Portland State University
ECE571 – Intro to SystemVerilog
Fall 2021

AXI4 LITE
VERIFICATION REPORT

Group 1

Pradeep Reddy M

Narendra Srinivas R

Naveen Manivannan

Table of Contents

Introduction	3
Block Diagram	4
Handshake Protocol	5
Write Transaction	6
Read Transaction	7
Signals Used	8
States of the Design	10
Unit level testing and testcases	14
Assertions used	15
Bugs detected	17
Coverage report	18
Lessons learned	19
Next steps	20
Work split-up	21

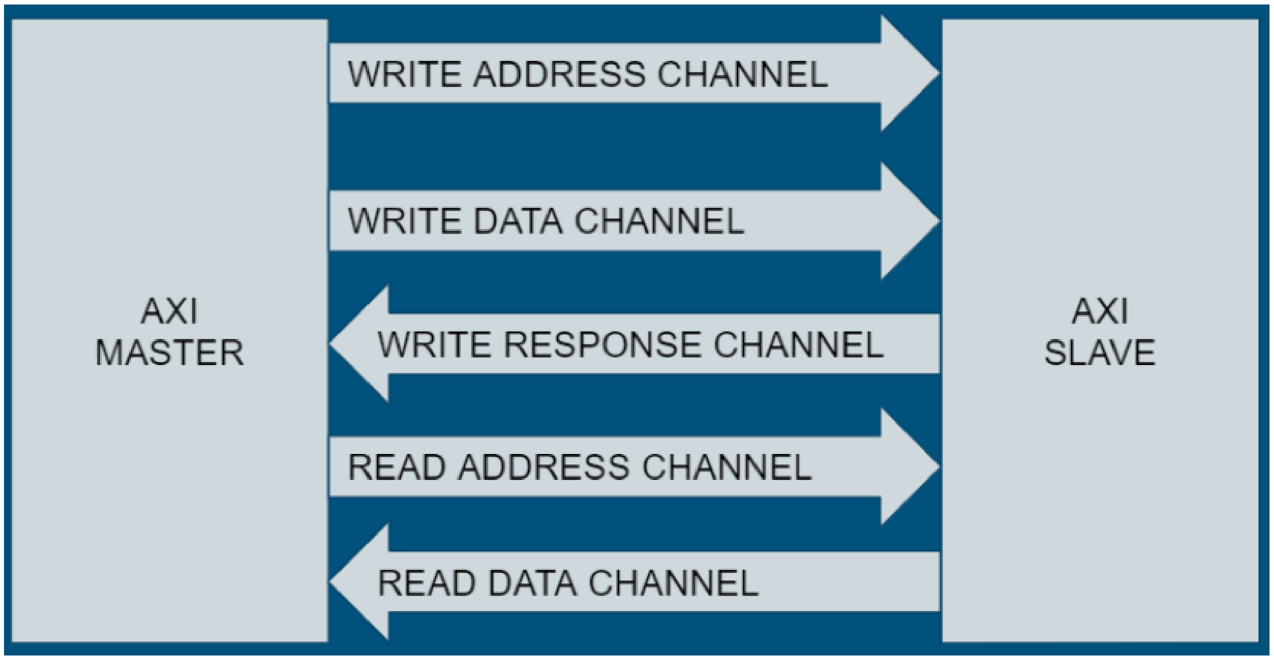
Introduction:-

The AMBA AXI-Lite protocol is the traditional version and subset of AXI(Advance eXtensible Interface used for communication with simpler control register style interfaces within components. It can have multiple masters and slaves which are synchronous has high performance and frequency. It is mainly used for on-chip communication.

In our project we are using a single master and slave environment which communicates using five channels to ensure there are successful transactions(Read and Writes) between the master and slave. We need to make sure that the Reads and writes of this master slave system is working as expected. The five channels being used here are Write Address Channel, Write Data Channel, Write Response Channel, Read Address Channel, Read Data Channel.

Each channel has its own VALID/READY and uses the same for handshake, in order to transfer control and data information Each channel uses specific signals which are listed in the upcoming slides. The functionalities, description and direction of the signals in each channel and who drives the signal is listed in upcoming tables. We have also listed the Testcases using which we intend to find the bugs in the Design.

Block Diagram:-



- Master slave communicates using the following slides.
- Uses unidirectional communication.
- Master is a processor and slave is a memory system.
- Master and slave can be designed using two FSMs each.

Five Channels and their description:

Write Address Channel: Write Address channels carry the control information of the data to be transferred by the master into the slave memory.
Write Data Channel: Write Data channel involves writing the data from master to slave
Write Response Channel: Slave asserts the valid signal once the write completes that was initiated by the master.
Read Address Channel: Read Address channels carry the control information of the data to be transferred between the master and the slave.
Read Data Channel: Read Data channel involves reading the data at a specified address from slave memory by master.

