

Design & Verification of AI Enabled Reconfigurable SRAM Controller for Automobile Application

Shaila C K¹, I.Thusnavis Bella Mary², Divya P.S.³, Vinodhini A⁴, G Manoj⁵, G Naveen Sundar^{6*}

¹Department of Electronics and communication, Karunya Institute of Technology and Science (KITS), Coimbatore, India

²Department of Electronics and Communication, Karunya Institute of Technology and Science (KITS), Coimbatore, India

³Department of Electronics and Communication, Karunya Institute of Technology and Science (KITS), Coimbatore, India

⁴Department of Electronics and communication, Karunya Institute of Technology and Science (KITS), Coimbatore, India

⁵Department of Electronics and communication, Karunya Institute of Technology and Science (KITS), Coimbatore, India

manojpillai@karunya.edu

^{6*}Department of Computer Science and Engineering, Karunya Institute of Technology and Science (KITS), Coimbatore, India

Corresponding Author: naveensundar@karunya.edu

ARTICLE INFO

ABSTRACT

Received: 30 Dec 2024

Revised: 05 Feb 2025

Accepted: 25 Feb 2025

The rapid advancements in artificial intelligence (AI) and automotive technology have necessitated the development of sophisticated and intelligent memory management systems. This research focuses on the design and validation of an AI-enabled reconfigurable Static Random-Access Memory (SRAM) controller tailored for automotive applications. By leveraging AI-based algorithms, the proposed controller enhances the efficiency of memory allocation, reduces access time, and optimizes power consumption, thereby improving the overall performance and reliability of automotive electronic systems. The controller's reconfigurable architecture allows for dynamic adaptation to varying workloads, ensuring efficient data handling in real-time applications such as in-vehicle entertainment systems and Advanced Driver Assistance Systems (ADAS).

To ensure operational accuracy and optimal performance, the design is coded in Hardware Description Language (HDL) and verified through simulation and FPGA-based prototyping. The verification process employs industry-standard methods, such as Universal Verification Methodology (UVM), to assess the SRAM controller's reliability and resilience. Test results indicate significant improvements in speed, power efficiency, and flexibility compared to conventional SRAM controllers, positioning it as a suitable candidate for automotive systems of the future

Keywords: Reconfigurable memory, automotive electronics, hardware verification, FPGA prototype, AI-enabled SRAM controller, and UVM

I. INTRODUCTION

Intelligent and effective memory management solutions are becoming more and more necessary as artificial intelligence (AI) is incorporated into automobile systems. High-speed, low-power, and dependable memory controllers are necessary for automotive applications including Advanced Driver Assistance Systems (ADAS), autonomous driving, and in-car infotainment in order to manage intricate data processing duties in real time. Despite their effectiveness, traditional SRAM controllers are not flexible enough to handle dynamic workloads in contemporary automobile applications.

The design and validation of an AI-enabled reconfigurable SRAM controller especially suited for automotive applications are presented in this research. The suggested controller minimizes power consumption, lowers latency, and improves memory access efficiency by utilizing AI-based optimization techniques. The system's total performance is enhanced by its reconfigurable architecture, which allows for dynamic adaptability to changing computational needs.

To guarantee functional accuracy and dependability, the design is implemented using Hardware Description Language (HDL) and validated using industry-standard techniques such as FPGA-based prototyping and Universal

Verification Methodology (UVM). The goal of this project is to develop a memory management system for next-generation automobile electronic systems that is intelligent, effective, and reliable.

II. STATIC RANDOM ACCESS MEMORY

Static RAM, commonly referred to as SRAM, is a type of random access memory that retains data bits as long as power is supplied. SRAM offers higher performance and consumes less power compared to dynamic RAM (DRAM) because it does not need constant refreshing. Two key control lines in SRAM are the write enable (WE) and output enable (OE) lines, which are used for reading and writing data. Additionally, there is a chip select line, which indicates which RAM chip is active when multiple chips are present on a board. This line is essential for directing data to the appropriate RAM chip. Finally, there is a line connected to the microcontroller's system clock, which helps synchronize operations within the SRAM.

This design allows the chip and controller to synchronize effectively, ensuring that all data is recorded accurately without any values omitted in between. Additionally, it helps the SRAM determine the precise moment when eight bits have been input, making them ready for reading and storage in memory. The controller routes several address and data lines to the memory area, utilizing eight data lines and fifteen address lines for data transmission. The AXI protocol defines five separate channels: Write Address Channel, Write Data Channel, Write Response Channel, Read Address Channel, and Read Data Channel. Through the Write and Read Address Channels, the AXI4 master can request access from the AXI4 SRAM slave and exchange transaction data. The AXI4 SRAM is composed of three main functional blocks: the Slave Interface Logic, Main Control Logic, and SRAM Interface Control Logic. A simple block diagram of the AXI4 SRAM microcontroller architecture is illustrated in Figure 1. The AXI4 SRAM slave interface and the attached AXI4 master work together to exchange data. The master requests access to the fabric memory by sending write or read requests through the appropriate Write Address or Read Address Channel.

