A VHDL IMPLEMENTATION OF UART DESIGN WITH BIST CAPABILITY

Mohd Yamani Idna Idris, Mashkuri Yaacob, Zaidi Razak
Faculty of Computer Science and Information Technology
University of Malaya
50603 Kuala Lumpur
Malaysia
Email: {yamani, mashkuri, zaidi}@um.edu.my

ABSTRACT

The increasing growth of sub-micron technology has resulted in the difficulty of testing. Design and test engineers have no choice but to accept new responsibilities that had been performed by groups of technicians in the previous years. Design engineers who do not design systems with full testability in mind open themselves to the increased possibility of product failures and missed market opportunities. BIST is a design technique that allows a circuit to test itself. 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.

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

1.0 INTRODUCTION

Manufacturing processes are extremely complex, inducing manufacturers to consider testability as a requirement to assure the reliability and the functionality of each of their designed circuits. One of the most popular test techniques is called Built-In-Self-Test (BIST). A BIST Universal Asynchronous Receive/Transmit (UART) has the objectives of firstly to satisfy specified testability requirements, and secondly to generate the lowest-cost with the highest performance implementation. UART has been an important input/output tool for decades and is still widely used. Although BIST techniques are becoming more common in industry, the additional BIST circuit that increases the hardware overhead increases design time and performance degradation is often cited as the reason for the limited use of BIST [1].

This paper focuses on the design of a UART chip with embedded BIST architecture using Field Programmable Gate Array (FPGA) technology. The paper describes the problems of Very-Large-Scale-Integrated (VLSI) testing followed by the behavior of UART circuit using VHISC Hardware Description Language (VHDL). In the implementation phase, the BIST technique will be incorporated into the UART design before the overall design is synthesized by means of reconfiguring the existing design to match testability requirements. The UART is targeted at broadband modem, base station, cell phone, and PDA designs.

2.0 VLSI TESTING PROBLEMS

Today's highly integrated multi-layer boards with fine-pitch ICs are virtually impossible to be accessed physically for testing. Traditional board test methods which include functional test, only accesses the board's primary I/Os, providing limited coverage and poor diagnostics for board-network fault. In circuit testing, another traditional test method works by physically accessing each wire on the board via costly "bed of nails" probes and testers. To identify reliable testing methods which will reduce the cost of test equipment, a research to verify each VLSI testing problems has been conducted. The major problems detected so far are as follows:

a) Test generation problems;
b) The input combinatorial problems; and
c) The gate to I/O pin ratio problems.

2.1 Test Generation Problems

The large number of gates in VLSI circuits has pushed computer automatic-test-generation times to weeks or months of computation. The numbers of test patterns are becoming too large to be handled by an external tester and this has resulted in high computation costs and has outstripped reasonable available time for production testing.

Another test generation problem is that computer algorithms providing Automatic Test Pattern Generation (ATPG) work well for combinatorial logic but rather poorly for sequential logic circuits. Sequential circuits demand too much computer memory and computation since many more time states must be evaluated [2a].

2.2 The Input Combinatorial Problem

A combinatorial logic circuit with N primary input nodes has a total set of 2^N possible input vectors. This is the number of test vectors required to exhaustively test a circuit for those functions that a customer might use. In contrast to MSI (Medium-Scale-Integrated) circuits, the number of test vectors needed to exhaustively examine a VLSI circuit such as 32-bits microprocessor is prohibitive. However, a finite number of test vectors can still be applied to an IC and follow the economic rules of production. The finite number of test vectors is much lesser than the full exhaustive test set of a VLSI circuit [2a].

2.3 The Gate to I/O Pin Ratio Problem

As ICs grow in gate counts, it is no longer true that most gate nodes are directly accessible by one of the pins on the package. This makes testing of internal nodes more difficult as they could neither no longer be easily controlled by signal from an input pin (controllability) nor easily observed at an output pin (observe ability). Pin counts go at a much slower rate than gate counts, which worsens the controllability and observe ability of internal gate nodes [2a].

The VLSI testing problems described above have motivated designers to identify reliable test methods in solving these difficulties. An insertion of special test circuitry on the VLSI circuit that allows efficient test coverage is the answer to the matter. The need for the insertion has been addressed by the need for design for testability and hence the need for BIST.

3.0 VHDL AND BIST

It is increasingly common for design for testability (DFT) issues to be addressed at design reviews prior to circuit tape-out approval. Previously, in the age of schematics, this often requires design and test engineers to sift through pages of manuals and data sheets looking for things like asynchronous set/reset circuit configurations, derived or internally generated clocks, and combinatorial and sequential feedback loops. The review inevitably occurred late in the design cycle; adversely affecting project schedules if glitches were found, and making for an uncomfortable process for the circuit designer. However, with today's design practices, schematics are mostly outdated [3]. Designers can take more control of the DFT review by performing DFT rule checking at the register transfer language (RTL) (VHDL) level. Nevertheless, finding DFT problems in language-based designs is still not a simple task for humans.

The acceptance of the design for test techniques has been largely due to the possibility of VHDL support to this design style. It is desirable to eventually have available a BIST approach with similarly VHDL support. The high degree of standardization makes it possible to have most testability feature previously added to a design using VHDL [4] [5].

