OPEN ACCESS INTERNATIONAL JOURNAL OF SCIENCE & ENGINEERING

DESIGN FOR TESTABILITY TECHNIQUE OF REVERSIBLE LOGIC CIRCUITS BASED ON EXCLUSIVE TESTING

M.Praneeth[1], Dr.Manish Bilgaye[2] , Y.David Solomon Raju[3]

[1] PG Scholar, Dept of Electronics and communication engineering, Holymary Institute Of Technology And Science, Bogaram(V), Keesara (M),Hyderabad -501 301

[2] Professor, Dept of Electronics and communication engineering, Holymary Institute Of Technology And Science, Bogaram(V), Keesara (M),Hyderabad -501 301

[3]Associate Professor, Head of Dept, Electronics and communication engineering, Holymary Institute Of Technology And Science, Bogaram(V), Keesara (M),Hyderabad -501 301

Abstract** The emerging technology of reversible circuits offers a potential solution to the synthesis of ultra low-power quantum computing systems. A reversible circuit can be envisaged as a cascade of reversible gates only, such as Toffoli gate, which has two components: k control bits and a target bit (k-CNOT), k ≥ 1. While analyzing testability issues in a reversible circuit, the missing-gate fault model is often used for modeling physical defects in quantum k-CNOT gates. In this paper, we propose a new design for testability technique for quantum reversible circuits in which the gates of a circuit are grouped into different sets and the gates from each set are attached to an additional input line via an extra control. Such arrangement makes it possible to test the gates belonging to a set separately. Our algorithm exploits the feature of many reversible circuits in which the high quantum cost gates have target on the same line and this line is devoid of any control of other gates. All these gates skip addition of extra control for testing. The proposed technique offers less quantum cost in comparison to other DFT techniques published so far. Testing is an essential step that ensures the designed circuit realizes desired functionality. Testing an integrated circuit is the time-consuming task nowadays. Different methods are needed to get shorter test time. This paper presents an original approach and a practical system for implementation and testing of reversible logic. These testing methods, based on DFT (Design for testability) techniques. Reversible ALU is a testing circuit in this process. This ALU is tested by using two techniques of DFT which improves the controllability and observability of internal nodes, so that embedded functions can be tested. A node is said to be testable if it is easily controlled and observed. The techniques are 1) Ad hoc method and 2) Simple BIST (Built-in self-test) method, BIST belongs to the structured technique of DFT. This design is with Verilog HDL and simulated using ISIM simulator and implemented on Spartan3E (XC3S500E-FG320-5) FPGA. This proposed designed architecture provides delay of 41.054ns for Simple BIST and 40.774ns for Ad-hoc test methods, with an area coverage of 7% and 5% on Spartan 3E implemented by using Xilinx ISE Design Suite.

***Keywords**: DFT (design for testability), Simple BIST (Built- In-Self Test), Reversible Logic*

I INTRODUCTION

Fault finding and correction of the faults are the two vital stages of testing the circuits. In the first stage, the fault is identified and in the second stage the fault is corrected. In a chip, several faults occur during manufacturing like bridging faults, missing gate faults etc. Each stage of correction the circuit has its own technique and complexity. Technically VLSI circuits are needed to be tested so that the tested product obtains designed functionality. s having more advantages with reversible logic in digital circuits, they need to be tested before release. Such testing has different techniques with advantages. The above Figure.1 shows the flow diagram of the design the circuit is designed and circuit is optimized and the optimized circuit is sent to verification. If the verification fails it is sent to debugging mode. In this debugging mode, the source of the error is identified if the error is found then it is sent to synthesis and circuit optimization mode for error correction and if the verification holds then the circuit holds the logic verification and the circuit is said to be ok. Reversible computation is an alternative computation paradigm where there is no information loss. A function is reversible if the number of inputs is equal to the number of outputs and there is bijection mapping between input and output. Information loss leads to some amount of energy dissipation, as in irreversible computation. According to Landauer’s principle, loss of 1 bit of information dissipates kTlog2 joules of energy, where k is Boltzmann’s constant and T is the temperature at which the erasing is done. In 1973, Bennett showed that in order to prevent this kTlog2 joules of energy dissipation per bit of information loss, the computation has to be reversible. Although in the current VLSI circuit, this energy loss is negligible compared to total energy consumption but with the increase in packing density of electronics component in ICs this factor may lead to considerable amount of heat dissipation in future. Thus reversible logic can be used to perform computation with less energy dissipation and is an attractive alternative in low power computation. Another important application of reversible logic is quantum computation. Quantum computation is inherently reversible. It is a powerful alternative to classical computation and can solve many important problems in exponentially less time complexity. Other applications of reversible computation includes optical computing, digital signal processing, communication, cryptography, nanotechnology, DNA technology, and low-power CMOS design. A Reversible function is implemented by reversible circuits which is nothing but a cascade of reversible gates. Here, we consider the problem of testing a reversible circuit. Testing of reversible circuits is comparatively easier than irreversible circuits. Agrawal has shown that fault detection probability is greatest when information output of a circuit is maximized [1]. Previously, several researchers have studied the problem of fault modelling and testing of reversible logic circuits [2]–[19]. Initially the stuck-at fault model is used but later it was replaced by Missing gate fault model proposed by J. P. Hayes [2]. Although Testing of Reversible circuits are comparatively easier than irreversible circuits, the DFT techniques available till date incurrs high quantum costs. Thus a suitable design for testability technique provides less quantum cost is very necessary. In this paper, we propose a Design-for-Testability technique for a given reversible circuit. We make an ($n \times n$) reversible circuit implemented with k-CNOT gates easily testable by adding some extra input lines and some additional toffoli gates so that all missing gate faults are detected.

