ISSN 2319-5991 www.ijerst.com
© 2026 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Research Paper

IMPLEMENTATION OF OPTIMIZED RECONFIGURABLE BUILT-IN SELF-REPAIR SCHEME FOR RAMS IN SOCS

[1]K. Gouthami gouthamis210@gmail.com
[2]Dr. N. Suresh[3] Ecehod@avniet.ac.in
Mr. Ch. Sreedhar Sreephd8@gmail.com

[1]Scholar, Master of Technology, Dept of Electronics and Communication Engineering, AVN Institute of Engineering and Technology Koheda Road M.P.Patel Guda Post Ibrahimpatnam, Dist., Hyderabad, Telangana 501510
[2]Professor& Hod, AVN Institute of Engineering and Technology Koheda Road M.P.Patel Guda Post Ibrahimpatnam, Dist., Hyderabad, Telangana 501510
[3]Assistant Professor, Dept of Electronics and Communication Engineering, AVN Institute of Engineering and Technology Koheda Road M.P.Patel Guda Post Ibrahimpatnam, Dist., Hyderabad, Telangana 501510

ABSTRACT

Random Access Memory (RAM) is a critical component of modern System-on-Chip (SoC) designs, and its reliability directly affects overall chip performance and manufacturing yield. As VLSI technology advances, memory circuits become more susceptible to manufacturing defects and operational faults, making efficient testing and repair mechanisms essential. Conventional Built-In Self-Test (BIST) techniques can detect memory faults but may require additional hardware resources and may not provide an effective repair mechanism for different memory configurations.

This project presents an optimized Reconfigurable Built-In Self-Repair (ReBISR) scheme for embedded RAMs in SoCs. The proposed approach integrates memory testing, fault detection, redundancy analysis, and repair configuration to improve memory reliability. A reconfigurable Built-In Redundancy Analysis (ReBIRA) circuit is designed to support different RAM sizes and redundancy organizations, while an adaptive fusing methodology aims to reduce repair setup time. The document also describes Cellular Automata Linear Feedback Shift Register (CA-LFSR)-based test pattern generation as an alternative to conventional LFSR-based testing, along with FPGA-based design verification using hardware description languages and simulation tools.

The proposed approach aims to reduce hardware area overhead, improve fault detection and repair efficiency, and increase memory yield and reliability. It can be useful in embedded systems, automotive electronics, consumer devices, aerospace applications, and other SoC-based systems where dependable memory operation is essential. Overall, the project emphasizes the importance of reconfigurable memory testing and self-repair techniques in developing reliable and cost-effective VLSI systems.

Keywords:

System-on-Chip (SoC), Random Access Memory (RAM), Built-In Self-Test (BIST), Built-In Self-Repair (BISR), Reconfigurable Built-In Redundancy Analysis (ReBIRA), Cellular Automata LFSR, VLSI, Fault Detection, Memory Repair.

I. INTRODUCTION

System-on-Chip (SoC) technology integrates multiple functional components, such as

processors, memory units, controllers, and communication interfaces, into a single integrated circuit. Random Access Memory (RAM) is one of the most important components in an SoC because it stores data and instructions required for system operation. As VLSI technology advances, the complexity and density of integrated circuits continue to increase, making memory testing and fault management important challenges in semiconductor design. Manufacturing defects in memory cells, word lines, and bit lines can affect system reliability and reduce the manufacturing yield of the chip. Therefore, efficient memory testing and repair techniques are required to improve the quality and dependability of modern electronic systems.

Built-In Self-Test (BIST) is a widely used Design-for-Testability (DFT) technique that incorporates testing capabilities directly into an integrated circuit. It uses a Test Pattern Generator (TPG) to generate test patterns, applies them to the Circuit Under Test (CUT), and analyzes the output responses to identify faults. Although conventional BIST techniques reduce dependence on external testing equipment and help improve fault detection, they may introduce additional hardware overhead and may not independently repair defective memory cells. Built-In Self-Repair (BISR) addresses this limitation by combining fault detection with redundancy-based repair mechanisms. Spare rows or columns can be allocated to replace defective memory locations, thereby improving the usability and reliability of embedded RAMs.

