Implementation of Cost-Effective Self-Healing Approach for Reliable Hardware Systems

S.G. Niveditha[1], Nelam Nagaraja Kumari[2]
[1] *Pursuing MTech (VLSD), dept. of ECE, Malineni Perumallu Educational Society's Group of Institutions, Pulladigunta, Vatticherukur, Prathipadu Road, Guntur, Andhra Pradesh*
[2]*Assistant Professor Dept. of ECE, Malineni Perumallu Educational Society's Group of Institutions, Pulladigunta, Vatticherukur, Prathipadu Road, Guntur, Andhra Pradesh*

***Abstract*- The advance of sub-micron technology has resulted in the adversity of VLSI testing. Analysis and architecture for testability are accustomed today as analytical to an acknowledged design. Field Programmable Gate Arrays (FPGAs) accept been acclimated in abounding areas of agenda design. Because FPGAs are reprogrammable, faults can be calmly adequate already accountability sites are located. This cardboard presents a new able self-healing arrangement that achieves 100% accountability advantage with low breadth overhead, and after any modification of the ambit beneath analysis (CUT) as 64-bit ALU, i.e., no analysis point insertion.**

Effective Self-Healing Approach for Reliable Hardware Systems. The proposed self-healing approach is applied to investigate for existing cell based and extension ALU array.

Self-healing system components which observe not only potential problems but can also take steps to continue operation under

Index Terms- Memory BIST, Address Generation, ALU-based implementation.

1. INTRODUCTION

Abnormal conditions. Whether due to long-term normal wear-and-tear or sudden combat damage. Self-healing is defined by the ability of a system to detect faults or failures and fix them. Fault detection and Correction functions are implemented to help protect the device when a severe fault occurs. Self-healing concept for hardware systems is investigated and a new approach is proposed.

Xilinx based simulation results obtained for a self-healing mechanism for circuit under test emphasize. To retain the true nature of the output in the event of occurrence of faults at the interconnect level of cascaded digital circuits. The operational pattern of the computational circuit facilitates the creation of a self-healing attribute and ensures the reliability of the digital architecture. The proposed scheme inserts faults randomly into the system at the interconnect levels and fosters to predict its behavior in response to a fault. The scheme be-hives the formation of a self-checking mechanism to aid in the process of analyzing the signals at different stages for faults.

1.1BIST

Built-In Self-Test (BIST) has become an accepted automated convenience for testing memories back anamnesis cores aggregate a above allotment of the die area. One of the key accoutrements of anamnesis BIST is the address artist (AG). In acclimation to ascertain speed-related faults, the address artist must achieve acclimatized address sequences to accede for acclimatized address transitions. Its complication is an above architecture issue, back it requires ample breadth and banned the BIST speed.

The capital drivers for the boundless development of BIST techniques are the fast-rising costs of ATE testing and the growing complication of chip circuits. It is now accepted to see circuitous accessories that accept functionally assorted blocks congenital on altered technologies central them. Such circuitous accessories crave high-end mixed-signal testers that access appropriate agenda and analog testing capabilities. BIST can be acclimated to accomplish these appropriate tests with added on-chip analysis circuits, eliminating the charge to access such high-end testers.

1.2DFT means Design-for Testability:

It is an alignment of IC architecture which simplifies added IC testing (like scan path admittance etc.)

BIST is a Design-for-Testability (DFT) technique, because it makes the electrical testing of a dent easier, faster, added efficient, and beneath costly. The abstraction of BIST is applicative to just about any affectionate of circuit, so its accomplishing can alter as broadly as the artefact assortment that it caters to.

1.3BIST is a Design-for-Testability (DFT):
Technique of designing added accouterments and software appearance into dent circuits to acquiesce them to accomplish self-testing, i.e., testing of their own operation (functionally, parametrically, or both) application their own circuits, thereby abbreviation assurance on an alien automatic analysis accessories (ATE). It makes the electrical testing easier, faster, added efficient, and beneath costly. The abstraction of BIST is applicative to just about any affectionate of circuit, so its accomplishing can alter as broadly as the artefact assortment that it caters to. As an example, an accepted BIST access for DRAM's includes the assimilation assimilate the dent of added circuits for arrangement generation, timing, approach selection, and go-/no-go analytic tests.

1.4 Disadvantages of implementing BIST include:

1. Additional silicon breadth and fab processing requirements for the BIST circuits.
2. Reduced admission times.
3. Additional pin (and possibly bigger amalgamation size) requirements, back the BIST chip charge a way to interface with the alfresco apple to be effective.
4. Possible issues with the definiteness of BIST results, back the on-chip testing accoutrements itself can fail.

