Creating and Adding Custom IP

Zynq
14.2 Version

Objectives

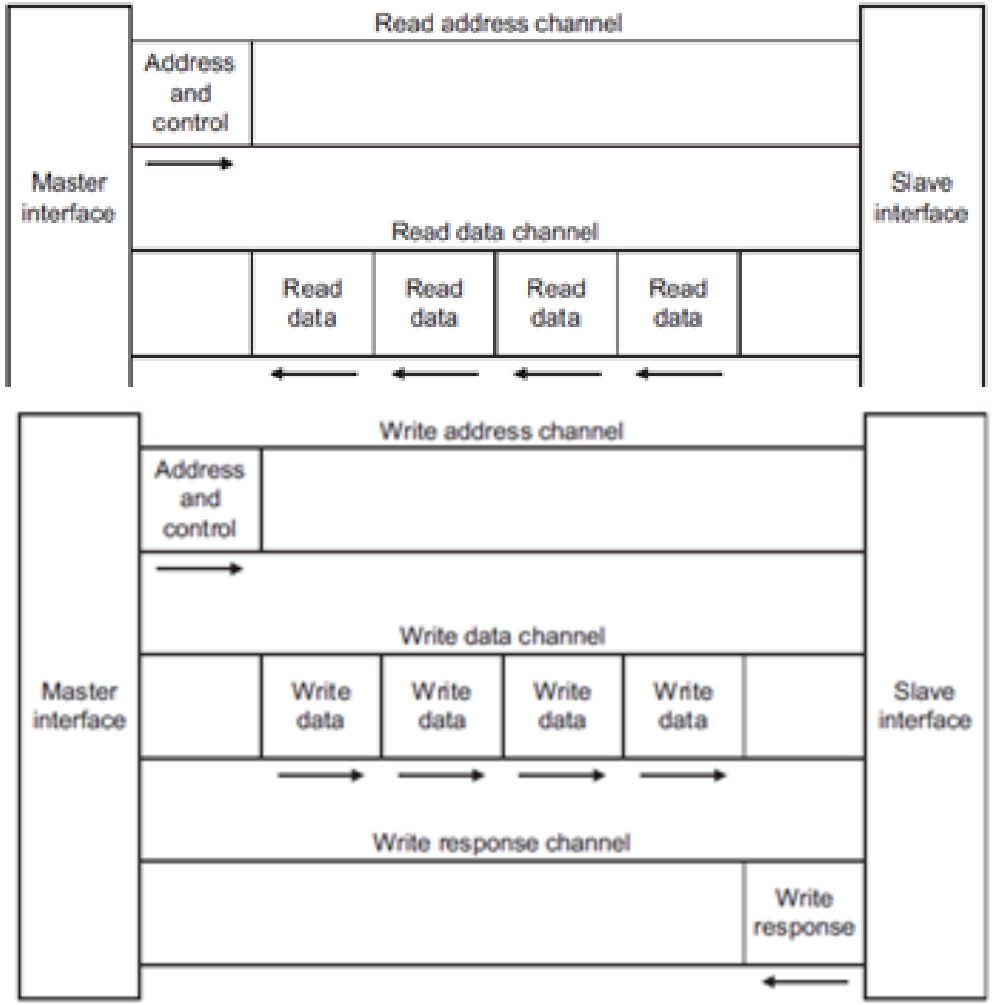
- **After completing this module, you will be able to:**
  - Describe the AXI4 transactions
  - Summarize the AXI4 valid/ready acknowledgment model
  - Discuss the AXI4 transactional modes of overlap and simultaneous operations
  - Describe the operation of the AXI4 streaming protocol
  - List the steps involved in creating peripherals using Create/Import IP wizard

Outline

- ***AXI4 Transactions***
  - AXI4 Lite Slave
  - AXI4 Lite Master
  - AXI4 Slave
  - AXI4 Master
- **Create and Import Peripheral Wizard**
- **Incorporating your IP functionality**
- **Summary**

Basic AXI Transaction Channels

- **Read address channel**
- **Read data channel**
- **Write address channel**
- **Write data channel**
- **Write response channel**
  - Non-posted write model
    - There will always be a "write response"



All AXI Channels Use A Basic “VALID/READY” Handshake

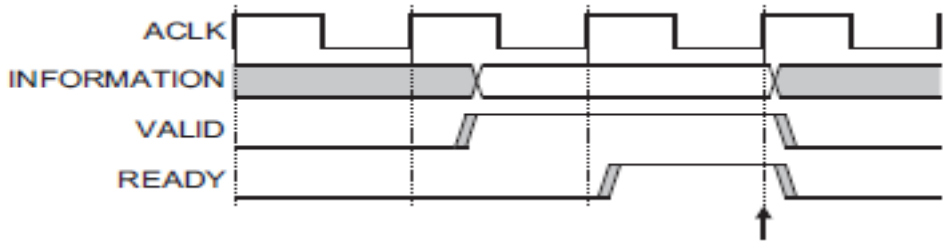
- ***SOURCE*** **asserts and holds VALID when DATA is available**
- ***DESTINATION*** **asserts READY if able to accept DATA**

- **DATA transferred when VALID and READY = 1**

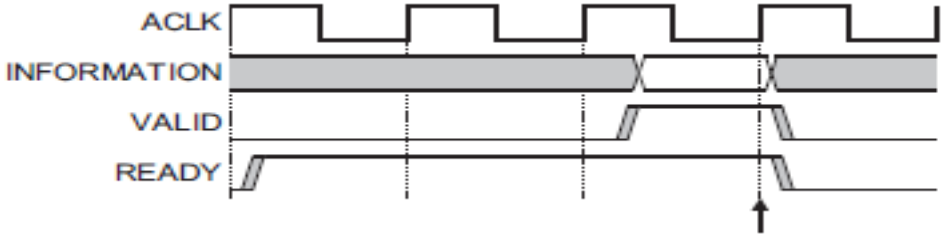
- ***SOURCE*** **sends next DATA (if an actual data channel) or deasserts VALID**
- ***DESTINATION*** **deasserts READY if no longer able to accept DATA**

AXI Interface: Handshaking

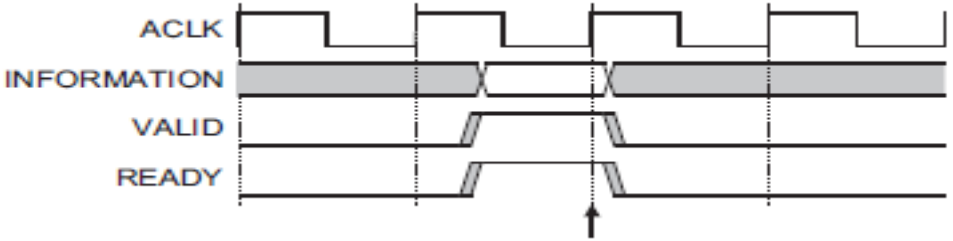
- **AXI uses a valid/ready handshake acknowledge**
- **Each channel has its own valid/ready**
  - Address (read/write)
  - Data (read/write)
  - Response (write only)
- **Flexible signaling functionality**
  - Inserting wait states
  - Always ready
  - Same cycle acknowledge



Inserting Wait States



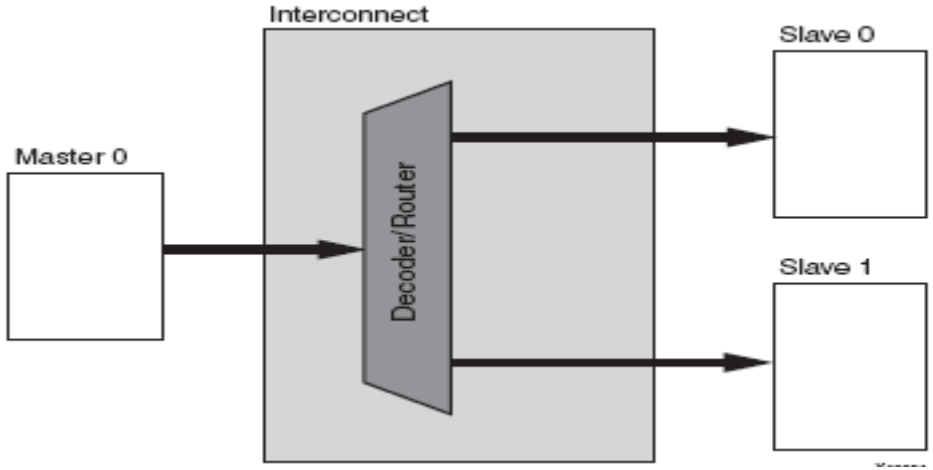
Always Ready

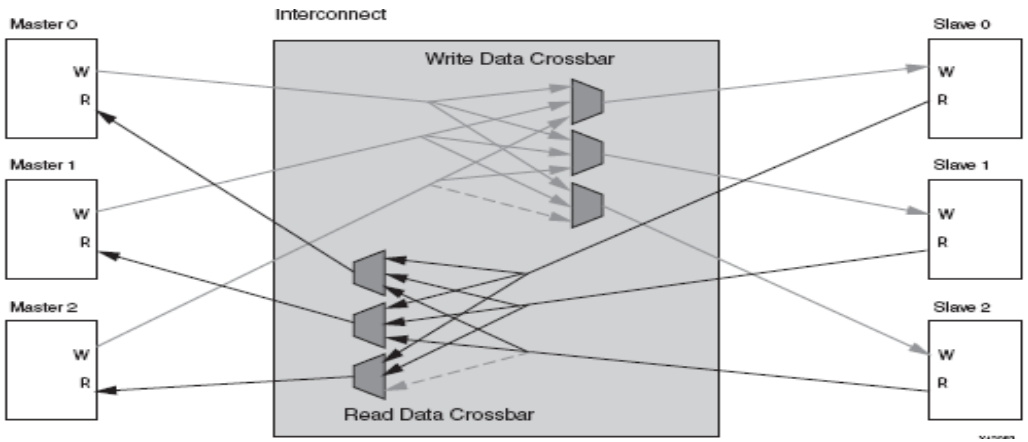


Same Cycle Acknowledge

AXI Interconnect

- **axi_interconnect component**
  - Highly configurable
    - Pass Through
    - Conversion Only
    - N-to-1 Interconnect
    - 1-to-N Interconnect
    - N-to-M Interconnect – full crossbar
    - N-to-M Interconnect – shared bus structure
- **Decoupled master and slave interfaces**
- **Xilinx provides three configurable IPIC**
  - AXI4 Lite Slave
  - AXI4 Lite Master
  - AXI4 Slave Burst
- **Xilinx AXI Reference Guide(UG761)**





AXI4 Signals (AXI4, AXI4-Lite)

	AXI4	AXI4-Lite
Glbl	ACLK	
	ARESETN	
Write Address	AWID	
	AWADDR	
	AWLEN	
	AWSIZE	
	AWBURST	
	AWLOCK	
	AWCACHE	
	AWPROT	
	AWQOS	
	AWSIZE	
	AWREGION	
	AWLOCK	
	AWUSER	
	AWVALID	
	AWREADY	

	AXI4	AXI4-Lite
Write Data	WDATA	WDATA
	WSTRB	WSTRB
	WLAST	
	WUSER	
	WVALID	
	WREADY	
Write Resp.	BID	
	BRESP	BRESP
	BUSER	
	BVALID	
	BREADY	

	AXI4	AXI4-Lite
Read Address	ARID	
	ARADDR	
	ARLEN	
	ARSIZE	
	ARBURST	
	ARLOCK	
	ARCACHE	ARCACHE
	ARPROT	ARPROT
	ARQOS	
	ARREGION	
	ARUSER	
	ARVALID	
	ARREADY	