Functional blocks of AXI4SRAM:

- Slave Interface Logic
- Main Control Logic
- SRAM Interface Control Logic

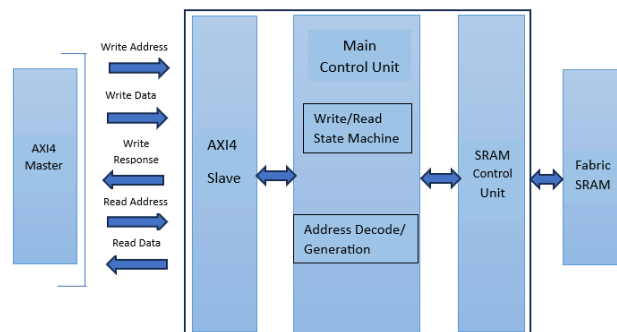


Fig. 1. General Block Diagram of SRAM Controller with AXI Interface

Fig. 2 displays the SRAM controller's flow chart. It suggests how SRAM functions. It functions in tandem with the clock signal. In order to send data, it waits for the reset and ready signal to be active. The address is fetched, and the data is subsequently read or written into the address to complete the read and write process.

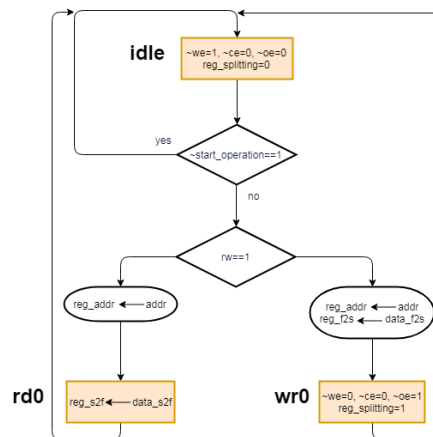


Fig. 2. Flowchart of SRAM Controller

Table 1 represents the parameters that can be varied in SRAM controller like data width and address width.

TABLE I. RECONFIGURABLE SRAM PARAMETERS

Parameters	Size
Memory Data Width	upto 1024 bits
Memory Address Width	upto 1024 bits
Read and Write Data FIFO Depth	8,16
AXI Data Width	upto 1024 bits
AXI Address Width	upto 1024 bits

III. AMBA AXI PROTOCOL

Rather than defining the interface of an interconnect, the interface of IP blocks is defined by AXI. AXI interface types in AXI3 and AXI4 are slave and master. There is symmetry in these interface types. There are master interfaces and slave interfaces connected by all AXI connections. Since the signals in AXI interconnect interfaces are the same, integrating various IP is not too difficult. The master and slave interfaces are connected by AXI connections, as seen in the previous diagram. Without additional logic, the direct connection provides the greatest bandwidth between the slave and master components. Additionally, there is just one protocol to validate when using AXI.[6] Important characteristics that are intended to reduce transaction delay and increase bandwidth:

- Independent read and write channels

One set of channels is supported by AXI for writing operations, and the other for read operations. The interfaces bandwidth performances are enhanced by having two separate sets of channels. This is because of simultaneous nature of read and write operations.

- Multiple outstanding addresses

Multiple outstanding addresses are supported by AXI. As a result, a master can start new transactions without having to wait for older ones to finish. The capability to process transactions in parallel is one way that this can enhance system performance.

- No connection in timing between data operations and addresses.

The address and data operations do not have a strict time relationship when using AXI. This implies that a master may, for instance, provide an address for the write operation on the Write Address channel; but, the master is not constrained in terms of time when it must supply the associated data in order to write on the Write Data channel.

- Unaligned data transfers

The first bytes accessed in any burst that consists of data transformation larger than eight bit may not line up with the boundary within natural address. A 32-bit packets of data beginning at 0x1002 which is the byte address, for instance, is not automatically set to the border of the 32-bit address.

- Transaction completion during Out-of-order

In AXI, transactions can be completed out of order. Transaction identifiers are part of the AXI protocol, thus completing transactions with varied ID values is not restricted. This means that several logical ports, each handling a transaction in order, can function as a physical port which is single to accommodate out-of-order transactions.

