S2IMAC External Peripheral Controller (EPC) v1.02a

DS031 February 25, 2011

Product Extension Specification

Introduction

This specification defeines the extension to the original XPS EPC IP Core by Xilinx. The original controller supports data transfers between the Processor Local Bus (PLB V4.6) and the external synchronous and / or asynchronous peripheral devices such as USB and LAN devices.

As well as the original controller the S2IMAC EPC supports peripheral devices in the same manner and closes the leak of dowdy interrupt handling.

CORE Facts		
Core Specifics		
Supported Device Family	Virtex®-6, Spartan®-6, Virtex-5/5FX, Virtex-4/4QV/4Q, Automotive Spartan-3/3A/3A DSP/3E, Spartan-3E, Spartan-3, Spartan-3A, Spartan-3A DSP	
Version of Core	s2imac_epc	v1.02a
Resources Used		
	Min	Max
Slices	Refer to DS581 to Table 5, Table 6. Table 7 and TaDS581ble 8	
LUTs		
FFs		
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	N/A	
Verification	N/A	
Instantiation Template	N/A	
Design Tool Requirement		
Xilinx Implementation Tools	ISE® 11.3 or later	
Verification	–	
Simulation	–	
Synthesis	XST	
Support		
Provided by Xilinx, Inc. for the XPS EPC part. Provided by Li-Pro.Net for the S2IMAC EPC part.		

Features

S2IMAC Extension

- Single Interrupt Handling (simple pass-through)

XPS EPC Standard Features

- Connects as a 32-bit slave on PLB V4.6 buses of 32-bit, 64-bit or 128-bit
- PLB interface with 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 of peripheral devices with the support for a separate clock domain for synchronous peripheral devices
- Supports both multiplexed and non-multiplexed address and data buses
- The data width of peripheral devices is independently configured to 8-bit, 16-bit or 32-bit with the provision to enable data width matching when the PLB data width is greater than that of peripheral device
- Configurable timing parameters for peripheral bus interface

Functional Description

The S2IMAC External Peripheral Controller (S2IMAC EPC) is based on Xilinx's original IP Core XPS EPC. The interface diagram shown in Figure 1 depicts the overall interfaces of the core design. New in S2IMAC is the fast forward pass-through of a single interrupt line on top level.

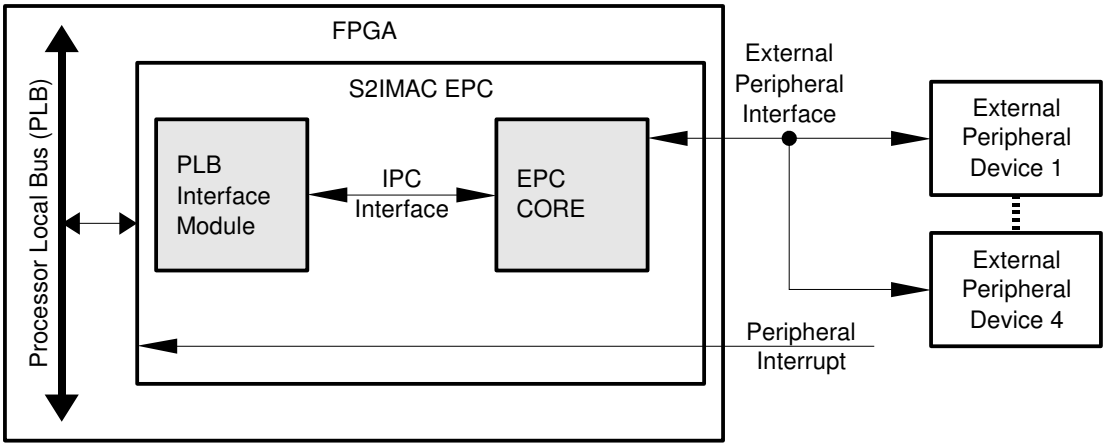

Figure 1: **S2IMAC EPC Interface Diagram**

As well as the XPS EPC the S2IMAC EPC IP Core design provides a general purpose interface to external peripheral devices and the PLB. The S2IMAC EPC IP Core can be configured to provide support for multiple external peripherals (non-memory peripherals like USB, LAN etc.) up to maximum of four devices. For more functional description read Xilinx original product specification DS581.

As a new extension the S2IMAC EPC provides a simple pass-through handling for a single peripheral interrupt line. There is no interrupt control functionality.

S2IMAC EPC IP Core Design Parameters

To allow user to create the S2IMAC EPC that is uniquely tailored for the user´s system, certain feature can be parameterized in the S2IMAC EPC design. All the design parameters presented by XPS EPC are represented by S2IMAC EPC. For more informations read Xilinx original product specification DS581.

The additional feautures that are parameterizable in the extended S2IMAC EPC core are as shown in Table 1.

Table 1: **S2IMAC EPC IP Core Extended Design Parameters**

Generic	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
S2IMAC EPC Extended Interface Paraemters					
G41	S2IMAC Interrupt	C_INTERRUPT_ PRESENT	0 = Interrupt pass-through ist not present 1 = Interrupt pass-through is present	0	integer

S2IMAC EPC IP Core I/O Signals

The extended S2IMAC EPC core I/O signals are listed and described in the Table 2.

Table 2: **S2IMAC EPC IP Core Extended Design Parameters**

Port	Signal Name	Interface	Signal Type	Initial State	Description
S2IMAC EPC Extended Signals					
P57	PRH_Int	S2IMAC	I	–	External peripheral interrupt input
P58	IP2INTC_Irpt	S2IMAC Interrupt	O	0	S2IMAC Interrupt Active high signal

Parameter – Port Dependencies

In addition, when certian features are parameterized away, the related logic is removed. The dependencies between the S2IMAC EPC extended design parameters and the I/O ports are shown in Table 3.

Table 3: **S2IMAC EPC IP Core Extended Parameter – Port Dependencies**

Generic or Port	Parameter	Affects	Depends	Relationship Description
G41	C_INTERRUPT_PRESENT	P57, P58	–	When C_INTERRUPT_PRESENT is 1, interrupt input is connected to interrupt output. Otherwise the interrupt output is driven with an inactive level and the input is unused.

Specification Exceptions

N/A

Support

Xilinx provides technical support for XPS EPC IP Core when used as described in the product documentation. For more informations read Xilinx original product specification DS581. Li-Pro.Net provides limited support for the S2IMAC EPC IP Core when used in commercial relationship between Li-Pro.Net and its customers.

Reference Documents

The following documents contain information that may be required in understanding the S2IMAC EPC IP Core reference designs:

1. DS581 XPS External Peripheral Controller (EPC), Product Specification version 2.0
2. IBM CoreConnect 128-Bit Processor Local Bus: Architecture Specifications version 4.6

Revision History

Date	Version	Revision
2/25/11	1.0	Initial Li-Pro.Net release.

Notice of Disclaimer

XPS External Peripheral Controller (EPC) v1.02a

DS581 July 20, 2009 Product Specification

Introduction

This specification defines the architecture and interface requirements for the External Peripheral Controller (XPS EPC IP Core). The controller supports data transfers between the Processor Local Bus (PLB V4.6) and the external synchronous and/or asynchronous peripheral devices such as USB and LAN devices.

Examples of peripheral devices supported by the XPS 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 PLB V4.6 buses of 32-bit, 64-bit or 128-bit
- PLB interface with 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 of peripheral devices with the support for a separate clock domain for synchronous peripheral devices
- Supports both multiplexed and non-multiplexed address and data buses
- The data width of peripheral devices is independently configured to 8-bit, 16-bit or 32-bit with the provision to enable data width matching when the PLB data width is greater than that of peripheral device
- Configurable timing parameters for peripheral bus interface
- Tested with the SMSC LAN91C111 and the Cypress CY7C67300 USB Controller device

LogiCORE™ Facts		
Core Specifics		
Supported Device Family	See EDK Supported Device Families.	
Version of Core	xps_epc	v1.02a
Resources Used		
	Min	Max
Slices	Refer to Table 5, Table 6, Table 7 and Table 8	
LUTs		
FFs		
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	N/A	
Verification	N/A	
Instantiation Template	N/A	
Design Tool Requirements		
Xilinx Implementation Tools	See Tools for requirements.	
Verification		
Simulation		
Synthesis		
Support		
Provided by Xilinx, Inc.		

Functional Description

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

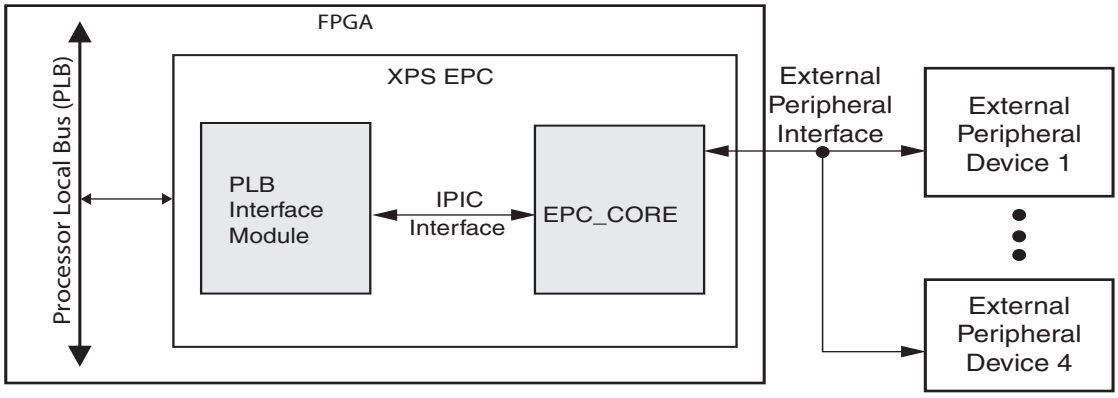

Figure 1: **XPS EPC Interface Diagram**

The XPS EPC IP Core design provides a general purpose interface to external peripheral devices and the PLB. It is a PLB slave device. The XPS EPC IP Core can be configured to provide support for multiple external peripherals (non-memory peripherals like USB, LAN etc.) up to a maximum of four devices and each device is independently configured to respond either in synchronous or in asynchronous mode. The timing parameters governing the access cycles such as setup/hold time, cycle access time, cycle recovery time, etc. are configured by the user. It receives read or write operation commands from the PLB and generates a corresponding access cycle to one of the four peripheral devices. It is recommended that peripherals like LAN, USB which have embedded interface should be used with the core.

The XPS EPC IP Core is comprised of the following modules:

- PLB Interface Module
- EPC CORE

PLB Interface Module

The PLB Interface Module provides an interface between the EPC CORE and the PLB. The PLB interface module implements the basic functionality of the PLB interface operation and does the necessary protocol and timing translation between the PLB and the IPIC interface.

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 XPS EPC is shown in Figure 2.

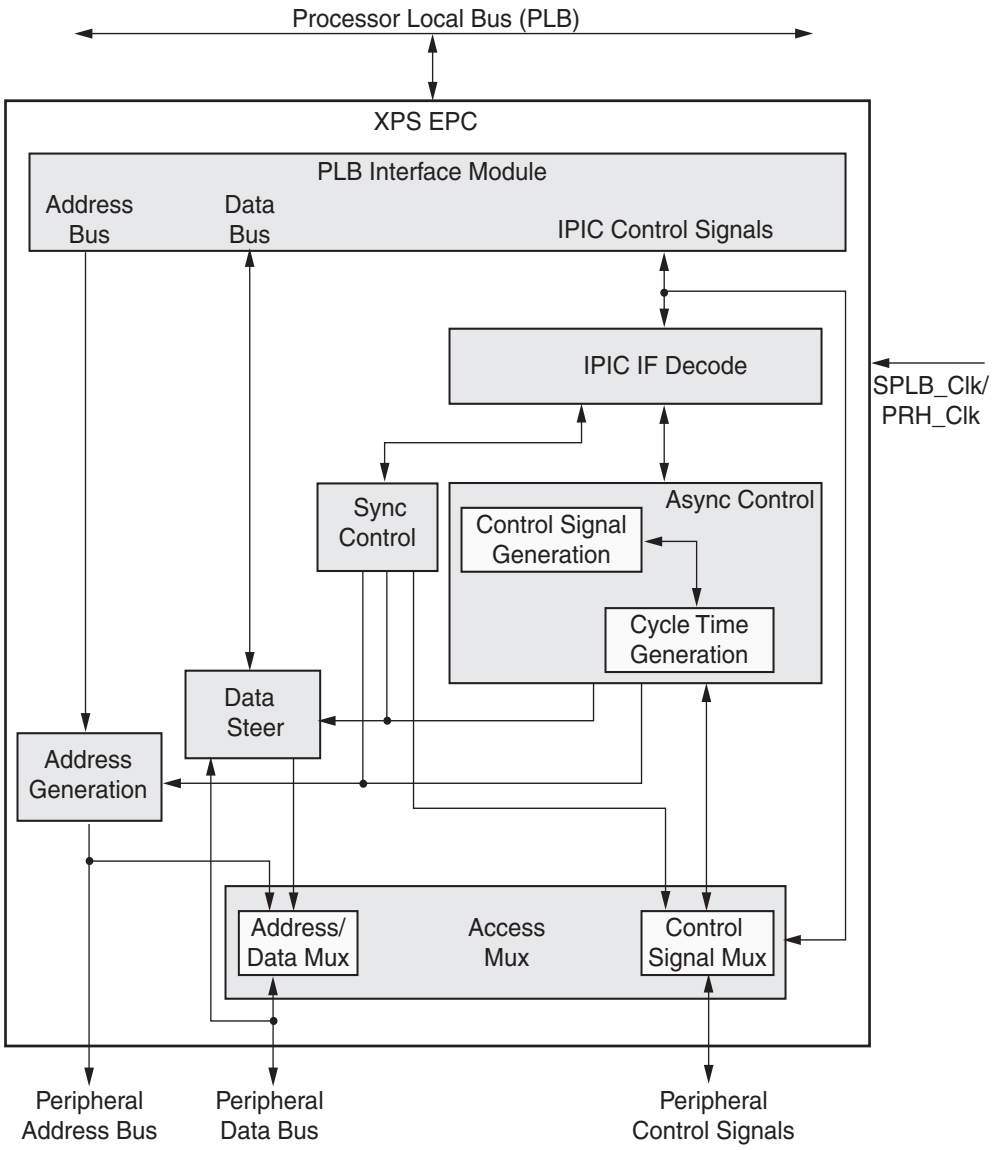

Figure 2: **XPS EPC Block Diagram**

The XPS EPC core consists of:

- The PLB Interface Module: This module provides the address decoding logic and necessary interface between PLB and XPS EPC core signals.
- The IPIC IF Decode module: This module provides decoding of IPIC signals of PLB Interface Module and synchronization of control signals.
- The Sync Control module: This module implements the state machine which controls the synchronous interface.
- The Async Control module: This module implements the state machine which controls the asynchronous interface including the asynchronous timing parameters.
- The Data Steer module: This module provides the data bus width matching and data steering logic.
- The Address Generation module: This module provides the generation of the lower address bits.

- The Access Mux module: This module provides the multiplexing of address, data and other control signals.

A detailed description of the above modules is provided below.

IPIC IF Decode Module

The IPIC IF Decode module implements the interface to the PLB. 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 user parameter settings.

Sync Control and Async Control Modules

In Figure 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 C_PRHx_SYNC. If synchronous and asynchronous external peripheral devices exist simultaneously then both the Sync Control and Async Control modules will be implemented.

The Sync Control Module operates either on the PLB clock (SPLB_Clk) or on the external peripheral clock (PRH_Clk) depending on the generic C_PRH_CLK_SUPPORT. If C_PRH_CLK_SUPPORT = 1, then the Sync Control module operates on external device peripheral clock (PRH_Clk) that is different from the PLB clock. 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 PLB clock in the IPIC IF Decode module. If C_PRH_CLK_SUPPORT = 0, the Sync Control module operates on the PLB clock and the IPIC Decode module interface does not perform synchronization of control signals.

The Async Control Module operates on the PLB clock only. This module generates the control signals to the initiate read and write access cycles to the external peripheral device based on the asynchronous timing parameters set by the user.

Data Steer and Address Generation Modules

The data bus of the external device must be less than or equal to the PLB data width and may be 8-bit, 16-bit or 32-bit. When the width of the external peripheral data bus is less than that of PLB and if C_PRHx_DWIDTH_MATCH = 1 for a particular device, then the Data Steer Module will generate multiple read or write cycles to the external device to match a single access on the PLB.

In order to map a single 32-bit PLB 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 8-bit wide, then four read or write cycles to the device will be performed in order to match a single 32-bit read or write transaction of PLB. For a write cycle, the first byte of the PLB data bus (PLB_wrDBus[0 : 7]) 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 PLB data bus (PLB_wrDBus[8 : 15]) is presented on the peripheral data bus (PRH_Data[0 : 7]) and so on. When the last byte of the PLB data bus (PLB_wrDBus[24 : 31]) is accepted by the peripheral, the data acknowledge signal is generated and send to the PLB 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 PLB data bus (PLB_wrDBus[0 : 7]) 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 PLB data bus (PLB_wrDBus[8 : 15]) and so on. When all four bytes are read from the external device an acknowledge is generated to the PLB Interface Module to indicate that the data is ready to be transferred to PLB 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 i.e 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 C_PRHx_BUS_MULTIPLEX. If C_PRHx_BUS_MULTIPLEX = 1, the address and the data bus are multiplexed and presented to the corresponding external device on PRH_Data bus. The address will be valid on the PRH_Data bus as long as the address strobe is active (PRH_ADS). This access will be performed in two phases (address phase and data phase). The data phase will be followed by the address phase. If C_PRHx_BUS_MULTIPLEX = 0, then the address and the data are presented to the device on separate buses (PRH_Addr bus and PRH_Data bus) and the access cycle will contain only one phase.

