AXI Universal Serial Bus (USB) 2.0 Device v5.0

LogiCORE IP Product Guide

Vivado Design Suite

PG137 November 18, 2015

XILINX
ALL PROGRAMMABLE™

Table of Contents

Introduction

The AXI USB 2.0 Device core provides a USB interface to a user design. This interface is suitable for USB-centric, high-performance designs, bridges, and legacy port replacement operations.

Features

- AXI interface based on AXI4 specification.
- Burst lengths of 1 to 256 beats with INCR type transfers.
- Optional DMA mode to increase throughput during the data transfers.
- Optional feature support for unaligned transfers on the master interface.
- Supports High Speed and Full Speed USB 2.0 specification.
- Supports high speed, high bandwidth isochronous transactions.
- Supports up to eight endpoints, including one control endpoint 0. Endpoints 1 to 7 can be bulk, interrupt, or isochronous. Endpoints are individually configurable.
- Two ping-pong buffers for each endpoint except for endpoint 0.
- Optional USB error detection, updates error count, and generates error interrupt.
- Supports ULPI (Universal Transceiver Macrocell Interface (UTMI) + Low Pin Interface) to an external USB PHY.
- Access to ULPI PHY registers and parameterized ULPI PHY reset.
- Resume and reset detection in low-power mode.
- Resuming of host from low-power mode with remote wake-up signaling.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale+™ Families, UltraScale™ Architecture, Zynq®-7000 All Programmable SoC, Virtex®-7, Kintex®-7, Artix®-7
Supported User Interfaces	AXI4, ULPI
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Encrypted RTL
Example Design	Verilog and VHDL
Test Bench	Provided
Constraints File	Xilinx Design Constraints (XDC)
Simulation Model	Not Provided
Supported S/W Driver[2]	Standalone and Linux
Tested Design Flows[3]	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. Standalone driver details can be found in the SDK directory (<*install_directory*>/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from the Xilinx Wiki page.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The USB 2.0 protocol multiplexes many devices over a single, half-duplex, serial bus. The bus runs at 480 Mb/s (High Speed) or at 12 Mb/s (Full Speed) and is designed to be plug-and-play. The host always controls the bus and sends tokens to each device specifying the required action. Each device has an address on the USB 2.0 bus and has one or more endpoints that are sources or sinks of data. All devices have the system control endpoint (endpoint 0).

Functional Description

The AXI USB 2.0 Device has eight endpoints: one control endpoint (endpoint 0) and seven user endpoints. Endpoint 0 has different requirements compared to the seven user endpoints. Endpoint 0 handles control transactions only, which start with an 8-byte setup packet and are then followed by zero or more data packets. The setup packet is always stored in a dedicated location in the Dual Port Random Access Memory (DPRAM) at an address offset of 0x80. When a setup packet is received, the SETUPPkt bit of the Interrupt Status Register (ISR) is set. Data packets are a maximum of 64 bytes. These data packets are stored in a single bidirectional data buffer set up by the configuration memory of Endpoint 0, located at the address offset 0x0 in the DPRAM. When a data packet is transmitted or received successfully, the FIFOBufFree and FIFOBufRdy bits of the Interrupt Status Register (ISR) are set respectively.

The seven user endpoints of the AXI USB 2.0 Device can be configured as bulk, interrupt, or isochronous. In addition, endpoints can be configured as INPUT (to the host) or OUTPUT (from the host). Each of these endpoints has two ping-pong buffers of the same size for endpoint data. The user endpoints data buffers are unidirectional, and are configured by the Endpoint Configuration and Status register of the respective endpoint. The size of the buffers can be configured from 0 to 512 bytes for bulk, 64 bytes for interrupt, and up to 1,024 bytes for isochronous endpoints.

The AXI USB 2.0 Device core with the AXI and ULPI interfaces is shown in Figure 1-1.

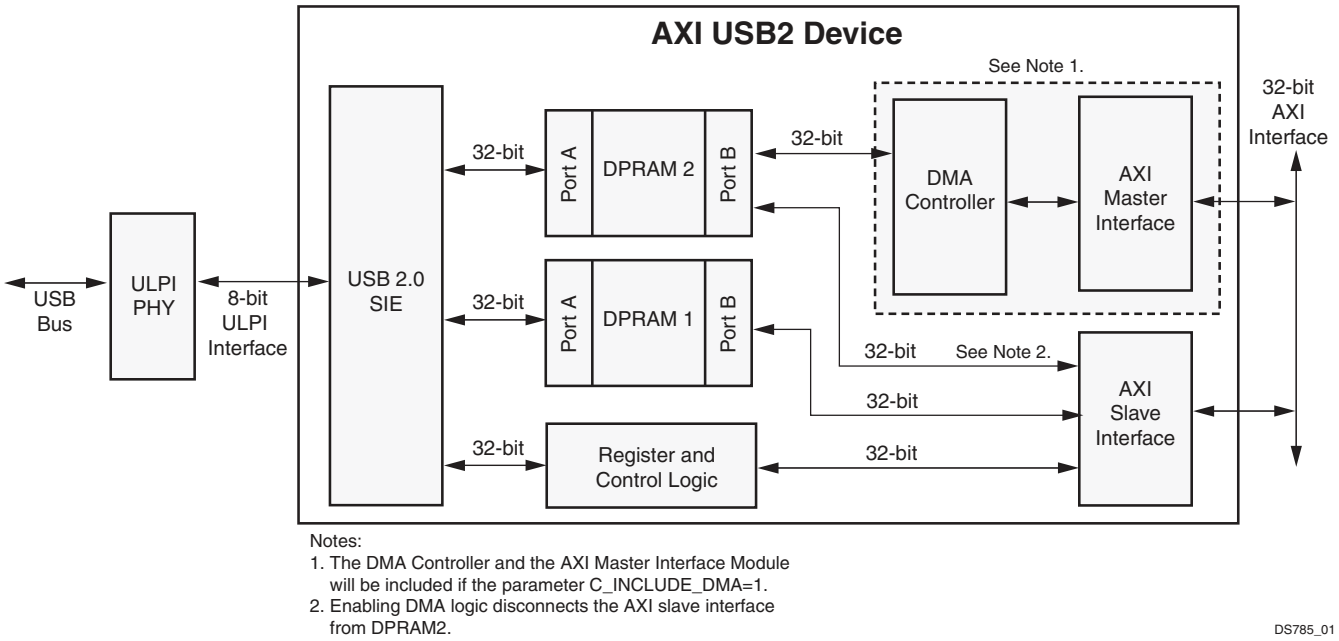

Figure 1-1: **AXI USB 2.0 Device with AXI and ULPI Interfaces**

When the host attempts to send data to an endpoint of the device, it sends a token which consists of an OUT PID along with the address of the device and the endpoint number, followed by the data. The device uses handshake packets to report the status of the data transaction if, and only if, the token and data are received without any USB errors, such as Bit Stuff, PID, and CRC errors. Handshake packets indicate successful reception of data, flow control, and halt conditions.

To receive data, the host sends a token which consists of an IN PID along with the device address and endpoint number, then waits for data from the device. The device responds with the requested data to the host if the token is received without any USB errors. The device then waits for the handshake packet.

Register and Control Logic

The AXI USB 2.0 Device core includes 32-bit registers, which provide control and status information for the core. These registers are accessible from the AXI Slave interface. For more information on the registers, see Register Description in Chapter 2.

USB 2.0 Serial Interface Engine (SIE)

The USB 2.0 Serial Interface Engine (SIE) handles the serialization and de-serialization of USB traffic at the byte level and the multiplexing and demultiplexing of USB data to and from the endpoints of the core. The SIE also handles USB 2.0 state transitions, such as suspend, resume, USB reset, and remote wake-up signaling (to wake up the host from the suspend mode).

The SIE interfaces to the PHY using a ULPI interface that requires 12 pins. Data to the FPGA from the USB is received from the PHY, error checked, and loaded into the appropriate area of the DPRAM. Data from the FPGA that is to be sent over the USB is loaded from the DPRAM, protocol wrapped and then when the protocol allows, presented to the PHY, 1-byte at a time. Details of the ULPI interface are beyond the scope of this document.

If any packet has USB errors, the SIE ignores and discards it. The SIE increments the respective USB error count in the Error Count Register (ECR) and sets the respective USB error bits in the Interrupt Status Register (ISR).

The status of the current USB transactions is signaled by the SIE to the Interrupt Status Register (ISR). Certain conditions can be enabled through the IER to generate an interrupt.

Control of the SIE comes from four sources:

1. The lower 64 bytes of the DPRAM contain the control and status locations for each endpoint.
2. The Control and Status registers provide overall start and stop, status indication, enabling of interrupts using status register bits, address control on USB, current Start of Frame timing information, and endpoint Buffer Ready indication.
3. The logic of the SIE module is coded to reflect the requirements of Chapter 8 of the *USB 2.0 Specification*.
4. The USB physical bus is connected to the SIE over the ULPI PHY interface. The SIE natively implements the ULPI protocol.

Dual Port Block RAM (DPRAM)

The DPRAM is the data storage area between the SIE and AXI interface. Port A of the DPRAM is used by the SIE and Port B is used by the Processor/DMA Controller. Both ports are 32-bits wide. The AXI USB 2.0 Device uses two sets of four Block RAMs implemented as 64 × 8 bits (DPRAM1) and 2 K × 8 bits each (DPRAM2), with dual asynchronous clock ports. Data from the USB 2.0 Device is routed to the DPRAM by the SIE through Port A. The firmware or hardware being used accesses the data through Port B over the AXI Slave Interface. Data being input to the AXI USB 2.0 Device is loaded through the AXI Slave Interface to Port B, into appropriate locations in the DPRAM. When the host requests data from the device, the SIE accesses this data from Port A.

The DPRAM is seen by the SIE as DPRAM1 and DPRAM2. DPRAM2 has seven endpoint FIFOs for endpoint 1 to 7. DPRAM1 has the endpoint 0 FIFO and the control register area which defines how the memory is arranged and reports the status of each FIFO buffer (ready, not ready or count). Each FIFO is double buffered to help support the high throughput possible with USB 2.0. One buffer can be used for a current USB transaction, while the other buffer is available to the user application for processing. The SIE treats the storage area as FIFOs. The firmware or hardware can access the storage as ordinary RAM over the AXI Slave Interface.

AXI Slave Interface

The AXI Slave interface in the core performs the following operations:

- Responds to AXI transactions to read from or write to the 32-bit control registers, status registers, and DPRAM.
- Supports byte, half word, and word transfers for the DPRAM, but only word transfers are supported for the registers.

AXI Master Interface

The AXI Master interface in the core performs the following operations:

- Performs read and write transactions as a AXI master in DMA mode.
- In DMA mode, interrupts are generated based on DMA Done and DMA Error conditions.

Direct Memory Access (DMA) Controller

The DMA controller is included in the AXI USB 2.0 Device core if the "Enable DMA Support" parameter is set to 1. The DMA controller provides simple DMA services between DPRAM2 and an external memory device or peripheral on the AXI interface. The DMA controller transfers data from a source address to a destination address without processor intervention for a given length.

It provides programmable registers (for both directions read from and write to DPRAM2), source address, destination address, and transfer length. The source and destination address counters are increment only. It also supports AXI burst transfers. The DMA controller in the core performs data transfers between DPRAM2K and external memory only.

In a DMA transfer, the DPRAM2 address should always be a word-aligned address. A user-configurable parameter "Enable unaligned transactions" is defined to configure the support of un-aligned external memory addresses. When the parameter "Enable unaligned transactions" is set to 1, the DMA controller takes care of the data re-alignment and the DMA source or destination registers can be un-aligned addresses depending on the direction bit in the DMA control register. When "Enable unaligned transactions" is set to 0, the DMA controller supports only aligned addresses.

ULPI PHY

The USB PHY can be any ULPI-compliant PHY. The primary function of the PHY is to manage the bit-level serialization and de-serialization of USB 2.0 traffic. To do so, it must detect and recover the USB clock. The clock runs at 480 MHz, a speed that is too fast for practical implementation on the FPGA. Because 480 MHz is also too fast for the USB SIE clock, the

PHY interfaces to the SIE on a byte serial basis and generates a 60 MHz clock which runs on the SIE side of the USB 2.0 Device.

Applications

The AXI USB 2.0 Device core supports interrupt, bulk and isochronous transactions. The core can be used in applications such as:

- USB-enabled keyboard or mouse devices
- Mass storage devices
- Microphone and speaker device applications

Unsupported Features

The following features are not supported in this core.

- Host functionality
- USB On-The-Go (OTG)

Licensing and Ordering Information

License Checkers

If the IP requires a license key, the key must be verified. The Vivado design tools have several license check points for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with error. License checkpoints are enforced by the following Vivado design tools.

- 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.*

License Type

This Xilinx® LogiCORE™ IP module is provided under the terms of the Xilinx Core License Agreement. The module is shipped as part of the Vivado® Design Suite. For full access to all core functionalities in simulation and in hardware, you must purchase a license for the core. Contact your local Xilinx sales representative for information on pricing and availability.

For more information, visit the AXI USB 2.0 Device product web page.

Information about 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

Standards

The AXI USB 2.0 Device core supports the following industry standards at the interface level.

- AXI4 Full/lite interface
- ULPI interface

This core supports the AXI4-Lite interface for register access and AXI4 interface for DMA operations. The ULPI interface is used for PHY connectivity.

Performance

The AXI USB 2.0 Device core is characterized according to the benchmarking methodology described in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1]. Table 2-1 shows the results of the characterization runs.

Table 2-1: **Maximum Frequencies**

Family	Speed Grade	F_{max} (MHz)	
		AXI4-Lite	AXI4
Virtex-7	-1	160	160
	-2	200	200
	-3	220	220
Kintex-7	-1	160	160
	-2	200	200
	-3	220	220
Artix-7	-1	150	150
	-2	180	180
	-3	200	200

The target F_{Max} is influenced by the exact system and is provided for guidance. It is not a guaranteed value across all systems.

Note: Characterization results for Zynq-7000 devices are similar to the results for the devices listed in Table 2-1.

Throughput

To measure the system throughput of the AXI USB 2.0 Device core, a Kintex®-7 system is used similar to Figure 1-1 and tested on a Xilinx KC705 board.

