International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7214
***Article Received**: 28 April 2023 **Revised:** 14 June 2023 **Accepted:** 30 June 2023*

Design and Develop Efficient Arbitration Technique to Handle the Multiple Refresh Requests in Multi-Processor SoC

Chintam Shravan[1], Kaleem Fatima[2], Paidimarry Chandra sekhar[3]
[1] Department of Electronics and Communication Engineering
University College of Engineering, Osmania University
Hyderabad, India
shravanchintam@gmail.com
[2] Department of Electronics and Communication Engineering
Muffakham Jah College of Engineering & Technology
Hyderabad, India
kaleemfatima@mjcollege.ac.in
[3] Department of Electronics and Communication Engineering
University College of Engineering, Osmania University
Hyderabad, India
sekharp@osmania.ac.in

Abstract—Emerging memory technologies, such as Gain Cell-embedded Dynamic Random Access Memory (GC-eDRAM), play an essential part in the process of improving the overall performance of current multi-processor systems. GC-eDRAM, on the other hand, has its own set of distinct issues, particularly with regard to refresh operations. The number of cores and threads in contemporary processors continues to expand, which in turn leads to an increase in the number of concurrent refresh requests. This might cause contention, which in turn can lead to a possible performance decrease. In this article, we present an efficient arbitration method that was developed in order to precisely address the issues that are associated with numerous requests for a refresh in GC-eDRAM. This method takes use of the inherent parallelism of GC-eDRAM modules to make it possible to execute simultaneous refresh operations. As a result, contention is effectively reduced, and the overall performance of the system is improved. We provide a new arbitration method that prioritizes the pending refresh requests according to their level of urgency and optimizes the allocation of GC-eDRAM resources in order to guarantee that refresh operations are carried out in an effective manner. Our method modifies the arbitration priority in a dynamic manner according to the characteristics of the active workload. These characteristics include the request arrival rate, memory access patterns, and data location, among other considerations.

Keywords- GC-eDRAM, Efficient Arbitration, Multi-Processor SoC, Memory controller, Refresh requests, AWT, Burst length, Fixed-Priority Scheduling.

I. INTRODUCTION

Because of their capacity to give high-performance computing capabilities across a broad variety of applications in today's fast developing technological environment, multi-processor Systems-on-Chip (SoCs) have become more widespread [1]. This is due to their ability to deliver high-performance computing capabilities. This popularity is a direct result of the integration of numerous processors into a single chip, which, in turn, creates a variety of issues regarding the effective administration of available resources. The efficient management of many refresh requests in multi-processor SoCs is one of the most important challenges that needs careful attention [2].

In order to avoid data loss caused by charge leakage and maintain the system's data integrity and dependability, the dynamic random-access memory, or DRAM, in a computer has to have frequent refresh operations performed on it [3]. Multiple requests for a refresh may be made all at once in systems on a chip (SoC) that have more than one processor and where each CPU may have its own set of DRAM modules. It is very necessary to effectively manage these requests in order to maximize the efficiency of the system and minimize any possible bottlenecks [4].

When it comes to processing numerous refresh requests in multi-processor SoCs, having an effective arbitration system is one of the most important components that can be implemented. Arbitration mechanisms decide the priority and order in which

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7214
***Article Received**: 28 April 2023 **Revised:** 14 June 2023 **Accepted:** 30 June 2023*

refresh requests are served. This helps to ensure that resources are allocated fairly and keeps wait times to a minimum. In order to develop a method of effective arbitration, it is necessary to take into consideration a number of criteria, including the minimization of refresh latency, the reduction of overhead, and the optimization of overall system performance [5]. Efficient memory management strategies are crucial due to the increasing complexity of multi-processor SoC designs and their demanding requirements. Dealing with repeated refresh requests sent to the memory controller of Gain Cell-embedded Dynamic Random Access Memory (GC-eDRAM) poses a significant challenge [6]. To ensure optimal performance and resource utilization, effective arbitration in these systems is necessary. GC-eDRAM is a specialized memory technology commonly utilized in high-performance computing environments and graphics-intensive applications. Its purpose is to enable fast data retrieval and manipulation by providing quick access to memory resources, thereby reducing wait times. However, as the number of processors in an SoC continues to rise, the memory controller faces an escalating number of simultaneous refresh requests from multiple processors. This can lead to contention and a decrease in performance.

The arbitration method must address various challenges, including request prioritization, efficient utilization of shared resources, low latency processing, fairness among processors, scalability with increasing processor counts, and power efficiency. Solving these problems is crucial to improving overall system performance and maintaining the stability of multi-processor SoC designs. By inventing an effective arbitration approach, our goal is to enhance the performance of GC-eDRAM in multi-processor SoCs. This will facilitate the smooth and uninterrupted execution of graphics-intensive tasks, real-time applications, and high-performance computing workloads. Effective implementation of this technology will improve memory management and contribute to the overall advancement of multi-processor SoC designs.

