*IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*
*Volume 4, Issue 6, Ver. II (Nov - Dec. 2014), PP 04-10*
*e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197*
*www.iosrjournals.org*

# Power Analysis and Synthesis of BIST Technique on UART

Nishtha Singh[1], Sangeeta Mangesh[2]
[1](Department of ECE, JSS Academy of Technical Education, Noida, India)
[2](Department of ECE, JSS Academy of Technical Education, Noida, India)

***Abstract:*** *The increasing growth of sub – micron technology has resulted in shrinking of the device size leading to increase in device density. In the modern System-on-a-Chip (SoC) design, many cores are integrated into a single chip, some of them are embedded. This increases the functional complexity of the chip. The internal sub – circuits of the chip cannot be accessed directly from the primary inputs of the chips. So the testing of the chip is becoming very time consuming and costly. Such SoC designs make the test of these embedded cores become a great challenge. Thus Automatic Testing Equipments (ATE) is becoming costly process for testing. To reduce the cost of testing the chips, Built in Self Test (BIST) has emerged as an cheaper alternative. BIST is a design technique that allows the chip to test itself. In this paper, the BIST is implemented on UART using Verilog. The simulation and synthesis of the design are performed using ModelSim SE PLUS 6.5 simulator and XILINX ISE 14.5 synthesis tool.*

***Keywords:*** *ATE, BIST, BILBO, Power Analysis, UART*

## I. Introduction

Asynchronous serial communication has advantages of less transmission line, high reliability, and long transmission distance, therefore is widely used in data exchange between computer and peripherals. Asynchronous serial communication is usually implemented by Universal Asynchronous Receiver Transmitter (UART). The universal designation indicates that the data format and transmission speeds are configurable.

Universal Asynchronous Receiver Transmitter (UART) is mostly used for short distance, low speed, low cost data exchange between processor and peripherals. UART allows full duplex serial communication link, and is used in data communication and control system. UARTs are used for asynchronous serial data communication by converting data from parallel to serial at transmitter with some extra overhead bits using shift register and vice versa at receiver. Serial communication reduces the distortion of a signal, therefore makes data transfer between two systems separated in great distance possible. UARTs are commonly used in conjunction with communication standards such as EIA, RS-232, RS-422 or RS-485. It is generally connected between a processor and a peripheral, to the processor the UART appears as an 8-bit read/write parallel port. In actual applications, usually only a few key features of UART are needed. Specific interface chip will cause waste of resources and increased cost. Particularly in the field of electronic design, SOC technology is recently becoming increasingly mature. This situation results in the requirement of realizing the whole system function in a single or a very few chips.

There is a need for realizing the UART function in a single or a very few chips. Further, design systems without full testability are open to the increased possibility of product failures and missed market opportunities. Also, there is a need to ensure the data transfer is error proof. In this paper the introduction of Built-in self test (BIST) to UART is utilised to overcome the above two constraints of testability and data integrity [1]. A proper designed Built-In-Self-Test (BIST) is able to offset the cost of added test hardware while at the same time ensuring the reliability, testability and reduces maintenance cost. BIST uses embedded hardware test generators and test response analyzers to generate and apply test patterns on-chip at the speed of the circuit, thereby eliminating the need for an external tester. The technique proposed for the BIST architecture in this paper is built in logic block observer (BILBO).

## II. Operation of UART

UART is a serial communication protocol. It is used for asynchronous serial data communication. The term asynchronous is used because the transmitter and receiver do not share the same clock signal. It is used for short distance, low cost, low speed serial communication between the computer and the peripherals. It is mostly used in conjunction with RS232. It transmits data serially. The architecture of UART is shown in Fig 1 [2]. The main modules of UART are as follows:

