International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 11 Issue: 06 | Jun 2024 www.irjet.net p-ISSN: 2395-0072

Design and Implementation of Memory Controller for Byte Access from Data Memory for SoC's Devices

Rajesh N*, Manjesh yadav, Thejash H. S.**, Avinash Vishnu**, Vinayaka V.****

**Assistant Professor Department of Electronics and Communication Engineering, Maharaja Institute of Technology Mysore*
Belwadi, Srirangapatna, Mandya Dist, Karnataka, India rajeshn_ece@mitmysore.in
*** UG students Department of Electronics and Communication Engineering, Maharaja Institute of Technology Mysore*
Belwadi, Srirangapatna, Mandya Dist, Karnataka, India

Abstract- A System-on-Chip (SoC) architecture with multiple processors, alongside integrated memory and control logic, is a powerful approach in modern computing. The multiprocessing-based SoC architecture is commonly used in the latest electronic devices such as Smartphone, tablets and smart wristwatches with large memory sizes. The data handling in these highly memory-dense devices is a critical task, and it needs special attention for smooth operation of the devices. The project proposes a memory controller to exchange data between various processors and input-output devices to tackle these challenges. The controller block uses an AXI (Advanced eXtensible Interface) which is parallel protocol to communicate with other blocks of SoC's. A proposed controller block controls the data flow between memory and different SoC components and processors. A memory access controller (MAC) is presented to manage and accelerate data transmission speed and reduce the processors' activity for SoC-based devices.

Index Terms- System-on-Chip (SoC), Advanced eXtensible Interface (AXI), Memory Access Controller (MAC)

I. INTRODUCTION

Amemory controller is a fundamental component within computing systems, serving as a crucial interface between the central processing unit (CPU) and various types of memory modules. Its primary function is to manage the efficient flow of data between the processor and memory subsystems, ensuring that read and write operations occur seamlessly. Acting as a bridge, the memory controller translates memory access requests initiated by the CPU into specific commands that are understood by the memory devices. This translation process involves address decoding, where memory addresses generated by the CPU are mapped to physical locations in the memory modules.

Furthermore, memory controllers are responsible for coordinating the timing and protocol necessary for data transfer between the CPU and memory. They generate control signals and timing sequences to synchronize the data exchange process, adhering to the timing constraints dictated by the memory standards. By orchestrating data transfers with precision, memory controllers optimize memory performance and minimize latency, thereby enhancing overall system efficiency.

In addition to managing data transfer, memory controllers often incorporate features for error detection and correction. Moreover, memory controllers may support advanced memory management techniques such as interleaving and memory banking, which further optimize memory access and utilization.

In VLSI (Very Large-Scale Integration) design, memory controllers are pivotal components that facilitate efficient data exchange between the processor and memory modules. Acting as an intermediary, these controllers manage various aspects of memory operations, including data transfer, address decoding, and timing control. They translate memory access requests from the CPU into appropriate commands for the memory devices, ensuring seamless communication between the two components. Memory controllers play a critical role in optimizing system performance by orchestrating data transfers with minimal latency and maximizing memory bandwidth utilization. Additionally, they often incorporate features for error detection and correction, enhancing system reliability and data integrity. As computing systems evolve and memory technologies advance, memory controllers continue to adapt, offering support for new memory standards and optimizing memory access for diverse workloads and applications. In essence, memory controllers are fundamental to the efficient operation of modern computing systems, enabling seamless interaction between the processor and memory subsystems.

Overall, memory controllers play a critical role in enabling efficient communication between the CPU and memory subsystems in computing systems. By overseeing data transfer, address decoding, timing control, and error

detection, they contribute to the seamless operation and optimal performance of modern computing platforms.

II. METHODOLOGY

The procedure of designing and implementing involves a systematic approach. Initially, clear requirements are specified, encompassing byte-level access capability, supported memory types, and performance metrics. Next, a thorough understanding of the memory interface protocols and timing constraints is crucial for compatibility. SystemVerilog is then utilized to implement the design, employing modules, interfaces, and state machines to encapsulate functionality and promote modularity. Throughout the process, thorough testing and verification ensure that the memory controller meets specifications and performs efficiently.

