AXI External Peripheral Controller (EPC) v2.0

LogiCORE IP Product Guide

PG127 October 5, 2016

XILINX®

XILINX

Table of Contents

Introduction

The Xilinx LogiCORE™ IP AXI External Peripheral Controller (EPC) core supports data transfers between the AXI4 interface and the external synchronous and/or asynchronous peripheral devices such as USB and LAN devices, which have processor interface.

Examples of peripheral devices supported by the AXI EPC include the 10/100 non-PCI Ethernet single chip (SMSC LAN91C111) from SMSC and CY7C67300 USB Controller from Cypress Semiconductor devices.

Features

- Connects as a 32-bit slave on AXI4-Lite Slave
- Byte enable support
- Parameterized support of up to four external peripheral devices with each device configured with separate base address and high address range
- Supports both synchronous and asynchronous access modes
- Supports both multiplexed and non-multiplexed address and data buses
- Configurable peripheral data widths (8-bit, 16-bit, and 32-bit)
- Support of data width matching if the AXI data width is greater than that of peripheral data width
- Configurable timing parameters for peripheral bus interface

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale+™, UltraScale™, Zynq®-7000, 7 Series
Supported User Interfaces	AXI4-Lite
Resources	See Table 2-2.
Provided with Core	
Design Files	VHDL
Example Design	VHDL
Test Bench	VHDL
Constraints File	Not Provided
Simulation Model	Not Provided
Supported S/W Driver	N/A
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 derivative devices, see the Vivado IP catalog.
2. Standalone driver details can be found in the software development kit (SDK) directory <install_directory>/SDK.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

Functional Description

The AXI External Peripheral Controller (EPC) interface diagram shown in Figure 1-1 depicts the overall interfaces of the core design.

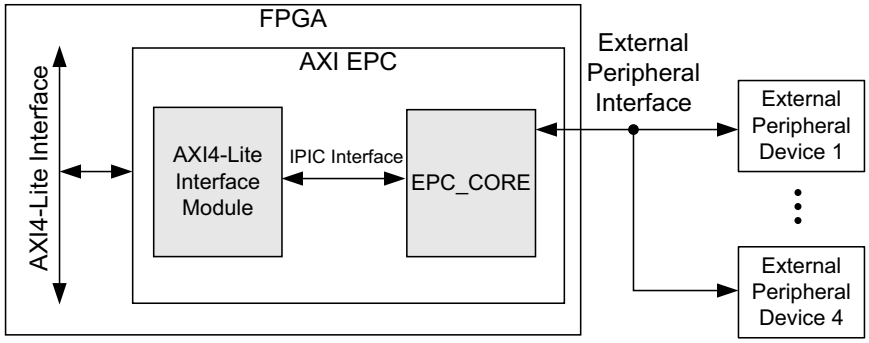

Figure 1-1: **AXI EPC Interface Diagram**

The AXI EPC IP core design provides a general purpose interface to external peripheral devices and the AXI4-Lite. It is a AXI4-Lite slave device. The AXI EPC IP core can be configured to provide support for multiple external peripherals (non-memory peripherals like USB, LAN, and etc.) up to a maximum of four devices. Each device is independently configured to respond either in synchronous or in asynchronous mode. You configure the timing parameters governing the access cycles such as setup/hold time, cycle access time, and cycle recovery time. It receives read or write operation commands from the AXI4 and generates a corresponding access cycle to one of the four peripheral devices.

RECOMMENDED: *Xilinx recommends that peripherals such as LAN and USB, which have embedded interfaces, be used with the core.*

The AXI EPC IP core is comprised of the following modules:

- AXI4-Lite Interface Module
- EPC_CORE

AXI4-Lite Interface Module

The AXI4-Lite interface module provides an interface between the EPC_CORE and the AXI4. The AXI4-Lite interface module implements the basic functionality of the AXI4 interface operation and performs the necessary protocol and timing translation between the AXI4 and the IPIC interface.

Note: The AXI4-Lite write access register is updated by the 32-bit AXI Write Data (`*_wdata`) signal, and is not impacted by the AXI Write Data Strobe (`*_wstrb`) signal. For a Write access, both the AXI Write Address Valid (`*_awvalid`) and AXI Write Data Valid (`*_wvalid`) signals should be asserted together.

EPC_CORE

The EPC_CORE provides an interface between the IPIC interface and the external peripheral devices. The EPC_CORE consists of the logic necessary to convert the access cycles on the IPIC interface to the corresponding access cycles on the peripheral bus adhering to the device specific timing parameters.

The block diagram for the AXI EPC is shown in Figure 1-2.

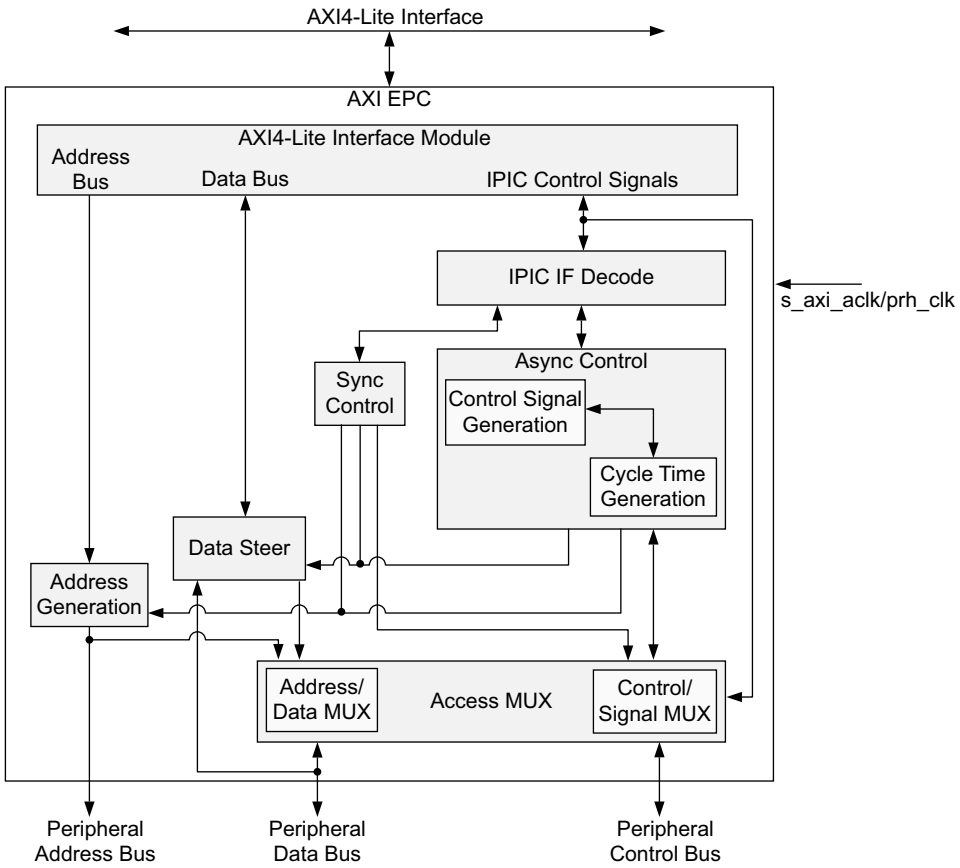

Figure 1-2: **AXI EPC Block Diagram**

The AXI EPC IP core consists of the following:

- **AXI4-Lite Interface** – Provides the address decoding logic and necessary interface between AXI4 and EPC core signals
- **IPIC IF Decode** – Provides decoding of IPIC signals of AXI4-Lite Interface Module and synchronization of control signals
- **Sync Control** – Implements the state machine which controls the synchronous interface
- **Async Control** – Implements the state machine which controls the asynchronous interface including the asynchronous timing parameters
- **Data Steer** – Provides the data bus width matching and data steering logic
- **Address Generation** – Provides the generation of the lower address bits
- **Access MUX** – Provides the multiplexing of address, data, and other control signals

A detailed description of these modules is provided in the following sections.

IPIC IF Decode Module

The IPIC IF Decode module implements the interface to the AXI. It also configures the EPC_CORE and interfaces to the Async Control module and Sync Control module by driving the necessary control signals based on your parameter settings.

Sync Control and Async Control Modules

In Figure 1-2 the Sync Control and Async Control modules depict the synchronous and asynchronous paths of the EPC_CORE. This ensures that the read and write accesses to the external device(s) adheres to the specific timing parameters defined for the external device(s). Implementation of the Sync Control and Async Control modules is dependent on the parameter Enable SYNC Mode. If synchronous and asynchronous external peripheral devices exist simultaneously, both the Sync Control and Async Control modules are implemented.

The Sync Control module operates either on the AXI clock (`s_axi_aclk`) or on the external peripheral clock (`prh_clk`) depending on the parameter EPC Clock. If EPC Clock = Peripheral Clock, then the Sync Control module operates on external device peripheral clock (`PRH_Clk`) that is different from the AXI clock. Peripheral clock (`prh_clk`) frequency should always be less than the AXI clock frequency. If more than one device is synchronous, then the frequency for the `prh_clk` should be chosen as the minimum of the operating frequencies of those devices.

The IPIC control signals that are inputs to the Sync Control module are synchronized to the `prh_clk` in the IPIC IF Decode module to indicate the start of a transaction. Similarly, the control signals from the Sync Control module to the IPIC interface such as data acknowledge are synchronized to the AXI clock in the IPIC IF Decode module.

If EPC Clock = AXI Clock, the Sync Control module operates on the AXI clock and the IPIC Decode module interface does not perform synchronization of control signals.

The Async Control module operates on the AXI clock only. This module generates the control signals to the initiate read and write access cycles to the external peripheral device based on your asynchronous timing parameter settings.

Data Steer and Address Generation Modules

The data bus of the external device must be ≤ AXI data width and can be 8, 16, or 32 bits. When the width of the external peripheral data bus is less than that of AXI and if Data Bus Width = 1 for a particular device, then the Data Steer module generates multiple read or write cycles to the external device to match a single access on the AXI.

To map a single 32-bit AXI access to multiple 8-bit or 16-bit accesses, the lower bits of the address bus are internally generated within the Address Generation module to provide the correct address to the external peripheral device. The address bus increments as each transaction completes.

For example, if the external device is eight bits wide, four read or write cycles to the device are performed to match a single 32-bit read or write transaction of AXI. For a write cycle, the first byte of the AXI data bus (`s_axi_wdata[7:0]`) is presented on the peripheral data bus (`prh_data[0:7]`). When the external device accepts the transaction, a new write cycle is generated and the second byte of the AXI data bus (`s_axi_wdata[17:8]`) is presented on the peripheral data bus (`prh_data[0:7]`) and etc. When the last byte of the AXI data bus (`s_axi_wdata[31:24]`) is accepted by the peripheral, the data acknowledge signal is generated and sent to the AXI4-Lite Interface module to indicate that the access is complete on the peripheral interface.

Similarly for a read cycle, when the external device indicates it is ready to complete the transaction, the data on the peripheral data bus (`prh_data[0:7]`) is internally registered as the first byte to be presented to AXI data bus (`s_axi_rdata[7:0]`) followed by initiation of new read cycle on the peripheral interface. The second read access on the peripheral data bus (`prh_data[0:7]`) is internally registered as the second byte to be presented to AXI data bus (`s_axi_rdata[15:8]`) and etc. When all four bytes are read from the external device, an acknowledge is generated to the AXI4-Lite interface module to indicate that the data is ready to be transferred to AXI data bus.

When support for data width match is enabled for any of the external devices, then the access to that device should respect data alignment that is a half word access should be aligned to a 16-bit boundary and a word access should be aligned to a 32-bit boundary.

Access MUX Module

The interface to the external peripherals supports both multiplexed and non-multiplexed address and data bus to the external devices. The Access MUX module controls the multiplexing of the peripheral address and data buses based on the parameter Enable ADDR/DATA Multiplexing. If Enable ADDR/DATA Multiplexing = 1, the address and the data bus are multiplexed and presented to the corresponding external device on `prh_data` bus. The address is valid on the `prh_data` bus as long as the address strobe is active (`prh_ads`). This access is performed in two phases (address phase and data phase). The data phase is followed by the address phase. If Enable ADDR/DATA Multiplexing = 0, the address and data are presented to the device on separate buses (`prh_addr` bus and `prh_data` bus) and the access cycle contains only one phase.

Applications

Example Peripheral Connection

An example peripheral device supported by the AXI EPC IP core is the SMSC 10/100 non-PCI Ethernet Single Chip MAC + PHY LAN91C111. The AXI EPC IP core allows AXI masters to access the device both in synchronous and asynchronous mode. The device connections for synchronous and asynchronous mode of operations are outlined in the Asynchronous Mode and Synchronous Mode sections.

Asynchronous Mode

Figure 1-3 illustrates the SMSC LAN91C111 connections to the AXI EPC IP core in asynchronous mode with non-multiplexed address and data buses. In asynchronous mode, the input clock source (`prh_clk`) to the external device must be tied to Vcc. When the SMSC LAN91C111 is accessed through the asynchronous mode, the rising edge of `PRH_Rd_n` and `PRH_Wr_n` signals are used to control the read and the write operations, respectively. As the device uses Asynchronous Ready (`ARDY`) signal to indicate its readiness in asynchronous mode. This signal is connected to the `prh_rdy` input of the AXI EPC IP core. Because the address and the data buses are not multiplexed, `prh_ads` is driven Low by the AXI EPC IP core. The timing parameters should be set according to the SMSC LAN91C111 Async mode specifications.