- Burst transactions based on start address

The starting address is only provided by AXI masters for the initial transfer. The slave will use the burst type to determine the next transfer address for any subsequent transfers. Two interfaces—a slave and a master—is described in the AXI standard which is an end-to-end protocol.

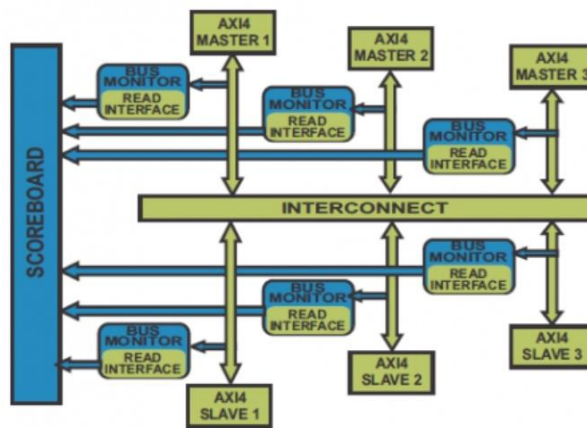


Fig. 3. Block Diagram of AMBA AXI4 IP Verification

Block diagram of AMBA AXI IP verification is shown in Fig 3 which contains all verification components such as master, slave, monitor and scoreboard. It can create multiple instances of verification IP (VIP) components as per the verification needs. VIP can also be used in a multi environment scenario where a user can enable or disable various components.[3] More features are added to AXI4 compared to AXI3. Two features from AXI3 is removed. The comparison between AXI3 and AXI4 is shown in Table 2.

TABLE II. COMPARISON BETWEEN AXI3 AND AXI4

Parameters	AXI3	AXI4
Burst Length	16 Beats	256 Beats
Write Interleaving	Yes	No
Locked Transfer	Yes	No
QoS	No	Yes
User Signals	No	Yes
Regions	No	Yes
Cache	No	Yes

IV. AI ACCELERATOR

Artificial intelligence (AI) accelerators are specialized hardware parts designed to maximize and accelerate machine learning (ML) and AI workloads, as opposed to general-purpose CPUs. Applications for them are numerous and include robotics, edge computing, driverless cars, high-performance computing, and healthcare. AI accelerators perform better on AI jobs because they are designed for deep learning algorithms, matrix multiplications, and parallel calculations.

Accelerators come in a variety of forms, such as FPGA-based accelerators, which are hardware devices that may be reconfigured to do certain AI tasks. delivers good performance and minimal latency, but requires particular programming. The other kind is for ASIC chips, which are designed especially for AI applications and provide the best performance and lowest power consumption.

Key Features:

High Parallelism: Capable of handling thousands of calculations simultaneously.

Fit for Matrix Functions: efficient handling of AI workloads, including convolutions, tensor multiplications, and deep learning inference.

Energy Efficiency: Compared to traditional CPUs, AI tasks use less energy. It is perfect for real-time AI applications such as edge AI, robotics, and autonomous driving because of its high throughput and low latency.

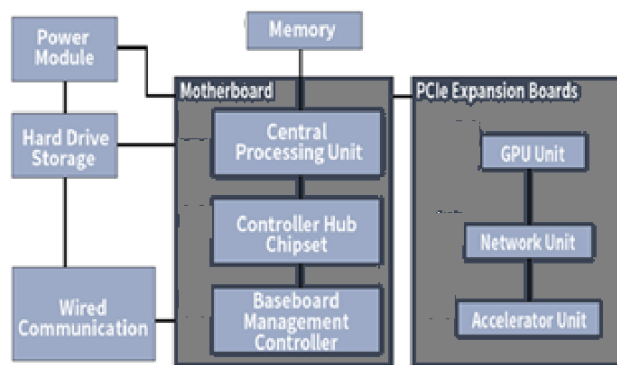


Fig. 4. Block Diagram of AI Accelerator

Key Elements in the Block Diagram of an AI Accelerator:

CPU (Central Processing Unit): This is the brain of the server, handling calculations and general tasks, executing commands, and overseeing the overall operation of the AI system;

GPU (Graphics Processing Unit): Used for parallel processing, GPUs are crucial for AI tasks involving large datasets, such as machine learning and deep learning, as they speed up the computational workload by handling multiple tasks at once;

Memory: AI servers need high-bandwidth memory to handle large volumes of data, which is usually composed of both specialized memory units for data storage during training and inference processes and DRAM (Dynamic Random-Access Memory) for quick access.

Storage: Large datasets used to train AI models are stored on AI servers using high-capacity storage (such as SSDs or NVMe drives). This is important because AI systems frequently need large amounts of data for testing and training.

