Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 3 (2025)

SMS Memory Built in Self Test Verification

Dr. Sandeep Santosh

Department of Electronics and Communication, National Institute of Technology, Kurukshetra, Haryana, India.

Email: profsandeepkkr@gmail.com

Article History:

Received: 19-03-2025

Revised: 24-07-2025

Accepted: 21-09-2025

Abstract:

In today's VLSI SoC designs, embedded memories dominate die area and often limit manufacturing yield. Design-for-Testability (DFT) is crucial for detecting manufacturing defects early and ensuring robust design validation. Among DFT techniques, Memory Built-In Self-Test (MBIST) plays a pivotal role by embedding test and repair logic within memory subsystems. MBIST facilitates detection of memory faults such as stuck-at, transition, coupling, and retention faults, using March algorithms and fault models.

Keywords: STAR Memory System (SMS), Design-for-Testability (DFT), Memory Built-In Self-Test (MBIST).

1 Introduction

The STAR Memory System (SMS) by Synopsys provides an integrated, scalable, and silicon-proven framework for self-test and repair of embedded memories. This paper discusses its architecture, operation, and verification flow, emphasizing the relevance of high-quality memory testing in achieving time-to-yield and production efficiency.

Memory Built-In Self-Test (MBIST) is an essential technique used for the testing and verification of memory components in electronic systems. In modern electronic devices, memory plays a critical role in storing and retrieving data. Ensuring the reliability and functionality of memory units is crucial, especially in high-performance systems such as processors, embedded systems, and complex digital circuits. MBIST refers to the ability of a memory system to test itself without the need for external test equipment or external devices. This self-testing process allows the memory to detect faults or errors in its cells, addressing potential failures before the system goes into production or deployment. MBIST is typically implemented directly in the hardware of the memory subsystem, ensuring that the memory can test itself during manufacturing, in-system operation, and during periodic maintenance.[1]

2 Objective of the Work

To address the testing, repair, and diagnostic needs of embedded memories in System-on-Chip (SoC) designs. STAR memory system is versatile, supporting both repairable and non repairable memories across various foundries, process nodes, and memory IP vendors. It has

been silicon-proven in over a billion chips, demonstrating its reliability and effectiveness in a wide range of applications and technologies, including advanced nodes like FinFET. The main goal of the Synopsys Self-Test and Repair (STAR) Memory System™ is to improve the dependability, efficiency, and production yield of embedded memories within System-on Chip (SoC) designs. It delivers a unified and all-inclusive framework that tackles the critical aspects of memory testing, fault diagnosis, and repair. The STAR Memory System is specifically designed to meet the following key objectives:

a) Enhancing Yield: To increase the overall yield of semiconductor manufacturing by incorporating robust repair mechanisms that can identify and fix defective memory cells, thereby reducing the number of discarded chips.
b) Minimizing Time-to-Yield: To expedite the process of achieving high yield in volume production by providing advanced diagnostic tools that help identify and resolve manufacturing issues more efficiently.
c) Cost-Effectiveness: To lower the overall cost of testing and repair by integrating these capabilities directly into the memory, reducing the need for expensive external test equipment and minimizing production costs.
d) Supporting Advanced Technologies: To ensure compatibility with a wide range of process nodes, including advanced technologies like FinFET, and to support various types of embedded memories, such as SRAM, DRAM, and Flash.
e) Scalability and Flexibility: To offer a scalable solution that can be adapted to different SoC designs, from small, low-power devices to large, high-performance systems, and to be compatible with any foundry, process node, or memory IP vendor.
f) Ensuring Reliability: To provide a silicon-proven solution that has been validated in over a billion chips, ensuring its reliability and effectiveness in real-world applications.[2]

By achieving these objectives, the STAR Memory System aims to provide SoC designers with a powerful tool for improving the quality, yield, and efficiency of embedded memory testing and repair, ultimately leading to more reliable and cost-effective semiconductor products.

3. STAR (Self Test and Repair) Memory System

Embedded memory are the most compact elements of a system on a chip (SoC), making up to 90% of the chip's total area [3]. Due to their strict design criteria, embedded memory are more susceptible to manufacturing faults and field reliability issues compared to other cores on the chip. Therefore, the overall productivity of a System on a Chip (SoC) is greatly dependent on the yield of its memory. Ensuring a high memory yield is crucial to achieve a lower cost for the silicon. Frequently, newly identified systematic flaws are observed as faults that restrict the yield, caused by factors that are already recognized, such as the reduction in geometries. To identify the underlying reasons for the factors that limit yield, a thorough diagnosis and failure analysis is necessary. Based on this analysis, efforts can be taken to enhance the process. The Synopsys Design Ware The STAR Memory System (SMS) offers a comprehensive solutionthat enables users to create, seamlessly incorporate, and validate

