GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

A BUILT-IN SELF REAIR ANALYER FOR WORD-ORIENTED MEMORIES

O.Vignesh
Teaching Fellow, Department of ECE, Anna University Regional Office Coimbatore, India

ABSTRACT

A built-in self repair analyzer with the optimal repair rate for memory arrays with redundancy. The existing techniques use depth first search using a stack and a finite-state machine. Instead, our formulation for the circuit allows us to use the parallel prefix algorithm, it can be configured in various ways to meet area and test time requirements. The total area of our infrastructure is dominated by the number of content addressable memory entries to store the fault addresses, and it only grows quadratically with respect to the number of repair elements. The linear feedback shift register used to count the next state and also requires high transitions in BIST architecture. The content addressable memory is used to identify the fault address to store in must-repair analyzer. The proposed methods using the bit swapping linear feedback shift register to reduce both the transition and the power consumption. It requires only a single test, even in the worst case. By performing the must-repair analysis during the test, it selectively stores fault addresses, and the final analysis to find a solution is performed on the stored fault addresses. The infrastructure is also extended to support various types of word-oriented memories.

Keywords: *BIST, Parallel prefix algorithm Repair analysis.*

I. INTRODUCTION

A circuit is tested once and for all, with the hope that once the circuit is verified to be fault free it would not fail during its expected life-time, it is called off-line testing. However, this assumption does not hold for modern day ICs, based on deep sub-micron technology, because they may develop failures even during operation within expected life time. To cater to this problem sometimes redundant circuitry are kept on-chip which replace the faulty parts. To enable replacement of faulty circuitry, the ICs are tested before each time they startup. If a fault is found, a part of the circuit (having the fault) is replaced with a corresponding redundant circuit part (by re-adjusting connections). Testing a circuit every time before they startup, is called Built-In-Self-Test (BIST). Once BIST finds a fault, the readjustment in connections to replace the faulty part with a fault free one is a design problem.

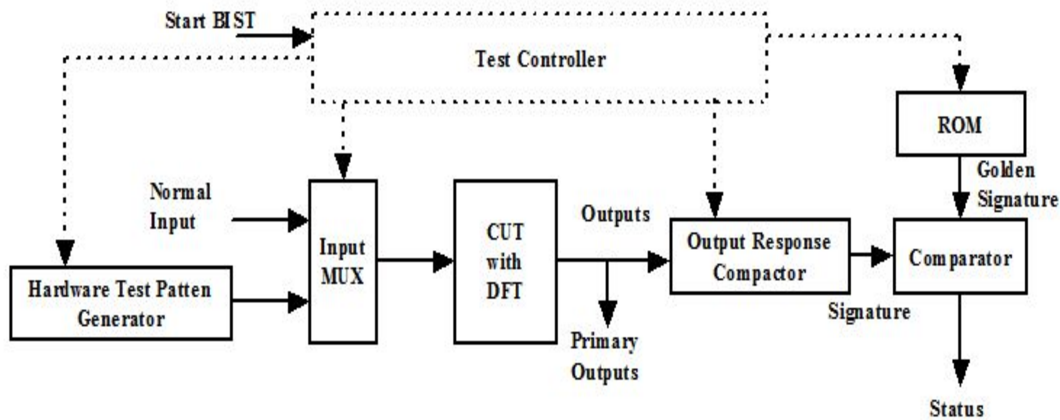

Fig 1: Basic Architecture of BIST

This module generates the test patterns required to sensitize the faults and propagate the effect to the outputs (of the CUT). As the test pattern generator is a circuit (not equipment) its area is limited. So storing and then generating test patterns obtained by ATPG algorithms on the CUT using the hardware test pattern generator is not feasible. In other words, the test pattern generator cannot be a memory where all test patters obtained by running ATPG algorithms (or random pattern generation algorithms) on the CUT are stored and applied during execution of the BIST. Instead, the test pattern generator is basically a type of register which generates random patterns which act as test patterns.

ISSN 2348 – 8034
Impact Factor- 3.155

The main emphasis of the register design is to have low area yet generate as many different patterns (from 0 to 2^n, if there are n flip-flops in the register) as possible.

