LogiCORE IP AXI Lite IPIF (axi_lite_ipif) (v1.01.a)

DS765 December 14, 2010 **Product Specification**

Introduction

The AXI Lite IPIF is a part of the Xilinx family of Advanced RISC Machine (ARM®) Advanced Microcontroller Bus Architecture (AMBA®) Advanced eXtensible Interface (AXI) control interface compatible products. It provides a point-to-point bi-directional interface between a user IP core and the AXI interconnect. This version of the AXI Lite IP Interface (IPIF) has been optimized for slave operation on the AXI. It does not provide support for 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 the higher priority over write
- Reads to the holes in the address space returns 0x00000000
- Writes to the holes in the address space after the register map are ignored and responded with an OKAY response.
- Both AXI and IP Interconnect (IPIC) are little endian.

LogiCORE IP Facts Table				
Core Specifics				
Supported Device Family[1]	Virtex-6 Spartan-6			
Supported User Interfaces	AXI4-Lite			
	Resources[2]			Frequency
Configuration	**Slices**	FFs	LUTs	Max Freq
	See Table 5 and Table 6			
Provided with Core				
Documentation	Product Specification			
Design Files	VHDL			
Example Design	Not Provided			
Test Bench	Not Provided			
Constraints File	None			
Simulation Model	None			
Tested Design Tools				
Design Entry Tools	XPS 12.4 software			
Simulation	Mentor Graphics ModelSim q6.5.c			
Synthesis Tools	XST 12.4			
Support				
Provided by Xilinx, Inc.				

1. For a complete listing of supported devices, see the release notes for this core.
2. Resources are for Spartan-6 and Virtex-6 devices. For more complete device performance numbers, see Table 5 and Table 6.

Functional Description

The AXI Lite IPIF is designed to provide the user with a quick way to implement a light-weight interface between the ARM AXI and a User IP core. This slave service allows for multiple User IPs to be interfaced to the AXI providing address decoding over various address ranges as configured by the user. Figure 1 shows a block diagram of the AXI Lite IPIF. The port references and groupings are detailed in Table 1.

The base element of the design is the Slave Attachment. This block provides the basic functionality for slave operation. It implements the protocol and timing translation between the AXI and the IPIC.

The Address Decoder module generates the necessary chip select and read/write chip enable signals based upon the user requirement. The timeout counter will be added in the design if the C_DPHASE_TIMEOUT parameter is non-zero. If C_DPHASE_TIMEOUT = 0, the user must make sure that the core will generate the acknowledge signals for all the transactions.

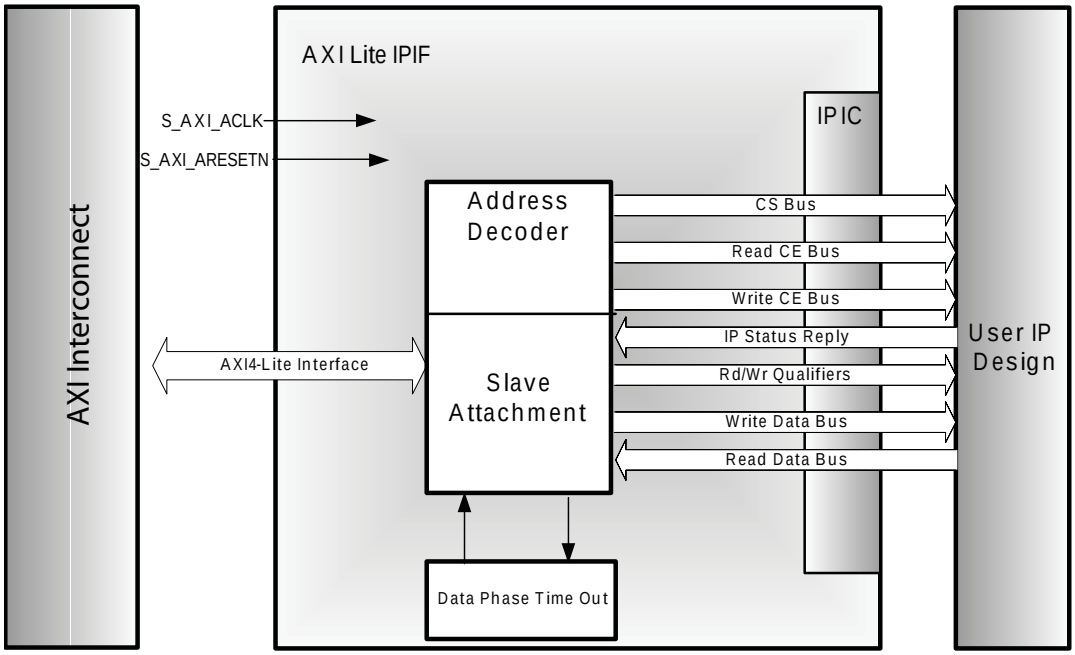

Figure 1: **AXI Lite IPIF Block Diagram**

I/O Signals

The AXI Lite IPIF signals are listed and described in Table 1.

Table 1: **IO Signal Description**

Port	Signal Name	Interface	I/O	Initial State	Description
AXI Global System Signals (1)					
P1	S_AXI_ACLK	AXI	I	-	AXI Clock
P2	S_AXI_ARESETN	AXI	I	-	AXI Reset, active low
AXI Write Address Channel Signals (1)					
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)					
P6	S_AXI_WDATA[C_S_AXI_DATA_WIDTH - 1:0]	AXI	I	-	Write data
P7	S_AXI_WSTB[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)					
P10	S_AXI_BRESP[1:0] (4)	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.
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)					
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 (2)	AXI	I	-	Read address valid. This signal indicates, when high, that the read address and control information is valid and will remain 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.