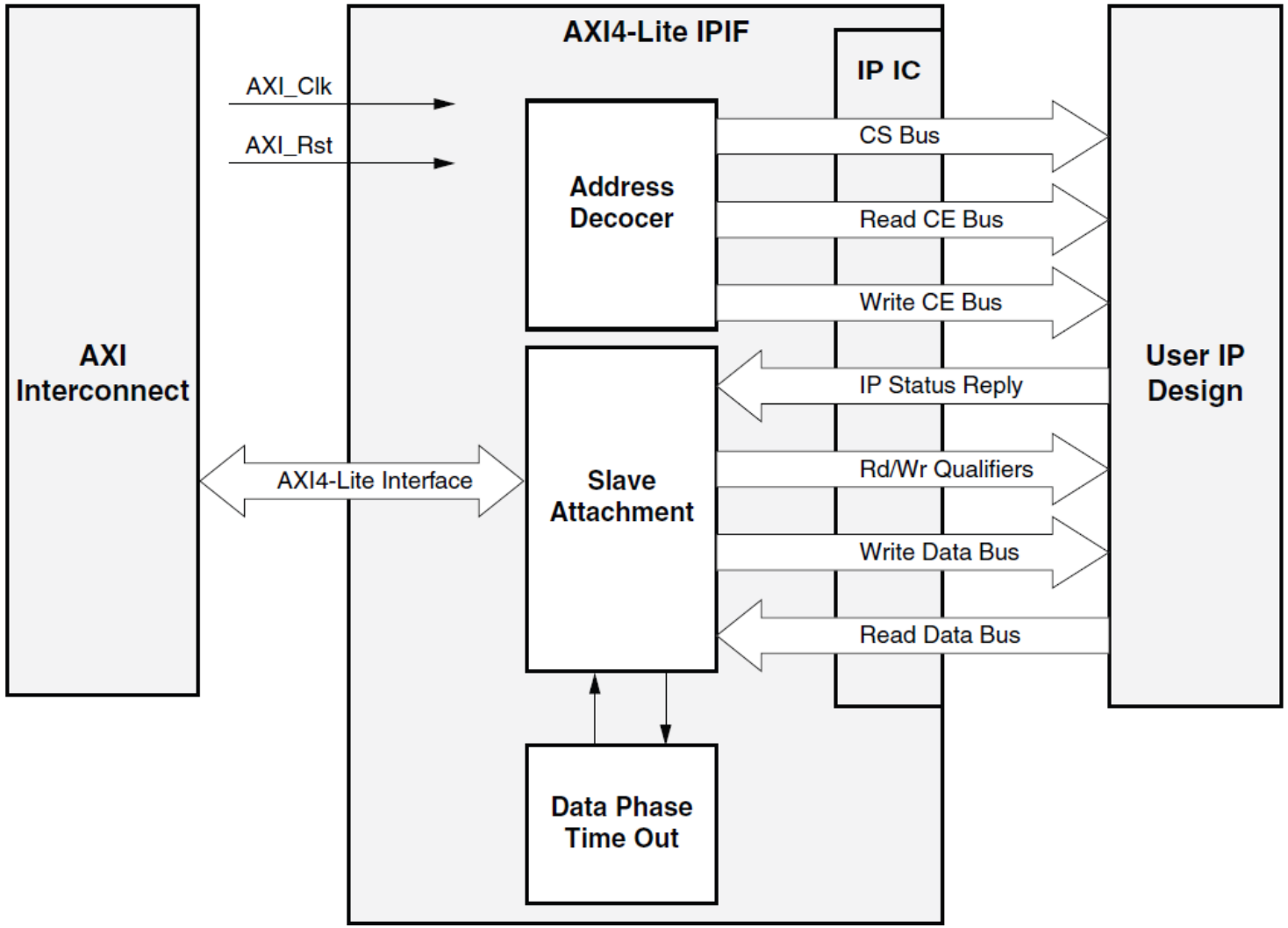
	AXI4	AXI4-Lite
Read Data	RID	
	RDATA	RDATA
	RRESP	RRESP
	RLAST	
	RUSER	
	RVALID	
	WREADY	

Outline

- ***AXI4 Transactions***
  - *AXI4 Lite Slave*
  - AXI4 Lite Master
  - AXI4 Slave
  - AXI4 Master
- **Create and Import Peripheral Wizard**
- **Incorporating your IP functionality**
- **Summary**

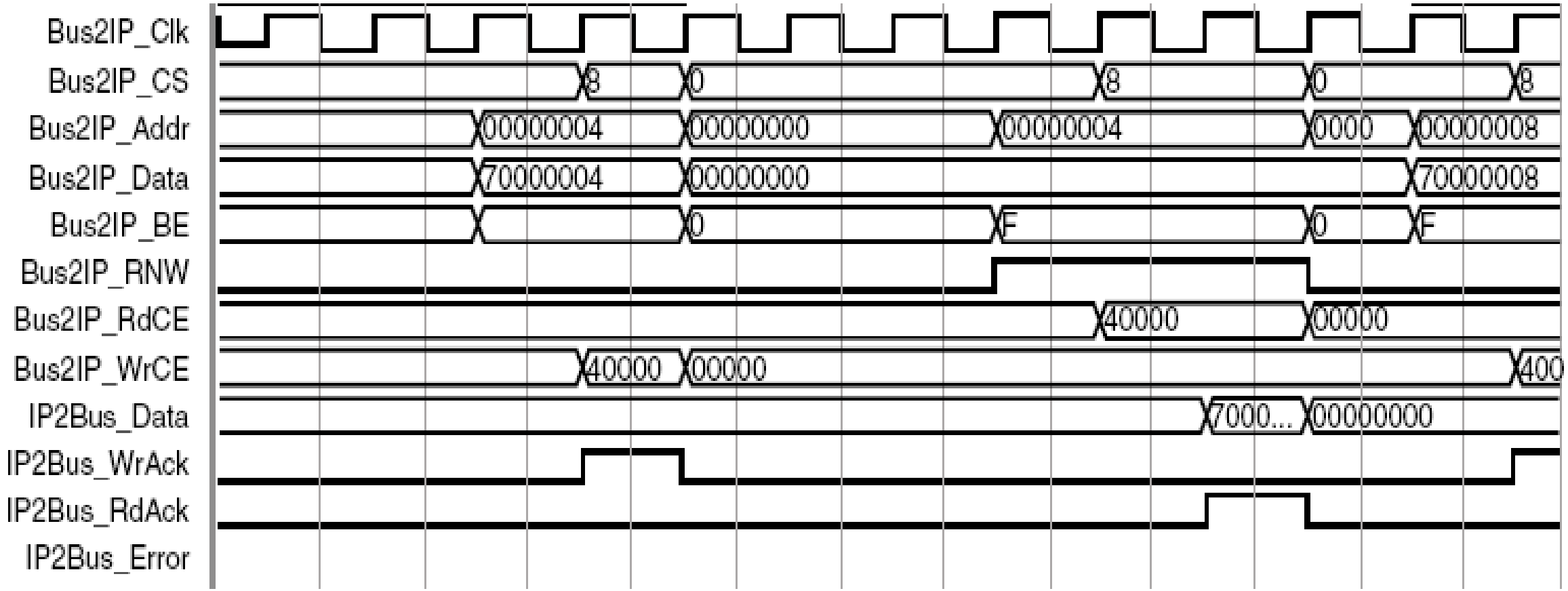
AXI Lite IPIF – Block Diagram

- **Basic services**
  - Slave attachment
  - Address decoding
  - Timeout generation
  - Byte strobe forwarding
- **Optional services**
  - Master user logic
  - Soft reset core
  - User logic software registers, and
  - Timeout logic inclusion



AXI Lite IPIC

Single Data Phase Write and Read Cycle

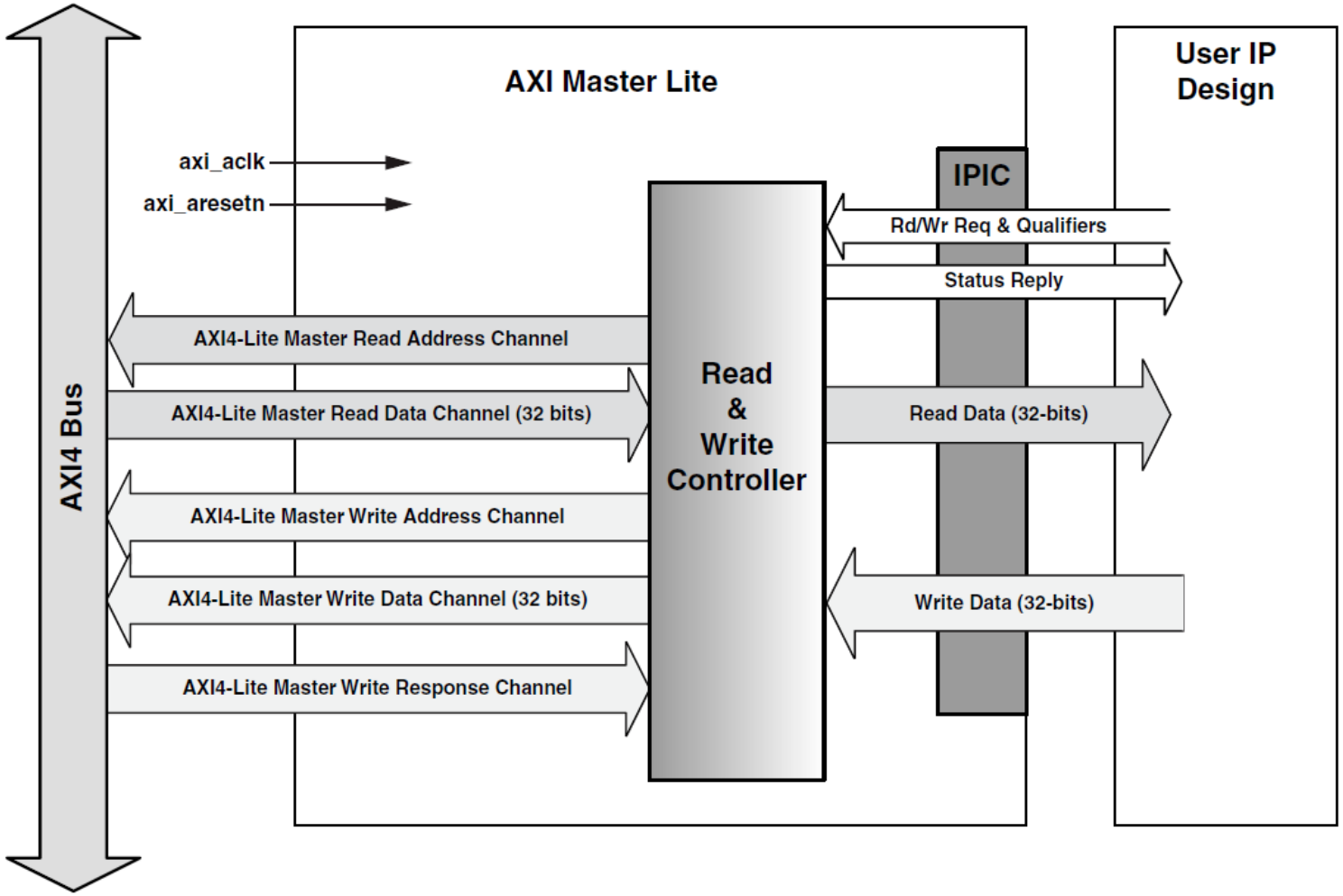


Outline

- ***AXI4 Transactions***
  - AXI4 Lite Slave
  - *AXI4 Lite Master*
  - AXI4 Slave
  - AXI4 Master
- **Create and Import Peripheral Wizard**
- **Incorporating your IP functionality**
- **Summary**

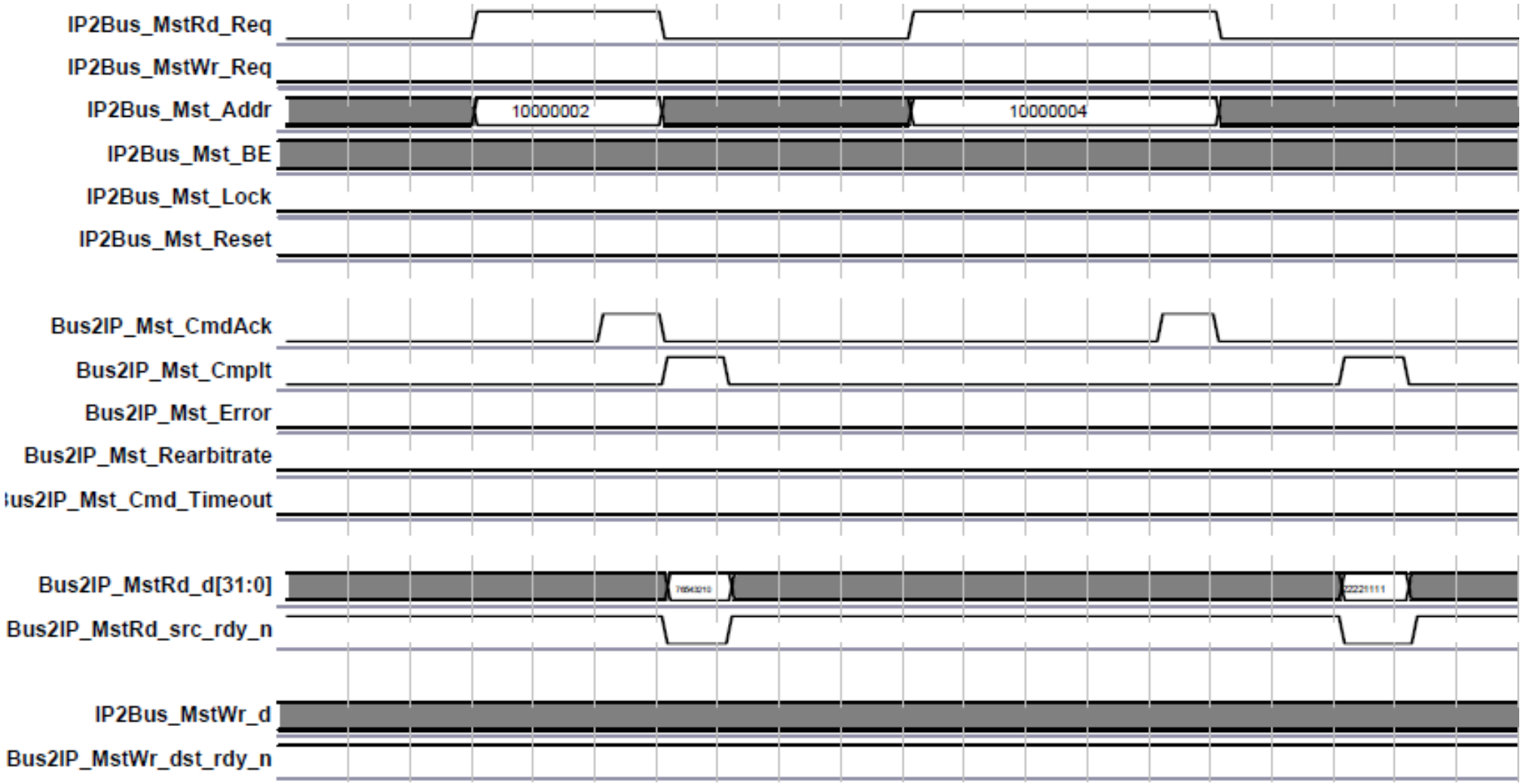
AXI4 Lite Master – Block Diagram

- **AXI4 Lite mastering capability**
- **Single data phase only**
  - One to four bytes
- **Only 32-bit data width**



