PLBv46 Slave Burst (v1.01a)

Introduction

The PLBv46 Slave Burst core is a part of the Xilinx family of PLB v4.6 compatible products. It provides a bi-directional interface between a User IP core and the PLB v4.6 bus standard. This version of the core has been optimized for slave operation on the version 4.6 PLB Bus. It does not provide support for DMA and IP Master Services.

Features

- Compatible with Xilinx PLB v4.6 32, 64 and 128-bit PLB
- Supports access by 32, 64, and 128-Bit Masters
- Supports 32, 64 or 128-Bit slave configuration
- Supports Single Beat read and write data transfers
- Supports Fixed Burst read and write data transfers
- Supports Cacheline read and write data transfers
- Supports Low latency PLB Point-to-Point topology

LogiCORE™ Facts		
Core Specifics		
Supported Device Family	Virtex-4®, Virtex-5, Virtex-6, Spartan®-6, Spartan-3E, Automotive Spartan-3E, Spartan-3, Automotive Spartan-3, Spartan-3A, Automotive Spartan-3A, Spartan-3A DSP, Automotive Spartan-3A DSP	
Version of core	plbv46_slave_burst	v1.01a
Resources Used		
	Min	Max
Slices	NA	NA
LUTs	218	448
FFs	151	511
Block RAMs	None	
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	UCF	
Verification	VHDL Test bench	
Instantiation Template	VHDL Wrapper	
Additional Items	None	
Design Tool Requirements		
Xilinx Implementation Tools	ISE 8.1 or later	
Verification	ModelSim PE 6.1d	
Simulation	ModelSim PE 6.1d	
Synthesis	XST	
Support		
Provided by Xilinx, Inc.		

Functional Description

The PLBv46 Slave Burst core is designed to provide a User with a quick way to implement an interface between the IBM PLB Bus and a User IP core. This slave service allows for multiple User IP's to be interfaced to the PLB bus providing address decoding over various address ranges as configured by the user. Optionally the PLBv46 Slave Burst core can be optimized for a point to point connection reducing FPGA resources and improving latency. Figure 1 shows a block diagram of the PLBv46 Slave Burst core. 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 PLB Bus and the IPIC.

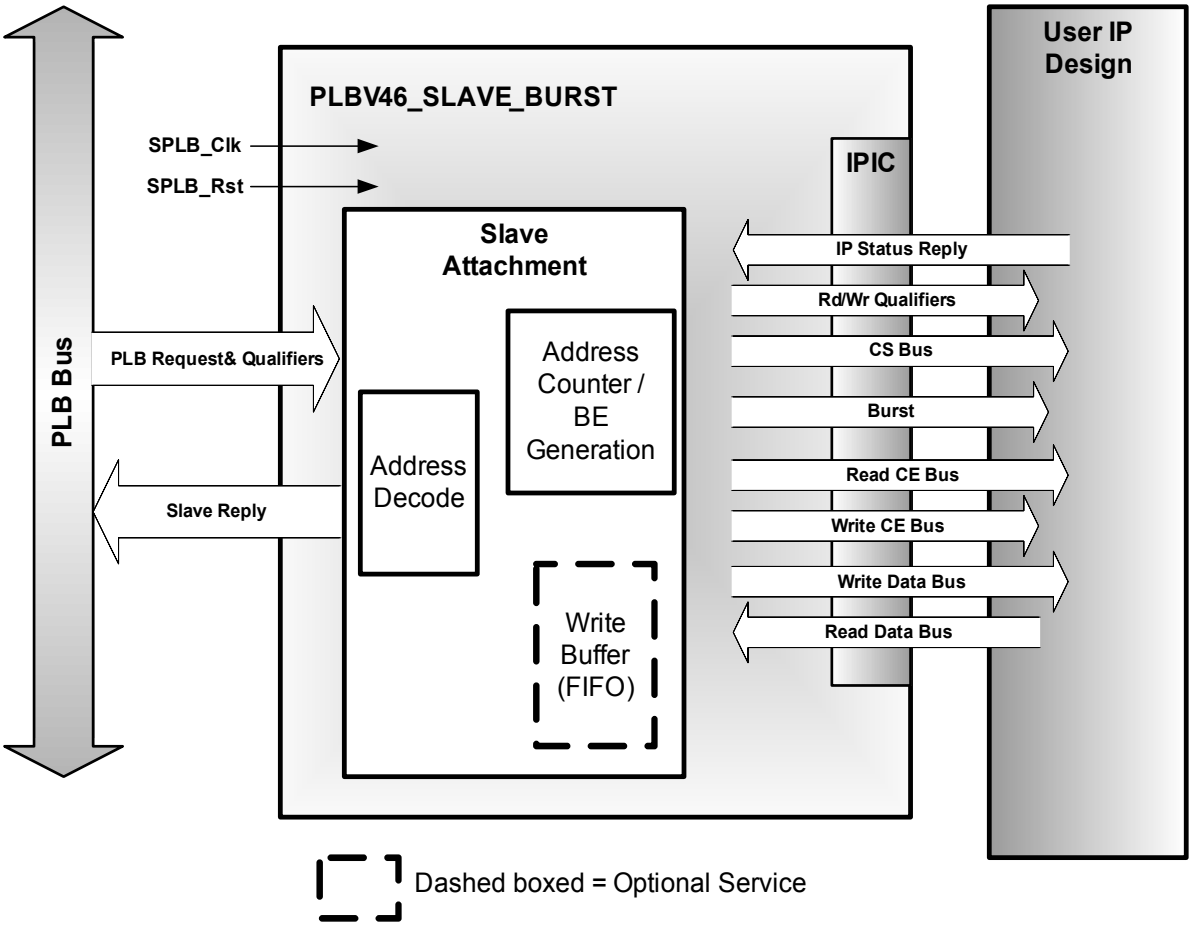

Figure 1: **PLBv46 Slave Burst Core Block Diagram**

I/O Signals

The PLBv46 Slave Burst core signals are listed and described in Table 1.

Table 1: **PLBv46 Slave Burst Core I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
PLB Bus Request and Qualifier Signals				
SPLB_Clk	PLB Bus	I		PLB main bus clock. See table note 1.
SPLB_Rst	PLB Bus	I		PLB main bus reset. See table note 1.
PLB_ABus(0:**31**)	PLB Bus	I		See table note 1.
PLB_PAValid	PLB Bus	I		See table note 1.
PLB_masterID(0:**C_SPLB_MID_WIDTH**-1)	PLB Bus	I		See table note 1.
PLB_RNW	PLB Bus	I		See table note 1.
PLB_BE(0:[**C_SPLB_DWIDTH**/8]-1)	PLB Bus	I		See table note 1.
PLB_wrBurst	PLB Bus	I		See table note 1.
PLB_rdBurst	PLB Bus	I		See table note 1.
PLB_size(0:3)	PLB Bus	I		See table note 1.
PLB_type(0:2)	PLB Bus	I		See table note 1.
PLB_wrDBus(0:**C_SPLB_DWIDTH**-1)	PLB Bus	I		See table note 1.
PLB_MSize(0:1)	PLB Bus	I		See table note 1.
Sl_addrAck	PLB Bus	O	0	See table note 1.
Sl_SSize(0:1)	PLB Bus	O	0	See table note 1.
Sl_wait	PLB Bus	O	0	See table note 1.
Sl_rearbitrate	PLB Bus	O	0	See table note 1.
Sl_wrDack	PLB Bus	O	0	See table note 1.
Sl_wrComp	PLB Bus	O	0	See table note 1.
Sl_wrBTerm	PLB Bus	O	0	See table note 1.
Sl_rdBus(0:**C_SPLB_DWIDTH**-1)	PLB Bus	O	0	See table note 1.
Sl_rdDAck	PLB Bus	O	0	See table note 1.
Sl_rdComp	PLB Bus	O	0	See table note 1.
Sl_rdBTerm	PLB Bus	O	0	See table note 1.
Sl_rdWdAddr(0:3)	PLB Bus	O	0	See table note 1.
Sl_MBusy(0:**C_SPLB_NUM_MASTERS**-1)	PLB Bus	O	0	See table note 1.

Table 1: **PLBv46 Slave Burst Core I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
SI_MWrErr(0:**C_SPLB_NUM_MASTERS**-1)	PLB Bus	O	0	See table note 1.
SI_MRdErr(0:**C_SPLB_NUM_MASTERS**-1)	PLB Bus	O	0	See table note 1.
Unused PLB Signals				
PLB_UABus(0:**31**))	PLB Bus	I		Unused
PLB_SAValid	PLB Bus	I		Unused
PLB_rdPrim	PLB Bus	I		Unused
PLB_wrPrim	PLB Bus	I		Unused
PLB_abort	PLB Bus	I		Unused
PLB_busLock	PLB Bus	I		Unused
PLB_TAttribute(0:15)	PLB Bus	I		Unused
PLB_lockerr	PLB Bus	I		Unused
PLB_wrPendReq	PLB Bus	I		Unused
PLB_rdPendReq	PLB Bus	I		Unused
PLB_rdPendPri(0:1)	PLB Bus	I		Unused
PLB_wrPendPri(0,1)	PLB Bus	I		Unused
PLB_reqPri(0:1)	PLB Bus	I		Unused
SI_MIRQ(0:**C_SPLB_NUM_MASTERS**-1)	PLB Bus	O	0	Unused
User IP Signals				
Bus2IP_Clk	User IP	O	0	Synchronization clock provided to User IP. This is the same as SPLB_Clk.
Bus2IP_Reset	User IP	O	0	Active high reset for use by the User IP. It is a pass through of the SPLB_Rst input.
IP2Bus_Data(0:**C_SIPIF_DWIDTH** -1)	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.
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 1: **PLBv46 Slave Burst Core I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
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.
IP2Bus_AddrAck	User IP	I		Active high signal that advances the address counter and request state during multiple data beat transfers.
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.
Bus2IP_Addr(0:**C_SPLB_AWIDTH** - 1)	User IP	O	0	Address bus indicating the desired address of the requested read or write operation.
Bus2IP_Data(0:**C_SIPIF_DWIDTH**-1)	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.
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.
Bus2IP_BE(0:(**C_SIPIF_DWIDTH**/8)-1)	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.
Bus2IP_Burst	User IP	O	0	Active high signal indicating that the active read or write operation with the User IP is utilizing bursting protocol. This signal is asserted at the initiation of a burst transaction with the User IP and de-asserted at the completion of the second to last data beat of the burst data transfer.
Bus2IP_BurstLength(0:$\log_2$(**16*C_SIPIF_DWIDTH/8**))(3)	User IP	O	0	This value is an indication of the number of bytes being requested for transfer and is valid when the cycle is of burst type when Bus2IP_CS=1.
Bus2IP_CS(0:(**C_ARD_ADDR_RANGE_ARRAY**'length/2) - 1)	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: **PLBv46 Slave Burst Core I/O Signal Description**

Signal Name	Interface	Signal Type	Init Status	Description
Bus2IP_WrReq	User IP	O	0	Active high signal indicating the initiation of a write operation with the IP. It is asserted for 1 Bus2IP_Clk during single data beat transactions and remains high to completion on burst write operations
Bus2IP_RdReq	User IP	O	0	Active high signal indicating the initiation of a read operation with the IP. It is asserted for 1 Bus2IP_Clk during single data beat transactions and remains high to completion on burst read operations.
Bus2IP_RdCE(0: see note 2)	User IP	O	0	Active high chip enable bus. Chip enables are assigned per the User's 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.
Bus2IP_WrCE(0: see note 2)	User IP	O	0	Active high chip enable bus. Chip enables are assigned per the Users 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.

Note:

1. This signal's function and timing is defined in the IBM® **128-Bit Processor Local Bus Architecture Specification Version 4.6.**
2. 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**
3. $\log_2$ represents a logarithm function of base 2. For example, $\log_2(1)=0$, $\log_2(2)=1$, $\log_2(4)=2$, $\log_2(8)=3$, $\log_2(16)=4$, etc.

Design Parameters

The PLBv46 Slave Burst core provides for User interface tailoring via VHDL Generic parameters. These parameters are detailed in Table 2.

The FPGA Family Type parameter is used to select the target FPGA family type. Currently, this design supports Virtex-4, Virtex-5 and Spartan-3 family of devices.