4.0 UNIVERSAL ASYNCHRONOUS RECEIVE/TRANSMIT (UART)

Serial data is transmitted via its serial port. 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, 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 [6] [7].

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.

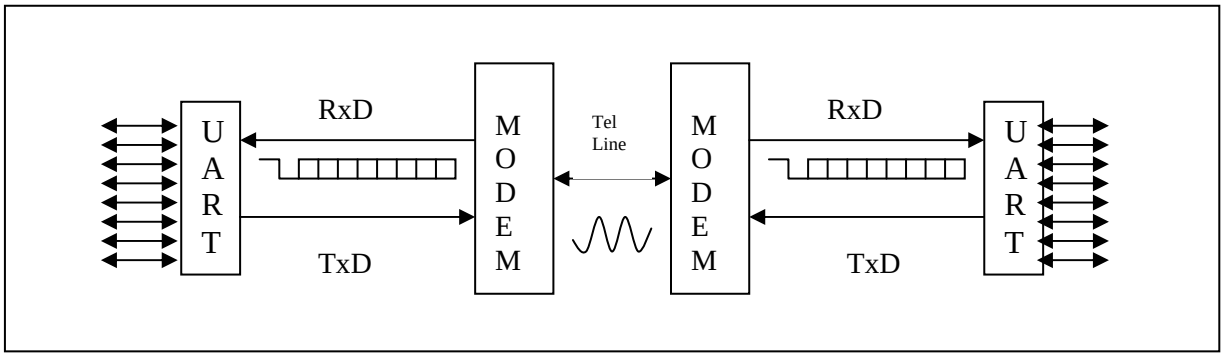

Fig. 1: Serial Data Transmission and Receive

5.0 THE FEATURES

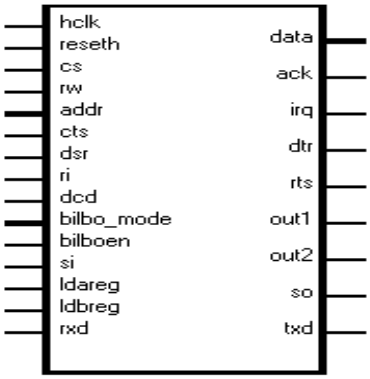

Fig. 2: UART with BIST top level design symbol

Table 1 describes the function and the description of all UART pins available at the top-level design (Fig. 2) of the VHDL implementation. The UART are capable of the following [8]:

a) Fully programmable serial interface characteristics:
   - 5-, 6-, 7-, or 8-bit characters
   - Even-, odd-, or no-parity bit generation and detection
   - 1-, or 2-stop bit generation
   - Baud generation

b) False-start bit detection

c) Line break generation and detection
d) Hold and shift registers eliminate the need for precise synchronization between the CPU and serial data
e) Programmable baud rate generator allows division of any input reference clock by 1 to (2^{16} –1) and generates an internal 16× clock
f) Standard asynchronous communication bits (start, stop, and parity) added to or deleted from the serial data stream
g) Modem control functions (CTS\, RTS\, DSR\, DTR\, RI\, and DCD\)
h) Independent receiver clock input
i) Transmit, receive, line status, and data set interrupts independently controlled
j) Internal diagnostic capabilities:
   - Loopback controls for communications link fault isolation
   - Break, parity, overrun, and framing error simulation
   - BIST

Table 1: UART pins description

Pin	In/out	Description
Hclk	IN	***Host Clock*** Clock for baud rate generation (e.g. operation frequency = 8MHz therefore the time period should be 1/8MHz=125ns)
Reseth	IN	***Asynchronous Reset*** Clears all registers and control logic when input is high
Cs	IN	***Chip Select*** Enables communication between the UART and the CPU. CS is an active low signal latches address strobe for completing chip selection.
Rw	IN	***Read/Write*** '1' = CPU can read status or data from the selected UART register '0' = CPU can write control or data into the selected UART register (Note: cs should be active)
Addr	IN	***Host Address*** 4 bit inputs, which select UART register.
Cts	IN	***Clear to Send*** When low indicates that the modem or data set is ready to exchange data
Dsr	IN	***Data Set Ready*** When low indicates that the modem or data set is ready to establish communication link with the UART.
Ri	IN	***Ring Indicator*** When low indicates that the telephone ringing signal has been received by the modem or data set
Dcd	IN	***Data Carrier Detect*** When low indicates that the modem or data set has detected a data carrier.
Bilbo_mode	IN	***BILBO Operating Mode*** "00" = shift register mode "01" = PRPG mode "10" = normal mode "11" = MISR mode
Bilboen	IN	***BILBO Enable*** '0' = disables BILBO '1' = enables BILBO
Si	IN	***BILBO Serial In*** A "seed" value to initialize BILBO
Ldareg	IN	Load BILBO Register 'A'