Handshake protocol description:

In AXI4 Lite protocol, each channel has its own VALID/READY and uses the same for handshake, in order to transfer control and data information. When the data or control information is available, the source asserts the VALID signal to indicate. The destination asserts the READY signal to indicate that it is available to accept the data or control information. The transfer happens only if both the VALID and READY signals are ASSERTED. Source sends the next DATA or de-asserts VALID. Destination de-asserts READY if it is no longer able to accept DATA.

Write Transaction:-

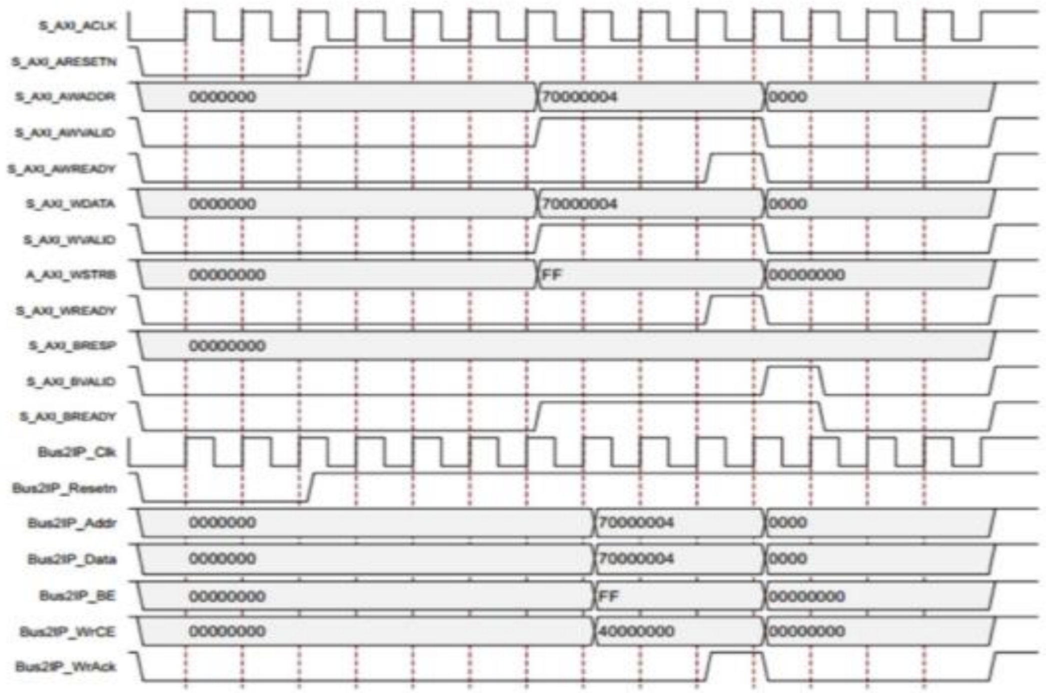
1. Master gets the address available on Write Address Channel and gets the data available on Write Data channel, then indicates address and data are valid by asserting AWVALID and WVALID respectively. Master also asserts BREADY to indicate its ready to receive response from the slave.

2. Slave will assert AWREADY and WREADY in response on Write Address Channel and Write data Channel respectively.

3. READY signals on both Write Address and Data channel handshake happens
and then the READY signals can be deasserted.

4. Slave will then assert BVALID to indicate a valid response on the Write Response Channel and on the clock transaction is considered done.

The write transaction can be depicted using below figure.



