Design of Multi Master Single Slave AXI4 Protocol for RISC-V Processor

A Project Report
submitted by

Segu Suresh

in partial fulfillment of the requirements
for the award of the degree of
Master of Technology

Under the guidance of
Dr. V. Kamakoti

Department of Electrical Engineering
Indian Institute of Technology Madras

May 2015

Department of Electrical Engineering
Indian Institute of Technology Madras

THESIS CERTIFICATE

This is to certify that the thesis titled **Design of Multi Master Single Slave AXI4 Protocol for RISC-V Processor**, submitted by **Segu Suresh**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. V. Kamakoti
Project Guide
Associate Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place : Chennai
Date :

ACKNOWLEDGEMENT

I Would like to express my deepest gratitude to my guide, ***Dr. V. Kamakoti*** for his valuable guidance, encouragement and advice. His immense motivation helped me in making firm commitment towards my project work.

My special thanks to ***Mr. G.S. Madhusudan*** for his encouragement and motivation through out the project. His valuable suggestions and constructive feedback were very helpful in moving ahead in my project work.

I would like to thank my faculty advisor ***Dr. Nagendra Krishnapura*** who patiently listened, evaluated, and guided us through out our course.

My special thanks to project team members Neel, Arjun , Rahul, Anand, venkatesh, sukrat,Laxmeesha, naidhu,Sunnihith for their help and support.

Abstract

System-on-a-Chip (SoC) design has become more and more complex because Intellectual Property (IP) core with different functions are integrated within a single chip. Each IPs have completed design and verification but the integration of all IPs may not work together. The more common problem is violation bus protocol or transaction error. The bus-based architecture has become the major integration methodology for implementing a SoC. The main issue is to ensure that the IP works correctly after integrating to the corresponding bus architecture.Advanced extensible interface 4 (AXI4) is an on chip bus architecture introduced by ARM to interact with its peripherals.

AXI4 bus architecture is an SoC communication protocol that aims at high performance and low power consumption by partitioning based on the bandwidth with which the devices operate, within the system.

This project thesis talks about the design of Advanced Extensible Interface(AXI) 4.0 in a four master single slave system, and implemented in Bluespec sysytem verilog. Which supports data burst of maximal length of 16 transfers, multiple outstanding transactions and unaligned data transfers using byte strobes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

To speed up SoC integration and promote IP reuse, several bus-based communication architecture standards have emerged over the past several years. Since the early 1990s, several on-chip bus-based communication architecture standards have been proposed to handle the communication needs of emerging SoC design. Some of the popular standards include ARM Microcontroller Bus Architecture (AMBA) versions 2.0 and 3.0.

ARM introduced the Advanced Michocontroller Bus Architecture (AMBA) 4.0 specifications in March 2010, which includes Advanced eXtensiable Interface (AXI) 4.0. AMBA bus protocol has become the de facto standard SoC bus. That means more and more existing IPs must be able to communicate with AMBA 4.0 bus.

The AXI specifications describe an interface between a single AXI master and a single AXI slave, representing IP cores that exchange information with each other. Memory mapped AXI masters and slaves can be connected together using a structure called an Interconnect block. The simple block diagram of AXI bus having single master and slave is shown figure 1.1.

The main components are:

AXI Master: AXI master initiates transfer on the bus, and receives data.

AXI Slave: AXI slave responds to the transaction initiated by the master. Fully configured wait states and slave error response. Storing ability as internal memory and supplying data on the demand.

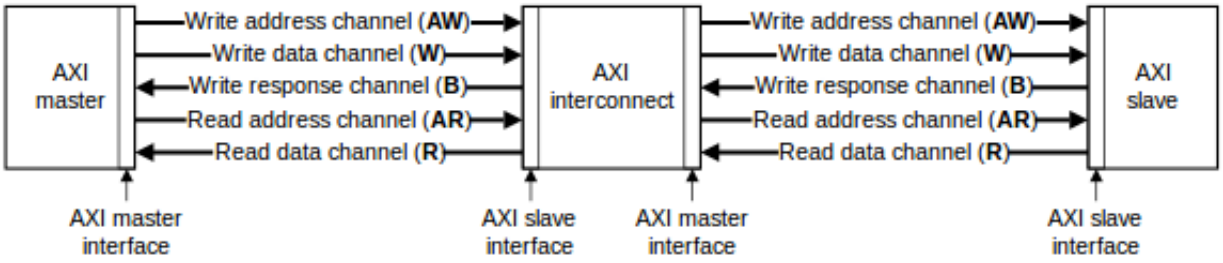

Figure 1.1: Different channels of AXI bus

AXI Interconnect/Arbiter: AXI interconnect can take multiple masters at a time and arbitrates for the bus using configurable priority scheme.

It consist of five different channels:

- Read Address Channel
- Write Address Channel
- Read Data Channel
- Write Data Channel
- Write Response Channel

Read and write channels are independent, so data can move in both directions between the master and slave simultaneously, and data transfer sizes can vary. The limit in AXI4 is a burst transaction of up to 256 data transfers. Each transaction is burst-based which has address and control information on the address channel that describes the nature of the data to be transferred. The data is transferred between master and slave using a write data channel to the slave or a read data channel to the master.

1.1 About AXI4 protocol

The AMBA AXI4 protocol is aimed towards high-frequency system designs and includes a number of features that make it suitable for a high speed sub -micron interconnect.

The key features of the AXI4 protocol are:

- Separate address/control and data phases
- Support for unaligned data transfers using byte strobes
- Burst-based transactions with only start address issued, so it can reduces the utilization of address channel
- separate read and write data channels to enable low-cost Direct Memory Access (DMA)
- Ability to issue multiple outstanding addresses
- out-of-order transaction completion
- easy addition of register stages to provide timing closure

The AXI4 protocol supports the following mechanisms:

- variable-length bursts, from 1 to 16 data transfers per burst
- bursts with a transfer size of 8,16,32,64.128,256 and 1024 bits
- wrapping, incrementing, and non-incrementing bursts
- atomic operations, using exclusive or locked accesses
- system-level caching and buffering control

1.2 Channel Architecture

The AXI protocol supports burst-based transactions. Every transaction has address and control information on the write and read address channel that describes the nature of the data to be transferred between master and slave.

In read transactions of the AXI protocol, the master drives address and control information on Read address channel to the slave and slave drives the data and read response signal to the master on Read data channel. Figure 1-2 shows how a read transaction uses the read address and read data channels.

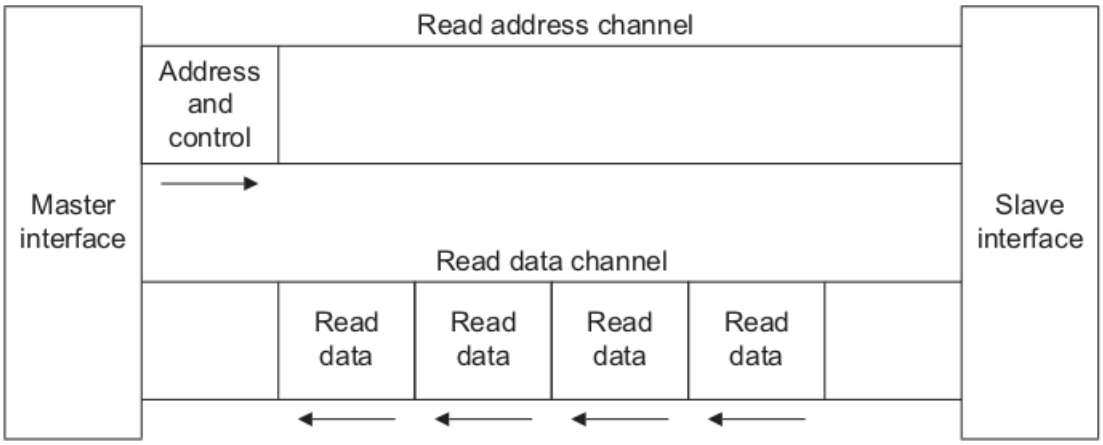

Figure 1.2: Channel architecture of read

In write transactions of the AXI protocol, all the data flows from the master to the slave, and it has an additional write response channel to allow the slave to acknowledge the master about the completion of the write transaction. Figure 1-3 shows how a write transaction uses the write address, write data, and write response channels.

The AXI protocol enables:

- address information to be issued ahead of the actual data transfer
- support for multiple outstanding transactions
- support for out-of-order completion of transactions.

