CHETAN N, et. al. International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 11, Issue 7, (Series-VI) July 2021, pp. 01-08

RESEARCH ARTICLE OPEN ACCESS

Verification of SPI protocol Single Master Multiple Slaves using Systemverilog and Universal Verification Methodology (UVM)

CHETAN N*, R KRISHNA**
** M. Tech, Department of ECE, Bangalore Institute of Technology, Karnataka, India*
*** Associate Professor, Department of ECE, Bangalore Institute of Technology, Karnataka, India*

ABSTRACT
Integrated circuit designs are ever expanding which makes the verification process increasingly difficult and progressively time consuming therefore there is a need for effective verification of such circuit designs. This results in the need for effective testbench hierarchy, one with significant generic verification components that are quite reusable and can be easily extendable across designs. UVM (Universal Verification Methodology) hierarchy is one such architecture that can realize testbench architectures with coverage driven verification environments with CRT (constrained Random Test). The present work duly focuses on the UVM based verification of SPI Single Master and Multiple Slave protocol in accordance with the verification plan concocted after a full-scale analysis of SPI protocol specifications. The UVM Testbench focuses on generating random vectors that are driven to the SPI module or the DUT (Design Under Test) and makes comparison with the captured response obtained using scoreboard, and this mechanism helps to verify the functionality of SPI. Testbench also substantiates the effective functionality and characteristic features of SPI by applying suitable or appropriate test cases and provides the cumulatively coverage report of the design at the end of the test.
Keywords – EDA-playground mentor questa simulator, Questsim, SPI Protocol, Systemverilog, UVM.

Date of Submission: 10-07-2021 Date of Acceptance: 26-07-2021

I. INTRODUCTION

SPI gained a solid key role in embedded systems which involved system on chip processors, that included higher end 16-bit or 32-bit processors such as the ones used in ARM, Power PC or MIC with other microcontrollers which involves PIC, AVR (Advanced Virtual RISC) and others etc. Chips like these make use of SPI controllers that are capable of running in either Master/ Slave mode or sometimes even in both modes. In-system, programmable AVR controllers are programmed by making use of an SPI interface. Sometimes, Chips or FPGA based designs make use of SPI protocol for communication. That is why, SPI is a quite preferred technology nowadays for communication with peripheral components where data is transferred readily and within given real time constraints. Many serial interfaces are available when observed, right from USB, Morse code telegraphy, Fire wire, RS232, Ethernet [2] and many more. With each of these interfaces offering advantages with some limitations to many designs, depending on certain criteria [1] which involves considerations such as needed data rate, space availability, and noise. Serial Peripheral Interface i.e., SPI is one such technology that was invented to significantly replace parallel interfaces so as to avoid routing parallel bus around PCB [3] providing high-speed data transfer between the devices. The first company to come up with such mechanism for data transfer between two or more devices which involved a master device was Motorola [1]. SPI communication protocol was developed by Motorola in the mid-1980's for inter-chip processing and communication at relatable speed. It is called as a full-duplex synchronous serial communication protocol [2], signifying that the data can be simultaneously transmitted in both directions (bi-directional). The fine advantageous factor of SPI protocol is that, it can transfer the data without any interruption occurrence. This way, many bits can be transmitted and received through this protocol at a time. This protocol depicts the Master-Slave relationship where data reception and transmission occurs simultaneously. The Master device is the main component that controls effectively the Slave device to which it communicates, the Slave device on the other hand is bound to accept the instruction from the master device during communication.

The simplest kind of arrangement for the Serial Peripheral Interface (SPI) is the combination of a single Slave with a single Master [2]. But, however one Master device or module can communicate with more than one Slave device i.e.,

with multiple Slave devices. SPI technology comes with a high-speed, full-duplex and synchronous communication bus protocol, which enables for information transmission between any microcontroller device and related peripherals. However, looking into the aspect of verification, Systemverilog which is considered as a Hardware description language is used in developing a test plan environment for the SPI protocol by implementing oops programming language. With Systemverilog, advanced features help in developing a potential verification environment, but still a standardized verification approach can be done with the implementation of UVM (Universal Verification Methodology).