There are some other challenges in efficient arbitration in multi-processor Systems-on-Chip. The first major obstacle is the occurrence of multiple refresh requests from different processors within the SoC. Each processor may have different priority levels or degrees of urgency in accessing shared resources. Therefore, it is crucial to devise an arbitration method capable of effectively managing and prioritizing these requests. Utilizing available system resources effectively is another challenge. Refresh requests often involve accessing and modifying data stored in shared memory or cache lines. Coordinating and managing these operations across multiple processors while minimizing conflicts and maximizing resource utilization is essential [7]. Furthermore, minimizing latency in the arbitration process is a crucial consideration. The time taken to process refresh requests should be minimized to avoid unnecessary delays in the overall system operation, particularly in real-time systems where timely responses are critical.

Ensuring fairness in arbitration presents another difficulty. It is important to establish a mechanism that distributes resources equitably among processors, preventing any single processor from monopolizing the system's resources. Maintaining fairness is vital to prevent system instability and performance degradation. Scalability is also a significant factor to address. As the number of processors in the SoC increases, the arbitration method should handle the growing complexity and workload efficiently. The system should exhibit good scalability, accommodating an expanding number of processors without sacrificing performance or introducing bottlenecks. Lastly, power efficiency remains an ongoing concern in SoC design. The arbitration method should strive to minimize power consumption during refresh operations. Incorporating power-aware design principles and techniques is necessary to optimize energy usage while meeting performance requirements.

The objective of this research is to design and develop an efficient arbitration approach tailored to managing the numerous refresh requests in the memory controller of GC-eDRAM within a multi-processor SoC. The research work specifically aims to create an effective technique for handling multiple refresh requests, ensuring equitable access to memory resources while minimizing delays and maximizing resource utilization. The organization of the paper as follows: Literature is discussed in section-II and the proposed method is explained in section-III. The simulation results are discussed in section-IV.

II. LITERATURE

In research work [8], authors construct a FIFO using GC-eDRAM by making use of the specific access patterns revealed by the FIFO system. The results of this work show that the functionality of this GC-eDRAM based FIFO is equal to that of an SRAM-based FIFO. The proposed FIFO eliminates all of these problems, in contrast to conventional FIFOs, which have access blockage time as a result of refresh operations and issues that are connected to data integrity. As a consequence of this, it is capable of replacing FIFOs in both present systems and future designs with no compatibility issues whatsoever. In addition, as comparison to SRAM, the proposed FIFO provides considerable benefits in terms of space and power efficiency, with possible savings of up to two times.

In the research article [9], author offer an original implementation of a single-well mixed 3T GC that uses the 28 nm FD-SOI technology. The proposed GC is equipped with body-bias control to increase the DRT by reducing leakage via the write port and to prolong the maximum operating frequency

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7214
***Article Received**: 28 April 2023 **Revised:** 14 June 2023 **Accepted:** 30 June 2023*

by forward body-biasing the read port. Both of these improvements are made possible thanks to the GC's incorporation of body-bias control. The proposed 3T GC was implemented in a 24 kbit GC-eDRAM macro that was fabricated in 28 nm FD-SOI technology. The end result was the highest density logic-compatible embedded memory fabricated in any 28 nm process, with over 2 times the density of a 6T SRAM cell, over 4 times the DRT of a conventional 3T GC, and 38-47 times lower static power compared to conventional single-ported and two-ported SRAMs.

Marco Widmer et.al [10] proposed a platform that utilizes FPGAs and incorporates defective eDRAM emulation. The emulation process involves advanced retention time models and silicon measurements. The primary objective of this platform is to assess the statistical error resistance of applications within a comprehensive embedded system. In their study, the authors analyze the statistical quality of service for numerous benchmarks by employing different sub-critical refresh rates and retention time distributions.

The research work [11] suggests a 5T GC-eDRAM bit cell based on Fin FET that tackles the SRAM leakage power problem without requiring bandwidth-intensive refreshes. Furthermore, the gain-cell design is based on the Fin FET, which solves the short-channel issues, to demonstrate the viability of scaling down transistors. To assure the static preservation of both "1" and "0" data, the proposed 5T bit cell offers more internal feedback than the 4T all-nMOS completely depleted-silicon on insulator gain cell. A 7-nm Fin FET predictive technology model was used to simulate the 2-kB array using HSPICE. The simulation findings demonstrate that, in comparison to the 4T all-nMOS GC-eDRAM cell with a 28-nm FD-SOI technology, the proposed 5T bit-cell provides around 13 smaller data retention power and 10 area reduction.

In the research work [12], Changmin Lee et.al propose a Non-Volatile Dual In-line Memory Module architecture that incorporates several system-wide techniques to enable non-deterministic timing, specifically asynchronous timing, on synchronous DDR4 memory interfaces. To validate our proposal, we develop a functional prototype of the memory architecture on an x86-64 server system. The prototype is thoroughly tested using a combination of simulated workloads and actual workloads to assess its performance and functionality.

Jinhua Cui et.al [13] proposed a unique partial-refresh data refresh technique for 3D NAND flash memory in the context of cyber-physical systems. partial-refresh uses Low-Density Parity-Check codes' detectability to pinpoint cells that are more prone to mistakes. The SSD's lifetime is increased and refresh costs are decreased as a consequence of PR's selective relocation of these sensitive bits to new pages as opposed to duplicating a complete page. Our test findings show that a PR-enhanced flash memory system beats cutting-edge options by boosting refresh speed by 28.2% and extending SSD lifetime by 4.6% while retaining excellent data dependability.