2 PROPOSED WORK

The operation of the proposed address can be explained as follows: If an accountability is detected in a cell, the acquaintance corpuscle will accept an ascendancy arresting from ambassador to run this adulterated task. The alive corpuscle divides run time amid its assignment and adulterated beef assignment application a bi fold bend triggered" DET" cell. So, at the aboriginal bisected of the alarm aeon with the absolute bend the a live corpuscle run its aboriginal task. Figure1 Proposed cell structure shows the corpuscle anatomy of the proposed work. There are two inputs activated to DET corpuscle and alternative of one of them depends on the amount of ascendancy arresting (C) and alarm (clock) value. If an accountability happens, the apprehension block detects accountability area and the ascendancy block pulls the arresting C for this specific location. Thus, the DET corpuscle drives accustomed ascribe (I1) if clock amount rises and selects corpuscle (I2) if clock amount drops as presented in Figure2 Dual edge triggered circuits.

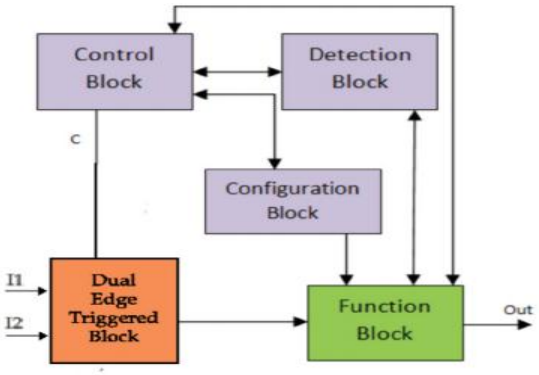

Figure1 Proposed cell structure

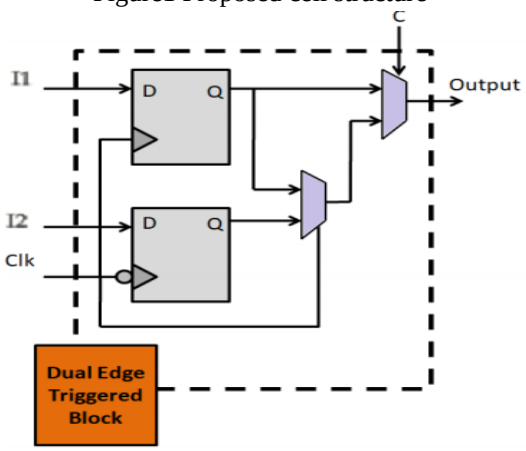

Figure2 Dual edge triggered circuits

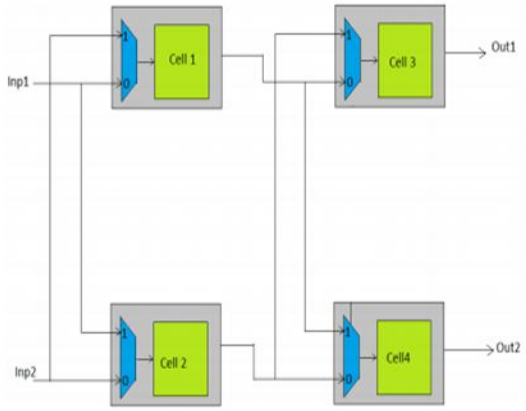

Figure3 Block diagram of the proposed technique.

Figure3 shows a simple arrangement to authenticate the idea. Assume there is a accountability in corpuscle 1, according to that the apprehension block it sends a arresting for the neighbors to analysis there activation. Now cell2 will atone the adulterated cell1 and ascendancy arresting C1 will be one. Cell2 has two inputs activated to DET's input, one comes from the accustomed ascribe and the additional which is declared to go the ascribe of cell1. The alternative ascribe of DET selects one ascribe to access the corpuscle 2 depending on the alarm value. Cell2 has the operation of corpuscle 1 assignment such that if cell2 receives an ascendancy arresting C1, the cell2 selects the operation of corpuscle 1 from agreement bore to be done. For the abrogating amount of alarm and whether C1 equals 1 or 0 the DET block selects input2 and runs an action of corpuscle 2. During the absolute amount of alarm and C1 equals 1, the DET selects input1 and runs an action of corpuscle 1. Also, the achievement of corpuscle 2 during the action time of corpuscle 1's assignment is fed to corpuscle 3. This will appear on any corpuscle breadth archetypal every corpuscle has the set of functions to be configured. So, instead of abacus additional beef and access breadth overhead, the self-healing is done on the arrangement application the alive beef itself. If there are assorted faults added alive corpuscle will atone faults.