The project, "Implementation of Optimized Reconfigurable Built-in Self-Repair Scheme for RAMs in SoCs," focuses on developing a reconfigurable memory testing and repair architecture that supports RAMs with different sizes and redundancy organizations. The proposed ReBISR scheme incorporates a Reconfigurable Built-In Redundancy Analysis (ReBIRA) circuit to analyze detected faults and allocate available redundant resources efficiently. An adaptive fusing methodology is also considered to reduce repair setup time during normal memory operation. In addition, the document discusses Cellular Automata Linear Feedback Shift Register (CA-LFSR)-based test pattern generation as an alternative to conventional LFSR-based approaches. The design and verification process uses hardware description languages, simulation, and FPGA development tools. The overall objective is to reduce hardware area overhead, improve memory repair efficiency, and enhance the yield and reliability of SoC designs.

II. LITERATURE REVIEW

The increasing integration of embedded memories in System-on-Chip (SoC) designs has made memory testing, fault diagnosis, and repair essential for improving manufacturing yield and system reliability. Tseng, Li, and Hsu proposed a Reconfigurable Built-In Self-Repair (ReBISR) scheme that supports multiple RAMs with different sizes and redundancy organizations. Their approach uses a reconfigurable Built-In Redundancy Analysis (ReBIRA) circuit to allocate spare memory resources efficiently and an adaptive fusing methodology to reduce repair setup time [1]. Similarly, Shekar Babu et al. presented a BISR architecture for multiple RAMs in an SoC, emphasizing the use of spare rows, spare columns, and spare cells to improve repair efficiency [2]. Sujana and Asha Rani investigated redundancy analyzermodeling for RAMs, where fault information obtained from memory testing is analyzed to select suitable redundant elements [3]. These studies demonstrate that effective redundancy allocation and reconfigurable repair architectures can improve memory yield while controlling hardware overhead.

Further research has explored different repair strategies, memory fault diagnosis techniques, and optimized redundancy analysis algorithms. Tseng and colleagues introduced a defect-aware BISR scheme for single-port and multiport RAMs, focusing on identifying defects and improving repair efficiency through specialized defect-location algorithms [4]. Research on built-in self-repair for flash memories has also demonstrated the importance of integrating self-test, redundancy analysis, and on-chip repair to address manufacturing defects in nonvolatile memory [5]. Karthy and Sivakumar investigated an optimized counting-threshold built-in redundancy analysis method to improve memory repair decisions and address yield-related challenges [6]. In addition, Siva Mangai discussed a redundancy-based BISR architecture for SRAM that combines Built-In Self-Test (BIST) and Built-In Redundancy Analysis (BIRA) to handle memory cell failures [7]. Collectively, these approaches highlight the need for accurate fault identification, efficient redundancy utilization, and reduced repair latency in modern memory systems.

Test-pattern generation is another important aspect of memory testing because the quality of the generated patterns influences fault coverage and test efficiency. Abraitis and Tamoševičius studied LFSR- and cellular-automata-based techniques for delay testing in ASICs and FPGAs, examining their effectiveness in generating test sequences [8]. Fišer and Kubátová analyzed pseudorandom test-pattern generators, including LFSR and cellular automata approaches, and discussed their influence on fault coverage [9]. More recently, Sontakke and Atchina reviewed memory built-in self-repair and correction techniques, covering redundancy-based repair, built-in repair analysis, and error-correction methods for improving memory yield and reliability [10]. Based on these studies, the present project focuses on an optimized and reconfigurable memory testing and repair architecture, with efficient redundancy allocation and consideration of Cellular Automata LFSR-based test-pattern generation. The goal is to improve repair efficiency, reduce hardware overhead, and support reliable RAM operation in SoC designs.

