AXI4-Lite IPIF v3.0

LogiCORE IP Product Guide

Vivado Design Suite

PG155 April 6, 2016

XILINX
ALL PROGRAMMABLE™

Table of Contents

Appendix B: Debugging

Appendix C: Additional Resources

Introduction

The LogiCORE™ IP AXI4-Lite IP Interface (IPIF) is a part of the Xilinx family of ARM® AMBA® AXI control interface compatible products. It provides a point-to-point bidirectional interface between a user IP core and the Xilinx LogiCORE IP AXI Interconnect core. This version of the AXI4-Lite IPIF has been optimized for slave operation on the AXI interface. It does not provide support for Direct Memory Access (DMA) and IP Master Services.

Features

- Supports 32-bit slave configuration
- Supports read and write data transfers of 32-bit width
- Supports multiple address ranges
- Read has higher priority over write
- Reads from holes in the address space return 0x00000000
- Writes to holes in the address space after the register map are ignored and receive an OKAY response
- Both AXI and IP Interconnect (IPIC) are little endian
- Provides word-aligned addresses at the IPIC interface

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family	UltraScale™ Architecture, Zynq®-7000 All Programmable SoC, 7 Series FPGAs
Supported User Interfaces	AXI4-Lite
Resources	See Table 2-2 through Table 2-4
Provided with Core	
Design Files	VHDL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	Not Provided
Supported S/W Driver	Not Applicable
Tested Design Flows	
Design Entry	Vivado® Design Suite(1)(2)
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. AXI4-Lite IPIF is a helper core, not available in the IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The AXI4-Lite IPIF core is designed to provide a quickly implemented, light-weight interface between the ARM® AXI interconnect and a user IP core. This slave service allows you to configure multiple IP cores interfaced to the AXI Interconnect core by providing address decoding over various address ranges. Figure 1-1 shows a block diagram of the AXI4-Lite IPIF.

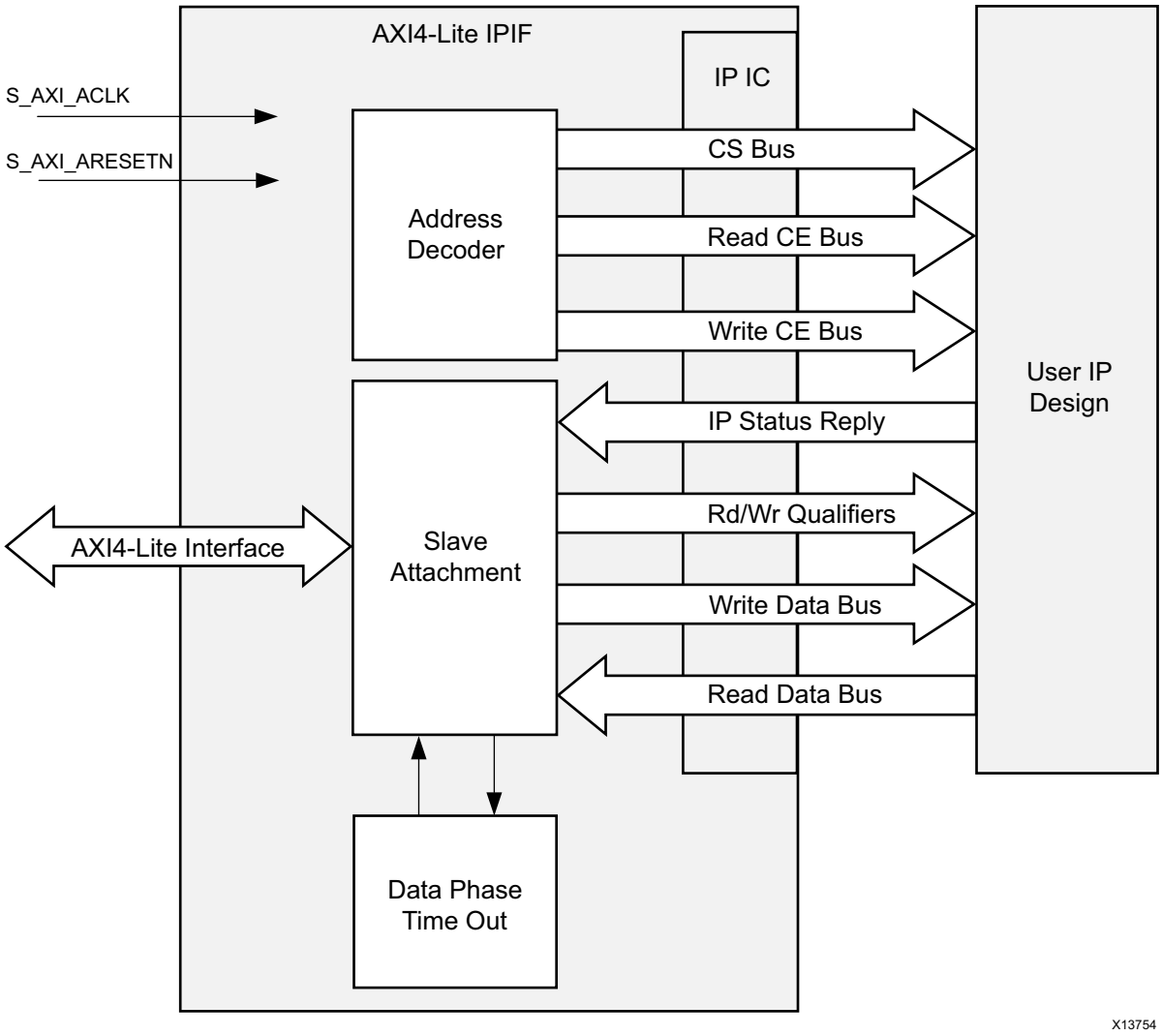

Figure 1-1: **AXI4-Lite IPIF Block Diagram**

The slave attachment is the base element of the design. This block provides the basic functionality for slave operation implementing the protocol and timing translation between the AXI Interconnect and the IP interconnect (IPIC).

The address decoder module generates the necessary chip select and read/write chip enable signals based on the user requirement. The timeout counter is added into the design if the C_DPHASE_TIMEOUT parameter is non-zero. If C_DPHASE_TIMEOUT = 0, make sure that the core generates the acknowledge signals for all transactions.