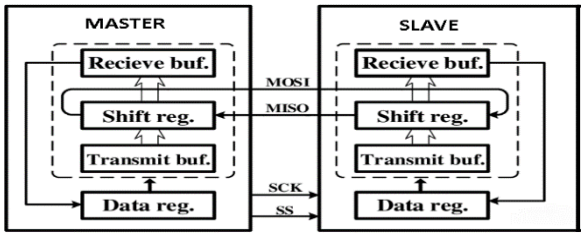


Fig.1: Single Master-Single Slave configuration.

The devices in SPI protocol are connected in Master–Slave relationship fashion as a multi–point interface. At this type of interface, one device takes the role of Master (usually a Microcontroller) and other devices so connected (PICs or even other Microcontrollers) are considered as Slaves.

1.1. Single Slave configuration

A SPI protocol has only one master but many slave devices. The SPI bus protocol consists of 4 signaling pins [4]. They are: -

- (MOSI) Master-Out / Slave-In
- (MISO) Master-In / Slave-Out
- (SCLK or SCK or MK) Serial Clock or Master clock
- (SS) Slave Select pin / (CS) Chip Select pin

The operational functionality of each signal pin is mentioned here: -

• SCK or MK (Serial clock or Master clock): - This pin provides clock signals to Slave or Slaves and only Master can control this clock signal. however, this pin remains in idle state.i.e. inactive (tri-state) when no operation is performed.

• SS or CS (Slave Select or Chip Select): -This pin selects the Slave to which Master module wants to communicate or transfer data.

• MOSI (Master-Out/Slave-In): - This stands as Master output and Slave input pin. This pin is used in transmission of data from Master module to the Slave module. It is a unidirectional pin.

• MISO (Master-In/Slave-Out): - This pin is known as Master input and a Slave output pin. This pin is used in transmission of data from the Slave Module to the Master Module. It is also a unidirectional pin.

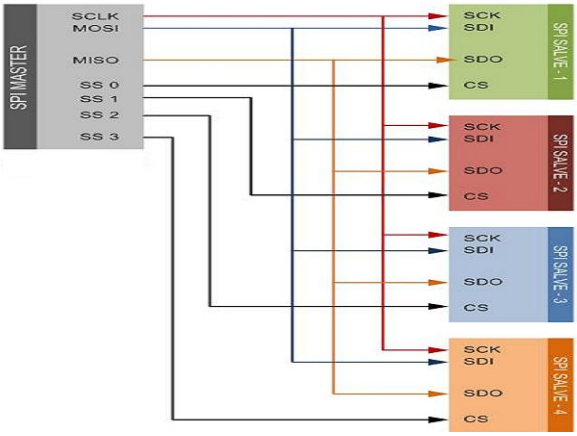


Fig.2: Single Master- Multi Slave.

1.2. Multi Slave Configuration

As multiple Slaves can be implemented with a single SPI Master. The Slaves can be connected as individual modules or in a daisy-chain fashion. In individual configuration, there is individual Chip Select(CS) pin for every Slave module that is controlled by Master module. With the Chip Select (CS) signal being enabled by the Master, the clock generated by the Master module along with the data on the MOSI / MISO lines are accessible for the selected Slave. But, when multiple Chip Select(CS) signal pins are activated, it leads to data corruption on MISO line, since there is no way the Master can identify which Slave is transmitting or receiving the data. By taking a look at Figure 2, with the increase in number of Slaves, the number of Chip Select (CS) pins of the Master gradually increases. This readily adds to the increased number of input and output pins needed which is available from the Master and bounds or limits the number of Slaves that can be implemented. However, different techniques can be adopted to gradually increase the number of Slaves in individual configuration, for instance using a mux module to control a Chip Select(CS) signal.

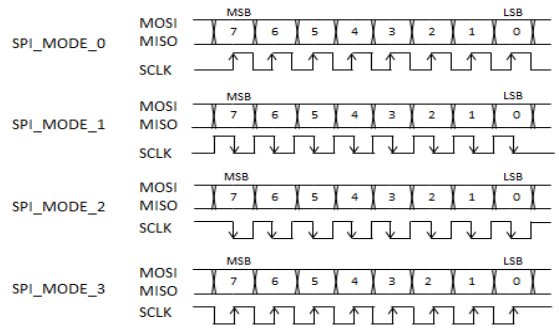


Fig.3: Transfer Modes in SPI.