XPS EPC IP Core Design Parameters

To allow the user to create the XPS EPC that is uniquely tailored for the user's system, certain features can be parameterized in the XPS EPC design. Some of these parameters control the interface to the PLB while others control the interface to the peripheral devices. This allows the user to have a design that utilizes only the minimum resources required by the system and runs at the best possible performance.

The features that are parameterizable in the XPS EPC core are as shown in Table 1.

Table 1: **XPS EPC IP Core Design Parameters**

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
System Parameter					
G1	Target FPGA family	C_FAMILY	See C_FAMILY parameter values.		string
PLB Parameters					
G2	PLB address width	C_SPLB_AWIDTH	32	32	integer
G3	PLB data width	C_SPLB_DWIDTH	32, 64, 128	32	integer
G4	Selects point-to-point or shared PLB topology	C_SPLB_P2P	0 = Shared Bus Topology	0	integer
G5	PLB Master ID Bus Width	C_SPLB_MID_WIDTH	$\log_2$(C_SPLB_NUM_MASTERS) with a minimum value of 1	1	integer

Table 1: **XPS EPC IP Core Design Parameters *(Contd)***

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G6	Number of PLB Masters	C_SPLB_NUM_ MASTERS	1 - 16	1	integer
G7	Width of the Slave Data Bus	C_SPLB_NATIVE_ DWIDTH	32	32	integer
G8	PLB supports burst transactions	C_SPLB_SUPPORT_B URSTS	0	0	integer
XPS EPC Interface Parameters					
G9	PLB clock period	C_SPLB_CLK_ PERIOD_PS(1)	Integer number of picoseconds	10000	integer
G10	Peripheral clock period	C_PRH_CLK_ PERIOD_PS	Integer number of picoseconds	20000	integer
G11	Number of peripherals	C_NUM_ PERIPHERALS	1- 4	1	integer
G12	Maximum of address bus width of all external peripherals	C_PRH_MAX_ AWIDTH	3 - 32	32	integer
G13	Maximum of data bus width of all external peripherals	C_PRH_MAX_ DWIDTH	8, 16 , 32	32	integer
G14	Maximum of data bus width of all peripherals and address bus with of peripherals employing address/data multiplexing	C_PRH_MAX_ ADWIDTH(2)	8 - 32	32	integer
G15	Peripheral clock support	C_PRH_CLK_ SUPPORT(1)	0 = Peripheral device interface operates at the PLB clock 1 = Peripheral device interface operates at external peripheral clock	0	integer
G16	PLB burst support	C_PRH_BURST_ SUPPORT(3)	0 = No burst support from PLB	0	integer
XPS EPC Peripheral Address Space					
G17	External peripheral base address	C_PRHx_ BASEADDR(4)	Valid address(5,6)	User must set values(6)	std_logic _vector
G18	External peripheral high address	C_PRHx_ HIGHADDR(4)	Valid address(5,6)	User must set values(6)	std_logic _vector
XPS EPC Peripheral Interface Parameters					

Table 1: **XPS EPC IP Core Design Parameters *(Contd)***

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G19	Support for access to FIFO within the external peripheral	C_PRHx_FIFO_ ACCESS(4)	0 = No support for the external peripheral FIFO access 1 = Access to FIFO structures within the external peripheral device is supported	0	integer
G20	External peripheral FIFO offset from peripheral base address	C_PRHx_FIFO_ OFFSET(4,8,9)	Any valid offset within the base and high address range assigned to the peripheral device	0	integer
G21	Address bus width of peripherals	C_PRHx_AWIDTH(4)	3 - 32	32	integer
G22	Data bus width of peripherals	C_PRHx_DWIDTH(4)	8, 16 or 32	32	integer
G23	Support for data width match when the peripheral device data width is less than the PLB data width	C_PRHx_DWIDTH_ MATCH(4,10)	0 = No multiple cycles on the peripheral interface for single PLB read or write cycle 1 = Run multiple cycles on the peripheral interface for single PLB read or write cycle	0	integer
G24	Peripheral access mode	C_PRHx_SYNC(1,4)	0 = External device is asynchronous 1 = External device is synchronous	1	integer
G25	Peripheral bus type	C_PRHx_BUS_ MULTIPLEX(4)	0 = External device has separate address and data bus 1 = External device has multiplexed address and data bus	0	integer
XPS EPC Timing Parameters					
G26	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)	C_PRHx_ADDR_ TSU(4,11)	Integer number of picoseconds	User must set values.(12) Refer targeted device's data sheet.	integer

Table 1: **XPS EPC IP Core Design Parameters *(Contd)***

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G27	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)	C_PRHx_ADDR_ TH(4,13,17,20)	Integer number of picoseconds	User must set values.(12) Refer targeted device's data sheet.	integer
G28	Minimum pulse width of address strobe (PRH_ADS)	C_PRHx_ADS_ WIDTH(4)	Integer number of picoseconds	User must set values.(12) Refer targeted device's data sheet.	integer
G29	Chip select (PRH_CS_n) setup with respect to falling edge of read/write (PRH_Rd_n/ PRH_Wr_n)	C_PRHx_CSN_ TSU(4,11)	Integer number of picoseconds	User must set values.(14) Refer targeted device's data sheet.	integer
G30	Chip select (PRH_CS_n) hold with respect to rising edge of read/write (PRH_Rd_n/ PRH_Wr_n)	C_PRHx_CSN_ TH(4,17,20)	Integer number of picoseconds	User must set values.(14) Refer targeted device's data sheet.	integer
G31	Minimum pulse width of write signal (PRH_Wr_n)	C_PRHx_WRN_ WIDTH(4,15,16,17)	Integer number of picoseconds	User must set values.(14) Refer targeted device's data sheet.	integer
G32	Cycle time of write signal (PRH_Wr_n)	C_PRHx_WR_ CYCLE(4,16,17)	Integer number of picoseconds	User must set values.(14) Refer targeted device's data sheet.	integer

Table 1: **XPS EPC IP Core Design Parameters *(Contd)***

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G33	Data bus (PRH_Data) setup with respect to falling edge of write signal (PRH_Wr_n)	C_PRHx_DATA_TSU(4,15)	Integer number of picoseconds	User must set values.(14) Refer targeted device's data sheet.	integer
G34	Data bus (PRH_Data) hold with respect to rising edge of write signal (PRH_Wr_n)	C_PRHx_DATA_TH(4,17)	Integer number of picoseconds	User must set values.(14) Refer targeted device's data sheet.	integer
G35	Minimum pulse width of read signal (PRH_Rd_n)	C_PRHx_RDN_WIDTH(4,18,19,20)	Integer number of picoseconds	User must set values.(14) Refer targeted device's data sheet.	integer
G36	Cycle time of read signal (PRH_Rd_n)	C_PRHx_RD_CYCLE(4,19,20)	Integer number of picoseconds	User must set values.(14) Refer targeted device's data sheet.	integer
G37	Data bus (PRH_Data) validity from falling edge of read signal (PRH_Rd_n)	C_PRHx_DATA_TOUT(4,18)	Integer number of picoseconds	User must set values.(14) Refer targeted device's data sheet.	integer
G38	Data bus (PRH_Data) high impedance from rising edge of read (PRH_Rd_n)	C_PRHx_DATA_TINV(4,20)	Integer number of picoseconds	User must set values.(14) Refer targeted device's data sheet.	integer

Table 1: **XPS EPC IP Core Design Parameters *(Contd)***

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G39	Device ready (PRH_Rdy) validity from the falling edge of read or write (PRH_Rd_n/ PRH_Wr_n)	C_PRHx_RDY_ TOUT(4,21,23)	Integer number of picoseconds	User must set values.(14) Refer targeted device's data sheet.	integer
G40	Maximum period of device ready signal (PRH_Rdy) to wait before device timeout	C_PRHx_RDY_ WIDTH(4,22,23)	Integer number of picoseconds	User must set values.(23) Refer targeted device's data sheet.	integer

Table 1: **XPS EPC IP Core Design Parameters *(Contd)***

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type

Notes:

1. The generic C_PRH_CLK_SUPPORT is relevant only when the device is configured for synchronous access i.e. C_PRHx_SYNC = 1. If more than one device is synchronous, then the frequency for the peripheral clock (PRH_Clk) should be chosen as the minimum of the operating frequencies of those devices.
2. The C_PRH_MAX_ADWIDTH determines the size of the data bus. For all non multiplexed devices the C_PRH_MAX_ADWIDTH reflects the maximum of data bus width of all external devices i.e. C_PRH_MAX_ADWIDTH equals C_PRH_MAX_DWIDTH. However, if any of the devices is configured for multiplexed address and data bus, then C_PRH_MAX_ADWIDTH should be set as the maximum of data bus of all external devices and the address bus of device(s) employing multiplexed address and data bus.
3. Current version of the XPS EPC do not support the burst transactions to/from the PLB.
4. 'x' in the generic refers to the number of the peripheral device and takes a value in the range of 0 to C_NUM_PERIPHERALS - 1.
5. XPS EPC design can accommodate up to four peripheral devices. The address range for the devices are designated as C_PRHx_BASEADDR, C_PRHx_HIGHADDR etc.
6. The range specified by C_PRHx_BASEADDR and C_PRHx_HIGHADDR must comprise a contiguous range and the size of the range must be a power of two i.e. size of range = 2^m. Further, the 'm' least significant bits of C_PRHx_BASEADDR must be zero. The base and high address range assigned to different peripherals must be mutually exclusive.
7. No default value will be specified to ensure that the actual value is set i.e. if the value is not set, a compiler error will be generated.
8. C_PRHx_FIFO_ACCESS must be set to 1 if the support for access to FIFO within external peripheral device is to be included.
9. C_PRHx_FIFO_OFFSET is the byte offset of the external peripheral FIFO from the base address (C_PRHx_BASEADDR) of the peripheral irrespective of the data width of the peripheral device. If C_PRHx_FIFO_ACCESS = 1, then C_PRHx_FIFO_OFFSET must be set to a valid offset within the address range assigned to the peripheral.
10. The generic C_PRHx_DWIDTH_MATCH is relevant only when the width of the peripheral data (PRH_Data) bus is less than the PLB data bus (PLB_wrDBus). The generic C_PRHx_DWIDTH_MATCH must be set to '1' in such cases.
11. Address setup time is with respect to falling edge of address strobe (PRH_ADS) if the address and the data bus are multiplexed i.e. C_PRHx_BUS_MULTIPLEX = 1 and must be set both in synchronous and asynchronous mode. Address setup time is with respect to falling edge of read/write signals (PRH_Wr_n/PRH_Rd_n), if the device is non multiplexed and is relevant only if the access mode is asynchronous.
12. Value for the parameter must be assigned if the address and the data bus are multiplexed i.e. C_PRHx_BUS_MULTIPLEX = 1. This parameter assignment is applicable for synchronous and asynchronous devices.
13. Address hold time is with respect to falling edge of address strobe (PRH_ADS) if the address and the data bus are multiplexed i.e. C_PRHx_BUS_MULTIPLEX = 1 and must be set both in synchronous and asynchronous mode. Address hold time is with respect to rising edge of read/write signals (PRH_Wr_n/PRH_Rd_n) if the address and the data bus are separate and is relevant only if the access mode is asynchronous.
14. Value must be assigned if the access mode of the peripherals is asynchronous i.e. C_PRHx_SYNC = 0. If the access mode of the peripheral is synchronous i.e. C_PRHx_SYNC = 1, then the zero should be assigned to the parameter.
15. Write signal (PRH_Wr_n) low time is the maximum of C_PRHx_WRN_WIDTH and C_PRHx_DATA_TSU.
16. The value of C_PRHx_WRN_WIDTH must be smaller than C_PRHx_WR_CYCLE. The C_PRHx_WR_CYCLE time will be considered for the buffer period between consecutive writes.
17. In non-multiplexed address and data bus mode, write recovery time will be maximum of C_PRHx_ADDR_TH, C_PRHx_CSN_TH, C_PRHx_DATA_TH and PRH_Wr_n high time (i.e. C_PRHx_WR_CYCLE minus C_PRHx_WRN_WIDTH). If the peripheral uses multiplexed address and data bus, then the write recovery time will be maximum of C_PRHx_CSN_TH, C_PRHx_DATA_TH and PRH_Wr_n high time.
18. Read signal (PRH_Rd_n) low time is the maximum of C_PRHx_RDN_WIDTH and C_PRHx_DATA_TOUT.
19. The value of C_PRHx_RDN_WIDTH must be smaller than C_PRHx_RD_CYCLE.The C_PRHx_RD_CYCLE time will be considered for the buffer period between consecutive reads.
20. In non-multiplexed address and data bus mode, read recovery time will be maximum of C_PRHx_ADDR_TH, C_PRHx_CSN_TH, C_PRHx_DATA_TINV and PRH_Rd_n high time (i.e. C_PRHx_RD_CYCLE minus C_PRHx_RDN_WIDTH). If the peripheral uses multiplexed address and data bus, then the read recovery time will be maximum of C_PRHx_CSN_TH, C_PRHx_DATA_TINV and PRH_Rd_n high time.
21. Device ready validity period (C_PRHx_RDY_TOUT) must be set as the maximum of the device ready values specified for read and write transactions for that device.
22. Device ready pulse width (C_PRHx_RDY_WIDTH) must be set as the maximum of the device ready values specified for read and write transactions for that device.
23. Device ready validity (C_PRHx_RDY_TOUT) period must be less than device ready signal width (C_PRHx_RDY_WIDTH). Device ready signal width (C_PRHx_RDY_WIDTH) must be set for both synchronous and asynchronous access mode in order to prevent the device from holding the PLB indefinitely.

XPS EPC IP Core I/O Signals

The XPS EPC IP Core I/O signals are listed and described in the Table 2.

Table 2: **XPS EPC IP Core I/O Signal Description**

Port	Signal Name	Interface	Signal Type	Initial State	Description
System Signals					
P1	SPLB_Clk	PLB	I	-	PLB clock
P2	SPLB_Rst	PLB	I	-	PLB reset. Active high
PLB Slave Interface Input Signals					
P3	PLB_ABus[0 : 31]	PLB	I	-	PLB address bus
P4	PLB_PAValid	PLB	I	-	PLB primary address valid
P5	PLB_masterID[0 : C_SPLB_MID_WIDTH - 1]	PLB	I	-	PLB current master identifier
P6	PLB_RNW	PLB	I	-	PLB read not write
P7	PLB_BE[0 : (C_SPLB_DWIDTH/8) - 1]	PLB	I	-	PLB byte enables
P8	PLB_size[0 : 3]	PLB	I	-	PLB size of requested transfer
P9	PLB_type(0 : 2]	PLB	I	-	PLB transfer type
P10	PLB_wrDBus[0 : C_SPLB_DWIDTH - 1]	PLB	I	-	PLB write data bus
Unused PLB Slave Interface Input Signals					
P11	PLB_UABus[0 : 31]	PLB	I	-	PLB upper address bits
P12	PLB_SAValid	PLB	I	-	PLB secondary address valid
P13	PLB_rdPrim	PLB	I	-	PLB secondary to primary read request indicator
P14	PLB_wrPrim	PLB	I	-	PLB secondary to primary write request indicator
P15	PLB_abort	PLB	I	-	PLB abort bus request
P16	PLB_busLock	PLB	I	-	PLB bus lock
P17	PLB_MSize[0 : 1]	PLB	I	-	PLB data bus width indicator
P18	PLB_lockErr	PLB	I	-	PLB lock error
P19	PLB_wrBurst	PLB	I	-	PLB burst write transfer
P20	PLB_rdBurst	PLB	I	-	PLB burst read transfer
P21	PLB_wrPendReq	PLB	I	-	PLB pending bus write request
P22	PLB_rdPendReq	PLB	I	-	PLB pending bus read request
P23	PLB_wrPendPri[0 : 1]	PLB	I	-	PLB pending write request priority