Ldbreg	IN	Load BILBO Register 'B'
Rxd	IN	***Received Data*** Serial data input from a communication link (data from peripheral devices, modem or data set).
Data	INOUT	***Data Bus*** Provides bi-directional communication between the UART and the CPU. Data, control words and status information are transferred via the data bus.
Ack	OUT	***Acknowledge*** Active low signal, which transfers acknowledge signal to host.
Irq	OUT	***Interrupt Request*** The interrupt request will be generated for the following conditions: 1. Data are ready in Receiver Hold Register (RHR) (if INTMSK is enabled). 2. Transmitter Hold Register (THR) is empty (if transmitter and INTMSK are enabled).
Dtr	OUT	***Data Terminal Ready*** When low, informs the modem or data set that the UART is ready to establish a communication link.
Rts	OUT	***Request to Send*** When low, informs the modem or data set that the UART is ready to exchange data.
Out1	OUT	***User designated output*** Can be set to active low by programming bit 2 of the Modem Control Register (MCR).
Out2	OUT	***User designated output*** Can be set to active low by programming bit 3 of the Modem Control Register (MCR).
So	OUT	***BILBO Serial Out*** The serial signature produced by MISR to be compared with the correct signature
Txd	OUT	***Transmitter's Data*** Serial data output to a communication link (i.e. peripheral devices, modem and data set).

6.0 UART WITH BILBO REGISTER AND TESTER

The problem of testing sequential network 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 [9] [10].

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 2 [11]:

Table 2: BILBO operating modes

B1B2	**Operating Mode**
00	Shift register
01	LFSR/PRPG
10	Normal
11	MISR

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 2^8 = 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

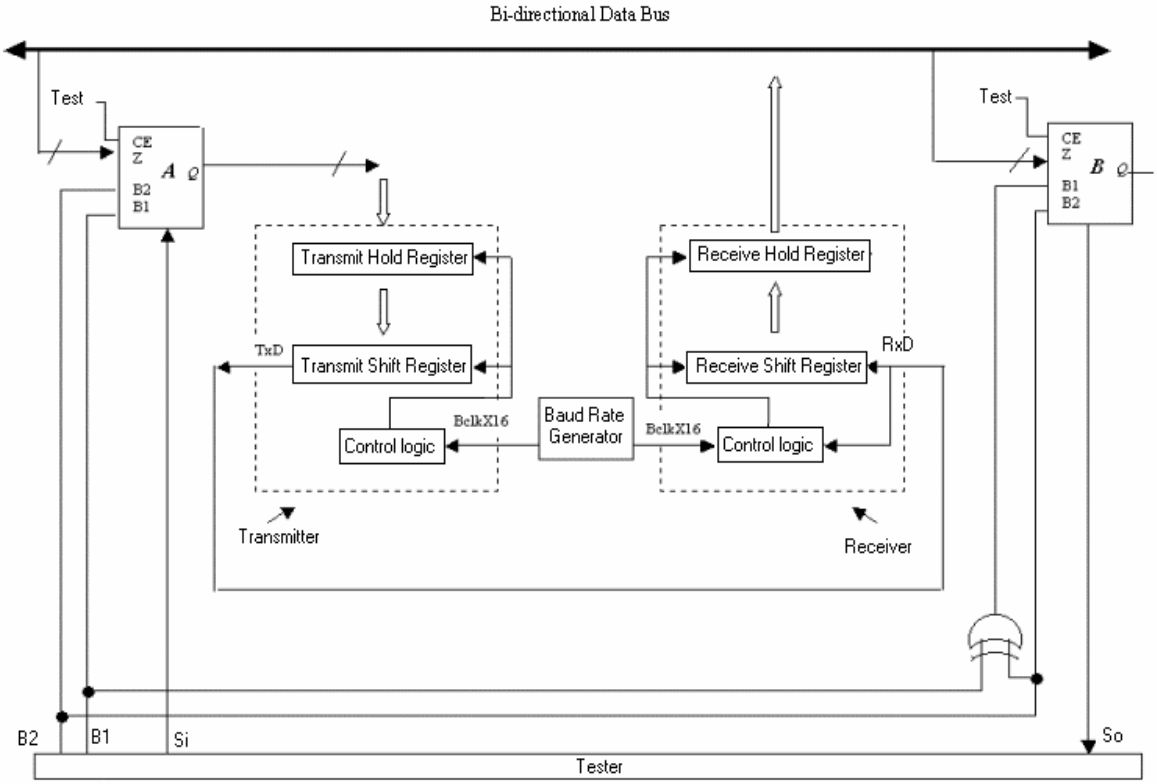

Fig. 3: UART with BILBO register and tester

7.0 SIMULATION

During BILBO’s normal mode (Bilbo_mode = 10), the chip acts as a normal UART chip. The transmitter and receiver simulation under normal mode is presented next followed by the simulation of UART under testing mode in succeeding section.

7.1 UART Normal Mode

7.1.1 Transmitter Simulation

The simulation shows the transmission of an 8-bit UART frame format with 1 stop bit and without a parity bit. The transmission was set at 115.2kbps using 40MHz clock, which is equal to 25ns (1/40MHz = 25ns) period.

Fig. 4 shows the signal results of the 8-bit data ("00000111_B") transmission via DATA[7:0]. The transmitted UART frame format can be observed at TXD (1 low start bit, 8 data bits (LSB to MSB), and 1 high stop bit). The 3-bit high data is equal to 26.05us. Therefore 1 data bit is equal to 26.05us/3 = 8.68us. From this information, the baud rate can be calculated as 1/8.68us, which is equal to 115,207 ≈ 115.2k.