Table 1: **IO Signal Description** ***(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
AXI Read Data Channel Signals (1)					
P16	S_AXI_RDATA[C_S_AXI_DATA_WIDTH -1:0]	AXI	O	0	Read data
P17	S_AXI_RRESP[1:0] (4)	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.
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 the 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 a active transaction request to the chip select's target address space.

Table 1: **IO Signal Description *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P34	Bus2IP_RdCE [see note (3) : 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.
P35	Bus2IP_WrCE [see note (3) : 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.

1. These function and timing of these signals are defined in the *AMBA AXI Protocol Version: 2.0 Specification*.
2. Read transactions have higher priority than write transactions.
3. 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.
4. For signals like, S_AXI_RRESP[1:0] and S_AXI_BRESP[1:0], the IP core does not generate the Decode Error ('11') response. Other responses like '00' (OKAY) and '10' (SLVERR) are generated by the core based upon certain conditions.

Design Parameters

To allow the user to create an AXI Lite IPIF that is uniquely tailored for the system, certain features can be parameterized in the AXI Lite IPIF design. This allows the user to have a design that only utilizes the resources required by the system and operates at the best possible performance. The AXI Lite IPIF design parameters are shown in Table 2.

In addition to the parameters listed in this table, there are also parameters that are inferred for each AXI interface in the EDK tools. Through the design, these EDK-inferred parameters control the behavior of the AXI Interconnect. For a complete list of the interconnect settings related to the AXI interface, see DS768, *AXI Interconnect IP Data Sheet*.

Table 2: **Design Parameters**

Generic	Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
System Parameters					
G1	Target FPGA family	C_FAMILY	virtex6, spartan6	virtex6	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_IPIF_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)	See Parameter Detailed Descriptions	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)	See Parameter Detailed Descriptions	User must set values.	INTEGER_ARRAY_TYPE (3)

1. If the C_USE_WSTRB = 0, the Bus2IP_BE = "1111". Otherwise Bus2IP_BE = S_AXI_WSTB.
2. The C_S_AXI_IPIF_MIN_SIZE will indicate the minimum size of the address space required by the IP. The min size of the address space is IP specific and must be a power of 2.
3. This VHDL parameter type is a custom type defined in the ipif_pkg.vhd.

Parameter - Port Dependencies

Allowable Parameter Combinations

Table 3: **Parameter - Port Dependencies**

Generic or Port	Name	Affects	Depends	Relationship Description
Design Parameters				
G4	C_S_AXI_ADDR_WIDTH	P3, P13	-	Defines the width of the ports
G5	C_S_AXI_DATA_WIDTH	P6, P7, P16, P25, P30, P32	-	Defines the width of the ports
G7	C_ARD_ADDR_RANGE_ARRAY	P33	-	The vector width of Bus2IP_CS is the number of elements in C_ARD_ADDR_RANGE_ARRAY/2.
G8	C_ARD_NUM_CE_ARRAY	P34, P35	-	The vector width of Bus2IP_WrCE is the number of elements in C_ARD_NUM_CE_ARRAY.
I/O Signals				
P3	S_AXI_AWADDR[C_S_AXI_ADDR_WIDTH-1:0]	-	G4	Port width depends on the generic C_S_AXI_ADDR_WIDTH.
P6	S_AXI_WDATA[C_S_AXI_DATA_WIDTH-1:0]	-	G5	Port width depends on the generic C_S_AXI_DATA_WIDTH.
P7	S_AXI_WSTB[C_S_AXI_DATA_WIDTH/8-1:0]	-	G5	Port width depends on the generic C_S_AXI_DATA_WIDTH.
P13	S_AXI_ARADDR[C_S_AXI_ADDR_WIDTH -1 :0]	-	G4	Port width depends on the generic C_S_AXI_ADDR_WIDTH.
P16	S_AXI_RDATA[C_S_AXI_DATA_WIDTH -1:0]	-	G5	Port width depends on the generic C_S_AXI_DATA_WIDTH.
P25	IP2Bus_Data[C_S_AXI_DATA_WIDTH -1:0]	-	G5	Port width depends on the generic C_S_AXI_DATA_WIDTH.
P29	Bus2IP_Addr[C_S_AXI_ADDR_WIDTH - 1:0]	-	G4	Port width depends on the generic C_S_AXI_ADDR_WIDTH.
P30	Bus2IP_Data[C_S_AXI_DATA_WIDTH-1:0]	-	G5	Port width depends on the generic C_S_AXI_DATA_WIDTH.
P32	Bus2IP_BE[(C_S_AXI_DATA_WIDTH/8)-1:0]	-	G5	Port width depends on the generic C_S_AXI_DATA_WIDTH.
P33	Bus2IP_CS	-	G7	The vector width of Bus2IP_CS is the number of elements in C_ARD_ADDR_RANGE_ARRAY/2.
P34	Bus2IP_WrCE	-	G8	The vector width of Bus2IP_WrCE is the number of elements in C_ARD_NUM_CE_ARRAY.
P35	Bus2IP_RdCE	-	G8	The vector width of Bus2IP_RdCE is the number of elements in C_ARD_NUM_CE_ARRAY.

