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

Preethi K[1], Pruthvika R[2], Vaishnavika SG[3], Vathsala HK[4], Manasa MG[5]

[1,2,3,4,5]Department of Electronics and Communication Engineering, Sapthagiri College of Engineering, Bengaluru Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi

Abstract:
A System-on Modern computing systems, particularly System-on-Chip (SoC) architectures, incorporate multiple processors, integrated memory, and control logic to enhance efficiency. These architectures are prevalent in contemporary electronic devices like smartphones, tablets, and smartwatches, all of which demand high-performance memory management. Ensuring smooth data exchange within these memory-intensive devices is crucial for optimal functionality. This paper presents a specialized memory controller designed to regulate data transfer between multiple processors and peripheral components. The controller is based on the Advanced eXtensible Interface (AXI) protocol, which facilitates parallel communication between various SoC components. The proposed memory access controller (MAC) efficiently manages data transmission speeds, minimizes processor workload, and enhances overall system performance.

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

I. Introduction

A memory controller plays a pivotal role in computing systems by serving as an intermediary between the central processing unit (CPU) and different memory modules. It is responsible for directing data flow between these units, ensuring efficient read and write operations. The memory controller converts CPU-generated access requests into commands compatible with memory hardware. This involves address decoding, where CPU-specified memory locations are mapped to their physical counterparts in memory modules.

Additionally, the controller synchronizes data transfers, adhering to predefined timing constraints and memory protocols. It generates control signals and ensures precise coordination to maintain system efficiency. Many modern memory controllers integrate error detection and correction mechanisms to enhance reliability.

control signals and ensures precise coordination to maintain system efficiency. Many modern memory controllers integrate error detection and correction mechanisms to enhance reliability. Advanced features such as memory interleaving and banking further optimize performance by improving data access speed and distribution.

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 the domain of Very Large-Scale Integration (VLSI) design, memory controllers are indispensable, as they manage multiple aspects of data exchange, from command translation to execution. By optimizing memory bandwidth and reducing access latency, these controllers contribute significantly to overall system efficiency. As computing technology evolves, memory controllers continue to incorporate cutting-edge features, supporting new memory standards and adapting to various workload requirements.
Overall enabling efficient communication subsystems in computing systems. By overseeing data transfer, address decoding, timing control, procedure of designing and implementing involves a systematic approach. Initially, clear requirements are specified, encompassing byte-level access capability, of the memory interface protocols and timing constraints is crucial for timings.

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.

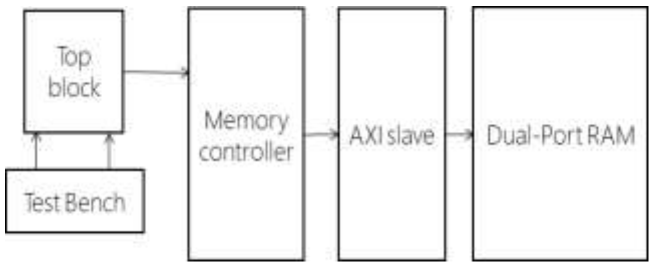


Fig. 1: AXI Memory-controller block

The proposed design consists of several interconnected components, as illustrated in Figure 1

1. Dual-Port RAM: A 32-bit wide, 8-bit deep memory unit that supports simultaneous read and write operations. It is an active- high block, and signal configuration allows access to either port.
2. Memory Controller: The core unit that regulates data flow, stores essential information such as addresses, and commands the AXI slave interface.
3. AXI Slave: An intermediate communication module between the memory controller and RAM, facilitating parallel data transfer.

The memory controller serves as the primary processing unit, issuing commands to the AXI slave to coordinate dual-port RAM operations.
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.
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.

Memory Controller:

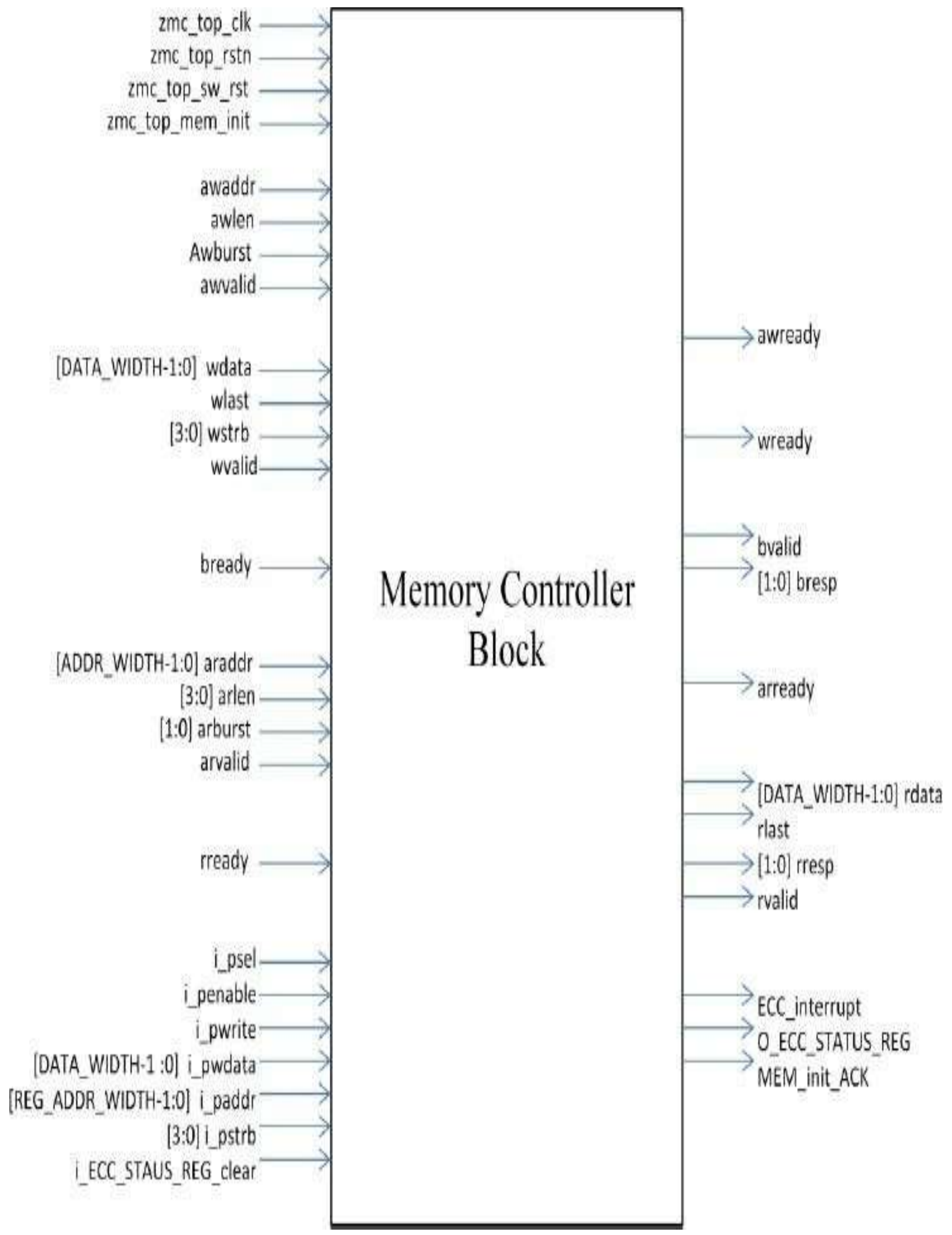


Fig. 2: Memory controller block

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. The memory controller serves as a crucial component of the system, functioning as an intermediate control unit that facilitates communication between different blocks. It operates as the core processing unit of the design, receiving commands from the top-level module and instructing the AXI slave to execute operations on the dual-port RAM
The dual-port RAM features a 32-bit data width and an 8-bit depth, enabling simultaneous read and write operations through two distinct ports. As an active-high module, it only functions when the

corresponding control signals are activated. The memory controller regulates which port is accessed by configuring control signals appropriately.

The memory controller is responsible for storing memory addresses and data, ensuring efficient communication between memory and processing units. Acting as the primary control entity, it sends directives to the AXI slave, which then facilitates data transfer between the memory and other system components. By optimizing data flow, it enhances system performance and minimizes delays.

Figure 2 depicts the memory controller block, which includes several operational signals:

- Control signals for memory management
- Write channel address signals
- Write data channel signals
- Write response channel signals
- Read address channel signals
- Read data channel signals
- Register control signals

AXI slave:

The AXI slave module functions as a link between the memory controller and dual-port RAM, utilizing a parallel communication protocol to ensure smooth data exchange. It regulates the data flow, ensuring proper interaction between the memory controller and RAM, thereby playing a critical role in system efficiency.

