SPI Controller Core: Verification

Nidhi Gopal
Graduate Apprentice, Bharat Heavy Electricals Limited, Haridwar, Uttarakhand, India

Abstract

This paper mainly deals with the study of Serial Peripheral Interface and logical Implementation through RTL, Synthesis and Simulation by making Test benches of various modules involved using Universal Verification Methodology. It is done with the help of Questasim 10.0b software. Further, output in both Batch and GUI Mode has been observed and discussed. Various test cases of SPI Protocol are taken into consideration, functional coverage, code coverage, and assertion coverage has been verified by synthesis of various blocks involved in top level architecture of SPI.

Keywords: *Questasim, RTL, SPI, Synthesis, Simulation, Testcases.*

I. INTRODUCTION

In this paper, SPI Controller Core Verification, done using Questasim 10.0 b tool in Linux Environment is discussed. Universal Verification Methodology and System Verilog is used for analysis and verification of this, and, RTL and Test bench coding was done, and results were simulated as well as analyzed. Also, in order to minimize the bugs, all the possible scenarios/steps were taken care in the programming.

II. THEORETICAL BACKGROUND

The Serial Peripheral Interface (SPI) bus is a synchronous serial communication interface specification used for short distance communication, primarily in embedded systems. SPI devices communicate in full duplex mode using a master-slave architecture with a single master. The master device originates the frame for reading and writing. Multiple slave devices are supported through selection with individual slave select (SS) lines.

Sometimes SPI is called a *four-wire* serial bus, contrasting with three-, two-, and one-wire serial buses. The SPI may be accurately described as a synchronous serial interface, but it is different from the Synchronous Serial Interface (SSI) protocol, which is also a four-wire synchronous serial communication protocol, but employs differential Signaling and provides only a single simplex communication channel.

Synchronous Serial Interfaces are widely used to provide economical board-level interfaces between different devices such as microcontrollers, DAC's and ADC's and other. Although there is no single standard for synchronous serial bus, there are industry wide accepted guidelines based on two popular implementations:

- SPI(a trademark of Motorola semiconductors)
- Microwire Plus(a trademark of Motorola semiconductors).

The SPI Master core is compatible with both above mentioned protocols with some additional functionality. At the hosts side, the core acts like a WISHBONE complaint slave device.

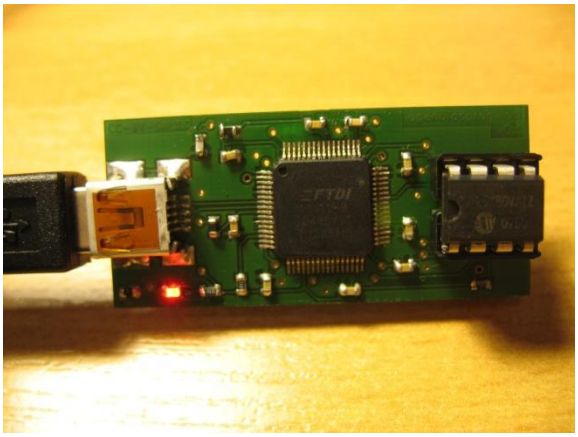


Figure 1: SPI Device

A. Features of SPI

- Full duplex synchronous serial data transfer.
- Variable length data word up to 128 bits.
- MSB or LSB first data transfer.
- Rx and Tx on both rising and falling edge of serial clock independently.
- 8 slave select lines.
- Fully static synchronous design with one clock domain.
- Technology independent Verilog.
- Fully synthesizable.

B. Interfaces of SPI :

The SPI bus specifies four logic signals:

- SCLK : Serial Clock (output from master).
- MOSI : Master Output, Slave Input (output from master).
- MISO : Master Input, Slave Output (output from slave).
- SS : Slave Select (active low, output from master).

Alternative naming conventions are also widely used, and SPI port pin names for particular IC

products may differ from those depicted in these illustrations:
Serial Clock:

- SCLK: SCK, CLK.
  Master Output --> Slave Input.
- MOSI : SIMO, SDO (for master devices), SDI(for slave devices), DI, DIN, SI, MTST.
  Master Input <-- Slave Output.
- MISO: SOMI, SDO (for slave devices), SDI(for master devices), DO, DOUT, SO, MRSR.
  Slave Select:
- SS: nCS, CS, CSB, CSN, EN, nSS, STE, SYNC.

