input	AXI _ID _WIDTH	m_aclk	NA	Active	BID: The data stream identifier that indicates different streams of data.
m_axi_bready	Output	1	m_aclk	LEVEL _HIGH	Active	BREADY: Indicates that the master can accept a transfer in the current cycle.
m_axi_bresp	Input	1	m_aclk	NA	Active	BRESP: Indicates the status of the write transaction. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.
m_axi_buser	Input	AXI _BUSER _WIDTH	m_aclk	NA	Active	BUSER: The user-defined sideband information that can be transmitted alongside the data stream.
m_axi_bvalid	Input	1	m_aclk	LEVEL _HIGH	Active	BVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both BVALID and BREADY are asserted
m_axi_rdata	Input	AXI _DATA _WIDTH	m_aclk	NA	Active	RDATA: The primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
m_axi_rid	Input	AXI _ID _WIDTH	m_aclk	NA	Active	RID: The data stream identifier that indicates different streams of data.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
m_axi_rlast	Input	1	m_aclk	LEVEL _HIGH	Active	RLAST: Indicates the boundary of a packet.
m_axi_rready	Output	1	m_aclk	LEVEL _HIGH	Active	RREADY: Indicates that the master can accept a transfer in the current cycle.
m_axi_rresp	Input	1	m_aclk	NA	Active	RRESP: Indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.
m_axi_ruser	Input	AXI _RUSER _WIDTH	m_aclk	NA	Active	RUSER: The user-defined sideband information that can be transmitted alongside the data stream.
m_axi_rvalid	Input	1	m_aclk	LEVEL _HIGH	Active	RVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both RVALID and RREADY are asserted
m_axi_wdata	Output	AXI _DATA _WIDTH	m_aclk	NA	Active	WDATA: The primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
m_axi_wlast	Output	1	m_aclk	LEVEL _HIGH	Active	WLAST: Indicates the boundary of a packet.
m_axi_wready	Input	1	m_aclk	LEVEL _HIGH	Active	WREADY: Indicates that the master can accept a transfer in the current cycle.
m_axi_wstrb	Output	AXI _DATA _WIDTH / 8	m_aclk	NA	Active	WSTRB: The byte qualifier that indicates whether the content of the associated byte of TDATA is processed as a data byte or a position byte. For a 64-bit DATA, bit 0 corresponds to the least significant byte on DATA, and bit 0 corresponds to the least significant byte on DATA, and bit 7 corresponds to the most significant byte. For example: • STROBE[0] = 1b, DATA[7:0] is valid • STROBE[7] = 0b, DATA[63:56] is not valid
m_axi_wuser	Output	AXI _WUSER _WIDTH	m_aclk	NA	Active	WUSER: The user-defined sideband information that can be transmitted alongside the data stream.
m_axi_wvalid	Output	1	m_aclk	LEVEL _HIGH	Active	WVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both WVALID and WREADY are asserted
prog_empty_rdch	Output	1	m_aclk	LEVEL _HIGH	DoNotCare	Programmable Empty- This signal is asserted when the number of words in the Read Data Channel FIFO is less than or equal to the programmable empty threshold value. It is de-asserted when the number of words in the Read Data Channel FIFO exceeds the programmable empty threshold value.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
prog_empty_wdch	Output	1	m_aclk	LEVEL_HIGH	DoNotCare	Programmable Empty- This signal is asserted when the number of words in the Write Data Channel FIFO is less than or equal to the programmable empty threshold value. It is de-asserted when the number of words in the Write Data Channel FIFO exceeds the programmable empty threshold value.
prog_full_rdch	Output	1	s_aclk	LEVEL_HIGH	DoNotCare	Programmable Full: This signal is asserted when the number of words in the Read Data Channel FIFO is greater than or equal to the programmable full threshold value. It is de-asserted when the number of words in the Read Data Channel FIFO is less than the programmable full threshold value.
prog_full_wdch	Output	1	s_aclk	LEVEL_HIGH	DoNotCare	Programmable Full: This signal is asserted when the number of words in the Write Data Channel FIFO is greater than or equal to the programmable full threshold value. It is de-asserted when the number of words in the Write Data Channel FIFO is less than the programmable full threshold value.
rd_data_count_rdch	Output	RD_DATA_COUNT_WIDTH_RDCH	m_aclk	NA	DoNotCare	Read Data Count- This bus indicates the number of words available for reading in the Read Data Channel FIFO.
rd_data_count_wdch	Output	RD_DATA_COUNT_WIDTH_WDCH	m_aclk	NA	DoNotCare	Read Data Count- This bus indicates the number of words available for reading in the Write Data Channel FIFO.
s_aclk	Input	1	NA	EDGE_RISING	Active	Slave Interface Clock: All signals on slave interface are sampled on the rising edge of this clock.
s_aresetn	Input	1	NA	LEVEL_LOW	Active	Active low asynchronous reset.
s_axi_araddr	Input	AXI_ADDR_WIDTH	s_aclk	NA	Active	ARADDR: The read address bus gives the initial address of a read burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.
s_axi_arburst	Input	1	s_aclk	NA	Active	ARBURST: The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
s_axi_arcache	Input	1	s_aclk	NA	Active	ARCACHE: Indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction.
s_axi_arid	Input	AXI_ID_WIDTH	s_aclk	NA	Active	ARID: The data stream identifier that indicates different streams of data.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
s_axi_arlen	Input	AXI _LEN _WIDTH	s_aclk	NA	Active	ARLEN: The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
s_axi_arlock	Input	1	s_aclk	NA	Active	ARLOCK: This signal provides additional information about the atomic characteristics of the transfer.
s_axi_arprot	Input	1	s_aclk	NA	Active	ARPROT: Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access.
s_axi_arqos	Input	1	s_aclk	NA	Active	ARQOS: Quality of Service (QoS) sent on the write address channel for each write transaction.
s_axi_arready	Output	1	s_aclk	LEVEL _HIGH	Active	ARREADY: Indicates that the slave can accept a transfer in the current cycle.
s_axi_arregion	Input	1	s_aclk	NA	Active	ARREGION: Region Identifier sent on the write address channel for each write transaction.
s_axi_arsize	Input	1	s_aclk	NA	Active	ARSIZE: Indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update.
s_axi_aruser	Input	AXI _ARUSER _WIDTH	s_aclk	NA	Active	ARUSER: The user-defined sideband information that can be transmitted alongside the data stream.
s_axi_arvalid	Input	1	s_aclk	LEVEL _HIGH	Active	ARVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both ARVALID and ARREADY are asserted
s_axi_awaddr	Input	AXI _ADDR _WIDTH	s_aclk	NA	Active	AWADDR: The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst.
s_axi_awburst	Input	1	s_aclk	LEVEL _HIGH	Active	AWBURST: The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
s_axi_awcache	Input	1	s_aclk	LEVEL _HIGH	Active	AWCACHE: Indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction.
s_axi_awid	Input	AXI _ID _WIDTH	s_aclk	NA	Active	AWID: Identification tag for the write address group of signals.
s_axi_awlen	Input	AXI _LEN _WIDTH	s_aclk	NA	Active	AWLEN: The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
s_axi_awlock	Input	1	s_aclk	LEVEL _HIGH	Active	AWLOCK: This signal provides additional information about the atomic characteristics of the transfer.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
s_axi_awprot	Input	1	s_aclk	LEVEL _HIGH	Active	AWPROT: Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access.
s_axi_awqos	Input	1	s_aclk	LEVEL _HIGH	Active	AWQOS: Quality of Service (QoS) sent on the write address channel for each write transaction.
s_axi_awready	Output	1	s_aclk	LEVEL _HIGH	Active	AWREADY: Indicates that the slave can accept a transfer in the current cycle.
s_axi_awregion	Input	1	s_aclk	LEVEL _HIGH	Active	AWREGION: Region Identifier sent on the write address channel for each write transaction.
s_axi_awsize	Input	1	s_aclk	LEVEL _HIGH	Active	AWSIZE: Indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update.
s_axi_awuser	Input	AXI _AWUSER _WIDTH	s_aclk	NA	Active	AWUSER: The user-defined sideband information that can be transmitted alongside the data stream.
s_axi_awvalid	Input	1	s_aclk	LEVEL _HIGH	Active	AWVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both AWVALID and AWREADY are asserted
s_axi_bid	Output	AXI _ID _WIDTH	s_aclk	NA	Active	BID: The data stream identifier that indicates different streams of data.
s_axi_bready	Input	1	s_aclk	LEVEL _HIGH	Active	BREADY: Indicates that the slave can accept a transfer in the current cycle.
s_axi_bresp	Output	1	s_aclk	NA	Active	BRESP: Indicates the status of the write transaction. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.
s_axi_buser	Output	AXI _BUSER _WIDTH	s_aclk	NA	Active	BUSER: The user-defined sideband information that can be transmitted alongside the data stream.
s_axi_bvalid	Output	1	s_aclk	LEVEL _HIGH	Active	BVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both BVALID and BREADY are asserted
s_axi_rdata	Output	AXI _DATA _WIDTH	s_aclk	NA	Active	RDATA: The primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
s_axi_rid	Output	AXI _ID _WIDTH	s_aclk	NA	Active	RID: The data stream identifier that indicates different streams of data.
s_axi_rlast	Output	1	s_aclk	LEVEL _HIGH	Active	RLAST: Indicates the boundary of a packet.
s_axi_rready	Input	1	s_aclk	LEVEL _HIGH	Active	RREADY: Indicates that the slave can accept a transfer in the current cycle.
s_axi_rresp	Output	1	s_aclk	NA	Active	RRESP: Indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
s_axi_ruser	Output	AXI _RUSER _WIDTH	s_aclk	NA	Active	RUSER: The user-defined sideband information that can be transmitted alongside the data stream.
s_axi_rvalid	Output	1	s_aclk	LEVEL _HIGH	Active	RVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both RVALID and RREADY are asserted
s_axi_wdata	Input	AXI _DATA _WIDTH	s_aclk	NA	Active	WDATA: The primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
s_axi_wlast	Input	1	s_aclk	LEVEL _HIGH	Active	WLAST: Indicates the boundary of a packet.
s_axi_wready	Output	1	s_aclk	LEVEL _HIGH	Active	WREADY: Indicates that the slave can accept a transfer in the current cycle.
s_axi_wstrb	Input	AXI _DATA _WIDTH / 8	s_aclk	NA	Active	WSTRB: The byte qualifier that indicates whether the content of the associated byte of TDATA is processed as a data byte or a position byte. For a 64-bit DATA, bit 0 corresponds to the least significant byte on DATA, and bit 0 corresponds to the least significant byte on DATA, and bit 7 corresponds to the most significant byte. For example: • STROBE[0] = 1b, DATA[7:0] is valid • STROBE[7] = 0b, DATA[63:56] is not valid
s_axi_wuser	Input	AXI _WUSER _WIDTH	s_aclk	NA	Active	WUSER: The user-defined sideband information that can be transmitted alongside the data stream.
s_axi_wvalid	Input	1	s_aclk	LEVEL _HIGH	Active	WVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both WVALID and WREADY are asserted
sbiterr_rdch	Output	1	m_aclk	LEVEL _HIGH	DoNotCare	Single Bit Error- Indicates that the ECC decoder detected and fixed a single-bit error.
sbiterr_wdch	Output	1	m_aclk	LEVEL _HIGH	DoNotCare	Single Bit Error- Indicates that the ECC decoder detected and fixed a single-bit error.
wr_data_count _rdch	Output	WR _DATA _COUNT _WIDTH _RDCH	s_aclk	NA	DoNotCare	Write Data Count: This bus indicates the number of words written into the Read Data Channel FIFO.
wr_data_count _wdch	Output	WR _DATA _COUNT _WIDTH _WDCH	s_aclk	NA	DoNotCare	Write Data Count: This bus indicates the number of words written into the Write Data Channel FIFO.

Design Entry Method

Instantiation	No
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
AXI_ADDR_WIDTH	DECIMAL	1 to 64	32	Defines the width of the ADDR ports, s_axi_araddr, s_axi_awaddr, m_axi_araddr and m_axi_awaddr
AXI_ARUSER_WIDTH	DECIMAL	1 to 1024	1	Defines the width of the ARUSER port, s_axi_aruser and m_axi_aruser
AXI_AWUSER_WIDTH	DECIMAL	1 to 1024	1	Defines the width of the AWUSER port, s_axi_awuser and m_axi_awuser
AXI_BUSER_WIDTH	DECIMAL	1 to 1024	1	Defines the width of the BUSER port, s_axi_buser and m_axi_buser
AXI_DATA_WIDTH	DECIMAL	8 to 1024	32	Defines the width of the DATA ports, s_axi_rdata, s_axi_wdata, m_axi_rdata and m_axi_wdata NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
AXI_ID_WIDTH	DECIMAL	1 to 32	1	Defines the width of the ID ports, s_axi_awid, s_axi_wid, s_axi_bid, s_axi_ar_id, s_axi_rid, m_axi_awid, m_axi_wid, m_axi_bid, m_axi_ar_id, and m_axi_rid
AXI_LEN_WIDTH	DECIMAL	8 to 8	8	Defines the width of the LEN ports, s_axi_arlen, s_axi_awlen, m_axi_arlen and m_axi_awlen
AXI_RUSER_WIDTH	DECIMAL	1 to 1024	1	Defines the width of the RUSER port, s_axi_ruser and m_axi_ruser
AXI_WUSER_WIDTH	DECIMAL	1 to 1024	1	Defines the width of the WUSER port, s_axi_wuser and m_axi_wuser
CDC_SYNC_STAGES	DECIMAL	2 to 8	2	Specifies the number of synchronization stages on the CDC path. Applicable only if CLOCKING_MODE = "independent_clock"
CLOCKING_MODE	STRING	"common_clock", "independent_clock"	"common_clock"	Designate whether AXI Memory Mapped FIFO is clocked with a common clock or with independent clocks- • "common_clock"- Common clocking; clock both write and read domain s_aclk • "independent_clock"- Independent clocking; clock write domain with s_aclk and read domain with m_aclk
ECC_MODE_RDCH	STRING	"no_ecc", "en_ecc"	"no_ecc"	• "no_ecc" - Disables ECC • "en_ecc" - Enables both ECC Encoder and Decoder

Attribute	Type	Allowed Values	Default	Description
ECC_MODE_WDCH	STRING	"no_ecc", "en_ecc"	"no_ecc"	• "no_ecc" - Disables ECC • "en_ecc" - Enables both ECC Encoder and Decoder
FIFO_DEPTH_RACH	DECIMAL	16 to 4194304	2048	Defines the AXI Memory Mapped FIFO Write Depth, must be power of two NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
FIFO_DEPTH_RDCH	DECIMAL	16 to 4194304	2048	Defines the AXI Memory Mapped FIFO Write Depth, must be power of two NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
FIFO_DEPTH_WACH	DECIMAL	16 to 4194304	2048	Defines the AXI Memory Mapped FIFO Write Depth, must be power of two NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
FIFO_DEPTH_WDCH	DECIMAL	16 to 4194304	2048	Defines the AXI Memory Mapped FIFO Write Depth, must be power of two NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
FIFO_DEPTH_WRCH	DECIMAL	16 to 4194304	2048	Defines the AXI Memory Mapped FIFO Write Depth, must be power of two NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
FIFO_MEMORY_TYPE_RACH	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the fifo memory primitive (resource type) to use- • "auto"- Allow Vivado Synthesis to choose • "block"- Block RAM FIFO • "distributed"- Distributed RAM FIFO • "ultra"- URAM FIFO NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with FIFO_MEMORY_TYPE_RACH set to "auto".
FIFO_MEMORY_TYPE_RDCH	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the fifo memory primitive (resource type) to use- • "auto"- Allow Vivado Synthesis to choose • "block"- Block RAM FIFO • "distributed"- Distributed RAM FIFO • "ultra"- URAM FIFO NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with FIFO_MEMORY_TYPE_RDCH set to "auto".

Attribute	Type	Allowed Values	Default	Description
FIFO_MEMORY_TYPE_WACH	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the fifo memory primitive (resource type) to use- • "auto"- Allow Vivado Synthesis to choose • "block"- Block RAM FIFO • "distributed"- Distributed RAM FIFO • "ultra"- URAM FIFO NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with FIFO_MEMORY_TYPE_WACH set to "auto".
FIFO_MEMORY_TYPE_WDCH	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the fifo memory primitive (resource type) to use- • "auto"- Allow Vivado Synthesis to choose • "block"- Block RAM FIFO • "distributed"- Distributed RAM FIFO • "ultra"- URAM FIFO NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with FIFO_MEMORY_TYPE_WDCH set to "auto".
FIFO_MEMORY_TYPE_WRCH	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the fifo memory primitive (resource type) to use- • "auto"- Allow Vivado Synthesis to choose • "block"- Block RAM FIFO • "distributed"- Distributed RAM FIFO • "ultra"- URAM FIFO NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with FIFO_MEMORY_TYPE_WRCH set to "auto".
PACKET_FIFO	STRING	"false", "true"	"false"	• "true"- Enables Packet FIFO mode • "false"- Disables Packet FIFO mode NOTE: Packet Mode is available only for Common Clock FIFOs.

Attribute	Type	Allowed Values	Default	Description
PROG_EMPTY _THRESH_RDCH	DECIMAL	5 to 4194301	10	Specifies the minimum number of read words in the FIFO at or below which prog_empty is asserted. • Min_Value = 5 • Max_Value = FIFO_WRITE_DEPTH - 5 NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.
PROG_EMPTY _THRESH_WDCH	DECIMAL	5 to 4194301	10	Specifies the minimum number of read words in the FIFO at or below which prog_empty is asserted. • Min_Value = 5 • Max_Value = FIFO_WRITE_DEPTH - 5 NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.
PROG_FULL_THRESH _RDCH	DECIMAL	5 to 4194301	10	Specifies the maximum number of write words in the FIFO at or above which prog_full is asserted. • Min_Value = 5 + CDC_SYNC_STAGES • Max_Value = FIFO_WRITE_DEPTH - 5 NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.
PROG_FULL_THRESH _WDCH	DECIMAL	5 to 4194301	10	Specifies the maximum number of write words in the FIFO at or above which prog_full is asserted. • Min_Value = 5 + CDC_SYNC_STAGES • Max_Value = FIFO_WRITE_DEPTH - 5 NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.
RD_DATA_COUNT _WIDTH_RDCH	DECIMAL	1 to 23	1	Specifies the width of rd_data_count_rdch. To reflect the correct value, the width should be log2(FIFO_DEPTH)+1.
RD_DATA_COUNT _WIDTH_WDCH	DECIMAL	1 to 23	1	Specifies the width of rd_data_count_wdch. To reflect the correct value, the width should be log2(FIFO_DEPTH)+1.

Attribute	Type	Allowed Values	Default	Description
SIM_ASSERT_CHK	DECIMAL	0 to 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.
USE_ADV_FEATURES _RDCH	STRING	String	"1000"	Enables rd_data_count_rdch, prog_empty_rdch, wr_data_count_rdch, prog_full_rdch sideband signals. • Setting USE_ADV_FEATURES_RCCH[1] to 1 enables prog_full_rdch flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES_RCCH[2] to 1 enables wr_data_count_rdch; Default value of this bit is 0 • Setting USE_ADV_FEATURES_RCCH[9] to 1 enables prog_empty_rdch flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES_RCCH[10] to 1 enables rd_data_count_rdch; Default value of this bit is 0
USE_ADV_FEATURES _WDCH	STRING	String	"1000"	Enables rd_data_count_wdch, prog_empty_wdch, wr_data_count_wdch, prog_full_wdch sideband signals. • Setting USE_ADV_FEATURES_WDCH[1] to 1 enables prog_full_wdch flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES_WDCH[2] to 1 enables wr_data_count_wdch; Default value of this bit is 0 • Setting USE_ADV_FEATURES_WDCH[9] to 1 enables prog_empty_wdch flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES_WDCH[10] to 1 enables rd_data_count_wdch; Default value of this bit is 0
WR_DATA_COUNT _WIDTH_RDCH	DECIMAL	1 to 23	1	Specifies the width of wr_data_count_rdch. To reflect the correct value, the width should be log2(FIFO_DEPTH)+1.
WR_DATA_COUNT _WIDTH_WDCH	DECIMAL	1 to 23	1	Specifies the width of wr_data_count_wdch. To reflect the correct value, the width should be log2(FIFO_DEPTH)+1.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_fifo_axif: AXI Memory Mapped (AXI Full) FIFO
-- Xilinx Parameterized Macro, version 2020.1

xpm_fifo_axif_inst : xpm_fifo_axif
generic map (
   AXI_ADDR_WIDTH => 32,             -- DECIMAL
   AXI_ARUSER_WIDTH => 1,            -- DECIMAL
   AXI_AWUSER_WIDTH => 1,            -- DECIMAL
   AXI_BUSER_WIDTH => 1,             -- DECIMAL
   AXI_DATA_WIDTH => 32,             -- DECIMAL
   AXI_ID_WIDTH => 1,                -- DECIMAL
   AXI_LEN_WIDTH => 8,               -- DECIMAL
   AXI_RUSER_WIDTH => 1,             -- DECIMAL
   AXI_WUSER_WIDTH => 1,             -- DECIMAL
   CDC_SYNC_STAGES => 2,             -- DECIMAL
   CLOCKING_MODE => "common_clock", -- String
   ECC_MODE_RDCH => "no_ecc",        -- String
   ECC_MODE_WDCH => "no_ecc",        -- String
   FIFO_DEPTH_RACH => 2048,          -- DECIMAL
   FIFO_DEPTH_RDCH => 2048,          -- DECIMAL
   FIFO_DEPTH_WACH => 2048,          -- DECIMAL
   FIFO_DEPTH_WDCH => 2048,          -- DECIMAL
   FIFO_DEPTH_WRCH => 2048,          -- DECIMAL
   FIFO_MEMORY_TYPE_RACH => "auto", -- String
   FIFO_MEMORY_TYPE_RDCH => "auto", -- String
   FIFO_MEMORY_TYPE_WACH => "auto", -- String
   FIFO_MEMORY_TYPE_WDCH => "auto", -- String
   FIFO_MEMORY_TYPE_WRCH => "auto", -- String
   PACKET_FIFO => "false",           -- String
   PROG_EMPTY_THRESH_RDCH => 10,     -- DECIMAL
   PROG_EMPTY_THRESH_WDCH => 10,     -- DECIMAL
   PROG_FULL_THRESH_RDCH => 10,      -- DECIMAL
   PROG_FULL_THRESH_WDCH => 10,      -- DECIMAL
   RD_DATA_COUNT_WIDTH_RDCH => 1,    -- DECIMAL
   RD_DATA_COUNT_WIDTH_WDCH => 1,    -- DECIMAL
   SIM_ASSERT_CHK => 0,            -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   USE_ADV_FEATURES_RDCH => "1000", -- String
   USE_ADV_FEATURES_WDCH => "1000", -- String
   WR_DATA_COUNT_WIDTH_RDCH => 1,    -- DECIMAL
   WR_DATA_COUNT_WIDTH_WDCH => 1     -- DECIMAL
)
port map (
   dbiterr_rdch => dbiterr_rdch,            -- 1-bit output: Double Bit Error- Indicates that the ECC
                                            -- decoder detected a double-bit error and data in the FIFO
                                                -- core is corrupted.

   dbiterr_wdch => dbiterr_wdch,            -- 1-bit output: Double Bit Error- Indicates that the ECC
                                            -- decoder detected a double-bit error and data in the FIFO
                                                -- core is corrupted.

   m_axi_araddr => m_axi_araddr,            -- AXI_ADDR_WIDTH-bit output: ARADDR: The read address bus
                                            -- gives the initial address of a read burst transaction. Only
                                            -- the start address of the burst is provided and the control
                                            -- signals that are issued alongside the address detail how the
                                            -- address is calculated for the remaining transfers in the
                                                -- burst.

   m_axi_arburst => m_axi_arburst,          -- 2-bit output: ARBURST: The burst type, coupled with the size
                                            -- information, details how the address for each transfer
                                                -- within the burst is calculated.

   m_axi_arcache => m_axi_arcache,          -- 2-bit output: ARCACHE: Indicates the bufferable, cacheable,
                                            -- write-through, write-back, and allocate attributes of the
                                                -- transaction.

   m_axi_arid => m_axi_arid,                -- AXI_ID_WIDTH-bit output: ARID: The data stream identifier
```

```
                                        -- that indicates different streams of data.
m_axi_arlen => m_axi_arlen,             -- AXI_LEN_WIDTH-bit output: ARLEN: The burst length gives the
                                        -- exact number of transfers in a burst. This information
                                        -- determines the number of data transfers associated with the
                                            -- address.
m_axi_arlock => m_axi_arlock,           -- 2-bit output: ARLOCK: This signal provides additional
                                            -- information about the atomic characteristics of the
                                            -- transfer.
m_axi_arprot => m_axi_arprot,           -- 2-bit output: ARPROT: Indicates the normal, privileged, or
                                        -- secure protection level of the transaction and whether the
                                        -- transaction is a data access or an instruction access.
m_axi_arqos => m_axi_arqos,             -- 2-bit output: ARQOS: Quality of Service (QoS) sent on the
                                            -- write address channel for each write transaction.
m_axi_arregion => m_axi_arregion,       -- 2-bit output: ARREGION: Region Identifier sent on the write
                                            -- address channel for each write transaction.
m_axi_arsize => m_axi_arsize,           -- 2-bit output: ARSIZE: Indicates the size of each transfer in
                                        -- the burst. Byte lane strobes indicate exactly which byte
                                            -- lanes to update.
m_axi_aruser => m_axi_aruser,           -- AXI_ARUSER_WIDTH-bit output: ARUSER: The user-defined
                                        -- sideband information that can be transmitted alongside the
                                            -- data stream.
m_axi_arvalid => m_axi_arvalid,         -- 1-bit output: ARVALID: Indicates that the master is driving
                                        -- a valid transfer. A transfer takes place when both ARVALID
                                            -- and ARREADY are asserted
m_axi_awaddr => m_axi_awaddr,           -- AXI_ADDR_WIDTH-bit output: AWADDR: The write address bus
                                        -- gives the address of the first transfer in a write burst
                                        -- transaction. The associated control signals are used to
                                        -- determine the addresses of the remaining transfers in the
                                            -- burst.
m_axi_awburst => m_axi_awburst,         -- 2-bit output: AWSIZE: The burst type, coupled with the size
                                        -- information, details how the address for each transfer
                                            -- within the burst is calculated.
m_axi_awcache => m_axi_awcache,         -- 2-bit output: AWCACHE: Indicates the bufferable, cacheable,
                                        -- write-through, write-back, and allocate attributes of the
                                            -- transaction.
m_axi_awid => m_axi_awid,               -- AXI_ID_WIDTH-bit output: AWID: Identification tag for the
                                            -- write address group of signals.
m_axi_awlen => m_axi_awlen,             -- AXI_LEN_WIDTH-bit output: AWLEN: The burst length gives the
                                        -- exact number of transfers in a burst. This information
                                        -- determines the number of data transfers associated with the
                                            -- address.
m_axi_awlock => m_axi_awlock,           -- 2-bit output: AWLOCK: This signal provides additional
                                            -- information about the atomic characteristics of the
                                            -- transfer.
m_axi_awprot => m_axi_awprot,           -- 2-bit output: AWPROT: Indicates the normal, privileged, or
                                        -- secure protection level of the transaction and whether the
                                        -- transaction is a data access or an instruction access.
m_axi_awqos => m_axi_awqos,             -- 2-bit output: AWQOS: Quality of Service (QoS) sent on the
                                            -- write address channel for each write transaction.
m_axi_awregion => m_axi_awregion,       -- 2-bit output: AWREGION: Region Identifier sent on the write
                                            -- address channel for each write transaction.
m_axi_awsize => m_axi_awsize,           -- 2-bit output: AWSIZE: Indicates the size of each transfer in
                                        -- the burst. Byte lane strobes indicate exactly which byte
                                            -- lanes to update.
m_axi_awuser => m_axi_awuser,           -- AXI_AWUSER_WIDTH-bit output: AWUSER: The user-defined
                                        -- sideband information that can be transmitted alongside the
                                            -- data stream.
m_axi_awvalid => m_axi_awvalid,         -- 1-bit output: AWVALID: Indicates that the master is driving
                                        -- a valid transfer. A transfer takes place when both AWVALID
```

```
                                        -- and AWREADY are asserted
m_axi_bready => m_axi_bready,           -- 1-bit output: BREADY: Indicates that the master can accept a
                                        -- transfer in the current cycle.
m_axi_rready => m_axi_rready,           -- 1-bit output: RREADY: Indicates that the master can accept a
                                        -- transfer in the current cycle.
m_axi_wdata => m_axi_wdata,             -- AXI_DATA_WIDTH-bit output: WDATA: The primary payload that
                                        -- is used to provide the data that is passing across the
                                        -- interface. The width of the data payload is an integer
                                        -- number of bytes.
m_axi_wlast => m_axi_wlast,             -- 1-bit output: WLAST: Indicates the boundary of a packet.
m_axi_wstrb => m_axi_wstrb,             -- AXI_DATA_WIDTH/8-bit output: WSTRB: The byte qualifier that
                                        -- indicates whether the content of the associated byte of
                                        -- TDATA is processed as a data byte or a position byte. For a
                                        -- 64-bit DATA, bit 0 corresponds to the least significant byte
                                        -- on DATA, and bit 0 corresponds to the least significant byte
                                        -- on DATA, and bit 7 corresponds to the most significant byte.
                                        -- For example: STROBE[0] = 1b, DATA[7:0] is valid STROBE[7] =
                                        -- 0b, DATA[63:56] is not valid
m_axi_wuser => m_axi_wuser,             -- AXI_WUSER_WIDTH-bit output: WUSER: The user-defined sideband
                                        -- information that can be transmitted alongside the data
                                        -- stream.
m_axi_wvalid => m_axi_wvalid,           -- 1-bit output: WVALID: Indicates that the master is driving a
                                        -- valid transfer. A transfer takes place when both WVALID and
                                        -- WREADY are asserted
prog_empty_rdch => prog_empty_rdch,     -- 1-bit output: Programmable Empty- This signal is asserted
                                        -- when the number of words in the Read Data Channel FIFO is
                                        -- less than or equal to the programmable empty threshold
                                        -- value. It is de-asserted when the number of words in the
                                        -- Read Data Channel FIFO exceeds the programmable empty
                                        -- threshold value.
prog_empty_wdch => prog_empty_wdch,     -- 1-bit output: Programmable Empty- This signal is asserted
                                        -- when the number of words in the Write Data Channel FIFO is
                                        -- less than or equal to the programmable empty threshold
                                        -- value. It is de-asserted when the number of words in the
                                        -- Write Data Channel FIFO exceeds the programmable empty
                                        -- threshold value.
prog_full_rdch => prog_full_rdch,       -- 1-bit output: Programmable Full: This signal is asserted
                                        -- when the number of words in the Read Data Channel FIFO is
                                        -- greater than or equal to the programmable full threshold
                                        -- value. It is de-asserted when the number of words in the
                                        -- Read Data Channel FIFO is less than the programmable full
                                        -- threshold value.
prog_full_wdch => prog_full_wdch,       -- 1-bit output: Programmable Full: This signal is asserted
                                        -- when the number of words in the Write Data Channel FIFO is
                                        -- greater than or equal to the programmable full threshold
                                        -- value. It is de-asserted when the number of words in the
                                        -- Write Data Channel FIFO is less than the programmable full
                                        -- threshold value.
rd_data_count_rdch => rd_data_count_rdch, -- RD_DATA_COUNT_WIDTH_RDCH-bit output: Read Data Count- This
                                        -- bus indicates the number of words available for reading in
                                        -- the Read Data Channel FIFO.
rd_data_count_wdch => rd_data_count_wdch, -- RD_DATA_COUNT_WIDTH_WDCH-bit output: Read Data Count- This
                                        -- bus indicates the number of words available for reading in
                                        -- the Write Data Channel FIFO.
s_axi_arready => s_axi_arready,         -- 1-bit output: ARREADY: Indicates that the slave can accept a
                                        -- transfer in the current cycle.
s_axi_awready => s_axi_awready,         -- 1-bit output: AWREADY: Indicates that the slave can accept a
                                        -- transfer in the current cycle.
s_axi_bid => s_axi_bid,                 -- AXI_ID_WIDTH-bit output: BID: The data stream identifier
                                        -- that indicates different streams of data.
s_axi_bresp => s_axi_bresp,             -- 2-bit output: BRESP: Indicates the status of the write
                                        -- transaction. The allowable responses are OKAY, EXOKAY,
                                        -- SLVERR, and DECERR.
```

```
s_axi_buser => s_axi_buser,                 -- AXI_BUSER_WIDTH-bit output: BUSER: The user-defined sideband
                                            -- information that can be transmitted alongside the data
                                                -- stream.

s_axi_bvalid => s_axi_bvalid,               -- 1-bit output: BVALID: Indicates that the master is driving a
                                            -- valid transfer. A transfer takes place when both BVALID and
                                                -- BREADY are asserted

s_axi_rdata => s_axi_rdata,                 -- AXI_DATA_WIDTH-bit output: RDATA: The primary payload that
                                            -- is used to provide the data that is passing across the
                                            -- interface. The width of the data payload is an integer
                                                -- number of bytes.

s_axi_rid => s_axi_rid,                     -- AXI_ID_WIDTH-bit output: RID: The data stream identifier
                                                -- that indicates different streams of data.

s_axi_rlast => s_axi_rlast,                 -- 1-bit output: RLAST: Indicates the boundary of a packet.
s_axi_rresp => s_axi_rresp,                 -- 2-bit output: RRESP: Indicates the status of the read
                                            -- transfer. The allowable responses are OKAY, EXOKAY, SLVERR,
                                                -- and DECERR.

s_axi_ruser => s_axi_ruser,                 -- AXI_RUSER_WIDTH-bit output: RUSER: The user-defined sideband
                                            -- information that can be transmitted alongside the data
                                                -- stream.

s_axi_rvalid => s_axi_rvalid,               -- 1-bit output: RVALID: Indicates that the master is driving a
                                            -- valid transfer. A transfer takes place when both RVALID and
                                                -- RREADY are asserted

s_axi_wready => s_axi_wready,               -- 1-bit output: WREADY: Indicates that the slave can accept a
                                                -- transfer in the current cycle.

sbiterr_rdch => sbiterr_rdch,               -- 1-bit output: Single Bit Error- Indicates that the ECC
                                                -- decoder detected and fixed a single-bit error.

sbiterr_wdch => sbiterr_wdch,               -- 1-bit output: Single Bit Error- Indicates that the ECC
                                                -- decoder detected and fixed a single-bit error.

wr_data_count_rdch => wr_data_count_rdch,   -- WR_DATA_COUNT_WIDTH_RDCH-bit output: Write Data Count: This
                                            -- bus indicates the number of words written into the Read Data
                                                -- Channel FIFO.

wr_data_count_wdch => wr_data_count_wdch,   -- WR_DATA_COUNT_WIDTH_WDCH-bit output: Write Data Count: This
                                            -- bus indicates the number of words written into the Write
                                                -- Data Channel FIFO.

injectdbiterr_rdch => injectdbiterr_rdch,   -- 1-bit input: Double Bit Error Injection- Injects a double
                                                -- bit error if the ECC feature is used.

injectdbiterr_wdch => injectdbiterr_wdch,   -- 1-bit input: Double Bit Error Injection- Injects a double
                                                -- bit error if the ECC feature is used.

injectsbiterr_rdch => injectsbiterr_rdch,   -- 1-bit input: Single Bit Error Injection- Injects a single
                                                -- bit error if the ECC feature is used.

injectsbiterr_wdch => injectsbiterr_wdch,   -- 1-bit input: Single Bit Error Injection- Injects a single
                                                -- bit error if the ECC feature is used.

m_aclk => m_aclk,                           -- 1-bit input: Master Interface Clock: All signals on master
                                            -- interface are sampled on the rising edge of this clock.

m_axi_arready => m_axi_arready,             -- 1-bit input: ARREADY: Indicates that the master can accept a
                                                -- transfer in the current cycle.

m_axi_awready => m_axi_awready,             -- 1-bit input: AWREADY: Indicates that the master can accept a
                                                -- transfer in the current cycle.

m_axi_bid => m_axi_bid,                     -- AXI_ID_WIDTH-bit input: BID: The data stream identifier that
                                                -- indicates different streams of data.

m_axi_bresp => m_axi_bresp,                 -- 2-bit input: BRESP: Indicates the status of the write
                                            -- transaction. The allowable responses are OKAY, EXOKAY,
                                                -- SLVERR, and DECERR.

m_axi_buser => m_axi_buser,                 -- AXI_BUSER_WIDTH-bit input: BUSER: The user-defined sideband
                                            -- information that can be transmitted alongside the data
                                                -- stream.
```

```
m_axi_bvalid => m_axi_bvalid,             -- 1-bit input: BVALID: Indicates that the master is driving a
                                          -- valid transfer. A transfer takes place when both BVALID and
                                             -- BREADY are asserted

m_axi_rdata => m_axi_rdata,               -- AXI_DATA_WIDTH-bit input: RDATA: The primary payload that is
                                             -- used to provide the data that is passing across the
                                          -- interface. The width of the data payload is an integer
                                             -- number of bytes.

m_axi_rid => m_axi_rid,                   -- AXI_ID_WIDTH-bit input: RID: The data stream identifier that
                                             -- indicates different streams of data.

m_axi_rlast => m_axi_rlast,               -- 1-bit input: RLAST: Indicates the boundary of a packet.
m_axi_rresp => m_axi_rresp,                  -- 2-bit input: RRESP: Indicates the status of the read
                                          -- transfer. The allowable responses are OKAY, EXOKAY, SLVERR,
                                             -- and DECERR.

m_axi_ruser => m_axi_ruser,               -- AXI_RUSER_WIDTH-bit input: RUSER: The user-defined sideband
                                          -- information that can be transmitted alongside the data
                                             -- stream.

m_axi_rvalid => m_axi_rvalid,             -- 1-bit input: RVALID: Indicates that the master is driving a
                                          -- valid transfer. A transfer takes place when both RVALID and
                                             -- RREADY are asserted

m_axi_wready => m_axi_wready,             -- 1-bit input: WREADY: Indicates that the master can accept a
                                             -- transfer in the current cycle.

s_aclk => s_aclk,                         -- 1-bit input: Slave Interface Clock: All signals on slave
                                          -- interface are sampled on the rising edge of this clock.

s_aresetn => s_aresetn,                      -- 1-bit input: Active low asynchronous reset.
s_axi_araddr => s_axi_araddr,             -- AXI_ADDR_WIDTH-bit input: ARADDR: The read address bus gives
                                          -- the initial address of a read burst transaction. Only the
                                          -- start address of the burst is provided and the control
                                          -- signals that are issued alongside the address detail how the
                                          -- address is calculated for the remaining transfers in the
                                             -- burst.

s_axi_arburst => s_axi_arburst,           -- 2-bit input: ARBURST: The burst type, coupled with the size
                                          -- information, details how the address for each transfer
                                             -- within the burst is calculated.

s_axi_arcache => s_axi_arcache,           -- 2-bit input: ARCACHE: Indicates the bufferable, cacheable,
                                          -- write-through, write-back, and allocate attributes of the
                                             -- transaction.

s_axi_arid => s_axi_arid,                 -- AXI_ID_WIDTH-bit input: ARID: The data stream identifier
                                             -- that indicates different streams of data.

s_axi_arlen => s_axi_arlen,               -- AXI_LEN_WIDTH-bit input: ARLEN: The burst length gives the
                                          -- exact number of transfers in a burst. This information
                                          -- determines the number of data transfers associated with the
                                             -- address.

s_axi_arlock => s_axi_arlock,                -- 2-bit input: ARLOCK: This signal provides additional
                                             -- information about the atomic characteristics of the
                                             -- transfer.

s_axi_arprot => s_axi_arprot,             -- 2-bit input: ARPROT: Indicates the normal, privileged, or
                                          -- secure protection level of the transaction and whether the
                                          -- transaction is a data access or an instruction access.

s_axi_arqos => s_axi_arqos,               -- 2-bit input: ARQOS: Quality of Service (QoS) sent on the
                                             -- write address channel for each write transaction.

s_axi_arregion => s_axi_arregion,         -- 2-bit input: ARREGION: Region Identifier sent on the write
                                             -- address channel for each write transaction.

s_axi_arsize => s_axi_arsize,             -- 2-bit input: ARSIZE: Indicates the size of each transfer in
                                          -- the burst. Byte lane strobes indicate exactly which byte
                                             -- lanes to update.

s_axi_aruser => s_axi_aruser,                -- AXI_ARUSER_WIDTH-bit input: ARUSER: The user-defined
                                          -- sideband information that can be transmitted alongside the
                                             -- data stream.

s_axi_arvalid => s_axi_arvalid,           -- 1-bit input: ARVALID: Indicates that the master is driving a
                                          -- valid transfer. A transfer takes place when both ARVALID and
```

```
                                        -- ARREADY are asserted
s_axi_awaddr => s_axi_awaddr,           -- AXI_ADDR_WIDTH-bit input: AWADDR: The write address bus
                                        -- gives the address of the first transfer in a write burst
                                        -- transaction. The associated control signals are used to
                                        -- determine the addresses of the remaining transfers in the
                                        -- burst.
s_axi_awburst => s_axi_awburst,         -- 2-bit input: AWBURST: The burst type, coupled with the size
                                        -- information, details how the address for each transfer
                                        -- within the burst is calculated.
s_axi_awcache => s_axi_awcache,         -- 2-bit input: AWCACHE: Indicates the bufferable, cacheable,
                                        -- write-through, write-back, and allocate attributes of the
                                        -- transaction.
s_axi_awid => s_axi_awid,               -- AXI_ID_WIDTH-bit input: AWID: Identification tag for the
                                        -- write address group of signals.
s_axi_awlen => s_axi_awlen,             -- AXI_LEN_WIDTH-bit input: AWLEN: The burst length gives the
                                        -- exact number of transfers in a burst. This information
                                        -- determines the number of data transfers associated with the
                                        -- address.
s_axi_awlock => s_axi_awlock,           -- 2-bit input: AWLOCK: This signal provides additional
                                        -- information about the atomic characteristics of the
                                        -- transfer.
s_axi_awprot => s_axi_awprot,           -- 2-bit input: AWPROT: Indicates the normal, privileged, or
                                        -- secure protection level of the transaction and whether the
                                        -- transaction is a data access or an instruction access.
s_axi_awqos => s_axi_awqos,             -- 2-bit input: AWQOS: Quality of Service (QoS) sent on the
                                        -- write address channel for each write transaction.
s_axi_awregion => s_axi_awregion,       -- 2-bit input: AWREGION: Region Identifier sent on the write
                                        -- address channel for each write transaction.
s_axi_awsize => s_axi_awsize,           -- 2-bit input: AWSIZE: Indicates the size of each transfer in
                                        -- the burst. Byte lane strobes indicate exactly which byte
                                        -- lanes to update.
s_axi_awuser => s_axi_awuser,           -- AXI_AWUSER_WIDTH-bit input: AWUSER: The user-defined
                                        -- sideband information that can be transmitted alongside the
                                        -- data stream.
s_axi_awvalid => s_axi_awvalid,         -- 1-bit input: AWVALID: Indicates that the master is driving a
                                        -- valid transfer. A transfer takes place when both AWVALID and
                                        -- AWREADY are asserted
s_axi_bready => s_axi_bready,           -- 1-bit input: BREADY: Indicates that the slave can accept a
                                        -- transfer in the current cycle.
s_axi_rready => s_axi_rready,           -- 1-bit input: RREADY: Indicates that the slave can accept a
                                        -- transfer in the current cycle.
s_axi_wdata => s_axi_wdata,             -- AXI_DATA_WIDTH-bit input: WDATA: The primary payload that is
                                        -- used to provide the data that is passing across the
                                        -- interface. The width of the data payload is an integer
                                        -- number of bytes.
s_axi_wlast => s_axi_wlast,             -- 1-bit input: WLAST: Indicates the boundary of a packet.
s_axi_wstrb => s_axi_wstrb,             -- AXI_DATA_WIDTH/8-bit input: WSTRB: The byte qualifier that
                                        -- indicates whether the content of the associated byte of
                                        -- TDATA is processed as a data byte or a position byte. For a
                                        -- 64-bit DATA, bit 0 corresponds to the least significant byte
                                        -- on DATA, and bit 0 corresponds to the least significant byte
                                        -- on DATA, and bit 7 corresponds to the most significant byte.
                                        -- For example: STROBE[0] = 1b, DATA[7:0] is valid STROBE[7] =
                                        -- 0b, DATA[63:56] is not valid
s_axi_wuser => s_axi_wuser,             -- AXI_WUSER_WIDTH-bit input: WUSER: The user-defined sideband
                                        -- information that can be transmitted alongside the data
                                        -- stream.
s_axi_wvalid => s_axi_wvalid            -- 1-bit input: WVALID: Indicates that the master is driving a
                                        -- valid transfer. A transfer takes place when both WVALID and
```

```
                                          -- WREADY are asserted

);

-- End of xpm_fifo_axif_inst instantiation
```

Verilog Instantiation Template

```
// xpm_fifo_axif: AXI Memory Mapped (AXI Full) FIFO
// Xilinx Parameterized Macro, version 2020.1

xpm_fifo_axif #(
   .AXI_ADDR_WIDTH(32),             // DECIMAL
   .AXI_ARUSER_WIDTH(1),            // DECIMAL
   .AXI_AWUSER_WIDTH(1),            // DECIMAL
   .AXI_BUSER_WIDTH(1),             // DECIMAL
   .AXI_DATA_WIDTH(32),             // DECIMAL
   .AXI_ID_WIDTH(1),                // DECIMAL
   .AXI_LEN_WIDTH(8),               // DECIMAL
   .AXI_RUSER_WIDTH(1),             // DECIMAL
   .AXI_WUSER_WIDTH(1),             // DECIMAL
   .CDC_SYNC_STAGES(2),             // DECIMAL
   .CLOCKING_MODE("common_clock"), // String
   .ECC_MODE_RDCH("no_ecc"),        // String
   .ECC_MODE_WDCH("no_ecc"),        // String
   .FIFO_DEPTH_RACH(2048),          // DECIMAL
   .FIFO_DEPTH_RDCH(2048),          // DECIMAL
   .FIFO_DEPTH_WACH(2048),          // DECIMAL
   .FIFO_DEPTH_WDCH(2048),          // DECIMAL
   .FIFO_DEPTH_WRCH(2048),          // DECIMAL
   .FIFO_MEMORY_TYPE_RACH("auto"), // String
   .FIFO_MEMORY_TYPE_RDCH("auto"), // String
   .FIFO_MEMORY_TYPE_WACH("auto"), // String
   .FIFO_MEMORY_TYPE_WDCH("auto"), // String
   .FIFO_MEMORY_TYPE_WRCH("auto"), // String
   .PACKET_FIFO("false"),           // String
   .PROG_EMPTY_THRESH_RDCH(10),     // DECIMAL
   .PROG_EMPTY_THRESH_WDCH(10),     // DECIMAL
   .PROG_FULL_THRESH_RDCH(10),      // DECIMAL
   .PROG_FULL_THRESH_WDCH(10),      // DECIMAL
   .RD_DATA_COUNT_WIDTH_RDCH(1),    // DECIMAL
   .RD_DATA_COUNT_WIDTH_WDCH(1),    // DECIMAL
   .SIM_ASSERT_CHK(0),              // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .USE_ADV_FEATURES_RDCH("1000"), // String
   .USE_ADV_FEATURES_WDCH("1000"), // String
   .WR_DATA_COUNT_WIDTH_RDCH(1),    // DECIMAL
   .WR_DATA_COUNT_WIDTH_WDCH(1)     // DECIMAL
)
xpm_fifo_axif_inst (
   .dbiterr_rdch(dbiterr_rdch),             // 1-bit output: Double Bit Error- Indicates that the ECC
                                            // decoder detected a double-bit error and data in the FIFO core
                                                  // is corrupted.

   .dbiterr_wdch(dbiterr_wdch),             // 1-bit output: Double Bit Error- Indicates that the ECC
                                            // decoder detected a double-bit error and data in the FIFO core
                                                  // is corrupted.

   .m_axi_araddr(m_axi_araddr),             // AXI_ADDR_WIDTH-bit output: ARADDR: The read address bus gives
                                            // the initial address of a read burst transaction. Only the
                                            // start address of the burst is provided and the control
                                            // signals that are issued alongside the address detail how the
                                            // address is calculated for the remaining transfers in the
                                                  // burst.

   .m_axi_arburst(m_axi_arburst),           // 2-bit output: ARBURST: The burst type, coupled with the size
                                            // information, details how the address for each transfer within
                                                  // the burst is calculated.

   .m_axi_arcache(m_axi_arcache),           // 2-bit output: ARCACHE: Indicates the bufferable, cacheable,
                                            // write-through, write-back, and allocate attributes of the
                                                  // transaction.

   .m_axi_arid(m_axi_arid),                 // AXI_ID_WIDTH-bit output: ARID: The data stream identifier
                                                  // that indicates different streams of data.

   .m_axi_arlen(m_axi_arlen),               // AXI_LEN_WIDTH-bit output: ARLEN: The burst length gives the
```

```
                                    // exact number of transfers in a burst. This information
                                    // determines the number of data transfers associated with the
                                        // address.
    .m_axi_arlock(m_axi_arlock),        // 2-bit output: ARLOCK: This signal provides additional
                                    // information about the atomic characteristics of the transfer.
    .m_axi_arprot(m_axi_arprot),    // 2-bit output: ARPROT: Indicates the normal, privileged, or
                                    // secure protection level of the transaction and whether the
                                    // transaction is a data access or an instruction access.
    .m_axi_arqos(m_axi_arqos),      // 2-bit output: ARQOS: Quality of Service (QoS) sent on the
                                        // write address channel for each write transaction.
    .m_axi_arregion(m_axi_arregion),// 2-bit output: ARREGION: Region Identifier sent on the write
                                        // address channel for each write transaction.
    .m_axi_arsize(m_axi_arsize),    // 2-bit output: ARSIZE: Indicates the size of each transfer in
                                    // the burst. Byte lane strobes indicate exactly which byte
                                        // lanes to update.
    .m_axi_aruser(m_axi_aruser),        // AXI_ARUSER_WIDTH-bit output: ARUSER: The user-defined
                                    // sideband information that can be transmitted alongside the
                                        // data stream.
    .m_axi_arvalid(m_axi_arvalid),  // 1-bit output: ARVALID: Indicates that the master is driving a
                                    // valid transfer. A transfer takes place when both ARVALID and
                                        // ARREADY are asserted
    .m_axi_awaddr(m_axi_awaddr),    // AXI_ADDR_WIDTH-bit output: AWADDR: The write address bus
                                    // gives the address of the first transfer in a write burst
                                    // transaction. The associated control signals are used to
                                    // determine the addresses of the remaining transfers in the
                                        // burst.
    .m_axi_awburst(m_axi_awburst),  // 2-bit output: AWSIZE: The burst type, coupled with the size
                                    // information, details how the address for each transfer within
                                        // the burst is calculated.
    .m_axi_awcache(m_axi_awcache),  // 2-bit output: AWCACHE: Indicates the bufferable, cacheable,
                                    // write-through, write-back, and allocate attributes of the
                                        // transaction.
    .m_axi_awid(m_axi_awid),        // AXI_ID_WIDTH-bit output: AWID: Identification tag for the
                                        // write address group of signals.
    .m_axi_awlen(m_axi_awlen),      // AXI_LEN_WIDTH-bit output: AWLEN: The burst length gives the
                                    // exact number of transfers in a burst. This information
                                    // determines the number of data transfers associated with the
                                        // address.
    .m_axi_awlock(m_axi_awlock),        // 2-bit output: AWLOCK: This signal provides additional
                                    // information about the atomic characteristics of the transfer.
    .m_axi_awprot(m_axi_awprot),    // 2-bit output: AWPROT: Indicates the normal, privileged, or
                                    // secure protection level of the transaction and whether the
                                    // transaction is a data access or an instruction access.
    .m_axi_awqos(m_axi_awqos),      // 2-bit output: AWQOS: Quality of Service (QoS) sent on the
                                        // write address channel for each write transaction.
    .m_axi_awregion(m_axi_awregion),// 2-bit output: AWREGION: Region Identifier sent on the write
                                        // address channel for each write transaction.
    .m_axi_awsize(m_axi_awsize),    // 2-bit output: AWSIZE: Indicates the size of each transfer in
                                    // the burst. Byte lane strobes indicate exactly which byte
                                        // lanes to update.
    .m_axi_awuser(m_axi_awuser),        // AXI_AWUSER_WIDTH-bit output: AWUSER: The user-defined
                                    // sideband information that can be transmitted alongside the
                                        // data stream.
    .m_axi_awvalid(m_axi_awvalid),  // 1-bit output: AWVALID: Indicates that the master is driving a
                                    // valid transfer. A transfer takes place when both AWVALID and
                                        // AWREADY are asserted
    .m_axi_bready(m_axi_bready),    // 1-bit output: BREADY: Indicates that the master can accept a
                                        // transfer in the current cycle.
```

```
.m_axi_rready(m_axi_rready),              // 1-bit output: RREADY: Indicates that the master can accept a
                                             // transfer in the current cycle.

.m_axi_wdata(m_axi_wdata),                // AXI_DATA_WIDTH-bit output: WDATA: The primary payload that is
                                             // used to provide the data that is passing across the
                                          // interface. The width of the data payload is an integer number
                                             // of bytes.

.m_axi_wlast(m_axi_wlast),                // 1-bit output: WLAST: Indicates the boundary of a packet.
.m_axi_wstrb(m_axi_wstrb),                // AXI_DATA_WIDTH/8-bit output: WSTRB: The byte qualifier that
                                          // indicates whether the content of the associated byte of TDATA
                                          // is processed as a data byte or a position byte. For a 64-bit
                                          // DATA, bit 0 corresponds to the least significant byte on
                                          // DATA, and bit 0 corresponds to the least significant byte on
                                          // DATA, and bit 7 corresponds to the most significant byte. For
                                          // example: STROBE[0] = 1b, DATA[7:0] is valid STROBE[7] = 0b,
                                             // DATA[63:56] is not valid

.m_axi_wuser(m_axi_wuser),                // AXI_WUSER_WIDTH-bit output: WUSER: The user-defined sideband
                                          // information that can be transmitted alongside the data
                                             // stream.

.m_axi_wvalid(m_axi_wvalid),              // 1-bit output: WVALID: Indicates that the master is driving a
                                          // valid transfer. A transfer takes place when both WVALID and
                                             // WREADY are asserted

.prog_empty_rdch(prog_empty_rdch),        // 1-bit output: Programmable Empty- This signal is asserted
                                          // when the number of words in the Read Data Channel FIFO is
                                          // less than or equal to the programmable empty threshold value.
                                          // It is de-asserted when the number of words in the Read Data
                                          // Channel FIFO exceeds the programmable empty threshold value.

.prog_empty_wdch(prog_empty_wdch),        // 1-bit output: Programmable Empty- This signal is asserted
                                          // when the number of words in the Write Data Channel FIFO is
                                          // less than or equal to the programmable empty threshold value.
                                          // It is de-asserted when the number of words in the Write Data
                                          // Channel FIFO exceeds the programmable empty threshold value.

.prog_full_rdch(prog_full_rdch),          // 1-bit output: Programmable Full: This signal is asserted when
                                          // the number of words in the Read Data Channel FIFO is greater
                                          // than or equal to the programmable full threshold value. It is
                                          // de-asserted when the number of words in the Read Data Channel
                                          // FIFO is less than the programmable full threshold value.

.prog_full_wdch(prog_full_wdch),          // 1-bit output: Programmable Full: This signal is asserted when
                                          // the number of words in the Write Data Channel FIFO is greater
                                          // than or equal to the programmable full threshold value. It is
                                          // de-asserted when the number of words in the Write Data
                                          // Channel FIFO is less than the programmable full threshold
                                             // value.

.rd_data_count_rdch(rd_data_count_rdch), // RD_DATA_COUNT_WIDTH_RDCH-bit output: Read Data Count- This
                                          // bus indicates the number of words available for reading in
                                             // the Read Data Channel FIFO.

.rd_data_count_wdch(rd_data_count_wdch), // RD_DATA_COUNT_WIDTH_WDCH-bit output: Read Data Count- This
                                          // bus indicates the number of words available for reading in
                                             // the Write Data Channel FIFO.

.s_axi_arready(s_axi_arready),            // 1-bit output: ARREADY: Indicates that the slave can accept a
                                             // transfer in the current cycle.

.s_axi_awready(s_axi_awready),            // 1-bit output: AWREADY: Indicates that the slave can accept a
                                             // transfer in the current cycle.

.s_axi_bid(s_axi_bid),                    // AXI_ID_WIDTH-bit output: BID: The data stream identifier that
                                             // indicates different streams of data.

.s_axi_bresp(s_axi_bresp),                // 2-bit output: BRESP: Indicates the status of the write
                                          // transaction. The allowable responses are OKAY, EXOKAY,
                                             // SLVERR, and DECERR.

.s_axi_buser(s_axi_buser),                // AXI_BUSER_WIDTH-bit output: BUSER: The user-defined sideband
                                          // information that can be transmitted alongside the data
                                             // stream.

.s_axi_bvalid(s_axi_bvalid),              // 1-bit output: BVALID: Indicates that the master is driving a
                                          // valid transfer. A transfer takes place when both BVALID and
                                             // BREADY are asserted
```

```
.s_axi_rdata(s_axi_rdata),                 // AXI_DATA_WIDTH-bit output: RDATA: The primary payload that is
                                              // used to provide the data that is passing across the
                                           // interface. The width of the data payload is an integer number
                                              // of bytes.

.s_axi_rid(s_axi_rid),                     // AXI_ID_WIDTH-bit output: RID: The data stream identifier that
                                              // indicates different streams of data.

.s_axi_rlast(s_axi_rlast),                 // 1-bit output: RLAST: Indicates the boundary of a packet.
.s_axi_rresp(s_axi_rresp),                    // 2-bit output: RRESP: Indicates the status of the read
                                           // transfer. The allowable responses are OKAY, EXOKAY, SLVERR,
                                              // and DECERR.

.s_axi_ruser(s_axi_ruser),                 // AXI_RUSER_WIDTH-bit output: RUSER: The user-defined sideband
                                           // information that can be transmitted alongside the data
                                              // stream.

.s_axi_rvalid(s_axi_rvalid),               // 1-bit output: RVALID: Indicates that the master is driving a
                                           // valid transfer. A transfer takes place when both RVALID and
                                              // RREADY are asserted

.s_axi_wready(s_axi_wready),               // 1-bit output: WREADY: Indicates that the slave can accept a
                                              // transfer in the current cycle.

.sbiterr_rdch(sbiterr_rdch),               // 1-bit output: Single Bit Error- Indicates that the ECC
                                              // decoder detected and fixed a single-bit error.

.sbiterr_wdch(sbiterr_wdch),               // 1-bit output: Single Bit Error- Indicates that the ECC
                                              // decoder detected and fixed a single-bit error.

.wr_data_count_rdch(wr_data_count_rdch), // WR_DATA_COUNT_WIDTH_RDCH-bit output: Write Data Count: This
                                           // bus indicates the number of words written into the Read Data
                                              // Channel FIFO.

.wr_data_count_wdch(wr_data_count_wdch), // WR_DATA_COUNT_WIDTH_WDCH-bit output: Write Data Count: This
                                           // bus indicates the number of words written into the Write Data
                                              // Channel FIFO.

.injectdbiterr_rdch(injectdbiterr_rdch), // 1-bit input: Double Bit Error Injection- Injects a double bit
                                              // error if the ECC feature is used.

.injectdbiterr_wdch(injectdbiterr_wdch), // 1-bit input: Double Bit Error Injection- Injects a double bit
                                              // error if the ECC feature is used.

.injectsbiterr_rdch(injectsbiterr_rdch), // 1-bit input: Single Bit Error Injection- Injects a single bit
                                              // error if the ECC feature is used.

.injectsbiterr_wdch(injectsbiterr_wdch), // 1-bit input: Single Bit Error Injection- Injects a single bit
                                              // error if the ECC feature is used.

.m_aclk(m_aclk),                           // 1-bit input: Master Interface Clock: All signals on master
                                           // interface are sampled on the rising edge of this clock.

.m_axi_arready(m_axi_arready),             // 1-bit input: ARREADY: Indicates that the master can accept a
                                              // transfer in the current cycle.

.m_axi_awready(m_axi_awready),             // 1-bit input: AWREADY: Indicates that the master can accept a
                                              // transfer in the current cycle.

.m_axi_bid(m_axi_bid),                     // AXI_ID_WIDTH-bit input: BID: The data stream identifier that
                                              // indicates different streams of data.

.m_axi_bresp(m_axi_bresp),                    // 2-bit input: BRESP: Indicates the status of the write
                                           // transaction. The allowable responses are OKAY, EXOKAY,
                                              // SLVERR, and DECERR.

.m_axi_buser(m_axi_buser),                 // AXI_BUSER_WIDTH-bit input: BUSER: The user-defined sideband
                                           // information that can be transmitted alongside the data
                                              // stream.

.m_axi_bvalid(m_axi_bvalid),               // 1-bit input: BVALID: Indicates that the master is driving a
                                           // valid transfer. A transfer takes place when both BVALID and
                                              // BREADY are asserted

.m_axi_rdata(m_axi_rdata),                 // AXI_DATA_WIDTH-bit input: RDATA: The primary payload that is
                                              // used to provide the data that is passing across the
                                           // interface. The width of the data payload is an integer number
                                              // of bytes.
```

```
.m_axi_rid(m_axi_rid),                  // AXI_ID_WIDTH-bit input: RID: The data stream identifier that
                                            // indicates different streams of data.

.m_axi_rlast(m_axi_rlast),              // 1-bit input: RLAST: Indicates the boundary of a packet.
.m_axi_rresp(m_axi_rresp),                  // 2-bit input: RRESP: Indicates the status of the read
                                        // transfer. The allowable responses are OKAY, EXOKAY, SLVERR,
                                            // and DECERR.

.m_axi_ruser(m_axi_ruser),              // AXI_RUSER_WIDTH-bit input: RUSER: The user-defined sideband
                                        // information that can be transmitted alongside the data
                                            // stream.

.m_axi_rvalid(m_axi_rvalid),            // 1-bit input: RVALID: Indicates that the master is driving a
                                        // valid transfer. A transfer takes place when both RVALID and
                                            // RREADY are asserted

.m_axi_wready(m_axi_wready),            // 1-bit input: WREADY: Indicates that the master can accept a
                                            // transfer in the current cycle.

.s_aclk(s_aclk),                        // 1-bit input: Slave Interface Clock: All signals on slave
                                        // interface are sampled on the rising edge of this clock.

.s_aresetn(s_aresetn),                      // 1-bit input: Active low asynchronous reset.
.s_axi_araddr(s_axi_araddr),            // AXI_ADDR_WIDTH-bit input: ARADDR: The read address bus gives
                                        // the initial address of a read burst transaction. Only the
                                        // start address of the burst is provided and the control
                                        // signals that are issued alongside the address detail how the
                                        // address is calculated for the remaining transfers in the
                                            // burst.

.s_axi_arburst(s_axi_arburst),          // 2-bit input: ARBURST: The burst type, coupled with the size
                                        // information, details how the address for each transfer within
                                            // the burst is calculated.

.s_axi_arcache(s_axi_arcache),          // 2-bit input: ARCACHE: Indicates the bufferable, cacheable,
                                        // write-through, write-back, and allocate attributes of the
                                            // transaction.

.s_axi_arid(s_axi_arid),                // AXI_ID_WIDTH-bit input: ARID: The data stream identifier that
                                            // indicates different streams of data.

.s_axi_arlen(s_axi_arlen),              // AXI_LEN_WIDTH-bit input: ARLEN: The burst length gives the
                                        // exact number of transfers in a burst. This information
                                        // determines the number of data transfers associated with the
                                            // address.

.s_axi_arlock(s_axi_arlock),                // 2-bit input: ARLOCK: This signal provides additional
                                        // information about the atomic characteristics of the transfer.

.s_axi_arprot(s_axi_arprot),            // 2-bit input: ARPROT: Indicates the normal, privileged, or
                                        // secure protection level of the transaction and whether the
                                        // transaction is a data access or an instruction access.

.s_axi_arqos(s_axi_arqos),              // 2-bit input: ARQOS: Quality of Service (QoS) sent on the
                                            // write address channel for each write transaction.

.s_axi_arregion(s_axi_arregion),        // 2-bit input: ARREGION: Region Identifier sent on the write
                                            // address channel for each write transaction.

.s_axi_arsize(s_axi_arsize),            // 2-bit input: ARSIZE: Indicates the size of each transfer in
                                        // the burst. Byte lane strobes indicate exactly which byte
                                            // lanes to update.

.s_axi_aruser(s_axi_aruser),            // AXI_ARUSER_WIDTH-bit input: ARUSER: The user-defined sideband
                                        // information that can be transmitted alongside the data
                                            // stream.

.s_axi_arvalid(s_axi_arvalid),          // 1-bit input: ARVALID: Indicates that the master is driving a
                                        // valid transfer. A transfer takes place when both ARVALID and
                                            // ARREADY are asserted

.s_axi_awaddr(s_axi_awaddr),            // AXI_ADDR_WIDTH-bit input: AWADDR: The write address bus gives
                                            // the address of the first transfer in a write burst
                                        // transaction. The associated control signals are used to
                                        // determine the addresses of the remaining transfers in the
                                            // burst.

.s_axi_awburst(s_axi_awburst),          // 2-bit input: AWBURST: The burst type, coupled with the size
```

```
                                        // information, details how the address for each transfer within
                                            // the burst is calculated.

   .s_axi_awcache(s_axi_awcache),       // 2-bit input: AWCACHE: Indicates the bufferable, cacheable,
                                        // write-through, write-back, and allocate attributes of the
                                            // transaction.

   .s_axi_awid(s_axi_awid),             // AXI_ID_WIDTH-bit input: AWID: Identification tag for the
                                            // write address group of signals.

   .s_axi_awlen(s_axi_awlen),           // AXI_LEN_WIDTH-bit input: AWLEN: The burst length gives the
                                        // exact number of transfers in a burst. This information
                                        // determines the number of data transfers associated with the
                                            // address.

   .s_axi_awlock(s_axi_awlock),             // 2-bit input: AWLOCK: This signal provides additional
                                        // information about the atomic characteristics of the transfer.

   .s_axi_awprot(s_axi_awprot),         // 2-bit input: AWPROT: Indicates the normal, privileged, or
                                        // secure protection level of the transaction and whether the
                                        // transaction is a data access or an instruction access.

   .s_axi_awqos(s_axi_awqos),           // 2-bit input: AWQOS: Quality of Service (QoS) sent on the
                                            // write address channel for each write transaction.

   .s_axi_awregion(s_axi_awregion),     // 2-bit input: AWREGION: Region Identifier sent on the write
                                            // address channel for each write transaction.

   .s_axi_awsize(s_axi_awsize),         // 2-bit input: AWSIZE: Indicates the size of each transfer in
                                        // the burst. Byte lane strobes indicate exactly which byte
                                            // lanes to update.

   .s_axi_awuser(s_axi_awuser),         // AXI_AWUSER_WIDTH-bit input: AWUSER: The user-defined sideband
                                        // information that can be transmitted alongside the data
                                            // stream.

   .s_axi_awvalid(s_axi_awvalid),       // 1-bit input: AWVALID: Indicates that the master is driving a
                                        // valid transfer. A transfer takes place when both AWVALID and
                                            // AWREADY are asserted

   .s_axi_bready(s_axi_bready),         // 1-bit input: BREADY: Indicates that the slave can accept a
                                            // transfer in the current cycle.

   .s_axi_rready(s_axi_rready),         // 1-bit input: RREADY: Indicates that the slave can accept a
                                            // transfer in the current cycle.

   .s_axi_wdata(s_axi_wdata),           // AXI_DATA_WIDTH-bit input: WDATA: The primary payload that is
                                            // used to provide the data that is passing across the
                                        // interface. The width of the data payload is an integer number
                                            // of bytes.

   .s_axi_wlast(s_axi_wlast),           // 1-bit input: WLAST: Indicates the boundary of a packet.
   .s_axi_wstrb(s_axi_wstrb),           // AXI_DATA_WIDTH/8-bit input: WSTRB: The byte qualifier that
                                        // indicates whether the content of the associated byte of TDATA
                                        // is processed as a data byte or a position byte. For a 64-bit
                                        // DATA, bit 0 corresponds to the least significant byte on
                                        // DATA, and bit 0 corresponds to the least significant byte on
                                        // DATA, and bit 7 corresponds to the most significant byte. For
                                        // example: STROBE[0] = 1b, DATA[7:0] is valid STROBE[7] = 0b,
                                            // DATA[63:56] is not valid

   .s_axi_wuser(s_axi_wuser),           // AXI_WUSER_WIDTH-bit input: WUSER: The user-defined sideband
                                        // information that can be transmitted alongside the data
                                            // stream.

   .s_axi_wvalid(s_axi_wvalid)          // 1-bit input: WVALID: Indicates that the master is driving a
                                        // valid transfer. A transfer takes place when both WVALID and
                                            // WREADY are asserted

);

// End of xpm_fifo_axif_inst instantiation
```

For More Information

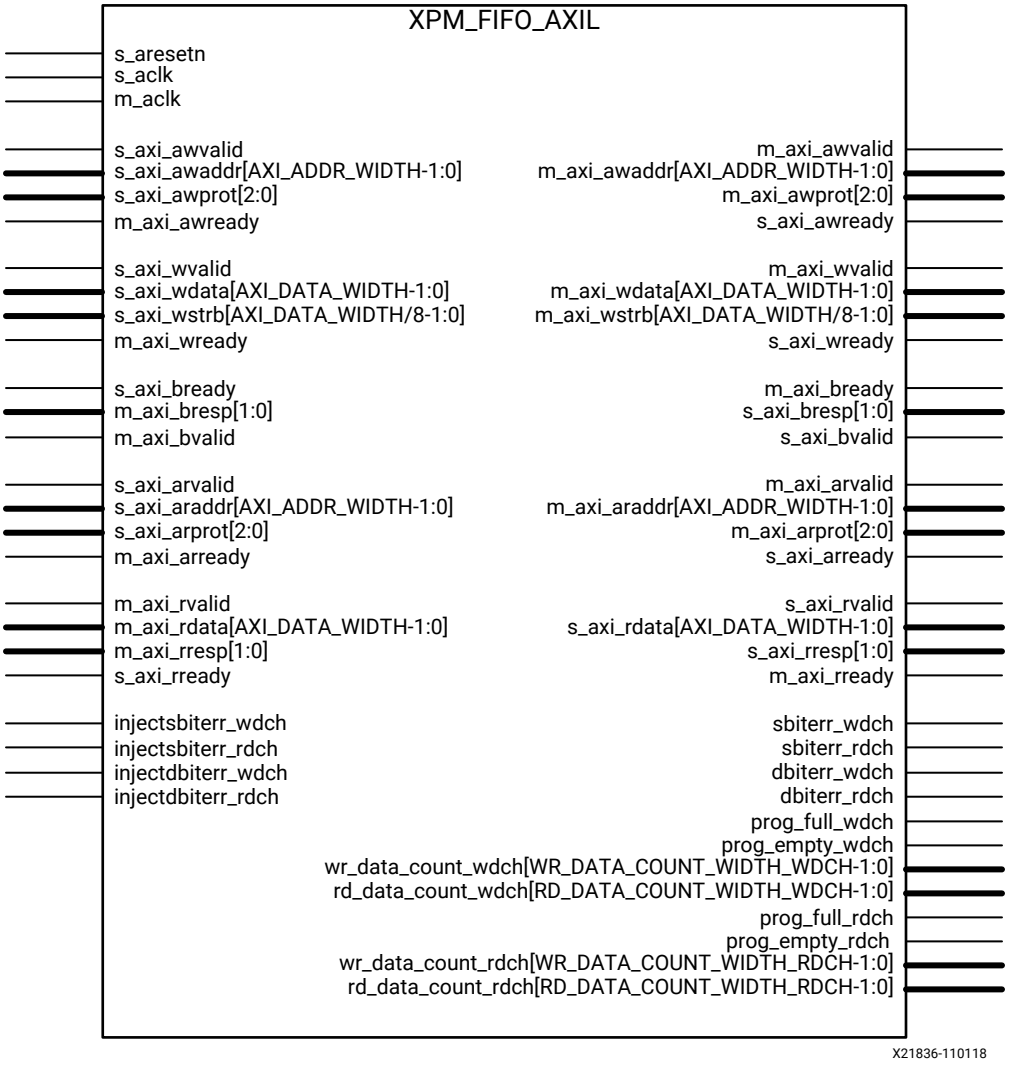
- XPM FIFO Testbench File

XPM_FIFO_AXIL

Parameterized Macro: AXI Memory Mapped (AXI Lite) FIFO

MACRO_GROUP: XPM

MACRO_SUBGROUP: XPM_FIFO



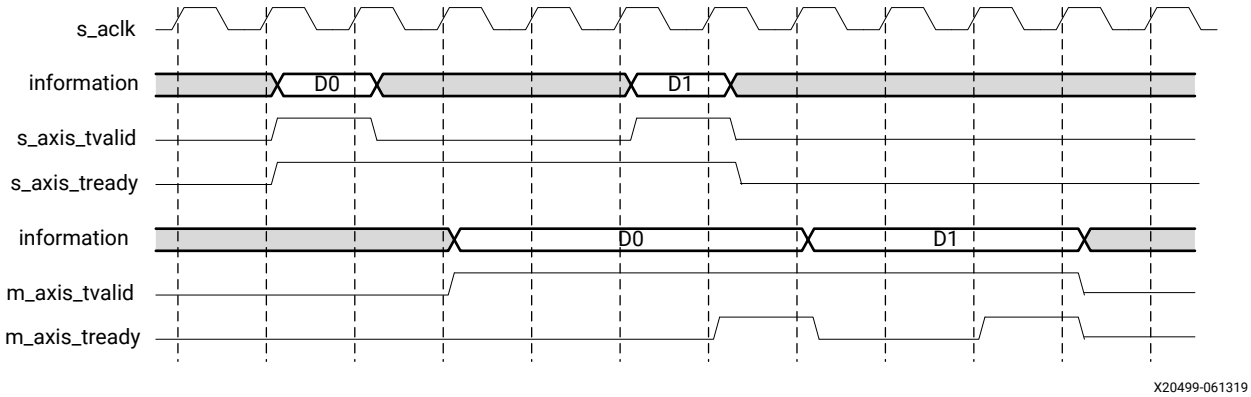
Introduction

This macro is used to instantiate AXI Memory Mapped (AXI Lite) FIFO.

AXI4 FIFO is derived from the XPM_FIFO_SYNC and XPM_FIFO_ASYNC. The AXI interface protocol uses a two-way valid and ready handshake mechanism. The information source uses the valid signal to show when valid data or control information is available on the channel. The information destination uses the ready signal to show when it can accept the data.

Timing Diagrams

Figure 15: **Timing for Read and Write Operations to the AXI Stream FIFO**



In the timing diagram above, the information source generates the valid signal to indicate when the data is available. The destination generates the ready signal to indicate that it can accept the data, and transfer occurs only when both the valid and ready signals are High.

Because AXI4 FIFO is derived from XPM_FIFO_SYNC and XPM_FIFO_ASYNC, much of the behavior is common between them. The ready signal is generated based on availability of space in the FIFO and is held high to allow writes to the FIFO. The ready signal is pulled Low only when there is no space in the FIFO left to perform additional writes. The valid signal is generated based on availability of data in the FIFO and is held High to allow reads to be performed from the FIFO. The valid signal is pulled Low only when there is no data available to be read from the FIFO. The information signals are mapped to the din and dout bus of XPM_FIFO_SYNC and XPM_FIFO_ASYNC. The width of the AXI4-Full FIFO is determined by concatenating all of the information signals of the AXI interface. The information signals include all AXI signals except for the valid and ready handshake signals.

AXI4 FIFO operates only in First-Word Fall-Through mode. The First-Word Fall-Through (FWFT) feature provides the ability to look ahead to the next word available from the FIFO without issuing a read operation. When data is available in the FIFO, the first word falls through the FIFO and appears automatically on the output data bus.

Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
dbiterr_rdch	Output	1	m_aclk	LEVEL _HIGH	DoNotCare	Double Bit Error- Indicates that the ECC decoder detected a double-bit error and data in the FIFO core is corrupted.
dbiterr_wdch	Output	1	m_aclk	LEVEL _HIGH	DoNotCare	Double Bit Error- Indicates that the ECC decoder detected a double-bit error and data in the FIFO core is corrupted.
injectdbiterr _rdch	Input	1	s_aclk	LEVEL _HIGH	0	Double Bit Error Injection- Injects a double bit error if the ECC feature is used.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
injectdbiterr_wdch	Input	1	s_aclk	LEVEL_HIGH	0	Double Bit Error Injection- Injects a double bit error if the ECC feature is used.
injectsbiterr_rdch	Input	1	s_aclk	LEVEL_HIGH	0	Single Bit Error Injection- Injects a single bit error if the ECC feature is used.
injectsbiterr_wdch	Input	1	s_aclk	LEVEL_HIGH	0	Single Bit Error Injection- Injects a single bit error if the ECC feature is used.
m_aclk	Input	1	NA	EDGE_RISING	Active	Master Interface Clock: All signals on master interface are sampled on the rising edge of this clock.
m_axi_araddr	Output	AXI_ADDR_WIDTH	m_aclk	NA	Active	ARADDR: The read address bus gives the initial address of a read burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.
m_axi_arprot	Output	1	m_aclk	NA	Active	ARPROT: Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access.
m_axi_arready	Input	1	m_aclk	LEVEL_HIGH	Active	ARREADY: Indicates that the master can accept a transfer in the current cycle.
m_axi_arvalid	Output	1	m_aclk	LEVEL_HIGH	Active	ARVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both ARVALID and ARREADY are asserted
m_axi_awaddr	Output	AXI_ADDR_WIDTH	m_aclk	NA	Active	AWADDR: The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst.
m_axi_awprot	Output	1	m_aclk	NA	Active	AWPROT: Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access.
m_axi_awready	Input	1	m_aclk	LEVEL_HIGH	Active	AWREADY: Indicates that the master can accept a transfer in the current cycle.
m_axi_awvalid	Output	1	m_aclk	LEVEL_HIGH	Active	AWVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both AWVALID and AWREADY are asserted
m_axi_bready	Output	1	m_aclk	LEVEL_HIGH	Active	BREADY: Indicates that the master can accept a transfer in the current cycle.
m_axi_bresp	Input	1	m_aclk	NA	Active	BRESP: Write Response. Indicates the status of the write transaction. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
m_axi_bvalid	Input	1	m_aclk	LEVEL _HIGH	Active	BVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both BVALID and BREADY are asserted
m_axi_rdata	Input	AXI _DATA _WIDTH	m_aclk	NA	Active	RDATA: The primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
m_axi_rready	Output	1	m_aclk	LEVEL _HIGH	Active	RREADY: Indicates that the master can accept a transfer in the current cycle.
m_axi_rresp	Input	1	m_aclk	NA	Active	RRESP: Indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.
m_axi_rvalid	Input	1	m_aclk	LEVEL _HIGH	Active	RVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both RVALID and RREADY are asserted
m_axi_wdata	Output	AXI _DATA _WIDTH	m_aclk	NA	Active	WDATA: The primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
m_axi_wready	Input	1	m_aclk	LEVEL _HIGH	Active	WREADY: Indicates that the master can accept a transfer in the current cycle.
m_axi_wstrb	Output	AXI _DATA _WIDTH / 8	m_aclk	NA	Active	WSTRB: The byte qualifier that indicates whether the content of the associated byte of TDATA is processed as a data byte or a position byte. For a 64-bit DATA, bit 0 corresponds to the least significant byte on DATA, and bit 0 corresponds to the least significant byte on DATA, and bit 7 corresponds to the most significant byte. For example: • STROBE[0] = 1b, DATA[7:0] is valid • STROBE[7] = 0b, DATA[63:56] is not valid
m_axi_wvalid	Output	1	m_aclk	LEVEL _HIGH	Active	WVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both WVALID and WREADY are asserted
prog_empty_rdch	Output	1	m_aclk	LEVEL _HIGH	DoNotCare	Programmable Empty- This signal is asserted when the number of words in the Read Data Channel FIFO is less than or equal to the programmable empty threshold value. It is de-asserted when the number of words in the Read Data Channel FIFO exceeds the programmable empty threshold value.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
prog_empty_wdch	Output	1	m_aclk	LEVEL_HIGH	DoNotCare	Programmable Empty- This signal is asserted when the number of words in the Write Data Channel FIFO is less than or equal to the programmable empty threshold value. It is de-asserted when the number of words in the Write Data Channel FIFO exceeds the programmable empty threshold value.
prog_full_rdch	Output	1	s_aclk	LEVEL_HIGH	DoNotCare	Programmable Full: This signal is asserted when the number of words in the Read Data Channel FIFO is greater than or equal to the programmable full threshold value. It is de-asserted when the number of words in the Read Data Channel FIFO is less than the programmable full threshold value.
prog_full_wdch	Output	1	s_aclk	LEVEL_HIGH	DoNotCare	Programmable Full: This signal is asserted when the number of words in the Write Data Channel FIFO is greater than or equal to the programmable full threshold value. It is de-asserted when the number of words in the Write Data Channel FIFO is less than the programmable full threshold value.
rd_data_count_rdch	Output	RD_DATA_COUNT_WIDTH_RDCH	m_aclk	NA	DoNotCare	Read Data Count- This bus indicates the number of words available for reading in the Read Data Channel FIFO.
rd_data_count_wdch	Output	RD_DATA_COUNT_WIDTH_WDCH	m_aclk	NA	DoNotCare	Read Data Count- This bus indicates the number of words available for reading in the Write Data Channel FIFO.
s_aclk	Input	1	NA	EDGE_RISING	Active	Slave Interface Clock: All signals on slave interface are sampled on the rising edge of this clock.
s_aresetn	Input	1	NA	LEVEL_LOW	Active	Active low asynchronous reset.
s_axi_araddr	Input	AXI_ADDR_WIDTH	s_aclk	NA	Active	ARADDR: The read address bus gives the initial address of a read burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.
s_axi_arprot	Input	1	s_aclk	NA	Active	ARPROT: Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access.
s_axi_arready	Output	1	s_aclk	LEVEL_HIGH	Active	ARREADY: Indicates that the slave can accept a transfer in the current cycle.
s_axi_arvalid	Input	1	s_aclk	LEVEL_HIGH	Active	ARVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both ARVALID and ARREADY are asserted

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
s_axi_awaddr	Input	AXI _ADDR _WIDTH	s_aclk	NA	Active	AWADDR: The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst.
s_axi_awprot	Input	1	s_aclk	LEVEL _HIGH	Active	AWPROT: Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access.
s_axi_awready	Output	1	s_aclk	LEVEL _HIGH	Active	AWREADY: Indicates that the slave can accept a transfer in the current cycle.
s_axi_awvalid	Input	1	s_aclk	LEVEL _HIGH	Active	AWVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both AWVALID and AWREADY are asserted
s_axi_bready	Input	1	s_aclk	LEVEL _HIGH	Active	BREADY: Indicates that the slave can accept a transfer in the current cycle.
s_axi_bresp	Output	1	s_aclk	NA	Active	BRESP: Write Response. Indicates the status of the write transaction. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.
s_axi_bvalid	Output	1	s_aclk	LEVEL _HIGH	Active	BVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both BVALID and BREADY are asserted
s_axi_rdata	Output	AXI _DATA _WIDTH	s_aclk	NA	Active	RDATA: The primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
s_axi_rready	Input	1	s_aclk	LEVEL _HIGH	Active	RREADY: Indicates that the slave can accept a transfer in the current cycle.
s_axi_rresp	Output	1	s_aclk	NA	Active	RRESP: Indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.
s_axi_rvalid	Output	1	s_aclk	LEVEL _HIGH	Active	RVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both RVALID and RREADY are asserted
s_axi_wdata	Input	AXI _DATA _WIDTH	s_aclk	NA	Active	WDATA: The primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
s_axi_wready	Output	1	s_aclk	LEVEL _HIGH	Active	WREADY: Indicates that the slave can accept a transfer in the current cycle.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
s_axi_wstrb	Input	AXI _DATA _WIDTH / 8	s_aclk	NA	Active	WSTRB: The byte qualifier that indicates whether the content of the associated byte of TDATA is processed as a data byte or a position byte. For a 64-bit DATA, bit 0 corresponds to the least significant byte on DATA, and bit 0 corresponds to the least significant byte on DATA, and bit 7 corresponds to the most significant byte. For example: • STROBE[0] = 1b, DATA[7:0] is valid • STROBE[7] = 0b, DATA[63:56] is not valid
s_axi_wvalid	Input	1	s_aclk	LEVEL _HIGH	Active	WVALID: Indicates that the master is driving a valid transfer. • A transfer takes place when both WVALID and WREADY are asserted
sbiterr_rdch	Output	1	m_aclk	LEVEL _HIGH	DoNotCare	Single Bit Error- Indicates that the ECC decoder detected and fixed a single-bit error.
sbiterr_wdch	Output	1	m_aclk	LEVEL _HIGH	DoNotCare	Single Bit Error- Indicates that the ECC decoder detected and fixed a single-bit error.
wr_data_count _rdch	Output	WR _DATA _COUNT _WIDTH _RDCH	s_aclk	NA	DoNotCare	Write Data Count: This bus indicates the number of words written into the Read Data Channel FIFO.
wr_data_count _wdch	Output	WR _DATA _COUNT _WIDTH _WDCH	s_aclk	NA	DoNotCare	Write Data Count: This bus indicates the number of words written into the Write Data Channel FIFO.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
AXI_ADDR_WIDTH	DECIMAL	1 to 64	32	Defines the width of the ADDR ports, s_axi_araddr, s_axi_awaddr, m_axi_araddr and m_axi_awaddr
AXI_DATA_WIDTH	DECIMAL	8 to 1024	32	Defines the width of the DATA ports, s_axi_rdata, s_axi_wdata, m_axi_rdata and m_axi_wdata NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.

Attribute	Type	Allowed Values	Default	Description
CDC_SYNC_STAGES	DECIMAL	2 to 8	2	Specifies the number of synchronization stages on the CDC path. Applicable only if CLOCKING_MODE = "independent_clock"
CLOCKING_MODE	STRING	"common_clock", "independent_clock"	"common_clock"	Designate whether AXI Memory Mapped FIFO is clocked with a common clock or with independent clocks- • "common_clock"- Common clocking; clock both write and read domain s_aclk • "independent_clock"- Independent clocking; clock write domain with s_aclk and read domain with m_aclk
ECC_MODE_RDCH	STRING	"no_ecc", "en_ecc"	"no_ecc"	• "no_ecc" - Disables ECC • "en_ecc" - Enables both ECC Encoder and Decoder
ECC_MODE_WDCH	STRING	"no_ecc", "en_ecc"	"no_ecc"	• "no_ecc" - Disables ECC • "en_ecc" - Enables both ECC Encoder and Decoder
FIFO_DEPTH_RACH	DECIMAL	16 to 4194304	2048	Defines the AXI Memory Mapped FIFO Write Depth, must be power of two NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
FIFO_DEPTH_RDCH	DECIMAL	16 to 4194304	2048	Defines the AXI Memory Mapped FIFO Write Depth, must be power of two NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
FIFO_DEPTH_WACH	DECIMAL	16 to 4194304	2048	Defines the AXI Memory Mapped FIFO Write Depth, must be power of two NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
FIFO_DEPTH_WDCH	DECIMAL	16 to 4194304	2048	Defines the AXI Memory Mapped FIFO Write Depth, must be power of two NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
FIFO_DEPTH_WRCH	DECIMAL	16 to 4194304	2048	Defines the AXI Memory Mapped FIFO Write Depth, must be power of two NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.

Attribute	Type	Allowed Values	Default	Description
FIFO_MEMORY_TYPE_RACH	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the fifo memory primitive (resource type) to use- • "auto"- Allow Vivado Synthesis to choose • "block"- Block RAM FIFO • "distributed"- Distributed RAM FIFO • "ultra"- URAM FIFO NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with FIFO_MEMORY_TYPE_RACH set to "auto".
FIFO_MEMORY_TYPE_RDCH	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the fifo memory primitive (resource type) to use- • "auto"- Allow Vivado Synthesis to choose • "block"- Block RAM FIFO • "distributed"- Distributed RAM FIFO • "ultra"- URAM FIFO NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with FIFO_MEMORY_TYPE_RDCH set to "auto".
FIFO_MEMORY_TYPE_WACH	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the fifo memory primitive (resource type) to use- • "auto"- Allow Vivado Synthesis to choose • "block"- Block RAM FIFO • "distributed"- Distributed RAM FIFO • "ultra"- URAM FIFO NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with FIFO_MEMORY_TYPE_WACH set to "auto".
FIFO_MEMORY_TYPE_WDCH	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the fifo memory primitive (resource type) to use- • "auto"- Allow Vivado Synthesis to choose • "block"- Block RAM FIFO • "distributed"- Distributed RAM FIFO • "ultra"- URAM FIFO NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with FIFO_MEMORY_TYPE_WDCH set to "auto".

Attribute	Type	Allowed Values	Default	Description
FIFO_MEMORY_TYPE _WRCH	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the fifo memory primitive (resource type) to use- • "auto"- Allow Vivado Synthesis to choose • "block"- Block RAM FIFO • "distributed"- Distributed RAM FIFO • "ultra"- URAM FIFO NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with FIFO_MEMORY_TYPE_WRCH set to "auto".
PROG_EMPTY _THRESH_RDCH	DECIMAL	5 to 4194301	10	Specifies the minimum number of read words in the FIFO at or below which prog_empty is asserted. • Min_Value = 5 • Max_Value = FIFO_WRITE_DEPTH - 5 NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.
PROG_EMPTY _THRESH_WDCH	DECIMAL	5 to 4194301	10	Specifies the minimum number of read words in the FIFO at or below which prog_empty is asserted. • Min_Value = 5 • Max_Value = FIFO_WRITE_DEPTH - 5 NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.
PROG_FULL_THRESH _RDCH	DECIMAL	5 to 4194301	10	Specifies the maximum number of write words in the FIFO at or above which prog_full is asserted. • Min_Value = 5 + CDC_SYNC_STAGES • Max_Value = FIFO_WRITE_DEPTH - 5 NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.

Attribute	Type	Allowed Values	Default	Description
PROG_FULL_THRESH _WDCH	DECIMAL	5 to 4194301	10	Specifies the maximum number of write words in the FIFO at or above which prog_full is asserted. • Min_Value = 5 + CDC_SYNC_STAGES • Max_Value = FIFO_WRITE_DEPTH - 5 NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.
RD_DATA_COUNT _WIDTH_RDCH	DECIMAL	1 to 23	1	Specifies the width of rd_data_count_rdch. To reflect the correct value, the width should be log2(FIFO_DEPTH)+1.
RD_DATA_COUNT _WIDTH_WDCH	DECIMAL	1 to 23	1	Specifies the width of rd_data_count_wdch. To reflect the correct value, the width should be log2(FIFO_DEPTH)+1.
SIM_ASSERT_CHK	DECIMAL	0 to 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.
USE_ADV_FEATURES _RDCH	STRING	String	"1000"	Enables rd_data_count_rdch, prog_empty_rdch, wr_data_count_rdch, prog_full_rdch sideband signals. • Setting USE_ADV_FEATURES_RDCH[1] to 1 enables prog_full_rdch flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES_RDCH[2] to 1 enables wr_data_count_rdch; Default value of this bit is 0 • Setting USE_ADV_FEATURES_RDCH[9] to 1 enables prog_empty_rdch flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES_RDCH[10] to 1 enables rd_data_count_rdch; Default value of this bit is 0

Attribute	Type	Allowed Values	Default	Description
USE_ADV_FEATURES _WDCH	STRING	String	"1000"	Enables rd_data_count_wdch, prog_empty_wdch, wr_data_count_wdch, prog_full_wdch sideband signals. • Setting USE_ADV_FEATURES_WDCH[1] to 1 enables prog_full_wdch flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES_WDCH[2] to 1 enables wr_data_count_wdch; Default value of this bit is 0 • Setting USE_ADV_FEATURES_WDCH[9] to 1 enables prog_empty_wdch flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES_WDCH[10] to 1 enables rd_data_count_wdch; Default value of this bit is 0
WR_DATA_COUNT _WIDTH_RDCH	DECIMAL	1 to 23	1	Specifies the width of wr_data_count_rdch. To reflect the correct value, the width should be log2(FIFO_DEPTH)+1.
WR_DATA_COUNT _WIDTH_WDCH	DECIMAL	1 to 23	1	Specifies the width of wr_data_count_wdch. To reflect the correct value, the width should be log2(FIFO_DEPTH)+1.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_fifo_axil: AXI Memory Mapped (AXI Lite) FIFO
-- Xilinx Parameterized Macro, version 2020.1

xpm_fifo_axil_inst : xpm_fifo_axil
generic map (
   AXI_ADDR_WIDTH => 32,            -- DECIMAL
   AXI_DATA_WIDTH => 32,            -- DECIMAL
   CDC_SYNC_STAGES => 2,            -- DECIMAL
   CLOCKING_MODE => "common_clock", -- String
   ECC_MODE_RDCH => "no_ecc",       -- String
   ECC_MODE_WDCH => "no_ecc",       -- String
   FIFO_DEPTH_RACH => 2048,         -- DECIMAL
   FIFO_DEPTH_RDCH => 2048,         -- DECIMAL
   FIFO_DEPTH_WACH => 2048,         -- DECIMAL
   FIFO_DEPTH_WDCH => 2048,         -- DECIMAL
   FIFO_DEPTH_WRCH => 2048,         -- DECIMAL
   FIFO_MEMORY_TYPE_RACH => "auto", -- String
   FIFO_MEMORY_TYPE_RDCH => "auto", -- String
   FIFO_MEMORY_TYPE_WACH => "auto", -- String
   FIFO_MEMORY_TYPE_WDCH => "auto", -- String
   FIFO_MEMORY_TYPE_WRCH => "auto", -- String
   PROG_EMPTY_THRESH_RDCH => 10,    -- DECIMAL
   PROG_EMPTY_THRESH_WDCH => 10,    -- DECIMAL
   PROG_FULL_THRESH_RDCH => 10,     -- DECIMAL
   PROG_FULL_THRESH_WDCH => 10,     -- DECIMAL
   RD_DATA_COUNT_WIDTH_RDCH => 1,   -- DECIMAL
   RD_DATA_COUNT_WIDTH_WDCH => 1,   -- DECIMAL
   SIM_ASSERT_CHK => 0,             -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
```

```
   USE_ADV_FEATURES_RDCH => "1000", -- String
   USE_ADV_FEATURES_WDCH => "1000", -- String
   WR_DATA_COUNT_WIDTH_RDCH => 1,   -- DECIMAL
   WR_DATA_COUNT_WIDTH_WDCH => 1    -- DECIMAL
)
port map (
   dbiterr_rdch => dbiterr_rdch,            -- 1-bit output: Double Bit Error- Indicates that the ECC
                                            -- decoder detected a double-bit error and data in the FIFO
                                                -- core is corrupted.

   dbiterr_wdch => dbiterr_wdch,            -- 1-bit output: Double Bit Error- Indicates that the ECC
                                            -- decoder detected a double-bit error and data in the FIFO
                                                -- core is corrupted.

   m_axi_araddr => m_axi_araddr,            -- AXI_ADDR_WIDTH-bit output: ARADDR: The read address bus
                                            -- gives the initial address of a read burst transaction. Only
                                            -- the start address of the burst is provided and the control
                                            -- signals that are issued alongside the address detail how the
                                            -- address is calculated for the remaining transfers in the
                                                -- burst.

   m_axi_arprot => m_axi_arprot,            -- 2-bit output: ARPROT: Indicates the normal, privileged, or
                                            -- secure protection level of the transaction and whether the
                                            -- transaction is a data access or an instruction access.

   m_axi_arvalid => m_axi_arvalid,          -- 1-bit output: ARVALID: Indicates that the master is driving
                                            -- a valid transfer. A transfer takes place when both ARVALID
                                                -- and ARREADY are asserted

   m_axi_awaddr => m_axi_awaddr,            -- AXI_ADDR_WIDTH-bit output: AWADDR: The write address bus
                                            -- gives the address of the first transfer in a write burst
                                            -- transaction. The associated control signals are used to
                                            -- determine the addresses of the remaining transfers in the
                                                -- burst.

   m_axi_awprot => m_axi_awprot,            -- 2-bit output: AWPROT: Indicates the normal, privileged, or
                                            -- secure protection level of the transaction and whether the
                                            -- transaction is a data access or an instruction access.

   m_axi_awvalid => m_axi_awvalid,          -- 1-bit output: AWVALID: Indicates that the master is driving
                                            -- a valid transfer. A transfer takes place when both AWVALID
                                                -- and AWREADY are asserted

   m_axi_bready => m_axi_bready,            -- 1-bit output: BREADY: Indicates that the master can accept a
                                                -- transfer in the current cycle.

   m_axi_rready => m_axi_rready,            -- 1-bit output: RREADY: Indicates that the master can accept a
                                                -- transfer in the current cycle.

   m_axi_wdata => m_axi_wdata,              -- AXI_DATA_WIDTH-bit output: WDATA: The primary payload that
                                            -- is used to provide the data that is passing across the
                                            -- interface. The width of the data payload is an integer
                                                -- number of bytes.

   m_axi_wstrb => m_axi_wstrb,              -- AXI_DATA_WIDTH/8-bit output: WSTRB: The byte qualifier that
                                            -- indicates whether the content of the associated byte of
                                            -- TDATA is processed as a data byte or a position byte. For a
                                            -- 64-bit DATA, bit 0 corresponds to the least significant byte
                                            -- on DATA, and bit 0 corresponds to the least significant byte
                                            -- on DATA, and bit 7 corresponds to the most significant byte.
                                            -- For example: STROBE[0] = 1b, DATA[7:0] is valid STROBE[7] =
                                                -- 0b, DATA[63:56] is not valid

   m_axi_wvalid => m_axi_wvalid,            -- 1-bit output: WVALID: Indicates that the master is driving a
                                            -- valid transfer. A transfer takes place when both WVALID and
                                                -- WREADY are asserted

   prog_empty_rdch => prog_empty_rdch,      -- 1-bit output: Programmable Empty- This signal is asserted
                                            -- when the number of words in the Read Data Channel FIFO is
                                            -- less than or equal to the programmable empty threshold
                                            -- value. It is de-asserted when the number of words in the
                                            -- Read Data Channel FIFO exceeds the programmable empty
                                                -- threshold value.

   prog_empty_wdch => prog_empty_wdch,      -- 1-bit output: Programmable Empty- This signal is asserted
                                            -- when the number of words in the Write Data Channel FIFO is
                                            -- less than or equal to the programmable empty threshold
                                            -- value. It is de-asserted when the number of words in the
                                            -- Write Data Channel FIFO exceeds the programmable empty
```

```
                                        -- threshold value.

prog_full_rdch => prog_full_rdch,       -- 1-bit output: Programmable Full: This signal is asserted
                                        -- when the number of words in the Read Data Channel FIFO is
                                        -- greater than or equal to the programmable full threshold
                                        -- value. It is de-asserted when the number of words in the
                                        -- Read Data Channel FIFO is less than the programmable full
                                            -- threshold value.

prog_full_wdch => prog_full_wdch,       -- 1-bit output: Programmable Full: This signal is asserted
                                        -- when the number of words in the Write Data Channel FIFO is
                                        -- greater than or equal to the programmable full threshold
                                        -- value. It is de-asserted when the number of words in the
                                        -- Write Data Channel FIFO is less than the programmable full
                                            -- threshold value.

rd_data_count_rdch => rd_data_count_rdch, -- RD_DATA_COUNT_WIDTH_RDCH-bit output: Read Data Count- This
                                        -- bus indicates the number of words available for reading in
                                            -- the Read Data Channel FIFO.

rd_data_count_wdch => rd_data_count_wdch, -- RD_DATA_COUNT_WIDTH_WDCH-bit output: Read Data Count- This
                                        -- bus indicates the number of words available for reading in
                                            -- the Write Data Channel FIFO.

s_axi_arready => s_axi_arready,         -- 1-bit output: ARREADY: Indicates that the slave can accept a
                                            -- transfer in the current cycle.

s_axi_awready => s_axi_awready,         -- 1-bit output: AWREADY: Indicates that the slave can accept a
                                            -- transfer in the current cycle.

s_axi_bresp => s_axi_bresp,             -- 2-bit output: BRESP: Write Response. Indicates the status of
                                        -- the write transaction. The allowable responses are OKAY,
                                            -- EXOKAY, SLVERR, and DECERR.

s_axi_bvalid => s_axi_bvalid,           -- 1-bit output: BVALID: Indicates that the master is driving a
                                        -- valid transfer. A transfer takes place when both BVALID and
                                            -- BREADY are asserted

s_axi_rdata => s_axi_rdata,             -- AXI_DATA_WIDTH-bit output: RDATA: The primary payload that
                                        -- is used to provide the data that is passing across the
                                        -- interface. The width of the data payload is an integer
                                            -- number of bytes.

s_axi_rresp => s_axi_rresp,             -- 2-bit output: RRESP: Indicates the status of the read
                                        -- transfer. The allowable responses are OKAY, EXOKAY, SLVERR,
                                            -- and DECERR.

s_axi_rvalid => s_axi_rvalid,           -- 1-bit output: RVALID: Indicates that the master is driving a
                                        -- valid transfer. A transfer takes place when both RVALID and
                                            -- RREADY are asserted

s_axi_wready => s_axi_wready,           -- 1-bit output: WREADY: Indicates that the slave can accept a
                                            -- transfer in the current cycle.

sbiterr_rdch => sbiterr_rdch,           -- 1-bit output: Single Bit Error- Indicates that the ECC
                                            -- decoder detected and fixed a single-bit error.

sbiterr_wdch => sbiterr_wdch,           -- 1-bit output: Single Bit Error- Indicates that the ECC
                                            -- decoder detected and fixed a single-bit error.

wr_data_count_rdch => wr_data_count_rdch, -- WR_DATA_COUNT_WIDTH_RDCH-bit output: Write Data Count: This
                                        -- bus indicates the number of words written into the Read Data
                                            -- Channel FIFO.

wr_data_count_wdch => wr_data_count_wdch, -- WR_DATA_COUNT_WIDTH_WDCH-bit output: Write Data Count: This
                                        -- bus indicates the number of words written into the Write
                                            -- Data Channel FIFO.

injectdbiterr_rdch => injectdbiterr_rdch, -- 1-bit input: Double Bit Error Injection- Injects a double
                                            -- bit error if the ECC feature is used.

injectdbiterr_wdch => injectdbiterr_wdch, -- 1-bit input: Double Bit Error Injection- Injects a double
                                            -- bit error if the ECC feature is used.

injectsbiterr_rdch => injectsbiterr_rdch, -- 1-bit input: Single Bit Error Injection- Injects a single
                                            -- bit error if the ECC feature is used.

injectsbiterr_wdch => injectsbiterr_wdch, -- 1-bit input: Single Bit Error Injection- Injects a single
                                            -- bit error if the ECC feature is used.
```

```
m_aclk => m_aclk,                       -- 1-bit input: Master Interface Clock: All signals on master
                                        -- interface are sampled on the rising edge of this clock.

m_axi_arready => m_axi_arready,         -- 1-bit input: ARREADY: Indicates that the master can accept a
                                            -- transfer in the current cycle.

m_axi_awready => m_axi_awready,         -- 1-bit input: AWREADY: Indicates that the master can accept a
                                            -- transfer in the current cycle.

m_axi_bresp => m_axi_bresp,             -- 2-bit input: BRESP: Write Response. Indicates the status of
                                        -- the write transaction. The allowable responses are OKAY,
                                            -- EXOKAY, SLVERR, and DECERR.

m_axi_bvalid => m_axi_bvalid,           -- 1-bit input: BVALID: Indicates that the master is driving a
                                        -- valid transfer. A transfer takes place when both BVALID and
                                            -- BREADY are asserted

m_axi_rdata => m_axi_rdata,             -- AXI_DATA_WIDTH-bit input: RDATA: The primary payload that is
                                            -- used to provide the data that is passing across the
                                        -- interface. The width of the data payload is an integer
                                            -- number of bytes.

m_axi_rresp => m_axi_rresp,                 -- 2-bit input: RRESP: Indicates the status of the read
                                        -- transfer. The allowable responses are OKAY, EXOKAY, SLVERR,
                                            -- and DECERR.

m_axi_rvalid => m_axi_rvalid,           -- 1-bit input: RVALID: Indicates that the master is driving a
                                        -- valid transfer. A transfer takes place when both RVALID and
                                            -- RREADY are asserted

m_axi_wready => m_axi_wready,           -- 1-bit input: WREADY: Indicates that the master can accept a
                                            -- transfer in the current cycle.

s_aclk => s_aclk,                       -- 1-bit input: Slave Interface Clock: All signals on slave
                                        -- interface are sampled on the rising edge of this clock.

s_aresetn => s_aresetn,                     -- 1-bit input: Active low asynchronous reset.
s_axi_araddr => s_axi_araddr,           -- AXI_ADDR_WIDTH-bit input: ARADDR: The read address bus gives
                                        -- the initial address of a read burst transaction. Only the
                                        -- start address of the burst is provided and the control
                                        -- signals that are issued alongside the address detail how the
                                        -- address is calculated for the remaining transfers in the
                                            -- burst.

s_axi_arprot => s_axi_arprot,           -- 2-bit input: ARPROT: Indicates the normal, privileged, or
                                        -- secure protection level of the transaction and whether the
                                        -- transaction is a data access or an instruction access.

s_axi_arvalid => s_axi_arvalid,         -- 1-bit input: ARVALID: Indicates that the master is driving a
                                        -- valid transfer. A transfer takes place when both ARVALID and
                                            -- ARREADY are asserted

s_axi_awaddr => s_axi_awaddr,           -- AXI_ADDR_WIDTH-bit input: AWADDR: The write address bus
                                        -- gives the address of the first transfer in a write burst
                                        -- transaction. The associated control signals are used to
                                        -- determine the addresses of the remaining transfers in the
                                            -- burst.

s_axi_awprot => s_axi_awprot,           -- 2-bit input: AWPROT: Indicates the normal, privileged, or
                                        -- secure protection level of the transaction and whether the
                                        -- transaction is a data access or an instruction access.

s_axi_awvalid => s_axi_awvalid,         -- 1-bit input: AWVALID: Indicates that the master is driving a
                                        -- valid transfer. A transfer takes place when both AWVALID and
                                            -- AWREADY are asserted

s_axi_bready => s_axi_bready,           -- 1-bit input: BREADY: Indicates that the slave can accept a
                                            -- transfer in the current cycle.

s_axi_rready => s_axi_rready,           -- 1-bit input: RREADY: Indicates that the slave can accept a
                                            -- transfer in the current cycle.

s_axi_wdata => s_axi_wdata,             -- AXI_DATA_WIDTH-bit input: WDATA: The primary payload that is
                                            -- used to provide the data that is passing across the
                                        -- interface. The width of the data payload is an integer
                                            -- number of bytes.

s_axi_wstrb => s_axi_wstrb,             -- AXI_DATA_WIDTH/8-bit input: WSTRB: The byte qualifier that
```

```
                                        -- indicates whether the content of the associated byte of
                                        -- TDATA is processed as a data byte or a position byte. For a
                                        -- 64-bit DATA, bit 0 corresponds to the least significant byte
                                        -- on DATA, and bit 0 corresponds to the least significant byte
                                        -- on DATA, and bit 7 corresponds to the most significant byte.
                                        -- For example: STROBE[0] = 1b, DATA[7:0] is valid STROBE[7] =
                                              -- 0b, DATA[63:56] is not valid

   s_axi_wvalid => s_axi_wvalid         -- 1-bit input: WVALID: Indicates that the master is driving a
                                        -- valid transfer. A transfer takes place when both WVALID and
                                              -- WREADY are asserted

);

-- End of xpm_fifo_axil_inst instantiation
```

Verilog Instantiation Template

```
// xpm_fifo_axil: AXI Memory Mapped (AXI Lite) FIFO
// Xilinx Parameterized Macro, version 2020.1

xpm_fifo_axil #(
   .AXI_ADDR_WIDTH(32),             // DECIMAL
   .AXI_DATA_WIDTH(32),             // DECIMAL
   .CDC_SYNC_STAGES(2),             // DECIMAL
   .CLOCKING_MODE("common_clock"), // String
   .ECC_MODE_RDCH("no_ecc"),        // String
   .ECC_MODE_WDCH("no_ecc"),        // String
   .FIFO_DEPTH_RACH(2048),          // DECIMAL
   .FIFO_DEPTH_RDCH(2048),          // DECIMAL
   .FIFO_DEPTH_WACH(2048),          // DECIMAL
   .FIFO_DEPTH_WDCH(2048),          // DECIMAL
   .FIFO_DEPTH_WRCH(2048),          // DECIMAL
   .FIFO_MEMORY_TYPE_RACH("auto"), // String
   .FIFO_MEMORY_TYPE_RDCH("auto"), // String
   .FIFO_MEMORY_TYPE_WACH("auto"), // String
   .FIFO_MEMORY_TYPE_WDCH("auto"), // String
   .FIFO_MEMORY_TYPE_WRCH("auto"), // String
   .PROG_EMPTY_THRESH_RDCH(10),     // DECIMAL
   .PROG_EMPTY_THRESH_WDCH(10),     // DECIMAL
   .PROG_FULL_THRESH_RDCH(10),      // DECIMAL
   .PROG_FULL_THRESH_WDCH(10),      // DECIMAL
   .RD_DATA_COUNT_WIDTH_RDCH(1),    // DECIMAL
   .RD_DATA_COUNT_WIDTH_WDCH(1),    // DECIMAL
   .SIM_ASSERT_CHK(0),              // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .USE_ADV_FEATURES_RDCH("1000"), // String
   .USE_ADV_FEATURES_WDCH("1000"), // String
   .WR_DATA_COUNT_WIDTH_RDCH(1),    // DECIMAL
   .WR_DATA_COUNT_WIDTH_WDCH(1)     // DECIMAL
)
xpm_fifo_axil_inst (
   .dbiterr_rdch(dbiterr_rdch),             // 1-bit output: Double Bit Error- Indicates that the ECC
                                            // decoder detected a double-bit error and data in the FIFO core
                                                  // is corrupted.

   .dbiterr_wdch(dbiterr_wdch),             // 1-bit output: Double Bit Error- Indicates that the ECC
                                            // decoder detected a double-bit error and data in the FIFO core
                                                  // is corrupted.

   .m_axi_araddr(m_axi_araddr),             // AXI_ADDR_WIDTH-bit output: ARADDR: The read address bus gives
                                            // the initial address of a read burst transaction. Only the
                                            // start address of the burst is provided and the control
                                            // signals that are issued alongside the address detail how the
                                            // address is calculated for the remaining transfers in the
                                                  // burst.

   .m_axi_arprot(m_axi_arprot),             // 2-bit output: ARPROT: Indicates the normal, privileged, or
                                            // secure protection level of the transaction and whether the
                                            // transaction is a data access or an instruction access.

   .m_axi_arvalid(m_axi_arvalid),           // 1-bit output: ARVALID: Indicates that the master is driving a
                                            // valid transfer. A transfer takes place when both ARVALID and
                                                  // ARREADY are asserted

   .m_axi_awaddr(m_axi_awaddr),             // AXI_ADDR_WIDTH-bit output: AWADDR: The write address bus
                                            // gives the address of the first transfer in a write burst
```

```
                                    // transaction. The associated control signals are used to
                                    // determine the addresses of the remaining transfers in the
                                        // burst.

.m_axi_awprot(m_axi_awprot),        // 2-bit output: AWPROT: Indicates the normal, privileged, or
                                    // secure protection level of the transaction and whether the
                                    // transaction is a data access or an instruction access.

.m_axi_awvalid(m_axi_awvalid),      // 1-bit output: AWVALID: Indicates that the master is driving a
                                    // valid transfer. A transfer takes place when both AWVALID and
                                        // AWREADY are asserted

.m_axi_bready(m_axi_bready),        // 1-bit output: BREADY: Indicates that the master can accept a
                                        // transfer in the current cycle.

.m_axi_rready(m_axi_rready),        // 1-bit output: RREADY: Indicates that the master can accept a
                                        // transfer in the current cycle.

.m_axi_wdata(m_axi_wdata),          // AXI_DATA_WIDTH-bit output: WDATA: The primary payload that is
                                        // used to provide the data that is passing across the
                                    // interface. The width of the data payload is an integer number
                                        // of bytes.

.m_axi_wstrb(m_axi_wstrb),          // AXI_DATA_WIDTH/8-bit output: WSTRB: The byte qualifier that
                                    // indicates whether the content of the associated byte of TDATA
                                    // is processed as a data byte or a position byte. For a 64-bit
                                    // DATA, bit 0 corresponds to the least significant byte on
                                    // DATA, and bit 0 corresponds to the least significant byte on
                                    // DATA, and bit 7 corresponds to the most significant byte. For
                                    // example: STROBE[0] = 1b, DATA[7:0] is valid STROBE[7] = 0b,
                                        // DATA[63:56] is not valid

.m_axi_wvalid(m_axi_wvalid),        // 1-bit output: WVALID: Indicates that the master is driving a
                                    // valid transfer. A transfer takes place when both WVALID and
                                        // WREADY are asserted

.prog_empty_rdch(prog_empty_rdch),  // 1-bit output: Programmable Empty- This signal is asserted
                                    // when the number of words in the Read Data Channel FIFO is
                                    // less than or equal to the programmable empty threshold value.
                                    // It is de-asserted when the number of words in the Read Data
                                    // Channel FIFO exceeds the programmable empty threshold value.

.prog_empty_wdch(prog_empty_wdch),  // 1-bit output: Programmable Empty- This signal is asserted
                                    // when the number of words in the Write Data Channel FIFO is
                                    // less than or equal to the programmable empty threshold value.
                                    // It is de-asserted when the number of words in the Write Data
                                    // Channel FIFO exceeds the programmable empty threshold value.

.prog_full_rdch(prog_full_rdch),    // 1-bit output: Programmable Full: This signal is asserted when
                                    // the number of words in the Read Data Channel FIFO is greater
                                    // than or equal to the programmable full threshold value. It is
                                    // de-asserted when the number of words in the Read Data Channel
                                    // FIFO is less than the programmable full threshold value.

.prog_full_wdch(prog_full_wdch),    // 1-bit output: Programmable Full: This signal is asserted when
                                    // the number of words in the Write Data Channel FIFO is greater
                                    // than or equal to the programmable full threshold value. It is
                                    // de-asserted when the number of words in the Write Data
                                    // Channel FIFO is less than the programmable full threshold
                                        // value.

.rd_data_count_rdch(rd_data_count_rdch), // RD_DATA_COUNT_WIDTH_RDCH-bit output: Read Data Count- This
                                    // bus indicates the number of words available for reading in
                                        // the Read Data Channel FIFO.

.rd_data_count_wdch(rd_data_count_wdch), // RD_DATA_COUNT_WIDTH_WDCH-bit output: Read Data Count- This
                                    // bus indicates the number of words available for reading in
                                        // the Write Data Channel FIFO.

.s_axi_arready(s_axi_arready),      // 1-bit output: ARREADY: Indicates that the slave can accept a
                                        // transfer in the current cycle.

.s_axi_awready(s_axi_awready),      // 1-bit output: AWREADY: Indicates that the slave can accept a
                                        // transfer in the current cycle.

.s_axi_bresp(s_axi_bresp),          // 2-bit output: BRESP: Write Response. Indicates the status of
                                    // the write transaction. The allowable responses are OKAY,
                                        // EXOKAY, SLVERR, and DECERR.
```

```
.s_axi_bvalid(s_axi_bvalid),            // 1-bit output: BVALID: Indicates that the master is driving a
                                        // valid transfer. A transfer takes place when both BVALID and
                                           // BREADY are asserted

.s_axi_rdata(s_axi_rdata),              // AXI_DATA_WIDTH-bit output: RDATA: The primary payload that is
                                           // used to provide the data that is passing across the
                                        // interface. The width of the data payload is an integer number
                                           // of bytes.

.s_axi_rresp(s_axi_rresp),                 // 2-bit output: RRESP: Indicates the status of the read
                                        // transfer. The allowable responses are OKAY, EXOKAY, SLVERR,
                                           // and DECERR.

.s_axi_rvalid(s_axi_rvalid),            // 1-bit output: RVALID: Indicates that the master is driving a
                                        // valid transfer. A transfer takes place when both RVALID and
                                           // RREADY are asserted

.s_axi_wready(s_axi_wready),            // 1-bit output: WREADY: Indicates that the slave can accept a
                                           // transfer in the current cycle.

.sbiterr_rdch(sbiterr_rdch),            // 1-bit output: Single Bit Error- Indicates that the ECC
                                           // decoder detected and fixed a single-bit error.

.sbiterr_wdch(sbiterr_wdch),            // 1-bit output: Single Bit Error- Indicates that the ECC
                                           // decoder detected and fixed a single-bit error.

.wr_data_count_rdch(wr_data_count_rdch), // WR_DATA_COUNT_WIDTH_RDCH-bit output: Write Data Count: This
                                        // bus indicates the number of words written into the Read Data
                                           // Channel FIFO.

.wr_data_count_wdch(wr_data_count_wdch), // WR_DATA_COUNT_WIDTH_WDCH-bit output: Write Data Count: This
                                        // bus indicates the number of words written into the Write Data
                                           // Channel FIFO.

.injectdbiterr_rdch(injectdbiterr_rdch), // 1-bit input: Double Bit Error Injection- Injects a double bit
                                           // error if the ECC feature is used.

.injectdbiterr_wdch(injectdbiterr_wdch), // 1-bit input: Double Bit Error Injection- Injects a double bit
                                           // error if the ECC feature is used.

.injectsbiterr_rdch(injectsbiterr_rdch), // 1-bit input: Single Bit Error Injection- Injects a single bit
                                           // error if the ECC feature is used.

.injectsbiterr_wdch(injectsbiterr_wdch), // 1-bit input: Single Bit Error Injection- Injects a single bit
                                           // error if the ECC feature is used.

.m_aclk(m_aclk),                        // 1-bit input: Master Interface Clock: All signals on master
                                        // interface are sampled on the rising edge of this clock.

.m_axi_arready(m_axi_arready),          // 1-bit input: ARREADY: Indicates that the master can accept a
                                           // transfer in the current cycle.

.m_axi_awready(m_axi_awready),          // 1-bit input: AWREADY: Indicates that the master can accept a
                                           // transfer in the current cycle.

.m_axi_bresp(m_axi_bresp),              // 2-bit input: BRESP: Write Response. Indicates the status of
                                        // the write transaction. The allowable responses are OKAY,
                                           // EXOKAY, SLVERR, and DECERR.

.m_axi_bvalid(m_axi_bvalid),            // 1-bit input: BVALID: Indicates that the master is driving a
                                        // valid transfer. A transfer takes place when both BVALID and
                                           // BREADY are asserted

.m_axi_rdata(m_axi_rdata),              // AXI_DATA_WIDTH-bit input: RDATA: The primary payload that is
                                           // used to provide the data that is passing across the
                                        // interface. The width of the data payload is an integer number
                                           // of bytes.

.m_axi_rresp(m_axi_rresp),                 // 2-bit input: RRESP: Indicates the status of the read
                                        // transfer. The allowable responses are OKAY, EXOKAY, SLVERR,
                                           // and DECERR.

.m_axi_rvalid(m_axi_rvalid),            // 1-bit input: RVALID: Indicates that the master is driving a
                                        // valid transfer. A transfer takes place when both RVALID and
                                           // RREADY are asserted

.m_axi_wready(m_axi_wready),            // 1-bit input: WREADY: Indicates that the master can accept a
                                           // transfer in the current cycle.
```

```
   .s_aclk(s_aclk),                        // 1-bit input: Slave Interface Clock: All signals on slave
                                           // interface are sampled on the rising edge of this clock.

   .s_aresetn(s_aresetn),                      // 1-bit input: Active low asynchronous reset.
   .s_axi_araddr(s_axi_araddr),            // AXI_ADDR_WIDTH-bit input: ARADDR: The read address bus gives
                                           // the initial address of a read burst transaction. Only the
                                           // start address of the burst is provided and the control
                                           // signals that are issued alongside the address detail how the
                                           // address is calculated for the remaining transfers in the
                                               // burst.

   .s_axi_arprot(s_axi_arprot),            // 2-bit input: ARPROT: Indicates the normal, privileged, or
                                           // secure protection level of the transaction and whether the
                                           // transaction is a data access or an instruction access.

   .s_axi_arvalid(s_axi_arvalid),          // 1-bit input: ARVALID: Indicates that the master is driving a
                                           // valid transfer. A transfer takes place when both ARVALID and
                                               // ARREADY are asserted

   .s_axi_awaddr(s_axi_awaddr),            // AXI_ADDR_WIDTH-bit input: AWADDR: The write address bus gives
                                               // the address of the first transfer in a write burst
                                           // transaction. The associated control signals are used to
                                           // determine the addresses of the remaining transfers in the
                                               // burst.

   .s_axi_awprot(s_axi_awprot),            // 2-bit input: AWPROT: Indicates the normal, privileged, or
                                           // secure protection level of the transaction and whether the
                                           // transaction is a data access or an instruction access.

   .s_axi_awvalid(s_axi_awvalid),          // 1-bit input: AWVALID: Indicates that the master is driving a
                                           // valid transfer. A transfer takes place when both AWVALID and
                                               // AWREADY are asserted

   .s_axi_bready(s_axi_bready),            // 1-bit input: BREADY: Indicates that the slave can accept a
                                               // transfer in the current cycle.

   .s_axi_rready(s_axi_rready),            // 1-bit input: RREADY: Indicates that the slave can accept a
                                               // transfer in the current cycle.

   .s_axi_wdata(s_axi_wdata),              // AXI_DATA_WIDTH-bit input: WDATA: The primary payload that is
                                               // used to provide the data that is passing across the
                                           // interface. The width of the data payload is an integer number
                                               // of bytes.

   .s_axi_wstrb(s_axi_wstrb),              // AXI_DATA_WIDTH/8-bit input: WSTRB: The byte qualifier that
                                           // indicates whether the content of the associated byte of TDATA
                                           // is processed as a data byte or a position byte. For a 64-bit
                                           // DATA, bit 0 corresponds to the least significant byte on
                                           // DATA, and bit 0 corresponds to the least significant byte on
                                           // DATA, and bit 7 corresponds to the most significant byte. For
                                           // example: STROBE[0] = 1b, DATA[7:0] is valid STROBE[7] = 0b,
                                               // DATA[63:56] is not valid

   .s_axi_wvalid(s_axi_wvalid)             // 1-bit input: WVALID: Indicates that the master is driving a
                                           // valid transfer. A transfer takes place when both WVALID and
                                               // WREADY are asserted

);

// End of xpm_fifo_axil_inst instantiation
```

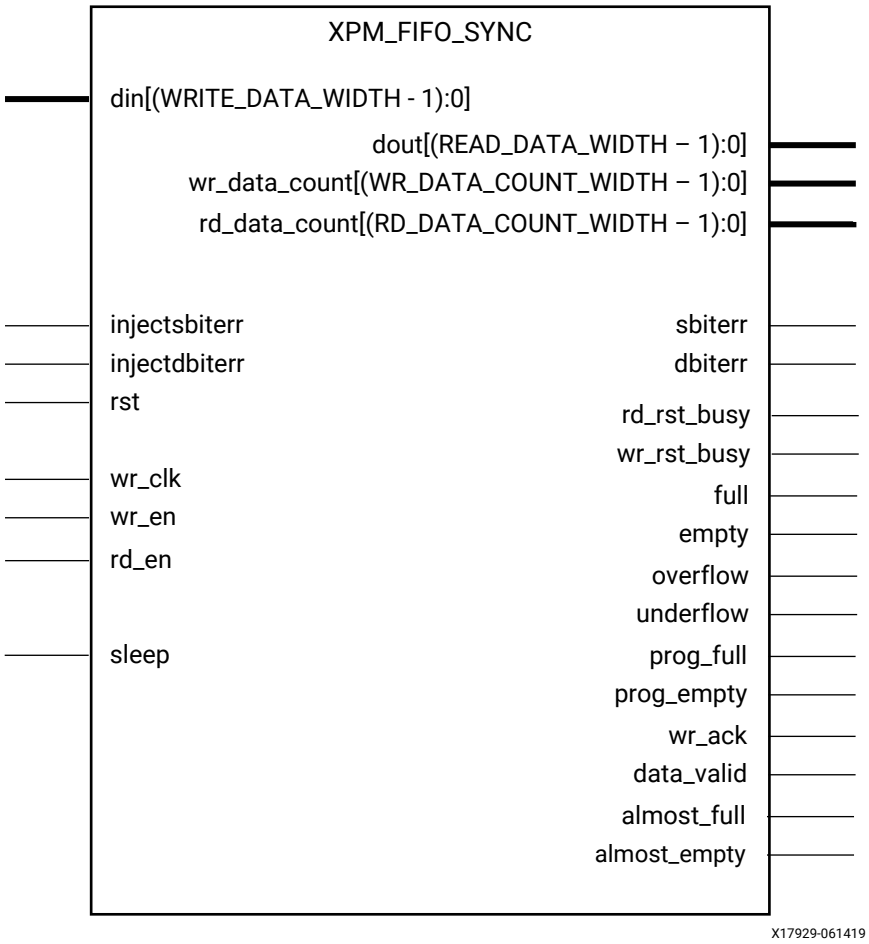
For More Information

- XPM FIFO Testbench File

XPM_FIFO_SYNC

Parameterized Macro: Synchronous FIFO

MACRO_GROUP: XPM
MACRO_SUBGROUP: XPM_FIFO



Introduction

This macro is used to instantiate synchronous FIFO.

The following describes the basic write and read operation of an XPM_FIFO instance.

- All synchronous signals are sensitive to the rising edge of wr_clk, which is assumed to be a buffered and toggling clock signal behaving according to target device and FIFO/memory primitive requirements.
- A write operation is performed when the FIFO is not full and wr_en is asserted on each wr_clk cycle.

- A read operation is performed when the FIFO is not empty and rd_en is asserted on each wr_clk cycle.
- The number of clock cycles required for XPM FIFO to react to dout, full and empty changes depends on the CLOCK_DOMAIN, READ_MODE, and FIFO_READ_LATENCY settings.
  - It might take more than one wr_clk cycle to deassert empty due to write operation (wr_en = 1).
  - It might take more than one wr_clk cycle to present the read data on dout port upon assertion of rd_en.
  - It might take more than one wr_clk cycle to deassert full due to read operation (rd_en = 1).
- All write operations are gated by the value of wr_en and full on the initiating wr_clk cycle.
- All read operations are gated by the value of rd_en and empty on the initiating wr_clk cycle.
- The wr_en input has no effect when full is asserted on the coincident wr_clk cycle.
- The rd_en input has no effect when empty is asserted on the coincident wr_clk cycle.
- Undriven or unknown values provided on module inputs will produce undefined output port behavior.
- wr_en/rd_en should not be toggled when reset (rst) or wr_rst_busy or rd_rst_busy is asserted.
- Assertion/deassertion of prog_full happens only when full is deasserted.
- Assertion/deassertion of prog_empty happens only when empty is deasserted.