embedded memory test and repair intellectual property (IP) within a System-on-Chip (SoC) design. The SMS Architecture, depicted in the following slide, offers a simplified perspective, beginning with the foundational component: memory. Memories within this framework encompass diverse types, including ROM/RAM variants or single and multiple port configurations. Each memory is encapsulated within a wrapper, as illustrated in the figure. This wrapper comprises two core elements: an Intelligent Wrapper and a Control Wrapper. The Intelligent Wrapper assumes responsibility for creating a tailored interface conducive to memory testing. Additionally, it integrates DFT control logic to augment the memory's controllability and observability during scan operations. Conversely, the Control Wrapper functions as the recipient of instructions or opcodes transmitted from the processor. While the depiction showcases a singular wrapper per memory, it is worth noting that in more advanced implementations, users may opt to share wrappers among identical memories. This strategic approach serves to mitigate the overall area requirement, thereby optimizing resource utilization within the architecture. An integral component within the architecture is the STAR Processor, occupying a pivotal role in orchestrating the system's functionality. Tasked with the critical functions of instruction decoding and executing appropriate actions via the wrappers, the processor serves as the central hub of control. Additionally, it manages the transition of clock frequencies, facilitating the seamless switch from the fast clock utilized during testing to the slower IEEE 1500 clock, essential for tasks such as shifting out status registers.

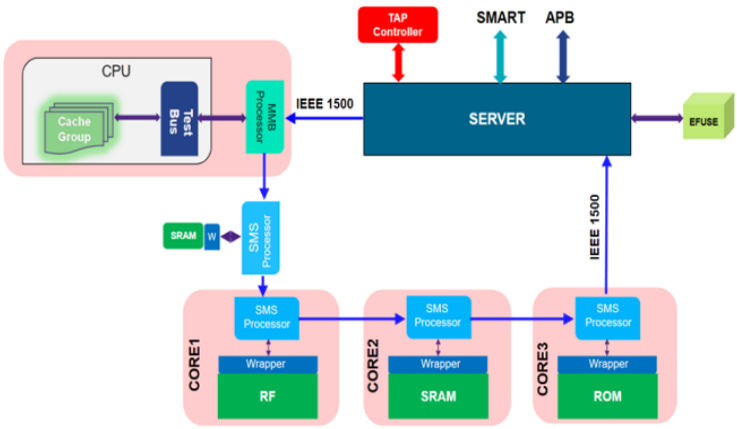

Fig. 1 STAR Memory System

The architecture accommodates various types of processors tailored to the specific characteristics of the targeted memories. For instance, the diagram illustrates a processor designed for ROM, SRAM, and RF memories.[4] Interface standardization is achieved primarily through adherence to the IEEE 1500 standard, ensuring compatibility and interoperability across diverse components, specialized categories of processors cater to unique architectural nuances, such as the Multi-memory Bus Processor. This variant is tailored for testing memories operating within a shared bus architecture, a configuration commonly encountered in processor cores. Such processors facilitate efficient testing and validation, optimizing the performance and reliability of the system.

At the heart of this architecture lies the SERVER, serving as the vital conduit connecting the system to the external environment. Functioning as the interface to facilitate testing procedures, the SERVER assumes a central role in ensuring the comprehensive evaluation of the System on-Chip (SoC).

