Design and Implementation of SPI Bus Protocol with Built-In-Self-Test Capability over FPGA

Shumit Saha, Md. Ashikur Rahman, Amit Thakur
Department of Electronics & Communication Engineering
Khulna University of Engineering & Technology, Khulna, Bangladesh
shumit.ece.kuet@gmail.com, ashik_ece_kuet@yahoo.com, amit_kuet2k8@yahoo.com

***Abstract*— The Serial-Peripheral Interface (SPI) protocol is one of the important bus protocols for connecting with peripheral devices form microprocessor. The complexity of the circuits has aroused with the enormous advancement of IC technology. So, in order to lessen the product failure self-testability in hardware is demanded a lot in recent times. The necessity of self-testability will lead to a solution called Built-in-self-test (BIST). BIST is an effective solution to reduce the huge circuit testing cost. This paper represents designing and implementation of SPI protocol with BIST capability over FPGA. The need of programming for setting up a network with two devices is no longer needed in this proposed system. To accomplish compact, stable and reliable data transmission, the SPI is designed with Verilog HDL language and synthesized on Spartan 2 FPGA. An EEPROM and FPGA Spartan 2 are used for the communication testing where the FPGA is master and EEPROM is a Slave.**

***Keywords*— Serial-Peripheral Interface; Embedded built-in-self-test architecture; Verilog HDL; FPGA**

I. INTRODUCTION

SPI or Serial-Peripheral Interface is a worldwide accepted standard communication protocol. SPI protocol was invented by Motorola. SPI protocol is considered as one of the very best among the systems that are connected to a number of devices and make the communication smooth and fast. SPI as well as others serial protocols such as I2C and 1-wire for instance, are well fitted for data communications from integrated circuits for low or medium data transfer speed to peripherals which are on chip board [1].

Several works have been done using VHDL in designing SPI. A comparison between SPI and I2C Implementation over FPGA is shown in [2]. On that paper, a comparative study of those two protocols on FPGA platform is presented and the entire design has been coded in VHDL. For various controlling purposes SPI is implemented. SPI is presented for motion controller in [3]. FPGA Implementation of SPI of FlexRay Controller is presented in [4].

This paper emphasizes on a new approach of designing SPI with embedded BIST capability using Field Programmable Gate Array (FPGA) technology. Testing of a circuit has become increasingly tough as the scale of integration grows. SPI with the BIST capability provides the specified testability requisites and lowest-price with the highest performance implementation. Much lesser blocks and modules are used to design this SPI so that the testing complexity can be reduced. This system can be fabricated into a single chip. Verilog HDL is used for the coding of this system & designed, tested and evaluated using the ISE 6.0 tool of Xilinx and VeriloggerPro 6.5. For the design implementation the Xilinx Spartan-2 FPGA (XC2S150) is used [8].

The SPI protocol architecture, implementation technique of the system, circuit schematic and simulation results will be discussed briefly in the following sections. The system demands of high integration, low bit error rate and low cost can be satisfied by this SPI.

II. SPI PROTOCOL ARCHITECHTURE

In this paper, the SPI protocol implementation uses four logic signals: SCLK, MOSI (Master Output-Slave Input), MISO (Master Input-Slave Output), SS (Slave Select).SCLK is the clock, a unidirectional bus, which fed into the slave devices. MOSI is defined as output from master which is also known as serial data out. MISO is defined as output from slave which is also known as serial data in. SS is an active low signal which is used to select the slave devices. A full duplex data transmission is occurred in SPI clock cycle. Fig. 1 shows the data transfer system of SPI.

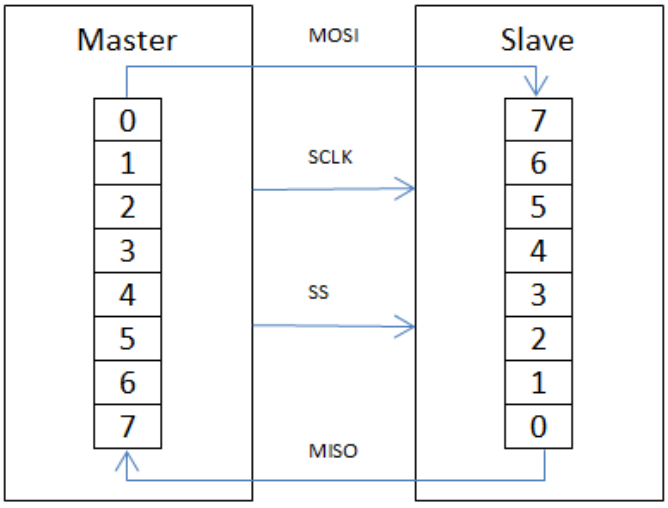

Fig. 1. Data Transfer Type of SPI.