For this test, the firmware running on the KC705 evaluation board configured as a mass storage device or isochronous device with different applications. A Windows application running on a Windows Desktop PC sets up the mass storage device for the throughput test on its BULK IN endpoint. The Windows application has an option of selecting the amount of traffic being sent on the bus for the throughput test. When the option was selected, the Windows application sets up the mass storage device to send the selected amount of data. Upon completion of the data transfer, the result is displayed on the Windows screen in MB/s. The throughput is measured using the amount of data sent over the bus and the time taken for transmission. A LeCroy USB analyzer is used to capture the transactions on the bus. The throughput results are based on the number of USB data packets received between two SOF packets. The throughput numbers are shown in Table 2-2 and Table 2-3.

Table 2-2: **AXI DMA System Performance for Bulk Transactions**

Throughput in the Functional Simulations	Throughput on the KC705 Board
12 packets per micro frame = 49,152,000 B/s	11 packets per micro frame = 45,056,000 B/s

Note: In BULK IN mode, the performance of the device depends on the number of data transfer packets that the host application can initiate between the two SOFs. The throughput differs with the OS capability and load.

Table 2-3: **AXI DMA System Performance for Isochronous Transactions**

Throughput in the Functional Simulations	Throughput on KC705 Board
3 packets per micro frame = 24 MB/s	~3 packets per micro frame = 23 MB/s

Resource Utilization

For details about Resource Utilization, visit Performance and Resources Utilization web page

Port Descriptions

A description of the I/O signals for the AXI USB 2.0 Device core is provided in Table 2-4.

Table 2-4: **AXI USB I/O Signal Description**

Signal Name	Interface	I/O	Initial State	Description
m_axi_aclk	Clock	I	NA	AXI Master Clock
m_axi_aresetn	Reset	I	NA	AXI Master Reset, active-Low
s_axi_aclk	Clock	I	NA	AXI Slave Clock
s_axi_aresetn	Reset	I	NA	AXI Slave Reset, active-Low
m_axi_*	M_AXI	NA	NA	AXI4 master interface signals for read and write channels. The m_axi interface is only available when DMA support is enabled(2).
s_axi_*	S_AXI	NA	NA	AXI4-Lite Slave Interface signals(2)
usb_irpt	System	O	0	USB Interrupt
ulpi_clock	ULPI	I	NA	All USB protocol interface signals are synchronous to this clock
ulpi_dir	ULPI	I	NA	Direction of data flow between the PHY and SIE
ulpi_next	ULPI	I	NA	Indicator of when the PHY is ready for the next bit
ulpi_stop	ULPI	O	0	Indicator that transmission of last byte is complete
ulpi_data_i[7:0]	ULPI	I	NA	Input data from the PHY to the SIE
ulpi_reset(2)	ULPI	O	1	Reset to the ULPI PHY
ulpi_data_o[7:0]	ULPI	O	0	Output data from the SIE to the PHY
ulpi_data_t	ULPI	O	0	ULPI_Data is a 3-state port with ULPI_Data_I as the IN port, ULPI_Data_O as the OUT port, and ULPI_Data_T as the 3-state output
configured	Debug (Optional)	O	0	Used for USB 2.0 Certification – Test mode NAK
spare1	Debug (Optional)	O	0	Used for USB 2.0 Certification – Test mode J

Table 2-4: **AXI USB I/O Signal Description *(Cont'd)***

Signal Name	Interface	I/O	Initial State	Description
spare2	Debug (Optional)	O	0	Used for USB 2.0 Certification – Test mode K
vbus_detect	Debug (Optional)	O	0	0: Valid VBUS has not been detected 1: Valid VBUS has been detected
show_currentspeed	Debug (Optional)	O	0	0: Full Speed 1: High Speed
running	Debug (Optional)	O	0	0: The SIE is in reset state and does not respond to USB traffic 1: The SIE is finished USB reset and is ready to respond to USB traffic
suspended	Debug (Optional)	O	0	0: AXI USB 2.0 Device core has not been suspended 1: AXI USB 2.0 Device core has been suspended
disconnected	Debug (Optional)	O	0	0: AXI USB 2.0 Device core is connected 1: AXI USB 2.0 Device is disconnected

1. See the *Vivado AXI Reference Guide* (UG1037) [Ref 2] for AXI4, AXI4-Lite and AXI4-Stream Signals

Register Space

Register Bit Ordering

All registers use little endian bit ordering where bit 31 is the MSB and bit 0 is the LSB. Table 2-5 shows the bit ordering.

Table 2-5: **Register Bit Ordering**

31	30	29	. . . . .	2	1	0
MSB			. . . . .			LSB

Register Description

The memory map for the AXI USB 2.0 Device core is shown in Table 2-6 which includes endpoint configuration space (offset 0x0000), setup packet storage space (offset 0x0080), RAM for endpoint 0 buffer (offset 0x0088), register space for the USB registers (offset 0x0100), and RAM for endpoint 1 to 7 buffers (offset 0x4000). Table 2-7 lists the mapping for endpoint configuration space. All offsets are word offsets.

Table 2-6: **Register Address Map**

Register Name	Base Address + Offset (hex)	Reset Value (hex)	Access[1][2]
Endpoint Configuration and Status Registers	C_BASEADDR + 0x0000	0x00000000	R/W
Setup Packet Storage Word 0[2]	C_BASEADDR + 0x0080	0x00000000	R
Setup Packet Storage Word 1[2]	C_BASEADDR + 0x0084	0x00000000	R
RAM for Endpoint 0 Buffer[6]	C_BASEADDR + 0x0088	0x00000000	R/W
USB Address Register	C_BASEADDR + 0x0100	0x00000000	R/W
Control Register	C_BASEADDR + 0x0104	0x00000000	R/W
Interrupt Status Register	C_BASEADDR + 0x0108	0x00000000	R
Frame Number Register	C_BASEADDR + 0x010C	0x00000000	R
Interrupt Enable Register	C_BASEADDR + 0x0110	0x00000000	R/W
Buffer Ready Register	C_BASEADDR + 0x0114	0x00000000	R/W
Test Mode Register	C_BASEADDR + 0x0118	0x00000000	R/W
Error Count Register[7]	C_BASEADDR + 0x011C	0x00000000	R
ULPI PHY Access Register[8]	C_BASEADDR + 0x0120	0x00000000	R/W
DMA Software Reset Register[3][4]	C_BASEADDR + 0x0200	0x00000000	W
DMA Control Register[4]	C_BASEADDR + 0x0204	0x00000000	R/W
DMA Source Address Register[10]	C_BASEADDR + 0x0208	0x00000000	R/W
DMA Source Address Register (LSB 32 bit)[9]	C_BASEADDR + 0x0308	0x00000000	R/W
DMA Source Address Register (MSB 32 bit)[9]	C_BASEADDR + 0x030C	0x00000000	R/W
DMA Destination Address Register[10]	C_BASEADDR + 0x020C	0x00000000	R/W
DMA Destination Address Register (LSB 32 bit)[9]	C_BASEADDR + 0x0310	0x00000000	R/W
DMA Destination Address Register (MSB 32 bit) [9]	C_BASEADDR + 0x0314	0x00000000	R/W
DMA Length Register[4]	C_BASEADDR + 0x0210	0x00000000	R/W
DMA Status Register[4]	C_BASEADDR + 0x0214	0x00000000	R/W

Table 2-6: **Register Address Map *(Cont'd)***

Register Name	Base Address + Offset (hex)	Reset Value (hex)	Access[1][2]
RAM for Endpoint 1–7 Buffers [6]	C_BASEADDR + 0x4000	0x00000000	R/W

Notes:

1. R/W – Read and Write access
2. R – Read Only access. Write to these registers has no effect
3. W – Write Only access. Reading of these registers returns zero
4. This register is included in the design if the parameter C_INCLUDE_DMA = 1
5. RAM for endpoint 0 buffer should be 64 bytes, EpBase_ADDRESS (endpoint 0)+ 0x40 should not exceed 0x00FF and master address width is 32.
6. RAM for endpoint 1–7 buffers range (C_BASEADDR + 0x4000) to (C_BASEADDR + 0x4000) + 0x1FFF. When the DMA logic is enabled the RAM for endpoint 1–7 buffers cannot be accessed through the AXI slave interface; It is only accessible through the DMA controller.
7. This register is not included in the design if the parameter C_INCLUDE_USBERR_LOGIC = 0. But the SIE performs USB error checks as part of the USB 2.0 specification and ignores any error packets received.
8. This register is defined to access the ULPI PHY registers.
9. This register is valid only when dma is enabled and master address width is greater than 32.
10. This register is included in the design if the parameter C_INCLUDE_DMA = 1 and master address width is equal to 32.

Endpoint Configuration and Status Registers

The Endpoint Configuration and Status registers control the operational characteristics of each endpoint and reports its current condition. The total endpoint configuration register space is divided between the eight endpoints of the USB 2.0 Device as shown in Table 2-7.

Table 2-7: **Endpoint Configuration Registers (*C_BASEADDR + Address Offset*)**

Address Offset	Memory/Register Space
0x0000	Endpoint 0
0x0010	Endpoint 1
0x0020	Endpoint 2
0x0030	Endpoint 3
0x0040	Endpoint 4
0x0050	Endpoint 5
0x0060	Endpoint 6
0x0070	Endpoint 7

Each endpoint has four 32-bit registers that describe the behavior of the endpoint. These registers are located sequentially and arranged by endpoint number as shown in Table 2-8.

Table 2-8: **Endpoint Configuration Register**

Address Offset	Memory/Register Space
0x0000	Endpoint configuration and status register
0x0004	Reserved
0x0008	Buffer 0 count: 0 to 1,024
0x000C	Buffer 1 count: 0 to 1,024

The bit description for the Endpoint Configuration and Status registers is given in Table 2-9. All the bits of this register can be modified by the firmware. Under normal operation, some of the bits are modified by the USB SIE itself, and only their initial values need to be set by the firmware.

Table 2-9: **Endpoint Configuration and Status Register (*C_BASEADDR + 0x0000*)**

Bits	Name[(1)(2)]	Access	Reset Value	Description
31	EpValid[(3)]	R/W	0	0: Disables the endpoint 1: Enables the endpoint
30	EpStall	R/W	0	0: Endpoint accepts INs and OUTs 1: Endpoint responds to the host only with a STALL
29	EpOutIn	R/W	0	0: Core receives data from the host (OUT from host) 1: Core sends data to the host (IN to host)

Table 2-9: **Endpoint Configuration and Status Register (*C_BASEADDR + 0x0000*) *(Cont'd)***

Bits	Name[1][2]	Access	Reset Value	Description
28	EpIso[4]	R/W	0	0: Not an isochronous endpoint 1: Isochronous endpoint
27	EpDataTgl[5]	R/W	0	0: Next DATA packet must be a DATA0 packet 1: Next DATA packet must be a DATA1 packet
26	EpBufSel[6]	R/W	0	0: Buffer 0 is used 1: Buffer 1 is used
25–15	EpPktSze[7]	R/W	0	Endpoint packet size
14–13	EpIsoTrans	R/W	0	Number of Isochronous transfers supported in a (micro)frame by the endpoint. 00: Supports one isochronous transaction 01: Supports two isochronous transactions 10: Supports three isochronous transactions 11: Reserved
12–0	EpBase[8]	R/W	0	Base offset of the buffers in the DPRAM

Notes:

1. The VALID, STALL, OUT_IN, and ISO bits are set by the firmware to define how the endpoint operates. For example, to set up the endpoint to receive bulk OUTs from the host, set EpValid = 1, EpOutIn = 0, EpStall = 0, and EpIso = 0.
2. The EpDataTgl bit is toggled automatically by the SIE during the data stage of a transfer for the DATA0/DATA1 synchronization that is required. This bit can also be explicitly set when needed, for example during the status stage of a control transfer. The EpBufSel bit is also automatically modified by the SIE; the firmware only needs to set its initial value.
3. EpValid is an Master Enable bit for the respective endpoint.
4. EpIso enables endpoint as an isochronous endpoint.
5. The SIE uses EpDataTgl to detect DATA PID toggle errors during the data transfers and its weak form of synchronization technique. This bit can be explicitly set in response to a firmware command.
6. Used to support ping-pong buffers during data transfers. If this bit is set then the SIE toggles the buffer access during data transfers, one at each time.
7. EpPktSze refers to maximum packet size limited to the respective endpoint.
8. EpBase is word addressable and the endpoint buffer address should always be right shifted by 2 before writing EpBase.

Buffer Count Register0 (BCR0)

The Buffer 0 Count registers shown in Table 2-10, indicate the size of data in bytes. If the endpoint is an OUT endpoint, then the SIE sets the value of this register at the end of a successful reception from the host. If the endpoint is an IN endpoint, then the firmware sets the value of this register before transmission.

These registers are 32-bit wide and have R/W access.

Table 2-10: **Buffer Count Register0 (*C_BASEADDR + 0x0008*)**

Bits	Name	Access	Reset Value	Description
31–11	Reserved	NA	-	Reserved
10–0	OutInPktCnt	R/W	0	Packet count in the buffer

Buffer Count Register1 (BCR1)

The Buffer 1 Count registers shown in Table 2-11 indicate the size of data in bytes. If the endpoint is an OUT endpoint, the SIE sets the value of this register at the end of a successful reception from the host. If the endpoint is an IN endpoint, the firmware sets the value of this register before transmission. These registers are 32-bit wide and have R/W access.

Table 2-11: **Buffer Count Register1 (*C_BASEADDR + 0x000C*)**

Bits	Name	Access	Reset Value	Description
31–11	Reserved	NA	-	Reserved
10–0	OutInPktCnt	R/W	0	Packet count in the buffer

USB Address Register (UAR)

The USB Address register, shown in Table 2-12 contains the host-assigned USB address of the device. There are 128 possible USB devices on the USB; therefore, the register takes values from 0 to 127. The lower seven bits of the register [6:0] are used to set the address. An address of 0 indicates that the device is un-enumerated. Address 0 is the default address of all USB devices at plug-in time and the address value on hardware reset. This register is 32-bits wide and has R/W access.

Table 2-12: **USB Address Register** *(C_BASEADDR + 0x0100)*

Bits	Name	Access	Reset Value	Description
31–7	Reserved	NA	-	Reserved
6–0	USBAddr	R/W	0	Indicates the USB address of the device

Control Register (CR)

As shown in Table 2-13, the MstRdy bit indicates SIE operation. When clear, the USB SIE is paused and does not respond to any USB activity. When set, the SIE operates normally. The RmteWkup bit initiates remote wake-up signaling to the host when the device has been suspended by the host. The core generates a USBRsm interrupt after the successful completion of remote wake-up signaling with the host. The RmteWkup bit remains set until the firmware clears it.