II LITERATURE SURVEY

A. Reversible logic

Reversible computation is a method of computation in which the computation process is time-invertible i.e from one state of the computational process it is always possible to generate the previous state. In reversible computation no information about a state is lost. A function is reversible if there exists a bijective mapping between inputs and outputs. In a reversible function the output can be uniquely determined from input and input can be recovered from the output. In a reversible function the output. The concept of reversible logic gates is used for reducing power consumption and loss of data. Reversible computing is the application principle of recycling to the computing. It is because input can be reconstructed from the output. This logic uses the reversible gates which have the same number of inputs and outputs it is shown in the Figure.2. Some of the cost metrics like garbage outputs, number of gates, Quantum cost, constant outputs are used to estimate the performance of reversible circuits. A Reversible circuit design can be modelled as a Sequence of discrete time slices and depth is a summation of total time slices. Reversible computation is a method of computation in which the computation process is time-invertible i.e from one state of the computational process it is always possible to generate the previous state. In reversible computation no information about a state is lost. A function is reversible if there exists a bijective mapping between inputs and outputs. In a reversible function the output can be uniquely determined from input and input can be recovered from the output. In a reversible function the output 1) Single Missing Gate Fault (SMGF): A single missinggate fault (SMGF) occurs when there is removal of one k- CNOT gate from the circuit. An SMGF occuring in the circuit of Fig. 2(a) is shown in Fig. 2(b) . The box in Fig. 2(b) encloses the missing gate. An SMGF is detected by an input vector that produces a 1 at all control inputs of the missing gate. If we apply (1,1,1,1) at the input of the fault-free circuit, the output would be (1,1,1,0), whereas in the presence of the SMGF the output will be (1,1,0,1). Thus (1,1,1,1) is the test for this fault. 2) Repeated-Gate Fault (RGF): The repeated-gate fault (RGF) occurs when a gate in a circuit is repeated one or multiple times. If the number of occurrences is odd, the RGF is identical to that of an SMGF at the same gate. Hence, the RGF model is subsumed by the SMGF model. Fig. 2(c) shows the faulty circuit inflicted with a RGF, where the first gate is repeated. If we apply (1,1,1,1), the output of the fault-free circuit in Fig. 2(a) be (1,1,1,0), whereas in the presence of the the above RGF, the output will be (1,1,0,1), thus detecting the fault. 3) Multiple Missing-Gate Fault

(MMGF): The multiple missing-gate fault (MMGF) occurs when several consecutive gates are missing simultaneously in the circuit. A MMGF is detected by a test which detects SMGF of one or more of the missing gates. Fig. 2(d) shows a MMGF in which the position of the consecutive missing gates are shown by a box. If we apply (0,1,1,1), the output of the fault-free circuit in Fig. 2(a) would be (0,1,0,1), whereas in the presence of above MMGF, the output will be (0,1,1,1), thus detecting the fault. 4) Partial Missing-Gate Fault (PMGF): A partial missinggate fault (PMGF) corresponds to removing m out of k control inputs in a k-CNOT gate, thus transforming it into a (k-m)- CNOT gate. The number m is called the order of the PMGF. Fig. 2(e) shows an PMGF with a box marking the position of the missing control. A PMGF is detected by a vector which produces 0 at the missing control and 1 at all other controls. If we apply (0,1,1,1), the output of the fault-free circuit would be
(0,1,0,1), whereas in the presence of the first-order PMGF as shown in Fig. 2(e), the output will be (0,1,1,1), thus detecting the fault. For a gate with k control lines, a first-order PMGF is detected by setting missing control line to 0, and setting 1's to all other control lines. All k first-order PGMFs of a gate have detection conditions that conflict with each other ; hence, k test vectors are required to test for all first order PMGF. However, these vectors are guaranteed to cover all higher-order PMGFs at the same gate [10]. 5) Detectable and undetectable faults: If a fault does not change the functionality of the circuit that fault is undetectable. For RGF, if the gate which is being repeated, is additionally repeated for even number of times, then the circuit functionality does not change. In that case this fault is not detectable. There may be some special circuit where missing of some consequetive gates may not change the functionality of the circuit. In that case that MMGF is undetectable.

