VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnanasangama, Macche, Santibastwada Road
Belagavi-590018, Karnataka

A
BE MINI-PROJECT (19EC6DCMPR) REPORT
on

AMBA AXI Implementation on FPGA

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Engineering
in
Electronics & Communications Engineering - ECE
by

1DS19EC003	**Abhishek Singh**
1DS19EC010	**Anish Ingale**
1DS19EC012	**Anmol Singh**
1DS19EC015	**Arpit Kumar**

Under the guidance
of

Dr. Rajagopal A
Associate Professor
ECE Dept., DSCE, Bengaluru

Department of Electronics & Communication Engineering
(An Autonomous College affiliated to VTU Belgaum, accredited by NBA & NAAC, Ranked by NIRF)
Shavige Malleshwara Hills, Kumaraswamy Layout,
Bengaluru-560078, Karnataka, India

2021-22

Certificate

Certified that the mini-project work **(19EC6DCMPR)** entitled "AMBA AXI Implementation on FPGA" carried out by Abhishek Singh(1DS19EC003), Anish Ingale(1DS19EC010), Anmol Singh(1DS19EC012), Arpit Kumar(1DS193C015) are bonafide students of the ECE Dept. of Dayananda Sagar College of Engineering, Bangalore, Karnataka, India in partial fulfillment for the award of Bachelor of Engineering in Electronics & Communication Engineering of the Visvesvaraya Technological University, Belagavi, Karnataka for the VI Semester course during the academic year 2021-22. It is certified that all corrections / suggestions indicated for the mini-project work have been incorporated in the mini-report submitted to the ECE department. This Mini-Project report has been approved as it satisfies the academic requirement in respect of mini-project work prescribed for the said degree.

Mini-Project Guide Sign : _____

Name : **Dr. Rajagopal A**

Mini-Project Section Coordinator Sign : _____

Name : **Prof. Kumar P**

Mini-Project Convener & Chief Coordinator Sign : _____

Name : **Dr. Shashi Raj K.**

Dr. T.C. Manjunath : _____

HOD, ECE, DSCE

Dr. C.P.S. Prakash : _____

Principal, DSCE

External Mini-Project Viva-Voce (SEE)

Name of the mini-project examiners (int & ext) with date :

1 : Signature : _____

2 : Signature : _____

Declaration

Certified that the mini-project work entitled, "AMBA AXI Implementation on FPGA" with the course code **19EC6DCMPR** (3 Credits, 100 Marks, CIE & SEE 50 marks each) is a bonafide work that was carried out by ourselves in partial fulfillment for the award of degree of Bachelor of Engineering in Electronics & Communication Engg. of the Visvesvaraya Technological University, Belagavi, Karnataka during the academic year 2021-22 for the VI Semester Autonomous Course. We, the students of the mini-project group/batch no. A1 do hereby declare that the entire mini-project has been done on our own & we have not copied or duplicated any other's work. The results embedded in this mini-project report has not been submitted elsewhere for the award of any type of degree.

Mr. Abhishek Singh
USN : 1DS19EC003

Sign : _____

Mr. Anish Ingale
USN : 1DS19EC010

Sign : _____

Mr. Anmol Singh
USN : 1DS19EC012

Sign : _____

Mr. Arpit Kumar
USN : 1DS19EC015

Sign : _____

Date : /08/2022

Place : Bengaluru -78

Acknowledgement

We want to thank our esteemed institution Dayananda Sagar College of Engineering, for providing this opportunity to learn and explore the field of VLSI. From the depth of our hearts, we're thankful to our beloved Chairman, Dr Hemachndra Sagar and Vice Chairman, Dr Premachandra Sagar. In addition, we're grateful to the honorable secretary of our esteemed institute Mr Galiswamy and the Joint secretary Tintisha Sagar.

We're eternally grateful to our principal Dr C.P.S Prakash for this opportunity which helped us understand and explore a new field. This project wouldn't be complete without the unwavering support of the Head of Department of the department of Electronics and Communication, Dayananda Sagar College of Engineering, Dr T.C. Manjunath.

We want to thank our guide, pillar of strength, Dr. Rajagopal A, who stayed with us throughout this project. Without his help and guidance, this project wouldn't have taken the shape it has. We would also like to thank Prof. Kumar P. for his permission and guidance through this project.

This project wouldn't be possible without Dr Shashi Raj K, who coordinated this curriculum across 90 odd batches.

On the oust, we'd like to thank our beloved families for their constant support and encouragement. We would also like to thank the Almighty for bestowing his blessings on us and making us capable of achieving the objectives of this project.

Abstract

System on Chip (SoC) is not only an integration of chips, software and different components but also how they interact with each other and how they interconnect for realizing a chipset. And as it has low power consumption, these are significant in embedded systems, VLSI IC realizations etc. Therefore all the subsystems housed on the chip are interfaced by a bus, a single set of wires used to connect multiple subsystems.

The bus, which acts as a means of communication, relies on a Bus Protocol that determines the features like the clock timings, signaling lists, allocating the resources, types of transaction, addressing and arbitration schemes for achieving on-chip communication. There are many bus protocols like AMBA, Avalon Bus, IBM Core Connect, Wishbone, etc, among which AMBA protocols got widely accepted by the SoC designers.

The Advanced Microcontroller Bus Architecture, or AMBA, is an open-standard, on-chip interconnect specification for connecting and managing functional blocks in system-on-a-chip (SoC) designs. So essentially, the AMBA protocol defines how the functional blocks communicate with each other. The recent period of AMBA - AXI, The Advanced Extensible Interface, a point-to-point interconnect offers services for high bandwidth and low latencies, succeeding the constraints of a shared bus protocol in terms of the number of agents. So in this project, we will implement the AMBA-AXI4 Lite protocols on a DE-10 Lite FPGA using Intel Quartus Prime and ModelSim for the Verilog coding.

Keywords : System on Chip(SoC), Advanced Microcontroller Bus Architecture (AMBA), Advanced Extensible Interface (AXI), Field Programmable Gate Array (FPGA).