To transfer a data from master device to slave, there are three types of data formats required. Fig. 2 shows the data format of SPI protocol.

978-1-4799-4819-2/14/$31.00 ©2014 IEEE

A. Control Address

It is 8 bit data format. In Fig 2 the first 0 to 7 bits represent the control bus. SPIE is interrupt enable signal which enable the SPI interrupt flag. SPE is the bit for enabling SPI. DORD is used to determine the data order. If DORD is 0 then LSB will be transmitted first. MSTR is used to select the master or slave mode. CPOL & CPHA are clock polarity & clock phase used for determine the shifted edges of MISO & MOSI data. SPR1 & SPR0 are used to determine the clock rate.

B. Status Address

It is also 8 bit data format. In Fig 2 the 8 to 15 number bits represent the status bus. Here, SPIF is the bit used to determine the serial transfer. WCOL is for determine the collision of transfer. Bit 10 to 14 is reserved bit. SPI2X is used to double the clock speed.

C. Data Value

Data values are 8 bit long. In Fig 2 the 16 to 23 number bits represent the data bus. These are the values which transmit from master to slave and vise-versa.

Control	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
Status	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	SPIF	WCOL	-	-	-	-	-	SPI2X
Data	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
	MSB	-	-	-	-	-	-	LSB

Fig. 2. Format of SPI Protocol Bus.

III. PROPOSED ARCHITECTURE

The Proposed structure consists of two modes. One is BIST mode where the SPI can test itself. Another is normal mode. In normal mode the device works like usual SPI protocol.

A. BIST Module

Built-In-Self-Test (BIST) is a design technique where a circuit can test itself. This technique can be easily used in various devices like combinational and sequential logic, memories, multipliers, and other embedded logic blocks. Advanced chip or SOC design is incorporated with large number of core blocks. This is very much difficult to access these chips. So, it is a great challenge to test such embedded chips from outside. Some main challenges among them are the extra testing equipment, cost of testing, level of testing and the testing speed. All these main challenges can be solved by using BIST. The main feature of the BIST system is it gives high speed testing and it can be tested at different test levels. Moreover, no expensive test equipment is needed. Since, BIST is far cheaper than conventional system [5], [7].

Fig. 3 shows the structure of the SPI with BIST. The BIST control signal controls the BIST module. In the BIST module, there are four sub blocks. They are three random pattern generators and a comparator.

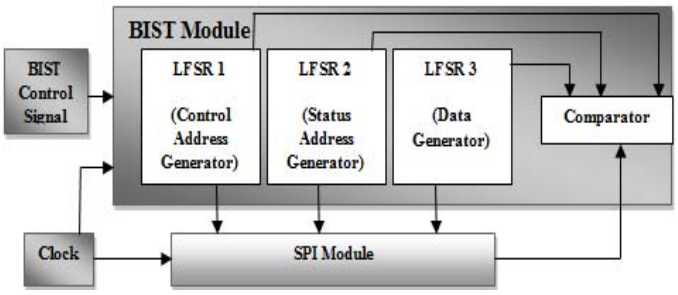

Fig. 3. BIST Structure.

1) Random Pattern Generators (RPG): Random Pattern Generator (RPG) generates random patterns which can be used for the verification of device like SPI. The RPG is a part of the BIST in the verification of the circuits. Many methods have been proposed for the BIST equipment design [6], [9]. To produce bytes to test the circuit the method of a random pattern generator (RPG) is used.

This RPG consists of three LFSRs. LFSR 1 is used to generate the control address. LFSR 2 produces status address. LFSR 3 gives the data. The generated bytes are used directly in the main SPI to obtain better fault coverage. A comparator evaluates the response of the SPI with these bytes.