AXI4 Lite Master

Single Data Phase Read Cycle



Outline

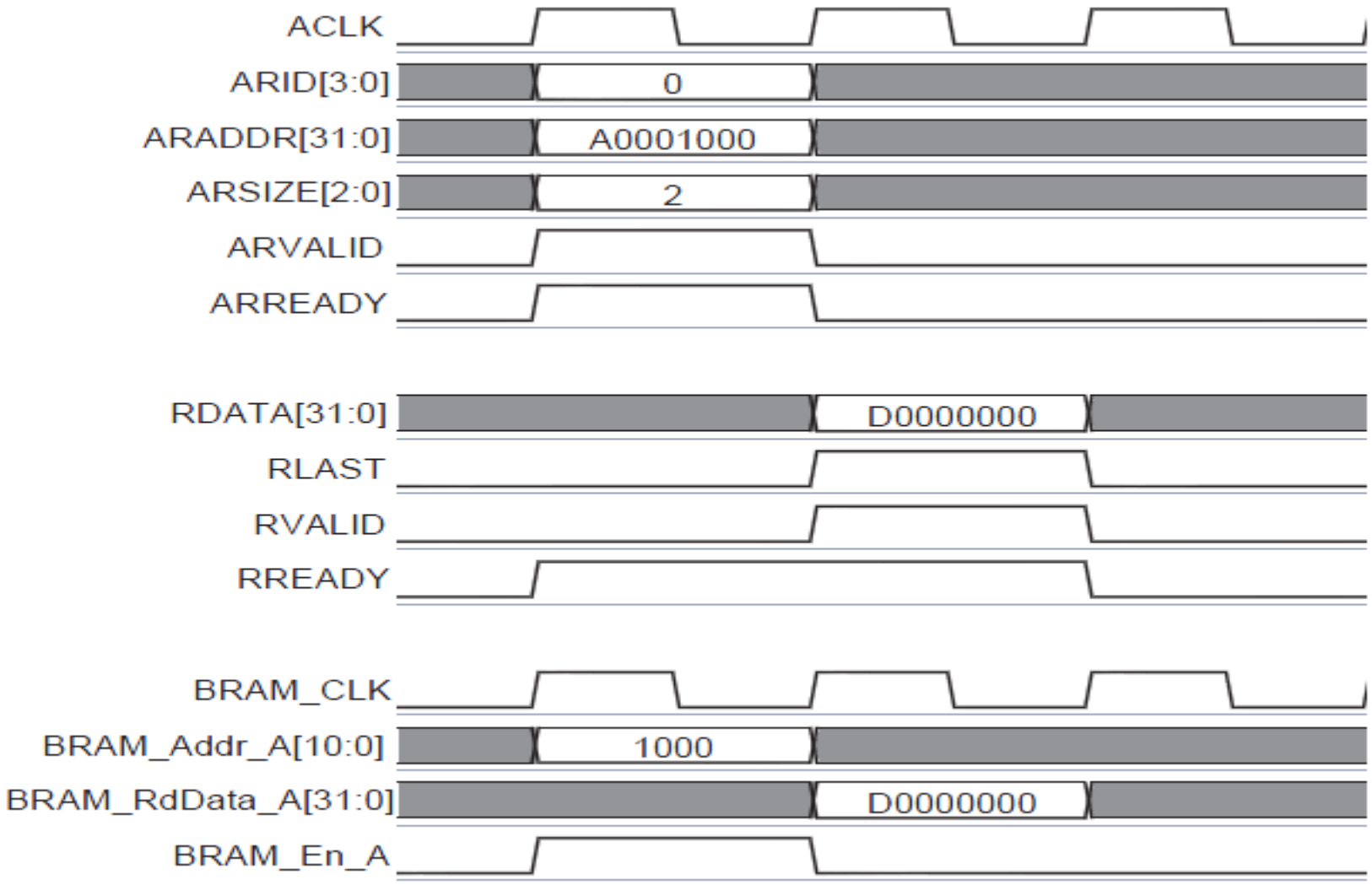
- ***AXI4 Transactions***
  - AXI4 Lite Slave
  - AXI4 Lite Master
  - *AXI4 Slave*
  - AXI4 Master
- **Create and Import Peripheral Wizard**
- **Incorporating your IP functionality**
- **Summary**

AXI4 Slave

Duties, Options, and Configuration

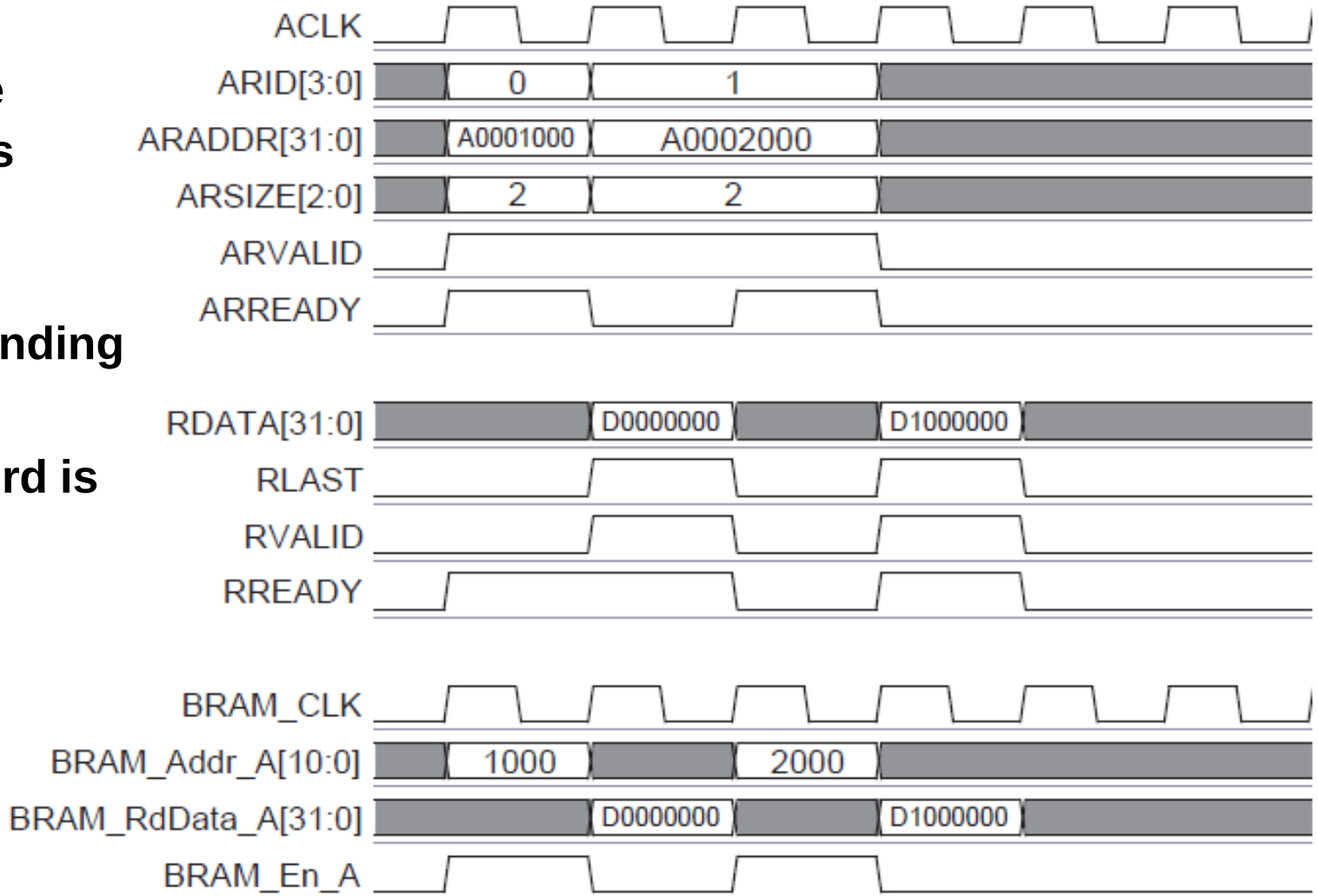
- **Slave IPIC duties and configurable options**
  - Address decode and acknowledge
  - One or more address spaces
    - Memory interface; that is, chip enable
    - User registers
  - Single or burst data phase acknowledgement
  - Software reset/MIR register
  - Read FIFOs
  - Automatic timeout on user slave logic
  - Your custom slave attachments

AXI4 Read Transaction

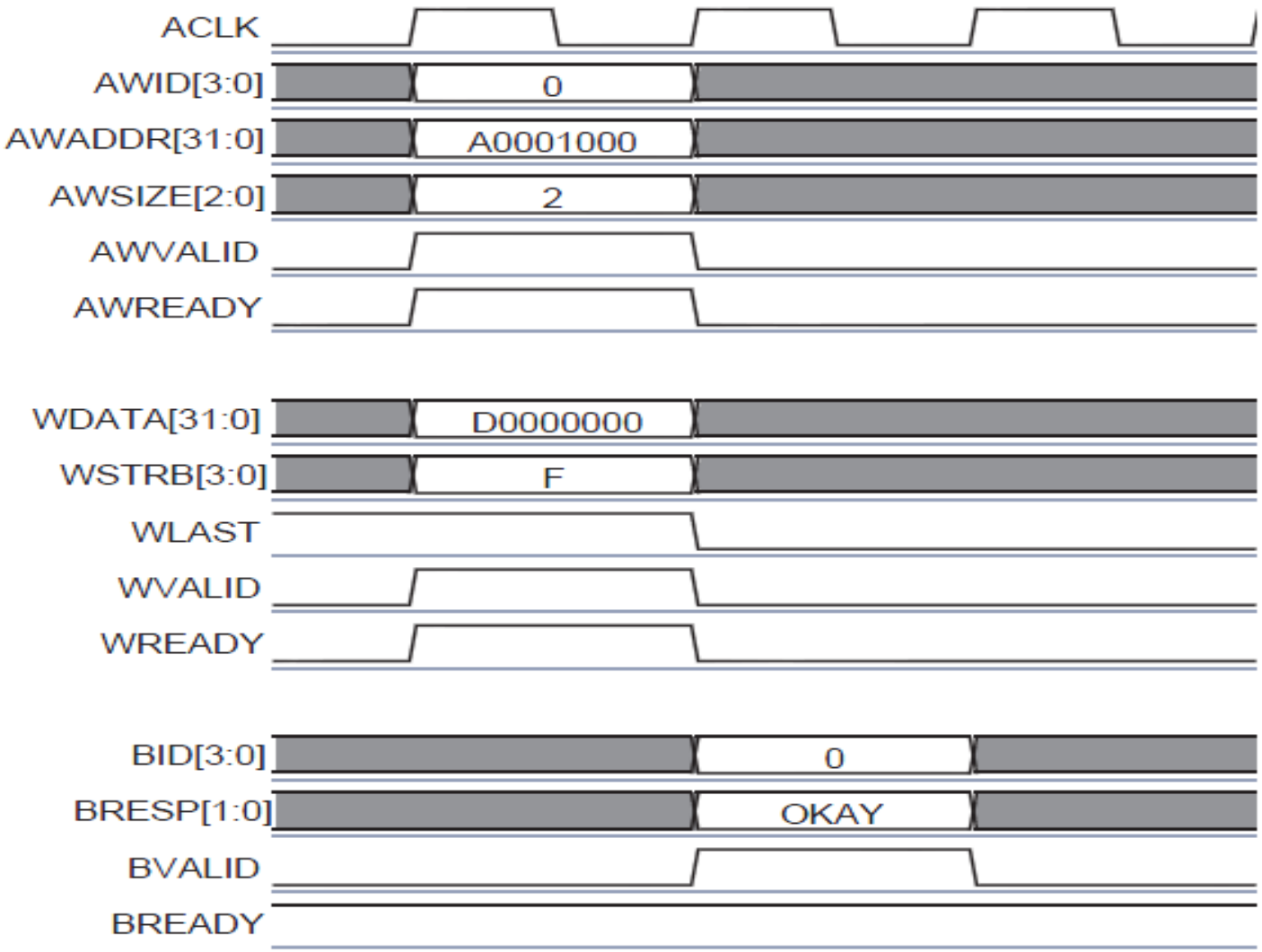


AXI4 Read Transaction
Multiple reads, not burst transaction

- **Multiple read transactions are identified by different read IDs**
  - For a burst transaction only one read ID would have been used
- **Separate RLAST for corresponding read transactions**
- **ARSIZE=2 indicates entire word is being read**