Parameter Detailed Descriptions

Address Range Definition Arrays

One of the primary functions of the AXI Lite IPIF is to provide address decoding and Chip Enable/Chip Select control signal generation.

The AXI Lite IPIF employs VHDL Generics that are defined as unconstrained arrays as the method for customizing address space decoding. These parameters are called the Address Range Definition (ARD) arrays. There are two of these arrays used for address space definition in the AXI Lite IPIF. They can be recognized by the "C_ARD" prefix of the Generic name. The ARD Generics are:

- C_ARD_ADDR_RANGE_ARRAY
- C_ARD_NUM_CE_ARRAY

One of the big advantages of using unconstrained arrays for address decode space description is that it allows the user to specify as few or as many unique and non-contiguous AXI address spaces as the peripheral design needs. The Slave Attachment decoding logic will be optimized to recognize and respond to only those defined address spaces during active AXI transaction requests.

Since the number of entries in the arrays can grow or shrink based on each User Application, the slave attachment is designed to analyze the user entries in the arrays and then automatically add or remove resources, and interconnections based on the contents of the arrays. A special case arises when there is a single entry in an unconstrained array. See the section titled Single Entry in Unconstrained Array Parameters for hints on entering data for this case.

The ordering of a set of address space entries within the ARD arrays is not important. Each address space is processed independently from any of the other address space entries. However, once an ordering is established in any one of the arrays, that ordering of the entries must be maintained in the other ARD array. That is, the first two entries in C_ARD_ADRR_RANGE_ARRAY will be associated with the first CE Number entry in the C_ARD_NUM_CE_ARRAY.

C_ARD_ADDR_RANGE_ARRAY

The actual address range for an address space definition is entered in this array. Each address space is by definition a contiguous block of addresses as viewed from the host microprocessor's total addressable space. Its specification requires a pair of entries in this array. The first entry of the pair is the Base Address (starting address) of the block, the second entry is the High Address (ending address) of the block. These addresses are byte relative addresses. The array elements are defined as std_logic_vector(0 to 63) in the ipif_pkg.vhd file in Processor Common (proc_common) library. Currently, the biggest address bus used on the AXI is 32 bits. However, 64 bit values have been allocated for future growth in address bus width.

```
C_ARD_ADDR_RANGE_ARRAY : SLV64_ARRAY_TYPE :=
        -- Base address and high address pairs .
         (
         "X0000_0000_A000_0000",   -- user control reg bank base address
         "X0000_0000_A000_000F",   -- user control reg bank high address
         "X0000_0000_A000_0100",   -- user status reg bank base address
         "X0000_0000_A000_013F"    -- user status reg bank high address
         );
```

Figure 2: **Address Range Specification Example**

The user must follow several rules when assigning values to the address pairs. These rules assure that the address range will be correctly decoded in the Slave Attachment. First, the user must decide the required address range to be defined. The block size (in bytes) must be a power of 2 (that is 2, 4, 8,16,32,64,128,256 and so on). Secondly, the Base Address must start on an address boundary that is a multiple of the chosen block size. For example, an address space is needed that will include 2048 bytes (0x800 hex) of the system memory space. Valid Base Address entries are 0x00000000, 0x00000800, 0xFFFFF000, 0x90001000, etc. A value of 0x00000120 is not valid because it is not a multiple of 0x800 (2048). Thirdly, the High Address entry is equal to the assigned Base Address plus the block size minus 1. Continuing the example of a 2048 byte block size, a Base Address of 0x00000000 yields a High Address of 0x000007FF; a Base Address of 0x00000800 would require a corresponding High Address value of 0x00000FFF.

In the preceding example the first address range has four registers and the high address of the first address range is 0x0000_0000_A000_000F. As there are four registers only, to reduce the decode logic the high address should be as per the requirement. It should not be more than required. In this example it should not be 0x0000_0000_A000_001F, as this space can accommodate 8 registers.

C_ARD_NUM_CE_ARRAY

The slave decoding logic provides the user the ability to generate multiple chip enables within a single address space. This is primarily used to support a bank of registers that need an individual chip enable for each register. The user enters the desired number of chip enables for an address space in the C_ARD_NUM_CE_ARRAY. The values entered are positive integers that are powers of 2 (1, 2, 4, 8, 16, 32, and so on). Each address space must have at least 1 chip enable specified. The address space range will be subdivided and sequentially assigned a chip enable based on a data width or 32 bits.

The user must ensure that the address space for a group of chip enables is greater than or equal to the specified width of the memory space in bytes (32 / 8 = 4) times the number of desired chip enables.

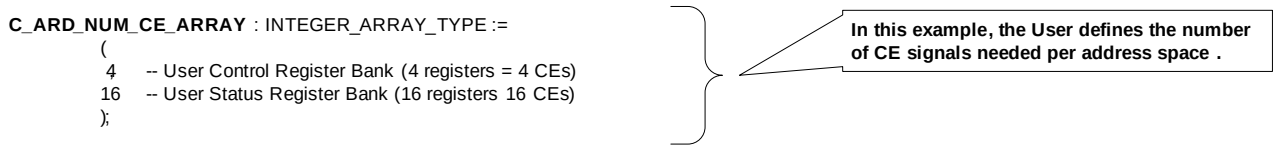

Figure 3: **Chip Enable Specification Example**

For example if the user IP has only 3 registers in an address range, the C_ARD_NUM_CE_ARRAY should configure for 4 registers as 3 is not a power of 2.