1.2.1 Channel definition

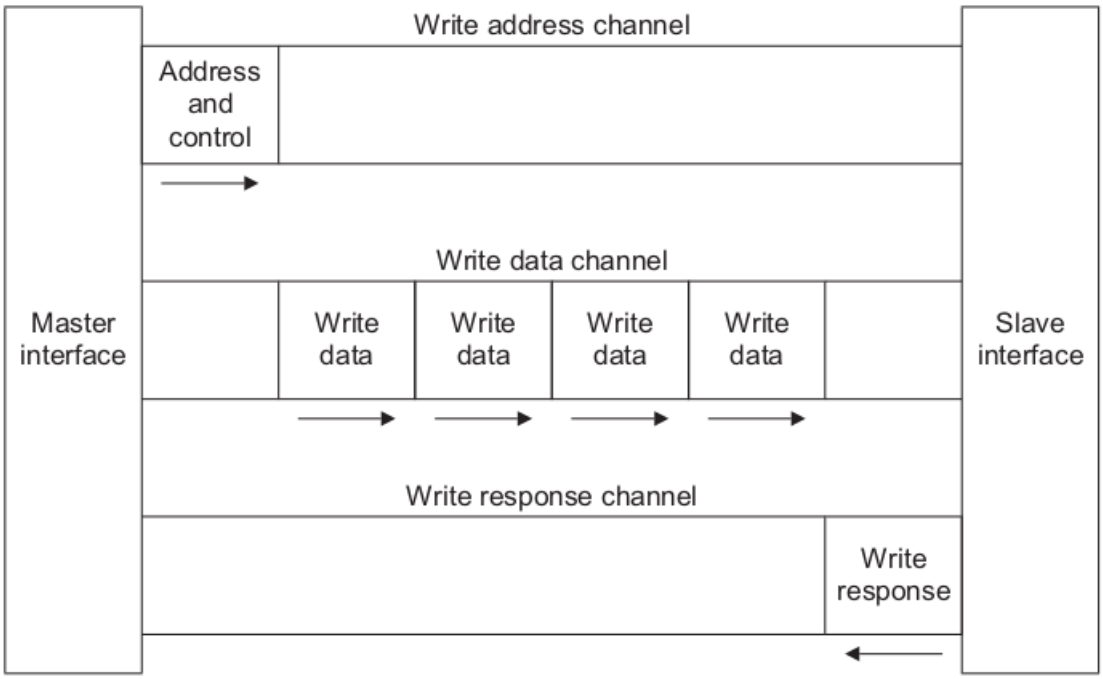

Figure 1.3: Channel architecture of write

AXI protocol five channel are independent to each other, consists of a set of information signals and uses a two-way VALID and READY handshake mechanism.

The information source uses the VALID signal to show when valid address and control information or data is available on the channels. The destination uses the READY signal to show when it can accept the data. Both the read data channel and the write data channel also include a LAST signal to indicate when the transfer of the final data item within a transaction takes place.

Read and write address channels

In AXI protocol Read and write transactions have their own address channel. The address channels carries the address and control information for a transaction. The address is nothing but the starting address of transaction and control information tells what is the length, size and type of transaction.

Read data channel

The read data channel conveys both the read data and read response information from the slave back to the master. The read data channel :

- data bus can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide
- read response indicating the completion status of the read transaction.

Write data channel

The write data channel conveys the write data and strobe signal from the master to the slave and includes:

- the data bus, that can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide
- one byte lane strobe for every eight data bits, indicating which bytes of the data bus are valid.

Write data channel information is buffered, so that the master can send another write transactions without slave acknowledgement of previous write transactions.

Write response channel

The write response channel provides a way for the slave to respond to write transactions. All write transactions use completion signaling. The completion signal occurs once for each burst, not for each individual data transfer within the burst.

1.2.2 Interface and interconnect

Any system may consists of a number of master and slave devices. They connected together through some form of interconnect. This project have four masters and single slave connected through interconnect as shown in Figure 1-4.

The AXI protocol provides a single interface definition for describing interfaces:

- between a master and the interconnect in multi master and multi slave, or multi master and single slave systems.

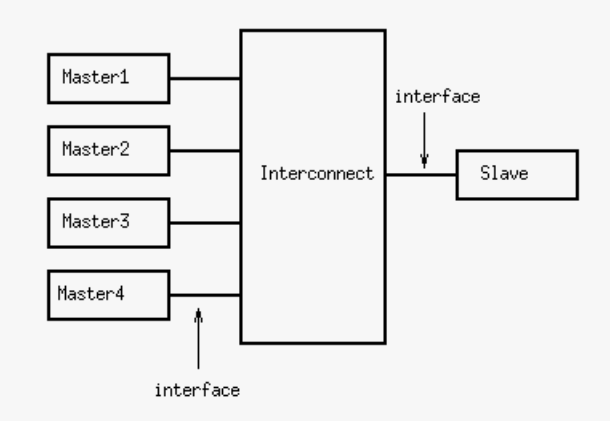

Figure 1.4: Interface and interconnect

- between a slave and the interconnect in multi master and multi slave, or multi master and single slave systems.
- between a master and a slave in single master and slave system.

The interface definition enables a variety of different interconnect implementations between masters and slaves. The interconnect between devices is equivalent to another device with symmetrical master and slave ports to which real master and slave devices can be connected.

Most systems use one of the following three interconnect approaches:

- shared address and data buses, in a systems where address and data channel bandwidth requirement is less.
- shared address buses and multiple data buses, in a systems where address channel bandwidth requirement is less than data channel.
- multilayer, with multiple address and data buses, in a systems where address and data channel bandwidth requirement is high.

1.2.3 Register slices

All channel in AXI protocol transfers information in only one direction, so there is no need for fixed relationship between various channels. This enables the insertion of register slices in any channel and maximize the operating frequency, at the cost of additional cycle latency.

It is also possible to use register slices at almost any point within a given interconnect. It can be advantageous to use a direct, fast connection between a processor and high-performance memory, but to use simple register slices to isolate a longer path to less performance-critical peripherals.*

1.3 Basic transactions

This section explains basic transaction of AXI protocol. All transactions uses a two-way VALID and READY handshake mechanism. The transfer of address information or data occur when both VALID and READY handshake signals are HIGH between master and interconnect or interconnect and slave.

1.3.1 Read Burst

Figure 1-5 shows a read burst transaction with four transfers. As show in figure the master drives the address and slave accepts it after one cycle. The master also drives control information, but it not shown on figure for clarity.

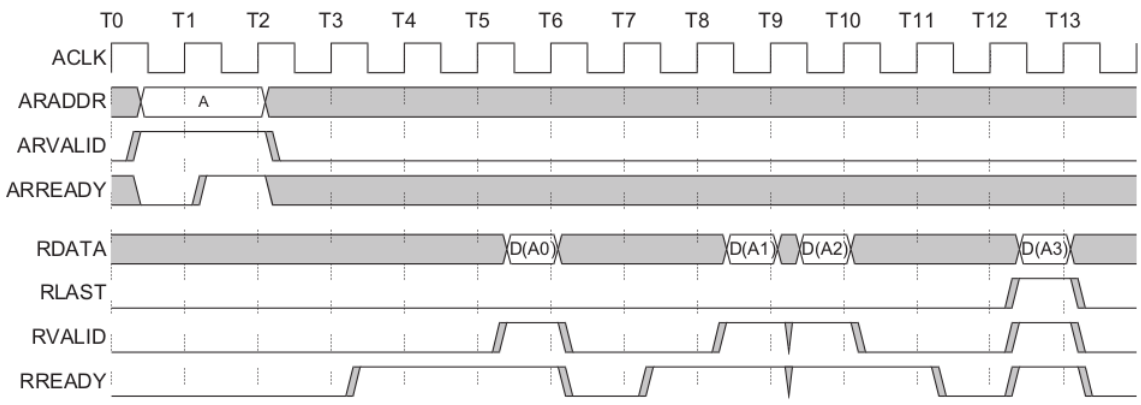

Figure 1.5: Read burst

After address appear at slave, slave reads the data from memory and assert the VALID signal, then transfers the data on read data channel. The slave keeps the VALID signal LOW until the read data is available. At the final data transfer of the burst transaction, the slave asserts the RLAST signal to show that the last data item is being transferred.

1.3.2 Overlapping read burst

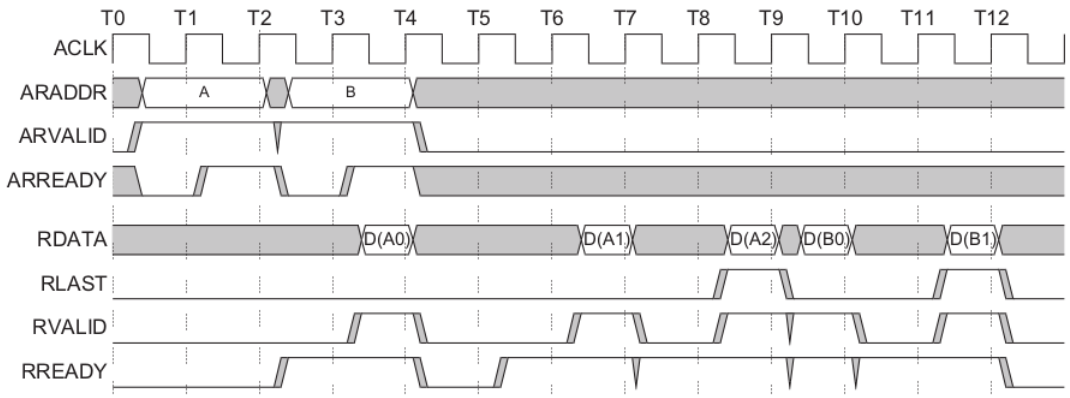

Figure 1.6: Overlapping read burst

As shown in fig 1-6 master is driving another burst address before slave accepting the first address. The slave can process data for second burst in parallel with the completion of the first burst.

1.3.3 Write burst

Write transaction is show in fig 1-7. It starts when the master sends an address and control information on the write address channel. The master then sends each item of write data over the write data channel. When the master sends the last data item, the WLAST signal goes HIGH. When the slave accepted all the data items, it drives a write response back to the master to indicate that the write transaction is complete.