Table of Contents

List of Figures

Nomenclature and Acronyms

Abbreviations (Alphabetical Order) :

AMBA	Advanced Microcontroller Bus Architecture
AXI	Advanced Extensible Interface
DSCE	Dayananda Sagar College of Engineering
ECE	Electronics & Communication Engineering
FPGA	Field Programmable Gate Array
IEEE	Institute of Electrical & Electronics Engineers

CHAPTER 1
INTRODUCTION

1.1 Overview of the Mini Project

In this project we are going to implement AMBA AXI4-Lite protocols on a FPGA. The main functionalities of AXI4-Lite are that all the transactions being taken place are of burst length 1 and all data accesses use the full width of the data bus. This supports the data bus width of 32-bit or 64-bit, but in this project we are going to implement it on a 32-bit data bus width. AXI4-Lite has five interfacing signals as: - Write address channel, Write data channel, Write response channel, Read address channel, Read data channel. These all will be discussed in depth in the further sections. So the first thing we are going to do is to create all the five signals each for Master and Slave. Then later on we are going to create a top level module to call these all sub modules together and work as whole one system. The tools used in this project are ModelSim and Quartus Prime. Quartus Prime is being used for the final implementation on the FPGA.

1.2 Literature survey

The discourse will be carried out on the critical works and contributions made by various researchers, on employment of on chip bus protocols as an interface for different applications continuing with, projecting the present work i.e. Design and interface of AXI4-Lite Master core with a customized Memory. Survey on AMBA Buses: In System on Chip (SoC) design various components of computer or other electronic sub-systems are mounted onto a single chip. The Notorious Advanced Microcontroller Bus Architecture (AMBA) treated as an active agent in dispensing constructive framework in SoC designs, contributes for the digital glue that binds IP process together adequately. Some of the previous works which are relevant and carry basic information for the present work is depicted[1][2].

Due to the increased customer demands design complexity of system on chip (SOC) increases day by day. Hence there is always a productivity gap and right protocol is chosen for the respective applications. To Improve interconnect performance, Improve Quality of Service, Reduces wiring congestion migration is necessary from AXI4 to AXI4-Lite which allows the processor or masters to access the registers(small and mini peripherals) we prefer AXI4- Lite. And also taking the aid of above references and having an eye, a scenario arises where the Master agent frequently want to access information that might be in small for which it uses a simple registers like cache to interact, an effective and proper bus interface have to be picked for satisfying the need[3]. This had motivated the work "Design and interface of AXI4-Lite Master core with a customized Memory" and been developed in Verilog HDL. AXI4-Lite READ operation takes place from Slave to Master and the WRITE operation takes places from Master to Slave. Current task emphasizing design of interface for AXI4-Lite Master core with a customized Memory, the work is detailed in further sessions [4].

1.3 Objective

This Project aims to implement the **AXI4-Lite protocol** on an FPGA with the help of Quartus Prime Tool.

1.4 Methodology

Verilog HDL was used for the logic coding of the mini-project

The 5 transaction channels were independently created along with different modules for master and slave sides. Next a top level module was used to call upon all the different channels on the master and slave sides in the proper order. For the implementation on FPGA data entry of 32 bits was shortened to 4 switches due to limitations of the FPGA. The address channel

address was pre entered. We showed the capability of the project by writing data to a slave and then fetching the same data back

1.5 Organization of Mini-project report

The master and slave side implementations of each channel are created in separate modules. The write transaction channels write address, write data and write response are connected to each other and follow proper chronological order for a transaction which is: address, data, response. Similarly for the read transaction the order is: read address, data. All modules get called by a top level module which takes care of the order and chronology of the functioning of the channels

CHAPTER 2
Block Diagram and Specifications

2.1 Block Diagram

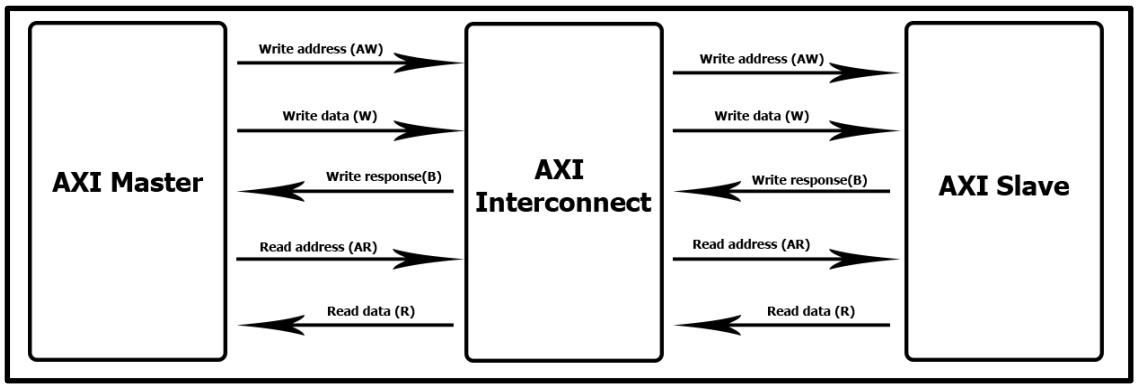

Fig 1: Main Block Diagram

Global	ACLK	ARESETn	—	—
Write address channel	AWVALID	AWREADY	AWADDR	AWPROT
Write data channel	WVALID	WREADY	DATA	WSTRB
Write response channel	BVALID	BREADY	BRESP	—
Read address channel	ARVALID	ARREADY	ARADDR	ARPROT
Read data channel	RVALID	RREADY	RDATA	RRESP

Fig 2: List of Signals

As we can see that there are about five transaction channels which are the means of communication between Master and Slave as shown in the Fig1. There is one AXI Master and the other is the AXI Slave, in between there an AXI Interconnect. The Five transaction channels are as follows:

1. *Write Address Channel*
2. *Write Data Channel*
3. *Write Response Channel*
4. *Read Address Channel*
5. *Read Data Channel*

2.2 AXI4-Lite Specifications

1. **Bus Width:**

   AXI4-Lite has a fixed bus width and all transactions are the same width as the data bus. The data bus width must be either 32-bits or 64-bits. And as mentioned earlier, in this project we are going to implement for 32-bits.

2. **Write Strobes:**

   The AXI4-Lite protocol supports write strobes. This means multi-sized registers can be implemented and also supports memory structures that require support for 8-bit and 16-bit accesses.