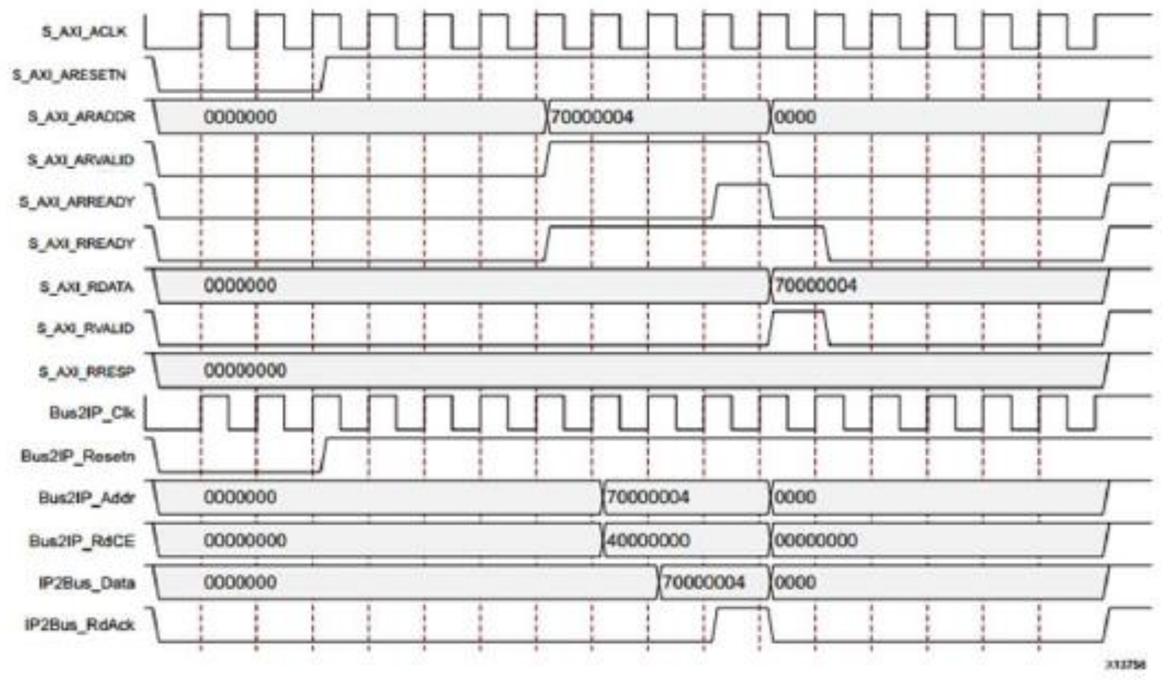
Read Transaction:-

1. Master gets the address available on Read Address Channel as well as assert ARVALID to indicate that address is valid, and then assert RREADY to indicate master is ready to accept data from slave.

2. After the slave indicates its ready to accept address, the handshake process begins and the slave will place data on the Read Address Channel and assert RVALID to indicate data is valid.

3. Since both RREADY and RVALID is asserted the transaction is consdeired to be completed in next cycle and both the signals will be deasserted further.

The read transaction can be depicted using below figure.



