Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.199

IJCSMC, Vol. 8, Issue. 5, May 2019, pg.100 – 108

Development of Verification Environment for I2C Controller Using System Verilog and UVM

Mohamed Azheruddin[1]; Anand M J[2]
[1]Department of ECE, PESCE, Mandya, India
[2]Assistant Professor, Department of ECE, PESCE, Mandya, India
[1] azheruddin799@gmail.com; [2] anamysore@gmail.com

Abstract— The I2C (Inter-Integrated Circuit) or I²C is one of the serial wired communication protocols, having only two-wire, bi-directional serial peripheral bus which provides serial communication between Processors and microcontrollers. It is generally utilized for joining lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communications. Number of communication protocols are used for both long and short distance communication purpose, I2C(Inter Integrated circuit) is used for short distance communication protocols, it is also used as the Interface between EEPROMS, ADC, DAC and RTC. Nearly 70 % of design effort goes to verification, thus verification methodology is needed to check whether design specifications are preserved or not, which shows the product failure if specifications are not preserved. This paper mainly focuses on Verification of I2C controller that transmits and receives the data to or from a peripheral device by simulation using system verilog and Universal Verification Methodology(UVM), simulated using Questa Sim tool. By the application of OOP(Object Oriented Programming) in system verilog through "class data types", which makes code more manageable and re-usable. The Functionality of I2C is checked through functional coverage for each write and read operation, and with multiple test cases. Functional coverage for Write and Read operation are obtained as 100% and comparison results are obtained in Transcript window.

Keywords— I2C, EEPROM, ADC, DAC, RTC, System verilog, UVM, OOP.

I. INTRODUCTION

As the electronics market is changing quickly and its growth being tremendous it induces designers to go for complex IC design and packing them into small areas. So systems on chip (SOC) are created. Nearly 70 % of design effort goes to verification. Checking of complex design, protecting intellectual property (IP), testing of SOC makes verification a difficult task. Therefore to reduce the complexity of verification, use of System verilog[1] and Universal verification methodology(UVM) verification is the best way[2]. There are various protocols suitable for different communication needs, USB(Universal Serial Bus) used for long distance communication purposes. I2C and SPI are used for short distance communication protocols[5]. I2C (Inter-Integrated Circuit) Controller is a two-wire, bi-directional serial peripheral bus that provides no data loss on transmission and reception. I2C bus was developed by NXP semiconductors formerly Philips semiconductors. The I2C communication is mainly between master and slave. I2C provides chip-to-chip serial communication where master is a device which is used to initiate a data transfer, and slave responds to master through

Acknowledge[6]. Both master and salve can Transmit and receive the Data. The main advantage of I2C protocol is data is transmitted with no Data loss, as compared to the other protocols like SPI[8]. The Data loss is almost Zero because the slave acknowledges for every eight bit of Data transfer. Each I^2C controller consists of only two signals: SCL and SDA. SCL is the clock signal, and SDA is the data signal. The clock signal is always generated by the current bus master. And as there are only two lines the I2C implementation is cost effective and it is widely used for connecting peripherals to the microprocessor or microcontroller. The System Verilog Universal verification methodology(UVM) is based on OVM version 2.1.1 and is created by Accellera. The Class Library of UVM gives the building blocks needed to rapidly develop well-constructed and reusable verification components and test environments[10].

II. INTER INTEGRATED CIRCUIT

I2C consolidates the best features of SPI and UARTs. I2C supports 100 kbps, 400 kbps, 3.4 Mbps data speeds. A few variations additionally underpins 10 Kbps and 1 Mbps. The Figure 1 shows the I2C communication in Two ICs. The I2C Interface mainly Consists of master, Slave, SCL(Serial clock line) and SDA(Serial Data line). Which are the important parameters in I2C communication each of these are described as follows.

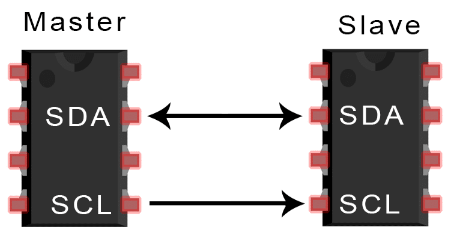


Figure.1: Serial Peripheral Interface

a. Serial Data line (SDA) – It is signal generated by master for transmission through which Start and Stop conditions, Address, Read/Write bit , Data, and Acknowledge bit are transmitted to Slave.
b. Serial Clock Line (SCL) – It is signal generated by master which initiates the data transfer. Used for Synchronizing the Data transfer.
c. Master – The master is basically a IC which always generates the Clock and initiates the data transfer, and addresses the slave.
d. Slave – The slave is also a IC which is addressed by the master and responds to master after every eight bit of data received by sending a Acknowledge.