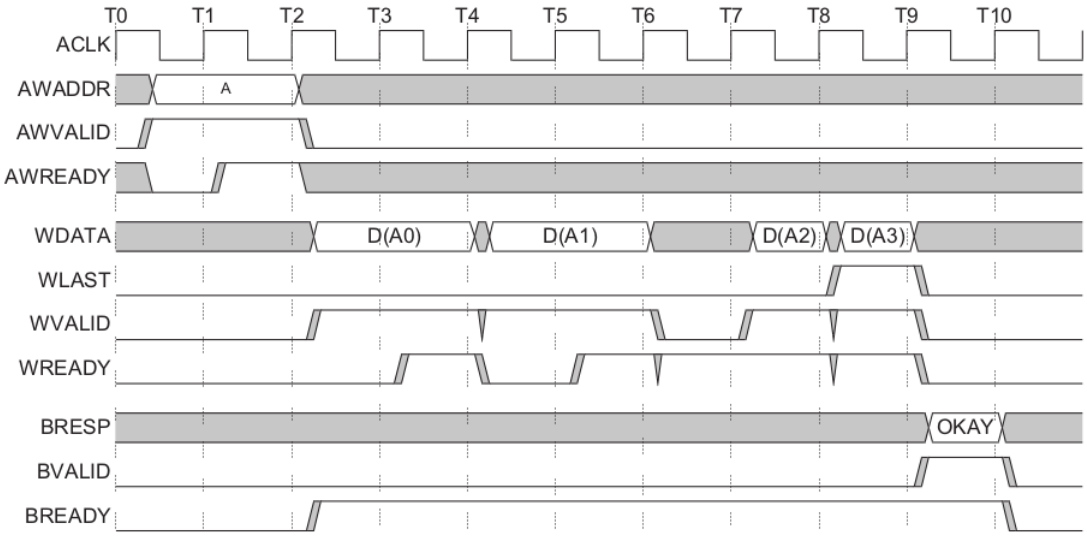

Figure 1.7: Write burst

1.3.4 Transaction ordering

The AXI protocol enables out-of-order transaction completion. It gives an ID tag to every transaction across the interface. The protocol requires that transactions with the same ID tag are completed in order, but transactions with different ID tags can be completed out of order.

Out-of-order transactions can improve system performance in two ways:

- The interconnect can enable transactions with fast-responding slaves to complete in advance of earlier transactions with slower slaves.
- Complex slaves can return read data out of order. For example, a data item for a later access might be available from an internal buffer before the data for an earlier access is available.

If a master requires that transactions are completed in the same order that they are issued, then they must all have the same ID tag. If, however, a master does not require in-order transaction completion, it can supply the transactions with different ID tags, enabling them to be completed in any order.

In a multi master system, the interconnect is responsible for appending additional information to the ID tag to ensure that ID tags from all masters are unique. The ID tag is similar to a master number, but with the extension that each master can implement multiple virtual masters within the same port by supplying an ID tag to indicate the virtual master number.

Although complex devices can make use of the out-of-order facility, simple devices are not required to use it. Simple masters can issue every transaction with the same ID tag, and simple slaves can respond to every transaction in order, irrespective of the ID tag

Chapter 2

Signal Descriptions

This chapter defines the AXI signals. Although bus width and transaction ID width are implementation-specific, the tables in this chapter show a 32-bit data bus, a four-bit write data strobe, and four-bit ID fields.

2.1 Write address channel

The write address channel signals and description of each signal is shown in figure 2.1

2.2 Write data channel signals

The write data channel signals and description of each signal is shown in figure 2.2

2.3 Write Response channel signals

The write Response channel signals and description of each signal is shown in figure 2.3

Table 2.1: AXI write address channel signals

Signal	Source	Description
AWID[3:0]	Master	This id tag signal. It tells which master is driving address and control information on write address channel.
AWADDR[31:0]	Master	This is Write address signal. It gives the address of the first transfer in a write burst transaction.
AWLEN[3:0]	Master	Burst length. It gives the exact number of transfers in a burst
AWSIZE[2:0]	Master	Burst size. This signal indicates the size of each transfer in the burst
AWBURST[1:0]	Master	Burst type. This tells how the address for each transfer within the the burst is calculated
AWVALID	Master	Write address valid. This signal indicates whether the valid write address and control information are available or not
AWREADY	Slave	Write address ready. This signal indicates whether the slave is ready to accept an address and control information or not

2.4 Read address channel signals

The Read address channel signals and description of each signal is shown in figure 2.4

2.5 Read data channel signals

The Read data channel signals and description of each signal is shown in figure 2.5

Table 2.2: AXI write data channel signals

Signal	Source	Description
WID[3:0]	Master	This id tag signal. It tells which master is driving the data on write data channel.
WDATA[31:0]	Master	This is Write data signal. The write data bus can be 8, 16, 32, 64, 128, 256, 512,or 1024 bits wide.
WSTRB[3:0]	Master	Write strobes. This signal indicates which byte lanes to update in memory. There is one write strobe for each eight bits of the write data bus.
WLAST	Master	Write last. This signal indicates the last transfer in a write burst.
WVALID	Master	Write data valid. This signal indicates whether the valid data is available or not.
WREADY	Slave	Write address ready. This signal indicates whether the slave is ready to accept data or not.

Table 2.3: AXI write Response channel signals

Signal	Source	Description
BID[3:0]	Slave	Write Response ID. It tells to which master, slave driving the Write Response signal. The BID value must match the AWID value of the write transaction to which the slave is responding.
BRESP[1:0]	Slave	Write response. This signal indicates the status of the write transaction.
BVALID	Slave	Write Response valid. This signal indicates that a valid write response is available: 1 = write response available 0 = write response not available.
BREADY	Master	Write Response ready. This signal indicates whether the master is ready to accept Write response signal or not

Table 2.4: Read address channel signals

Signal	Source	Description
ARID[3:0]	Master	This id tag signal. It tells which master is driving address and control information on read address channel.
ARADDR[31:0]	Master	This is read address signal. It gives the address of the first transfer in a write burst transaction.
ARLEN[3:0]	Master	Burst length. It gives the exact number of transfers in a burst.
ARSIZE[2:0]	Master	Burst size. This signal indicates the size of each transfer in the burst.
ARBURST[1:0]	Master	Burst type. This tells how the address for each transfer within the the burst is calculated.
ARVALID	Master	Read address valid. This signal indicates whether the valid write address and control information are available or not
ARREADY	Slave	Read address ready. This signal indicates whether the slave is ready to accept an address and control information or not

Table 2.5: AXI read data channel signals

Signal	Source	Description
RID[3:0]	Slave	This id tag signal. It tells to which master the slave driving the data on read data channel.
RDATA[31:0]	Slave	This is Read data signal. The Read data bus can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide.
RLAST	Slave	Read last. This signal indicates the last transfer in a read burst.
RVALID	Master	Read data valid. This signal indicates whether the valid data is available or not .
RREADY	Master	Read address ready. This signal indicates whether the master is ready to accept data or not.

Chapter 3

Channel Handshake

This chapter describes the master to AXI interconnect and AXI slave handshake process and outlines the relationships and default values of the READY and VALID handshake signals

3.1 Handshake process

All five channels of AXI protocol use the same VALID/READY handshake signals to transfer address and control information, or data. This two-way flow control mechanism enables the master, interconnect and slave, to control the rate at which the data and control information moves.

The source asserts the VALID signal when it's driving the valid address and control information or data on respective channels. The destination generates the READY signal to indicate that it is accepts the address and control information or data. Transfer occurs only when both the VALID and READY signals are HIGH.

Figure 3-1 to Figure 3-3 show examples of the handshake sequence. In Figure 3-1, the source presents the data or control information and drives the VALID signal HIGH. The data or control information from the source remains stable until the destination drives the READY signal HIGH, indicating that it accepts the data or control information. The arrow shows when the transfer occurs.

It is not permitted to wait until READY is asserted before asserting VALID. Once VALID is asserted it must remain asserted until the handshake occurs.

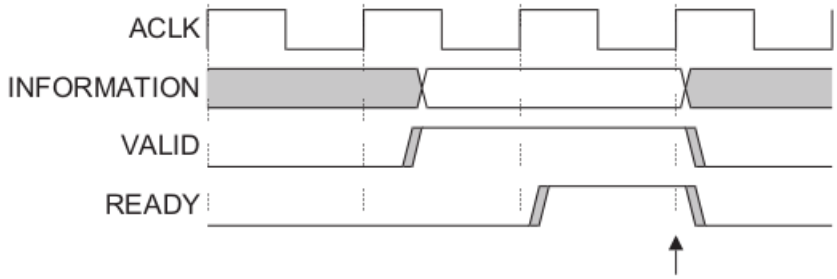

Figure 3.1: VALID before READY handshake

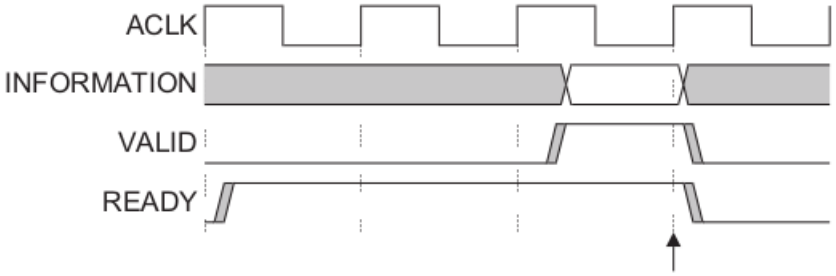

Figure 3.2: READY before VALID handshake

In Figure 3-2, the destination drives READY HIGH before the data or control information is valid. This indicates that the destination can accept the data or control information in a single cycle as soon as it becomes valid. The arrow shows when the transfer occurs.

