LogiCORE IP XADC Wizard v2.4

Product Guide

PG091 December 18, 2012

XILINX®

Table of Contents

Introduction

The LogiCORE™ IP Xilinx® Analog-to-Digital Converter (XADC) Wizard provides related information, including recommended design experience, additional resources, technical support, and submitting feedback to Xilinx.

Features

- Analog to digital conversion
- FPGA temperature and voltage monitoring
- Generate alarms based on user set parameters
- Optional AXI4-Lite interface based on the AXI4 specification
- Simple user interface
- Easy configuration of various modes and parameters
- Simple interface for channel selection and configuration
- Ability to select/deselect alarm outputs and set alarm limits
- Calculates all the parameters and register values

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	Virtex®-7, Kintex™-7, Artix™-7
Supported User Interfaces	AXI4-Lite
Resources	See Table 2-1.
Provided with Core	
Design Files	Verilog and VHDL
Example Design	Verilog and VHDL
Test Bench	Verilog and VHDL
Constraints File	Vivado™: XDC
Simulation Model	Not Provided
Supported S/W Driver(2)	Standalone
Tested Design Flows(3)	
Design Entry	Vivado Design Suite v2012.4(4)
Simulation	Vivado Simulator, Mentor Graphics ModelSim, Cadence Incisive Enterprise Simulator (IES), Synopsys VCS and VCS MX
Synthesis	Vivado Synthesis Synopsys Synplify PRO
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For a complete listing of supported devices, see the release notes for this core.
2. Standalone driver details can be found in the EDK or SDK directory (<*install_directory*>/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from //wiki.xilinx.com.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.
4. Supported only 7 series devices.

Overview

The XADC Wizard generates Verilog or VHDL Register Transfer Level (RTL) source code to configure the XADC primitive in Xilinx 7 series FPGAs. An example design and simulation test bench demonstrate how to integrate the core into user designs.

The XADC Wizard is included with the ISE and Vivado Design Suite software. For information about system requirements and installation, see Chapter 2, Installing the Wizard. Version 2.4 of the XADC Wizard is a Vivado-only update. The information in this version of the product guide covers use of the wizard in Vivado Design Suite only.

The top-level block diagram for the LogiCORE IP XADC core is shown in Figure 1-1.

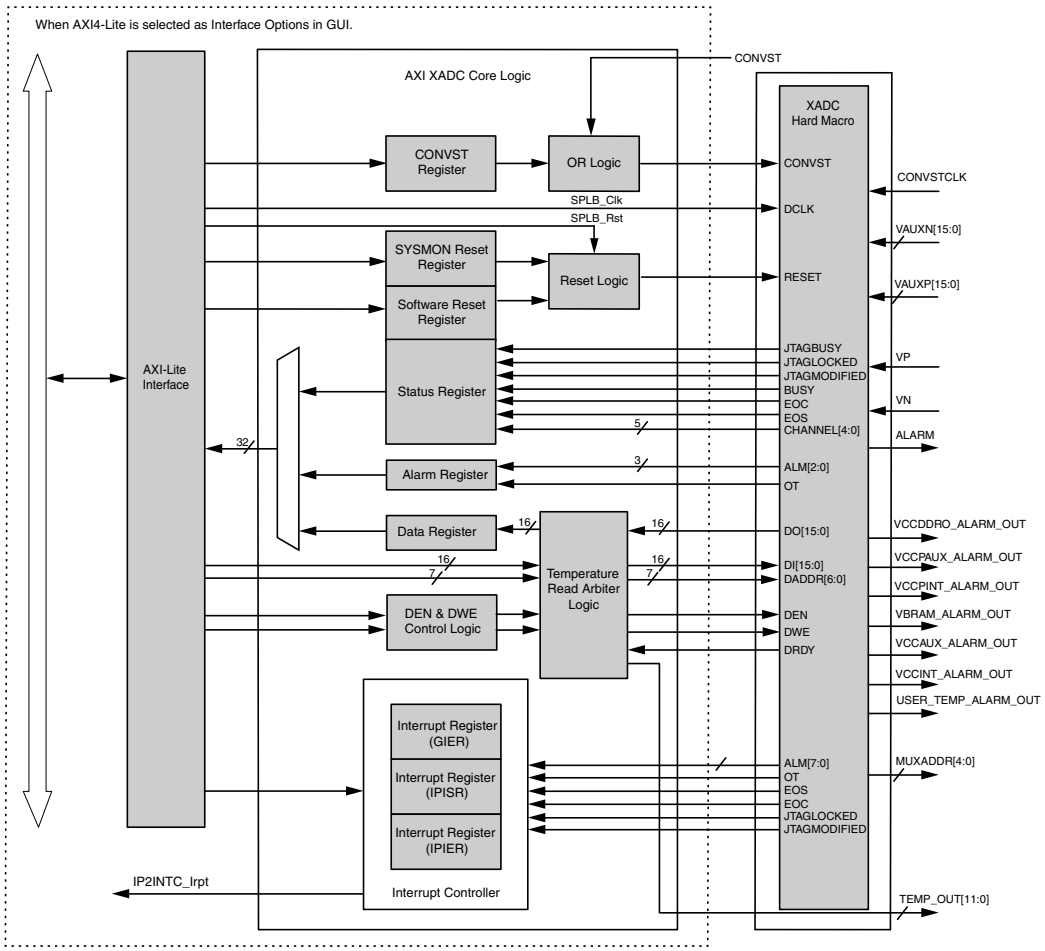


Figure 1-1: **XADC IP Core Block Diagram**

Operating System Requirements

For a list of system requirements, see the Xilinx Design Tools: Release Notes Guide.

Feature Summary

- Analog to digital conversion
- FPGA temperature and voltage monitoring
- Generate alarms based on user set parameters
- Optional AXI4-Lite interface based on the AXI4 specification
- Simple user interface
- Easy configuration of various modes and parameters
- Simple interface for channel selection and configuration
- Ability to select/deselect alarm outputs and set alarm limits
- Calculates all the parameters and register values

Applications

The XADC Wizard is ideally suited for high-volume applications such as Multi-Function Printers (MFP), digital SLR cameras, Motor Control, Power Conversion, Touch/Gesture-based HMI, anti-tamper security, and system management.

Before You Begin

Before installing the Wizard, you must have a MySupport account. If you already have an account and have the software installed, go to Installing the Wizard; otherwise,

- Click **Login** at the top of the Xilinx home page then follow the onscreen instructions to create a MySupport account.

Installing the Wizard

Version 2.4 of the XADC Wizard is included with the Vivado Design Suite 2012.4, and is accessed from the Vivado IP catalog within that toolset. v2012.4 of the Vivado Design Suite can be downloaded from the Xilinx Download Center, www.xilinx.com/support/download/index.htm.

For details, see the Vivado Design Suite Release Notes and Installation Guide.

Verifying Your Installation

Use the following procedure to verify that you have successfully installed the XADC Wizard in the Vivado tools.

1. Start Vivado.
2. After creating a new 7 series family project or opening an existing one, the IP catalog appears at the right side of the window, as shown in Figure 1-2.

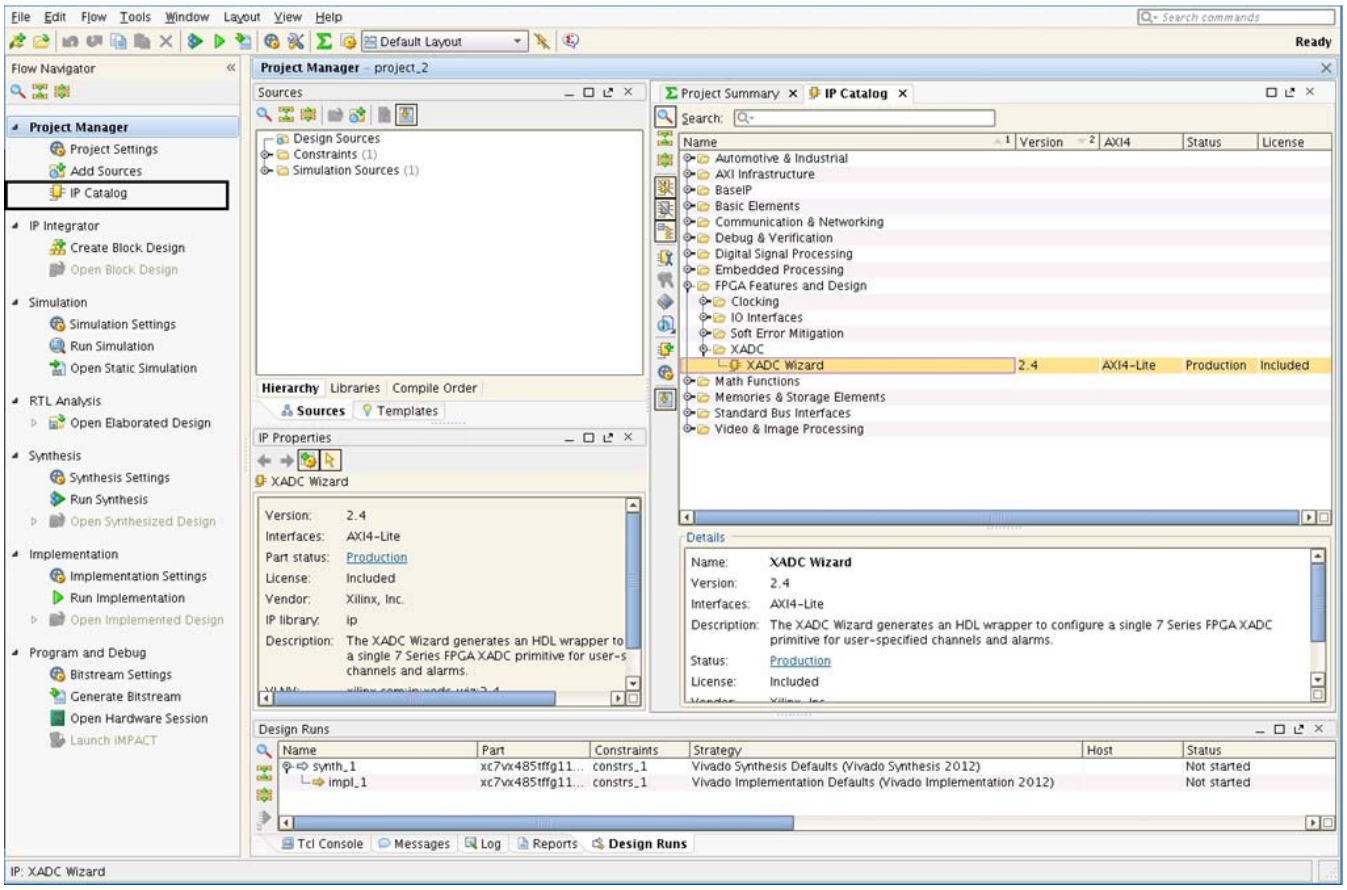


Figure 1-2: **Vivado IP Catalog**

3. Determine if the installation was successful by verifying that XADC Wizard appears at the following location in the catalog list:
   `/FPGA Features and Design/XADC`.

Licensing and Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx Vivado Design Suite tool under the terms of the Xilinx End User License Agreement. The Wizard can be generated by the Xilinx Vivado IP catalog, which is a standard component of the Xilinx Vivado Design Suite. This version of the core can be generated using the Vivado IP catalog 2012.4. For more information, visit the Architecture Wizards web page.

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

This chapter describes the GUI and follows the same flow required to set up the XADC primitive using v2.4 of the wizard. Tool tips are available in the GUI for most features; place your mouse over the relevant text, and additional information is provided in a popup dialog.

Functional Overview

The XADC Wizard is an interactive graphical user interface (GUI) that instantiates a XADC block configured to your requirements. Using the wizard, users can explicitly configure the XADC to operate in the desired mode. The GUI allows you to select the channels, enable alarms, and set the alarm limits. You can select AXI4-Lite, DRP, or None interface using the Interface selection.

XADC Functional Features

Major functional XADC features can be used to determine an appropriate mode of operation. These features include:

- Analog to digital conversion
- FPGA temperature and voltage monitoring
- Generate alarms based on user set parameters

Standards

The LogiCORE™ IP XADC Wizard core contains an AXI4-Lite interface, which is based on the AMBA® AXI4 specification.

Performance

If the user enables averaging of the channel, then data capture rate gets reduced depending on the averaging selected. Choose the appropriate value to match your requirement. Analog Input noise from the supply or board can deviate the expected 12-bit digital output.

Maximum Frequencies

The maximum S_AXI_ACLK/DCLK clock frequency supported is 250 MHz.

Resource Utilization

Because the XADC Wizard IP core can be used with other design modules in the FPGA, the utilization numbers reported in this section are estimates only.

Table 2-1 details the XADC Wizard IP Core resource utilization when AXI4-Lite is selected as the interface. These are measured with Xilinx 7 series FPGAs as the target device for interrupt logic enabled or disabled.

Table 2-1: **7 Series FPGAs Resource Estimates**

Parameter Values (Other parameters at default values)	Device Resources		Performance
C_HAS_TEMP_BUS	Slice Flip-Flops	LUTs	F_{MAX} (MHz)
0	236	211	100
1	427	473	100

When DRP interface is selected, XADC Wizard uses XADC primitive only. Therefore, no LUTs are used as resource.

Port Descriptions

Table 2-2 describes the input and output ports provided from the XADC Wizard. Availability of ports is controlled by user-selected parameters. For example, when Dynamic Reconfiguration is selected, only ports associated with Dynamic Reconfiguration are exposed. Any port that is not exposed is tied off or connected to a signal labeled as “unused” in the delivered source code.