PSU (Power Supply Unit): A reliable and efficient power supply is essential to preserving server stability because AI workloads can be extremely demanding, particularly in multi-GPU configurations.

Components of networking: AI servers need high-speed networking components (such as Ethernet and InfiniBand) in order to connect to other servers and devices for effective distributed processing and sharing of big datasets.

V. METHODOLOGY

The LUTs serve as the fundamental logic implementation unit in every FPGA device. There are several different FPGA families on the market today. Every FPGA device has a unique internal layout; some even have embedded clock control tiles, massive DSP slices for fast multiplication, RAM or ROM blocks, and/or hardcore or softcore CPU architecture. All time, the designer should be fully knowledgeable on the internal operations of the chosen FPGA device(s). This ensures the digital system could be customized into any desired FPGA device; the logic must be developed or designed in such a way that it will exploit fully the possible utilization of the internal resources of the FPGA Fabrics. This includes clearing the DRC for the placement and routing of all internal cells. The designer must ensure that the logic satisfies all nonfunctional needs while retaining logical soundness to write elaborate circuit descriptions. Today, the designer optimizes a logic description by using the tools available. The designer first has to provide the logical function in the simplest possible terms. Designer applies optimisation techniques that alter the logical description to meet other objectives which do not oppose the original functions.

It is obvious that they must select an FPGA in order to transfer their design onto it. It takes more than just familiarity with the FPGA brand and family to convert logic into a configuration. varied amounts of I/O pins and varied quantities of logic components are accessible in different FPGA families, which also affects how much interconnectivity there is on the chip. Logical Elements (LEs) counts, also known as LUTs, and the number of pins on the device are typically used to determine the size of an FPGA. The LEs on the chip in relation to the pin count most of the time. To obtain the required performance, it binds the multiple logical gates during the routing process. It is primarily utilized when there is an extreme need for timing.

VI. SIMULATION RESULTS

The System Verilog Code written for SRAM AXI interface protocol is simulated and synthesized in Xilinx Vivado with the PYNQ Z2 board using ZYNQ XC7Z020-1CLG400C. The read and write operation which is obtained by simulation is shown in Fig 5 and its performance matrix is in Table 3. The simulated output shows about how read and write operation takes place with memory data and AXI data channel, also similarly with memory address and AXI address channel.

TABLE III. PERFORMANCE MATRIX OF SRAM

		Throughput		Latency
Single Data Transfer	Write	16 beats/16 cycles	100%	
	Read	16 beats/19 cycles	84%	3 cycles/burst
AXI Transfer	Write	16 beats/16 cycles	100%	
	Read	16 beats/17 cycles	94%	1 cycles/burst

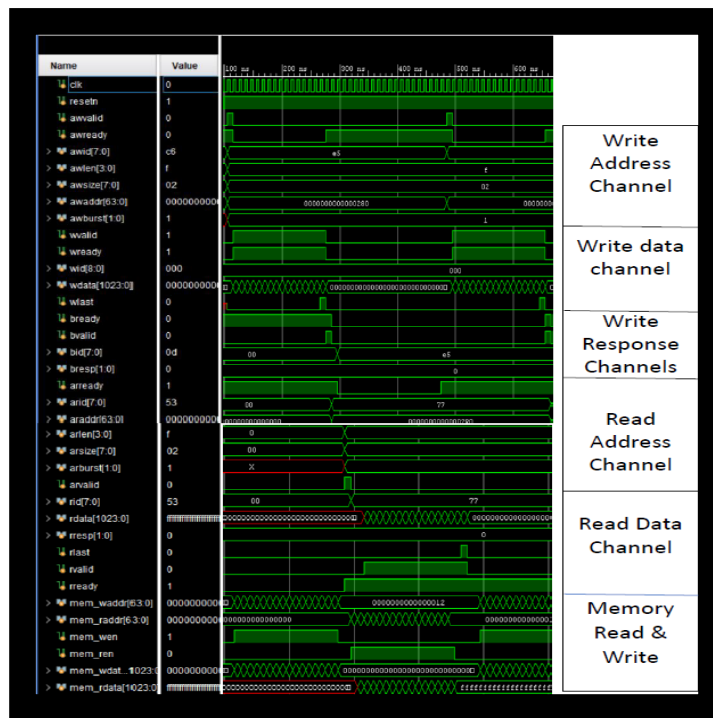


Fig. 5. Read and Write operation in SRAM using AXI

Experimental platform and environment

The proposed SRAM was implemented using PYNQ Z2 Board. The whole processor connection with SRAM protocol is shown in Fig. 6.