The following parameter settings apply for the connections shown in Figure 1-3.

- EPC Clock = AXI Clock = 0
- Peripheral-0 Address Width = 16
- Peripheral-0 Data Bus Width = 32
- Peripheral-0 Enable SYNC Mode = 0

- Peripheral-0 Enable Data Width Matching = 0
- Peripheral-0 Enable ADDR/DATA Multiplexing = 0

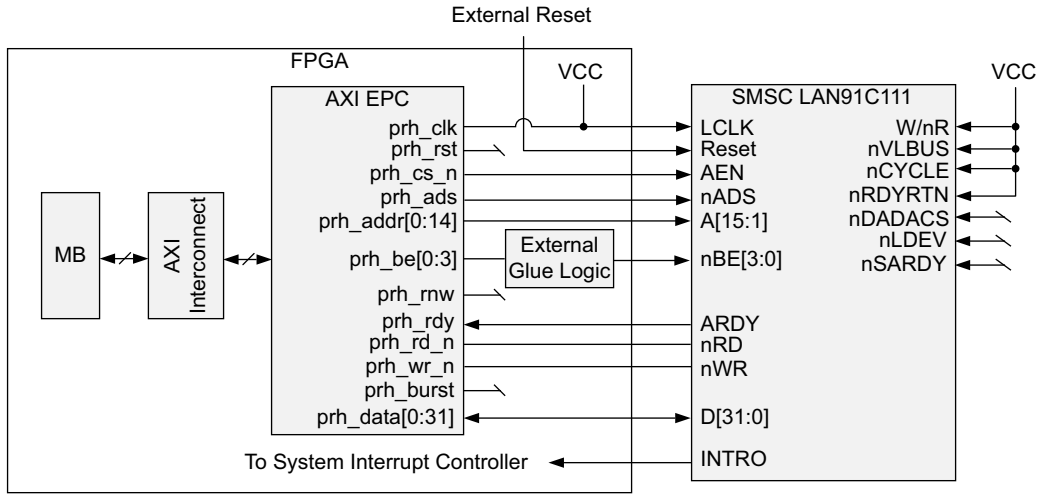

Figure 1-3: **SMSC LAN91C111 Connection to AXI EPC IP Core in Asynchronous Mode**

External FPGA Logic

IMPORTANT: *Special attention must be given while interfacing the AXI EPC IP core to the SMSC LAN91C111 in asynchronous mode.*

The AXI EPC IP core drives the `prh_be` active-High while the SMSC LAN91C111 requires the input byte enable to be active-Low. To match this requirement, external FPGA logic is required to invert the `prh_be` signals before they are interfaced with the byte enable signals of the SMSC LAN91C111.

Synchronous Mode

Figure 1-4 illustrates the example for SMSC LAN91C111 connection to the AXI EPC IP core in synchronous VL Bus mode with non-multiplexed address and data buses. In synchronous mode, the device requires an input clock source to the LCLK pin with a maximum operating frequency of 50 MHz. The local clock generated by the DCM for the peripheral interface must be routed to the LCLK input of SMSC LAN91C111 and `prh_clk` of AXI EPC IP core. When the device is accessed, `prh_rnw` is used as the control signal to indicate a read/write access. When `PRH_RNW` is High, it indicates a read operation and when Low, it indicates a write operation. The SMSC LAN91C111 uses `nSRDY` signal to indicate the device readiness. This signal is connected to `PRH_Rdy` input of the AXI_EPC. As the address and the data bus are not multiplexed, `prh_ads` is driven Low.

The following parameter settings apply for the connections shown in Figure 1-4.

- EPC Clock = AXI Clock = 1
- Peripheral-0 Address Width = 16
- Peripheral-0 Data Bus Width = 32
- Peripheral-0 Enable SYNC Mode = 1
- Peripheral-0 Enable Data Width Matching = 0
- Peripheral-0 Enable ADDR/DATA Multiplexing = 0

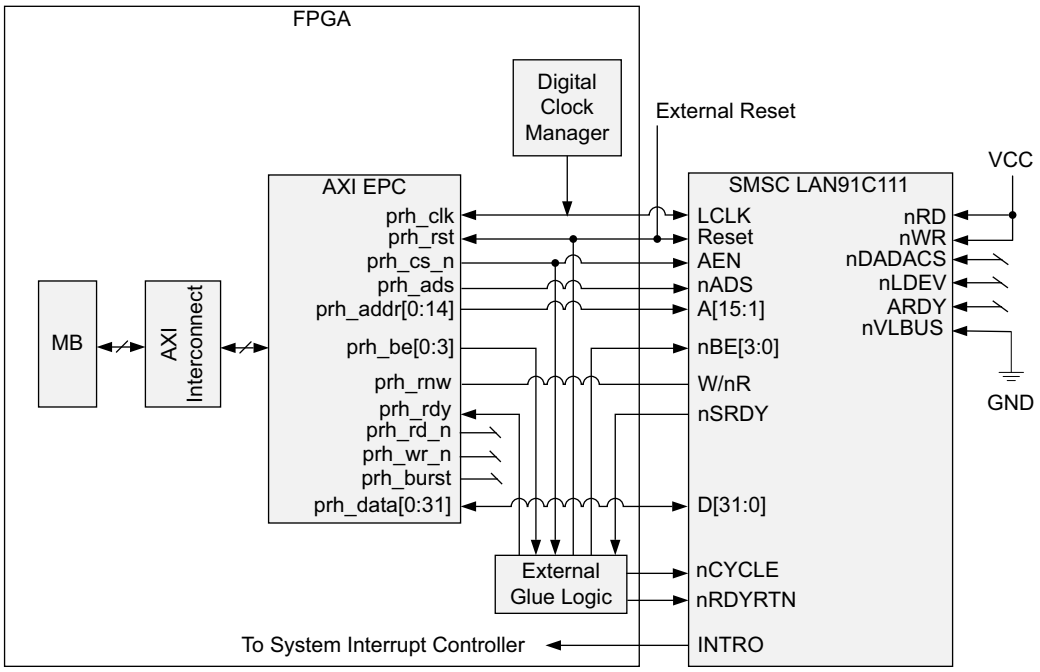

Figure 1-4: **SMSC LAN91C111 Connection to AXI EPC IP Core in Synchronous Mode**

External FPGA Logic

The SMSC LAN91C111 device requires two inputs, `nCYCLE` and `nRDYRTN`, to control synchronous operation. The definition of these signals is outside the scope of this document and can be found in the document listed in [Ref 1]. The `nCYCLE` and `nRDYRTN` signals are generated in external FPGA logic. `nCYCLE` can be generated from `prh_cs_n` and is driven Low for a single clock cycle on a High to Low transition on `prh_cs_n`. `nRDYRTN` is the registered value of the `nSRDY` signal.

In case of synchronous multiplexing mode, the generation of `nCYCLE` signal is little different. The `prh_cs_n` signal is generated twice from the core, first during the address phase and second time during the data phase. You should take care while writing the logic for `nCYCLE` signal generation in external FPGA logic. The `nCYCLE` should be generated in data phase when there is `prh_cs_n` asserted and `prh_ads` is not present.

Design Considerations

When the AXI EPC IP core is interfaced to the SMSC LAN91C111 with data width match enabled (Enable Data Width Matching = 1) and the internal SRAM of the peripheral is being accessed, the device has to be configured in the address auto increment mode. Address auto increment mode can be configured by setting "AUTO INCR" bit in the Bank 2 Pointer register of SMSC LAN91C111. For more information on the registers of Bank 2, see the *SMSC LAN91C111, 10/100 Non-PCI Ethernet Single Chip MAC + PHY Data Sheet* [Ref 1].

Example Peripheral Connection

The Cypress Semiconductor CY7C67300 EZ-Host™ Programmable Embedded USB Host/ Peripheral Controller is another example of a peripheral device supported by the AXI EPC IP core. Because the Cypress USB controller operates in asynchronous mode, the AXI EPC IP core must be configured to support asynchronous mode operation.

Asynchronous Mode

Figure 1-5 illustrates the CY7C67300 USB Controller connection to the AXI EPC IP core in asynchronous mode with non-multiplexed address and data buses. This interface is tested in hardware.

The following parameter settings apply for the connection shown in Figure 1-5.

- EPC Clock = AXI Clock = 0
- Peripheral-0 Address Width = 4
- Peripheral-0 Data Bus Width = 16
- Peripheral-0 Enable SYNC Mode = 0
- Peripheral-0 Enable Data Width Matching = 1
- Peripheral-0 Enable ADDR/DATA Multiplexing = 0

Configuration of CY7C67300 USB Controller

The CY7C67300 USB Controller is configured as a Host Processor Interface (HPI). The boot-up sequence code for the CY7C67300 USB Controller in this particular example is present on the external CompactFlash memory device and is accessed through the AXI SYSACE interface. This code is downloaded to the CY7C67300 USB Controller through the HPI interface. After this boot-up sequence is executed by the CY7C67300 USB Controller, it becomes ready to operate in normal HPI mode with the external host controller (AXI EPC IP core). The AXI EPC address bits which corresponds to word boundary are only used in the 2-bit address bus interface with the Cypress Semiconductor USB device.

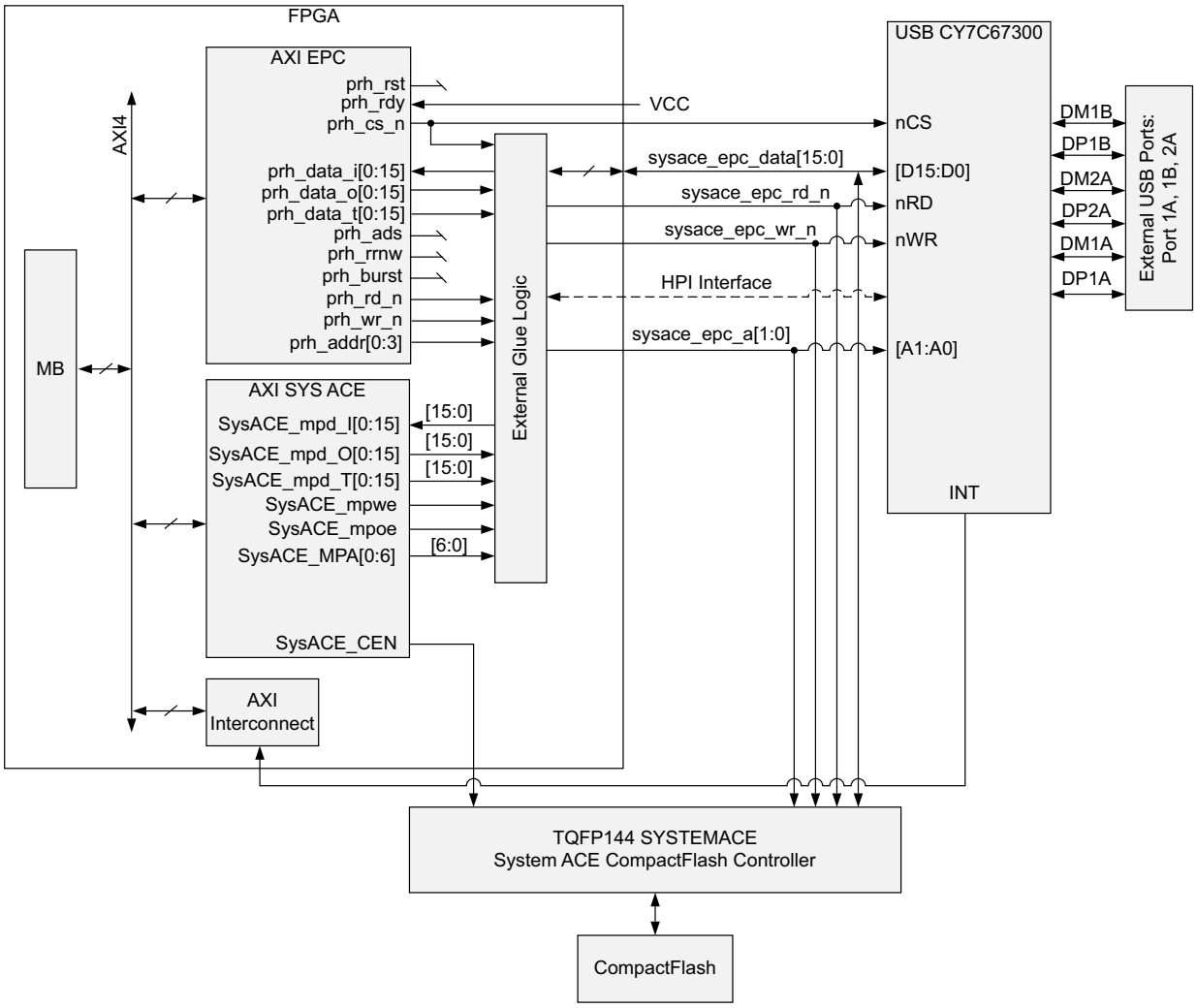

Figure 1-5: **CY7C67300 USB Controller Connection to AXI EPC in Asynchronous Mode**

External FPGA Logic

In the interface diagram example shown in Figure 1-5, the AXI EPC IP core, as well as the AXI SYSACE share the interface for data, address, and control lines on the board. This requires external multiplex logic to be placed outside of IP cores in the FPGA. The AXI EPC IP core control signals `prh_rd_n` and `prh_wr_n` are MUXed with the `MEM_CEN` and `MEM_WEN` signals of the AXI SYSACE. When CY7C67300 USB Controller is accessed through the asynchronous mode, the rising edge of `sysace_epc_rd_n` and `sysace_epc_wr_n` signals are used to control the read and the write operations respectively. For more information on AXI SYSACE, see the CY7C67300, EZ-Host™ Programmable Embedded USB Host/Peripheral Controller Data Sheet [Ref 2].