Timing Diagrams

Figure 16: **Reset Behavior**

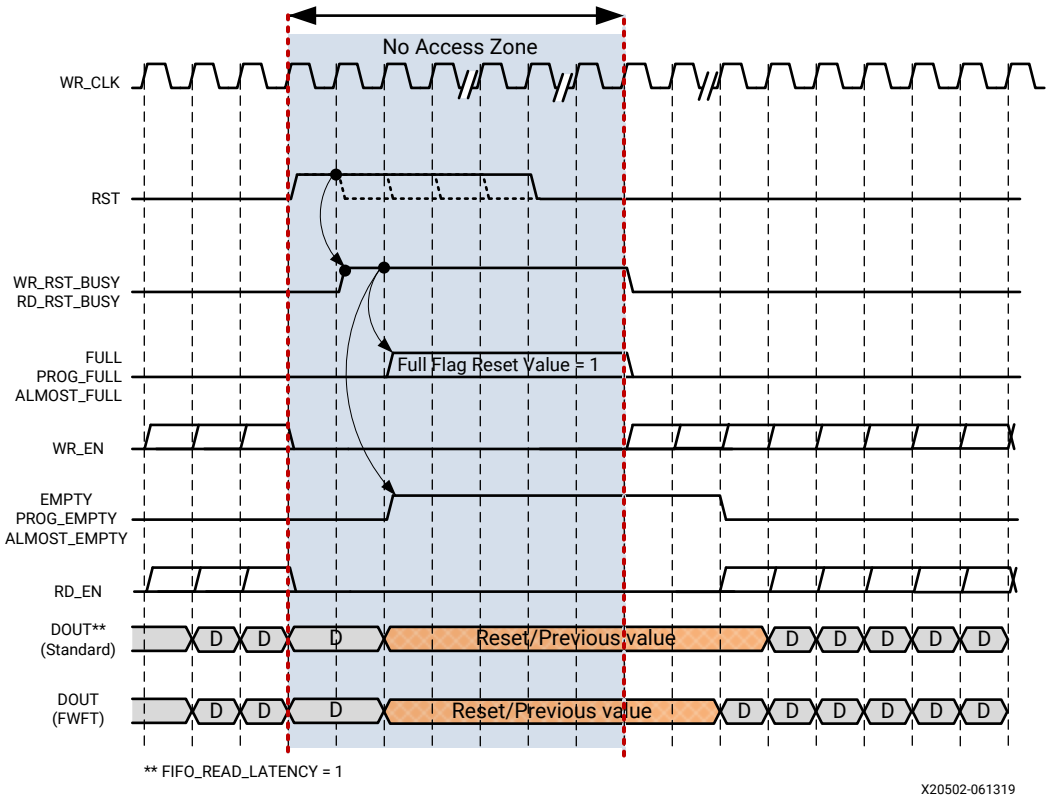


Figure 17: **Standard Write Operation**

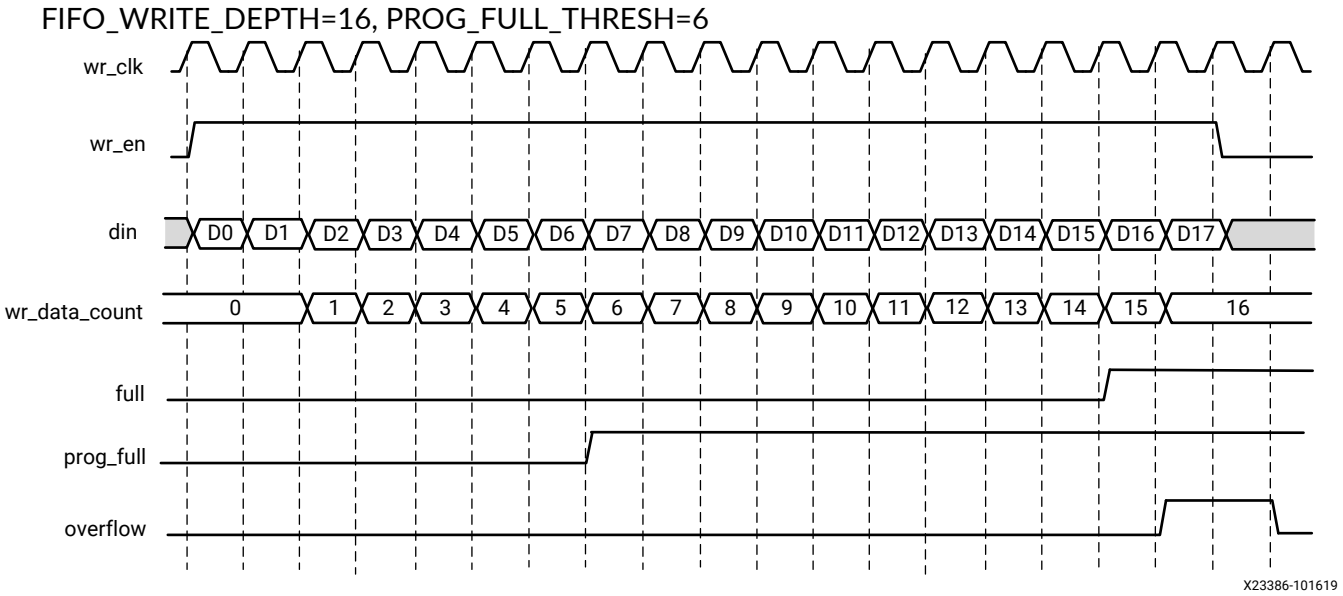


Figure 18: **Standard Read Operation**

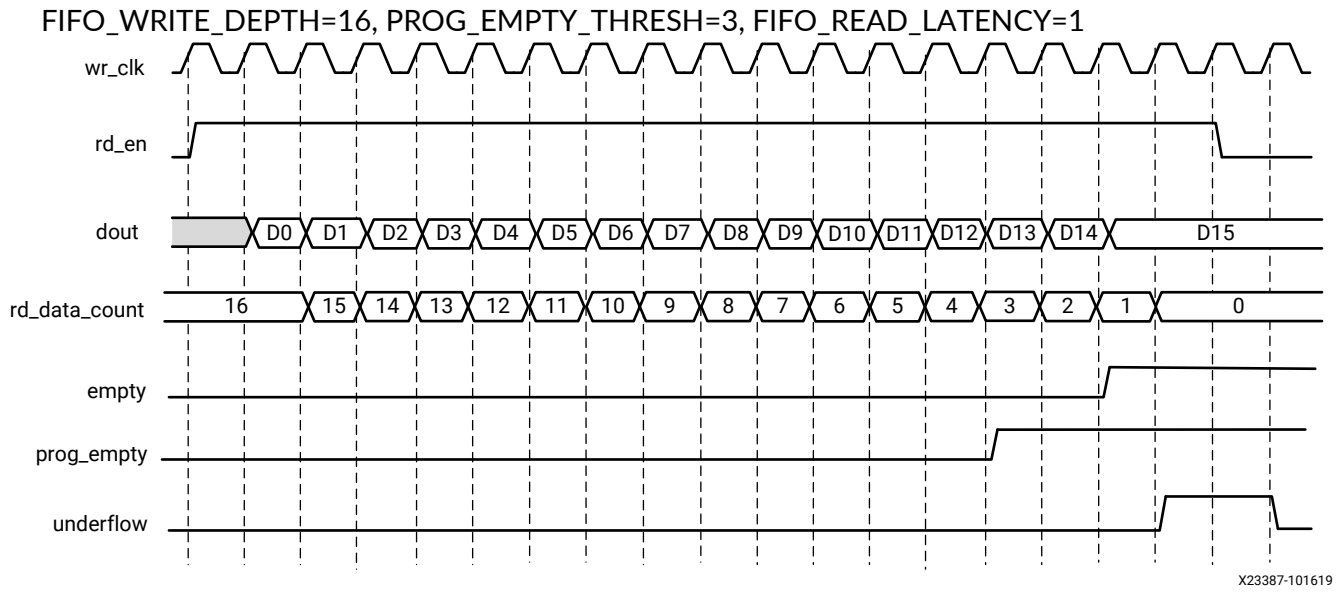


Figure 19: **Standard Read Operation**

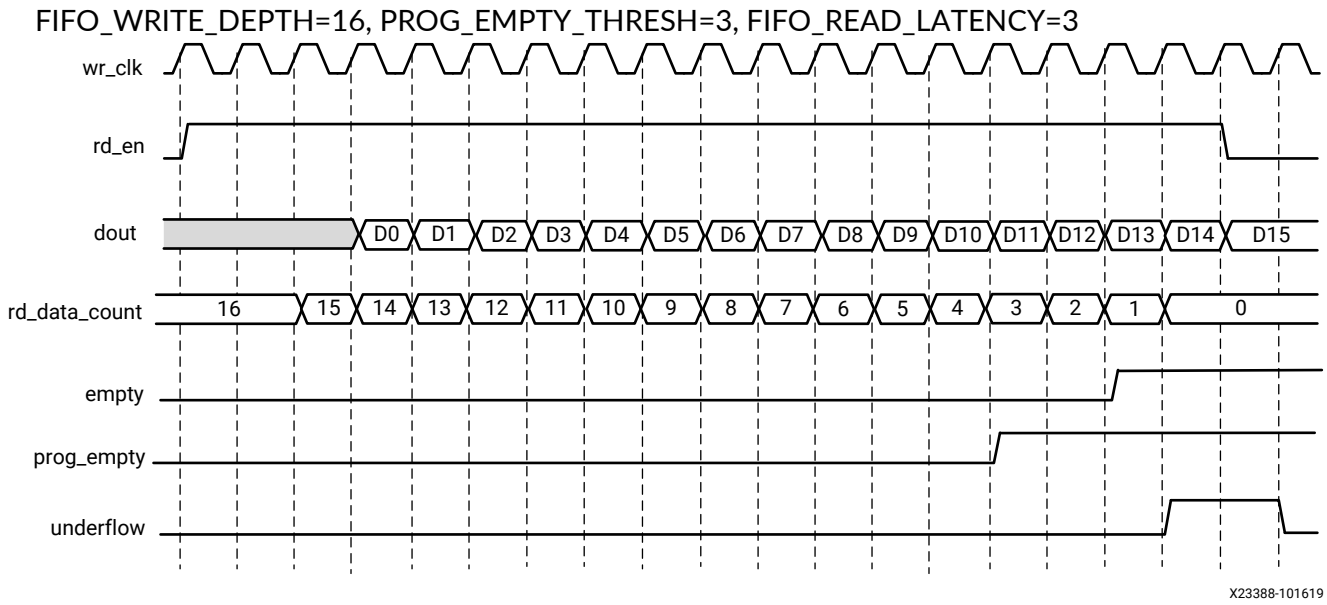


Figure 20: **Write Operation**

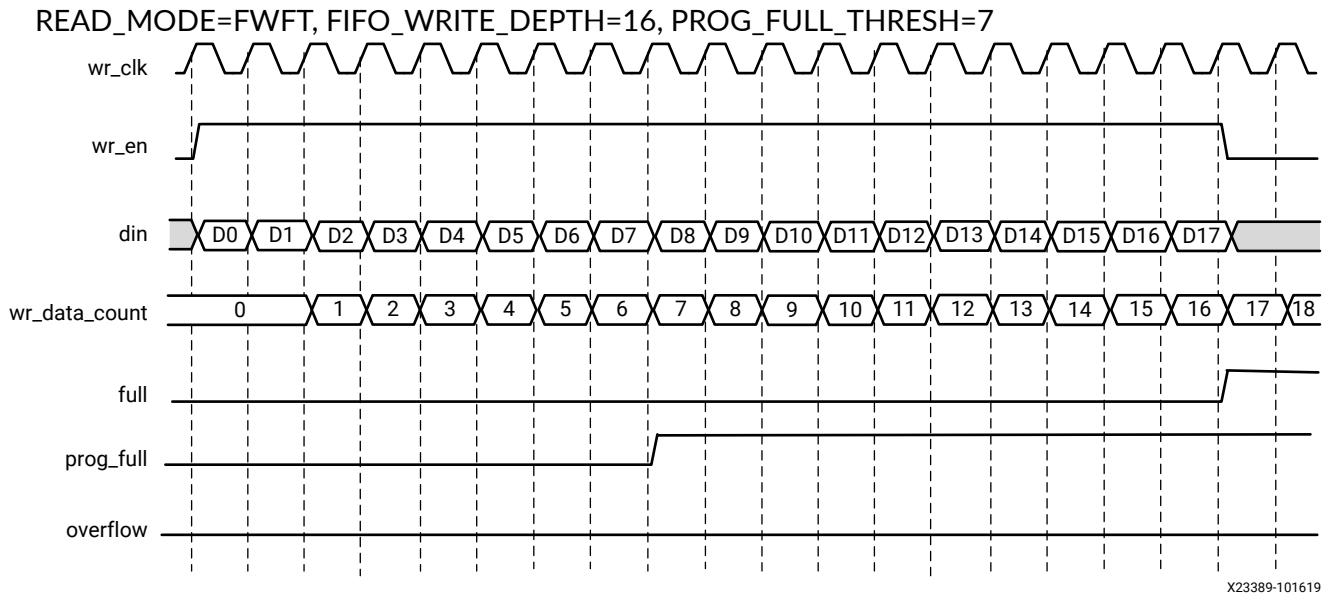


Figure 21: **Read Operation**

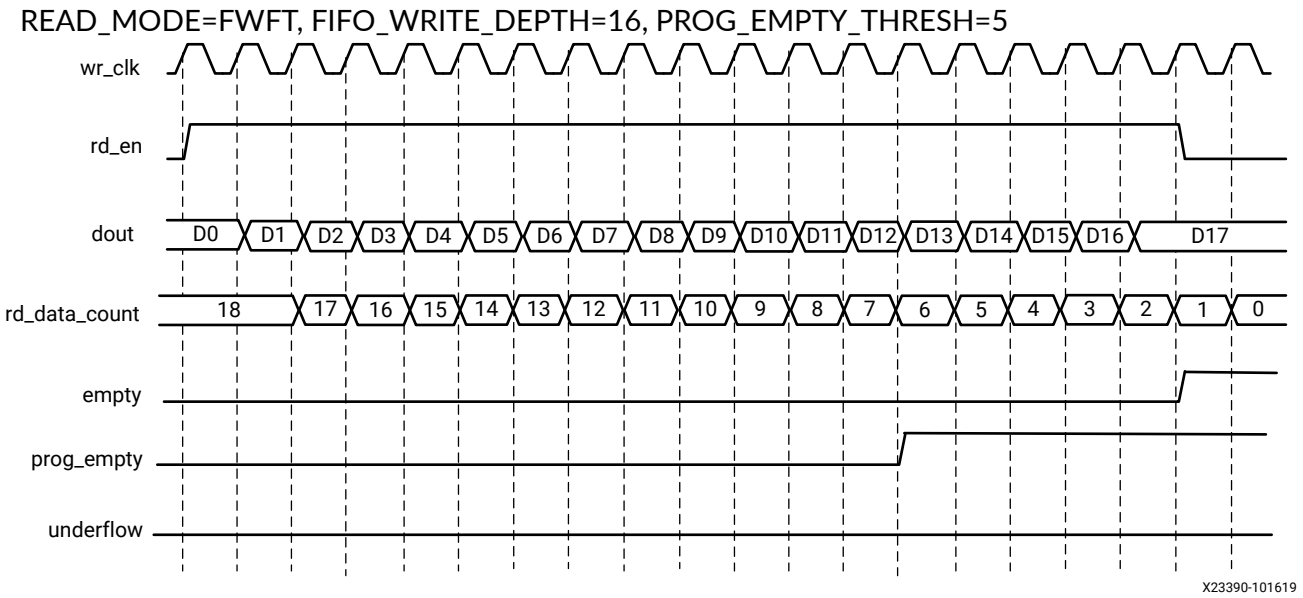


Figure 22: **Standard Write Operation with Empty Deassertion**

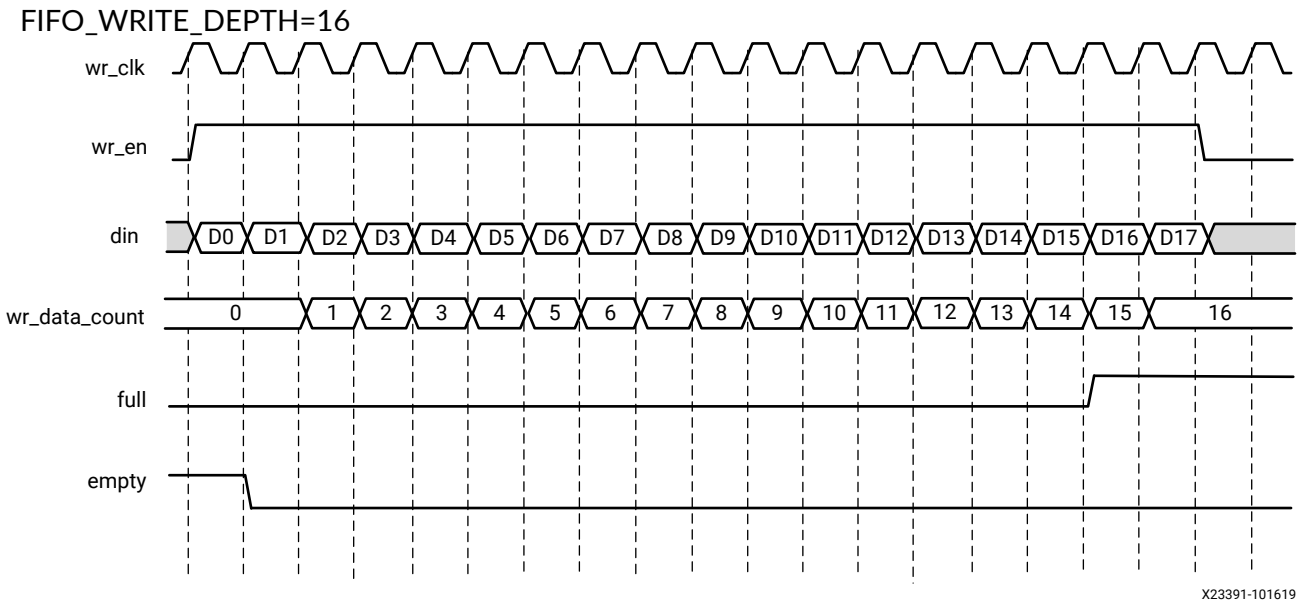
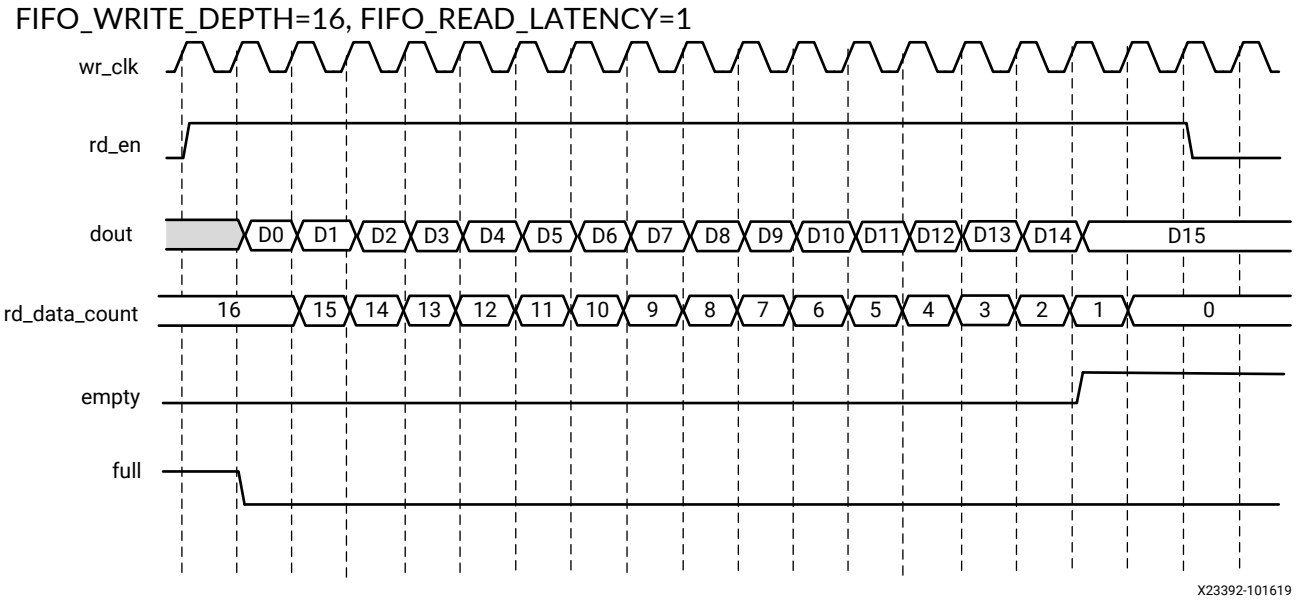


Figure 23: **Standard Read Operation with Full Deassertion**



Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
almost_empty	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Almost Empty : When asserted, this signal indicates that only one more read can be performed before the FIFO goes to empty.
almost_full	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Almost Full: When asserted, this signal indicates that only one more write can be performed before the FIFO is full.
data_valid	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Read Data Valid: When asserted, this signal indicates that valid data is available on the output bus (dout).
dbiterr	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Double Bit Error: Indicates that the ECC decoder detected a double-bit error and data in the FIFO core is corrupted.
din	Input	WRITE _DATA _WIDTH	wr_clk	NA	Active	Write Data: The input data bus used when writing the FIFO.
dout	Output	READ _DATA _WIDTH	wr_clk	NA	Active	Read Data: The output data bus is driven when reading the FIFO.
empty	Output	1	wr_clk	LEVEL _HIGH	Active	Empty Flag: When asserted, this signal indicates that the FIFO is empty. Read requests are ignored when the FIFO is empty, initiating a read while empty is not destructive to the FIFO.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
full	Output	1	wr_clk	LEVEL _HIGH	Active	Full Flag: When asserted, this signal indicates that the FIFO is full. Write requests are ignored when the FIFO is full, initiating a write when the FIFO is full is not destructive to the contents of the FIFO.
injectdbiterr	Input	1	wr_clk	LEVEL _HIGH	0	Double Bit Error Injection: Injects a double bit error if the ECC feature is used on block RAMs or UltraRAM macros.
injectsbiterr	Input	1	wr_clk	LEVEL _HIGH	0	Single Bit Error Injection: Injects a single bit error if the ECC feature is used on block RAMs or UltraRAM macros.
overflow	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Overflow: This signal indicates that a write request (wren) during the prior clock cycle was rejected, because the FIFO is full. Overflowing the FIFO is not destructive to the contents of the FIFO.
prog_empty	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Programmable Empty: This signal is asserted when the number of words in the FIFO is less than or equal to the programmable empty threshold value. It is de-asserted when the number of words in the FIFO exceeds the programmable empty threshold value.
prog_full	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Programmable Full: This signal is asserted when the number of words in the FIFO is greater than or equal to the programmable full threshold value. It is de-asserted when the number of words in the FIFO is less than the programmable full threshold value.
rd_data_count	Output	RD _DATA _COUNT _WIDTH	wr_clk	NA	DoNotCare	Read Data Count: This bus indicates the number of words read from the FIFO.
rd_en	Input	1	wr_clk	LEVEL _HIGH	Active	Read Enable: If the FIFO is not empty, asserting this signal causes data (on dout) to be read from the FIFO. • Must be held active-low when rd_rst_busy is active high.
rd_rst_busy	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Read Reset Busy: Active-High indicator that the FIFO read domain is currently in a reset state.
rst	Input	1	wr_clk	LEVEL _HIGH	Active	Reset: Must be synchronous to wr_clk. The clock(s) can be unstable at the time of applying reset, but reset must be released only after the clock(s) is/are stable.
sbiterr	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Single Bit Error: Indicates that the ECC decoder detected and fixed a single-bit error.
sleep	Input	1	NA	LEVEL _HIGH	0	Dynamic power saving- If sleep is High, the memory/fifo block is in power saving mode.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
underflow	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Underflow: Indicates that the read request (rd_en) during the previous clock cycle was rejected because the FIFO is empty. Under flowing the FIFO is not destructive to the FIFO.
wr_ack	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Write Acknowledge: This signal indicates that a write request (wr_en) during the prior clock cycle is succeeded.
wr_clk	Input	1	NA	EDGE _RISING	Active	Write clock: Used for write operation. wr_clk must be a free running clock.
wr_data_count	Output	WR _DATA _COUNT _WIDTH	wr_clk	NA	DoNotCare	Write Data Count: This bus indicates the number of words written into the FIFO.
wr_en	Input	1	wr_clk	LEVEL _HIGH	Active	Write Enable: If the FIFO is not full, asserting this signal causes data (on din) to be written to the FIFO • Must be held active-low when rst or wr_rst_busy or rd_rst_busy is active high
wr_rst_busy	Output	1	wr_clk	LEVEL _HIGH	DoNotCare	Write Reset Busy: Active-High indicator that the FIFO write domain is currently in a reset state.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DOUT_RESET_VALUE	STRING	String	"0"	Reset value of read data path.
ECC_MODE	STRING	"no_ecc", "en_ecc"	"no_ecc"	• "no_ecc" - Disables ECC • "en_ecc" - Enables both ECC Encoder and Decoder NOTE: ECC_MODE should be "no_ecc" if FIFO_MEMORY_TYPE is set to "auto". Violating this may result incorrect behavior.

Attribute	Type	Allowed Values	Default	Description
FIFO_MEMORY_TYPE	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the fifo memory primitive (resource type) to use- • "auto"- Allow Vivado Synthesis to choose • "block"- Block RAM FIFO • "distributed"- Distributed RAM FIFO • "ultra"- URAM FIFO NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with FIFO_MEMORY_TYPE set to "auto".
FIFO_READ _LATENCY	DECIMAL	0 to 100	1	Number of output register stages in the read data path • If READ_MODE = "fwft", then the only applicable value is 0
FIFO_WRITE_DEPTH	DECIMAL	16 to 4194304	2048	Defines the FIFO Write Depth, must be power of two • In standard READ_MODE, the effective depth = FIFO_WRITE_DEPTH • In First-Word-Fall-Through READ_MODE, the effective depth = FIFO_WRITE_DEPTH+2 NOTE: The maximum FIFO size (width x depth) is limited to 150-Megabits.
FULL_RESET_VALUE	DECIMAL	0 to 1	0	Sets full, almost_full and prog_full to FULL_RESET_VALUE during reset
PROG_EMPTY _THRESH	DECIMAL	3 to 4194304	10	Specifies the minimum number of read words in the FIFO at or below which prog_empty is asserted. • Min_Value = 3 + (READ_MODE_VAL*2) • Max_Value = (FIFO_WRITE_DEPTH-3) - (READ_MODE_VAL*2) If READ_MODE = "std", then READ_MODE_VAL = 0; Otherwise READ_MODE_VAL = 1. NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.

Attribute	Type	Allowed Values	Default	Description
PROG_FULL_THRESH	DECIMAL	3 to 4194301	10	Specifies the maximum number of write words in the FIFO at or above which prog_full is asserted. • Min_Value = 3 + (READ_MODE_VAL*2*(FIFO_WRITE_DEPTH/ FIFO_READ_DEPTH)) • Max_Value = (FIFO_WRITE_DEPTH-3) - (READ_MODE_VAL*2*(FIFO_WRITE_DEPTH/ FIFO_READ_DEPTH)) If READ_MODE = "std", then READ_MODE_VAL = 0; Otherwise READ_MODE_VAL = 1. NOTE: The default threshold value is dependent on default FIFO_WRITE_DEPTH value. If FIFO_WRITE_DEPTH value is changed, ensure the threshold value is within the valid range though the programmable flags are not used.
RD_DATA_COUNT _WIDTH	DECIMAL	1 to 23	1	Specifies the width of rd_data_count. To reflect the correct value, the width should be log2(FIFO_READ_DEPTH)+1. • FIFO_READ_DEPTH = FIFO_WRITE_DEPTH*WRITE_DATA_WIDTH/ READ_DATA_WIDTH
READ_DATA_WIDTH	DECIMAL	1 to 4096	32	Defines the width of the read data port, dout • Write and read width aspect ratio must be 1:1, 1:2, 1:4, 1:8, 8:1, 4:1 and 2:1 • For example, if WRITE_DATA_WIDTH is 32, then the READ_DATA_WIDTH must be 32, 64,128, 256, 16, 8, 4. NOTE: • READ_DATA_WIDTH should be equal to WRITE_DATA_WIDTH if FIFO_MEMORY_TYPE is set to "auto". Violating this may result incorrect behavior. • The maximum FIFO size (width x depth) is limited to 150-Megabits.
READ_MODE	STRING	"std", "fwft"	"std"	• "std"- standard read mode • "fwft"- First-Word-Fall-Through read mode
SIM_ASSERT_CHK	DECIMAL	0 to 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.

Attribute	Type	Allowed Values	Default	Description
USE_ADV_FEATURES	STRING	String	"0707"	Enables data_valid, almost_empty, rd_data_count, prog_empty, underflow, wr_ack, almost_full, wr_data_count, prog_full, overflow features. • Setting USE_ADV_FEATURES[0] to 1 enables overflow flag; Default value of this bit is 1 • Setting USE_ADV_FEATURES[1] to 1 enables prog_full flag; Default value of this bit is 1 • Setting USE_ADV_FEATURES[2] to 1 enables wr_data_count; Default value of this bit is 1 • Setting USE_ADV_FEATURES[3] to 1 enables almost_full flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES[4] to 1 enables wr_ack flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES[8] to 1 enables underflow flag; Default value of this bit is 1 • Setting USE_ADV_FEATURES[9] to 1 enables prog_empty flag; Default value of this bit is 1 • Setting USE_ADV_FEATURES[10] to 1 enables rd_data_count; Default value of this bit is 1 • Setting USE_ADV_FEATURES[11] to 1 enables almost_empty flag; Default value of this bit is 0 • Setting USE_ADV_FEATURES[12] to 1 enables data_valid flag; Default value of this bit is 0
WAKEUP_TIME	DECIMAL	0 to 2	0	• 0 - Disable sleep • 2 - Use Sleep Pin NOTE: WAKEUP_TIME should be 0 if FIFO_MEMORY_TYPE is set to "auto". Violating this may result incorrect behavior.
WR_DATA_COUNT _WIDTH	DECIMAL	1 to 23	1	Specifies the width of wr_data_count. To reflect the correct value, the width should be log2(FIFO_WRITE_DEPTH)+1.

Attribute	Type	Allowed Values	Default	Description
WRITE_DATA_WIDTH	DECIMAL	1 to 4096	32	Defines the width of the write data port, din • Write and read width aspect ratio must be 1:1, 1:2, 1:4, 1:8, 8:1, 4:1 and 2:1 • For example, if WRITE_DATA_WIDTH is 32, then the READ_DATA_WIDTH must be 32, 64,128, 256, 16, 8, 4. NOTE: • WRITE_DATA_WIDTH should be equal to READ_DATA_WIDTH if FIFO_MEMORY_TYPE is set to "auto". Violating this may result incorrect behavior. • The maximum FIFO size (width x depth) is limited to 150-Megabits.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_fifo_sync: Synchronous FIFO
-- Xilinx Parameterized Macro, version 2020.1

xpm_fifo_sync_inst : xpm_fifo_sync
generic map (
   DOUT_RESET_VALUE => "0",     -- String
   ECC_MODE => "no_ecc",        -- String
   FIFO_MEMORY_TYPE => "auto", -- String
   FIFO_READ_LATENCY => 1,      -- DECIMAL
   FIFO_WRITE_DEPTH => 2048,    -- DECIMAL
   FULL_RESET_VALUE => 0,       -- DECIMAL
   PROG_EMPTY_THRESH => 10,     -- DECIMAL
   PROG_FULL_THRESH => 10,      -- DECIMAL
   RD_DATA_COUNT_WIDTH => 1,    -- DECIMAL
   READ_DATA_WIDTH => 32,       -- DECIMAL
   READ_MODE => "std",          -- String
   SIM_ASSERT_CHK => 0,        -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   USE_ADV_FEATURES => "0707", -- String
   WAKEUP_TIME => 0,            -- DECIMAL
   WRITE_DATA_WIDTH => 32,      -- DECIMAL
   WR_DATA_COUNT_WIDTH => 1     -- DECIMAL
)
port map (
   almost_empty => almost_empty,   -- 1-bit output: Almost Empty : When asserted, this signal indicates that
                                   -- only one more read can be performed before the FIFO goes to empty.

   almost_full => almost_full,     -- 1-bit output: Almost Full: When asserted, this signal indicates that
                                        -- only one more write can be performed before the FIFO is full.

   data_valid => data_valid,       -- 1-bit output: Read Data Valid: When asserted, this signal indicates
                                        -- that valid data is available on the output bus (dout).

   dbiterr => dbiterr,                  -- 1-bit output: Double Bit Error: Indicates that the ECC decoder
                                   -- detected a double-bit error and data in the FIFO core is corrupted.

   dout => dout,                   -- READ_DATA_WIDTH-bit output: Read Data: The output data bus is driven
                                        -- when reading the FIFO.
```

```
   empty => empty,                 -- 1-bit output: Empty Flag: When asserted, this signal indicates that
                                   -- the FIFO is empty. Read requests are ignored when the FIFO is empty,
                                      -- initiating a read while empty is not destructive to the FIFO.

   full => full,                   -- 1-bit output: Full Flag: When asserted, this signal indicates that the
                                   -- FIFO is full. Write requests are ignored when the FIFO is full,
                                   -- initiating a write when the FIFO is full is not destructive to the
                                      -- contents of the FIFO.

   overflow => overflow,           -- 1-bit output: Overflow: This signal indicates that a write request
                                   -- (wren) during the prior clock cycle was rejected, because the FIFO is
                                   -- full. Overflowing the FIFO is not destructive to the contents of the
                                      -- FIFO.

   prog_empty => prog_empty,       -- 1-bit output: Programmable Empty: This signal is asserted when the
                                   -- number of words in the FIFO is less than or equal to the programmable
                                   -- empty threshold value. It is de-asserted when the number of words in
                                      -- the FIFO exceeds the programmable empty threshold value.

   prog_full => prog_full,         -- 1-bit output: Programmable Full: This signal is asserted when the
                                      -- number of words in the FIFO is greater than or equal to the
                                   -- programmable full threshold value. It is de-asserted when the number
                                   -- of words in the FIFO is less than the programmable full threshold
                                      -- value.

   rd_data_count => rd_data_count, -- RD_DATA_COUNT_WIDTH-bit output: Read Data Count: This bus indicates
                                      -- the number of words read from the FIFO.

   rd_rst_busy => rd_rst_busy,     -- 1-bit output: Read Reset Busy: Active-High indicator that the FIFO
                                      -- read domain is currently in a reset state.

    sbiterr => sbiterr,               -- 1-bit output: Single Bit Error: Indicates that the ECC decoder
                                      -- detected and fixed a single-bit error.

   underflow => underflow,         -- 1-bit output: Underflow: Indicates that the read request (rd_en)
                                   -- during the previous clock cycle was rejected because the FIFO is
                                      -- empty. Under flowing the FIFO is not destructive to the FIFO.

   wr_ack => wr_ack,               -- 1-bit output: Write Acknowledge: This signal indicates that a write
                                      -- request (wr_en) during the prior clock cycle is succeeded.

   wr_data_count => wr_data_count, -- WR_DATA_COUNT_WIDTH-bit output: Write Data Count: This bus indicates
                                      -- the number of words written into the FIFO.

   wr_rst_busy => wr_rst_busy,     -- 1-bit output: Write Reset Busy: Active-High indicator that the FIFO
                                      -- write domain is currently in a reset state.

   din => din,                     -- WRITE_DATA_WIDTH-bit input: Write Data: The input data bus used when
                                      -- writing the FIFO.

   injectdbiterr => injectdbiterr, -- 1-bit input: Double Bit Error Injection: Injects a double bit error if
                                      -- the ECC feature is used on block RAMs or UltraRAM macros.

   injectsbiterr => injectsbiterr, -- 1-bit input: Single Bit Error Injection: Injects a single bit error if
                                      -- the ECC feature is used on block RAMs or UltraRAM macros.

   rd_en => rd_en,                 -- 1-bit input: Read Enable: If the FIFO is not empty, asserting this
                                   -- signal causes data (on dout) to be read from the FIFO. Must be held
                                      -- active-low when rd_rst_busy is active high.

   rst => rst,                     -- 1-bit input: Reset: Must be synchronous to wr_clk. The clock(s) can be
                                   -- unstable at the time of applying reset, but reset must be released
                                      -- only after the clock(s) is/are stable.

   sleep => sleep,                 -- 1-bit input: Dynamic power saving- If sleep is High, the memory/fifo
                                      -- block is in power saving mode.

   wr_clk => wr_clk,               -- 1-bit input: Write clock: Used for write operation. wr_clk must be a
                                      -- free running clock.

   wr_en => wr_en                  -- 1-bit input: Write Enable: If the FIFO is not full, asserting this
                                   -- signal causes data (on din) to be written to the FIFO Must be held
                                   -- active-low when rst or wr_rst_busy or rd_rst_busy is active high

);

-- End of xpm_fifo_sync_inst instantiation
```

Verilog Instantiation Template

```
// xpm_fifo_sync: Synchronous FIFO
// Xilinx Parameterized Macro, version 2020.1

xpm_fifo_sync #(
   .DOUT_RESET_VALUE("0"),    // String
   .ECC_MODE("no_ecc"),       // String
   .FIFO_MEMORY_TYPE("auto"), // String
   .FIFO_READ_LATENCY(1),     // DECIMAL
   .FIFO_WRITE_DEPTH(2048),   // DECIMAL
   .FULL_RESET_VALUE(0),      // DECIMAL
   .PROG_EMPTY_THRESH(10),    // DECIMAL
   .PROG_FULL_THRESH(10),     // DECIMAL
   .RD_DATA_COUNT_WIDTH(1),   // DECIMAL
   .READ_DATA_WIDTH(32),      // DECIMAL
   .READ_MODE("std"),         // String
   .SIM_ASSERT_CHK(0),        // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .USE_ADV_FEATURES("0707"), // String
   .WAKEUP_TIME(0),           // DECIMAL
   .WRITE_DATA_WIDTH(32),     // DECIMAL
   .WR_DATA_COUNT_WIDTH(1)    // DECIMAL
)
xpm_fifo_sync_inst (
   .almost_empty(almost_empty),   // 1-bit output: Almost Empty : When asserted, this signal indicates that
                                  // only one more read can be performed before the FIFO goes to empty.

   .almost_full(almost_full),     // 1-bit output: Almost Full: When asserted, this signal indicates that
                                     // only one more write can be performed before the FIFO is full.

   .data_valid(data_valid),       // 1-bit output: Read Data Valid: When asserted, this signal indicates
                                     // that valid data is available on the output bus (dout).

   .dbiterr(dbiterr),             // 1-bit output: Double Bit Error: Indicates that the ECC decoder detected
                                     // a double-bit error and data in the FIFO core is corrupted.

   .dout(dout),                   // READ_DATA_WIDTH-bit output: Read Data: The output data bus is driven
                                     // when reading the FIFO.

   .empty(empty),                 // 1-bit output: Empty Flag: When asserted, this signal indicates that the
                                  // FIFO is empty. Read requests are ignored when the FIFO is empty,
                                     // initiating a read while empty is not destructive to the FIFO.

   .full(full),                   // 1-bit output: Full Flag: When asserted, this signal indicates that the
                                     // FIFO is full. Write requests are ignored when the FIFO is full,
                                  // initiating a write when the FIFO is full is not destructive to the
                                     // contents of the FIFO.

   .overflow(overflow),           // 1-bit output: Overflow: This signal indicates that a write request
                                  // (wren) during the prior clock cycle was rejected, because the FIFO is
                                  // full. Overflowing the FIFO is not destructive to the contents of the
                                     // FIFO.

   .prog_empty(prog_empty),       // 1-bit output: Programmable Empty: This signal is asserted when the
                                  // number of words in the FIFO is less than or equal to the programmable
                                  // empty threshold value. It is de-asserted when the number of words in
                                     // the FIFO exceeds the programmable empty threshold value.

   .prog_full(prog_full),         // 1-bit output: Programmable Full: This signal is asserted when the
                                     // number of words in the FIFO is greater than or equal to the
                                  // programmable full threshold value. It is de-asserted when the number of
                                  // words in the FIFO is less than the programmable full threshold value.

   .rd_data_count(rd_data_count), // RD_DATA_COUNT_WIDTH-bit output: Read Data Count: This bus indicates the
                                     // number of words read from the FIFO.

   .rd_rst_busy(rd_rst_busy),     // 1-bit output: Read Reset Busy: Active-High indicator that the FIFO read
                                     // domain is currently in a reset state.

   .sbiterr(sbiterr),             // 1-bit output: Single Bit Error: Indicates that the ECC decoder detected
                                     // and fixed a single-bit error.

   .underflow(underflow),         // 1-bit output: Underflow: Indicates that the read request (rd_en) during
                                  // the previous clock cycle was rejected because the FIFO is empty. Under
                                     // flowing the FIFO is not destructive to the FIFO.

   .wr_ack(wr_ack),               // 1-bit output: Write Acknowledge: This signal indicates that a write
```

```
                                     // request (wr_en) during the prior clock cycle is succeeded.

   .wr_data_count(wr_data_count), // WR_DATA_COUNT_WIDTH-bit output: Write Data Count: This bus indicates
                                     // the number of words written into the FIFO.

   .wr_rst_busy(wr_rst_busy),     // 1-bit output: Write Reset Busy: Active-High indicator that the FIFO
                                     // write domain is currently in a reset state.

   .din(din),                     // WRITE_DATA_WIDTH-bit input: Write Data: The input data bus used when
                                     // writing the FIFO.

   .injectdbiterr(injectdbiterr), // 1-bit input: Double Bit Error Injection: Injects a double bit error if
                                     // the ECC feature is used on block RAMs or UltraRAM macros.

   .injectsbiterr(injectsbiterr), // 1-bit input: Single Bit Error Injection: Injects a single bit error if
                                     // the ECC feature is used on block RAMs or UltraRAM macros.

   .rd_en(rd_en),                 // 1-bit input: Read Enable: If the FIFO is not empty, asserting this
                                  // signal causes data (on dout) to be read from the FIFO. Must be held
                                     // active-low when rd_rst_busy is active high.

   .rst(rst),                     // 1-bit input: Reset: Must be synchronous to wr_clk. The clock(s) can be
                                  // unstable at the time of applying reset, but reset must be released only
                                     // after the clock(s) is/are stable.

   .sleep(sleep),                 // 1-bit input: Dynamic power saving- If sleep is High, the memory/fifo
                                     // block is in power saving mode.

   .wr_clk(wr_clk),               // 1-bit input: Write clock: Used for write operation. wr_clk must be a
                                     // free running clock.

   .wr_en(wr_en)                  // 1-bit input: Write Enable: If the FIFO is not full, asserting this
                                  // signal causes data (on din) to be written to the FIFO Must be held
                                  // active-low when rst or wr_rst_busy or rd_rst_busy is active high

);

// End of xpm_fifo_sync_inst instantiation
```

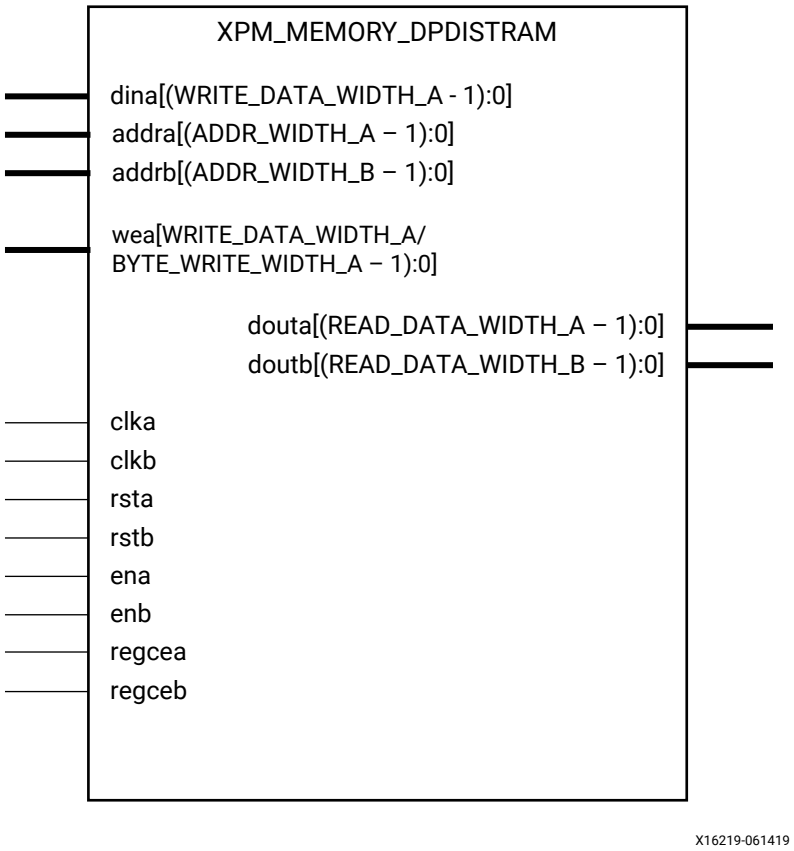
For More Information

- XPM FIFO Testbench File

XPM_MEMORY_DPDISTRAM

Parameterized Macro: Dual Port Distributed RAM

MACRO_GROUP: XPM
MACRO_SUBGROUP: XPM_MEMORY



Introduction

This macro is used to instantiate Dual Port Distributed RAM. Port-A can be used to perform both read and write operations and simultaneously port B can be used to perform read operations from the memory. Write operations are not allowed through port B.

The following describes the basic read and write port usage of an XPM_MEMORY instance. It does not distinguish between ports A and B.

- All synchronous signals are sensitive to the rising edge of clk[a|b], which is assumed to be a buffered and toggling clock signal behaving according to target device and memory primitive requirements.
- A read operation is implicitly performed to address addr[a|b] combinatorially. The data output is registered each clk[a|b] cycle that en[a|b] is asserted.

- Read data appears on the dout[a|b] port READ_LATENCY_[A|B] clk[a|b] cycles after the associated read operation.
- A write operation is explicitly performed, writing dina to address addra, when both ena and wea are asserted on each clka cycle.
- All read and write operations are gated by the value of en[a|b] on the initiating clk[a|b] cycle, regardless of input or output latencies. The addra and wea inputs have no effect when ena is de-asserted on the coincident clka cycle.
- For each clk[a|b] cycle that rst[a|b] is asserted, the final output register is immediately but synchronously reset to READ_RESET_VALUE_[A|B], irrespective of READ_LATENCY_[A|B].
- For each clk[a|b] cycle that regce[a|b] is asserted and rst[a|b] is de-asserted, the final output register captures and outputs the value from the previous pipeline register.
- Undriven or unknown values provided on module inputs will produce undefined memory array and output port behavior.
- When MEMORY_INIT_PARAM is used, the maximum supported memory size 4K bits.

***Note*:**

- When the attribute “CLOCKING_MODE” is set to “common_clock”, all read/write operations to memory through port A and port B are performed on clka. If this attribute is set to “independent_clock”, then read/write operations through port A are performed based on clka, and read/write operations through port B are performed based on `clkb`.
- Writing to an out-of-range address location may overwrite a valid address location when effective address bits match to a physical memory address location.
- set_false_path constraint is needed for the independent clock distributed RAM based memory if the design takes care of avoiding address collision (write address != read address at any given point of time). Set USE_EMBEDDED_CONSTRAINT = 1 if XPM_MEMORY needs to take care of necessary constraints. If USE_EMBEDDED_CONSTRAINT = 0, Vivado may trigger Timing-6 or Timing-7 or both. Alternatively, you can also add the constraint when USE_EMBEDDED_CONSTRAINT = 0. An example of adding this constraint is provided below. If Port-B also has write permissions for an Independent clock configuration, then a similar constraint needs to be added for clkb as well.

```
set_false_path -from [filter [all_fanout -from [get_ports clka]
-flat -endpoints_only] {IS_LEAF}] -through [get_pins -of_objects
[get_cells -hier * -filter {PRIMITIVE_SUBGROUP==LUTRAM ||
PRIMITIVE_SUBGROUP==dram || PRIMITIVE_SUBGROUP==drom}]
-filter {DIRECTION==OUT}]
```

- If "CLOCKING_MODE" is set to "independent_clock", Vivado may trigger a false positive CDC-1 warning and can be ignored.

Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
addra	Input	ADDR_WIDTH_A	clka	NA	Active	Address for port A write and read operations.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
addrb	Input	ADDR _WIDTH _B	clkb	NA	Active	Address for port B write and read operations.
clka	Input	1	NA	EDGE _RISING	Active	Clock signal for port A. Also clocks port B when parameter CLOCKING_MODE is "common_clock".
clkb	Input	1	NA	EDGE _RISING	Active	Clock signal for port B when parameter CLOCKING_MODE is "independent_clock". Unused when parameter CLOCKING_MODE is "common_clock".
dina	Input	WRITE _DATA _WIDTH _A	clka	NA	Active	Data input for port A write operations.
douta	Output	READ _DATA _WIDTH _A	clka	NA	Active	Data output for port A read operations.
doutb	Output	READ _DATA _WIDTH _B	clkb	NA	Active	Data output for port B read operations.
ena	Input	1	clka	LEVEL _HIGH	Active	Memory enable signal for port A. Must be high on clock cycles when read or write operations are initiated. Pipelined internally.
enb	Input	1	clkb	LEVEL _HIGH	Active	Memory enable signal for port B. Must be high on clock cycles when read or write operations are initiated. Pipelined internally.
regcea	Input	1	clka	LEVEL _HIGH	1	Clock Enable for the last register stage on the output data path.
regceb	Input	1	clkb	LEVEL _HIGH	Active	Do not change from the provided value.
rsta	Input	1	clka	LEVEL _HIGH	Active	Reset signal for the final port A output register stage. Synchronously resets output port douta to the value specified by parameter READ_RESET_VALUE_A.
rstb	Input	1	clkb	LEVEL _HIGH	Active	Reset signal for the final port B output register stage. Synchronously resets output port doutb to the value specified by parameter READ_RESET_VALUE_B.
wea	Input	WRITE _DATA _WIDTH _A / BYTE _WRITE _WIDTH _A	clka	LEVEL _HIGH	Active	Write enable vector for port A input data port dina. 1 bit wide when word-wide writes are used. In byte-wide write configurations, each bit controls the writing one byte of dina to address addra. For example, to synchronously write only bits [15-8] of dina when WRITE_DATA_WIDTH_A is 32, wea would be 4'b0010.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ADDR_WIDTH_A	DECIMAL	1 to 20	6	Specify the width of the port A address port addra, in bits. Must be large enough to access the entire memory from port A, i.e. >= $clog2(MEMORY_SIZE/[WRITE|READ]_DATA_WIDTH_A).
ADDR_WIDTH_B	DECIMAL	1 to 20	6	Specify the width of the port B address port addrb, in bits. Must be large enough to access the entire memory from port B, i.e. >= $clog2(MEMORY_SIZE/[WRITE|READ]_DATA_WIDTH_B).
BYTE_WRITE_WIDTH _A	DECIMAL	1 to 4608	32	To enable byte-wide writes on port A, specify the byte width, in bits. • 8- 8-bit byte-wide writes, legal when WRITE_DATA_WIDTH_A is an integer multiple of 8 • 9- 9-bit byte-wide writes, legal when WRITE_DATA_WIDTH_A is an integer multiple of 9 Or to enable word-wide writes on port A, specify the same value as for WRITE_DATA_WIDTH_A.
CLOCKING_MODE	STRING	"common _clock", "independent _clock"	"common _clock"	Designate whether port A and port B are clocked with a common clock or with independent clocks- • "common_clock"- Common clocking; clock both port A and port B with clka • "independent_clock"- Independent clocking; clock port A with clka and port B with clkb

Attribute	Type	Allowed Values	Default	Description
MEMORY_INIT_FILE	STRING	String	"none"	Specify "none" (including quotes) for no memory initialization, or specify the name of a memory initialization file- Enter only the name of the file with .mem extension, including quotes but without path (e.g. "my_file.mem"). File format must be ASCII and consist of only hexadecimal values organized into the specified depth by narrowest data width generic value of the memory. See the Memory File (MEM) section for more information on the syntax. Initialization of memory happens through the file name specified only when parameter MEMORY_INIT_PARAM value is equal to "". When using XPM_MEMORY in a project, add the specified file to the Vivado project as a design source.
MEMORY_INIT _PARAM	STRING	String	"0"	Specify "" or "0" (including quotes) for no memory initialization through parameter, or specify the string containing the hex characters. Enter only hex characters with each location separated by delimiter (,). Parameter format must be ASCII and consist of only hexadecimal values organized into the specified depth by narrowest data width generic value of the memory. For example, if the narrowest data width is 8, and the depth of memory is 8 locations, then the parameter value should be passed as shown below. parameter MEMORY_INIT_PARAM = "AB,CD,EF,1,2,34,56,78" Where "AB" is the 0th location and "78" is the 7th location.
MEMORY _OPTIMIZATION	STRING	"true", "false"	"true"	Specify "true" to enable the optimization of unused memory or bits in the memory structure. Specify "false" to disable the optimization of unused memory or bits in the memory structure
MEMORY_SIZE	DECIMAL	2 to 150994944	2048	Specify the total memory array size, in bits. For example, enter 65536 for a 2kx32 RAM.
MESSAGE_CONTROL	DECIMAL	0 to 1	0	Specify 1 to enable the dynamic message reporting such as collision warnings, and 0 to disable the message reporting
READ_DATA_WIDTH_A	DECIMAL	1 to 4608	32	Specify the width of the port A read data output port douta, in bits. The values of READ_DATA_WIDTH_A and WRITE_DATA_WIDTH_A must be equal.
READ_DATA_WIDTH_B	DECIMAL	1 to 4608	32	Specify the width of the port B read data output port doutb, in bits. The values of READ_DATA_WIDTH_B and WRITE_DATA_WIDTH_B must be equal.

Attribute	Type	Allowed Values	Default	Description
READ_LATENCY_A	DECIMAL	0 to 100	2	Specify the number of register stages in the port A read data pipeline. Read data output to port douta takes this number of clka cycles. To target block memory, a value of 1 or larger is required- 1 causes use of memory latch only; 2 causes use of output register. To target distributed memory, a value of 0 or larger is required- 0 indicates combinatorial output. Values larger than 2 synthesize additional flip-flops that are not retimed into memory primitives.
READ_LATENCY_B	DECIMAL	0 to 100	2	Specify the number of register stages in the port B read data pipeline. Read data output to port doutb takes this number of clkb cycles (clka when CLOCKING_MODE is "common_clock"). To target block memory, a value of 1 or larger is required- 1 causes use of memory latch only; 2 causes use of output register. To target distributed memory, a value of 0 or larger is required- 0 indicates combinatorial output. Values larger than 2 synthesize additional flip-flops that are not retimed into memory primitives.
READ_RESET_VALUE _A	STRING	String	"0"	Specify the reset value of the port A final output register stage in response to rsta input port is assertion. The value mentioned must be accomodated in READ_DATA_WIDTH_A number of bits.
READ_RESET_VALUE _B	STRING	String	"0"	Specify the reset value of the port B final output register stage in response to rstb input port is assertion. The value mentioned must be accomodated in READ_DATA_WIDTH_B number of bits.
RST_MODE_A	STRING	"SYNC", "ASYNC"	"SYNC"	Describes the behaviour of the reset • "SYNC" - when reset is applied, synchronously resets output port douta to the value specified by parameter READ_RESET_VALUE_A • "ASYNC" - when reset is applied, asynchronously resets output port douta to zero
RST_MODE_B	STRING	"SYNC", "ASYNC"	"SYNC"	Describes the behaviour of the reset • "SYNC" - when reset is applied, synchronously resets output port doutb to the value specified by parameter READ_RESET_VALUE_B • "ASYNC" - when reset is applied, asynchronously resets output port doutb to zero
SIM_ASSERT_CHK	DECIMAL	0 to 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.

Attribute	Type	Allowed Values	Default	Description
USE_EMBEDDED _CONSTRAINT	DECIMAL	0 to 1	0	Specify 1 to enable the set_false_path constraint addition between clka of Distributed RAM and doutb_reg on clkb
USE_MEM_INIT	DECIMAL	0 to 1	1	Specify 1 to enable the generation of below message and 0 to disable generation of the following message completely. "INFO - MEMORY_INIT_FILE and MEMORY_INIT_PARAM together specifies no memory initialization. Initial memory contents will be all 0s." NOTE: This message gets generated only when there is no Memory Initialization specified either through file or Parameter.
WRITE_DATA_WIDTH_A	DECIMAL	1 to 4608	32	Specify the width of the port A write data input port dina, in bits. The values of WRITE_DATA_WIDTH_A and READ_DATA_WIDTH_A must be equal.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_memory_dpdistram: Dual Port Distributed RAM
-- Xilinx Parameterized Macro, version 2020.1

xpm_memory_dpdistram_inst : xpm_memory_dpdistram
generic map (
   ADDR_WIDTH_A => 6,               -- DECIMAL
   ADDR_WIDTH_B => 6,               -- DECIMAL
   BYTE_WRITE_WIDTH_A => 32,        -- DECIMAL
   CLOCKING_MODE => "common_clock", -- String
   MEMORY_INIT_FILE => "none",      -- String
   MEMORY_INIT_PARAM => "0",        -- String
   MEMORY_OPTIMIZATION => "true",   -- String
   MEMORY_SIZE => 2048,             -- DECIMAL
   MESSAGE_CONTROL => 0,            -- DECIMAL
   READ_DATA_WIDTH_A => 32,         -- DECIMAL
   READ_DATA_WIDTH_B => 32,         -- DECIMAL
   READ_LATENCY_A => 2,             -- DECIMAL
   READ_LATENCY_B => 2,             -- DECIMAL
   READ_RESET_VALUE_A => "0",       -- String
   READ_RESET_VALUE_B => "0",       -- String
   RST_MODE_A => "SYNC",            -- String
   RST_MODE_B => "SYNC",            -- String
   SIM_ASSERT_CHK => 0,             -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   USE_EMBEDDED_CONSTRAINT => 0,    -- DECIMAL
   USE_MEM_INIT => 1,               -- DECIMAL
   WRITE_DATA_WIDTH_A => 32         -- DECIMAL
)
port map (
   douta => douta,   -- READ_DATA_WIDTH_A-bit output: Data output for port A read operations.
   doutb => doutb,   -- READ_DATA_WIDTH_B-bit output: Data output for port B read operations.
   addra => addra,   -- ADDR_WIDTH_A-bit input: Address for port A write and read operations.
   addrb => addrb,   -- ADDR_WIDTH_B-bit input: Address for port B write and read operations.
   clka => clka,     -- 1-bit input: Clock signal for port A. Also clocks port B when parameter
                     -- CLOCKING_MODE is "common_clock".

   clkb => clkb,     -- 1-bit input: Clock signal for port B when parameter CLOCKING_MODE is
                     -- "independent_clock". Unused when parameter CLOCKING_MODE is "common_clock".

   dina => dina,     -- WRITE_DATA_WIDTH_A-bit input: Data input for port A write operations.
```

```
   ena => ena,       -- 1-bit input: Memory enable signal for port A. Must be high on clock cycles when read
                        -- or write operations are initiated. Pipelined internally.

   enb => enb,       -- 1-bit input: Memory enable signal for port B. Must be high on clock cycles when read
                        -- or write operations are initiated. Pipelined internally.

   regcea => regcea, -- 1-bit input: Clock Enable for the last register stage on the output data path.
    regceb => regceb, -- 1-bit input: Do not change from the provided value.
   rsta => rsta,     -- 1-bit input: Reset signal for the final port A output register stage. Synchronously
                     -- resets output port douta to the value specified by parameter READ_RESET_VALUE_A.

   rstb => rstb,     -- 1-bit input: Reset signal for the final port B output register stage. Synchronously
                     -- resets output port doutb to the value specified by parameter READ_RESET_VALUE_B.

   wea => wea        -- WRITE_DATA_WIDTH_A/BYTE_WRITE_WIDTH_A-bit input: Write enable vector for port A
                     -- input data port dina. 1 bit wide when word-wide writes are used. In byte-wide write
                     -- configurations, each bit controls the writing one byte of dina to address addra. For
                     -- example, to synchronously write only bits [15-8] of dina when WRITE_DATA_WIDTH_A is
                        -- 32, wea would be 4'b0010.

);

-- End of xpm_memory_dpdistram_inst instantiation
```

Verilog Instantiation Template

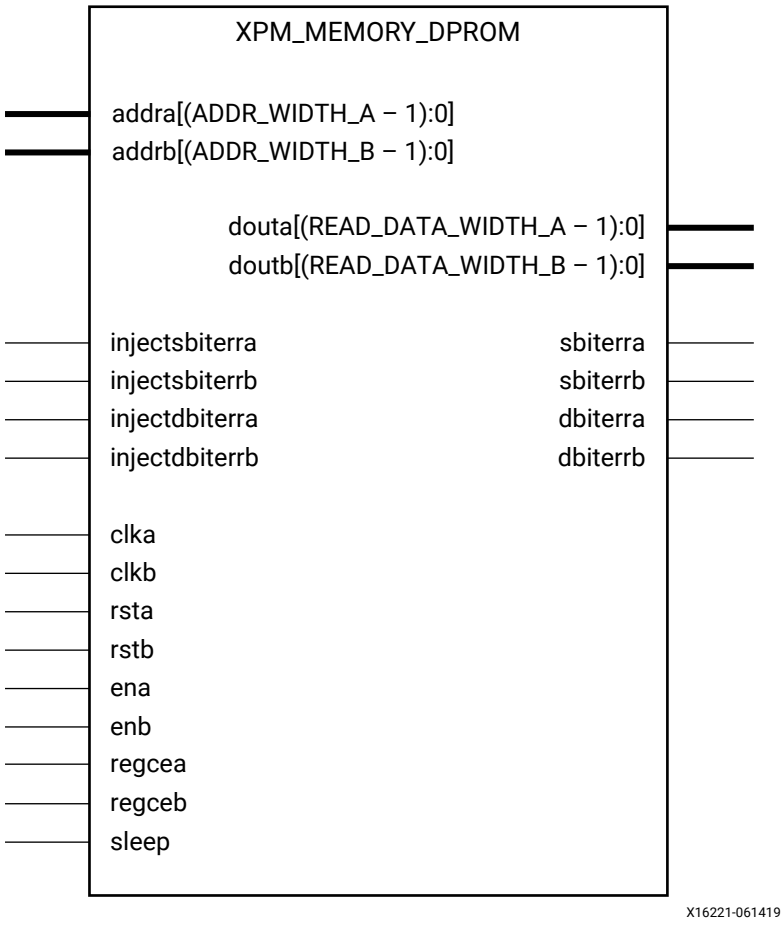
```
// xpm_memory_dpdistram: Dual Port Distributed RAM
// Xilinx Parameterized Macro, version 2020.1

xpm_memory_dpdistram #(
   .ADDR_WIDTH_A(6),               // DECIMAL
   .ADDR_WIDTH_B(6),               // DECIMAL
   .BYTE_WRITE_WIDTH_A(32),        // DECIMAL
   .CLOCKING_MODE("common_clock"), // String
   .MEMORY_INIT_FILE("none"),      // String
   .MEMORY_INIT_PARAM("0"),        // String
   .MEMORY_OPTIMIZATION("true"),   // String
   .MEMORY_SIZE(2048),             // DECIMAL
   .MESSAGE_CONTROL(0),            // DECIMAL
   .READ_DATA_WIDTH_A(32),         // DECIMAL
   .READ_DATA_WIDTH_B(32),         // DECIMAL
   .READ_LATENCY_A(2),             // DECIMAL
   .READ_LATENCY_B(2),             // DECIMAL
   .READ_RESET_VALUE_A("0"),       // String
   .READ_RESET_VALUE_B("0"),       // String
   .RST_MODE_A("SYNC"),            // String
   .RST_MODE_B("SYNC"),            // String
   .SIM_ASSERT_CHK(0),          // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .USE_EMBEDDED_CONSTRAINT(0),    // DECIMAL
   .USE_MEM_INIT(1),               // DECIMAL
   .WRITE_DATA_WIDTH_A(32)         // DECIMAL
)
xpm_memory_dpdistram_inst (
   .douta(douta),   // READ_DATA_WIDTH_A-bit output: Data output for port A read operations.
   .doutb(doutb),   // READ_DATA_WIDTH_B-bit output: Data output for port B read operations.
   .addra(addra),   // ADDR_WIDTH_A-bit input: Address for port A write and read operations.
   .addrb(addrb),   // ADDR_WIDTH_B-bit input: Address for port B write and read operations.
   .clka(clka),     // 1-bit input: Clock signal for port A. Also clocks port B when parameter CLOCKING_MODE
                       // is "common_clock".

   .clkb(clkb),        // 1-bit input: Clock signal for port B when parameter CLOCKING_MODE is
                       // "independent_clock". Unused when parameter CLOCKING_MODE is "common_clock".

   .dina(dina),        // WRITE_DATA_WIDTH_A-bit input: Data input for port A write operations.
   .ena(ena),       // 1-bit input: Memory enable signal for port A. Must be high on clock cycles when read
                       // or write operations are initiated. Pipelined internally.

   .enb(enb),       // 1-bit input: Memory enable signal for port B. Must be high on clock cycles when read
                       // or write operations are initiated. Pipelined internally.

   .regcea(regcea), // 1-bit input: Clock Enable for the last register stage on the output data path.
    .regceb(regceb), // 1-bit input: Do not change from the provided value.
   .rsta(rsta),     // 1-bit input: Reset signal for the final port A output register stage. Synchronously
                    // resets output port douta to the value specified by parameter READ_RESET_VALUE_A.

   .rstb(rstb),     // 1-bit input: Reset signal for the final port B output register stage. Synchronously
```

```
                    // resets output port doutb to the value specified by parameter READ_RESET_VALUE_B.

   .wea(wea)        // WRITE_DATA_WIDTH_A/BYTE_WRITE_WIDTH_A-bit input: Write enable vector for port A input
                       // data port dina. 1 bit wide when word-wide writes are used. In byte-wide write
                    // configurations, each bit controls the writing one byte of dina to address addra. For
                    // example, to synchronously write only bits [15-8] of dina when WRITE_DATA_WIDTH_A is
                       // 32, wea would be 4'b0010.

);

// End of xpm_memory_dpdistram_inst instantiation
```

XPM_MEMORY_DPROM

Parameterized Macro: Dual Port ROM

MACRO_GROUP: XPM
MACRO_SUBGROUP: XPM_MEMORY



Introduction

This macro is used to instantiate True Dual Port ROM. Read operations from the memory can be performed from Port A and Port B simulataneously.

The following describes the basic read and write port usage of an XPM_MEMORY instance. It does not distinguish between ports A and B.

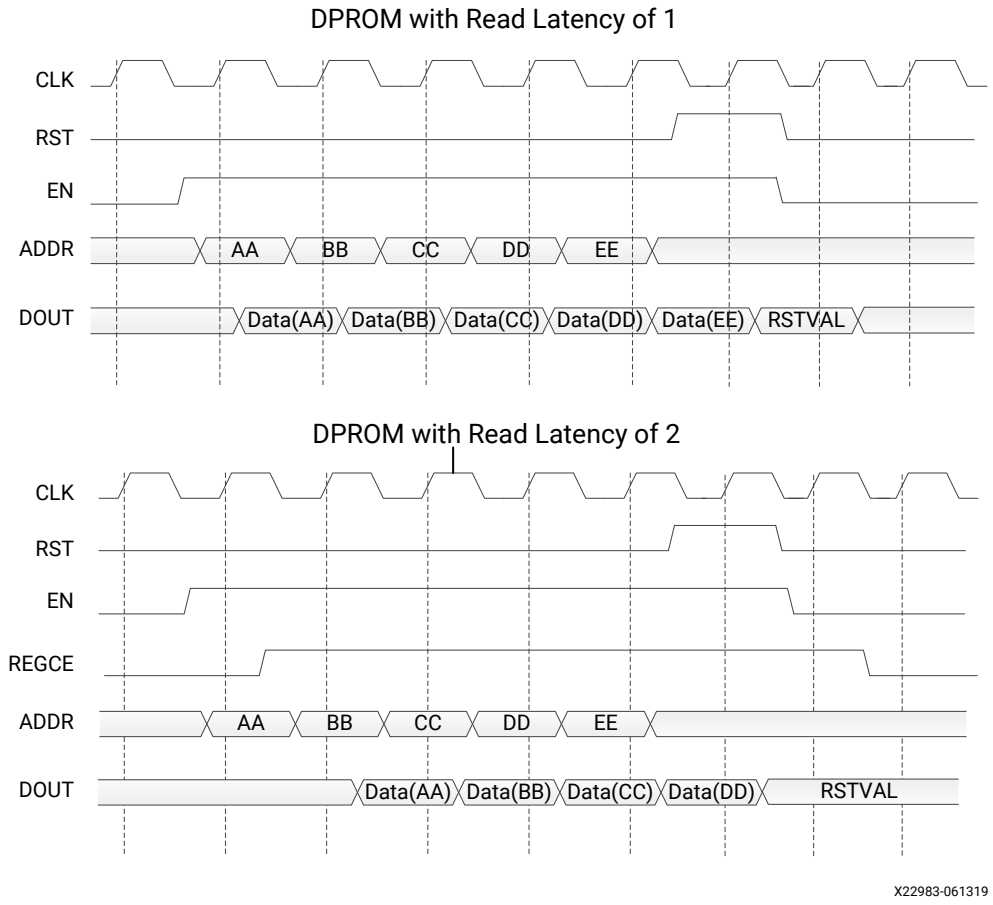
- All synchronous signals are sensitive to the rising edge of clk[a|b], which is assumed to be a buffered and toggling clock signal behaving according to target device and memory primitive requirements.

- A read operation is implicitly performed to address addr[a|b] combinatorially. The data output is registered each clk[a|b] cycle that en[a|b] is asserted.
- Read data appears on the dout[a|b] port READ_LATENCY_[A|B] clk[a|b] cycles after the associated read operation.
- All read operations are gated by the value of en[a|b] on the initiating clk[a|b] cycle, regardless of input or output latencies.
- For each clk[a|b] cycle that rst[a|b] is asserted, the final output register is immediately but synchronously reset to READ_RESET_VALUE_[A|B], irrespective of READ_LATENCY_[A|B].
- For each clk[a|b] cycle that regce[a|b] is asserted and rst[a|b] is de-asserted, the final output register captures and outputs the value from the previous pipeline register.
- Undriven or unknown values provided on module inputs will produce undefined memory array and output port behavior.
- When MEMORY_INIT_PARAM is used, the maximum supported memory size 4K bits.
- WRITE_MODE_A must be set to “read_first” in Dual Port ROM configurations. Violating this will result in a DRC error.

***Note*:**

- When the attribute “CLOCKING_MODE” is set to “common_clock”, all read/write operations to memory through port A and port B are performed on clka. If this attribute is set to “independent_clock”, then read/write operations through port A are performed based on clka, and read/write operations through port B are performed based on `clkb`.
- set_false_path constraint is needed for the independent clock distributed RAM based memory if the design takes care of avoiding address collision (write address != read address at any given point of time).
- For larger memories (≥2 MB), the recommended read latency must be > 8 because the default cascade height used by Vivado synthesis is 8.

Timing Diagrams



***Note*:** The above waveforms do not distinguish between port A and port B. The behavior shown in the above waveforms is true for both port A and port B.

Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
addra	Input	ADDR_WIDTH_A	clka	NA	Active	Address for port A read operations.
addrb	Input	ADDR_WIDTH_B	clkb	NA	Active	Address for port B read operations.
clka	Input	1	NA	EDGE_RISING	Active	Clock signal for port A. Also clocks port B when parameter CLOCKING_MODE is "common_clock".
clkb	Input	1	NA	EDGE_RISING	Active	Clock signal for port B when parameter CLOCKING_MODE is "independent_clock". Unused when parameter CLOCKING_MODE is "common_clock".

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
dbiterra	Output	1	clka	LEVEL_HIGH	DoNotCare	Leave open.
dbiterrb	Output	1	clkb	LEVEL_HIGH	DoNotCare	Leave open.
douta	Output	READ_DATA_WIDTH_A	clka	NA	Active	Data output for port A read operations.
doutb	Output	READ_DATA_WIDTH_B	clkb	NA	Active	Data output for port B read operations.
ena	Input	1	clka	LEVEL_HIGH	Active	Memory enable signal for port A. Must be high on clock cycles when read operations are initiated. Pipelined internally.
enb	Input	1	clkb	LEVEL_HIGH	Active	Memory enable signal for port B. Must be high on clock cycles when read operations are initiated. Pipelined internally.
injectdbiterra	Input	1	clka	LEVEL_HIGH	0	Do not change from the provided value.
injectdbiterrb	Input	1	clkb	LEVEL_HIGH	0	Do not change from the provided value.
injectsbiterra	Input	1	clka	LEVEL_HIGH	0	Do not change from the provided value.
injectsbiterrb	Input	1	clkb	LEVEL_HIGH	0	Do not change from the provided value.
regcea	Input	1	clka	LEVEL_HIGH	1	Do not change from the provided value.
regceb	Input	1	clkb	LEVEL_HIGH	1	Do not change from the provided value.
rsta	Input	1	clka	LEVEL_HIGH	Active	Reset signal for the final port A output register stage. Synchronously resets output port douta to the value specified by parameter READ_RESET_VALUE_A.
rstb	Input	1	clkb	LEVEL_HIGH	Active	Reset signal for the final port B output register stage. Synchronously resets output port doutb to the value specified by parameter READ_RESET_VALUE_B.
sbiterra	Output	1	clka	LEVEL_HIGH	DoNotCare	Leave open.
sbiterrb	Output	1	clkb	LEVEL_HIGH	DoNotCare	Leave open.
sleep	Input	1	NA	LEVEL_HIGH	0	sleep signal to enable the dynamic power saving feature.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ADDR_WIDTH_A	DECIMAL	1 to 20	6	Specify the width of the port A address port addra, in bits. Must be large enough to access the entire memory from port A, i.e. >= $clog2(MEMORY_SIZE/READ_DATA_WIDTH_A).
ADDR_WIDTH_B	DECIMAL	1 to 20	6	Specify the width of the port B address port addrb, in bits. Must be large enough to access the entire memory from port B, i.e. >= $clog2(MEMORY_SIZE/READ_DATA_WIDTH_B).
AUTO_SLEEP_TIME	DECIMAL	0 to 15	0	Must be set to 0 0 - Disable auto-sleep feature
CASCADE_HEIGHT	DECIMAL	0 to 64	0	0- No Cascade Height, Allow Vivado Synthesis to choose. 1 or more - Vivado Synthesis sets the specified value as Cascade Height.
CLOCKING_MODE	STRING	"common _clock", "independent _clock"	"common _clock"	Designate whether port A and port B are clocked with a common clock or with independent clocks- "common_clock"- Common clocking; clock both port A and port B with clka "independent_clock"- Independent clocking; clock port A with clka and port B with clkb
ECC_MODE	STRING	"no_ecc", "both _encode _and _decode", "decode _only", "encode _only"	"no_ecc"	• "no_ecc" - Disables ECC • "encode_only" - Enables ECC Encoder only • "decode_only" - Enables ECC Decoder only • "both_encode_and_decode" - Enables both ECC Encoder and Decoder
MEMORY_INIT_FILE	STRING	String	"none"	Specify "none" (including quotes) for no memory initialization, or specify the name of a memory initialization file- Enter only the name of the file with .mem extension, including quotes but without path (e.g. "my_file.mem"). File format must be ASCII and consist of only hexadecimal values organized into the specified depth by narrowest data width generic value of the memory. See the Memory File (MEM) section for more information on the syntax. Initialization of memory happens through the file name specified only when parameter MEMORY_INIT_PARAM value is equal to "". | When using XPM_MEMORY in a project, add the specified file to the Vivado project as a design source.

Attribute	Type	Allowed Values	Default	Description
MEMORY_INIT _PARAM	STRING	String	"0"	Specify "" or "0" (including quotes) for no memory initialization through parameter, or specify the string containing the hex characters. Enter only hex characters with each location separated by delimiter (,). Parameter format must be ASCII and consist of only hexadecimal values organized into the specified depth by narrowest data width generic value of the memory. For example, if the narrowest data width is 8, and the depth of memory is 8 locations, then the parameter value should be passed as shown below. parameter MEMORY_INIT_PARAM = "AB,CD,EF,1,2,34,56,78" Where "AB" is the 0th location and "78" is the 7th location.
MEMORY _OPTIMIZATION	STRING	"true", "false"	"true"	Specify "true" to enable the optimization of unused memory or bits in the memory structure. Specify "false" to disable the optimization of unused memory or bits in the memory structure
MEMORY _PRIMITIVE	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the memory primitive (resource type) to use- "auto"- Allow Vivado Synthesis to choose "distributed"- Distributed memory "block"- Block memory
MEMORY_SIZE	DECIMAL	2 to 150994944	2048	Specify the total memory array size, in bits. For example, enter 65536 for a 2kx32 ROM.
MESSAGE_CONTROL	DECIMAL	0 to 1	0	Specify 1 to enable the dynamic message reporting such as collision warnings, and 0 to disable the message reporting
READ_DATA_WIDTH_A	DECIMAL	1 to 4608	32	Specify the width of the port A read data output port douta, in bits.
READ_DATA_WIDTH_B	DECIMAL	1 to 4608	32	Specify the width of the port B read data output port doutb, in bits.
READ_LATENCY_A	DECIMAL	0 to 100	2	Specify the number of register stages in the port A read data pipeline. Read data output to port douta takes this number of clka cycles. To target block memory, a value of 1 or larger is required- 1 causes use of memory latch only; 2 causes use of output register. To target distributed memory, a value of 0 or larger is required- 0 indicates combinatorial output. Values larger than 2 synthesize additional flip-flops that are not retimed into memory primitives.
READ_LATENCY_B	DECIMAL	0 to 100	2	Specify the number of register stages in the port B read data pipeline. Read data output to port doutb takes this number of clkb cycles (clka when CLOCKING_MODE is "common_clock"). To target block memory, a value of 1 or larger is required- 1 causes use of memory latch only; 2 causes use of output register. To target distributed memory, a value of 0 or larger is required- 0 indicates combinatorial output. Values larger than 2 synthesize additional flip-flops that are not retimed into memory primitives.

Attribute	Type	Allowed Values	Default	Description
READ_RESET_VALUE _A	STRING	String	"0"	Specify the reset value of the port A final output register stage in response to rsta input port is assertion. For example, to reset the value of port douta to all 0s when READ_DATA_WIDTH_A is 32, specify 32HHHHh0.
READ_RESET_VALUE _B	STRING	String	"0"	Specify the reset value of the port B final output register stage in response to rstb input port is assertion.
RST_MODE_A	STRING	"SYNC", "ASYNC"	"SYNC"	Describes the behaviour of the reset • "SYNC" - when reset is applied, synchronously resets output port douta to the value specified by parameter READ_RESET_VALUE_A • "ASYNC" - when reset is applied, asynchronously resets output port douta to zero
RST_MODE_B	STRING	"SYNC", "ASYNC"	"SYNC"	Describes the behaviour of the reset • "SYNC" - when reset is applied, synchronously resets output port doutb to the value specified by parameter READ_RESET_VALUE_B • "ASYNC" - when reset is applied, asynchronously resets output port doutb to zero
SIM_ASSERT_CHK	DECIMAL	0 to 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.
USE_MEM_INIT	DECIMAL	0 to 1	1	Specify 1 to enable the generation of below message and 0 to disable generation of the following message completely. "INFO - MEMORY_INIT_FILE and MEMORY_INIT_PARAM together specifies no memory initialization. Initial memory contents will be all 0s." NOTE: This message gets generated only when there is no Memory Initialization specified either through file or Parameter.
WAKEUP_TIME	STRING	"disable _sleep", "use _sleep _pin"	"disable _sleep"	Specify "disable_sleep" to disable dynamic power saving option, and specify "use_sleep_pin" to enable the dynamic power saving option

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_memory_dprom: Dual Port ROM
-- Xilinx Parameterized Macro, version 2020.1

xpm_memory_dprom_inst : xpm_memory_dprom
generic map (
   ADDR_WIDTH_A => 6,               -- DECIMAL
   ADDR_WIDTH_B => 6,               -- DECIMAL
   AUTO_SLEEP_TIME => 0,            -- DECIMAL
   CASCADE_HEIGHT => 0,             -- DECIMAL
   CLOCKING_MODE => "common_clock", -- String
   ECC_MODE => "no_ecc",            -- String
   MEMORY_INIT_FILE => "none",      -- String
   MEMORY_INIT_PARAM => "0",        -- String
   MEMORY_OPTIMIZATION => "true",   -- String
   MEMORY_PRIMITIVE => "auto",      -- String
   MEMORY_SIZE => 2048,             -- DECIMAL
   MESSAGE_CONTROL => 0,            -- DECIMAL
   READ_DATA_WIDTH_A => 32,         -- DECIMAL
   READ_DATA_WIDTH_B => 32,         -- DECIMAL
   READ_LATENCY_A => 2,             -- DECIMAL
   READ_LATENCY_B => 2,             -- DECIMAL
   READ_RESET_VALUE_A => "0",       -- String
   READ_RESET_VALUE_B => "0",       -- String
   RST_MODE_A => "SYNC",            -- String
   RST_MODE_B => "SYNC",            -- String
   SIM_ASSERT_CHK => 0,          -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   USE_MEM_INIT => 1,               -- DECIMAL
   WAKEUP_TIME => "disable_sleep"   -- String
)
port map (
   dbiterra => dbiterra,             -- 1-bit output: Leave open.
   dbiterrb => dbiterrb,             -- 1-bit output: Leave open.
   douta => douta,                -- READ_DATA_WIDTH_A-bit output: Data output for port A read operations.
   doutb => doutb,                -- READ_DATA_WIDTH_B-bit output: Data output for port B read operations.
   sbiterra => sbiterra,             -- 1-bit output: Leave open.
   sbiterrb => sbiterrb,             -- 1-bit output: Leave open.
   addra => addra,                   -- ADDR_WIDTH_A-bit input: Address for port A read operations.
   addrb => addrb,                   -- ADDR_WIDTH_B-bit input: Address for port B read operations.
   clka => clka,                  -- 1-bit input: Clock signal for port A. Also clocks port B when
                                     -- parameter CLOCKING_MODE is "common_clock".

   clkb => clkb,                  -- 1-bit input: Clock signal for port B when parameter CLOCKING_MODE is
                                     -- "independent_clock". Unused when parameter CLOCKING_MODE is
                                     -- "common_clock".

   ena => ena,                    -- 1-bit input: Memory enable signal for port A. Must be high on clock
                                  -- cycles when read operations are initiated. Pipelined internally.

   enb => enb,                    -- 1-bit input: Memory enable signal for port B. Must be high on clock
                                  -- cycles when read operations are initiated. Pipelined internally.

   injectdbiterra => injectdbiterra, -- 1-bit input: Do not change from the provided value.
   injectdbiterrb => injectdbiterrb, -- 1-bit input: Do not change from the provided value.
   injectsbiterra => injectsbiterra, -- 1-bit input: Do not change from the provided value.
   injectsbiterrb => injectsbiterrb, -- 1-bit input: Do not change from the provided value.
   regcea => regcea,                 -- 1-bit input: Do not change from the provided value.
   regceb => regceb,                 -- 1-bit input: Do not change from the provided value.
   rsta => rsta,                  -- 1-bit input: Reset signal for the final port A output register
                                  -- stage. Synchronously resets output port douta to the value specified
                                     -- by parameter READ_RESET_VALUE_A.

   rstb => rstb,                  -- 1-bit input: Reset signal for the final port B output register
                                  -- stage. Synchronously resets output port doutb to the value specified
                                     -- by parameter READ_RESET_VALUE_B.
```

```
   sleep => sleep                   -- 1-bit input: sleep signal to enable the dynamic power saving feature.
);

-- End of xpm_memory_dprom_inst instantiation
```

Verilog Instantiation Template

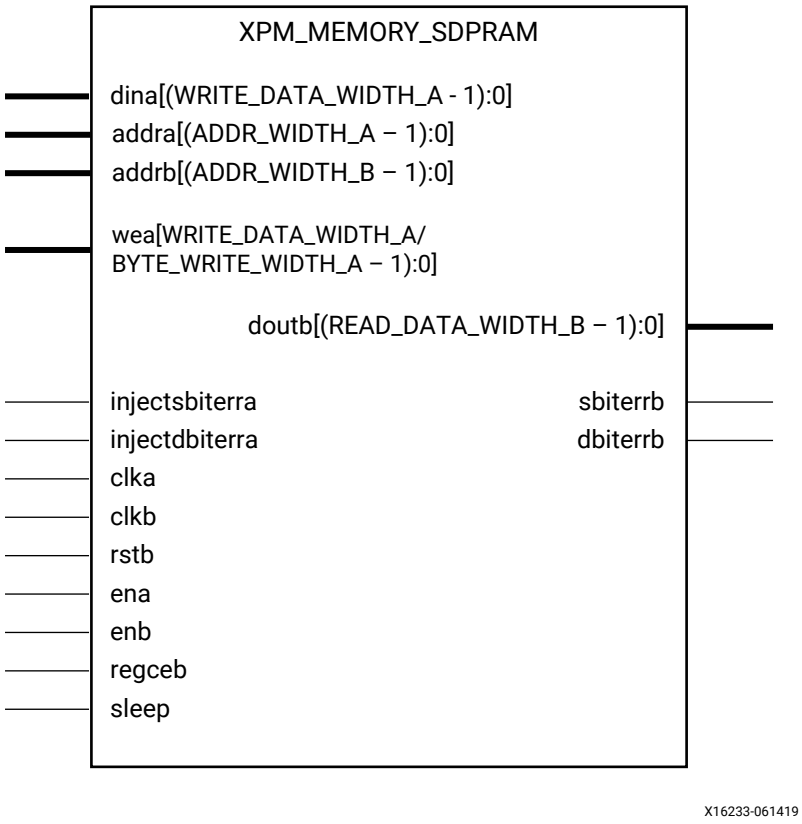
```
// xpm_memory_dprom: Dual Port ROM
// Xilinx Parameterized Macro, version 2020.1

xpm_memory_dprom #(
   .ADDR_WIDTH_A(6),               // DECIMAL
   .ADDR_WIDTH_B(6),               // DECIMAL
   .AUTO_SLEEP_TIME(0),            // DECIMAL
   .CASCADE_HEIGHT(0),             // DECIMAL
   .CLOCKING_MODE("common_clock"), // String
   .ECC_MODE("no_ecc"),            // String
   .MEMORY_INIT_FILE("none"),      // String
   .MEMORY_INIT_PARAM("0"),        // String
   .MEMORY_OPTIMIZATION("true"),   // String
   .MEMORY_PRIMITIVE("auto"),      // String
   .MEMORY_SIZE(2048),             // DECIMAL
   .MESSAGE_CONTROL(0),            // DECIMAL
   .READ_DATA_WIDTH_A(32),         // DECIMAL
   .READ_DATA_WIDTH_B(32),         // DECIMAL
   .READ_LATENCY_A(2),             // DECIMAL
   .READ_LATENCY_B(2),             // DECIMAL
   .READ_RESET_VALUE_A("0"),       // String
   .READ_RESET_VALUE_B("0"),       // String
   .RST_MODE_A("SYNC"),            // String
   .RST_MODE_B("SYNC"),            // String
   .SIM_ASSERT_CHK(0),           // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .USE_MEM_INIT(1),               // DECIMAL
   .WAKEUP_TIME("disable_sleep")   // String
)
xpm_memory_dprom_inst (
   .dbiterra(dbiterra),             // 1-bit output: Leave open.
   .dbiterrb(dbiterrb),             // 1-bit output: Leave open.
   .douta(douta),                 // READ_DATA_WIDTH_A-bit output: Data output for port A read operations.
   .doutb(doutb),                 // READ_DATA_WIDTH_B-bit output: Data output for port B read operations.
   .sbiterra(sbiterra),             // 1-bit output: Leave open.
   .sbiterrb(sbiterrb),             // 1-bit output: Leave open.
   .addra(addra),                   // ADDR_WIDTH_A-bit input: Address for port A read operations.
   .addrb(addrb),                   // ADDR_WIDTH_B-bit input: Address for port B read operations.
   .clka(clka),                     // 1-bit input: Clock signal for port A. Also clocks port B when
                                    // parameter CLOCKING_MODE is "common_clock".

   .clkb(clkb),                   // 1-bit input: Clock signal for port B when parameter CLOCKING_MODE is
                                    // "independent_clock". Unused when parameter CLOCKING_MODE is
                                    // "common_clock".

   .ena(ena),                     // 1-bit input: Memory enable signal for port A. Must be high on clock
                                  // cycles when read operations are initiated. Pipelined internally.

   .enb(enb),                     // 1-bit input: Memory enable signal for port B. Must be high on clock
                                  // cycles when read operations are initiated. Pipelined internally.

   .injectdbiterra(injectdbiterra), // 1-bit input: Do not change from the provided value.
   .injectdbiterrb(injectdbiterrb), // 1-bit input: Do not change from the provided value.
   .injectsbiterra(injectsbiterra), // 1-bit input: Do not change from the provided value.
   .injectsbiterrb(injectsbiterrb), // 1-bit input: Do not change from the provided value.
   .regcea(regcea),                 // 1-bit input: Do not change from the provided value.
   .regceb(regceb),                 // 1-bit input: Do not change from the provided value.
   .rsta(rsta),                   // 1-bit input: Reset signal for the final port A output register stage.
                                  // Synchronously resets output port douta to the value specified by
                                    // parameter READ_RESET_VALUE_A.

   .rstb(rstb),                   // 1-bit input: Reset signal for the final port B output register stage.
                                  // Synchronously resets output port doutb to the value specified by
                                    // parameter READ_RESET_VALUE_B.

   .sleep(sleep)                  // 1-bit input: sleep signal to enable the dynamic power saving feature.
);

// End of xpm_memory_dprom_inst instantiation
```

XPM_MEMORY_SDPRAM

Parameterized Macro: Simple Dual Port RAM

MACRO_GROUP: XPM

MACRO_SUBGROUP: XPM_MEMORY



Introduction

This macro is used to instantiate Simple Dual Port RAM. Port A is used to perform write operations from the memory and port B can be used to read from the memory.

The following describes the basic read and write port usage of an XPM_MEMORY instance. It does not distinguish between port A and port B.

- All synchronous signals are sensitive to the rising edge of clk[a|b], which is assumed to be a buffered and toggling clock signal behaving according to target device and memory primitive requirements.
- A read operation is implicitly performed to address addrb combinatorially. The data output is registered each clkb cycle that enb is asserted.

- Read data appears on the doutb port READ_LATENCY_B clkb cycles after the associated read operation.
- A write operation is explicitly performed, writing dina to address addra, when both ena and wea are asserted on each clka cycle.
- All read and write operations are gated by the value of en[a|b] on the initiating clk[a|b] cycle, regardless of input or output latencies. The addra and wea inputs have no effect when ena is de-asserted on the coincident clk[a|b] cycle.
- For each clkb cycle that rstb is asserted, the final output register is immediately but synchronously reset to READ_RESET_VALUE_B, irrespective of READ_LATENCY_B.
- For each clkb cycle that regceb is asserted and rstb is de-asserted, the final output register captures and outputs the value from the previous pipeline register.
- Undriven or unknown values provided on module inputs will produce undefined memory array and output port behavior.
- When MEMORY_INIT_PARAM is used, the maximum supported memory size 4K bits.
- In Simple Dual Port RAM configuration, only WRITE_MODE_B is considered (though port A has the write permissions, WRITE_MODE_B is used because the output data will be connected to port B, and the same mode value is applied to WRITE_MODE_A internally when passing to the primitive). Choosing the Invalid Configuration will result in a DRC.

***Note*:**

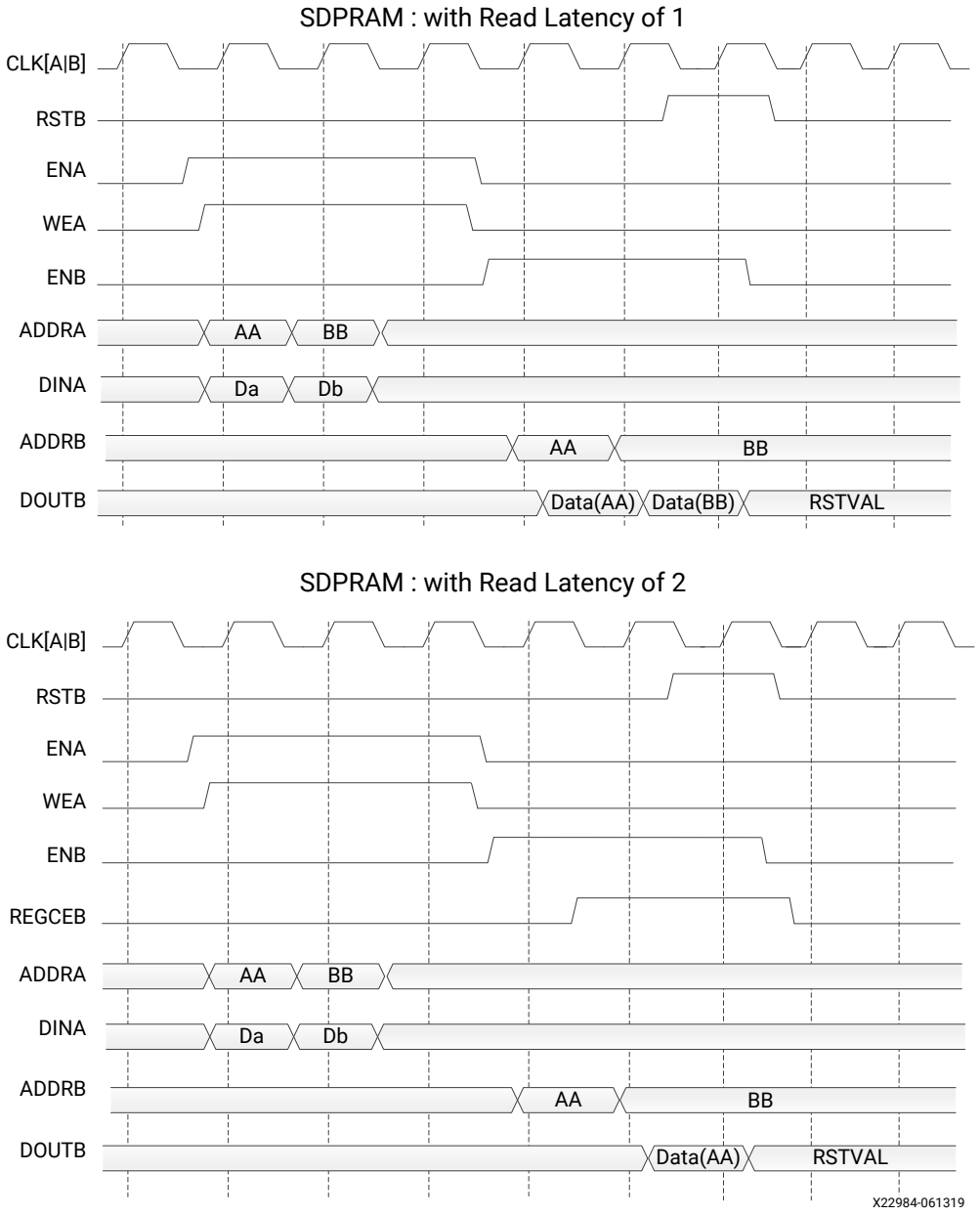
- When the attribute “CLOCKING_MODE” is set to “common_clock”, all read/write operations to memory through port A and port B are performed on clka. If this attribute is set to “independent_clock”, then read/write operations through port A are performed based on clka, and read/write operations through port B are performed based on `clkb`.
- Writing to an out-of-range address location may overwrite a valid address location when effective address bits match to a physical memory address location.
- set_false_path constraint is needed for the independent clock distributed RAM based memory if the design takes care of avoiding address collision (write address != read address at any given point of time). Set USE_EMBEDDED_CONSTRAINT = 1 if XPM_MEMORY needs to take care of necessary constraints. If USE_EMBEDDED_CONSTRAINT = 0, Vivado may trigger Timing-6 or Timing-7 or both. Alternatively, you can also add the constraint when USE_EMBEDDED_CONSTRAINT = 0. An example of adding this constraint is provided below. If Port-B also has write permissions for an Independent clock configuration, then a similar constraint needs to be added for `clkb` as well.

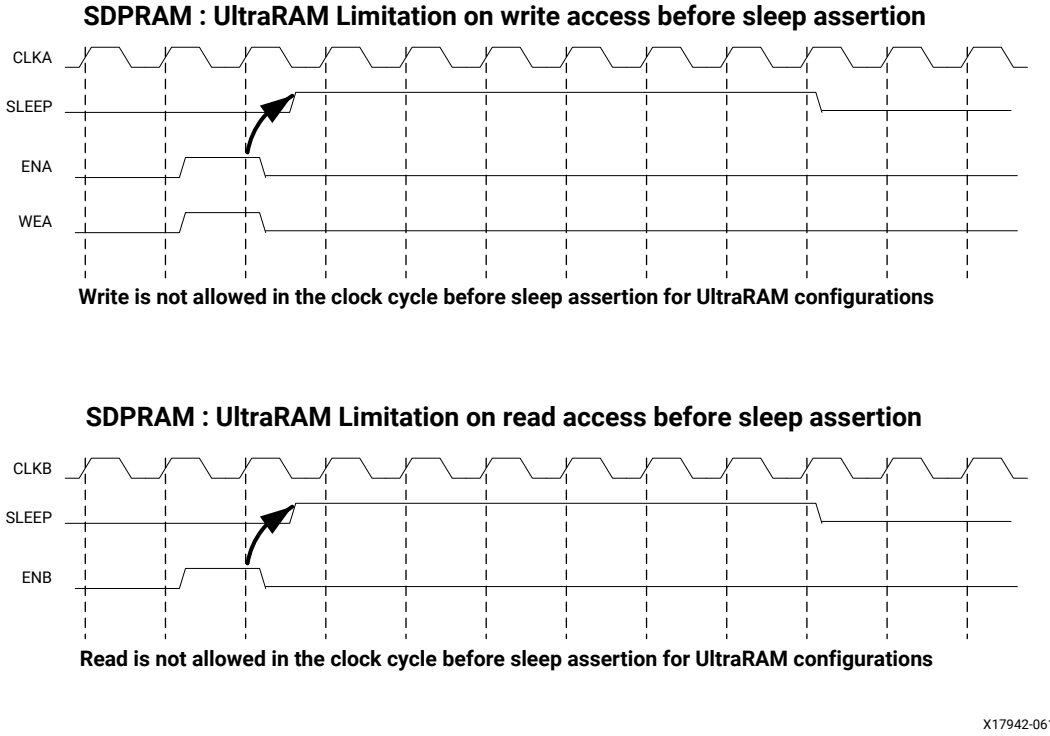
```
set_false_path -from [filter [all_fanout -from [get_ports clka]
-flat -endpoints_only] {IS_LEAF}] -through [get_pins -of_objects
[get_cells -hier * -filter {PRIMITIVE_SUBGROUP==LUTRAM ||
PRIMITIVE_SUBGROUP==dram || PRIMITIVE_SUBGROUP==drom}]
-filter {DIRECTION==OUT}]
```

- If"CLOCKING_MODE" is set to "independent_clock", Vivado may trigger a false positive CDC-1 warning and can be ignored.
- The use of UltraRAM’s dedicated input and output registers are controlled by synthesis based on the READ_LATENCY_B value. For example, if 4 UltraRAMs are in cascade and the READ_LATENCY_B is ≥ 4, then synthesis will absorb as much registers inside UltraRAM primitive as possible.

- For UltraRAM's, the enablement of OREG depends on the READ_LATENCY_B and WRITE_MODE_B. OREG enabled when READ_LATENCY_B ≥ 3 in READ_FIRST mode and READ_LATENCY_B ≥4 in WRITE_FIRST mode.
- For larger memories (≥2 MB), the recommended read latency must be > 8 because the default cascade height used by Vivado synthesis is 8.

Timing Diagrams





***Note*:** The UltraRAM primitive does not support Write/Read access in the clock cycle just before assertion of sleep gets recognized on the positive edge of the clock when its OREG attribute is set to TRUE. For UltraRAM configurations, Write/Read access to the memory is not allowed in the clock cycle just before the assertion of sleep.

ECC Modes

Both Block RAM and UltraRAM primitives support ECC when the memory type is set to Simple Dual Port RAM. The three ECC modes supported are:

- Both encode and decode
- Encode only
- Decode only

The read and write usage of the three ECC Modes are the same as described in the Introduction section above. See the "Built-in Error Correction" section of the *7 Series FPGAs Memory Resources User Guide* (UG473) for more details on this feature like Error Injection and syndrome bits calculations.

There are restrictions on the attributes WRITE_DATA_WIDTH_A, READ_DATA_WIDTH_B, and MEMORY_SIZE in each of the above ECC modes.

- **Both encode and decode** WRITE_DATA_WIDTH_A and READ_DATA_WIDTH_B must be multiples of 64-bits. Violating this rule will results in a DRC in XPM_Memory.

- **Encode only** WRITE_DATA_WIDTH_A must be a multiple of 64 bits and READ_DATA_WIDTH_B must be a multiple of 72-bits. MEMORY_SIZE must be a multiple of READ_DATA_WIDTH_B. Violating these rules will result in a DRC.
- **Decode only** WRITE_DATA_WIDTH_A must be a multiple of 72 bits and READ_DATA_WIDTH_B must be a multiple of 64-bits. MEMORY_SIZE must be a multiple of WRITE_DATA_WIDTH_A. Violating these rules will result in a DRC.

When ECC is enabled the following are not supported:

- Asymmetry
- Initialization
- Reset (neither non-zero reset value nor reset assertion)

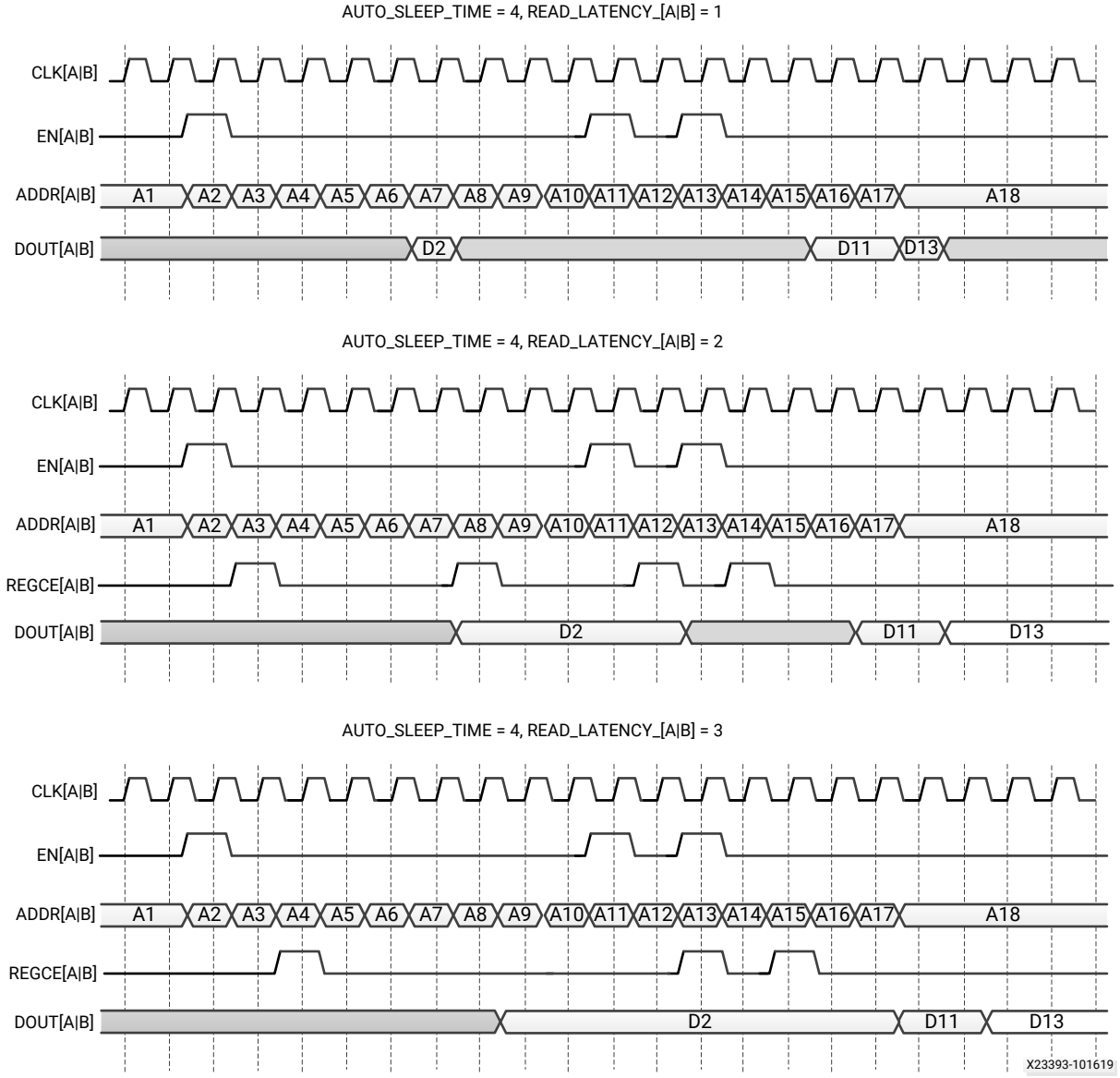
Note: ECC uses a hard-ECC block available in the BRAM/URAM macro and the data width should be multiples of 64/72. Use ECC IP for other data width combinations.

Auto Sleep Mode

- This feature is applicable only when MEMORY_PRIMITIVE is URAM and is controlled internally in the UltraRAM to check if it can be put in sleep mode and when it needs to wake up. Thus power savings are obtained automatically without having to explicitly control the SLEEP Pin.
- When AUTO_SLEEP_TIME is 0, the feature is disabled. When AUTO_SLEEP_TIME is non-zero, XPM_MEMORY constructs the pipeline registers equal to AUTO_SLEEP_TIME value on all input signals except `rst[a|b]`.
- If AUTO_SLEEP_TIME is too low, then UltraRAM goes into sleep and wakeup too often, which can cause more power to be consumed.
- The number of sleep cycles achieved is calculated by following formula:
  - If number of consecutive inactive cycles is < AUTO_SLEEP_TIME, then number of sleep cycles = 0.
  - If number of consecutive inactive cycles is ≥AUTO_SLEEP_TIME, Then number of consecutive sleep cycles = Number of consecutive inactive cycles – 3.
  - Inactive cycle is defined as a cycle where there is no Read/Write operation from either port.
- The latency between the read operation and the data arrival at dout[a|b] is AUTO_SLEEP_TIME + READ_LATENCY_[A|B] clock cycles (Assuming that REGCE is high when the output data pipe line exists).
- When the READ_LATENCY_[A|B] is set to 1 or 2, XPM_Memory behaviorally models the AUTO SLEEP feature and forces ‘x’ on DOUT[A|B] when the RAM is in Auto Sleep Mode. For READ_LATENCY_[A|B] greater than 2, the propagation of ‘x’ cannot happen to the DOUT[A|B] as the output registers gets the clock enable (delayed read enable) after UltraRAM comes out of sleep mode.

- The Auto Sleep mode is most effective for larger memory sizes or any memory with very little activity.

Timing diagrams for Auto Sleep Mode at various read latencies are shown below.



Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
addra	Input	ADDR_WIDTH_A	clka	NA	Active	Address for port A write operations.
addrb	Input	ADDR_WIDTH_B	clkb	NA	Active	Address for port B read operations.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
clka	Input	1	NA	EDGE _RISING	Active	Clock signal for port A. Also clocks port B when parameter CLOCKING_MODE is "common_clock".
clkb	Input	1	NA	EDGE _RISING	Active	Clock signal for port B when parameter CLOCKING_MODE is "independent_clock". Unused when parameter CLOCKING_MODE is "common_clock".
dbiterrb	Output	1	clkb	LEVEL _HIGH	DoNotCare	Status signal to indicate double bit error occurrence on the data output of port B.
dina	Input	WRITE _DATA _WIDTH _A	clka	NA	Active	Data input for port A write operations.
doutb	Output	READ _DATA _WIDTH _B	clkb	NA	Active	Data output for port B read operations.
ena	Input	1	clka	LEVEL _HIGH	Active	Memory enable signal for port A. Must be high on clock cycles when write operations are initiated. Pipelined internally.
enb	Input	1	clkb	LEVEL _HIGH	Active	Memory enable signal for port B. Must be high on clock cycles when read operations are initiated. Pipelined internally.
injectdbiterra	Input	1	clka	LEVEL _HIGH	0	Controls double bit error injection on input data when ECC enabled (Error injection capability is not available in "decode_only" mode).
injectsbiterra	Input	1	clka	LEVEL _HIGH	0	Controls single bit error injection on input data when ECC enabled (Error injection capability is not available in "decode_only" mode).
regceb	Input	1	clkb	LEVEL _HIGH	1	Clock Enable for the last register stage on the output data path.
rstb	Input	1	clkb	LEVEL _HIGH	Active	Reset signal for the final port B output register stage. Synchronously resets output port doutb to the value specified by parameter READ_RESET_VALUE_B.
sbiterrb	Output	1	clkb	LEVEL _HIGH	DoNotCare	Status signal to indicate single bit error occurrence on the data output of port B.
sleep	Input	1	NA	LEVEL _HIGH	0	sleep signal to enable the dynamic power saving feature.
wea	Input	WRITE _DATA _WIDTH _A / BYTE _WRITE _WIDTH _A	clka	LEVEL _HIGH	Active	Write enable vector for port A input data port dina. 1 bit wide when word-wide writes are used. In byte-wide write configurations, each bit controls the writing one byte of dina to address addra. For example, to synchronously write only bits [15-8] of dina when WRITE_DATA_WIDTH_A is 32, wea would be 4'b0010.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ADDR_WIDTH_A	DECIMAL	1 to 20	6	Specify the width of the port A address port addra, in bits. Must be large enough to access the entire memory from port A, i.e. >= $clog2(MEMORY_SIZE/WRITE_DATA_WIDTH_A).
ADDR_WIDTH_B	DECIMAL	1 to 20	6	Specify the width of the port B address port addrb, in bits. Must be large enough to access the entire memory from port B, i.e. >= $clog2(MEMORY_SIZE/READ_DATA_WIDTH_B).
AUTO_SLEEP_TIME	DECIMAL	0 to 15	0	Number of clk[a|b] cycles to auto-sleep, if feature is available in architecture. • 0 - Disable auto-sleep feature • 3-15 - Number of auto-sleep latency cycles Do not change from the value provided in the template instantiation.
BYTE_WRITE_WIDTH _A	DECIMAL	1 to 4608	32	To enable byte-wide writes on port A, specify the byte width, in bits. • 8- 8-bit byte-wide writes, legal when WRITE_DATA_WIDTH_A is an integer multiple of 8 • 9- 9-bit byte-wide writes, legal when WRITE_DATA_WIDTH_A is an integer multiple of 9 Or to enable word-wide writes on port A, specify the same value as for WRITE_DATA_WIDTH_A.
CASCADE_HEIGHT	DECIMAL	0 to 64	0	0- No Cascade Height, Allow Vivado Synthesis to choose. 1 or more - Vivado Synthesis sets the specified value as Cascade Height.
CLOCKING_MODE	STRING	"common_clock", "independent_clock"	"common_clock"	Designate whether port A and port B are clocked with a common clock or with independent clocks. • "common_clock"- Common clocking; clock both port A and port B with clka • "independent_clock"- Independent clocking; clock port A with clka and port B with clkb

Attribute	Type	Allowed Values	Default	Description
ECC_MODE	STRING	"no_ecc", "both _encode _and _decode", "decode _only", "encode _only"	"no_ecc"	• "no_ecc" - Disables ECC • "encode_only" - Enables ECC Encoder only • "decode_only" - Enables ECC Decoder only • "both_encode_and_decode" - Enables both ECC Encoder and Decoder
MEMORY_INIT_FILE	STRING	String	"none"	Specify "none" (including quotes) for no memory initialization, or specify the name of a memory initialization file. Enter only the name of the file with .mem extension, including quotes but without path (e.g. "my_file.mem"). File format must be ASCII and consist of only hexadecimal values organized into the specified depth by narrowest data width generic value of the memory. See the Memory File (MEM) section for more information on the syntax. Initialization of memory happens through the file name specified only when parameter MEMORY_INIT_PARAM value is equal to "". When using XPM_MEMORY in a project, add the specified file to the Vivado project as a design source.
MEMORY_INIT _PARAM	STRING	String	"0"	Specify "" or "0" (including quotes) for no memory initialization through parameter, or specify the string containing the hex characters. Enter only hex characters with each location separated by delimiter (,). Parameter format must be ASCII and consist of only hexadecimal values organized into the specified depth by narrowest data width generic value of the memory. For example, if the narrowest data width is 8, and the depth of memory is 8 locations, then the parameter value should be passed as shown below. parameter MEMORY_INIT_PARAM = "AB,CD,EF,1,2,34,56,78" Where "AB" is the 0th location and "78" is the 7th location.
MEMORY _OPTIMIZATION	STRING	"true", "false"	"true"	Specify "true" to enable the optimization of unused memory or bits in the memory structure. Specify "false" to disable the optimization of unused memory or bits in the memory structure

Attribute	Type	Allowed Values	Default	Description
MEMORY _PRIMITIVE	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the memory primitive (resource type) to use. • "auto"- Allow Vivado Synthesis to choose • "distributed"- Distributed memory • "block"- Block memory • "ultra"- Ultra RAM memory NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with MEMORY_PRIMITIVE set to "auto".
MEMORY_SIZE	DECIMAL	2 to 150994944	2048	Specify the total memory array size, in bits. For example, enter 65536 for a 2kx32 RAM. • When ECC is enabled and set to "encode_only", then the memory size has to be multiples of READ_DATA_WIDTH_B • When ECC is enabled and set to "decode_only", then the memory size has to be multiples of WRITE_DATA_WIDTH_A
MESSAGE_CONTROL	DECIMAL	0 to 1	0	Specify 1 to enable the dynamic message reporting such as collision warnings, and 0 to disable the message reporting
READ_DATA_WIDTH_B	DECIMAL	1 to 4608	32	Specify the width of the port B read data output port doutb, in bits. • When ECC is enabled and set to "encode_only", then READ_DATA_WIDTH_B has to be multiples of 72-bits • When ECC is enabled and set to "decode_only" or "both_encode_and_decode", then READ_DATA_WIDTH_B has to be multiples of 64-bits
READ_LATENCY_B	DECIMAL	0 to 100	2	Specify the number of register stages in the port B read data pipeline. Read data output to port doutb takes this number of clkb cycles (clka when CLOCKING_MODE is "common_clock"). To target block memory, a value of 1 or larger is required- 1 causes use of memory latch only; 2 causes use of output register. To target distributed memory, a value of 0 or larger is required- 0 indicates combinatorial output. Values larger than 2 synthesize additional flip-flops that are not retimed into memory primitives.
READ_RESET_VALUE _B	STRING	String	"0"	Specify the reset value of the port B final output register stage in response to rstb input port is assertion. As this parameter is a string, please specify the hex values inside double quotes. As an example, If the read data width is 8, then specify READ_RESET_VALUE_B = "EA"; When ECC is enabled, reset value is not supported.

Attribute	Type	Allowed Values	Default	Description
RST_MODE_A	STRING	"SYNC", "ASYNC"	"SYNC"	Describes the behaviour of the reset • "SYNC" - when reset is applied, synchronously resets output port douta to the value specified by parameter READ_RESET_VALUE_A • "ASYNC" - when reset is applied, asynchronously resets output port douta to zero
RST_MODE_B	STRING	"SYNC", "ASYNC"	"SYNC"	Describes the behaviour of the reset • "SYNC" - when reset is applied, synchronously resets output port doutb to the value specified by parameter READ_RESET_VALUE_B • "ASYNC" - when reset is applied, asynchronously resets output port doutb to zero
SIM_ASSERT_CHK	DECIMAL	0 to 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.
USE_EMBEDDED _CONSTRAINT	DECIMAL	0 to 1	0	Specify 1 to enable the set_false_path constraint addition between clka of Distributed RAM and doutb_reg on clkb
USE_MEM_INIT	DECIMAL	0 to 1	1	Specify 1 to enable the generation of below message and 0 to disable generation of the following message completely. "INFO - MEMORY_INIT_FILE and MEMORY_INIT_PARAM together specifies no memory initialization. Initial memory contents will be all 0s." NOTE: This message gets generated only when there is no Memory Initialization specified either through file or Parameter.
WAKEUP_TIME	STRING	"disable _sleep", "use _sleep _pin"	"disable _sleep"	Specify "disable_sleep" to disable dynamic power saving option, and specify "use_sleep_pin" to enable the dynamic power saving option
WRITE_DATA_WIDTH_A	DECIMAL	1 to 4608	32	multiples of 64-bits When ECC is enabled and set to "decode_only", then WRITE_DATA_WIDTH_A has to be multiples of 72-bits
WRITE_MODE_B	STRING	"no _change", "read _first", "write _first"	"no _change"	Write mode behavior for port B output data port, doutb.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_memory_sdpram: Simple Dual Port RAM
-- Xilinx Parameterized Macro, version 2020.1

xpm_memory_sdpram_inst : xpm_memory_sdpram
generic map (
   ADDR_WIDTH_A => 6,               -- DECIMAL
   ADDR_WIDTH_B => 6,               -- DECIMAL
   AUTO_SLEEP_TIME => 0,            -- DECIMAL
   BYTE_WRITE_WIDTH_A => 32,        -- DECIMAL
   CASCADE_HEIGHT => 0,             -- DECIMAL
   CLOCKING_MODE => "common_clock", -- String
   ECC_MODE => "no_ecc",            -- String
   MEMORY_INIT_FILE => "none",      -- String
   MEMORY_INIT_PARAM => "0",        -- String
   MEMORY_OPTIMIZATION => "true",   -- String
   MEMORY_PRIMITIVE => "auto",      -- String
   MEMORY_SIZE => 2048,             -- DECIMAL
   MESSAGE_CONTROL => 0,            -- DECIMAL
   READ_DATA_WIDTH_B => 32,         -- DECIMAL
   READ_LATENCY_B => 2,             -- DECIMAL
   READ_RESET_VALUE_B => "0",       -- String
   RST_MODE_A => "SYNC",            -- String
   RST_MODE_B => "SYNC",            -- String
   SIM_ASSERT_CHK => 0,             -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   USE_EMBEDDED_CONSTRAINT => 0,    -- DECIMAL
   USE_MEM_INIT => 1,               -- DECIMAL
   WAKEUP_TIME => "disable_sleep",  -- String
   WRITE_DATA_WIDTH_A => 32,        -- DECIMAL
   WRITE_MODE_B => "no_change"      -- String
)
port map (
   dbiterrb => dbiterrb,            -- 1-bit output: Status signal to indicate double bit error occurrence
                                    -- on the data output of port B.

   doutb => doutb,                  -- READ_DATA_WIDTH_B-bit output: Data output for port B read operations.
   sbiterrb => sbiterrb,            -- 1-bit output: Status signal to indicate single bit error occurrence
                                    -- on the data output of port B.

   addra => addra,                  -- ADDR_WIDTH_A-bit input: Address for port A write operations.
   addrb => addrb,                  -- ADDR_WIDTH_B-bit input: Address for port B read operations.
   clka => clka,                    -- 1-bit input: Clock signal for port A. Also clocks port B when
                                    -- parameter CLOCKING_MODE is "common_clock".

   clkb => clkb,                    -- 1-bit input: Clock signal for port B when parameter CLOCKING_MODE is
                                    -- "independent_clock". Unused when parameter CLOCKING_MODE is
                                    -- "common_clock".

   dina => dina,                    -- WRITE_DATA_WIDTH_A-bit input: Data input for port A write operations.
   ena => ena,                      -- 1-bit input: Memory enable signal for port A. Must be high on clock
                                    -- cycles when write operations are initiated. Pipelined internally.

   enb => enb,                      -- 1-bit input: Memory enable signal for port B. Must be high on clock
                                    -- cycles when read operations are initiated. Pipelined internally.

   injectdbiterra => injectdbiterra, -- 1-bit input: Controls double bit error injection on input data when
                                    -- ECC enabled (Error injection capability is not available in
                                    -- "decode_only" mode).

   injectsbiterra => injectsbiterra, -- 1-bit input: Controls single bit error injection on input data when
                                    -- ECC enabled (Error injection capability is not available in
                                    -- "decode_only" mode).

   regceb => regceb,                -- 1-bit input: Clock Enable for the last register stage on the output
                                    -- data path.

   rstb => rstb,                    -- 1-bit input: Reset signal for the final port B output register
```

```
                                   -- stage. Synchronously resets output port doutb to the value specified
                                        -- by parameter READ_RESET_VALUE_B.

   sleep => sleep,                 -- 1-bit input: sleep signal to enable the dynamic power saving feature.
   wea => wea                      -- WRITE_DATA_WIDTH_A/BYTE_WRITE_WIDTH_A-bit input: Write enable vector
                                   -- for port A input data port dina. 1 bit wide when word-wide writes
                                   -- are used. In byte-wide write configurations, each bit controls the
                                        -- writing one byte of dina to address addra. For example, to
                                   -- synchronously write only bits [15-8] of dina when WRITE_DATA_WIDTH_A
                                        -- is 32, wea would be 4'b0010.

);

-- End of xpm_memory_sdpram_inst instantiation
```

Verilog Instantiation Template

```
// xpm_memory_sdpram: Simple Dual Port RAM
// Xilinx Parameterized Macro, version 2020.1

xpm_memory_sdpram #(
   .ADDR_WIDTH_A(6),               // DECIMAL
   .ADDR_WIDTH_B(6),               // DECIMAL
   .AUTO_SLEEP_TIME(0),            // DECIMAL
   .BYTE_WRITE_WIDTH_A(32),        // DECIMAL
   .CASCADE_HEIGHT(0),             // DECIMAL
   .CLOCKING_MODE("common_clock"), // String
   .ECC_MODE("no_ecc"),            // String
   .MEMORY_INIT_FILE("none"),      // String
   .MEMORY_INIT_PARAM("0"),        // String
   .MEMORY_OPTIMIZATION("true"),   // String
   .MEMORY_PRIMITIVE("auto"),      // String
   .MEMORY_SIZE(2048),             // DECIMAL
   .MESSAGE_CONTROL(0),            // DECIMAL
   .READ_DATA_WIDTH_B(32),         // DECIMAL
   .READ_LATENCY_B(2),             // DECIMAL
   .READ_RESET_VALUE_B("0"),       // String
   .RST_MODE_A("SYNC"),            // String
   .RST_MODE_B("SYNC"),            // String
   .SIM_ASSERT_CHK(0),           // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .USE_EMBEDDED_CONSTRAINT(0),    // DECIMAL
   .USE_MEM_INIT(1),               // DECIMAL
   .WAKEUP_TIME("disable_sleep"),  // String
   .WRITE_DATA_WIDTH_A(32),        // DECIMAL
   .WRITE_MODE_B("no_change")      // String
)
xpm_memory_sdpram_inst (
   .dbiterrb(dbiterrb),            // 1-bit output: Status signal to indicate double bit error occurrence
                                        // on the data output of port B.

   .doutb(doutb),                  // READ_DATA_WIDTH_B-bit output: Data output for port B read operations.
   .sbiterrb(sbiterrb),            // 1-bit output: Status signal to indicate single bit error occurrence
                                        // on the data output of port B.

   .addra(addra),                       // ADDR_WIDTH_A-bit input: Address for port A write operations.
   .addrb(addrb),                       // ADDR_WIDTH_B-bit input: Address for port B read operations.
   .clka(clka),                         // 1-bit input: Clock signal for port A. Also clocks port B when
                                        // parameter CLOCKING_MODE is "common_clock".

   .clkb(clkb),                    // 1-bit input: Clock signal for port B when parameter CLOCKING_MODE is
                                        // "independent_clock". Unused when parameter CLOCKING_MODE is
                                        // "common_clock".

   .dina(dina),                    // WRITE_DATA_WIDTH_A-bit input: Data input for port A write operations.
   .ena(ena),                      // 1-bit input: Memory enable signal for port A. Must be high on clock
                                   // cycles when write operations are initiated. Pipelined internally.

   .enb(enb),                      // 1-bit input: Memory enable signal for port B. Must be high on clock
                                   // cycles when read operations are initiated. Pipelined internally.

   .injectdbiterra(injectdbiterra), // 1-bit input: Controls double bit error injection on input data when
                                        // ECC enabled (Error injection capability is not available in
                                        // "decode_only" mode).

   .injectsbiterra(injectsbiterra), // 1-bit input: Controls single bit error injection on input data when
                                        // ECC enabled (Error injection capability is not available in
```

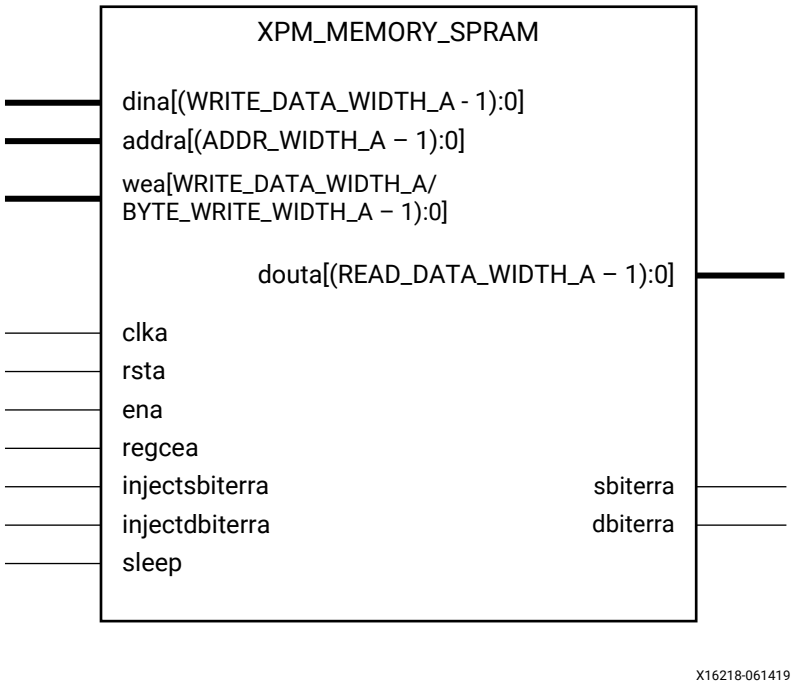
```
                                        // "decode_only" mode).

   .regceb(regceb),                 // 1-bit input: Clock Enable for the last register stage on the output
                                        // data path.

   .rstb(rstb),                     // 1-bit input: Reset signal for the final port B output register stage.
                                    // Synchronously resets output port doutb to the value specified by
                                        // parameter READ_RESET_VALUE_B.

   .sleep(sleep),                   // 1-bit input: sleep signal to enable the dynamic power saving feature.
   .wea(wea)                        // WRITE_DATA_WIDTH_A/BYTE_WRITE_WIDTH_A-bit input: Write enable vector
                                    // for port A input data port dina. 1 bit wide when word-wide writes are
                                    // used. In byte-wide write configurations, each bit controls the
                                        // writing one byte of dina to address addra. For example, to
                                    // synchronously write only bits [15-8] of dina when WRITE_DATA_WIDTH_A
                                        // is 32, wea would be 4'b0010.

);

// End of xpm_memory_sdpram_inst instantiation
```

XPM_MEMORY_SPRAM

Parameterized Macro: Single Port RAM

MACRO_GROUP: XPM

MACRO_SUBGROUP: XPM_MEMORY



Introduction

This macro is used to instantiate Single Port RAM. Reads and writes to the memory can be done through Port A.

The following describes the basic read and write port usage of an XPM_MEMORY instance.

- All synchronous signals are sensitive to the rising edge of clka, which is assumed to be a buffered and toggling clock signal behaving according to target device and memory primitive requirements.
- A read operation is implicitly performed to address addra combinatorially. The data output is registered each clka cycle that ena is asserted.
- Read data appears on the douta port READ_LATENCY_A clka cycles after the associated read operation.
- A write operation is explicitly performed, writing dina to address addra, when both ena and wea are asserted on each clka cycle.

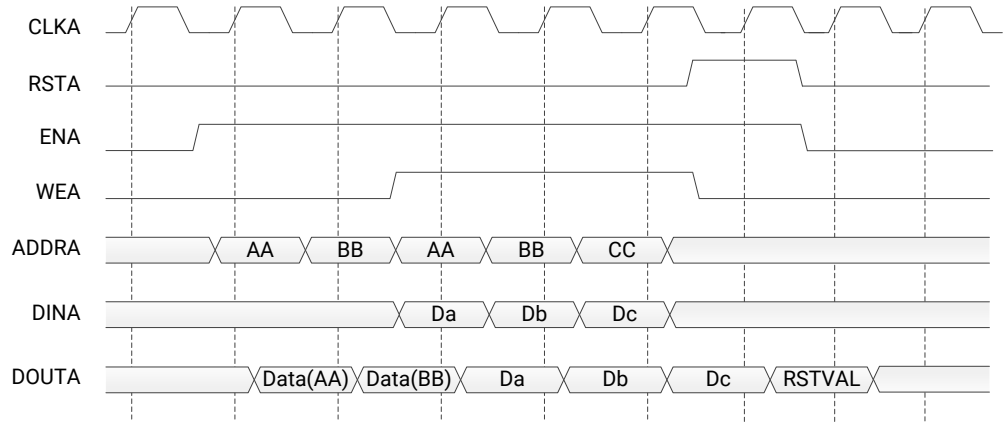
- All read and write operations are gated by the value of ena on the initiating clka cycle, regardless of input or output latencies. The addra and wea inputs have no effect when ena is de-asserted on the coincident clka cycle.
- The behavior of douta with respect to the combination of dina and addra is a function of WRITE_MODE_A.
- For each clka cycle that rsta is asserted, the final output register is immediately but synchronously reset to READ_RESET_VALUE_A, irrespective of READ_LATENCY_A.
- For each clka cycle that regcea is asserted and rsta is de-asserted, the final output register captures and outputs the value from the previous pipeline register.
- Undriven or unknown values provided on module inputs will produce undefined memory array and output port behavior.
- When MEMORY_INIT_PARAM is used, the maximum supported memory size 4K bits.
- Choosing the Invalid Configuration will result in a DRC error.

***Note*:** Writing to an out-of-range address location may overwrite a valid address location when effective address bits match to a physical memory address location.

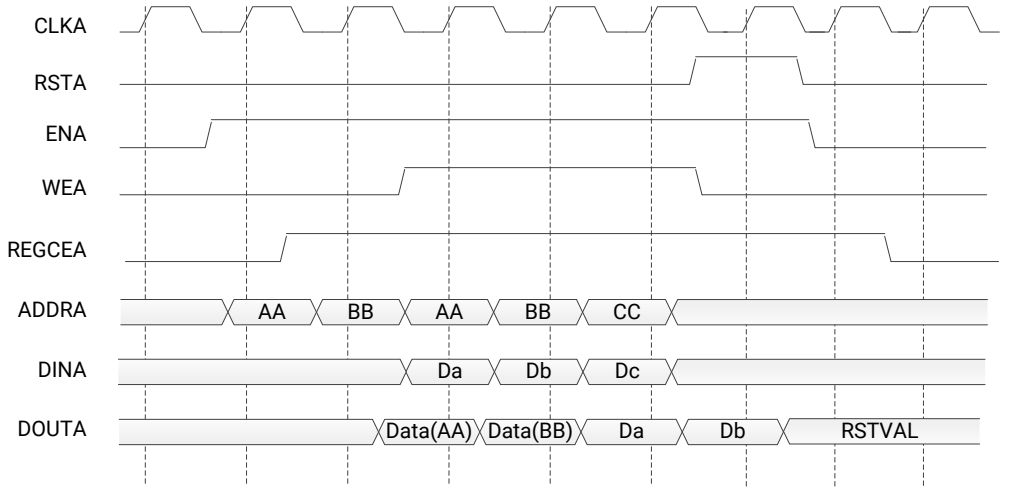
- The use of UltraRAM's dedicated input and output registers are controlled by synthesis based on the READ_LATENCY_B value. For example, if 4 UltraRAMs are in cascade and the READ_LATENCY_B is ≥ 4, then synthesis will absorb as much registers inside UltraRAM primitive as possible.
- For UltraRAM's, OREG enabled when READ_LATENCY_B >= 3 in all write modes.
- For larger memories (≥2 MB), the recommended read latency must be > 8 because the default cascade height used by Vivado synthesis is 8.

Timing Diagrams

SPRAM : Write First Mode with Read Latency of 1

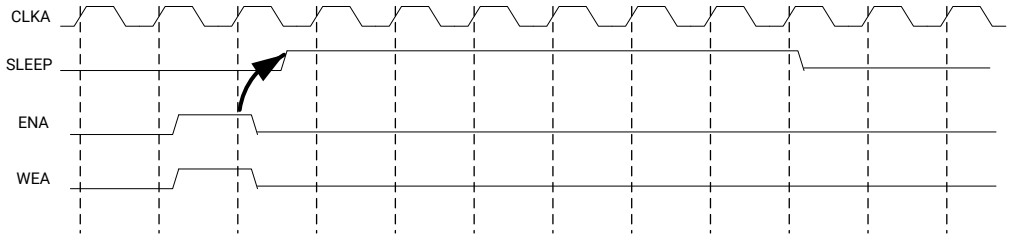


SPRAM : Write First Mode with Read Latency of 2



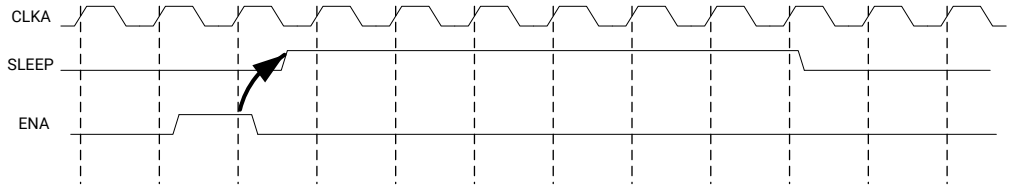
X22985-061319

SPRAM : UltraRAM Limitation on write access before sleep assertion



Write is not allowed in the clock cycle before sleep assertion for UltraRAM configurations

SPRAM : UltraRAM Limitation on read access before sleep assertion



Read is not allowed in the clock cycle before sleep assertion for UltraRAM configurations

X17940-061319

***Note*:** The UltraRAM primitive does not support Write/Read access in the clock cycle just before assertion of sleep gets recognized on the positive edge of the clock when its OREG attribute is set to TRUE. For UltraRAM configurations, Write/Read access to the memory is not allowed in the clock cycle just before the assertion of sleep.

ECC Modes

Only the UltraRAM primitives support ECC when the memory type is set to Single Port RAM. The three ECC modes supported are:

- Both encode and decode
- Encode only
- Decode only

The read and write usage of the three ECC Modes are the same as described in the Introduction section above. See the “Built-in Error Correction” section of the *7 Series FPGAs Memory Resources User Guide* (UG473) for more details on this feature like Error Injection and syndrome bits calculations.

There are restrictions on the attributes WRITE_DATA_WIDTH_A, READ_DATA_WIDTH_A, and MEMORY_SIZE in each of the above ECC modes.