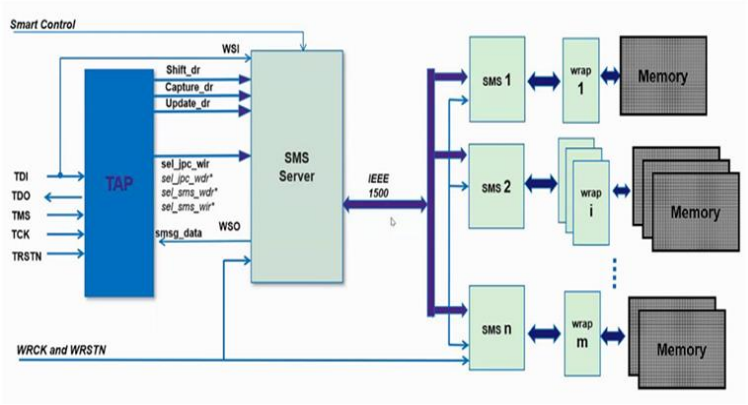

Fig. 2 SMS Architecture

Foremost among its interfaces is the IEEE 1500 compliant Test Access Port (TAP) pins, positioned at the SoC level. This standardized interface facilitates seamless communication with Automated Test Equipment (ATE), enabling robust testing protocols to be executed. In addition to the TAP pins, the SERVER interfaces with other communication protocols such as the Advanced Peripheral Bus (APB) and SMART. The APB, a widely adopted protocol, furnishes a standard framework for peripheral communication within the system. Leveraging an APB master integrated into the user design, operators can effectively oversee the operations of the SERVER. Moreover, the SMART interface augments the SERVER's capabilities by offering a low pin count solution tailored for predefined tasks, including Built-In Self-Test (BIST) and hard repair operations. This interface is particularly instrumental in facilitating in system testing, a critical requirement for Automotive System-on-Chips (SoCs). Within the context of in-system testing, a safety manager harnesses the SMART pins to trigger designated operations and monitor ensuing status signals. This proactive approach empowers stakeholders to promptly identify and address any anomalies or failures detected during the testing phase, thereby enhancing the reliability and safety of the overall system.

4 Components of SMS Subsystem

A. STAR Processor Architecture

The SMS Subsystem is designed as a comprehensive RTL assembly aimed at facilitating at speed testing, diagnostics, and repair of embedded memories. It incorporates Built-In Self Test (BIST), Built-In Self Diagnostic (BISD), and Built-In Redundancy Allocation (BIRA) functionalities. The architecture of the SMS Subsystem is hierarchical, comprising two primary components: Intelligent Wrapper(s) (IW) and the STAR Processor. Below is a block diagram illustrating the general structure of the SMS Subsystem at the top level.[5]

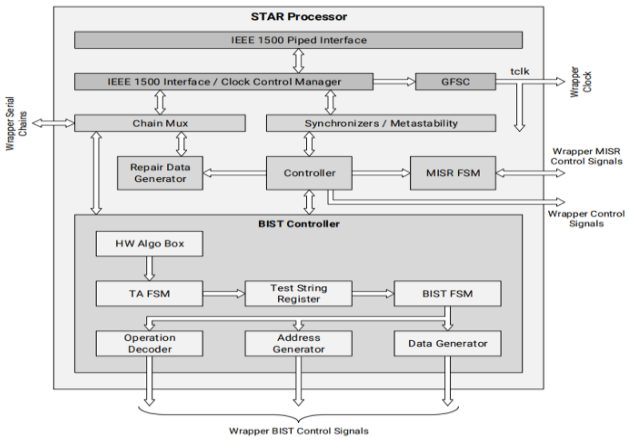

Fig. 3STAR Processor Structure

The STAR Processor serves as the central component of the SMS Subsystem, offering an external interface for RTL assembly compliant with IEEE 1500 standards. It manages the control of memories via Intelligent Wrapper(s) to execute testing and repair operations. Figure 3 illustrates the data flow andprovides a simplified depiction of the STAR Processor's structure.

a) IEEE 1500 Piped Interface: The STAR Memory System utilizes an IEEE 1500 serial pipelined interface, employing a two-stage pipelined design approach to enhance the timing performance of the interface. This technique allows for more efficient data processing and communication, reducing latency and increasing throughput. By breaking down the data transmission into two distinct stages, the system can better manage timing constraints and optimize the flow of information. This design not only improves the overall performance but also ensures greater reliability and accuracy in memory operations, making it a robust solution for complex embedded systems.

b) GFCS: The Glitch-Free Clock Switching (GFCS) mechanism, managed by the Clock/Reset management unit, facilitates seamless transitions between the SMS Subsystem's clock sources, WRCK and clk_sms. This mechanism ensures that clock switching occurs without introducing glitches, which can lead to errors or instability in system operations. By maintaining a stable clock signal during transitions, GFCS enhances the reliability and performance of the subsystem. This capability is particularly important in environments where precise timing is critical, as it prevents disruptions that could affect the integrity of memory testing and repair processes.

c) Synchronizers/Metastability: The STAR Processor is equipped with optional Reset Synchronizers and Metastability units designed to address synchronization needs related to the processor's reset signals. These components ensure smooth transitions and signal integrity as data moves between the slow (WRCK) and fast (clk_sms) clock domains within the STAR Processor. By effectively managing synchronization, these units help prevent issues such as signal degradation or timing errors, which can arise when signals traverse different clock domains. This capability is crucial for maintaining system stability and performance, especially in complex operations where precise timing is essential. Furthermore, the inclusion

of these units enhances the processor's adaptability, allowing it to efficiently handle varying operational conditions without compromising on reliability or accuracy.

d) Chain Multiplexer: The Chain Multiplexer is a unit responsible for multiplexing the chains within the STAR Memory System. This functionality allows for the efficient routing and management of multiple data streams, optimizing the use of system resources. By consolidating various chains into a single pathway, the Chain Multiplexer enhances the system's ability to handle complex operations and data flows. This not only improves overall performance but also simplifies the architecture, making it easier to manage and maintain. Additionally, the Chain Multiplexer contributes to the flexibility of the STAR Memory System, enabling it to adapt to different configurations and requirements with ease.