Table 2-2: **XADC I/O Signals**

Port	Direction	Description
DI_IN[15:0]	Input	Input data bus for the dynamic reconfiguration port (DRP).
DO_OUT[15:0]	Output	Output data bus for the dynamic reconfiguration port.
DADDR_IN[6:0]	Input	Address bus for the dynamic reconfiguration port.
DEN_IN	Input	Enable signal for the dynamic reconfiguration port.
DWE_IN	Input	Write enable for the dynamic reconfiguration port.
DCLK_IN	Input	Clock input for the dynamic reconfiguration port.
DRDY_OUT	Output	Data ready signal for the dynamic reconfiguration port.
RESET_IN	Input	Reset signal for the XADC control logic and maximum/minimum registers.
CONVST_IN	Input	Convert start input. This input is used to control the sampling instant on the ADC input and is only used in Event Mode Timing (see Event-Driven Sampling in the *7 Series FPGAs XADC User Guide* [Ref 2].
CONVSTCLK_IN	Input	Convert start input. This input is connected to a global clock input on the interconnect. Like CONVST, this input is used to control the sampling instant on the ADC inputs and is only used in Event Mode Timing.
VP_IN VN_IN	Input	One dedicated analog-input pair. The XADC has one pair of dedicated analog-input pins that provide a differential analog input.
VAUXP15[15:0] VAUXN15[15:0]	Inputs	16 auxiliary analog-input pairs. Also, the XADC uses 16 differential digital-input pairs as low-bandwidth differential analog inputs. These inputs are configured as analog during FPGA configuration.
USER_TEMP_ALARM_OUT	Output	XADC temperature-sensor alarm output.
VCCINT_ALARM_OUT	Output	XADC VCCINT-sensor alarm output.
VCCAUX_ALARM_OUT	Output	XADC VCCAUX-sensor alarm output.
OT_OUT	Output	Over-Temperature alarm output.
CHANNEL_OUT[4:0]	Outputs	Channel selection outputs. The ADC input MUX channel selection for the current ADC conversion is placed on these outputs at the end of an ADC conversion.
EOC_OUT	Output	End of Conversion signal. This signal transitions to an active-High at the end of an ADC conversion when the measurement result is written to the status registers. For detailed information, see the XADC Timing section in the *7 Series FPGAs XADC User Guide* [Ref 2].
EOS_OUT	Output	End of Sequence. This signal transitions to an active-High when the measurement data from the last channel in the Channel Sequencer is written to the status registers. For detailed information, see the XADC Timing section in the *7 Series FPGAs XADC User Guide* [Ref 2].
BUSY_OUT	Output	ADC busy signal. This signal transitions High during an ADC conversion. This signal transitions High for an extended period during calibration.
JTAGLOCKED_OUT	Output	Used to indicate that DRP port has been locked by the JTAG interface.
JTAGMODIFIED_OUT	Output	Used to indicate that a JTAG write to the DRP has occurred

Table 2-2: **XADC I/O Signals *(Cont'd)***

Port	Direction	Description
JTAGBUSY_OUT	Output	Used to indicate that a JTAG DRP transaction is in progress
VBRAM_ALARM_OUT	Output	XADC VBRAM sensor alarm output.
VCCPINT_ALARM_OUT	Output	XADC VCCPINT sensor alarm output.
VCCPAUX_ALARM_OUT	Output	XADC VCCPAUX sensor alarm output.
VCCDDRO_ALARM_OUT	Output	XADC VCCDDRO sensor alarm output.
MUXADDR_OUT[4:0]	Output	Use in external multiplexer mode to decode external MUX channel.
ALARM_OUT	Output	Logic OR of alarms. Can be used to flag occurrence of any alarm.
S_AXI_ACLK	Input	AXI Clock
S_AXI_ARESETN	Input	AXI Reset, Active-Low
S_AXI_AWADDR[(C_S_AXI_ADDR_WIDTH - 1):0]	Input	AXI Write address. The write address bus gives the address of the write transaction.
S_AXI_AWVALID	Input	Write address valid. This signal indicates that a valid write address and control information are available.
S_AXI_AWREADY	Output	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
S_AXI_WDATA[(C_S_AXI_DATA_WIDTH - 1):0]	Input	Write data
S_AXI_WSTB[((C_S_AXI_DATA_WIDTH/8) - 1):0]	Input	Write strobes. This signal indicates which byte lanes to update in memory.
S_AXI_WVALID	Input	Write valid. This signal indicates that valid write data and strobes are available.
S_AXI_WREADY	Output	Write ready. This signal indicates that the slave can accept the write data.
S_AXI_BRESP[1:0]	Output	Write response. This signal indicates the status of the write transaction 00 = OKAY (normal response) 10 =SLVERR (error condition) 11 = DECERR (not issued by core)
S_AXI_BVALID	Output	Write response valid. This signal indicates that a valid write response is available.
S_AXI_BREADY	Input	Response ready. This signal indicates that the master can accept the response information.
S_AXI_ARADDR[(C_S_AXI_ADDR_WIDTH - 1):0]	Input	Read address. The read address bus gives the address of a read transaction.
S_AXI_ARVALID	Input	Read address valid. This signal indicates, when HIGH, that the read address and control information is valid and remains stable until the address acknowledgement signal, S_AXI_ARREADY, is high.
S_AXI_ARREADY	Output	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
S_AXI_RDATA[(C_S_AXI_DATA_WIDTH - 1):0]	Output	Read data

Table 2-2: **XADC I/O Signals *(Cont'd)***

Port	Direction	Description
S_AXI_RRESP[1:0]	Output	Read response. This signal indicates the status of the read transfer. 00 = OKAY (normal response) 10 =SLVERR (error condition) 11 = DECERR (not issued by core)
S_AXI_RVALID	Output	Read valid. This signal indicates that the required read data is available and the read transfer can complete.
S_AXI_RREADY	Input	Read ready. This signal indicates that the master can accept the read data and response information.
TEMP_OUT[11:0]	Output	12-bit temperature output bus for MIG. This should be connected to xadc_device_temp_i_pin of MIG.

Notes:
1. AXI4-Lite ports are available only when Interface selection is AXI4-Lite.
2. DRP, JTAG, and RESET_IN ports are not available when AXI4-Lite interface is selected.

Register Space

The XADC functionality is configured through control registers (See the Register File Interface sections in the *7 Series FPGAs XADC User Guide* [Ref 2]). Table 2-3 lists the attributes associated with these control registers. These control registers can be initialized using HDL by attaching HDL attributes to the XADC primitive instance and configuring them according to Table 2-3. The control registers can also be initialized through the AXI4-Lite or DRP at run time. The XADC Wizard simplifies the initialization of these control registers in the HDL instantiation by automatically configuring them to implement the operating behavior you specify using the IP core GUI.

Table 2-3: **XADC Attributes**

Attribute	Name	Control Reg Address	Description
INIT_40	Configuration register 0	40h	XADC configuration registers. For detailed information, see the *7 Series FPGAs XADC User Guide* [Ref 2].
INIT_41	Configuration register 1	41h	
INIT_42	Configuration register 2	42h	
INIT_48 to INIT_4F	Sequence registers	48h to 4Fh	Sequence registers used to program the Channel Sequencer function in the XADC. For detailed information, see the *7 Series FPGAs XADC User Guide* [Ref 2].
INIT_50 to INIT_5F	Alarm Limits registers	50h to 5Fh	Alarm threshold registers for the XADC alarm function. For detailed information, seethe *7 Series FPGAs XADC User Guide* [Ref 2].

Table 2-3: **XADC Attributes *(Cont'd)***

Attribute	Name	Control Reg Address	Description
SIM_MONITOR_FILE	Simulation Analog Entry File	–	This is the text file that contains the analog input stimulus. This is used for simulation.
SIM_DEVICE	Device family information	–	Specifies the device family. For 7 Series devices, this value is "7Series".

XADC Wizard Register Descriptions for AXI4-Lite Interface

Table 2-4 shows the XADC Wizard IP Core registers and their corresponding addresses.

Table 2-4: **IP Core Registers**

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
XADC Wizard Local Register Grouping				
C_BASEADDR + 0x00	Software Reset Register (SRR)	Write(1)	N/A	Software reset register
C_BASEADDR + 0x04	Status Register (SR)	Read(2)	N/A	Status register
C_BASEADDR + 0x08	Alarm Output Status Register (AOSR)	Read(2)	0x0	Alarm output status register
C_BASEADDR + 0x0C	CONVST Register (CONVSTR)	Write(1)	N/A	Bit[0] = ADC convert start register(3) Bit[1] = Enable temperature update logic Bit[17:2] = Wait cycle for temperature update
C_BASEADDR + 0x10	XADC Reset Register (SYSMONRR)	Write(1)	N/A	XADC hard macro reset register
XADC Wizard Interrupt Controller Register Grouping				
C_BASEADDR + 0x5C	Global Interrupt Enable Register (GIER)	R/W	0x0	Global interrupt enable register

Table 2-4: **IP Core Registers *(Cont'd)***

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
C_BASEADDR + 0x60	IP Interrupt Status Register (IPISR)	R/TOW[4]	N/A	IP interrupt status register
C_BASEADDR + 0x68	IP Interrupt Enable Register (IPIER)	R/W	0x0	IP interrupt enable register
XADC Wizard Hard Macro Register Grouping[5]				
C_BASEADDR + 0x200	Temperature	Read[6]	N/A	The 12-bit Most Significant Bit (MSB) justified result of on-device temperature measurement is stored in this register.
C_BASEADDR + 0x204	V_{CCINT}	Read[6]	N/A	The 12-bit MSB justified result of on-device V_{CCINT} supply monitor measurement is stored in this register.
C_BASEADDR + 0x208	V_{CCAUX}	Read[6]	N/A	The 12-bit MSB justified result of on-device V_{CCAUX} Data supply monitor measurement is stored in this register.
C_BASEADDR + 0x20C	V_P/V_N	R/W[7]	0x0	When read: The 12-bit MSB justified result of A/D conversion on the dedicated analog input channel (Vp/Vn) is stored in this register. When written: Write to this register resets the XADC hard macro. No specific data is required.
C_BASEADDR + 0x210	V_{REFP}	Read[6]	0x0	The 12-bit MSB justified result of A/D conversion on the reference input V_{REFP} is stored in this register.
C_BASEADDR + 0x214	V_{REFN}	Read[6]	0x0	The 12-bit MSB justified result of A/D conversion on the reference input V_{REFN} is stored in this register.
C_BASEADDR + 0x218	V_{BRAM}	Read[6]	0x0	The 12-bit MSB justified result of A/D conversion on the reference input V_{BRAM} is stored in this register.
C_BASEADDR + 0x21C	Undefined	N/A	Undefined	These locations are unused and contain invalid data.
C_BASEADDR + 0x220	Supply Offset	Read[6]	N/A	The calibration coefficient for the supply sensor offset is stored in this register.
C_BASEADDR + 0x224	ADC Offset	Read[6]	N/A	The calibration coefficient for the ADC offset calibration is stored in this register.
C_BASEADDR + 0x228	Gain Error	Read[6]	N/A	The calibration coefficient for the gain error is stored in this register.

Table 2-4: **IP Core Registers *(Cont'd)***

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
C_BASEADDR + 0x22C to C_BASEADDR + 0x23C	Undefined	N/A	Undefined	These locations are unused and contain invalid data.
C_BASEADDR + 0x240	$V_{AUXP}[0]$/ $V_{AUXN}[0]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 0 is stored in this register.
C_BASEADDR + 0x244	$V_{AUXP}[1]$/ $V_{AUXN}[1]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 1 is stored in this register.
C_BASEADDR + 0x248	$V_{AUXP}[2]$/ $V_{AUXN}[2]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 2 is stored in this register.
C_BASEADDR + 0x24C	$V_{AUXP}[3]$/ $V_{AUXN}[3]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 3 is stored in this register.
C_BASEADDR + 0x250	$V_{AUXP}[4]$/ $V_{AUXN}[4]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 4 is stored in this register.
C_BASEADDR + 0x254	$V_{AUXP}[5]$/ $V_{AUXN}[5]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 5 is stored in this register.
C_BASEADDR + 0x258	$V_{AUXP}[6]$/ $V_{AUXN}[6]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 6 is stored in this register.
C_BASEADDR + 0x25C	$V_{AUXP}[7]$/ $V_{AUXN}[7]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 7 is stored in this register.
C_BASEADDR + 0x260	$V_{AUXP}[8]$/ $V_{AUXN}[8]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 8 is stored in this register.
C_BASEADDR + 0x264	$V_{AUXP}[9]$/ $V_{AUXN}[9]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 9 is stored in this register.
C_BASEADDR + 0x268	$V_{AUXP}[10]$/ $V_{AUXN}[10]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 10 is stored in this register.
C_BASEADDR + 0x26C	$V_{AUXP}[11]$/ $V_{AUXN}[11]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 11 is stored in this register.
C_BASEADDR + 0x270	$V_{AUXP}[12]$/ $V_{AUXN}[12]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 12 is stored in this register.

Table 2-4: **IP Core Registers *(Cont'd)***

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
C_BASEADDR + 0x274	$V_{AUXP}[13]$/ $V_{AUXN}[13]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 13 is stored in this register.
C_BASEADDR + 0x278	$V_{AUXP}[14]$/ $V_{AUXN}[14]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 14 is stored in this register.
C_BASEADDR + 0x27C	$V_{AUXP}[15]$/ $V_{AUXN}[15]$	Read(6)	0x0	The 12-bit MSB justified result of A/D conversion on the auxiliary analog input 15 is stored in this register.
C_BASEADDR + 0x280	Max Temp	Read(6)	N/A	The 12-bit MSB justified maximum temperature measurement.
C_BASEADDR + 0x284	Max V_{CCINT}	Read(6)	N/A	The 12-bit MSB justified maximum V_{CCINT} measurement.
C_BASEADDR + 0x288	Max V_{CCAUX}	Read(6)	N/A	The 12-bit MSB justified maximum V_{CCAUX} measurement.
C_BASEADDR + 0x28C	Max V_{BRAM}	Read(6)	N/A	The 12-bit MSB justified maximum V_{BRAM} measurement.
C_BASEADDR + 0x290	Min Temp	Read(6)	N/A	The 12-bit MSB justified minimum temperature measurement
C_BASEADDR + 0x294	Min V_{CCINT}	Read(6)	N/A	The 12-bit MSB justified minimum V_{CCINT} measurement
C_BASEADDR + 0x298	Min V_{CCAUX}	Read(6)	N/A	The 12-bit MSB justified minimum V_{CCAUX} measurement.
C_BASEADDR + 0x29C	Min V_{BRAM}	Read(6)	N/A	The 12-bit MSB justified minimum V_{BRAM} measurement.
C_BASEADDR + 0x2A0 to C_BASEADDR + 0x2F8	Undefined	N/A	Undefined	These locations are unused and contain invalid data.
C_BASEADDR + 0x2FC	Flag Register	Read(6)	N/A	The 16-bit register gives general status information of ALARM, Over Temperature (OT), Disable information of XADC and information whether the XADC is using internal reference voltage or external reference voltage.
C_BASEADDR + 0x300	Configuration Register 0	R/W(8)	0x0	XADC Configuration register 0
C_BASEADDR + 0x304	Configuration Register 1	R/W(8)	0x0	XADC Configuration register 1
C_BASEADDR + 0x308	Configuration Register 2	R/W(8)	0x1E00	XADC Configuration register 2