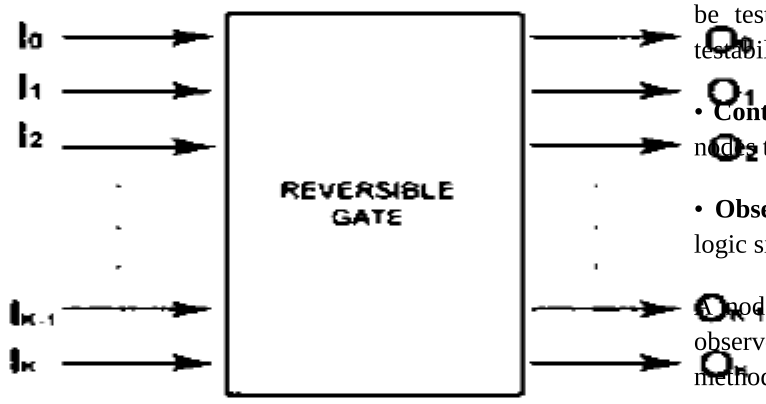


Figure 1. A n × n Reversible Gate

III PROPOSED SYSTEM

3.1 ALU AND DESIGN FOR TESTABILITY

Arithmetic and logic unit (ALU):
An ALU is the fundamental building block of the central processing unit (CPU). An ALU performs basic arithmetic and logic operations. Examples of arithmetic operations are addition, subtraction, multiplication, and division. Examples of logic operations are comparisons of values such as NOT, AND, and OR.

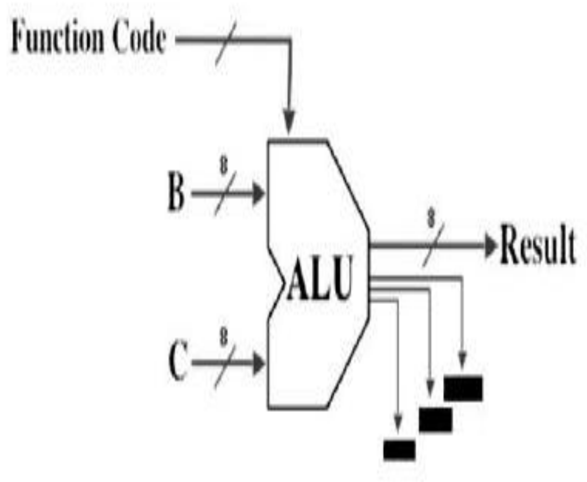


Figure 2. A simple 8 bit ALU

In the Above Figure.10, the function code is referred as the selection of ALU operations which can be either logical or arithmetic operation

Design for testability (DFT):

Design-for-testability improves controllability and observability of the circuit. Hence the function embedded can be tested. The two fundamental properties determine the testability of a node are:

• **Controllability:** The complexity of setting internal circuit nodes to 0 or 1 by assigning values to primary inputs (PIs)

• **Observability:** The difficulty of observing the state of a logic signal to a primary output (PO).

A node is said to be testable if it is easily controlled and observed. Design for testability (DFT) refers to those design methods that make the task of subsequent testing easier.

DFT TECHNIQUES

There also is no single DFT technique, which is effective for all kinds of circuits. DFT techniques can largely be divided into two categories, i.e., ad hoc techniques and structured (systematic) techniques.

DFT methods for digital circuits:

1) Ad-hoc methods

2) Structured methods: Built-in self-test

Ad-hoc techniques:

Circuits that are large should be subdivided into smaller circuits to lessen the test cost. One of the vital steps in designing a chip for testing is appropriately partitioning the chip. The circuit must be divided into modules at every level of the design. The divided sub circuit should be functional according to the operational boundaries. Partitioning is done using multiplexers and/or scan chain. The goal was to target modules of circuits that are difficult to test to add circuitry (test point) to improve Controllability Observability.[22] Basic Idea of Ad-hoc techniques is Add MUXs to provide access to/from internal circuitry Controllability & Observability, add gates to provide control to internal circuitry Controllability only. Add these "test points" only where needed in circuit Low area overhead penalty. Little (if any) performance impact Critical paths can often be avoided Target difficult to test sub circuits Potential for significant increase in fault coverage. Creative testability solutions on a case-by-case basis but, we must figure out what & where those are. Some Benefits Provide test points for controllability & observability and easier initialization for logic simulation and design verification. Partitioning the logic into more comfortable test pieces provide access to embedded blocks; Core tests can be re-used Bypass clock generation circuits (oscillators, one-shots, etc.). Avoid or bypass asynchronous logic Break feedback loops (when they are the problem) break up large counters into smaller ones. Disable intentional redundant logic for testing.

