Design of BIST(Built-In-Self-Test)Embedded Master-Slave communication using SPI Protocol

Bitty Jose
Department of Electronics and Communication Engineering
Karunya Institute of Technology and Sciences
Coimbatore, India
bittyjose@karunya.edu.in

J.Samson Immanuel
Department of Electronics and Communication Engineering
Karunya Institute of Technology and Sciences
Coimbatore, India
samsonimmanuel@karunya.edu

2021 3rd International Conference on Signal Processing and Communication (ICPSC) | 978-1-6654-2864-4/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICSPC51351.2021.9451702

***Abstract*—The Serial-Peripheral Interface(SPI) Protocol also called as synchronous serial interface specification is used for communication between single master and single/multiple slaves. With the increase in number of slaves causing high complexity of circuit creates a demand in self testability feature for SPI module in order to test for fault free circuits. Built-In-Self-Test(BIST) is the answer for self-test in circuits as well as it helps in reduction of maintenance and testing cost. Design of BIST embedded SPI module with Single Master and Single Slave configuration has been introduced in this paper, here 8-bit data is transferred across the module ,where the circuit under test(CUT) is being self-tested with BIST feature for it's correctness. This SPI module is designed using Verilog Hardware Description Language(HDL) using EDA playground platform for applications like Application Specific Integrated Circuit(ASIC)or System on Chip(SOC).**

Keywords— SPI,BIST,MISO,MOSI,TPG,ORA

I. INTRODUCTION

Motorola invented Serial Peripheral Interface(SPI) protocol in the mid 1980's to substitute parallel interfaces and provide high speed transfer of data between modules. SPI become most favoured serial communication protocol because of easy interfacing and high speed transfer. SPI follows full duplex, master-slave communication while transferring and receiving data between them which synchronizes on the rising edge or falling edge of the clock. Data transmission can happen at the same time for both master and slave. There are basically two types of SPI interface, it can be either 3-wired or 4-wired. This paper focuses on the popular 4-wired SPI interface.

SPI Protocol is followed even in embedded systems like soc processors and microcontrollers like Programmable Interface Controller(PIC),Advanced Virtual Risc(AVR). These chips may work as master or slave block which is operated by the inbuilt SPI controller in them. SPI is mostly used in those applications where high speed data transfer has to take place.(1)

SPI's exclusive features include master/slave operation, double buffered data register for transmission and reception, polarity and phase synchronization with serial clock, interrupt capability of CPU with fault deduction. SPI's functional registers are responsible for their recognition. The clock polarity(CPOL) and clock phase(CPHA) bits of SPI are used to control the polarity and phase of synchronous clock.(2)

Numerous communication protocols are available for both long distance and short distance communication purpose. For long distance communication we have Universal Serial Bus (USB), Serial Advanced Technology Attachment (SATA), EXPRESS & ETHERNET. While Inter Integrated Circuit(I2C) and SPI protocols are used for short distance communication purposes. SPI has high transmission speed compared to other protocols, and it is simple to use. The SPI module consists of a main SPI clock which is generated by the master when data exchange has been processed. For On-Board communication the processing of data is often reckoned as the "little" protocol communication.(3)

As SPI is an interfacing protocol. The word "Interface" can be elucidated as a common boundary shared by two different components of computer's software or hardware parts in order to exchange information between them. On the contrary, "protocols" are primarily stated as set of rules that are not related to communications of any sort rather bartering of information across blocks to commune and acknowledge each other is what interfacing protocol stands for.(4)

A priority based master slave communication can be designed by making the slaves to generate interrupts from a pin called as interrupt pin .This design can be best in use in places where master has to interact with the connected slave provided with the highest priority for communication. A real time scenario would be where we have a microcontroller as master to which we have two slaves namely an alarm and a sensor connected to the microcontroller .This particular system follows an SPI protocol and here the temperature is monitored regularly till it reaches it's maximum level, an alarm should respond to that threshold value. So accordingly in the given scenario the master has to interact with the slaves on priority basis.(6)

With increase in technology and rapid increase in circuit complexity ,testing of circuit also became very difficult to be executed .In this work self-testability feature has been introduced inside SPI module, where a CUT is introduced within SPI slave and the designed CUT has the ability to self-test itself. Thus, the failure of circuits will reduce as the self testability feature helps in providing fault free circuits.(5)

II. RELATED WORKS

S. Choudhury et al(2014) proposed a configurable Intellectual Property(IP) module for Serial Peripheral Interface(SPI) protocol. The IP module has been designed and verified using Verilog HDL and System Verilog respectively.