Table 2-13: **Control Register** *(C_BASEADDR + 0x0104)*

Bits	Name	Access	Reset Value	Description
31	MstRdy	R/W	0	0: SIE is paused and does not respond to any USB activity 1: SIE operates normally
30	RmteWkup	R/W	0	0: SIE does nothing 1: SIE sends remote wake-up signaling to host

Table 2-13: **Control Register** *(C_BASEADDR + 0x0104)* ***(Cont'd)***

Bits	Name	Access	Reset Value	Description
29	SIE_SoftRst	R/W	0	0: Normal operation 1: SIE gets reset and starts from PHY re-initialization. This bit resets the Address and Buffer Ready Registers.
28–0	Reserved	R	-	Reserved

Interrupt Status Register (ISR)

The Interrupt Status Register (ISR) shown in Table 2-14, reports status on the operation of the AXI USB 2.0 Device core. The bits in this register are cleared as soon as they are read.

Table 2-14: **Interrupt Status Register** *(C_BASEADDR + 0x0108)*

Bits	Name	Access	Reset Value	Description
31	Reserved	NA	-	Reserved
30	ULPIAccComp	R	0	0: ULPI PHY Access Complete has not occurred 1: ULPI PHY Access Complete has occurred
29	BitStufErr(2)	R	0	0: Bit Stuff error has not occurred 1: Bit Stuff error has occurred
28	PidErr(2)	R	0	0: PID error has not occurred 1: PID error has occurred
27	CrcErr(2)	R	0	0: CRC error has not occurred 1: CRC error has occurred
26	DmaDne(1)	R	0	0: DMA operation is not done 1: DMA operation is done
25	DmaErr(1)	R	0	0: DMA error has not occurred 1: DMA error has occurred
24	USBRsm(3)	R	0	0: Core has not received resume signaling from host 1: Core received resume signaling from host
23	USBRst(3)	R	0	0: Core has not received USB reset from host 1: Core received USB reset from host
22	USBSpnd(3)	R	0	0: Core not in suspend state 1: Core in suspend state
21	USBDsc(3)	R	0	0: Core connected to host 1: Core is disconnected from host
20	FIFOBufRdy	R	0	0: Endpoint 0 packet has not been received by core 1: Endpoint 0 packet has been received by core
19	FIFOBufFree	R	0	0: Endpoint 0 packet has not been transmitted by core 1: Endpoint 0 packet has been transmitted by core This bit is valid only when bit 0 in the ISR is set.

Table 2-14: **Interrupt Status Register** *(C_BASEADDR + 0x0108)* ***(Cont'd)***

Bits	Name	Access	Reset Value	Description
18	SETUPPkt	R	0	0: Endpoint 0 Setup packet has not been received by core 1: Endpoint 0 Setup packet has been received
17	SOFPkt	R	0	0: Start of Frame packet has not been received by core 1: Start of Frame packet has been received by core
16	HighSpd	R	0	0: Core is running at Full Speed 1: Core is running at High Speed
15	Ep7ProcBuf1	R	0	0: Endpoint 7, buffer 1 not processed 1: Endpoint 7, buffer 1 processed
14	Ep6ProcBuf1	R	0	0: Endpoint 6, buffer 1 not processed 1: Endpoint 6, buffer 1 processed
13	Ep5ProcBuf1	R	0	0: Endpoint 5, buffer 1 not processed 1: Endpoint 5, buffer 1 processed
12	Ep4ProcBuf1	R	0	0: Endpoint 4, buffer 1 not processed 1: Endpoint 4, buffer 1 processed
11	Ep3ProcBuf1	R	0	0: Endpoint 3, buffer 1 not processed 1: Endpoint 3, buffer 1 processed
10	Ep2ProcBuf1	R	0	0: Endpoint 2, buffer 1 not processed 1: Endpoint 2, buffer 1 processed
9	Ep1ProcBuf1	R	0	0: Endpoint 1, buffer 1 not processed 1: Endpoint 1, buffer 1 processed
8	Reserved	NA	-	Reserved
7	Ep7ProcBuf0	R	0	0: Endpoint 7, buffer 0 not processed 1: Endpoint 7, buffer 0 processed
6	Ep6ProcBuf0	R	0	0: Endpoint 6, buffer 0 not processed 1: Endpoint 6, buffer 0 processed
5	Ep5ProcBuf0	R	0	0: Endpoint 5, buffer 0 not processed 1: Endpoint 5, buffer 0 processed
4	Ep4ProcBuf0	R	0	0: Endpoint 4, buffer 0 not processed 1: Endpoint 4, buffer 0 processed
3	Ep3ProcBuf0	R	0	0: Endpoint 3, buffer 0 not processed 1: Endpoint 3, buffer 0 processed
2	Ep2ProcBuf0	R	0	0: Endpoint 2, buffer 0 not processed 1: Endpoint 2, buffer 0 processed
1	Ep1ProcBuf0	R	0	0: Endpoint 1, buffer 0 not processed 1: Endpoint 1, buffer 0 processed

Table 2-14: **Interrupt Status Register** *(C_BASEADDR + 0x0108)* ***(Cont'd)***

Bits	Name	Access	Reset Value	Description
0	Ep0ProcBuf0	R	0	0: Endpoint 0, buffer 0 not processed 1: Endpoint 0, buffer 0 processed

Notes:

1. This bit is undefined if the parameter C_INCLUDE_DMA = 0.
2. This bit is undefined if the parameter C_INCLUDE_USBERR_LOGIC = 0.
3. This bit indicates the current status of the AXI USB 2.0 Device core.

The AXI USB 2.0 Device core has a single interrupt line (`USB_Irpt`) to indicate an interrupt. Interrupts are indicated by asserting the `USB_Irpt` signal (transition of the `USB_Irpt` from a logic 0 to a logic 1).

The Interrupt Enable Register (IER) allows specific bits of the Interrupt Status Register (ISR) to generate interrupts. The MstEnbl bit of the Interrupt Enable Register (IER) register allows all interrupts to be disabled simultaneously. The interrupt condition is cleared when the corresponding bit of the Interrupt Status Register (ISR) is cleared by writing a 1 to it. During power on, the `USB_Irpt` signal is driven Low.

The following two conditions cause the `USB_Irpt` signal to be asserted:

- If a bit in the ISR is 1 and the corresponding bit in the IER is 1.
- Changing an IER bit from a 0 to 1 when the corresponding bit in the ISR is already 1.

Two conditions cause the `USB_Irpt` signal to be deasserted:

- Clearing a bit in the ISR, that is, by reading the ISR, provided that the corresponding bit in the IER is 1.
- Changing an IER bit from 1 to 0, when the corresponding bit in the ISR is 1.

When both deassertion and assertion conditions occur simultaneously, the `USB_Irpt` signal is deasserted first, then is reasserted if the assertion condition remains TRUE.

Frame Number Register (FNR)

The Frame Number Register (FNR) shown in Table 2-15 is composed of two fields:

- Frame
- Microframe

Frames are sent once every 1 ms and denote the beginning of a USB frame. Host scheduling starts at the start of Frame Time.

CAUTION! *The Microframe field is the result of additional Start of Frame tokens, sent once every 125 µs. When the USB is operated in High Speed mode, this can generate a potentially high rate of interrupts. Therefore, the interrupt enable of Start of Frame should be used with caution.*

Frame count values are of 11 bits and Microframe count values are of 3 bits.

Table 2-15: **Frame Number Register** *(C_BASEADDR + 0x010C)*

Bits	Name	Access	Reset Value	Description
31–14	Reserved	NA	-	Reserved for future use
13–3	FrmNum	R	0	Frame numbers—0 to 2,047
2–0	uFrmNum	R	0	Microframe numbers—0 to 7

Interrupt Enable Register (IER)

The Interrupt Enable Register (IER) shown in Table 2-16 allows specific bits of the Interrupt Status Register (ISR) to generate interrupts. The MstEnbl bit of this register allows all interrupts to be disabled simultaneously. The interrupt condition is cleared when the corresponding bit of the Interrupt Status Register (ISR) is cleared. A specific bit of the IER can be cleared to prevent a long duration condition, such as USB reset, from continuously generating an interrupt.

Table 2-16: **Interrupt Enable Register** *(C_BASEADDR + 0x0110)*

Bits	Name	Access	Reset Value	Description
31	MstEnbl	R/W	0	0: Disables the setting of all other interrupts 1: Enables setting of all other interrupts
30	ULPIAccComp	R/W	0	0: Disables ULPI PHY Access Complete interrupt 1: Enables ULPI PHY Access Complete interrupt
29	BitStufErr(2)	R/W	0	0: Disables Bit Stuff error interrupt 1: Enable Bit Stuff error interrupt
28	PidErr(2)	R/W	0	0: Disables PID error interrupt 1: Enables PID error interrupt
27	CrcErr(2)	R/W	0	0: Disables CRC error interrupt 1: Enables CRC error interrupt
26	DmaDne(1)	R/W	0	0: Disables DMA Done interrupt 1: Enables DMA Done interrupt
25	DmaErr(1)	R/W	0	0: Disables DMA Error interrupt 1: Enables DMA Error interrupt
24	USBRsm	R/W	0	0: Disables USB Resume interrupt 1: Enables USB Resume interrupt
23	USBRst	R/W	0	0: Disables USB reset interrupt 1: Enables USB reset interrupt
22	USBSpnd	R/W	0	0: Disables USB Suspend interrupt 1: Enables USB Suspend interrupt
21	USBDsc	R/W	0	0: Disables USB Disconnect interrupt 1: Enables USB Disconnect interrupt
20	FIFOBufRdy	R/W	0	0: Disables FIFIOBufRdy interrupt 1: Enables FIFIOBufRdy interrupt

Table 2-16: **Interrupt Enable Register** *(C_BASEADDR + 0x0110)* ***(Cont'd)***

Bits	Name	Access	Reset Value	Description
19	FIFOBufFree	R/W	0	0: Disables FIFOBufFree interrupt 1: Enables FIFOBufFree interrupt
18	SETUPPkt	R/W	0	0: Disables Setup Packet received interrupt 1: Enables Setup Packet received interrupt
17	SOFPkt	R/W	0	0: Disables Start of Frame received interrupt 1: Enables Start of Frame received interrupt
16	HighSpd	R/W	0	0: Disables core operates in High Speed interrupt 1: Enables core operates in High Speed interrupt
15	Ep7ProcBuf1	R/W	0	0: Disables endpoint 7, buffer 1 processed interrupt 1: Enables endpoint 7, buffer 1 processed interrupt
14	Ep6ProcBuf1	R/W	0	0: Disables endpoint 6, buffer 1 processed interrupt 1: Enables endpoint 6, buffer 1 processed interrupt
13	Ep5ProcBuf1	R/W	0	0: Disables endpoint 5, buffer 1 processed interrupt 1: Enables endpoint 5, buffer 1 processed interrupt
12	Ep4ProcBuf1	R/W	0	0: Disables endpoint 4, buffer 1 processed interrupt 1: Enables endpoint 4, buffer 1 processed interrupt
11	Ep3ProcBuf1	R/W	0	0: Disables endpoint 3, buffer 1 processed interrupt 1: Enables endpoint 3, buffer 1 processed interrupt
10	Ep2ProcBuf1	R/W	0	0: Disables endpoint 2, buffer 1 processed interrupt 1: Enables endpoint 2, buffer 1 processed interrupt
9	Ep1ProcBuf1	R/W	0	0: Disables endpoint 1, buffer 1 processed interrupt 1: Enables endpoint 1, buffer 1 processed interrupt
8	Reserved	NA	-	Reserved
7	Ep7ProcBuf0	R/W	0	0: Disables endpoint 7, buffer 0 processed interrupt 1: Enables endpoint 7, buffer 0 processed interrupt
6	Ep6ProcBuf0	R/W	0	0: Disables endpoint 6, buffer 0 processed interrupt 1: Enables endpoint 6, buffer 0 processed interrupt
5	Ep5ProcBuf0	R/W	0	0: Disables endpoint 5, buffer 0 processed interrupt 1: Enables endpoint 5, buffer 0 processed interrupt
4	Ep4ProcBuf0	R/W	0	0: Disables endpoint 4, buffer 0 processed interrupt 1: Enables endpoint 4, buffer 0 processed interrupt
3	Ep3ProcBuf0	R/W	0	0: Disables endpoint 3, buffer 0 processed interrupt 1: Enables endpoint 3, buffer 0 processed interrupt
2	Ep2ProcBuf0	R/W	0	0: Disables endpoint 2, buffer 0 processed interrupt 1: Enables endpoint 2, buffer 0 processed interrupt
1	Ep1ProcBuf0	R/W	0	0: Disables endpoint 1, buffer 0 processed interrupt 1: Enables endpoint 1, buffer 0 processed interrupt
0	Ep0ProcBuf0	R/W	0	0: Disables endpoint 0, buffer 0 processed interrupt 1: Enables endpoint 0, buffer 0 processed interrupt

Notes:

1. This bit is undefined if the parameter C_INCLUDE_DMA = 0.
2. This bit is undefined if the parameter C_INCLUDE_USBERR_LOGIC = 0.

Buffer Ready Register (BRR)

The Buffer Ready Register (BRR) has a buffer ready bit corresponding to each buffer of each endpoint, as shown in Table 2-17. The firmware sets each bit when that buffer is ready for either USB IN or USB OUT traffic. Until that bit is set, an attempted IN or OUT to/from the buffer results in a NAK to the host. The ability of the buffer to handle an IN or OUT is determined by the EpOutIn bit in the Configuration and Status register of the corresponding endpoint. It should be noted per the *USB 2.0 Specification*, endpoint 0 has only one buffer that handles IN or OUT.

