UltraScale Architecture Memory Resources

User Guide

UG573 (v1.12) March 17, 2021

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/17/2021	1.12	Added additional bullet to list before Table 1-3. Revised values, default, and type for IS_XX_INVERTED attributes in Table 2-5.
08/18/2020	1.11	Revised important notes in Block RAM Summary, Common-Clock/Single-Clock FIFO, and Block RAM and UltraRAM Differences.
02/04/2019	1.10	Added information on common and independent clocks in Address Collision. Revised port A to write port and port B to read port in Simple Dual-Port Block RAM. Removed content clear from UltraRAM Key Features.
02/09/2018	1.9	Revised description for EN_A and EN_B ports in Table 2-2. Revised Figure 2-16.
11/14/2017	1.8	Revised important note in Common-Clock/Single-Clock FIFO and added note to Set Enable Auto Sleep Mode – EN_AUTO_SLEEP_MODE. Revised UltraRAM Summary. Removed Figure 2-28 and associated verbiage.
08/10/2017	1.7	Added important note to Common-Clock/Single-Clock FIFO. Revised ECC Encode-Only Read.
05/04/2017	1.6	Updated Table 2-5.
03/15/2017	1.5	Revised important note in Block RAM Summary. Added note 2 to Table 1-16. Revised Reset – RST_A, RST_B. Added UltraRAM Timing Diagrams.
07/20/2016	1.4	Revised Differences from Previous Generations, Synchronous Dual-Port and Single-Port RAMs, Power Saving – RDADDRCHANGE[A|B], Power Gating Enable Input – SLEEP, Data-Out Buses – DOUT_A, DOUT_B, Read Status Output – RDACCESS_A, RDACCESS_B, Optional Cascade Register Stage – REG_CAS_[A|B], Cascading UltraRAM and Matrix Configurations, and Cascade User Attributes. Added note to Table 1-16 and Table 1-32. Updated Table 2-5. Added AVG_CONS_INACTIVE_CYCLES, MATRIX_ID, NUM_URAM_IN_MATRIX, and NUM_UNIQUE_SELF_ADDR_A|B Attributes. Updated Figure 2-1 and Figure 2-5.
11/24/2015	1.3	Added UltraScale+ device information. Updated Introduction to the UltraScale Architecture to include UltraScale+ information. Updated important note in Block RAM Summary. Added introductory paragraph to RAMB18/36 Unused Inputs. Reorganized user guide by incorporating previous Chapter 2 (Built-in FIFO) and Chapter 3 (Built-in Error Correction) into Chapter 1, Block RAM Resources, and adding a new Chapter 2, UltraRAM Resources.
02/24/2015	1.2	Updated Content Initialization – INIT_xx.

Date	Version	Revision
08/14/2014	1.1	Updated bullet 11 in Block RAM Summary. Changed description of CASDINPA[3:0] and CASDINB[31:0] in Table 1-8. Updated description of DOB_REG in Table 1-16. Updated WDADDREN input in Figure 1-6. Added Table 1-14. Updated Cascadable Block RAM. Minor changes in SLEEP and Power Saving – SLEEP_ASYNC. Minor changes to descriptions in Common-Clock/Single-Clock FIFO. Added SLEEP input to Figure 1-21 and Figure 1-22. Added SLEEP port and changed descriptions of CASOREGIMUX, CASOREGIMUXEN, CASDOMUX, and CASDOMUXEN in Table 1-25. Updated Table 1-26. Added Table 1-27. Updated PROG_EMPTY_THRESH, PROG_FULL_THRESH, and REGISTER_MODE in Chapter 1. Removed PROG_EMPTY_THRESH Range for FIFO18E2/FIFO36E2 and PROG_FULL_THRESH Range for FIFO18E2/FIFO36E2 tables in Chapter 2. Updated CASOUTSBITERR, CASINDBITERR, and CASOUTSBITERR in Table 1-8.
12/10/2013	1.0	Initial Xilinx release.

Table of Contents

Chapter 1: Block RAM Resources

Chapter 2: UltraRAM Resources

Appendix A: Additional Resources and Legal Notices

Chapter 1

Block RAM Resources

Introduction to the UltraScale Architecture

The Xilinx® UltraScale™ architecture is the first ASIC-class architecture to enable multi-hundred gigabit-per-second levels of system performance with smart processing, while efficiently routing and processing data on-chip. UltraScale architecture-based devices address a vast spectrum of high-bandwidth, high-utilization system requirements by using industry-leading technical innovations, including next-generation routing, ASIC-like clocking, 3D-on-3D ICs, multiprocessor SoC (MPSoC) technologies, and new power reduction features. The devices share many building blocks, providing scalability across process nodes and product families to leverage system-level investment across platforms.

Virtex® UltraScale+™ devices provide the highest performance and integration capabilities in a FinFET node, including both the highest serial I/O and signal processing bandwidth, as well as the highest on-chip memory density. As the industry's most capable FPGA family, the Virtex UltraScale+ devices are ideal for applications including 1+Tb/s networking and data center and fully integrated radar/early-warning systems.

Virtex UltraScale devices provide the greatest performance and integration at 20 nm, including serial I/O bandwidth and logic capacity. As the industry's only high-end FPGA at the 20 nm process node, this family is ideal for applications including 400G networking, large scale ASIC prototyping, and emulation.

Kintex® UltraScale+ devices provide the best price/performance/watt balance in a FinFET node, delivering the most cost-effective solution for high-end capabilities, including transceiver and memory interface line rates as well as 100G connectivity cores. Our newest mid-range family is ideal for both packet processing and DSP-intensive functions and is well suited for applications including wireless MIMO technology, Nx100G networking, and data center.

Kintex UltraScale devices provide the best price/performance/watt at 20 nm and include the highest signal processing bandwidth in a mid-range device, next-generation transceivers, and low-cost packaging for an optimum blend of capability and cost-effectiveness. The family is ideal for packet processing in 100G networking and data centers applications as well as DSP-intensive processing needed in next-generation medical imaging, 8k4k video, and heterogeneous wireless infrastructure.

Zynq® UltraScale+ MPSoC devices provide 64-bit processor scalability while combining real-time control with soft and hard engines for graphics, video, waveform, and packet processing. Integrating an Arm®-based system for advanced analytics and on-chip programmable logic for task acceleration creates unlimited possibilities for applications including 5G Wireless, next generation ADAS, and Industrial Internet-of-Things.

This user guide describes the UltraScale architecture memory resources and is part of the UltraScale architecture documentation suite available at: www.xilinx.com/documentation.

Block RAM Summary

The block RAM in UltraScale architecture-based devices stores up to 36 Kbits of data and can be configured as either two independent 18 Kb RAMs, or one 36 Kb RAM. Each block RAM has two write and two read ports. A 36 Kb block RAM can be configured with independent port widths for each of those ports as 32K x 1, 16K x 2, 8K x 4, 4K x 9, 2K x 18 or 1K x 36 (when used as true dual-port (TDP) memory). If only one write and one read port are used, a 36 Kb block RAM can additionally be configured with a port width of 512 x 72 bits (when used as simple dual-port (SDP) memory). An 18 Kb block RAM can be configured with independent port widths for each of those ports as 16K x 1, 8K x 2, 4K x 4, 2K x 9 or 1K x 18 (when used as TDP memory). If only one write and one read port are used, an 18 Kb block RAM can additionally be configured with a port width of 512 x 36 bits (when used as SDP memory).

Similar to the 7 series FPGA block RAMs, write and read are synchronous operations. The two ports are symmetrical and totally independent, sharing only the stored data. Each port can be configured in one of the available widths, independent of the other port. In addition, the read port width can be different from the write port width for each port. The memory content can be initialized or cleared by the configuration bitstream. During a write operation, the memory can be set to have the data output remain unchanged, reflect the new data being written or the previous data now being overwritten.

The block RAM features include:

- Per-block memory storage capability where each block RAM can store up to 36 Kbits of data.
- Support of two independent 18 Kb blocks, or a single 36 Kb block RAM.
- Each 36 Kb block RAM can be used with a single read and write port (SDP), doubling data width of the block RAM to 72 bits. The 18 Kb block RAM can also be used with a single read and write port, doubling data width to 36 bits.
- When used as RAMB36 SDP memory, one port width is fixed (i.e., 512 x 64 or 512 x 72). The other port width can then be 32K x 1 through 512 x 72. When used as RAMB18 SDP memory, one port width is fixed (i.e., 512 x 36). The other port width can then be 16K x 1 through 512 x 36.

- The data outputs of the lower to upper adjacent block RAMs can be cascaded to build large block RAM blocks. Optional pipeline registers are available to support maximum performance.
- One 64-bit error correction coding (ECC) block is provided per 36 Kb block RAM or 36 Kb FIFO. Independent encode/decode functionality is available. ECC mode has the capability of injecting errors.
- Synchronous set/reset of the outputs to an initial value is available for both the latch and register modes of the block RAM output.
- Separate synchronous set/reset pins independently control the set/reset of the optional output registers and output latch stages in the block RAM.
- An attribute to configure the block RAM as a common-clock/single-clock FIFO to eliminate flag latency uncertainty.
- 18, 36, or 72-bit wide block RAM ports can have an individual write enable per byte. This feature is popular for interfacing to a microprocessor.
- Each block RAM contains optional address sequencing and control circuitry to operate as a built-in independent-clock FIFO memory. The block RAM can be configured as an 18 Kb or 36 Kb FIFO.
- All inputs are registered with the port clock and have a setup-to-clock timing specification.
- All outputs have a read function or a read-during-write function, depending on the state of the write enable (WE) pin. The outputs are available after the clock-to-out timing interval. The read-during-write outputs have one of three operating modes: WRITE_FIRST, READ_FIRST, and NO_CHANGE.
- A write operation requires one clock edge.
- A read operation requires one clock edge.
- All output ports are latched or registered (optional). The state of the output port does not change until the port executes another read or write operation. The default block RAM output is register mode.

✓ **RECOMMENDED:** *The output datapath has an optional internal pipeline register. Using the register mode is strongly recommended. This allows a higher clock rate. However, it adds a clock cycle latency of one.*

The block RAM usage rules include:

- The block RAM synchronous output registers (optional) are set or reset (SRVAL) with RSTREG when DO_REG = 1. The RSTREG_PRIORITY attribute determines if RSTREG has priority over REGCE. The synchronous output latches are set or reset (SRVAL) with RSTRAM when DO_REG is 0 or 1.

IMPORTANT: *The clock minimum pulse width and setup/hold time of the block RAM address, and write enable pins must not be violated. Violating the clock minimum pulse width or these setup/hold times (even if write enable is Low) can corrupt the data contents of the block RAM. This most commonly occurs during an unstable clock or when flip-flops driving block RAM control pins are asynchronously reset, such as a system wide reset. To avoid this issue, ensure stable clocks and design with synchronous resets for both assertion and deassertion. When the clock is not stable, disable the clock buffer or disable logic driving the block RAM control pins or deassert the block RAM EN input.*

- The block RAM register mode RSTREG requires REGCE = 1 to reset the output DO register value if the RSTREG_PRIORITY is set to REGCE. The block RAM array data output latch does not get reset in this mode. The block RAM latch mode RSTRAM requires the block RAM enable, EN = 1, to reset the output DO latch value.
- There are two block RAM primitives: RAMB36E2 and RAMB18E2.
- Different read and write port width choices are available when using specific block RAM primitives. The parity bits are only available for the x9, x18, and x36 port widths. The parity bits should not be used when the read width is x1, x2, or x4. If the read width is x1, x2, or x4, the effective write width is x1, x2, x4, x8, x16, or x32. Similarly, when a write width is x1, x2, or x4, the actual available read width is x1, x2, x4, x8, x16, or x32 even though the primitive attribute is set to 1, 2, 4, 9, 18, or 36, respectively. Table 1-1 shows some possible scenarios.

Table 1-1: **Parity Use Scenarios**

Primitive	Settings		Effective Read Width	Effective Write Width
	Read Width	Write Width		
RAMB18E2	1, 2, or 4	9 or 18	Same as setting	8 or 16
RAMB18E2	9 or 18	1, 2, or 4	8 or 16	Same as setting
RAMB18E2	1, 2, or 4	1, 2, or 4	Same as setting	Same as setting
RAMB18E2	9 or 18	9 or 18	Same as setting	Same as setting
RAMB36E2	1, 2, or 4	9, 18, or 36	Same as setting	8, 16, or 32
RAMB36E2	9, 18, or 36	1, 2, or 4	8, 16, or 32	Same as setting
RAMB36E2	1, 2, or 4	1, 2, or 4	Same as setting	Same as setting
RAMB36E2	9, 18, or 36	9, 18, or 36	Same as setting	Same as setting

Notes:

1. Do not use parity bits DINP/DOUTP when one port width is less than 9 and another port width is 9 or greater.

Differences from Previous Generations

Changes from 7 Series FPGAs

- When used as SDP memory, all write modes are supported (READ_FIRST, WRITE_FIRST, NO_CHANGE).
- UltraScale architecture-based devices have a new data cascading scheme. Large block RAMs can now be built in a bottom-up fashion directly in the block RAM column without additional use of logic resources.
- An address enable feature has been added to the block RAM. If disabled, the new address is not latched in the block.
- A dynamic power gating capability has been added. The block RAM can be put into sleep mode while preserving the data content.
- The RAM_MODE attribute has been removed. The Vivado® tools automatically determine if a block RAM is used in TDP or SDP mode.
- The built-in FIFOs and IP FIFOs have been harmonized as much as possible. This makes it easier to switch between soft and hard FIFO implementations.
- FIFOs allow cascading of multiple FIFO36s and FIFO18s for building deeper FIFOs in hardware.
- A synchronous FIFO reset replacing the asynchronous reset in previous generations has been added.
- FIFO latencies of the deassertion of the EMPTY/PROGEMPTY flag for a write operation and the FULL/PROGFULL flag for a read operation have changed.
- The behavior of WRERR and RDERR during reset has changed.
- FIFO asymmetric ports are now supported. The write port and read port can each be configured independently as x4, x9, x18, x36, or x72 for the FIFO36E2, and x4, x9, x18, or x36 for the FIFO18E2.
- The combination of output operating modes (standard and first-word-fall-through) and output register stages configurations has changed.
- WRCOUNT and RDCOUNT now support additional user-selectable functionality.
- The block RAM ECC has additional pipeline registers for improved F_{MAX}.
- Hardware FIFOs are not backward compatible with 7 series FIFOs.

Block RAM Introduction

In addition to distributed RAM and high-speed SelectIO™ memory interfaces, UltraScale architecture-based devices feature a large number of 36 Kb block RAMs. Each 36 Kb block RAM contains two independently controlled 18 Kb RAMs. Block RAMs are placed in columns within the clock regions (CRs) and across the device. The block RAM data output blocks are cascadable to enable a deeper memory implementation, have a sleep mode for power savings, and have selectable write mode operations.

Synchronous Dual-Port and Single-Port RAMs

Data Flow

The true dual-port 36 Kb block RAM dual-port memories consist of a 36 Kb storage area and two completely independent access ports, A and B. Similarly, each 18 Kb block RAM dual-port memory consists of an 18 Kb storage area and two completely independent access ports, A and B. The structure is fully symmetrical, and both ports are interchangeable. Figure 1-1 illustrates the true dual-port data flow of a RAMB36. Table 1-2 lists the port functions and descriptions.

Data can be written to either or both ports and can be read from either or both ports. Each write operation is synchronous, and each port has its own address, data in, data out, clock, clock enable, and write enable. The read and write operations are synchronous and require a clock edge.

There is no dedicated monitor to arbitrate the effect of identical addresses on both ports.

IMPORTANT: *The two clocks must be timed appropriately. Conflicting simultaneous writes to the same location never cause any physical damage but can result in data uncertainty.*

Note: The Vivado tools automatically determine if a block RAM is used in SDP or TDP mode.

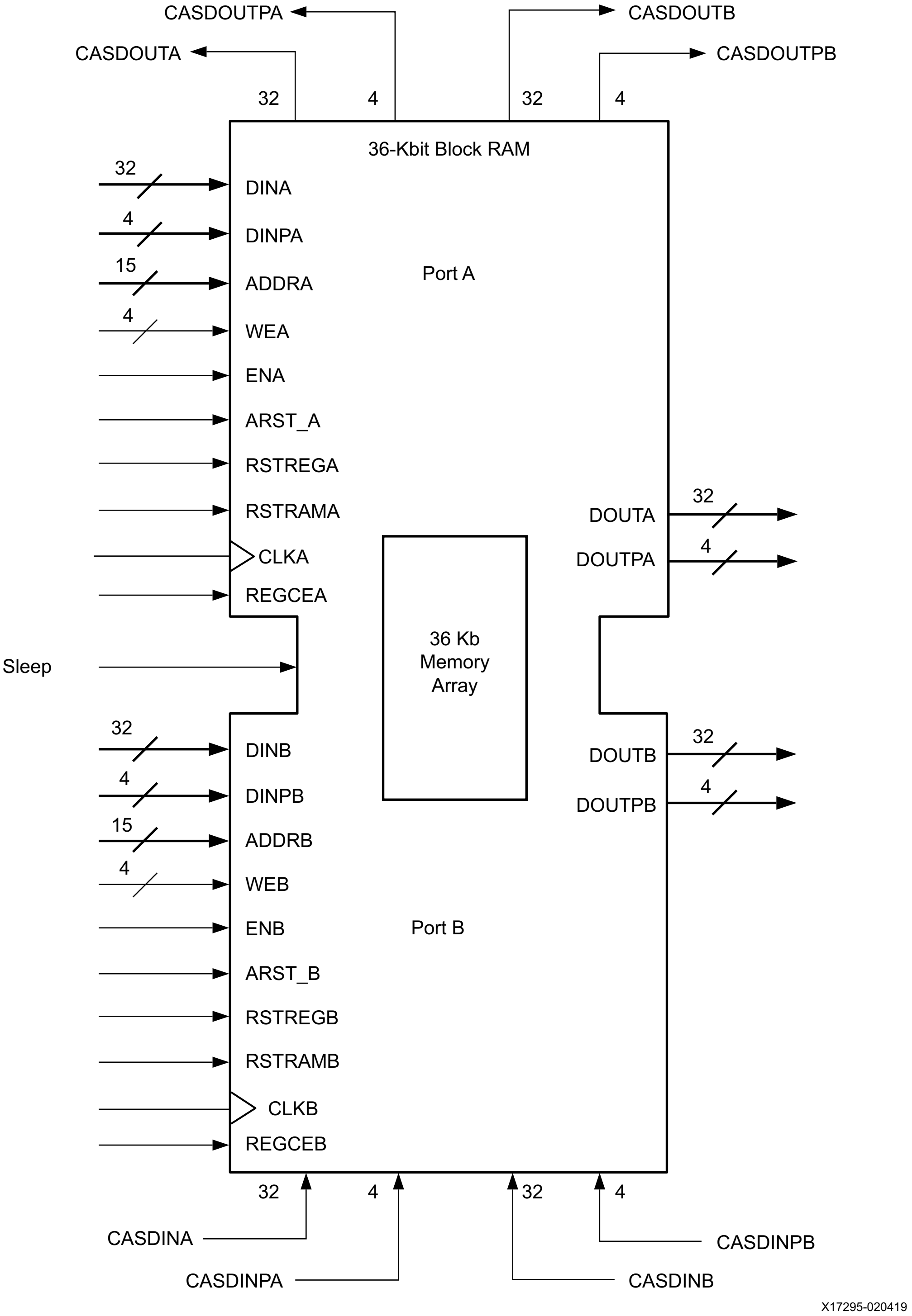


Figure 1-1: **RAMB36 Usage in a True Dual-Port Data Flow**

Table 1-2: **True Dual-Port Functions and Descriptions**

Port Function	Description
DIN[A|B]	Data input bus.
DINP[A|B] (1)	Data input parity bus. Can be used for additional data inputs.
ADDR[A|B]	Address bus.
ADDREN[A|B]	Address latching enable. If Low, the old address is latched.
WE[A|B]	Byte-wide write enable.
EN[A|B]	When inactive, no data is written to the block RAM and the output bus remains in its previous state.
RSTREG[A|B]	Synchronous set/reset of the output registers (DO_REG = 1). The RSTREG_PRIORITY attribute determines the priority over REGCE.
RSTRAM[A|B]	Synchronous set/reset of the output data latches.
CLK[A|B]	Clock input.
DOUT[A|B]	Data output bus.
DOUTP[A|B] (1)	Data output parity bus. Can be used for additional data outputs.
REGCE[A|B]	Output register clock enable.
CASDIN[A|B]	Cascade data input bus.
CASDINP[A|B]	Cascade parity input bus.
CASDOUT[A|B]	Cascade data output bus.
CASDOUTP[A|B]	Cascade parity output bus.
SLEEP	Dynamic shutdown power saving. If SLEEP is active, the block is in power saving mode.

Notes:

1. Data-In Buses – DINADIN, DINPADINP, DINBDIN, and DINPBDINP, page 35 has more information on data parity pins.
2. Block RAM primitive port names can be different from the port function names.
3. For a more complete cascade data flow and port descriptions, see Cascadable Block RAM, page 20 and Block RAM Library Primitives, page 28.

Read Operation

In latch mode, the read operation uses one clock edge. The read address is registered on the read port, and the stored data is loaded into the output latches after the RAM access time. When using the output register, the read operation takes one extra latency cycle.

Write Operation

A write operation is a single clock-edge operation. The write address is registered on the write port, and the data input is stored in memory.

Write Modes

Three settings of the write mode determine the behavior of the data available on the output latches after a write clock edge: WRITE_FIRST, READ_FIRST, and NO_CHANGE. Write mode selection is set by configuration. The write mode attribute can be individually selected for each port. The default mode is WRITE_FIRST. WRITE_FIRST outputs the newly written data onto the output bus. READ_FIRST outputs the previously stored data while new data is being written. NO_CHANGE maintains the output previously generated by a read operation.

WRITE_FIRST or Transparent Mode (Default)

In WRITE_FIRST mode, the input data is simultaneously written into memory and stored in the data output (transparent write), as shown in Figure 1-2. These waveforms correspond to latch mode when the optional output pipeline register is not used.

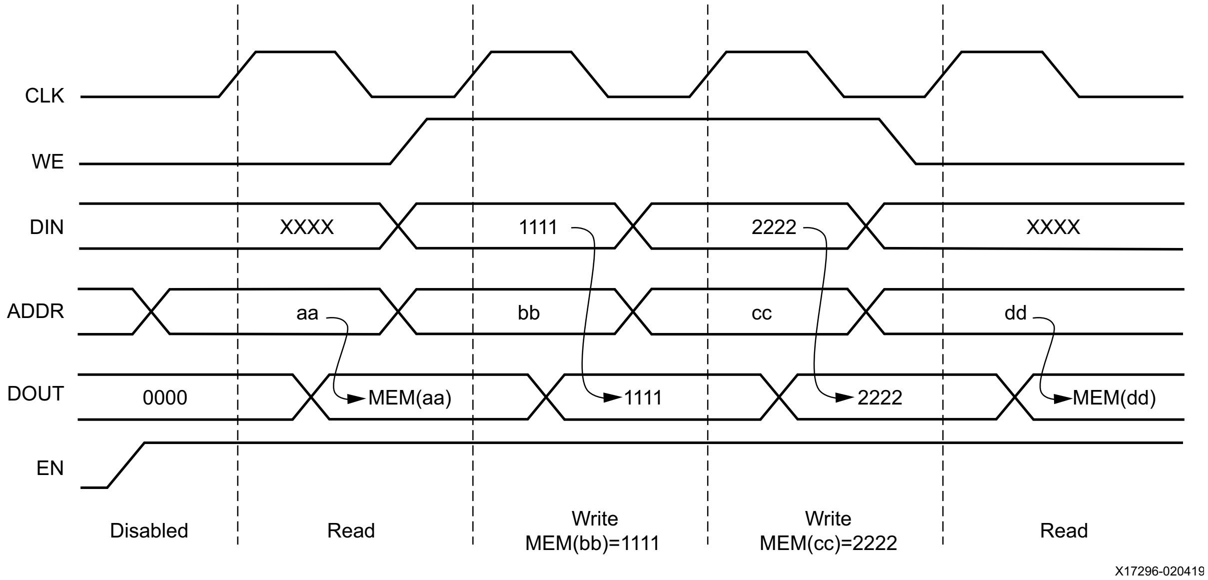


Figure 1-2: **WRITE_FIRST Mode Waveforms**

READ_FIRST or Read-Before-Write Mode

In READ_FIRST mode, data previously stored at the write address appears on the output latches while the input data is being stored in memory (read before write). The waveforms in Figure 1-3 correspond to latch mode when the optional output pipeline register is not used.

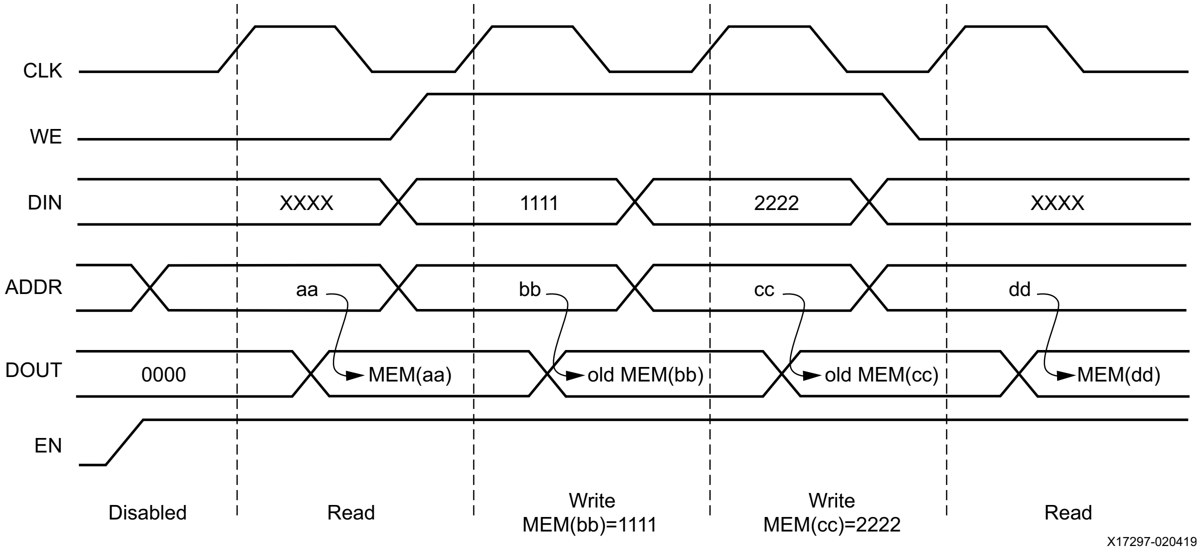


Figure 1-3: **READ_FIRST Mode Waveforms**

NO_CHANGE Mode

In NO_CHANGE mode, the output latches remain unchanged during a write operation. As shown in Figure 1-4, data output remains the last read data and is unaffected by a write operation on the same port. These waveforms correspond to latch mode when the optional output pipeline register is not used. NO_CHANGE mode is the most power efficient.

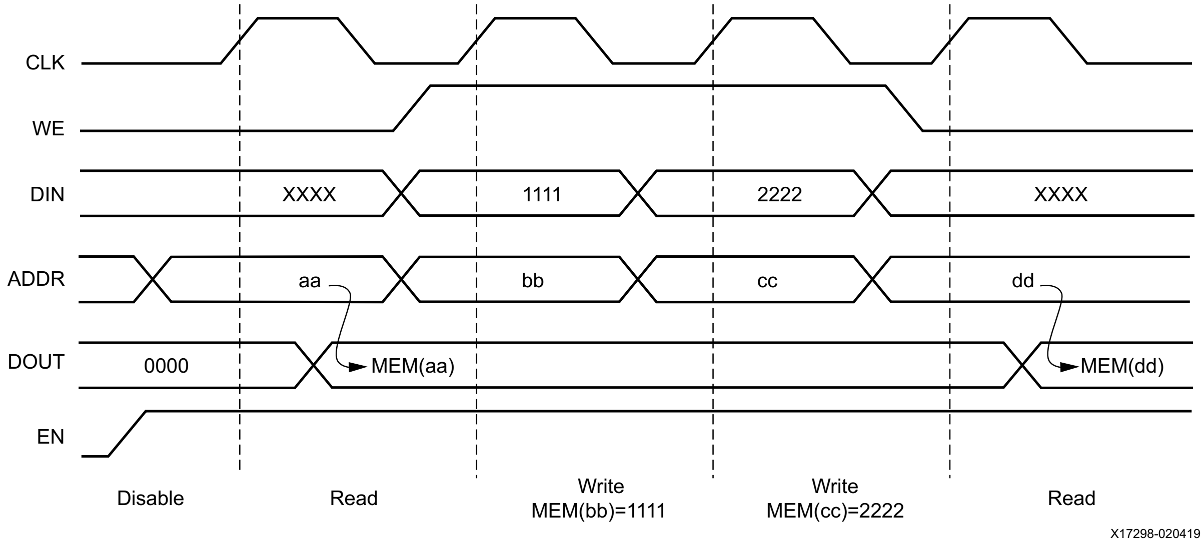


Figure 1-4: **NO_CHANGE Mode Waveforms**

Address Collision

An address collision is when both block RAM ports access the same address location in the same clock cycle. There are two fundamental clock type setups, common clock and independent clock. Common (synchronous) clocks are driven by a common clock buffer driver. All other CLKA and CLKB connections are considered independent (asynchronous) clocks. The CLOCK_DOMAINS attribute must also be set appropriately. See Table 1-16 for legal and default values. If no address collisions are expected or possible (SDP configurations) to save power, the recommended write mode is NO_CHANGE. Using READ_FIRST mode has a 15% power penalty over NO_CHANGE and should only be used when necessary for functionality or address collision mitigation.

- When both ports are reading, the operations complete successfully.
- When both ports are writing different data, the memory location is written with non-deterministic data.
- When one port is writing and the other port is reading, the write is always successful but the resulting read memory value can vary. See Table 1-3 and Table 1-4.

In Table 1-3 and Table 1-4:

- Write enable is active-High, 1 = Write, 0 = Read
- RF = READ_FIRST, WF = WRITE_FIRST, NC = NO_CHANGE
- X = Undeterministic value
- DIA = Port A data input, DIB = Port B data input
- Resulting data out applies to DOUT, DOUTP, SBITERR, and DBITERR

Table 1-3: **Common Clock**

Clock Type	Write Mode Port A	Write Mode Port B	Write Enable Port A (Data)	Write Enable Port B (Data)	Resulting Data Out Port A	Resulting Data Out Port B	Resulting Memory Value
Common	RF/WF/NC	RF/WF/NC	0	0	Old memory data	Old memory data	No change
Common	RF	RF/WF/NC	1 (DIA)	0	Old memory data	Old memory data	DIA
Common	WF	RF/WF/NC	1 (DIA)	0	DIA	X	DIA
Common	NC	RF/WF/NC	1 (DIA)	0	No change	X	DIA
Common	RF/WF/NC	RF	0	1 (DIB)	Old memory data	Old memory data	DIB
Common	RF/WF/NC	WF	0	1 (DIB)	X	DIB	DIB
Common	RF/WF/NC	NC	0	1 (DIB)	X	No change	DIB
Common	RF/WF/NC	RF/WF/NC	1	1	X	X	X

Note: Common clocked access collision is when the port addresses are the same for the same clock cycle.

Table 1-4: **Independent Clock**

Clock Type	Write Mode Port A	Write Mode Port B	Write Enable Port A (Data)	Write Enable Port B (Data)	Resulting Data Out Port A	Resulting Data Out Port B	Resulting Memory Value
Independent	RF/WF/NC	RF/WF/NC	0	0	Old memory data	Old memory data	No change
Independent	RF	RF/WF/NC	1 (DIA)	0	Old memory data	X	DIA
Independent	WF	RF/WF/NC	1 (DIA)	0	DIA	X	DIA
Independent	NC	RF/WF/NC	1 (DIA)	0	No change	X	DIA
Independent	RF/WF/NC	RF	0	1 (DIB)	X	Old memory data	DIB
Independent	RF/WF/NC	WF	0	1 (DIB)	X	DIB	DIB
Independent	RF/WF/NC	NC	0	1 (DIB)	X	No change	DIB
Independent	RF/WF/NC	RF/WF/NC	1	1	X	X	X

Note: An independently clocked access collision might occur when the port addresses are the same and when the clock edges of the two ports are within the same clock cycle. The UNISIM can report an error during simulation for collisions when the SIM_COLLISION_CHECK attribute is set to ALL (default).

Additional Block RAM Features

Optional Output Registers

The optional output registers improve design performance by eliminating routing delay to the configurable logic block (CLB) flip-flops for pipelined operation. An independent clock and clock enable input is provided for these output registers. As a result, the output data registers hold the value independent of the input register operation. Figure 1-5 shows the optional output register.

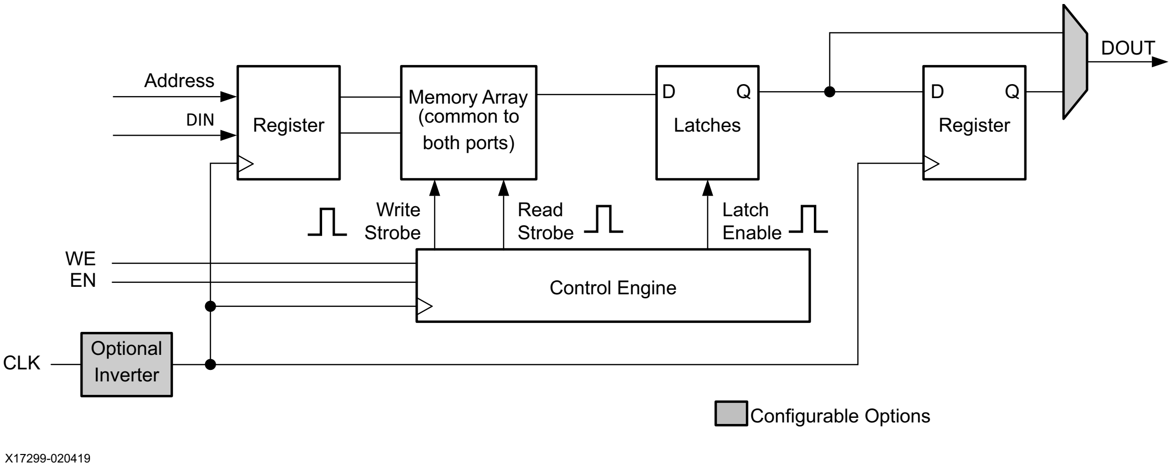


Figure 1-5: **Block RAM Logic Diagram (One Port Shown)**

Independent Read and Write Port Width Selection

Each block RAM port has control over data width and address depth (aspect ratio). The true dual-port block RAM extends this flexibility to read and write where each individual port can be configured with different data bit widths. For example, port A can have a 36-bit read width and a 9-bit write width, and port B can have an 18-bit read width and a 36-bit write width.

If the read port width differs from the write port width and is configured in WRITE_FIRST mode, DOUT shows valid new data for all the enabled write bytes. The DOUT port outputs the original data stored in memory for all not-enabled bytes.

Independent read and write port width selection increases the efficiency of implementing a content addressable memory (CAM) in block RAM. This option is available for all UltraScale architecture-based devices true dual-port RAM port sizes and modes.

Simple Dual-Port Block RAM

Each 18 Kb block and 36 Kb block can also be configured in a SDP RAM mode. In this mode, the block RAM port width doubles to 36 bits for the 18 Kb block RAM and 72 bits for the 36 Kb block RAM. When the block RAM is used as SDP memory, independent read and write operations can occur simultaneously, where port A is designated as the write port and port B as the read port. When the read and write port access the same data location at the same time, it is treated as a collision, identical to the port collision in true dual-port mode. UltraScale architecture-based devices support these modes when the block RAM is used as SDP memory (READ_FIRST, WRITE_FIRST, NO_CHANGE).

Figure 1-6 shows the simple dual-port data flow for RAMB36 when the block RAM is used as SDP memory.

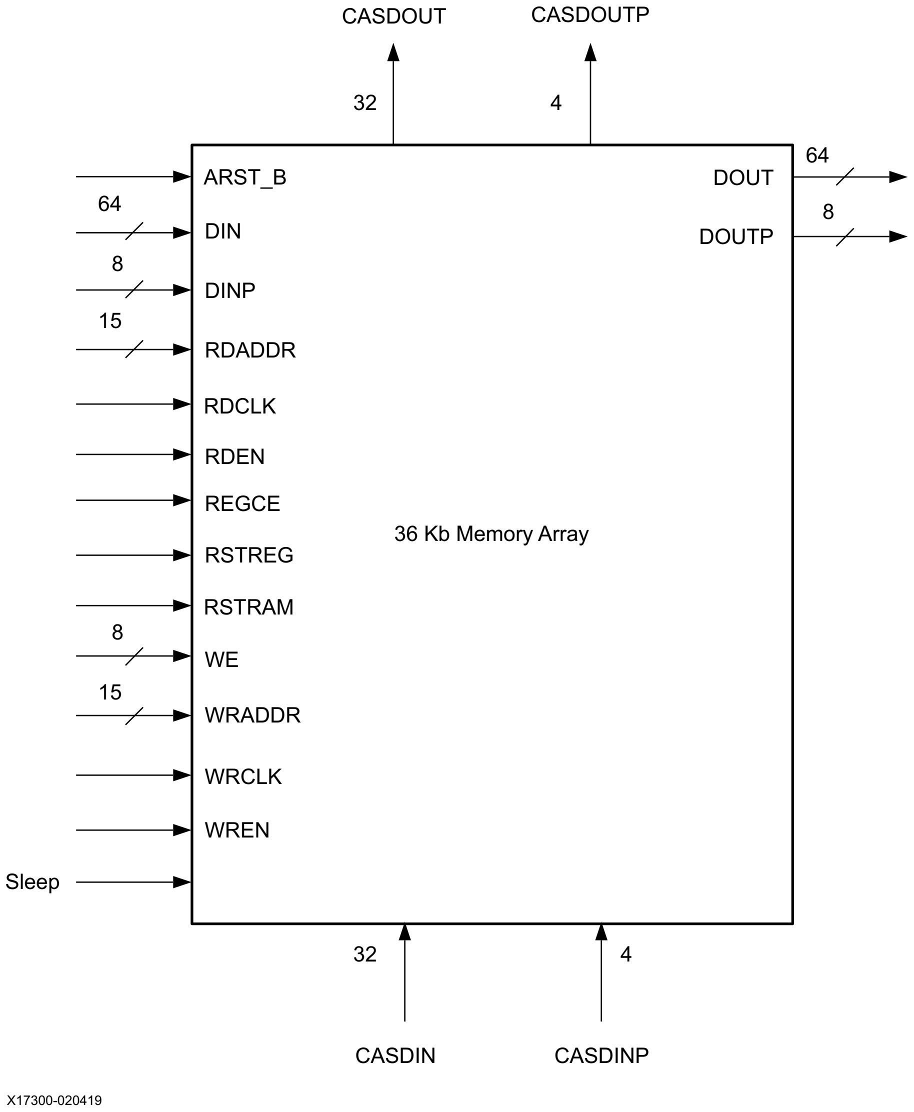


Figure 1-6: **RAMB36 Usage in a Simple Dual-Port Data Flow**

Table 1-5 lists the simple dual-port functions and descriptions.

Table 1-5: **Simple Dual-Port Functions and Descriptions**

Port Function	Description
DOUT	Data output bus.
DOUTP	Data output parity bus.
DIN	Data input bus.
DINP	Data input parity bus.
RDADDR	Read data address bus.
RDCLK	Read data clock.
RDEN	Read port enable.
REGCE	Output register clock enable.
RSTREG	Synchronous set/reset of the output registers.
RSTRAM	Synchronous set/reset of the output data latches.
WRADDR	Write data address bus.
WRCLK	Write data clock.
WREN	Write port enable.
SLEEP	Dynamic shutdown power saving. If Sleep is High, the block is in power-saving mode
CASDIN[A|B]	Cascade data input bus.
CASDINP[A|B]	Cascade parity input bus.
CASDOUT[A|B]	Cascade data output bus.
CASDOUTP[A|B]	Cascade parity output bus.
RDADDREN/WRADDREN	Address latching enable. If Low, the old address is latched.

Notes:

1. For a more complete cascade data flow and port descriptions, see Cascadable Block RAM, page 20 and Block RAM Library Primitives, page 28.

Cascadable Block RAM

UltraScale architecture-based devices provide the capability to cascade data out from one RAMB36 to the next RAMB36 serially to make a deeper block RAM in a bottom-up fashion. The data out cascading feature is supported for all RAMB36 port widths. The block RAM cascade supports all the features supported by the RAMB36E2 module.

Note: The 64K x 1 cascade functionality provided in previous architectures has been removed. The same 64K x 1 cascade feature can be achieved using the new cascade block RAM implementation.

The data flow is always from lower block RAM to upper block RAM. All of the signal routings and the control logic for the cascading feature are implemented in hardware. Multiple block RAMs can be cascaded, as required. In cascade mode, a single, common clock source must drive the same block RAM inputs (RDCLK or WRCLK). Furthermore, the data cascade capability allows that the lower RAMB18 of the lower RAMB36 can be independently cascaded to the lower RAMB18 of the upper RAMB36. Similarly, the upper RAMB18 of lower RAMB36 can be cascaded to the upper RAMB18 of the upper RAMB36 site.

IMPORTANT: *All block RAMs in a cascade chain must have matching configurations for certain features (e.g., common inputs such as the port width must be identical).*

Figure 1-7 shows a high-level, conceptual view of four cascaded block RAMs.

