SPI Controller for Open-PowerProcessor based Fabless SoC

Dr. Paulraj Epsiba, Professor, Dept. Ece, Sri Indu College Of Engineering And Technology
Darshanam Sandhya Rani, Assistant Professor, Dept. Ece, Sri Indu College Of Engineering And Technology
G.Anitha, Assistant Professor, Dept. Ece, Sri Indu College Of Engineering And Technology

Abstract:- Since integrated circuit designs continuously expanding, which makes the verification process more difficult and time-consuming, effective verification of such circuit designs is essential. As a result, a strong testbench structure is required, one that includes major generic verification components that are highly reusable and are simple to adapt to new designs. The UVM hierarchy is one such design capable of realizing testbench architectures with coverage-driven verification environments with CRT (constrained Random Test). According to the verification plan devised following a thorough review of the SPI protocol requirements, the current effort is appropriately concentrated on SPI One Master and Multi Slave protocol verification using UVM. The UVM Testbench concentrates on generating random vectors that are sent to the SPI module or the DUT (Design Under Test). This method aids in verifying the functionality of SPI by making comparisons with the captured response received via scoreboard. By using acceptable or appropriate test cases, Testbench also validates the functionality and distinguishing characteristics of SPI, and at the conclusion of the test, it delivers a cumulative coverage report of the design.

Keywords:- *UVM, System Verilog, SPI Protocol, Questsim, and EDA-playground mentor.*

I. INTRODUCTION

For applications that need to transport data at 8 or 16-bits per second, the SPI acts as a "3-wire plus chip select" serial bus.Information is transmitted between bus-connected devices through the three wires. Every gadget on the bus serves as both a sender and a recipient simultaneously. Two out of the three lines, one in every direction, are used to transfer data; the third line acts as a serial clock. Some devices may only be senders, while others may only be receivers. The transmitting gadget typically has the ability to receive data as well. A receive-only device is an example of an SPI display, whereas a receiver and send device is an EEPROM. SPI bus-connected devices can be categorised as Master/Slave devices. A master device creates clock and control signals and starts an information transfer on the bus. Through a slave choose (chip enable) line, a slave device is managed by the master and is only operational when selected. For each slave device, a separate select line is typically needed. In a multi-master mode arrangement, the same device can act as both a master and a slave, but only one master can ever control the bus at any given moment. Any non-selected slave device has to withdraw (make the slave output line high impedance). Data is timed into and out of the active devices before being transferred using a standard shift register data transmission method on the SPI bus. SPI devices operate in full duplex mode by transmitting and receiving in this way. The lines on the SPI bus are entirely one-way. Master generates the clock signal, which generally used to synchronize data transfer. Data is transmitted from the master to the slave on the master-output slave-input (MOSI) line and from the slave to the master on the master-input slave-output(MISO) line. The master selects each slave device using a separate select line. Over the SPI bus, data transfer rates range from almost 0 bits per second to 1 MB per second. Typically, data is transferred in blocks of eight or sixteen bits. The serial clock synchronises all data transport (SCLK). Each clock cycle transfers one bit of data. The values of the clock phase and polarity bits dictate four different clock modes for the SPI bus. When new data is to be transferred onto the bus, the clock phase determines which clock edge to use, and the clock polarities establishes the intensity of the clock idle state. Any hardware component with multiple operating modes will have a mechanism for choosing the value of these bits.

Four logic signals are listed for the SPI bus:

- Serial Peripheral Interface Clock (SCLK)
- Master-Output Slave-Input (MOSI).
- Master-Input Slave-Output (MISO).
- Active low Slave Select (SS).

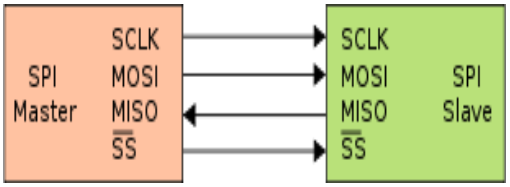


Fig1.Serial Peripheral Interface Architecture

II. RESEARCH GOALS

Building an usable test bench to validate the SPI slave model-based SPI master controller and the AXI bus function model is the goal of this research project. The upcoming goals aid in achieving the desired outcome:

- One must comprehend the SPI protocol architecture and the particular requirements of the AXI platform in order to link the test bench components to the core controller.
- Employing state-of-the-art verification techniques, such as Coverage Driven Functional Verification and Universal Verification Methodology.
- To provide a AXI-compliant SPI master component with reusable Verification IP.