Feature Summary

- IPIC interface generates a word boundary address
- Byte strobes are replica of Write strobes from AXI4 interface during write transactions
- The number of `Bus2IP_RdCE` and `Bu2IP_WrCE` signals are dependent upon the user input for the particular address range
- One chip select signal per address range
- `Bus2IP_Resetn` is active-Low and propagates the AXI4 interface reset signal

Applications

Application details are relevant to the slave core which instantiates the AXI4-Lite IPIF core.

Unsupported Features

- Protection unit support is limited, AxPROT signals are ignored.
- Low-power interface is not implemented.
- AXI data bus and address bus widths are fixed to 32 bits.
- Multiple outstanding transactions are not supported.
- If there is a simultaneous read/write on the AXI interface, the read has higher priority over write.
- Reads from the holes in the address space return 0x00000000 and an OKAY response.
- Writes to the holes in the address space after the register map are ignored with an OKAY response.
- AXI4-Lite IPIF does not perform endian conversion. Both AXI and IPIC are little-endian.

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 about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Standards

The AXI4-Lite IPIF core conform to the Advanced Microcontroller Bus Architecture (AMBA®) AXI version 4 specification from Advanced RISC Machine (ARM®), including the AXI4-Lite control register interface subset. See *ARM AMBA AXI Protocol v2.0* [Ref 1].

Performance

This section details the performance information for various core configurations.

The AXI4-Lite IPIF core operates maximum frequencies for 7 series devices are shown in Table 2-1. Zynq®-7000 devices are expected to have frequency characteristics similar to 7 series FPGAs.

Table 2-1: **AXI4-Lite IPIF Maximum Frequencies**

Family	F_{MAX} (MHz)
Virtex®-7	180
Kintex®-7	150
Artix®-7	120

Resource Utilization

The performance and resource utilization numbers for the AXI4-Lite IPIF core are generated from a common benchmarking setup implementing a multi-masters/multi-slave AXI4 interface system. For more details on this setup, see the "Vivado IP Optimization (Fmax Characterization)" appendix in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Because the AXI4-Lite IPIF is used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates. As the AXI4-Lite IPIF is combined with other design elements, the utilization of FPGA resources and timing varies from the results reported here.

The resource utilization of this version of the AXI4-Lite IPIF is shown here for some example configurations. The slave attachment was synthesized using Vivado synthesis. The resource utilization report was then used as the source data for the table. The AXI4-Lite IPIF benchmarks for Zynq-7000 devices are expected to be similar to results for 7 series FPGAs.

Note: UltraScale™ architecture results are expected to be similar to 7 series device results.

The AXI4-Lite IPIF benchmarks are shown in Table 2-2 for a Virtex-7 (XC7V285TFFG784-3) FPGA.

Table 2-2: **AXI4-Lite IPIF FPGA Performance and Resource Utilization for Virtex-7 FPGA**

Parameter Values				Device Resources for Virtex-7		
C_ARD_ADDR_RANGE_ARRAY Pairs	C_ARD_NUM_CE_ARRAY	C_DPHASE_TIMEOUT	C_USE_WSTRB	Slices	Flip-Flops	LUTs
2	4, 8	8	0	31	49	54
2	4, 8	8	1	32	49	56
4	4, 8, 16, 8	512	0	49	59	97
4	4, 8, 16, 8	512	1	47	59	99
4	4, 8, 16, 8	0	0	43	54	92
4	4, 8, 16, 8	0	1	48	58	98

Notes:

1. For the utilization calculation, the parameter C_S_AXI_MIN_SIZE = X"000001FF" is used.

The AXI4-Lite IPIF benchmarks are shown in Table 2-3 for a Kintex-7 (XC7K410TFFG676-3) FPGA.

Table 2-3: **AXI4-Lite IPIF FPGA Performance and Resource Utilization for Kintex-7 FPGA**

Parameter Values				Device Resources for Kintex-7		
C_ARD_ADDR_RANGE_ARRAY Pairs	C_ARD_NUM_CE_ARRAY	C_DPHASE_TIMEOUT	C_USE_WSTRB	Slices	Flip-Flops	LUTs
2	4, 8	8	0	26	49	54
2	4, 8	8	1	30	49	56
4	4, 8, 16, 8	512	0	44	59	97
4	4, 8, 16, 8	512	1	50	59	99
4	4, 8, 16, 8	0	0	42	54	92
4	4, 8, 16, 8	0	1	49	58	98

Notes:

1. For the utilization calculation, the parameter C_S_AXI_MIN_SIZE = X"000001FF" is used.

The AXI4-Lite IPIF benchmarks are shown in Table 2-4 for a Artix-7 (XC7A350TFBG676-3) FPGA.

Table 2-4: **AXI4-Lite IPIF FPGA Performance and Resource Utilization for Artix-7 FPGA**

Parameter Values				Device Resources for Artix-7		
C_ARD_ADDR_RANGE_ARRAY Pairs	C_ARD_NUM_CE_ARRAY	C_DPHASE_TIMEOUT	C_USE_WSTRB	Slices	Flip-Flops	LUTs
2	4, 8	8	0	28	49	30
2	4, 8	8	1	32	49	32
4	4, 8, 16, 8	512	0	45	59	66
4	4, 8, 16, 8	512	1	44	59	68
4	4, 8, 16, 8	0	0	39	56	77
4	4, 8, 16, 8	0	1	45	58	67

Notes:

1. For the utilization calculation, the parameter C_S_AXI_MIN_SIZE = X"000001FF" is used.

Port Descriptions

The AXI4-Lite IPIF signals are listed and described in Table 2-5.

Table 2-5: **AXI4-Lite IPIF Port Descriptions**

Port	Signal Name	Interface	I/O	Initial State	Description
AXI Global System Signals[(1)(2)]					
P1	S_AXI_ACLK	AXI	I	-	AXI Clock.
P2	S_AXI_ARESETN	AXI	I	-	AXI Reset, active-Low.
AXI Write Address Channel Signals[(1)(2)]					
P3	S_AXI_AWADDR[C_S_AXI_ADDR_WIDTH–1:0]	AXI	I	-	AXI Write address. The write address bus gives the address of the write transaction.
P4	S_AXI_AWVALID	AXI	I	-	Write address valid. This signal indicates that valid write address and control information are available.
P5	S_AXI_AWREADY	AXI	O	0	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
AXI Write Data Channel Signals[(1)(2)]					
P6	S_AXI_WDATA[C_S_AXI_DATA_WIDTH–1:0]	AXI	I	-	Write data.
P7	S_AXI_WSTRB[C_S_AXI_DATA_WIDTH/8–1:0]	AXI	I	-	Write strobes. This signal indicates which byte lanes to update in memory.
P8	S_AXI_WVALID	AXI	I	-	Write valid. This signal indicates that valid write data and strobes are available.
P9	S_AXI_WREADY	AXI	O	0	Write ready. This signal indicates that the slave can accept the write data.
AXI Write Response Channel Signals[(1)(2)]					
P10	S_AXI_BRESP[1:0][(5)]	AXI	O	0	Write response. This signal indicates the status of the write transaction: “00” = OKAY “10” = SLVERR
P11	S_AXI_BVALID	AXI	O	0	Write response valid. This signal indicates that a valid write response is available.