This multiplexer is to allow normal inputs to the circuit when it is operational and test inputs from the pattern generator when BIST is executed. The control input of the multiplexer is fed by a central test controller. Output response compacter performs lossy compression of the outputs of the CUT. As in the case of off-line testing, in BIST the output of the CUT is to be compared with the expected response (called golden signature); if CUT output does not match the expected response, fault is detected. Similar to the situation for test pattern generator, expected output responses cannot be stored explicitly in a memory and compared with the responses of the CUT. So CUT response needs to be compacted such that comparisons with expected responses (golden signatures) become simpler in terms of area of the memory that stores the golden signatures.

II. MUST REPAIR ANALYZER FOR CAM

CRESTA has the sub-analyzers for all solution candidates, which provides the optimal repair rate with a single test. The sub-analyzer consists of a row content addressable memory (CAM) with entries and a column CAM with entries, and CRESTA requires sub-analyzers. Since this may not be affordable in memories with many spare elements, so have to reduce hardware complexity. Analyzing test and repair times for 2D integrated memory built-in test and repair, evaluates each possible solution one by one and thus does not require the parallel sub-analyzers. Such serial implementations may increase the overall test time, but the number of possible solutions is reduced using the must-repair analysis. Also the hardware complexity for the must-repair analysis is only quadratic with respect to the number of repair elements. The experimental results show that high repair rate can be obtained.

Our infrastructure provides the optimal repair rate with a single test as in CRESTA and has the same requirements for the number of CAM entries. Instead of a stack and a finite-state machine (FSM) used to enumerate all possible solutions , we propose a combinational circuit, which can be configured in various ways to meet the requirements for area and test time. For the fastest configuration, it can generate the next solution candidate in a single cycle.

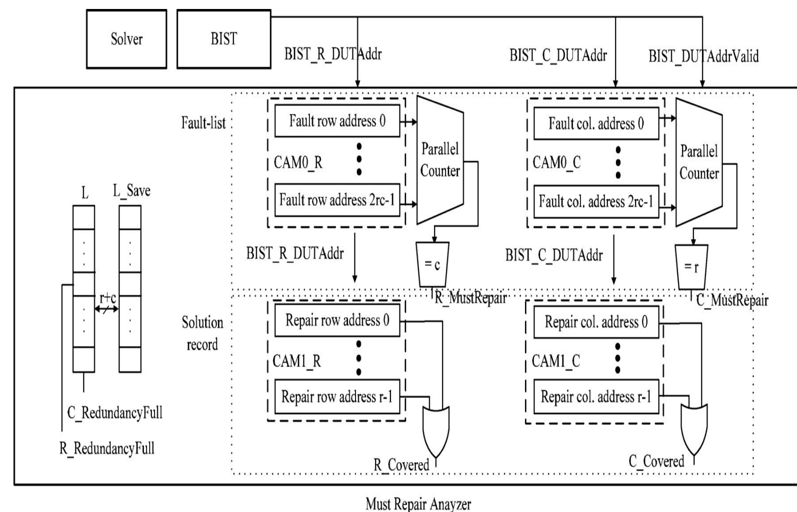

Fig 2: Must Repair Analyzer

The linear feedback shift register used to count the next state and also requires high transitions in BIST architecture. The proposed methods using the bit swapping linear feedback shift register to reduce both the transition and the power consumption and also using precomputation CAM to reduce the time delay in the MRA. Unlike most repair analysis work, we show that the proposed method can work for word-oriented memories. The Must Repair Analyzer (MRA) circuit diagram is as shown in Fig 1. It consists of a pair of CAMs for fault addresses, called the fault-list, and a pair of CAMs for a repair solution, called the solution record. The memory is repaired during testing by storing faulty addresses in registers. These addresses can be streamed out after test completion. Furthermore, the application can be started immediately after the memory BIST passes. In the fault-list, each CAM has one extra valid bit for each word, and the valid bits are initialized to "0" in the beginning. Since the CAMs assert "1" at the valid bit position for write and match operation, only written entries can be matched. During the test, if the BIST

engine detects a fault, it sends the fault address to the MRA on the fly through BIST_R_DUTAddr and BIST_C_DUTAddr, continues the test. The row (column) fault address is compared against row (column) CAM entries.