C. *Advantages Of SPI:*

- Full duplex communication in the default version of this protocol.
- Push-pull drivers (as opposed to open drain) provide good signal integrity and high speed.
- Higher throughput than I²C .
- Complete protocol flexibility for the bits transferred
- Not limited to 8-bit words

D. *Disadvantages Of SPI:*

- Requires more pins on IC packages than I²C, even in the *three-wire* variant.
- No in-band addressing; out-of-band chip select signals are required on shared buses.
- No hardware flow control by the slave (but the master can delay the next clock edge to slow the transfer rate).
- No hardware slave acknowledgment (the master could be transmitting to nowhere and not know it).
- Supports only one master device.

E. *Applications of SPI :*

SPI is used to talk to a variety of peripherals, such as:

- Sensors: temperature, pressure, ADC, touch screens, video game controllers,
- Control devices: audio codecs, digital potentiometers, DAC.
- Camera lenses: Canon EF lens mount,
- Communications: Ethernet, USB, USART , CAN, IEEE 802.15.4, IEEE 802.11, handheld video games,
- Memory: flash and EEPROM,
- Real-time clocks,
- LCD, sometimes even for managing image data,
- Any MMC or SD card (including SDIO variant).

For high performance systems, FPGAs sometimes use SPI to interface as a slave to a host, as a master to sensors, or for flash memory used to bootstrap if they are SRAM-based.

III. DATA TRANSMISSION IN SPI

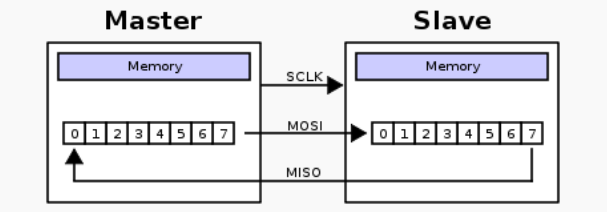


Figure 2 : A typical hardware setup using two shift registers to form an inter-chip circular buffer

To begin communication, the bus master configures the clock, using a frequency supported by the slave device, typically up to a few MHz. The master then selects the slave device with a logic level 0 on the select line. If a waiting period is required, such as for analog-to-digital conversion, the master must wait for at least that period of time before issuing clock cycles.
During each SPI clock cycle, a full duplex data transmission occurs. The master sends a bit on the MOSI line and the slave reads it, while the slave sends a bit on the MISO line and the master reads it. This sequence is maintained even when only one-directional data transfer is intended.

Transmissions normally involve two shift registers of some given word size, such as eight bits, one in the master and one in the slave; they are connected in a virtual ring topology. Data is usually shifted out with the most-significant bit first, while shifting a new least-significant bit into the same register. After that register has been shifted out, the master and slave have exchanged register values. If more data needs to be exchanged, the shift registers are reloaded and the process repeats. Transmission may continue for any number of clock cycles. When complete, the master stops toggling the clock signal, and typically deselects the slave.
Transmissions often consist of 8-bit words. However, other word sizes are also common, for example, 16-bit words for touch screen controllers or audio codec, such as the TSC2101 by Texas Instruments, or 12-bit words for many digital-to-analog or analog-to-digital converters.

Every slave on the bus that has not been activated using its chip select line must disregard the input clock and MOSI signals, and must not drive MISO. The master must select only one slave at a time.

IV. TRANSCATION DETAILS, VERIFICATION PLAN AND TESTCASES OF SPI PROTOCOL

Following are the SPI interface signals, internal connections and registers used in SPI device.

A. *WISHBONE Interface Signals*

S.No	PORT	WIDTH	DIRECTION	DESCRIPTION
1	wb_clk_i	1	input	Master clock
2	wb_rst_i	1	input	Synchronous reset, active high
3	wb_add_i	5	input	Lower address bits
4	wb_data_i	32	input	Data towards the core
5	wb_stb_i	1	input	Strobe signal/core select input
6	wb_cyc_i	1	input	Valid bus cycle input
7	wb_we_i	1	input	Write enable input
8	wb_data_o	32	output	Data from the core
9	wb_ack_o	1	output	Bus cycle acknowledge output
10	wb_err_o	1	output	Bus cycle error output
11	wb_int_o	1	output	Interrupt signal output
12	wb_sel_i	1	input	Byte select signal

All WISHBONE signals will be registered, driven and latched on rising edge of wb_clk_i.