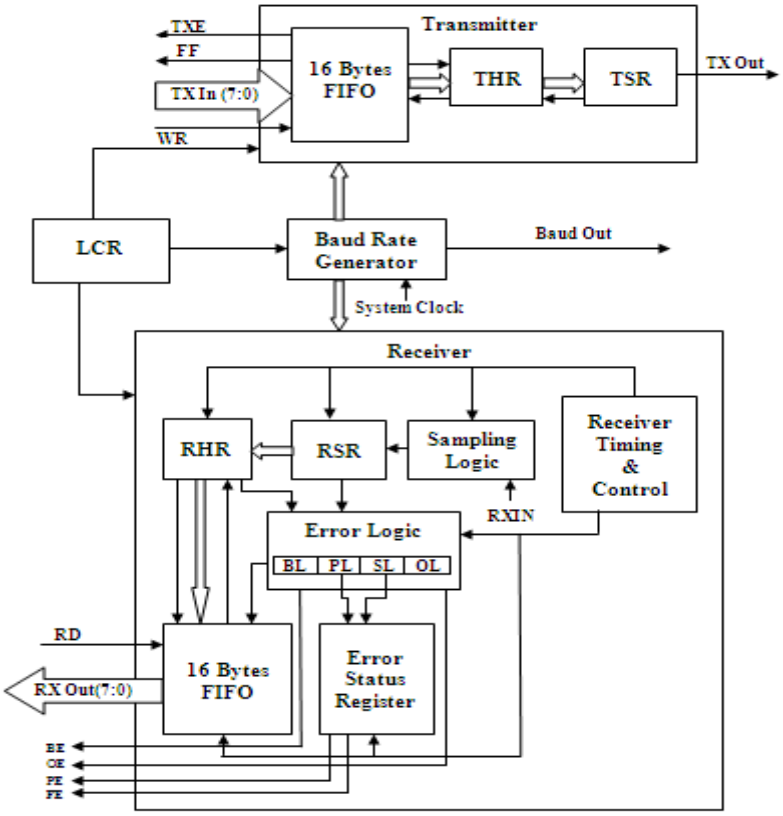



**Fig 1:** architecture of UART

### 1.1. Line Control Register

LCR is a byte register and is used for specifying the frame format precisely. The stop bit, parity bit, data word length; baud rate selection can be controlled by using the respective bits in the LCR. The LCR format is shown in fig 2 [2].

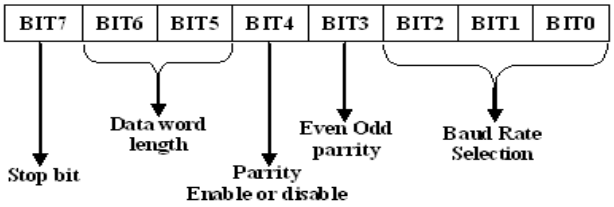



**Fig 2:** LCR format

### 1.2. Baud Rate Generator

It is basically a frequency divider. The system clock used is divided by a specified baud rate to get a baud clock. Here 16 is a divisor value. The value of divisor can be changed to get different baud clock at different frequency. The formula for calculation of baud clock is:

$$\text{baud clock} = \frac{\text{sysclk}}{\text{baud rate}} * \frac{1}{16} \quad (1)$$

### 1.3. Transmitter Module

The function of a transmitter is to convert the 8 bit parallel data serially. It accepts a parallel byte data and converts it into a serial form and then transmits data serially bit by bit. In this module, the transmitter module has three sub modules: the 16 bytes fifo for storing the data bytes, transmitter hold register for holding the data and the transmitter shift register (TSR) for shifting the data bits.

### 1.3.1. 16 bytes transmitter FIFO

The basic function of the fifo is to act as a storage memory. The data bytes to be sent are stored in the fifo. The principle of FIFO is as the name suggests first in first out. The data written first in the fifo is read first and the process goes on. The control signals used in this design are as follows:

buff_full: when this signal is high the fifo buffer is full.
buff_empty: when this signal is high the fifo buffer is empty.
w_flag: when this signal is high the fifo is not full and the fifo is ready to accept data bytes.
wr_en: when this signal is high the data bytes can be written into the fifo.
rd_en: when this signal is high the data can be read from the fifo.
wr_ptr: this is a write pointer. It points to the memory location at which the data can be written. Whenever a data is written the wr_ptr is increamented by 1 and points to the next memory location.
rd_ptr: this is a read pointer. It poits to the memory location from which the data can be read. Whenever a data is read, the rd_ptr is increamented by 1 and poits to the next location.