1.3. SPI Modes of Operation

0^{th} MODE:

In this mode Clock Phase is 0 and Clock Polarity is LOW (CPHA = 0 and CPOL = 0). At Mode 0 configuration, data is sampled during rising edge and pushed or shifted on the falling edge.

1^{st} MODE:

In this mode Clock Phase is 1 and Clock Polarity is LOW (CPHA = 1 and CPOL = 0). At Mode 1 configuration, data is sampled during the falling edge and pushed or shifted on the rising edge.

2^{nd} MODE:

In this mode Clock Phase is 0 and Clock Polarity is HIGH (CPHA= 0 and CPOL = 1). At Mode 2 configuration, data is sampled during the rising edge and pushed or shifted on the falling edge.

3^{rd} MODE:

In this mode Clock Phase is 1 and Clock Polarity is HIGH (CPHA= 1 and CPOL = 1). At Mode 3 configuration, data is sampled during the falling edge and pushed or shifted on the rising edge.

1.4. Features of SPI: -

1. Comes with full duplex communication.
2. Throughput is better and Higher than TWI i.e., I2C (integrated Interface circuit).
3. Not limited to specific bit size, In the case of bit transferring.
4. With Better and Simple hardware interfacing compared to UART and I2C.
5. Power requirement is quite low.
6. Without any need of precision oscillators for Slaves as Slaves uses master's clock.
7. Lower power requirements than I2C due to less circuitry.

II. DESIGN METHODOLOGY

The current work involves a single Master and single Slave SPI configuration, where single Slave module is used as an instance for multiple modules (sub Slaves) those which connects to it. The Figure shown below provides a glimpse of the Design Under Test (DUT) with Master and Slave module which can operate at different frequency with different Slave modules.

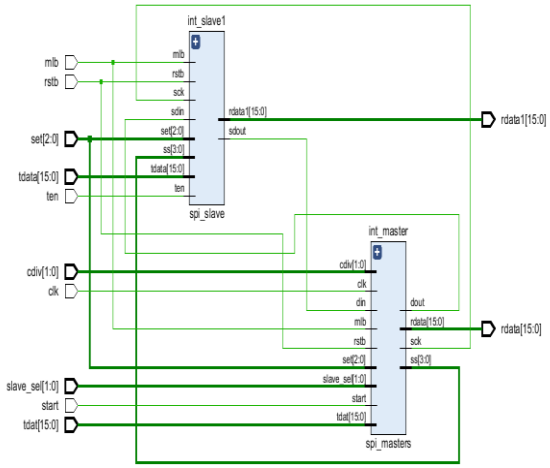


Fig.4: DUT (Design Under test).

2.1. Features of the current DUT: -

• With 16-bit shift register both Slave and Master module.
• The Receive buffer register (Rreg) with 16-bit capacity present in Master and Slave module.
• The Transmit data register (Treg) with 16-bit capacity present in Master and Slave module.
• Clock generator with up to 8-bit baud rate, can also can be extended up to 16 bit.
• Serial clock (SCK) pin.
• Master-Out-Slave-In (MOSI) pin.
• Master-In-Slave-Out (MISO) pin.
• Multiple Chip Select or Slave Select (SS) pins (4 pin mode only).
• SPI Clock Frequency ranging from 62MHz to 160MHz.
• Stream signal length (112 bits up to 256 bits), currently supporting 112 bits.
• Adjustable delay settings for even and odd burst case when paired with FIFO (First-In-First-Out).
• SPI transfer mode.
• Interrupt capability i.e., Master Done and Slave Done signals.
• With Up to 62 MHz operation, with max clock frequency being 133MHz for 16bit data transfer.

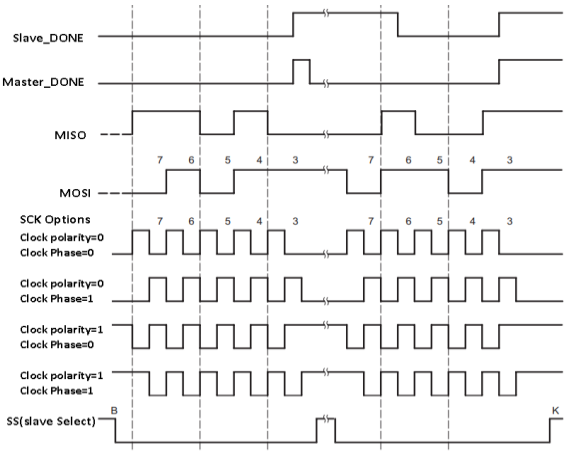


Fig.5: Signals involving data transmission.