And the number of matched entries is efficiently counted by a parallel counter. If the number of the matched entries equals in the row (column) CAM, the row (column) indicated by the fault address satisfies the must-repair condition andR_MustRepair (C_MustRepair) signal is asserted. If the fault address triggers neither the row nor column must-repair condition, MRA writes the row and column address in the row and column CAMs, respectively. Due to Corollary 2, we can limit the size of the fault-list as 2, and if the overflow of the fault-list occurs, the memory array can be determined as unrepeatable, and the test can be terminated early. If a particular row or column is identified as must-repair, the row or column address must be part of the solution. Thus the MRA writes the row or column address in the solution record. The L registers are used as valid bits for the solution record.

III. SOLVER AND PRE-COMPUTATION CAM

The SOLVER has a register to store the cost of the current repair strategy, or Used Repair Elements. The SOLVER has a register to store the cost of the current repair strategy, or Used Repair Elements. The SOLVER also has registers to store the repair strategy with minimum cost so far and the minimum cost, or Repair Strategy Opt and Used Repair ElOpt. The current cost is compared against the minimum cost so far, which generates the Better signal. If the Better signal goes down during the evaluation of the current repair strategy, the SOLVER immediately asserts the RESTART signal and moves on to the next repair strategy. If the Better signal stays at "1" until the end of the evaluation, the SOLVER saves the current repair strategy and its cost. Since the SOLVER continues to search for a better solution even after finding a solution, the MRA may not have the optimal solution after the last repair strategy is evaluated. The Must-Repair Analysis repairs rows and columns for which there is no other choice.

To reduce area, we have stored the optimal repair strategy instead of the optimal solution. If the solution is directly stored, the size of the solution record should be doubled. Thus we need to recover the solution from the repair strategy, and the SOLVER goes into recovery phase. Since the SOLVER continues to search for a better solution even after finding a solution, the MRA may not have the optimal solution after the last repair strategy is evaluated. The solver and MRA circuit operation on shown in Fig 3.2, the MRA gives the fault address to solver. In the recovery phase, the optimal repair strategy, stored in the Repair Strategy Opt, goes into the Repair Strategy register and the strategy is evaluated again and the final analysis ends up with the optimal solution. The Final Analysis is a branch and bound search in the space of partial solutions, each node records the spare row and column assignments.

The BS-LFSR is combined with a scan-chain-ordering algorithm that orders the cells in a way that reduces the average and peak power (scan and capture) in the test cycle or while scanning out a response to a signature analyzer. These techniques have a substantial effect on average- and peak-power reductions with negligible effect on fault coverage or test application time. The BS-LFSR is combined with a scan-chain-ordering algorithm that orders the cells in a way that reduces the average and peak power (scan and capture).

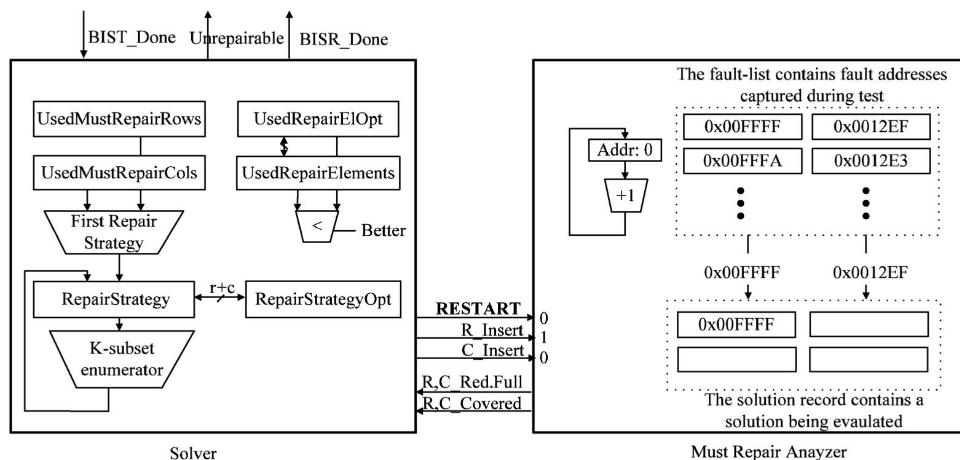

Fig 3 Solver Circuit

TABLE 1: Bit representations of repair strategies

Repair Strategy	Bit Representation
RRCC	1100
RCRC	1010
RCCR	1001
CRRC	0110
CRCR	0101
CCRR	0011

