

Design and verification of daisy chain serial peripheral interface using system Verilog and universal verification methodology

Rajesh Thumma, Pilli Prashanth

Department of Electronics and Communication Engineering, Anurag University, Hyderabad, India

Article Info

Article history:

Received Aug 30, 2021

Revised Nov 12, 2022

Accepted Nov 22, 2022

Keywords:

Daisy-chain

I2C

SV

UVM

ABSTRACT

Serial peripheral interface (SPI) transfers the data between electronic devices like micro controllers and other peripherals. SPI consists of two control lines: select signal and clock signal, and two data lines: input and output. In single master-single slave, the communication is in between master and slave only which will make the design complex and costly, area will increase. In regular SPI mode, the number of chip-select lines is increased if the number of slaves increases. Due to this, the input data received by the master from the slaves are corrupted at master input slave output (MISO). The proposed daisy chain method is used to overcome this problem. The daisy chain method requires only one chip select line at master compared to the regular SPI mode. When the chip-select line is active low, all the slaves are active, and the clock is initiated to all the slaves to transfer the data from the master to the first slave through the master output slave input (MOSI). In this paper, the daisy-chain SPI is designed and developed using Verilog. The proposed design is verified using system Verilog (SV) and universal verification methodology (UVM) in QuestaSim.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Rajesh Thumma

Department of Electronics and Communication Engineering, Anurag University

Venkatapur, Narapally, Hyderabad, India

Email: rajesh.thumma88@gmail.com

1. INTRODUCTION

System on chip (SoC) architecture requires different components to develop an application. For the communication and operation between these components, Interfaces are utilized. The communication speed of the SoC depends on the types of interfaces. The speed rate of the serial peripheral interface (SPI) interface is 1.1 Mbps. The interfaces are classified into two types, based on the data transmission. They are a serial interface and a parallel interface.

According to the SoC component's protocols, serial or parallel communication interfaces are used [1]-[17]. The motorola semiconductors are the first developers of SPI. The SPI and inter-integrated circuit (I2C) protocols are used [2] to transfer the data in sequential communication. These two protocols are appropriate for interchanges between coordinated circuits and with onboard peripherals. The inter-integrated circuit (I2C) transport utilizes two signals, a sequential clock signal (SCL) and a sequential information signal SDA, to move information among numerous devices. When contrasted with I2C, SPI utilizes four signals to move among various devices [3]-[25]. For intra chip communication, SPI is usually used. Both the master and slave perform the dual role of transmitter and receiver in the SPI. The SPI master slave is designed from the initial specifications to final system verification by using Verilog hardware description language (HDL) and achieved 71 to 75 megabytes second by implementing in Virtex-5 field programmable gate array (FPGA) [4]-[24]. Various SPI design techniques are proposed and compared [5] their implementation with respect to chip selects lines.

However, the number of chip-select lines are increased in the SPI conventional method, while in the daisy chain technique, the number of chip-select lines is reduced in implementation and easy to design [5]-[21]. The master-slave communication protocol is designed by assigning the priority to each slave by communicating with the master based on the highest priority slave [6]-[23]. However, the design will consume less power and utilization sources compared with the other complex design. The high-speed SPI [7] bus is designed in vertex5 to control and handle two slaves at a time and compared with the existing architectures. However, accessing multiple devices using a master-slave will be overcome by applying the standard Serial peripheral interface [8] and single/master communication protocol. The SPI has been designed with five 32-bit registers using a compatible wishbone interface [9], [10] for serial synchronous communication. In this, 100% of functional code coverage achieved with up to 64-bits of full-duplex communication is verified. The interfacing and monitoring of battery-operated electric vehicles [11], [12] is designed using complementary metal-oxide-semiconductor (CMOS) to transfer the data rate upto 1 Mbps. System Verilog is the most promising language to reduce the system-on-a-chip (SOC) verification and reusable components of the complex SOC design. using system Verilog [21], various components are designed, implemented with object-oriented programming [13], [14] and applied a random technique to find functional coverage. However, universal verification methodology [15], [17], [18] will reduce the complexity, time, and rewriting code by accessing the inbuilt classes. Using universal verification methodology (UVM), the SPI master-slave is designed and verified the 100% functional coverage and code coverage [19], [20].

The SPI is the most used in various interfacing circuits like analog to digital converters (ADCs), static random access memory (SRAM), sensors, digital to analog converters (DACs), shift registers and others. SPI is a master, slave-based synchronous, full-duplex interface [5]-[17]. The data is synchronized from the master or slave at the falling or rising edge of the clock [22]. Both slave and master can send information (data) simultaneously. This article proposes a daisy chain technique to design an SPI Master-slave interface using Verilog and verified with system Verilog and UVM. The simulation verification is performed in a model sim and QuestaSim. The simulation results are obtained in Xilinx Vivado, and both the verification methods covered 100% of functional coverage, code coverage. In section 2 covers basic information of the SPI theory, daisy chain method and operation. Section 3 covers designing a daisy chain SPI using Verilog and is verified in system Verilog and UVM. Section 4 covers the simulation results of the proposed design Xilinx and QuestaSim.

