Generation of Graph for Serial Peripheral Interface Verification

Mr.R.Gowtham
Department of Electronics and Communication Engineering
KPR Institute of Engineering and Technology
Coimbatore, India
e-mail:gowthamkpriet@gmail.com

Mr.V.Govindaraj
Department of Electronics and Communication Engineering
KPR Institute of Engineering and Technology
Coimbatore, India
e-mail:see1govind@gmail .com

***Abstract*—** With the rapid strides in Semiconductor processing technologies, the density of transistors on the die is increasing in line with Moore's law which in turn is increasing the complexity of the whole SOC (System on Chip) design. With manufacturing yield and time-to-market schedules crucial SOC, it is important to select verification and analysis solutions that offer the best possible performance, while minimizing iteration time and data volume. The main objective is to verify SERIAL PERIPHERAL INTERFACE using graph based scenario model. This technique includes generation of graph of APB_WB (Advanced Peripheral Bus Wishbone) and integration of test cases generated by the graph to APB. The graph is generated by the software Trek. This software automatically generates test cases which are self-verifying. The test cases generated from graph-based scenario model captures intended system behavior. Trek takes input information from scenario models describing the desired outcomes, developed by the user.

Keywords- *SPI interface; serial; Registers; Graph Scenario mode; Trek*

*****

I. INTRODUCTION

The field of electronic systems is facing a constant demand for better performance achieved by systems becoming smaller in size. This results in the increase of both complexity and density of electronic hardware, including semiconductor chips and circuit boards. Those trends are emphasizing the importance of saving both space and power. One common way of saving both space and power, within a chip or a board, is to use serial communication. In contrast to parallel communication, which demands a large amount of wires and connecting pins, serial communication can be performed by using only minimal amount of interconnections. The Serial Peripheral Interface bus is a serial data link that operates in full duplex mode. Devices communicate in master / slave mode where the master device initiates the data frame. Serial Peripheral Interface (SPI) is an interface bus commonly used to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards. It uses separate clock and data lines, along with a select line to choose the device we wish to talk to multiple slave devices are allowed with individual slave select (chip select) lines. SPI has an advantage over I²C bus by its throughput and flexibility for the bits transferred.

II. SPI BUS PRINCIPLES

The Serial Peripheral Interface Bus or SPI bus is a synchronous serial data link de facto standard, named by Motorola operates in full duplex mode. Devices communicate in master/slave mode where the master device initiates the data frame. Multiple slave devices are allowed with individual slave select (chip select) lines. Sometimes SPI is called a four-wire serial bus, contrasting with three-, two-, and one-wire serial buses. SPI is often referred to as SSI (Synchronous Serial Interface).SPI is using as a sensor in variety of embedded applications like temperature, pressure, ADC, touch screens, video game controllers and as a control devices in audio codec, digital potentiometer. SPI has an advantage of higher throughput than SMB bus and full duplex communication with no arbitration. Slaves don't need an unique address.

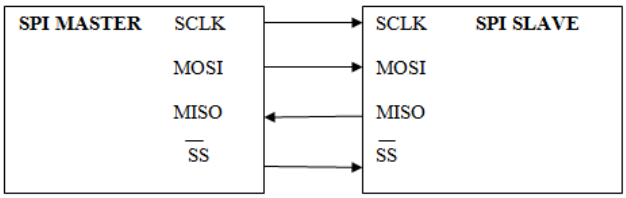

Figure 1. SPI Block Diagram.

II. SPI MODES AND CONFIGURATION

In the independent slave configuration, there is an independent chip select line for each slave. This is the way SPI is normally used. Since the MISO pins of the slaves are connected together, they are required to be tri-state pins. In daisy chain configuration, master and cooperative slaves some products with SPI bus are designed to be capable of being connected in a daisy chain configuration, the first slave output being connected to the second slave input, etc. The SPI port of

each slave is designed to send out during the second group of clock pulses an exact copy of what it received during the first group of clock pulses. The whole chain acts as an SPI communication shift register; daisy chaining is often done with shift registers to provide a bank of inputs or outputs through SPI. Such a feature only requires a single SS line from the master, rather than a separate SS line for each slave.