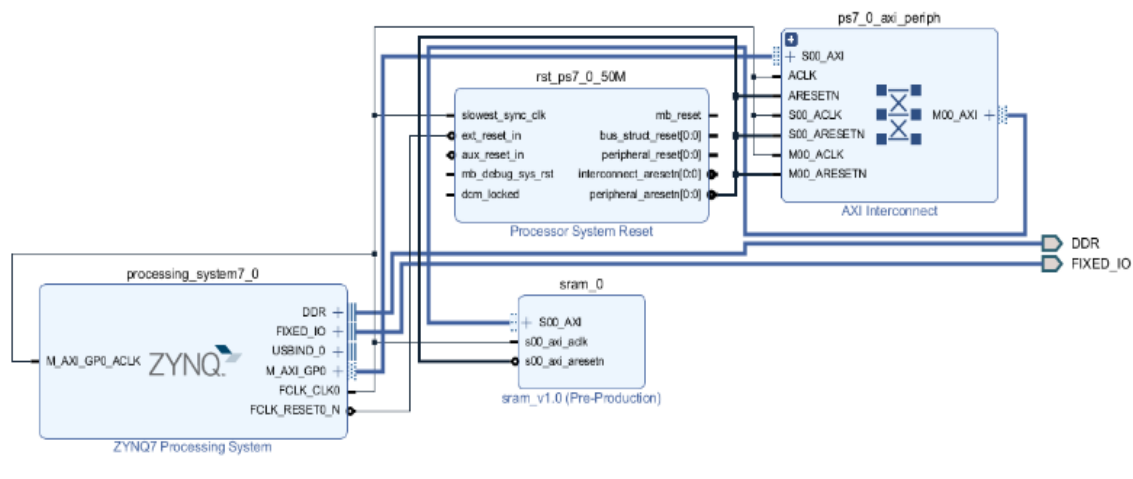


Fig. 6. SRAM RTL simulation with ZYNQ XC7Z020-1CLG400C

Schematic of SRAM

The schematic of SRAM AXI is in Fig 7. It represents the various components used and control, read and write signals used and its connection.

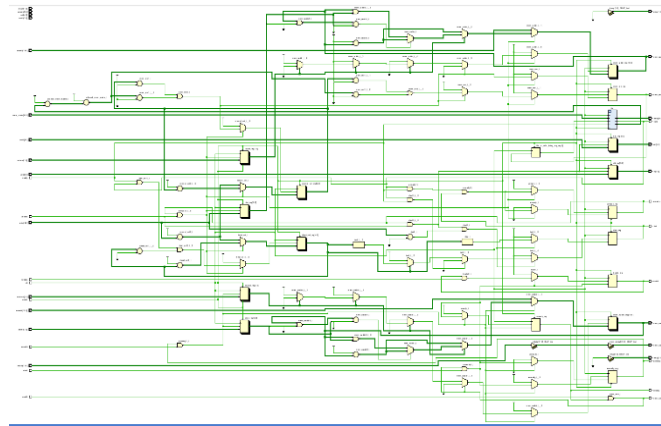


Fig. 7. Schematic of SRAM

Utilization Report of SRAM

The utilization report lists the clock, DSP slices, and BRAM resources, among other logic components and resources used in the design. It also pinpoints bottlenecks and gives crucial time information. This report helps with resource management and optimization for FPGA design. It is examined how the design is used at the hierarchical, user-defined P block, or SLR levels using the Utilization Report shown in Table 4.

TABLE IV. UTILIZATION REPORT OF SRAM

Resource	Utilization	Available	Utilization %
Slice LUTs	395	53200	<1%
Slice Registers	603	106400	<1%
Slice LUT as Logic	165	13300	1.2%
LUT as Memory	335	53200	<1%
LUT as Memory	60	17400	<1%
BUFGCTRL	1	32	3.1%

Power Analysis of SRAM

Power analysis is an essential procedure that aids in the assessment of Field Programmable Gate Arrays (FPGAs') power consumption at different phases, such as post-synthesis, implementation, and RTL-level estimation. Clock gating and power-aware placement are two tactics that designers can use to improve their designs for reduced power consumption. This research offers insightful information on static, dynamic, and overall power utilization. Fig 8. shows the static and dynamic power utilization SRAM using AXI. These values represent the power consumption of the device under specific conditions and can provide insight into its efficiency.