The following are the main contributions of this paper::

1. Become familiar with SPI sub-system architecture, UVM, and System Verilog.
2. Developing an AXI bus function model to enable closed-loop verification testing by serving as a connection between both the test bench and the SPI master device under test (DUT) and SPI slave model.
3. Use UVM libraries, assertions, coverage, limited random stimuli, and System Verilog structures to create hierarchical testbench components.
4. Verify data transmission by experimenting with different character widths and data types.

One of the most often utilized serial protocols in a SoC is SPI since it operates at better bandwidth and throughput than other protocols like UART and I2C. Microcontrollers on the host side and slave devices are frequently made to interact more easily through the SPI Protocol. It is well-liked since it operates with less control signals. The specific SPI core that was the focus of this study serves as a slave AXI compliant device on the host side. Serial Shift Interface (SSI), Clock Generator (CG), and AXI Connection (AXI) are the three fundamental components of the SPI Master Core Controller. The AXI interface that allows for changes to be made to the five 32-bit registers of the SPI core controller.

The serial Peripheral interface consists of serial clock signal and slave select signal. Implementing a high-speed SPI Master/Slave between 900 and 1000 MHz is possible. When two slaves are present, for instance, the core can be built with more flexible SPI-bus control handling. the core's control register, which determines whether the SPI module runs in master or slave mode, which is a key component, can be altered. The SPI status register provides information about running data transfer operations, including their state and whether they have finished or not. One more crucial factor is the adaptability of SPI Interface IPs built with a parameterization technique for different devices. Time Sharing Multiplex (TSM), an advanced design method, in multi-master systems, is utilized to automatically detect the master/slave devices. TSM is used to fix communication issues between many devices. Verification has been harder as a result of the current SoC's increased complexity. The truth is that difficult SoC verification consumes 70% of the time needed for product development.

Reduced verification work is the answer to the issue of time to market. This increasing complexity is managed using contemporary verification techniques. For IP verification, extensive functional coverage employing constraint random simulation technique is required.Several tools are used for this, including scoreboards and coverage monitors. It is crucial to verify that a communication protocol, such as the SPI communication protocol, complies with the design specifications. By adopting a restricted random technique for better functional coverage, effective verification can be accomplished. More recent verification methods and languages have long been advocated by EDA businesses. For a system level verification technique or language to be effective, it must be scalable and reusable of the produced verification components. The combination of System Verilog and object-oriented programming is one of the most promising methods for top - level functional verification for modern complex SOC systems. System Verilog provides a complete verification environment that includes metrics based on coverage, direct and limited random generation, and assertion-based verification. Base class libraries created in System Verilog are used by the Universal Verification Methodology (UVM), the most recent functional verification approach.The AVM from Mentor, the OVM from Mentor & Cadence, the eRM from Verisity, and the VMM-RAL from Synopsys are some of the older technique libraries on which UVM is built.Thanks to this standardization, users can now create highly compatible, portable verification modules. The term "verification components" refers to these modules. For whole systems, modules, or protocols, they are encased and transformed into useable, flexible verification environments. These apps are built upon the extensive base class library. It offers hardware acceleration, emulation, assertion-based verification, coverage-driven restricted random verification, and restricted random verification. It emphasizes simulation.

III. DESIGN OF SPI

Combining a single Master with a single Slave is the simplest configuration for the Serial Peripheral Interface (SPI) [2]. However, interaction between a single Master unit or module and several Slave devices is possible i.e., with more than one Slave device. Any microcontroller device and accompanying peripherals can exchange information utilizes the SPI technology's high-speed, full-duplex, and synchronous communication bus protocol. But when it comes to verification, System Verilog, which is regarded as a hardware description language(HDL), is used to implement the oops programming language and provide a test environment for the SPI protocol. Advanced SystemVerilog features aid in creating a prospective verification environment, yet UVM implementation still enables a standardised verification technique (Universal VerificationMethodology).