2) Comparator: This is a comparator which is used to compare the received and transmitted bit pattern. And then it gives the value of error. If the comparator gives bit stream of 101 then the device is perfect and running good. If it gives 001 then there are some faults occur in the protocol.

B. SPI Structure

Fig. 4 shows the basic SPI structure. In Fig 4 it is depicted that there are three registers. They are Control, Status & Data registers. Data register is a shift register. Here, as the data goes from master to slave it is Serial Data Out (SDO) for master and Serial Data In (SDI) for slave. And when the slave register is full, it starts to transmit data to master. And then SDO and SDI are reversed. Master Clock Generator generates the clock and gives it to the slave. The Slave Select Decoder is a decoder controlled by the control register. This slave select decoder selects the slave devices when multiple peripheral devices are needed to be connected. Table I demonstrates the operating modes with reset and reset_n signal. In the table it is seen that, BIST mode is on when reset pin is set low i.e. 0 & reset_n is 1 and vice versa.

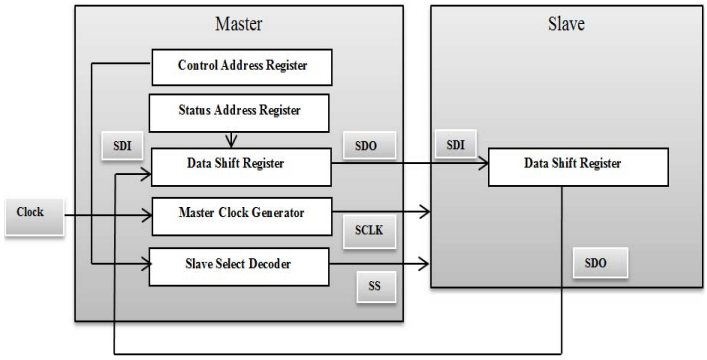

Fig. 4. SPI Module Architecture.

TABLE I. OPERATING MODES OF SPI

reset_n	reset	BIST Mode	NORMAL Mode
0	1	OFF	ON
1	0	ON	OFF

In Fig. 4 the architecture of SPI module is depicted. Here, it is seen that the master is consists of five main modules. Here, three registers are described broadly in previous section. The slave select decoder is used to select the peripheral devices. In Table I, the two main modes are shown. It is shown that when reset = 1 and reset_n = 0, then normal mode begins and master starts to communicate with its slave devices. Afterwards, when reset = 0 and reset_n = 1, then the BIST mode is turned on and the circuit tests itself.

IV. SYSTEM SYNTHESIZE & IMPLEMENTATION

A. Circuit Schematic

Fig. 5 & Fig. 6 show the pin diagram of SPI and SPI with BIST capability respectively. Table II & Table III shows the pin descriptions of the top level schematics of the Verilog HDL implementation shown in the Fig. 5 & Fig 6. In those tables, the input and output pins are also specified.

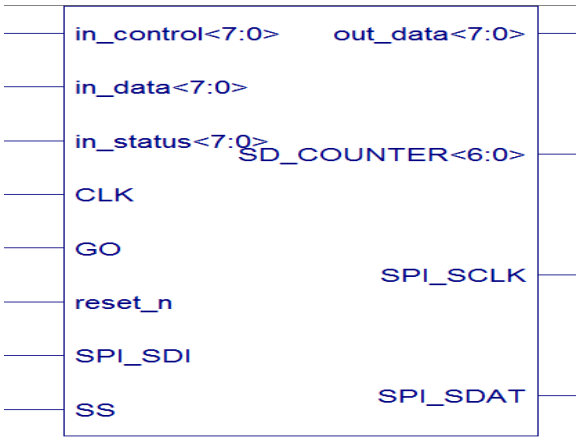

Fig. 5. Pin Diagram of SPI.

TABLE II. Main SPI PIN DESCRIPTION

Pin	IN/OUT	Description
CLK	IN	Clock generator
reset_n	IN	Control Bit for Normal & BIST Mode
GO	IN	Control bit of the SPI
in_data	IN	Input data byte of the SPI
in_control	IN	Input control byte of the SPI
in_address	IN	Input status address of the SPI
Out_data	OUT	Output Data byte of Master
SD_COUNTER	OUT	CLK pulse counter for BIST Mode
SPI_SCLK	OUT	Output pin for SPI CLK
SPI_SDAT	OUT	Output data bus
SPI_SDI	IN	Input data bus

In Fig. 7, the top level schematic of BIST module is shown. As described earlier, BIST module consists of three LFSRs and one comparator is depicted in the Fig. 7.