Vineeth B, Dr. B. Bala Tripura Sundari (2018) proposed a functional verification plan for SPI protocol using Universal Verification Methodology(UVM) based testbench architecture. A 100% functional coverage has been achieved for the SPI model through systematic implementation of admissible test cases. The proposed UVM testbench generates random responses and does a comparative analysis with the expected response with the help of scoreboard to verify it's functionality. The functional and cross coverage report of this work gives a clear idea that the entire operation of SPI is exercised correctly and the proposed work is best in use for the verification of SoCs featuring SPI protocol .

Pallavi Polsani (2020) presented a SPI model with single master-single slave using independent slave configuration and single-master-multiple slave using daisy chain configuration. The proposed SPI model has been designed using Verilog HDL for synthesis and verified using System Verilog along with the implementation of the model in Spartan 3E.

Muhammad Hafeez, Azilah Saparon (2019) proposed a model for APB interfacing with SPI by designing an IP core for the latter. In the presented work the SPI can transmit or receive data from a connected slave and APB-SPI controller with very good frequency of 16Mhz along with high flexibility of data. The proposed work is simulated in Model sim and synthesized using Quartus lite 16 to produce the gdsii file for tape out. In this work the modelled design is a basic interface with very simple connections to master's IO port and can be utilised for interfacing with APB.

Shumit Saha (2014) presented a BIST capability embedded FPGA implementation of SPI protocol. The entire module has been designed using Verilog HDL and the implementation of the design has been done in Xilinx Spartan-2 FPGA. The proposed SPI model is more advantageous than the conventional SPI model in terms of speed, flexibility, stability and cost. The presented work can benefit fabrication industry as just a click of a switch can test itself .Thus, saving huge time and testing cost.

Deepika, Jayanthi K Murthy (2020) talks about a priority based master-slave interaction used by SPI protocol for communication purpose where the priority to each slave is actually designated by the arbiter block. The design mainly focuses on priority-based transfer of data from a single master to multiple connected slaves. Where, the slaves are selected on priority basis by the master to communicate to that particular slave when the interrupt pin goes high. Here, the proposed design consumes less power as it uses few resources on comparison with interrupt based designs.

III. SPI ARCHITECTURE

SPI is frequently used as a communication protocol by various types of devices. One major benefit of SPI is that any number of data can be transmitted in a continuous flow without any interruption.
All the device communication happening through SPI protocol are in a master-slave arrangement.

Master is the device which helps in generating the main SPI clock with respect to which the data's are synchronized. The frequencies supported by the SPI are much higher than other serial protocols. SPI protocol can only have a single master but in case of slave configuration single to more than one slaves can be connected.
Fig 1 shows connection between a typical SPI Master and SPI Slave blocks. Here, we can see that an active low signal from the master i.e. chip select signal is used to select a connected slave to the master block. When more than one slave blocks are connected to the master block parallelly. Then each slave block will have it's own chip select signal and the signal will be turned high for disconnecting from the master block. There are two data lines for SPI block i.e. MOSI (Master Out Slave In-signal from master to slave) and MISO(Master In Slave Out-signal from slave to master).