BIT SWAPPING LFSR

Low-transition linear feedback shifts register (LFSR) that is based on some new observations about the output sequence of a conventional LFSR. The proposed design, called bit-swapping LFSR (BS-LFSR), is composed of an LFSR and a 2 × 1multiplexer. When used to generate test patterns for scan-based built-in self-tests, it reduces the number of transitions that occur at the scan-chain input during scan shift operation by 50% when compared to those patterns produced by a conventional LFSR. Hence, it reduces the overall switching activity in the circuit under test during test applications. The proposed BS-LFSR reduces the average and instantaneous weighted switching activity (WSA) during test operation by reducing the number of transitions in the scan input of the CUT.

The BSLFSR is used together with the proposed scan-chain-ordering algorithm the average and peak powers are substantially reduced.

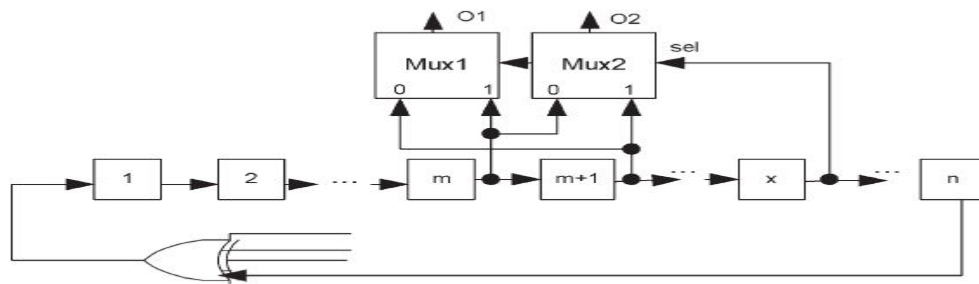

Fig 4 Bit Swapping LFSR

The bit-swapping linear feedback shift register (BS-LFSR), that is based on a simple bit swapping technique applied to the output sequence of a conventional LFSR and designed using a conventional LFSR and a 2 × 1 multiplexer. The BS-LFSR reduces the average and instantaneous weighted switching activity (WSA) during test operation by reducing the number of transitions in the scan input of the CUT. If the selection line is 0 the output will be swaped, if the selection line is 1 the output will be unswaped.
The BS-LFSR is combined with a scan-chain-ordering algorithm that reduces the switching activity in both the test cycle (capture power) and the scanning cycles (scanning power). These techniques have a substantial effect on average- and peak-power reductions with negligible effect on fault coverage or test application time. Experimental results on benchmark circuits show up to 65% and 55% reductions in average and peak power, respectively.

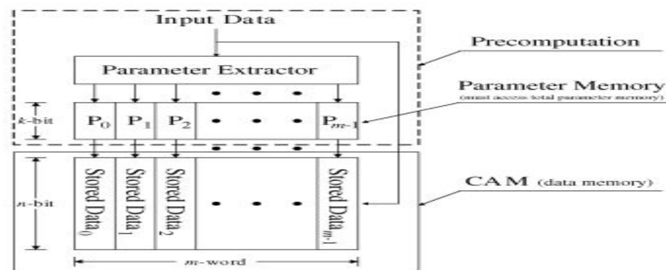

Fig 5 Precomputation CAM Circuit

ISSN 2348 – 8034
Impact Factor- 3.155

The Precomputation (Content Addressable Memory) CAM to reduce massive comparison operations for data searches, the operation is divided into two parts. In the first part, the parameter extractor extracts a parameter from the input data, which is then compared to parameters stored in parallel in the parameter memory. If no match is returned in the first part, it means that the input data mismatch the data related to the stored parameter. Otherwise, the data related to those stored parameters have to be compared in the second part. It should be noted that although the first part must access the entire parameter memory, the parameter memory is far smaller than that of the CAM (data memory). Moreover, since comparisons made in the first part have already filtered out the unmatched data, the second part only needs to compare the data that match from the first part. The PB-CAM exploits this characteristic to reduce the comparison operations, thereby saving power. Therefore, the parameter extractor of the PB-CAM is critical, because it deter-mines the number of comparison operations in the second part.