III.DATA TRANSMISSION IN I2C

I2C provides the serial cmmunication through SCL and SDA lines, where address and data are in synchronizaation with the SCL through SDA. SDA is a single data line which have the multiple states through which Address and Data is transmitted from Master to Salve. Data through SDA is transmitted with the following stages as shown in Figure 2.

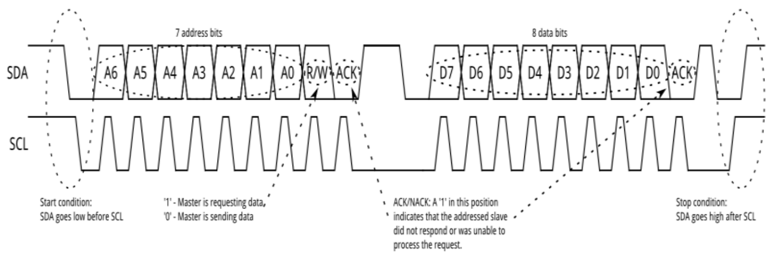

7 address bits
8 data bits
SDA
SCL
A6
A5
A4
A3
A2
A1
A0
R/W
ACK
D7
D6
D5
D4
D3
D2
D1
D0
ACK
Start condition: SDA goes low before SCL
'1' - Master is requesting data, '0' - Master is sending data
ACK/NACK: A '1' in this position indicates that the addressed slave did not respond or was unable to process the request.
Stop condition: SDA goes high after SCL

Figure.2 : I2C frame format

A. Start and Stop condition : Comminication in I2C starts with Start condition and terminates the transmission with a Stop codition. Start codition is when SCL is high SDA makes a transition from High to Low and Stop condition is when SCL is High SDA makes a transition from Low to High.

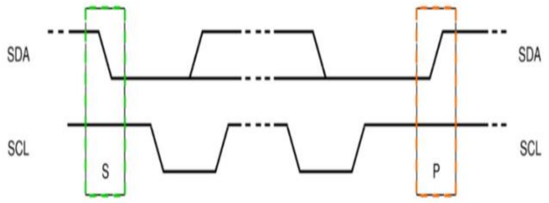


Figure.3 : Start and stop conditions

B. Write operation : Write happens to Slave when slave address followed by R/W bit in Frame becomes 0 (zero), then slave is written with the data send by master. Write condition in I2C frame is shown in Figure 4.

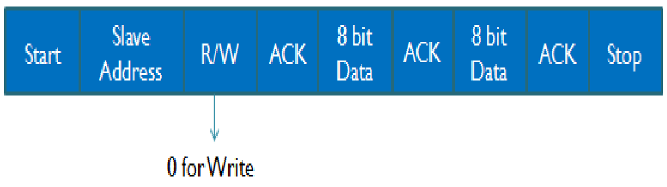


Figure.4 : Write operation in I2C frame

C. Read Operation : Master read data from Slave when slave address followed by R/W bit in Frame becomes 1 (one). The I2C frame during Read condition is shown in Figure 5.

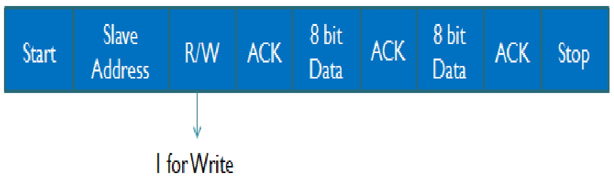


Figure.5 : Read operation in I2C frame

D. Repeated Start Condition : There is often the need, when a data has transmitted and read back in same frame if master wants to read the data written previously, then I2C frame goes through Repeated start condition.

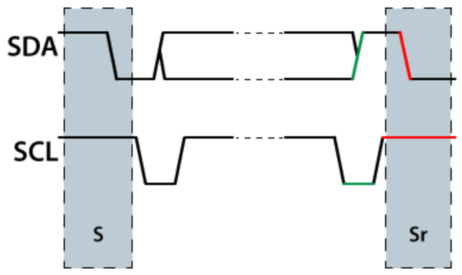


Figure.6 : Repeated start condition in I2C frame

E. Slave Address : Master addresses the slave by sending the correspondent address of its interest to either write or read data to Slave. basically I2C supports 7 and 10 bit addressing.