Lukas Steiner et.al [14] presented DRAMSys4.0, an open-source DRAM simulator that is believed to be the fastest and most feature-rich in its class. The groundbreaking architecture of DRAMSys4.0 incorporates a Domain Specific Language, enabling seamless adaptation to new DRAM standards. Authors described optimization techniques employed to enhance simulation performance while ensuring accurate temporal representation. Additionally, authors conducted a comprehensive analysis and comparison of various widely used open-source DRAM simulators, assessing their features, analysis capabilities, and simulation speed.

III. PROPOSED MODEL

This section describes the proposed efficient arbitration technique to handle the multiple refresh requests in multi-processor SoC. We implemented this technique while taking the arbiter architecture into account in order to accelerate the conventional research work. In this effort, our main goal is to give low data transactions priority over other requests made to the core processor. As a result, these low data transactions which need fewer cycles to complete than big data transactions are given first access to off-chip memory.

In order to handle multiple refresh requests, Advanced eXtensible Interface (AXI) Interconnect is used. The proposed architecture for the AXI Interconnect, which includes an arbiter, is shown in Figure 1. Through the AXI Interconnect, the design links four core processors (AXI masters) and off-chip embedded memory (slave). All four core processors may start write and read operations utilizing the AXI4 interface protocol, employing a 32-bit data size and different burst lengths. The four cores may all produce read/write operations and refresh request to/from the off-chip embedded memory simultaneously.

A static fixed priority technique is used by the AXI Interconnect to handle these numerous burst duration requests. We used a static fixed priority approach to allocate priorities to the core processors (masters). For off-chip memory access, the core processor with the fewest data transactions is given priority, whereas the core processor with the most data transactions is given the lowest priority. The main goal of these priorities is to decrease the core processors' total average waiting time.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7214
***Article Received**: 28 April 2023 **Revised:** 14 June 2023 **Accepted:** 30 June 2023*

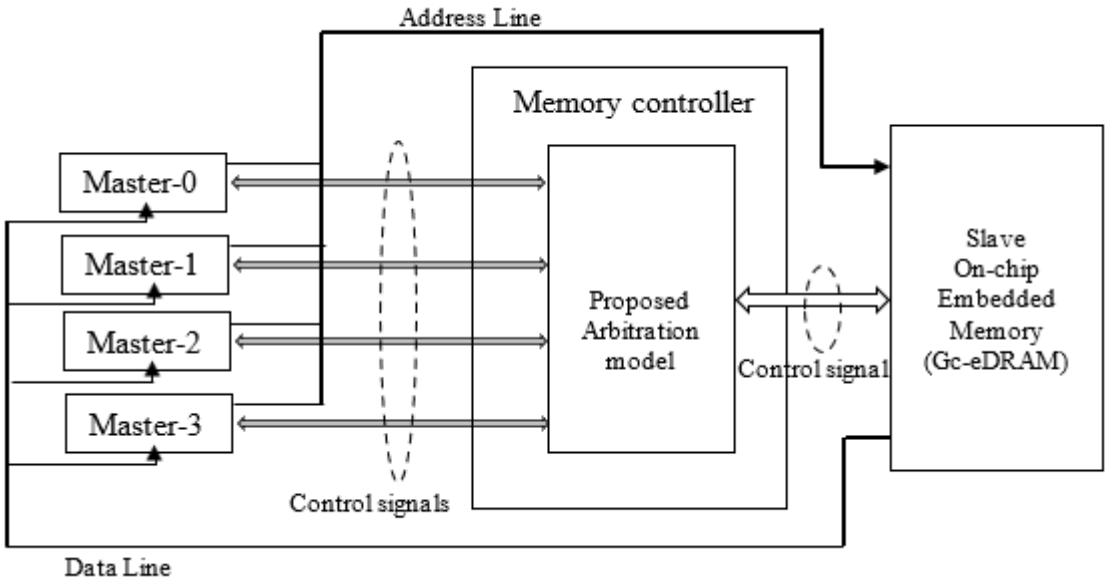

Figure 1: Proposed Model

3.1 Static Fixed Priority Technique

This section explores and presents the static fixed priority algorithm -based finite state machine (FSM) of the AXI Arbiter. When there are no requests from any master to access the common interface and start a transaction with the AXI arbiter, the initial state, also known as the idle state, takes place. Any core processor (master) may request access to the shared interface and start the transaction when the clock begins and the reset signal rises. The state returns to the idle state after the final core processor's transactions are complete, as depicted in Figure 2 below.

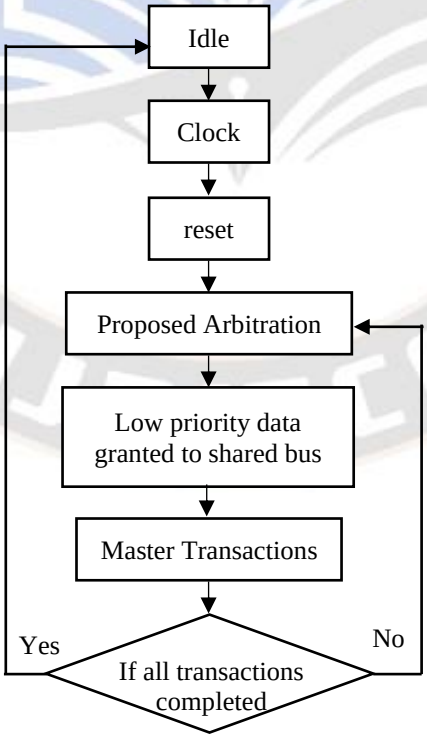

Figure 2: Static fixed priority algorithm -based FSM of the AXI Arbiter.