Table 2: **XPS EPC IP Core I/O Signal Description** ***(Contd)***

Port	Signal Name	Interface	Signal Type	Initial State	Description
P24	PLB_rdPendPri[0 : 1]	PLB	I	-	PLB pending read request priority
P25	PLB_reqPri[0 : 1]	PLB	I	-	PLB current request priority
P26	PLB_TAttribute[0 : 15]	PLB	I	-	PLB transfer attribute
PLB Slave Interface Output Signals					
P27	Sl_addrAck	PLB	O	0	Slave address acknowledge
P28	Sl_SSize[0 : 1]	PLB	O	0	Slave data bus size
P29	Sl_wait	PLB	O	0	Slave wait
P30	Sl_rearbitrate	PLB	O	0	Slave bus rearbitrate
P31	Sl_wrDAck	PLB	O	0	Slave write data acknowledge
P32	Sl_wrComp	PLB	O	0	Slave write transfer complete
P33	Sl_rdDBus[0 : C_SPLB_DWIDTH - 1]	PLB	O	0	Slave read data bus
P34	Sl_rdDAck	PLB	O	0	Slave read data acknowledge
P35	Sl_rdComp	PLB	O	0	Slave read transfer complete
P36	Sl_MBusy[0 : C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Slave busy
P37	Sl_MWrErr[0 : C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Slave write error
P38	Sl_MRdErr[0 : C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Slave read error
Unused PLB Slave Interface Output Signals					
P39	Sl_wrBTerm	PLB	O	0	Slave terminate write burst transfer
P40	Sl_rdWdAddr[0 : 3]	PLB	O	0	Slave read word address
P41	Sl_rdBTerm	PLB	O	0	Slave terminate read burst transfer
P42	Sl_MIRQ[0 : C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Master interrupt request
XPS EPC Signals					
P43	PRH_Clk(1)	EPC	I	-	External peripheral clock input
P44	PRH_Rst	EPC	I	-	External peripheral reset input
P45	PRH_CS_n[0 : C_NUM_PERIPHERALS - 1]	EPC	O	1	External peripheral chip select. Active low signal

Table 2: **XPS EPC IP Core I/O Signal Description** ***(Contd)***

Port	Signal Name	Interface	Signal Type	Initial State	Description
P46	PRH_Addr[0 : C_PRH_MAX_AWIDTH - 1][2]	EPC	O	-	External peripheral address bus
P47	PRH_ADS	EPC	O	0	External peripheral address strobe in case of multiplexed address and data bus. Active high signal
P48	PRH_BE[0 : C_PRH_MAX_DWIDTH/8 - 1]	EPC	O	-	External peripheral byte enables
P49	PRH_RNW	EPC	O	1	External peripheral read/write signal for synchronous access mode
P50	PRH_Rd_n	EPC	O	1	External peripheral read signal for asynchronous access mode. Active low signal
P51	PRH_Wr_n	EPC	O	1	External peripheral write signal for asynchronous access mode. Active low signal
P52	PRH_Burst	EPC	O	0	Burst cycle indication to external peripheral. Active high signal
P53	PRH_Rdy[0 : C_NUM_PERIPHERALS - 1]	EPC	I	-	Peripheral ready signal. This signal is used by the external peripheral to extend the transaction. Active high signal
P54	PRH_Data_I[0 : C_PRH_MAX_ADWIDTH - 1]	EPC	I	-	External peripheral input data bus
P55	PRH_Data_O[0 : C_PRH_MAX_ADWIDTH - 1]	EPC	O	-	External peripheral output data bus
P56	PRH_Data_T[0 : C_PRH_MAX_ADWIDTH - 1]	EPC	O	-	3-state control for external peripheral output data bus

Notes:
1. PRH_Clk is utilized only when C_PRH_CLK_SUPPORT = 1 and the interface to the external peripheral is synchronous.
2. External peripheral devices are considered as byte addressable irrespective of the data bus width.

Parameter - Port Dependencies

The dependencies between the XPS EPC design parameters and the I/O ports are shown in Table 3. The width of the XPS EPC signals depend on some of the parameters. In addition, when certain features are parameterized out of the design, the related logic will no longer be a part of the design. The unused input signals and unrelated output signals are set to a specified value.

Table 3: **XPS EPC IP Core Parameter - Port Dependencies**

Generic or Port	Name	Affects	Depends	Description
Design Parameters				
G3	C_SPLB_DWIDTH	P7, P10, P33	-	The PLB data width parameter affects the number of byte enables configured for the PLB data bus, width of the PLB data bus and the width of the PLB slave read data bus
G5	C_SPLB_MID_WIDTH	P5	G6	Affects the number of bits required for the PLB_masterID input bus for slave devices. This value is equal to $\log_2$(C_SPLB_NUM_MASTERS)
G6	C_SPLB_NUM_MASTERS	P36, P37, P38, P42	-	Affects the number of PLB masters
G11	C_NUM_PERIPHERALS	P45, P53	-	The number of peripherals in the system determines the number of chip select signals driven and the number of device ready inputs decoded by the PLB external peripheral controller
G12	C_PRH_MAX_AWIDTH	P46	-	The width of the peripheral address bus is set to the maximum of address bus width of all peripheral devices
G13	C_PRH_MAX_DWIDTH	P48	-	The number of byte enables for the peripheral data bus is determined by the maximum of data bus width of the peripheral devices
G14	C_PRH_MAX_ADWIDTH	P54, P55, P56	-	The width of the peripheral input data bus, peripheral output data bus and the peripheral output data bus 3-state control are determined by the maximum of data bus width of all peripherals and the address bus width of peripherals employing address / data bus multiplexing
G15	C_PRH_CLK_SUPPORT	P43	-	If C_PRH_CLK_SUPPORT = 0 then all external devices operate on SPLB clock. The input PRH_Clk must be driven high externally
G23	C_PRHx_DWIDTH_MATCH	P52	-	If the device is configured for synchronous mode with data width matching enabled then PRH_Burst will be driven high until all but the last byte of data is flushed to the device. For asynchronous mode and synchronous mode with data width match disabled, PRH_Burst is driven low.

Table 3: **XPS EPC IP Core Parameter - Port Dependencies *(Contd)***

Generic or Port	Name	Affects	Depends	Description
G24	C_PRHx_SYNC	P43, P49, P50, P51, P52	-	If the device is configured for asynchronous mode: • With peripheral clock support, the peripheral clock input (PRH_Clk) must be driven high externally • PRH_RNW is driven high as PRH_Rd_n and PRH_Wr_n should be used as read and write strobe • With data width matching enabled, the XPS EPC drives PRH_Burst to its default low If the device is configured for synchronous mode: • The XPS EPC drives PRH_Rd_n and PRH_Wr_n outputs to their default state of high since PRH_RNW output is used as read/write strobe
G25	C_PRHx_BUS_ MULTIPLEX	P47	-	PRH_ADS is driven low if the device does not use multiplexed address and data bus i.e. C_PRHx_BUS_MULTIPLEX = 0
I/O Signals				
P5	PLB_masterID	-	G5	The PLB master ID is determined by the C_SPLB_MID_WIDTH parameter
P7	PLB_BE	-	G3	The number of byte enables for the PLB data bus is determined by the C_SPLB_DWIDTH parameter
P10	PLB_wrDBus	-	G3	The PLB data bus width is determined by the C_SPLB_DWIDTH parameter
P33	Sl_rdDBus	-	G3	The width of the PLB slave read data bus is determined by the C_SPLB_DWIDTH parameter
P36	Sl_MBusy	-	G6	The width of PLB slave busy is determined by the C_SPLB_NUM_MASTERS parameter
P37	Sl_MWrErr	-	G6	The width of PLB slave write error is determined by the C_SPLB_NUM_MASTERS parameter
P38	Sl_MRdErr	-	G6	The width of PLB slave read error is determined by the C_SPLB_NUM_MASTERS parameter
P42	Sl_MIRQ	-	G6	The width of PLB slave master interrupt request is determined by the C_SPLB_NUM_MASTERS parameter
P43	PRH_Clk	-	G15, G24	PRH_Clk is selected as operating clock only if the device is configured for synchronous mode with peripheral clock support i.e. C_PRHx_SYNC = 1 and C_PRH_CLK_SUPPORT = 1. Asynchronous interface always operates on SPLB_Clk. Therefore, if no device is configured for synchronous mode with peripheral clock support enabled then the input PRH_Clk must be driven high

Table 3: **XPS EPC IP Core Parameter - Port Dependencies *(Contd)***

Generic or Port	Name	Affects	Depends	Description
P45	PRH_CS_n	-	G11	The number of chip select signals driven by the PLB external peripheral controller is determined by the C_NUM_PERIPHERALS parameter
P46	PRH_Addr	-	G12	The width of the peripheral address bus is determined by the C_PRH_MAX_AWIDTH parameter
P47	PRH_ADS	-	G25	PRH_ADS is driven low if the device does not use multiplexed address and data bus i.e. C_PRHx_BUS_MULTIPLEX = 0
P48	PRH_BE	-	G13	The number of byte enables for the peripheral data bus is determined by the C_PRH_MAX_DWIDTH parameter
P49	PHR_RNW	-	G24	If the device is synchronous i.e. C_PRHx_SYNC = 1, then PRH_RNW should be used as the read/write control signal. For asynchronous mode of operation, PRH_RNW is driven high
P50	PRH_Rd_n	-	G24	If the device is asynchronous i.e C_PRHx_SYNC = 0, then PRH_Rd_n should be used as the read strobe. For synchronous mode of operation, PRH_Rd_n is driven high
P51	PRH_Wr_n	-	G24	If the device is asynchronous i.e C_PRHx_SYNC = 0, then PRH_Wr_n should be used as the write strobe. For synchronous mode of operation, PRH_Wr_n is driven high
P52	PRH_Burst	-	G23, G24	If the device is synchronous with data width matching enabled i.e. C_PRHx_SYNC = 1 and C_PRHx_DWIDTH_MATCH = 1, then PRH_Burst will be driven high until all but the last byte of data is flushed to the device. For asynchronous mode and synchronous mode with data width match disabled, PRH_Burst is driven low
P53	PRH_Rdy	-	G11	The number of device ready inputs decoded by the PLB external peripheral controller is determined by the C_NUM_PERIPHERALS parameter
P54	PRH_Data_I	-	G14	The peripheral input data bus width of PLB external peripheral controller is determined by the C_PRH_MAX_ADWIDTH parameter
P55	PRH_Data_O	-	G14	The peripheral output data bus width of PLB external peripheral controller is determined by the C_PRH_MAX_ADWIDTH parameter
P56	PRH_Data_T	-	G14	The peripheral output data bus 3-state control width of PLB external peripheral controller is determined by the C_PRH_MAX_ADWIDTH parameter

Allowable Parameter Combinations

When the peripheral devices are configured in non multiplexed mode, the C_PRH_MAX_AWIDTH should be the maximum of C_PRHx_AWIDTH of all peripherals in the system i.e if C_NUM_PERIPHERALS is 2, then C_PRH_MAX_AWIDTH should be maximum of C_PRH0_AWIDTH and C_PRH1_AWIDTH.

C_PRH_MAX_DWIDTH should be the maximum of C_PRHx_DWIDTH of all peripherals in the system i.e. if C_NUM_PERIPHERALS is 2, then C_PRH_MAX_DWIDTH should be maximum of C_PRH0_DWIDTH and C_PRH1_DWIDTH.

If any of the peripheral devices are configured for a multiplexed address and data buses, then the parameter C_PRH_MAX_ADWIDTH should be set as the maximum of the data bus and address bus of the device(s) employing a multiplexed address and data bus. If all devices employ non-multiplexed address and data buses, then C_PRH_MAX_ADWIDTH reflects the maximum of data bus width of all external devices. Therefore, for any configuration the parameter C_PRH_MAX_ADWIDTH should be greater than or equal to C_PRH_MAX_DWIDTH.

The generic C_PRH_CLK_SUPPORT is relevant only when the devices are configured for the synchronous access i.e. C_PRHx_SYNC = 1. If more than one of the devices are synchronous, then the frequency for the peripheral clock should be chosen as the minimum of the operating frequencies of those devices.

The range specified by C_PRHx_BASEADDR and C_PRHx_HIGHADDR parameters must comprise a contiguous range and the size of the range must be a power of two i.e. size of range = 2^m. Furthermore, the 'm' least significant bits of C_PRHx_BASEADDR must be zero. The base and high address range assigned to different peripherals must be mutually exclusive. No default value will be specified for C_PRHx_BASEADDR and C_PRHx_HIGHADDR in order to enforce that the user configures these parameters with the actual values. If the values are not set for C_PRHx_BASEADDR and C_PRHx_HIGHADDR a compiler error will be generated.

In order to access FIFO like structures within the external peripheral devices, C_PRHx_FIFO_ACCESS must be set to '1'. When FIFO access is enabled, C_PRHx_FIFO_OFFSET specifies the offset of the FIFO in terms of number of byte locations from the base address (C_PRHx_BASEADDR) of the peripheral and should be set to an offset that lies within the address range assigned to the peripheral device i.e. C_PRHx_BASEADDR + C_PRHx_FIFO_OFFSET ≤ C_PRHx_HIGHADDR. Furthermore, the FIFO offset should be within the range addressable by the address bus i.e. C_PRHx_FIFO_OFFSET < 2^n where n = C_PRHx_AWIDTH.

The width of the read/write strobe must be less than the cycle time of the corresponding access i.e. C_PRHx_WRN_WIDTH < C_PRHx_WR_CYCLE and C_PRHx_RDN_WIDTH < C_PRHx_RD_CYCLE. The value of device ready validity period must be less than the read/write strobe width and the maximum pulse width of the device ready signal i.e. C_PRHx_RDY_TOUT < minimum of C_PRHx_WRN_WIDTH, C_PRHx_RDN_WIDTH and C_PRHx_RDY_WIDTH. The time period between two consecutive writes (when the data width matching is enabled) will be decided by maximum of (C_PRHx_WR_CYCLE-C_PRHx_WRN_WIDTH), C_PRHx_CSN_TH and C_PRHx_DATA_TH parameters. Similar calculation is applicable for consecutive reads.

Please make sure that you have configured the timing parameters properly while using XPS EPC IP Core at different frequencies on various FPGA devices.

XPS EPC Design Considerations

The XPS EPC IP Core is PLB 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. The user must take the following considerations into account while designing with the XPS EPC.

How to Provide the Timing Parameters in Async Mode of Operation?

Internally in the XPS EPC IP Core, the timing parameters are inter-related and some calculation is done based upon the values provided by the user. These values and internal calculations are as follows. In case of asynchronous mode of operation following parameters are used. These parameters are,-

- C_PRHx_ADDR_TSU - address set up time
- C_PRHx_ADDR_TH - address hold time
- C_PRHx_ADS_WIDTH - address strobe width
- C_PRHx_CSN_TSU - chip select set up time
- C_PRHx_CSN_TH - chip select hold time
- C_PRHx_WRN_WIDTH - write control width time
- C_PRHx_WR_CYCLE - write cycle time i.e. time between two consecutive writes
- C_PRHx_DATA_TSU - data set up time
- C_PRHx_DATA_TH - data hold time
- C_PRHx_RDN_WIDTH - read control width time
- C_PRHx_RD_CYCLE - read cycle time i.e. time between two consecutive reads
- C_PRHx_DATA_TOUT - time taken by the data to be out at read condition
- C_PRHx_DATA_TINV - data line tri-state after the read control signal is de-activated
- C_PRHx_RDY_TOUT - time for device ready signal to go high, once device is selected
- C_PRHx_RDY_WIDTH - maximum time till the XPS EPC IP Core can wait for device to be ready

How these parameters are linked in the design?

Address hold time - This parameter is calculated as,
(C_PRHx_ADDR_TH/C_BUS_CLOCK_PERIOD_PS)

Chip select/Data/Address hold time - Three parameters are used to calculate the hold time as,
((max2(max2(C_PRHx_DATA_TH,C_PRHx_CSN_TH),C_PRHx_ADDR_TH)/
C_BUS_CLOCK_PERIOD_PS)

Read control signal width time - This parameter is calculated as,
(C_PRHx_RDN_WIDTH/C_BUS_CLOCK_PERIOD_PS)

Write control signal width time - This parameter is calculated as,
(C_PRHx_WRN_WIDTH/C_BUS_CLOCK_PERIOD_PS)

Address strobe signal width time - Three parameters are used to calculate the address strobe width as,
((max2(max2(C_PRHx_ADDR_TSU, C_PRHx_ADS_WIDTH), C_PRHx_CSN_WIDTH)/
C_BUS_CLOCK_PERIOD_PS)

Write recovery time for non-multiplexed case is calculated as, ((max2(max2(C_PRHx_CSN_TH, C_PRHx_DATA_TH),max2(C_PRHx_ADDR_TH,PRH_Wr_nx))/C_BUS_CLOCK_PERIOD_PS). where PRH_Wr_nx = (C_PRHx_WR_CYCLE - C_PRHx_WRN_WIDTH).

Read recovery time for non-multiplexed case is calculated as, ((max2(max2(C_PRHx_CSN_TH, C_PRHx_DATA_TINV),max2(C_PRHx_ADDR_TH,PRH_Rd_nx))/C_BUS_CLOCK_PERIOD_PS)

Write recovery time for multiplexed case is calculated as, ((max2(C_PRHx_CSN_TH, C_PRHx_DATA_TH),PRH_Wr_nx))/C_BUS_CLOCK_PERIOD_PS) . Where PRH_Wr_nx = (C_PRHx_WR_CYCLE - C_PRHx_WRN_WIDTH).

Read recovery time for multiplexed case is calculated as, ((max2(C_PRHx_CSN_TH, C_PRHx_DATA_TINV),PRH_Rd_nx))/C_BUS_CLOCK_PERIOD_PS). Where PRH_Rd_nx = (C_PRHx_RD_CYCLE - C_PRHx_RDN_WIDTH).