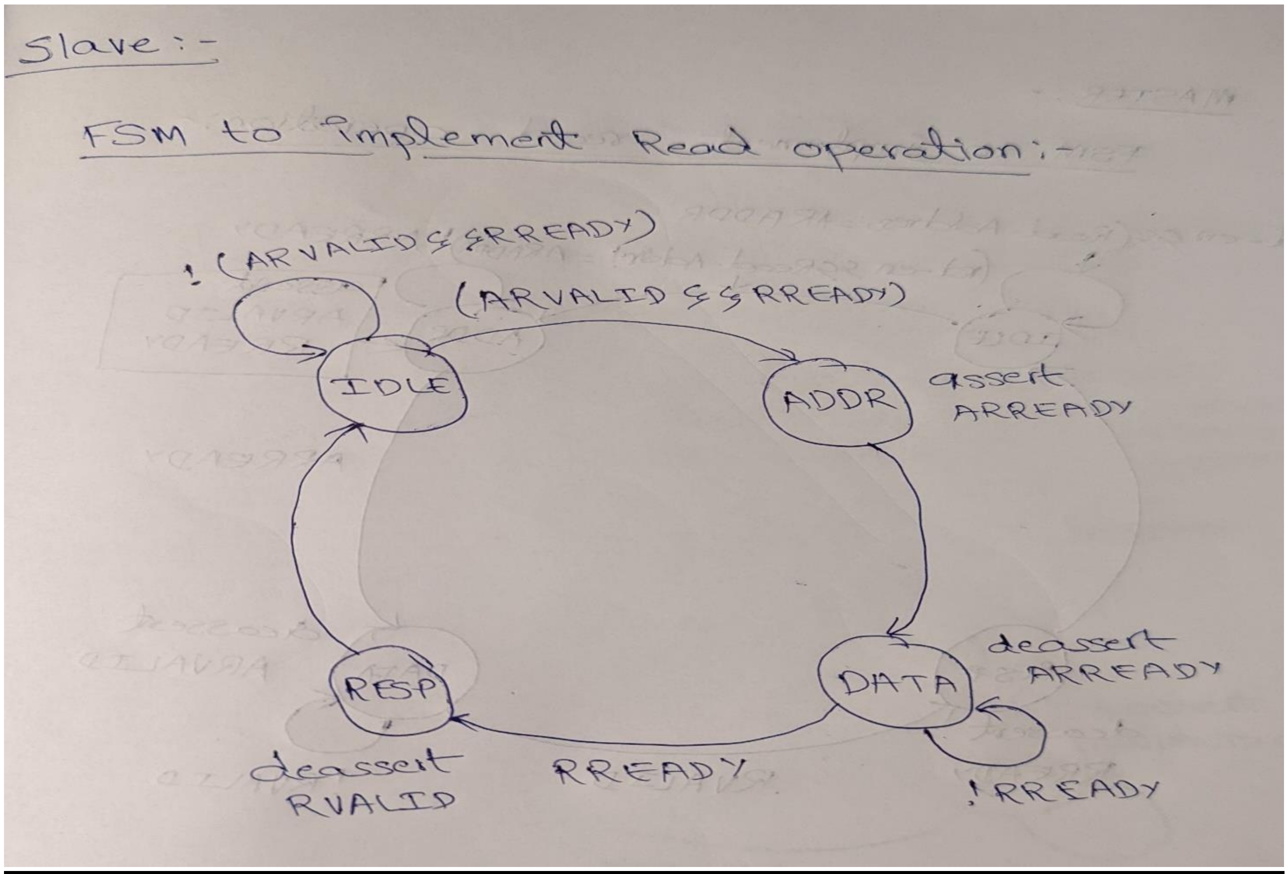
Signals used:-

NO	SIGNAL NAME	DESCRIPTION	DIRECTION	CHANNEL
1	ARESETN	SYSTEM RESET (ACTIVE LOW)	NONE	ALL CHANNEL
2	ACLK	SYSTEM CLOCK	NONE	ALL CHANNEL
3	AWADDR	WRITE ADDRESS VALID, ADDRESS TO BEWRITTEN INTO	NONE	WRITE ADDRESS CHANNEL
4	AWVALID	WRITE ADDRESSVALID, SETS HIGH IF ADDRESS IS VALID	MASTER $\rightarrow$ SLAVE	WRITE ADDRESS CHANNEL
5	AWREADY	WRITE ADDRESS READY, WHEN SLAVE ISREADY TO ACCEPT	SLAVE $\rightarrow$ MASTER	WRITE ADDRESS CHANNEL
6	WDATA	WRITE DATA, DATA TO BE WRITTEN INTO	NONE	WRITE DATA CHANNEL
7	WVALID	WRITE VALID, SETS HIGH IF DATA IS VALID	MASTER $\rightarrow$SLAVE	WRITE DATA CHANNEL
8	WREADY	WRITE READY, WHEN SLAVE IS READY TOACCEPT	SLAVE $\rightarrow$ MASTER	WRITE DATA CHANNEL
9	BVALID	WRITE RESPONSE VALID, SLAVE GENERATESWHEN WRITE RESPONSE ON BUS VALID	SLAVE $\rightarrow$ MASTER	WRITE RESPONSE CHANNEL
10	BREADY	READ ADDRESS READY, WHEN MASTER CANACCEPT WRITE RESPONSE	MASTER $\rightarrow$SLAVE	WRITE RESPONSE CHANNEL
11	ARADDR	READ ADDRESS, ADDRESS TO BE READ	NONE	READ ADDRESS CHANNEL
12	ARVALID	READ ADDRESS VALID, SETS HIGH IF READADDRESS AND CONTROL SIGNALS ARE VALID	MASTER $\rightarrow$ SLAVE	READ ADDRESS CHANNEL

13	ARREADY	READ ADDRESS READY, WHEN SLAVE ISREADY TO ACCEPT READ ADDRESS ANDCONTROL SIGNAL	SLAVE → MASTER	READ ADDRESS CHANNEL
14	RDATA	READ DATA, DATA TO BE READ	NONE	READ DATA CHANNEL
15	RVALID	READ VALID, SETS HIGH IF READ DATA ISVALID	SLAVE → MASTER	READ DATA CHANNEL
16	RREADY	READ READY, WHEN MASTER CAN ACCEPTREAD DATA	MASTER→ SLAVE	READ DATA CHANNEL
17	rd_en	READ ENABLE, SETS HIGH TO ACCESS ALLREAD CHANNEL	NONE	CONTROL SIGNAL
18	wr_en	WRITE ENABLE, SETS HIGH TO ACCESS ALLWRITE CHANNEL	NONE	CONTROL SIGNAL