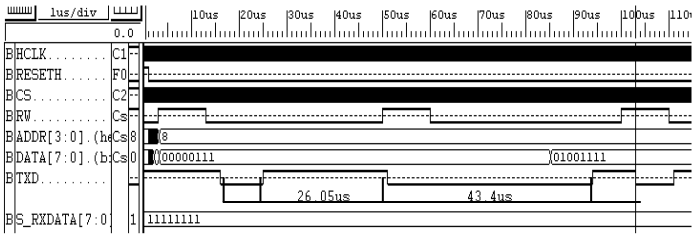

Fig. 4: Serial 8-bits data transmission at TXD

7.1.2 Receiver Simulation

Fig. 5 shows how an 8-bit serial data from RXD is received. The data consists of 4 high bits and 4 low bits (1 bit = 8.675us). The data is then converted to parallel S_RXDATA at 93.01us. S_RXDATA is the internal parallel data received at the receiver. The output of the received parallel data is then routed to DATA[7:0] output's pin. The parallel data will be activated when the data output is enabled (s_dataoe = '0'). The received data can be observed between the high impedance "ZZ_H" at DATA[7:0] (if the time/div is widened).

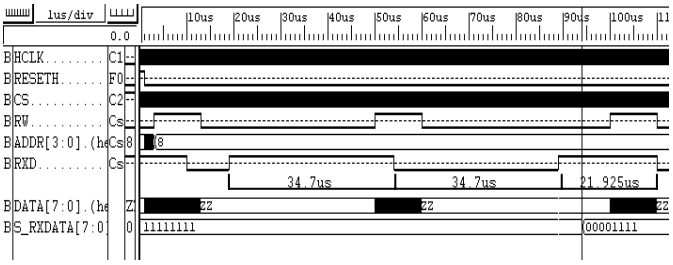

Fig. 5: 8-bits data received at DATA[7:0] when Data Output enabled. Internal Data Received at S_RXDATA

7.2 UART Testing Mode1

7.2.1 BIST Simulation

In this particular simulation case, the UART is set to internal loop back mode (Fig. 6). This mode is used to test both the transmitter and receiver of the UART. The mode will loop-back the serial data and transmit the data back to the receiver.

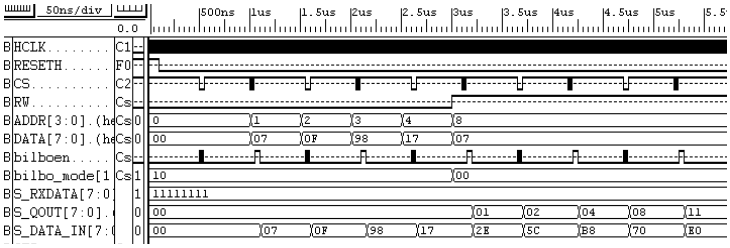

Fig. 6: BIST setup

To start the test, the BILBO_MODE (before 3us) is set to "10_B" to operate in normal mode. From 3us to 13us, the BILBO_MODE is set to "00_B" and acts as a shift register. BILBO shift register then shifts the value of S_DATA_IN from right to left (LSB to MSB) by using a serial in (si) pin (Fig. 7). The shifting is conducted to initialize the LFSR and MISR with a 'seed' data. S_DATA_IN then pushes the data to S_QOUT after 2-clock delay. Table 3 shows S_QOUT and S_DATA_IN in binary. As can be observed at the end of Table 3, the LFSR is initialized to "$0F_H$" and MISR is initialized to "$2E_H$".

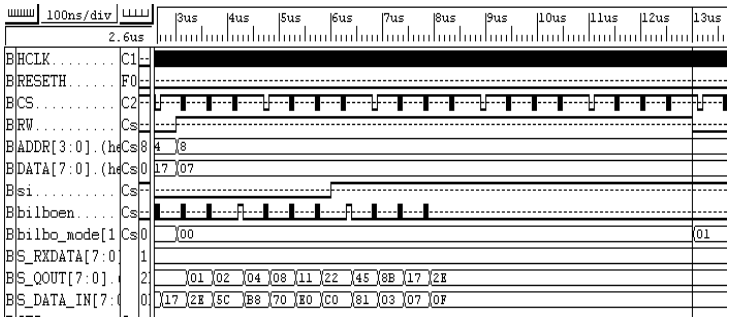

Fig. 7: BILBO acts as a Shift Register

Table 3: S_QOUT and S_DATA_IN in binary

S_QOUT (HEX)	**S_QOUT (BINARY)**	**S_DATA_IN (HEX)**	**S_DATA_IN (BINARY)**
01	00000001	2E	**0**0101110
02	00000010	5C	**0**1011100
04	0000010**0**	B8	**1**0111000
08	0000100**0**	70	**0**1110000
11	0001000**1**	E0	**1**1100000
22	0010001**0**	C0	**1**1000000
45	0100010**1**	81	**1**0000001
8B	1000101**1**	03	**0**0000011
17	0001011**1**	07	0000011**1**
2E	0010111**0**	0F	0000111**1**

Fig. 8 shows that the BILBO_MODE is set to "01_B". In this mode, the BILBO at the transmitter is configured as an LFSR and BILBO at the receiver is configured as an MISR. During BILBO_MODE = "01_B" DATA[7:0] can be ignored.