Unsupported Features

Many peripheral devices have the device specific input/output ports such as status, remote reset, remote wake-up, and interrupts. The AXI EPC IP core does not have any provision to support these device specific input/output ports. Therefore, if the external device has any such device specific ports, then these input/output ports can be connected directly to system general purpose input output controller or to the system interrupt controller. If the external device has interrupt capability, then the interrupt outputs of the external device should be connected directly to the system interrupt controller.

Many peripheral devices support DMA capability. However, the AXI EPC IP core is a AXI slave device and does not support DMA operations from the external peripheral devices.

Licensing and Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx Vivado Design Suite under the terms of the Xilinx End User License.

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

Product Specification

Standards

The AXI EPC IP core is AXI4 and AXI4-Lite compliant.

Performance

The AXI EPC IP is characterized as per the benchmarking methodology described in "Vivado IP Optimization (F_{Max} Characterization) appendix," in the *Vivado Design Suite User Guide: Designing With IP* (UG896) [Ref 3]. Table 2-1 shows the results of the characterization runs.

Note: Maximum frequency numbers for Zynq®-7000 devices are expected to be similar to 7 series device numbers.

Table 2-1: **Maximum Frequencies**

FPGA	Speed Grade	F_{Max} (MHz)
		AXI4/AXI4-Stream
Virtex-7	–1	180
Kintex-7		180
Artix-7		120
Virtex-7	–2	200
Kintex-7		200
Artix-7		140
Virtex-7	–3	220
Kintex-7		220
Artix-7		160

Resource Utilization

Note: Resources numbers for UltraScale™ architecture and Zynq-7000 devices are expected to be similar to 7 series device numbers.

7 Series FPGAs

Table 2-2 provides approximate resource counts for the various core options using 7 series FPGAs.

Table 2-2: **Device Utilization – 7 Series FPGAs**

Parameter Values (Other Parameters at Default Value)							Device Resources		
No. of Peripherals	EPC Clock	Address Width	Data Bus Width	Enable SYNC Mode	Enable ADDR/DATA Multiplexing	Enable Data Width Matching	Slices	Slice Flip- Flops	LUTs
1	AXI Clock	3	32	0	0	0	54	153	109
2	AXI Clock	12, 12	32, 32	0, 0	0, 1	0, 0	82	187	182
2	Peripheral Clock	12, 12	32, 32	1, 1	0, 1	0, 0	63	150	135
3	AXI Clock	6, 12, 12	32, 16, 8	0, 0, 0	0, 0, 0	1, 1, 1	63	123	132
3	AXI Clock	12, 12, 12	32, 16, 8	1, 1, 1	0, 0, 0	1, 1, 1	135	179	323
3	Peripheral Clock	12, 12, 12	32, 16, 8	1, 1, 1	1, 1, 1	1, 1, 1	116	156	296
4	AXI Clock	12, 12, 12, 12	16, 8, 16, 32	0, 0, 0, 0	0, 0, 0, 0	1, 1, 1, 1	133	171	334
4	AXI Clock	12, 12, 12, 12	32, 8, 32, 8	0, 0, 0, 0	0, 0, 0, 0	0, 0, 0, 0	143	187	332

1. The bus multiplex timing parameters and asynchronous timing parameters are set to typical values for all devices.

Port Descriptions

The AXI EPC IP core I/O signals are listed and described in Table 2-3.

Table 2-3: **AXI EPC IP Core I/O Signal Description**

Signal Name	Interface	I/O	Initial State	Description
System Signals				
s_axi_aclk	AXI	I	–	AXI clock
s_axi_aresetn	AXI	I	–	AXI reset. Active-Low.
s_axi_*	S_AXI	–	–	AXI4-Lite Slave Interface signals. See Appendix A of the *AXI Reference Guide* (UG761) [Ref 4] for AXI4, AXI4-Lite, and AXI4-Stream signals.
AXI EPC Signals				
prh_clk(1)	EPC	I	–	External peripheral clock input
prh_rst	EPC	I	–	External peripheral reset input
prh_cs_n	EPC	O	1	External peripheral chip select. Active-Low signal
prh_addr(2)	EPC	O	–	External peripheral address bus
prh_ads	EPC	O	0	External peripheral address strobe in case of multiplexed address and data bus. Active-High signal
prh_be	EPC	O	–	External peripheral byte enables
prh_rnw	EPC	O	1	External peripheral read/write signal for synchronous access mode
prh_rd_n	EPC	O	1	External peripheral read signal for asynchronous access mode. Active-Low signal
prh_wr_n	EPC	O	1	External peripheral write signal for asynchronous access mode. Active-Low signal
prh_burst	EPC	O	0	Burst cycle indication to external peripheral. Active-High signal
prh_rdy	EPC	I	–	Peripheral ready signal. This signal is used by the external peripheral to extend the transaction. Active-High signal
prh_data_i	EPC	I	–	External peripheral input data bus
prh_data_o	EPC	O	–	External peripheral output data bus

Table 2-3: **AXI EPC IP Core I/O Signal Description *(Cont'd)***

Signal Name	Interface	I/O	Initial State	Description
prh_data_t	EPC	O	–	3-state control for external peripheral output data bus

Notes:

1. prh_clk is utilized only when EPC Clock is set to Peripheral Clock and the interface to the external peripheral is synchronous.
2. External peripheral devices are considered as byte addressable irrespective of the data bus width.

Additional Design Information

FIFO Transactions

When Enable FIFO Access = 1, the AXI EPC IP core supports access to FIFOs within the external peripheral devices. When the FIFO within the external device is accessed, then the peripheral address bus (`prh_addr`) must remain constant representing the FIFO address within the address range assigned to the peripheral device. When data width matching is enabled and the access corresponds to the FIFO, the AXI EPC IP core does not increment the peripheral address (`prh_addr`) bus.

Abnormal Terminations

If `Prh_rdy` is deasserted for more than the maximum period as specified by Maximum Ready Assertion Period, then the AXI EPC terminates the current access to the external device and signals slave error to the AXI master by asserting either `s_axi_bresp` or `s_axi_rresp` on the AXI. In this case, the access to the external device is terminated abnormally. Therefore, the external device can be in an indeterminate state and the exceptions should be handled appropriately at the system level.

Interrupt Handling

If the external device has interrupt capability, then the interrupt outputs of the external device should be connected directly to the system interrupt controller.

Device Ready Signal (prh_rdy Signal)

The AXI EPC IP core read/write access cycles are executed only when the external device assert the device ready signal (`prh_rdy`).

As the `prh_rdy` signal becomes `prh_rdy`, this indicates that the peripheral device is ready for communication. After the `prh_rdy` goes active, it is expected that it remains in the same active state until that particular transaction is completed by the AXI EPC IP core.

If the AXI EPC IP core is implemented with data width match enabled, then the activeness of the prh_rdy signal is always checked for every transaction, regardless of it being active throughout the transaction.

You should see the data sheet for the external peripheral, to define the value of prh_rdy time period (that is, Ready Valid Period parameter value should be filled based on the mentioned information). It is also expected that you provide maximum waiting period for prh_rdy signal. This waiting period is indicated by Maximum Ready Assertion Period parameter. The Ready Valid Period parameter value should be less than the Maximum Ready Assertion Period parameter value. The Maximum Ready Assertion Period time period should be lower than the AXI IPIF timeout value if C_DPHASE_TIMEOUT > 0 (that is, Ready Valid Period < Maximum Ready Assertion Period < C_DPHASE_TIMEOUT). Starting from the falling edge of the prh_wr_n or, prh_rd_n signal, the internal counter for Ready Valid Period and Maximum Ready Assertion Period starts. The AXI EPC IP core expects that the prh_rdy signal should appear before any of these timer ends. If the prh_rdy does not become active before the Ready Valid Period counter ends, then the AXI EPC IP core waits until the end of Maximum Ready Assertion Period counter. In case prh_rdy becomes active, then internal state machine proceeds to complete the further process steps. If prh_rdy does not become active even before the Maximum Ready Assertion Period ends, the AXI EPC IP core generates an error and terminate the transaction. You can re-initiate the same transaction later on.

Figure 2-1 shows the diagrammatic representation of how the Async state machine reacts with the prh_rdy signal. Consider this figure as representation only. In actual design, there are some dummy states added to meet the design requirements. In case of asynchronous reset to the core, all states go to IDLE by default.

If the external peripheral device does not have a device ready signal, then the prh_rdy input of the AXI EPC for that particular device must be tied to logic High.

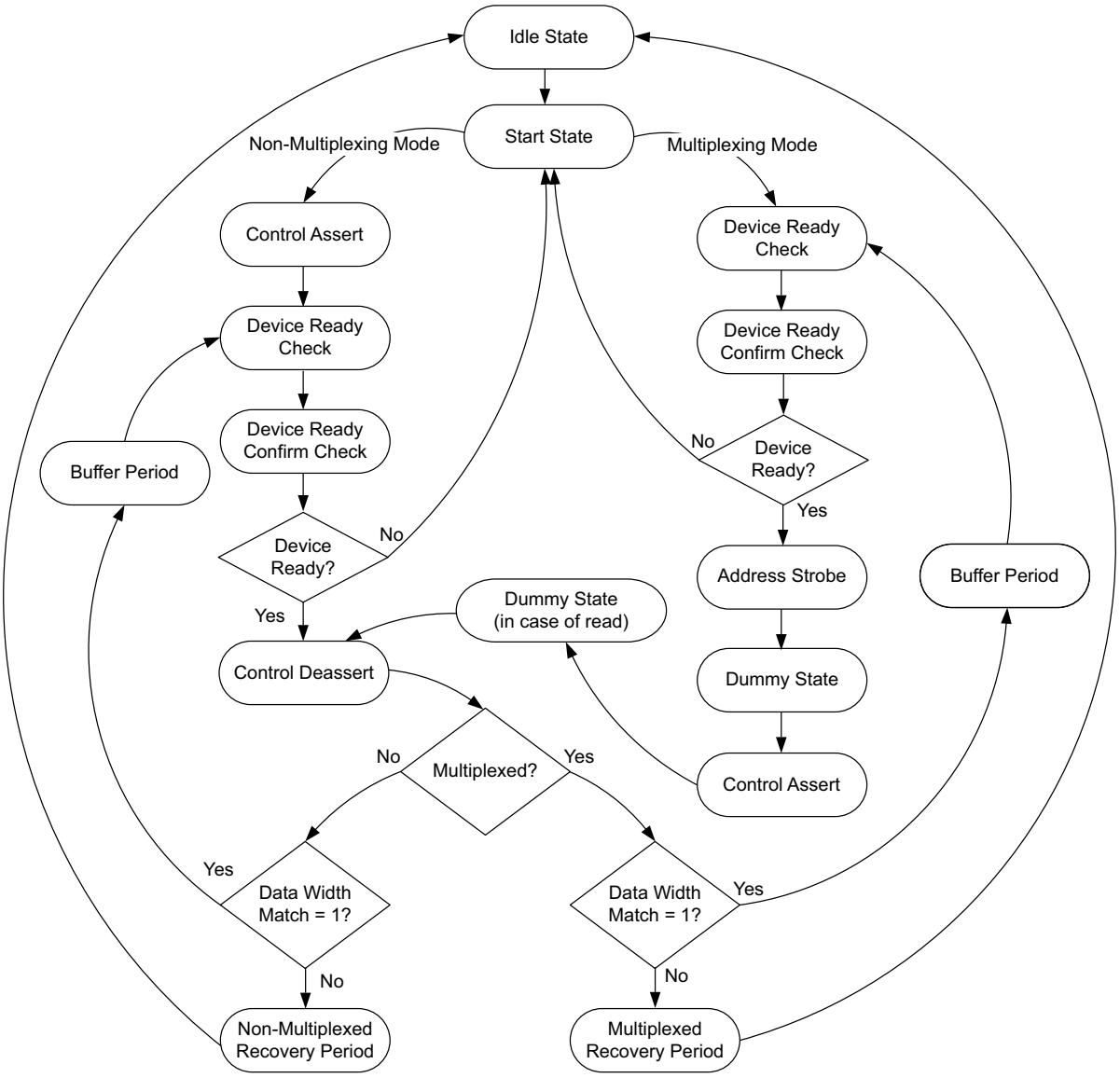

Figure 2-1: **Async State Machine Flow**

Peripheral Data Bus Mapping

The peripheral data bus (`prh_data`) uses big endian bit labeling (that is, bit-0 is Most Significant Bit (MSB) and bit-31 is Least Significant Bit (LSB) for a 32-bit bus) and is sized according to the Maximum AD Width parameter. Peripherals that have smaller widths should connect to this bus starting at bit-0 (MSB). For example, if three external devices are present in the system with data bus width of 8, 16, and 32 bits, the 8-bit device should connect to `prh_data[0:7]`, the 16-bit wide peripheral should connect to `prh_data[0:15]`, and the 32-bit peripheral should connect to `prh_data[0:31]`.

The bit and byte labeling for the big endian data types is shown in Figure 2-2.