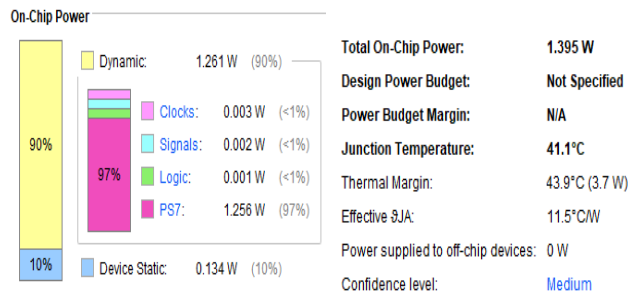


Fig. 8. Power Utilization and Estimation of SRAM

Comparison of Power/Area of SRAM and BRAM

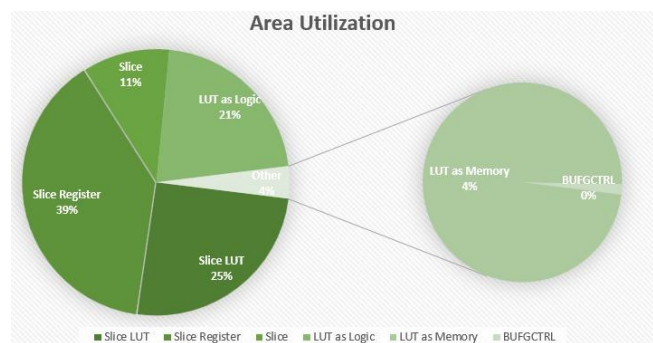


Fig. 9. Area Utilization of BRAM

The above Fig 9, shows the area utilization of BRAM. When we compare the area of SRAM and BRAM, it is seen that BRAM occupies more area for memory 4% where as SRAM occupies very less space <1%.

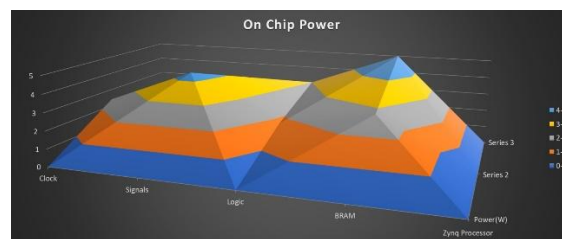


Fig. 10. Comparison of Power for SRAM and BRAM

Comparison of Power for SRAM and BRAM is shown in Fig 10. It is seen than SRAM consumes less power compared to BRAM.

Implementation Results

The SRAM protocol is implemented in PYNQ Z2 board which has ZYNQ XC7Z020-1CLG400C processor. The board is powered up using USB JTAG connector and the data is transmitted through the memory. The serial communication which takes place through the communication port is viewed through Vivado Vitis. The implementation is shown in Fig.11.

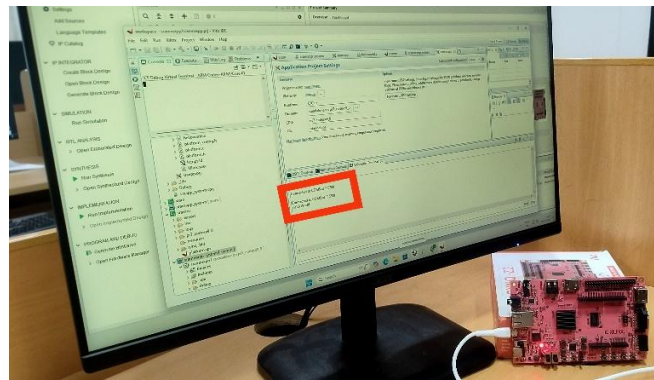


Fig. 11. SRAM Protocol implementation using PYNQ Z2

ASIC Layout

The procedure for putting an integrated circuit (IC) design into physical form. It entails converting the schematic, or logical circuit description, into geometric representations of the different layers that are utilized in the production of semiconductors. The layout for I2C protocol is shown in figure 12.

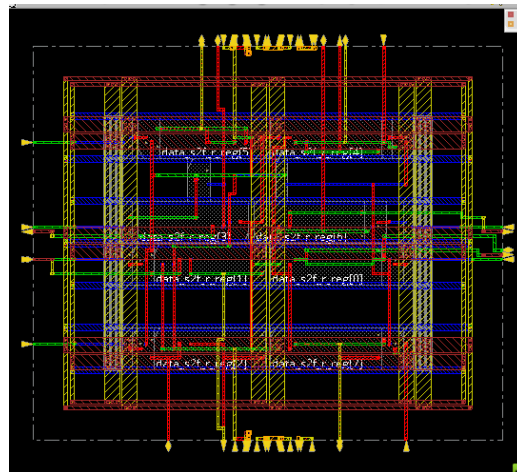


Fig. 12. Layout for SRAM

Design Rule Check

The procedure that confirms a layout complies with a technology process design guideline. DRC is crucial because a produced chip may not function as intended if design guidelines are broken. The DRC (Design Rule Check) report of I2C is shown in Fig 13.