Table 2-17: **Buffer Ready Register** *(C_BASEADDR + 0x0114)*

Bits	Name	Access	Reset Value	Description
31–16	Reserved	R	0	Reserved for future use
15	Ep7Buf1Rdy	R/W	0	0: Endpoint 7, buffer 1 is not ready for SIE transfer 1: Endpoint 7, buffer 1 is ready for SIE transfer
14	Ep6Buf1Rdy	R/W	0	0: Endpoint 6, buffer 1 is not ready for SIE transfer 1: Endpoint 6, buffer 1 is ready for SIE transfer
13	Ep5Buf1Rdy	R/W	0	0: Endpoint 5, buffer 1 is not ready for SIE transfer 1: Endpoint 5, buffer 1 is ready for SIE transfer
12	Ep4Buf1Rdy	R/W	0	0: Endpoint 4, buffer 1 is not ready for SIE transfer 1: Endpoint 4, buffer 1 is ready for SIE transfer
11	Ep3Buf1Rdy	R/W	0	0: Endpoint 3, buffer 1 is not ready for SIE transfer 1: Endpoint 3, buffer 1 is ready for SIE transfer
10	Ep2Buf1Rdy	R/W	0	0: Endpoint 2, buffer 1 is not ready for SIE transfer 1: Endpoint 2, buffer 1 is ready for SIE transfer
9	Ep1Buf1Rdy	R/W	0	0: Endpoint 1, buffer 1 is not ready for SIE transfer 1: Endpoint 1, buffer 1 is ready for SIE transfer
8	Reserved	NA	-	Reserved
7	Ep7Buf0Rdy	R/W	0	0: Endpoint 7, buffer 0 is not ready for SIE transfer 1: Endpoint 7, buffer 0 is ready for SIE transfer
6	Ep6Buf0Rdy	R/W	0	0: Endpoint 6, buffer 0 is not ready for SIE transfer 1: Endpoint 6, buffer 0 is ready for SIE transfer
5	Ep5Buf0Rdy	R/W	0	0: Endpoint 5, buffer 0 is not ready for SIE transfer 1: Endpoint 5, buffer 0 is ready for SIE transfer
4	Ep4Buf0Rdy	R/W	0	0: Endpoint 4, buffer 0 is not ready for SIE transfer 1: Endpoint 4, buffer 0 is ready for SIE transfer
3	Ep3Buf0Rdy	R/W	0	0: Endpoint 3, buffer 0 is not ready for SIE transfer 1: Endpoint 3, buffer 0 is ready for SIE transfer
2	Ep2Buf0Rdy	R/W	0	0: Endpoint 2, buffer 0 is not ready for SIE transfer 1: Endpoint 2, buffer 0 is ready for SIE transfer
1	Ep1Buf0Rdy	R/W	0	0: Endpoint 1, buffer 0 is not ready for SIE transfer 1: Endpoint 1, buffer 0 is ready for SIE transfer
0	Ep0Buf0Rdy	R/W	0	0: Endpoint 0, buffer 0 is not ready for SIE transfer 1: Endpoint 0, buffer 0 is ready for SIE transfer

Test Mode Register (TMR)

The Test Mode Register (TMR) described in Table 2-18 defines the different test modes in which the AXI USB 2.0 Device operates. The USB Implementors Forum, the organization that controls USB logo certification, requires all USB 2.0 devices that operate at High Speed support these test modes.

Table 2-18: **Test Mode Register** *(C_BASEADDR + 0x0118)*

Bits	Name	Access	Reset Value	Description
31–3	Reserved	R	0	Reserved for future use
2–0	TestMde	R/W	0	Value defines the test mode. 0x0: Normal Mode 0x1: Test Mode J 0x2: Test Mode K 0x3: Test Mode NAK 0x4: Test Mode Packet

The AXI USB 2.0 Device provides test mode support to facilitate compliance testing.

Four test modes are supported:

- Test Mode J – Core transmits a continuous chirp J and remains in this state until the time when it is reset.
- Test Mode K – Core transmits a continuous chirp K and remains in this state until the time when it is reset.
- Test Mode NAK – Core searches for any IN token with a valid crc5. If crc5 is valid, the core sends a NAK, otherwise it waits for the next valid IN token. The core remains in this state until it is reset.
- Test Mode Packet – As specified by the USB 2.0 Specification, the core transmits a test packet which is composed of a predefined sequence of bytes and is used for analog testing of the USB in High Speed mode. The packet data is loaded into a predefined sequence of locations in the DPRAM. This routine repeats continuously until the core is reset.

Error Count Register (ECR)

The Error Count Register (ECR) is described in Table 2-19. This register contains three counters each of 8-bit width. They are BitStuffErrCnt, PidErrCnt, and CrcErrCnt. When USB reset or read to this register is requested, all these counters are cleared and assigned to reset values. When the PHY detects seven consecutive ones on the bus (*bit stuff error condition*), the SIE increments BitStuffErrCnt by one and BitStufErr bit of Interrupt Status Register (ISR) is set.

Whenever four PID check bits are not complement to their respective packet identifier bits while receiving the packet, the SIE increments PidErrCnt by one and PidErr bit of Interrupt Status Register (ISR) is set. Whenever the received CRC does not match with the CRC calculated on the received packet (that is, for CRC5 while receiving token and for CRC16

while receiving data), the SIE increments CrcErrCnt by one and CrcErr bit of Interrupt Status Register (ISR) is set.

Table 2-19: **Error Count Register** *(C_BASEADDR + 0x011C)* (1)(2)

Bits	Name	Access	Reset Value	Description
31–24	Reserved	NA	-	Reserved
23–16	BitStuffErrCnt	R	0x0	Bit Stuff Error Counter
15–8	PidErrCnt	R	0x0	PID Error Counter
7–0	CrcErrCnt	R	0x0	CRC Error Counter

Notes:

1. This register is read-only.
2. When any of the counters reach 255, they roll back and start counting from 0.

ULPI PHY Access Register (UPAR)

The ULPI PHY Access Register (UPAR) shown in Table 2-20 defines the type of access (read or write) to the ULPI PHY registers.

Read – When type of access is configured as read, the user application:

- Writes the address of the PHY register into PHYReg Addr and sets the WriteNotRead bit to 0.
- Core asserts the busy bit of the ULPI PHY Access Register (UPAR) until a successful read is performed from the respective PHY register.
- SIE updates the PHY register read data into PHYRdWrData after a successful read from the PHY register.
- Core clears the busy bit of the ULPI PHY Access Register (UPAR) and sets the ULPIAccComp bit of the Interrupt Status Register (ISR).

Write – When the type of access is configured as write, the user application:

- Writes the address of the PHY register into PHYRegAddr and PHYRdWrData, and sets the WriteNotRead bit to 1.
- Core asserts the busy bit of the ULPI PHY Access Register (UPAR) until a successful write is performed on the respective PHY register.
- Core clears the busy bit of the ULPI PHY Access Register (UPAR) after a successful write to the PHY register and sets ULPIAccComp bit of the Interrupt Status Register (ISR).

Table 2-20: **ULPI PHY Access Register** *(C_BASEADDR + 0x0120)* (1)(2)(3)

Bits	Name	Access	Reset Value	Description
31	busy	R	0	0: SIE is not busy 1: SIE is busy
30–16	Reserved	NA	-	Reserved
15–8	PHYRdWrData	R/W	0x0	PHY register read or write data
7	Reserved	NA	-	Reserved
6	WriteNotRead	R/W	0	0: SIE reads the PHY register 1: SIE writes the PHY register
5–0	PHYRegAddr	R/W	0x0	PHY register address

Notes:

1. The SIE performs the register access onto the ULPI PHY register when ULPI Bus is IDLE (for example, when there is no data exchange between the ULPI PHY and the SIE). If the ULPI Bus is busy with ongoing data transfer, the SIE waits for ULPI Bus IDLE to perform the ULPI PHY register access.
2. Unsuccessful writes to the ULPI PHY access register result when the busy bit is set to 1.
3. ULPI PHY register access is not supported without assertion of MstRdy bit of Control Register (CR).

DMA Software Reset Register (DSRR)

The DMA Software Reset Register (DSRR) shown in Table 2-21 defines reset to the DMA modules by the AXI USB 2.0 Device core when C_INCLUDE_DMA is set to 1.

Table 2-21: **DMA Software Reset Register** *(C_BASEADDR + 0x0200)*

Bits	Name	Access	Reset Value	Description
31–0	DmaRst	W	NA	A write of 0x0000000A causes the reset to the DMA modules.

DMA Control Register (DMACR)

The DMA Control Register (DMACR) described in Table 2-22 defines the direction of the transfer as either Read or Write to DPRAM2. The EpBufSel bit determines whether the SIE side Buffer Ready status is updated by the hardware when the DMA is complete or by the firmware through the BRR register. If the EpBufSel bit is set to 1, the DMA Controller updates the Buffer Ready status to the SIE based on bits [16:31] of the register at the end of a successful DMA transfer only.

Table 2-22: **DMA Control Register** *(C_BASEADDR + 0x0204)*

Bits	Name	Access	Reset Value	Description
31	Direction(1) (2)	R/W	0	0: Write data into DPRAM2 1: Read data from DPRAM2
30	EpBufSel(3)	R/W	0	0: Buffer Ready set by firmware 1: Buffer Ready set by DMA Controller

Table 2-22: **DMA Control Register** *(C_BASEADDR + 0x0204)* ***(Cont'd)***

Bits	Name	Access	Reset Value	Description
29–16	Reserved	NA	-	Reserved
15	Ep7Buf1Rdy	R/W	0	0: Endpoint 7, buffer 1 not ready for SIE transfer after the DMA operation completes 1: Endpoint 7, buffer 1 ready for SIE transfer after the DMA operation completes
14	Ep6Buf1Rdy	R/W	0	0: Endpoint 6, buffer 1 not ready for SIE transfer after the DMA operation completes 1: Endpoint 6, buffer 1 ready for SIE transfer after the DMA operation completes
13	Ep5Buf1Rdy	R/W	0	0: Endpoint 5, buffer 1 not ready for SIE transfer after the DMA operation completes 1: Endpoint 5, buffer 1 ready for SIE transfer after the DMA operation completes
12	Ep4Buf1Rdy	R/W	0	0: Endpoint 4, buffer 1 not ready for SIE transfer after the DMA operation completes 1: Endpoint 4, buffer 1 ready for SIE transfer after the DMA operation completes
11	Ep3Buf1Rdy	R/W	0	0: Endpoint 3, buffer 1 not ready for SIE transfer after the DMA operation completes 1: Endpoint 3, buffer 1 ready for SIE transfer after the DMA operation completes
10	Ep2Buf1Rdy	R/W	0	0: Endpoint 2, buffer 1 not ready for SIE transfer after the DMA operation completes 1: Endpoint 2, buffer 1 ready for SIE transfer after the DMA operation completes
9	Ep1Buf1Rdy	R/W	0	0: Endpoint 1, buffer 1 not ready for SIE transfer after the DMA operation completes 1: Endpoint 1, buffer 1 ready for SIE transfer after the DMA operation completes
8	Reserved	NA	-	Reserved
7	Ep7Buf0Rdy	R/W	0	0: Endpoint 7, buffer 0 not ready for SIE transfer after the DMA operation completes 1: Endpoint 7, buffer 0 ready for SIE transfer after the DMA operation completes
6	Ep6Buf0Rdy	R/W	0	0: Endpoint 6, buffer 0 not ready for SIE transfer after the DMA operation completes 1: Endpoint 6, buffer 0 ready for SIE transfer after the DMA operation completes
5	Ep5Buf0Rdy	R/W	0	0: Endpoint 5, buffer 0 not ready for SIE transfer after the DMA operation completes 1: Endpoint 5, buffer 0 ready for SIE transfer after the DMA operation completes

Table 2-22: **DMA Control Register** *(C_BASEADDR + 0x0204)* ***(Cont'd)***

Bits	Name	Access	Reset Value	Description
4	Ep4Buf0Rdy	R/W	0	0: Endpoint 4, buffer 0 not ready for SIE transfer after the DMA operation completes 1: Endpoint 4, buffer 0 ready for SIE transfer after the DMA operation completes
3	Ep3Buf0Rdy	R/W	0	0: Endpoint 3, buffer 0 not ready for SIE transfer after the DMA operation completes 1: Endpoint 3, buffer 0 ready for SIE transfer after the DMA operation completes
2	Ep2Buf0Rdy	R/W	0	0: Endpoint 2, buffer 0 not ready for SIE transfer after the DMA operation completes 1: Endpoint 2, buffer 0 ready for SIE transfer after the DMA operation completes
1	Ep1Buf0Rdy	R/W	0	0: Endpoint 1, buffer 0 not ready for SIE transfer after the DMA operation completes 1: Endpoint 1, buffer 0 ready for SIE transfer after the DMA operation completes
0	Reserved	NA	-	Reserved

Notes:

1. Write data into DPRAM means that the DMA controller writes data to DPRAM by reading the data from the external memory
2. Read data from DPRAM means that the DMA controller reads data from DPRAM and writes the data to the external memory
3. In DMA mode (C_INCLUDE_DMA = 1), the Buffer Ready status can be updated by either firmware (through the BRR register)
   or hardware. The setting is controlled by the EpBufSel bit of the DMA Control register.

DMA Source Address Register (DSAR)

The DMA Source Address register described in Table 2-23 defines the source address for the current DMA transfer. When data is moved from the source address, this register updates to track the current source address. If the Direction bit of the DMA Control Register (DMACR) is set to 0, the DMA Source Address of the current DMA transfer should be external memory address. If the Direction bit of the DMA Control Register (DMACR) is set to 1, the DMA Source Address of the current DMA transfer should be DPRAM address.

Table 2-23: **DMA Source Address Register** *(C_BASEADDR + 0x0208)*

Bits	Name	Access	Reset Value	Description
31–0	DmaSrcAddr	R/W	0	Source Address for the current DMA transfer.

DMA Destination Address Register (DDAR)

The DMA Destination Address register described in Table 2-24 defines the destination address for the current DMA transfer. When data is moved to the destination address, this register updates to track the current destination address. If Direction bit of the DMA Control

Register (DMACR) is set to 0, the DMA Destination Address of the current DMA transfer should be DPRAM address. If Direction bit of theDMA Control Register (DMACR) is set to 1, the DMA Destination Address of the current DMA transfer should be external memory address.

Table 2-24: **DMA Destination Address Register** *(C_BASEADDR + 0x020C)*

Bits	Name	Access	Reset Value	Description
31–0	DmaDstAddr	R/W	0	Destination Address for the current DMA transfer.

DMA Length Register (DMALR)

The DMA Length register shown in Table 2-25 defines the total number of bytes to be transferred from source address to destination address. This register is written only after configuring the DMA Control Register (DMACR), the DMA Source Address Register (DSAR), and the DMA Destination Address Register (DDAR), with their values and any other setup is complete. As bytes are successfully written to the destination, the DMA Length Register (DMALR) decrements to reflect the number of bytes remaining to be transferred. The DMA Length Register (DMALR) is zero after a successful DMA operation.

Table 2-25: **DMA Length Register** *(C_BASEADDR + 0x0210)*

Bits	Name	Access	Reset Value	Description
31–0	DmaLen	R/W	0	Number of bytes to be transferred from source to destination

DMA Status Register (DMASR)

The DMA Status Register (DMASR) described in Table 2-26 defines the status of DMA controller as DmaBusy or DmaErr. When DMA operation is in progress, the DmaBusy is set to 1 until the DMA operation has been finished. If the DMA operation encounters any error condition, the DmaErr is set to 1.