2.1 Self-Healing on ALU

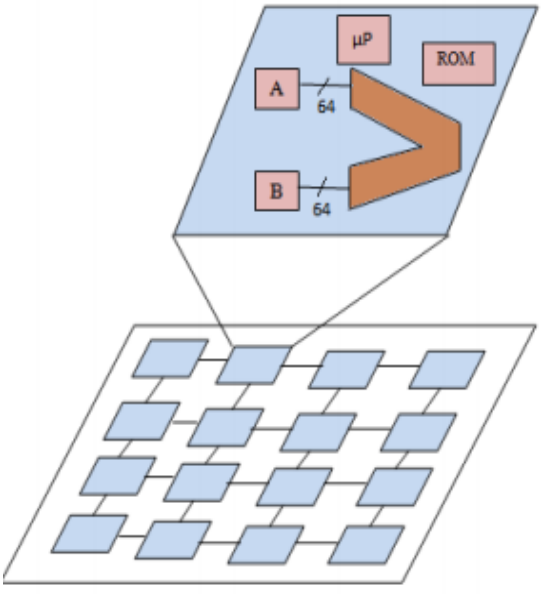

Figure4 The Architecture of cells include ALU operations

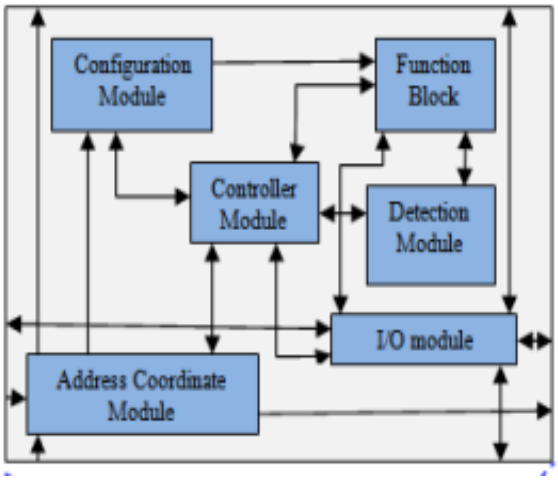

Figure5 Architecture of Embryonic Hardware

Figure5 shows accepted cellular structure, it is composed of six modules; I/O module, abode module, agreement module, ascendancy module, action module, and apprehension module. The proposed system simplifies the self-repairing mechanism and also it gives enough efficiency when number of hardware increases. While adding spare modules it is ensuring good fault-coverage. Self-repairing system composed of structural layer and gene control layer. Structural layer consists of both working cell (WC), spare cell (SC) and their interconnection. Gene control layer consist of Index Changing Unit (ICU), Differentiation Unit (DU). It determines the proper spare module in the structural layer to replace the faulty module without collision. This self-repairing mechanism is operated in parallel. So, even several faults occur in different modules at same time, the system can recover them.

2.2 Design of arithmetic logic unit

ALU was advised to accomplish the accession and analytic operations for the controller. Accession operations performed are the 64-bit addition, subtraction, and multiplication. Analytic operations performed are AND, OR, XOR, NAND, NOR, XNOR, NOT and Data Buffer. For designing the ALU, an adjustable architecture that consists of smaller, but added acquiescent blocks, some of which can be re-used. Designing of half-adder, 2-bit multiplier, 4-bit Brent-Kung, 4-bit multiplier, 8-bit Brent-Kung adder, 8-bit multiplier, 8-bit abounding adder, 8-bit subtractor, 64-bit Brent-Kung adder, 64-bit multiplier, 64-bit abounding adder, 64-bit

subtractor, 64-bit accession unit, analytic assemblage and 64-bit ALU has been done.

2.2.1 ARITHMETIC UNIT

ALU was designed to perform the arithmetic and logical operations for the controller. Arithmetic operations performed are the 64-bit addition, subtraction, and multiplication. Logical operations performed are AND, OR, XOR, NAND, NOR, XNOR, NOT and Data Buffer.