e) Controller: The Controller unit is responsible for generating control signals and gathering status bits from both the Intelligent Wrapper(s) and the STAR Processor. This unit plays a crucial role in coordinating the operations within the system, ensuring that each component functions harmoniously. By managing control signals, the Controller directs the flow of operations, enabling efficient communication and execution of tasks. Additionally, by collecting status bits, it provides valuable feedback on the system's performance and health, allowing for timely adjustments and optimizations. This capability is essential for maintaining the reliability and effectiveness of the overall system, ensuring that it operates smoothly and efficiently under various conditions.

f) MISR FSM: The MISR FSM is specifically designed for ROM memories, where it generates the necessary control signals for the Intelligent Wrapper to execute the MISRalgorithm. This functionality is crucial for ensuring accurate and efficient testing and diagnostics of ROM memory components. By providing tailored control signals, the MISR FSM facilitates the implementation of the MISR algorithm, which is essential for error detection and data integrity verification. This targeted approach enhances the reliability of ROM memory operations, allowing for precise monitoring and maintenance.

g) Test Algorithm FSM: The Test Algorithm FSM is designed to store and execute a hardwired default Test Algorithm, while also offering the option to include an externally accessible Test Algorithm Register for enhanced programmability. This dual capability allows the unit to perform standard testing procedures efficiently while providing the flexibility to adapt and customize algorithms based on specific requirements. By supporting both fixed and programmable test algorithms, the Test Algorithm FSM ensures comprehensive testing coverage and adaptability to evolving testing needs.

h) Test String Register: The Test String Register unit serves as a repository for a single March Element, acting as a buffer during the execution of BIST with either hardwired or programmable test algorithms. It facilitates communication between the Test Algorithm FSM, which operates at a slower clock speed, and the BIST FSM, which functions at a faster clock speed. This buffering capability ensures smooth and efficient data transfer between different clock domains, enhancing the overall performance of the testing process. Additionally, the Test String Register unit enables the execution of concise single March Element algorithms, such as initialization or rapid execution algorithms, which can be selected during the generation phase.

i) Hardwired Algorithms Container: The Hardwired Algorithms Container is responsible for storing test algorithms that are embedded within the STAR Processor's RTL. This unit ensures that essential testing procedures are readily available and can be executed efficiently without the need for external programming. By housing these algorithms directly within the processor, the Hardwired Algorithms Container facilitates quick and reliable access to predefined testing protocols, streamlining the testing process.

j) BIST FSM: The BIST FSM is responsible for executing a single March Element, which can be stored in the Test String Register, if available, or hardwired within the TA FSM unit. It generates the necessary control signals, along with address and data sequencing, to facilitate operations within the Intelligent Wrapper(s). This unit is equipped with a Stop-on-Nth-Error diagnostic feature, allowing for precise error detection and analysis during testing. Additionally, it implements an Address Looping (Hammering) capability, which is useful for stress testing and ensuring the robustness of memory components.[5]

B. Wrapper Architecture

The Intelligent Wrapper (IW) is a key component of the SMS Subsystem, serving as the interface between the STAR Processor and the memory units. This wrapper facilitates communication and coordination, ensuring that the processor can effectively manage and interact with the memory components.[7] By providing a structured interface, the Intelligent Wrapper enables seamless execution of testing, diagnostics, and repair operations, enhancing the overall functionality of the system. Additionally, the IW plays a crucial role in optimizing data flow and control signals, allowing for efficient memory management and operation. Its integration into the SMS Subsystem underscores its importance in maintaining system reliability and performance, making it an indispensable element in complex embedded memory systems.[6]