Table 2: **PLBv46 Slave Burst Core Design Parameters**

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
Decoder Address Range Definition				
Array of Base Address / High Address Pairs for each Address Range	C_ARD_ADDR_RANGE_ARRAY	See "Parameter Detailed Descriptions" on page 9	User must set values.	SLV64_ARRAY_TYPE(1)
Array of the desired number of chip enables for each address range	C_ARD_NUM_CE_ARRAY	See "Parameter Detailed Descriptions" on page 9	User must set values.	INTEGER_ARRAY_TYPE(1)
PLB I/O Specification				
PLB Master ID Bus Width	C_SPLB_MID_WIDTH	$\log_2$(**C_SPLB_NUM_MASTERS**) with a minimum value of 1	3	integer
Number of PLB Masters	C_SPLB_NUM_MASTERS	1 to 16	8	integer
Width of the PLB Least Significant Address Bus	C_SPLB_AWIDTH	32	32	integer
Width of the PLB Data Bus	C_SPLB_DWIDTH	32, 64, 128	32	integer
Width of the smallest master that will be interacting with this slave.	C_SPLB_SMALLEST_MASTER	32, 64, 128	32	integer
Slave Attachment I/O Specification				
Width of the Slave Data Bus	C_SIPIF_DWIDTH	32, 64, 128	32	integer
Selects point-to-point or shared plb topology.	C_SPLB_P2P	0 = Shared Bus Topology 1 = Point-to-Point Bus Topology	0	integer
Write Buffer Depth	C_WR_BUFFER_DEPTH	0, 16, 32, or 64 0 = no write buffer implemented	16	integer
Cache Line Addressing Mode	C_CACHLINE_ADDR_MODE[2]	0 = Target word first on reads. 1 = Line word First on reads	0	integer

Table 2: **PLBv46 Slave Burst Core Design Parameters**

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
Burst length type	C_BURSTLENGTH_TYPE	0 = Bus2IP_BurstLength is specified in Bytes 1 = Bus2IP_BurstLength is specified in Data Beats - 1	0	integer
Data Phase Timer Configuration	C_INCLUDE_DPHASE_TIMER	0 = Exclude Data Phase Timeout Timer 1= Include Data Phase Timeout Timer	1	integer
FPGA Family Type				
Xilinx FPGA Family	C_FAMILY	spartan6, spartan3a, aspartan3a, spartan3, aspartan3, spartan3e, aspartan3e, spartan3adsp, aspartan3adsp, virtex4, virtex5, virtex6,	virtex4	string

Note:

1. This Parameter VHDL type is a custom type defined in the ipif_pkg.vhd.
2. Cacheline writes are always line word first.

Allowable Parameter Combinations

Parameter - Port Dependencies

Table 3: **PLBv46 Slave Burst Core Parameter-Port Dependencies**

Name (Generic or Port)	Affects (Port)	Depends (Generic)	Relationship Description
Design Parameters			
C_ARD_ADDR_RANGE_ARRAY	Bus2IP_CS		The vector width of Bus2IP_CS is the number of elements in **C_ARD_ADDR_RANGE_ARRAY/2**
C_ARD_NUM_CE_ARRAY	Bus2IP_WrCE		The vector width of Bus2IP_WrCE is the number of elements in **C_ARD_NUM_CE_ARRAY**
C_ARD_NUM_CE_ARRAY	Bus2IP_RdCE		The vector width of Bus2IP_WrCE is the number of elements in **C_ARD_NUM_CE_ARRAY**
I/O Signals			

Table 3: **PLBv46 Slave Burst Core Parameter-Port Dependencies**

Name (Generic or Port)	Affects (Port)	Depends (Generic)	Relationship Description
Bus2IP_CS		C_ARD_ADDR_ RANGE_ARRAY	The vector width of Bus2IP_CS is the number of elements in **C_ARD_ADDR_RANGE_ARRAY/2**
Bus2IP_WrCE		C_ARD_NUM_CE_ ARRAY	The vector width of Bus2IP_WrCE is the number of elements in **C_ARD_NUM_CE_ARRAY**
Bus2IP_RdCE		C_ARD_NUM_CE_ ARRAY	The vector width of Bus2IP_WrCE 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 PLBV46 Slave Burst is to provide address decoding, Chip Enable/Chip Select control signal generation and write burst buffering.

The PLBv46 Slave Burst core 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 PLBv46 Slave Burst core. 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 PLB Bus 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 PLB Bus 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's entries in the arrays and then automatically add or remove resources, and interconnections based on the arrays' contents. A special case arises when there is a single entry in an unconstrained array. Refer to the section titled "Single Entry in Unconstrained Array Parameters" on page 34 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. It's 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 PLB bus is 32 bits. However, 64 bit values have been allocated for future growth in address bus width.

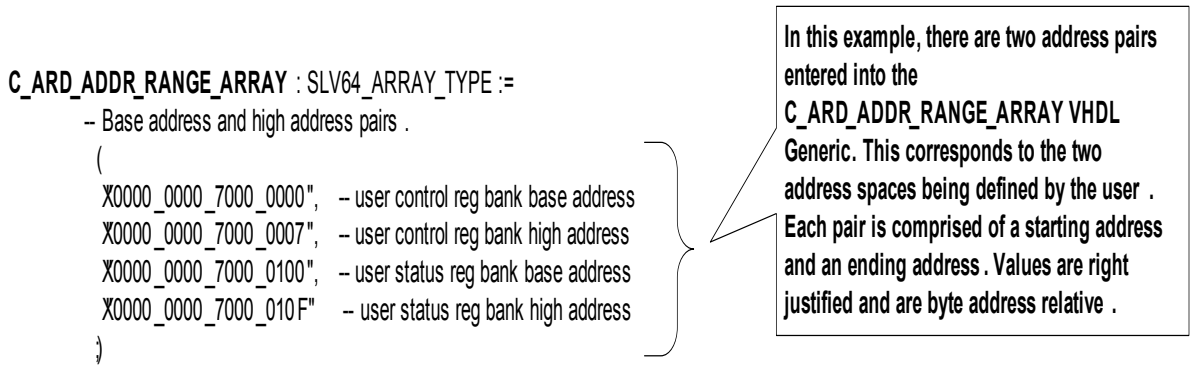

Figure 2: **Address Range Specification Example**

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 (i.e. 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.

If the Slave Attachment is configured for Point-To-Point (C_SPLB_P2P=1) then the number of address pair will affect whether or not the address decode service is instantiated in the Slave Attachment. If only 1 address pair is defined then the address decode service is NOT instantiated. If more than 1 address pair is defined then the address decode service must be instantiated. (See"Point-To-Point Configuration" on page 33.

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.

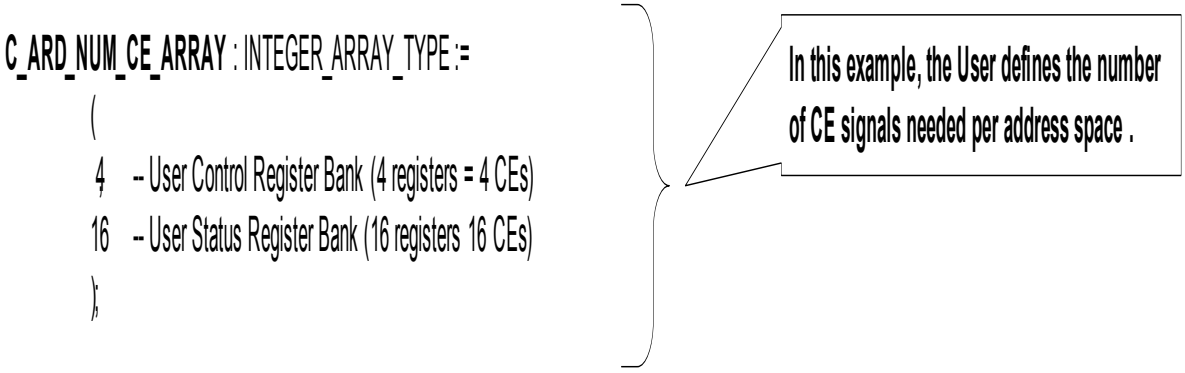

Figure 3: **Chip Enable Specification Example**

C_SPLB_P2P

This parameter is defined as an integer. Setting this parameter to 0 will configure the PLBv46 Slave Burst core for a plb shared bus application. Setting this parameter to 1 will configure the PLBv46 Slave Burst core for a plb point-to-point bus application. In a point-to-point configuration the slave attachment acknowledges all address cycles on the plb and only address decodes based on the number CE's configured for the User IP. This reduces some FPGA resources. Latency is also reduced in a point-to-point configuration. **Note: If more than 1 address range is defined in C_ARD_ADDR_RANGE_ARRAY in a point-to-point configuration (i.e. when C_SPLB_P2P = 1) then the slave attachment will instantiate the address decode logic to distinguish multiple address ranges. This will require a small amount of addition FPGA resources to be used. For the least amount of required FPGA resources only define 1 address pair in C_ARD_ADDR_RANGE_ARRAY when C_SPLB_P2P = 1.** (See"Point-To-Point Configuration" on page 33)

C_SPLB_MID_WIDTH

This parameter is defined as an integer and has a minimum value of 1. It is equal to $\log_2$ of the number of PLB Masters connected to the PLB bus or 1, whichever is greater. It is used to size the PLB_masterID bus input from the PLB Bus to the Slave Attachment. For example, if eight PLB Masters are connected to the PLB Bus, then this parameter must be set to $\log_2(8)$ which is equal to 3. The PLB Bus PLB_masterID bus will be sized to 3 bits wide. If only one master exists, then the parameter needs to be set to 1. Also, when C_SPLB_P2P = 1 set C_SPLB_MID_WIDTH = 1.

C_SPLB_NUM_MASTERS

This parameter is defined as an integer and is equal to the number of Masters connected to the PLB bus. This parameter is used to size the Sl_MBusy and Sl_MErr slave reply buses to the PLB. For example, if eight PLB Masters are connected to the PLB Bus, then this parameter must be set to 8. The Sl_MBusy bus and Sl_MErr bus will be sized to 8 bits wide each. Also, when C_SPLB_P2P = 1 set C_SPLB_NUM_MASTERS = 1.

C_SPLB_SMALLEST_MASTER

This parameter is defined as an integer and is equal to the native data width of the smallest Master connected to the PLB bus that will be accessing the PLBv46 Slave Burst core attachment. This generic is used to generate and optimize steering logic in the slave attachment.

C_SPLB_AWIDTH

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

C_SPLB_DWIDTH

This integer parameter is used by the PLB Slave to size PLB data bus related components within the Slave Attachment. This value should be set to match the actual width of the PLB bus, 32, 64 or 128-Bits.

C_SIPIF_DWIDTH

This integer parameter is used to specify the native data width of the slave attachment. This parameter is used to interface various width Slave devices with various width PLB Buses. Valid values for this parameter are 32, 64, or 128 Bits. **To conserve FPGA resources the value of C_SIPIF_DWIDTH should be set to the same value as C_SPLB_DWIDTH.**

C_WR_BUFFER_DEPTH

This integer parameter specifies the Write Buffer Depth. The slave attachment utilizes a write buffer to allow back-to-back data beat transfers on the PLB Bus even when the User IP is not capable of acknowledging the writes this quickly. This allows the PLB Bus to be freed up for Master transactions to other cores that may be on the PLB Bus. The Write buffer utilizes an SRL16 type fifo and can be set to a depth of 16, 32 or 64. Setting C_WR_BUFFER_DEPTH to 0 will cause the write buffer to **not** be instantiated. This will save FPGA resources for applications that do not require a write buffer or for those applications that instantiate a buffer in the user logic.

NOTE: To meet the timing requirements of the PLB v4.6 bus some of the pipe lining is removed when C_WR_BUFFER_DEPTH = 0. Specifically IP2Bus_WrAck feeds Sl_wrDAck directly and combinatorially feeds Sl_wrComp and Sl_wrBTerm through 1 level of logic. Also PLB_wDBus is combinatorially connected to Bus2IP_Data. This pipeline stage removal could affect frequency adversely for a heavily loaded PLB bus. Therefore it is highly recommended that IP2Bus_WrAck and Bus2IP_Data be registered in applications where a write buffer is not instantiated in the slave attachment.

NOTE: When the write buffer is instantiated Sl_MWrErr is tied to 0, thus Bus2ip_Error has no affect during write transfers. This is due to the posted write nature of the write buffer and the PLB v4.6 requirement that Sl_MWrErr must be driven coincident with Sl_wrDAck.

C_CACHLINE_ADDR_MODE

This integer value determines the address mode used during cache line transfers. The default value of 0, sets the addressing mode to Target Word First for reads and Line Word first for writes. Setting the parameter to 1 sets the addressing mode to Line Word First Mode for both reads and writes. In Target Word First Mode the address presented on the PLB Bus is the first address used in the cache line burst. Subsequent data beats will increment the address from there. In Line Word First Mode the address of the line is used as the starting address.

C_BURSTLENGTH_TYPE

This integer value determines the value type of Bus2IP_BurstLength. When set to 0 (legacy), Bus2IP_BurstLength equals the total number of bytes being requested during a burst. When set to 1, Bus2IP_BurstLength equals the total number of data beats - 1.

C_INCLUDE_DPHASE_TIMER

This integer value determines whether or not the data phase timeout timer is included. Setting this value to a 1 will include the data phase timeout timer and setting this value to a 0 will exclude the data phase timeout timer.

C_FAMILY

This parameter is defined as a string. It specifies the target FPGA technology for implementation of the PLB 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.

IP Interconnect (IPIC) Signal Description

Bus2IP_Addr

Bus2IP_Addr is a 32 Bit vector that drives valid when Bus2IP_CS and Bus2IP_RdCE or Bus2IP_WrCE drives high. For burst type transfers, the User IP must assert IP2Bus_AddrAck to advance the address on the IPIC.

Bus2IP_Data

Bus2IP_Data is a vector of width C_SIPIF_DWIDTH and drives valid on writes when Bus2IP_WrCE or Bus2IP_WrReq drive 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_SIPIF_DWIDTH/8. This vector indicates which bytes are valid on Bus2IP_DATA. Bus2IP_BE becomes valid coincident with Bus2IP_CS. During burst type transfers Bus2IP_BE will change with the IP2Bus_WrAck or IP2Bus_RdAck.