2.2. Other DUT Operations: -

• Supports MOSI and MISO copying(boosting).

• Supports FIFO and SPRAM modules as Slaves with delay burst read setting(programmable).

• Instant Data reading mechanism using SPRAM module without resorting to Ideal State.

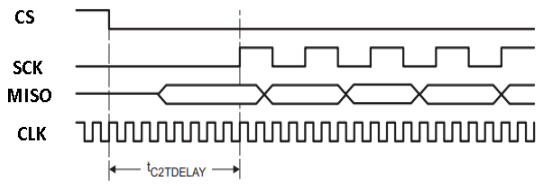


Fig.6: Chip-select-active-to-transmit-start delay.

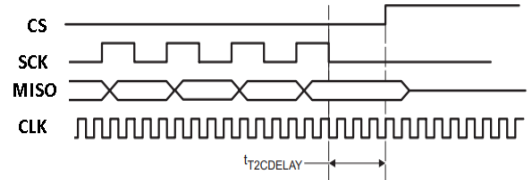


Fig.7: Transmit-end-to-chip-select-inactive delay.

• T2CDELAY is accessible and implied only in Master mode. As it instigates or indicates hold time to the Slave device or module, that delays the chip select deactivation process solely based on clock cycles after the last bit is transferred. T2CDELAY can be set up or configured between 2 to 16 SPI module clock cycle, currently the present DUT module supports 2, but can also be extended upto 8.

• C2TDELAY is accessible and implied only in Master mode. It instigates or indicates setup time for the Slave device or module, that delays the data transmission from the chip select active edge based on the transition edge of clock cycles. C2TDELAY can be configured between 2 to16 SPI module clock cycles, currently the present DUT module supports 16.

III. VERIFICATION METHODOLOGY

Verification involves test plan like a road map which provides an indication as to how to achieve the necessary goal for testing the DUT. The test plan provides, a blue print which includes introduction, assumptions, test cases to be run, different features that can be tested, what kind of approach to be taken. All this criterion helps the verification engineer to observe and understand how the verification process should be approached and done. The verification test plan can be expected in different methods or ways, such as document, simple text file or a spreadsheet. The descriptions to a Testbench architecture and description of each component and its functionality is a needed criterion for verification.

Systemverilog being an effective and potential hardware description language (HDL) provides good verification environment, it duly employs Constraint Random Generation (CRG), Assertion Based testing and Verification, also provides Coverage Driven Verification. These aspects provided by the systemverilog improvises the verification process gradually. The Feature of systemverilog is that it delivers enhanced hardware-modeling, which gradually and effectively improve the RTL design productivity and simplify the test process for the given DUT. Direct Programming Interface is a programming interface which is a part of Systemverilog that can be implemented to interface foreign languages with Systemverilog. Foreign languages that systemverilog supports can be C, C++, SystemC as well as others.

With the Universal Verification Methodology (UVM) on the other hand is one such methodologies apart from Systemverilog which was created for the need to automate DUT verification. UVM is an effective collection of API's and comes with a set of proven verification instructions or guidelines that is written for Systemverilog which can help the vilification engineers to develop an effective and efficient verification environment. It is an accessible open-source standard which is maintained by Accellera. Since the use of UVM methodology, engineers began to develop verification components that were significantly generic and which could be used from one project to another, this elevated the cooperation and sharing of methods and techniques among different verification users. It also greatly promoted and encouraged the expansion of verification components without modifying the original code.

3.1. Components of Systemverilog.

Transaction class: It defines the pin level activity created or generated by agent (having to drive

stimulus to the DUT through the driver) or the activity to be observed by the agent.

Generator class: In this class the stimulus is Generated (created through randomization) and then sent to the Driver.

Driver class: This stimulus is received in this class (transaction) which is passed from the generator and then pushes or drives the packet level data to the transaction into pin level (to DUT).