When the w_flag is high and the wr_en is high and the buff_full is low, the data can be written into the fifo. Suppose the fifo is empty, then the wr_ptr points to the $0^{th}$ memory location and data is written at the $0^{th}$ memory location. After the data is written the wr_ptr is increamented and points to the next memory location.

### 1.3.2. TSR

It is used for shifting the data bits to the output pin (out_tsr). The control signal used is ready_thr when this is high the data bits are fetched from the fifo and stored into a register data_tsr. In this module the data frame is formed and stored in a temporary register temp_tsr. The data frame consists of a start bit (assigned '0'), 8 bit data bits, a parity bit and a stop bit (assigned '1'). After the data frame is formed the 12 bit data is then serially shifted out. The frame format is shown in Fig 3. The UART data frame is composed of a start bit of one bit-length logic 0, 5 to 8 bit-length data bits, parity bit and a stop bit of one, two bit-length.

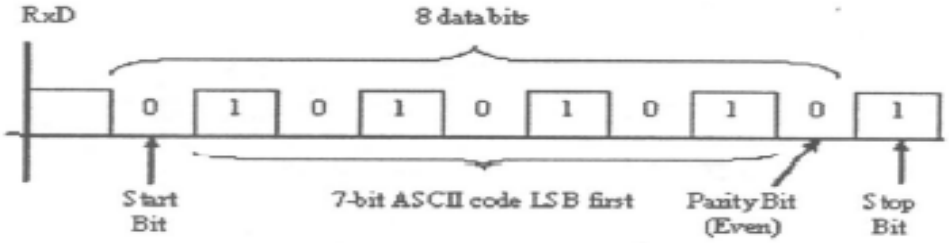



**Fig 3:** frame format of UART

### 1.4. Receiver Module

The function of the receiver module is to convert the serial data bits from the transmitter to 8 bit parallel data. It accepts the serially shifted data bits at the transmitter output and converts them into a 8 bit data bits. The reception of the data bits and sending the 8 bit data at the output of the receiver completes the transmission process. 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. It consists of three sub modules: receiver shift register (RSR), error logic and the 16 bytes fifo.

### 1.4.1. RSR

It is used for accepting the data bits shifted serially from the transmitter and giving a output of 8 bits data at the output pin (out_rsr). The data bits are received at data_rsr pin. When a start bit (i.e '0') is detected, the reception of data bits starts. The received data bits are stored into the temporary register (temp_rsr). After the 8 bits data are stored in temp_rsr, the receiver waits for a stop bit (i.e '1'). When stop bit is received the the 8 bit data stored in temp_rsr is copied to the output pin out_rsr.

### 1.4.2. Error Logic

The error logic block handles four types of errors: parity error, frame error, overrun error and break error. If the received parity does not match with the parity generated from data bits PL bit will be set which indicates that parity error occurred. If receiver fails to detect correct stop bit or when 4 samples do not match frame error occurs and SL bit is set. If the receiver FIFO is full and other data arrives at the RHR overrun error occurs and OL bit is set. If the RXIN pin is held low for long time than the frame time then there is a break in received data and break error occurs and BL bit is set.

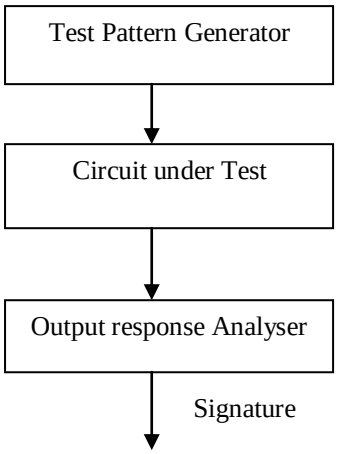
### 1.4.3. 16 bytes receiver fifo

The function of the receiver fifo is to store the data bytes received from out_rsr. It is a 16 bytes fifo. The control signals used in this fifo are also same the ones used in the transmitter fifo. The data byte is read from the fifo through the output pin buff_out.