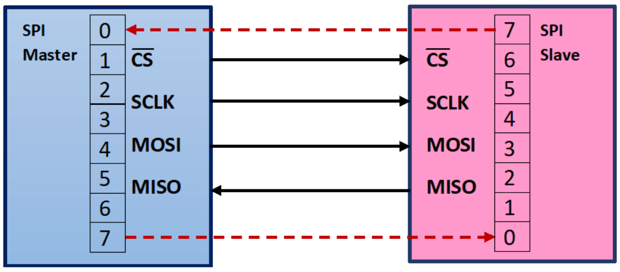


Fig1: SPI with Master-Slave configuration

A. Transfer Operation in SPI

- The master generates the SPI clock which is used for synchronizing the data sent and received.
- Then Master block makes CS pin to low voltage, which in turn selects a slave.
- There are two 8 bit shift registers in SPI module. One in master and other in slave block where the data is shifted from master shift register's MSB to the slave shift register's LSB.
- The Master then sends 8 bit data from the master to the slave one bit at a time along the MOSI line. As they are received the slave reads the sent data bitwise.
- If slave has to communicate by sending data, it will return data one bit at a time from it's shift register's MSB to master's shift register's LSB via MISO pin.

IV. BIST ARCHITECTURE

Built-In-Self Test(BIST) is a technique that is used as a self-testability feature where all the testing functions are placed externally to the CUT ,where the CUT has to undergo testing for checking any faults present in it.

The BIST architecture has namely three important basic blocks connected to the CUT:

- Test Pattern Generator(TPG) block
- Output Response Analyzer(ORA) block
- Test Controller

As shown in Fig2: the test pattern generator block generates random pattern for the CUT. Some of the pattern generators are memory with stored patterns, a counter and linear feedback shift registers(LFSR).

And output response analyzer block consists of a compactor and comparator block. Where, the compactor block uses MISR(Multiple Input Serial Register) technique to compact the responses coming from the CUT. Then these compacted response also known as output signature are then compared to the golden signature stored in a ROM which directs in correction of CUT.

A test controller block is responsible for activation of all tests scheduled to happen as well as analyze those responses.

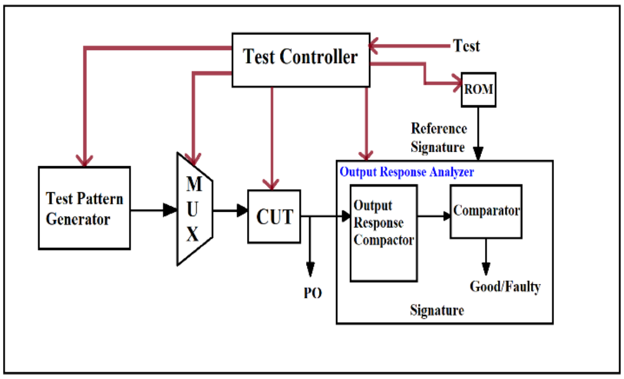


Fig2: BIST architecture block diagram

V. PROPOSED ALGORITHM

In this paper an SPI module having self-testing capability has been introduced. The basic components of BIST architecture has been introduced within SPI architecture block diagram where the designed CUT is able to self-test itself.

The fig3 is the block diagram for BIST embedded SPI protocol , where the test patterns are generated by the TPG block and they are send through Master block's MOSI pin to the Slave.

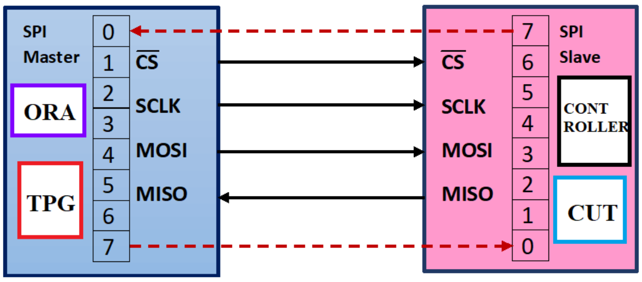


Fig3: BIST embedded SPI Master-Slave block

Where slave reads the data sent from the master and segregates the received data and then finally passes to the CUT designed within the SPI slave block for performing the operation.

After completion of the arithmetic operation. The results from the CUT are sent from the Slave block to the Master block through MISO pin .The received ALU results is then fed to the ORA block designed within the SPI Master for checking the correctness of the CUT.