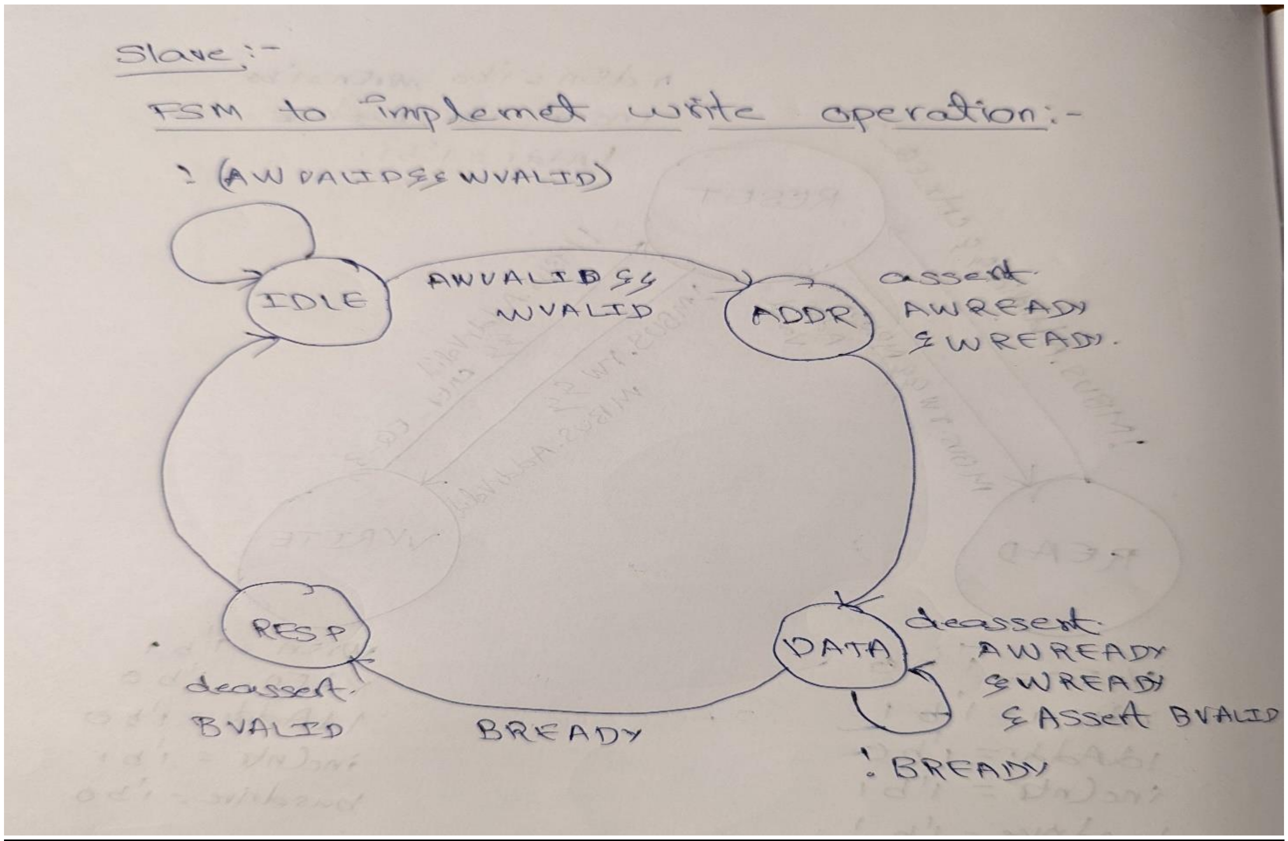
States of the design:-

FSM for slave read operation:-



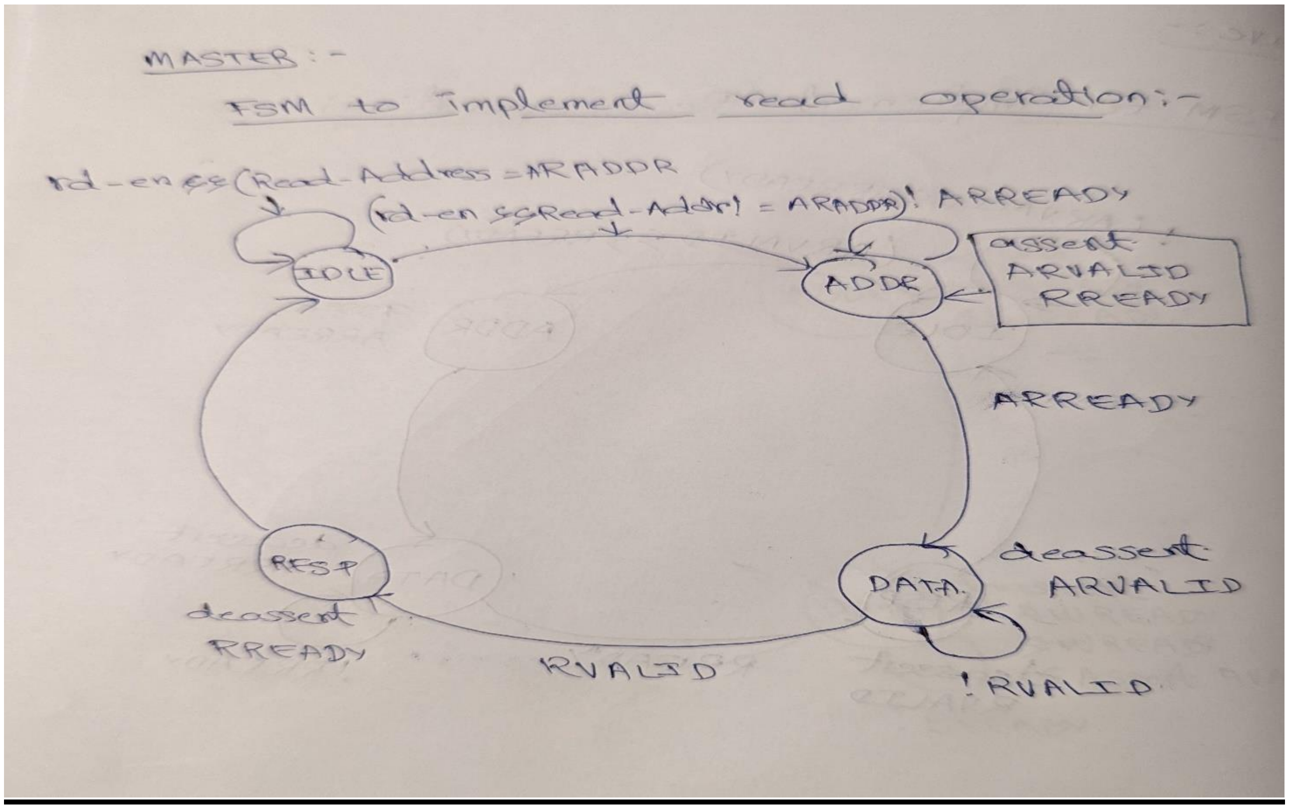
STATE	SIGNAL/S	NEXT STATE
IDLE	!(ARVALID && RREADY)	IDLE
IDLE	ARVALID && RREADY	ADDR
ADDR	[Assert] ARREADY	DATA
DATA	[Deassert] ARREADY, [Assert] RVALID, !RREADY	DATA
DATA	RREADY	RESP
RESP	[Deassert] RVALID	IDLE