Monitor class: Pin level activity is observed on the interface signals through this class and then converts it into packet level signals which is then pushed or sent to the components or class such as scoreboard.

Agent class: This class contains other classes such as generator, driver, and monitor that is specific to protocol or Interface.

Scoreboard class: This class receives data items from monitor and performs comparison with expected values. However, the expected values are generated from the reference model also a copy can be taken from driver class.

Environment class: It is also a container class for grouping and containing the components such as agent and scoreboard.

Test program:

1. Configures testbench
2. Initiates the construction process of testbench components.
3. Initiates the process of stimulus driving.

Testbench Top class: The topmost module is the testbench top, where the DUT and Testbench is connected. This module consists of instances of DUT, interface classes and Test program, where the interface connects the DUT and Testbench.

3.2. Components of UVM.

Sequence-item: The class consist of variables (data) or inputs that are necessary for generating the desired stimulus. For the stimulus generation, the sequence-items are needed to be randomized in manner of sequences. Thus, the data variables defined in sequence-items must specifically be declared with rand keyword and can also contain constraints if necessary. Sequence-item in UVM is constructed through extending the uvm_sequence_item.

Sequence: A Sequence creates or develops a series of sequence_item's and pushes it to the driver through sequencer, Sequence process is performed by extending the uvm_sequence.

Sequencer: The Sequencer is an extended class of the uvm_sequencer that manages the flow of response between the sequence and the driver. TLM Interface is used by Sequencer and Driver to establish transaction communication.

Driver: Driver is constructed through extending the uvm_driver. TLM port (seq_item_port) must be addressed for interaction between sequencer and driver. Through Interface connection driver drives the data to DUT.

Monitor: Monitor class is constructed through extending the uvm_monitor class, it is a passive type component which perform the sampling of DUT signals at the Virtual interface level and transforms the activities at signal level to the transaction level activities. However, Monitor class does not drive DUT signals.

Agent: Agent is constructed through extending the class uvm_agent. The agent contains or groups the verification components like driver, monitor, collector and sequencer. It is used to connect the above mentioned components using TLM connections. The agent comes with one of the operating modes that is either active or passive sometimes both.

Environment: Environment class is constructed through extending the uvm_env class. This class groups and contains other classes like agents, scoreboard, top-level monitor.

Test: The Test is constructed through extending the uvm_test. It is the upper most class. The Test class being the top most class is in charge of Testbench construction, its configuration along with initiation of components involved in it.

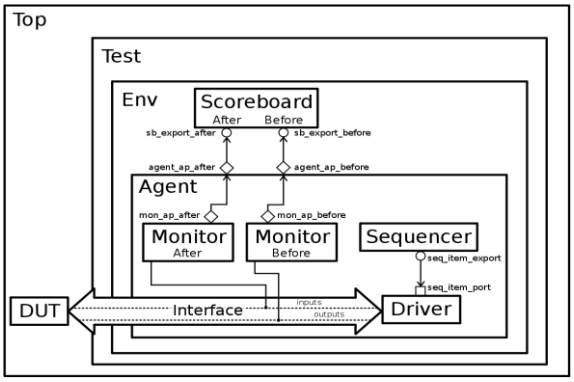


Fig.8: UVM Architecture.

IV. SIMULATION RESULTS

This section provides an insight about the simulation results of the present Design Under Test (DUT) i.e., Single Master Single Slave with Slave interface supporting many sub Salves. The DUT is interfaced with generator module, interface module along with driver and monitor modules in order to obtain effective performance in the constrained mode. After integration part, the simulation is performed using QuestaSim with mentor questa 2020.1 simulator tool. Also, simulation results for DUT verification using UVM methodology is

performed using DOULOS EDA-Playground tool with using synopsys VCS 2020.03 simulator along with mentor questa 2020.1 simulator tool.