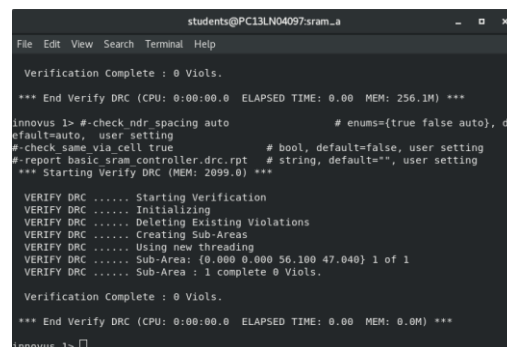


Fig. 13. DRC Report for SRAM

Graphical Data System

The file format used in Electronic Design Automation (EDA) to exchange integrated circuit or layout data. The GDS-II (Graphical Data System II) file for I2C is shown in Fig 14.

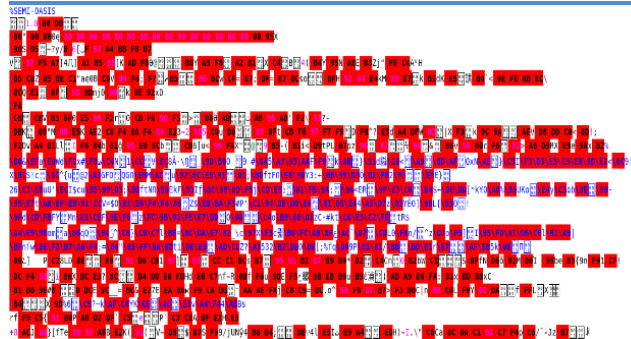
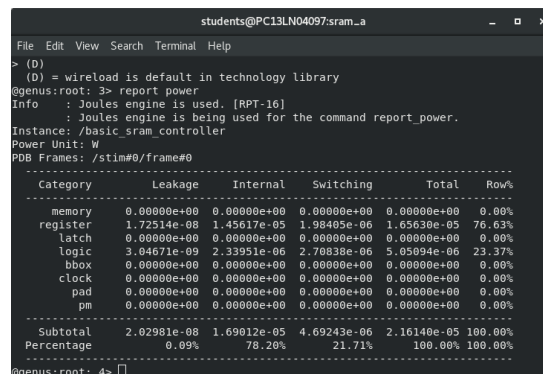


Fig. 14. GDSII output for SRAM

Power Analysis of ASIC

The primary tool used for ASIC power analysis is "Voltus," which offers a comprehensive approach to evaluating power usage. This includes examining dynamic and leakage power, as well as assessing power integrity throughout the entire design process, from the initial RTL stage to the final layout. By utilizing this tool, designers can pinpoint areas of high-power consumption and make necessary adjustments to their ASIC designs, ultimately reducing overall power usage. Fig 15 shows the ASIC power analysis for I2C.



Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	1.72514e-08	1.45617e-05	1.98405e-06	1.65630e-05	76.63%
latch	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
logic	3.04671e-09	2.33951e-06	2.70838e-06	5.05094e-06	23.37%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	2.02981e-08	1.69012e-05	4.69243e-06	2.16140e-05	100.00%
Percentage	0.09%	78.20%	21.71%	100.00%	100.00%

Fig. 15. ASIC Power Report for SRAM

CONCLUSION

A novel SRAM controller for automotive applications, enhanced with AI capabilities, has been successfully developed, validated, and evaluated. This innovative controller is particularly well-suited for modern vehicle electrical systems, as it employs AI-driven optimization techniques to enhance memory access efficiency, reduce latency, and minimize power consumption. The controller's adaptable architecture enables dynamic adjustment to fluctuating workloads, ensuring optimal performance for various applications, including in-vehicle entertainment systems and Advanced Driver Assistance Systems (ADAS).

Extensive testing using FPGA-based prototyping and Universal Verification Methodology (UVM) has validated the controller's effectiveness, reliability, and efficiency. Simulation outcomes demonstrate that the AI-enhanced approach improves memory management efficiency while maintaining low energy consumption, which is essential for automotive applications.

To conclude, the AI-powered adaptable SRAM controller demonstrates enhanced versatility, performance, and reliability, positioning it as a viable solution for future automotive systems. Ongoing research will focus on enhancing

AI algorithms, implementing advanced security protocols, and ensuring compliance with emerging automotive standards to meet the growing needs of smart vehicle electronics.