C_S_AXI_ADDR_WIDTH

This integer parameter is used by the AXI Slave to size the AXI address related components within the Slave Attachment. This value should be set to 32 bits.

C_S_AXI_DATA_WIDTH

This integer parameter is used by the AXI slave to size AXI data bus related components within the Slave Attachment. This value should be set to 32 bits.

C_S_AXI_IPIF_MIN_SIZE

This parameter indicates the minimum size required for the slave's register map. This parameter must be a power of 2 and must be greater than or equal to the high address of the last address range. For example, if the high address of the last address range is 0x0000030F, then the C_S_AXI_IPIF_MIN_SIZE should be 0x000003FF. If the high address of the last address range is 0x000003FF, then the C_S_AXI_IPIF_MIN_SIZE should be same as the high address of the last address range, that is, 0x000003FF. The actual address range assigned to the IP in the system may be larger than C_S_AXI_MIN_SIZE, as that would help reduce address decode logic in the system and must also comply with the AXI 4KB minimum range size rule.

C_USE_WSTRB

If this parameter set to 0, the byte enables passed to IPIC will be always "1111". If this parameter set to 1, the byte enables from the AXI will be transferred to IPIC.

C_DPHASE_TIMEOUT

Read or write transactions in to the holes in the address space will be addressed by the Data Phase Time out. If there is no response from the IP for a particular read or write transaction, the IPIF will wait for the C_DPHASE_TIMEOUT clocks cycles from the assertion of S_AXI_AWVALID/S_AXI_ARVALID and send the response to AXI interconnect. If the C_DPHASE_TIMEOUT = 0, the counter will not be implemented in the design.

C_FAMILY

This parameter is defined as a string. It specifies the target FPGA technology for implementation of the AXI Slave. This parameter is required for proper selection of FPGA primitives. The configuration of these primitives can vary from one FPGA technology family to another.

IPIC Port Detailed Descriptions

All of the IPIC signals are in little endian format in line with AXI.

Bus2IP_Addr

Bus2IP_Addr is a 32-bit vector that drives valid when Bus2IP_CS and Bus2IP_RdCE or Bus2IP_WrCE drives high.

Bus2IP_Data

Bus2IP_Data is a vector of width C_S_AXI_DATA_WIDTH and drives valid on writes when Bus2IP_WrCE is high.

Bus2IP_RNW

Bus2IP_RNW is a signal indicating the type of transfer in progress and is valid when Bus2IP_CS and Bus2IP_WrCE or Bus2IP_RdCE is asserted. A high on Bus2IP_RNW indicates the transfer request is a read of the User IP. A low on Bus2IP_RNW indicates the transfer request is a write to the User IP.

Bus2IP_BE

Bus2IP_BE is a vector of width C_S_AXI_DATA_WIDTH/8. This vector indicates which bytes are valid on Bus2IP_DATA. Bus2IP_BE becomes valid coincident with Bus2IP_CS. As all of the AXI transactions are 32-bit wide the BusIP_BE will be tied to "1111".

Bus2IP_CS

Bus2IP_CS is a vector of width C_ARD_ADDR_RANGE_ARRAY'length / 2. In other words, for each address pair defined in C_ARD_ADDR_RANGE_ARRAY there is 1 Bus2IP_CS defined. This signal 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 to memory type devices.

Bus2IP_RdCE

Bus2IP_RdCE is a vector of a width that is the sum total of the values defined in C_ARD_NUM_CE_ARRAY. For each address pair defined in C_ARD_ADDR_RANGE_ARRAY a number of CEs can be defined in C_ARD_NUM_CE_ARRAY. Bus2IP_RdCE goes high coincident with Bus2IP_CS for read type transfers and is especially suited for reading registers.

Bus2IP_WrCE

Bus2IP_WrCE is a vector of a width that is the sum total of the values defined in C_ARD_NUM_CE_ARRAY. For each address pair defined in C_ARD_ADDR_RANGE_ARRAY a number of CEs 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

IP2Bus_Data is a vector of width C_S_AXI_DATA_WIDTH and is the read data bus. Read data should be valid when IP2Bus_RdAck is asserted by the User IP.

IP2Bus_RdAck

IP2Bus_RdAck is the read data acknowledge signal. This signal is used by the User IP to acknowledge a read cycle and will cause read control signals, Bus2IP_RdCE, Bus2IP_CS and Bus2IP_RNW to deassert.

IP2Bus_WrAck

IP2Bus_WrAck is the write data acknowledge signal. This signal is used by the User IP to acknowledge a write cycle and will cause write control signals, Bus2IP_WrCE, and Bus2IP_CS to deassert.

IP2Bus_Error

IP2Bus_Error is used by the User IP to indicate an error has occurred. This signal is only sampled with IP2Bus_WrAck or IP2Bus_RdAck and is ignored all other times. This will be sent to the interconnect as a slave error.