- **Both encode and decode** WRITE_DATA_WIDTH_A and READ_DATA_WIDTH_A must be multiples of 64-bits. Violating this rule will results in a DRC in XPM_Memory.

- **Encode only** WRITE_DATA_WIDTH_A must be a multiple of 64 bits and READ_DATA_WIDTH_A must be a multiple of 72-bits. MEMORY_SIZE must be a multiple of READ_DATA_WIDTH_A. Violating these rules will result in a DRC.
- **Decode only** WRITE_DATA_WIDTH_A must be a multiple of 72 bits and READ_DATA_WIDTH_A must be a multiple of 64-bits. MEMORY_SIZE must be a multiple of WRITE_DATA_WIDTH_A. Violating these rules will result in a DRC.

When ECC is enabled the following are not supported:

- Asymmetry
- Initialization
- Reset (neither non-zero reset value nor reset assertion)

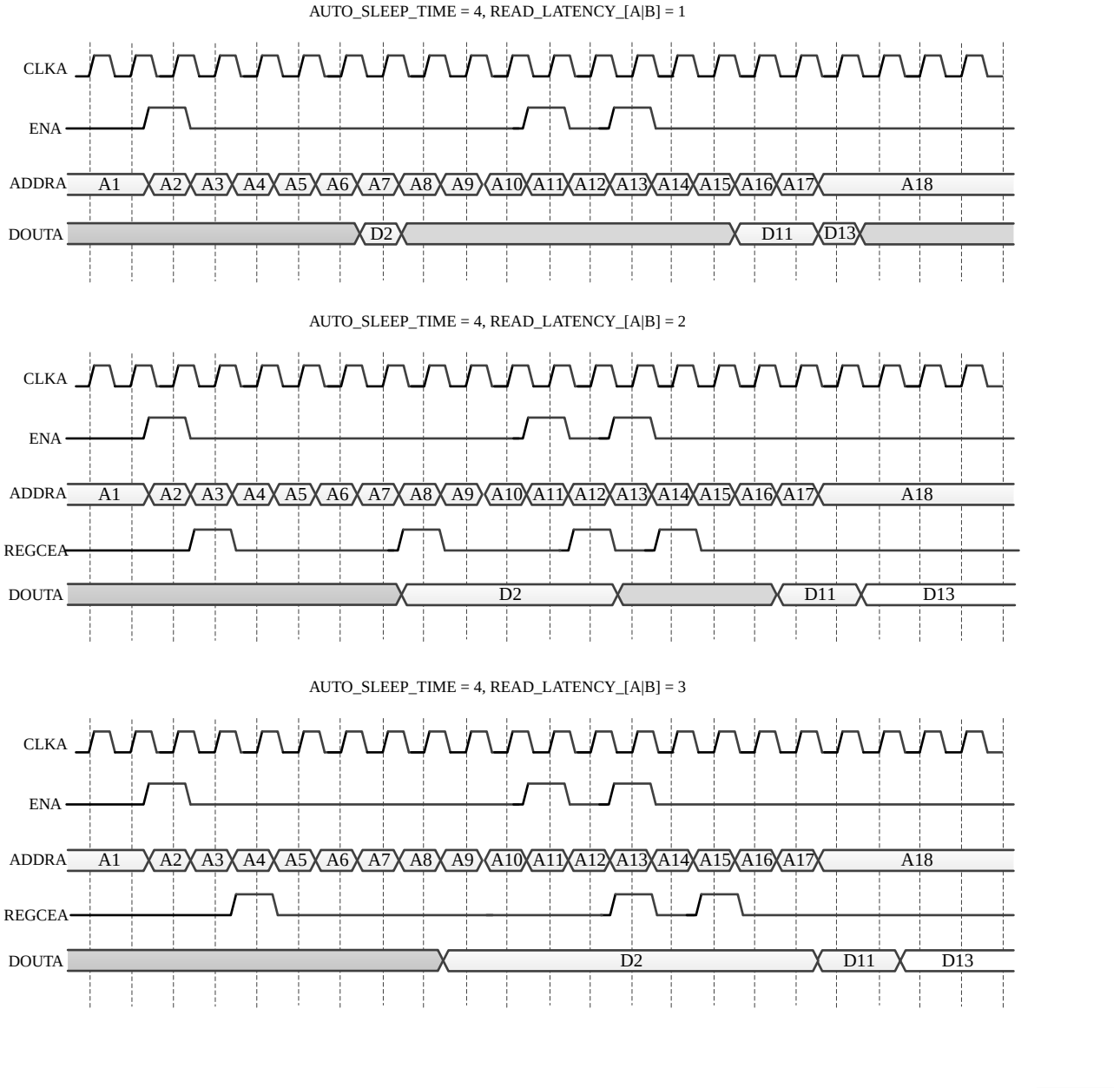
Note: ECC uses a hard-ECC block available in the BRAM/URAM macro and the data width should be multiples of 64/72. Use ECC IP for other data width combinations.

Auto Sleep Mode

- This feature is applicable only when MEMORY_PRIMITIVE is URAM and is controlled internally in the UltraRAM to check if it can be put in sleep mode and when it needs to wake up. Thus power savings are obtained automatically without having to explicitly control the SLEEP Pin.
- When AUTO_SLEEP_TIME is 0, the feature is disabled. When AUTO_SLEEP_TIME is non-zero, XPM_MEMORY constructs the pipeline registers equal to AUTO_SLEEP_TIME value on all input signals except rst[a|b].
- If AUTO_SLEEP_TIME is too low, then UltraRAM goes into sleep and wakeup too often, which can cause more power to be consumed.
- The number of sleep cycles achieved is calculated by following formula:
  - If number of consecutive inactive cycles is <AUTO_SLEEP_TIME, then number of sleep cycles = 0.
  - If number of consecutive inactive cycles is ≥AUTO_SLEEP_TIME, Then number of consecutive sleep cycles = Number of consecutive inactive cycles – 3.
  - Inactive cycle is defined as a cycle where there is no Read/Write operation from either port.
- The latency between the read operation and the data arrival at dout[a|b] is AUTO_SLEEP_TIME + READ_LATENCY_[A|B] clock cycles (Assuming that REGCE is high when the output data pipe line exists).
- When the READ_LATENCY_[A|B] is set to 1 or 2, XPM_Memory behaviorally models the AUTO SLEEP feature and forces ‘x’ on DOUT[A|B] when the RAM is in Auto Sleep Mode. For READ_LATENCY_[A|B] greater than 2, the propagation of ‘x’ cannot happen to the DOUT[A|B] as the output registers gets the clock enable (delayed read enable) after UltraRAM comes out of sleep mode.

- The Auto Sleep mode is most effective for larger memory sizes or any memory with very little activity.

Timing diagrams for Auto Sleep Mode at various read latencies are shown below.



Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
addra	Input	ADDR_WIDTH_A	clka	NA	Active	Address for port A write and read operations.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
clka	Input	1	NA	EDGE _RISING	Active	Clock signal for port A.
dbiterra	Output	1	clka	LEVEL _HIGH	DoNotCare	Status signal to indicate double bit error occurrence on the data output of port A.
dina	Input	WRITE _DATA _WIDTH _A	clka	NA	Active	Data input for port A write operations.
douta	Output	READ _DATA _WIDTH _A	clka	NA	Active	Data output for port A read operations.
ena	Input	1	clka	LEVEL _HIGH	Active	Memory enable signal for port A. Must be high on clock cycles when read or write operations are initiated. Pipelined internally.
injectdbiterra	Input	1	clka	LEVEL _HIGH	0	Controls double bit error injection on input data when ECC enabled (Error injection capability is not available in "decode_only" mode).
injectsbiterra	Input	1	clka	LEVEL _HIGH	0	Controls single bit error injection on input data when ECC enabled (Error injection capability is not available in "decode_only" mode).
regcea	Input	1	clka	LEVEL _HIGH	1	Clock Enable for the last register stage on the output data path.
rsta	Input	1	clka	LEVEL _HIGH	Active	Reset signal for the final port A output register stage. Synchronously resets output port douta to the value specified by parameter READ_RESET_VALUE_A.
sbiterra	Output	1	clka	LEVEL _HIGH	DoNotCare	Status signal to indicate single bit error occurrence on the data output of port A.
sleep	Input	1	NA	LEVEL _HIGH	0	sleep signal to enable the dynamic power saving feature.
wea	Input	WRITE _DATA _WIDTH _A / BYTE _WRITE _WIDTH _A	clka	LEVEL _HIGH	Active	Write enable vector for port A input data port dina. 1 bit wide when word-wide writes are used. In byte-wide write configurations, each bit controls the writing one byte of dina to address addra. For example, to synchronously write only bits [15-8] of dina when WRITE_DATA_WIDTH_A is 32, wea would be 4'b0010.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ADDR_WIDTH_A	DECIMAL	1 to 20	6	Specify the width of the port A address port addra, in bits. Must be large enough to access the entire memory from port A, i.e. >= $clog2(MEMORY_SIZE/[WRITE|READ]_DATA_WIDTH_A).
AUTO_SLEEP_TIME	DECIMAL	0 to 15	0	Specify the number of clka cycles to auto-sleep, if feature is available in architecture. • 0 - Disable auto-sleep feature • 3-15 - Number of auto-sleep latency cycles Do not change from the value provided in the template instantiation.
BYTE_WRITE_WIDTH _A	DECIMAL	1 to 4608	32	To enable byte-wide writes on port A, specify the byte width, in bits. • 8- 8-bit byte-wide writes, legal when WRITE_DATA_WIDTH_A is an integer multiple of 8 • 9- 9-bit byte-wide writes, legal when WRITE_DATA_WIDTH_A is an integer multiple of 9 Or to enable word-wide writes on port A, specify the same value as for WRITE_DATA_WIDTH_A.
CASCADE_HEIGHT	DECIMAL	0 to 64	0	0- No Cascade Height, Allow Vivado Synthesis to choose. 1 or more - Vivado Synthesis sets the specified value as Cascade Height.
ECC_MODE	STRING	"no_ecc", "both _encode _and _decode", "decode _only", "encode _only"	"no_ecc"	• "no_ecc" - Disables ECC • "encode_only" - Enables ECC Encoder only • "decode_only" - Enables ECC Decoder only • "both_encode_and_decode" - Enables both ECC Encoder and Decoder
MEMORY_INIT_FILE	STRING	String	"none"	Specify "none" (including quotes) for no memory initialization, or specify the name of a memory initialization file- Enter only the name of the file with .mem extension, including quotes but without path (e.g. "my_file.mem"). File format must be ASCII and consist of only hexadecimal values organized into the specified depth by narrowest data width generic value of the memory. See the Memory File (MEM) section for more information on the syntax. Initialization of memory happens through the file name specified only when parameter MEMORY_INIT_PARAM value is equal to "". When using XPM_MEMORY in a project, add the specified file to the Vivado project as a design source.

Attribute	Type	Allowed Values	Default	Description
MEMORY_INIT _PARAM	STRING	String	"0"	Specify "" or "0" (including quotes) for no memory initialization through parameter, or specify the string containing the hex characters. Enter only hex characters with each location separated by delimiter (,). Parameter format must be ASCII and consist of only hexadecimal values organized into the specified depth by narrowest data width generic value of the memory. For example, if the narrowest data width is 8, and the depth of memory is 8 locations, then the parameter value should be passed as shown below. parameter MEMORY_INIT_PARAM = "AB,CD,EF,1,2,34,56,78" Where "AB" is the 0th location and "78" is the 7th location.
MEMORY _OPTIMIZATION	STRING	"true", "false"	"true"	Specify "true" to enable the optimization of unused memory or bits in the memory structure. Specify "false" to disable the optimization of unused memory or bits in the memory structure
MEMORY _PRIMITIVE	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the memory primitive (resource type) to use. • "auto"- Allow Vivado Synthesis to choose • "distributed"- Distributed memory • "block"- Block memory • "ultra"- Ultra RAM memory NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with MEMORY_PRIMITIVE set to "auto".
MEMORY_SIZE	DECIMAL	2 to 150994944	2048	Specify the total memory array size, in bits. For example, enter 65536 for a 2kx32 RAM. • When ECC is enabled and set to "encode_only", then the memory size has to be multiples of READ_DATA_WIDTH_A • When ECC is enabled and set to "decode_only", then the memory size has to be multiples of WRITE_DATA_WIDTH_A
MESSAGE_CONTROL	DECIMAL	0 to 1	0	Specify 1 to enable the dynamic message reporting such as collision warnings, and 0 to disable the message reporting

Attribute	Type	Allowed Values	Default	Description
READ_DATA_WIDTH_A	DECIMAL	1 to 4608	32	Specify the width of the port A read data output port douta, in bits. The values of READ_DATA_WIDTH_A and WRITE_DATA_WIDTH_A must be equal. When ECC is enabled and set to "encode_only", then READ_DATA_WIDTH_A has to be multiples of 72-bits. When ECC is enabled and set to "decode_only" or "both_encode_and_decode", then READ_DATA_WIDTH_A has to be multiples of 64-bits.
READ_LATENCY_A	DECIMAL	0 to 100	2	Specify the number of register stages in the port A read data pipeline. Read data output to port douta takes this number of clka cycles. • To target block memory, a value of 1 or larger is required- 1 causes use of memory latch only; 2 causes use of output register. • To target distributed memory, a value of 0 or larger is required- 0 indicates combinatorial output. • Values larger than 2 synthesize additional flip-flops that are not retimed into memory primitives.
READ_RESET_VALUE _A	STRING	String	"0"	Specify the reset value of the port A final output register stage in response to rsta input port is assertion. Since this parameter is a string, you must specify the hex values inside double quotes. For example, If the read data width is 8, then specify READ_RESET_VALUE_A = "EA"; When ECC is enabled, then reset value is not supported.
RST_MODE_A	STRING	"SYNC", "ASYNC"	"SYNC"	Describes the behaviour of the reset • "SYNC" - when reset is applied, synchronously resets output port douta to the value specified by parameter READ_RESET_VALUE_A • "ASYNC" - when reset is applied, asynchronously resets output port douta to zero
SIM_ASSERT_CHK	DECIMAL	0 to 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.
USE_MEM_INIT	DECIMAL	0 to 1	1	Specify 1 to enable the generation of below message and 0 to disable generation of the following message completely. "INFO - MEMORY_INIT_FILE and MEMORY_INIT_PARAM together specifies no memory initialization. Initial memory contents will be all 0s." NOTE: This message gets generated only when there is no Memory Initialization specified either through file or Parameter.

Attribute	Type	Allowed Values	Default	Description
WAKEUP_TIME	STRING	"disable _sleep", "use _sleep _pin"	"disable _sleep"	Specify "disable_sleep" to disable dynamic power saving option, and specify "use_sleep_pin" to enable the dynamic power saving option
WRITE_DATA_WIDTH_A	DECIMAL	1 to 4608	32	Specify the width of the port A write data input port dina, in bits. The values of WRITE_DATA_WIDTH_A and READ_DATA_WIDTH_A must be equal. When ECC is enabled and set to "encode_only" or "both_encode_and_decode", then WRITE_DATA_WIDTH_A must be multiples of 64-bits. When ECC is enabled and set to "decode_only", then WRITE_DATA_WIDTH_A must be multiples of 72-bits.
WRITE_MODE_A	STRING	"read _first", "no _change", "write _first"	"read _first"	Write mode behavior for port A output data port, douta.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_memory_spram: Single Port RAM
-- Xilinx Parameterized Macro, version 2020.1

xpm_memory_spram_inst : xpm_memory_spram
generic map (
   ADDR_WIDTH_A => 6,               -- DECIMAL
   AUTO_SLEEP_TIME => 0,            -- DECIMAL
   BYTE_WRITE_WIDTH_A => 32,        -- DECIMAL
   CASCADE_HEIGHT => 0,             -- DECIMAL
   ECC_MODE => "no_ecc",            -- String
   MEMORY_INIT_FILE => "none",      -- String
   MEMORY_INIT_PARAM => "0",        -- String
   MEMORY_OPTIMIZATION => "true",   -- String
   MEMORY_PRIMITIVE => "auto",      -- String
   MEMORY_SIZE => 2048,             -- DECIMAL
   MESSAGE_CONTROL => 0,            -- DECIMAL
   READ_DATA_WIDTH_A => 32,         -- DECIMAL
   READ_LATENCY_A => 2,             -- DECIMAL
   READ_RESET_VALUE_A => "0",       -- String
   RST_MODE_A => "SYNC",            -- String
   SIM_ASSERT_CHK => 0,          -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   USE_MEM_INIT => 1,               -- DECIMAL
   WAKEUP_TIME => "disable_sleep", -- String
   WRITE_DATA_WIDTH_A => 32,        -- DECIMAL
   WRITE_MODE_A => "read_first"     -- String
)
port map (
   dbiterra => dbiterra,            -- 1-bit output: Status signal to indicate double bit error occurrence
                                    --   on the data output of port A.

   douta => douta,                  -- READ_DATA_WIDTH_A-bit output: Data output for port A read operations.
   sbiterra => sbiterra,            -- 1-bit output: Status signal to indicate single bit error occurrence
                                    --   on the data output of port A.

   addra => addra,                  -- ADDR_WIDTH_A-bit input: Address for port A write and read operations.
   clka => clka,                    --   1-bit input: Clock signal for port A.
   dina => dina,                    -- WRITE_DATA_WIDTH_A-bit input: Data input for port A write operations.
   ena => ena,                      -- 1-bit input: Memory enable signal for port A. Must be high on clock
```

```
                                    -- cycles when read or write operations are initiated. Pipelined
                                       -- internally.

   injectdbiterra => injectdbiterra, -- 1-bit input: Controls double bit error injection on input data when
                                       -- ECC enabled (Error injection capability is not available in
                                       -- "decode_only" mode).

   injectsbiterra => injectsbiterra, -- 1-bit input: Controls single bit error injection on input data when
                                       -- ECC enabled (Error injection capability is not available in
                                       -- "decode_only" mode).

   regcea => regcea,                 -- 1-bit input: Clock Enable for the last register stage on the output
                                       -- data path.

   rsta => rsta,                     -- 1-bit input: Reset signal for the final port A output register
                                     -- stage. Synchronously resets output port douta to the value specified
                                       -- by parameter READ_RESET_VALUE_A.

   sleep => sleep,                   -- 1-bit input: sleep signal to enable the dynamic power saving feature.
   wea => wea                        -- WRITE_DATA_WIDTH_A/BYTE_WRITE_WIDTH_A-bit input: Write enable vector
                                     -- for port A input data port dina. 1 bit wide when word-wide writes
                                     -- are used. In byte-wide write configurations, each bit controls the
                                       -- writing one byte of dina to address addra. For example, to
                                     -- synchronously write only bits [15-8] of dina when WRITE_DATA_WIDTH_A
                                       -- is 32, wea would be 4'b0010.

);

-- End of xpm_memory_spram_inst instantiation
```

Verilog Instantiation Template

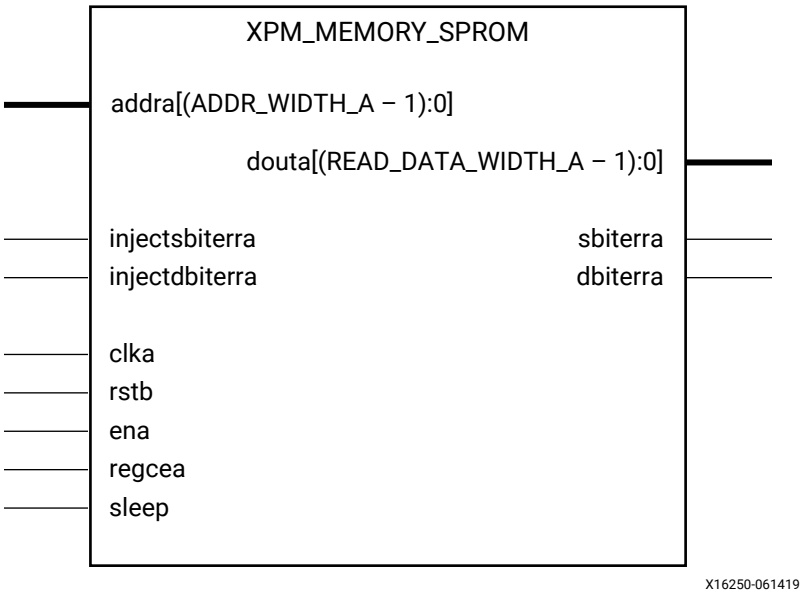
```
// xpm_memory_spram: Single Port RAM
// Xilinx Parameterized Macro, version 2020.1

xpm_memory_spram #(
   .ADDR_WIDTH_A(6),               // DECIMAL
   .AUTO_SLEEP_TIME(0),            // DECIMAL
   .BYTE_WRITE_WIDTH_A(32),        // DECIMAL
   .CASCADE_HEIGHT(0),             // DECIMAL
   .ECC_MODE("no_ecc"),            // String
   .MEMORY_INIT_FILE("none"),      // String
   .MEMORY_INIT_PARAM("0"),        // String
   .MEMORY_OPTIMIZATION("true"),   // String
   .MEMORY_PRIMITIVE("auto"),      // String
   .MEMORY_SIZE(2048),             // DECIMAL
   .MESSAGE_CONTROL(0),            // DECIMAL
   .READ_DATA_WIDTH_A(32),         // DECIMAL
   .READ_LATENCY_A(2),             // DECIMAL
   .READ_RESET_VALUE_A("0"),       // String
   .RST_MODE_A("SYNC"),            // String
   .SIM_ASSERT_CHK(0),           // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .USE_MEM_INIT(1),               // DECIMAL
   .WAKEUP_TIME("disable_sleep"), // String
   .WRITE_DATA_WIDTH_A(32),        // DECIMAL
   .WRITE_MODE_A("read_first")     // String
)
xpm_memory_spram_inst (
   .dbiterra(dbiterra),            // 1-bit output: Status signal to indicate double bit error occurrence
                                     // on the data output of port A.

   .douta(douta),                  // READ_DATA_WIDTH_A-bit output: Data output for port A read operations.
   .sbiterra(sbiterra),            // 1-bit output: Status signal to indicate single bit error occurrence
                                     // on the data output of port A.

   .addra(addra),                  // ADDR_WIDTH_A-bit input: Address for port A write and read operations.
   .clka(clka),                      // 1-bit input: Clock signal for port A.
   .dina(dina),                    // WRITE_DATA_WIDTH_A-bit input: Data input for port A write operations.
   .ena(ena),                      // 1-bit input: Memory enable signal for port A. Must be high on clock
                                     // cycles when read or write operations are initiated. Pipelined
                                     // internally.

   .injectdbiterra(injectdbiterra), // 1-bit input: Controls double bit error injection on input data when
                                     // ECC enabled (Error injection capability is not available in
                                     // "decode_only" mode).
```

```
   .injectsbiterra(injectsbiterra), // 1-bit input: Controls single bit error injection on input data when
                                    // ECC enabled (Error injection capability is not available in
                                    // "decode_only" mode).

   .regcea(regcea),                 // 1-bit input: Clock Enable for the last register stage on the output
                                    // data path.

   .rsta(rsta),                     // 1-bit input: Reset signal for the final port A output register stage.
                                    // Synchronously resets output port douta to the value specified by
                                    // parameter READ_RESET_VALUE_A.

   .sleep(sleep),                   // 1-bit input: sleep signal to enable the dynamic power saving feature.
   .wea(wea)                        // WRITE_DATA_WIDTH_A/BYTE_WRITE_WIDTH_A-bit input: Write enable vector
                                    // for port A input data port dina. 1 bit wide when word-wide writes are
                                    // used. In byte-wide write configurations, each bit controls the
                                    // writing one byte of dina to address addra. For example, to
                                    // synchronously write only bits [15-8] of dina when WRITE_DATA_WIDTH_A
                                    // is 32, wea would be 4'b0010.

);

// End of xpm_memory_spram_inst instantiation
```

XPM_MEMORY_SPROM

Parameterized Macro: Single Port ROM

MACRO_GROUP: XPM
MACRO_SUBGROUP: XPM_MEMORY



Introduction

This macro is used to instantiate Single Port ROM. Read operations from the memory can be performed from Port A.

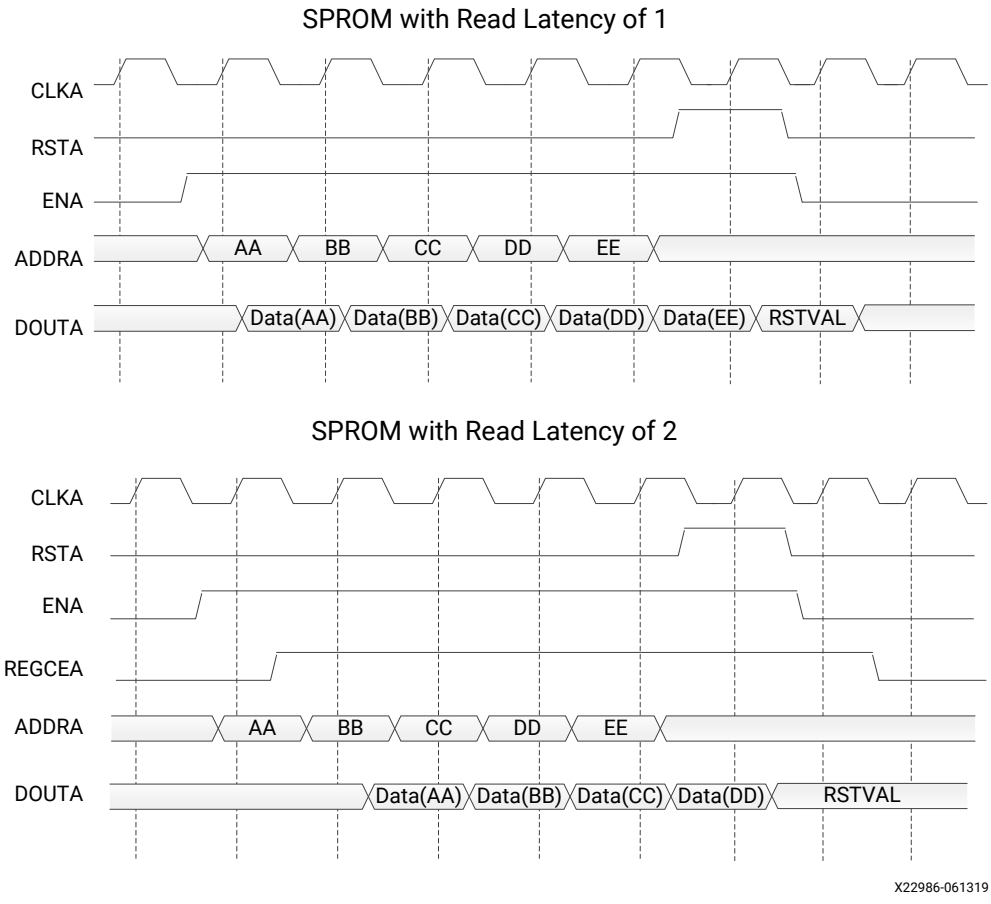
The following describes the basic read and write port usage of an XPM_MEMORY instance.

- All synchronous signals are sensitive to the rising edge of clka, which is assumed to be a buffered and toggling clock signal behaving according to target device and memory primitive requirements.
- A read operation is implicitly performed to address addra combinatorially. The data output is registered each clka cycle that ena is asserted.
- Read data appears on the douta port READ_LATENCY_A CLKA cycles after the associated read operation.
- All read operations are gated by the value of ena on the initiating clka cycle, regardless of input or output latencies.

- For each clka cycle that rsta is asserted, the final output register is immediately but synchronously reset to READ_RESET_VALUE_A, irrespective of READ_LATENCY_A.
- For each clka cycle that regcea is asserted and rsta is de-asserted, the final output register captures and outputs the value from the previous pipeline register.
- Undriven or unknown values provided on module inputs will produce undefined memory array and output port behavior.
- When MEMORY_INIT_PARAM is used, the maximum supported memory size 4K bits.
- WRITE_MODE_A must be set to “read_first” in Single Port ROM configurations. Violating this will result in a DRC error.

***Note*:** For larger memories (≥2 MB), the recommended read latency must be > 8 because the default cascade height used by Vivado synthesis is 8.

Timing Diagrams



Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
addra	Input	ADDR_WIDTH_A	clka	NA	Active	Address for port A read operations.
clka	Input	1	NA	EDGE_RISING	Active	Clock signal for port A.
dbiterra	Output	1	clka	LEVEL_HIGH	DoNotCare	Leave open.
douta	Output	READ_DATA_WIDTH_A	clka	NA	Active	Data output for port A read operations.
ena	Input	1	clka	LEVEL_HIGH	Active	Memory enable signal for port A. Must be high on clock cycles when read operations are initiated. Pipelined internally.
injectdbiterra	Input	1	clka	LEVEL_HIGH	0	Do not change from the provided value.
injectsbiterra	Input	1	clka	LEVEL_HIGH	0	Do not change from the provided value.
regcea	Input	1	clka	LEVEL_HIGH	1	Do not change from the provided value.
rsta	Input	1	clka	LEVEL_HIGH	Active	Reset signal for the final port A output register stage. Synchronously resets output port douta to the value specified by parameter READ_RESET_VALUE_A.
sbiterra	Output	1	clka	LEVEL_HIGH	DoNotCare	Leave open.
sleep	Input	1	NA	LEVEL_HIGH	0	sleep signal to enable the dynamic power saving feature.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ADDR_WIDTH_A	DECIMAL	1 to 20	6	Specify the width of the port A address port addra, in bits. Must be large enough to access the entire memory from port A, i.e. >= $clog2(MEMORY_SIZE/READ_DATA_WIDTH_A).
AUTO_SLEEP_TIME	DECIMAL	0 to 15	0	Must be set to 0 0 - Disable auto-sleep feature

Attribute	Type	Allowed Values	Default	Description
CASCADE_HEIGHT	DECIMAL	0 to 64	0	0- No Cascade Height, Allow Vivado Synthesis to choose. 1 or more - Vivado Synthesis sets the specified value as Cascade Height.
ECC_MODE	STRING	"no_ecc", "both _encode _and _decode", "decode _only", "encode _only"	"no_ecc"	• "no_ecc" - Disables ECC • "encode_only" - Enables ECC Encoder only • "decode_only" - Enables ECC Decoder only • "both_encode_and_decode" - Enables both ECC Encoder and Decoder
MEMORY_INIT_FILE	STRING	String	"none"	Specify "none" (including quotes) for no memory initialization, or specify the name of a memory initialization file- Enter only the name of the file with .mem extension, including quotes but without path (e.g. "my_file.mem"). File format must be ASCII and consist of only hexadecimal values organized into the specified depth by narrowest data width generic value of the memory. See the Memory File (MEM) section for more information on the syntax. Initialization of memory happens through the file name specified only when parameter MEMORY_INIT_PARAM value is equal to "". When using XPM_MEMORY in a project, add the specified file to the Vivado project as a design source.
MEMORY_INIT _PARAM	STRING	String	"0"	Specify "" or "0" (including quotes) for no memory initialization through parameter, or specify the string containing the hex characters. Enter only hex characters with each location separated by delimiter (,). Parameter format must be ASCII and consist of only hexadecimal values organized into the specified depth by narrowest data width generic value of the memory. For example, if the narrowest data width is 8, and the depth of memory is 8 locations, then the parameter value should be passed as shown below. parameter MEMORY_INIT_PARAM = "AB,CD,EF,1,2,34,56,78" Where "AB" is the 0th location and "78" is the 7th location.
MEMORY _OPTIMIZATION	STRING	"true", "false"	"true"	Specify "true" to enable the optimization of unused memory or bits in the memory structure. Specify "false" to disable the optimization of unused memory or bits in the memory structure
MEMORY _PRIMITIVE	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the memory primitive (resource type) to use- "auto"- Allow Vivado Synthesis to choose "distributed"- Distributed memory "block"- Block memory
MEMORY_SIZE	DECIMAL	2 to 150994944	2048	Specify the total memory array size, in bits. For example, enter 65536 for a 2kx32 ROM.
MESSAGE_CONTROL	DECIMAL	0 to 1	0	Specify 1 to enable the dynamic message reporting such as collision warnings, and 0 to disable the message reporting

Attribute	Type	Allowed Values	Default	Description
READ_DATA_WIDTH_A	DECIMAL	1 to 4608	32	Specify the width of the port A read data output port douta, in bits.
READ_LATENCY_A	DECIMAL	0 to 100	2	Specify the number of register stages in the port A read data pipeline. Read data output to port douta takes this number of clka cycles. To target block memory, a value of 1 or larger is required- 1 causes use of memory latch only; 2 causes use of output register. To target distributed memory, a value of 0 or larger is required- 0 indicates combinatorial output. Values larger than 2 synthesize additional flip-flops that are not retimed into memory primitives.
READ_RESET_VALUE _A	STRING	String	"0"	Specify the reset value of the port A final output register stage in response to rsta input port is assertion. For example, to reset the value of port douta to all 0s when READ_DATA_WIDTH_A is 32, specify 32HHHHh0.
RST_MODE_A	STRING	"SYNC", "ASYNC"	"SYNC"	Describes the behaviour of the reset • "SYNC" - when reset is applied, synchronously resets output port douta to the value specified by parameter READ_RESET_VALUE_A • "ASYNC" - when reset is applied, asynchronously resets output port douta to zero
SIM_ASSERT_CHK	DECIMAL	0 to 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.
USE_MEM_INIT	DECIMAL	0 to 1	1	Specify 1 to enable the generation of below message and 0 to disable generation of the following message completely. "INFO - MEMORY_INIT_FILE and MEMORY_INIT_PARAM together specifies no memory initialization. Initial memory contents will be all 0s." NOTE: This message gets generated only when there is no Memory Initialization specified either through file or Parameter.
WAKEUP_TIME	STRING	"disable _sleep", "use _sleep _pin"	"disable _sleep"	Specify "disable_sleep" to disable dynamic power saving option, and specify "use_sleep_pin" to enable the dynamic power saving option

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_memory_sprom: Single Port ROM
-- Xilinx Parameterized Macro, version 2020.1

xpm_memory_sprom_inst : xpm_memory_sprom
generic map (
   ADDR_WIDTH_A => 6,               -- DECIMAL
   AUTO_SLEEP_TIME => 0,            -- DECIMAL
   CASCADE_HEIGHT => 0,             -- DECIMAL
   ECC_MODE => "no_ecc",            -- String
   MEMORY_INIT_FILE => "none",      -- String
   MEMORY_INIT_PARAM => "0",        -- String
   MEMORY_OPTIMIZATION => "true",   -- String
   MEMORY_PRIMITIVE => "auto",      -- String
   MEMORY_SIZE => 2048,             -- DECIMAL
   MESSAGE_CONTROL => 0,            -- DECIMAL
   READ_DATA_WIDTH_A => 32,         -- DECIMAL
   READ_LATENCY_A => 2,             -- DECIMAL
   READ_RESET_VALUE_A => "0",       -- String
   RST_MODE_A => "SYNC",            -- String
   SIM_ASSERT_CHK => 0,           -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   USE_MEM_INIT => 1,               -- DECIMAL
   WAKEUP_TIME => "disable_sleep"   -- String
)
port map (
   dbiterra => dbiterra,              -- 1-bit output: Leave open.
   douta => douta,                  -- READ_DATA_WIDTH_A-bit output: Data output for port A read operations.
   sbiterra => sbiterra,              -- 1-bit output: Leave open.
   addra => addra,                    -- ADDR_WIDTH_A-bit input: Address for port A read operations.
   clka => clka,                      -- 1-bit input: Clock signal for port A.
   ena => ena,                      -- 1-bit input: Memory enable signal for port A. Must be high on clock
                                    -- cycles when read operations are initiated. Pipelined internally.

   injectdbiterra => injectdbiterra, -- 1-bit input: Do not change from the provided value.
   injectsbiterra => injectsbiterra, -- 1-bit input: Do not change from the provided value.
   regcea => regcea,                  -- 1-bit input: Do not change from the provided value.
   rsta => rsta,                    -- 1-bit input: Reset signal for the final port A output register
                                    -- stage. Synchronously resets output port douta to the value specified
                                       -- by parameter READ_RESET_VALUE_A.

   sleep => sleep                   -- 1-bit input: sleep signal to enable the dynamic power saving feature.
);

-- End of xpm_memory_sprom_inst instantiation
```

Verilog Instantiation Template

```
// xpm_memory_sprom: Single Port ROM
// Xilinx Parameterized Macro, version 2020.1

xpm_memory_sprom #(
   .ADDR_WIDTH_A(6),               // DECIMAL
   .AUTO_SLEEP_TIME(0),            // DECIMAL
   .CASCADE_HEIGHT(0),             // DECIMAL
   .ECC_MODE("no_ecc"),            // String
   .MEMORY_INIT_FILE("none"),      // String
   .MEMORY_INIT_PARAM("0"),        // String
   .MEMORY_OPTIMIZATION("true"),   // String
   .MEMORY_PRIMITIVE("auto"),      // String
   .MEMORY_SIZE(2048),             // DECIMAL
   .MESSAGE_CONTROL(0),            // DECIMAL
   .READ_DATA_WIDTH_A(32),         // DECIMAL
   .READ_LATENCY_A(2),             // DECIMAL
   .READ_RESET_VALUE_A("0"),       // String
```

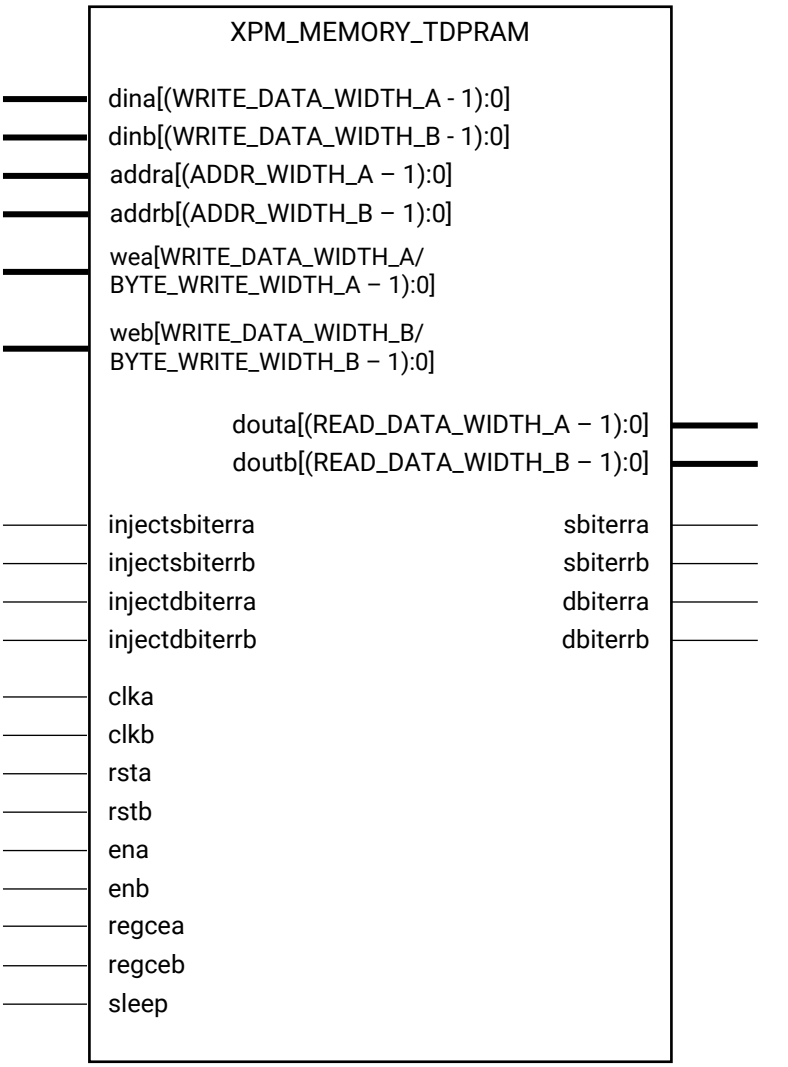
```
   .RST_MODE_A("SYNC"),            // String
   .SIM_ASSERT_CHK(0),          // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .USE_MEM_INIT(1),               // DECIMAL
   .WAKEUP_TIME("disable_sleep")  // String
)
xpm_memory_sprom_inst (
   .dbiterra(dbiterra),              // 1-bit output: Leave open.
   .douta(douta),                  // READ_DATA_WIDTH_A-bit output: Data output for port A read operations.
   .sbiterra(sbiterra),              // 1-bit output: Leave open.
   .addra(addra),                    // ADDR_WIDTH_A-bit input: Address for port A read operations.
   .clka(clka),                      // 1-bit input: Clock signal for port A.
   .ena(ena),                      // 1-bit input: Memory enable signal for port A. Must be high on clock
                                   // cycles when read operations are initiated. Pipelined internally.

   .injectdbiterra(injectdbiterra), // 1-bit input: Do not change from the provided value.
   .injectsbiterra(injectsbiterra), // 1-bit input: Do not change from the provided value.
   .regcea(regcea),                  // 1-bit input: Do not change from the provided value.
   .rsta(rsta),                    // 1-bit input: Reset signal for the final port A output register stage.
                                   // Synchronously resets output port douta to the value specified by
                                      // parameter READ_RESET_VALUE_A.

   .sleep(sleep)                   // 1-bit input: sleep signal to enable the dynamic power saving feature.
);

// End of xpm_memory_sprom_inst instantiation
```

XPM_MEMORY_TDPRAM

Parameterized Macro: True Dual Port RAM

MACRO_GROUP: XPM

MACRO_SUBGROUP: XPM_MEMORY



X16251-061419

Introduction

This macro is used to instantiate True Dual Port RAM. Reads and writes to the memory can be done through port A and port B simultaneously.

The following describes the basic read and write port usage of an XPM_MEMORY instance. It does not distinguish between port A and port B.

- All synchronous signals are sensitive to the rising edge of clk[a|b], which is assumed to be a buffered and toggling clock signal behaving according to target device and memory primitive requirements.
- A read operation is implicitly performed to address addr[a|b] combinatorially. The data output is registered each clk[a|b] cycle that en[a|b] is asserted.
- Read data appears on the dout[a|b] port READ_LATENCY_[A|B] clk[a|b] cycles after the associated read operation.
- A write operation is explicitly performed, writing din[a|b] to address addr[a|b], when both en[a|b] and we[a|b] are asserted on each clk[a|b] cycle.
- All read and write operations are gated by the value of en[a|b] on the initiating clk[a|b] cycle, regardless of input or output latencies. The addr[a|b] and we[a|b] inputs have no effect when en[a|b] is de-asserted on the coincident clk[a|b] cycle.
- The behavior of dout[a|b] with respect to the combination of din[a|b] and addr[a|b] is a function of WRITE_MODE_[A|B].
- For each clk[a|b] cycle that rst[a|b] is asserted, the final output register is immediately but synchronously reset to READ_RESET_VALUE_[A|B], irrespective of READ_LATENCY_[A|B].
- For each clk[a|b] cycle that regce[a|b] is asserted and rst[a|b] is de-asserted, the final output register captures and outputs the value from the previous pipeline register.
- Undriven or unknown values provided on module inputs will produce undefined memory array and output port behavior.
- When MEMORY_INIT_PARAM is used, the maximum supported memory size 4K bits.
- Choosing the Invalid Configuration will result in a DRC error.

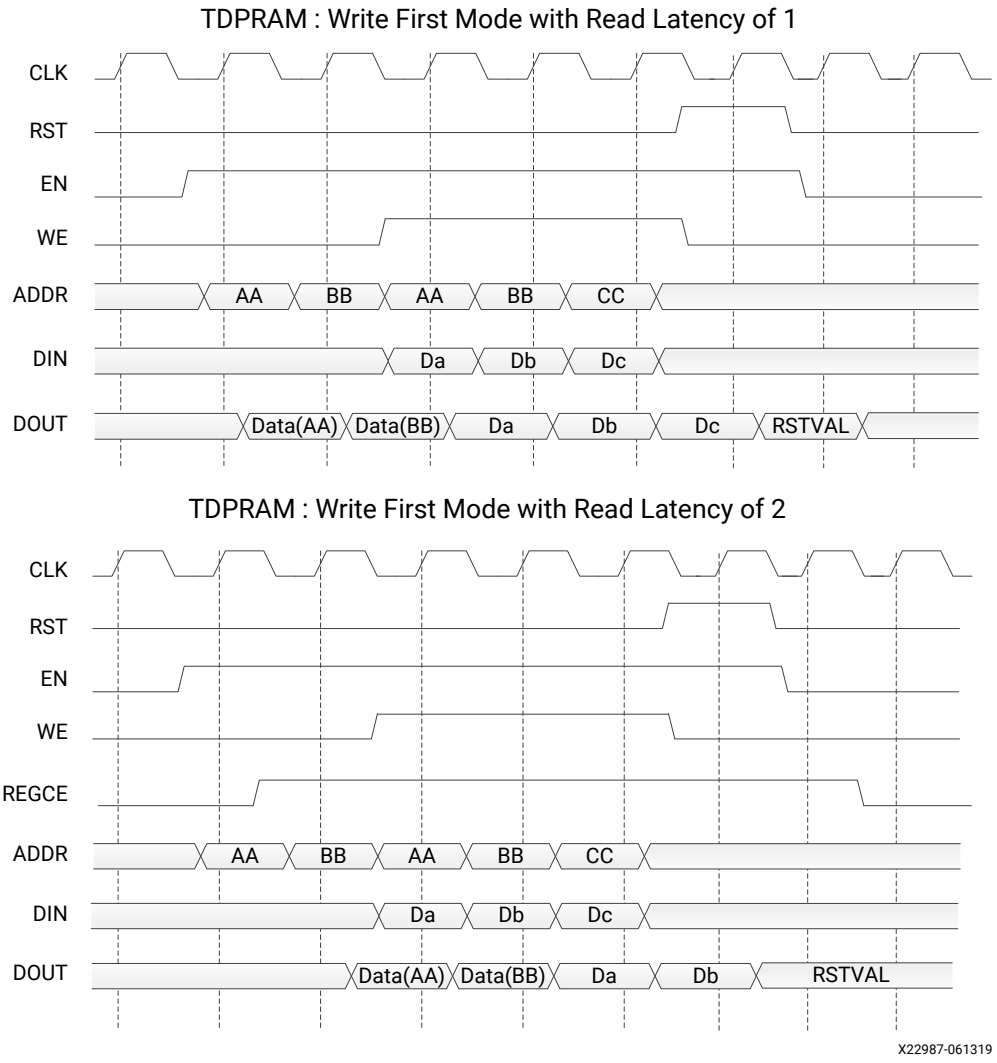
***Note*:**

- When the attribute “CLOCKING_MODE” is set to “common_clock”, all read/write operations to memory through port A and port B are performed on clka. If this attribute is set to “independent_clock”, then read/write operations through port A are performed based on clka, and read/write operations through port B are performed based on `clkb`.
- Writing to an out-of-range address location may overwrite a valid address location when effective address bits match to a physical memory address location.
- set_false_path constraint is needed for the independent clock distributed RAM based memory if the design takes care of avoiding address collision (write address != read address at any given point of time). Set USE_EMBEDDED_CONSTRAINT = 1 if XPM_MEMORY needs to take care of necessary constraints. If USE_EMBEDDED_CONSTRAINT = 0, Vivado may trigger Timing-6 or Timing-7 or both. Alternatively, you can also add the constraint when USE_EMBEDDED_CONSTRAINT = 0. An example of adding this constraint is provided below. If Port-B also has write permissions for an Independent clock configuration, then a similar constraint needs to be added for clkb as well.

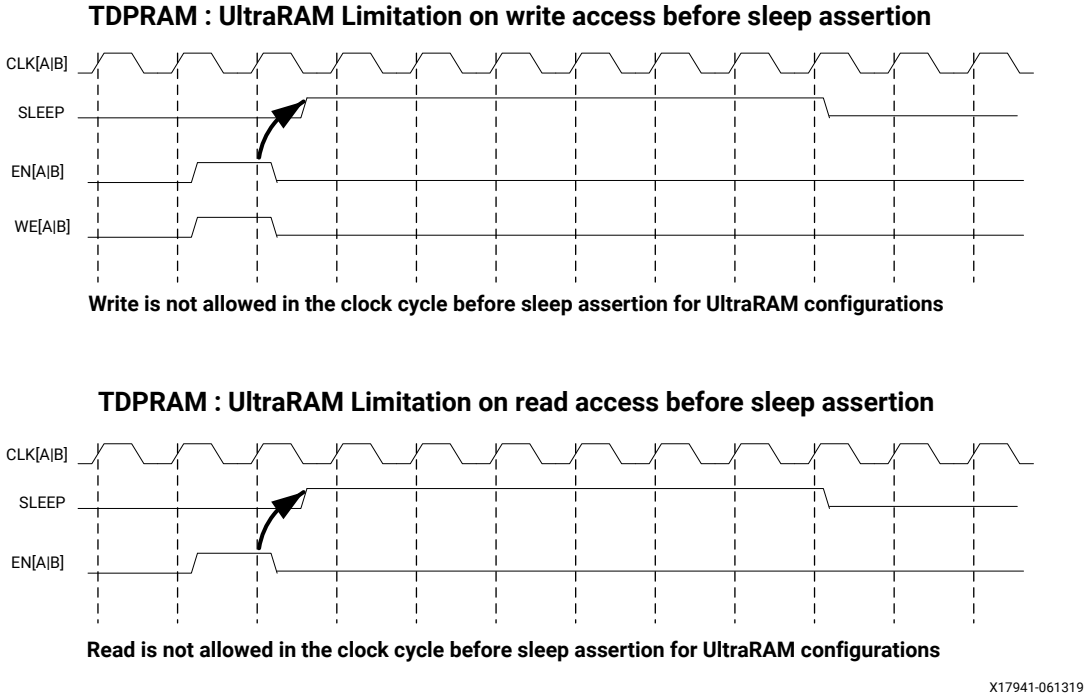
```
set_false_path -from [filter [all_fanout -from [get_ports clka]
-flat -endpoints_only] {IS_LEAF}] -through [get_pins -of_objects
[get_cells -hier * -filter {PRIMITIVE_SUBGROUP==LUTRAM ||
PRIMITIVE_SUBGROUP==dram || PRIMITIVE_SUBGROUP==drom}]
-filter {DIRECTION==OUT}]
```

- If "CLOCKING_MODE" is set to "independent_clock", Vivado may trigger a false positive CDC-1 warning and can be ignored.
- The use of UltraRAM's dedicated input and output registers are controlled by synthesis based on the READ_LATENCY_B value. For example, if 4 UltraRAMs are in cascade and the READ_LATENCY_B is ≥ 4, then synthesis will absorb as much registers inside UltraRAM primitive as possible.
- For UltraRAM's, OREG enabled when READ_LATENCY_B ≥ 3 in NO_CHANGE mode.
- For larger memories (≥2 MB), the recommended read latency must be > 8 because the default cascade height used by Vivado synthesis is 8.

Timing Diagrams



***Note*:** The above waveforms do not distinguish between port A and port B. The behavior shown in the above waveforms is true for both port A and port B.



***Note*:** The UltraRAM primitive does not support Write/Read access in the clock cycle just before assertion of sleep gets recognized on the positive edge of the clock when its OREG attribute is set to TRUE. For UltraRAM configurations, Write/Read access to the memory is not allowed in the clock cycle just before the assertion of sleep.

ECC Modes

Only the UltraRAM primitives support ECC when the memory type is set to True Dual Port RAM. The three ECC modes supported are:

- Both encode and decode
- Encode only
- Decode only

The read and write usage of the three ECC Modes are the same as described in the Introduction section above. See the “Built-in Error Correction” section of the *7 Series FPGAs Memory Resources User Guide* (UG473) for more details on this feature like Error Injection and syndrome bits calculations.

There are restrictions on the attributes WRITE_DATA_WIDTH_[A|B], READ_DATA_WIDTH_[A|B], and MEMORY_SIZE in each of the above ECC modes.

- **Both encode and decode** WRITE_DATA_WIDTH_[A|B] and READ_DATA_WIDTH_[A|B] must be multiples of 64-bits. Violating this rule will results in a DRC in XPM_Memory.

- **Encode only** WRITE_DATA_WIDTH_[A|B] must be a multiple of 64 bits and READ_DATA_WIDTH_[A|B] must be a multiple of 72-bits. MEMORY_SIZE must be a multiple of READ_DATA_WIDTH_[A|B]. Violating these rules will result in a DRC.
- **Decode only** WRITE_DATA_WIDTH_[A|B] must be a multiple of 72 bits and READ_DATA_WIDTH_[A|B] must be a multiple of 64-bits. MEMORY_SIZE must be a multiple of WRITE_DATA_WIDTH_[A|B]. Violating these rules will result in a DRC.

When ECC is enabled the following are not supported:

- Asymmetry
- Initialization
- Reset (neither non-zero reset value nor reset assertion)

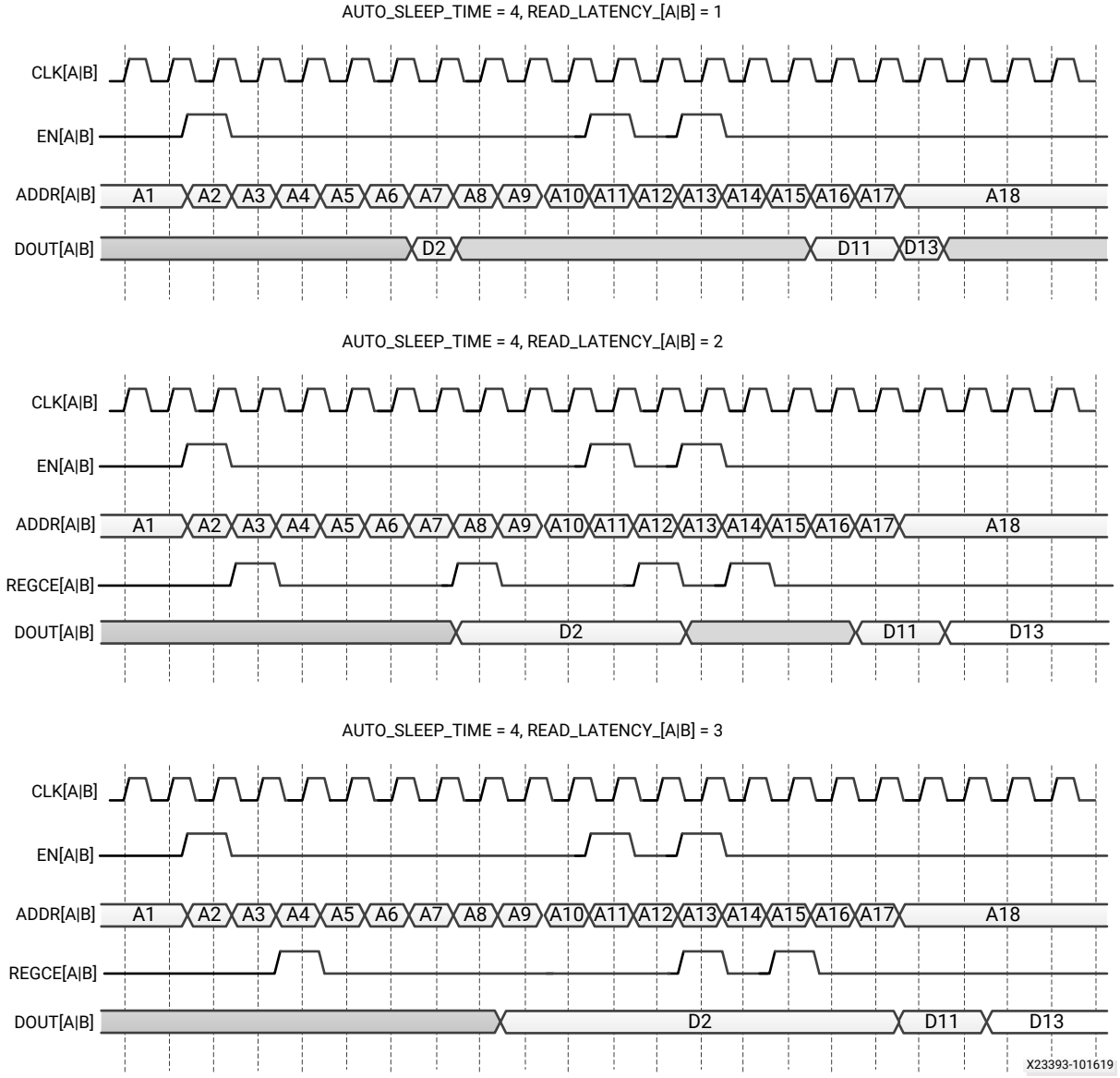
Note: ECC uses a hard-ECC block available in the BRAM/URAM macro and the data width should be multiples of 64/72. Use ECC IP for other data width combinations.

Auto Sleep Mode

- This feature is applicable only when MEMORY_PRIMITIVE is URAM and is controlled internally in the UltraRAM to check if it can be put in sleep mode and when it needs to wake up. Thus power savings are obtained automatically without having to explicitly control the SLEEP Pin.
- When AUTO_SLEEP_TIME is 0, the feature is disabled. When AUTO_SLEEP_TIME is non-zero, XPM_MEMORY constructs the pipeline registers equal to AUTO_SLEEP_TIME value on all input signals except `rst[a|b]`.
- If AUTO_SLEEP_TIME is too low, then UltraRAM goes into sleep and wakeup too often, which can cause more power to be consumed.
- The number of sleep cycles achieved is calculated by following formula:
  - If number of consecutive inactive cycles is < AUTO_SLEEP_TIME, then number of sleep cycles = 0.
  - If number of consecutive inactive cycles is ≥AUTO_SLEEP_TIME, Then number of consecutive sleep cycles = Number of consecutive inactive cycles – 3.
  - Inactive cycle is defined as a cycle where there is no Read/Write operation from either port.
- The latency between the read operation and the data arrival at dout[a|b] is AUTO_SLEEP_TIME + READ_LATENCY_[A|B] clock cycles (Assuming that REGCE is high when the output data pipe line exists).
- When the READ_LATENCY_[A|B] is set to 1 or 2, XPM_Memory behaviorally models the AUTO SLEEP feature and forces ‘x’ on DOUT[A|B] when the RAM is in Auto Sleep Mode. For READ_LATENCY_[A|B] greater than 2, the propagation of ‘x’ cannot happen to the DOUT[A|B] as the output registers gets the clock enable (delayed read enable) after UltraRAM comes out of sleep mode.

- The Auto Sleep mode is most effective for larger memory sizes or any memory with very little activity.

Timing diagrams for Auto Sleep Mode at various read latencies are shown below.



Port Descriptions

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
addra	Input	ADDR_WIDTH_A	clka	NA	Active	Address for port A write and read operations.
addrb	Input	ADDR_WIDTH_B	clkb	NA	Active	Address for port B write and read operations.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
clka	Input	1	NA	EDGE _RISING	Active	Clock signal for port A. Also clocks port B when parameter CLOCKING_MODE is "common_clock".
clkb	Input	1	NA	EDGE _RISING	Active	Clock signal for port B when parameter CLOCKING_MODE is "independent_clock". Unused when parameter CLOCKING_MODE is "common_clock".
dbiterra	Output	1	clka	LEVEL _HIGH	DoNotCare	Status signal to indicate double bit error occurrence on the data output of port A.
dbiterrb	Output	1	clkb	LEVEL _HIGH	DoNotCare	Status signal to indicate double bit error occurrence on the data output of port A.
dina	Input	WRITE _DATA _WIDTH _A	clka	NA	Active	Data input for port A write operations.
dinb	Input	WRITE _DATA _WIDTH _B	clkb	NA	Active	Data input for port B write operations.
douta	Output	READ _DATA _WIDTH _A	clka	NA	Active	Data output for port A read operations.
doutb	Output	READ _DATA _WIDTH _B	clkb	NA	Active	Data output for port B read operations.
ena	Input	1	clka	LEVEL _HIGH	Active	Memory enable signal for port A. Must be high on clock cycles when read or write operations are initiated. Pipelined internally.
enb	Input	1	clkb	LEVEL _HIGH	Active	Memory enable signal for port B. Must be high on clock cycles when read or write operations are initiated. Pipelined internally.
injectdbiterra	Input	1	clka	LEVEL _HIGH	0	Controls double bit error injection on input data when ECC enabled (Error injection capability is not available in "decode_only" mode).
injectdbiterrb	Input	1	clkb	LEVEL _HIGH	0	Controls double bit error injection on input data when ECC enabled (Error injection capability is not available in "decode_only" mode).
injectsbiterra	Input	1	clka	LEVEL _HIGH	0	Controls single bit error injection on input data when ECC enabled (Error injection capability is not available in "decode_only" mode).
injectsbiterrb	Input	1	clkb	LEVEL _HIGH	0	Controls single bit error injection on input data when ECC enabled (Error injection capability is not available in "decode_only" mode).
regcea	Input	1	clka	LEVEL _HIGH	1	Clock Enable for the last register stage on the output data path.
regceb	Input	1	clkb	LEVEL _HIGH	1	Clock Enable for the last register stage on the output data path.

Port	Direction	Width	Domain	Sense	Handling if Unused	Function
rsta	Input	1	clka	LEVEL_HIGH	Active	Reset signal for the final port A output register stage. Synchronously resets output port douta to the value specified by parameter READ_RESET_VALUE_A.
rstb	Input	1	clkb	LEVEL_HIGH	Active	Reset signal for the final port B output register stage. Synchronously resets output port doutb to the value specified by parameter READ_RESET_VALUE_B.
sbiterra	Output	1	clka	LEVEL_HIGH	DoNotCare	Status signal to indicate single bit error occurrence on the data output of port A.
sbiterrb	Output	1	clkb	LEVEL_HIGH	DoNotCare	Status signal to indicate single bit error occurrence on the data output of port B.
sleep	Input	1	NA	LEVEL_HIGH	0	sleep signal to enable the dynamic power saving feature.
wea	Input	WRITE_DATA_WIDTH_A / BYTE_WRITE_WIDTH_A	clka	LEVEL_HIGH	Active	Write enable vector for port A input data port dina. 1 bit wide when word-wide writes are used. In byte-wide write configurations, each bit controls the writing one byte of dina to address addra. For example, to synchronously write only bits [15-8] of dina when WRITE_DATA_WIDTH_A is 32, wea would be 4'b0010.
web	Input	WRITE_DATA_WIDTH_B / BYTE_WRITE_WIDTH_B	clkb	LEVEL_HIGH	Active	Write enable vector for port B input data port dinb. 1 bit wide when word-wide writes are used. In byte-wide write configurations, each bit controls the writing one byte of dinb to address addrb. For example, to synchronously write only bits [15-8] of dinb when WRITE_DATA_WIDTH_B is 32, web would be 4'b0010.

Design Entry Method

Instantiation	Yes
Inference	No
IP and IP Integrator Catalog	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ADDR_WIDTH_A	DECIMAL	1 to 20	6	Specify the width of the port A address port addra, in bits. Must be large enough to access the entire memory from port A, i.e. >= $clog2(MEMORY_SIZE/[WRITE|READ]_DATA_WIDTH_A).
ADDR_WIDTH_B	DECIMAL	1 to 20	6	Specify the width of the port B address port addrb, in bits. Must be large enough to access the entire memory from port B, i.e. >= $clog2(MEMORY_SIZE/[WRITE|READ]_DATA_WIDTH_B).

Attribute	Type	Allowed Values	Default	Description
AUTO_SLEEP_TIME	DECIMAL	0 to 15	0	Number of clk[a|b] cycles to auto-sleep, if feature is available in architecture 0 - Disable auto-sleep feature 3-15 - Number of auto-sleep latency cycles Do not change from the value provided in the template instantiation
BYTE_WRITE_WIDTH _A	DECIMAL	1 to 4608	32	To enable byte-wide writes on port A, specify the byte width, in bits- 8- 8-bit byte-wide writes, legal when WRITE_DATA_WIDTH_A is an integer multiple of 8 9- 9-bit byte-wide writes, legal when WRITE_DATA_WIDTH_A is an integer multiple of 9 Or to enable word-wide writes on port A, specify the same value as for WRITE_DATA_WIDTH_A.
BYTE_WRITE_WIDTH _B	DECIMAL	1 to 4608	32	To enable byte-wide writes on port B, specify the byte width, in bits- 8- 8-bit byte-wide writes, legal when WRITE_DATA_WIDTH_B is an integer multiple of 8 9- 9-bit byte-wide writes, legal when WRITE_DATA_WIDTH_B is an integer multiple of 9 Or to enable word-wide writes on port B, specify the same value as for WRITE_DATA_WIDTH_B.
CASCADE_HEIGHT	DECIMAL	0 to 64	0	0- No Cascade Height, Allow Vivado Synthesis to choose. 1 or more - Vivado Synthesis sets the specified value as Cascade Height.
CLOCKING_MODE	STRING	"common _clock", "independent _clock"	"common _clock"	Designate whether port A and port B are clocked with a common clock or with independent clocks- "common_clock"- Common clocking; clock both port A and port B with clka "independent_clock"- Independent clocking; clock port A with clka and port B with clkb
ECC_MODE	STRING	"no_ecc", "both _encode _and _decode", "decode _only", "encode _only"	"no_ecc"	• "no_ecc" - Disables ECC • "encode_only" - Enables ECC Encoder only • "decode_only" - Enables ECC Decoder only • "both_encode_and_decode" - Enables both ECC Encoder and Decoder
MEMORY_INIT_FILE	STRING	String	"none"	Specify "none" (including quotes) for no memory initialization, or specify the name of a memory initialization file- Enter only the name of the file with .mem extension, including quotes but without path (e.g. "my_file.mem"). File format must be ASCII and consist of only hexadecimal values organized into the specified depth by narrowest data width generic value of the memory. See the Memory File (MEM) section for more information on the syntax. Initialization of memory happens through the file name specified only when parameter MEMORY_INIT_PARAM value is equal to "". | When using XPM_MEMORY in a project, add the specified file to the Vivado project as a design source.

Attribute	Type	Allowed Values	Default	Description
MEMORY_INIT _PARAM	STRING	String	"0"	Specify "" or "0" (including quotes) for no memory initialization through parameter, or specify the string containing the hex characters. Enter only hex characters with each location separated by delimiter (,). Parameter format must be ASCII and consist of only hexadecimal values organized into the specified depth by narrowest data width generic value of the memory. For example, if the narrowest data width is 8, and the depth of memory is 8 locations, then the parameter value should be passed as shown below. parameter MEMORY_INIT_PARAM = "AB,CD,EF,1,2,34,56,78" Where "AB" is the 0th location and "78" is the 7th location.
MEMORY _OPTIMIZATION	STRING	"true", "false"	"true"	Specify "true" to enable the optimization of unused memory or bits in the memory structure. Specify "false" to disable the optimization of unused memory or bits in the memory structure
MEMORY _PRIMITIVE	STRING	"auto", "block", "distributed", "ultra"	"auto"	Designate the memory primitive (resource type) to use- "auto"- Allow Vivado Synthesis to choose "distributed"- Distributed memory "block"- Block memory "ultra"- Ultra RAM memory NOTE: There may be a behavior mismatch if Block RAM or Ultra RAM specific features, like ECC or Asymmetry, are selected with MEMORY_PRIMITIVE set to "auto".
MEMORY_SIZE	DECIMAL	2 to 150994944	2048	Specify the total memory array size, in bits. For example, enter 65536 for a 2kx32 RAM. When ECC is enabled and set to "encode_only", then the memory size has to be multiples of READ_DATA_WIDTH_[A|B] When ECC is enabled and set to "decode_only", then the memory size has to be multiples of WRITE_DATA_WIDTH_[A|B]
MESSAGE_CONTROL	DECIMAL	0 to 1	0	Specify 1 to enable the dynamic message reporting such as collision warnings, and 0 to disable the message reporting
READ_DATA_WIDTH_A	DECIMAL	1 to 4608	32	Specify the width of the port A read data output port douta, in bits. The values of READ_DATA_WIDTH_A and WRITE_DATA_WIDTH_A must be equal. When ECC is enabled and set to "encode_only", then READ_DATA_WIDTH_A has to be multiples of 72-bits When ECC is enabled and set to "decode_only" or "both_encode_and_decode", then READ_DATA_WIDTH_A has to be multiples of 64-bits
READ_DATA_WIDTH_B	DECIMAL	1 to 4608	32	Specify the width of the port B read data output port doutb, in bits. The values of READ_DATA_WIDTH_B and WRITE_DATA_WIDTH_B must be equal. When ECC is enabled and set to "encode_only", then READ_DATA_WIDTH_B has to be multiples of 72-bits When ECC is enabled and set to "decode_only" or "both_encode_and_decode", then READ_DATA_WIDTH_B has to be multiples of 64-bits

Attribute	Type	Allowed Values	Default	Description
READ_LATENCY_A	DECIMAL	0 to 100	2	Specify the number of register stages in the port A read data pipeline. Read data output to port douta takes this number of clka cycles. To target block memory, a value of 1 or larger is required- 1 causes use of memory latch only; 2 causes use of output register. To target distributed memory, a value of 0 or larger is required- 0 indicates combinatorial output. Values larger than 2 synthesize additional flip-flops that are not retimed into memory primitives.
READ_LATENCY_B	DECIMAL	0 to 100	2	Specify the number of register stages in the port B read data pipeline. Read data output to port doutb takes this number of clkb cycles (clka when CLOCKING_MODE is "common_clock"). To target block memory, a value of 1 or larger is required- 1 causes use of memory latch only; 2 causes use of output register. To target distributed memory, a value of 0 or larger is required- 0 indicates combinatorial output. Values larger than 2 synthesize additional flip-flops that are not retimed into memory primitives.
READ_RESET_VALUE _A	STRING	String	"0"	Specify the reset value of the port A final output register stage in response to rsta input port is assertion. As this parameter is a string, please specify the hex values inside double quotes. As an example, If the read data width is 8, then specify READ_RESET_VALUE_A = "EA"; When ECC is enabled, then reset value is not supported
READ_RESET_VALUE _B	STRING	String	"0"	Specify the reset value of the port B final output register stage in response to rstb input port is assertion. As this parameter is a string, please specify the hex values inside double quotes. As an example, If the read data width is 8, then specify READ_RESET_VALUE_B = "EA"; When ECC is enabled, then reset value is not supported
RST_MODE_A	STRING	"SYNC", "ASYNC"	"SYNC"	Describes the behaviour of the reset • "SYNC" - when reset is applied, synchronously resets output port douta to the value specified by parameter READ_RESET_VALUE_A • "ASYNC" - when reset is applied, asynchronously resets output port douta to zero
RST_MODE_B	STRING	"SYNC", "ASYNC"	"SYNC"	Describes the behaviour of the reset • "SYNC" - when reset is applied, synchronously resets output port doutb to the value specified by parameter READ_RESET_VALUE_B • "ASYNC" - when reset is applied, asynchronously resets output port doutb to zero
SIM_ASSERT_CHK	DECIMAL	0 to 1	0	0- Disable simulation message reporting. Messages related to potential misuse will not be reported. 1- Enable simulation message reporting. Messages related to potential misuse will be reported.

Attribute	Type	Allowed Values	Default	Description
USE_EMBEDDED _CONSTRAINT	DECIMAL	0 to 1	0	Specify 1 to enable the set_false_path constraint addition between clka of Distributed RAM and doutb_reg on clkb
USE_MEM_INIT	DECIMAL	0 to 1	1	Specify 1 to enable the generation of below message and 0 to disable generation of the following message completely. "INFO - MEMORY_INIT_FILE and MEMORY_INIT_PARAM together specifies no memory initialization. Initial memory contents will be all 0s." NOTE: This message gets generated only when there is no Memory Initialization specified either through file or Parameter.
WAKEUP_TIME	STRING	"disable _sleep", "use _sleep _pin"	"disable _sleep"	Specify "disable_sleep" to disable dynamic power saving option, and specify "use_sleep_pin" to enable the dynamic power saving option
WRITE_DATA_WIDTH_A	DECIMAL	1 to 4608	32	Specify the width of the port A write data input port dina, in bits. The values of WRITE_DATA_WIDTH_A and READ_DATA_WIDTH_A must be equal. When ECC is enabled and set to "encode_only" or "both_encode_and_decode", then WRITE_DATA_WIDTH_A has to be multiples of 64-bits When ECC is enabled and set to "decode_only", then WRITE_DATA_WIDTH_A has to be multiples of 72-bits
WRITE_DATA_WIDTH _B	DECIMAL	1 to 4608	32	Specify the width of the port B write data input port dinb, in bits. The values of WRITE_DATA_WIDTH_B and READ_DATA_WIDTH_B must be equal. When ECC is enabled and set to "encode_only" or "both_encode_and_decode", then WRITE_DATA_WIDTH_B has to be multiples of 64-bits When ECC is enabled and set to "decode_only", then WRITE_DATA_WIDTH_B has to be multiples of 72-bits
WRITE_MODE_A	STRING	"no _change", "read _first", "write _first"	"no _change"	Write mode behavior for port A output data port, douta.
WRITE_MODE_B	STRING	"no _change", "read _first", "write _first"	"no _change"	Write mode behavior for port B output data port, doutb.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library xpm;
use xpm.vcomponents.all;
```

```
-- xpm_memory_tdpram: True Dual Port RAM
-- Xilinx Parameterized Macro, version 2020.1

xpm_memory_tdpram_inst : xpm_memory_tdpram
generic map (
   ADDR_WIDTH_A => 6,               -- DECIMAL
   ADDR_WIDTH_B => 6,               -- DECIMAL
   AUTO_SLEEP_TIME => 0,            -- DECIMAL
   BYTE_WRITE_WIDTH_A => 32,        -- DECIMAL
   BYTE_WRITE_WIDTH_B => 32,        -- DECIMAL
```

```
   CASCADE_HEIGHT => 0,              -- DECIMAL
   CLOCKING_MODE => "common_clock", -- String
   ECC_MODE => "no_ecc",             -- String
   MEMORY_INIT_FILE => "none",       -- String
   MEMORY_INIT_PARAM => "0",         -- String
   MEMORY_OPTIMIZATION => "true",    -- String
   MEMORY_PRIMITIVE => "auto",       -- String
   MEMORY_SIZE => 2048,              -- DECIMAL
   MESSAGE_CONTROL => 0,             -- DECIMAL
   READ_DATA_WIDTH_A => 32,          -- DECIMAL
   READ_DATA_WIDTH_B => 32,          -- DECIMAL
   READ_LATENCY_A => 2,              -- DECIMAL
   READ_LATENCY_B => 2,              -- DECIMAL
   READ_RESET_VALUE_A => "0",        -- String
   READ_RESET_VALUE_B => "0",        -- String
   RST_MODE_A => "SYNC",             -- String
   RST_MODE_B => "SYNC",             -- String
   SIM_ASSERT_CHK => 0,            -- DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   USE_EMBEDDED_CONSTRAINT => 0,     -- DECIMAL
   USE_MEM_INIT => 1,                -- DECIMAL
   WAKEUP_TIME => "disable_sleep",   -- String
   WRITE_DATA_WIDTH_A => 32,         -- DECIMAL
   WRITE_DATA_WIDTH_B => 32,         -- DECIMAL
   WRITE_MODE_A => "no_change",      -- String
   WRITE_MODE_B => "no_change"       -- String
)
port map (
   dbiterra => dbiterra,             -- 1-bit output: Status signal to indicate double bit error occurrence
                                        -- on the data output of port A.

   dbiterrb => dbiterrb,             -- 1-bit output: Status signal to indicate double bit error occurrence
                                        -- on the data output of port A.

   douta => douta,                   -- READ_DATA_WIDTH_A-bit output: Data output for port A read operations.
   doutb => doutb,                   -- READ_DATA_WIDTH_B-bit output: Data output for port B read operations.
   sbiterra => sbiterra,             -- 1-bit output: Status signal to indicate single bit error occurrence
                                        -- on the data output of port A.

   sbiterrb => sbiterrb,             -- 1-bit output: Status signal to indicate single bit error occurrence
                                        -- on the data output of port B.

   addra => addra,                   -- ADDR_WIDTH_A-bit input: Address for port A write and read operations.
   addrb => addrb,                   -- ADDR_WIDTH_B-bit input: Address for port B write and read operations.
   clka => clka,                     -- 1-bit input: Clock signal for port A. Also clocks port B when
                                        -- parameter CLOCKING_MODE is "common_clock".

   clkb => clkb,                     -- 1-bit input: Clock signal for port B when parameter CLOCKING_MODE is
                                        -- "independent_clock". Unused when parameter CLOCKING_MODE is
                                        -- "common_clock".

   dina => dina,                     -- WRITE_DATA_WIDTH_A-bit input: Data input for port A write operations.
   dinb => dinb,                     -- WRITE_DATA_WIDTH_B-bit input: Data input for port B write operations.
   ena => ena,                       -- 1-bit input: Memory enable signal for port A. Must be high on clock
                                     -- cycles when read or write operations are initiated. Pipelined
                                        -- internally.

   enb => enb,                       -- 1-bit input: Memory enable signal for port B. Must be high on clock
                                     -- cycles when read or write operations are initiated. Pipelined
                                        -- internally.

   injectdbiterra => injectdbiterra, -- 1-bit input: Controls double bit error injection on input data when
                                        -- ECC enabled (Error injection capability is not available in
                                        -- "decode_only" mode).

   injectdbiterrb => injectdbiterrb, -- 1-bit input: Controls double bit error injection on input data when
                                        -- ECC enabled (Error injection capability is not available in
                                        -- "decode_only" mode).

   injectsbiterra => injectsbiterra, -- 1-bit input: Controls single bit error injection on input data when
                                        -- ECC enabled (Error injection capability is not available in
                                        -- "decode_only" mode).

   injectsbiterrb => injectsbiterrb, -- 1-bit input: Controls single bit error injection on input data when
                                        -- ECC enabled (Error injection capability is not available in
                                        -- "decode_only" mode).

   regcea => regcea,                 -- 1-bit input: Clock Enable for the last register stage on the output
                                        -- data path.
```

```
   regceb => regceb,                  -- 1-bit input: Clock Enable for the last register stage on the output
                                         -- data path.

   rsta => rsta,                      -- 1-bit input: Reset signal for the final port A output register
                                      -- stage. Synchronously resets output port douta to the value specified
                                         -- by parameter READ_RESET_VALUE_A.

   rstb => rstb,                      -- 1-bit input: Reset signal for the final port B output register
                                      -- stage. Synchronously resets output port doutb to the value specified
                                         -- by parameter READ_RESET_VALUE_B.

   sleep => sleep,                    -- 1-bit input: sleep signal to enable the dynamic power saving feature.
   wea => wea,                        -- WRITE_DATA_WIDTH_A/BYTE_WRITE_WIDTH_A-bit input: Write enable vector
                                      -- for port A input data port dina. 1 bit wide when word-wide writes
                                      -- are used. In byte-wide write configurations, each bit controls the
                                         -- writing one byte of dina to address addra. For example, to
                                      -- synchronously write only bits [15-8] of dina when WRITE_DATA_WIDTH_A
                                         -- is 32, wea would be 4'b0010.

   web => web                         -- WRITE_DATA_WIDTH_B/BYTE_WRITE_WIDTH_B-bit input: Write enable vector
                                      -- for port B input data port dinb. 1 bit wide when word-wide writes
                                      -- are used. In byte-wide write configurations, each bit controls the
                                         -- writing one byte of dinb to address addrb. For example, to
                                      -- synchronously write only bits [15-8] of dinb when WRITE_DATA_WIDTH_B
                                         -- is 32, web would be 4'b0010.

);

-- End of xpm_memory_tdpram_inst instantiation
```

Verilog Instantiation Template

```
// xpm_memory_tdpram: True Dual Port RAM
// Xilinx Parameterized Macro, version 2020.1

xpm_memory_tdpram #(
   .ADDR_WIDTH_A(6),               // DECIMAL
   .ADDR_WIDTH_B(6),               // DECIMAL
   .AUTO_SLEEP_TIME(0),            // DECIMAL
   .BYTE_WRITE_WIDTH_A(32),        // DECIMAL
   .BYTE_WRITE_WIDTH_B(32),        // DECIMAL
   .CASCADE_HEIGHT(0),             // DECIMAL
   .CLOCKING_MODE("common_clock"), // String
   .ECC_MODE("no_ecc"),            // String
   .MEMORY_INIT_FILE("none"),      // String
   .MEMORY_INIT_PARAM("0"),        // String
   .MEMORY_OPTIMIZATION("true"),   // String
   .MEMORY_PRIMITIVE("auto"),      // String
   .MEMORY_SIZE(2048),             // DECIMAL
   .MESSAGE_CONTROL(0),            // DECIMAL
   .READ_DATA_WIDTH_A(32),         // DECIMAL
   .READ_DATA_WIDTH_B(32),         // DECIMAL
   .READ_LATENCY_A(2),             // DECIMAL
   .READ_LATENCY_B(2),             // DECIMAL
   .READ_RESET_VALUE_A("0"),       // String
   .READ_RESET_VALUE_B("0"),       // String
   .RST_MODE_A("SYNC"),            // String
   .RST_MODE_B("SYNC"),            // String
   .SIM_ASSERT_CHK(0),           // DECIMAL; 0=disable simulation messages, 1=enable simulation messages
   .USE_EMBEDDED_CONSTRAINT(0),    // DECIMAL
   .USE_MEM_INIT(1),               // DECIMAL
   .WAKEUP_TIME("disable_sleep"),  // String
   .WRITE_DATA_WIDTH_A(32),        // DECIMAL
   .WRITE_DATA_WIDTH_B(32),        // DECIMAL
   .WRITE_MODE_A("no_change"),     // String
   .WRITE_MODE_B("no_change")      // String
)
xpm_memory_tdpram_inst (
   .dbiterra(dbiterra),            // 1-bit output: Status signal to indicate double bit error occurrence
                                      // on the data output of port A.

   .dbiterrb(dbiterrb),            // 1-bit output: Status signal to indicate double bit error occurrence
                                      // on the data output of port A.

   .douta(douta),                  // READ_DATA_WIDTH_A-bit output: Data output for port A read operations.
   .doutb(doutb),                  // READ_DATA_WIDTH_B-bit output: Data output for port B read operations.
```

```
   .sbiterra(sbiterra),             // 1-bit output: Status signal to indicate single bit error occurrence
                                    // on the data output of port A.

   .sbiterrb(sbiterrb),             // 1-bit output: Status signal to indicate single bit error occurrence
                                    // on the data output of port B.

   .addra(addra),                   // ADDR_WIDTH_A-bit input: Address for port A write and read operations.
   .addrb(addrb),                   // ADDR_WIDTH_B-bit input: Address for port B write and read operations.
   .clka(clka),                     // 1-bit input: Clock signal for port A. Also clocks port B when
                                    // parameter CLOCKING_MODE is "common_clock".

   .clkb(clkb),                     // 1-bit input: Clock signal for port B when parameter CLOCKING_MODE is
                                    // "independent_clock". Unused when parameter CLOCKING_MODE is
                                    // "common_clock".

   .dina(dina),                     // WRITE_DATA_WIDTH_A-bit input: Data input for port A write operations.
   .dinb(dinb),                     // WRITE_DATA_WIDTH_B-bit input: Data input for port B write operations.
   .ena(ena),                       // 1-bit input: Memory enable signal for port A. Must be high on clock
                                    // cycles when read or write operations are initiated. Pipelined
                                    // internally.

   .enb(enb),                       // 1-bit input: Memory enable signal for port B. Must be high on clock
                                    // cycles when read or write operations are initiated. Pipelined
                                    // internally.

   .injectdbiterra(injectdbiterra), // 1-bit input: Controls double bit error injection on input data when
                                    // ECC enabled (Error injection capability is not available in
                                    // "decode_only" mode).

   .injectdbiterrb(injectdbiterrb), // 1-bit input: Controls double bit error injection on input data when
                                    // ECC enabled (Error injection capability is not available in
                                    // "decode_only" mode).

   .injectsbiterra(injectsbiterra), // 1-bit input: Controls single bit error injection on input data when
                                    // ECC enabled (Error injection capability is not available in
                                    // "decode_only" mode).

   .injectsbiterrb(injectsbiterrb), // 1-bit input: Controls single bit error injection on input data when
                                    // ECC enabled (Error injection capability is not available in
                                    // "decode_only" mode).

   .regcea(regcea),                 // 1-bit input: Clock Enable for the last register stage on the output
                                    // data path.

   .regceb(regceb),                 // 1-bit input: Clock Enable for the last register stage on the output
                                    // data path.

   .rsta(rsta),                     // 1-bit input: Reset signal for the final port A output register stage.
                                    // Synchronously resets output port douta to the value specified by
                                    // parameter READ_RESET_VALUE_A.

   .rstb(rstb),                     // 1-bit input: Reset signal for the final port B output register stage.
                                    // Synchronously resets output port doutb to the value specified by
                                    // parameter READ_RESET_VALUE_B.

   .sleep(sleep),                   // 1-bit input: sleep signal to enable the dynamic power saving feature.
   .wea(wea),                       // WRITE_DATA_WIDTH_A/BYTE_WRITE_WIDTH_A-bit input: Write enable vector
                                    // for port A input data port dina. 1 bit wide when word-wide writes are
                                    // used. In byte-wide write configurations, each bit controls the
                                    // writing one byte of dina to address addra. For example, to
                                    // synchronously write only bits [15-8] of dina when WRITE_DATA_WIDTH_A
                                    // is 32, wea would be 4'b0010.

   .web(web)                        // WRITE_DATA_WIDTH_B/BYTE_WRITE_WIDTH_B-bit input: Write enable vector
                                    // for port B input data port dinb. 1 bit wide when word-wide writes are
                                    // used. In byte-wide write configurations, each bit controls the
                                    // writing one byte of dinb to address addrb. For example, to
                                    // synchronously write only bits [15-8] of dinb when WRITE_DATA_WIDTH_B
                                    // is 32, web would be 4'b0010.

);

// End of xpm_memory_tdpram_inst instantiation
```

Unimacros

About Unimacros

This section describes the unimacros that can be used with 7 series FPGAs and Zynq®-7000 SoC devices devices. The unimacros are organized alphabetically.

The following information is provided for each unimacro, where applicable:

- Name and description
- Schematic symbol
- Logic table (if any)
- Introduction
- Port descriptions
- Design Entry Method
- Available attributes
- Example instantiation templates
- Links to additional information

Instantiation Templates

Instantiation templates for Unimacros are also available in Vivado, as well as in a downloadable ZIP file. Because PDF includes headers and footers if you copy text that spans pages, you should copy templates from Vivado or the downloaded ZIP file whenever possible.

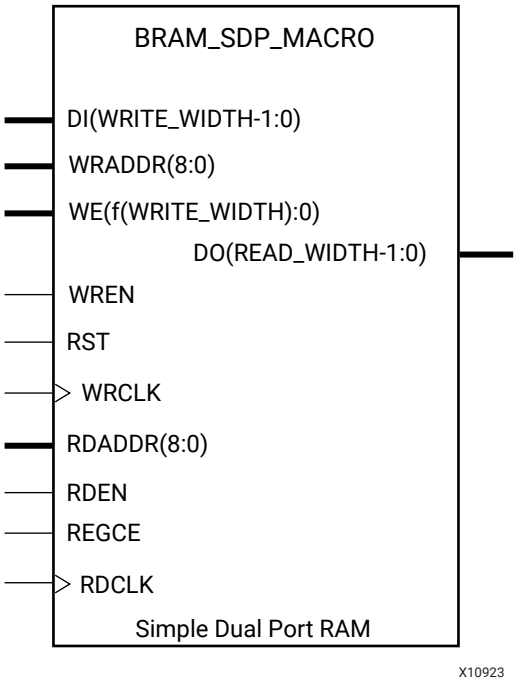
Instantiation templates can be found on the Web in the Instantiation Templates for 7 Series Devices file.

List of UniMacros

Design Element	Description
BRAM_SDP_MACRO	Macro: Simple Dual Port RAM
BRAM_SINGLE_MACRO	Macro: Single Port RAM
BRAM_TDP_MACRO	Macro: True Dual Port RAM
ADDMACC_MACRO	Macro: Adder/Multiplier/Accumulator
ADDSUB_MACRO	Macro: Adder/Subtractor
COUNTER_LOAD_MACRO	Macro: Loadable Counter
COUNTER_TC_MACRO	Macro: Counter with Terminal Count
EQ_COMPARE_MACRO	Macro: Equality Comparator
MACC_MACRO	Macro: Multiplier/Accumulator
MULT_MACRO	Macro: Multiplier
FIFO_DUALCLOCK_MACRO	Macro: Dual Clock First-In, First-Out (FIFO) RAM Buffer
FIFO_SYNC_MACRO	Macro: Synchronous First-In, First-Out (FIFO) RAM Buffer

BRAM_SDP_MACRO

Macro: Simple Dual Port RAM



Introduction

7 series FPGA devices contain several block RAM memories that can be configured as general-purpose 36 Kb or 18 Kb RAM/ROM memories. These block RAM memories offer fast and flexible storage of large amounts of on-chip data. Both read and write operations are fully synchronous to the supplied clock(s) of the component. However, READ and WRITE ports can operate fully independently and asynchronously to each other, accessing the same memory array. Byte-enable write operations are possible, and an optional output register can be used to reduce the clock-to-out times of the RAM.

***Note*:** This element must be configured so that read and write ports have the same width.

Port Descriptions

Port	Direction	Width	Function
DO	Output	See Port Configuration table	Data output bus addressed by RDADDR.
DI	Input	See Port Configuration table	Data input bus addressed by WRADDR.

Port	Direction	Width	Function
WRADDR, RDADDR	Input	See Port Configuration table	Write/Read address input buses.
WE	Input	See Port Configuration table	Byte-Wide Write enable.
WREN, RDEN	Input	1	Write/Read enable
RST	Input	1	Input reset.
REGCE	Input	1	Output register clock enable input (valid only when DO_REG=1).
WRCLK, RDCLK	Input	1	Write/Read clock input.

Port Configuration

This unimacro is a parameterizable version of the primitive, and can be instantiated only. Use this table to correctly configure the unimacro to meet design needs.

DATA_WIDTH	BRAM_SIZE	ADDR	WE
72 - 37	36 Kb	9	8
36 - 19	36 Kb	10	4
	18 Kb	9	
18 - 10	36 Kb	11	2
	18 Kb	10	
9 - 5	36 Kb	12	1
	18 Kb	11	
4 - 3	36 Kb	13	1
	18 Kb	12	
2	36 Kb	14	1
	18 Kb	13	
1	36 Kb	15	1
	18 Kb	14	

Design Entry Method

This unimacro is a parameterizable version of the primitive, and can be instantiated only. Consult the Port Configuration section to correctly configure this element to meet your design needs.

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	Recommended

Available Attributes

Attribute	Type	Allowed Values	Default	Description
BRAM_SIZE	STRING	"36Kb", "18Kb"	"18Kb"	Configures RAM as "36Kb" or "18Kb" memory.
DEVICE	STRING	"7SERIES"	"7SERIES"	Target hardware architecture.
DO_REG	INTEGER	0, 1	0	A value of 1 enables to the output registers to the RAM enabling quicker clock-to-out from the RAM at the expense of an added clock cycle of read latency. A value of 0 allows a read in one clock cycle but will have slower clock to out timing.
INIT	HEX	Any 72-Bit Value	All zeros	Specifies the initial value on the output after configuration.
READ_WIDTH, WRITE_WIDTH	INTEGER	1-72	36	Specifies the size of the DI and DO buses. The following combinations are allowed: • READ_WIDTH = WRITE_WIDTH • If asymmetric, READ_WIDTH and WRITE_WIDTH must be in the ratio of 2, or must be values allowed by the unisim (1, 2, 4, 8, 9, 16, 18, 32, 36, 64, 72)
INIT_FILE	STRING	String representing file name and location.	"NONE"	Name of the file containing initial values.
SIM_COLLISION_CHECK	STRING	"ALL", "WARNING_ONLY", "GENERATE_X_ONLY","NONE"	"ALL"	Allows modification of the simulation behavior if a memory collision occurs. The output is affected as follows: • "ALL": Warning produced and affected outputs/memory location go unknown (X). • "WARNING_ONLY": Warning produced and affected outputs/ memory retain last value. • "GENERATE_X_ONLY": No warning. However, affected outputs/memory go unknown (X). • "NONE" : No warning and affected outputs/memory retain last value. ***Note*:** Setting this to a value other than "ALL" can allow problems in the design go unnoticed during simulation. Care should be taken when changing the value of this attribute. Please see the *Synthesis and Simulation Design Guide* for more information.

Attribute	Type	Allowed Values	Default	Description
SRVAL	HEX	Any 72-Bit Value	All zeros	Specifies the output value of on the DO port upon the assertion of the synchronous reset (RST) signal.
INIT_00 to INIT_7F	HEX	Any 256-Bit Value	All zeros	Specifies the initial contents of the 16 Kb or 32 Kb data memory array.
INITP_00 to INITP_0F	HEX	Any 256-Bit Value	All zeros	Specifies the initial contents of the 2 Kb or 4 Kb parity data memory array.

VHDL Instantiation Template

Unless they already exist, copy the following four statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
library UNIMACRO;
use unimacro.Vcomponents.all;
```

```
-- BRAM_SDP_MACRO: Simple Dual Port RAM
--                 7 Series
-- Xilinx HDL Language Template, version 2020.1

-- Note -  This Unimacro model assumes the port directions to be "downto".
--         Simulation of this model with "to" in the port directions could lead to erroneous results.

-----------------------------------------------------------------------
--  READ_WIDTH | BRAM_SIZE | READ Depth  | RDADDR Width |            --
-- WRITE_WIDTH |           | WRITE Depth | WRADDR Width |  WE Width  --
-- ============|===========|=============|==============|============--
--    37-72    |  "36Kb"   |      512    |     9-bit    |    8-bit   --
--    19-36    |  "36Kb"   |     1024    |    10-bit    |    4-bit   --
--    19-36    |  "18Kb"   |      512    |     9-bit    |    4-bit   --
--    10-18    |  "36Kb"   |     2048    |    11-bit    |    2-bit   --
--    10-18    |  "18Kb"   |     1024    |    10-bit    |    2-bit   --
--     5-9     |  "36Kb"   |     4096    |    12-bit    |    1-bit   --
--     5-9     |  "18Kb"   |     2048    |    11-bit    |    1-bit   --
--     3-4     |  "36Kb"   |     8192    |    13-bit    |    1-bit   --
--     3-4     |  "18Kb"   |     4096    |    12-bit    |    1-bit   --
--       2     |  "36Kb"   |    16384    |    14-bit    |    1-bit   --
--       2     |  "18Kb"   |     8192    |    13-bit    |    1-bit   --
--       1     |  "36Kb"   |    32768    |    15-bit    |    1-bit   --
--       1     |  "18Kb"   |    16384    |    14-bit    |    1-bit   --
-----------------------------------------------------------------------


BRAM_SDP_MACRO_inst : BRAM_SDP_MACRO
generic map (
   BRAM_SIZE => "18Kb", -- Target BRAM, "18Kb" or "36Kb"
   DEVICE => "7SERIES", -- Target device: "VIRTEX5", "VIRTEX6", "7SERIES", "SPARTAN6"
   WRITE_WIDTH => 0,     -- Valid values are 1-72 (37-72 only valid when BRAM_SIZE="36Kb")
   READ_WIDTH => 0,      -- Valid values are 1-72 (37-72 only valid when BRAM_SIZE="36Kb")
   DO_REG => 0, -- Optional output register (0 or 1)
   INIT_FILE => "NONE",
   SIM_COLLISION_CHECK => "ALL", -- Collision check enable "ALL", "WARNING_ONLY",
                                 -- "GENERATE_X_ONLY" or "NONE"
   SRVAL => X"0000000000000000", --  Set/Reset value for port output
   WRITE_MODE => "WRITE_FIRST", -- Specify "READ_FIRST" for same clock or synchronous clocks
                                --  Specify "WRITE_FIRST for asynchrononous clocks on ports
   INIT => X"0000000000000000", --  Initial values on output port
   -- The following INIT_xx declarations specify the initial contents of the RAM
   INIT_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_07 => X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
    INIT_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_0F => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_10 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_11 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_12 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_13 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_14 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_15 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_16 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_17 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_18 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_19 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1F => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_20 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_21 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_22 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_23 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_24 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_25 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_26 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_27 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_28 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_29 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2F => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_30 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_31 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_32 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_33 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_34 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_35 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_36 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_37 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_38 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_39 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3F => X"0000000000000000000000000000000000000000000000000000000000000000",

    -- The next set of INIT_xx are valid when configured as 36Kb
    INIT_40 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_41 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_42 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_43 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_44 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_45 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_46 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_47 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_48 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_49 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4F => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_50 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_51 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_52 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_53 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_54 => X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
   INIT_55 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_56 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_57 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_58 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_59 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_60 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_61 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_62 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_63 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_64 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_65 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_66 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_67 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_68 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_69 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_70 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_71 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_72 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_73 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_74 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_75 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_76 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_77 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_78 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_79 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7F => X"0000000000000000000000000000000000000000000000000000000000000000",

   -- The next set of INITP_xx are for the parity bits
   INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000",

   -- The next set of INIT_xx are valid when configured as 36Kb
   INITP_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
port map (
   DO => DO,         -- Output read data port, width defined by READ_WIDTH parameter
   DI => DI,         -- Input write data port, width defined by WRITE_WIDTH parameter
   RDADDR => RDADDR, -- Input read address, width defined by read port depth
   RDCLK => RDCLK,   -- 1-bit input read clock
   RDEN => RDEN,     -- 1-bit input read port enable
   REGCE => REGCE,   -- 1-bit input read output register enable
   RST => RST,       -- 1-bit input reset
   WE => WE,         -- Input write enable, width defined by write port depth
   WRADDR => WRADDR, -- Input write address, width defined by write port depth
   WRCLK => WRCLK,   -- 1-bit input write clock
   WREN => WREN      -- 1-bit input write port enable
);
-- End of BRAM_SDP_MACRO_inst instantiation
```

Verilog Instantiation Template

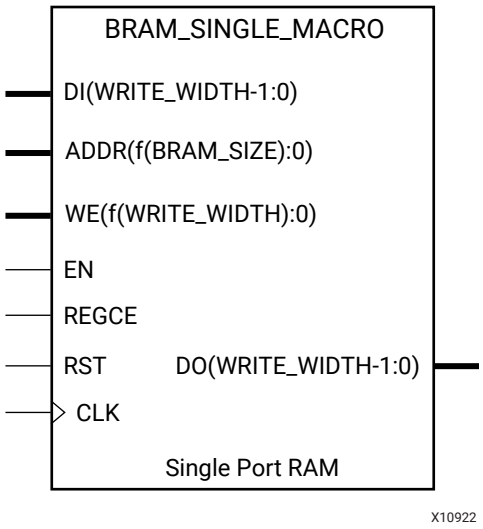
```
// BRAM_SDP_MACRO: Simple Dual Port RAM
//                 7 Series
// Xilinx HDL Language Template, version 2020.1

//////////////////////////////////////////////////////////////////////
//  READ_WIDTH | BRAM_SIZE | READ Depth  | RDADDR Width |            //
// WRITE_WIDTH |           | WRITE Depth | WRADDR Width |  WE Width  //
// ============|===========|=============|==============|============//
//    37-72    |  "36Kb"   |      512    |     9-bit    |    8-bit   //
//    19-36    |  "36Kb"   |     1024    |    10-bit    |    4-bit   //
//    19-36    |  "18Kb"   |      512    |     9-bit    |    4-bit   //
//    10-18    |  "36Kb"   |     2048    |    11-bit    |    2-bit   //
//    10-18    |  "18Kb"   |     1024    |    10-bit    |    2-bit   //
//     5-9     |  "36Kb"   |     4096    |    12-bit    |    1-bit   //
//     5-9     |  "18Kb"   |     2048    |    11-bit    |    1-bit   //
//     3-4     |  "36Kb"   |     8192    |    13-bit    |    1-bit   //
//     3-4     |  "18Kb"   |     4096    |    12-bit    |    1-bit   //
//       2     |  "36Kb"   |    16384    |    14-bit    |    1-bit   //
//       2     |  "18Kb"   |     8192    |    13-bit    |    1-bit   //
//       1     |  "36Kb"   |    32768    |    15-bit    |    1-bit   //
//       1     |  "18Kb"   |    16384    |    14-bit    |    1-bit   //
//////////////////////////////////////////////////////////////////////

BRAM_SDP_MACRO #(
   .BRAM_SIZE("18Kb"), // Target BRAM, "18Kb" or "36Kb"
   .DEVICE("7SERIES"), // Target device: "7SERIES"
   .WRITE_WIDTH(0),    // Valid values are 1-72 (37-72 only valid when BRAM_SIZE="36Kb")
   .READ_WIDTH(0),     // Valid values are 1-72 (37-72 only valid when BRAM_SIZE="36Kb")
   .DO_REG(0),         // Optional output register (0 or 1)
   .INIT_FILE ("NONE"),
   .SIM_COLLISION_CHECK ("ALL"), // Collision check enable "ALL", "WARNING_ONLY",
                                 //   "GENERATE_X_ONLY" or "NONE"
   .SRVAL(72'h000000000000000000), // Set/Reset value for port output
   .INIT(72'h000000000000000000),  // Initial values on output port
   .WRITE_MODE("WRITE_FIRST"),  // Specify "READ_FIRST" for same clock or synchronous clocks
                                //   Specify "WRITE_FIRST for asynchronous clocks on ports
   .INIT_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_08(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_09(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0F(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_10(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_11(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_12(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_13(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_14(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_15(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_16(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_17(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_18(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_19(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1F(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_20(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_21(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_22(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_23(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_24(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_25(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_26(256'h0000000000000000000000000000000000000000000000000000000000000000),
```

```
    .INIT_27(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_28(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_29(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_30(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_31(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_32(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_33(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_34(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_35(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_36(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_37(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_38(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_39(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3F(256'h0000000000000000000000000000000000000000000000000000000000000000),

    // The next set of INIT_xx are valid when configured as 36Kb
    .INIT_40(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_41(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_42(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_43(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_44(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_45(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_46(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_47(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_48(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_49(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_50(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_51(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_52(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_53(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_54(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_55(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_56(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_57(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_58(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_59(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_60(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_61(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_62(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_63(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_64(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_65(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_66(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_67(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_68(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_69(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_70(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_71(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_72(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_73(256'h0000000000000000000000000000000000000000000000000000000000000000),
```

```
   .INIT_74(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_75(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_76(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_77(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_78(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_79(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7F(256'h0000000000000000000000000000000000000000000000000000000000000000),

   // The next set of INITP_xx are for the parity bits
   .INITP_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_07(256'h0000000000000000000000000000000000000000000000000000000000000000),

   // The next set of INITP_xx are valid when configured as 36Kb
   .INITP_08(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_09(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0F(256'h0000000000000000000000000000000000000000000000000000000000000000)
) BRAM_SDP_MACRO_inst (
   .DO(DO),          // Output read data port, width defined by READ_WIDTH parameter
   .DI(DI),          // Input write data port, width defined by WRITE_WIDTH parameter
   .RDADDR(RDADDR), // Input read address, width defined by read port depth
   .RDCLK(RDCLK),   // 1-bit input read clock
   .RDEN(RDEN),     // 1-bit input read port enable
   .REGCE(REGCE),   // 1-bit input read output register enable
   .RST(RST),       // 1-bit input reset
   .WE(WE),         // Input write enable, width defined by write port depth
   .WRADDR(WRADDR), // Input write address, width defined by write port depth
   .WRCLK(WRCLK),   // 1-bit input write clock
   .WREN(WREN)      // 1-bit input write port enable
);

// End of BRAM_SDP_MACRO_inst instantiation
```

BRAM_SINGLE_MACRO

Macro: Single Port RAM



Introduction

7 series FPGA devices contain several block RAM memories that can be configured as general-purpose 36 Kb or 18 Kb RAM/ROM memories. These single-port, block RAM memories offer fast and flexible storage of large amounts of on-chip data. Byte-enable write operations are possible, and an optional output register can be used to reduce the clock-to-out times of the RAM.

Port Descriptions

Port	Direction	Width	Function
DO	Output	See Port Configuration table	Data output bus addressed by ADDR.
DI	Input	See Port Configuration table	Data input bus addressed by ADDR.
ADDR	Input	See Port Configuration table	Address input bus.
WE	Input	See Port Configuration table	Byte-Wide Write enable.
EN	Input	1	Write/Read enables.
RST	Input	1	Output registers synchronous reset.
REGCE	Input	1	Output register clock enable input (valid only when DO_REG=1).

Port	Direction	Width	Function
CLK	Input	1	Clock input.

Port Configuration

This unimacro is a parameterizable version of the primitive, and can be instantiated only. Use this table to correctly configure the unimacro to meet design needs.

WRITE_WIDTH	READ_WIDTH	BRAM_SIZE	ADDR	WE
72 - 37	72 - 37	36 Kb	9	8
	36 - 19		10	
	18 - 10		11	
	9 - 5		12	
	4 - 3		13	
	2		14	
	1		15	
36 - 19	36 - 19	36 Kb	10	4
	18-10		11	
	9 - 5		12	
	4 - 3		13	
	2		14	
	1		15	
18 - 10	36 - 19	36 Kb	11	2
	18-10		11	
	9 - 5		12	
	4 - 3		13	
	2		14	
	1		15	
9 - 5	36-19	36 Kb	12	1
	18-10		12	
	9 - 5		12	
	4 - 3		13	
	2		14	
	1		15	
4 - 3	36-19	36 Kb	13	1
	18-10		13	
	9 - 5		13	
	4 - 3		13	
	2		14	
	1		15	

WRITE_WIDTH	READ_WIDTH	BRAM_SIZE	ADDR	WE
2	36-19	36 Kb	14	1
	18-10		14	
	9 - 5		14	
	4 - 3		14	
	2		14	
	1		15	
1	36 - 19	36 Kb	15	1
	18 - 10		15	
	9 - 5		15	
	3 - 4		15	
	2		15	
	1		15	
18-10	18-10	18 Kb	10	2
	9 - 5		11	
	4 - 3		12	
	2		13	
	1		14	
9 - 5	18-10	18 Kb	11	1
	9 - 5		11	
	4 - 3		12	
	2		13	
	1		14	
4 - 3	18-10	18 Kb	12	1
	9 - 5		12	
	4 - 3		12	
	2		13	
	1		14	
2	18-10	18 Kb	13	1
	9 - 5		13	
	4 - 3		13	
	2		13	
	1		14	
1	18-10	18 Kb	14	1
	9 - 5		14	
	4 - 3		14	
	2		14	
	1		14	

Design Entry Method

This unimacro is a parameterizable version of the primitive, and can be instantiated only. Consult the Port Configuration section to correctly configure this element to meet your design needs.

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	Recommended

Available Attributes

Attribute	Type	Allowed Values	Default	Description
BRAM_SIZE	STRING	"36Kb", "18Kb"	"18Kb"	Configures RAM as "36Kb" or "18Kb" memory.
DEVICE	STRING	"7SERIES"	"7SERIES"	Target hardware architecture.
DO_REG	INTEGER	0, 1	0	A value of 1 enables to the output registers to the RAM enabling quicker clock-to-out from the RAM at the expense of an added clock cycle of read latency. A value of 0 allows a read in one clock cycle but will have slower clock to out timing.
READ_WIDTH, WRITE_WIDTH	INTEGER	1 - 36	1	Specifies the size of the DI and DO buses. The following combinations are allowed: • READ_WIDTH = WRITE_WIDTH • If asymmetric, READ_WIDTH and WRITE_WIDTH must be in the ratio of 2, or must be values allowed by the unisim (1, 2, 4, 8, 9, 16, 18, 32, 36, 64, 72)
INIT_FILE	STRING	String representing file name and location.	None	Name of the file containing initial values.
WRITE_MODE	STRING	"READ_FIRST", "WRITE_FIRST", "NO_CHANGE"	"WRITE_FIRST"	Specifies write mode to the memory.
INIT	HEX	Any 72-Bit Value	All zeros	Specifies the initial value on the output after configuration.
SRVAL	HEX	Any 72-Bit Value	All zeros	Specifies the output value of on the DO port upon the assertion of the synchronous reset (RST) signal.
INIT_00 to INIT_FF	HEX	Any 256-Bit Value	All zeros	Specifies the initial contents of the 16 Kb or 32 Kb data memory array.
INITP_00 to INITP_0F	HEX	Any 256-Bit Value	All zeros	Specifies the initial contents of the 2 Kb or 4 Kb parity data memory array.

VHDL Instantiation Template

Unless they already exist, copy the following four statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
library UNIMACRO;
use unimacro.Vcomponents.all;
```

```
-- BRAM_SINGLE_MACRO: Single Port RAM
--                    7 Series
-- Xilinx HDL Language Template, version 2020.1

-- Note -  This Unimacro model assumes the port directions to be "downto".
--         Simulation of this model with "to" in the port directions could lead to erroneous results.

-----------------------------------------------------------------------
--  READ_WIDTH | BRAM_SIZE | READ Depth  | ADDR Width |           --
-- WRITE_WIDTH |           | WRITE Depth |            |  WE Width  --
-- ============|===========|=============|============|============--
--    37-72    |  "36Kb"   |      512    |    9-bit   |    8-bit   --
--    19-36    |  "36Kb"   |     1024    |   10-bit   |    4-bit   --
--    19-36    |  "18Kb"   |      512    |    9-bit   |    4-bit   --
--    10-18    |  "36Kb"   |     2048    |   11-bit   |    2-bit   --
--    10-18    |  "18Kb"   |     1024    |   10-bit   |    2-bit   --
--     5-9     |  "36Kb"   |     4096    |   12-bit   |    1-bit   --
--     5-9     |  "18Kb"   |     2048    |   11-bit   |    1-bit   --
--     3-4     |  "36Kb"   |     8192    |   13-bit   |    1-bit   --
--     3-4     |  "18Kb"   |     4096    |   12-bit   |    1-bit   --
--       2     |  "36Kb"   |    16384    |   14-bit   |    1-bit   --
--       2     |  "18Kb"   |     8192    |   13-bit   |    1-bit   --
--       1     |  "36Kb"   |    32768    |   15-bit   |    1-bit   --
--       1     |  "18Kb"   |    16384    |   14-bit   |    1-bit   --
-----------------------------------------------------------------------

BRAM_SINGLE_MACRO_inst : BRAM_SINGLE_MACRO
generic map (
   BRAM_SIZE => "18Kb", -- Target BRAM, "18Kb" or "36Kb"
   DEVICE => "7SERIES", -- Target Device: "VIRTEX5", "7SERIES", "VIRTEX6, "SPARTAN6"
   DO_REG => 0, -- Optional output register (0 or 1)
   INIT => X"000000000000000000",   --  Initial values on output port
   INIT_FILE => "NONE",
   WRITE_WIDTH => 0,   -- Valid values are 1-72 (37-72 only valid when BRAM_SIZE="36Kb")
   READ_WIDTH => 0,   -- Valid values are 1-72 (37-72 only valid when BRAM_SIZE="36Kb")
   SRVAL => X"000000000000000000",   -- Set/Reset value for port output
   WRITE_MODE => "WRITE_FIRST", -- "WRITE_FIRST", "READ_FIRST" or "NO_CHANGE"
   -- The following INIT_xx declarations specify the initial contents of the RAM
   INIT_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_07 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_10 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_11 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_12 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_13 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_14 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_15 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_16 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_17 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_18 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_19 => X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
   INIT_1A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_20 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_21 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_22 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_23 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_24 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_25 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_26 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_27 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_28 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_29 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_30 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_31 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_32 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_33 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_34 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_35 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_36 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_37 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_38 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_39 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3F => X"0000000000000000000000000000000000000000000000000000000000000000",

   -- The next set of INIT_xx are valid when configured as 36Kb
   INIT_40 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_41 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_42 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_43 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_44 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_45 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_46 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_47 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_48 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_49 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_4A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_4B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_4C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_4D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_4E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_4F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_50 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_51 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_52 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_53 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_54 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_55 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_56 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_57 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_58 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_59 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_60 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_61 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_62 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_63 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_64 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_65 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_66 => X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
   INIT_67 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_68 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_69 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_70 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_71 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_72 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_73 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_74 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_75 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_76 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_77 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_78 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_79 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7F => X"0000000000000000000000000000000000000000000000000000000000000000",

   -- The next set of INITP_xx are for the parity bits
   INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000",

   -- The next set of INIT_xx are valid when configured as 36Kb
   INITP_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0F => X"0000000000000000000000000000000000000000000000000000000000000000")
port map (
   DO => DO,      -- Output data, width defined by READ_WIDTH parameter
   ADDR => ADDR,  -- Input address, width defined by read/write port depth
   CLK => CLK,    -- 1-bit input clock
   DI => DI,      -- Input data port, width defined by WRITE_WIDTH parameter
   EN => EN,      -- 1-bit input RAM enable
   REGCE => REGCE, -- 1-bit input output register enable
   RST => RST,    -- 1-bit input reset
   WE => WE       -- Input write enable, width defined by write port depth
);

-- End of BRAM_SINGLE_MACRO_inst instantiation
```

Verilog Instantiation Template

```
// BRAM_SINGLE_MACRO: Single Port RAM
//                    7 Series
// Xilinx HDL Language Template, version 2020.1

////////////////////////////////////////////////////////////////////
//  READ_WIDTH | BRAM_SIZE | READ Depth  | ADDR Width |            //
// WRITE_WIDTH |           | WRITE Depth |            |  WE Width  //
// ============|===========|=============|============|============//
//    37-72    |  "36Kb"   |      512    |    9-bit   |    8-bit   //
//    19-36    |  "36Kb"   |     1024    |   10-bit   |    4-bit   //
//    19-36    |  "18Kb"   |      512    |    9-bit   |    4-bit   //
//    10-18    |  "36Kb"   |     2048    |   11-bit   |    2-bit   //
//    10-18    |  "18Kb"   |     1024    |   10-bit   |    2-bit   //
//     5-9     |  "36Kb"   |     4096    |   12-bit   |    1-bit   //
//     5-9     |  "18Kb"   |     2048    |   11-bit   |    1-bit   //
//     3-4     |  "36Kb"   |     8192    |   13-bit   |    1-bit   //
```

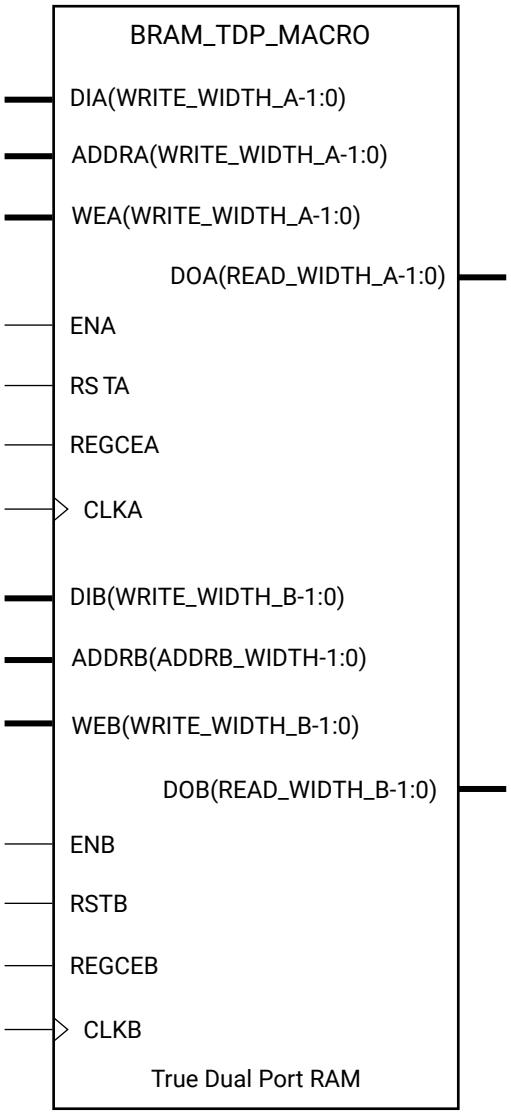
```
//      3-4      |  "18Kb"   |     4096    |   12-bit   |    1-bit   //
//        2      |  "36Kb"   |    16384    |   14-bit   |    1-bit   //
//        2      |  "18Kb"   |     8192    |   13-bit   |    1-bit   //
//        1      |  "36Kb"   |    32768    |   15-bit   |    1-bit   //
//        1      |  "18Kb"   |    16384    |   14-bit   |    1-bit   //
////////////////////////////////////////////////////////////////////////

BRAM_SINGLE_MACRO #(
   .BRAM_SIZE("18Kb"), // Target BRAM, "18Kb" or "36Kb"
   .DEVICE("7SERIES"), // Target Device: "7SERIES"
   .DO_REG(0), // Optional output register (0 or 1)
   .INIT(36'h000000000), // Initial values on output port
   .INIT_FILE ("NONE"),
   .WRITE_WIDTH(0), // Valid values are 1-72 (37-72 only valid when BRAM_SIZE="36Kb")
   .READ_WIDTH(0),  // Valid values are 1-72 (37-72 only valid when BRAM_SIZE="36Kb")
   .SRVAL(36'h000000000), // Set/Reset value for port output
   .WRITE_MODE("WRITE_FIRST"), // "WRITE_FIRST", "READ_FIRST", or "NO_CHANGE"
   .INIT_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_08(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_09(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0F(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_10(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_11(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_12(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_13(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_14(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_15(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_16(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_17(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_18(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_19(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1F(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_20(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_21(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_22(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_23(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_24(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_25(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_26(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_27(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_28(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_29(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_2A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_2B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_2C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_2D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_2E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_2F(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_30(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_31(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_32(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_33(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_34(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_35(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_36(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_37(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_38(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_39(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_3A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_3B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_3C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_3D(256'h0000000000000000000000000000000000000000000000000000000000000000),
```

```
   .INIT_3E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_3F(256'h0000000000000000000000000000000000000000000000000000000000000000),

   // The next set of INIT_xx are valid when configured as 36Kb
   .INIT_40(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_41(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_42(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_43(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_44(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_45(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_46(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_47(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_48(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_49(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_4A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_4B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_4C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_4D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_4E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_4F(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_50(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_51(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_52(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_53(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_54(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_55(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_56(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_57(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_58(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_59(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_5A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_5B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_5C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_5D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_5E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_5F(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_60(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_61(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_62(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_63(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_64(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_65(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_66(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_67(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_68(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_69(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_6A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_6B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_6C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_6D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_6E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_6F(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_70(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_71(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_72(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_73(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_74(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_75(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_76(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_77(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_78(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_79(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7F(256'h0000000000000000000000000000000000000000000000000000000000000000),

   // The next set of INITP_xx are for the parity bits
   .INITP_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
```

```
   // The next set of INIT_xx are valid when configured as 36Kb
   .INITP_08(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_09(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0F(256'h0000000000000000000000000000000000000000000000000000000000000000)
) BRAM_SINGLE_MACRO_inst (
   .DO(DO),       // Output data, width defined by READ_WIDTH parameter
   .ADDR(ADDR),   // Input address, width defined by read/write port depth
   .CLK(CLK),     // 1-bit input clock
   .DI(DI),       // Input data port, width defined by WRITE_WIDTH parameter
   .EN(EN),       // 1-bit input RAM enable
   .REGCE(REGCE), // 1-bit input output register enable
   .RST(RST),     // 1-bit input reset
   .WE(WE)        // Input write enable, width defined by write port depth
);

// End of BRAM_SINGLE_MACRO_inst instantiation
```

BRAM_TDP_MACRO

Macro: True Dual Port RAM



Introduction

7 series FPGA devices contain several block RAM memories that can be configured as general-purpose 36 kb or 18 kb RAM/ROM memories. These block RAM memories offer fast and flexible storage of large amounts of on-chip data. Both read and write operations are fully synchronous to the supplied clock(s) of the component. However, READ and WRITE ports can operate fully independently and asynchronous to each other, accessing the same memory array. Byte-enable write operations are possible, and an optional output register can be used to reduce the clock-to-out times of the RAM.

Port Descriptions

Port	Direction	Width	Function
DOA	Output	See Port Configuration table	Data output bus addressed by ADDRA.
DOB	Output	See Port Configuration table	Data output bus addressed by ADDRB.
DIA	Input	See Port Configuration table	Data input bus addressed by ADDRA.
DIB	Input	See Port Configuration table	Data input bus addressed by ADDRB.
ADDRA, ADDRB	Input	See Port Configuration table	Address input buses for Port A, B.
WEA, WEB	Input	See Port Configuration table	Write enable for Port A, B.
ENA, ENB	Input	1	Write/Read enables for Port A, B.
RSTA, RSTB	Input	1	Output registers synchronous reset for Port A, B.
REGCEA, REGCEB	Input	1	Output register clock enable input for Port A, B (valid only when DO_REG=1).
CLKA, CLKB	Input	1	Write/Read clock input for Port A, B.

Port Configuration

This unimacro is a parameterizable version of the primitive, and can be instantiated only. Use this table to correctly configure the unimacro to meet design needs.

WRITE_WIDTH_A/B-DIA/DIB	READ_WIDTH_A/B-DOA/DOB	BRAM_SIZE	ADDRA/B	WEA/B
36 - 19	36 - 19	36 Kb	10	4
	18-10		11	
	9 - 5		12	
	4 - 3		13	
	2		14	
	1		15	
18 - 10	36 - 19	36 Kb	11	2
	18-10		11	
	9 - 5		12	
	4 - 3		13	
	2		14	
	1		15	
9 - 5	36-19	36 Kb	12	1
	18-10		12	
	9 - 5		12	
	4 - 3		13	
	2		14	
	1		15	
4 - 3	36-19	36 Kb	13	1
	18-10		13	
	9 - 5		13	
	4 - 3		13	
	2		14	
	1		15	
2	36-19	36 Kb	14	1
	18-10		14	
	9 - 5		14	
	4 - 3		14	
	2		14	
	1		15	
1	36-19	36 Kb	15	1
	18-10		15	
	9 - 5		15	
	4 - 3		15	
	2		15	
	1		15	

WRITE_WIDTH_A/B-DIA/DIB	READ_WIDTH_A/B-DOA/DOB	BRAM_SIZE	ADDRA/B	WEA/B
18-10	18-10	18 Kb	10	2
	9 - 5		11	
	4 - 3		12	
	2		13	
	1		14	
9 - 5	18-10	18 Kb	11	1
	9 - 5		11	
	4 - 3		12	
	2		13	
	1		14	
4 - 3	18-10	18 Kb	12	1
	9 - 5		12	
	4 - 3		12	
	2		13	
	1		14	
2	18-10	18 Kb	13	1
	9 - 5		13	
	4 - 3		13	
	2		13	
	1		14	
1	18-10	18 Kb	14	1
	9 - 5		14	
	4 - 3		14	
	2		14	
	1		14	

Design Entry Method

This unimacro is a parameterizable version of the primitive, and can be instantiated only. Consult the Port Configuration section to correctly configure this element to meet your design needs.

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	Recommended

Available Attributes

Attribute(s)	Type	Allowed Values	Default	Description
BRAM_SIZE	STRING	"36Kb", "18Kb"	"18Kb"	Configures RAM as "36Kb" or "18Kb" memory.

Attribute(s)	Type	Allowed Values	Default	Description
DEVICE	STRING	"7SERIES"	"7SERIES"	Target hardware architecture.
DO_REG	INTEGER	0, 1	0	A value of 1 enables to the output registers to the RAM enabling quicker clock-to-out from the RAM at the expense of an added clock cycle of read latency. A value of 0 allows a read in one clock cycle but will have slower clock to out timing.
INIT	HEX	Any 72-Bit Value	All zeros	Specifies the initial value on the output after configuration.
INIT_FILE	STRING	String representing file name and location.	NONE	Name of file containing initial values.
READ_WIDTH, WRITE_WIDTH	INTEGER	1 - 72	36	Specifies the size of the DI and DO buses. The following combinations are allowed: • READ_WIDTH = WRITE_WIDTH • If asymmetric, READ_WIDTH and WRITE_WIDTH must be in the ratio of 2, or must be values allowed by the unisim (1, 2, 4, 8, 9, 16, 18, 32, 36)
SIM_COLLISION _CHECK	STRING	"ALL","WARNING_ ONLY","GENERATE _X_ONLY","NONE"	"ALL"	Allows modification of the simulation behavior if a memory collision occurs. The output is affected as follows: • "ALL": Warning produced and affected outputs/memory location go unknown (X). • "WARNING_ONLY": Warning produced and affected outputs/memory retain last value. • "GENERATE_X_ONLY": No warning. However, affected outputs/memory go unknown (X). • "NONE": No warning and affected outputs/ memory retain last value. ***Note*:** Setting this to a value other than "ALL" can allow problems in the design go unnoticed during simulation. Care should be taken when changing the value of this attribute. Please see the *Synthesis and Simulation Design Guide* for more information.
SRVAL A, SRVAL_B	HEX	Any 72-Bit Value	All zeros	Specifies the output value of on the DO port upon the assertion of the synchronous reset (RST) signal.
INIT_00 to INIT_FF	HEX	Any 256-Bit Value	All zeros	Specifies the initial contents of the 16 Kb or 32 Kb data memory array.
INITP_00 to INITP_0F	HEX	Any 256-Bit Value	All zeros	Specifies the initial contents of the 2 Kb or 4 Kb parity data memory array.

VHDL Instantiation Template

Unless they already exist, copy the following four statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
library UNIMACRO;
use unimacro.Vcomponents.all;
```

```
-- BRAM_TDP_MACRO: True Dual Port RAM
--                 7 Series
-- Xilinx HDL Language Template, version 2020.1

-- Note -  This Unimacro model assumes the port directions to be "downto".
--         Simulation of this model with "to" in the port directions could lead to erroneous results.

-----------------------------------------------------------------------
-- DATA_WIDTH_A/B | BRAM_SIZE | RAM Depth | ADDRA/B Width | WEA/B Width --
-- ===============|===========|===========|===============|=============--
--     19-36      |  "36Kb"   |    1024   |    10-bit     |    4-bit    --
--     10-18      |  "36Kb"   |    2048   |    11-bit     |    2-bit    --
--     10-18      |  "18Kb"   |    1024   |    10-bit     |    2-bit    --
--      5-9       |  "36Kb"   |    4096   |    12-bit     |    1-bit    --
--      5-9       |  "18Kb"   |    2048   |    11-bit     |    1-bit    --
--      3-4       |  "36Kb"   |    8192   |    13-bit     |    1-bit    --
--      3-4       |  "18Kb"   |    4096   |    12-bit     |    1-bit    --
--        2       |  "36Kb"   |   16384   |    14-bit     |    1-bit    --
--        2       |  "18Kb"   |    8192   |    13-bit     |    1-bit    --
--        1       |  "36Kb"   |   32768   |    15-bit     |    1-bit    --
--        1       |  "18Kb"   |   16384   |    14-bit     |    1-bit    --
-----------------------------------------------------------------------

BRAM_TDP_MACRO_inst : BRAM_TDP_MACRO
generic map (
   BRAM_SIZE => "18Kb", -- Target BRAM, "18Kb" or "36Kb"
   DEVICE => "7SERIES", -- Target Device: "VIRTEX5", "VIRTEX6", "7SERIES", "SPARTAN6"
   DOA_REG => 0, -- Optional port A output register (0 or 1)
   DOB_REG => 0, -- Optional port B output register (0 or 1)
   INIT_A => X"000000000", -- Initial values on A output port
   INIT_B => X"000000000", -- Initial values on B output port
   INIT_FILE => "NONE",
   READ_WIDTH_A => 0,   -- Valid values are 1-36 (19-36 only valid when BRAM_SIZE="36Kb")
   READ_WIDTH_B => 0,   -- Valid values are 1-36 (19-36 only valid when BRAM_SIZE="36Kb")
   SIM_COLLISION_CHECK => "ALL", -- Collision check enable "ALL", "WARNING_ONLY",
                                 -- "GENERATE_X_ONLY" or "NONE"
   SRVAL_A => X"000000000",   -- Set/Reset value for A port output
   SRVAL_B => X"000000000",   -- Set/Reset value for B port output
   WRITE_MODE_A => "WRITE_FIRST", -- "WRITE_FIRST", "READ_FIRST" or "NO_CHANGE"
   WRITE_MODE_B => "WRITE_FIRST", -- "WRITE_FIRST", "READ_FIRST" or "NO_CHANGE"
   WRITE_WIDTH_A => 0, -- Valid values are 1-36 (19-36 only valid when BRAM_SIZE="36Kb")
   WRITE_WIDTH_B => 0, -- Valid values are 1-36 (19-36 only valid when BRAM_SIZE="36Kb")
   -- The following INIT_xx declarations specify the initial contents of the RAM
   INIT_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_07 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_10 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_11 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_12 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_13 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_14 => X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
    INIT_15 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_16 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_17 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_18 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_19 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1F => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_20 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_21 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_22 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_23 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_24 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_25 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_26 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_27 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_28 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_29 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2F => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_30 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_31 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_32 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_33 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_34 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_35 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_36 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_37 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_38 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_39 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3F => X"0000000000000000000000000000000000000000000000000000000000000000",

    -- The next set of INIT_xx are valid when configured as 36Kb
    INIT_40 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_41 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_42 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_43 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_44 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_45 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_46 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_47 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_48 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_49 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_4F => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_50 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_51 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_52 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_53 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_54 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_55 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_56 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_57 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_58 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_59 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_5A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_5B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_5C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_5D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_5E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_5F => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_60 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_61 => X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
   INIT_62 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_63 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_64 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_65 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_66 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_67 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_68 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_69 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_70 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_71 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_72 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_73 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_74 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_75 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_76 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_77 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_78 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_79 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7F => X"0000000000000000000000000000000000000000000000000000000000000000",

   -- The next set of INITP_xx are for the parity bits
   INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000",

   -- The next set of INIT_xx are valid when configured as 36Kb
   INITP_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0F => X"0000000000000000000000000000000000000000000000000000000000000000")
port map (
   DOA => DOA,       -- Output port-A data, width defined by READ_WIDTH_A parameter
   DOB => DOB,       -- Output port-B data, width defined by READ_WIDTH_B parameter
   ADDRA => ADDRA,   -- Input port-A address, width defined by Port A depth
   ADDRB => ADDRB,   -- Input port-B address, width defined by Port B depth
   CLKA => CLKA,     -- 1-bit input port-A clock
   CLKB => CLKB,     -- 1-bit input port-B clock
   DIA => DIA,       -- Input port-A data, width defined by WRITE_WIDTH_A parameter
   DIB => DIB,       -- Input port-B data, width defined by WRITE_WIDTH_B parameter
   ENA => ENA,       -- 1-bit input port-A enable
   ENB => ENB,       -- 1-bit input port-B enable
   REGCEA => REGCEA, -- 1-bit input port-A output register enable
   REGCEB => REGCEB, -- 1-bit input port-B output register enable
   RSTA => RSTA,     -- 1-bit input port-A reset
   RSTB => RSTB,     -- 1-bit input port-B reset
   WEA => WEA,       -- Input port-A write enable, width defined by Port A depth
   WEB => WEB        -- Input port-B write enable, width defined by Port B depth
);

-- End of BRAM_TDP_MACRO_inst instantiation
```

Verilog Instantiation Template

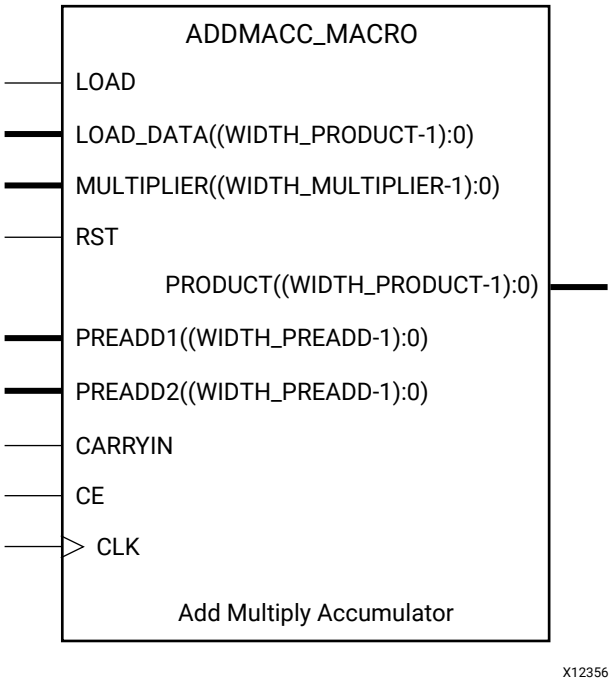
```
// BRAM_TDP_MACRO: True Dual Port RAM
//                 7 Series
// Xilinx HDL Language Template, version 2020.1

//////////////////////////////////////////////////////////////////////////////
// DATA_WIDTH_A/B | BRAM_SIZE | RAM Depth | ADDRA/B Width | WEA/B Width //
// ===============|===========|===========|===============|=============//
//     19-36      |  "36Kb"   |    1024   |    10-bit     |    4-bit    //
//     10-18      |  "36Kb"   |    2048   |    11-bit     |    2-bit    //
//     10-18      |  "18Kb"   |    1024   |    10-bit     |    2-bit    //
//      5-9       |  "36Kb"   |    4096   |    12-bit     |    1-bit    //
//      5-9       |  "18Kb"   |    2048   |    11-bit     |    1-bit    //
//      3-4       |  "36Kb"   |    8192   |    13-bit     |    1-bit    //
//      3-4       |  "18Kb"   |    4096   |    12-bit     |    1-bit    //
//        2       |  "36Kb"   |   16384   |    14-bit     |    1-bit    //
//        2       |  "18Kb"   |    8192   |    13-bit     |    1-bit    //
//        1       |  "36Kb"   |   32768   |    15-bit     |    1-bit    //
//        1       |  "18Kb"   |   16384   |    14-bit     |    1-bit    //
//////////////////////////////////////////////////////////////////////////////

BRAM_TDP_MACRO #(
   .BRAM_SIZE("18Kb"), // Target BRAM: "18Kb" or "36Kb"
   .DEVICE("7SERIES"), // Target device: "7SERIES"
   .DOA_REG(0),        // Optional port A output register (0 or 1)
   .DOB_REG(0),        // Optional port B output register (0 or 1)
   .INIT_A(36'h0000000),  // Initial values on port A output port
   .INIT_B(36'h00000000), // Initial values on port B output port
   .INIT_FILE ("NONE"),
   .READ_WIDTH_A (0),   // Valid values are 1-36 (19-36 only valid when BRAM_SIZE="36Kb")
   .READ_WIDTH_B (0),   // Valid values are 1-36 (19-36 only valid when BRAM_SIZE="36Kb")
   .SIM_COLLISION_CHECK ("ALL"), // Collision check enable "ALL", "WARNING_ONLY",
                                 //   "GENERATE_X_ONLY" or "NONE"
   .SRVAL_A(36'h00000000), // Set/Reset value for port A output
   .SRVAL_B(36'h00000000), // Set/Reset value for port B output
   .WRITE_MODE_A("WRITE_FIRST"), // "WRITE_FIRST", "READ_FIRST", or "NO_CHANGE"
   .WRITE_MODE_B("WRITE_FIRST"), // "WRITE_FIRST", "READ_FIRST", or "NO_CHANGE"
   .WRITE_WIDTH_A(0), // Valid values are 1-36 (19-36 only valid when BRAM_SIZE="36Kb")
   .WRITE_WIDTH_B(0), // Valid values are 1-36 (19-36 only valid when BRAM_SIZE="36Kb")
   .INIT_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_08(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_09(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0F(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_10(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_11(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_12(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_13(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_14(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_15(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_16(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_17(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_18(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_19(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1F(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_20(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_21(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_22(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_23(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_24(256'h0000000000000000000000000000000000000000000000000000000000000000),
```

```
    .INIT_25(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_26(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_27(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_28(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_29(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_30(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_31(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_32(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_33(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_34(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_35(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_36(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_37(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_38(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_39(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3F(256'h0000000000000000000000000000000000000000000000000000000000000000),

    // The next set of INIT_xx are valid when configured as 36Kb
    .INIT_40(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_41(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_42(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_43(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_44(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_45(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_46(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_47(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_48(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_49(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_50(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_51(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_52(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_53(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_54(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_55(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_56(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_57(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_58(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_59(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_60(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_61(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_62(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_63(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_64(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_65(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_66(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_67(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_68(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_69(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_70(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_71(256'h0000000000000000000000000000000000000000000000000000000000000000),
```

```
   .INIT_72(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_73(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_74(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_75(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_76(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_77(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_78(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_79(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_7F(256'h0000000000000000000000000000000000000000000000000000000000000000),

   // The next set of INITP_xx are for the parity bits
   .INITP_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_07(256'h0000000000000000000000000000000000000000000000000000000000000000),

   // The next set of INITP_xx are valid when configured as 36Kb
   .INITP_08(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_09(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0F(256'h0000000000000000000000000000000000000000000000000000000000000000)
) BRAM_TDP_MACRO_inst (
   .DOA(DOA),       // Output port-A data, width defined by READ_WIDTH_A parameter
   .DOB(DOB),       // Output port-B data, width defined by READ_WIDTH_B parameter
   .ADDRA(ADDRA),   // Input port-A address, width defined by Port A depth
   .ADDRB(ADDRB),   // Input port-B address, width defined by Port B depth
   .CLKA(CLKA),     // 1-bit input port-A clock
   .CLKB(CLKB),     // 1-bit input port-B clock
   .DIA(DIA),       // Input port-A data, width defined by WRITE_WIDTH_A parameter
   .DIB(DIB),       // Input port-B data, width defined by WRITE_WIDTH_B parameter
   .ENA(ENA),       // 1-bit input port-A enable
   .ENB(ENB),       // 1-bit input port-B enable
   .REGCEA(REGCEA), // 1-bit input port-A output register enable
   .REGCEB(REGCEB), // 1-bit input port-B output register enable
   .RSTA(RSTA),     // 1-bit input port-A reset
   .RSTB(RSTB),     // 1-bit input port-B reset
   .WEA(WEA),       // Input port-A write enable, width defined by Port A depth
   .WEB(WEB)        // Input port-B write enable, width defined by Port B depth
);

// End of BRAM_TDP_MACRO_inst instantiation
```

ADDMACC_MACRO

Macro: Adder/Multiplier/Accumulator



Introduction

ADDMACC_MACRO simplifies the instantiation of the DSP48 block when used as a pre-add, multiply accumulate function. It features parameterizable input and output widths and latency that ease the integration of DSP48 block into HDL.