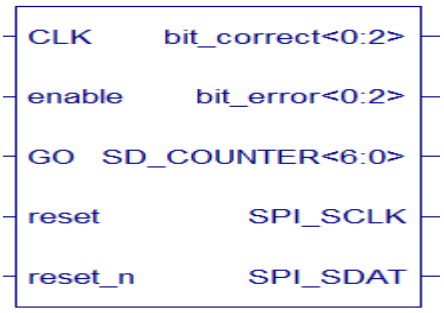

Fig. 6. Pin Diagram of SPI with BIST capability.

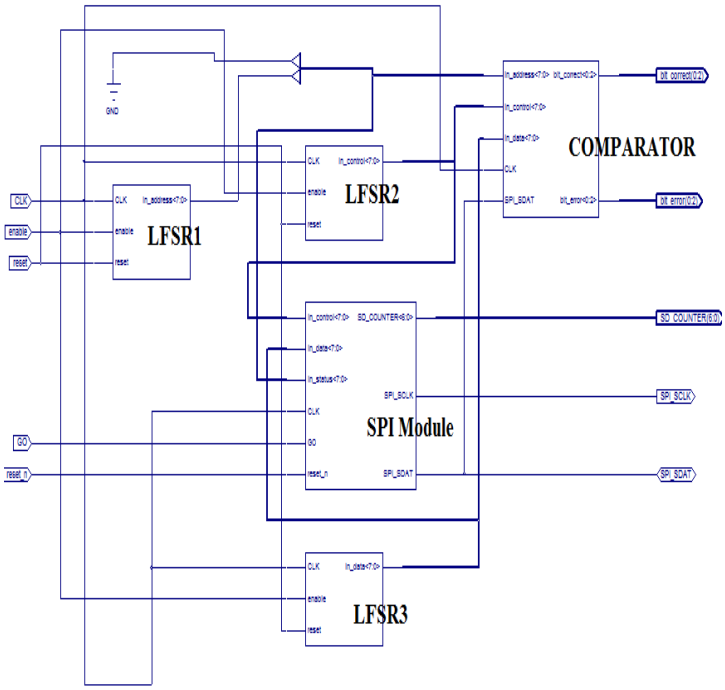

Fig. 7. Top level schematics of SPI with BIST Module.

TABLE III. SPI with BIST PIN DESCRIPTION

Pin	IN/OUT	Description
CLK	IN	Clock generator
reset_n	IN	Control Bit for Normal & BIST Mode
reset	IN	Control Bit for Normal & BIST Mode
enable	IN	Enables the Random Pattern Generator
GO	IN	Control bit of the I2C
SD_COUNTER	OUT	CLK pulse counter for BIST Mode
SPI_SCLK	OUT	Output pin for I2C CLK for BIST Mode
SPI_SDAT	OUT	Output data bus for BIST Mode
bit_correct	OUT	Output pin of Comparator for correct bits
bit_error	OUT	Output pin of Comparator for wrong bits

B. *Simulation Results*

The timing diagrams are achieved from Testbencher (VeriLogger Pro 6.5). The design is tested in the Xilinx FPGA where it also gave the correct output. In the timing diagram, the 8 bits of outputs are converted here into 2-digits Hexadecimal numbers.

1) *Simulation Results for BIST Mode*

a) *LFSR 8-bit Random Bit Pattern Generator*

Fig. 8 demonstrates the output of the LFSR 1 in random pattern generator. Same type of outputs also come from the LFSR 2 and 3. These ouputs from LFSRs are directly goes into the main SPI module.