2.3 Transaction Channels

1. **Write Address Channel:**

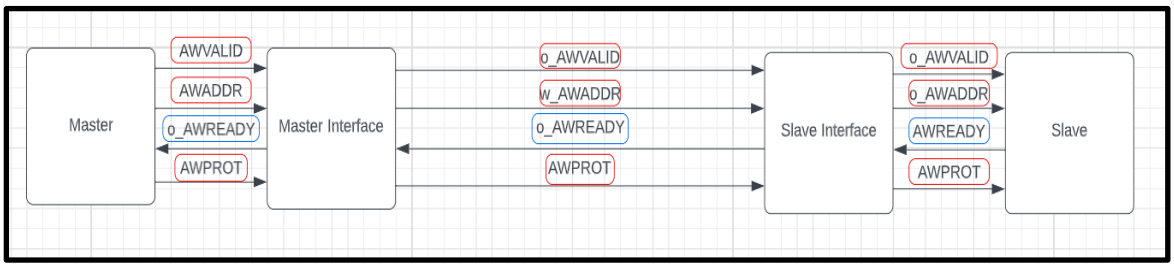

Fig 3: Write Address Channel

Write Address Channel has four signals as shown in the above Fig3 and is explained as follows:

- AWVALID: Write address valid. Master generates this signal when Write Address and control signals are valid Direction is from master to slave.
- AWREADY: Write address ready. Slave generates this signal when it can accept Write Address and control signals Direction is from Slave to master.

- ❖ AWADDR: Write address, usually 32-bits wide. Direction is from master to slave.
- ❖ AWPROT: Protection type.3 bit signal which defines the protection type Direction is from master to slave.

2. Write Data Channel:

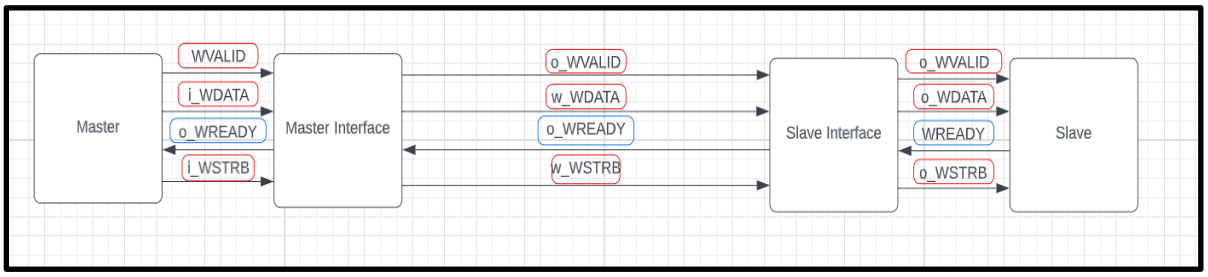

Fig 4 Write Data Channel

Write Data Channel has four signals as shown in the above Fig4 and is explained as follows:

- ❖ WVALID: *Write data valid*. Master generates this signal when Write data and control signals are valid . Direction is from master to slave.
- ❖ WREADY: *Write data ready*. Slave generates this signal when it can accept Write data and control signals Direction is from Slave to master.
- ❖ WDATA: *Write data*, usually 32-bits wide. Direction is from master to slave.
- ❖ WSTRB: *Write strobes*. 4-bit signal indicating which of the 4-bytes of Write Data. Slaves can choose assume all bytes are valid Direction is from master to slave

3. Write Response Channel:

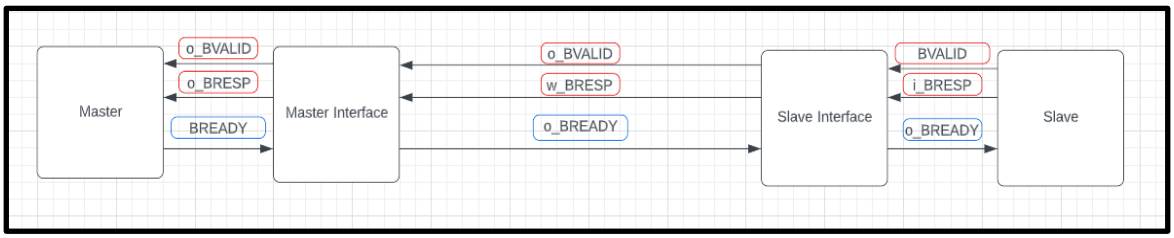

Fig. 5: Write Response Channel

Write response channel has three signals as shown in the above Fig5.

- ❖ BVALID: Write response. This signal indicates the status of the write transaction. Direction is from slave to master.
- ❖ BREADY: Write response valid. Slave generates this signal when the write response on the bus is valid. Direction is from master to slave.
- ❖ BRESP: Write response. This signal indicates the status of the write transaction. Direction is from slave to master.

4. Read Address Channel:

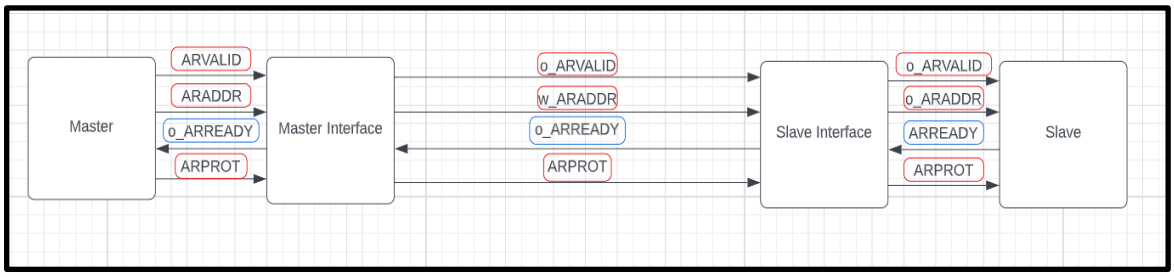

Fig 6: Read Address Channel

Read Address channel has 4 signals as shown in the above Fig6

- ARVALID: Read address valid. Master generates this signal when Read Address and the control signals are valid. Direction is from master to slave.
- READY: Read address ready. Slave generates this signal when it can accept the read address and control signals. Direction is from Slave to master.
- ARADDR: Read address, usually 32-bit wide. Direction is from master to slave.
- ARPROT: A 3 bit signal used to define the protection type Direction is from master to slave.

5. Read Data Channel:

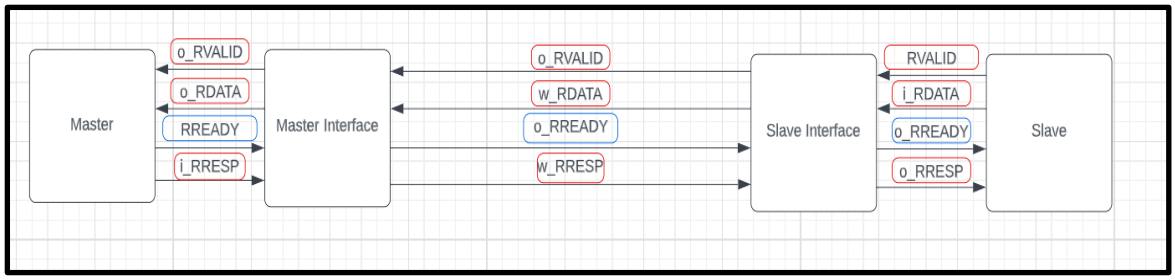

Fig 7: Read Data Channel

Read Data channel has 4 signals as shown in the above Fig7.