It is permitted to wait for VALID to be asserted before the corresponding READY is asserted. If READY is asserted, it is permitted to deassert READY before VALID is asserted.

In Figure 3-3 shown, both the source and destination happen to indicate in the same cycle that they can transfer the data or control information. In this case the transfer occurs immediately. The arrow shows when the transfer occurs.

3.1.1 Write address channel

The master can assert the AWVALID signal when it drives valid address and control information on write address channel between master to interconnect.

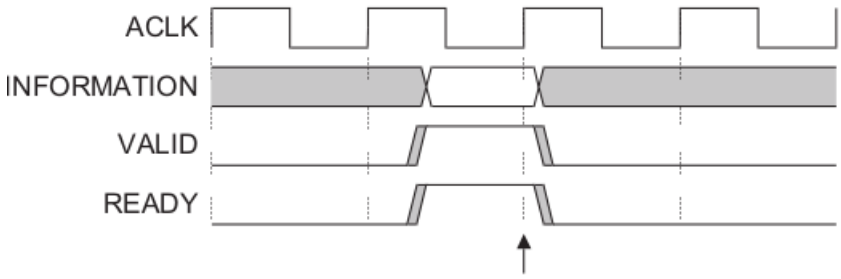

Figure 3.3: VALID with READY handshake

It must remain asserted until the Interconnect accepts the address and control information. When interconnect free, accepts the address and control information by asserting the associated AWREADY signal.

Similarly when interconnect has valid address and control information, assert the AWVALID signal high on write address channel between interconnect to slave. When slave free, accepts the address and control information by asserting the associated AWREADY signal high.

The recommended default value AWREADY and AWVALID signal are HIGH and LOW. A default value of LOW is possible but not recommended for AWREADY signal, because the transfer takes at least two cycles, one to assert AWVALID and another to assert AWREADY.

3.1.2 Write data channel

During a write burst transaction, the master asserts the WVALID signal when it drives valid write data on Write data channel between master to interconnect. When interconnect free, accepts the data information by asserting the associated AWREADY signal.

Similarly the Interconnect asserts the WVALID signal when it drives valid write data on Write data channel between interconnect to slave. When slave free, accepts the address and control information by asserting the associated AWREADY signal high.

The recommended default value WREADY and WVALID signal are HIGH and LOW. When WVALID is LOW, the WSTRB [3:0] signals can take any default value.

3.1.3 Write response channel

The slave asserts the BVALID signal and write response signal only when it receives last data in burst. BVALID must remain asserted until the master accepts the write response and asserts BREADY. The default value of BREADY can be HIGH.

3.1.4 Read address channel

The master asserts the ARVALID signal only when it drives valid address and control information on read address channel between master and interconnect. It must remain asserted until the Interconnect accepts the address and control information. When interconnect free, it asserts the associated ARREADY signal, then address and control information transfer.

Similarly the address and control information transfer from Interconnect to Slave when both AREADY and ARVALID signal are asserted.

The recommended default value ARREADY and ARVALID signal are HIGH and LOW. A default value of LOW is possible but not recommended for ARREADY, because the transfer takes at least two cycles, one to assert AWVALID and another to assert AWREADY.

3.1.5 Read data channel

The slave asserts the RVALID signal only when it drives valid read data on read data channel between slave and interconnect. RVALID must remain asserted until the interconnect accepts the data and asserts the RREADY signal. Even if a slave has only one source of read data, it must assert the RVALID signal only in response to a request for the data.

The interconnect uses the RREADY signal to indicate that it accepts the data. The default value of RREADY can be HIGH. The slave asserts the RLAST signal when it drives the final read transfer in the burst.

3.2 Relationships between the channels

The relationship between the address, read, write, and write response channels is flexible.

For example, the write data can appear at an interface before or after or in the same cycle as write address that relates to it. The data appear before the write address when the write address channel contains more register stages than the write data channel, and after when the when the write data channel contains more register stages than the write address channel .

The interconnect realign the write address and write data, and sends to slave when it is ready. With the help of register slices and ID tag interconnect realigns the write address and write data, we can see this in block diagram on page.

Two relationships that must be maintained are:

- read data must always follow the address to which the data relates.
- a write response must always follow the last write transfer in the write transaction to which the write response relates.

3.3 Dependencies between channel handshake signals

To prevent a deadlock situation, you must observe the dependencies that exist between the handshake signals.

One example of deadlock situation is, during a write transaction, a master must not wait for AWREADY to be asserted before driving WVALID. This could cause a deadlock condition if the slave is waiting for WVALID before asserting AWREADY.

In any transaction:

- the VALID signal of one AXI component must not be dependent on the READY signal of the other component in the transaction
- the READY signal can wait for assertion of the VALID signal.

Figure 3-4 and Figure 3-5 show the handshake signal dependencies of read and write address channel. The single-headed arrows point to signals that can be asserted before or after the previous signal is asserted. Double-headed arrows point to signals that must be asserted only after assertion of the previous signal.

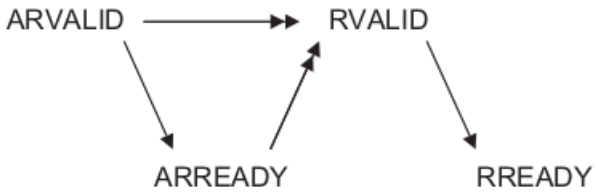

Figure 3.4: Read transaction handshake dependencies

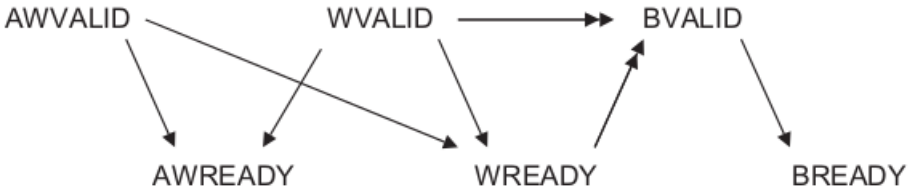

Figure 3.5: Write transaction handshake dependencies

The figure 3-4 shows that, in a read transaction:

- the slave can wait for ARVALID to be asserted before it asserts ARREADY
- the slave must wait for both ARVALID and ARREADY to be asserted before it starts to return read data by asserting RVALID.

The figure 3-5 shows that, in a write transaction:

- the master must not wait for the slave to assert AWREADY or WREADY before asserting AWVALID or WVALID
- the slave can wait for AWVALID or WVALID, or both, before asserting AWREADY
- the slave can wait for AWVALID or WVALID, or both, before asserting WREADY
- the slave must wait for both WVALID and WREADY to be asserted before asserting BVALID.

Chapter 4

Addressing Options

This chapter describes AXI burst types and how to calculate addresses and byte lanes for transfers within a burst.

4.1 About addressing options

The AXI protocol is burst-based, and the master begins each burst by driving transfer control information and the address of the first byte in the transfer. As the burst transaction progresses, it is the responsibility of the slave to calculate the addresses of subsequent transfers in the burst.

4.2 Burst length

Table 4-1 shows how AWLEN or ARLEN signal specifies the number of data transfers that occur within each burst. Every transaction must have the number of transfers specified by ARLEN or AWLEN. No component can terminate a burst early to reduce the number of data transfers. During a write burst, the master can disable further writing by deasserting all the write strobes, but it must complete the remaining transfers in the burst. During a read burst, the master can discard further read data, but it must complete the remaining transfers in the burst.

Table 4.1: Burst length encoding

ARLEN[3:0], AWLEN[3:0]	Number of data transfers
b'0000	1
b'0001	2
b'0010	3
b'0011	4
.....	
b'1101	14
b'1110	15
b'1111	16

Table 4.2: Burst size encoding

ARSIZE[2:0], AWSIZE[2:0]	Bytes in transfer
b'000	1
b'001	2
b'010	4
b'011	8
b'100	16
b'101	32
b'110	64
b'111	128

4.3 Burst size

Table 4-2 shows how the ARSIZE or AWSIZE signal specifies the maximum number of data bytes to transfer in each beat, or data transfer, within a burst. The AXI determines from the transfer address which byte lanes of the data bus to use for each transfer in a read data transaction. The size of any transfer must not exceed the data bus width of the components in the transaction.

4.4 Burst type

- Fixed burst.
- Incrementing Burst.
- Wrapping Burst.

Table 4.3: Burst type encoding

ARBURST[1:0] AWBURST[1:0]	**Burst type**	**Description**	**Access**
b'00	Fixed	Fixed-address burst	FIFO-type
b'01	INCR	Incrementing-address burst	Normal sequential memory
b'10	WRAP	Incrementing-address burst that wraps to a lower address at the wrap boundary	Cache line
b'11	Reserved		

Fixed burst

In a fixed burst, the address remains the same for every transfer in the burst. This burst type is for repeated accesses to the same location such as when loading or emptying a peripheral FIFO.