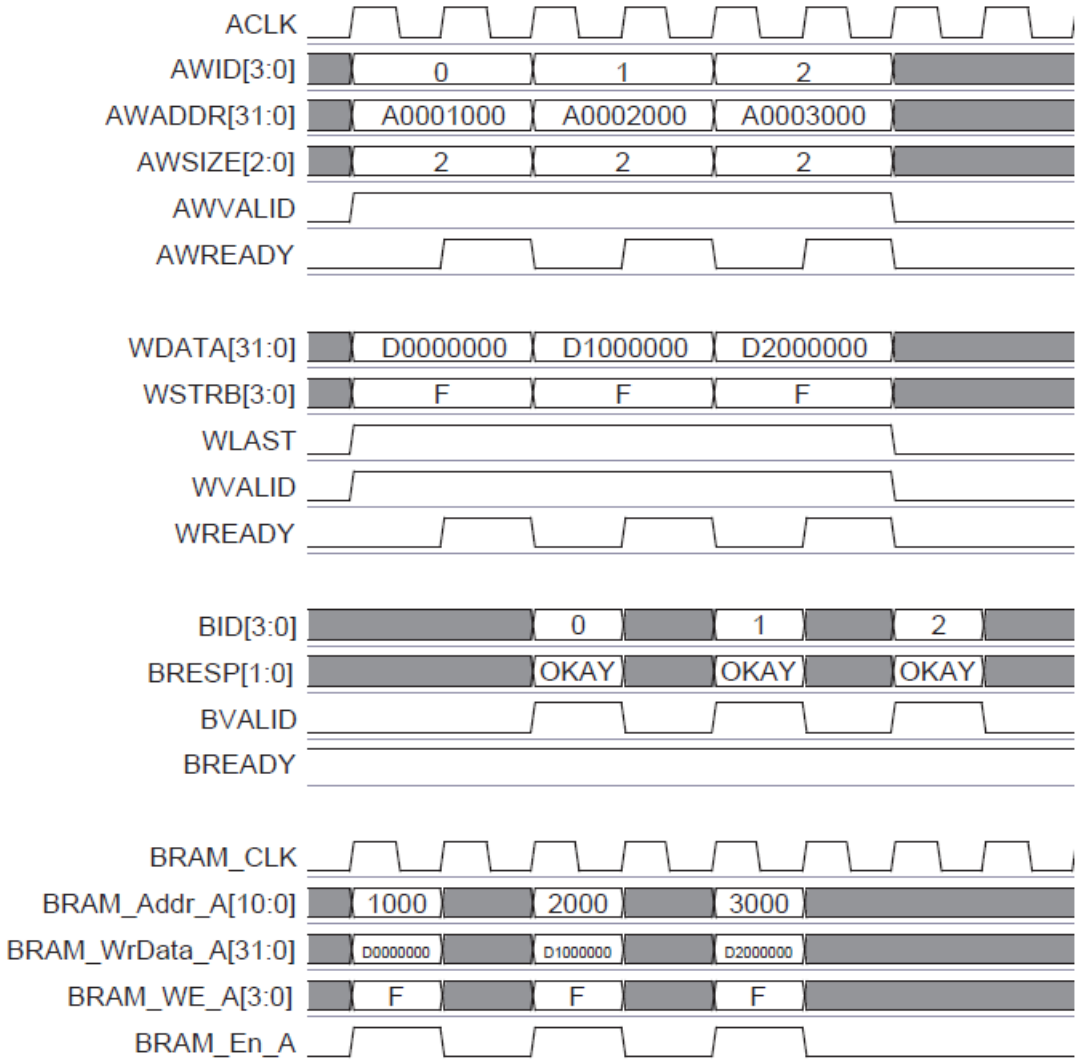


AXI4 Write Transaction

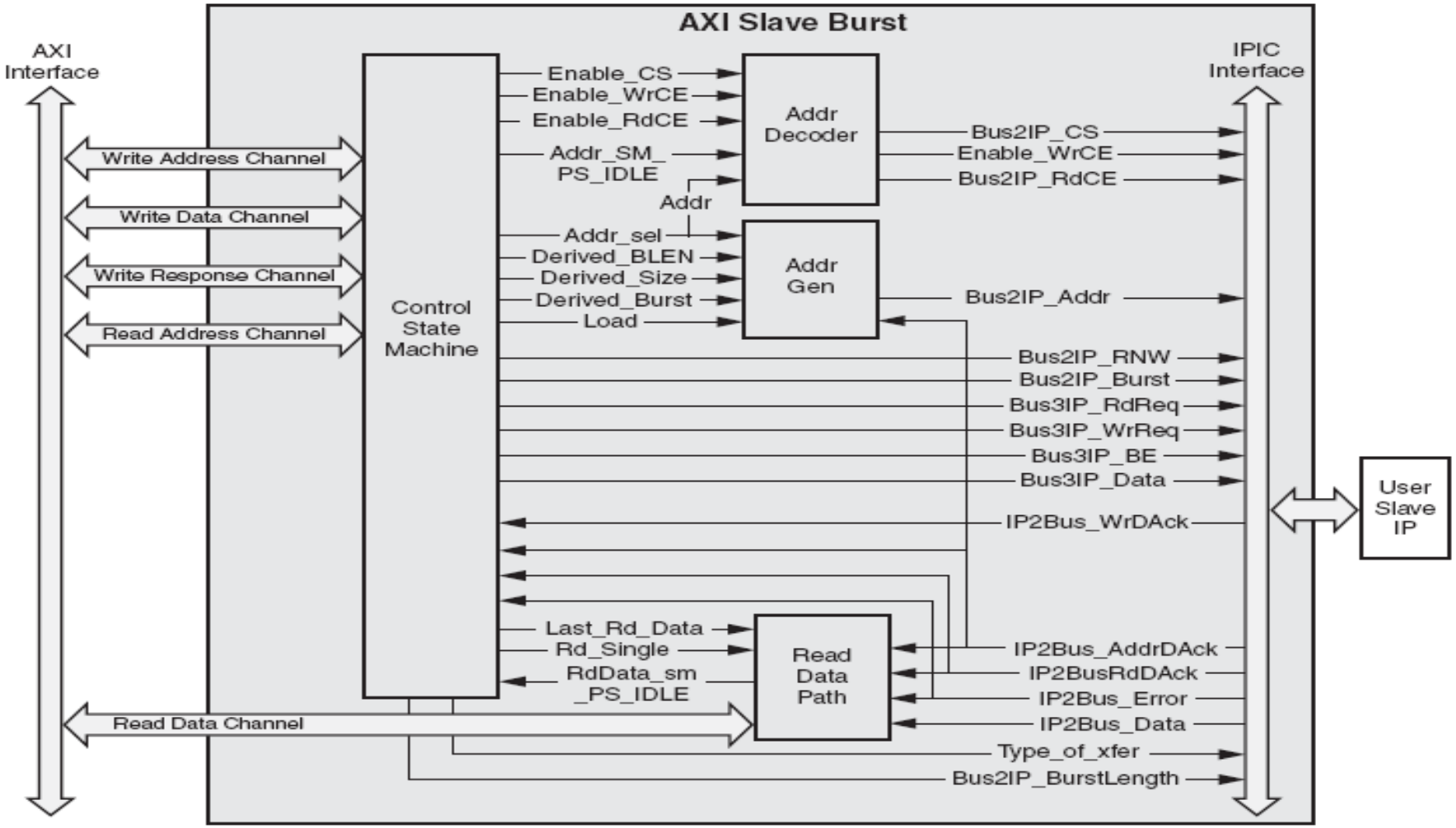


AXI4 Write Transaction
Multiple write, not burst

- **Multiple write transactions are identified by different write IDs (AWID)**
  - For a burst transaction only one write ID would have been used
- **Separate OKAY status for corresponding write transactions**
- **WSTRB=0xF indicates entire word is being written**