Hence the data transmitted is received at the receiver end. This whole process completes the transmission and reception of the data.

## III. Built In Self Test

Built in self test is a design for testability technique that places the circuits on the chip as pattern generators and verifiers. It is a better alternative for automatic test pattern generators. The test generation and verification are done by the circuits built into the chip. The basic BIST architecture is shown in figure that has three components: test pattern generator, circuit under test and an output response analyzer [3].




There are number of architectures in which BIST can be implemented. In this paper one of the architecture is discussed in detail i.e. Built in Logic Block Observer (BILBO). The BILBO tcchnique has been recognized as a method that can help to reduce the test and maintenance cost of chip production. It offers advantages concerning fault covcrage, detection of delay faults, and test application time. BILBO is a scan register that can be modified to serve as a shift register, pattern generator or a signature register. The pattern generator for BIST can be implemented by storing the test vectors in memory and retrieving them during test mode, but this approach requires a relatively large amount of memory. An alternative is to use linear feedback shift register (LFSR) as the test pattern generator. Multiple input signature register (MISR) is used as the response analyzer. Hence BILBO operates in four modes i.e. shift register mode (00), pattern generator mode (01), normal mode (10) and signature register mode (11). The four modes are explained below:

Shift Register Mode: In this mode the register acts as a simple shift register. It accepts the input data from the user. The seed to be used for initialising the LFSR is also scanned in using this mode.

LFSR Mode: In this mode the patterns are generated using LFSR. It is a shift register whose input bit is a linear function of the previous state. The most commonly used linear function is XOR. This means that LFSR has feedback exclusive-ORed from selected stages called as taps along the register to form the serial input to the first stage. Thus LFSR is a shift register whose input bit is driven by the exclusive-or of some bits of overall shift register. The initial value of LFSR is called as the seed. The register has finite number of states given by $2^{(n-1)}$, where n is number of bits. The behaviour of an LFSR is determined by its initial seed and the feedback coefficients. When feedback coefficients are efficiently chosen an n-stage LFSR will count in pseudorandom manner through $2^{(n-1)}$ states before repeating its sequence, this is known as maximum length pseudorandom sequence [4]. A standard n - stage LFSR is shown in Fig 4.

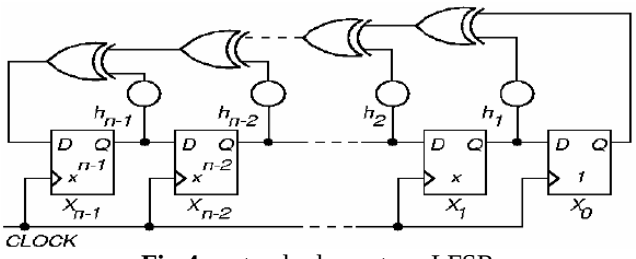



**Fig 4:** a standard n – stage LFSR

MISR Mode: This mode is used to generate a final compressed signature. In BILBO there are many test response inputs as there are stages in the BILBO register giving a multiple input signature register configuration. Hence as the BILBO input register applies its sequence of input test vectors, the output BILBO register simultaneously clocks through a random sequence depending upon the parallel – input test response data, giving a final test signature at the end of the sequence. The MISR register is same as that of the LFSR register.

## IV. UART With BIST

The hardware architecture of implementation of UART with BILBO is shown in Fig 5. It has three main blocks: register A, UART and register B. Register A and register B act as the BILBO register. They can be configured into scan register, test pattern generator, signature analyser and normal application mode. The operating modes for BILBO are present in table [5].