S_DATA_IN acts as an LFSR that produces parallel pseudo random pattern (PRPG) signals to the UART's transmitter. These parallel signals are then converted to serial data in a communication line and will be looped back to the receiver. The receiver converts the data back to parallel and forwards it to S_RXDATA. S_QOUT (MISR) compresses all the received pseudo random parallel data (S_RXDATA) into one signature. The produced signature is then compared with the correct signature.

S_DATA_IN (PRPG) is achieved by XOR-ing bit 1, 2, 3 and 7 (MSB…LSB, b7…b0) and the XORed result is placed to bit 0 (LSB). The other remaining bits (b6…b0) are shifted to the left.

The S_QOUT (MISR) is achieved as S_DATA_IN except that the produced data (PRPG) is then XORed with S_RXDATA.

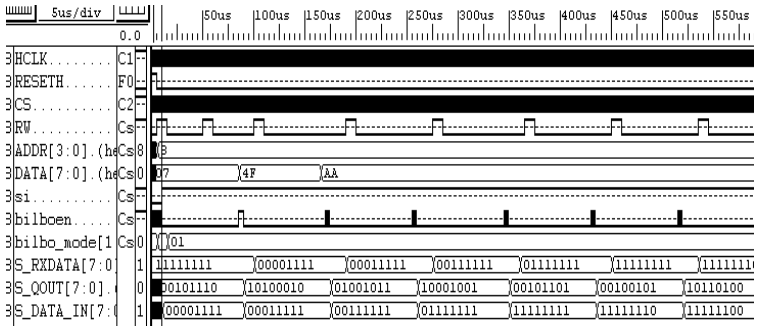

Fig. 8: BILBO acts as an LFSR and MISR

The process of feeding the transmitter with pseudo random data and compressing it with MISR is complete after 255 clocks iteration. The final result of MISR (S_QOUT) in Fig. 9 can be observed as "01010001_B". However, S_QOUT (MISR) is the internal data of the designed UART. Therefore, there should be a method to send out the signature without sacrificing extra observance output pins. The signature is shifted out at serial data out (so) output's pin. To shift out the signature, BILBO_MODE is set back to "00_B" and it acts as a shift register. In Fig. 10, 'bilboen' signal enables the signature to be transmitted as a serial out data (so). As can be observed, 'so' is transmitted as the following sequence: 1 low bit, 1 high, 1 low, 1 high, 3 low and 1 high. The result is the serial data of "01010001_B", which is equal to the value of MISR at S_QOUT. The signature produced is also similar with the correct signature achieved from the simulation of the entire self-test sequence approach using C programming. The RTL schematic is shown in Fig. 11.

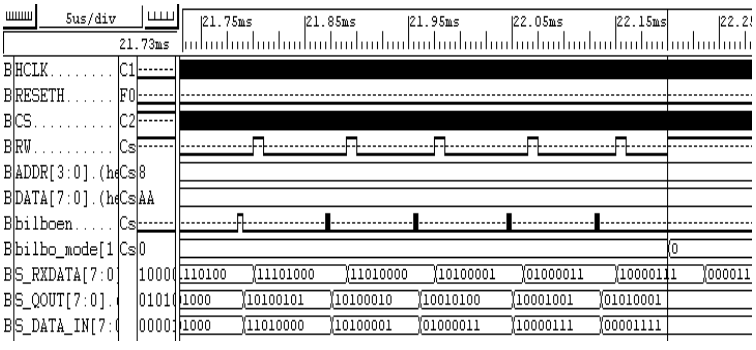

Fig. 9: Final value of MISR (S_QOUT) to be shifted out as a serial signature

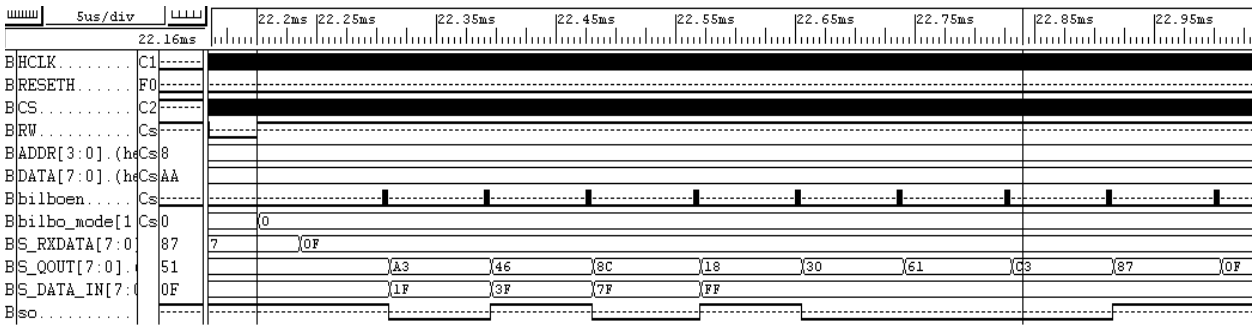

Fig. 10: Serial Data Out Signature at “so”

