AXI MASTER

Design Document

Contents

1. Introduction

AXI Master block acts as an interface between the processor and the slave system for writing data into the Memory or reading the data from memory. AXI Master communicates with the SLAVE with AXI protocol at one side and communicates with processor at other side. The MASTER system can write into or read from memory using AXI Protocol. The data from the processor is latched in the MASTER system and data from the MASTER side is latched into the slave block and then passed to the memory side with enable and address.
The AXI Master block contains five independent channels depending on the operation

I. Write Address Channel
II. Write Data Channel
III. Write Response Channel
IV. Read Addresses Channel
V. Read Data Channel

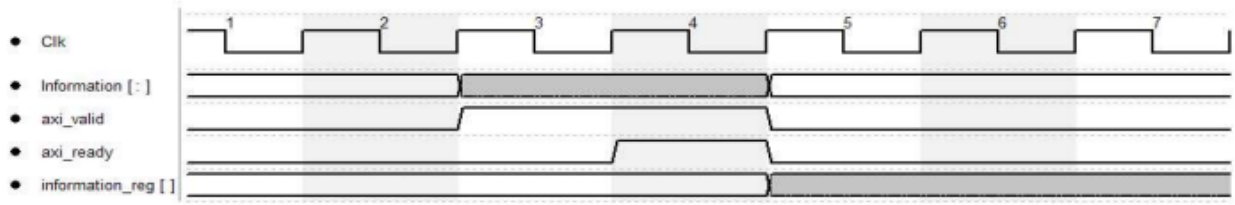
Each channel consists of a set of information signals and operates on handshaking mechanism.

The main responsibility of the AXI Master block is to accept the data from the Processor system and send that data to the SLAVE system according to AXI Protocol or receive the data from the SLAVE system which reads data from memory. Data from the MASTER system will be written into the memory according to the address and control signals. The data from the memory is read out to the MASTER system according to the address and control signals from the MASTER system. And the read data is received by the MASTER and is sent out to the Processor.

2. AXI Channel functionality

Handshake operation

All the five channels at Master side operate based on handshaking process to transfer the control signal or data. Source generates axi_valid signal to indicate the data or control signal is ready to be transferred and destination generates axi_ready signal to indicate that it is ready to accept the data or control information. Transfer occurs only after both valid and ready are high and data is registered in the next clock cycle.



Control signals

For MASTER system AXI Slave is burst based. Master should begin the transaction by driving control information and start the address of the transaction. Slave will calculate the addresses of subsequent transfers depending on the control information.

Burst Length

Burst length is specified by

Write Burst length = **axi_awlen[7:0] + 1.**

Read Burst length = **axi_arlen[7:0]+1.**

Burst length signal for both write and read transaction is of 8-bit. FIXED burst type supports 1 to 256 transfers and INCR burst type supports 1 to 256 transfers.

Burst type

AXI Slave is capable of transacting with two burst types,

i. FIXED burst type

ii. INCR burst type.

iii. WRAP burst type.

FIXED burst type

In this burst type the address will remain same for every transfer in the burst. Each transfer will be pointing to the same address(start address) throughout the burst length, sent by the MASTER system.

axi_awburst[1:0]/axi_arburst[1:0] = 2'b00 initiate fixed burst transmission.

INCR burst type

In this type the address will be incrementing for every transfer in the burst. The first transfer will be pointed to start address sent by MASTER system, next transfers will be pointing to the addresses calculated by the AXI Slave. Each transfer will give one increment to the address until the burst length is reached.

axi_awburst[1:0]/axi_arburst[1:0] = 2'b01 initiate INCR burst transmission.

WRAP burst type

Wrapping burst is similar to an incrementing burst, except that the address wraps around to a lower address if an upper address limit is reached. The following restrictions apply to wrapping bursts:

• the start address must be aligned to the size of each transfer
• the length of the burst must be 2, 4, 8, or 16 transfers.

Write strobe

The axi_wstrb[3:0] specify which byte lanes of the data bus contain valid information. Any of the bit high specify that byte starting from that bit position contain valid information.
Eg: **axi_wstrb** = 4'b0101 then **axi_wdata[23:16]** and **axi_wdata[7:0]** lanes contain valid information.