Operation modes of UART

| **B1 B2** | **Operating Modes** |
|---|---|
| 00 | Scan register |
| 01 | TPG (LFSR) |
| 10 | Normal mode |
| 11 | SA (MISR) |

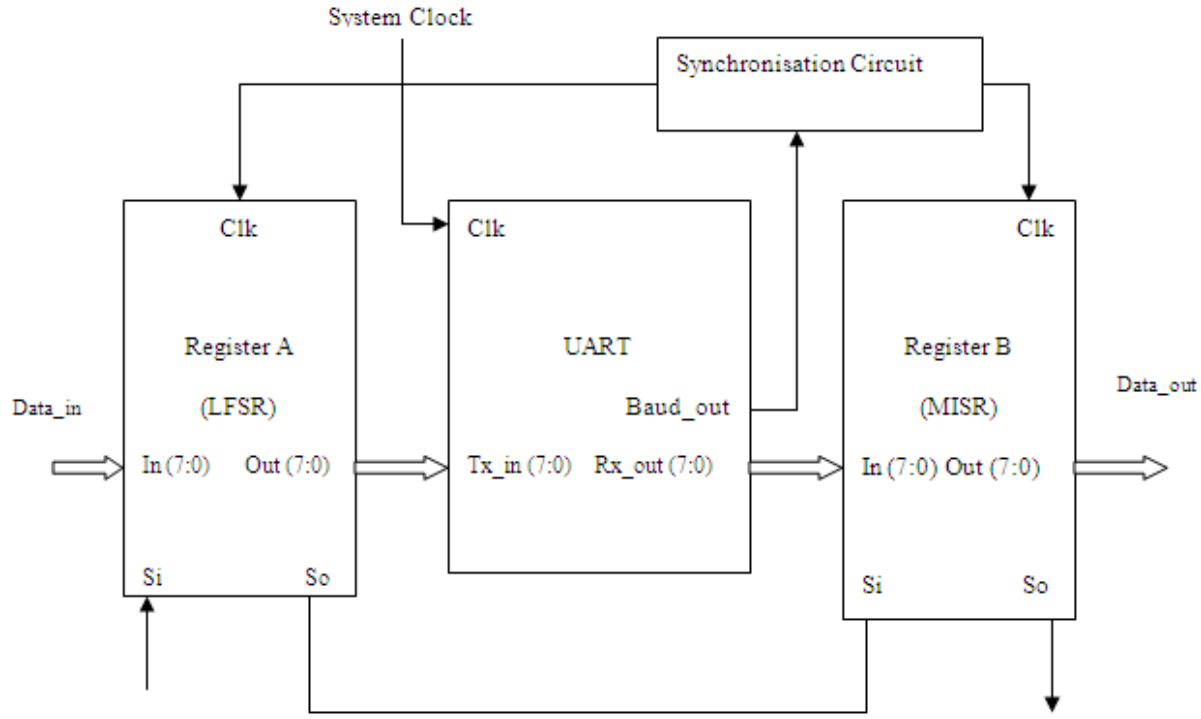



**Fig 5:** UART with BILBO registers

In this architecture LFSR is used as the pattern generator and MISR is used as the signature analyser. When we are not testing the UART i.e. the system is not in testing mode then, 10 mode is selected, in which the system operates as a UART and the register A and register B act as a normal 8 bit register. To start the testing procedure, the register A and register B are operated in mode 00 i.e. as the scan register. In this mode the seed to be initialised to the LFSR and MISR is scanned in. It takes 8 cycles to scan in the seed. After the seed is scanned, the register A and register B are configured in LFSR (01) and MISR (11) mode respectively. Register A produces b bit random parallel data which is fed to the transmitter fifo of the UART, from where it is transmitter serially to the receiver which receives it bit by bit and converts it back into the parallel data which is written into the receiver fifo and read by register B. Since the number of bits is 8, thus it takes 255 cycles to complete the testing procedure. After 255 cycles the signature generated by the MISR is scanned out by configuring the register A and register B to scan mode (00). After this the signature is compared by the the signature calculated by the equation [6]:

$$R_m(x) = x \{[I_{m-1}(x) + x\, I_{m-2}(x) + ..... + x_{n-2}\, I_1(x) + x_{n-1}\, I_0(x)] \bmod D(x)\}. \quad (2)$$

If the signature generated matches the signature calculated from the equation (2), then it is concluded that the test was successful.