Port Descriptions

Port	Direction	Width	Function
PRODUCT	Output	Variable width, equals the value of the WIDTH_A attribute plus the value of the WIDTH_B attribute.	Primary data output.

Port	Direction	Width	Function
PREADD1	Input	Variable, see WIDTH_PREADD attribute.	Preadder data input.
PREADD2	Input	Variable, see WIDTH_PREADD attribute.	Preadder data input
MULTIPLIER	Input	Variable, see WIDTH_MULTIPLIER attribute.	Multiplier data input
CARRYIN	Input	1	Carry input
CLK	Input	1	Clock
CE	Input	1	Clock enable
LOAD	Input	1	Load
LOAD_DATA	Input	Variable, see WIDTH_PRODUCT attribute.	In a DSP slice, when LOAD is asserted, loads P with A*B +LOAD_DATA.
RST	Input	1	Synchronous Reset

Design Entry Method

This unimacro is a parameterizable version of the primitive, and can be instantiated only.

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	Recommended

Available Attributes

Attribute	Type	Allowed Values	Default	Description
WIDTH_PREADD	INTEGER	1 to 24	24	Controls the width of PREADD1 and PREADD2 inputs.
WIDTH_MULTIPLIER	INTEGER	1 to 18	18	Controls the width of MULTIPLIER input.
WIDTH_PRODUCT	INTEGER	1 to 48	48	Controls the width of MULTIPLIER output.

Attribute	Type	Allowed Values	Default	Description
LATENCY	INTEGER	0, 1, 2, 3, 4	3	Number of pipeline registers • 1: MREG == 1 • 2: AREG == BREG == 1 and MREG == 1 or MREG == 1 and PREG == 1 • 3: AREG == BREG == 1 and MREG == 1 and PREG == 1 • 4: AREG == BREG == 2 and MREG == 1 and PREG == 1
DEVICE	STRING	"7SERIES"	"7SERIES"	Target hardware architecture.

VHDL Instantiation Template

Unless they already exist, copy the following four statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
library UNIMACRO;
use unimacro.Vcomponents.all;
```

```
-- ADDMACC_MACRO: Add and Multiple Accumulate Function implemented in a DSP48E
--                7 Series
-- Xilinx HDL Language Template, version 2020.1

ADDMACC_MACRO_inst : ADDMACC_MACRO
generic map (
   DEVICE => "7SERIES",  -- Target Device: "7SERIES", "VIRTEX6", "SPARTAN6"
   LATENCY => 4,         -- Desired clock cycle latency, 1-4
   WIDTH_PREADD => 25,   -- Pre-Adder input bus width, 1-25
   WIDTH_MULTIPLIER => 18, -- Multiplier input bus width, 1-18
   WIDTH_PRODUCT => 48)    -- MACC output width, 1-48
port map (
   PRODUCT => PRODUCT,     -- MACC result output, width defined by WIDTH_PRODUCT generic
   MULTIPLIER => MULTIPLIER, -- Multiplier data input, width determined by WIDTH_MULTIPLIER generic
   PREADDER1 => PREADDER1,   -- Preadder data input, width determined by WIDTH_PREADDER generic
   PREADDER2 => PREADDER2,   -- Preadder data input, width determined by WIDTH_PREADDER generic
   CARRYIN => CARRYIN, -- 1-bit carry-in input
   CE => CE,       -- 1-bit input clock enable
   CLK => CLK,     -- 1-bit clock input
   LOAD => LOAD, -- 1-bit accumulator load input
   LOAD_DATA => LOAD_DATA, -- Accumulator load data input, width defined by WIDTH_PRODUCT generic
   RST => RST    -- 1-bit input active high synchronous reset
);
-- End of ADDMACC_MACRO_inst instantiation
```

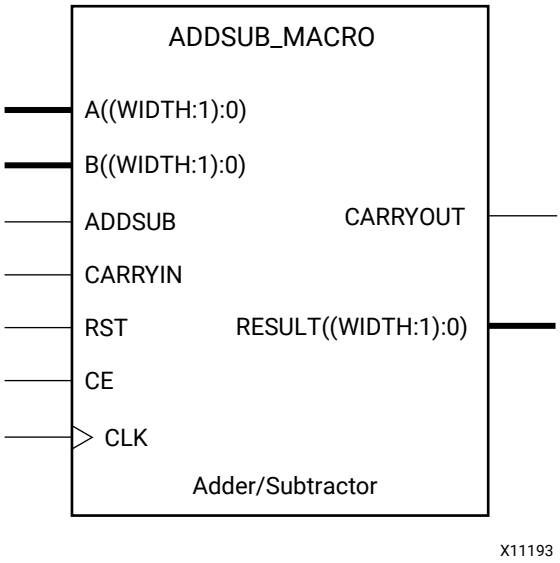
Verilog Instantiation Template

```
// ADDMACC_MACRO: Variable width & latency - Pre-Add -> Multiplier -> Accumulate
//                function implemented in a DSP48E
//                7 Series
// Xilinx HDL Language Template, version 2020.1

ADDMACC_MACRO #(
   .DEVICE("7SERIES"),     // Target Device: "7SERIES"
   .LATENCY(4),            // Desired clock cycle latency, 0-4
   .WIDTH_PREADD(25),      // Pre-adder input width, 1-25
   .WIDTH_MULTIPLIER(18), // Multiplier input width, 1-18
   .WIDTH_PRODUCT(48)      // MACC output width, 1-48
) ADDMACC_MACRO_inst (
```

```
   .PRODUCT(PRODUCT),   // MACC result output, width defined by WIDTH_PRODUCT parameter
   .CARRYIN(CARRYIN),   // 1-bit carry-in input
   .CLK(CLK),           // 1-bit clock input
   .CE(CE),             // 1-bit clock enable input
   .LOAD(LOAD),         // 1-bit accumulator load input
   .LOAD_DATA(LOAD_DATA),   // Accumulator load data input, width defined by WIDTH_PRODUCT parameter
   .MULTIPLIER(MULTIPLIER), // Multiplier data input, width defined by WIDTH_MULTIPLIER parameter
   .PREADD2(PREADD2),   // Preadder data input, width defined by WIDTH_PREADD parameter
   .PREADD1(PREADD1),   // Preadder data input, width defined by WIDTH_PREADD parameter
   .RST(RST)            // 1-bit active high synchronous reset
);

// End of ADDMACC_MACRO_inst instantiation
```

ADDSUB_MACRO

Macro: Adder/Subtracter



Introduction

ADDSUB_MACRO simplifies the instantiation of the DSP48 block when used as a simple adder/subtracter. It features parameterizable input and output widths and latency that ease the integration of the DSP48 block into HDL.

Port Descriptions

Port	Direction	Width	Function
CARRYOUT	Output	1	Carry Out
RESULT	Output	Variable, see WIDTH attribute.	Data output bus addressed by RDADDR.
ADDSUB	Input	1	When high, RESULT is an addition. When low, RESULT is a subtraction.
A	Input	Variable, see WIDTH attribute.	Data input to add/sub.
B	Input	Variable, see WIDTH attribute.	Data input to add/sub.
CE	Input	1	Clock Enable.
CARRYIN	Input	1	Carry In.
CLK	Input	1	Clock.
RST	Input	1	Synchronous Reset.

Design Entry Method

This unimacro is a parameterizable version of the primitive, and can be instantiated only.

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	Recommended

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DEVICE	STRING	"7SERIES"	"7SERIES"	Target hardware architecture.
LATENCY	INTEGER	0, 1, 2	2	Number of pipeline registers. • 1: PREG == 1 • 2: AREG == BREG == CREG == PREG
WIDTH	INTEGER	1-48	48	Result port width override.

VHDL Instantiation Template

Unless they already exist, copy the following four statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
library UNIMACRO;
use unimacro.Vcomponents.all;
```

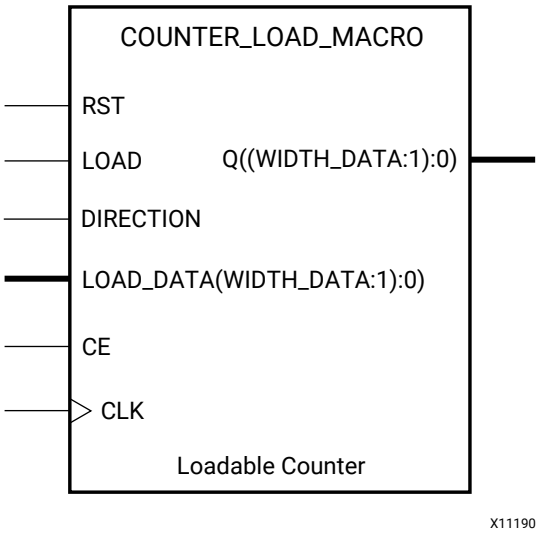
```
-- ADDSUB_MACRO: Variable width & latency - Adder / Subtrator implemented in a DSP48E
--               7 Series
-- Xilinx HDL Language Template, version 2020.1

ADDSUB_MACRO_inst : ADDSUB_MACRO
generic map (
   DEVICE => "7SERIES", -- Target Device: "VIRTEX5", "7SERIES", "SPARTAN6"
   LATENCY => 2,        -- Desired clock cycle latency, 0-2
   WIDTH => 48)         -- Input / Output bus width, 1-48
port map (
   CARRYOUT => CARRYOUT, -- 1-bit carry-out output signal
   RESULT => RESULT,     -- Add/sub result output, width defined by WIDTH generic
   A => A,               -- Input A bus, width defined by WIDTH generic
   ADD_SUB => ADD_SUB,   -- 1-bit add/sub input, high selects add, low selects subtract
   B => B,               -- Input B bus, width defined by WIDTH generic
   CARRYIN => CARRYIN,   -- 1-bit carry-in input
   CE => CE,             -- 1-bit clock enable input
   CLK =>CLK,           -- 1-bit clock input
   RST => RST            -- 1-bit active high synchronous reset
);
-- End of ADDSUB_MACRO_inst instantiation
```

Verilog Instantiation Template

```
// ADDSUB_MACRO: Variable width & latency - Adder / Subtracter implemented in a DSP48E
//               7 Series
// Xilinx HDL Language Template, version 2020.1

ADDSUB_MACRO #(
   .DEVICE("7SERIES"), // Target Device: "7SERIES"
   .LATENCY(2),        // Desired clock cycle latency, 0-2
   .WIDTH(48)          // Input / output bus width, 1-48
) ADDSUB_MACRO_inst (
   .CARRYOUT(CARRYOUT), // 1-bit carry-out output signal
   .RESULT(RESULT),     // Add/sub result output, width defined by WIDTH parameter
   .A(A),               // Input A bus, width defined by WIDTH parameter
   .ADD_SUB(ADD_SUB),   // 1-bit add/sub input, high selects add, low selects subtract
   .B(B),               // Input B bus, width defined by WIDTH parameter
   .CARRYIN(CARRYIN),   // 1-bit carry-in input
   .CE(CE),             // 1-bit clock enable input
   .CLK(CLK),           // 1-bit clock input
   .RST(RST)            // 1-bit active high synchronous reset
);

// End of ADDSUB_MACRO_inst instantiation
```

COUNTER_LOAD_MACRO

Macro: Loadable Counter



Introduction

COUNTER_LOAD_MACRO simplifies the instantiation of the DSP48 block when used as dynamic loading up/down counter. It features parameterizable output width and count by values that ease the integration of the DSP48 block into HDL.

Port Descriptions

Port	Direction	Width	Function
Q	Output	Variable, see WIDTH_DATA attribute.	Counter output.
CE	Input	1	Clock Enable.
CLK	Input	1	Clock.
LOAD	Input	Variable, see WIDTH_DATA attribute.	When asserted, loads the counter from LOAD_DATA (two-clock latency).
LOAD_DATA	Input	Variable, see WIDTH_DATA attribute.	In a DSP slice, asserting the LOAD pin will force this data into the P register with a latency of 2 clocks.
DIRECTION	Input	1	High for Up and Low for Down (two-clock latency)
RST	Input	1	Synchronous Reset

Design Entry Method

This unimacro is a parameterizable version of the primitive, and can be instantiated only.

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	Recommended

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DEVICE	STRING	"7SERIES"	"7SERIES"	Target hardware architecture.
COUNT_BY	HEX	Any 48-bit value.	000000000001	Count by *n*; takes precedence over WIDTH_DATA.
WIDTH_DATA	INTEGER	1-48	48	Specifies counter width.

VHDL Instantiation Template

Unless they already exist, copy the following four statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
library UNIMACRO;
use unimacro.Vcomponents.all;
```

```
-- COUNTER_LOAD_MACRO: Loadable variable counter implemented in a DSP48E
--                     7 Series
-- Xilinx HDL Language Template, version 2020.1

COUNTER_LOAD_MACRO_inst : COUNTER_LOAD_MACRO
generic map (
   COUNT_BY => X"000000000001", -- Count by value
   DEVICE => "7SERIES",         -- Target Device: "VIRTEX5", "7SERIES", "SPARTAN6"
   WIDTH_DATA => 48)            -- Counter output bus width, 1-48
port map (
   Q => Q,                 -- Counter ouput, width determined by WIDTH_DATA generic
   CLK => CLK,             -- 1-bit clock input
   CE => CE,               -- 1-bit clock enable input
   DIRECTION => DIRECTION, -- 1-bit up/down count direction input, high is count up
   LOAD => LOAD,           -- 1-bit active high load input
   LOAD_DATA => LOAD_DATA, -- Counter load data, width determined by WIDTH_DATA generic
   RST => RST              -- 1-bit active high synchronous reset
);
-- End of COUNTER_LOAD_MACRO_inst instantiation
```

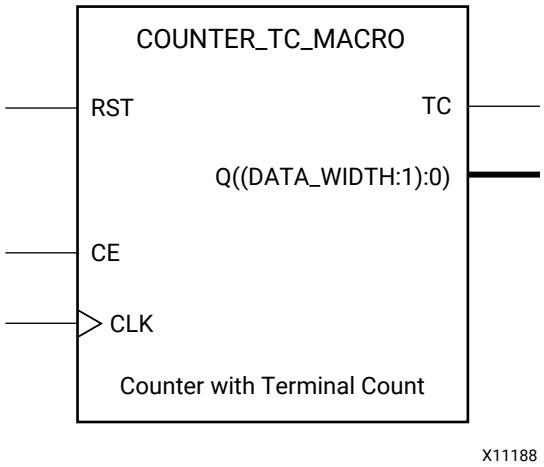
Verilog Instantiation Template

```
// COUNTER_LOAD_MACRO: Loadable variable counter implemented in a DSP48E
//                     7 Series
// Xilinx HDL Language Template, version 2020.1

COUNTER_LOAD_MACRO #(
   .COUNT_BY(48'h000000000001), // Count by value
   .DEVICE("7SERIES"), // Target Device: "7SERIES"
   .WIDTH_DATA(48)     // Counter output bus width, 1-48
) COUNTER_LOAD_MACRO_inst (
   .Q(Q),                 // Counter output, width determined by WIDTH_DATA parameter
```

```
   .CLK(CLK),              // 1-bit clock input
   .CE(CE),                // 1-bit clock enable input
   .DIRECTION(DIRECTION), // 1-bit up/down count direction input, high is count up
   .LOAD(LOAD),            // 1-bit active high load input
   .LOAD_DATA(LOAD_DATA), // Counter load data, width determined by WIDTH_DATA parameter
   .RST(RST)               // 1-bit active high synchronous reset
);

// End of COUNTER_LOAD_MACRO_inst instantiation
```

COUNTER_TC_MACRO

Macro: Counter with Terminal Count



Introduction

COUNTER_TC_MACRO simplifies the instantiation of the DSP48 block when used as a terminal count, up/down counter. It features parameterizable output width, terminal count values, count by and count direction to ease the integration of DSP48 block into HDL.

Port Descriptions

Port	Direction	Width	Function
TC	Output	1	Terminal count goes high when TC_VALUE is reached
Q	Output	Variable, see WIDTH_DATA attribute.	Counter output
CE	Input	1	Clock Enable
CLK	Input	1	Clock
RST	Input	1	Synchronous Reset

Design Entry Method

This unimacro is a parameterizable version of the primitive, and can be instantiated only.

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	Recommended

Available Attributes

Attribute	Type	Allowed Values	Default	Description
RESET_UPON_TC	BOOLEAN	True, False	False	Specifies whether to reset the counter upon reaching terminal count
DEVICE	STRING	"7SERIES"	"7SERIES"	Target hardware architecture.
DIRECTION	STRING	"UP", "DOWN"	"UP"	Count up versus count down.
COUNT_BY	HEX	Any 48-bit value	000000000001	Count by *n*; takes precedence over WIDTH_DATA.
TC_VALUE	HEX	Any 48-bit value	All zeros	Terminal count value.
WIDTH_DATA	INTEGER	1-48	48	Specifies counter width.

VHDL Instantiation Template

Unless they already exist, copy the following four statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
library UNIMACRO;
use unimacro.Vcomponents.all;
```

```
-- COUNTER_TC_MACRO: Counter with terminal count implemented in a DSP48E
--                   7 Series
-- Xilinx HDL Language Template, version 2020.1

COUNTER_TC_MACRO_inst : COUNTER_TC_MACRO
generic map (
   COUNT_BY => X"000000000001", -- Count by value
   DEVICE => "7SERIES",         -- Target Device: "VIRTEX5", "7SERIES"
   DIRECTION => "UP",            -- Counter direction "UP" or "DOWN"
   RESET_UPON_TC => "FALSE",      -- Reset counter upon terminal count, TRUE or FALSE
   TC_VALUE => X"000000000000", -- Terminal count value
   WIDTH_DATA => 48)            -- Counter output bus width, 1-48
port map (
   Q => Q,         -- Counter ouput, width determined by WIDTH_DATA generic
   TC => TC,       -- 1-bit terminal count output, high = terminal count is reached
   CLK => CLK,     -- 1-bit clock input
   CE => CE,       -- 1-bit clock enable input
   RST => RST       -- 1-bit active high synchronous reset
);
-- End of COUNTER_TC_MACRO_inst instantiation
```

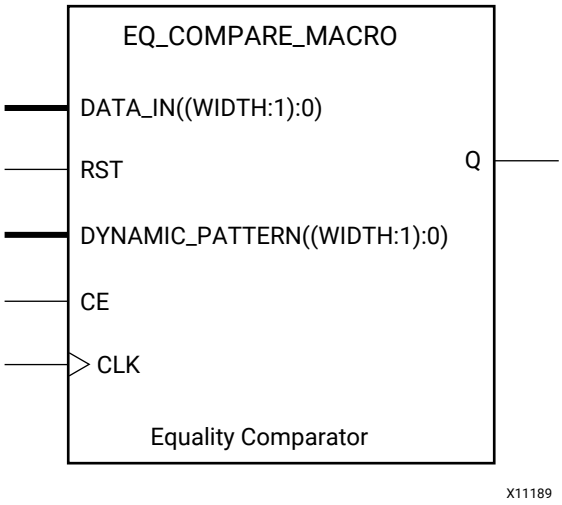
Verilog Instantiation Template

```
// COUNTER_TC_MACRO: Counter with terminal count implemented in a DSP48E
//                   7 Series
// Xilinx HDL Language Template, version 2020.1

COUNTER_TC_MACRO #(
   .COUNT_BY(48'h000000000001), // Count by value
   .DEVICE("7SERIES"),          // Target Device: "7SERIES"
   .DIRECTION("UP"),            // Counter direction, "UP" or "DOWN"
   .RESET_UPON_TC("FALSE"), // Reset counter upon terminal count, "TRUE" or "FALSE"
   .TC_VALUE(48'h000000000000), // Terminal count value
   .WIDTH_DATA(48)              // Counter output bus width, 1-48
) COUNTER_TC_MACRO_inst (
   .Q(Q),     // Counter output bus, width determined by WIDTH_DATA parameter
   .TC(TC),   // 1-bit terminal count output, high = terminal count is reached
   .CLK(CLK), // 1-bit positive edge clock input
```

```
   .CE(CE),   // 1-bit active high clock enable input
   .RST(RST)  // 1-bit active high synchronous reset
);

// End of COUNTER_TC_MACRO_inst instantiation
```

EQ_COMPARE_MACRO

Macro: Equality Comparator



Introduction

EQ_COMPARE_MACRO simplifies the instantiation of the DSP48 block when used as an equality comparator. It features parameterizable input and output widths, latencies, mask, and input sources that ease the integration of the DSP48 block into HDL.

Port Descriptions

Port	Direction	Width	Function
Q	Output	1	Active-High pattern detection. Detects match of DATA_IN and the selected DYNAMIC_PATTERN gated by the MASK. Result arrives on the same cycle as P.
DATA_IN	Input	Variable width, equals the value of the WIDTH attribute.	Input data to be compared.
DYNAMIC_PATTERN	Input	Variable width, equals the value of the WIDTH attribute.	Dynamic data to be compared to DATA_IN.
CLK	Input	1	Clock.
CE	Input	1	Clock enable.
RST	Input	1	Synchronous Reset.

Design Entry Method

This unimacro is a parameterizable version of the primitive, and can be instantiated only.

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	Recommended

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DEVICE	STRING	"7SERIES"	"7SERIES"	Target hardware architecture.
SEL_PATTERN	INTEGER	1 to 24	24	Controls the width of PREADD1 and PREADD2 inputs.
MASK	HEX	48 hex	all zeros	Mask to be used for pattern detector.
STATIC_PATTERN	HEX	48 hex	all zeros	Pattern to be used for pattern detector.
SEL_MASK	STRING	"MASK", "DYNAMIC_PATTERN"	"MASK"	Selects whether to use the static MASK or the C input for the mask of the pattern detector.
WIDTH	INTEGER	1 to 48	48	Width of DATA_IN and DYNAMIC_PATTERN.
LATENCY	INTEGER	0, 1, 2, 3	2	Number of pipeline registers. • 1: QREG == 1 • 2: AREG == BREG == CREG == QREG == 1 • 3: AREG == BREG == 2 and CREG == QREG == 1

VHDL Instantiation Template

Unless they already exist, copy the following four statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
library UNIMACRO;
use unimacro.Vcomponents.all;
```

```
-- EQ_COMPARE_MACRO: Equiality Comparator implemented in a DSP48E
--                   7 Series
-- Xilinx HDL Language Template, version 2020.1

EQ_COMPARE_MACRO_inst : EQ_COMPARE_MACRO
generic map (
   DEVICE => "7SERIES",          -- Target Device: "VIRTEX5", "7SERIES"
   LATENCY => 2,                 -- Desired clock cycle latency, 0-2
   MASK => X"000000000000",      -- Select bits to be masked, must set
                                 -- SEL_MASK = "MASK"
   SEL_MASK => "MASK",           -- "MASK" = use MASK generic,
                                 -- "DYNAMIC_PATTERN = use DYNAMIC_PATTERN input bus
   SEL_PATTERN => "DYNAMIC_PATTERN", -- "DYNAMIC_PATTERN" = use DYNAMIC_PATTERN input bus
                                   -- "STATIC_PATTERN" = use STATIC_PATTERN generic
   STATIC_PATTERN => X"000000000000", -- Specify static pattern,
                                    -- must set SEL_PATTERN = "STATIC_PATTERN
```

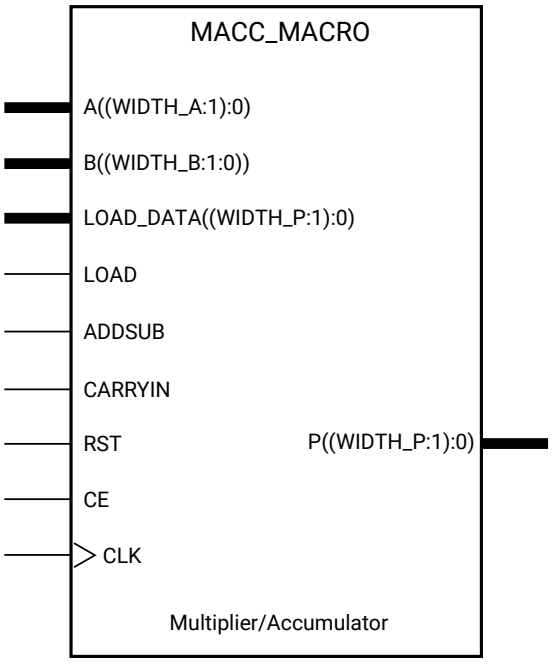
```
   WIDTH => 48)             -- Comparator output bus width, 1-48
port map (
   Q => Q,         -- 1-bit output indicating a match
   CE => CE,       -- 1-bit active high input clock enable input
   CLK => CLK,     -- 1-bit positive edge clock input
   DATA_IN => DATA_IN, -- Input Data Bus, width determined by WIDTH generic
   DYNAMIC_PATTERN, => DYNAMIC_PATTERN, -- Input Dynamic Match/Mask Bus, width determined by WIDTH generic
   RST => RST        -- 1-bit input active high reset
);
-- End of EQ_COMPARE_MACRO_inst instantiation
```

Verilog Instantiation Template

```
// EQ_COMPARE_MACRO: Equality Comparator implemented in a DSP48E
//                   7 Series
// Xilinx HDL Language Template, version 2020.1

EQ_COMPARE_MACRO #(
   .DEVICE("7SERIES"),       // Target Device: "7SERIES"
   .LATENCY(2),              // Desired clock cycle latency, 0-2
   .MASK(48'h000000000000),  // Select bits to be masked, must set SEL_MASK="MASK"
   .SEL_MASK("MASK"),        // "MASK" = use MASK parameter,
                             //   "DYNAMIC_PATTERN" = use DYNAMIC_PATTERN input bus
   .SEL_PATTERN("STATIC_PATTERN"), // "STATIC_PATTERN" = use STATIC_PATTERN parameter,
                                   //   "DYNAMIC_PATTERN = use DYNAMIC_PATTERN input bus
   .STATIC_PATTERN(48'h000000000000), // Specify static pattern, must set SEL_PATTERN = "STATIC_PATTERN"
   .WIDTH(48)                // Comparator output bus width, 1-48
) EQ_COMPARE_MACRO_inst (
   .Q(Q),     // 1-bit output indicating a match
   .CE(CE),   // 1-bit active high input clock enable
   .CLK(CLK), // 1-bit positive edge clock input
   .DATA_IN(DATA_IN), // Input Data Bus, width determined by WIDTH parameter
   .DYNAMIC_PATTERN(DYNAMIC_PATTERN), // Input Dynamic Match/Mask Bus, width determined by WIDTH parameter
   .RST(RST)  // 1-bit input active high reset
);

// End of EQ_COMPARE_MACRO_inst instantiation
```

MACC_MACRO

Macro: Multiplier/Accumulator



Introduction

MACC_MACRO simplifies the instantiation of the DSP48 block when used in simple signed multiplier/accumulator mode. It features parameterizable input and output widths and latencies that ease the integration of the DSP48 block into HDL.

Port Descriptions

Port	Direction	Width	Function
P	Output	Variable width, equals the value of the WIDTH_A attribute plus the value of the WIDTH_B attribute.	Primary data output.

Port	Direction	Width	Function
A	Input	Variable, see WIDTH_A attribute.	Multiplier data input.
B	Input	Variable, see WIDTH_B attribute.	Multiplier data input.
CARRYIN	Input	1	Carry input.
CE	Input	1	Clock enable.
CLK	Input	1	Clock.
LOAD	Input	1	Load.
LOAD_DATA	Input	Variable width, equals the value of the WIDTH_A attribute plus the value of the WIDTH_B attribute.	In a DSP slice, when LOAD is asserted, loads P with A*B +LOAD_DATA.
RST	Input	1	Synchronous Reset.
ADDSUB	Input	1	High sets accumulator in addition mode; low sets accumulator in subtraction mode.

Design Entry Method

This unimacro is a parameterizable version of the primitive, and can be instantiated only.

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	Recommended

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DEVICE	STRING	"7SERIES"	"7SERIES"	Target hardware architecture.
WIDTH_P	INTEGER	1 to 48	48	Accumulator output bus width.
WIDTH_A	INTEGER	1 to 25	25	Multiplier A-input bus width.
WIDTH_B	INTEGER	1 to 18	18	Multiplier B-input bus width.

Attribute	Type	Allowed Values	Default	Description
LATENCY	INTEGER	1, 2, 3, 4	3	Number of pipeline registers. • 1: MREG == 1 • 2: AREG == BREG == 1 and MREG == 1 or MREG == 1 and PREG == 1 • 3: AREG == BREG == 1 and MREG == 1 and PREG == 1 • 4: AREG == BREG == 2 and MREG == 1 and PREG == 1

VHDL Instantiation Template

Unless they already exist, copy the following four statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
library UNIMACRO;
use unimacro.Vcomponents.all;
```

```
-- MACC_MACRO: Multiple Accumulate Function implemented in a DSP48E
--             7 Series
-- Xilinx HDL Language Template, version 2020.1

MACC_MACRO_inst : MACC_MACRO
generic map (
   DEVICE => "7SERIES",  -- Target Device: "VIRTEX5", "7SERIES", "SPARTAN6"
   LATENCY => 3,         -- Desired clock cycle latency, 1-4
   WIDTH_A => 25,        -- Multiplier A-input bus width, 1-25
   WIDTH_B => 18,        -- Multiplier B-input bus width, 1-18
   WIDTH_P => 48)        -- Accumulator output bus width, 1-48
port map (
   P => P,      -- MACC ouput bus, width determined by WIDTH_P generic
   A => A,      -- MACC input A bus, width determined by WIDTH_A generic
   ADDSUB => ADDSUB, -- 1-bit add/sub input, high selects add, low selects subtract
   B => B,           -- MACC input B bus, width determined by WIDTH_B generic
   CARRYIN => CARRYIN, -- 1-bit carry-in input to accumulator
   CE => CE,       -- 1-bit active high input clock enable
   CLK => CLK,     -- 1-bit positive edge clock input
   LOAD => LOAD, -- 1-bit active high input load accumulator enable
   LOAD_DATA => LOAD_DATA, -- Load accumulator input data,
                           -- width determined by WIDTH_P generic
   RST => RST    -- 1-bit input active high reset
);

-- End of MACC_MACRO_inst instantiation
```

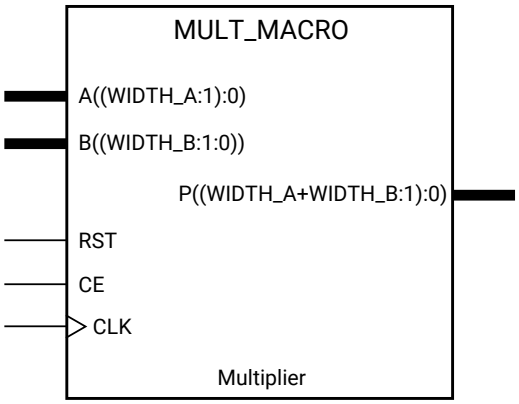
Verilog Instantiation Template

```
// MACC_MACRO: Multiply Accumulate Function implemented in a DSP48E
//             7 Series
// Xilinx HDL Language Template, version 2020.1

MACC_MACRO #(
   .DEVICE("7SERIES"), // Target Device: "7SERIES"
   .LATENCY(3),        // Desired clock cycle latency, 1-4
   .WIDTH_A(25),       // Multiplier A-input bus width, 1-25
   .WIDTH_B(18),       // Multiplier B-input bus width, 1-18
   .WIDTH_P(48)        // Accumulator output bus width, 1-48
) MACC_MACRO_inst (
   .P(P),     // MACC output bus, width determined by WIDTH_P parameter
```

```
   .A(A),      // MACC input A bus, width determined by WIDTH_A parameter
   .ADDSUB(ADDSUB), // 1-bit add/sub input, high selects add, low selects subtract
   .B(B),      // MACC input B bus, width determined by WIDTH_B parameter
   .CARRYIN(CARRYIN), // 1-bit carry-in input to accumulator
   .CE(CE),      // 1-bit active high input clock enable
   .CLK(CLK),    // 1-bit positive edge clock input
   .LOAD(LOAD), // 1-bit active high input load accumulator enable
   .LOAD_DATA(LOAD_DATA), // Load accumulator input data, width determined by WIDTH_P parameter
   .RST(RST)     // 1-bit input active high reset
);

// End of MACC_MACRO_inst instantiation
```

MULT_MACRO

Macro: Multiplier



X11191

Introduction

MULT_MACRO simplifies the instantiation of the DSP48 block when used as a simple signed multiplier. It features parameterizable input and output widths and latencies that ease the integration of the DSP48 block into HDL.

Port Descriptions

Port	Direction	Width	Function
P	Output	Variable, equals WIDTH_A + WIDTH_B.	Primary data output.
A	Input	Variable, see WIDTH_A attribute.	Multiplier data input.
B	Input	Variable, see WIDTH_B attribute.	Multiplier data input.
CE	Input	1	Clock Enable.
CLK	Input	1	Clock.
RST	Input	1	Synchronous Reset.

Design Entry Method

This unimacro is a parameterizable version of the primitive, and can be instantiated only.

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	Recommended

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DEVICE	STRING	"7SERIES"	"7SERIES"	Target hardware architecture.
WIDTH_A	INTEGER	1 to 25	18	Multiplier A-input bus width.
WIDTH_B	INTEGER	1 to 18	18	Multiplier B-input bus width.
LATENCY	INTEGER	0, 1, 2, 3, 4	3	Number of pipeline registers. • 1: MREG == 1 • 2: AREG == BREG == 1 and MREG == 1 or MREG == 1 and PREG == 1 • 3: AREG == BREG == 1 and MREG == 1 and PREG == 1 • 4: AREG == BREG == 2 and MREG == 1 and PREG == 1

VHDL Instantiation Template

Unless they already exist, copy the following four statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
library UNIMACRO;
use unimacro.Vcomponents.all;
```

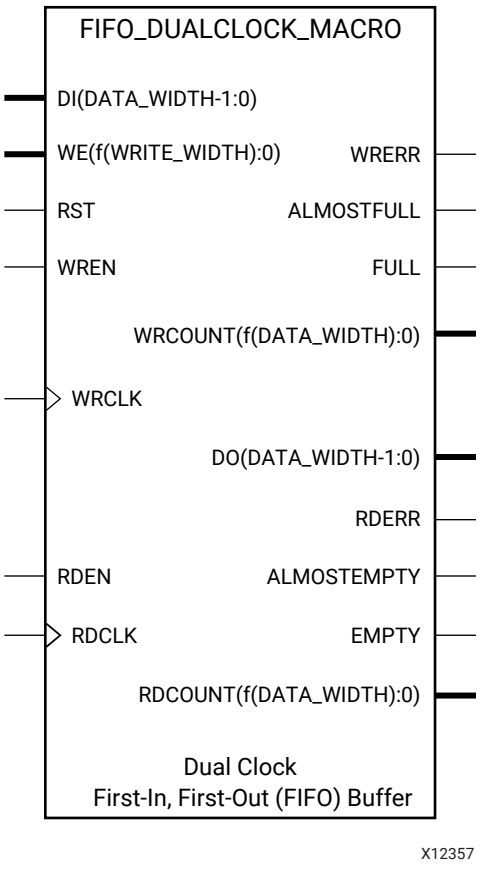
```
-- MULT_MACRO: Multiply Function implemented in a DSP48E
--             7 Series
-- Xilinx HDL Language Template, version 2020.1

MULT_MACRO_inst : MULT_MACRO
generic map (
   DEVICE => "7SERIES",     -- Target Device: "VIRTEX5", "7SERIES", "SPARTAN6"
   LATENCY => 3,            -- Desired clock cycle latency, 0-4
   WIDTH_A => 18,           -- Multiplier A-input bus width, 1-25
   WIDTH_B => 18)           -- Multiplier B-input bus width, 1-18
port map (
   P => P,      -- Multiplier ouput bus, width determined by WIDTH_P generic
   A => A,      -- Multiplier input A bus, width determined by WIDTH_A generic
   B => B,      -- Multiplier input B bus, width determined by WIDTH_B generic
   CE => CE,    -- 1-bit active high input clock enable
   CLK => CLK,  -- 1-bit positive edge clock input
   RST => RST   -- 1-bit input active high reset
);
-- End of MULT_MACRO_inst instantiation
```

Verilog Instantiation Template

```
// MULT_MACRO: Multiply Function implemented in a DSP48E
//             7 Series
// Xilinx HDL Language Template, version 2020.1

MULT_MACRO #(
   .DEVICE("7SERIES"), // Target Device: "7SERIES"
   .LATENCY(3),        // Desired clock cycle latency, 0-4
   .WIDTH_A(18),       // Multiplier A-input bus width, 1-25
   .WIDTH_B(18)        // Multiplier B-input bus width, 1-18
) MULT_MACRO_inst (
   .P(P),     // Multiplier output bus, width determined by WIDTH_P parameter
   .A(A),     // Multiplier input A bus, width determined by WIDTH_A parameter
   .B(B),     // Multiplier input B bus, width determined by WIDTH_B parameter
   .CE(CE),   // 1-bit active high input clock enable
   .CLK(CLK), // 1-bit positive edge clock input
   .RST(RST)  // 1-bit input active high reset
);

// End of MULT_MACRO_inst instantiation
```

FIFO_DUALCLOCK_MACRO

Macro: Dual Clock First-In, First-Out (FIFO) RAM Buffer



Introduction

FPGA devices contain several block RAM memories that can be configured as general-purpose 36 Kb or 18 Kb RAM/ROM memories. Dedicated logic in the block RAM enables you to easily implement FIFOs. The FIFO can be configured as an 18 Kb or 36 Kb memory. This unimacro configures the FIFO for using independent read and writes clocks. Data is read from the FIFO on the rising edge of the read clock and written to the FIFO on the rising edge of write clock.

Depending on the offset between read and write clock edges, the Empty, Almost Empty, Full and Almost Full flags can deassert one cycle later. Due to the asynchronous nature of the clocks, the simulation model only reflects the deassertion latency cycles listed in the User Guide.

Port Descriptions

Port	Direction	Width	Function
ALMOSTEMPTY	Output	1	Almost all valid entries in FIFO have been read.
ALMOSTFULL	Output	1	Almost all entries in FIFO memory have been filled.
DO	Output	See Port Configuration table.	Data output bus addressed by ADDR.
EMPTY	Output	1	FIFO is empty.
FULL	Output	1	All entries in FIFO memory are filled.
RDCOUNT	Output	See Port Configuration table.	FIFO data read pointer.
RDERR	Output	1	When the FIFO is empty, any additional read operation generates an error flag.
WRCOUNT	Output	See Port Configuration table.	FIFO data write pointer.
WRERR	Output	1	When the FIFO is full, any additional write operation generates an error flag.
DI	Input	See Port Configuration table.	Data input bus addressed by ADDR.
RDCLK	Input	1	Clock for Read domain operation.
RDEN	Input	1	Read Enable.
RST	Input	1	Asynchronous reset.
WRCLK	Input	1	Clock for Write domain operation.
WREN	Input	1	Write Enable.

Port Configuration

This unimacro is a parameterizable version of the primitive, and can be instantiated only. Use this table to correctly configure the unimacro to meet design needs.

DATA_WIDTH	FIFO_SIZE	WRCOUNT	RDCOUNT
72 - 37	36 Kb	9	9
36 - 19	36 Kb	10	10
	18 Kb	9	9
18 - 10	36 Kb	11	11
	18 Kb	10	10
9-5	36 Kb	12	12
	18 Kb	11	11
1-4	36 Kb	13	13
	18 Kb	12	12

Design Entry Method

This unimacro is a parameterizable version of the primitive, and can be instantiated only. Consult the Port Configuration section to correctly configure this element to meet your design needs.

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	Recommended

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ALMOST_EMPTY_OFFSET	HEX	13-Bit Value	All zeros	Setting determines the difference between EMPTY and ALMOSTEMPTY conditions. Must be set using hexadecimal notation.
ALMOST_FULL_OFFSET	HEX	13-Bit Value	All zeros	Setting determines the difference between FULL and ALMOSTFULL conditions. Must be set using hexadecimal notation.
DATA_WIDTH	INTEGER	1 - 72	4	Width of DI/DO bus.
DEVICE	STRING	"7SERIES"	"7SERIES"	Target hardware architecture.
FIFO_SIZE	STRING	"18Kb", "36Kb"	"18Kb"	Configures the FIFO as 18 Kb or 36 Kb memory.
FIRST_WORD_FALL_THROUGH	BOOLEAN	FALSE, TRUE	FALSE	If TRUE, the first word written into the empty FIFO appears at the FIFO output without RDEN asserted.

VHDL Instantiation Template

Unless they already exist, copy the following four statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
library UNIMACRO;
use unimacro.Vcomponents.all;
```

```
-- FIFO_DUALCLOCK_MACRO: Dual-Clock First-In, First-Out (FIFO) RAM Buffer
--                       7 Series
-- Xilinx HDL Language Template, version 2020.1

-- Note -  This Unimacro model assumes the port directions to be "downto".
--         Simulation of this model with "to" in the port directions could lead to erroneous results.

-----------------------------------------------------------------
-- DATA_WIDTH | FIFO_SIZE | FIFO Depth | RDCOUNT/WRCOUNT Width --
-- ===========|===========|============|=======================--
--   37-72    |  "36Kb"   |     512    |         9-bit         --
--   19-36    |  "36Kb"   |    1024    |        10-bit         --
--   19-36    |  "18Kb"   |     512    |         9-bit         --
--   10-18    |  "36Kb"   |    2048    |        11-bit         --
--   10-18    |  "18Kb"   |    1024    |        10-bit         --
--    5-9     |  "36Kb"   |    4096    |        12-bit         --
--    5-9     |  "18Kb"   |    2048    |        11-bit         --
--    1-4     |  "36Kb"   |    8192    |        13-bit         --
--    1-4     |  "18Kb"   |    4096    |        12-bit         --
-----------------------------------------------------------------
```

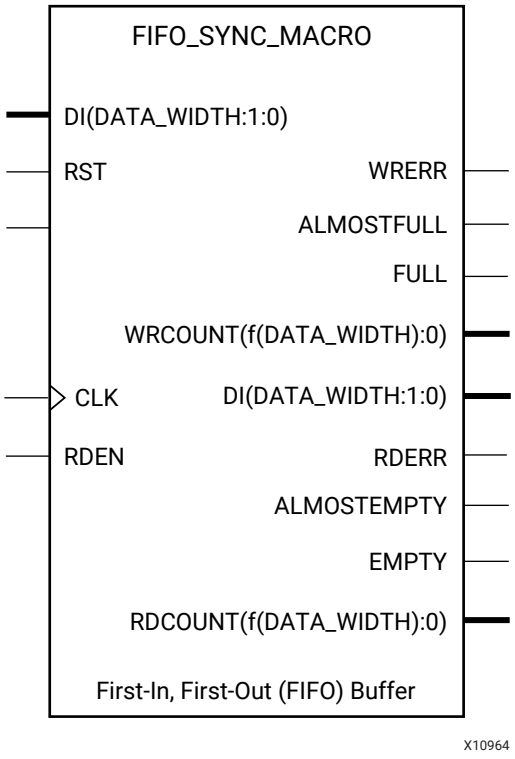
```
FIFO_DUALCLOCK_MACRO_inst : FIFO_DUALCLOCK_MACRO
generic map (
   DEVICE => "7SERIES",            -- Target Device: "VIRTEX5", "VIRTEX6", "7SERIES"
   ALMOST_FULL_OFFSET => X"0080",  -- Sets almost full threshold
   ALMOST_EMPTY_OFFSET => X"0080", -- Sets the almost empty threshold
   DATA_WIDTH => 0,   -- Valid values are 1-72 (37-72 only valid when FIFO_SIZE="36Kb")
   FIFO_SIZE => "18Kb",            -- Target BRAM, "18Kb" or "36Kb"
   FIRST_WORD_FALL_THROUGH => FALSE) -- Sets the FIFO FWFT to TRUE or FALSE
port map (
   ALMOSTEMPTY => ALMOSTEMPTY,   -- 1-bit output almost empty
   ALMOSTFULL => ALMOSTFULL,     -- 1-bit output almost full
   DO => DO,                     -- Output data, width defined by DATA_WIDTH parameter
   EMPTY => EMPTY,               -- 1-bit output empty
   FULL => FULL,                 -- 1-bit output full
   RDCOUNT => RDCOUNT,           -- Output read count, width determined by FIFO depth
   RDERR => RDERR,               -- 1-bit output read error
   WRCOUNT => WRCOUNT,           -- Output write count, width determined by FIFO depth
   WRERR => WRERR,               -- 1-bit output write error
   DI => DI,                     -- Input data, width defined by DATA_WIDTH parameter
   RDCLK => RDCLK,               -- 1-bit input read clock
   RDEN => RDEN,                 -- 1-bit input read enable
   RST => RST,                   -- 1-bit input reset
   WRCLK => WRCLK,               -- 1-bit input write clock
   WREN => WREN                  -- 1-bit input write enable
);
-- End of FIFO_DUALCLOCK_MACRO_inst instantiation
```

Verilog Instantiation Template

```
// FIFO_DUALCLOCK_MACRO: Dual Clock First-In, First-Out (FIFO) RAM Buffer
//                       7 Series
// Xilinx HDL Language Template, version 2020.1

//////////////////////////////////////////////////////////////////
// DATA_WIDTH | FIFO_SIZE | FIFO Depth | RDCOUNT/WRCOUNT Width //
// ===========|===========|============|=======================//
//   37-72    |  "36Kb"   |     512    |         9-bit         //
//   19-36    |  "36Kb"   |    1024    |        10-bit         //
//   19-36    |  "18Kb"   |     512    |         9-bit         //
//   10-18    |  "36Kb"   |    2048    |        11-bit         //
//   10-18    |  "18Kb"   |    1024    |        10-bit         //
//    5-9     |  "36Kb"   |    4096    |        12-bit         //
//    5-9     |  "18Kb"   |    2048    |        11-bit         //
//    1-4     |  "36Kb"   |    8192    |        13-bit         //
//    1-4     |  "18Kb"   |    4096    |        12-bit         //
//////////////////////////////////////////////////////////////////

FIFO_DUALCLOCK_MACRO  #(
   .ALMOST_EMPTY_OFFSET(9'h080), // Sets the almost empty threshold
   .ALMOST_FULL_OFFSET(9'h080),  // Sets almost full threshold
   .DATA_WIDTH(0),   // Valid values are 1-72 (37-72 only valid when FIFO_SIZE="36Kb")
   .DEVICE("7SERIES"),  // Target device: "7SERIES"
   .FIFO_SIZE ("18Kb"), // Target BRAM: "18Kb" or "36Kb"
   .FIRST_WORD_FALL_THROUGH ("FALSE") // Sets the FIFO FWFT to "TRUE" or "FALSE"
) FIFO_DUALCLOCK_MACRO_inst (
   .ALMOSTEMPTY(ALMOSTEMPTY), // 1-bit output almost empty
   .ALMOSTFULL(ALMOSTFULL),   // 1-bit output almost full
   .DO(DO),                   // Output data, width defined by DATA_WIDTH parameter
   .EMPTY(EMPTY),             // 1-bit output empty
   .FULL(FULL),               // 1-bit output full
   .RDCOUNT(RDCOUNT),         // Output read count, width determined by FIFO depth
   .RDERR(RDERR),             // 1-bit output read error
   .WRCOUNT(WRCOUNT),         // Output write count, width determined by FIFO depth
   .WRERR(WRERR),             // 1-bit output write error
   .DI(DI),                   // Input data, width defined by DATA_WIDTH parameter
   .RDCLK(RDCLK),             // 1-bit input read clock
   .RDEN(RDEN),               // 1-bit input read enable
   .RST(RST),                 // 1-bit input reset
   .WRCLK(WRCLK),             // 1-bit input write clock
   .WREN(WREN)                // 1-bit input write enable
);

// End of FIFO_DUALCLOCK_MACRO_inst instantiation
```

FIFO_SYNC_MACRO

Macro: Synchronous First-In, First-Out (FIFO) RAM Buffer



Introduction

FPGA devices contain several block RAM memories that can be configured as general-purpose 36 Kb or 18 Kb RAM/ROM memories. Dedicated logic in the block RAM enables you to easily implement FIFOs. The FIFO can be configured as an 18 Kb or 36 Kb memory. This unimacro configures the FIFO such that it uses one clock for reading as well as writing.

Port Descriptions

Port	Direction	Width	Function
ALMOSTEMPTY	Output	1	Almost all valid entries in FIFO have been read.
ALMOSTFULL	Output	1	Almost all entries in FIFO memory have been filled.
DO	Output	See Port Configuration table.	Data output bus addressed by ADDR.
EMPTY	Output	1	FIFO is empty.
FULL	Output	1	All entries in FIFO memory are filled.

Port	Direction	Width	Function
RDCOUNT	Output	See Port Configuration table.	FIFO data read pointer.
RDERR	Output	1	When the FIFO is empty, any additional read operation generates an error flag.
WRCOUNT	Output	See Port Configuration table.	FIFO data write pointer.
WRERR	Output	1	When the FIFO is full, any additional write operation generates an error flag.
CLK	Input	1	Clock for Read/Write domain operation.
DI	Input	See Port Configuration table.	Data input bus addressed by ADDR.
RDEN	Input	1	Read Enable
RST	Input	1	Asynchronous reset.
WREN	Input	1	Write Enable

Port Configuration

This unimacro is a parameterizable version of the primitive, and can be instantiated only. Use this table to correctly configure the unimacro to meet design needs.

DATA_WIDTH	FIFO_SIZE	WRCOUNT	RDCOUNT
72 - 37	36 Kb	9	9
36 - 19	36 Kb	10	10
	18 Kb	9	9
18 - 10	36 Kb	11	11
	18 Kb	10	10
9-5	36 Kb	12	12
	18 Kb	11	11
1-4	36 Kb	13	13
	18 Kb	12	12

Design Entry Method

This unimacro is a parameterizable version of the primitive, and can be instantiated only. Consult the Port Configuration section to correctly configure this element to meet your design needs.

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	Recommended

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ALMOST_EMPTY _OFFSET	HEX	13-bit HEX	All zeros	Setting determines the difference between EMPTY and ALMOSTEMPTY conditions. Must be set using hexadecimal notation.
ALMOST_FULL _OFFSET	HEX	13-bit HEX	All zeros	Setting determines the difference between FULL and ALMOSTFULL conditions. Must be set using hexadecimal notation.
DATA_WIDTH	INTEGER	1 - 72	4	Width of DI/DO bus.
DEVICE	STRING	"7SERIES"	"7SERIES"	Target hardware architecture.
DO_REG	BINARY	0,1	1	DO_REG must be set to 0 for flags and data to follow a standard synchronous FIFO operation. When DO_REG is set to 1, effectively a pipeline register is added to the output of the synchronous FIFO. Data then has a one clock cycle latency. However, the clock-to-out timing is improved.
FIFO_SIZE	STRING	"18Kb", "36Kb"	"18Kb"	Configures FIFO as "18Kb" or "36Kb" memory.

VHDL Instantiation Template

Unless they already exist, copy the following four statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
library UNIMACRO;
use unimacro.Vcomponents.all;
```

```
-- FIFO_SYNC_MACRO: Synchronous First-In, First-Out (FIFO) RAM Buffer
--                  7 Series
-- Xilinx HDL Language Template, version 2020.1

-- Note -  This Unimacro model assumes the port directions to be "downto".
--         Simulation of this model with "to" in the port directions could lead to erroneous results.

-----------------------------------------------------------------
-- DATA_WIDTH | FIFO_SIZE | FIFO Depth | RDCOUNT/WRCOUNT Width --
-- ===========|===========|============|=======================--
--   37-72    |  "36Kb"   |     512    |         9-bit         --
--   19-36    |  "36Kb"   |    1024    |        10-bit         --
--   19-36    |  "18Kb"   |     512    |         9-bit         --
--   10-18    |  "36Kb"   |    2048    |        11-bit         --
--   10-18    |  "18Kb"   |    1024    |        10-bit         --
--    5-9     |  "36Kb"   |    4096    |        12-bit         --
--    5-9     |  "18Kb"   |    2048    |        11-bit         --
--    1-4     |  "36Kb"   |    8192    |        13-bit         --
--    1-4     |  "18Kb"   |    4096    |        12-bit         --
-----------------------------------------------------------------


FIFO_SYNC_MACRO_inst : FIFO_SYNC_MACRO
generic map (
   DEVICE => "7SERIES",            -- Target Device: "VIRTEX5, "VIRTEX6", "7SERIES"
   ALMOST_FULL_OFFSET => X"0080",  -- Sets almost full threshold
   ALMOST_EMPTY_OFFSET => X"0080", -- Sets the almost empty threshold
   DATA_WIDTH => 0,   -- Valid values are 1-72 (37-72 only valid when FIFO_SIZE="36Kb")
   FIFO_SIZE => "18Kb")            -- Target BRAM, "18Kb" or "36Kb"
port map (
   ALMOSTEMPTY => ALMOSTEMPTY,   -- 1-bit output almost empty
   ALMOSTFULL => ALMOSTFULL,     -- 1-bit output almost full
   DO => DO,                     -- Output data, width defined by DATA_WIDTH parameter
```

```
   EMPTY => EMPTY,               -- 1-bit output empty
   FULL => FULL,                 -- 1-bit output full
   RDCOUNT => RDCOUNT,           -- Output read count, width determined by FIFO depth
   RDERR => RDERR,               -- 1-bit output read error
   WRCOUNT => WRCOUNT,           -- Output write count, width determined by FIFO depth
   WRERR => WRERR,               -- 1-bit output write error
   CLK => CLK,                   -- 1-bit input clock
   DI => DI,                     -- Input data, width defined by DATA_WIDTH parameter
   RDEN => RDEN,                 -- 1-bit input read enable
   RST => RST,                   -- 1-bit input reset
   WREN => WREN                  -- 1-bit input write enable
);
-- End of FIFO_SYNC_MACRO_inst instantiation
```

Verilog Instantiation Template

```
// FIFO_SYNC_MACRO: Synchronous First-In, First-Out (FIFO) RAM Buffer
//                  7 Series
// Xilinx HDL Language Template, version 2020.1

////////////////////////////////////////////////////////////////////
// DATA_WIDTH | FIFO_SIZE | FIFO Depth | RDCOUNT/WRCOUNT Width //
// ===========|===========|============|=======================//
//   37-72    |  "36Kb"   |     512    |         9-bit         //
//   19-36    |  "36Kb"   |    1024    |        10-bit         //
//   19-36    |  "18Kb"   |     512    |         9-bit         //
//   10-18    |  "36Kb"   |    2048    |        11-bit         //
//   10-18    |  "18Kb"   |    1024    |        10-bit         //
//    5-9     |  "36Kb"   |    4096    |        12-bit         //
//    5-9     |  "18Kb"   |    2048    |        11-bit         //
//    1-4     |  "36Kb"   |    8192    |        13-bit         //
//    1-4     |  "18Kb"   |    4096    |        12-bit         //
////////////////////////////////////////////////////////////////////

FIFO_SYNC_MACRO  #(
   .DEVICE("7SERIES"), // Target Device: "7SERIES"
   .ALMOST_EMPTY_OFFSET(9'h080), // Sets the almost empty threshold
   .ALMOST_FULL_OFFSET(9'h080),  // Sets almost full threshold
   .DATA_WIDTH(0), // Valid values are 1-72 (37-72 only valid when FIFO_SIZE="36Kb")
   .DO_REG(0),     // Optional output register (0 or 1)
   .FIFO_SIZE ("18Kb")  // Target BRAM: "18Kb" or "36Kb"
) FIFO_SYNC_MACRO_inst (
   .ALMOSTEMPTY(ALMOSTEMPTY), // 1-bit output almost empty
   .ALMOSTFULL(ALMOSTFULL),   // 1-bit output almost full
   .DO(DO),                   // Output data, width defined by DATA_WIDTH parameter
   .EMPTY(EMPTY),             // 1-bit output empty
   .FULL(FULL),               // 1-bit output full
   .RDCOUNT(RDCOUNT),         // Output read count, width determined by FIFO depth
   .RDERR(RDERR),             // 1-bit output read error
   .WRCOUNT(WRCOUNT),         // Output write count, width determined by FIFO depth
   .WRERR(WRERR),             // 1-bit output write error
   .CLK(CLK),                 // 1-bit input clock
   .DI(DI),                   // Input data, width defined by DATA_WIDTH parameter
   .RDEN(RDEN),               // 1-bit input read enable
   .RST(RST),                 // 1-bit input reset
   .WREN(WREN)                // 1-bit input write enable
 );

// End of FIFO_SYNC_MACRO_inst instantiation
```

Functional Categories

This section categorizes, by function, the circuit design elements described in detail later in this guide. The elements (primitives and macros) are listed in alphanumeric order under each functional category.

Advanced

Design Element	Description
XADC	Primitive: Dual 12-Bit 1MSPS Analog-to-Digital Converter

Arithmetic Functions

Design Element	Description
DSP48E1	Primitive: 48-bit Multi-Functional Arithmetic Block

Clock Components

Design Element	Description
BUFG	Primitive: Global Clock Simple Buffer
BUFGCE	Primitive: Global Clock Buffer with Clock Enable
BUFGCE_1	Primitive: Global Clock Buffer with Clock Enable and Output State 1
BUFGCTRL	Primitive: Global Clock Control Buffer
BUFGMUX	Primitive: Global Clock Mux Buffer
BUFGMUX_1	Primitive: Global Clock Mux Buffer with Output State 1
BUFGMUX_CTRL	Primitive: 2-to-1 Global Clock MUX Buffer
BUFH	Primitive: HROW Clock Buffer for a Single Clocking Region
BUFHCE	Primitive: HROW Clock Buffer for a Single Clocking Region with Clock Enable
BUFIO	Primitive: Local Clock Buffer for I/O
BUFMR	Primitive: Multi-Region Clock Buffer
BUFMRCE	Primitive: Multi-Region Clock Buffer with Clock Enable
BUFR	Primitive: Regional Clock Buffer for I/O and Logic Resources within a Clock Region
MMCME2_ADV	Primitive: Advanced Mixed Mode Clock Manager

Design Element	Description
MMCME2_BASE	Primitive: Base Mixed Mode Clock Manager
PLLE2_ADV	Primitive: Advanced Phase Locked Loop (PLL)
PLLE2_BASE	Primitive: Base Phase Locked Loop (PLL)

Config/BSCAN Components

Design Element	Description
BSCANE2	Primitive: Boundary-Scan User Instruction
CAPTUREE2	Primitive: Register Capture
DNA_PORT	Primitive: Device DNA Access Port
EFUSE_USR	Primitive: 32-bit non-volatile design ID
FRAME_ECCE2	Primitive: Configuration Frame Error Correction
ICAPE2	Primitive: Internal Configuration Access Port
STARTUPE2	Primitive: STARTUP Block
USR_ACCESSE2	Primitive: Configuration Data Access

I/O Components

Design Element	Description
DCIRESET	Primitive: Digitally Controlled Impedance Reset Component
IBUF	Primitive: Input Buffer
IBUF_IBUFDISABLE	Primitive: Single-ended Input Buffer with Input Disable
IBUF_INTERMDISABLE	Primitive: Single-ended Input Buffer with Input Termination Disable and Input Disable
IBUFDS	Primitive: Differential Signaling Input Buffer
IBUFDS_DIFF_OUT	Primitive: Differential Signaling Input Buffer With Differential Output
IBUFDS_DIFF_OUT _IBUFDISABLE	Primitive: Input Differential Buffer with Input Disable and Differential Output
IBUFDS_DIFF_OUT _INTERMDISABLE	Primitive: Input Differential Buffer with Input Termination Disable, Input Disable, and Differential Output
IBUFDS_IBUFDISABLE	Primitive: Input Differential Buffer with Input Path Disable
IBUFDS_INTERMDISABLE	Primitive: Input Differential Buffer with Input Termination Disable and Input Disable
IBUFDS_GTE2	Primitive: Gigabit Transceiver Buffer
IDELAYCTRL	Primitive: IDELAYE2/ODELAYE2 Tap Delay Value Control
IDELAYE2	Primitive: Input Fixed or Variable Delay Element
IN_FIFO	Primitive: Input First-In, First-Out (FIFO)
IOBUF	Primitive: Bi-Directional Buffer
IOBUF_DCIEN	Primitive: Bi-Directional Single-ended Buffer with DCI and Input Disable.
IOBUF_INTERMDISABLE	Primitive: Bi-Directional Single-ended Buffer with Input Termination Disable and Input Path Disable
IOBUFDS	Primitive: 3-State Differential Signaling I/O Buffer with Active-Low Output Enable
IOBUFDS_DCIEN	Primitive: Bi-Directional Differential Buffer with DCI Enable/Disable and Input Disable

Design Element	Description
IOBUFDS_DIFF_OUT	Primitive: Differential Bi-directional Buffer with Differential Output
IOBUFDS_DIFF_OUT_DCIEN	Primitive: Bi-Directional Differential Buffer with DCI Disable, Input Disable, and Differential Output
IOBUFDS_DIFF_OUT _INTERMDISABLE	Primitive: Bi-Directional Differential Buffer with Input Termination Disable, Input Disable, and Differential Output
IOBUFDS_INTERMDISABLE	Primitive: Bi-Directional Differential Buffer with Input Termination Disable and Input Disable
ISERDESE2	Primitive: Input SERial/DESerializer with Bitslip
KEEPER	Primitive: KEEPER Symbol
OBUF	Primitive: Output Buffer
OBUFDS	Primitive: Differential Signaling Output Buffer
OBUFT	Primitive: 3-State Output Buffer with Active-Low Output Enable
OBUFTDS	Primitive: 3-State Output Buffer with Differential Signaling, Active-Low Output Enable
ODELAYE2	Primitive: Output Fixed or Variable Delay Element
OSERDESE2	Primitive: Output SERial/DESerializer with bitslip
OUT_FIFO	Primitive: Output First-In, First-Out (FIFO) Buffer
PULLDOWN	Primitive: Resistor to GND for Input Pads, Open-Drain, and 3-State Outputs
PULLUP	Primitive: Resistor to VCC for Input PADs, Open-Drain, and 3-State Outputs

RAM/ROM

Design Element	Description
FIFO18E1	Primitive: 18Kb FIFO (First-In-First-Out) Block RAM Memory
FIFO36E1	Primitive: 36Kb FIFO (First-In-First-Out) Block RAM Memory
RAM128X1D	Primitive: 128-Deep by 1-Wide Dual Port Random Access Memory (Select RAM)
RAM128X1S	Primitive: 128-Deep by 1-Wide Random Access Memory (Select RAM)
RAM256X1S	Primitive: 256-Deep by 1-Wide Random Access Memory (Select RAM)
RAM32M	Primitive: 32-Deep by 8-bit Wide Multi Port Random Access Memory (Select RAM)
RAM32X1D	Primitive: 32-Deep by 1-Wide Static Dual Port Synchronous RAM
RAM32X1S	Primitive: 32-Deep by 1-Wide Static Synchronous RAM
RAM32X1S_1	Primitive: 32-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock
RAM32X2S	Primitive: 32-Deep by 2-Wide Static Synchronous RAM
RAM64M	Primitive: 64-Deep by 4-bit Wide Multi Port Random Access Memory (Select RAM)
RAM64X1D	Primitive: 64-Deep by 1-Wide Dual Port Static Synchronous RAM
RAM64X1S	Primitive: 64-Deep by 1-Wide Static Synchronous RAM
RAM64X1S_1	Primitive: 64-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock
RAMB18E1	Primitive: 18K-bit Configurable Synchronous Block RAM
RAMB36E1	Primitive: 36K-bit Configurable Synchronous Block RAM
ROM128X1	Primitive: 128-Deep by 1-Wide ROM
ROM256X1	Primitive: 256-Deep by 1-Wide ROM
ROM32X1	Primitive: 32-Deep by 1-Wide ROM

Design Element	Description
ROM64X1	Primitive: 64-Deep by 1-Wide ROM

Registers/Latches

Design Element	Description
FDCE	Primitive: D Flip-Flop with Clock Enable and Asynchronous Clear
FDPE	Primitive: D Flip-Flop with Clock Enable and Asynchronous Preset
FDRE	Primitive: D Flip-Flop with Clock Enable and Synchronous Reset
FDSE	Primitive: D Flip-Flop with Clock Enable and Synchronous Set
IDDR	Primitive: Input Double Data-Rate Register
IDDR_2CLK	Primitive: Input Double Data-Rate Register with Dual Clock Inputs
LDCE	Primitive: Transparent Data Latch with Asynchronous Clear and Gate Enable
LDPE	Primitive: Transparent Data Latch with Asynchronous Preset and Gate Enable
ODDR	Primitive: Dedicated Double Data Rate (DDR) Output Register

Slice/CLB Primitives

Design Element	Description
CARRY4	Primitive: Fast Carry Logic with Look Ahead
CFGLUT5	Primitive: 5-input Dynamically Reconfigurable Look-Up Table (LUT)
LUT1	Primitive: 1-Bit Look-Up Table with General Output
LUT2	Primitive: 2-Bit Look-Up Table with General Output
LUT3	Primitive: 3-Bit Look-Up Table with General Output
LUT4	Primitive: 4-Bit Look-Up-Table with General Output
LUT5	Primitive: 5-Input Lookup Table with General Output
LUT6	Primitive: 6-Input Lookup Table with General Output
LUT6_2	Primitive: Six-input, 2-output, Look-Up Table
MUXF7	Primitive: 2-to-1 Look-Up Table Multiplexer with General Output
MUXF8	Primitive: 2-to-1 Look-Up Table Multiplexer with General Output
SRL16E	Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Clock Enable
SRLC32E	Primitive: 32 Clock Cycle, Variable Length Shift Register Look-Up Table (LUT) with Clock Enable

Design Elements

About Design Elements

This section describes the design elements that can be used with 7 series FPGAs and Zynq®-7000 SoC devices devices. The design elements are organized alphabetically.

The following information is provided for each design element, where applicable:

- Name of element
- Brief description
- Schematic symbol (if any)
- Logic table (if any)
- Port descriptions
- Design Entry Method
- Available attributes (if any)
- Example instantiation templates
- For more information

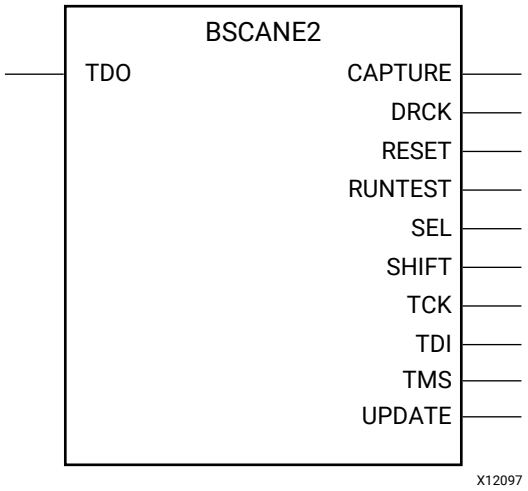
Instantiation Templates

Instantiation templates for library elements are also available in Vivado, as well as in a downloadable ZIP file. Because PDF includes headers and footers if you copy text that spans pages, you should copy templates from Vivado or the downloaded ZIP file whenever possible.

Instantiation templates can be found on the Web in the Instantiation Templates for 7 Series Devices file.

BSCANE2

Primitive: Boundary-Scan User Instruction



Introduction

This design element allows access to and from internal logic by the JTAG Boundary Scan logic controller. This allows for communication between the internal running design and the dedicated JTAG pins of the FPGA. Each instance of this design element will handle one JTAG USER instruction (USER1 through USER4) as set with the JTAG_CHAIN attribute.

To handle all four USER instructions, instantiate four of these elements, and set the JTAG_CHAIN attribute appropriately.

For specific information on boundary scan for an architecture, see the Configuration User Guide for the specific device.

Port Descriptions

Port	Type	Width	Function
CAPTURE	Output	1	CAPTURE output from TAP controller.
DRCK	Output	1	Gated TCK output. When SEL is asserted, DRCK toggles when CAPTURE or SHIFT are asserted.
RESET	Output	1	Reset output for TAP controller.
RUNTEST	Output	1	Output asserted when TAP controller is in Run Test/Idle state.
SEL	Output	1	USER instruction active output.
SHIFT	Output	1	SHIFT output from TAP controller.
TCK	Output	1	Test Clock output. Fabric connection to TAP Clock pin.
TDI	Output	1	Test Data Input (TDI) output from TAP controller.

Port	Type	Width	Function
TDO	Input	1	Test Data Output (TDO) input for USER function.
TMS	Output	1	Test Mode Select output. Fabric connection to TAP.
UPDATE	Output	1	UPDATE output from TAP controller.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
JTAG_CHAIN	DECIMAL	1, 2, 3, 4	1	Value for USER command.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BSCANE2: Boundary-Scan User Instruction
--          7 Series
-- Xilinx HDL Language Template, version 2020.1

BSCANE2_inst : BSCANE2
generic map (
   JTAG_CHAIN => 1  -- Value for USER command.
)
port map (
   CAPTURE => CAPTURE, -- 1-bit output: CAPTURE output from TAP controller.
   DRCK => DRCK,       -- 1-bit output: Gated TCK output. When SEL is asserted, DRCK toggles when CAPTURE or
                       -- SHIFT are asserted.

   RESET => RESET,     -- 1-bit output: Reset output for TAP controller.
   RUNTEST => RUNTEST, -- 1-bit output: Output asserted when TAP controller is in Run Test/Idle state.
   SEL => SEL,         -- 1-bit output: USER instruction active output.
   SHIFT => SHIFT,     -- 1-bit output: SHIFT output from TAP controller.
   TCK => TCK,         -- 1-bit output: Test Clock output. Fabric connection to TAP Clock pin.
   TDI => TDI,         -- 1-bit output: Test Data Input (TDI) output from TAP controller.
   TMS => TMS,         -- 1-bit output: Test Mode Select output. Fabric connection to TAP.
   UPDATE => UPDATE,   -- 1-bit output: UPDATE output from TAP controller
   TDO => TDO          -- 1-bit input: Test Data Output (TDO) input for USER function.
);

-- End of BSCANE2_inst instantiation
```

Verilog Instantiation Template

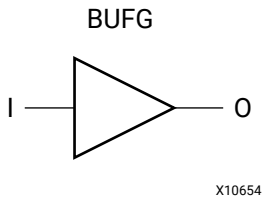
```
// BSCANE2: Boundary-Scan User Instruction
//          7 Series
// Xilinx HDL Language Template, version 2020.1

BSCANE2 #(
   .JTAG_CHAIN(1)  // Value for USER command.
)
BSCANE2_inst (
   .CAPTURE(CAPTURE), // 1-bit output: CAPTURE output from TAP controller.
   .DRCK(DRCK),       // 1-bit output: Gated TCK output. When SEL is asserted, DRCK toggles when CAPTURE or
                      // SHIFT are asserted.

   .RESET(RESET),     // 1-bit output: Reset output for TAP controller.
   .RUNTEST(RUNTEST), // 1-bit output: Output asserted when TAP controller is in Run Test/Idle state.
   .SEL(SEL),         // 1-bit output: USER instruction active output.
   .SHIFT(SHIFT),     // 1-bit output: SHIFT output from TAP controller.
   .TCK(TCK),         // 1-bit output: Test Clock output. Fabric connection to TAP Clock pin.
   .TDI(TDI),         // 1-bit output: Test Data Input (TDI) output from TAP controller.
   .TMS(TMS),         // 1-bit output: Test Mode Select output. Fabric connection to TAP.
   .UPDATE(UPDATE),   // 1-bit output: UPDATE output from TAP controller
   .TDO(TDO)          // 1-bit input: Test Data Output (TDO) input for USER function.
);

// End of BSCANE2_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configuration Guide* (UG470).

BUFG

Primitive: Global Clock Simple Buffer



Introduction

This design element is a high-fanout buffer that connects signals to the global routing resources for low skew distribution of the signal. BUFGs are typically used on clock nets as well other high fanout nets like sets/resets and clock enables.

Port Descriptions

Port	Direction	Width	Function
I	Input	1	Clock input.
O	Output	1	Clock output.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFG: Global Clock Simple Buffer
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

BUFG_inst : BUFG
port map (
```

```
   O => O, -- 1-bit output: Clock output
   I => I  -- 1-bit input: Clock input
);

-- End of BUFG_inst instantiation
```

Verilog Instantiation Template

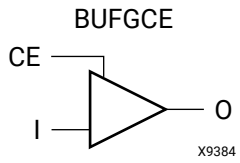
```
// BUFG: Global Clock Simple Buffer
//       7 Series
// Xilinx HDL Language Template, version 2020.1

BUFG BUFG_inst (
   .O(O), // 1-bit output: Clock output
   .I(I)  // 1-bit input: Clock input
);

// End of BUFG_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

BUFGCE

Primitive: Global Clock Buffer with Clock Enable



Introduction

This design element is a global clock buffer with a single gated input. Its O output is "0" when clock enable (CE) is Low (inactive). When clock enable (CE) is High, the I input is transferred to the O output.

Logic Table

Inputs		Outputs
I	**CE**	**O**
X	0	0
I	1	I

Port Descriptions

Port	Direction	Width	Function
CE	Input	1	Clock buffer active-High enable.
I	Input	1	Clock input.
O	Output	1	Clock output.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFGCE: Global Clock Buffer with Clock Enable
--         7 Series
-- Xilinx HDL Language Template, version 2020.1

BUFGCE_inst : BUFGCE
port map (
   O => O,   -- 1-bit output: Clock output
   CE => CE, -- 1-bit input: Clock enable input for I0
   I => I    -- 1-bit input: Primary clock
);

-- End of BUFGCE_inst instantiation
```

Verilog Instantiation Template

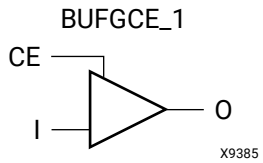
```
// BUFGCE: Global Clock Buffer with Clock Enable
//         7 Series
// Xilinx HDL Language Template, version 2020.1

BUFGCE BUFGCE_inst (
   .O(O),   // 1-bit output: Clock output
   .CE(CE), // 1-bit input: Clock enable input for I0
   .I(I)    // 1-bit input: Primary clock
);

// End of BUFGCE_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

BUFGCE_1

Primitive: Global Clock Buffer with Clock Enable and Output State 1



Introduction

This design element is a global clock buffer with a single gated input. Its O output is "1" when clock enable (CE) is Low (inactive). When clock enable (CE) is High, the I input is transferred to the O output.

Logic Table

Inputs		Outputs
I	**CE**	**O**
X	0	1
I	1	I

Port Descriptions

Port	Direction	Width	Function
CE	Input	1	Clock buffer active-High enable.
I	Input	1	Clock input.
O	Output	1	Clock output.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFGCE_1: Global Clock Buffer with Clock Enable and Output State 1
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

BUFGCE_1_inst : BUFGCE_1
port map (
   O => O,   -- 1-bit output: Clock output
   CE => CE, -- 1-bit input: Clock enable input for I0
   I => I    -- 1-bit input: Primary clock
);

-- End of BUFGCE_1_inst instantiation
```

Verilog Instantiation Template

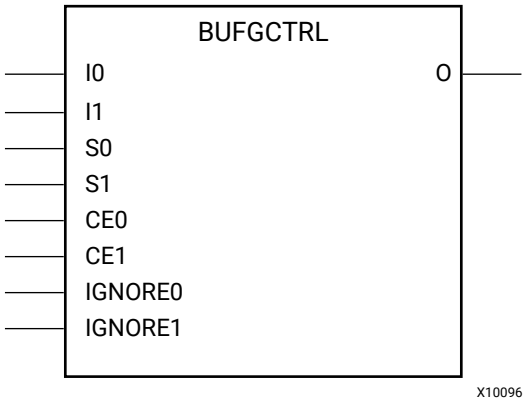
```
// BUFGCE_1: Global Clock Buffer with Clock Enable and Output State 1
//           7 Series
// Xilinx HDL Language Template, version 2020.1

BUFGCE_1 BUFGCE_1_inst (
   .O(O),   // 1-bit output: Clock output
   .CE(CE), // 1-bit input: Clock enable input for I0
   .I(I)    // 1-bit input: Primary clock
);

// End of BUFGCE_1_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

BUFGCTRL

Primitive: Global Clock Control Buffer



Introduction

BUFGCTRL primitive is a 7 series global clock buffer that is designed as a synchronous/asynchronous "glitch free" 2:1 multiplexer with two clock inputs. Unlike global clock buffers that are found in previous generations of FPGAs, these clock buffers are designed with more control pins to provide a wider range of functionality and more robust input switching. BUFGCTRL is not limited to clocking applications.

Port Descriptions

Port	Direction	Width	Function
CE0	Input	1	Clock enable input for the I0 clock input. A setup/hold time must be guaranteed when you are using the CE0 pin to enable this input. Failure to meet this requirement could result in a clock glitch.
CE1	Input	1	Clock enable input for the I1 clock input. A setup/hold time must be guaranteed when you are using the CE1 pin to enable this input. Failure to meet this requirement could result in a clock glitch.
IGNORE0	Input	1	Clock ignore input for I0 input. Asserting the IGNORE pin will bypass the BUFGCTRL from detecting the conditions for switching between two clock inputs. In other words, asserting IGNORE causes the MUX to switch the inputs at the instant the select pin changes. IGNORE0 causes the output to switch away from the I0 input immediately when the select pin changes, while IGNORE1 causes the output to switch away from the I1 input immediately when the select pin changes.

Port	Direction	Width	Function
IGNORE1	Input	1	Clock ignore input for I1 input. Asserting the IGNORE pin will bypass the BUFGCTRL from detecting the conditions for switching between two clock inputs. In other words, asserting IGNORE causes the MUX to switch the inputs at the instant the select pin changes. IGNORE0 causes the output to switch away from the I0 input immediately when the select pin changes, while IGNORE1 causes the output to switch away from the I1 input immediately when the select pin changes.
I0	Input	1	Primary clock input into the BUFGCTRL enabled by the CE0 input and selected by the S0 input.
I1	Input	1	Secondary clock input into the BUFGCTRL enabled by the CE1 input and selected by the S1 input.
O	Output	1	Clock output
S0	Input	1	Clock select input for I0. The S pins represent the clock select pin for each clock input. When using the S pin as input select, there is a setup/hold time requirement. Unlike CE pins, failure to meet this requirement will not result in a clock glitch. However, it can cause the output clock to appear one clock cycle later.
S1	Input	1	Clock select input for I1. The S pins represent the clock select pin for each clock input. When using the S pin as input select, there is a setup/hold time requirement. Unlike CE pins, failure to meet this requirement will not result in a clock glitch. However, it can cause the output clock to appear one clock cycle later.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT_OUT	DECIMAL	0, 1	0	Initializes the BUFGCTRL output to the specified value after configuration.
PRESELECT_I0	BOOLEAN	FALSE, TRUE	FALSE	If TRUE, BUFGCTRL output uses I0 input after configuration.
PRESELECT_I1	BOOLEAN	FALSE, TRUE	FALSE	If TRUE, BUFGCTRL output uses I1 input after configuration.

***Note*:** Both PRESELECT attributes might not be TRUE at the same time.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFGCTRL: Global Clock Control Buffer
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

BUFGCTRL_inst : BUFGCTRL
generic map (
   INIT_OUT => 0,         -- Initial value of BUFGCTRL output ($VALUES;)
   PRESELECT_I0 => FALSE, -- BUFGCTRL output uses I0 input ($VALUES;)
   PRESELECT_I1 => FALSE  -- BUFGCTRL output uses I1 input ($VALUES;)
)
port map (
   O => O,              -- 1-bit output: Clock output
   CE0 => CE0,          -- 1-bit input: Clock enable input for I0
   CE1 => CE1,          -- 1-bit input: Clock enable input for I1
   I0 => I0,            -- 1-bit input: Primary clock
   I1 => I1,            -- 1-bit input: Secondary clock
   IGNORE0 => IGNORE0,  -- 1-bit input: Clock ignore input for I0
   IGNORE1 => IGNORE1,  -- 1-bit input: Clock ignore input for I1
   S0 => S0,            -- 1-bit input: Clock select for I0
   S1 => S1             -- 1-bit input: Clock select for I1
);

-- End of BUFGCTRL_inst instantiation
```

Verilog Instantiation Template

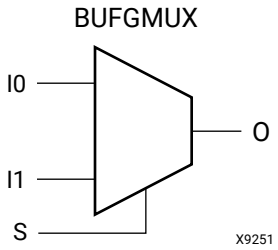
```
// BUFGCTRL: Global Clock Control Buffer
//           7 Series
// Xilinx HDL Language Template, version 2020.1

BUFGCTRL #(
   .INIT_OUT(0),           // Initial value of BUFGCTRL output ($VALUES;)
   .PRESELECT_I0("FALSE"), // BUFGCTRL output uses I0 input ($VALUES;)
   .PRESELECT_I1("FALSE")  // BUFGCTRL output uses I1 input ($VALUES;)
)
BUFGCTRL_inst (
   .O(O),              // 1-bit output: Clock output
   .CE0(CE0),          // 1-bit input: Clock enable input for I0
   .CE1(CE1),          // 1-bit input: Clock enable input for I1
   .I0(I0),            // 1-bit input: Primary clock
   .I1(I1),            // 1-bit input: Secondary clock
   .IGNORE0(IGNORE0), // 1-bit input: Clock ignore input for I0
   .IGNORE1(IGNORE1), // 1-bit input: Clock ignore input for I1
   .S0(S0),            // 1-bit input: Clock select for I0
   .S1(S1)             // 1-bit input: Clock select for I1
);

// End of BUFGCTRL_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

BUFGMUX

Primitive: Global Clock Mux Buffer



Introduction

This design element is a global clock buffer, based on BUFGCTRL, that can select between two input clocks: I0 and I1. When the select input (S) is Low, the signal on I0 is selected for output (O). When the select input (S) is High, the signal on I1 is selected for output. BUFGMUX and BUFGMUX_1 are distinguished by the state the output assumes when it switches between clocks in response to a change in the select input. BUFGMUX assumes output state 0 and BUFGMUX_1 assumes output state 1.

Logic Table

Inputs			Outputs
I0	**I1**	**S**	**O**
I0	X	0	I0
X	I1	1	I1
X	X	↑	0
X	X	↓	0

Port Descriptions

Port	Direction	Width	Function
I0	Input	1	Clock buffer input. This input is reflected on the output O when the S input is zero.
I1	Input	1	Clock buffer input. This input is reflected on the output O when the S input is one.
O	Output	1	Clock buffer output.
S	Input	1	Clock buffer select input. Selects the I0 input when Low and the I1 input when High.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFGMUX: Global Clock Mux Buffer
--          7 Series
-- Xilinx HDL Language Template, version 2020.1

BUFGMUX_inst : BUFGMUX
port map (
   O => O,   -- 1-bit output: Clock output
   I0 => I0, -- 1-bit input: Clock input (S=0)
   I1 => I1, -- 1-bit input: Clock input (S=1)
   S => S    -- 1-bit input: Clock select
);

-- End of BUFGMUX_inst instantiation
```

Verilog Instantiation Template

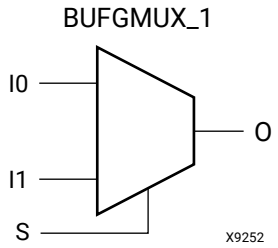
```
// BUFGMUX: Global Clock Mux Buffer
//          7 Series
// Xilinx HDL Language Template, version 2020.1

BUFGMUX #(
)
BUFGMUX_inst (
   .O(O),   // 1-bit output: Clock output
   .I0(I0), // 1-bit input: Clock input (S=0)
   .I1(I1), // 1-bit input: Clock input (S=1)
   .S(S)    // 1-bit input: Clock select
);

// End of BUFGMUX_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

BUFGMUX_1

Primitive: Global Clock Mux Buffer with Output State 1



Introduction

This design element is a global clock buffer, based on BUFGCTRL, that can select between two input clocks: I0 and I1. When the select input (S) is Low, the signal on I0 is selected for output (O). When the select input (S) is High, the signal on I1 is selected for output. BUFGMUX and BUFGMUX_1 are distinguished by the state the output assumes when it switches between clocks in response to a change in the select input. BUFGMUX assumes output state 0 and BUFGMUX_1 assumes output state 1.

Logic Table

Inputs			Outputs
I0	**I1**	**S**	**O**
I0	X	0	I0
X	I1	1	I1
X	X	↑	1
X	X	↓	1

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFGMUX_1: Global Clock Mux Buffer with Output State 1
--            7 Series
-- Xilinx HDL Language Template, version 2020.1

BUFGMUX_1_inst : BUFGMUX_1
port map (
   O => O,   -- 1-bit output: Clock output
   I0 => I0, -- 1-bit input: Clock input (S=0)
   I1 => I1, -- 1-bit input: Clock input (S=1)
   S => S    -- 1-bit input: Clock select
);

-- End of BUFGMUX_1_inst instantiation
```

Verilog Instantiation Template

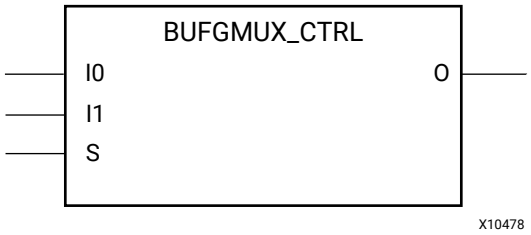
```
// BUFGMUX_1: Global Clock Mux Buffer with Output State 1
//            7 Series
// Xilinx HDL Language Template, version 2020.1

BUFGMUX_1 #(
)
BUFGMUX_1_inst (
   .O(O),   // 1-bit output: Clock output
   .I0(I0), // 1-bit input: Clock input (S=0)
   .I1(I1), // 1-bit input: Clock input (S=1)
   .S(S)    // 1-bit input: Clock select
);

// End of BUFGMUX_1_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

BUFGMUX_CTRL

Primitive: 2-to-1 Global Clock MUX Buffer



Introduction

This design element is a global clock buffer with two clock inputs, one clock output, and a select line used to cleanly select between one of two clocks driving the global clocking resource. This component is based on BUFGCTRL, with some pins connected to logic High or Low. This element uses the S pin as the select pin for the 2-to-1 MUX. S can switch anytime without causing a glitch on the output clock of the buffer.

Port Descriptions

Port	Direction	Width	Function
I0	Input	1	Clock buffer input. This input is reflected on the output O when the S input is zero.
I1	Input	1	Clock buffer input. This input is reflected on the output O when the S input is one.
O	Output	1	Clock buffer output.
S	Input	1	Clock buffer select input. When low, selects I0 input and when high, the I1 input is selected.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFGMUX_CTRL: 2-to-1 Global Clock MUX Buffer
--               7 Series
-- Xilinx HDL Language Template, version 2020.1

BUFGMUX_CTRL_inst : BUFGMUX_CTRL
port map (
   O => O,   -- 1-bit output: Clock output
   I0 => I0, -- 1-bit input: Clock input (S=0)
   I1 => I1, -- 1-bit input: Clock input (S=1)
   S => S    -- 1-bit input: Clock select
);

-- End of BUFGMUX_CTRL_inst instantiation
```

Verilog Instantiation Template

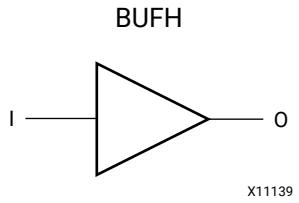
```
// BUFGMUX_CTRL: 2-to-1 Global Clock MUX Buffer
//               7 Series
// Xilinx HDL Language Template, version 2020.1

BUFGMUX_CTRL BUFGMUX_CTRL_inst (
   .O(O),   // 1-bit output: Clock output
   .I0(I0), // 1-bit input: Clock input (S=0)
   .I1(I1), // 1-bit input: Clock input (S=1)
   .S(S)    // 1-bit input: Clock select
);

// End of BUFGMUX_CTRL_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

BUFH

Primitive: HROW Clock Buffer for a Single Clocking Region



Introduction

The BUFH primitive allows direct access to the clock region entry point of the global buffer (BUFG) resource. This allows access to unused portions of the global clocking network to be used as high-speed, low skew local (single clock region) routing resources.

Port Descriptions

Port	Direction	Width	Function
I	Input	1	Clock input.
O	Output	1	Clock output.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFH: HROW Clock Buffer for a Single Clocking Region
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

BUFH_inst : BUFH
port map (
```

```
   O => O, -- 1-bit output: Clock output
   I => I  -- 1-bit input: Clock input
);

-- End of BUFH_inst instantiation
```

Verilog Instantiation Template

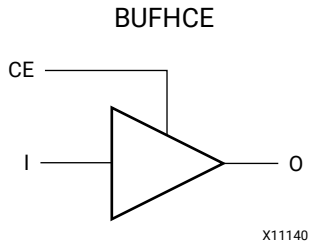
```
// BUFH: HROW Clock Buffer for a Single Clocking Region
//       7 Series
// Xilinx HDL Language Template, version 2020.1

BUFH BUFH_inst (
   .O(O), // 1-bit output: Clock output
   .I(I)  // 1-bit input: Clock input
);

// End of BUFH_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

BUFHCE

Primitive: HROW Clock Buffer for a Single Clocking Region with Clock Enable



Introduction

The BUFHCE primitive allows direct access to the clock region entry point of the global buffer (BUFG) resource. This allows access to unused portions of the global clocking network to be used as high-speed, low skew local (single clock region) routing resources. Additionally, the clock enable input (CE) allows for finer-grained control of clock enabling or gating to allow for power reduction for circuitry or portions of the design not constantly used.

Port Descriptions

Port	Direction	Width	Function
CE	Input	1	Enables propagation of signal from I to O. When low, performs a glitchless transition of the output to INIT_OUT value.
I	Input	1	Clock input.
O	Output	1	Clock output.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
CE_TYPE	STRING	"SYNC", "ASYNC"	"SYNC"	Sets clock enable behavior where "SYNC" allows for a glitchless transition to and from the INIT_OUT value. "ASYNC" is generally used to create a more immediate transition such as when you can expect the clock to be stopped or when using the BUFHCE for a high fanout control or data path routing instead of a clock buffer.
INIT_OUT	DECIMAL	0, 1	0	Initial output value, also indicates stop low vs. stop high behavior.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFHCE: HROW Clock Buffer for a Single Clocking Region with Clock Enable
--         7 Series
-- Xilinx HDL Language Template, version 2020.1

BUFHCE_inst : BUFHCE
generic map (
   CE_TYPE => "SYNC", -- "SYNC" (glitchless switching) or "ASYNC" (immediate switch)
   INIT_OUT => 0      -- Initial output value (0-1)
)
port map (
   O => O,   -- 1-bit output: Clock output
   CE => CE, -- 1-bit input: Active high enable
   I => I    -- 1-bit input: Clock input
);

-- End of BUFHCE_inst instantiation
```

Verilog Instantiation Template

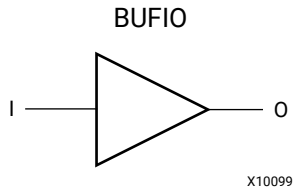
```
// BUFHCE: HROW Clock Buffer for a Single Clocking Region with Clock Enable
//         7 Series
// Xilinx HDL Language Template, version 2020.1

BUFHCE #(
   .CE_TYPE("SYNC"), // "SYNC" (glitchless switching) or "ASYNC" (immediate switch)
   .INIT_OUT(0)      // Initial output value (0-1)
)
BUFHCE_inst (
   .O(O),   // 1-bit output: Clock output
   .CE(CE), // 1-bit input: Active high enable
   .I(I)    // 1-bit input: Clock input
);

// End of BUFHCE_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

BUFIO

Primitive: Local Clock Buffer for I/O



Introduction

This design element is a local clock-in, clock-out buffer. It drives a dedicated clock net within the I/O column, independent of the global clock resources and is ideally suited for source-synchronous data capture (forwarded/receiver clock distribution). BUFIO elements can be driven by a dedicated MRCC I/O located in the same clock region, or a BUFMRCE/BUFMR component capable of clocking multiple clock regions. BUFIO can only drive I/O components within the bank in which they exist. They cannot directly drive logic resources (CLB, block RAM, etc.) because the I/O clock network only reaches the I/O column.

Port Descriptions

Port	Direction	Width	Function
I	Input	1	Input port to clock buffer. Connect this to an IBUF connected to a top-level port or an associated BUFMR buffer.
O	Output	1	Output port from clock buffer. Connect this to the clock inputs to synchronous I/O components like the ISERDESE2, OSERDESE2, IDDR, ODDR or register connected directly to an I/O port (inferred or instantiated).

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFIO: Local Clock Buffer for I/O
--        7 Series
-- Xilinx HDL Language Template, version 2020.1

BUFIO_inst : BUFIO
port map (
   O => O, -- 1-bit output: Clock output (connect to I/O clock loads).
   I => I  -- 1-bit input: Clock input (connect to an IBUF or BUFMR).
);

-- End of BUFIO_inst instantiation
```

Verilog Instantiation Template

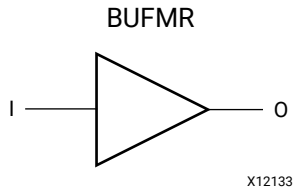
```
// BUFIO: Local Clock Buffer for I/O
//        7 Series
// Xilinx HDL Language Template, version 2020.1

BUFIO BUFIO_inst (
   .O(O), // 1-bit output: Clock output (connect to I/O clock loads).
   .I(I)  // 1-bit input: Clock input (connect to an IBUF or BUFMR).
);

// End of BUFIO_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

BUFMR

Primitive: Multi-Region Clock Buffer



Introduction

The BUFMR is a multi-region clock-in/clock-out buffer. The BUFMR replaces the multi-region/bank support of the BUFR and BUFIO available in prior Virtex architectures. There are two BUFMRs in every bank and each buffer can be driven by one specific MRCC in the same bank. The BUFMRs drive the BUFIOs and/or BUFRs in the same region/banks and in the region above and below via the I/O clocking backbone. Do not use a BUFMR when driving BUFRs using clock dividers (not in bypass), but instead use a BUFMRCE component.

Port Descriptions

Port	Direction	Width	Function
I	Input	1	BUFMR clock input pin. Connect to an IBUF input that in turn is directly connected to a MRCC I/O port.
O	Output	1	BUFMR clock output pin. Connect to BUFIOs and/or BUFRs to be driven in adjacent regions.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFMR: Multi-Region Clock Buffer
--        7 Series
-- Xilinx HDL Language Template, version 2020.1

BUFMR_inst : BUFMR
port map (
   O => O, -- 1-bit output: Clock output (connect to BUFIOs/BUFRs)
   I => I  -- 1-bit input: Clock input (Connect to IBUF)
);

-- End of BUFMR_inst instantiation
```

Verilog Instantiation Template

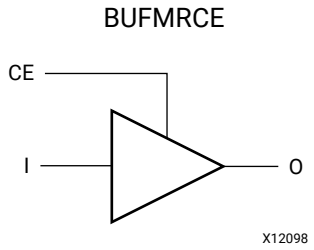
```
// BUFMR: Multi-Region Clock Buffer
//        7 Series
// Xilinx HDL Language Template, version 2020.1

BUFMR BUFMR_inst (
   .O(O), // 1-bit output: Clock output (connect to BUFIOs/BUFRs)
   .I(I)  // 1-bit input: Clock input (Connect to IBUF)
);

// End of BUFMR_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

BUFMRCE

Primitive: Multi-Region Clock Buffer with Clock Enable



Introduction

The BUFMRCE is a multi-region clock-in/clock-out buffer with clock with clock enable (CE). Deasserting CE stops the output clock to a user specified value. The BUFMRCE replaces the multi-region/bank support of the BUFR and BUFIO available in prior Virtex architectures. There are two BUFMRCEs in every bank and each buffer can be driven by one specific MRCC in the same bank. The BUFMRCE drives the BUFIOs and/or BUFRs in the same region/banks and in the region above and below via the I/O clocking backbone. When using BUFR dividers (not in bypass), the BUFMRCE must be disabled by deasserting the CE pin, the BUFR must be reset (cleared by asserting CLR), and then the CE signal should be asserted. This sequence ensures that all BUFR output clocks are phase aligned. If the dividers within the BUFRs are not used, then this additional circuitry is not necessary. If the clock enable circuitry is not needed, a BUFMR component should be used in place of a BUFMRCE.

Port Descriptions

Port	Direction	Width	Function
CE	Input	1	Active-High buffer enable input. When low, output will settle to INIT_OUT value.
I	Input	1	BUFMR clock input pin. Connect to an IBUF input that in turn is directly connected to a MRCC I/O port.
O	Output	1	BUFMR clock output pin. Connect to BUFIOs and/or BUFRs to be driven in the same and adjacent regions.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
CE_TYPE	STRING	"SYNC", "ASYNC"	"SYNC"	Set to "SYNC" for CE to be synchronous to input I and create a glitchless output. Set to "ASYNC" for stopped clock or non-clock operation of the CE signal.
INIT_OUT	DECIMAL	0, 1	0	Initial output value, also indicates stop low vs. stop high behavior

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFMRCE: Multi-Region Clock Buffer with Clock Enable
--          7 Series
-- Xilinx HDL Language Template, version 2020.1

BUFMRCE_inst : BUFMRCE
generic map (
   CE_TYPE => "SYNC", -- SYNC, ASYNC
   INIT_OUT => 0      -- Initial output and stopped polarity, (0-1)
)
port map (
   O => O,   -- 1-bit output: Clock output (connect to BUFIOs/BUFRs)
   CE => CE, -- 1-bit input: Active high buffer enable
   I => I    -- 1-bit input: Clock input (Connect to IBUF)
);

-- End of BUFMRCE_inst instantiation
```

Verilog Instantiation Template

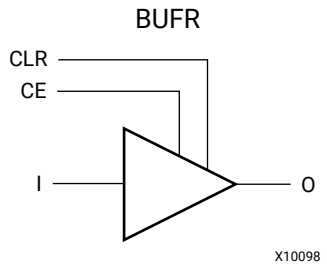
```
// BUFMRCE: Multi-Region Clock Buffer with Clock Enable
//          7 Series
// Xilinx HDL Language Template, version 2020.1

BUFMRCE #(
   .CE_TYPE("SYNC"), // SYNC, ASYNC
   .INIT_OUT(0)      // Initial output and stopped polarity, (0-1)
)
BUFMRCE_inst (
   .O(O),   // 1-bit output: Clock output (connect to BUFIOs/BUFRs)
   .CE(CE), // 1-bit input: Active high buffer enable
   .I(I)    // 1-bit input: Clock input (Connect to IBUF)
);

// End of BUFMRCE_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

BUFR

Primitive: Regional Clock Buffer for I/O and Logic Resources within a Clock Region



Introduction

The BUFR is a regional clock buffer in 7 series devices that drives clock signals to a dedicated clock net within a clock region, independent from the global clock tree. Each BUFR can drive the regional clock nets in the region in which it is located. Unlike BUFIO components, BUFR components can drive the I/O logic and logic resources (CLB, block RAM, etc.) in the existing clock region. They can be driven by the output from an IBUF, BUFMRCE, MMCM or local interconnect, and are capable of generating divided clock outputs with respect to the clock input. The divide value is an integer between one and eight. BUFR components are ideal for source-synchronous applications requiring clock domain crossing or serial-to-parallel conversion. There are two BUFR components in a typical clock region (two regional clock networks). If local clocking is needed in multiple clock regions, the BUFMRCE can drive multiple BUFR components in adjacent clock regions to further extend this clocking capability. For more details, refer to BUFMRCE.

Port Descriptions

Port	Direction	Width	Function
CE	Input	1	Clock enable port. When asserted low, this port disables the output clock. When asserted high, the clock is propagated to the output port (O). This pin cannot be used in "BYPASS" mode. Connect to vcc when BUFR_DIVIDE is set to "BYPASS" or if not used.
CLR	Input	1	Counter asynchronous clear for divided clock output. When asserted high, this port resets the counter used to produce the divided clock output and the output is asserted low. This pin cannot be used in "BYPASS" mode. Connect to gnd when BUFR_DIVIDE is set to "BYPASS" or if not used.
I	Input	1	Clock input port. This port is the clock source port for BUFR. It can be driven by an IBUF, BUFMRCE, MMCM, or local interconnect.
O	Output	1	Clock output port. This port drives the clock tracks in the clock region of the BUFR. It connects to FPGA clocked components.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed_Values	Default	Description
BUFR_DIVIDE	STRING	"BYPASS", "1", "2", "3", "4", "5", "6", "7", "8"	"BYPASS"	Specifies whether the output clock is a divided version of the input clock.
SIM_DEVICE	STRING	"7SERIES"	"7SERIES"	For correct simulation behavior, this attribute must be set to "7SERIES" when targeting a 7 series device.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFR: Regional Clock Buffer for I/O and Logic Resources within a Clock Region
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

BUFR_inst : BUFR
generic map (
   BUFR_DIVIDE => "BYPASS",   -- Values: "BYPASS, 1, 2, 3, 4, 5, 6, 7, 8"
   SIM_DEVICE => "7SERIES"  -- Must be set to "7SERIES"
)
port map (
   O => O,     -- 1-bit output: Clock output port
   CE => CE,   -- 1-bit input: Active high, clock enable (Divided modes only)
   CLR => CLR, -- 1-bit input: Active high, asynchronous clear (Divided modes only)
   I => I      -- 1-bit input: Clock buffer input driven by an IBUF, MMCM or local interconnect
);

-- End of BUFR_inst instantiation
```

Verilog Instantiation Template

```
// BUFR: Regional Clock Buffer for I/O and Logic Resources within a Clock Region
//       7 Series
// Xilinx HDL Language Template, version 2020.1

BUFR #(
   .BUFR_DIVIDE("BYPASS"),   // Values: "BYPASS, 1, 2, 3, 4, 5, 6, 7, 8"
   .SIM_DEVICE("7SERIES")  // Must be set to "7SERIES"
)
BUFR_inst (
   .O(O),     // 1-bit output: Clock output port
   .CE(CE),   // 1-bit input: Active high, clock enable (Divided modes only)
```

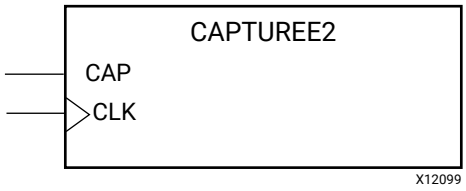
```
   .CLR(CLR), // 1-bit input: Active high, asynchronous clear (Divided modes only)
   .I(I)      // 1-bit input: Clock buffer input driven by an IBUF, MMCM or local interconnect
);

// End of BUFR_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

CAPTUREE2

Primitive: Register Capture



Introduction

This element provides user control and synchronization over when and how the capture register (flip-flop and latch) information task is requested. The readback function is provided through dedicated configuration port instructions. However, without this element, the readback data is synchronized to the configuration clock. Only register (flip-flop and latch) states can be captured. Although LUT RAM, SRL, and block RAM states are readback, they cannot be captured. An asserted high CAP signal indicates that the registers in the device are to be captured at the next Low-to-High clock transition. By default, data is captured after every trigger when transition on CLK while CAP is asserted. To limit the readback operation to a single data capture, add the ONESHOT=TRUE attribute to this element.

Port Descriptions

Port	Direction	Width	Function
CAP	Input	1	Capture Input
CLK	Input	1	Clock Input

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ONESHOT	STRING	"TRUE", "FALSE"	"TRUE"	Specifies the procedure for performing single readback per CAP trigger.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- CAPTUREE2: Register Capture
--            7 Series
-- Xilinx HDL Language Template, version 2020.1

CAPTUREE2_inst : CAPTUREE2
generic map (
   ONESHOT => "TRUE"  -- Specifies the procedure for performing single readback per CAP trigger.
)
port map (
   CAP => CAP, -- 1-bit input: Capture Input
   CLK => CLK  -- 1-bit input: Clock Input
);

-- End of CAPTUREE2_inst instantiation
```

Verilog Instantiation Template

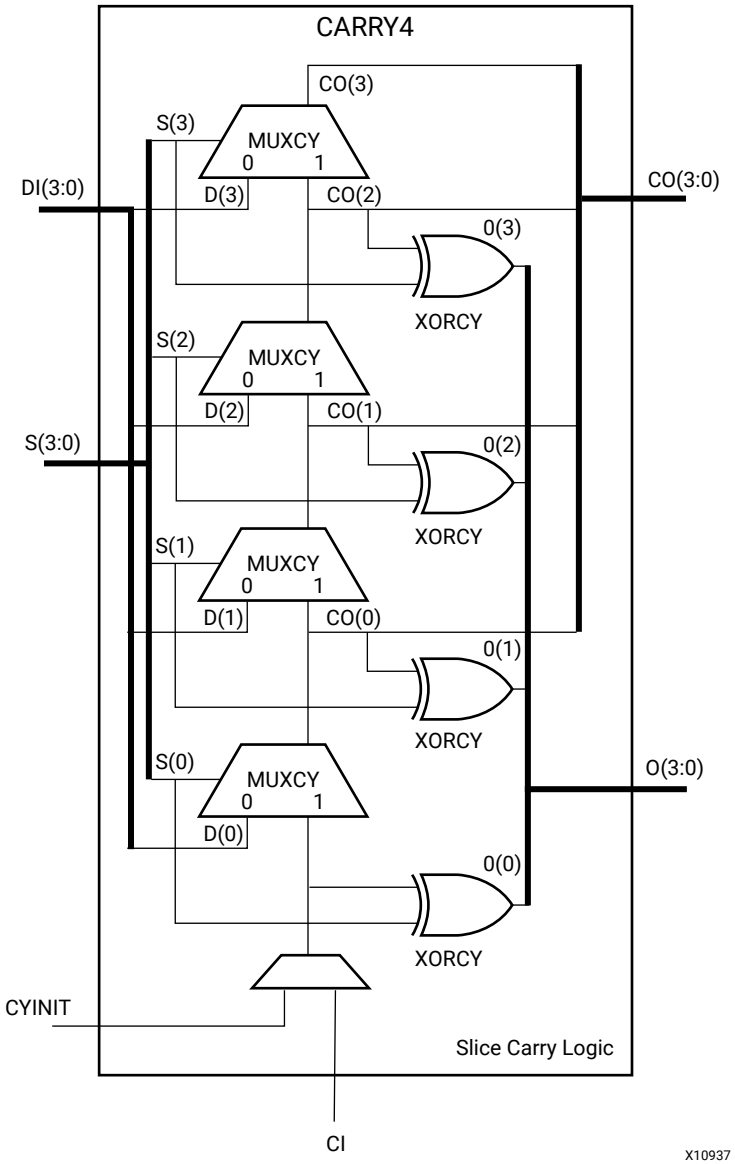
```
// CAPTUREE2: Register Capture
//            7 Series
// Xilinx HDL Language Template, version 2020.1

CAPTUREE2 #(
   .ONESHOT("TRUE")  // Specifies the procedure for performing single readback per CAP trigger.
)
CAPTUREE2_inst (
   .CAP(CAP), // 1-bit input: Capture Input
   .CLK(CLK)  // 1-bit input: Clock Input
);

// End of CAPTUREE2_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configuration Guide* (UG470).

CARRY4

Primitive: Fast Carry Logic with Look Ahead



Introduction

This circuit design represents the fast carry logic for a slice. The carry chain consists of a series of four MUXes and four XORs that connect to the other logic (LUTs) in the slice via dedicated routes to form more complex functions. The fast carry logic is useful for building arithmetic functions like adders, counters, subtracters and add/subs, as well as such other logic functions as wide comparators, address decoders, and some logic gates (specifically, AND and OR).

Port Descriptions

Port	Direction	Width	Function
O	Output	4	Carry chain XOR general data out.
CO	Output	4	Carry-out of each stage of the carry chain.
DI	Input	4	Carry-MUX data input.
S	Input	4	Carry-MUX select line.
CYINIT	Input	1	Carry-in initialization input.
CI	Input	1	Carry cascade input.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- CARRY4: Fast Carry Logic Component
--         7 Series
-- Xilinx HDL Language Template, version 2020.1

CARRY4_inst : CARRY4
port map (
   CO => CO,          -- 4-bit carry out
   O => O,            -- 4-bit carry chain XOR data out
   CI => CI,          -- 1-bit carry cascade input
   CYINIT => CYINIT,  -- 1-bit carry initialization
   DI => DI,          -- 4-bit carry-MUX data in
   S => S             -- 4-bit carry-MUX select input
);

-- End of CARRY4_inst instantiation
```

Verilog Instantiation Template

```
// CARRY4: Fast Carry Logic Component
//         7 Series
// Xilinx HDL Language Template, version 2020.1

CARRY4 CARRY4_inst (
   .CO(CO),          // 4-bit carry out
   .O(O),            // 4-bit carry chain XOR data out
   .CI(CI),          // 1-bit carry cascade input
   .CYINIT(CYINIT), // 1-bit carry initialization
```

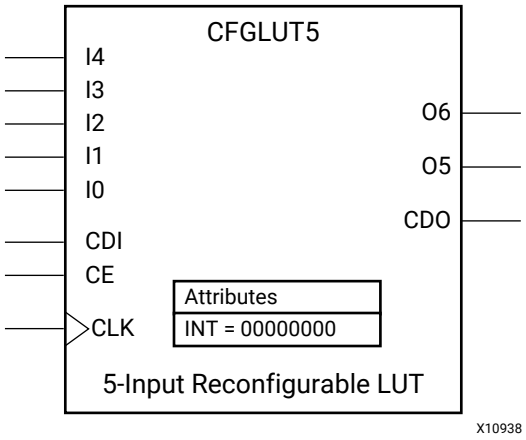
```
   .DI(DI),         // 4-bit carry-MUX data in
   .S(S)            // 4-bit carry-MUX select input
);

// End of CARRY4_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

CFGLUT5

Primitive: 5-input Dynamically Reconfigurable Look-Up Table (LUT)



Introduction

This element is a runtime, dynamically reconfigurable, 5-input look-up table (LUT) that enables the changing of the logical function of the LUT during circuit operation. Using the CDI pin, a new INIT value can be synchronously shifted in serially to change the logical function. The O6 output pin produces the logical output function, based on the current INIT value loaded into the LUT and the currently selected I0-I4 input pins. Optionally, you can use the O5 output in combination with the O6 output to create two individual 4-input functions sharing the same inputs or a 5-input function and a 4-input function that uses a subset of the 5-input logic (see the following tables). This component occupies one of the four LUT6 components within a Slice-M.

To cascade this element, connect the CDO pin from each element to the CDI input of the next element. This will allow a single serial chain of data (32-bits per LUT) to reconfigure multiple LUTs.

Port Descriptions

Port	Direction	Width	Function
O6	Output	1	5-LUT output.
O5	Output	1	4-LUT output.
I0, I1, I2, I3, I4	Input	1	LUT inputs.
CDO	Output	1	Reconfiguration data cascaded output (optionally connect to the CDI input of a subsequent LUT).
CDI	Input	1	Reconfiguration data serial input.
CLK	Input	1	Reconfiguration clock.
CE	Input	1	Active-High reconfiguration clock enable.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

- Connect the CLK input to the clock source used to supply the reconfiguration data.
- Connect the CDI input to the source of the reconfiguration data.
- Connect the CE pin to the active-High logic if you need to enable/disable LUT reconfiguration.
- Connect the I4-I0 pins to the source inputs to the logic equation. The logic function is output on O6 and O5.
- To cascade this element, connect the CDO pin from each element to the CDI input of the next element to allow a single serial chain of data to reconfigure multiple LUTs.

The INIT attribute should be placed on this design element to specify the initial logical function of the LUT. A new INIT can be loaded into the LUT any time during circuit operation by shifting in 32-bits per LUT in the chain, representing the new INIT value. Disregard the O6 and O5 output data until all 32-bits of new INIT data has been clocked into the LUT. The logical function of the LUT changes as new INIT data is shifted into it. Data should be shifted in MSB (INIT[31]) first and LSB (INIT[0]) last.

In order to understand the O6 and O5 logical value based on the current INIT, see the following table.

Table 9: **Logic Table**

I4 I3 I2 I1 I0	O6 Value	O5 Value
1 1 1 1 1	INIT[31]	INIT[15]
1 1 1 1 0	INIT[30]	INIT[14]
. . .	. . .	. . .
1 0 0 0 1	INIT[17]	INIT[1]
1 0 0 0 0	INIT[16]	INIT[0]
0 1 1 1 1	INIT[15]	INIT[15]
0 1 1 1 0	INIT[14]	INIT[14]
. . .	. . .	. . .
0 0 0 0 1	INIT[1]	INIT[1]
0 0 0 0 0	INIT[0]	INIT[0]

For instance, the INIT value of FFFF8000 would represent the following logical equations:

- O6 = I4 or (I3 and I2 and I1 and I0)

- O5 = I3 and I2 and I1 and I0

To use these elements as two, 4-input LUTs with the same inputs but different functions, tie the I4 signal to a logical one. The INIT[31:16] values apply to the logical values of the O6 output and INIT [15:0] apply to the logical values of the O5 output.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 32-bit value	All zeros	Specifies the initial logical expression of this element.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- CFGLUT5: Reconfigurable 5-input LUT (Mapped to SliceM LUT6)
--          7 Series
-- Xilinx HDL Language Template, version 2020.1

CFGLUT5_inst : CFGLUT5
generic map (
   INT => X"00000000")
port map (
   CDO => CDO, -- Reconfiguration cascade output
   O5 => O5,   -- 4-LUT output
   O6 => O6,   -- 5-LUT output
   CDI => CDI, -- Reconfiguration data input
   CE => CE,   -- Reconfiguration enable input
   CLK => CLK, -- Clock input
   I0 => I0,   -- Logic data input
   I1 => I1,   -- Logic data input
   I2 => I2,   -- Logic data input
   I3 => I3,   -- Logic data input
   I4 => I4    -- Logic data input
);

-- End of CFGLUT5_inst instantiation
```

Verilog Instantiation Template

```
// CFGLUT5: Reconfigurable 5-input LUT  (Mapped to a SliceM LUT6)
//          7 Series
// Xilinx HDL Language Template, version 2020.1

CFGLUT5 #(
   .INIT(32'h00000000) // Specify initial LUT contents
) CFGLUT5_inst (
   .CDO(CDO), // Reconfiguration cascade output
   .O5(O5),   // 4-LUT output
   .O6(O6),   // 5-LUT output
   .CDI(CDI), // Reconfiguration data input
   .CE(CE),   // Reconfiguration enable input
   .CLK(CLK), // Clock input
   .I0(I0),   // Logic data input
   .I1(I1),   // Logic data input
   .I2(I2),   // Logic data input
```

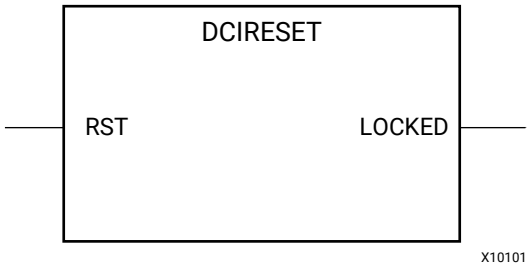
```
   .I3(I3),   // Logic data input
   .I4(I4)    // Logic data input
);

// End of CFGLUT5_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

DCIRESET

Primitive: Digitally Controlled Impedance Reset Component



Introduction

This design element is used to reset the Digitally Controlled Impedance (DCI) state machine after configuration has been completed. By toggling the RST input to the DCIRESET primitive while the device is operating, the DCI state-machine is reset and both phases of impedance adjustment proceed in succession. All I/Os using DCI will be unavailable until the LOCKED output from the DCIRESET block is asserted.

Port Descriptions

Port	Direction	Width	Function
LOCKED	Output	1	DCI state-machine LOCK status output. When low, DCI I/O impedance is being calibrated and DCI I/Os are unavailable. Upon a low-to-high assertion, DCI I/Os are available for use.
RST	Input	1	Active-High asynchronous reset input to DCI state-machine. After RST is asserted, I/Os using DCI will be unavailable until LOCKED is asserted.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- DCIRESET: Digitally Controlled Impedance Reset Component
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

DCIRESET_inst : DCIRESET
port map (
   LOCKED => LOCKED, -- 1-bit output: LOCK status output
   RST => RST        -- 1-bit input: Active-high asynchronous reset input
);

-- End of DCIRESET_inst instantiation
```

Verilog Instantiation Template

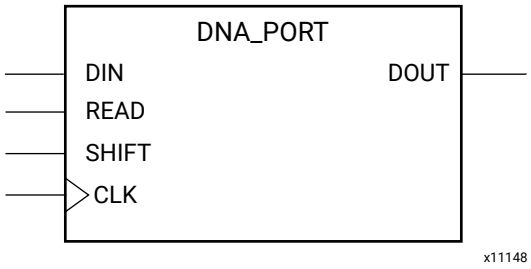
```
// DCIRESET: Digitally Controlled Impedance Reset Component
//           7 Series
// Xilinx HDL Language Template, version 2020.1

DCIRESET DCIRESET_inst (
   .LOCKED(LOCKED), // 1-bit output: LOCK status output
   .RST(RST)        // 1-bit input: Active-high asynchronous reset input
);

// End of DCIRESET_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

DNA_PORT

Primitive: Device DNA Access Port



Introduction

The DNA_PORT allows access to a dedicated shift register that can be loaded with the Device DNA data bits (factory-programmed, read-only ID) for a given 7 Series device. In addition to shifting out the DNA data bits, this component allows for the inclusion of supplemental bits of your data, or allows for the DNA data to rollover (repeat DNA data after initial data has been shifted out). This component is primarily used with other circuitry to build added copy protection for the FPGA bitstream from possible theft. Connect all inputs and outputs to the design to ensure proper operation.

To access the Device DNA data, first load the shift register by setting the active-High READ signal for one clock cycle. After the shift register is loaded, the data can be synchronously shifted out by enabling the active-High SHIFT input and capturing the data out the DOUT output port. Additional data can be appended to the end of the 57-bit shift register by connecting the appropriate logic to the DIN port. If DNA data rollover is desired, connect the DOUT port directly to the DIN port to allow for the same data to be shifted out after completing the 57-bit shift operation. If no additional data is necessary, the DIN port can be tied to a logic zero. The attribute SIM_DNA_VALUE can be optionally set to allow for simulation of a possible DNA data sequence. By default, the Device DNA data bits are all zeros in the simulation model.

Port Descriptions

Port	Direction	Width	Function
CLK	Input	1	Clock input.
DIN	Input	1	User data input pin.
DOUT	Output	1	DNA output data.
READ	Input	1	Active-High load DNA, active-Low read input.
SHIFT	Input	1	Active-High shift enable input.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
SIM_DNA_VALUE	HEX	57-bit HEX value	All zeros	Specifies a sample 57-bit DNA value for simulation.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- DNA_PORT: Device DNA Access Port
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

DNA_PORT_inst : DNA_PORT
generic map (
   SIM_DNA_VALUE => X"000000000000000"  -- Specifies a sample 57-bit DNA value for simulation
)
port map (
   DOUT => DOUT,   -- 1-bit output: DNA output data.
   CLK => CLK,     -- 1-bit input: Clock input.
   DIN => DIN,     -- 1-bit input: User data input pin.
   READ => READ,   -- 1-bit input: Active high load DNA, active low read input.
   SHIFT => SHIFT  -- 1-bit input: Active high shift enable input.
);

-- End of DNA_PORT_inst instantiation
```