Bus2IP_Burst

Bus2IP_Burst indicates that the current cycle is a burst cycle when asserted or a single beat transfer when not asserted. Bus2IP_Burst will assert coincident with Bus2IP_CS and will remain asserted until the second to last data is acknowledge by the User IP with a IP2Bus_RdAck or IP2Bus_WrAck as the case may be.

Bus2IP_BurstLength

Indicates in bytes the length of the burst. The burst length is valid when Bus2IP_CS is active.

Bus2IP_WrReq

Bus2IP_WrReq indicates when write data is valid on Bus2IP_Data bus. Bus2IP_WrReq has a unique feature in that for single beat write transfers, Bus2IP_WrReq will assert for 1 SPLB_Clk cycle regardless on whether the User IP acknowledges the cycle or not. For burst cycles Bus2IP_WrReq will remain asserted for the entire burst sequence.

Bus2IP_RdReq

Bus2IP_RdReq indicates the requested cycle is a read type transfer. Bus2IP_RdReq will assert for 1 SPLB_Clk cycle for single beat read transfers coincident with Bus2IP_CS and Bus2IP_RdCE. For burst

cycles, Bus2IP_RdReq will assert for each valid address of the burst cycle. For example if a 4 data beat burst read is being requested and the UserIP drives IP2Bus_AddrAck ahead of IP2Bus_RdAck then Bus2IP_RdReq will deassert after the 4 address is acknowledged which will be before the 4th data beat is acknowledged. See "Leading IP2Bus_AddrAck Operation" on page 25 and Figure 14.

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 CE's 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 CE's 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_SIPIF_DWIDTH 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 de-assert. This signal should only be driven high during read cycles.

Note: During read cycles if the User IP does not acknowledge the read request within 128 SPLB_Clk cycles the slave attachment will timeout, de-assert the request to the User IP (i.e. de-assert Bus2IP_CS and Bus2ip_RdCE) and complete the read cycle on the PLB by driving sl_rddack with sl_rddbus equaling all zeros.

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, Bus2IP_CS to de-assert. This signal should only be driven high during write cycles.

Note: During write cycles if the User IP does not acknowledge the write request within 128 SPLB_Clk cycles the slave attachment will timeout, de-assert the request to the User IP (i.e. de-assert Bus2IP_CS and Bus2ip_WrCE) and complete the write cycle on the PLB by driving sl_wrdack.

IP2Bus_Error

IP2Bus_Error is used by the User IP to indicate and error has occurred. This signal is only sampled with IP2Bus_WrAck or IP2Bus_RdAck and is ignored all other times. This signal should only be driven during read or write cycles.

IPIC Transaction Timing

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

32 Bit Master Single Beat Read Operation

(32 Bit Master, 32 Bit Slave, C_SPLB_P2P = 0)

The PLB V4.6 Slave Attachment supports single beat transfers of bytes, half-words, words, double-words, and quad words. Figure 4 shows two single beat read requests from a 32-Bit Master (PLB_mSize=00b) to the 32-Bit Slave (Sl_SSize = 00b).

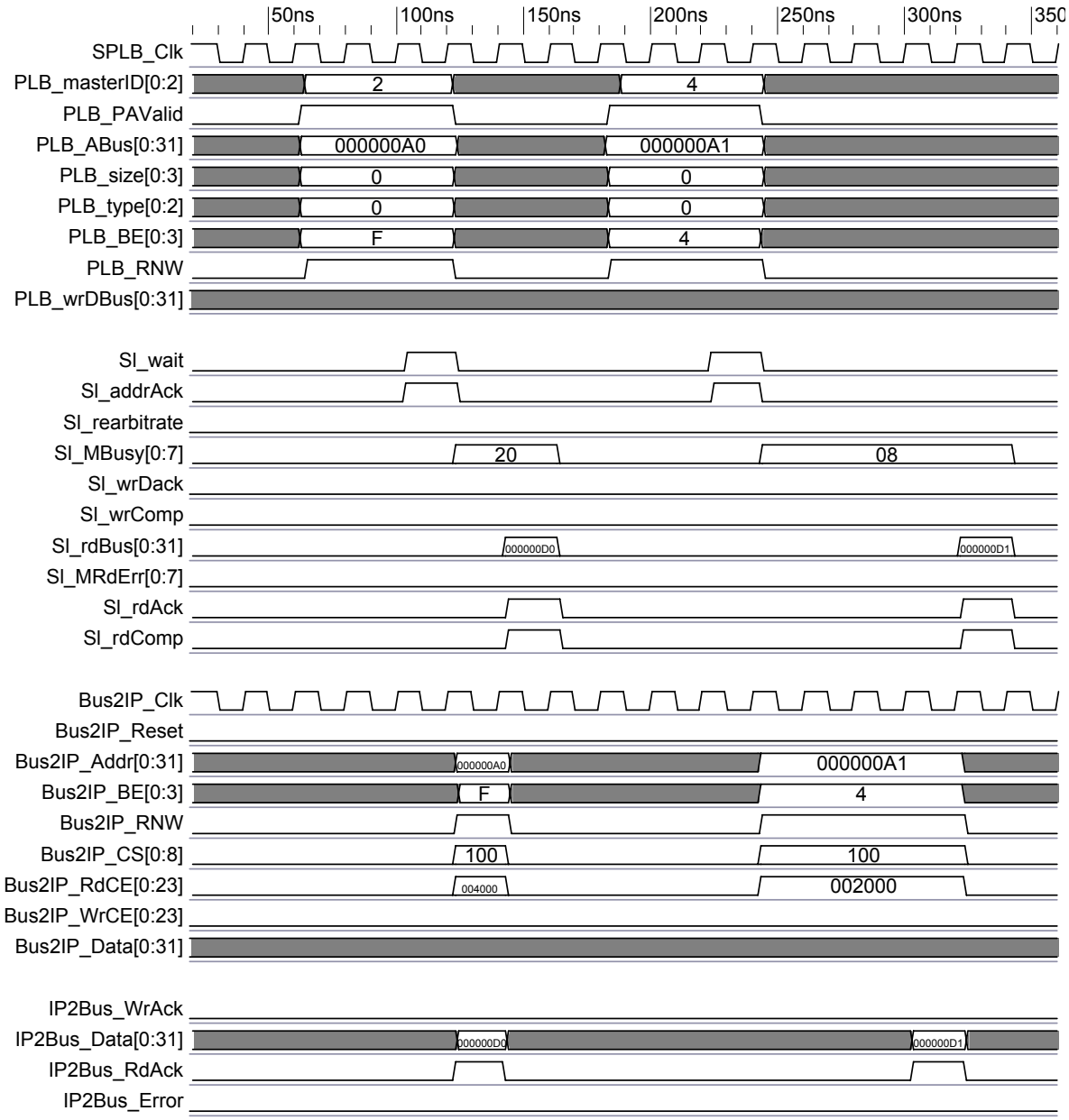

Figure 4: **32 Bit Master Single Beat Read Timing (C_SPLB_P2P=0)**

32 Bit Master Single Beat Write Operation

(32 Bit Master, 32 Bit Slave, C_SPLB_P2P = 0)

The PLB V4.6 Slave Attachment supports single beat transfers of bytes, half-words, words, double-words, and quad words. Figure 5 shows two single beat write requests from a 32-Bit Master (PLB_mSize=00b) to the 32-Bit Slave (Sl_SSize = 00b).

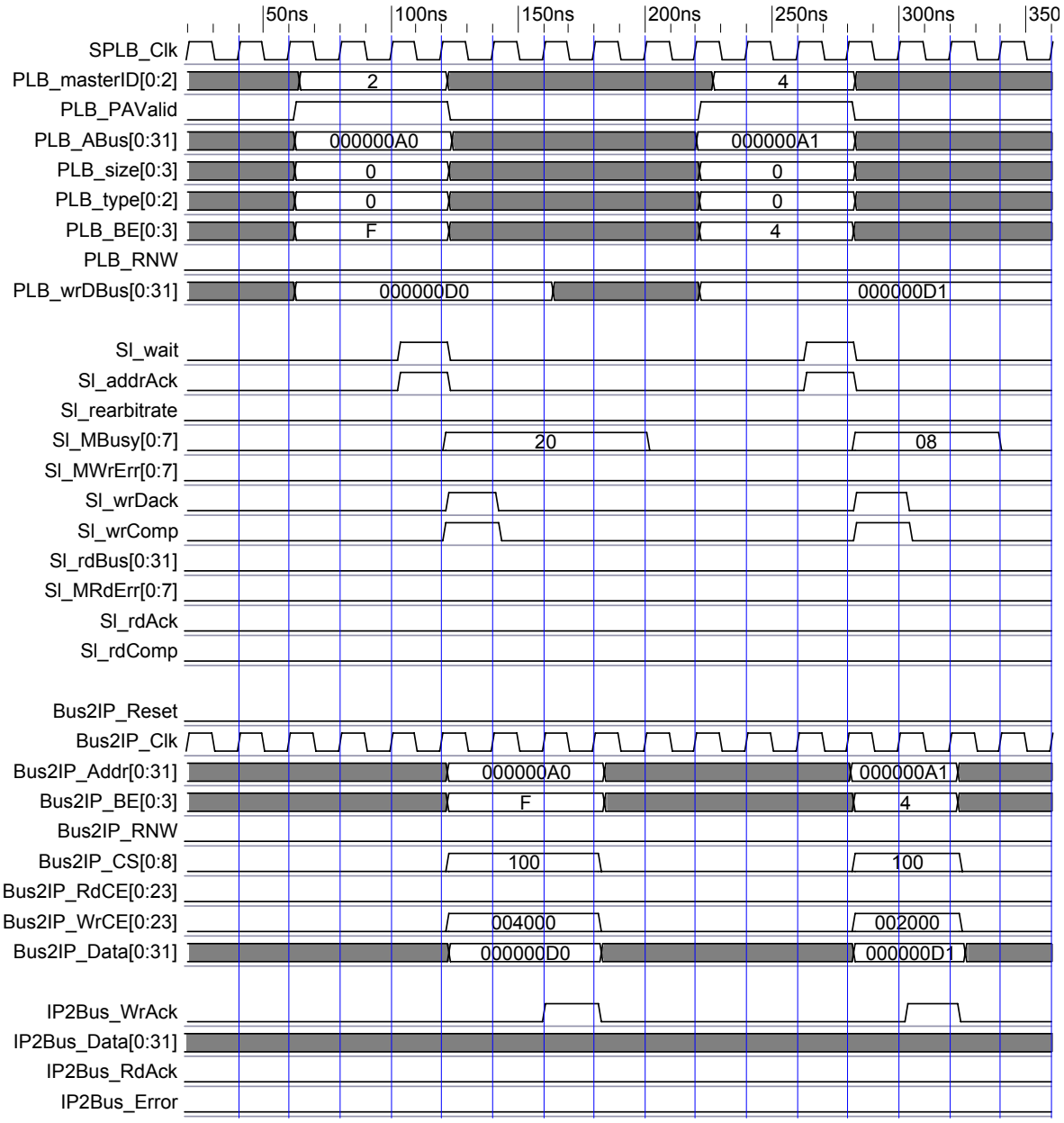

Figure 5: **32 Bit Master Single Beat Write Timing (C_SPLB_P2P=0)**

32 Bit Master Fixed Length Operation

(32 Bit Master, 32 Bit Slave, C_SPLB_P2P = 0)

The PLB V4.6 Slave Attachment supports fixed bursts of words, double-words, and quad words with transfer lengths of 2 to 16 data beats. Figure 6 shows a burst request from a 32-Bit Master (PLB_mSize=00b) to the 32-Bit Slave (Sl_SSize = 00b).