- RVALID: Read valid. Slave generates this signal when Read Data is valid Direction is from slave to master.
- RREADY: Read ready. Master generates this signal when it can accept the Read Data and response. Direction is from master to slave.
- RDATA: Read Data (32-bit only). Direction is from slave to master.
- RRESP: Read response. This signal indicates the status of data transfer. Direction is from slave to master.

CHAPTER 3
WORKING

- The Basic function of every channel is handshaking. Every channel has a READY and VALID signal for this operation. It is the first thing the channel will do before the transaction takes place. The sender will send a VALID signal, and the receiver will send a READY signal to indicate their readiness to send and receive data. When both of these signals are high, the transaction takes place.
- Once the connection is established after the handshake, the channels will send their respective information from either master to slave or slave to master (This direction depends on the channel).
- Transactions can broadly be classified into two categories: Read and write transactions. Read transactions are used to read data from the slave, and write transactions are used to write data to the slave.
- For the write operation, the master sends the address of the slave it wants to write in (write address channel) and the data that needs to be written (write data channel). Once the data has been received, the slave sends a response (write response channel).
- For read operation the same data that was written in the write operation is read from the slave (read address channel). In this case the master sends the address from which it wants to read data. The data is provided by the slave to the master(read data channel).

3.1 Handshaking:

All five transaction channels use the same VALID/READY handshake process to transfer address, data, and control information. This two-way flow control mechanism means both the master and slave can control the rate at which the information moves between master and slave. The source generates the VALID signal to indicate when the address, data or control information is available. The destination generates the READY signal to indicate that it can accept the information. Transfer occurs only when both the VALID and READY signals are HIGH. On master and slave interfaces there must be no combinatorial paths between input and output signals.

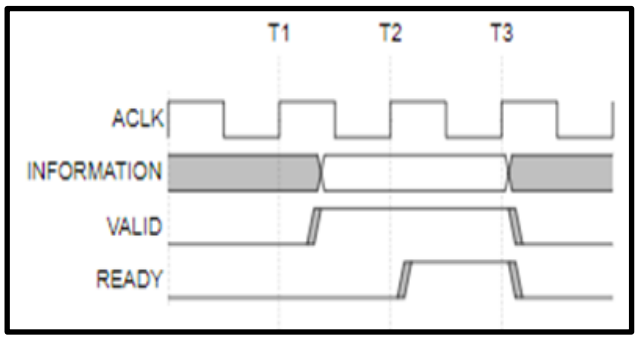

Fig 8: Ready after Valid

In Figure 8, the source presents the address, data or control information after T1 and asserts the VALID signal. The destination asserts the READY signal after T2, and the source must keep its information stable until the transfer occurs at T3, when this assertion is recognized.

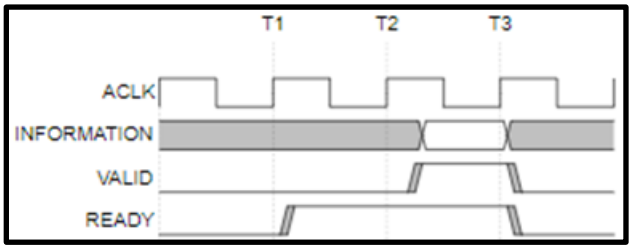

Fig 9: Ready Before Valid

A source is not permitted to wait until READY is asserted before asserting VALID. Once VALID is asserted it must remain asserted until the handshake occurs, at a rising clock edge at which VALID and READY are both asserted. In Figure 9, the destination asserts READY, after T1, before the address, data or control information is valid, indicating that it can accept the information. The source presents the information, and asserts VALID, after T2, and the transfer occurs at T3, when this assertion is recognized. In this case, transfer occurs in a single cycle.

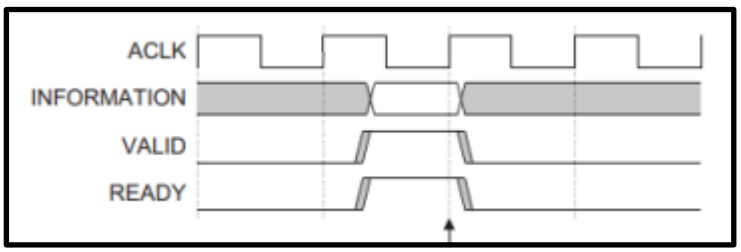

Fig 10: Valid and Ready Together

A destination is permitted to wait for VALID to be asserted before asserting the corresponding READY. If READY is asserted, it is permitted to deassert READY before VALID is asserted. In Figure 10, both the source and destination happen to indicate, after T1, that they can transfer the address, data or control information. In this case the transfer occurs at the rising clock edge when the assertion of both VALID and READY can be recognized. This means the transfer occurs at T2.

3.2 Read Operation:

Below, the sequence for an AXI4-Lite read is shown in the Fig11:

1. The Master puts an address on the Read Address channel as well as asserting ARVALID,indicating the address is valid, and RREADY, indicating the master is ready to receive data from the slave.
2. The Slave asserts ARREADY, indicating that it is ready to receive the address on the bus.
3. Since both ARVALID and ARREADY are asserted, on the next rising clock edge the handshake occurs, after this the master and slave deassert ARVALID and the ARREADY, respectively. (At this point, the slave has received the requested address).
4. The Slave puts the requested data on the Read Data channel and asserts RVALID, indicating the data in the channel is valid. The slave can also put a response on RRESP, though this does not occur here.
5. Since both RREADY and RVALID are asserted, the next rising clock edge completes the transaction. RREADY and RVALID can now be deasserted.