Incrementing burst

In an incrementing burst, the address for each transfer in the burst is an increment of the previous transfer address. The increment value depends on the size of the transfer. For example, the address for each transfer in a burst with a size of four bytes is the previous address plus four.

Wrapping burst

A wrapping burst is similar to an incrementing burst, in that the address for each transfer in the burst is an increment of the previous transfer address. However, in a wrapping burst the address wraps around to a lower address when a wrap

boundary is reached. The wrap boundary is the size of each transfer in the burst multiplied by the total number of transfers in the burst.

Two restrictions apply to wrapping bursts:

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

4.5 Burst address calculation

Use these equations to determine addresses of transfers within a burst:

- Start Address = ADDR
- Number Bytes = 2SIZE
- Burst Length = LEN + 1
- Aligned Address = (INT(Start Address / Number Bytes)) x Number Bytes

Use this equation to determine the address of any transfer after the first transfer in a burst:

- AddressN = Aligned Address + (N 1) x Number Bytes.

The variables in formulas are

Start Address	The start address issued by the master.
Number Bytes	The maximum number of bytes in each data transfer.
Data Bus Bytes	The number of byte lanes in the data bus.
Aligned Address	The aligned version of the start address.
Burst Length	The total number of data transfers within a burst.
AddressN	The address of transfer N within a burst. N is an integer from 1-16.

Chapter 5

Transaction Ordering Model

This chapter describes how the AXI protocol uses transaction ID tags to enable the issuing of multiple outstanding addresses

5.1 About the Ordering model

The AXI protocol enables out-of-order transaction completion and the issuing of multiple outstanding addresses. These features enable the implementation of a high-performance interconnect, maximizing data throughput and system efficiency.

The ID signals support out-of-order transactions by enabling each port to act as multiple ordered ports. All transactions with a given ID must be ordered, but there is no restriction on the ordering of transactions with different IDs. The five transaction IDs are:

The ability to complete transactions out of order means that transactions to faster memory regions can complete without waiting for earlier transactions to slower memory regions. This feature can also improve system performance because it reduces the effect of transaction latency.

In this project i am doing transactions between four masters and single slave, so i am not implementing out of order feature of AXI protocol.

Table 5.1: Transaction IDs

ID	Description
AWID	The ID tag for the write address group of signals.
WID	The write ID tag for a write transaction. Along with the write data, the master transfers a **WID** to match the **AWID** of the corresponding address.
BID	The ID tag for the write response. The slave transfers a **BID** to match the **AWID** and **WID** of the transaction to which it is responding.
ARID	The ID tag for the read address group of signals.
RID	The read ID tag for a read transaction. The slave transfers an **RID** to match the **ARID** of the transaction to which it is responding.

5.2 Transfer ID fields

The AXI protocol provides an ID field to enable a master to issue a number of separate transactions, each of which must be returned in order.

A master can use the ARID or AWID field of a transaction to provide additional information about the ordering requirements of the master. The rules governing the ordering of transactions are as follows:

- Transactions from different masters have no ordering restrictions. They can complete in any order. Here i am surviving transactions in Round robin fashion.
- Transactions from the same master, but with different ID values, have no ordering restrictions. They can complete in any order. Here i am using different ID values, because of single slave.
- The data for a sequence of write transactions with the same AWID value must complete in the same order that the master issued the addresses in.
- The data for a sequence of read transactions with the same ARID value must be returned in order that:
  - when reads with the same ARID are from the same master then the slave must ensure that the read data returns in the same order that the addresses are received.

  - when reads with the same ARID are from master, the interconnect must ensure that the read data returns in the same order that the master issued the addresses in. This rule is for multi master and multi slave system.

- There are no ordering restrictions between read and write transactions

5.3 Read ordering

At a master interface, read data from read transactions with the same ARID value must arrive in the same order in which the master issued the addresses. Data from read transactions with different ARID values can return in any order and it is also acceptable to interleave the read data of transactions with different ARID fields.

A slave must return read data from a sequence of read transactions with the same ARID value in the same order in which it received the addresses. In a sequence of read transactions with different ARID values, the slave can return the read data in a different order than that in which the transactions arrived.

The slave must ensure that the RID value of any returned read data matches the ARID value of the address to which it is responding.

The interconnect must ensure that a sequence of read transactions with the same ARID value from different slaves complete in order.

5.4 Write ordering

If a slave does not support write data interleaving, the master must issue the data of write transactions in the same order in which it issues the transaction addresses.

Most slave designs do not support write data interleaving and consequently these types of slave design must receive write data in the same order that they receive the addresses. If the interconnect combines write transactions from different masters to one slave, it must ensure that it combines the write data in address order.

Chapter 6

Data Buses

This chapter describes how varying sizes of data transfers on AXI read and write data bus and how slave interface uses byte-invariant endianness to handle mixed-endian transfers.

6.1 About the data buses

The AXI protocol has two independent data buses, one for read data and one for write data. Because these data buses have their own individual handshake signals, it is possible for data transfers to occur on both buses at the same time.

Every transfer generated by a master must be the same width as or narrower than the data bus for the transfer.

6.2 Write and Read strobe

The write and read strobe signals, WSTRB, RSTRB enables align and unaligned data transfer data transfer on the write data bus and read data bus. Each strobe signal corresponds to one byte of the data bus. When asserted, strobe signal indicates that the corresponding byte lane of the data bus contains valid information to be updated in memory or read from memory.

In read transaction of AXI bus, the slave generates the strobe signals from starting address, burst length and burst size information. In appendix i give bluespec logic for strobe signal implementation.

In write transaction of AXI bus, the master interface send the write strobe signal along with the write data to slave interface. I already shown strobe signals in table 3.2.

In figure 6.1, 64 bit data bus is shown and corresponding write strobe signal **WSTRB[n]** is **WDATA[(8 n) + 7: (8 n)]**, and read strobe signal **RSTRB[n]** is **RDATA[(8 n) + 7: (8 n)]**.

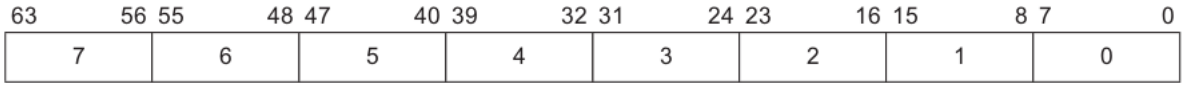

Figure 6.1: Byte lane mapping

6.3 Narrow transfers

When a master generates a transfer that is narrower than its data bus, the address and control information determine which byte lanes the transfer uses. In incrementing or wrapping bursts, different byte lanes transfer the data on each beat of the burst. In a fixed burst, the address and byte lanes remains constant. Narrow data transfer in read data transaction is explained below with example.

Example the master sent following information to slave

- starting address is 0
- burst length or number of transfers is 5
- burst size is 8 bits
- data bus size is 32 bits

The strobe signals generated by slave is shown in table 6.1, and corresponding valid data on data bus is shown in figure 6.2.

Table 6.1: Strobe signals

Transfer	Strobe signals
1st	b'0001
2nd	b'0010
3rd	b'0100
4th	b'1000
5th	b'0001

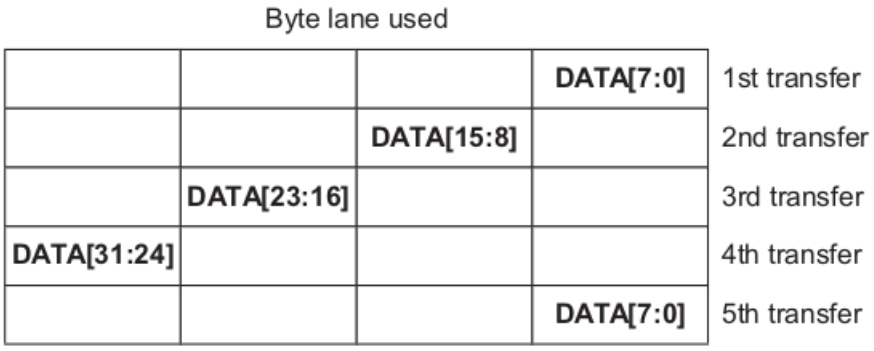

Figure 6.2: Narrow transfer example with 8-bit transfers

Chapter 7

Unaligned Transfers

7.1 About unaligned transfers

The AXI protocol uses burst-based addressing, which means that each transaction consists of a number of data transfers. Typically, each data transfer is aligned to the size of the transfer. For example, a 32-bit wide transfer is usually aligned to four-byte boundaries. However, there are times when it is desirable to begin a burst at an unaligned address.

For any burst that is made up of data transfers wider than one byte, it is possible that the first bytes that have to be accessed do not align with the natural data width boundary. For example, a 32-bit (four-byte) data packet that starts at a byte address of 0x1002 is not aligned to a 32-bit boundary.

The AXI protocol does not require the slave to take special action based on any alignment information from the master. The master can also simply provide an aligned address and, in a write transaction, rely on the byte lane strobes to provide the information about which byte lanes the data is using.

