International Journal of Engineering Research & Management Technology ISSN: 2348-4039

March- 2015 Volume 2, Issue-2

Email: editor@ijermt.org www.ijermt.org

Verilog Implementation of Uart Design with Bist Capability

Mr. M Suhaib Abbasi
Assistant Professor

Prachi Vatsa
PG Student (VLSI Design)

Department of Electronics and Communication Engineering
SRM University NCR Campus
Modinagar

ABSTRACT-
A Universal Asynchronous Receive/Transmit (UART) with BIST capability has the technique of testing the UART on chip itself and no external devices are required to perform the test. The advancement of technology has result in the difficulty of testing the circuit. Universal Asynchronous Receiver Transmitter (UART) is a popular two wire serial communication interface between two microcomputer based systems. Logic BIST applies a large number of test patterns so that more defects can be detected. The UART with BIST technique can be implemented with verilog in order to achieve shorter test time compared to an externally applied test to the circuit, less tester memory requirement and also coast effective.

Keywords: Built-In-Self-Test (BIST), UART, simulation, synthesis.

INTRODUCTION: Testing of integrated circuits (ICs) is important tool in order to decide the quality of a product. In the modern System-on-a-Chip (SoC) Design, many cores are integrated into a single chip. Some of them are embedded and cannot be accessed directly from the outside of the chip. Such SoC designs make the test of these embedded cores become a great challenge. The Built-In-Self-Test (BIST) is one of most popular test solutions to test the embedded cores. The design of the circuit included UART having transmitter, receiver and baud rate generator with BIST having LFSR and MISR. Built-in self-test is a techniques that enable a chip to test itself. In this methodology, test patterns are generated and test responses are analyzed on-chip. BIST Universal Asynchronous Receive/Transmit (UART) has the feature to satisfy specified testability requirements and to generate the lowest-cost with the highest performance implementation. This paper focuses on the design of a UART chip with embedded BIST architecture and problems of Very-Large-Scale-Integrated (VLSI) testing followed by the behaviour of UART circuit using verilog Language.

TESTING PROBLEM IN VLSI

- Cost-efficient testing is needed
- Long test-pattern generation Problem
- The gate to I/O pin ratio problems.
- The Input Combinatorial Problem
- Lack of skilled test engineers
- Diagnosis and repair time
- Maintenance test requires the presence of an expensive ATE at the site of the failing system with significant cost
- Chip/Board Area Cost vs. Tester Cost

UART Architecture-

Universal Asynchronous Receiver Transmitter (UART) is the heart of serial communication**.** Asynchronous serial Communication has advantages of less transmission lines, high reliability and long transmission distance therefore is widely used in data exchange between computer and peripherals. A serial port is one of the most universal parts of a computer. It is a connector where serial line is attached and connected to peripheral devices such as mouse, modem, and printer and even to another computer. In contrast to parallel communication, these peripheral devices communicate using a serial bit stream protocol (where data is sent one bit at a time). The serial port is usually connected to UART, an integrated circuit which handles the conversion between serial and parallel data.

UART allows full-duplex communication in serial link, thus has been widely used in the data communications and control system. UART supports asynchronous communication in which clock information is not shared between transmitter and receiver; several overhead bits are sent along with data bits for synchronization purpose. UART is parallel- to-serial conversion when transmitting, and serial-to-parallel conversion when receiving. The design of UART has Baud Rate Generator (BRG), Transmitter and Receiver as its functional units.

1 BAUD RATE GENERATOR:

The rate at which the data is transmitted is known as Baud Rate. UART receiver operates on the frequency which is 8 or 16 times higher than transmitter. Baud rate generator provides two clocks one for transmitter and another for receiver to maintain data integrity between transmitter and receiver. Baud rate generator is shared between UART transmitter and receiver. The baud rate generator is used to produce local clock signal which is much higher than the baud rate to the UART receiver and transmitter. The baud rate generator is actually a frequency divider. Different values of baud rates are available: ex 300, 1200, 4800, 9600, 19200, 38400, and 57600 etc.