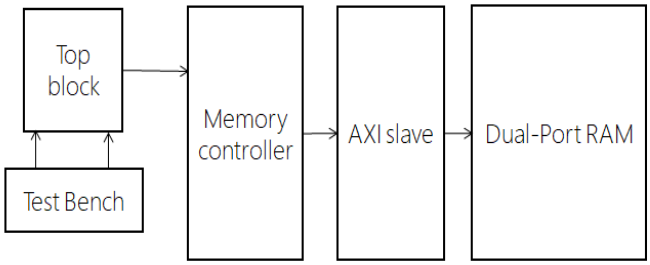

Fig. 1: AXI Memory-controller block

Figure 1 above shows the overview of the design. We construct the blocks separately and then the complete designs are instantiated by the top block. The test benches are applied to the top block.

Dual-port RAM is 32-bit width and 8-bit depth. It consists of two ports for dual operation of read-write. It is active high designed block. We can access the any of the two ports by configuring the signals.

Memory controller block is the main part of the system. It is the controlling unit of the system. It is brain of the system. It commands the dual-port RAM. It stores the details like address and data to be transmitted into and from the memory. It acts as master commanding the AXI slave.

AXI slave is the intermediate between memory controller and RAM it is parallel protocol. It controls the data flow between the memory controller and RAM. It acts as a bridge between memory controller and RAM.

Memory Controller:

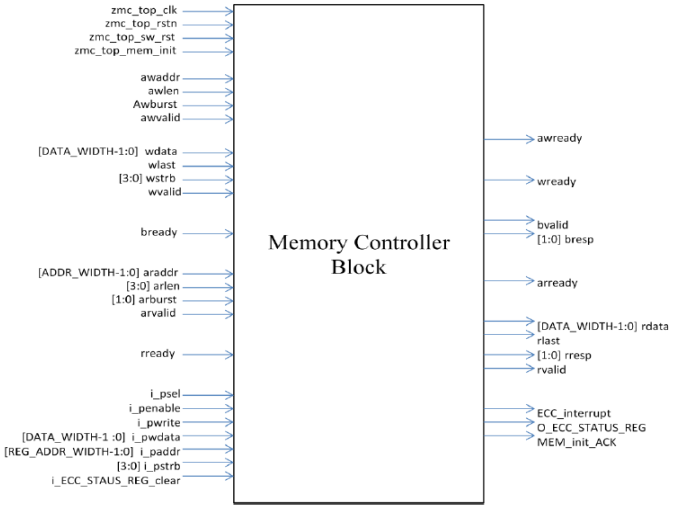

Fig. 2: Memory controller block

Memory controller is the intermediate block unit which is the heart of our design. It takes the command from the top block. We can say this unit as CPU of our design. It gives the command to AXI slave to operate the dual-port RAM.

It is the main unit which performs the operation of read-write into and from the dual-port RAM.

Figure 2 shows the block of memory controller. It has the following signals to operate:

- Memory controller signals
- Write address channel signals
- Write data channel signals
- Write response channel signals
- Read address channel signals
- Read data channel signals
- Registers signals

A memory controller is a critical component in digital systems responsible for managing the interaction between the CPU or processing unit and the memory subsystem. Its primary function is to ensure efficient and reliable access to memory resources, including RAM, ROM, and other storage devices.

At its core, the memory controller orchestrates the flow of data between the CPU and memory modules, handling tasks

such as address decoding, data transfer, and timing control. It translates memory access requests from the CPU into the appropriate commands and signals for accessing the memory modules, ensuring that data is read from or written to the correct memory locations in a timely manner.

Moreover, memory controllers often incorporate features for optimizing memory performance and reliability, such as caching, error detection and correction, and memory interleaving. These capabilities help enhance system throughput, reduce access latency, and ensure data integrity, thereby improving overall system performance and stability.

In summary, the memory controller plays a crucial role in enabling efficient and reliable communication between the CPU and memory subsystem in digital systems, contributing significantly to their overall performance and functionality. Its proper design and implementation are essential for achieving optimal system performance and meeting the requirements of various computing applications.