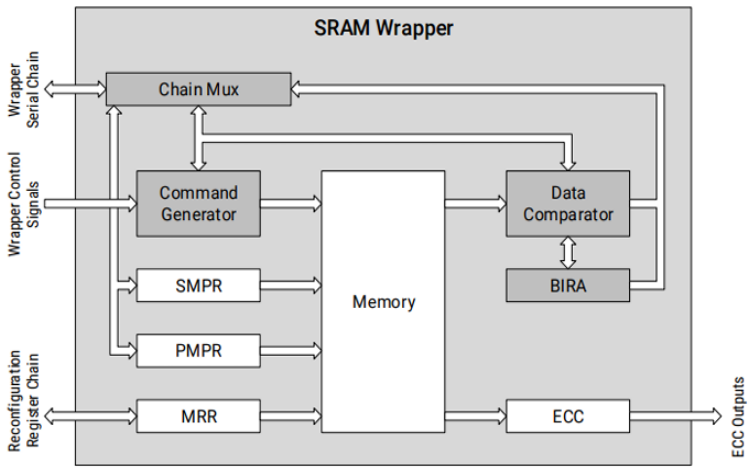

Fig. 4STAR Processor Structure

Figure 4 depicts the simplified structure of the Intelligent Wrapper tailored for SRAM memories. This design facilitates efficient interaction with the STAR Processor, optimizing memory operations such as testing, diagnostics, and repair. The wrapper ensures smooth data flow and control signal management, enhancing the performance and reliability of SRAM units. Its scalable architecture supports the integration of multiple memory units, making it a following component in embedded memory systems:[7]

a) Command Generator: The Command Generator module creates control signals based on the March Operation code command data from the STAR Processor. It translates operation codes into actionable signals, ensuring efficient and precise system operations during memory testing and diagnostics. This module enhances system flexibility by adapting to various operation codes, allowing for customized testing procedures and maintaining overall reliability and performance.

b) Chain Multiplexer: The Chain Multiplexer is responsible for multiplexing the scan chains of the Intelligent Wrapper between the wr_si and wr_so pins. This process is controlled by the sel_chain input, which determines the selection of the chain to be routed. By efficiently managing the multiplexing of scan chains, the Chain Multiplexer optimizes data flow and enhances the flexibility of the system. This capability is crucial for adapting to different testing configurations and requirements, ensuring that the system can handle complex operations with ease.

c) Data Comparator: The Data Comparator functions either as an integrated component within the memory or as part of the SMS hierarchy, tasked with detecting failures in memory operations. It analyzes the memory error output, known as ErrDataOut, to produce a one-bit error signal indicating the presence of an error. When operating within the SMS hierarchy, the Data Comparator compares the expected data with the actual output from the memory, generating a one-bit error signal if discrepancies are found. This mechanism is essential for ensuring data integrity and reliability, allowing for prompt identification and correction of errors. By providing accurate error detection, the Data Comparator enhances the robustness of memory systems, contributing to overall and performance.

d) BIRA Engine: The BIRA Engine is designed to analyse information regarding memory faults, including faulty addresses and bit positions, to determine the optimal strategy for allocating available redundancy. It generates a corresponding repair signature that facilitates the correction of identified errors. This engine plays a critical role in enhancing the reliability and longevity of memory systems by efficiently managing redundancy resources. By selecting the best approach for redundancy allocation, the BIRA Engine ensures that memory faults are addressed effectively, minimizing the impact on system performance. Additionally, its ability to generate repair signatures contributes to streamlined maintenance processes, allowing for quick and accurate fault resolution.

e) Memory Reconfiguration Register (MRR): The External Memory Reconfiguration Register (MRR) is utilized for ASAP memories, which lack redundant elements. This register stores the memory reconfiguration signature and becomes accessible when the Wrapper External Repair option is activated (= 1). The MRR plays a crucial role in managing memory configurations, allowing for adjustments and optimizations even in systems without built-in

redundancy. By storing reconfiguration signatures, it provides a mechanism for implementing necessary changes to enhance memory performance and reliability. This capability is particularly valuable in scenarios where external repair solutions are required, offering a flexible approach to maintaining system integrity.

f) Scannable/Programmable Memory Parameters Registers (SMPR/PMPR): The Scannable/Programmable Memory Parameters Registers (SMPR/PMPR) serve as auxiliary registers that offer an alternative serial access to memory input/output ports, including read margin control signals and TEST1, among others. These registers are essential for fine tuning memory operations, allowing for precise adjustments to various parameters that influence performance and reliability. By providing serial access, SMPR/PMPR enable efficient configuration and monitoring of memory settings, facilitating enhanced control over memory behavior. This capability is particularly useful in optimizing system performance, ensuring that memory components operate within desired specifications.[6]

5 SMS Verification

I. Integrity Checker

Integrity checking is important when STAR Memory System is partitioned into modules (STP and memory wrappers) prior to implementation in customer's design hierarchy. SMS top-level module ($proc_name_sms module) that establishes interconnections between SMS modules is not used in customer's design in this case. To ensure the proper functioning of SMS, user must establish interconnections between the SMS module instances in customer's design exactly in the same way they are established within the $proc_name_sms module. The SMS Integrity Checker allows User to verify interconnections between SMS modules in customer's design against the reference interconnections established within SMS top-level $proc_name_sms module.[7]