Table 2-4: **IP Core Registers *(Cont'd)***

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
C_BASEADDR + 0x30C to C_BASEADDR + 0x31C	Test register 0 to 4	N/A	N/A	XADC Test register 0 to 4 (for factory test only)
C_BASEADDR + 0x320	Sequence Register 0	R/W	0x0	XADC Sequence register 0 (ADC channel selection)
C_BASEADDR + 0x324	Sequence Register 1	R/W	0x0	XADC Sequence register 1 (ADC channel selection)
C_BASEADDR + 0x328	Sequence Register 2	R/W	0x0	XADC Sequence register 2 (ADC channel averaging enable)
C_BASEADDR + 0x32C	Sequence Register 3	R/W	0x0	XADC Sequence register 3 (ADC channel averaging enable)
C_BASEADDR + 0x330	Sequence Register 4	R/W	0x0	XADC Sequence register 4 (ADC channel analog-input mode)
C_BASEADDR + 0x334	Sequence Register 5	R/W	0x0	XADC Sequence register 5 (ADC channel analog-input mode)
C_BASEADDR + 0x338	Sequence Register 6	R/W	0x0	XADC Sequence register 6 (ADC channel acquisition time)
C_BASEADDR + 0x33C	Sequence Register 7	R/W	0x0	XADC Sequence register 7 (ADC channel acquisition time)
C_BASEADDR + 0x340	Alarm Threshold Register 0	R/W	0x0	The 12-bit MSB justified alarm threshold register 0 (Temperature Upper)
C_BASEADDR + 0x344	Alarm Threshold Register 1	R/W	0x0	The 12-bit MSB justified alarm threshold register 1 (V_{CCINT} Upper)
C_BASEADDR + 0x348	Alarm Threshold Register 2	R/W	0x0	The 12-bit MSB justified alarm threshold register 2 (V_{CCAUX} Upper)
C_BASEADDR + 0x34C	Alarm Threshold Register 3	R/W(8)(9)	0x0	The 12-bit MSB justified alarm threshold register 3 (OT Upper)
C_BASEADDR + 0x350	Alarm Threshold Register 4	R/W	0x0	The 12-bit MSB justified alarm threshold register 4 (Temperature Lower)
C_BASEADDR + 0x354	Alarm Threshold Register 5	R/W	0x0	The 12-bit MSB justified alarm threshold register 5 (V_{CCINT} Lower)
C_BASEADDR + 0x358	Alarm Threshold Register 6	R/W	0x0	The 12-bit MSB justified alarm threshold register 6 (V_{CCAUX} Lower)

Table 2-4: **IP Core Registers *(Cont'd)***

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
C_BASEADDR + 0x35C	Alarm Threshold Register 7	R/W	0x0	The 12-bit MSB justified alarm threshold register 7 (OT Lower)
C_BASEADDR + 0x360	Alarm Threshold Register 8	R/W	0x0	The 12-bit MSB justified alarm threshold register 8 (VBRAM Upper)
C_BASEADDR + 0x364 to C_BASEADDR + 0x36C	Undefined	N/A	Undefined	These locations are unused and contain invalid data.
C_BASEADDR + 0x370	Alarm Threshold Register 9	R/W	0x0	The 12-bit MSB justified alarm threshold register 9 (VBRAM Lower)
C_BASEADDR + 0x374 to C_BASEADDR + 0x37C	Undefined	N/A	Undefined	These locations are unused and contain invalid data.
C_BASEADDR + 0x380 to C_BASEADDR + 0x3FC	Undefined	N/A	Undefined	Do not read/write these register.

Note:

1. Reading of this register returns an undefined value.
2. Writing into this register has no effect.
3. Used in event-driven sampling mode only.
4. TOW = Toggle On Write. Writing a 1 to a bit position within the register causes the corresponding bit position in the register to toggle.
5. These are 16-bit registers internal to XADC. These are mapped to the lower halfword boundary on 32-bit XADC Wizard IP core registers.
6. Writing to this XADC hard macro register is not allowed. The XADC hard macro data registers are 16 bits in width. The XADC hard macro specification guarantees the first 12 MSB bits accuracy; so only these bits are used for reference.
7. Writing to this register resets the XADC hard macro. No specific data pattern is required to reset the XADC hard macro. Reading of this register gives the details of Vp/Vn port.
8. Read the XADC User Guide, for setting the different bits available in configuration registers for 7 series devices.
9. The OT upper register is a user-configurable register for the upper threshold level of temperature. If this register is left unconfigured, then the XADC considers 125°C as the upper threshold value for OT. While configuring this register, the last four bits must be set to 0011, that is, Alarm Threshold Register 3[3:0] = 0011. The upper 12 bits of this register are user configurable.

XADC Wizard Local Register Grouping for AXI4-Lite Interface

It is expected that the XADC Wizard IP core registers are accessed in their preferred-access mode only. If the write attempt is made to read-only registers, then there is not any effect on register contents. If the write-only registers are read, then it results in undefined data. All

the internal registers of the core have to be accessed in 32-bit format. If any other kind of access (like halfword or byte access) is done for XADC Wizard IP core's local 32-bit registers, the transaction is completed with a generation of errors for the corresponding transaction.

Software Reset Register (SRR)

The Software Reset Register permits the programmer to reset the XADC Wizard IP core including the XADC hard macro output ports (except JTAG related outputs), independently of other IP cores in the systems. To activate software reset, the value 0x0000_000A must be written to the register. Any other access, read or write, has undefined results. The bit assignment in the software reset register is shown in Figure 2-1 and described in Table 2-5.

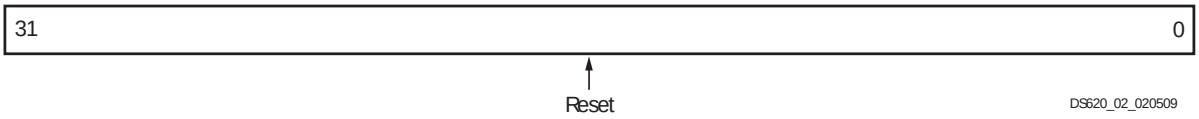


Figure 2-1: **Software Reset Register**

Table 2-5: **Software Reset Register Description (C_BASEADDR + 0x00)**

Bits	Name	Core Access	Reset Value	Description
31:0	Reset	Write only	N/A	The only allowed operation on this register is a write of 0x0000_000A, which resets the XADC Wizard IP Core. The reset is active only for 16 clock cycles.

Status Register (SR)

Status Register contains the XADC Wizard IP core channel status, EOC, EOS, and JTAG access signals. This register is read only. Any attempt to write the bits of the register is not able to change the bits. The Status Register bit definitions are shown in Figure 2-2 and explained in Table 2-6.

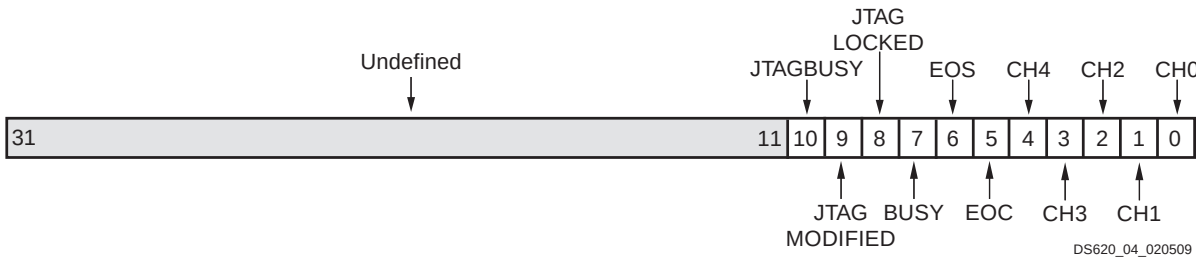


Figure 2-2: **Status Register**

Table 2-6: **Status Register (C_BASEADDR + 0x04)**

Bits	Name	Core Access	Reset Value	Description
31:11	Undefined	N/A	N/A	Undefined
10	JTAGBUSY	Read	0	Used to indicate that a JTAG DRP transaction is in progress.
9	JTAG MODIFIED	Read	0	Used to indicate that a write to DRP through JTAG interface has occurred. This bit is cleared when a successful DRP read/write operation through the FPGA logic is performed. The DRP read/write through the FPGA logic fails, if JTAGLOCKED = 1
8	JTAG LOCKED	Read	0	Used to indicate that a DRP port lock request has been made by the Joint Test Action Group (JTAG) interface.
7	BUSY	Read	N/A	ADC busy signal. This signal transitions high during an ADC conversion.
6	EOS	Read	N/A	End of Sequence. This signal transitions to an active-High when the measurement data from the last channel in the auto sequence is written to the status registers. This bit is cleared when a read operation is performed on status register.
5	EOC	Read	N/A	End of Conversion signal. This signal transitions to an active-High at the end of an ADC conversion when the measurement is written to the XADC hard macro's status register. This bit is cleared when a read operation is performed on status register.
4:0	CHANNEL [4:0]	Read	N/A	Channel selection outputs. The ADC input MUX channel selection for the current ADC conversion is placed on these outputs at the end of an ADC conversion.

Alarm Output Status Register (AOSR)

Alarm Output Status Register contains all the alarm outputs for the XADC Wizard IP core. This register is read only. Any attempt to write the bits of the register is not able to change the bits. The Alarm Output Status Register bit definitions are shown in Figure 2-3 and explained in Table 2-7.

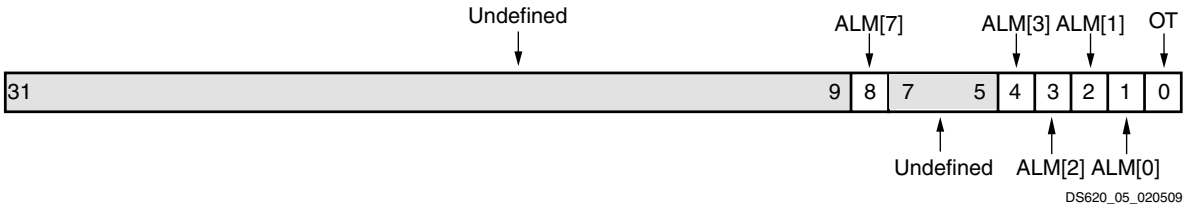


Figure 2-3: **Alarm Output Status Register**

Table 2-7: **Alarm Output Status Register (C_BASEADDR + 0x08)**

Bits	Name	Core Access	Reset Value	Description
31:9	Undefined	N/A	N/A	Undefined
8	ALM[7]	Read	0	Logical ORing of ALARM bits 0 to 7. This is direct output from the XADC macro.
7:5	Undefined	N/A	N/A	Undefined
4	ALM[3]	Read	0	**XADC V_{BRAM}-Sensor Status.** XADCV_{BRAM}-sensor alarm output interrupt occurs when V_{BRAM} exceeds user-defined threshold.
3	ALM[2]	Read	0	**XADC V_{CCAUX}-Sensor Status.** XADC V_{CCAUX}-sensor alarm output interrupt occurs when V_{CCAUX} exceeds user-defined threshold.
2	ALM[1]	Read	0	**XADC V_{CCINT}-Sensor Status.** XADC V_{CCINT}-sensor alarm output interrupt occurs when V_{CCINT} exceeds user-defined threshold.
1	ALM[0]	Read	0	**XADC Temperature-Sensor Status.** XADC temperature-sensor alarm output interrupt occurs when device temperature exceeds user-defined threshold.
0	OT	Read	0	**XADC Over-Temperature Alarm Status.** Over-Temperature alarm output interrupt occurs when the die temperature exceeds a factory set limit of 125°C.

CONVST Register (CONVSTR)

The CONVST Register is used for initiating a new conversion in the event-driven sampling mode. The output of this register is logically ORed with the external CONVST input signal. This register also defines enable for the Temperature Bus update logic and the wait cycle count. The attempt to read this register results in undefined data. The CONVST Register bit definitions are shown in Figure 2-4 and explained in Table 2-8.

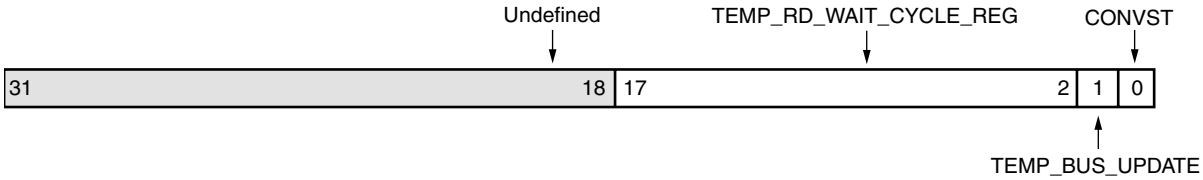


Figure 2-4: **CONVST Register**

Table 2-8: **CONVST Register (C_BASEADDR + 0x0C)**

Bits	Name	Core Access	Reset Value	Description
31:18	Undefined	N/A	N/A	Undefined
17:2	TEMP_RD_WAIT_CYCLE_REG	Write	0x03E8	Wait cycle for temperature update. Temperature update logic waits for this count of the S_AXI_ACLK.
1	TEMP_BUS_UPDATE	Write	0	Enable temperature update logic enables the temperature read from XADC and updates of TEMP_OUT port.
0	CONVST	Write	0	A rising edge on the CONVST input initiates start of ADC conversion in event-driven sampling mode. For the selected channel the CONVST bit in the register needs to be set to 1 and again reset to 0 to start a new conversion cycle. The conversion cycle ends with EOC bit going High.

XADC Reset Register

The XADC Reset Register is used to reset only the XADC hard macro. As soon as the reset is released the ADC begins with a new conversion. If sequencing is enabled this conversion is the first in the sequence. This register resets the OT and ALM[n] output from the XADC hard macro. This register does not reset the interrupt registers if they are included in the design. Also any reset from the FPGA logic does not affect the RFI (Register File Interface) contents of XADC hard macro. The attempt to read this register results in undefined data. The XADC Reset Register bit definitions are shown in Figure 2-5 and explained in Table 2-9.

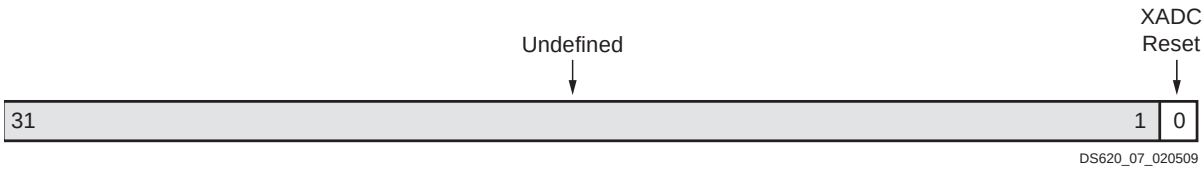


Figure 2-5: **XADC Reset Register**

Table 2-9: **XADC Reset Register (C_BASEADDR + 0x10)**

Bits	Name	Core Access	Reset Value	Description
31:1	Undefined	N/A	N/A	Undefined
0	XADC Reset	Write	0	Writing 1 to this bit position resets the XADC hard macro. The reset is released only after 0 is written to this register.