2. TRANSMITTER:

Transmitter takes parallel data and sends it serially on the TxD pin. The transmitter consists of TDR (Transmit Data Register), TSR (Transmit Shift Register) and controller. As load signal goes high transmitter transfers data from TDR to TSR and outputs start bit “0” to the TxD pin then shifts TSR right eight times to transmit 8 bits. When eight data bits transmitted ,transmitter sends parity bit and finally outputs stop bit”1” to the TxD pin and signal “transmitted” goes high.

3. RECEIVER:

Receiver takes data serially in RxD pin, and provides the parallel to the Data out pin. UART receiver consists of RDR (Received Data Register) and controller. The transmitted data from the TXOUT pin is available on the RXIN pin. The received data is applied to the sampling logic block. The receiver timing and control is used for synchronization of clock signal between transmitter and receiver When the UART detects start bit receiver reads and shifts 8 data bits serially into a temporary register. When 8 data bits has been received and parity check passes then after stop bit has been received controller transfers data from Temporary register to RDR and received signal goes high.

Fig. 1 shows how the UART receives a byte of parallel data and converts it to a sequence of voltage to represent 0s and 1s on a single wire (serial). To transfer data on a telephone line, the data must be converted from 0s and 1s to audio tones or sounds (the audio tones are sinusoidal shaped signals). This conversion is performed by a peripheral device called a modem (modulator/demodulator). The modem takes the signal on the single wire and converts it to sounds. At the other end, the modem converts the sound back to voltages, and another UART converts the stream of 0s and 1s back to bytes of parallel data

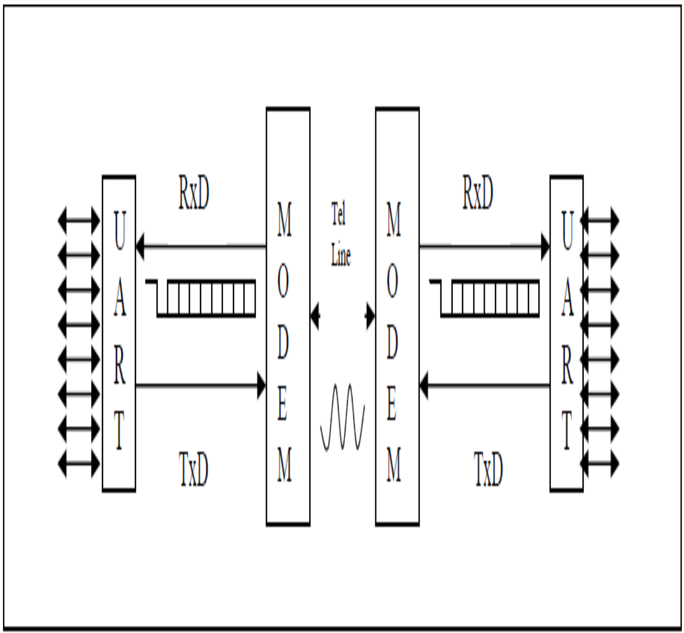


Figure1 Serial Data Transmission and Receive

The design of UART, shown in fig., has Baud Rate Generator (BRG), Transmitter and Receiver as its functional units.

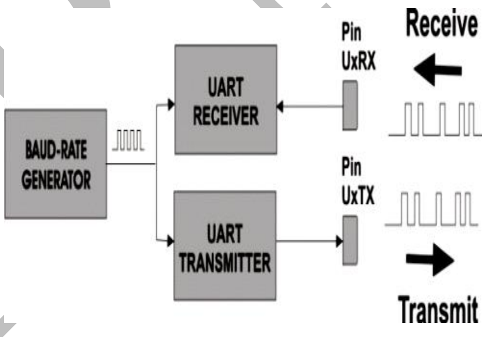


Figure2 UART Architecture

UART WITH BILBO REGISTER AND TESTER

Testing of sequential circuit network problem is simplified by observing the state of all the flip-flops instead of just observing the outputs. For each state of the flip-flops and for each input combination, the network outputs need to be verified. One approach would be to connect the output of each flip-flop within the IC being tested to one of the IC pins. Since the number of pins on the IC is limited, this approach is not practical. The solution to the question is by arranging flip-flops to form a shift register. The state of the flip-flop will be shifted out bit-by-bit using a single serial-output pin on the IC. This is called scan path testing BILBO is a scan register that can be modified to serve as a state register, a pattern generator, a signature register, or a shift register. In summary the BILBO operating modes are presented in Table 1