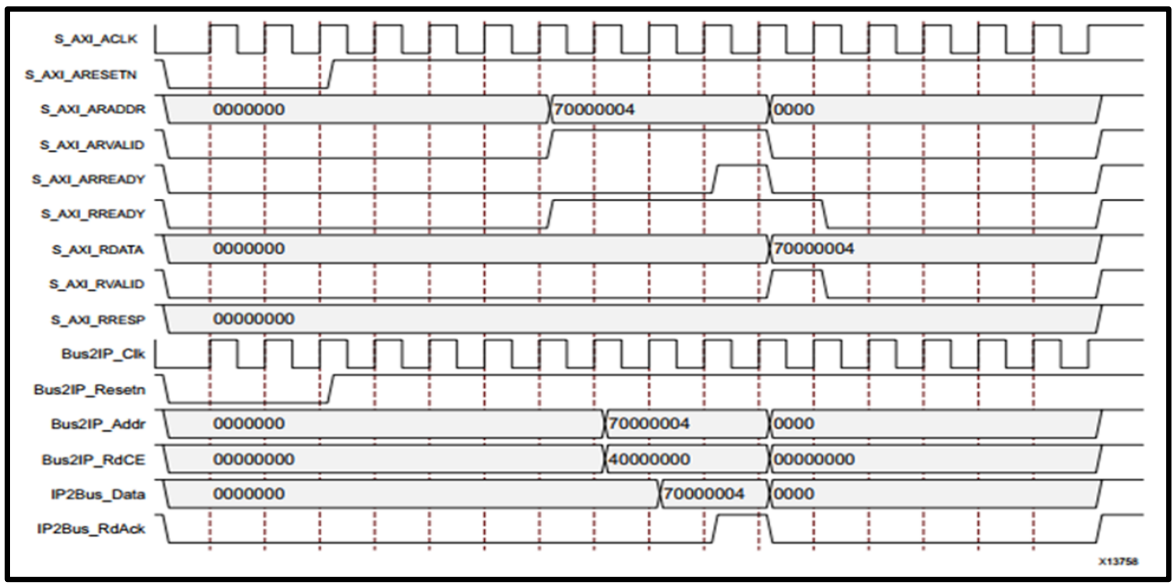

Fig 11: Read Operation

3.3 Write Operation:

Below, the sequence for an AXI4-Lite write is shown Fig12:

1. The Master puts an address on the Write Address channel and data on the Write data channel. At the same time it asserts AWVALID and WVALID indicating the address and data on the respective channels is valid. BREADY is also asserted by the Master, indicating it is ready to receive a response.
2. The Slave asserts AWREADY and WREADY on the Write Address and Write Data channels, respectively.
3. Since Valid and Ready signals are present on both the Write Address and Write Data channels, the handshakes on those channels occur and the associated Valid and Ready signals can be deasserted. (After both handshakes occur, the slave has the write address and data)
4. The Slave asserts BVALID, indicating there is a valid reponse on the Write response channel. (in this case the response is 2'b00, that being 'OKAY').
5. The next rising clock edge completes the transaction, with both the Ready and Valid signals on the write response channel high.

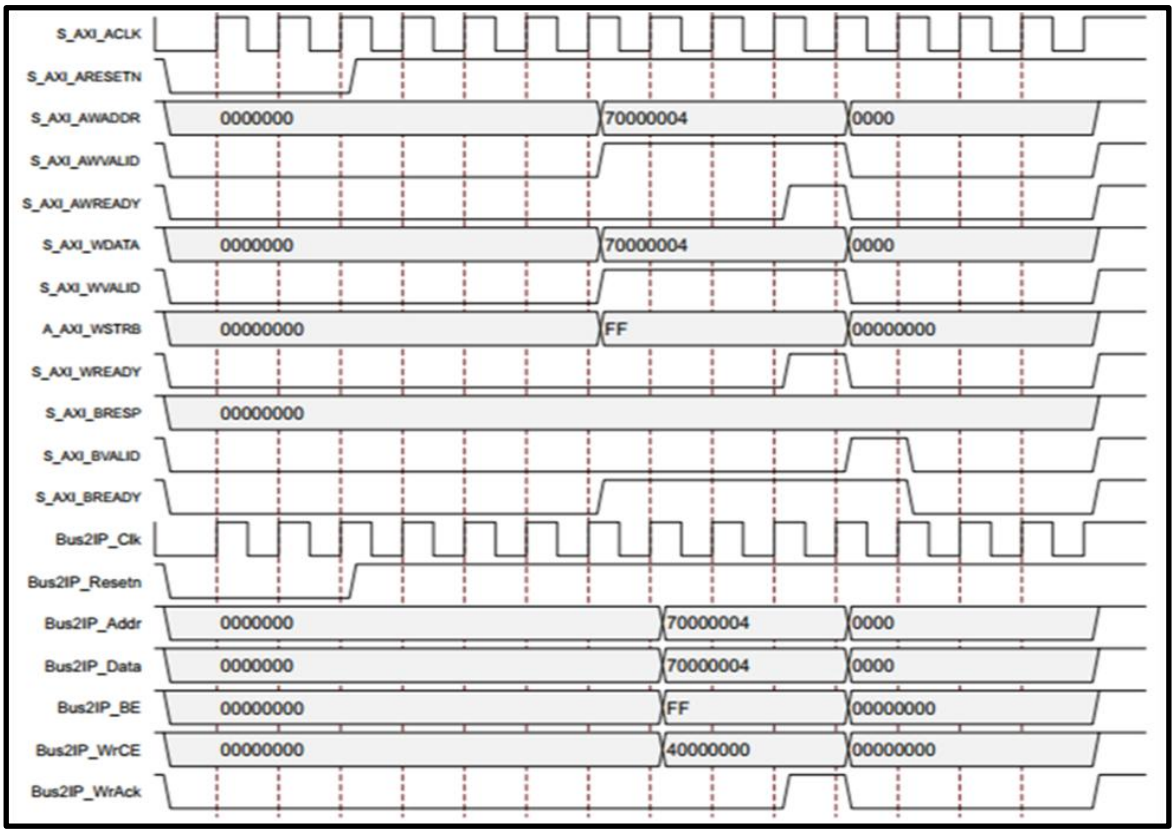

Fig 12: Write Operation

3.4 Flowchart:

3.5 Algorithm:

1. Designing individual channels and channel interfaces.
2. Each channel consists of unique features specified in the specific sheet.
3. Channels can be categorized into two parts: read channels and write channels.
4. The two read channels are responsible for transferring of read address and read data.
5. The three write channels are responsible for the transfer of write address write data and write response.
6. We are combining all these channels into a top-level module to interface dummy master and dummy slave.
7. The operation between these components can be seen on a FPGA.

CHAPTER 4

RESULTS

4.1 Results:

In this project we have

- Designed and verified the functionality of all channels in AXI4-lite.
- Implemented slave and master interface for all channels.
- Successfully realized various features of AXI4-lite like handshaking and write strobe.