Table 2-26: **DMA Status Register** *(C_BASEADDR + 0x0214)*

Bits	Name	Access	Reset Value	Description
31	DmaBusy	R	0	0: DMA operation is not initiated 1: DMA operation is in progress
30	DmaErr	R	0	0: DMA Error has not occurred 1: DMA Error has occurred
29–0	Reserved	NA	-	Reserved

Designing with the Core

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

Clocking

The clock to the USB 2.0 Device runs at 480 MHz when operating at High Speed. Because this frequency is too high for the SIE clock, as well as for the FPGA, the PHY interfaces with the SIE and generates a 60 MHz clock.

The AXI USB 2.0 Device uses at least two clocks, depending on the configuration:

- `S_AXI_ACLK` – The AXI Bus interface, Port B of the DPRAM, and the control and status registers use this clock. The minimum S_AXI_ACLK frequency needed to achieve the maximum performance of 480 MHz is 60 MHz.
- `ULPI_CLK` – The SIE interface and Port A of the DPRAM operate using this clock. The ULPI clock is generated by the PHY and has a fixed frequency of 60 MHz. An additional clock is used when the DMA controller is enabled.
- `M_AXI_ACLK` – The DMA controller uses this clock to read and write to DPRAM from other bus peripherals.

Reset

The AXI USB 2.0 Device core is reset using the `S_AXI_ARESETN` signal which resets all devices that are connected to the AXI Bus in the processor system. The minimum duration of the reset pulse required to reset the logic on the SIE side is 1 ULPI clock period.

Programming Sequence

This section describes how to program the AXI USB 2.0 Device for various operations. For applications that use the Xilinx driver, change the Vendor ID before using the AXI USB 2.0 Device.

Initialization Sequence

1. The AXI USB 2.0 Device core issues ULPI_reset to the PHY during power-on reset.
2. The firmware enables the USBDsc, USBSpnd, and USBRst bits of the Interrupt Enable Register (IER).
3. The firmware enables the SIE by setting the MstRdy bit of the Control Register (CR).
4. When the MstRdy bit of the Control Register (CR) is set to 1 by the firmware, the SIE configures the PHY to device mode by writing OTG Control Register (CR) to 0x0, Function Control register to 0x41 (Sets TermSelect to 0x0, XcvrSelect to 0x1 (FS), and Opmode to 0x0 (Normal Operation)).
5. If the AXI USB 2.0 Device core was not connected to the host, the PHY updates Session End (VBUS not present) RXCMD to the SIE. At that point, the SIE waits for a connection to the host.
6. When the AXI USB 2.0 Device core connects to the host, the PHY updates VBUS Valid (VBUS present) RXCMD to the SIE.
7. The SIE pulls up the D+ line of the PHY by writing 0x45 (Sets TermSelect to 0x1, XcvrSelect to 0x1 (FS), by writing Opmode to 0x0 (Normal Operation)) to the Function Control register of the PHY, and moves to Full Speed mode. The SIE updates the USBDsc bit of the Interrupt Status Register (ISR) to 1.
8. When the PHY pulls up the D+ line, the host detects the AXI USB 2.0 Device connection and starts issuing a USB reset to bring the device into the default unconfigured state.
9. The SIE detects USB reset from the host and updates the USBRst bit of the Interrupt Status Register (ISR) to 1 until USB reset signaling is finished between the host and the AXI USB 2.0 Device.
10. After the successful completion of the USB reset by the host, the SIE updates the HSEn bit (0 = Full Speed, 1 = High Speed) and starts receiving SOFs every 1 ms in Full Speed and every 125 µs in High Speed.
11. The AXI USB 2.0 Device responds to transfers from the host based on the endpoint configuration and status registers programmed.

Operating in Interrupt Mode

The device can be configured to operate in the interrupt mode after enabling the desired interrupts in the Interrupt Enable Register (IER).

- Interrupts for USBRst, USBSpnd, USBRsm (Remote-wake-up), and USBDsc can be enabled by writing to those specific bits of the IER.
- To generate an interrupt when the core is operating in High Speed, the HighSpd bit of the IER should be set.

- To generate an interrupt when a start-up packet or Start Of Frame (SOF) packet is received, those bits of the IER must be set.
- For endpoint 0, set the FIFO Buffer Ready, and FIFO Buffer Free bits of the IER to generate interrupts when packets are received or transmitted respectively.
- For all other endpoints, set the respective Buffer Complete bit of the IER to generate interrupts when that endpoint buffer is complete.

Setting the USB 2.0 Device Address

Set the device to the zero state by writing an address of 0 to the USB Address Register (UAR) register.

Configuring an Endpoint

Program the individual Endpoint Configuration and Status registers to configure the respective endpoints:

- A specific endpoint can be enabled by writing a 1 to the EpValid bit of the endpoint configuration and status register.
- The direction of an endpoint can be set to IN by writing a 1, or to OUT by writing 0 to the EpOutIn bit of the register.
- The endpoint can be configured as an isochronous endpoint by writing a 1, or as a bulk endpoint by writing 0 to the EpIso bit of the register.
- The packet size for the endpoint can be set by writing to the EpPktSze bits of the register.
- The base offset of the endpoint buffers in the DPRAM can be set by writing to the EpBase bits of the register.

Handling a Control Packet

1. Wait for the SETUPPkt interrupt to detect the reception of the setup packet.
2. Read the setup packet from the buffer location of Endpoint 0 in the DPRAM, which causes the SETUPPkt bit of the Interrupt Status Register (ISR) to be cleared.
3. Process the received command (as detailed in Chapter 9 of the *Universal Serial Bus Specification, Revision 2.0* [Ref 3]) and prepare a buffer for a response that must be sent for the subsequent IN packet.

Handling Bulk/Isochronous IN Transactions

For a Bulk or Isochronous IN transaction in No DMA Mode, perform the following steps:

1. Write the IN packet into the selected endpoint buffer location in the DPRAM.
2. Write the packet count into the specific endpoint buffer Buffer Count register.
3. Set the Buffer Ready bit for the selected endpoint buffer in the Buffer Ready Register (BRR).
4. Wait for the Buffer Ready interrupt of the selected endpoint buffer in the Buffer Ready Register (BRR) (this bit must be cleared) to ensure that the transmitted data has been received by the host and the buffer is available for the next write.

For a Bulk or Isochronous IN transaction in DMA-Mode, perform the following steps:

1. Write the packet count into the specific endpoint buffer Buffer Count register.
2. Configure DMA by writing into Direction bit of DMA Control Register (DMACR) to 0, DMA Destination Address (same as EpBase of the endpoint configuration register of the respective endpoint), DMA Source Address Register (DSAR) and DMA Length Register (DMALR).
3. DMA generates DmaDne interrupt after successfully writing the configured length of data into the DPRAM.
4. Set the Buffer Ready bit for the selected endpoint buffer in the Buffer Ready Register (BRR).
5. Wait for the Buffer Complete interrupt of the selected endpoint buffer in the Interrupt Status Register (ISR) to ensure that the transmitted data has been received by the host and the buffer is available for the next write.

Handling Bulk/Isochronous OUT Transactions

When the host sends an OUT packet to an endpoint buffer on the device while the buffer is in use, the device core sends a NAK to the host if EpIsoc bit is not set. When the buffer is free, the packet is written into the buffer and an ACK is automatically sent to the host if EpIsoc bit is not set.

On reception of the OUT packet in No DMA Mode, perform the following steps:

1. Poll the Buffer Complete bit of the selected endpoint buffer in the Interrupt Status Register (ISR) to detect the reception of a packet.
2. Check that the received packet count matches with the specified packet count in the Buffer Count register.
3. Read the OUT packet from the selected endpoint buffer location in the DPRAM.
4. Set the Buffer Ready bits of the selected endpoint buffer to prepare it for the next transaction.

On reception of the OUT packet in DMA-Mode, perform the following steps:

1. Poll the Buffer Complete bit of the selected endpoint buffer in the Interrupt Status Register (ISR) to detect the reception of a packet.
2. Check the received packet count matches with the specified packet count in the Buffer Count register.
3. Read the OUT packet by configure DMA by writing into Direction bit of DMA Control Register (DMACR) to 1, DMA Source Address Register (DSAR) (same as EpBase of the endpoint configuration register of the respective endpoint), DMA Destination Address Register (DDAR) and DMA Length Register (DMALR).
4. DMA generates DmaDne interrupt after successfully reading the configured length of data into the DPRAM2.
5. Set the Buffer Ready bits of the selected endpoint buffer to prepare it for the next transaction.

Protocol Description in ULPI Mode

USB Reset signaling

When the host wants to start communicating with a device it starts by applying a "reset" condition which sets the device to its default unconfigured state. This "reset" should not be confused with a micro-controller power-on type reset. It is a USB protocol reset to ensure that the device USB signaling starts from a known state. The High Speed capable AXI USB 2.0 Device core can be reset while the device is in the Powered, Default, Address, Configured, or Suspended states. The AXI USB 2.0 Device can be successfully reset by any host (even an USB1.x host).

The host performs USB reset signaling High Speed capable AXI USB 2.0 Device as follows.

1. The host drives SE0; the start of SE0 is referred to as time T0.
2. The SIE detects assertion of SE0.
    a. If the SIE detects SE0 from the USB Suspend state, the SIE begins the High Speed handshake detection after the detection of SE0 for no less than 2.5 µs.
    b. If the SIE detects SE0 from a non-suspended Full Speed mode, the SIE begins a High Speed handshake detection after the detection of SE0 for no less than 2.5 µs and no more than 3.0 ms.
    c. If the SIE detects SE0 from a non-suspended High Speed mode, the SIE wait for 3.0 ms before reverting to Full Speed mode. After reverting to Full Speed, the SIE checks the line state update from the PHY for SE0 (to check whether the host has issued suspend or USB reset), no less than 1 ms from the point the SIE reverted to Full Speed. If the SIE detects SE0 by the RXCMD updating from the PHY, the SIE begins the High Speed handshake detection.
3. Because the AXI USB 2.0 Device is a High Speed capable device, the SIE follows High Speed Handshake Detection protocol.

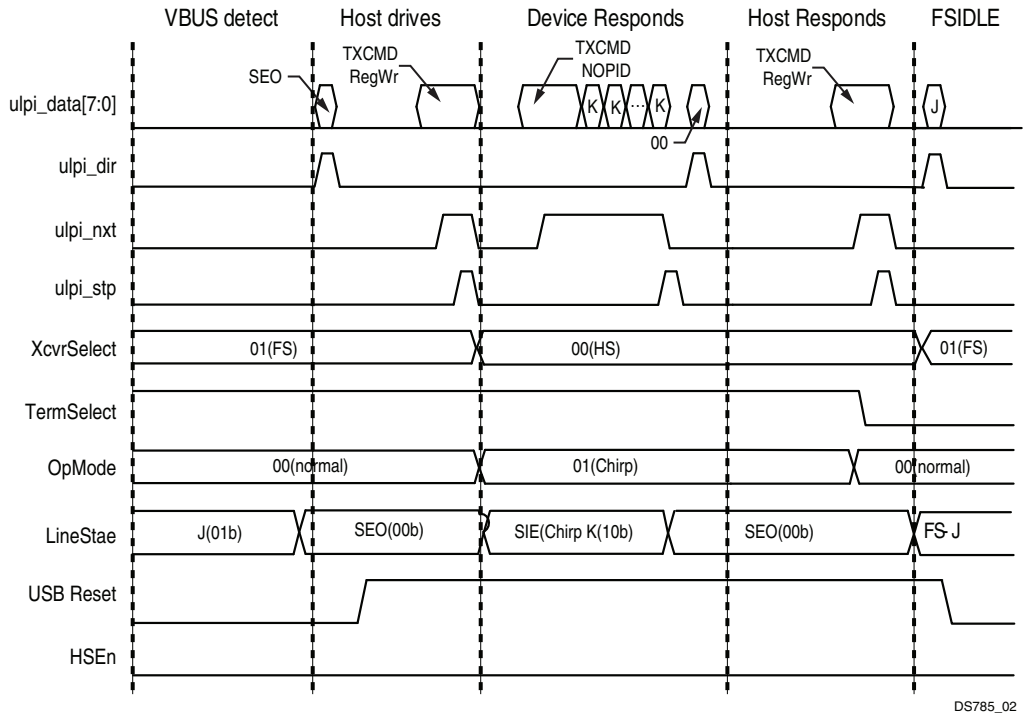

Figure 3-1: **USB Reset signaling with USB1.x Host – Full Speed**

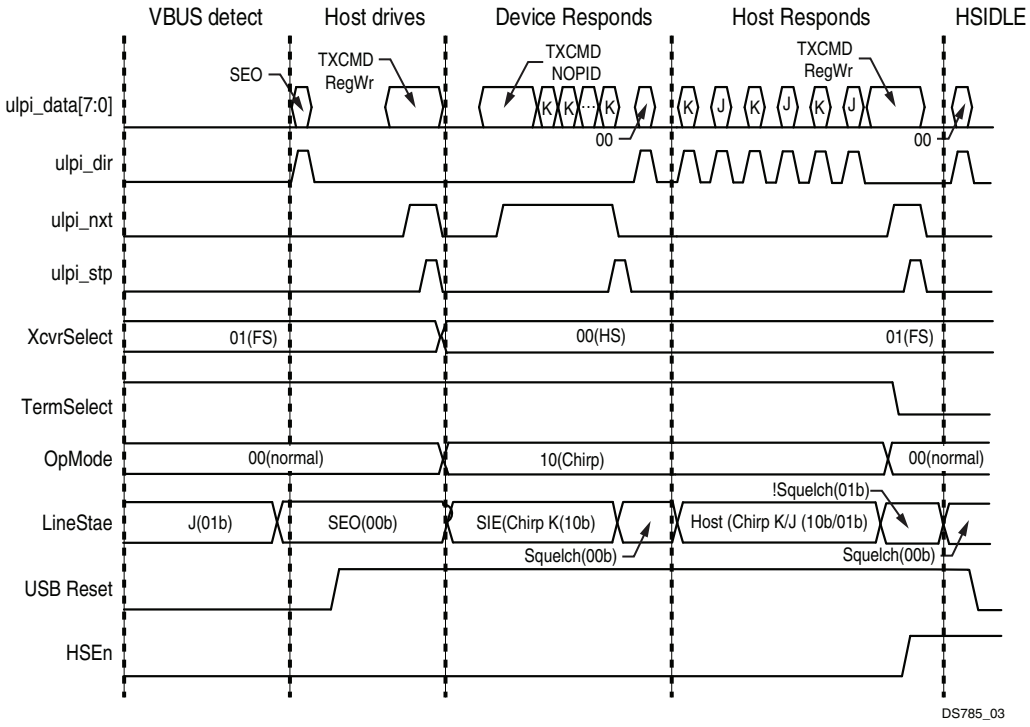

Figure 3-2: **USB Reset signaling with USB 2.0 Host – High Speed Handshaking**

High Speed Handshake Detection Protocol

1. The SIE writes 0x54 (sets XcvrSelect to 0x0, TermSelect to 0x1, and Opmode to 0x2 (Chirping)) to the Function Control register of the PHY.
2. Immediately after the write, the SIE puts TXCMD as 0x40 (NOPID) and starts transmitting a chirpK for 2 ms.
3. After 2 ms of ChirpK transmission, the PHY updates RXCMD with line state as SE0.
4. The SIE detects SE0 and waits for 100 µs to look for a High Speed host chirp, for example, an alternating sequence of ChirpKs and ChirpJs.
5. If the SIE does not observe any High Speed host chirp, the SIE configures the PHY to Full Speed mode by writing 0x45 (sets XcvrSelect to 0x1, TermSelect to 0x1, and Opmode to 0x0 (Normal Operation)) to the Function Control register of the PHY. Now the SIE is in Full Speed mode and waits for Full Speed USB traffic from the host.
6. If the SIE observes the High Speed host chirp, the SIE checks for a minimum of chirp K-J-K-J-K-J.
7. The SIE also checks each individual chirpK and chirpJ for at least 2.5 µs.
8. After detecting the minimum sequence, the SIE configures the PHY to High Speed mode by writing 0x40 (Sets TermSelect to 0x0, XcvrSelect to 0x0 (HS,) and Opmode to 0x0 (Normal Operation)) to the Function Control register of the PHY and updates the HSEn bit of the Interrupt Status Register (ISR) to 1.
9. Now the SIE is in High Speed mode and sees line-state as !squelch (linestate != 00) on the RXCMD update.
10. If the SIE observes squelch as linestate update from the PHY (for example, RXCMD[1:0]), the SIE treats that the host has completed the chirp and stops updating the status as USBRst in the Interrupt Status Register (ISR).
11. Now the SIE waits for High Speed USB traffic from the host.