REFERENCES

- [1] V Karthikeyan, K Balamurugan, Lakshmana Rao Namamula, F Jeya Brindha. "Development of SRAM-APB protocol interface and verification", Engineering Research Express, 2023.
- [2] Xiaohong Peng, Shengxia Wang, Hong Wang, Haonan Tang, Yu Wang, Sen Wang, Meijuan Chen. "Function Verification of SRAM Controller Based on UVM", 2019 IEEE 13th International Conference on Anticounterfeiting, Security, and Identification (ASID), 2019.
- [3] C K Shaila, G Manoj, P.S. Divya, M Vijila. "Functional Verification of SPI Protocol using UVM based on AMBA Architecture for Flash Memory Applications", 2023 4th International Conference on Signal Processing and Communication (ICSPC), 2023.
- [4] Harshitha G M, Sowmya Madhavan, Ashwini N Hegde. "Design of Transmission Gate Based SRAM Using Dynamic Modification Scheme", 2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIEE), 2023
- [5] Pedro Araújo. "Development of a Reconfigurable Multi-Protocol Verification Environment Using UVM Methodology", Repositório Aberto da Universidade do Porto, 2014.
- [6] Introduction to AMBA AXI, Arm Limited, 2020.
- [7] Arun Tigadi, Dr Hansraj Guhilot. "Design and Implementation of SRAM Controller on Reconfigurable Platform", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 14, Number 15 pp. 3325-3329, 2019.
- [8] "AMBA Overview", Arm Developer Limited.
- [9] Sandeep Raval "System Verilog-based Verification of Write Operation in SDRAM using Memory Controller" International Journal for Scientific Research & Development 2 2014 242-4, 2014
- [10] MaC, Liu Z and MaX, "Design and implementation of APB bridge based on AMBA4.0", Int. Conf. on Consumer Electronics, Communications and Networks (CECNet) (IEEE) pp 193-6, 2011.
- [11] Rawat K, Sahni Kand Pandey S, "RTL Implementation for AMBA ASB APB Protocol at System on Chip level" 2nd Int. Conf. on Signal Processing and Integrated Networks (SPIN) (IEEE) pp 927-30, 2015
- [12] C. Spear and G. Tumbush, SystemVerilog for Verification, Third Edition: A Guide to Learning the Testbench Language Features. Springer Publishing Company, Incorporated, 2012
- [13] I. Kastelan and Z. Krajacevic, "Synthesizable system verilog assertions as a methodology for soc," First IEEE Eastern European Conference on the Engineering of Computer Based Systems, pp. 120-127. Sept 2009.
- [14] W. Ni and X. Wang, "Functional coverage-driven uvm-based uart ip verification," IEEE 11th International Conference on ASIC (ASICON), Nov 2015, pp. 1-4, 2015.
- [15] Y. N. Yun, J. B. Kim, N. D. Kim, and B. Min, "Beyond uvm for practical soc verification," International SoC Design Conference, pp. 158-162, Nov 2011
- [16] J. Podivinsky, O. Cekan, J. Lojda and Z. Kotasek, "Functional verification as a tool for monitoring impact of faults in SRAM-based FPGAs," International Conference on Field-Programmable Technology IEEE, pp. 1-3, December 2016.
- [17] L. Shiva and L. Saiteja, "UVM based reusable verification IP for wishbone compliant SPI master core" International Journal of VLSI Design and Communication Systems pp. 21-29, October 2018.
- [18] Tamilselvi M, Vedhanayagi P and Ramasamy K, "VLSI design of low power fault detection in SRAM using BIST" International journal of advanced research in electrical, electronics and instrumentation engineering 6 1-9, 2017.
- [19] Reddy KS, Soujanya P and Sudha D K, "ASIC design and verification of AMBA APB protocol using UVM" International Journal of Innovative Technology and Exploring Engineering (IJITEE) 9 636-42, 2020.
- [20] Dumpa Srinivasa Reddy, Dr. Vijay Prakash Singh, Dr. N. Ashok Kumar, "A Design of 6t-Auto Sensing Mode of Sram for High Speed and Storage Applications" International Journal of Advances in Engineering and Management (IJAEM) Volume 5, Issue 6, pp: 108-113, June 2023.
- [21] Tuanhui Xu; Junlin Huang; Mingen Bu; Zhe Jiang, "An SRAM Test Quality Improvement Method For Automotive chips" IEEE International Test Conference in Asia (ITC-Asia), 2021.

- [22] Suraj Patel, Prof. Paras Gupta, "Low Power Static RAM Design using Memory Banking for Digital Storage" Journal of Computing Technologies (JCT), Page Number: 01-04, Volume-11, Issue-06, June 2022.