B. *SPI External Connections*

SR NO	PORT	WIDTH	DIRECTION	DESCRIPTION
1	ss_pad_o	8	output	Slave select output signals
2	sclk_pad_o	1	output	Serial Clock Output
3	mosi_pad_o	1	output	Master out slave in data signal output
4	miso_pad_i	1	input	Master in slave out data signal input

C. *Registers*

1) *Core Registers List*

Sr. No.	NAME	ADDRESS	WIDTH	ACCESS	DESCRIPTION
1.	Rx0	0x00	32	R	Data receive reg 0
2.	Rx1	0x04	32	R	Data receive reg 1
3.	Rx2	0x08	32	R	Data receive reg 2
4.	Rx3	0x0c	32	R	Data receive reg 3
5.	Tx0	0x00	32	R/W	Data transmit reg 0
6.	Tx1	0x04	32	R/W	Data transmit reg 1
7.	Tx2	0x08	32	R/W	Data transmit reg 2
8.	Tx3	0x0c	32	R/W	Data transmit reg 3
9.	CTRL	0x10	32	R/W	Control & status
10.	DIVIDER	0x14	32	R/W	Clock divider register
11.	SS	0x18	32	R/W	Slave select register

All registers are 32-bit wide and accessible only with 32 bits (all wb_sel_i signals must be active).

2) *Data Receive registers [RxX]*

Bit#	31:0
Access	R
Name	Rx

Reset Value : 0x00000000

RxX : The data receive registers hold the value of received data of the last executed transfer. Valid bits depend on the character length field in the CTRL register(i.e. if CTRL[6:0] is set to 0x08, bit RxL [7:0] holds the received data). If character length is less or equal to 32 bits, Rx1, Rx2, and Rx3 are not used, if character length is less than 64 bits, Rx2 and Rx3 are not used and so on.

3) **Data Transmit register [TxX]**

Bit#	31:0
Access	R
Name	Rx

Reset Value : 0x00000000

TxX: The data receive registers hold the data to be transmitted in the next transfer. Valid bits depend on the character length field in the CTRL register (i.e. if CTRL[6:0] is set to 0x08, the bit Tx0 [7:0] will be transmitted in next transfer). If character length is less or equal to 32 bits, Tx1, Tx2, and Tx3 are not used, if character length is less than 64 bits, Tx2 and Tx3 are not used and so on.

4) ***Control and Status register[CTRL]***

Bit#	31:14	13	12	11	10	9	8	7	6:0
Access	R	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Name	Reserved	ASS	IE	LSB	Tx_NEG	Rx_NEG	GO_BSY	Reserved	CHAR_LEN

Reset Value : 0x00000000

ASS: If this bit is set, ss_pad_o signals are generated automatically. This means that slave select signal, which is selected in SS Register is asserted by SPI Controller, when transfer is started by setting CTRL[GO_BSY] and is de-asserted after transfer is finished. If this bit is cleared, slave select signals are asserted and deasserted by writing and clearing bits in SS Register.

IE: If this bit is set, the interrupt output Is set active after a transfer is finished. The interrupt signal is deasserted after a Read or Write to any register.

LSB: If this bit is set, the LSB is sent first on the line (bit TxL[0]), and the first bit received from the line will be put in the LSB position in the Rx register (bit RxL[0]). If this bit is cleared, the MSB is transmitted/received first (which bit in TxX/ RxX register that s depends on the CHAR_LEN field in the CTRL register).

Tx_NEG: If this bit is set, the mosi_pad_o signal is changed on the falling edge of a sclk_pad_o clock signal, or otherwise the mosi_pad_o signal is changed on the rising edge of sclk_pad_o.

Rx_NEG: If this bit is set, the miso_pad_i signal is latched on the falling edge of a sclk_pad_o clock signal, or otherwise the miso_pad_i signal is latched on the rising edge of sclk_pad_o.