III. SPI TRANSMISSION

To begin a communication, the bus master first configures the clock, using a frequency less than or equal to the maximum frequency the slave device supports. Such frequencies are commonly in the range of 10 kHz–100 MHZ. The master then transmits the logic 0 for the desired chip over the chip select line. Logic 0 is transmitted because the chip select line is active low, meaning its *off* state is a logic 1; *on* is asserted with a logic 0. If a waiting period is required (such as for analog-to-digital conversion), then the master must wait for at least that period of time before starting to issue clock cycles.

During each SPI clock cycle, a full duplex data transmission occurs while the master sends a bit on the MOSI line; the slave reads it from that same line and the slave sends a bit on the MISO line; the master reads it from that same line. Not all transmissions require all four of these operations to be meaningful, but they do happen. 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 ring. 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 Then each device takes that value and does something with it, such as writing it to memory. If there is more data to exchange, the shift registers are loaded with new data and the process repeats. Transmissions may involve any number of clock cycles. When there is no more data to be transmitted, the master stops toggling its clock. Normally, it then deselects the slave. Transmissions often consist of 8-bit words, and a master can initiate multiple such transmissions if it wishes/needs. However, other word sizes are also common, such as 16-bit words for touch screen controllers or audio codec's, like the TSC2101 from Texas Instruments for many digital-to-analog or analog-to-digital converters. Every slave on the bus that hasn't 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. To begin communication, the master first configures the clock, using a frequency less than or equal to the maximum frequency the slave device supports. Such frequencies are commonly in the range of 1–70 MHZ. The master then pulls the chip select (SPI_SS) low for the desired chip.

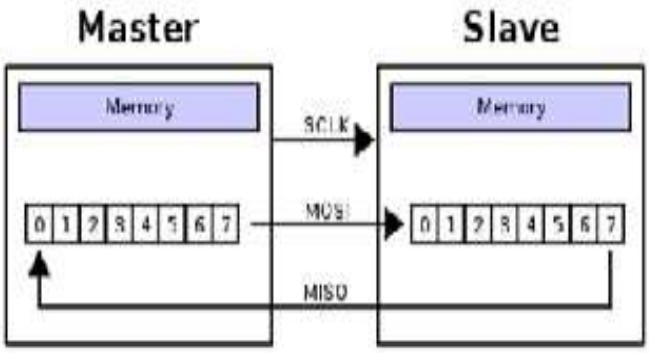

Figure 2. SPI Transmission.

IV. SPI REGISTER

SPI master core basically consists of data transmit register, data receive register, control and status register, clock divider register and slave select register.

A. *Data Receive Register*

Data receive registers are read only registers. SPI master core contains total of 4 receive registers each having 32 bit data width. It is read only register. It holds the received data of the last executed transfer. The number of bits received per transfer depends on the character field of control and status register of the SPI master core. If character field is less than 32 bit, only Rx0 is active and for 64 bits Rx0 and Rx1 are active and so on.

B. *Data Transmit Register*

Data transmit registers are read/write registers. The number of bits transfer in given transfer depends on the character field of control and status register. If an character field is less than 32 bits, the Tx0 is used to transfer the data to SPI slave. If the character field is 64 bits, the TX0 and Tx1 is used and so on.

C. *Control And Status Register*

Control and Status register has the following names of bits with respective operation.

Character length is 7 bit and indicates number bit transfer in a single transfer of SPI. Suppose, CHAR_LEN is set as 1 bit, 1 bit transfers in a single transfer. If it is 64 bits, 64 bits transfer in a single transfer. Likewise maximum of 128 bits are transferred using this character length field in control and status register.

Go busy is bit which indicates start of the data transfer in SPI. If this bit is set indicates start of transfer. The feature of

 ISSN: 2321-8169
1513 - 1516

this bit is that it gets automatically to zero once data transfer is finished indicating end of data transfer.