2. SERIAL PERIPHERAL INTERFACE

SPI the data transfer between the master and slave devices are depends on the control signals and data signals of the serial peripheral interface. There are two types of control signals, namely slave select (SS) or chip select (CS), a clock signal (SCLK). The master output slave input (MOSI), master input slave output (MISO) are two data signals. When the chip select line is active low its selects, the respected slave and the data will read or write based on the clock polarity (CPOL) and clock phase (CPHA). The clock signal reads the data addresses the clock pulses and writes when the chip select signal is high. The data is transferred from master to slave and slave to the master through MOSI and MISO signals. There are four modes to transfer the data with clock polarity and clock phase. Table 1 represents the SPI Modes with clock polarity.

Daisy chain SPI based on the number of slaves, the serial peripheral interface is classified into a single master-single slave, single master-multiple slaves. In single master-single slave, the communication is in between master and slave only. By using single master-single slave, the area will increase by increasing the master-slaves, and the design will be complex, making the increase in cost and the area. So, the single master-single slave is not preferable, and single master-multiple slaves are used in most of cases. The single master-multiple slaves are further classified into regular SPI method and daisy chain method.

Table 1. SPI modes with CPOL and CPHA

SPI mode	CPOL	CPHA	Clock polarity in idle state	Clock phase used to sample and/or shift the data
0	0	0	Logic low	On the rising edge, data was sampled, and on the falling edge, it was shifted out
1	0	1	Logic low	On the lowering edge, data was collected, and on the rising edge, it was shifted out
2	1	1	Logic high	On the lowering edge, data was collected, and on the rising edge, it was shifted out
3	1	0	Logic high	On the rising edge, data was sampled, and on the falling edge, it was shifted out

In regular SPI mode, the number of chip-select lines is increased if the number of slaves increases. Figure 1 shows the regular SPI mode of operation. Due to this, the input data received by the master from the slaves are corrupted at MISO. Because of this daisy chain method is most preferable to overcome this problem. The daisy chain method requires only one chip select line at master compared to the regular SPI mode. When the chip-select line is active low, all the slaves are active, and the clock is initiated to all the slaves to transfer the data from the master to the first slave through the MOSI.

The first slave's output is shared with a second slave, the second slave with the third, and the last slave output is shared with the master; this forms a daisy chain configuration. The primary serial peripheral interface with a single master multiple slave configuration shows in Figure 1. The proposed daisy chain SPI configuration shows in Figure 2.

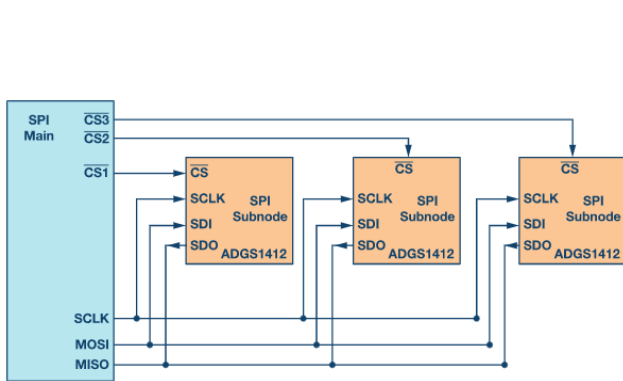


Figure 1. SPI single master-multiple slave configuration [26]

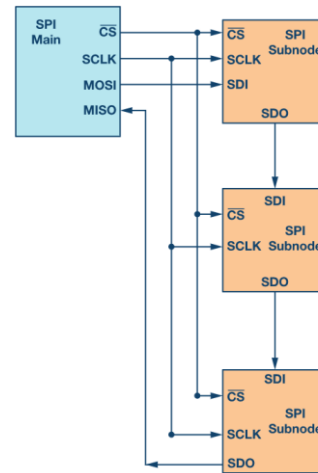


Figure 2. A daisy chain SPI: single master-multiple slave configuration [26]

3. VERIFICATION

The daisy chain SPI is designed with Verilog and verified using the system Verilog and universal verification methodology. The flow chart of the verification methodology is shown in Figure 3. The Verilog code is compiled and finds the zero errors which create the elaborated design. After simulation of the design, the following are observed by taking the register-transfer level (RTL) analysis: RTL schematic, physical design, elaborated design, and power report. To implement in FPGA, the constraint files are written for generating a bitstream and dumped in the FPGA board.

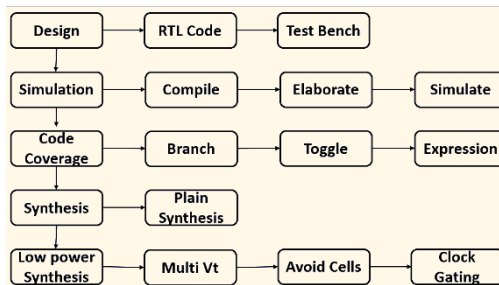


Figure 3. Flow chart of the verification methodology [1]

3.1. System Verilog

The daisy chain verification environment is implemented with the system Verilog components. The input and outputs are instantiated in class_packet. The class_packet is included in the generator and initiated naming as packet 1. A mailbox is used between the generator and the driver to transfer the generator's data to the driver. The generator is included in the driver, and a virtual interface is provided to access the inputs from the interface in a test. A mailbox driver can transfer the data between the driver to the scoreboard and the receiver to the scoreboard.

The data received from the driver and receiver will compare on the scoreboard and send to display. The class_driver is added in the receiver; design under test (DUT) output values are added to packet 2 using a virtual interface. The packet 1 data from the driver to the scoreboard and the packet 2 data from the receiver to the scoreboard are compared and included in the coverage. Different random values are added to the inputs using coverage groups, and coverage is added to the environment.