## V. Simulation and Synthesis Results

The simulations are done using ModelSim SE PLUS 6.5. During BILBO's normal mode ("10") the system acts as a UART. Fig 6 shows the normal mode of operation of the system till 590 ns. After 590 ns, the mode of operation is changed to testing mode, by selecting ctrl = '1'. Now BILBO is in scan mode from 590 ns to 910 ns. In this mode the seed for initialising the LSR is scanned in. The seed scanned in is "0F" for LFSR and "2E" for MISR, as shown in fig 6 at signals bilbo_a/Q and bilbo_b/Q for LFSR and MISR. It takes 8 cycles to scan in the seed.

After 8 cycles, at 950 ns the mode of BILBO A is changed to "01", the LFSR mode and the mode of BILBO B is "11", the MISR mode as shown in fig 7. In LFSR mode the pseudorandom patterns are generated. The pseudorandom patterns are obtained by xoring 1st, 2nd, 3rd, and 7th bits of the seed. The patterns are observed at signal bilbo_a/Q, as shown in fig. In the MISR mode, the process of obtaining the patterns is same as that of the LFSR mode except that the pattern generated by xoring 1st, 2nd, 3rd and 7th bits ids then again xored with the data received at the input of BILBO B. The pattherns are observed at bilbo_b/Q, as shown in fig 7. This process repeats for 255 cycles.

After 255 cycles the signature is generated as shown in fig 8. The signature generated here is "A9"which is same as the signature calculated from the equation (2). Thus testing is successful.