A. TPG:

The TPG block proposed here uses LFSR(linear feedback shift register) technique to generate test patterns.

Linear feedback shift register is also called as pseudo random pattern generator. An n bit LFSR has maximum sequence length of 2^n-1.

For example if we take n=4 bit LFSR. The test patterns generation can be explained from the fig4.

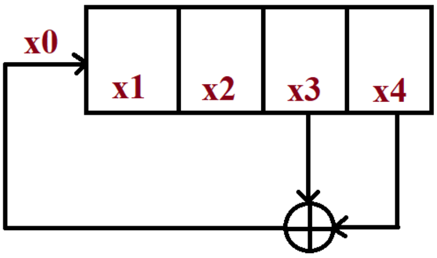


Fig4: 4bit Linear Feedback Shift Register(LFSR)

This 4 bit LFSR consists of a 4 bit shift register along with an exclusive OR gate in the feedback path. From the figure it is seen that 3rd and 4th bit position of the data stream is exored and give as serial input to the first input.

This exoring of 3rd and 4th bit are selected by a primitive polynomial(i.e. $F(x)=x^4+x^3+1$).Here, from the polynomial we take 4th bit and 3rd bit for exoring and it's output is sent to x^0 (i.e. 1). Primitive polynomial is defined as a characteristic polynomial of an LFSR which determines maximum-length of the sequence generation. So for 4-bit LFSR, the number of pattern that can be generated is 15(2^4-1).

Table 1: Pseudo pattern generation for 4-bit LFSR

X1	X2	X3	X4	X3^X4=X0	Clock cycle
1	1	1	1	0	1
0	1	1	1	0	2
0	0	1	1	0	3
0	0	0	1	1	4
1	0	0	0	0	5
0	1	0	0	0	6
0	0	1	0	1	7
1	0	0	1	1	8
1	1	0	0	0	9
0	1	1	0	1	10
1	0	1	1	0	11
0	1	0	1	1	12
1	0	1	0	1	13
1	1	0	1	1	14
1	1	1	0	1	15
1	1	1	1	0	16 (cycle repeats)

From the table it is clearly depicted that the LFSR goes through all possible code permutations except one(in this case it is 0000).This state is known as a locked state and such a state is eliminated due to it's locking nature.

In the proposed work an 8 bit LFSR has been designed which generates 255 pseudo random patterns(2^8-1).

B. ORA:

The ORA block presented in this paper uses MISR(Multiple-Input-Serial-Register) technique for compaction of the responses coming from the CUT . The compacted response is then compared to the golden signature which determines if there is any fault present or not.

MISR operation can be explained from fig5.

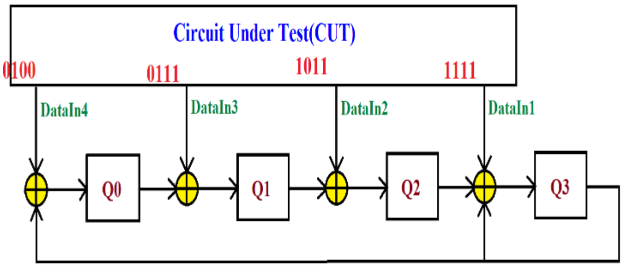


Fig5: Proposed MISR model

Here we can see that four inputs enter into the MISR block from the CUT block for compaction of the responses received. The MISR block consists of 4 exor blocks before each D flip flop. Where the data inputs from the cut is exored one bit at a time. The right most bit from each data Input stream is exored first. Here also it is seen that output of Q3 goes as feedback to first and last exor which also follows the same primitive polynomial.

Table2: Signature analysis for the given inputs

Clock cycle	Q0	Q1	Q2	Q3
0	0	0	0	0
1	0	1	1	1
2	1	1	0	1
3	0	0	1	0
4	0	0	1	0

From the table it is shown how generated inputs from the CUT is compacted into a single output signature.