3.2 BUILT-IN SELF-TEST (BIST)

A built-in self-test (BIST) is a technique that enables a chip to self-test. BIST is a design-for-testability technique that places the testing functions physically with the circuit under test (CUT). The basic BIST architecture requires the addition of three hardware blocks to a digital circuit: a test pattern generator, a response analyzer, and a check controller. The test pattern generator (Linear Feedback Shift Register (LFSR) is implemented in this design) generates the test patterns for the CUT shown in the Figure.11.

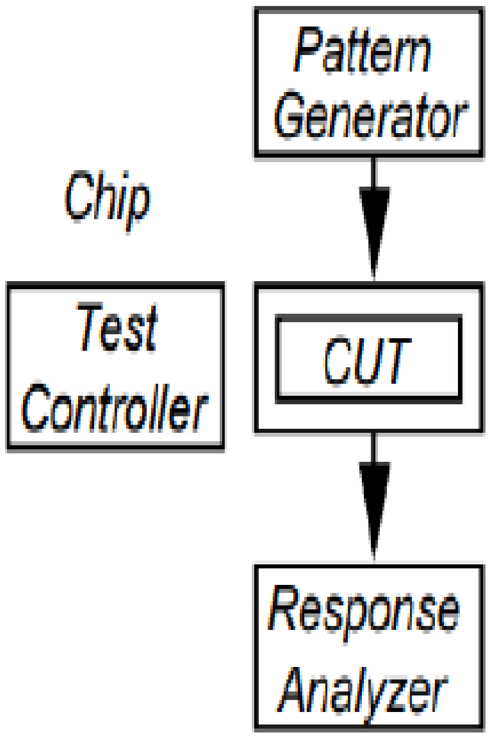


Figure 3. Test process

[21]The main purpose of BIST is to reduce the complexity, and thereby decrease the cost and reduce reliance upon external (pattern-programmed) test equipment. BIST reduces cost in two ways: Reduces test-cycle duration Reduces the complexity of the test/probe setup, by reducing the number of I/O signals that must be driven/ examined under tester control. [20]BIST can be used for non-concurrent, on-line testing of the logic and memory parts of a system. It can readily be configured for event-triggered testing, in which case, the BIST control can be tied to the system reset so that testing occurs during system start-up or shut down. BIST can also be designed for periodic testing with low fault latency. This requires integrating a testing process into the CUT that promises the recognition of all target faults within a fixed time. On-line BIST is implemented with dual objectives of total fault coverage and less fault latency. Hence, the test generation (TG) and response monitor (RM) are generally designed to guarantee coverage of specific fault models, minimum hardware overhead, and reasonable set size. These goals are met by different techniques in different parts of the system.