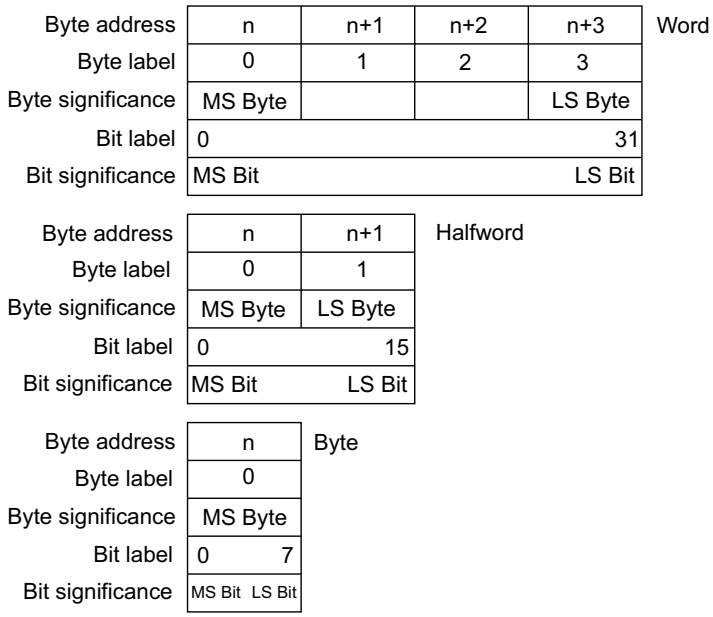

Figure 2-2: **AXI EPC Big Endian Data Type**

Designing with the Core

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

Design Considerations

The AXI EPC IP core is AXI slave device. It receives read or write instructions from the processor and generates a corresponding access cycle on the peripheral interface. Examples of read and write accesses are illustrated in the Timing Diagrams section. You must take the following considerations into account while designing with the AXI EPC IP core.

Timing Parameters in Async Mode of Operation

Internally in the AXI EPC IP core, the timing parameters are inter-related and some calculation is done based on the values you provided. In the asynchronous mode of operation, the following parameters are used:

- **Address Setup Period** – Address setup time
- **Address Hold Period** – Address hold time
- **Minimum Width of Address Strobe** – Address strobe width
- **Chip Select Setup Period** – Chip select setup time
- **Chip Select Hold Period** – Chip select hold time
- **Minimum Write Pulse Width** – Write control width time
- **Minimum Write Cycle Period** – Write cycle time (that is, time between two consecutive writes)
- **Data Bus Setup Period** – Data setup time
- **Data Bus Hold Period** – Data hold time
- **Minimum Read Pulse Width** – Read control width time
- **Minimum Read Cycle Period** – Read cycle time (that is, time between two consecutive reads)

- **Minimum Read Data Valid Period** – Time taken by the data to be out at read condition
- **Read Data High-Z Period** – Data line 3-state after the read control signal is de-activated
- **Ready Valid Period** – Time for device ready signal to go High, after device is selected
- **Maximum Ready Assertion Period** – Maximum time until the AXI EPC IP core can wait for device to be ready

These parameters are linked in the design as follows:

- **Address Hold Time** – Parameter is calculated as:
(Address Hold Period / AXI Clock Period)
- **Chip Select/Data/Address Hold Time** – Three parameters are used to calculate the hold time as:
((max2(max2(Data Bus Hold Period, Chip Select Setup Period), Address Hold Period)/ AXI Clock Period)
- **Read Control Signal Width Time** – Parameter is calculated as:
(Minimum Read Pulse Width / AXI Clock Period)
- **Write Control Signal Width Time** – Parameter is calculated as:
(Minimum Write Pulse Width / AXI Clock Period)
- **Address Strobe Signal Width Time** – Three parameters are used to calculate the address strobe width as:
((max2(max2(Address Setup Period, Minimum Width of Address Strobe), Maximum Chip Select Width)/ AXI Clock Period)
- **Write Recovery Time for Non-Multiplexed Case** – Parameter is calculated as:
((max2(max2(Chip Select Setup Period, Data Bus Hold Period), max2(Address Hold Period, PRH_Wr_nx)) / AXI Clock Period)
Where PRH_Wr_nx = (Minimum Write Cycle Period - Minimum Write Pulse Width)
- **Read Recovery Time for Non-Multiplexed Case** – Parameter is calculated as:
((max2(max2(Chip Select Setup Period, Read Data High-Z Period), max2(Address Hold Period, PRH_Rd_nx)) / AXI Clock Period)
- **Write Recovery Time for Multiplexed Case** – Parameter is calculated as:
((max2(Chip Select Setup Period, Data Bus Hold Period), PRH_Wr_nx)) / AXI Clock Period)
Where PRH_Wr_nx = (Minimum Write Cycle Period - Minimum Write Pulse Width)
- **Read Recovery Time for Multiplexed Case** – Parameter is calculated as:
((max2(Chip Select Setup Period, Read Data High-Z Period), PRH_Rd_nx)) / AXI Clock Period)
Where PRH_Rd_nx = (Minimum Read Cycle Period - Minimum Read Pulse Width)

- **Device Ready Signal Time** – Parameter is calculated as: (Ready Valid Period / AXI Clock Period)
- **Maximum Time for Device Ready Signal to be Active** – Parameter is calculated as: (Maximum Ready Assertion Period / AXI Clock Period)

Some of the calculations are repeated for different uses for internal purpose. There are some dummy states included in between the states to meet the design requirements. You should make sure that these timing parameter value calculations do not exceed the AXI timeout value (C_DPHASE_TIMEOUT) if C_DPHASE_TIMEOUT > 0.

Table 3-1 indicates the ranges for the mentioned parameters.

Table 3-1: **Example of Timing Ranges for AXI EPC IP Core in Asynchronous Mode of Operation**

Parameter Name	Timing Parameter Range		
Address Setup Period	6,000(set 1)	20,000(set 2)	40,000(set 3)
Address Hold Period	6,000	20,000	30,000
Minimum Width of Address Strobe	10,000	20,000	40,000
Chip Select Setup Period	6,000	20,000	40,000
Chip Select Hold Period	6,000	20,000	30,000
Minimum Write Pulse Width	15,000	30,000	30,000
Minimum Write Cycle Period	30,000	60,000	60,000
Data Bus Setup Period	10,000	20,000	30,000
Data Bus Hold Period	5,000	20,000	30,000
Minimum Read Pulse Width	15,000	30,000	30,000
Minimum Read Cycle Period	30,000	60,000	60,000
Minimum Read Data Valid Period	5,000	5,000	15,000
Read Data High-Z Period	10,000	15,000	25,000
Ready Valid Period	10,000(set 1)	50,000(set 2)	60,000(set 3)
Maximum Ready Assertion Period	50,000	100,000	120,000

Notes:

1. The above range of timing parameters (set 1) are used for internal device utilization and while testing on USB and LAN. While using the parameters, care has been taken that all parameters of same set are used.
2. The above timing sets (set 2 and set 3) are just an example of how the timing parameters can be given to AXI EPC IP core. You can have different timing parameter values. In such cases, note the calculation given above.

External Peripheral Connections

The AXI EPC IP core interface to the external device is based upon the width of the AXI data bus, address and data width of the peripheral subsystem, number of peripherals in the system, address/data multiplex support, mode of operation (synchronous or asynchronous) and if synchronous device, the operating clock for the synchronous datapath.

Determining Address and Data Width

The address bus width of the peripheral subsystem is the maximum width of the address bus of all peripheral devices connected to the AXI EPC IP core. If all devices employ non-multiplexed address and data bus, the data bus width of the peripheral subsystem is the maximum of the data bus width of all external devices. If any of the devices are configured for multiplexed address and data bus, then the data bus width of the peripheral subsystem should be set as the maximum of the data bus and the address bus of the device(s) employing a multiplexed address and data bus.

Endian Considerations

The peripheral address and data bus of the AXI EPC IP core is labeled with big endian bit labeling (D0 is the MSB and D31 is the LSB for a 32-bit bus) while most peripheral devices are either endian agnostic (that is, they can be connected either way or little endian (D31 is the MSB and D0 is the LSB for a 32-bit data bus)).

CAUTION! *Caution must be exercised with the connection to the external peripheral devices to avoid incorrect data and address connections. See Data Steer and Address Generation Modules for a data steering example.*

Clocking

When connecting to a synchronous external device, the AXI EPC IP core can operate either on the AXI clock (`s_axi_aclk`) or on a peripheral clock (`prh_clk`). The operating clock for the synchronous interface is based on the generic `epc_clock`. If `epc_clock` = 1, the peripheral clock is used else the AXI clock is used. The external devices connected to the AXI EPC IP core determines the operating frequency of the peripheral clock. The minimum of the operating frequencies of various devices connected to AXI EPC IP core should be used as the operating frequency of the peripheral clock (`prh_clk`).

Figure 3-1 illustrates a block diagram of one of the methods for generating a device specific clock source using two DCMs. The DCM modules are located within the FPGA but are external to the AXI EPC IP core. An external clock source is used to generate the AXI clock (`s_axi_aclk`) to the AXI EPC. The device clock source (`prh_clk`) is also generated from the same DCM (DCM0) using the CLKFX output and must be routed on the board to be fed back to the FPGA. The parameters of DCM0 like C_CLKFX_MULTIPLY and C_CLKFX_DIVIDE should be set according to the required frequency output at CLKFX pin of DCM0. Another DCM (DCM1) is used to synchronize the device peripheral clock to the `prh_clk` signal of AXI EPC.

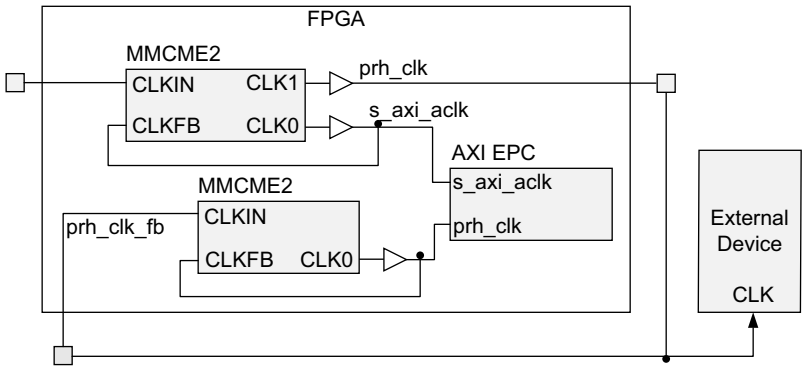

Figure 3-1: **External Peripherals Clocked by FPGA Output with Feedback**

Resets

The AXI EPC IP core uses a single active-Low reset associated with AXI Slave interface. The reset signal `s_axi_aresetn` is a synchronous reset input that resets the whole AXI EPC IP core.

Protocol Description

Timing Diagrams

The timing diagrams in the figures show various AXI transactions and the resulting access cycles on the peripheral interface. Timing diagrams are not shown for all possible combinations of generics and the resulting access cycles. However, the timing diagrams shown are sufficient for the basic understanding of various access cycles supported by the AXI EPC IP core.

Figure 3-2 shows the write-read transactions to device memory when the bus is not multiplexed and data width matching is disabled (EPC Clock = AXI Clock). This assumes the peripheral device is ready.

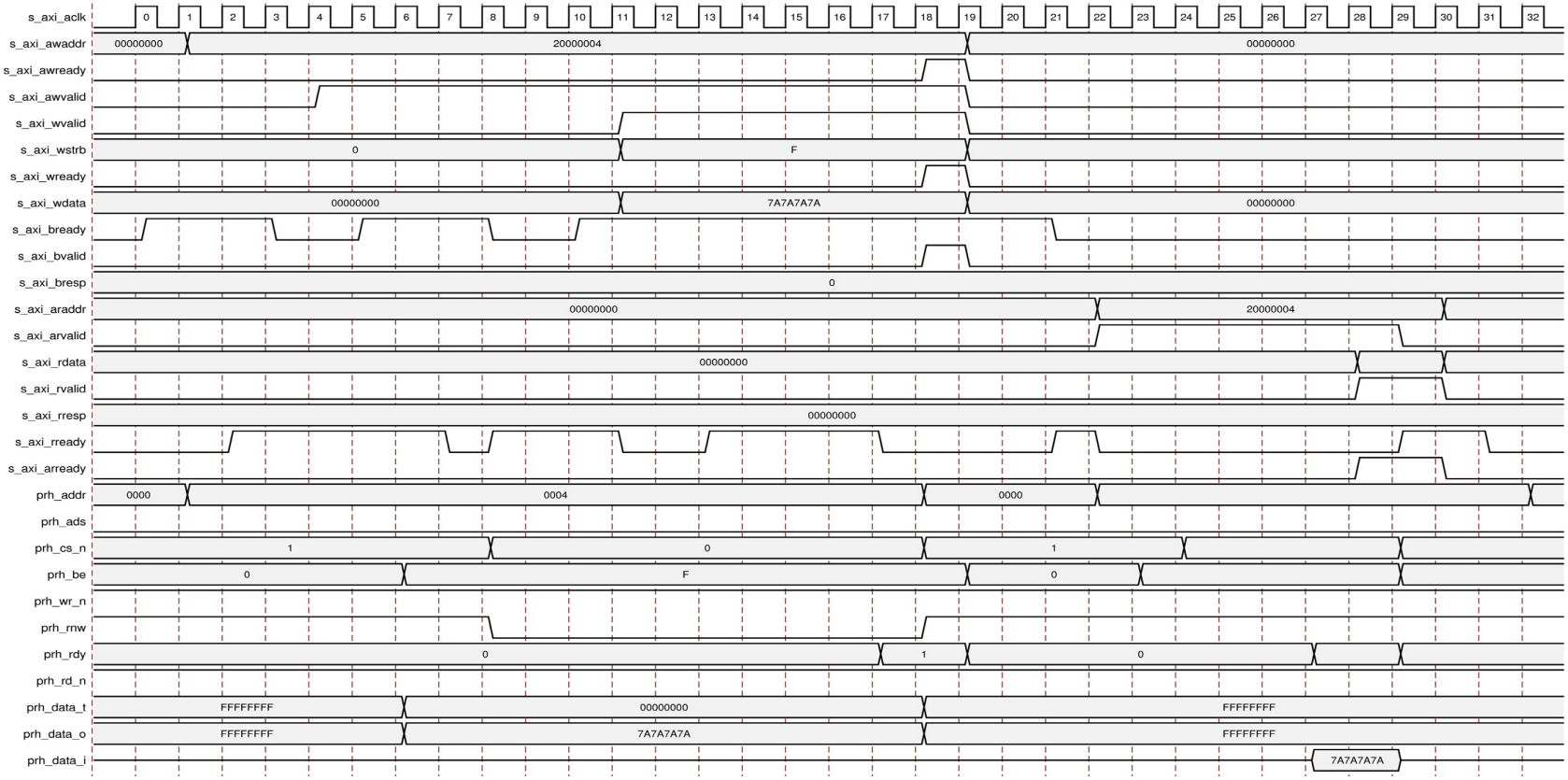

Figure 3-2: **Synchronous Write-Read Transactions to Device Memory**

Figure 3-3 shows the synchronous write and read transactions to device memory when the bus is multiplexed and data width matching is enabled (EPC Clock = AXI Clock).