III. EXISTING SYSTEM

The existing system for “Implementation of Optimized Reconfigurable Built-in Self-Repair Scheme for RAMs in SoCs” primarily relies on conventional Built-In Self-Test (BIST) techniques to identify faults in embedded Random Access Memories (RAMs). In this approach, a Test Pattern Generator (TPG) generates test patterns and applies them to the memory under test. The output responses are then analyzed to detect errors caused by defective memory cells, word lines, or bit lines. Conventional BIST helps reduce the need for external testing equipment and supports on-chip memory testing. However, fault detection alone does not necessarily repair defective memory locations.

Traditional Built-In Self-Repair (BISR) systems use redundancy, such as spare rows or columns, to replace faulty memory locations. A redundancy analysis circuit determines whether the available spare resources can repair the detected faults. However, conventional approaches may have limitations when supporting RAMs of different sizes and redundancy configurations. They can also require additional hardware resources and repair-configuration time, potentially increasing area overhead and affecting repair efficiency.

IV. PROPOSED SYSTEM

The proposed system implements an Optimized Reconfigurable Built-In Self-Repair (ReBISR) scheme for embedded

RAMs in System-on-Chip (SoC) designs. It integrates memory testing, fault detection, redundancy analysis, and repair configuration into a reconfigurable architecture. Unlike conventional approaches that may require separate repair circuits for different memories, the proposed scheme is designed to support RAMs of different sizes and redundancy organizations using a Reconfigurable Built-In Redundancy Analysis (ReBIRA) circuit.

Initially, the Built-In Self-Test (BIST) module tests the RAM using test patterns and identifies faulty memory locations. The detected fault information is stored and passed to the redundancy analysis circuit. The ReBIRA circuit analyzes the faults and allocates available spare rows or columns to replace defective memory elements. After the repair decision is made, the repair information is stored in the appropriate configuration registers or fuse structures so that faulty locations can be redirected to available redundant elements.

The system also uses an adaptively reconfigurable fusing methodology to reduce repair setup time. By configuring the repair mechanism according to the memory being tested, the design aims to reduce unnecessary hardware overhead and improve repair efficiency. The overall objective is to improve RAM reliability and SoC manufacturing yield while supporting multiple memory configurations. The research paper reports low area overhead and reduced repair setup time for its evaluated configurations.

V. METHODOLOGY

The methodology for the Implementation of Optimized Reconfigurable Built-in Self-Repair Scheme for RAMs in SoCs consists of memory testing, fault detection, redundancy analysis, and repair configuration. The proposed approach is designed to identify defective memory locations and use available redundant memory resources to improve the reliability of embedded RAMs.

1. System Design

The system architecture is designed with a Built-In Self-Test (BIST) module, RAM under test, fault information storage, a Reconfigurable Built-In Redundancy Analysis (ReBIRA) circuit, and a repair configuration unit. These components work together to detect memory faults and determine suitable repair actions.

2. Test Pattern Generation

The BIST module generates test patterns and applies them to the RAM under test. These patterns exercise memory cells and help identify faults during read and write operations. The document also discusses Cellular Automata Linear Feedback Shift Register (CA-LFSR)-based test-pattern generation as an alternative to conventional LFSR-based generation.

3. Fault Detection

The RAM responds to the applied test patterns, and its output is compared with the expected response. Any mismatch indicates a potential memory fault. The detected fault information is recorded and forwarded to the redundancy analysis module for further processing.

4. Redundancy Analysis

The ReBIRA circuit analyzes the detected faulty locations and determines whether available spare rows or columns can be assigned to repair them. Its reconfigurable design aims to support different RAM sizes and redundancy organizations while making efficient use of the available repair resources.