XADC Wizard Interrupt Controller Register Grouping for AXI4-Lite Interface

The Interrupt Controller Module is included in the XADC Wizard IP core design when C_INCLUDE_INTR = 1. The XADC Wizard has several distinct interrupts that are sent to the Interrupt Controller Module which is one of the submodules of XADC Wizard IP Core. The Interrupt Controller Module allows each interrupt to be enabled independently (via the IP interrupt enable register (IPIER)). All the interrupt signals are rising-edge sensitive.

Interrupt registers are strictly 32-bit accessible. If byte/halfword or without byte enables access is made, the core behavior is not guaranteed.

The interrupt registers are in the Interrupt Controller Module. The XADC Wizard IP core permits multiple conditions for an interrupt or an interrupt strobe which occurs only after the completion of a transfer.

Global Interrupt Enable Register (GIER)

The Global Interrupt Enable Register (GIER) is used to globally enable the final interrupt output from the Interrupt Controller as shown in Figure 2-6 and described in Table 2-10. This bit is a read/write bit and is cleared upon reset.

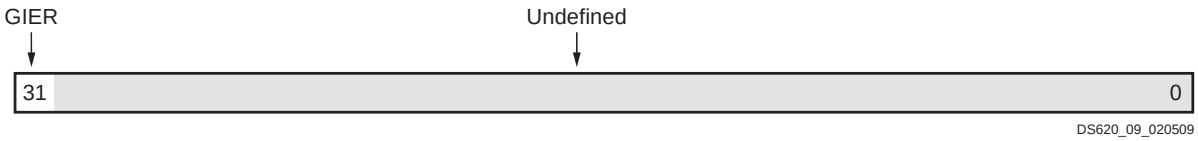


Figure 2-6: **Global Interrupt Enable Register (GIER)**

Table 2-10: **Global Interrupt Enable Register (GIER) Description (C_BASEADDR + 0x5C)**

Bits	Name	Access	Reset Value	Description
31	GIER	R/W	0	**Global Interrupt Enable Register.** It enables all individually enabled interrupts to be passed to the interrupt controller. 0 = Disabled 1 = Enabled
30:0	Undefined	N/A	N/A	Undefined.

IP Interrupt Status Register (IPISR)

Six unique interrupt conditions are possible in the XADC Wizard IP core.

The Interrupt Controller has a register that can enable each interrupt independently. Bit assignment in the Interrupt register for a 32-bit data bus is shown in Figure 2-7 and described in Table 2-11. The interrupt register is a read/toggle on write register and by

writing a 1 to a bit position within the register causes the corresponding bit position in the register to toggle. All register bits are cleared upon reset.

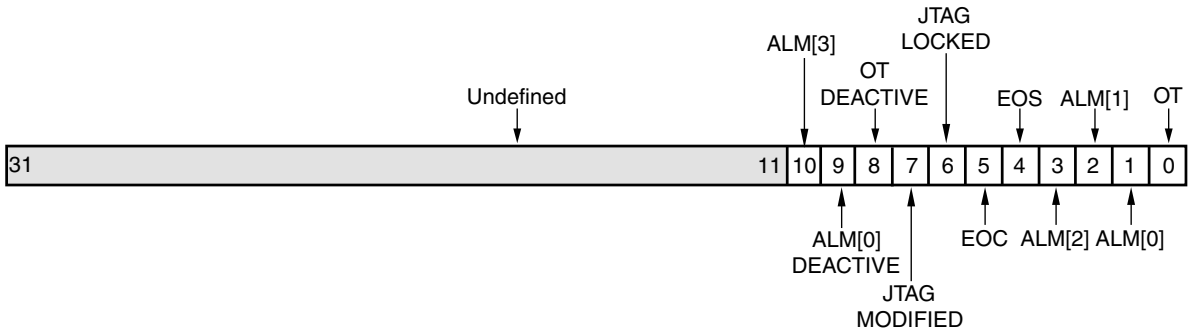


Figure 2-7: **IP Interrupt Status Register (IPISR)**

Table 2-11: **IP Interrupt Status Register (IPISR) Description (C_BASEADDR + 0x60)**

Bits	Name	Access	Reset Value	Description
31:11	Undefined	N/A	N/A	Undefined
10	ALM[3]	R/TOW(1)(2)	0	**XADC V_{BRAM}-Sensor Interrupt.** XADC V_{BRAM}-sensor alarm output interrupt occurs when V_{BRAM} exceeds user-defined threshold.
9	ALM[0] Deactive	R/TOW	0	**ALM[0] Deactive Interrupt.** This signal indicates that the falling edge of the Over Temperature signal is detected. It is cleared by writing 1 to this bit position. The ALM[0] signal is generated locally from the core. This signal indicates that the XADC macro has deactivated the Over Temperature signal output.
8	OT Deactive	R/TOW(1)	0	**OT Deactive Interrupt.** This signal indicates that falling edge of the Over Temperature signal is detected. It is cleared by writing 1 to this bit position. The OT Deactive signal is generated locally from the core. This signal indicates that the XADC macro has deactivated the Over Temperature signal output.
7	JTAG MODIFIED	R/TOW(1)(2)	0	**JTAGMODIFIED Interrupt.** This signal indicates that a write to DRP through the JTAG interface has occurred. It is cleared by writing 1 to this bit position.
6	JTAG LOCKED	R/TOW(1)(2)	0	**JTAGLOCKED Interrupt.** This signal is used to indicate that a DRP port lock request has been made by the Joint Test Action Group (JTAG) interface.
5	EOC	R/TOW(1)(2)	N/A	**End of Conversion Signal Interrupt.** This signal transitions to an active-High at the end of an ADC conversion when the measurement is written to the XADC hard macro's status register.
4	EOS	R/TOW(1)(2)	N/A	**End of Sequence Interrupt.** This signal transitions to an active-High when the measurement data from the last channel in the auto sequence is written to the status registers.

Table 2-11: **IP Interrupt Status Register (IPISR) Description (C_BASEADDR + 0x60) *(Cont'd)***

Bits	Name	Access	Reset Value	Description
3	ALM[2]	R/TOW(1)(2)	0	**XADC V_{CCAUX}-Sensor Interrupt.** XADC V_{CCAUX}-sensor alarm output interrupt occurs when V_{CCAUX} exceeds the user-defined threshold.
2	ALM[1]	R/TOW(1)(2)	0	**XADC V_{CCINT}-Sensor Interrupt.** XADC V_{CCINT}-sensor alarm output interrupt occurs when V_{CCINT} exceeds the user-defined threshold.
1	ALM[0]	R/TOW(1)(2)	0	**XADC Temperature-Sensor Interrupt.** XADC temperature-sensor alarm output interrupt occurs when device temperature exceeds the user-defined threshold.
0	OT	R/TOW(1)(2)	0	**Over-Temperature Alarm Interrupt.** Over-Temperature alarm output interrupt occurs when the die temperature exceeds a factory set limit of 125 °C.

Note:

1. TOW = Toggle On Write. Writing a 1 to a bit position within the register causes the corresponding bit position in the register to toggle.
2. This interrupt signal is directly generated from the XADC hard macro.

IP Interrupt Enable Register (IPIER)

The IPIER has an enable bit for each defined bit of the IPISR as shown in Figure 2-8 and described in Table 2-12. All bits are cleared upon reset.

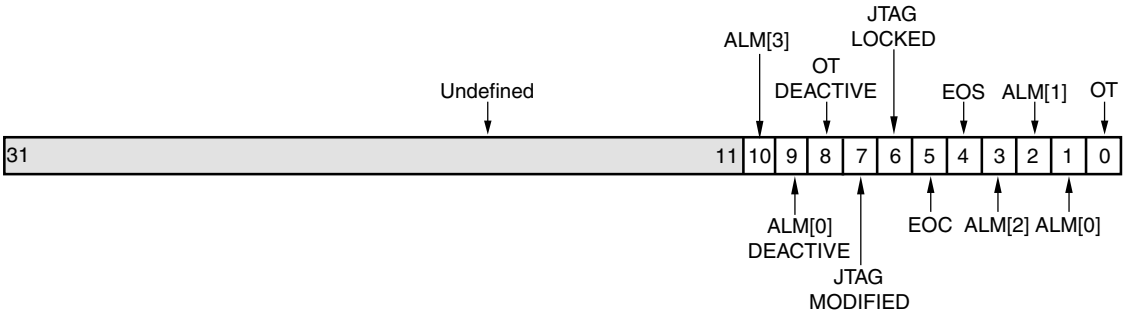


Figure 2-8: **IP Interrupt Enable Register (IPIER)**

Table 2-12: **IP Interrupt Enable Register (IPIER) Description (C_BASEADDR + 0x68)**

Bits	Name	Access	Reset Value	Description
31:11	Undefined	N/A	N/A	Undefined
10	ALM[3]	R/W	0	XADC V_{BRAM}-Sensor Interrupt 0 = Disabled 1 = Enabled
9	ALM[0] Deactive	R/W	0	ALM[0] Deactive Interrupt 0 = Disabled 1 = Enabled

Table 2-12: **IP Interrupt Enable Register (IPIER) Description (C_BASEADDR + 0x68) *(Cont'd)***

Bits	Name	Access	Reset Value	Description
8	OT Deactive	R/W	0	OT Deactive Interrupt 0 = Disabled 1 = Enabled
7	JTAG MODIFIED	R/W	0	JTAGMODIFIED Interrupt 0 = Disabled 1 = Enabled
6	JTAG LOCKED	R/W	0	JTAGLOCKED Interrupt 0 = Disabled 1 = Enabled
5	EOC	R/W	0	End of Conversion Signal Interrupt 0 = Disabled 1 = Enabled
4	EOS	R/W	0	End of Sequence Interrupt 0 = Disabled 1 = Enabled
3	ALM[2]	R/W	0	XADC V_{CCAUX}-Sensor Interrupt 0 = Disabled 1 = Enabled
2	ALM[1]	R/W	0	XADC V_{CCINT}-Sensor Interrupt 0 = Disabled 1 = Enabled
1	ALM[0]	R/W	0	XADC Temperature-Sensor Interrupt 0 = Disabled 1 = Enabled
0	OT	R/W	0	Over-Temperature Alarm Interrupt 0 = Disabled 1 = Enabled

More about Locally Generated Interrupt Bits in IPIER and IPISR

The interrupt bits ranging from Bit[16] to Bit[0] in IPISR as well as IPIER are direct output signals of the XADC hard macro. The signals like OT Deactive (Bit[8]), ALM[0] Deactive (Bit[9]), are locally generated in the core. These interrupts are generated on the falling edge of the Over Temperature and AML[0] signals. The falling edge of these signals can be used in controlling external things like controlling the fan or air-conditioning of the system.

XADC Hard Macro Register (DRP Register) Grouping for AXI4-Lite Interface

The XADC hard macro register set consists of all the registers present in the XADC hard macro on 7 series FPGAs. The addresses of these registers are mentioned in Table 2-4. Because these registers are 16 bits wide but the processor data bus is 32 bits wide, the hard macro register data resides on the lower 16 bits of the 32-bit data bus as shown in Figure 2-9. The 12-bit MSB aligned A/D converted value of different channels from XADC hard macro are left-shifted and reside from bit position 15 to 6 of the processor data bus. The remaining bit positions from 5 to 0 should be ignored while considering the ADC data for different channels. Along with 16-bit data, the JTAGMODIFIED and JTAGLOCKED bits are passed that can be used by the software driver application for determining the validity of the DRP read data. The JTAGMODIFIED bit is cleared when a DRP read/write operation through the FPGA logic is successful. A DRP read/write through the FPGA logic fails, if JTAGLOCKED = 1. The JTAGLOCKED signal is independently controlled through JTAG TAP. It is expected that these XADC hard macro registers should be accessed in their preferred access-mode only. The XADC Wizard IP core is not able to differentiate any non-preferred access to the XADC hard macro registers.

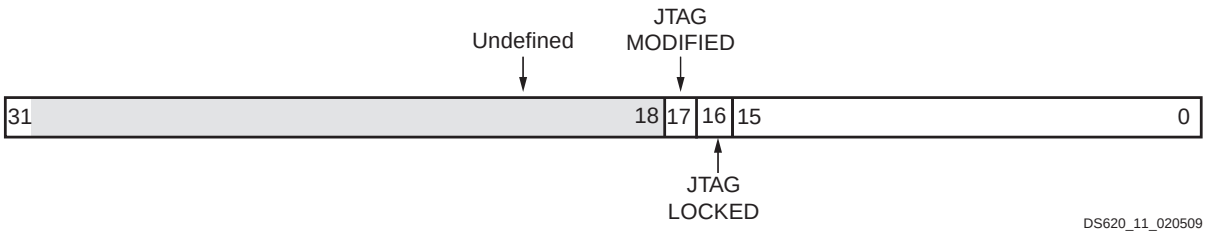


Figure 2-9: **XADC Hard Macro Register**

DRP registers are accessed as part of the core local registers.

IMPORTANT: *These registers must be accessed through the core local registers. Any attempt to access these registers in byte or halfword manner returns the error response from core.*

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

XADC Design Parameters

AXI4-Lite Selected

To allow you to obtain an XADC Wizard IP core that is uniquely tailored for your system, certain features can be parameterized in the XADC Wizard design. This allows you to configure a design that utilizes the resources required by the system only and that operates with the best possible performance. The features that can be parameterized are described in Table 3-1.

Table 3-1: **XADC Wizard Design Parameters**

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
System Parameters					
G1	Target FPGA family	C_FAMILY	Kintex7, Virtex7, Artix7	Kintex7	String
AXI4 Parameters					
G2	AXI Address Bus Width	C_S_AXI_ADDR_WIDTH	11	11	Integer
G3	AXI Data Bus Width	C_S_AXI_DATA_WIDTH	32	32	Integer
XADC Wizard Parameters					
G4	Include/Exclude Interrupt Support	C_INCLUDE_INTR	0 = Exclude interrupt support 1 = Include interrupt support	1	Integer

Table 3-1: **XADC Wizard Design Parameters *(Cont'd)***

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G5	File name for Analog input stimuli	C_SIM_MONITOR_FILE	string	Design.txt	String
G6	Include 12-bit temperature output bus for MIG	C_HAS_TEMP_BUS	0 = Exclude TEMP_OUT port and its update logic 1 = Include TEMP_OUT port and its update logic	0	Integer

Parameter – Port Dependencies

The dependencies between the XADC Wizard IP core design parameters and I/O signals are described in Table 3-2.