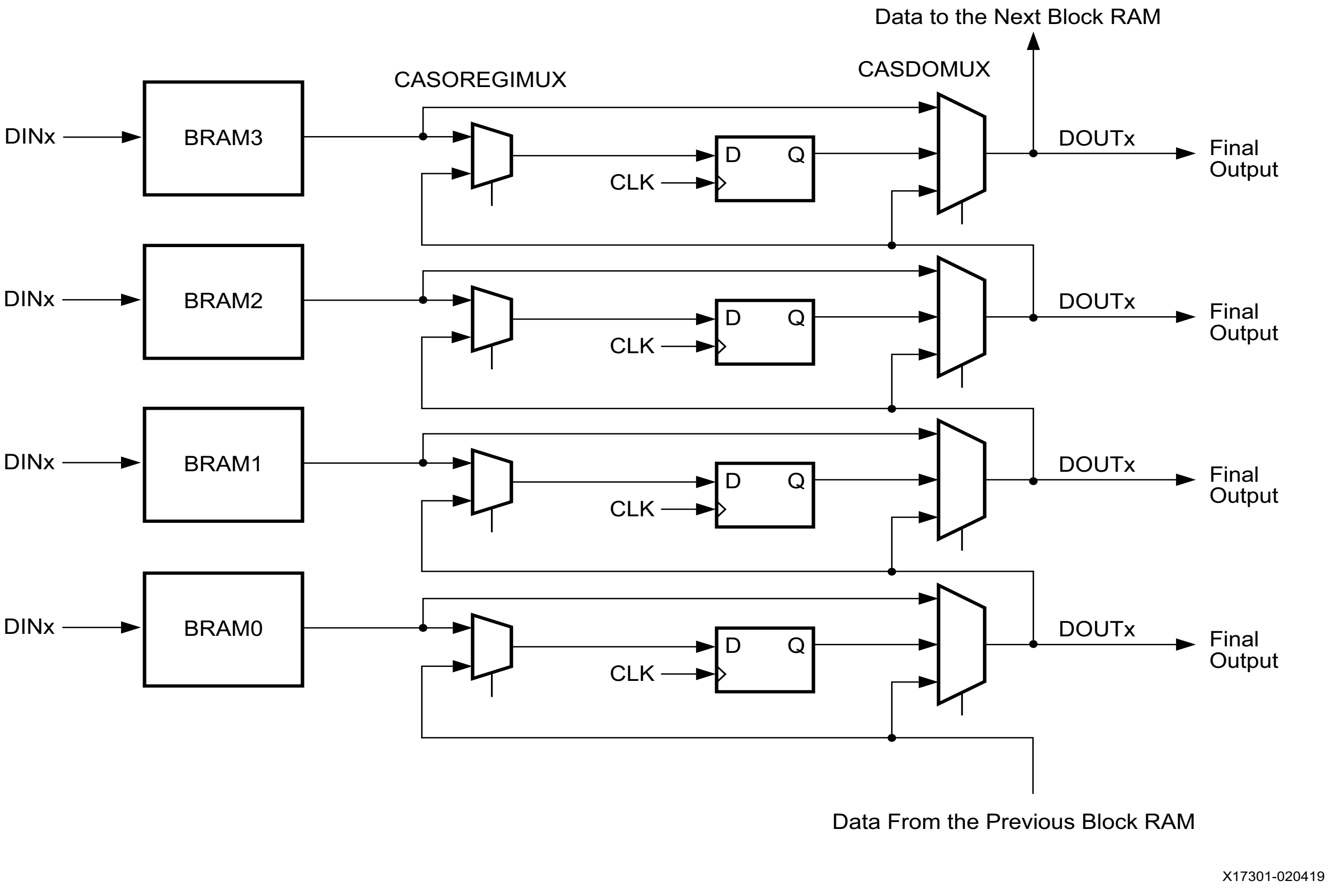


Figure 1-7: **High-level View of the Block RAM Cascade Architecture**

The block RAM provides flexibility to support many different implementations of the cascade feature. The three multiplexers (Figure 1-7) that select datapaths and pipeline registers can be dynamically controlled with the input pins.

Figure 1-8 shows a more detailed diagram of the functional implementation in a single block RAM block. Three cascade multiplexer selection pins are available when the block RAM is in cascade mode. CASDIMUX selects between either the cascade input data or the direct data input. CASOREGIMUX selects the data output of the block RAM or the cascaded data input to the block RAM's optional output register. This control pin allows pipelined cascading for maximum performance. CASDOMUX selects the data output of the block RAM (with or without the optional register) or the cascaded data input. The latter two cascade multiplexer select pins are registered at the input and have an enable control pin. CASDOUT and CASDIN have dedicated interconnects within a block RAM column. Both the cascade connections and data connection to and from the block RAM are available at the same time.

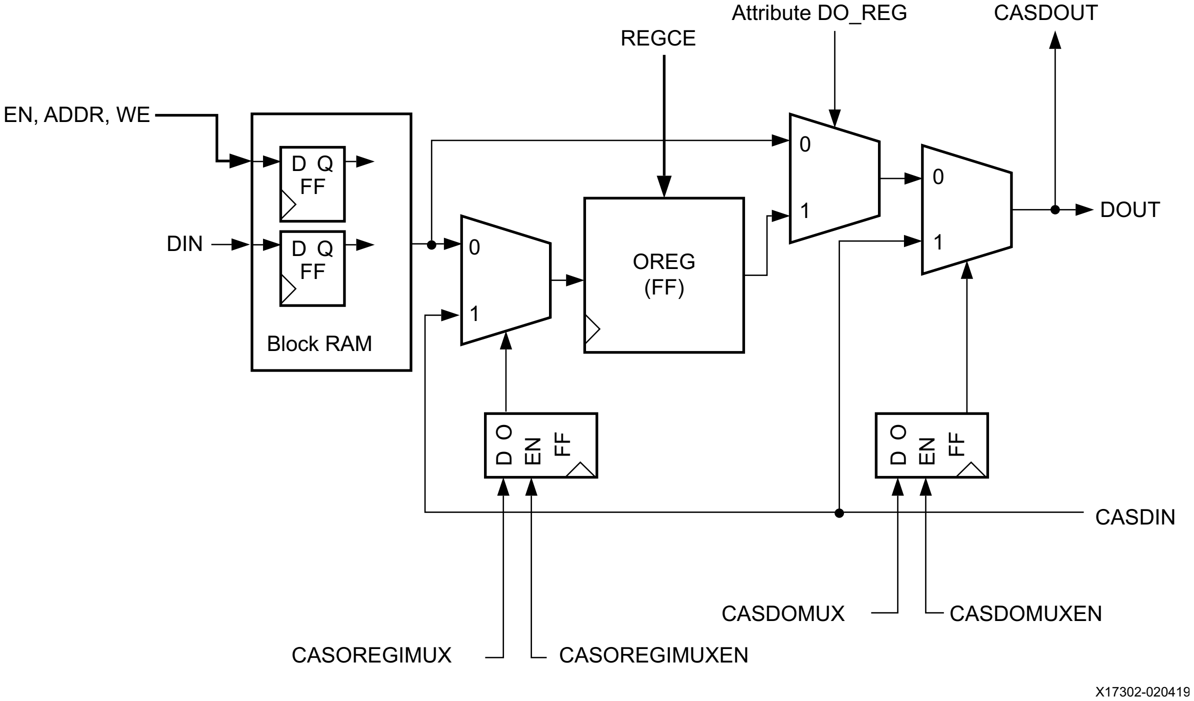


Figure 1-8: **Cascade Functional Diagram**

Although many different use cases can be implemented using the block RAM data cascade feature, this chapter describes three of the most common use cases. The examples shown are based on cascading three block RAM blocks, but more block RAM blocks can be cascaded with some limitations as required by the application in the same fashion.

Standard Data Output Cascade Mode

In this cascade use case, the data out of the lower block RAM is multiplexed to the final output multiplexer of the upper block RAM (Figure 1-9). The cascading can be applied to an entire block RAM column. This case yields a very deep RAM that can be implemented using only a few logic resources that might be required to drive the EN pins, drive the pins of the block RAM, determine the correct select value for the cascade muxes, and align the data if the DO_REG is used. The input multiplexer always selects DIN to write to the block RAM, the block RAM output multiplexer always selects the block RAM output data, and the last output multiplexer selects the current block RAM data (optionally registered) or the cascaded data from the block RAM below. The length of the block RAM chain impacts the final clock-to-out performance, which might slow down the performance depending on how many block RAMs are cascaded. All features of the block RAM are supported.

IMPORTANT: *The attribute CASCADE_ORDER defines the placement sequence within a block RAM column while the DO_REG attribute turns the optional block RAM register on or off.*

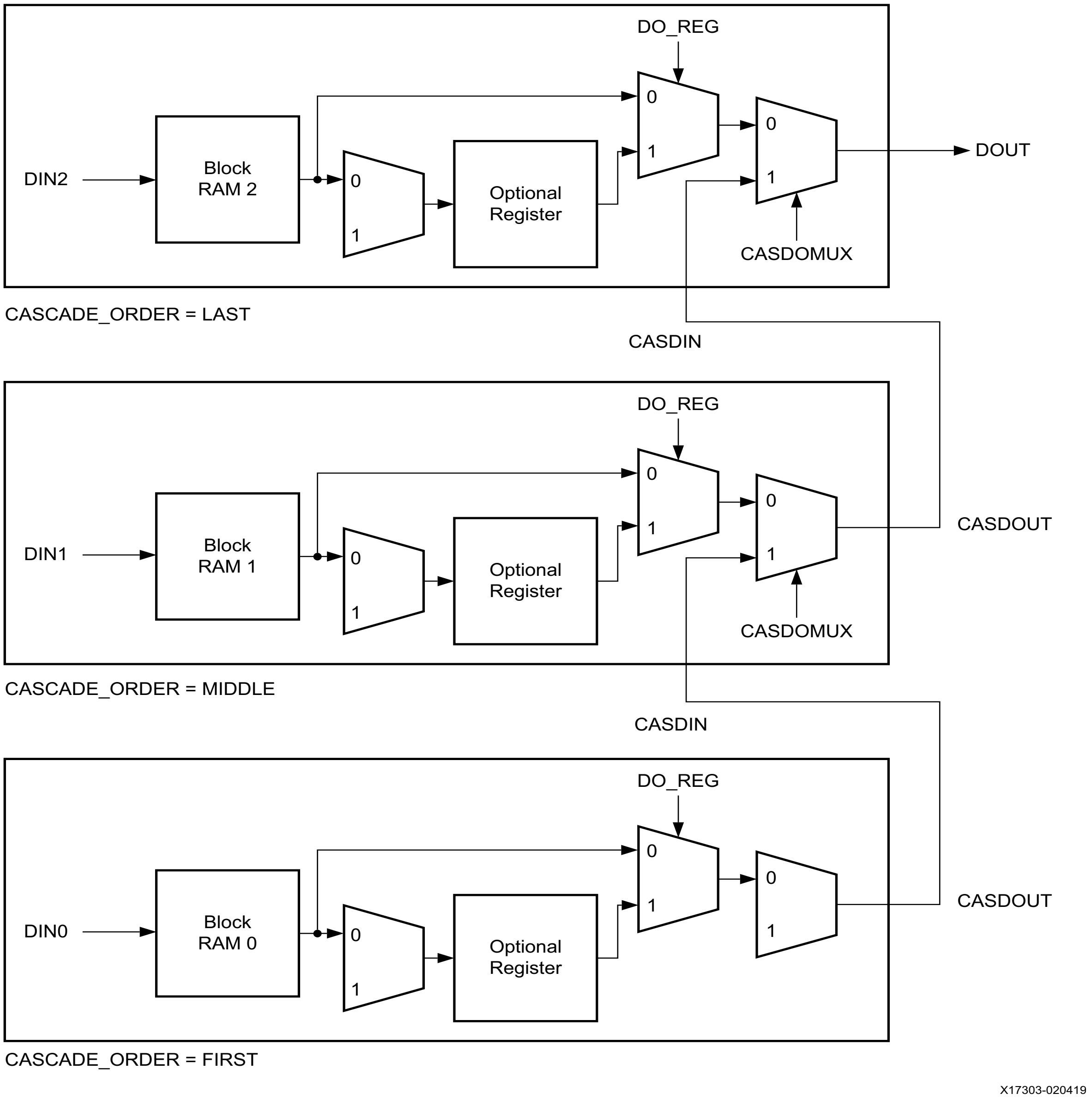


Figure 1-9: **Block RAM Cascade – Standard Data Out Cascade**

Data Out Cascade in Pipeline Mode

The block RAM pipeline cascade mode is similar to the standard data output cascade mode but allows the application to use the cascade mode at higher frequencies (Figure 1-10). The cascading data output propagates through the regular block RAM output registers because they are used as additional pipeline stages to achieve higher frequencies in this cascade mode. The external CASOREGIMUX pin controls the multiplexer that selects the input to the optional register. Thus, the data from the block RAM below or the current block RAM can be stored into the output register. The input multiplexer always selects DIN to write to the block RAM, the block RAM output multiplexer selects the block RAM output data, or the cascaded data from the block RAM below to write to the register. The final output multiplexer for each of the cascade stages always selects the data from the register for the final output data. All the DO_REG attributes have to be set to TRUE in this case. In this cascading mode, the length of the cascade chain is limited to within one clock region.

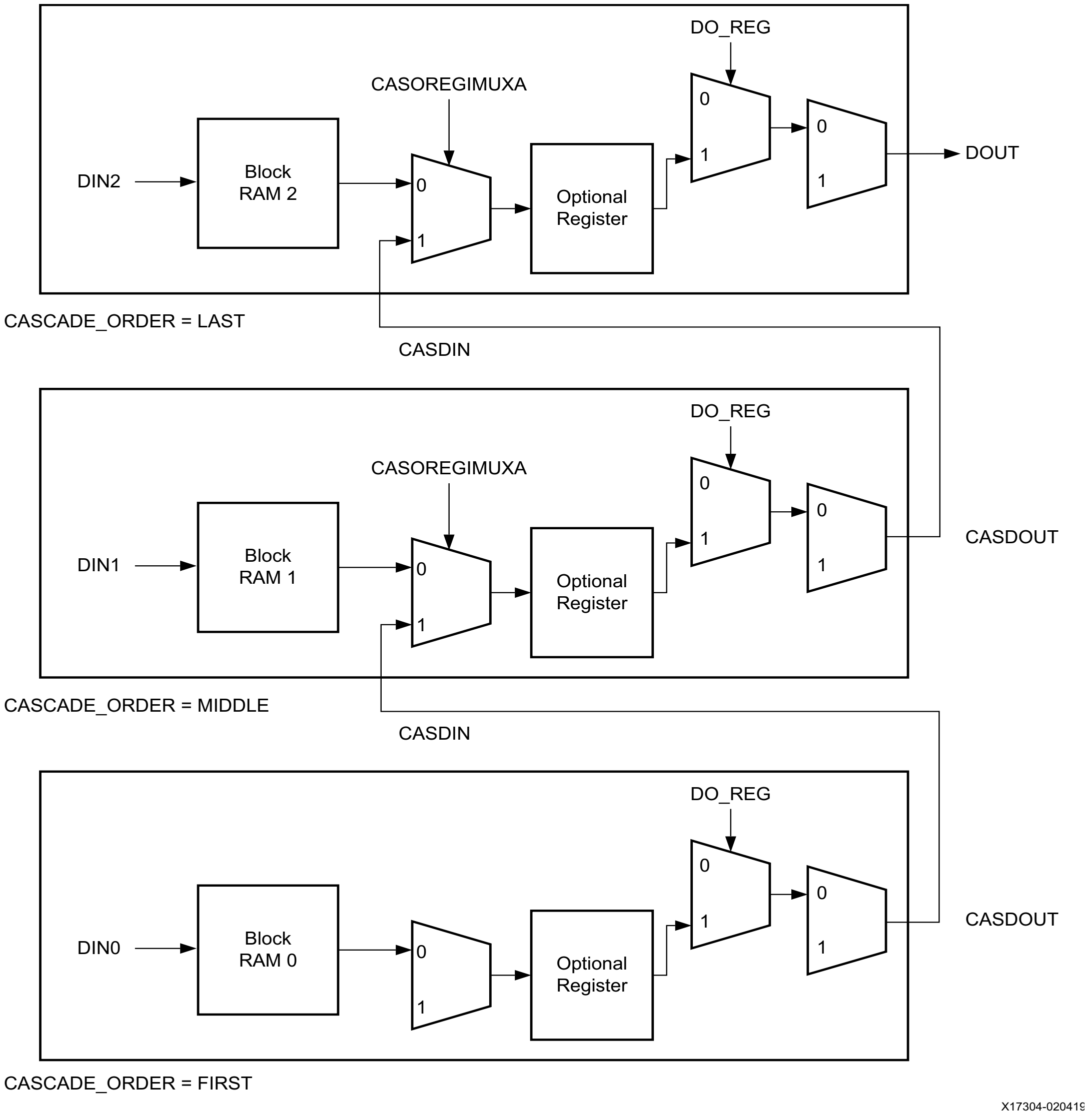


Figure 1-10: **Block RAM Cascade – Pipelined Data Out Cascade**

Data in Cascade for a Block RAM Array Matrix (Systolic) Mode

The block RAM systolic mode allows an application to write input data or cascaded data into a block RAM (Figure 1-11). At a later cycle, the application can then select to read data from a lower block RAM and write into the next upper block RAM. Data can be read from any dynamically selected block RAM in the cascade chain. The input multiplexer dynamically selects the DIN data or the cascaded data output from the lower block RAM to write to the current block RAM. The block RAM output multiplexer always selects the block RAM output data that is then presented on the data output directly or via the optional register. The DO_REG attribute determines if the optional register is used. In this cascade mode, the length of the cascade chain is limited to within one clock region.

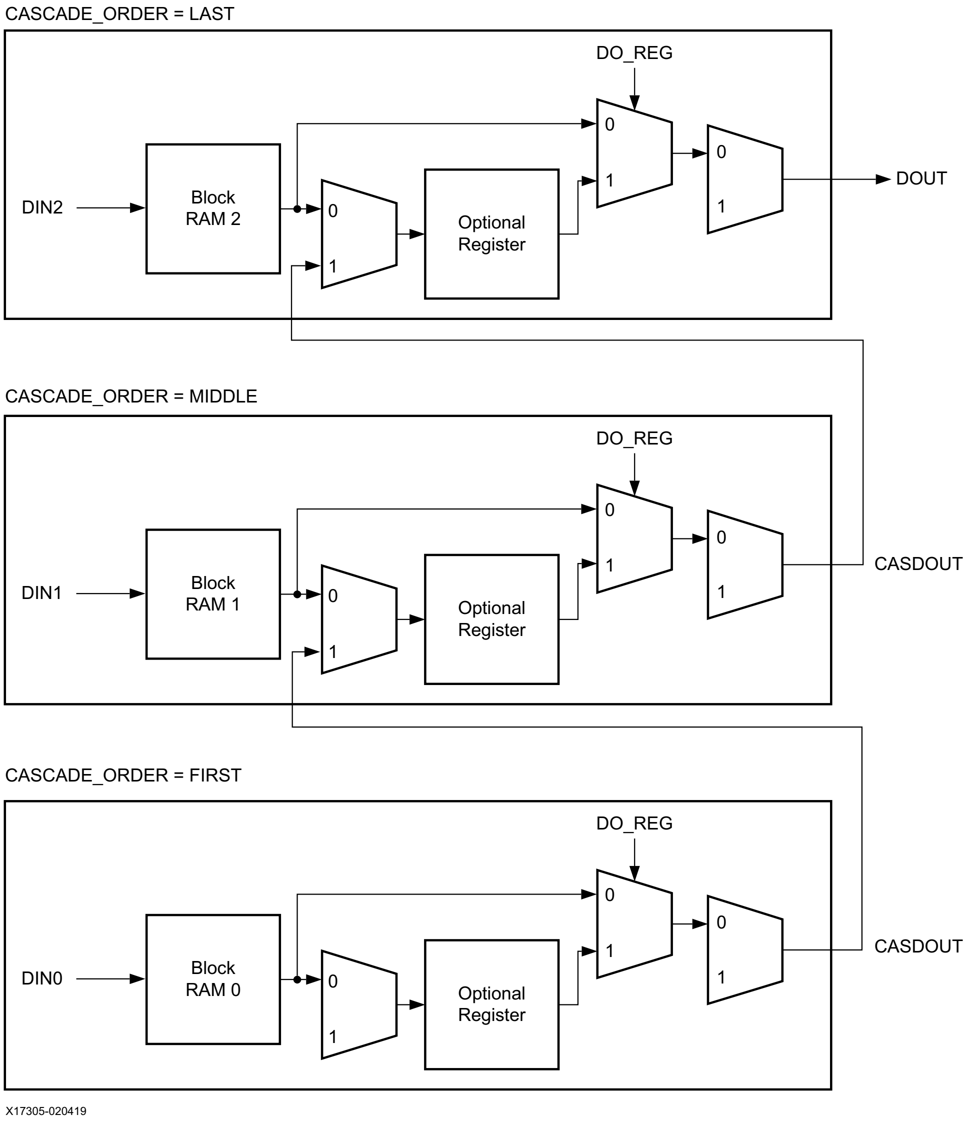


Figure 1-11: **Block RAM Cascade – Arrayed (Systolic) Data in Cascade**

Address Enable

This feature allows the new address to be captured only when address EN is High. If the address EN is Low, the old address remains internally latched and is used for internal access. Any change in address input is ignored. This feature is controlled by the ENADDREN attribute. See Figure 1-12.

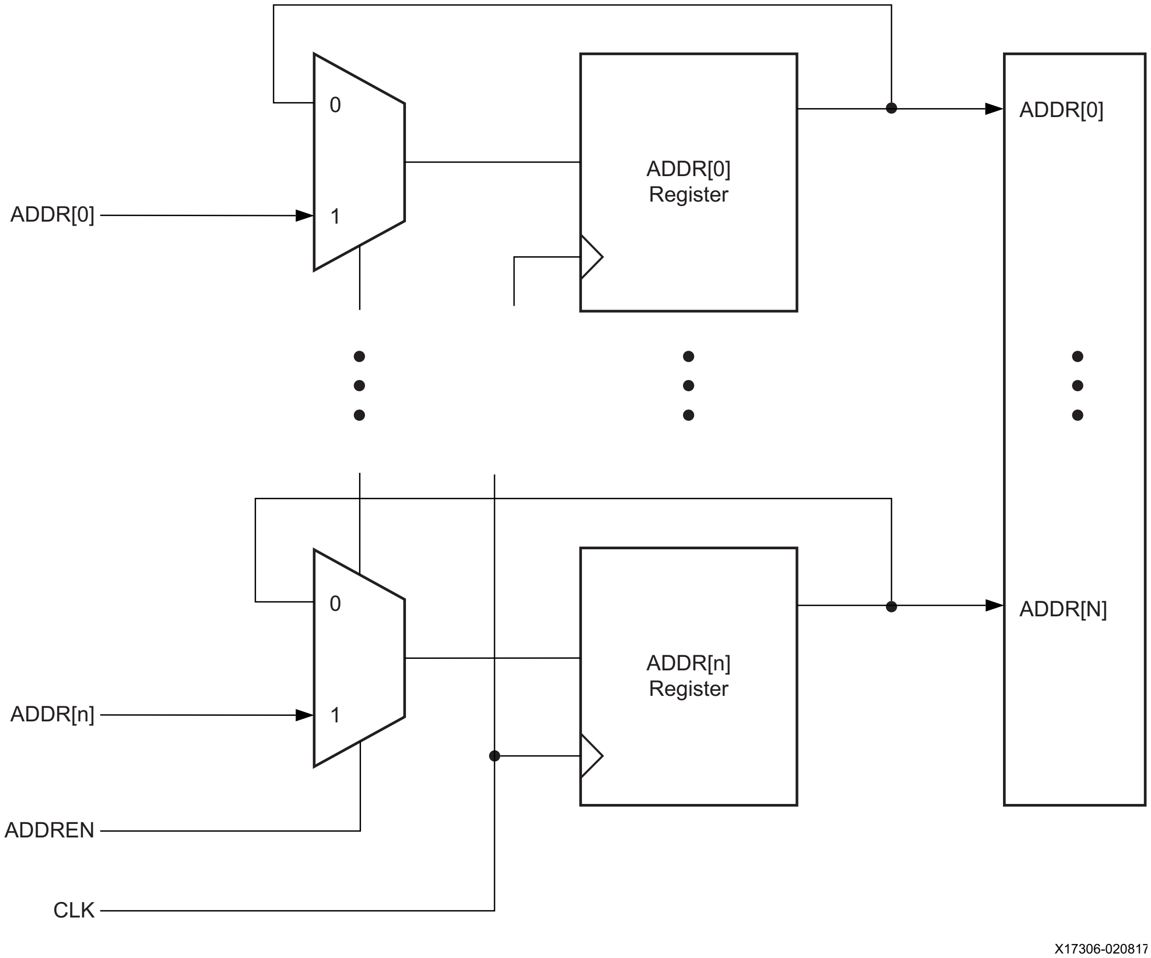


Figure 1-12: **Address Latching Enable**

Byte-Wide Write Enable

The byte-wide write enable feature of the block RAM enables the writing of eight-bit (one byte) portions of incoming data. There are four independent byte-wide write enable inputs to the RAMB36E2 true dual-port RAM. In TDP mode for RAMB36E2, there are two ports, A and B, each of which have a 4-bit write enable bus (one bit corresponding to each data byte). In SDP mode for RAMB36E2, there is one write port, which has an 8-bit write enable bus (one bit corresponding to each data byte). Table 1-6 summarizes the byte-wide write enables for the 36 Kb and 18 Kb block RAM. Each byte-wide write enable is associated with one byte of input data and one parity bit. The byte-wide write enable inputs must be driven in accordance with the data width configurations. This feature is useful when using block RAM to interface with a microprocessor. Byte-wide write enable is not available in the ECC mode. Byte-wide write enable is further described in Additional RAMB18E2 and RAMB36E2 Primitive Design Considerations, page 49. Figure 1-13 shows the byte-wide write enable timing diagram for the RAMB36E2.

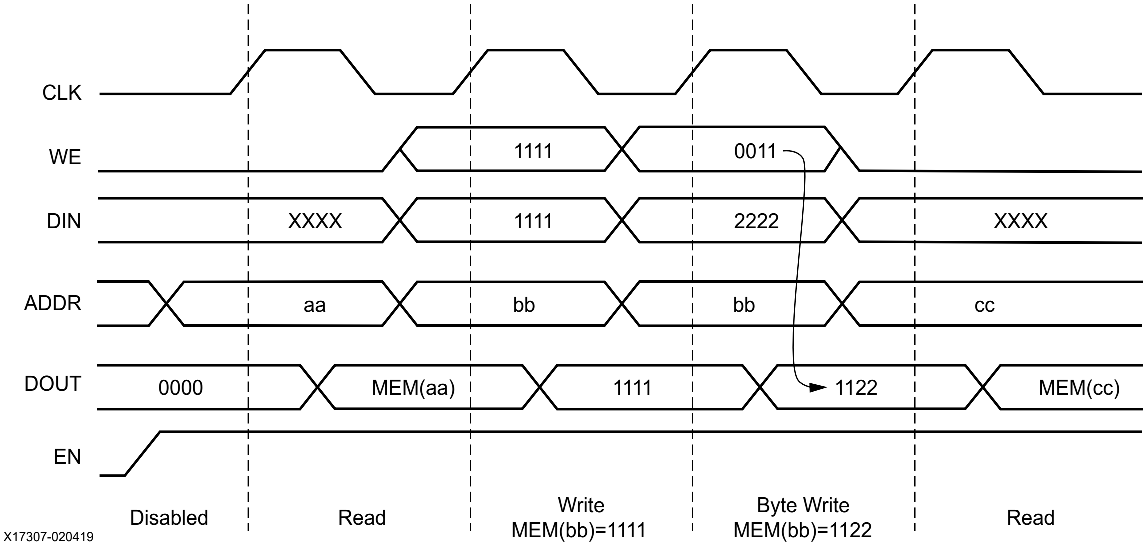


Figure 1-13: **Byte-wide Write Operation Waveforms (x36 WRITE_FIRST)**

Table 1-6: **Available Byte-Wide Write Enables**

Primitive	Maximum Bit Width	Number of Byte-Wide Write Enables
RAMB36E2 TDP usage	36	4
RAMB36E2 SDP usage	72	8
RAMB18E2 TDP usage	18	2
RAMB18E2 SDP usage	36	4

When the RAMB36E2 is configured for a 36-bit or 18-bit wide datapath, any port can restrict writing to specified byte locations within the data word. If configured in READ_FIRST mode, the DOUT bus shows the previous content of the whole addressed word. In WRITE_FIRST mode, DOUT shows a combination of the newly written enabled byte(s), and the initial memory contents of the unwritten bytes.

Block RAM Error Correction Code

Both block RAM and FIFO implementations of the 36 Kb block RAM support a 64-bit ECC implementation. The code is used to detect single- and double-bit errors in block RAM data read out. Single-bit errors are then corrected in the output data.

Power Gating of Unused Block RAMs

UltraScale architecture-based devices power down unused/uninstantiated block RAM blocks at an 18 Kb granularity. Power gating is enabled on every 18 Kb block that is not instantiated in the design to save power. Power-gated 18 Kb blocks are not initialized during configuration and retain their house keeping value of zero. A valid bitstream is required for configuration and readback. Blank bitstreams are not allowed. The access to uninstantiated block RAM is prevented by disabling the internal operation.

Block RAM Library Primitives

The block RAM library primitives, RAMB18E2 and RAMB36E2, are the basic building blocks for all block RAM configurations. Other block RAM primitives and macros are based on these primitives. Some block RAM attributes can only be configured using one of these primitives (e.g., pipeline register, cascade).

The input and output data buses are represented by two buses for 9-bit width (8 + 1), 18-bit width (16 + 2), and 36-bit width (32 + 4) configurations. The ninth bit associated with each byte can store parity/error correction bits or serve as additional data bits. No specific function is performed on the ninth bit. The separate bus for parity bits facilitates some designs. However, other designs safely use a 9-bit, 18-bit, or 36-bit bus by merging the regular data bus with the parity bus. Read/write and storage operations are identical for all bits, including the parity bits.

Figure 1-14 illustrates all the I/O ports of the 36 Kb true dual-port block RAM primitive (RAMB36). Table 1-7 lists these primitives.

Note: ECC pins are not shown in Figure 1-14. For more information, see Built-in Error Correction.

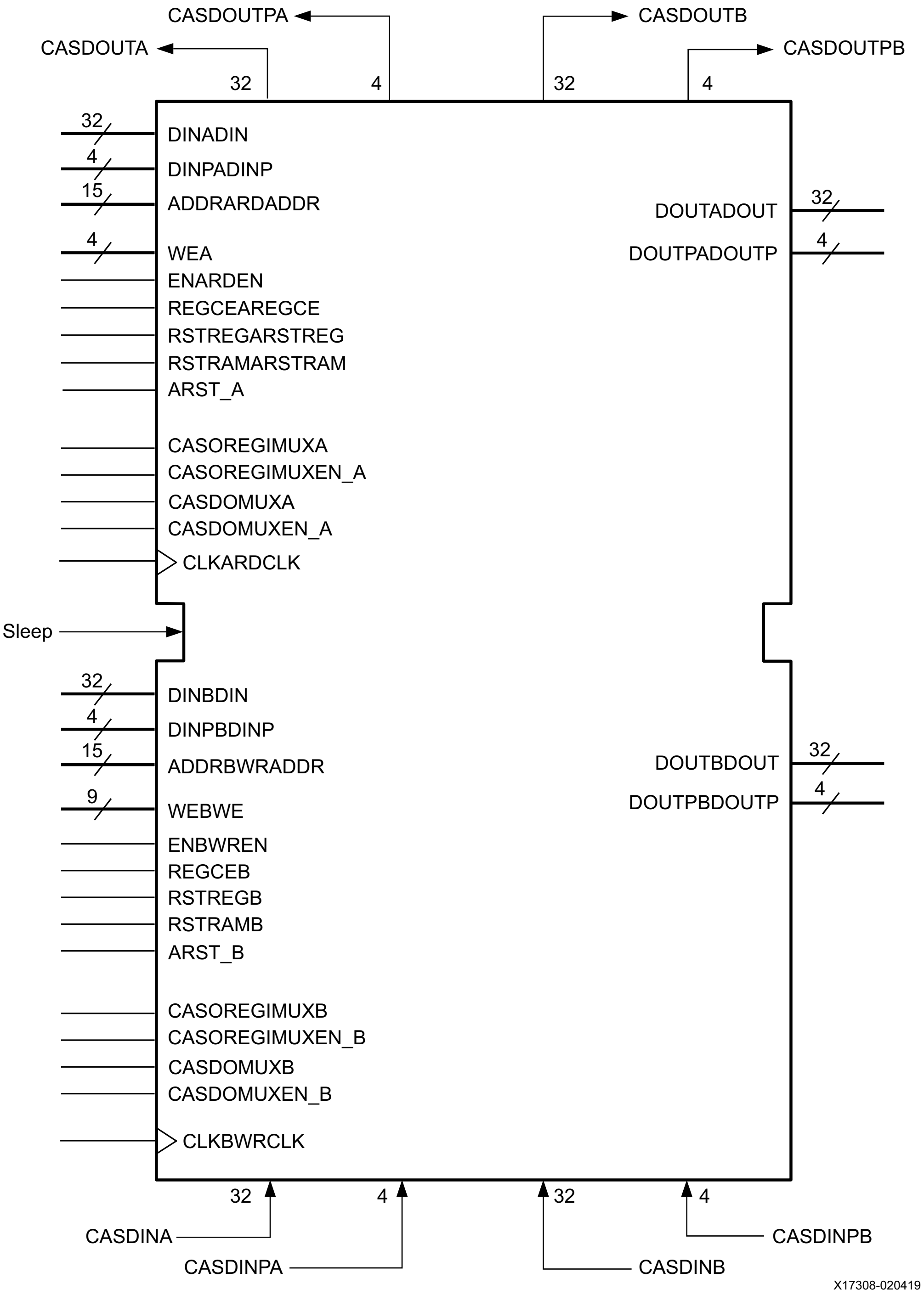


Figure 1-14: **Block RAM Port Signals (RAMB36E2)**

Table 1-7: **Block RAM and FIFO Primitives**

Primitive	Description
RAMB36E2	• When used as TDP memory, RAMB36E2 supports port widths of x1, x2, x4, x9, x18, and x36. • When used as SDP memory, the read or write port width is x64 or x72. Alternate port widths are x1, x2, x4, x9, x18, x36, and x72. In ECC mode, RAMB36E2 supports 64-bit ECC encoding and decoding.
RAMB18E2	• When used as TDP memory, RAMB18E2 supports port widths of x1, x2, x4, x9, and x18. • When used as SDP memory, the read or write port width is x32 or x36. Alternate port widths are x1, x2, x4, x9, x18, and x36.
FIFO36E2	FIFO36E2 supports port widths of x4, x9, x18, x36, and x72 for either port. When the port width is x72, ECC is optionally supported.
FIFO18E2	The FIFO18E2 supports port widths of x4, x9, x18, and x36 for either port.

Table 1-8 shows the port names and descriptions of the primitives outlined in Table 1-7. The ECC ports are described in Built-in Error Correction.

Table 1-8: **RAMB36E2 and RAMB18E2 Port Names and Descriptions**

Port Name	Description
DINADIN[31:0]	Port A data inputs addressed by ADDRARDADDR. See Table 1-13, page 35 for SDP usage port name mapping.
DINPADINP[3:0]	Port A data parity inputs addressed by ADDRARDADDR. See Table 1-13, page 35 for SDP usage port name mapping.
DINBDIN[31:0]	Port B data inputs addressed by ADDRBWRADDR. See Table 1-13, page 35 for SDP usage port name mapping.
DINPBDINP[3:0]	Port A data parity inputs addressed by ADDRBWRADDR. See Table 1-13, page 35 for SDP usage port name mapping.
ADDRARDADDR [14:0]	Port A address input bus. When used as SDP memory, this is the RDADDR bus.
ADDRBWRADDR[14:0]	Port B address input bus. When used as SDP memory, this is the WRADDR bus.
ADDRENA	ADDRENA enables or disables the capture of a new address on port A. When disabled (Low) the old, latched address is used.
ADDRENB	ADDRENB enables or disables the capture of a new address on port B. When disabled (Low) the old, latched address is used.
WEA[3:0]	Port A byte-wide write enable. When used as SDP memory, this port is not used.
WEBWE[7:0]	Port B byte-wide write enable. In SDP mode, this is the byte-wide write enable.
ENARDEN	Port A enable. When used as SDP memory, this is RDEN.
ENBWREN	Port B enable. When used as SDP memory, this is WREN.
RSTREGARSTREG	Synchronous output register set/reset as initialized by SRVAL_A (DOA_REG = 1). RSTREG_PRIORITY_A determines the priority over REGCE. When used as SDP memory, this is RSTREG.
RSTREGB	Synchronous output register set/reset as initialized by SRVAL_B (DOA_REG = 1). RSTREG_PRIORITY_B determines the priority over REGCE.

Table 1-8: **RAMB36E2 and RAMB18E2 Port Names and Descriptions *(Cont'd)***

Port Name	Description
RSTRAMARSTRAM	Synchronous output latch set/reset as initialized by SRVAL_A (DOB_REG = 0). When used as SDP memory, this is RSTRAM.
RSTRAMB	Synchronous output latch set/reset as initialized by SRVAL_B (DOB_REG = 0).
CLKARDCLK	Port A clock input. When used as SDP memory, this is RDCLK.
CLKBWRCLK	Port B clock input. When used as SDP memory, this is WRCLK.
REGCEAREGCE	Port A output register clock enable (DOA_REG = 1). When used as SDP memory, this is REGCE.
REGCEB	Port B output register clock enable (DOB_REG = 1).
CASDINA[31:0]	Port A cascade data input connected to data output of lower block RAM. For RAMB18E2: CASDINA[15:0].
CASDINPA[3:0]	Port A cascade parity data input connected to parity data output of lower block RAM. For RAMB18E2: CASDINPA[1:0].
CASDINB[31:0]	Port B cascade data input connected to data output of lower block RAM. For RAMB18E2: CASDINB[15:0].
CASDINPB[3:0]	Port B cascade parity data input connected to parity data output of lower block RAM. For RAMB18E2: CASDINPB[1:0].
CASDOUTA[31:0]	Port A cascade data output connected to CASDINA[31:0] of upper block RAM. For RAMB18E2: CASDOUTA[15:0].
CASDOUTPA[3:0]	Port A cascade parity data output connected to CASDINPA[3:0] of upper block RAM. For RAMB18E2: CASDOUTPA[1:0].
CASDOUTB[31:0]	Port B cascade data output connected to CASDINB[31:0] of upper block RAM. For RAMB18E2: CASDOUTB[15:0].
CASDOUTPB[3:0]	Port B cascade parity data output connected to CASDINPB[3:0] of upper block RAM. For RAMB18E2: CASDOUTPB[1:0].
CASDOMUXA	Selects input to control the data cascade output multiplexer for port A.
CASDOMUXEN_A	Enables control for the CASDOMUXA register.
CASDOMUXB	Selects input to control the data cascade output multiplexer for port B.
CASDOMUXEN_B	Enables control for the CASDOMUXB register. When used as SDP memory, this port is not used.
CASOREGIMUXA	Selects input to control the cascade multiplexer before the output register for Port A.
CASOREGIMUXEN_A	Enables control for the CASOREGIMUXA register.
CASOREGIMUXB	Selects input to control the cascade multiplexer before the output register for Port B. When used as SDP memory, this port is not used.
CASOREGIMUXEN_B	Enables control for the CASOREGIMUXB register. When used as SDP memory, this port is not used.
CASDIMUXA	Selects input to control the cascade DIN multiplexer for Port A.
CASDIMUXB	Selects input to control the cascade DIN multiplexer for Port B. When used as SDP memory, this port is not used.

Table 1-8: **RAMB36E2 and RAMB18E2 Port Names and Descriptions *(Cont'd)***

Port Name	Description
DOUTADOUT[31:0]	Port A data output bus addressed by ADDRARDADDR. See Table 1-13, page 35 for SDP usage port name mapping. RAMB18E2: DOUTADOUT[15:0].
DOUTPADOUTP[3:0]	Port A parity output bus addressed by ADDRARDADDR. See Table 1-13, page 35 for SDP usage port name mapping. RAMB18E2: DOUTPADOUTP[1:0].
DOUTBDOUT[31:0]	Port B data output bus addressed by ADDRBWRADDR. See Table 1-13, page 35 for SDP usage port name mapping. RAMB18E2: DOUTBDOUT[15:0].
DOUTPBDOUTP[3:0]	Port B parity output bus addressed by ADDRBWRADDR. See Table 1-13, page 35 for SDP usage port name mapping. RAMB18E2: DOUTPBDOUTP[1:0].
SLEEP	Dynamic power gating.

Block RAM Port Signals

Each block RAM port operates independently of the other while accessing the same set of 36 Kbit memory cells.

Clock – CLKARDCLK and CLKBWRCLK

Each port is fully synchronous with independent clock pins. All port input pins have setup time referenced to the port CLK pin. The output data bus has a clock-to-out time referenced to the CLK pin. Clock polarity is configurable (rising edge by default). When used as SDP memory, the CLKA port is the RDCLK and the CLKB port is the WRCLK.

Enable – ENARDEN and ENBWREN

The enable pin affects the read, write, and set/reset functionality of the port. Ports with an inactive enable pin keep the output pins in the previous state and do not write data to the memory cells. Enable polarity is configurable (active-High by default). When used as SDP memory, the ENA port is the RDEN and the ENB port is the WREN.

Byte-Wide Write Enable – WEA and WEBWE

To write the content of the data input bus into the addressed memory location, both EN and WE must be active within a setup time before the active clock edge. The output latches are loaded or not loaded according to the write configuration (WRITE_FIRST, READ_FIRST, NO_CHANGE). When WE is inactive and EN is active, a read operation occurs, and the contents of the memory cells referenced by the address bus appear on the data-out bus, regardless of the write mode attribute. Write enable polarity is not configurable (active-High). When used as SDP memory, the WEBWE[7:0] port is the byte-write enable. When used as TDP memory, the WEA[3:0] and WEB[3:0] are byte-write enables for port A and port B, respectively. See also Byte-Wide Write Enable, page 50.

Register Enable – REGCEAREGCE and REGCEB

The register enable pin (REGCE) controls the optional output register. When the block RAM is in register mode, REGCE = 1 registers the output into a register at a clock edge. The polarity of REGCE is not configurable (active-High). When used as SDP memory, the REGCEA port is the REGCE.

Set/Reset

RSTREGARSTREG, RSTREGB, RSTRAMARSTRAM, and RSTRAMB

In latch mode, the RSTRAM pin synchronously forces the data output latches to contain the value SRVAL. When the optional output registers are enabled (DO_REG = 1), the RSTREG signal synchronously forces the data output registers containing the SRVAL value. The priority of RSTREG over REGCE is determined using the RSTREG_PRIORITY attribute. The data output latches or output registers are synchronously asserted to 0 or 1, including the parity bit. Each port has an independent SRVAL[A|B] attribute of 36 bits. This operation does not affect RAM memory cells and does not disturb write operations on the other port. The polarity for both signals is configurable (active-High by default). When used as SDP memory, the RSTREGA port is the RSTREG, and the RSTRAMA port is the RSTRAM.

Address Bus – ADDRARDADDR and ADDRBWRADDR

The address bus selects the memory cells for read or write. When used as SDP memory, the ADDRA port is the RDADDR and the ADDRB port is the WRADDR. The data bit width of the port determines the required address bus width for a single RAMB18E2 or RAMB36E2, as shown in Table 1-9, Table 1-10, Table 1-11, and Table 1-12.

Table 1-9: **Port Aspect Ratio for RAMB18E2 (When Used as TDP Memory)**

Port Data Width	Port Address Width	Depth	ADDR Bus	DIN Bus DOUT Bus	DINP Bus DOUTP Bus
1	14	16,384	[13:0]	[0]	NA
2	13	8,192	[13:1]	[1:0]	NA
4	12	4,096	[13:2]	[3:0]	NA
9	11	2,048	[13:3]	[7:0]	[0]
18	10	1,024	[13:4]	[15:0]	[1:0]

Table 1-10: **Port Aspect Ratio for RAMB18E2 (When Used as SDP Memory)**

Port Data Width[(1)]	Alternate Port Width	Port Address Width	Depth	ADDR Bus	DIN Bus DOUT Bus	DINP Bus DOUTP Bus
32	1	14	16,384	[13:0]	[0]	NA
32	2	13	8,192	[13:1]	[1:0]	NA
32	4	12	4,096	[13:2]	[3:0]	NA
36	9	11	2,048	[13:3]	[7:0]	[0]
36	18	10	1,024	[13:4]	[15:0]	[1:0]
36	36	9	512	[13:5]	[31:0]	[3:0]

Notes:

1. Either the read or write port is a fixed width of x32 or x36.

Table 1-11: **Port Aspect Ratio for RAMB36E2 (When Used as TDP Memory)**

Port Data Width	Port Address Width	Depth	ADDR Bus	DIN Bus DOUT Bus	DINP Bus DOUTP Bus
1	15	32,768	[14:0]	[0]	NA
2	14	16,384	[14:1]	[1:0]	NA
4	13	8,192	[14:2]	[3:0]	NA
9	12	4,096	[14:3]	[7:0]	[0]
18	11	2,048	[14:4]	[15:0]	[1:0]
36	10	1,024	[14:5]	[31:0]	[3:0]
1 (Cascade)	16	65,536	[15:0]	[0]	NA

Table 1-12: **Port Aspect Ratio for RAMB36E2 (When Used as SDP Memory)**

Port Data Width[(1)]	Alternate Port Width	Port Address Width	Depth	ADDR Bus	DIN Bus DOUT Bus	DINP Bus DOUTP Bus
64	1	15	32,768	[14:0]	[0]	NA
64	2	14	16,384	[14:1]	[1:0]	NA
64	4	13	8,192	[14:2]	[3:0]	NA
72	9	12	4,096	[14:3]	[7:0]	[0]
72	18	11	2,048	[14:4]	[15:0]	[1:0]
72	36	10	1,024	[14:5]	[31:0]	[3:0]
72	72	9	512	[14:6]	[63:0]	[7:0]

Notes:

1. Either the read or write port is a fixed width of x64 or x72.

For block RAMs used as SDP memories, the port name mapping is listed in Table 1-13. Figure 1-6 shows the SDP data flow.