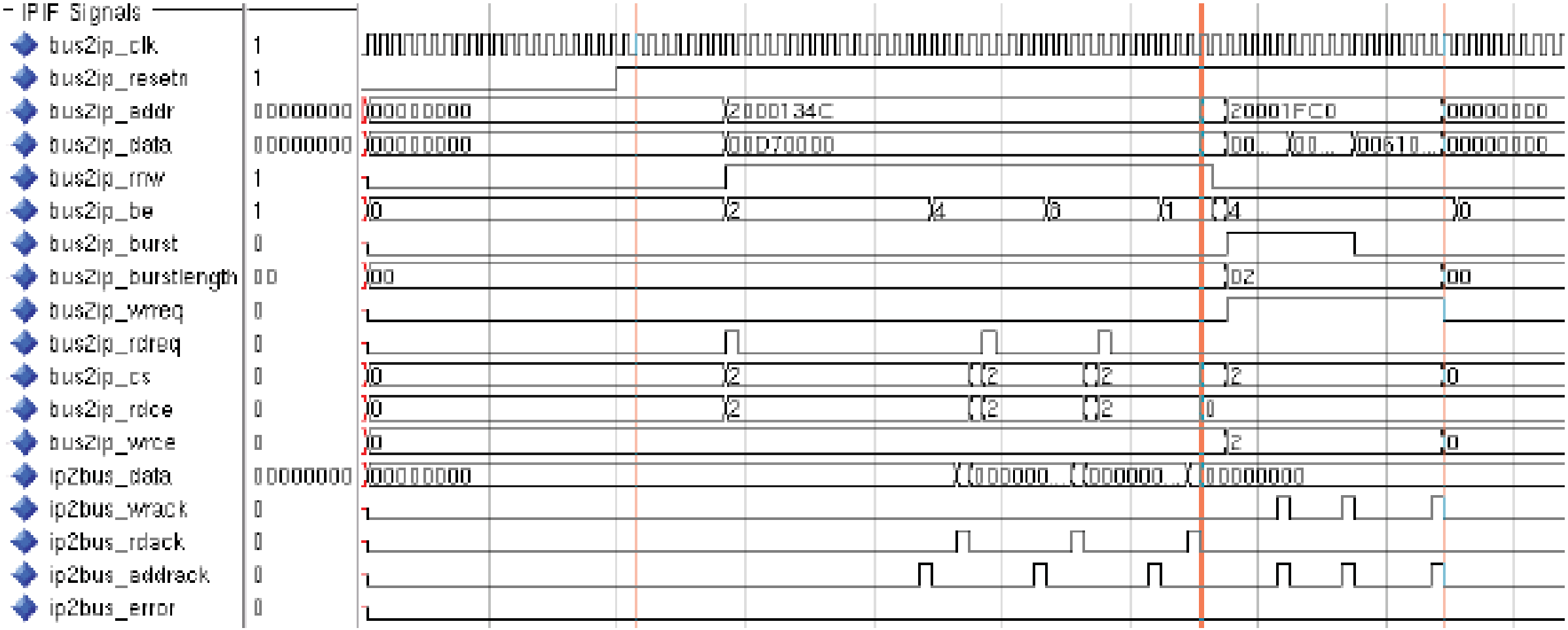


AXI Slave Burst – Block Diagram



AXI Slave Burst

Burst Data Phase 3 Reads and 3 Writes



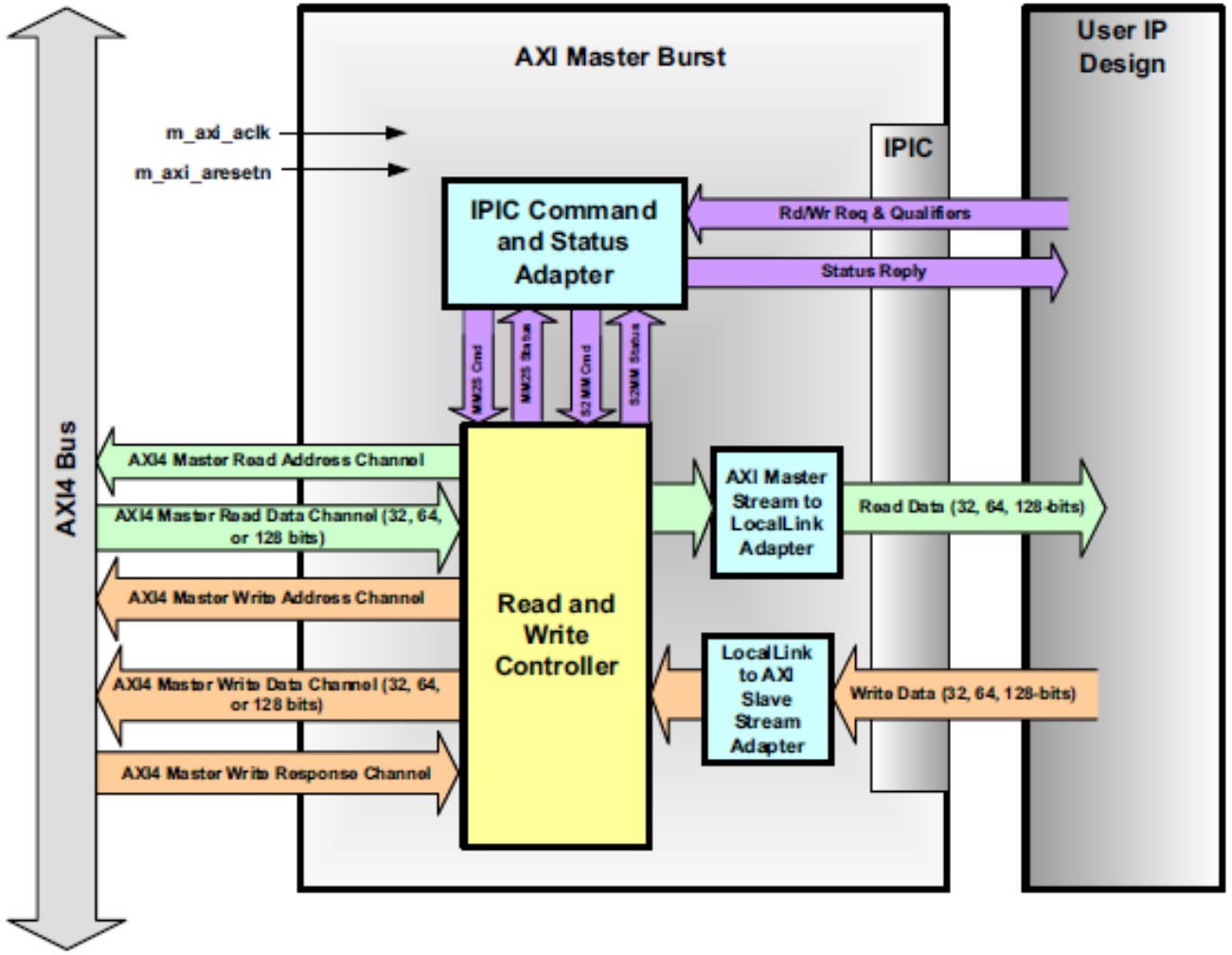
XILINX ALL PROGRAMMABLE.

Outline

- ***AXI4 Transactions***
  - AXI4 Lite Slave
  - AXI4 Lite Master
  - AXI4 Slave
  - *AXI4 Master*
- **Create and Import Peripheral Wizard**
- **Incorporating your IP functionality**
- **Summary**

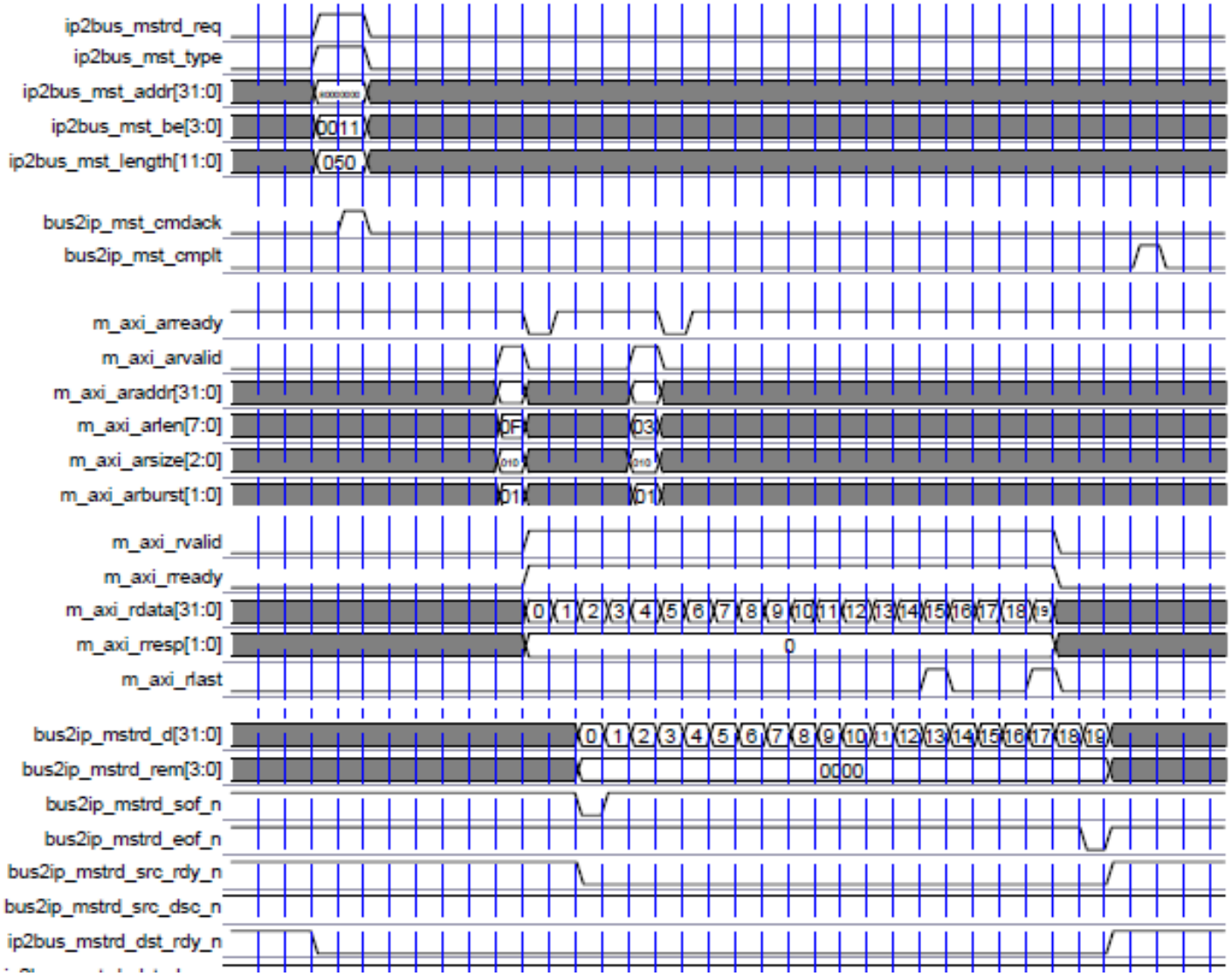
AXI4 Master Burst – Block Diagram

- **Parameterizable data width**
  - 32, 64, 128
- **Data burst**
  - 16, 32, 64, 128, 256 data beats



AXI4 Master Burst Read

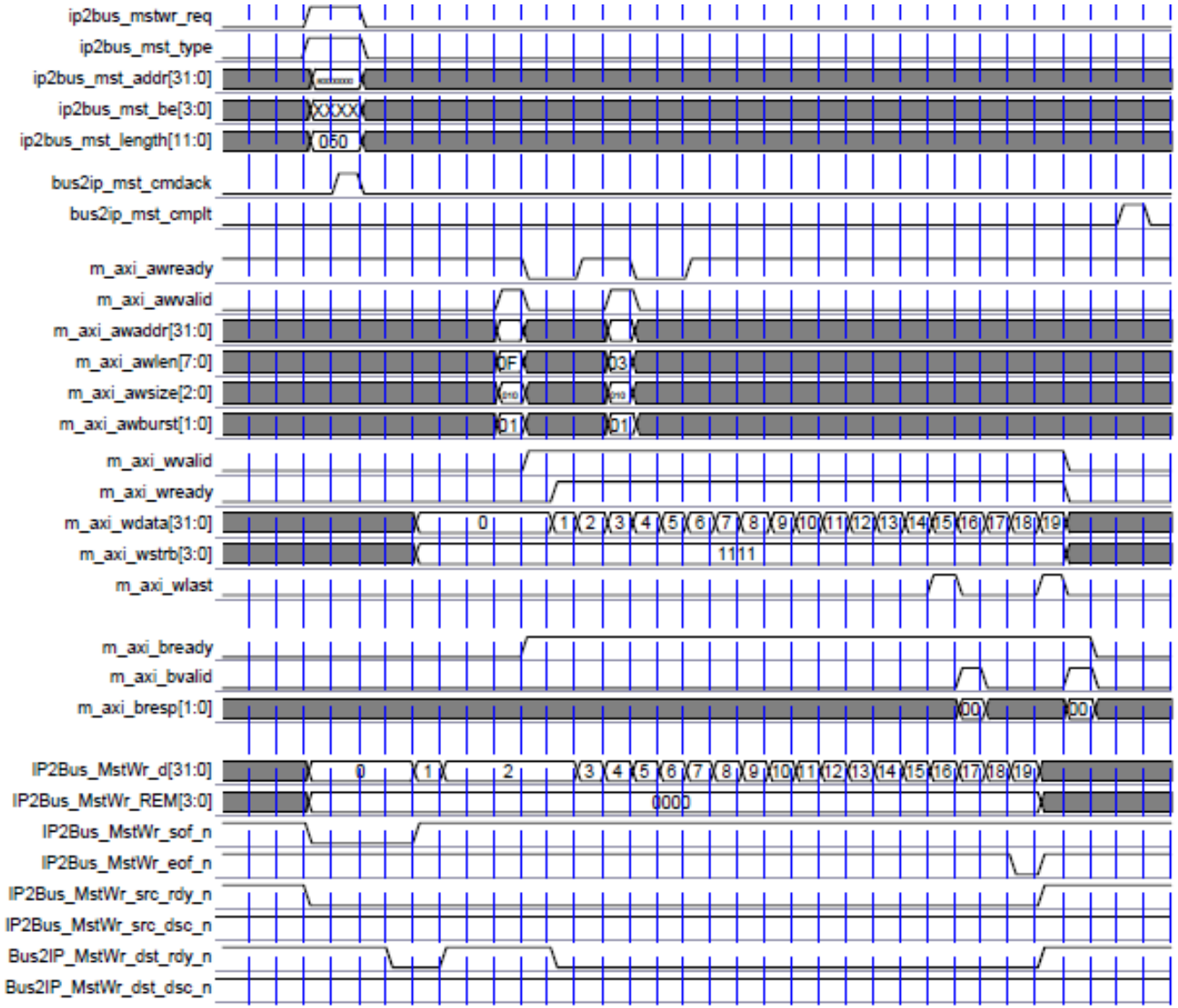
- **80 bytes of burst read**
  - ip2bus_mst_length signal
- **Transaction delimited by**
  - bus2ip_mst_cmdack and bus2ip_mst_cmplt signals
- **Burst read broken into two transactions**
  - 16 data beats (64 bytes)
  - 4 data beats (16 bytes)
- **AXI master receive data**
  - m_axi_rdata
- **User logic receive data**
  - bus2ip_mstrd_d
- **Data framing**
  - bus2ip_mstrd_sof_n
  - bus2ip_mstrd_eof_n



Only relevant signals are shown for readability purpose

AXI4 Master Burst Write

- **80 bytes of burst write**
  - ip2bus_mst_length signal
- **Transaction delimited by**
  - bus2ip_mst_cmdack and bus2ip_mst_cmplt signals
- **Burst write broken into two transactions**
  - 16 data beats (64 bytes)
  - 4 data beats (16 bytes)
- **User logic writes data**
  - IP2Bus_MstWr_d
- **AXI master write data**
  - m_axi_wdata
- **Data framing**
  - IP2Bus_MstWr_sof_n
  - IP2Bus_MstWr_eof_n