Table 3-2: **Parameter – Port Dependencies**

Generic or Port	Name	Affects	Depends	Relationship Description
Design Parameters				
G2	C_S_AXI_ADDR_WIDTH	P3, P13	–	Affects the number of bits in address bus
G3	C_S_AXI_DATA_WIDTH	P6, P7, P16	–	Affects the number of bits in data bus
I/O Signals				
P3	S_AXI_AWADDR[(C_S_AXI_ADDR_WIDTH - 1) : 0]	–	G4	Width of the S_AXI_AWADDR varies with C_S_AXI_ADDR_WIDTH.
P6	S_AXI_WDATA[(C_S_AXI_DATA_WIDTH - 1) : 0]	–	G5	Width of the S_AXI_WDATA varies according to C_S_AXI_DATA_WIDTH.
P7	S_AXI_WSTB[((C_S_AXI_DATA_WIDTH/8) - 1) : 0]	–	G5	Width of the S_AXI_WSTB varies according to C_S_AXI_DATA_WIDTH.
P13	S_AXI_ARADDR[(C_S_AXI_ADDR_WIDTH - 1) : 0]	–	G4	Width of the S_AXI_ARADDR varies with C_S_AXI_ADDR_WIDTH.
P16	S_AXI_RDATA[(C_S_AXI_DATA_WIDTH - 1) : 0]	–	G5	Width of the S_AXI_RDATA varies according to C_S_AXI_DATA_WIDTH.

Clocking

The clock to XADC primitive is DCLK. When AXI4-Lite is selected as the Bus interface, DCLK is connected to the S_AXI_ACLK clock. Hence the ADCCLK division factor must be programmed taking into consideration the S_AXI_ACLK frequency.

When DRP or None interface is selected, DCLK clock is at the top level of the IP and ADCCLK division factor must be programmed taking into consideration the DCLK frequency.

Resets

When AXI4-Lite is selected as the Bus interface, certain registers of the IP can be reset by writing a value 0xA to register 0x00. The AXI4-Lite interface also has its own reset pin.

When DRP or None interface is selected, RESET_IN is the input port at the top level of the IP.

Protocol Description

For more detailed information, see the AXI4-Lite protocol specifications. Figure 3-1 shows the simulation snapshots for Temperature value read from XADC register.

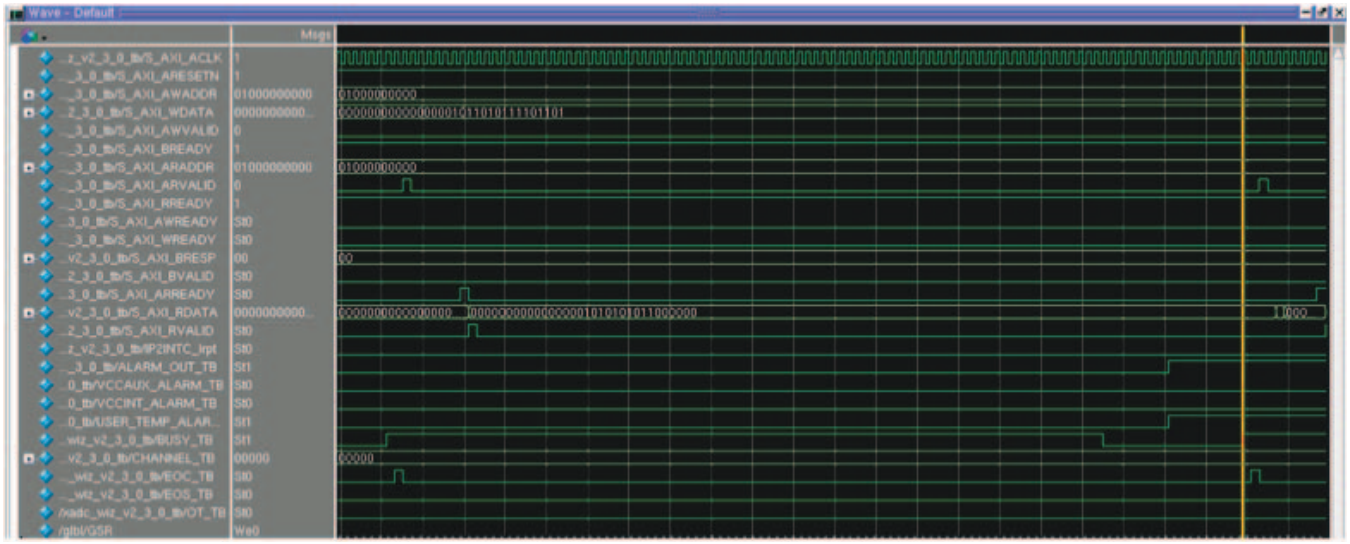


Figure 3-1: **Temperature Simulation Snapshot**

Customizing and Generating the Core

This chapter includes information about using Xilinx tools to customize and generate the core in the Vivado™ Design Suite environment.

GUI

This section describes how to set up a project in the Vivado Design Suite flow. For information on setting up a project in the ISE® Design Suite flow, consult the user guide for XADC Wizard at www.xilinx.com/support/documentation/ip_documentation/xadc_wiz/v2_1/ug772_xadc_wiz.pdf.

Before generating the example design, set up the project as described in Creating a Directory and Setting the Project Options of this guide.

Creating a Directory

To set up the example project, first create a directory using the following steps:

1. Change directory to the desired location. This example uses the following location and directory name:

   ```
   /Projects/xadc_example
   ```

2. Start Vivado™ Design Suite software.

   For help starting and using Vivado, see the *Vivado Help*, available in the *Vivado documentation* [Ref 7].

3. Choose **File > New Project** (Figure 4-1).
4. Change the name of the `.xpr` file (optional).
5. Create project with default settings.

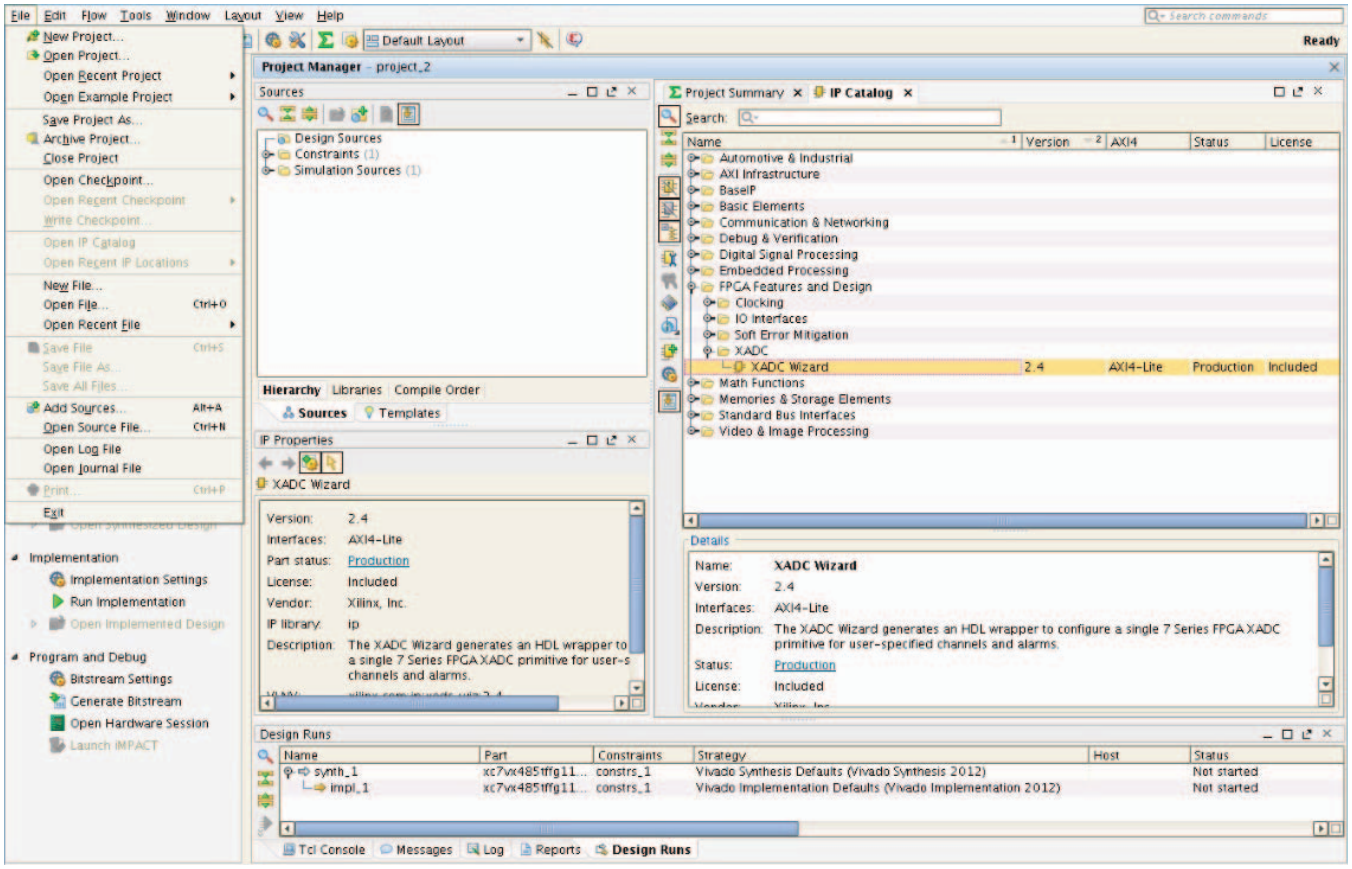


Figure 4-1: **New Project**

Setting the Project Options

Set the project options using the following steps:

1. Click **Project Part** in the option tree.
2. Select a **7 series FPGA** from the Family list.
3. Select a device from the Device list that support XADC primitive.
4. Select an appropriate package from the Package list. This example uses the XC7K235T device (see Figure 4-2).

   Note: If an unsupported silicon family is selected, the XADC Wizard remains light gray in the taxonomy tree and cannot be customized. Only devices containing the XADC are supported by the Wizard. See the *7 Series FPGAs Overview* [Ref 1] for a list of devices containing XADC.
5. Select either Verilog or VHDL as the target language.
6. Click **OK**.

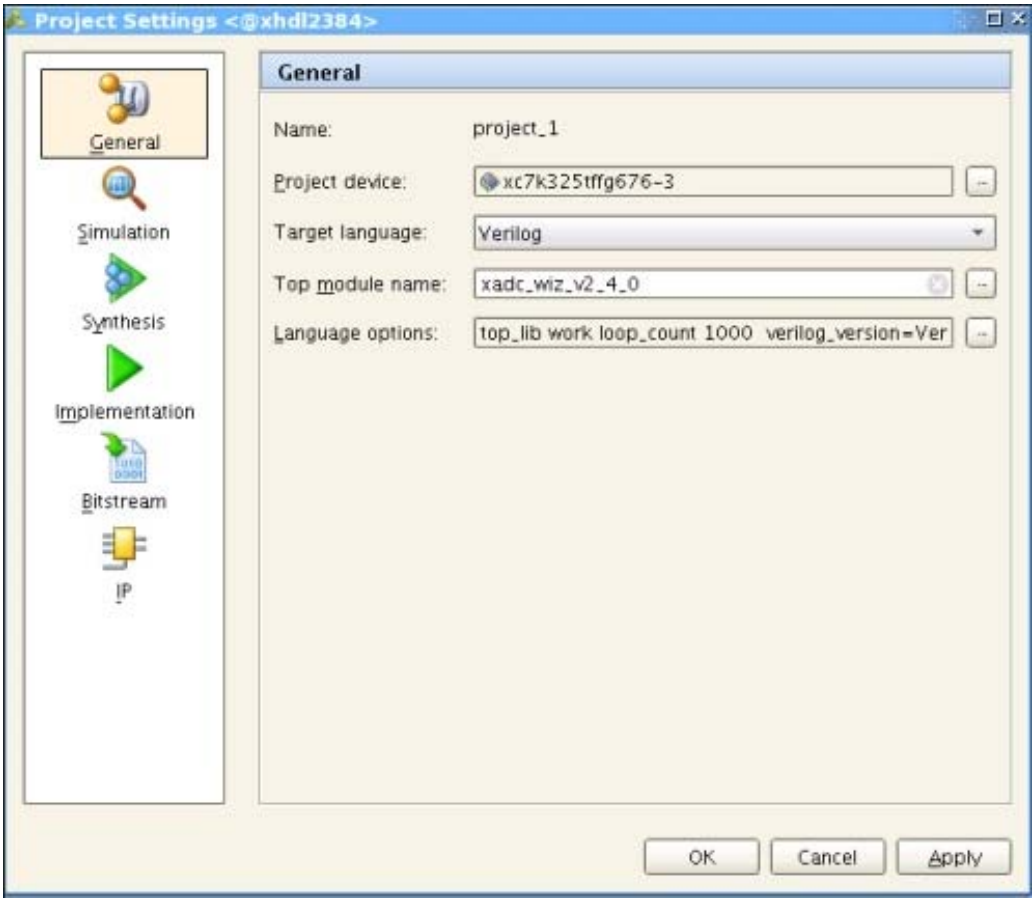


Figure 4-2: **Target Architecture Setting**

Generating the Core for 7 Series Devices

This section provides instructions for generating an example XADC design using the default values. The wrapper and its supporting files, including the example design, are generated in the project directory. For additional details about the example design files and directories provided with the XADC Wizard, see Detailed Example Design, page 43.

1. Select the **IP Catalog** button under the **Project Manager** tab to get the IP taxonomy view.
2. Locate the XADC Wizard in the taxonomy tree under:

   `/FPGA Features and Design/XADC.` (see Figure 4-1)

3. Double-click XADC Wizard to launch the Wizard.

After the wizard is launched, the IP catalog displays a series of screens that allow you to configure the XADC Wizard.

XADC Setup

The XADC Wizard screen (Figure 4-3) allows you to select the component name, interface type, startup channel mode, timing mode, analog stimulus file name, DRP timing options, and control and status ports.