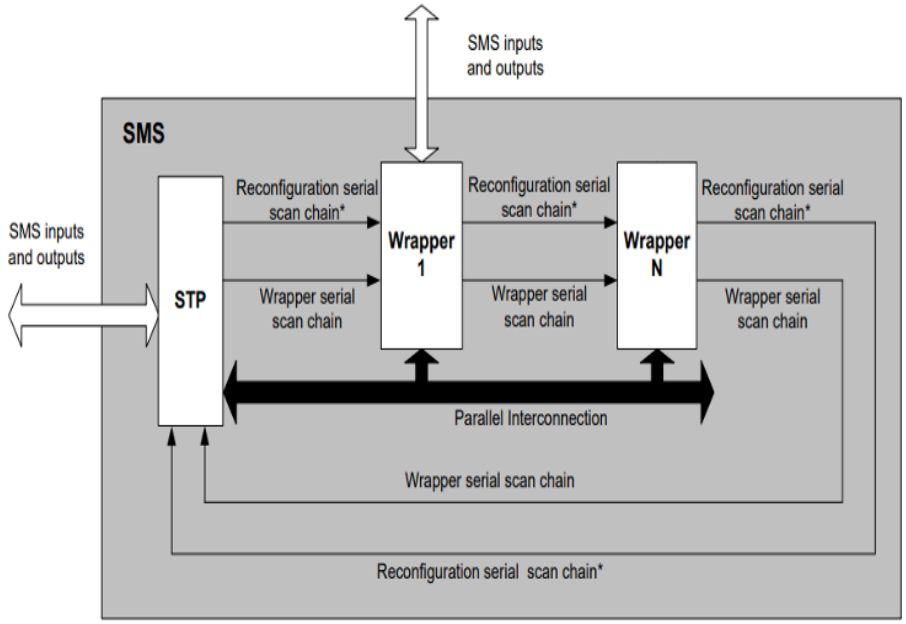

Fig. 5 Integrity Checker Block Diagram

The SMS Integrity Checker contains a set of specific test modules for customer net verification. The Integrity Checker modules port names are identical to the corresponding functional modules port names. The Integrity Checker modules generate predetermined stimulus and examine the corresponding responses for each interconnection in the customer net, thus having integrity of SMS module instances be checked.[7] For the verification of interconnections between SMS modules in customer's design User must simulate the Integrity Checker modules and the customer's module that establishes the customer net and contains the entire hierarchy of all SMS module instances (similar to SMS top-level module $proc_name_sms).

II. Testbench Verification

STAR Processor level components – STAR Processor and Wrapper provide a set of scripts and testbenches to verify the functionality (component/level testbench) and check RTL code against internally recommended coding guideline (code checker script). STAR Memory System hierarchical architecture implies on STAR Processor level verification flow that also has a hierarchical structure. The generic verification flow at STAR Processor level is illustrated in the figure below.[7]

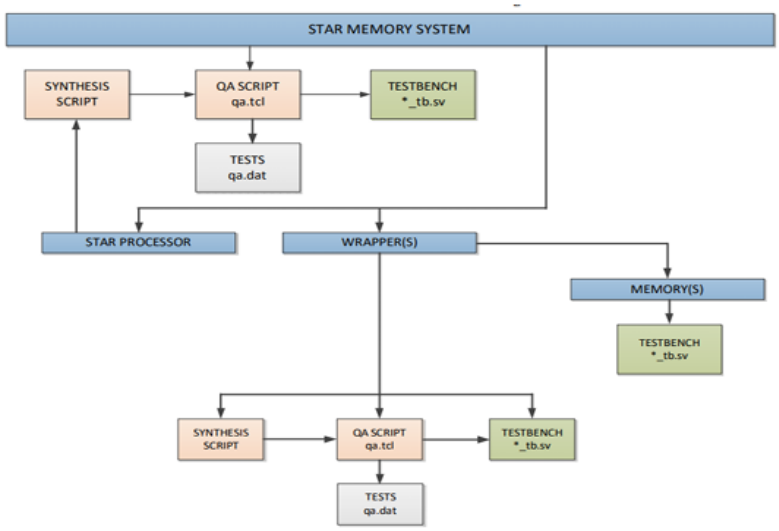

Fig. 6 STAR Processor Level

As it is shown on figure above, to state that STAR Processor level verification is passed it is required thateach component/level testbench is executed with recommended set of arguments (tests) and none of these tests is failed both at RTL and gate levels, code checker script is executed on each component RTL files and it does not violate any coding guideline within the defined set of rules.

STAR Memory System is provided with dedicated testbenches for RTL and gate level simulation and verification of STAR Memory level components functionality as a complete assembly. The STAR Memory System testbench is designed to exercise the STAR Memory

System in different functional modes as the sequences of STAR Processor instructions and IEEE 1500 interface operations.