Usage of the IPIF

Here is the usage description for the AXI Lite IPIF. If the C_S_AXI_IPIF_MIN_SIZE = 0x1FF, the S_AXI_AWADDR[11:0] will be used for the address decoding. The width of the Bus2IP_CS will be 2 and the width of the BusIP_WrCE, Bus2IP_RdCE will be 20. Considered C_DPHASE_TIMEOUT = 16.

```
C_ARD_ADDR_RANGE_ARRAY : SLV64_ARRAY_TYPE :=
        -- Base address and high address pairs .
         (
         "X0000_0000_000_0000",  -- user control reg bank base address
         "X0000_0000_000_000F",  -- user control reg bank high address
         "X0000_0000_0000_0100",-- user status reg bank base address
         "X0000_0000_0000_013F" -- user status reg bank high address
         )

C_ARD_NUM_CE_ARRAY : INTEGER_ARRAY_TYPE :=
        -- Number of registers in the each address range .
         (
          4 -- User Control Register Bank (4 registers = 4 CEs)

         16-- User Status Register Bank (16 registers = 16 CEs)
         );
```

Figure 4: **Chip Select/ Chip Enable Example**

Case 1

If there is a write transaction for the address 0x00000000, the IPIF will assert the Bus2IP_CS(0) and BusIP_WrCE(19) and wait for the IP2Bus_WrAck from the IP.

If there is a read transaction for the address 0x00000000, the IPIF will assert the Bus2IP_CS(0) and BusIP_RdCE(19) and wait for the IP2Bus_RdAck from the IP.

Case 2

If there is a write transaction for the address 0x000000F0, the IPIF will not generate the Bus2IP_CS and BusIP_WrCE as this address fall in between the two address ranges. The IPIF will send the OKAY response to AXI.

If there is a read transaction for the address 0x000000F0, the IPIF will not generate the Bus2IP_CS and BusIP_RdCE as this address fall in between the two address ranges. The IPIF will send the OKAY response to AXI.

In both of the preceding use cases, the user should take care to put 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 the address 0x00000100, the IPIF will assert the Bus2IP_CS(1) and BusIP_WrCE(15) and wait for the IP2Bus_WrAck from the IP.

If there is a read transaction for the address 0x00000100, the IPIF will assert the Bus2IP_CS(1) and BusIP_RdCE(15) and wait for the IP2Bus_RdAck from the IP.

Case 4

If there is a write transaction for the address 0x00000140, the IPIF will not generate the Bus2IP_CS and BusIP_WrCE as this address fall after the high address of the second address range. The IPIF will send the OKAY response to AXI.

If there is a read transaction for the address 0x00000140, the IPIF will not generate the Bus2IP_CS and BusIP_RdCE as this address fall after the high address of the second address range. The IPIF will send the OKAY response to AXI.

In both of the preceding use cases, the user should take care to put 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 the address 0x00000200, the IPIF will assert the Bus2IP_CS(0) and BusIP_WrCE(19) and wait for the IP2Bus_WrAck from the IP. This is because after the address 0x00000200 address wrapping will happen.

If there is a read transaction for the address 0x00000200, the IPIF will assert the Bus2IP_CS(0) and BusIP_RdCE(19) and wait for the IP2Bus_RdAck from the IP. This is because after the address 0x00000200 address wrapping will happen.

IPIC Transaction Timing

The following section shows timing relationships for AXI and Slave Attachment interface signals during various read and write accesses.