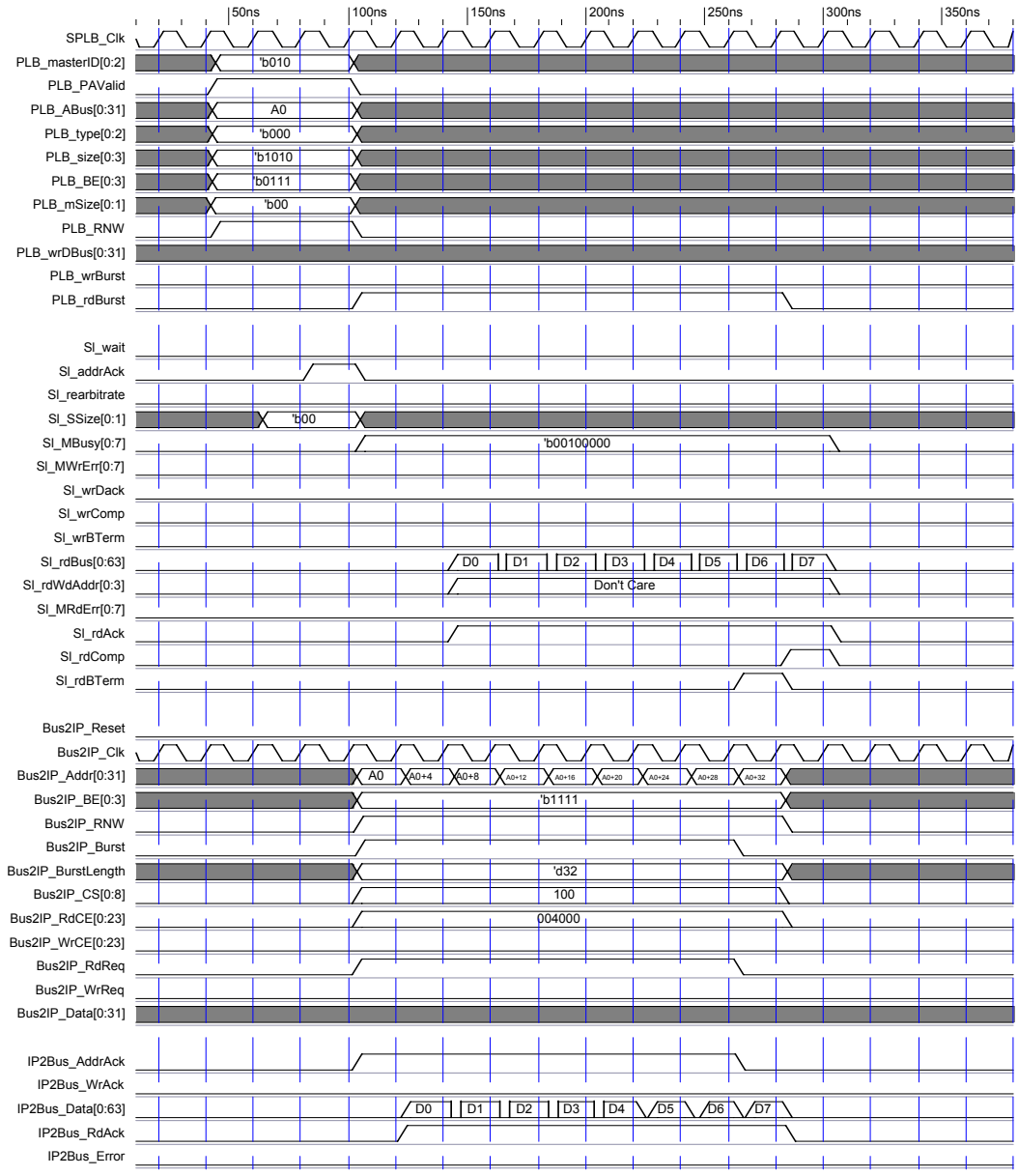

Figure 6: **32 Bit Master Fixed Burst Read Timing (C_SPLB_P2P=0)**

64 Bit Master Fixed Length Burst Read Operation

(64 Bit Master, 32 Bit Slave, C_SPLB_P2P = 0)

Figure 7 shows a double-word burst request from a 64-Bit Master (PLB_mSize=01b) to the 32-Bit Slave (Sl_SSize = 00b). Because the slave is only 32-Bits wide and can only transfer 32-bit words, the slave attachment is required to double the number of data beats returned. For example the master is requesting 3 double-words and then the slave will respond with 2*3 = 6 words.

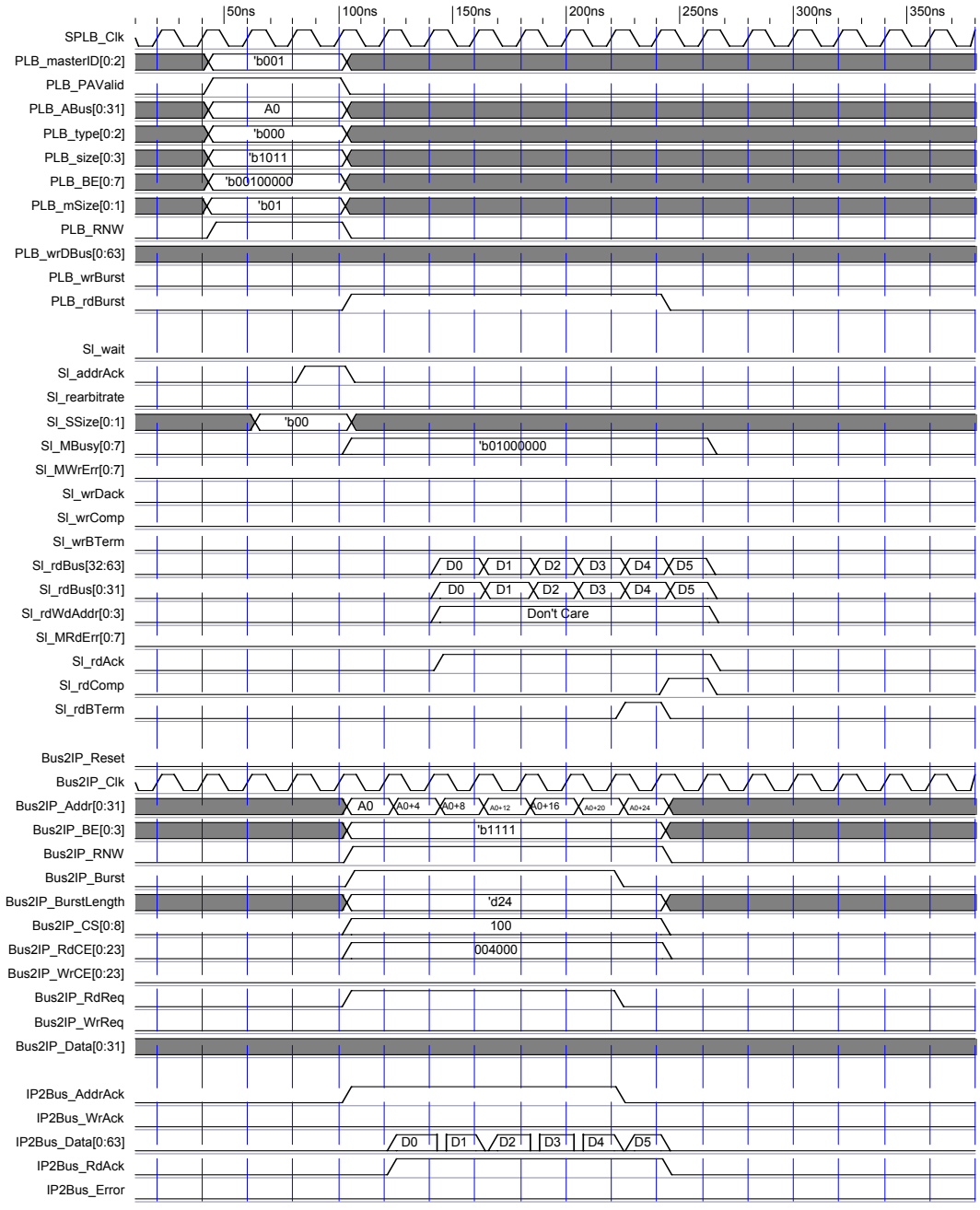

Figure 7: **64 Bit Master Fixed Burst Read Timing (C_SPLB_P2P=0)**

32 Bit Master Fixed Length Burst Write Operation - Write Buffer

(32 Bit Master, 32 Bit Slave, Write Buffer Included, C_SPLB_P2P = 0)

Figure 8 shows a burst write request from a 32-Bit Master (PLB_mSize=00b) to the 32-Bit Slave (Sl_SSize = 00b). Due to the write buffer being instantiated there is a 2 clock latency from the assertion of Bus2IP_CS and the write data, Bus2IP_Data, being valid on the IPIC interface. Bus2IP_WrReq, Bus2IP_WrCE, and Bus2ip_Burst will assert when the write data is valid. Also note that the write data is acknowledge on the PLB bus (Sl_wrDack=1) immediately following Sl_addrack, regardless of the User IP data acknowledge. Write data on the PLB bus side is acknowledge as the data is written into the write buffer.

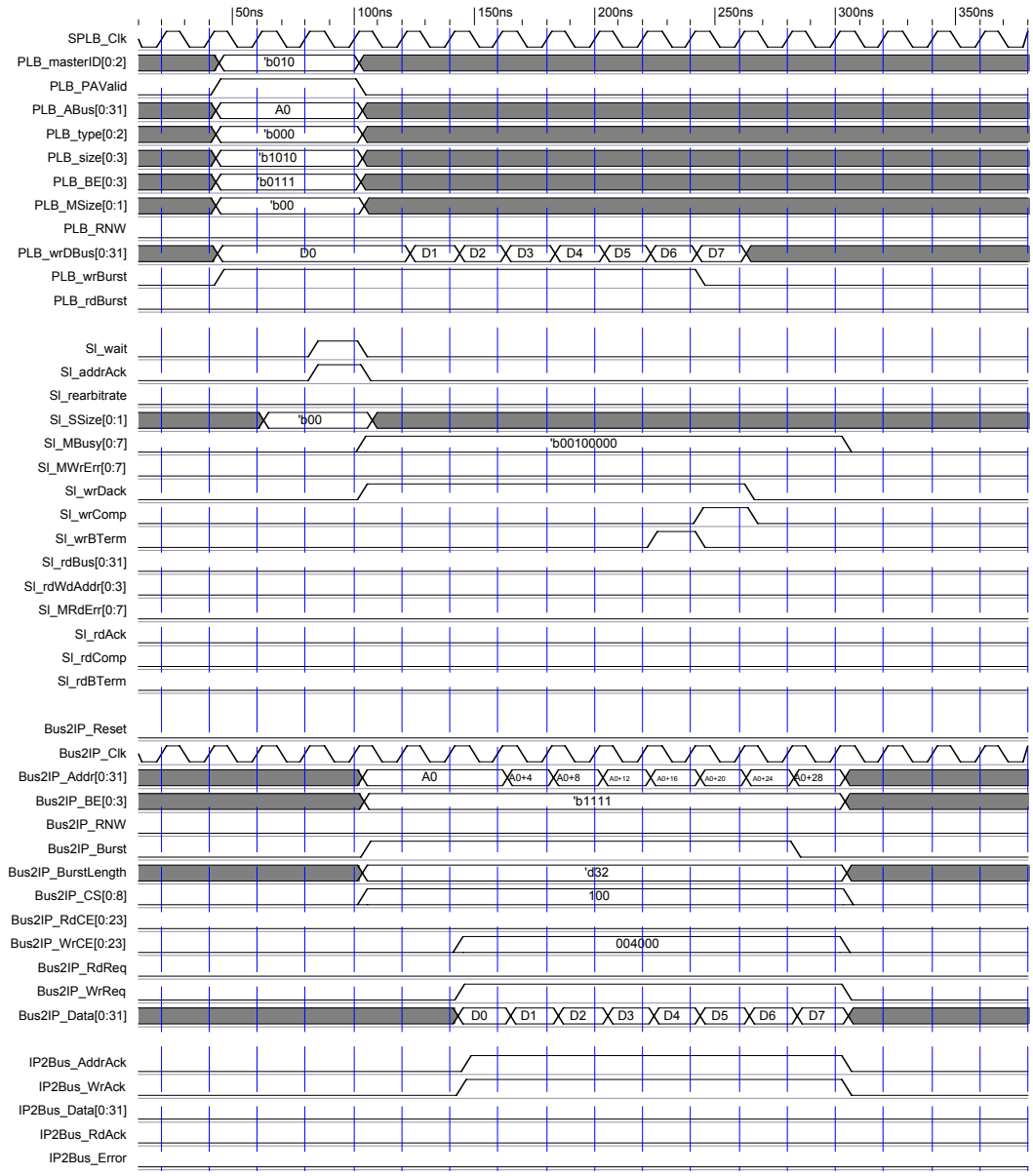

Figure 8: **32 Bit Master Fixed Burst Write Timing (Write Buffer, C_SPLB_P2P=0)**

32 Bit Master Fixed Length Burst Write Operation - No Write Buffer, Fast User IP

(32 Bit Master, 32 Bit Slave, No Write Buffer Included, C_SPLB_P2P = 0)

Figure 9 shows a burst write request from a 32-Bit Master (PLB_mSize=00b) to the 32-Bit Slave (Sl_SSize = 00b). For the case where the write buffer is not instantiated, latency is reduced between a write cycle start on the PLB bus and valid data and control asserts on the IPIC interface. It should be noted that Sl_wrDack acknowledges the first data beat during Sl_addAck to fill the 1 register pipe stage with data. There Sl_wrDack follows Bus2IP_WrAck. What this means is that the PLB bus will be held up until the User as accepted all of the write data. So if the User IP throttles the data on the IPIC side the cycle will be throttled on the PLB bus. Also note that Sl_wrDack is combinatorially based on Bus2IP_WrAck. It is recommended that Bus2IP_WrAck be registered by the User IP to help prevent long timing paths.