Table 1-13: **Port Name Mapping for Block RAMs Used as SDP Memories**

RAMB18E2 Used as SDP Memory		RAMB36E2 Used as SDP Memory	
X36 Mode (Width = 36)	**X18 Mode (Width ≤ 18)**	**X72 Mode (Width = 72)**	**X36 Mode (Width ≤ 36)**
DIN[15:0] = DINADIN[15:0]	DIN[15:0] = DINBDIN[15:0]	DIN[31:0] = DINADIN[31:0]	DIN[31:0] = DINBDIN[31:0]
DINP[1:0] = DINPADIN[1:0]	DINP[1:0] = DINPBDINP[1:0]	DINP[3:0] = DINPADIN[3:0]	DINP[3:0] = DINPBDINP[3:0]
DIN[31:16] = DINBDIN[15:0]		DIN[63:32] = DINBDIN[31:0]	
DINP[3:2] = DINPBDINP[1:0]		DINP[7:4] = DINPBDINP[3:0]	
DOUT[15:0] = DOUTADOUT[15:0]	DOUT[15:0] = DOUTADOUT[15:0]	DOUT[31:0] = DOUTADOUT[31:0]	DOUT[31:0] = DOUTADOUT[31:0]
DOUTP[1:0] = DOUTPADOUTP[1:0]	DOUTP[1:0] = DOUTPADOUTP[1:0]	DOUTP[3:0] = DOUTPADOUTP[3:0]	DOUTP[3:0] = DOUTPADOUTP[3:0]
DOUT[31:16] = DOUTBDOUT[15:0]		DOUT[63:32] = DOUTBDOUT[31:0]	
DOUTP[3:2] = DOUTPBDOUTP[1:0]		DOUTP[7:4] = DOUTPBDOUTP[3:0]	

Data-In Buses – DINADIN, DINPADINP, DINBDIN, and DINPBDINP

Data-in buses provide the new data value to be written into RAM. The regular data-in bus (DIN), plus the data-in parity bus (DINP), when available, have a total width equal to the port width. For example, the 36-bit port data width is represented by DIN[31:0] and DINP[3:0], as shown in Table 1-9, page 33 through Table 1-12. See Table 1-13 for port name mapping for block RAMs used as SDP memories.

Data-Out Buses – DOUTADOUT, DOUTPADOUTP, DOUTBDOUT, and DOUTPBDOUTP

Data-out buses reflect the contents of memory cells referenced by the address bus at the last active clock edge during a read operation. During a write operation (WRITE_FIRST or READ_FIRST configuration), the data-out buses reflect either the data being written or the stored value before write. During a write operation in NO_CHANGE mode, data-out buses are not changed. The regular data-out bus (DOUT) plus the parity data-out bus (DOUTP) (when available) have a total width equal to the port width, as shown in Table 1-9, page 33 through Table 1-12, page 34. See Table 1-13, page 35 for port name mapping for block RAMs used as SDP memories.

ADDRENA

ADDRENA enables latching of the A port address. When the block RAM is enabled and ADDRENA is Low, the old address remains latched in the block RAM. If High, the address is captured and active. This feature is controlled by the ENADDRENA attribute. In SDP mode, the ADDRENA port is the RDADDREN.

ADDRENB

ADDRENB enables latching of the B port address. When the block RAM is enabled and ADDRENB is Low, the old address remains latched in the block RAM. If High, the address is captured and active. This feature is controlled by the ENADDRENB attribute. In SDP mode, the ADDRENB port is the WRADDREN.

CASDINA

This is the data input cascade for port A from the block RAM below.

Note: For further details on cascading, see Cascadable Block RAM, page 20.

CASDINB

This is the data input cascade for port B from the block RAM below.

CASDINPA

This is the parity input cascade for port A from the block RAM below.

CASDINPB

This is the parity input cascade for port B from the block RAM below.

CASDOUTA

This is the data output cascade for port A to the block RAM above.

CASDOUTB

This is the data output cascade for port B to the block RAM above.

CASDOUTPA

This is the parity output cascade for port A to the block RAM above.

CASDOUTPB

This is the parity output cascade for port B to the block RAM above.

Cascade Selection – CASDIMUX

This is the input multiplexer select line to select between regular data input (DIN) or cascade data input (CASDIN) when the block RAM is in cascade mode. When the block RAM is not used in cascade mode, DIN is always selected.

Cascade Selection – CASOREGIMUX

This is the D input to the register that drives the multiplexer select line to select between regular data from the block RAM output or the cascade input (CASDIN) when the block RAM is in cascade mode. This multiplexer is before the optional output register and adds a pipeline stage in cascade mode. When the block RAM is not used in cascade mode, block RAM data is always selected.

Cascade Selection – CASOREGIMUXEN

This is the enable control input to the register that drives the multiplexer select line to select between regular data from the block RAM output or the cascade input (CASDIN).

Cascade Selection – CASDOMUX

This is the register D input that drives the output multiplexer select line to select between regular data from the block RAM output or the cascade input (CASDIN) when the block RAM is in cascade mode. This multiplexer is after the optional output register. When the block RAM is not used in cascade mode, block RAM data is always selected.

Cascade Selection – CASDOMUXEN

This is the enable control input to the register that drives the select line to the cascade output multiplexer of the block RAM outputs in cascade mode.

SLEEP

The SLEEP pin provides a dynamic power gating capability for periods when the block RAM is not actively used for an extended period of time. While SLEEP is active (High) the EN pins on both ports must be held Low. The data content of the memory is preserved during this mode. There is a wake-up time requirement of two clock cycles regardless of the SLEEP_ASYNC mode setting. Any block RAM access prior to the wake-up time requirement is not guaranteed and might cause memory content corruption. The attribute SLEEP_ASYNC determines the behavior of this pin with respect to the clocks. For more details, see Block RAM Attributes, page 40.

Inverting Control Pins

For each port, the eight control pins (CLK, EN, RSTREG, and RSTRAM) each have an individual inversion option. EN, RSTREG, and RSTRAM control signals can be configured as active-High or Low, and the clock can be active on a rising or falling edge (active-High on rising edge by default), without requiring other logic resources.

RAMB18/36 Unused Inputs

Unused input pins require a certain, defined constant input value for the block RAM to function properly. If left unconnected (Verilog), the Vivado tools automatically tie them to the appropriate constant value. However, if the inputs are connected to a constant in the design (VHDL), then the values listed in Table 1-14 are required.

The unused inputs are shown in Table 1-14.

Table 1-14: **RAMB18/36 Unused Inputs**

RAMB18/36	Constant	Comments
ADDRENA	1	
ADDRENB	1	
CLKARDCLK	0	
CLKBRDCLK	0	
CLKAWRCLK	0	
CLKBWRCLK	0	
ENARDEN	0	
ENBWREN	0	
REGCEAREGCE	1	Xilinx recommends setting to 0 when DOA_REG = 0 for power saving
REGCEB	1	Xilinx recommends setting to 0 when DOB_REG = 0 for power saving
REGCLKARDRCLK	0	
REGCLKB	0	

Table 1-14: **RAMB18/36 Unused Inputs *(Cont'd)***

RAMB18/36	Constant	Comments
RSTREGARSTREG	0	
RSTREGB	0	
RSTRAMARSTRAM	0	
RSTRAMB	0	
RSTRAMARSTRAM	0	
RSTRAMB	0	
SLEEP	0	
WEA<3:0>	1	TDP: When not using port A for write, (WRITE_WIDTH_A = 0), WEA<0> must be connected to 0
WEBWE<7:0>	1	TDP: When not using port B for write (WRITE_WIDTH_B=0), WEB<0> must be connected to 0
CASDOMUXA	0	
CASDOMUXB	0	
CASOREGIMUXA	0	
CASOREGIMUXB	0	
CASDIMUXA	0	
CASDIMUXB	0	
CASDOMUXEN_A	1	
CASDOMUXEN_B	1	
CASOREGIMUXEN_A	1	
CASOREGIMUXEN_B	1	
INJECTSBITERR	0	
INJECTDBITERR	0	

Block RAM Address Mapping

Each port accesses the same set of 18,432 or 36,864 memory cells using an addressing scheme dependent on whether it is a RAMB18E2 or RAMB36E2. The physical RAM locations addressed for a particular width are determined using these formulae (of interest only when the two ports use different aspect ratios):

$$\text{END} = ((\text{ADDR} + 1) \times \text{Width}) - 1$$

$$\text{START} = \text{ADDR} \times \text{Width}$$

Table 1-15 shows low-order address mapping for each port width.

Table 1-15: **Port Address Mapping**

<table>
<tr><th>Port Width</th><th colspan="4">Parity Locations</th><th colspan="32">Data Locations</th></tr>
<tr><td>1</td><td colspan="4" rowspan="3">N.A.</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>2</td><td colspan="2">15</td><td colspan="2">14</td><td colspan="2">13</td><td colspan="2">12</td><td colspan="2">11</td><td colspan="2">10</td><td colspan="2">9</td><td colspan="2">8</td><td colspan="2">7</td><td colspan="2">6</td><td colspan="2">5</td><td colspan="2">4</td><td colspan="2">3</td><td colspan="2">2</td><td colspan="2">1</td><td colspan="2">0</td></tr>
<tr><td>4</td><td colspan="4">7</td><td colspan="4">6</td><td colspan="4">5</td><td colspan="4">4</td><td colspan="4">3</td><td colspan="4">2</td><td colspan="4">1</td><td colspan="4">0</td></tr>
<tr><td>8 + 1</td><td>3</td><td>2</td><td>1</td><td>0</td><td colspan="8">3</td><td colspan="8">2</td><td colspan="8">1</td><td colspan="8">0</td></tr>
<tr><td>16 + 2</td><td colspan="2">1</td><td colspan="2">0</td><td colspan="16">1</td><td colspan="16">0</td></tr>
<tr><td>32 + 4</td><td colspan="4">0</td><td colspan="32">0</td></tr>
</table>

Block RAM Attributes

Table 1-16 lists the RAMB18E2 and RAMB36E2 attributes. All attribute code examples are discussed in Block RAM Initialization in VHDL or Verilog Code, page 48. Further information on using these attributes is available in Additional RAMB18E2 and RAMB36E2 Primitive Design Considerations, page 49.

Table 1-16: **RAMB18E2 and RAMB36E2 Attributes**

Attributes	Values	Default	Type	Description
CASCADE_ORDER_A	FIRST, MIDDLE, LAST, NONE	NONE	String	Specifies the order of the cascaded block RAMs from the bottom to the top of the chain for port A.
CASCADE_ORDER_B	FIRST, MIDDLE, LAST, NONE	NONE	String	Specifies the order of the cascaded block RAMs from the bottom to the top of the chain for port B.
CLOCK_DOMAINS	INDEPENDENT, COMMON	INDEPENDENT	String	Either independent clocks connected to port A and B or a single, common clock connected to port A and B.
DOA_REG	0, 1	1	Decimal	A value of 1 enables the optional output registers of the RAM port A. Applies to all port A outputs in both TDP and SDP memory usage.
DOB_REG	0, 1	1	Decimal	A value of 1 enables the optional output registers of the RAM port B. Applies to all port B outputs in both TDP and SDP memory usage.

Table 1-16: **RAMB18E2 and RAMB36E2 Attributes *(Cont'd)***

Attributes	Values	Default	Type	Description
ENADDRENA	FALSE, TRUE	FALSE	String	Specifies if the address enable pin on port A is enabled.
ENADDRENB	FALSE, TRUE	FALSE	String	Specifies if the address enable pin on port B is enabled.
INIT_A	RAMB18E2: 18-bit hex value RAMB36E2: 36-bit hex value	RAMB18E2: `18'h00000000` RAMB36E2: `36'h00000000 00000000`	Hex	Specifies the initial value of the port A outputs after configuration. Applies to all port A outputs in both TDP and SDP memory usage.
INIT_B	RAMB18E2: 18-bit hex value RAMB36E2: 36-bit hex value	RAMB18E2: `18'h00000000` RAMB36E2: `36'h00000000 00000000`	Hex	Specifies the initial value of the port B outputs after configuration. Applies to all port B outputs in both TDP and SDP memory usage.
RAMB18E2: INIT_00 to INIT_3F RAMB36E2: INIT_00 to INIT_7F	A 256-bit hex value	All 0	Hex	Initializes the data content of the block RAM.
RAMB18E2: INITP_00 to INITP_07 RAMB36E2: INITP_00 to INITP_0F	A 256-bit hex value	All 0	Hex	Initializes the parity content of the block RAM.
RDADDRCHANGEA(1)	FALSE, TRUE	FALSE	String	Specifies if the port A read address compare feature is turned on.
RDADDRCHANGEB(1)	FALSE, TRUE	FALSE	String	Specifies if the port B read address compare feature is turned on.
READ_WIDTH_A	RAMB18E2: 0, 1, 2, 4, 9, 18, 36 (SDP usage) RAMB36E2: 0, 1, 2, 4, 9, 18, 36, 72 (SDP usage)	0	Decimal	Specifies the data width for read port A, including parity bits. This value must be 0 if port A is not used.
READ_WIDTH_B	RAMB18E2: 0, 1, 2, 4, 9, 18 RAMB36E2: 0, 1, 2, 4, 9, 18, 36	0	Decimal	Specifies the data width for read port B including parity bits. This value must be 0 if port B is not used. Not used for SDP memory usage.
RSTREG_PRIORITY_A	RSTREG, REGCE	RSTREG	String	Selects the priority of RESET or CE for the optional output registers. Applies to all port A outputs in both TDP and SDP memory usage.

Table 1-16: **RAMB18E2 and RAMB36E2 Attributes *(Cont'd)***

Attributes	Values	Default	Type	Description
RSTREG_PRIORITY_B	RSTREG, REGCE	RSTREG	String	Selects the priority of RESET or CE for the optional output registers. Applies to all port B outputs in both TDP and SDP memory usage.
SLEEP_ASYNC	FALSE, TRUE	FALSE	String	Determines if the SLEEP pin is synchronous or asynchronous to the clock.
SRVAL_A	RAMB18E2: 18-bit hex value RAMB36E2: 36-bit hex value	RAMB18E2: `18'h00000000` RAMB36E2: `36'h00000000 00000000`	Hex	Specifies the initialization value of the output latches or register when the synchronous reset (RSTREG) is asserted. Applies to all port A outputs in both TDP and SDP memory usage.
SRVAL_B	RAMB18E2: 18-bit hex value RAMB36E2: 36-bit hex value	RAMB18E2: `18'h00000000` RAMB36E2: `36'h00000000 00000000`	Hex	Specifies the initialization value of the output latches or register when the synchronous reset (RSTREG) is asserted. Applies to all port B outputs in both TDP and SDP memory usage.
WRITE_MODE_A(2)	WRITE_FIRST, NO_CHANGE, READ_FIRST	WRITE_FIRST	String	Specifies output behavior of write port A. See Write Modes, page 14.
WRITE_MODE_B(2)	WRITE_FIRST, NO_CHANGE, READ_FIRST	WRITE_FIRST	String	Specifies output behavior of write port B. See Write Modes, page 14.
WRITE_WIDTH_A	RAMB18E2: 0, 1, 2, 4, 9, 18 RAMB36E2: 0, 1, 2, 4, 9, 18, 36	0	Decimal	Specifies the data width for write port A, including parity bits. This value must be 0 if the port is not used. When used as SDP memory, this attribute is not valid.
WRITE_WIDTH_B	RAMB18E2: 0, 1, 2, 4, 9, 18, 36 (SDP usage) RAMB36E2: 0, 1, 2, 4, 9, 18, 36, 72 (SDP usage)	0	Decimal	Specifies the data width for write Port B, including parity bits. This value must be 0 if port B is not used.

Notes:

1. In the UltraScale family (not UltraScale+), this feature is only supported for uninitialized block RAM content and registers (initialized to the default of zero).
2. In SDP mode, the WRITE_MODE_A and WRITE_MODE_B must have the same value.

Data Cascading – CASCADE_ORDER

Specifies the order of the cascaded block RAM. The first block RAM is at the bottom in the cascade chain, the last one is on the top of the cascade, and the middle ones are the block RAM(s) in between bottom and top. This applies to ports A and/or B.

Clocking – CLOCK_DOMAINS

This attribute defines if the clocks to ports A and B are independent/asynchronous or common/synchronous. Clocks driven by the same clock source (CLKA and CLKB are connected together) are common. All other CLKA and CLKB connections are independent.

Enable Address Latching – ENADDREN

This attribute activates or disables the address enable pin (ADDRENA/B). If this attribute is set to TRUE and the corresponding ADDREN pin is Low, the address from the previous clock cycle is used.

Content Initialization – INIT_xx

The memory content can be initialized or cleared in the configuration bitstream. A standard, valid bitstream is required for block RAM initialization or readback due to the power gating feature. For more details on initialization and readback of uninstantiated (power gated) block RAM, see Power Gating of Unused Block RAMs.

IMPORTANT: *The bitstream RSA authentication feature uses certain block RAMs to hold interim rolling keys. For a given block RAM column, each 36K block RAM block in the bottom of a clock region is affected. The first 36K block RAM starting at the bottom of a device is used and then every twelfth 36K block RAM after that in a column is used (BRAM36_X*Y0, BRAM36_X*Y12, BRAM36_X*Y24, etc.) These block RAMs cannot be initialized to user defined values. The block RAMs are always initialized to 0 after configuration.*

INIT_xx attributes define the initial memory contents. By default, block RAM is initialized with all zeros during the device configuration sequence. The 64 initialization attributes from INIT_00 through INIT_3F for the RAMB18E2, and the 128 initialization attributes from INIT_00 through INIT_7F for the RAMB36E2 represent the regular memory contents. Each INIT_xx is a 64-digit hex-encoded bit vector. The memory contents can be partially initialized and are automatically completed with zeros.

The following formula is used to determine the bit positions for each INIT_xx attribute. Given yy = conversion hex-encoded to decimal (xx), INIT_xx corresponds to the memory cells as follows:

- from $[(yy + 1) \times 256] - 1$
- to $(yy) \times 256$

For example, for the attribute INIT_1F, the conversion is:

- yy = conversion hex-encoded to decimal (xx) `"1F"` = 31
- from [(31+1) × 256] - 1 = 8,191
- to 31 × 256 = 7,936

More examples are given in Table 1-17.

Table 1-17: **Block RAM Initialization Attributes**

Attribute	Memory Location	
	From	To
INIT_00	255	0
INIT_01	511	256
INIT_02	767	512
...	...	...
INIT_0E	3839	3584
INIT_0F	4095	3840
INIT_10	4351	4096
...	...	...
INIT_1F	8191	7936
INIT_20	8447	8192
...	...	...
INIT_2F	12287	12032
INIT_30	12543	12288
...	...	...
INIT_3F	16383	16128
...	...	...
INIT_7F	32767	32512

Content Initialization – INITP_xx

INITP_xx attributes define the initial contents of the memory cells corresponding to DINP/DOUTP buses (parity bits). By default, these memory cells are also initialized to all zeros. The initialization attributes represent the memory contents of the parity bits. The eight initialization attributes are INITP_00 through INITP_07 for the RAMB18E2. The 16 initialization attributes are INITP_00 through INITP_0F for the RAMB36E2. Each INITP_xx is a 64-digit hex-encoded bit vector with a regular INIT_xx attribute behavior. The same formula can be used to calculate the bit positions initialized by a particular INITP_xx attribute.

Output Latches Initialization – INIT (INIT_A or INIT_B)

The INIT (single-port) or INIT_A and INIT_B (dual-port) attributes define the output latches or output register values after configuration. The width of the INIT (INIT_A and INIT_B) attribute is the port width, as shown in Table 1-18. These attributes are hex-encoded bit vectors, and the default value is 0. In cascade mode, both the upper and lower block RAM should be initialized to the same value.

Power Saving – RDADDRCHANGE[A|B]

IMPORTANT: *In the UltraScale device family (not UltraScale+), the address compare feature is only supported if both the block RAM content and registers are not initialized to values other than zero (default).*

This attribute is a power-saving feature and enables the read address change (compare) detection circuit. When RDADDRCHANGE is TRUE and the read address and output registers are identical to the previous read cycle, and would therefore result in the same output, no block RAM access is performed to save power. This is most useful if the block RAM is permanently enabled. This feature is only available in the COMMON clock domain case.

Read Width – READ_WIDTH_[A|B]

This attribute determines the A/B read port width of the block RAM. The valid values are: 0 (default), 1, 2, 4, 9, 18, 36, and 72 for the RAMB36E2 when used as SDP memory.

Reset or CE Priority – RSTREG_PRIORITY_[A|B]

This attribute determines the priority of RSTREG or REGCE while asserting RSTREG when DO_REG = 1. Valid values are RSTREG or REGCE. When RSTREG has priority, the RSTREG input resets the optional output register, regardless of the state of REGCE. When REGCE has priority, the RSTREG input resets the optional output register only when REGCE = 1.

Power Saving – SLEEP_ASYNC

This attribute determines if the SLEEP pin is to be used in synchronous or asynchronous mode. Synchronous mode (SLEEP_ASYNC = FALSE) should be used when either both clocks are identical or have a fixed phase relationship. In this mode, ENA and ENB must be deasserted (disabled) in the clock cycle prior to asserting SLEEP. The assertion and deassertion of SLEEP must meet the setup and hold times with respect to both CLKA and CLKB. ENA and ENB must only be asserted again after the block RAM returns from its sleep mode after two clock cycles.

Asynchronous mode (SLEEP_ASYNC = TRUE) should be used when both clocks are truly independent (asynchronous to each other). In this mode, ENA and ENB must be deasserted (disabled) in the clock cycle for the slowest clock prior to asserting SLEEP. SLEEP can then be asserted with the next clock cycle of the same clock. The deassertion of SLEEP causes the block RAM to activate (wake up) up after two clock cycles. Only after the memory wakeup can ENA and ENB be asserted again.

Output Latches/Registers Synchronous Set/Reset (SRVAL_[A|B])

The SRVAL (single-port) or SRVAL_A and SRVAL_B (dual-port) attributes define output latch values when the RSTRAM/RSTREG input is asserted. The width of the SRVAL (SRVAL_A and SRVAL_B) attribute is the port width, as shown in Table 1-18. These attributes are hex-encoded bit vectors and the default value is 0. This attribute sets the value of the output register when the optional output register attribute is set. When the register is not used, the latch gets set to the SRVAL instead. Table 1-18 and Table 1-19 show how the SRVAL and INIT bit locations map to the DOUT outputs for the block RAM primitives and the SDP macro.

Table 1-18: **RAMB18E2 and RAMB36E2, SRVAL and INIT Mapping for Port A and Port B**

Port Width	SRVAL/INIT_(A/B) Full Width	SRVAL/INIT_(A/B) Mapping to DOUT		SRVAL/INIT_(A/B) Mapping to DOUTP	
		DOUTADOUT/ DOUTBDOUT	(SRVAL/INIT)_(A/B)	DOUTP(A/B)/ DOUTP	SRVAL/INIT_(A/B)
1	[0]	[0]	[0]	N/A	N/A
2	[1:0]	[1:0]	[1:0]	N/A	N/A
4	[3:0]	[3:0]	[3:0]	N/A	N/A
9	[8:0]	[7:0]	[7:0]	[0]	[8]
18	[17:0]	[15:0]	[15:0]	[1:0]	[17:16]
36 (only for RAMB36E2)	[35:0]	[31:0]	[31:0]	[3:0]	[35:32]

Table 1-19: **SDP Mapping for RAMB18E2 and RAMB36E2**

Port Width	SRVAL/INIT Full Width	SRVAL/INIT Mapping to DOUT		SRVAL/INIT Mapping to DOUTP	
		DOUT	SRVAL/INIT	DOUTP	SRVAL/INIT
36-bit wide RAMB18E2	[35:0]	[31:0]	[33:18]/[15:0]	[3:0]	[35:34]/[17:16]
72-bit wide RAMB36E2	[71:0]	[63:0]	[67:36]/[31:0]	[7:0]	[71:68]/[35:32]

Optional Output Register On/Off Switch – DOUT[A|B]_REG

This attribute sets the optional pipeline registers at the A/B output of the block RAM improving the clock-to-out timing. If turned on, this adds an extra cycle of read latency. When turned off, the block RAM data is read in the same clock cycle, however with a slower clock-to-out. The valid values are 0 (default) or 1.

Write Width – WRITE_WIDTH_[A|B]

This attribute determines the A/B write port width of the block RAM. The valid values are: 0 (default), 1, 2, 4, 9, 18, 36, and 72 for the RAMB36E2 when used as SDP memory.

Write Mode – WRITE_MODE_[A|B]

This attribute determines the write mode of the A/B input ports. The possible values are WRITE_FIRST (default), READ_FIRST, and NO_CHANGE. Additional information on the write modes is in Write Modes, page 14.

SIM_COLLISION_CHECK

This attribute sets the level of collision checking and behavior in the simulation model. Possible values are ALL (default), GENERATE_X_ONLY, NONE, and WARNING_ONLY.

INIT_FILE

This attribute points to an optional RAM initialization file (initial content). The values are NONE (default) or a STRING (the file name). For the file format, see the *Vivado Design Suite User Guide: Embedded Processor Hardware Design* (UG898).

Block RAM and FIFO Placement

Figure 1-15 shows the allowed upper/lower placement combinations into a single RAMB36 location and the full size RAMB36 allocations.

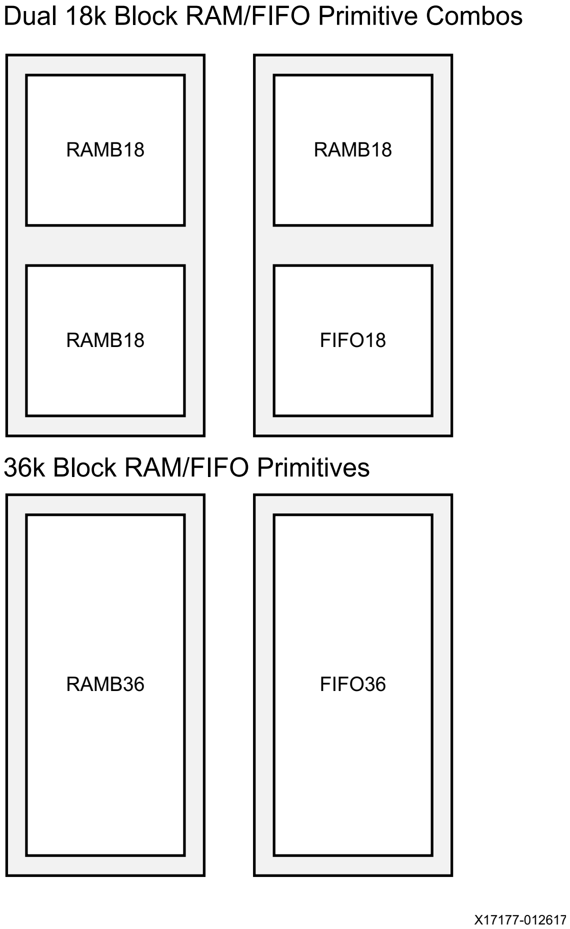


Figure 1-15: **Block RAM and FIFO Placement**

Block RAM Initialization in VHDL or Verilog Code

Block RAM attributes and content can be initialized in VHDL or Verilog code for both synthesis and simulation by using generic maps (VHDL) or defparams (Verilog) within the instantiated component. Modifying the values of the generic map or defparam affects both the simulation behavior and the implemented synthesis results. Inferred block RAM can be initialized as well. The Vivado® Design Suite templates include the code to instantiate the RAMB primitives.

Additional RAMB18E2 and RAMB36E2 Primitive Design Considerations

The RAMB18E2 and RAMB36E2 primitives are integral in the block RAM solution.

Optional Output Registers

Optional output registers can be used at either or both A|B output ports of RAMB18E2 and RAMB36E2. The choice is made using the DO[A|B]_REG attribute. The two independent clock enable pins are REGCE[A|B]. When using the optional output registers at port [A|B], assertion of the synchronous set/reset (RSTREG and RSTRAM) pins of ports [A|B] causes the value specified by the attribute SRVAL to be registered at the output. Figure 1-16 shows an optional output register.

Independent Read and Write Port Width

IMPORTANT: *To specify the port widths using the dual-port mode of the block RAM, designers must use the READ_WIDTH_[A|B] and WRITE_WIDTH_[A|B] attributes.*

These rules should be considered:

- Designing a single-port block RAM requires the port pair widths of one write and one read to be set (e.g., READ_WIDTH_A and WRITE_WIDTH_A).
- Designing a dual-port block RAM requires all port widths to be set.
- In simple dual-port mode, one side of the ports is fixed while the other side can have a variable width. The RAMB18E2 has a data port width of up to 36, while the RAMB36E2 has a data port width of up to 72. When using the block RAM as read-only memory, only the READ_WIDTH_A/B is used.

RAMB18E2 and RAMB36E2 Port Mapping Design Rules

The block RAMs are configurable to various port widths and sizes. Depending on the configuration, some data pins and address pins are not used. Table 1-9, page 33 through Table 1-12, page 34 show the pins used in various configurations. In addition to the information in these tables, these rules are useful to determine the RAMB port connections:

- When using RAMB36E2, if the DIN[A|B] pins are less than 32 bits wide, concatenate (32 - DIN_BIT_WIDTH) logic zeros to the front of DIN[A|B].
- If the DINP[A|B] pins are less than 4 bits wide, concatenate (4 - DINP_BIT_WIDTH) logic zeros to the front of DINP[A|B]. DINP[A|B] can be left unconnected when not in use.

- DOUT[A|B] pins must be 32 bits wide. However, valid data are only found on pins DOUT_BIT_WIDTH – 1 down to 0.
- DOUTP[A|B] pins must be 4 bits wide. However, valid data are only found on pins DOUTP_BIT_WIDTH – 1 down to 0. DOUTP[A|B] can be left unconnected when not in use.
- For the RAMB18E2, ADDR[A/B] is 14 bits wide and for the RAMB32E2, ADDR[A/B] is 15 bits wide. Address width is defined in Table 1-9, page 33.

Byte-Wide Write Enable

Consider these rules when using the byte-wide write enable feature:

- For RAMB36E1
    - In x72 SDP mode, WEBWE[7:0] is used to connect the eight WE inputs for the write port. WEA[3:0] is not used.
    - In x36 mode, WEA[3:0] is used to connect the four WE inputs for port A and WEBWE[3:0] is used to connect the four WE inputs for port B. WEBWE[7:4] is not used.
    - In x18 mode, WEA[1:0] is used to connect the two user WE inputs for port A and WEBWE[1:0] is used to connect the two WE inputs for port B. WEA[3:2] and WEBWE[7:2] are not used.
    - In x9 or smaller port width mode, WEA[0] is used to connect the single user WE input for port A and WEBWE[0] is used to connect the single WE input for port B. WEA[3:1] and WEBWE[7:1] are not used.
- For RAMB18E1
    - In x36 SDP mode, WEBWE[3:0] is used to connect the four WE inputs for the write port. WEA[1:0] is not used.
    - In x18 mode, WEA[1:0] is used to connect the two WE inputs for port A and WEBWE[1:0] is used to connect the two WE inputs for port B. WEBWE[3:2] is not used.
    - In x9 or smaller port width mode, WEA[0] is used to connect the single user WE input for port A and WEBWE[0] is used to connect the single WE input for port B. WEA[1] and WEBWE[3:1] are not used.

Block RAM Applications

Block RAM RSTREG in Register Mode

A block RAM RSTREG in register mode can be used to control the output register as a true pipeline register independent of the block RAM. As shown in Figure 1-16, block RAMs can be read and written independent of register enable or set/reset. In register mode, RSTREG sets DOUT to the SRVAL and data can be read from the block RAM to DBRAM. Data at DBRAM can be clocked out (DOUT) on the next cycle. The timing diagrams in Figure 1-17 through Figure 1-19 show different cases of the RSTREG operation.

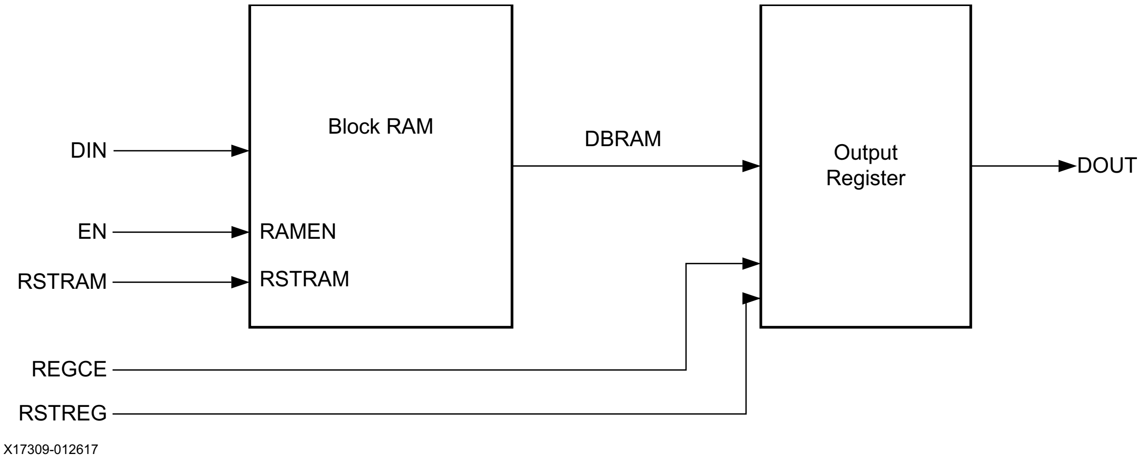


Figure 1-16: **Block RAM RSTREG in Register Mode**

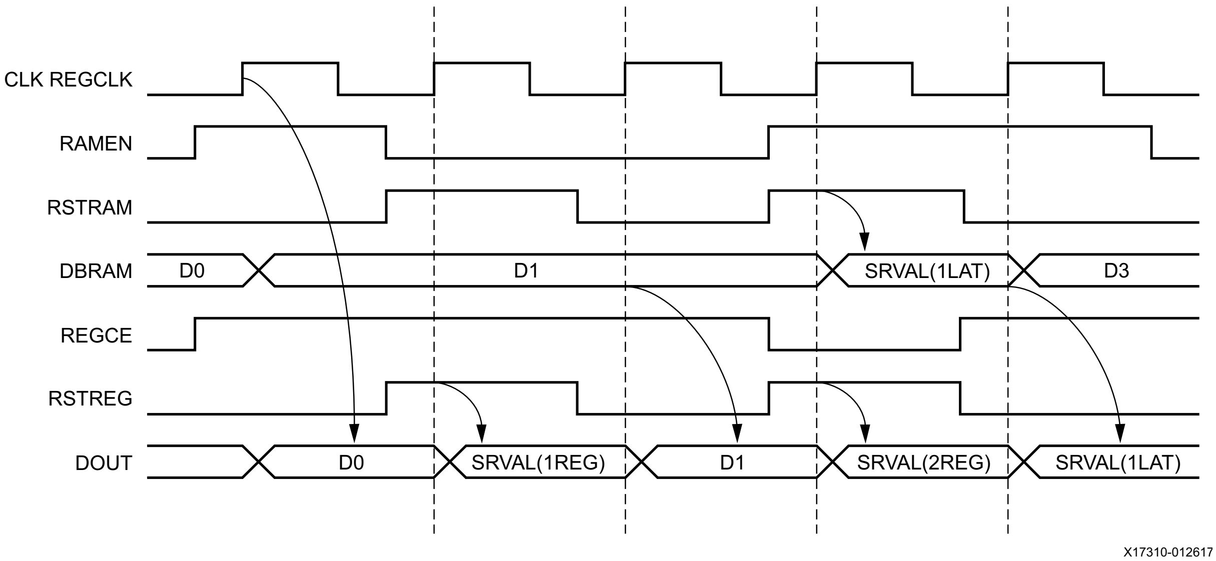


Figure 1-17: **Block RAM Reset Operation in RSTREG Mode**

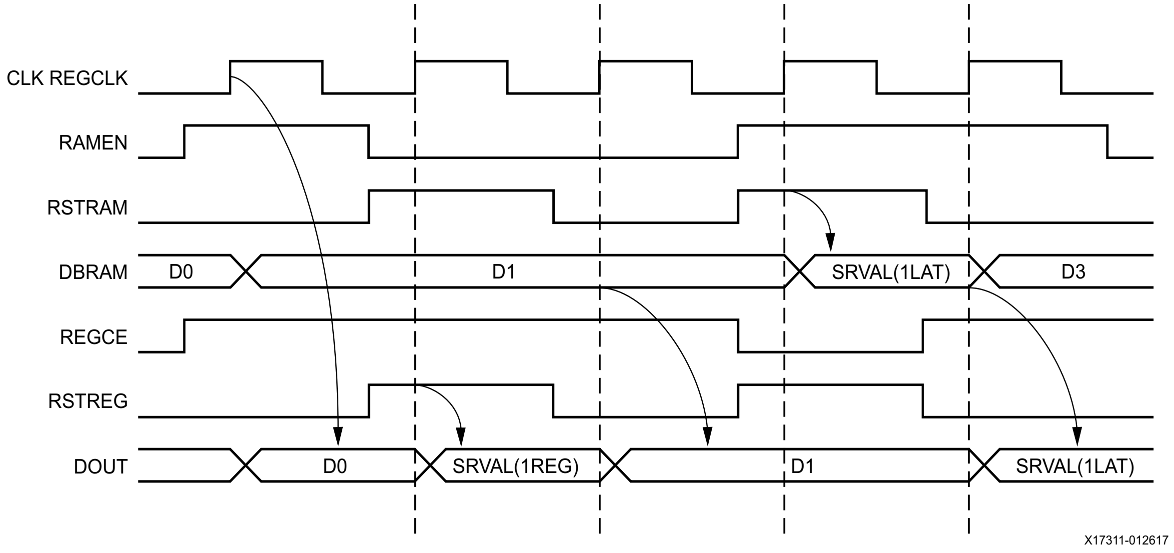


Figure 1-18: **Block RAM Reset Operation in REGCE Mode**

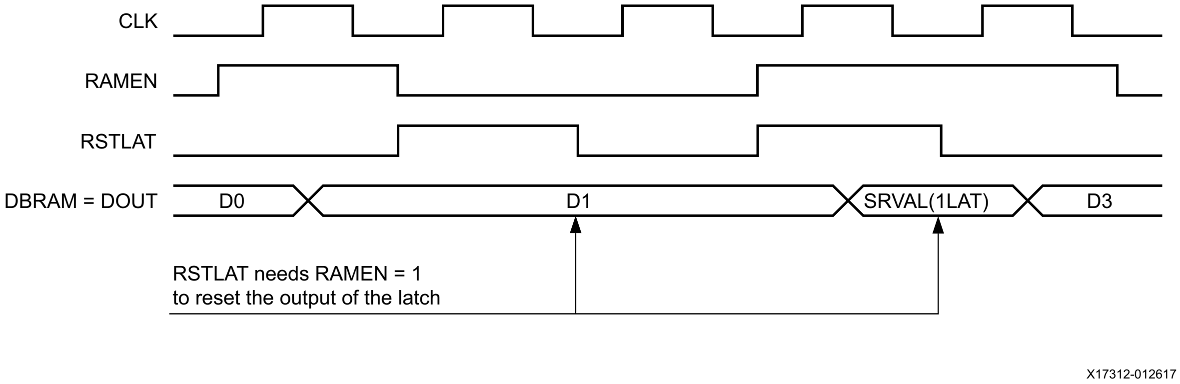


Figure 1-19: **Block RAM Reset Operation in Latch Mode**

Built-in FIFO

Overview

Many designs use block RAMs to implement FIFOs. Common-clock or independent-clock FIFOs can be easily implemented with the dedicated logic in the block RAM. This eliminates the need for additional CLB logic for counter, comparator, or status flag generation, and uses just one block RAM resource per FIFO. Both standard and first-word fall-through (FWFT) modes are supported.

The FIFO can be configured as an 18 Kb or 36 Kb memory. For the 18 Kb mode, the supported configurations are 4K x 4, 2K x 9, 1K x 18, and 512 x 36. The supported configurations for the 36 Kb FIFO are 8K x 4, 4K x 9, 2K x 18, 1K x 36, and 512 x 72. The FIFO ports can now be configured in an asymmetrical fashion.

The block RAM can be configured as a first-in/first-out (FIFO) memory with common or independent read and write clocks. Port A of the block RAM is used as a FIFO read port, and Port B is a FIFO write port. Data is read from the FIFO on the rising edge of the read clock and written to the FIFO on the rising edge of the write clock.

Independent-Clock/Dual-Clock FIFO

The independent-clock FIFO (also referred to as a dual-clock or sometimes asynchronous FIFO) is a first-in/first-out queue where the write interface and the read interface exist in different clock domains. To configure the FIFO as an independent-clock FIFO, the attribute CLOCK_DOMAINS should be set to INDEPENDENT.

The independent-clock FIFO offers a simple write interface and a simple read interface, both of which could be free-running clocks with no frequency or phase relationship between the clocks. As such, it is ideal for situations where:

- WRCLK and RDCLK have different but related frequencies
- WRCLK and RDCLK are out-of-phase with each other
- WRCLK and RDCLK are completely asynchronous (have no relationship)

The independent-clock FIFO can support clock frequencies up to the specified maximum limit. The dual-clock FIFO design avoids ambiguity, glitches, or metastability problems, and provides a convenient way to pass data between differing clock domains.

The write interface is synchronous to the WRCLK domain, writing the data word available on DIN into the FIFO whenever WREN is active one setup time prior to the rising edge of WRCLK.

The read interface is synchronous to the RDCLK domain, triggering a read operation in the FIFO whenever RDEN is active prior to the rising edge of RDCLK, and presenting the next word of data on DOUT following the rising edge of RDCLK in standard mode.

Due to the internal synchronization between the WRCLK domain and the RDCLK domain, certain transitions take an extended number of clock cycles. For example, it takes several clock cycles (both WRCLK and RDCLK clock cycles) for the write operation to synchronize to the RDCLK domain. Only after the write operation has synchronized to the RDCLK domain is that write operation reflected in the status of the RDCLK outputs EMPTY and PROGEMPTY, possibly causing those flags to deassert.

Similarly, the internal synchronization between the RDCLK domain and the WRCLK domain also takes an extended number of clock cycles. For example, it takes several clock cycles (both RDCLK and WRCLK clock cycles) for the read operation to synchronize to the WRCLK domain. Only after the read operation has synchronized to the WRCLK domain is that read operation reflected in the status of the WRCLK outputs FULL and PROGFULL, possibly causing those flags to deassert.