Only relevant signals are shown for readability purpose

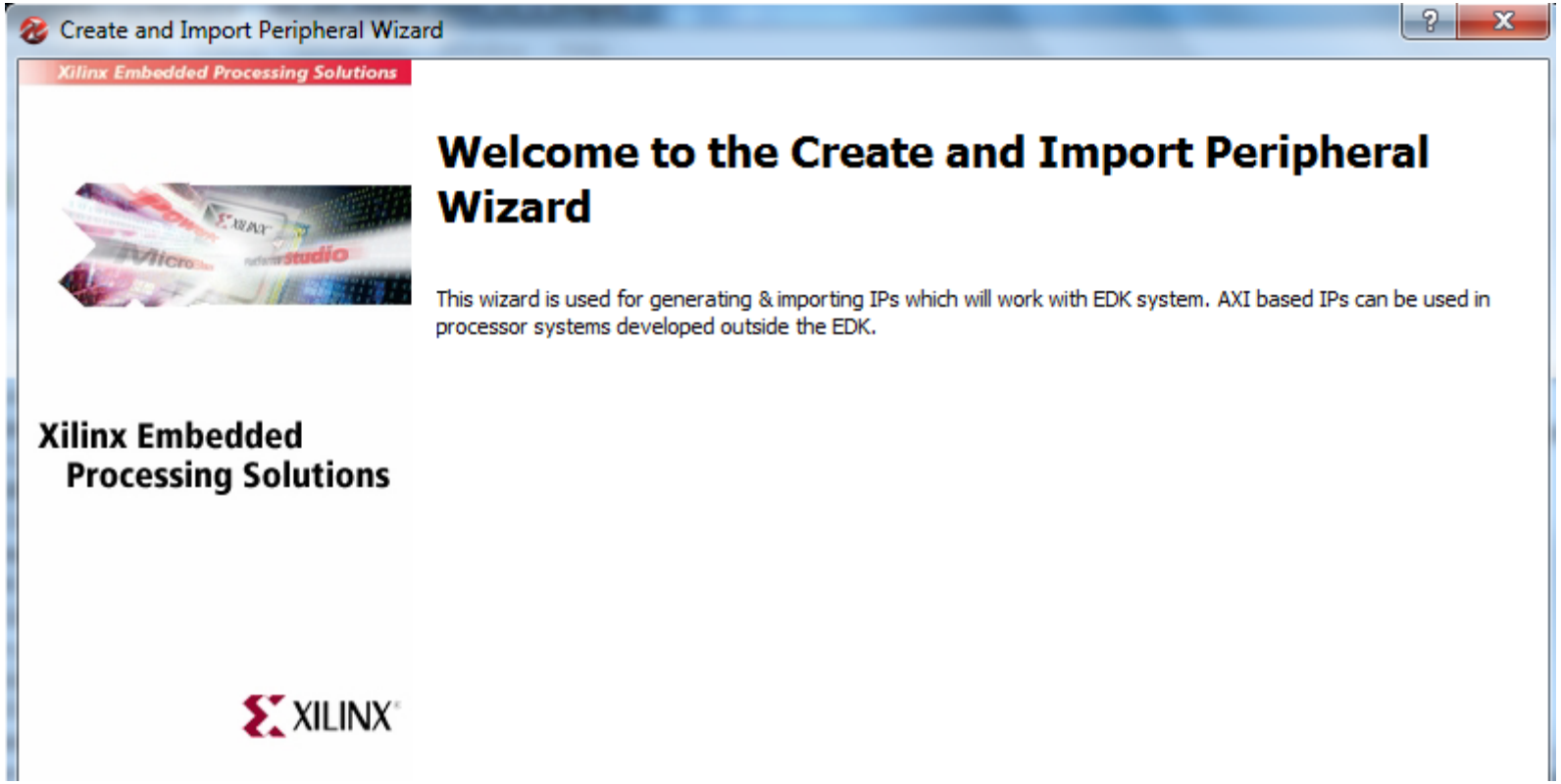
Outline

- AXI4 Transactions
- ***Create and Import Peripheral Wizard***
- Incorporating your IP functionality
- Summary

Create and Import Peripheral Wizard (CIP Wizard)

- **The wizard helps you create your own peripheral. It can also allow you to import a previously created peripheral into your design**
- **The wizard generates the necessary core description files into the user-selected directory**
- **You can start the wizard after creating a new project or opening an existing project in XPS**
- **The user peripheral can be imported directly through the wizard by skipping the creation option provided the peripheral already exists**
  - Ensure that the peripheral complies with Xilinx implementation of AXI4 interface

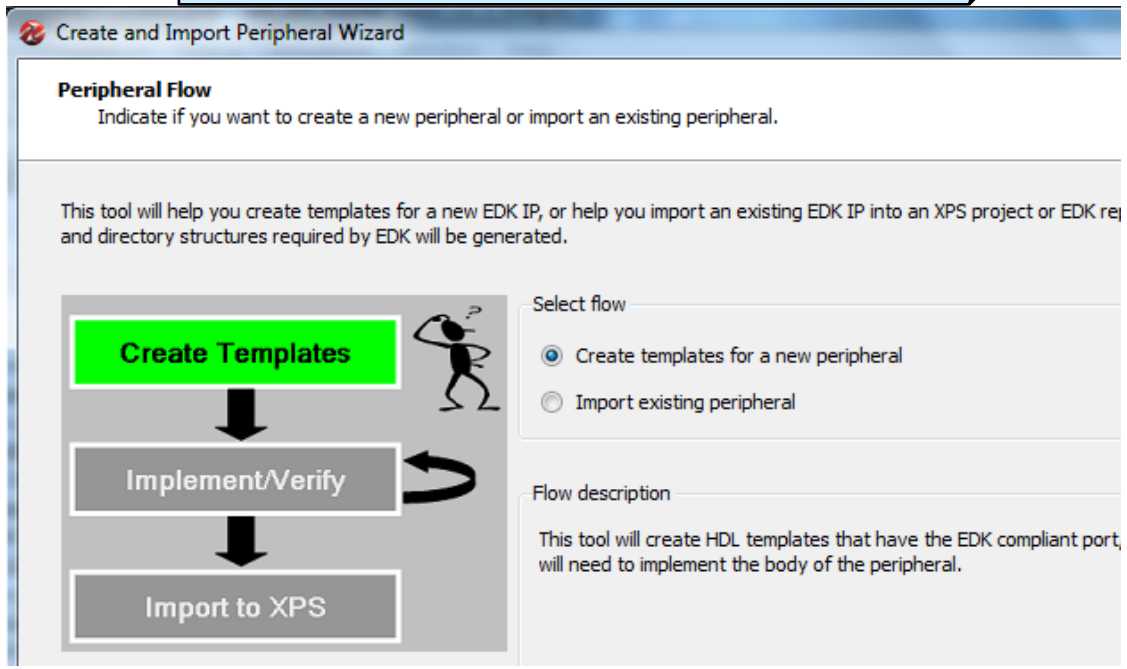
Starting the CIP Wizard



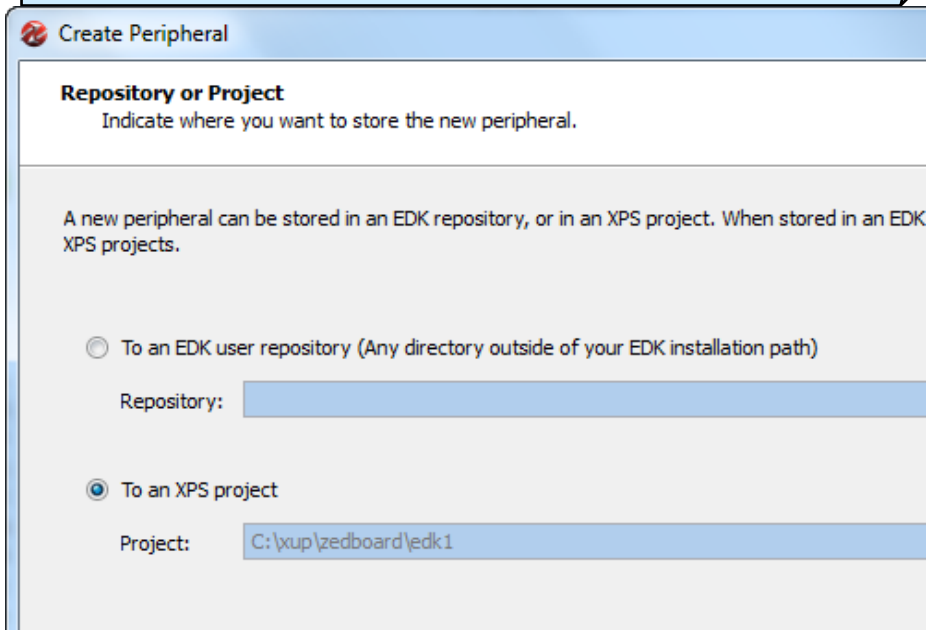
The Create and Import Peripheral Wizard can be started after creating a project and using **Hardware → Create or Import Peripheral …** or using **Start → All Programs → Xilinx Design Tools → ISE Design Suite 14.x → EDK → Tools → Create and Import Peripheral Wizard**

Select the Flow and Directory

1 Select Create Peripheral flow



2 Select the target directory – project directory or user repository

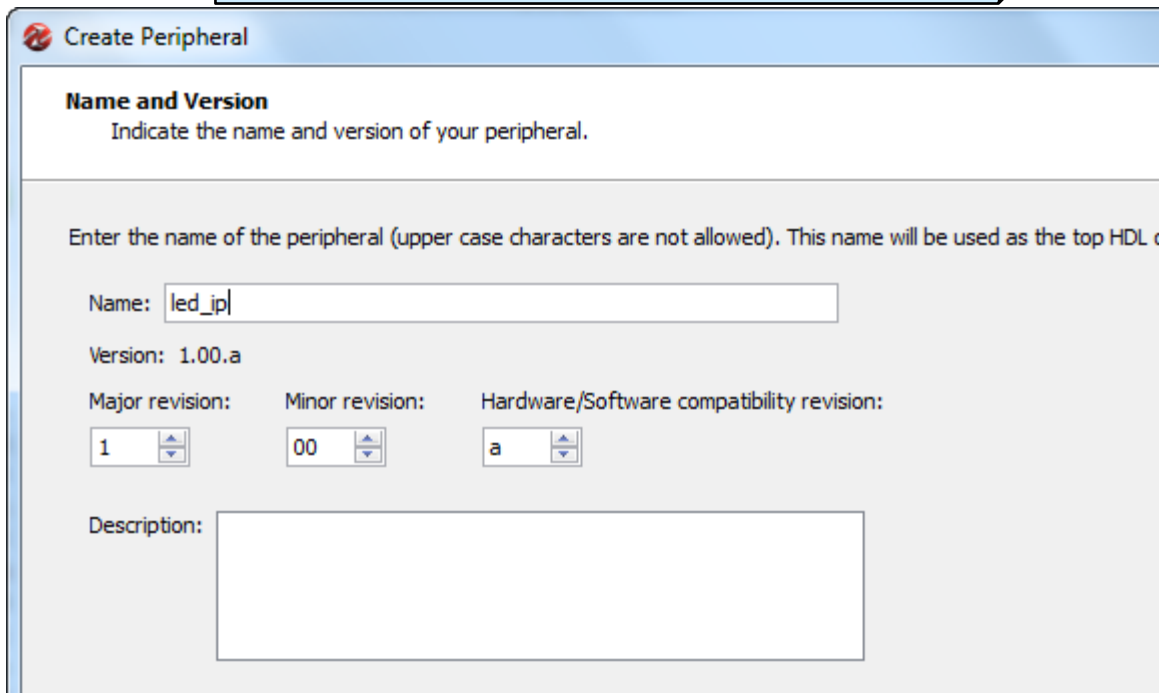


The project directory assigned as the target directory will allow the peripheral to be available to the project without importing it. User repository will allow multiple projects to access the same peripheral by importing it in a project