IDs

IDs are used to identify the slave. The AXI Slave has the ID as 4'b0011. Slave will accept the transaction only if the specified ID is present on the ID bus. All the channels have ID bus

axi_awid [3:0] – Write address ID
axi_wid [3:0] – Write data ID
axi_bid [3:0] – Bresponse ID
axi_arid [3:0] – Read Address ID
axi_rid [3:0] – Read Data ID.

All the ID signals should carry the same ID as AXI Slave ID (4'b0011).

3. Signal Description

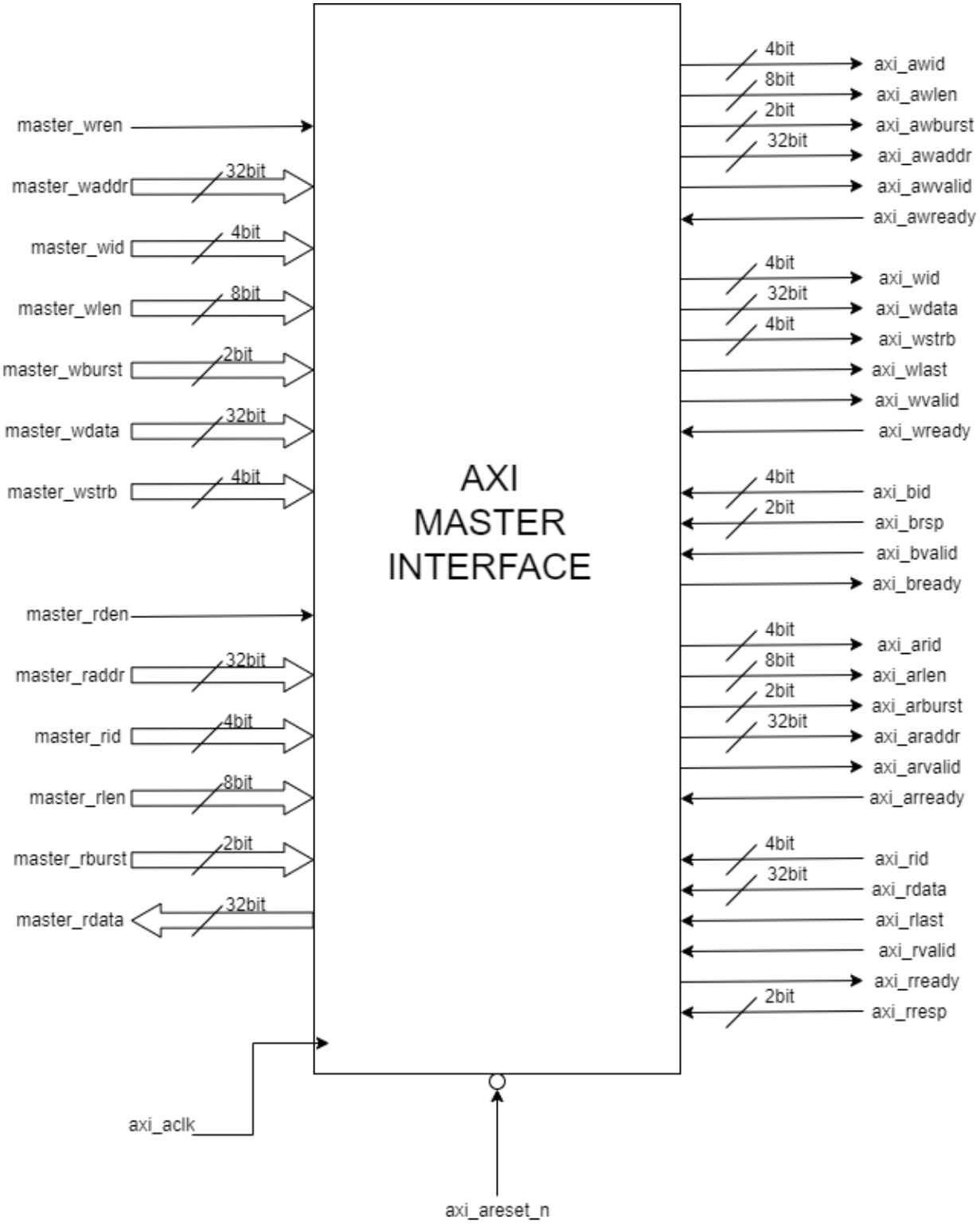
Pin Name	Default value	Direction	Width	Description
GLOBAL SIGNALS				
axi_areset_n	1’d1	Input	1-bit	Active low reset. Slave is in operating mode when the value of ARESETn is high. Pulling down the value to logic low will make the slave block to reset making all the outputs zero.
axi_aclk	1’d0	Input	1-bit	Global clock signal for AXI Master. Default clock signal for the master block
WRITE ADDRESS CHANNEL				
axi_awaddr	32’d0	Output	32-bit	Write Address. Write address bus carry the address required for write transactions
axi_awvalid	1’d0	Output	1-bit	Write Address Valid. Handshaking output from the master system indicating that valid address is present on the write address bus.
axi_awready	1’d0	Input	1-bit	Write Address Ready. Handshaking input by AXI slave indicating that slave is ready to accept the address present on adress bus.
axi_awid	4’d0	Output	4-bit	Write Address ID. Master should send the write address with valid slave ID on this signal to capture the write address at slave.
axi_awlen	8’d0	Output	8-bit	Write Burst Length. Master will specify the number of transfers(beats) in the write burst to the slave with this signal.
axi_awburst	2’d0	Output	2-bit	Write Burst Type. Master will specify the type of burst that will be used for transfer of write data with this signal.
WRITE DATA CHANNEL				