AXI slave:

Advanced eXtensible Interface (AXI) is a part of ARM Advanced Microcontroller Bus architecture specifications. It is a parallel high-performance, synchronous, high-frequency, multi-initiator, multi-target communication interface, mainly designed for on-chip communication. The AXI-4 protocol is the fourth generation of the AMBA interface specifications. It is an update to AXI3 which is designed to enhance the performance. It includes the following enhancements: Support for burst lengths up to 256 bits, as shown in figure 3.

The AXI protocol is burst-based and defines the following independent transaction channels:

- read address
- read data
- write address
- write data
- Write response.

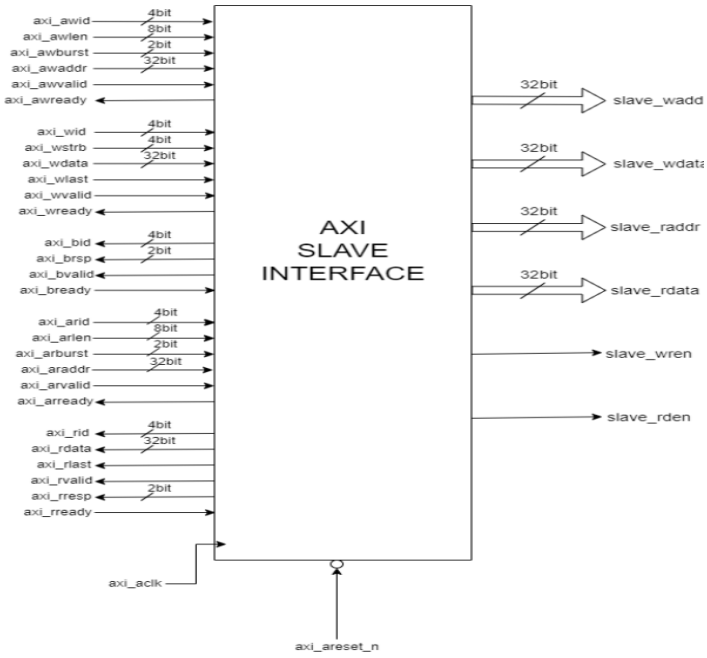

Fig. 3: AXI slave to drive dual port RAM

Dual-port RAM:

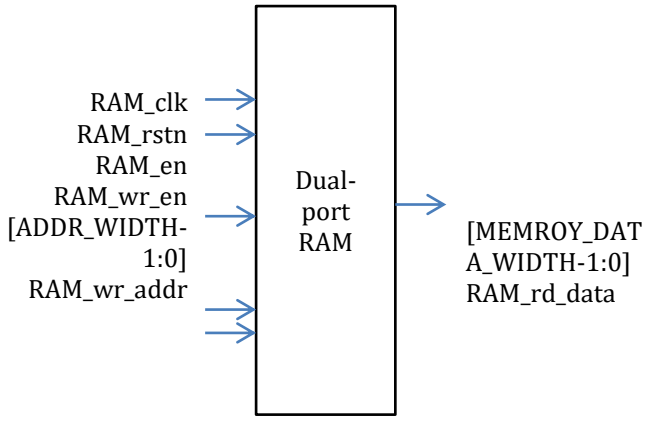

Fig. 4: Dual-port RAM

Dual-port RAM (Random Access Memory) is a type of memory that allows two independent accesses to the memory array simultaneously. Each port typically has its own data input, data output, and address input, allowing two different devices or components to access the memory simultaneously without interfering with each other, as shown in figure 4.