5. Repair Configuration

After redundancy analysis, the selected spare rows or columns are assigned to replace the defective memory locations. The repair information is stored in the repair registers or appropriate fuse structures. During normal memory operation, faulty addresses are

redirected to their corresponding spare locations when the repair configuration supports the required mapping.

6. Adaptive Reconfiguration

The architecture configures the redundancy analysis and repair process according to the RAM being tested. The adaptive fusing methodology aims to reduce repair setup time, particularly when multiple RAMs with different configurations are integrated into an SoC.

7. Simulation and Verification

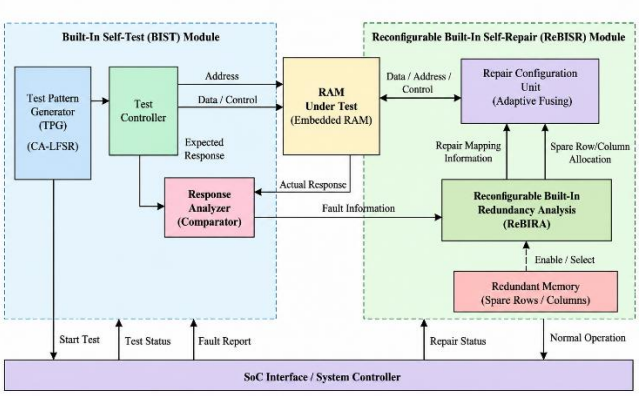
The design can be described using a hardware description language such as Verilog or VHDL and verified using simulation tools. Test cases can be used to evaluate fault detection, redundancy allocation, repair decisions, and correct memory operation after repair. If FPGA implementation is included in the actual project, synthesis and implementation results can also be analyzed for hardware resource usage and timing.

8. Performance Evaluation

The proposed system is evaluated by considering repair rate, hardware area overhead, redundancy analysis time, and repair setup time. These measurements help determine whether the reconfigurable architecture improves repair efficiency compared with a conventional BISR design. The original research reports an area cost of approximately 2.7% for its evaluated four-RAM configuration and an approximately 86.94% reduction in repair setup time for a specified multi-RAM configuration. These are published-paper results, not measurements of a separate implementation.

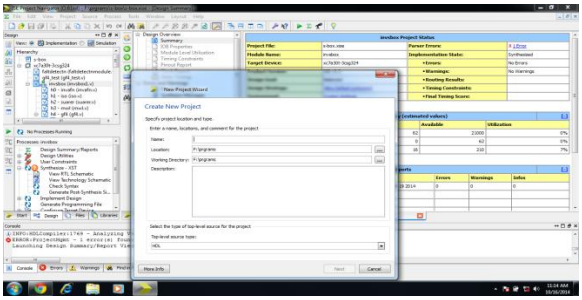
VI. SYSTEM MODEL

System Architecture

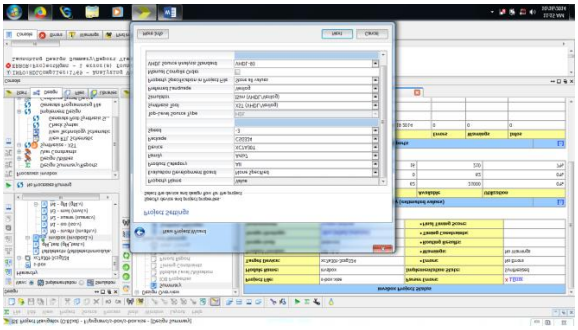


VII. RESULTS AND DISCUSSIONS

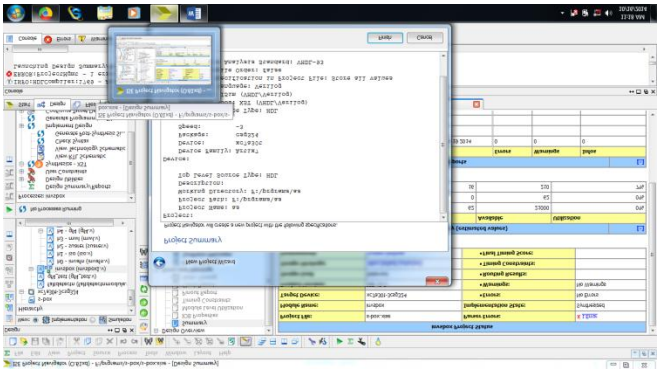
must be entered in the dialog box to proceed.