axi_wdata	32’0	Output	32-bit	Write Data. Write data bus carry the 32 bit data sent from master.
axi_wvalid	1’d0	Output	1-bit	Write Data Valid. Handshaking output from the master system indicating that valid data is present on the write data bus.
axi_wready	1’d0	Input	1-bit	Write Data Ready. Handshaking input produced from AXI slave indicating that slave is ready to accept the data present on adress bus.
axi_wid	4’d0	Output	4-bit	Write Data ID. Master sends the data to be written, with valid slave ID on this signal to capture the data at slave.
axi_wlast	1’d0	Output	1-bit	Write Last. Master produces write last signal along with the last beat of the burst transaction to indicate the end of the burst.
axi_wstrb	4’d0	Output	4-bit	Write Strobe. Master will specify which byte lines of the data bus contain valid data from this signal.
WRITE RESPONSE CHANNEL				
axi_bresp	2’b00	Input	2-bits	Write Response or B Response. B response bus carries responses for every transaction indicating the status of transaction.
axi_bvalid	1’d0	Input	1-bit	Bresponse Valid. Handshaking input produced from AXI slave indicating that valid B response is present on the response bus.
axi_bready	1’d0	Output	1-bit	Bresponse Ready. Handshaking output produced from the master system indicating that the master is ready to accept the response present on the response bus.
axi_bid	4’d0	Input	4-bit	Write Response ID. Slave sends the write response along with its valid ID on this signal.
READ ADDRESS CHANNEL				

axi_araddr	32’d0	Output	32-bits	Read Address. Read address bus carries the address required from which data has to be read.
axi_arvalid	1’d0	Output	1-bit	Read Address Valid. Handshaking output produced from the master indicating that a valid address is present on the read address bus.
axi_arready	1’d0	Input	1-bit	Read Address Ready. Handshaking input from AXI slave indicating that slave is ready to accept the read address present on adress bus.
axi_arid	4’d0	Output	4-bit	Read Address ID. Master sends the read address with valid slave ID on this signal to capture the read address at slave.
axi_arlen	8’d0	Output	8-bit	Read Burst Length. Master will specify the number of transfers(beats) should be there in the read burst from the slave with this signal.
axi_arburst	2’d0	Output	2-bit	Read Burst Type. Master will specify type of burst should slave use for transfer of read data with this signal.
READ DATA CHANNEL				
axi_rdata	32’d0	Input	32-bit	Read Data. Read data bus carries the 32 bit data sent from the slave.
axi_rvalid	1’d0	Input	1-bit	Read Data Valid. Handshaking input from AXI slave indicating that valid read data is present on read data bus.
axi_rready	1’d0	Output	1-bit	Read Data Ready. Handshaking output produced by the master indicating that master is ready to accept the data present on the read data bus.
axi_rid	4’d0	Input	4-bit	Read ID. Slave sends the read data along with its valid ID on this signal.
axi_rlast	1’d0	Input	1-bit	Read Last. Slave sends read last signal along with the last beat of burst transaction to indicate end of the burst.
axi_rresp	2’b00	Input	2-bit	Read response. This two bit signal indicates the status of the read transaction.