F. Acknowledge : This is bit sent by slave in response to the Address or Data sent by Master to make sure that Data has been received with no error.

IV. SYSTEM VERILOG

The System Verilog is a hardware description language as well as hardware verification language, which is the most popular Hardware Description Language used for SOC design and verification in semiconductor industry. The HDL design can be simulated by using System Verilog and test case can be used to verify them. Layered test bench is used for complex design. The Design language and Verification language gap is reduced by using System Verilog. System Verilog features has been inherited from Verilog. Object Oriented Programming (OOP) technique is applied in verification environment of System Verilog [1]. The verification environment for I2C using system verilog is as shown in Figure.7.

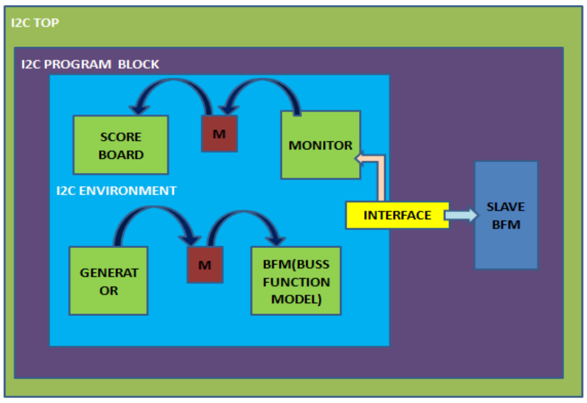


Figure.7 : System Verilog Verification Environment

Components of Verification environment are Generator, BFM(Bus Function Model), Monitor, Scoreboard, interface and Mailbox.

a. Environment is a container class which has components like Generator, BFM, Monitor, Scoreboard, Interface and Mailbox in it. For verifying the protocol a Slave BFM is connected with the Environment through Interface.
b. Where as in environment the Generator component generates different scenarios which are needed to be verified, BFM gets the data from generator through Mailbox.
c. Mailbox is the inter process communication unit which gets data from generator and puts the same to BFM.
d. BFM drives the signals to Interface of slave BFM which here is DUT(Design under test).

The components like Monitor and Scoreboard are used for obtaining the coverage, whose functionality remains the same in system verilog and UVM. Which are described in the later sections. The results obtained for above verification architecture are discussed in the Results section VII.

V. UNIVERSAL VERIFICATION METHODOLOGY(UVM)

Universal Verification Methodology was developed to provide a well structured and reusable verification environment which does not interfere with the design under test (DUT). It is based on OVM version 2.1.1 and is. It's Class Library gives the building blocks needed to rapidly develop well-constructed and reusable verification components and test environments. In UVM, classes are mainly three types namely uvm_object, uvm_transaction and uvm_component[10].

a. UVM_OBJECT : Core methods like copy, print and compare etc.. are defined in UVM_OBJECT class.
b. UVM_TRANSACTION : Stimulus generation and analysis is done in UVM_TRANSACTION class .
c. UVM_COMPONENT : This class is intended to model permanent structures of testbench like monitor, driver and environment.

All UVM components are Dynamic in nature so it is important to have synchronization between all components of Testbench, to do so UVM introduces the concept of Phases. The whole verification environment in UVM is structured of UVM Phases. The UVM phases are the set of function and task, the phases are described below. The phases of UVM are shown Figure 8.

Build Phase

a. **Build phase (uvm_build_Phase)** : The Build phase create, configure the testbench structure.
b. **Connect phase (uvm_connect_phase**) : Connect phase establishes cross-component connections in the components of verification.
c. **End of elaboration phase (uvm_end_of_elaboration_phase)** : The End of elaboration phase fine-tunes the testbench such that simulation could be started.
d. **Start of simulation phase (uvm_start_of_simulation_phase)** : Start of simulation phase gets ready the DUT in the verification plan to be simulated for given stimulus by verification environment.

Run phase

a. **Run phase (uvm_run_phase)** : Run phase is task which consumes the simulation time, which have four steps in it. They are Reset, configure, main and shutdown. This phase simulates the Design Under test (DUT).