III. Operation

The design consists of Memory controller acting as master and dual port RAM as slave to perform the read-write. It has five channels to perform the operation, as shown in figure 5:

- Read address channel
- Read data channel
- Write address channel
- Write data channel
- Write response channel

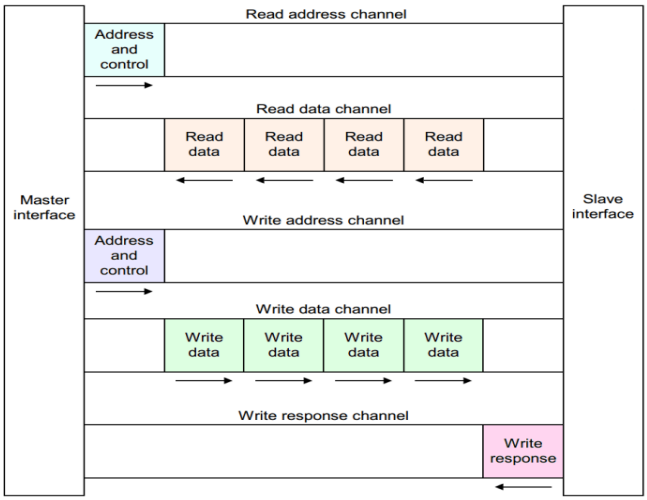

Fig. 5: Operations of Memory controller and AXI slave to drive dual port RAM

IV. Implementation

FPGA or Field Programmable Gate Arrays can be programmed or configured by the user or designer after manufacturing and during implementation. Hence, they are otherwise known as On-Site programmable. Unlike a Programmable Array Logic (PAL) or other programmable device, their structure is similar to that of a gate-array or an ASIC. Thus, they are used to rapidly prototype ASICs, or as a substitute for places where an ASIC will eventually be used. This is done when it is important to get the design to the market first. Later on, when the ASIC is produced in bulk to reduce the NRE cost, it can replace the FPGA. The programming of the FPGA is done using a logic circuit diagram or a source code using a Hardware Description Language (HDL) to specify how the chip should work. FPGAs have programmable logic components called ‚logic blocks, and a hierarchy or reconfigurable interconnects which facilitate the ‚wiring' of the blocks together. The programmable logic blocks are referred to as configurable logic blocks and reconfigurable interconnects are referred to as switch boxes. CLBs can be programmed to perform complex combinational functions, or simple logic gates. In most FPGAs the logic blocks also include memory elements, which can be as simple as flip-flops, or as complex as complete blocks of memory.

Major Implementation steps:

1. Understanding of AXI protocol and Read-Write-Modify operations to know the data flow from and into the Memory block.
2. Design of dual port RAMS to store and retrieve the data.
3. Design of Memory controller unit to control the data flow in the dual port RAMS.
4. Design of AXI slave for AXI protocol-based data transmission and reception from the dual port RAM.
5. Design of top block to instantiate the lower-level blocks.
6. Verifying the functionalities of top block by using test bench.

Tools and Resources:

1. RTL Designing: Cadence xcelium
2. Verification: Simvision
3. Synthesis: Genus

V. Results

The test case used here is using direct test bench. Simply, memory flash is assumed to be initially stores some data in specific addresses. The scenario as follows, the host sends read command and then the initially stored data is transferred into the buffer to be serialized and sent to the host on the data serial link. Write command is issued by the host to store new data in the memory flash core. The host sends the data form on the serial data link to be stored in the buffer. Then the data in the buffer is transferred to the memory flash core. The old data and new data both are exist in the memory core, i.e. no overwrite occurred. Finally, erase command is issued by the host to delete all stored data in the memory flash core. The command is transferred from host to device serially. It takes 19 simulation clock cycles to store the whole command frame in the device registers. Data to be transferred between the memory flash core and the buffer takes just 1 simulation clock cycle. Notice that before the host

begins the communication with the device, it must reset the device; so hardware reset will be issued before the read command.

- First, the host assert hardware reset pin for 1 clock cycle, after this reset pin de-asserted again as shown in figure 6 .