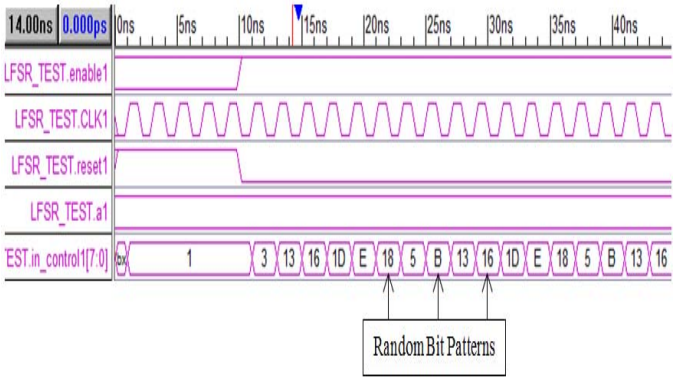

Fig. 8. LFSR output.

b) *Comparator Module Output*

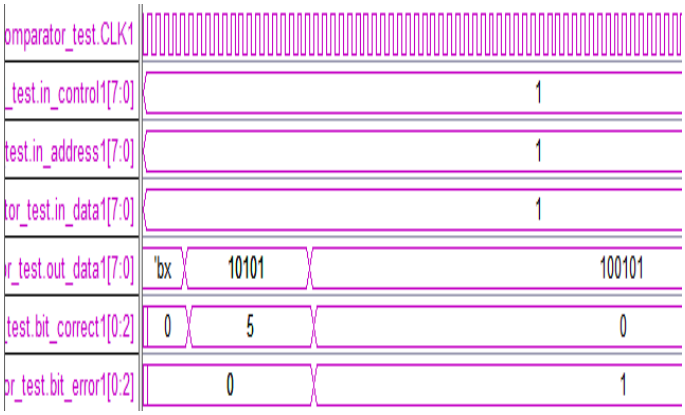

Fig. 9. Comparator Output.

Fig. 9 shows the timing diagram of the comparator module. In Table III, the comparator output bits are depicted. When the control, status and data bits are same as the output data of SPI module, the bit correct signal form comparator module is on. Here, the bit correct signature value is 101. In case of an occurrence of an error in the output stream of SPI, the bit_error signal is turned on then. The signature value of bit error signal is set as 001. From Fig. 9, it is shown that when the all the control, address and data bits are set as 1, then the data stream is 010101. So, from SPI module, if the output data is 10101 then, the signature value should be 101 which mean no error in the data. Afterwards, when the output stream changed to 100101, there is an error occur. And the bit_error line goes to 001. So, form these two signature values, the error can be easily tested. In this process, the efficiency and bit error rate of the SPI can be self-tested.

Table III. COMPARATOR OUTPUT FORMAT

Comparator Output	Signature Value	Hexadecimal Value
Bit_Correct	101	5
Bit_Error	001	1

2) *Simulation Results NORMAL Mode*

Fig. 10 depicts the output of the SPI bus at "NORMAL Mode". When the signal "GO" is high, the SD_Counter starts counting. The Control address, Status Address and Data in hexadecimal are "14", "00" and "AA" respectively which are found in the SPI bus as "00010100", "00000000" and "10101010" respectively when the "reset_n" and "SS" are low. By the SPI_SDAT line it is clearly seen that, the given data are accurately obtained. SD_Counter values from 3 to A represent the first 8 bits which are control bits. Then 1 bit is left intentionally blank to reduce the data collision. SD_counter values from C to 13 & 15 to 1C are showed status address & data byte respectively. This output is justified by Table IV.

Table IV. RECEIVER OUTPUT FORMAT

Input Type	Binary	Hexadecimal
Control Byte	00010100	14
Status Address	00000000	00
Data	10101010	AA

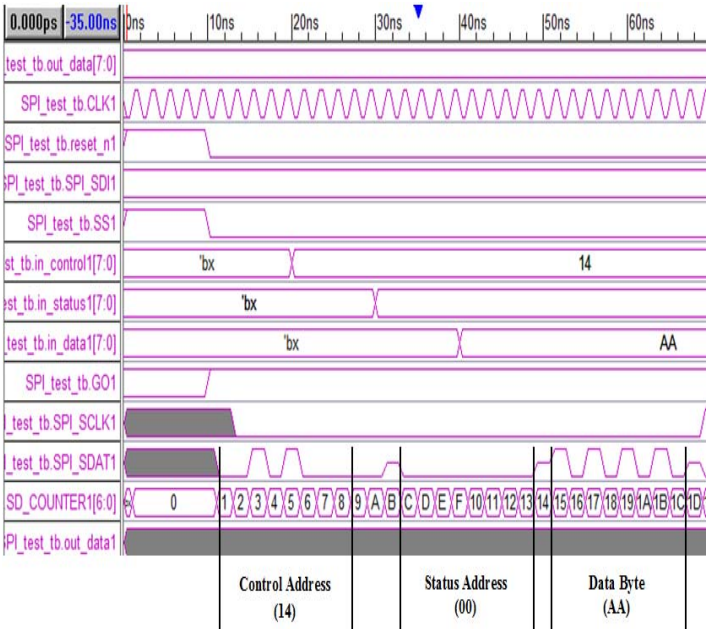

Fig. 10. Normal Mode Output Stream.