PROCESSOR TO MASTER INPUTS				
master_wren	1’d0	Input	1-bit	Write Enable. Write enable signal to master..
master_waddr	32’d0	Input	32-bit	Write Address. Carries the address of the location to which data has to be written.
master_wdata	32’d0	Input	32-bit	Write data. It carries the data to be written in memory. The data is written into the memory pointed by the write address.
master_wid	4’d0	Input	4-bit	Write ID. It carries the slave ID to which the data write operation is to be performed.
master_wlen	8’d0	Input	8-bit	Write Burst Length. Master will specify the number of transfers(beats) in the write burst.
master_wburst	2’d0	Input	2-bit	Write Burst Type. Master will specify the type of burst that will be used for transfer of write data.
master_rden	1’d0	Input	1-bit	Read Enable. Read enable signal to master.
master_rdata	32’d0	Input	32-bit	Read data. It carries the data to be read from memory. The data is readout from the memory pointed by the read address.
master_rid	4’d0	Input	4-bit	Read ID. It carries the slave ID from which the data Read operation is to be performed.
master_rlen	8’d0	Input	8-bit	Read Burst Length. Master will specify the number of transfers(beats) in the read burst.
master_rburst	2’d0	Input	2-bit	Read Burst Type. Master will specify the type of burst that will be used for transfer of read data.
master_rdata	32’d0	Output	32-bit	Read data. It carries the data to be read from memory. The data is readout from the memory pointed by the read address.