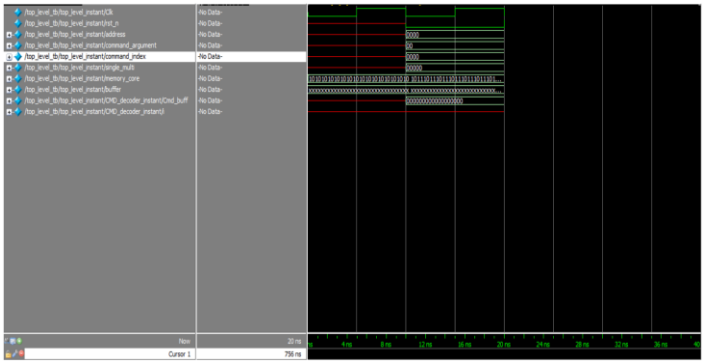

Fig. 6: Reset pin asserted for 1 clock cycle and the de-asserted

- Memory flash core is initially sore some random data blocks – 6 data blocks from address 0h to address 5h as shown in figure 6.

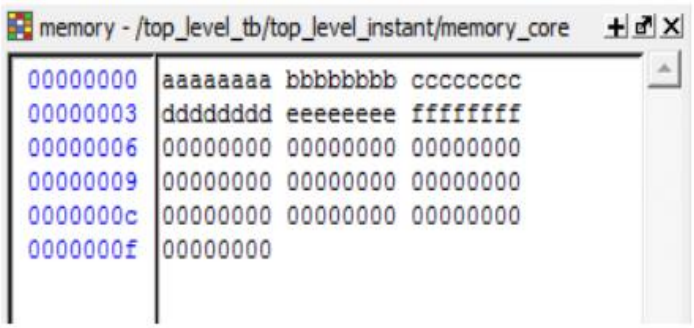

Fig. 6: Memory flash core is initialized with random data

- Host starts to issue read command as shown in figure 7, at clock cycle 220 the read command frame is completely stored in the device.