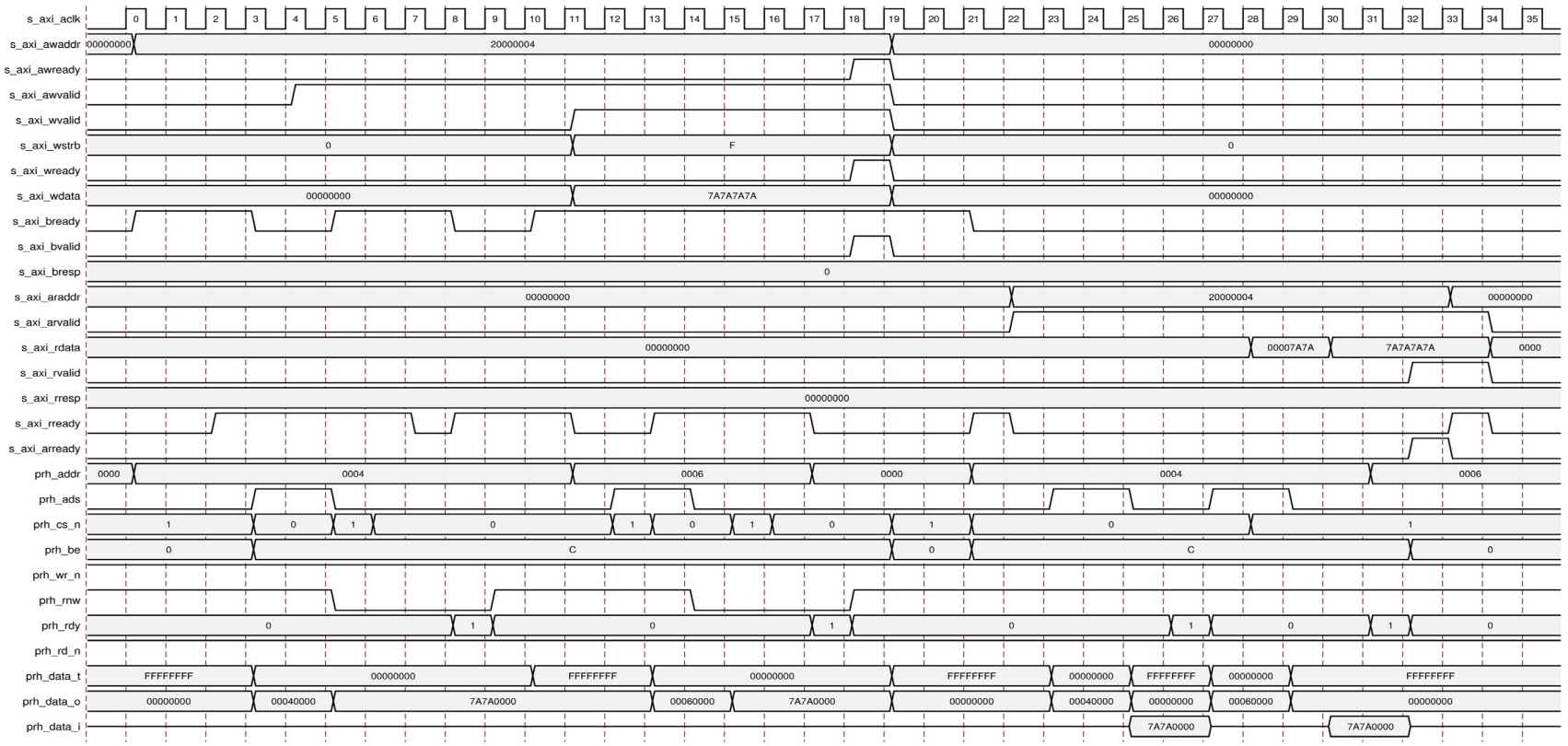

Figure 3-3: **Synchronous Write and Read Transactions to Device Memory**

Figure 3-4 shows the synchronous read and write transactions to device memory when the bus is not multiplexed and data width matching is enabled (EPC Clock = AXI Clock).

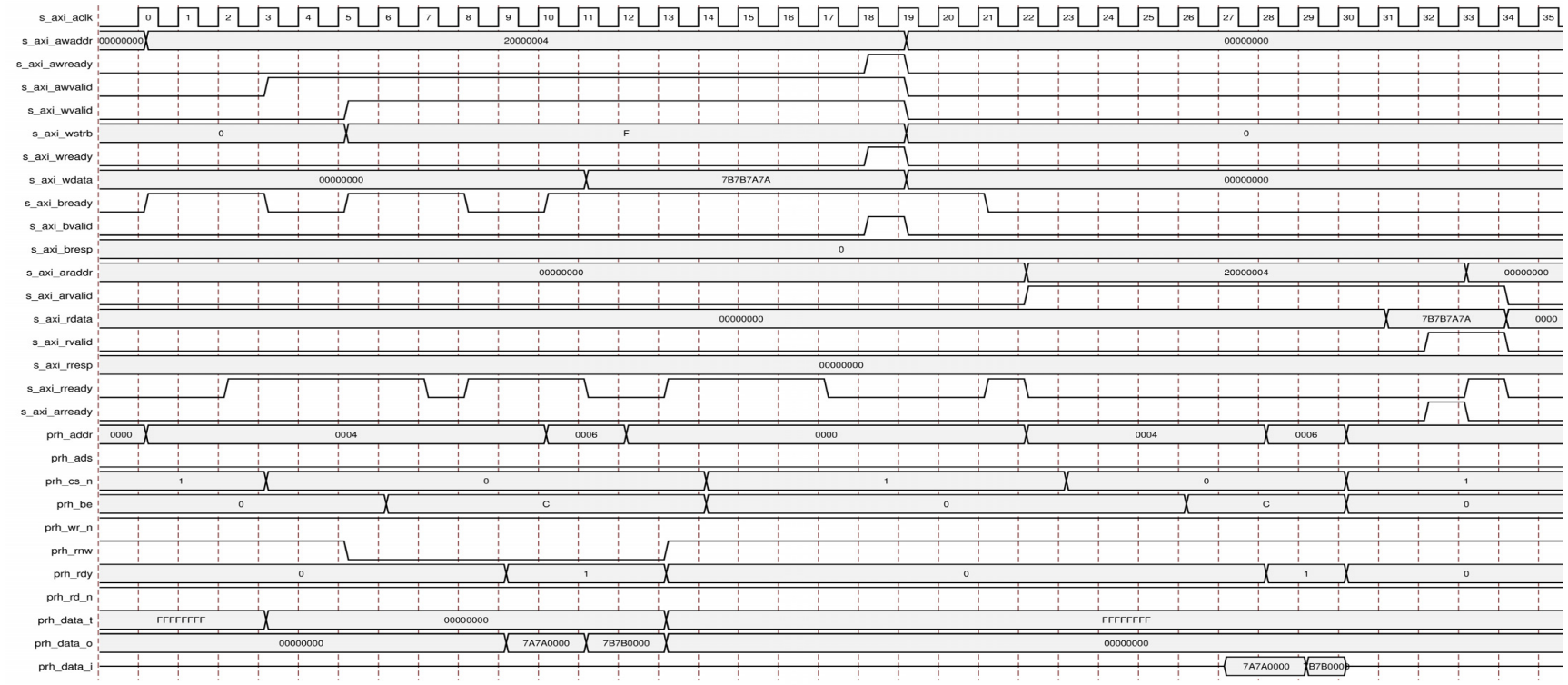

Figure 3-4: **Synchronous Read and Write Transactions to Device Memory**

Figure 3-5 shows the synchronous read and write transactions to device memory when the bus is multiplexed and data width matching is disabled (EPC Clock = AXI Clock).

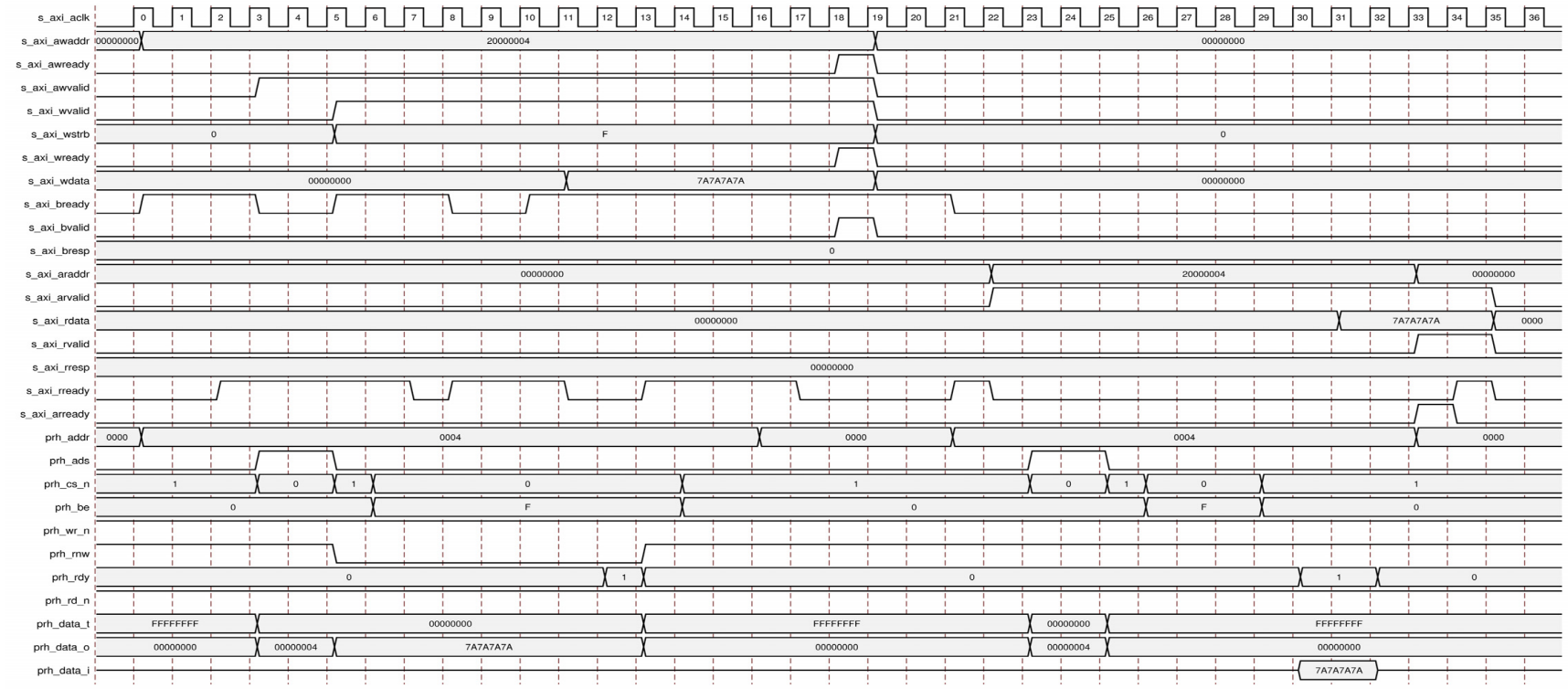

Figure 3-5: **Synchronous Read and Write Transactions to Device Memory**

Figure 3-6 shows the asynchronous write-read transactions to device memory when the bus is not multiplexed and data width matching is disabled (EPC Clock = AXI Clock).