Single Write Operation

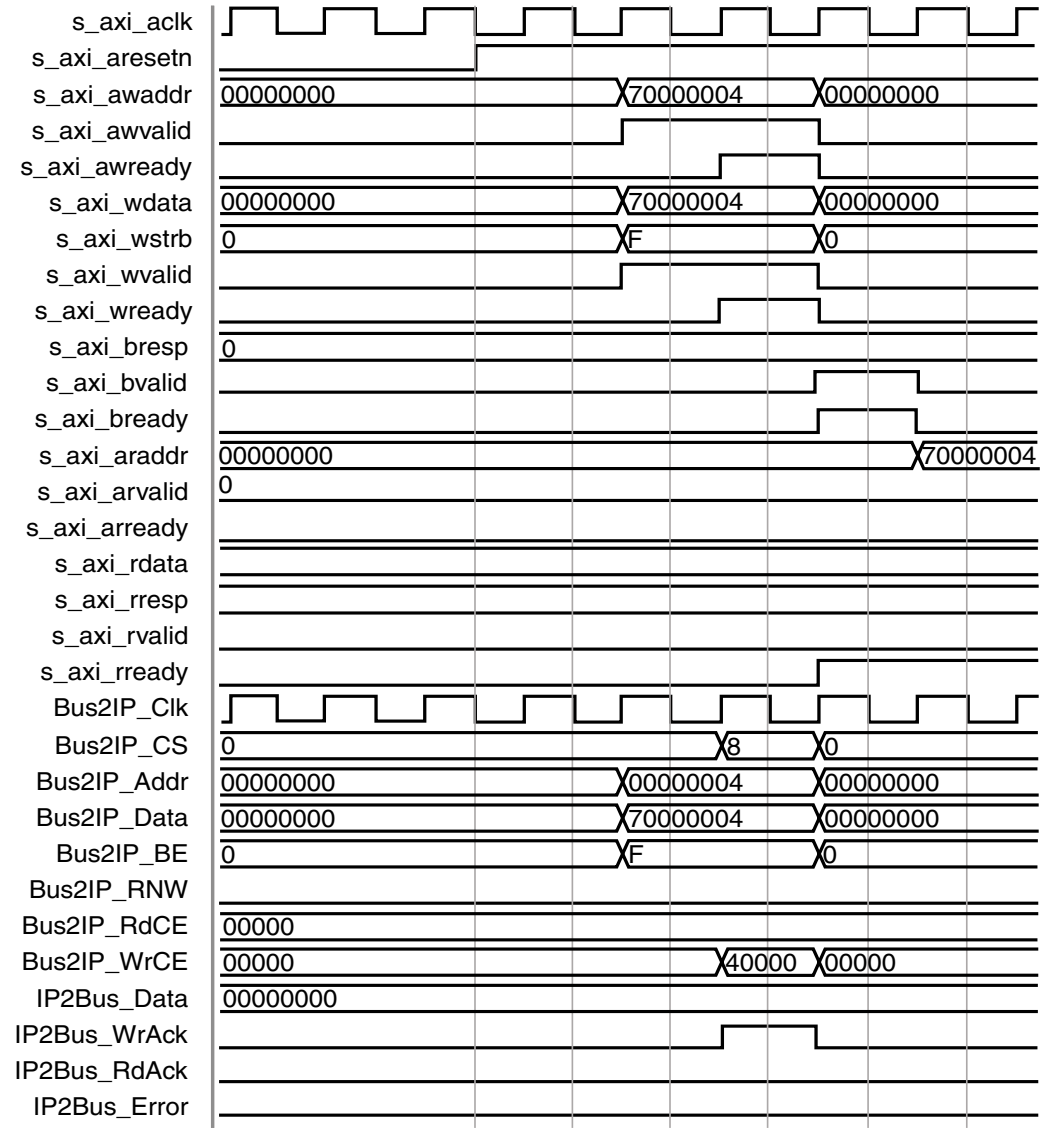

Figure 5: **AXI Lite IPIF Single Write Operation**

Single Read Operation

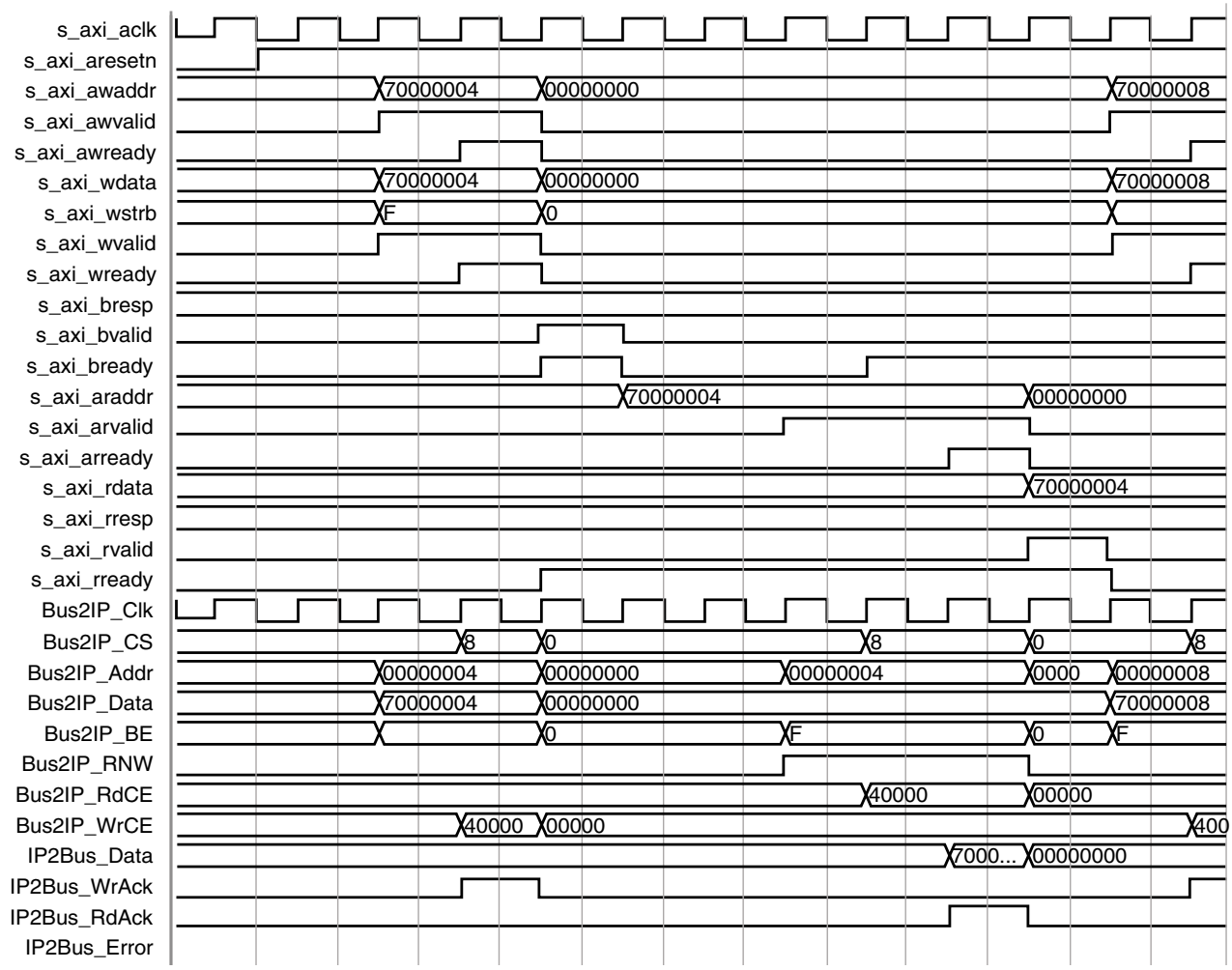

Figure 6: **AXI Lite IPIF Single Read Operation**

User Application Topics

Understanding and Using IPIC Chip Selects and Chip Enables

Implementing Chip Select (CS) and Chip Enable (CE) signals is a common design task that is needed within microprocessor-based systems to qualify the selection of registers, ports, and memory via an address decoding function. The AXI Lite IPIF implements a flexible technique for providing these signals to users via the ARD parameters. As such, the user must understand the relationship between the population of the ARD array parameters and the Bus2IP_CS, the Bus2IP_RdCE, and the Bus2IP_WrCE buses that are available to the user at the IPIC interface with the Slave Attachment. An example of ARD Array population and the resulting CS and CE bus generation is shown in Figure 7. The signal set to use for User IP functions is up to the user and the design requirements. Unused CE and CS signals and associated generation logic will be trimmed during synthesis and PAR phases of FPGA development.

Chip Select Bus (Bus2IP_CS(n:0))