GO_BSY: Writing 1 to this bit starts the transfer. This bit remains set during the transfer and is automatically cleared after the transfer finished. Writing 0 to this bit has no effect.

CHAR_LEN: This field specifies how many bits are transmitted in one transfer. UPto 64 bits can be transmitted.

CHAR_LEN =0x01…1 bit
CHAR_LEN=0X02…2 bit
.
.
.
CHAR_LEN=0x7f…127 bits
CHAR_LEN=0x00…128 bits

5) **Divider Register**

Bit#	31:16	15:0
Access	R	R/W
Name	Reserved	DIVIDER

Reset Value: 0x0000ffff

DIVIDER: The value in this field is the frequency divider of the system clock wb_clk_i to generate the serial clock on the output sclk_pad_o. The desired frequency is obtained according to the following equation:

$$Fsclk = fwb\_clk/(DIVIDER+1)*2$$

6) ***Slave Select Register[SS]:***

Bit#	31:8	7:0
Access	R	R/W
Name	Reserved	SS

Reset Value: 0x00000000

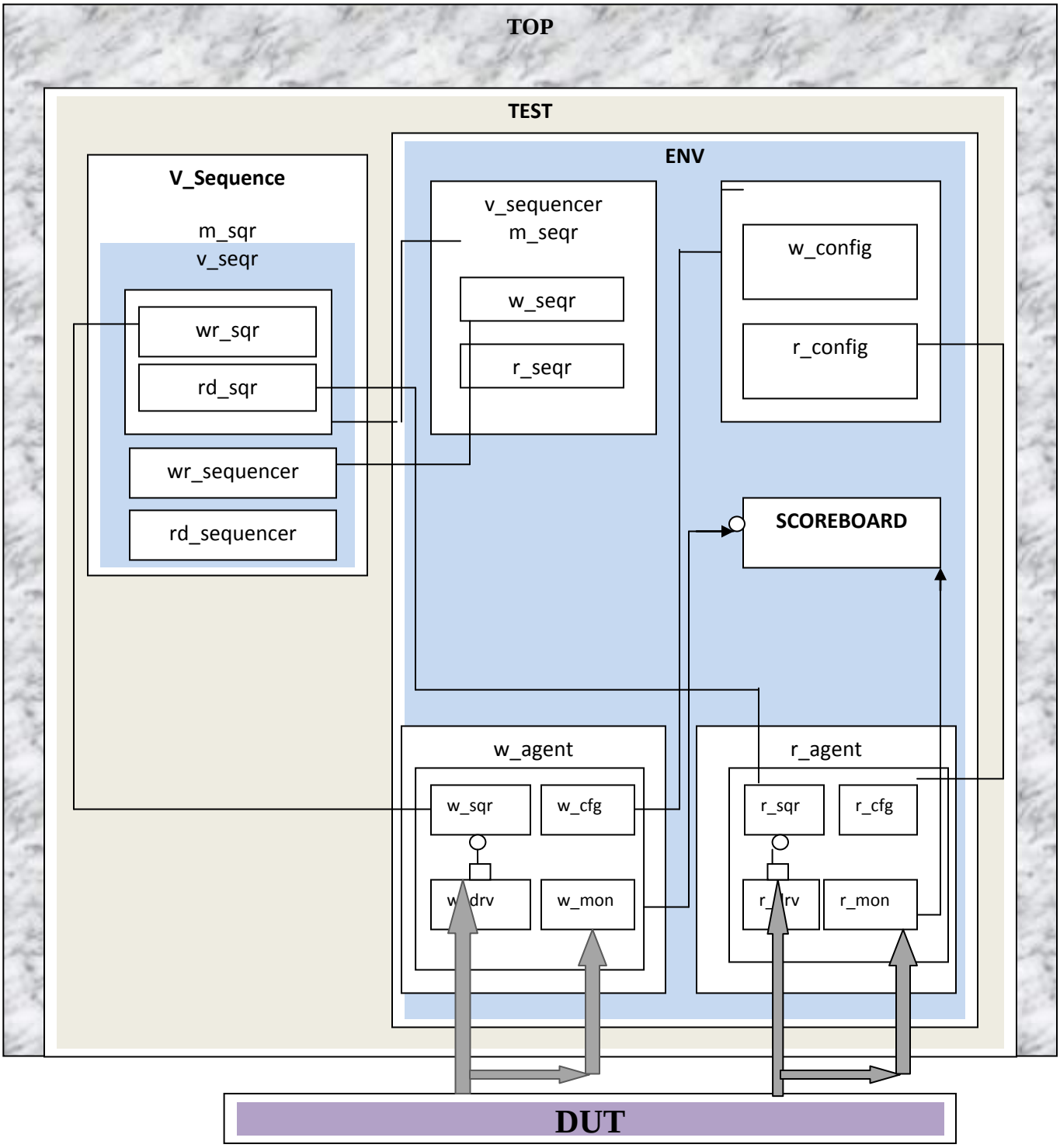
SS: If CTRL[ASS] bit is cleared, writing 1 to any bit location of this field sets the proper ss_pad_o line to an active state and writing 0 sets the line back to inactive state. If CTRL[ASS] bit is set, writing 1 to any bit location of this field will select appropriate ss_pad_o line to be automatically driven to active state for the duration of the transfer, and will be driven to inactive state for the rest of the time.