7.2 Examples

Figure 10-1 show examples of aligned and unaligned transfers on buses with different widths. Each row in the figures represents a transfer. The shaded cells indicate bytes that are not transferred, based on the address and control information.

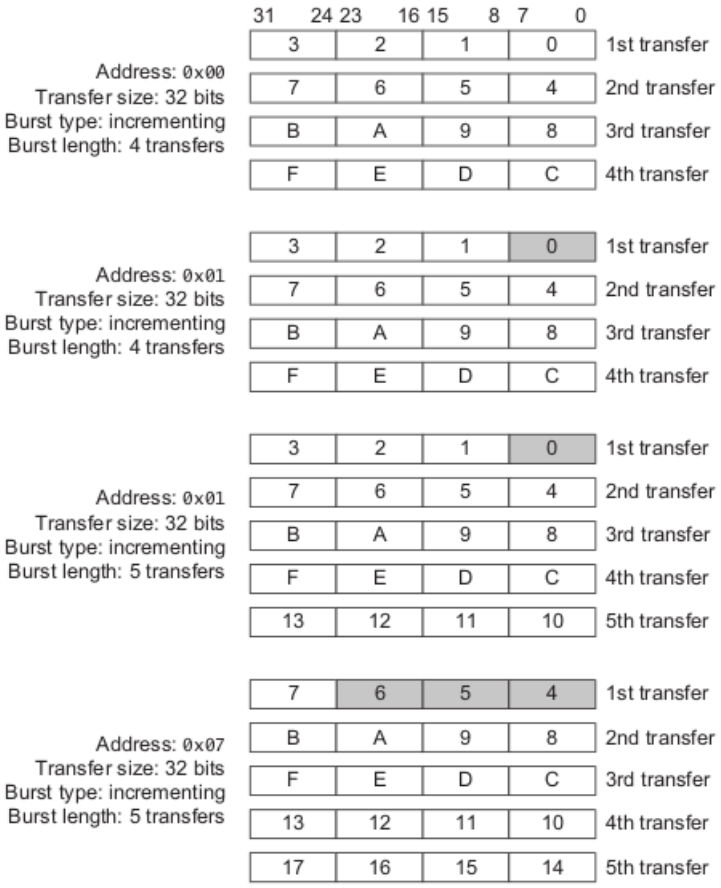

Figure 7.1: Aligned and unaligned word transfers on a 32-bit bus

Chapter 8

Design and Implementation

This chapter describes the design of four masters and single slave AXI4 protocol, how each block works, and how each channel uses the same VALID/READY handshake mechanism to transfer control and data information.

This project is not have all signals present of AXI4 protocol, this is also shown in chapter2 Signal Descriptions. Here i am not implementing the all features of AXI4 protocol. It has the AXI4 protocol signals, which provides burst based transaction, align and unaligned data transactions, and multiple outstanding transactions. It does not have the signal to provide protection and error support.

8.1 Architecture

The block diagram of complete system is shown in fig 8.1, it having four master and single slave connecting through AXI interconnect. Internal diagram of each component is in figure 8.2 to 8.5, each signal description given the table 8.1.

Funtionality of each module explained below

8.1.1 Master interface

AXI Master interface block diagram is shown in figure 8.2. The master is driving information on write address, write data and read address channels, and receiving information on read data and write response channel. The signal description of each channel is given in chapter2.

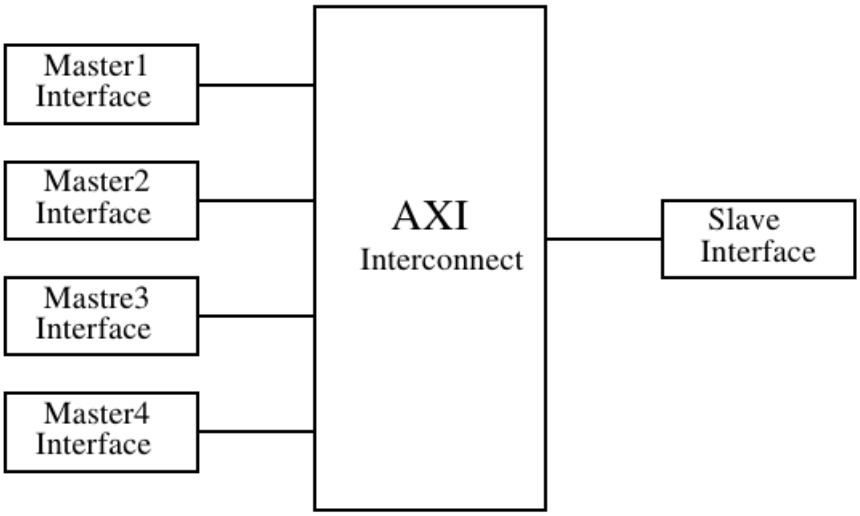

Figure 8.1: Block diagram of multi master single slave AXI protocol

Table 8.1: Signals and Description

Signal	**Description**
WA	It has all Write address channel signals, except READY signal
WD	It has all Write data channel signals, except READY signal
RA	It has all Read address channel , except READY and signal
RD	It has all Read data channel, except READY signal
WR	It has all Write response channel signal, except READY signal
WAREADY,WAVALID	READY and VALID signals on write address channel
RAREADY,RAVALID	READY and VALID signals on read address channel
WDREADY,WDVALID	READY and VALID signals on write data channel
RAREADY,RAVALID	READY and VALID signals on read address channel
NF,NE	NOT EMPTY and NOT FULL signal of FIFO

8.1.2 Interconnect

Interconnect internal block diagram is shown in figure 8.3 and 8.4. The modules and their function is explained below.

FIFO

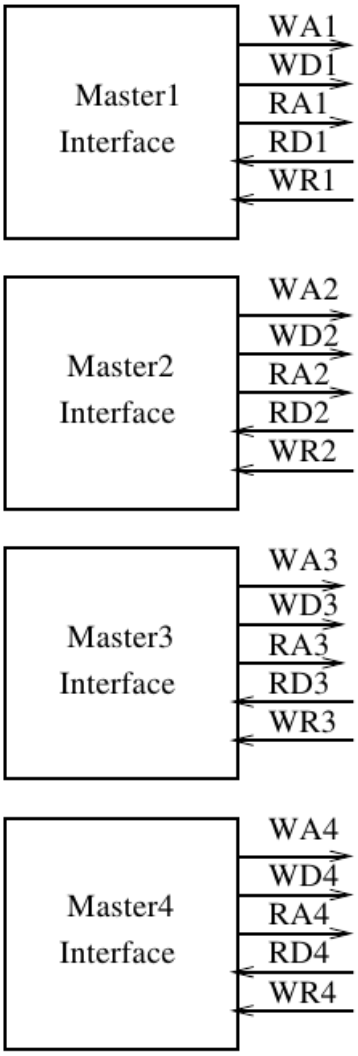

Figure 8.2: Master interface

The block diagram of First-in first-out memory is shown in figure 8.6. It has input and output, and two status signal outputs, which tells whether FIFO is EMPTY or FULL. In figures from 8.2 to 8.5, the number besides FIFO word tells FIFO number and the number in closed bracket tells size of FIFO.

Because of FIFOs in Interconnect

- master interface can send information on write address channel before or after or in the same cycle as of write data channel