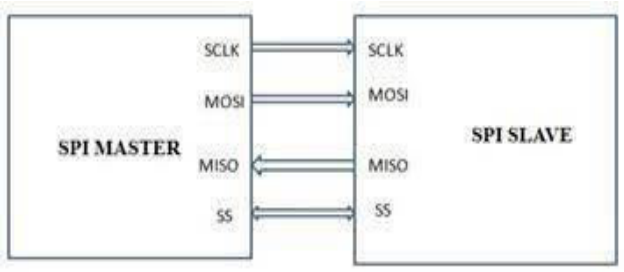


Fig2. Single master- single slave configuration

As a multi-point interface, the SPI protocol connects the devices in a Master-Slave relationship. At this kind of interface, a single device typically a microcontroller plays the role of the Master, while other linked devices.

Single Slave Configuration:
A serial peripheral interface device has multiple slave devices and only one master device. There are 4 signalling pins in the SPI bus protocol . These are them:

- Master-Output Slave-Input (MOSI)
- Master-Input Slave-Output(MISO)

- Serial Clock or master Clock (SCLK, SCK, or MK)
- Slave Select pin (SS)/chip select pin(CS)

Here, each signal pin's operational functionality is listed:

- Serial clock or master clock, often known as SCK or MK: One or more slaves can receive clock signals from this pin, but only the master can modify those signals. But this pin continues to be inactive. When there is no operation, the device is inactive (tri-state).
- Slave Select (SS): This pin is used by the Master module to select the Slave it wants to connect with or send data to.
- Master-Output Slave-Input (MOSI): The Master output pin and the Slave input pin are shared by this. This pin is utilized for data transfer as from Master module to the Slave module. A single direction is present in the pin.
- Master-Input Slave-Output (MISO): Both the Master input and Slave output pins are referred to as this pin. From the Slave unit to the Master unit, data is transmitted using this port. Additionally, it has a single direction.

Multi-Slave Configuration:

A single SPI Master can implement numerous Slaves. The Slaves may be connected as separate components or in a daisy-chain arrangement. Every Slave module that is under the direction of the Master module has its own individual Chip Select (CS) pin in an individual configuration. When the Master activates the Slave Select (SS) port, the selected Slave has access to the information on the MOSI and MISO lines as well as the clock produced by the Master module. However, since the Master cannot tell which Slave is sending or receiving the information, data corruption on the MISO line results when multiple Slave Select (SS) signal ports are active. Figure 3 shows that when the number of Slaves rises, the Master's Chip Select (CS) pin count also gradually rises.

Features of SPI:

- Has full duplex communication capabilities.
- I2C has higher and greater throughput than TWI (integrated Interface circuit).
- For bit transferring, there is no restriction on a particular bit size.
- With more superior and straightforward hardware interfaces than UART and I2C.
- The power need is very little.
- Slaves don't require precise oscillators because they use the master's clock.
- Less circuitry than I2C results in lower power consumption.

Different strategies can be used, such as employing a multiplexer module to regulate a Slave Select (SS) signal, to progressively increase the number of Slaves in each configuration.

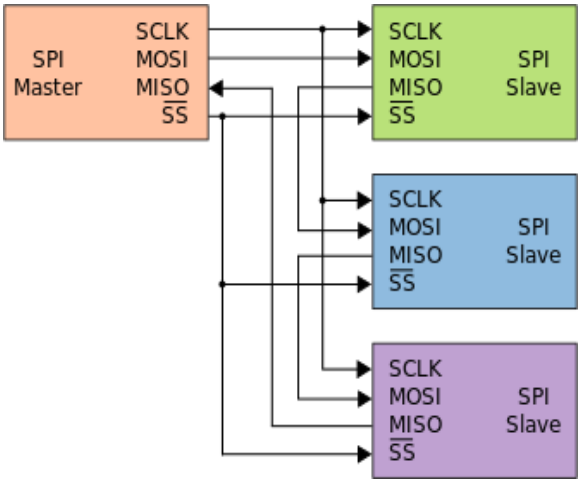


Fig3. Single Master Multi Slave Configuration

IV. UVM METHODOLOGY

Verification involves a test plan, which functions as a roadmap and gives instructions on how to complete the necessary task for verifying the design. The verification plan offers a road map that details the introduction, working hypotheses, test cases to execute, various aspects that can be tested, and the strategy to be used. All of these requirements allow the validation engineer to observe and understand how the testing procedure should be carried out. The verification test plan may be delivered in a variety of formats, including a document, plain text file, or spreadsheet. A Testbench's design and every module's functionality must be described in order to be verified.