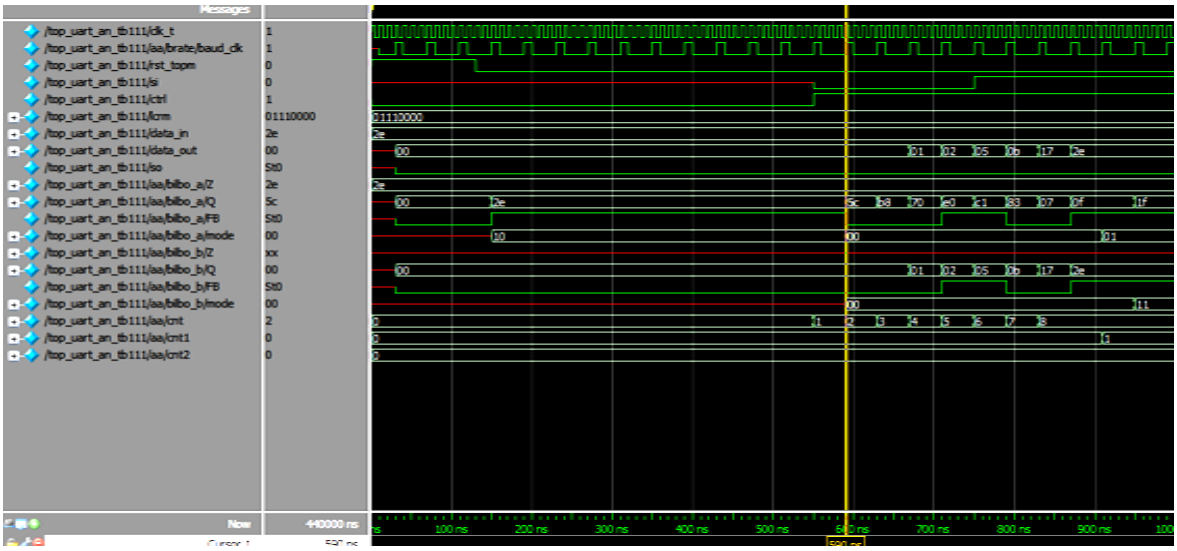


**Fig 6:** BILBO registers in scan mode

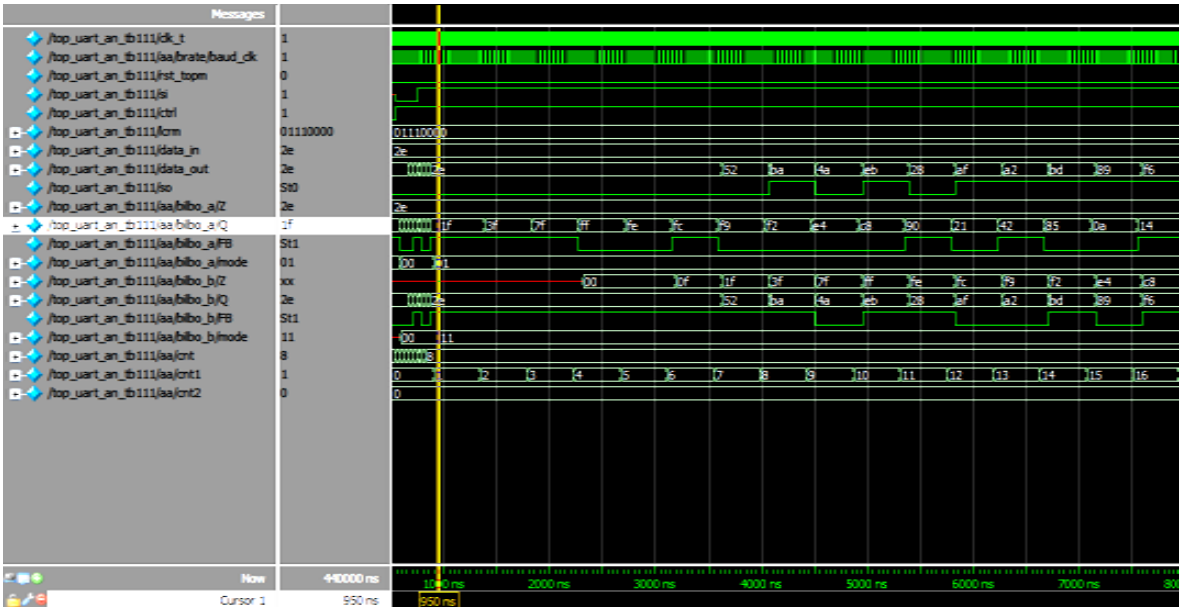


**Fig 7:** BILBO A in LFSR mode and BILBO B in MISR mode

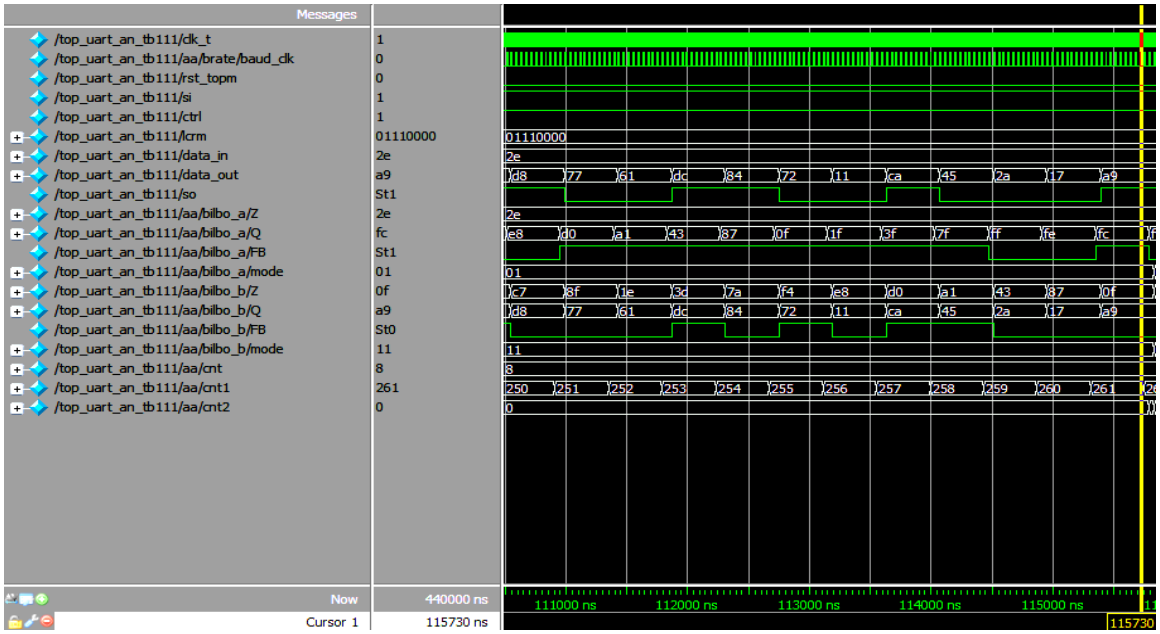



**Fig 8:** signature generated by MISR

The synthesis of the design is performed using Xilinx ISE 14.5. The VIRTEX-5 field programmable array is used for realisation of the design UART with BILBO.

The Device Utilisation Summary

Number of Slice Registers: 491 out of 12480 3%
Number of Slice LUTs: 629 out of 12480 5%
Number of fully used LUT-FF pairs: 245 out of 875 28%
Number of IOs: 34
Number of bonded IOBs: 25 out of 172 14%

The power analysis of the design is performed using Xilinx Power Analyzer. The result obtained from the power analysis is shown below.

On-Chip Power Summary

| On Chip | Power (mW) | Used | Available | Utilization (%) |
|---|---|---|---|---|
| Clocks | 13.12 | 2 | - | - |
| Logic | 0.91 | 591 | 12480 | 5 |
| Signals | 3.12 | 869 | - | - |
| IOs | 0.16 | 25 | 172 | 15 |
| Static Power | 320.82 | | | |
| Total | 338.13 | | | |

Power Supply Summary

| | Total | Dynamic | Quiescent |
|---|---|---|---|
| Supply Power (mW) | 338.13 | 17.31 | 320.82 |