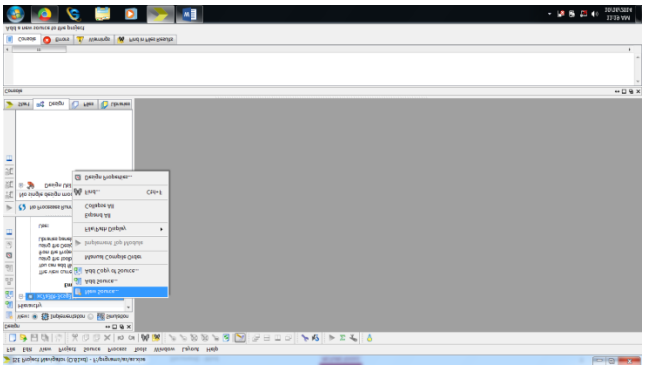
2.The second dialog box is the Device Settings dialog box, wherein users can specify the device configuration and project settings. Press Next to continue.



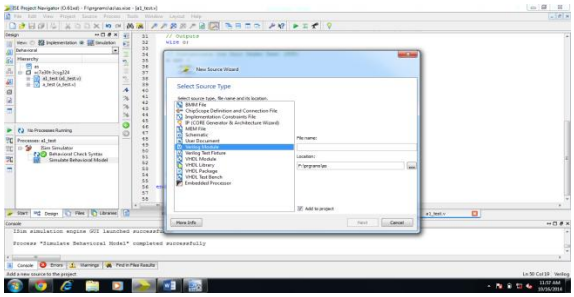
3.A summary dialog box appears holding the project description. At this point, click Finish, and the new project is created.



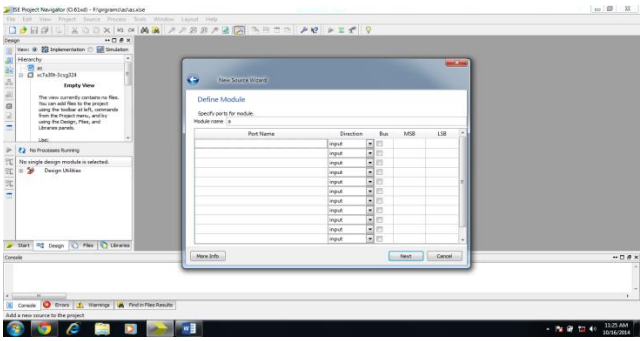
4. Following the construction of the project, the user may add some Verilog files. Right-click on the project name to view options.



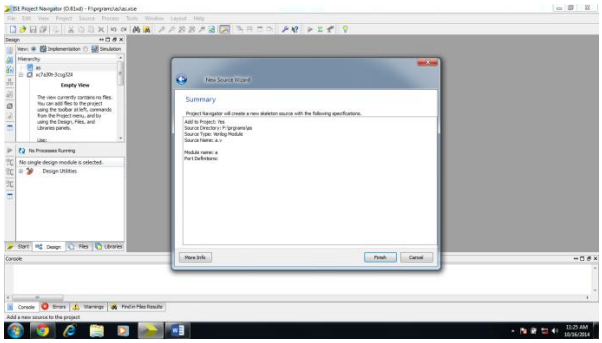
5. Choose New Source from the list. In the appearing dialog box, select Verilog Module for file types, type in the filename, click Next.