Selecting a Peripheral Name and Bus Interface

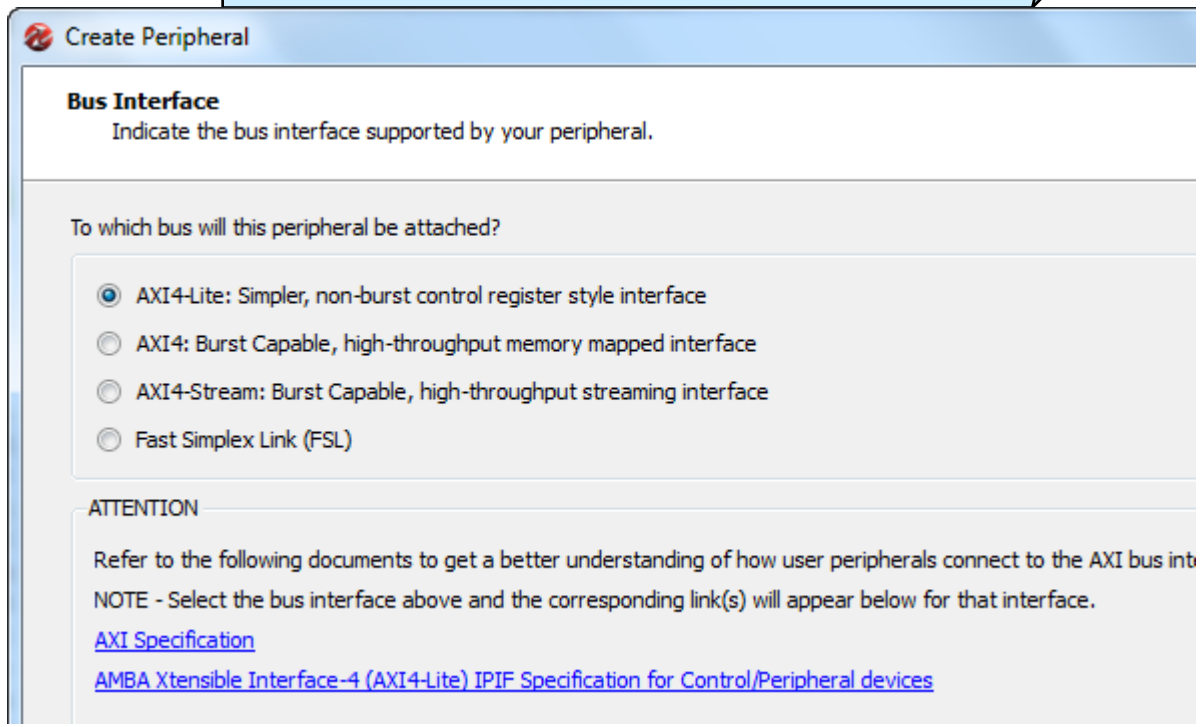
3

Provide the peripheral name and version



4

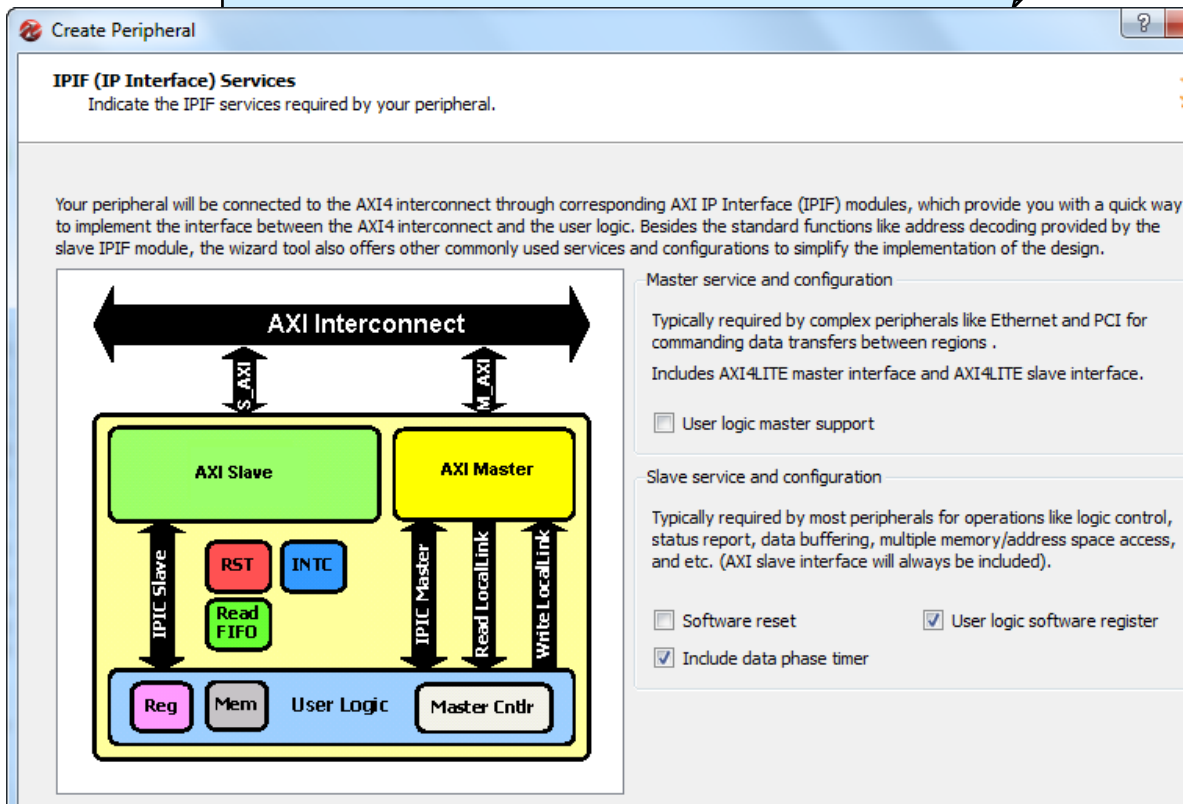
Select the interface to which the peripheral will attach