IV. RESULTS AND DISCUSSION

ERROR DETECTION

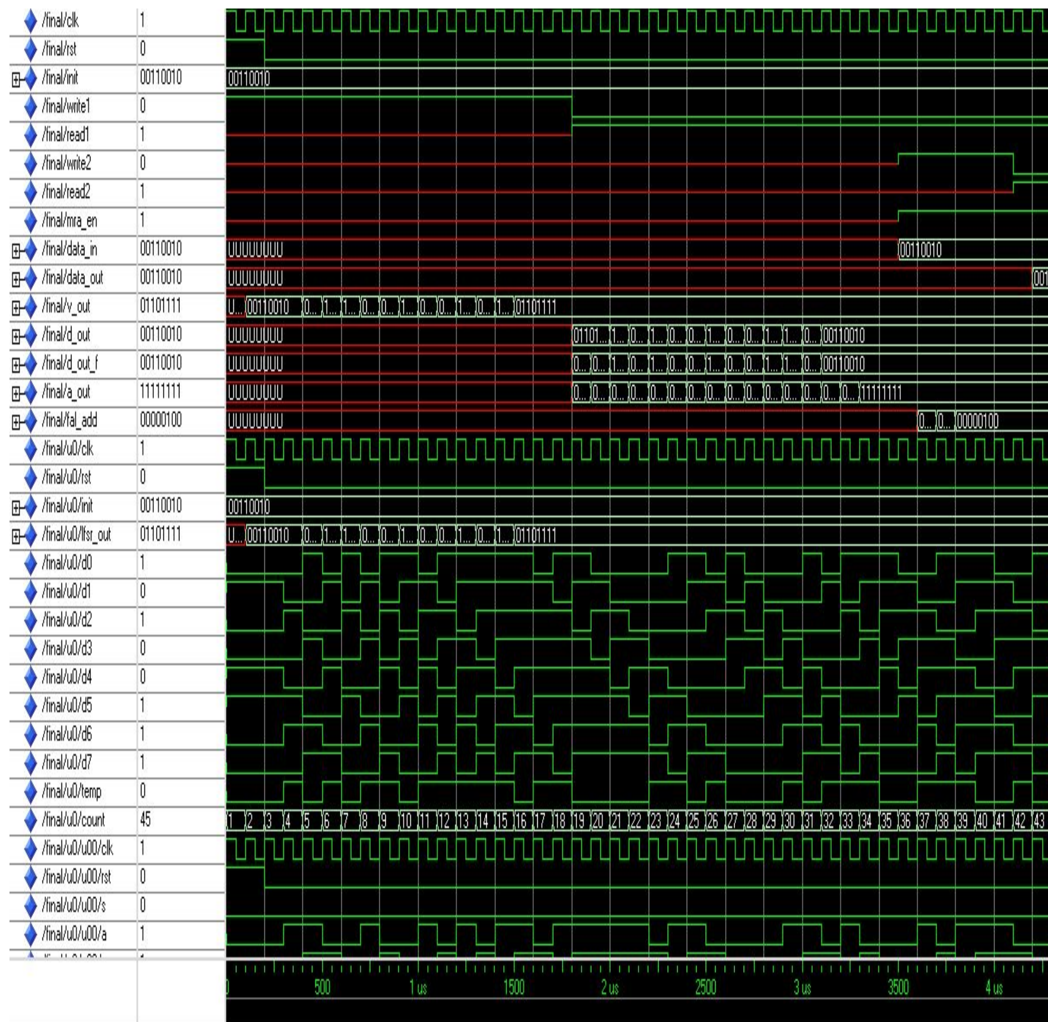

Fig 6 Simulation Output for Error Detection

VHDL is commonly used to write text models that describe a logic circuit. Such a model is processed by a synthesis program, only if it is part of the logic design. A simulation program is used to test the logic design using simulation models to represent the logic circuits that interface to the design. This collection of simulation models is commonly called a test bench.