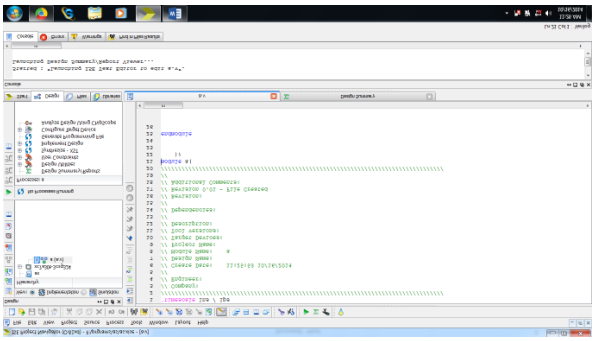
6.The tool then prompts for inputs, outputs, and inouts, which the user may specify at this point or within the Verilog code later. Having made the selections, click on the Next button.



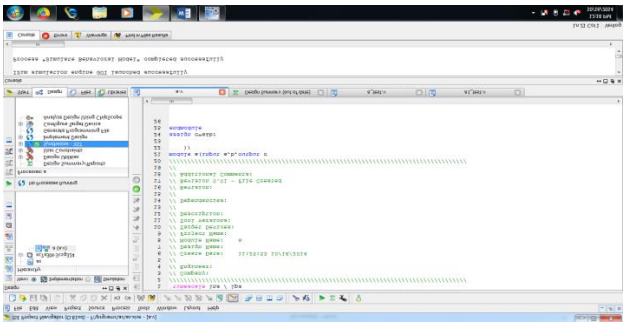
6. A last confirmation box pops up with the details of the newly created file. Finally, hit the Next button and the new file will be created.



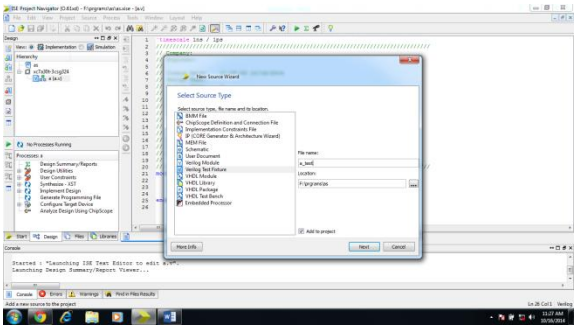
8.In the project window will most probably appear a blank editor associated with the file. Double-click the filename on the editor window to open it and begin writing Verilog code.



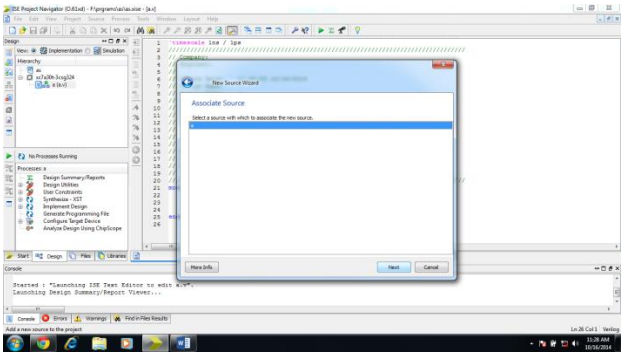
9.After coding is complete, the file must be selected and Synthesis run for syntax and design errors to be checked, with the results appearing in the message window below.



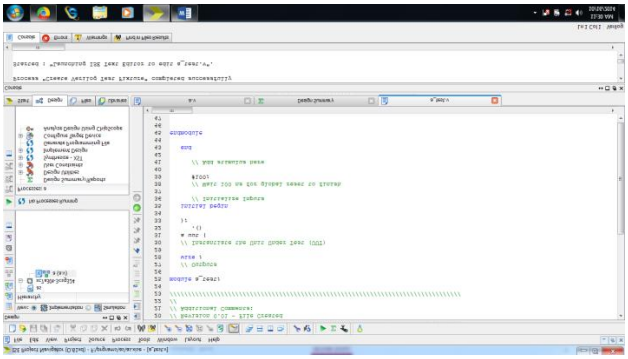
10.Upon successful synthesis, a testbench file must be created. Right-click the project and select New Source, and then name the testbench file.