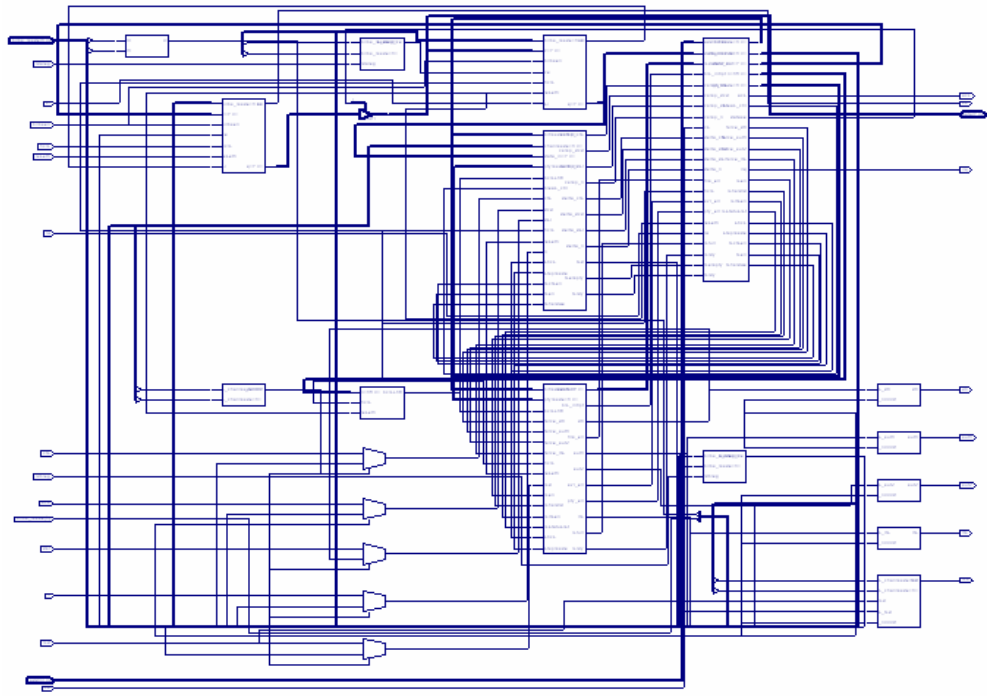

Fig. 11: RTL schematic

8.0 HARDWARE TEST

The simulation presented in the previous section is not adequate to assure that the designed circuit will function as intended. Therefore, there is a need to verify the design implementation on a real hardware. This section asserts the methods of testing the real FPGA that has been designed using Xilinx Foundation Series 2i (XSE2i). The hardware test uses XSTOOLs and XS40 V1.1 prototyping board provided by XESS Corporation [12]**.** The test is admittedly lacking of tact or taste but will serve if access to better equipment is not possible.

8.1 Pseudo Random Pattern Generation (PRPG)

To feed a “seed” value for initializing the LFSR, the programmed FPGA (BILBO) will need to be configured as a shift register using GXSPORT from XSTOOLs (“Bilbo_mode” = “00” or e.g “PC_D2 & PC_D1” = “00” (depends on pin location using user constraint file (**.ucf))). GXSPORT will output a byte of binary bits onto the eight data pins

(PC_D7, PC_D6, ……PC_D0) of the parallel port. It will be used to force logic levels onto the input pins of the FPGA to test a downloaded logic circuit. The shift register will shift the value of "si" (PC_D4). In this test, "si" is set to '1' (PC_D4 = '1') to initialize the LFSR to "11111111" (i.e. "00000000", "00000001", "00000011", "00000111", ……. "01111111", "11111111", 'si'='1' shifts from LSB to MSB each clock cycle). For further test, the LFSR can be initialized by changing the "si" states using a programmable signal generator and its appropriate software.

After the LFSR has been initialized, the programmed FPGA (BILBO) needs to be set to PRPG mode ("PC_D2 & PC_D1" = "01") to generate pseudo random pattern signals. The result of the pseudo random pattern generator (PRPG) waveforms can be observed using oscilloscope or logic analyzer.

The left most data on Fig. 12 (the dotted line) is observed as "00111011" (MSB….LSB) followed by "01110110". The waveforms obtained have proven the result of 8-bits PRPG in simulation and theory. The PRPG is achieved by XOR-ing bit 1,2,3 and 7 (MSB…LSB, b7…b0) then the XORed result is then placed to bit 0 (LSB). The other remaining bits (b6…b0) are then shifted to the left. Therefore, the result will be "01110110". How the LSB is achieved is shown below:

'1' *(bit 1)* **XOR** ***'0'*** *(bit 2)* **XOR** ***'1'*** *(bit 3)*
XOR ***'0'*** *(bit 7)* = ***'0'*** *(LSB)*