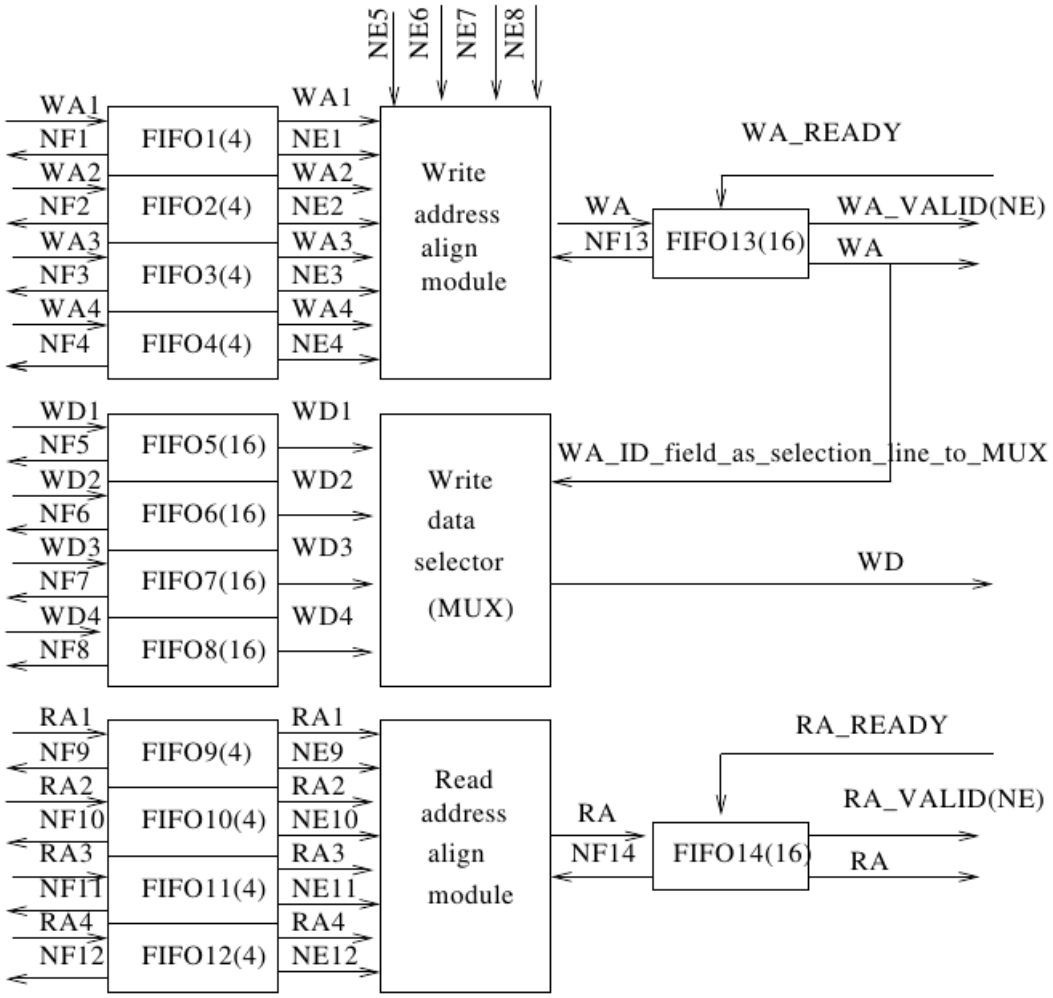

Figure 8.3: AXI Interconnect WA, WD and RA channels

- master can send next write address request, write data and read address before completion of previous request
- slave can send read data before previous data reaching master
- slave can send write response information before previous information reaching master

From above statement we can say that FIFOs are helping us to implement multiple outstanding transaction feature of AXI4 protocol.

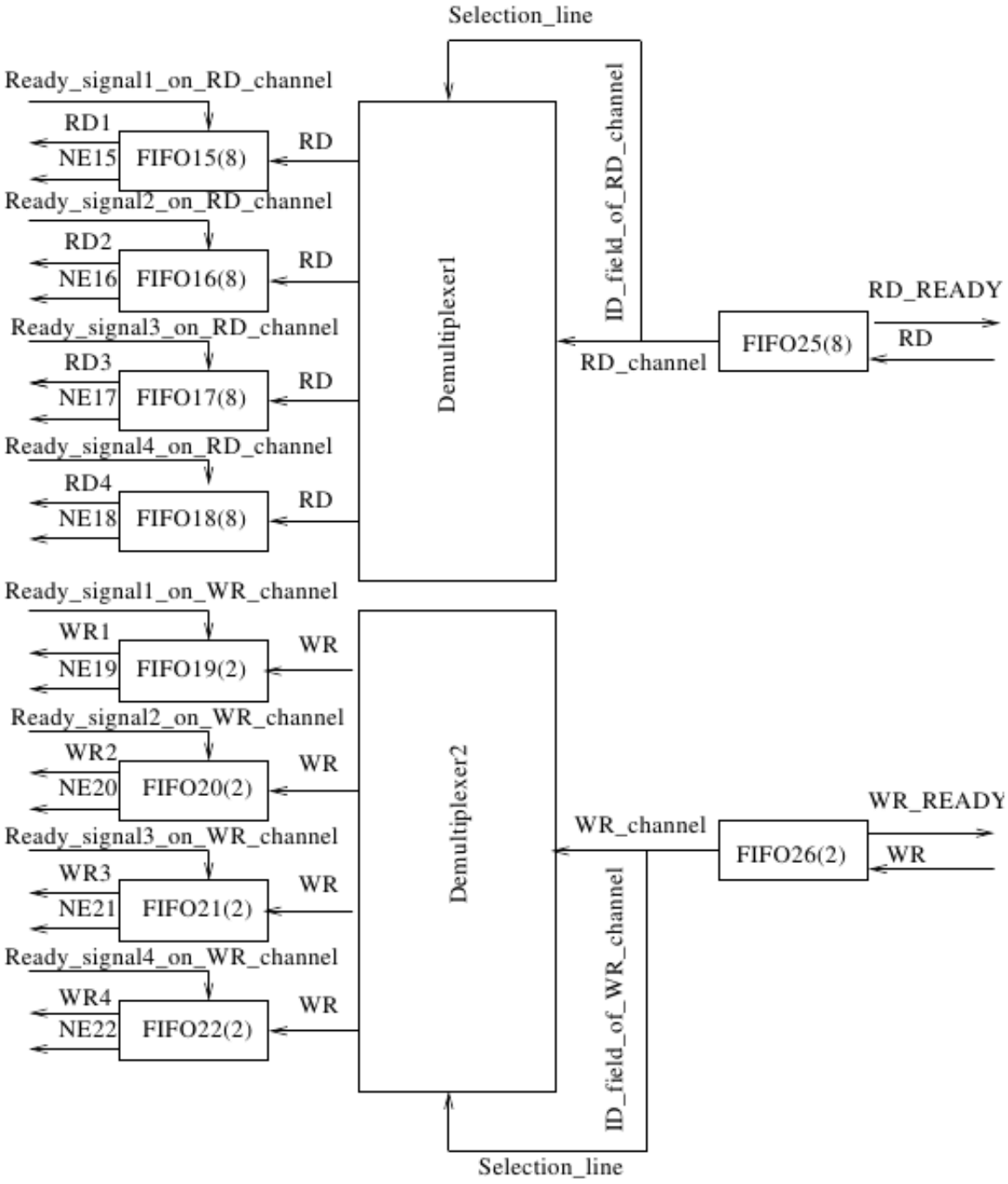

Figure 8.4: AXI Interconnect RD and WS channels

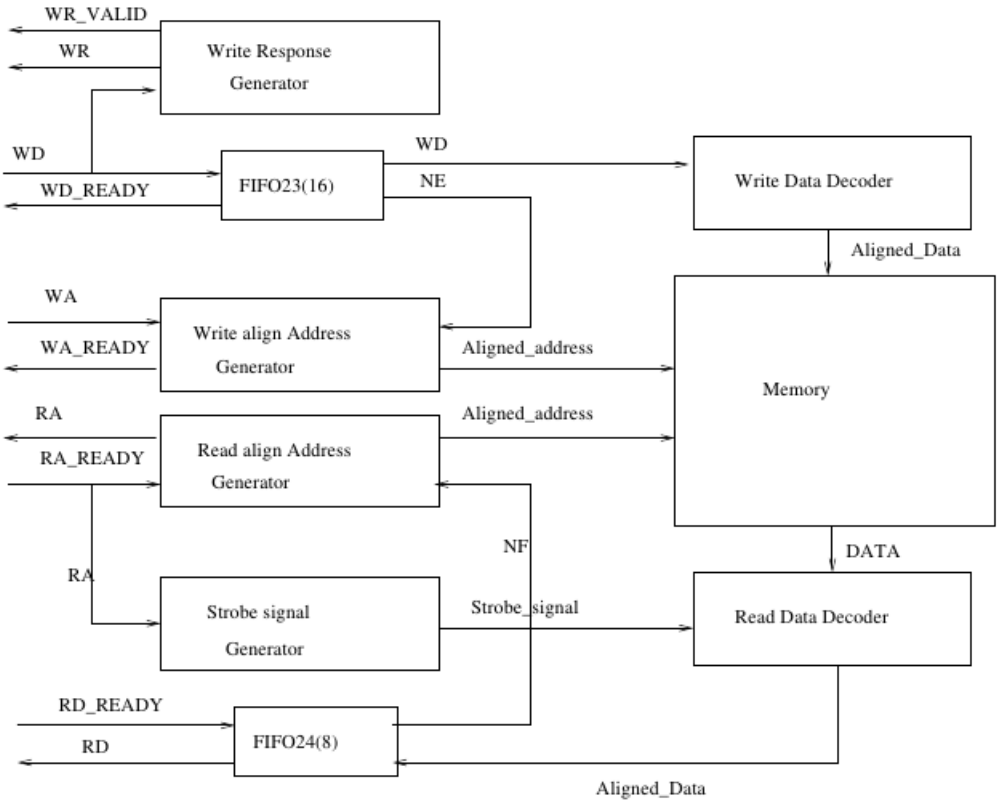

Figure 8.5: Slave interface

Write address align module

It checks the status of write address channel FIFOs and corresponding write data channel FIFOs in every cycle, and sends write address channel requests to FIFO13 in a Round Robin fashion, if both write address channel FIFO and corresponding data channel FIFO are NOT EMPTY. This is explaining with the example in table 8.2. In this table NO means no information is coming.

Write data selector

This module takes ID field on write address channel as input, based on ID field, it read data from corresponding FIFO, and sends to slave interface.

Read address align module

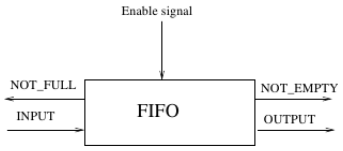

Figure 8.6: Block diagram of FIFO

Table 8.2: Write address align module operation

cycle	**Masters sending write address request**	**Masters sending write data**	**which master request going to FIFO13**
1	master1,master2,master3	No	No
2	No	No	No
3	master4	No	No
4	No	master4	No
5	No	master2,master3	master4
6	No	master1	master2
7	No	No	master3
8	NO	No	master1

This module checks the status of read address channel FIFOs in every cycle, and sends write address channel requests to FIFO13 in a Round Robin fashion. This is explaining with the example in table 8.3. In this table NO means no information is coming.