Device Ready signal time - This parameter is calculated as, (C_PRHx_RDY_TOUT/C_BUS_CLOCK_PERIOD_PS) .

Maximum time for Device Ready signal to be active - This parameter is calculated as, (C_PRHx_RDY_WIDTH/C_BUS_CLOCK_PERIOD_PS) .

Some of the above calculations are repeated for different use for internal purpose. Please note that there are some dummy states included in between the states to meet the design requirements. The user shall make sure that these timing parameter value calculations will not exceed the maximum time of 128 clock cycles (which is PLB limitation).

Following Table 4 indicates the ranges for the above mentioned parameters.

Table 4: **Example of Timing Ranges for XPS EPC IP Core in Asynchronous Mode of Operation**

Parameter Name	Timing parameter's range		
C_PRHx_ADDR_TSU	6000 (set 1)	20000 (set 2)	40000 (set 3)
C_PRHx_ADDR_TH	6000	20000	30000
C_PRHx_ADS_WIDTH	10000	20000	40000
C_PRHx_CSN_TSU	6000	20000	40000
C_PRHx_CSN_TH	6000	20000	30000
C_PRHx_WRN_WIDTH	15000	30000	30000
C_PRHx_WR_CYCLE	30000	60000	60000
C_PRHx_DATA_TSU	10000	20000	30000
C_PRHx_DATA_TH	5000	20000	30000
C_PRHx_RDN_WIDTH	15000	30000	30000
C_PRHx_RD_CYCLE	30000	60000	60000
C_PRHx_DATA_TOUT	5000	5000	15000
C_PRHx_DATA_TINV	10000	15000	25000

Table 4: **Example of Timing Ranges for XPS EPC IP Core in Asynchronous Mode of Operation**

Parameter Name	Timing parameter's range		
C_PRHx_RDY_TOUT	10000(set 1)	50000(set 2)	60000(set 3)
C_PRHx_RDY_WIDTH	50000	100000	120000

Note:

1. Please note that the above range of timing parameters (set 1) are used for internal device utilization and while testing on USB and LAN. While using the above parameters care had been taken that all parameters of same set are used.
2. Please note that the above timing sets (set 2 and set 3) are just an example of how the timing parameters can be given to XPS EPC IP Core. The user can have different timing parameter values as per the targeted device's data sheet. In such cases, please note the calculation given above.

FIFO Transactions

When C_PRHx_FIFO_ACCESS = 1, the XPS EPC IP Core supports access to FIFO's 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 XPS EPC IP Core does not increment the peripheral address (PRH_Addr) bus.

Abnormal Terminations

If PRH_Rdy is de-asserted for more than the maximum period as specified by C_PRHx_RDY_WIDTH then the XPS EPC terminates the current access to the external device and signals an error to the PLB master by asserting either Sl_MWrErr or Sl_MRdErr on the PLB. Similarly, when the PLB master terminates the current access (master abort on the PLB), XPS EPC terminates the access to the external device immediately. In both cases, the access to the external device is terminated abnormally. Therefore, the external device may 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 XPS 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 active, this indicates that the peripheral device is ready for communication. Once the PRH_Rdy goes active, it is expected that it will remain in the same active state till that particular transaction is completed by the XPS EPC IP Core. Please note that, if the XPS EPC IP Core is implemented with data width match enabled, then the activeness of the PRH_Rdy signal will always be checked for every transaction, regardless of it being active throughout the transacction.

User should refer to the data sheet of the external peripheral, to define the value of PRH_Rdy time period i.e. C_PRHx_RDY_TOUT parameter value should be filled based on the above information. It is also expected that the user will provide maximum waiting period for PRH_Rdy signal. This waiting period will be indicated by C_PRHx_RDY_WIDTH parameter. The C_PRHx_RDY_TOUT parameter value should be less than the C_PRHx_RDY_WIDTH parameter value. The C_PRHx_RDY_WIDTH time period should be lower than the PLB IPIF time out value i.e. C_PRHx_RDY_TOUT <

C_PRHx_RDY_WIDTH < IPIF time out(128 PLB cycles). Starting from the falling edge of the PRH_WR_n or, PRH_RD_n signal, the internal counter for C_PRHx_RDY_TOUT & C_PRHx_RDY_WIDTH will start. The XPS EPC IP Core expects that the PRH_Rdy signal should appear before any of these timer ends. If the PRH_Rdy doesn't become active before the C_PRHx_RDY_TOUT counter ends, then the XPS EPC IP Core will wait till the end of C_PRHx_RDY_WIDTH counter. In case, PRH_Rdy becomes active then internal state machine will proceed to complete the further process steps. If PRH_Rdy doesn't become active even before the C_PRHx_RDY_WIDTH ends, the XPS EPC IP Core will generate an error and terminate the transaction. User can re-initiate the same transaction later on.

Figure 3 below shows the diagrammatic representation of how the async state machine reacts with the PRH_Rdy signal. Please 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 the states will go to IDLE by default.

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

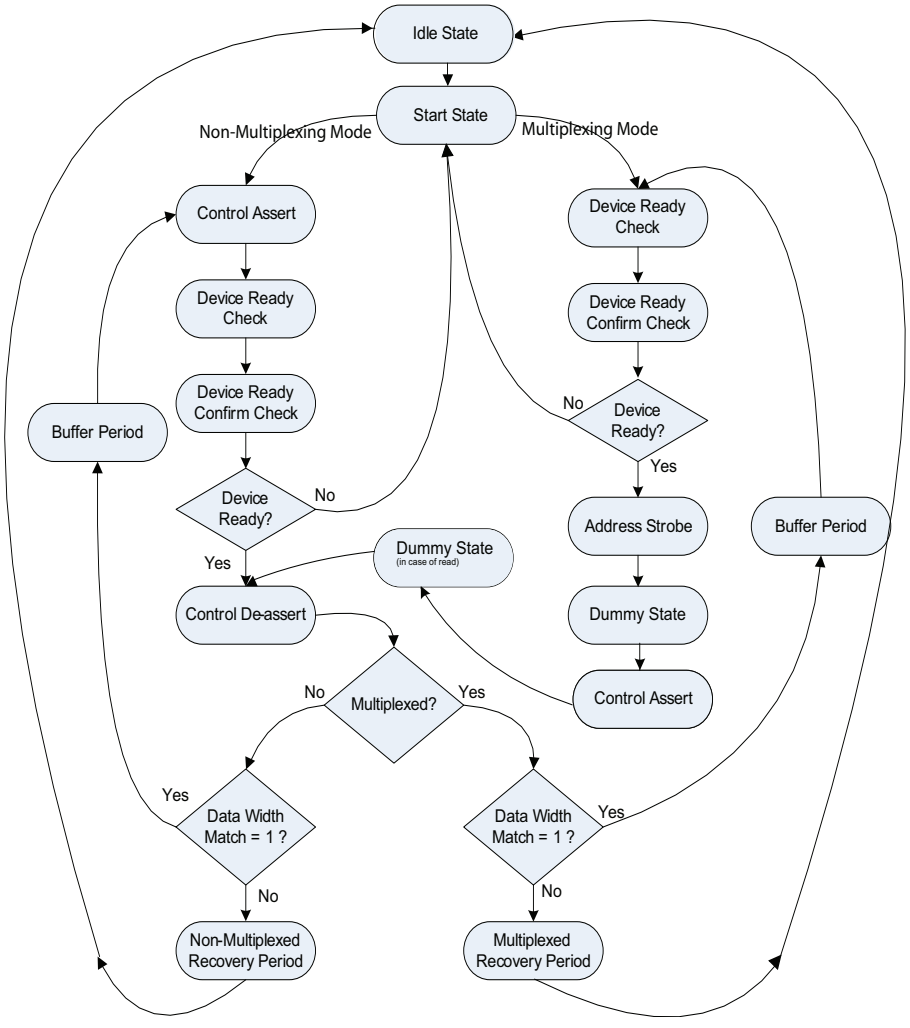

Figure 3: **Diagrammatic Representation of Async State Machine Flow**

Peripheral Data Bus Mapping

The peripheral data bus (PRH_Data) uses big endian bit labelling (i.e. 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 C_PRH_MAX_ADWIDTH 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-bit, 16-bit and 32-bit, 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 4.

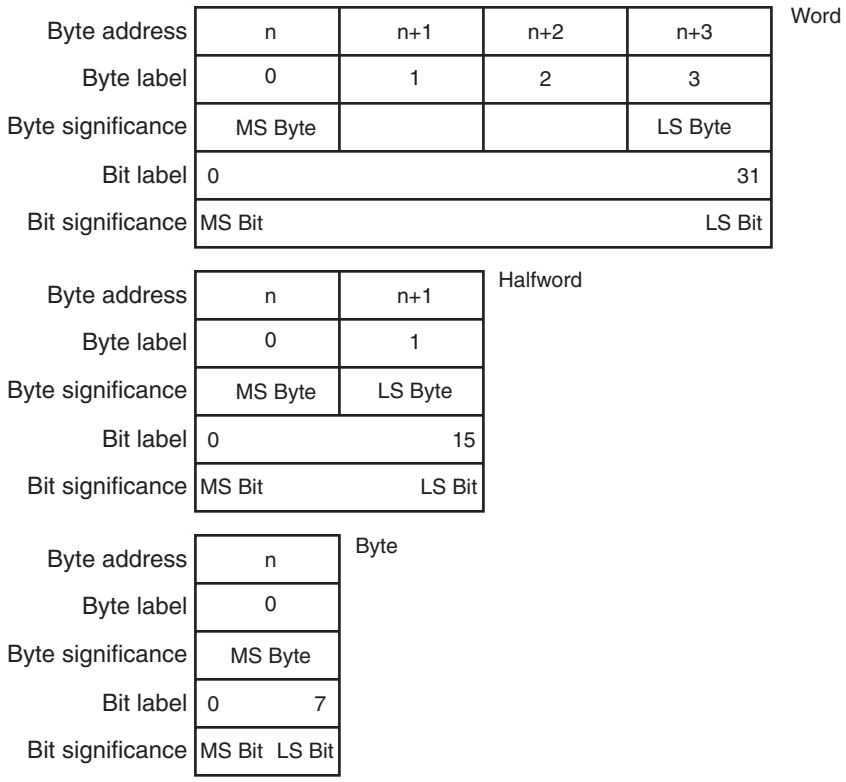

Figure 4: **XPS EPC Big Endian Data Type**

Unsupported Features

Many peripheral devices have the device specific input/output ports such as status, remote reset, remote wakeup, interrupts etc. The XPS 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 may 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 XPS EPC IP Core is a PLB slave device and therefore does not support DMA operations from the external peripheral devices.

Current version of the XPS EPC IP Core does not support burst transactions to/from the PLB. So the parameter C_PRH_BURST_SUPPORT must be always assigned to '0'.

XPS EPC IP Core External Peripheral Connections

The XPS EPC IP Core interface to the external device is based upon the width of the PLB 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 data path.

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 XPS 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 XPS 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 i.e. 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 must be exercised with the connection to the external peripheral devices to avoid incorrect data and address connections.

Clock Generation

When connecting to a synchronous external device, the XPS EPC IP Core may operate either on the PLB clock (SPLB_Clk) or on a peripheral clock (PRH_Clk). The operating clock for the synchronous interface is based on the generic C_PRH_CLK_SUPPORT. If C_PRH_CLK_SUPPORT = 1, the peripheral clock is used else the PLB clock is used. The external devices connected to the XPS EPC IP Core will determine the operating frequency of the peripheral clock. The minimum of the operating frequencies of various devices connected to XPS EPC IP Core should be used as the operating frequency of the peripheral clock (PRH_Clk).

Figure 5 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 XPS EPC IP Core. An external clock source is used to generate the XPS clock (SPLB_Clk) to the XPS 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 XPS_EPC.

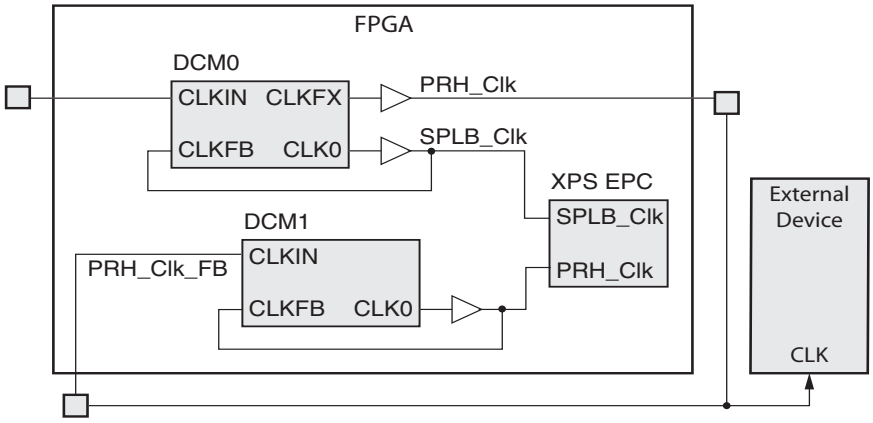

Figure 5: **External Peripherals clocked by FPGA output with feedback**

Example Peripheral Connections

Example: SMSC LAN91C111 10/100 non-PCI Ethernet Single Chip MAC + PHY

An example peripheral device supported by the XPS EPC IP Core is the SMSC 10/100 non-PCI Ethernet Single Chip MAC + PHY LAN91C111. The XPS EPC IP Core allows PLB 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 6 illustrates the SMSC LAN91C111 connections to the XPS 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 XPS EPC IP Core. Since the address and the data buses are not multiplexed, PRH_ADS will be driven low by the XPS EPC IP Core. The timing parameters should be set according to the SMSC LAN 91c111 Async mode specifications.

The following parameter settings apply for the connections shown in Figure 6.

- C_PRH_CLK_SUPPORT = 0
- C_PRH0_AWIDTH = 16
- C_PRH0_DWIDTH = 32
- C_PRH0_SYNC = 0
- C_PRH0_DWIDTH_MATCH = 0
- C_PRH0_BUS_MULTIPLEX = 0

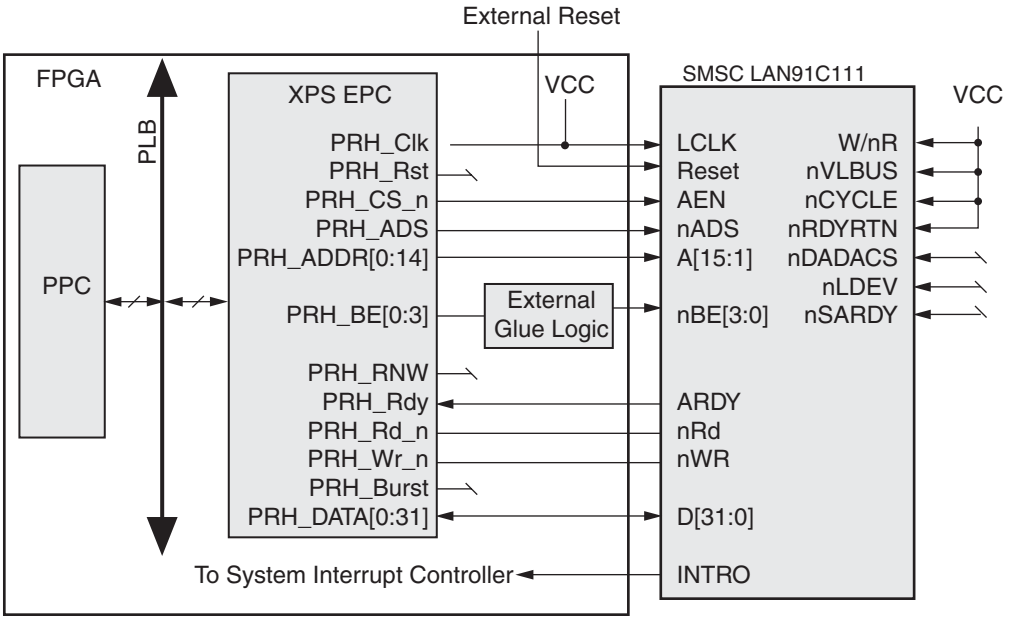

Figure 6: **SMSC LAN91C111 Connection to XPS EPC IP Core in Asynchronous Mode**

External Glue Logic

Special attention must be given while interfacing the XPS EPC IP Core to the SMSC LAN91C111 in asynchronous mode. The XPS 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 glue 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 7 illustrates the example for SMSC LAN91C111 connection to the XPS 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 50MHz. 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 XPS 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 XPS_EPC. As the address and the data bus are not multiplexed, PRH_ADS will be driven low.

The following parameter settings apply for the connections shown in Figure 7.