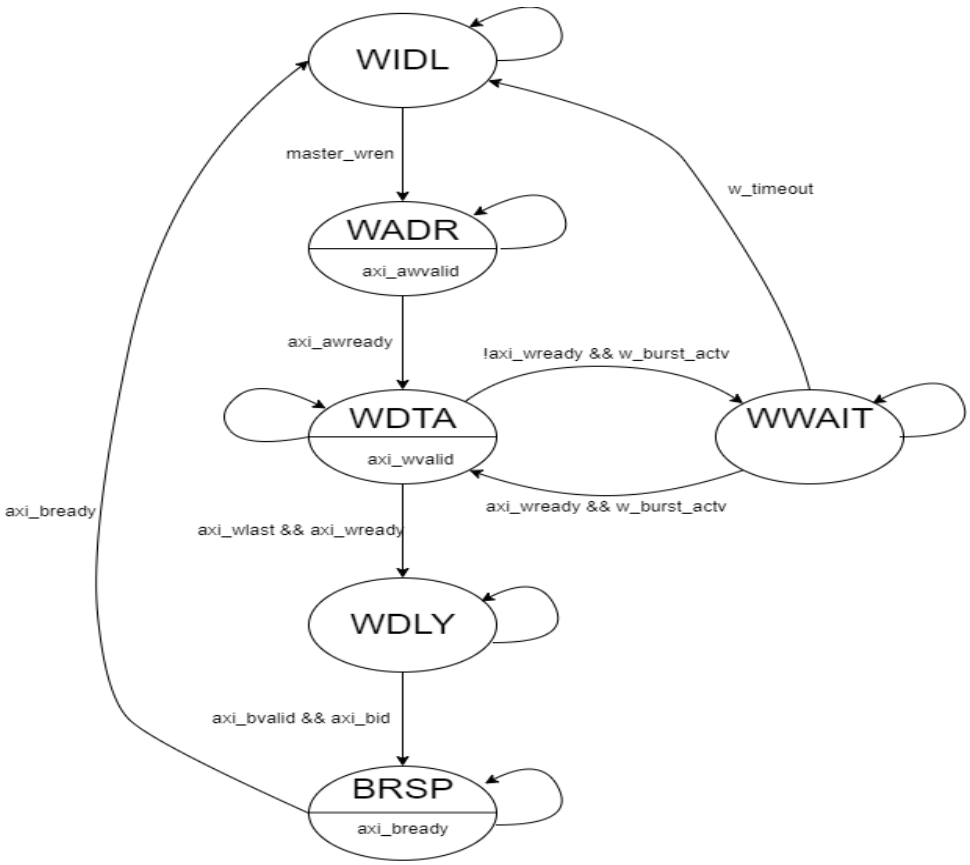
4. AXI Master Block Diagram



5. AXI Master FSMs

All the Handshaking signals are generated depending on write FSM and read FSM, write FSM produces handshaking signals for write address channel, write data channel and write response channel and read FSM produces handshaking signals for read address channel and read data channel.

Write FSM

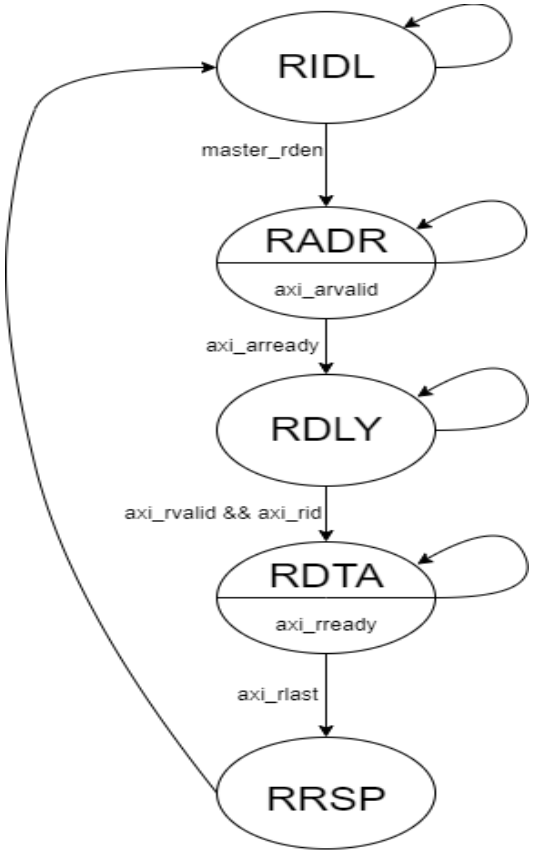


All the handshaking signals from the master for the write operation to the slave system are generated by the Write FSM. Write handshaking signals from Slave system are inputs to the write FSM and each state of the FSM is responsible for generating respective handshake signal to the Slave system.

- For valid address on the address line FSM produces axi_awvalid signal along with axi_awaddr to indicate that valid address is present at the awaddr bus.
- For axi_wvalid signal, the FSM asserts wvalid along with write data on axi_wdata bus indicating that valid data is present on the wdata bus. The valid is asserted only when the master receives awready which indicates that the address sent in address line is registered in the slave.

- MASTER system will assert axi_wlast signal along with last beat (single transfer in a burst) of the burst, FSM produces axi_wlast based on the length of the transaction. The along with last beat of the data burst the wlast signal is asserted.
- When the data burst is received by the slave it produces Bresponse indicating that all the data is received by the slave along with axi_bvalid request, th FSM produces axi_bready signal indicating that it is ready to receive th Bresponse from the slave system.

Read FSM



All the handshaking signals from the master for the read operation to the Slave system are generated by the read FSM. Read handshaking signals from the slave system are inputs to the read FSM. Each state of the FSM is responsible for generating a respective handshake signal to the Slave system.

- The axi_arvalid request is produced from FSM axi_arready grant signal indicating that the slave is ready to accept the address is received in response to the valid signal..
- The master sends the axi_araddr signals that are received from the processor along with the arvalid request to the slave module.
- AXI Master asserts axi_rready signal in response to the axi_rvalid request from the Slave.

- Read response (axi_rresp[1:0]) is sent when the last data is accepted by the MASTER system.

6. Verification goals

For AXI Master block Test bench acts as a Slave system as well as Processor block. Since AXI Master is an interface block, the testbench should cover the ports at both sides of the master block.