In addition to using Assertion-Based Testing (ABT), Coverage Driven Validation (CDV), and Constraint Randomized Generation (CRG), System Verilog, an efficient and promising hardware description language (HDL), also offers a good verification environment. These System Verilog-provided features steadily improve the verification process. System Verilog's feature is its improved hardware modelling, which streamlines the test procedure for the given DUT and gradually and efficiently increases RTL design productivity. Programming Direct Interface is the name of a System Verilog module that is a programming interface that can be used to interact with other language families. System Verilog may handle a variety of foreign languages, including C, C++, System C, and others.

The Universal Verification Methodology (UVM), a replacement for System Verilog, is created to meet the requirement for automated DUT testing. UVM is a powerful set of System Verilog APIs that also includes a set of tried-and-true verification standards that can assist verification engineers in creating a productive and effective atmosphere for testing. Accellera keeps up an approachable open platform. Engineers began developing validation components that were mostly generic and could be applied to other projects as a result of the implementation of UVM approach, which boosted interaction and method and technology sharing across different verification users. The growth of validation components

without altering the source code was also highly encouraged and advocated.

UVM COMPONENTS

Sequence-item: The group consists of the elements or stimuli necessary to provide the intended stimulus. The sequence-items must be randomized in some way in order for the stimulus to be generated. The data variables produced by sequence-items must therefore be explicitly defined using the rand prefix and may also include limitations. The UVM sequence item is extended to create the sequence-item in UVM. Sequence: A sequence grows or produces a collection of sequence items and sends them through a sequencer to the driver. Extending UVM sequence is how the sequence process is carried out.

Sequencer: The UVM sequencer is changed into a class called a sequencer, which manages the response exchange between the sequence and the driver. Driver and Sequencer both use TLM Interface to establish transaction communication.

Driver: By extending the UVM driver, a driver is created. In order for the sequencer and driver to communicate, the TLM port (seq_item_port) needs to be addressed. Driver sends data to DUT via interface connection.

Monitor: A passive type component, by modifying the UVM monitor class, the monitor class is produced. At the virtual interface level, it samples DUT signals and transforms signal-level actions into transaction-level operations. Monitor class, drives DUT signals, but does not.

Agent: The UVM agent class is extended to create the Agent. Driver, monitor, collector, and sequencer are some examples of the verification components that the agent incorporates or groups. It is used to establish TLM connections between the aforementioned components. The agent has an operational mode that can be either active or passive, and occasionally both. Scoreboard: This class accepts data from the monitor and compares the numbers to what is expected. The reference model generates the expected values; however, the driver class can also be used to retrieve a copy.

Environment: The uvm_env class is extended to create the environment class. Other classes like agents, scoreboards, and top-level monitors are grouped under this class.

Test: By expanding the uvm_test, the Test is created. It belongs to the very top class. The Test class, which is the top-level class, is in responsibility of creating the Testbench, configuring it, and starting the many components that go into it.

Interface: Interface serves as a link between the verification environment and the design-under-test, as depicted in Figure 4. All pin-level connections to the DUT are contained within the interface. A collection of nets or variables make up an interface.