Table 2-5: **AXI4-Lite IPIF Port Descriptions *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P12	S_AXI_BREADY	AXI	I	-	Response ready. This signal indicates that the master can accept the response information.
AXI Read Address Channel Signals[(1)(2)]					
P13	S_AXI_ARADDR[C_S_AXI_ADDR_WIDTH–1:0]	AXI	I	-	Read address. The read address bus gives the address of a read transaction.
P14	S_AXI_ARVALID[(3)]	AXI	I	-	Read address valid. When High, this signal indicates that the read address and control information is valid and remains stable until the address acknowledgement signal, S_AXI_ARREADY, is High.
P15	S_AXI_ARREADY	AXI	O	0	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
AXI Read Data Channel Signals[(1)(2)]					
P16	S_AXI_RDATA[C_S_AXI_DATA_WIDTH–1:0]	AXI	O	0	Read data.
P17	S_AXI_RRESP[1:0][(5)]	AXI	O	0	Read response. This signal indicates the status of the read transfer.
P18	S_AXI_RVALID	AXI	O	0	Read valid. This signal indicates that the required read data is available and the read transfer can complete.
P19	S_AXI_RREADY	AXI	I	-	Read ready. This signal indicates that the master can accept the read data and response information.
User IP Signals					
P23	Bus2IP_Clk	User IP	O	0	Synchronization clock provided to user IP. This is the same as S_AXI_ACLK.
P24	Bus2IP_Resetn	User IP	O	0	Active-Low reset for use by the user IP.
P25	IP2Bus_Data[C_S_AXI_DATA_WIDTH–1:0]	User IP	I	-	Input read data bus from the user IP. Data is qualified with the assertion of IP2Bus_RdAck signal and the rising edge of the Bus2IP_Clk.
P26	IP2Bus_WrAck	User IP	I	-	Active-High write data qualifier. Write data on the Bus2IP_Data bus is deemed accepted by the user IP at the rising edge of the Bus2IP_Clk and IP2Bus_WrAck asserted High by the user IP.

Table 2-5: **AXI4-Lite IPIF Port Descriptions *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P27	IP2Bus_RdAck	User IP	I	-	Active-High read data qualifier. Read data on the `IP2Bus_Data` bus is deemed valid at the rising edge of `Bus2IP_Clk` and the assertion of the `IP2Bus_RdAck` signal by the user IP.
P28	IP2Bus_Error	User IP	I	-	Active-High signal indicating the user IP has encountered an error with the requested operation. This signal is asserted in conjunction with `IP2Bus_RdAck` or the `IP2Bus_WrAck`.
P29	Bus2IP_Addr[C_S_AXI_ADDR_ WIDTH–1:0]	User IP	O	0	Address bus indicating the desired address of the requested read or write operation.
P30	Bus2IP_Data[C_S_AXI_DATA_ WIDTH–1:0]	User IP	O	0	Write data bus to the user IP. Write data is accepted by the IP during a write operation by assertion of the `IP2Bus_WrAck` signal and the rising edge of `Bus2IP_Clk`.
P31	Bus2IP_RNW	User IP	O	0	This signal indicates the sense of a requested operation with the user IP. High is a read; Low is a write.
P32	Bus2IP_BE[(C_S_AXI_DATA_ WIDTH/8)-1]:0]	User IP	O	0	Byte enable qualifiers for the requested read or write operation with the user IP. Bit 0 corresponds to byte lane 0, bit 1 to byte lane 1, and so on.
P33	Bus2IP_CS[(C_ARD_ADDR_RANGE_ ARRAY'length/2) –1:0]	User IP	O	0	Active-High chip select bus. Each bit of the bus corresponds to an address pair entry in the C_ARD_ADDR_RANGE_ARRAY. Assertion of a chip select indicates an active transaction request to the chip select target address space.
P34	Bus2IP_RdCE[(4):0]	User IP	O	0	Active-High chip enable bus. Chip enables are assigned per the user entries in the C_ARD_NUM_CE_ARRAY. These chip enables are asserted only during active read transaction requests with the target address space and in conjunction with the corresponding sub-address within the space.

Table 2-5: **AXI4-Lite IPIF Port Descriptions *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P35	Bus2IP_WrCE[(4):0]	User IP	O	0	Active-High chip enable bus. Chip enables are assigned per the user entries in the C_ARD_NUM_CE_ARRAY. These chip enables are asserted only during active write transaction requests with the target address space and in conjunction with the corresponding sub-address within the space.

Notes:

1. For a description of AXI4, AXI4-Lite and AXI Stream signals, see Appendix A of the *AXI Reference Guide* (UG1037) [Ref 3].
2. The function and timing of these signals is defined in the *AMBA AXI Protocol Version: 2.0* specification [Ref 1].
3. Read transactions have higher priority than write transactions.
4. The size of the `Bus2IP_RdCE` and the `Bus2IP_WrCE` buses is the sum of the integer values entered in the C_ARD_NUM_CE_ARRAY.
5. For signals, such as `S_AXI_RRESP[1:0]` and `S_AXI_BRESP[1:0]`, the IP core does not generate the decode error ('11') response. Other responses such as '00' (OKAY) and '10' (SLVERR) are generated by the core based on certain conditions.

IP Interconnect Port Detailed Descriptions

All of the IP Interconnect (IPIC) signals are in little-endian format in line with the AXI interconnect specification.