IV SIMULATION RESULTS AND DISCUSSION

Technology Schematic

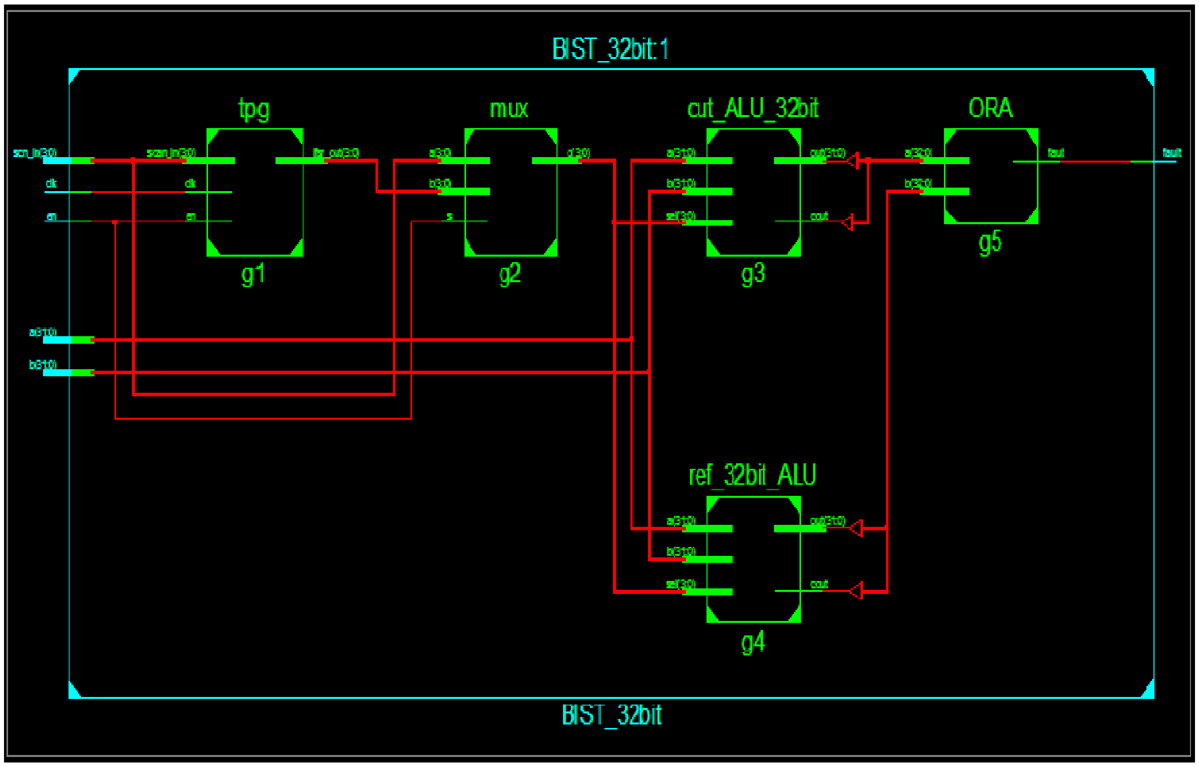


Figure 4. RTL schematic of BIST Test