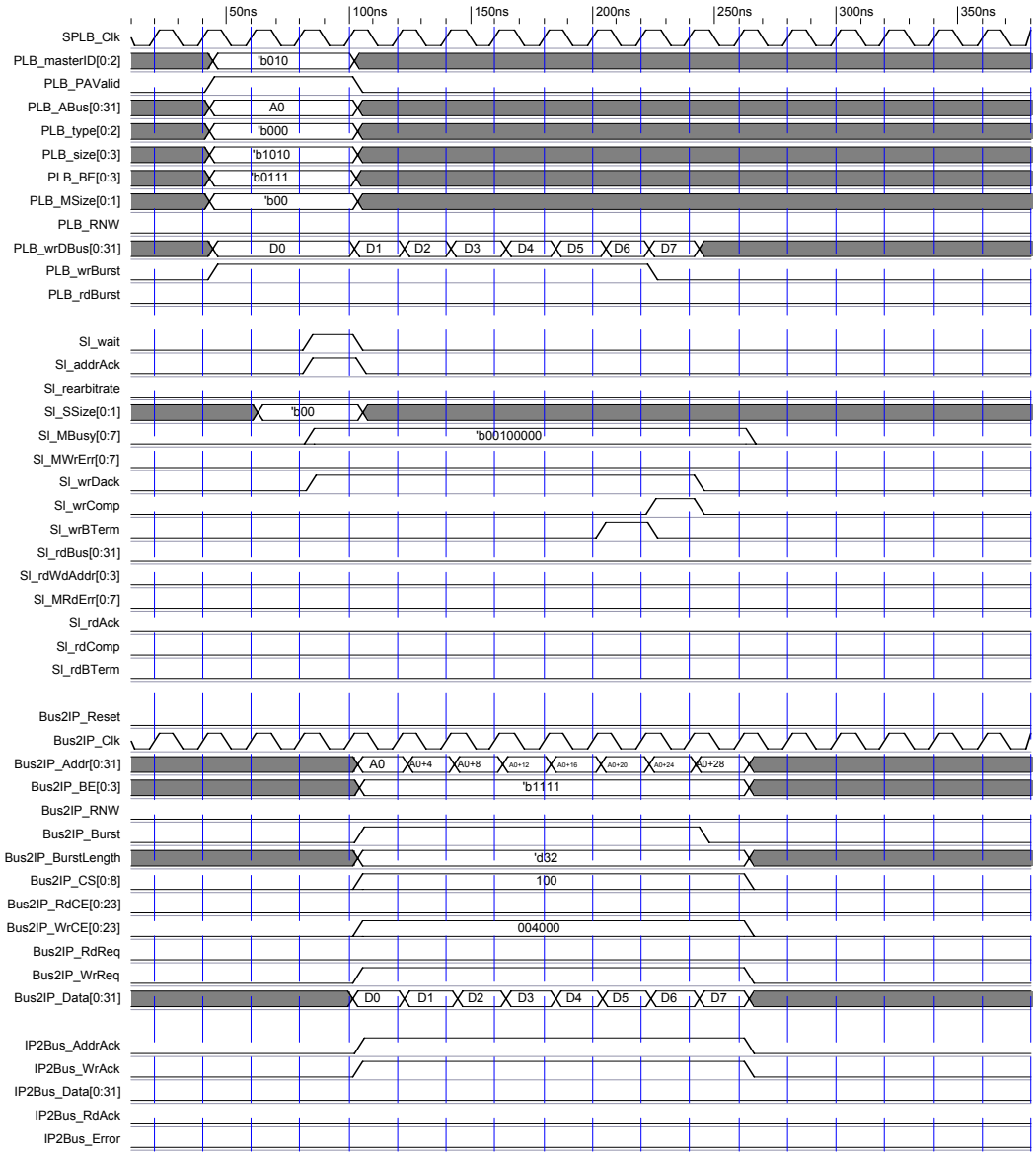

Figure 9: **32 Bit Master Fixed Burst Write Timing (No Write Buffer, Fast, C_SPLB_P2P=0)**

64 Bit Master Fixed Length Burst Write Operation - Write Buffer, Early Burst Terminate

(64 Bit Master, 64 Bit Slave, Write Buffer Included, C_SPLB_P2P = 0)

Figure 10 shows a burst write request from a 64-Bit Master (PLB_mSize=01b) to the 64-Bit Slave (Sl_sSize = 01b). In this case the Master is requesting a fixed burst of length of 24 (PLB_BE(0:7) = 00010111b) double words which is greater than the maximum of 16 data beats. The Slave Attachment will automatically terminate the burst at 16 data beats.

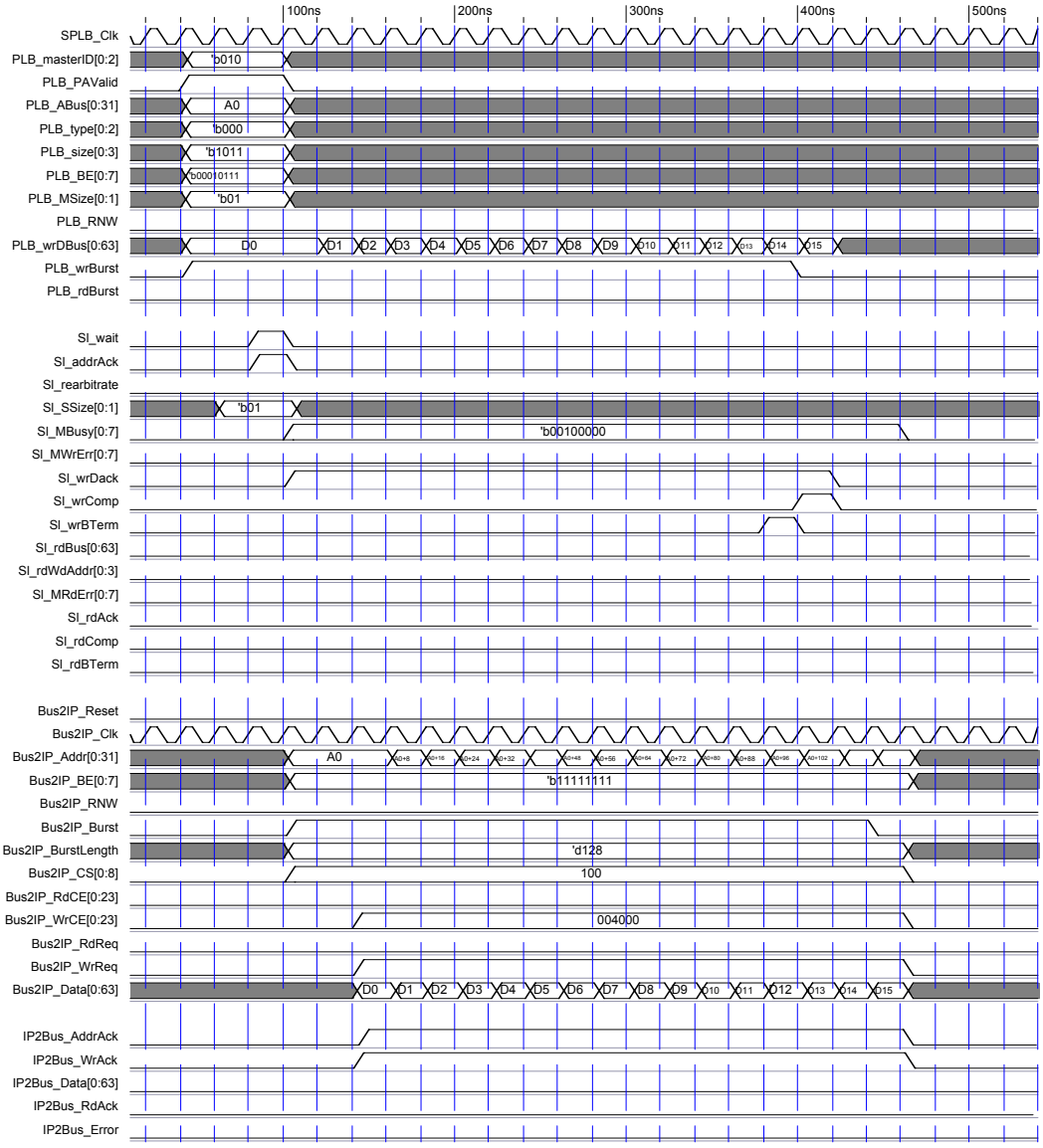

Figure 10: **64 Bit Master Early Terminated Fixed Burst Write Timing (Write Buffer, Fast, C_SPLB_P2P=0)**

32 Bit Master Fixed Length Burst Write Operation - No Write Buffer, Slow User IP

(32 Bit Master, 32 Bit Slave, No Write Buffer Included, C_SPLB_P2P = 0)

Figure 11 shows a burst write request from a 32-Bit Master (PLB_mSize=00b) to the 32-Bit Slave (Sl_SSize = 00b). The figure illustrates a User IP that is not capable of accepting the write data on each clock cycle. This then causes throttling on the PLB bus.

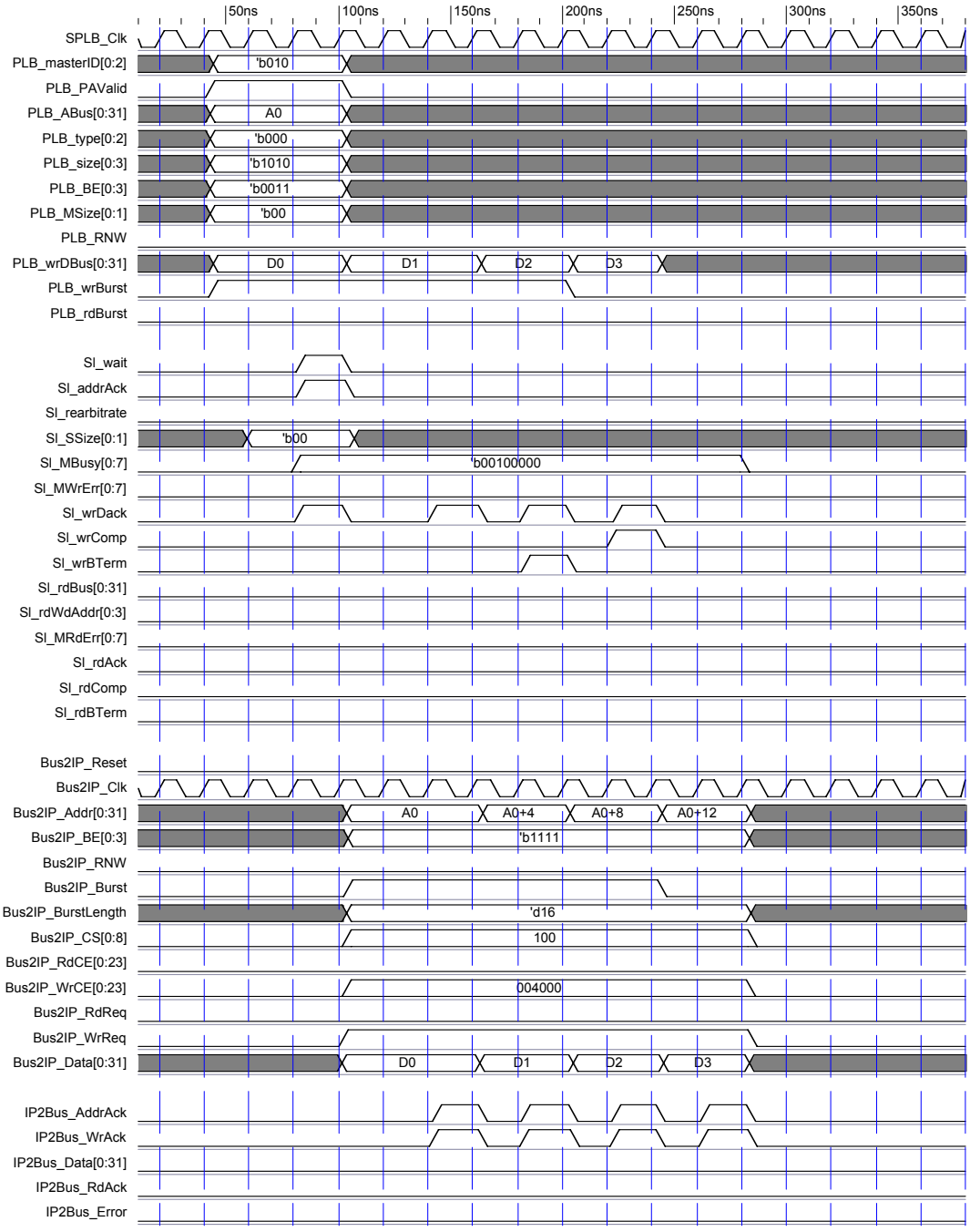

Figure 11: **32 Bit Master Fixed Burst Write Timing (No Write Buffer, Slow, C_SPLB_P2P=0)**

8-Word Cacheline Read Operation

(64 Bit Master, 64 Bit Slave, C_SPLB_P2P = 0)

Figure 11 shows a 8-Word Cacheline read request from a 64-Bit Master (PLB_mSize=01b) to the 64-Bit Slave (Sl_SSize = 01b).