Demultiplexer1

Based on ID tag field of Read data channel, it sends read data to corresponding master FIFO.

Demultiplexer2

Based on ID tag field of Read data channel, it sends read data to corresponding master FIFO.

Table 8.3: Read address align module operation

cycle	**Masters sending read address request**	**which master request going to FIFO13**
1	Master1,Master2,Master4	No
2	No	Master1
3	master3	Master2
4	No	Master3
5	No	master4

8.1.3 Slave interface

Slave interface internal block diagram is shown in figure 8.5. The modules and their function is explained below.

Write and Read align address generator

This modules take starting address of burst and control information on WA and WR channel, and calculates the next address of each transfer in burst. The equation used is shown in section 4.5. The control information are burst length, burst size and burst type.

The address output of this modules called align address as shown in figure 8.5. This align address goes to the memory, and data read or write into corresponding location based read or write transaction. As shown in figure 8.5 when not full signal of FIFO24 false, it stop working and restore the present values.

Strobe signal generator

This module take starting address of burst and control information, and calculates the stobe signals for each transfer in burst. This signal tell which byte lines of data bus has valid data. About strobe signals explained in section 6.2.

Read data decoder

This module inputs are strobe signals and 32 bit data from memory. Based strobe signals it send valid data as output.

In read transaction, strobe signal generator and read data decoder modules are helping to implement narrow and unaligned data transfer features of AXI4 protocol. With example narrow transfers is explained in section 6.3, and unaligned data transfer is explained in section 7.2.

Write data decoder

This module send valid data to memory based on strobe signal. From section 2.2, we can know that strobe signals and data are available on write data channel, so no need to generate strobe signal at slave interface in write transaction.

Write Response Generator

It sends response signal on write response channel to interconnect, when it receives last data on write data channel.

8.2 Channel Handshake

From chapter 3, we know that each channel uses the same VALID/READY handshake to transfer control and data information. This two way flow control mechanism enables both the master and slave to control the rate at which the data and control information moves. The source generates the VALID signal to indicate when the data or control information is available. The destination generates the READY signal to indicate that it accepts the data or control information. Transfer occurs only when both the VALID and READY signals are HIGH. There must be no combinatorial paths between input and output signals on both master and slave interfaces. In below explaining how this handshake mechanism is implemented on each AXI channel.

8.2.1 Write Address Channel (WA)

The AXI Master interface drives the signal AWID,AWADDR, AWBURST, AWLEN, AWSIZE, with AWVALID as HIGH indicating that the driven signals are valid. It does not drive the AWVALID signal as LOW, until it receives the AWREADY signal, which is driven by the AXI Interconnect, indicating that, it has received the address write command signals. If AWREADY is LOW, then AXI Mas-

ter retains the same values. Figure 3.5 shows the state diagram for write transaction.

Between master interface and interconnect on WA channel, FIFOs 1,2,3,4 drives READY signals to corresponding master and masters drives VALID signals. NF(not full) signal of FIFOs act as READY signal, and default value is TRUE. Once masters has valid information, they asserts VALID signal TRUE, send WA to corresponding FIFO. FIFOs deassert READY signals when get FULL.

Between interconnect and slave interface on WA channel, write align address generator drives READY signal and FIFO13 drives VALID signal. The default value of READY signal from write align address generator is TRUE. NE(not empty) signal of FIFO13 act as VALID signal, and default value is FALSE, once it get WA, it assert VALID signal, send WA to write align address generator, once it receives WA, it deassert the READY signal until present request processing completed.

8.2.2 Write Data Channel (WD)

The AXI Master drives these Write Data signals, after sending the write address command signals. AXI Master drives the WDATA signal with WVALID as HIGH, it holds the same value until it receives the WREADY signal from AXI Interconnect. If WREADY is HIGH, it drives the next WDATA.

Between master interface and interconnect on WD channel, FIFOs 5,6,7,8 drives READY signals to corresponding master and masters drives VALID signals. NF(not full) signal of FIFOs act as READY signal, and default value is TRUE. Once masters has valid information, they asserts VALID signal TRUE, send WA to corresponding FIFO. FIFOs deassert READY signals when get FULL.

Between interconnect and slave interface on WD channel, FIFO23 drives READY signal and write data selector module drives VALID signal. NF(not full) signal of FIFO23 act as READY signal, and default value is TRUE. Once write data selection module gets ID field from output of FIFO13, it assert VALID signal, send WD to FIFO23, once it get full, it deassert the READY signal.

8.2.3 Write Response Channel (WR)

The AXI Slave waits for WLAST signal. After receiving the WLAST signal, it drives the response signals, with BVALID as HIGH. It holds the same value until it receives the BREADY signal from the AXI Interconnect; otherwise it retains the same value.

Between slave interface and interconnect on WR channel, FIFO26 drives READY signal and write response generator drives VALID signal. NF(not full) signal of FIFO26 act as READY signal, and default value is TRUE. Once write response channel receives WLAST signal, asserts VALID signal and drives WR to interconnect.

Between interconnect and master interface on WR channel, FIFOs 19,20,21,22 drives VALID signals to corresponding master and masters drives READY signals. NE(not full) signal of FIFOs act as VALID signal, and default value is FALSE. Once masters are free, they asserts READY signal TRUE, receives WR from FIFO. FIFOs deasserts VALID signal when get EMPTY.

8.2.4 Read Address Channel (RA)

The address read command signals driven by the AXI Master are - ARID, ARADDR, ARBURST, ARLEN, ARSIZE, with ARVALID as HIGH indicating that the driven signals are valid. The AXI Master does not drive the ARVALID signal as LOW, until it receives the ARREADY signal from AXI interconnect. If ARREADY is LOW, then AXI Master retains the same values.

Between master interface and interconnect on RA channel, FIFOs 9,10,11,12 drives READY signals to corresponding master and masters drives VALID signals. NF(not full) signal of FIFOs act as READY signal, and default value is TRUE. Once masters has valid information, they asserts VALID signal TRUE, send RA to corresponding FIFO. FIFOs deasserts READY signal when get FULL.

Between interconnect and slave interface on RA channel, read align address generator drives READY signal and FIFO14 drives VALID signal. The default value of READY signal from read align address generator is TRUE. NE(not empty) signal of FIFO14 act as VALID signal, and default value is FALSE, once it get RA, it assert VALID signal, send WA to write align address generator, once it receives WA , it

deasserts the READY signal untill present request processing completed.

8.2.5 Read Data Channel (RD)

The AXI slave drives the RDATA signal with RVALID as HIGH, after receiving read command. It holds the same value until it receives the RREADY signal from AXI Interconnect. If RREADY is HIGH, it drives the next RDATA.

Between slave interface and interconnect on RD channel, FIFO25 drives READY signal and FIFO24 drives VALID signal. NF(not full) signal of FIFO25 act as READY signal, and default value is TRUE. NE(not empty) signal of FIFO24 act as VALID signal, and default value is FALSE.

Between interconnect and master interface on RD channel, FIFOs 15,16,17,18 drives VALID signals to corresponding master and masters drives READY signals. NE(not full) signal of FIFOs act as VALID signal, and default value is FALSE. Once masters are free, they asserts READY signal TRUE, receives WD from FIFO. FIFOs deasserts VALID signal when get EMPTY.

8.3 Synthesis Evaluation

The design has been synthesized using Xilinx ISE for Virtex 5 xc5vlx110t-1-ff1136.

The synthesis report for write operation is shown below

Number of Slice Registers used : 232

Number of Slice LUTs used : 696

Number of LUT Flip Flop Pairs used : 696

Number of Block RAMs used : 4

Max Frequency of operation : 224 MHz

The synthesis report for read operation is shown below

Number of Slice Registers used : 130

Number of Slice LUTs used : 429

Number of LUT Flip Flop Pairs used : 432

Number of Block RAMs used : 2

Max Frequency of operation : 360 MHz

Chapter 9

CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

AMBA AXI4 is a plug and play IP protocol released by ARM, defines both bus specification and a technology independent methodology for designing, and implementing high-integration embedded interfaces. The data to be read or written to the slave is assumed to be given by the master and is read or written to a particular address location of slave through decoder.

The read transaction takes minimum five cycles when read request is not waiting for previous requests to complete. The write transaction on the other hand also takes minimum five cycles, when write address and write data coming at same cycle and the write request is not waiting for previous request to complete. The design takes one cycle between each transaction for both write as well as read operations. As per synthesis results, the maximum freqency of operation for read operation is 224 MHz, and for write opeartion, it is 360MHz.

9.2 Future scope

The AMBA AXI4 has limitations with respect to the burst data and beats of information to be transferred. The burst must not cross the 4k boundary. Bursts longer than 16 beats are only supported for the INCR burst type. Both WRAP and FIXED burst types remain constrained to a maximum burst length of 16 beats. These are the drawbacks of AMBA AXI4 system which need to be overcome.

Bibliography

[1] ARM, AMBA. "AXI Protocol Specification (Rev 2.0)." Availableat http://www.arm. com (2010).

[2] Bluespec Inc, Bluespec SystemVerilog Reference Guide Revision: 30 January 2012.