The functions of generator, driver, receiver, scoreboard and coverage are instantiated in the environment. The environment is included in the test to perform the test for the environment. A daisy-chain SPI DUT is developed with a single master and two slaves. A detailed code operation is written for master, slave 1, slave 2 and included in the DUT. Input and outputs are instantiated in the interface to access virtually. The test, DUT and Interfaces are included in the top block. The top block generates the clock, reset, start signals. The system Verilog verification environment is shown in Figure 4.

3.2. Universal verification methodology

A rich set of standard rules and guidelines systematically doing the things is called methodology. It provides the necessary infrastructure to build a robust, reliable and complex verification environment. It contains a base class library set, which we can use to build our test benches. A methodology should support coverage driven verification (CDV), transaction based verification (TBV), assertion based verification (ABV), constrained random testing (CRT). There are various verification methodologies, advanced verification methodology (AVM) developed by mentor graphics using system C and system Verilog. Reference verification methodology (RVM) is developed by synopsys using open vera. Open verification methodology (OVM) developed by mentor graphics using system Verilog. Verification methodology manual (VMM) by synopsys using system Verilog.

A technical subcommittee of acellera voted to establish the UVM and decided to build this new standard using the open verification methodology as its foundation. UVM is derived mainly from the OVM. Advantages of UVM are Common test bench structure and run flow. Reusability through test bench, Time required to build test bench is very less. It avoids poor coding practices, and Debugging is simple. The complete UVM verification environment is shown in Figure 5.

To understand the UVM, it is required to understand the verification environment of system Verilog. The architectures of system Verilog and UVM are similar, but generators are replaced with sequencers and agents are introduced in UVM. At this moment, a design under test (DUT) is used. To test the functionality of the DUT, an environment is required to connect the DUT. For this, a sequencer block is used to generate sequences of bits to transmit into the DUT. Generally, sequencers are responsible for generating data sequences, and they pass the data to another block called a driver. Why because the sequencer is unaware of the communication bus. So, the sequencer transmits the data to the driver. Now, the driver starts communicating with the DUT and by feeding the received data from the sequencer. A monitor block is used to communicate between the driver and the DUT to evaluate the DUT's responses. Monitors try to predict the expected result by sampling the inputs and the outputs of the DUT. They send the prediction and result of the DUT to the block called the scoreboard. The predicted data are compared and evaluated in the scoreboard.

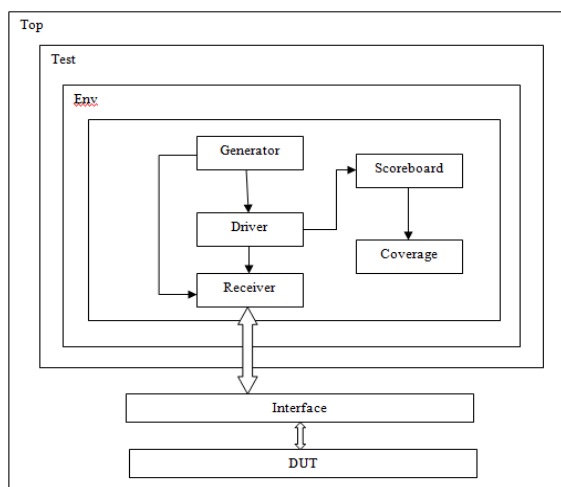


Figure 4. System Verilog verification architecture [9]

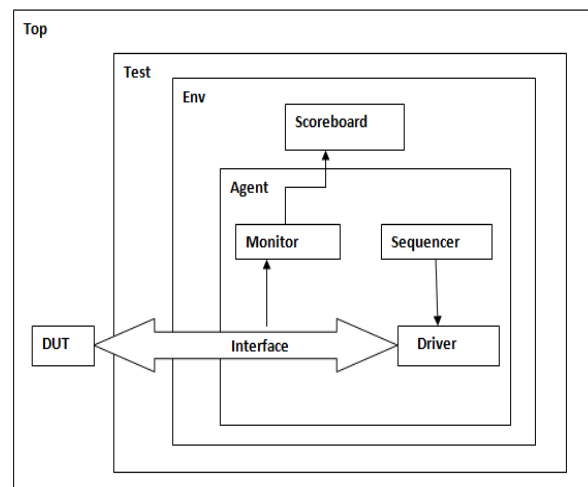


Figure 5. Verification environment of universal verification methodology [3]

A typical system is formed with all these blocks to perform the verification. The UVM test benches use the same structure [13]. Figure 5 represents the verification environment of the UVM. A test block is used generally to control all the blocks of the UVM. The Top block controls all the blocks and sub-blocks of the test bench. This implies that just by changing a couple of code lines, it is possible to add, eliminate and abrogate blocks in the test bench and construct various conditions without reworking the entire test. To delineate the benefit of this verification, need to add a monitor and a driver to the Verification, the environment changes to the I2C from the serial communication SPI and vice versa.

The serial peripheral interface is widely used due to its advantages. In single master–single slave SPI communication, there is no difficulty at master input slave output (MISO) to transfer the data compared to the single master–multiple slave SPI. To solve the above-mentioned problem, different design techniques are proposed by the different authors. They are interrupt enabled priority-based SPI, parameterization method using time-sharing multiples technique, high-speed SPI and wishbone compliant SPI. Table 2 represents the comparison of design techniques and their design or verification languages.