- **Bus2IP_Addr**: A 32-bit vector that is valid when `Bus2IP_CS` and `Bus2IP_RdCE`, or `Bus2IP_WrCE` is High.
- **Bus2IP_Data**: A vector of width C_S_AXI_DATA_WIDTH and is valid on writes when `Bus2IP_WrCE` is High.
- **Bus2IP_RNW**: Indicates the type of transfer in progress and is valid when `Bus2IP_CS` and `Bus2IP_WrCE`, or `Bus2IP_RdCE` is asserted. A High level indicates the transfer request is a user IP read. A Low level indicates the transfer request is a user IP write.
- **Bus2IP_BE**: A vector of width C_S_AXI_DATA_WIDTH/8. This vector indicates which bytes are valid on `Bus2IP_Data`. Becomes valid coincident with `Bus2IP_CS`. Because all of the AXI transactions are 32-bit wide, the `Bus2IP_BE` vector is tied to `1111`.
- **Bus2IP_CS**: A vector of width C_ARD_ADDR_RANGE_ARRAY length / 2. For each address pair defined in C_ARD_ADDR_RANGE_ARRAY there is one `Bus2IP_CS` defined. Asserts at the beginning of a valid cycle on the IPIC. This signal, used in conjunction with `Bus2IP_RNW`, is especially suited for reading and writing memory devices.
- **Bus2IP_RdCE**: A vector of a width that is the sum of the values defined in C_ARD_NUM_CE_ARRAY. For each address pair defined in C_ARD_ADDR_RANGE_ARRAY, a group of associated `CE` signals can be defined in C_ARD_NUM_CE_ARRAY.

Bus2IP_RdCE goes High coincident with Bus2IP_CS for read transfers and is especially suited for reading registers.

- **Bus2IP_WrCE**: A vector of a width that is the sum of the values defined in C_ARD_NUM_CE_ARRAY. For each address pair defined in C_ARD_ADDR_RANGE_ARRAY, a group of associated CE signals can be defined in C_ARD_NUM_CE_ARRAY. Bus2IP_WrCE goes High when the write data is valid on Bus2IP_WrCE and is especially suited for writing to registers.
- **IP2Bus_Data**: The read data bus vector of width C_S_AXI_DATA_WIDTH. Read data should be valid when IP2Bus_RdAck is asserted by the user IP.
- **IP2Bus_RdAck**: The read data acknowledge signal. Acknowledges an IP read cycle and causes read control signals Bus2IP_RdCE, Bus2IP_CS and Bus2IP_RNW to deassert.
- **IP2Bus_WrAck**: The write data acknowledge signal. Acknowledges an IP write cycle and causes write control signals Bus2IP_WrCE and Bus2IP_CS to deassert.
- **IP2Bus_Error**: Indicates an IP error has occurred. This signal is only sampled with IP2Bus_WrAck or IP2Bus_RdAck and is otherwise ignored. Sent to the interconnect as a slave error.

Designing with the Core

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

General Design Guidelines

AXI4-Lite IPIF Core Usage

If the C_S_AXI_MIN_SIZE = `0x1FF`, the `S_AXI_AWADDR[8:0]` is used for address decoding. The width of the `Bus2IP_CS` is 2 and the width of the `Bu2IP_WrCE` and `Bus2IP_RdCE` is 20. C_DPHASE_TIMEOUT = 16 is considered.

A chip select/chip enable example using C_ARD_ADDR_RANGE_ARRAY and C_ARD_NUM_CE_ARRAY is shown in Figure 3-1.

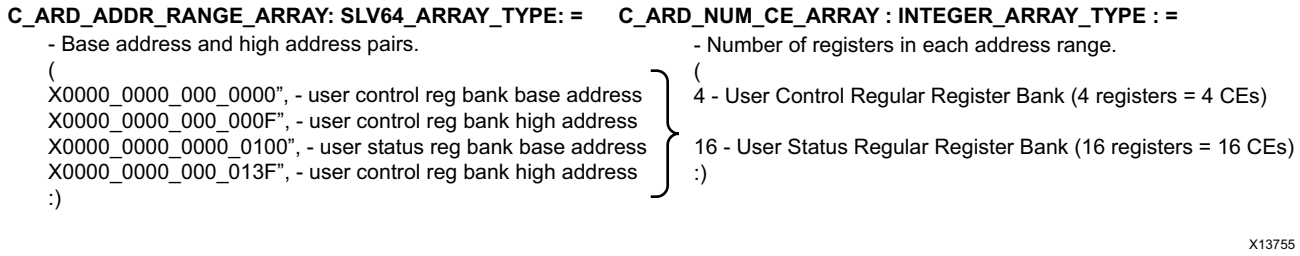

Figure 3-1: **Chip Select/Chip Enable Example**

The example shown in Figure 3-1 applies to these cases:

Case 1

If there is a write transaction for address `0x00000000`, the AXI4-Lite IPIF asserts `Bus2IP_CS(0)` and `Bu2IP_WrCE(19)` and waits for `IP2Bus_WrAck` from the IP.

If there is a read transaction for address `0x00000000`, the AXI4-Lite IPIF asserts `Bus2IP_CS(0)` and `Bus2IP_RdCE(19)` and waits for `IP2Bus_RdAck` from the IP.

Case 2

If there is a write transaction for address `0x000000F0`, the AXI4-Lite IPIF does not generate `Bus2IP_CS` and `Bu2IP_WrCE` because this address falls between the two address ranges. The IPIF sends the OKAY response to the AXI Interconnect.

If there is a read transaction for address `0x000000F0`, the AXI4-Lite IPIF does not generate `Bus2IP_CS` and `Bus2IP_RdCE` because this address falls between the two address ranges. The IPIF sends the OKAY response to the AXI Interconnect.

In both preceding use cases, take care to set C_DPHASE_TIMEOUT /=0. This non-zero value should be greater than the number of cycles taken for normal register access.

Case 3

If there is a write transaction for address `0x00000100`, the AXI4-Lite IPIF asserts `Bus2IP_CS(0)` and `Bu2IP_WrCE(15)` and waits for `IP2Bus_WrAck` from the IP.

If there is a read transaction for address `0x00000100`, the AXI4-Lite IPIF asserts `Bus2IP_CS(0)` and `Bus2IP_RdCE(15)` and waits for `IP2Bus_RdAck` from the IP.

Case 4

If there is a write transaction for address `0x00000140`, the AXI4-Lite IPIF does not generate `Bus2IP_CS` or `Bu2IP_WrCE` because this address falls after the high address of the second address range. The AXI4-Lite IPIF sends the OKAY response to the AXI Interconnect.

If there is a read transaction for address `0x00000140`, the AXI4-Lite IPIF does not generate `Bus2IP_CS` or `Bus2IP_RdCE` because this address falls after the high address of the second address range. The AXI4-Lite IPIF sends the OKAY response to the AXI Interconnect.