Clean up phase

a. **Extract phase (uvm_extract_phase)** : Extract phase extract data from different points of the verification environment.
b. **Check phase (uvm_check_phase)**: Check phase checks for any unexpected conditions in the verification environment.
c. **Report phase (uvm_report_phase)**: Report phase reports results of the test to Scoreboard.
d. **Final phase (uvm_final_phase)** : Final phase ends the simulation.

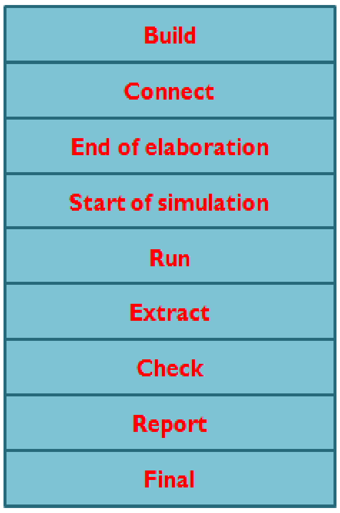


Figure.8 : Phases of UVM

VI. ARCHITECTURE OF I2C VERIFICATION ENVIRONMENT

The Figure 9 shows the UVM verification environment for I2C. Where the Master and Salve are implemented as agents using UVM methodology. Here for verification, the slave is made as DUT which takes input from Master and responds to the request by the master, by data and acknowledge. The Components are inherited from UVM_COMPONENT base class, which act as the parent component to most of the components of verification. All components are the extended components of uvm_base class library hierarchy form where pre-existed parent component is modified based upon the need of the verification.

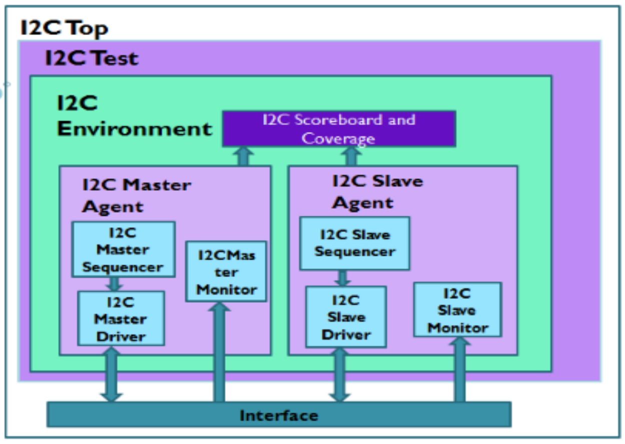
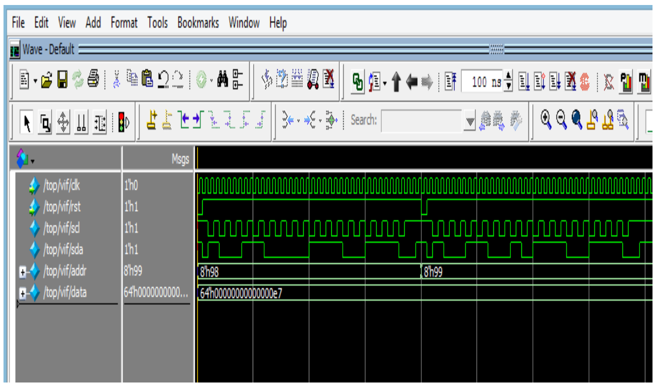


Figure.9 : UVM Verification Environment

The verification environment mainly consists of I2C Top which is top most module, which consists of I2C Test and Interface, where I2C Test class consists of I2C environment. I2C Environment class is a container class which is having I2C master agent, I2C slave agents, and I2C scoreboard and coverage. Each agent in I2C environment has Monitor, Driver and Sequencer. I2C master sequencer generates data transaction as class objects and sends it to the Driver, sequencer is the series of transactions. I2C monitor class is able to capture the signal activity from design Interface and translate it into transaction level data objects that can be sent to other components. The slave I2C driver class is able to receive data from Interface which gets the data from master agent and same operations are performed by the I2C Slave agent components as that of I2C master components.