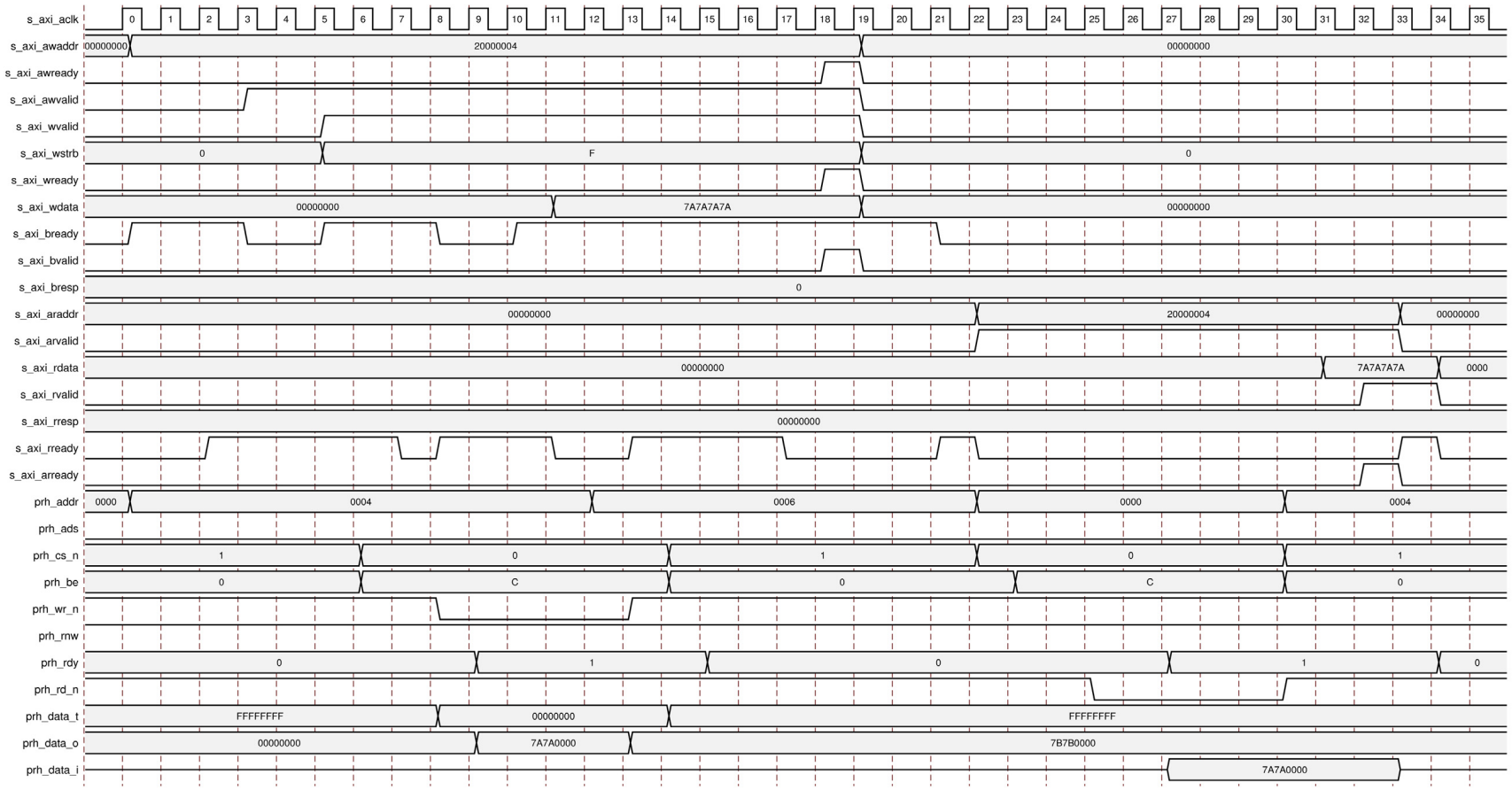

Figure 3-6: **Asynchronous Write-Read Transactions to Device Memory**

Figure 3-7 shows the asynchronous read and write transactions to device memory when the bus is multiplexed and data width matching is enabled (EPC Clock = AXI Clock).

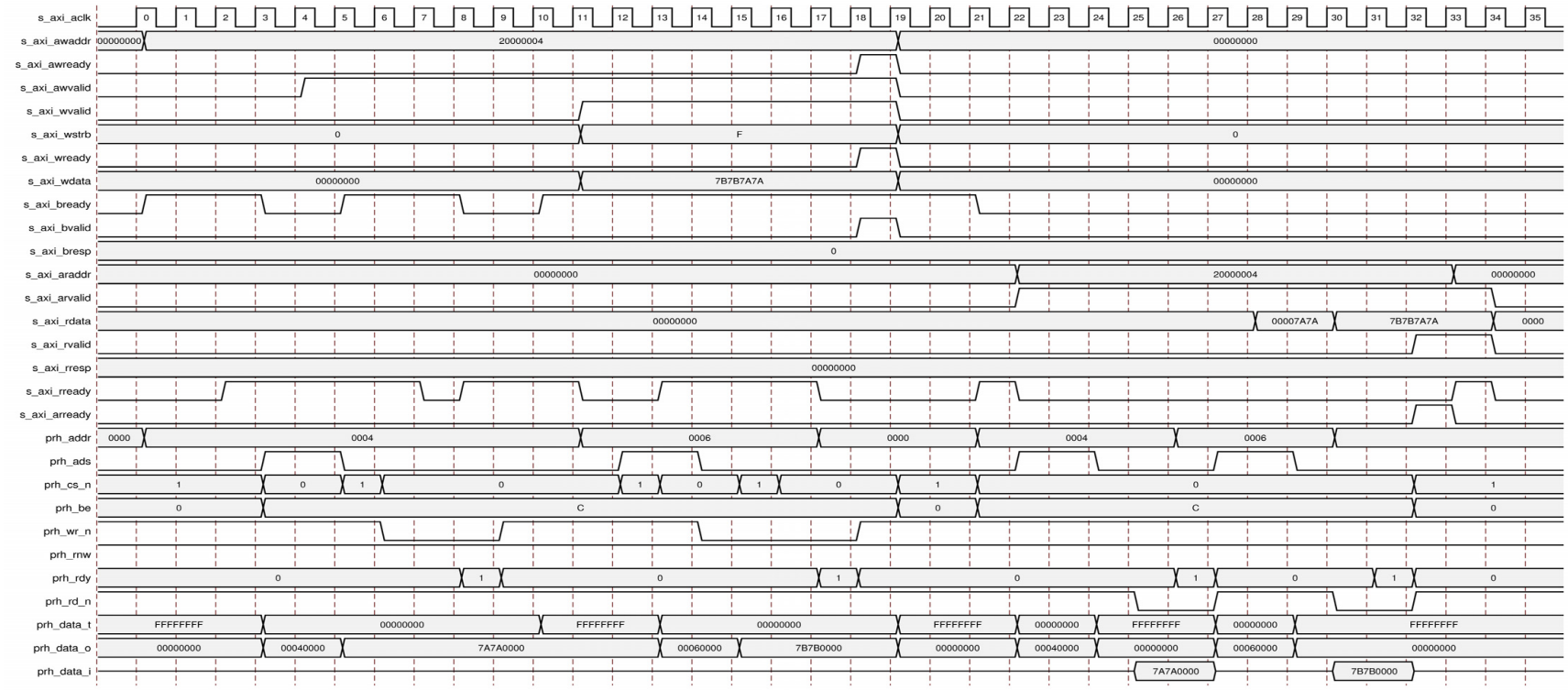

Figure 3-7: **Asynchronous Read and Write Transactions to Device Memory**

Figure 3-8 shows the asynchronous read and write transactions to device memory when the bus is not multiplexed and data width matching is enabled.

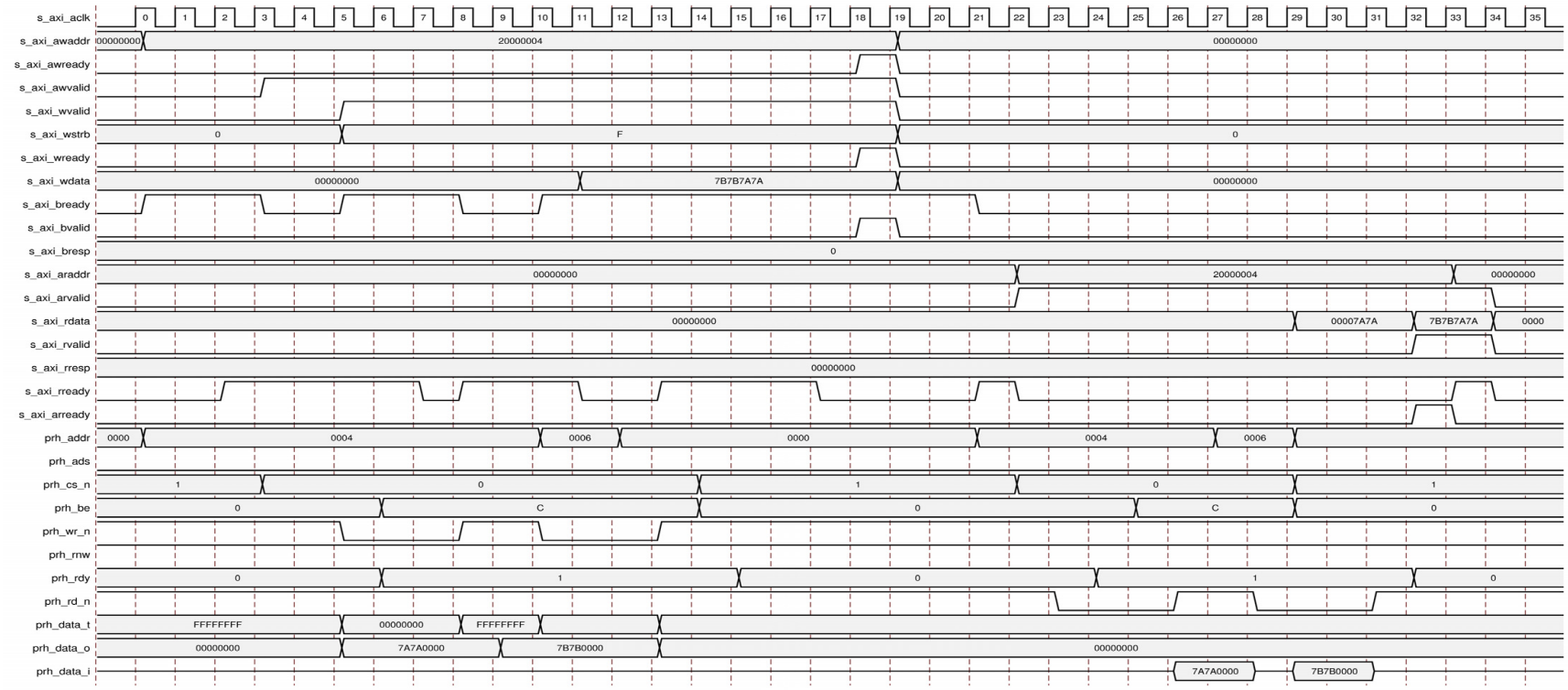

Figure 3-8: **Asynchronous Read and Write Transactions to Device Memory**

Figure 3-9 shows the asynchronous read and write transactions to device memory when the bus is multiplexed and data width matching is disabled.

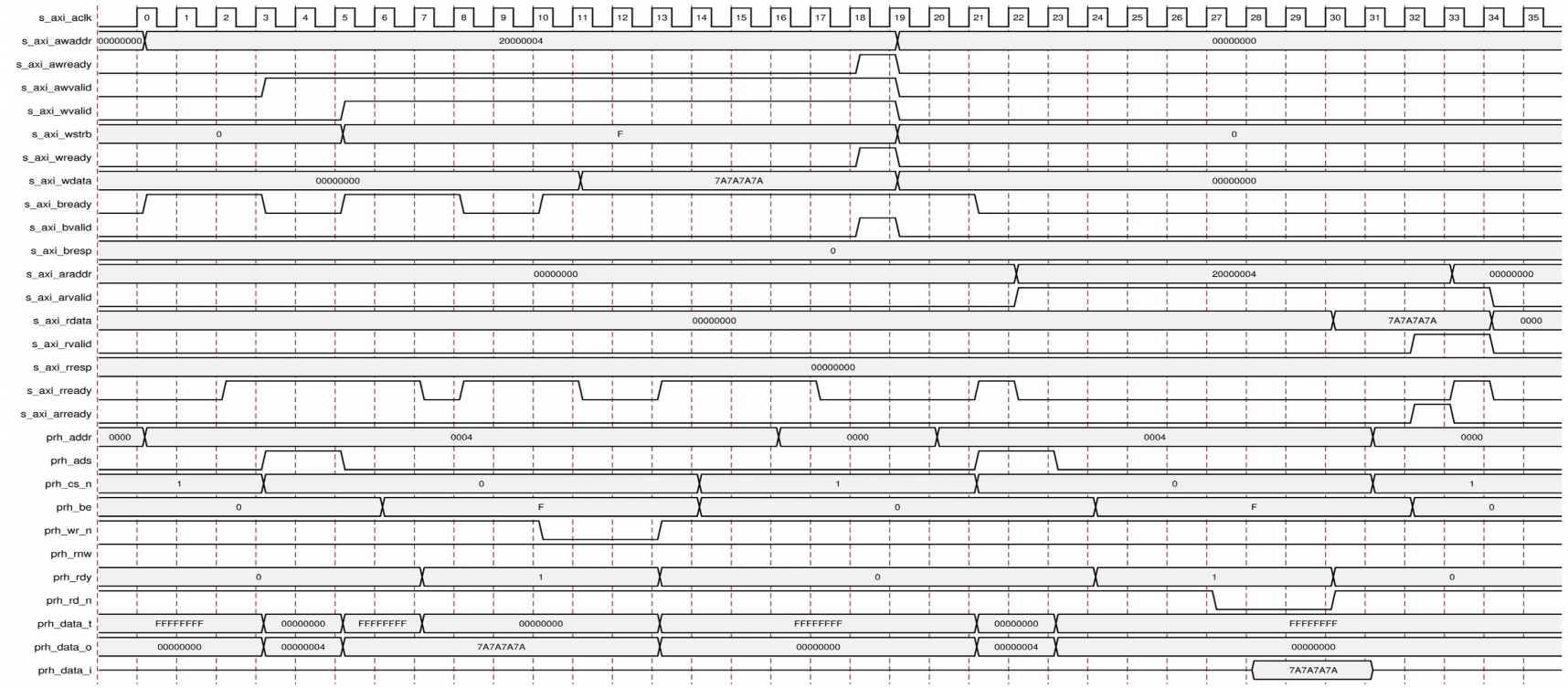

Figure 3-9: **Asynchronous Read and Write Transactions to Device Memory**

Scenario When the External Device is Not Ready for Transaction

In scenarios where the AXI EPC IP core has initiated the transaction, but the external peripheral is not ready (`prh_rdy` signal is not activated), the control signals from the core is extended until the internal Maximum Ready Assertion Period counter expires. The core completes the transaction with error signal generation.