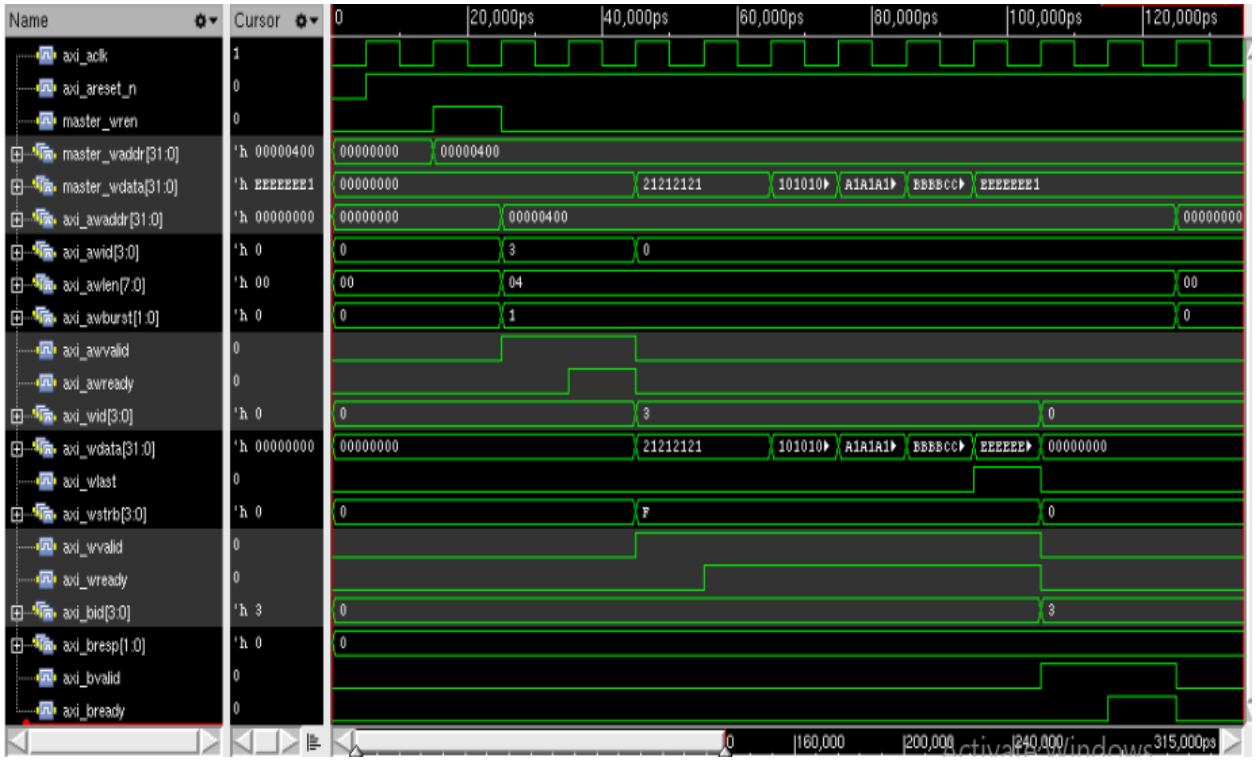
- Master should be capable of generating handshaking signals for all the five channels.
- Information should be registered in the master system and then sent through buses to the slave..
- The control signals should be sent properly along with the address and data..
- The data read by the slave should be registered in the master and sent to to the processor as output.

7. Test Cases

Serial No.	Test Case	Results
1.	Axi_awaddr is sent without wren signal	When the wren signal is not sent the transaction is not initiated and the master system does not go to the address state.
2.	Axi_awready is sent after some time delay	The master system maintains the address along with the awvalid signal until the awready is received..
3.	Axi_bvalid is kept asserted for three clock cycles.	The bready signal is asserted after receiving the bvalid for one clock cycle.
4.	Axi_araddr is sent without rden signal	The Master system does not initiate the read operation since the rden was not asserted.
5.	Axi_rvalid is not asserted for some time after araddr is sent	The system starts registering the data only after rvalid is received and ready is produced from the system.
6.	Axi_wready is not asserted	The master system waits until the timeout signal is produced and then goes back to the idle state.

8. Simulation Results

Write Burst



Read Burst