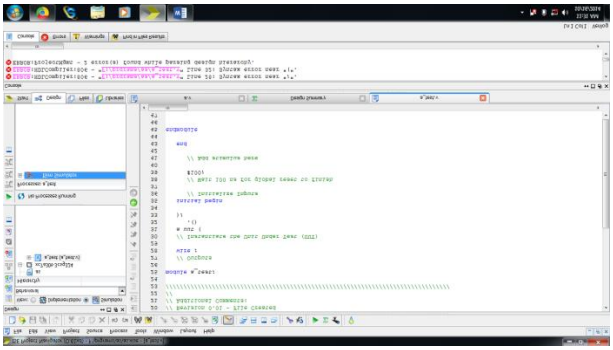
11.In case there are multiple design files, select the one on which the testbench is being created; afterward, click Next.



12.The new testbench file becomes part of the project window. Open it and specify the input stimulus needed for verification.

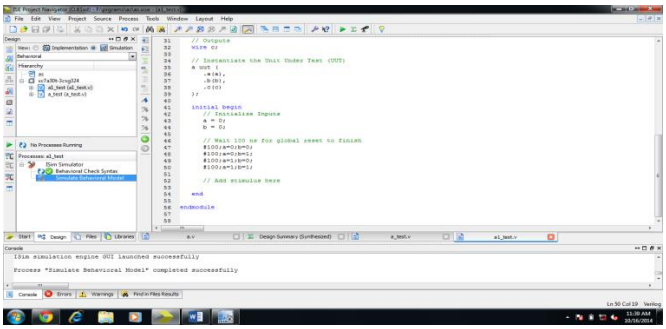


13.In preparation for testbench verification, change the view in the project navigator from implementation to Simulation.

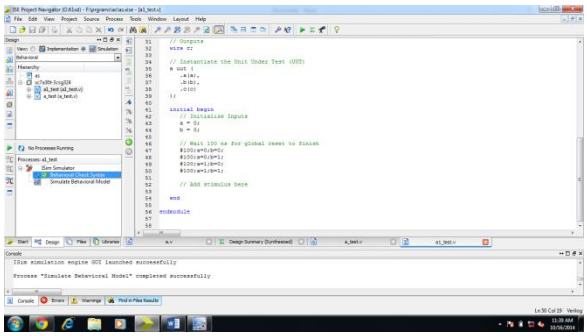


14. In the hierarchy, expand ISim Simulator, then run Behavioral Check Syntax so that you make sure the testbench has no errors.

15. Once the verification has succeeded, Simulate Behavioral Model can run to produce waveforms according to the input from the testbench and the behavior of the design.



16.Zooming in and out of the waveform window allows for detailed analysis of signal transitions to verify that the circuit works according to its intent.



VIII. CONCLUSION

The project, “Implementation of Optimized Reconfigurable Built-in Self-Repair Scheme for RAMs in SoCs,” focuses on improving the reliability and manufacturing yield of embedded RAMs through an optimized Reconfigurable Built-In Self-Repair (ReBISR) architecture. The proposed system integrates Built-In Self-Test (BIST), fault detection, redundancy analysis, and repair configuration to identify defective memory locations and replace them using available spare rows or columns.

The Reconfigurable Built-In Redundancy Analysis (ReBIRA) circuit is designed to support RAMs with different sizes and redundancy configurations, while the adaptive fusing methodology aims to reduce repair setup time. By efficiently utilizing redundant memory resources, the proposed approach aims to reduce hardware area overhead and improve memory repair efficiency compared with conventional approaches.