Verilog Instantiation Template

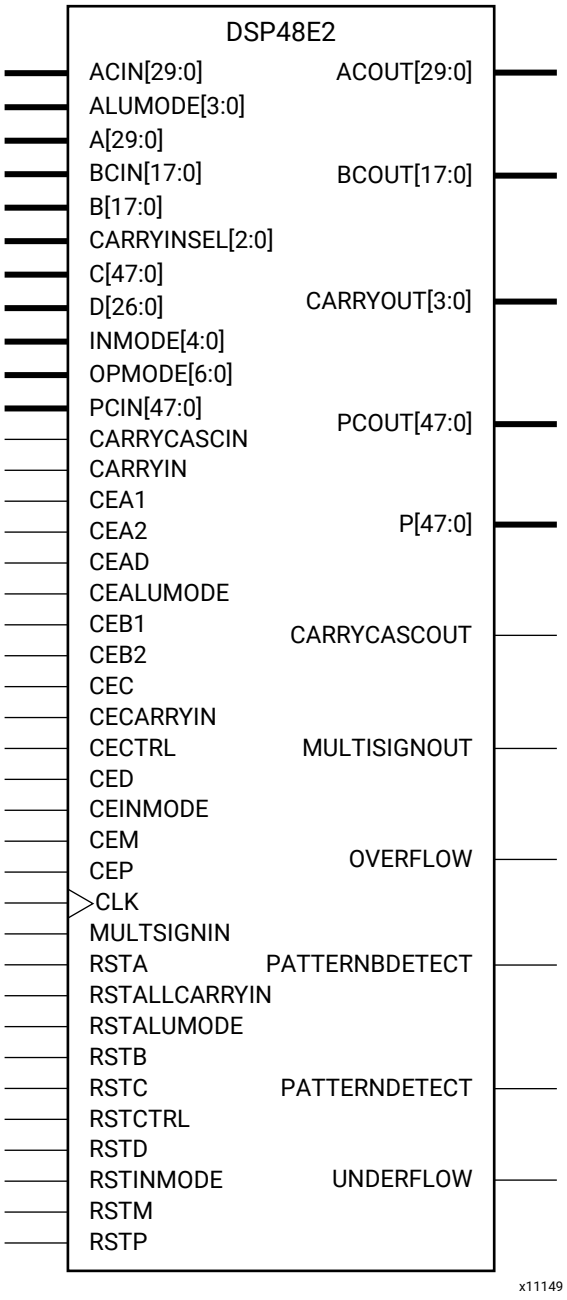
```
// DNA_PORT: Device DNA Access Port
//           7 Series
// Xilinx HDL Language Template, version 2020.1

DNA_PORT #(
   .SIM_DNA_VALUE(57'h000000000000000)  // Specifies a sample 57-bit DNA value for simulation
)
DNA_PORT_inst (
   .DOUT(DOUT),   // 1-bit output: DNA output data.
   .CLK(CLK),     // 1-bit input: Clock input.
   .DIN(DIN),     // 1-bit input: User data input pin.
   .READ(READ),   // 1-bit input: Active high load DNA, active low read input.
   .SHIFT(SHIFT)  // 1-bit input: Active high shift enable input.
);

// End of DNA_PORT_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configuration Guide* (UG470).

DSP48E1

Primitive: 48-bit Multi-Functional Arithmetic Block



Introduction

This design element is a scalable dedicated block in 7 series devices that lets you create compact, high-speed, arithmetic-intensive operations such as those seen for many DSP algorithms. Functions that the block is capable of include multiplication, addition, subtraction, accumulation, shifting, logical operations, and pattern detection.

Port Descriptions

Port	Direction	Width	Function
A<29:0>	Input	30	Data input for pre-adder, multiplier, adder/subtracter/ accumulator, ALU, or concatenation operations. • When used with the multiplier or pre-adder, 25-bits of data (A[24:0]) is used and upper bits (A[29:25]) are unused and may be tied to ground. • When using the internal adder/subtracter/accumulator or ALU circuit, all 30-bits are used (A[29:0]). • When used in concatenation mode, all 30-bits are used and this constitutes the MSB (upper) bits of the concatenated vector.
ACIN<29:0>	Input	30	Cascaded data input from ACOUT of previous DSP48E1 slice (multiplexed with A). If not used, tie port to all zeros.
ACOUT<29:0>	Output	30	Cascaded data output to ACIN of next DSP48E1 slice. If not used, leave unconnected.
ALUMODE<3:0>	Input	4	Controls the selection of the logic function in the DSP48E1 slice.
B<17:0>	Input	18	The B input of the multiplier. B[17:0] are the least significant bits (LSBs) of the A:B concatenated input to the second-stage adder/subtracter or logic function.
BCIN<17:0>	Input	18	Cascaded data input from BCOUT of previous DSP48E1 slice (muxed with B). If not used, tie port to all zeros.
BCOUT<17:0>	Output	18	Cascaded data output to BCIN of next DSP48E1 slice. If not used, leave unconnected.
C<47:0>	Input	48	Data input to the second-stage adder/subtracter, pattern detector, or logic function.
CARRYCASCIN	Input	1	Cascaded carry input from CARRYCASCOUT of previous DSP48E1 slice.
CARRYCASCOUT	Output	1	Cascaded carry output to CARRYCASCIN of next DSP48E1 slice. This signal is internally fed back into the CARRYINSEL multiplexer input of the same DSP48E1 slice.
CARRYIN	Input	1	Carry input from the FPGA logic.

Port	Direction	Width	Function
CARRYINSEL <2:0>	Input	3	Selects the carry source: • 0 1 1 - PCIN[47]: Rounding PCIN (round towards zero) • 1 0 0 - CARRYCASCOUT: For larger add/sub/acc (sequential operation via internal feedback). Must select with PREG=1 • 1 0 1 - ~P[47]: Rounding P (round towards infinity). Must select with PREG=1 • 1 1 0 - A[24]: XNOR B[17] Rounding A x B • 1 1 1 - P[47]: For rounding P (round towards zero). Must select with PREG=1
CARRYOUT<3:0>	Output	4	4-bit carry output from each 12-bit field of the accumulate/ adder/logic unit. Normal 48-bit operation uses only CARRYOUT3. SIMD operation can use four carry out bits (CARRYOUT[3:0]).
CEAD	Input	1	Active-High clock enable for the pre-adder output AD pipeline register. Tie to logic one if not used and ADREG=1. Tie to logic zero if ADREG=0.
CEALUMODE	Input	1	Active-High clock enable for ALUMODE (control inputs) registers (ALUMODEREG=1). Tie to logic one if not used.
CEA1	Input	1	Active-High clock enable for the first A (input) register. This port is only used if AREG=2 or INMODE0 = 1. Tie to logic one if not used and AREG=2. Tie to logic zero if AREG=0 or 1. When two registers are used, this is the first sequentially. When Dynamic AB Access is used, this clock enable is applied for INMODE[0]=1.
CEA2	Input	1	Active-High clock enable for the second A (input) register. This port is only used if AREG=1 or 2. Tie to logic one if not used and AREG=1 or 2. Tie to logic zero if AREG=0. When two registers are used, this is the second sequentially. When one register is used (AREG=1), CEA2 is the clock enable.
CEB1	Input	1	Active-High, Clock enable for the first B (input) register. This port is only used if BREG=2 or INMODE4=1. Tie to logic one if not used and BREG=2. Tie to logic zero if BREG=0 or 1. When two registers are used, this is the first sequentially. When Dynamic AB Access is used, this clock enable is applied for INMODE[4]=1.
CEB2	Input	1	Active-High clock enable for the second B (input) register. This port is only used if BREG=1 or 2. Tie to logic one if not used and BREG=1 or 2. Tie to logic zero if BREG=0. When two registers are used, this is the second sequentially. When one register is used (BREG=1), CEB2 is the clock enable.
CEC	Input	1	Active-High clock enable for the C (input) register (CREG=1). Tie to logic one if not used.
CECARRYIN	Input	1	Active-High clock enable for the CARRYIN (input from fabric) register (CARRYINREG=1). Tie to logic one if not used.
CECTRL	Input	1	Active-High clock enable for the OPMODE and CARRYINSEL (control inputs) registers (OPMODEREG=1 or CARRYINSELREG=1). Tie to logic one if not used.
CED	Input	1	Active-High Clock enable for the D (input) registers (DREG=1). Tie to logic one if not used.

Port	Direction	Width	Function
CEINMODE	Input	1	Active-High clock enable for the INMODE control input registers (INMODEREG=1). Tie to logic one if not used.
CEM	Input	1	Active-High Clock enable for the post-multiply M (pipeline) register and the internal multiply round CARRYIN register (MREG=1). Tie to logic one if not used.
CEP	Input	1	Active-High clock enable for the P (output) register (PREG=1). Tie to logic one if not used.
CLK	Input	1	The DSP48E1 input clock common to all internal registers and flip-flops.
D<24:0>	Input	25	25-bit data input to the pre-adder or alternative input to the multiplier. The pre-adder implements D + A as determined by the INMODE3 signal.
INMODE<4:0>	Input	5	These five control bits select the functionality of the pre-adder, the A, B, and D inputs, and the input registers. These bits should be tied to all zeros if not used.
MULTSIGNIN	Input	1	Sign of the multiplied result from the previous DSP48E1 slice for MACC extension. Either connect to the MULTSIGNOUT of another DSP block or tie to ground if not used.
MULTSIGNOUT	Output	1	Sign of the multiplied result cascaded to the next DSP48E1 slice for MACC extension. Either connect to the MULTSIGNIN of another DSP block or tie to ground if not used.
OPMODE<6:0>	Input	7	Controls the input to the X, Y, and Z multiplexers in the DSP48E1 slice dictating the operation or function of the DSP slice.
OVERFLOW	Output	1	Active-High Overflow indicator when used with the appropriate setting of the pattern detector and PREG=1.
P<47:0>	Output	48	Data output from second stage adder/subtracter or logic function.
PATTERNBDETECT	Output	1	Active-High match indicator between P[47:0] and the pattern bar.
PATTERNDETECT	Output	1	Active-High Match indicator between P[47:0] and the pattern gated by the MASK. Result arrives on the same cycle as P.
PCIN<47:0>	Input	48	Cascaded data input from PCOUT of previous DSP48E1 slice to adder. If used, connect to PCOUT of upstream cascaded DSP slice. If not used, tie port to all zeros.
PCOUT<47:0>	Output	48	Cascaded data output to PCIN of next DSP48E1 slice. If used, connect to PCIN of downstream cascaded DSP slice. If not used, leave unconnected.
RSTA	Input	1	Active-High synchronous Reset for both A (input) registers (AREG=1 or 2). Tie to logic zero if not used.
RSTALLCARRYIN	Input	1	Active-High, synchronous reset for the Carry (internal path) and the CARRYIN registers (CARRYINREG=1). Tie to logic zero if not used.
RSTALUMODE	Input	1	Active-High synchronous Reset for ALUMODE (control inputs) registers (ALUMODEREG=1). Tie to logic zero if not used.
RSTB	Input	1	Active-High, synchronous Reset for both B (input) registers (BREG=1 or 2). Tie to logic zero if not used.
RSTC	Input	1	Active-High synchronous reset for the C (input) registers (CREG=1). Tie to logic zero if not used.

Port	Direction	Width	Function
RSTCTRL	Input	1	Active-High synchronous reset for OPMODE and CARRYINSEL (control inputs) registers (OPMODEREG=1 and/or CARRYINSELREG=1). Tie to logic zero if not used.
RSTD	Input	1	Active-High synchronous reset for the D (input) register and for the pre-adder (output) AD pipeline register (DREG=1 and/or ADREG=1). Tie to logic zero if not used.
RSTINMODE	Input	1	Active-High synchronous reset for the INMODE (control input) registers (INMODEREG=1). Tie to logic zero if not used.
RSTM	Input	1	Active-High synchronous reset for the M (pipeline) registers (MREG=1). Tie to logic zero if not used.
RSTP	Input	1	Active-High, synchronous reset for the P (output) registers (PREG=1). Tie to logic zero if not used.
UNDERFLOW	Output	1	Active-High underflow indicator when used with the appropriate setting of the pattern detector and PREG=1.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	Yes
Macro support	Yes

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ACASCREG	DECIMAL	1, 0, 2	1	In conjunction with AREG, selects the number of A input registers on the A cascade path, ACOUT. This attribute must be equal to or one less than the AREG value: • AREG=0: ACASCREG must be 0. • AREG=1: ACASCREG must be 1. • AREG=2: ACASCREG can be 1 or 2.
ADREG	DECIMAL	1, 0	1	Selects the number of AD pipeline registers. Set to 1 to use the AD pipeline registers.
A_INPUT	STRING	"DIRECT", "CASCADE"	"DIRECT"	Selects the input to the A port between parallel input ("DIRECT") or the cascaded input from the previous slice ("CASCADE").
ALUMODEREG	DECIMAL	1, 0	1	Selects the number of ALUMODE input registers. Set to 1 to register the ALUMODE inputs.
AREG	DECIMAL	1, 0, 2	1	Selects the number of A input pipeline registers.

Attribute	Type	Allowed Values	Default	Description
AUTORESET _PATDET	STRING	"NO_RESET", "RESET_MATCH", "RESET_NOT _MATCH"	"NO_RESET"	Automatically resets the P Register (accumulated value or counter value) on the next clock cycle, if a pattern detect event has occurred on this clock cycle. The "RESET_MATCH" and "RESET_NOT_MATCH" settings distinguish between whether the DSP48E1 slice should cause an auto reset of the P Register on the next cycle: • if the pattern is matched or • whenever the pattern is not matched on the current cycle but was matched on the previous clock cycle.
BCASCREG	DECIMAL	1, 0, 2	1	In conjunction with BREG, selects the number of B input registers on the B cascade path, BCOUT. This attribute must be equal to or one less than the BREG value: • BREG=0: BCASCREG must be 0. • BREG=1: BCASCREG must be 1. • BREG=2: BCASCREG can be 1 or 2.
B_INPUT	STRING	"DIRECT", "CASCADE"	"DIRECT"	Selects the input to the B port between parallel input ("DIRECT") or the cascaded input from the previous slice ("CASCADE").
BREG	DECIMAL	1, 0, 2	1	Selects the number of B input registers.
CARRYINREG	DECIMAL	1, 0	1	Selects the number of CARRYIN input registers. Set to 1 to register the CARRYIN inputs.
CARRYINSELREG	DECIMAL	1, 0	1	Selects the number of CARRYINSEL input registers. Set to 1 to register the CARRYINSEL inputs.
CREG	DECIMAL	1, 0	1	Selects the number of C input registers. Set to 1 to register the C inputs.
DREG	DECIMAL	1, 0	1	Selects the number of D input registers. Set to 1 to register the D inputs.
INMODEREG	DECIMAL	1, 0	1	Selects the number of INMODE input registers. Set to 1 to register the INMODE inputs.
MASK	HEX	48-bit HEX	All ones	This 48-bit value is used to mask out certain bits during a pattern detection. • When a MASK bit is set to 1, the corresponding pattern bit is ignored. • When a MASK bit is set to 0, the pattern bit is compared.
MREG	DECIMAL	1, 0	1	Selects the number of multiplier output (M) pipeline register stages. Set to 1 to use the M pipeline registers.
OPMODEREG	DECIMAL	1, 0	1	Selects the number of OPMODE input registers. Set to 1 to register the OPMODE inputs.
PATTERN	HEX	48-bit HEX	All zeros	This 48-bit value is used in the pattern detector.

Attribute	Type	Allowed Values	Default	Description
PREG	DECIMAL	1, 0	1	Selects the number of P output registers. Set to 1 to register the P outputs. The registered outputs will include: • CARRYOUT • CARRYCASCOUT • MULTSIGNOUT • PATTERNB_DETECT • PCOUT
SEL_MASK	STRING	"MASK", "C", "ROUNDING_MODE1", "ROUNDING_MODE2"	"MASK"	Selects the mask to be used for the pattern detector. The C and MASK settings are for standard uses of the pattern detector (counter, overflow detection, etc.). ROUNDING_MODE1 (Cbar left shifted by 1) and ROUNDING_MODE2 (C-bar left shifted by 2) select special masks based off of the optionally registered C port. These rounding modes can be used to implement convergent rounding in the DSP48E1 slice using the pattern detector.
SEL_PATTERN	STRING	"PATTERN", "C"	"PATTERN"	Selects the input source for the pattern field. The input source can either be a 48-bit dynamic C input or a 48-bit static PATTERN attribute field.
USE_DPORT	BOOLEAN	FALSE, TRUE	FALSE	Determines whether the pre-adder and the D Port are used or not.
USE_MULT	STRING	"MULTIPLY", "DYNAMIC", "NONE"	"MULTIPLY"	Selects usage of the multiplier. Set to "NONE" to save power when using only the Adder/Logic Unit. The "DYNAMIC" setting indicates that the user is switching between A*B and A:B operations on the fly and therefore needs to get the worst-case timing of the two paths.
USE_PATTERN_DETECT	STRING	"NO_PATDET", "PATDET"	"NO_PATDET"	Selects whether the pattern detector and related features are used ("PATDET") or not used ("NO_PATDET"). This attribute is used for speed specification and Simulation Model purposes only.
USE_SIMD	STRING	"ONE48", "FOUR12", "TWO24"	"ONE48"	Selects the mode of operation for the adder/subtracter. The attribute setting can be one 48-bit adder mode ("ONE48"), two 24- bit adder mode ("TWO24"), or four 12-bit adder mode ("FOUR12"). Selecting "ONE48" mode is compatible with Virtex-5 DSP48 operation and is not actually a true SIMD mode. Typical Multiply-Add operations are supported when the mode is set to "ONE48". When either "TWO24" or "FOUR12" mode is selected, the multiplier must not be used, and USE_MULT must be set to "NONE".

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- DSP48E1: 48-bit Multi-Functional Arithmetic Block
--          7 Series
-- Xilinx HDL Language Template, version 2020.1

DSP48E1_inst : DSP48E1
generic map (
   -- Feature Control Attributes: Data Path Selection
   A_INPUT => "DIRECT",               -- Selects A input source, "DIRECT" (A port) or "CASCADE" (ACIN port)
   B_INPUT => "DIRECT",               -- Selects B input source, "DIRECT" (B port) or "CASCADE" (BCIN port)
   USE_DPORT => FALSE,                -- Select D port usage (TRUE or FALSE)
   USE_MULT => "MULTIPLY",            -- Select multiplier usage ("MULTIPLY", "DYNAMIC", or "NONE")
   USE_SIMD => "ONE48",               -- SIMD selection ("ONE48", "TWO24", "FOUR12")
   -- Pattern Detector Attributes: Pattern Detection Configuration
   AUTORESET_PATDET => "NO_RESET",    -- "NO_RESET", "RESET_MATCH", "RESET_NOT_MATCH"
   MASK => X"3fffffffffff",           -- 48-bit mask value for pattern detect (1=ignore)
   PATTERN => X"000000000000",        -- 48-bit pattern match for pattern detect
   SEL_MASK => "MASK",                -- "C", "MASK", "ROUNDING_MODE1", "ROUNDING_MODE2"
   SEL_PATTERN => "PATTERN",          -- Select pattern value ("PATTERN" or "C")
   USE_PATTERN_DETECT => "NO_PATDET", -- Enable pattern detect ("PATDET" or "NO_PATDET")
   -- Register Control Attributes: Pipeline Register Configuration
   ACASCREG => 1,                     -- Number of pipeline stages between A/ACIN and ACOUT (0, 1 or 2)
   ADREG => 1,                        -- Number of pipeline stages for pre-adder (0 or 1)
   ALUMODEREG => 1,                   -- Number of pipeline stages for ALUMODE (0 or 1)
   AREG => 1,                         -- Number of pipeline stages for A (0, 1 or 2)
   BCASCREG => 1,                     -- Number of pipeline stages between B/BCIN and BCOUT (0, 1 or 2)
   BREG => 1,                         -- Number of pipeline stages for B (0, 1 or 2)
   CARRYINREG => 1,                   -- Number of pipeline stages for CARRYIN (0 or 1)
   CARRYINSELREG => 1,                -- Number of pipeline stages for CARRYINSEL (0 or 1)
   CREG => 1,                         -- Number of pipeline stages for C (0 or 1)
   DREG => 1,                         -- Number of pipeline stages for D (0 or 1)
   INMODEREG => 1,                    -- Number of pipeline stages for INMODE (0 or 1)
   MREG => 1,                         -- Number of multiplier pipeline stages (0 or 1)
   OPMODEREG => 1,                    -- Number of pipeline stages for OPMODE (0 or 1)
   PREG => 1                          -- Number of pipeline stages for P (0 or 1)
)
port map (
   -- Cascade: 30-bit (each) output: Cascade Ports
   ACOUT => ACOUT,                    -- 30-bit output: A port cascade output
   BCOUT => BCOUT,                    -- 18-bit output: B port cascade output
   CARRYCASCOUT => CARRYCASCOUT,      -- 1-bit output: Cascade carry output
   MULTSIGNOUT => MULTSIGNOUT,        -- 1-bit output: Multiplier sign cascade output
   PCOUT => PCOUT,                    -- 48-bit output: Cascade output
   -- Control: 1-bit (each) output: Control Inputs/Status Bits
   OVERFLOW => OVERFLOW,              -- 1-bit output: Overflow in add/acc output
   PATTERNBDETECT => PATTERNBDETECT,  -- 1-bit output: Pattern bar detect output
   PATTERNDETECT => PATTERNDETECT,    -- 1-bit output: Pattern detect output
   UNDERFLOW => UNDERFLOW,            -- 1-bit output: Underflow in add/acc output
   -- Data: 4-bit (each) output: Data Ports
   CARRYOUT => CARRYOUT,              -- 4-bit output: Carry output
   P => P,                            -- 48-bit output: Primary data output
   -- Cascade: 30-bit (each) input: Cascade Ports
   ACIN => ACIN,                      -- 30-bit input: A cascade data input
   BCIN => BCIN,                      -- 18-bit input: B cascade input
   CARRYCASCIN => CARRYCASCIN,        -- 1-bit input: Cascade carry input
   MULTSIGNIN => MULTSIGNIN,          -- 1-bit input: Multiplier sign input
   PCIN => PCIN,                      -- 48-bit input: P cascade input
   -- Control: 4-bit (each) input: Control Inputs/Status Bits
   ALUMODE => ALUMODE,                -- 4-bit input: ALU control input
   CARRYINSEL => CARRYINSEL,          -- 3-bit input: Carry select input
   CLK => CLK,                        -- 1-bit input: Clock input
   INMODE => INMODE,                  -- 5-bit input: INMODE control input
   OPMODE => OPMODE,                  -- 7-bit input: Operation mode input
   -- Data: 30-bit (each) input: Data Ports
   A => A,                            -- 30-bit input: A data input
   B => B,                            -- 18-bit input: B data input
   C => C,                            -- 48-bit input: C data input
```

```
   CARRYIN => CARRYIN,               -- 1-bit input: Carry input signal
   D => D,                           -- 25-bit input: D data input
   -- Reset/Clock Enable: 1-bit (each) input: Reset/Clock Enable Inputs
   CEA1 => CEA1,                     -- 1-bit input: Clock enable input for 1st stage AREG
   CEA2 => CEA2,                     -- 1-bit input: Clock enable input for 2nd stage AREG
   CEAD => CEAD,                     -- 1-bit input: Clock enable input for ADREG
   CEALUMODE => CEALUMODE,           -- 1-bit input: Clock enable input for ALUMODE
   CEB1 => CEB1,                     -- 1-bit input: Clock enable input for 1st stage BREG
   CEB2 => CEB2,                     -- 1-bit input: Clock enable input for 2nd stage BREG
   CEC => CEC,                       -- 1-bit input: Clock enable input for CREG
   CECARRYIN => CECARRYIN,           -- 1-bit input: Clock enable input for CARRYINREG
   CECTRL => CECTRL,                 -- 1-bit input: Clock enable input for OPMODEREG and CARRYINSELREG
   CED => CED,                       -- 1-bit input: Clock enable input for DREG
   CEINMODE => CEINMODE,             -- 1-bit input: Clock enable input for INMODEREG
   CEM => CEM,                       -- 1-bit input: Clock enable input for MREG
   CEP => CEP,                       -- 1-bit input: Clock enable input for PREG
   RSTA => RSTA,                     -- 1-bit input: Reset input for AREG
   RSTALLCARRYIN => RSTALLCARRYIN,   -- 1-bit input: Reset input for CARRYINREG
   RSTALUMODE => RSTALUMODE,         -- 1-bit input: Reset input for ALUMODEREG
   RSTB => RSTB,                     -- 1-bit input: Reset input for BREG
   RSTC => RSTC,                     -- 1-bit input: Reset input for CREG
   RSTCTRL => RSTCTRL,               -- 1-bit input: Reset input for OPMODEREG and CARRYINSELREG
   RSTD => RSTD,                     -- 1-bit input: Reset input for DREG and ADREG
   RSTINMODE => RSTINMODE,           -- 1-bit input: Reset input for INMODEREG
   RSTM => RSTM,                     -- 1-bit input: Reset input for MREG
   RSTP => RSTP                      -- 1-bit input: Reset input for PREG
);

-- End of DSP48E1_inst instantiation
```

Verilog Instantiation Template

```
// DSP48E1: 48-bit Multi-Functional Arithmetic Block
//          7 Series
// Xilinx HDL Language Template, version 2020.1

DSP48E1 #(
   // Feature Control Attributes: Data Path Selection
   .A_INPUT("DIRECT"),               // Selects A input source, "DIRECT" (A port) or "CASCADE" (ACIN port)
   .B_INPUT("DIRECT"),               // Selects B input source, "DIRECT" (B port) or "CASCADE" (BCIN port)
   .USE_DPORT("FALSE"),              // Select D port usage (TRUE or FALSE)
   .USE_MULT("MULTIPLY"),            // Select multiplier usage ("MULTIPLY", "DYNAMIC", or "NONE")
   .USE_SIMD("ONE48"),               // SIMD selection ("ONE48", "TWO24", "FOUR12")
   // Pattern Detector Attributes: Pattern Detection Configuration
   .AUTORESET_PATDET("NO_RESET"),    // "NO_RESET", "RESET_MATCH", "RESET_NOT_MATCH"
   .MASK(48'h3fffffffffff),          // 48-bit mask value for pattern detect (1=ignore)
   .PATTERN(48'h000000000000),       // 48-bit pattern match for pattern detect
   .SEL_MASK("MASK"),                // "C", "MASK", "ROUNDING_MODE1", "ROUNDING_MODE2"
   .SEL_PATTERN("PATTERN"),          // Select pattern value ("PATTERN" or "C")
   .USE_PATTERN_DETECT("NO_PATDET"), // Enable pattern detect ("PATDET" or "NO_PATDET")
   // Register Control Attributes: Pipeline Register Configuration
   .ACASCREG(1),                     // Number of pipeline stages between A/ACIN and ACOUT (0, 1 or 2)
   .ADREG(1),                        // Number of pipeline stages for pre-adder (0 or 1)
   .ALUMODEREG(1),                   // Number of pipeline stages for ALUMODE (0 or 1)
   .AREG(1),                         // Number of pipeline stages for A (0, 1 or 2)
   .BCASCREG(1),                     // Number of pipeline stages between B/BCIN and BCOUT (0, 1 or 2)
   .BREG(1),                         // Number of pipeline stages for B (0, 1 or 2)
   .CARRYINREG(1),                   // Number of pipeline stages for CARRYIN (0 or 1)
   .CARRYINSELREG(1),                // Number of pipeline stages for CARRYINSEL (0 or 1)
   .CREG(1),                         // Number of pipeline stages for C (0 or 1)
   .DREG(1),                         // Number of pipeline stages for D (0 or 1)
   .INMODEREG(1),                    // Number of pipeline stages for INMODE (0 or 1)
   .MREG(1),                         // Number of multiplier pipeline stages (0 or 1)
   .OPMODEREG(1),                    // Number of pipeline stages for OPMODE (0 or 1)
   .PREG(1)                          // Number of pipeline stages for P (0 or 1)
)
DSP48E1_inst (
   // Cascade: 30-bit (each) output: Cascade Ports
   .ACOUT(ACOUT),                   // 30-bit output: A port cascade output
   .BCOUT(BCOUT),                   // 18-bit output: B port cascade output
   .CARRYCASCOUT(CARRYCASCOUT),     // 1-bit output: Cascade carry output
   .MULTSIGNOUT(MULTSIGNOUT),       // 1-bit output: Multiplier sign cascade output
   .PCOUT(PCOUT),                   // 48-bit output: Cascade output
   // Control: 1-bit (each) output: Control Inputs/Status Bits
   .OVERFLOW(OVERFLOW),             // 1-bit output: Overflow in add/acc output
   .PATTERNBDETECT(PATTERNBDETECT), // 1-bit output: Pattern bar detect output
```

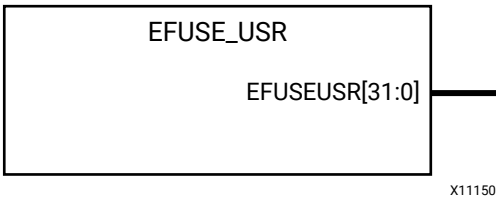
```
   .PATTERNDETECT(PATTERNDETECT),   // 1-bit output: Pattern detect output
   .UNDERFLOW(UNDERFLOW),           // 1-bit output: Underflow in add/acc output
   // Data: 4-bit (each) output: Data Ports
   .CARRYOUT(CARRYOUT),             // 4-bit output: Carry output
   .P(P),                           // 48-bit output: Primary data output
   // Cascade: 30-bit (each) input: Cascade Ports
   .ACIN(ACIN),                     // 30-bit input: A cascade data input
   .BCIN(BCIN),                     // 18-bit input: B cascade input
   .CARRYCASCIN(CARRYCASCIN),       // 1-bit input: Cascade carry input
   .MULTSIGNIN(MULTSIGNIN),         // 1-bit input: Multiplier sign input
   .PCIN(PCIN),                     // 48-bit input: P cascade input
   // Control: 4-bit (each) input: Control Inputs/Status Bits
   .ALUMODE(ALUMODE),               // 4-bit input: ALU control input
   .CARRYINSEL(CARRYINSEL),         // 3-bit input: Carry select input
   .CLK(CLK),                       // 1-bit input: Clock input
   .INMODE(INMODE),                 // 5-bit input: INMODE control input
   .OPMODE(OPMODE),                 // 7-bit input: Operation mode input
   // Data: 30-bit (each) input: Data Ports
   .A(A),                           // 30-bit input: A data input
   .B(B),                           // 18-bit input: B data input
   .C(C),                           // 48-bit input: C data input
   .CARRYIN(CARRYIN),               // 1-bit input: Carry input signal
   .D(D),                           // 25-bit input: D data input
   // Reset/Clock Enable: 1-bit (each) input: Reset/Clock Enable Inputs
   .CEA1(CEA1),                     // 1-bit input: Clock enable input for 1st stage AREG
   .CEA2(CEA2),                     // 1-bit input: Clock enable input for 2nd stage AREG
   .CEAD(CEAD),                     // 1-bit input: Clock enable input for ADREG
   .CEALUMODE(CEALUMODE),           // 1-bit input: Clock enable input for ALUMODE
   .CEB1(CEB1),                     // 1-bit input: Clock enable input for 1st stage BREG
   .CEB2(CEB2),                     // 1-bit input: Clock enable input for 2nd stage BREG
   .CEC(CEC),                       // 1-bit input: Clock enable input for CREG
   .CECARRYIN(CECARRYIN),           // 1-bit input: Clock enable input for CARRYINREG
  .CECTRL(CECTRL),                // 1-bit input: Clock enable input for OPMODEREG and CARRYINSELREG
   .CED(CED),                       // 1-bit input: Clock enable input for DREG
   .CEINMODE(CEINMODE),             // 1-bit input: Clock enable input for INMODEREG
   .CEM(CEM),                       // 1-bit input: Clock enable input for MREG
   .CEP(CEP),                       // 1-bit input: Clock enable input for PREG
   .RSTA(RSTA),                     // 1-bit input: Reset input for AREG
   .RSTALLCARRYIN(RSTALLCARRYIN),   // 1-bit input: Reset input for CARRYINREG
   .RSTALUMODE(RSTALUMODE),         // 1-bit input: Reset input for ALUMODEREG
   .RSTB(RSTB),                     // 1-bit input: Reset input for BREG
   .RSTC(RSTC),                     // 1-bit input: Reset input for CREG
   .RSTCTRL(RSTCTRL),               // 1-bit input: Reset input for OPMODEREG and CARRYINSELREG
   .RSTD(RSTD),                     // 1-bit input: Reset input for DREG and ADREG
   .RSTINMODE(RSTINMODE),           // 1-bit input: Reset input for INMODEREG
   .RSTM(RSTM),                     // 1-bit input: Reset input for MREG
   .RSTP(RSTP)                      // 1-bit input: Reset input for PREG
);

// End of DSP48E1_inst instantiation
```

For More Information

- See the *7 Series DSP43E1 Slice User Guide* (UG479).

EFUSE_USR

Primitive: 32-bit non-volatile design ID



Introduction

Provides internal access to the 32 non-volatile, user-programmable eFUSE bits.

Port Descriptions

Port	Direction	Width	Function
EFUSEUSR<31:0>	Output	32	User eFUSE register value output.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
SIM_EFUSE_VALUE	HEX	32'h00000000 to 32'hffffffff	32'h00000000	Value of the 32-bit non-volatile value used in simulation.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- EFUSE_USR: 32-bit non-volatile design ID
--            7 Series
-- Xilinx HDL Language Template, version 2020.1

EFUSE_USR_inst : EFUSE_USR
generic map (
```

```
   SIM_EFUSE_VALUE => X"00000000"  -- Value of the 32-bit non-volatile value used in simulation
)
port map (
   EFUSEUSR => EFUSEUSR  -- 32-bit output: User eFUSE register value output
);

-- End of EFUSE_USR_inst instantiation
```

Verilog Instantiation Template

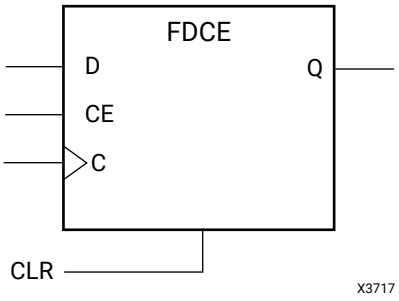
```
// EFUSE_USR: 32-bit non-volatile design ID
//            7 Series
// Xilinx HDL Language Template, version 2020.1

EFUSE_USR #(
   .SIM_EFUSE_VALUE(32'h00000000)  // Value of the 32-bit non-volatile value used in simulation
)
EFUSE_USR_inst (
   .EFUSEUSR(EFUSEUSR)  // 32-bit output: User eFUSE register value output
);

// End of EFUSE_USR_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configuration Guide* (UG470).

FDCE

Primitive: D Flip-Flop with Clock Enable and Asynchronous Clear



Introduction

This design element is a single D-type flip-flop with clock enable and asynchronous clear.

- When clock enable (CE) is High and asynchronous clear (CLR) is Low, the data on the data input (D) of this design element is transferred to the corresponding data output (Q) during the Low-to-High clock (C) transition.
- When CLR is High, it overrides all other inputs and resets the data output (Q) Low.
- When CE is Low, clock transitions are ignored.

This flip-flop is asynchronously cleared, outputs Low, when power is applied. Power-on conditions are simulated when global set/reset (GSR) is active. GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_E2 symbol.

Logic Table

Inputs				Outputs
CLR	**CE**	**D**	**C**	**Q**
1	X	X	X	0
0	0	X	X	No Change
0	1	D	↑	D

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	BINARY	1, 0	0	Sets the initial value of Q output after configuration.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- FDCE: Single Data Rate D Flip-Flop with Asynchronous Clear and
--       Clock Enable (posedge clk).
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

FDCE_inst : FDCE
generic map (
   INIT => '0') -- Initial value of register ('0' or '1')
port map (
   Q => Q,       -- Data output
   C => C,       -- Clock input
   CE => CE,     -- Clock enable input
   CLR => CLR,   -- Asynchronous clear input
   D => D        -- Data input
);

-- End of FDCE_inst instantiation
```

Verilog Instantiation Template

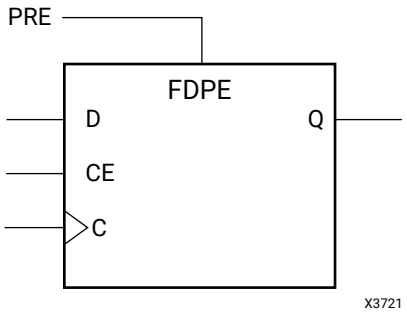
```
// FDCE: Single Data Rate D Flip-Flop with Asynchronous Clear and
//       Clock Enable (posedge clk).
//       7 Series
// Xilinx HDL Language Template, version 2020.1

FDCE #(
   .INIT(1'b0) // Initial value of register (1'b0 or 1'b1)
) FDCE_inst (
   .Q(Q),      // 1-bit Data output
   .C(C),      // 1-bit Clock input
   .CE(CE),    // 1-bit Clock enable input
   .CLR(CLR),  // 1-bit Asynchronous clear input
   .D(D)       // 1-bit Data input
);

// End of FDCE_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

FDPE

Primitive: D Flip-Flop with Clock Enable and Asynchronous Preset



Introduction

This design element is a single D-type flip-flop with data (D), clock enable (CE), and asynchronous preset (PRE) inputs and data output (Q). The asynchronous PRE, when High, overrides all other inputs and sets the (Q) output High. Data on the (D) input is loaded into the flip-flop when PRE is Low and CE is High on the Low-to-High clock (C) transition. When CE is Low, the clock transitions are ignored.

This flip-flop is asynchronously preset, outputs High, when power is applied.Power-on conditions are simulated when global set/reset (GSR) is active. GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_E2 symbol.

Logic Table

Inputs				Outputs
PRE	**CE**	**D**	**C**	**Q**
1	X	X	X	1
0	0	X	X	No Change
0	1	D	↑	D

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	BINARY	0, 1	1	Sets the initial value of Q output after configuration.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- FDPE: Single Data Rate D Flip-Flop with Asynchronous Preset and
--       Clock Enable (posedge clk).
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

FDPE_inst : FDPE
generic map (
   INIT => '0') -- Initial value of register ('0' or '1')
port map (
   Q => Q,       -- Data output
   C => C,       -- Clock input
   CE => CE,     -- Clock enable input
   PRE => PRE,   -- Asynchronous preset input
   D => D        -- Data input
);

-- End of FDPE_inst instantiation
```

Verilog Instantiation Template

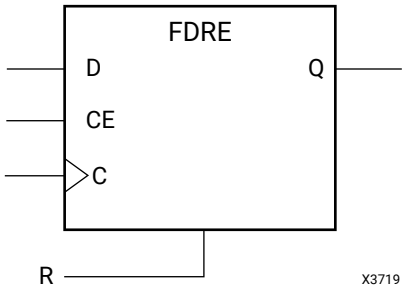
```
// FDPE: Single Data Rate D Flip-Flop with Asynchronous Preset and
//       Clock Enable (posedge clk).
//       7 Series
// Xilinx HDL Language Template, version 2020.1

FDPE #(
   .INIT(1'b0) // Initial value of register (1'b0 or 1'b1)
) FDPE_inst (
   .Q(Q),      // 1-bit Data output
   .C(C),      // 1-bit Clock input
   .CE(CE),    // 1-bit Clock enable input
   .PRE(PRE),  // 1-bit Asynchronous preset input
   .D(D)       // 1-bit Data input
);

// End of FDPE_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

FDRE

Primitive: D Flip-Flop with Clock Enable and Synchronous Reset



Introduction

This design element is a single D-type flip-flop with data (D), clock enable (CE), and synchronous reset (R) inputs and data output (Q). The synchronous reset (R) input, when High, overrides all other inputs and resets the (Q) output Low on the Low-to-High clock (C) transition. The data on the (D) input is loaded into the flip-flop when R is Low and CE is High during the Low-to-High clock transition.

This flip-flop is asynchronously cleared, outputs Low, when power is applied. Power-on conditions are simulated when global set/reset (GSR) is active. GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_E2 symbol.

Logic Table

Inputs				Outputs
R	**CE**	**D**	**C**	**Q**
1	X	X	↑	0
0	0	X	X	No Change
0	1	D	↑	D

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	BINARY	0, 1	0	Sets the initial value of Q output after configuration.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- FDRE: Single Data Rate D Flip-Flop with Synchronous Reset and
--       Clock Enable (posedge clk).
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

FDRE_inst : FDRE
generic map (
   INIT => '0') -- Initial value of register ('0' or '1')
port map (
   Q => Q,       -- Data output
   C => C,       -- Clock input
   CE => CE,     -- Clock enable input
   R => R,       -- Synchronous reset input
   D => D        -- Data input
);

-- End of FDRE_inst instantiation
```

Verilog Instantiation Template

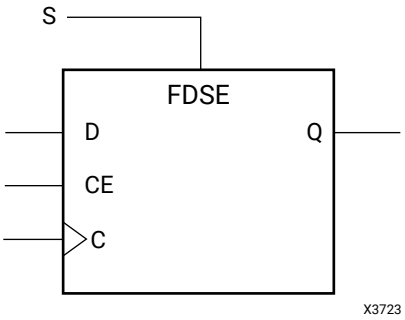
```
// FDRE: Single Data Rate D Flip-Flop with Synchronous Reset and
//       Clock Enable (posedge clk).
//       7 Series
// Xilinx HDL Language Template, version 2020.1

FDRE #(
   .INIT(1'b0) // Initial value of register (1'b0 or 1'b1)
) FDRE_inst (
   .Q(Q),      // 1-bit Data output
   .C(C),      // 1-bit Clock input
   .CE(CE),    // 1-bit Clock enable input
   .R(R),      // 1-bit Synchronous reset input
   .D(D)       // 1-bit Data input
);

// End of FDRE_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

FDSE

Primitive: D Flip-Flop with Clock Enable and Synchronous Set



Introduction

FDSE is a single D-type flip-flop with data (D), clock enable (CE), and synchronous set (S) inputs and data output (Q). The synchronous set (S) input, when High, overrides the clock enable (CE) input and sets the Q output High during the Low-to-High clock (C) transition. The data on the D input is loaded into the flip-flop when S is Low and CE is High during the Low-to-High clock (C) transition.

This flip-flop is asynchronously preset, outputs High, when power is applied.Power-on conditions are simulated when global set/reset (GSR) is active. GSR defaults to active-High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_E2 symbol.

Logic Table

Inputs				Outputs
S	**CE**	**D**	**C**	**Q**
1	X	X	↑	1
0	0	X	X	No Change
0	1	D	↑	D

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	BINARY	0, 1	1	Sets the initial value of Q output after configuration.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- FDSE: Single Data Rate D Flip-Flop with Synchronous Set and
--       Clock Enable (posedge clk).
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

FDSE_inst : FDSE
generic map (
   INIT => '0') -- Initial value of register ('0' or '1')
port map (
   Q => Q,       -- Data output
   C => C,       -- Clock input
   CE => CE,     -- Clock enable input
   S => S,       -- Synchronous Set input
   D => D        -- Data input
);

-- End of FDSE_inst instantiation
```

Verilog Instantiation Template

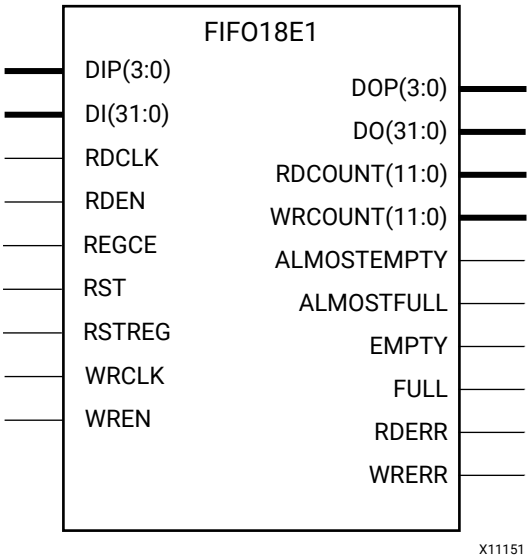
```
// FDSE: Single Data Rate D Flip-Flop with Synchronous Set and
//       Clock Enable (posedge clk).
//       7 Series
// Xilinx HDL Language Template, version 2020.1

FDSE #(
   .INIT(1'b0) // Initial value of register (1'b0 or 1'b1)
) FDSE_inst (
   .Q(Q),      // 1-bit Data output
   .C(C),      // 1-bit Clock input
   .CE(CE),    // 1-bit Clock enable input
   .S(S),      // 1-bit Synchronous set input
   .D(D)       // 1-bit Data input
);

// End of FDSE_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

FIFO18E1

Primitive: 18Kb FIFO (First-In-First-Out) Block RAM Memory



Introduction

7 series devices contain several block RAM memories, each of which can be separately configured as a FIFO, an automatic error-correction RAM, or as a general-purpose 36 Kb or 18 Kb RAM/ROM memory. These Block RAM memories offer fast and flexible storage of large amounts of on-chip data. The FIFO18E1 uses the FIFO control logic and the 18 Kb Block RAM. This primitive can be used in a 4-bit wide by 4K deep, 9-bit wide by 2K deep, 18-bit wide by 1K deep, or a 36-bit wide by 512 deep configuration. The primitive can be configured in either synchronous or dual-clock (asynchronous) mode, with all associated FIFO flags and status signals.

When using the dual-clock mode with independent clocks, depending on the offset between read and write clock edges, the Empty, Almost Empty, Full and Almost Full flags can deassert one cycle later. Due to the asynchronous nature of the clocks the simulation model only reflects the deassertion latency cycles listed in the User Guide.

***Note*:** For a 36-bit wide by 512 deep FIFO, the "FIFO18_36" mode must be used. For deeper or wider configurations of the FIFO, the FIFO36E1 can be used. If error-correction circuitry is desired, the FIFO36E1 with "FIFO36_72" mode must be used.

Port Descriptions

Port	Direction	Width	Function
ALMOSTEMPTY	Output	1	Programmable flag to indicate the FIFO is almost empty. The ALMOST_EMPTY_OFFSET attribute specifies the threshold where this flag is triggered relative to full/empty.
ALMOSTFULL	Output	1	Programmable flag to indicate that the FIFO is almost full. The ALMOST_FULL_OFFSET attribute specifies the threshold where this flag is triggered relative to full/empty.
DI<31:0>	Input	32	FIFO data input bus.
DIP<3:0>	Input	4	FIFO parity data input bus.
DO<31:0>	Output	32	FIFO data output bus.
DOP<3:0>	Output	4	FIFO parity data output bus.
EMPTY	Output	1	Active-High logic to indicate that the FIFO is currently empty.
FULL	Output	1	Active-High logic indicates that the FIFO is full.
RDCLK	Input	1	Rising edge read clock.
RDCOUNT<11:0>	Output	12	Read count.
RDEN	Input	1	Active-High FIFO read enable.
RDERR	Output	1	Read error occurred.
REGCE	Input	1	Output register clock enable for pipelined synchronous FIFO. DO_REG must be set to 1 if using this enable.
RST	Input	1	Active-High (FIFO logic) asynchronous reset (for dual-clock FIFO), synchronous reset (for synchronous FIFO). Must be held for a minimum of 5 WRCLK/RDCLK cycles.
RSTREG	Input	1	Output register synchronous set/reset. DO_REG must be set to 1 if using this reset.
WRCLK	Input	1	Rising edge write clock.
WRCOUNT<11:0>	Output	12	Write count.
WREN	Input	1	Active-High FIFO write enable.
WRERR	Output	1	Write error occurred. When the FIFO is full, any additional write operation generates an error flag. Synchronous with WRCLK.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	Recommended

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ALMOST_EMPTY _OFFSET	HEX	13'h0000 to 13'h1fff	13'h0080	Specifies the amount of data contents in the RAM to trigger the ALMOST_EMPTY flag.

Attribute	Type	Allowed Values	Default	Description
ALMOST_FULL _OFFSET	HEX	13'h0000 to 13'h1fff	13'h0080	Specifies the amount of data contents in the RAM to trigger the ALMOST_FULL flag.
DATA_WIDTH	DECIMAL	4, 9, 18, 36	4	Specifies the desired data width for the FIFO. ***Note*:** If set to 36, FIFO_MODE must be set to FIFO18_36.
DO_REG	DECIMAL	1, 0	1	Data pipeline register for EN_SYN.
EN_SYN	BOOLEAN	FALSE, TRUE	FALSE	EN_SYN denotes whether the FIFO is operating in either dual-clock (two independent clocks) or synchronous (a single clock) mode. Dual-clock must use DO_REG=1.
FIFO_MODE	STRING	"FIFO18", "FIFO18_36"	"FIFO18"	Selects "FIFO18" or "FIFO18_36" mode. ***Note*:** If set to "FIFO18_36", DATA_WIDTH must be set to 36.
FIRST_WORD_FALL _THROUGH	BOOLEAN	FALSE, TRUE	FALSE	If TRUE, the first write to the FIFO will appear on DO without a first RDEN assertion.
INIT	HEX	36 bit HEX	All zeros	Specifies the initial value on the DO output after configuration.
SIM_DEVICE	STRING	"7SERIES"	"7SERIES"	Must be set to "7SERIES" to exhibit proper simulation behavior under all conditions.
SRVAL	HEX	36 bit HEX	All zeros	Specifies the output value of the FIFO upon assertion of the synchronous reset (RSTREG) signal. Only valid for DO_REG=1.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- FIFO18E1: 18Kb FIFO (First-In-First-Out) Block RAM Memory
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

FIFO18E1_inst : FIFO18E1
generic map (
   ALMOST_EMPTY_OFFSET => X"0080",   -- Sets the almost empty threshold
   ALMOST_FULL_OFFSET => X"0080",    -- Sets almost full threshold
   DATA_WIDTH => 4,                  -- Sets data width to 4-36
   DO_REG => 1,                      -- Enable output register (1-0) Must be 1 if EN_SYN = FALSE
   EN_SYN => FALSE,                  -- Specifies FIFO as dual-clock (FALSE) or Synchronous (TRUE)
   FIFO_MODE => "FIFO18",            -- Sets mode to FIFO18 or FIFO18_36
   FIRST_WORD_FALL_THROUGH => FALSE, -- Sets the FIFO FWFT to FALSE, TRUE
   INIT => X"000000000",             -- Initial values on output port
   SIM_DEVICE => "7SERIES",          -- Must be set to "7SERIES" for simulation behavior
   SRVAL => X"000000000"             -- Set/Reset value for output port
)
port map (
   -- Read Data: 32-bit (each) output: Read output data
   DO => DO,                   -- 32-bit output: Data output
   DOP => DOP,                 -- 4-bit output: Parity data output
   -- Status: 1-bit (each) output: Flags and other FIFO status outputs
   ALMOSTEMPTY => ALMOSTEMPTY, -- 1-bit output: Almost empty flag
   ALMOSTFULL => ALMOSTFULL,   -- 1-bit output: Almost full flag
```

```
   EMPTY => EMPTY,             -- 1-bit output: Empty flag
   FULL => FULL,               -- 1-bit output: Full flag
   RDCOUNT => RDCOUNT,         -- 12-bit output: Read count
   RDERR => RDERR,             -- 1-bit output: Read error
   WRCOUNT => WRCOUNT,         -- 12-bit output: Write count
   WRERR => WRERR,             -- 1-bit output: Write error
   -- Read Control Signals: 1-bit (each) input: Read clock, enable and reset input signals
   RDCLK => RDCLK,             -- 1-bit input: Read clock
   RDEN => RDEN,               -- 1-bit input: Read enable
   REGCE => REGCE,             -- 1-bit input: Clock enable
   RST => RST,                 -- 1-bit input: Asynchronous Reset
   RSTREG => RSTREG,           -- 1-bit input: Output register set/reset
   -- Write Control Signals: 1-bit (each) input: Write clock and enable input signals
   WRCLK => WRCLK,             -- 1-bit input: Write clock
   WREN => WREN,               -- 1-bit input: Write enable
   -- Write Data: 32-bit (each) input: Write input data
   DI => DI,                   -- 32-bit input: Data input
   DIP => DIP                  -- 4-bit input: Parity input
);

-- End of FIFO18E1_inst instantiation
```

Verilog Instantiation Template

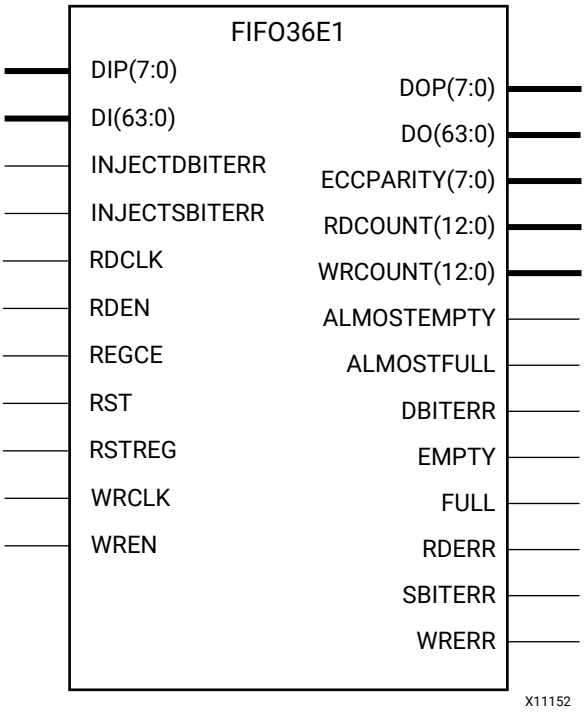
```
// FIFO18E1: 18Kb FIFO (First-In-First-Out) Block RAM Memory
//           7 Series
// Xilinx HDL Language Template, version 2020.1

FIFO18E1 #(
   .ALMOST_EMPTY_OFFSET(13'h0080),    // Sets the almost empty threshold
   .ALMOST_FULL_OFFSET(13'h0080),     // Sets almost full threshold
   .DATA_WIDTH(4),                    // Sets data width to 4-36
   .DO_REG(1),                        // Enable output register (1-0) Must be 1 if EN_SYN = FALSE
   .EN_SYN("FALSE"),                  // Specifies FIFO as dual-clock (FALSE) or Synchronous (TRUE)
   .FIFO_MODE("FIFO18"),              // Sets mode to FIFO18 or FIFO18_36
   .FIRST_WORD_FALL_THROUGH("FALSE"), // Sets the FIFO FWFT to FALSE, TRUE
   .INIT(36'h000000000),              // Initial values on output port
   .SIM_DEVICE("7SERIES"),            // Must be set to "7SERIES" for simulation behavior
   .SRVAL(36'h000000000)              // Set/Reset value for output port
)
FIFO18E1_inst (
   // Read Data: 32-bit (each) output: Read output data
   .DO(DO),                    // 32-bit output: Data output
   .DOP(DOP),                  // 4-bit output: Parity data output
   // Status: 1-bit (each) output: Flags and other FIFO status outputs
   .ALMOSTEMPTY(ALMOSTEMPTY), // 1-bit output: Almost empty flag
   .ALMOSTFULL(ALMOSTFULL),   // 1-bit output: Almost full flag
   .EMPTY(EMPTY),              // 1-bit output: Empty flag
   .FULL(FULL),                // 1-bit output: Full flag
   .RDCOUNT(RDCOUNT),          // 12-bit output: Read count
   .RDERR(RDERR),              // 1-bit output: Read error
   .WRCOUNT(WRCOUNT),          // 12-bit output: Write count
   .WRERR(WRERR),              // 1-bit output: Write error
   // Read Control Signals: 1-bit (each) input: Read clock, enable and reset input signals
   .RDCLK(RDCLK),              // 1-bit input: Read clock
   .RDEN(RDEN),                // 1-bit input: Read enable
   .REGCE(REGCE),              // 1-bit input: Clock enable
   .RST(RST),                  // 1-bit input: Asynchronous Reset
   .RSTREG(RSTREG),            // 1-bit input: Output register set/reset
   // Write Control Signals: 1-bit (each) input: Write clock and enable input signals
   .WRCLK(WRCLK),              // 1-bit input: Write clock
   .WREN(WREN),                // 1-bit input: Write enable
   // Write Data: 32-bit (each) input: Write input data
   .DI(DI),                    // 32-bit input: Data input
   .DIP(DIP)                   // 4-bit input: Parity input
);

// End of FIFO18E1_inst instantiation
```

For More Information

- See the *7 Series FPGAs Memory Resources User Guide* (UG473).

FIFO36E1

Primitive: 36 Kb FIFO (First-In-First-Out) Block RAM Memory



Introduction

7 series devices contain several block RAM memories that can be configured as FIFOs, automatic error-correction RAM, or general-purpose 36 Kb or 18 Kb RAM/ROM memories. These Block RAM memories offer fast and flexible storage of large amounts of on-chip data. The FIFO36E1 allows access to the Block RAM in the 36 Kb FIFO configurations. This component can be configured and used as a 4-bit wide by 8K deep, 9-bit by 4K deep, 18-bit by 2K deep, 36-bit wide by 1K deep, or 72-bit wide by 512 deep synchronous or dual-clock (asynchronous) FIFO RAM with all associated FIFO flags.

When using the dual-clock mode with independent clocks, depending on the offset between read and write clock edges, the Empty, Almost Empty, Full and Almost Full flags can deassert one cycle later. Due to the asynchronous nature of the clocks the simulation model only reflects the deassertion latency cycles listed in the User Guide.

***Note*:** For a 72-bit wide by 512 deep FIFO, the "FIFO36_72" mode must be used. For smaller configurations of the FIFO, the FIFO18E1 can be used. If error-correction circuitry is desired, the "FIFO36_72" mode must be used.

Port Descriptions

Port	Direction	Width	Function
ALMOSTEMPTY	Output	1	Programmable flag to indicate the FIFO is almost empty. The ALMOST_EMPTY_OFFSET attribute specifies where to trigger this flag.
ALMOSTFULL	Output	1	Programmable flag to indicate the FIFO is almost full. The ALMOST_FULL_OFFSET attribute specifies where to trigger this flag.
DBITERR	Output	1	Status output from ECC function to indicate a double bit error was detected. EN_ECC_READ needs to be TRUE to use this functionality.
DI<63:0>	Input	64	FIFO data input bus.
DIP<7:0>	Input	8	FIFO parity data input bus.
DO<63:0>	Output	64	FIFO data output bus.
DOP<7:0>	Output	8	FIFO parity data output bus.
ECCPARITY<7:0>	Output	8	8-bit data generated by the ECC encoder used by the ECC decoder for memory error detection and correction.
EMPTY	Output	1	Active-High logic to indicate that the FIFO is currently empty.
FULL	Output	1	Active-High logic indicates that the FIFO is full.
INJECTDBITERR	Input	1	Inject a double bit error if ECC feature is used.
INJECTSBITERR	Input	1	Inject a single bit error if ECC feature is used.
RDCLK	Input	1	Rising edge read clock.
RDCOUNT<12:0>	Output	13	Read count.
RDEN	Input	1	Active-High FIFO read enable.
RDERR	Output	1	Read error occurred.
REGCE	Input	1	Output register clock enable for pipelined synchronous FIFO. DO_REG must be 1 to use this enable.
RST	Input	1	Active-High (FIFO logic) asynchronous reset (for dual-clock FIFO), synchronous reset (synchronous FIFO) for 5 CLK cycles.
RSTREG	Input	1	Output register synchronous set/reset. DO_REG must be 1 to use this reset.
SBITERR	Output	1	Status output from ECC function to indicate a single bit error was detected. EN_ECC_READ needs to be TRUE to use this functionality.
WRCLK	Input	1	Write clock and enable input signals.
WRCOUNT<12:0>	Output	13	Write count.
WREN	Input	1	Active-High FIFO write enable.
WRERR	Output	1	Write error occurred. When the FIFO is full, any additional write operation generates an error flag. Synchronous with WRCLK.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	Recommended

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ALMOST_EMPTY _OFFSET	HEX	13'h0000 to 13'h1fff	13'h0080	Specifies the amount of data contents in the RAM to trigger the ALMOST_EMPTY flag.
ALMOST_FULL _OFFSET	HEX	13'h0000 to 13'h1fff	13'h0080	Specifies the amount of data contents in the RAM to trigger the ALMOST_FULL flag.
DATA_WIDTH	DECIMAL	4, 9, 18, 36, 72	4	Specifies the desired data width for the FIFO. For data widths of 72, FIFO_MODE must be set to "FIFO36_72."
DO_REG	DECIMAL	1, 0	1	Enable output register to the FIFO for improved clock-to-out timing at the expense of added read latency (one pipeline delay). DO_REG must be 1 when EN_SYN is set to FALSE.
EN_ECC_READ	BOOLEAN	FALSE, TRUE	FALSE	Enable the ECC decoder circuitry.
EN_ECC_WRITE	BOOLEAN	FALSE, TRUE	FALSE	Enable the ECC encoder circuitry.
EN_SYN	BOOLEAN	FALSE, TRUE	FALSE	When FALSE, specifies the FIFO to be used in asynchronous mode (two independent clock) or when TRUE in synchronous (a single clock) operation.
FIFO_MODE	STRING	"FIFO36", "FIFO36_72"	"FIFO36"	Selects regular "FIFO36" or the wide "FIFO36_72" mode. If set to "FIFO36_72", the DATA_WIDTH attribute has to be 72.
FIRST_WORD_FALL _THROUGH	BOOLEAN	FALSE, TRUE	FALSE	If TRUE, the first write to the FIFO will appear on DO without an RDEN assertion.
INIT	HEX	72 bit HEX	All zeros	Specifies the initial value on the DO output after configuration.
SIM_DEVICE	STRING	"7SERIES"	"7SERIES"	Must be set to "7SERIES" to exhibit proper simulation behavior under all conditions.
SRVAL	HEX	72 bit HEX	All zeros	Specifies the output value of the FIFO upon assertion of the synchronous reset (RSTREG) signal. Only valid for DO_REG=1.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- FIFO36E1: 36Kb FIFO (First-In-First-Out) Block RAM Memory
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

FIFO36E1_inst : FIFO36E1
generic map (
   ALMOST_EMPTY_OFFSET => X"0080",   -- Sets the almost empty threshold
   ALMOST_FULL_OFFSET => X"0080",    -- Sets almost full threshold
   DATA_WIDTH => 4,                  -- Sets data width to 4-72
   DO_REG => 1,                      -- Enable output register (1-0) Must be 1 if EN_SYN = FALSE
   EN_ECC_READ => FALSE,             -- Enable ECC decoder, FALSE, TRUE
   EN_ECC_WRITE => FALSE,            -- Enable ECC encoder, FALSE, TRUE
   EN_SYN => FALSE,                  -- Specifies FIFO as Asynchronous (FALSE) or Synchronous (TRUE)
   FIFO_MODE => "FIFO36",            -- Sets mode to "FIFO36" or "FIFO36_72"
   FIRST_WORD_FALL_THROUGH => FALSE, -- Sets the FIFO FWFT to FALSE, TRUE
   INIT => X"000000000000000000",    -- Initial values on output port
   SIM_DEVICE => "7SERIES",          -- Must be set to "7SERIES" for simulation behavior
   SRVAL => X"000000000000000000"    -- Set/Reset value for output port
)
port map (
   -- ECC Signals: 1-bit (each) output: Error Correction Circuitry ports
   DBITERR => DBITERR,               -- 1-bit output: Double bit error status
   ECCPARITY => ECCPARITY,           -- 8-bit output: Generated error correction parity
   SBITERR => SBITERR,               -- 1-bit output: Single bit error status
   -- Read Data: 64-bit (each) output: Read output data
   DO => DO,                         -- 64-bit output: Data output
   DOP => DOP,                       -- 8-bit output: Parity data output
   -- Status: 1-bit (each) output: Flags and other FIFO status outputs
   ALMOSTEMPTY => ALMOSTEMPTY,       -- 1-bit output: Almost empty flag
   ALMOSTFULL => ALMOSTFULL,         -- 1-bit output: Almost full flag
   EMPTY => EMPTY,                   -- 1-bit output: Empty flag
   FULL => FULL,                     -- 1-bit output: Full flag
   RDCOUNT => RDCOUNT,               -- 13-bit output: Read count
   RDERR => RDERR,                   -- 1-bit output: Read error
   WRCOUNT => WRCOUNT,               -- 13-bit output: Write count
   WRERR => WRERR,                   -- 1-bit output: Write error
   -- ECC Signals: 1-bit (each) input: Error Correction Circuitry ports
   INJECTDBITERR => INJECTDBITERR, -- 1-bit input: Inject a double bit error input
   INJECTSBITERR => INJECTSBITERR,
   -- Read Control Signals: 1-bit (each) input: Read clock, enable and reset input signals
   RDCLK => RDCLK,                   -- 1-bit input: Read clock
   RDEN => RDEN,                     -- 1-bit input: Read enable
   REGCE => REGCE,                   -- 1-bit input: Clock enable
   RST => RST,                       -- 1-bit input: Reset
   RSTREG => RSTREG,                 -- 1-bit input: Output register set/reset
   -- Write Control Signals: 1-bit (each) input: Write clock and enable input signals
   WRCLK => WRCLK,                   -- 1-bit input: Rising edge write clock.
   WREN => WREN,                     -- 1-bit input: Write enable
   -- Write Data: 64-bit (each) input: Write input data
   DI => DI,                         -- 64-bit input: Data input
   DIP => DIP                        -- 8-bit input: Parity input
);

-- End of FIFO36E1_inst instantiation
```

Verilog Instantiation Template

```
// FIFO36E1: 36Kb FIFO (First-In-First-Out) Block RAM Memory
//           7 Series
// Xilinx HDL Language Template, version 2020.1

FIFO36E1 #(
   .ALMOST_EMPTY_OFFSET(13'h0080),    // Sets the almost empty threshold
```

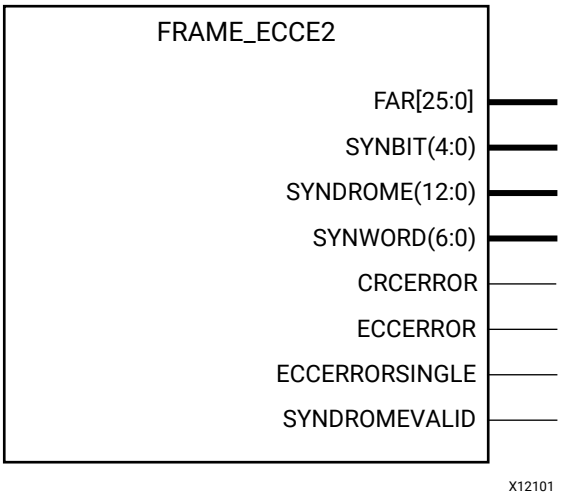
```
   .ALMOST_FULL_OFFSET(13'h0080),     // Sets almost full threshold
   .DATA_WIDTH(4),                    // Sets data width to 4-72
   .DO_REG(1),                        // Enable output register (1-0) Must be 1 if EN_SYN = FALSE
   .EN_ECC_READ("FALSE"),             // Enable ECC decoder, FALSE, TRUE
   .EN_ECC_WRITE("FALSE"),            // Enable ECC encoder, FALSE, TRUE
  .EN_SYN("FALSE"),                // Specifies FIFO as Asynchronous (FALSE) or Synchronous (TRUE)
   .FIFO_MODE("FIFO36"),              // Sets mode to "FIFO36" or "FIFO36_72"
   .FIRST_WORD_FALL_THROUGH("FALSE"), // Sets the FIFO FWFT to FALSE, TRUE
   .INIT(72'h000000000000000000),     // Initial values on output port
   .SIM_DEVICE("7SERIES"),            // Must be set to "7SERIES" for simulation behavior
   .SRVAL(72'h000000000000000000)     // Set/Reset value for output port
)
FIFO36E1_inst (
   // ECC Signals: 1-bit (each) output: Error Correction Circuitry ports
   .DBITERR(DBITERR),             // 1-bit output: Double bit error status
   .ECCPARITY(ECCPARITY),         // 8-bit output: Generated error correction parity
   .SBITERR(SBITERR),             // 1-bit output: Single bit error status
   // Read Data: 64-bit (each) output: Read output data
   .DO(DO),                       // 64-bit output: Data output
   .DOP(DOP),                     // 8-bit output: Parity data output
   // Status: 1-bit (each) output: Flags and other FIFO status outputs
   .ALMOSTEMPTY(ALMOSTEMPTY),     // 1-bit output: Almost empty flag
   .ALMOSTFULL(ALMOSTFULL),       // 1-bit output: Almost full flag
   .EMPTY(EMPTY),                 // 1-bit output: Empty flag
   .FULL(FULL),                   // 1-bit output: Full flag
   .RDCOUNT(RDCOUNT),             // 13-bit output: Read count
   .RDERR(RDERR),                 // 1-bit output: Read error
   .WRCOUNT(WRCOUNT),             // 13-bit output: Write count
   .WRERR(WRERR),                 // 1-bit output: Write error
   // ECC Signals: 1-bit (each) input: Error Correction Circuitry ports
   .INJECTDBITERR(INJECTDBITERR), // 1-bit input: Inject a double bit error input
   .INJECTSBITERR(INJECTSBITERR),
   // Read Control Signals: 1-bit (each) input: Read clock, enable and reset input signals
   .RDCLK(RDCLK),                 // 1-bit input: Read clock
   .RDEN(RDEN),                   // 1-bit input: Read enable
   .REGCE(REGCE),                 // 1-bit input: Clock enable
   .RST(RST),                     // 1-bit input: Reset
   .RSTREG(RSTREG),               // 1-bit input: Output register set/reset
   // Write Control Signals: 1-bit (each) input: Write clock and enable input signals
   .WRCLK(WRCLK),                 // 1-bit input: Rising edge write clock.
   .WREN(WREN),                   // 1-bit input: Write enable
   // Write Data: 64-bit (each) input: Write input data
   .DI(DI),                       // 64-bit input: Data input
   .DIP(DIP)                      // 8-bit input: Parity input
);

// End of FIFO36E1_inst instantiation
```

For More Information

- See the *7 Series FPGAs Memory Resources User Guide* (UG473).

FRAME_ECCE2

Primitive: Configuration Frame Error Correction



Introduction

This design element enables the dedicated, built-in Error Correction Code (ECC) for the configuration memory of the FPGA. This element contains outputs that allow monitoring of the status of the ECC circuitry and the status of the readback CRC circuitry.

Port Descriptions

Port	Direction	Width	Function
CRCERROR	Output	1	Output indicating a CRC error.
ECCERROR	Output	1	Output indicating an ECC error.
ECCERRORSINGLE	Output	1	Output Indicating single-bit Frame ECC error detected.
FAR<25:0>	Output	26	Frame Address Register Value output.
SYNBIT<4:0>	Output	5	Output bit address of error.
SYNDROME<12:0>	Output	13	Output location of erroneous bit.
SYNDROMEVALID	Output	1	Frame ECC output indicating the SYNDROME output is valid.
SYNWORD<6:0>	Output	7	Word output in the frame where an ECC error has been detected.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
FARSRC	STRING	"EFAR", "FAR"	"EFAR"	Sedts whether the output of the FAR[25:0] configuration register points to the FAR or EFAR. Sets configuration option register bit CTL0[7].
FRAME_RBT_IN _FILENAME	STRING	String representing file name and location	“NONE”	This file is output by the ICAP_E2 model and it contains Frame Data information for the Raw Bitstream (RBT) file. The FRAME_ECCE2 model will parse this file, calculate ECC and output any error conditions.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- FRAME_ECCE2: Configuration Frame Error Correction
--              7 Series
-- Xilinx HDL Language Template, version 2020.1

FRAME_ECCE2_inst : FRAME_ECCE2
generic map (
   FARSRC => "EFAR",               -- Determines if the output of FAR[25:0] configuration register points
                                   -- to the FAR or EFAR. Sets configuration option register bit CTL0[7].
   FRAME_RBT_IN_FILENAME => "NONE"  -- This file is output by the ICAP_E2 model and it contains Frame Data
                                   -- information for the Raw Bitstream (RBT) file. The FRAME_ECCE2 model
                                   -- will parse this file, calculate ECC and output any error conditions.
)
port map (
   CRCERROR => CRCERROR,              -- 1-bit output: Output indicating a CRC error.
   ECCERROR => ECCERROR,              -- 1-bit output: Output indicating an ECC error.
   ECCERRORSINGLE => ECCERRORSINGLE, -- 1-bit output: Output Indicating single-bit Frame ECC error detected.
   FAR => FAR,                        -- 26-bit output: Frame Address Register Value output.
   SYNBIT => SYNBIT,                  -- 5-bit output: Output bit address of error.
   SYNDROME => SYNDROME,              -- 13-bit output: Output location of erroneous bit.
   SYNDROMEVALID => SYNDROMEVALID,   -- 1-bit output: Frame ECC output indicating the SYNDROME output is
                                      -- valid.

   SYNWORD => SYNWORD                -- 7-bit output: Word output in the frame where an ECC error has been
                                      -- detected.

);

-- End of FRAME_ECCE2_inst instantiation
```

Verilog Instantiation Template

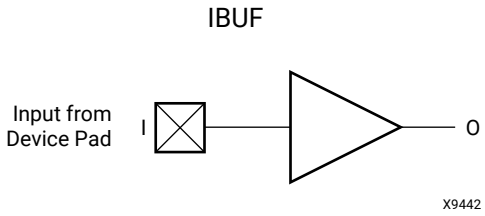
```
// FRAME_ECCE2: Configuration Frame Error Correction
//              7 Series
// Xilinx HDL Language Template, version 2020.1

FRAME_ECCE2 #(
   .FARSRC("EFAR"),               // Determines if the output of FAR[25:0] configuration register points to
                                  // the FAR or EFAR. Sets configuration option register bit CTL0[7].
   .FRAME_RBT_IN_FILENAME("NONE")  // This file is output by the ICAP_E2 model and it contains Frame Data
                                  // information for the Raw Bitstream (RBT) file. The FRAME_ECCE2 model
                                  // will parse this file, calculate ECC and output any error conditions.
)
FRAME_ECCE2_inst (
    .CRCERROR(CRCERROR),              // 1-bit output: Output indicating a CRC error.
    .ECCERROR(ECCERROR),              // 1-bit output: Output indicating an ECC error.
   .ECCERRORSINGLE(ECCERRORSINGLE), // 1-bit output: Output Indicating single-bit Frame ECC error detected.
    .FAR(FAR),                        // 26-bit output: Frame Address Register Value output.
    .SYNBIT(SYNBIT),                  // 5-bit output: Output bit address of error.
    .SYNDROME(SYNDROME),              // 13-bit output: Output location of erroneous bit.
   .SYNDROMEVALID(SYNDROMEVALID),   // 1-bit output: Frame ECC output indicating the SYNDROME output is
                                     // valid.

   .SYNWORD(SYNWORD)                // 7-bit output: Word output in the frame where an ECC error has been
                                     // detected.

);

// End of FRAME_ECCE2_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configuration Guide* (UG470).

IBUF

Primitive: Input Buffer



Introduction

This design element is automatically inserted (inferred) by the synthesis tool to any signal directly connected to a top-level input or in-out port of the design. You should generally let the synthesis tool infer this buffer. However, it can be instantiated into the design if required. In order to do so, connect the input port (I) directly to the associated top-level input or in-out port, and connect the output port (O) to the logic sourced by that port. Modify any necessary generic maps (VHDL) or named parameter value assignment (Verilog) to change the default behavior of the component.

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Buffer output.
I	Input	1	Buffer input.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

In general, IBUFs are inferred by the synthesis tool for specified top-level input ports to the design, so it is not necessary to specify them in the source code. However, if desired, they can be manually instantiated by copying the instantiation code from the appropriate Libraries Guide HDL template and pasting it into the top-level entity/module of your code. You should always put all I/O components on the top-level of the design to help facilitate hierarchical design methods. Connect the I port directly to the top-level input port of the design and the O port to the logic in which this input is to source. Specify the desired generic/defparam values to configure the proper behavior of the buffer.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
IBUF_LOW_PWR	BOOLEAN	TRUE, FALSE	TRUE	When set to TRUE, allows for reduced power when using differential or referenced (requiring V_{REF}) input standards like LVDS or HSTL. A setting of FALSE demands more power but delivers higher performance characteristics.Consult the*7 Series FPGA SelectIO Resources User Guide* (UG471) for details.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IBUF: Single-ended Input Buffer
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

IBUF_inst : IBUF
generic map (
   IBUF_LOW_PWR => TRUE, -- Low power (TRUE) vs. performance (FALSE) setting for referenced I/O standards
   IOSTANDARD => "DEFAULT")
port map (
   O => O,      -- Buffer output
   I => I       -- Buffer input (connect directly to top-level port)
);

-- End of IBUF_inst instantiation
```

Verilog Instantiation Template

```
// IBUF: Single-ended Input Buffer
//       7 Series
// Xilinx HDL Language Template, version 2020.1

IBUF #(
   .IBUF_LOW_PWR("TRUE"),  // Low power (TRUE) vs. performance (FALSE) setting for referenced I/O standards
   .IOSTANDARD("DEFAULT")  // Specify the input I/O standard
) IBUF_inst (
   .O(O),      // Buffer output
   .I(I)       // Buffer input (connect directly to top-level port)
);

// End of IBUF_inst instantiation
```

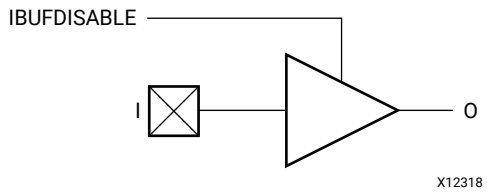
For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IBUF_IBUFDISABLE

Primitive: Single-ended Input Buffer with Input Disable

IBUF_IBUFDISABLE



Introduction

This design element is an input buffer used to connect internal logic to an external pin. This element includes an input path disable as an additional power saving feature when the I/O is not used for a sustained amount of time.

Port Descriptions

Port	Direction	Width	Function
I	Input	1	Input port connection. Connect directly to top-level port in the design.
IBUFDISABLE	Input	1	Disables input path through the buffer and forces to a logic high when USE_IBUFDISABLE is set to "TRUE" and this signal is asserted high. If USE_IBUFDISABLE is set to "FALSE" this input is ignored and should be tied to ground. This feature is generally used to reduce power at times when the I/O is idle for a period of time.
O	Output	1	Buffer output representing the input path to the device.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
IBUF_LOW_PWR	STRING	"TRUE", "FALSE"	"TRUE"	Allows a trade off of lower power consumption versus highest performance when referenced I/O standards are used.

Attribute	Type	Allowed Values	Default	Description
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
USE_IBUFDISABLE	STRING	"TRUE", "FALSE"	"TRUE"	Enables or disables the feature of IBUFDISABLE.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IBUF_IBUFDISABLE: Single-ended Input Buffer with Disable
--                   7 Series
-- Xilinx HDL Language Template, version 2020.1

IBUF_IBUFDISABLE_inst : IBUF_IBUFDISABLE
generic map (
   IBUF_LOW_PWR => "TRUE", -- Low power (TRUE) vs. performance (FALSE) setting for referenced I/O standards
   IOSTANDARD => "DEFAULT", -- Specify the input I/O standard
   USE_IBUFDISABLE => "TRUE") -- Set to "TRUE" to enable IBUFDISABLE feature
port map (
   O => O,      -- Buffer output
   I => I,      -- Buffer input (connect directly to top-level port)
   IBUFDISABLE => IBUFDISABLE -- Buffer disable input, low=disable
);

-- End of IBUF_IBUFDISABLE_inst instantiation
```

Verilog Instantiation Template

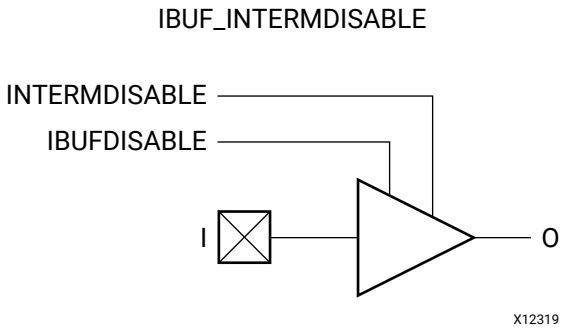
```
// IBUF_IBUFDISABLE: Single-ended Input Buffer with Disable
//                   7 Series
// Xilinx HDL Language Template, version 2020.1

IBUF_IBUFDISABLE #(
   .IBUF_LOW_PWR("TRUE"),   // Low power ("TRUE") vs. performance ("FALSE") for referenced I/O standards
   .IOSTANDARD("DEFAULT"),  // Specify the input I/O standard
   .USE_IBUFDISABLE("TRUE") // Set to "TRUE" to enable IBUFDISABLE feature
) IBUF_IBUFDISABLE_inst (
   .O(O),                   // Buffer output
   .I(I),                   // Buffer input (connect directly to top-level port)
   .IBUFDISABLE(IBUFDISABLE) // Buffer disable input, high=disable
);

// End of IBUF_IBUFDISABLE_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IBUF_INTERMDISABLE

Primitive: Single-ended Input Buffer with Input Termination Disable and Input Disable



Introduction

This design element is an input buffer used to connect internal logic to an external pin. This element includes an input termination (INTERM) enable/disable as well as an input path disable as additional power saving features when the I/O is not being used for a sustained amount of time.

Port Descriptions

Port	Direction	Width	Function
I	Input	1	Input port connection. Connect directly to top-level port in the design.
IBUFDISABLE	Input	1	Disables input path through the buffer and forces to a logic high when USE_IBUFDISABLE is set to "TRUE" and this signal is asserted high. If USE_IBUFDISABLE is set to "FALSE" this input is ignored and should be tied to ground. This feature is generally used to reduce power at times when the I/O is idle for a period of time.
INTERMDISABLE	Input	1	Disables input termination. This feature is generally used to reduce power at times when the I/O is idle.
O	Output	1	Buffer output representing the input path to the device.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
IBUF_LOW_PWR	STRING	"TRUE", "FALSE"	"TRUE"	Allows a trade off of lower power consumption vs. highest performance when referenced I/O standards are used.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
USE_IBUFDISABLE	STRING	"TRUE", "FALSE"	"TRUE"	Enables or disables the feature of IBUFDISABLE.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IBUF_INTERMDISABLE: Single-ended Input Buffer with Termination Input Disable
--                     May only be placed in High Range (HR) Banks
--                     7 Series
-- Xilinx HDL Language Template, version 2020.1

IBUF_INTERMDISABLE_inst : IBUF_INTERMDISABLE
generic map (
   IBUF_LOW_PWR => "TRUE", -- Low power (TRUE) vs. performance (FALSE) setting for referenced I/O standards
   IOSTANDARD => "DEFAULT", -- Specify the input I/O standard
   USE_IBUFDISABLE => "TRUE") -- Set to "TRUE" to enable IBUFDISABLE feature
port map (
   O => O,      -- Buffer output
   I => I,      -- Buffer input (connect directly to top-level port)
   INTERMDISABLE => INTERMDISABLE, -- Input Termination Disable
   IBUFDISABLE => IBUFDISABLE -- Buffer disable input, low=disable
);

-- End of IBUF_INTERMDISABLE_inst instantiation
```

Verilog Instantiation Template

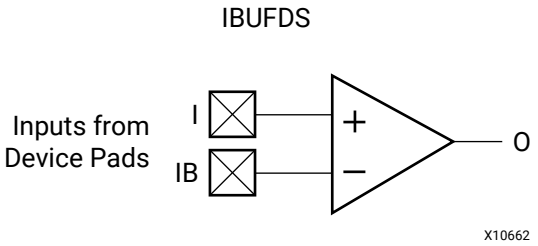
```
// IBUF_INTERMDISABLE: Single-ended Input Buffer with Termination Input Disable
//                     May only be placed in High Range (HR) Banks
//                     7 Series
// Xilinx HDL Language Template, version 2020.1

IBUF_INTERMDISABLE #(
   .IBUF_LOW_PWR("TRUE"),   // Low power ("TRUE") vs. performance ("FALSE") for referenced I/O standards
   .IOSTANDARD("DEFAULT"),  // Specify the input I/O standard
   .USE_IBUFDISABLE("TRUE") // Set to "TRUE" to enable IBUFDISABLE feature
) IBUF_INTERMDISABLE_inst (
   .O(O),                     // Buffer output
   .I(I),                     // Buffer input (connect directly to top-level port)
   .IBUFDISABLE(IBUFDISABLE),    // Buffer disable input, high=disable
   .INTERMDISABLE(INTERMDISABLE) // Input Termination Disable
);

// End of IBUF_INTERMDISABLE_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IBUFDS

Primitive: Differential Signaling Input Buffer



Introduction

This design element is an input buffer that supports low-voltage, differential signaling. In IBUFDS, a design level interface signal is represented as two distinct ports (I and IB), one deemed the "master" and the other the "slave." The master and the slave are opposite phases of the same logical signal (for example, MYNET_P and MYNET_N). Optionally, a programmable differential termination feature is available to help improve signal integrity and reduce external components.

Logic Table

Inputs		Outputs
I	**IB**	**O**
0	0	No Change
0	1	0
1	0	1
1	1	No Change

Port Descriptions

Port	Direction	Width	Function
I	Input	1	Diff_p Buffer Input.
IB	Input	1	Diff_n Buffer Input.
O	Output	1	Buffer Output.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Put all I/O components on the top-level of the design to help facilitate hierarchical design methods. Connect the I port directly to the top-level "master" input port of the design, the IB port to the top-level "slave" input port, and the O port to the logic in which this input is to source. Specify the desired generic/defparam values to configure the proper behavior of the buffer.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	BOOLEAN	TRUE, FALSE	FALSE	Turns the built-in differential termination on (TRUE) or off (FALSE).
IBUF_LOW_PWR	BOOLEAN	TRUE, FALSE	TRUE	When set to TRUE, allows for reduced power when using differential or referenced (requiring V_{REF}) input standards like LVDS or HSTL. A setting of FALSE demands more power but delivers higher performance characteristics. Consult the *7 Series FPGA SelectIO Resources User Guide* (UG471) for details.
IOSTANDARD	STRING	See Data Sheet.	"DEFAULT"	Assigns an I/O standard to the element.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IBUFDS: Differential Input Buffer
--         7 Series
-- Xilinx HDL Language Template, version 2020.1

IBUFDS_inst : IBUFDS
generic map (
   DIFF_TERM => FALSE, -- Differential Termination
   IBUF_LOW_PWR => TRUE, -- Low power (TRUE) vs. performance (FALSE) setting for referenced I/O standards
   IOSTANDARD => "DEFAULT")
port map (
   O => O,  -- Buffer output
   I => I,  -- Diff_p buffer input (connect directly to top-level port)
   IB => IB -- Diff_n buffer input (connect directly to top-level port)
);

-- End of IBUFDS_inst instantiation
```

Verilog Instantiation Template

```
// IBUFDS: Differential Input Buffer
//         7 Series
// Xilinx HDL Language Template, version 2020.1

IBUFDS #(
   .DIFF_TERM("FALSE"),       // Differential Termination
   .IBUF_LOW_PWR("TRUE"),     // Low power="TRUE", Highest performance="FALSE"
   .IOSTANDARD("DEFAULT")     // Specify the input I/O standard
) IBUFDS_inst (
   .O(O),  // Buffer output
```

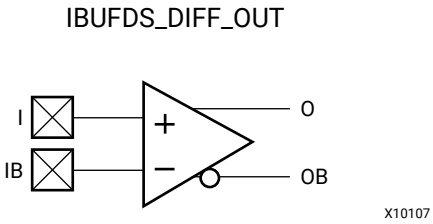
```
   .I(I),  // Diff_p buffer input (connect directly to top-level port)
   .IB(IB) // Diff_n buffer input (connect directly to top-level port)
);

// End of IBUFDS_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IBUFDS_DIFF_OUT

Primitive: Differential Signaling Input Buffer With Differential Output



Introduction

This design element is an input buffer that supports differential signaling. In IBUFDS_DIFF_OUT, a design level interface signal is represented as two distinct ports (I and IB), one deemed the "master" and the other the "slave." The master and the slave are opposite phases of the same logical signal (for example, MYNET_P and MYNET_N). The IBUFDS_DIFF_OUT differs from the IBUFDS in that it allows internal access to both phases of the differential signal. Optionally, a programmable differential termination feature is available to help improve signal integrity and reduce external components.

Logic Table

Inputs		Outputs	
I	**IB**	**O**	**OB**
0	0	No Change	No Change
0	1	0	1
1	0	1	0
1	1	No Change	No Change

Port Descriptions

Port	Direction	Width	Function
I	Input	1	Diff_p Buffer Input (connect to top-level port in the design).
IB	Input	1	Diff_n Buffer Input (connect to top-level port in the design).
O	Output	1	Diff_p Buffer Output.
OB	Output	1	Diff_n Buffer Output.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

It is suggested to put all I/O components on the top-level of the design to help facilitate hierarchical design methods. Connect the I port directly to the top-level "master" input port of the design, the IB port to the top-level "slave" input port, and the O and OB ports to the logic in which this input is to source. Specify the desired generic/parameter values to configure the proper behavior of the buffer.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	BOOLEAN	TRUE, FALSE	FALSE	Turns the built-in differential termination on (TRUE) or off (FALSE).
IBUF_LOW_PWR	BOOLEAN	TRUE, FALSE	TRUE	When set to TRUE, allows for reduced power when using differential or referenced (requiring V_{REF}) input standards like LVDS or HSTL. A setting of FALSE demands more power but delivers higher performance characteristics. Consult the *7 Series FPGA SelectIO Resources User Guide* (UG471) for details.
IOSTANDARD	STRING	See Data Sheet.	"DEFAULT"	Assigns an I/O standard to the element.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IBUFDS_DIFF_OUT: Differential Input Buffer with Differential Output
--                  7 Series
-- Xilinx HDL Language Template, version 2020.1

IBUFDS_DIFF_OUT_inst : IBUFDS_DIFF_OUT
generic map (
   DIFF_TERM => FALSE, -- Differential Termination
   IBUF_LOW_PWR => TRUE, -- Low power (TRUE) vs. performance (FALSE) setting for referenced I/O standards
   IOSTANDARD => "DEFAULT") -- Specify the input I/O standard
port map (
   O => O,      -- Buffer diff_p output
   OB => OB,    -- Buffer diff_n output
   I => I,  -- Diff_p buffer input (connect directly to top-level port)
   IB => IB -- Diff_n buffer input (connect directly to top-level port)
);

-- End of IBUFDS_DIFF_OUT_inst instantiation
```

Verilog Instantiation Template

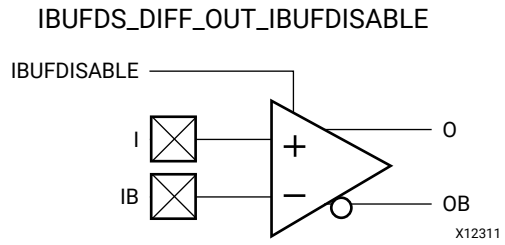
```
// IBUFDS_DIFF_OUT: Differential Input Buffer with Differential Output
//                  7 Series
// Xilinx HDL Language Template, version 2020.1

IBUFDS_DIFF_OUT #(
   .DIFF_TERM("FALSE"),   // Differential Termination, "TRUE"/"FALSE"
   .IBUF_LOW_PWR("TRUE"), // Low power="TRUE", Highest performance="FALSE"
   .IOSTANDARD("DEFAULT") // Specify the input I/O standard
) IBUFDS_DIFF_OUT_inst (
   .O(O),   // Buffer diff_p output
   .OB(OB), // Buffer diff_n output
   .I(I),   // Diff_p buffer input (connect directly to top-level port)
   .IB(IB)  // Diff_n buffer input (connect directly to top-level port)
);

// End of IBUFDS_DIFF_OUT_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IBUFDS_DIFF_OUT_IBUFDISABLE

Primitive: Input Differential Buffer with Input Disable and Differential Output



Introduction

This design element is a differential input buffer used to connect internal logic to an external bidirectional pin. This element includes an input path disable as an additional power saving feature when the input is idle for a sustained time. The IOBUFDS_DIFF_OUT_IBUFDISABLE differs from the IOBUFDS_IBUFDISABLE in that it allows internal access to both phases of the differential signal.

Port Descriptions

Port	Direction	Width	Function
I	Input	1	Input p-side port connection. Connect directly to top-level port in the design.
IB	Input	1	Input n-side port connection. Connect directly to top-level port in the design.
IBUFDISABLE	Input	1	Disables input path through the buffer and forces to a logic high when USE_IBUFDISABLE is set to "TRUE" and this signal is asserted high. If USE_IBUFDISABLE is set to "FALSE" this input is ignored and should be tied to ground. This feature is generally used to reduce power at times when the I/O is idle for a period of time.
O	Output	1	Buffer p-side output representing the input path to the device.
OB	Output	1	Buffer n-side output representing the input path to the device.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	STRING	"TRUE", "FALSE"	"FALSE"	Turns the built-in differential termination on (TRUE) or off (FALSE).
IBUF_LOW_PWR	STRING	"TRUE", "FALSE"	"TRUE"	Allows a trade off of lower power consumption vs. highest performance when referenced I/O standards are used.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
USE_IBUFDISABLE	STRING	"TRUE", "FALSE"	"TRUE"	Enables or disables the feature of IBUFDISABLE.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IBUFDS_DIFF_OUT_IBUFDISABLE: Differential Input Buffer with Differential Output w/ Disable
--                              7 Series
-- Xilinx HDL Language Template, version 2020.1

IBUFDS_DIFF_OUT_IBUFDISABLE_inst : IBUFDS_DIFF_OUT_IBUFDISABLE
generic map (
   DIFF_TERM => "FALSE", -- Differential Termination
   IBUF_LOW_PWR => "TRUE", -- Low power "TRUE" vs. performance "FALSE" setting for referenced I/O standards
   IOSTANDARD => "DEFAULT", -- Specify the input I/O standard
   USE_IBUFDISABLE => "TRUE") -- Set to "TRUE" to enable IBUFDISABLE feature
port map (
   O => O,      -- Buffer diff_p output
   OB => OB,    -- Buffer diff_n output
   I => I,  -- Diff_p buffer input (connect directly to top-level port)
   IB => IB, -- Diff_n buffer input (connect directly to top-level port)
   IBUFDISABLE => IBUFDISABLE -- Buffer disable input, low=disable
);

-- End of IBUFDS_DIFF_OUT_IBUFDISABLE_inst instantiation
```

Verilog Instantiation Template

```
// IBUFDS_DIFF_OUT_IBUFDISABLE: Differential Input Buffer with Differential Output with Input Disable
//                              7 Series
// Xilinx HDL Language Template, version 2020.1

IBUFDS_DIFF_OUT_IBUFDISABLE #(
   .DIFF_TERM("FALSE"),     // Differential Termination, "TRUE"/"FALSE"
   .IBUF_LOW_PWR("TRUE"),   // Low power="TRUE", Highest performance="FALSE"
   .IOSTANDARD("DEFAULT"),  // Specify the input I/O standard
   .USE_IBUFDISABLE("TRUE") // Set to "TRUE" to enable IBUFDISABLE feature
) IBUFDS_DIFF_OUT_IBUFDISABLE_inst (
   .O(O),   // Buffer diff_p output
   .OB(OB), // Buffer diff_n output
   .I(I),   // Diff_p buffer input (connect directly to top-level port)
   .IB(IB), // Diff_n buffer input (connect directly to top-level port)
   .IBUFDISABLE(IBUFDISABLE)  // Buffer disable input, high=disable
);

// End of IBUFDS_DIFF_OUT_IBUFDISABLE_inst instantiation
```

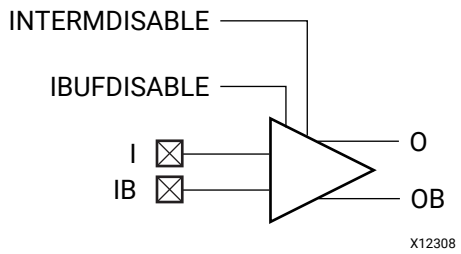
For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IBUFDS_DIFF_OUT_INTERMDISABLE

Primitive: Input Differential Buffer with Input Termination Disable, Input Disable, and Differential Output

IBUFDS_DIFF_OUT_INTERMDISABLE



Introduction

This design element is a differential input buffer used to connect internal logic to an external bidirectional pin. This element includes an uncalibrated input termination (INTERM) disable as well as input path disable as additional power saving features when the I/O is idle for a sustained time. The IOBUFDS_DIFF_OUT_INTERMDISABLE differs from the IOBUFDS_INTERMDISABLE in that it allows internal access to both phases of the differential signal. This element can only be placed in High Range (HR) banks in the 7 series devices.

Port Descriptions

Port	Direction	Width	Function
I	Input	1	Input p-side port connection. Connect directly to a top-level port in the design.
IB	Input	1	Input n-side port connection. Connect directly to a top-level port in the design.
IBUFDISABLE	Input	1	Disables input path through the buffer and forces to a logic High when USE_IBUFDISABLE is set to "TRUE" and this signal is asserted high. If USE_IBUFDISABLE is set to "FALSE" this input is ignored and should be tied to ground. This feature is generally used to reduce power at times when the I/O is idle.
INTERMDISABLE	Input	1	Disables input termination. This feature is generally used to reduce power at times when the I/O is idle.
O	Output	1	Buffer p-side output representing the input path to the device.
OB	Output	1	Buffer n-side output representing the input path to the device.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	STRING	"TRUE", "FALSE"	"FALSE"	Turns the built-in differential termination on (TRUE) or off (FALSE).
IBUF_LOW_PWR	STRING	"TRUE", "FALSE"	"TRUE"	Allows a trade off of lower power consumption vs. highest performance when referenced I/O standards are used.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
USE_IBUFDISABLE	STRING	"TRUE", "FALSE"	"TRUE"	Enables or disables the feature of IBUFDISABLE.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IBUFDS_DIFF_OUT_INTERMDISABLE: Differential Input Buffer with Differential Output w/ Disable
--                                7 Series
-- Xilinx HDL Language Template, version 2020.1

IBUFDS_DIFF_OUT_INTERMDISABLE_inst : IBUFDS_DIFF_OUT_INTERMDISABLE
generic map (
   DIFF_TERM => "FALSE", -- Differential Termination
   IBUF_LOW_PWR => "TRUE", -- Low power "TRUE" vs. performance "FALSE" setting for referenced I/O standards
   IOSTANDARD => "DEFAULT", -- Specify the input I/O standard
   USE_IBUFDISABLE => "TRUE") -- Set to "TRUE" to enable IBUFDISABLE feature
port map (
   O => O,      -- Buffer diff_p output
   OB => OB,    -- Buffer diff_n output
   I => I,  -- Diff_p buffer input (connect directly to top-level port)
   IB => IB, -- Diff_n buffer input (connect directly to top-level port)
   IBUFDISABLE => IBUFDISABLE, -- Buffer disable input, low=disable
   INTERMDISABLE => INTERMDISABLE -- Input termination disable
);

-- End of IBUFDS_DIFF_OUT_INTERMDISABLE_inst instantiation
```

Verilog Instantiation Template

```
// IBUFDS_DIFF_OUT_INTERMDISABLE: Differential Input Buffer with Differential Output with Input Termination Disable
//                                May only be placed in High Range (HR) Banks
//                                7 Series
// Xilinx HDL Language Template, version 2020.1

IBUFDS_DIFF_OUT_INTERMDISABLE  #(
   .DIFF_TERM("FALSE"),     // Differential Termination, "TRUE"/"FALSE"
   .IBUF_LOW_PWR("TRUE"),   // Low power="TRUE", Highest performance="FALSE"
```

```
   .IOSTANDARD("DEFAULT"),  // Specify the input I/O standard
   .USE_IBUFDISABLE("TRUE") // Set to "TRUE" to enable IBUFDISABLE feature
) IBUFDS_DIFF_OUT_INTERMDISABLE_inst (
   .O(O),   // Buffer diff_p output
   .OB(OB), // Buffer diff_n output
   .I(I),   // Diff_p buffer input (connect directly to top-level port)
   .IB(IB), // Diff_n buffer input (connect directly to top-level port)
   .IBUFDISABLE(IBUFDISABLE),    // Buffer disable input, high=disable
   .INTERMDISABLE(INTERMDISABLE) // Input Termination Disable
);

// End of IBUFDS_DIFF_OUT_INTERMDISABLE_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IBUFDS_GTE2

Primitive: Gigabit Transceiver Buffer

Introduction

IBUFDS_GTE2 is the gigabit transceiver input pad buffer component in 7 series devices. The REFCLK signal should be routed to the dedicated reference clock input pins on the serial transceiver, and you should instantiate the IBUFDS_GTE2 primitive in your design. See the *7 Series FPGAs GTX/GTH Transceivers User Guide* (UG476) for more information on PCB layout requirements, including reference clock requirements.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Recommended
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IBUFDS_GTE2: Gigabit Transceiver Buffer
--              7 Series
-- Xilinx HDL Language Template, version 2020.1

IBUFDS_GTE2_inst : IBUFDS_GTE2
generic map (
   CLKCM_CFG => TRUE,    -- Refer to Transceiver User Guide
   CLKRCV_TRST => TRUE,  -- Refer to Transceiver User Guide
   CLKSWING_CFG => '11'  -- Refer to Transceiver User Guide
)
port map (
   O => O,          -- 1-bit output: Refer to Transceiver User Guide
   ODIV2 => ODIV2, -- 1-bit output: Refer to Transceiver User Guide
   CEB => CEB,      -- 1-bit input: Refer to Transceiver User Guide
   I => I,          -- 1-bit input: Refer to Transceiver User Guide
   IB => IB         -- 1-bit input: Refer to Transceiver User Guide
);

-- End of IBUFDS_GTE2_inst instantiation
```

Verilog Instantiation Template

```
// IBUFDS_GTE2: Gigabit Transceiver Buffer
//              7 Series
// Xilinx HDL Language Template, version 2020.1

IBUFDS_GTE2 #(
```

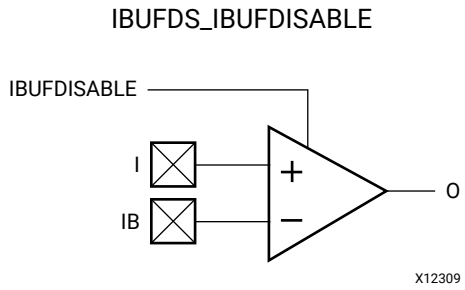
```
   .CLKCM_CFG("TRUE"),   // Refer to Transceiver User Guide
   .CLKRCV_TRST("TRUE"), // Refer to Transceiver User Guide
   .CLKSWING_CFG(2'b11)  // Refer to Transceiver User Guide
)
IBUFDS_GTE2_inst (
   .O(O),         // 1-bit output: Refer to Transceiver User Guide
   .ODIV2(ODIV2), // 1-bit output: Refer to Transceiver User Guide
   .CEB(CEB),     // 1-bit input: Refer to Transceiver User Guide
   .I(I),         // 1-bit input: Refer to Transceiver User Guide
   .IB(IB)        // 1-bit input: Refer to Transceiver User Guide
);

// End of IBUFDS_GTE2_inst instantiation
```

For More Information

- See the *7 Series FPGAs GTX/GTH Transceivers User Guide* (UG476).
- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IBUFDS_IBUFDISABLE

Primitive: Input Differential Buffer with Input Path Disable



Introduction

This design element is an input differential buffer used to connect internal logic to an external bidirectional pin. This element includes an input path disable as an additional power saving feature when the I/O is either is an unused state for a sustained amount of time.

Port Descriptions

Port	Direction	Width	Function
I	Input	1	Input p-side port connection. Connect directly to a top-level port in the design.
IB	Input	1	Input n-side port connection. Connect directly to a top-level port in the design.
IBUFDISABLE	Input	1	Disables input path through the buffer and forces to a logic High when USE_IBUFDISABLE is set to "TRUE" and this signal is asserted High. If USE_IBUFDISABLE is set to "FALSE" this input is ignored and should be tied to ground. This feature is generally used to reduce power at times when the I/O is idle.
O	Output	1	Buffer output representing the input path to the device.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Put all I/O components on the top-level of the design to help facilitate hierarchical design methods. Connect the I port directly to the top-level "master" input port of the design, the IB port to the top-level "slave" input port, and the O port to the logic in which this input is to source. Specify the desired generic/defparam values to configure the proper behavior of the buffer.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	STRING	"TRUE", "FALSE"	"FALSE"	Turns the built-in differential termination on (TRUE) or off (FALSE).
IBUF_LOW_PWR	STRING	"TRUE", "FALSE"	"TRUE"	Allows a trade off of lower power consumption vs. highest performance when referenced I/O standards are used.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
USE_IBUFDISABLE	STRING	"TRUE", "FALSE"	"TRUE"	Enables or disables the feature of IBUFDISABLE.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IBUFDS_IBUFDISABLE: Differential Input Buffer w/ Disable
--                     7 Series
-- Xilinx HDL Language Template, version 2020.1

IBUFDS_IBUFDISABLE_inst : IBUFDS_IBUFDISABLE
generic map (
   DIFF_TERM => "FALSE", -- Differential Termination
   IBUF_LOW_PWR => "TRUE", -- Low power (TRUE) vs. performance (FALSE) setting for referenced I/O standards
   IOSTANDARD => "DEFAULT", -- Specify the input I/O standard
   USE_IBUFDISABLE => "TRUE") -- Set to "TRUE" to enable IBUFDISABLE feature
port map (
   O => O,  -- Buffer output
   I => I,  -- Diff_p buffer input (connect directly to top-level port)
   IB => IB, -- Diff_n buffer input (connect directly to top-level port)
   IBUFDISABLE => IBUFDISABLE -- Buffer disable input, low=disable
);

-- End of IBUFDS_IBUFDISABLE_inst instantiation
```

Verilog Instantiation Template

```
// IBUFDS_IBUFDISABLE: Differential Input Buffer with Input Disable
//                     7 Series
// Xilinx HDL Language Template, version 2020.1

IBUFDS_IBUFDISABLE #(
   .DIFF_TERM("FALSE"),      // Differential Termination
   .IBUF_LOW_PWR("TRUE"),    // Low power="TRUE", Highest performance="FALSE"
   .IOSTANDARD("DEFAULT"),   // Specify the input I/O standard
   .USE_IBUFDISABLE("TRUE")  // Set to "TRUE" to enable IBUFDISABLE feature
) IBUFDS_IBUFDISABLE_inst (
   .O(O),   // Buffer output
```

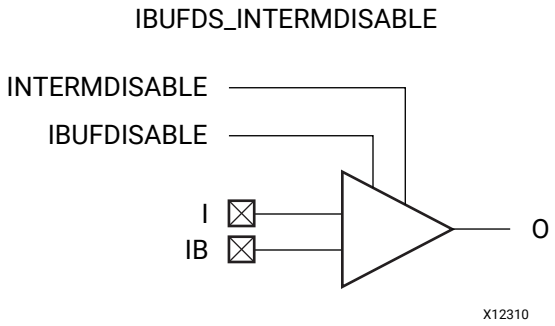
```
   .I(I),   // Diff_p buffer input (connect directly to top-level port)
   .IB(IB), // Diff_n buffer input (connect directly to top-level port)
   .IBUFDISABLE(IBUFDISABLE) // Buffer disable input, high=disable
);

// End of IBUFDS_IBUFDISABLE_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IBUFDS_INTERMDISABLE

Primitive: Input Differential Buffer with Input Termination Disable and Input Disable



Introduction

This design element is an input differential buffer used to connect internal logic to an external bidirectional pin. This element includes an uncalibrated input termination (INTERM) disable as well as input path disable as additional power saving features when the input is idle for a sustained amount of time. This element may only be placed in High Range (HR) banks in the 7 series devices.

Port Descriptions

Port	Direction	Width	Function
I	Input	1	Input p-side port connection. Connect directly to a top-level port in the design.
IB	Input	1	Input n-side port connection. Connect directly to a top-level port in the design.
IBUFDISABLE	Input	1	Disables input path through the buffer and forces to a logic High when USE_IBUFDISABLE is set to "TRUE" and this signal is asserted High. If USE_IBUFDISABLE is set to "FALSE" this input is ignored and should be tied to ground. This feature is generally used to reduce power at times when the I/O is idle.
INTERMDISABLE	Input	1	Disables input termination. This feature is generally used to reduce power at times when the I/O is idle.
O	Output	1	Buffer output representing the input path to the device.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	STRING	"TRUE", "FALSE"	"FALSE"	Turns the built-in differential termination on (TRUE) or off (FALSE).
IBUF_LOW_PWR	STRING	"TRUE", "FALSE"	"TRUE"	Allows a trade off of lower power consumption versus. highest performance when referenced I/O standards are used.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
USE_IBUFDISABLE	STRING	"TRUE", "FALSE"	"TRUE"	Enables or disables the IBUFDISABLE feature. Generally used when it is not desirable to disable the input path in order to allow a read during write operation.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IBUFDS_INTERMDISABLE: Differential Input Buffer with Input Termination Disable
--                       May only be placed in High Range (HR) Banks
--                       7 Series
-- Xilinx HDL Language Template, version 2020.1

IBUFDS_INTERMDISABLE_inst : IBUFDS_INTERMDISABLE
generic map (
   DIFF_TERM => "FALSE", -- Differential Termination
   IBUF_LOW_PWR => "TRUE", -- Low power (TRUE) vs. performance (FALSE) setting for referenced I/O standards
   IOSTANDARD => "DEFAULT", -- Specify the input I/O standard
   USE_IBUFDISABLE => "TRUE") -- Set to "TRUE" to enable IBUFDISABLE feature
port map (
   O => O,  -- Buffer output
   I => I,  -- Diff_p buffer input (connect directly to top-level port)
   IB => IB, -- Diff_n buffer input (connect directly to top-level port)
   IBUFDISABLE => IBUFDISABLE, -- Buffer disable input, low=disable
   INTERMDIBSALE => INTERMDISABLE -- Input termination disable
);

-- End of IBUFDS_IBUFDISABLE_inst instantiation
```

Verilog Instantiation Template

```
// IBUFDS_INTERMDISABLE: Differential Input Buffer with Input Termination Disable
//                       May only be placed in High Range (HR) Banks
//                       7 Series
// Xilinx HDL Language Template, version 2020.1

IBUFDS_INTERMDISABLE #(
   .DIFF_TERM("FALSE"),      // Differential Termination
   .IBUF_LOW_PWR("TRUE"),    // Low power="TRUE", Highest performance="FALSE"
   .IOSTANDARD("DEFAULT"),   // Specify the input I/O standard
   .USE_IBUFDISABLE("TRUE")  // Set to "TRUE" to enable IBUFDISABLE feature
) IBUFDS_INTERMDISABLE_inst (
   .O(O),   // Buffer output
   .I(I),   // Diff_p buffer input (connect directly to top-level port)
   .IB(IB), // Diff_n buffer input (connect directly to top-level port)
```

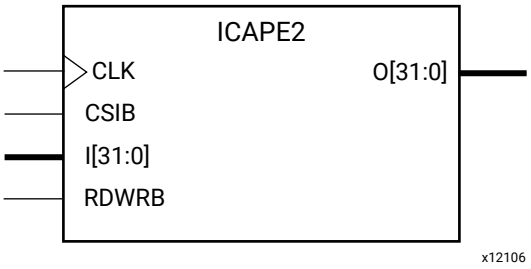
```
   .IBUFDISABLE(IBUFDISABLE),    // Buffer disable input, high=disable
   .INTERMDISABLE(INTERMDISABLE) // Input Termination Disable
);

// End of IBUFDS_INTERMDISABLE_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

ICAPE2

Primitive: Internal Configuration Access Port



Introduction

This design element gives you access to the configuration functions of the FPGA from the FPGA fabric. Using this component, commands and data can be written to and read from the configuration logic of the FPGA array. Because the improper use of this function can have a negative effect on the functionality and reliability of the FPGA, you should not use this element unless you are very familiar with its capabilities.

Port Descriptions

Port	Direction	Width	Function
CLK	Input	1	Clock Input.
CSIB	Input	1	Active-Low ICAP Enable.
I<31:0>	Input	32	Configuration data input bus.
O<31:0>	Output	32	Configuration data output bus.
RDWRB	Input	1	Read/Write Select input.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DEVICE_ID	HEX	32'h03651093, 32'h036A2093, 32'h036A4093, 32'h036A6093, 32'h036BF093, 32'h036B1093, 32'h036B3093, 32'h036C2093, 32'h036C4093, 32'h036C6093, 32'h036DF093, 32'h036D1093, 32'h036D3093, 32'h036D5093, 32'h036D9093, 32'h0362C093, 32'h0362D093, 32'h0363B093, 32'h0364C093, 32'h0371F093, 32'h0372C093, 32'h0377F093, 32'h03627093, 32'h03628093, 32'h03631093, 32'h03636093, 32'h03642093, 32'h03647093, 32'h03656093, 32'h03667093, 32'h03671093, 32'h03676093, 32'h03680093, 32'h03681093, 32'h03682093, 32'h03687093, 32'h03691093, 32'h03692093, 32'h03696093, 32'h03702093, 32'h03704093, 32'h03711093, 32'h03722093, 32'h03727093, 32'h03731093, 32'h03747093, 32'h03751093, 32'h03752093, 32'h03762093, 32'h03771093, 32'h03782093	0'h3651093	Specifies the pre-programmed Device ID value to be used for simulation purposes.
ICAP_WIDTH	STRING	"X32", "X8", "X16"	"X32"	Specifies the input and output data width.
SIM_CFG_FILE_NAME	STRING	String representing file name and location	"NONE"	Specifies the Raw Bitstream (RBT) file to be parsed by the simulation model.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- ICAPE2: Internal Configuration Access Port
--         7 Series
-- Xilinx HDL Language Template, version 2020.1

ICAPE2_inst : ICAPE2
generic map (
   DEVICE_ID => X"3651093",     -- Specifies the pre-programmed Device ID value to be used for simulation
                                -- purposes.
   ICAP_WIDTH => "X32",         -- Specifies the input and output data width.
   SIM_CFG_FILE_NAME => "NONE"  -- Specifies the Raw Bitstream (RBT) file to be parsed by the simulation
                                -- model.
)
port map (
   O => O,          -- 32-bit output: Configuration data output bus
   CLK => CLK,      -- 1-bit input: Clock Input
   CSIB => CSIB,    -- 1-bit input: Active-Low ICAP Enable
   I => I,          -- 32-bit input: Configuration data input bus
   RDWRB => RDWRB   -- 1-bit input: Read/Write Select input
);

-- End of ICAPE2_inst instantiation
```

Verilog Instantiation Template

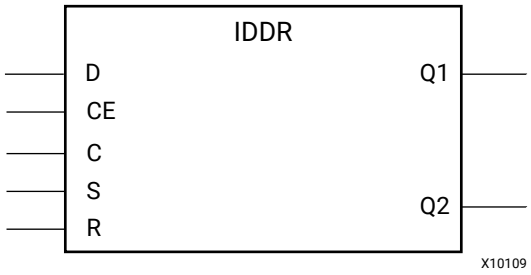
```
// ICAPE2: Internal Configuration Access Port
//         7 Series
// Xilinx HDL Language Template, version 2020.1

ICAPE2 #(
   .DEVICE_ID(0'h3651093),     // Specifies the pre-programmed Device ID value to be used for simulation
                               // purposes.
   .ICAP_WIDTH("X32"),         // Specifies the input and output data width.
   .SIM_CFG_FILE_NAME("NONE")  // Specifies the Raw Bitstream (RBT) file to be parsed by the simulation
                               // model.
)
ICAPE2_inst (
   .O(O),          // 32-bit output: Configuration data output bus
   .CLK(CLK),      // 1-bit input: Clock Input
   .CSIB(CSIB),    // 1-bit input: Active-Low ICAP Enable
   .I(I),          // 32-bit input: Configuration data input bus
   .RDWRB(RDWRB)   // 1-bit input: Read/Write Select input
);

// End of ICAPE2_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configuration Guide* (UG470).

IDDR

Primitive: Input Double Data-Rate Register



Introduction

This design element is a dedicated input register designed to receive external double data rate (DDR) signals into Xilinx® FPGAs. The IDDR is available with modes that present the data to the FPGA fabric at the time and clock edge they are captured, or on the same clock edge. This feature allows you to avoid additional timing complexities and resource usage.

- **OPPOSITE_EDGE mode** Data is recovered in the classic DDR methodology. Given a DDR data and clock at pin D and C respectively, Q1 changes after every positive edge of clock C, and Q2 changes after every negative edge of clock C.
- **SAME_EDGE mode** Data is still recovered by opposite edges of clock C. However, an extra register has been placed behind the negative edge data register. This extra register is clocked with positive clock edge of clock signal C. As a result, DDR data is now presented into the FPGA fabric at the same clock edge. However, because of this feature, the data pair appears to be "separated." Q1 and Q2 no longer have pair 1 and 2. Instead, the first pair presented is Pair 1 and DONT_CARE, followed by Pair 2 and 3 at the next clock cycle.
- **SAME_EDGE_PIPELINED mode** Recovers data in a similar fashion as the SAME_EDGE mode. In order to avoid the "separated" effect of the SAME_EDGE mode, an extra register has been placed in front of the positive edge data register. A data pair now appears at the Q1 and Q2 pin at the same time. However, using this mode costs you an additional cycle of latency for Q1 and Q2 signals to change.

IDDR also works with the SelectIO™ features, such as the IDELAYE2.

Note: For high speed interfaces, you can use the IDDR_2CLK to specify two independent clocks to capture the data. Use this component when the performance requirements of the IDDR are not adequate, because the IDDR_2CLK requires more clocking resources and can imply placement restrictions that are not necessary when using the IDDR component.

Port Descriptions

Port	Direction	Width	Function
Q1 - Q2	Output	1	The IDDR output pins that connect to the FPGA fabric.
C	Input	1	Clock input pin.
CE	Input	1	The enable pin affects the loading of data into the DDR flip-flop. When Low, clock transitions are ignored and new data is not loaded into the DDR flip-flop. CE must be high to load new data into the flip-flop.
D	Input	1	Input to the IDDR module.This pin connects to a top-level input or bidirectional port, and IDELAYE2 configured for an input delay or to an appropriate input or bidirectional buffer.
R	Input	1	Active-High reset forcing Q1 and Q2 to a logic zero. Can be synchronous or asynchronous based on the SRTYPE attribute.
S	Input	1	Active-High reset forcing Q1 and Q2 to a logic one. Can be synchronous or asynchronous based on the SRTYPE attribute.

***Note*:** You cannot have an active set and an active reset in this component. One or both of the signals R and S must be tied to ground.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DDR_CLK_EDGE	STRING	"OPPOSITE_EDGE", "SAME_EDGE", "SAME_EDGE_PIPELINED"	"OPPOSITE_EDGE"	Sets the IDDR mode of operation with respect to clock edge.
INIT_Q1	BINARY	0, 1	0	Initial value on the Q1 pin after configuration startup or when GSR is asserted.
INIT_Q2	BINARY	0, 1	0	Initial value on the Q2 pin after configuration startup or when GSR is asserted.
SRTYPE	STRING	"SYNC" or "ASYNC"	"SYNC"	Set/reset type selection. "SYNC" specifies the behavior of the reset (R) and set (S) pins to be synchronous to the positive edge of the C clock pin. "ASYNC" specifies an asynchronous set/reset function.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IDDR: Double Data Rate Input Register with Set, Reset
--       and Clock Enable.
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

IDDR_inst : IDDR
generic map (
   DDR_CLK_EDGE => "OPPOSITE_EDGE", -- "OPPOSITE_EDGE", "SAME_EDGE"
                                    -- or "SAME_EDGE_PIPELINED"
   INIT_Q1 => '0', -- Initial value of Q1: '0' or '1'
   INIT_Q2 => '0', -- Initial value of Q2: '0' or '1'
   SRTYPE => "SYNC") -- Set/Reset type: "SYNC" or "ASYNC"
port map (
   Q1 => Q1, -- 1-bit output for positive edge of clock
   Q2 => Q2, -- 1-bit output for negative edge of clock
   C => C,   -- 1-bit clock input
   CE => CE, -- 1-bit clock enable input
   D => D,   -- 1-bit DDR data input
   R => R,   -- 1-bit reset
   S => S    -- 1-bit set
   );

-- End of IDDR_inst instantiation
```

Verilog Instantiation Template

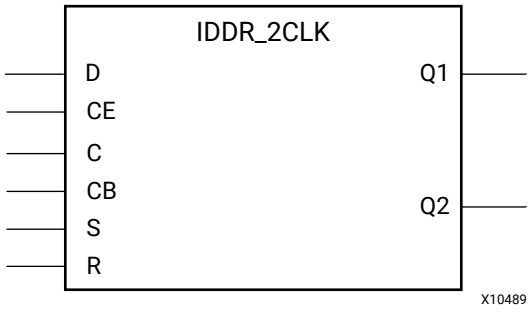
```
// IDDR: Input Double Data Rate Input Register with Set, Reset
//       and Clock Enable.
//       7 Series
// Xilinx HDL Language Template, version 2020.1

IDDR #(
   .DDR_CLK_EDGE("OPPOSITE_EDGE"), // "OPPOSITE_EDGE", "SAME_EDGE"
                                   //    or "SAME_EDGE_PIPELINED"
   .INIT_Q1(1'b0), // Initial value of Q1: 1'b0 or 1'b1
   .INIT_Q2(1'b0), // Initial value of Q2: 1'b0 or 1'b1
   .SRTYPE("SYNC") // Set/Reset type: "SYNC" or "ASYNC"
) IDDR_inst (
   .Q1(Q1), // 1-bit output for positive edge of clock
   .Q2(Q2), // 1-bit output for negative edge of clock
   .C(C),   // 1-bit clock input
   .CE(CE), // 1-bit clock enable input
   .D(D),   // 1-bit DDR data input
   .R(R),   // 1-bit reset
   .S(S)    // 1-bit set
);

// End of IDDR_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IDDR_2CLK

Primitive: Input Double Data-Rate Register with Dual Clock Inputs



Introduction

This design element is a dedicated input register designed to receive external double data rate (DDR) signals into Xilinx® FPGAs. You should only use the IDDR_2CLK for very high speed interfaces, because it requires more clocking resources, more power, and can imply certain placement restrictions that are not necessary when using the IDDR component. The IDDR component is also easier to use, uses fewer resources, and has fewer restrictions, though it cannot operate at the same high I/O speeds. The IDDR_2CLK is available with modes that present the data to the FPGA fabric at the time and clock edge they are captured, or on the same clock edge. This feature allows designers to avoid additional timing complexities and resource usage.

- **OPPOSITE_EDGE mode:** Data is presented in the classic DDR methodology. Given a DDR data and clock at pin D and C respectively, Q1 changes after every positive edge of clock C, and Q2 changes after every positive edge of clock CB.
- **SAME_EDGE mode:** Data is still presented by positive edges of each clock. However, an extra register has been placed in front of the CB clocked data register. This extra register is clocked with positive clock edge of clock signal C. As a result, DDR data is now presented into the FPGA fabric at the positive edge of clock C. However, because of this feature, the data pair appears to be "separated." Q1 and Q2 no longer have pair 1 and 2. Instead, the first pair presented is Pair 1 and DON'T CARE, followed by Pair 2 and 3 at the next clock cycle.
- **SAME_EDGE_PIPELINED mode:** Presents data in a similar fashion as the SAME_EDGE mode. In order to avoid the "separated" effect of the SAME_EDGE mode, an extra register has been placed in front of the C clocked data register. A data pair now appears at the Q1 and Q2 pin at the same time during the positive edge of C. However, using this mode requires an additional cycle of latency for Q1 and Q2 signals to change.

IDDR also works with SelectIO™ features, such as the IODELAYE2.

Port Descriptions

Port	Direction	Width	Function
Q1 : Q2	Output	1	These pins are the IDDR output that connects to the FPGA fabric. Q1 is the first data output and Q2 is the second data output.
C	Input	1	Primary clock input pin used to capture the positive edge data.
CB	Input	1	Secondary clock input pin (typically 180 degrees out of phase with the primary clock) used to capture the negative edge data.
CE	Input	1	When asserted Low, this port disables the output clock at port O.
D	Input	1	This pin is where the DDR data is presented into the IDDR module. It connects to a top-level input or bi-directional port, and IODELAY configured for an input delay or to an appropriate input or bidirectional buffer.
R	Input	1	Active-High reset forcing Q1 and Q2 to a logic one. Can be synchronous or asynchronous based on the SRTYPE attribute.
S	Input	1	Active-High reset forcing Q1 and Q2 to a logic zero. Can be synchronous or asynchronous based on the SRTYPE attribute.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

- Connect the C pin to the appropriate clock source, representing the positive clock edge and CB to the clock source representing the negative clock edge.
- Connect the D pin to the top-level input, or bidirectional port, an IODELAY, or an instantiated input or bidirectional buffer.
- The Q1 and Q2 pins should be connected to the appropriate data sources.
- CE should be tied high when not used, or connected to the appropriate clock enable logic.
- R and S pins should be tied low, if not used, or to the appropriate set or reset generation logic.
- Set all attributes to the component to represent the desired behavior.
- Always instantiate this component in pairs with the same clocking, and to LOC those to the appropriate P and N I/O pair in order not to sacrifice possible I/O resources.
- Always instantiate this component in the top-level hierarchy of your design, along with any other instantiated I/O components for the design. This helps facilitate hierarchical design flows/practices.

- To minimize CLK skew, both CLK and CLKB should come from global routing (MMCM) and not from the local inversion. MMCM de-skews these clocks whereas the local inversion adds skew.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DDR_CLK_EDGE	STRING	"OPPOSITE_EDGE","SAME_EDGE""SAME_EDGE_PIPELINED"	"OPPOSITE_EDGE"	DDR clock mode recovery mode selection. See Introduction for more explanation.
INIT_Q1	BINARY	0, 1	0	Initial value on the Q1 pin after configuration startup or when GSR is asserted.
INIT_Q2	BINARY	0, 1	0	Initial value on the Q2 pin after configuration startup or when GSR is asserted.
SRTYPE	STRING	"SYNC" or "ASYNC"	"SYNC"	Set/reset type selection. SYNC" specifies the behavior of the reset (R) and set (S) pins to be synchronous to the positive edge of the C clock pin. "ASYNC" specifies an asynchronous set/reset function.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IDDR_2CLK: Dual-Clock, Input Double Data Rate Input Register with
--            Set, Reset and Clock Enable.
--            7 Series
-- Xilinx HDL Language Template, version 2020.1

IDDR_2CLK_inst : IDDR_2CLK
generic map (
   DDR_CLK_EDGE => "OPPOSITE_EDGE", -- "OPPOSITE_EDGE", "SAME_EDGE"
                                    -- or "SAME_EDGE_PIPELINED"
   INIT_Q1 => '0', -- Initial value of Q1: '0' or '1'
   INIT_Q2 => '0', -- Initial value of Q2: '0' or '1'
   SRTYPE => "SYNC") -- Set/Reset type: "SYNC" or "ASYNC"
port map (
   Q1 => Q1, -- 1-bit output for positive edge of clock
   Q2 => Q2, -- 1-bit output for negative edge of clock
   C => C,   -- 1-bit primary clock input
   CB => CB, -- 1-bit secondary clock input
   CE => CE, -- 1-bit clock enable input
   D => D,   -- 1-bit DDR data input
   R => R,   -- 1-bit reset
   S => S    -- 1-bit set
   );

-- End of IDDR_2CLK_inst instantiation
```

Verilog Instantiation Template

```
// IDDR_2CLK: Dual-Clock, Input Double Data Rate Input Register with
//            Set, Reset and Clock Enable.
//            7 Series
// Xilinx HDL Language Template, version 2020.1
```

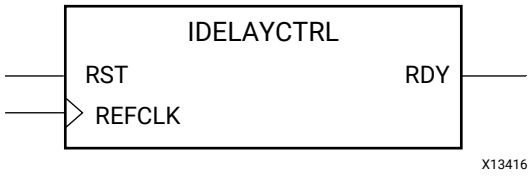
```
IDDR_2CLK #(
   .DDR_CLK_EDGE("OPPOSITE_EDGE"), // "OPPOSITE_EDGE", "SAME_EDGE"
                                   //    or "SAME_EDGE_PIPELINED"
   .INIT_Q1(1'b0), // Initial value of Q1: 1'b0 or 1'b1
   .INIT_Q2(1'b0), // Initial value of Q2: 1'b0 or 1'b1
   .SRTYPE("SYNC") // Set/Reset type: "SYNC" or "ASYNC"
) IDDR_2CLK_inst (
   .Q1(Q1), // 1-bit output for positive edge of clock
   .Q2(Q2), // 1-bit output for negative edge of clock
   .C(C),   // 1-bit primary clock input
   .CB(CB), // 1-bit secondary clock input
   .CE(CE), // 1-bit clock enable input
   .D(D),   // 1-bit DDR data input
   .R(R),   // 1-bit reset
   .S(S)    // 1-bit set
);

// End of IDDR_2CLK_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IDELAYCTRL

Primitive: IDELAYE2/ODELAYE2 Tap Delay Value Control



Introduction

At least one of these design elements must be instantiated when using IDELAYE2 or ODELAYE2. The IDELAYCTRL module provides a reference clock input that allows internal circuitry to derive a voltage bias, independent of PVT (process, voltage, and temperature) variations, to define precise delay tap values for the associated IDELAYE2 and ODELAYE2 components. Use the IODELAY_GROUP attribute when instantiating this component to distinguish which IDELAYCTRL is associated with which IDELAYE2 and ODELAYE2.

Port Descriptions

Port	Direction	Width	Function
RDY	Output	1	The ready (RDY) signal indicates when the IDELAYE2 and ODELAYE2 modules in the specific region are calibrated. The RDY signal is deasserted if REFCLK is held High or Low for one clock period or more. If RDY is deasserted Low, the IDELAYCTRL module must be reset. If not needed, RDY to be unconnected/ignored.
REFCLK	Input	1	Time reference to IDELAYCTRL to calibrate all IDELAYE2 and ODELAYE2 modules in the same region. REFCLK can be supplied directly from a user-supplied source or the MMCME2/PLLE2 and must be routed on a global clock buffer.
RST	Input	1	Active-High asynchronous reset. To ensure proper IDELAYE2 and ODELAYE2 operation, IDELAYCTRL must be reset after configuration and the REFCLK signal is stable. A reset pulse width Tidelayctrl_rpw is required.

RST (Module reset) Resets the IDELAYCTRL circuitry. The RST signal is an active-High asynchronous reset. To reset the IDELAYCTRL, assert it High for at least 50 ns.

REFCLK (Reference Clock) Provides a voltage bias, independent of process, voltage, and temperature variations, to the tap-delay lines in the IOBs. The frequency of REFCLK must be 200 MHz to guarantee the tap-delay value specified in the applicable data sheet.

RDY (Ready Output) Indicates the validity of the reference clock input, REFCLK. When REFCLK disappears (i.e., REFCLK is held High or Low for one clock period or more), the RDY signal is deasserted.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IDELAYCTRL: IDELAYE2/ODELAYE2 Tap Delay Value Control
--             7 Series
-- Xilinx HDL Language Template, version 2020.1

IDELAYCTRL_inst : IDELAYCTRL
port map (
   RDY => RDY,       -- 1-bit output: Ready output
   REFCLK => REFCLK, -- 1-bit input: Reference clock input
   RST => RST        -- 1-bit input: Active high reset input
);

-- End of IDELAYCTRL_inst instantiation
```

Verilog Instantiation Template

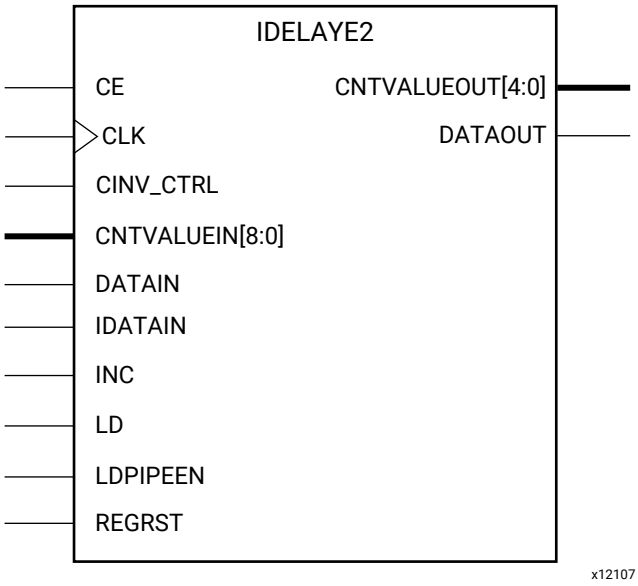
```
// IDELAYCTRL: IDELAYE2/ODELAYE2 Tap Delay Value Control
//             7 Series
// Xilinx HDL Language Template, version 2020.1

(* IODELAY_GROUP = <iodelay_group_name> *) // Specifies group name for associated IDELAYs/ODELAYs and IDELAYCTRL

IDELAYCTRL IDELAYCTRL_inst (
   .RDY(RDY),       // 1-bit output: Ready output
   .REFCLK(REFCLK), // 1-bit input: Reference clock input
   .RST(RST)        // 1-bit input: Active high reset input
);

// End of IDELAYCTRL_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IDELAYE2

Primitive: Input Fixed or Variable Delay Element



Introduction

Every I/O block contains a programmable absolute delay element called IDELAYE2. The IDELAYE2 can be connected to an input register/ISERDESE2 or driven directly into FPGA logic. The IDELAYE2 is a 31-tap, wraparound, delay element with a calibrated tap resolution. Refer to the 7 series FPGA Data Sheet for delay values. The IDELAYE2 allows incoming signals to be delayed on an individual basis. The tap delay resolution is varied by selecting an IDELAYCTRL reference clock from the range specified in the 7 series FPGA Data Sheet.

Port Descriptions

Port	Direction	Width	Function
C	Input	1	All control inputs to IDELAYE2 primitive (RST, CE, and INC) are synchronous to the clock input (C). A clock must be connected to this port when IDELAYE2 is configured in "VARIABLE", "VAR_LOAD" or "VAR_LOAD_PIPE" mode. C can be locally inverted, and must be supplied by a global or regional clock buffer. This clock should be connected to the same clock in the SelectIO logic resources (when using ISERDESE2 and OSERDESE2, C is connected to CLKDIV).
CE	Input	1	Active-High enable for increment/decrement function.
CINVCTRL	Input	1	The CINVCTRL pin is used for dynamically switching the polarity of C pin. This is for use in applications when glitches are not an issue. When switching the polarity, do not use the IDELAYE2 control pins for two clock cycles.

Port	Direction	Width	Function
CNTVALUEIN <4:0>	Input	5	Counter value from FPGA logic for dynamically loadable tap value input.
CNTVALUEOUT <4:0>	Output	5	The CNTVALUEOUT pins are used for reporting the dynamically switching value of the delay element. CNTVALUEOUT is only available when IDELAYE2 is in "VAR_LOAD" or "VAR_LOAD_PIPE" mode.
DATAIN	Input	1	The DATAIN input is directly driven by the FPGA logic providing a logic accessible delay line. The data is driven back into the FPGA logic through the DATAOUT port with a delay set by the IDELAY_VALUE. DATAIN can be locally inverted. The data cannot be driven to an I/O.
DATAOUT	Output	1	Delayed data from either the IDATAIN or DATAIN input paths. DATAOUT connects to an ISERDESE2, input register or FPGA logic.
IDATAIN	Input	1	The IDATAIN input is driven by its associated I/O. The data can be driven to either an ISERDESE2 or input register block, directly into the FPGA logic, or to both through the DATAOUT port with a delay set by the IDELAY_VALUE.
INC	Input	1	Selects whether tap delay numbers will be incremented or decremented. INC = 1 increments when CE is high. INC=0 decrements.
LD	Input	1	• In “VARIABLE” mode, loads the value set by the IDELAY_VALUE attribute. The default value is zero. • In “VAR_LOAD” mode, loads the value of CNTVALUEIN. The value present at CNTVALUEIN[4:0] will be the new tap value. • In “VAR_LOAD_PIPE” mode, loads the value currently in the pipeline register. The value present in the pipeline register will be the new tap value.
LDPIPEEN	Input	1	When High, loads the pipeline register with the value currently on the CNTVALUEIN pins.
REGRST	Input	1	When high, resets the pipeline register to all zeros. Only used in "VAR_LOAD_PIPE" mode.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
CINVCTRL_SEL	STRING	"FALSE", "TRUE"	"FALSE"	Enables the CINVCTRL_SEL pin to dynamically switch the polarity of the C pin.

Attribute	Type	Allowed Values	Default	Description
DELAY_SRC	STRING	"IDATAIN", "DATAIN"	"IDATAIN"	Select the delay source input to the IDELAYE2. • "IDATAIN": IDELAYE2 chain input is IDATAIN. • "DATAIN": IDELAYE2 chain input is DATAIN.
HIGH _PERFORMANCE _MODE	STRING	"FALSE", "TRUE"	"FALSE"	When TRUE, this attribute reduces the output jitter. When FALSE, power consumption is reduced. The difference in power consumption is quantified in the Xilinx Power Estimator tool.
IDELAY_TYPE	STRING	"FIXED", "VARIABLE", "VAR_LOAD", "VAR_LOAD_PIPE"	"FIXED"	Sets the type of tap delay line. • "FIXED": Sets a static delay value. • "VARIABLE": Dynamically adjust (increment/ decrement) delay value. • "VAR_LOAD": Dynamically loads tap values. • "VAR_LOAD_PIPE": Pipelined dynamically loadable tap values.
IDELAY_VALUE	DECIMAL	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31	0	Specifies the fixed number of delay taps in fixed mode or the initial starting number of taps in "VARIABLE" mode (input path). When IDELAY_TYPE is set to "VAR_LOAD" or "VAR_LOAD_PIPE" mode, this value is ignored.
PIPE_SEL	STRING	"FALSE", "TRUE"	"FALSE"	Select pipelined mode.
REFCLK _FREQUENCY	1 significant digit FLOAT	190-210, 290-310 MHz	200.0	Sets the tap value (in MHz) used by the timing analyzer for static timing analysis and functional/ timing simulation. The frequency of REFCLK must be within the given datasheet range to guarantee the tap-delay value and performance.
SIGNAL_PATTERN	STRING	"DATA", "CLOCK"	"DATA"	Causes the timing analyzer to account for the appropriate amount of delay-chain jitter in the data or clock path.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IDELAYE2: Input Fixed or Variable Delay Element
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

IDELAYE2_inst : IDELAYE2
generic map (
   CINVCTRL_SEL => "FALSE",          -- Enable dynamic clock inversion (FALSE, TRUE)
   DELAY_SRC => "IDATAIN",           -- Delay input (IDATAIN, DATAIN)
   HIGH_PERFORMANCE_MODE => "FALSE", -- Reduced jitter ("TRUE"), Reduced power ("FALSE")
   IDELAY_TYPE => "FIXED",           -- FIXED, VARIABLE, VAR_LOAD, VAR_LOAD_PIPE
   IDELAY_VALUE => 0,                -- Input delay tap setting (0-31)
   PIPE_SEL => "FALSE",              -- Select pipelined mode, FALSE, TRUE
   REFCLK_FREQUENCY => 200.0,        -- IDELAYCTRL clock input frequency in MHz (190.0-210.0, 290.0-310.0).
   SIGNAL_PATTERN => "DATA"          -- DATA, CLOCK input signal
```

```
)
port map (
   CNTVALUEOUT => CNTVALUEOUT, -- 5-bit output: Counter value output
   DATAOUT => DATAOUT,         -- 1-bit output: Delayed data output
   C => C,                     -- 1-bit input: Clock input
   CE => CE,                   -- 1-bit input: Active high enable increment/decrement input
   CINVCTRL => CINVCTRL,       -- 1-bit input: Dynamic clock inversion input
   CNTVALUEIN => CNTVALUEIN,   -- 5-bit input: Counter value input
   DATAIN => DATAIN,           -- 1-bit input: Internal delay data input
   IDATAIN => IDATAIN,         -- 1-bit input: Data input from the I/O
   INC => INC,                 -- 1-bit input: Increment / Decrement tap delay input
   LD => LD,                   -- 1-bit input: Load IDELAY_VALUE input
   LDPIPEEN => LDPIPEEN,       -- 1-bit input: Enable PIPELINE register to load data input
   REGRST => REGRST            -- 1-bit input: Active-high reset tap-delay input
);

-- End of IDELAYE2_inst instantiation
```