FSM for Slave write operation:-



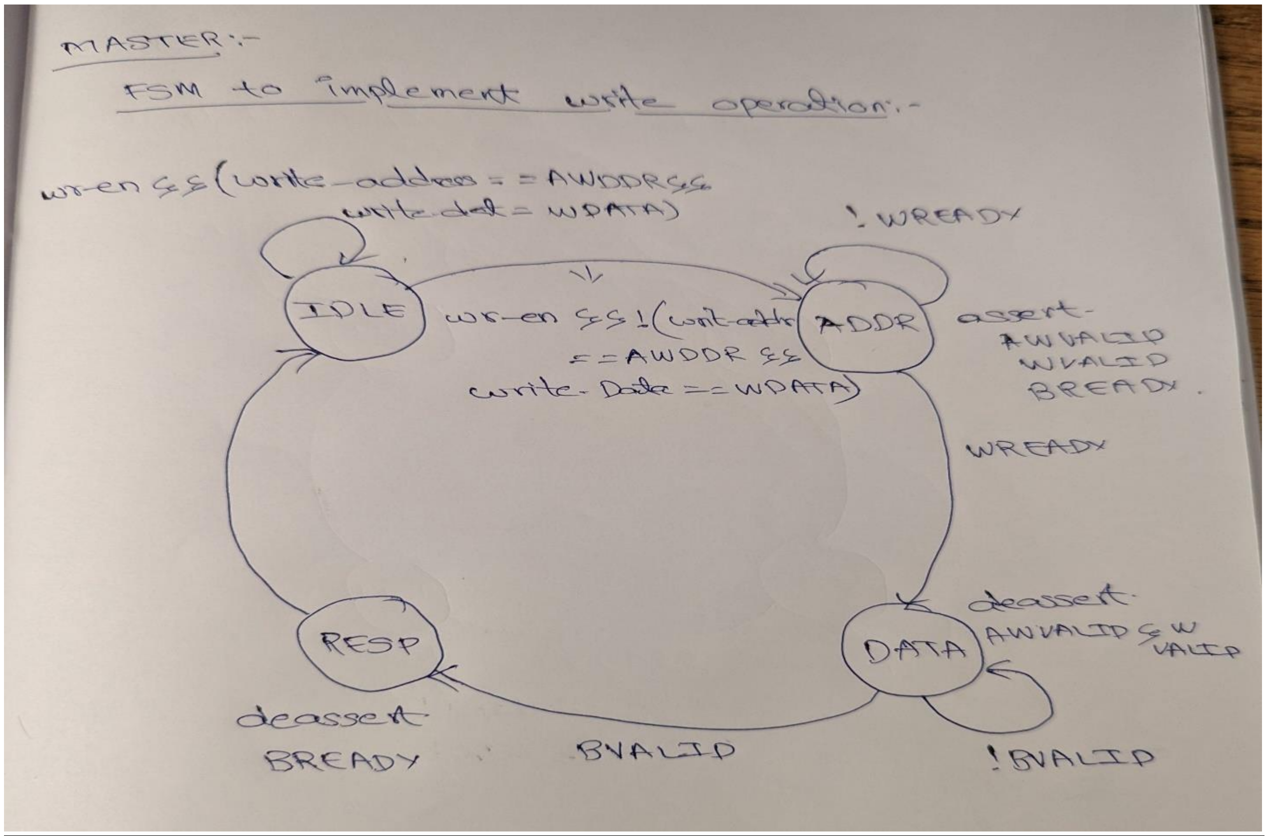
STATE	SIGNAL/S	NEXT STATE
IDLE	!(AWVALID && WREADY)	IDLE
IDLE	AWVALID && WVALID	ADDR
ADDR	[Assert] AWREADY, [Assert] WREADY	DATA
DATA	[Deassert] (AWREADY && WREADY), [Assert] BVALID, !BREADY	DATA
DATA	BREADY	RESP
RESP	[Deassert] BVALID	IDLE

FSM for Master read:-



STATE	SIGNAL/S	NEXT STATE
IDLE	Read_Enable && ARADDR	IDLE
IDLE	Read_Enable && !(ARADDR)	ADDR
ADDR	[Assert] (ARVALID, RREADY), !ARREADY	ADDR
ADDR	ARREADY	DATA
DATA	[Deassert] (ARVALID), !RVALID	DATA
DATA	RVALID	RESP
RESP	[Deassert] RREADY	IDLE