Table 2. Comparison of various SPI design techniques

SPI design techniques	Single master- single slave SPI	Single master- multiple slaves SPI	Verilog	SV	SV coverage	UVM	UVM coverage
Proposed design	-	Yes	Yes	Yes	Yes	Yes	Yes
Wishbone compliant SPI [10]	Yes	-	Yes	-	-	-	-
High speed SPI [7]	-	Yes	Yes	-	-	-	-
Interrupt enabled priority based SPI [6]	-	Yes	Yes	-	-	-	-
Parameterization method, time sharing multiplex (TSM) [8]	-	Yes	Yes	-	-	-	-
SPI master interface using System Verilog [13]	Yes	-	-	Yes	Yes	-	-
SPI master slave core [9]	Yes	-	Yes	Yes	Yes	-	-
SPI master slave core using UVM [3]	Yes	-	Yes	-	-	Yes	Yes

Table 2 compares various SPI design techniques designed using the Verilog, system Verilog (SV), and UVM. All the SPI design techniques are designed with the Verilog, but few of them are designed with advanced verification methodologies like SV and UVM. SPI master interface using system Verilog [13], SPI master-slave core [9] are verified using SV and SPI master-slave core using UVM [3] is verified in UVM achieves the 100% code coverage. The proposed daisy-chain SPI is designed using Verilog HDL, and a verification environment is implemented using SV and UVM to achieve 100% code coverage.

4. SIMULATION AND RESULTS

The daisy chain Serial Peripheral interface is designed and verified using various VLSI tools like Xilinx Vivado 2015.2, ModelSim, Xilinx integrated synthesis environment (ISE), QuestaSim. Single master-two slaves are configured in the daisy-chain. The following three inputs are applied in this design; they are $din = 10110110$, $din1 = 11001101$, $din2 = 10010011$ are given to the *master*, *slave1*, *slave2*, respectively. The clock, reset, start and chip select lines control the proposed project's entire design. The output ports are instantiated as *dout*, *dout1* and *dout2* ports and these ports are used to transfer the data between the master and slaves shift registers. A Verilog code is written and compiled in Xilinx Vivado 2015.2, and a detailed RTL code, *test_bench*, are designed with Verilog HDL. By successfully running the behavioural simulation with 0 errors and 0 warnings, the following simulation results are shown in Figure 6.

The data outputs $din = dout1$, $din1 = dout2$, and $din2 = dout$ define that the data flow is transferred in a daisy-chain fashion. The complete schematic diagram of daisy-chain SPI is obtained by applying an elaborated design is shown in Figure 7, and Figure 8 shows the elaborated circuit design cells. After adding the constraints to the xdc file, the design is synthesized. The synthesized device, synthesized schematic of top-level and circuit level are shown in Figure 9, Figure 10 and Figure 11. After generating the bitstream, it is dumped into the FPGA board to verify the functionality.

The design is verified in the verification environment using the system Verilog by using QuestaSim 10.0b tool. The inputs from the driver $d_in:1$, $d_in_1:10$, $d_in_2:11$, and outputs from the receiver $dout:11$, $dout1:1$, $dout2:10$ are compared at scoreboard. After comparison of the data received from the driver and receiver in scoreboard has matched. The daisy chain SPI simulation results of the system Verilog is shown in Figure 12.