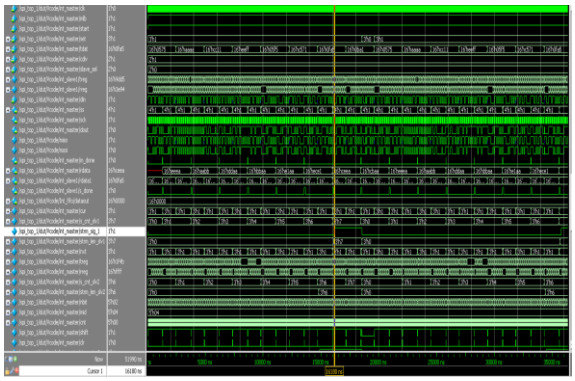


Fig.9: Master communicating with FIFO Slave.

The Figure 9 shows the DUT simulation with FIFO implemented with it. Here the data is stacked with write signal (wr) being high over a specific duration of time throughout the simulation. It is a two clock pulse frequency mode (cdiv=0). The stream signal is high for every seven times 16-bit data transmission indicating transfer of 112-bits. However, it can be extended up to 256-bits.

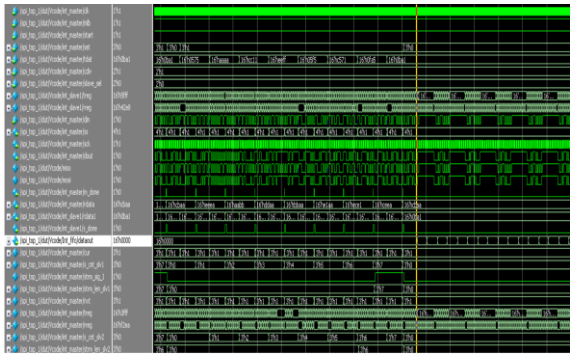


Fig.10: Master stream signal being high.

The above Figure 10, depicts or shows that the transmitted Slave data that was stacked is read by enabling read (rd) signal, which is enabled for a given specific period of time. Thus the data is obtained in the form of packet signals with avoiding to revert to ideal state.

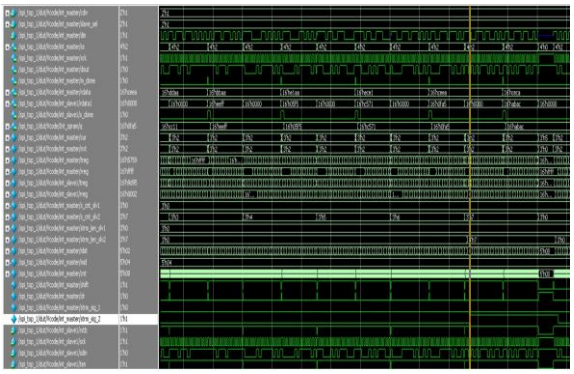


Fig.11: Master communication with SPRAM.

Figure 11 shows the simulation result of SPI interfaced with SPRAM (Single Port Random Access Memory) module operating at four pulse clock frequency mode (cdiv=1), here the RAM reads the data with one Master clock cycle delay without resorting to ideal state. This mechanism helps in faster Data read, also in this mechanism Stream signal is high for every seven times 16-bit data transmission indicating transfer of a total 112-bits. However, it can be extended up to 256-bits.

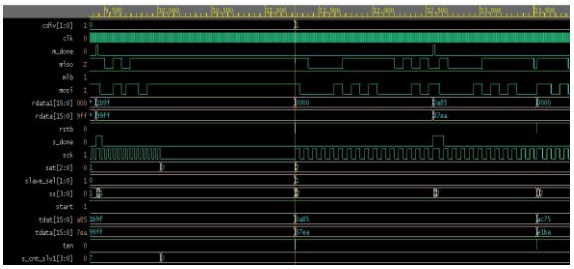


Fig.12: Transmission Delay.

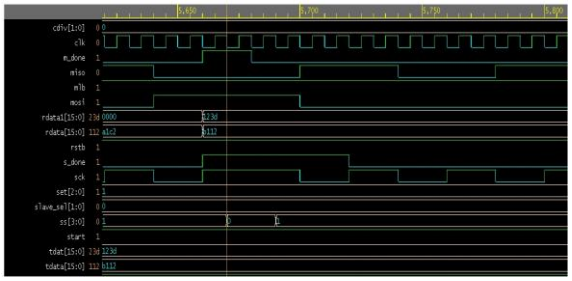
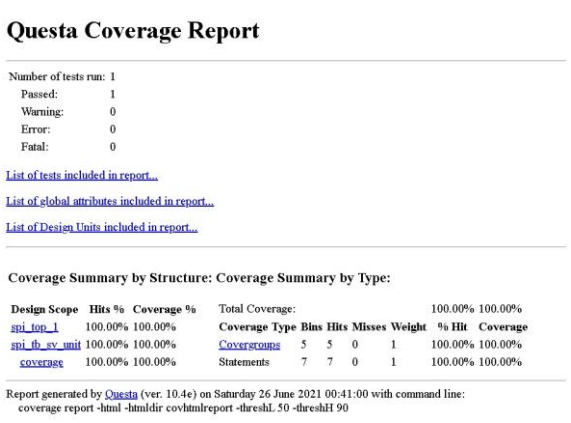