Verilog Instantiation Template

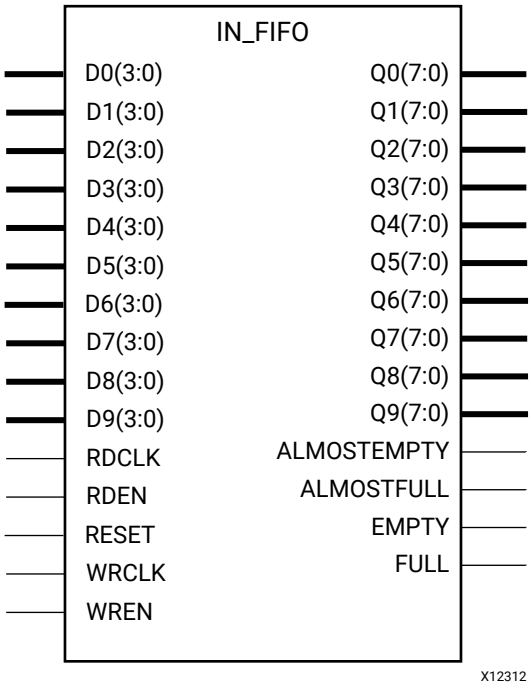
```
// IDELAYE2: Input Fixed or Variable Delay Element
//           7 Series
// Xilinx HDL Language Template, version 2020.1

(* IODELAY_GROUP = <iodelay_group_name> *) // Specifies group name for associated IDELAYs/ODELAYs and IDELAYCTRL

IDELAYE2 #(
   .CINVCTRL_SEL("FALSE"),          // Enable dynamic clock inversion (FALSE, TRUE)
   .DELAY_SRC("IDATAIN"),           // Delay input (IDATAIN, DATAIN)
   .HIGH_PERFORMANCE_MODE("FALSE"), // Reduced jitter ("TRUE"), Reduced power ("FALSE")
   .IDELAY_TYPE("FIXED"),           // FIXED, VARIABLE, VAR_LOAD, VAR_LOAD_PIPE
   .IDELAY_VALUE(0),                // Input delay tap setting (0-31)
   .PIPE_SEL("FALSE"),              // Select pipelined mode, FALSE, TRUE
   .REFCLK_FREQUENCY(200.0),       // IDELAYCTRL clock input frequency in MHz (190.0-210.0, 290.0-310.0).
   .SIGNAL_PATTERN("DATA")          // DATA, CLOCK input signal
)
IDELAYE2_inst (
   .CNTVALUEOUT(CNTVALUEOUT), // 5-bit output: Counter value output
   .DATAOUT(DATAOUT),         // 1-bit output: Delayed data output
   .C(C),                     // 1-bit input: Clock input
   .CE(CE),                   // 1-bit input: Active high enable increment/decrement input
   .CINVCTRL(CINVCTRL),       // 1-bit input: Dynamic clock inversion input
   .CNTVALUEIN(CNTVALUEIN),   // 5-bit input: Counter value input
   .DATAIN(DATAIN),           // 1-bit input: Internal delay data input
   .IDATAIN(IDATAIN),         // 1-bit input: Data input from the I/O
   .INC(INC),                 // 1-bit input: Increment / Decrement tap delay input
   .LD(LD),                   // 1-bit input: Load IDELAY_VALUE input
   .LDPIPEEN(LDPIPEEN),       // 1-bit input: Enable PIPELINE register to load data input
   .REGRST(REGRST)            // 1-bit input: Active-high reset tap-delay input
);

// End of IDELAYE2_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IN_FIFO

Primitive: Input First-In, First-Out (FIFO)



Introduction

The Input FIFO is a new resource located next to the I/O. This dedicated hardware is designed to help transition the data from the input port, input register, IDDR, or ISERDESE2 to the fabric. It has two basic modes. The first is a 4x4 mode where the data coming into the FIFO goes out at the same rate. The second mode is a 4x8 mode where the data coming out is de-serialized by a factor of 2. In other words in 4x8 mode 4-bits go to the IN_FIFO and 8-bits come out. Features of this component include:

- Array dimensions: 80 wide, 8 deep (4x8 mode); 40 wide, 8 deep (4x4 mode)
- Empty and Full flags
- Programmable Almost Empty and Almost Full flags

Port Descriptions

Port	Direction	Width	Function
ALMOSTEMPTY	Output	1	Active-High output flag indicating the FIFO is almost empty. The threshold of the almost empty flag is set by the ALMOST_EMPTY_VALUE attribute.

Port	Direction	Width	Function
ALMOSTFULL	Output	1	Active-High output flag indicating the FIFO is almost full. The threshold of the almost empty flag is set by the ALMOST_FULL_VALUE attribute.
D0<3:0>	Input	4	Channel 0 input bus.
D1<3:0>	Input	4	Channel 1 input bus.
D2<3:0>	Input	4	Channel 2 input bus.
D3<3:0>	Input	4	Channel 3 input bus.
D4<3:0>	Input	4	Channel 4 input bus.
D5<7:0>	Input	8	Channel 5 input bus.
D6<7:0>	Input	8	Channel 6 input bus.
D7<3:0>	Input	4	Channel 7 input bus.
D8<3:0>	Input	4	Channel 8 input bus.
D9<3:0>	Input	4	Channel 9 input bus.
EMPTY	Output	1	Active-High output flag indicating the FIFO is empty.
FULL	Output	1	Active-High output flag indicating the FIFO is full.
Q0<7:0>	Output	8	Channel 0 input bus.
Q1<7:0>	Output	8	Channel 1 output bus.
Q2<7:0>	Output	8	Channel 2 output bus.
Q3<7:0>	Output	8	Channel 3 output bus.
Q4<7:0>	Output	8	Channel 4 output bus.
Q5<7:0>	Output	8	Channel 5 output bus.
Q6<7:0>	Output	8	Channel 6 output bus.
Q7<7:0>	Output	8	Channel 7 output bus.
Q8<7:0>	Output	8	Channel 8 output bus.
Q9<7:0>	Output	8	Channel 9 output bus.
RDCLK	Input	1	Read clock.
RDEN	Input	1	Active-High read enable.
RESET	Input	1	Active-High asynchronous reset.
WRCLK	Input	1	Write clock.
WREN	Input	1	Active-High write enable.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ALMOST_EMPTY _VALUE	DECIMAL	1, 2	1	Specifies the number of entries left before asserting the ALMOSTEMPTY output signal.
ALMOST_FULL _VALUE	DECIMAL	1, 2	1	Specifies the number of entries left before asserting the ALMOSTFULL output signal.
ARRAY_MODE	STRING	"ARRAY_MODE_4_X_8", "ARRAY_MODE_4_X_4"	"ARRAY_MODE_4_X_8"	Specifies deserializer mode: • "ARRAY_MODE_4_X_8": Four bits in, eight bits out • "ARRAY_MODE_4_X_4": Four bits in, four bits out
SYNCHRONOUS _MODE	STRING	"FALSE"	"FALSE"	Specify whether the RDCLK and WRCLK are synchronous to each other.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IN_FIFO: Input First-In, First-Out (FIFO)
--          7 Series
-- Xilinx HDL Language Template, version 2020.1

IN_FIFO_inst : IN_FIFO
generic map (
   ALMOST_EMPTY_VALUE => 1,          -- Almost empty offset (1-2)
   ALMOST_FULL_VALUE => 1,           -- Almost full offset (1-2)
   ARRAY_MODE => "ARRAY_MODE_4_X_8", -- ARRAY_MODE_4_X_8, ARRAY_MODE_4_X_4
   SYNCHRONOUS_MODE => "FALSE"       -- Clock synchronous (FALSE)
)
port map (
   -- FIFO Status Flags: 1-bit (each) output: Flags and other FIFO status outputs
   ALMOSTEMPTY => ALMOSTEMPTY, -- 1-bit output: Almost empty
   ALMOSTFULL => ALMOSTFULL,   -- 1-bit output: Almost full
   EMPTY => EMPTY,             -- 1-bit output: Empty
   FULL => FULL,               -- 1-bit output: Full
   -- Q0-Q9: 8-bit (each) output: FIFO Outputs
   Q0 => Q0,                   -- 8-bit output: Channel 0
   Q1 => Q1,                   -- 8-bit output: Channel 1
   Q2 => Q2,                   -- 8-bit output: Channel 2
   Q3 => Q3,                   -- 8-bit output: Channel 3
   Q4 => Q4,                   -- 8-bit output: Channel 4
   Q5 => Q5,                   -- 8-bit output: Channel 5
   Q6 => Q6,                   -- 8-bit output: Channel 6
   Q7 => Q7,                   -- 8-bit output: Channel 7
   Q8 => Q8,                   -- 8-bit output: Channel 8
   Q9 => Q9,                   -- 8-bit output: Channel 9
   -- D0-D9: 4-bit (each) input: FIFO inputs
   D0 => D0,                   -- 4-bit input: Channel 0
   D1 => D1,                   -- 4-bit input: Channel 1
   D2 => D2,                   -- 4-bit input: Channel 2
   D3 => D3,                   -- 4-bit input: Channel 3
   D4 => D4,                   -- 4-bit input: Channel 4
   D5 => D5,                   -- 8-bit input: Channel 5
   D6 => D6,                   -- 8-bit input: Channel 6
```

```
   D7 => D7,                   -- 4-bit input: Channel 7
   D8 => D8,                   -- 4-bit input: Channel 8
   D9 => D9,                   -- 4-bit input: Channel 9
   -- FIFO Control Signals: 1-bit (each) input: Clocks, Resets and Enables
   RDCLK => RDCLK,             -- 1-bit input: Read clock
   RDEN => RDEN,               -- 1-bit input: Read enable
   RESET => RESET,             -- 1-bit input: Reset
   WRCLK => WRCLK,             -- 1-bit input: Write clock
   WREN => WREN                -- 1-bit input: Write enable
);

-- End of IN_FIFO_inst instantiation
```

Verilog Instantiation Template

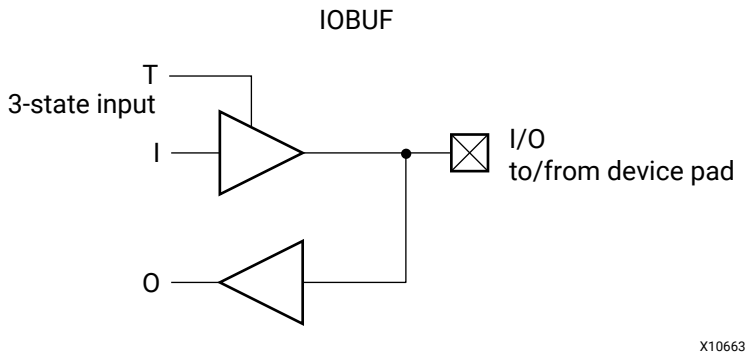
```
// IN_FIFO: Input First-In, First-Out (FIFO)
//          7 Series
// Xilinx HDL Language Template, version 2020.1

IN_FIFO #(
   .ALMOST_EMPTY_VALUE(1),          // Almost empty offset (1-2)
   .ALMOST_FULL_VALUE(1),           // Almost full offset (1-2)
   .ARRAY_MODE("ARRAY_MODE_4_X_8"), // ARRAY_MODE_4_X_8, ARRAY_MODE_4_X_4
   .SYNCHRONOUS_MODE("FALSE")       // Clock synchronous (FALSE)
)
IN_FIFO_inst (
   // FIFO Status Flags: 1-bit (each) output: Flags and other FIFO status outputs
   .ALMOSTEMPTY(ALMOSTEMPTY), // 1-bit output: Almost empty
   .ALMOSTFULL(ALMOSTFULL),   // 1-bit output: Almost full
   .EMPTY(EMPTY),             // 1-bit output: Empty
   .FULL(FULL),               // 1-bit output: Full
   // Q0-Q9: 8-bit (each) output: FIFO Outputs
   .Q0(Q0),                   // 8-bit output: Channel 0
   .Q1(Q1),                   // 8-bit output: Channel 1
   .Q2(Q2),                   // 8-bit output: Channel 2
   .Q3(Q3),                   // 8-bit output: Channel 3
   .Q4(Q4),                   // 8-bit output: Channel 4
   .Q5(Q5),                   // 8-bit output: Channel 5
   .Q6(Q6),                   // 8-bit output: Channel 6
   .Q7(Q7),                   // 8-bit output: Channel 7
   .Q8(Q8),                   // 8-bit output: Channel 8
   .Q9(Q9),                   // 8-bit output: Channel 9
   // D0-D9: 4-bit (each) input: FIFO inputs
   .D0(D0),                   // 4-bit input: Channel 0
   .D1(D1),                   // 4-bit input: Channel 1
   .D2(D2),                   // 4-bit input: Channel 2
   .D3(D3),                   // 4-bit input: Channel 3
   .D4(D4),                   // 4-bit input: Channel 4
   .D5(D5),                   // 8-bit input: Channel 5
   .D6(D6),                   // 8-bit input: Channel 6
   .D7(D7),                   // 4-bit input: Channel 7
   .D8(D8),                   // 4-bit input: Channel 8
   .D9(D9),                   // 4-bit input: Channel 9
   // FIFO Control Signals: 1-bit (each) input: Clocks, Resets and Enables
   .RDCLK(RDCLK),             // 1-bit input: Read clock
   .RDEN(RDEN),               // 1-bit input: Read enable
   .RESET(RESET),             // 1-bit input: Reset
   .WRCLK(WRCLK),             // 1-bit input: Write clock
   .WREN(WREN)                // 1-bit input: Write enable
);

// End of IN_FIFO_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IOBUF

Primitive: Bi-Directional Buffer



Introduction

The design element is a bidirectional single-ended I/O Buffer used to connect internal logic to an external bidirectional pin.

Logic Table

Inputs		Bidirectional	Outputs
T	**I**	**IO**	**O**
1	X	Z	IO
0	1	1	1
0	0	0	0

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Buffer output.
IO	In/out	1	Buffer In/out..
I	Input	1	Buffer input.
T	Input	1	3-State enable input.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DRIVE	INTEGER	2, 4, 6, 8, 12, 16, 24	12	Selects output drive strength (mA) for the SelectIO™ buffers that use the LVTTL, LVCMOS12, LVCMOS15, LVCMOS18, LVCMOS25, or LVCMOS33 interface I/O standard.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
SLEW	STRING	"SLOW", "FAST"	"SLOW"	Sets the output rise and fall time.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IOBUF: Single-ended Bi-directional Buffer
--        7 Series
-- Xilinx HDL Language Template, version 2020.1

IOBUF_inst : IOBUF
generic map (
   DRIVE => 12,
   IOSTANDARD => "DEFAULT",
   SLEW => "SLOW")
port map (
   O => O,      -- Buffer output
   IO => IO,    -- Buffer inout port (connect directly to top-level port)
   I => I,      -- Buffer input
   T => T       -- 3-state enable input, high=input, low=output
);

-- End of IOBUF_inst instantiation
```

Verilog Instantiation Template

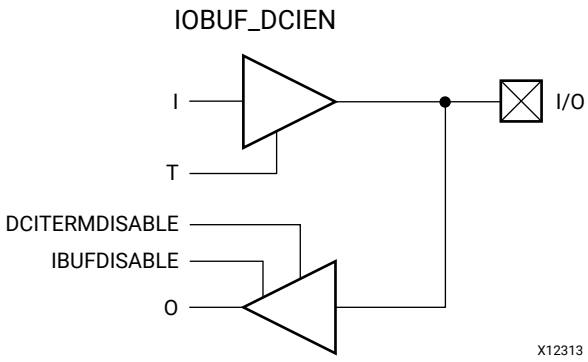
```
// IOBUF: Single-ended Bi-directional Buffer
//        All devices
// Xilinx HDL Language Template, version 2020.1

IOBUF #(
   .DRIVE(12), // Specify the output drive strength
   .IBUF_LOW_PWR("TRUE"),  // Low Power - "TRUE", High Performance = "FALSE"
   .IOSTANDARD("DEFAULT"), // Specify the I/O standard
   .SLEW("SLOW") // Specify the output slew rate
) IOBUF_inst (
   .O(O),     // Buffer output
   .IO(IO),   // Buffer inout port (connect directly to top-level port)
   .I(I),     // Buffer input
   .T(T)      // 3-state enable input, high=input, low=output
);

// End of IOBUF_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IOBUF_DCIEN

Primitive: Bi-Directional Single-ended Buffer with DCI and Input Disable



Introduction

This design element is a bidirectional single ended I/O buffer used to connect internal logic to an external bidirectional pin. This element includes Digitally Controlled Impedance (DCI) termination enable/disable as well as input path disable as additional power saving features when the I/O is either in an unused state or being used as an output for a sustained amount of time. This element may only be placed in High Performance (HP) banks in the 7 series devices.

Port Descriptions

Port	Direction	Width	Function
IO	In/out	1	Bi-directional port connection. Connect directly to top-level port in the design.
I	Input	1	Buffer input representing the output path to the device.
IBUFDISABLE	Input	1	Disables input path. When this signal is asserted HIGH and the attribute USE_IBUFDISABLE is set to "TRUE", the input path through the input buffer is disabled and forced to a logic HIGH. If USE_IBUFDISABLE is set to "FALSE" this input is ignored and should be tied to ground. This feature is generally used to reduce power at times when the I/O is either idle or during sustained write (output) conditions.
DCITERMDISABLE	Input	1	Disables DCI termination. When this signal is asserted HIGH, DCI termination is disabled. This feature is generally used to reduce power at times when the I/O is either idle or during sustained write (output) conditions.
T	Input	1	Sets the I/O in a high impedance 3-state mode when the I/O is being used for a read (input) operation. The T pin also affects the IBUFDISABLE function when USE_IBUFDISABLE = "TRUE".
O	Output	1	Buffer output representing the input path to the device.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DRIVE	INTEGER	2, 4, 6, 8, 12, 16, 24	12	Selects output drive strength (mA) for the SelectIO™ buffers.
IBUF_LOW_PWR	STRING	"TRUE", "FALSE"	"TRUE"	Allows a trade off of lower power consumption vs. highest performance.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
SLEW	STRING	"SLOW", "FAST",	"SLOW"	Sets the output rise and fall time. See the Data Sheet for recommendations of the best setting for this attribute.
USE_IBUFDISABLE	STRING	"TRUE", "FALSE"	"TRUE"	Enables or disables the feature of IBUFDISABLE. Set to FALSE when it is not desirable to have the T pin disable input path to allow a read during write operation. When set to TRUE deasserting T (IO used as output) or asserting IBUFDISABLE will disable the input path through the buffer and forces to a logic high.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IOBUF_DCIEN: Single-ended Bi-directional Buffer with Digital Controlled Impedance (DCI)
--              and Input path enable/disable
--              May only be placed in High Performance (HP) Banks
--              7 Series
-- Xilinx HDL Language Template, version 2020.1

IOBUF_DCIEN_inst : IOBUF_DCIEN
generic map (
   DRIVE => 12,
   IOSTANDARD => "DEFAULT",
   IBUF_LOW_PWR => "TRUE",
   SLEW => "SLOW")
port map (
   O => O,      -- Buffer output
   IO => IO,    -- Buffer inout port (connect directly to top-level port)
   DCITERMDISABLE => DCITERMDISABLE, -- DCI Termination enable input
   I => I,      -- Buffer input
   IBUFDISABLE => IBUFDISABLE, -- Input disable input, high=disable
   T => T       -- 3-state enable input, high=input, low=output
);

-- End of IOBUF_DCIEN_inst instantiation
```

Verilog Instantiation Template

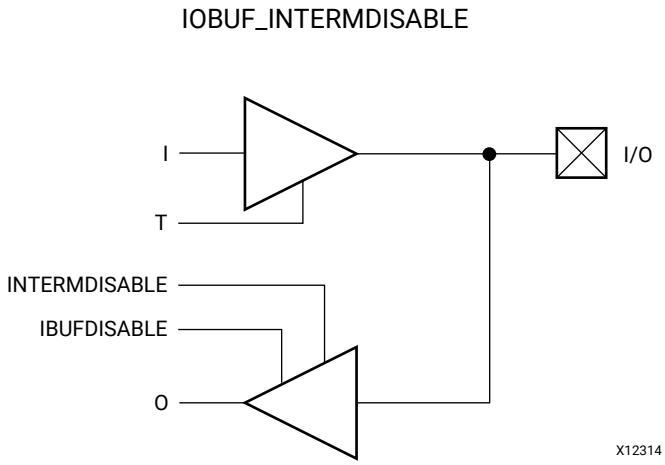
```
// IOBUF_DCIEN: Single-ended Bi-directional Buffer with Digital Controlled Impedance (DCI)
//              and Input path enable/disable
//              May only be placed in High Performance (HP) Banks
//              7 Series
// Xilinx HDL Language Template, version 2020.1

IOBUF_DCIEN #(
   .DRIVE(12), // Specify the output drive strength
   .IBUF_LOW_PWR("TRUE"),  // Low Power - "TRUE", High Performance = "FALSE"
   .IOSTANDARD("DEFAULT"), // Specify the I/O standard
   .SLEW("SLOW"),          // Specify the output slew rate
   .USE_IBUFDISABLE("TRUE") // Use IBUFDISABLE function, "TRUE" or "FALSE"
) IOBUF_DCIEN_inst (
   .O(O),     // Buffer output
   .IO(IO),   // Buffer inout port (connect directly to top-level port)
   .DCITERMDISABLE(DCITERMDISABLE), // DCI Termination enable input
   .I(I),     // Buffer input
   .IBUFDISABLE(IBUFDISABLE), // Input disable input, high=disable
   .T(T)      // 3-state enable input, high=input, low=output
);

// End of IOBUF_DCIEN_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IOBUF_INTERMDISABLE

Primitive: Bi-Directional Single-ended Buffer with Input Termination Disable and Input Path Disable



Introduction

The design element is a bidirectional single-ended I/O Buffer used to connect internal logic to an external bidirectional pin. This element include uncalibrated input termination (INTERM) disable as well as input path disable as additional power saving features when the I/O is either is an unused state or being used as an output for several clock cycles. This element may only be placed in High Range (HR) banks in the 7 series devices.

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Buffer output representing the input path to the device.
IO	In/out	1	Bi-directional port connection. Connect directly to top-level port in the design.
I	Input	1	Buffer input representing the output path to the device.
IBUFDISABLE	Input	1	Disables input path through the buffer and forces to a logic high when USE_IBUFDISABLE is set to "TRUE". If USE_IBUFDISABLE is set to "FALSE" this input is ignored and should be tied to ground. This feature is generally used to reduce power at times when the I/O is either idle or during sustained write (output) conditions.
INTERMDISABLE	Input	1	Disables input termination. This feature is generally used to reduce power at times when the I/O is either idle or during sustained write (output) conditions.
T	Input	1	Sets the I/O in a high impedance 3-state mode when the I/O is being used for a read (input) operation. The T pin also affects the IBUFDISABLE function when USE_IBUFDISABLE = "TRUE". The T pin also disables INTERM when in a write (output) mode.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DRIVE	INTEGER	2, 4, 6, 8, 12, 16, 24	12	Selects output drive strength (mA) for the SelectIO™ buffers.
IBUF_LOW_PWR	STRING	"TRUE", "FALSE"	"TRUE"	Allows a trade off of lower power consumption vs. highest performance.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
SLEW	STRING	"SLOW", "FAST"	"SLOW"	Sets the output rise and fall time. See the Data Sheet for recommendations of the best setting for this attribute.
USE_IBUFDISABLE	STRING	"TRUE", "FALSE"	"TRUE"	Enables or disables the feature of IBUFDISABLE. Generally used when it is not desirable to have the T pin disable input path to allow a read during write operation.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IOBUF_INTERMDISABLE: Single-ended Bi-directional Buffer with Input Termination
--                      and Input path enable/disable
--                      May only be placed in High Range (HR) Banks
--                      7 Series
-- Xilinx HDL Language Template, version 2020.1

IOBUF_INTERMDISABLE_inst : IOBUF_INTERMDISABLE
generic map (
   DRIVE => 12,
   IOSTANDARD => "DEFAULT", -- Specify the I/O standard
   IBUF_LOW_PWR => "TRUE", -- Low Power - "TRUE", High Performance = "FALSE"
   USE_IBUFDISABLE => "TRUE", -- Use IBUFDISABLE function "TRUE" or "FALSE"
   SLEW => "SLOW")
port map (
   O => O,      -- Buffer output
   IO => IO,    -- Buffer inout port (connect directly to top-level port)
   DCITERMDISABLE => DCITERMDISABLE, -- DCI Termination enable input
   I => I,      -- Buffer input
   IBUFDISABLE => IBUFDISABLE, -- Input disable input, high=disable
   INTERMDISABLE => INTERMDISABLE, -- Input termination disable input
   T => T       -- 3-state enable input, high=input, low=output
);

-- End of IOBUF_DCIEN_inst instantiation
```

Verilog Instantiation Template

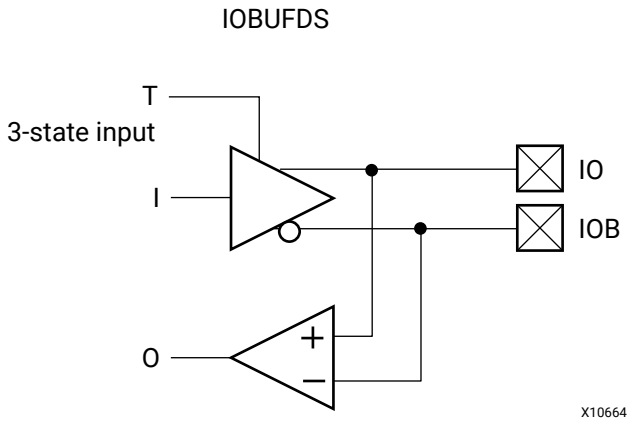
```
// IOBUF_INTERMDISABLE: Single-ended Bi-directional Buffer with Input Termination
//                      and Input path enable/disable
//                      May only be placed in High Range (HR) Banks
//                      7 Series
// Xilinx HDL Language Template, version 2020.1

IOBUF_INTERMDISABLE #(
   .DRIVE(12), // Specify the output drive strength
   .IBUF_LOW_PWR("TRUE"),  // Low Power - "TRUE", High Performance = "FALSE"
   .IOSTANDARD("DEFAULT"), // Specify the I/O standard
   .SLEW("SLOW"),          // Specify the output slew rate
   .USE_IBUFDISABLE("TRUE") // Use IBUFDISABLE function, "TRUE" or "FALSE"
) IOBUF_INTERMDISABLE_inst (
   .O(O),     // Buffer output
   .IO(IO),   // Buffer inout port (connect directly to top-level port)
   .I(I),     // Buffer input
   .IBUFDISABLE(IBUFDISABLE), // Input disable input, high=disable
   .INTERMDISABLE(INTERMDISABLE), // Input termination disable input
   .T(T)      // 3-state enable input, high=input, low=output
);

// End of IOBUF_INTERMDISABLE_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IOBUFDS

Primitive: 3-State Differential Signaling I/O Buffer with active-Low Output Enable



Introduction

The design element is a bidirectional buffer that supports low-voltage, differential signaling. For the IOBUFDS, a design level interface signal is represented as two distinct ports (IO and IOB), one deemed the "master" and the other the "slave." The master and the slave are opposite phases of the same logical signal (for example, MYNET_P and MYNET_N). Optionally, a programmable differential termination feature is available to help improve signal integrity and reduce external components. Also available is a programmable delay is to assist in the capturing of incoming data to the device.

Logic Table

Inputs		Bidirectional		Outputs
I	**T**	**IO**	**IOB**	**O**
X	1	Z	Z	No Change
0	0	0	1	0
I	0	1	0	1

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Buffer output.
IO	In/out	1	Diff_p In/out.
IOB	In/out	1	Diff_n In/out.
I	Input	1	Buffer input.

Port	Direction	Width	Function
T	Input	1	3-state enable input.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	BOOLEAN	TRUE, FALSE	FALSE	Turns the built-in differential termination on (TRUE) or off (FALSE).
IBUF_LOW_PWR	BOOLEAN	TRUE, FALSE	TRUE	When set to TRUE, allows for reduced power when using differential or referenced (requiring V_{REF}) input standards like LVDS or HSTL. A setting of FALSE demands more power but delivers higher performance characteristics. Consult the*7 Series FPGA SelectIO Resources User Guide* (UG471) for details.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
SLEW	STRING	"SLOW" or "FAST"	"SLOW"	Specifies the slew rate of the output driver. Consult the product Data Sheet for recommendations of the best setting for this attribute.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IOBUFDS: Differential Bi-directional Buffer
--          7 Series
-- Xilinx HDL Language Template, version 2020.1

IOBUFDS_inst : IOBUFDS
generic map (
   DIFF_TERM => FALSE, -- Differential Termination (TRUE/FALSE)
   IBUF_LOW_PWR => TRUE, -- Low Power - TRUE, High Performance = FALSE
   IOSTANDARD => "BLVDS_25", -- Specify the I/O standard
   SLEW => "SLOW")       -- Specify the output slew rate
port map (
   O => O,     -- Buffer output
   IO => IO,   -- Diff_p inout (connect directly to top-level port)
   IOB => IOB, -- Diff_n inout (connect directly to top-level port)
   I => I,     -- Buffer input
   T => T      -- 3-state enable input, high=input, low=output
);

-- End of IOBUFDS_inst instantiation
```

Verilog Instantiation Template

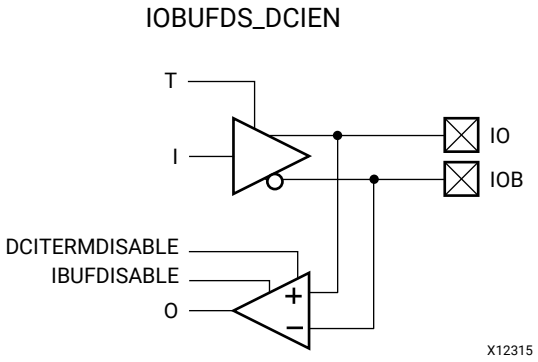
```
// IOBUFDS: Differential Bi-directional Buffer
//          7 Series
// Xilinx HDL Language Template, version 2020.1

IOBUFDS #(
   .DIFF_TERM("FALSE"),     // Differential Termination ("TRUE"/"FALSE")
   .IBUF_LOW_PWR("TRUE"),   // Low Power - "TRUE", High Performance = "FALSE"
   .IOSTANDARD("BLVDS_25"), // Specify the I/O standard
   .SLEW("SLOW")            // Specify the output slew rate
) IOBUFDS_inst (
   .O(O),     // Buffer output
   .IO(IO),   // Diff_p inout (connect directly to top-level port)
   .IOB(IOB), // Diff_n inout (connect directly to top-level port)
   .I(I),     // Buffer input
   .T(T)      // 3-state enable input, high=input, low=output
);

// End of IOBUFDS_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IOBUFDS_DCIEN

Primitive: Bi-Directional Differential Buffer with DCI Enable/Disable and Input Disable



Introduction

This design element is a bidirectional differential I/O buffer used to connect internal logic to an external bidirectional pin. This element includes Digitally Controlled Impedance (DCI) termination enable/disable as well as input path disable as additional power saving features when the I/O is either in an unused state or being used as an output for a sustained amount of time. This element may only be placed in High Performance (HP) banks in the 7 series devices.

Port Descriptions

Port	Direction	Width	Function
IO	In/out	1	Bi-directional p-side port connection. Connect directly to top-level port in the design.
IOB	In/out	1	Bi-directional p-side port connection. Connect directly to top-level port in the design.
I	Input	1	Buffer input representing the output path to the device.
IBUFDISABLE	Input	1	Disables input path. When this signal is asserted HIGH and the attribute USE_IBUFDISABLE is set to "TRUE", the input path through the input buffer is disabled and forced to a logic HIGH.. If USE_IBUFDISABLE is set to "FALSE" this input is ignored and should be tied to ground. This feature is generally used to reduce power at times when the I/O is either idle or during sustained write (output) conditions.
DCITERMDISABLE	Input	1	Disables DCI termination. When this signal is asserted HIGH, DCI termination is disabled. This feature is generally used to reduce power at times when the I/O is either idle or during sustained write (output) conditions.
T	Input	1	Sets the I/O in a high impedance 3-state mode when the I/O is being used for a read (input) operation. The T pin also affects the IBUFDISABLE function when USE_IBUFDISABLE = "TRUE".
O	Output	1	Buffer output representing the input path to the device.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	STRING	"TRUE", "FALSE"	"FALSE"	Turns the built-in differential termination on (TRUE) or off (FALSE).
IBUF_LOW_PWR	STRING	"TRUE", "FALSE"	"TRUE"	Allows a trade off of lower power consumption vs. highest performance.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
SLEW	STRING	"SLOW", "FAST",	"SLOW"	Sets the output rise and fall time. See the Data Sheet for recommendations of the best setting for this attribute.
USE_IBUFDISABLE	STRING	"TRUE", "FALSE"	"TRUE"	Enables or disables the feature of IBUFDISABLE. Set to FALSE when it is not desirable to have the T pin disable input path to allow a read during write operation. When set to TRUE deasserting T (IO used as output) or asserting IBUFDISABLE will disable the input path through the buffer and forces to a logic high.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IOBUFDS_DCIEN: Differential Bi-directional Buffer with Digital Controlled Impedance (DCI)
--                and Input path enable/disable
--                May only be placed in High Performance (HP) Banks
--                7 Series
-- Xilinx HDL Language Template, version 2020.1

IOBUFDS_DCIEN_inst : IOBUFDS_DCIEN
generic map (
   DIFF_TERM => "FALSE", -- Differential termination (TRUE/FALSE)
   IBUF_LOW_PWR => "TRUE", -- Low Power - TRUE, HIGH Performance = FALSE
   IOSTANDARD => "BLVDS_25", -- Specify the I/O standard
   SLEW => "SLOW",   -- Specify the output slew rate
   USE_IBUFDISABLE => "TRUE") -- Use IBUFDISABLE function "TRUE" or "FALSE"
port map (
   O => O,     -- Buffer output
   IO => IO,   -- Diff_p inout (connect directly to top-level port)
   IOB => IOB, -- Diff_n inout (connect directly to top-level port)
   DCITERMDISABLE => DCITERMDISABLE, -- DCI Termination enable input
   I => I,     -- Buffer input
```

```
   IBUFDISABLE => IBUFDISABLE, -- Input disable input, high=disable
   T => T      -- 3-state enable input, high=input, low=output
);

-- End of IOBUFDS_DCIEN_inst instantiation
```

Verilog Instantiation Template

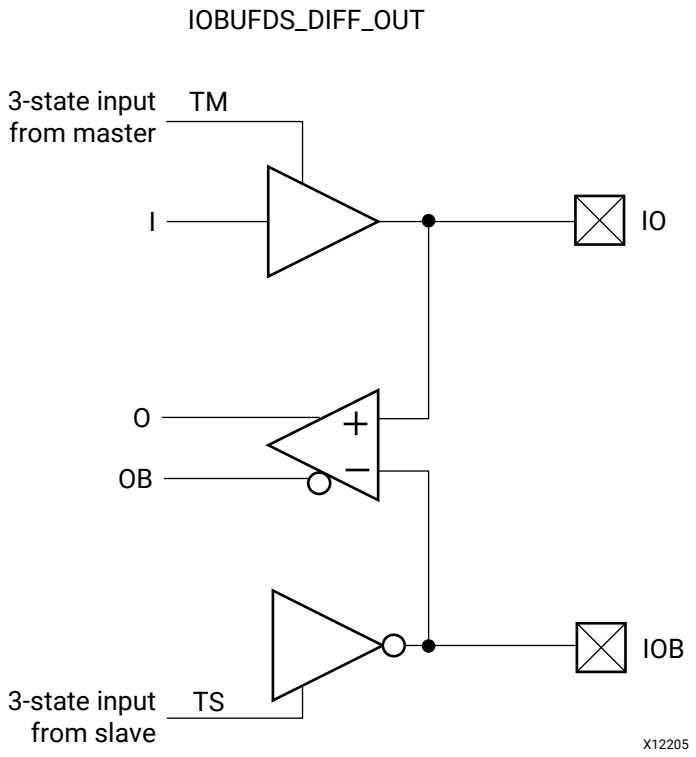
```
// IOBUFDS_DCIEN: Differential Bi-directional Buffer with Digital Controlled Impedance (DCI)
//                and Input path enable/disable
//                May only be placed in High Performance (HP) Banks
//                7 Series
// Xilinx HDL Language Template, version 2020.1

IOBUFDS_DCIEN #(
   .DIFF_TERM("FALSE"),     // Differential Termination ("TRUE"/"FALSE")
   .IBUF_LOW_PWR("TRUE"),   // Low Power - "TRUE", High Performance = "FALSE"
   .IOSTANDARD("BLVDS_25"), // Specify the I/O standard
   .SLEW("SLOW"),           // Specify the output slew rate
   .USE_IBUFDISABLE("TRUE") // Use IBUFDISABLE function, "TRUE" or "FALSE"
) IOBUFDS_DCIEN_inst (
   .O(O),     // Buffer output
   .IO(IO),   // Diff_p inout (connect directly to top-level port)
   .IOB(IOB), // Diff_n inout (connect directly to top-level port)
   .DCITERMDISABLE(DCITERMDISABLE), // DCI Termination enable input
   .I(I),                           // Buffer input
   .IBUFDISABLE(IBUFDISABLE),       // Input disable input, high=disable
   .T(T)      // 3-state enable input, high=input, low=output
);

// End of IOBUFDS_DCIEN_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IOBUFDS_DIFF_OUT

Primitive: Differential Bi-directional Buffer with Differential Output



Introduction

This design element is a bidirectional buffer that supports low-voltage, differential signaling. For the IOBUFDS_DIFF_OUT, a design level interface signal is represented as two distinct ports (IO and IOB), one deemed the "master" and the other the "slave." The master and the slave are opposite phases of the same logical signal (for example, MYNET_P and MYNET_N). The IOBUFDS_DIFF_OUT differs from the IOBUFDS in that it allows internal access to both phases of the differential signal. Optionally, a programmable differential termination feature is available to help improve signal integrity and reduce external components.

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Buffer p-side output.
OB	Output	1	Buffer n-side output.
IO	In/out	1	Diff_p In/out (connect directly to top-level port).
IOB	In/out	1	Diff_n In/out (connect directly to top-level port).

Port	Direction	Width	Function
I	Input	1	Buffer input.
TM	Input	1	3-state enable input from master OLOGIC, high=input, low=output.
TS	Input	1	3-state enable input from slave OLOGIC, high=input, low=output.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	BOOLEAN	TRUE, FALSE	FALSE	Turns the built-in differential termination on (TRUE) or off (FALSE).
IBUF_LOW_PWR	BOOLEAN	TRUE, FALSE	TRUE	When set to TRUE, allows for reduced power when using differential or referenced (requiring V_{REF}) input standards like LVDS or HSTL. A setting of FALSE demands more power but delivers higher performance characteristics. Consult the*7 Series FPGA SelectIO Resources User Guide* (UG471) for details.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IOBUFDS_DIFF_OUT: Differential Bi-directional Buffer with Diffirential Output
--                   7 Series
-- Xilinx HDL Language Template, version 2020.1

IOBUFDS_DIFF_OUT_inst : IOBUFDS_DIFF_OUT
generic map (
   DIFF_TERM => FALSE, -- Differential Termination (TRUE/FALSE)
   IBUF_LOW_PWR => TRUE, -- Low Power - TRUE, High Performance = FALSE
   IOSTANDARD => "BLVDS_25") -- Specify the I/O standard
port map (
   O => O,      -- Buffer p-side output
   OB => OB,    -- Buffer n-side output
   IO => IO,    -- Diff_p inout (connect directly to top-level port)
   IOB => IOB, -- Diff_n inout (connect directly to top-level port)
   I => I,      -- Buffer input
```

```
   TM => TM,    -- 3-state enable input, high=input, low=output
   TS => TS     -- 3-state enable input, high=input, low=output
);

-- End of IOBUFDS_DIFF_OUT_inst instantiation
```

Verilog Instantiation Template

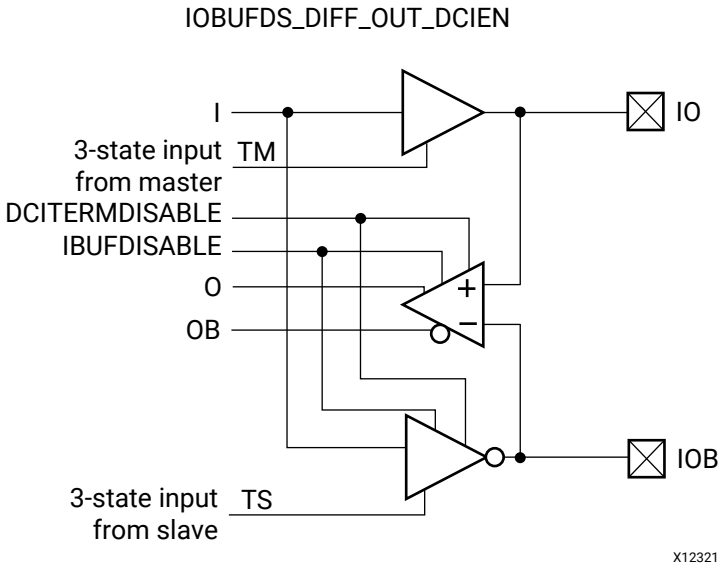
```
// IOBUFDS_DIFF_OUT: Differential Bi-directional Buffer with Differential Output
//                   7 Series
// Xilinx HDL Language Template, version 2020.1

IOBUFDS_DIFF_OUT #(
   .DIFF_TERM("FALSE"),     // Differential Termination ("TRUE"/"FALSE")
   .IBUF_LOW_PWR("TRUE"),   // Low Power - "TRUE", High Performance = "FALSE"
   .IOSTANDARD("BLVDS_25") // Specify the I/O standard
) IOBUFDS_DIFF_OUT_inst (
   .O(O),      // Buffer p-side output
   .OB(OB),    // Buffer n-side output
   .IO(IO),    // Diff_p inout (connect directly to top-level port)
   .IOB(IOB), // Diff_n inout (connect directly to top-level port)
   .I(I),      // Buffer input
   .TM(TM),    // 3-state enable input, high=input, low=output
   .TS(TS)     // 3-state enable input, high=input, low=output
);

// End of IOBUFDS_DIFF_OUT_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IOBUFDS_DIFF_OUT_DCIEN

Primitive: Bi-Directional Differential Buffer with DCI Disable, Input Disable, and Differential Output



Introduction

This design element is a bidirectional differential I/O buffer used to connect internal logic to an external bidirectional pin. This element includes Digitally Controlled Impedance (DCI) termination enable/ disable as well as input path disable as additional power saving features when the I/O is in an unused state or being used as an output for a sustained period of time. The IOBUFDS_DIFF_OUT_DCIEN differs from the IOBUFDS_DCIEN in that it allows internal access to both phases of the differential signal. This element may only be placed in High Performance (HP) banks in the 7 series devices.

Port Descriptions

Port	Direction	Width	Function
IO	In/out	1	Bi-directional p-side port connection. Connect directly to top-level port in the design.
IOB	In/out	1	Bi-directional n-side port connection. Connect directly to top-level port in the design.
I	Input	1	Buffer input representing the output path to the device.

Port	Direction	Width	Function
IBUFDISABLE	Input	1	Disables input path. • When this signal is asserted HIGH and the attribute USE_IBUFDISABLE is set to "TRUE", the input path through the input buffer is disabled and forced to a logic HIGH. • If USE_IBUFDISABLE is set to "FALSE" this input is ignored and should be tied to ground. This feature is generally used to reduce power at times when the I/O is either idle or during sustained write (output) conditions
DCITERMDISABLE	Input	1	Disables DCI termination. When this signal is asserted HIGH, DCI termination is disabled. This feature is generally used to reduce power at times when the I/O is either idle or during sustained write (output) conditions.
TM	Input	1	P-side or master side of the high impedance 3-state mode when the I/O is being used for a read (input) operation. The TM pin also affects the IBUFDISABLE function when USE_IBUFDISABLE = "TRUE".
TS	Input	1	N-side or slave side of the high impedance 3-state mode when the I/O is being used for a read (input) operation. The TM pin also affects the IBUFDISABLE function when USE_IBUFDISABLE = "TRUE".
O	Output	1	Buffer p-side output representing the input path to the device.
OB	Output	1	Buffer n-side output representing the input path to the device.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	STRING	"TRUE", "FALSE"	"FALSE"	Turns the built-in differential termination on (TRUE) or off (FALSE).
IBUF_LOW_PWR	STRING	"TRUE", "FALSE"	"TRUE"	Allows a trade off of lower power consumption vs highest performance.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
USE_IBUFDISABLE	STRING	"TRUE", "FALSE"	"TRUE"	Enables or disables the feature of IBUFDISABLE. Set to FALSE when it is not desirable to have the T pin disable input path to allow a read during write operation. When set to TRUE deasserting T (IO used as output) or asserting IBUFDISABLE will disable the input path through the buffer and forces to a logic high.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IOBUFDS_DIFF_OUT_DCIEN: Differential Bi-directional Buffer with Differential Output,
--                         Digital Controlled Impedance (DCI)and Input path enable/disable
--                         May only be placed in High Performance (HP) Banks
--                         7 Series
-- Xilinx HDL Language Template, version 2020.1

IOBUFDS_DIFF_OUT_DCIEN_inst : IOBUFDS_DIFF_OUT_DCIEN
generic map (
   DIFF_TERM => "FALSE", -- Differential Termination (TRUE/FALSE)
   IBUF_LOW_PWR => "TRUE", -- Low Power - TRUE, High Performance = FALSE
   IOSTANDARD => "BLVDS_25", -- Specify the I/O standard
   USE_IBUFDISABLE => "TRUE") -- Use IBUFDISABLE function, "TRUE" or "FALSE"
port map (
   O => O,      -- Buffer p-side output
   OB => OB,    -- Buffer n-side output
   IO => IO,    -- Diff_p inout (connect directly to top-level port)
   IOB => IOB, -- Diff_n inout (connect directly to top-level port)
   DCITERMDISABLE => DCITERMDISABLE, -- DCI Termination enable input
   I => I,      -- Buffer input
   IBUFTERMDISABLE => IBUFTERMDISABLE, -- input disable input, low=disable
   TM => TM,    -- 3-state enable input, high=input, low=output
   TS => TS     -- 3-state enable input, high=output, low=input
);

-- End of IOBUFDS_DIFF_OUT_DCIEN_inst instantiation
```

Verilog Instantiation Template

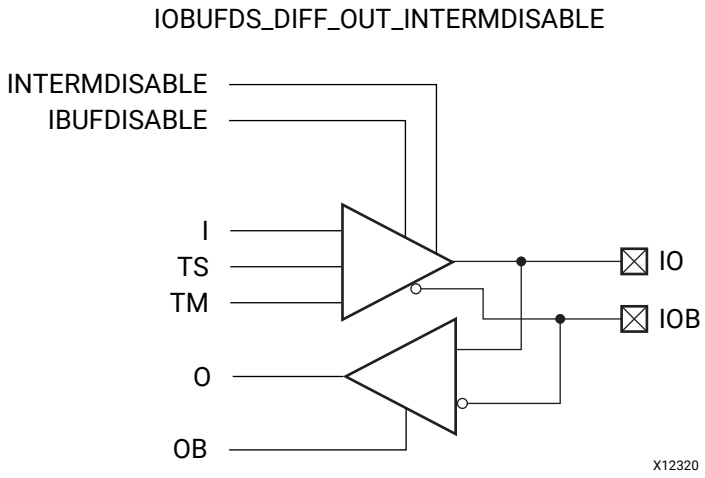
```
// IOBUFDS_DIFF_OUT_DCIEN: Differential Bi-directional Buffer with Differential Output,
//                         Digital Controlled Impedance (DCI)and Input path enable/disable
//                         May only be placed in High Performance (HP) Banks
//                         7 Series
// Xilinx HDL Language Template, version 2020.1

IOBUFDS_DIFF_OUT_DCIEN #(
   .DIFF_TERM("FALSE"),     // Differential Termination ("TRUE"/"FALSE")
   .IBUF_LOW_PWR("TRUE"),   // Low Power - "TRUE", High Performance = "FALSE"
   .IOSTANDARD("BLVDS_25"), // Specify the I/O standard
   .USE_IBUFDISABLE("TRUE") // Use IBUFDISABLE function, "TRUE" or "FALSE"
) IOBUFDS_DIFF_OUT_DCIEN_inst (
   .O(O),     // Buffer p-side output
   .OB(OB),   // Buffer n-side output
   .IO(IO),   // Diff_p inout (connect directly to top-level port)
   .IOB(IOB), // Diff_n inout (connect directly to top-level port)
   .DCITERMDISABLE(DCITERMDISABLE), //  DCI Termination enable input
   .I(I),                           // Buffer input
   .IBUFDISABLE(IBUFDISABLE),       // Input disable input, high=disable
   .TM(TM),   // 3-state enable input, high=input, low=output
   .TS(TS)    // 3-state enable input, high=input, low=output
);

// End of IOBUFDS_DIFF_OUT_DCIEN_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IOBUFDS_DIFF_OUT_INTERMDISABLE

Primitive: Bi-Directional Differential Buffer with Input Termination Disable, Input Disable, and Differential Output



Introduction

This design element is a bidirectional differential I/O Buffer used to connect internal logic to an external bidirectional pin. This element includes an uncalibrated input termination (INTERM) disable as well as input path disable as additional power saving features when the I/O is either is an unused state or being used as an output for several clock cycles. The IOBUFDS_DIFF_OUT_INTERMDISABLE differs from the IOBUFDS_INTERMDISABLE in that it allows internal access to both phases of the differential signal. This element may only be placed in High Range (HR) banks in the 7 series devices.

Port Descriptions

Port	Direction	Width	Function
IO	In/out	1	Bi-directional p-side port connection. Connect directly to top-level port in the design.
IOB	In/out	1	Bi-directional n-side port connection. Connect directly to top-level port in the design.
I	Input	1	Buffer input representing the output path to the device.
IBUFDISABLE	Input	1	Disables input path through the buffer and forces to a logic high when USE_IBUFDISABLE is set to "TRUE". If USE_IBUFDISABLE is set to "FALSE" this input is ignored and should be tied to ground. This feature is generally used to reduce power at times when the I/O is either idle or during sustained write (output) conditions.
INTERMDISABLE	Input	1	Disables input termination. This feature is generally used to reduce power at times when the I/O is either idle or during sustained write (output) conditions.

Port	Direction	Width	Function
TM	Input	1	P-side or master side of the high impedance 3-state mode when the I/O is being used for a read (input) operation. The TM pin also affects the IBUFDISABLE function when USE_IBUFDISABLE = "TRUE", and disables INTERM when in a write (output) mode.
TS	Input	1	N-side or slave side of the high impedance 3-state mode when the I/O is being used for a read (input) operation. The TS pin also affects the IBUFDISABLE function when USE_IBUFDISABLE = "TRUE", and disables INTERM when in a write (output) mode.
O	Output	1	Buffer p-side output representing the input path to the device.
OB	Output	1	Buffer n-side output representing the input path to the device.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	STRING	"TRUE", "FALSE"	"FALSE"	Turns the built-in differential termination on (TRUE) or off (FALSE).
IBUF_LOW_PWR	STRING	"TRUE", "FALSE"	"TRUE"	Allows a trade off of lower power consumption vs. highest performance when referenced I/O standards are used.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
USE_IBUFDISABLE	STRING	"TRUE", "FALSE"	"TRUE"	Enables or disables the feature of IBUFDISABLE. Generally used when it is not desirable to have the T pin disable input path to allow a read during write operation.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IOBUFDS_DIFF_OUT_INTERMDISABLE:  Differential Global Clock Buffer with Differential Output
--                                  Input Termination and Input Path Disable
--                                  May only be placed in High Range (HR) Banks
--                                  7 Series
-- Xilinx HDL Language Template, version 2020.1

IOBUFDS_DIFF_OUT_INTERMDISABLE_inst : IOBUFDS_DIFF_OUT_INTERMDISABLE
```

```
generic map (
   DIFF_TERM => "FALSE", -- Differential Termination (TRUE/FALSE)
   IBUF_LOW_PWR => "TRUE", -- Low Power - TRUE, High Performance = FALSE
   IOSTANDARD => "BLVDS_25", -- Specify the I/O standard
   USE_IBUFDISABLE => "TRUE") -- Use IBUFDISABLE function, "TRUE" or "FALSE"
port map (
   O => O,      -- Buffer p-side output
   OB => OB,    -- Buffer n-side output
   IO => IO,    -- Diff_p inout (connect directly to top-level port)
   IOB => IOB, -- Diff_n inout (connect directly to top-level port)
   I => I,      -- Buffer input
   IBUFDISABLE => IBUFDISABLE, -- input disable input, high=disable
   INTERMDISABLE => INTERMDISABLE, -- Input termination disable input
   TM => TM,    -- 3-state enable input, high=input, low=output
   TS => TS     -- 3-state enable input, high=output, low=input
);

-- End of IOBUFDS_DIFF_OUT_INTERMDISABLE_inst instantiation
```

Verilog Instantiation Template

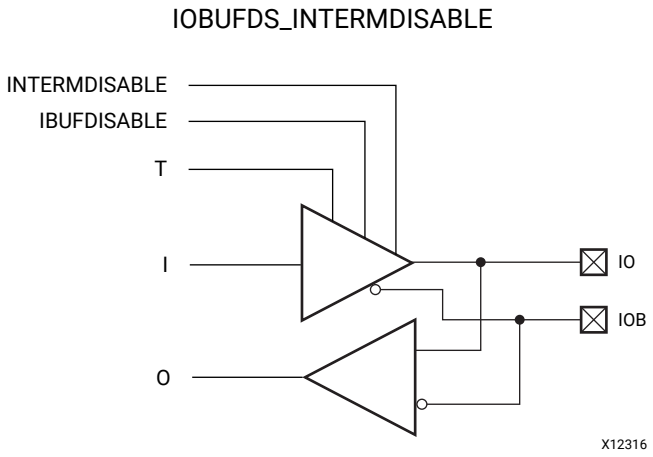
```
// IOBUFDS_DIFF_OUT_INTERMDISABLE: Differential Global Clock Buffer with Differential Output
//                                 Input Termination and Input Path Disable
//                                 May only be placed in High Range (HR) Banks
//                                 7 Series
// Xilinx HDL Language Template, version 2020.1

IOBUFDS_DIFF_OUT_INTERMDISABLE #(
   .DIFF_TERM("FALSE"),     // Differential Termination, "TRUE"/"FALSE"
   .IBUF_LOW_PWR("TRUE"),   // Low power="TRUE", Highest performance="FALSE"
   .IOSTANDARD("DEFAULT"),  // Specify the input I/O standard
   .USE_IBUFDISABLE("TRUE") // Set to "TRUE" to enable IBUFDISABLE feature
) IOBUFDS_DIFF_OUT_INTERMDISABLE_inst (
   .O(O),     // Buffer p-side output
   .OB(OB),   // Buffer n-side output
   .IO(IO),   // Diff_p inout (connect directly to top-level port)
   .IOB(IOB), // Diff_n inout (connect directly to top-level port)
   .I(I),     // Buffer input
   .INTERMDISABLE(INTERMDISABLE), //  Input termination disable input
   .IBUFDISABLE(IBUFDISABLE),     // Input disable input, high=disable
   .TM(TM),   // 3-state enable input, high=input, low=output
   .TS(TS)    // 3-state enable input, high=input, low=output
);

// End of IOBUFDS_DIFF_OUT_INTERMDISABLE_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

IOBUFDS_INTERMDISABLE

Primitive: Bi-Directional Differential Buffer with Input Termination Disable and Input Disable



Introduction

This design element is a bidirectional differential I/O buffer used to connect internal logic to an external bidirectional pin. This element includes an uncalibrated input termination (INTERM) disable as well as an input path disable as additional power saving features when the I/O is either is an unused state or being used as an output for a sustained amount of time. This element may only be placed in High Range (HR) banks in 7 series devices.

Port Descriptions

Port	Direction	Width	Function
IO	In/out	1	Bi-directional p-side port connection. Connect directly to top-level port in the design.
IOB	In/out	1	Bi-directional n-side port connection. Connect directly to top-level port in the design.
I	Input	1	Buffer input representing the output path to the device.
IBUFDISABLE	Input	1	Disables input path through the buffer and forces to a logic high when USE_IBUFDISABLE is set to "TRUE". If USE_IBUFDISABLE is set to "FALSE" this input is ignored and should be tied to ground. This feature is generally used to reduce power at times when the I/O is either idle or during sustained write (output) conditions.
INTERMDISABLE	Input	1	Disables input termination. This feature is generally used to reduce power at times when the I/O is either idle or during sustained write (output) conditions.
T	Input	1	Sets the I/O in a high impedance 3-state mode when the I/O is being used for a read (input) operation. The T pin also affects the IBUFDISABLE function when USE_IBUFDISABLE = "TRUE". The T pin also disables INTERM when in a write (output) mode.

Port	Direction	Width	Function
O	Output	1	Buffer output representing the input path to the device.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DIFF_TERM	STRING	"TRUE", "FALSE"	"FALSE"	Turns the built-in differential termination on (TRUE) or off (FALSE).
IBUF_LOW_PWR	STRING	"TRUE", "FALSE"	"TRUE"	Allows a trade off of lower power consumption vs. highest performance when referenced I/O standards are used.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
SLEW	STRING	"SLOW" or "FAST"	"SLOW"	Sets the output rise and fall time. See the Data Sheet for recommendations of the best setting for this attribute.
USE_IBUFDISABLE	STRING	"TRUE", "FALSE"	"TRUE"	Enables or disables the feature of IBUFDISABLE. Generally used when it is not desirable to have the T pin disable input path to allow a read during write operation.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- IOBUFDS_INTERMDISABLE: Differential Bi-directional Buffer with Input Termination
--                        and Input path enable/disable
--                        May only be placed in High Range (HR) Banks
--                        7 Series
-- Xilinx HDL Language Template, version 2020.1

IOBUFDS_INTERMDISABLE_inst : IOBUFDS_INTERMDISABLE
generic map (
   DIFF_TERM => "FALSE", -- Differential termination (TRUE/FALSE)
   IBUF_LOW_PWR => "TRUE", -- Low Power - TRUE, HIGH Performance = FALSE
   IOSTANDARD => "BLVDS_25", -- Specify the I/O standard
   SLEW => "SLOW",   -- Specify the output slew rate
   USE_IBUFDISABLE => "TRUE") -- Use IBUFDISABLE function "TRUE" or "FALSE"
port map (
   O => O,     -- Buffer output
   IO => IO,   -- Diff_p inout (connect directly to top-level port)
   IOB => IOB, -- Diff_n inout (connect directly to top-level port)
   DCITERMDISABLE => DCITERMDISABLE, -- DCI Termination enable input
   I => I,     -- Buffer input
   IBUFDISABLE => IBUFDISABLE, -- Input disable input, high=disable
```

```
   INTERMDISABLE => INTERMDISABLE, -- Input termination disable input
   T => T      -- 3-state enable input, high=input, low=output
);

-- End of IOBUFDS_INTERMDISABLE_inst instantiation
```

Verilog Instantiation Template

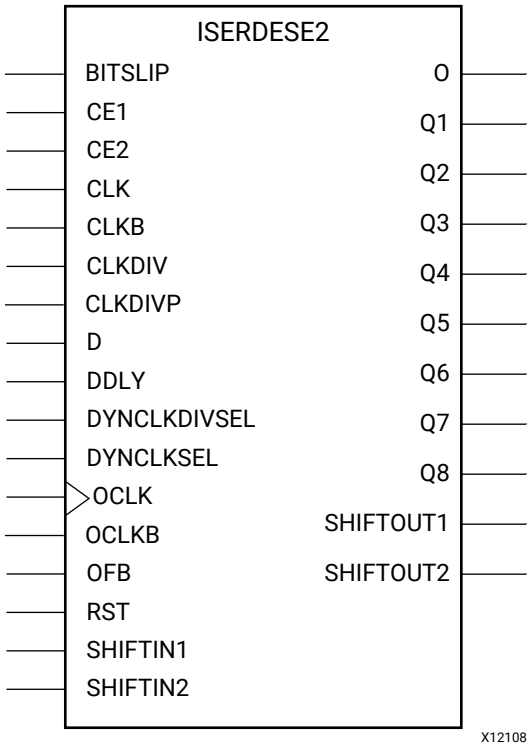
```
// IOBUFDS_INTERMDISABLE: Differential Bi-directional Buffer with Input Termination
//                        and Input path enable/disable
//                        May only be placed in High Range (HR) Banks
//                        7 Series
// Xilinx HDL Language Template, version 2020.1

IOBUFDS_INTERMDISABLE #(
   .DIFF_TERM("FALSE"),     // Differential Termination ("TRUE"/"FALSE")
   .IBUF_LOW_PWR("TRUE"),   // Low Power - "TRUE", High Performance = "FALSE"
   .IOSTANDARD("BLVDS_25"), // Specify the I/O standard
   .SLEW("SLOW"),           // Specify the output slew rate
   .USE_IBUFDISABLE("TRUE") // Use IBUFDISABLE function, "TRUE" or "FALSE"
) IOBUFDS_INTERMDISABLE_inst (
   .O(O),     // Buffer output
   .IO(IO),   // Diff_p inout (connect directly to top-level port)
   .IOB(IOB), // Diff_n inout (connect directly to top-level port)
   .I(I),     // Buffer input
   .IBUFDISABLE(IBUFDISABLE),     // Input disable input, high=disable
   .INTERMDISABLE(INTERMDISABLE), // Input termination disable input
   .T(T)      // 3-state enable input, high=input, low=output
);

// End of IOBUFDS_INTERMDISABLE_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

ISERDESE2

Primitive: Input SERial/DESerializer with Bitslip



Introduction

The ISERDESE2 in 7 series FPGAs is a dedicated serial-to-parallel converter with specific clocking and logic features designed to facilitate the implementation of high-speed source-synchronous applications. The ISERDESE2 avoids the additional timing complexities encountered when designing deserializers in the FPGA fabric. ISERDESE2 features include:

- Dedicated Deserializer/Serial-to-Parallel Converter, which enables high-speed data transfer without requiring the FPGA fabric to match the input data frequency. This converter supports both single data rate (SDR) and double data rate (DDR) modes. In SDR mode, the serial-to-parallel converter creates a 2-, 3-, 4-, 5-, 6-, 7-, or 8-bit wide parallel word. In DDR mode, the serial-to-parallel converter creates a 4-, 6-, 8-, 10-, or 14-bit-wide parallel word.
- Bitslip Submodule, which lets designers reorder the sequence of the parallel data stream going into the FPGA fabric. This can be used for training source-synchronous interfaces that include a training pattern.
- Dedicated Support for Strobe-based Memory Interfaces, including the OCLK input pin, to handle the strobe-to-FPGA clock domain crossover entirely within the ISERDESE2 block. This allows for higher performance and a simplified implementation.

- Dedicated Support for Networking Interfaces
- Dedicated Support for Memory Interfaces

Port Descriptions

Port	Direction	Width	Function
BITSLIP	Input	1	The BITSLIP pin performs a Bitslip operation synchronous to CLKDIV when asserted (active-High). Subsequently, the data seen on the Q1 to Q8 output ports will shift, as in a barrel-shifter operation, one position every time Bitslip is invoked (DDR operation is different from SDR).
CE1, CE2	Input	1	Each ISERDESE2 block contains an input clock enable module. • When NUM_CE = 1, the CE2 input is not used, and the CE1 input is an active-High clock enable connected directly to the input registers in the ISERDESE2. • When NUM_CE = 2, the CE1 and CE2 inputs are both used, with CE1 enabling the ISERDESE2 for half of a CLKDIV cycle, and CE2 enabling the ISERDESE2 for the other half. The clock enable module functions as a 2:1 serial-to-parallel converter, clocked by CLKDIV. The clock enable module is needed specifically for bidirectional memory interfaces when ISERDESE2 is configured for 1:4 deserialization in DDR mode. • When the attribute NUM_CE = 2, the clock enable module is enabled and both CE1 and CE2 ports are available. • When NUM_CE = 1, only CE1 is available and functions as a regular clock enable.
CLK	Input	1	The high-speed clock input (CLK) is used to clock in the input serial data stream.
CLKB	Input	1	The high-speed secondary clock input (CLKB) is used to clock in the input serial data stream. In any mode other than "MEMORY_QDR", connect CLKB to an inverted version of CLK. In "MEMORY_QDR" mode CLKB should be connected to a unique, phase shifted clock.
CLKDIV	Input	1	The divided clock input (CLKDIV) is typically a divided version of CLK (depending on the width of the implemented deserialization). It drives the output of the serial-to-parallel converter, the Bitslip submodule, and the CE module.
CLKDIVP	Input	1	Only supported in MIG. Sourced by PHASER_IN divided CLK in MEMORY_DDR3 mode. All other modes connect to ground.
D	Input	1	The serial input data port (D) is the serial (high-speed) data input port of the ISERDESE2. This port works in conjunction only with the 7 series FPGA I/O resource.
DDLY	Input	1	The serial input data port (DDLY) is the serial (high-speed) data input port of the ISERDESE2. This port works in conjunction only with the 7 series FPGA IDELAYE2 resource.
DYNCLKDIVSEL	Input	1	Dynamically select CLKDIV inversion.
DYNCLKSEL	Input	1	Dynamically select CLK and CLKB inversion.

Port	Direction	Width	Function
O	Output	1	The combinatorial output port (O) is an unregistered output of the ISERDESE2 module. This output can come directly from the data input (D), or from the data input (DDLY) via the IDELAYE2.
OCLK	Input	1	The OCLK clock input synchronizes data transfer in strobe-based memory interfaces. The OCLK clock is only used when INTERFACE_TYPE is set to "MEMORY". The OCLK clock input is used to transfer strobe-based memory data onto a free-running clock domain. OCLK is a free-running FPGA clock at the same frequency as the strobe on the CLK input. The timing of the domain transfer is set by the user by adjusting the delay of the strobe signal to the CLK input (e.g., using IDELAY). Examples of setting the timing of this domain transfer are given in the Memory Interface Generator (MIG). When INTERFACE_TYPE is "NETWORKING", this port is unused and should be connected to GND.
OCLKB	Input	1	The OCLK clock input synchronizes data transfer in strobe-based memory interfaces. The OCLKB clock is only used when INTERFACE_TYPE is set to "MEMORY".
OFB	Input	1	The serial input data port (OFB) is the serial (high-speed) data input port of the ISERDESE2. This port works in conjunction only with the 7 series FPGA OSERDESE2 port OFB.
Q1 - Q8	Output	1	The output ports Q1 to Q8 are the registered outputs of the ISERDESE2 module. One ISERDESE2 block can support up to eight bits (i.e., a 1:8 deserialization). Bit widths greater than eight (up to 14) can be supported using Width Expansion. The first data bit received appears on the highest order Q output. The bit ordering at the input of an OSERDESE2 is the opposite of the bit ordering at the output of an ISERDESE2 block. For example, the least significant bit A of the word FEDCBA is placed at the D1 input of an OSERDESE2, but the same bit A emerges from the ISERDESE2 block at the Q8 output. In other words, D1 is the least significant input to the OSERDESE2, while Q8 is the least significant output of the ISERDESE2 block. When width expansion is used, D1 of the master OSERDESE2 is the least significant input, while Q7 of the slave ISERDESE2 block is the least significant output.
RST	Input	1	The reset input causes the outputs of all data flip-flops in the CLK and CLKDIV domains to be driven low asynchronously. ISERDESE2 circuits running in the CLK domain where timing is critical use an internal, dedicated circuit to retime the RST input to produce a reset signal synchronous to the CLK domain. Similarly, there is a dedicated circuit to retime the RST input to produce a reset signal synchronous to the CLKDIV domain. Because the ISERDESE2 is driven into reset asynchronously but comes out of reset synchronously it must be treated as a synchronous reset to the CLKDIV time domain and have a minimum pulse of one CLKDIV cycle. When building an interface consisting of multiple ISERDESE2 ports, all ISERDESE2 ports in the interface must be synchronized. The internal retiming of the RST input is designed so that all ISERDESE2 blocks that receive the same reset pulse come out of reset synchronized with one another.
SHIFTIN1, SHIFTIN2	Input	1	If SERDES_MODE="SLAVE", connect SHIFTIN1/2 to the master ISERDESE2 SHIFTOUT1/2 outputs. Otherwise, leave SHIFTOUT1/2 unconnected and/or SHIFTIN1/2 grounded.

Port	Direction	Width	Function
SHIFTOUT1, SHIFTOUT2	Output	1	If SERDES_MODE="MASTER" and two ISERDESE2s are to be cascaded, connect SHIFTOUT1/2 to the slave ISERDESE2 SHIFTIN1/2 inputs.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DATA_RATE	STRING	"DDR", "SDR"	"DDR"	The DATA_RATE attribute defines whether the incoming data stream is processed as single data rate (SDR) or double data rate (DDR).
DATA_WIDTH	DECIMAL	4, 2, 3, 5, 6, 7, 8, 10, 14	4	Defines the width of the serial-to-parallel converter. The legal value depends on the DATA_RATE attribute (SDR or DDR). • If DATA_RATE = DDR, value is limited to 4, 6, 8, 10 or 14. • If DATA_RATE = SDR, value is limited to 2, 3, 4, 5, 6, 7, or 8.
DYN_CLKDIV_INV_EN	STRING	"FALSE", "TRUE"	"FALSE"	Enables DYNCLKDIVINVSEL inversion when “TRUE” and disables HDL inversions on CLKDIV pin.
DYN_CLK_INV_EN	STRING	"FALSE", "TRUE"	"FALSE"	Enables DYNCLKINVSEL inversion when “TRUE” and disables HDL inversions on CLK and CLKB pins.
INIT_Q1, INIT_Q2, INIT_Q3, INIT_Q4	BINARY	1'b0 to 1'b1	1'b0	Specifies the initial value on the Q1 through Q4 outputs after configuration.
INTERFACE_TYPE	STRING	"MEMORY", "MEMORY_DDR3", "MEMORY_QDR", "NETWORKING", "OVERSAMPLE"	"MEMORY"	Specifies the mode of operation for the ISERDESE2. For details on each mode, please refer to the 7 series FPGA SelectIO Resources User Guide.

Attribute	Type	Allowed Values	Default	Description
IOBDELAY	STRING	"NONE", "BOTH", "IBUF", "IFD"	"NONE"	Specifies the input sources for the ISERDESE2 module. The D and DDLY pins are dedicated inputs to the ISERDESE2. The D input is a direct connection to the I/O. The DDLY pin is a direct connection to the IODELAYE2. This allows the user to either have a delayed or non-delayed version of the input to the registered (Q1- Q6) or combinatorial path (O) output. The attribute IOBDELAY determines the input applied the output. • "NONE": O => D | Q1-Q6 => D • "IBUF": O => DDLY | Q1-Q6 => D • "IFD": O => D | Q1-Q6 => DDLY • "BOTH": O => DDLY | Q1-Q6 => DDLY
NUM_CE	DECIMAL	2, 1	2	The NUM_CE attribute defines the number of clock enables (CE1 and CE2) used.
OFB_USED	STRING	"FALSE", "TRUE"	"FALSE"	Enables the path from the OLOGIC, OSERDESE2 OFB pin to the ISERDESE2 OFB pin. Disables the use of the D input pin.
SERDES_MODE	STRING	"MASTER", "SLAVE"	"MASTER"	Specifies whether the ISERDESE2 module is a master or slave when using width expansion. Set to "MASTER" when not using width expansion.
SRVAL_Q1, SRVAL_Q2, SRVAL_Q3, SRVAL_Q4	BINARY	1'b0 to 1'b1	1'b0	Specifies the value (set or reset) of Q1 through Q4 outputs when the SR pin is invoked.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- ISERDESE2: Input SERial/DESerializer with Bitslip
--            7 Series
-- Xilinx HDL Language Template, version 2020.1

ISERDESE2_inst : ISERDESE2
generic map (
   DATA_RATE => "DDR",           -- DDR, SDR
   DATA_WIDTH => 4,              -- Parallel data width (2-8,10,14)
   DYN_CLKDIV_INV_EN => "FALSE", -- Enable DYNCLKDIVINVSEL inversion (FALSE, TRUE)
   DYN_CLK_INV_EN => "FALSE",    -- Enable DYNCLKINVSEL inversion (FALSE, TRUE)
   -- INIT_Q1 - INIT_Q4: Initial value on the Q outputs (0/1)
   INIT_Q1 => '0',
   INIT_Q2 => '0',
   INIT_Q3 => '0',
   INIT_Q4 => '0',
   INTERFACE_TYPE => "MEMORY",   -- MEMORY, MEMORY_DDR3, MEMORY_QDR, NETWORKING, OVERSAMPLE
   IOBDELAY => "NONE",           -- NONE, BOTH, IBUF, IFD
   NUM_CE => 2,                  -- Number of clock enables (1,2)
   OFB_USED => "FALSE",          -- Select OFB path (FALSE, TRUE)
   SERDES_MODE => "MASTER",      -- MASTER, SLAVE
   -- SRVAL_Q1 - SRVAL_Q4: Q output values when SR is used (0/1)
   SRVAL_Q1 => '0',
   SRVAL_Q2 => '0',
```

```
   SRVAL_Q3 => '0',
   SRVAL_Q4 => '0'
)
port map (
   O => O,                          -- 1-bit output: Combinatorial output
   -- Q1 - Q8: 1-bit (each) output: Registered data outputs
   Q1 => Q1,
   Q2 => Q2,
   Q3 => Q3,
   Q4 => Q4,
   Q5 => Q5,
   Q6 => Q6,
   Q7 => Q7,
   Q8 => Q8,
   -- SHIFTOUT1, SHIFTOUT2: 1-bit (each) output: Data width expansion output ports
   SHIFTOUT1 => SHIFTOUT1,
   SHIFTOUT2 => SHIFTOUT2,
   BITSLIP => BITSLIP,          -- 1-bit input: The BITSLIP pin performs a Bitslip operation synchronous to
                                -- CLKDIV when asserted (active High). Subsequently, the data seen on the
                                -- Q1 to Q8 output ports will shift, as in a barrel-shifter operation, one
                                -- position every time Bitslip is invoked (DDR operation is different from
                                     -- SDR).

   -- CE1, CE2: 1-bit (each) input: Data register clock enable inputs
   CE1 => CE1,
   CE2 => CE2,
   CLKDIVP => CLKDIVP,              -- 1-bit input: TBD
   -- Clocks: 1-bit (each) input: ISERDESE2 clock input ports
   CLK => CLK,                      -- 1-bit input: High-speed clock
   CLKB => CLKB,                    -- 1-bit input: High-speed secondary clock
   CLKDIV => CLKDIV,                -- 1-bit input: Divided clock
   OCLK => OCLK,                -- 1-bit input: High speed output clock used when INTERFACE_TYPE="MEMORY"
   -- Dynamic Clock Inversions: 1-bit (each) input: Dynamic clock inversion pins to switch clock polarity
   DYNCLKDIVSEL => DYNCLKDIVSEL, -- 1-bit input: Dynamic CLKDIV inversion
   DYNCLKSEL => DYNCLKSEL,          -- 1-bit input: Dynamic CLK/CLKB inversion
   -- Input Data: 1-bit (each) input: ISERDESE2 data input ports
   D => D,                          -- 1-bit input: Data input
   DDLY => DDLY,                    -- 1-bit input: Serial data from IDELAYE2
   OFB => OFB,                      -- 1-bit input: Data feedback from OSERDESE2
   OCLKB => OCLKB,                  -- 1-bit input: High speed negative edge output clock
   RST => RST,                      -- 1-bit input: Active high asynchronous reset
   -- SHIFTIN1, SHIFTIN2: 1-bit (each) input: Data width expansion input ports
   SHIFTIN1 => SHIFTIN1,
   SHIFTIN2 => SHIFTIN2
);

-- End of ISERDESE2_inst instantiation
```

Verilog Instantiation Template

```
// ISERDESE2: Input SERial/DESerializer with Bitslip
//            7 Series
// Xilinx HDL Language Template, version 2020.1

ISERDESE2 #(
   .DATA_RATE("DDR"),           // DDR, SDR
   .DATA_WIDTH(4),              // Parallel data width (2-8,10,14)
   .DYN_CLKDIV_INV_EN("FALSE"), // Enable DYNCLKDIVINVSEL inversion (FALSE, TRUE)
   .DYN_CLK_INV_EN("FALSE"),    // Enable DYNCLKINVSEL inversion (FALSE, TRUE)
   // INIT_Q1 - INIT_Q4: Initial value on the Q outputs (0/1)
   .INIT_Q1(1'b0),
   .INIT_Q2(1'b0),
   .INIT_Q3(1'b0),
   .INIT_Q4(1'b0),
   .INTERFACE_TYPE("MEMORY"),   // MEMORY, MEMORY_DDR3, MEMORY_QDR, NETWORKING, OVERSAMPLE
   .IOBDELAY("NONE"),           // NONE, BOTH, IBUF, IFD
   .NUM_CE(2),                  // Number of clock enables (1,2)
   .OFB_USED("FALSE"),          // Select OFB path (FALSE, TRUE)
   .SERDES_MODE("MASTER"),      // MASTER, SLAVE
   // SRVAL_Q1 - SRVAL_Q4: Q output values when SR is used (0/1)
   .SRVAL_Q1(1'b0),
   .SRVAL_Q2(1'b0),
   .SRVAL_Q3(1'b0),
   .SRVAL_Q4(1'b0)
)
ISERDESE2_inst (
```

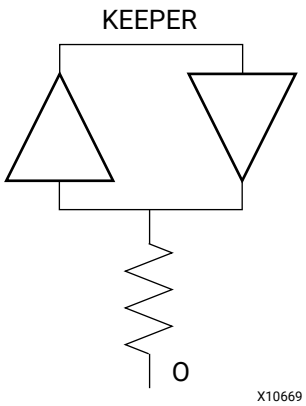
```
   .O(O),                          // 1-bit output: Combinatorial output
   // Q1 - Q8: 1-bit (each) output: Registered data outputs
   .Q1(Q1),
   .Q2(Q2),
   .Q3(Q3),
   .Q4(Q4),
   .Q5(Q5),
   .Q6(Q6),
   .Q7(Q7),
   .Q8(Q8),
   // SHIFTOUT1, SHIFTOUT2: 1-bit (each) output: Data width expansion output ports
   .SHIFTOUT1(SHIFTOUT1),
   .SHIFTOUT2(SHIFTOUT2),
   .BITSLIP(BITSLIP),              // 1-bit input: The BITSLIP pin performs a Bitslip operation synchronous to
                                   // CLKDIV when asserted (active High). Subsequently, the data seen on the Q1
                                   // to Q8 output ports will shift, as in a barrel-shifter operation, one
                                   // position every time Bitslip is invoked (DDR operation is different from
                                       // SDR).

   // CE1, CE2: 1-bit (each) input: Data register clock enable inputs
   .CE1(CE1),
   .CE2(CE2),
   .CLKDIVP(CLKDIVP),              // 1-bit input: TBD
   // Clocks: 1-bit (each) input: ISERDESE2 clock input ports
   .CLK(CLK),                      // 1-bit input: High-speed clock
   .CLKB(CLKB),                    // 1-bit input: High-speed secondary clock
   .CLKDIV(CLKDIV),                // 1-bit input: Divided clock
   .OCLK(OCLK),                    // 1-bit input: High speed output clock used when INTERFACE_TYPE="MEMORY"
   // Dynamic Clock Inversions: 1-bit (each) input: Dynamic clock inversion pins to switch clock polarity
   .DYNCLKDIVSEL(DYNCLKDIVSEL), // 1-bit input: Dynamic CLKDIV inversion
   .DYNCLKSEL(DYNCLKSEL),          // 1-bit input: Dynamic CLK/CLKB inversion
   // Input Data: 1-bit (each) input: ISERDESE2 data input ports
   .D(D),                          // 1-bit input: Data input
   .DDLY(DDLY),                    // 1-bit input: Serial data from IDELAYE2
   .OFB(OFB),                      // 1-bit input: Data feedback from OSERDESE2
   .OCLKB(OCLKB),                  // 1-bit input: High speed negative edge output clock
   .RST(RST),                      // 1-bit input: Active high asynchronous reset
   // SHIFTIN1, SHIFTIN2: 1-bit (each) input: Data width expansion input ports
   .SHIFTIN1(SHIFTIN1),
   .SHIFTIN2(SHIFTIN2)
);

// End of ISERDESE2_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

KEEPER

Primitive: KEEPER Symbol



Introduction

The design element is a weak keeper element that retains the value of the net connected to its bidirectional O pin. For example, if a logic 1 is being driven onto the net, KEEPER drives a weak/resistive 1 onto the net. If the net driver is then 3-stated, KEEPER continues to drive a weak/resistive 1 onto the net.

Port Descriptions

Port	Direction	Width	Function
O	Output	1-Bit	Keeper output

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	No

This element can be connected to a net in the following locations on a top-level schematic file:

- A net connected to an input IO Marker.
- A net connected to both an output IO Marker and 3-statable IO element, such as an OBUFT.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- KEEPER: I/O Buffer Weak Keeper
--         7 Series
-- Xilinx HDL Language Template, version 2020.1

KEEPER_inst : KEEPER
port map (
   O => O     -- Keeper output (connect directly to top-level port)
);

-- End of KEEPER_inst instantiation
```

Verilog Instantiation Template

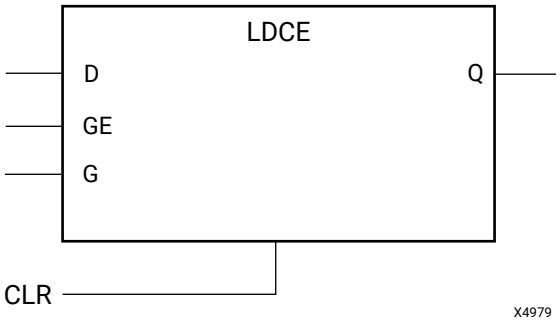
```
// KEEPER: I/O Buffer Weak Keeper
//         7 Series
// Xilinx HDL Language Template, version 2020.1

KEEPER KEEPER_inst (
   .O(O)     // Keeper output (connect directly to top-level port)
);

// End of KEEPER_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

LDCE

Primitive: Transparent Data Latch with Asynchronous Clear and Gate Enable



Introduction

This design element is a transparent data latch with asynchronous clear and gate enable. When the asynchronous clear input (CLR) is High, it overrides the other inputs and resets the data (Q) output Low. Q reflects the data (D) input while the gate (G) input and gate enable (GE) are High and CLR is Low. If (GE) is Low, data on (D) cannot be latched. The data on the (D) input during the High-to-Low gate transition is stored in the latch. The data on the (Q) output remains unchanged as long as (G) or (GE) remains low.

This latch is asynchronously cleared, outputs Low, when power is applied. Power-on conditions are simulated when global set/reset (GSR) is active.

Logic Table

Inputs				Outputs
CLR	**GE**	**G**	**D**	**Q**
1	X	X	X	0
0	0	X	X	No Change
0	1	1	D	D
0	1	0	X	No Change
0	1	↓	D	D

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	BINARY	0, 1	0	Sets the initial value of Q output after configuration.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- LDCE: Transparent latch with Asynchronous Reset and
--        Gate Enable.
--        7 Series
-- Xilinx HDL Language Template, version 2020.1

LDCE_inst : LDCE
generic map (
   INIT => '0') -- Initial value of latch ('0' or '1')
port map (
   Q => Q,       -- Data output
   CLR => CLR,   -- Asynchronous clear/reset input
   D => D,       -- Data input
   G => G,       -- Gate input
   GE => GE      -- Gate enable input
);

-- End of LDCE_inst instantiation
```

Verilog Instantiation Template

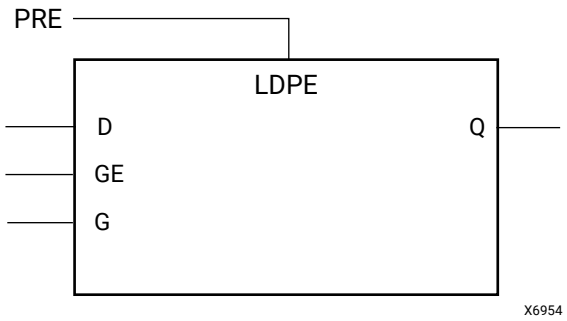
```
// LDCE:  Transparent latch with Asynchronous Reset and Gate Enable.
//        7 Series
// Xilinx HDL Language Template, version 2020.1

LDCE #(
   .INIT(1'b0) // Initial value of latch (1'b0 or 1'b1)
) LDCE_inst (
   .Q(Q),      // Data output
   .CLR(CLR),  // Asynchronous clear/reset input
   .D(D),      // Data input
   .G(G),      // Gate input
   .GE(GE)     // Gate enable input
);

// End of LDCE_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

LDPE

Primitive: Transparent Data Latch with Asynchronous Preset and Gate Enable



Introduction

This design element is a transparent data latch with asynchronous preset and gate enable. When the asynchronous preset (PRE) is High, it overrides the other input and presets the data (Q) output High. Q reflects the data (D) input while the gate (G) input and gate enable (GE) are High. The data on the (D) input during the High-to-Low gate transition is stored in the latch. The data on the (Q) output remains unchanged as long as (G) or (GE) remains Low.

This latch is asynchronously preset, output High, when power is applied. Power-on conditions are simulated when global set/reset (GSR) is active.

Logic Table

Inputs				Outputs
PRE	**GE**	**G**	**D**	**Q**
1	X	X	X	1
0	0	X	X	No Change
0	1	1	D	D
0	1	0	X	No Change
0	1	↓	D	D

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	BINARY	0, 1	1	Specifies the initial value upon power-up or the assertion of GSR for the (Q) port.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- LDPE:  Transparent latch with Asynchronous Set and
--        Gate Enable.
--        7 Series
-- Xilinx HDL Language Template, version 2020.1

LDPE_inst : LDPE
generic map (
   INIT => '0') -- Initial value of latch ('0' or '1')
port map (
   Q => Q,       -- Data output
   CLR => CLR,   -- Asynchronous preset/set input
   D => D,       -- Data input
   G => G,       -- Gate input
   GE => GE      -- Gate enable input
);

-- End of LDPE_inst instantiation
```

Verilog Instantiation Template

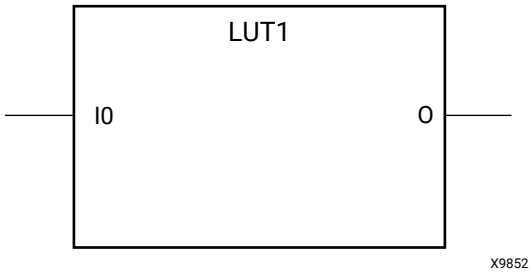
```
// LDPE:  Transparent latch with Asynchronous Preset and Gate Enable.
//        7 Series
// Xilinx HDL Language Template, version 2020.1

LDPE #(
   .INIT(1'b1) // Initial value of latch (1'b0 or 1'b1)
) LDPE_inst (
   .Q(Q),      // Data output
   .PRE(PRE),  // Asynchronous preset/set input
   .D(D),      // Data input
   .G(G),      // Gate input
   .GE(GE)     // Gate enable input
);

// End of LDPE_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

LUT1

Primitive: 1-Bit Look-Up Table with General Output



Introduction

This design element is a 1-bit look-up table (LUT) with general output (O).

An INIT attribute with an appropriate number of hexadecimal digits for the number of inputs must be attached to the LUT to specify its function. This element provides a look-up table version of a buffer or inverter. These elements are the basic building blocks. Multiple variants of LUTs accommodate additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- **The Logic Table Method** Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.
- **The Equation Method** Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs	Outputs
I0	**O**
0	INIT[0]
1	INIT[1]
INIT = Binary number assigned to the INIT attribute	

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 2-Bit Value	All zeros	Initializes look-up tables.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- LUT1: 1-input Look-Up Table with general output
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

LUT1_inst : LUT1
generic map (
   INIT => "00")
port map (
   O => O,    -- LUT general output
   I0 => I0   -- LUT input
);

-- End of LUT1_inst instantiation
```

Verilog Instantiation Template

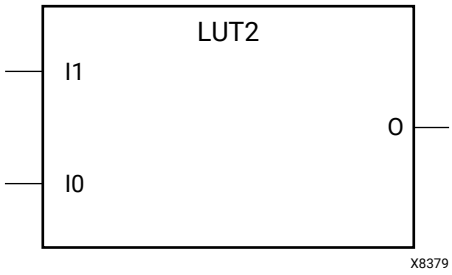
```
// LUT1: 1-input Look-Up Table with general output (Mapped to a LUT6)
//       7 Series
// Xilinx HDL Language Template, version 2020.1

LUT1 #(
   .INIT(2'b00)  // Specify LUT Contents
) LUT1_inst (
   .O(O),    // LUT general output
   .I0(I0)   // LUT input
);

// End of LUT1_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

LUT2

Primitive: 2-Bit Look-Up Table with General Output



Introduction

This design element is a 2-bit look-up table (LUT) with general output (O).

An INIT attribute with an appropriate number of hexadecimal digits for the number of inputs must be attached to the LUT to specify its function. This element provides a look-up table version of a buffer or inverter. These elements are the basic building blocks. Multiple variants of LUTs accommodate additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- **The Logic Table Method** Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.
- **The Equation Method** Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs		Outputs
I1	**I0**	**O**
0	0	INIT[0]
0	1	INIT[1]
1	0	INIT[2]

Inputs		Outputs
I1	**I0**	**O**
1	1	INIT[3]
INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute		

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 4-Bit Value	All zeros	Initializes look-up tables.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- LUT2: 2-input Look-Up Table with general output
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

LUT2_inst : LUT2
generic map (
   INIT => X"0")
port map (
   O => O,   -- LUT general output
   I0 => I0, -- LUT input
   I1 => I1  -- LUT input
);

-- End of LUT2_inst instantiation
```

Verilog Instantiation Template

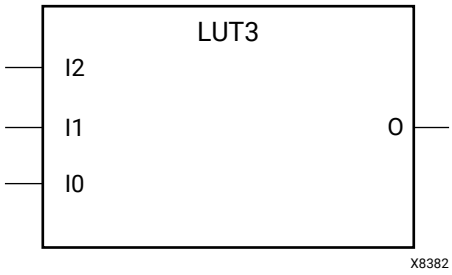
```
// LUT2: 2-input Look-Up Table with general output (Mapped to a LUT6)
//       7 Series
// Xilinx HDL Language Template, version 2020.1

LUT2 #(
   .INIT(4'h0)  // Specify LUT Contents
) LUT2_inst (
   .O(O),   // LUT general output
   .I0(I0), // LUT input
   .I1(I1)  // LUT input
);

// End of LUT2_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

LUT3

Primitive: 3-Bit Look-Up Table with General Output



Introduction

This design element is a 3-bit look-up table (LUT) with general output (O). A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

An INIT attribute with an appropriate number of hexadecimal digits for the number of inputs must be attached to the LUT to specify its function. This element provides a look-up table version of a buffer or inverter. These elements are the basic building blocks. Multiple variants of LUTs accommodate additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- **The Logic Table Method** Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.
- **The Equation Method** Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs			Outputs
I2	**I1**	**I0**	**O**
0	0	0	INIT[0]

Inputs			Outputs
I2	**I1**	**I0**	**O**
0	0	1	INIT[1]
0	1	0	INIT[2]
0	1	1	INIT[3]
1	0	0	INIT[4]
1	0	1	INIT[5]
1	1	0	INIT[6]
1	1	1	INIT[7]
INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute			

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 8-Bit Value	All zeros	Initializes look-up tables.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- LUT3: 3-input Look-Up Table with general output (Mapped to a LUT6)
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

LUT3_inst : LUT3
generic map (
   INIT => X"00")
port map (
   O => O,    -- LUT general output
   I0 => I0, -- LUT input
   I1 => I1, -- LUT input
   I2 => I2  -- LUT input
);

-- End of LUT3_inst instantiation
```

Verilog Instantiation Template

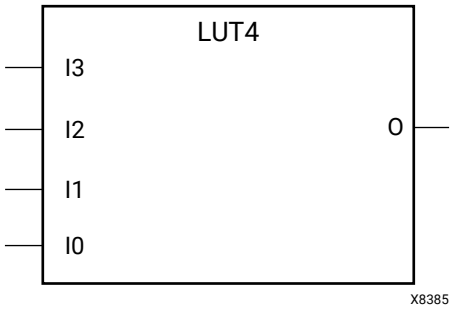
```
// LUT3: 3-input Look-Up Table with general output (Mapped to a LUT6)
//       7 Series
// Xilinx HDL Language Template, version 2020.1

LUT3 #(
   .INIT(8'h00)  // Specify LUT Contents
) LUT3_inst (
   .O(O),   // LUT general output
   .I0(I0), // LUT input
   .I1(I1), // LUT input
   .I2(I2)  // LUT input
);

// End of LUT3_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

LUT4

Primitive: 4-Bit Look-Up-Table with General Output



Introduction

This design element is a 4-bit look-up table (LUT) with general output (O).

An INIT attribute with an appropriate number of hexadecimal digits for the number of inputs must be attached to the LUT to specify its function. This element provides a look-up table version of a buffer or inverter. These elements are the basic building blocks. Multiple variants of LUTs accommodate additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation.

The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- **The Logic Table Method** Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.
- **The Equation Method** Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs				Outputs
I3	**I2**	**I1**	**I0**	**O**
0	0	0	0	INIT[0]
0	0	0	1	INIT[1]

Inputs				Outputs
I3	**I2**	**I1**	**I0**	**O**
0	0	1	0	INIT[2]
0	0	1	1	INIT[3]
0	1	0	0	INIT[4]
0	1	0	1	INIT[5]
0	1	1	0	INIT[6]
0	1	1	1	INIT[7]
1	0	0	0	INIT[8]
1	0	0	1	INIT[9]
1	0	1	0	INIT[10]
1	0	1	1	INIT[11]
1	1	0	0	INIT[12]
1	1	0	1	INIT[13]
1	1	1	0	INIT[14]
1	1	1	1	INIT[15]
INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute				

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 16-Bit Value	All zeros	Initializes look-up tables.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- LUT4: 4-input Look-Up Table with general output
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

LUT4_inst : LUT4
generic map (
   INIT => X"0000")
port map (
   O => O,   -- LUT general output
```

```
   I0 => I0, -- LUT input
   I1 => I1, -- LUT input
   I2 => I2, -- LUT input
   I3 => I3  -- LUT input
);

-- End of LUT4_inst instantiation
```

Verilog Instantiation Template

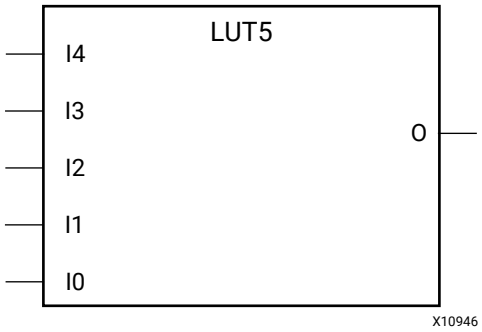
```
// LUT4: 4-input Look-Up Table with general output (Mapped to a LUT6)
//       7 Series
// Xilinx HDL Language Template, version 2020.1

LUT4 #(
   .INIT(16'h0000)  // Specify LUT Contents
) LUT4_inst (
   .O(O),   // LUT general output
   .I0(I0), // LUT input
   .I1(I1), // LUT input
   .I2(I2), // LUT input
   .I3(I3)  // LUT input
);

// End of LUT4_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

LUT5

Primitive: 5-Input look-up Table with General Output



Introduction

This design element is a 5-input, 1-output look-up table (LUT) that can either act as an asynchronous 32-bit ROM (with 5-bit addressing) or implement any 5-input logic function. LUTs are the basic logic building blocks and are used to implement most logic functions of the design. One LUT5 is packed into a LUT6 within a slice, or two LUT5s can be packed into a single LUT6 with some restrictions. The functionality of the LUT5, LUT5_L and LUT5_D is the same. However, the LUT5_L and LUT5_D allow the additional specification to connect the LUT5 output signal to an internal slice or CLB connection using the LO output. The LUT5_L specifies that the only connections from the LUT5 will be within a slice or CLB, while the LUT5_D allows the specification to connect the output of the LUT to both inter-slice/CLB logic and external logic as well. The LUT5 does not state any specific output connections and should be used in all cases except where internal slice or CLB signal connections must be implicitly specified.

An INIT attribute consisting of a 32-bit hexadecimal value must be specified to indicate the LUTs logical function. The INIT value is calculated by assigning a 1 to the corresponding INIT bit value when the associated inputs are applied. For instance, a Verilog INIT value of 32'h80000000 (X"80000000" for VHDL) makes the output zero unless all of the inputs are one (a 5-input AND gate). A Verilog INIT value of 32'hfffffffe (X"FFFFFFFE" for VHDL) makes the output one unless all zeros are on the inputs (a 5-input OR gate).

The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- **The Logic Table Method** Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

- **The Equation Method** Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs					Outputs
I4	**I3**	**I2**	**I1**	**I0**	**LO**
0	0	0	0	0	INIT[0]
0	0	0	0	1	INIT[1]
0	0	0	1	0	INIT[2]
0	0	0	1	1	INIT[3]
0	0	1	0	0	INIT[4]
0	0	1	0	1	INIT[5]
0	0	1	1	0	INIT[6]
0	0	1	1	1	INIT[7]
0	1	0	0	0	INIT[8]
0	1	0	0	1	INIT[9]
0	1	0	1	0	INIT[10]
0	1	0	1	1	INIT[11]
0	1	1	0	0	INIT[12]
0	1	1	0	1	INIT[13]
0	1	1	1	0	INIT[14]
0	1	1	1	1	INIT[15]
1	0	0	0	0	INIT[16]
1	0	0	0	1	INIT[17]
1	0	0	1	0	INIT[18]
1	0	0	1	1	INIT[19]
1	0	1	0	0	INIT[20]
1	0	1	0	1	INIT[21]
1	0	1	1	0	INIT[22]
1	0	1	1	1	INIT[23]
1	1	0	0	0	INIT[24]
1	1	0	0	1	INIT[25]
1	1	0	1	0	INIT[26]
1	1	0	1	1	INIT[27]
1	1	1	0	0	INIT[28]
1	1	1	0	1	INIT[29]
1	1	1	1	0	INIT[30]
1	1	1	1	1	INIT[31]

Inputs					Outputs
I4	**I3**	**I2**	**I1**	**I0**	**LO**
INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute					

Port Descriptions

Port	Direction	Width	Function
O	Output	1	5-LUT output
I0, I1, I2, I3, I4	Input	1	LUT inputs

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 32-Bit Value	All zeros	Initializes look-up tables.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- LUT5: 5-input Look-Up Table with general output (Mapped to SliceM LUT6)
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

LUT5_inst : LUT5
generic map (
   INIT => X"00000000") -- Specify LUT Contents
port map (
   O => O,  -- LUT general output
   I0 => I0,    -- LUT input
   I1 => I1,    -- LUT input
   I2 => I2,    -- LUT input
   I3 => I3,    -- LUT input
   I4 => I4     -- LUT input
);

-- End of LUT5_inst instantiation
```

Verilog Instantiation Template

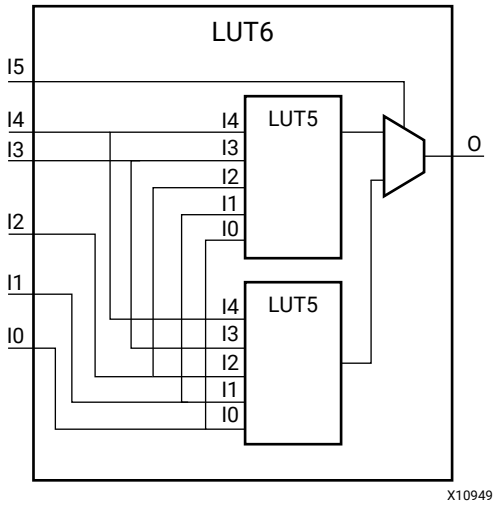
```
// LUT5: 5-input Look-Up Table with general output (Mapped to a LUT6)
//       7 Series
// Xilinx HDL Language Template, version 2020.1

LUT5 #(
   .INIT(32'h00000000)  // Specify LUT Contents
) LUT5_inst (
   .O(O),   // LUT general output
   .I0(I0), // LUT input
   .I1(I1), // LUT input
   .I2(I2), // LUT input
   .I3(I3), // LUT input
   .I4(I4)  // LUT input
);

// End of LUT5_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

LUT6

Primitive: 6-Input Look-Up Table with General Output



Introduction

This design element is a 6-input, 1-output look-up table (LUT) that can either act as an asynchronous 64-bit ROM (with 6-bit addressing) or implement any 6-input logic function. LUTs are the basic logic building blocks and are used to implement most logic functions of the design. A LUT6 is mapped to one of the four look-up tables in the slice. The functionality of the LUT6, LUT6_L and LUT6_D is the same. However, the LUT6_L and LUT6_D allow the additional specification to connect the LUT6 output signal to an internal slice, or CLB connection, using the LO output. The LUT6_L specifies that the only connections from the LUT6 will be within a slice, or CLB, while the LUT6_D allows the specification to connect the output of the LUT to both inter-slice/CLB logic and external logic as well. The LUT6 does not state any specific output connections and should be used in all cases except where internal slice or CLB signal connections must be implicitly specified.

An INIT attribute consisting of a 64-bit Hexadecimal value must be specified to indicate the LUTs logical function. The INIT value is calculated by assigning a 1 to corresponding INIT bit value when the associated inputs are applied. For instance, a Verilog INIT value of 64'h8000000000000000 (X"8000000000000000" for VHDL) makes the output zero unless all of the inputs are one (a 6-input AND gate). A Verilog INIT value of 64'hfffffffffffffffe (X"FFFFFFFFFFFFFFFE" for VHDL) makes the output one unless all zeros are on the inputs (a 6-input OR gate).

The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- **The Logic Table Method** Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.
- **The Equation Method** Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs						Outputs
I5	**I4**	**I3**	**I2**	**I1**	**I0**	**O**
0	0	0	0	0	0	INIT[0]
0	0	0	0	0	1	INIT[1]
0	0	0	0	1	0	INIT[2]
0	0	0	0	1	1	INIT[3]
0	0	0	1	0	0	INIT[4]
0	0	0	1	0	1	INIT[5]
0	0	0	1	1	0	INIT[6]
0	0	0	1	1	1	INIT[7]
0	0	1	0	0	0	INIT[8]
0	0	1	0	0	1	INIT[9]
0	0	1	0	1	0	INIT[10]
0	0	1	0	1	1	INIT[11]
0	0	1	1	0	0	INIT[12]
0	0	1	1	0	1	INIT[13]
0	0	1	1	1	0	INIT[14]
0	0	1	1	1	1	INIT[15]
0	1	0	0	0	0	INIT[16]
0	1	0	0	0	1	INIT[17]
0	1	0	0	1	0	INIT[18]
0	1	0	0	1	1	INIT[19]
0	1	0	1	0	0	INIT[20]
0	1	0	1	0	1	INIT[21]
0	1	0	1	1	0	INIT[22]
0	1	0	1	1	1	INIT[23]
0	1	1	0	0	0	INIT[24]
0	1	1	0	0	1	INIT[25]

Inputs						Outputs
I5	**I4**	**I3**	**I2**	**I1**	**I0**	**O**
0	1	1	0	1	0	INIT[26]
0	1	1	0	1	1	INIT[27]
0	1	1	1	0	0	INIT[28]
0	1	1	1	0	1	INIT[29]
0	1	1	1	1	0	INIT[30]
0	1	1	1	1	1	INIT[31]
1	0	0	0	0	0	INIT[32]
1	0	0	0	0	1	INIT[33]
1	0	0	0	1	0	INIT[34]
1	0	0	0	1	1	INIT[35]
1	0	0	1	0	0	INIT[36]
1	0	0	1	0	1	INIT[37]
1	0	0	1	1	0	INIT[38]
1	0	0	1	1	1	INIT[39]
1	0	1	0	0	0	INIT[40]
1	0	1	0	0	1	INIT[41]
1	0	1	0	1	0	INIT[42]
1	0	1	0	1	1	INIT[43]
1	0	1	1	0	0	INIT[44]
1	0	1	1	0	1	INIT[45]
1	0	1	1	1	0	INIT[46]
1	0	1	1	1	1	INIT[47]
1	1	0	0	0	0	INIT[48]
1	1	0	0	0	1	INIT[49]
1	1	0	0	1	0	INIT[50]
1	1	0	0	1	1	INIT[51]
1	1	0	1	0	0	INIT[52]
1	1	0	1	0	1	INIT[53]
1	1	0	1	1	0	INIT[54]
1	1	0	1	1	1	INIT[55]
1	1	1	0	0	0	INIT[56]
1	1	1	0	0	1	INIT[57]
1	1	1	0	1	0	INIT[58]
1	1	1	0	1	1	INIT[59]
1	1	1	1	0	0	INIT[60]
1	1	1	1	0	1	INIT[61]
1	1	1	1	1	0	INIT[62]
1	1	1	1	1	1	INIT[63]
INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute						

Port Descriptions

Port	Direction	Width	Function
O	Output	1	6/5-LUT output
I0, I1, I2, I3, I4, I5	Input	1	LUT inputs

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 64-Bit Value	All zeros	Initializes look-up tables.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- LUT6: 6-input Look-Up Table with general output
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

LUT6_inst : LUT6
generic map (
   INIT => X"0000000000000000") -- Specify LUT Contents
port map (
   O => O,  -- LUT general output
   I0 => I0,   -- LUT input
   I1 => I1,   -- LUT input
   I2 => I2,   -- LUT input
   I3 => I3,   -- LUT input
   I4 => I4,   -- LUT input
   I5 => I5    -- LUT input
);

-- End of LUT6_inst instantiation
```

Verilog Instantiation Template

```
// LUT6: 6-input Look-Up Table with general output
//       7 Series
// Xilinx HDL Language Template, version 2020.1

LUT6 #(
   .INIT(64'h0000000000000000)  // Specify LUT Contents
) LUT6_inst (
   .O(O),   // LUT general output
   .I0(I0), // LUT input
```

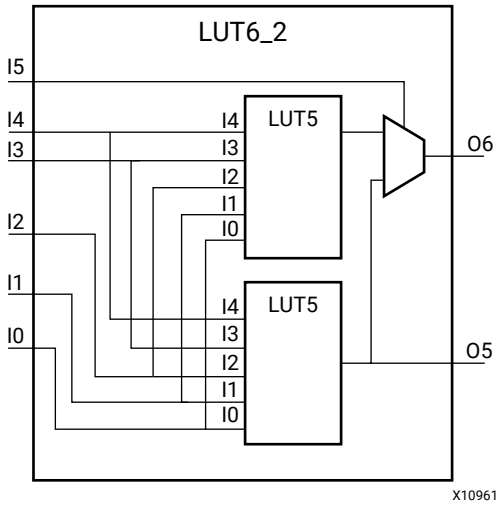
```
   .I1(I1), // LUT input
   .I2(I2), // LUT input
   .I3(I3), // LUT input
   .I4(I4), // LUT input
   .I5(I5)  // LUT input
);

// End of LUT6_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

LUT6_2

Primitive: Six-input, 2-output, Look-Up Table



Introduction

This design element is a 6-input, 2-output look-up table (LUT) that can either act as a dual asynchronous 32-bit ROM (with 5-bit addressing), implement any two 5-input logic functions with shared inputs, or implement a 6-input logic function and a 5-input logic function with shared inputs and shared logic values. LUTs are the basic logic building blocks and are used to implement most logic functions of the design. A LUT6_2 will be mapped to one of the four look-up tables in the slice.

An INIT attribute consisting of a 64-bit hexadecimal value must be specified to indicate the LUTs logical function. The INIT value is calculated by assigning a 1 to corresponding INIT bit value when the associated inputs are applied. For instance, a Verilog INIT value of 64'hfffffffffffffffe (X"FFFFFFFFFFFFFFFE" for VHDL) makes the O6 output 1 unless all zeros are on the inputs and the O5 output a 1, or unless I[4:0] are all zeros (a 5-input and 6-input OR gate). The lower half (bits 31:0) of the INIT values apply to the logic function of the O5 output.

The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- **The Logic Table Method:** Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.

- **The Equation Method:** Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs						Outputs	
I5	**I4**	**I3**	**I2**	**I1**	**I0**	**O5**	**O6**
0	0	0	0	0	0	INIT[0]	INIT[0]
0	0	0	0	0	1	INIT[1]	INIT[1]
0	0	0	0	1	0	INIT[2]	INIT[2]
0	0	0	0	1	1	INIT[3]	INIT[3]
0	0	0	1	0	0	INIT[4]	INIT[4]
0	0	0	1	0	1	INIT[5]	INIT[5]
0	0	0	1	1	0	INIT[6]	INIT[6]
0	0	0	1	1	1	INIT[7]	INIT[7]
0	0	1	0	0	0	INIT[8]	INIT[8]
0	0	1	0	0	1	INIT[9]	INIT[9]
0	0	1	0	1	0	INIT[10]	INIT[10]
0	0	1	0	1	1	INIT[11]	INIT[11]
0	0	1	1	0	0	INIT[12]	INIT[12]
0	0	1	1	0	1	INIT[13]	INIT[13]
0	0	1	1	1	0	INIT[14]	INIT[14]
0	0	1	1	1	1	INIT[15]	INIT[15]
0	1	0	0	0	0	INIT[16]	INIT[16]
0	1	0	0	0	1	INIT[17]	INIT[17]
0	1	0	0	1	0	INIT[18]	INIT[18]
0	1	0	0	1	1	INIT[19]	INIT[19]
0	1	0	1	0	0	INIT[20]	INIT[20]
0	1	0	1	0	1	INIT[21]	INIT[21]
0	1	0	1	1	0	INIT[22]	INIT[22]
0	1	0	1	1	1	INIT[23]	INIT[23]
0	1	1	0	0	0	INIT[24]	INIT[24]
0	1	1	0	0	1	INIT[25]	INIT[25]
0	1	1	0	1	0	INIT[26]	INIT[26]
0	1	1	0	1	1	INIT[27]	INIT[27]
0	1	1	1	0	0	INIT[28]	INIT[28]
0	1	1	1	0	1	INIT[29]	INIT[29]
0	1	1	1	1	0	INIT[30]	INIT[30]
0	1	1	1	1	1	INIT[31]	INIT[31]

Inputs						Outputs	
I5	**I4**	**I3**	**I2**	**I1**	**I0**	**O5**	**O6**
1	0	0	0	0	0	INIT[0]	INIT[32]
1	0	0	0	0	1	INIT[1]	INIT[33]
1	0	0	0	1	0	INIT[2]	INIT[34]
1	0	0	0	1	1	INIT[3]	INIT[35]
1	0	0	1	0	0	INIT[4]	INIT[36]
1	0	0	1	0	1	INIT[5]	INIT[37]
1	0	0	1	1	0	INIT[6]	INIT[38]
1	0	0	1	1	1	INIT[7]	INIT[39]
1	0	1	0	0	0	INIT[8]	INIT[40]
1	0	1	0	0	1	INIT[9]	INIT[41]
1	0	1	0	1	0	INIT[10]	INIT[42]
1	0	1	0	1	1	INIT[11]	INIT[43]
1	0	1	1	0	0	INIT[12]	INIT[44]
1	0	1	1	0	1	INIT[13]	INIT[45]
1	0	1	1	1	0	INIT[14]	INIT[46]
1	0	1	1	1	1	INIT[15]	INIT[47]
1	1	0	0	0	0	INIT[16]	INIT[48]
1	1	0	0	0	1	INIT[17]	INIT[49]
1	1	0	0	1	0	INIT[18]	INIT[50]
1	1	0	0	1	1	INIT[19]	INIT[51]
1	1	0	1	0	0	INIT[20]	INIT[52]
1	1	0	1	0	1	INIT[21]	INIT[53]
1	1	0	1	1	0	INIT[22]	INIT[54]
1	1	0	1	1	1	INIT[23]	INIT[55]
1	1	1	0	0	0	INIT[24]	INIT[56]
1	1	1	0	0	1	INIT[25]	INIT[57]
1	1	1	0	1	0	INIT[26]	INIT[58]
1	1	1	0	1	1	INIT[27]	INIT[59]
1	1	1	1	0	0	INIT[28]	INIT[60]
1	1	1	1	0	1	INIT[29]	INIT[61]
1	1	1	1	1	0	INIT[30]	INIT[62]
1	1	1	1	1	1	INIT[31]	INIT[63]
INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute.							

Port Descriptions

Port	Direction	Width	Function
O6	Output	1	6/5-LUT output.
O5	Output	1	5-LUT output.

Port	Direction	Width	Function
I0, I1, I2, I3, I4, I5	Input	1	LUT inputs.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 64-Bit Value	All zeros	Specifies the LUT5/6 output function.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- LUT6_2: 6-input  2 output Look-Up Table
--         7 Series
-- Xilinx HDL Language Template, version 2020.1

LUT6_2_inst : LUT6_2
generic map (
   INIT => X"0000000000000000") -- Specify LUT Contents
port map (
   O6 => O6,  -- 6/5-LUT output (1-bit)
   O5 => O5,  -- 5-LUT output (1-bit)
   I0 => I0,   -- LUT input (1-bit)
   I1 => I1,   -- LUT input (1-bit)
   I2 => I2,   -- LUT input (1-bit)
   I3 => I3,   -- LUT input (1-bit)
   I4 => I4,   -- LUT input (1-bit)
   I5 => I5    -- LUT input (1-bit)
);

-- End of LUT6_2_inst instantiation
```

Verilog Instantiation Template

```
// LUT6_2: 6-input, 2 output Look-Up Table
//         7 Series
// Xilinx HDL Language Template, version 2020.1

LUT6_2 #(
   .INIT(64'h0000000000000000) // Specify LUT Contents
) LUT6_2_inst (
   .O6(O6), // 1-bit LUT6 output
   .O5(O5), // 1-bit lower LUT5 output
   .I0(I0), // 1-bit LUT input
   .I1(I1), // 1-bit LUT input
   .I2(I2), // 1-bit LUT input
   .I3(I3), // 1-bit LUT input
```

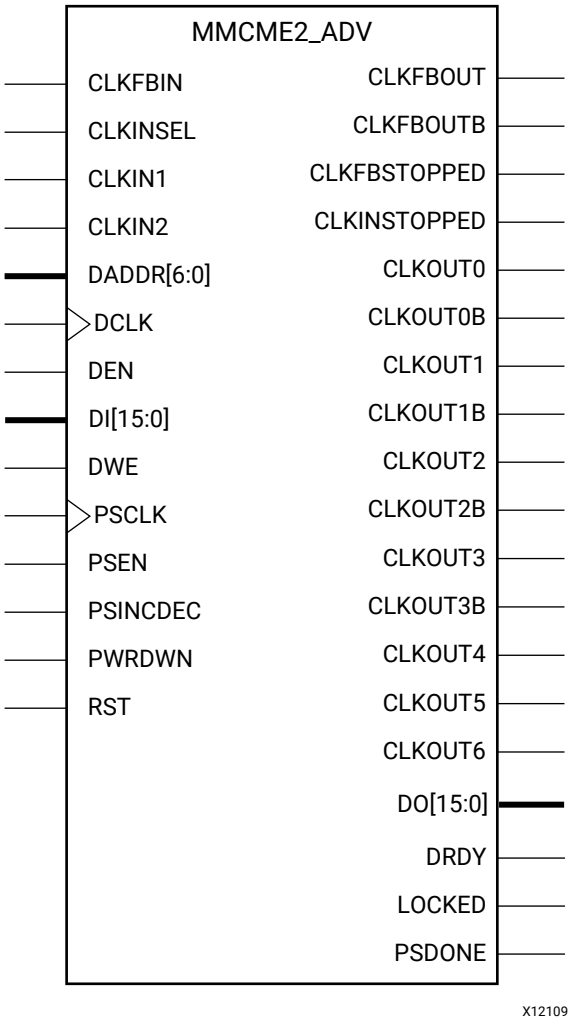
```
   .I4(I4), // 1-bit LUT input
   .I5(I5)  // 1-bit LUT input (fast MUX select only available to O6 output)
);

// End of LUT6_2_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

MMCME2_ADV

Primitive: Advanced Mixed Mode Clock Manager



Introduction

The MMCME2 is a mixed signal block designed to support frequency synthesis, clock network deskew, and jitter reduction. The clock outputs can each have an individual divide, phase shift and duty cycle based on the same VCO frequency. Additionally, the MMCME2 supports dynamic phase shifting and fractional divides.

Port Descriptions

Port	Direction	Width	Function
CLKFBIN	Input	1	Feedback clock pin to the MMCM.

Port	Direction	Width	Function
CLKFBOUT	Output	1	Dedicated MMCM Feedback clock output.
CLKFBOUTB	Output	1	Inverted CLKFBOUT.
CLKFBSTOPPED	Output	1	Status pin indicating that the feedback clock has stopped.
CLKINSEL	Input	1	Controls the state of the input MUX. • High = CLKIN1 • Low = CLKIN2
CLKINSTOPPED	Output	1	Status pin indicating that the input clock has stopped.
CLKIN1	Input	1	Primary clock input.
CLKIN2	Input	1	Secondary clock input to dynamically switch the MMCM reference clock.
CLKOUT0	Output	1	CLKOUT0 output.
CLKOUT0B	Output	1	Inverted CLKOUT0 output.
CLKOUT1	Output	1	CLKOUT1 output.
CLKOUT1B	Output	1	Inverted CLKOUT1 output.
CLKOUT2	Output	1	CLKOUT2 output.
CLKOUT2B	Output	1	Inverted CLKOUT2 output.
CLKOUT3	Output	1	CLKOUT3 output.
CLKOUT3B	Output	1	Inverted CLKOUT3 output.
CLKOUT4	Output	1	CLKOUT4 output.
CLKOUT5	Output	1	CLKOUT5 output.
CLKOUT6	Output	1	CLKOUT6 output.
DADDR<6:0>	Input	7	Dynamic reconfiguration address. Provides a reconfiguration address for the dynamic reconfiguration. When not used, all bits must be assigned zeros.
DCLK	Input	1	The reference clock for the dynamic reconfiguration port.
DEN	Input	1	Dynamic reconfiguration enable. Provides the enable control signal to access the dynamic reconfiguration feature. When the dynamic reconfiguration feature is not used, DEN must be tied Low.
DI<15:0>	Input	16	Dynamic reconfiguration data input. Provides reconfiguration data. When not used, all bits must be set to zero.
DO<15:0>	Output	16	Dynamic reconfiguration output. Provides MMCM data output when using dynamic reconfiguration.
DRDY	Output	1	Dynamic reconfiguration ready output. Provides the response to the DEN signal for the MMCMs dynamic reconfiguration feature.
DWE	Input	1	Dynamic reconfiguration write enable. Provides the write enable control signal to write the DI data into the DADDR address. When not used, it must be tied Low.

Port	Direction	Width	Function
LOCKED	Output	1	An output from the MMCM that indicates when the MMCM has achieved phase alignment within a predefined window and frequency matching within a predefined PPM range. The MMCM automatically locks after power on, no extra reset is required. LOCKED will be deasserted if the input clock stops or the phase alignment is violated (e.g., input clock phase shift). The MMCM must be reset after LOCKED is deasserted.
PSCLK	Input	1	Phase shift clock.
PSDONE	Output	1	Phase shift done.
PSEN	Input	1	Phase shift enable
PSINCDEC	Input	1	Phase shift increment/decrement control.
PWRDWN	Input	1	Powers down instantiated but unused MMCMs.
RST	Input	1	Asynchronous reset signal. The MMCM will synchronously re-enable itself when this signal is released (i.e., MMCM re-enabled). A reset is required when the input clock conditions change (e.g., frequency).

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Recommended
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
BANDWIDTH	STRING	"OPTIMIZED", "HIGH", "LOW"	"OPTIMIZED"	Specifies the MMCM programming algorithm affecting the jitter, phase margin and other characteristics of the MMCM.
CLKFBOUT_MULT_F	3 significant digit FLOAT	2.000 to 64.000	5.000	Specifies the amount to multiply all CLKOUT clock outputs if a different frequency is desired. This number, in combination with the associated CLKOUT#_DIVIDE value and DIVCLK_DIVIDE value, will determine the output frequency.
CLKFBOUT_PHASE	3 significant digit FLOAT	-360.000 to 360.000	0.000	Specifies the phase offset in degrees of the clock feedback output. Shifting the feedback clock results in a negative phase shift of all output clocks to the MMCM.

Attribute	Type	Allowed Values	Default	Description
CLKIN1_PERIOD, CLKIN2_PERIOD	FLOAT (nS)	0.000 to 100.000	0.000	Specifies the input period in ns to the MMCM CLKIN inputs. Resolution is down to the ps. For example, a value of 33.333 would indicate a 30 MHz input clock. This information is mandatory and must be supplied. CLKIN1_PERIOD relates to the input period on the CLKIN1 input while CLKIN2_PERIOD relates to the input clock period on the CLKIN2 input.
CLKOUT1_DIVIDE, CLKOUT2_DIVIDE, CLKOUT3_DIVIDE, CLKOUT4_DIVIDE, CLKOUT5_DIVIDE, CLKOUT6_DIVIDE	DECIMAL	1 to 128	1	Specifies the amount to divide the associated CLKOUT clock output if a different frequency is desired. This number in combination with the CLKFBOUT_MULT_F and DIVCLK_DIVIDE values will determine the output frequency.
CLKOUT0_DIVIDE _F	3 significant digit FLOAT	1.000 to 128.000	1.000	Specifies the amount to divide the associated CLKOUT clock output if a different frequency is desired. This number in combination with the CLKFBOUT_MULT_F and DIVCLK_DIVIDE values will determine the output frequency.
CLKOUT0_DUTY _CYCLE to CLKOUT6_DUTY _CYCLE	3 significant digit FLOAT	0.001 to 0.999	0.500	Specifies the Duty Cycle of the associated CLKOUT clock output in percentage (i.e., 0.50 will generate a 50% duty cycle).
CLKOUT0_PHASE to CLKOUT6_PHASE	3 significant digit FLOAT	-360.000 to 360.000	0.000	Specifies the phase offset in degrees of the clock feedback output. Shifting the feedback clock results in a negative phase shift of all output clocks to the MMCM.
CLKOUT4 _CASCADE	BOOLEAN	FALSE, TRUE	FALSE	Cascades the output divider (counter) into the input of the CLKOUT4 divider for an output clock divider that is greater than 128.
COMPENSATION	STRING	"ZHOLD", "BUF_IN", "EXTERNAL", "INTERNAL"	"ZHOLD"	Clock input compensation. Should be set to ZHOLD. Defines how the MMCM feedback is configured. • "ZHOLD": MMCM is configured to provide a negative hold time at the I/O registers. • "INTERNAL": MMCM is using its own internal feedback path so no delay is being compensated. • "EXTERNAL": a network external to the FPGA is being compensated. • "BUF_IN": configuration does not match with the other compensation modes and no delay will be compensated. This is the case if a clock input is driven by a BUFG/ BUFH/BUFR/GT.

Attribute	Type	Allowed Values	Default	Description
DIVCLK_DIVIDE	DECIMAL	1 to 106	1	Specifies the division ratio for all output clocks with respect to the input clock. Effectively divides the CLKIN going into the PFD.
REF_JITTER1, REF_JITTER2	3 significant digit FLOAT	0.000 to 0.999	0.010	Allows specification of the expected jitter on the CLKIN inputs to better optimize MMCM performance. A bandwidth setting of OPTIMIZED will attempt to choose the best parameter for input clocking when unknown. If known, then the value provided should be specified in terms of the UI percentage (the maximum peak to peak value) of the expected jitter on the input clock. REF_JITTER1 relates to the input jitter on CLKIN1 while REF_JITTER2 relates to the input jitter on CLKIN2.
SS_EN	STRING	"FALSE", "TRUE"	"FALSE"	Enables the spread spectrum feature for the MMCM. Used with SS_MODE and SS_MOD_PERIOD attributes.
SS_MOD_PERIOD	DECIMAL (nS)	4000 to 40000	10000	Specifies the spread spectrum modulation period (ns).
SS_MODE	STRING	"CENTER_HIGH", "CENTER_LOW", "DOWN_HIGH", "DOWN_LOW"	"CENTER _HIGH"	Controls the spread spectrum frequency deviation and the spread type.
STARTUP_WAIT	BOOLEAN	FALSE, TRUE	FALSE	Delays configuration DONE signal from asserting until MMCM is locked.
CLKFBOUT_USE _FINE_PS to CLKOUT6_USE _FINE_PS	BOOLEAN	FALSE, TRUE	FALSE	Counter variable fine phase shift enable.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- MMCME2_ADV: Advanced Mixed Mode Clock Manager
--             7 Series
-- Xilinx HDL Language Template, version 2020.1

MMCME2_ADV_inst : MMCME2_ADV
generic map (
   BANDWIDTH => "OPTIMIZED",      -- Jitter programming (OPTIMIZED, HIGH, LOW)
   CLKFBOUT_MULT_F => 5.0,        -- Multiply value for all CLKOUT (2.000-64.000).
   CLKFBOUT_PHASE => 0.0,         -- Phase offset in degrees of CLKFB (-360.000-360.000).
   -- CLKIN_PERIOD: Input clock period in ns to ps resolution (i.e. 33.333 is 30 MHz).
   CLKIN1_PERIOD => 0.0,
   CLKIN2_PERIOD => 0.0,
   -- CLKOUT0_DIVIDE - CLKOUT6_DIVIDE: Divide amount for CLKOUT (1-128)
   CLKOUT1_DIVIDE => 1,
   CLKOUT2_DIVIDE => 1,
   CLKOUT3_DIVIDE => 1,
   CLKOUT4_DIVIDE => 1,
   CLKOUT5_DIVIDE => 1,
   CLKOUT6_DIVIDE => 1,
   CLKOUT0_DIVIDE_F => 1.0,       -- Divide amount for CLKOUT0 (1.000-128.000).
```

```
   -- CLKOUT0_DUTY_CYCLE - CLKOUT6_DUTY_CYCLE: Duty cycle for CLKOUT outputs (0.01-0.99).
   CLKOUT0_DUTY_CYCLE => 0.5,
   CLKOUT1_DUTY_CYCLE => 0.5,
   CLKOUT2_DUTY_CYCLE => 0.5,
   CLKOUT3_DUTY_CYCLE => 0.5,
   CLKOUT4_DUTY_CYCLE => 0.5,
   CLKOUT5_DUTY_CYCLE => 0.5,
   CLKOUT6_DUTY_CYCLE => 0.5,
   -- CLKOUT0_PHASE - CLKOUT6_PHASE: Phase offset for CLKOUT outputs (-360.000-360.000).
   CLKOUT0_PHASE => 0.0,
   CLKOUT1_PHASE => 0.0,
   CLKOUT2_PHASE => 0.0,
   CLKOUT3_PHASE => 0.0,
   CLKOUT4_PHASE => 0.0,
   CLKOUT5_PHASE => 0.0,
   CLKOUT6_PHASE => 0.0,
   CLKOUT4_CASCADE => FALSE,       -- Cascade CLKOUT4 counter with CLKOUT6 (FALSE, TRUE)
   COMPENSATION => "ZHOLD",        -- ZHOLD, BUF_IN, EXTERNAL, INTERNAL
   DIVCLK_DIVIDE => 1,             -- Master division value (1-106)
   -- REF_JITTER: Reference input jitter in UI (0.000-0.999).
   REF_JITTER1 => 0.0,
   REF_JITTER2 => 0.0,
   STARTUP_WAIT => FALSE,          -- Delays DONE until MMCM is locked (FALSE, TRUE)
   -- Spread Spectrum: Spread Spectrum Attributes
   SS_EN => "FALSE",               -- Enables spread spectrum (FALSE, TRUE)
   SS_MODE => "CENTER_HIGH",       -- CENTER_HIGH, CENTER_LOW, DOWN_HIGH, DOWN_LOW
   SS_MOD_PERIOD => 10000,         -- Spread spectrum modulation period (ns) (VALUES)
   -- USE_FINE_PS: Fine phase shift enable (TRUE/FALSE)
   CLKFBOUT_USE_FINE_PS => FALSE,
   CLKOUT0_USE_FINE_PS => FALSE,
   CLKOUT1_USE_FINE_PS => FALSE,
   CLKOUT2_USE_FINE_PS => FALSE,
   CLKOUT3_USE_FINE_PS => FALSE,
   CLKOUT4_USE_FINE_PS => FALSE,
   CLKOUT5_USE_FINE_PS => FALSE,
   CLKOUT6_USE_FINE_PS => FALSE
)
port map (
   -- Clock Outputs: 1-bit (each) output: User configurable clock outputs
   CLKOUT0 => CLKOUT0,             -- 1-bit output: CLKOUT0
   CLKOUT0B => CLKOUT0B,           -- 1-bit output: Inverted CLKOUT0
   CLKOUT1 => CLKOUT1,             -- 1-bit output: CLKOUT1
   CLKOUT1B => CLKOUT1B,           -- 1-bit output: Inverted CLKOUT1
   CLKOUT2 => CLKOUT2,             -- 1-bit output: CLKOUT2
   CLKOUT2B => CLKOUT2B,           -- 1-bit output: Inverted CLKOUT2
   CLKOUT3 => CLKOUT3,             -- 1-bit output: CLKOUT3
   CLKOUT3B => CLKOUT3B,           -- 1-bit output: Inverted CLKOUT3
   CLKOUT4 => CLKOUT4,             -- 1-bit output: CLKOUT4
   CLKOUT5 => CLKOUT5,             -- 1-bit output: CLKOUT5
   CLKOUT6 => CLKOUT6,             -- 1-bit output: CLKOUT6
   -- DRP Ports: 16-bit (each) output: Dynamic reconfiguration ports
   DO => DO,                       -- 16-bit output: DRP data
   DRDY => DRDY,                   -- 1-bit output: DRP ready
   -- Dynamic Phase Shift Ports: 1-bit (each) output: Ports used for dynamic phase shifting of the outputs
   PSDONE => PSDONE,               -- 1-bit output: Phase shift done
   -- Feedback Clocks: 1-bit (each) output: Clock feedback ports
   CLKFBOUT => CLKFBOUT,           -- 1-bit output: Feedback clock
   CLKFBOUTB => CLKFBOUTB,         -- 1-bit output: Inverted CLKFBOUT
   -- Status Ports: 1-bit (each) output: MMCM status ports
   CLKFBSTOPPED => CLKFBSTOPPED, -- 1-bit output: Feedback clock stopped
   CLKINSTOPPED => CLKINSTOPPED, -- 1-bit output: Input clock stopped
   LOCKED => LOCKED,               -- 1-bit output: LOCK
   -- Clock Inputs: 1-bit (each) input: Clock inputs
   CLKIN1 => CLKIN1,               -- 1-bit input: Primary clock
   CLKIN2 => CLKIN2,               -- 1-bit input: Secondary clock
   -- Control Ports: 1-bit (each) input: MMCM control ports
   CLKINSEL => CLKINSEL,           -- 1-bit input: Clock select, High=CLKIN1 Low=CLKIN2
   PWRDWN => PWRDWN,               -- 1-bit input: Power-down
   RST => RST,                     -- 1-bit input: Reset
   -- DRP Ports: 7-bit (each) input: Dynamic reconfiguration ports
   DADDR => DADDR,                 -- 7-bit input: DRP address
   DCLK => DCLK,                   -- 1-bit input: DRP clock
   DEN => DEN,                     -- 1-bit input: DRP enable
   DI => DI,                       -- 16-bit input: DRP data
   DWE => DWE,                     -- 1-bit input: DRP write enable
   -- Dynamic Phase Shift Ports: 1-bit (each) input: Ports used for dynamic phase shifting of the outputs
   PSCLK => PSCLK,                 -- 1-bit input: Phase shift clock
   PSEN => PSEN,                   -- 1-bit input: Phase shift enable
   PSINCDEC => PSINCDEC,           -- 1-bit input: Phase shift increment/decrement
```

```
   -- Feedback Clocks: 1-bit (each) input: Clock feedback ports
   CLKFBIN => CLKFBIN            -- 1-bit input: Feedback clock
);

-- End of MMCME2_ADV_inst instantiation
```