B1B2	OPERATING MODE
00	SHIFT REGISTER
01	LFSR/PRPG
10	NORMAL
11	MISR

TABLE 1 BILBO OPERATING MODE

Fig. 3 illustrates how to apply BILBO registers to test the UART design. In this structure, “Register A” and “Register B” may be configured by mode control (“bilbo_mode”) signal to act as either a shift register, a test pattern generator (PRPG), normal application mode function (normal) or a data compressor (MISR). The test starts with the initialization of the BILBO by applying a “seed” to its serial-in (si) pin. The initialization can be obtained by configuring BILBO’s operating mode (“bilbo_mode”) to “00” (shift register mode). Following the initialization, the bilbo_mode is set to “01” so that “Register A” is configured as LFSR (“bilbo_mode” = “01”) and Register B as MISR (“bilbo_mode” = “11” (Note: XOR force “01” to “11”)). “Register A” (LFSR) produces an 8-bits pseudo random pattern data in parallel. The parallel data is then fed to the UART’s transmitter. The UART converts the pseudo random parallel data to serial data which is then looped back to its receiver to create an internal diagnostic capability. The UART’s receiver converts the serial data back to parallel and will be accepted by “Register B” (MISR).

A signature will be produced after 255 clock iterations (8 data bits produce 28= 256 PRPG) and this completes the test. The signature is scanned out from serial output (so) pin by configuring bilbo_mode to “00”. Following the scan, it is compared with the correct signature achieved from the simulation of the entire self-test sequence approach in a tester. If the signature produced by MISR is similar to the correct signature, it can be concluded that the UART is working properly

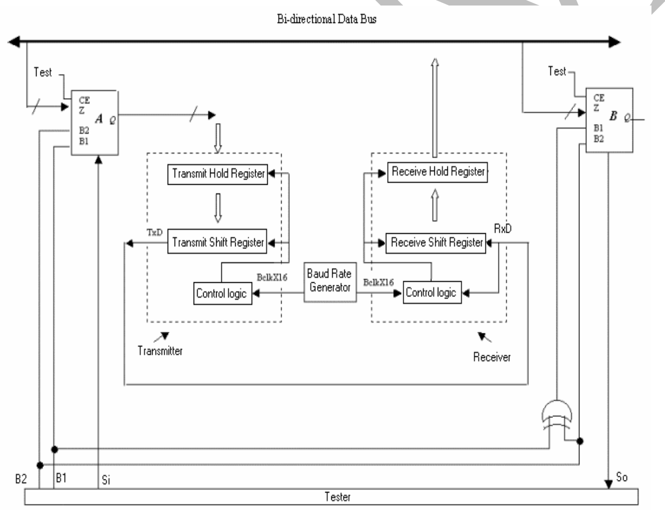


Fig. 3: UART with BILBO register and tester

BIST COMPONENT AND OPERATION:-

a) LFSR (Linear Feedback Shift Register: LFSR is a shift register whose input bit is a linear function of its previous state. The most commonly used linear function of single bits is XOR. Thus, an LFSR is most often a shift register whose input bit is driven by the exclusive-or (XOR) of some bits of the overall shift register value. The

Initial value of the LFSR is called the seed. Because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits which appears random and which has a very long cycle. Applications of LFSRs include generating pseudo-random numbers, pseudo-noise sequences, fast digital counters, and whitening sequences. Both hardware and software implementations of LFSRs are common.