In both preceding use cases, take care to set C_DPHASE_TIMEOUT /=0. This non-zero value should be greater than the number of cycles taken for normal register access.

Case 5

If there is a write transaction for address `0x00000200`, the AXI4-Lite IPIF asserts `Bus2IP_CS(0)` and `Bu2IP_WrCE(19)` and waits for `IP2Bus_WrAck` from the IP. This is because address wrapping occurs after address `0x00000200`.

If there is a read transaction for address `0x00000200`, the AXI4-Lite IPIF asserts `Bus2IP_CS(0)` and `Bus2IP_RdCE(19)` and waits for `IP2Bus_RdAck` from the IP. This is because address wrapping occurs after address `0x00000200`.

IP Interconnect Chip Selects and Chip Enables

Implementing chip select (CS) and chip enable (CE) signals is a common design task needed within microprocessor-based systems to qualify the selection of registers, ports, and memory through an address decoding function. The AXI4-Lite IPIF implements a flexible technique for providing these signals through the ARD parameters. It is necessary to understand the relationship between the population of the ARD array parameters and the Bus2IP_CS, the Bus2IP_RdCE, and the Bus2IP_WrCE buses available at the IP Interconnect interface with the slave attachment. An example of ARD array population and the resulting CS and CE bus generation is shown in Figure 3-2. The selection of the signal set for IP functions is up to you and the design requirements. Unused CE and CS signals and associated generation logic are trimmed during the synthesis and place and route (PAR) phases of FPGA development.

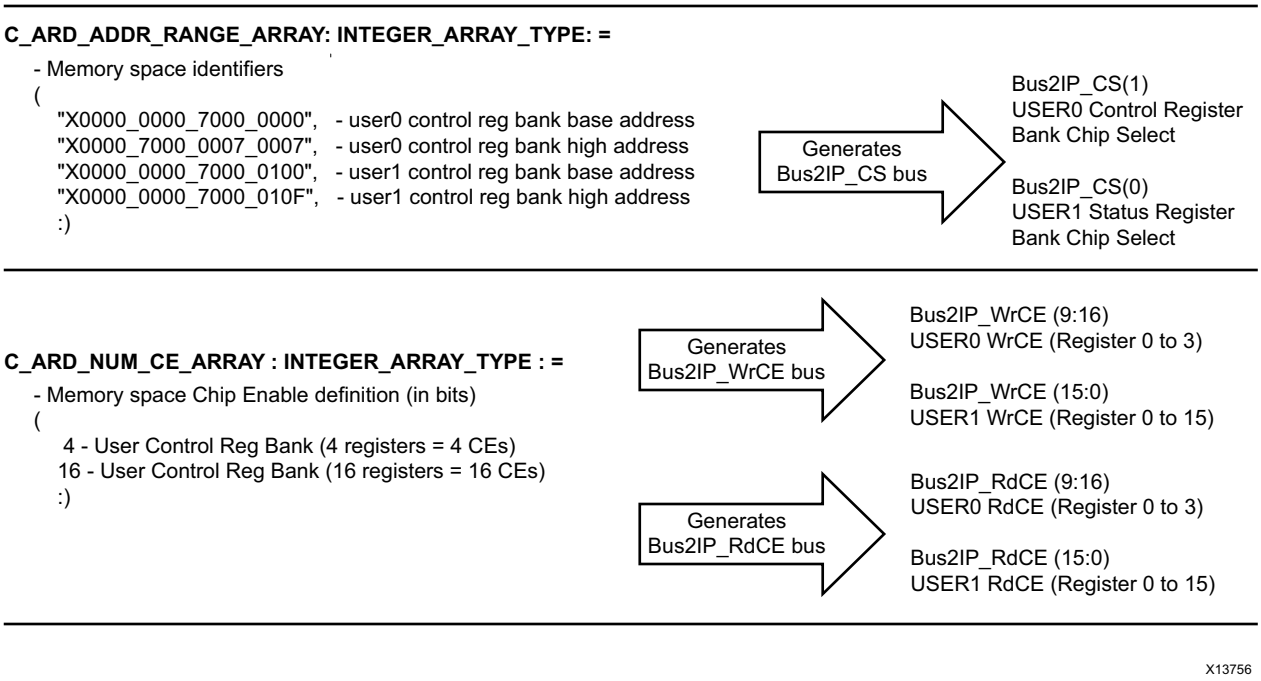

Figure 3-2: **ARD Arrays and CS/CE Relationship Example**

Chip Select Bus (Bus2IP_CS(n:0))

A single chip select signal is assigned to each address space defined by the designer in the ARD arrays. The chip select is asserted (active-High) whenever a valid access (read or write) is requested from the valid address space and has been address acknowledged. It remains asserted until the data phase of the transfer between the slave attachment and the addressed target has completed. The Bus2IP_CS port is provided as part of the IPIC signal set. The Bus2IP_CS bus has a one-to-one correlation to the number and ordering of address pairs in the C_ARD_ADDR_RANGE_ARRAY parameter. For example, if the C_ARD_ADDR_RANGE_ARRAY has 10 entries, the Bus2IP_CS bus is sized as 0 to 4.

Bus2IP_CS(0) corresponds to the first address space, Bus2IP_CS(1) to the second address space, and so on. The nature of the chip select bus requires the user IP to provide any additional address discrimination within the address space as well as qualification with the Bus2IP_RNW signal.

Read Chip Enable Bus (Bus2IP_RdCE(y:0))

Bus2IP_RdCE is the chip enable bus for read transactions. Each address space defined in the ARD arrays is allowed to have one or more chip enable signals assigned to it. Chip enables are used for subdividing an address space into smaller spaces that are each less than or equal to the AXI interconnect width. Generally, this subdivision is useful for selecting registers and ports during read or write transactions. The slave attachment accomplishes this subdivision through parameters entered in the C_ARD_NUM_CE_ARRAY. For each defined address space, enter the number of desired chip enable signals to be generated for each space. The core implementation requires a value of at least 1 for each space. The data width of the space, set at 32-bits, determines the size of the address slice assigned to each CE signal for the address space. Bus2IP_RdCE asserts if the request transaction is a read.

Write Chip Enable Bus (Bus2IP_WrCE(y:0))

The Bus2IP_WrCE bus is the same size as the Bus2IP_RdCE bus except that the Bus2IP_WrCE signals are only asserted if the requested transaction is a write.