- C_PRH_CLK_SUPPORT = 1
- C_PRH0_AWIDTH = 16
- C_PRH0_DWIDTH = 32
- C_PRH0_SYNC = 1
- C_PRH0_DWIDTH_MATCH = 0
- C_PRH0_BUS_MULTIPLEX = 0

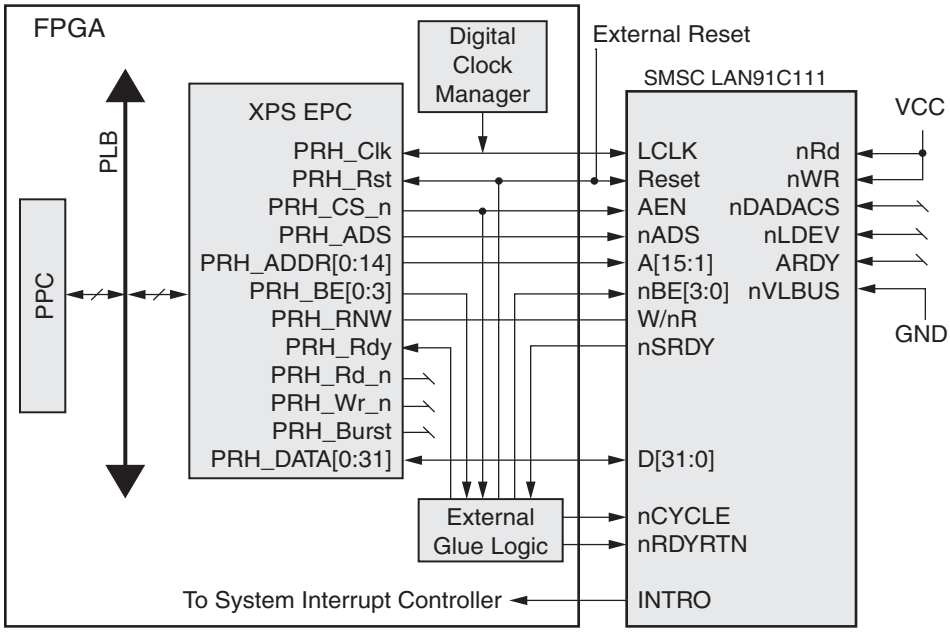

Figure 7: **SMSC LAN91C111 Connection to XPS EPC IP Core in Synchronous Mode**

External Glue 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 documents listed in the Reference Documents section. The nCYCLE and nRDYRTN signals are generated in external glue logic. nCYCLE may 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 simply 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 will be generated twice from the core, first during the address phase and second time during the data phase. User should take care while writing the logic for nCYCLE signal generation in external glue 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 XPS EPC IP Core is interfaced to the SMSC LAN91C111 with data width match enabled (C_PRHx_DWIDTH_MATCH = 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. More information can be obtained by referring the registers of Bank 2 in SMSC LAN91C111 data sheet. See the Reference Documents section.

Example: Cypress Semiconductor's CY7C67300 EZ-Host Programmable Embedded USB Host/Peripheral Controller

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

Asynchronous Mode

Figure 8 illustrates the CY7C67300 USB Controller connection to the XPS 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 8.

- C_PRH_CLK_SUPPORT = 0
- C_PRH0_AWIDTH = 4
- C_PRH0_DWIDTH = 16
- C_PRH0_SYNC = 0
- C_PRH0_DWIDTH_MATCH = 1
- C_PRH0_BUS_MULTIPLEX = 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 Compact Flash memory device and is accessed through the XPS SYSACE interface. This code is downloaded to the CY7C67300 USB Controller through the HPI interface. Once 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 (XPS EPC IP Core). The XPS EPC address bits which corresponds to word boundary are only used in the 2-bit address bus interface with the Cypress Semiconductor's USB device.

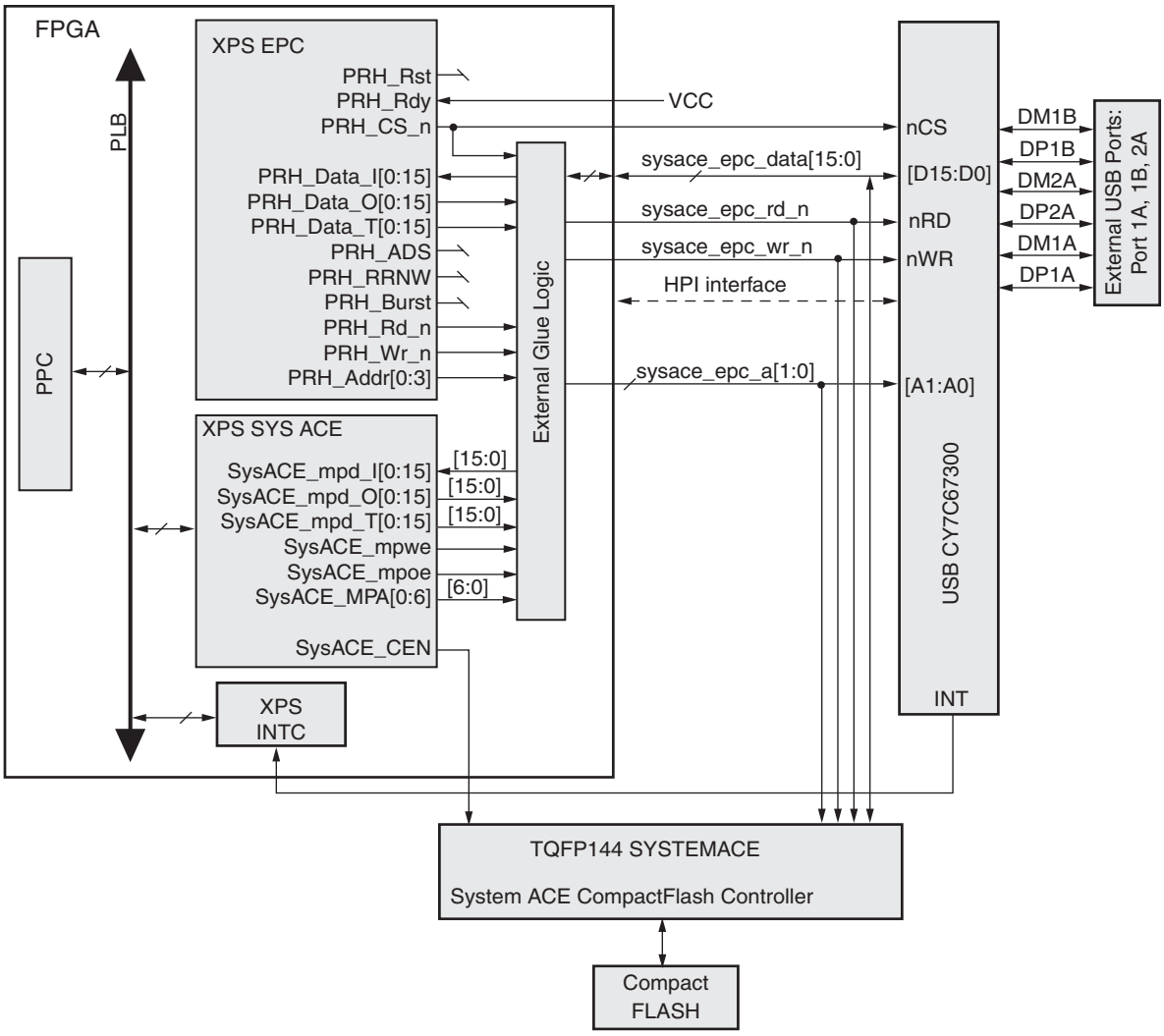

Figure 8: **CY7C67300 USB Controller connection to XPS EPC in asynchronous mode**

External Glue Logic

In the interface diagram example shown in Figure 8, the XPS EPC IP Core as well as the XPS 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 XPS EPC IP Core control signals PRH_RD_n and PRH_WR_n are muxed with the MEM_CEN and MEM_WEN signals of the XPS 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. See the Reference Documents section for more information on XPS SYSACE.

Design Constraints

Timing Constraints

Timing constraints must be placed on the system clock and the peripheral clock, setting the frequency to meet the bus timing requirements. An example is shown in Figure 9.

Following constraint must be placed on the system clock:

```
NET SPLB_Clk TNM_NET = SPLB_Clk;
TIMESPEC TS_SPLB_Clk = PERIOD SPLB_Clk 10 ns HIGH 50%;
```

If C_PRH_CLK_SUPPORT=1, then the following constraint must be placed on the peripheral clock:

```
NET PRH_Clk TNM_NET = PRH_Clk;
TIMESPEC TS_PRH_Clk = PERIOD PRH_Clk 20 ns HIGH 50%;
```

Figure 9: **XPS EPC Timing Constraints**

Design Implementation

Timing Diagrams

The timing diagrams in the figures below show various PLB 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 XPS EPC IP Core .

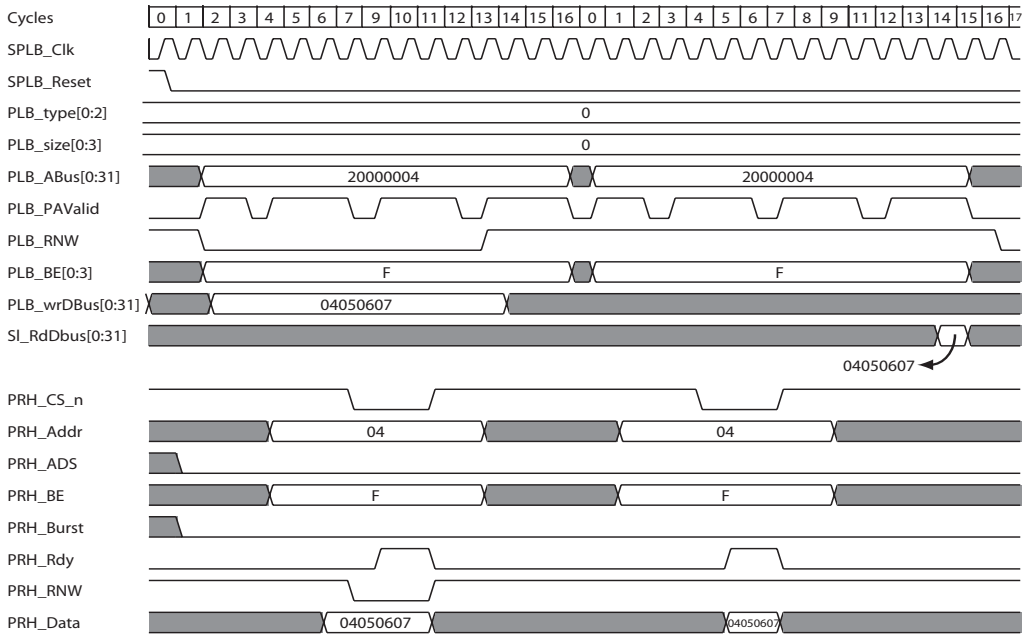

Figure 10: **Synchronous Write-Read Transactions to Device Memory When Bus is not Multiplexed and Data Width Matching is Disabled (C_PRH_CLK_SUPPORT = 0), assuming the peripheral device is ready**

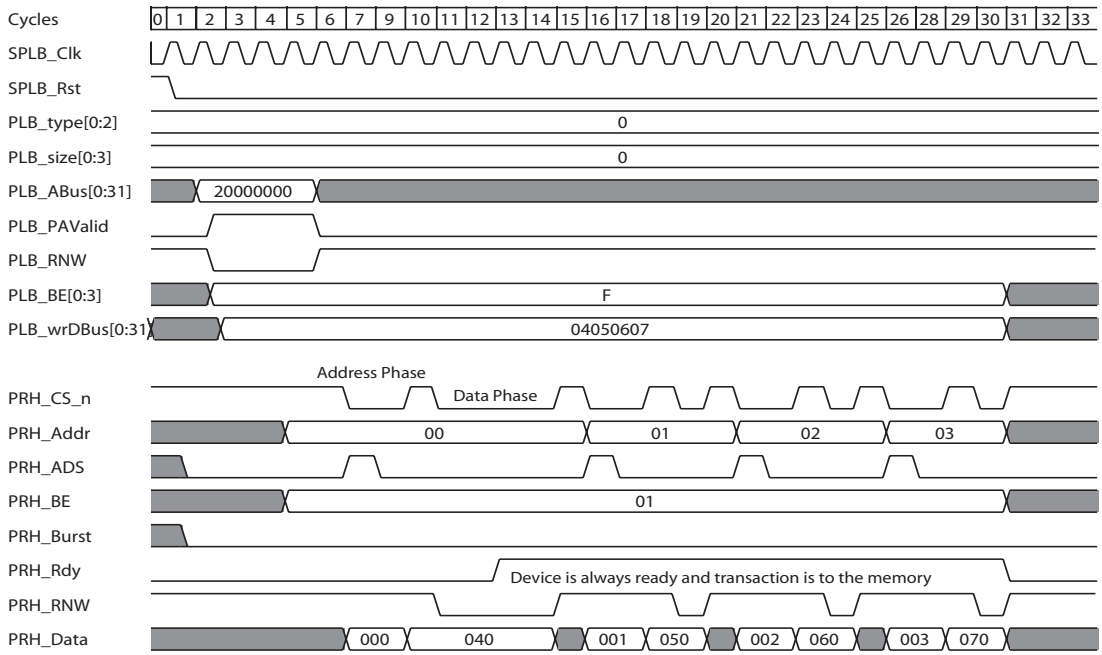

Figure 11: **Synchronous Write Transactions to Device Memory When Bus is Multiplexed and Data Width Matching is Enabled (C_PRH_CLK_SUPPORT = 0)**

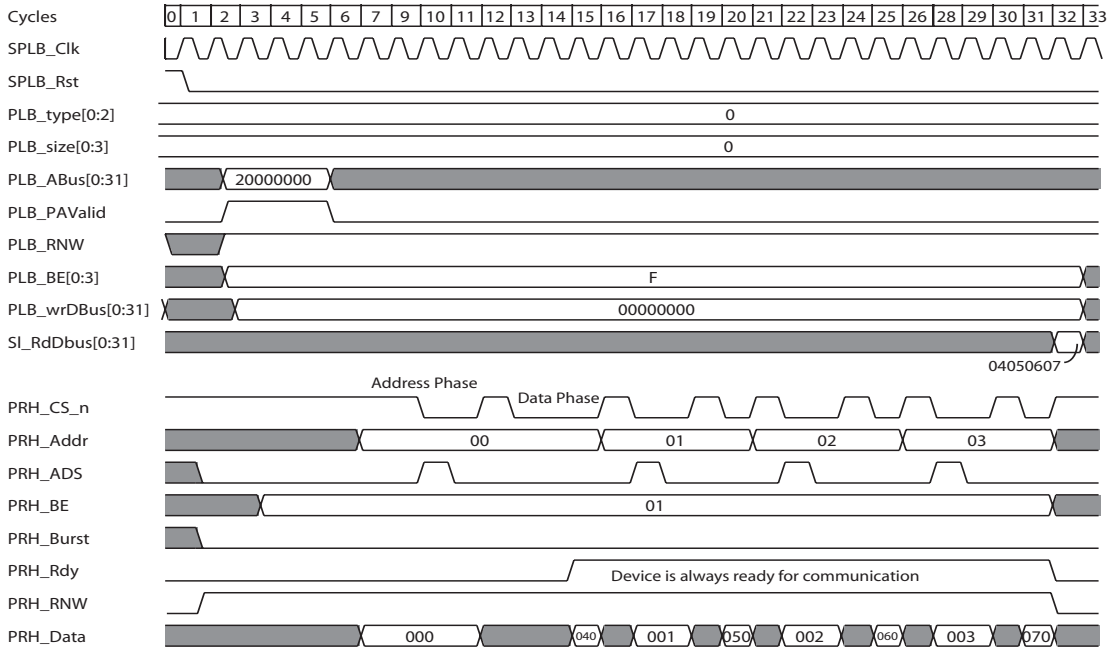

Figure 12: **Synchronous Read Transactions to Device Memory When Bus is Multiplexed and Data Width Matching is Enabled (C_PRH_CLK_SUPPORT = 0)**

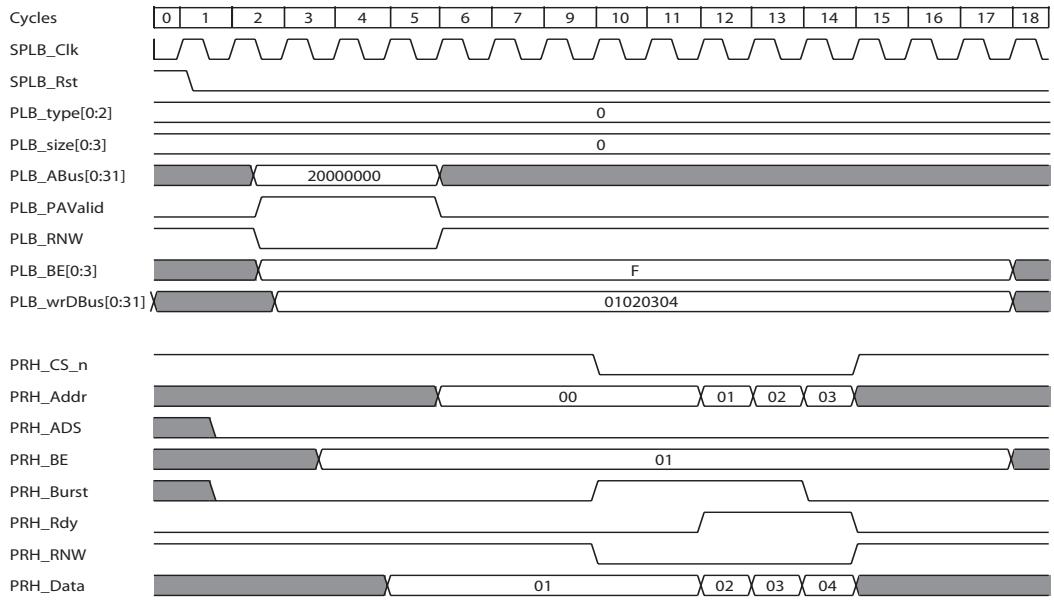

Figure 13: **Synchronous Write Transactions to Device Memory When Bus is Not Multiplexed and Data Width Matching is Enabled (C_PRH_CLK_SUPPORT = 0)**