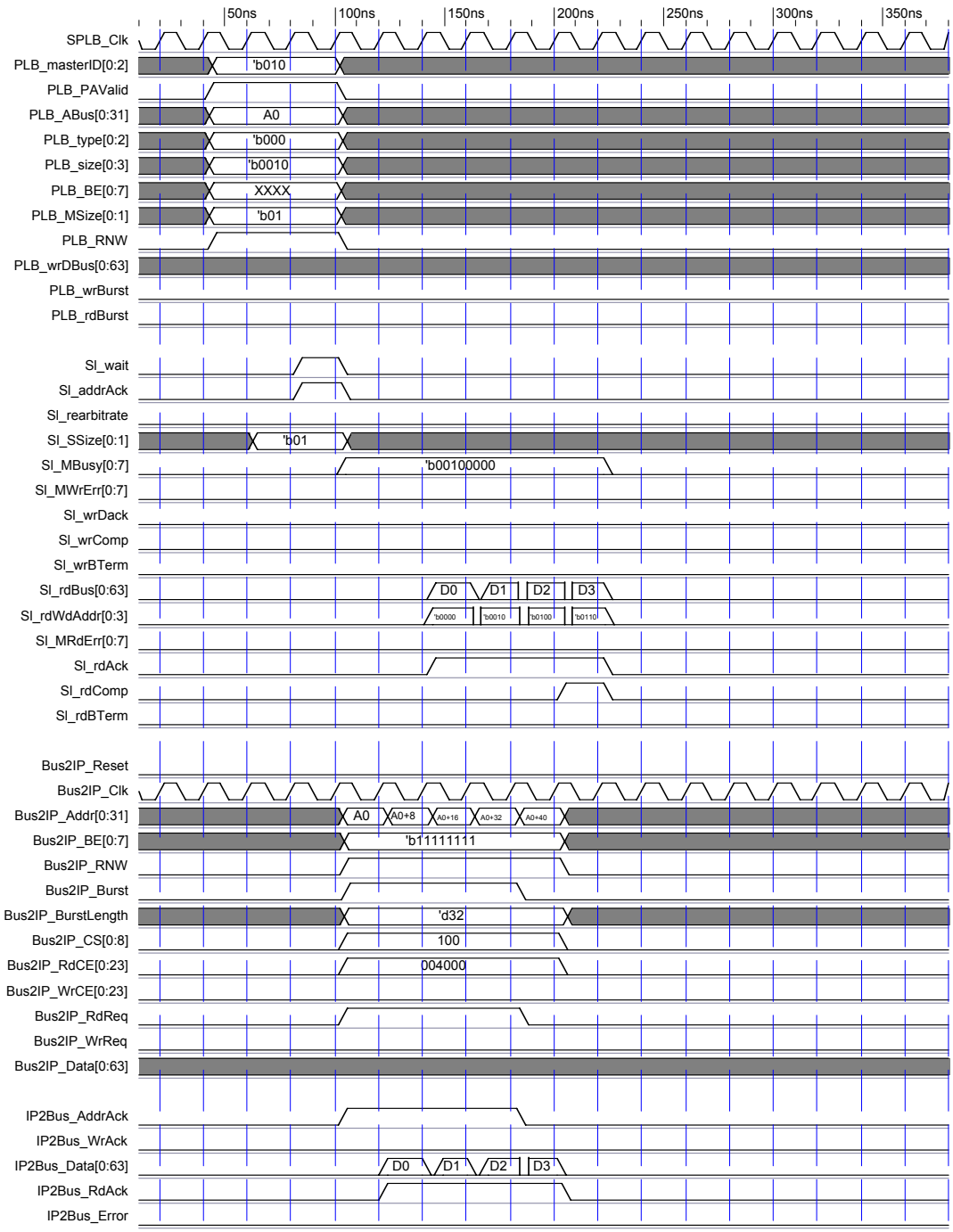

Figure 12: **8-Word Cacheline Read Timing (C_SPLB_P2P=0)**

8-Word Cacheline Write Operation - Write Buffer

(64 Bit Master, 64 Bit Slave, Write Buffer, C_SPLB_P2P = 0)

Figure 11 shows a 8-Word Cacheline write request from a 64-Bit Master (PLB_mSize=01b) to the 64-Bit Slave (Sl_SSize = 01b).

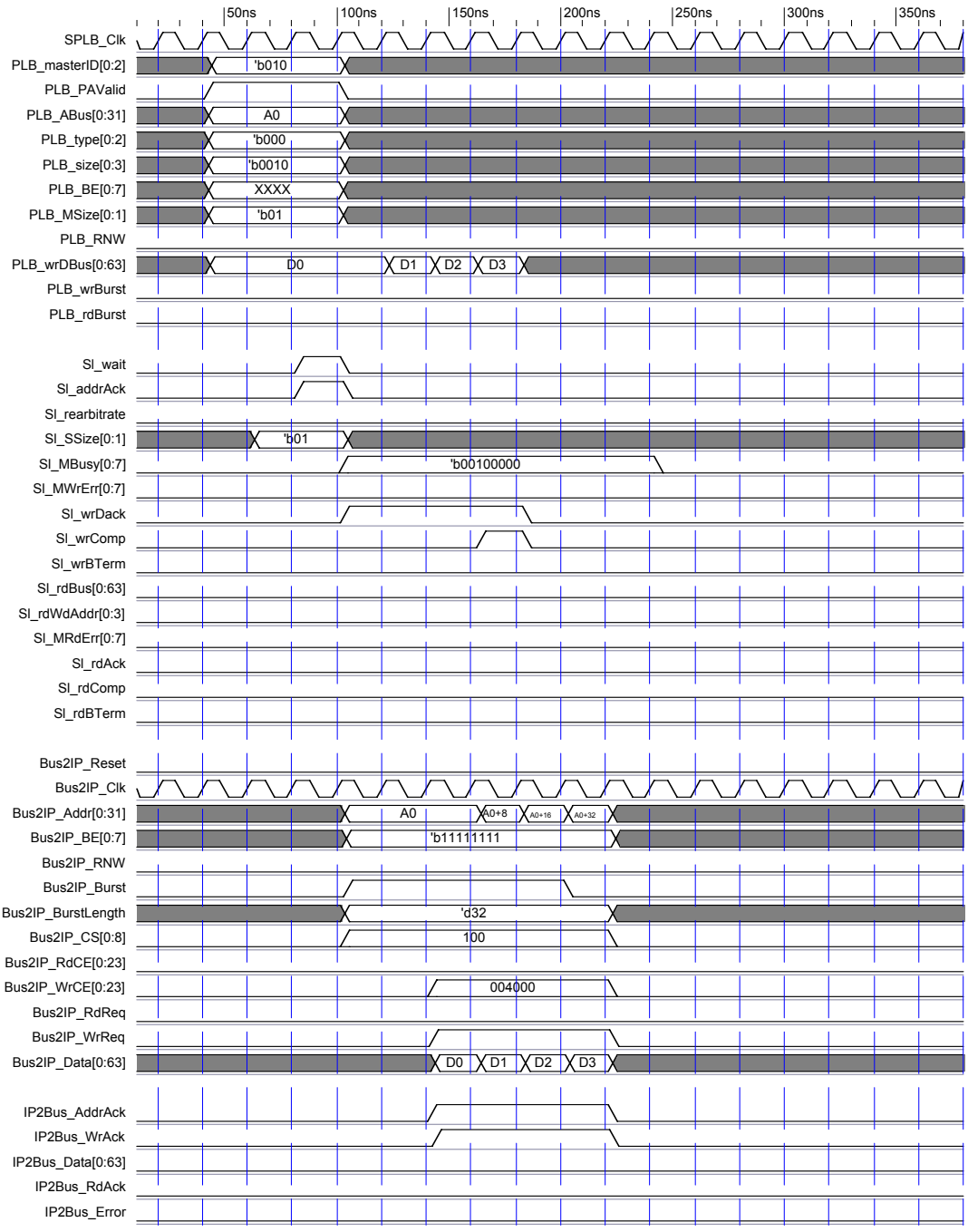

Figure 13: **8-Word Cacheline Write Timing (Write Buffer, C_SPLB_P2P=0)**

Leading IP2Bus_AddrAck Operation

(64 Bit Master, 64 Bit Slave, C_SPLB_P2P = 0)

IP2Bus_AddrAck allows the User IP to control the advance of the Slave Attachment's address counter and control state independent of the data acknowledge assertion. Figure 14 shows a burst read request from a 64-Bit Master (PLB_mSize=01b) to the 64-Bit Slave (Sl_SSize = 01b) where the User IP is advancing the address ahead of the data acknowledge by 2 clocks. This is done to optimize the burst transfer by keeping fresh data moving in the data pipeline.

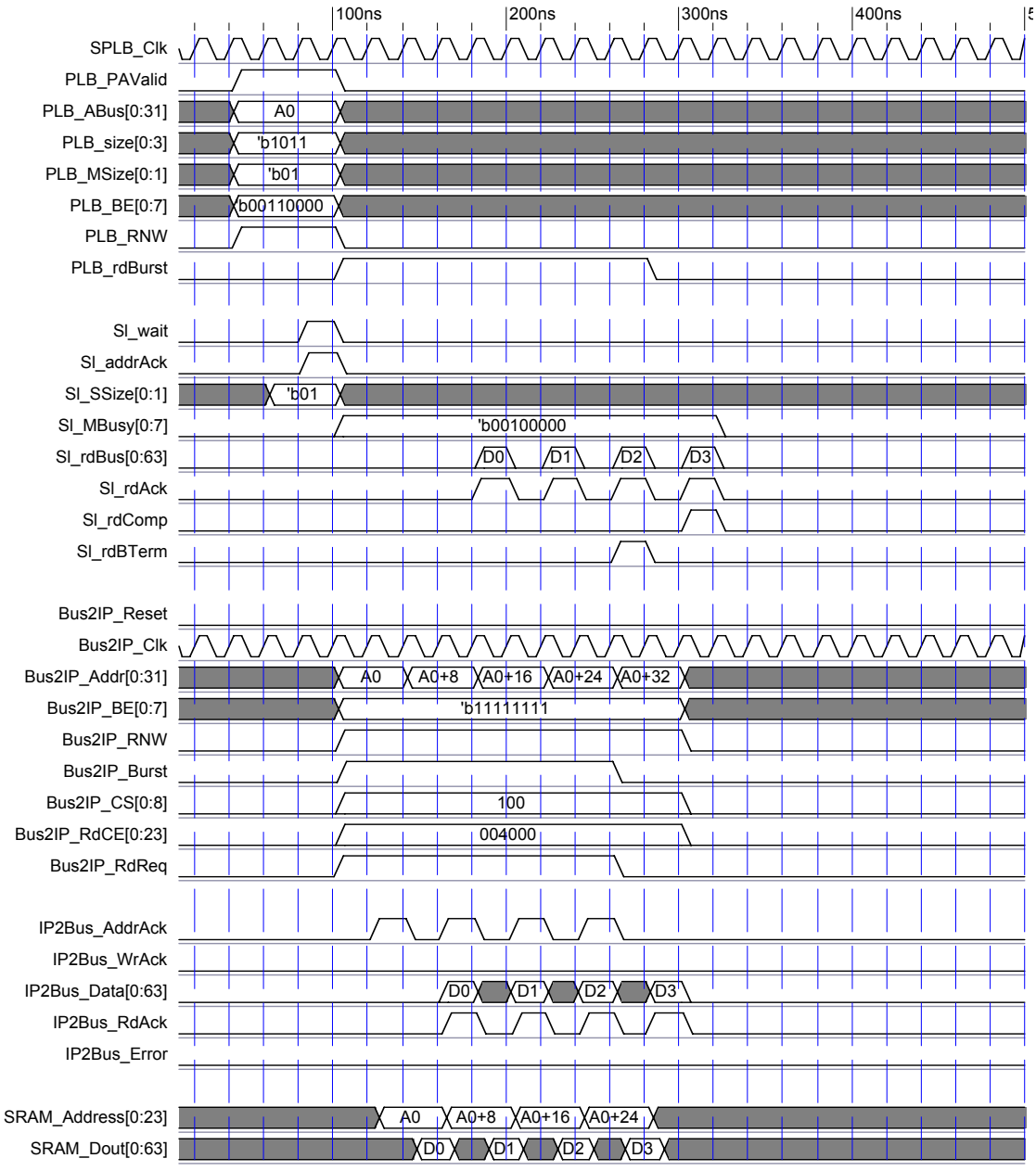

Figure 14: **Leading IP2Bus_AddrAck Timing (C_SPLB_P2P=0)**

Point-to-Point Single Beat Read Operation

(32 Bit Master, 32 Bit Slave, C_SPLB_P2P = 1)

Figure 15 shows a back to back read operation in a point to point configuration. The PLBv46 Slave Burst core Sl_AddrAck's the read cycle immediately with the assertion of PLB_PAValid. The second read cycle is Sl_AddrAck'ed in the next clock cycle after Sl_rdComp.