VHDL has constructs to handle the parallelism inherent in hardware designs, but these constructs (processes) differ in syntax from the parallel constructs in Ada (tasks). Like Ada, VHDL is strongly typed and is not case sensitive. In order to directly represent operations which are common in hardware, there are many features of VHDL which are not found in Ada, such as an extended set of Boolean operators including nand and nor. VHDL also allows arrays to be indexed in either ascending or descending direction; both conventions are used in hardware, whereas in Ada and most programming languages only ascending indexing is available.

The MODELSIM simulation output of error detection as shown in Fig 6. First the input data to write give write-'1' to enable. After to read the input data give read-'1'to enable and write-'0'to disable. In the three data the last bit is error occurs, the fault data bits are shown in d-out-f and the input data bits are shown in d-out.

Error Correction

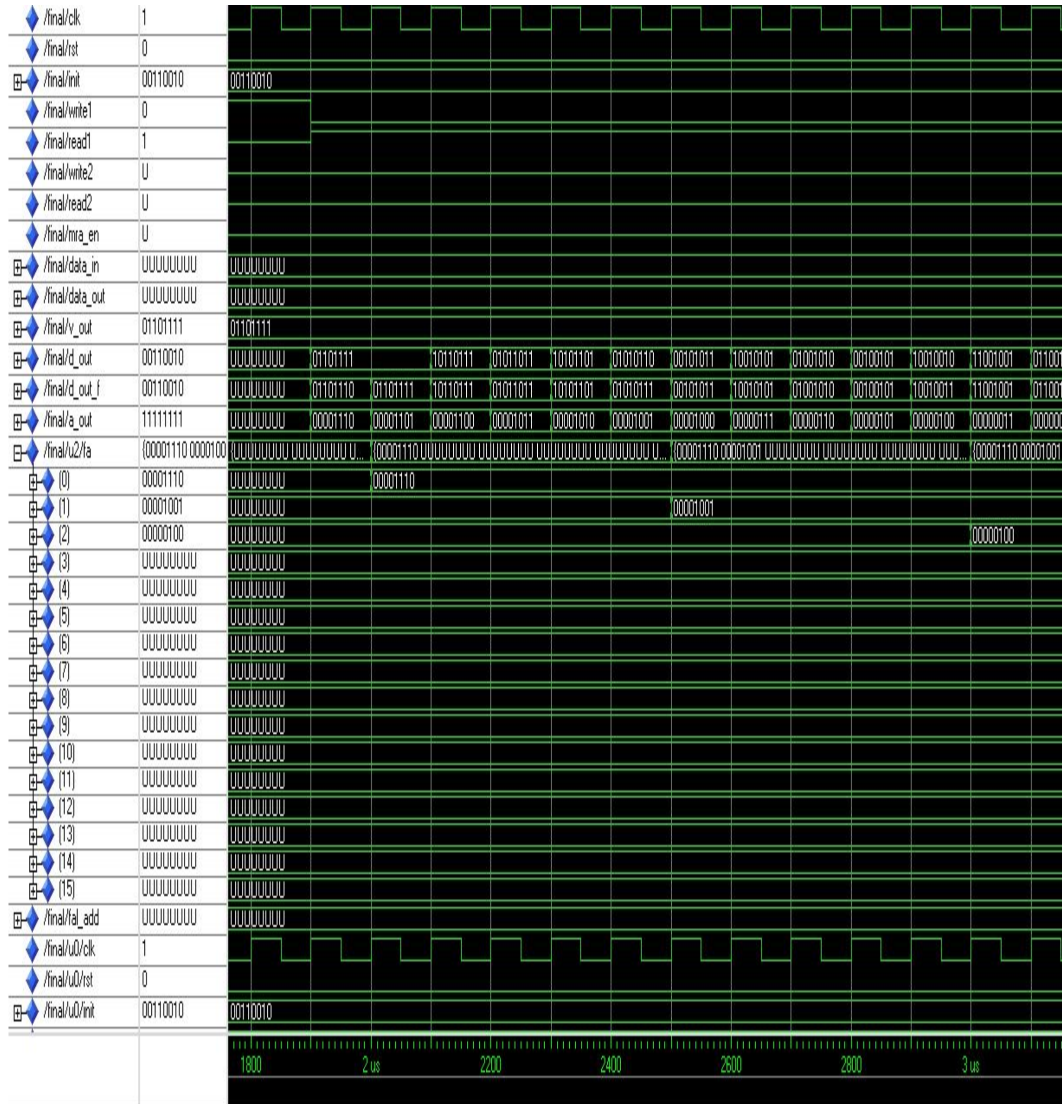

Fig 7 Simulation Output for Error Correction

The MODELSIM simulation output of error correction as shown in Fig 7. The fault data addresses are store in fault/u2/fa. Using the spare wires to correct the error bits and also again to read and write the data using give to write2-'0' and read2-'1'. Finally, to check the error correction output data bits shown in d-out.