All of the FIFO's inputs and outputs are synchronous to either the WRCLK or RDCLK domain. Due to the uncertainty of two unrelated clock domains, one memory location is reserved in the implementation to prevent errors.

Common-Clock/Single-Clock FIFO

The common-clock FIFO (also referred to as a single-clock or Synchronous FIFO) is a first-in/first-out queue where the write interface and the read interface share a common clock domain. When using synchronous FIFOs, the CLOCK_DOMAINS attribute should be set to COMMON to eliminate clock cycle latency when asserting or deasserting flags.

The interface of the common-clock FIFO is identical to that of the independent-clocks FIFO, except that either:

- There is only one clock input (CLK), or
- There are two clock inputs (WRCLK and RDCLK) that must be tied to the same clock source (clock buffer)

Because a common-clock FIFO requires no synchronization between clock domains, the internal latencies from a write operation to the deassertion of EMPTY or PROGEMPTY, or from a read operation to the deassertion of FULL or PROGFULL are much faster than in an equivalent independent-clock FIFO.

Also, because a common-clock FIFO does not need to deal with the uncertainty of two unrelated clock domains, it can use the entire memory contents for FIFO storage, rather than reserving a memory location to prevent errors. Because of this, the depth of a common-clock FIFO is one word larger than an equivalent independent-clock FIFO. Table 1-20 shows the FIFO capacity in the standard and FWFT modes.

Table 1-20: **Common-Clock FIFO Capacity Without Output Registers and with Symmetric Ports**

Standard Mode		**FWFT Mode**	
18 Kb FIFO	**36 Kb FIFO**	**18 Kb FIFO**	**36 Kb FIFO**
4K entries by 4 bits	8K entries by 4 bits	4K + 1 entries by 4 bits	8K + 1 entries by 4 bits
2K entries by 9 bits	4K entries by 9 bits	2K + 1 entries by 9 bits	4K + 1 entries by 9 bits
1K entries by 18 bits	2K entries by 18 bits	1K + 1 entries by 18 bits	2K + 1 entries by 18 bits
512 entries by 36 bits	1K entries by 36 bits	512 + 1 entries by 36 bits	1K + 1 entries by 36 bits
–	512 entries by 72 bits	–	512 + 1 entries by 72 bits

Notes:

1. There are minor variances in depth based on certain mode settings and output register stages.

IMPORTANT: *Both the block RAM and FIFO require clean, free running clocks. The FIFO cannot be recovered if it is in reset or data content can be corrupted while an unstable clock, that does not meet the minimum pulse width requirement, is applied. If any of the FIFO clocks become unstable, it is recommended to either disable the appropriate clock buffers via the CE pin or only reset the FIFO after a stable clock has returned. The most common use case for this is when a gigabit transceiver loses its recovered clock (e.g., the CDR loses alignment).*

FIFO Architecture: Top-Level View

Figure 1-20 shows a top-level view of the FIFO. The read pointer, write pointer, and status flag logic are dedicated for FIFO use only.

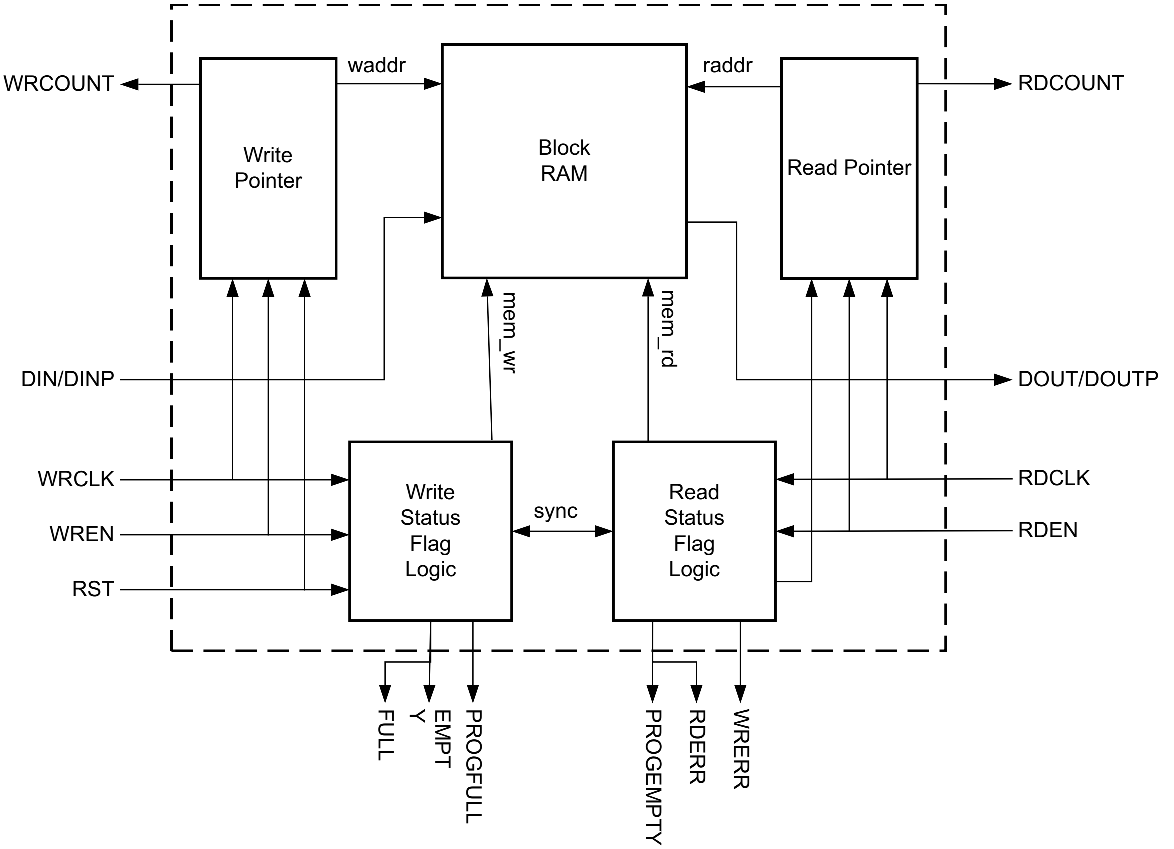


Figure 1-20: **Top-Level View of FIFO in Block RAM**

FIFO Port Width and Depth

The FIFOs support asymmetric read and write ports based on the block RAM's asymmetric port capability to support different port widths for each port. The FIFO18E2 supports independent read/write port width combinations of 4, 8, 16, and 32, which can be expanded to 9, 18, and 36 when utilizing the DINP bits. The FIFO36E2 supports independent read/write port width combinations of 4, 8, 16, 32, and 64, which can be expanded to 9, 18, 36, and 72 when utilizing the DINP bits.

When considering features such as output register stages, FWFT mode, or asymmetric ports, FIFO depth varies. When using asymmetric port widths, the FIFO depth differs depending on the number of write words in the WRCLK domain and the number of read words in the RDCLK domain.

FIFO depth in the WRCLK domain is the number of write words that, when written to a FIFO, causes it to go FULL. In a special case, if the read port is narrower than the write port, it is possible that a partial word exists in the FIFO, causing FULL to assert one clock earlier than expected.

FIFO depth in the RDCLK domain is the number of read words that would be in the FIFO if the FIFO were FULL. Because the read port width might not be the same as the write port width, the depth differs when expressed in the RDCLK domain. Also, in a special case where the write port is narrower than the read port, it is possible that a partial word exists in the FIFO that is not available to be read and therefore does not count toward the FIFO depth in the RDCLK domain.

The FIFO depth can be used to understand and calculate:

- When using WRCOUNT, the depth determines how many more writes can be performed before the FIFO goes FULL (in the WRCLK domain). This calculation is:
    - “FIFO Depth” minus “Number of Words in FIFO” where the number of words available in the FIFO is given by the WRCOUNT (for WRCOUNT_TYPE SIMPLE_DATACOUNT mode for standard FIFOs with no output stages, or EXTENDED_DATACOUNT count mode when using output stages or FWFT).
- How to calculate the PROG_FULL_THRESH to set the threshold at a specific distance from FULL.
- Determine the range of PROG_FULL_THRESH.
- Determine the range of PROG_EMPTY_THRESH.
- Determine all cases of FULL.

Table 1-21 to Table 1-24 list the FIFO depths in both the WRCLK and RDCLK domains for all possible FIFO configurations and widths.

Note: The tables do not cover the EN_ECC_PIPE = TRUE configurations, which increase read port depth by 1.

Table 1-21: **Independent Clocks FIFO Port Width and Depths – FIFO36E2**

		Latch Mode (REGISTER_MODE = "UNREGISTERED")				Register Mode (REGISTER_MODE = "REGISTERED")			
		Standard		FWFT		Standard		FWFT	
Write Port Width	**Read Port Width**	**Write Port Depth**	**Read Port Depth**	**Write Port Depth**	**Read Port Depth**	**Write Port Depth**	**Read Port Depth**	**Write Port Depth**	**Read Port Depth**
4	4	8191	8191	8192	8192	8192	8192	8193	8193
4	8	8191	4095(1)	8193	4096(1)	8193	4096(1)	8195	4097(1)
4	16	8191	2047(1)	8195	2048(1)	8195	2048(1)	8199	2049(1)
4	32	8191	1023(1)	8199	1024(1)	8199	1024(1)	8207	1025(1)
4	64	8191	511(1)	8207	512(1)	8207	512(1)	8223	513(1)
8	4	4095	8190	4095(2)	8191	4095(2)	8191	4096	8192
9	9	4095	4095	4096	4096	4096	4096	4097	4097
9	18	4095	2047(1)	4097	2048(1)	4097	2048(1)	4099	2049(1)
9	36	4095	1023(1)	4099	1024(1)	4099	1024(1)	4103	1025(1)
9	72	4095	511(1)	4103	512(1)	4103	512(1)	4111	513(1)
16	4	2047	8188	2047(2)	8189	2047(2)	8189	2047(2)	8190
18	9	2047	4094	2047(2)	4095	2047(2)	4095	2048	4096
18	18	2047	2047	2048	2048	2048	2048	2049	2049
18	36	2047	1023(1)	2049	1024(1)	2049	1024(1)	2051	1025(1)
18	72	2047	511(1)	2051	512(1)	2051	512(1)	2055	513(1)
32	4	1023	8184	1023(2)	8185	1023(2)	8185	1023(2)	8186
36	9	1023	4092	1023(2)	4093	1023(2)	4093	1023(2)	4094
36	18	1023	2046	1023(2)	2047	1023(2)	2047	1024	2048
36	36	1023	1023	1024	1024	1024	1024	1025	1025
36	72	1023	511(1)	1025	512(1)	1025	512(1)	1027	513(1)
64	4	511	8176	511(2)	8177	511(2)	8177	511(2)	8178
72	9	511	4088	511(2)	4089	511(2)	4089	511(2)	4090
72	18	511	2044	511(2)	2045	511(2)	2045	511(2)	2046
72	36	511	1022	511(2)	1023	511(2)	1023	512	1024
72	72	511	511	512	512	512	512	513	513

Notes:

1. When the read port depth has a fractional part, the read depth value is rounded down. The FIFO does not allow partial words to be read from the FIFO, so a partial word can exist in the FIFO even when the FIFO is EMPTY. Therefore, one or more writes might be needed before the final data in the FIFO can be read. When the FIFO is FULL, a single read deasserts FULL, after which two or more write operations are required to get it back to FULL again.
2. When the write port depth has a fractional part, the write depth value is rounded down. Because it takes multiple reads to read a complete write word, there might be partial write words in the FIFO at some points in time. If FULL, one or more additional read operations might free up enough space to make room for an additional write operation.

Table 1-22: **Common-Clock FIFO Port Width and Depths – FIFO36E2**

		Latch Mode (REGISTER_MODE = "UNREGISTERED")				Register Mode (REGISTER_MODE = "REGISTERED")			
		Standard		FWFT		Standard		FWFT	
Write Port Width	**Read Port Width**	**Write Port Depth**	**Read Port Depth**	**Write Port Depth**	**Read Port Depth**	**Write Port Depth**	**Read Port Depth**	**Write Port Depth**	**Read Port Depth**
4	4	8192	8192	8193	8193	8193	8193	8194	8194
4	8	8192	4096	8194	4097	8194	4097	8196	4098
4	16	8192	2048	8196	2049	8196	2049	8200	2050
4	32	8192	1024	8200	1025	8200	1025	8208	1026
4	64	8192	512	8208	513	8208	513	8224	514
8	4	4096	8192	4096(1)	8193	4096(1)	8193	4097	8194
9	9	4096	4096	4097	4097	4097	4097	4098	4098
9	18	4096	2048	4098	2049	4098	2049	4100	2050
9	36	4096	1024	4100	1025	4100	1025	4104	1026
9	72	4096	512	4104	513	4104	513	4112	514
16	4	2048	8192	2048(1)	8193	2048(1)	8193	2048(1)	8194
18	9	2048	4096	2048(1)	4097	2048(1)	4097	2049	4098
18	18	2048	2048	2049	2049	2049	2049	2050	2050
18	36	2048	1024	2050	1025	2050	1025	2052	1026
18	72	2048	512	2052	513	2052	513	2056	514
32	4	1024	8192	1024(1)	8193	1024(1)	8193	1024(1)	8194
36	9	1024	4096	1024(1)	4097	1024(1)	4097	1024(1)	4098
36	18	1024	2048	1024(1)	2049	1024(1)	2049	1025	2050
36	36	1024	1024	1025	1025	1025	1025	1026	1026
36	72	1024	512	1026	513	1026	513	1028	514
64	4	512	8192	512(1)	8193	512(1)	8193	512(1)	8194
72	9	512	4096	512(1)	4097	512(1)	4097	512(1)	4098
72	18	512	2048	512(1)	2049	512(1)	2049	512(1)	2050
72	36	512	1024	512(1)	1025	512(1)	1025	513	1026
72	72	512	512	513	513	513	513	514	514

Notes:

1. When the write port depth has a fractional part, the write depth value is rounded down. Because it takes multiple reads to read a complete write word, there might be partial write words in the FIFO at some points in time. If FULL, one or more additional read operations might free up enough space to make room for an additional write operation.

Table 1-23: **Independent-Clock FIFO Port Width and Depths – FIFO18E2**

		Latch Mode (REGISTER_MODE = "UNREGISTERED")				Register Mode (REGISTER_MODE = "REGISTERED")			
		Standard		FWFT		Standard		FWFT	
Write Port Width	**Read Port Width**	**Write Port Depth**	**Read Port Depth**	**Write Port Depth**	**Read Port Depth**	**Write Port Depth**	**Read Port Depth**	**Write Port Depth**	**Read Port Depth**
4	4	4095	4095	4096	4096	4096	4096	4097	4097
4	8	4095	2047 (1)	4097	2048 (1)	4097	2048 (1)	4099	2049 (1)
4	16	4095	1023 (1)	4099	1024 (1)	4099	1024 (1)	4103	1025 (1)
4	32	4095	511 (1)	4103	512 (1)	4103	512 (1)	4111	513 (1)
8	4	2047	4094	2047 (2)	4095	2047 (2)	4095	2048	4096
9	9	2047	2047	2048	2048	2048	2048	2049	2049
9	18	2047	1023 (1)	2049	1024 (1)	2049	1024 (1)	2051	1025 (1)
9	36	2047	511 (1)	2051	512 (1)	2051	512 (1)	2055	513 (1)
16	4	1023	4092	1023 (2)	4093	1023 (2)	4093	1023 (2)	4094
18	9	1023	2046	1023 (2)	2047	1023 (2)	2047	1024	2048
18	18	1023	1023	1024	1024	1024	1024	1025	1025
18	36	1023	511 (1)	1025	512 (1)	1025	512 (1)	1027	513 (1)
32	4	511	4088	511 (2)	4089	511 (2)	4089	511 (2)	4090
36	9	511	2044	511 (2)	2045	511 (2)	2045	511 (2)	2046
36	18	511	1022	511 (2)	1023	511 (2)	1023	512	1024
36	36	511	511	512	512	512	512	513	513

Notes:

1. When the read port depth has a fractional part, the read depth value is rounded down. The FIFO does not allow partial words to be read from the FIFO, so a partial word can exist in the FIFO even when the FIFO is EMPTY. Therefore, one or more writes might be needed before the final data in the FIFO can be read. When the FIFO is FULL, a single read deasserts FULL, after which two or more write operations are required to get it back to FULL again.
2. When the write port depth has a fractional part, the write depth value is rounded down. Because it takes multiple reads to read a complete write word, there might be partial write words in the FIFO at some points in time. If FULL, one or more additional read operations might free up enough space to make room for an additional write operation.

Table 1-24: **Common-Clock FIFO Port Width and Depths – FIFO18E2**

		Latch Mode (REGISTER_MODE = "UNREGISTERED")				Register Mode (REGISTER_MODE = "REGISTERED")			
		Standard		FWFT		Standard		FWFT	
Write Port Width	Read Port Width	Write Port Depth	Read Port Depth	Write Port Depth	Read Port Depth	Write Port Depth	Read Port Depth	Write Port Depth	Read Port Depth
4	4	4096	4096	4097	4097	4097	4097	4098	4098
4	8	4096	2048	4098	2049	4098	2049	4100	2050
4	16	4096	1024	4100	1025	4100	1025	4104	1026
4	32	4096	512	4104	513	4104	513	4112	514
8	4	2048	4096	2048 (1)	4097	2048 (1)	4097	2049	4098
9	9	2048	2048	2049	2049	2049	2049	2050	2050
9	18	2048	1024	2050	1025	2050	1025	2052	1026
9	36	2048	512	2052	513	2052	513	2056	514
16	4	1024	4096	1024 (1)	4097	1024 (1)	4097	1024 (1)	4098
18	9	1024	2048	1024 (1)	2049	1024 (1)	2049	1025	2050
18	18	1024	1024	1025	1025	1025	1025	1026	1026
18	36	1024	512	1026	513	1026	513	1028	514
32	4	512	4096	512 (1)	4097	512 (1)	4097	512 (1)	4098
36	9	512	2048	512 (1)	2049	512 (1)	2049	512 (1)	2050
36	18	512	1024	512 (1)	1025	512 (1)	1025	513	1026
36	36	512	512	513	513	513	513	514	514

Notes:

1. When the write port depth has a fractional part, the write depth value is rounded down. Because it takes multiple reads to read a complete write word, there might be partial write words in the FIFO at some points in time. If FULL, one or more additional read operations might free up enough space to make room for an additional write operation.

FIFO Primitives

Figure 1-21 shows the FIFO36E2 used as FIFO36.

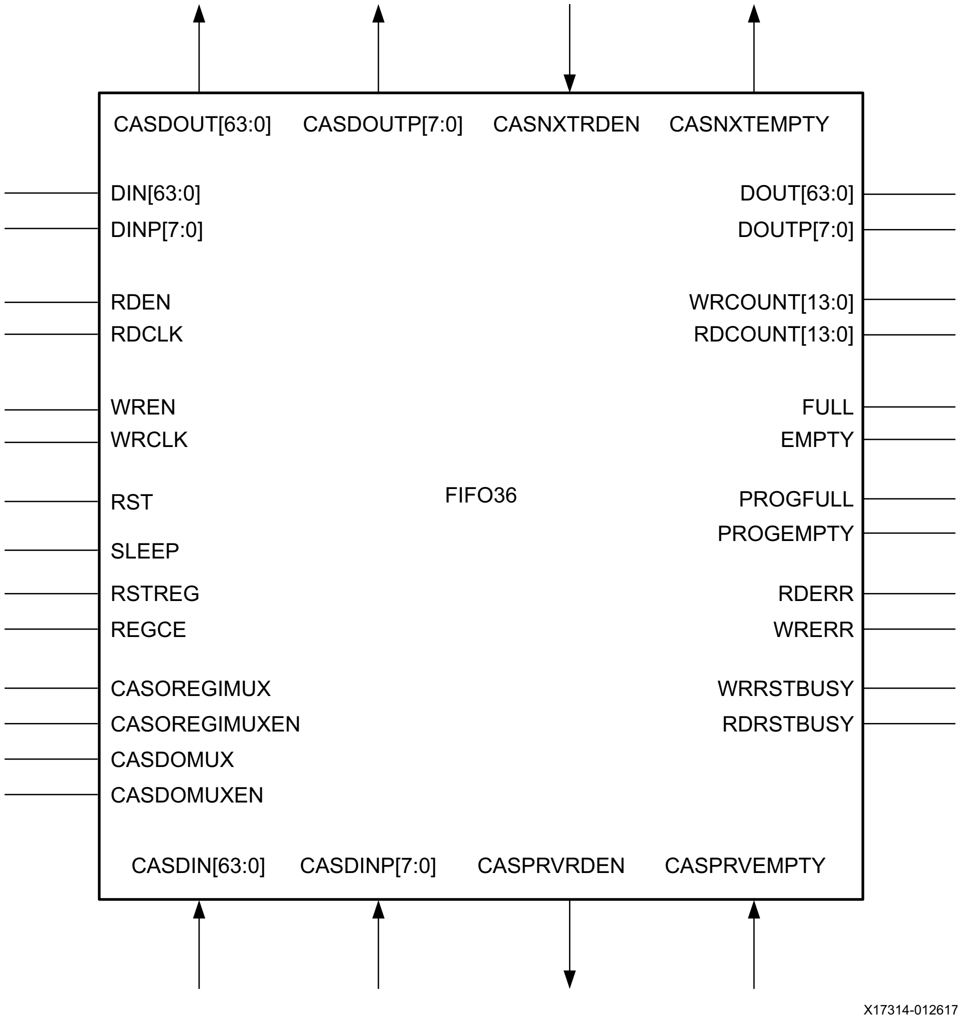


Figure 1-21: **FIFO36**

The ports for using the FIFO36E2 in ECC mode are described in Built-in Error Correction.

Figure 1-22 shows the FIFO18E2 used as FIFO18.

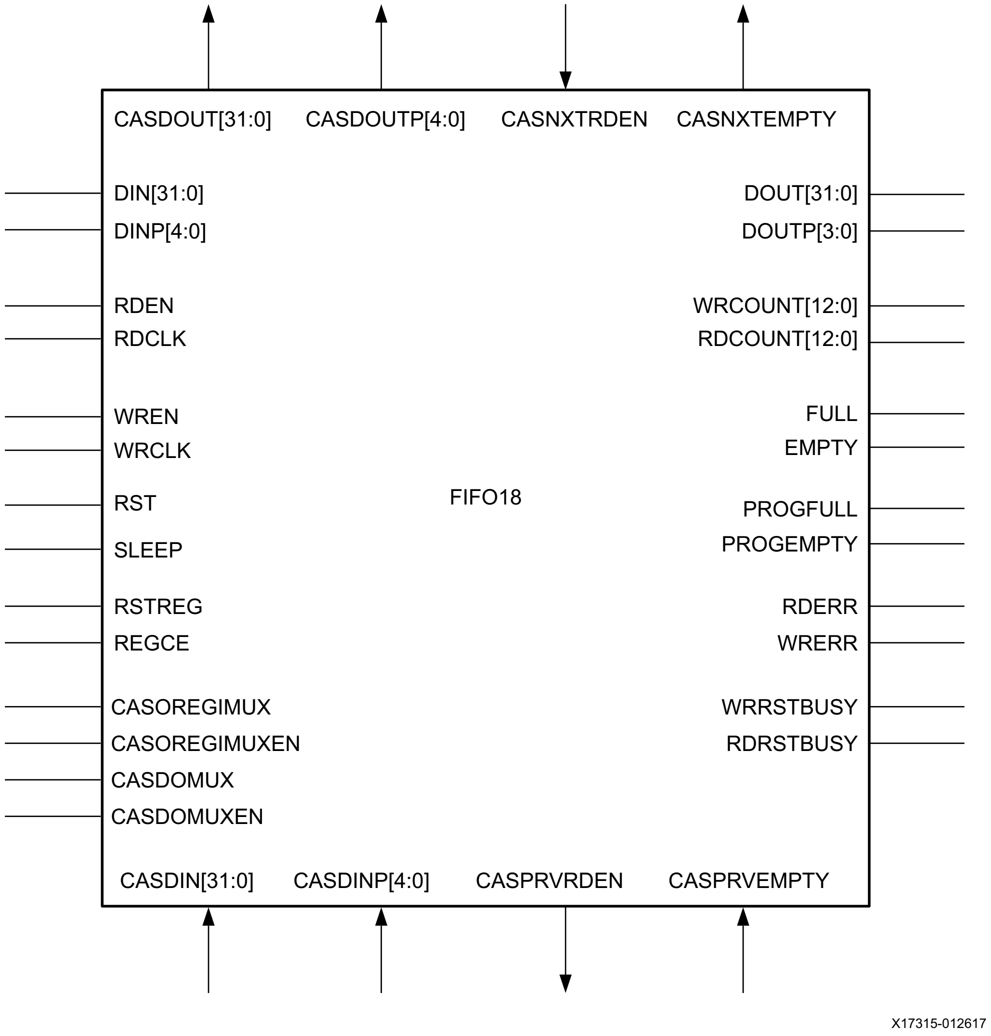


Figure 1-22: **FIFO18**

FIFO Port Descriptions and Attributes

Table 1-22 lists the FIFO I/O port names and descriptions.

Table 1-25: **FIFO18E2 and FIFO36E2 Port Names and Descriptions**

Port	Direction	Description	Configurations
PROGEMPTY	Output	Programmable flag to indicate the FIFO is almost empty (contains less than or equal to the number of words specified by PROG_EMPTY_THRESH). Synchronous to RDCLK.	All configurations. Controlled by PROG_EMPTY_THRESH.
PROGFULL	Output	Programmable flag to indicate the FIFO is almost full (contains greater than or equal to the number of words specified by PROG_FULL_THRESH). Synchronous to WRCLK.	All configurations. Controlled by PROG_FULL_THRESH.
FIFO18: DIN<31:0> FIFO36: DIN<63:0>	Input	FIFO data input bus. Synchronous to WRCLK.	All configurations, width controlled by WRITE_WIDTH.
FIFO18: DINP<3:0> FIFO36: DINP<7:0>	Input	FIFO parity input bus. Synchronous to WRCLK.	All configurations, width controlled by WRITE_WIDTH.
FIFO18: DOUT<31:0> FIFO36: DOUT<63:0>	Output	FIFO data output bus. Synchronous to RDCLK.	All configurations, width controlled by READ_WIDTH.
FIFO18: DOUTP<3:0> FIFO36: DOUTP<7:0>	Output	FIFO parity output bus. Synchronous to RDCLK.	All configurations, width controlled by READ_WIDTH.
EMPTY	Output	Active-High flag to indicate when the FIFO is empty. Synchronous to RDCLK.	All configurations.
FULL	Output	Active-High flag to indicate when the FIFO is full. Synchronous to WRCLK.	All configurations.
RDCLK	Input	Read clock.	All configurations.
FIFO18: RDCOUNT<12:0> FIFO36: RDCOUNT<13:0>	Output	Output of either the internal FIFO read pointer, or a count of the number of words in the FIFO. Synchronous to RDCLK.	All configurations, output value controlled by RDCOUNT_TYPE.
RDEN	Input	Active-High read enable relative to RDCLK.	All configurations.
RDERR	Output	Indicates that a read operation failed due to the FIFO being EMPTY, or FIFO in a reset condition. Synchronous to RDCLK.	All configurations.

Table 1-25: **FIFO18E2 and FIFO36E2 Port Names and Descriptions *(Cont'd)***

Port	Direction	Description	Configurations
RDRSTBUSY	Output	Active-High indicator that the FIFO is currently in a reset state. Synchronous to RDCLK.	Indicates that the RDCLK domain FIFO logic is currently in a reset state. Any attempt to perform a read operation while RDRSTBUSY = 1 causes a RDERR.
REGCE	Input	Active-High enable for output register stage relative to RDCLK.	Only when REGISTER_MODE = DO_PIPELINED.
RST	Input	Active-High synchronous reset. Synchronous to WRCLK.	RST input. Must be synchronous to the WRCLK domain. RST can be set to 0 when not used.
RSTREG	Input	Active-High enable for output register reset relative to RDCLK.	Only when REGISTER_MODE = DO_PIPELINED.
SLEEP(2)	Input	Dynamic shutdown power saving. If SLEEP is High, the block RAM memory array is in power-saving mode.	All configurations.
WRCLK	Input	Write clock.	All configurations.
FIFO18: WRCOUNT<12:0> FIFO36: WRCOUNT<13:0>	Output	Output of either the internal FIFO write pointer, or a count of the number of words in the FIFO. Synchronous to WRCLK.	All configurations, output value controlled by WRCOUNT_TYPE.
WREN	Input	Active-High write enable relative to WRCLK.	All configurations.
WRERR	Output	Indicates that a write operation failed due to the FIFO being FULL, or FIFO in a reset condition. Synchronous to WRCLK.	All configurations.
WRRSTBUSY	Output	Active-High indicator that the FIFO is currently in a reset state. Synchronous to WRCLK.	Indicates that the WRCLK domain FIFO logic is currently in a reset state. Any attempt to perform a write operation while WRRSTBUSY = 1 causes a WRERR.
FIFO18: CASDIN<31:0> FIFO36: CASDIN<63:0>	Input	FIFO data input bus from previous FIFO when cascading FIFOs serially or in parallel to extend depth.	Only used when CASCADE_ORDER = MIDDLE, LAST, or PARALLEL.
FIFO18: CASDINP<3:0> FIFO36: CASDINP<7:0>	Input	FIFO parity data input bus from previous FIFO when cascading FIFOs serially or in parallel to extend depth.	Only used when CASCADE_ORDER = MIDDLE, LAST, or PARALLEL.
FIFO18: CASDOUT<31:0> FIFO36: CASDOUT<63:0>	Output	FIFO data output bus to next FIFO when cascading FIFOs serially or in parallel to extend depth.	Only used when CASCADE_ORDER = FIRST, MIDDLE, or PARALLEL.

Table 1-25: **FIFO18E2 and FIFO36E2 Port Names and Descriptions *(Cont'd)***

Port	Direction	Description	Configurations
FIFO18: CASDOUTP<3:0> FIFO36: CASDOUTP<7:0>	Output	FIFO parity data output bus to next FIFO when cascading FIFOs serially or in parallel to extend depth.	Only used when CASCADE_ORDER = FIRST, MIDDLE, or PARALLEL.
CASPRVEMPTY	Input	Cascaded EMPTY input from previous FIFO, used for cascading FIFOs serially to extend depth. Connects to the CASNXTEMPTY of the previous FIFO.	Only used when CASCADE_ORDER = MIDDLE or LAST and cascading FIFOs serially.
CASPRVRDEN	Output	Control output driving the cascaded RDEN input of the previous FIFO, used for cascading FIFOs serially to extend depth. Connects to CASNXTRDEN of the previous FIFO.	Only used when CASCADE_ORDER = MIDDLE or LAST and cascading FIFOs serially.
CASNXTRDEN	Input	Cascaded RDEN input from next FIFO, used for cascading FIFOs serially to extend depth. Connects to CASPRVRDEN of the next FIFO.	Only used when CASCADE_ORDER = FIRST or MIDDLE and cascading FIFOs serially.
CASNXTEMPTY	Output	Cascaded EMPTY output to next FIFO, used for cascading FIFOs serially to extend depth. Connects to CASPRVEMPTY of the next FIFO.	Only used when CASCADE_ORDER = FIRST or MIDDLE and cascading FIFOs serially.
CASOREGIMUX	Input	D input to flip-flop that drives the select line to the cascade multiplexer before the output registers.	Only used when REGISTER_MODE = DO_PIPELINED and CASCADE_ORDER = PARALLEL.
CASOREGIMUXEN	Input	EN input to flip-flop that drives the select line to the cascade multiplexer before the output registers.	Only used when REGISTER_MODE = DO_PIPELINED and CASCADE_ORDER = PARALLEL.
CASDOMUX	Input	D input to flip-flop that drives the select line to the cascade multiplexer on the block RAM outputs.	Only used when CASCADE_ORDER = PARALLEL.
CASDOMUXEN	Input	EN input to the flip-flop that drives the select line to the cascade multiplexer on the block RAM outputs.	Only used when CASCADE_ORDER = PARALLEL.

Notes:

1. The ports for using the FIFO36E2 in ECC mode are described in Built-in Error Correction.
2. See block RAM SLEEP pin and attribute descriptions for more information. For the FIFO sleep mode, the assertion and deassertion of RDEN/WREN requirements deviate from the block RAM rules depending on the clock mode (independent/common), read/write clock frequency ratios, and other FIFO configurations, such as FWFT and output/pipeline registers. It is recommended to simulate the design to determine the exact behavior for specific configurations

Table 1-26: **FIFO18E2 and FIFO36E2 Attributes and Descriptions**

Attribute	Values	Default	Description
FIFO18: PROG_EMPTY_THRESH<11:0> FIFO36: PROG_EMPTY_THRESH<12:0>	Decimal User Selectable		Specifies the minimum number of read words in the FIFO at or below which PROGEMPTY is asserted.
FIFO18: PROG_FULL_THRESH<11:0> FIFO36: PROG_FULL_THRESH<12:0>	Decimal User Selectable		Specifies the maximum number of write words in the FIFO at or above which PROGFULL is asserted.
WRITE_WIDTH	Integer 4, 9, 18, 36, 72(FIFO36)		Indicates the total port width of the DIN and DINP ports.
READ_WIDTH	Integer 4, 9, 18, 36, 72(FIFO36)		Indicates the total port width of the DOUT and DOUTP ports.
REGISTER_MODE	UNREGISTERED, REGISTERED, DO_PIPELINED	UNREGISTERED	UNREGISTERED: No output register stage. REGISTERED: Output register is controlled automatically by the FIFO controller to behave like an additional FIFO word. DO_PIPELINED: Output register is controlled by external REGCE and RSTREG inputs.
CLOCK_DOMAINS	COMMON, INDEPENDENT	INDEPENDENT	COMMON: Common clock/single clock/synchronous FIFO. INDEPENDENT: Independent clock/dual clock/asynchronous FIFO.
FIRST_WORD_FALL_THROUGH	String: TRUE/FALSE	FALSE	TRUE: Use FWFT FIFO output behavior. FALSE: Use standard FIFO output behavior.
INIT(1)	FIFO18: 36-bit hex FIFO36: 72 bit hex	`36'h000000000` `72'h0000000000000000000`	Specifies the initial value on DOUT after configuration. This initial value always applies to the block RAM/FIFO's output latches, and also specifies the initial value of the output registers.

Table 1-26: **FIFO18E2 and FIFO36E2 Attributes and Descriptions *(Cont'd)***

Attribute	Values	Default	Description
SRVAL[1]	FIFO18: 36 bit hex FIFO36: 72 bit hex	`36'h000000000` `72'h0000000000000000000`	Specifies the reset value of the FIFO output bus. This is the reset value of both the RAM output latches and the output registers, regardless of the state of REGISTER_MODE. When REGISTER_MODE = DO_PIPELINED, the RSTREG input resets the DOUT output of the FIFO's output register stage to this value. Otherwise, DOUT is reset by the FIFO's RST input.
WRCOUNT_TYPE	RAW_PNTR, SYNC_PNTR, SIMPLE_DATACOUNT, EXTENDED_DATACOUNT	RAW_PNTR	Defines the behavior of the WRCOUNT output. WRCOUNT_TYPE can be configured to provide internal FIFO counter information or a count of the FIFO contents.
RDCOUNT_TYPE	RAW_PNTR, SYNC_PNTR, SIMPLE_DATACOUNT, EXTENDED_DATACOUNT	RAW_PNTR	Defines the behavior of the RDCOUNT output. RDCOUNT_TYPE can be configured to provide internal FIFO counter information or a count of the FIFO contents.
RSTREG_PRIORITY	RSTREG, REGCE	RSTREG	Only used when REGISTER_MODE = DO_PIPELINED. If set to RSTREG, the RSTREG input resets the output register to SRVAL regardless of the state of the REGCE input. If set to REGCE, the RSTREG input resets the output register to SRVAL only if the REGCE input is 1.
CASCADE_ORDER	NONE, FIRST, MIDDLE, LAST, PARALLEL	NONE	Defines the FIFO cascade mode (serial or parallel) and cascade order when cascading FIFOs in series.

Notes:

1. The attributes for using the FIFO36E2 in ECC mode are described in Built-in Error Correction.

FIFO18/36 Unused Inputs

The unused inputs are shown in Table 1-27.

Table 1-27: **FIFO18/36 Unused Inputs**

FIFO18/36	Constant
RDCLK	0
WRCLK	0
RDEN	0
WREN	0
REGCLK	0
RSTREG	0
REGCE	1
RST	0
SLEEP	0
CASDOMUX	0
CASOREGIMUX	0
CASDOMUXEN	1
CASOREGIMUXEN	1
INJECTSBITERR	0
INJECTDBITERR	0

FIFO Attribute Descriptions

PROG_EMPTY_THRESH

PROG_EMPTY_THRESH is a user-defined threshold that defines when a specified number of words have been read from the FIFO. When the number of words is less than or equal to PROG_EMPTY_THRESH, the PROGEMPTY signal is asserted. If PROGEMPTY = 0, the number of words in the FIFO is greater than PROG_EMPTY_THRESH. The PROGEMPTY flag conservatively represents the number of words in the FIFO relative to the PROG_EMPTY_THRESH setting. Therefore, PROGEMPTY considers all read operations but might not immediately update due to write operations synchronization latency. Thus, it can sometimes present the FIFO as more empty than it actually is.

PROG_FULL_THRESH

PROG_FULL_THRESH is a user-defined threshold that defines when a specified number of words have been written to the FIFO, and can be used to determine the amount of available space remaining in the FIFO. When the number of words is more than or equal to PROG_FULL_THRESH, the PROGFULL signal is asserted. If PROGFULL = 0, the number of words in the FIFO is less than PROG_FULL_THRESH. Similar to the PROGEMPTY assertion, the PROGFULL flag conservatively represents the number of words in the FIFO relative to the PROG_FULL_THRESH setting. Therefore, PROGFULL considers all write operations but might not immediately update due to read operations synchronization latency. Thus, it can sometimes present the FIFO as being more full than it actually is.

WRITE_WIDTH

This attribute controls the total data width of the DIN and DINP ports together. Valid values of WRITE_WIDTH are 4, 9, 18, 36, and 72 (FIFO36E2).

READ_WIDTH

This attribute controls the total data width of the DOUT and DOUTP ports together. Valid values of READ_WIDTH are 4, 9, 18, 36, and 72 (FIFO36E2).

REGISTER_MODE

UNREGISTERED indicates that the FIFO does not use the block RAM output register. REGISTERED indicates that the output register is used and that the FIFO controls it such that the FIFO's DOUT behaves like an additional FIFO word. This setting does not add any latency to the DOUT but has an impact on clock-to-out and WRCLK to EMPTY deassertion latency.

DO_PIPELINED indicates that the output register adds an additional pipeline stage to the DOUT path, with REGCE and RSTREG inputs so that the output register can be controlled. This setting has an impact on clock-to-out and WRCLK to EMPTY deassertion latency.

CLOCK_DOMAINS

COMMON indicates that the FIFO is a common-clock FIFO, and there is only one clock input (CLK) or two clock inputs (WRCLK and RDCLK) that are tied to the same clock source (single clock buffer).

INDEPENDENT indicates that the FIFO has two independent and perhaps asynchronous clocks (WRCLK and RDCLK) coming from two different clock buffers.

FIRST_WORD_FALL_THROUGH

This attribute defines the read (output) behavior of the FIFO when EMPTY. If TRUE (FWFT), the DIN data is placed on the DOUT bus before RDEN is asserted. When writing the first word to an EMPTY FIFO, the first word "falls through" to the output, and appears on the DOUT bus at the same time EMPTY is deasserted. If FALSE (standard read interface), RDEN presents the DIN data (data in the FIFO) on DOUT after the next rising edge of RDCLK. When the EMPTY flag asserts, no more data is available to be read, and any additional reads cause RDERR to assert after the next RDCLK edge. See also Operating Mode, page 72.

INIT and SRVAL

INIT defines the values of the output latches (DOUT and DOUTP) or output register values after configuration. SRVAL defines values of the output latches or output register when the RST/RSTREG input is asserted. See Table 1-28.

Table 1-28: **FIFO18E2 and FIFO36E2, SRVAL and INIT Mapping**

Port Width	SRVAL/INIT Full Width	SRVAL/INIT Mapping to DOUT		SRVAL/INIT Mapping to DOUTP	
		DOUT	SRVAL/INIT	DOUTP	SRVAL/INIT
1	[0]	[0]	[0]	N/A	N/A
2	[1:0]	[1:0]	[1:0]	N/A	N/A
4	[3:0]	[3:0]	[3:0]	N/A	N/A
9	[8:0]	[7:0]	[7:0]	[0]	[8]
18	[17:0]	[15:0]	[15:0]	[1:0]	[17:16]
36	[35:0]	[31:0]	[31:0]	[3:0]	[35:32]
72 (only for FIFO36E2)	[71:0]	[63:0]	[63:0]	[7:0]	[71:64]

WRCOUNT_TYPE

WRCOUNT_TYPE defines the way status information about the internal state of the FIFO counters or the number of words in the FIFO is provided to the WRCOUNT output:

- RAW_PNTR = FIFO memory write pointer, synchronous to WRCLK domain.
- SYNC_PNTR = FIFO memory write pointer, synchronized to RDCLK domain.
- SIMPLE_DATACOUNT = Subtraction of read pointer (synchronized to the WRCLK domain) from write pointer (in the WRCLK domain) to indicate the number of words in the memory in the WRCLK domain. Does not account for additional words stored in output register stages.
- EXTENDED_DATACOUNT = Subtraction of read pointer (synchronized to the WRCLK domain) from write pointer (in the WRCLK domain) plus 0, 1, or 2 (depending on the output stages) to indicate the number of words in the memory and in the output stages in the WRCLK domain.

RDCOUNT_TYPE

RDCOUNT_TYPE defines the way status information about the internal state of the FIFO counters is provided to the RDCOUNT output.

- RAW_PNTR – RDCOUNT = FIFO memory read pointer, synchronous to RDCLK domain.
- SYNC_PNTR – RDCOUNT = FIFO memory read pointer, synchronized to WRCLK domain (delayed). Not supported for common-clock FIFO.
- SIMPLE_DATACOUNT – RDCOUNT = Subtraction of read pointer (in the RDCLK domain) from write pointer (synchronized to RDCLK domain) to indicate the number of words in the memory in the RDCLK domain. Does not account for additional words stored in output register stages.
- EXTENDED_DATACOUNT – RDCOUNT = Subtraction of read pointer (in the RDCLK domain) from write pointer (synchronized to the RDCLK domain) plus 0, 1, or 2 (depending on output stages) to indicate the number of words in the memory and in the output stages in the RDCLK domain.

RSTREG_PRIORITY

This attribute determines the RSTREG priority over REGCE when using the optional output registers (DO_REG = 1). If set to RSTREG, the RSTREG input resets the output register to SRVAL regardless of the state of the REGCE input. If set to REGCE, the RSTREG input resets the output register to SRVAL only if the REGCE input is 1.

CASCADE_ORDER

This attribute defines the FIFO order when cascading FIFOs serially or sets the cascading to parallel. When cascading FIFOs in series, the first FIFO (the FIFO with the write interface) must have CASCADE_ORDER = "FIRST", the last FIFO (the FIFO with the read interface) must have CASCADE_ORDER = "LAST", and all other FIFOs in the chain must have CASCADE_ORDER = "MIDDLE". When expanding the FIFO in parallel mode, this attribute should be set to PARALLEL (see also Cascading FIFOs Serially and in Parallel to Extend Depth, page 81).