Available Support Functions for Automatic Separation of CE and CS Buses

It might be convenient to use some predefined functions developed by Xilinx to automatically separate signals from the Bus2IP_CS, Bus2IP_WrCE, and Bus2IP_RdCE buses. These functions facilitate bus separation regardless of the order or composition of user functions in the ARD Arrays. This is extremely useful if user parameterization adds or removes user IP functions (which changes the size and ordering of the CS and CE buses). Table 3-1 lists and details these functions which are declared and defined in the ipif_pkg.vhd source file located in the Vivado IP catalog at this path:

```
\data\ip\xilinx\proc_common_v4_0\hdl\src\vhdl\ipif_pkg.vhd
```

Ensure that the library declaration appears in your VHDL source:

```
library proc_common_v4_0;
use proc_common_v4_0.ipif_pkg.all;
```

A usage example of these functions is shown in Figure 3-3.

Table 3-1: **Slave Attachment Support VHDL Functions**

VHDL Function Name	Input Parameter Name	Input Parameter Type	Return Type	Description
calc_num_ce			Integer	This function is used to get the total number of signals that make up each of the Bus2IP_RdCE and the Bus2IP_WrCE buses (they are all the same size and order). The information is derived from the ce_num_array parameter. Example: constant CE_BUS_SIZE : integer := calc_num_ce(C_ARD_NUM_CE_ARRAY);
	ce_num_array	INTEGER_ ARRAY_TYPE		
calc_start_ce_index			Integer	This function is used to get the starting index of the CE or range of CEs to separate from the Bus2IP_RdCE and the Bus2IP_WrCE buses relating to the index value of the address space entry in the ARD Arrays. The information is derived from the ce_num_ce_array parameter and an index of the address range of interest. Example: To find start ce index for the third address pair, pass 2 into the calc_start_ce_index function. constant USER0_START_CE_INDEX : integer := calc_start_ce_index(C_ARD_NUM_CE_ ARRAY, 2);
	ce_num_array	INTEGER_ ARRAY_TYPE		
	index	integer		

```
architecture USER_ARCH of user_top_level;

    - Extract the number of CEs assigned to the second address pair
      Constant NUM_USER_01_CE : integer: = C_ARD_NUM_CE_ARRAY(1)
    - Extract the number of CE indexes to use for the second Register CE
      Constant NUM_USER_01_CE : integer: = C_ARD_NUM_CE_ARRAY(1)
    Constant USER_01_END_CE_INDEX: integer: = USER_01_CE_INDEX + NUM_USER_01_CE - 1;

    - Declare signals
      signal user_01_cs: std_logic;
      signal user_01_rdce : std_logic_vector(NUM_USER_01_CE-1 down to 0);
      signal user_01_wrce : std_logic_vector(NUM_USER_01_CE-1 down to 0);

begin (architecture)

    - Now rip the buses and connect
      user_01_cs        <= Bus2IP_CS(1)
      user_01_rdce      <= Bus2IP_RdCE(USER_01_END_CE_INDEX down to USER_01_START_CE_INDEX);
      user_01_wrce      <= Bus2IP_WrCE(USER_01_END_CE_INDEX down to USER_01_START_CE_INDEX);
```

X13757

Figure 3-3: **Bus Separation Example**

Single Entry in Unconstrained Array Parameters

Synthesis tools sometimes have problems with positional association of VHDL unconstrained arrays that have only one entry. To avoid this issue, use named association for the single array entry. For example:

```
C_ARD_NUM_CE_ARRAY(16); -- VHDL positional association - can cause synthesis type
                           conflict error for single entry!
C_ARD_NUM_CE_ARRAY(0 => 16); -- VHDL named association....avoids type conflict error
                                for single entry.
```

Clocking

The AXI4-Lite IPIF core is driven by the same clock provided by the AXI4 interconnect.

Resets

The AXI4-Lite IPIF core receives the reset signal driven by either interconnect or it can be assigned to the output of the dedicated reset core. The AXI4-Lite IPIF core operates on an active-Low reset and produces an active-Low reset at the IPIC interface.

Protocol Description

This section shows timing relationships for AXI and slave attachment interface signals during various read and write accesses.

Single Read Operation

A single read operation timing diagram is shown in Figure 3-4.