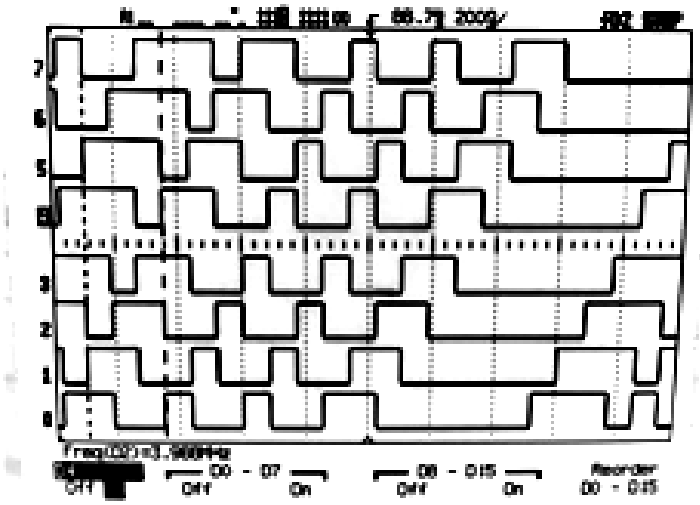

Fig. 12: Pseudo random pattern

8.2 Multiple Input Signature Register (MISR)

The test setup for MISR is almost the same as the PRPG except that the operation mode needs to be set to MISR ("PC_D2 & PC_D1" = "11") after initialized with "seed" value (using "si" and shift register). For this test, the test data (z<7, 6, 5,…..1, 0>) are set to "00000111" using an external circuit and switch (or use XSTEND board provided by Xess Corp). These data act as the data output of the circuit under test. The MISR outputs are then observed at outputs q<7, 6, 5, ….. 1, 0> using mixed signal oscilloscope.

Fig. 13 shows the MISR result observed with an oscilloscope. The left most data (the dotted line) is observed as "00101010" followed by "01010011". The MISR is achieved in a similar procedure as the PRPG but the produced data (PRPG) is then XORed with the output data of the circuit under test (z<7, 6, 5, ….. 1, 0>). How the signal result is produced is shown below:
"00101010" XORed and shift will produce "01010100"
"01010100" XORed with "00000111" will produce "01010011"

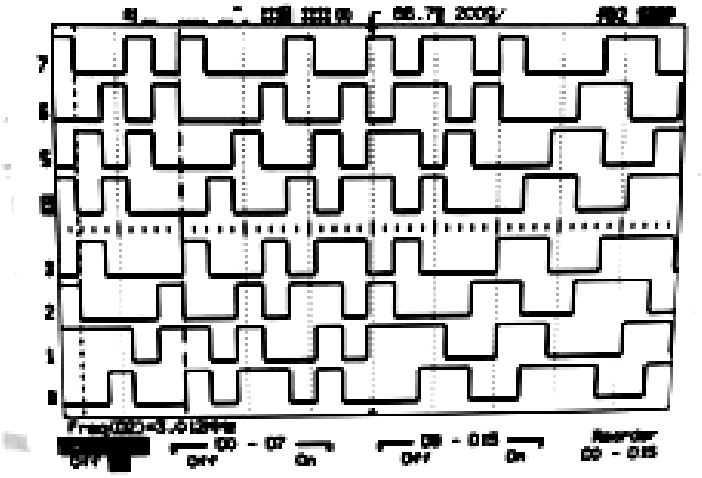

Fig. 13: Multiple Input Signature Register (MISR)

9.0 BIST DESIGN SUMMARY

The simulations and hardware tests have shown and ensured the capability and reliability of the designed UART chip. However, as stated before, the reasons for the limited use of BIST are due to area overhead, performance degradation and increased design time. In this section, the reports after the optimization process will be used as a basis for comparing the UART design before and after the implementation of the BIST technique. Table 4 shows the UART design and timing summary generated with and without the implementation of BIST technique. The summary is obtained from the implementation log file of Xilinx XSE2i Project Manager Software. By comparing these reports, it can be shown that the reliability of the chosen technique for the testable UART chip can be proven.

From the design summary, it can be observed that the number of Configurable Logic Block (CLB) used after the implementation of BIST technique is increased from 81% to 96%. The difference of 15% of the total CLBs area overhead results in 1.803 ns (i.e. 13.626 ns - 11.823 ns) increment of the maximum net delay. The implementation of BIST technique has also resulted in the decrease of the maximum frequency from 34.87 MHz to 31.98 MHz. This shows that the UART with embedded BIST design may not work well if its host clock frequency exceeded the maximum frequency of 31.98 MHz. The reason why the design may not cater for high-speed clock is due to the possibility of a real time delay, which may be caused by temperature or the delay within the FPGA design itself (e.g. long carry propagation times). The delay will limit the capability of data to be captured at some critical point. The faulty data captured may lead to errors at the output pins.

10.0 CONCLUSION

The simulated waveforms presented in this paper have proven the reliability of the VHDL 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. The test as shown earlier in Fig. 10 is completed at 22.2ms using 40 MHz clock speed transmitting at 115.2k baud rate. Even though it seems not to be as fast as it should be when BIST is implemented (the receiver needs to wait the signal from the transmitter), the UART module still takes advantage of the 100% fault coverage. This is the most important thing that should not be left out by any designer to ensure the reliability of their design.

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. The LFSR replaces the function of the external tester features such as a test pattern generator by automatically generating pseudo random patterns to give 100% fault coverage to the UART module. The MISR acts as a compression tool, compressing the output result when automatic pseudo random pattern is fed to the UART. The shift register minimized the input/output overhead by shifting the parallel signature produced by MISR into serial signature. The reduction of the test cost will lead to the reduction of overall production cost.