Customizing and Generating the Core

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

Vivado Integrated Design Environment

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

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or popup menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5].

If you are customizing and generating the core in the Vivado IP integrator, see *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994) [Ref 6] for detailed information. Vivado Integrated Design Environment (IDE) might auto-compute certain configuration values when validating or generating the design, as noted in this section. You can view the parameter value after successful completion of `validate_bd_design` command.

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

Figure 4-1 shows the Customize IP window with the EPC tab settings for AXI EPC IP core.

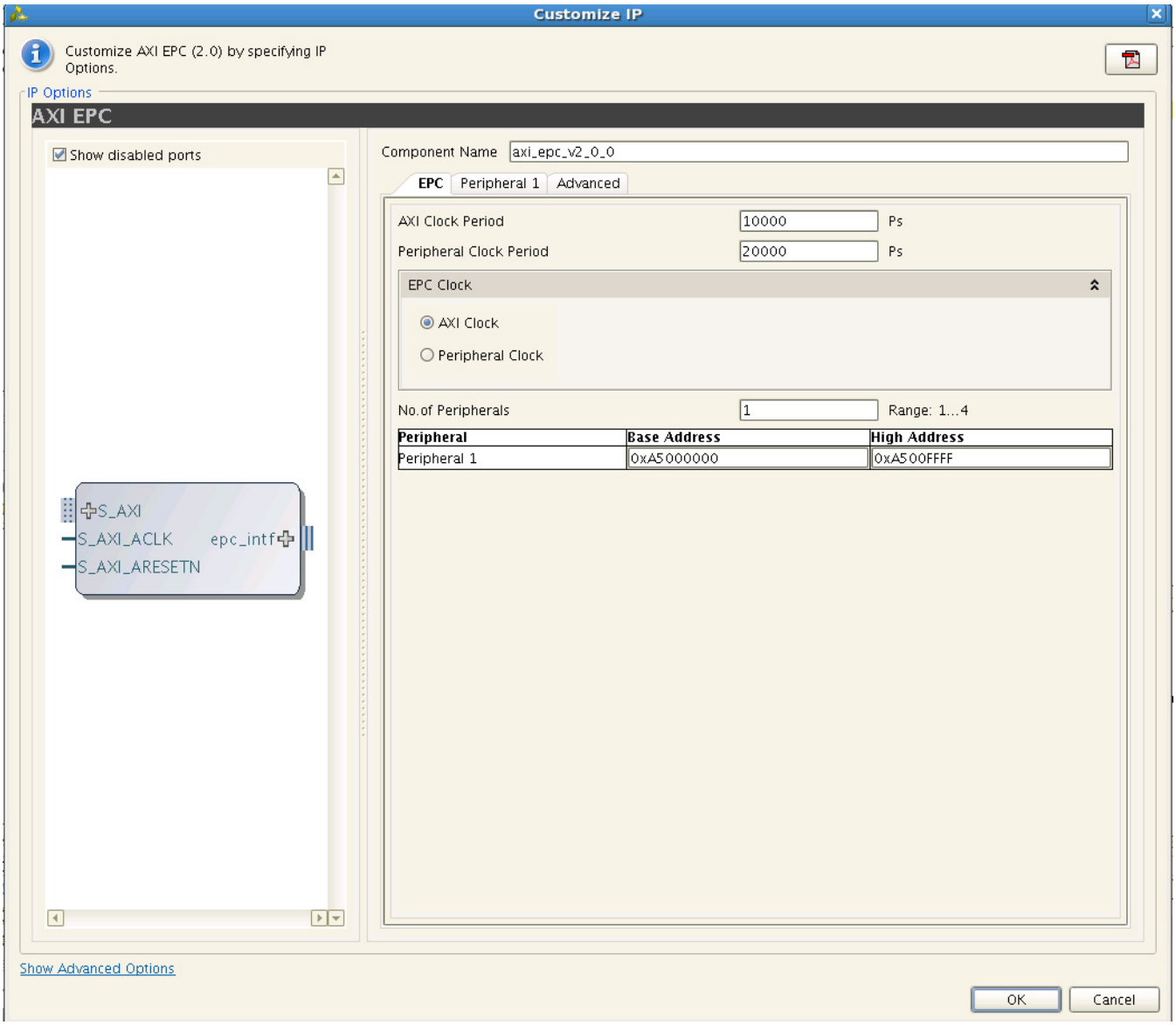

Figure 4-1: **Vivado Customize IP Dialog Box – EPC Tab**

Figure 4-2 shows the EPC tab settings with IP integrator.

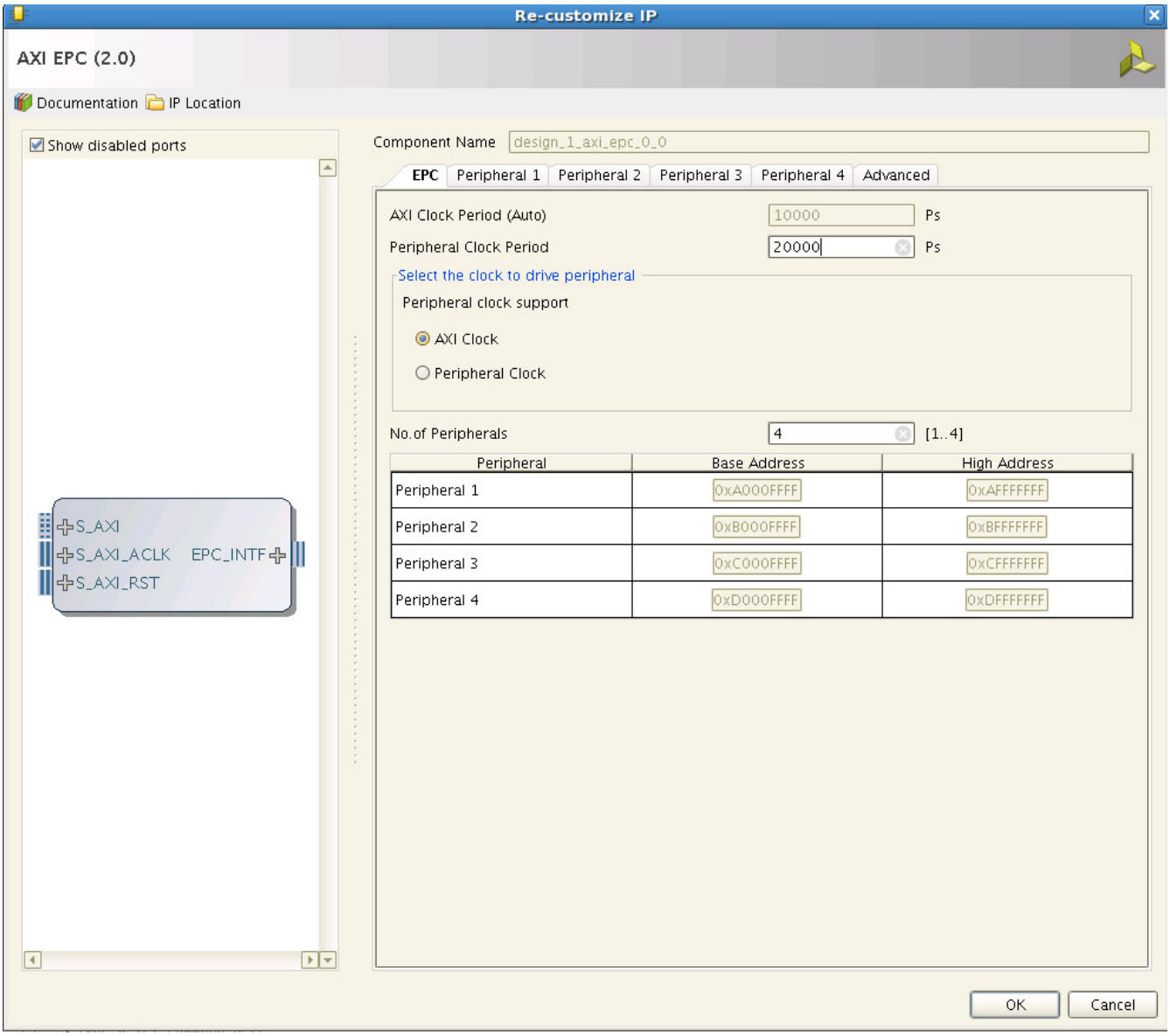

Figure 4-2: **Vivado Customize IP Dialog Box – EPC Tab with IP Integrator**

- **AXI Clock Period** – AXI clock period in picoseconds

  Note: When you are using IP integrator, this parameter is grayed-out and auto-updated.
- **Peripheral Clock Period** – Peripheral clock period in picoseconds
- **EPC Clock** – AXI clock and Peripheral clock options
- **Number of Peripherals** – Range of 1 to 4

  Note: When you are using IP integrator, the Base and High Addresses of all peripherals are grayed-out and auto-updated.

Figure 4-3 shows the AXI EPC IP core Peripheral 1 tab. A similar tab appears for each peripheral. As the number of peripheral parameter value increases, a new tab is added such as Peripheral 2, Peripheral 3, and the final Peripheral 4.

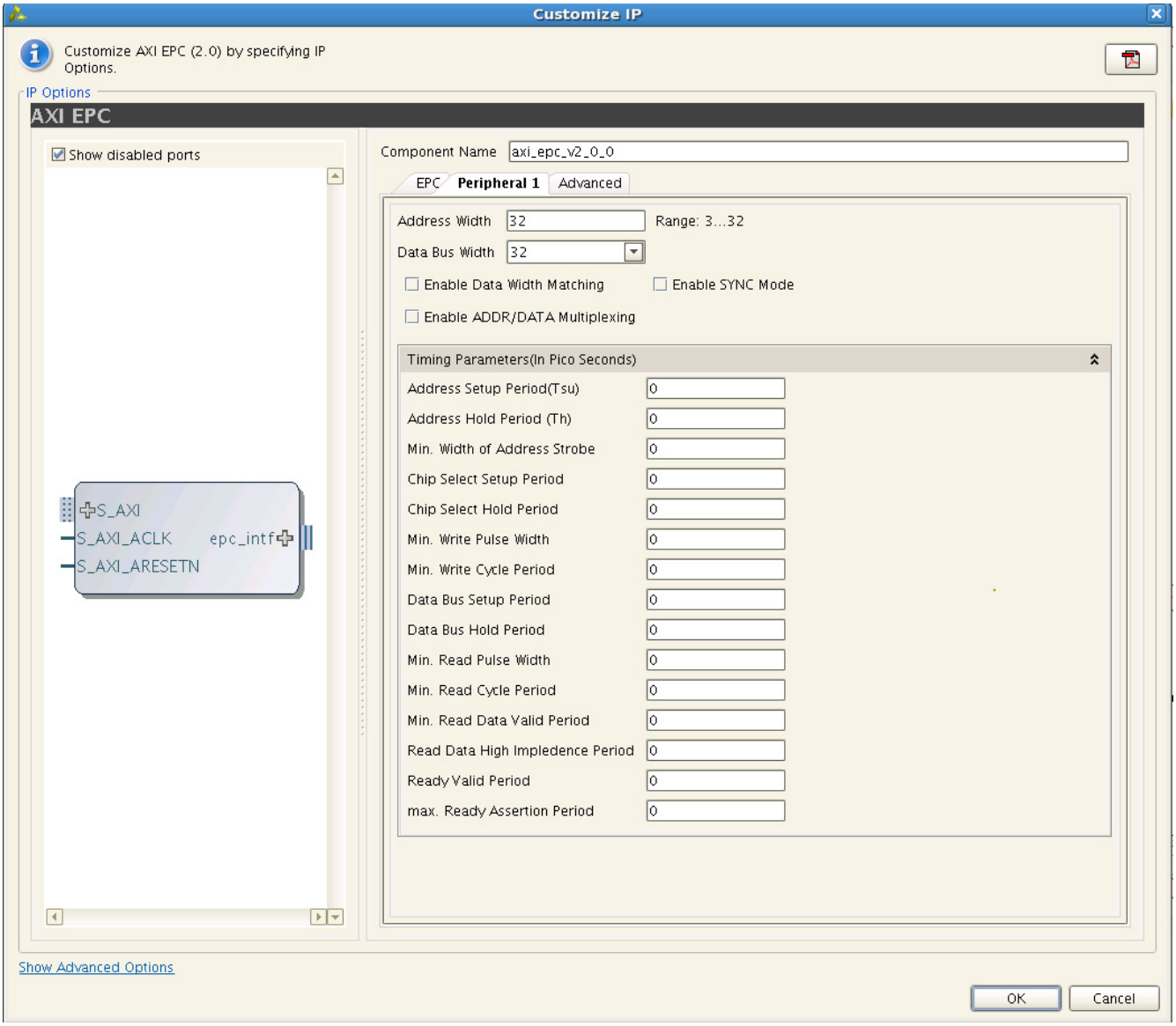

Figure 4-3: **Vivado Customize IP Dialog Box – Peripheral 1 Tab**

- **Address Width** – Maximum of address bus width of all external peripherals
- **Data Bus Width** – Maximum of data bus width of all external peripherals
- **Enable Data Width Matching** – Support for data width match when the peripheral device data width is less than the AXI data width
- **Enable SYNC Mode** – Peripheral access mode
- **Enable ADDR/DATA Multiplexing** – Peripheral bus type