FM for master Write:-



STATE	SIGNAL/S	NEXT STATE
IDLE	Write_Enable && AWADDR && WDATA	IDLE
IDLE	Write_Enable && !(AWADDR && WDATA)	ADDR
ADDR	[Assert] (AWVALID, WVALID, BREADY), !WREADY	ADDR
ADDR	WREADY	DATA
DATA	[Deassert] (AWVALID, WVALID), !BVALID	DATA
DATA	BVALID	RESP
RESP	[Deassert] BREADY	IDLE

Unit Level Testing and Test cases:-

We will be testing the 5 channels of AXI4-LITE namely Write Address channel, Write Data channel, Write Response channel, Read Address channel, Read Data channel. We will be writing directed test cases to cover the functionalityof each unit by simulating the behavior of these channels using assertions and checkers.

Global Test cases:-

- When ARESETN is low, all handshake signals should be low.
- Testing the address and data bus for 32 bit width.
- Check validity of ACLK before applying stimulus.

General test cases:

- To test if the design is working as expected.
- Write to a location
- Read from a location
- Read and write to a location which is not in memory.

The names and additional tests are clearly mentioned in the below table.

Test Case Name	Description
Test_WR_Location	Write to a given location in the memory
Test_RD_Location	Read from a given location in the memory
Test_WR_Last	Write to the last location in the memory
Test_RD_Last	Read from the last location in the memory
Test_WR_Outofrange	Write to the location not in memory
Test_RD_Outofrange	Read from a location not in memory
Test_WR_RD	Write and then read that from the same location to make sure that the data is written correctly
Test_RD_WR	Read and then write that location in the memory
Test_WR_Successive	Multiple writes to the memory to check for any issues in the system
Test_RD_Successive	Multiple reads to the memory to check for any issues in the system

Assertions Used:

Various assertions are written and deployed in each channel to verify the AXI-4 lite design and identify bugs.

The names of the assertions and their description is given below.

Assertion Name	**Description**	**Channel**
AWADDR_STABLE_a	AWADDR remains stable when AWVALID is asserted and AWREADY is LOW	Write address channel
AWVALID_STABLE_a	When AWVALID is asserted, then it remains asserted until AWREADY is HIGH	Write address channel
AWADDR_xcheck_a	A value of X on AWADDR is not permitted when AWVALID is HIGH	Write address channel
WDATA_stable_a	WDATA remains stable when WVALID is asserted and WREADY is LOW	Write Data Channel
WVALID_stable_a	When WVALID is asserted, then it must remain asserted until WREADY is HIGH	Write Data Channel
WDATA_xcheck_a	A value of X on WDATA valid byte lanes is not permitted when WVALID is HIGH	Write Data Channel
ARADDR_stable_a	ARADDR remains stable when ARVALID is asserted and ARREADY is LOW	Read Address Channel
ARVALID_stable_a	When ARVALID is asserted, then it remains asserted until ARREADY is HIGH	Read Address Channel
ARADDR_xcheck_a	A value of X on ARADDR is not permitted when ARVALID is HIGH	Read Address Channel
RDATA_stable_a	RDATA remains stable when RVALID is asserted, and RREADY is LOW	Read Data Channel

RVALID_stable_a	When RVALID is asserted, then it must remain asserted until RREADY is HIGH	Read Data Channel
RDATA_xcheck_a	A value of X on RDATA is not permitted when RVALID is HIGH	Read Data Channel

<u>Bugs detected using test cases and assertions in the broken code:-</u>

ASSERTION ERRORS FOUND:

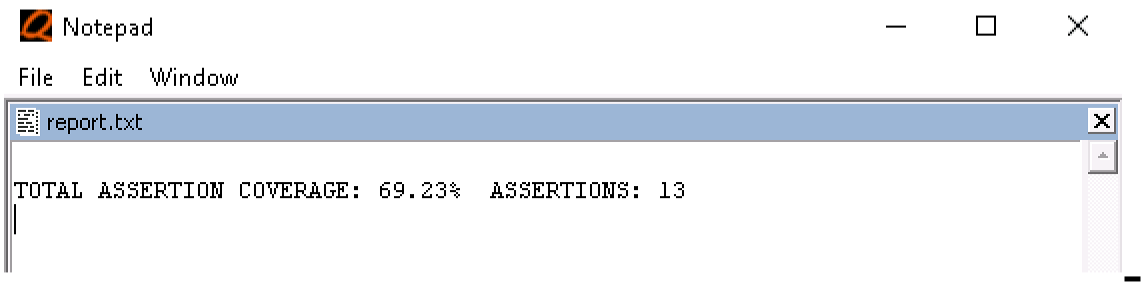
RDATA_xcheck_a:- The RDATA contains x values when RVALID was high.
AWADDR_stable_a :- AWADDR was not stable, when AWVALID was asserted and AWREADY was low
WDATA_stable_a:- WDATA was not stable, when WVALID was asserted and WREADY was low.