Overall, the project demonstrates the importance of reconfigurable memory testing and self-repair techniques in modern System-on-Chip (SoC) designs. Such techniques can contribute to more reliable and cost-effective VLSI systems. The effectiveness of the implementation should ultimately be confirmed through simulation results and measurements of fault detection, repair success, hardware utilization, and repair setup time.

IX. FUTURE WORK

In the future, the proposed Reconfigurable Built-In Self-Repair (ReBISR) scheme can be enhanced to support larger and more complex System-on-Chip (SoC) designs containing multiple RAMs with different sizes and redundancy configurations. The Reconfigurable Built-In Redundancy Analysis (ReBIRA) circuit can be further optimized to improve fault analysis speed, redundancy allocation, and repair efficiency while minimizing hardware area overhead.

The system can also be extended to support advanced memory fault models and more complex defect patterns. Improved test-pattern generation techniques, including optimized Cellular Automata Linear Feedback Shift Register (CA-LFSR) methods, could be explored to enhance fault coverage and reduce testing time. Furthermore, the adaptive fusing mechanism can be improved to reduce repair setup time and support flexible repair configurations.

Future research can focus on implementing and evaluating the design on advanced FPGA platforms and modern VLSI technologies. Performance comparisons can be conducted using metrics such as fault coverage, repair success rate, area utilization, power consumption, and repair setup time. These improvements could help develop scalable, cost-effective, and reliable memory testing and self-repair solutions for future SoC applications.

X. REFERENCES

[1] T. W. Tseng, J. F. Li, and C. C. Hsu, “ReBISR: A Reconfigurable Built-In Self-Repair Scheme for Random Access Memories in SoCs,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 6, pp. 921–932, 2010. DOI: 10.1109/TVLSI.2009.2017906.

[2] M. Shekar Babu, R. Sumanth Kumar Reddy, and S. V. V. Sateesh, “Built-In Self-Repair for Multiple RAMs with Different Redundancies in a SOC,” International Journal of Computer Applications, vol. 24, no. 8, pp. 26–29, 2011. DOI: 10.5120/2971-3995.

[3] R. Sujana and M. Asha Rani, “Modeling of Redundancy Analyzer in Built In Self Repair for RAMs,” Journal of Innovation in Electronics and Communication Engineering, vol. 2, no. 2, pp. 27–31, 2012.

[4] T. W. Tseng, Y. J. Huang, and J. F. Li, “DABISR: A Defect-Aware Built-In Self-Repair Scheme for Single/Multi-Port RAMs in SoCs,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2010. Research record.

[5] “Built-In Self-Repair Schemes for Flash Memories,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 29, no. 8, pp. 1243–1256, 2010. DOI: 10.1109/TCAD.2010.2049051.

[6] G. Karthy and P. Sivakumar, "Optimized Counting Threshold Built-in Redundancy Analysis for Memories," Microprocessors and Microsystems, vol. 81, article 103682, 2021. DOI: 10.1016/j.micpro.2020.103682.
[7] N. M. Siva Mangai, "Built In Self Repair Architecture for SRAM Using Redundancy Logic," IETE Journal of Education, pp. 50–59. DOI: 10.1080/09747338.2013.10876105.
[8] V. Abraitis and Ž. Tamoševičius, "LSFR and BIST Based Delay Test for ASIC and FPGA," ElektronikairElektrotechnika, vol. 87, no. 7, pp. 45–48, 2008. DOI: 10.5755/j02.eie.11209.
[9] P. Fišer and H. Kubátová, "Pseudorandom Testing – A Study of the Effect of the Generator Type," Acta Polytechnica, vol. 45, no. 2, 2005. DOI: 10.14311/686.
[10] V. Sontakke and D. Atchina, "Memory Built-in Self-Repair and Correction for Improving Yield: A Review," International Journal of Electrical and Computer Engineering, vol. 14, no. 1, pp. 140–156, 2024. DOI: 10.11591/ijece.v14i1.pp140-156