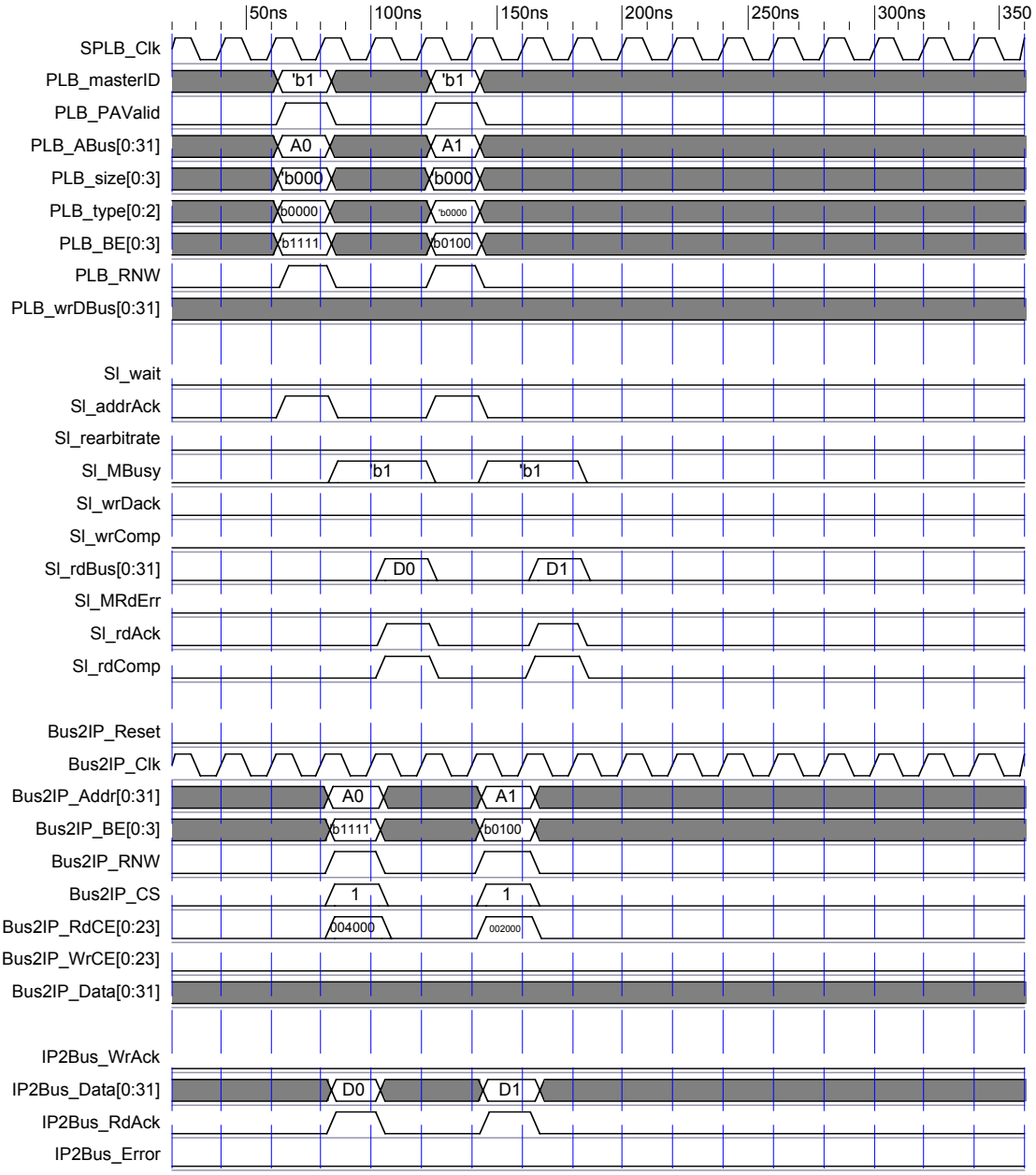

Figure 15: **Point-to-Point Single Beat Read Timing (C_SPLB_P2P=1)**

Point-to-Point Single Beat Write Operation - No Write Buffer

(32 Bit Master, 32 Bit Slave, No Write Buffer, C_SPLB_P2P = 1)

Figure 16 shows back to back single beat write operations in a point to point configuration. The PLBv46 Slave Burst core Sl_AddrAck's the write cycle immediately with the assertion of PLB_PAValid. If the User IP Acknowledges the write, IP2Bus_WrAck as soon as the cycle qualifiers (Bus2IP_CS, Bus2IP_WrCE, etc) assert then Sl_wrDack and Sl_wrComp will assert in the clock cycle following Sl_AddrAck allowing for the next cycle to be acknowledge 1 cycle later.

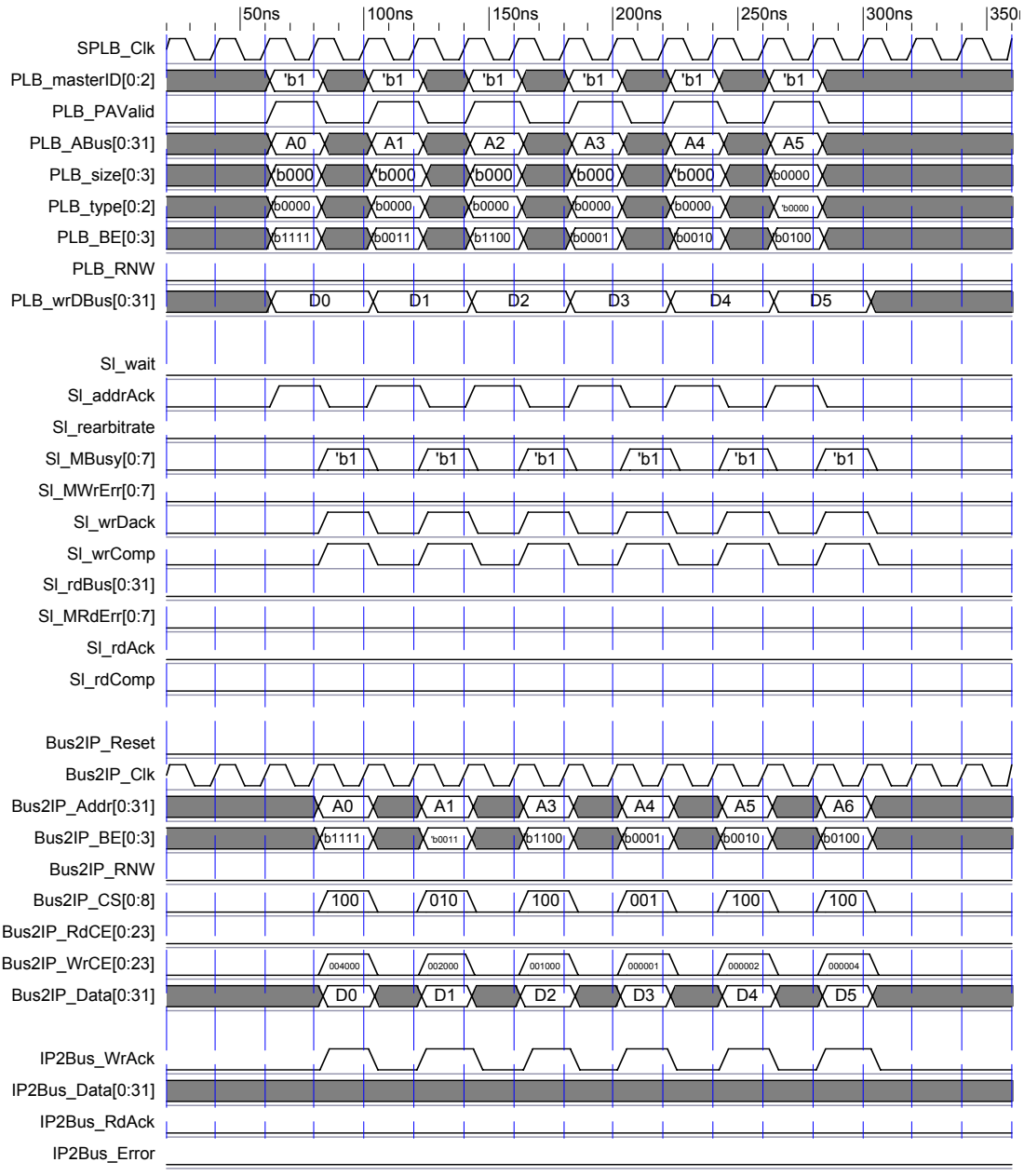

Figure 16: **Point-to-Point Single Beat Write Timing (No Write Buffer, C_SPLB_P2P=1)**

Point-to-Point Fixed Burst Read/Write Operation - No Write Buffer

(32 Bit Master, 32 Bit Slave, No Write Buffer, C_SPLB_P2P = 1)

Figure 17 shows a back to back write to read operation in a point to point configuration. The PLBv46 Slave Burst core Sl_AddrAck's the write cycle immediately with the assertion of PLB_PAValid. The read cycle is Sl_AddrAck'ed in the next clock cycle after Sl_wrComp.

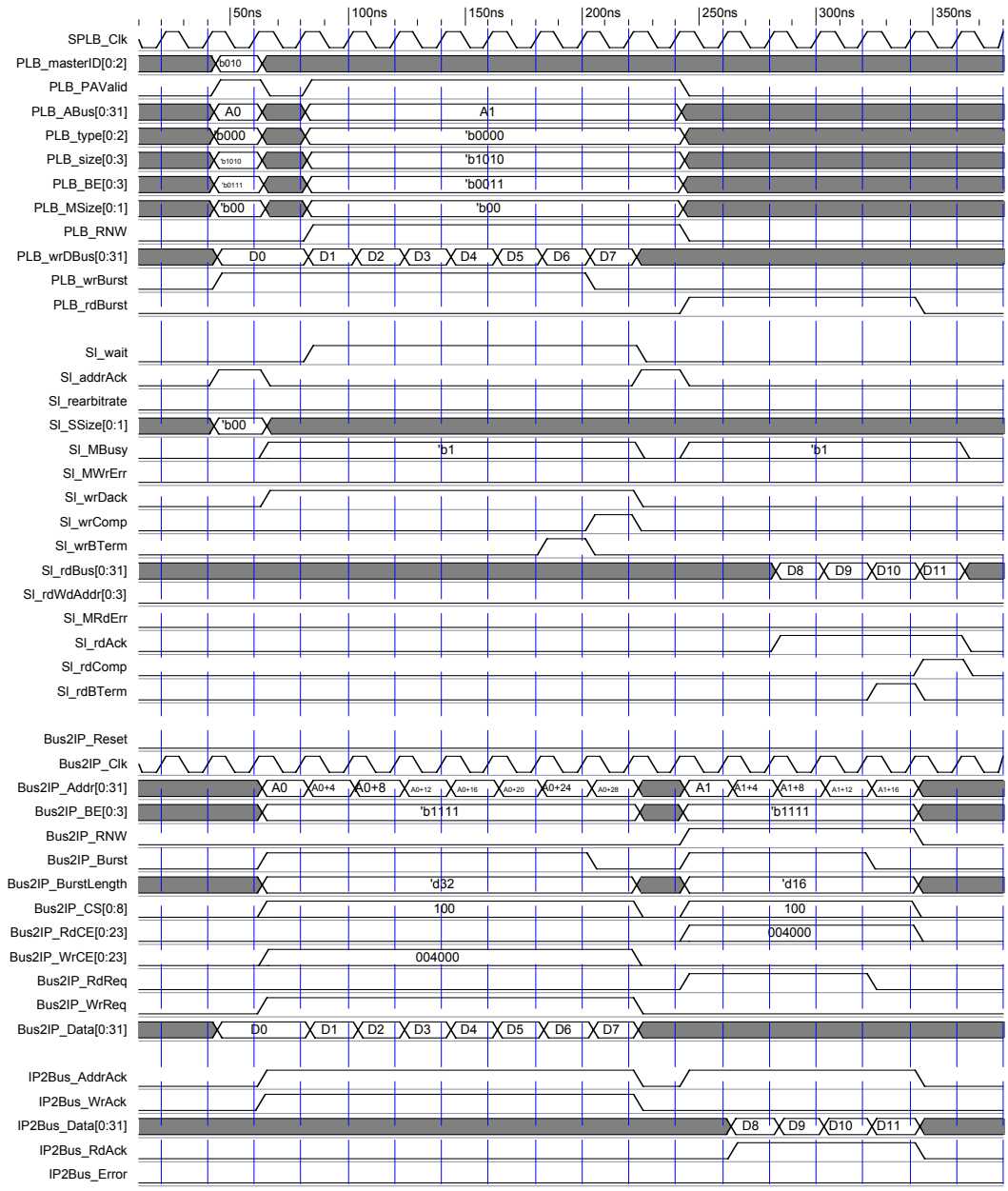

Figure 17: **Point-to-Point Fixed Burst Read/Write Timing (No Write Buffer, C_SPLB_P2P=1)**

Point-to-Point Fixed Burst Read/Write Operation - Write Buffer

(32 Bit Master, 32 Bit Slave, Write Buffer, C_SPLB_P2P = 1)

Figure 18 shows a back to back write to read operation in a point to point configuration with the write buffer instantiated. The PLBv46 Slave Burst core Sl_AddrAck's the write cycle immediately with the assertion of PLB_PAValid. The read cycle is Sl_AddrAck'ed in the cycle following the completion of the write transaction on the IPIC.