Table 4: Design summary

```
Design Summary:
  Number of errors:       0
  Number of warnings:     2
  Number of CLBs:         160 out of   196   81%
    CLB Flip Flops:     133

    4 input LUTs:       279 (1 used as route-throughs)
    3 input LUTs:       49 (16 used as route-throughs)
  Number of bonded IOBs:      28 out of    61   45%
    IOB Flops:          9
    IOB Latches:        0
  Number of clock IOB pads:   1 out of     8   12%
  Number of primary CLKs:     1 out of     4   25%
  Number of secondary CLKs:   1 out of     4   25%
  Number of TBUFs:            3 out of   448    1%
  Number of startup:          1 out of     1  100%

Total equivalent gate count for design: 2789

Additional JTAG gate count for IOBs:    1344

Timing summary:
---------------------

Timing errors: 0  Score: 0

Constraints cover 2523 paths, 373 nets, and 1235 connections (100.0% coverage)

Design statistics:
  Minimum period:   28.678ns (Maximum frequency: 34.870MHz)
  Maximum combinational path delay:  33.437ns
  Maximum net delay:  11.823ns
```

(a) UART without BIST Design Summary

```
Design Summary:
  Number of errors:       0
  Number of warnings:     2
  Number of CLBs:         190 out of   196   96%
    CLB Flip Flops:     158
    4 input LUTs:       342 (1 used as route-throughs)
    3 input LUTs:       54 (18 used as route-throughs)
  Number of bonded IOBs:      35 out of    61   57%
    IOB Flops:          2
    IOB Latches:        0
  Number of clock IOB pads:   1 out of     8   12%
  Number of primary CLKs:     1 out of     4   25%
  Number of secondary CLKs:   1 out of     4   25%
  Number of TBUFs:            3 out of   448    1%
  Number of startup:          1 out of     1  100%
  4 unrelated functions packed into 4 CLBs.

 (2% of the CLBs used are affected.)
Total equivalent gate count for design: 3289
Additional JTAG gate count for IOBs:    1680

Timing summary:
---------------------

Timing errors: 0  Score: 0

Constraints cover 2848 paths, 443 nets, and 1498 connections (100.0% coverage)

Design statistics:
  Minimum period:   31.270ns (Maximum frequency: 31.980MHz)
  Maximum net delay:  13.626ns
```

(b) UART with BIST Design Summary

REFERENCES

[1] S. Wang**, "**Generation of low power dissipation and high fault coverage patterns for scan-based BIST", in *Proceedings of International Test Conference*, 2002, pp. 834 –843.

[2] M. Ibrahim Abubakar, "A Built in Self Testable Bit-Slice Processor", Faculty of Computer Science & Information Technology, University of Malaya, May 1995.

[3] J. Turino, "RTL DFT Rule Checking – The Circuit Designer's Secret Weapon", *Integrated System Design Magazine*, 2000.

[4] A. P. Stroele, and H. J. Wunderlich, "Hardware-Optimal Test Register Insertion", in *IEEE Transactions On Computer-Aided Design of Integrated Circuits and Systems*, June 1998, Vol. 17, No. 6, pp. 531-539.

[5] Z. Navabi, "VHDL Analysis and Modeling of Digital Systems", McGraw-Hill Inc., 1991.

[6] M. S. Michael, "A Comparison of the INS8250, NS16450 and NS16550AF Series of UARTs", *National Semiconductor Application Note 493*, April 1989.

[7] "PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs", *National Semiconductor Application Note*, June 1995.

[8] M. S. Harvey, *Generic UART Manual*, SiliconValley, December 1999.

[9] O. A. Petlin, and S. B. Furber, "Built-In-Self-Testing of Micropipelines" in *Advanced Research in Asynchronous Circuits and Systems*, IEEE, 1997, pp 22-29.

[10] J. Turino, "RTL DFT Rule Checking – The Circuit Designer's Secret Weapon", *Integrated System Design Magazine*, 2000.

[11] C. H. Roth, *Digital System Design Using VHDL*, PWS Publishing Company, 1998

[12] http://www.xess.com

BIOGRAPHY

Mohd. Yamani Idna Idris obtained both his B.Eng. (Hons) in Electrical Engineering (2000) and Master of Computer Science in Chip Design (2002) from the University of Malaya. Currently he is a lecturer in the Department of Computer System & Technology, University of Malaya. His research interest is in the area of System on Chip and digital design.

Mashkuri Yaacob graduated with a B.Eng. (Hons.) degree in Electrical Engineering from the University of New South Wales, Sydney, Australia 1976 which was supported by the Colombo Plan Scholarship. He obtained both his M.Sc (1977) and PhD (1980) degrees in Computer Engineering from UMIST under a University of Malaya (UM) Scholarship. Currently is on secondment to MIMOS Berhad as the Director of Grid Computing and Bioinformatics Lab. His current research interests are in bioinformatics, computer architecture, grid computing, computer networks and VLSI chip design.

Zaidi Razak obtained both his Bachelor of Computer Science (Hons) degree (1997) and Master of Computer Science in Chip Design (2000) from the University of Malaya. He has participated in several completed research grants working on chip design as a researcher. Currently he is pursuing his doctorate in the field of System on Chip.