Suspend signaling

When the host does not want to continue any further communication with the connected device, it keeps the AXI USB 2.0 Device in the suspend mode by ceasing all of its transmissions (including SOFs). The AXI USB 2.0 Device recognizes the suspend condition in High Speed mode as follows:

1. Initially, the host and the AXI USB 2.0 Device are either in Full Speed or High Speed mode.
2. When the SIE sees no bus activity for 3 ms, it enters Full Speed mode by writing 0x45 (Sets TermSelect to 0x1, XcvrSelect to 0x1 (FS), and Opmode to 0x0 (Normal Operation)) to the Function Control register.
3. After 1 ms, the SIE samples the line state from the RXCMD update of the PHY.

4. If the line state is Full Speed J, the SIE enters into the Suspend state after 7 ms and updates the status as suspended by asserting the USBSpnd bit of the Interrupt Status Register (ISR).
5. The SIE is now SIE in the Suspended state.
6. Resume or USB reset signaling can bring the SIE out from suspended state.

Resume signaling

When the AXI USB 2.0 Device in the suspend state, host wake-up the device with a resume signaling. The AXI USB 2.0 Device detects Resume signaling as listed in the steps:

1. When both the host and the AXI USB 2.0 Device are in Low Power mode.
2. If the SIE observes Full Speed K as line state (that is, RXCMD[1:0]) update from the PHY then the SIE treats it as Resume signaling from the PHY and updates the status as Resume by setting USBRsm bit of the Interrupt Status Register (ISR).
3. The SIE waits to detect the End of Resume, that is, SE0 as line state update from the PHY.
4. If the SIE detects SE0 linestate update from the PHY:
    a. SIE moves to Full Speed if the SIE was in Full Speed prior to entering into suspend state.
    b. SIE moves to High Speed by writing 0x40 (Sets TermSelect to 0x0, XcvrSelect to 0x0 (HS), and Opmode to 0x0 (Normal Operation)) to the Function Control register of the PHY if the SIE was in High Speed prior to entering into the suspend state.
5. The SIE updates the HSEn bit of Interrupt Status Register (ISR) accordingly.

The Suspend signaling followed by Resuming signaling from the host is shown in Figure 3-3 (Full Speed) and Figure 3-4 (High Speed).

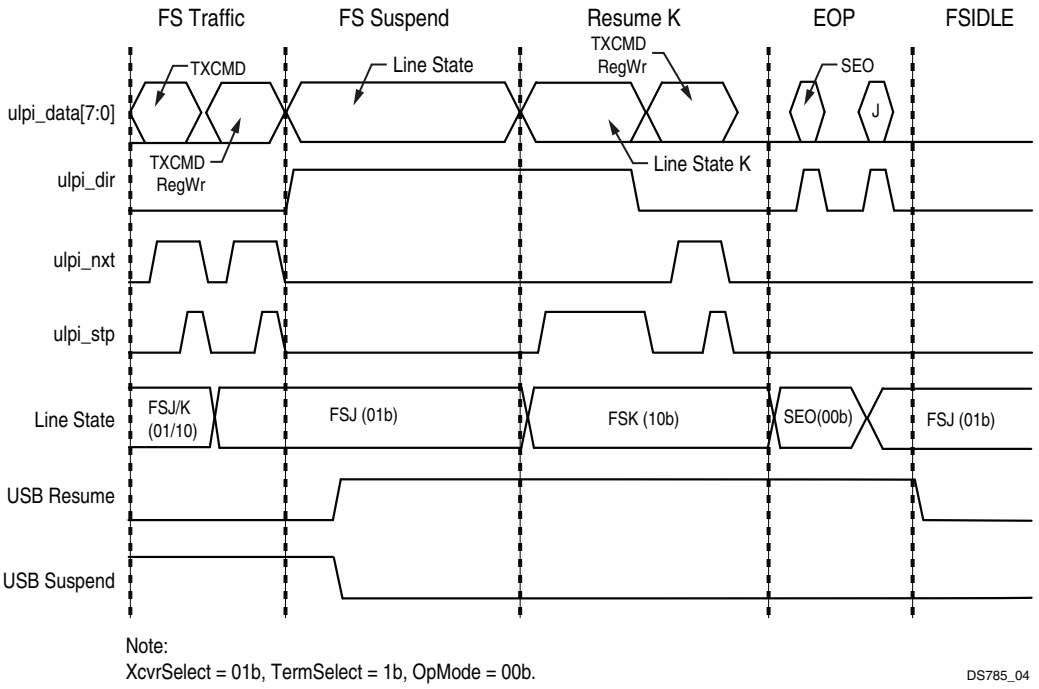

Figure 3-3: **Suspend signaling Followed with Resume signaling from Host – Full Speed**

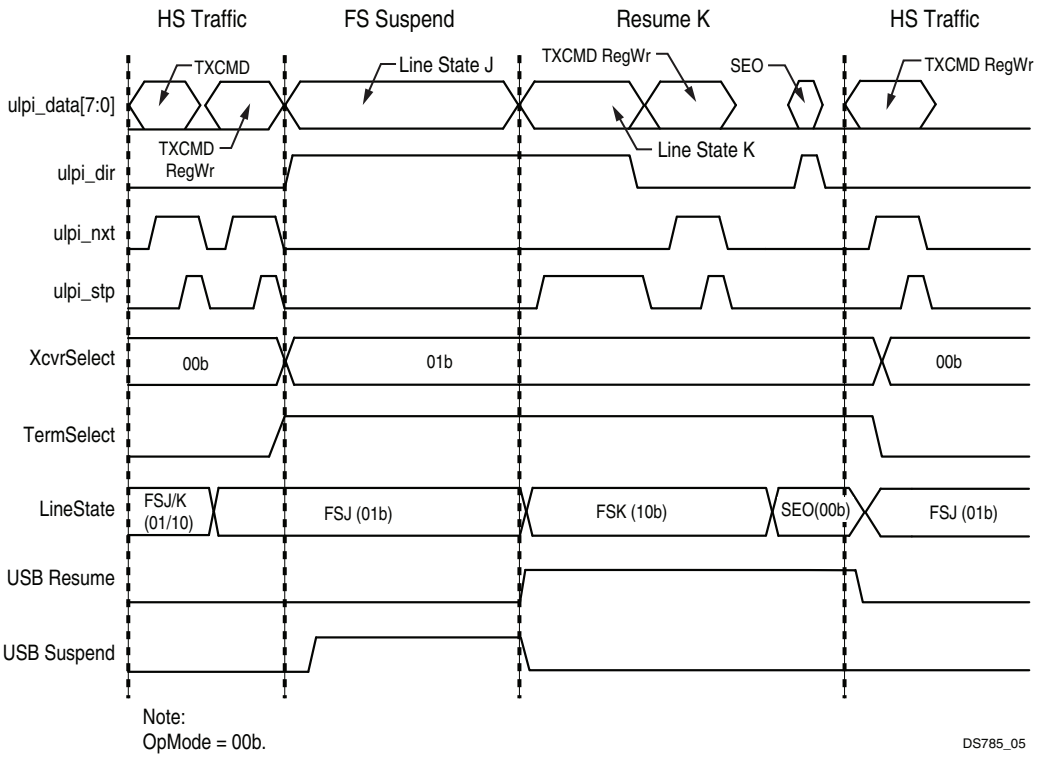

Figure 3-4: **Suspend signaling Followed with Resume signaling from Host – High Speed**

Remote Wake-Up signaling

When the AXI USB 2.0 Device is suspended by the host, it supports the Remote Wake-up feature and initiates a resume itself. The AXI USB 2.0 Device initiates the Remote Wake-up signaling as follows. The Suspend signaling is followed by Remote Wake-up signaling from the AXI USB 2.0 Device, followed by a Resume signaling from the host as shown in Figure 3-5 (Full Speed) and Figure 3-6 (High Speed):

1. When both the host and the AXI USB 2.0 Device are in Low Power Mode.
2. If the RmteWkup bit of the Control Register (CR) is set to 1, the SIE begins Remote Wake-up signaling by writing 0x54 (sets XcvrSelect to 0x0, TermSelect to 0x1, and Opmode to 0x2 (Chirping)) to the Function Control register of the PHY.
3. Immediately after the write, the SIE puts TXCMD as 0x40 (NOPID) following a Full Speed K for 3 ms and updates the status as Resume by setting the USBRsm bit of the Interrupt Status Register (ISR).
4. The host takes over driving the Resume K within 1 ms after detecting the Remote Wake-up from the SIE.
5. The SIE waits to detect the End of Resume, for example, SE0 as line state update from the PHY.
6. If the SIE detects an SE0 linestate update from the PHY,
    a. SIE moves to Full Speed by writing 0x45 (Sets TermSelect to 0x1, XcvrSelect to 0x1 (FS), and Opmode to 0x0 (Normal Operation)) to the Function Control register of the PHY, if the SIE was in Full Speed prior to entering into the suspend state.
    b. SIE moves to High Speed by writing 0x40 (Sets TermSelect to 0x0, XcvrSelect to 0x0 (HS), and Opmode to 0x0 (Normal Operation)) to the Function Control register of the PHY, if the SIE was in High Speed prior to entering into the suspend state.
7. The SIE updates the HSEn bit of Interrupt Status Register (ISR) accordingly.

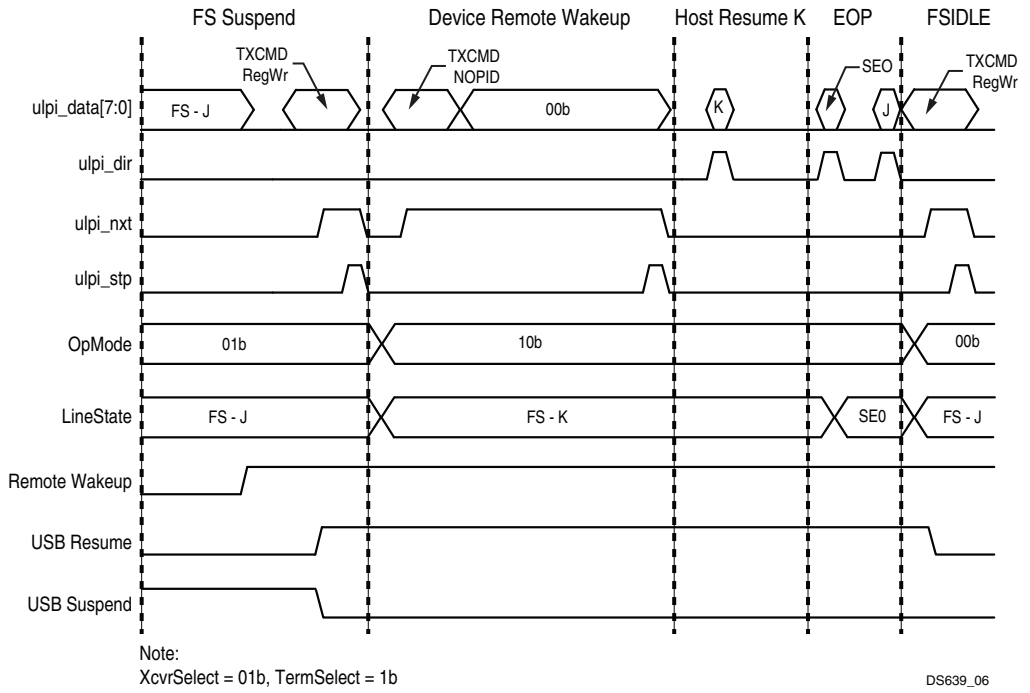

Figure 3-5: **Suspend signaling Followed by Remote Wake-Up signaling from SIE – Full Speed**