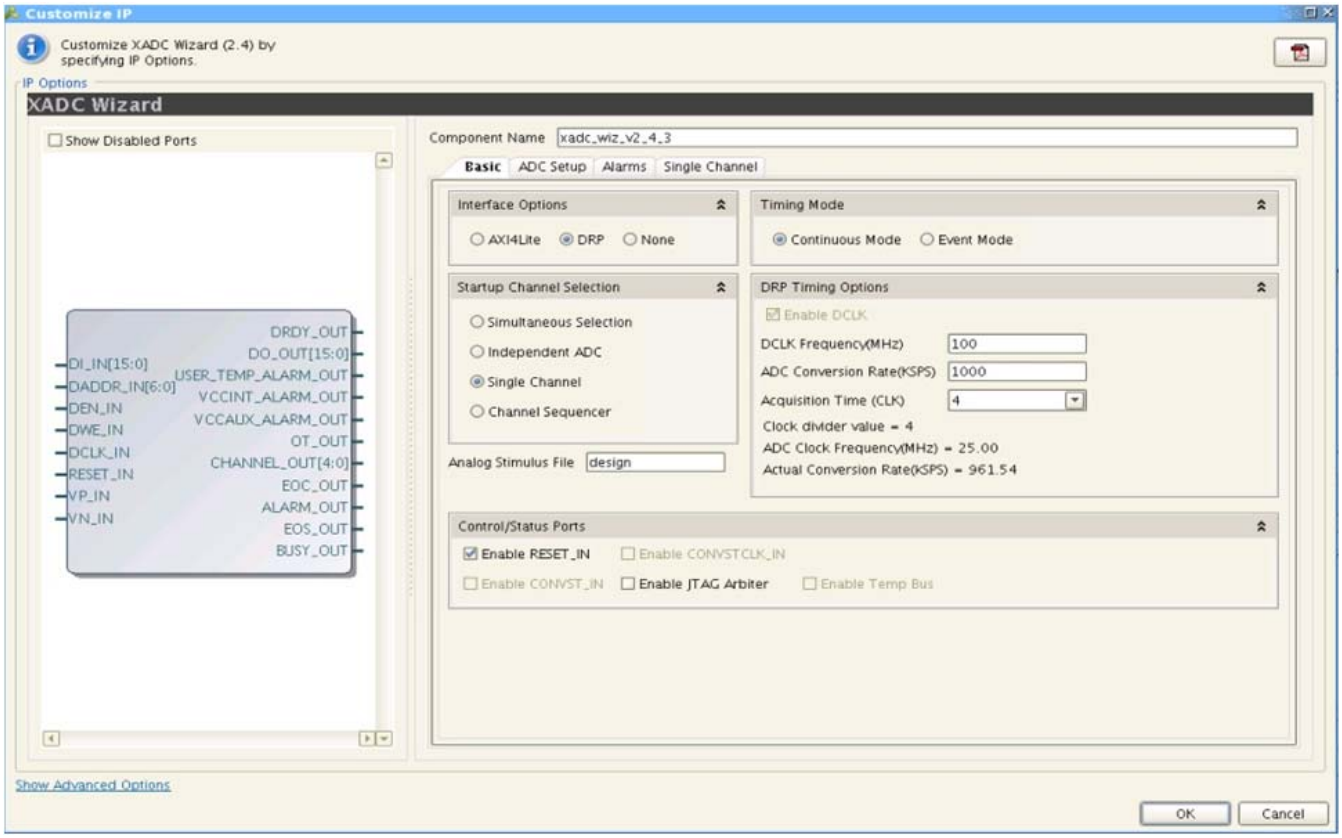


Figure 4-3: **XADC Setup Tab**

- **Component Name** – User selectable component name is available. Component names must not contain any reserved words in Verilog or VHDL.

Basic Tab

The following describes the Basic options in the XADC Wizard core.

- **Interface Options** – Use this field to select the interface for the XADC Wizard. DRP is the default option. User can select AXI4Lite, DRP, or None option. DRP port is the FPGA logic interface for XADC. It facilitates access to the register file interface of the XADC. The XADC control registers can be read or written using this port. This port can only be enabled when DCLK clock is present.
- **Startup Channel Selection** – XADC can be configured in one of the four modes listed:
  - **Simultaneous Selection** – This mode allows you to monitor two external channels simultaneously. For more information about this mode see the *7 Series FPGAs XADC*

User Guide [Ref 2].

- **Independent ADC** – This mode allows you to run the XADC in independent mode. Here, the XADC independently monitors the externals channels and at the same time monitors the FPGA voltages and temperature.
- **Single Channel** – In this mode, you can select only one channel to monitor.
- **Channel Sequencer** – Choosing this mode, allows you to select any number of channels to monitor. The channels to be used for this mode can be selected on Figure 4-6, page 40.

- **Timing Mode** – XADC can operate in two timing modes:
  - **Continuous Mode** – In this mode, the XADC continues to sample and convert the selected channel/channels.
  - **Event Mode** – This mode requires an external trigger event, CONVST or CONVSTCLK, to start a conversion on the selected channel. Event Mode should only be used with external channels.
- **DRP Timing Options** – The XADC clock (ADCCLK) is derived from the dynamic reconfiguration port (DRP) clock DCLK. The XADC supports a DRP clock frequency of up to 250 MHz. The XADC can also operate in absence of DCLK. For more information on the DRP see the *7 Series FPGAs XADC User Guide* [Ref 2].

  The ADCCLK clock, should be in the range of 4–26 MHz. To support this lower frequency clock the XADC has an internal clock divider. The GUI allows an external DCLK frequency and required ADC conversion rate (maximum 1 Msps) to be specified. Based on the value of DCLK clock, the wizard then calculates the appropriate clock divider value based on the values of DCLK clock and ADC conversion.

  The wizard also displays the ADC Clock frequency value and the actual conversion rate of the ADC.

- **Analog Stimulus File** – Use this field to customize the name of the XADC analog stimulus file.

Control/Status Ports

The Control/Status Port Selection (Figure 4-4) allows you to select the I/O ports on the XADC primitive.

- **Control Ports** – This section allows you to select control input ports:
  - RESET_IN allows an external input reset signal to be connected to the XADC
  - CONVST_IN and/or CONVSTCLK_IN as trigger sources for Event Mode Timing
- **Enable Temp Bus** – There is only one XADC primitive available in a 7 series device. If the XADC Wizard core is used in a system which uses MIG, the TEMP_OUT bus should be connected to the xadc_device_temp_i input port of the DDR3_SDRAM (MIG) block.

This disables inference of the XADC hard block in DDR3_SDRAM. Enabling this provides 12-bit TEMP_OUT port with the temperature update logic. This checkbox is available only when the interface option is AXI4-Lite.

- **Status Outputs** – Output status signals are also provided to facilitate interfacing of the XADC to a user design. For more information, see the *7 Series FPGAs XADC User Guide* [Ref 2].

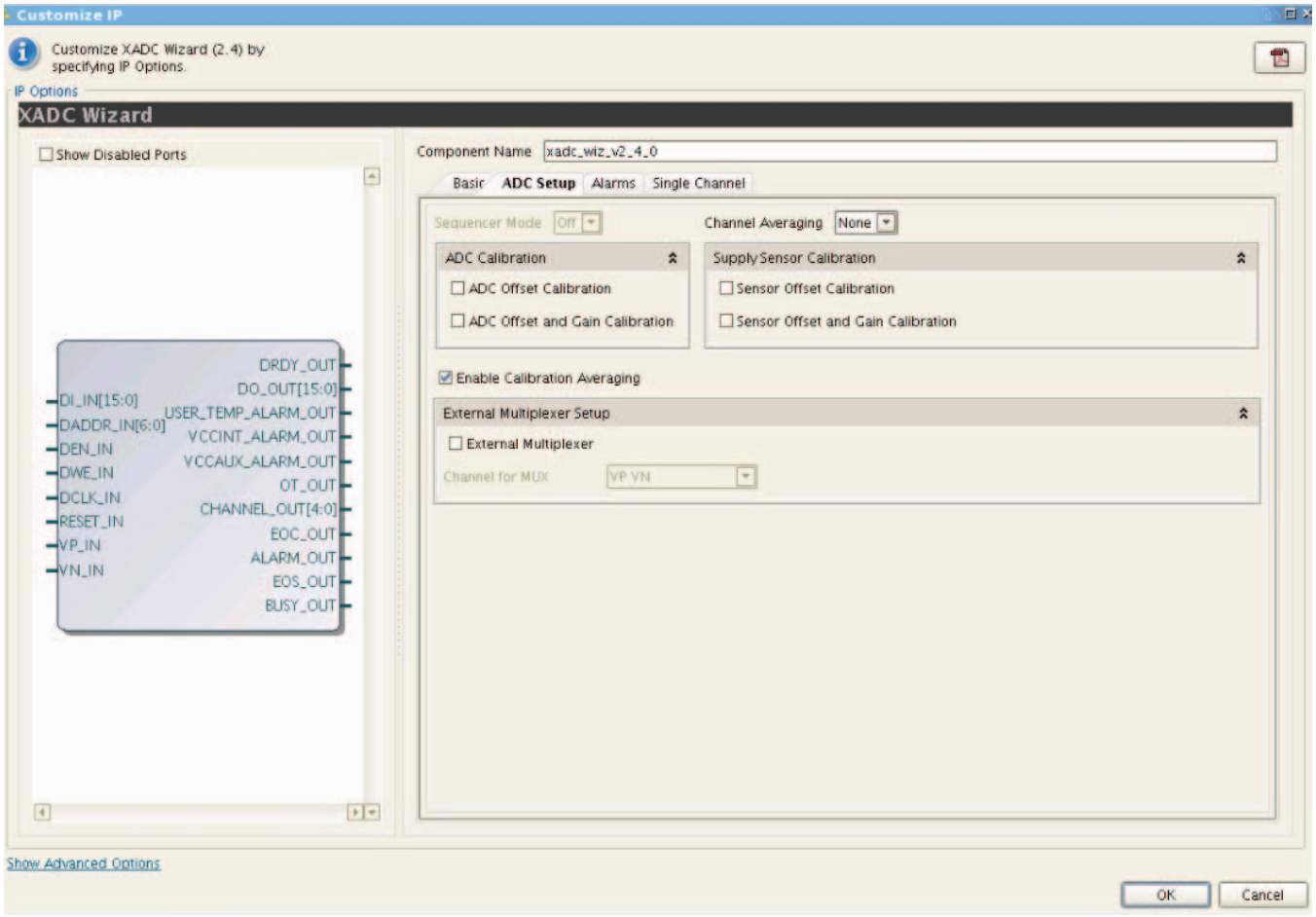


Figure 4-4: **ADC Setup Tab**

- **ADC Setup** – If the XADC is configured for Channel Sequencer, Simultaneous Sampling or Independent ADC mode, you can choose the required sequencer mode. The available options are Continuous, One-pass or Default mode.

  The Channel Averaging drop-down menu allows you to select the required averaging value. The available options are None, 16, 64, and 256.

  You can select the type of ADC Calibration and/or Supply Sensor Calibration by checking the respective checkboxes. Calibration Averaging is enabled by default in XADC. You can disable this by deselecting the box.

- **External Multiplexer Setup** – XADC supports a new timing mode that allows users to use an external analog multiplexer in situations where FPGA I/O resources might be limited or auxiliary analog I/O are more valuable when used to implement another interface.

  You can opt to use this feature by checking the box against Use External MUX. If checked, it is necessary to specify the external channel to which the MUX connects. Select this channel using the drop-down menu.

Alarm Setup

The Alarms (Figure 4-5) allows the alarm outputs to be enabled for the on-chip sensors. If a measurement of an on-chip sensor lies outside the specified limits, then a logic output goes active if enabled. For a detailed description of the alarm functionality see the *7 Series FPGAs XADC User Guide* [Ref 2].

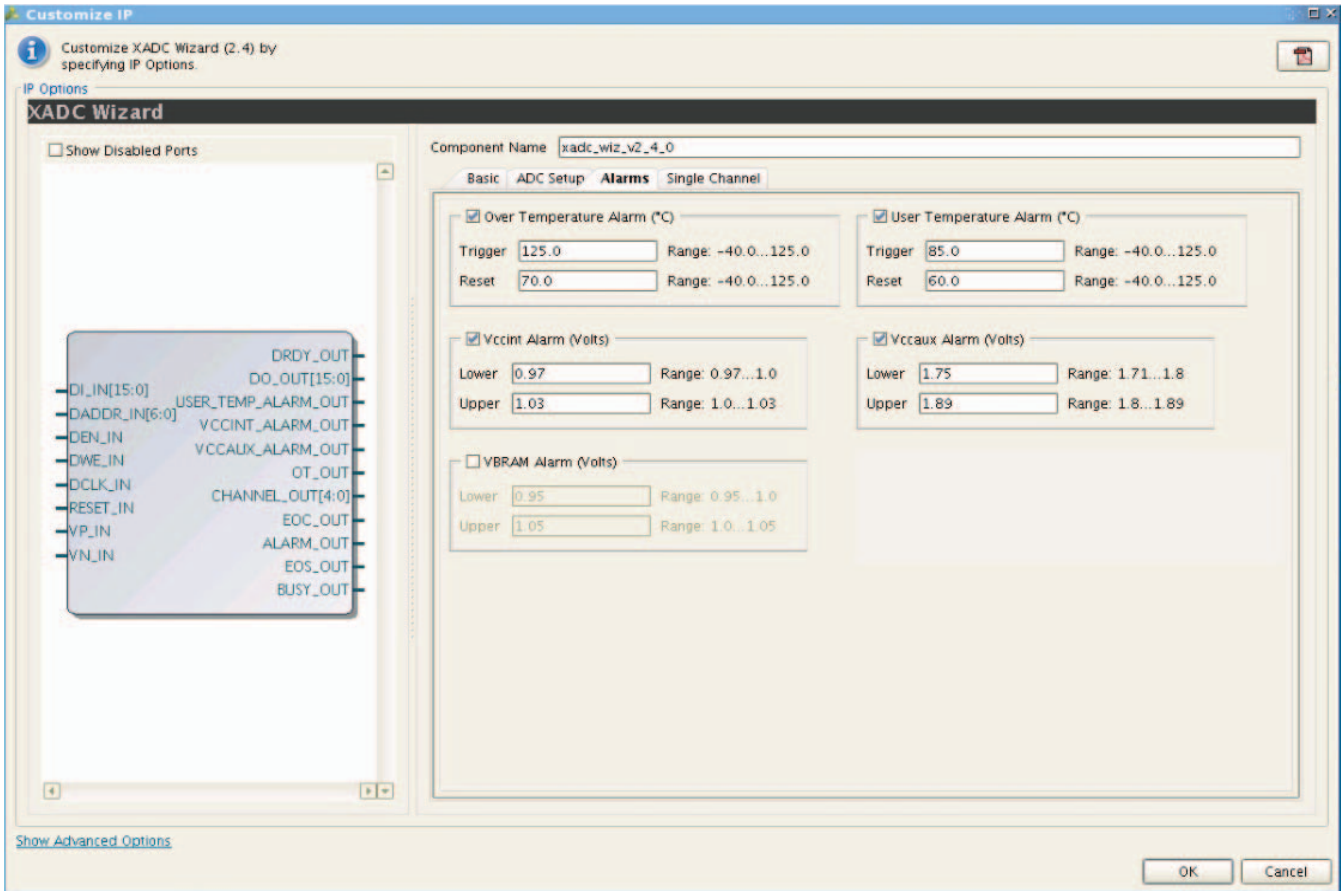


Figure 4-5: **Alarms Tab**

- **Enable Alarms** – Use the checkboxes to enable alarm logic outputs. The seven options are:
  - Over temperature alarm

- User temperature alarm
- V_{CCINT} alarm
- V_{CCAUX} alarm
- V_{BRAM} alarm

- **Temperature Alarm Limits** – Trigger and Reset levels for temperature alarm output can be entered using these fields. You can set both; the trigger as well as reset levels for the OT alarm.
- **V_{CCINT}, V_{CCAUX}, and V_{BRAM} Limits** – Both upper and lower alarm thresholds can be specified for the on-chip power supplies. If the measured value moves outside these limits the alarm logic output goes active. The alarm output is reset when a measurement inside these limits is generated. The default limits in the GUI represent ±5% on the nominal supply value.