A single Chip Select signal is assigned to each address space defined by the user 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 user is provided the Bus2IP_CS port 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 in it, the Bus2IP_CS bus will be sized as 0 to 4. Bus2IP_CS(0) will correspond 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 are allowed to have one or more Chip Enables 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 width. Generally this is useful for selecting registers and ports during read or write transactions. The Slave Attachment allows the user to do this via parameters entered in the C_ARD_NUM_CE_ARRAY. For each defined address space, the user enters the number of desired Chip Enable signals to be generated for each space. Current 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.

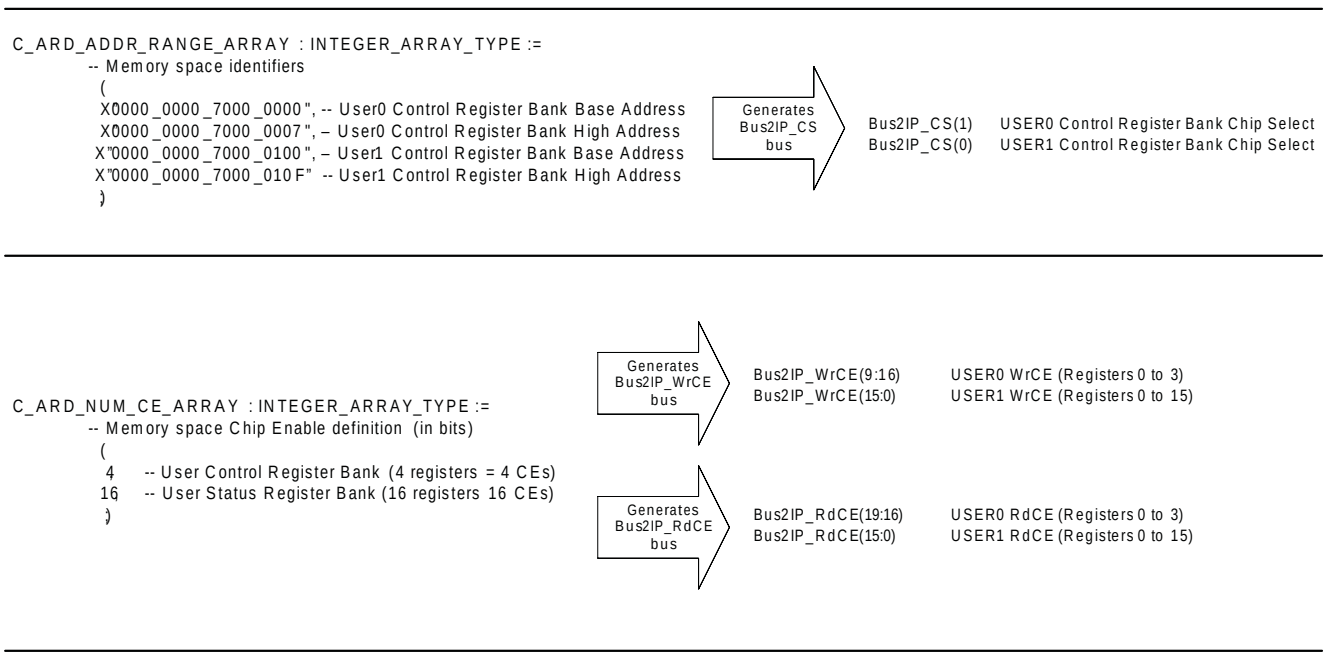

Figure 7: **ARD Arrays and CS/CE Relationship Example**

Available Support Functions for Automatic Separation of CE and CS Buses

The user may find it 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 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 4 lists and details these functions. These functions are declared and defined in the ipif_pkg.vhd source file that is located in the Xilinx EDK at the following path:

\EDK\hw\XilinxProcessorIPLib\pcores\proc_common_v3_00_a\hdl\vhdl\ipif_pkg.vhd.

The following library declaration must appear in the user VHDL source:

```
library proc_common_v3_00_a;
use proc_common_v3_00_a.ipif_pkg.all;
```

An example of how these functions are used is shown in Figure 8.