6 Results

Successfully completed the SMS verification of various IPs, which included both integrity test verification and full mode verification, encompassing a comprehensive range of checks. This process involved a deep understanding of the underlying checks, as well as the memory and wrapper configurations. Utilizing Synopsys Verdi, I effectively tracked and analysed the signals entering and exiting the memory wrappers, ensuring thorough verification and robust system integrity.

```
15/04/25  18:35:10:     Simulating +chk_bist +fsim_algo +verify_algo
15/04/25  18:35:11:     Simulating +chk_bist +treg_algo +verify_algo
15/04/25  18:35:12:     Simulating +chk_bist +fsim_algo +loop_string +verify_algo
15/04/25  18:35:13:     Simulating +chk_bist +fsim_algo +loop_action +verify_algo
15/04/25  18:35:14:     Simulating +chk_bist +fsim_algo +loop_nested +verify_algo
15/04/25  18:35:15:     Simulating +chk_bist +loop_string +infinite
15/04/25  18:35:17:     Simulating +chk_bist +loop_action +infinite
15/04/25  18:35:18:     Simulating +chk_bist +loop_nested +infinite
15/04/25  18:35:19:     Simulating +chk_opmask +verify_algo
15/04/25  18:35:21:     Simulating +chk_bist +loop_string +inv_pattern
15/04/25  18:35:22:     Simulating +chk_bisd +algo_sel=0 +verify_algo
15/04/25  18:35:23:     Simulating +chk_bisd +auto_incr +algo_sel=0 +verify_algo
15/04/25  18:35:24:     Simulating +chk_bisd +fbist_run
15/04/25  18:35:25:     Simulating +chk_bisd +fsim_algo +verify_algo
15/04/25  18:35:26:     Simulating +chk_bisd +auto_incr +fsim_algo +verify_algo
15/04/25  18:35:28:     Simulating +red_set +ext_algo
15/04/25  18:35:29:     Simulating +soft_repair +ext_algo
15/04/25  18:35:30:     Simulating +get_repair_data +ext_algo
15/04/25  18:35:31:     Simulating +ext_repair +ext_algo
15/04/25  18:35:32:     Simulating +chk_mrr2inbr +ext_algo
15/04/25  18:35:33:     Compiling +define+MASIS_ERR_INJECT
15/04/25  18:35:36:     Simulating +chk bist +algo sel=0 +verify algo
```

Figure 7 Compilation Run for SMS

The image is a log of a simulation process, detailing various testing and diagnostic operations in SMS verification. The log entries are timestamped, indicating the sequence and timing of each operation. The repeated mention of chk_bist suggests that the system is undergoing Built In Self-Test (BIST) procedures, a technique that allows a system to test itself for faults, ensuring it can detect and diagnose issues autonomously. In SMS verification, BIST is used to verify the integrity and functionality of the system components responsible for processing and verifying Memories, ensuring they operate correctly and reliably. Additionally, the entries with chk_bisd indicate diagnostic checks being performed on the system, which are crucial for identifying and analyzing faults or errors within a system. In SMS verification systems, these checks could ensure that the system can accurately diagnose issues related to memory or data processing, critical for maintaining the integrity and security of SMS verification processes. The mention of soft_repair suggests that the system includes mechanisms for repairing or correcting errors in memory or other components. Soft repair techniques are used to address faults without requiring hardware replacement, often by reconfiguring or adjusting system parameters.