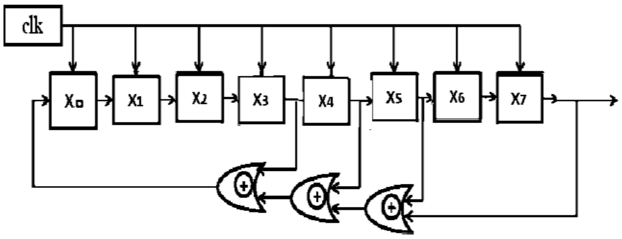


Fig 4 shows 8 bit LFSR

MISR (Multiple Input Signature Register):A Multiple Input Signature-analysis Register (MISR) is one which can be used to reduce the amount of hardware required to compress a multiple bit stream. The MISR provides an alternative to using multiple Linear Feedback Shift Registers (LFSRs) in parallel and separately comparing the error polynomials. Test patterns for BIST can be generated at-speed by an LFSR with only a clock input. Then the outputs of the DUT (Device under Test) must be compared to the known good response which is termed as the golden signature.

In general, collecting each output response and offloading it from the DUT for comparison is too inefficient to be practical. This will consume a huge storage capacity. A reasonable solution is to make some form of data compression on these responses before making the reference comparison. In order to compress the entire output stream into a single signature value, a multiple input signature analysis register can be used. The compressed response is referred to as the signature of the Device under Test. Signature analysis is the most popular compaction technique used today. Multiple input signature register (MISR) is the solution that compact all outputs into a single LFSR. It works because LFSR is linear and obeys superposition

Principle. All responses are superimposed into one LFSR. The final remainder is the XOR sum of remainders of polynomial divisions of each Primary Output by the characteristic polynomial. Its output develops a signature based on the effect of all the bits fed into it. If any bit is wrong, the signature will be different from the expected value and a fault will have been detected.

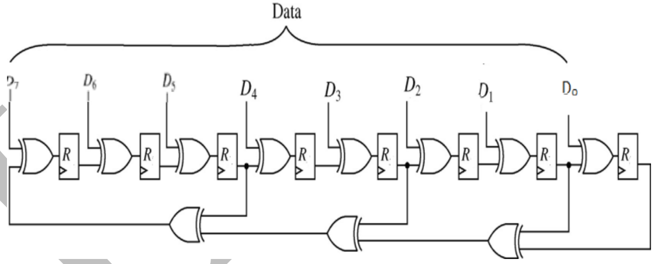


Fig5 shows 8 bit MISR

Simulation Result:

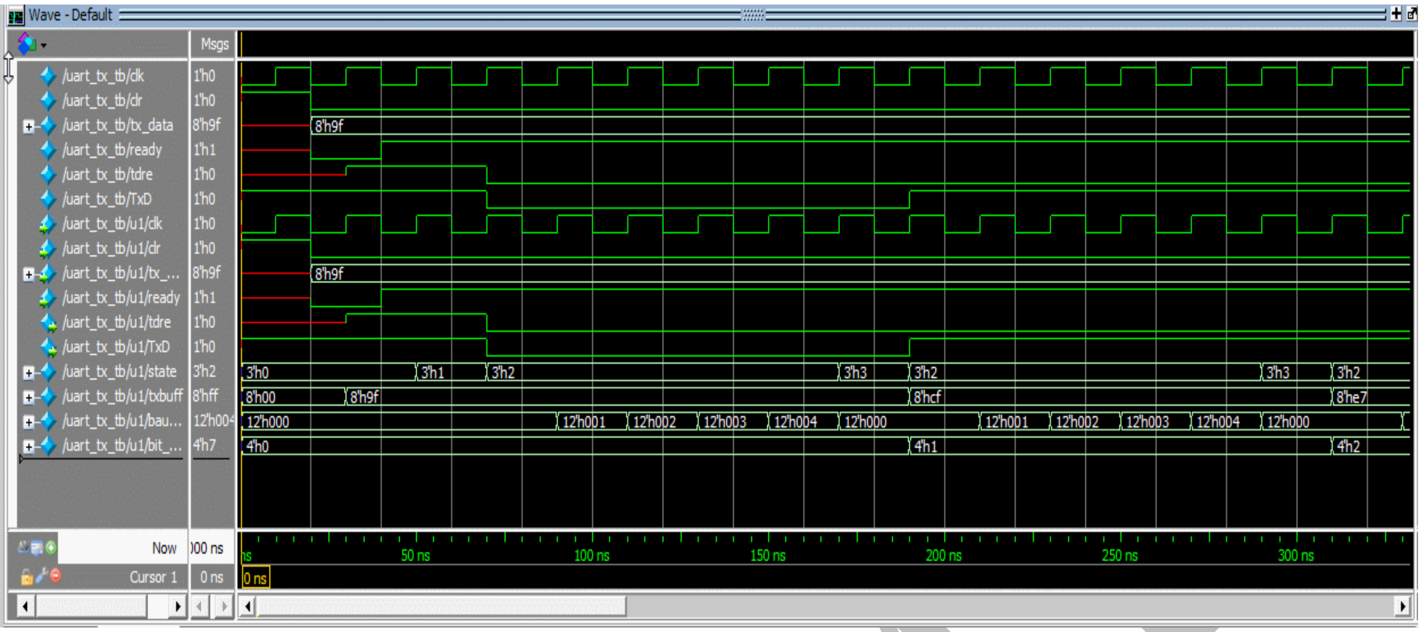

Wave - Default
Msgs
/uart_tx_tb/clk 1'h0
/uart_tx_tb/clr 1'h0
/uart_tx_tb/tx_data 8'h9f
/uart_tx_tb/ready 1'h1
/uart_tx_tb/tdre 1'h0
/uart_tx_tb/TxD 1'h0
/uart_tx_tb/u1/clk 1'h0
/uart_tx_tb/u1/clr 1'h0
/uart_tx_tb/u1/tx_... 8'h9f
/uart_tx_tb/u1/ready 1'h1
/uart_tx_tb/u1/tdre 1'h0
/uart_tx_tb/u1/TxD 1'h0
/uart_tx_tb/u1/state 3'h2
/uart_tx_tb/u1/txbuff 8'hff
/uart_tx_tb/u1/bau... 12'h004
/uart_tx_tb/u1/bit_... 4'h7
8'h9f
3'h0 3'h1 3'h2 3'h3 3'h2 3'h3 3'h2
8'h00 8'h9f 8'hcf 8'he7
12'h000 12'h001 12'h002 12'h003 12'h004 12'h000
4'h0 4'h1 4'h2
Now 300 ns
Cursor 1 0 ns
50 ns 100 ns 150 ns 200 ns 250 ns 300 ns