Verilog Instantiation Template

```
// MMCME2_ADV: Advanced Mixed Mode Clock Manager
//             7 Series
// Xilinx HDL Language Template, version 2020.1

MMCME2_ADV #(
   .BANDWIDTH("OPTIMIZED"),        // Jitter programming (OPTIMIZED, HIGH, LOW)
   .CLKFBOUT_MULT_F(5.0),          // Multiply value for all CLKOUT (2.000-64.000).
   .CLKFBOUT_PHASE(0.0),           // Phase offset in degrees of CLKFB (-360.000-360.000).
   // CLKIN_PERIOD: Input clock period in ns to ps resolution (i.e. 33.333 is 30 MHz).
   .CLKIN1_PERIOD(0.0),
   .CLKIN2_PERIOD(0.0),
   // CLKOUT0_DIVIDE - CLKOUT6_DIVIDE: Divide amount for CLKOUT (1-128)
   .CLKOUT1_DIVIDE(1),
   .CLKOUT2_DIVIDE(1),
   .CLKOUT3_DIVIDE(1),
   .CLKOUT4_DIVIDE(1),
   .CLKOUT5_DIVIDE(1),
   .CLKOUT6_DIVIDE(1),
   .CLKOUT0_DIVIDE_F(1.0),         // Divide amount for CLKOUT0 (1.000-128.000).
   // CLKOUT0_DUTY_CYCLE - CLKOUT6_DUTY_CYCLE: Duty cycle for CLKOUT outputs (0.01-0.99).
   .CLKOUT0_DUTY_CYCLE(0.5),
   .CLKOUT1_DUTY_CYCLE(0.5),
   .CLKOUT2_DUTY_CYCLE(0.5),
   .CLKOUT3_DUTY_CYCLE(0.5),
   .CLKOUT4_DUTY_CYCLE(0.5),
   .CLKOUT5_DUTY_CYCLE(0.5),
   .CLKOUT6_DUTY_CYCLE(0.5),
   // CLKOUT0_PHASE - CLKOUT6_PHASE: Phase offset for CLKOUT outputs (-360.000-360.000).
   .CLKOUT0_PHASE(0.0),
   .CLKOUT1_PHASE(0.0),
   .CLKOUT2_PHASE(0.0),
   .CLKOUT3_PHASE(0.0),
   .CLKOUT4_PHASE(0.0),
   .CLKOUT5_PHASE(0.0),
   .CLKOUT6_PHASE(0.0),
   .CLKOUT4_CASCADE("FALSE"),      // Cascade CLKOUT4 counter with CLKOUT6 (FALSE, TRUE)
   .COMPENSATION("ZHOLD"),         // ZHOLD, BUF_IN, EXTERNAL, INTERNAL
   .DIVCLK_DIVIDE(1),              // Master division value (1-106)
   // REF_JITTER: Reference input jitter in UI (0.000-0.999).
   .REF_JITTER1(0.0),
   .REF_JITTER2(0.0),
   .STARTUP_WAIT("FALSE"),         // Delays DONE until MMCM is locked (FALSE, TRUE)
   // Spread Spectrum: Spread Spectrum Attributes
   .SS_EN("FALSE"),                // Enables spread spectrum (FALSE, TRUE)
   .SS_MODE("CENTER_HIGH"),        // CENTER_HIGH, CENTER_LOW, DOWN_HIGH, DOWN_LOW
   .SS_MOD_PERIOD(10000),          // Spread spectrum modulation period (ns) (VALUES)
   // USE_FINE_PS: Fine phase shift enable (TRUE/FALSE)
   .CLKFBOUT_USE_FINE_PS("FALSE"),
   .CLKOUT0_USE_FINE_PS("FALSE"),
   .CLKOUT1_USE_FINE_PS("FALSE"),
   .CLKOUT2_USE_FINE_PS("FALSE"),
   .CLKOUT3_USE_FINE_PS("FALSE"),
   .CLKOUT4_USE_FINE_PS("FALSE"),
   .CLKOUT5_USE_FINE_PS("FALSE"),
   .CLKOUT6_USE_FINE_PS("FALSE")
)
MMCME2_ADV_inst (
   // Clock Outputs: 1-bit (each) output: User configurable clock outputs
   .CLKOUT0(CLKOUT0),            // 1-bit output: CLKOUT0
   .CLKOUT0B(CLKOUT0B),          // 1-bit output: Inverted CLKOUT0
   .CLKOUT1(CLKOUT1),            // 1-bit output: CLKOUT1
   .CLKOUT1B(CLKOUT1B),          // 1-bit output: Inverted CLKOUT1
   .CLKOUT2(CLKOUT2),            // 1-bit output: CLKOUT2
   .CLKOUT2B(CLKOUT2B),          // 1-bit output: Inverted CLKOUT2
   .CLKOUT3(CLKOUT3),            // 1-bit output: CLKOUT3
   .CLKOUT3B(CLKOUT3B),          // 1-bit output: Inverted CLKOUT3
   .CLKOUT4(CLKOUT4),            // 1-bit output: CLKOUT4
   .CLKOUT5(CLKOUT5),            // 1-bit output: CLKOUT5
```

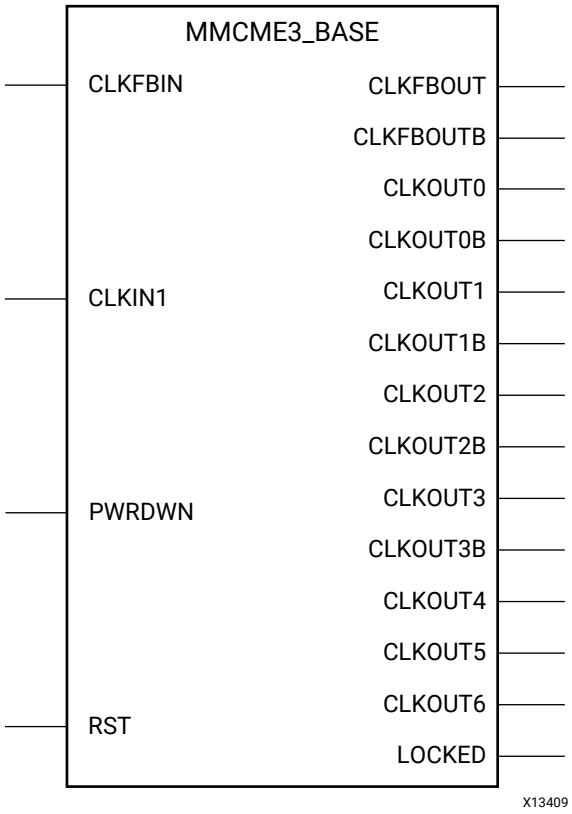
```
   .CLKOUT6(CLKOUT6),           // 1-bit output: CLKOUT6
   // DRP Ports: 16-bit (each) output: Dynamic reconfiguration ports
   .DO(DO),                     // 16-bit output: DRP data
   .DRDY(DRDY),                 // 1-bit output: DRP ready
  // Dynamic Phase Shift Ports: 1-bit (each) output: Ports used for dynamic phase shifting of the outputs
   .PSDONE(PSDONE),             // 1-bit output: Phase shift done
   // Feedback Clocks: 1-bit (each) output: Clock feedback ports
   .CLKFBOUT(CLKFBOUT),         // 1-bit output: Feedback clock
   .CLKFBOUTB(CLKFBOUTB),       // 1-bit output: Inverted CLKFBOUT
   // Status Ports: 1-bit (each) output: MMCM status ports
   .CLKFBSTOPPED(CLKFBSTOPPED), // 1-bit output: Feedback clock stopped
   .CLKINSTOPPED(CLKINSTOPPED), // 1-bit output: Input clock stopped
   .LOCKED(LOCKED),             // 1-bit output: LOCK
   // Clock Inputs: 1-bit (each) input: Clock inputs
   .CLKIN1(CLKIN1),             // 1-bit input: Primary clock
   .CLKIN2(CLKIN2),             // 1-bit input: Secondary clock
   // Control Ports: 1-bit (each) input: MMCM control ports
   .CLKINSEL(CLKINSEL),         // 1-bit input: Clock select, High=CLKIN1 Low=CLKIN2
   .PWRDWN(PWRDWN),             // 1-bit input: Power-down
   .RST(RST),                   // 1-bit input: Reset
   // DRP Ports: 7-bit (each) input: Dynamic reconfiguration ports
   .DADDR(DADDR),               // 7-bit input: DRP address
   .DCLK(DCLK),                 // 1-bit input: DRP clock
   .DEN(DEN),                   // 1-bit input: DRP enable
   .DI(DI),                     // 16-bit input: DRP data
   .DWE(DWE),                   // 1-bit input: DRP write enable
  // Dynamic Phase Shift Ports: 1-bit (each) input: Ports used for dynamic phase shifting of the outputs
   .PSCLK(PSCLK),               // 1-bit input: Phase shift clock
   .PSEN(PSEN),                 // 1-bit input: Phase shift enable
   .PSINCDEC(PSINCDEC),         // 1-bit input: Phase shift increment/decrement
   // Feedback Clocks: 1-bit (each) input: Clock feedback ports
   .CLKFBIN(CLKFBIN)            // 1-bit input: Feedback clock
);

// End of MMCME2_ADV_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

MMCME2_BASE

Primitive: Base Mixed Mode Clock Manager



Introduction

The MMCME2 is a mixed signal block designed to support frequency synthesis, clock network deskew, and jitter reduction. The clock outputs can each have an individual divide, phase shift and duty cycle based on the same VCO frequency. Additionally, the MMCME2 supports dynamic phase shifting and fractional divides.

Port Descriptions

Port	Direction	Width	Function
CLKFBIN	Input	1	Feedback clock pin to the MMCM.
CLKFBOUT	Output	1	Dedicated MMCM Feedback clock output.
CLKFBOUTB	Output	1	Inverted CLKFBOUT output.
CLKOUT0	Output	1	CLKOUT0 output.
CLKOUT0B	Output	1	Inverted CLKOUT0 output.
CLKOUT1	Output	1	CLKOUT1 output.

Port	Direction	Width	Function
CLKOUT1B	Output	1	Inverted CLKOUT1 output.
CLKOUT2	Output	1	CLKOUT2 output.
CLKOUT2B	Output	1	Inverted CLKOUT2 output.
CLKOUT3	Output	1	CLKOUT3 output.
CLKOUT3B	Output	1	Inverted CLKOUT3 output.
CLKOUT4	Output	1	CLKOUT4 output.
CLKOUT5	Output	1	CLKOUT5 output.
CLKOUT6	Output	1	CLKOUT6 output.
CLKIN1	Input	1	General clock input.
PWRDWN	Input	1	Powers down instantiated but unused MMCMs.
RST	Input	1	Asynchronous reset signal. The MMCM will synchronously re-enable itself when this signal is released (i.e., MMCM re-enabled). A reset is required when the input clock conditions change (e.g., frequency).
LOCKED	Output	1	An output from the MMCM that indicates when the MMCM has achieved phase alignment within a predefined window and frequency matching within a predefined PPM range. The MMCM automatically locks after power on, no extra reset is required. LOCKED will be deasserted if the input clock stops or the phase alignment is violated (that is, input clock phase shift). The MMCM must be reset after LOCKED is deasserted.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Recommended
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
BANDWIDTH	STRING	"OPTIMIZED", "HIGH", "LOW"	"OPTIMIZED"	Specifies the MMCM programming algorithm affecting the jitter, phase margin and other characteristics of the MMCM.
CLKFBOUT_MULT_F	3 significant digit FLOAT	2.000 to 64.000	5.000	Specifies the amount to multiply all CLKOUT clock outputs if a different frequency is desired. This number, in combination with the associated CLKOUT#_DIVIDE value and DIVCLK_DIVIDE value, will determine the output frequency.
CLKFBOUT_PHASE	3 significant digit FLOAT	-360.000 to 360.000	0.000	Specifies the phase offset in degrees of the clock feedback output. Shifting the feedback clock results in a negative phase shift of all output clocks to the MMCM.

Attribute	Type	Allowed Values	Default	Description
CLKIN1_PERIOD	FLOAT(nS)	0.000 to 100.000	0.000	Specifies the input period in ns to the MMCM CLKIN1 input. Resolution is down to the ps (3 decimal places). For example, a value of 33.333 would indicate a 30 MHz input clock. This information is mandatory and must be supplied.
CLKOUT1_DIVIDE, CLKOUT2_DIVIDE, CLKOUT3_DIVIDE, CLKOUT4_DIVIDE, CLKOUT5_DIVIDE, CLKOUT6_DIVIDE	DECIMAL	1 to 128	1	Specifies the amount to divide the associated CLKOUT clock output if a different frequency is desired. This number in combination with the CLKFBOUT_MULT_F and DIVCLK_DIVIDE values will determine the output frequency.
CLKOUT0_DIVIDE_F	3 significant digit FLOAT	1.000 to 128.000	1.000	Specifies the amount to divide the associated CLKOUT clock output if a different frequency is desired. This number in combination with the CLKFBOUT_MULT_F and DIVCLK_DIVIDE values will determine the output frequency.
CLKOUT0_DUTY _CYCLE to CLKOUT6_DUTY _CYCLE	3 significant digit FLOAT	0.001 to 0.999	0.500	Specifies the Duty Cycle of the associated CLKOUT clock output in percentage (i.e., 0.50 will generate a 50% duty cycle).
CLKOUT0_PHASE to CLKOUT6_PHASE	3 significant digit FLOAT	-360.000 to 360.000	0.000	Specifies the phase offset in degrees of the clock feedback output. Shifting the feedback clock results in a negative phase shift of all output clocks to the MMCM.
CLKOUT4_CASCADE	BOOLEAN	FALSE, TRUE	FALSE	Cascades the output divider (counter) CLKOUT6 into the input of the CLKOUT4 divider for an output clock divider that is greater than 128.
DIVCLK_DIVIDE	DECIMAL	1 to 106	1	Specifies the division ratio for all output clocks with respect to the input clock. Effectively divides the CLKIN going into the PFD.
REF_JITTER1	3 significant digit FLOAT	0.000 to 0.999	0.010	Allows specification of the expected jitter on CLKIN1 in order to better optimize MMCM performance. A bandwidth setting of OPTIMIZED will attempt to choose the best parameter for input clocking when unknown. If known, then the value provided should be specified in terms of the UI percentage (the maximum peak to peak value) of the expected jitter on the input clock.
STARTUP_WAIT	BOOLEAN	FALSE, TRUE	FALSE	Delays configuration DONE signal from asserting until MMCM is locked.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- MMCME2_BASE: Base Mixed Mode Clock Manager
--              7 Series
-- Xilinx HDL Language Template, version 2020.1

MMCME2_BASE_inst : MMCME2_BASE
generic map (
   BANDWIDTH => "OPTIMIZED",  -- Jitter programming (OPTIMIZED, HIGH, LOW)
   CLKFBOUT_MULT_F => 5.0,    -- Multiply value for all CLKOUT (2.000-64.000).
   CLKFBOUT_PHASE => 0.0,     -- Phase offset in degrees of CLKFB (-360.000-360.000).
   CLKIN1_PERIOD => 0.0,      -- Input clock period in ns to ps resolution (i.e. 33.333 is 30 MHz).
   -- CLKOUT0_DIVIDE - CLKOUT6_DIVIDE: Divide amount for each CLKOUT (1-128)
   CLKOUT1_DIVIDE => 1,
   CLKOUT2_DIVIDE => 1,
   CLKOUT3_DIVIDE => 1,
   CLKOUT4_DIVIDE => 1,
   CLKOUT5_DIVIDE => 1,
   CLKOUT6_DIVIDE => 1,
   CLKOUT0_DIVIDE_F => 1.0,   -- Divide amount for CLKOUT0 (1.000-128.000).
   -- CLKOUT0_DUTY_CYCLE - CLKOUT6_DUTY_CYCLE: Duty cycle for each CLKOUT (0.01-0.99).
   CLKOUT0_DUTY_CYCLE => 0.5,
   CLKOUT1_DUTY_CYCLE => 0.5,
   CLKOUT2_DUTY_CYCLE => 0.5,
   CLKOUT3_DUTY_CYCLE => 0.5,
   CLKOUT4_DUTY_CYCLE => 0.5,
   CLKOUT5_DUTY_CYCLE => 0.5,
   CLKOUT6_DUTY_CYCLE => 0.5,
   -- CLKOUT0_PHASE - CLKOUT6_PHASE: Phase offset for each CLKOUT (-360.000-360.000).
   CLKOUT0_PHASE => 0.0,
   CLKOUT1_PHASE => 0.0,
   CLKOUT2_PHASE => 0.0,
   CLKOUT3_PHASE => 0.0,
   CLKOUT4_PHASE => 0.0,
   CLKOUT5_PHASE => 0.0,
   CLKOUT6_PHASE => 0.0,
   CLKOUT4_CASCADE => FALSE,  -- Cascade CLKOUT4 counter with CLKOUT6 (FALSE, TRUE)
   DIVCLK_DIVIDE => 1,        -- Master division value (1-106)
   REF_JITTER1 => 0.0,        -- Reference input jitter in UI (0.000-0.999).
   STARTUP_WAIT => FALSE      -- Delays DONE until MMCM is locked (FALSE, TRUE)
)
port map (
   -- Clock Outputs: 1-bit (each) output: User configurable clock outputs
   CLKOUT0 => CLKOUT0,      -- 1-bit output: CLKOUT0
   CLKOUT0B => CLKOUT0B,    -- 1-bit output: Inverted CLKOUT0
   CLKOUT1 => CLKOUT1,      -- 1-bit output: CLKOUT1
   CLKOUT1B => CLKOUT1B,    -- 1-bit output: Inverted CLKOUT1
   CLKOUT2 => CLKOUT2,      -- 1-bit output: CLKOUT2
   CLKOUT2B => CLKOUT2B,    -- 1-bit output: Inverted CLKOUT2
   CLKOUT3 => CLKOUT3,      -- 1-bit output: CLKOUT3
   CLKOUT3B => CLKOUT3B,    -- 1-bit output: Inverted CLKOUT3
   CLKOUT4 => CLKOUT4,      -- 1-bit output: CLKOUT4
   CLKOUT5 => CLKOUT5,      -- 1-bit output: CLKOUT5
   CLKOUT6 => CLKOUT6,      -- 1-bit output: CLKOUT6
   -- Feedback Clocks: 1-bit (each) output: Clock feedback ports
   CLKFBOUT => CLKFBOUT,    -- 1-bit output: Feedback clock
   CLKFBOUTB => CLKFBOUTB, -- 1-bit output: Inverted CLKFBOUT
   -- Status Ports: 1-bit (each) output: MMCM status ports
   LOCKED => LOCKED,        -- 1-bit output: LOCK
   -- Clock Inputs: 1-bit (each) input: Clock input
   CLKIN1 => CLKIN1,        -- 1-bit input: Clock
   -- Control Ports: 1-bit (each) input: MMCM control ports
   PWRDWN => PWRDWN,        -- 1-bit input: Power-down
   RST => RST,              -- 1-bit input: Reset
```

```
   -- Feedback Clocks: 1-bit (each) input: Clock feedback ports
   CLKFBIN => CLKFBIN      -- 1-bit input: Feedback clock
);

-- End of MMCME2_BASE_inst instantiation
```

Verilog Instantiation Template

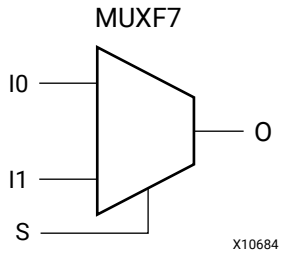
```
// MMCME2_BASE: Base Mixed Mode Clock Manager
//              7 Series
// Xilinx HDL Language Template, version 2020.1

MMCME2_BASE #(
   .BANDWIDTH("OPTIMIZED"),   // Jitter programming (OPTIMIZED, HIGH, LOW)
   .CLKFBOUT_MULT_F(5.0),     // Multiply value for all CLKOUT (2.000-64.000).
   .CLKFBOUT_PHASE(0.0),      // Phase offset in degrees of CLKFB (-360.000-360.000).
   .CLKIN1_PERIOD(0.0),       // Input clock period in ns to ps resolution (i.e. 33.333 is 30 MHz).
   // CLKOUT0_DIVIDE - CLKOUT6_DIVIDE: Divide amount for each CLKOUT (1-128)
   .CLKOUT1_DIVIDE(1),
   .CLKOUT2_DIVIDE(1),
   .CLKOUT3_DIVIDE(1),
   .CLKOUT4_DIVIDE(1),
   .CLKOUT5_DIVIDE(1),
   .CLKOUT6_DIVIDE(1),
   .CLKOUT0_DIVIDE_F(1.0),    // Divide amount for CLKOUT0 (1.000-128.000).
   // CLKOUT0_DUTY_CYCLE - CLKOUT6_DUTY_CYCLE: Duty cycle for each CLKOUT (0.01-0.99).
   .CLKOUT0_DUTY_CYCLE(0.5),
   .CLKOUT1_DUTY_CYCLE(0.5),
   .CLKOUT2_DUTY_CYCLE(0.5),
   .CLKOUT3_DUTY_CYCLE(0.5),
   .CLKOUT4_DUTY_CYCLE(0.5),
   .CLKOUT5_DUTY_CYCLE(0.5),
   .CLKOUT6_DUTY_CYCLE(0.5),
   // CLKOUT0_PHASE - CLKOUT6_PHASE: Phase offset for each CLKOUT (-360.000-360.000).
   .CLKOUT0_PHASE(0.0),
   .CLKOUT1_PHASE(0.0),
   .CLKOUT2_PHASE(0.0),
   .CLKOUT3_PHASE(0.0),
   .CLKOUT4_PHASE(0.0),
   .CLKOUT5_PHASE(0.0),
   .CLKOUT6_PHASE(0.0),
   .CLKOUT4_CASCADE("FALSE"), // Cascade CLKOUT4 counter with CLKOUT6 (FALSE, TRUE)
   .DIVCLK_DIVIDE(1),         // Master division value (1-106)
   .REF_JITTER1(0.0),         // Reference input jitter in UI (0.000-0.999).
   .STARTUP_WAIT("FALSE")     // Delays DONE until MMCM is locked (FALSE, TRUE)
)
MMCME2_BASE_inst (
   // Clock Outputs: 1-bit (each) output: User configurable clock outputs
   .CLKOUT0(CLKOUT0),     // 1-bit output: CLKOUT0
   .CLKOUT0B(CLKOUT0B),   // 1-bit output: Inverted CLKOUT0
   .CLKOUT1(CLKOUT1),     // 1-bit output: CLKOUT1
   .CLKOUT1B(CLKOUT1B),   // 1-bit output: Inverted CLKOUT1
   .CLKOUT2(CLKOUT2),     // 1-bit output: CLKOUT2
   .CLKOUT2B(CLKOUT2B),   // 1-bit output: Inverted CLKOUT2
   .CLKOUT3(CLKOUT3),     // 1-bit output: CLKOUT3
   .CLKOUT3B(CLKOUT3B),   // 1-bit output: Inverted CLKOUT3
   .CLKOUT4(CLKOUT4),     // 1-bit output: CLKOUT4
   .CLKOUT5(CLKOUT5),     // 1-bit output: CLKOUT5
   .CLKOUT6(CLKOUT6),     // 1-bit output: CLKOUT6
   // Feedback Clocks: 1-bit (each) output: Clock feedback ports
   .CLKFBOUT(CLKFBOUT),   // 1-bit output: Feedback clock
   .CLKFBOUTB(CLKFBOUTB), // 1-bit output: Inverted CLKFBOUT
   // Status Ports: 1-bit (each) output: MMCM status ports
   .LOCKED(LOCKED),       // 1-bit output: LOCK
   // Clock Inputs: 1-bit (each) input: Clock input
   .CLKIN1(CLKIN1),       // 1-bit input: Clock
   // Control Ports: 1-bit (each) input: MMCM control ports
   .PWRDWN(PWRDWN),       // 1-bit input: Power-down
   .RST(RST),             // 1-bit input: Reset
   // Feedback Clocks: 1-bit (each) input: Clock feedback ports
   .CLKFBIN(CLKFBIN)      // 1-bit input: Feedback clock
);

// End of MMCME2_BASE_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

MUXF7

Primitive: 2-to-1 Look-Up Table Multiplexer with General Output



Introduction

This design element is a two input multiplexer which, in combination with two LUT6 elements will let you create any 7-input function, an 8-to-1 multiplexer, or other logic functions up to 12-bits wide. Local outputs of the LUT6 element are connected to the I0 and I1 inputs of the MUXF7. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The O output is a general interconnect.

Logic Table

Inputs			Outputs
S	**I0**	**I1**	**O**
0	I0	X	I0
1	X	I1	I1
X	0	0	0
X	1	1	1

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Output of MUX to general routing.
I0	Input	1	Input (tie to LUT6 LO out).
I1	Input	1	Input (tie to LUT6 LO out).
S	Input	1	Input select to MUX.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- MUXF7: CLB MUX to tie two LUT6's together with general output
--        7 Series
-- Xilinx HDL Language Template, version 2020.1

MUXF7_inst : MUXF7
port map (
   O => O,     -- Output of MUX to general routing
   I0 => I0,   -- Input (tie to LUT6 O6 pin)
   I1 => I1,   -- Input (tie to LUT6 O6 pin)
   S => S      -- Input select to MUX
);

-- End of MUXF7_inst instantiation
```

Verilog Instantiation Template

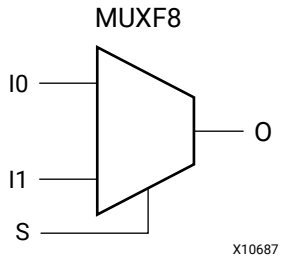
```
// MUXF7: CLB MUX to tie two LUT6's together with general output
//        7 Series
// Xilinx HDL Language Template, version 2020.1

MUXF7 MUXF7_inst (
   .O(O),     // Output of MUX to general routing
   .I0(I0),   // Input (tie to LUT6 O6 pin)
   .I1(I1),   // Input (tie to LUT6 O6 pin)
   .S(S)      // Input select to MUX
);

// End of MUXF7_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

MUXF8

Primitive: 2-to-1 Look-Up Table Multiplexer with General Output



Introduction

This design element is a two input multiplexer which, in combination with two MUXF7 multiplexers and their four associated LUT6 elements, will let you create any 8-input function, a 16-to-1 multiplexer, or other logic functions up to 24-bits wide. Local outputs of MUXF7 are connected to the I0 and I1 inputs of the MUXF8. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1.

The O output is a general interconnect.

Logic Table

Inputs			Outputs
S	**I0**	**I1**	**O**
0	I0	X	I0
1	X	I1	I1
X	0	0	0
X	1	1	1

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Output of MUX to general routing.
I0	Input	1	Input (tie to MUXF7 LO out).
I1	Input	1	Input (tie to MUXF7 LO out).
S	Input	1	Input select to MUX.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- MUXF8: CLB MUX to tie two MUXF7's together with general output
--        7 Series
-- Xilinx HDL Language Template, version 2020.1

MUXF8_inst : MUXF8
port map (
   O => O,     -- Output of MUX to general routing
   I0 => I0,   -- Input (tie to MUXF7 L/LO out)
   I1 => I1,   -- Input (tie to MUXF7 L/LO out)
   S => S      -- Input select to MUX
);

-- End of MUXF8_inst instantiation
```

Verilog Instantiation Template

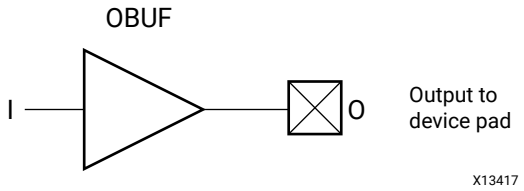
```
// MUXF8: CLB MUX to tie two MUXF7's together with general output
//        7 Series
// Xilinx HDL Language Template, version 2020.1

MUXF8 MUXF8_inst (
   .O(O),     // Output of MUX to general routing
   .I0(I0),   // Input (tie to MUXF7 L/LO out)
   .I1(I1),   // Input (tie to MUXF7 L/LO out)
   .S(S)      // Input select to MUX
);

// End of MUXF8_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

OBUF

Primitive: Output Buffer



Introduction

This design element is a simple output buffer used to drive output signals to the FPGA device pins that do not need to be 3-stated (constantly driven). Either an OBUF, OBUFT, OBUFDS, or OBUFTDS must be connected to every output port in the design.

This element isolates the internal circuit and provides drive current for signals leaving a chip. It exists in input/output blocks (IOB). Its output (O) is connected to an OPAD or an IOPAD. The interface standard used by this element is LVCMOS18. Also, this element has selectable drive and slew rates using the DRIVE and SLOW or FAST constraints.

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Output of OBUF to be connected directly to top-level output port.
I	Input	1	Input of OBUF. Connect to the logic driving the output port.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DRIVE	INTEGER	2, 4, 6, 8, 12, 16, 24	12	Specifies the output current drive strength of the I/O. It is suggested that you set this to the lowest setting tolerable for the design drive and timing requirements.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.

Attribute	Type	Allowed Values	Default	Description
SLEW	STRING	"SLOW" or "FAST"	"SLOW"	Specifies the slew rate of the output driver. Consult the product Data Sheet for recommendations of the best setting for this attribute.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- OBUF: Single-ended Output Buffer
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

OBUF_inst : OBUF
generic map (
   DRIVE => 12,
   IOSTANDARD => "DEFAULT",
   SLEW => "SLOW")
port map (
   O => O,     -- Buffer output (connect directly to top-level port)
   I => I      -- Buffer input
);

-- End of OBUF_inst instantiation
```

Verilog Instantiation Template

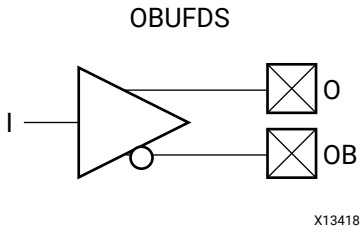
```
// OBUF: Single-ended Output Buffer
//       7 Series
// Xilinx HDL Language Template, version 2020.1

OBUF #(
   .DRIVE(12),   // Specify the output drive strength
   .IOSTANDARD("DEFAULT"), // Specify the output I/O standard
   .SLEW("SLOW") // Specify the output slew rate
) OBUF_inst (
   .O(O),     // Buffer output (connect directly to top-level port)
   .I(I)      // Buffer input
);

// End of OBUF_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

OBUFDS

Primitive: Differential Signaling Output Buffer



Introduction

This design element is a single output buffer that supports low-voltage, differential signaling. OBUFDS isolates the internal circuit and provides drive current for signals leaving the chip. Its output is represented as two distinct ports (O and OB), one deemed the "master" and the other the "slave." The master and the slave are opposite phases of the same logical signal (for example, MYNET and MYNETB).

Logic Table

Inputs	Outputs	
I	**O**	**OB**
0	0	1
1	1	0

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Diff_p output (connect directly to top level port).
OB	Output	1	Diff_n output (connect directly to top level port).
I	Input	1	Buffer input.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
SLEW	STRING	"SLOW" or "FAST"	"SLOW"	Specifies the slew rate of the output driver. Consult the product Data Sheet for recommendations of the best setting for this attribute.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- OBUFDS: Differential Output Buffer
--         7 Series
-- Xilinx HDL Language Template, version 2020.1

OBUFDS_inst : OBUFDS
generic map (
   IOSTANDARD => "DEFAULT", -- Specify the output I/O standard
   SLEW => "SLOW")          -- Specify the output slew rate
port map (
   O => O,      -- Diff_p output (connect directly to top-level port)
   OB => OB,    -- Diff_n output (connect directly to top-level port)
   I => I       -- Buffer input
);

-- End of OBUFDS_inst instantiation
```

Verilog Instantiation Template

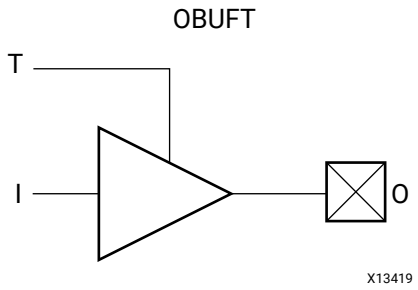
```
// OBUFDS: Differential Output Buffer
//         7 Series
// Xilinx HDL Language Template, version 2020.1

OBUFDS #(
   .IOSTANDARD("DEFAULT"), // Specify the output I/O standard
   .SLEW("SLOW")           // Specify the output slew rate
) OBUFDS_inst (
   .O(O),      // Diff_p output (connect directly to top-level port)
   .OB(OB),    // Diff_n output (connect directly to top-level port)
   .I(I)       // Buffer input
);

// End of OBUFDS_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

OBUFT

Primitive: 3-State Output Buffer with Active Low Output Enable



Introduction

This design element is a single, 3-state output buffer with input I, output O, and active-Low output enables (T). This element uses the LVCMOS18 standard and has selectable drive and slew rates using the DRIVE and SLOW or FAST constraints.

When T is Low, data on the inputs of the buffers is transferred to the corresponding outputs. When T is High, the output is high impedance (off or Z state). OBUFTs are generally used when a single-ended output is needed with a 3-state capability, such as the case when building bidirectional I/O.

Logic Table

Inputs		Outputs
T	**I**	**O**
1	X	Z
0	1	1
0	0	0

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Buffer output (connect directly to top-level port).
I	Input	1	Buffer input.
T	Input	1	3-state enable input.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DRIVE	INTEGER	2, 4, 6, 8, 12, 16, 24	12	Specifies the output current drive strength of the I/O. It is suggested that you set this to the lowest setting tolerable for the design drive and timing requirements.
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
SLEW	STRING	"SLOW" or "FAST"	"SLOW"	Specifies the slew rate of the output driver. See the Data Sheet for recommendations of the best setting for this attribute.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- OBUFT: Single-ended 3-state Output Buffer
--        7 Series
-- Xilinx HDL Language Template, version 2020.1

OBUFT_inst : OBUFT
generic map (
   DRIVE => 12,
   IOSTANDARD => "DEFAULT",
   SLEW => "SLOW")
port map (
   O => O,     -- Buffer output (connect directly to top-level port)
   I => I,     -- Buffer input
   T => T      -- 3-state enable input
);

-- End of OBUFT_inst instantiation
```

Verilog Instantiation Template

```
// OBUFT: Single-ended 3-state Output Buffer
//        All devices
// Xilinx HDL Language Template, version 2020.1

OBUFT #(
   .DRIVE(12),   // Specify the output drive strength
   .IOSTANDARD("DEFAULT"), // Specify the output I/O standard
   .SLEW("SLOW") // Specify the output slew rate
) OBUFT_inst (
   .O(O),     // Buffer output (connect directly to top-level port)
```

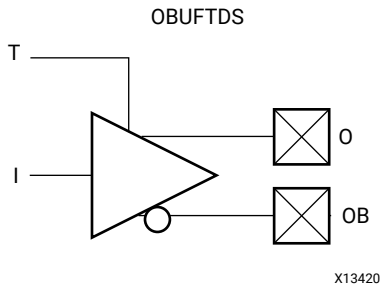
```
   .I(I),      // Buffer input
   .T(T)       // 3-state enable input
);

  // End of OBUFT_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

OBUFTDS

Primitive: 3-State Output Buffer with Differential Signaling, Active-Low Output Enable



Introduction

This design element is an output buffer that supports low-voltage, differential signaling. For the OBUFTDS, a design level interface signal is represented as two distinct ports (O and OB), one deemed the "master" and the other the "slave." The master and the slave are opposite phases of the same logical signal (for example, MYNET_P and MYNET_N).

Logic Table

Inputs		Outputs	
I	**T**	**O**	**OB**
X	1	Z	Z
0	0	0	1
1	0	1	0

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Diff_p output (connect directly to top level port).
OB	Output	1	Diff_n output (connect directly to top level port).
I	Input	1	Buffer input.
T	Input	1	3-state enable input.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
IOSTANDARD	STRING	See Data Sheet	"DEFAULT"	Assigns an I/O standard to the element.
SLEW	STRING	"SLOW" or "FAST"	"SLOW"	Specifies the slew rate of the output driver. Consult the product Data Sheet for recommendations of the best setting for this attribute.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- OBUFTDS: Differential 3-state Output Buffer
--          7 Series
-- Xilinx HDL Language Template, version 2020.1

OBUFTDS_inst : OBUFTDS
generic map (
   IOSTANDARD => "DEFAULT")
port map (
   O => O,     -- Diff_p output (connect directly to top-level port)
   OB => OB,   -- Diff_n output (connect directly to top-level port)
   I => I,     -- Buffer input
   T => T      -- 3-state enable input
);

-- End of OBUFTDS_inst instantiation
```

Verilog Instantiation Template

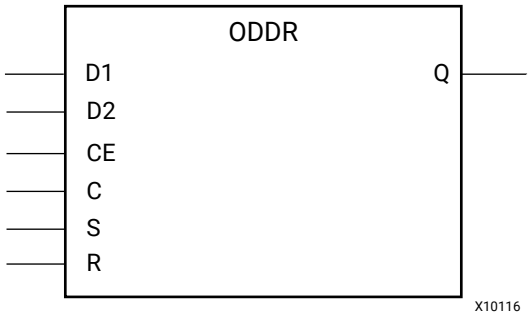
```
// OBUFTDS: Differential 3-state Output Buffer
//          7 Series
// Xilinx HDL Language Template, version 2020.1

OBUFTDS #(
   .IOSTANDARD("DEFAULT"), // Specify the output I/O standard
   .SLEW("SLOW")           // Specify the output slew rate
) OBUFTDS_inst (
   .O(O),     // Diff_p output (connect directly to top-level port)
   .OB(OB),   // Diff_n output (connect directly to top-level port)
   .I(I),     // Buffer input
   .T(T)      // 3-state enable input
);

// End of OBUFTDS_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

ODDR

Primitive: Dedicated Dual Data Rate (DDR) Output Register



Introduction

This design element is a dedicated output register for use in transmitting double data rate (DDR) signals from FPGA devices. The ODDR interface with the FPGA fabric is not limited to opposite clock edges. It can be configured to present date from the FPGA fabric at the same clock edge. This feature allows designers to avoid additional timing complexities and CLB usage. The ODDR also works with SelectIO™ features.

ODDR Modes

This element has two modes of operation. These modes are set by the DDR_CLK_EDGE attribute.

- **OPPOSITE_EDGE mode** The data transmit interface uses classic DDR methodology. Given a data and clock at pin D1-2 and C respectively, D1 is sampled at every positive edge of clock C and D2 is sampled at every negative edge of clock C. Q changes every clock edge.
- **SAME_EDGE mode** Data is still transmitted at the output of the ODDR by opposite edges of clock C. However, the two inputs to the ODDR are clocked with a positive clock edge of clock signal C and an extra register is clocked with a negative clock edge of clock signal C. Using this feature, DDR data can now be presented into the ODDR at the same clock edge.

Port Descriptions

Port	Direction	Width	Function
Q	Output	1	Data Output (DDR): The ODDR output that connects to the IOB pad.
C	Input	1	Clock Input: The C pin represents the clock input pin.
CE	Input	1	Clock Enable Input: When asserted High, this port enables the clock input on port C.
D1 : D2	Input	1 (each)	Data Input: This pin is where the DDR data is presented into the ODDR module.

Port	Direction	Width	Function
R	Input	1	Reset: Depends on how SRTYPE is set.
S	Input	1	Set: Active-High asynchronous set pin. This pin can also be Synchronous depending on the SRTYPE attribute.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DDR_CLK_EDGE	STRING	"OPPOSITE_EDGE", "SAME_EDGE"	"OPPOSITE_EDGE"	DDR clock mode recovery mode selection.
INIT	INTEGER	0, 1	0	Q initialization value.
SRTYPE	STRING	"SYNC", "ASYNC"	"SYNC"	Set/Reset type selection.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- ODDR: Output Double Data Rate Output Register with Set, Reset
--       and Clock Enable.
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

ODDR_inst : ODDR
generic map(
   DDR_CLK_EDGE => "OPPOSITE_EDGE", -- "OPPOSITE_EDGE" or "SAME_EDGE"
   INIT => '0',   -- Initial value for Q port ('1' or '0')
   SRTYPE => "SYNC") -- Reset Type ("ASYNC" or "SYNC")
port map (
   Q => Q,    -- 1-bit DDR output
   C => C,    -- 1-bit clock input
   CE => CE,  -- 1-bit clock enable input
   D1 => D1,  -- 1-bit data input (positive edge)
   D2 => D2,  -- 1-bit data input (negative edge)
   R => R,    -- 1-bit reset input
   S => S     -- 1-bit set input
);

-- End of ODDR_inst instantiation
```

Verilog Instantiation Template

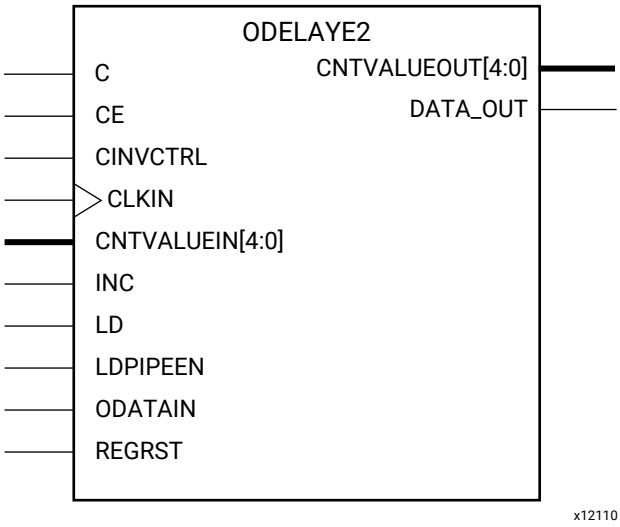
```
// ODDR: Output Double Data Rate Output Register with Set, Reset
//       and Clock Enable.
//       7 Series
// Xilinx HDL Language Template, version 2020.1

ODDR #(
   .DDR_CLK_EDGE("OPPOSITE_EDGE"), // "OPPOSITE_EDGE" or "SAME_EDGE"
   .INIT(1'b0),    // Initial value of Q: 1'b0 or 1'b1
   .SRTYPE("SYNC") // Set/Reset type: "SYNC" or "ASYNC"
) ODDR_inst (
   .Q(Q),   // 1-bit DDR output
   .C(C),   // 1-bit clock input
   .CE(CE), // 1-bit clock enable input
   .D1(D1), // 1-bit data input (positive edge)
   .D2(D2), // 1-bit data input (negative edge)
   .R(R),   // 1-bit reset
   .S(S)    // 1-bit set
);

// End of ODDR_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

ODELAYE2

Primitive: Output Fixed or Variable Delay Element



Introduction

This design element can be used to provide a fixed delay or an adjustable delay to the output path of the 7 series FPGA. This delay can be useful for the purpose of external data alignment, external phase offset and simultaneous switching noise (SSN) mitigation, as well as allowing for the tracking of external data alignment over process, temperature, and voltage (PVT). When used with the IDELAYCTRL component circuitry, can provide precise time increments of delay. When used in variable mode, the output path can be adjusted for increasing and decreasing amounts of delay. The ODELAYE2 is not available on the High Range (HR) banks in the 7 series devices.

Port Descriptions

Port	Direction	Width	Function
C	Input	1	All control inputs to ODELAYE2 primitive (CNTVALUEIN, RST, CE, LD, LDPIPEEN and INC) are synchronous to the clock input (C). A clock must be connected to this port when the ODELAYE2 is configured in "VARIABLE", "VAR_LOAD" or "VAR_LOAD_PIPE" mode. C can be locally inverted, and must be supplied by a global or regional clock buffer. This clock should be connected to the same clock in the SelectIO logic resources (when using OSERDESE2, C is connected to CLKDIV). If the ODELAYE2 is configured as "FIXED", connect this port to gnd.
CE	Input	1	Active-High enable increment/decrement function. If the ODELAYE2 is configured as "FIXED", connect this port to gnd.

Port	Direction	Width	Function
CINVCTRL	Input	1	The CINVCTRL pin is used for dynamically switching the polarity of C pin. This is for use in applications when glitches are not an issue. When switching the polarity, do not use the ODELAYE2 control pins for two clock cycles. If the ODELAYE2 is configured as "FIXED", connect this port to gnd.
CLKIN	Input	1	Delayed Clock input into the ODELAYE2.
CNTVALUEIN<4:0>	Input	5	Counter value from FPGA logic for dynamically loadable tap value input when configured in "VAR_LOAD" or "VAR_LOAD_PIPE" modes. If the ODELAYE2 is configured as "FIXED" or "VARIABLE", connect this port to gnd.
CNTVALUEOUT<4:0>	Output	5	The CNTVALUEOUT pins are used for reporting the dynamically switching value of the delay element. CNTVALUEOUT is only available when ODELAYE2 is in "VAR_LOAD" or "VAR_LOAD_PIPE" mode.
DATAOUT	Output	1	Delayed data/clock from either the CLKIN or ODATAIN ports. DATAOUT connects to an I/O port in the case of data or back to the clocking structure in the case of a clock..
INC	Input	1	The increment/decrement is controlled by the enable signal (CE). This interface is only available when ODELAYE2 is in VARIABLE, VAR_LOAD, or VAR_LOAD_PIPE mode.
LD	Input	1	Load initial value or loaded value to the counter.
LDPIPEEN	Input	1	Enable PIPELINE register to load data from LD pins.
ODATAIN	Input	1	The ODATAIN input is the output data to be delayed driven by the OSERDESE2 or output register.
REGRST	Input	1	The REGRST signal is an active-High reset and is synchronous to the input clock signal (C). When asserted, the tap value reverts to a zero state unless LDPIPEEN is also asserted in which case the tap value results in the value on the CNTVALUEIN port.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
CINVCTRL_SEL	STRING	"FALSE", "TRUE"	"FALSE"	Enables the CINVCTRL_SEL pin to dynamically switch the polarity of the C pin.
DELAY_SRC	STRING	"ODATAIN", "CLKIN"	"ODATAIN"	Select the data input source: • "ODATAIN": ODELAYE2 chain input is ODATAIN • "CLKIN": ODELAYE2 chain input is CLKIN

Attribute	Type	Allowed Values	Default	Description
HIGH _PERFORMANCE _MODE	STRING	"FALSE", "TRUE"	"FALSE"	When TRUE, this attribute reduces the output jitter. When FALSE, power consumption is reduced. The difference in power consumption is quantified in the Xilinx Power Estimator tool.
ODELAY_TYPE	STRING	"FIXED", "VARIABLE", "VAR_LOAD", "VAR_LOAD_PIPE"	"FIXED"	Sets the type of tap delay line. • "FIXED": Sets a static delay value • "VARIABLE": Dynamically adjust (increment/decrement) delay value • "VAR_LOAD": Dynamically loads tap values • "VAR_LOAD_PIPE": Pipelined dynamically loadable tap values
ODELAY_VALUE	DECIMAL	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31	0	Specifies the fixed number of delay taps in fixed mode or the initial starting number of taps in "VARIABLE" mode (output path). When IDELAY_TYPE is set to "VAR_LOAD" or "VAR_LOAD_PIPE" mode, this value is ignored.
PIPE_SEL	STRING	"FALSE", "TRUE"	"FALSE"	Select pipelined mode.
REFCLK _FREQUENCY	1 significant digit FLOAT	190-210, 290-310 Mhz	200.0	Sets the tap value (in Mhz) used by the Timing Analyzer for static timing analysis and functional/timing simulation. The frequency of REFCLK must be within the given datasheet range to guarantee the tap-delay value and performance.
SIGNAL_PATTERN	STRING	"DATA", "CLOCK"	"DATA"	Causes timing analysis to account for the appropriate amount of delay-chain jitter when presented with either a "DATA" pattern with irregular transitions or a "CLOCK" pattern with a regular rise/fall pattern.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- ODELAYE2: Output Fixed or Variable Delay Element
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

ODELAYE2_inst : ODELAYE2
generic map (
   CINVCTRL_SEL => "FALSE",          -- Enable dynamic clock inversion (FALSE, TRUE)
   DELAY_SRC => "ODATAIN",           -- Delay input (ODATAIN, CLKIN)
   HIGH_PERFORMANCE_MODE => "FALSE", -- Reduced jitter ("TRUE"), Reduced power ("FALSE")
   ODELAY_TYPE => "FIXED",           -- FIXED, VARIABLE, VAR_LOAD, VAR_LOAD_PIPE
   ODELAY_VALUE => 0,                -- Output delay tap setting (0-31)
   PIPE_SEL => "FALSE",              -- Select pipelined mode, FALSE, TRUE
   REFCLK_FREQUENCY => 200.0,        -- IDELAYCTRL clock input frequency in MHz (190.0-210.0, 290.0-310.0).
   SIGNAL_PATTERN => "DATA"          -- DATA, CLOCK input signal
)
port map (
   CNTVALUEOUT => CNTVALUEOUT, -- 5-bit output: Counter value output
   DATAOUT => DATAOUT,         -- 1-bit output: Delayed data/clock output
   C => C,                     -- 1-bit input: Clock input
   CE => CE,                   -- 1-bit input: Active high enable increment/decrement input
```

```
   CINVCTRL => CINVCTRL,       -- 1-bit input: Dynamic clock inversion input
   CLKIN => CLKIN,             -- 1-bit input: Clock delay input
   CNTVALUEIN => CNTVALUEIN,   -- 5-bit input: Counter value input
   INC => INC,                 -- 1-bit input: Increment / Decrement tap delay input
   LD => LD,                   -- 1-bit input: Loads ODELAY_VALUE tap delay in VARIABLE mode, in VAR_LOAD or
                               -- VAR_LOAD_PIPE mode, loads the value of CNTVALUEIN

   LDPIPEEN => LDPIPEEN,       -- 1-bit input: Enables the pipeline register to load data
   ODATAIN => ODATAIN,         -- 1-bit input: Output delay data input
   REGRST => REGRST            -- 1-bit input: Active-high reset tap-delay input
);

-- End of ODELAYE2_inst instantiation
```

Verilog Instantiation Template

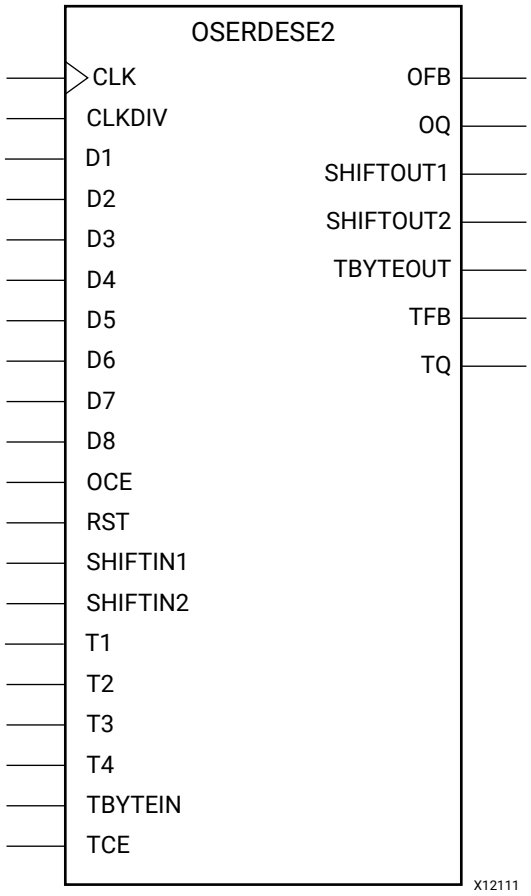
```
// ODELAYE2: Output Fixed or Variable Delay Element
//           7 Series
// Xilinx HDL Language Template, version 2020.1

(* IODELAY_GROUP = <iodelay_group_name> *) // Specifies group name for associated IDELAYs/ODELAYs and IDELAYCTRL

ODELAYE2 #(
   .CINVCTRL_SEL("FALSE"),          // Enable dynamic clock inversion (FALSE, TRUE)
   .DELAY_SRC("ODATAIN"),           // Delay input (ODATAIN, CLKIN)
   .HIGH_PERFORMANCE_MODE("FALSE"), // Reduced jitter ("TRUE"), Reduced power ("FALSE")
   .ODELAY_TYPE("FIXED"),           // FIXED, VARIABLE, VAR_LOAD, VAR_LOAD_PIPE
   .ODELAY_VALUE(0),                // Output delay tap setting (0-31)
   .PIPE_SEL("FALSE"),              // Select pipelined mode, FALSE, TRUE
   .REFCLK_FREQUENCY(200.0),       // IDELAYCTRL clock input frequency in MHz (190.0-210.0, 290.0-310.0).
   .SIGNAL_PATTERN("DATA")          // DATA, CLOCK input signal
)
ODELAYE2_inst (
   .CNTVALUEOUT(CNTVALUEOUT), // 5-bit output: Counter value output
   .DATAOUT(DATAOUT),         // 1-bit output: Delayed data/clock output
   .C(C),                     // 1-bit input: Clock input
   .CE(CE),                   // 1-bit input: Active high enable increment/decrement input
   .CINVCTRL(CINVCTRL),       // 1-bit input: Dynamic clock inversion input
   .CLKIN(CLKIN),             // 1-bit input: Clock delay input
   .CNTVALUEIN(CNTVALUEIN),   // 5-bit input: Counter value input
   .INC(INC),                 // 1-bit input: Increment / Decrement tap delay input
   .LD(LD),                  // 1-bit input: Loads ODELAY_VALUE tap delay in VARIABLE mode, in VAR_LOAD or
                              // VAR_LOAD_PIPE mode, loads the value of CNTVALUEIN

   .LDPIPEEN(LDPIPEEN),       // 1-bit input: Enables the pipeline register to load data
   .ODATAIN(ODATAIN),         // 1-bit input: Output delay data input
   .REGRST(REGRST)            // 1-bit input: Active-high reset tap-delay input
);

// End of ODELAYE2_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

OSERDESE2

Primitive: Output SERial/DESerializer with bitslip



Introduction

The OSERDESE2 is a dedicated parallel-to-serial converter with specific clocking and logic resources designed to facilitate the implementation of high-speed source-synchronous interfaces. Every OSERDESE2 module includes a dedicated serializer for data and 3-state control. Both data and 3-state serializers can be configured in single data rate (SDR) and double data rate (DDR) mode. Data serialization can be up to 8:1 (10:1 or 14:1 if using OSERDESE2 Width Expansion). 3-state serialization can be up to 4:1.

Port Descriptions

Port	Direction	Width	Function
CLK	Input	1	A high speed clock input that drives the serial side of the parallel-to-serial converters.

Port	Direction	Width	Function
CLKDIV	Input	1	A divided high-speed clock input that drives the parallel side of the parallel-to-serial converters. This clock is the divided version of the clock connected to the CLK port.
D1 - D8	Input	1	Incoming parallel data enters the module through ports D1 to D8. These ports are connected to the FPGA fabric, and can be configured from two to eight bits (i.e., a 8:1 serialization). Bit widths greater than six (up to 14) can be supported by using a second OSERDESE2 in SLAVE mode.
OCE	Input	1	OCE is an active-High clock enable for the data path.
OFB	Output	1	The output feedback port (OFB) is the serial (high-speed) data output port of the OSERDESE2.
OQ	Output	1	The OQ port is the data output port of the module. Data at the input port D1 will appear first at OQ. This port connects the output of the data parallel-to-serial converter to the data input of the IOB. This port can not drive the ODELAYE2; the OFB pin must be used.
RST	Input	1	The reset input causes the outputs of all data flip-flops in the CLK and CLKDIV domains to be driven Low asynchronously. OSERDESE2 circuits running in the CLK domain where timing is critical use an internal, dedicated circuit to retime the RST input to produce a reset signal synchronous to the CLK domain. Similarly, there is a dedicated circuit to retime the RST input to produce a reset signal synchronous to the CLKDIV domain. Because there are OSERDESE2 circuits that retime the RST input, you only need to provide a reset pulse to the RST input that meets timing on the CLKDIV frequency domain (synchronous to CLKDIV). Therefore, RST should be driven High for a minimum of one CLKDIV cycle. When building an interface consisting of multiple OSERDESE2 ports, all ports must be synchronized. The internal retiming of the RST input is designed so that all OSERDESE2 blocks that receive the same reset pulse come out of reset synchronized with one another.
SHIFTIN1 / SHIFTIN2	Input	1	Cascade Input for data input expansion. Connect to SHIFTOUT1/2 of slave.
SHIFTOUT1 / SHIFTOUT2	Output	1	Cascade out for data input expansion. Connect to SHIFTIN1/2 of master.
TBYTEIN	Input	1	Byte group tristate input from source
TBYTEOUT	Output	1	Byte group tristate output to IOB
TCE	Input	1	Active-High clock enable for the 3-state control path.
TFB	Output	1	3-state control output of the module sent to the ODELAYE2. When used, this port connects the output of the 3-state parallel-to-serial converter to the control/3-state input of the ODELAYE2.
TQ	Output	1	This port is the 3-state control output of the module. When used, this port connects the output of the 3-state parallel-to-serial converter to the control/3-state input of the IOB.
T1 - T4	Input	1	Parallel 3-state signals enter the module through ports T1 to T4. The ports are connected to the FPGA fabric, and can be configured as one, two, or four bits.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
DATA_RATE_OQ	STRING	"DDR", "SDR"	"DDR"	Defines whether data is processed as single data rate (SDR) or double data rate (DDR).
DATA_RATE_TQ	STRING	"DDR", "BUF", "SDR"	"DDR"	Defines whether 3-state control is to be processed as single data rate (SDR) or double data rate (DDR).
DATA_WIDTH	DECIMAL	4, 2, 3, 5, 6, 7, 8, 10, 14	4	Defines the parallel data input width of the parallel-to-serial converter. Possible values depend on the DATA_RATE_OQ attribute. When DATA_RATE_OQ is SDR, possible values are 2, 3, 4, 5, 6, 7, and 8. When DATA_RATE_OQ is DDR, the possible values for the DATA_WIDTH attribute are 4, 6, 8, 10 and 14. When DATA_WIDTH is larger than eight, a pair of OSERDESE2 must be configured into a master-slave configuration.
INIT_OQ	BINARY	1'b0 to 1'b1	1'b0	Defines the initial value of OQ output.
INIT_TQ	BINARY	1'b0 to 1'b1	1'b0	Defines the initial value of TQ output.
SERDES_MODE	STRING	"MASTER", "SLAVE"	"MASTER"	Defines whether the module is a master or slave when using width expansion.
SRVAL_OQ	BINARY	1'b0 to 1'b1	1'b0	Defines the value of OQ outputs when the SR is invoked.
SRVAL_TQ	BINARY	1'b0 to 1'b1	1'b0	Defines the value of YQ outputs when the SR is invoked.
TBYTE_CTL	STRING	"FALSE", "TRUE"	"FALSE"	Enable Tristate BYTE operation for DDR3 mode. This allows the tristate signal to take value from one of the tristate outputs which is acting as a source.
TBYTE_SRC	STRING	"FALSE", "TRUE"	"FALSE"	Enable OSERDESE2 to act as a source for Tristate Byte operation in DDR3 mode.
TRISTATE_WIDTH	DECIMAL	4, 1	4	Defines the parallel 3-state input width of the 3-state control parallel-to-serial converter. Possible values depend on the DATA_RATE_TQ attribute. When DATA_RATE_TQ is SDR or BUF, the TRISTATE_WIDTH attribute can only be set to 1. When DATA_RATE_TQ = DDR, the possible values for the TRISTATE_WIDTH attribute is 4. TRISTATE_WIDTH cannot be set to widths larger than 4. When a DATA_WIDTH is larger than four, set the TRISTATE_WIDTH to 1.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- OSERDESE2: Output SERial/DESerializer with bitslip
--            7 Series
-- Xilinx HDL Language Template, version 2020.1

OSERDESE2_inst : OSERDESE2
generic map (
   DATA_RATE_OQ => "DDR",   -- DDR, SDR
   DATA_RATE_TQ => "DDR",   -- DDR, BUF, SDR
   DATA_WIDTH => 4,         -- Parallel data width (2-8,10,14)
   INIT_OQ => '0',          -- Initial value of OQ output (1'b0,1'b1)
   INIT_TQ => '0',          -- Initial value of TQ output (1'b0,1'b1)
   SERDES_MODE => "MASTER", -- MASTER, SLAVE
   SRVAL_OQ => '0',         -- OQ output value when SR is used (1'b0,1'b1)
   SRVAL_TQ => '0',         -- TQ output value when SR is used (1'b0,1'b1)
   TBYTE_CTL => "FALSE",    -- Enable tristate byte operation (FALSE, TRUE)
   TBYTE_SRC => "FALSE",    -- Tristate byte source (FALSE, TRUE)
   TRISTATE_WIDTH => 4      -- 3-state converter width (1,4)
)
port map (
   OFB => OFB,              -- 1-bit output: Feedback path for data
   OQ => OQ,                -- 1-bit output: Data path output
   -- SHIFTOUT1 / SHIFTOUT2: 1-bit (each) output: Data output expansion (1-bit each)
   SHIFTOUT1 => SHIFTOUT1,
   SHIFTOUT2 => SHIFTOUT2,
   TBYTEOUT => TBYTEOUT,    -- 1-bit output: Byte group tristate
   TFB => TFB,              -- 1-bit output: 3-state control
   TQ => TQ,                -- 1-bit output: 3-state control
   CLK => CLK,              -- 1-bit input: High speed clock
   CLKDIV => CLKDIV,        -- 1-bit input: Divided clock
   -- D1 - D8: 1-bit (each) input: Parallel data inputs (1-bit each)
   D1 => D1,
   D2 => D2,
   D3 => D3,
   D4 => D4,
   D5 => D5,
   D6 => D6,
   D7 => D7,
   D8 => D8,
   OCE => OCE,              -- 1-bit input: Output data clock enable
   RST => RST,              -- 1-bit input: Reset
   -- SHIFTIN1 / SHIFTIN2: 1-bit (each) input: Data input expansion (1-bit each)
   SHIFTIN1 => SHIFTIN1,
   SHIFTIN2 => SHIFTIN2,
   -- T1 - T4: 1-bit (each) input: Parallel 3-state inputs
   T1 => T1,
   T2 => T2,
   T3 => T3,
   T4 => T4,
   TBYTEIN => TBYTEIN,      -- 1-bit input: Byte group tristate
   TCE => TCE               -- 1-bit input: 3-state clock enable
);

-- End of OSERDESE2_inst instantiation
```

Verilog Instantiation Template

```
// OSERDESE2: Output SERial/DESerializer with bitslip
//            7 Series
// Xilinx HDL Language Template, version 2020.1

OSERDESE2 #(
   .DATA_RATE_OQ("DDR"),   // DDR, SDR
   .DATA_RATE_TQ("DDR"),   // DDR, BUF, SDR
```

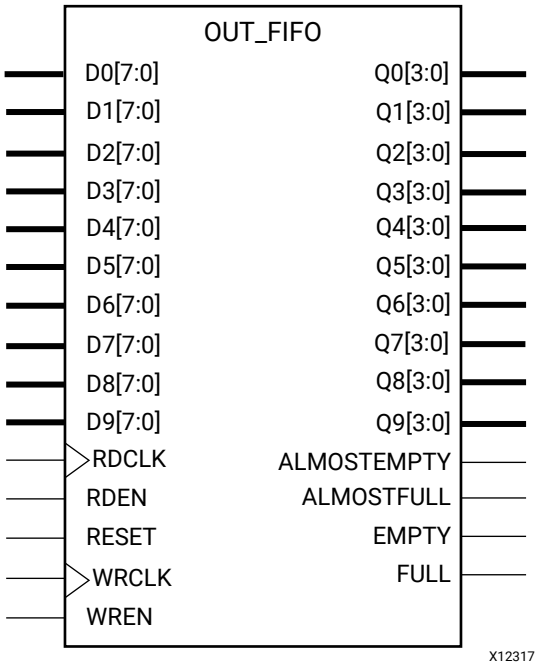
```
   .DATA_WIDTH(4),          // Parallel data width (2-8,10,14)
   .INIT_OQ(1'b0),          // Initial value of OQ output (1'b0,1'b1)
   .INIT_TQ(1'b0),          // Initial value of TQ output (1'b0,1'b1)
   .SERDES_MODE("MASTER"), // MASTER, SLAVE
   .SRVAL_OQ(1'b0),         // OQ output value when SR is used (1'b0,1'b1)
   .SRVAL_TQ(1'b0),         // TQ output value when SR is used (1'b0,1'b1)
   .TBYTE_CTL("FALSE"),     // Enable tristate byte operation (FALSE, TRUE)
   .TBYTE_SRC("FALSE"),     // Tristate byte source (FALSE, TRUE)
   .TRISTATE_WIDTH(4)       // 3-state converter width (1,4)
)
OSERDESE2_inst (
   .OFB(OFB),              // 1-bit output: Feedback path for data
   .OQ(OQ),                // 1-bit output: Data path output
   // SHIFTOUT1 / SHIFTOUT2: 1-bit (each) output: Data output expansion (1-bit each)
   .SHIFTOUT1(SHIFTOUT1),
   .SHIFTOUT2(SHIFTOUT2),
   .TBYTEOUT(TBYTEOUT),    // 1-bit output: Byte group tristate
   .TFB(TFB),              // 1-bit output: 3-state control
   .TQ(TQ),                // 1-bit output: 3-state control
   .CLK(CLK),              // 1-bit input: High speed clock
   .CLKDIV(CLKDIV),        // 1-bit input: Divided clock
   // D1 - D8: 1-bit (each) input: Parallel data inputs (1-bit each)
   .D1(D1),
   .D2(D2),
   .D3(D3),
   .D4(D4),
   .D5(D5),
   .D6(D6),
   .D7(D7),
   .D8(D8),
   .OCE(OCE),              // 1-bit input: Output data clock enable
   .RST(RST),              // 1-bit input: Reset
   // SHIFTIN1 / SHIFTIN2: 1-bit (each) input: Data input expansion (1-bit each)
   .SHIFTIN1(SHIFTIN1),
   .SHIFTIN2(SHIFTIN2),
   // T1 - T4: 1-bit (each) input: Parallel 3-state inputs
   .T1(T1),
   .T2(T2),
   .T3(T3),
   .T4(T4),
   .TBYTEIN(TBYTEIN),      // 1-bit input: Byte group tristate
   .TCE(TCE)               // 1-bit input: 3-state clock enable
);

// End of OSERDESE2_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

OUT_FIFO

Primitive: Output First-In, First-Out (FIFO) Buffer



The Output FIFO is a new resource located next to the I/O. This dedicated hardware is designed to help transition the data from fabric to the I/O, ODDR or OSERDESE2. It has two basic modes the first is a 4x4 mode where the data coming into the FIFO goes out at the same rate. The second mode is a 8x4 mode where the data coming out is serialized by a factor of 2. In other words, in 8x4 mode 8-bits go to the OUT_FIFO and 4-bits come out.

The Output FIFO is a new resource located next to the I/O. This dedicated hardware is designed to help transition the data from fabric to the I/O, ODDR or OSERDESE2. It has two basic modes the first is a 4x4 mode where the data coming into the FIFO goes out at the same rate. The second mode is a 8x4 mode where the data coming out is serialized by a factor of 2. In other words in 8x4 mode, 8-bits go to the OUT_FIFO and 4-bits come out. Features of this component include:

- Array dimensions: 80 wide, 8 deep (8x4 mode); 40 wide, 8 deep (4x4 mode)
- Empty and Full flags
- Programmable Almost Empty and Almost Full flags

Port Descriptions

Port	Type	Width	Function
ALMOSTEMPTY	Output	1	Active-High output flag indicating the FIFO is almost empty. The threshold of the almost empty flag is set by the ALMOST_EMPTY_VALUE attribute.
ALMOSTFULL	Output	1	Active-High output flag indicating the FIFO is almost full. The threshold of the almost empty flag is set by the ALMOST_FULL_VALUE attribute.
D0<7:0>	Input	8	Channel 0 input bus.
D1<7:0>	Input	8	Channel 1 input bus.
D2<7:0>	Input	8	Channel 2 input bus.
D3<7:0>	Input	8	Channel 3 input bus.
D4<7:0>	Input	8	Channel 4 input bus.
D5<7:0>	Input	8	Channel 5 input bus.
D6<7:0>	Input	8	Channel 6 input bus.
D7<7:0>	Input	8	Channel 7 input bus.
D8<7:0>	Input	8	Channel 8 input bus.
D9<7:0>	Input	8	Channel 9 input bus.
EMPTY	Output	1	Active-High output flag indicating the FIFO is empty.
FULL	Output	1	Active-High output flag indicating the FIFO is full.
Q0<3:0>	Output	4	Channel 0 output bus.
Q1<3:0>	Output	4	Channel 1 output bus.
Q2<3:0>	Output	4	Channel 2 output bus.
Q3<3:0>	Output	4	Channel 3 output bus.
Q4<3:0>	Output	4	Channel 4 output bus.
Q5<7:0>	Output	8	Channel 5 output bus.
Q6<7:0>	Output	8	Channel 6 output bus.
Q7<3:0>	Output	4	Channel 7 output bus.
Q8<3:0>	Output	4	Channel 8 output bus.
Q9<3:0>	Output	4	Channel 9 output bus.
RDCLK	Input	1	Read clock.
RDEN	Input	1	Active-High read enable.
RESET	Input	1	Active-High asynchronous reset.
WRCLK	Input	1	Write clock.
WREN	Input	1	Active-High write enable.

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
ALMOST_EMPTY _VALUE	DECIMAL	1, 2	1	Specifies the number of entries left before asserting the ALMOSTEMPTY output signal.
ALMOST_FULL _VALUE	DECIMAL	1, 2	1	Specifies the number of entries left before asserting the ALMOSTFULL output signal.
ARRAY_MODE	STRING	"ARRAY_MODE_8_X_4", "ARRAY_MODE_4_X_4"	"ARRAY_MODE _8_X_4"	Specifies serializer mode: • "ARRAY_MODE_4_X_4": Four bits in, four bits out • "ARRAY_MODE_4_X_8": Four bits in, eight bits out
OUTPUT _DISABLE	STRING	"FALSE", "TRUE"	"FALSE"	Disable output.
SYNCHRONOUS _MODE	STRING	"FALSE"	"FALSE"	Must always be set to false.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- OUT_FIFO: Output First-In, First-Out (FIFO) Buffer
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

OUT_FIFO_inst : OUT_FIFO
generic map (
   ALMOST_EMPTY_VALUE => 1,          -- Almost empty offset (1-2)
   ALMOST_FULL_VALUE => 1,           -- Almost full offset (1-2)
   ARRAY_MODE => "ARRAY_MODE_8_X_4", -- ARRAY_MODE_8_X_4, ARRAY_MODE_4_X_4
   OUTPUT_DISABLE => "FALSE",        -- Disable output (FALSE, TRUE)
   SYNCHRONOUS_MODE => "FALSE"       -- Must always be set to false.
)
port map (
   -- FIFO Status Flags: 1-bit (each) output: Flags and other FIFO status outputs
   ALMOSTEMPTY => ALMOSTEMPTY, -- 1-bit output: Almost empty flag
   ALMOSTFULL => ALMOSTFULL,   -- 1-bit output: Almost full flag
   EMPTY => EMPTY,             -- 1-bit output: Empty flag
   FULL => FULL,               -- 1-bit output: Full flag
   -- Q0-Q9: 4-bit (each) output: FIFO Outputs
   Q0 => Q0,                   -- 4-bit output: Channel 0 output bus
   Q1 => Q1,                   -- 4-bit output: Channel 1 output bus
   Q2 => Q2,                   -- 4-bit output: Channel 2 output bus
   Q3 => Q3,                   -- 4-bit output: Channel 3 output bus
   Q4 => Q4,                   -- 4-bit output: Channel 4 output bus
   Q5 => Q5,                   -- 8-bit output: Channel 5 output bus
   Q6 => Q6,                   -- 8-bit output: Channel 6 output bus
   Q7 => Q7,                   -- 4-bit output: Channel 7 output bus
   Q8 => Q8,                   -- 4-bit output: Channel 8 output bus
   Q9 => Q9,                   -- 4-bit output: Channel 9 output bus
   -- D0-D9: 8-bit (each) input: FIFO inputs
   D0 => D0,                   -- 8-bit input: Channel 0 input bus
   D1 => D1,                   -- 8-bit input: Channel 1 input bus
   D2 => D2,                   -- 8-bit input: Channel 2 input bus
   D3 => D3,                   -- 8-bit input: Channel 3 input bus
   D4 => D4,                   -- 8-bit input: Channel 4 input bus
   D5 => D5,                   -- 8-bit input: Channel 5 input bus
```

```
   D6 => D6,                   -- 8-bit input: Channel 6 input bus
   D7 => D7,                   -- 8-bit input: Channel 7 input bus
   D8 => D8,                   -- 8-bit input: Channel 8 input bus
   D9 => D9,                   -- 8-bit input: Channel 9 input bus
   -- FIFO Control Signals: 1-bit (each) input: Clocks, Resets and Enables
   RDCLK => RDCLK,             -- 1-bit input: Read clock
   RDEN => RDEN,               -- 1-bit input: Read enable
   RESET => RESET,             -- 1-bit input: Active high reset
   WRCLK => WRCLK,             -- 1-bit input: Write clock
   WREN => WREN                -- 1-bit input: Write enable
);

-- End of OUT_FIFO_inst instantiation
```

Verilog Instantiation Template

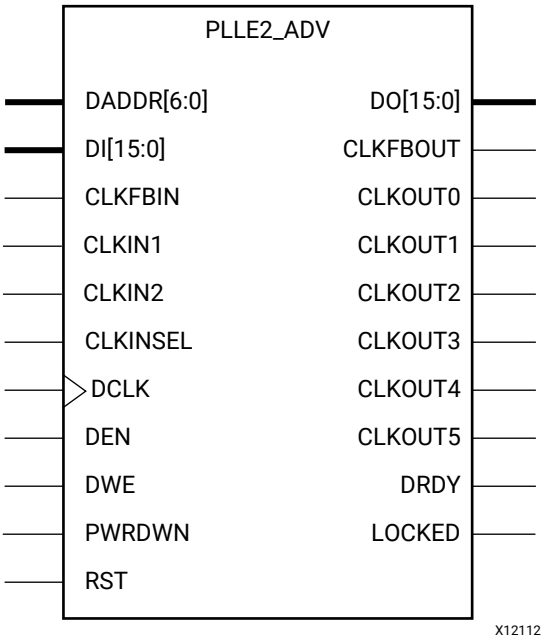
```
// OUT_FIFO: Output First-In, First-Out (FIFO) Buffer
//           7 Series
// Xilinx HDL Language Template, version 2020.1

OUT_FIFO #(
   .ALMOST_EMPTY_VALUE(1),          // Almost empty offset (1-2)
   .ALMOST_FULL_VALUE(1),           // Almost full offset (1-2)
   .ARRAY_MODE("ARRAY_MODE_8_X_4"), // ARRAY_MODE_8_X_4, ARRAY_MODE_4_X_4
   .OUTPUT_DISABLE("FALSE"),        // Disable output (FALSE, TRUE)
   .SYNCHRONOUS_MODE("FALSE")       // Must always be set to false.
)
OUT_FIFO_inst (
   // FIFO Status Flags: 1-bit (each) output: Flags and other FIFO status outputs
   .ALMOSTEMPTY(ALMOSTEMPTY), // 1-bit output: Almost empty flag
   .ALMOSTFULL(ALMOSTFULL),   // 1-bit output: Almost full flag
   .EMPTY(EMPTY),             // 1-bit output: Empty flag
   .FULL(FULL),               // 1-bit output: Full flag
   // Q0-Q9: 4-bit (each) output: FIFO Outputs
   .Q0(Q0),                   // 4-bit output: Channel 0 output bus
   .Q1(Q1),                   // 4-bit output: Channel 1 output bus
   .Q2(Q2),                   // 4-bit output: Channel 2 output bus
   .Q3(Q3),                   // 4-bit output: Channel 3 output bus
   .Q4(Q4),                   // 4-bit output: Channel 4 output bus
   .Q5(Q5),                   // 8-bit output: Channel 5 output bus
   .Q6(Q6),                   // 8-bit output: Channel 6 output bus
   .Q7(Q7),                   // 4-bit output: Channel 7 output bus
   .Q8(Q8),                   // 4-bit output: Channel 8 output bus
   .Q9(Q9),                   // 4-bit output: Channel 9 output bus
   // D0-D9: 8-bit (each) input: FIFO inputs
   .D0(D0),                   // 8-bit input: Channel 0 input bus
   .D1(D1),                   // 8-bit input: Channel 1 input bus
   .D2(D2),                   // 8-bit input: Channel 2 input bus
   .D3(D3),                   // 8-bit input: Channel 3 input bus
   .D4(D4),                   // 8-bit input: Channel 4 input bus
   .D5(D5),                   // 8-bit input: Channel 5 input bus
   .D6(D6),                   // 8-bit input: Channel 6 input bus
   .D7(D7),                   // 8-bit input: Channel 7 input bus
   .D8(D8),                   // 8-bit input: Channel 8 input bus
   .D9(D9),                   // 8-bit input: Channel 9 input bus
   // FIFO Control Signals: 1-bit (each) input: Clocks, Resets and Enables
   .RDCLK(RDCLK),             // 1-bit input: Read clock
   .RDEN(RDEN),               // 1-bit input: Read enable
   .RESET(RESET),             // 1-bit input: Active high reset
   .WRCLK(WRCLK),             // 1-bit input: Write clock
   .WREN(WREN)                // 1-bit input: Write enable
);

// End of OUT_FIFO_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

PLLE2_ADV

Primitive: Advanced Phase Locked Loop (PLL)



Introduction

PLLE2 is a mixed signal block designed to support frequency synthesis, clock network deskew, and jitter reduction. The clock outputs can each have an individual divide (1 to 128), phase shift, and duty cycle based on the same VCO frequency. Output clocks are phase aligned to each other (unless phase shifted) and aligned to the input clock with a proper feedback configuration.

PLLE2 complements the MMCM element by supporting higher speed clocking while MMCM has more features to handle most general clocking needs. PLLE2_BASE is intended for most uses of this PLL component while PLLE2_ADV is intended for use when clock switch-over or dynamic reconfiguration is required.

Port Descriptions

Port	Direction	Width	Function
CLKFBIN	Input	1	Feedback clock pin to the PLL.
CLKFBOUT	Output	1	Dedicated PLL Feedback clock output.
CLKINSEL	Input	1	Signal controls the state of the input MUX. • High = CLKIN1 • Low = CLKIN2

Port	Direction	Width	Function
CLKIN1	Input	1	Primary clock input.
CLKIN2	Input	1	Secondary clock input.
CLKOUT0	Output	1	CLKOUT0 output.
CLKOUT1	Output	1	Configurable clock output CLKOUT1.
CLKOUT2	Output	1	Configurable clock output CLKOUT2.
CLKOUT3	Output	1	Configurable clock output CLKOUT3.
CLKOUT4	Output	1	Configurable clock output CLKOUT4.
CLKOUT5	Output	1	Configurable clock output CLKOUT5.
DADDR<6:0>	Input	7	The dynamic reconfiguration address (DADDR) input bus provides a reconfiguration address for the dynamic reconfiguration. When not used, all bits must be assigned zeros.
DCLK	Input	1	The DCLK signal is the reference clock for the dynamic reconfiguration port.
DEN	Input	1	The dynamic reconfiguration enable (DEN) provides the enable control signal to access the dynamic reconfiguration feature. When the dynamic reconfiguration feature is not used, DEN must be tied Low.
DI<15:0>	Input	16	The dynamic reconfiguration data input (DI) bus provides reconfiguration data. When not used, all bits must be set to zero.
DO<15:0>	Output	16	The dynamic reconfiguration output bus provides PLL data output when using dynamic reconfiguration.
DRDY	Output	1	The dynamic reconfiguration ready output (DRDY) provides the response to the DEN signal for the PLLs dynamic reconfiguration feature.
DWE	Input	1	The dynamic reconfiguration write enable (DWE) input pin provides the write enable control signal to write the DI data into the DADDR address. When not used, it must be tied Low.
LOCKED	Output	1	An output from the PLL that indicates when the PLL has achieved phase alignment within a predefined window and frequency matching within a predefined PPM range. The PLL automatically locks after power on, no extra reset is required. LOCKED will be deasserted if the input clock stops or the phase alignment is violated (that is., input clock phase shift). The PLL automatically reacquires lock after LOCKED is deasserted.
PWRDWN	Input	1	Powers down instantiated but unused PLLs.
RST	Input	1	The RST signal is an asynchronous reset for the PLL. The PLL will synchronously re-enable itself when this signal is released and go through a new phase alignment and lock cycle. A reset is required when the input clock conditions change (that is., frequency).

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
BANDWIDTH	STRING	"OPTIMIZED", "HIGH", "LOW"	"OPTIMIZED"	Specifies the PLLE2 programming algorithm affecting the jitter, phase margin and other characteristics of the PLLE2.
CLKFBOUT_MULT	DECIMAL	2 to 64	5	Specifies the amount to multiply all CLKOUT clock outputs if a different frequency is desired. This number, in combination with the associated CLKOUT#_DIVIDE value and DIVCLK_DIVIDE value, will determine the output frequency.
CLKFBOUT_PHASE	3 significant digit FLOAT	-360.000 to 360.000	0.000	Specifies the phase offset in degrees of the clock feedback output. Shifting the feedback clock results in a negative phase shift of all output clocks to the PLL.
CLKIN1_PERIOD, CLKIN2_PERIOD	FLOAT (nS)	0.000 to 52.631	0.000	Specifies the input period in ns to the PLLE2 CLKIN inputs. Resolution is down to the ps. For example a value of 33.333 would indicate a 30 MHz input clock. This information is mandatory and must be supplied. CLKIN1_PERIOD relates to the input period on the CLKIN1 input while CLKIN2_PERIOD relates to the input clock period on the CLKIN2 input.
CLKOUT0_DIVIDE, CLKOUT1_DIVIDE, CLKOUT2_DIVIDE, CLKOUT3_DIVIDE, CLKOUT4_DIVIDE, CLKOUT5_DIVIDE	DECIMAL	1 to 128	1	Specifies the amount to divide the associated CLKOUT clock output if a different frequency is desired. This number in combination with the CLKFBOUT_MULT and DIVCLK_DIVIDE values will determine the output frequency.
CLKOUT0_DUTY_CYCLE, CLKOUT1_DUTY_CYCLE, CLKOUT2_DUTY_CYCLE, CLKOUT3_DUTY_CYCLE, CLKOUT4_DUTY_CYCLE, CLKOUT5_DUTY_CYCLE	3 significant digit FLOAT	0.001 to 0.999	0.500	Specifies the Duty Cycle of the associated CLKOUT clock output in percentage (for example., 0.500 will generate a 50% duty cycle).
CLKOUT0_PHASE, CLKOUT1_PHASE, CLKOUT2_PHASE, CLKOUT3_PHASE, CLKOUT4_PHASE, CLKOUT5_PHASE	3 significant digit FLOAT	-360.000 to 360.000	0.000	Specifies the phase offset in degrees of the clock feedback output. Shifting the feedback clock results in a negative phase shift of all output clocks to the PLL.

Attribute	Type	Allowed Values	Default	Description
COMPENSATION	STRING	"ZHOLD", "BUF_IN", "EXTERNAL", "INTERNAL"	"ZHOLD"	Clock input compensation. Suggested to be set to "ZHOLD". Defines how the PLL feedback is configured. • "ZHOLD": PLL is configured to provide a negative hold time at the I/O registers. • "INTERNAL": PLL is using its own internal feedback path so no delay is being compensated. • "EXTERNAL": A network external to the FPGA is being compensated. • "BUF_IN": The configuration does not match with the other compensation modes and no delay will be compensated.
DIVCLK_DIVIDE	DECIMAL	1 to 56	1	Specifies the division ratio for all output clocks with respect to the input clock. Effectively divides the CLKIN going into the PFD.
REF_JITTER1, REF_JITTER2	3 significant digit FLOAT	0.000 to 0.999	0.010	Allows specification of the expected jitter on the CLKIN inputs to better optimize PLL performance. A bandwidth setting of OPTIMIZED will attempt to choose the best parameter for input clocking when unknown. If known, then the value provided should be specified in terms of the UI percentage (the maximum peak to peak value) of the expected jitter on the input clock. REF_JITTER1 relates to the input jitter on CLKIN1 while REF_JITTER2 relates to the input jitter on CLKIN2.
STARTUP_WAIT	STRING	"FALSE", "TRUE"	"FALSE"	When "TRUE", wait for the PLLE2(s) that have this attribute attached to them will delay DONE from going high until a LOCK is achieved.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- PLLE2_ADV: Advanced Phase Locked Loop (PLL)
--            7 Series
-- Xilinx HDL Language Template, version 2020.1

PLLE2_ADV_inst : PLLE2_ADV
generic map (
   BANDWIDTH => "OPTIMIZED",  -- OPTIMIZED, HIGH, LOW
   CLKFBOUT_MULT => 5,        -- Multiply value for all CLKOUT, (2-64)
   CLKFBOUT_PHASE => 0.0,     -- Phase offset in degrees of CLKFB, (-360.000-360.000).
   -- CLKIN_PERIOD: Input clock period in nS to ps resolution (i.e. 33.333 is 30 MHz).
   CLKIN1_PERIOD => 0.0,
   CLKIN2_PERIOD => 0.0,
   -- CLKOUT0_DIVIDE - CLKOUT5_DIVIDE: Divide amount for CLKOUT (1-128)
   CLKOUT0_DIVIDE => 1,
```

```
   CLKOUT1_DIVIDE => 1,
   CLKOUT2_DIVIDE => 1,
   CLKOUT3_DIVIDE => 1,
   CLKOUT4_DIVIDE => 1,
   CLKOUT5_DIVIDE => 1,
   -- CLKOUT0_DUTY_CYCLE - CLKOUT5_DUTY_CYCLE: Duty cycle for CLKOUT outputs (0.001-0.999).
   CLKOUT0_DUTY_CYCLE => 0.5,
   CLKOUT1_DUTY_CYCLE => 0.5,
   CLKOUT2_DUTY_CYCLE => 0.5,
   CLKOUT3_DUTY_CYCLE => 0.5,
   CLKOUT4_DUTY_CYCLE => 0.5,
   CLKOUT5_DUTY_CYCLE => 0.5,
   -- CLKOUT0_PHASE - CLKOUT5_PHASE: Phase offset for CLKOUT outputs (-360.000-360.000).
   CLKOUT0_PHASE => 0.0,
   CLKOUT1_PHASE => 0.0,
   CLKOUT2_PHASE => 0.0,
   CLKOUT3_PHASE => 0.0,
   CLKOUT4_PHASE => 0.0,
   CLKOUT5_PHASE => 0.0,
   COMPENSATION => "ZHOLD",    -- ZHOLD, BUF_IN, EXTERNAL, INTERNAL
   DIVCLK_DIVIDE => 1,         -- Master division value (1-56)
   -- REF_JITTER: Reference input jitter in UI (0.000-0.999).
   REF_JITTER1 => 0.0,
   REF_JITTER2 => 0.0,
   STARTUP_WAIT => "FALSE"     -- Delay DONE until PLL Locks, ("TRUE"/"FALSE")
)
port map (
   -- Clock Outputs: 1-bit (each) output: User configurable clock outputs
   CLKOUT0 => CLKOUT0,   -- 1-bit output: CLKOUT0
   CLKOUT1 => CLKOUT1,   -- 1-bit output: CLKOUT1
   CLKOUT2 => CLKOUT2,   -- 1-bit output: CLKOUT2
   CLKOUT3 => CLKOUT3,   -- 1-bit output: CLKOUT3
   CLKOUT4 => CLKOUT4,   -- 1-bit output: CLKOUT4
   CLKOUT5 => CLKOUT5,   -- 1-bit output: CLKOUT5
   -- DRP Ports: 16-bit (each) output: Dynamic reconfiguration ports
   DO => DO,             -- 16-bit output: DRP data
   DRDY => DRDY,         -- 1-bit output: DRP ready
   -- Feedback Clocks: 1-bit (each) output: Clock feedback ports
   CLKFBOUT => CLKFBOUT, -- 1-bit output: Feedback clock
   LOCKED => LOCKED,     -- 1-bit output: LOCK
   -- Clock Inputs: 1-bit (each) input: Clock inputs
   CLKIN1 => CLKIN1,     -- 1-bit input: Primary clock
   CLKIN2 => CLKIN2,     -- 1-bit input: Secondary clock
   -- Control Ports: 1-bit (each) input: PLL control ports
   CLKINSEL => CLKINSEL, -- 1-bit input: Clock select, High=CLKIN1 Low=CLKIN2
   PWRDWN => PWRDWN,     -- 1-bit input: Power-down
   RST => RST,           -- 1-bit input: Reset
   -- DRP Ports: 7-bit (each) input: Dynamic reconfiguration ports
   DADDR => DADDR,       -- 7-bit input: DRP address
   DCLK => DCLK,         -- 1-bit input: DRP clock
   DEN => DEN,           -- 1-bit input: DRP enable
   DI => DI,             -- 16-bit input: DRP data
   DWE => DWE,           -- 1-bit input: DRP write enable
   -- Feedback Clocks: 1-bit (each) input: Clock feedback ports
   CLKFBIN => CLKFBIN    -- 1-bit input: Feedback clock
);

-- End of PLLE2_ADV_inst instantiation
```

Verilog Instantiation Template

```
// PLLE2_ADV: Advanced Phase Locked Loop (PLL)
//            7 Series
// Xilinx HDL Language Template, version 2020.1

PLLE2_ADV #(
   .BANDWIDTH("OPTIMIZED"),  // OPTIMIZED, HIGH, LOW
   .CLKFBOUT_MULT(5),        // Multiply value for all CLKOUT, (2-64)
   .CLKFBOUT_PHASE(0.0),     // Phase offset in degrees of CLKFB, (-360.000-360.000).
   // CLKIN_PERIOD: Input clock period in nS to ps resolution (i.e. 33.333 is 30 MHz).
   .CLKIN1_PERIOD(0.0),
   .CLKIN2_PERIOD(0.0),
   // CLKOUT0_DIVIDE - CLKOUT5_DIVIDE: Divide amount for CLKOUT (1-128)
   .CLKOUT0_DIVIDE(1),
   .CLKOUT1_DIVIDE(1),
   .CLKOUT2_DIVIDE(1),
```

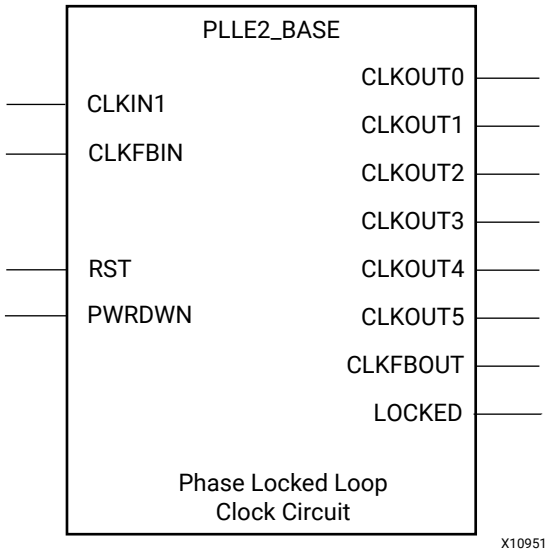
```
   .CLKOUT3_DIVIDE(1),
   .CLKOUT4_DIVIDE(1),
   .CLKOUT5_DIVIDE(1),
   // CLKOUT0_DUTY_CYCLE - CLKOUT5_DUTY_CYCLE: Duty cycle for CLKOUT outputs (0.001-0.999).
   .CLKOUT0_DUTY_CYCLE(0.5),
   .CLKOUT1_DUTY_CYCLE(0.5),
   .CLKOUT2_DUTY_CYCLE(0.5),
   .CLKOUT3_DUTY_CYCLE(0.5),
   .CLKOUT4_DUTY_CYCLE(0.5),
   .CLKOUT5_DUTY_CYCLE(0.5),
   // CLKOUT0_PHASE - CLKOUT5_PHASE: Phase offset for CLKOUT outputs (-360.000-360.000).
   .CLKOUT0_PHASE(0.0),
   .CLKOUT1_PHASE(0.0),
   .CLKOUT2_PHASE(0.0),
   .CLKOUT3_PHASE(0.0),
   .CLKOUT4_PHASE(0.0),
   .CLKOUT5_PHASE(0.0),
   .COMPENSATION("ZHOLD"),   // ZHOLD, BUF_IN, EXTERNAL, INTERNAL
   .DIVCLK_DIVIDE(1),        // Master division value (1-56)
   // REF_JITTER: Reference input jitter in UI (0.000-0.999).
   .REF_JITTER1(0.0),
   .REF_JITTER2(0.0),
   .STARTUP_WAIT("FALSE")    // Delay DONE until PLL Locks, ("TRUE"/"FALSE")
)
PLLE2_ADV_inst (
   // Clock Outputs: 1-bit (each) output: User configurable clock outputs
   .CLKOUT0(CLKOUT0),   // 1-bit output: CLKOUT0
   .CLKOUT1(CLKOUT1),   // 1-bit output: CLKOUT1
   .CLKOUT2(CLKOUT2),   // 1-bit output: CLKOUT2
   .CLKOUT3(CLKOUT3),   // 1-bit output: CLKOUT3
   .CLKOUT4(CLKOUT4),   // 1-bit output: CLKOUT4
   .CLKOUT5(CLKOUT5),   // 1-bit output: CLKOUT5
   // DRP Ports: 16-bit (each) output: Dynamic reconfiguration ports
   .DO(DO),             // 16-bit output: DRP data
   .DRDY(DRDY),         // 1-bit output: DRP ready
   // Feedback Clocks: 1-bit (each) output: Clock feedback ports
   .CLKFBOUT(CLKFBOUT), // 1-bit output: Feedback clock
   .LOCKED(LOCKED),     // 1-bit output: LOCK
   // Clock Inputs: 1-bit (each) input: Clock inputs
   .CLKIN1(CLKIN1),     // 1-bit input: Primary clock
   .CLKIN2(CLKIN2),     // 1-bit input: Secondary clock
   // Control Ports: 1-bit (each) input: PLL control ports
   .CLKINSEL(CLKINSEL), // 1-bit input: Clock select, High=CLKIN1 Low=CLKIN2
   .PWRDWN(PWRDWN),     // 1-bit input: Power-down
   .RST(RST),           // 1-bit input: Reset
   // DRP Ports: 7-bit (each) input: Dynamic reconfiguration ports
   .DADDR(DADDR),       // 7-bit input: DRP address
   .DCLK(DCLK),         // 1-bit input: DRP clock
   .DEN(DEN),           // 1-bit input: DRP enable
   .DI(DI),             // 16-bit input: DRP data
   .DWE(DWE),           // 1-bit input: DRP write enable
   // Feedback Clocks: 1-bit (each) input: Clock feedback ports
   .CLKFBIN(CLKFBIN)    // 1-bit input: Feedback clock
);

// End of PLLE2_ADV_inst instantiation
```

For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

PLLE2_BASE

Primitive: Base Phase Locked Loop (PLL)



Introduction

PLLE2 is a mixed signal block designed to support frequency synthesis, clock network deskew, and jitter reduction. The clock outputs can each have an individual divide (1 to 128), phase shift, and duty cycle based on the same VCO frequency. Output clocks are phase aligned to each other (unless phase shifted) and aligned to the input clock with a proper feedback configuration.

PLLE2 complements the MMCM element by supporting higher speed clocking while MMCM has more features to handle most general clocking needs. PLLE2_BASE is intended for most uses of this PLL component while PLLE2_ADV is intended for use when clock switch-over or dynamic reconfiguration is required.

Port Descriptions

Port	DIrection	Width	Function
CLKFBIN	Input	1	Feedback clock pin to the PLL.
CLKFBOUT	Output	1	Dedicated PLL Feedback clock output.
CLKIN1	Input	1	General clock input.
CLKOUT0	Output	1	Configurable clock output CLKOUT0.
CLKOUT1	Output	1	Configurable clock output CLKOUT1.
CLKOUT2	Output	1	Configurable clock output CLKOUT2.
CLKOUT3	Output	1	Configurable clock output CLKOUT3.
CLKOUT4	Output	1	Configurable clock output CLKOUT4.

Port	DIrection	Width	Function
CLKOUT5	Output	1	Configurable clock output CLKOUT5.
LOCKED	Output	1	An output from the PLL that indicates when the PLL has achieved phase alignment within a predefined window and frequency matching within a predefined PPM range. The PLL automatically locks after power on, no extra reset is required. LOCKED will be deasserted if the input clock stops or the phase alignment is violated (e.g., input clock phase shift). The PLL automatically reacquires lock after LOCKED is deasserted.
PWRDWN	Input	1	Powers down instantiated but unused PLLs.
RST	Input	1	The RST signal is an asynchronous reset for the PLL. The PLL will synchronously re-enable itself when this signal is released and go through a new phase alignment and lock cycle. A reset is required when the input clock conditions change (e.g., frequency).

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Yes
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
BANDWIDTH	STRING	"OPTIMIZED", "HIGH", "LOW"	"OPTIMIZED"	Specifies the PLLE2 programming algorithm affecting the jitter, phase margin and other characteristics of the PLLE2.
CLKFBOUT_MULT	DECIMAL	2 to 64	5	Specifies the amount to multiply all CLKOUT clock outputs if a different frequency is desired. This number, in combination with the associated CLKOUT#_DIVIDE value and DIVCLK_DIVIDE value, will determine the output frequency.
CLKFBOUT_PHASE	3 significant digit FLOAT	-360.000 to 360.000	0.000	Specifies the phase offset in degrees of the clock feedback output. Shifting the feedback clock results in a negative phase shift of all output clocks to the PLL.
CLKIN1_PERIOD	FLOAT (nS)	0.000 to 52.631	0.000	Specifies the input period in ns to the PLL CLKIN1 input. Resolution is down to the ps (3 decimal places). For example a value of 33.333 would indicate a 30 MHz input clock. This information is mandatory and must be supplied.
CLKOUT0_DIVIDE, CLKOUT1_DIVIDE, CLKOUT2_DIVIDE, CLKOUT3_DIVIDE, CLKOUT4_DIVIDE, CLKOUT5_DIVIDE	DECIMAL	1 to 128	1	Specifies the amount to divide the associated CLKOUT clock output if a different frequency is desired. This number in combination with the CLKFBOUT_MULT and DIVCLK_DIVIDE values will determine the output frequency.

Attribute	Type	Allowed Values	Default	Description
CLKOUT0_DUTY_CYCLE, CLKOUT1_DUTY_CYCLE, CLKOUT2_DUTY_CYCLE, CLKOUT3_DUTY_CYCLE, CLKOUT4_DUTY_CYCLE, CLKOUT5_DUTY_CYCLE	3 significant digit FLOAT	0.001 to 0.999	0.500	Specifies the Duty Cycle of the associated CLKOUT clock output in percentage (i.e., 0.500 will generate a 50% duty cycle).
CLKOUT0_PHASE, CLKOUT1_PHASE, CLKOUT2_PHASE, CLKOUT3_PHASE, CLKOUT4_PHASE, CLKOUT5_PHASE	3 significant digit FLOAT	-360.000 to 360.000	0.000	Specifies the phase offset in degrees of the clock feedback output. Shifting the feedback clock results in a negative phase shift of all output clocks to the PLL.
DIVCLK_DIVIDE	DECIMAL	1 to 56	1	Specifies the division ratio for all output clocks with respect to the input clock. Effectively divides the CLKIN going into the PFD.
REF_JITTER1	3 significant digit FLOAT	0.000 to 0.999	0.010	Allows specification of the expected jitter on CLKIN1 in order to better optimize PLL performance. A bandwidth setting of OPTIMIZED will attempt to choose the best parameter for input clocking when unknown. If known, then the value provided should be specified in terms of the UI percentage (the maximum peak to peak value) of the expected jitter on the input clock.
STARTUP_WAIT	STRING	"FALSE", "TRUE"	"FALSE"	When "TRUE", wait for the PLLE2(s) that have this attribute attached to them will delay DONE from going high until a LOCK is achieved.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- PLLE2_BASE: Base Phase Locked Loop (PLL)
--             7 Series
-- Xilinx HDL Language Template, version 2020.1

PLLE2_BASE_inst : PLLE2_BASE
generic map (
   BANDWIDTH => "OPTIMIZED",  -- OPTIMIZED, HIGH, LOW
   CLKFBOUT_MULT => 5,        -- Multiply value for all CLKOUT, (2-64)
   CLKFBOUT_PHASE => 0.0,     -- Phase offset in degrees of CLKFB, (-360.000-360.000).
   CLKIN1_PERIOD => 0.0,      -- Input clock period in ns to ps resolution (i.e. 33.333 is 30 MHz).
   -- CLKOUT0_DIVIDE - CLKOUT5_DIVIDE: Divide amount for each CLKOUT (1-128)
   CLKOUT0_DIVIDE => 1,
   CLKOUT1_DIVIDE => 1,
   CLKOUT2_DIVIDE => 1,
   CLKOUT3_DIVIDE => 1,
   CLKOUT4_DIVIDE => 1,
   CLKOUT5_DIVIDE => 1,
```

```
   -- CLKOUT0_DUTY_CYCLE - CLKOUT5_DUTY_CYCLE: Duty cycle for each CLKOUT (0.001-0.999).
   CLKOUT0_DUTY_CYCLE => 0.5,
   CLKOUT1_DUTY_CYCLE => 0.5,
   CLKOUT2_DUTY_CYCLE => 0.5,
   CLKOUT3_DUTY_CYCLE => 0.5,
   CLKOUT4_DUTY_CYCLE => 0.5,
   CLKOUT5_DUTY_CYCLE => 0.5,
   -- CLKOUT0_PHASE - CLKOUT5_PHASE: Phase offset for each CLKOUT (-360.000-360.000).
   CLKOUT0_PHASE => 0.0,
   CLKOUT1_PHASE => 0.0,
   CLKOUT2_PHASE => 0.0,
   CLKOUT3_PHASE => 0.0,
   CLKOUT4_PHASE => 0.0,
   CLKOUT5_PHASE => 0.0,
   DIVCLK_DIVIDE => 1,         -- Master division value, (1-56)
   REF_JITTER1 => 0.0,         -- Reference input jitter in UI, (0.000-0.999).
   STARTUP_WAIT => "FALSE"     -- Delay DONE until PLL Locks, ("TRUE"/"FALSE")
)
port map (
   -- Clock Outputs: 1-bit (each) output: User configurable clock outputs
   CLKOUT0 => CLKOUT0,   -- 1-bit output: CLKOUT0
   CLKOUT1 => CLKOUT1,   -- 1-bit output: CLKOUT1
   CLKOUT2 => CLKOUT2,   -- 1-bit output: CLKOUT2
   CLKOUT3 => CLKOUT3,   -- 1-bit output: CLKOUT3
   CLKOUT4 => CLKOUT4,   -- 1-bit output: CLKOUT4
   CLKOUT5 => CLKOUT5,   -- 1-bit output: CLKOUT5
   -- Feedback Clocks: 1-bit (each) output: Clock feedback ports
   CLKFBOUT => CLKFBOUT, -- 1-bit output: Feedback clock
   LOCKED => LOCKED,     -- 1-bit output: LOCK
   CLKIN1 => CLKIN1,     -- 1-bit input: Input clock
   -- Control Ports: 1-bit (each) input: PLL control ports
   PWRDWN => PWRDWN,     -- 1-bit input: Power-down
   RST => RST,           -- 1-bit input: Reset
   -- Feedback Clocks: 1-bit (each) input: Clock feedback ports
   CLKFBIN => CLKFBIN    -- 1-bit input: Feedback clock
);

-- End of PLLE2_BASE_inst instantiation
```

Verilog Instantiation Template

```
// PLLE2_BASE: Base Phase Locked Loop (PLL)
//             7 Series
// Xilinx HDL Language Template, version 2020.1

PLLE2_BASE #(
   .BANDWIDTH("OPTIMIZED"),  // OPTIMIZED, HIGH, LOW
   .CLKFBOUT_MULT(5),        // Multiply value for all CLKOUT, (2-64)
   .CLKFBOUT_PHASE(0.0),     // Phase offset in degrees of CLKFB, (-360.000-360.000).
   .CLKIN1_PERIOD(0.0),      // Input clock period in ns to ps resolution (i.e. 33.333 is 30 MHz).
   // CLKOUT0_DIVIDE - CLKOUT5_DIVIDE: Divide amount for each CLKOUT (1-128)
   .CLKOUT0_DIVIDE(1),
   .CLKOUT1_DIVIDE(1),
   .CLKOUT2_DIVIDE(1),
   .CLKOUT3_DIVIDE(1),
   .CLKOUT4_DIVIDE(1),
   .CLKOUT5_DIVIDE(1),
   // CLKOUT0_DUTY_CYCLE - CLKOUT5_DUTY_CYCLE: Duty cycle for each CLKOUT (0.001-0.999).
   .CLKOUT0_DUTY_CYCLE(0.5),
   .CLKOUT1_DUTY_CYCLE(0.5),
   .CLKOUT2_DUTY_CYCLE(0.5),
   .CLKOUT3_DUTY_CYCLE(0.5),
   .CLKOUT4_DUTY_CYCLE(0.5),
   .CLKOUT5_DUTY_CYCLE(0.5),
   // CLKOUT0_PHASE - CLKOUT5_PHASE: Phase offset for each CLKOUT (-360.000-360.000).
   .CLKOUT0_PHASE(0.0),
   .CLKOUT1_PHASE(0.0),
   .CLKOUT2_PHASE(0.0),
   .CLKOUT3_PHASE(0.0),
   .CLKOUT4_PHASE(0.0),
   .CLKOUT5_PHASE(0.0),
   .DIVCLK_DIVIDE(1),        // Master division value, (1-56)
   .REF_JITTER1(0.0),        // Reference input jitter in UI, (0.000-0.999).
   .STARTUP_WAIT("FALSE")    // Delay DONE until PLL Locks, ("TRUE"/"FALSE")
)
PLLE2_BASE_inst (
```

```
   // Clock Outputs: 1-bit (each) output: User configurable clock outputs
   .CLKOUT0(CLKOUT0),   // 1-bit output: CLKOUT0
   .CLKOUT1(CLKOUT1),   // 1-bit output: CLKOUT1
   .CLKOUT2(CLKOUT2),   // 1-bit output: CLKOUT2
   .CLKOUT3(CLKOUT3),   // 1-bit output: CLKOUT3
   .CLKOUT4(CLKOUT4),   // 1-bit output: CLKOUT4
   .CLKOUT5(CLKOUT5),   // 1-bit output: CLKOUT5
   // Feedback Clocks: 1-bit (each) output: Clock feedback ports
   .CLKFBOUT(CLKFBOUT), // 1-bit output: Feedback clock
   .LOCKED(LOCKED),     // 1-bit output: LOCK
   .CLKIN1(CLKIN1),     // 1-bit input: Input clock
   // Control Ports: 1-bit (each) input: PLL control ports
   .PWRDWN(PWRDWN),     // 1-bit input: Power-down
   .RST(RST),           // 1-bit input: Reset
   // Feedback Clocks: 1-bit (each) input: Clock feedback ports
   .CLKFBIN(CLKFBIN)    // 1-bit input: Feedback clock
);

// End of PLLE2_BASE_inst instantiation
```

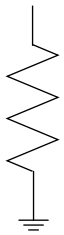
For More Information

- See the *7 Series FPGAs Clocking Resource User Guide* (UG472).

PULLDOWN

Primitive: Resistor to GND for Input Pads, Open-Drain, and 3-State Outputs

PULLDOWN



X10690

Introduction

This resistor element is connected to input, output, or bidirectional pads to guarantee a logic Low level for nodes that might float.

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Pulldown output (connect directly to top level port)

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- PULLDOWN: I/O Buffer Weak Pull-down
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

PULLDOWN_inst : PULLDOWN
```

```
port map (
   O => O     -- Pulldown output (connect directly to top-level port)
);

-- End of PULLDOWN_inst instantiation
```

Verilog Instantiation Template

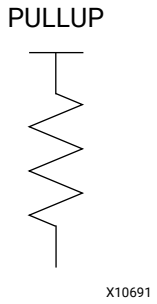
```
// PULLDOWN: I/O Buffer Weak Pull-down
//           7 Series
// Xilinx HDL Language Template, version 2020.1

PULLDOWN PULLDOWN_inst (
   .O(O)     // Pulldown output (connect directly to top-level port)
);

// End of PULLDOWN_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

PULLUP

Primitive: Resistor to VCC for Input PADs, Open-Drain, and 3-State Outputs



Introduction

This design element allows for an input, 3-state output, or bi-directional port to be driven to a weak high value when not being driven by an internal or external source. This element establishes a High logic level for open-drain elements and macros when all the drivers are off.

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Pullup output (connect directly to top level port)

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- PULLUP: I/O Buffer Weak Pull-up
--         7 Series
-- Xilinx HDL Language Template, version 2020.1

PULLUP_inst : PULLUP
```

```
port map (
   O => O     -- Pullup output (connect directly to top-level port)
);

-- End of PULLUP_inst instantiation
```

Verilog Instantiation Template

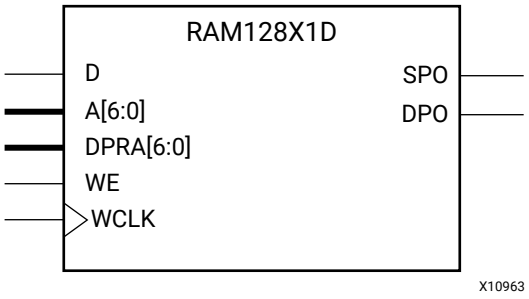
```
// PULLUP: I/O Buffer Weak Pull-up
//         7 Series
// Xilinx HDL Language Template, version 2020.1

PULLUP PULLUP_inst (
   .O(O)     // Pullup output (connect directly to top-level port)
);

// End of PULLUP_inst instantiation
```

For More Information

- See the *7 Series FPGA SelectIO Resources User Guide* (UG471).

RAM128X1D

Primitive: 128-Deep by 1-Wide Dual Port Random Access Memory (Select RAM)



Introduction

This design element is a 128-bit deep by 1-bit wide random access memory and has a read/write port that writes the value on the D input data pin when the write enable (WE) is high to the memory cell specified by the A address bus. This happens shortly after the rising edge of the WCLK and that same value is reflected in the data output SPO. When WE is low, an asynchronous read is initiated in which the contents of the memory cell specified by the A address bus is output asynchronously to the SPO output. The read port can perform asynchronous read access of the memory by changing the value of the address bus DPRA, and by outputting that value to the DPO data output.

Port Descriptions

Port	Direction	Width	Function
SPO	Output	1	Read/Write port data output addressed by A.
DPO	Output	1	Read port data output addressed by DPRA.
D	Input	1	Write data input addressed by A.
A	Input	7	Read/Write port address bus.
DPRA	Input	7	Read port address bus.
WE	Input	1	Write Enable.
WCLK	Input	1	Write clock (reads are asynchronous).

If instantiated, the following connections should be made to this component:

- Tie the WCLK input to the desired clock source, the D input to the data source to be stored and the DPO output to an FDCE D input or other appropriate data destination.
- Optionally, the SPO output can also be connected to the appropriate data destination or else left unconnected.
- Connect the WE clock enable pin to the proper write enable source in the design.

- Connect the 7-bit A bus to the source for the read/write addressing and the 7-bit DPRA bus to the appropriate read address connections.

You can use the INIT attribute to specify the initial contents of the RAM. If left unspecified, the initial contents default to all zeros.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 128-bit value	All zeros	Specifies the initial contents of the RAM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAM128X1D: 128-deep by 1-wide positive edge write, asynchronous read
--            dual-port distributed LUT RAM (Mapped to two SliceM LUT6s)
--            7 Series
-- Xilinx HDL Language Template, version 2020.1

RAM128X1D_inst : RAM128X1D
generic map (
   INIT => X"00000000000000000000000000000000")
port map (
   DPO => DPO,      -- Read/Write port 1-bit ouput
   SPO => SPO,      -- Read port 1-bit output
   A => A,          -- Read/Write port 7-bit address input
   D => D,          -- RAM data input
   DPRA => DPRA,    -- Read port 7-bit address input
   WCLK => WCLK,    -- Write clock input
   WE => WE         -- RAM data input
);

-- End of RAM128X1D_inst instantiation
```

Verilog Instantiation Template

```
// RAM128X1D: 128-deep by 1-wide positive edge write, asynchronous read  (Mapped to two SliceM LUT6s)
//            dual-port distributed LUT RAM
//            7 Series
// Xilinx HDL Language Template, version 2020.1

RAM128X1D #(
   .INIT(128'h00000000000000000000000000000000)
) RAM128X1D_inst (
   .DPO(DPO),   // Read port 1-bit output
```

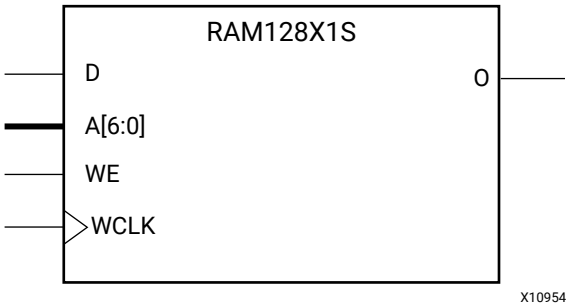
```
   .SPO(SPO),   // Read/write port 1-bit output
   .A(A),       // Read/write port 7-bit address input
   .D(D),       // RAM data input
   .DPRA(DPRA), // Read port 7-bit address input
   .WCLK(WCLK), // Write clock input
   .WE(WE)      // Write enable input
);

// End of RAM128X1D_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

RAM128X1S

Primitive: 128-Deep by 1-Wide Random Access Memory (Select RAM)



Introduction

This design element is a 128-bit deep by 1-bit wide random access memory with synchronous write and asynchronous read capability. This RAM is implemented using the LUT resources of the device (also known as Select RAM), and does not consume any of the block RAM resources of the device. If a synchronous read capability is preferred, a register can be attached to the output and placed in the same slice as long as the same clock is used for both the RAM and the register. The RAM128X1S has an active-High write enable, WE, so that when that signal is High, and a rising edge occurs on the WCLK pin, a write is performed recording the value of the D input data pin into the memory array. The output O displays the contents of the memory cell addressed by A, regardless of the WE value. When a write is performed, the output is updated to the new value shortly after the write completes.

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Read/Write port data output addressed by A.
D	Input	1	Write data input addressed by A.
A	Input	7	Read/Write port address bus.
WE	Input	1	Write Enable.
WCLK	Input	1	Write clock (reads are asynchronous).

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

If instantiated, the following connections should be made to this component:

- Tie the WCLK input to the desired clock source, the D input to the data source to be stored, and the O output to an FDCE D input or other appropriate data destination.
- Connect the WE clock enable pin to the proper write enable source in the design.
- Connect the 7-bit A bus to the source for the read/write.

You can use the INIT attribute to specify the initial contents of the RAM. If left unspecified, the initial contents default to all zeros.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 128-bit value	All zeros	Specifies the initial contents of the RAM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAM128X1S: 128-deep x 1 positive edge write, asynchronous read
--            single-port distributed RAM (Mapped to SliceM LUT6)
--            7 Series
-- Xilinx HDL Language Template, version 2020.1

RAM128X1S_inst : RAM128X1S
generic map (
   INIT => X"00000000000000000000000000000000")
port map (
   O => O,      -- 1-bit data output
   A0 => A0,    -- Address[0] input bit
   A1 => A1,    -- Address[1] input bit
   A2 => A2,    -- Address[2] input bit
   A3 => A3,    -- Address[3] input bit
   A4 => A4,    -- Address[4] input bit
   A5 => A5,    -- Address[5] input bit
   A6 => A6,    -- Address[6] input bit
   D => D,          -- 1-bit data input
   WCLK => WCLK,    -- Write clock input
   WE => WE         -- RAM data input
);

-- End of RAM128X1S_inst instantiation
```

Verilog Instantiation Template

```
// RAM128X1S: 128 x 1 positive edge write, asynchronous read single-port
//            distributed RAM (Mapped to two SliceM LUT6s)
//            7 Series
// Xilinx HDL Language Template, version 2020.1

RAM128X1S #(
   .INIT(128'h00000000000000000000000000000000) // Initial contents of RAM
) RAM128X1S_inst (
   .O(O),         // 1-bit data output
```

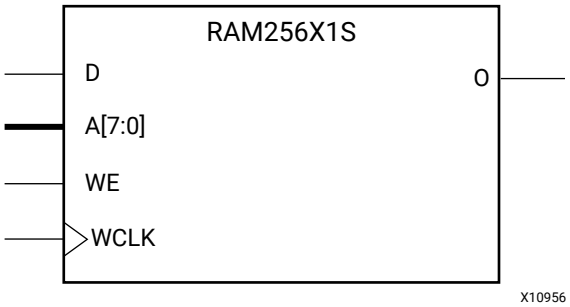
```
   .A0(A0),      // Address[0] input bit
   .A1(A1),      // Address[1] input bit
   .A2(A2),      // Address[2] input bit
   .A3(A3),      // Address[3] input bit
   .A4(A4),      // Address[4] input bit
   .A5(A5),      // Address[5] input bit
   .A6(A6),      // Address[6] input bit
   .D(D),        // 1-bit data input
   .WCLK(WCLK),  // Write clock input
   .WE(WE)       // Write enable input
);

// End of RAM128X1S_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

RAM256X1S

Primitive: 256-Deep by 1-Wide Random Access Memory (Select RAM)



Introduction

This design element is a 256-bit deep by 1-bit wide random access memory with synchronous write and asynchronous read capability. This RAM is implemented using the LUT resources of the device (also known as Select RAM), and does not consume any of the block RAM resources of the device. If a synchronous read capability is preferred, a register can be attached to the output and placed in the same slice as long as the same clock is used for both the RAM and the register. The RAM256X1S has an active-High write enable, WE, so that when that signal is High, and a rising edge occurs on the WCLK pin, a write is performed recording the value of the D input data pin into the memory array. The output O displays the contents of the memory cell addressed by A, regardless of the WE value. When a write is performed, the output is updated to the new value shortly after the write completes.

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Read/Write port data output addressed by A.
D	Input	1	Write data input addressed by A.
A	Input	8	Read/Write port address bus.
WE	Input	1	Write Enable.
WCLK	Input	1	Write clock (reads are asynchronous).

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

If instantiated, the following connections should be made to this component:

- Tie the WCLK input to the desired clock source, the D input to the data source to be stored, and the O output to an FDCE D input or other appropriate data destination.
- Connect the WE clock enable pin to the proper write enable source in the design.
- Connect the 8-bit A bus to the source for the read/write.

You can use the INIT attribute to specify the initial contents of the RAM. If left unspecified, the initial contents default to all zeros.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 256-bit value	All zeros	Specifies the initial contents of the RAM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAM256X1S: 256-deep by 1-wide positive edge write, asynchronous read
--            single-port distributed LUT RAM (Mapped to four SliceM LUT6s)
--            7 Series
-- Xilinx HDL Language Template, version 2020.1

RAM256X1S_inst : RAM256X1S
generic map (
   INIT => X"0000000000000000000000000000000000000000000000000000000000000000")
port map (
   O => O,  -- Read/Write port 1-bit ouput
   A => A,  -- Read/Write port 8-bit address input
   D => D,  -- RAM data input
   WCLK => WCLK,  -- Write clock input
   WE => WE -- Write enable input
);

-- End of RAM256X1S_inst instantiation
```

Verilog Instantiation Template

```
// RAM256X1S: 256-deep by 1-wide positive edge write, asynchronous read  (Mapped to four SliceM LUT6s)
//            single-port distributed LUT RAM
//            7 Series
// Xilinx HDL Language Template, version 2020.1

RAM256X1S #(
   .INIT(256'h0000000000000000000000000000000000000000000000000000000000000000)
) RAM256X1S_inst (
   .O(O),       // Read/write port 1-bit output
   .A(A),       // Read/write port 8-bit address input
   .WE(WE),     // Write enable input
```

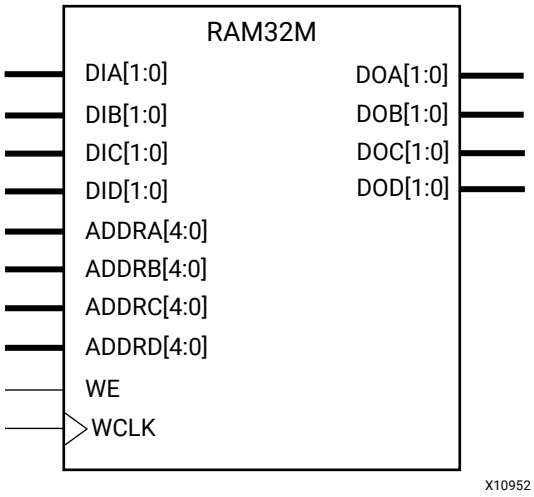
```
   .WCLK(WCLK), // Write clock input
   .D(D)        // RAM data input
);

// End of RAM256X1S_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

RAM32M

Primitive: 32-Deep by 8-bit Wide Multi Port Random Access Memory (Select RAM)



Introduction

This design element is a 32-bit deep by 8-bit wide, multi-port, random access memory with synchronous write and asynchronous independent, 2-bit, wide-read capability. This RAM is implemented using the LUT resources of the device known as SelectRAM™+, and does not consume any of the Block RAM resources of the device. The RAM32M is implemented in a single slice and consists of one 8-bit write, 2-bit read port and three separate 2-bit read ports from the same memory, which allows for byte-wide write and independent 2-bit read access RAM.

- If the DIA, DIB, DIC, and DID inputs are all tied to the same data inputs, the RAM can become a 1 read/write port, 3 independent read port, 32x2 quad port memory.
- If DID is grounded, DOD is not used.
- If ADDRA, ADDRB, and ADDRC are tied to the same address, the RAM becomes a 32x6 simple dual port RAM.
- If ADDRD is tied to ADDRA, ADDRB, and ADDRC, then the RAM is a 32x8 single port RAM.

There are several other possible configurations for this RAM.

Port Descriptions

Port	Direction	Width	Function
DOA	Output	2	Read port data outputs addressed by ADDRA.
DOB	Output	2	Read port data outputs addressed by ADDRB.
DOC	Output	2	Read port data outputs addressed by ADDRC.

Port	Direction	Width	Function
DOD	Output	2	Read/Write port data outputs addressed by ADDRD.
DIA	Input	2	Write data inputs addressed by ADDRD (read output is addressed by ADDRA).
DIB	Input	2	Write data inputs addressed by ADDRD (read output is addressed by ADDRB).
DIC	Input	2	Write data inputs addressed by ADDRD (read output is addressed by ADDRC).
DID	Input	2	Write data inputs addressed by ADDRD.
ADDRA	Input	5	Read address bus A.
ADDRB	Input	5	Read address bus B.
ADDRC	Input	5	Read address bus C.
ADDRD	Input	5	8-bit data write port, 2-bit data read port address bus D.
WE	Input	1	Write Enable.
WCLK	Input	1	Write clock (reads are asynchronous).

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

This element can be inferred by some synthesis tools by describing a RAM with a synchronous write and asynchronous read capability. Consult your synthesis tool documentation for details on RAM inference capabilities and coding examples. You should instantiate this component if you have a need to implicitly specify the RAM function, or if you need to manually place or relationally place the component. If a synchronous read capability is desired, the outputs can be connected to an FDRSE (FDCPE if asynchronous reset is needed) in order to improve the output timing of the function. However, this is not necessary for the proper operation of the RAM. If you want to have the data clocked on the negative edge of a clock, an inverter can be described on the clock input to this component. This inverter will be absorbed into the block giving the ability to write to the RAM on falling clock edges.

If instantiated, the following connections should be made to this component:

- Connect the WCLK input to the desired clock source
- Connect the DIA, DIB, DIC, and DID inputs to the data source to be stored
- Connect the DOA, DOB, DOC, and DOD outputs to an FDCE D input or other appropriate data destination, or leave unconnected if not used
- Connect the WE clock enable pin to the proper write enable source in the design
- Connect the ADDRD bus to the source for the read/write addressing

- Connect the ADDRA, ADDRB, and ADDRC buses to the appropriate read address connections

The optional INIT_A, INIT_B, INIT_C and INIT_D attributes let you specify the initial memory contents of each port using a 64-bit hexadecimal value. The INIT value correlates to the RAM addressing by the following equation: ADDRy[z] = INIT_y[2*z+1:2*z]. For instance, if the RAM ADDRC port is addressed to 00001, then the INIT_C[3:2] values would be the initial values shown on the DOC port before the first write occurs at that address. If left unspecified, the initial contents will be all zeros.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT_A	HEX	Any 64-bit value	All zeros	Specifies the initial contents of the RAM on port A.
INIT_B	HEX	Any 64-bit value	All zeros	Specifies the initial contents of the RAM on port B.
INIT_C	HEX	Any 64-bit value	All zeros	Specifies the initial contents of the RAM on port C.
INIT_D	HEX	Any 64-bit value	All zeros	Specifies the initial contents of the RAM on port D.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAM32M: 32-deep by 8-wide Multi Port LUT RAM (Mapped to four SliceM LUT6s)
--         7 Series
-- Xilinx HDL Language Template, version 2020.1

RAM32M_inst : RAM32M
generic map (
   INIT_A => X"0000000000000000",   -- Initial contents of A port
   INIT_B => X"0000000000000000",   -- Initial contents of B port
   INIT_C => X"0000000000000000",   -- Initial contents of C port
   INIT_D => X"0000000000000000")   -- Initial contents of D port
port map (
   DOA => DOA, -- Read port A 2-bit output
   DOB => DOB, -- Read port B 2-bit output
   DOC => DOC, -- Read port C 2-bit output
   DOD => DOD, -- Read/Write port D 2-bit output
   ADDRA => ADDRA,   -- Read port A 5-bit address input
   ADDRB => ADDRB,   -- Read port B 5-bit address input
   ADDRC => ADDRC,   -- Read port C 5-bit address input
   ADDRD => ADDRD,   -- Read/Write port D 5-bit address input
   DIA => DIA, -- RAM 2-bit data write input addressed by ADDRD,
               -- read addressed by ADDRA
   DIB => DIB, -- RAM 2-bit data write input addressed by ADDRD,
               -- read addressed by ADDRB
   DIC => DIC, -- RAM 2-bit data write input addressed by ADDRD,
               -- read addressed by ADDRC
   DID => DID, -- RAM 2-bit data write input addressed by ADDRD,
               -- read addressed by ADDRD
   WCLK => WCLK,  -- Write clock input
   WE => WE       -- Write enable input
);
-- End of RAM32M_inst instantiation
```

Verilog Instantiation Template

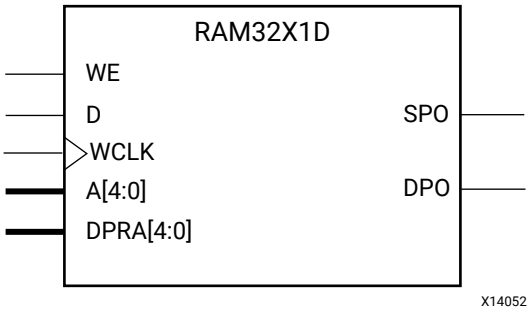
```
// RAM32M: 32-deep by 8-wide Multi Port LUT RAM (Mapped to four SliceM LUT6s)
//         7 Series
// Xilinx HDL Language Template, version 2020.1

RAM32M #(
   .INIT_A(64'h0000000000000000), // Initial contents of A Port
   .INIT_B(64'h0000000000000000), // Initial contents of B Port
   .INIT_C(64'h0000000000000000), // Initial contents of C Port
   .INIT_D(64'h0000000000000000)  // Initial contents of D Port
) RAM32M_inst (
   .DOA(DOA),     // Read port A 2-bit output
   .DOB(DOB),     // Read port B 2-bit output
   .DOC(DOC),     // Read port C 2-bit output
   .DOD(DOD),     // Read/write port D 2-bit output
   .ADDRA(ADDRA), // Read port A 5-bit address input
   .ADDRB(ADDRB), // Read port B 5-bit address input
   .ADDRC(ADDRC), // Read port C 5-bit address input
   .ADDRD(ADDRD), // Read/write port D 5-bit address input
   .DIA(DIA),     // RAM 2-bit data write input addressed by ADDRD,
                  //   read addressed by ADDRA
   .DIB(DIB),     // RAM 2-bit data write input addressed by ADDRD,
                  //   read addressed by ADDRB
   .DIC(DIC),     // RAM 2-bit data write input addressed by ADDRD,
                  //   read addressed by ADDRC
   .DID(DID),     // RAM 2-bit data write input addressed by ADDRD,
                  //   read addressed by ADDRD
   .WCLK(WCLK),   // Write clock input
   .WE(WE)        // Write enable input
);

// End of RAM32M_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

RAM32X1D

Primitive: 32-Deep by 1-Wide Static Dual Port Synchronous RAM



Introduction

This design element is a 32-bit deep by 1-bit wide static dual port random access memory with synchronous write capability. The device has two separate address ports: the read address (DPRA4:DPRA0) and the write address (A4:A0). These two address ports are completely asynchronous. The read address controls the location of the data driven out of the output pin (DPO), and the write address controls the destination of a valid write transaction. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is High, any positive transition on WCLK loads the data on the data input (D) into the memory cell selected by the 5-bit write address. For predictable performance, write address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block. You can initialize RAM32X1D during configuration using the INIT attribute. Mode selection is shown in the following logic table.