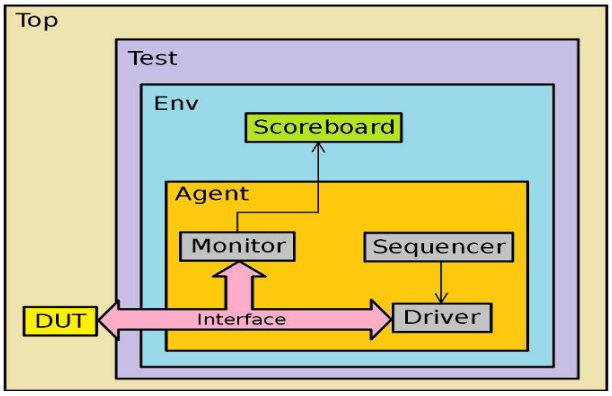


Fig4. UVM ARCHITECTURE

V. VERIFICATION OF SPI

A sequence item is the first component of the UVM environment's architecture. Typically, uvm transaction or uvm sequence item classes are expanded to create sequence item class objects. It is composed of all required information exchange which can be either randomised or constrained to a specific boundary by using UVM structures. Sequences are created by extending uvm sequences to produce additional sequence components. To drive DUT pins, the driver receives the created sequences. There are duties in the SPI master core driver. The following sequence item must be obtained as the driver's initial action. Second, we control data transmission. Third, the sequence item is finished once the packet is written to the UVM analysis port. A fork...join call is used to run the jobs concurrently. The design of the testbench entails the creation of a monitor that keeps track of how the DUT interacts with the testbench. When the protocols are broken, an error is reported after watching the pin level transaction at the DUT's outputs. These UVM components are all connected by the agent. On the scoreboard, the DUT's actual reaction is contrasted with the expected response after predicting the expected output of the DUT in the monitor. The env class handles the creation and connection of the agent and scoreboard.

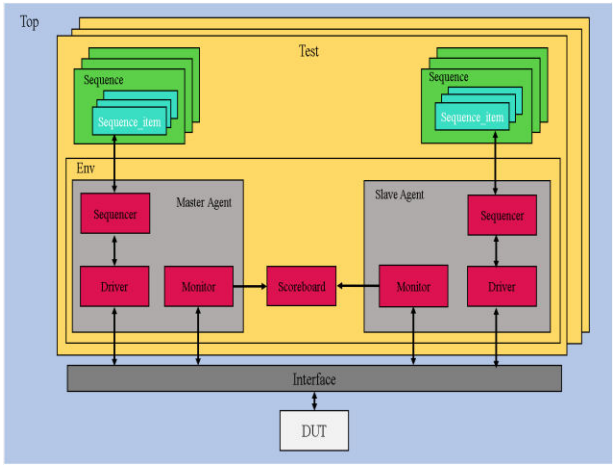


Fig5. VIP Development of SPI Architecture

As listed here, each UVM class includes a variety of simulation stages.

BUILD PHASE: The UVM testbench simulation starts with the construction phase, which creates instances of each UVM component.

CONNECT PHASE: Connections are made between the subcomponents during this phase. Using TLM ports, testbench connections are established.

ELABORATION PHASE: The elaboration phase involves checking connections and setting up address ranges, values, and pointers.

SIMULATION PHASE: Phase one of simulation involves setting up initial runtime configurations and validating UVM testbench topology.

RUN PHASE: Run phase is a task-based simulation because of this it takes more time than other phases. Time 0 begins this phase.

EXTRACT PHASE: In this stage, to build final statistics, information is gathered from the DUT and the scoreboard.

CHECK PHASE: It is used to verify that the DUT functioned as expected and to spot any mistakes that may have happened while the test bench was being run.

REPORT PHASE: In the Report phase, simulation results are provided for the verification engineer to review.

FINAL PHASE: It is used to perform any additional unfinished tasks that have not yet been handled by the test bench.