Fig 6 shows simulation result of transmitter

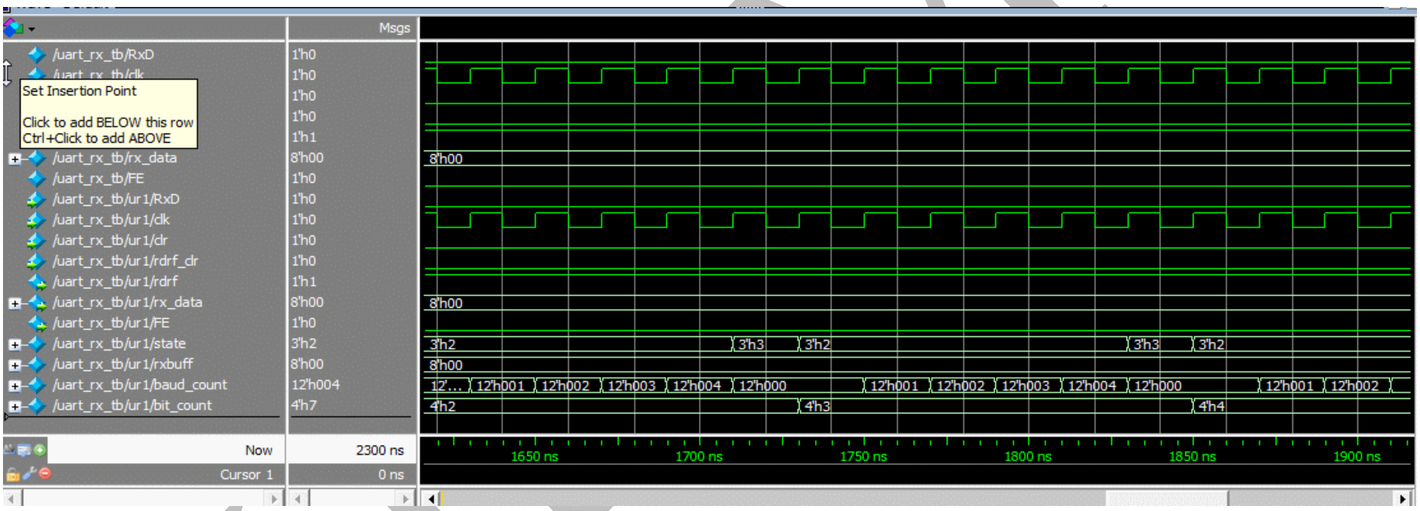

Msgs
/uart_rx_tb/RxD 1'h0
/uart_rx_tb/clk 1'h0
Set Insertion Point
Click to add BELOW this row
Ctrl+Click to add ABOVE
1'h0
1'h0
1'h1
/uart_rx_tb/rx_data 8'h00
/uart_rx_tb/FE 1'h0
/uart_rx_tb/ur1/RxD 1'h0
/uart_rx_tb/ur1/clk 1'h0
/uart_rx_tb/ur1/clr 1'h0
/uart_rx_tb/ur1/rdrf_clr 1'h0
/uart_rx_tb/ur1/rdrf 1'h1
/uart_rx_tb/ur1/rx_data 8'h00
/uart_rx_tb/ur1/FE 1'h0
/uart_rx_tb/ur1/state 3'h2
/uart_rx_tb/ur1/rxbuff 8'h00
/uart_rx_tb/ur1/baud_count 12'h004
/uart_rx_tb/ur1/bit_count 4'h7
3'h2 3'h3 3'h2 3'h3 3'h2
12'... 12'h001 12'h002 12'h003 12'h004 12'h000
4'h2 4'h3 4'h4
Now 2300 ns
Cursor 1 0 ns
1650 ns 1700 ns 1750 ns 1800 ns 1850 ns 1900 ns