VII. RESULTS AND DISCUSSIONS

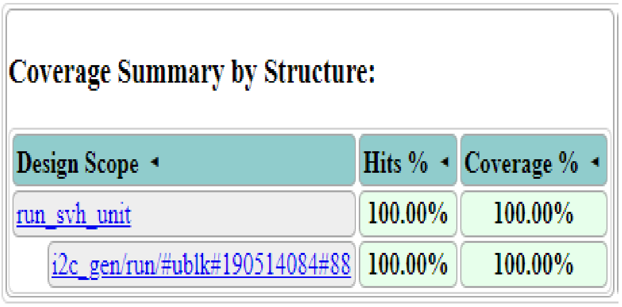
In the proposed system simulation results for multiple test cases of the I2C are described in this section. The standard of using System verilog and UVM based verification gives more reliable results. In Figure 10 a byte of data is transmitted first from master to salve with address 1001100 and W/R bit 0 in first transfer indicating write operation is happening. Followed by same data is Read back with the same address 1001100 and R/W bit 1 indicating read operation. The results shown in Figure 10 are results for system verilog verification environment.



```
/top/vif/clk    1'h0
/top/vif/rst    1'h1
/top/vif/scl    1'h1
/top/vif/sda    1'h1
/top/vif/addr   8'h99               8'h98   8'h99
/top/vif/data   64'h0000000000...   64'h00000000000000e7
```

Figure.10 : Write and Read in I2C for single byte transfer

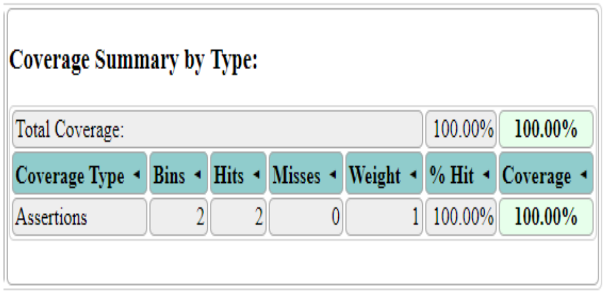
The coverage results for single byte data transfer in the proposed System verilog verification environment is shown in the Table 1. The coverage report Questa Coverage Summary is generated by Questa Sim tool which shows the coverage for write and read of single byte in verification environment of I2C controller.



Coverage Summary by Structure:

Design Scope	Hits %	Coverage %
run_svh_unit	100.00%	100.00%
i2c_gen/run/#ublk#190514084#88	100.00%	100.00%

Table.1.a : Coverage obtained for single data byte



Coverage Summary by Type:

Total Coverage:					100.00%	100.00%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Assertions	2	2	0	1	100.00%	100.00%

Table.1.b : Coverage obtained for single data byte

The multiple byte transfer in the proposed UVM verification environment is shown in Figure 11. Where multiple bytes of data is written into the slave and simultaneously read back. The multiple bytes of data in I2C is transmitted through SDA where, a ACK bit use to represent each byte of data received. Also the Figure 11 shows the four Bytes of data transfer which is written first to the slave and Read back simultaneously from same address.

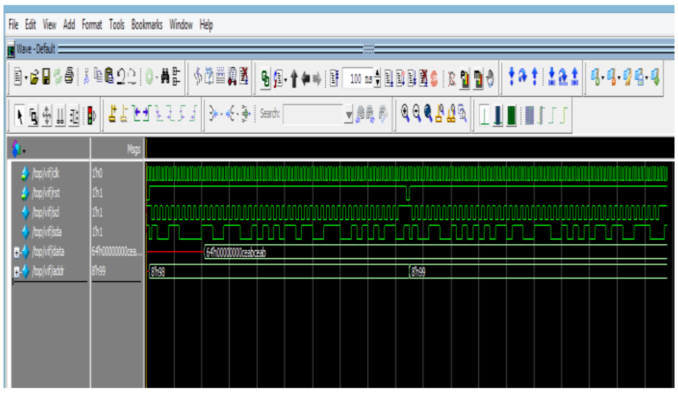
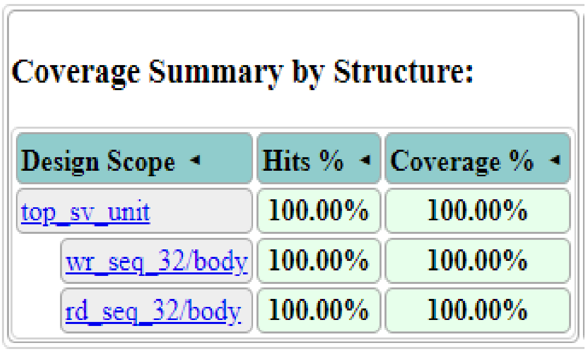


