ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 4, Issue 8, August 2015

RTL DESIGN AND VERIFICATION OF SPI MASTER-SLAVE USING UVM

Roopesh D
M.Tech, Dept. of ECE, Dr.AIT, Bengaluru

Siddesha K
Asst. Prof., Dept of ECE, Dr.AIT, Bengaluru

Kavitha Narayan B M
Asst. Prof., Dept of TCE, Dr.AIT, Bengaluru

***Abstract* – The main objective of this paper is to design the SPI master-slave and verify the design using UVM. SPI stands for Serial Peripheral Interface. As the name suggests, SPI is a serial synchronous interface. It provides connection between the hosts usually, a microcontroller and slave devices. SPI interface is an On-chip interface. SPI protocol is one of the widely used serial protocols used in a SoC. The reason for its wide usage is its simplicity to use and have few signals to control. The SPI is designed in Verilog using Xilinx and the verification is done in UVM using QuestaSim.**

Keywords – Soc, SPI, Wishbone, UVM

I. INTRODUCTION

Generally, in a SoC there are multiple components working together for a specific application. For these components to operate and communicate, interfaces are used. So, interfaces play an important role in the speed of communication in the SoC. SPI interface can operate at a high speed of up to 1.1Mbps.

Based on the way the data is transmitted, interfaces are divided into two types, namely – Serial interface and Parallel interface. The usage of these interfaces depends on the protocols supported by the SoC components. One of the well known Serial interfaces is the Serial Peripheral Interface (SPI).

SPI was first developed by Motorola Semiconductor. Many IC manufactures develop products which are compatible with SPI protocol. SPI interfaces are usually full duplex in nature and operate as master-slave. In this paper Wishbone interface acts as the host to the SPI core.

II. SPI

SPI core consists of three parts, Clock generator, Shift register and a Wishbone interface. The Clock generator is the host side clock which acts as the master clock for the whole design. The Shift register is used transfer and sample the data. The SPI master can support up to 8 slaves. But, at a time only one slave can access the SPI master.

There are 4 Shift registers, each of 32 bits. So, SPI core can support data transmission up to 128 bits. It has the ability to transfer MSB or LSB first data transfer. It also supports transmit and receive at both rising and falling edge of the serial clock. The data transfer occurs serially between the master and slave devices. Also, the data transfer is completely synchronous to serial clock and full duplex data transfer.

The serial data transfer involves 4 serial signals – Serial clock (SCLK), Master in Slave out (MISO), Master out Slave in (MOSI) and Slave Select (SS). Serial clock generates clock based on the master clock using the formula as shown below. All the serial data transfer depends on this Serial clock. MISO signal is used transfer data from slave device to the SPI master. MOSI signal is used to transfer data from SPI master to slave device. The actual data transmission occurs in MISO and MOSI signals which are single bit. The Slave Select signal is an active low signal, which selects one of the 8 slaves. Hence, the Slave Select signal is of 8 bits.

The SPI operates in 4 different modes, based on the data transmitting and receiving on rising or falling edge of the serial clock. The 4 different modes are controlled the two registers, namely – Clock Phase register (CPHA) and Clock Polarity register (CPOL). The CPHA register decides the clock edge at which the data to be transmitted. The CPOL register decides the clock edge at which the data to be sampled.

Apart from these registers, there are also Divider register, Slave select register and Control and Status register. The Divider register stores the value which is used in generating the serial clock. The Slave select

register stores the value for which slave to be selected. The Control and Status register holds the status of all the control signals and can be changed here.

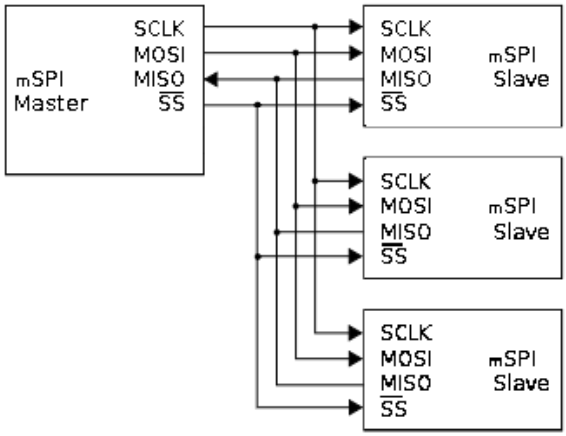


Fig. 1 SPI Block Diagram

III. WISHBONE INTERFACE

The Wishbone is the host for the SPI core. The Wishbone initiates the write and read cycle. Through Wishbone the SPI is configured to perform the required data transfer. The master clock is generated by the wb_clk_i signal. The reset signal (wb_rst_i) is synchronous and active high. The addresses are generated by 32 bits wb_adr_i signal. The 32 bits data to the SPI core is transmitted through the wb_dat_i signal. The 32 bits data from the SPI core is obtained from wb_dat_o signal. The byte select signal (wb_sel_i) is used to byte select the 32 bits data i.e., partial selection of the data.