4.2 Simulation Results:

- **ModelSim Output:**

The ModelSim output is shown in the below Fig13.

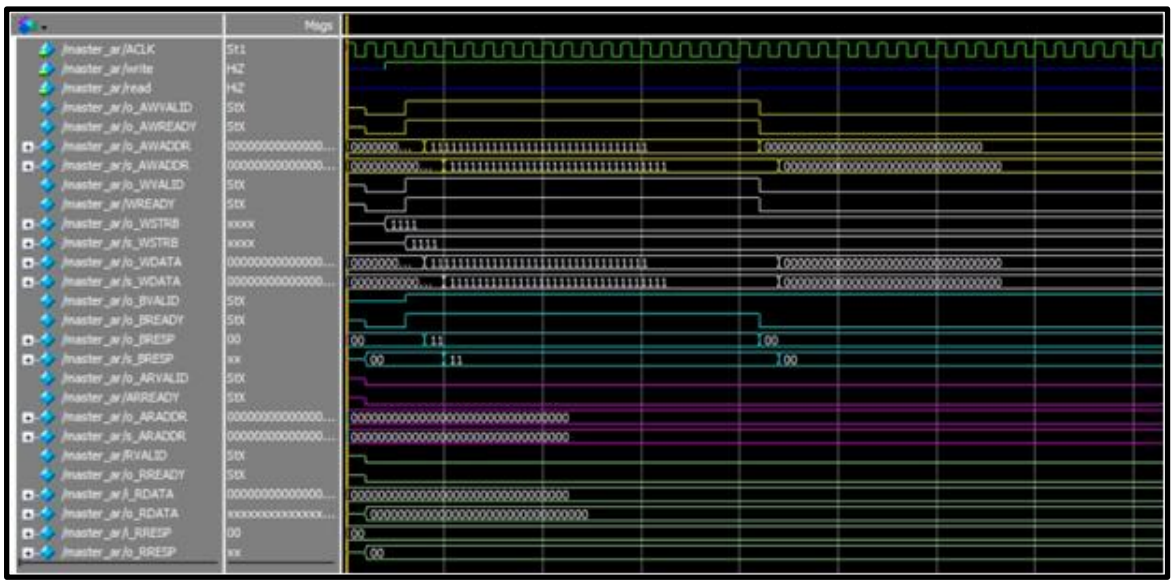

Fig 13: ModelSim Output

❖ **Output on DE-10 Lite (FPGA):**

The output on DE-10 Lite(FPGA) is given in the below Fig14.

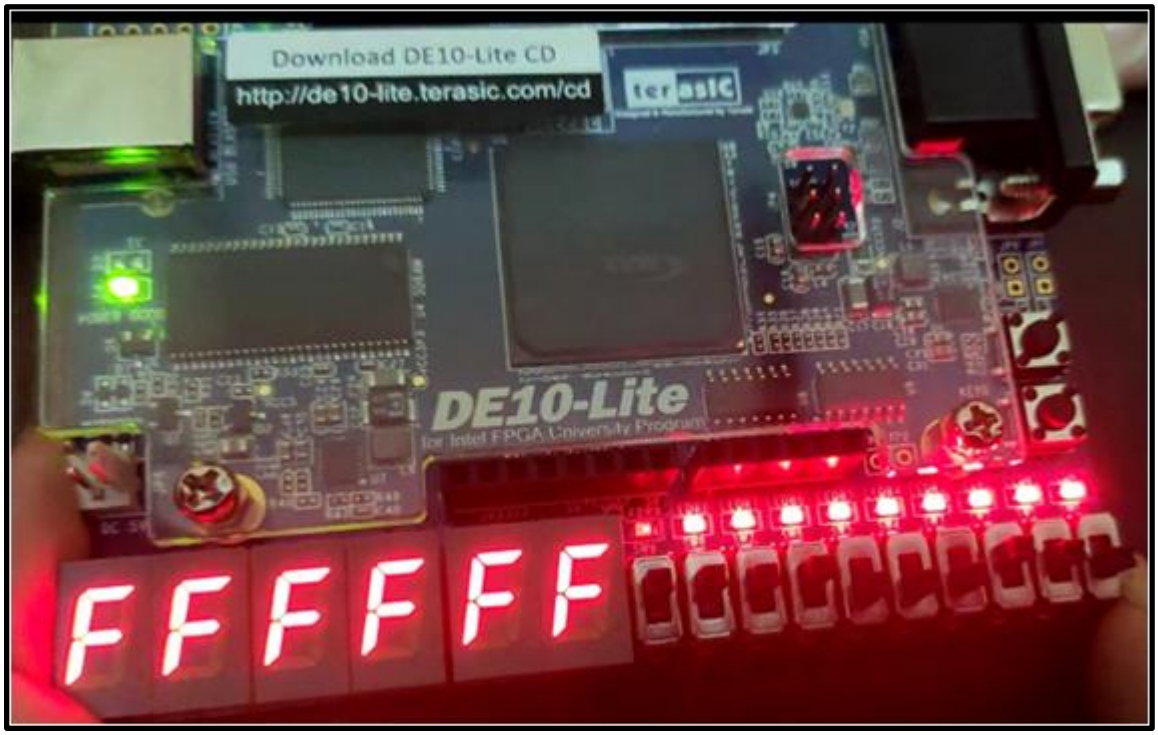

Fig 14: FPGA Output

CHAPTER 5

APPLICATIONS AND LIMITATIONS

5.1 Applications:

- Widely used on ASIC and SOC parts.
- Used in smartphones, IoT subsystems, etc
- It enables efficient IP reuse
- It allows compatibility between IP components from different design teams or vendors.
- It is widely implemented and supported throughout the semiconductor industry.

5.2 Advantages:

- Protocol of the day- the AMBA AXI protocol is highly in demand in the industry today. It's wide applications make it a tool used in many applications and ventures
- Better - The protocol is better than it's competition like wishbone which is much slower protocol and has same channel for address and data.
- Advanced- features like protection levels and strobe functionality make the protocol an advanced and up to today's standards protocol.
- Higher speed - it provides more speed than all it's competition and is hence widely used.