Figure.11 : Write and Read in I2C for multiple data byte transfer

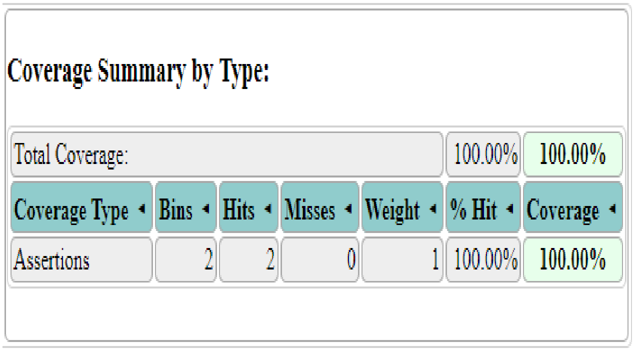
The coverage results for multiple byte data transfer in the proposed UVM verification environment is shown in the below Table 2. This coverage report shows functional coverage of multiple byte data transfer in verification of I2C controller indicating that both the operations write and read happened with no data loss and all specifications of I2C are met.



Coverage Summary by Structure:

Design Scope	Hits %	Coverage %
top_sv_unit	100.00%	100.00%
wr_seq_32/body	100.00%	100.00%
rd_seq_32/body	100.00%	100.00%

Table.2.a : Coverage obtained for multiple data byte

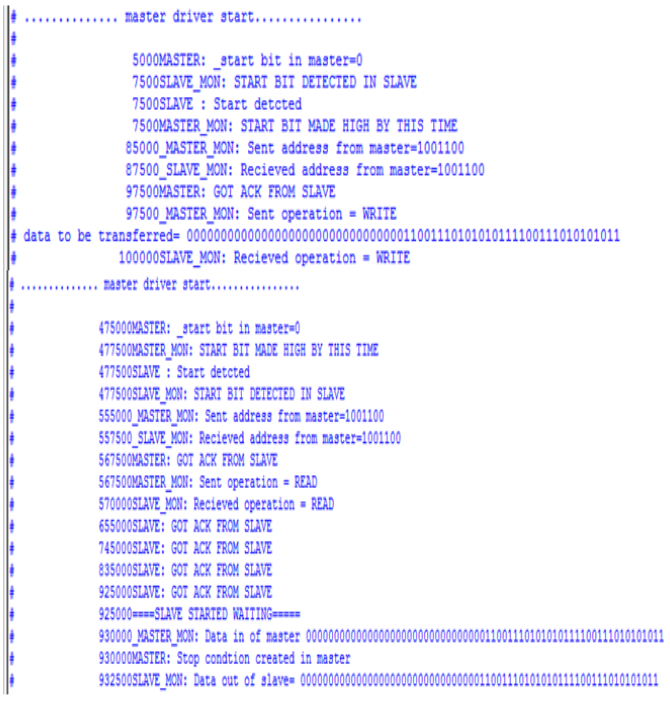


Coverage Summary by Type:

Total Coverage:					100.00%	100.00%
Coverage Type	Bins	Hits	Misses	Weight	% Hit	Coverage
Assertions	2	2	0	1	100.00%	100.00%