Channel Sequencer Setup

Channel Sequencer (Figure 4-6) is used to configure the XADC sequence registers when the XADC is configured in Channel Sequencer, Simultaneous sampling, or Independent ADC mode. All the possible channels that can be included in the sequence are listed in the table spread across Figure 4-6 of the Wizard:

- Use the Channel Sequencer Setup screen to select Channels for monitoring, enable Averaging for selected channels, enable Bipolar mode for external channels and increase the Acquisition time for the selected channels.
- In the case of Simultaneous sampling mode, selecting channel Vauxp[0]/Vauxn[0] would automatically select channel Vauxp[8]/Vauxn[8]. Similarly selecting channel Vauxp[1]/Vauxn[1] would select channel Vauxp[9]/Vauxn[9] and so on.
- In case of Independent ADC mode, only external channels are listed and can be user-selected.

For more information about the simultaneous sampling mode and Independent ADC mode, see the *7 Series FPGAs XADC User Guide* [Ref 2].

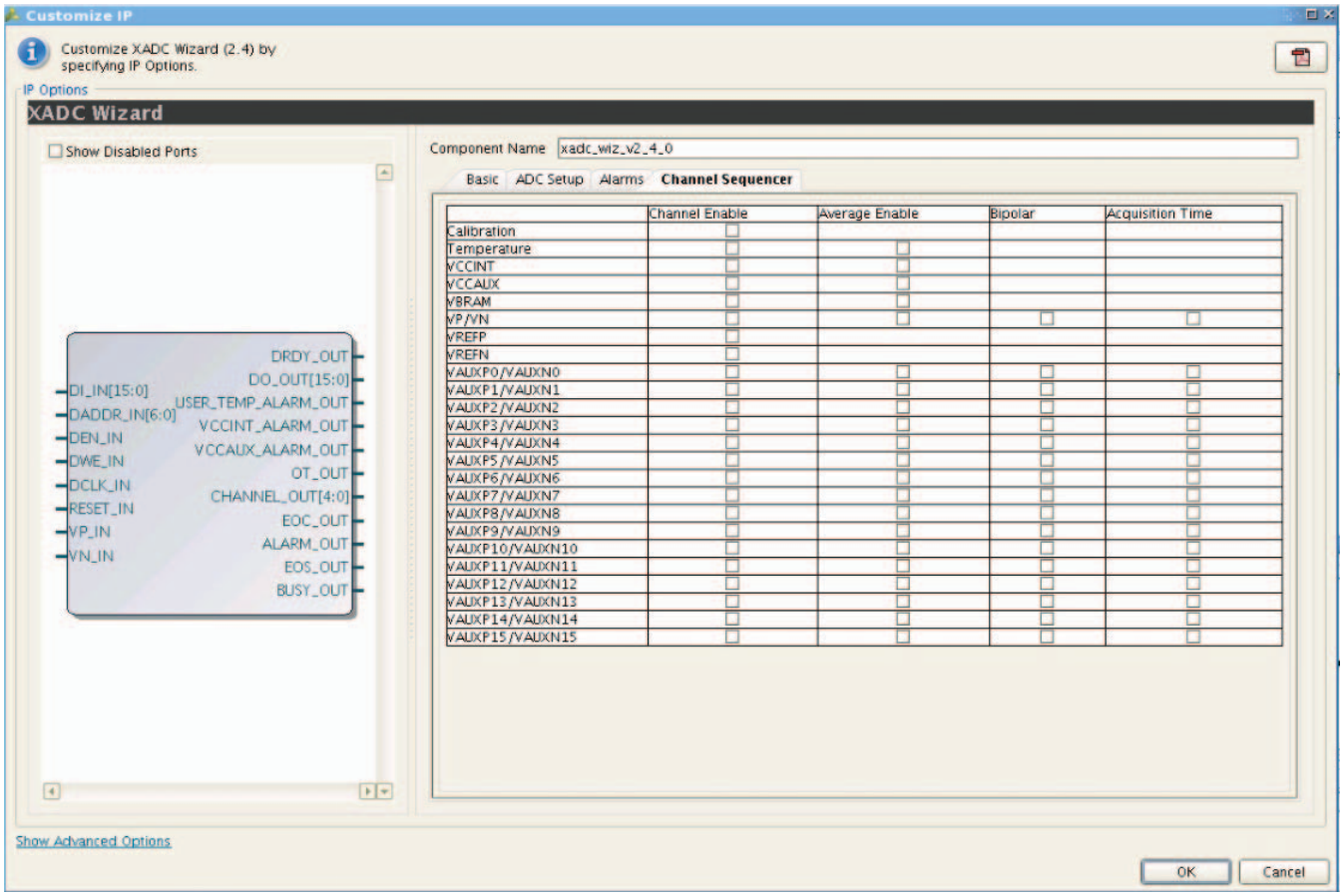


Figure 4-6: **Channel Sequencer Tab**

Single Channel Mode

If the Single Channel Mode operation (see Startup Channel Selection) is selected (Figure 4-3, page 35), the Single Channel Setup is displayed on Figure 4-7. The Single Channel allows you to select the channel for measurement and the analog input mode if the channel is an external analog input (that is, unipolar or bipolar).

The columns Channel Enable, Average Enable, and Increase Acquisition Time are disabled and are shown only information and ease.

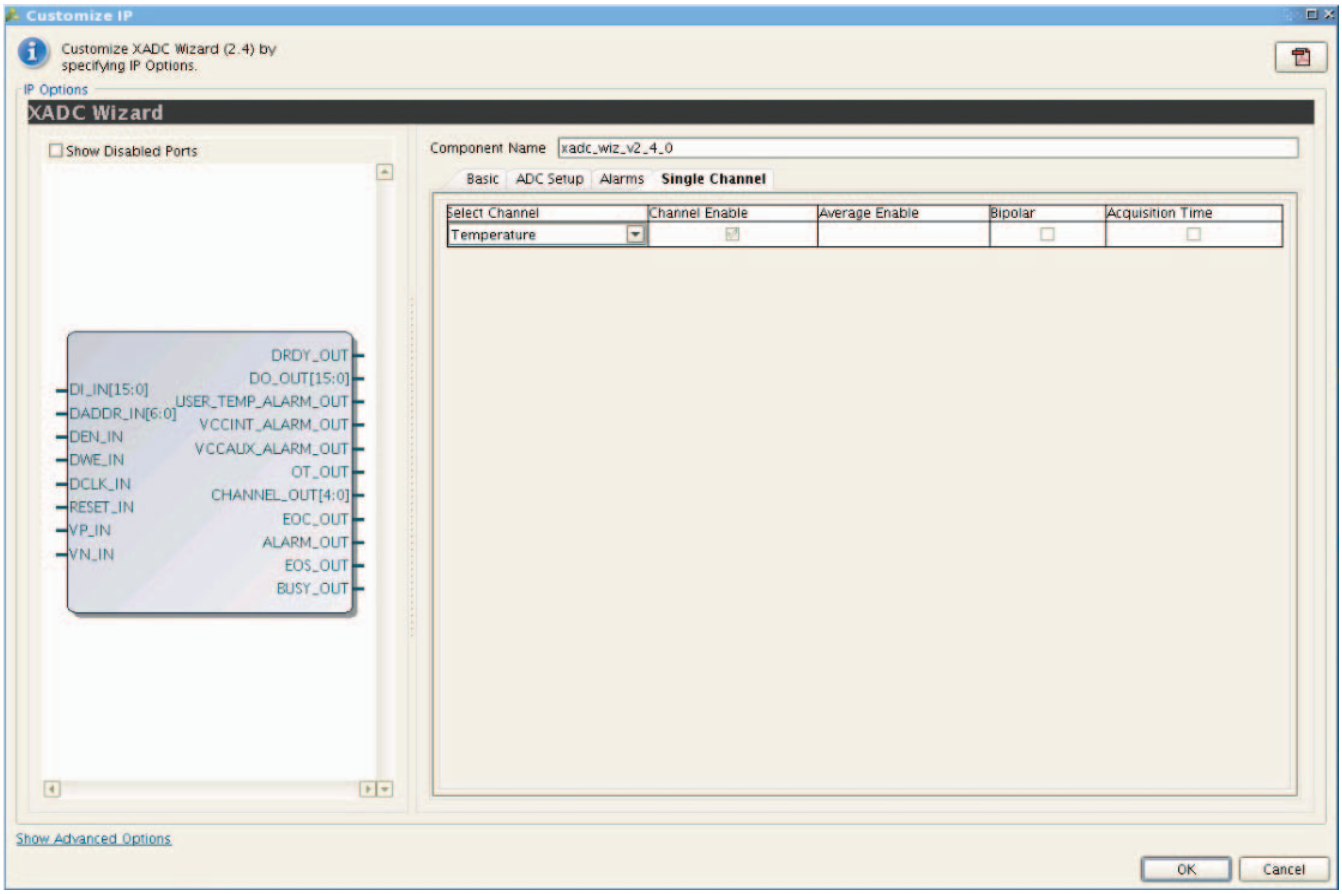


Figure 4-7: **Single Channel Tab**

Generating the HDL Wrapper

After selecting the configuration options, click **OK** on the Wizard screen to generate the HDL wrapper and other Wizard outputs.

The output files are placed in the `<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/` directory you selected or created when setting up a new Vivado project.

Output Generation

For more information on file and directory structure, see Directory and File Contents, page 43.

Constraining the Core

This chapter contains information about constraining the core in the Vivado™ Design Suite environment.

Required Constraints

For AXI4-Lite interface, the create_clock -name S_AXI_ACLK -period <clock period in ns> [get_ports S_AXI_ACLK].

For DRP, create_clock -name DCLK_IN -period <clock period in ns> [get_ports DCLK_IN].

Device, Package, and Speed Grade Selections

The XADC Wizard core supports all parts and packages.

Clock Frequencies

The clock frequencies supports from 8 to 250 MHz.

Clock Management

Depending on configuration, ADC clock is internally divided by the XADC primitive to achieve the desired sampling rate.

Detailed Example Design

This chapter contains information about the provided example design in the Vivado™ Design Suite environment.

Directory and File Contents

<project_name>/<project_name>.srcs/sources_1/ip/
Top-level project directory; name is user-defined

<project_name>/<project_name>.srcs/sources_1/ip/<component name>
Core release notes file

<component name>/doc
Product documentation

<component name>/example design
Verilog or VHDL design files

<component name>/implement/results
Results directory, created after implementation scripts are run, and contains implement script results

<component name>/simulation
Simulation scripts

simulation/functional
Functional simulation files

simulation/timing
Timing simulation files

The XADC Wizard directories and their associated files are defined in the following sections.

<project_name>/<project_name>.srcs/sources_1/ip/

The `<project_name>/<project_name>.srcs/sources_1/ip/` contains all the Vivado project files when DRP or None interface is selected.

Table 6-1: **Project Directory**

Name	Description
<project_name>/<project_name>.srcs/sources_1/ip/	
<component_name>.v[hd]	Verilog or VHDL synthesis/simulation model.
<component_name>.xci	Vivado project-specific option file; can be used as an input to the Vivado tools.
<component_name>_flist.txt	List of files delivered with the core.
<component_name>.{veo|vho}	VHDL or Verilog instantiation template.
<component_name>.xdc	Constraint file for core.

Back to Top

Additional files/directories generated when AXI4-Lite interface is selected.

Table 6-2: **Project Files/Directories**

Name	Description
Files	
<component_name>_axi_xadc.vhd	VHDL synthesis/simulation model.
<component_name>_axi_core_drp.vhd	VHDL synthesis/simulation model.
temp_rd_arbiter.vhd	VHDL synthesis and simulation model when Enable_Temp_Bus is TRUE.
drp_arbiter.vhd	VHDL synthesis and simulation model when Enable_Temp_Bus is TRUE.
drp_rdwr_fsm.vhd	VHDL synthesis and simulation model when Enable_Temp_Bus is TRUE.
Directories	
<component_name>/axi_lite_ipif_v1_01_a/	Directory containing AXI4-Lite interface VHDL files.
<component_name>/interrupt_control_v2_01_a/	Directory containing interrupt control VHDL files.
<component_name>/proc_common_v3_00_a/	Library used by AXI4-Lite and interrupt modules.

Back to Top

<project_name>/<project_name>.srcs/sources_1/ip/<component name>

The <component name> directory contains the readme file provided with the core, which can include last-minute changes and updates.

Table 6-3: **Component Name Directory**

Name	Description
<project_name>/<project_name>.srcs/sources_1/ip/<component_name>	
xadc_wiz_v2_4_readme.txt	Core readme file.

Back to Top

<component name>/doc

The doc directory contains the PDF documentation provided with the core.

Table 6-4: **Doc Directory**

Name	Description
<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/doc	
pg091-xadc-wiz.pdf	*LogiCORE IP XADC Wizard Product Guide*

Back to Top

<component name>/example design

The example design directory contains the example design files provided with the core.

Table 6-5: **Example Design Directory**

Name	Description
<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/example_design	
<component_name>_exdes.v(hd)	Verilog and VHDL top-level example design file.
<component_name>_exdes.xdc	Constraint file for example design.

Back to Top

<component name>/implement/results

The results directory should be created by the user and implementation files should be copied to the results directory before running timing simulations.

Table 6-6: **Results Directory**

Name	Description
<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/implement/results	
Implement script result files.	

Back to Top

<component name>/simulation

The `simulation` directory contains the simulation scripts provided with the core.

Table 6-7: **Simulation Directory**

Name	Description
`<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/simulation`	
`<component_name>_tb.v[hd]`	Demonstration test bench.

Back to Top

simulation/functional

The `functional` directory contains functional simulation scripts provided with the core.

Table 6-8: **Functional Directory**

Name	Description
`<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/simulation/functional`	
`simulate_xsim.sh` `simulate_xsim.bat`	Linux and Windows simulation scripts for Vivado simulator.
`simulate_mti.do`	ModelSim simulation script.
`simulate_ncsim.sh`	Linux script for running simulation using Cadence Incisive Enterprise Simulator (IES).
`simulate_vcs.sh`	Linux script for running simulation using VCS MX.

Back to Top

simulation/timing

The `timing` directory contains timing simulation scripts provided with the core.