VI. RESULTS

A. TPG:

In fig6, testpatterns were generated for 8 bit LFSR whose maximum sequence length will be 255 sequences.After which the complete signal becomes high indicating one cycle has been completed

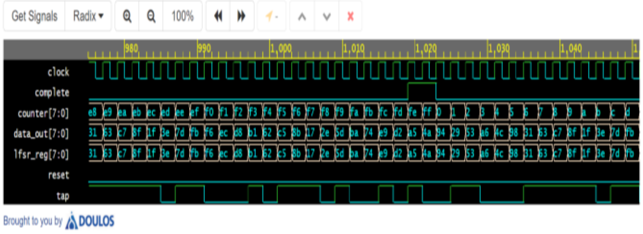


Fig6: Proposed Test Pattern Generator for 8-bit LFSR

B. ALU:

In this paper a simple ALU block is introduced as CUT. Here the data from the LFSR output is taken and fed to the circuit. The LFSR output is segregated and fed as two inputs to CUT which can be seen from fig7.

When select signal becomes low the ALU performs subtraction and when it becomes high, ALU performs addition. These results are then sent to ORA block for analysis of those responses.

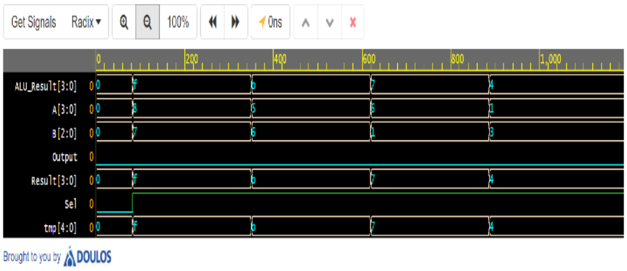


Fig7: Proposed Arithmetic Logic Unit(ALU) for SPI

C. ORA:

The responses from the CUT are then fed to Output response Analyzer block where the compaction of received responses are done. Those compacted responses are then compared to the golden signature.

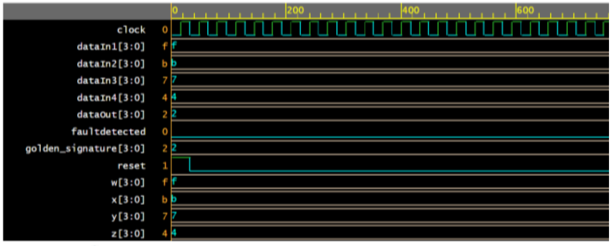
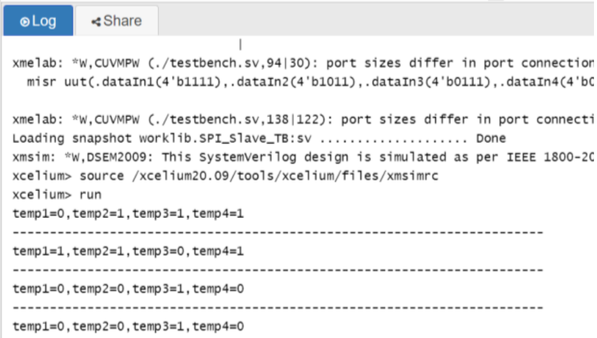


Fig8: Proposed ORA for SPI

From fig8 , it is clear that no faults are detected as the golden signature and data output are matched.



```
xmelab: *W,CUVMPW (./testbench.sv,94|30): port sizes differ in port connectior
  misr uut(.dataIn1(4'b1111),.dataIn2(4'b1011),.dataIn3(4'b0111),.dataIn4(4'bC

xmelab: *W,CUVMPW (./testbench.sv,138|122): port sizes differ in port connecti
Loading snapshot worklib.SPI_Slave_TB:sv .................... Done
xmsim: *W,DSEM2009: This SystemVerilog design is simulated as per IEEE 1800-2C
xcelium> source /xcelium20.09/tools/xcelium/files/xmsimrc
xcelium> run
temp1=0,temp2=1,temp3=1,temp4=1
----------------------------------------------------------------------
temp1=1,temp2=1,temp3=0,temp4=1
----------------------------------------------------------------------
temp1=0,temp2=0,temp3=1,temp4=0
----------------------------------------------------------------------
temp1=0,temp2=0,temp3=1,temp4=0
```

Fig9: Log Window of MISR output sequence

From fig 9, the log window gives us clarity on why the data which is stored as golden signature is 4’b0010.