Table.2.b : Coverage obtained for multiple data byte

The results obtained in transcript window are in comparison with the data transfer, the shake hand between master and slave is depicted in the Transcript window where the request and response of master and slave respectively is shown in reference with the simulation results in Figure 12. The shake hand in I2C protocol shows the corresponding request and response of master and slave, after each conditions of Protocol standard are executed a comment indicating that condition is displayed in the Transcript window.



```
# .............. master driver start................
#
#                 5000MASTER: _start bit in master=0
#                 7500SLAVE_MON: START BIT DETECTED IN SLAVE
#                 7500SLAVE : Start detcted
#                 7500MASTER_MON: START BIT MADE HIGH BY THIS TIME
#                85000_MASTER_MON: Sent address from master=1001100
#                87500_SLAVE_MON: Recieved address from master=1001100
#                97500MASTER: GOT ACK FROM SLAVE
#                97500_MASTER_MON: Sent operation = WRITE
# data to be transferred= 0000000000000000000000000000000011001110101010111100111010101011
#               100000SLAVE_MON: Recieved operation = WRITE
# ............. master driver start...............
#
#            475000MASTER: _start bit in master=0
#            477500MASTER_MON: START BIT MADE HIGH BY THIS TIME
#            477500SLAVE : Start detcted
#            477500SLAVE_MON: START BIT DETECTED IN SLAVE
#            555000_MASTER_MON: Sent address from master=1001100
#            557500_SLAVE_MON: Recieved address from master=1001100
#            567500MASTER: GOT ACK FROM SLAVE
#            567500MASTER_MON: Sent operation = READ
#            570000SLAVE_MON: Recieved operation = READ
#            655000SLAVE: GOT ACK FROM SLAVE
#            745000SLAVE: GOT ACK FROM SLAVE
#            835000SLAVE: GOT ACK FROM SLAVE
#            925000SLAVE: GOT ACK FROM SLAVE
#            925000====SLAVE STARTED WAITING=====
#            930000_MASTER_MON: Data in of master 0000000000000000000000000000000011001110101010111100111010101011
#            930000MASTER: Stop condtion created in master
#            932500SLAVE_MON: Data out of slave= 0000000000000000000000000000000011001110101010111100111010101011
```

Figure.12 : Results in transcript window

VIII. CONCLUSION

The verification environment was created using System Verilog and UVM. proposed UVM verification environment consists of Agents, Monitor, Driver, Sequencer, Sequence and Scoreboard. Which are implemented using Object oriented Programming(OOP) concept. Assertion based technique is used in the verification environment. Functional coverage obtained was 100% in Questa Coverage Summary. Further, this work can be extended to verify multiple masters and slaves, and Interconnect can be implemented.

ACKNOWLEDGEMENT

Centre of Excellence in VLSI design, Department of Electronics and Communication Engineering PESCE, Mandya and VLSI Guru Bangalore. First I would like to express my gratitude to my Guide Mr. Anand M J and also our HOD Dr. K A.Radhakrishna Rao who gave me the opportunity to do this project on the topic Development of Verification Environrmnet for I2C Controller Using System Verilog and UVM, I would also like to thanks to my parents and friends.

REFERENCES

[1]. K.Gavaskar and M.Sukhanya , Kongu Engineering College Erode, India “Functional Verification Environment for I2C Master Controller using System Verilog” International Conference on Signal Processing, Communications and Networking (ICSCN -2017) IEEE ,March 2017, Chennai, INDIA.
[2]. Deepa Kaith, Janakkumar B.Patel, and Neeraj Gupta, “An Introduction to Functional Verification of I2C Protocol using UVM,” International Journal of Computer Applications, vol.121, Issue.13, pp. 10-14, July 2015.
[3]. Shoaib. Shah Sobhan, Sudipta. Das and Iqbalur Rahman “Implementation of I²C using System Verilog and FPGA” International Conference on Advancements in Electronics and Power Engineering (ICAEPE'2011) Bangkok Dec., 2011. [4]. I2C-bus specification and user manual Rev. 6 — 4 April 2014 by NXP Semiconductors i.e early Philips Semiconductors.
[5]. F.Leens, February 2009. An Introduction to I2C and SPI Protocols, IEEE Instrumentation & Measurement Magazine, pp. 8-13.
[6]. Shanthipriya and Lakshmi Chennai, Tamil Nadu, India “design and verification of low speed peripheral subsystem supporting protocols like spi, i2c and uart” ARPN Journal of Engineering and Applied Sciences VOL. 12, NO. 24, DECEMBER 2017 ISSN 1819-6608, Chennai, Tamil Nadu, India.
[7]. Anuja Dhar, Ekta Dudi, and Hema Tiwari, “Coverage Driven Verification of I2C Protocol using System Verilog,” International Journal of Advanced Research in Engineering and Technology, vol.7, Issue.3, pp. 103-113, May-June 2016.
[8]. A note on Why to use I2C from Website https://learn.sparkfun.com/tutorials/i2c/all#i2c-at-the-hardware-level
[9]. Yashawini Pandit, and P.B.Manoj, “Verification of I2C Single- Master Multiple-Slave Bus Controller using System Verilog,” International Journal of Electrical, Electronics and Computer Systems, vol.2, Issue.4, pp. 26-28, 2014**.**
[10]. An Accellera Organization, June 2011. Universal Verification Methodology (UVM) 1.1 Class Reference.