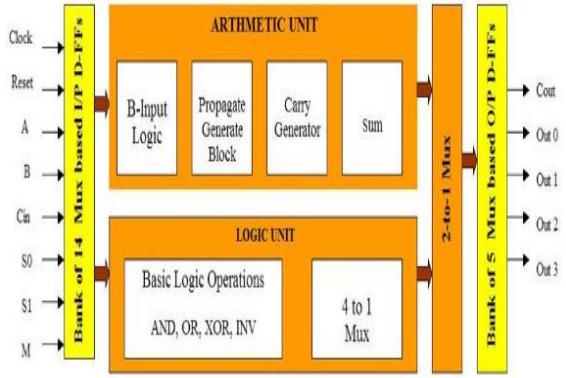

Figure 6 Block Diagram of Arithmetic Logic Unit

2.2.2LOGICAL UNIT

For designing of the analytic unit, the achievement of argumentation circuits accepts been analysed by employing the frequently acclimated argumentation gates and a multiplexer. An Argumentation assemblage does the assorted operations such as Analytic AND, OR, XOR, NOT, NAND, NOR, XNOR, and abstracts buffer. In accession to this addition assemblage and analytic unit, they accept been accumulated into the addition argumentation units. The schematic block diagram of an addition argumentation assemblage is apparent in Figure7, that is accessible. The achievement of the ALU and Analytic Assemblage is 64 bits. the ascendancy chat for ALU operations Everyman aisle adjournment for assorted multipliers. This area as well deals with the quantitative and allusive aftereffect assay of the altered access to Vedic mathematics through assorted multipliers and adder architecture and implementation. Additionally, to validate the proposed Vedic mathematics based assorted multipliers and adder designs and their implementation, the amalgam and the apish after-effects accept been compared with some added contemporary multiplier structures which are advised based on the altered multiplication algorithms.

3. SIMULATION RESULTS

The Simulation Output class contains all simulation outputs, including workspace variables. Test bench waveforms, which you have been using to simulate each of the modules you have designed so far. Schematic diagram Register-transfer-level (RTL) abstraction is used in hardware description language Verilog to create high-level representations of a circuit, from which lower-level representations and ultimately actual wiring was derived.