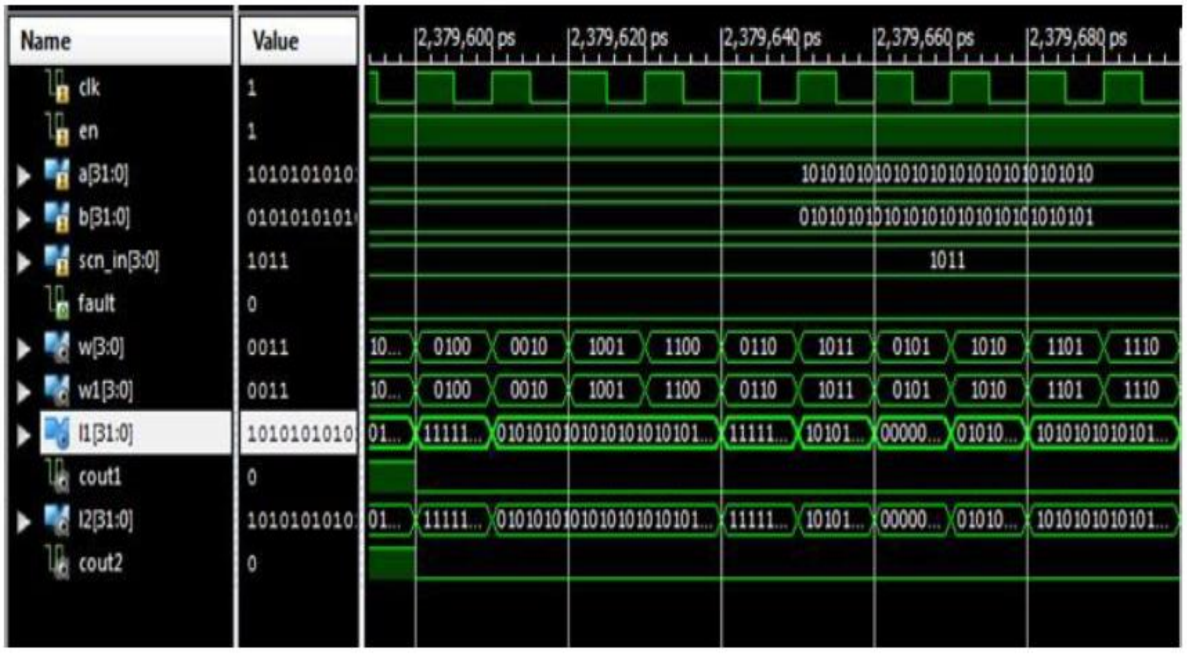


Figure 5. Simulation results of BIST Test

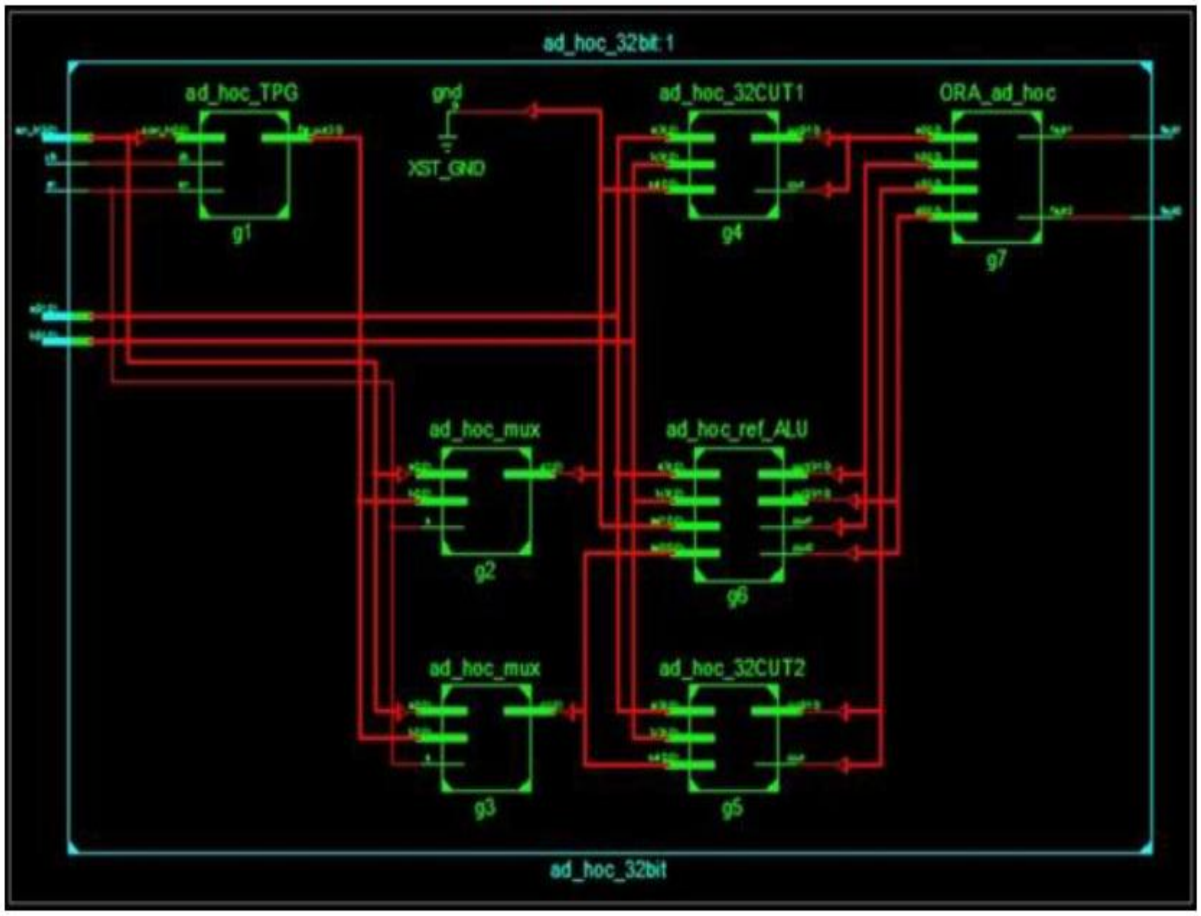
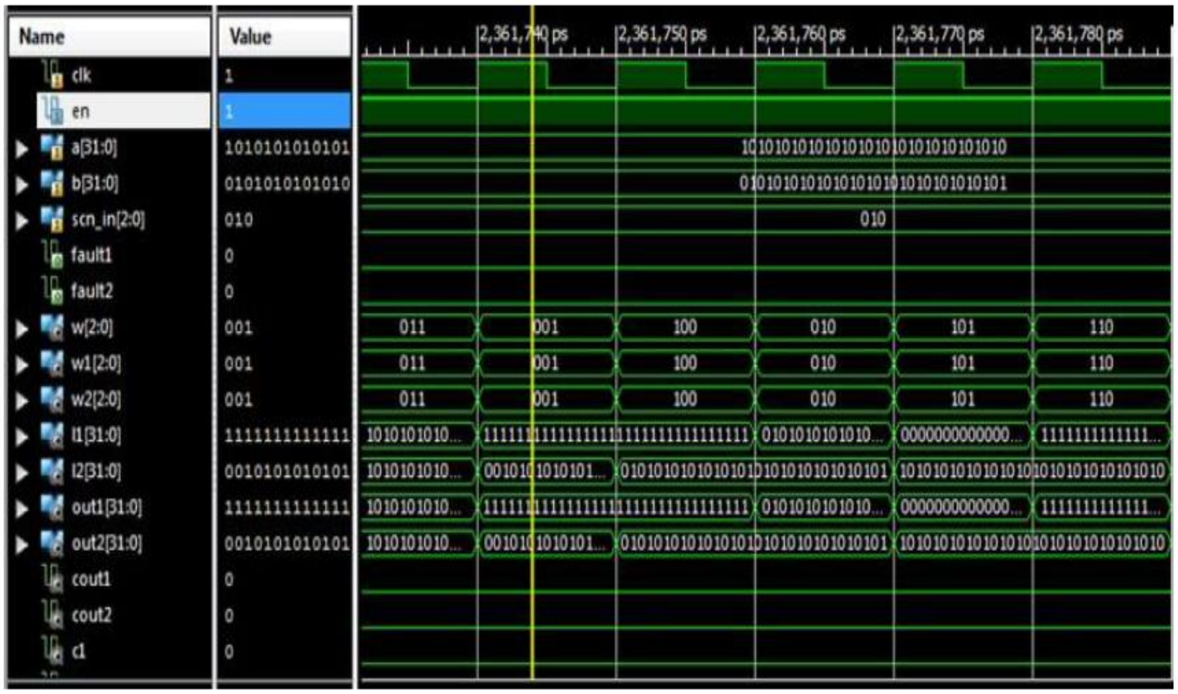