Rx_Neg bit indicates reception of the data transfer on the particular edge of serial clock cycle. If Rx_Neg=1, it shows data is received on the falling edge of serial clock SCLK else on the rising edge of SCLK.

Tx_Neg bit indicates transmission of the data on the particular edge of SCLK. If Tx_Neg=1, it show that data is transmitted on the falling edge of clock cycle else on the rising edge of SCLK.

LSB bit indicates mode of transfer i.e. either from LSB to MSB or MSB to LSB. If LSB=1, LSB is transmitted first otherwise MSB first.

If IE bit is set, the interrupt output is set active after a transfer is finished. The interrupt signal is deasserted after read or write to any register.

If ASS bit is set, slave is selected automatically. If this bit is cleared, slave is asserted and de-asserted by writing and clearing bits in SS register.

D. Slave Select Register

If slave select bit is cleared, writing 1 to any bit location of this field sets the proper slave select pad line to an active state and writing 0 sets the line back to inactive state. If slave select bit is set, writing 1 to any bit location of this field will select appropriate slave select pad 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.

E. Divider Register

The value in this field is the frequency divider of the system clock to generate the serial clock on the output serial clock pad.

V. GENERATION OF GRAPH

The generation of graph consists two modes of operation is transmit and receive. The option is either it goes transmit or receive. In transmission, the graph has to generate random value of transmitting data, slave register values for selecting the slave select register and finally control and status register value for enabling the transmission. The same process takes for reception of data also. But here first the graph generates the random value of receiving data then the graph generates random slave select register value and control and status register value for reception of data.

```
[gowtham@gowtham run]$ make run
rm -fr *.treko
trekcc ../graph/select.trek -o select.treko
trek -l ./select.treko -e SPI
Loading goals from './select.treko'...
Trek Copyright (C) 2004-2013 Breker Verification Systems http://www.brekersystem
s.com/
trek: info: Initial random seed: 0x9a7ff0db
trek: info: ./select.treko: found 33 goals
trek: info: Elaborating Scenario Model...
Evaluating goal 'SPI'...
trek: info: Rx_data_MSG : Random No assigned to Variable: 0x6c
trek: info: SS4_R_MSG : Random No assigned to Variable: 0x10
trek: info: Ctrl_Reg_R_MSG : Random No assigned to Variable: 0x0
[gowtham@gowtham run]$
```

Figure 3. Values from the graph.

Initially, the graph will select goal for transmit, then from that graph proceeds by selecting either transfer or receive sequence goal. The transfer sequence goal is sequentially process children sequence goal setup for transmission and the leaf goal wait for transmission complete. In setup goal process leaf goal of transmitting data, select goal slave select, leaf goals control and status, enable clock. The leaf goal of transmitting data is to generate a 32 bit the random value for transmission. Sequentially, the select goal of slave select is to select either on of the leaf goals slave select varies from 0 to 8. Each and every leaf goal of slave select has the value of 32 bit value. In this first 8 bit has the value R\W for enabling the slave select register for the transmission of transmitting data. The sequence goal of setup data for transmission has already two process of writing random transmission data to the SPI bus and selecting slave select register for data transmission. In these sequence, the next process is setup the control and status register by setting a 32 bit value 0s according to the register specification. Then final process of setup a clock via leaf goal of enable goal for transmission. After these transmission process completion the leaf goal wait for transmission complete is takes place wait until the transmission process complete.

If the transfer goal select the child goal receive then again the above process takes place reception of data. The given below window shows that compilation of program run by a command run then the compiler found 33 goals and select random seed of memory data and elaborating the scenario model. And then the compiler is evaluating the initial SPI goal

and the compiler will generate the three values either transmission or reception.