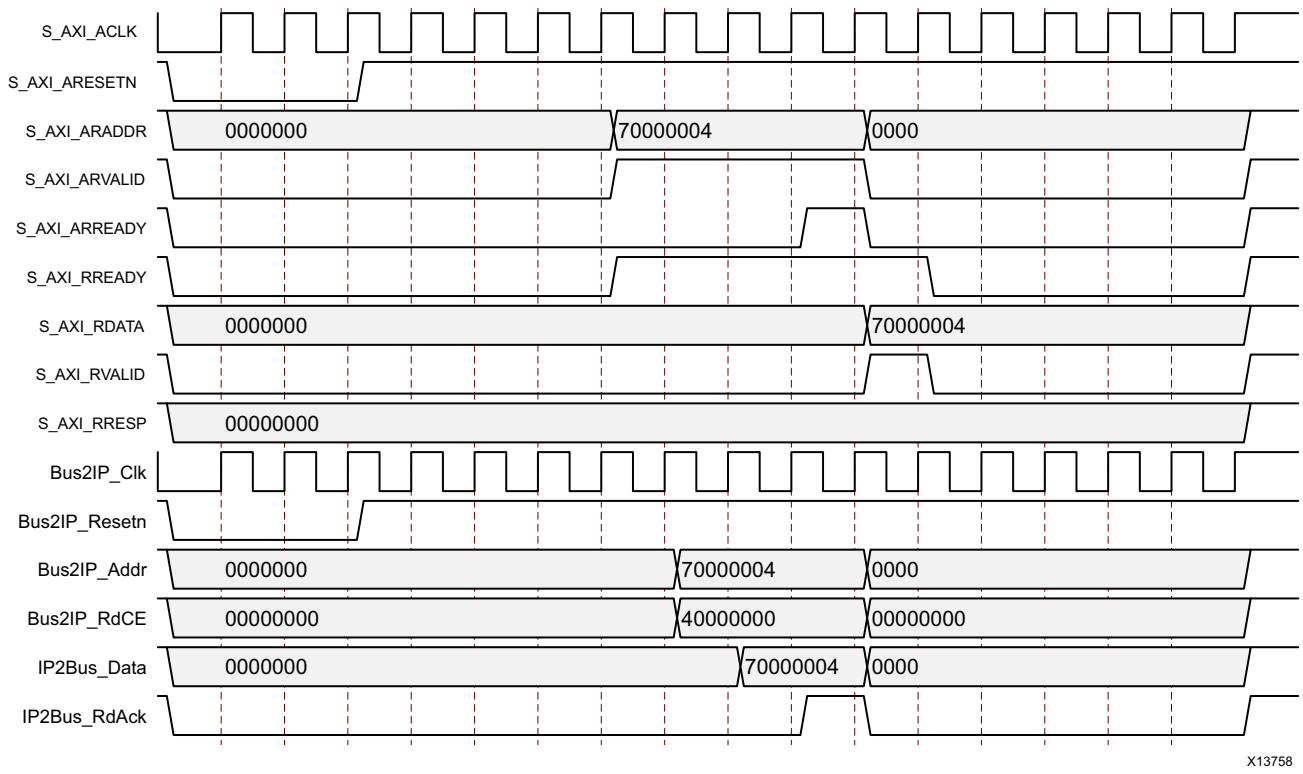

Figure 3-4: **AXI4-Lite IPIF Single Read Operation**

Single Write Operation

A single write operation timing diagram is shown in Figure 3-5.

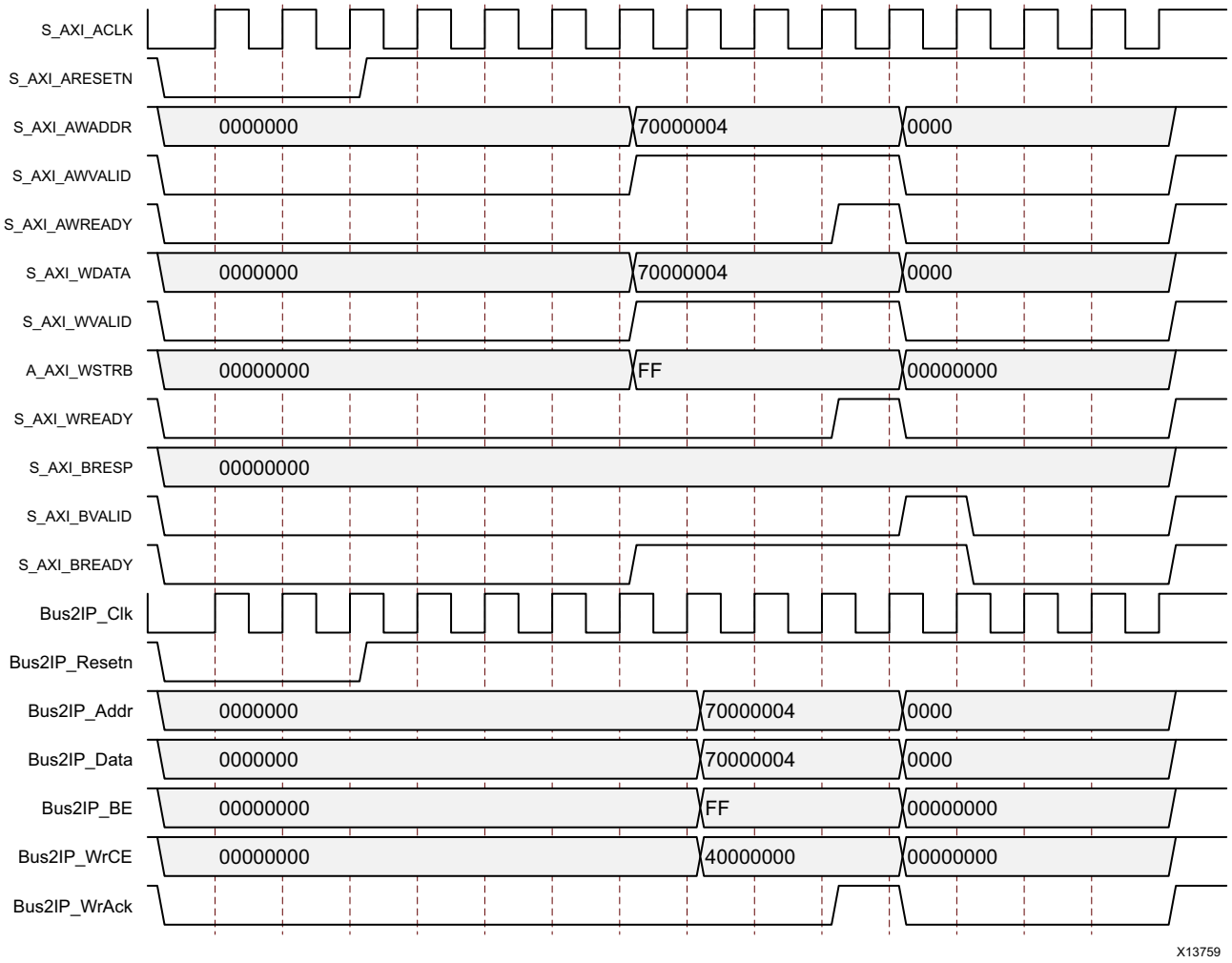

Figure 3-5: **AXI4-Lite IPIF Single Write Operation**

Design Flow Steps

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

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 1]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4]

Customizing and Generating the Core

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

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

You can customize the 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 Vivado IP catalog.
2. Double-click the selected IP or select the **Customize IP** command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3].

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

Design Parameters

To create a uniquely tailored AXI4-Lite IPIF core, certain features can be parameterized in the core design. This allows you to create a design that only utilizes the resources required by the system and provides optimal performance. The AXI4-Lite IPIF design parameters are shown in Table 4-1.

In addition to the parameters listed in this table, there are also parameters that are inferred for each AXI interface in the Vivado Design Suite. Through the design, these inferred parameters control the behavior of the AXI Interconnect. For a complete list of the interconnect settings related to the AXI interface, see the *AXI Interconnect LogiCORE IP Product Guide* (PG059) [Ref 4].