The SPO output reflects the data in the memory cell addressed by A4:A0. The DPO output reflects the data in the memory cell addressed by DPRA4:DPRA0. The write process is not affected by the address on the read address port.

You can use the INIT attribute to specify the initial contents of the RAM. If left unspecified, the initial contents default to all zeros.

Logic Table

Inputs			Outputs	
WE (Mode)	**WCLK**	**D**	**SPO**	**DPO**
0 (read)	X	X	data_a	data_d
1 (read)	0	X	data_a	data_d
1 (read)	1	X	data_a	data_d

Inputs			Outputs	
WE (Mode)	**WCLK**	**D**	**SPO**	**DPO**
1 (write)	↑	D	D	data_d
1 (read)	↓	X	data_a	data_d

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 32-bit value	All zeros	Specifies the initial contents of the RAM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAM32X1D: 32 x 1 positive edge write, asynchronous read
--              dual-port distributed RAM (Mapped to SliceM LUT6)
--              7 Series
-- Xilinx HDL Language Template, version 2020.1

RAM32X1D_inst : RAM32X1D
generic map (
   INIT => X"00000000") -- Initial contents of RAM
port map (
   DPO => DPO,      -- Read-only 1-bit data output
   SPO => SPO,      -- R/W 1-bit data output
   A0 => A0,        -- R/W address[0] input bit
   A1 => A1,        -- R/W address[1] input bit
   A2 => A2,        -- R/W address[2] input bit
   A3 => A3,        -- R/W address[3] input bit
   A4 => A4,        -- R/W address[4] input bit
   D => D,          -- Write 1-bit data input
   DPRA0 => DPRA0, -- Read-only address[0] input bit
   DPRA1 => DPRA1, -- Read-only address[1] input bit
   DPRA2 => DPRA2, -- Read-only address[2] input bit
   DPRA3 => DPRA3, -- Read-only address[3] input bit
   DPRA4 => DPRA4, -- Read-only address[4] input bit
   WCLK => WCLK,    -- Write clock input
   WE => WE         -- Write enable input
);

-- End of RAM32X1D_inst instantiation
```

Verilog Instantiation Template

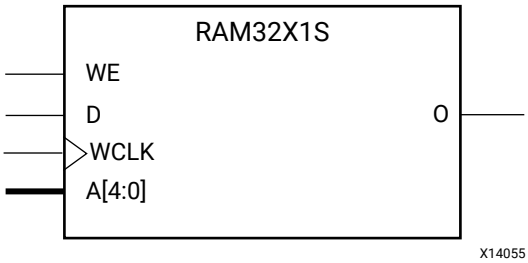
```
// RAM32X1D: 32 x 1 positive edge write, asynchronous read dual-port
//           distributed RAM (Mapped to a SliceM LUT6)
//           7 Series
// Xilinx HDL Language Template, version 2020.1

RAM32X1D #(
   .INIT(32'h00000000) // Initial contents of RAM
) RAM32X1D_inst (
   .DPO(DPO),     // Read-only 1-bit data output
   .SPO(SPO),     // Rw/ 1-bit data output
   .A0(A0),       // Rw/ address[0] input bit
   .A1(A1),       // Rw/ address[1] input bit
   .A2(A2),       // Rw/ address[2] input bit
   .A3(A3),       // Rw/ address[3] input bit
   .A4(A4),       // Rw/ address[4] input bit
   .D(D),         // Write 1-bit data input
   .DPRA0(DPRA0), // Read-only address[0] input bit
   .DPRA1(DPRA1), // Read-only address[1] input bit
   .DPRA2(DPRA2), // Read-only address[2] input bit
   .DPRA3(DPRA3), // Read-only address[3] input bit
   .DPRA4(DPRA4), // Read-only address[4] input bit
   .WCLK(WCLK),   // Write clock input
   .WE(WE)        // Write enable input
);

// End of RAM32X1D_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

RAM32X1S

Primitive: 32-Deep by 1-Wide Static Synchronous RAM



Introduction

This design element is a 32-bit deep by 1-bit wide static random access memory with synchronous write capability. When the write enable is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When (WE) is High, any positive transition on (WCLK) loads the data on the data input (D) into the memory cell selected by the 5-bit address (A4-A0). For predictable performance, address and data inputs must be stable before a Low-to-High (WCLK) transition. This RAM block assumes an active-High (WCLK). However, (WCLK) can be active-High or active-Low. Any inverter placed on the (WCLK) input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can use the INIT attribute to specify the initial contents of the RAM. If left unspecified, the initial contents default to all zeros.

Logic Table

Inputs			Outputs
WE (Mode)	**WCLK**	**D**	**O**
0 (read)	X	X	Data
1 (read)	0	X	Data
1 (read)	1	X	Data
1 (write)	↑	D	D
1 (read)	↓	X	Data

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 32-bit value	All zeros	Specifies initial contents of the RAM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAM32X1S: 32 x 1 posedge write distributed (LUT) RAM (Mapped to SliceM LUT6)
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

RAM32X1S_inst : RAM32X1S
generic map (
   INIT => X"00000000")
port map (
   O => O,        -- RAM output
   A0 => A0,      -- RAM address[0] input
   A1 => A1,      -- RAM address[1] input
   A2 => A2,      -- RAM address[2] input
   A3 => A3,      -- RAM address[3] input
   A4 => A4,      -- RAM address[4] input
   D => D,        -- RAM data input
   WCLK => WCLK,  -- Write clock input
   WE => WE       -- Write enable input
);

-- End of RAM32X1S_inst instantiation
```

Verilog Instantiation Template

```
// RAM32X1S: 32 x 1 posedge write distributed (LUT) RAM (Mapped to a SliceM LUT6)
//           7 Series
// Xilinx HDL Language Template, version 2020.1

RAM32X1S #(
   .INIT(32'h00000000)  // Initial contents of RAM
) RAM32X1S_inst (
   .O(O),       // RAM output
   .A0(A0),     // RAM address[0] input
   .A1(A1),     // RAM address[1] input
   .A2(A2),     // RAM address[2] input
   .A3(A3),     // RAM address[3] input
   .A4(A4),     // RAM address[4] input
   .D(D),       // RAM data input
```

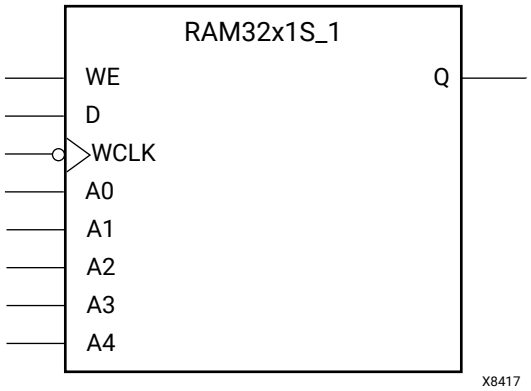
```
   .WCLK(WCLK), // Write clock input
   .WE(WE)      // Write enable input
);

// End of RAM32X1S_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

RAM32X1S_1

Primitive: 32-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock



Introduction

This design element is a 32-bit deep by 1-bit wide static random access memory with synchronous write capability. When the write enable is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When (WE) is High, any negative transition on (WCLK) loads the data on the data input (D) into the memory cell selected by the 5-bit address (A4:A0). For predictable performance, address and data inputs must be stable before a High-to-Low (WCLK) transition. This RAM block assumes an active-Low (WCLK). However, (WCLK) can be active-High or active-Low. Any inverter placed on the (WCLK) input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can use the INIT attribute to specify the initial contents of the RAM. If left unspecified, the initial contents default to all zeros.

Logic Table

Inputs			Outputs
WE (Mode)	**WCLK**	**D**	**O**
0 (read)	X	X	Data
1 (read)	0	X	Data
1 (read)	1	X	Data
1 (write)	↓	D	D
1 (read)	↑	X	Data
Data = memory cell addressed by bits A4:A0			

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 32-bit value	All zeros	Specifies the initial contents of the RAM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAM32X1S_1: 32 x 1 negedge write distributed (LUT) RAM (Mapped to SliceM LUT6)
--             7 Series
-- Xilinx HDL Language Template, version 2020.1

RAM32X1S_1_inst : RAM32X1S_1
generic map (
   INIT => X"00000000")
port map (
   O => O,        -- RAM output
   A0 => A0,      -- RAM address[0] input
   A1 => A1,      -- RAM address[1] input
   A2 => A2,      -- RAM address[2] input
   A3 => A3,      -- RAM address[3] input
   A4 => A4,      -- RAM address[4] input
   D => D,        -- RAM data input
   WCLK => WCLK,  -- Write clock input
   WE => WE       -- Write enable input
);

-- End of RAM32X1S_1_inst instantiation
```

Verilog Instantiation Template

```
// RAM32X1S_1: 32 x 1 negedge write distributed (LUT) RAM (Mapped to a SliceM LUT6)
//             7 Series
// Xilinx HDL Language Template, version 2020.1

RAM32X1S_1 #(
   .INIT(32'h00000000)  // Initial contents of RAM
)RAM32X1S_1_inst (
   .O(O),       // RAM output
   .A0(A0),     // RAM address[0] input
   .A1(A1),     // RAM address[1] input
   .A2(A2),     // RAM address[2] input
   .A3(A3),     // RAM address[3] input
   .A4(A4),     // RAM address[4] input
   .D(D),       // RAM data input
```

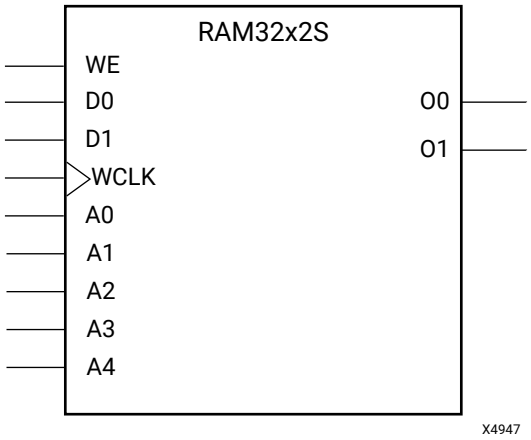
```
   .WCLK(WCLK), // Write clock input
   .WE(WE)      // Write enable input
);

// End of RAM32X1S_1_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

RAM32X2S

Primitive: 32-Deep by 2-Wide Static Synchronous RAM



Introduction

This design element is a 32-bit deep by 2-bit wide static random access memory with synchronous write capability. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When (WE) is High, any positive transition on (WCLK) loads the data on the data input (D1-D0) into the word selected by the 5-bit address (A4-A0). For predictable performance, address and data inputs must be stable before a Low-to-High (WCLK) transition. This RAM block assumes an active-High (WCLK). However, (WCLK) can be active-High or active-Low. Any inverter placed on the (WCLK) input net is absorbed into the block. The signal output on the data output pins (O1-O0) is the data that is stored in the RAM at the location defined by the values on the address pins.

You can use the INIT_00 and INIT_01 properties to specify the initial contents of RAM32X2S.

Logic Table

Inputs			Outputs
WE (Mode)	**WCLK**	**D**	**O0-O1**
0 (read)	X	X	Data
1 (read)	0	X	Data
1 (read)	1	X	Data
1 (write)	↑	D1:D0	D1:D0
1 (read)	↓	X	Data
Data = word addressed by bits A4:A0			

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Descriptions
INIT_00	HEX	Any 32-bit value	All zeros	INIT for bit 0 of RAM.
INIT_01	HEX	Any 32-bit value	All zeros	INIT for bit 1 of RAM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAM32X2S: 32 x 2 posedge write distributed (LUT) RAM (Mapped to SliceM LUT6)
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

RAM32X2S_inst : RAM32X2S
generic map (
   INIT_00 => X"00000000", --  INIT for bit 0 of RAM
   INIT_01 => X"00000000") --  INIT for bit 1 of RAM
port map (
   O0 => O0,      -- RAM data[0] output
   O1 => O1,      -- RAM data[1] output
   A0 => A0,      -- RAM address[0] input
   A1 => A1,      -- RAM address[1] input
   A2 => A2,      -- RAM address[2] input
   A3 => A3,      -- RAM address[3] input
   A4 => A4,      -- RAM address[4] input
   D0 => D0,      -- RAM data[0] input
   D1 => D1,      -- RAM data[1] input
   WCLK => WCLK,  -- Write clock input
   WE => WE       -- Write enable input
);

-- End of RAM32X2S_inst instantiation
```

Verilog Instantiation Template

```
// RAM32X2S: 32 x 2 posedge write distributed (LUT) RAM (Mapped to a SliceM LUT6)
//           7 Series
// Xilinx HDL Language Template, version 2020.1

RAM32X2S #(
   .INIT_00(32'h00000000), // INIT for bit 0 of RAM
   .INIT_01(32'h00000000)  // INIT for bit 1 of RAM
) RAM32X2S_inst (
   .O0(O0),      // RAM data[0] output
   .O1(O1),      // RAM data[1] output
   .A0(A0),      // RAM address[0] input
   .A1(A1),      // RAM address[1] input
   .A2(A2),      // RAM address[2] input
```

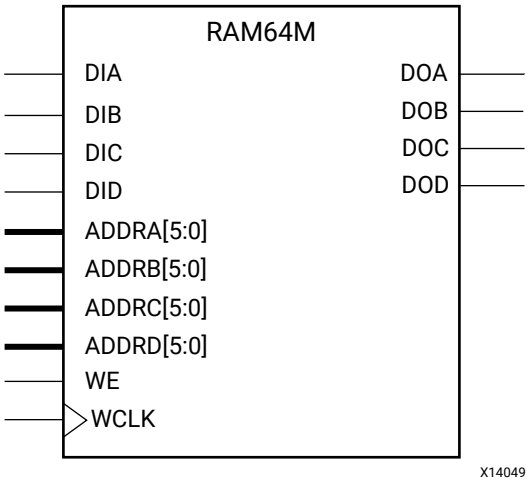
```
   .A3(A3),     // RAM address[3] input
   .A4(A4),     // RAM address[4] input
   .D0(D0),     // RAM data[0] input
   .D1(D1),     // RAM data[1] input
   .WCLK(WCLK), // Write clock input
   .WE(WE)      // Write enable input
);

// End of RAM32X2S_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

RAM64M

Primitive: 64-Deep by 4-bit Wide Multi Port Random Access Memory (Select RAM)



Introduction

This design element is a 64-bit deep by 4-bit wide, multi-port, random access memory with synchronous write and asynchronous independent bit wide read capability. This RAM is implemented using the LUT resources of the device (also known as SelectRAM™+) and does not consume any of the block RAM resources of the device. The RAM64M component is implemented in a single slice, and consists of one 4-bit write, 1-bit read port, and three separate 1-bit read ports from the same memory allowing for 4-bit write and independent bit read access RAM.

- If the DIA, DIB, DIC, and DID inputs are all tied to the same data inputs, the RAM can become a 1 read/write port, 3 independent read port 64x1 quad port memory.
- If DID is grounded, DOD is not used.
- If ADDRA, ADDRB, and ADDRC are tied to the same address, the RAM becomes a 64x3 simple dual port RAM.
- If ADDRD is tied to ADDRA, ADDRB, and ADDRC, the RAM is a 64x4 single port RAM.

There are several other possible configurations for this RAM.

Port Descriptions

Port	Direction	Width	Function
DOA	Output	1	Read port data outputs addressed by ADDRA.
DOB	Output	1	Read port data outputs addressed by ADDRB.

Port	Direction	Width	Function
DOC	Output	1	Read port data outputs addressed by ADDRC.
DOD	Output	1	Read/Write port data outputs addressed by ADDRD.
DIA	Input	1	Write data inputs addressed by ADDRD (read output is addressed by ADDRA).
DIB	Input	1	Write data inputs addressed by ADDRD (read output is addressed by ADDRB).
DIC	Input	1	Write data inputs addressed by ADDRD (read output is addressed by ADDRC).
DID	Input	1	Write data inputs addressed by ADDRD.
ADDRA	Input	6	Read address bus A.
ADDRB	Input	6	Read address bus B.
ADDRC	Input	6	Read address bus C.
ADDRD	Input	6	4-bit data write port, 1-bit data read port address bus D.
WE	Input	1	Write Enable.
WCLK	Input	1	Write clock (reads are asynchronous).

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

This element can be inferred by some synthesis tools by describing a RAM with a synchronous write and asynchronous read capability. Consult your synthesis tool documentation for details on RAM inference capabilities and coding examples. Xilinx suggests that you instantiate this component if you have a need to implicitly specify the RAM function, or if you need to manually place or relationally place the component. If a synchronous read capability is desired, the outputs can be connected to an FDRE (FDCE if asynchronous reset is needed) to improve the output timing of the function. However, this is not necessary for the proper operation of the RAM. If you want to have the data clocked on the negative edge of a clock, an inverter can be described on the clock input to this component. This inverter will be absorbed into the block giving the ability to write to the RAM on falling clock edges.

If instantiated, the following connections should be made to this component:

- Connect the WCLK input to the desired clock source, the DIA, DIB, DIC
- Connect the DIA, DIB, DIC, and DID inputs to the data source to be stored
- Connect the DOA, DOB, DOC, and DOD outputs to an FDCE D input or other appropriate data destination, or leave unconnected if not used
- Connect the WE clock enable pin to the proper write enable source in the design
- Connect the ADDRD bus to the source for the read/write addressing

- Connect the ADDRA, ADDRB, and ADDRC buses to the appropriate read address connections

The optional INIT_A, INIT_B, INIT_C and INIT_D attributes let you specify the initial memory contents of each port using a 64-bit hexadecimal value. The INIT value correlates to the RAM addressing by the following equation: $ADDRy[z] = INIT\_y[z]$. For instance, if the RAM ADDRC port is addressed to 00001, then the INIT_C[1] values would be the initial values shown on the DOC port before the first write occurs at that address. If left unspecified, the initial contents will default to all zeros.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT_A	HEX	Any 64-bit value	All zero	Specifies the initial contents of the RAM on port A.
INIT_B	HEX	Any 64-bit value	All zero	Specifies the initial contents of the RAM on port B.
INIT_C	HEX	Any 64-bit value	All zero	Specifies the initial contents of the RAM on port C.
INIT_D	HEX	Any 64-bit value	All zero	Specifies the initial contents of the RAM on port D.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAM64M: 64-deep by 4-wide Multi Port LUT RAM (Mapped to four SliceM LUT6s)
--         7 Series
-- Xilinx HDL Language Template, version 2020.1

RAM64M_inst : RAM64M
generic map (
   INIT_A => X"0000000000000000",   -- Initial contents of A port
   INIT_B => X"0000000000000000",   -- Initial contents of B port
   INIT_C => X"0000000000000000",   -- Initial contents of C port
   INIT_D => X"0000000000000000")   -- Initial contents of D port
port map (
   DOA => DOA, -- Read port A 1-bit output
   DOB => DOB, -- Read port B 1-bit output
   DOC => DOC, -- Read port C 1-bit output
   DOD => DOD, -- Read/Write port D 1-bit output
   ADDRA => ADDRA,   -- Read port A 6-bit address input
   ADDRB => ADDRB,   -- Read port B 6-bit address input
   ADDRC => ADDRC,   -- Read port C 6-bit address input
   ADDRD => ADDRD,   -- Read/Write port D 6-bit address input
   DIA => DIA, -- RAM 1-bit data write input addressed by ADDRD,
               -- read addressed by ADDRA
   DIB => DIB, -- RAM 1-bit data write input addressed by ADDRD,
               -- read addressed by ADDRB
   DIC => DIC, -- RAM 1-bit data write input addressed by ADDRD,
               -- read addressed by ADDRC
   DID => DID, -- RAM 1-bit data write input addressed by ADDRD,
               -- read addressed by ADDRD
   WCLK => WCLK,  -- Write clock input
   WE => WE       -- Write enable input
);
-- End of RAM64M_inst instantiation
```

Verilog Instantiation Template

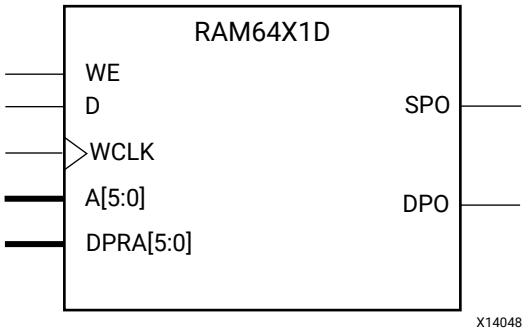
```
// RAM64M: 64-deep by 4-wide Multi Port LUT RAM (Mapped to four SliceM LUT6s)
//         7 Series
// Xilinx HDL Language Template, version 2020.1

RAM64M #(
   .INIT_A(64'h0000000000000000), // Initial contents of A Port
   .INIT_B(64'h0000000000000000), // Initial contents of B Port
   .INIT_C(64'h0000000000000000), // Initial contents of C Port
   .INIT_D(64'h0000000000000000)  // Initial contents of D Port
) RAM64M_inst (
   .DOA(DOA),     // Read port A 1-bit output
   .DOB(DOB),     // Read port B 1-bit output
   .DOC(DOC),     // Read port C 1-bit output
   .DOD(DOD),     // Read/write port D 1-bit output
   .DIA(DIA),     // RAM 1-bit data write input addressed by ADDRD,
                  //   read addressed by ADDRA
   .DIB(DIB),     // RAM 1-bit data write input addressed by ADDRD,
                  //   read addressed by ADDRB
   .DIC(DIC),     // RAM 1-bit data write input addressed by ADDRD,
                  //   read addressed by ADDRC
   .DID(DID),     // RAM 1-bit data write input addressed by ADDRD,
                  //   read addressed by ADDRD
   .ADDRA(ADDRA), // Read port A 6-bit address input
   .ADDRB(ADDRB), // Read port B 6-bit address input
   .ADDRC(ADDRC), // Read port C 6-bit address input
   .ADDRD(ADDRD), // Read/write port D 6-bit address input
   .WE(WE),       // Write enable input
   .WCLK(WCLK)    // Write clock input
);

// End of RAM64M_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

RAM64X1D

Primitive: 64-Deep by 1-Wide Dual Port Static Synchronous RAM



Introduction

This design element is a 64-bit deep by 1-bit wide static dual port random access memory with synchronous write capability. The device has two separate address ports: the read address (DPRA5:DPRA0) and the write address (A5:A0). These two address ports are completely asynchronous. The read address controls the location of the data driven out of the output pin (DPO), and the write address controls the destination of a valid write transaction. When the write enable (WE) is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected.

When WE is High, any positive transition on WCLK loads the data on the data input (D) into the memory cell selected by the 6-bit (A0:A5) write address. For predictable performance, write address and data inputs must be stable before a Low-to-High WCLK transition. This RAM block assumes an active-High WCLK. WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The SPO output reflects the data in the memory cell addressed by A5:A0. The DPO output reflects the data in the memory cell addressed by DPRA5:DPRA0. The write process is not affected by the address on the read address port.

You can use the INIT attribute to specify the initial contents of the RAM. If left unspecified, the initial contents default to all zeros.

Logic Table

Inputs			Outputs	
WE (mode)	**WCLK**	**D**	**SPO**	**DPO**
0 (read)	X	X	data_a	data_d
1 (read)	0	X	data_a	data_d
1 (read)	1	X	data_a	data_d

Inputs			Outputs	
WE (mode)	**WCLK**	**D**	**SPO**	**DPO**
1 (write)	↑	D	D	data_d
1 (read)	↓	X	data_a	data_d
data_a = memory cell addressed by bits A5:A0 data_d = memory cell addressed by bits DPRA5:DPRA0				

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 64-bit value	All zeros	Specifies the initial contents of the RAM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAM64X1D: 64 x 1 negative edge write, asynchronous read
--           dual-port distributed RAM (Mapped to SliceM LUT6)
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

RAM64X1D_1_inst : RAM64X1D_1
generic map (
   INIT => X"0000000000000000") -- Initial contents of RAM
port map (
   DPO => DPO,      -- Read-only 1-bit data output
   SPO => SPO,      -- R/W 1-bit data output
   A0 => A0,        -- R/W address[0] input bit
   A1 => A1,        -- R/W address[1] input bit
   A2 => A2,        -- R/W address[2] input bit
   A3 => A3,        -- R/W address[3] input bit
   A4 => A4,        -- R/W address[4] input bit
   A5 => A5,        -- R/W address[5] input bit
   D => D,          -- Write 1-bit data input
   DPRA0 => DPRA0,  -- Read-only address[0] input bit
   DPRA1 => DPRA1,  -- Read-only address[1] input bit
   DPRA2 => DPRA2,  -- Read-only address[2] input bit
   DPRA3 => DPRA3,  -- Read-only address[3] input bit
   DPRA4 => DPRA4,  -- Read-only address[4] input bit
   DPRA5 => DPRA5,  -- Read-only address[5] input bit
   WCLK => WCLK,    -- Write clock input
   WE => WE         -- Write enable input
);

-- End of RAM64X1D_1_inst instantiation
```

Verilog Instantiation Template

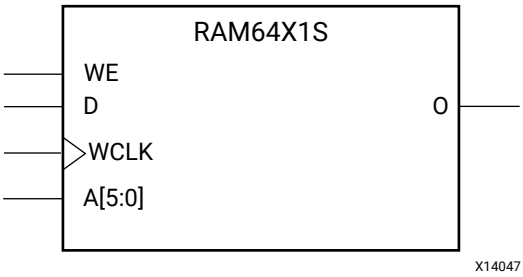
```
// RAM64X1D: 64 x 1 positive edge write, asynchronous read dual-port
//           distributed RAM (Mapped to a SliceM LUT6)
//           7 Series
// Xilinx HDL Language Template, version 2020.1

RAM64X1D #(
   .INIT(64'h0000000000000000) // Initial contents of RAM
) RAM64X1D_inst (
   .DPO(DPO),     // Read-only 1-bit data output
   .SPO(SPO),     // Rw/ 1-bit data output
   .A0(A0),       // Rw/ address[0] input bit
   .A1(A1),       // Rw/ address[1] input bit
   .A2(A2),       // Rw/ address[2] input bit
   .A3(A3),       // Rw/ address[3] input bit
   .A4(A4),       // Rw/ address[4] input bit
   .A5(A5),       // Rw/ address[5] input bit
   .D(D),         // Write 1-bit data input
   .DPRA0(DPRA0), // Read-only address[0] input bit
   .DPRA1(DPRA1), // Read-only address[1] input bit
   .DPRA2(DPRA2), // Read-only address[2] input bit
   .DPRA3(DPRA3), // Read-only address[3] input bit
   .DPRA4(DPRA4), // Read-only address[4] input bit
   .DPRA5(DPRA5), // Read-only address[5] input bit
   .WCLK(WCLK),   // Write clock input
   .WE(WE)        // Write enable input
);

// End of RAM64X1D_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

RAM64X1S

Primitive: 64-Deep by 1-Wide Static Synchronous RAM



Introduction

This design element is a 64-bit deep by 1-bit wide static random access memory (RAM) with synchronous write capability. When the write enable is set Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When WE is set High, any positive transition on WCLK loads the data on the data input (D) into the memory cell selected by the 6-bit address (A5:A0). This RAM block assumes an active-High WCLK. However, WCLK can be active-High or active-Low. Any inverter placed on the WCLK input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the memory cell defined by the values on the address pins.

You can use the INIT attribute to specify the initial contents of the RAM. If left unspecified, the initial contents default to all zeros.

Logic Table

Mode selection is shown in the following logic table.

Inputs			Outputs
WE (mode)	**WCLK**	**D**	**O**
0 (read)	X	X	Data
1 (read)	0	X	Data
1 (read)	1	X	Data
1 (write)	↑	D	D
1 (read)	↓	X	Data
Data = memory cell addressed by bits A5:A0			

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 64-bit value	All zeros	Specifies the initial contents of the RAM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAM64X1S: 64 x 1 positive edge write, asynchronous read single-port distributed RAM (Mapped to SliceM LUT6)
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

RAM64X1S_inst : RAM64X1S
generic map (
   INIT => X"0000000000000000")
port map (
   O => O,         -- 1-bit data output
   A0 => A0,       -- Address[0] input bit
   A1 => A1,       -- Address[1] input bit
   A2 => A2,       -- Address[2] input bit
   A3 => A3,       -- Address[3] input bit
   A4 => A4,       -- Address[4] input bit
   A5 => A5,       -- Address[5] input bit
   D => D,         -- 1-bit data input
   WCLK => WCLK,   -- Write clock input
   WE => WE        -- Write enable input
);

-- End of RAM64X1S_inst instantiation
```

Verilog Instantiation Template

```
// RAM64X1S: 64 x 1 positive edge write, asynchronous read single-port
//           distributed RAM (Mapped to a SliceM LUT6)
//           7 Series
// Xilinx HDL Language Template, version 2020.1

RAM64X1S #(
   .INIT(64'h0000000000000000) // Initial contents of RAM
) RAM64X1S_inst (
   .O(O),         // 1-bit data output
   .A0(A0),       // Address[0] input bit
   .A1(A1),       // Address[1] input bit
   .A2(A2),       // Address[2] input bit
   .A3(A3),       // Address[3] input bit
   .A4(A4),       // Address[4] input bit
   .A5(A5),       // Address[5] input bit
   .D(D),         // 1-bit data input
```

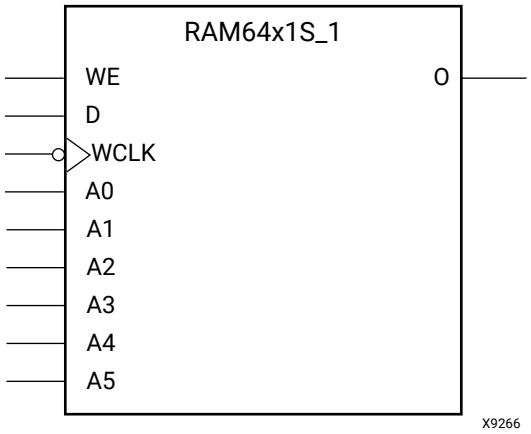
```
   .WCLK(WCLK),  // Write clock input
   .WE(WE)       // Write enable input
);

// End of RAM64X1S_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

RAM64X1S_1

Primitive: 64-Deep by 1-Wide Static Synchronous RAM with Negative-Edge Clock



Introduction

This design element is a 64-bit deep by 1-bit wide static random access memory with synchronous write capability. When the write enable is Low, transitions on the write clock (WCLK) are ignored and data stored in the RAM is not affected. When (WE) is High, any negative transition on (WCLK) loads the data on the data input (D) into the memory cell selected by the 6-bit address (A5:A0). For predictable performance, address and data inputs must be stable before a High-to-Low (WCLK) transition. This RAM block assumes an active-Low (WCLK). However, (WCLK) can be active-High or active-Low. Any inverter placed on the (WCLK) input net is absorbed into the block.

The signal output on the data output pin (O) is the data that is stored in the memory cell defined by the values on the address pins.

You can use the INIT attribute to specify the initial contents of the RAM. If left unspecified, the initial contents default to all zeros.

Logic Table

Inputs			Outputs
WE (mode)	**WCLK**	**D**	**O**
0 (read)	X	X	Data
1 (read)	0	X	Data
1 (read)	1	X	Data
1 (write)	↓	D	D
1 (read)	↑	X	Data

Inputs			Outputs
WE (mode)	**WCLK**	**D**	**O**
Data = memory cell addressed by bits A5:A0			

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 64-bit value	All zeros	Specifies the initial contents of the RAM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAM64X1S_1: 64 x 1 negative edge write, asynchronous read single-port distributed RAM (Mapped to SliceM LUT6)
--             7 Series
-- Xilinx HDL Language Template, version 2020.1

RAM64X1S_1_inst : RAM64X1S_1
generic map (
   INIT => X"0000000000000000")
port map (
   O => O,         -- 1-bit data output
   A0 => A0,       -- Address[0] input bit
   A1 => A1,       -- Address[1] input bit
   A2 => A2,       -- Address[2] input bit
   A3 => A3,       -- Address[3] input bit
   A4 => A4,       -- Address[4] input bit
   A5 => A5,       -- Address[5] input bit
   D => D,         -- 1-bit data input
   WCLK => WCLK,   -- Write clock input
   WE => WE        -- Write enable input
);

-- End of RAM64X1S_1_inst instantiation
```

Verilog Instantiation Template

```
// RAM64X1S_1: 64 x 1 negative edge write, asynchronous read single-port
//             distributed RAM (Mapped to a SliceM LUT6)
//             7 Series
// Xilinx HDL Language Template, version 2020.1

RAM64X1S_1 #(
   .INIT(64'h0000000000000000) // Initial contents of RAM
) RAM64X1S_1_inst (
   .O(O),         // 1-bit data output
```

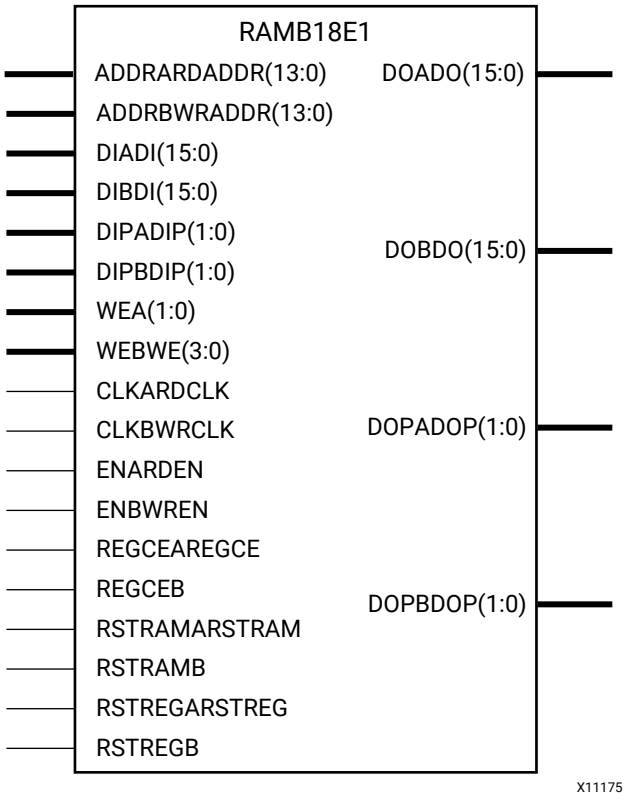
```
   .A0(A0),      // Address[0] input bit
   .A1(A1),      // Address[1] input bit
   .A2(A2),      // Address[2] input bit
   .A3(A3),      // Address[3] input bit
   .A4(A4),      // Address[4] input bit
   .A5(A5),      // Address[5] input bit
   .D(D),        // 1-bit data input
   .WCLK(WCLK),  // Write clock input
   .WE(WE)       // Write enable input
);

// End of RAM64X1S_1_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

RAMB18E1

Primitive: 18K-bit Configurable Synchronous Block RAM



Introduction

7 series devices contain several block RAM memories that can be configured as FIFOs, automatic error correction RAM, or general-purpose 36Kb or 18Kb RAM/ROM memories. These block RAM memories offer fast and flexible storage of large amounts of on-chip data. The RAMB18E1 allows access to the block RAM in the 18Kb configuration.

This element can be configured and used as a 1-bit wide by 16K deep to an 18-bit wide by 1024-bit deep true dual port RAM. This element can also be configured as a 36-bit wide by 512 deep simple dual port RAM. Both read and write operations are fully synchronous to the supplied clock(s) to the component. However, the READ and WRITE ports can operate fully independent and asynchronous to each other, accessing the same memory array. When configured in the wider data width modes, byte-enable write operations are possible, and an optional output register can be used to reduce the clock-to-out times of the RAM.

Port Descriptions

Port	Direction	Width	Function
ADDRARDADDR <13:0>	Input	14	Port A address input bus/Read address input bus.
ADDRBWRADDR <13:0>	Input	14	Port B address input bus/Write address input bus.
CLKARDCLK	Input	1	Rising edge port A clock input/Read clock input.
CLKBWRCLK	Input	1	Rising edge port B clock input/Write clock input.
DIADI<15:0>	Input	16	Port A data input bus/Data input bus addressed by WRADDR. When RAM_MODE="SDP", DIADI is the logical DI<15:0>.
DIBDI<15:0>	Input	16	Port B data input bus/Data input bus addressed by WRADDR. When RAM_MODE="SDP", DIBDI is the logical DI<31:16>.
DIPADIP<1:0>	Input	2	Port A parity data input bus/Data parity input bus addressed by WRADDR. When RAM_MODE="SDP", DIPADIP is the logical DIP<1:0>.
DIPBDIP<1:0>	Input	2	Port B parity data input bus/Data parity input bus addressed by WRADDR. When RAM_MODE="SDP", DIPBDIP is the logical DIP<3:2>.
DOADO<15:0>	Output	16	Port A data output bus/Data output bus addressed by RDADDR. When RAM_MODE="SDP", DOADO is the logical DO<15:0>.
DOBDO<15:0>	Output	16	Port B data output bus/Data output bus addressed by RDADDR. When RAM_MODE="SDP", DOBDO is the logical DO<31:16>.
DOPADOP<1:0>	Output	2	Port A parity data output bus/Data parity output bus addressed by RDADDR. When RAM_MODE="SDP", DOPADOP is the logical DOP<1:0>.
DOPBDOP<1:0>	Output	2	Port B parity data output bus/Data parity output bus addressed by RDADDR. When RAM_MODE="SDP", DOPBDOP is the logical DOP<3:2>.
ENARDEN	Input	1	Port A RAM enable/Read enable.
ENBWREN	Input	1	Port B RAM enable/Write enable.
REGCEAREGCE	Input	1	Port A output register clock enable input/Output register clock enable input (valid only when DOA_REG=1).
REGCEB	Input	1	Port B output register clock enable (valid only when DOB_REG=1 and RAM_MODE="TDP").
RSTRAMARSTRAM	Input	1	Synchronous data latch set/reset to value indicated by SRVAL_A. RSTRAMARSTRAM sets/resets the BRAM data output latch when DO_REG=0 or 1. If DO_REG=1 there is a cycle of latency between the internal data latch node that is reset by RSTRAMARSTRAM and the DO output of the BRAM. This signal resets port A RAM output when RAM_MODE="TDP" and the entire RAM output when RAM_MODE="SDP".
RSTRAMB	Input	1	Synchronous data latch set/reset to value indicated by SRVAL_B. RSTRAMB sets/resets the BRAM data output latch when DO_REG=0 or 1. If DO_REG=1 there is a cycle of latency between the internal data latch node that is reset by RSTRAMB and the DO output of the BRAM. Not used when RAM_MODE="SDP".

Port	Direction	Width	Function
RSTREGARSTREG	Input	1	Synchronous output register set/reset to value indicated by SRVAL_A. RSTREGARSTREG sets/resets the output register when DO_REG=1. RSTREG_PRIORITY_A determines if this signal gets priority over REGCEAREGCE. This signal resets port A output when RAM_MODE="TDP" and the entire output port when RAM_MODE="SDP".
RSTREGB	Input	1	Synchronous output register set/reset to value indicated by SRVAL_B. RSTREGB sets/resets the output register when DO_REG=1. RSTREG_PRIORITY_B determines if this signal gets priority over REGCEB. Not used when RAM_MODE="SDP".
WEA<1:0>	Input	2	Port A byte-wide write enable. Not used when RAM_MODE="SDP". See User Guide for WEA mapping for different port widths.
WEBWE<3:0>	Input	4	Port B byte-wide write enable/Write enable. See User Guide for WEBWE mapping for different port widths.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	Yes
Macro support	Yes

Available Attributes

Attribute	Type	Allowed Values	Default	Description
RDADDR_COLLISION_HWCONFIG	STRING	"DELAYED_WRITE", "PERFORMANCE"	"DELAYED_WRITE"	When set to "PERFORMANCE" allows for higher clock performance (frequency) in READ_FIRST mode. If using the same clock on both ports of the RAM with "PERFORMANCE" mode, the address overlap collision rules apply where in "DELAYED_WRITE" mode, you can safely use the BRAM without incurring collisions.
SIM_COLLISION_CHECK	STRING	"ALL", "GENERATE_X_ONLY", "NONE", "WARNING_ONLY"	"ALL"	Allows modification of the simulation behavior so that if a memory collision occurs • "ALL" = warning produced and affected outputs/memory go unknown (X) • "WARNING_ONLY" = warning produced and affected outputs/memory retain last value • "GENERATE_X_ONLY" = no warning and affected outputs/memory go unknown (X) • "NONE" = no warning and affected outputs/memory retain last value ***Note***: Use this setting carefully. Setting it to a value other than "ALL" can mask design problems during simulation.

Attribute	Type	Allowed Values	Default	Description
DOA_REG, DOB_REG	DECIMAL	0, 1	0	A value of 1 enables the output registers to the RAM enabling quicker clock-to-out from the RAM at the expense of an added clock cycle of read latency. A value of 0 allows a read in one clock cycle but will result in slower clock-to-out timing. Applies to port A/B in TDP mode and up to 18 lower bits (including parity bits) in SDP mode.
INIT_A, INIT_B	HEX	18 bit HEX	18'h00000	Specifies the initial value on the port output after configuration. Applies to Port A/B in TDP mode and up to 18 lower bits (including parity bits) in SDP mode.
INIT_00 to INIT_3F	HEX	256 bit HEX	All zeros	Allows specification of the initial contents of the 16Kb data memory array.
INIT_FILE	STRING	String representing file name and location	None	File name of file used to specify initial RAM contents.
INITP_00 to INITP_07	HEX	256 bit HEX	All zeros	Allows specification of the initial contents of the 2Kb parity data memory array.
RAM_MODE	STRING	"TDP", "SDP"	"TDP"	Selects simple dual port (SDP) or true dual port (TDP) mode.
READ_WIDTH_A	DECIMAL	0, 1, 2, 4, 9, 18, 36, 72	0	Specifies the desired data width for a read on Port A, including parity bits. This value must be 0 if the Port A is not used. Otherwise, it should be set to the desired port width. In "SDP" mode, this is the read width including parity bits.
READ_WIDTH_B	DECIMAL	0, 1, 2, 4, 9, 18	0	Specifies the desired data width for a read on Port B including parity bits. This value must be 0 if the Port B is not used. Otherwise, it should be set to the desired port width. Not used for "SDP" mode.
RSTREG_PRIORITY_A, RSTREG_PRIORITY_B	STRING	"RSTREG", "REGCE"	"RSTREG"	Selects register priority for RSTREG or REGCE. Applies to port A/B in TDP mode and up to 18 lower bits (including parity bits) in SDP mode.
SIM_DEVICE	STRING	"7SERIES"	""7SERIES""	Must be set to "7SERIES" in order to exhibit proper simulation behavior under all conditions.
SRVAL_A, SRVAL_B	HEX	18 bit HEX	18'h00000	Specifies the output value of the RAM upon assertion of the synchronous reset (RSTREG) signal.

Attribute	Type	Allowed Values	Default	Description
WRITE_MODE_A, WRITE_MODE_B	STRING	"WRITE_FIRST", "NO_CHANGE", "READ_FIRST"	"WRITE_FIRST"	Specifies output behavior of the port being written to. • "WRITE_FIRST": Written value appears on output port of the RAM • "READ_FIRST": Previous RAM contents for that memory location appear on the output port • "NO_CHANGE": Previous value on the output port remains the same. When RAM_MODE="SDP", WRITE_MODE can not be set to "NO_CHANGE". For simple dual port implementations you should set this attribute to "READ_FIRST" if using the same clock on both ports, or set it to "WRITE_FIRST" if using different clocks. This generally yields an improved collision or address overlap behavior.
WRITE_WIDTH_A	DECIMAL	0, 1, 2, 4, 9, 18	0	Specifies the desired data width for a write to Port A including parity bits. This value must be 0 if the port is not used. Otherwise should be set to the desired write width. Not used in SDP mode.
WRITE_WIDTH_B	DECIMAL	0, 1, 2, 4, 9, 18, 36, 72	0	Specifies the desired data width for a write to Port B including parity bits. This value must be 0 if the port is not used. Otherwise should be set to the desired write width. In SDP mode, this is the write width including parity bits.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAMB18E1: 18K-bit Configurable Synchronous Block RAM
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

RAMB18E1_inst : RAMB18E1
generic map (
   -- Address Collision Mode: "PERFORMANCE" or "DELAYED_WRITE"
   RDADDR_COLLISION_HWCONFIG => "DELAYED_WRITE",
   -- Collision check: Values ("ALL", "WARNING_ONLY", "GENERATE_X_ONLY" or "NONE")
   SIM_COLLISION_CHECK => "ALL",
   -- DOA_REG, DOB_REG: Optional output register (0 or 1)
   DOA_REG => 0,
   DOB_REG => 0,
   -- INITP_00 to INITP_07: Initial contents of parity memory array
   INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000",
   -- INIT_00 to INIT_3F: Initial contents of data memory array
```

```
    INIT_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_07 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_0F => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_10 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_11 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_12 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_13 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_14 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_15 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_16 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_17 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_18 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_19 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_1F => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_20 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_21 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_22 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_23 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_24 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_25 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_26 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_27 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_28 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_29 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_2F => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_30 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_31 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_32 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_33 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_34 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_35 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_36 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_37 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_38 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_39 => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3A => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3B => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3C => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3D => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3E => X"0000000000000000000000000000000000000000000000000000000000000000",
    INIT_3F => X"0000000000000000000000000000000000000000000000000000000000000000",
    -- INIT_A, INIT_B: Initial values on output ports
    INIT_A => X"00000",
    INIT_B => X"00000",
    -- Initialization File: RAM initialization file
    INIT_FILE => "NONE",
    -- RAM Mode: "SDP" or "TDP"
    RAM_MODE => "TDP",
    -- READ_WIDTH_A/B, WRITE_WIDTH_A/B: Read/write width per port
    READ_WIDTH_A => 0,                                                              -- 0-72
    READ_WIDTH_B => 0,                                                              -- 0-18
    WRITE_WIDTH_A => 0,                                                             -- 0-18
    WRITE_WIDTH_B => 0,                                                             -- 0-72
    -- RSTREG_PRIORITY_A, RSTREG_PRIORITY_B: Reset or enable priority ("RSTREG" or "REGCE")
    RSTREG_PRIORITY_A => "RSTREG",
    RSTREG_PRIORITY_B => "RSTREG",
```

```
   -- SRVAL_A, SRVAL_B: Set/reset value for output
   SRVAL_A => X"00000",
   SRVAL_B => X"00000",
   -- Simulation Device: Must be set to "7SERIES" for simulation behavior
   SIM_DEVICE => "7SERIES",
   -- WriteMode: Value on output upon a write ("WRITE_FIRST", "READ_FIRST", or "NO_CHANGE")
   WRITE_MODE_A => "WRITE_FIRST",
   WRITE_MODE_B => "WRITE_FIRST"
)
port map (
   -- Port A Data: 16-bit (each) output: Port A data
   DOADO => DOADO,                 -- 16-bit output: A port data/LSB data
   DOPADOP => DOPADOP,             -- 2-bit output: A port parity/LSB parity
   -- Port B Data: 16-bit (each) output: Port B data
   DOBDO => DOBDO,                 -- 16-bit output: B port data/MSB data
   DOPBDOP => DOPBDOP,             -- 2-bit output: B port parity/MSB parity
   -- Port A Address/Control Signals: 14-bit (each) input: Port A address and control signals (read port
   -- when RAM_MODE="SDP")
   ADDRARDADDR => ADDRARDADDR,     -- 14-bit input: A port address/Read address
   CLKARDCLK => CLKARDCLK,         -- 1-bit input: A port clock/Read clock
   ENARDEN => ENARDEN,             -- 1-bit input: A port enable/Read enable
   REGCEAREGCE => REGCEAREGCE,     -- 1-bit input: A port register enable/Register enable
   RSTRAMARSTRAM => RSTRAMARSTRAM, -- 1-bit input: A port set/reset
   RSTREGARSTREG => RSTREGARSTREG, -- 1-bit input: A port register set/reset
   WEA => WEA,                     -- 2-bit input: A port write enable
   -- Port A Data: 16-bit (each) input: Port A data
   DIADI => DIADI,                 -- 16-bit input: A port data/LSB data
   DIPADIP => DIPADIP,             -- 2-bit input: A port parity/LSB parity
   -- Port B Address/Control Signals: 14-bit (each) input: Port B address and control signals (write port
   -- when RAM_MODE="SDP")
   ADDRBWRADDR => ADDRBWRADDR,     -- 14-bit input: B port address/Write address
   CLKBWRCLK => CLKBWRCLK,         -- 1-bit input: B port clock/Write clock
   ENBWREN => ENBWREN,             -- 1-bit input: B port enable/Write enable
   REGCEB => REGCEB,               -- 1-bit input: B port register enable
   RSTRAMB => RSTRAMB,             -- 1-bit input: B port set/reset
   RSTREGB => RSTREGB,             -- 1-bit input: B port register set/reset
   WEBWE => WEBWE,                 -- 4-bit input: B port write enable/Write enable
   -- Port B Data: 16-bit (each) input: Port B data
   DIBDI => DIBDI,                 -- 16-bit input: B port data/MSB data
   DIPBDIP => DIPBDIP              -- 2-bit input: B port parity/MSB parity
);

-- End of RAMB18E1_inst instantiation
```

Verilog Instantiation Template

```
// RAMB18E1: 18K-bit Configurable Synchronous Block RAM
//           7 Series
// Xilinx HDL Language Template, version 2020.1

RAMB18E1 #(
   // Address Collision Mode: "PERFORMANCE" or "DELAYED_WRITE"
   .RDADDR_COLLISION_HWCONFIG("DELAYED_WRITE"),
   // Collision check: Values ("ALL", "WARNING_ONLY", "GENERATE_X_ONLY" or "NONE")
   .SIM_COLLISION_CHECK("ALL"),
   // DOA_REG, DOB_REG: Optional output register (0 or 1)
   .DOA_REG(0),
   .DOB_REG(0),
   // INITP_00 to INITP_07: Initial contents of parity memory array
   .INITP_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
   // INIT_00 to INIT_3F: Initial contents of data memory array
   .INIT_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
```

```
    .INIT_08(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_09(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_0A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_0B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_0C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_0D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_0E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_0F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_10(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_11(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_12(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_13(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_14(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_15(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_16(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_17(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_18(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_19(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_1A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_1B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_1C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_1D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_1E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_1F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_20(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_21(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_22(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_23(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_24(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_25(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_26(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_27(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_28(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_29(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_30(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_31(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_32(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_33(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_34(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_35(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_36(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_37(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_38(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_39(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    // INIT_A, INIT_B: Initial values on output ports
    .INIT_A(18'h00000),
    .INIT_B(18'h00000),
    // Initialization File: RAM initialization file
    .INIT_FILE("NONE"),
    // RAM Mode: "SDP" or "TDP"
    .RAM_MODE("TDP"),
    // READ_WIDTH_A/B, WRITE_WIDTH_A/B: Read/write width per port
    .READ_WIDTH_A(0),                                                                   // 0-72
    .READ_WIDTH_B(0),                                                                   // 0-18
    .WRITE_WIDTH_A(0),                                                                  // 0-18
    .WRITE_WIDTH_B(0),                                                                  // 0-72
    // RSTREG_PRIORITY_A, RSTREG_PRIORITY_B: Reset or enable priority ("RSTREG" or "REGCE")
    .RSTREG_PRIORITY_A("RSTREG"),
    .RSTREG_PRIORITY_B("RSTREG"),
    // SRVAL_A, SRVAL_B: Set/reset value for output
    .SRVAL_A(18'h00000),
    .SRVAL_B(18'h00000),
    // Simulation Device: Must be set to "7SERIES" for simulation behavior
    .SIM_DEVICE("7SERIES"),
    // WriteMode: Value on output upon a write ("WRITE_FIRST", "READ_FIRST", or "NO_CHANGE")
    .WRITE_MODE_A("WRITE_FIRST"),
    .WRITE_MODE_B("WRITE_FIRST")
```

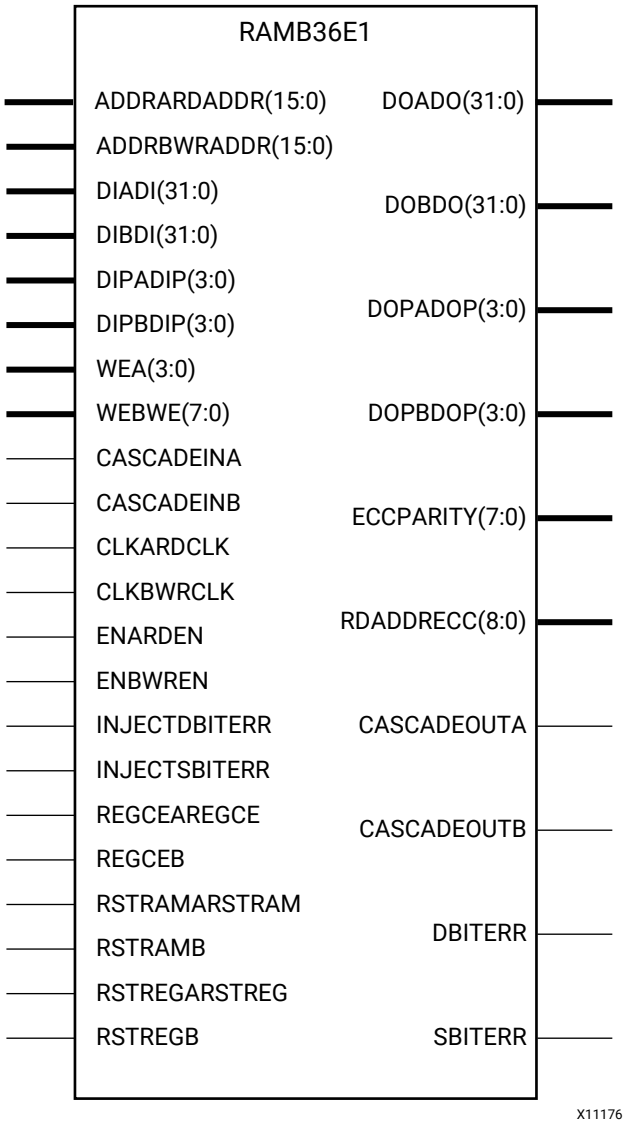
```
)
RAMB18E1_inst (
   // Port A Data: 16-bit (each) output: Port A data
   .DOADO(DOADO),                  // 16-bit output: A port data/LSB data
   .DOPADOP(DOPADOP),              // 2-bit output: A port parity/LSB parity
   // Port B Data: 16-bit (each) output: Port B data
   .DOBDO(DOBDO),                  // 16-bit output: B port data/MSB data
   .DOPBDOP(DOPBDOP),              // 2-bit output: B port parity/MSB parity
  // Port A Address/Control Signals: 14-bit (each) input: Port A address and control signals (read port
   // when RAM_MODE="SDP")
   .ADDRARDADDR(ADDRARDADDR),      // 14-bit input: A port address/Read address
   .CLKARDCLK(CLKARDCLK),          // 1-bit input: A port clock/Read clock
   .ENARDEN(ENARDEN),              // 1-bit input: A port enable/Read enable
   .REGCEAREGCE(REGCEAREGCE),      // 1-bit input: A port register enable/Register enable
   .RSTRAMARSTRAM(RSTRAMARSTRAM), // 1-bit input: A port set/reset
   .RSTREGARSTREG(RSTREGARSTREG), // 1-bit input: A port register set/reset
   .WEA(WEA),                      // 2-bit input: A port write enable
   // Port A Data: 16-bit (each) input: Port A data
   .DIADI(DIADI),                  // 16-bit input: A port data/LSB data
   .DIPADIP(DIPADIP),              // 2-bit input: A port parity/LSB parity
  // Port B Address/Control Signals: 14-bit (each) input: Port B address and control signals (write port
   // when RAM_MODE="SDP")
   .ADDRBWRADDR(ADDRBWRADDR),      // 14-bit input: B port address/Write address
   .CLKBWRCLK(CLKBWRCLK),          // 1-bit input: B port clock/Write clock
   .ENBWREN(ENBWREN),              // 1-bit input: B port enable/Write enable
   .REGCEB(REGCEB),                // 1-bit input: B port register enable
   .RSTRAMB(RSTRAMB),              // 1-bit input: B port set/reset
   .RSTREGB(RSTREGB),              // 1-bit input: B port register set/reset
   .WEBWE(WEBWE),                  // 4-bit input: B port write enable/Write enable
   // Port B Data: 16-bit (each) input: Port B data
   .DIBDI(DIBDI),                  // 16-bit input: B port data/MSB data
   .DIPBDIP(DIPBDIP)               // 2-bit input: B port parity/MSB parity
);

// End of RAMB18E1_inst instantiation
```

For More Information

- See the *7 Series FPGAs Memory Resources User Guide* (UG473).

RAMB36E1

Primitive: 36K-bit Configurable Synchronous Block RAM



Introduction

7 series devices contain several block RAM memories that can be configured as FIFOs, automatic error correction RAM, or general-purpose 36 Kb or 18 Kb RAM/ROM memories. These block RAM memories offer fast and flexible storage of large amounts of on-chip data. The RAMB36E1 allows access to the block RAM in the 36 Kb configuration. This element can be cascaded to create a larger ram. This element can be configured and used as a 1-bit wide by 32K deep to a 36-bit wide by 1K deep true dual port RAM. This element can also be configured as a 72-bit wide

by 512 deep simple dual port RAM. Both read and write operations are fully synchronous to the supplied clock(s) to the component. However, the READ and WRITE ports can operate fully independent and asynchronous to each other, accessing the same memory array. When configured in the wider data width modes, byte-enable write operations are possible, and an optional output register can be used to reduce the clock-to-out times of the RAM. Error detection and correction circuitry can also be enabled to uncover and rectify possible memory corruptions.

Port Descriptions

Port	Direction	Width	Function
ADDRARDADDR<15:0>	Input	16	Port A address input bus/Read address input bus.
ADDRBWRADDR<15:0>	Input	16	Port B address input bus/Write address input bus.
CASCADEINA	Input	1	Port A cascade input. Never use when RAM_MODE="SDP".
CASCADEINB	Input	1	Port B cascade input. Never use when RAM_MODE="SDP".
CASCADEOUTA	Output	1	Port A cascade output. Never use when RAM_MODE="SDP".
CASCADEOUTB	Output	1	Port B cascade output. Never use when RAM_MODE="SDP".
CLKARDCLK	Input	1	Rising edge port A clock input/Read clock input.
CLKBWRCLK	Input	1	Rising edge port B clock input/Write clock input.
DBITERR	Output	1	Status output from ECC function to indicate a double bit error was detected. EN_ECC_READ needs to be TRUE to use this functionality. Not used when RAM_MODE="TDP".
DIADI<31:0>	Input	32	Port A data input bus/Data input bus addressed by WRADDR. When RAM_MODE="SDP", DIADI is the logical DI<31:0>.
DIBDI<31:0>	Input	32	Port B data input bus/Data input bus addressed by WRADDR. When RAM_MODE="SDP", DIBDI is the logical DI<63:32>.
DIPADIP<3:0>	Input	4	Port A parity data input bus/Data parity input bus addressed by WRADDR. When RAM_MODE="SDP", DIPADIP is the logical DIP<3:0>.
DIPBDIP<3:0>	Input	4	Port B parity data input bus/Data parity input bus addressed by WRADDR. When RAM_MODE="SDP", DIPBDIP is the logical DIP<7:4>.
DOADO<31:0>	Output	32	Port A data output bus/Data output bus addressed by RDADDR. When RAM_MODE="SDP", DOADO is the logical DO<31:0>.
DOBDO<31:0>	Output	32	Port B data output bus/Data output bus addressed by RDADDR. When RAM_MODE="SDP", DOBDO is the logical DO<63:32>.
DOPADOP<3:0>	Output	4	Port A parity data output bus/Data parity output bus addressed by RDADDR. When RAM_MODE="SDP", DOPADOP is the logical DOP<3:0>.
DOPBDOP<3:0>	Output	4	Port B parity data output bus/Data parity output bus addressed by RDADDR. When RAM_MODE="SDP", DOPBDOP is the logical DOP<7:4>.
ECCPARITY<7:0>	Output	8	8-bit data generated by the ECC encoder used by the ECC decoder for memory error detection and correction. Not used if RAM_MODE="TDP".

Port	Direction	Width	Function
ENARDEN	Input	1	Port A RAM enable/Read enable.
ENBWREN	Input	1	Port B RAM enable/Write enable.
INJECTDBITERR	Input	1	Inject a double bit error if ECC feature is used.
INJECTSBITERR	Input	1	Inject a single bit error if ECC feature is used.
RDADDRECC<8:0>	Output	9	ECC read address. Not used when RAM_MODE="TDP".
REGCEAREGCE	Input	1	Port A output register clock enable input/Output register clock enable input (valid only when DO_REG=1).
REGCEB	Input	1	Port B output register clock enable (valid only when DO_REG=1 and RAM_MODE="TDP").
RSTRAMARSTRAM	Input	1	Synchronous data latch set/reset to value indicated by SRVAL_A. RSTRAMARSTRAM sets/resets the BRAM data output latch when DO_REG=0 or 1. If DO_REG=1 there is a cycle of latency between the internal data latch node that is reset by RSTRAMARSTRAM and the DO output of the BRAM. This signal resets port A RAM output when RAM_MODE="TDP" and the entire RAM output when RAM_MODE="SDP".
RSTRAMB	Input	1	Synchronous data latch set/reset to value indicated by SRVAL_B. RSTRAMB sets/resets the BRAM data output latch when DO_REG=0 or 1. If DO_REG=1 there is a cycle of latency between the internal data latch node that is reset by RSTRAMB and the DO output of the BRAM. Not used when RAM_MODE="SDP".
RSTREGARSTREG	Input	1	Synchronous output register set/reset to value indicated by SRVAL_A. RSTREGARSTREG sets/resets the output register when DO_REG=1. RSTREG_PRIORITY_A determines if this signal gets priority over REGCEAREGCE. This signal resets port A output when RAM_MODE="TDP" and the entire output port when RAM_MODE="SDP".
RSTREGB	Input	1	Synchronous output register set/reset to value indicated by SRVAL_B. RSTREGB sets/resets the output register when DO_REG=1. RSTREG_PRIORITY_B determines if this signal gets priority over REGCEB. Not used when RAM_MODE="SDP".
SBITERR	Output	1	Status output from ECC function to indicate a single bit error was detected. EN_ECC_READ needs to be TRUE to use this functionality. Not used when RAM_MODE="TDP".
WEA<3:0>	Input	4	Port A byte-wide write enable. Not used when RAM_MODE="SDP". See User Guide for WEA mapping for different port widths.
WEBWE<7:0>	Input	8	Port B byte-wide write enable/Write enable. See User Guide for WEBWE mapping for different port widths.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	Yes
Macro support	Yes

Available Attributes

Attribute	Type	Allowed Values	Default	Description
RDADDR _COLLISION _HWCONFIG	STRING	"DELAYED_WRITE", "PERFORMANCE"	"DELAYED _WRITE"	When set to "PERFORMANCE" allows for higher clock performance (frequency) in READ_FIRST mode. If using the same clock on both ports of the RAM with "PERFORMANCE" mode, the address overlap collision rules apply where in "DELAYED_WRITE" mode, you can safely use the BRAM without incurring collisions.
SIM_COLLISION _CHECK	STRING	"ALL", "GENERATE_X_ONLY", "NONE", "WARNING_ONLY"	"ALL"	Allows modification of the simulation behavior so that if a memory collision occurs. • "ALL" = warning produced and affected outputs/memory go unknown (X) • "WARNING_ONLY" = warning produced and affected outputs/memory retain last value • "GENERATE_X_ONLY" = no warning and affected outputs/memory go unknown (X) • "NONE" = no warning and affected outputs/memory retain last value ***Note***: Use this setting carefully. Setting it to a value other than "ALL" can mask design problems during simulation.
DOA_REG, DOB_REG	DECIMAL	0, 1	0	A value of 1 enables the output registers to the RAM, which gives you quicker clock-to-out from the RAM at the expense of an added clock cycle of read latency. A value of 0 allows a read-in-one clock cycle but will result in slower clock-to-out timing. The number of registers activated is the same as the port width and includes parity bits. In SDP mode, DOA_REG and DOB_REG should always be set to the same value.
EN_ECC_READ	BOOLEAN	FALSE, TRUE	FALSE	Enable the ECC decoder circuitry.
EN_ECC_WRITE	BOOLEAN	FALSE, TRUE	FALSE	Enable the ECC encoder circuitry.
INIT_A, INIT_B	HEX	36 bit HEX	All zeros	Specifies the initial value on the port output after configuration. In SDP mode, INIT_A and INIT_B should always be set to the same value.
INIT_00 to INIT_7F	HEX	256 bit HEX	All zeros	Allows specification of the initial contents of the 32 Kb data memory array.
INIT_FILE	STRING	String representing file name and location	None	File name of file used to specify initial RAM contents.
INITP_00 to INITP_0F	HEX	256 bit HEX	All zeros	Allows specification of the initial contents of the 4 Kb parity data memory array.
RAM_EXTENSION _A, RAM_EXTENSION _B	STRING	"NONE", "LOWER", "UPPER"	"NONE"	Selects cascade mode. If not cascading two block RAMs to form a 64K x 1 RAM set to "NONE". If cascading RAMs, set to either "UPPER" or "LOWER" to indicate relative RAM location for proper configuration of the RAM. Not used if RAM_MODE="SDP".

Attribute	Type	Allowed Values	Default	Description
RAM_MODE	STRING	"TDP", "SDP"	"TDP"	Selects simple dual port (SDP) or true dual port (TDP) mode.
READ_WIDTH_A, READ_WIDTH_B, WRITE_WIDTH_A, WRITE_WIDTH_B	DECIMAL	0, 1, 2, 4, 9, 18, 36, 72	0	Specifies the desired data width for a read/ write on port A/B, including parity bits. This value must be 0 if the port is not used. Otherwise, it should be set to the desired port width.
RSTREG_PRIORITY _A, RSTREG_PRIORITY _B	STRING	"RSTREG", "REGCE"	"RSTREG"	Selects register priority for "RSTREG" or "REGCE". In SDP mode, RSTREG_PRIORITY_A and RSTREG_PRIORITY_B should always be set to the same value.
SIM_DEVICE	STRING	"7SERIES"	"7SERIES"	Must be set to "7SERIES" in order to exhibit proper simulation behavior under all conditions.
SRVAL_A, SRVAL_B	HEX	36 bit HEX	All zeros	Specifies the output value of the RAM upon assertion of the synchronous reset (RSTREG) signal. In SDP mode, SRVAL_A and SRVAL_B should always be set to the same value.
WRITE_MODE_A, WRITE_MODE_B	STRING	"WRITE_FIRST", "NO_CHANGE", "READ_FIRST"	"WRITE _FIRST"	Specifies output behavior of the port being written to. • "WRITE_FIRST" = written value appears on output port of the RAM • "READ_FIRST" = previous RAM contents for that memory location appears on the output port • "NO_CHANGE" = previous value on the output port remains the same When RAM_MODE="SDP", WRITE_MODE can not be set to "NO_CHANGE". For simple dual port implementations, it is generally suggested to set WRITE_MODE to "READ_FIRST" if using the same clock on both ports and to set it to "WRITE_FIRST" if using different clocks. This generally yields an improved collision or address overlap behavior when using the BRAM in this configuration.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- RAMB36E1: 36K-bit Configurable Synchronous Block RAM
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

RAMB36E1_inst : RAMB36E1
generic map (
   -- Address Collision Mode: "PERFORMANCE" or "DELAYED_WRITE"
   RDADDR_COLLISION_HWCONFIG => "DELAYED_WRITE",
   -- Collision check: Values ("ALL", "WARNING_ONLY", "GENERATE_X_ONLY" or "NONE")
   SIM_COLLISION_CHECK => "ALL",
```

```
   -- DOA_REG, DOB_REG: Optional output register (0 or 1)
   DOA_REG => 0,
   DOB_REG => 0,
   EN_ECC_READ => FALSE,                                                    -- Enable ECC decoder,
                                                                               -- FALSE, TRUE
   EN_ECC_WRITE => FALSE,                                                   -- Enable ECC encoder,
                                                                               -- FALSE, TRUE
   -- INITP_00 to INITP_0F: Initial contents of the parity memory array
   INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_0F => X"0000000000000000000000000000000000000000000000000000000000000000",
   -- INIT_00 to INIT_7F: Initial contents of the data memory array
   INIT_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_07 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_08 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_09 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_0F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_10 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_11 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_12 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_13 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_14 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_15 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_16 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_17 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_18 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_19 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_1F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_20 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_21 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_22 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_23 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_24 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_25 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_26 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_27 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_28 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_29 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_2F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_30 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_31 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_32 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_33 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_34 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_35 => X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
   INIT_36 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_37 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_38 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_39 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_40 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_41 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_42 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_43 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_44 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_45 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_46 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_47 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_48 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_49 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_4A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_4B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_4C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_4D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_4E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_4F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_50 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_51 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_52 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_53 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_54 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_55 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_56 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_57 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_58 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_59 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_5F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_60 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_61 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_62 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_63 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_64 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_65 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_66 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_67 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_68 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_69 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_6F => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_70 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_71 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_72 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_73 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_74 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_75 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_76 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_77 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_78 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_79 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7A => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7B => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7C => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7D => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_7F => X"0000000000000000000000000000000000000000000000000000000000000000",
   -- INIT_A, INIT_B: Initial values on output ports
   INIT_A => X"000000000",
   INIT_B => X"000000000",
   -- Initialization File: RAM initialization file
   INIT_FILE => "NONE",
```

```
   -- RAM Mode: "SDP" or "TDP"
   RAM_MODE => "TDP",
   -- RAM_EXTENSION_A, RAM_EXTENSION_B: Selects cascade mode ("UPPER", "LOWER", or "NONE")
   RAM_EXTENSION_A => "NONE",
   RAM_EXTENSION_B => "NONE",
   -- READ_WIDTH_A/B, WRITE_WIDTH_A/B: Read/write width per port
   READ_WIDTH_A => 0,                                                              -- 0-72
   READ_WIDTH_B => 0,                                                              -- 0-36
   WRITE_WIDTH_A => 0,                                                             -- 0-36
   WRITE_WIDTH_B => 0,                                                             -- 0-72
   -- RSTREG_PRIORITY_A, RSTREG_PRIORITY_B: Reset or enable priority ("RSTREG" or "REGCE")
   RSTREG_PRIORITY_A => "RSTREG",
   RSTREG_PRIORITY_B => "RSTREG",
   -- SRVAL_A, SRVAL_B: Set/reset value for output
   SRVAL_A => X"000000000",
   SRVAL_B => X"000000000",
   -- Simulation Device: Must be set to "7SERIES" for simulation behavior
   SIM_DEVICE => "7SERIES",
   -- WriteMode: Value on output upon a write ("WRITE_FIRST", "READ_FIRST", or "NO_CHANGE")
   WRITE_MODE_A => "WRITE_FIRST",
   WRITE_MODE_B => "WRITE_FIRST"
)
port map (
   -- Cascade Signals: 1-bit (each) output: BRAM cascade ports (to create 64kx1)
   CASCADEOUTA => CASCADEOUTA,     -- 1-bit output: A port cascade
   CASCADEOUTB => CASCADEOUTB,     -- 1-bit output: B port cascade
   -- ECC Signals: 1-bit (each) output: Error Correction Circuitry ports
   DBITERR => DBITERR,             -- 1-bit output: Double bit error status
   ECCPARITY => ECCPARITY,         -- 8-bit output: Generated error correction parity
   RDADDRECC => RDADDRECC,         -- 9-bit output: ECC read address
   SBITERR => SBITERR,             -- 1-bit output: Single bit error status
   -- Port A Data: 32-bit (each) output: Port A data
   DOADO => DOADO,                 -- 32-bit output: A port data/LSB data
   DOPADOP => DOPADOP,             -- 4-bit output: A port parity/LSB parity
   -- Port B Data: 32-bit (each) output: Port B data
   DOBDO => DOBDO,                 -- 32-bit output: B port data/MSB data
   DOPBDOP => DOPBDOP,             -- 4-bit output: B port parity/MSB parity
   -- Cascade Signals: 1-bit (each) input: BRAM cascade ports (to create 64kx1)
   CASCADEINA => CASCADEINA,       -- 1-bit input: A port cascade
   CASCADEINB => CASCADEINB,       -- 1-bit input: B port cascade
   -- ECC Signals: 1-bit (each) input: Error Correction Circuitry ports
   INJECTDBITERR => INJECTDBITERR, -- 1-bit input: Inject a double bit error
   INJECTSBITERR => INJECTSBITERR, -- 1-bit input: Inject a single bit error
   -- Port A Address/Control Signals: 16-bit (each) input: Port A address and control signals (read port
   -- when RAM_MODE="SDP")
   ADDRARDADDR => ADDRARDADDR,     -- 16-bit input: A port address/Read address
   CLKARDCLK => CLKARDCLK,         -- 1-bit input: A port clock/Read clock
   ENARDEN => ENARDEN,             -- 1-bit input: A port enable/Read enable
   REGCEAREGCE => REGCEAREGCE,     -- 1-bit input: A port register enable/Register enable
   RSTRAMARSTRAM => RSTRAMARSTRAM, -- 1-bit input: A port set/reset
   RSTREGARSTREG => RSTREGARSTREG, -- 1-bit input: A port register set/reset
   WEA => WEA,                     -- 4-bit input: A port write enable
   -- Port A Data: 32-bit (each) input: Port A data
   DIADI => DIADI,                 -- 32-bit input: A port data/LSB data
   DIPADIP => DIPADIP,             -- 4-bit input: A port parity/LSB parity
   -- Port B Address/Control Signals: 16-bit (each) input: Port B address and control signals (write port
   -- when RAM_MODE="SDP")
   ADDRBWRADDR => ADDRBWRADDR,     -- 16-bit input: B port address/Write address
   CLKBWRCLK => CLKBWRCLK,         -- 1-bit input: B port clock/Write clock
   ENBWREN => ENBWREN,             -- 1-bit input: B port enable/Write enable
   REGCEB => REGCEB,               -- 1-bit input: B port register enable
   RSTRAMB => RSTRAMB,             -- 1-bit input: B port set/reset
   RSTREGB => RSTREGB,             -- 1-bit input: B port register set/reset
   WEBWE => WEBWE,                 -- 8-bit input: B port write enable/Write enable
   -- Port B Data: 32-bit (each) input: Port B data
   DIBDI => DIBDI,                 -- 32-bit input: B port data/MSB data
   DIPBDIP => DIPBDIP              -- 4-bit input: B port parity/MSB parity
);

-- End of RAMB36E1_inst instantiation
```

Verilog Instantiation Template

```
// RAMB36E1: 36K-bit Configurable Synchronous Block RAM
//           7 Series
// Xilinx HDL Language Template, version 2020.1

RAMB36E1 #(
   // Address Collision Mode: "PERFORMANCE" or "DELAYED_WRITE"
   .RDADDR_COLLISION_HWCONFIG("DELAYED_WRITE"),
   // Collision check: Values ("ALL", "WARNING_ONLY", "GENERATE_X_ONLY" or "NONE")
   .SIM_COLLISION_CHECK("ALL"),
   // DOA_REG, DOB_REG: Optional output register (0 or 1)
   .DOA_REG(0),
   .DOB_REG(0),
   .EN_ECC_READ("FALSE"),                                                    // Enable ECC decoder,
                                                                                // FALSE, TRUE
   .EN_ECC_WRITE("FALSE"),                                                   // Enable ECC encoder,
                                                                                // FALSE, TRUE
   // INITP_00 to INITP_0F: Initial contents of the parity memory array
   .INITP_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_08(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_09(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INITP_0F(256'h0000000000000000000000000000000000000000000000000000000000000000),
   // INIT_00 to INIT_7F: Initial contents of the data memory array
   .INIT_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_08(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_09(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_0F(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_10(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_11(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_12(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_13(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_14(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_15(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_16(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_17(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_18(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_19(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1A(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1B(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1C(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1D(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1E(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_1F(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_20(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_21(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_22(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_23(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_24(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_25(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_26(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_27(256'h0000000000000000000000000000000000000000000000000000000000000000),
   .INIT_28(256'h0000000000000000000000000000000000000000000000000000000000000000),
```

```
    .INIT_29(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_2F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_30(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_31(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_32(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_33(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_34(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_35(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_36(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_37(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_38(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_39(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_3F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_40(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_41(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_42(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_43(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_44(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_45(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_46(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_47(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_48(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_49(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_4F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_50(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_51(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_52(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_53(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_54(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_55(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_56(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_57(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_58(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_59(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_5F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_60(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_61(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_62(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_63(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_64(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_65(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_66(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_67(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_68(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_69(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_6F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_70(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_71(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_72(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_73(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_74(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_75(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_76(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_77(256'h0000000000000000000000000000000000000000000000000000000000000000),
```

```
    .INIT_78(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_79(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_7A(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_7B(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_7C(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_7D(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_7E(256'h0000000000000000000000000000000000000000000000000000000000000000),
    .INIT_7F(256'h0000000000000000000000000000000000000000000000000000000000000000),
    // INIT_A, INIT_B: Initial values on output ports
    .INIT_A(36'h000000000),
    .INIT_B(36'h000000000),
    // Initialization File: RAM initialization file
    .INIT_FILE("NONE"),
    // RAM Mode: "SDP" or "TDP"
    .RAM_MODE("TDP"),
    // RAM_EXTENSION_A, RAM_EXTENSION_B: Selects cascade mode ("UPPER", "LOWER", or "NONE")
    .RAM_EXTENSION_A("NONE"),
    .RAM_EXTENSION_B("NONE"),
    // READ_WIDTH_A/B, WRITE_WIDTH_A/B: Read/write width per port
    .READ_WIDTH_A(0),                                                                  // 0-72
    .READ_WIDTH_B(0),                                                                  // 0-36
    .WRITE_WIDTH_A(0),                                                                 // 0-36
    .WRITE_WIDTH_B(0),                                                                 // 0-72
    // RSTREG_PRIORITY_A, RSTREG_PRIORITY_B: Reset or enable priority ("RSTREG" or "REGCE")
    .RSTREG_PRIORITY_A("RSTREG"),
    .RSTREG_PRIORITY_B("RSTREG"),
    // SRVAL_A, SRVAL_B: Set/reset value for output
    .SRVAL_A(36'h000000000),
    .SRVAL_B(36'h000000000),
    // Simulation Device: Must be set to "7SERIES" for simulation behavior
    .SIM_DEVICE("7SERIES"),
    // WriteMode: Value on output upon a write ("WRITE_FIRST", "READ_FIRST", or "NO_CHANGE")
    .WRITE_MODE_A("WRITE_FIRST"),
    .WRITE_MODE_B("WRITE_FIRST")
)
RAMB36E1_inst (
    // Cascade Signals: 1-bit (each) output: BRAM cascade ports (to create 64kx1)
    .CASCADEOUTA(CASCADEOUTA),      // 1-bit output: A port cascade
    .CASCADEOUTB(CASCADEOUTB),      // 1-bit output: B port cascade
    // ECC Signals: 1-bit (each) output: Error Correction Circuitry ports
    .DBITERR(DBITERR),              // 1-bit output: Double bit error status
    .ECCPARITY(ECCPARITY),          // 8-bit output: Generated error correction parity
    .RDADDRECC(RDADDRECC),          // 9-bit output: ECC read address
    .SBITERR(SBITERR),              // 1-bit output: Single bit error status
    // Port A Data: 32-bit (each) output: Port A data
    .DOADO(DOADO),                  // 32-bit output: A port data/LSB data
    .DOPADOP(DOPADOP),              // 4-bit output: A port parity/LSB parity
    // Port B Data: 32-bit (each) output: Port B data
    .DOBDO(DOBDO),                  // 32-bit output: B port data/MSB data
    .DOPBDOP(DOPBDOP),              // 4-bit output: B port parity/MSB parity
    // Cascade Signals: 1-bit (each) input: BRAM cascade ports (to create 64kx1)
    .CASCADEINA(CASCADEINA),        // 1-bit input: A port cascade
    .CASCADEINB(CASCADEINB),        // 1-bit input: B port cascade
    // ECC Signals: 1-bit (each) input: Error Correction Circuitry ports
    .INJECTDBITERR(INJECTDBITERR), // 1-bit input: Inject a double bit error
    .INJECTSBITERR(INJECTSBITERR), // 1-bit input: Inject a single bit error
   // Port A Address/Control Signals: 16-bit (each) input: Port A address and control signals (read port
    // when RAM_MODE="SDP")
    .ADDRARDADDR(ADDRARDADDR),      // 16-bit input: A port address/Read address
    .CLKARDCLK(CLKARDCLK),          // 1-bit input: A port clock/Read clock
    .ENARDEN(ENARDEN),              // 1-bit input: A port enable/Read enable
    .REGCEAREGCE(REGCEAREGCE),      // 1-bit input: A port register enable/Register enable
    .RSTRAMARSTRAM(RSTRAMARSTRAM), // 1-bit input: A port set/reset
    .RSTREGARSTREG(RSTREGARSTREG), // 1-bit input: A port register set/reset
    .WEA(WEA),                      // 4-bit input: A port write enable
    // Port A Data: 32-bit (each) input: Port A data
    .DIADI(DIADI),                  // 32-bit input: A port data/LSB data
    .DIPADIP(DIPADIP),              // 4-bit input: A port parity/LSB parity
   // Port B Address/Control Signals: 16-bit (each) input: Port B address and control signals (write port
    // when RAM_MODE="SDP")
    .ADDRBWRADDR(ADDRBWRADDR),      // 16-bit input: B port address/Write address
    .CLKBWRCLK(CLKBWRCLK),          // 1-bit input: B port clock/Write clock
    .ENBWREN(ENBWREN),              // 1-bit input: B port enable/Write enable
    .REGCEB(REGCEB),                // 1-bit input: B port register enable
    .RSTRAMB(RSTRAMB),              // 1-bit input: B port set/reset
    .RSTREGB(RSTREGB),              // 1-bit input: B port register set/reset
    .WEBWE(WEBWE),                  // 8-bit input: B port write enable/Write enable
    // Port B Data: 32-bit (each) input: Port B data
```

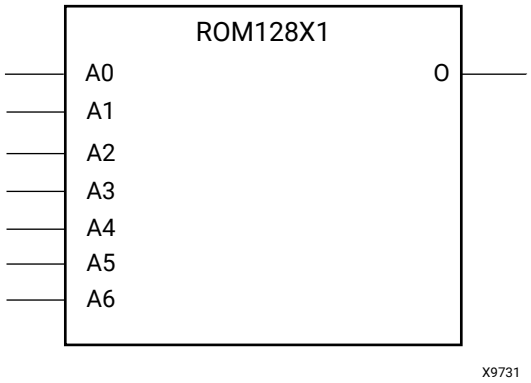
```
   .DIBDI(DIBDI),                 // 32-bit input: B port data/MSB data
   .DIPBDIP(DIPBDIP)              // 4-bit input: B port parity/MSB parity
);

// End of RAMB36E1_inst instantiation
```

For More Information

- See the *7 Series FPGAs Memory Resources User Guide* (UG473).

ROM128X1

Primitive: 128-Deep by 1-Wide ROM



Introduction

This design element is a 128-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 7-bit address (A6:A0). The ROM is initialized to a known value during configuration with the INIT parameter. The value consists of 32 hexadecimal digits that are written into the ROM from the most-significant digit A=FH to the least-significant digit A=0H.

An error occurs if INIT is not specified.

Logic Table

Input				Output
I0	**I1**	**I2**	**I3**	**O**
0	0	0	0	INIT(0)
0	0	0	1	INIT(1)
0	0	1	0	INIT(2)
0	0	1	1	INIT(3)
0	1	0	0	INIT(4)
0	1	0	1	INIT(5)
0	1	1	0	INIT(6)
0	1	1	1	INIT(7)
1	0	0	0	INIT(8)
1	0	0	1	INIT(9)
1	0	1	0	INIT(10)
1	0	1	1	INIT(11)
1	1	0	0	INIT(12)
1	1	0	1	INIT(13)

Input				Output
I0	**I1**	**I2**	**I3**	**O**
1	1	1	0	INIT(14)
1	1	1	1	INIT(15)

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 128-Bit Value	All zeros	Specifies the contents of the ROM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- ROM128X1: 128 x 1 Asynchronous Distributed (LUT) ROM
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

ROM128X1_inst : ROM128X1
generic map (
   INIT => X"00000000000000000000000000000000")
port map (
   O => O,    -- ROM output
   A0 => A0, -- ROM address[0]
   A1 => A1, -- ROM address[1]
   A2 => A2, -- ROM address[2]
   A3 => A3, -- ROM address[3]
   A4 => A4, -- ROM address[4]
   A5 => A5, -- ROM address[5]
   A6 => A6  -- ROM address[6]
);

-- End of ROM128X1_inst instantiation
```

Verilog Instantiation Template

```
// ROM128X1: 128 x 1 Asynchronous Distributed (LUT) ROM (Mapped to two SliceM LUT6s)
//           7 Series
// Xilinx HDL Language Template, version 2020.1

ROM128X1 #(
   .INIT(128'h00000000000000000000000000000000) // Contents of ROM
) ROM128X1_inst (
   .O(O),    // ROM output
   .A0(A0), // ROM address[0]
```

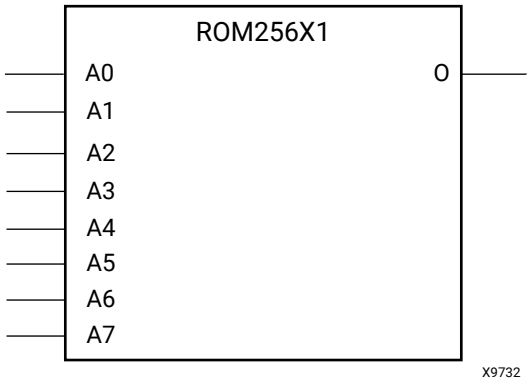
```
   .A1(A1), // ROM address[1]
   .A2(A2), // ROM address[2]
   .A3(A3), // ROM address[3]
   .A4(A4), // ROM address[4]
   .A5(A5), // ROM address[5]
   .A6(A6)  // ROM address[6]
);

// End of ROM128X1_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

ROM256X1

Primitive: 256-Deep by 1-Wide ROM



Introduction

This design element is a 256-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 8-bit address (A7:A0). The ROM is initialized to a known value during configuration with the INIT parameter. The value consists of 64 hexadecimal digits that are written into the ROM from the most-significant digit A=FH to the least-significant digit A=0H.

An error occurs if the INIT is not specified.

Logic Table

Input				Output
I0	**I1**	**I2**	**I3**	**O**
0	0	0	0	INIT(0)
0	0	0	1	INIT(1)
0	0	1	0	INIT(2)
0	0	1	1	INIT(3)
0	1	0	0	INIT(4)
0	1	0	1	INIT(5)
0	1	1	0	INIT(6)
0	1	1	1	INIT(7)
1	0	0	0	INIT(8)
1	0	0	1	INIT(9)
1	0	1	0	INIT(10)
1	0	1	1	INIT(11)
1	1	0	0	INIT(12)
1	1	0	1	INIT(13)

Input				Output
I0	**I1**	**I2**	**I3**	**O**
1	1	1	0	INIT(14)
1	1	1	1	INIT(15)

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 256-Bit Value	All zeros	Specifies the contents of the ROM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- ROM256X1: 256 x 1 Asynchronous Distributed (LUT) ROM
--           7 Series
-- Xilinx HDL Language Template, version 2020.1

ROM256X1_inst : ROM256X1
generic map (
   INIT => X"0000000000000000000000000000000000000000000000000000000000000000")
port map (
   O => O,    -- ROM output
   A0 => A0, -- ROM address[0]
   A1 => A1, -- ROM address[1]
   A2 => A2, -- ROM address[2]
   A3 => A3, -- ROM address[3]
   A4 => A4, -- ROM address[4]
   A5 => A5, -- ROM address[5]
   A6 => A6,  -- ROM address[6]
   A7 => A7  -- ROM address[7]
);

-- End of ROM256X1_inst instantiation
```

Verilog Instantiation Template

```
// ROM256X1: 256 x 1 Asynchronous Distributed (LUT) ROM (Mapped to four SliceM LUT6s)
//           7 Series
// Xilinx HDL Language Template, version 2020.1

ROM256X1 #(
   .INIT(256'h0000000000000000000000000000000000000000000000000000000000000000) // Contents of ROM
) ROM256X1_inst (
   .O(O),    // ROM output
```

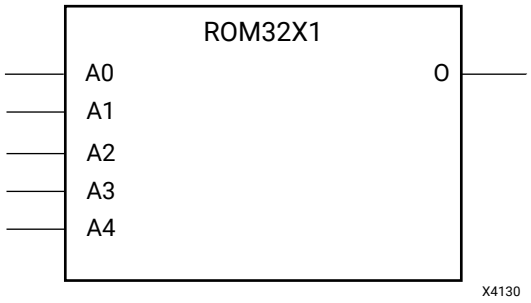
```
   .A0(A0), // ROM address[0]
   .A1(A1), // ROM address[1]
   .A2(A2), // ROM address[2]
   .A3(A3), // ROM address[3]
   .A4(A4), // ROM address[4]
   .A5(A5), // ROM address[5]
   .A6(A6), // ROM address[6]
   .A7(A7)  // ROM address[7]
);

// End of ROM256X1_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

ROM32X1

Primitive: 32-Deep by 1-Wide ROM



Introduction

This design element is a 32-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 5-bit address (A4:A0). The ROM is initialized to a known value during configuration with the INIT parameter. The value consists of eight hexadecimal digits that are written into the ROM from the most-significant digit A=1FH to the least-significant digit A=00H.

For example, INIT=10A78F39 produces the data stream: 0001 0000 1010 0111 1000 1111 0011 1001.

An error occurs if the INIT is not specified.

Logic Table

Input				Output
I0	**I1**	**I2**	**I3**	**O**
0	0	0	0	INIT(0)
0	0	0	1	INIT(1)
0	0	1	0	INIT(2)
0	0	1	1	INIT(3)
0	1	0	0	INIT(4)
0	1	0	1	INIT(5)
0	1	1	0	INIT(6)
0	1	1	1	INIT(7)
1	0	0	0	INIT(8)
1	0	0	1	INIT(9)
1	0	1	0	INIT(10)
1	0	1	1	INIT(11)
1	1	0	0	INIT(12)

Input				Output
I0	**I1**	**I2**	**I3**	**O**
1	1	0	1	INIT(13)
1	1	1	0	INIT(14)
1	1	1	1	INIT(15)

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 32-Bit Value	All zeros	Specifies the contents of the ROM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- ROM32X1: 32 x 1 Asynchronous Distributed (LUT) ROM
--          7 Series
-- Xilinx HDL Language Template, version 2020.1

ROM32X1_inst : ROM32X1
generic map (
   INIT => X"00000000")
port map (
   O => O,   -- ROM output
   A0 => A0, -- ROM address[0]
   A1 => A1, -- ROM address[1]
   A2 => A2, -- ROM address[2]
   A3 => A3, -- ROM address[3]
   A4 => A4  -- ROM address[4]
);
-- End of ROM32X1_inst instantiation
```

Verilog Instantiation Template

```
// ROM32X1: 32 x 1 Asynchronous Distributed (LUT) ROM (Mapped to a SliceM LUT6)
//          7 Series
// Xilinx HDL Language Template, version 2020.1

ROM32X1 #(
   .INIT(32'h00000000) // Contents of ROM
) ROM32X1_inst (
   .O(O),   // ROM output
   .A0(A0), // ROM address[0]
   .A1(A1), // ROM address[1]
```

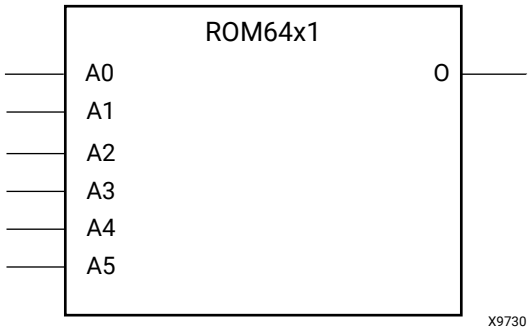
```
   .A2(A2), // ROM address[2]
   .A3(A3), // ROM address[3]
   .A4(A4)  // ROM address[4]
);

// End of ROM32X1_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

ROM64X1

Primitive: 64-Deep by 1-Wide ROM



Introduction

This design element is a 64-word by 1-bit read-only memory. The data output (O) reflects the word selected by the 6-bit address (A5:A0). The ROM is initialized to a known value during configuration with the INIT parameter. The value consists of 16 hexadecimal digits that are written into the ROM from the most-significant digit A=FH to the least-significant digit A=0H.

An error occurs if INIT is not specified.

Logic Table

Input				Output
I0	**I1**	**I2**	**I3**	**O**
0	0	0	0	INIT(0)
0	0	0	1	INIT(1)
0	0	1	0	INIT(2)
0	0	1	1	INIT(3)
0	1	0	0	INIT(4)
0	1	0	1	INIT(5)
0	1	1	0	INIT(6)
0	1	1	1	INIT(7)
1	0	0	0	INIT(8)
1	0	0	1	INIT(9)
1	0	1	0	INIT(10)
1	0	1	1	INIT(11)
1	1	0	0	INIT(12)
1	1	0	1	INIT(13)
1	1	1	0	INIT(14)

Input				Output
I0	**I1**	**I2**	**I3**	**O**
1	1	1	1	INIT(15)

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 64-Bit Value	All zeros	Specifies the contents of the ROM.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- ROM64X1: 64 x 1 Asynchronous Distributed (LUT) ROM
--          7 Series
-- Xilinx HDL Language Template, version 2020.1

ROM64X1_inst : ROM64X1
generic map (
   INIT => X"0000000000000000")
port map (
   O => O,    -- ROM output
   A0 => A0, -- ROM address[0]
   A1 => A1, -- ROM address[1]
   A2 => A2, -- ROM address[2]
   A3 => A3, -- ROM address[3]
   A4 => A4, -- ROM address[4]
   A5 => A5  -- ROM address[5]
);

-- End of ROM64X1_inst instantiation
```

Verilog Instantiation Template

```
// ROM64X1: 64 x 1 Asynchronous Distributed (LUT) ROM (Mapped to a SliceM LUT6)
//          7 Series
// Xilinx HDL Language Template, version 2020.1

ROM64X1 #(
   .INIT(64'h0000000000000000) // Contents of ROM
) ROM64X1_inst (
   .O(O),    // ROM output
   .A0(A0), // ROM address[0]
   .A1(A1), // ROM address[1]
   .A2(A2), // ROM address[2]
   .A3(A3), // ROM address[3]
```

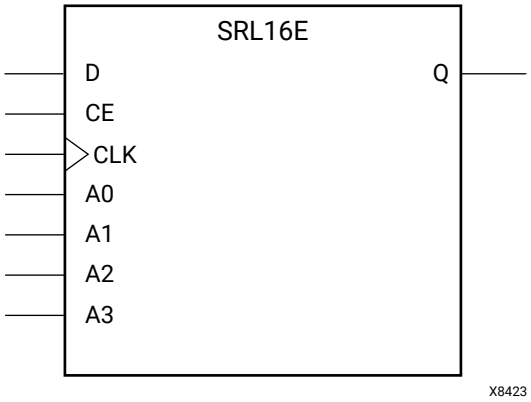
```
   .A4(A4), // ROM address[4]
   .A5(A5)  // ROM address[5]
);

// End of ROM64X1_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

SRL16E

Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Clock Enable



Introduction

This design element is a shift register look-up table (LUT). The inputs A3, A2, A1, and A0 select the depth of the shift register. The shift register can be of a fixed, static depth or it can be dynamically adjusted.

To create a fixed-depth shift register: Drive the A3 through A0 inputs with static values. The depth of the shift register can vary from 1 bit to 16 bits, as determined by the following formula:

Depth = (8 x A3) + (4 x A2) + (2 x A1) + A0 + 1

If A3, A2, A1, and A0 are all zeros (0000), the shift register is one bit deep. If they are all ones (1111), it is 16 bits deep.

To change the depth of the shift register dynamically: Change the values driving the A3 through A0 inputs. For example, if A2, A1, and A0 are all ones (111) and A3 toggles between a one (1) and a zero (0), the depth of the shift register changes from 16 bits to 8 bits. Internally, the depth of the shift register is always 16 bits and the input lines A3 through A0 select which of the 16 bits reach the output. The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

When CE is High, the data (D) is loaded into the first bit of the shift register during the clock (CLK) transition. During subsequent clock transitions, when CE is High, data shifts to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached. When CE is Low, the register ignores clock transitions and retains current data within the shift register.

Two SLR16E components may be placed within the same LUT within a CLBM as long as they have the same clock, clock enable and depth selection address signals as well as the same IS_CLK_INVERTED attribute value. This allows up to 16 SRL16E components to be placed into a single CLB. Optionally, LUTNM or HLUTNMs may be placed on two SRL16E components to specify specific grouping within a LUT.

***Note*:** When using SRLs with initialized values, you should use safe clock start-up techniques to ensure the initialized data is not corrupted upon completion of configuration. Refer to UG949: UltraFast Design Methodology Guide for details on controlling and synchronizing clock startup.

Logic Table

Inputs				Output
Am	**CE**	**CLK**	**D**	**Q**
Am	0	X	X	Q(Am)
Am	1	↑	D	Q(Am - 1)
m= 0, 1, 2, 3				

Port Descriptions

Port	Direction	Width	Function
CE	Input	1	Active-High clock enable
CLK	Input	1	Shift register clock. Polarity is determined by the IS_CLK_INVERTED attribute.
D	Input	1	SRL data input.
Q	Output	1	SRL data output.
A0	Input	1	The value placed on the A0 - A3 inputs specifies the shift register depth. ***Depth = (8 x A3) + (4 x A2) + (2 x A1) + A0 + 1***
A1	Input	1	The value placed on the A0 - A3 inputs specifies the shift register depth. ***Depth = (8 x A3) + (4 x A2) + (2 x A1) + A0 + 1***
A2	Input	1	The value placed on the A0 - A3 inputs specifies the shift register depth. ***Depth = (8 x A3) + (4 x A2) + (2 x A1) + A0 + 1***
A3	Input	1	The value placed on the A0 - A3 inputs specifies the shift register depth. ***Depth = (8 x A3) + (4 x A2) + (2 x A1) + A0 + 1***

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 16-Bit Value	All zeros	Specifies the initial contents in the shift register upon completion of configuration.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- SRL16E: 16-bit shift register LUT with clock enable operating on posedge of clock (Mapped to SliceM LUT6)
--         7 Series
-- Xilinx HDL Language Template, version 2020.1

SRL16E_inst : SRL16E
generic map (
   INIT => X"0000")
port map (
   Q => Q,         -- SRL data output
   A0 => A0,       -- Select[0] input
   A1 => A1,       -- Select[1] input
   A2 => A2,       -- Select[2] input
   A3 => A3,       -- Select[3] input
   CE => CE,       -- Clock enable input
   CLK => CLK,     -- Clock input
   D => D          -- SRL data input
);

-- End of SRL16E_inst instantiation
```

Verilog Instantiation Template

```
// SRL16E: 16-bit shift register LUT with clock enable operating
//         on posedge of clock (Mapped to a SliceM LUT6)
//         7 Series
// Xilinx HDL Language Template, version 2020.1

SRL16E #(
   .INIT(16'h0000) // Initial Value of Shift Register
) SRL16E_inst (
   .Q(Q),       // SRL data output
   .A0(A0),     // Select[0] input
   .A1(A1),     // Select[1] input
   .A2(A2),     // Select[2] input
   .A3(A3),     // Select[3] input
   .CE(CE),     // Clock enable input
```

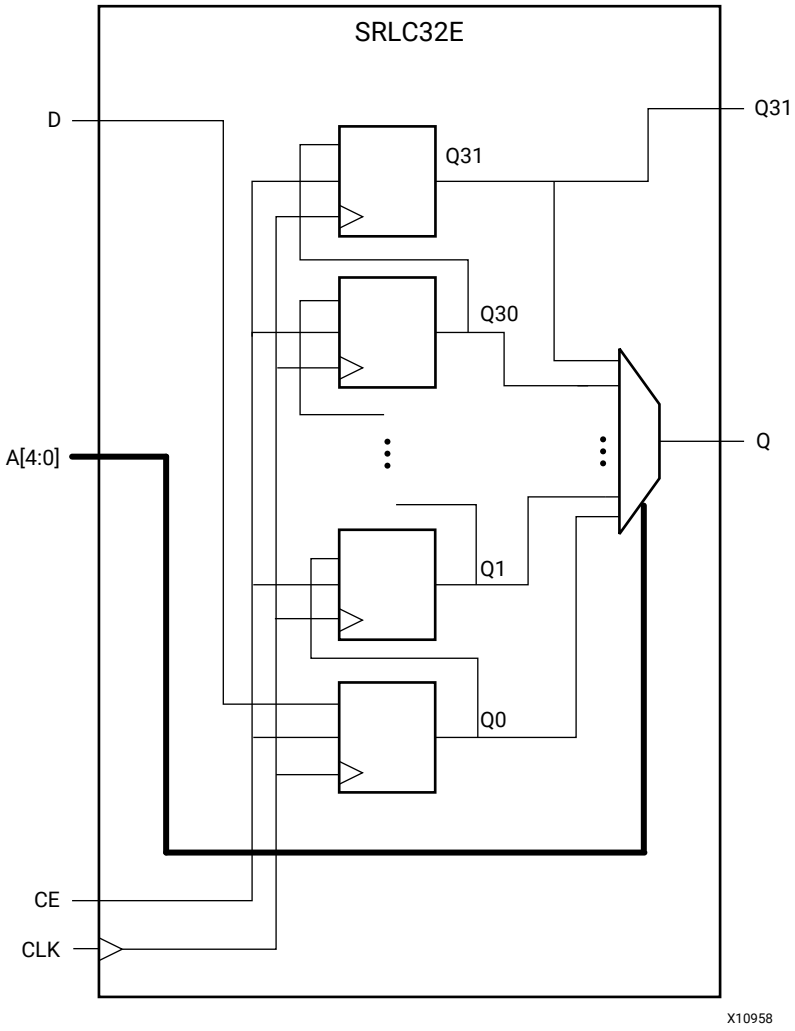
```
   .CLK(CLK),   // Clock input
   .D(D)        // SRL data input
);

// End of SRL16E_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

SRLC32E

Primitive: 32 Clock Cycle, Variable Length Shift Register Look-Up Table (LUT) with Clock Enable



Introduction

This design element is a shift register look-up table (LUT). The inputs A4, A3, A2, A1, and A0 select the depth of the shift register.

The shift register can be of a fixed, static depth or it can be dynamically adjusted.

To create a fixed-depth shift register: Drive the A4 through A0 inputs with static values. The depth of the shift register can vary from 1 bit to 32 bits, as determined by the following formula:

Depth = (16 x A4) + (8 x A3) + (4 x A2) + (2 x A1) + A0 + 1

If A4, A3, A2, A1, and A0 are all zeros (00000), the shift register is one bit deep. If they are all ones (11111), it is 32 bits deep.

To change the depth of the shift register dynamically: Change the values driving the A4 through A0 inputs. For example, if A3, A2, A1, and A0 are all ones (1111) and A4 toggles between a one (1) and a zero (0), the depth of the shift register changes from 32 bits to 16 bits. Internally, the depth of the shift register is always 32 bits and the input lines A4 through A0 select which of the 32 bits reach the output. The shift register LUT contents are initialized by assigning a eight-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of eight zeros (00000000) so that the shift register LUT is cleared during configuration.

When CE is High, the data (D) is loaded into the first bit of the shift register during the clock (CLK) transition. During subsequent clock transitions, when CE is High, data shifts to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached. When CE is Low, the register ignores clock transitions and retains current data within the shift register.

Two or more SLRC32E components may be cascaded to create deeper than 32-bit shift registers. To do so, connect the Q31 output of one SRLC32E component to the D input of another.

***Note*:** When using SRLs with initialized values, you should use safe clock start-up techniques to ensure the initialized data is not corrupted upon completion of configuration. Refer to UG949: UltraFast Design Methodology Guide for details on controlling and synchronizing clock startup.

Port Descriptions

Port	Direction	Width	Function
A<4:0>	Input	5	The value placed on the A0 - A3 inputs specifies the shift register depth. ***Depth = (16 x A4) + (8 x A3) + (4 x A2) + (2 x A1) + A0 + 1***
CE	Input	1	Active-High clock enable.
CLK	Input	1	Shift register clock. Polarity is determined by the IS_CLK_INVERTED attribute.
D	Input	1	SRL data input.
Q	Output	1	SRL data output.
Q31	Output	1	SRL data output used to connect more than one SRLC32E component to form deeper than 32-bit shift registers.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 32-Bit Value	All zeros	Specifies the initial contents in the shift register upon completion of configuration.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- SRLC32E: 32-bit variable length shift register LUT
--          with clock enable (Mapped to a SliceM LUT6)
--          7 Series
-- Xilinx HDL Language Template, version 2020.1

SRLC32E_inst : SRLC32E
generic map (
   INIT => X"00000000")
port map (
   Q => Q,         -- SRL data output
   Q31 => Q31,     -- SRL cascade output pin
   A => A,         -- 5-bit shift depth select input
   CE => CE,       -- Clock enable input
   CLK => CLK,     -- Clock input
   D => D          -- SRL data input
);

-- End of SRLC32E_inst instantiation
```

Verilog Instantiation Template

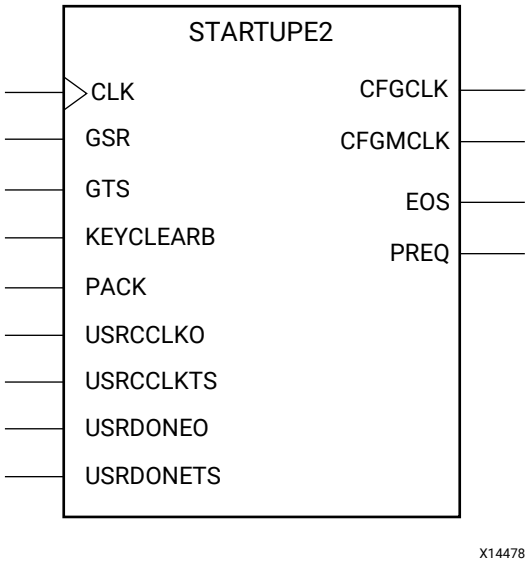
```
// SRLC32E: 32-bit variable length cascadable shift register LUT  (Mapped to a SliceM LUT6)
//          with clock enable
//          7 Series
// Xilinx HDL Language Template, version 2020.1

SRLC32E #(
   .INIT(32'h00000000) // Initial Value of Shift Register
) SRLC32E_inst (
   .Q(Q),      // SRL data output
   .Q31(Q31), // SRL cascade output pin
   .A(A),      // 5-bit shift depth select input
   .CE(CE),    // Clock enable input
   .CLK(CLK), // Clock input
   .D(D)       // SRL data input
);

// End of SRLC32E_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configurable Logic Block User Guide* (UG474).

STARTUPE2

Primitive: STARTUP Block



Introduction

This design element is used to interface device pins and logic to the global asynchronous set/reset (GSR) signal, the global 3-state (GTS) dedicated routing or the internal configuration signals or a few of the dedicated configuration pins.

Port Descriptions

Port	Direction	Width	Function
CFGCLK	Output	1	Configuration main clock output.
CFGMCLK	Output	1	Configuration internal oscillator clock output.
CLK	Input	1	User start-up clock input. For Spartan-7 7S6 and 7S15 devices, the user-defined CCLK (UserClk value for the BITSTREAM.STARTUP.STARTUPCLK property) for the start-up sequence is not supported.
EOS	Output	1	Active-High output signal indicating the End Of Startup.
GSR	Input	1	Global Set/Reset input (GSR cannot be used for the port name).
GTS	Input	1	Global 3-state input (GTS cannot be used for the port name).
KEYCLEARB	Input	1	Clear AES Decrypter Key input from battery-backed RAM (BBRAM).
PACK	Input	1	PROGRAM acknowledge input.
PREQ	Output	1	PROGRAM request to fabric output.

Port	Direction	Width	Function
USRCCLKO	Input	1	User CCLK input. For Zynq-7000 devices, this input must be tied to GND.
USRCCLKTS	Input	1	User CCLK 3-state enable input. For Zynq-7000 devices, this input must be tied to VCC.
USRDONEO	Input	1	User DONE pin output control.
USRDONETS	Input	1	User DONE 3-state enable output.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
PROG_USR	STRING	"FALSE", "TRUE"	"FALSE"	Activate program event security feature. Requires encrypted bitstreams.
SIM_CCLK_FREQ	FLOAT (nS)	0.0 to 10.0	0.0	Set the Configuration Clock Frequency (ns) for simulation.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- STARTUPE2: STARTUP Block
--            7 Series
-- Xilinx HDL Language Template, version 2020.1

STARTUPE2_inst : STARTUPE2
generic map (
   PROG_USR => "FALSE",  -- Activate program event security feature. Requires encrypted bitstreams.
   SIM_CCLK_FREQ => 0.0  -- Set the Configuration Clock Frequency(ns) for simulation.
)
port map (
   CFGCLK => CFGCLK,        -- 1-bit output: Configuration main clock output
   CFGMCLK => CFGMCLK,      -- 1-bit output: Configuration internal oscillator clock output
   EOS => EOS,              -- 1-bit output: Active high output signal indicating the End Of Startup.
   PREQ => PREQ,            -- 1-bit output: PROGRAM request to fabric output
   CLK => CLK,              -- 1-bit input: User start-up clock input
   GSR => GSR,            -- 1-bit input: Global Set/Reset input (GSR cannot be used for the port name)
   GTS => GTS,            -- 1-bit input: Global 3-state input (GTS cannot be used for the port name)
   KEYCLEARB => KEYCLEARB, -- 1-bit input: Clear AES Decrypter Key input from Battery-Backed RAM (BBRAM)
   PACK => PACK,            -- 1-bit input: PROGRAM acknowledge input
   USRCCLKO => USRCCLKO,    -- 1-bit input: User CCLK input
                            -- For Zynq-7000 devices, this input must be tied to GND
   USRCCLKTS => USRCCLKTS, -- 1-bit input: User CCLK 3-state enable input
                            -- For Zynq-7000 devices, this input must be tied to VCC
```

```
   USRDONEO => USRDONEO,    -- 1-bit input: User DONE pin output control
   USRDONETS => USRDONETS   -- 1-bit input: User DONE 3-state enable output
);

-- End of STARTUPE2_inst instantiation
```

Verilog Instantiation Template

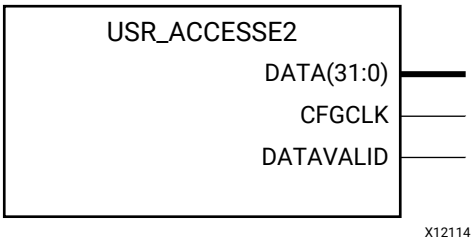
```
// STARTUPE2: STARTUP Block
//            7 Series
// Xilinx HDL Language Template, version 2020.1

STARTUPE2 #(
   .PROG_USR("FALSE"),  // Activate program event security feature. Requires encrypted bitstreams.
   .SIM_CCLK_FREQ(0.0)  // Set the Configuration Clock Frequency(ns) for simulation.
)
STARTUPE2_inst (
   .CFGCLK(CFGCLK),        // 1-bit output: Configuration main clock output
   .CFGMCLK(CFGMCLK),      // 1-bit output: Configuration internal oscillator clock output
   .EOS(EOS),              // 1-bit output: Active high output signal indicating the End Of Startup.
   .PREQ(PREQ),            // 1-bit output: PROGRAM request to fabric output
   .CLK(CLK),              // 1-bit input: User start-up clock input
   .GSR(GSR),            // 1-bit input: Global Set/Reset input (GSR cannot be used for the port name)
   .GTS(GTS),            // 1-bit input: Global 3-state input (GTS cannot be used for the port name)
   .KEYCLEARB(KEYCLEARB), // 1-bit input: Clear AES Decrypter Key input from Battery-Backed RAM (BBRAM)
   .PACK(PACK),            // 1-bit input: PROGRAM acknowledge input
   .USRCCLKO(USRCCLKO),    // 1-bit input: User CCLK input
                           // For Zynq-7000 devices, this input must be tied to GND
   .USRCCLKTS(USRCCLKTS), // 1-bit input: User CCLK 3-state enable input
                           // For Zynq-7000 devices, this input must be tied to VCC
   .USRDONEO(USRDONEO),    // 1-bit input: User DONE pin output control
   .USRDONETS(USRDONETS)   // 1-bit input: User DONE 3-state enable output
);

// End of STARTUPE2_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configuration Guide* (UG470).

USR_ACCESSE2

Primitive: Configuration Data Access



Introduction

The USR_ACCESSE2 design element enables access to the 32-bit AXSS register within the configuration logic. This enables FPGA logic to access static data that can be set from the bitstream. The USR_ACCESSE2 register AXSS can be used to provide a single 32-bit constant value to the FPGA logic. The register contents can be defined during bitstream generation, avoiding the need to re-compile the design as would be required if distributed RAM was used to hold the constant. A constant can be used to track the version of the design, or any other information you require.

Port Descriptions

Port	Direction	Width	Function
CFGCLK	Output	1	Configuration Clock output.
DATA<31:0>	Output	32	Configuration Data reflecting the contents of the AXSS register.
DATAVALID	Output	1	Active-High data valid output.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- USR_ACCESSE2: Configuration Data Access
--               7 Series
-- Xilinx HDL Language Template, version 2020.1

USR_ACCESSE2_inst : USR_ACCESSE2
port map (
   CFGCLK => CFGCLK,         -- 1-bit output: Configuration Clock output
   DATA => DATA,             -- 32-bit output: Configuration Data output
   DATAVALID => DATAVALID    -- 1-bit output: Active high data valid output
);

-- End of USR_ACCESSE2_inst instantiation
```

Verilog Instantiation Template

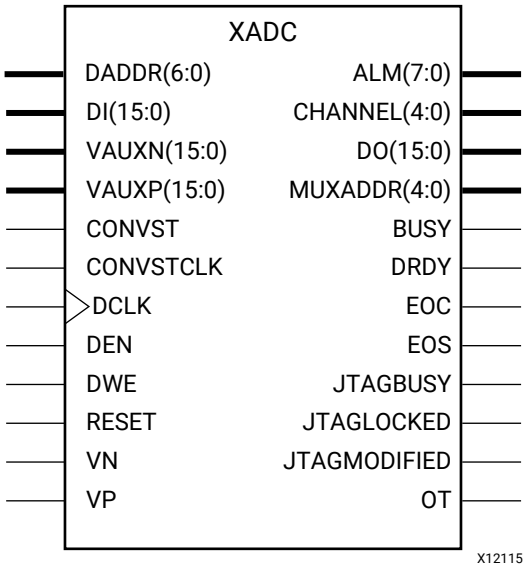
```
// USR_ACCESSE2: Configuration Data Access
//               7 Series
// Xilinx HDL Language Template, version 2020.1

USR_ACCESSE2 USR_ACCESSE2_inst (
   .CFGCLK(CFGCLK),         // 1-bit output: Configuration Clock output
   .DATA(DATA),             // 32-bit output: Configuration Data output
   .DATAVALID(DATAVALID)    // 1-bit output: Active high data valid output
);

// End of USR_ACCESSE2_inst instantiation
```

For More Information

- See the *7 Series FPGAs Configuration Guide* (UG470).

XADC

Primitive: Dual 12-Bit 1MSPS Analog-to-Digital Converter



Introduction

XADC includes a dual 12-bit, 1 Mega sample per second (MSPS) ADC and on-chip sensors. These ADCs are fully tested and specified (see the respective 7 series FPGAs data sheet). The ADCs provide a general-purpose, high-precision analog interface for a range of applications. The dual ADCs support a range of operating modes, for example, externally triggered and simultaneous sampling on both ADCs and various analog input signal types, for example, unipolar, and differential. The ADCs can access up to 17 external analog input channels.

XADC also includes a number of on-chip sensors that support measurement of the on-chip power supply voltages and die temperature. The ADC conversion data is stored in dedicated registers called status registers. These registers are accessible via the FPGA interconnect using a 16-bit synchronous read and write port called the Dynamic Reconfiguration Port (DRP). ADC conversion data is also accessible via the JTAG TAP. In the latter case, users are not required to instantiate the XADC because it is a dedicated interface that uses the existing FPGA JTAG infrastructure. If the XADC is not instantiated in a design, the device operates in a predefined mode (called default mode) that monitors on-chip temperature and supply voltages.

XADC operation is user defined by writing to the control registers using either the DRP or JTAG interface. It is also possible to initialize these register contents when the XADC is instantiated in a design using the block attributes.

Port Descriptions

Port	Direction	Width	Function
ALM<7:0>	Output	8	Output alarm for temperature, Vccint, Vccaux and Vccbram. • ALM[0]: XADC temperature sensor alarm output. • ALM[1]: XADC Vccint sensor alarm output. • ALM[2]: XADC Vccaux sensor alarm output. • ALM[3]: XADC Vccbram sensor alarm output. • ALM[6:4]: Not defined.
BUSY	Output	1	ADC busy signal. This signal transitions High during an ADC conversion. This signal also transitions High for an extended period during an ADC or sensor calibration.
CHANNEL<4:0>	Output	5	Channel selection outputs. The ADC input MUX channel selection for the current ADC conversion is placed on these outputs at the end of an ADC conversion.
CONVST	Input	1	Convert start input. This input controls the sampling instant on the ADC(s) input and is only used in event mode timing. This input comes from the general-purpose interconnect in the FPGA logic.
CONVSTCLK	Input	1	Convert start clock input. This input is connected to a clock net. Like CONVST, this input controls the sampling instant on the ADC(s) inputs and is only used in event mode timing. This input comes from the local clock distribution network in the FPGA logic. Thus, for the best control over the sampling instant (delay and jitter), a global clock input can be used as the CONVST source.
DADDR<6:0>	Input	7	Address bus for the dynamic reconfiguration port.
DCLK	Input	1	Clock input for the dynamic reconfiguration port.
DEN	Input	1	Enable signal for the dynamic reconfiguration port.
DI<15:0>	Input	16	Input data bus for the dynamic reconfiguration port.
DO<15:0>	Output	16	Output data bus for dynamic reconfiguration port.
DRDY	Output	1	Data ready signal for the dynamic reconfiguration port.
DWE	Input	1	Write enable for the dynamic reconfiguration port.
EOC	Output	1	End of Conversion signal. This signal transitions to an active High at the end of an ADC conversion when the measurement is written to the status registers.
EOS	Output	1	End of Sequence. This signal transitions to active-High when the measurement data from the last channel in an automatic channel sequence is written to the status registers.
JTAGBUSY	Output	1	Used to indicate that a JTAG DRP transaction is in progress.
JTAGLOCKED	Output	1	Indicates that a DRP port lock request has been made by the JTAG interface. This signal is also used to indicate that the DRP is ready for access (when Low).
JTAGMODIFIED	Output	1	Used to indicate that a JTAG Write to the DRP has occurred.
MUXADDR<4:0>	Output	5	These outputs are used in external multiplexer mode. They indicate the address of the next channel in a sequence to be converted. They provide the channel address for an external multiplexer.

Port	Direction	Width	Function
OT	Output	1	Over-Temperature alarm
RESET	Input	1	Reset signal for the XADC control logic.
VAUXN<15:0>	Input	16	N-side auxiliary analog input
VAUXP<15:0>	Input	16	P-side auxiliary analog input
VN	Input	1	N-side analog input
VP	Input	1	P-side analog input

Design Entry Method

Instantiation	Yes
Inference	No
IP Catalog	Recommended
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT_4A	HEX	16'h0000 to 16'hffff	16'h0000	Sequence register 2
INIT_4B	HEX	16'h0000 to 16'hffff	16'h0000	Sequence register 3
INIT_4C	HEX	16'h0000 to 16'hffff	16'h0000	Sequence register 4
INIT_4D	HEX	16'h0000 to 16'hffff	16'h0000	Sequence register 5
INIT_4E	HEX	16'h0000 to 16'hffff	16'h0000	Sequence register 6
INIT_4F	HEX	16'h0000 to 16'hffff	16'h0000	Sequence register 7
INIT_5C	HEX	16'h0000 to 16'hffff	16'h0000	Vbram lower alarm threshold
INIT_40	HEX	16'h0000 to 16'hffff	16'h0000	Configuration register 0
INIT_41	HEX	16'h0000 to 16'hffff	16'h0000	Configuration register 1
INIT_42	HEX	16'h0000 to 16'hffff	16'h0800	Configuration register 2
INIT_43	HEX	16'h0000 to 16'hffff	16'h0000	Test register 0
INIT_44	HEX	16'h0000 to 16'hffff	16'h0000	Test register 1
INIT_45	HEX	16'h0000 to 16'hffff	16'h0000	Test register 2
INIT_46	HEX	16'h0000 to 16'hffff	16'h0000	Test register 3
INIT_47	HEX	16'h0000 to 16'hffff	16'h0000	Test register 4
INIT_48	HEX	16'h0000 to 16'hffff	16'h0000	Sequence register 0
INIT_49	HEX	16'h0000 to 16'hffff	16'h0000	Sequence register 1
INIT_50	HEX	16'h0000 to 16'hffff	16'h0000	Alarm limit register 0
INIT_51	HEX	16'h0000 to 16'hffff	16'h0000	Alarm limit register 1
INIT_52	HEX	16'h0000 to 16'hffff	16'h0000	Alarm limit register 2
INIT_53	HEX	16'h0000 to 16'hffff	16'h0000	Alarm limit register 3
INIT_54	HEX	16'h0000 to 16'hffff	16'h0000	Alarm limit register 4
INIT_55	HEX	16'h0000 to 16'hffff	16'h0000	Alarm limit register 5
INIT_56	HEX	16'h0000 to 16'hffff	16'h0000	Alarm limit register 6

Attribute	Type	Allowed Values	Default	Description
INIT_57	HEX	16'h0000 to 16'hffff	16'h0000	Alarm limit register 7
INIT_58	HEX	16'h0000 to 16'hffff	16'h0000	Vbram upper alarm threshold
INIT_59, INIT_5A, INIT_5B, INIT_5D, INIT_5E, INIT_5F	HEX	16'h0000 to 16'hffff	16'h0000	Reserved for future use
SIM_DEVICE	STRING	"7SERIES", "ZYNQ"	"7SERIES"	Selects target device to allow for proper simulation.
SIM_MONITOR_FILE	STRING	String representing file name and location	"design.txt"	Specify the file name (and directory if different from simulation directory) of file containing analog voltage and temperature data for XADC simulation behavior.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- XADC: Dual 12-Bit 1MSPS Analog-to-Digital Converter
--       7 Series
-- Xilinx HDL Language Template, version 2020.1

XADC_inst : XADC
generic map (
   -- INIT_40 - INIT_42: XADC configuration registers
   INIT_40 => X"0000",
   INIT_41 => X"0000",
   INIT_42 => X"0800",
   -- INIT_48 - INIT_4F: Sequence Registers
   INIT_48 => X"0000",
   INIT_49 => X"0000",
   INIT_4A => X"0000",
   INIT_4B => X"0000",
   INIT_4C => X"0000",
   INIT_4D => X"0000",
   INIT_4F => X"0000",
   INIT_4E => X"0000",                  -- Sequence register 6
   -- INIT_50 - INIT_58, INIT5C: Alarm Limit Registers
   INIT_50 => X"0000",
   INIT_51 => X"0000",
   INIT_52 => X"0000",
   INIT_53 => X"0000",
   INIT_54 => X"0000",
   INIT_55 => X"0000",
   INIT_56 => X"0000",
   INIT_57 => X"0000",
   INIT_58 => X"0000",
   INIT_5C => X"0000",
   -- Simulation attributes: Set for proper simulation behavior
   SIM_DEVICE => "7SERIES",             -- Select target device (values)
   SIM_MONITOR_FILE => "design.txt"  -- Analog simulation data file name
)
port map (
   -- ALARMS: 8-bit (each) output: ALM, OT
   ALM => ALM,                  -- 8-bit output: Output alarm for temp, Vccint, Vccaux and Vccbram
   OT => OT,                    -- 1-bit output: Over-Temperature alarm
   -- Dynamic Reconfiguration Port (DRP): 16-bit (each) output: Dynamic Reconfiguration Ports
   DO => DO,                    -- 16-bit output: DRP output data bus
   DRDY => DRDY,                -- 1-bit output: DRP data ready
   -- STATUS: 1-bit (each) output: XADC status ports
   BUSY => BUSY,                -- 1-bit output: ADC busy output
   CHANNEL => CHANNEL,          -- 5-bit output: Channel selection outputs
   EOC => EOC,                  -- 1-bit output: End of Conversion
   EOS => EOS,                  -- 1-bit output: End of Sequence
```

```
   JTAGBUSY => JTAGBUSY,          -- 1-bit output: JTAG DRP transaction in progress output
   JTAGLOCKED => JTAGLOCKED,      -- 1-bit output: JTAG requested DRP port lock
   JTAGMODIFIED => JTAGMODIFIED, -- 1-bit output: JTAG Write to the DRP has occurred
   MUXADDR => MUXADDR,            -- 5-bit output: External MUX channel decode
   -- Auxiliary Analog-Input Pairs: 16-bit (each) input: VAUXP[15:0], VAUXN[15:0]
   VAUXN => VAUXN,                -- 16-bit input: N-side auxiliary analog input
   VAUXP => VAUXP,                -- 16-bit input: P-side auxiliary analog input
   -- CONTROL and CLOCK: 1-bit (each) input: Reset, conversion start and clock inputs
   CONVST => CONVST,              -- 1-bit input: Convert start input
   CONVSTCLK => CONVSTCLK,        -- 1-bit input: Convert start input
   RESET => RESET,                -- 1-bit input: Active-high reset
   -- Dedicated Analog Input Pair: 1-bit (each) input: VP/VN
   VN => VN,                      -- 1-bit input: N-side analog input
   VP => VP,                      -- 1-bit input: P-side analog input
   -- Dynamic Reconfiguration Port (DRP): 7-bit (each) input: Dynamic Reconfiguration Ports
   DADDR => DADDR,                -- 7-bit input: DRP address bus
   DCLK => DCLK,                  -- 1-bit input: DRP clock
   DEN => DEN,                    -- 1-bit input: DRP enable signal
   DI => DI,                      -- 16-bit input: DRP input data bus
   DWE => DWE                     -- 1-bit input: DRP write enable
);

-- End of XADC_inst instantiation
```

Verilog Instantiation Template

```
// XADC: Dual 12-Bit 1MSPS Analog-to-Digital Converter
//       7 Series
// Xilinx HDL Language Template, version 2020.1

XADC #(
   // INIT_40 - INIT_42: XADC configuration registers
   .INIT_40(16'h0000),
   .INIT_41(16'h0000),
   .INIT_42(16'h0800),
   // INIT_48 - INIT_4F: Sequence Registers
   .INIT_48(16'h0000),
   .INIT_49(16'h0000),
   .INIT_4A(16'h0000),
   .INIT_4B(16'h0000),
   .INIT_4C(16'h0000),
   .INIT_4D(16'h0000),
   .INIT_4F(16'h0000),
   .INIT_4E(16'h0000),                 // Sequence register 6
   // INIT_50 - INIT_58, INIT5C: Alarm Limit Registers
   .INIT_50(16'h0000),
   .INIT_51(16'h0000),
   .INIT_52(16'h0000),
   .INIT_53(16'h0000),
   .INIT_54(16'h0000),
   .INIT_55(16'h0000),
   .INIT_56(16'h0000),
   .INIT_57(16'h0000),
   .INIT_58(16'h0000),
   .INIT_5C(16'h0000),
   // Simulation attributes: Set for proper simulation behavior
   .SIM_DEVICE("7SERIES"),             // Select target device (values)
   .SIM_MONITOR_FILE("design.txt")  // Analog simulation data file name
)
XADC_inst (
   // ALARMS: 8-bit (each) output: ALM, OT
   .ALM(ALM),                    // 8-bit output: Output alarm for temp, Vccint, Vccaux and Vccbram
   .OT(OT),                      // 1-bit output: Over-Temperature alarm
   // Dynamic Reconfiguration Port (DRP): 16-bit (each) output: Dynamic Reconfiguration Ports
   .DO(DO),                      // 16-bit output: DRP output data bus
   .DRDY(DRDY),                  // 1-bit output: DRP data ready
   // STATUS: 1-bit (each) output: XADC status ports
   .BUSY(BUSY),                  // 1-bit output: ADC busy output
   .CHANNEL(CHANNEL),            // 5-bit output: Channel selection outputs
   .EOC(EOC),                    // 1-bit output: End of Conversion
   .EOS(EOS),                    // 1-bit output: End of Sequence
   .JTAGBUSY(JTAGBUSY),          // 1-bit output: JTAG DRP transaction in progress output
   .JTAGLOCKED(JTAGLOCKED),      // 1-bit output: JTAG requested DRP port lock
   .JTAGMODIFIED(JTAGMODIFIED), // 1-bit output: JTAG Write to the DRP has occurred
   .MUXADDR(MUXADDR),            // 5-bit output: External MUX channel decode
   // Auxiliary Analog-Input Pairs: 16-bit (each) input: VAUXP[15:0], VAUXN[15:0]
```

```
   .VAUXN(VAUXN),               // 16-bit input: N-side auxiliary analog input
   .VAUXP(VAUXP),               // 16-bit input: P-side auxiliary analog input
   // CONTROL and CLOCK: 1-bit (each) input: Reset, conversion start and clock inputs
   .CONVST(CONVST),             // 1-bit input: Convert start input
   .CONVSTCLK(CONVSTCLK),       // 1-bit input: Convert start input
   .RESET(RESET),               // 1-bit input: Active-high reset
   // Dedicated Analog Input Pair: 1-bit (each) input: VP/VN
   .VN(VN),                     // 1-bit input: N-side analog input
   .VP(VP),                     // 1-bit input: P-side analog input
   // Dynamic Reconfiguration Port (DRP): 7-bit (each) input: Dynamic Reconfiguration Ports
   .DADDR(DADDR),               // 7-bit input: DRP address bus
   .DCLK(DCLK),                 // 1-bit input: DRP clock
   .DEN(DEN),                   // 1-bit input: DRP enable signal
   .DI(DI),                     // 16-bit input: DRP input data bus
   .DWE(DWE)                    // 1-bit input: DRP write enable
);

// End of XADC_inst instantiation
```

For More Information

- See the *7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide* (UG480).

XILINX

Appendix A

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

Documentation Navigator and Design Hubs

Xilinx Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado IDE, select **Help → Documentation and Tutorials**.
- On Windows, select **Start → All Programs → Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS “XA” IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE (“SAFETY APPLICATION”) UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD (“SAFETY DESIGN”). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.