- **Address Setup Period** – Address bus (prh_addr) setup with respect to rising edge of address strobe (prh_ads) or falling edge of read/write (prh_rd_n/prh_wr_n)
- **Address Hold Period** – Address bus (prh_addr) hold with respect to falling edge of address strobe (prh_ads) or rising edge of read/write (prh_rd_n/ prh_wr_n)
- **Minimum Width of Address Strobe** – Minimum pulse width of address strobe (PRH_ADS)
- **Chip Select Setup Period** – Chip select (prh_cs_n) setup with respect to falling edge of read/write (prh_rd_n/ prh_wr_n)
- **Chip Select Hold Period** – Chip select (prh_cs_n) hold with respect to rising edge of read/write (prh_rd_n/prh_wr_n)
- **Minimum Write Pulse Width** – Minimum pulse width of write signal (prh_wr_n)
- **Minimum Write Cycle Period** – Cycle time of write signal (prh_wr_n)
- **Data Bus Setup Period** – Data bus (prh_data) setup with respect to falling edge of write signal (prh_wr_n)
- **Data Bus Hold Period** – Data bus (prh_data) hold with respect to rising edge of write signal (prh_wr_n)
- **Minimum Read Pulse Width** – Minimum pulse width of read signal (prh_rd_n)
- **Minimum Read Cycle Period** – Cycle time of read signal (prh_rd_n)
- **Minimum Read Data Valid Period** – Data bus (prh_data) validity from falling edge of read signal (prh_rd_n)
- **Read Data High-Z Period** – Data bus (prh_data) high impedance from rising edge of read (prh_rd_n)
- **Read_valid_period** – Device ready (prh_rdy) validity from the falling edge of read or write (prh_rd_n/ prh_wr_n)
- **Maximum Ready Assertion Period** – Maximum period of device ready signal (prh_rdy) to wait before device timeout

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3].

Constraining the Core

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

Clock Frequencies

The AXI clock should be greater than the peripheral clock. Also, both clocks should be synchronous to each other.

Timing Constraints

Timing constraints are delivered along with IP generation.

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 7].

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado® Design Suite User Guide: Designing with IP* (UG896) [Ref 3].

Example Design

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

The top module instantiates all components of the core and example design that are needed to implement the design in hardware, as shown in Figure 8-1. This includes clock generator (MMCME2), register configuration, data generator, and data checker modules.

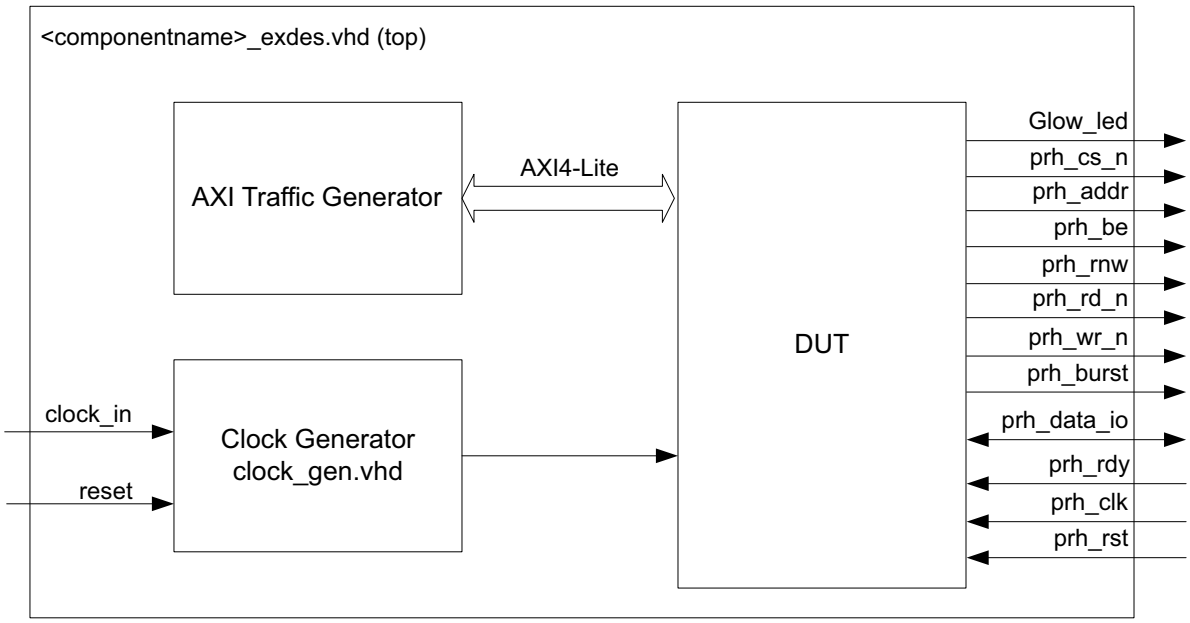

Figure 8-1: **AXI EPC Example Design Block Diagram**

- **Clock Generator** – MMCME2 generates the clocks for the example design. MMCME2 generates 100 MHz clock for `s_axi_aclk`. DUT and other modules of the example design are kept under reset until MMCME2 is locked.
- **AXI Traffic Generator (ATG)** – This module (IP) is configured in System Test Mode. All of the `AXI_EPC` related AXI4-Lite transactions are stored in the `coe/mif` file. For more information on AXI Traffic Generator, see *LogiCORE IP AXI Traffic Generator* (PG125) [Ref 9]. The ATG automatically starts the AXI4-Lite transaction after coming out of reset.

Implementing the Example Design

After following the steps described in Chapter 4, Customizing and Generating the Core to generate the core, implement the example design as follows:

1. Right-click the core in the Hierarchy window, and select **Open IP Example Design**.
2. A new window pops up, asking you to specify a directory for the example design. Select a new directory or keep the default directory.
3. A new project is automatically created in the selected directory and it is opened in a new Vivado window.
4. In the Flow Navigator (left-side pane), click **Run Implementation** and follow the directions.

ATG writes value x"AA to base address. This initiates a write transaction at external peripherals. The example design test bench checks for all the peripheral output values. When all external peripherals functions properly, "Test successfully completed" string is printed and `Glow_led` pin is 1. This ensures that the test has passed.

If `Glow_led` pin is High and the string "Test successfully completed" is not printed, This implies the test failed.

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 `<component_name>_example/<component_name>_example.srcs/` directory. This directory and its subdirectories contain all the source files that are required to create the AXI EPC controller example design.

Table 8-1 shows the files delivered in this `<component_name>_example/<component_name>_example.srcs/sources_1/imports/example_design/` directory.

Table 8-1: **Example Design Directory**

Name	Description
<component_name>_exdes.vhd	Top-level HDL file for the example design.
clock_gen.vhd	Clock generation module for example design.
atg_addr.coe	COE file of address. This file contains the EPC address.
atg_data.coe	COE file of data. This file contains the data to be written on the EPC data line.
atg_mask.coe	COE file to mask certain reads.
atg_ctrl.coe	COE file that contains control information of ATG.

Table 8-2 shows the files delivered in this `<component_name>_example/<component_name>_example.srcs/sources_1/sim_1/imports/simulation/` directory.

Table 8-2: **Simulation Directory**

Name	Description
<component_name>_exdes_tb.vhd	Test bench for the example design.

Table 8-3 shows the files delivered in this `<component_name>_example/<component_name>_example.srcs/sources_1/constrs_1/imports/example_design/` directory.

Table 8-3: **Constraints Directory**

Name	Description
<component_name>_exdes.xdc	Top-level constraints file for the example design.

Simulating the Example Design

Using the AXI EPC example design (delivered as part of the AXI EPC), you can quickly simulate and observe the behavior of the AXI EPC.

Setting Up the Simulation

The Xilinx simulation libraries must be mapped into the simulator. If the libraries are not set for your environment, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7] 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**.

Simulation Results

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

If the test passes, then the following message is displayed:

```
Test Completed Successfully
```

If the test fails or does not complete, then the following message is displayed:

```
Test Failed!! Test Timed Out.
```

Test Bench

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

Figure 9-1 shows the test bench for AXI EPC example design. The top-level test bench generates 200 MHz clock and drives initial reset to the example design.

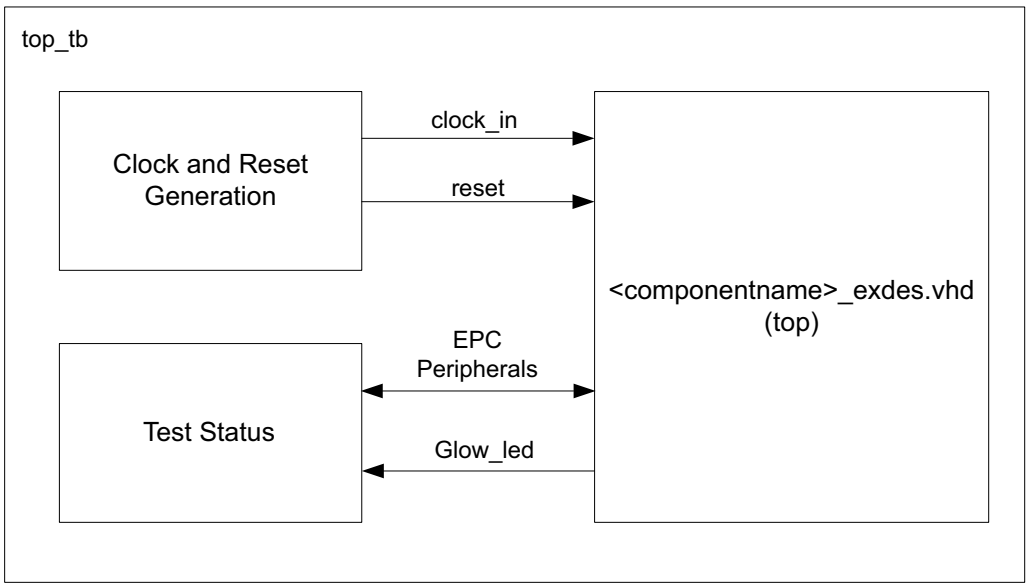

Figure 9-1: **AXI EPC Example Design Test Bench**

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 on migrating to the Vivado Design Suite, see *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 8].

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.

Debugging

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

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI External Peripheral Controller, 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 External Peripheral Controller. 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 also be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as:

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

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

Master Answer Record for the AXI External Peripheral Controller

AR: 54419

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, navigate to the Xilinx Support web page.

Note: Access to WebCase is not available in all cases. Login to the WebCase tool to see your specific support options.

Debug Tools

There are many tools available to address AXI External Peripheral Controller 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 represents the functionality in the Vivado IDE that is used for logic debugging and validation of a design running in Xilinx devices in hardware.

The Vivado lab tools 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 *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 10].

Additional Resources

Xilinx Resources

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

www.xilinx.com/support.

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

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

References

These documents provide supplemental material useful with this product guide:

1. LAN91C111, *10/100 Non-PCI Ethernet Single Chip MAC + PHY Data Sheet, SMSC*
2. CY7C67300, *EZ-Host™ Programmable Embedded USB Host/Peripheral Controller Data Sheet*, Cypress Semiconductor
3. *Vivado® Design Suite User Guide: Designing with IP* (UG896)
4. *AXI Reference Guide* (UG761)
5. *Vivado Design Suite User Guide: Getting Started* (UG910)
6. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994)
7. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
8. *ISE® to Vivado Design Suite Migration Guide* (UG911)
9. LogiCORE™ IP AXI Traffic Generator Product Guide (PG125)
10. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
11. *LogiCORE IP AXI4-Lite IPIF Product Guide* (PG155)
12. *LogiCORE IP AXI Interconnect Product Guide* (PG059)

13. *7 Series FPGAs Overview* (DS180)

14. *ARM® AMBA® AXI4 Protocol Version: 2.0 Specification*
http://www.arm.com/products/system-ip/amba/amba-open-specifications.php

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/05/2016	2.0	• Added a note in AXI4-Lite Interface Module section under Chapter 1, Overview • Added the Automotive Applications Disclaimer in Notice of Disclaimer
11/18/2015	2.0	Added support for UltraScale+ families.
12/18/2013	2.0	Added UltraScale support.
10/02/2013	2.0	• Revision number advanced to 2.0 to align with core version number 2.0. • Added IP Integrator. • Updated Performance section. • Updated Resource Utilization section. • Updated Clock Frequencies and Timing Constraints section. • Added Simulation, Synthesis, Example Design, and Test Bench chapters. • Updated Migrating Appendix.
03/20/2013	1.0	Initial Xilinx release of the product guide and replaces DS809. • Updated Table 2-1 Maximum Frequencies. • Updated Resource Utilization tables. • Updated Table 2-5 AXI EPC IP Core I/O Signal Description.

Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at http://www.xilinx.com/warranty.htm; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: http://www.xilinx.com/warranty.htm#critapps.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.