DATA ERRORS FOUND:

The Data fetched was not matched with expected data(TBMEM).The data was left shifted by 1 bit.

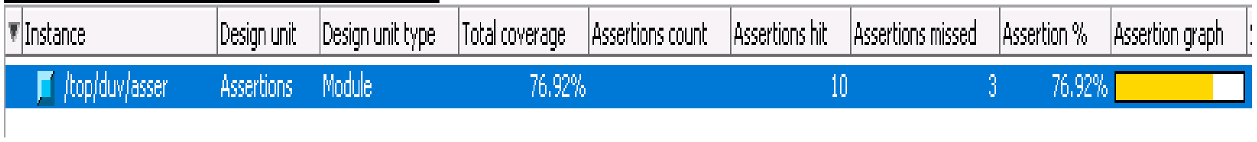
Coverage report from QuestaSim:-

ASSERTION COVERAGE:-



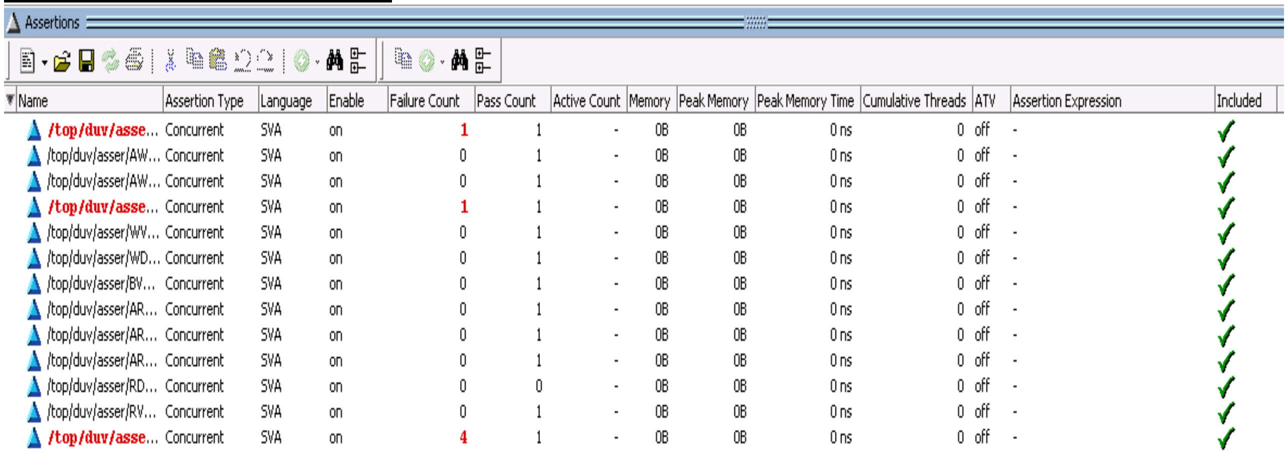
```
TOTAL ASSERTION COVERAGE: 69.23%  ASSERTIONS: 13
```

TOTAL COVERAGE:-



Instance	Design unit	Design unit type	Total coverage	Assertions count	Assertions hit	Assertions missed	Assertion %	Assertion graph
/top/duv/asser	Assertions	Module	76.92%		10	3	76.92%	

ASSERTIONS HIT:-



Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/top/duv/asse...	Concurrent	SVA	on	1	1	-	0B	0B	0 ns	0	off	-	✓
/top/duv/asser/AW...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/top/duv/asser/AW...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/top/duv/asse...	Concurrent	SVA	on	1	1	-	0B	0B	0 ns	0	off	-	✓
/top/duv/asser/WV...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/top/duv/asser/WD...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/top/duv/asser/BV...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/top/duv/asser/AR...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/top/duv/asser/AR...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/top/duv/asser/AR...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/top/duv/asser/RD...	Concurrent	SVA	on	0	0	-	0B	0B	0 ns	0	off	-	✓
/top/duv/asser/RV...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/top/duv/asse...	Concurrent	SVA	on	4	1	-	0B	0B	0 ns	0	off	-	✓

<u>Lessons learned:</u>

- Writing directed test cases and complete testcases
- Writing assertions
- Using Randomization to generate input.
- Using constraints to control the randomization.
- Creating a testbench environment with scoreboard, monitor, Transaction, Mailbox, Driver and others(although we weren’t able to complete this approach it was quite informative for all our team members).

<u>Next steps:-</u>

- If provided with extra time we would write test case for each and every scenario.

- We would have implemented a better testbench environment using Mailbox, Monitor, Scoreboard, Transactions using Oops and class hierarchy.

- We would have included more assertion to check the behavior of all the control signals.

Work Split-up:-

Responsibilities	**Members**
Environment Setup and Makefile	Pradeep,Srinivas,Naveen
Simulation	Pradeep,Srinivas
Unit level testing for Write Address,Write dataChannel	Srinivas,Naveen
Unit level testing for Read Address,Read dataChannel	Pradeep,Srinivas
Unit level testing for Write Response Channel andGlobal Coverage	Naveen,Pradeep