Table 4-1: **Design Parameters**

Generic	Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
System Parameters					
G1	Target FPGA family	C_FAMILY	zynq7000, virtex7, kintex7, artix7	virtex7	string
G2	Use the write strobes	C_USE_WSTRB(1)	0 to 1	0	integer
G3	Data phase time out	C_DPHASE_TIMEOUT	0 to 512	8	integer
AXI Parameters					
G4	AXI address bus width	C_S_AXI_ADDR_WIDTH	32	32	integer
G5	AXI data bus width	C_S_AXI_DATA_WIDTH	32	32	integer
G6	Minimum address range of the IP	C_S_AXI_MIN_SIZE	Valid range(2)	4KB = '0x1000'	std_logic_vector
Decoder Address Range Definition					
G7	Array of base address / high address pairs for each address range	C_ARD_ADDR_RANGE_ARRAY(3)		User must set values	SLV64_ARRAY_TYPE(3)
G8	Array of the desired number of chip enables for each address range	C_ARD_NUM_CE_ARRAY(3)		User must set values	INTEGER_ARRAY_TYPE(3)

Notes:

1. If the C_USE_WSTRB = 0, the Bus2IP_BE = 1111. Otherwise Bus2IP_BE = S_AXI_WSTRB.
2. The C_S_AXI_MIN_SIZE indicates the minimum size of the address space required by the IP. The minimum size of the address space is IP-specific and must be a power of two. Setting C_S_AXI_MIN_SIZE to a larger value than required by the IP will utilize more logic than needed.
3. This VHDL parameter is a custom type defined in the `ipif_pkg.vhd` file.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Constraining the Core

There are no IP-specific constraints other than the AXI clock constraint.

This is a helper IP. This IP needs to be instantiated in the design. You can add constraints as needed for your design requirements.

This section is not applicable for this core.

Required Constraints

This section is not applicable for this core.

Device, Package, and Speed Grade Selections

This section is not applicable for this core.

Clock Frequencies

This section is not applicable for this core.

Clock Management

This section is not applicable for this core.

Clock Placement

This section is not applicable for this core.

Banking

This section is not applicable for this core.

Transceiver Placement

This section is not applicable for this core.

I/O Standard and Placement

This section is not applicable for this core.

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4].

IMPORTANT: *For cores targeting 7 series or Zynq-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Migrating and Upgrading

This appendix contains information about migrating a design from ISE® Design Suite 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 Methodology Guide* (UG911) [Ref 6].

Upgrading in the Vivado Design Suite

AXI4-Lite IPIF is a helper core. For information on upgrading from older versions of the core to the current release, see the product guide for the core that instantiates AXI4-Lite IPIF.

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 core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the AXI4-Lite IPIF. 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 are listed below, and can also be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as:

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

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

Master Answer Record for the Core

AR: 54920

Technical Support

Xilinx provides technical support at 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.

Vivado Design Suite Debug Feature

There are many tools available to address AXI4-Lite IPIF design issues. It is important to know which tools are useful for debugging various situations.

Note: Because this is a helper core, debugging is accomplished in association with the slave core that instantiates the AXI4-Lite IPIF core.

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

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

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

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 6].

Interface Debug (AXI4-Lite Interfaces)

AXI4-Lite IPIF is a helper core. Instantiate this core in any of the slave peripheral cores which have the appropriate register set. These registers are targeted for write or read based upon their respective characteristics. If write is allowed to a given register, initiate a simple write command with valid data and determine if the transaction is completed with proper `ack` and `S_AXI_BVALID` signals. The same register can also be read back which should return the recently updated value with proper `ack` and `S_AXI_RVALID` signals at the AXI4-Lite interface. See Figure 3-4, page 23 for a read timing diagram. Output `S_AXI_ARREADY` asserts when the read address is valid, and output `S_AXI_RVALID` asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The `S_AXI_ACLK` and `aclk` inputs are connected and toggling.
- The interface is not being held in reset, and `S_AXI_ARESETN` is an active-Low reset.
- The interface is enabled, and `S_AXI_ACLK` is active-High (if used).
- The main core clocks are toggling and that the enables are also asserted.
- If the simulation has been run, verify in simulation and/or use the Vivado debug feature to verify that the waveform is correct for accessing the AXI4-Lite interface.
- At a given time, this core accepts either AXI read or AXI write transaction.
- All the AXI transactions are word accessed. No half word or byte accessible transactions are supported by the core.
- The AXI transaction will be passed to the downstream core through `Bus2IP_CS`, `Bus2IP_RdCE` (for read), `Bus2IP_WrCE` (for write along with `Bus2IP_Data`, `Bus2IP_BE`).
- The number of `Bus2IP_WrCE` and `Bus2IP_RdCE` are dependent on the number assigned in the C_ARD_NUM_CE_ARRAY.
- The number of `Bus2IP_CS` are dependent upon the number assigned in the C_ARD_ADDR_RANGE_ARRAY.
- All `Bus2IP_*` signals are held active until the downstream device completes the transaction by providing the read ack or write ack.
- To make sure the AXI transactions are not hanged indefinitely, keep the timeout counter (optional). Once the counter is over, the core completes the AXI transaction with slave error.
- Refer the correct assertion of `Bus2IP_RdCE` or `Bus2IP_WrCE` signals for reading the registers of downstream device.
- The `Bus2IP_Resetn` is active-Low.

Additional Resources

Xilinx Resources

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

www.xilinx.com/support

References

These documents provide supplemental material useful with this product guide:

1. ARM AMBA AXI Protocol v2.0 (ARM IHI 0022D)
2. *Vivado Design Suite User Guide: Designing with IP* (UG896)
3. *AXI Reference Guide* (UG1037)
4. *AXI Interconnect LogiCORE IP Product Guide* (PG059)
5. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
6. *ISE to Vivado Design Suite Migration Guide* (UG911)
7. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
8. *7 Series FPGA Overview* (DS180)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/06/2016	3.0	Updated the core version and restructured the product guide to conform to the current template.
03/20/2013	1.0	Initial release in product guide format. This document replaces DS765. The following items indicate new information not present in the prior document: • Added Feature Summary section • Added Maximum Frequencies section details • Convert all port names to lower case • Update file path names for Vivado • Add details to Clocking and Resets sections • Replace Figures 3-4 and 3-5 with updated images • Remove Single Read Error Operation section from Chapter 3 • Removed Parameter — Port Dependencies section from Chapter 4 • Removed Parameter Detailed Descriptions section from Chapter 4 • Revised Output Generation section • Added helper core description to Interface Debug (AXI4-Lite Interfaces) section of Appendix B
10/02/2013	2.0	Re-release of core v2.0 product guide, with the following changes made: • Version in this table updated to match core version. • Updated parameter in the ARD Arrays and CS/CE Relationship Example figure. • Restored port names to uppercase or mixed case. • Added further conditions to check for in the Interface Debug section (AXI4-Lite Interfaces). • Added Simulation, and Synthesis and Implementation chapters.
12/18/2013	2.0	Added UltraScale™ architecture support.

Notice of Disclaimer