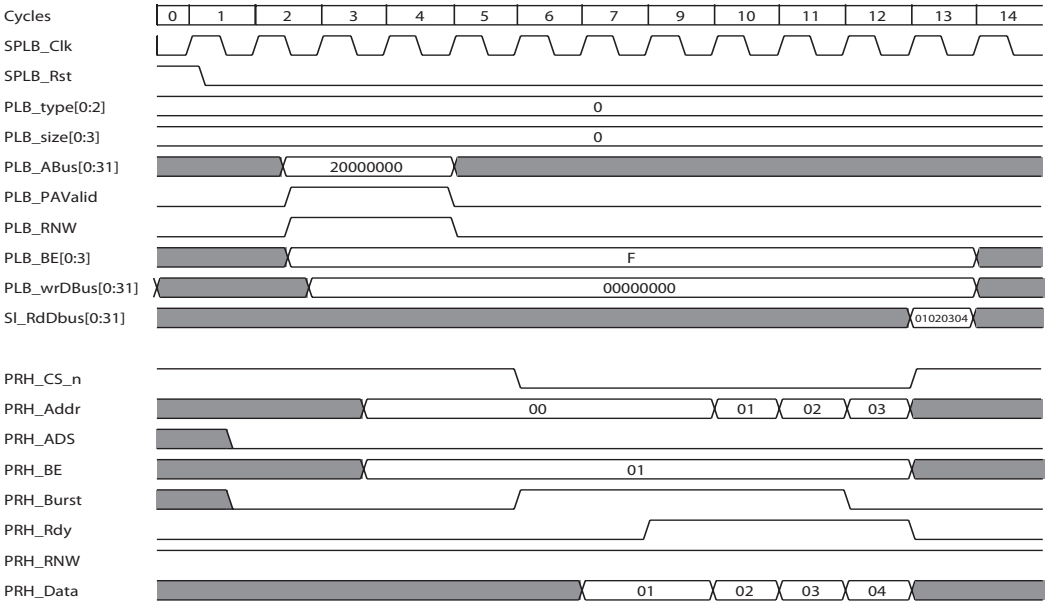

Figure 14: **Synchronous Read Transactions to Device Memory When Bus is Not Multiplexed and Data Width Matching is Enabled (C_PRH_CLK_SUPPORT = 0)**

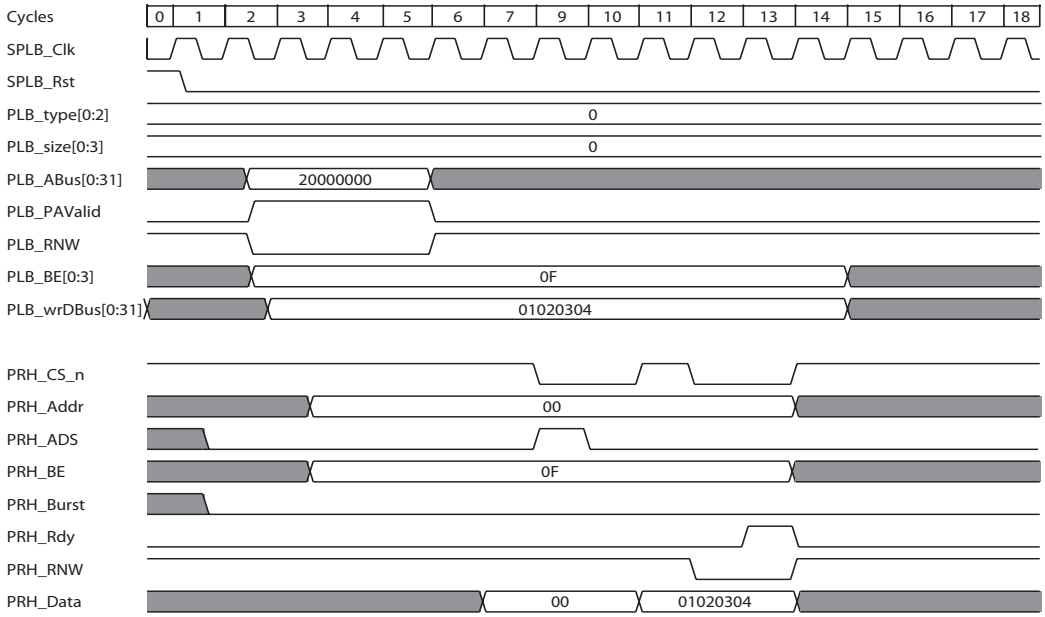

Figure 15: **Synchronous Write Transactions to Device Memory When Bus is Multiplexed and Data Width Matching is Disabled (C_PRH_CLK_SUPPORT = 0)**

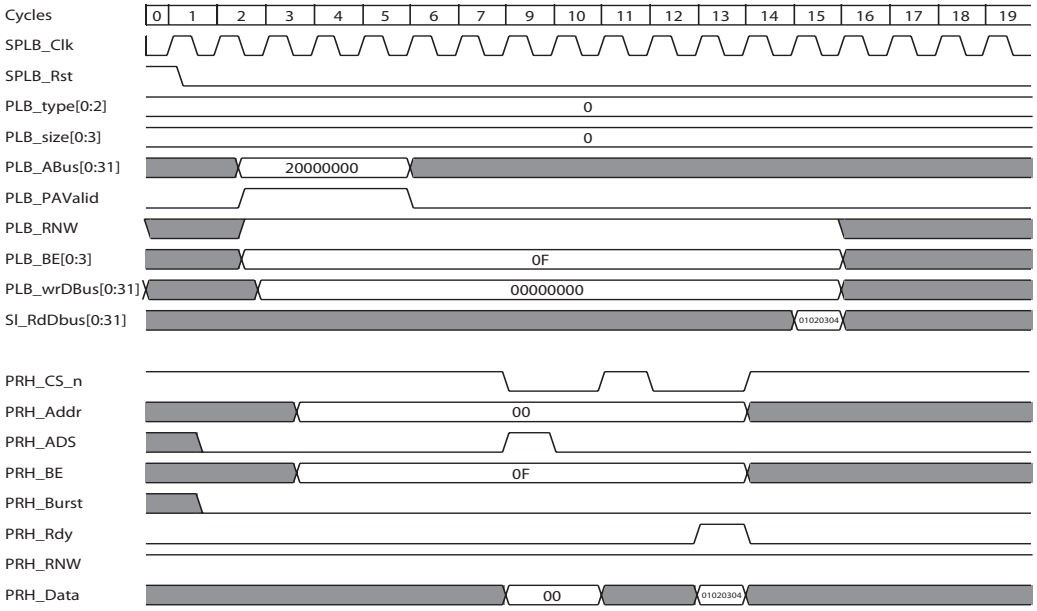

Figure 16: **Synchronous Read Transactions to Device Memory When Bus is Multiplexed and Data Width Matching is Disabled (C_PRH_CLK_SUPPORT = 0)**

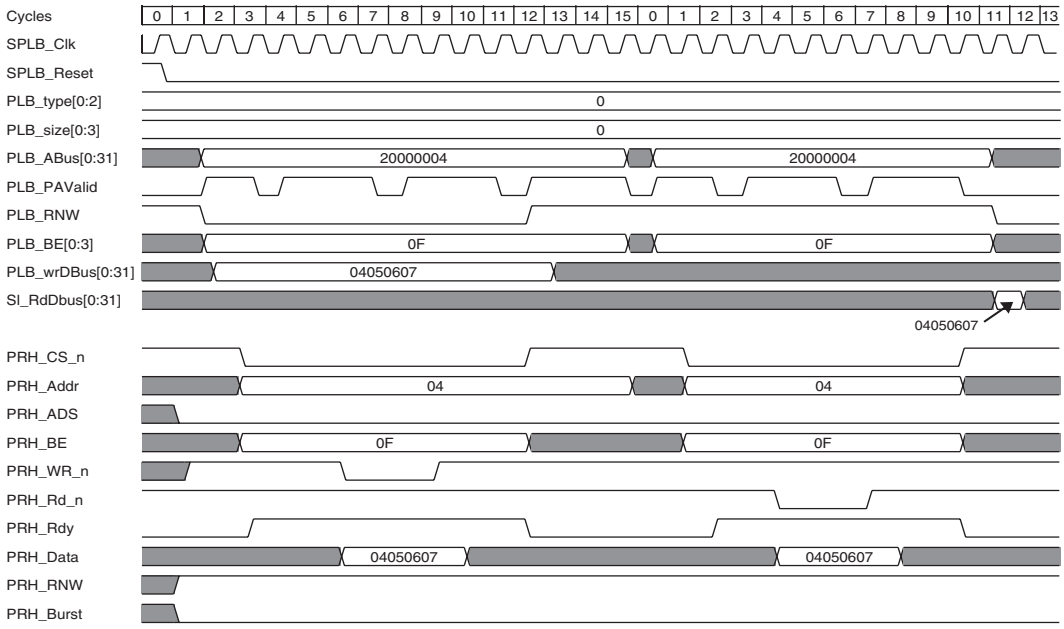

Figure 17: **Asynchronous Write-Read Transactions to Device Memory When Bus is Not Multiplexed and Data Width Matching is Disabled (C_PRH_CLK_SUPPORT = 0)**

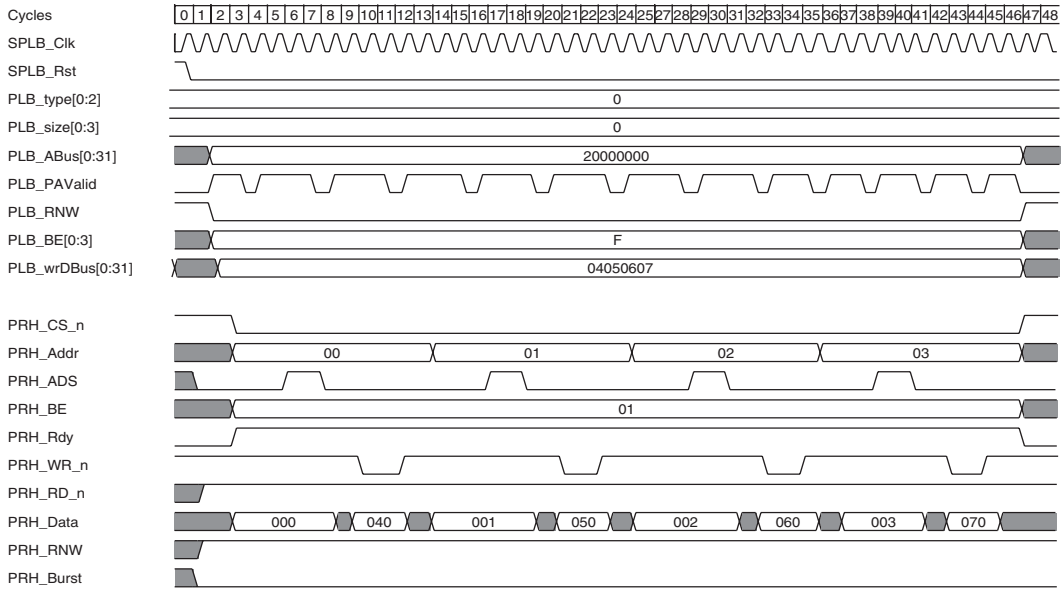

Figure 18: **Asynchronous Write Transactions to Device Memory When Bus is Multiplexed and Data Width Matching is Enabled (C_PRH_CLK_SUPPORT = 0)**

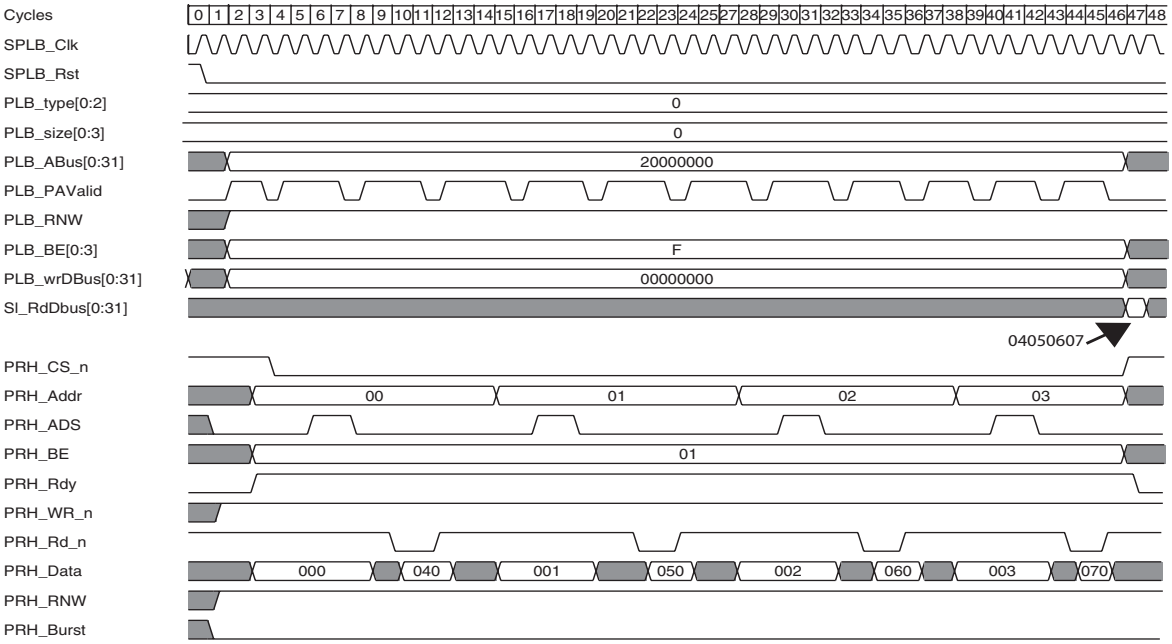

Figure 19: **Asynchronous Read Transactions to Device Memory When Bus is Multiplexed and Data Matching is Enabled (C_PRH_CLK_SUPPORT = 0)**

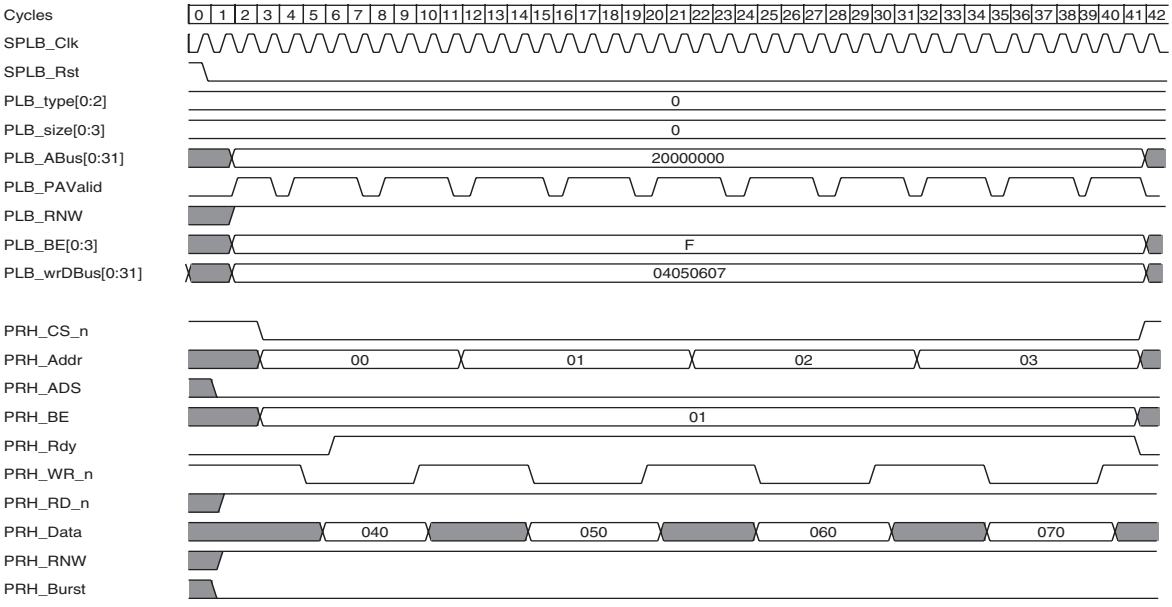

Figure 20: **Asynchronous Write Transactions to Device Memory When Bus is Not Multiplexed and Data Width Matching is Enabled**