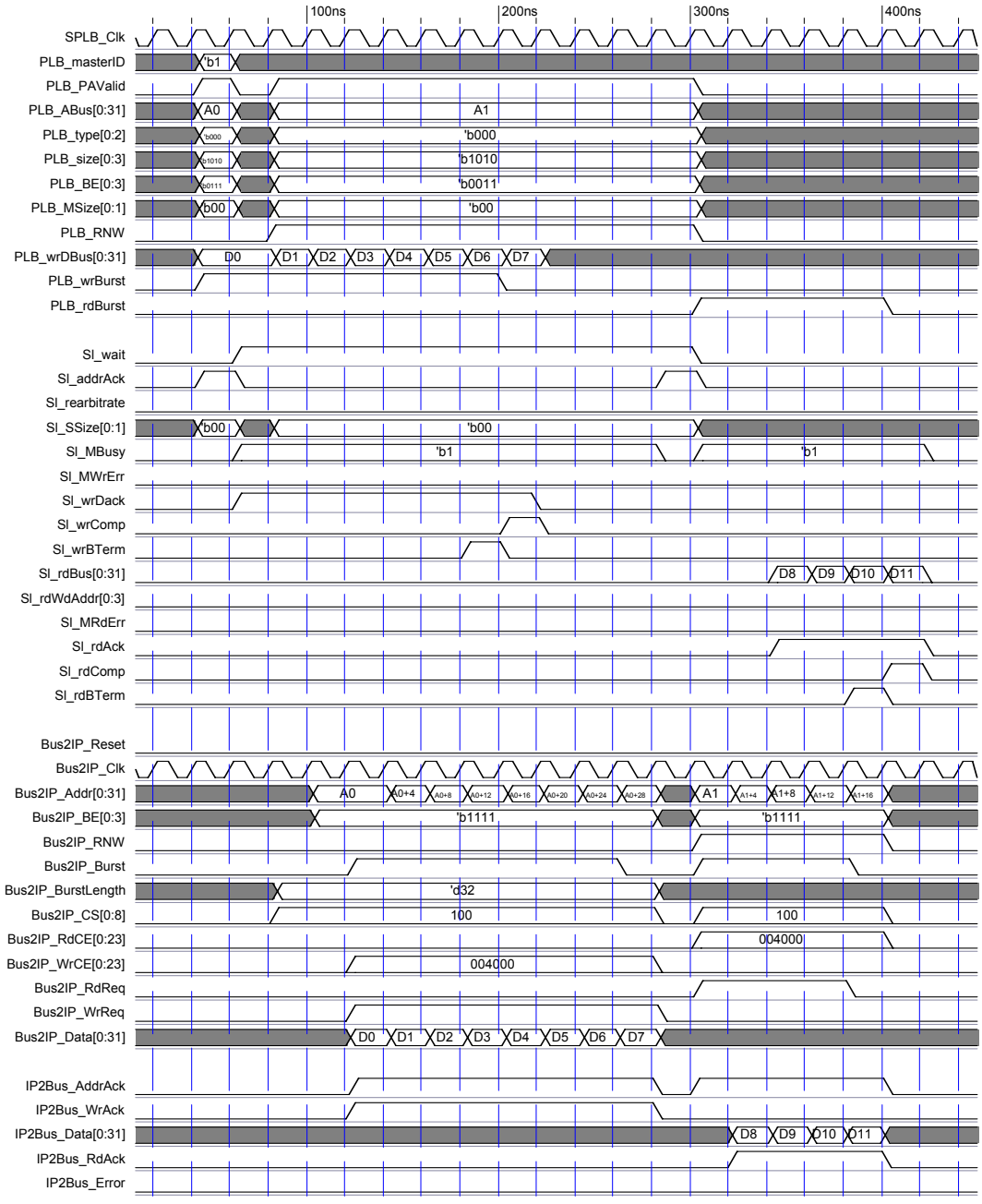

Figure 18: **Point-to-Point Fixed Burst Read/Write Timing (Write Buffer, C_SPLB_P2P=1)**

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 PLB Slave Attachment 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 19. The timing characteristics of these signals are shown in the section titled "IPIC Transaction Timing" on page 15. 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(0:n)

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 of the 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(0:y)

Bus2ip_RdCE is the chip enable bus for read transactions. Each address space defined in the ARD arrays are allowed to have 1 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 PLB Bus 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(0:y)

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.

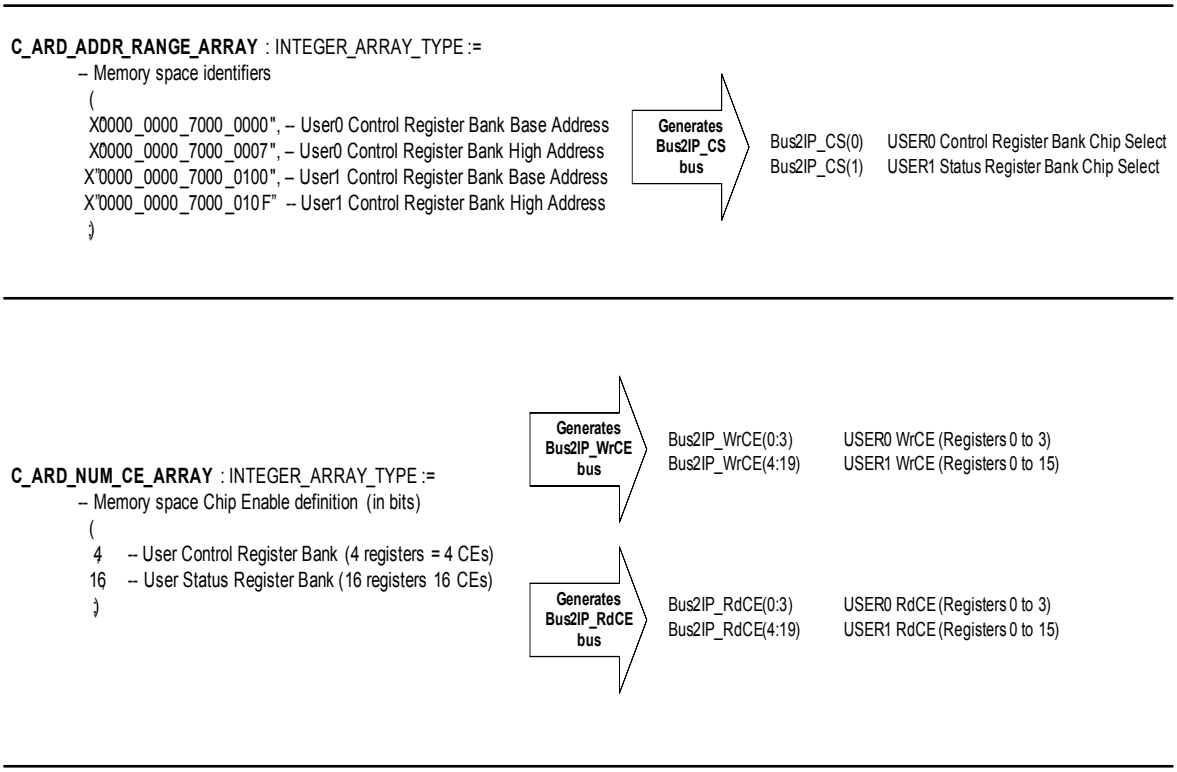

Figure 19: **ARD Arrays and CS/CE Relationship Example**

Available Support Functions for Automatic Ripping of CE and CS Buses.

The User may find it convenient to use some predefined functions developed by Xilinx to automatically rip signals from the Bus2IP_CS, Bus2IP_WrCE, and Bus2IP_RdCE buses. These functions facilitate bus ripping regardless of order or composition 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 on page 32 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_v2_00_a\hdl\vhdl\ipif_pkg.vhd.

The following library declaration must appear in the User's VHDL source:

library proc_common_v2_00_a;

use proc_common_v2_00_a.ipif_pkg.all;

An example of how these functions are used is shown in Figure 20.

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 rip 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(0 to NUM_USER_01_CE -1);
    signal  user_01_wrce                : std_logic_vector(0 to NUM_USER_01_CE -1);

begin (architecture)


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

Figure 20: **Bus Ripping Example**

Point-To-Point Configuration

A Point-To-Point configuration is defined as 1 Master talking to 1 Slave over the PLB Bus. If the PLB topology is configured such that it is a Point-To-Point configuration then some of the FPGA resource utilization can be reduced as well as some of the latency through the slave (See Figure 15 through Figure 18). To accomplish this the user must set C_SPLB_P2P = 1.

Single Address Pair in C_ARD_ADDR_RANGE_ARRAY

For situations where the User does not require additional slave services then 1 address pair should be defined in C_ARD_ADDR_RANGE_ARRAY. The address pair should span the entire address range of the system. For example, set the base address to X"0000_0000_0000_0000" and set the high address to X"FFFF_FFFF_FFFF_FFFF".

Multiple Address Pair in C_ARD_ADDR_RANGE_ARRAY

For situations where other slave services are required, such as interrupt control and/or soft reset services, then the user will need to define multiple address pairs in C_ARD_ADDR_RANGE_ARRAY. If the user defines multiple address pairs in C_ARD_ADDR_RANGE_ARRAY then from a system perspective the PLB Bus starts looking like a shared bus, i.e. one having multiple Slaves. For this configuration the bus no longer represents a point to point configuration. In other words the address decode logic must be instantiated in the Slave Attachment to decode the multiple address ranges. When multiple address pairs are defined in C_ARD_ADDR_RANGE_ARRAY and C_SPLB_P2P = 1

the Slave Attachment latency will be the same as if there was only 1 address pair defined in C_ARD_ADDR_RANGE_ARRAY, but some of the LUT savings of the Point-To-Point mode will not be realized.

For this situation the user should set the base address and high address for the services starting at X"0000_0000_0000_0000". The Users IP should then follow the service address ranges and have a high address of X"FFFF_FFFF_FFFF_FFFF". (See Figure 21)

```
C_ARD_ADDR_RANGE_ARRAY : SLV64_ARRAY_TYPE :=
        -- Base address and high address pairs .
         (
          X"0000_0000_0000_0000",   -- Interrupt Controller Base Address
          X"0000_0000_0000_003F",   -- Interrupt Controller High Address
          X"0000_0000_0000_0040",   -- Soft Reset Base Address
          X"0000_0000_0000_0043",   -- Soft Reset High Address
          X"0000_0000_0000_0100",   -- User IP Base Address
          X"FFFF_FFFF_FFFF_FFFF"  -- User IP High Address

          );
```

Figure 21: **C_ARD_ADDR_RANGE_ARRAY for Point-To-Point Configuration**

Note that the Slave Attachment will not acknowledge and address if it falls outside of an address range defined in C_ARD_ADDR_RANGE_ARRAY. For example, if C_ARD_ADDR_RANGE_ARRAY is defined as in Figure 21 and a master attempts to read or write to address 0x80 then the Slave attachment will NOT acknowledge the cycle.

Data Phase Timeout

A data phase timeout function has been incorporated into the plbv46_slave_single to provide a means to complete a PLB request even when the User IP does not respond with an IP2Bus_RdAck or IP2Bus_WrAck. If C_INCLUDE_DPHASE_TIMER = 1 and after 128 SPLB_Clk cycles, as measured from the assertion of Sl_AddrAck, the User IP does not respond with either an IP2Bus_RdAck or IP2Bus_WrAck the plbv46_slave_single will de-assert the User IP cycle request signals, Bus2IP_CS and Bus2IP_RdCE or Bus2IP_WrCE, and will assert Sl_rdDAck with Sl_rdDBus=zero for a read cycle or Sl_wrDAck for a write cycle. This will gracefully terminate the cycle. Note that the requesting master will have no knowledge that the data phase of the PLB request was terminated in this manner.

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:

Incorrect

C_ARD_NUM_CE_ARRAY(16); -- *VHDL **positional association**....may cause synthesis type conflict error for single entry!*

Correct

C_ARD_NUM_CE_ARRAY(0 => 16); -- *VHDL **named association**....avoids type conflict error for single entry.*

Register Descriptions

The PLBv46 Slave Burst core has no internal registers.

Design Implementation

Target Technology

The intended target technology is a Spartan-3e, Spartan-3a, Virtex-4 and Virtex-5 FPGA.

Device Utilization and Performance Benchmarks

Since the PLBv46 Slave Burst core 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 PLBv46 Slave Burst core 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 PLBv46 Slave Burst core is shown in Table 5 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 PLBv46 Slave Burst core benchmarks are shown in Table 5 for a Virtex-5 -2 FPGA.

.

Table 5: **PLBv46 Slave Burst Core FPGA Performance and Resource Utilization Benchmarks**

Parameter Values							Device Resources		
C_ARD_ADDR_RANGE_ARRAY Pairs	C_ARD_NUM_CE_ARRAY	C_SPLB_DWIDTH	C_SIPIF_DWIDTH	C_SPLB_SMALLEST_MASTER	C_WR_BUFFER_DEPTH	C_SPLB_P2P	**Slice Flip-Flops**	**Slice/LUTs**	**f_{MAX}**(1)
1	1	32	32	32	0	1	151	218	202.4
1	1	32	32	32	0	0	213	223	200.4
1	1	32	32	32	16	0	265	265	206.1

Table 5: **PLBv46 Slave Burst Core FPGA Performance and Resource Utilization Benchmarks**

1	2	32	32	32	16	0	267	271	201.0
1	4	32	32	32	16	0	272	272	202.6
2	4,4	32	32	32	16	0	280	277	203.3
2	4,4	64	64	32	16	0	357	383	203.6
2	4,4	128	64	32	16	0	368	385	202.0
2	4,4	128	128	128	16	0	511	448	201.0

Notes:

1. Fmax represents the maximum frequency of the PLBV46 Slave Burst 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.

Specification Exceptions

The following High Level PLB features are **not** supported by the PLBv46 Slave Burst core slave function.

- Bus Master
- Split Bus Transactions
- Address Pipelining
- Abort Transactions
- Indeterminate Burst
- Burst length requests greater than 16

Reference Documents

The following documents contain reference information important to understanding the PLB Slave Attachment design.

- IBM CoreConnect 128-Bit Processor Local Bus, Architectural Specification (v4.6).

Revision History

Date	Version	Revision
5/14/08	1.0	Initial Xilinx release.
1/15/09	1.1	Added a single beat read and single beat write timing diagrams for C_P2P=0
6/22/10	1.2	Added support for Virtex-6 and Spartan-6.