- NONE: Normal FIFO operation.
- FIRST: FIFO is the first in a series of FIFOs cascaded to extend depth. It has a normal write interface, but the read interface is modified to link with the next FIFO in the chain.
- MIDDLE: FIFO is part of a chain of FIFOs cascaded to extend depth. It connects to the previous FIFO configured as FIRST or MIDDLE and connects to the next FIFO configured as MIDDLE or LAST.
- LAST: FIFO is the last in a series of FIFOs cascaded to extend depth. It has a normal read interface, but the write interface is modified to link with the previous FIFO in the chain.
- PARALLEL: FIFO is used in a design where the output data is cascaded in parallel.

FIFO Operations

Reset

IMPORTANT: *The reset spec and implementation has been changed completely compared to 7 series FIFOs. RST is synchronous to WRCLK. To assert RST, it must be asserted for one setup time prior to the rising edge of WRCLK. RST has an optional inversion at the input.*

If RST is asserted, the WRRSTBUSY output asserts immediately after the rising edge of WRCLK, and remains asserted until the reset operation is complete. Following the assertion of WRRSTBUSY, the internal reset is synchronized to the RDCLK domain. Upon arrival in the RDCLK domain, the RDRSTBUSY is asserted, and is held asserted until the resetting of all RDCLK domain signals is complete, then RDRSTBUSY is deasserted. In common-clock Mode, this logic is simplified because the clock domain crossing is not required.

Operating Mode

There are two read operating modes in FIFO functions: standard and first-word fall-through (FWFT). The modes differ in output behavior immediately after the first word is written to a previously empty FIFO. For more details, see FWFT Mode.

Standard Mode

In a standard FIFO, EMPTY is deasserted when one or more words are available to be read from the FIFO. By asserting RDEN, the next piece of data appears on the DOUT bus after the next rising edge of RDCLK. When the EMPTY flag is asserted, no more data is available to be read, and any additional reads cause RDERR to assert after the next RDCLK edge. See Figure 1-23.

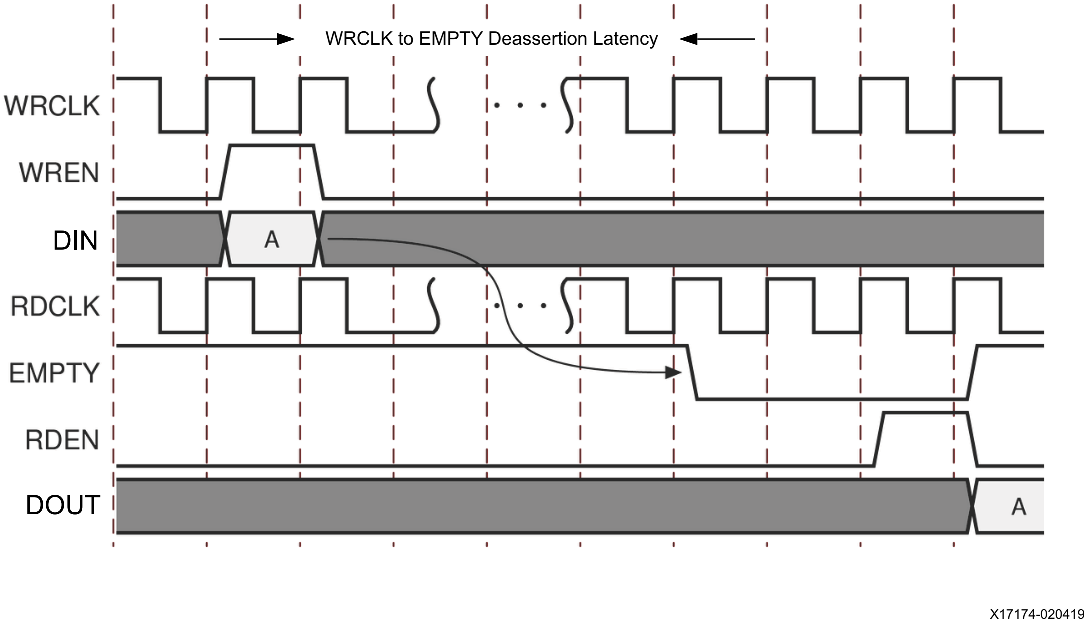


Figure 1-23: **Standard FIFO Interface**

FWFT Mode

In a first-word fall-through (FWFT) FIFO, the data is placed on the DOUT bus before RDEN is asserted. When writing the first word to an EMPTY FIFO, the first word is directly transferred to the output, and appears on the DOUT bus at the same time EMPTY is deasserted.

To prevent data loss in the system, this first output value does not disappear from the DOUT bus until RDEN is asserted. If no more data is available, EMPTY is asserted and the next word written to the FIFO again directly transfers to the output. Because the data appears on DOUT before the read operation, it can take one more last read operation for the FIFO to assert EMPTY (go empty). In either case, the same number of read operations as write operations are required to return the FIFO to an empty state. See Figure 1-24 and Figure 1-25.

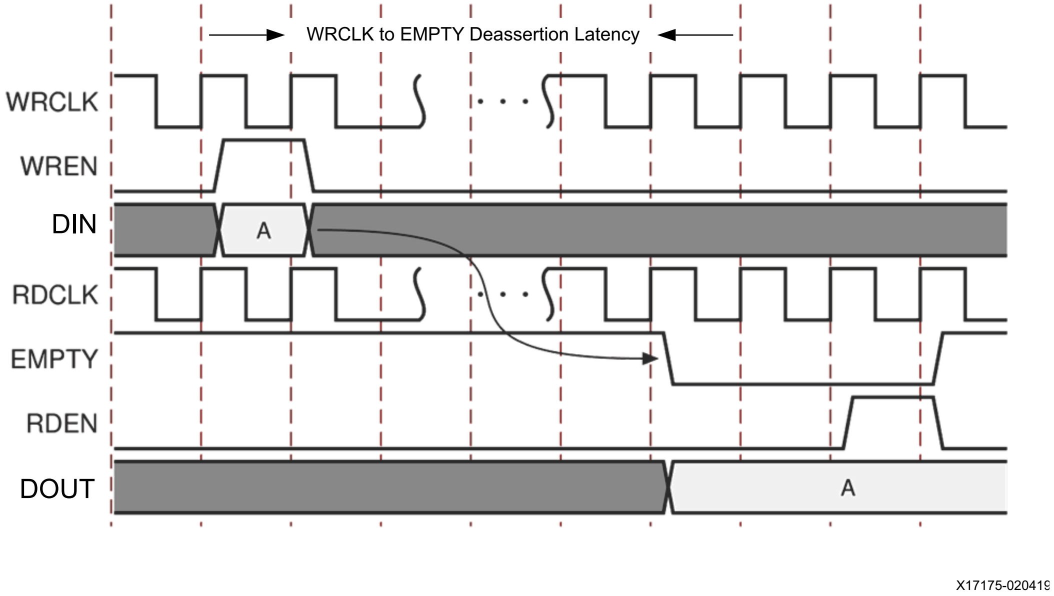


Figure 1-24: **FWFT FIFO with One Data Word in the FIFO**

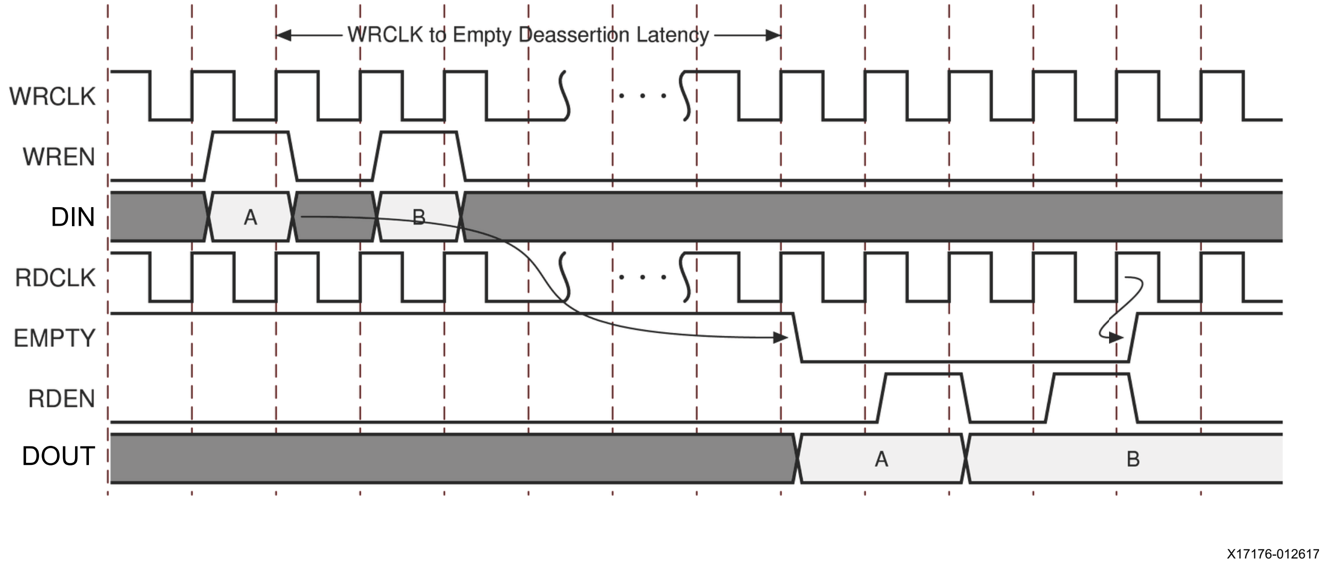


Figure 1-25: **FWFT FIFO with Two Data Words in the FIFO**

Flags

Empty Flag

If EMPTY is asserted, no data is available to be read from the FIFO, and any additional read operations cause a read error (RDERR=1). The relationship of EMPTY to the output data on the DOUT output port depends on whether the FIFO is configured as a standard FIFO or FWFT FIFO.

When writing to an empty FIFO, the number of clock cycles required for the EMPTY output to deassert depends on the FIFO configuration. For an independent-clocks FIFO, a write operation is synchronized internally to the RDCLK domain before it can influence the status of the Empty flag, resulting in a latency from the write operation to the deassertion of EMPTY that is a combination of a few write clocks followed by a few read clocks.

The Empty flag is synchronous to the RDCLK domain and is intended as a handshaking signal for logic reading from the FIFO.

PROGEMPTY Flag

If PROGEMPTY is asserted, the number of words in the FIFO is less than or equal to PROG_EMPTY_THRESH.

Because of the inherent latencies in the FIFO, especially for the independent-clocks FIFO, PROGEMPTY is always considered a pessimistic flag. This means that not all write operations might have synchronized to the RDCLK domain, and therefore the words in the FIFO might be reported as fewer than there actually are in the FIFO. However, because the PROGEMPTY flag is synchronous to the RDCLK domain, it is generally used to determine how many locations in the FIFO are available to be read, so the under-reporting of PROGEMPTY guarantees that the FIFO never underflows.

The number of clock cycles required for a write operation to cause PROGEMPTY to deassert depends on the FIFO configuration. For an independent-clocks FIFO, a write operation is first synchronized internally to the RDCLK domain before it can influence the status of the PROGEMPTY flag, resulting in a latency from the write operation to the deassertion of PROGEMPTY that is a combination of a few write clocks followed by a few read clocks.

The PROGEMPTY flag is synchronous to the RDCLK domain and is intended as a status signal for logic reading from the FIFO.

Read Error Flag

After the Empty flag has been asserted, any further read attempts do not increment the read address pointer but do trigger the Read Error (RDERR) flag. A RDERR also occurs if a write operation is performed while RDRSTBUSY is asserted. The RDERR flag is deasserted when Read Enable or Empty is deasserted. The RDERR flag is synchronous to RDCLK.

Full Flag

If FULL is asserted, the FIFO has no room for any additional words to be written to the FIFO, and any additional write operations cause a write error (WRERR=1). When reading from a full FIFO, the number of clock cycles required for the FULL output to deassert depend on the FIFO configuration. For an independent-clocks FIFO, a read operation must be synchronized internally to the WRCLK domain before it can influence the status of the FULL flag, resulting in a latency from the read operation to the deassertion of FULL that is a combination of a few read clocks followed by a few write clocks.

The FULL flag is synchronous to the WRCLK domain and is intended as a handshaking signal for logic writing to the FIFO.

Write Error Flag

After the Full flag is asserted, any further write attempts do not increment the write address pointer but do trigger the Write Error (WRERR) flag. A WRERR also occurs if a write operation is performed while WRRSTBUSY is asserted. The WRERR flag is deasserted when Write Enable or Full is deasserted. This signal is synchronous to WRCLK.

PROGFULL Flag

If PROGFULL is asserted, the number of words in the FIFO is greater than or equal to PROG_FULL_THRESH.

IMPORTANT: *The PROGFULL and PROGEMPTY flags have 1 clock cycle assertion latency. Consequently, the number of words can appear to be PROG_FULL_THRESH-1 by the time PROGFULL is asserted. See Table 1-29 and Table 1-30 for flag assertion latencies.*

Because of the inherent latencies in the FIFO, especially for the independent-clocks FIFO, PROGFULL is always considered a pessimistic flag. This means that not all read operations might have synchronized to the WRCLK domain, and therefore the words in the FIFO might be reported as more than there actually are in the FIFO. However, because the PROGFULL flag is synchronous to the WRCLK domain, PROGFULL is generally used to determine how many locations in the FIFO are available to be written, so the over-reporting of PROGFULL guarantees that too many words are never written to the FIFO.

The number of clock cycles required for a read operation to cause PROGFULL to deassert depends on the FIFO configuration. For an independent-clocks FIFO, a read operation is first synchronized internally to the WRCLK domain before it can influence the status of the PROGFULL flag, resulting in a latency from the read operation to the deassertion of PROGFULL that is a combination of a few read clocks followed by a few write clocks.

The PROGFULL flag is synchronous to the WRCLK domain and is intended as a status signal for logic writing to the FIFO.

Flag Assertion/Deassertion and Flag Latencies

Flag assertion and deassertion timing depends on the configuration of the FIFO. The common-clock FIFO configuration is not affected by the uncertainty of two unrelated clock domains, and requires no synchronization between clock domains. Therefore, the internal latencies from a write operation to the deassertion of EMPTY or PROGEMPTY, or from a read operation to the deassertion of FULL or PROGFULL, are much faster than in an equivalent independent-clock FIFO. Similarly, a FIFO configured with asymmetric ports has additional latencies depending on the port width ratios of the read and write port. The configuration of the REGISTER_MODE, FIRST_WORD_FALL_THROUGH, and EN_ECC_PIPE attributes can increase the latency from a write operation to the deassertion of EMPTY up to three additional RDCLK cycles.

Independent-clock FIFOs are synchronized between clock domains. Due to this internal synchronization between the WRCLK domain and the RDCLK domain, certain transitions take several clock cycles. For example, it takes several clock cycles (both WRCLK and RDCLK clock cycles) for the write operation to synchronize to the RDCLK domain. Only after the write operation is synchronized to the RDCLK domain is that write operation reflected in the status of the RDCLK outputs EMPTY and PROGEMPTY, and possibly cause these flags to deassert.

Similarly, the internal synchronization between the RDCLK domain and the WRCLK domain also takes several clock cycles. For example, it takes several clock cycles (both RDCLK and WRCLK clock cycles) for the read operation to synchronize to the WRCLK domain. Only after the read operation is synchronized to the WRCLK domain is that read operation reflected in the status of the WRCLK outputs FULL and PROGFULL, and possibly cause these flags to deassert. Due to the clock phase relationship uncertainty in the independent clock FIFO, the deassertion of the flags can vary by one clock cycle.

Table 1-29: **Independent-clock FIFO**

	Assertion[1]	Deassertion Standard FIFO[2]	Deassertion FWFT FIFO[2]
EMPTY	0 RDCLK	1 WRCLK and 4 or 5 RDCLK[3]	1 WRCLK and 5 or 6 RDCLK[3]
PROGEMPTY	1 RDCLK	1 WRCLK and 5 or 6 RDCLK	1 WRCLK and 5 or 6 RDCLK
FULL	0 WRCLK	1 RDCLK and 4 or 5 WRCLK	1 RDCLK and 4 or 5 WRCLK
PROGFULL	1 WRCLK	1 RDCLK and 5 or 6 WRCLK	1 RDCLK and 5 or 6 WRCLK

Notes:

1. Assertion latency is from the rising edge of the RDCLK/WRCLK with the RD/WR operation enabled if the operation caused the FIFO to go EMPTY (PROGEMPTY) or FULL (PROGFULL). A latency of zero indicates that the flag asserts immediately following the rising edge of the clock, and a latency of one indicates that one extra rising clock edge is required.
2. Deassertion latency starts from the rising edge of the RDCLK/WRCLK to the deassertion of the flag when the FIFO is no longer EMPTY (PROGEMPTY) or FULL (PROGFULL) (after the first read or write) and the read/write operation is enabled. Deassertion occurs after the first rising edge of the clock plus N cycles. N can vary due to the asynchronous nature of the clocks.
3. Registered mode adds one RDCLK clock cycle.

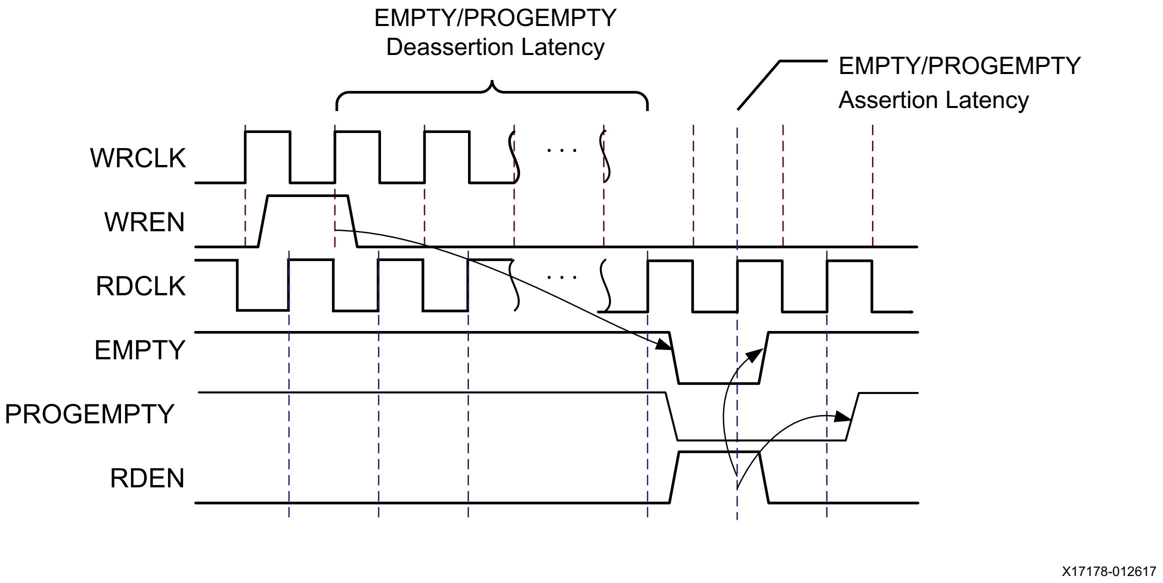


Figure 1-26: **Deassertion and Assertion Latencies of EMPTY and PROGEMPTY for Independent-Clock FIFO**

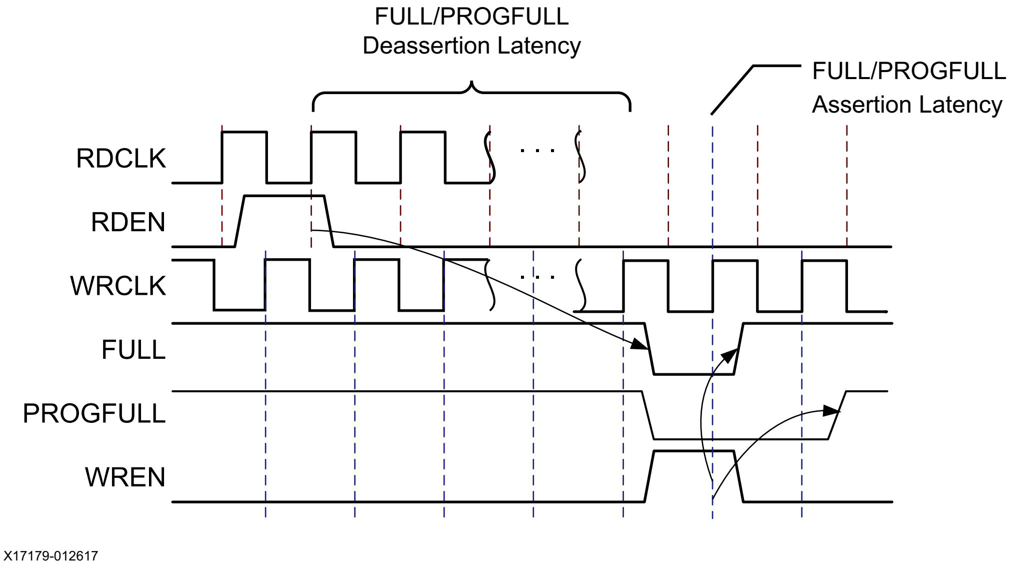


Figure 1-27: **Deassertion and Assertion Latencies of FULL and PROGFULL for Independent-Clock FIFO**

The programmable flags in Figure 1-26 and Figure 1-27 are asserted and deasserted based on their threshold settings and there is no dependency or relationship to the EMPTY/FULL flags.

Table 1-30: **Common-clock FIFO**

	Assertion(1)	**Deassertion Standard FIFO**(2)	**Deassertion FWFT FIFO**(2)
EMPTY	0 RDCLK	0 WRCLK(3)	1 WRCLK(3)
PROGEMPTY	1 RDCLK	1 WRCLK	1 WRCLK
FULL	0 WRCLK	0 RDCLK	0 RDCLK
PROGFULL	1 WRCLK	1 RDCLK	1 RDCLK

Notes:

1. Assertion latency is from the rising edge of the RD/WR with the RD/WR operation enabled if the operation caused the FIFO to go EMPTY (PROGEMPTY) or FULL (PROGFULL). A latency of zero indicates that the flag asserts immediately following the rising edge of the clock, and a latency of one indicates that one extra rising clock edge is required.
2. Deassertion latency is from the rising edge of the clock when the operation is enabled to the deassertion of the flag when the FIFO is no longer EMPTY (PROGEMPTY) or FULL (PROGFULL). A latency of zero indicates that the flag deasserts immediately following the rising edge of the clock, and a latency of one indicates that one extra rising clock edge is required.
3. Registered mode adds one RDCLK clock cycle.

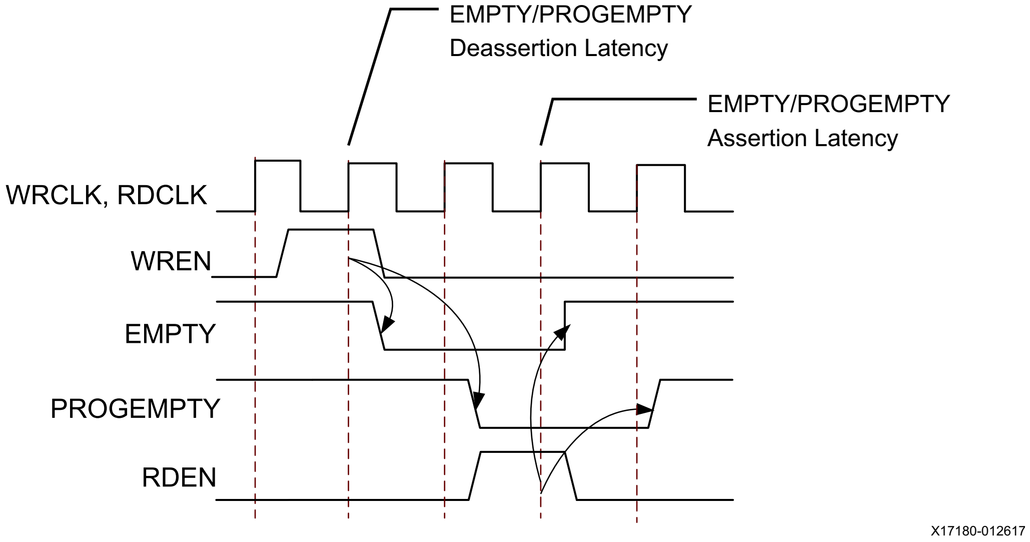


Figure 1-28: **Deassertion and Assertion Latencies of EMPTY and PROGEMPTY for Common-Clock FIFO**

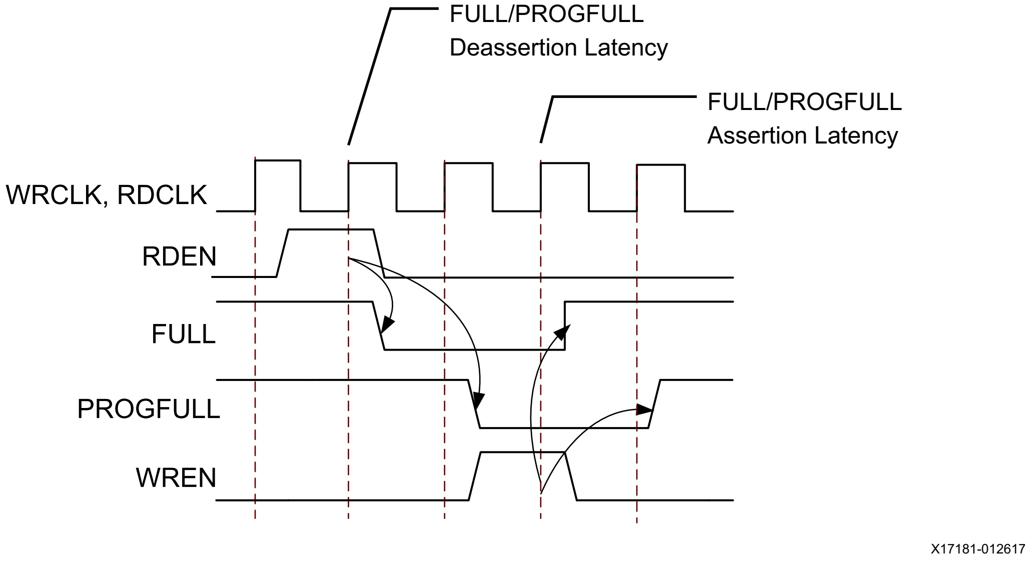


Figure 1-29: **Deassertion and Assertion Latencies of FULL and PROGFULL for Common-Clock FIFO**

The programmable flags in Figure 1-28 and Figure 1-29 are asserted and deasserted based on their threshold settings and there is no dependency or relationship to the EMPTY/FULL flags.

Cascading FIFOs Serially and in Parallel to Extend Depth

Cascading Serially

UltraScale™ architecture-based devices have built-in support for cascading FIFOs in series to extend depth without requiring logic resources. Dedicating routing and logic have been added to make this cascading mode available in hardware. Cascading FIFOs in series to expand depth is supported for FIFO18E2 primitives and for FIFO36E2 primitives, and two or more FIFOs can be cascaded up to a full column with some limitations (e.g., a PCIe® block interrupting the column). When cascading FIFOs serially, the first FIFO (the FIFO with the write side interface) must have CASCADE_ORDER = FIRST, the last FIFO (the one with the read interface) must have CASCADE_ORDER = LAST, and all other FIFOs in the chain must have CASCADE_ORDER = MIDDLE. FIFOs with a CASCADE_ORDER of FIRST or MIDDLE must be configured with FIRST_WORD_FALL_THROUGH = TRUE. The LAST FIFO in the chain can be in FWFT or standard mode. The FIFO control logic handles all the handshaking between the blocks and all the read and write interfaces. However, the RDCLK and WRCLK pins for all of the FIFOs in the chain must be connected in a specific way. The WRCLK input of the first FIFO should always use the user's WRCLK, the RDCLK input of the last FIFO should always use the user's RDCLK, and all other clock inputs should be connected to the faster of the two clocks. When both the WRCLK and RDCLK of a particular FIFO primitive is connected to the same clock input (same clock buffer source), the FIFO can be configured as a common clock FIFO to reduce the latencies through the FIFOs. The FIFOs can be configured in REGISTERED or UNREGISTERED mode. The REGISTERED mode provides maximum performance at the expense of increased latency for the WRCLK and RDCLK flag deassertion. If resetting the FIFO is required, the RST pins of the FIFOs must be connected by the application (tied to a single reset net). The FIFO control logic does not automatically handle the reset flags RDRSTBUSY and WRRSTBUSY, and therefore the application must monitor those if needed (e.g., ORing them).

Figure 1-30 shows a serial cascading example of three FIFOs.

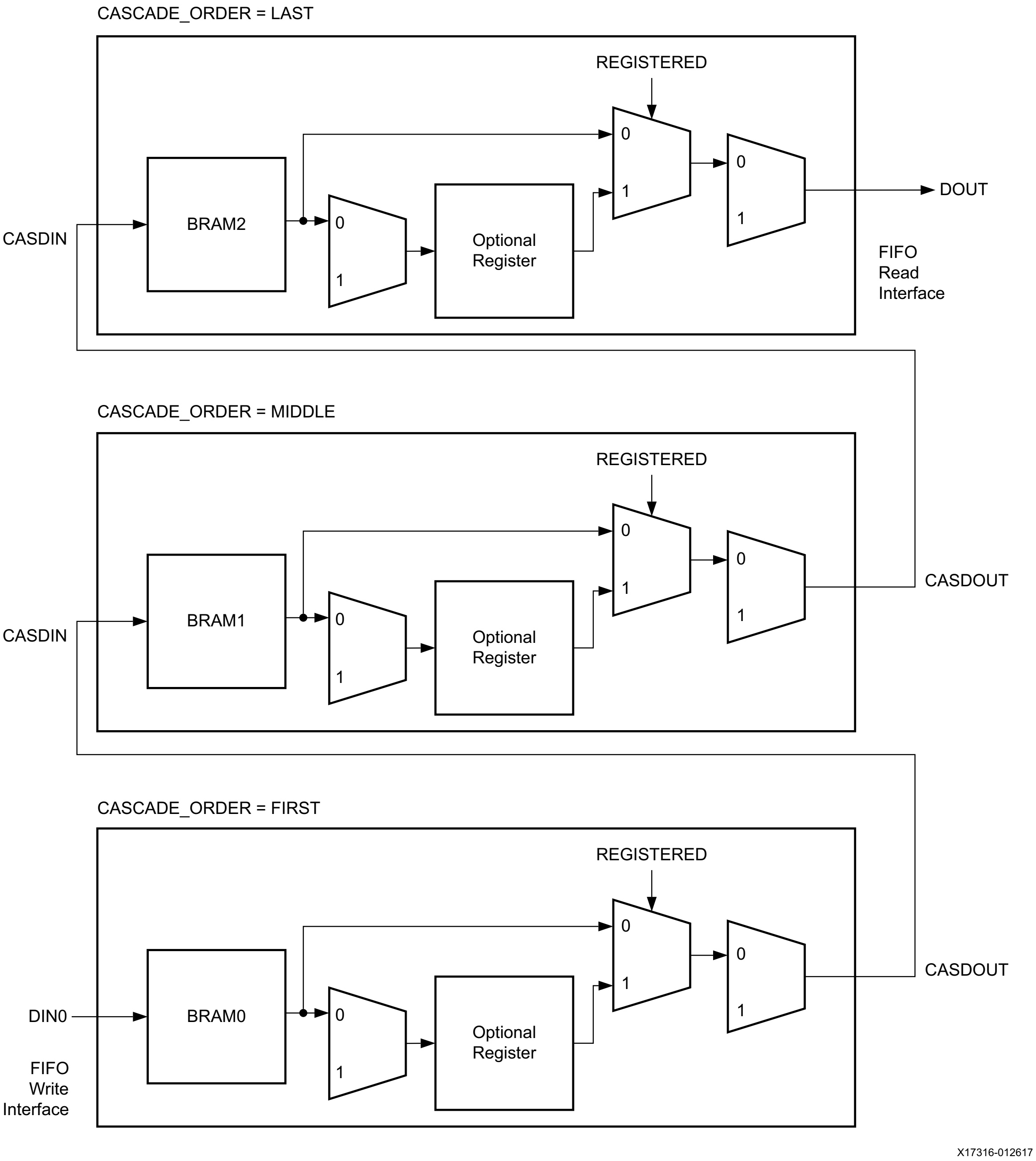


Figure 1-30: **FIFO Serial Cascade**

IMPORTANT: *In serial cascade mode, the multiplexer control signals are not accessible and are automatically configured.*

Aside from the dedicated data cascading pins CASDIN, CASDINP, CASDOUT, and CASDOUTP, the FIFO has four additional control pins to support serial cascading. These pins must be connected as follows:

- CASNXTEMPTY output: The CASNXTEMPTY output of a cascaded FIFO with CASCADE_ORDER = FIRST or MIDDLE is the cascaded EMPTY output from the current FIFO to the next in the chain. CASNXTEMPTY connects to the CASPRVEMPTY input on the next FIFO in the chain (configured with CASCADE_ORDER = MIDDLE or LAST), allowing the next FIFO to be aware of when this FIFO is not EMPTY and available for reading.
- CASPRVEMPTY input: The CASPRVEMPTY input of a cascaded FIFO with CASCADE_ORDER = MIDDLE or LAST is the cascaded EMPTY input from the previous FIFO in the chain to the current FIFO. CASPRVEMPTY connects to the CASNXTEMPTY output on the previous FIFO in the chain (configured with CASCADE_ORDER = FIRST or MIDDLE). When CASPRVEMPTY = 0, the current FIFO knows that it can transfer a word of data from the previous FIFO to the current FIFO.
- CASPRVRDEN output: The CASPRVRDEN output of a cascaded FIFO with CASCADE_ORDER = MIDDLE or LAST is the cascaded RDEN output from the current FIFO to the previous FIFO in the chain. CASPRVRDEN connects to the CASNXTRDEN input on the previous FIFO in the chain (configured with CASCADE_ORDER = FIRST or MIDDLE), indicating when to read from the previous FIFO as part of a data word transfer between the two FIFOs.
- CASNXTRDEN input: The CASNXTRDEN input of a cascaded FIFO with CASCADE_ORDER = FIRST or MIDDLE is the cascade RDEN input from the next FIFO to the current FIFO in the chain. CASNXTRDEN connects to the CASPRVRDEN output from the next FIFO in the chain (configured with CASCADE_ORDER = MIDDLE or LAST).

Cascading in Parallel

FIFOs can be cascaded in parallel mode based on the block RAM standard/pipelined data out cascade mode utilizing the same multiplexer pins available in the block RAM mode. This cascading mode is available in both FIFO18E2 and FIFO36E2. The parallel mode requires additional user logic and it is the application's responsibility to provide the appropriate logic for the read and write interfaces for each FIFO in the cascade chain as well as the multiplexer control.

When CASCADE_ORDER = PARALLEL, there are four additional FIFO inputs available that are used to control the cascade data multiplexers for that FIFO. These inputs are identical to the equivalent block RAM pins CASOREGIMUX and CASOREGIMUXEN (for controlling the pipeline register cascade multiplexer), and CASDOMUX and CASDOMUXEN (for controlling the output cascade multiplexer). In this mode, the special serial cascade control pins are not available (CASNXTEMPTY, CASPRVEMPTY, CASNXTRDEN, CASPRVRDEN).

IMPORTANT: *CASCADE_ORDER must be set to PARALLEL for the cascade input multiplexers to be available.*

Figure 1-31 shows a parallel cascading example of three FIFOs.

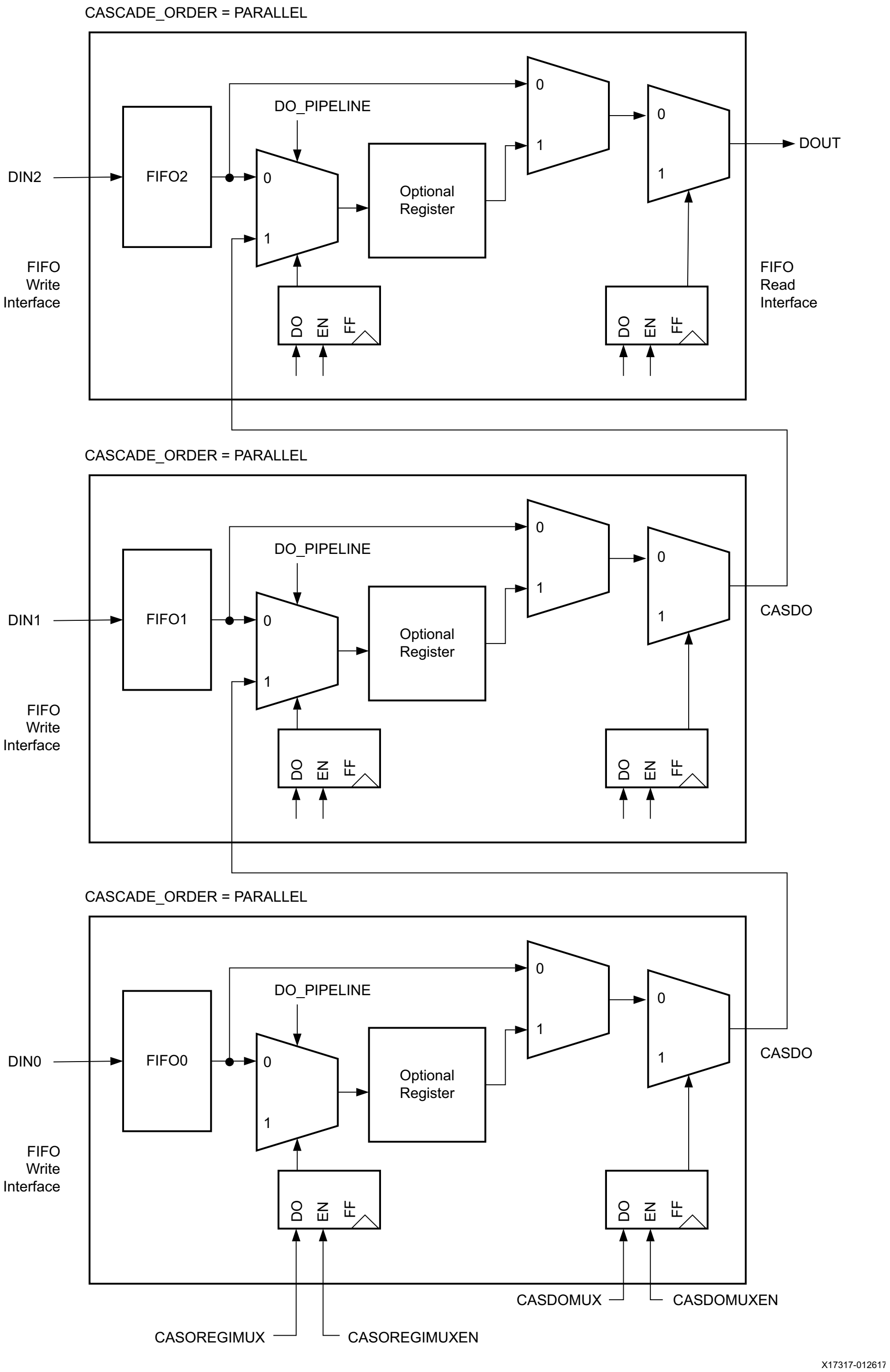


Figure 1-31: **FIFO Parallel Cascade**

IMPORTANT: *The CASOREGIMUX and CASOREGIMUXEN signals are only available in REGISTER_MODE = DO_PIPELINE.*

Built-in Error Correction

Overview

The RAMB36E2 in simple dual-port mode can be configured as a single 512 x 64 RAM with built-in Hamming code error correction using the extra eight bits in the 72-bit wide RAM. This operation is transparent.

Eight protection bits (ECCPARITY) are generated during each write operation and stored with the 64-bit data into the memory. These ECCPARITY bits are used during each read operation to correct any single-bit error, or to detect (but not correct) any double-bit error. The ECCPARITY bits are written into the memory and output to the FPGA logic at each rising edge of the WRCLK. There are no optional output registers available on the ECCPARITY output bits.

During each read operation, 72 bits of data (64 bits of data and 8 bits of parity) are read from the memory and fed into the ECC decoder. The ECC decoder generates two status outputs (SBITERR and DBITERR) that are used to indicate the three possible read results: No error, single-bit error corrected, and double-bit error detected. In the standard ECC mode, the read operation does not correct the error in the memory array, it only presents corrected data on DOUT. To improve F_{MAX}, optional registers controlled by the DO_REG attribute are available for data output (DOUT), SBITERR, and DBITERR. This is similar to the optional registers in the block RAM. For further F_{MAX} improvements, an additional ECC pipeline stage is available.

The ECC configuration option is available with a 36 Kb block RAM (RAMB36E2) in simple dual-port mode 72-bit width (64/8) (SDP) or a 36 Kb FIFO (FIFO36E2) in 72-bit width. Both read and write width must be 72 bits. The RAMB36E2 has the capability to inject errors. The RAMB36E2 has the ability to read back the address where the current data read out is stored. This feature better supports repairing a bit error or invalidating the content of that address for future access. The FIFO36E2 supports standard ECC mode with both the WRITE_WIDTH and READ_WIDTH set to 72 and has error-injection capability. FIFO36E2 does not output the address location being read.

The block RAM ECC also supports READ_FIRST, WRITE_FIRST, and NO_CHANGE modes in identical fashion to the SDP usage model.

ECC Modes

In the standard ECC mode (EN_ECC_READ = TRUE and EN_ECC_WRITE = TRUE), both encoder and decoder are enabled. During a write, 64-bit data and 8-bit ECC generated parity are stored in the array. The external parity bits are ignored. During a read, the 72-bit decoded data and parity (64-bit data and 8-bit parity) are read out, and the parity is checked against the data on DOUT.

The encoder and decoder can be accessed separately (independently) for external use in RAMB36E2 in simple dual-port mode and by extension in the FIFO36E2, both in 72-bit mode. To use the encoder by itself, the data needs to be sent through the DIN port, and the ECCPARITY output port can be sampled. To use the decoder by itself, the encoder is disabled, the data is written into the block RAM, and the corrected data and status bits are read out of the block RAM. See Block RAM and FIFO ECC Attributes, page 90.

The decoder can be used in two ways:

- To use the decoder in standard ECC mode, set (EN_ECC_WRITE = TRUE and EN_ECC_READ = TRUE).
- To use the decoder-only mode, set (EN_ECC_WRITE = FALSE and EN_ECC_READ = TRUE). The DIN data along with a user-generated parity is presented on the eight DINP ports. The data is not encoded. The read operation reads back the data on DOUT and performs the decoder parity check on the data.

The encoder can be used in two ways:

- To use the encoder in standard ECC mode, set (EN_ECC_WRITE = TRUE and EN_ECC_READ = TRUE).
- To use the encoder-only mode, set (EN_ECC_WRITE = TRUE and EN_ECC_READ = FALSE). The DIN data is presented, and the ECC encoded value of the DIN data is stored in the parity bits for every write operation. The read operation reads those eight bits without performing the encode function.

The functionality of the block RAM when using the ECC mode is described as follows:

- The block RAM ports still have independent address, clocks, and enable inputs, but one port is a dedicated write port, and the other is a dedicated read port (simple dual-port).
- DOUT represents the read data after correction.
- DOUT stays valid until the next active read operation.
- Simultaneous decoding and encoding of different read/write addresses is allowed. However, simultaneous decoding and encoding of the same read/write address is not allowed.
- In ECC configuration, the block RAM can be in either READ_FIRST, WRITE_FIRST, and NO_CHANGE mode. See also Address Collision in Chapter 1.