Table 2 Comparison Table for Power Consumption

S.NO	PROCESS	POWER CONSUMPTION (mW)
1.	Must Repair Analyzer	373
2.	Modified Must Repair Analyzer	113

TIMING DELAY

Existing Timing Delay (MRA)

Timing Summary:

Speed Grade: -7

Minimum period: 8.313ns (Maximum Frequency: 120.301MHz)
Minimum input arrival time before clock: 10.945ns
Maximum output required time after clock: 9.475ns
Maximum combinational path delay: No path found

Proposed Timing Delay (Modified MRA)

Timing Summary:

Speed Grade: -7

Minimum period: 2.872ns (Maximum Frequency: 348.189MHz)
Minimum input arrival time before clock: 3.903ns
Maximum output required time after clock: 6.140ns
Maximum combinational path delay: No path found

COMPARISON FOR TIMING DELAY

Table 3 Comparison Table for Timing Delay

S.NO	PROCESS	Timing Delay (ns)
1.	Existing	8.313
2.	Proposed	2.872

V. CONCLUSION

We have proposed an on-chip infrastructure for repair analysis with the optimal repair rate for word oriented memories. The built in self repair analyzer used to detect and correct the errors in word oriented memories with single test. The linear feedback shift register used to count the next state and also requires high transitions in BIST architecture, required 373mW power. We proposed methods using the bit swapping linear feedback shift register to reduce both the transition and the power consumption as 113mW. The CAM circuit is used to identify the fault address in the MRA circuit, it's searching time delay is 8.313ns. The proposed method using precomputation CAM circuit time delay is reduced 2.872ns.This has been achieved by using low-cost on-chip selection mechanisms,

which are instrumental in very accurate and power reduction identification of failing rows, columns, and word oriented memories. This work implemented in FPGA kit.

REFERENCES

1. *W. Jeong, J. Lee, T. Han, K. Lee, and S. Kang, "An advanced BIRA for memories with an optimal repair rate and fast analysis speed by using a branch analyzer," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 29, no. 12, pp. 2014–2026, Dec. 2010.*
2. *W. Jeong, I. Kang, K. Jin, and S. Kang, "A fast built-in redundancy analysis for memories with optimal repair rate using a line-based search tree," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no.12, pp. 1665–1678, Dec. 2009.*
3. *P. Oehler, S. Hellebrand, and H.-J. Wunderlich, "Analyzing test and repair times for 2D integrated memory built-in test and repair," in Proc. Design Diag. Electron. Circuits Syst., 2007, pp. 1–6.*
4. *C.-T. Huang, C.-F. Wu, J.-F. Li, and C.-W. Wu, "Built-in redundancy analysis for memory yield improvement," IEEE Trans. Reliab., vol. 52,no. 4, pp. 386–399, Dec. 2003.*
5. *S.-Y. Kuo and W. Fuchs, "Efficient spare allocation for le arrays," IEEE Design Test Comput., vol. 4, no. 1, pp. 24–31, Feb. 1987. C.-T. Huang, C.-F. Wu, J.-F. Li, and C.-W. Wu, "Built-in redundancy analysis for memory yield improvement," IEEE Trans. Reliab., vol. 52, no. 4, pp. 386–399, Dec. 2003.*
6. *X. Du and W.-T. Cheng, "At-speed built-in self-repair analyzer to embedded word-oriented memories," in Proc. Int. Conf. VLSI Design, 2004 , pp. 895–900.*
7. *B. Fitzgerald and E. Thoma, "Circuit implementation of fusible redundant addresses on RAMs for productivity enhancement," IBM J. Res. Develop., vol. 24, no. 3, pp. 291–298, 1980.*