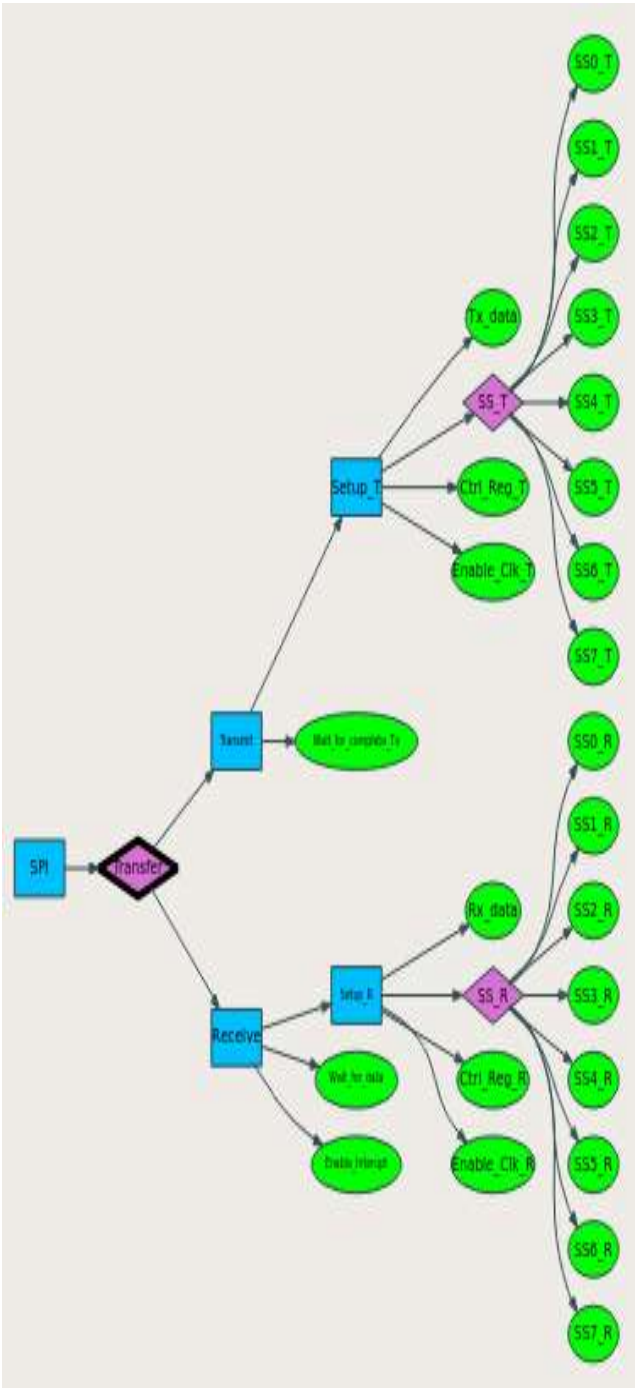

Figure 4. Graph for SPI verification.

VI. CONCLUTION

Serial Peripheral interface protocol has high transmission speed, ease of implementation and with pins advantages. The SPI can connect as many devices as many pins we have on the main microcontroller. The basic functionality and operation of SPI and description of registers, signals, pin is discussed. For SPI model, the graph is generated with some random transmit values and writing register values for slave select, control and status and enable clock using Trek software. The main objective of using Trek software is to generate test cases. So these test cases for SPI can be useful for SOC (which has SPI protocol) verification.

REFERENCES

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

[2]. Deepika Ahlawat, Neeraj Kr. Shukla, "DUT Verification Through an Efficient and Reusable Environment with Optimum Assertion and Functional Coverage in SystemVerilog",(IJACSA),Volume -5,No.4,2014.

[3]. Kiran Kumar.M, AnanthulaSrinivas , "Design and Verification of Serial Peripheral Interface", ISSN: 2321-9939 IJEDR.

[4]. Sandya1.M, Rajasekhar.K, "Design and Verification of Serial Peripheral Interface", International Journal of Engineering Trends and Technology-Volume3Issue4-2012.

[5]. Simon Srot "SPI Master Core Specification", Rev.0.6. March 15, 2004.

[6]. Tianxiang Liu "IP Design of Universal Multiple Devices SPI Interface" IEEE.978-1-61284-632-3, 2011.