UltraScale™ architecture-based devices have an ECC pipeline mode. This is in addition to the optional registers on the outputs. These registers effectively pipeline the decoder for further improvement in maximum performance (F_{MAX}) and clock-to-out in latch mode. If turned on, the latency increases by one clock cycle because while the current address is read from the block RAM, the previous address is being decoded. The ECC pipeline register has a user accessible ENABLE control but does not have a reset control. Asserting the block RAM reset pins RSTRAM and RSTREG has no impact on this register, and the previously registered data remains in the register. When using EN_ECC_PIPE = TRUE with the FIFO, the FIFO controller always automatically manages the ECC pipeline register and the associated ECCPIPECE pin. The effective read depth of the FIFO is increase by one word, and the read clock to write flags EMPTY/PROGEMPTY deassertion latency also increase by one. However, the read to DOUT latency does not change.

Top-Level View of the Block RAM ECC Architecture

Figure 1-32 shows the top-level view of a block RAM in ECC mode.

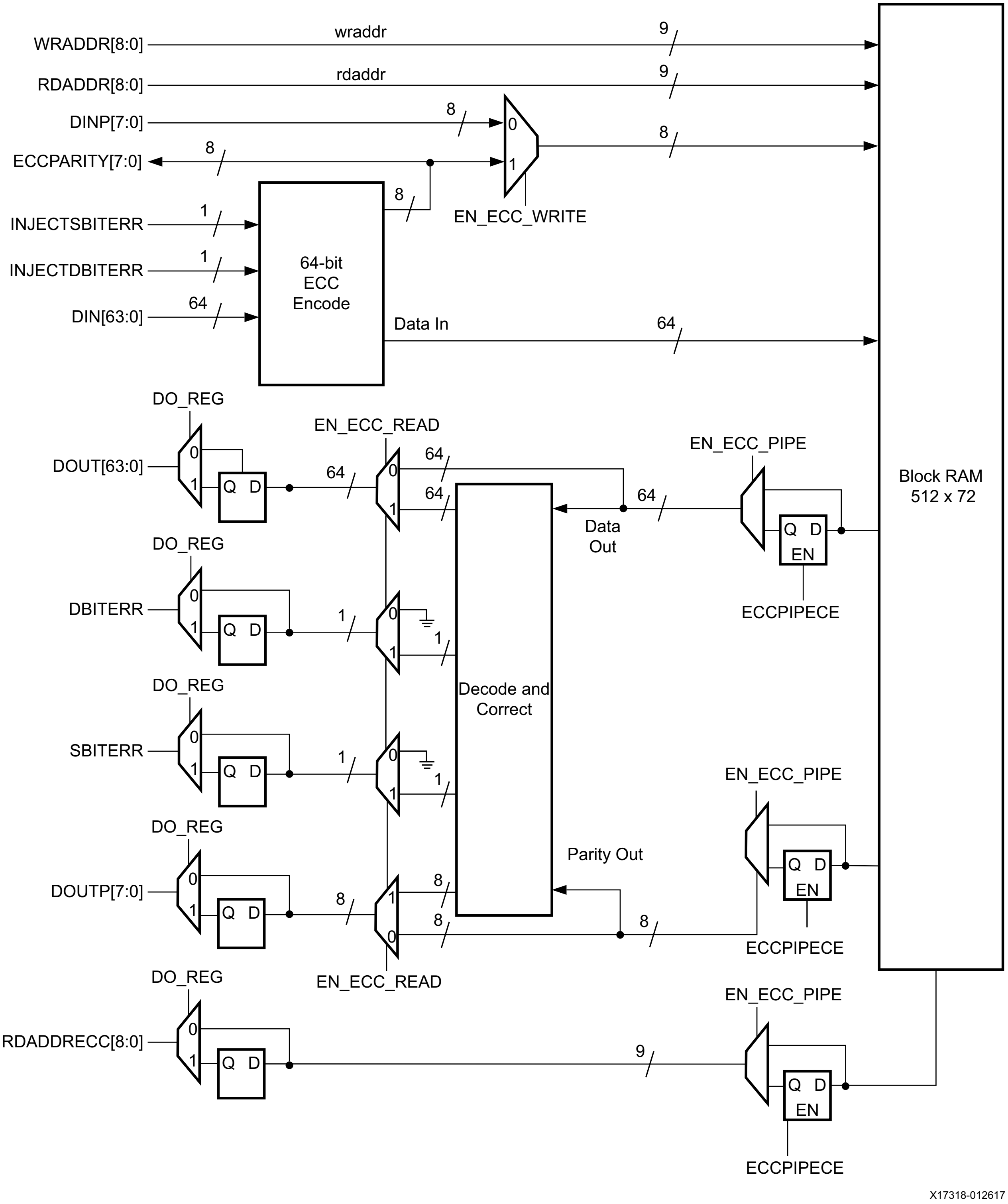


Figure 1-32: **Top-Level View of Block RAM ECC**

Block RAM and FIFO ECC Primitive

When using the RAMB36E2 and the FIFO36E2 in ECC mode, the input and output pins are identical to the block RAM and FIFO primitives and tables described earlier in this document. In addition to the pin names shown there, both block RAM and FIFO primitives have pins for their use in ECC mode. Table 1-31 and Table 1-32 describe the pins used in ECC mode only. The FIFO only supports standard ECC mode and does not support the RDADDRECC output.

Block RAM and FIFO ECC Port Descriptions

Table 1-31 lists and describes the block RAM and FIFO ECC-related I/O port names.

Table 1-31: **RAMB36E2 and FIFO32E2 ECC Port Names and Descriptions**

Port Name	Signal Description
INJECTSBERR	Injects a single-bit error if ECC is used. Creates a single-bit error at a particular block RAM bit location when asserted during write. The block RAM ECC logic corrects this error when this location is read back. The error is created in bit DIN[30].
INJECTDBERR	Injects a double-bit error if ECC is used. Creates a double-bit error at two particular block RAM bit locations when asserted during write. The block RAM ECC logic flags a double-bit error when this location is read back. When both INJECTSBERR and INJECTDBERR signals are simultaneously asserted, a double-bit error is injected. The errors are created in bits DIN[30] and DIN[62].
ECCPARITY[7:0]	ECC encoder output bus for ECC used in encode-only mode. This output cannot be cascaded.
SBERR	ECC single-bit error output status. See also the dedicated cascade pins in this table when using the block RAM and FIFO in ECC cascade mode.(1)
DBERR	ECC double-bit error output status. See also the dedicated cascade pins in this table when using the block RAM and FIFO in ECC cascade mode.(1)
RDADDRECC[8:0]	ECC read address. Address pointer to the data currently read out. The data and corresponding address are available in the same cycle. This output is not supported in the FIFO and cannot be cascaded.
CASINSBITERR	ECC single-bit error input in cascade mode. Cascade SBERR error bit status from the previous block RAM/FIFO.
CASOUTSBITERR	ECC single-bit error output in cascade mode. Cascade SBERR error bit status to the next block RAM/FIFO.
CASINDBITERR	ECC double-bit error input in cascade mode. Cascade DBERR error bit status from the previous block RAM/FIFO.
CASOUTDBITERR	ECC double-bit error output in cascade mode. Cascade DBERR error bit status to the next block RAM/FIFO.

Table 1-31: **RAMB36E2 and FIFO32E2 ECC Port Names and Descriptions *(Cont'd)***

Port Name	Signal Description
ECCPIPECE	ECC pipeline register clock enable when EN_ECC_PIPE = TRUE. This is available only in ECC mode when EN_ECC_READ = TRUE. In the FIFO, this function is controlled by the FIFO logic and not available to the application.

Notes:

1. Hamming code implemented in the block RAM ECC logic detects one of three conditions: no detectable error, single-bit error detected and corrected on DOUT (but not corrected in the memory), and double-bit error detected without correction. SBITERR and DBITERR indicate these three conditions.

Block RAM and FIFO ECC Attributes

Table 1-32 lists the block RAM and FIFO ECC attributes.

Table 1-32: **RAMB36E2 and FIFO32E2 Attributes related to ECC**

Attribute Name	Type	Values	Default	Notes
EN_ECC_WRITE	Boolean	TRUE, FALSE	FALSE	Set to TRUE to enable ECC encoder.
EN_ECC_READ	Boolean	TRUE, FALSE	FALSE	Set to TRUE to enable ECC decoder.
EN_ECC_PIPE	Boolean	TRUE, FALSE	FALSE	

Notes:

1. For UltraScale devices (not UltraScale+), in WRITE_FIRST or NO_CHANGE mode, if EN_ECC_READ/WRITE and EN_ECC_PIPE are set to TRUE, then both ENADDRENA and RDADDRCHANGEA cannot be set to TRUE at the same time. Consequently, the attribute combination of ENADDRENA and RDADDRCHANGEA and EN_ECC_READ and EN_ECC_PIPE cannot be set to TRUE at the same time in WRITE_FIRST or NO_CHANGE mode.

ECC Modes of Operation

There are three types of ECC operation: standard, encode only, and decode only. The standard ECC mode uses both the encoder and decoder.

Standard ECC

Set by Attributes

EN_ECC_READ = TRUE
EN_ECC_WRITE = TRUE

ECC Encode Only

Set by Attributes

EN_ECC_READ = FALSE
EN_ECC_WRITE = TRUE

ECC Encode-Only Read

ECC encode-only read is identical to normal block RAM read. The 64-bit data appears at DOUT[63:0] and 8-bit parity appears at DOUTP[7:0]. Single-bit error correction does not occur, and the error flags SBITERR and DBITERR are never asserted.

ECC Decode Only

Set by Attributes

EN_ECC_READ = TRUE
EN_ECC_WRITE = FALSE

In ECC decode-only mode, only the ECC decoder is enabled. The ECC encoder is disabled. Decode-only mode is used to inject single-bit or double-bit errors to test the functionality of the ECC decoder. The ECC parity bits must be externally supplied using the DINP[7:0] pins.

Creating 8 Parity Bits for a 64-bit Word

Using logic external to the block RAM (a large number of XOR circuits), 8 parity bits can be created for a 64-bit word. However, using ECC encoder-only mode, the 8 parity bits can be automatically created without additional logic by writing any 64-bit word into a separate block RAM. The encoded 8-bit ECC parity data is immediately available, or the complete 72-bit word can be read out.

Block RAM ECC VHDL and Verilog Templates

VHDL and Verilog templates are available in the Vivado Design Suite.

Chapter 2

UltraRAM Resources

UltraRAM Summary

UltraRAM is a single-clocked, two port, synchronous memory available in UltraScale+™ devices. Because UltraRAM is compatible with the columnar architecture, multiple UltraRAMs can be instantiated and directly cascaded in an UltraRAM column for the entire height of the device. A column in a single clock region contains 16 UltraRAM blocks. Devices with UltraRAM include multiple UltraRAM columns distributed in the device. Most of the devices in the UltraScale+ family include UltraRAM blocks. For the available quantity of UltraRAM in specific device families, see *UltraScale Architecture and Product Overview* (DS890) [Ref 1].

UltraRAM blocks are 288 Kb, single-clock, synchronous memory blocks arranged in one or more columns in the device. There are 16 UltraRAM blocks per clock region per column. Multiple UltraRAM blocks can be cascaded together within a column using dedicated cascade routing, and the only limit is the height of the device or a single super logic region (SLR) in a stacked silicon interconnect (SSI) device. In addition, multiple columns can be cascaded together using a small quantity of logic resources. There is no timing penalty with cascading UltraRAM blocks if they are appropriately pipelined.

UltraRAM is a flexible, high-density memory building block. Each UltraRAM block can store up to 288K bits of data and is configured as a 4K x 72 memory block. UltraRAM has eight times the capacity of a block RAM. Similar to the block RAM, there are multiple UltraRAM columns distributed on the device. UltraRAM has two ports, both of which address all 4K x 72 bits. Each port can independently perform either one read or one write operation per clock cycle per port. However, internally the SRAM array uses single port memory cells. Dual port operation is achieved by executing port A operation followed by port B operation in a single cycle. Therefore, both ports share a single clock input. Each port can only execute either a write or read operation in one cycle. When executing a write operation, the read outputs are unchanged and hold the previous value.

The 288 Kb blocks can be cascaded to facilitate deeper memory implementations. Most of the routing related to cascading is contained inside the UltraRAM columns. Therefore, very little or no general interconnect is required and timing penalties are not incurred due to routing if the UltraRAM blocks are appropriately pipelined.

UltraRAM contains up to four pipeline stages for each of the two port interfaces. In a standalone, non-cascaded mode, the UltraRAM can be configured for one to four clock cycles latency, though typically, only one to three cycles of latency are required, depending on the target frequency. Cascade mode latency is a function of the size of the UltraRAM chain, frequency target, and other constraints. Similarly, clock-to-out performance depends on the selected output registers. Use the Vivado tool set to determine the performance and clock-to-out timing for specific design implementations.

UltraRAM Key Features

The UltraRAM key features are:

- 288K bits of storage in a single block.
- Dual port, 4K x 72, single clock synchronous memory.
- UltraRAM cascade for building larger blocks. UltraRAM has dedicated routing resources for appropriate inputs and outputs to cascade from lower UltraRAM to upper UltraRAM.
- Error correction coding (ECC) on both ports with single bit error detection and correction and double bit error detection.
- Sleep power saving features.
- Automatic power savings through automatic invocation/release of the sleep mode in a chain of UltraRAM blocks.
- Optional pipeline flip-flops on the inputs, outputs, and cascade paths.
- Data out reset capability for outputs to be reset to all 0's.
- Single port width of x72.
- The UltraRAM memory is initialized to all 0's during power up or device reset. There is no user defined INIT attribute and therefore the content of the SRAM array cannot be initialized to user defined values.

UltraRAM Cascade

The UltraRAM cascade features are:

- UltraRAM has dedicated routing resources for most of the inputs and outputs to cascade from lower UltraRAM to upper UltraRAM.
- Built-in address decode logic for 11 bits of the MSB address is used when cascading UltraRAMs to automatically generate internal enable for the read and write operations.
- Cascading in a column is supported in one direction from bottom to top and can be implemented without using general interconnect resources.
- Cascading between columns requires the use of device routing and potentially logic resources at the entry and exit points of each column.

UltraRAM Error Correction Coding

One 64-bit ECC block is provided per UltraRAM block. Independent encode and decode functionality is also available. The ECC mode can inject errors. The UltraRAM ECC features are:

- Optional ECC encode and decode on both ports.
- Single and double bit error detection.
- Single bit error correction.
- Single bit or double bit error injection capability.
- Optional pipeline register after ECC decode logic for maximum performance.

Block RAM and UltraRAM Differences

The key differences between block RAM and UltraRAM are:

- UltraRAM has one single clock input, is fully synchronous and, unlike the block RAM, does not support independent clock interfaces directly.
- There is no support for configurable port widths of a single 4K x 72 UltraRAM or multiple clock domains. However, the byte write enable feature of the UltraRAM block can be used to support this outside the memory structure itself.
- UltraRAM can only support read or write per port per cycle.
- The simple dual-port (SDP) and true dual-port (TDP) block RAM modes do not directly apply to the UltraRAM. The UltraRAM port behavior can be viewed as a superset of SDP, but not TDP.
- Fixed read behavior; there are no user definable read-first, write-first, no-change modes with UltraRAM.
- Static data cascading; there are no dynamic cascade input or output multiplexer controls with UltraRAM.
- Address collision is not possible with UltraRAM.
- UltraRAM cascades data, address, and control signals, and not just the data lines.
- During the UltraRAM power saving mode (SLEEP), user operations are ignored and content corruption is not possible as long as setup and hold times are met. The memory content is preserved in the sleep power saving mode.
- Automatic power savings can be achieved by using the auto sleep feature that independently controls the wake-up and the sleep mode based on activity. This mode dynamically turns sleep mode on or off for selected UltraRAM blocks in a chain by predicting the activity of many cascaded UltraRAM blocks in a column or across several columns. For single UltraRAM block applications, using this feature would require many cycles of inactivity to be beneficial.

IMPORTANT: *The clock minimum pulse width and setup/hold time of the UltraRAM address, enable, and sleep pins must not be violated. Violating the clock minimum pulse width or these setup/hold times (even if write enable is Low) can corrupt the data contents of the UltraRAM. This most commonly occurs from an unstable clock or when flip-flops driving UltraRAM control pins are asynchronously reset, such as a system wide reset. To avoid this issue, ensure stable clocks and design with synchronous resets for both assertion and deassertion. When the clock is not stable, disable the clock buffer or disable logic driving the UltraRAM control pins or deassert the UltraRAM EN input.*

Block RAM and UltraRAM Comparison

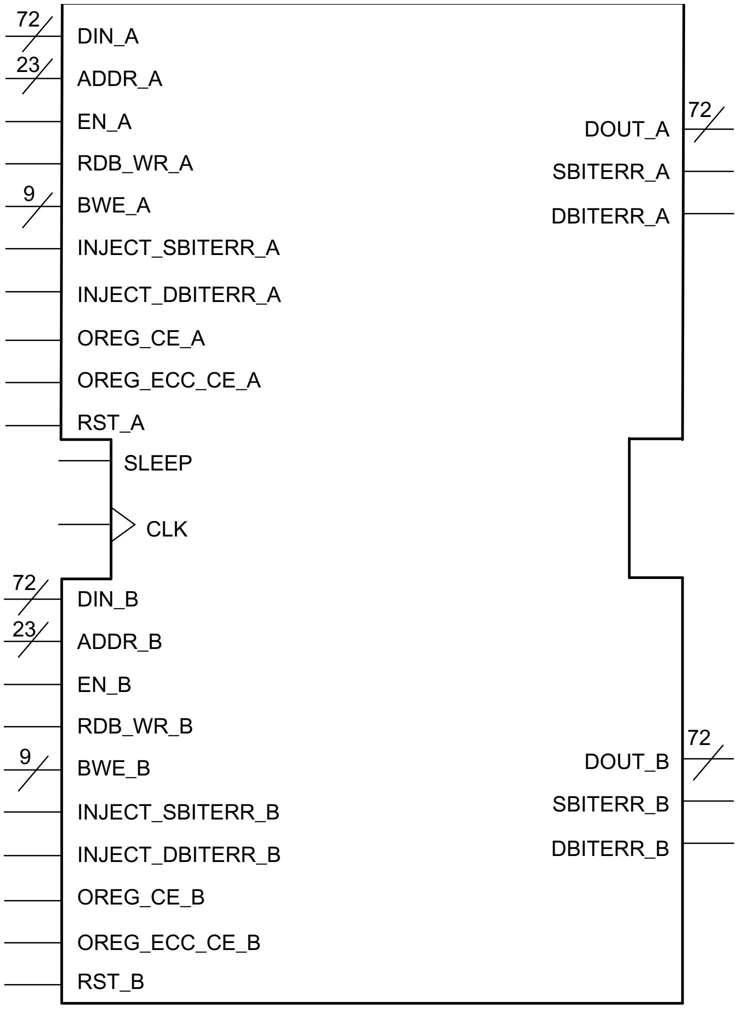
Table 2-1 shows a comparison of the block RAM and UltraRAM main features.

Table 2-1: **Block RAM and UltraRAM Comparison**

Feature	Block RAM	UltraRAM
Clocking	Two clocks	Single clock
Built-in FIFO	Yes	No
Data width	Configurable (1, 2, 4, 9, 18, 36, 72)	Fixed (72-bits)
Modes	SDP and TDP	Two ports, each can independently read or write (a superset of SDP)
ECC	64-bit SECDED Supported in 64-bit SDP only (one ECC decoder for port A and one ECC encoder for port B)	64-bit SECDED One set of complete ECC logic for each port to enable independent ECC operations (ECC encoder and decoder for both ports)
Cascade	• Cascade output only (input cascade implemented via logic resources) • Cascade within a single clock region	• Cascade both input and output (with global address decoding) • Cascade across clock regions in a column • Cascade across several columns with minimal logic resources
Power savings	One mode via manual signal assertion	One mode via manual signal assertion

UltraRAM Primitives

The UltraRAM URAM288 and URAM288_BASE library primitives are the basic building blocks for all UltraRAM configurations. The URAM288 primitive supports all possible configurations including cascade and ECC. The URAM288_BASE primitive is a subset and supports single UltraRAM block instances without cascade capability. The URAM288_BASE primitive is shown in Figure 2-1 and the URAM288 primitive is shown in Figure 2-2.



X17182-012617

Figure 2-1: **UltraRAM URAM288_BASE Primitive**

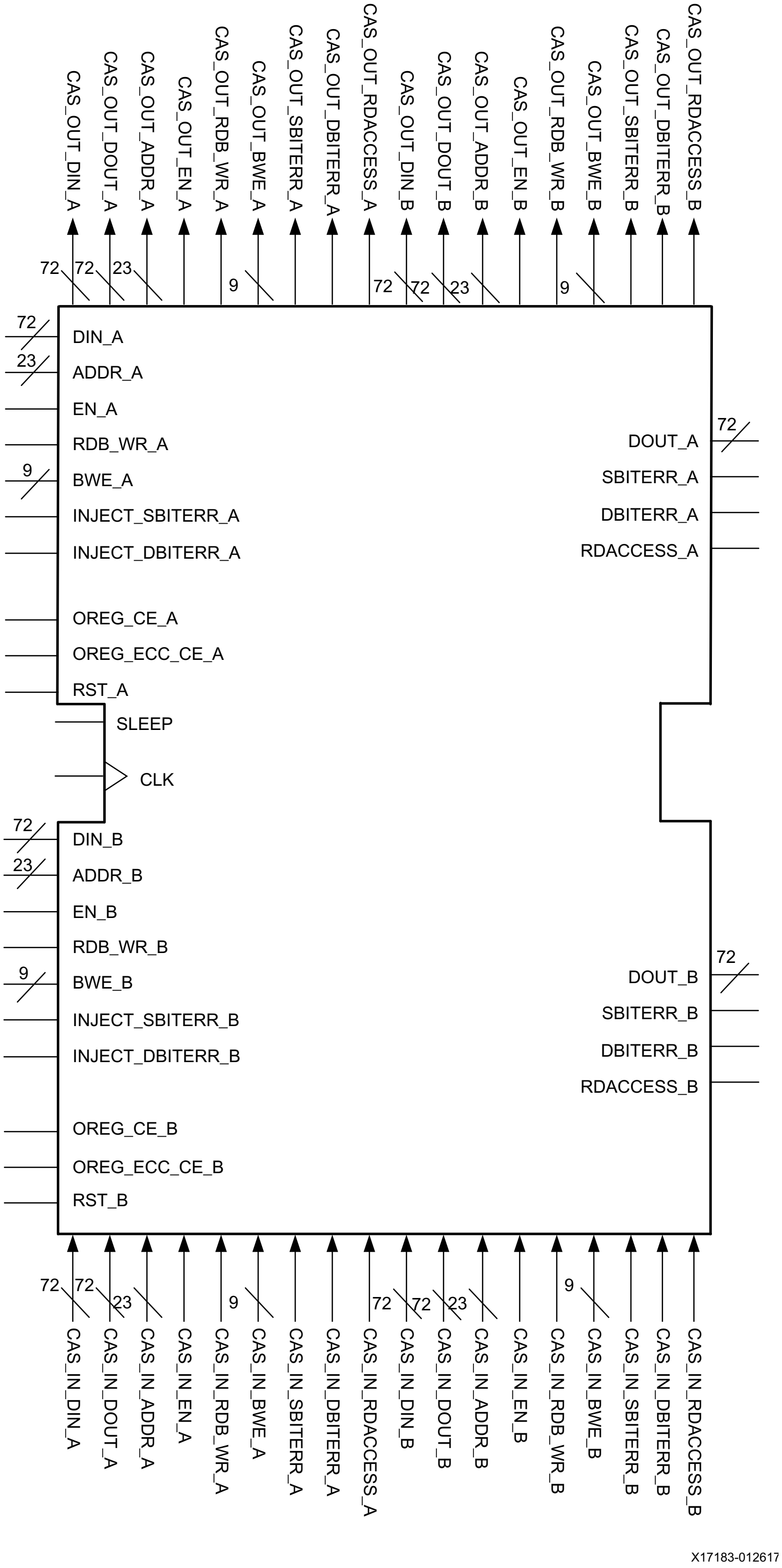


Figure 2-2: **UltraRAM URAM288 Primitive**

Figure 2-3 depicts the simplified single UltraRAM block diagram without cascade with one port shown.

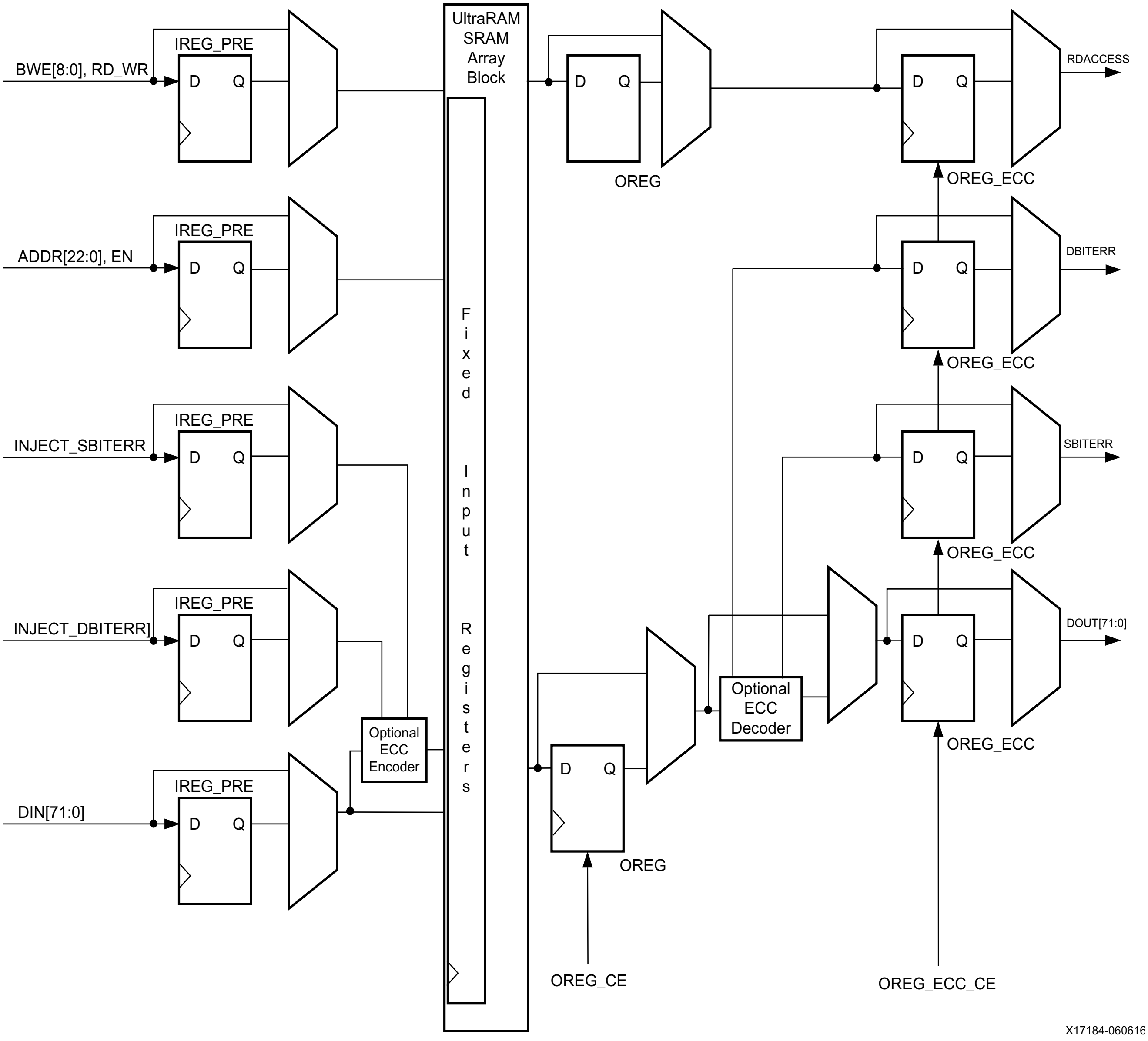


Figure 2-3: **Simplified Single UltraRAM Block Diagram without Cascade (One Port shown)**

UltraRAM Port Names and Description

This section describes the UltraRAM port names.

No Cascade Ports

Table 2-2 lists the UltraRAM no cascade ports.

Table 2-2: **No Cascade Ports**

Port Name	Description
CLK	UltraRAM clock source.
SLEEP	Dynamic power gating control.
Port A Inputs	
ADDR_A[22:0]	Port A address. ADDR_A[22:12] are only used in cascade mode.
EN_A	Port A enable. Enables or disables the read/write access to the block RAM memory core.
RDB_WR_A	Port A read or write mode input select. Read (BAR) is active Low (0 = read and 1 = write).
BWE_A[8:0]	Port A byte write enable.
DIN_A[71:0]	Port A write data in.
INJECT_SBITERR_A	Port A single-bit error injection during write.
INJECT_DBITERR_A	Port A double-bit error injection during write.
OREG_CE_A	Port A SRAM array core block read output pipeline register CLK enable.
OREG_ECC_CE_A	Port A ECC decoder output pipeline register CLK enable.
RST_A	Asynchronous or synchronous reset for port A output registers. Reset has priority over CE.
Port A Outputs	
DOUT_A[71:0]	Port A read data out.
RDACCESS_A	Port A read status output.
SBITERR_A	Port A single-bit error output status.
DBITERR_A	Port A double-bit error output status.
Port B Inputs	
ADDR_B[22:0]	Port B address. ADDR_B[22:12] are only used in cascade mode.
EN_B	Port B enable. Enables or disables the read/write access to the block RAM memory core.
BWE_B[8:0]	Port B byte write enable.
DIN_B[71:0]	Port B write data in.
INJECT_SBITERR_B	Port B single-bit error injection during write.

Table 2-2: **No Cascade Ports *(Cont'd)***

Port Name	Description
INJECT_DBITERR_B	Port B double-bit error injection during write.
OREG_CE_B	Port B SRAM array core block read output pipeline register CLK enable.
OREG_ECC_CE_B	Port B ECC decoder output pipeline register CLK enable.
RST_B	Asynchronous or synchronous reset for port B output registers. Reset has priority over CE.
Port B Outputs	
DOUT_B[71:0]	Port B read data out.
RDACCESS_B	Port B read status output.
SBITERR_B	Port B single-bit error output status.
DBITERR_B	Port B double-bit error output status.

Cascade Ports

Table 2-3 lists the UltraRAM cascade ports. Input ports are cascaded from the block below and output ports are cascaded to the block above.

Table 2-3: **Cascade Ports**

Port Name	Description
Port A Cascade Inputs	
CAS_IN_ADDR_A[22:0]	Port A input address input. In cascade mode, connect this port to CAS_OUT_ADDR_A.
CAS_IN_EN_A	Port A input enable input. In cascade mode connect this port to CAS_OUT_EN_A.
CAS_IN_BWE_A[8:0]	Port A input write mode port byte write enable. In cascade mode, connect this port to CAS_OUT_BWE_A.
CAS_IN_RDB_WR_A	Port A input read/write mode select. In cascade mode, connect this port to CAS_OUT_RDB_WR_A.
CAS_IN_DIN_A[71:0]	Port A input write mode. In cascade mode, connect this port to CAS_OUT_DIN_A.
CAS_IN_DOUT_A[71:0]	Port A input read mode data output. In cascade mode, connect this port to CAS_OUT_DOUT_A.
CAS_IN_RDACCESS_A	Port A input read mode read status. In cascade mode, connect this port to CAS_OUT_RDACCESS_A.
CAS_IN_SBITERR_A	Port A input read mode single-bit error flag input. In cascade mode, connect this port to CAS_OUT_SBITERR_A.
CAS_IN_DBITERR_A	Port A input read mode double-bit error flag input. In cascade mode, connect this port to CAS_OUT_SBITERR_A.
Port A Cascade Outputs	
CAS_OUT_ADDR_A[22:0]	Port A output address. In cascade mode, connect this port to CAS_IN_ADDR_A.

Table 2-3: **Cascade Ports *(Cont'd)***

Port Name	Description
CAS_OUT_EN_A	Port A output enable. In cascade mode, connect this port to CAS_IN_EN_A.
CAS_OUT_RDB_WR_A	Port A output read/write mode select. In cascade mode, connect this port to CAS_IN_RDB_WR_A.
CAS_OUT_BWE_A[8:0]	Port A output write mode byte write enable. In cascade mode, connect this port to CAS_IN_BWE_A.
CAS_OUT_DIN_A[71:0]	Port A output write mode data. In cascade mode, connect this port to CAS_IN_DIN_A.
CAS_OUT_DOUT_A[71:0]	Port A output read mode data. In cascade mode, connect this port to CAS_IN_DOUT_A.
CAS_OUT_RDACCESS_A	Port A output read mode read status flag. In cascade mode, connect this port to CAS_IN_RDACCESS_A.
CAS_OUT_SBITERR_A	Port A output read single-bit error flag. In cascade mode, connect this port to CAS_IN_SBITERR_A.
CAS_OUT_DBITERR_A	Port A output read mode double-bit error flag. In cascade mode, connect this port to CAS_IN_DBITERR_A.
Port B Cascade Inputs	
CAS_IN_ADDR_B[22:0]	Port B input address input. In cascade mode, connect this port to CAS_OUT_ADDR_B.
CAS_IN_EN_B	Port B input enable input. In cascade mode, connect this port to CAS_OUT_EN_B.
CAS_IN_BWE_B[8:0]	Port B input write mode port byte write enable. In cascade mode, connect this port to CAS_OUT_BWE_B.
CAS_IN_RDB_WR_B	Port B input read/write mode select. In cascade mode, connect this port to CAS_OUT_RDB_WR_B.
CAS_IN_DIN_B[71:0]	Port B input write mode. In cascade mode, connect this port to CAS_OUT_DIN_B.
CAS_IN_DOUT_B[71:0]	Port B input read mode data output. In cascade mode, connect this port to CAS_OUT_DOUT_B.
CAS_IN_RDACCESS_B	Port B input read mode read status. In cascade mode, connect this port to CAS_OUT_RDACCESS_B.
CAS_IN_SBITERR_B	Port B input read mode single-bit error flag input. In cascade mode, connect this port to CAS_OUT_SBITERR_B.
CAS_IN_DBITERR_B	Port B input read mode double-bit error flag input. In cascade mode, connect this port to CAS_OUT_DBITERR_B.

Table 2-3: **Cascade Ports *(Cont'd)***

Port Name	Description
Port B Cascade Outputs	
CAS_OUT_ADDR_B[22:0]	Port B output address. In cascade mode, connect this port to CAS_IN_ADDR_B.
CAS_OUT_EN_B	Port A output enable. In cascade mode, connect this port to CAS_IN_EN_B.
CAS_OUT_BWE_B[8:0]	Port B output write mode byte write enable. In cascade mode, connect this port to CAS_IN_BWE_B.
CAS_OUT_RDB_WR_B	Port B output read/write mode select. In cascade mode, connect this port to CAS_IN_RDB_WR_B.
CAS_OUT_DIN_B[71:0]	Port B output write mode data. In cascade mode, connect this port to CAS_IN_DIN_B.
CAS_OUT_DOUT_B[71:0]	Port B output read mode data. In cascade mode, connect this port to CAS_IN_DOUT_B.
CAS_OUT_RDACCESS_B	Port B output read mode read status flag. In cascade mode, connect this port to CAS_IN_RDACCESS_B.
CAS_OUT_SBITERR_B	Port B output read single-bit error flag. In cascade mode, connect this port to CAS_IN_SBITERR_B.
CAS_OUT_DBITERR_B	Port B output read mode double-bit error flag. In cascade mode, connect this port to CAS_IN_DBITERR_B.

UltraRAM Port Signals

Clock – CLK

Each port is fully synchronous with a single clock pin for both ports. All port input pins have the setup time referenced to this CLK pin. The output data bus has a clock-to-out time referenced to the CLK pin. Clock polarity is configurable (rising edge by default).

Power Gating Enable Input – SLEEP

The dynamic power gating capability can be used to save static power when the memory is not actively used for extended periods of time.

When sleep mode is asserted, and setup and hold times are met, the memory starts going into sleep mode in the next clock cycle. The SLEEP inputs disable the UltraRAM read and write operation. Consequently, if a read or write operation is attempted, it is ignored until after the wake-up time is satisfied. However, setup and hold times must be met. While in sleep mode, the output of the SRAM array and the OREG pipeline registers are synchronously reset to "0" with the next rising edge of clock. The other optional pipeline registers are not affected by the sleep mode. Therefore, the ultimate data output value of the UltraRAM is either held at its previous value or appears to be reset to "0" depending on the usage of the other pipeline registers. The output of the OREG register is held to "0" until the first valid read data (after wake-up time) flows through the pipeline.

The SLEEP pin controls the power gating of the RAM. When SLEEP = 1, the SRAM peripheral logic is powered down to save energy. The data in the SRAM array is retained but it cannot be read from or written to. SLEEP allows a two clock cycles wake-up time with no impact on SEU performance. The polarity of this pin is not configurable (active High).

Wake-up time defines when the EN pin can be asserted after SLEEP has been deasserted. The clock wake-up cycles mentioned previously assume no optional pipelines are enabled.

CAUTION! *If the OREG is used (OREG=TRUE) and a read operation is followed immediately by a SLEEP operation (SLEEP going active), then the read operation data does not exit the UltraRAM block because the OREG pipeline stage is powered down immediately. The RDACCESS signal is still asserted although the data is not observed at the output.*

Address Bus – ADDR_A, ADDR_B

The 23-bit address bus selects the memory cells for read or write. The lower 12 bits are used to select memory cells within the 4K location in each UltraRAM. The upper 11 bits select the UltraRAM that is used for cascading multiple UltraRAMs to form deep memory arrays. Each UltraRAM has a built-in comparator, which compares the upper 11 bit address with a unique SELF_ADDR attribute to identify if the UltraRAM has been selected. The SELF_MASK attribute defines how many of the 11 bit addresses should be used for the compare.

Enable – EN_A and EN_B

The enable pin affects the read and write functionality of the port. Ports with an inactive enable pin keep the output pins in the previous state and do not write data to the memory cells. Enable polarity is invertible (active High by default). However, during reset or the power saving mode (SLEEP), the outputs are reset to “0”.

Read/ Write Select – RDB_WR_A and RDB_WR_B

When this pin is “1”, it selects the write operation and when it is “0”, it selects the read operation. The polarity is invertible. Each port can only execute a read or write operation in one cycle. When a write operation is executed, the read outputs hold the previous value.

Byte-Wide Write Enable – BWE_A, BWE_B

Byte-wide write enable (BWE) is a 9-bit bus. Depending on the BWE_MODE_A/B attribute setting, bit 9 (BWE[8]) might not be used. In PARITY_INTERLEAVED mode, only the eight least significant bits are used. A single parity bit for each of the DIN bytes 0-7 corresponds to and is written to the related parity bit in the MSB of the DIN bus. This mode supports a custom parity scheme. In PARITY_INDEPENDENT mode, the nine BWE bits correspond to a byte of the DIN bus. Table 2-4 details how the BWE_A/B bits can be used to enable the corresponding DIN bits during a write operation. The byte write enable inputs are ignored during a read operation.

IMPORTANT: *All byte write enable bits must be set to "1" in ECC mode for proper operation.*

Data-In Buses – DIN_A, DIN_B

Data-in buses provide the new data value to be written into UltraRAM. The data bus is 72-bits wide with the lower 64 bits used for data and the upper 8 bits used for parity or for regular data inputs.

Table 2-4: **Byte Write Enable (URAM288)**

BWE_MODE_A/B	BWE_A/B	DIN_A/B
Parity Interleaved Mode		
PARITY_INTERLEAVED	BWE_A/B[7]	DIN_A/B[71,63:56]
PARITY_INTERLEAVED	BWE_A/B[6]	DIN_A/B[70,55:48]
PARITY_INTERLEAVED	BWE_A/B[5]	DIN_A/B[69,47:40]
PARITY_INTERLEAVED	BWE_A/B[4]	DIN_A/B[68,39:32]
PARITY_INTERLEAVED	BWE_A/B[3]	DIN_A/B[67,31:24]
PARITY_INTERLEAVED	BWE_A/B[2]	DIN_A/B[66,23:16]
PARITY_INTERLEAVED	BWE_A/B[1]	DIN_A/B[65,15:8]
PARITY_INTERLEAVED	BWE_A/B[0]	DIN_A/B[64,7:0]
Parity Independent Mode		
PARITY_INDEPENDENT	BWE_A/B[8]	DIN_A/B[71:64]
PARITY_INDEPENDENT	BWE_A/B[7]	DIN_A/B[63:56]
PARITY_INDEPENDENT	BWE_A/B[6]	DIN_A/B[55:48]
PARITY_INDEPENDENT	BWE_A/B[5]	DIN_A/B[47:40]
PARITY_INDEPENDENT	BWE_A/B[4]	DIN_A/B[39:32]
PARITY_INDEPENDENT	BWE_A/B[3]	DIN_A/B[31:24]
PARITY_INDEPENDENT	BWE_A/B[2]	DIN_A/B[23:16]
PARITY_INDEPENDENT	BWE_A/B[1]	DIN_A/B[15:8]
PARITY_INDEPENDENT	BWE_A/B[0]	DIN_A/B[7:0]

Inject Single and Double Bit Error Inputs – INJECT_SBITERR_A, INJECT_DBITERR_A, INJECT_SBITERR_B, INJECT_DBITERR_B

The inject error inputs can induce a single or double bit error on write data input for testing purposes.

Register Enable for OREG Pipeline Stage – OREG_CE_A, OREG_CE_B

This register enable pin controls the first optional output register. When this register is enabled using the OREG_A/B attribute, and the corresponding CE input is High, the read data is stored in the register at the rising clock edge. The polarity of CE inputs is not configurable (active High).

Register Enable for OREG_ECC Pipeline Stage – OREG_ECC_CE_A, OREG_ECC_CE_B

This register enable pin controls the ECC optional output register. When this register is enabled using the OREG_ECC_A/B attribute, and the corresponding CE input is High, the read data is stored in the register at the rising clock edge. The polarity of CE inputs is not configurable (active High).

Reset – RST_A, RST_B

There are two modes for the reset operation. The synchronous and asynchronous reset modes are controlled by the RST_MODE_A/B attributes. In synchronous reset mode, which is the default, all output flip-flops and latches are synchronously reset to "0". In the asynchronous reset mode, all output flip-flops and latches are reset to "0" without waiting for a CLK edge. This operation does not affect UltraRAM memory cells and does not disturb write operations on either of the ports. The polarity for both signals is configurable (active High by default).

When used in an UltraRAM matrix, the RST input is expected to be asserted (and deasserted) simultaneously at the input of all UltraRAMs in the matrix (in both SYNC and ASYNC reset modes). Consequently, after a RST operation, a new read data is available after N cycles (where N is the read latency of the matrix). However, if the read operation overlaps with the reset operation, the DOUT could change from the reset value to a new read value earlier than the N cycles. This occurs because the read output corresponding to read during or before reset might propagate to the output (since input pipelines IREG_PRE/IREG_CAS are not impacted by the reset). Consequently, this behavior also depends on the REG_CAS locations in the matrix. When the REG_CAS location changes, the DOUT behavior after reset can be different. See Figure 2-11 and Figure 2-12 in Read/Write Waveforms With Reset — With and Without Optional Output Pipeline Registers for the timing diagrams showing an example of this difference in behavior.