The arbiter gives the core processor with the greatest scheduling priority if many core processors concurrently transmit requests to use the shared interface and begin their transactions. This decision is made based on the static fixed

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7214
***Article Received**: 28 April 2023 **Revised:** 14 June 2023 **Accepted:** 30 June 2023*

priority method. The amount of data transactions the core processor must complete determines the priority; the core processor with fewer data transactions is given greater priority. The chosen core processor may start its transaction as soon as it gets the shared interface. The second core processor, which has less data to send, is then given access to the shared interface by the arbiter once the first core processor has finished its transaction. Using the static fixed priority mechanism, the arbiter keeps supplying the shared interface to the other core processors until each one has finished processing its transactions.

The simulation results from the proposed model are thoroughly explained in this section. The proposed model's Figure 1 shows how the AXI connection is used to create a connection between the off-chip memory (shown as the AXI Slave) and the four core processors (shown as the four AXI Masters). This research primarily focuses on the Arbiter, which is the primary element of the AXI interconnect. The arbiter gets requests from the multi-core processors and chooses how much off-chip memory to access depending on how long each data transaction takes.

The AXI4 interface protocol, which enables different burst lengths and a data size of 32 bits, is used to begin the transactions by the four core processors. The Vivado tool and the FPGA ZYNQ-7 ZC702 Evaluation Board were both used to simulate and synthesize the proposed design. The SoC performance of the recommended arbiter architecture in this research is improved, and the average waiting time in the ongoing project is reduced, by using the AXI4 interface protocol and a static fixed-priority scheduling method.

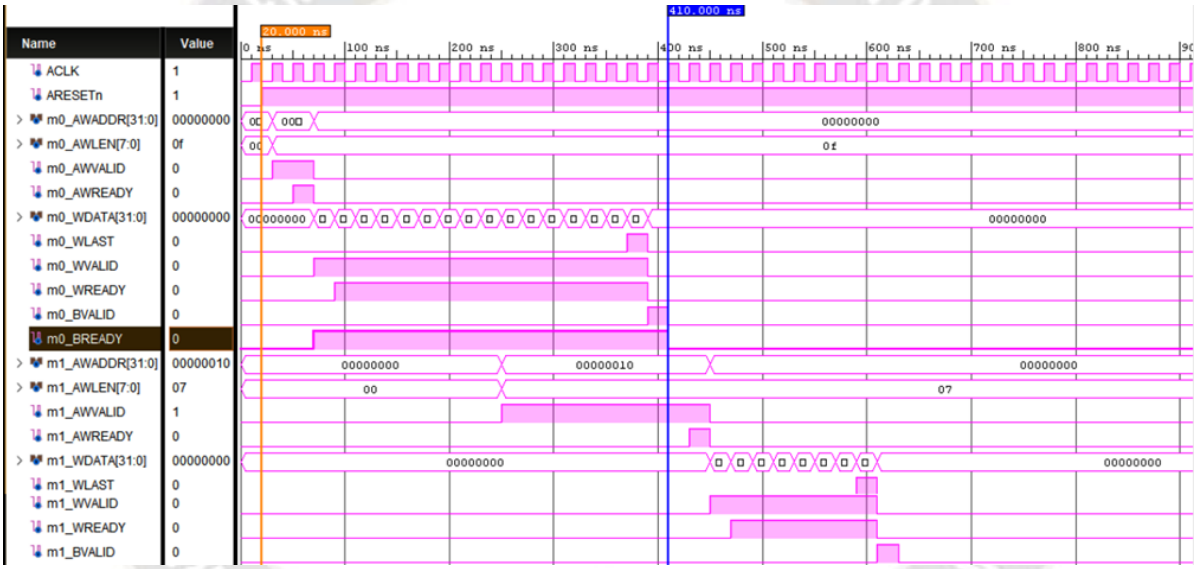

Figure 3: Without the arbitration, characterize the 1st and 2nd core processors' write ansaction channel signals.