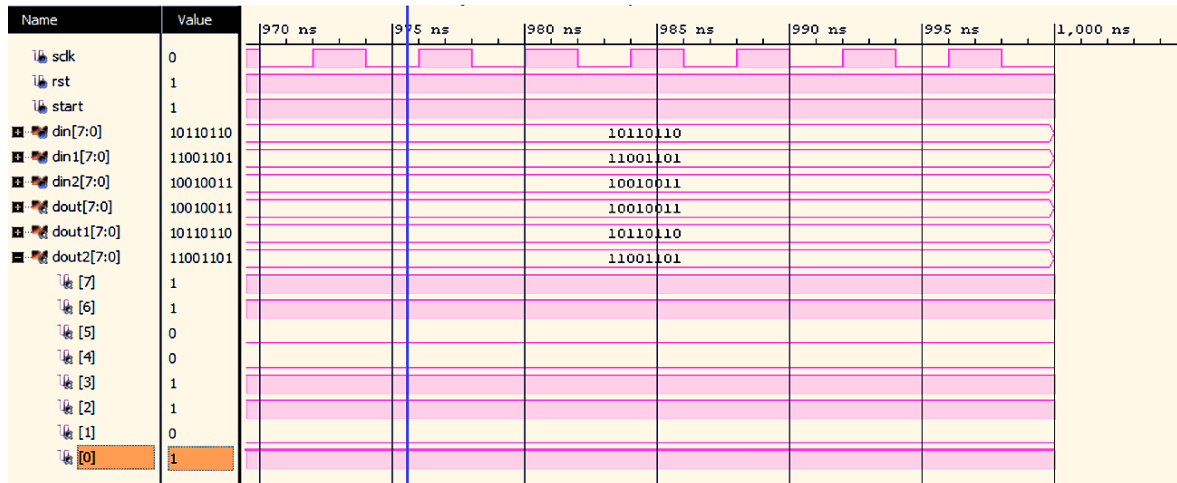


Figure 6. Simulation results of daisy-chain SPI

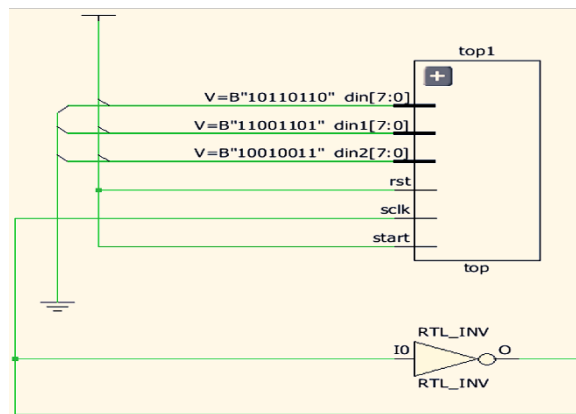


Figure 7. Elaborated schematic diagram

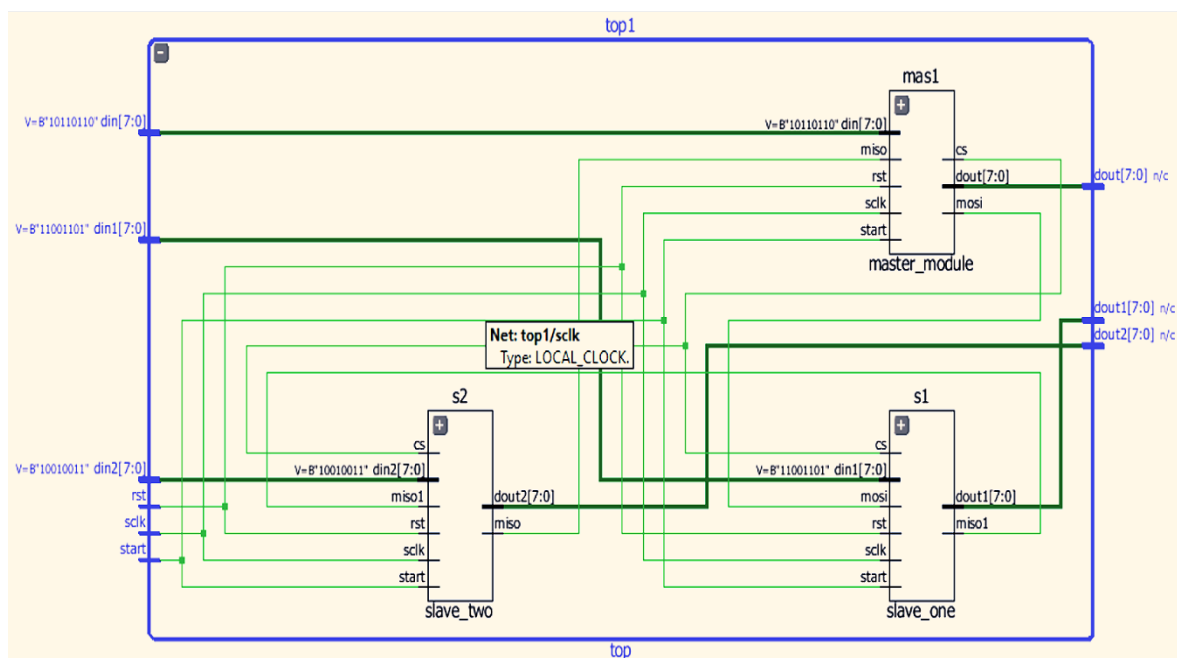


Figure 8. Elaborated circuit design

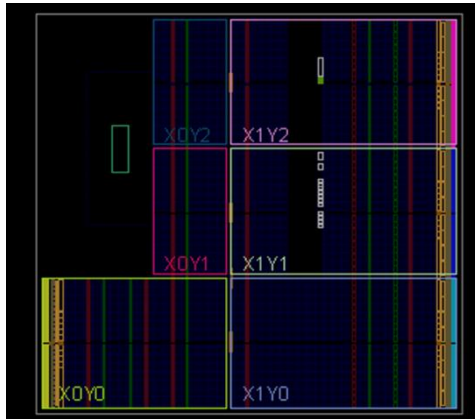


Figure 9. Synthesized device diagram

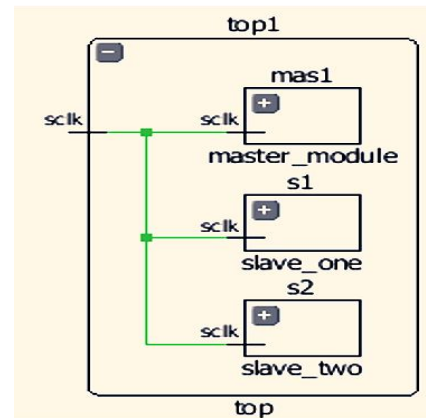


Figure 10. Synthesized design showing top level schematic

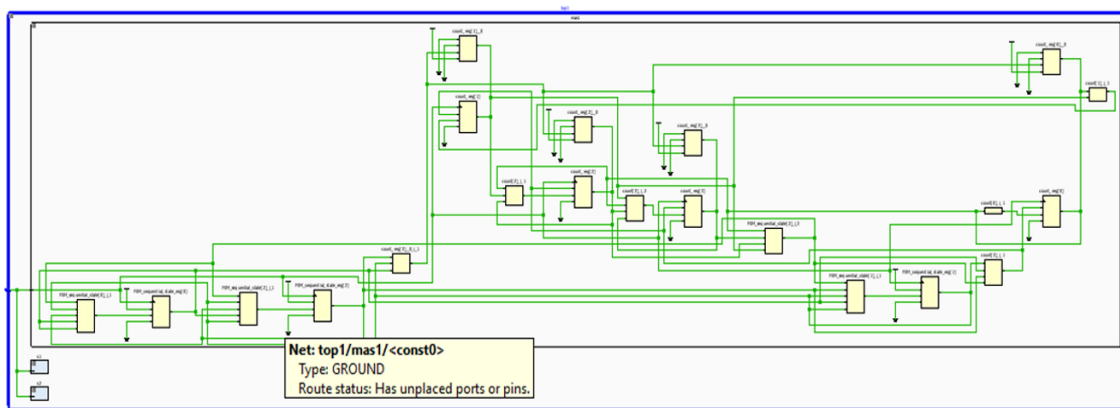


Figure 11: synthesized design schematic showing internal cells

```

# vsim work.top
# ** Note: (vsim-3812) Design is being optimized...
# Loading sv_std.std
# Loading work.mod_tcp_sv_unit
# Loading work.top(fast)
# Loading work.interl(fast)
# Loading work.slave_one(fast)
# Loading work.slave_two(fast)
# Loading work.stest_sv_unit
# Loading work.test(fast)
VSIM 22> run
# ** Warning: (vsim-8474) class_coverage.sv(7): A higher value '256' is found in bin 'd_in' of Coverpoint 'A'. It is invalid and will be ignored.
# Time: 0ns Iteration:3 Instance:/top/tl
# ** Warning: (vsim-8474) class_coverage.sv(8): A higher value '256' is found in bin 'd_in_1' of Coverpoint 'B'. It is invalid and will be ignored.
# Time: 0ns Iteration:3 Instance:/top/tl
# ** Warning: (vsim-8474) class_coverage.sv(9): A higher value '999' is found in bin 'd_in_1' of Coverpoint 'C'. It is invalid and will be ignored.
# Time: 0ns Iteration:3 Instance:/top/tl
# ** Warning: (vsim-8474) clas3_coverage.sv(9): A higher value '256' is found in bin 'd_in_1' of Coverpoint 'C'. It is invalid and will be ignored.
# Time: 0ns Iteration:3 Instance:/top/tl
# generator: '{d_in:1, d_in_1:10, d_in_2:11, dout:0, dout1:0, dout2:0}'
# driver pkt: '{d_in:1, d_in_1:10, d_in_2:11, dout:0, dout1:0, dout2:0}'
# riciver pkt: '{d_in:0, d_in_1:0, d_in_2:0, dout:11, dout1:1, dout2:10}'
# scorebordmatch
# generator: '{d_in:1 d_in_1:10, d_in_2:11, dout:0, dout1:0, dout2:0}'
# driver pkt: '{d_in:1, d_in_1:10, d_in_2:11, dout:0, dout1:0, dout2:0}'
# riciver pkt: '{d_in:0, d_in_1:0, d_in_2:0, dout:11, dout1:1, dout2:10}'
# scorebordmatch
# generator: '{d_in:1, d_in_1: 10 , d_in_2:11, dout:0, dout1:0, dout2:0}'
# driver pkt: '{d_in:1, d_in_1:10, d_in_2:11, dout:0, dout1:0, dout2:0}'
VSIM 23> run