Selecting Various Services

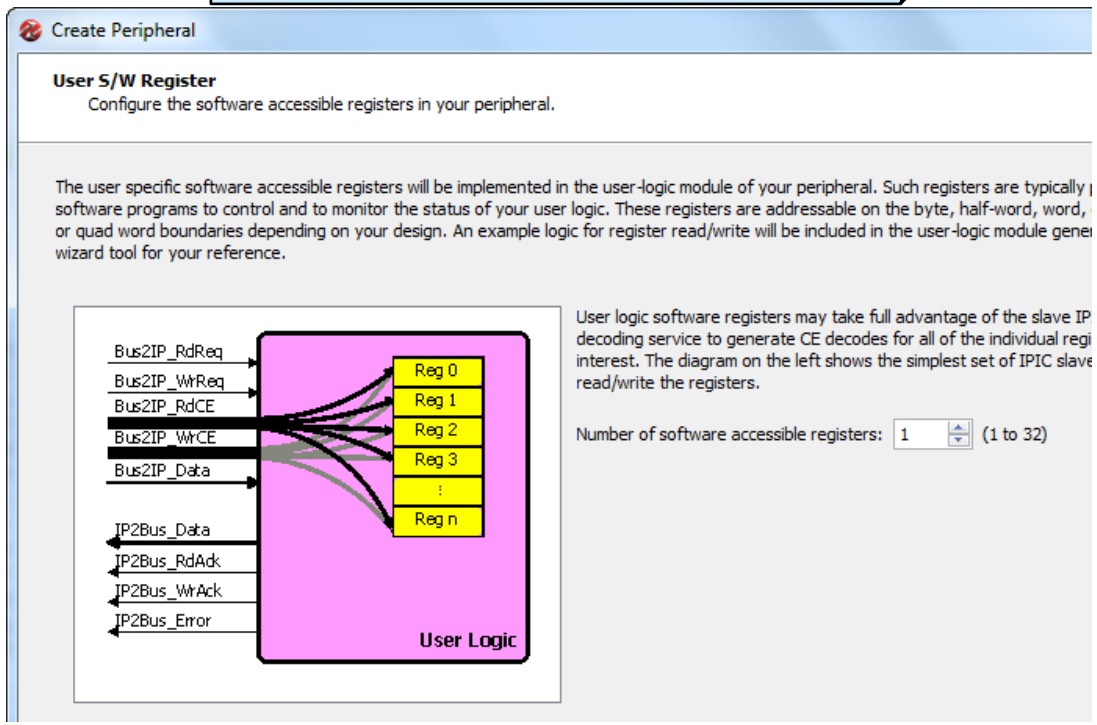
5

Select services



6

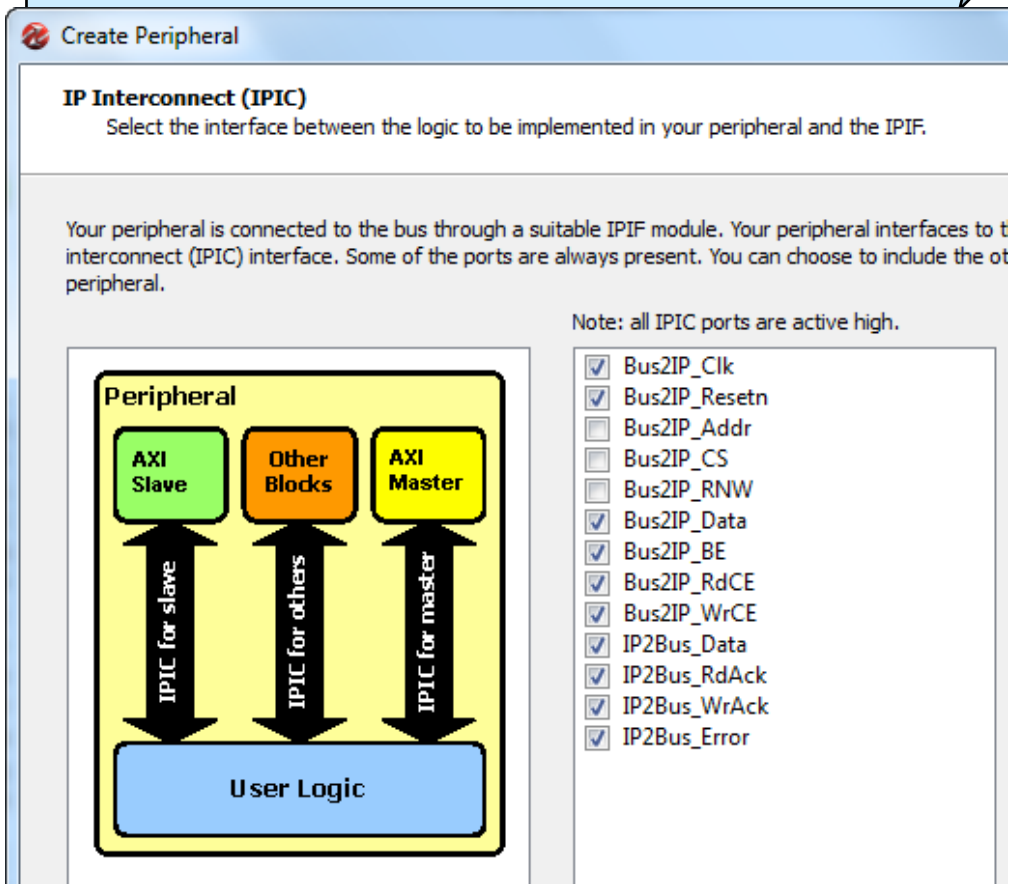
Configure the SW accessible registers



Select IPIC Signals and BFM

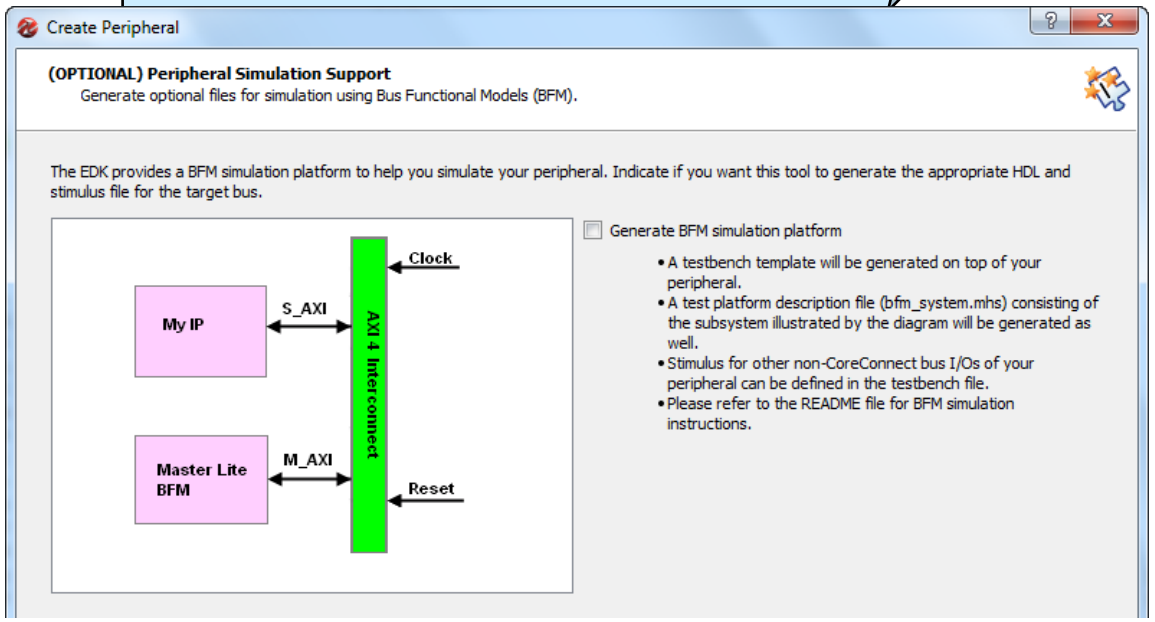
7

Select the IPIC signals available to the user logic



8

Optional Bus Functional Model Simulation Template

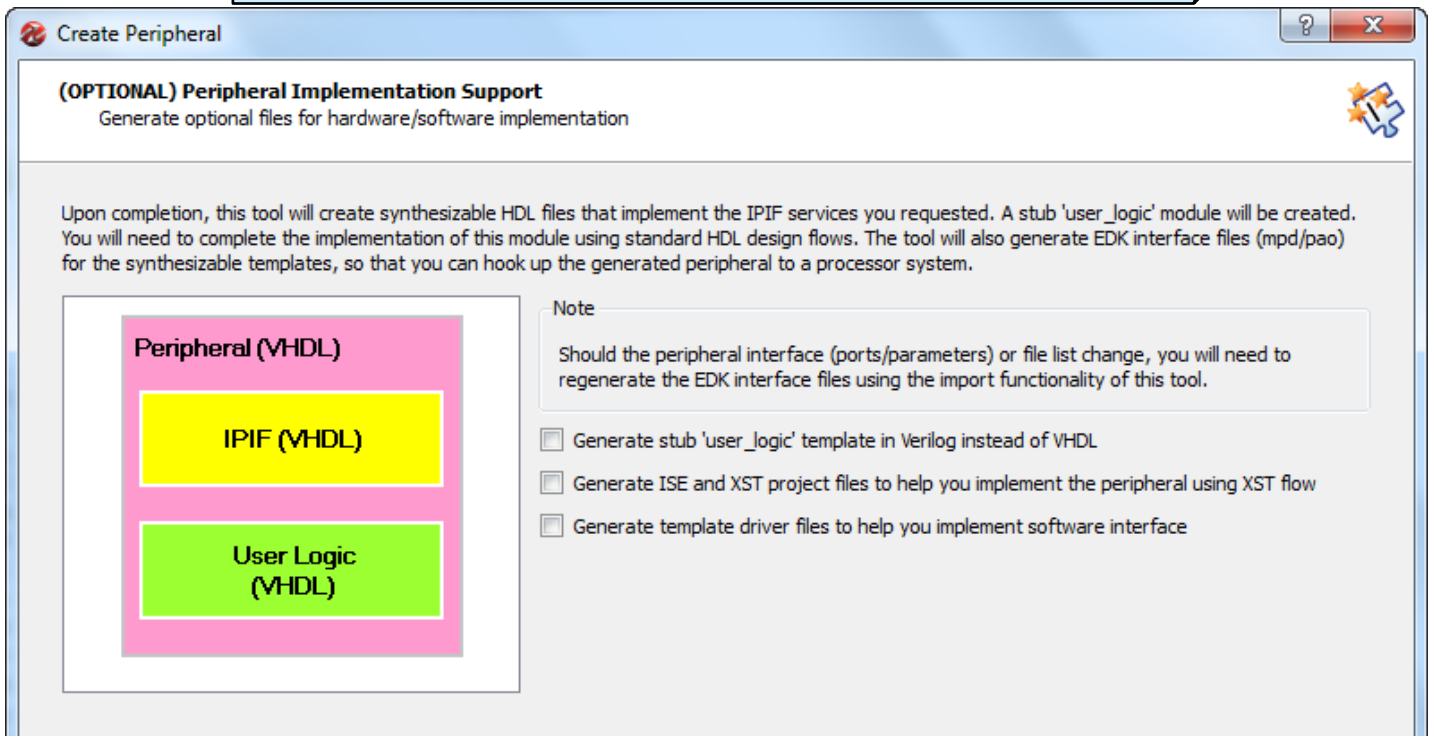


Select Optional Implementations Support

- **You can select to**
  - generate HDL in Verilog
  - generate project file so you can synthesize using XST and use ISE implementation tools
  - software drivers

9

Optional Implementation Tools Support

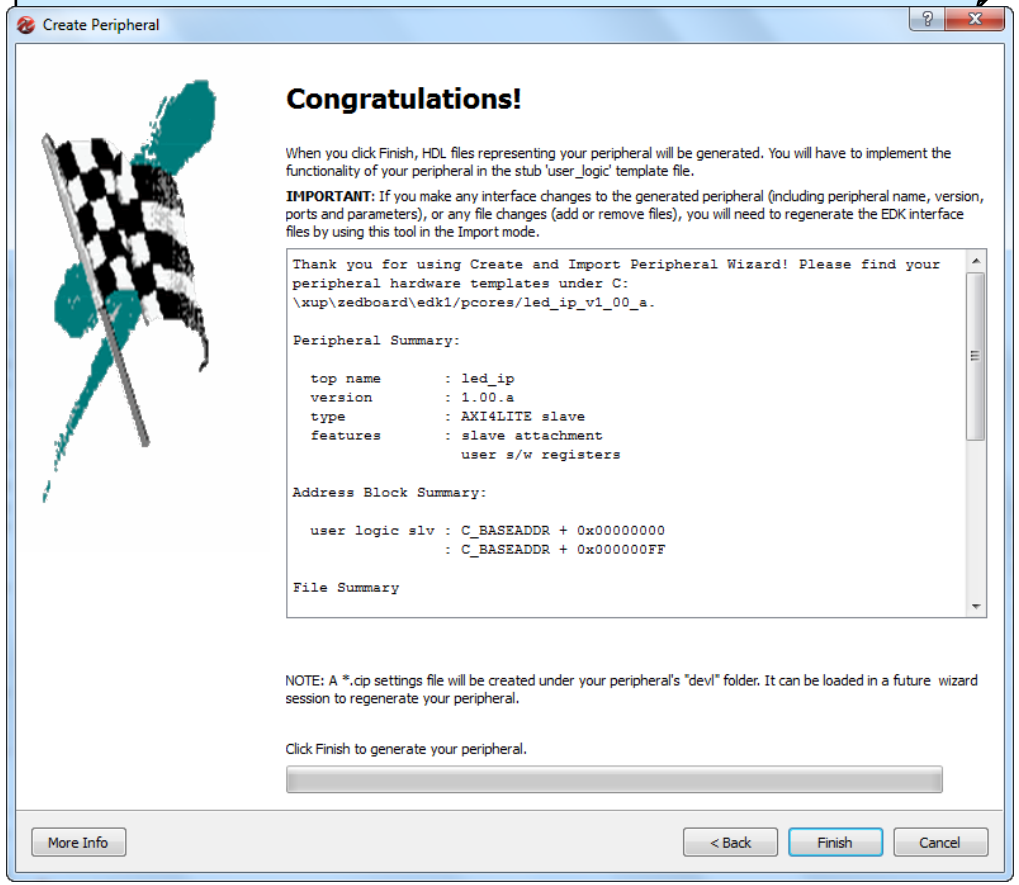


Generate Peripheral Template

- Since the project directory was assigned as the target directory the peripheral will appear in the IP Catalog under Project Local pcores folder

10

Finish



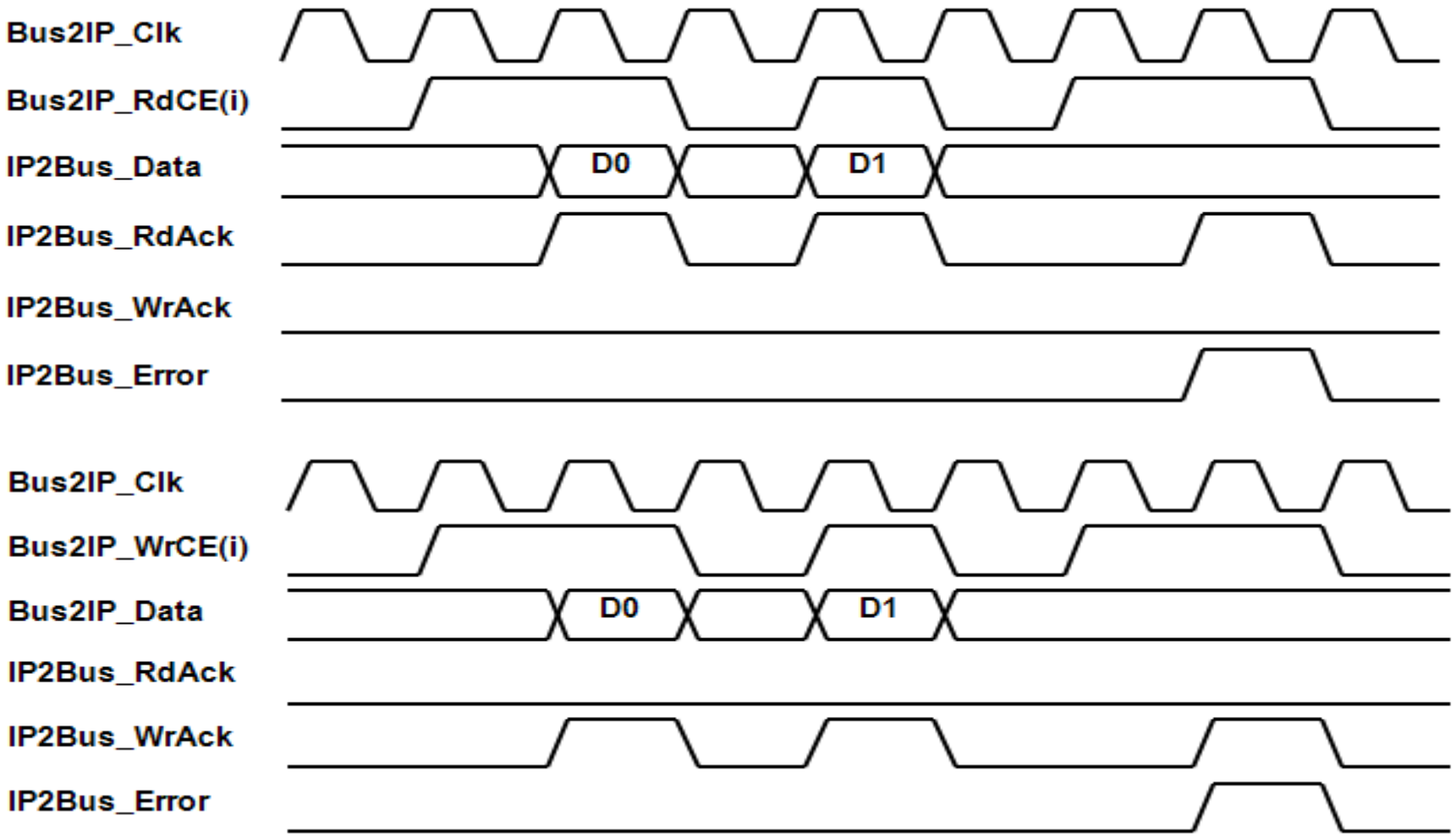
Outline

- AXI4 Transactions
- Create and Import Peripheral Wizard
- *Incorporating your IP functionality*
- Summary

Inside user_logic.vhd

- **Entity statement**
  - Add custom port signals
  - Add custom generics and parameters
- **Instantiate the rest of the design as a component**
- **Review the sample code provided for each option**
  - Registers implemented for *Bus2IP_WrCE* and *Bus2IP_RdCE* selects
  - Block RAM memory implemented for *Bus2IP_CS*
  - Example code to generate interrupts
  - Example code to transfer data between read/write FIFO
- **Modify/delete code to accommodate your application**
- **Only the needed IPIC signals will appear in *user_logic.vhd***

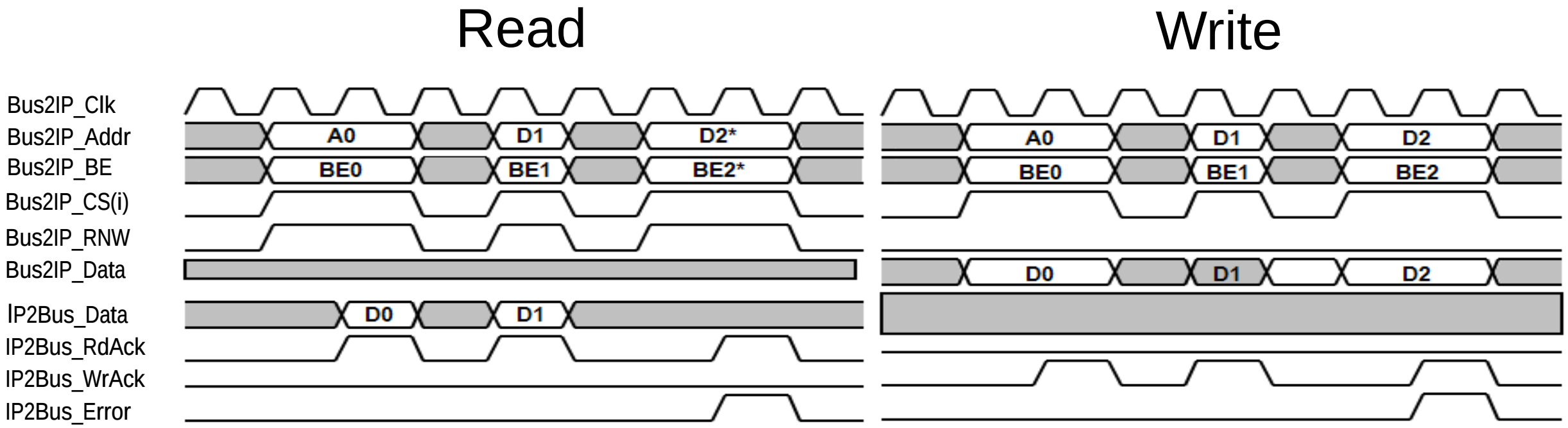
IPIC Register Read/Write



Read

Write

IPIC Memory Read/Write



Outline

- AXI4 Transactions
- Create and Import Peripheral Wizard
- Incorporating your IP functionality
- ***Summary***

Summary

- **AXI4 interface defines five channels**
  - All channels use basic VALID/READY handshake to complete a transfer
- **AXI Interconnect extends AXI interface by allowing 1-to-N, N-to-1, N-to-M, and M-to-N connections**
- **Custom IP can be created and/or imported using Create/Import Peripheral wizard**
- **The designer then needs to modify user_logic. vhd/v to include the desired functionality**
- **Bring up all user proprietary signals up though the hierarchy to the top-level *<ip_name>.vhd* file**
- **Modify mpd file to declare the user propriety signals so they are visible in System Assembly View**
- **Modify pao file so lower-level modules are compiled**