Fig. 11 shows, the serial data input into the master form slave. When the "SS" line goes high, that means there is no slave is selected. So, master is now receiving data form the slave. This data comes through the SPI_SDI data line. And the output is shown by the out data line. In Fig. 11, it is seen that the output data line is exactly same as the SPI_SDI line.

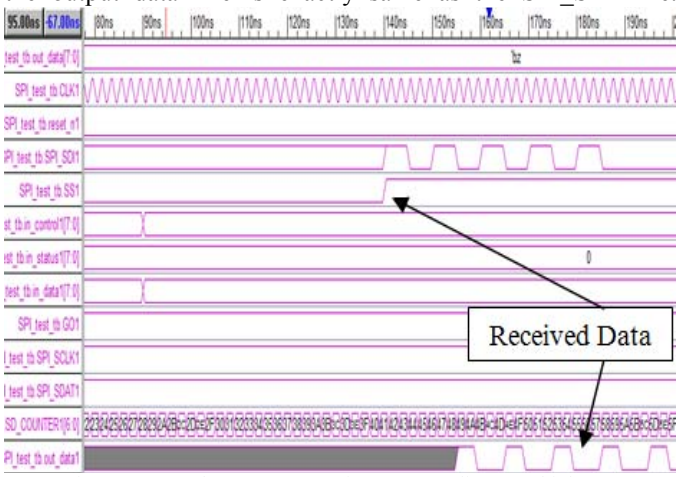

Fig. 11. SPI When Master Receive Data (SS line goes high)

C. FPGA Implementation

Any logical function can be implemented by the Field Programmable Gate Array (FPGA) and it should also be noted that FPGA design is more cost-effective than that of ASIC Design. They have lots of advantages over microcontrollers, such as greater speed, number of I/O ports and performance.

The proposed design is implemented on Xilinx Spartan-2 FPGA (XC2S150). So here, the master device is Xilinx Spartan-2 FPGA. The slave device used here is EEPROM in FPGA. Here, EEPROM means electrically erasable programmable read only memory. EEPROM is a non-volatile memory. It is used as a slave device to store small amounts of configuration information. Fig. 12 shows the FPGA implementation with Clock and Data buses.

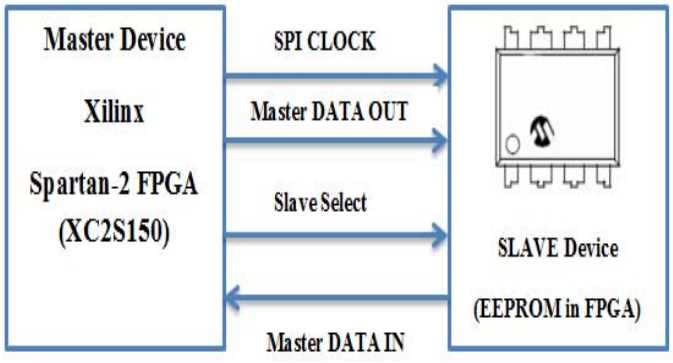

Fig. 12. FPGA Implementation of SPI Bus Protocol.

The Device utilization summary and timing summary are given in Table V and Table VI respectively. Both the tables are divided into two parts, Main SPI module and SPI with BIST capability. From the Table V, it is seen that total number of slices is only 9% for main SPI module and it creeps to 13% in case of SPI with BIST. The usage of input output buffers are only 26% & 20% respectively. So that, there are many more feathers can be added with the proposed architecture. In the timing summary, it is seen that the maximum delay for main SPI module is 7.491ns which is much lesser than conventional SPIs. In SPI with BIST, the delay is 7.782ns which is also nominal. Actually, the maximum delay is depended upon the Gate Delays. So, if the number of gates can be reduced, the delay will be much smaller. In the proposed design, the logic delay is 4.038ns and the route delay is 3.744ns. In future, we try to reduce this logic delay as low as possible.