# riciver pkt: '{d_in:0, d_in_l:0, d_in_2:0, dout:11, dout1:1, dout2:10}'
# scorebordmatch
# Simulation stop requested.
VSTM 24> 1

```

Figure 12: Output results of System Verilog using QuestaSim 10.0b

Coverage groups are developed for inputs d_in , d_in_1 , d_in_2 with random values, and a 100% functional coverage report is achieved. Figure 13 represents the coverage report of system Verilog using QuestaSim 10.0b. The design is verified in universal verification methodology by developing a verification environment.

In this, the inputs from monitor 1, outputs from monitor two are compared at the scoreboard. Here the inputs are consider as $d_in = 182$, $d_in_1 = 138$, $d_in_2 = 7$, the outputs from the monitor 2 are $dout = 7$, $dout1 = 182$, $dout2 = 138$ are matched at the scoreboard according to the daisy-chain method. Figure 14 represents the simulation and summary report of the UVM using QuestaSim 10.0b.

The coverage A, coverage B, coverage C are developed for the inputs with random values and achieved a 100% functional coverage and code coverage. Figure 15 represents the coverage report of the UVM. The functional and code coverage reports are executed using QuestaSim.

Name	Coverage	Goal	% of Goal	Status	Merge_instances	Get_inst_coverage	Comment
/stest_sv_unit/cove...							
TYPE cvg	100.0%	100	100.0%		0		
CVP cvg::A	100.0%	100	100.0%				
CVP cvg::B	100.0%	100	100.0%				
CVP cvg::C	100.0%	100	100.0%				

Figure 13: System Verilog coverage reports using QuestaSim 10.0b

```
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :14
# UVM_WARNING : 0
# UVM_ERROR : 0
# UVM_FATAL: 0
# ** Report counts by id
# [DRV] 2
# [MON1] 3
# [MON2] 3
# [RNTST] 1
# [SB] 2
# [SEQ] 2
# [TEST_DONE] 1
# ** Note: $finish : C:/questasim_10.0b/win32/./verilog_src/uvm-1.0pl/src/base/uvm_root.svh(392)
# Time: 99 ns Iteration: 44 Instance: /top
# 1
# Break at C:/questasim_10.0b/win32/./verilog_src/uvm-1.0pl/src/base/uvm_root.svh line 392
VSIM 4>
```