There are few control signals to control the flow of data. The write enable input signal (wb_we_i) determines whether the current cycle is write (if HIGH) or read (if LOW) cycle. A strobe signal (wb_stb_i) is used to select the core. A valid bus cycle input signal (wb_cyc_i) is used to indicate a valid bus cycle for data transmission. The wb_ack_o signal is the output from the core to indicate the completion of data transfer among the core and Wishbone interface. The wb_int_o signal is the output from the core to indicate that the data transfer between the SPI master and slave is completed. An error signal (wb_err_o) is used to indicate error bus cycle. The suffix 'i' in the signal names indicate that those signals are input to the SPI core and suffix 'o' indicates that those signals are output from the SPI core.

IV. VERIFICATION FLOW

The Design is coded in Verilog language. The RTL Code will be compiled to check the syntax and semantic errors. For verification purpose Universal Verification methodology (UVM) is used. After the simulation, code coverage is performed to increase the confidence of code functionality. Along with code coverage, functional coverage is also performed. Different coverage's should be observed such as Branch, Expression and Toggle. The synthesis can be carried out to generate a technology specific Gate-level net list. The figure 2 shows Verification methodology of the project.

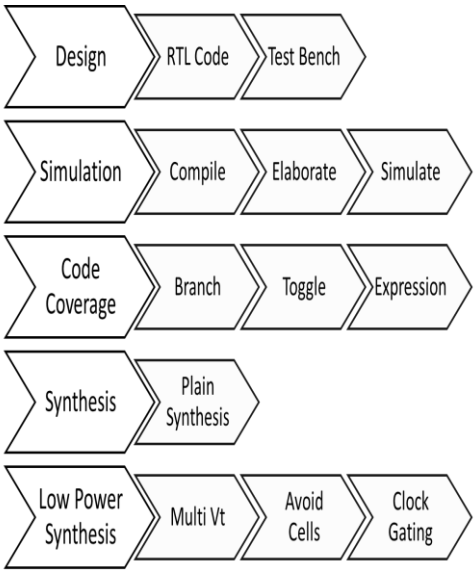


Fig. 2: Verification Methodology Flow

V. VERIFICATION ENVIRONMENT

The Verification Plan is based on System Verilog Hardware Verification Language. To verify the functionality of the design, a Verification environment is created in Universal Verification Methodology (UVM). The UVM environment is based on the System Verilog.

Based on the requirements for the project, the following points are considered while the verification architecture is built.

- Re-usability of the verification IP.
- Which building blocks the verification language can support.
- Controllability in generation of the stimulus.
- Next phase is building the Verification environment.

- Final phase would be to verify the DUT (RTL code) using the constructed verification environment.

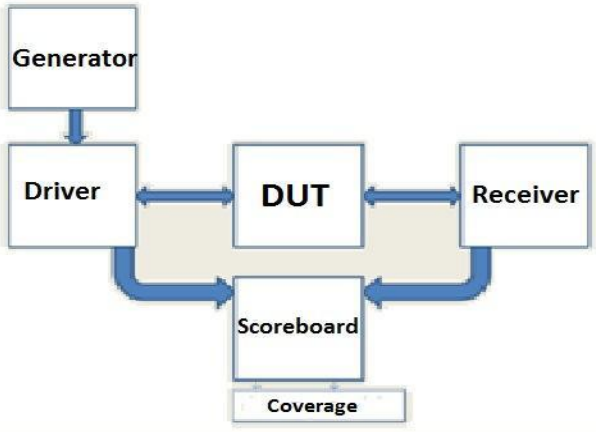


Fig. 3: The Verification Environment

The stimulus is generated and driven to the Design under Test (DUT) according to the SPI protocol. The functionality is verified in the scoreboard and a coverage report is generated.

VI. IMPLEMENTATION AND SIMULATION RESULTS

The complete design is developed in Verilog 2001 and synthesized. The synthesis code will be technology independent. The synthesized code is then verified in System Verilog based UVM.

SIMULATION RESULTS:

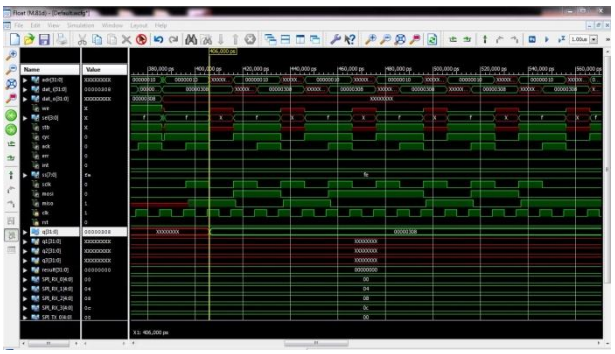


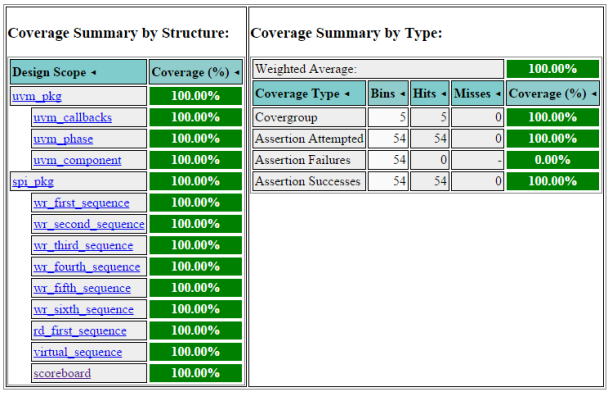
Fig. 4: Output waveform showing Full Duplex operation of the SPI module.

The synthesized code is simulated as shown in figure 4.

As per the SPI protocol the 8 bits data are transferred between the SPI master and slave. As seen, during simulation that the 8 bits data transfer takes 8 serial clock pulses. Also, the data transmission between the SPI master and slave is full duplex. The 8 bits data transfer occurs in MISO and MOSI serial signals.

VERIFICATION RESULTS:

Using System Verilog based UVM we have developed the verification environment and verified the synthesized RTL code. Then we have obtained the functional coverage report from QuestaSim. Figure 5 shows the functional coverage achieved which is obtained from the QuestaSim report.



Coverage Summary by Structure:

Design Scope	Coverage (%)
uvm_pkg	100.00%
uvm_callbacks	100.00%
uvm_phase	100.00%
uvm_component	100.00%
spi_pkg	100.00%
wr_first_sequence	100.00%
wr_second_sequence	100.00%
wr_third_sequence	100.00%
wr_fourth_sequence	100.00%
wr_fifth_sequence	100.00%
wr_sixth_sequence	100.00%
rd_first_sequence	100.00%
virtual_sequence	100.00%
scoreboard	100.00%

Coverage Summary by Type:

Weighted Average:				100.00%
Coverage Type	Bins	Hits	Misses	Coverage (%)
Covergroup	5	5	0	100.00%
Assertion Attempted	54	54	0	100.00%
Assertion Failures	54	0	-	0.00%
Assertion Successes	54	54	0	100.00%

Fig. 5: QuestaSim Report showing the functional coverage.

The whole design of the Serial Peripheral Interface (SPI) with Single Master and multiple Slaves configuration has been successfully completed showing that it can operate in Full Duplex Mode and also verified the SPI design using System Verilog based UVM with constrained Randomization method. Also, we obtained a code coverage and functional coverage of 100%.

VI. CONCLUSION

In this paper, the SPI master-slave is designed using Verilog and the developed design is verified using UVM. All the necessary functional registers are implemented. The data present in the SPI master is successfully verified with the data present in the slave for all the different cases. Finally a coverage report is generated for the functionality verification. The designed SPI can be added with other functionality as per future requirements. This SPI protocol mainly finds application in On-chip communication.

REFERENCES

[1] Ayas Kanta Swain and Kamala Kanta Mahapatra "Design and Verification of Wishbone Bus

Interface for System-on-Chip Integration" 2010 Annual IEEE India Conference (INDICON).

[2] K.Aditya, M. Sivakumar, Fazal Noorbasha and T. Praveen Blessington "Design and Functional Verification of A SPI Master Slave Core Using System Verilog" International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, May 2012.

[3] www.opencore.org.Simon Srot. "SPI Master Core Specification", May 16, 2007.

[4] Mohandeep Sharma and Dilip Kumar "WISHBONE bus architecture – a survey and comparison" International Journal of VLSI design & Communication Systems (VLSICS) Vol.3, No.2, April 2012.

[5] Sharma. M and Kumar D "Design and synthesis of Wishbone bus Dataflow interface architecture for SoC integration" India Conference (INDICON), 2012 Annual IEEE.

[6] Gao Yong, Yang Yuan and Fan Bei "The application of bus coding in designing of low power SoC based on Wishbone" Computer Science and Information Processing (CSIP), 2012 International Conference.

[7] Jianlong Zhang, Chunyu Wu, Wenjing Zhang and Jiwei Wang "The design and realization of a comprehensive SPI interface controller" Mechanic Automation and Control Engineering (MACE), 2011 Second International Conference.

[8] Zhili Zhou, Zheng Xie, Xin'an Wang and Teng Wang "Development of verification environment for SPI master interface using SystemVerilog" Signal Processing (ICSP), 2012 IEEE 11th International Conference.

[9] Chris spear "System verilog for verification" second edition.

[10] Abhijeet Kumar, Manish Kundu "Implementation of SPI protocol with clock domain crossing", International Journal of Research and Innovative Technology (IJRIT),Vol. 1,Issue 2, June 2014.