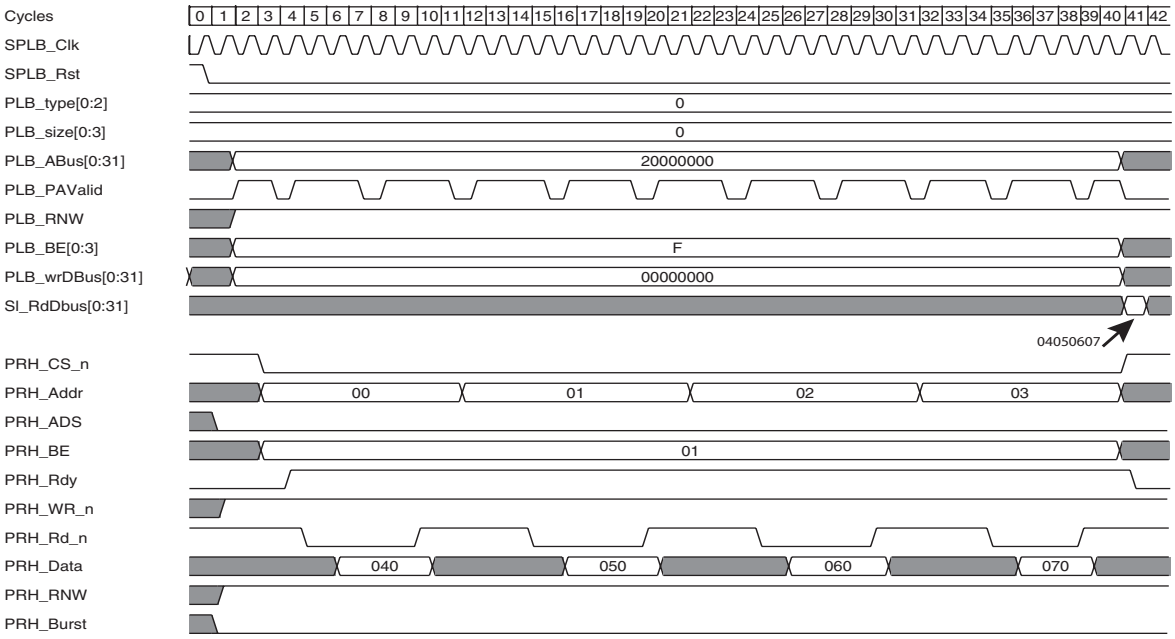

Figure 21: **Asynchronous Read Transactions to Device Memory When Bus is Not Multiplexed and Data Width Matching is Enabled**

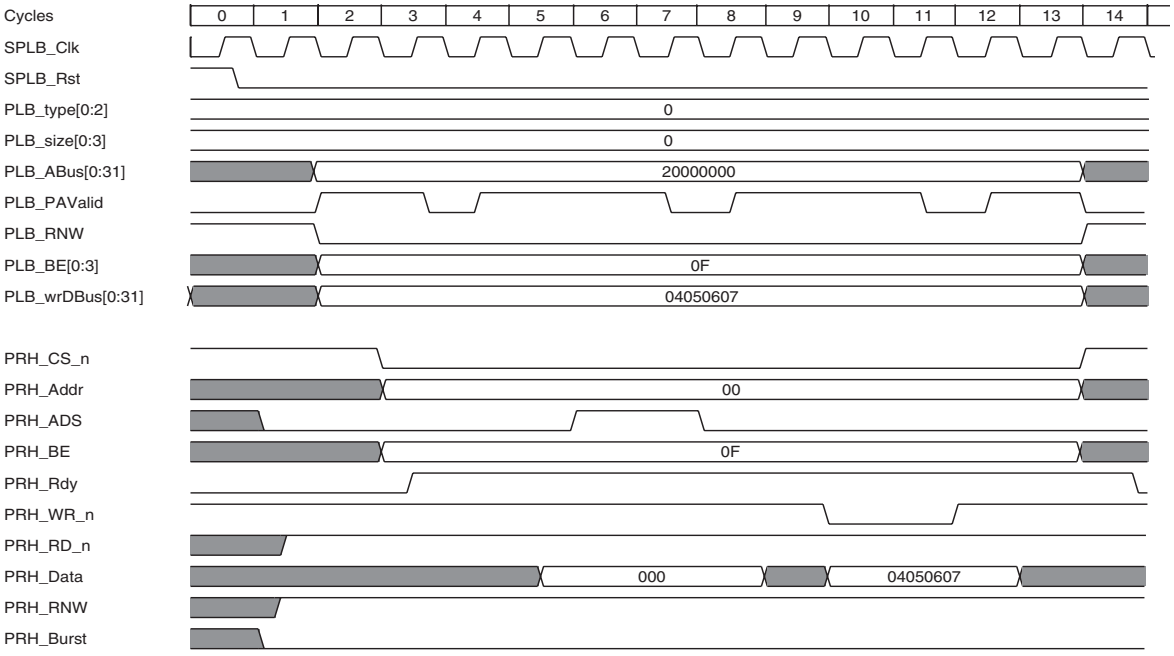

Figure 22: **Asynchronous Write Transactions to Device Memory When Bus is Multiplexed and Data Width Matching is Disabled**

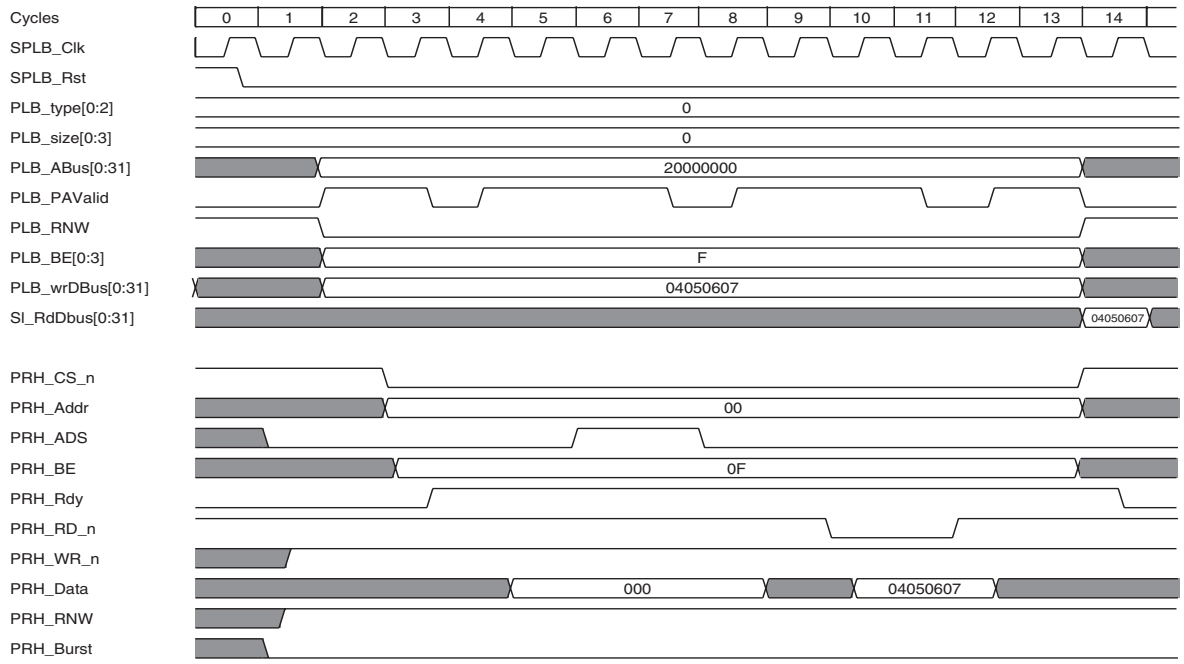

Figure 23: **Asynchronous Read Transactions to Device Memory When Bus is Multiplexed and Data Width Matching is Disabled**

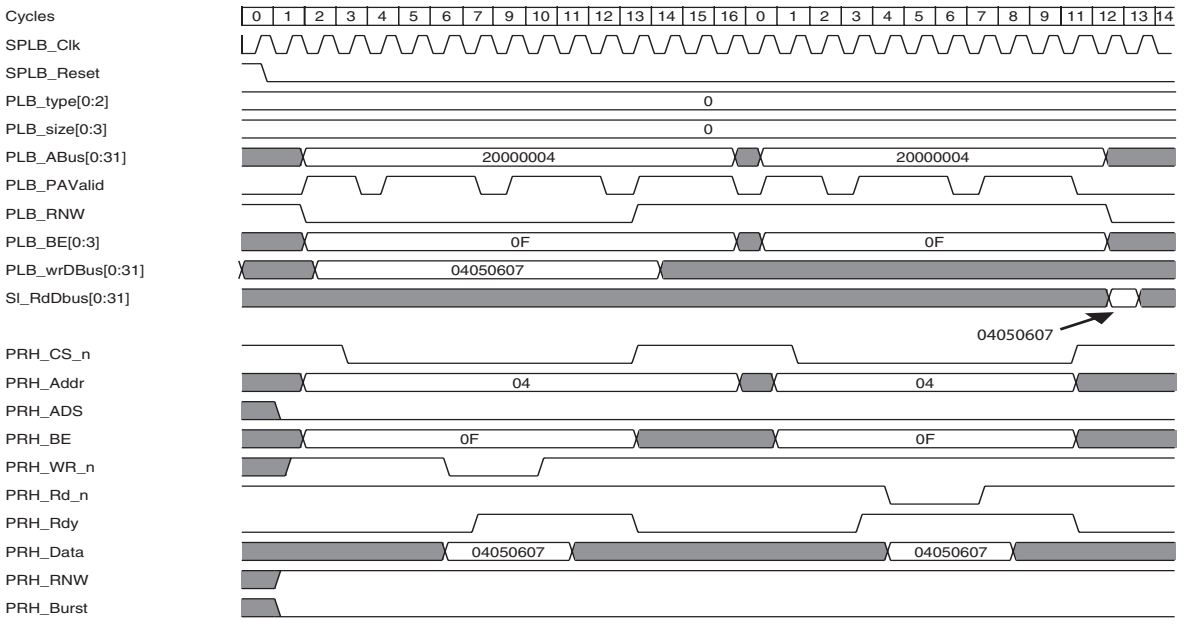

Figure 24: **Asynchronous Write-Read Transactions to Device Memory When Bus is Not Multiplexed and Data Width Matching is Disabled (C_PRH_CLK_SUPPORT = 0). Please note the delayed response of PRH_Rdy signal.**

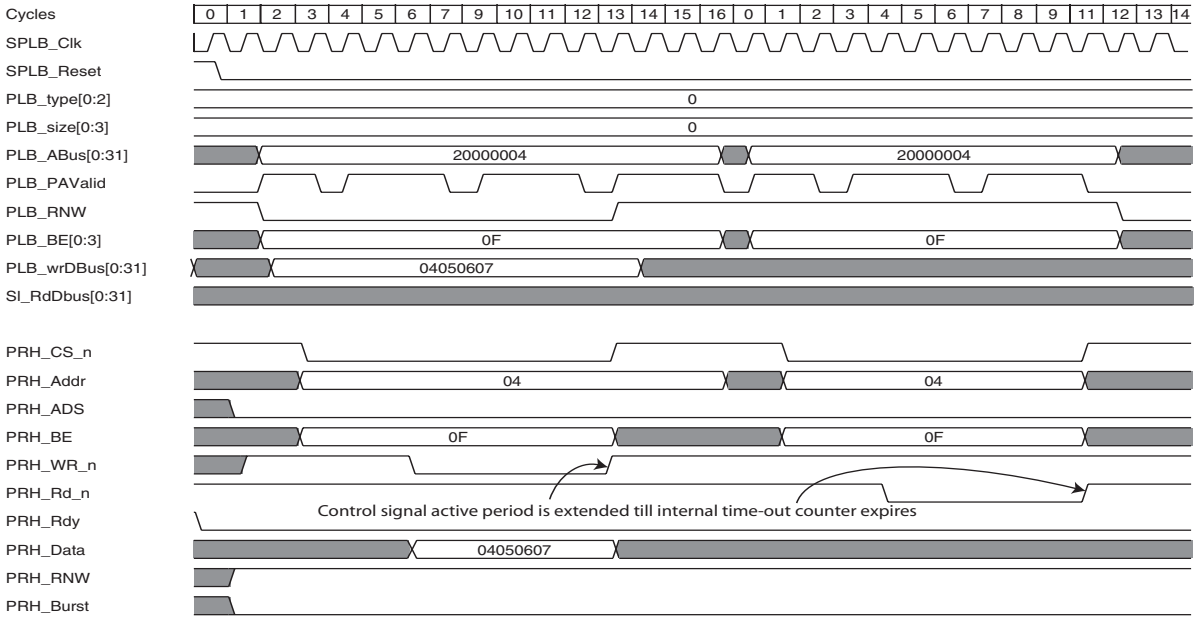

Figure 25: **Asynchronous Write-Read Transactions to Device Memory When Bus is Not Multiplexed and Data Width Matching is Disabled (C_PRH_CLK_SUPPORT = 0). Please note that the external peripheral device is not ready for communication and no PRH_Rdy comes.**

Scenario When The External Device Is Not Ready For Transaction

In scenarios where, the XPS EPC IP Core has initiated the transaction, but the external peripheral in not ready (PRH_Rdy signal will not be activated), the control signals from the core will be extended till the internal C_PRHx_RDY_WIDTH counter expires. The core completes the transaction with error signal generation.

Target Technology

The target technology is an FPGA listed in EDK Supported Device Families.

Device Utilization and Performance Benchmarks

Core Performance

Because the XPS EPC module will be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates. When the XPS EPC module is combined with other designs, the utilization of FPGA resources and timing of the XPS EPC IP Core design will vary from the results reported here.

The XPS EPC IP Core resource utilization for various parameter combinations measured with Virtex®-4 as the target device are detailed in Table 5.

Table 5: **Performance and Resource Utilization Benchmarks on Virtex-4 (Device: xc4vlx80-ff1148-10)**

Parameter Values							Device Resources			
C_NUM_PERIPHERALS	C_PRH_CLK_SUPPORT	C_PRHx_AWIDTH	C_PRHx_DWIDTH	C_PRHx_SYNC	C_PRHx_BUS_MULTIPLEX	C_PRHx_DWIDTH_MATCH	Slices	Slice Flip- Flops	LUTs	F_{Max} (MHz)
1	0	3	32	0	0	0	189	260	179	192
1	0	6	16	0	0	1	199	261	233	182
1	0	6	8	0	0	1	210	253	241	158
1	0	6	8	0	1	1	237	271	297	175
1	0	6	16	0	1	1	224	279	283	164
1	0	6	32	0	1	1	222	289	235	178
2	0	6,12	32,16	0,0	1,0	0,1	273	323	371	168
3	0	6,12,6	32,16,8	0,0,0	1,0,1	0,1,1	303	329	433	142
4	0	6,12,6,3	32,16,8,32	0,0,0,0	1,0,1,0	0,1,1,0	321	340	468	128
1	0	3	32	1	0	0	153	237	117	205
1	0	6	16	1	0	1	180	238	198	178
1	0	6	8	1	0	1	180	234	181	178
1	0	6	8	1	1	1	194	239	209	134
1	0	6	16	1	1	1	194	245	220	177
1	0	6	32	1	1	0	169	252	144	202
2	0	6,12	32,16	1,1	1,0	0,1	234	287	319	150
3	0	6,12,6	32,16,8	1,1,1	1,0,1	0,1,1	273	295	373	128
4	0	6,12,6,3	32,16,8,32	1,1,1,1	1,0,1,0	0,1,1,0	269	302	359	134

Table 5: **Performance and Resource Utilization Benchmarks on Virtex-4 (Device: xc4vlx80-ff1148-10) *(Contd)***

Parameter Values							Device Resources			
C_NUM_PERIPHERALS	C_PRH_CLK_SUPPORT	C_PRHx_AWIDTH	C_PRHx_DWIDTH	C_PRHx_SYNC	C_PRHx_BUS_MULTIPLEX	C_PRHx_DWIDTH_MATCH	Slices	Slice Flip- Flops	LUTs	F_{Max} (MHz)
3	0	12,6,12	32,32,32	0,1,1	0,0,0	0,0,0	275	364	281	140
3	0	12,12,12	32,32,32	1,0,0	0,0,0	0,0,0	276	364	282	155

Notes:

1. The bus multiplex timing parameters and asynchronous timing parameters are set to typical values for all devices. The values used for various timing parameters are as follows:
   - C_PRHx_ADDR_TSU = 6ns
   - C_PRHx_ADDR_TH = 6ns
   - C_PRHx_ADS_WIDTH = 10ns
   - C_PRHx_CSN_TSU = 6ns
   - C_PRHx_CSN_TH = 6ns
   - C_PRHx_WRN_WIDTH = 15ns
   - C_PRHx_WR_CYCLE = 30ns
   - C_PRHx_DATA_TSU = 10ns
   - C_PRHx_DATA_TH = 5ns
   - C_PRHx_RDN_WIDTH = 15ns
   - C_PRHx_RD_CYCLE = 30ns
   - C_PRHx_DATA_TOUT = 5ns
   - C_PRHx_DATA_TINV = 10ns
   - C_PRHx_RDY_TOUT = 10ns
   - C_PRHx_RDY_WIDTH = 50ns

The XPS EPC IP Core resource utilization measured with Virtex-5 as the target device is shown in the following Table 6.