IMPORTANT: *When in asynchronous reset mode, the UltraRAM does not have any built-in synchronizers on this input for the deassertion edge. Cascaded UltraRAM use cases need a common synchronizer (typically implemented at an upper level of hierarchy). The input in the fabric must be properly synchronized before it is supplied to the UltraRAM.*

Data-Out Buses – DOUT_A, DOUT_B

Data-out buses reflect the contents of memory cells referenced by the address bus at the last active clock edge during a read operation. During a write operation or no operation, data-out buses are not changed and the data is preserved from the previous cycle. This applies to both single UltraRAM and cascade/matrix configurations. Similarly for a cascaded UltraRAM, the read output at the end of the cascade chain (at exit point) also holds the previous data. The data bus is 72-bits wide with the lower 64 bits used for data and the upper 8 bits used for parity or as regular data outputs.

Read Status Output – RDACCESS_A, RDACCESS_B

The UltraRAM generates a read access status output (RDACCESS_A/B) to indicate that a read operation finished executing, indicating when new data is available at the output. This output has the same latency as the corresponding read data. This output can then be used at the top level to select the correct read data when cascading UltraRAMs across multiple columns. When this output is High, it indicates a read operation has been executed in that UltraRAM or in an UltraRAM below it that is part of the cascade chain. When crossing columns of cascaded UltraRAMs, CLB registers might be required to account for pipelining in the column cascade.

The main purpose of the RDACCESS signal is to support UltraRAMs that are arranged in a matrix fashion. It identifies which UltraRAM block in a matrix configuration is actively reading data in a given clock cycle. The application can then determine the appropriate read data that needs to be propagated to the final output for processing.

Figure 2-4 illustrates a use case where the RDACCESS signal is used to select the correct read output data and control the output data path of a matrix. The circuit holds the data for the inactive outputs. It is important to match the input delay between the matrix entry point (e.g., horizontal pipelining in the fabric for performance reasons) with identical delays on the output side (the delay/pipeline blocks shown in Figure 2-4).

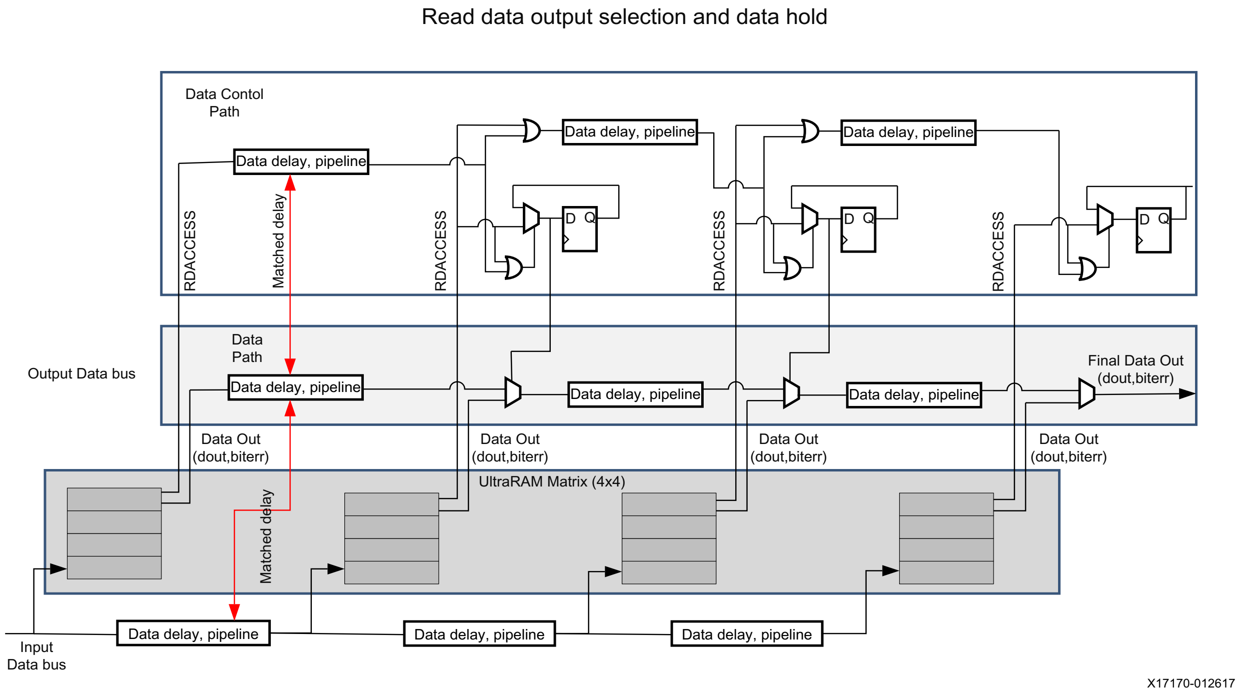


Figure 2-4: **RDACCESS Signal Use Case**

ECC Error Bit Output – SBITERR_A, DBITERR_A, SBITERR_B, DBITERR_B

The ECC error bit outputs are valid when EN_ECC_RD_A/B attributes are set to TRUE. These outputs are asserted when the ECC decoder identifies a single bit error or a double bit error.

Invertible Control Signal Pins

The five control pins CLK, EN_A/B, RST_A/B each have an individual inversion option. EN and RST control signals can be configured as active High or Low, and the clock can be active on a rising or falling edge (active High on a rising edge is the default) without requiring other logic resources.

UltraRAM Attributes

Table 2-5 describes the UltraRAM attributes.

Table 2-5: **UltraRAM Attributes**

Attributes	Values	Default	Type	Description
AUTO_SLEEP_LATENCY	3 to 15	8	DECIMAL	Sets the latency requirement for UltraRAM to sleep mode
AVG_CONS_INACTIVE_CYCLES	10 to 100000	10	DECIMAL	Sets the average consecutive inactive cycles in sleep mode. When in sleep mode, this is defined as the average number of cycles with no read/write operation on either port. Used by the power reporting tools. Set by the user.
BWE_MODE_A	PARITY_INTERLEAVED, PARITY_INDEPENDENT	PARITY_INTERLEAVED	STRING	Port A byte write control for either 1 byte/1 bit 8 bytes/1 byte parity
BWE_MODE_B	PARITY_INTERLEAVED, PARITY_INDEPENDENT	PARITY_INTERLEAVED	STRING	Port B byte write control for either 1 byte/1 bit 8 bytes/1 byte parity
CASCADE_ORDER_A	NONE, FIRST, MIDDLE, LAST	NONE	STRING	Port A position of UltraRAM block in the cascade chain
CASCADE_ORDER_B	NONE, FIRST, MIDDLE, LAST	NONE	STRING	Port B position of UltraRAM block in the cascade chain
EN_AUTO_SLEEP_MODE	FALSE, TRUE	FALSE	STRING	Enables UltraRAM to automatically go into power saving mode
EN_ECC_RD_A	FALSE, TRUE	FALSE	STRING	Port A ECC decoder used for data read or not
EN_ECC_RD_B	FALSE, TRUE	FALSE	STRING	Port B ECC decoder used for data read or not
EN_ECC_WR_A	FALSE, TRUE	FALSE	STRING	Port A ECC encoder used for data write or not
EN_ECC_WR_B	FALSE, TRUE	FALSE	STRING	Port B ECC encoder used for data write or not
IREG_PRE_A	FALSE, TRUE	FALSE	STRING	Inserts port A data, address, and control input pipeline registers
IREG_PRE_B	FALSE, TRUE	FALSE	STRING	Inserts port B data, address, and control input pipeline registers
IS_CLK_INVERTED	`1’b0` to `1’b1`	`1’b0`	BINARY	Optional inverter for CLK
IS_EN_A_INVERTED	`1’b0` to `1’b1`	`1’b0`	BINARY	Port A optional inverter for EN
IS_EN_B_INVERTED	`1’b0` to `1’b1`	`1’b0`	BINARY	Port B optional inverter for EN

Table 2-5: **UltraRAM Attributes *(Cont'd)***

Attributes	Values	Default	Type	Description
IS_RDB_WR_A_INVERTED	`1'b0` to `1'b1`	`1'b0`	BINARY	Port A optional inverter for RDB_WR
IS_RDB_WR_B_INVERTED	`1'b0` to `1'b1`	`1'b0`	BINARY	Port B optional inverter for RDB_WR
IS_RST_A_INVERTED	`1'b0` to `1'b1`	`1'b0`	BINARY	Port A optional inverter for reset input
IS_RST_B_INVERTED	`1'b0` to `1'b1`	`1'b0`	BINARY	Port B optional inverter for reset input
MATRIX_ID	Custom label	NONE	STRING	Custom label (string) to set a matrix ID name used by the power reporting tools to tag all of the UltraRAM blocks that belong to a cascade chain or matrix. Assign different names to each matrix. Single UltraRAM instances do not require a label. Used by the power reporting tools. Set by the user or synthesis tools.
NUM_URAM_IN_MATRIX	1 to 2048	1	DECIMAL	Defines the cascade/matrix size (the number of UltraRAMs in a matrix). Attach to the instances in a particular matrix. For single instances, set to 1. Used by the power reporting tools. Set by the user or synthesis tools.
NUM_UNIQUE_SELF_ADDR_A	1 to 2048	1	DECIMAL	The number of unique SELF_ADDR_A UltraRAM blocks in a cascade chain or matrix. Typically equal to the number of blocks in a cascade chain or matrix. In the broadcast case, the number could be smaller due to common SELF_ADDR_A settings. Used by the power reporting tools. Set by the user or synthesis tools.
NUM_UNIQUE_SELF_ADDR_B	1 to 2048	1	DECIMAL	The number of unique SELF_ADDR_B UltraRAM blocks in a cascade chain or matrix. Typically equal to the number of blocks in a cascade chain or matrix. In the broadcast case, the number is smaller due to common SELF_ADDR_B settings. Used by the power reporting tools. Set by the user or synthesis tools.
OREG_A	FALSE, TRUE	FALSE	STRING	Inserts port A SRAM array output optional pipeline register

Table 2-5: **UltraRAM Attributes *(Cont'd)***

Attributes	Values	Default	Type	Description
OREG_B	FALSE, TRUE	FALSE	STRING	Inserts port B SRAM array output optional pipeline register
OREG_ECC_A	FALSE, TRUE	FALSE	STRING	Inserts port A ECC decoder output optional pipeline register
OREG_ECC_B	FALSE, TRUE	FALSE	STRING	Inserts port B ECC decoder output optional pipeline register
REG_CAS_A	FALSE, TRUE	FALSE	STRING	Inserts port A cascade data input and data output pipeline registers
REG_CAS_B	FALSE, TRUE	FALSE	STRING	Inserts port B cascade data input and data output pipeline registers
RST_MODE_A	SYNC, ASYNC	SYNC	STRING	Port A reset mode
RST_MODE_B	SYNC, ASYNC	SYNC	STRING	Port B reset mode
SELF_ADDR_A	`11'h000 to 11'h7ff`	`11'h000`	HEX	Port A self-address value
SELF_ADDR_B	`11'h000 to 11'h7ff`	`11'h000`	HEX	Port B self-address value
SELF_MASK_A	`11'h000 to 11'h7ff`	`11'h7ff`	HEX	Port A self-address mask
SELF_MASK_B	`11'h000 to 11'h7ff`	`11'h7ff`	HEX	Port B self-address mask
USE_EXT_CE_A	FALSE, TRUE	FALSE	STRING	Port A attribute to allow either internal or external control for the CE pins on all output pipeline registers
USE_EXT_CE_B	FALSE, TRUE	FALSE	STRING	Port B attribute to allow either internal or external control for the CE pins on all output pipeline registers

Note: The URAM288_BASE primitive does not have any of the cascade attributes.

Auto Sleep Latency – AUTO_SLEEP_LATENCY

The auto sleep mode automatically utilizes the function provided by the SLEEP pin in an automated manner. To determine when to go to sleep and when to wake up, the UltraRAM looks ahead in terms of RAM access.

When the EN_AUTO_SLEEP_MODE attribute is set to TRUE, the AUTO_SLEEP_LATENCY attribute defines the number of clock cycles the enable and global address inputs EN_A, EN_B, ADDR_A[22:12], and ADDR_B[22:12] have to arrive prior to other inputs. This lookahead information is used to decide when the UltraRAM can go to sleep. The EN_A/B and ADDR_A/B[22:12] are internally delayed to implement this feature. The number of clock cycles required for the enable and global address signals to arrive early is set with the AUTO_SLEEP_LATENCY attribute, which can take values between 3 – 15. Therefore, for the UltraRAM to go into sleep mode, a minimum number of consecutive inactive clock cycles is required that is defined by the value of the AUTO_SLEEP_LATENCY attribute.

The number of sleep cycles achieved is calculated with this formula:

- If the number of consecutive inactive cycles is < AUTO_SLEEP_LATENCY, then the number of sleep cycles = 0.
- If the number of consecutive inactive cycles is >= AUTO_SLEEP_LATENCY, then the number of consecutive sleep cycles = the number of consecutive inactive cycles – 3.
- An inactive cycle is defined as a cycle where there is no RD/WR operation from either port.

For example, to obtain sleep cycles in any five or more consecutive cycles of inactivity, set the AUTO_SLEEP_LATENCY to five.

Once in auto sleep mode, the output of the OREG retains the old value for only one clock cycle. The data can be reset to "0" in the very next cycle or later depending on when the UltraRAM enters sleep mode. If there are no other pipeline registers enabled after OREG, the design must use the register output read data only during the last valid clock cycle. If there are other consecutive pipeline registers, then those pipeline registers will hold the last read data.

The auto sleep mode is most effective for large chain sizes or any chain with very little activity. While there is a default for this attribute, the application determines the effective power savings of this feature based on activity, latency, and other needs of the application. If AUTO_SLEEP_LATENCY is too low, the UltraRAM goes into sleep and wake-up too often, which can cause more power to be consumed. If it is too high, then the maximum of amount of power might not be saved.

Byte Write Enable Mode – BWE_MODE_[A|B]

This attribute determines the data and parity usage of the byte write enable (BWE_[A/B]) inputs. Either the one data byte\one parity bit mode is selected (PARITY_INTERLEAVED) or the eight data bytes/1 parity byte mode is selected (PARITY_INDEPENDENT). See Byte-Wide Write Enable – BWE_A, BWE_B for more information.

Cascade Chain Order – CASCADE_ORDER_[A|B]

This attribute indicates if an UltraRAM is part of a cascade and the location of the UltraRAM in the cascade chain. The values are NONE (default), FIRST, MIDDLE, and LAST. All UltraRAMs that are first in each column should be set to FIRST and all UltraRAMs that are last in each column should be set to LAST. All UltraRAMs in between must be set to MIDDLE. See Cascade User Attributes for more information.

AVG_CONS_INACTIVE_CYCLES, MATRIX_ID, NUM_URAM_IN_MATRIX, and NUM_UNIQUE_SELF_ADDR_A|B Attributes

These attributes have no functional impact on the design and are used for power estimation and power reporting. The Vivado tools set these attributes automatically when the UltraRAMs are inferred through synthesis.

✓ **RECOMMENDED:** When manually instantiated, the attributes should be set to reflect the actual usage for accurate power calculations. If not set, the power estimates will be pessimistic.

Note: In the broadcast use case, the SELF_MASK setting can affect the NUM_UNIQUE_SELF attribute. While the ADDR_A/B[22:12] inputs can be different on UltraRAM instances in a matrix, the SELF_MASK setting might result in a match for multiple instances in a matrix. See example shown in Figure 2-8.

Set Enable Auto Sleep Mode – EN_AUTO_SLEEP_MODE

This mode enables auto sleep mode for automatic power savings and overwrites the application's control of sleep mode. When set to TRUE, the user-controlled sleep input is disabled. Instead, UltraRAM internally puts itself in and out of sleep mode to achieve automatic power savings. The criteria for UltraRAM to go in and out of sleep mode is a measure of inactive clock cycles determined by the AUTO_SLEEP_LATENCY attribute.

Note: The use of USE_EXT_CE_A/B is not permitted when in AUTO_SLEEP mode. Both attributes are not permitted to be true at the same time.

Enable ECC Write – EN_ECC_WR_[A/B]

This attribute determines if the ECC encoder (write) is enabled or not.

Enable ECC Read – EN_ECC_RD_[A/B]

This attribute determines if the ECC decoder (read) is enabled or not.

Optional Input Register Stage – IREG_PRE_[A|B]

This attribute determines if EN/RDB_WR/BWE/ADDR/DIN/INJECT_SBITERR/INJECT_DBITERR UltraRAM inputs have their respective input pipeline registers enabled or not. IREG_PRE and REG_CAS are mutually exclusive except as noted under Optional Cascade Register Stage – REG_CAS_[A|B]. See Figure 2-5.

Optional Output Register Stage – OREG_[A|B]

This attribute determines if the SRAM array output has a pipeline stage enabled or not.

Optional ECC Output Register Stage – OREG_ECC_[A|B]

This attribute determines if the ECC error and data outputs of the ECC decode logic have the pipeline registers enabled or not.

Optional Cascade Register Stage – REG_CAS_[A|B]

Determines if both cascade data/controls/address inputs and outputs have their pipeline registers (IREG_CAS and OREG_CAS) enabled or not. These pipeline stages play a critical role in determining the maximum frequency of the UltraRAM. In cascade mode, these registers should be used in each block or every few blocks depending on the maximum frequency requirement. IREG_PRE and REG_CAS are mutually exclusive except when the CASCADE_ORDER attribute is set to MIDDLE or LAST, the IREG_PRE register can still be used in the cascade case for the error injection inputs INJECT_S/DBITERR. For all other inputs REG_CAS must be used. See Figure 2-5.

Reset Mode – RST_MODE_[A|B]

Determines if RST_[A/B] is a synchronous or asynchronous input to CLK.

Self Address – SELF_ADDR_[A|B]

This attribute determines the self-address of the UltraRAM and must be a unique value for each UltraRAM in the cascade chain. This determines the address of each UltraRAM in the cascade chain. It is an 11-bit value and can have any value from `11'h000` to `11'h7ff`. See Cascade User Attributes for more information.

Self Mask Value – SELF_MASK_[A|B]

This attribute determines how many bits in the cascaded address (SELF_ADDR) are used for the comparison (address decoding) with the ADDR input to determine if the input address matches the UltraRAM in the cascade chain. The number of address bits is determined by the total address space in the cascaded UltraRAMs. The MSB bits corresponding to the unused address bits should be set to "1". See Cascade User Attributes for more information.

External CE Usage – USE_EXT_CE_[A|B]

This attribute enables the use of external CE inputs to control all the output pipeline stages in non-cascade mode. By default, the design uses the internally generated CEs to control all the pipeline stages. This does not apply to the OREG_CAS registers enables. In cascade mode, the OREG_CAS register enables are automatically controlled by the UltraRAM. Using the RDACCESS output signal is not allowed when external CE mode is enabled.

IMPORTANT: *In cascade mode, USE_EXT_CE cannot be used and should be set to false. Consequently, in cascade mode, the external CE inputs (OREG_CE and OREG_ECC_CE) cannot be used. This attribute is only supported when CASCADE_ORDER=NONE.*

Dual Port SRAM Array Operations

The dual-port 288 Kb UltraRAM consists of a 288 Kb storage area and two independent access ports, A and B. Both ports share a single clock input.

In each clock cycle, each port can perform either a read or a write operation independent of the other port. Any combination of read/write is allowed on any of the two ports. The read and write operations are always synchronous to the clock. The operation of port A is always executed first followed by the operation of port B within the same clock cycle. Consequently, data access collision is not possible when both ports access the same address locations. Each port has its own address, data in, data out, enable, and write enable.

- If both ports are executing read and write for the same address, the behavior is defined as (see Table 2-6):
    - If port A is writing, port B is reading, then port B reads new data.
    - If port A is reading, port B is writing, then port A reads the old data.

- If port A and B are writing, then port B write overwrites the port A write. At the end of the clock cycle, the memory stores port B write data.

Table 2-6: **UltraRAM Port Access**

UltraRAM Port Access	Port A	Port B	Data Output
1 Read/1 Write	Read	Write	Old data
1 Read/1 Write	Write	Read	New data
1 Read/1 Read and Write	Read	Read/Write	Old data
1 Read/1 Read and Write	Read/Write	Read	New data
2 Read/2 Write	Read or Write	Write or Read	Depends on port A/B read/write combination

Read Operation

In default mode with no optional pipeline registers enabled, the read operation uses one clock edge. The read address is registered on the read port, and the stored data is loaded into the output latch after the SRAM access time. When using additional optional input/output registers, the read operation needs extra cycles depending on how many pipeline registers are used. The read data is held on the output until the next valid read operation or until a reset operation changes the output.

Write Operation

A write operation is a single clock-edge operation, unless the optional input register is enabled. The write address is registered on the write port, and the data input is stored in memory. The read output holds the previous value during a write operation, unless the reset input is asserted.

Optional Input Registers

The optional data, address, and control input registers (IREG_PRE registers) improve design performance by eliminating the routing delay from the CLB flip-flops for pipelined operation. Optional input registers (IREG_CAS registers) for cascading data, address and control are available. Either the data input or the cascade input registers can be used at any given time for an UltraRAM block depending on its configuration (input cascaded or not). Both the input and output cascade registers are enabled via the REG_CAS attribute simultaneously and cannot be turned on or off individually.

Optional Output Registers

The optional output registers improve design performance by eliminating the routing delay to the CLB flip-flops for pipelined operation. The first optional output register (OREG stage) is immediately after the SRAM array read operation. Additional optional output registers after the ECC decode logic (OREG_ECC stage) and cascade logic (OREG_CAS register) are available. By default, the design uses internally generated CE to control all the pipeline stages for power saving. However, an external CE port can be used by setting the USE_EXT_CE_A/B attribute. When the external CE is enabled, an independent clock enable input port is provided for these output registers. If the output data registers are disabled via their CE port, they hold their value independent of the input register operation.

RESET operation

An UltraRAM RST operation simultaneously resets the read data/status/ECC error outputs and all corresponding optional output/cascade pipeline registers. The reset function can be synchronous (the default) or asynchronous depending on the RST_MODE attribute setting. The reset operation has priority over any read operation and any of the CE inputs. After deasserting RST, the reset value is valid until a new read data value flows through the pipeline.

When in asynchronous reset mode, the UltraRAM does not have any built-in synchronizers on this input for deassertion. Therefore, a logic-based synchronizer might be required for the RST input.

IMPORTANT: *If the design utilizes cascade, then a common synchronizer should be used for all the UltraRAM RSTs in the chain.*

Byte Write Enable Function

The byte write enable feature allows a single byte of the input data to be written to the SRAM array. There are nine bits of write enable inputs for each port A and port B. There are two modes of operation that are selected by the BWE_MODE_[A/B] attribute. In PARITY_INTERLEAVED mode, each write enable bit enables eight data bits plus one parity bit. So each byte has a corresponding single parity bit. In the PARITY_INDEPENDENT mode, each write enable bit (BWE[7:0] enables the writing of eight data bits (one byte). The BWE bit number nine (BWE[8]) enable bit controls the one byte of eight parity bits. The byte write inputs are ignored during a read operation.

IMPORTANT: *If the ECC feature is used, all byte write enable bits must be set to "1" for proper operation of the ECC encoder/decoder.*

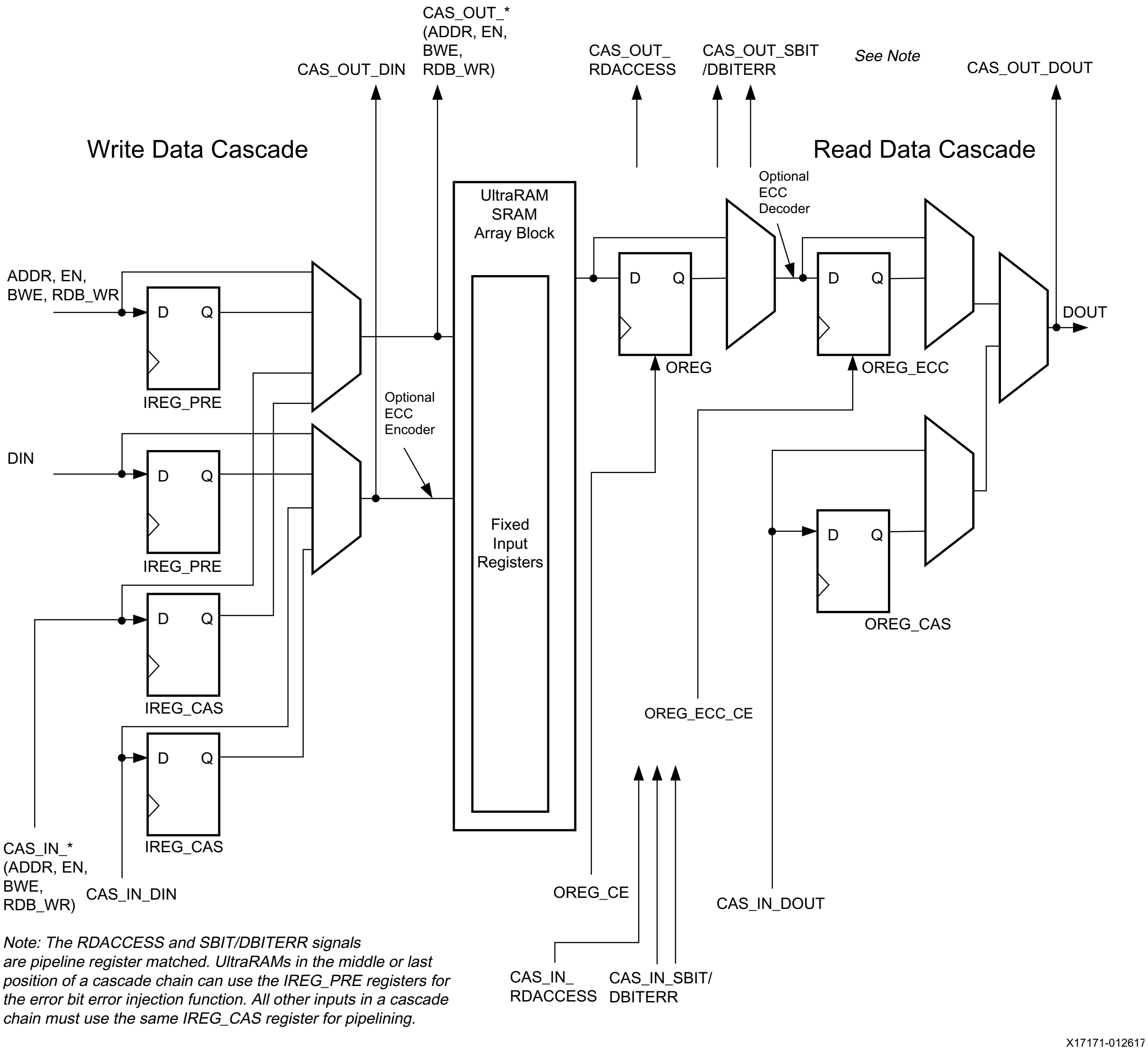
Cascading UltraRAM and Matrix Configurations

One of the advanced, built-in features of UltraRAM is the capability to build deeper RAMs by directly cascading UltraRAM blocks in a single column through a dedicated direct interconnect. Ports for data in, data out, ECC error, address, enables, read/write select, and a write mask attribute facilitate cascading (see Figure 2-5).

Cascading is supported in only one direction and is always in a bottom-up fashion. UltraRAM blocks can be cascade unlimited in a single column within an SLR without limitation and have built-in connections. Cascade pipeline registers (IREG_CAS and OREG_CAS stages are enabled by the REG_CAS attribute) are available options in each UltraRAM. These registers can be enabled as needed depending on the maximum frequency and latency requirements of the design. Cascading from one clock region to the next clock region above can require additional pipeline registers on both the input and output side of the cascade chain to avoid potential setup time violations.

Cascading UltraRAMs across different columns can be achieved using logic and routing resources. The UltraRAM generates a read access status output RDACCESS_A/B to indicate that a read operation was executed. This output has the same latency as the corresponding read data and can be used to determine the correct read data when cascading using multiple columns.

If there is no read operation being performed, the read output at the end of the cascade chain (at the block exit point) will hold the previous data.



Note: The RDACCESS and SBIT/DBITERR signals are pipeline register matched. UltraRAMs in the middle or last position of a cascade chain can use the IREG_PRE registers for the error bit error injection function. All other inputs in a cascade chain must use the same IREG_CAS register for pipelining.

Figure 2-5: **UltraRAM Cascade Block Diagram (One Port shown)**

Cascade User Attributes

- CASCADE_ORDER_A/B determines the UltraRAM block cascade order.
    - NONE (Default) – UltraRAM is not in cascade mode.
    - FIRST – UltraRAM is the first instance in a cascade chain in each column of the chain.
    - MIDDLE – UltraRAM is a the middle instance in a cascade chain.
    - LAST – UltraRAM is the last instance in a cascade chain in each column of the chain.
- SELF_MASK_A/B[22:12] determines the number of UltraRAM blocks in the cascade chain and therefore which of the ADDR_A/B[22:12] bits are used.
    - `11'h7ff` (Default) – Not in cascade mode. ADDR_A/B[22:12] inputs are masked.
    - `11'h7fe` – 2 UltraRAMs are cascaded. ADDR_A/B[22:13] inputs are masked.
    - `11'h7fc` – 3-4 UltraRAMs are cascaded. ADDR_A/B[22:14] inputs are masked
    - `11'h7f8` – 5-8 UltraRAMs are cascaded. ADDR_A/B[22:15] inputs are masked.
    - `11'h7f0` – 9-16 UltraRAMs are cascaded. ADDR_A/B[22:16] inputs are masked.
    - `11'h7e0` – 17-32 UltraRAMs are cascaded. ADDR_A/B[22:17] inputs are masked.
    - `11'h7c0` – 31-64 UltraRAMs are cascaded. ADDR_A/B[22:18] inputs are masked.
    - `11'h780` – 65-128 UltraRAMs are cascaded. ADDR_A/B[22:19] inputs are masked.
    - `11'h700` – 129-256 UltraRAMs are cascaded. ADDR_A/B[22:20] inputs are masked.
    - `11'h600` – 257-512 UltraRAMs are cascaded. ADDR_A/B[22:21] inputs are masked.
    - `11'h400` – 513-1024 UltraRAMs are cascaded. ADDR_A/B[22] input is masked.
    - `11'h000` – 1025-2048 (1036) UltraRAMs are cascaded. None of the address inputs are masked.
- SELF_ADDR_A/B[22:12]

  This attribute is used in cascade mode and must be set depending on which address bit in the UltraRAM cascade chain addresses the particular block to which it is attached. A particular UltraRAM block in the cascade chain is accessed when the self-address bit is set after masking with the address bits with the SELF_MASK_A/B attribute that matches the used ADDR_A/B address bits. The default is `11'h0`. (Figure 2-6).

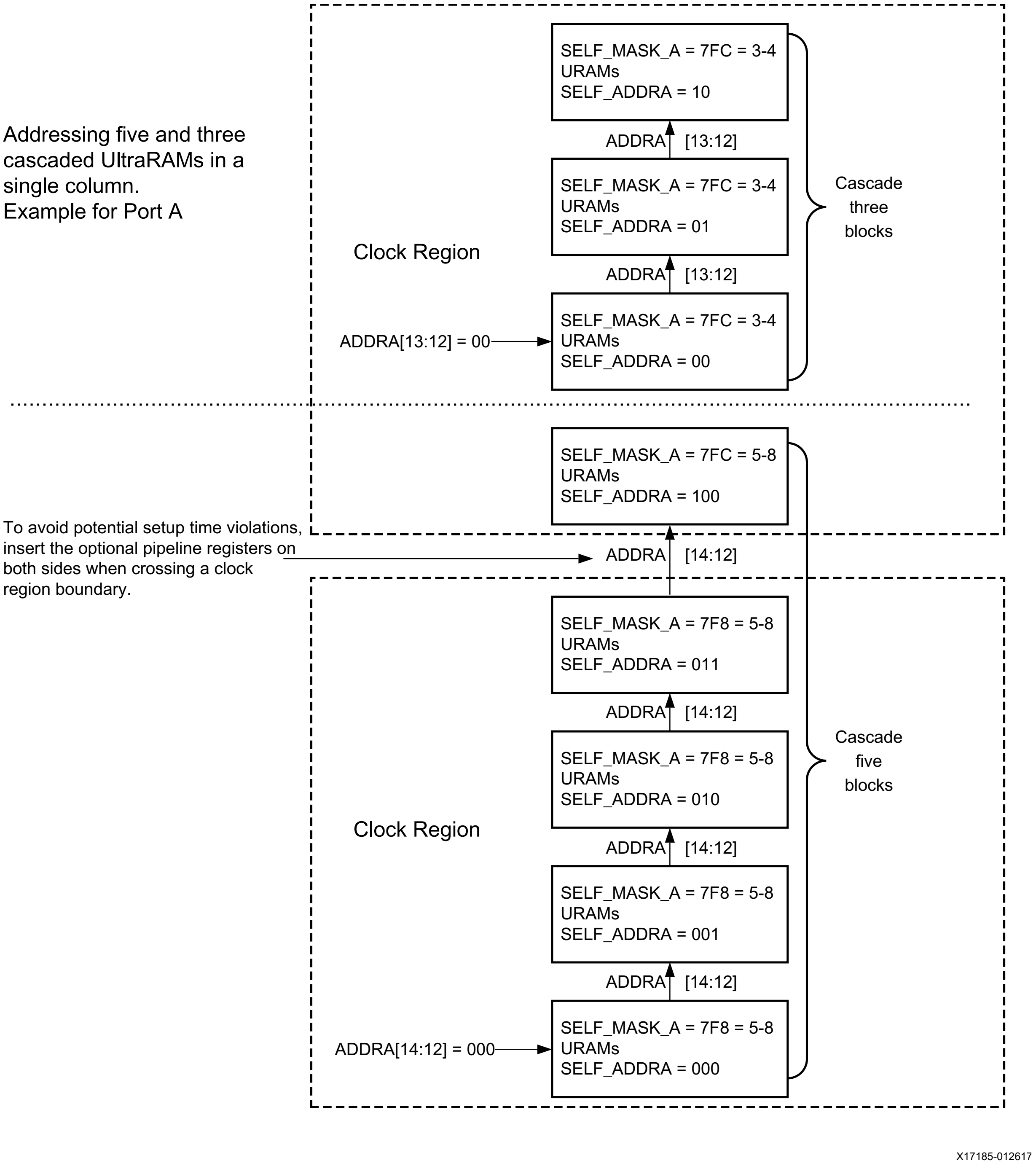


Figure 2-6: **Cascade Chain Examples**

Building a Matrix From Cascaded UltraRAMs

Applications can build a deep logical SRAM with multiple UltraRAMs. These UltraRAM instances form a matrix such that address, control signals, and input data arrive at the UltraRAM matrix at the bottom left and output data appear at the top right. Figure 2-7 illustrates the concept behind the UltraRAM matrix. In this X by Y (row x column) matrix, each matrix element is a single UltraRAM block cascaded vertically. To read/write from/to a matrix, address, control signals and input data (if write) enter the UltraRAM matrix in row 1. A write operation writes the input data to the addressed UltraRAM block at location row, column (R,C) and the word in it. Similarly, for a read operation, the output data reaches the output bus on top of the columns (always) by selecting an UltraRAM R,C and location in it. Figure 2-7 illustrates a 4x4 UltraRAM matrix.

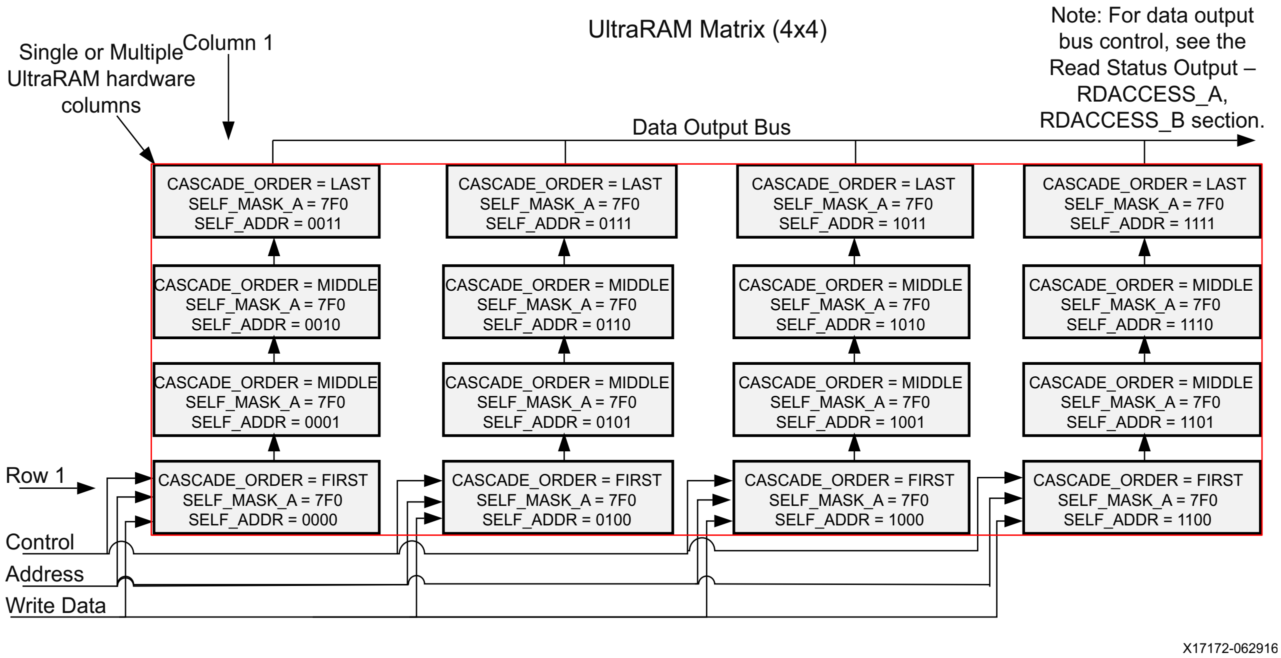


Figure 2-7: **4x4 UltraRAM Matrix**

With the individual address encoding scheme for each UltraRAM, each block individually determines if it should have data. SELF_ADDRESS and SELF_MASK allow for non-unique addresses in a matrix, which allows for the broadcasting of data to multiple UltraRAMs simultaneously in the same cycle. The SELF_ADDRESS can be used as a one-hot encoded address (even partially), and the SELF_MASK determines which address bits are important and which address bits can be ignored (one-cold). Consequently, a global address applied to all UltraRAMs in a matrix can apply to a set of predetermined UltraRAMs. Figure 2-8 illustrates a multicast write of data to two, more, or all UltraRAMs in row 1. In this example, the lower four block address bits are ORed via the SELF_ADDR settings for the block that must simultaneously receive data, while the SELF_MASK ignores address bits not to be decoded for a block. In this use case, the UltraRAM can only be used in 1 read/1 write mode.

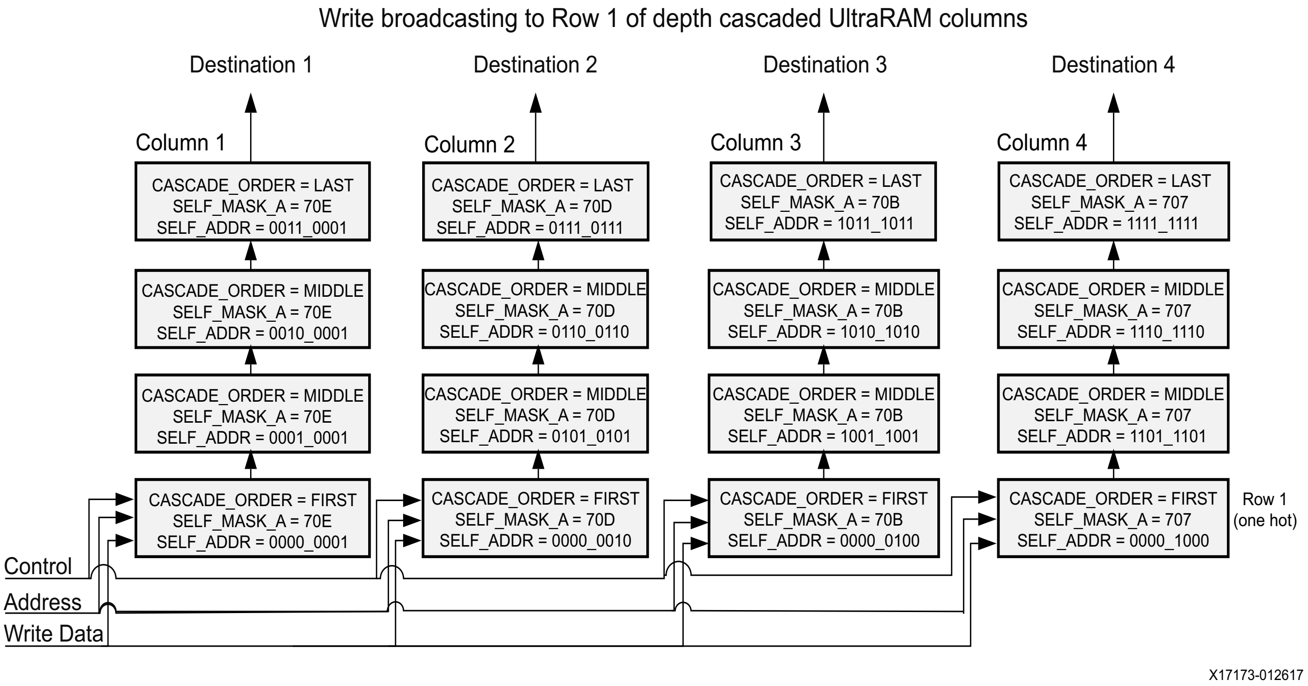


Figure 2-8: **Write Broadcasting to Row 1 of Depth Cascaded UltraRAM Columns**

Address Bit Decoding

- Row 1 Multicast

  ADDR15:12 = `0011` multicast to columns 1+2

  ADDR15:12 = `0101` multicast to columns 1+3

  ADDR15:12 = `0110` multicast to columns 2+3

  ADDR15:12 = `0111` multicast to columns 1+2+3

  ADDR15:12 = `1101` multicast to columns 1+2+4

  etc.

  ADDR15:12 = `1111` multicast to columns 1+2+3+4

- Row/Column Addr

  ADDR19:16 = `0001` – `0011` rows 2,3,4, column 1

  ADDR19:16 = `0101` – `0111` rows 2,3,4, column 2

  ADDR19:16 = `1001` – `1011` rows 2,3,4, column 3

  ADDR19:16 = `1101` – `1111` rows 2,3,4, column 4

The UltraScale+ device is supported by the Vivado Design Suite, which includes several code templates to help target the available silicon resources. There are three methods of RTL design entry to use the UltraRAM memories:

- Use the Xilinx Parameterized Macros (XPM) starting with the 2016.1 Vivado tools release.
- Infer an RTL memory and use the ram_style attribute set to “ultra”.
- Instantiate the device primitive.

Examples for these methods are in the Vivado language templates accessible from the main Vivado tools menu by selecting **Tools** -> **Language Templates**.

Note: XPM is the most effective method to obtain expected results with a high degree of customization. Details are provided in the *UltraScale Architecture Libraries Guide* (UG974) [Ref 3].

Built-in Error Detection and Correction

Each UltraRAM 4K x 72 RAM has built in optional Hamming code error correction for each port. The upper (MSB) 8 bits of the 72-bit data bus are used for parity when ECC is turned on. The ECC operation is transparent to the user. All byte write enable BWE_B[8:0] bits must be set to "1" (HIGH) in ECC mode for proper operation. ECC operations for port A and port B are identical.