D. Verification Plan

A verification plan has been made for SPI Protocol, which included various components like Environment, Top module, Test module, Virtual sequence, Virtual Sequencer, Scoreboard, Read agent , Write agent, connected to DUT(Design Under Test).

Following figure shows the connection between them.

All the sub-modules of the SPI are written using UVM. TB coding was done, and verified by Questasim 10.0b software, for all the components involved in SPI Architecture.



E. Testcases Of SPI Protocol

Following are the testcases of SPI Protocol which can be considered:

a) To check different transaction through character length as :
   1) 32 bits
   2) 8 bits
   3) 16 bits
   4) 64 bits, etc.

b) Check Reset condition.

c) Transaction through MSB Bit first

d) Transaction through LSB Bit first.

e) Transitions through posedge and receiving at negedge

f) Transitions through negedge and receiving at posedge.

g) By setting ASS Condition and checking.

h) Checking functionality with or without Interrupt.

III. CONCLUSION

Having few disadvantages like less number of pins, SPI Protocol is found to be very advantageous as it provides full duplex communication, good signal integrity and high speed., higher throughput than I^2C , complete protocol flexibility for the bits transferred and is not limited to 8-bit words. It is having wide range of applications , for example, Sensors: temperature, pressure, ADC, touch screens, video game controllers, Control devices: audio codecs, digital potentiometers, DAC, Camera lenses, Communications: Ethernet, USB, USART, CAN, IEEE 802.15.4, IEEE 802.11, handheld video games, Memory: flash and EEPROM, Real-time clocks, LCD, sometimes even for managing image data, Any MMC or SD card (including SDIO variant). In this paper, advantages, applications, limitations and internal architecture of SPI Protocol has been discussed, and RTL was done after study, in order to simulate , synthesize and verify the design of SPI Protocol.

ACKNOWLEDGEMENTS

Author would like to thank the faculty members of Maven Silicon Softech Pvt. Ltd., who provided their time to time guidance, study material and helped me to complete this project. Specifically, I would like to thank Mrs. Shanti Rao, UVM Faculty, and interns, of Maven Silicon. Author would also like to thank her parents, who provided me the opportunity to work upon this project deeply.

REFERENCES

[1] Prof Jai Karan Singh, prof Mukesh Tiwari, Karan Sharma;"Implementation of SPI SLAVE on FPGA"; International Journal of Advanced Engineering and technology; Volume :20

[2] T. Durga Prasad, B. Ramesh Babu; Design and Simulation of SPI Master / Slave UsingVerilog HDL; International Journal of Science and Research.; Volume:03, Issue:08, August 2014.

[3] Google; Wikipedia.

[4] Maven Silicon Study Material.

[5] Arunadevi A. ,Chitra K., GunaNandhini S., Raghupathi T., Rejusha M; "A Review on Area Efficient Parallel FIR Digital Filter Implementation";Volume:02, Issue:02, SSRG-IJVSP, March-April 2015.

[6] Shilpi Thawait, Jagveer Verma;" FPGA Implementation of Simple and High Speed Vedic Multiplier"; Volume:02; Issue:03, SSRG-IJVSP,May-June 2015.