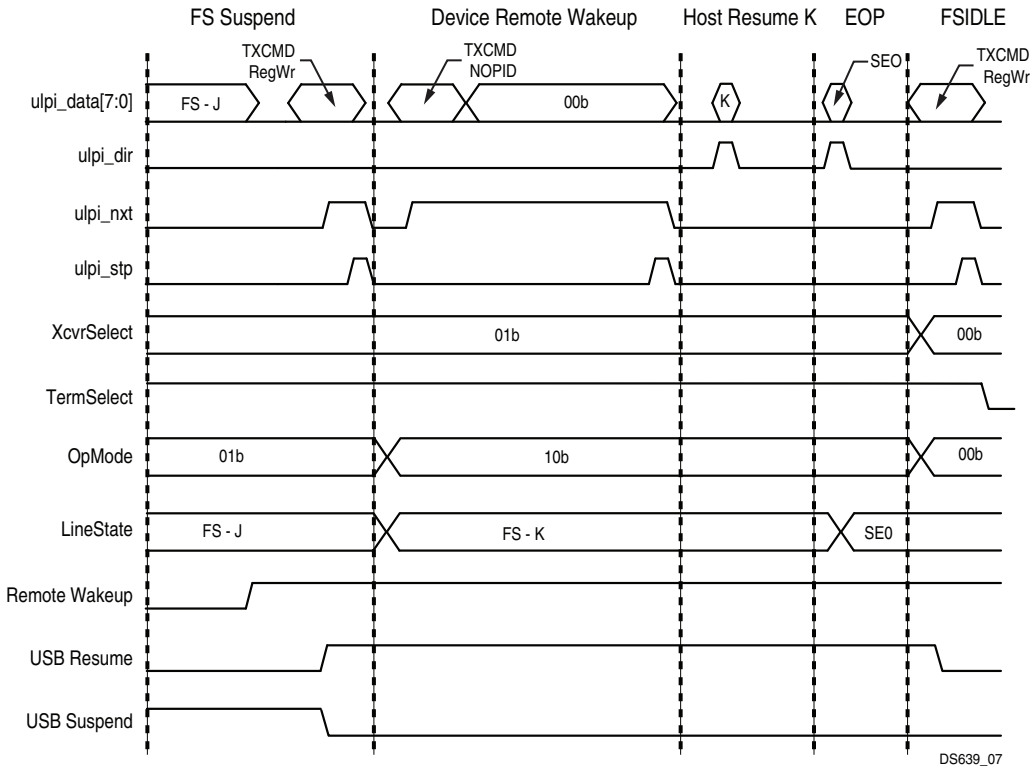

Figure 3-6: **Suspend signaling Followed by Remote Wake-Up signaling from SIE – High Speed**

Test Mode Operation

The default mode of operation for the USB 2.0 Device is the normal mode, for which all the bits of the Test Mode Register (TMR) are set to 0. To put the core in Test Mode operation, program the bits [2:0] of the Test Mode Register (TMR) for different test modes:

- To put the core in Test mode J, program TMR[2:0] = 001. In this mode, Chirp J sequences must be seen on the bus.
- To put the core in Test mode K, program TMR[2:0] = 010. In this mode, Chirp K sequences must be seen on the bus.
- To put the core in Test mode NAK, program TMR[2:0] = 011. In this mode, NAK must be seen on the bus.
- To put the core in Test mode packet, program TMR[2:0] = 100. In this mode, the test packet specified by the USB 2.0 Specification must be seen on the bus.
- For endpoint 0, set the FIFO Buffer Ready, and FIFO Buffer Free bits of the IER to generate interrupts when packets are received or transmitted respectively.
- For all other endpoints, set the respective Buffer Complete bit of the IER to generate interrupts when that endpoint buffer is complete.

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this core. More detailed information about the standard Vivado® design flows in IP Integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 4]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 6]

Customizing and Generating the Core

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

If you are customizing and generating the core in the Vivado IP Integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 1] for detailed information. IP Integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

You can customize the core for use in your design by specifying values for the various parameters associated with the core using the following steps:

1. Select the IP from the IP catalog (choose **Embedded processing, AXI Peripheral, High Speed Peripheral** in the View by Function pane).
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5].

Note: Figures in this chapter are illustrations of the Vivado IDE. The layout depicted here might vary from the current version.

Figure 4-1 shows the AXI USB 2.0 Device core Customize IP GUI in the Vivado Integrated Design Environment (IDE).

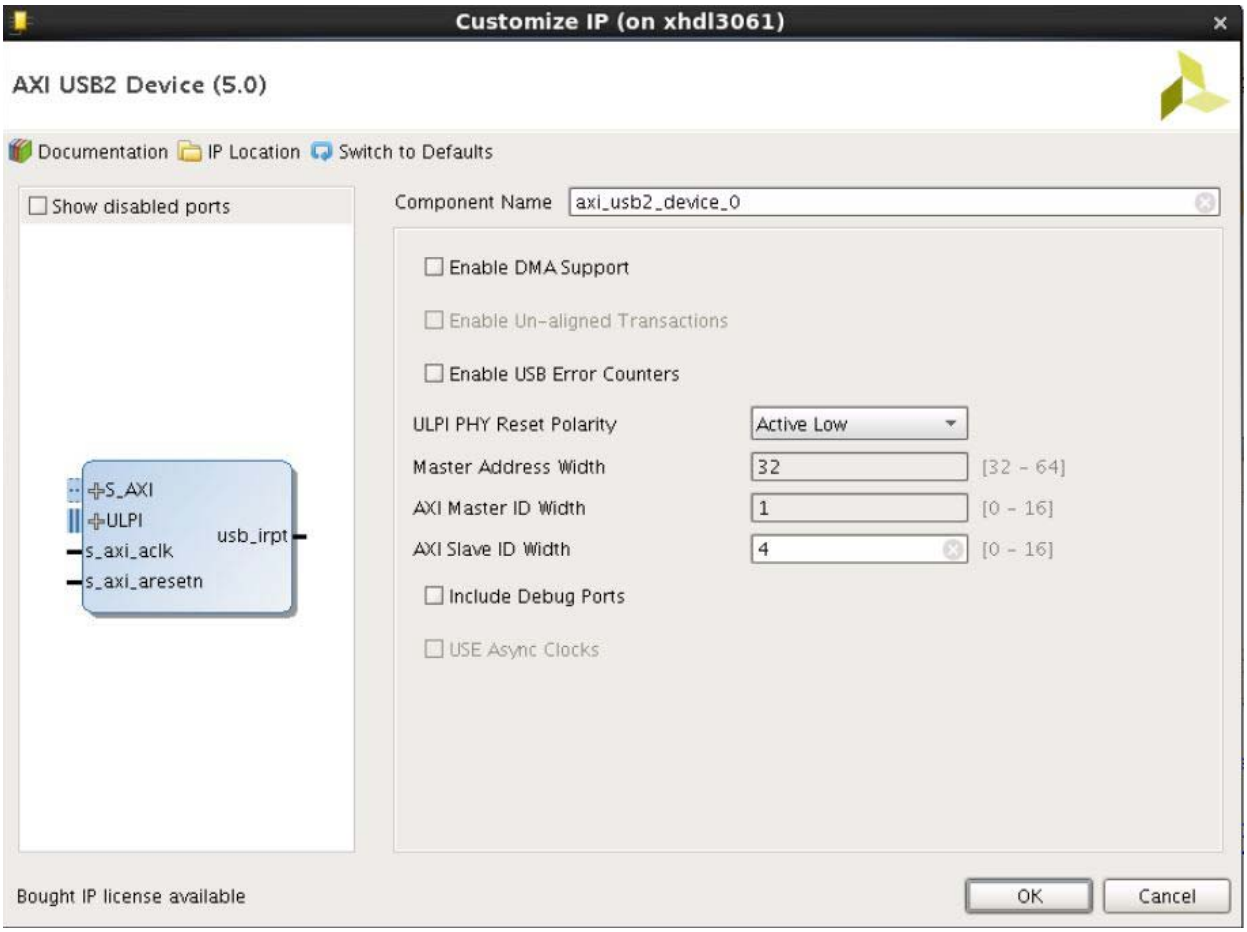

Figure 4-1: **Vivado IP Catalog AXI USB 2.0 Device Configuration**

Figure 4-2 shows the core configuration interface in IP Integrator.

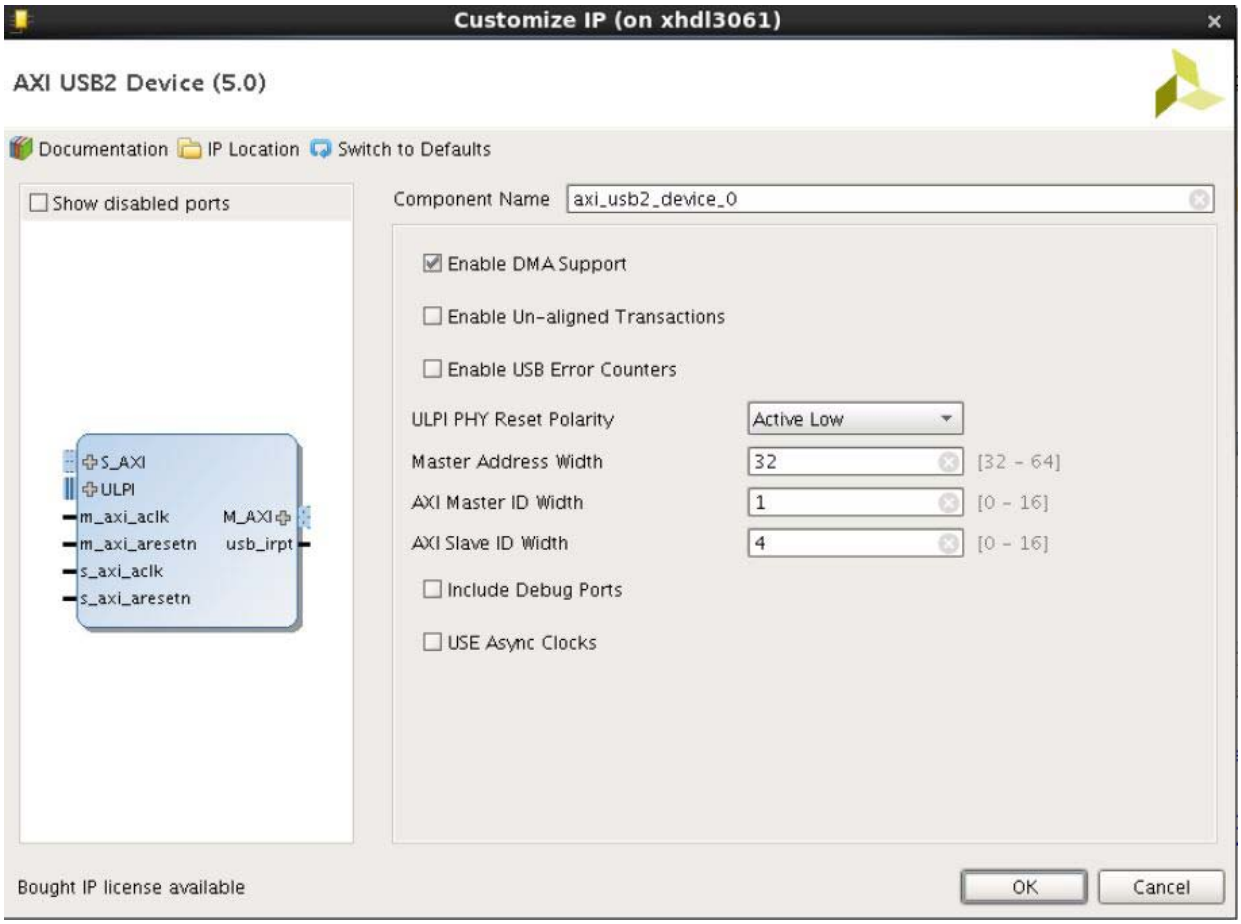

Figure 4-2: **AXI USB 2.0 Device Configuration in IP Integrator**

Table 4-1: **AXI USB 2.0 Device Parameters**

Parameter	Values	Default	Description
Enable DMA Support	0,1	0	Include DMA logic in the core
Enable Un-aligned Transactions	0,1	0	Support unaligned transactions.
Enable USB Error counters	0,1	0	Include error counter.
ULPI PHY Reset polarity	Active-High Active-Low	Active-Low	Polarity selection of the ULPI PHY Reset.
AXI Master ID Width	0 to 16	1	Width of the AXI Master ID signal.
AXI Master Address Width	32 to 64	32	Address width of AXI master
AXI Slave ID Width	0 to 16	4	Width of the AXI Slave ID signal. In IP Integrator, this parameter is auto-updated from the ID WIDTH parameter of the connected master.

Table 4-1: **AXI USB 2.0 Device Parameters *(Cont'd)***

Parameter	Values	Default	Description
Use Async Clock	0,1	0	Indicates that the AXI master and slave clocks are asynchronous when set. In IP Integrator, this parameter is auto-updated based on the clocks connected to the s_axi_aclk, and m_axi_aclk ports.
Include Debug Ports	0,1	0	Includes the debug ports as top level ports.

User Parameters

Table 4-2 shows the relationship between the GUI fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl console).

Table 4-2: **GUI Parameter to User Parameter Relationship**

GUI Parameter/Value(1)	User Parameter/Value(1)	Default Value
Enable DMA Support	C_INCLUDE_DMA	0
Enable Un-aligned Transactions	C_DMA_UA_TRANS_SUPPORT	0
Enable USB Error counters	C_INCLUDE_USBERR_LOGIC	0
ULPI PHY Reset polarity	C_PHY_RESET_TYPE	0
Active-High	1	
Active-Low	0	
AXI Master ID Width	C_M_AXI_THREAD_ID_WIDTH	1
AXI Master Address Width	C_M_AXI_ADDR_WIDTH	32
AXI Slave ID Width	C_S_AXI_ID_WIDTH	4
Use Async Clock	C_USE_ASYNC_CLOCKS	0
Include Debug Ports	C_INCLUDE_DEBUG	0

1. Parameter values are listed in the table where the GUI parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

The ULPI pins of the core should be connected to the corresponding pins of the ULPI PHY.

Timing Constraints

The core has the following clock domains:

- s_axi_aclk
- ulpi_clock
- m_axi_clk

The following constraints should be defined in the top level XDC constraints file. The false path constraints are provided in the IP level constraints file generated with the core.

ulpi_clk

The ULPI clock input is generated through the ULPI PHY and has a fixed frequency of 60 MHz.

```
# Set the ULPI_CLK constraints
create_clock -name ulpi_clock -period 16.667 [get_nets <ulpi_clock net at the top
level>]
```

s_axi_aclk

The slave axi clock provided to S_AXI_ACLK must be constrained for a clock frequency of 60 MHz to 200 MHz. If an internal MMCM is used to generate this clock, this clock is declared by the clock generator block, and the same constraint is propagated to this signal; otherwise you need to declare this clock using the following constraint.

```
create_clock -name s_axi_clk -period <period in ns> [get_nets <s_axi_aclk net at the
top level>]
```

If C_INCLUDE_DMA is set and a separate clock source is used for the master AXI clock, you should declare a clock constraint similar to the slave AXI clock above.