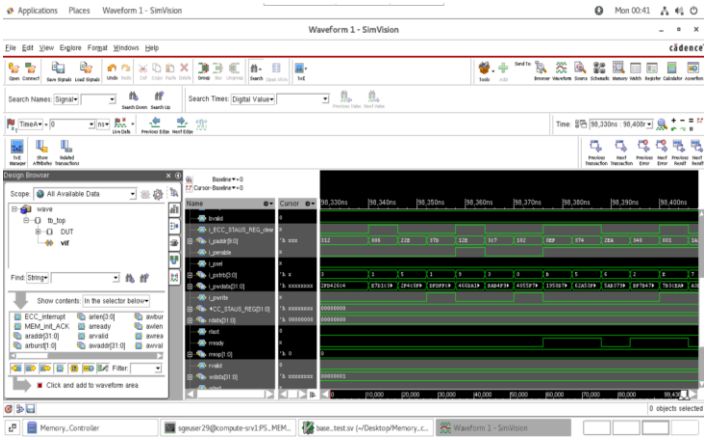

Fig. 7: read command frame is completely stored in device

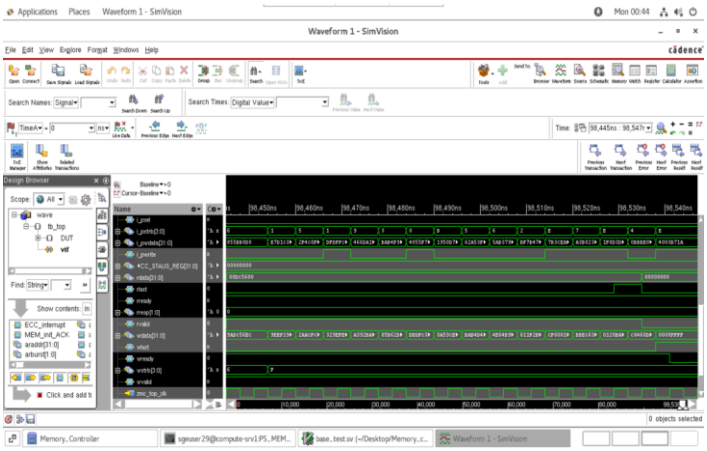

Fig. 7: Memory flash core is transferred successfully to the buffer

- New data is transferred from buffer to the flash memory core successfully

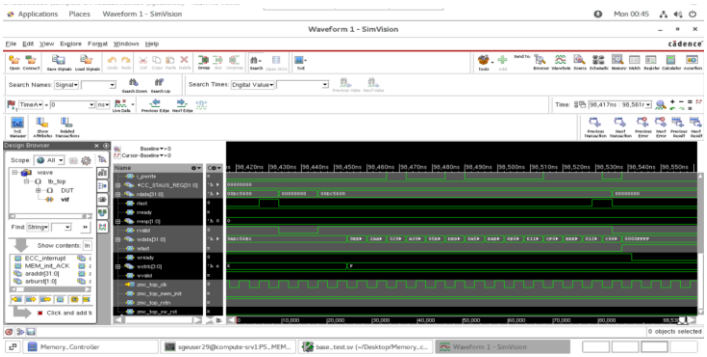

Fig. 8: New data is transferred from buffer to the memory core successfully

- The design of the memory controller built is synthesized using cadence genus where the gate

level netlist along with the block interleaving can be seen in the system.

- The figure 9 shows the design where individual block explained above in the block diagram can be seen.

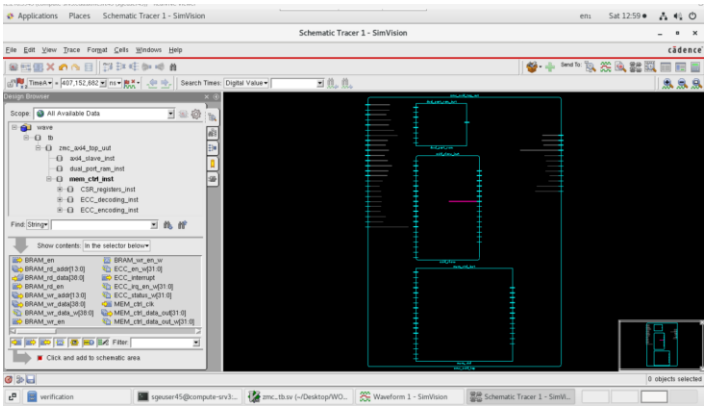

Fig. 9: Synthesis of Memory controller

- The design contains the Memory controller unit along with dual-port RAM which is interfaced using AXI protocol as shown in the figure 10 and figure 11.