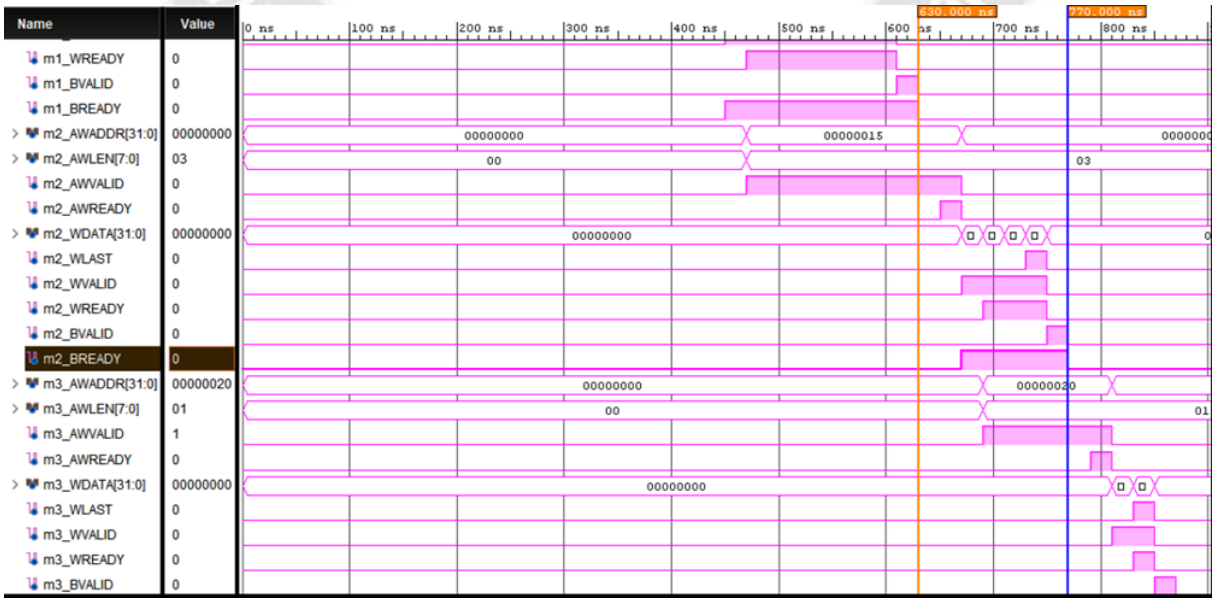

Figure 4: Without the arbitration, characterize the 3rd and 4th core processors' write transaction channel signals

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7214
***Article Received**: 28 April 2023 **Revised:** 14 June 2023 **Accepted:** 30 June 2023*

The conventional method [8] included a suggestion for a multi-core memory controller that might be used in a computer system that had four core processors. However, it did not take into account the arbiter architecture and instead used the AXI4-Lite interface protocol. Because of the way that architecture worked, the memory was always accessible by the first core CPU. This strategy was successful due to the fact that each core processor had a single burst length and needed the same amount of clock cycles to finish processing the transaction. However, an issue occurs when implementing the AXI4 full map and enabling each core processor to finish transactions with various burst lengths without taking into consideration the arbiter architecture. This creates a bottleneck in the processing of transactions. This problem is shown by the simulation results, which are shown in figures 3 and 4 respectively.

When utilizing the AXI4 full map interface protocol, you have access to a total of five standard channels that may be used for either reading or writing data. The implementation of the write transaction channels, including all of the required handshaking signals, for the core processors accessing external memory was the primary emphasis of this part of the simulation results. The signals that are sent via the write transaction channel for the first and second core processors are seen in Figure 3. According to the findings of the simulation that were shown previously in this paragraph, the data size is 32 bits, and the burst lengths are 16, with a respective 8, each time. The signals for the write transaction channel are shown in Figure 4 for the third and fourth core processors. The data size of these signals is 32 bits, and their burst lengths are variable, coming in at either 4 or 2, respectively.

According to the data shown in figure 3, the first core processor started the transmission of write signals 20 nanoseconds after the high level of the clock signal and simultaneously with the high state of the reset signal. After a total of 410 nanoseconds had passed, the m0_BVALID and m0_BREADY signals had both reached their maximum levels. This indicates that the activity of the first core processor with 16 burst lengths has been completed, and it provides the second core processor the permission to access the shared interface in order to get data from off-chip memory.

In addition, it suggests that the second core processor won't start its transaction for another 410 nanoseconds after the first one has finished. Figure 4 depicts a situation very similar to the one shown for the second and first processors. In this case, the third and fourth processors wait until 630 and 770 nanoseconds, respectively, before beginning their transactions. This waiting time is imposed on each and every core processor, which results in a delay of 457.5 nanoseconds on average.

The research presented a model that made use of an arbitrator architecture and used an approach that was characterized by static fixed-priority scheduling. This model resulted in an improvement in the average waiting time while the core processors carried out transactions using the AXI4 interface protocol with a variety of burst lengths. During the programming phase, the technique for prioritization is shown in figures 5, 6, and 7 respectively. To be more specific, we gave a higher priority to the core processor that had a smaller number of data transactions (a shorter burst length), while we gave a lower priority to the core processor that had a larger number of data transactions (a longer burst length). The goal of this prioritization was to cut down on the total amount of time spent waiting across all core processors, which led to an overall reduction in the average amount of time spent waiting.

The data shown in figure 5 indicates that the arbitrator gives authorization to the fourth core processor to use the common interface and access the off-chip memory before any of the other cores. After a delay of 20 nanoseconds, the fourth core processor started the transmission of write transaction signals when the clock signal and the reset signal were both high at the same time. At the time interval of 150 ns, the m3_BVALID and m3_BREADY signals are both high, indicating that the operation of the fourth core processor with two burst lengths has been completed. As a direct consequence of this, the third core processor may now make use of the shared interface and get access to the memory that is located off-chip. This is because the burst length of the processor with the third core is much shorter than that of the processors with the first and second cores. It also suggests that the transaction on the third core processor will not commence for another 150 nanoseconds after that point in time, which is the implication of the previous statement. Before beginning their respective transactions in the identical circumstance, as shown in figures 6 and 7, the second core and the first processor will wait until 410 ns and 710 ns, respectively. This can be seen in both of these figures. The amount of time spent waiting equals to 322.5 nanoseconds across all core processors taken into account.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7214
***Article Received**: 28 April 2023 **Revised:** 14 June 2023 **Accepted:** 30 June 2023*