5.3 Outcomes:

- Reusable: The IP Core can be used as a tool for other projects/products/applications to build upon
- Customizable : The Design is totally customisable for the needs of different users and applications
- One size fits all protocol, finds usage in numerous applications and industries

5.4 Limitations:

- The AXI-4 protocol has many other added features such as support for burst mode and IDs for slaves and masters, it also has capability to support more extensive architecture and applications
- Small uses- projects which do not require high data transfer rate or multiple slaves at once support may use a simpler protocol like SPI for communication
- Due it's parallel data transfer nature and no noise handling capabilities it's not designed to be used as a protocol for communication at long distances

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 CONCLUSIONS:

- AXI4-lite and the AXI protocol, in general, is an in-demand skill for the industry. It is widely used for communication between different peripherals in many applications such as mobile phones, laptops and other electronic gadgets.
- Developing this protocol from scratch using Verilog has given us a clear understanding of its architecture and functioning
- By the end of the project we were able to understand Digital Design and also acquire knowledge about Verilog.
- We also had a very good experience during this project, as we had a hands on the FPGA and this will surely help us in further upcoming projects.

6.2 FUTURE WORK:

- We plan to merge projects by other batches such as FIFO, SRAM by connecting them to our protocol as applications of our project.
- We plan to extend this protocol to add the functionality of multiple masters and multiple slaves
- We plan to connect APB, AHB protocols to our project.
- We plan to do the synthesis and most optimized layout design on the FPGA

REFERENCES

1. Sainath Chaithanya, A., Sulthana, Sameera, Yamuna, B. and Haritha, Ch (2020).' Design of AMBA AXI4-Lite for Effective Read/Write Transactions with a Customized Memory'. International Journal on Emerging Technologies

2. Xiao, Fu-ming & Li, Dong-sheng & Du, Gao-ming & Song, Yu-kun & Zhang, Duo-li & Gao, Ming-lun. (2009). Design of AXI bus based MPSoC on FPGA. 560 - 564. 10.1109/ICASID.2009.5277006.

3. N. Gaikwad and V. N. Patil, "Verification of AMBA AXI On-Chip Communication Protocol," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-5, doi: 10.1109/ICCUBEA.2018.8697587.

4. S.P, Guruprasad. (2011). Design and Analysis of Master module for AMBA AXI-4.

5. M. Makni, M. Baklouti, S. Niar and M. Abid, "Performance Exploration of AMBA AXI4 Bus Protocols for Wireless Sensor Networks," 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), 2017, pp. 1163-1169, doi: 10.1109/AICCSA.2017.26.

6. Shaila S Math, Manjula R B, "Survey of system on chip buses based on industry standards", Conference on Evolutionary Trends in Information Technology(CETIT), Belgaum,Karnataka, India, pp. 52, May 2011

7. Amba axi4 Lite protocol spec sheet

8. www.developer.arm.com

9. www.allaboutcircuits.com

10. www.arm.com

Awards and Recognition

★ **Participated in Synergy 2022(an IEEE event) at AIT Bangalore**

★ **Participated in Silicons Project Expo 2022 at RNSIT Bangalore**

Appendix

B1.1 Definition of AXI4-Lite

This section defines the functionality and signal requirements of AXI4-Lite components.

The key functionality of AXI4-Lite operation is:

- all transactions are of burst length 1
- all data accesses use the full width of the data bus
  - AXI4-Lite supports a data bus width of 32-bit or 64-bit.
- all accesses are Non-modifiable, Non-bufferable
- Exclusive accesses are not supported.

B1.1.1 Signal list

Table B1-1 shows the required signals on an AXI4-Lite interface.

Table B1-1 AXI4-Lite interface signals

Global	Write address channel	Write data channel	Write response channel	Read address channel	Read data channel
ACLK	AWVALID	WVALID	BVALID	ARVALID	RVALID
ARESETn	AWREADY	WREADY	BREADY	ARREADY	RREADY
–	AWADDR	WDATA	BRESP	ARADDR	RDATA
–	AWPROT	WSTRB	–	ARPROT	RRESP

AXI4 signals modified in AXI4-Lite

The AXI4-Lite interface does not fully support the following signals:

RRESP, BRESP

The EXOKAY response is not supported on the read data and write response channels.

AXI4 signals not supported in AXI4-Lite

The AXI4-Lite interface does not support the following signals:

AWLEN, ARLEN The burst length is defined to be 1, equivalent to an **AxLEN** value of zero.

AWSIZE, ARSIZE All accesses are defined to be the width of the data bus.

Note

AXI4-Lite requires a fixed data bus width of either 32-bits or 64-bits.

AWBURST, ARBURST

The burst type has no meaning because the burst length is 1.

AWLOCK, ARLOCK

All accesses are defined as Normal accesses, equivalent to an **AxLOCK** value of zero.

AWCACHE, ARCACHE

All accesses are defined as Non-modifiable, Non-bufferable, equivalent to an **AxCACHE** value of `0b0000`.

B1.2 Interoperability

This section describes the interoperability of AXI and AXI4-Lite masters and slaves. Table B1-2 shows the possible combinations of interface, and indicates that the only case requiring special consideration is an AXI master connecting to an AXI4-Lite slave.

Table B1-2 Full AXI and AXI4-Lite interoperability

Master	Slave	Interoperability
AXI	AXI	Fully operational.
AXI	AXI4-Lite	AXI ID reflection is required. Conversion might be required.
AXI4-Lite	AXI	Fully operational.
AXI4-Lite	AXI4-Lite	Fully operational.

B1.2.1 Bridge requirements of AXI4-Lite slaves

As Table B1-2 shows, the only interoperability case that requires special consideration is the connection of an AXI4-Lite slave interface to a full AXI master interface.

This connection requires AXI ID reflection. The AXI4-Lite slave must return the AXI ID associated with the address of a transaction with the read data or write response for that transaction. This is required because the master requires the returning ID to correctly identify the transaction response.

If an implementation cannot ensure that the AXI master interface only generates transactions in the AXI4-Lite subset, then some form of adaptation is required. See *Conversion, protection, and detection* on page B1-127.

B1.2.2 Direct connection requirements of AXI4-Lite slaves

An AXI4-Lite slave can be designed to include ID reflection logic. This means the slave can be used directly on a full AXI connection, without a bridge function, in a system that guarantees that the slave is accessed only by transactions that comply with the AXI4-Lite subset.

Note

This specification recommends that the ID reflection logic uses **AWID**, instead of **WID**, to ensure compatibility with both AXI3 and AXI4.

B1.3 Defined conversion mechanism

This section defines the requirements to convert any legal AXI transaction for use on an AXI4-Lite component. *Conversion, protection, and detection* on page B1-127 discusses the advantages and disadvantages of the various approaches that can be used.

B1.3.1 Conversion rules

Conversion requires that the AXI data width is equal to or greater than the AXI4-Lite data width. If this is not the case then the AXI data width must first be converted to the AXI4-Lite data width.