The Advanced eXtensible Interface (AXI) is a component of ARM's Advanced Microcontroller Bus Architecture (AMBA). It is a high- performance, synchronous, multi-initiator, multi-target interface designed primarily for on-chip data communication. The AXI-4 protocol, an advancement over AXI-3, introduces improved data handling capabilities, including support for burst lengths of up to 256 bits, as shown in Figure 3.

These channels operate simultaneously, enabling efficient memory transactions while reducing processing overhead. The AXI slave module ensures seamless coordination between the memory controller and dual-port RAM, improving overall system performance and reliability.

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

- read address
- read data
- write address
- write data

III. Operation

coordinating memory operations efficiently.

To develop the system, Verilog and SystemVerilog were utilized for The design consists of Memory
controller acting as master and design and verification. The functional correctness was validated using
dual port RAM as slave to perform the read-write. It has five testbenches, ensuring compliance
with performance metrics. The channels to perform the operation

- Address read channel
- Data read channel

- Address write channel
- Data write channel
- Response write channel

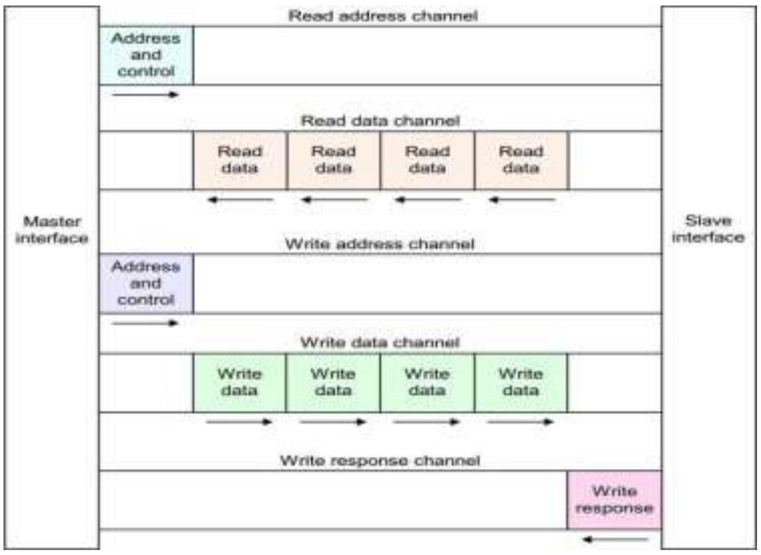


Fig. 5: Operations of Memory controller and AXI slave to drive dual port RAM

IV. Implementation

The implementation of the memory controller was carried out using a structured approach to ensure efficient functionality within an SoC environment. The design was developed on a Field-Programmable Gate Array (FPGA), which offers flexibility and reconfigurability, making it suitable for rapid prototyping. Unlike fixed-function ASICs, FPGAs allow designers to modify the architecture based on specific system requirements.

The memory controller consists of various interconnected components, including a dual-port RAM, an AXI-based slave module, and a top- level control unit. Each of these blocks was individually designed and later integrated into a complete system. The dual-port RAM facilitates simultaneous read and write operations, ensuring faster data access. The AXI slave serves as an intermediary between the memory controller and RAM, handling data transfer with parallel processing capabilities. The memory controller unit acts as the central control block, managing data flow, generating control signals, and synthesized design was implemented using Cadence Genus, which provided a gate-level netlist representation, optimizing area and power consumption. The proposed architecture successfully manages memory access, reducing processor workload while enhancing overall system performance.

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 .

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

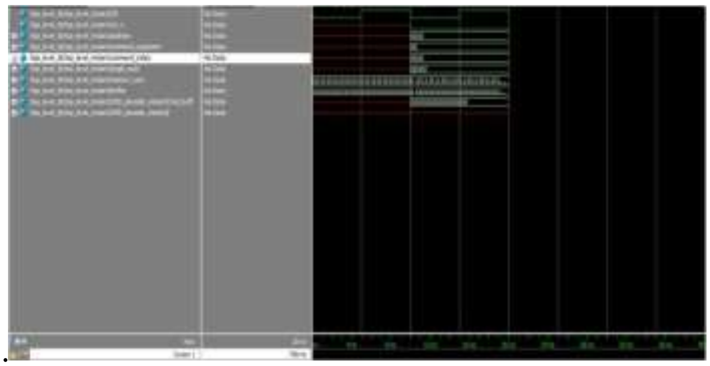
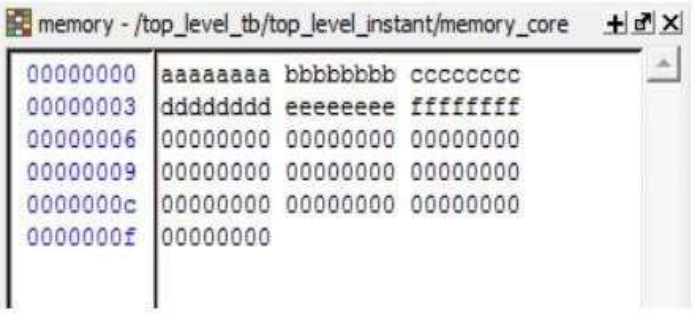