Eight protection bits (ECCPARITY) are generated during each write operation and stored with the 64-bit data into the memory. These ECCPARITY bits are used during each read operation to correct any single-bit error, or to detect (but not correct) any double-bit error. ECC data bits and status/control bits are synchronous to the CLK.

During each read operation, 72 bits of data (64 bits of data and 8 bits of parity) are read from the memory and presented to the ECC decoder. The ECC decoder generates two status outputs (SBITERR_A/B and DBITERR_A/B) that are used to indicate the three possible read results: no error, single-bit error corrected, or double-bit error detected. In the standard ECC mode, the read operation does not correct the error in the memory array. It only presents corrected data on DOUT. To improve FMAX, optional registers are available for data output (DO), SBITERR, and DBITERR.

If RST_A/B is asserted, all output registers are reset to “0”. Therefore, the SBITERR and DBITERR status signals are also RESET to “0” (LOW) indicating that the data output does not have a single bit or a double bit error.

The UltraRAM can also inject errors in either of the ports. ECC mode can inject single bit errors or double bit errors in any or all words. When INJECT_SBITERR is asserted during a write cycle, a single bit error is injected internally in the memory corresponding to DIN[30]. When INJECT_DBITERR is asserted during a write cycle, a double bit error is injected internally in the memory corresponding to DIN[30] and DIN[62]. If both INJECT_SBITERR and INJECT_DBITERR are asserted during a write cycle, a double bit error is injected at the same location as INJECT_DBITERR.

This capability is available in all ECC modes.

ECC Modes

In the standard ECC mode (EN_ECC_RD = TRUE and EN_ECC_WR = TRUE), both encoder and decoder are enabled. During write, 64-bit data and 8-bit ECC generated parity are stored in the array. The external parity input bits are ignored. During read, the 72-bit decoded data and parity are read out.

The most common use case is to enable both the ECC encoder and decoder in a port. However, the encoder and decoder can be enabled separately. To enable only the encoder, the data must be sent through the DI port, the ECCPARITY bits are written into the RAM, and the decoder is disabled. To use only the decoder, the encoder is disabled, the data is written into the RAM, and the corrected data and status bits are read out of the UltraRAM.

ECC Modes of Operation

There are three types of ECC operation:

- Full ECC mode
- ECC DECODE only mode
- ECC ENCODE only mode

The standard ECC mode uses both the encoder and decoder.

Standard ECC

Set by Attributes

EN_ECC_RD = TRUE

EN_ECC_WR = TRUE

Standard ECC Write

The ECC encoder uses DIN[63:0] to generate the corresponding 8 bits of ECC parity, appends it to the 64 data bits, and then writes into the memory. Because ECC parity is generated internally, the DIN[71:64] pins are not used.

The IREG_PRE optional pipeline stage is available before the ECC encode logic for all input pins. This stage can be enabled as needed to meet the maximum frequency requirement.

Standard ECC Read

During read operation, the 72-bit memory content, consisting of 64 bits of data and 8 bits of parity is read out from an address location and decoded internally. If there is no error, the original data and parity are output at DOUT[71:0]. If there is a single-bit error in either the data or the parity, the error is corrected, and SBITERR is High. If there is a double-bit error in the data and parity, the error is not corrected. The original data and parity is output and DBITERR is High.

The OREG optional pipeline stage is available just before the ECC decode logic and the OREG_ECC optional pipeline stage is available just after the ECC decode logic for all the DOUT and error bit outputs. Either or both of these stages can be enabled depending on the maximum frequency and latency requirements of the design.

ECC Encode Only

Set by Attributes

EN_ECC_RD = FALSE

EN_ECC_WR = TRUE

ECC Encode-Only Write

The ECC encoder uses DIN[63:0] to generate the corresponding 8 bits of ECC parity, appends it to the 64 data bits, and then writes into the memory. Because ECC parity is generated internally, the DIN[71:64] pins are not used.

The IREG_PRE optional pipeline stage is available before the ECC encode logic for all input pins. This stage can be enabled as needed to meet the Fmax requirement.

ECC Encode-Only Read

In ECC encode-only mode, read is identical to normal block RAM read. 64-bit data appears at DOUT[63:0] and 8-bit parity appears at DOUT[71:64]. Single-bit error correction does not occur, and the error flags SBITERR and DBITERR are never asserted.

ECC Decode Only

Set by Attributes

EN_ECC_RD = TRUE

EN_ECC_WR = FALSE

In ECC decode-only, only the ECC decoder is enabled. The ECC encoder is disabled. Decode-only mode is used to inject single-bit or double-bit errors to test the functionality of the ECC decoder. The ECC parity bits must be externally supplied using the DIN[71:64] pins.

Using ECC Decode Only to Inject Single-Bit Error

- DIN[71:0] with a single-bit error injected is written into the memory array.
- When the memory location is read out, the data is corrected as needed.
- SBITERR lines up with the corresponding DOUT data.

The ECC decoder also corrects single-bit errors in parity bits.

Using the ECC Decode-Only to Inject Double-Bit Error

- DIN[71:0] with double-bit error injected is written into the memory array.
- When the memory location is accessed, the corrupted data is read out and a double-bit error is detected.
- DBITERR lines up with the corresponding DOUT data.

The ECC decoder also detects when double-bit errors in parity bits occurs, and when a single-bit error in the data bits and a single-bit error in the corresponding parity bits occur.

UltraRAM Timing Diagrams

This section describes and illustrates the timing associated with the UltraRAM block. The timing diagrams show the behavior for read/write/reset operations in matrix and single block configuration, as well as the effects of different pipelining options and the clock enable function. Detailed timing diagrams of the sleep and auto sleep modes are shown with various pipelines and latency configurations.

Read/Write Waveforms With and Without Optional Pipeline Registers

Figure 2-9 and Figure 2-10 show the read/write waveforms with and without optional pipeline registers.

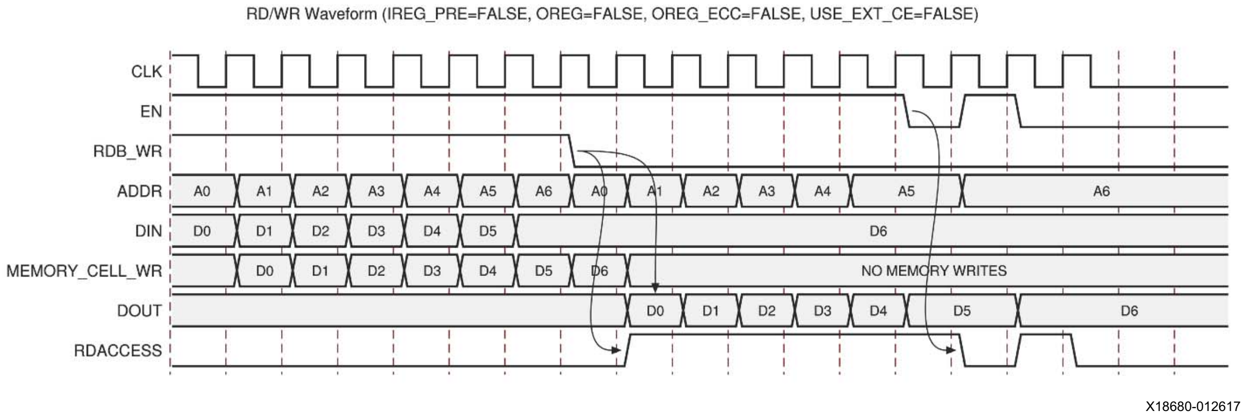


Figure 2-9: **Read/Write with Attributes IREG_PRE_A/B=FALSE, OREG_A/B=FALSE, OREG_ECC_A/B=FALSE, USE_EXT_CE_A/B=FALSE**

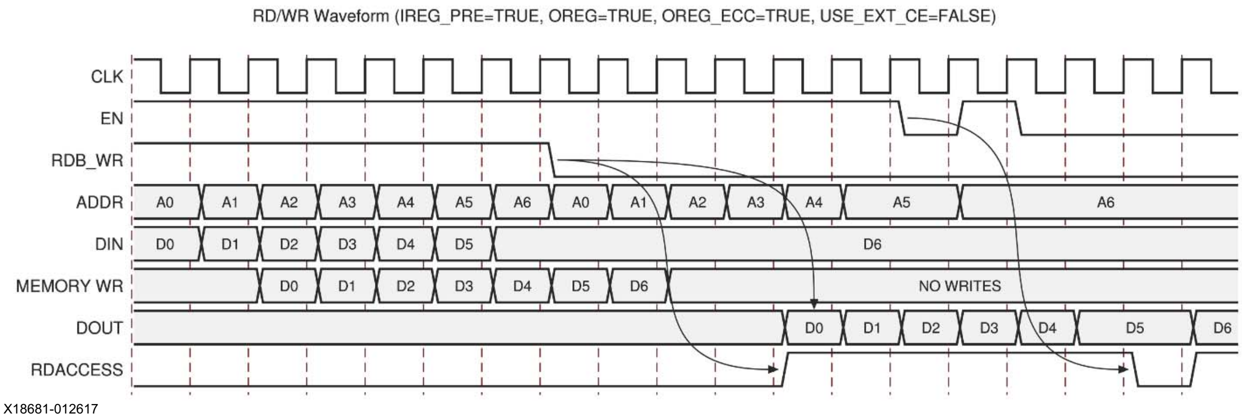


Figure 2-10: **Reset/Read/Write with Attributes IREG_PRE_A/B=TRUE, OREG_A/B=TRUE, OREG_ECC_A/B=TRUE, USE_EXT_CE_A/B=FALSE**

Read/Write Waveforms With Reset — With and Without Optional Output Pipeline Registers

Figure 2-11, Figure 2-12, and Figure 2-13 show the read/write waveforms with reset and with and without optional pipeline registers.

Note: Reset has priority over the read operation and reset has no impact on any write operation.

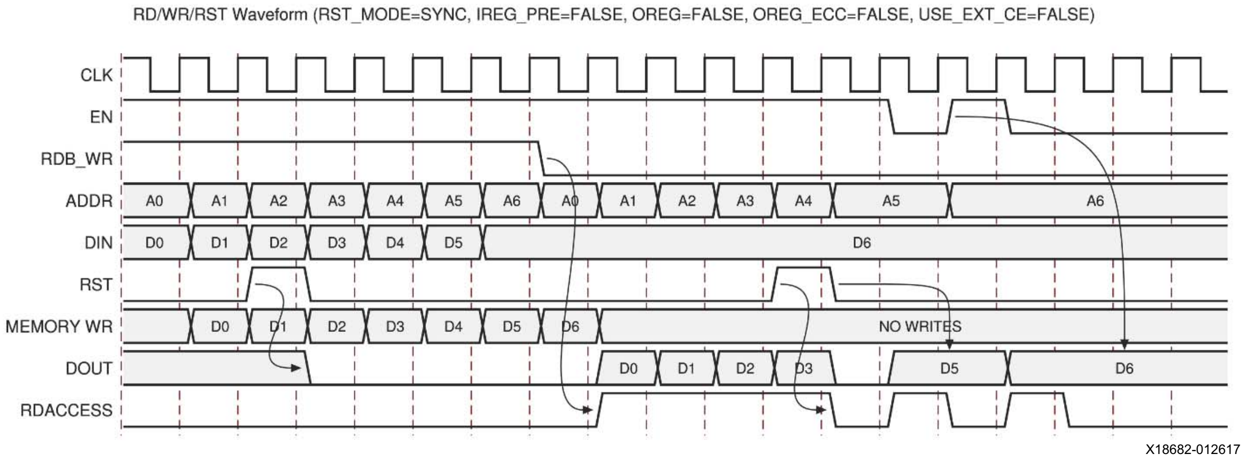


Figure 2-11: **Reset/Read/Write with Attributes RST_MODE=SYNC, IREG_PRE_A/B=FALSE, OREG_A/B=FALSE, OREG_ECC_A/B=FALSE, USE_EXT_CE_A/B=FALSE**

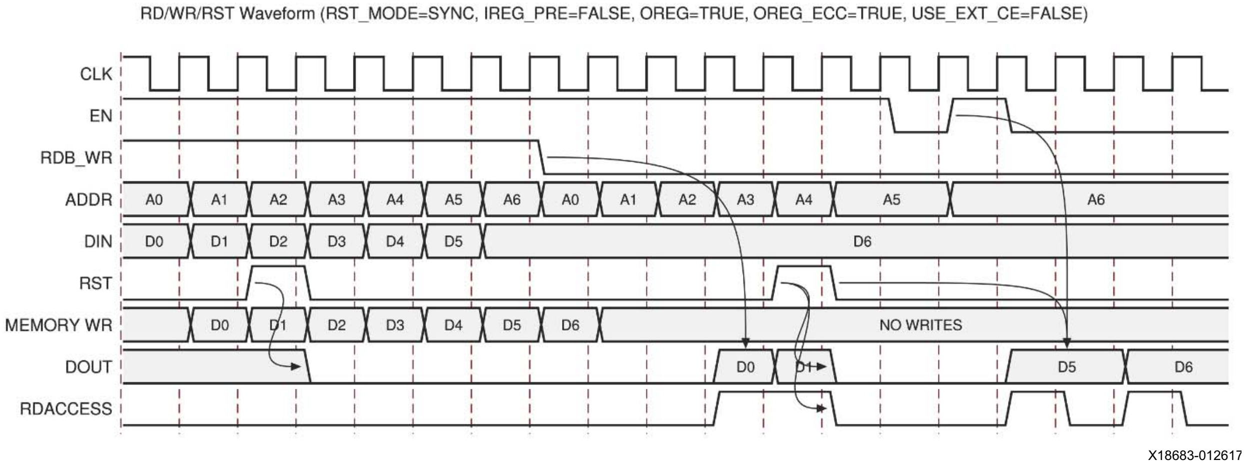


Figure 2-12: **Reset/Read/Write with Attributes RST_MODE=SYNC, IREG_PRE_A/B=FALSE, OREG_A/B=TRUE, OREG_ECC_A/B=TRUE, USE_EXT_CE_A/B=FALSE**

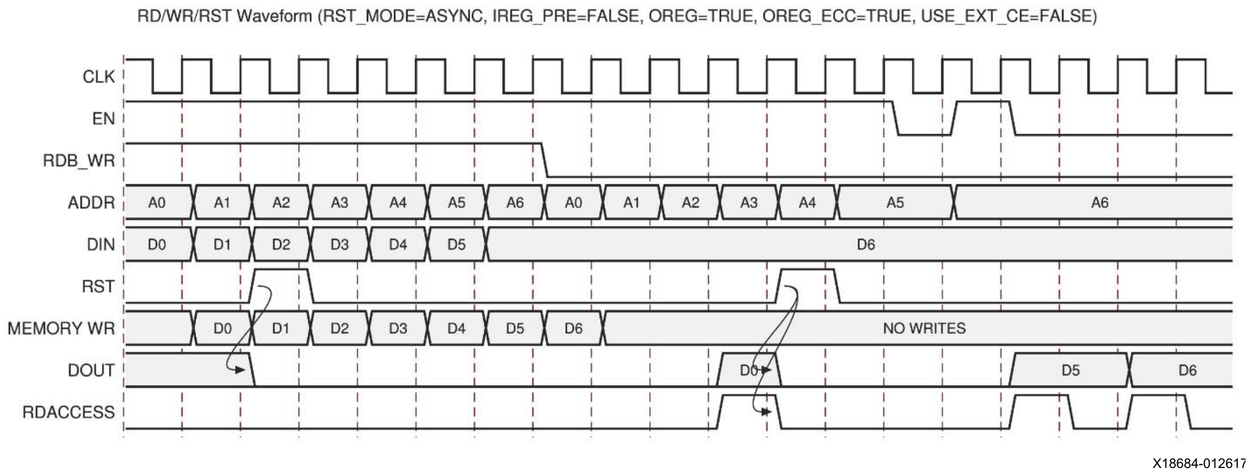


Figure 2-13: **Read/Write with Attributes RST_MODE=ASYNC, IREG_PRE_A/B=FALSE, OREG_A/B=TRUE, OREG_ECC_A/B=TRUE, USE_EXT_CE_A/B=FALSE**

Read/Write Waveforms With External CE

Figure 2-14 and Figure 2-15 show read/write waveforms with external CE. Using external CE is allowed for single UltraRAM blocks only, and is not supported for cascaded UltraRAMs. RDACCESS is not supported when USE_EXT_CE=TRUE.

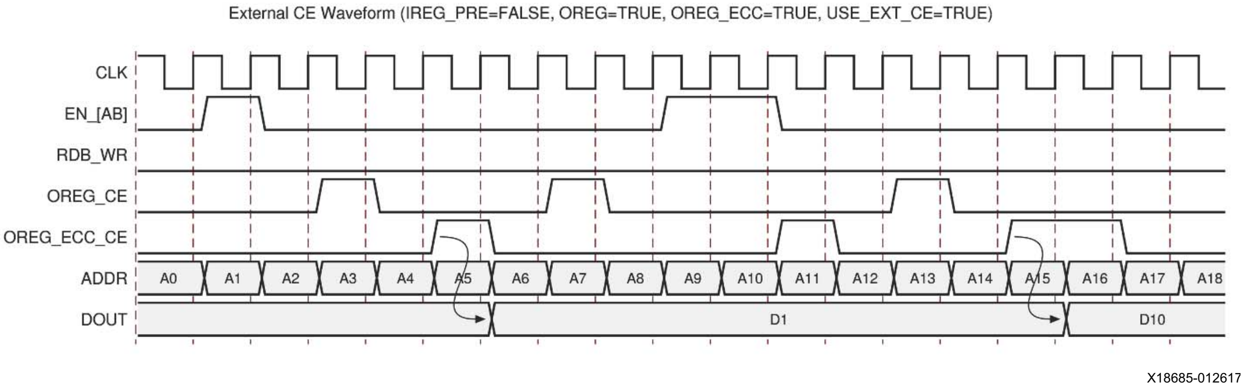


Figure 2-14: **External CE with Attributes IREG_PRE_A/B=FALSE, OREG_A/B=TRUE, OREG_ECC_A/B=TRUE, USE_EXT_CE_A/B=TRUE**

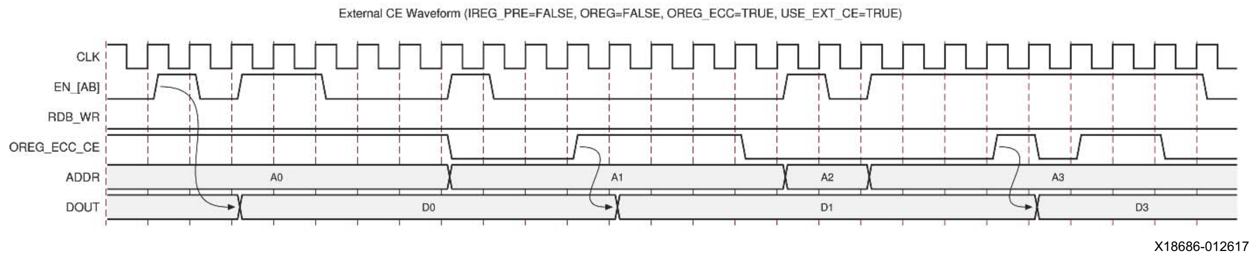


Figure 2-15: **External CE with Attributes IREG_PRE_A/B=FALSE, OREG_A/B=FALSE, OREG_ECC_A/B=TRUE, USE_EXT_CE_A/B=TRUE**

Read From Matrix Waveforms With Reset

The waveforms in Figure 2-16 and Figure 2-17 are for a three UltraRAM cascade case and assume OREG and OREG_ECC is set to TRUE for all three UltraRAMs in the cascade chain:

- RST input is simultaneously asserted or deasserted at input of all three UltraRAMs.
- All other inputs are driven at input of first UltraRAM. All outputs exit from last UltraRAM.

Note: The DOUT output behavior after reset might be different depending on the location of REG_CAS, as shown in Figure 2-16 and Figure 2-17.

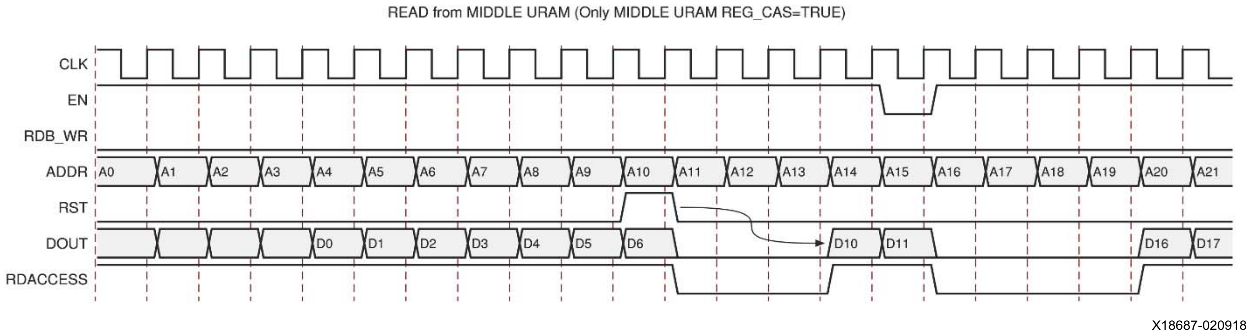


Figure 2-16: **Reading from Middle UltraRAM — Middle UltraRAM with REG_CAS=TRUE and Last UltraRAM with REG_CAS=TRUE**

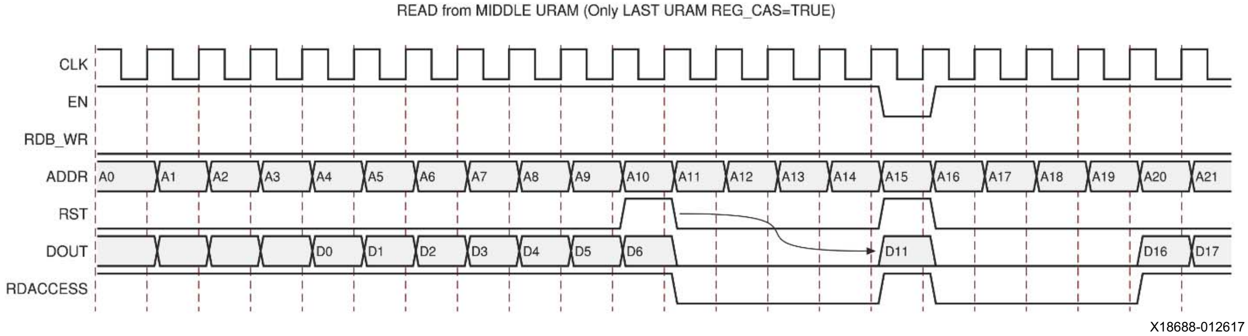


Figure 2-17: **Reading from Middle UltraRAM — Last UltraRAM with REG_CAS=TRUE and Middle UltraRAM With REG_CAS=FALSE**

Sleep Waveforms

Sleep has priority over EN. Any attempted memory writes are ignored and the previous memory content is preserved. Any attempted memory reads are also ignored. See Figure 2-18 to Figure 2-22.

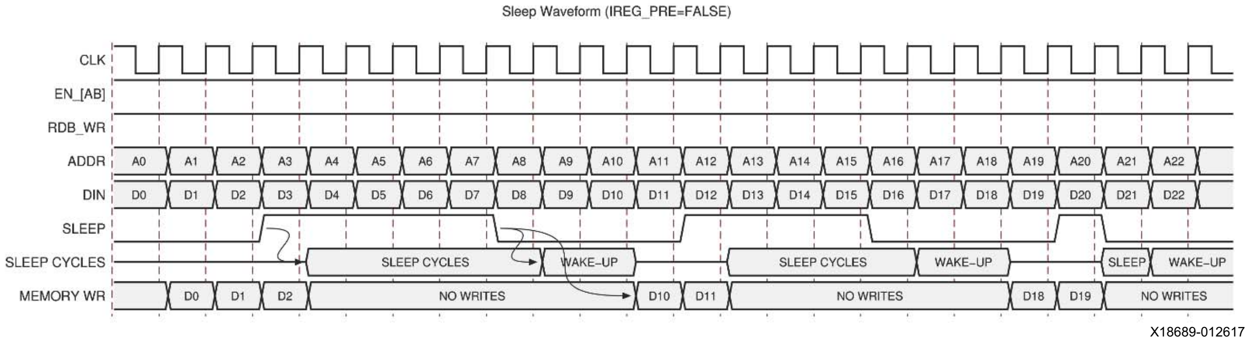


Figure 2-18: **Sleep Mode for Write Operations With Attribute IREG_PRE_A/B=FALSE**

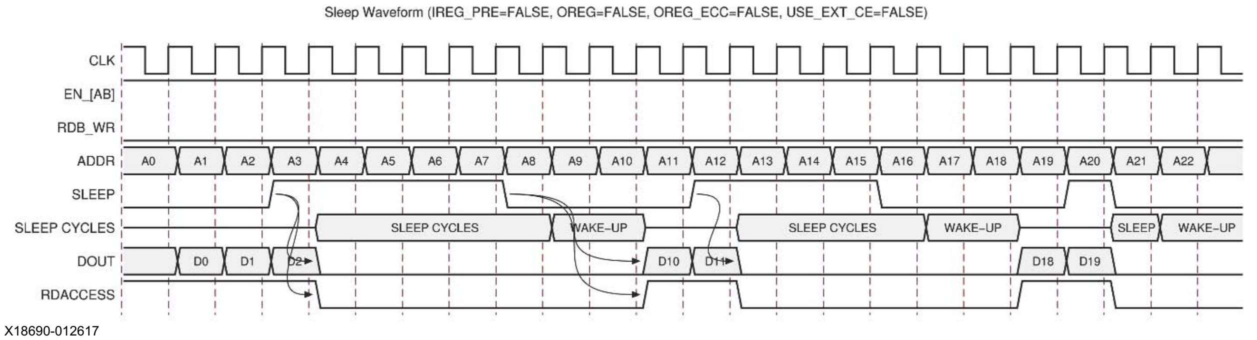


Figure 2-19: **Sleep Mode for Read Operations With Attributes IREG_PRE_A/B=FALSE, OREG_A/B= FALSE, OREG_ECC_A/B= FALSE, USE_EXT_CE_A/B= FALSE, DOUT is Forced to "0"**

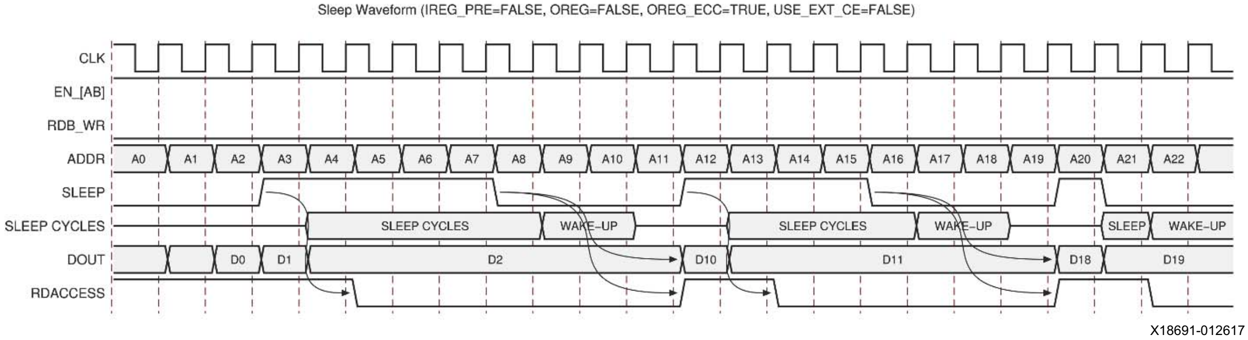


Figure 2-20: **Sleep Mode for Read operations With Attributes IREG_PRE_A/B=FALSE, OREG_A/B= FALSE, OREG_ECC_A/B= TRUE, USE_EXT_CE_A/B= FALSE, DOUT is Preserved With Previous Read Data**

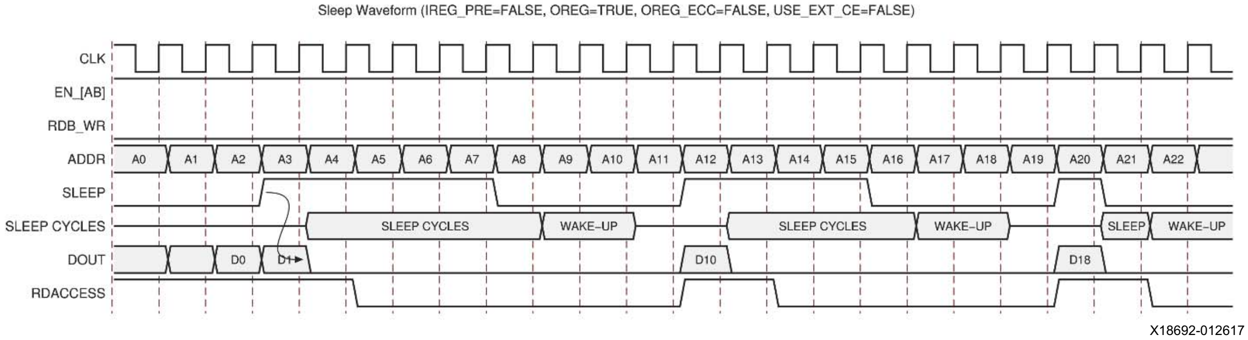


Figure 2-21: **Sleep Mode For Read Operations With Attributes IREG_PRE_A/B=FALSE, OREG_A/B= TRUE, OREG_ECC_A/B= FALSE, USE_EXT_CE_A/B= FALSE**

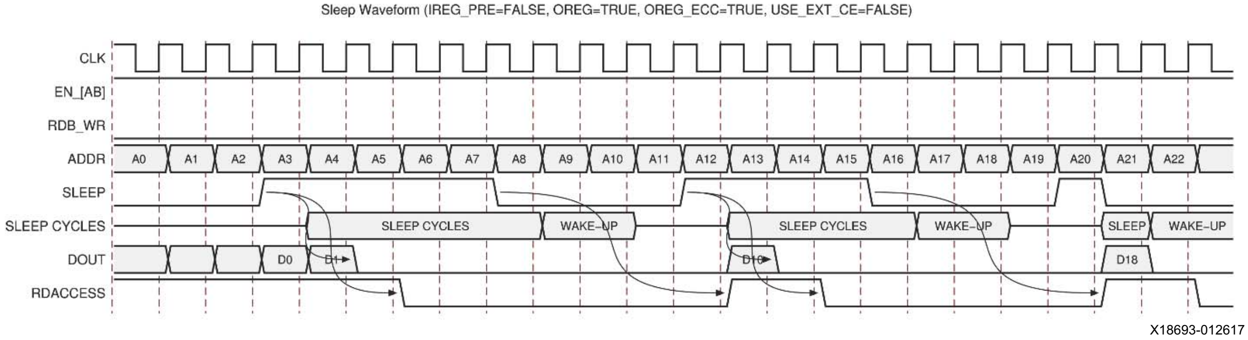


Figure 2-22: **Sleep Mode for Read Operations With Attributes IREG_PRE_A/B=FALSE, OREG_A/B= TRUE, OREG_ECC_A/B= TRUE, USE_EXT_CE_A/B= FALSE**

Note: Read outputs D2, D11, and D19 are lost due to sleep cycle immediately after read (when OREG=TRUE). Consequently, DOUT is driven to “0”. However, the RDACCESS is asserted since the internal read memory access is not blocked.

When OREG=TRUE, the read corresponding to address A2 internally is not blocked (since sleep is still Low in this cycle). However, since sleep goes High in the next cycle, the OREG has lost the data, and the output of OREG becomes “0”. In this case, even though OREG_ECC=TRUE, since the read itself was not blocked, the OREG_ECC becomes “0” because new read data is expected. This occurs if there is a read operation followed immediately by sleep with OREG=TRUE (irrespective of whether OREG_ECC is TRUE or FALSE).

Auto Sleep Waveforms

To determine when the UltraRAM can activate sleep and wake-up in the auto sleep mode, look-ahead information is required. The byte write enable, read/write, data, and lower address inputs must be delayed with respect to the enable and higher address inputs. These inputs are delayed by pipeline stages equal to the AUTO_SLEEP_LATENCY setting, which can be between 3 and 15. FIFOs or linear shift registers can be used to accomplish this in the FPGA fabric. Other signals, such as INJECT and CE inputs, must also be pipeline aligned if used. See Auto Sleep Latency – AUTO_SLEEP_LATENCY for more information.

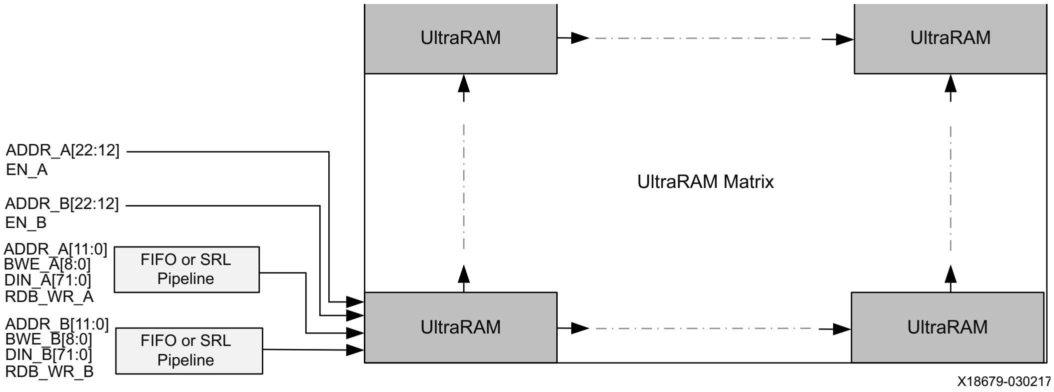


Figure 2-23: **Typical Fabric Implementation for Using Auto Sleep Mode**

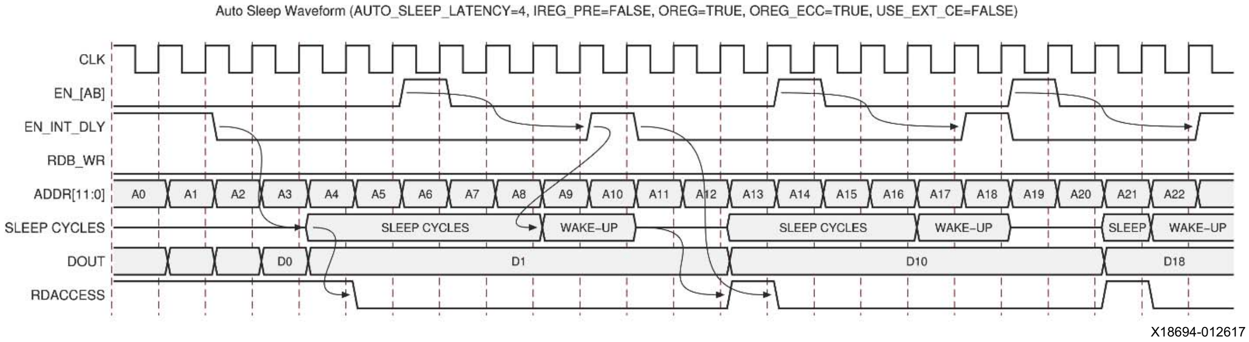


Figure 2-24: **Auto Sleep Mode for Read Operations With Attributes AUTO_SLEEP_LATENCY=4, IREG_PRE_A/B=FALSE, OREG_A/B= TRUE, OREG_ECC_A/B= TRUE, USE_EXT_CE_A/B= FALSE**

EN_INT_DLY is delayed by AUTO_SLEEP_LATENCY from input EN to show the pipeline alignment with other inputs. DOUT holds previous read data during sleep cycles when OREG_ECC=TRUE.

After EN_INT_DLY is deasserted, an extra idle cycle is inserted before the sleep cycle because of the OREG stage. If the OREG pipeline is set to TRUE, an extra cycle is needed to ensure the read data has propagated to the output before the UltraRAM enters sleep. Consequently, there is a wait of two cycles after EN_INT_DLY deasserts before starting the sleep cycles. Additionally, the sleep cycle is also a function of AUTO_SLEEP_LATENCY, which dictates the number of idle cycles needed to go to sleep (the delay from EN to EN_INT_DLY).

Auto sleep wake-up always occurs one cycle before EN_INT_DLY goes High to guarantee sufficient wakeup time before a next read or write cycle. Wake-up is only a function of EN_INT_DLY_A/B (and EN by extension) rising and no other inputs.

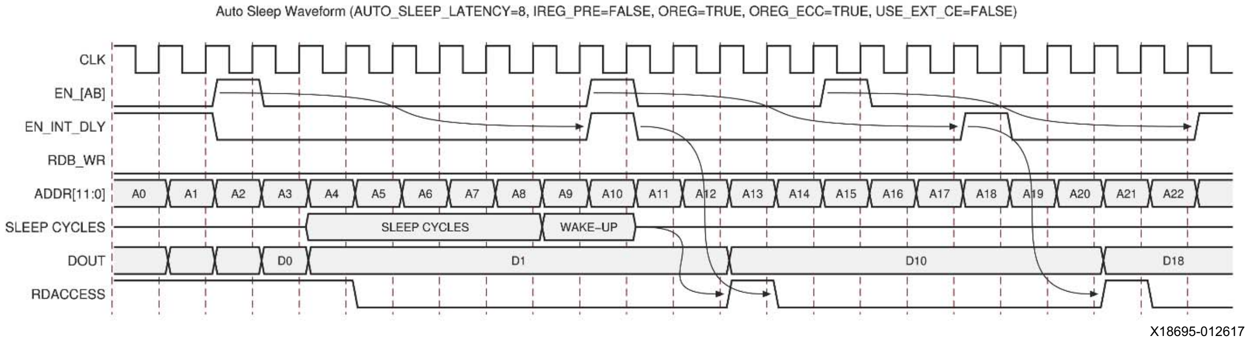


Figure 2-25: **Auto Sleep Mode for Read Operations With Attributes AUTO_SLEEP_LATENCY=8, IREG_PRE_A/B=FALSE, OREG_A/B= TRUE, OREG_ECC_A/B= TRUE, USE_EXT_CE_A/B= FALSE**

EN_INT_DLY is delayed by AUTO_SLEEP_LATENCY from input EN to show the pipeline alignment with other inputs. DOUT holds previous read data during sleep cycles when OREG_ECC=TRUE.

The AUTO_SLEEP_LATENCY dictates the number of idle cycles needed for the UltraRAM to go to sleep. In Figure 2-25, it is set to eight, which means at least eight cycles are needed between EN_INT_DLY deasserting and the next assertion of EN_INT_DLY to see any sleep cycles. In the later cycles (after sleep/wake-up), there is not enough idle time and the UltraRAM does not go to sleep again.

See the attribute description in Auto Sleep Latency – AUTO_SLEEP_LATENCY for the number of expected sleep cycles.

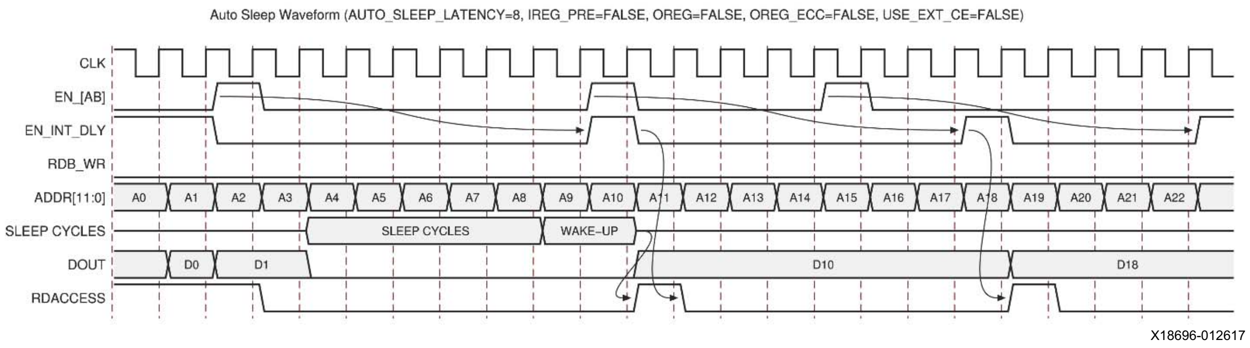


Figure 2-26: **Auto Sleep Mode for Read Operations With Attributes AUTO_SLEEP_LATENCY=8, IREG_PRE_A/B=FALSE, OREG_A/B= FALSE, OREG_ECC_A/B= FALSE, USE_EXT_CE_A/B= FALSE**

EN_INT_DLY is delayed by AUTO_SLEEP_LATENCY from input EN to show the pipeline alignment with other inputs. DOUT holds previous read data when there are no sleep cycles, but is driven to “0” during sleep cycles when OREG_ECC=FALSE.

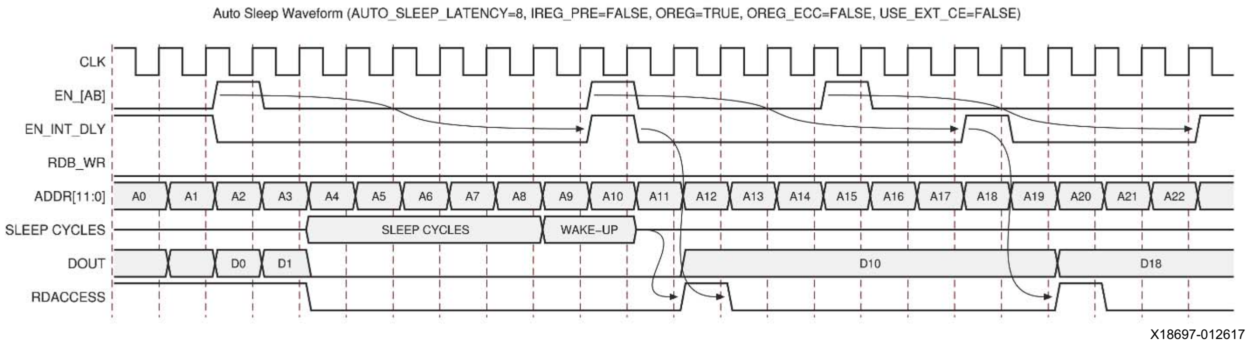


Figure 2-27: **Auto Sleep Mode for Read Operations With Attributes AUTO_SLEEP_LATENCY=8, IREG_PRE_A/B=FALSE, OREG_A/B= TRUE, OREG_ECC_A/B= FALSE, USE_EXT_CE_A/B= FALSE**

EN_INT_DLY is delayed by AUTO_SLEEP_LATENCY from input EN to show the pipeline alignment with other inputs. DOUT holds previous read data when there are no sleep cycles, but is driven to “0” during sleep cycles when OREG_ECC=FALSE.

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.

References

These documents provide supplemental material useful with this guide:

1. UltraScale and UltraScale+ device data sheets:
   - *UltraScale Architecture and Products Overview* (DS890)
   - *Zynq UltraScale+ MPSoC Overview* (DS891)
   - *Zynq UltraScale+ MPSoC Data Sheet: DC and AC Switching Characteristics* (DS925)
   - *Kintex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS892)
   - *Kintex UltraScale+ FPGAs Data Sheet: DC and AC Switching Characteristics* (DS922)
   - *Virtex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS893)
   - *Virtex UltraScale+ FPGAs Data Sheet: DC and AC Switching Characteristics* (DS923)
2. *7 Series FPGAs Memory Resources User Guide* (UG473)
3. *UltraScale Architecture Libraries Guide* (UG974)
4. *Vivado Design Suite User Guide: Embedded Processor Hardware Design* (UG898)

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.