D. BIST Embedded SPI :

From the below fig10,which depicts the waveform representation of BIST included SPI protocol.

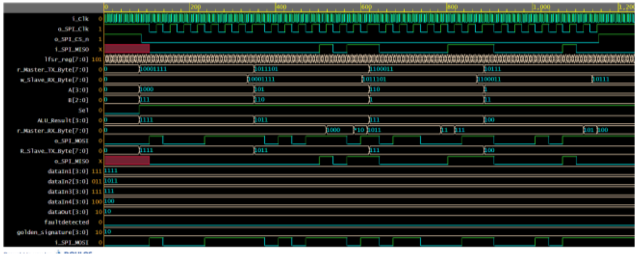


Fig10: Proposed BIST embedded SPI model

Only four numbers are selected randomly from the set of Patterns generated from LFSR. Here we see that the pseudo random patterns generated by the LFSR are fed to CUT through Master’s MOSI pin.

After CUT performs it's operation. It sends to analyzer for compaction and comparison purpose.

The final response of ORA block is checked for any fault present or not.

VII. CONCLUSION

In this paper, a BIST embedded SPI protocol with master-slave configuration has been designed successfully using Verilog HDL and simulated using Cadence Xcelium 20.09 in EDA Playground Platform. The self-testability feature of BIST has been well exercised in the proposed Model. Where, the designed CUT uses SPI protocol for data transfer and has the capability to test itself for checking the correctness of the Circuit under test.

VIII. REFERENCES

[1] Choudhury, G.K.Singh, R.M.Mehra, "Design and Verification Serial PeripheralInterface (SPI) Protocol for Low PowerApplications",International Journal of Innovative Research in Science,Engineering and Technology, Vol. 3, Issue 10, October 2014

[2] Vineeth B, Dr. B. Bala Tripura Sundari, "UVM Based Testbench Architecture for Coverage Driven Functional Verification of SPI Protocol", 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)

[3] PallaviPolsani, V. Priyanka B., Y. Padma Sai, "Design & Verification of Serial Peripheral Interface (SPI) Protocol", International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-6, March 2020

[4] Muhammad Hafeez, AzilahSaparon, " IP Core of Serial Peripheral Interface (SPI) with AMBA APB Interface", IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE),2019

[5] ShumitSaha, Md. Ashikur Rahman, Amit Thakur, "Design and Implementation of SPI Bus Protocol with Built-In-Self-Test Capability over FPGA" International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT) 2014

[6] Deepika, Jayanthi K Murthy "Interrupt Enabled Priority Based Master Slave Communication using SPI Protocol", International Journal of Innovative Technology and Exploring Engineering (IJITEE) , ISSN: 2278-3075, Volume-9 Issue-9, July 2020

[7] Xingchun Liu,Yandan Liu "Multi-functional Serial Communication Interface Design Based on FPGA", 3rd IEEE International Conference on Computer and Communications,2017

[8] Bhagyashri,G.Allawagol,Vidyashree AC,Narendra Kumar,, "Design of SPI IP to communicate with I2C bus of a microcontroller" International Journal of Industrial Electronics and Electrical Engineering, ISSN: 2347-6982, Special Issue, Sep.-2016

[9] Xiaole Cui, Miaomiao Zhang, Qiujun Lin, Xiaoxin Cui, Anqi Pang, "Design and Test of the In-Array Build-In Self-Test Scheme for the Embedded RRAM Array",IEEE ELECTRON DEVICE LETTERS, VOL. 38, NO. 5, MAY 2017

[10] Abhas Singh, Gurram Mahanth Kumar, Abhijit Aasti, "Controller Architecture for Memory BIST Algorithms" IEEE International Students' Conference on Electrical, Electronics and Computer Science,MAY 2020