Table 6-9: **Functional Directory**

Name	Description
`<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/simulation/timing`	
`simulate_xsim.sh` `simulate_xsim.bat`	Linux and Windows simulation scripts for Vivado simulator.
`simulate_mti.do`	ModelSim simulation script.
`simulate_ncsim.sh`	Linux script for running simulation using Cadence Incisive Enterprise Simulator (IES).

Back to Top

Simulation Scripts

Functional Simulation

The test scripts are a ModelSim, Cadence IES, VCS, VCS MX, or Vivado simulator macro that automate the simulation of the test bench. They are available from the following location:

```
<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/
simulation/functional/
```

The test script performs the following tasks:

- Compiles the structural UNISIM simulation model
- Compiles HDL Example Design source code
- Compiles the demonstration test bench
- Starts a simulation of the test bench
- Opens a Wave window and adds signals of interest
- Runs the simulation to completion

Timing Simulation

The test scripts are a ModelSim, Cadence IES, or Vivado simulator macro that automate the simulation of the demonstration test bench. They are available from the following location:

```
<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/
simulation/timing/
```

The test script performs the following tasks:

- Compiles the routed example design
- Compiles the demonstration test bench
- Starts a simulation of the test bench
- Opens a Wave window and adds signals of interest
- Runs the simulation to completion

Example Design

Top-Level Example Design

The following files describe the top-level example design for the XADC Wizard core.

VHDL

```
<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/
example_design/<component_name>_exdes.vhd
```

Verilog

```
<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/
example_design/<component_name>_exdes.v
```

The example design, instantiates the XADC core that is generated by the wizard.

Demonstration Test Bench

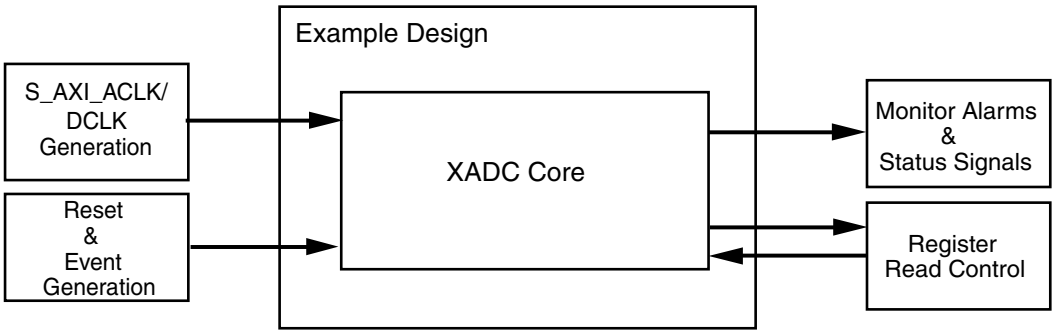


Figure 6-1: **Demonstration Test Bench for the XADC Wizard and Example Design**

The following files describe the demonstration test bench.

VHDL

```
<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/
simulation/<component_name>_tb.vhd
```

Verilog

```
<project_name>/<project_name>.srcs/sources_1/ip/<component_name>/
simulation/<component_name>_tb.v
```

The demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core.

The demonstration test bench performs the following tasks:

- Generates the input S_AXI_ACLK/DCLK clock signal
- Applies a reset to the example design
- Monitors the alarms and other status outputs
- Reads the respective registers when a conversion is complete

Open Example Project Flow

In the Vivado tools, the command

```
open_example_project [get_ips <component_name>]
```

in the tcl console invokes a separate example design project where it creates `<component_name>_exdes` as the top module for synthesis and `<component_name>_tb` as the top module for simulation. Implementation or simulation of the example design can be run from the example project.

Migrating

This appendix describes migrating from older versions of the IP to the current IP release.

For information on migrating to the Vivado™ Design Suite, see *Vivado Design Suite Migration Methodology Guide* (UG911) [Ref 6].

For a complete list of Vivado User and Methodology Guides, see the Vivado Design Suite - 2012.4 User Guides web page.

Parameter Changes in the XCI File

`Enable_Temp_Bus` parameter added for enabling 12-bit `TEMP_OUT` port.

Port Changes

`TEMP_OUT` added for 12-bit temperature data.

Functionality Changes

Temperature update logic added to support the systems using MIG and XADC Wizard.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools. In addition, this appendix provides a step-by-step process for debugging process and a flow diagram to guide you through debugging the XADC Wizard core.

The following topics are included in this appendix:

Finding Help on Xilinx.com

To help in the design and debug process when using the XADC Wizard, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for opening a Technical Support WebCase.

Documentation

This product guide is the main document associated with the XADC Wizard. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page (www.xilinx.com/support) or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page (www.xilinx.com/download). For more information about this tool and the features available, open the online help after installation.

Release Notes

Known issues for all cores, including the XADC Wizard are described in the IP Release Notes Guide (XTP025).

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.

Known Issues

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core are listed below, and can also be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as:

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Answer Records for the XADC Wizard

For the Answer Record, see http://xkb/pages/52088.aspx.

Contacting Technical Support

Xilinx provides premier technical support for customers encountering issues that require additional assistance.

To contact Xilinx Technical Support:

1. Navigate to www.xilinx.com/support.
2. Open a WebCase by selecting the WebCase link located under Support Quick Links.

When opening a WebCase, include:

- Target FPGA including package and speed grade.

- All applicable Xilinx Design Tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file(s) to include with the WebCase.

Debug Tools

There are many tools available to address XADC Wizard design issues. It is important to know which tools are useful for debugging various situations.

Example Design

The XADC Wizard is delivered with an example design that can be synthesized, complete with functional test benches. Information about the example design can be found in *Chapter 6, Example Design*.

ChipScope Pro Tool

The ChipScope™ Pro tool inserts logic analyzer, bus analyzer, and virtual I/O cores directly into your design. The ChipScope Pro tool allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed through the ChipScope Pro Logic Analyzer tool. For detailed information for using the ChipScope Pro tool, see www.xilinx.com/tools/cspro.htm.

Vivado Lab Tools

Vivado Lab Tools inserts logic analyzer, bus analyzer, and virtual I/O cores directly into your design. Vivado Lab Tools allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed.

Reference Boards

Various Xilinx development boards support XADC Wizard. These boards can be used to prototype designs and establish that the core can communicate with the system.

- 7 series evaluation boards
  - KC705
  - KC724

License Checkers

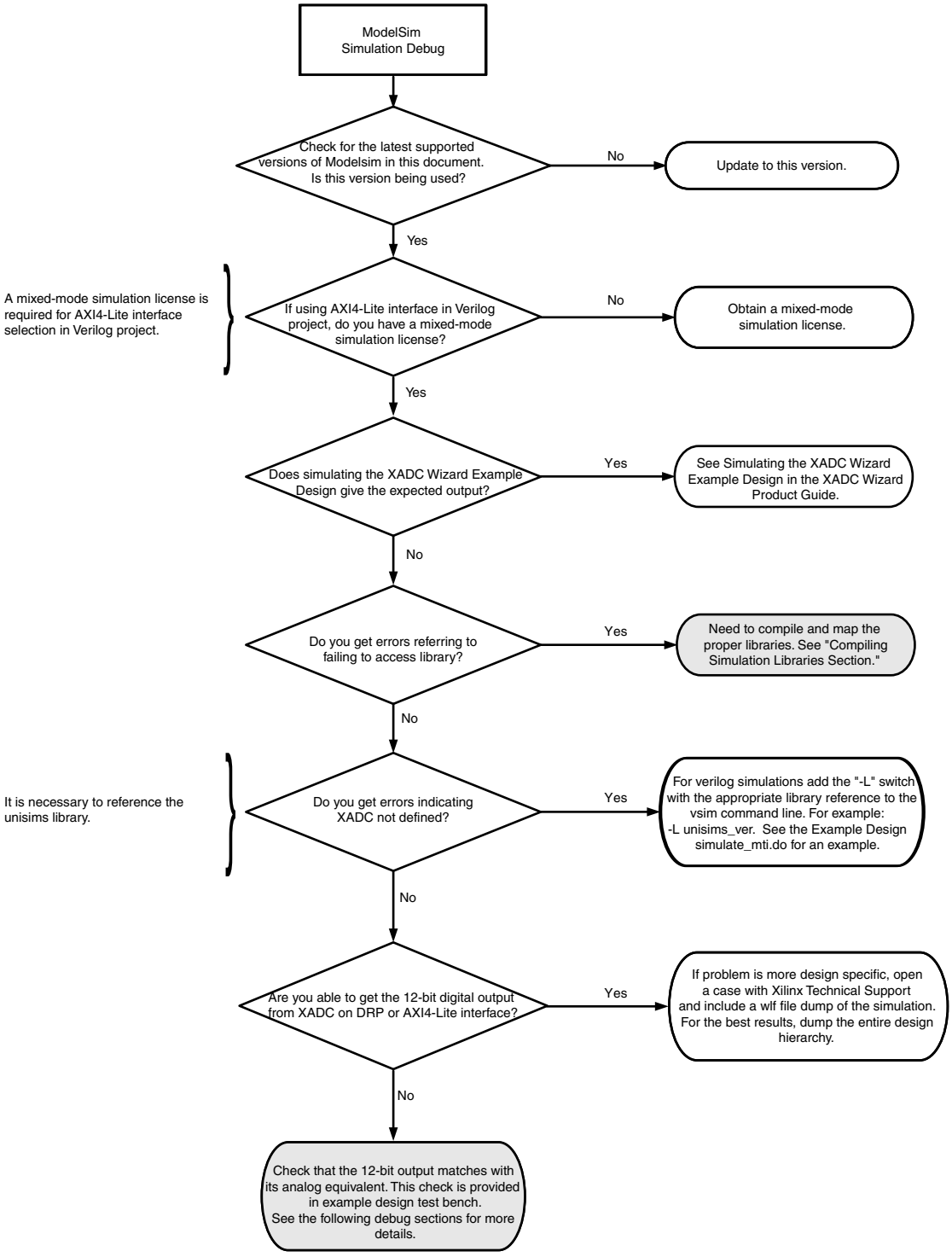
If the IP requires a license key, the key must be verified. The ISE and Vivado tool flows have a number of license check points for gating licensed IP through the flow. If the license check succeeds the IP may continue generation, otherwise generation halts with error. License checkpoints are enforced by the following tools:

- ISE flow: XST, NgdBuild, Bitgen
- Vivado flow: Vivado Synthesis, Vivado Implementation, write_bitstream (Tcl command)

IMPORTANT: *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

Simulation Debug

The simulation debug flow for ModelSim is illustrated below. A similar approach can be used with other simulators.


ModelSim
Simulation Debug
Check for the latest supported versions of Modelsim in this document. Is this version being used?
No
Update to this version.
Yes
A mixed-mode simulation license is required for AXI4-Lite interface selection in Verilog project.
If using AXI4-Lite interface in Verilog project, do you have a mixed-mode simulation license?
No
Obtain a mixed-mode simulation license.
Yes
Does simulating the XADC Wizard Example Design give the expected output?
Yes
See Simulating the XADC Wizard Example Design in the XADC Wizard Product Guide.
No
Do you get errors referring to failing to access library?
Yes
Need to compile and map the proper libraries. See "Compiling Simulation Libraries Section."
No
It is necessary to reference the unisims library.
Do you get errors indicating XADC not defined?
Yes
For verilog simulations add the "-L" switch with the appropriate library reference to the vsim command line. For example: -L unisims_ver. See the Example Design simulate_mti.do for an example.
No
Are you able to get the 12-bit digital output from XADC on DRP or AXI4-Lite interface?
Yes
If problem is more design specific, open a case with Xilinx Technical Support and include a wlf file dump of the simulation. For the best results, dump the entire design hierarchy.
No
Check that the 12-bit output matches with its analog equivalent. This check is provided in example design test bench. See the following debug sections for more details.

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The ChipScope tool is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the ChipScope tool for debugging the specific problems.

Many of these common issues can also be applied to debugging design simulations. Details are provided in the General Checks section.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the LOCKED port.
- If your outputs go to 0, check your licensing.

Interface Debug

AXI4-Lite Interfaces

Read from a register that does not have all 0s as a default to verify that the interface is functional. Output `s_axi_arready` asserts when the read address is valid, and output `s_axi_rvalid` asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The `S_AXI_ACLK and ACLK` inputs are connected and toggling.
- The interface is not being held in reset, and `S_AXI_ARESET` is an active-Low reset.
- The interface is enabled, and `s_axi_aclken` is active-High (if used).
- Ensure that the main core clocks are toggling and that the enables are also asserted.
- If the simulation has been run, verify in simulation and/or a ChipScope tool capture that the waveform is correct for accessing the AXI4-Lite interface.

Appendix C

Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Appendix B, Debugging and Xilinx Support website at:

www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:

www.xilinx.com/company/terms.htm.

References

Unless otherwise noted, IP references are for the product documentation page. These documents provide supplemental material useful with this product guide:

1. *7 Series FPGAs Overview* (DS180)
2. *7 Series FPGAs XADC User Guide* (UG480)
3. XADC Wizard Release Notes
4. *LogiCORE IP XADC Wizard User Guide* (UG772)
5. *LogiCORE IP AXI Lite IPIF (axi_lite_ipif) Data Sheet* (DS765)
6. *Vivado Design Suite Migration Methodology Guide* (UG911)
7. Vivado Design Suite user documentation

Technical Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the

documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled DO NOT MODIFY.

See the IP Release Notes Guide (XTP025) for more information on this core. For each core, there is a master Answer Record that contains the Release Notes and Known Issues list for the core being used. The following information is listed for each version of the core:

- New Features
- Resolved Issues
- Known Issues

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/16/12	1.0	• Initial Xilinx release of Product Guide and replaces UG772 • Added Vivado and Zynq-7000 information in IP Facts Table • Added support for AXI4-Lite interface • Added Fig. 1-1 block diagram and Vivado GUIs throughout book • Added Product Specification and Designing with the Core sections • Added Section II: Vivado Design Suite and Section III: Appendices
12/18/12	2.0	• Updated core v2.4 and Vivado Design Suite v2012.4. Removed Zynq references. • Updated block diagram in Fig. 1-1. • GUIs updated to v2.4. • Updated Table 2-1: 7 Series Devices Resource Estimates. • Added TEMP_OUT[11:0] to Table 2-2. • Updated description to “configuring HDL” in Register Space. • Updated CONVST Register (CONVSTR) Bits[17:1] • Added G6 C_HAS_TEMP_BUS to Table 3-1. • Added temp_rd_arbiter.vhd, drp_arbiter.vhd, and drp_rdwr_fsm.vhd to Table 6-2. • Updated Migration section. • Added new Debug section.

Notice of Disclaimer