Table V. DEVICE UTILIZATION SUMMARY

Name	Main SPI Module		SPI with BIST	
	Used Blocks	Percentages (%)	Used Blocks	Percentages (%)
Number of Slices	18 out of 192	9	26 out of 192	13
Number of Slice Flip Flops	19 out of 384 (FDRE:7 FDSE: 12)	4	34 out of 384 (FDR: 3 FDRE:28 FDSE :3)	8
Number of 4 input LUTs	33 out of 384	8	20 out of 384	5
Number of bonded IOBs	24 out of 90 (IBUF :7 OBUF : 16 OBUFT: 1)	26	18 out of 90 (IBUF : 4 OBUF : 14)	20
Number of GCLKs	1 out of 4	25	1 out of 4	25

Table VI. TIMING SUMMARY FOR MAIN SPI MODULE

Parameters	Main SPI Module	SPI with BIST
	Seconds	Seconds
Minimum period	7.491ns	7.782ns
Minimum input arrival time before clock	6.981ns	4.722ns
Maximum output required time after clock	7.913ns	6.959ns
Maximum delay	7.491ns	7.782ns

V. CONCLUSION

In this paper, an FPGA based implementation of SPI with BIST capability is presented. Here all the modules are designed and simulated with Verilog HDL. Then the system is downloaded in the Xilinx Spartan-2 FPGA (XC2S150). This SPI is much more flexible, speedy, low cost, and stable with respect to conventional one. This SPI control bus architecture can enable the industrial fabrication of chip in a way where only a pressing of one switch can test itself. So that, it would save valuable time and cost of testing circuits significantly.

References

[1] F. Leens, "An introduction to I^2C and SPI protocols," IEEE Instrumentation & Measurement Magazine, vol.12, no.1, pp.8-13, February 2009.

[2] A. K. Oudjida, M. L. Berrandjia, R. Tiar, A. Liacha, K. Tahraoui, "FPGA implementation of I^2C & SPI protocols: A comparative study," in Proc. *16th IEEE International Conference on Electronics, Circuits, and Systems,* pp.507- 510, Dec. 2009.

[3] N.Q. B. M. Noor and A. Saparon, "FPGA implementation of high speed serial peripheral interface for motion controller," in Proc. *2012 IEEE Symposium on Industrial Electronics and Applications (ISIEA),* pp.78-83, Sept. 2012.

[4] A.N. Gaidhane and M.P. Khorgade, "FPGA Implementation of Serial Peripheral Interface of FlexRay Controller," in Proc. *13th International Conference on Computer Modelling and Simulation (UKSim),* pp.128-132, Apr. 2011.

[5] M. Bushnell and V.D. Agarwal, *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*, Kluwer Academic Publishers, 2000.

[6] S. Jamuna and V.K. Agrawal, "Implementation of BIST structure using VHDL for VLSI circuits," *International Journal of Engineering Science and Technology,* vol. 3, no. 6, pp. 5041-5048, June 2011.

[7] V.K. Agrawal, C.R. Kime, K.K., Saluja, "A tutorial on BIST, part 1: Principles," *IEEE Design & Test of Computers*, vol. 10, No.1, pp.73-83, March 1993.

[8] J. Bhasker, *Verilog® HDL synthesis: a practical primer*, Star Galaxy Publishing, 1998.

[9] M.Y.I. Idris and M. Yaacob, "A VHDL implementation of BIST technique in UART design," in Proc. *Conference on Convergent Technologies for the Asia-Pacific Region (TENCON),* pp.1450-1454, Oct. 2003.

[10] S. Saha, M. A. Rahman, A. Thakur, "Design and Implementation of a BIST Embedded High Speed RS-422 Utilized UART over FPGA," in Proc. *2013 Fourth International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1-5, Jul. 2013.

[11] S. Saha, M. A. Rahman, A. Thakur, "Design and Implementation of a BIST Embedded Inter-Integrated Circuit Bus Protocol over FPGA," in Proc. *2013 International Conference on Electrical Information and Communication Technology (EICT)*, pp. 1-5, Feb. 2014.