Figure 14: Universal verification methodology output results using QuestaSim 10.0b

Name	Coverage	Goal	% of Goal	Status	Merge_instances	Get_inst_coverage	Comment
/TOP_sv_unit/spi_c...							
TYPE cvg	100.0%	100	100.0%		0		
CVP cvg::A	100.0%	100	100.0%				
CVP cvg::B	100.0%	100	100.0%				
CVP cvg::C	100.0%	100	100.0%				

Figure 15: Coverage reports of UVM using QuestaSim 10.0b

5. CONCLUSION

In this paper, the daisy-chain SPI is designed using Verilog. The developed design is verified using SV and UVM. The open-source design suite tool, ModelSim personal edition, was used to write the Verilog code, which gives the simulation results. The verification environment of the SV and UVM is developed using the QuestaSim tool. ModelSim and QuestaSim are Mentor Graphics products, which are used to implement the necessary functional registers. Universal verification methodology verifies the design in the most effective way. The daisy chain SPI functionality, operation, depiction of registers, pin and signals are discussed. Functional verification contains the verification platform's description using system Verilog for the design under the daisy-chain SPI test. The created verification environment validates the functionality and operation of configurable daisy-chain SPI. The verification environment developed for daisy-chain SPI protocol was reusable and using which design can be verified successfully. by using this verification environment, we can achieve 100% functional and assertion coverage. The designed daisy chain SPI from Verilog is implemented in FPGA.





REFERENCES

- [1] D. Roopesh and K. Siddesha, "RTL design and verification of SPI master-slave using UVM," *Computer Science*, vol. 4, no. 8. [Online]. Available: <https://www.semanticscholar.org/paper/RTL-DESIGN-AND-VERIFICATION-OF-SPI-MASTER-SLAVE-UVM-Roopesh-Siddesha/9db0eb90f17859812d50a677949ae2eff1730a09>
- [2] A. K. Shah, "High Speed SPI Slave Implementation in FPGA using Verilog HDL," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 4, no. 12, pp. 4365–4369, 2015. [Online] Available: <https://pdf4pro.com/amp/view/high-speed-spi-slave-implementation-in-fpga-using-1c923d.html>
- [3] K. V. A. Kumar and M. S. Krishna, "Design and Functional Verification of A SPI Master Slave Core using UVM," *International Journal of Scientific Engineering and Technology Research (IJSETR)*, vol. 4, no. 51, pp. 11023–11030, 2015. [Online]. Available: <http://ijsetr.com/uploads/423651IJSETR8063-1902.pdf>
- [4] A. K. Oudjida, M. L. Berrandjia, A. Liacha, R. Tiar, K. Tahraoui and Y. N. Alhoumays, "Design and test of general-purpose SPI Master/Slave IPs on OPB bus," *2010 7th International Multi- Conference on Systems, Signals and Devices*, 2010, pp. 1-6, doi: 10.1109/SSD.2010.5585592.
- [5] V. K. Verma, "Comparative Study of SPI Design Systems," *International Journal of Electrical, Electronics and Data Communication (IJEEEDC)*, vol. 7, no. 10, pp. 4-5, 2019. [Online]. Available: http://www.iraj.in/journal/journal_file/journal_pdf/1-605-15767465564-5.pdf
- [6] Deepika and J. K. Murthy, "Interrupt Enabled Priority Based Master Slave Communication using SPI Protocol," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 9, no. 9, pp. 564–567, 2020, doi: 10.35940/ijitee.i7649.079920.
- [7] Anand N, G. Joseph, S. S. Oommen, and R. Dhanabal, "Design and implementation of a high speed Serial Peripheral Interface," *2014 International Conference on Advances in Electrical Engineering (ICAEE)*, 2014, pp. 1-3, doi: 10.1109/ICAEE.2014.6838431.
- [8] T. Liu and Y. Wang, "IP design of universal multiple devices SPI interface," *2011 IEEE International Conference on Anti-Counterfeiting, Security and Identification*, 2011, pp. 169-172, doi: 10.1109/ASID.2011.5967443.
- [9] K. Aditya, M. Sivakumar, F. Noorbasha, and T. P. Blessington, "Design and Functional Verification of A SPI Master Slave Core Using System Verilog," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 2, pp. 2231–2307, 2012. [Online]. Available: <https://silo.tips/download/design-and-functional-verification-of-a-spi-master-slave-core-using-system-veril>
- [10] Purushottam S. and Naveenkumar M., "Design and Verification of wishbone Compliant Serial Peripheral Interface," *International Journal of Engineering Research & Technology (IJERT)*, *NCESC - 2018 Conference Proceedings*, 2018, vol. 6, no. 13, pp. 1-4. [Online]. Available: <https://www.ijert.org/research/design-and-verification-of-wishbone-compliant-serial-peripheral-interface-IJERTCONV6IS13178.pdf>
- [11] X. Wang, H. Zhang, L. Zhang, J. Zhang and Y. Hao, "A daisy-chain SPI interface in a battery voltage monitoring IC for electric vehicles," *2014 12th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, 2014, pp. 1-3, doi: 10.1109/ICSICT.2014.7021462.
- [12] Q. Zhang, Y. Yang, and C. Chai, "A high EMS daisy-chain SPI interface for battery monitor system," *Journal of Semiconductors*, vol. 38, no. 3, 2017, doi: 10.1088/1674-4926/38/3/035002.
- [13] Z. Zhou, Z. Xie, X. Wang, and T. Wang, "Development of verification environment for SPI master interface using SystemVerilog," *2012 IEEE 11th International Conference on Signal Processing*, 2012, pp. 2188-2192, doi: 10.1109/ICOSP.2012.6492015.
- [14] M. -K. You and G. -Y. Song, "SystemVerilog-based verification environment using SystemC custom hierarchical channel," *2009 IEEE 8th International Conference on ASIC*, 2009, pp. 1310-1313, doi: 10.1109/ASICON.2009.5351242.
- [15] Rajesh C., Shivananda, Shanthi V. A., "Design and development of verification environment to verify spi master core using UVM," *International Journal of Scientific Engineering and Technology (IJSET)* pp. 601–603, 2015. [Online]. Available: <https://www.ijset.in/wp-content/uploads/2015/06/10.2348.ijset06150601.pdf>
- [16] M. Sekhar, "Design and Verification of Serial Peripheral Interface using OVM," *Ijecet.Org*, vol. 2, no. 6, pp. 267–269, 2012.
- [17] P. Kumar M. B. and 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, no. 14, pp. 7-11, 2019. [Online]. Available: https://www.ijcseonline.org/pdf_spl_paper_view.php?paper_id=1079&2-IACIT%20-%20152.pdf
- [18] L. S. Kamireddy and L. Saiteja K., "UVM Based Reusable Verification IP for Wishbone Compliant SPI Master core," *arXiv*, 2018. [Online]. Available: <https://arxiv.org/pdf/1809.10845.pdf>
- [19] Shyamala S. C., Kalpana S., Manasa B., and Bindu L., "SIP Controller For Master Core Verification Using UVM", *International Journal of Scientific Development and Research (IJS DR)*, vol. 1, no. 9, 2016. [Online]. Available: <https://www.ijdsr.org/papers/IJS DR1609045.pdf>
- [20] A. Kulkarni and S. M. Sakthivel, "UVM methodology based functional Verification of SPI Protocol," *National Science, Engineering and Technology Conference (NCSET) 2020*, 2020, vol. 1716, doi: 10.1088/1742-6596/1716/1/012035.
- [21] S. Choudhury, G. K. Singh, and R. M. Mehra, "Design and Verification Serial Peripheral Interface (SPI) Protocol for Low Power Applications," *International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)*, vol. 3, no. 10, 2014, doi: 10.15680/IJIRSET.2014.0310048