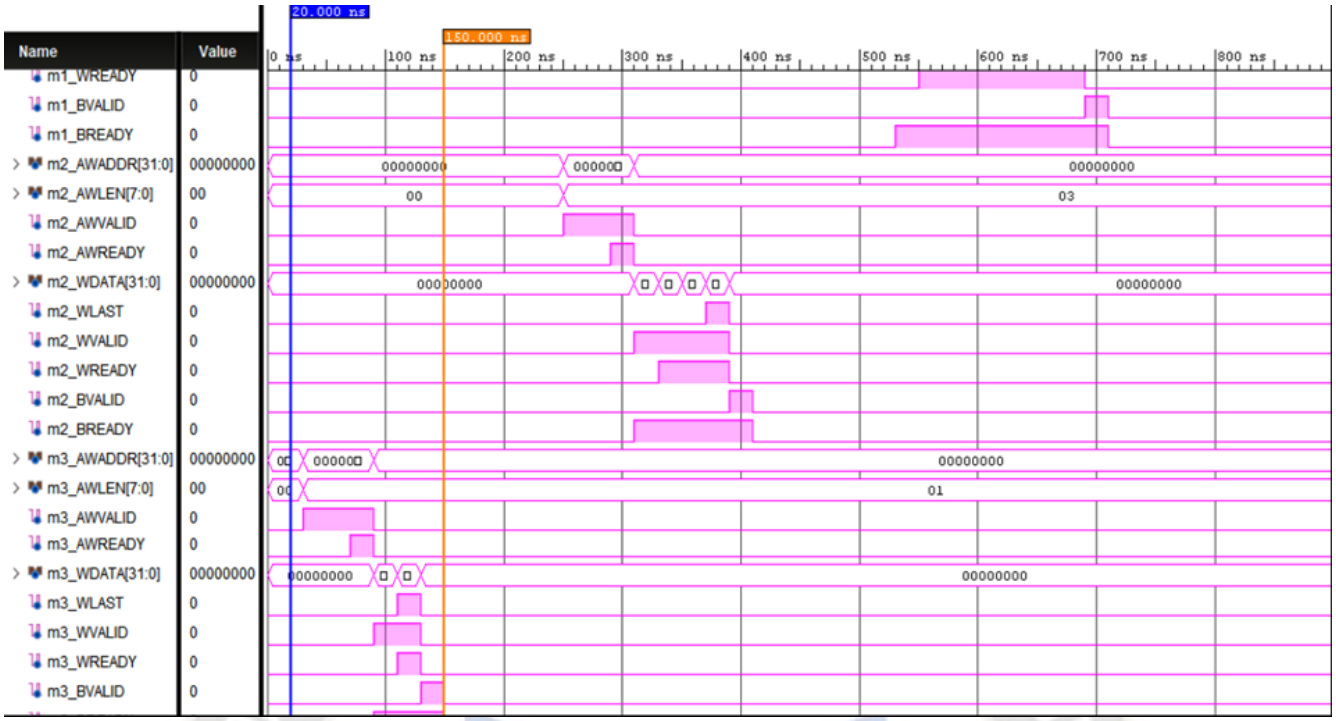

Figure 5: With arbitration, characterize the 1st and 2nd core processors' write transaction channel signals.