Figure.6. Simulation output



Name	Value
clk	1
en	1
a[31:0]	1010101010101
b[31:0]	0101010101010
scn_in[2:0]	010
fault1	0
fault2	0
w[2:0]	001
w1[2:0]	001
w2[2:0]	001
l1[31:0]	1111111111111
l2[31:0]	0010101010101
out1[31:0]	1111111111111
out2[31:0]	0010101010101
cout1	0
cout2	0
c1	0

Figure 7. Simulation results of Ad-hoc Test

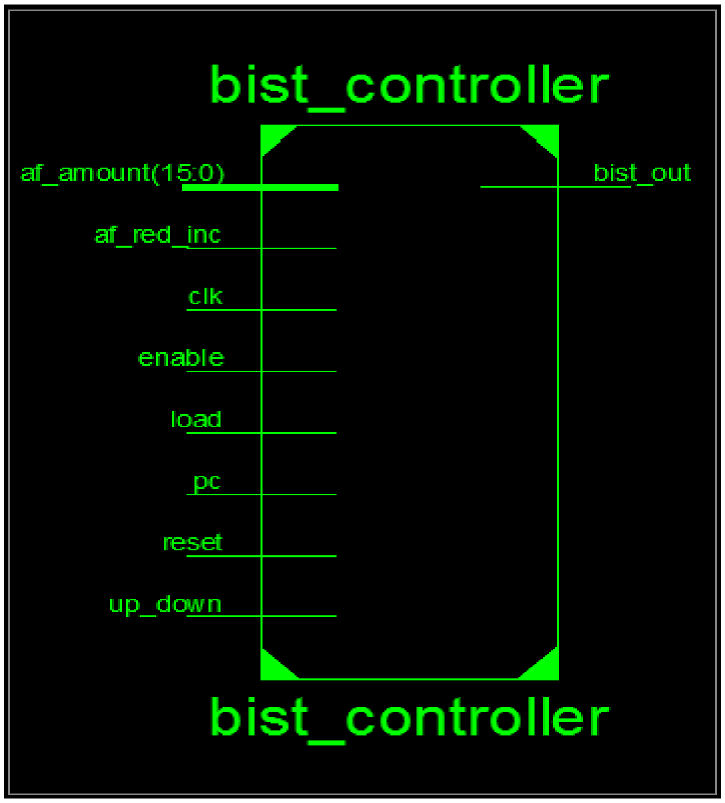


Fig 8 Pin diagram

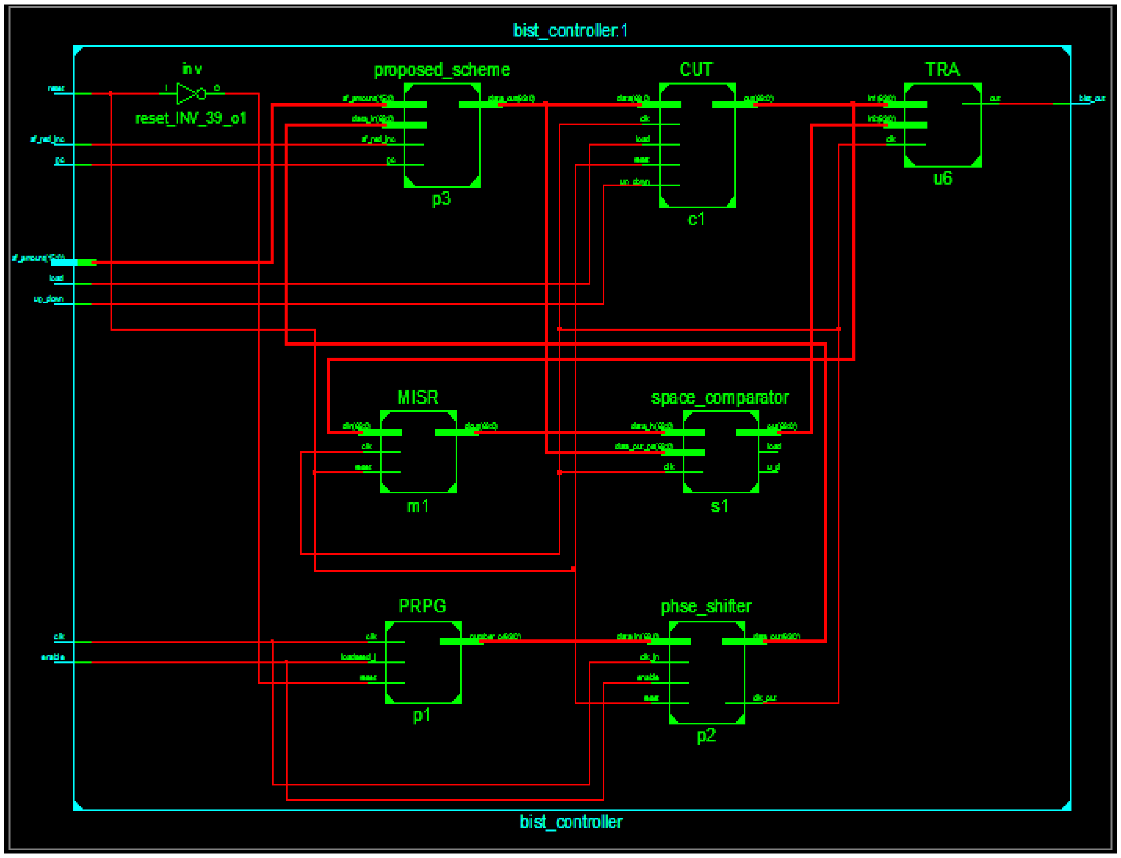


Fig 9 RTL schematic

V CONCLUSION

Our technique capitalizes on scan-based Logic Built-In SelfTest (LBIST) to put on the Testing of internal nodes in a complex circuitry is becoming a more significant problem, and thus it is essential that a designer must consider how the IC will be tested, and new structures will be incorporated in the design. Ad hoc and BIST techniques give the best way to check any combinational circuits. Such testing provides a valid circuitry. This design is implemented with Verilog HDL and simulated using ISIM simulator and implemented on Spartan3E (XC3S500E-FG320-5) FPGA. This proposed designed architecture provides path delay of 41.054ns for Simple BIST and 40.774ns for Ad-hoc technique. Ad-hoc technique occupies 2% less area when compared to Simple BIST. Ad-hoc test will reduce test time compared to BIST. Furthermore, testing methods are to be developed in Reversible logic. We have built a design-for-testability technique for reversible circuits. The design is based on the partitioning of the gates in some sets, where each set is made associated to an additional line via ab extra control. This arrangement helps to test the each set of gates separately. Algorithms are proposed to decrease the number of additional lines and finding the complete test set. The design detects all missing gate faults. This design incurs less quantum cost overhead as compared to previous designs.

REFERENCES

[1] Krishna, G. Vamsi, G. Srinivasa Rao, and Y. Amar Babu. "An FPGA Implementation of Low Dynamic Power & Area Optimized 32-Bit Reversible ALU." International Journal of Applied Engineering Research 13.5 (2018): 2926-2932.