Table 4: **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);

  -- Determine the CE indexes to use for the the second Register CE
    Constant USER_01_START_CE_INDEX   : integer := calc_start_ce_index(C_ARD_NUM_CE_ARRAY, 1);

    Constant USER_01_END_CE_INDEX    : integer := USER_01_START_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 downto 0);
    signal user_01_wrce               : std_logic_vector(NUM_USER_01_CE -1 downto 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 downto USER_01_START_CE_INDEX);
   user_01_wrce        <= Bus2IP_WrCE(USER_01_END_CE_INDEX downto USER_01_START_CE_INDEX);
```

Figure 8: **Bus Separation Example**

FPGA Design Application Hints

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 problem, the user should use ***named association*** for the single array entry. This is shown in the following example:

C_ARD_NUM_CE_ARRAY(16); -- *VHDL* ***positional association****....may 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.*

Register Descriptions

The AXI Lite IPIF has no internal registers.

Design Implementation

Target Technology

The intended target technology is a Spartan®-6 or Virtex®-6 FPGAs.

Device Utilization and Performance Benchmarks

Since the AXI Lite IPIF is a module that will be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are just estimates. As the AXI Lite IPIF is combined with other pieces of the FPGA design, the utilization of FPGA resources and timing will vary from the results reported here.

The resource utilization of this version of the AXI Lite IPIF is shown here for some example configurations. The Slave Attachment was synthesized using the Xilinx® XST tool. The XST resource utilization report was then used as the source data for the table.

The AXI Lite IPIF benchmarks are shown in Table 5 for a Spartan-6 (xc6slx45-2-fgg484) FPGA.

Table 5: **AXI Lite IPIF FPGA Performance and Resource Utilization Benchmarks**

Parameter Values				Device Resources for Spartan-6			Performance
C_ARD_ADDR_RANGE_ARRAY Pairs	C_ARD_NUM_CE_ARRAY	C_DPHASE_TIMEOUT	C_USE_WSTRB	Slices	Slice Flip-Flops	LUTs	f_{MAX}[1]
2	4, 8	8	0	49	60	97	110
2	4, 8	8	1	50	60	100	110
4	4, 8, 16, 8	512	0	49	65	100	110
4	4, 8, 16, 8	512	1	51	65	102	110
4	4, 8, 16, 8	0	0	38	60	77	110
4	4, 8, 16, 8	0	1	41	60	79	110

1. Fmax represents the maximum frequency of the AXI Lite IPIF in a standalone configuration. The actual maximum frequency will depend on the entire system and may be greater or less than what is recorded in this table.
2. For above utilization calculation the parameter C_S_AXI_MIN_SIZE = X"000001FF" is used.

The AXI Lite IPIF benchmarks are shown in Table 6 for a Virtex-6 (xc6vlx195t-1-ff1156) FPGA.

Table 6: **FPGA Performance and Resource Utilization Benchmarks**

Parameter Values				Device Resources for Virtex-6			Performance
C_ARD_ADDR_RANGE_ARRAY Pairs	C_ARD_NUM_CE_ARRAY	C_DPHASE_TIMEOUT	C_USE_WSTRB	Slices	Slice Flip-Flops	LUTs	f_{MAX}[1]
2	4, 8	8	0	49	60	97	200
2	4, 8	8	1	46	60	99	200
4	4, 8, 16, 8	512	0	45	61	99	200
4	4, 8, 16, 8	512	1	50	61	101	200
4	4, 8, 16, 8	0	0	41	55	91	200
4	4, 8, 16, 8	0	1	44	55	93	200

1. Fmax represents the maximum frequency of the AXI Lite IPIF in a standalone configuration. The actual maximum frequency will depend on the entire system and may be greater or less than what is recorded in this table.
2. For above utilization calculation the parameter C_S_AXI_MIN_SIZE = X"000001FF" is used.

Specification Usage

'The AXI Lite IPIF Slave has the following characteristics.

- 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.
- Does not support multiple outstanding transactions.
- If there is a simultaneous read/write on AXI, read has the higher priority over write.
- Reads to 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.
- IPIF will not do endian conversion. Both AXI and IP Interconnect (IPIC) are little endian.

Reference Documents

The *AMBA AXI Protocol Version: 2.0 Specification* contains important reference information for understanding the AXI4-Lite Slave Attachment design.

To download the specification:

1. Register at infocenter.arm.com
2. Navigate on left contents panel to AMBA > AMBA specifications > AMBA AXI Protocol Specification v2.

Support

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

Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx ISE® Design Suite Embedded Edition software under the terms of the Xilinx End User License. The core is generated using the Xilinx ISE Embedded Edition software (EDK).

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

List of Acronyms

Acronym	Spelled Out
AMBA	Advanced Microcontroller Bus Architecture
ARD	Address Range Definition
ARM	Advanced RISC Machine
AXI	Advanced eXtensible Interface
CE	Chip Enable
CS	Chip Select
DMA	Direct Memory Access
FF	Flip-Flop
FPGA	Field Programmable Gate Array
IP	Intellectual Property
IPIC	IP Interconnect
IPIF	IP Interface
ISE	Integrated Software Environment
LUT	Lookup Table
PAR	Place and Route
VHDL	VHSIC Hardware Description Language (VHSIC an acronym for Very High-Speed Integrated Circuits)
XST	Xilinx Synthesis Technology

Revision History

Date	Version	Revision
09/21/10	1.0	Initial Xilinx release
09/21/10	1.0.1	Documentation only. • Added inferred parameters text on page 6. • Updated P25 on page 7. • Corrected Figure 8 on page 15. • Updated Figure 9 on page 17.
12/14/10	1.1	Updated core version v1.01.a; updated to 12.4 design tools.

Notice of Disclaimer