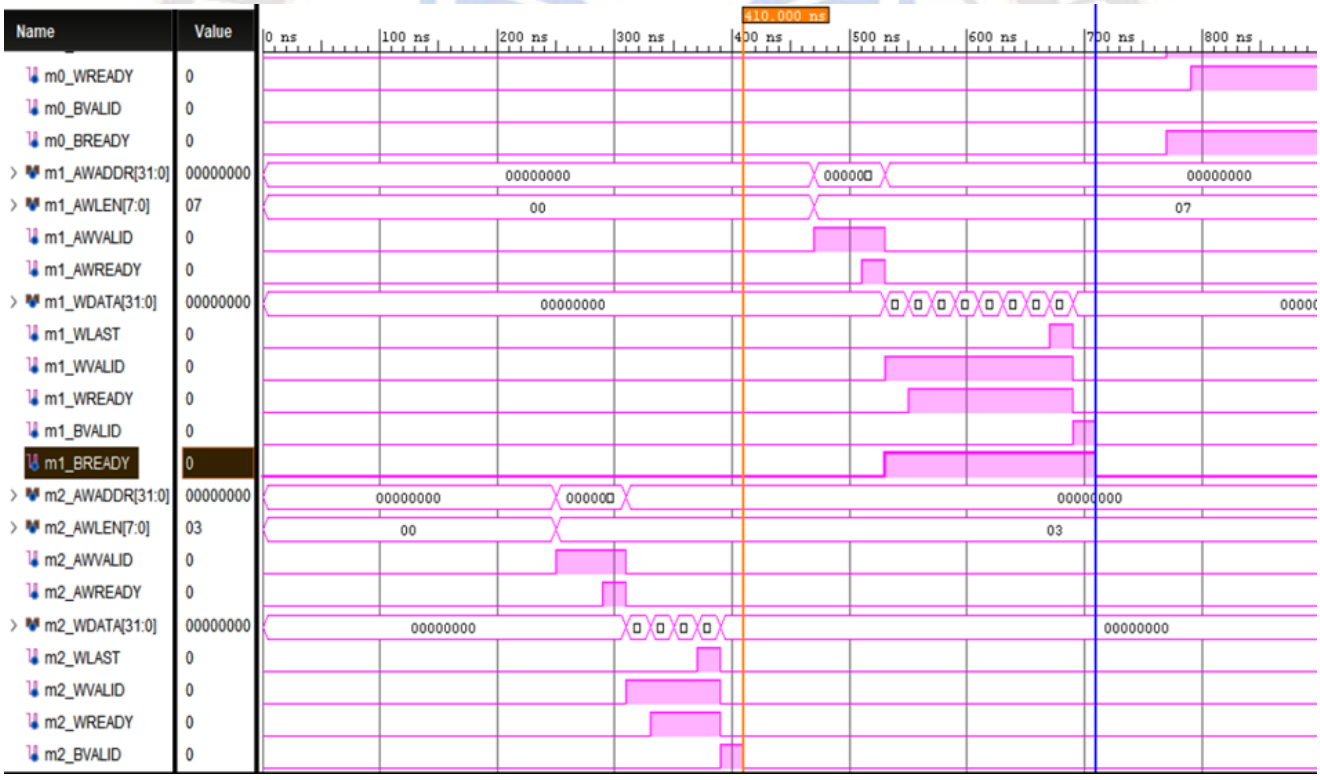

Figure 6: With arbitration, characterize the 2nd and 3rd core processors' write transaction channel signals.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7214
***Article Received**: 28 April 2023 **Revised:** 14 June 2023 **Accepted:** 30 June 2023*

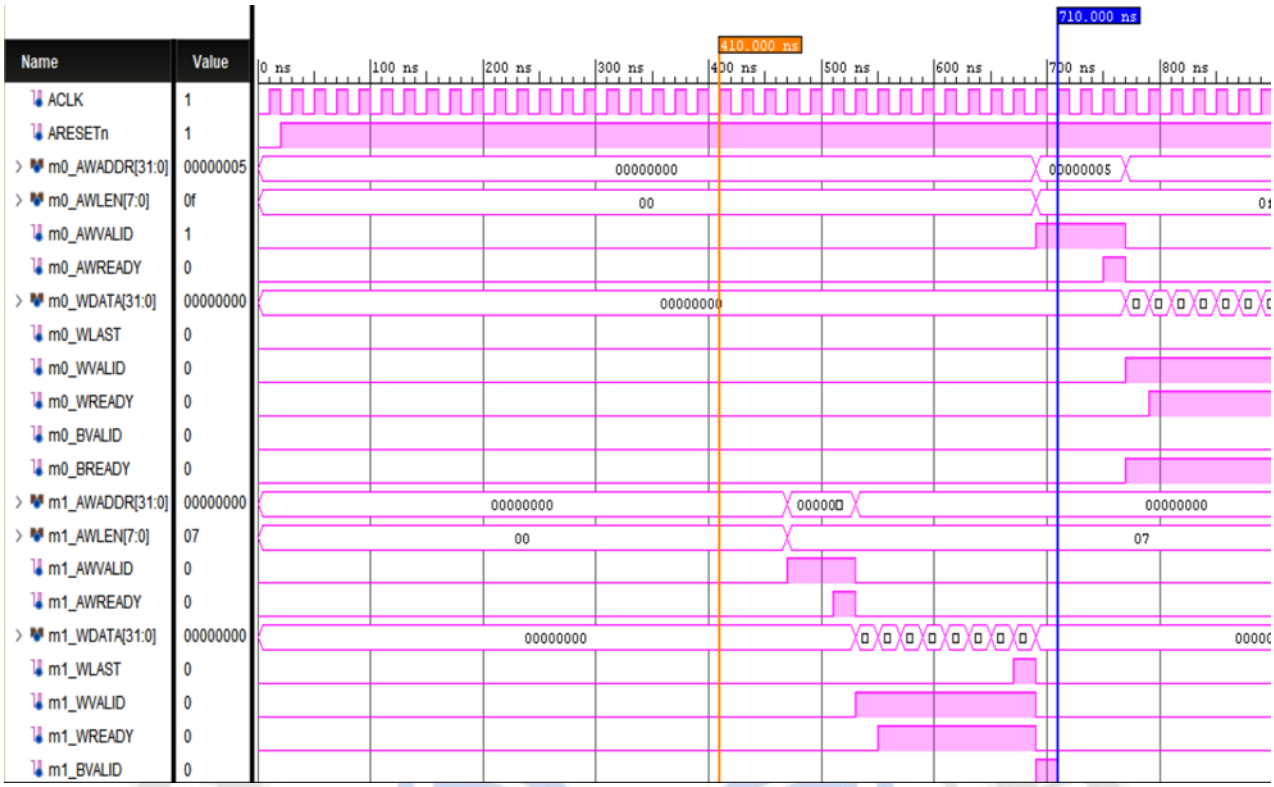

Figure 7: With arbitration, characterize the 3rd and 4th core processors' write transaction channel signals.