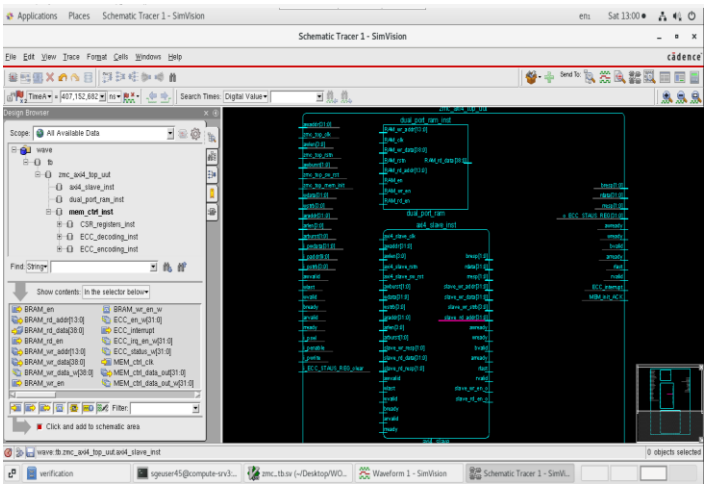

Fig. 10: AXI slave and dual-port RAM

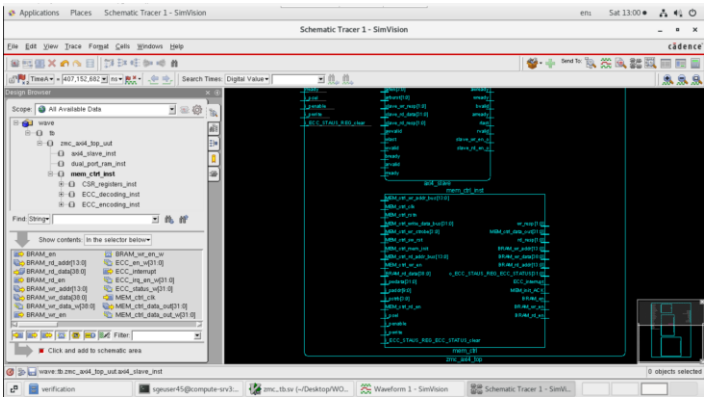

Fig. 11: Memory controller interfaced with AXI slave

VI. Conclusion

The memory controller architecture, designed and verified using Verilog and SystemVerilog UVM, respectively, showcases a balance between simplicity and integration, seamlessly merging Flash and DRAM memory types. Its design prioritizes power efficiency, aligning with global trends, and incorporates powerful features from six diverse protocols for enhanced functionality. With support for parallel operations, up to two simultaneous operations boost performance, while its multi-point to single-point model enables communication with multiple hosts, though limited to four in a switching manner. Manufacturers benefit from its configurable nature, adaptable to various applications, while data security remains paramount, with encryption implemented in both memory cores to safeguard against potential hacks. This architecture serves as a blueprint for future designs, providing invaluable insights into essential features for upcoming architectural developments.

VII. Future Scope

For every door your close in research, two new doors are opened. This section discusses interesting future work and open issues in the context of this work. Although this project contains a lot of important features, more advanced features can be implemented in by developing the used techniques in the proposed memory controller. The current design implementation can move in the rest of the FPGA design flow to get the memory controller. The future scope of memory controllers for byte access in System-on-Chip (SoC) designs is poised for significant advancements driven by emerging trends such as the proliferation of IoT and edge computing, the demand for AI and machine learning acceleration, and the adoption of heterogeneous memory architectures.

REFERENCES

[1]. Mohammed Altaf Ahmed and Jaber Aloufi, "A Smart Memory Controller forSystem on Chip-Based Devices",in Department of Computer Engineering, College of Computer Engineering & Sciences, Prince Sattan Bin Abdulaziz University,Alkharj-11942, Saudi Arabia.

[2]. Rashmi Samanth, Subramanya G. Nayak, "Design and SV Based Verification of AMBA AXI Protocol for SOC Integration" in International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878 (Online), Volume-8 Issue- 2, July 2019.

[3]. Khaled Khalifa, Haytham Fawzy, Sameh El-Ashry, Khaled Salah, "Memory Controller Architectures: A comparative Study" by Sameh El-Ashry on 20 September 2015.

[4]. Gregorio Zlotnik and Aaron Vansintjan, "Memory: An Extended Definition" in Clinique de la Migraine de Montreal, Montreal, QC, Canada, 2 Department of Film, Media and Cultural Studies, Birkbeck, University of London, London, United Kingdom.

[5]. Mohammed Altaf Ahmed, Abdullah Aljumah and M. Gulam Ahmad, "Design and Implementation of direct memory access controller for embedded systems" in Department of Computer Engineering, College of Computer Engineering & Sciences, Prince Sattam Bin Abdulaziz University, Alkharj 11942, Saudi Arabia.

[6]. Prof. Dr. Mohamed Rizk, Alexandria University Dr. Khaled Salah, Mentor Graphics Egypt, "Thesis book of Universal Memory Controller", Alexandria University, Egypt, July 2014.

[7] Balakrishna K. Rajesh N "Design of remote monitored solar powered grass cutter robot with obstacle avoidance using IoT", Global Transitions Proceedings Volume 3, Issue 1, June 2022, Pages 109-113

[8] Rajesh N Li-Fi (Light Fidelity): The Future Vision In Wireless Communication, IJRECE Volume 9,Issue3 JULY-SEPT2021 ISSN: 2393-9028 (PRINT) |ISSN: 2348-2281|

[9]M.Rajendra and N.Suresh Babu. "Speed and Area optimized Design of DDR3 SDRAM (Double Data Rate3 Synchronously Dynamic RAM) Controller for Digital TV Decoders", International Journal of Engineering Trends and Technology", 2013, Vol.06 Issue 4, pp 204-211.

[10] Ms.Seema Sinha and Md.Tariq Anwar. "design and verification of ddr3 memory controller".International Journal of Advanced Technology in Engineering and science, 2014, Vol.02, Issue 05,pp.199-207