Fig 7 shows simulation result of receiver

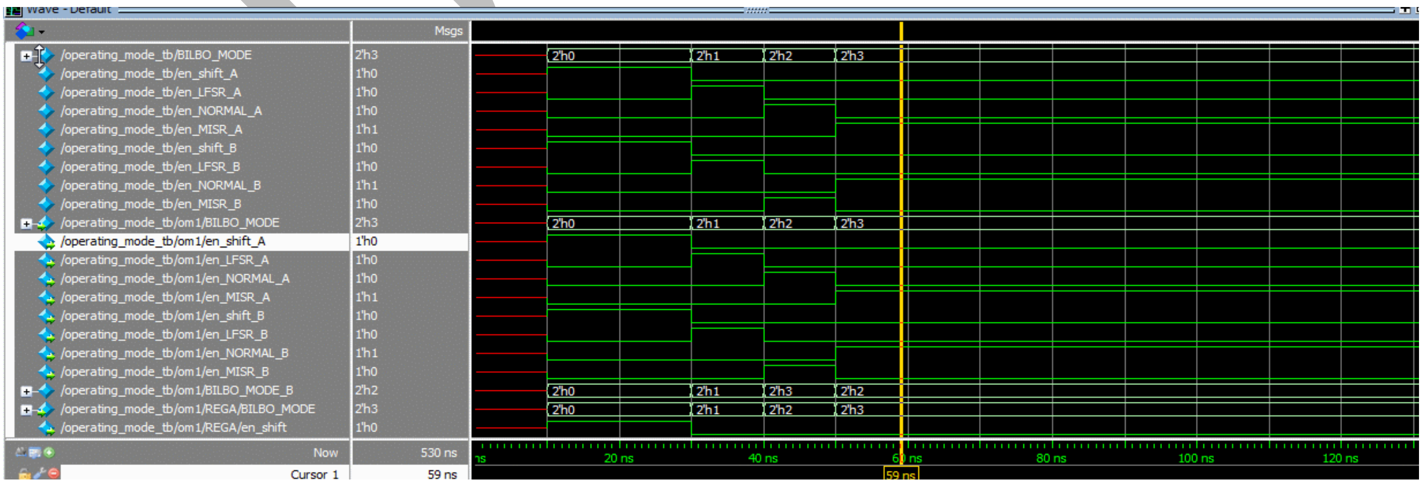
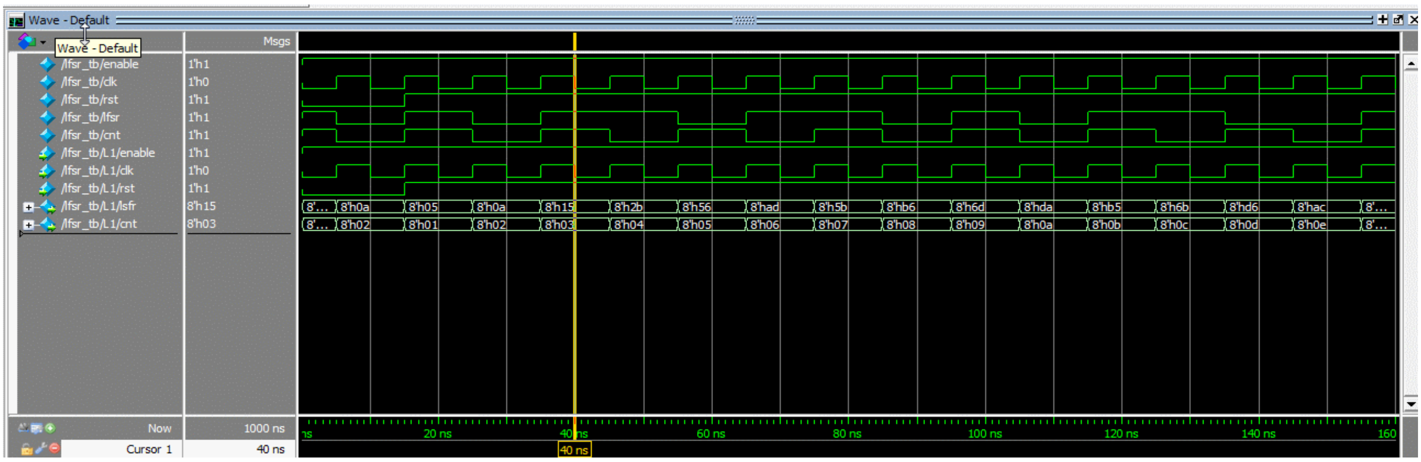

Wave - Default
Msgs
/operating_mode_tb/BILBO_MODE 2'h3
/operating_mode_tb/en_shift_A 1'h0
/operating_mode_tb/en_LFSR_A 1'h0
/operating_mode_tb/en_NORMAL_A 1'h0
/operating_mode_tb/en_MISR_A 1'h1
/operating_mode_tb/en_shift_B 1'h0
/operating_mode_tb/en_LFSR_B 1'h0
/operating_mode_tb/en_NORMAL_B 1'h1
/operating_mode_tb/en_MISR_B 1'h0
/operating_mode_tb/om1/BILBO_MODE 2'h3
/operating_mode_tb/om1/en_shift_A 1'h0
/operating_mode_tb/om1/en_LFSR_A 1'h0
/operating_mode_tb/om1/en_NORMAL_A 1'h0
/operating_mode_tb/om1/en_MISR_A 1'h1
/operating_mode_tb/om1/en_shift_B 1'h0
/operating_mode_tb/om1/en_LFSR_B 1'h0
/operating_mode_tb/om1/en_NORMAL_B 1'h1
/operating_mode_tb/om1/en_MISR_B 1'h0
/operating_mode_tb/om1/BILBO_MODE_B 2'h2
/operating_mode_tb/om1/REGA/BILBO_MODE 2'h3
/operating_mode_tb/om1/REGA/en_shift 1'h0
2'h0 2'h1 2'h2 2'h3
2'h0 2'h1 2'h3 2'h2
Now 530 ns
Cursor 1 59 ns
20 ns 40 ns 60 ns 80 ns 100 ns 120 ns
59 ns