- [22] S. -L. Chen *et al.*, "A Novel Low-Power Synchronous Preamble Data Line Chip Design for Oscillator Control Interface," *Electronics*, vol. 9, no. 9, doi: 10.3390/electronics9091509.
- [23] Shushma N. and R. C. Biradar, "Design and Verification of SPI Protocol," *International Journal of Engineering Science and Computing*, vol. 9, no. 5, 2019. [Online] Available: [https://ijesc.org/upload/13bdb8087a12b222cdfea635722ae159.Design%20and%20Verification%20of%20SPI%20Protocol%20\(1\).pdf](https://ijesc.org/upload/13bdb8087a12b222cdfea635722ae159.Design%20and%20Verification%20of%20SPI%20Protocol%20(1).pdf)
- [24] P. Polsani, V. Priyanka B., and Y. P. Sai, "Design & Verification of Serial Peripheral Interface (SPI) Protocol," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8 no. 6, pp. 793-796, 2020, doi: 10.35940/ijrte.F7356.038620.
- [25] M. Sandya and K. Rajasekhar, "Design and Verification of Serial Peripheral Interface," *International Journal of Engineering Trends and Technology (IJETT)*, vol. 3, no. 4, 2012. [Online]. Available: <http://ijettjournal.org/volume-3/issue-4/IJETT-V3I4P212.pdf>
- [26] P. Dhaker, "Introduction to SPI Interface," *Analog Dialogue*, vol. 52, 2018. [Online]. Available: <https://www.analog.com/media/en/analog-dialogue/volume-52/number-3/introduction-to-spi-interface.pdf>

BIOGRAPHIES OF AUTHORS



Rajesh Thumma     received his degrees B. Tech in Electronics and Communication engineering from SKEC, in 2007, M. Tech from the BITS khammam under JNTU Hyderabad, Telangana, India and PhD in School of Electronics Engineering from the KIIT University, Bhubaneswar, India in 2018. He has research and teaching experience of more than 13 years. He is visited University of west Bohemia, Pilsen, Czech Republic and Lublin University of Technology, Lublin, Poland as a part research work. He is currently working as an Associate Professor in the Department of Electronics and Communication engineering at Anurag University, Hyderabad, India since July 2010. His research interest includes low power VLSI, resonant Power converters and Machine learning. He can be contacted at email: rajesh.thumma88@gmail.com.



Pilli Prashanth     is a Post graduate (M. Tech) Student in department of Electronics and Communication with a specialization of VLSI System Design from Anurag university, Hyderabad, India (2021). He received his B. Tech degree in Electrical and Electronics Engineering from Indur institute of Engineering and Technology in 2017. His research interests include low power design VLSI, Functional verification of digital circuits, VLSI testing, Field Programmable Gate Arrays (FPGA). He can be contacted at email: prashanthnarsimlu@gmail.com.