Table 6: **Performance and Resource Utilization Benchmarks on Virtex-5 (Device: xc5vlx85-ff676-1)** (1)

Parameter Values							Device Resources		
C_NUM_PERIPHERALS	C_PRH_CLK_SUPPORT	C_PRHx_AWIDTH	C_PRHx_DWIDTH	C_PRHx_SYNC	C_PRHx_BUS_MULTIPLEX	C_PRHx_DWIDTH_MATCH	Slice Flip- Flops	LUTs	F_{Max} (MHz)
1	0	3	32	0	0	0	259	140	226
1	0	6	16	0	0	1	257	202	205
1	0	6	8	0	0	1	249	200	221
1	0	6	8	0	1	1	268	238	217
1	0	6	16	0	1	1	276	240	203
1	0	6	32	0	1	1	287	186	194
2	0	6,12	32,16	0,0	1,0	0,1	316	292	213
3	0	6,12,6	32,16,8	0,0,0	1,0,1	0,1,1	328	371	198
4	0	6,12,6,3	32,16,8,32	0,0,0,0	1,0,1,0	0,1,1,0	337	350	221
1	0	3	32	1	0	0	239	100	242
1	0	6	16	1	0	1	239	148	248
1	0	6	8	1	0	1	234	146	277
1	0	6	8	1	1	1	238	173	237
1	0	6	16	1	1	1	243	164	219
1	0	6	32	1	1	0	252	113	238
2	0	6,12	32,16	1,1	1,0	0,1	281	225	265
3	0	6,12,6	32,16,8	1,1,1	1,0,1	0,1,1	293	277	196
4	0	6,12,6,3	32,16,8,32	1,1,1,1	1,0,1,0	0,1,1,0	299	295	181

Table 6: **Performance and Resource Utilization Benchmarks on Virtex-5 (Device: xc5vlx85-ff676-1) *(Contd)*** [1]

Parameter Values							Device Resources		
C_NUM_PERIPHERALS	C_PRH_CLK_SUPPORT	C_PRHx_AWIDTH	C_PRHx_DWIDTH	C_PRHx_SYNC	C_PRHx_BUS_MULTIPLEX	C_PRHx_DWIDTH_MATCH	Slice Flip- Flops	LUTs	F_{Max} (MHz)
3	0	12,6,12	32,32,32	0,1,1	0,0,0	0,0,0	353	229	156
3	0	12,12,12	32,32,32	1,0,0	0,0,0	0,0,0	359	204	225

Notes:

1. The bus multiplex timing parameters and asynchronous timing parameters are set to typical values for all devices. The values used for various timing parameters are as follows:
   - C_PRHx_ADDR_TSU = 6ns
   - C_PRHx_ADDR_TH = 6ns
   - C_PRHx_ADS_WIDTH = 10ns
   - C_PRHx_CSN_TSU = 6ns
   - C_PRHx_CSN_TH = 6ns
   - C_PRHx_WRN_WIDTH = 15ns
   - C_PRHx_WR_CYCLE = 30ns
   - C_PRHx_DATA_TSU = 10ns
   - C_PRHx_DATA_TH = 5ns
   - C_PRHx_RDN_WIDTH = 15ns
   - C_PRHx_RD_CYCLE = 30ns
   - C_PRHx_DATA_TOUT = 5ns
   - C_PRHx_DATA_TINV = 10ns
   - C_PRHx_RDY_TOUT = 10ns
   - C_PRHx_RDY_WIDTH = 50ns

The XPS EPC IP Core resource utilization for various parameter combinations measured with Virtex®-6 as the target device are detailed in Table 7.

Table 7: **Performance and Resource Utilization Benchmarks on Virtex-6 (Device: xc6vlx130tff1156-1)**

Parameter Values							Device Resources			
C_NUM_PERIPHERALS	C_PRH_CLK_SUPPORT	C_PRHx_AWIDTH	C_PRHx_DWIDTH	C_PRHx_SYNC	C_PRHx_BUS_MULTIPLEX	C_PRHx_DWIDTH_MATCH	Slices	Slice Flip- Flops	LUTs	F_{Max} (MHz)
1	0	3	32	0	0	0	86	262	229	219
1	0	6	16	0	0	1	99	259	240	265
1	0	6	8	0	0	1	103	252	240	221
1	0	6	8	0	1	1	129	270	291	176
1	0	6	16	0	1	1	126	278	285	182
1	0	6	32	0	1	0	123	291	276	187
2	0	6,12	32,16	0,0	1,0	0,1	138	318	341	184
3	0	6,12,6	32,16,8	0,0,0	1,0,1	0,1,1	176	330	441	200
4	0	6,12,6,3	32,16,8,32	0,0,0,0	1,0,1,0	0,1,1,0	178	337	452	179
1	0	3	32	1	0	0	81	237	167	196
1	0	6	16	1	0	1	81	236	184	206
1	0	6	8	1	0	1	79	231	192	232
1	0	6	8	1	1	1	103	241	230	247
1	0	6	16	1	1	1	98	246	219	216
1	0	6	32	1	1	0	87	254	205	263
2	0	6,12	32,16	1,1	1,0	0,1	116	285	269	232
3	0	6,12,6	32,16,8	1,1,1	1,0,1	0,1,1	145	295	367	178
4	0	6,12,6,3	32,16,8,32	1,1,1,1	1,0,1,0	0,1,1,0	150	300	384	201

Table 7: **Performance and Resource Utilization Benchmarks on Virtex-6 (Device: xc6vlx130tff1156-1) *(Contd)***

Parameter Values							Device Resources			
C_NUM_PERIPHERALS	C_PRH_CLK_SUPPORT	C_PRHx_AWIDTH	C_PRHx_DWIDTH	C_PRHx_SYNC	C_PRHx_BUS_MULTIPLEX	C_PRHx_DWIDTH_MATCH	Slices	Slice Flip- Flops	LUTs	F_{Max} (MHz)
3	0	12,6,12	32,32,32	0,1,1	0,0,0	0,0,0	112	357	300	196
3	0	12,12,12	32,32,32	1,0,0	0,0,0	0,0,0	120	358	303	175

Notes:

1. The bus multiplex timing parameters and asynchronous timing parameters are set to typical values for all devices. The values used for various timing parameters are as follows:
   - C_PRHx_ADDR_TSU = 6ns
   - C_PRHx_ADDR_TH = 6ns
   - C_PRHx_ADS_WIDTH = 10ns
   - C_PRHx_CSN_TSU = 6ns
   - C_PRHx_CSN_TH = 6ns
   - C_PRHx_WRN_WIDTH = 15ns
   - C_PRHx_WR_CYCLE = 30ns
   - C_PRHx_DATA_TSU = 10ns
   - C_PRHx_DATA_TH = 5ns
   - C_PRHx_RDN_WIDTH = 15ns
   - C_PRHx_RD_CYCLE = 30ns
   - C_PRHx_DATA_TOUT = 5ns
   - C_PRHx_DATA_TINV = 10ns
   - C_PRHx_RDY_TOUT = 10ns
   - C_PRHx_RDY_WIDTH = 50ns

The XPS EPC IP Core resource utilization measured with Spartan®-6 as the target device is shown in the following Table 8.

Table 8: **Performance and Resource Utilization Benchmarks on Spartan-6 (Device: xc6slx16csg324-2)** (1)

Parameter Values							Device Resources			
C_NUM_PERIPHERALS	C_PRH_CLK_SUPPORT	C_PRHx_AWIDTH	C_PRHx_DWIDTH	C_PRHx_SYNC	C_PRHx_BUS_MULTIPLEX	C_PRHx_DWIDTH_MATCH	Slices	Slice Flip- Flops	LUTs	F_{Max} (MHz)
1	0	3	32	0	0	0	98	261	218	113
1	0	6	16	0	0	1	109	259	234	111
1	0	6	8	0	0	1	78	251	237	116
1	0	6	8	0	1	0	115	270	275	119
1	0	6	16	0	1	1	103	278	264	124
1	0	6	32	0	1	1	119	289	276	116
2	0	6,12	32,16	0,0	1,0	0,1	126	318	320	102
3	0	6,12,6	32,16,8	0,0,0	1,0,1	0,1,1	161	329	429	101
4	0	6,12,6,3	32,16,8,32	0,0,0,0	1,0,1,0	0,1,1,0	165	338	454	102
1	0	3	32	1	0	0	68	237	164	130
1	0	6	16	1	0	1	85	237	186	131
1	0	6	8	1	0	1	90	231	193	139
1	0	6	8	1	1	1	80	240	226	139
1	0	6	16	1	1	1	92	245	217	129
1	0	6	32	1	1	0	85	254	205	107
2	0	6,12	32,16	1,1	1,0	0,1	117	285	271	121
3	0	6,12,6	32,16,8	1,1,1	1,0,1	0,1,1	143	298	356	100
4	0	6,12,6,3	32,16,8,32	1,1,1,1	1,0,1,0	0,1,1,0	142	304	381	116

Table 8: **Performance and Resource Utilization Benchmarks on Spartan-6 (Device: xc6slx16csg324-2) *(Contd)*** (1)

Parameter Values							Device Resources			
C_NUM_PERIPHERALS	C_PRH_CLK_SUPPORT	C_PRHx_AWIDTH	C_PRHx_DWIDTH	C_PRHx_SYNC	C_PRHx_BUS_MULTIPLEX	C_PRHx_DWIDTH_MATCH	Slices	Slice Flip- Flops	LUTs	F_{Max} (MHz)
3	0	12,6,12	32,32,32	0,1,1	0,0,0	0,0,0	125	353	320	117
3	0	12,12,12	32,32,32	1,0,0	0,0,0	0,0,0	120	354	327	111

Notes:

1. The bus multiplex timing parameters and asynchronous timing parameters are set to typical values for all devices. The values used for various timing parameters are as follows:
   - C_PRHx_ADDR_TSU = 6ns
   - C_PRHx_ADDR_TH = 6ns
   - C_PRHx_ADS_WIDTH = 10ns
   - C_PRHx_CSN_TSU = 6ns
   - C_PRHx_CSN_TH = 6ns
   - C_PRHx_WRN_WIDTH = 15ns
   - C_PRHx_WR_CYCLE = 30ns
   - C_PRHx_DATA_TSU = 10ns
   - C_PRHx_DATA_TH = 5ns
   - C_PRHx_RDN_WIDTH = 15ns
   - C_PRHx_RD_CYCLE = 30ns
   - C_PRHx_DATA_TOUT = 5ns
   - C_PRHx_DATA_TINV = 10ns
   - C_PRHx_RDY_TOUT = 10ns
   - C_PRHx_RDY_WIDTH = 50ns

System Performance

To measure the system performance (Fmax) of this core, this core was added to a Virtex-4 system, a Virtex-5 system, and a Spartan-3A system as the Device Under Test (DUT) as shown in Figure 26, Figure 27, and Figure 28.

Because the XPS EPC IP core will be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When this core is combined with other designs in the system, the utilization of FPGA resources and timing of the core design will vary from the results reported here.

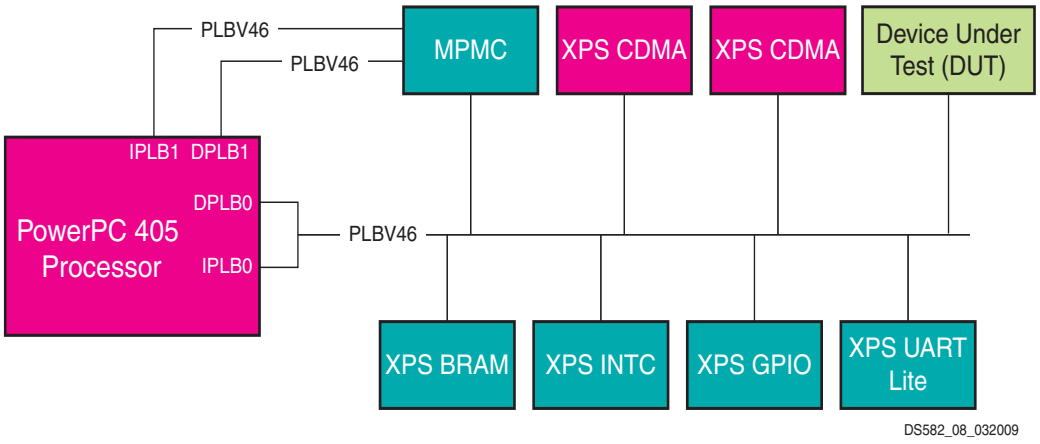

Figure 26: **Virtex-4 FX System**

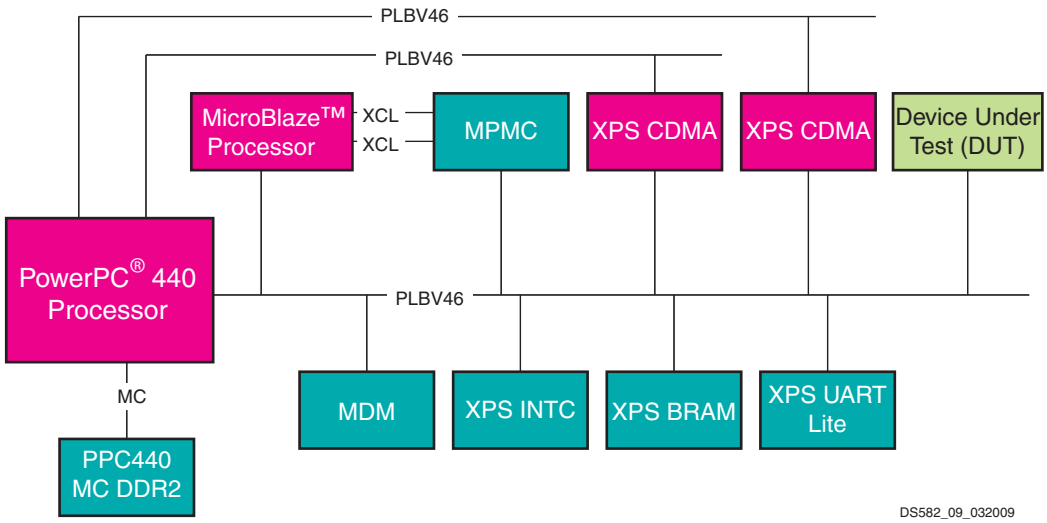

Figure 27: **Virtex-5 FX System**

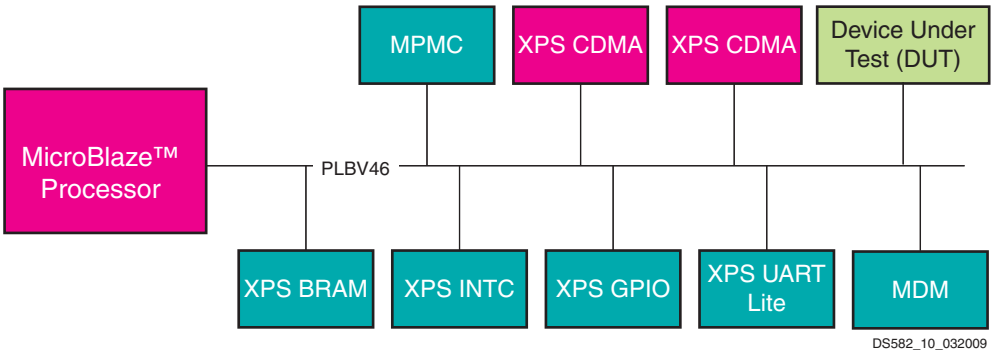

Figure 28: **Spartan-3A System**

The target FPGA was then filled with logic to drive the LUT and BRAM utilization to approximately 70% and the I/O utilization to approximately 80%. Using the default tool options and the slowest speed grade for the target FPGA, the resulting target F_{MAX} numbers are shown in Table 9.

Table 9: **XPS EPC System Performance**

Target FPGA	Target F_{MAX} (MHz)
S3A700 -4	90
V4FX60 -10	100
V5LXT50 -1	120

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

Specification Exceptions

N/A

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Reference Documents

The following documents contain information that may be required in understanding the XPS EPC IP Core reference design:

1. DS583 XPS SYSACE (System ACE) Interface Controller
2. DS561 PLBv46 Slave Single
3. 10/100 Non-PCI Ethernet Single Chip MAC + PHY LAN91C111 Data Sheet, SMSC
4. EZ-Host(TM) Programmable Embedded USB Host/Peripheral Controller CY7C67300 Data Sheet, Cypress Semiconductor
5. IBM CoreConnect 128-Bit Processor Local Bus: Architecture Specifications version 4.6

Notice of Disclaimer

Revision History

Date	Version	Revision
4/4/07	1.0	Initial Xilinx release.
4/20/07	1.1	Removed SP3-AF and added SP-3 support.
9/28/07	1.2	Added FMax Margin System Performance section.
11/27/07	1.3	Added SP-3A DSP support.
1/14/08	1.4	Added Virtex-II Pro support.
4/18/08	1.5	Added Automotive Spartan-3E, Automotive Spartan-3, Automotive Spartan-3A, and Automotive Spartan-3A DSP support .
7/03/08	1.6	Added figure 23 and 24
7/22/08	1.7	Added QPro Virtex-4 Hi-Rel and QPro Virtex-4 Rad Tolerant FPGA support.
8/18/08	1.8	Created new version of the core and updated for IPIF libraries. Removed Virtex II Pro support.
4/24/09	1.9	Replaced references to supported device families and tool name(s) with hyperlink to PDF file.
07/17/09	2.0	Updated core version. Updated the utilization matrix for S6, V6 devices.