Fig. 6: Reset pin asserted for 1 clock cycle and the de-asserted

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



```
memory - /top_level_tb/top_level_instant/memory_core
00000000  aaaaaaaa bbbbbbbb cccccccc
00000003  dddddddd eeeeeeee ffffffff
00000006  00000000 00000000 00000000
00000009  00000000 00000000 00000000
0000000c  00000000 00000000 00000000
0000000f  00000000
```

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.

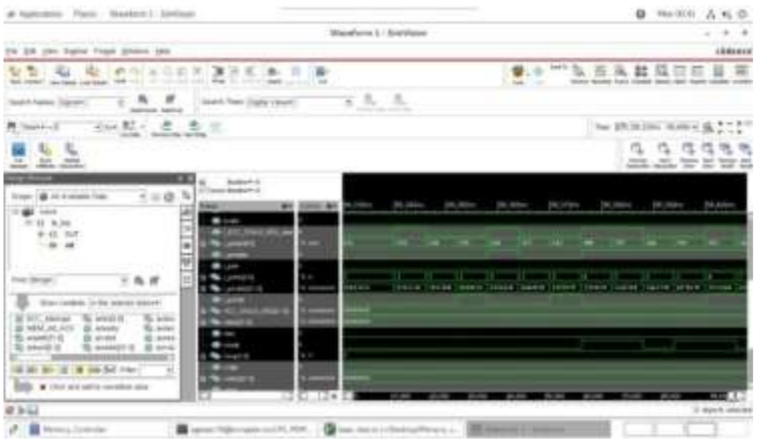


Fig. 7: read command frame is completely stored in device

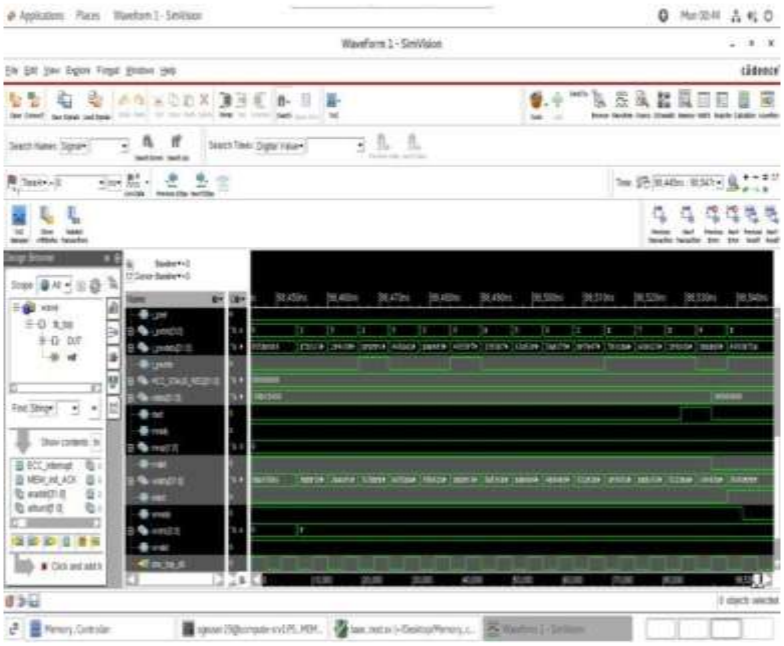


Fig. 7: Memory flash core is transferred successfully to the buffer

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

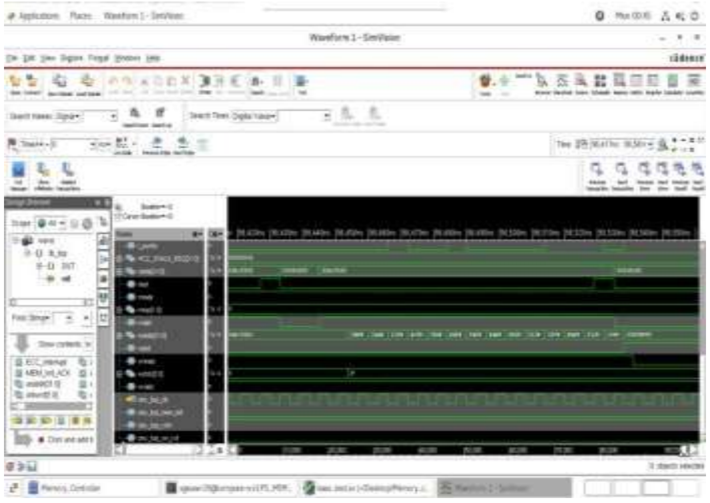


Fig. 8: New data is transferred from buffer to the memory core successfully

- 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.