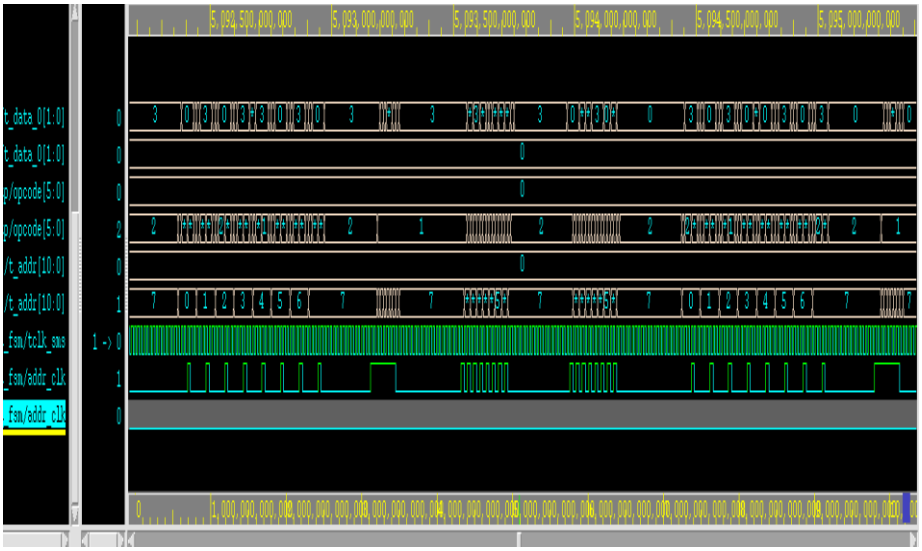

Figure 8 SMS Vedi Waveform

The image depicts a waveform diagram, in context of SMS verification, this waveform illustrate the functioning of a digital system tasked with processing or verifying SMS. On the left side of the image, signal names like t_data_0[1:0], t_data_1[1:0], opcode[5:0], and t_addr[10:0] are listed, representing various components of the system such as data buses, operation codes, addresses, and clock signals. Each signal is accompanied by a waveform that displays its value changes over time, with t_data_0[1:0] and t_data_1[1:0] indicating data being processed, and opcode[5:0] denoting different operations executed by the system. Clock signals like fsm/tclk_sms and fsm/addr_clk are crucial for synchronizing operations to execute the march algorithms within SMS, dictating when data is sampled and processed. The horizontal axis represents time, marked at specific intervals to help understand the sequence and duration of operations. The presence of signals such as fsm/tclk_sms suggests the use of a finite state machine (FSM) to manage the SMS verification process, as FSMs are often employed to control operation sequences based on inputs and current states. Overall, this waveform diagram likely illustrates the internal workings of a digital system involved in SMS verification, showing signal changes over time to help engineers ensure the system's correct and efficient functioning.[8]

7 Conclusion & Future Scope

In conclusion, the Star Memory System represents a highly efficient and scalable approach to managing memory modules within integrated circuits, leveraging a centralized control architecture that ensures robust communication and coordination. When combined with Built In Self-Test (BIST) capabilities, this system significantly enhances the reliability and fault tolerance of memory components by enabling autonomous, continuous, or periodic testing. This integration not only reduces the dependency on external testing equipment but also provides comprehensive fault coverage and simplifies maintenance processes.Driven by the increasing complexity and demands of modern electronic devices and systems. As technology continues to advance, the need for more reliable, efficient, and scalable memory solutions

will grow, positioning the Star Memory System as a critical component in various high-tech applications. In the realm of artificial intelligence and machine learning, the demand for high-performance memory systems is paramount, and the Star Memory System can provide the necessary reliability and fault tolerance required for these data-intensive applications. Additionally, with the proliferation of the Internet of Things (IoT), there will be a surge in the number of connected devices, each requiring robust memory solutions that can be efficiently managed and tested, making the Star Memory System an ideal choice.[9]

References

1. Y. -C. Tsai, W. -C. Ting, C. -C. Wang, C. -C. Chang and R. -S. Liu, "Built-in Self-Test and Built-in Self-Repair Strategies Without Golden Signature for Computing in Memory," 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE), Antwerp, Belgium, 2023.
2. H. Jahanirad, “A Low-cost BIST Design Supporting Offline and Online Tests,” Journal of Electronic Testing: Theory and Applications (JETTA), vol. 38, no. 1. pp. 107–123, 2022.
3. S. Shrivastava, S. Malviya, and N. Gupta, “Built in Self Test Architecture Using Logic Module,” Int. J. VLSI Des. Commun. Syst., vol. 8, no. 4, pp. 25–34, 2017.
4. Z. Peng, “Research on Low Power BIST Based on LFSR Reseeding,” Acad. J. Sci. Technol., vol. 3, no. 2, pp. 203–208, 2022.
5. A. Menbari and H. Jahanirad, “A Tunable Concurrent BIST Design Based on Reconfigurable LFSR,” J. Electron. Test. Theory Appl., vol. 39, no. 2, pp. 245–262, 2023.
6. M. H. and W. Waller, “New Test Pattern Generators for the BIST Pseudo-Exhaustive Testing based on Coding Theory Principles,” Commun. Appl. Electron., vol. 4, no. 8, pp. 29 44, 2016.
7. K. Darbinyan, G. Harutyunyan, S. Shoukourian, V. Vardanian and Y. Zorian, "A Robust Solution for Embedded Memory Test and Repair," *2011 Asian Test Symposium*, New Delhi, India, 2011, pp. 461-462.
8. Synopsys Inc. (2023). ”Verdi Advanced Debug Platform: Enhancing Verification Efficiency and Accuracy.” Synopsys Technical Documentation, 12(4), 45-67.
9. M. Kumar, “An Efficient Fault Detection of FPGA and Memory Using Built-in Self-Test [BIST],” Am. J. Electr. Comput. Eng., vol. 3, no. 1, p. 38, 2019.