## VI. Conclusion

In this paper, the test performance achieved with the implementation of BIST is proven to be adequate to offset the disincentive of the hardware overhead produced by the additional BIST circuit. The technique can provide shorter test time compared to an externally applied test and allows the use of low-cost test equipment during all stages of production.

## References

[1]. Nitin Patel; Naresh Patel, 4 July 2013. VHDL Implementation of UART with BIST capability. Computing, Communications and Networking Technologies (ICCCNT),2013 Fourth International Conference.
[2]. WAKHLE, G.B; AGGARWAL, I. ; GABA, S., 4 JUNE 2012. SYNTHESIS AND IMPLEMENTATION OF UART USING VHDL CODES. COMPUTER, CONSUMER AND CONTROL (IS3C), 2012 INTERNATIONAL SYMPOSIUM.
[3]. M.D. Mamun, M.S. Amin and J. Jalil, Single Core Hardware Modeling of Built-in-Self-Test for System on Chip Design, Research Journal of Applied Sciences Engineering and Technology, 4(7): 819-824, 2012
[4]. Shikha Kakar; Balwinder Singh ; Arun Khosla, 3, May 2009. Implementation of BIST Capability using LFSR Techniques in UART. International Journal of Recent Trends in Engineering ,Vol 1, No 3
[5]. Mohd.Yamani Idna Idris,Mashkuri Yaacob,Zaidi Razak. A VHDL Implementation of UART Design With BIST capability. Faculty of Computer Science and Information Technology,University of Malaya,50603 Kuala Lumpur,Malaysia.
[6]. Stanley L. Hurst, VLSI Testing Digital and Mixed Analogue/Digital Techniques (The Institution of Electrical Engineers, London, 1998)