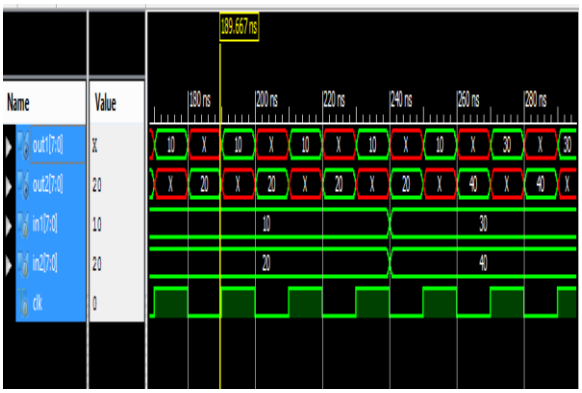

Figure 7 Simulation output of self healing buffer

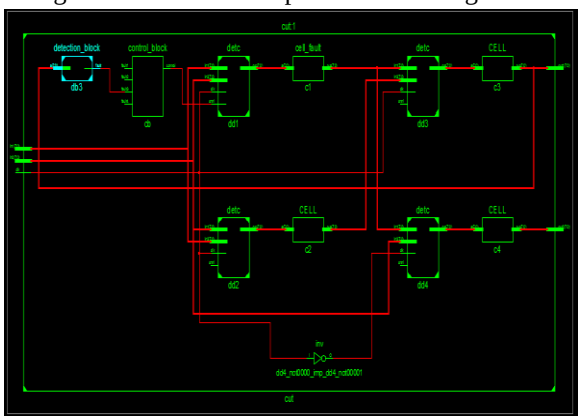

Figure 8 RTL schematic of proposed self healing buffer

Figure 9 Simulation output of 64-bit ALU

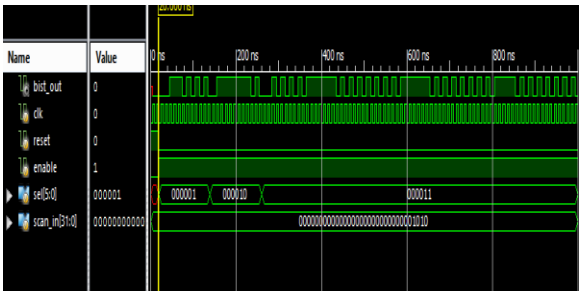

Figure 10 Simulation output of proposed self healing 64-bit ALU

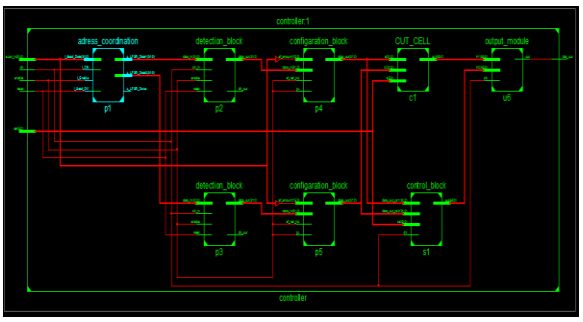

Figure 11 RTL schematic of proposed self healing 64-bit ALU

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	130	407600	0%
Number of Slice LUTs	6111	203800	2%
Number of fully used LUT-FF pairs	109	6132	1%
Number of bonded IOBs	42	500	8%
Number of BUFG/BUFGCTRLs/BUFHCEs	2	200	1%
Number of DSP48E1s	6	840	0%

Figure 12 Design summary of self-healing 64-bit ALU

```
Offset:             0.575ns (Levels of Logic = 1)
  Source:           u6/out (FF)
  Destination:      bist_out (PAD)
  Source Clock:     clk rising

  Data Path: u6/out to bist_out
                              Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FD:C->Q               1   0.236   0.339  u6/out (u6/out)
     OBUF:I->O                 0.000          bist_out_OBUF (bist_out)
    ----------------------------------------
    Total                      0.575ns (0.236ns logic, 0.339ns route)
                                       (41.0% logic, 59.0% route)
```

Figure 13 Time summary of self healing 64-bit ALU

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	2	--	--
Logic	0.000	5153	203800	3
Signals	0.000	5681	--	--
DSPs	0.000	6	840	1
IOs	0.000	42	500	8
Leakage	0.122			
Total	0.122			

Supply Source	Summary Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.000	0.068	0.000	0.068
Vccaux	1.800	0.028	0.000	0.028
Vcco18	1.800	0.001	0.000	0.001
Vccbram	1.000	0.001	0.000	0.001

Supply	Power (W)	Total	Dynamic	Quiescent
		0.122	0.000	0.122

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	1.8	84.8	25.2

Figure 14 Power summary of self healing 64 bit ALU

4 CONCLUSIONS

1. Self healing every block (cell) will perform as spare cell for its number and using time division multiplexing the active cell will perform its task operation and the task of the faulty cell. So, it gives the flexibility to repair 50% of cells without using spare cells.
2. Proposed cell-based approach integrates the ECC method and BISR.
3. The Extension approach is implemented for ALU array. ALU based self Healing system provides good scalability and fault coverage.
4. The proposed and Extension method is implemented using Verilog and the simulation results are obtained using ISE Xilinx 14.4

5.FUTURE SCOPE

1. It is anticipated that, with continued development and seamless integration of 'self-healing' materials into modern-day will open a new era of enhanced consumer experience, infrastructure maintenance as well as countless unprecedented and unique applications resembling those in science fictions.
2. Self-healing' technologies are being actively pursued across various industries and are being particularly highly valued by NASA for advancing space technologies to enhance system safety and reliability while lessening maintenance burden.

REFERENCES

[1] K. Khalil, O. Eldash, and M. Bayoumi, "A novel approach towards less area overhead self-healing hardware systems," in International Midwest Symposium on Circuits and Systems (MWSCAS). IEEE, 2017.

[2] Q.-Z. Zhou, X. Xie, J.-C. Nan, Y.-L. Xie, and S.-Y. Jiang, "Fault tolerant reconfigurable system with dual-module redundancy and dynamic reconfiguration," vol. 9, no. 2, 2011, pp. 167–173.

[3] M. R. Boesen, J. Madsen, and P. Pop, "Application-aware optimization of redundant resources for the reconfigurable self-healing eDNA hardware architecture," in 2011

NASA/ESA Conference on Adaptive Hardware and Systems (AHS), June 2011, pp. 66–73.

[4] K. Khalil, O. Eldash, and M. Bayoumi, "Self-healing router architecture for reliable network-on-chips," in IEEE International Conference on Electronics, Circuits and Systems (ICECS). IEEE, 2017.

[5] T. Koal, M. Ulbricht, P. Engelke, and H. T. Vierhaus, "On the feasibility of combining on-line-test and self repair for logic circuits," in 2013 IEEE 16th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS), April 2013, pp. 187–192.

[6] I. Ivan, C. Boja, and A. Zamfiroiu, "Self-healing for mobile applications," Journal of Mobile, Embedded and Distributed Systems, vol. 4, no. 2, pp. 96–106, 201