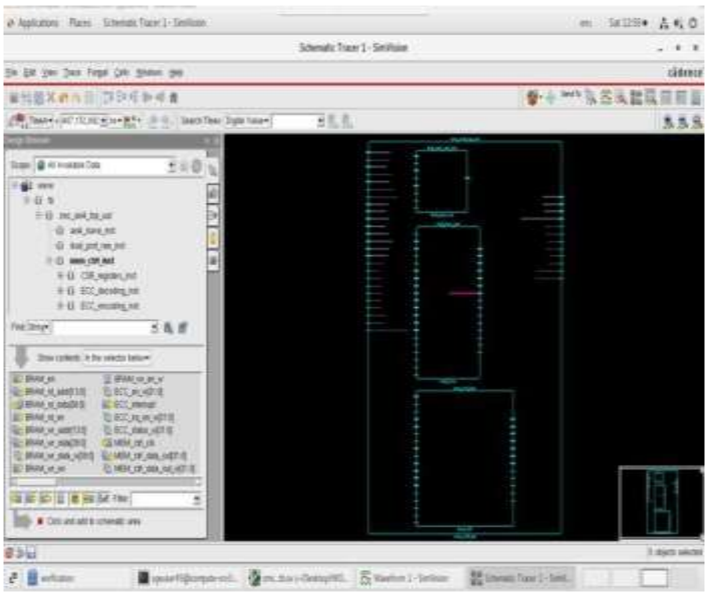


Fig. 9: Synthesis of Memory controller

- 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.

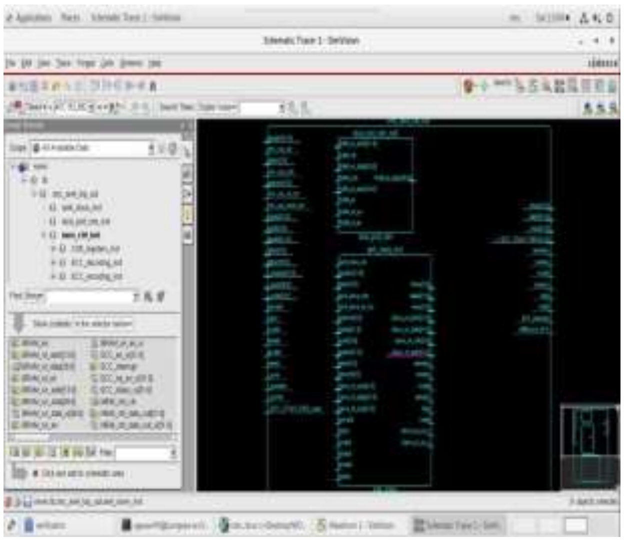


Fig. 10: AXI slave and dual-port RAM

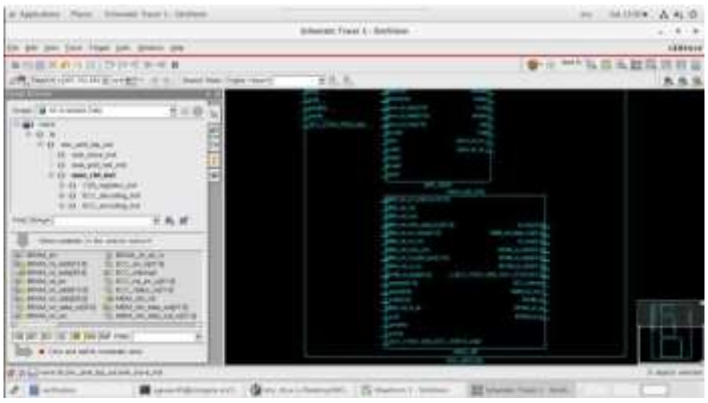


Fig. 11: Memory controller interfaced with AXI slave

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. It contains a important features, more advanced features by developing the used techniques in the proposed memory controller. The current design implementation can move 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 for System 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