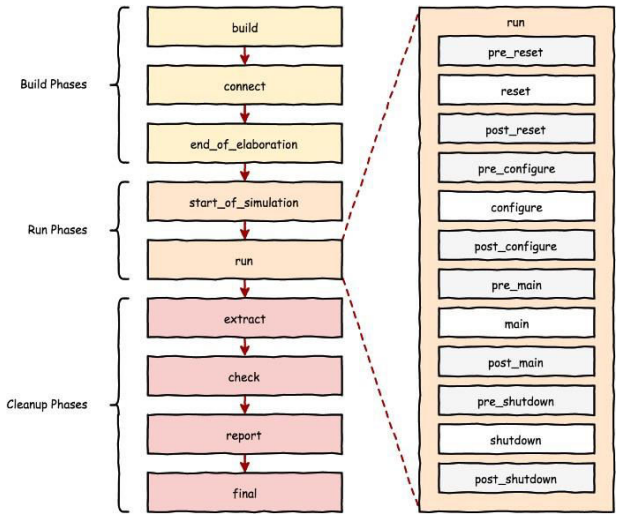


Fig6. UVM PHASES

VI. RESULTS

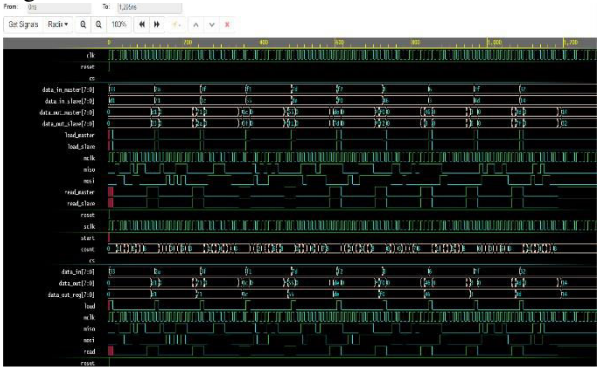
Transcript results

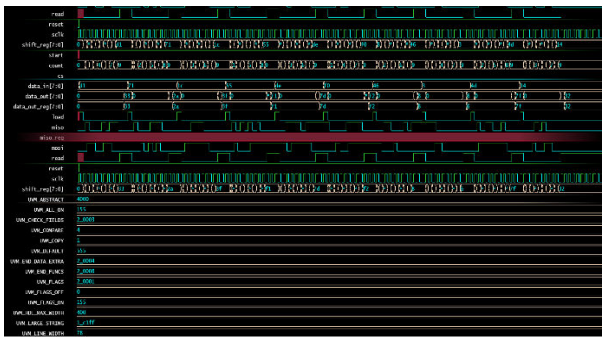
Fig7. Transcript Results

Waveform results:

The AXI bus function model is used by the SPI master and AXI bus function model to communicate. In terms of the AXI protocol, the read, write, and reset features of the architecture are its key functions. The sclk signal, which is used to synchronize communications between the master and slave. The control registers of the master and slave are both set up ahead of the transfer. The send and receive signals' sampling edges are specified by flags in the control register like tx negedge/rx posedge. These two flags ought to have distinct values to one another because SPI write output and read input both happen at the identical single buffer when using a shift register approach. once each and every SPI register has been set, the go flag of the control loop must be asserted in order for the transfer to begin.The testbench makes use of the current flag transfer to synchronize the watch each element of the endless loop. Finally, the transaction in progress signal is emitted, as seen in Figure, after 32 clock cycles, indicating that the AXI interface can now collect the data.

Fig8. SPI Master to Slave Communication





VII. CONCLUSION AND FUTURESCOPE

The results presented here are restricted to a single master and slave, but if utilized for several slaves, the components can be simply reused, as indicated by the structural model of the environment. System Verilog was employed in the project to build the verification environment for the SPI Interface. The objective was to obtain qualities like durability and renewability. In order to allow for configuration based on need, we built parameterized components. Reusability can be accomplished utilizing the flow provided in the paper, according to outcomes analysis.

REFERENCES

[1]. An introduction to i2c and SPI protocols ,by Frédéric Leens, IEEE Instrumentation & Measurement Magazine February 2009.

[2]. S. Sutherland, S. Davidmann, and P. Flake, System Verilog for Design: A Guide to Using System Verilog for Hardware Design and Modeling, 2nd ed. Springer Publishing Company, Incorporated, 2010.

[3]. IEEE Standard for Universal Verification Methodology Language Reference Manual, 2017.

[4]. Design and Verification of Serial Peripheral Interface, by M.Sandya, K.Rajasekhar, International Journal of Engineering Trends and Technology- Volume3Issue4-2012

[5]. IP Design of Universal Multiple Devices SPI Interface , by Tianxiang Liu , Yunfeng Wang.

[6]. S. Simon, SPI Master Core Specification, pp 1-13, www.opencores.org, 2004.

[7]. W. Ni and J. Zhang, “Research of reusability based on UVM verification,” in 2015 IEEE 11th International Conference on ASIC (ASICON), Nov 2015, pp. 1–4.

[8]. K. Fathy and K. Salah, “An Efficient Scenario Based Testing Methodology Using UVM,” in 2016 17th International Workshop on Microprocessor and SOC Test and Verification (MTV), Dec 2016, pp. 57–60.

[9]. P. Rajashekar Reddy, P. Sreekanth, and K. Arun Kumar, “Serial Peripheral Interface-Master Universal Verification Component using UVM,” International Journal of Advanced Scientific Technologies in Engineering and Management Sciences, vol. 3, p. 27, 06 2017.

[10]. R. Prasad and C. S. Rani, “UART IP CORE VERIFICATION BY USING UVM,” IRF International Conference, 15 2016.