Note

AXI4-Lite does not support EXOKAY responses, so the conversion rules do not consider this response.

The rules for conversion from a full AXI interface are as follows:

- If a transaction has a burst length greater than 1 then the burst is broken into multiple transactions of burst length 1. The number of transactions that are created depends on the burst length of the original transaction.
- When generating the address for subsequent beats of a burst, the conversion of bursts with a length greater than 1 must take into consideration the burst type. An unaligned start address must be incremented and aligned for subsequent beats of an INCR or WRAP burst. For a FIXED burst the same address is used for all beats.
- Where a write burst with length greater than 1 is converted into multiple write transactions, the component responsible for the conversion must combine the responses for all of the generated transactions, to produce a single response for the original burst. Any error response is sticky. That is, an error response received for any of the generated transactions is retained, and the single combined response indicates an error. If both a SLVERR and a DECERR are received then the first response received is the one that is used for the combined response.
- A transaction that is wider than the destination AXI4-Lite interface is broken into multiple transactions of the same width as the AXI4-Lite interface. For transactions with an unaligned start address, the breaking up of the burst occurs on boundaries that are aligned to the width of the AXI4-Lite interface.
- Where a wide transaction is converted to multiple narrower transactions, the component responsible for the conversion must combine the responses to all of the narrower transactions, to produce a single response for the original transaction. Any error response is sticky. If both a SLVERR and a DECERR are received then the first response received is used for the combined response.
- Transactions that are narrower than the AXI4-Lite interface are passed directly and are not converted.
- Write strobes are passed directly, unmodified.
- Write transactions with no strobes are passed directly.

  Note

  The AXI4-Lite protocol does not require these transactions to be suppressed.

- The **AxLOCK** signals are discarded for all transactions. For a sequence of locked transactions any lock guarantee is lost. However, the locked nature of the transaction is lost only at any downstream arbitration. For an exclusive sequence, the AXI signaling requirements mean any exclusive write access must fail.
- The **AxCACHE** signals are discarded. All transactions are treated as Non-modifiable and Non-bufferable.

  Note

  This is acceptable because AXI permits Modifiable accesses to be treated as Non-modifiable, and Bufferable accesses to be treated as Non-bufferable.

- The **AxPROT** signals are passed directly, unmodified.

A2.2 Write address channel signals

Table A2-2 shows the AXI write address channel signals. Unless the description indicates otherwise, a signal is used by AXI3 and AXI4.

Table A2-2 Write address channel signals

Signal	Source	Description
AWID	Master	Write address ID. This signal is the identification tag for the write address group of signals. See *Transaction ID* on page A5-77.
AWADDR	Master	Write address. The write address gives the address of the first transfer in a write burst transaction. See *Address structure* on page A3-44.
AWLEN	Master	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. This changes between AXI3 and AXI4. See *Burst length* on page A3-44.
AWSIZE	Master	Burst size. This signal indicates the size of each transfer in the burst. See *Burst size* on page A3-45.
AWBURST	Master	Burst type. The burst type and the size information, determine how the address for each transfer within the burst is calculated. See *Burst type* on page A3-45.
AWLOCK	Master	Lock type. Provides additional information about the atomic characteristics of the transfer. This changes between AXI3 and AXI4. See *Locked accesses* on page A7-95.
AWCACHE	Master	Memory type. This signal indicates how transactions are required to progress through a system. See *Memory types* on page A4-65.
AWPROT	Master	Protection type. This signal indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access. See *Access permissions* on page A4-71.
AWQOS	Master	*Quality of Service*, QoS. The QoS identifier sent for each write transaction. Implemented only in AXI4. See *QoS signaling* on page A8-98.
AWREGION	Master	Region identifier. Permits a single physical interface on a slave to be used for multiple logical interfaces. Implemented only in AXI4. See *Multiple region signaling* on page A8-99.
AWUSER	Master	User signal. Optional User-defined signal in the write address channel. Supported only in AXI4. See *User-defined signaling* on page A8-100.
AWVALID	Master	Write address valid. This signal indicates that the channel is signaling valid write address and control information. See *Channel handshake signals* on page A3-38.
AWREADY	Slave	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals. See *Channel handshake signals* on page A3-38.

B1.4 Conversion, protection, and detection

Connection of an AXI4-Lite slave to an AXI4 master requires some form of adaptation if it can not be ensured that the master only issues transactions that meet the AXI4-Lite requirements.

This section describes techniques that can be adopted in a system design to aid with the interoperability of components and the debugging of system design problems. These techniques are:

Conversion This requires the conversion of all transactions to a format that is compatible with the AXI4-Lite requirements.

Protection This requires the detection of any non-compliant transaction. The non-compliant transaction is discarded, and an error response is returned to the master that generated the transaction.

Detection This requires observing any transaction that falls outside the AXI4-Lite requirements and:

- notifying the controlling software of the unexpected access
- permitting the access to proceed at the hardware interface level.

B1.4.1 Conversion and protection levels

Different levels of conversion and protection can be implemented:

Full conversion

This converts all AXI transactions, as described in *Defined conversion mechanism* on page B1-125.

Simple conversion with protection

This propagates transactions that only require a simple conversion, but suppresses and error reports transactions that require a more complex task.

Examples of transactions that are propagated are the discarding of one or more of **AxLOCK** and **AxCACHE**.

Examples of transactions that are discarded and generate an error report are burst length or data width conversions.

Full protection

Suppress and generate an error for every transaction that does not comply with the AXI4-Lite requirements.

B1.4.2 Implementation considerations

A protection mechanism that discards transactions must provide a protocol-compliant error response to prevent deadlock. For example, in the full AXI protocol, read burst transactions require an error for each beat of the burst and a correctly asserted **RLAST** signal.

Using a combination of detection and conversion permits hardware implementations that:

- do not prevent unexpected accesses from occurring
- provide a mechanism for notifying the controlling software of the unexpected access, so speeding up the debug process.

In complex designs, the advantage of combining conversion and detection is that unforeseen future usage can be supported. For example, at design time it might be considered that only the processor programs the control register of a peripheral, but in practice, the peripheral might need to be programmed by other devices, for example a DSP or a DMA controller, that cannot generate exactly the required AXI4-Lite access.

The advantages and disadvantages of the different approaches are:

- Protection requires a lower gate count.
- Conversion ensures the interface can operate with unforeseen accesses.
- Conversion increases the portability of software from one system to another.