[2] C.H. Bennett, “Logical Reversibility of Computation”, IBM J. Research and Development, pp. 525-532, November 1973.

[3] C H Bennett, "Notes on the History of Reversible Computation", IBM Journal of Research and Development, vol. 32, pp. 16-23, 1998.

[4] A. Peres, “Reversible logic and quantum computers”, phys. rev.A, Gen.Phys., vol. 32, no. 6, pp. 32663276, Dec. 1985.

[5] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, ``Embedded deterministic test," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 23,no. 5, pp. 776_792, May 2004.

[6] F. Brglez, D. Bryan, K. Kozminski, "Combinational Profiles of sequential benchmark circuits", *Proc. IEEE ISCAS*, pp. 1929-1934, 1989.

[7] A. S. Abu-Issa and S. F. Quigley, “Bit-swapping LFSR and scan-chain ordering: A novel technique for peak- and average-power reduction in scan-based BIST,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 5, pp. 755–759, May 2009.

[8] V. D. Agrawal, C. R. Kime, and K. K. Saluja, “A tutorial on built-in self test. I. Principles,” *IEEE Des. Test Comput.*, vol. 10, no. 1, pp. 73–82,Mar. 1993.

[9] J. Rajski, J. Tyszer, G. Mrugalski, and B. Nadeau-Dostie, “Test generator with preselected toggling for low power built-in self-test,” in *Proc. Eur.Test Symp.*, May 2012, pp. 1–6.

[10] Y. Sato, S. Wang, T. Kato, K. Miyase, and S. Kajihara, “Low power BIST for scan-shift and capture power,” in *Proc. IEEE 21st Asian Test Symp.*, Nov. 2012, pp. 173–178.

[11] E. K. Moghaddam, J. Rajski, M. Kassab, and S. M. Reddy, “At-speedscan test with low switching activity,” in *Proc. IEEE VLSI Test Symp.*,Apr. 2010, pp. 177–182.

[12] A. Peres, “Reversible logic and quantum computers”, phys.rev.A, Gen.Phys., vol. 32, no. 6, pp. 32663276, Dec. 1985.

[13] J.M. Rabaey and M. Pedram, “Low Power Design Methodologies,” Kluwer Academic Publisher, 1997.

[14] T. Toffoli., “Reversible Computing”, Tech memo MIT/LCS/TM-151,MIT Lab for Computer Science 1980.

[15] M. L. Bushnell and V. D Agarwal, “Essentials of Electronic Testing” Kluwer academic Publishers, Norwell, MA, 2000.