Fig.13: Slave Select Delay.

The Figures 12 and 13 depicts the transmission delay and Chip Select (CS) or Slave Select (SS) hold delay, where the transmission delay is around 16 delay pulses and Chip Select delay is around 2 pulses.



Questa Coverage Report

Number of tests run: 1
Passed: 1
Warning: 0
Error: 0
Fatal: 0

List of tests included in report...
List of global attributes included in report...
List of Design Units included in report...

Coverage Summary by Structure: Coverage Summary by Type:

Design Scope	Hits %	Coverage %
spi_top_1	100.00%	100.00%
spi_tb_sv_unit	100.00%	100.00%
coverage	100.00%	100.00%

Total Coverage: 100.00% 100.00%

Coverage Type	Bins	Hits	Misses	Weight	% Hit	Coverage
Covergroups	5	5	0	1	100.00%	100.00%
Statements	7	7	0	1	100.00%	100.00%

Report generated by Questa (ver. 10.4e) on Saturday 26 June 2021 00:41:00 with command line:
coverage report -html -htmldir covhtmlreport -threshL 50 -threshH 90

Fig.14: Coverage Report.

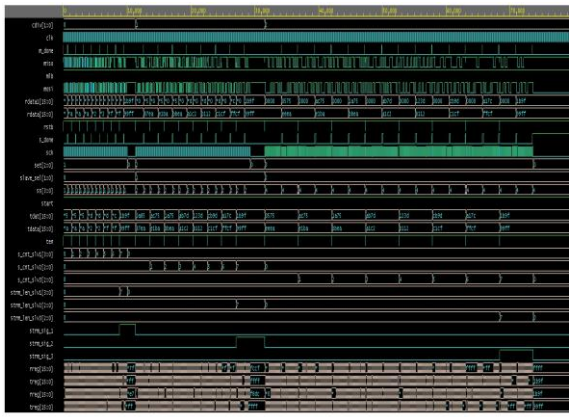


Fig.15: UVM Simulation result.

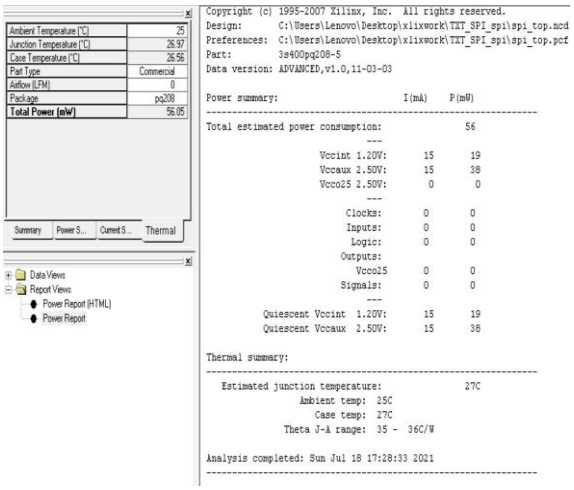
Above Figure 15 shows multiple Slaves (3 slaves) operating at different Master frequency with clock frequency (cdiv) kept at cdiv=0, cdiv=1, cdiv=2.

```
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 8.911ns (Maximum Frequency: 112.222MHz)
   Minimum input arrival time before clock: 8.117ns
   Maximum output required time after clock: 6.216ns
   Maximum combinational path delay: No path found

Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 7.500ns (frequency: 133.333MHz)
  Total number of paths / destination ports: 128 / 15
```