Fig 8 shows simulation result of operating mode



Wave - Default

	Msgs
/lfsr_tb/enable	1'h1
/lfsr_tb/clk	1'h0
/lfsr_tb/rst	1'h1
/lfsr_tb/lfsr	1'h1
/lfsr_tb/cnt	1'h1
/lfsr_tb/L1/enable	1'h1
/lfsr_tb/L1/clk	1'h0
/lfsr_tb/L1/rst	1'h1
/lfsr_tb/L1/lsfr	8'h15
/lfsr_tb/L1/cnt	8'h03
Now	1000 ns
Cursor 1	40 ns

8'… 8'h0a 8'h05 8'h0a 8'h15 8'h2b 8'h56 8'had 8'h5b 8'hb6 8'h6d 8'hda 8'hb5 8'h6b 8'hd6 8'hac 8'…
8'… 8'h02 8'h01 8'h02 8'h03 8'h04 8'h05 8'h06 8'h07 8'h08 8'h09 8'h0a 8'h0b 8'h0c 8'h0d 8'h0e 8'…
ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
160
40 ns

Fig 9 shows simulation result of LFSR

RESULT:

The simulated waveforms presented in this project have proven the reliability of the verilog implementation to describe the characteristics and the architecture of the designed UART with embedded BIST. The simulated waveforms also have shown the observer how long the test result can be achieved by using the BIST technique. In spite of the hardware overhead obtained with BIST implementation, the overhead is somehow reasonable considering the test performance obtained. The research has proven that implementing BIST in a design has effectively satisfied on chip test generation and evaluation. Although the technique was implemented on a low-end device, its usefulness as a testing process has been demonstrated. With the implementation of BIST, expensive tester requirements and testing procedures starting from circuit or logic level to field level testing are minimized.

CONCLUSION:

The process of testing involves driving control signals from an inbuilt controller, generating pseudorandom patterns on the chip using LFSR, passing the patterns through the DUT, compact the responses from the DUT into signatures, passing it through the signature analyzer for analyzing these signatures to match with the golden values and finally giving out the pass or fail signal. For a particular IC, the size is fixed and hence with just the clock signal and the test mode signals the particular IC can be tested. The output is also simplified with just the single signal indicating the pass or fail status after the signature analysis.

Logic BIST is highly efficient with a high fault coverage of above 90-95% and can be seen when combined with a fault coverage estimation tool and the best part of the scheme is that it is at-speed which reduces the interface to the tester, tester memory and tester time.

REFERENCES

1. Bushnell and V.D. Agarwal, "Essentials of Electronic Testing for Digital, Memory and Mixedsignal VLSI Circuits "Kluwer Academic Publishers, 2000.
2. Mohd Yamani Idna Idris "A VHDL Implementation Of UART design with BIST capability" Malaysian Journal of Computer Science, Vol. 19 (1), 2006
3. Z. Navabi, "VHDL Analysis and Modeling of Digital Systems", McGraw-Hill Inc., 1991
4. 2012 International Conference on Communication Systems and Network Technologies "FPGA Implementation of 8, 16 and 32 Bit LFSR with Maximum Length Feedback Polynomial using VHDL".
5. Bhasker J, "A VHDL Primer", PTR Prentice Hall, Pages 1-2, 4-13, 28-30.
6. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-4 Issue-7, December 2014 Design of UART Protocol with Interrupt Logic and Status Register by Salman Khan B. R, Arun Patro, Siva S. Yellampalli