ULPI Interface Constraints

The following constraints should be added in the top-level constraints file.

```
#offset in constraints are calculated as follows.
#clock frequency is 16.667ns
#ULPI PHY output delay for output clock mode for SMSC2 PHY is 3.5ns
#onboard trace delay would be 1ns
#input offset = (clock period - PHY outputdelay - trance delay)
#with 1ns hold requirement for the PHY, considering 200ps clock jitter
set ulpi_input {<list of top level input signals for the ULPI interface>}
set ulpi_output {<list of top level output signals for the ULPI interface>}
set_input_delay -max 4.5 -clock <ulpi_clock name> $ulpi_input
set_output_delay -max 7 -clock <ulpi_clock name> \$ulpi_output
set_max_delay 24 -from [get_ports ulpi_dir] -to [get_ports ulpi_data_io[*]]
-datapath_only
```

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 6].

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Example Design

The top module instantiates all components of the core and the example design required to implement the design in hardware, as shown in Figure 5-1. This includes the clock and reset generation modules, ATG lite agent, AXI slave interface, and ulpi_driver modules to drive USB packets.

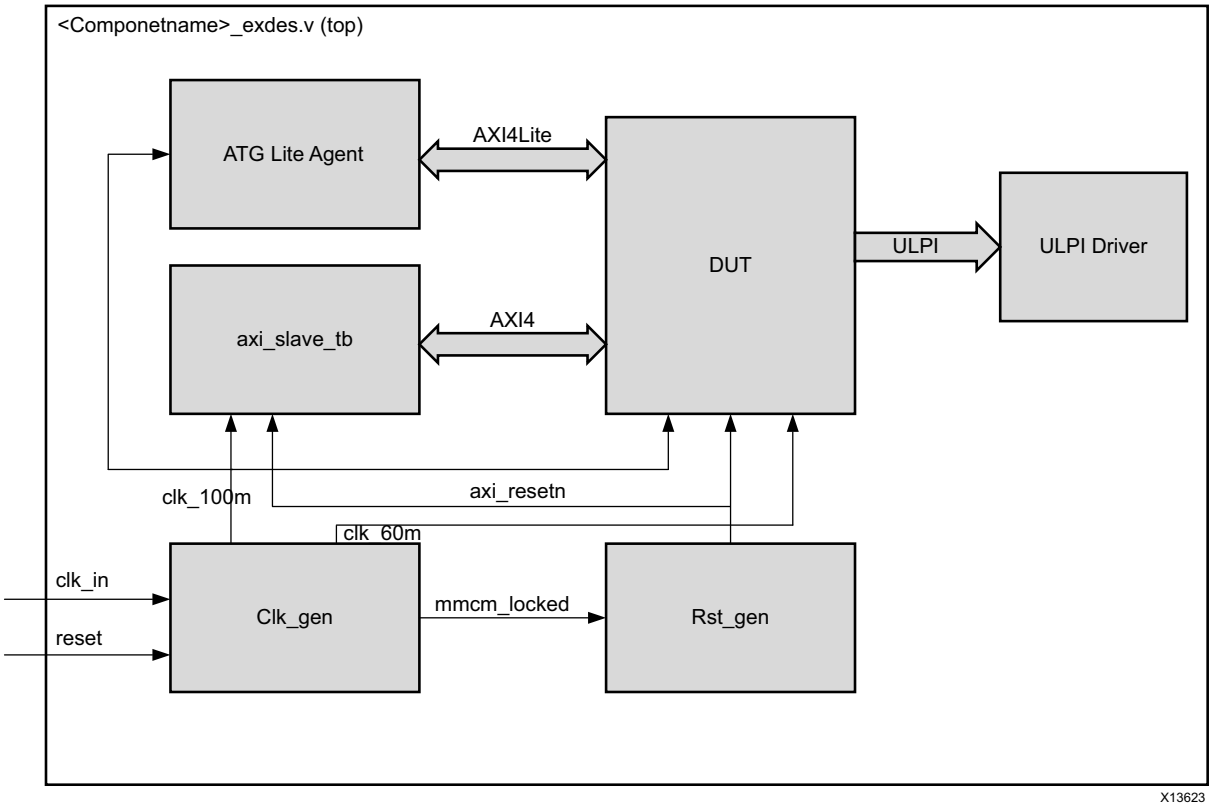

Figure 5-1: **Block Diagram of Example Design**

The example design sets the design under test (DUT) in a known state and illustrates the flow of a single bulk-out transaction on the ULPI interface, depending on the parameters selected.

ATG Lite Agent

The AXI Traffic Generator (ATG) core is used to configure the DUT to a known state. This block generates write transactions on the AXI-Lite interface to write into registers. The ATG

Lite Agent initializes the DUT with the transactions shown in Table 5-1.

Table 5-1: **ATG Lite Agent Transactions**

Register	Address	Data	Transaction
EP0 control reg	BASE_ADDR + 0x0000	0x80200000	Write
EP1 control reg	BASE_ADDR + 0x0010	0x80200000	Write
EP2 control reg	BASE_ADDR + 0x0020	0x80200400	Write
USB Address reg	BASE_ADDR + 0x0100	0x00000001	Write
USB control reg	BASE_ADDR + 0x0104	0x80000000	Write
BRR reg	BASE_ADDR + 0x0114	0x00020002	Write
PII ISR	BASE_ADDR + 0x0108	0x00000002	Read

If the Enable DMA Support parameter is set, the ATG core writes the DMA source address and destination address registers and DMA length register locations with appropriate values and waits for the DMA interrupt. Receipt of the DMA interrupt indicates success.

If the Enable DMA Support parameter is not set, the ATG core directly reads the endpoint 1 buffer locations to get the received packet data. Correct packet data indicates success.

DUT

The AXI USB 2.0 Device instance generated is instantiated as a DUT. The DUT instance changes depending on the parameters selected in the GUI.

clk_gen

The clock generator module instantiates MMCM to generate 100 MHz, 60 MHz, 240 MHz, and 200 MHz clocks. The input frequency is 200 MHz single ended clock from the board. A 100 MHz clock is provided to the ATG Lite Agent and AXI slave interface logic and DUT as an AXI clock. A 60 MHz clock is provided to the `ulpi_clock` pin of the DUT.

rst_gen

Reset generation logic uses a locked signal from the MMCM to release the reset to the example design. The reset signal is generated synchronous to the 100 MHz clock.

ULPI Driver

This block responds to the initialization sequence from the DUT. This block sends the bulk-out transaction with incremented data of 8 bytes and waits for an ACK from the DUT. When acknowledgement is received, transaction ends and moves to the IDLE state.

Implementing the Example Design

After following the steps to generate the core described in Chapter 4, Design Flow Steps, implement the example design as follows:

1. Right-click the core in the **Hierarchy** window, and select **Open IP Example Design**.

   A new window opens instructing you to specify a directory for the example design.

2. Select a new directory, or keep the default directory.

   A new project is created in the selected directory and opens in a new Vivado window.

3. Provide location constraints for the board.
4. In the Flow Navigator (left pane), click **Run Implementation** and follow the directions.

To see the results on the board, tap the pins assigned using the logic analyzer and observe the red, green, blue data signals incrementing by 1 for every valid clock cycle.

Example Design Directory Structure

In the current project directory, a new project with name `<component_name>_example` is created and the files are generated in the `<component_name>_example.src/sources_1/imports/<component_name>/` directory. This directory and its subdirectories contain all the source files required to create the AXI USB 2.0 Device example design. They also include the reference design clocking module, reset module, and example user design.

Table 5-2 shows the locations and names of example files.

Table 5-2: **Example Design Directory**

Directory	File Name	Description
`<component_name>/example_design`	`<component_name>_exdes.xdc`	Top level constraints file for the example design.
	`<component_name>_exdes.v`	The top-level HDL file for the AXI TFT controller example design
	`axi_slave_tb.v`	AXI slave interface logic
	`Clock_gen.v`	Clock generation modules
`hdl/src/example`	`ulpi_driver.v`	ULPI interface driver logic

The user constraints file is delivered in the example project directory:
`<component_name>_example.src/constrs_1/imports/example_design/<component_name>_exdes.xdc`

Simulating the Example Design

Using the AXI USB 2.0 Device example design delivered as part of the AXI USB 2.0 Device core, you can quickly simulate and observe the behavior of the AXI USB 2.0 Device core.

Setting up the Simulation

The Xilinx simulation libraries must be mapped into the simulator. If the libraries are not set for your environment, refer to the *Synthesis and Simulation Design Guide* (UG626) [Ref 3] for assistance compiling Xilinx simulation models and setting up the simulator environment. To switch simulators, click **Simulation Settings** in the Flow Navigator (left pane). In the Simulation options list, change **Target Simulator**.

This section contains instructions for running a functional simulation of the AXI TFT controller example design. The example design supports functional (behavioral) and post-synthesis simulations.

- To run a functional simulation, click **Run Simulation** in the Flow Navigator and then click **Run Behavioral Simulation**.
- To run a post-synthesis simulation, click **Run Simulation** in the Flow Navigator and then click **Run Post-Synthesis Functional Simulation**.

Simulation Results

The simulation script compiles the AXI USB 2.0 Device example design and supporting simulation files. It then runs the simulation and checks to ensure that it completed successfully.

If the test fails due to error, the following error message is displayed.

```
**************************************************************
******************** ERROR: TEST FAILED ***********************
********************** Timeout occurred ***********************
**************************************************************
```

If no failures occur, following message is displayed:

```
**************************************************************
******************** INFO: TEST PASSED ************************
**************************************************************
```

Test Bench

This chapter contains information about the test bench provided in the Vivado® Design Suite environment.

Figure 6-1 shows test bench for the AXI USB 2.0 Device example design. The top level test bench generates the top level 200MHz clock and drives initial reset to the example design. This block displays the test result in simulation.

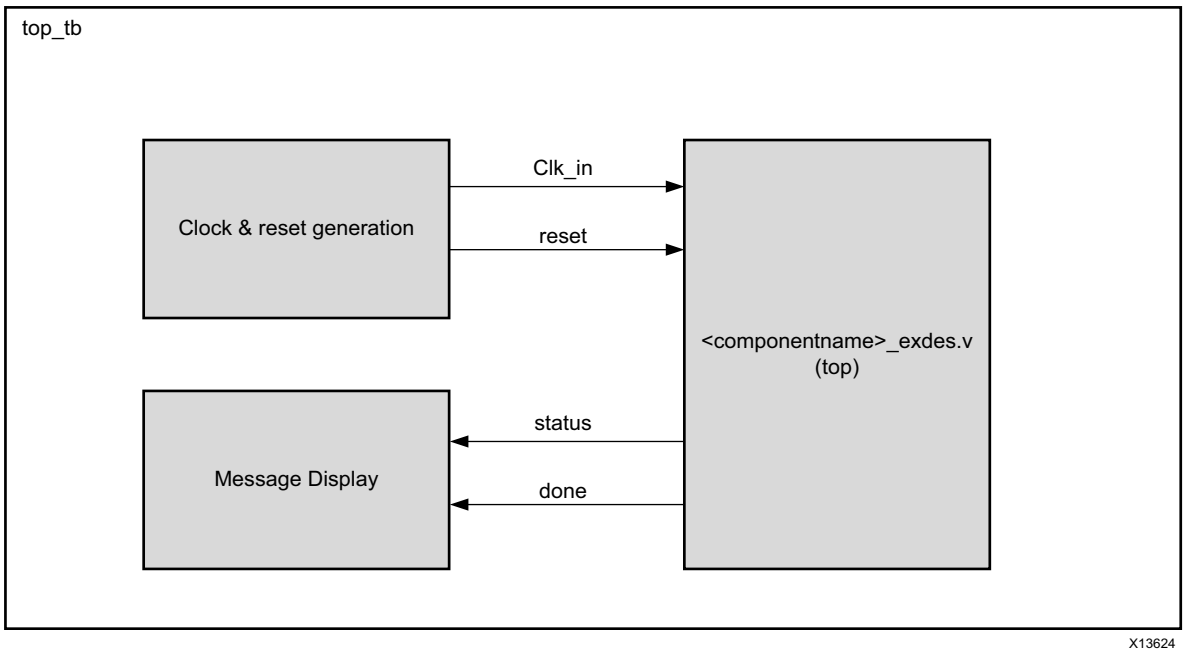

Figure 6-1: **AXI USB 2.0 Device Example Design Test Bench**

Migrating and Upgrading

This appendix contains information about migrating a design from ISE® to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information about migrating to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 7].

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

HSIC PHY support has been disabled. Please contact Xilinx Support for further details. The following parameters have been removed from the core.

- ENABLE HSIC PHY
- Connect Signaling Width

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

TIP: *If the IP generation halts with an error, there might be a license issue. See License Checkers in Chapter 1 for more details.*

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI USB 2.0 Device core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the AXI USB 2.0 Device core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

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 can 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.

Master Answer Record for the AXI USB 2.0 Device Core

AR 54432

Technical Support

Xilinx provides technical support in the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, Open a WebCase by selecting the WebCase link located under Support Quick Links.

Debug Tools

There are many tools available to address AXI USB 2.0 Device core design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 8].

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 Vivado lab tools are a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the Vivado lab tools for debugging the specific problems.

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation. Following are some general checks.

1. Check if all the location constraints are given correctly.
2. Check if the reset polarity is set as per the requirement of the PHY used.
3. Check if the USB PHY register space is accessible through writing and reading into the UPAR register.
4. If USB is detected by the host and not responding, check the USB control register contents and end-point configuration and status register contents.
5. Check if the interrupts are coming to the processor.
6. Check the suspended status bit in the ISR register to know if the core went to suspend state.
7. Use the Vivado lab tools to analyze the required signals in the IP.

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

References

Unless otherwise noted, IP references are for the product documentation page. To search for Xilinx documentation, go to Xilinx Support web page.

1. *Vivado Design Suite User Guide: Designing with IP* (UG896)
2. *Vivado AXI Reference Guide* (UG1037)
3. *Universal Serial Bus Specification, Revision 2.0* www.usb.org/developers/docs/
4. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
5. *Vivado Design Suite User Guide: Getting Started* (UG910)
6. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
7. *ISE to Vivado Design Suite Migration Guide* (UG911)
8. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/18/2015	5.0	Added support for UltraScale+ devices.
09/30/2015	5.0	Added 64 bit address width support when DMA support is enabled
11/19/2014	5.0	Removed HSIC PHY support
12/18/2013	5.0	Added UltraScale architecture support information
10/02/2013	5.0	• Added HSIC PHY information • Updated IP Facts table • Updated to Migrating and Upgrading appendix • Added Example Design and Test Bench chapters • Document version number advanced to match the core version number
03/20/2013	1.0	Initial Xilinx release of this product guide. This guide replaces DS785.

Please Read: Important Legal Notices