Fig.16: Timing Summary.



```
Copyright (c) 1995-2007 Xilinx, Inc.  All rights reserved.
Design:       C:\Users\Lenovo\Desktop\xlixwork\TXT_SPI_spi\spi_top.ncd
Preferences:  C:\Users\Lenovo\Desktop\xlixwork\TXT_SPI_spi\spi_top.pcf
Part:         3s400pq208-5
Data version: ADVANCED,v1.0,11-03-03

Power summary:                          I(mA)    P(mW)
---------------------------------------------------------------------
Total estimated power consumption:                 56
                                        ---
                        Vccint 1.20V:       15       19
                        Vccaux 2.50V:       15       38
                        Vcco25 2.50V:        0        0
                                        ---
                              Clocks:       0        0
                              Inputs:       0        0
                               Logic:       0        0
                             Outputs:
                               Vcco25       0        0
                             Signals:       0        0
                                        ---
                Quiescent Vccint 1.20V:       15       19
                Quiescent Vccaux 2.50V:       15       38

Thermal summary:
---------------------------------------------------------------------
   Estimated junction temperature:                 27C
                        Ambient temp:  25C
                           Case temp:  27C
                      Theta J-A range:  35 -  36C/W

Analysis completed: Sun Jul 18 17:28:33 2021
---------------------------------------------------------------------
```

Fig.17: Power report.

Figure 16 and 17 shows the timing report and power consumption of the DUT.

V. CONCLUSION

In this paper, a Systemverilog based UVM environment is developed for SPI protocol. The test bench is able to verify and validate the operation of full duplex serial data transfer between the single Master and multiple sub Slave modules with different clock frequencies with fixed stream signal length and also coverage report is generated. However, the present design supports 16-bit data transfer, it also can be extended to 32-bit with minor adjustments in delay aspect and the stream signal length can also be increased as it currently supports 112-bit transfer. On an overall note a reusable SPI protocol is designed and verified of its functionality using Systemverilog and UVM (Universal Verification Methodology).

REFERNCE

[1]. P. Rajashekar Reddy and P.Sreekanth, Assistant Professors of CVR College of Engineering College, ECE, Hyderabad, "*Serial Peripheral Interface-Master Universal Verification Component using UVM*", International Journal of Advanced Technologies in Engineering and Management Sciences (IJASTEMS-ISSN: 2454-356X) *Volume.3,Issue.6,June.2017.*

[2]. Ananthula Srinivas, M.Kiran Kumar , Jugal Kishore Bhandari "*Design and Verification of Serial Peripheral Interface*", *IJEDR, ISSN: 2321-9939.*

[3]. Pallavi Polsani, V. Priyanka B and Y. Padma Sai "*Design & Verification of Serial Peripheral Interface (SPI) Protocol*", International Journal of Recent Technology

and Engineering (IJRTE) *ISSN: 2277-3878, Volume-8 Issue-6, March 2020*.

[4]. Punith Kumar M B1 , Sreekantesha H N *"Design and Verification of Serial Peripheral Interface Master Core Using Universal Verification Methodology"*, International Journal of Computer Sciences and Engineering, *Vol.-7, Special Issue-14, May 2019 E-ISSN: 2347-2693.*

[5]. W.Ni and J.Zhang, "*Research of reusability based on UVM verification*," in 2015 IEEE 11th International Conference on ASIC, Nov 2015, pp. 1-4.

[6] Zhili Zhou, Zheng Xie, Xin‟an Wang and Teng Wang, "*Development of verification Environment for SPI Master Interface Using Systemverilog*", 978-1-4673-2197-6/12/$31.00 ©2012 IEEE.

[7]. "*IP Design of Universal Multiple Devices SPI Interface*" Tianxiang Liu1, Yunfeng Wang1 * Department of Electronic Engineering, Xiamen University, 2011 IEEE.

[8]. A.K. Swain and K. Mahapatra, "*Design and Verification of WISHBONE bus interface for System-On-Chip integration*," in india Conference (IINDICON),2010 Annual IEEE. IEEE,2010,pp. 1-4.

CHETAN N, et. al. "Verification of SPI protocol Single Master Multiple Slaves using Systemverilog and Universal Verification Methodology (UVM)." *International Journal of Engineering Research and Applications (IJERA)*, vol.11 (7), 2021, pp 01-08.