Incorporating the proposed model, which takes into account the arbiter architecture and uses the static fixed-priority scheduling strategy, resulted to a considerable improvement of 34.6% in the overall waiting time for all core processors, as evidenced by the results of the simulations. Additional simulation results for the four core processors, including a variety of burst lengths, have been immediately included into Table 1 and Table 2.

Additional simulation findings are shown in Table 1 and demonstrate the interaction that takes place between four core processors, which are represented as Masters, and off-chip memory, which is represented as Slaves. These findings provide a comparison between situations using and not involving the arbiter architecture. According to Table 1, the four core processors access the off-chip memory in bursts of varied lengths. The first, second, third, and fourth core processors access the memory for 32, 16, 8, and 2 clock cycles, respectively. There is a row labelled "waiting time" in the table, and underneath it is two sub-rows for analyzing the data. Without taking into account the arbiter architecture, the first sub-row displays the amount of time that is required for each core processor to start a transaction. Waiting takes up an average of 777.5 nanoseconds across all cores of the CPU. When taking the arbiter architecture into consideration, the waiting time for each core processor is represented by the second sub-row. Before beginning the transaction, each core processor pauses for an average of 362.5 nanoseconds to gather its resources. According to the findings shown in Table 1, the model that was used in this investigation was successful in achieving a 72.8 percentage point increase in overall quality.

In addition, the simulation results for waiting time for different burst length are shown in Table 2. These results indicate the interaction that occurs between the four core processors, which are referred to as Masters, and the off-chip memory, which is represented as Slaves, both when the arbiter architecture is taken into consideration and when it is not. The table shows that each of the four core processors accesses the off-chip memory using a distinct burst length to begin transactions. The burst lengths are as follows: 64, 32, 16, and 4 for the first, second, third, and fourth core processors, respectively. In the row labelled "waiting time," there are two sub-rows that might be used. When the arbiter architecture is not taken into consideration, the first sub-row displays the amount of time that must be spent waiting for each core processor before beginning the transaction. Waiting time for each of the four core processors is 1277.5 nanoseconds on average. The waiting time is shown in the second sub-row when the arbiter architecture is taken into consideration. Before beginning the transaction, the average amount of time that each core processor must wait is 397.5 nanoseconds. According to the findings that are shown in Table 2, the overall proposed model for this investigation displays a considerable increase of 105% in comparison to models that were used in earlier studies.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7214
Article Received: 28 April 2023 Revised: 14 June 2023 Accepted: 30 June 2023

Table 1: Burst length vs Waiting Time-Test case-1

	Master-3	Master-2	Master-1	Master-0	AWT (Avg. waiting Time)	Percentage
Waiting Time	**1250(ns)**	**1110(ns)**	**730(ns)**	**20(ns)**	**777(ns)**	**72.8%**
	362(ns)	**870(ns)**	**410(ns)**	**150(ns)**	**362(ns)**	
Burst Length	**2**	**8**	**16**	**32**		

Table 2: Burst length vs Waiting Time-Test case-2

	Master-3	Master-2	Master-1	Master-0	AWT (Avg. waiting Time)	Percentage
Waiting Time	**1970(ns)**	**1750(ns)**	**1370(ns)**	**20(ns)**	**1277(ns)**	**105%**
	890(ns)	**490(ns)**	**190(ns)**	**20(ns)**	**397(ns)**	
Burst Length	**4**	**16**	**32**	**64**		

IV. CONCLUSIONS

In conclusion, the arbitration method that we have provided offers a workable and efficient answer to the problem of repeated requests for a refresh in GC-eDRAM. Our method increases the overall performance of the system by using parallelism and optimizing the allocation of resources. This helps to guarantee that upcoming memory technologies are operated in an effective manner. These discoveries provide a contribution to the progression of memory management strategies and have the potential to be of use to a broad variety of computing systems that depend on GC-eDRAM as an essential memory component. The proposed model enhances the existing work by employing the AXI4 interface protocol, which triggers individual core processors to initiate transactions using distinct burst mode capabilities. Additionally, it incorporates an arbiter based on a static fixed priority mechanism. Simulation results, depicted in figures and tables, demonstrate that the proposed model achieves an average waiting time reduction of 34.6%, 72.8%, and 105% respectively. These improvements highlight the efficacy of the proposed model compared to the existing approach, as observed through the simulation results.

REFERENCES

[1]. Hoffmann, Henry, Axel Jantsch, and Nikil D. Dutt. "Embodied self-aware computing systems." Proceedings of the IEEE 108, no. 7 (2020): 1027-1046.

[2]. Uma, V., and Ramalatha Marimuthu. "D-wash–A dynamic workload aware adaptive cache coherance protocol for multi-core processor system." Microelectronics Journal 132 (2023): 105675.

[3]. Goel, Tanmay, Divyansh Maura, Kaustav Goswami, Shirshendu Das, and Dip Sankar Banerjee. "Towards row sensitive DRAM refresh through retention awareness." In 2021 22nd International Symposium on Quality Electronic Design (ISQED), pp. 450-456. IEEE, 2021.

[4]. Yang, Jia-Qin, Ye Zhou, and Su-Ting Han. "Functional applications of future data storage devices." Advanced Electronic Materials 7, no. 5 (2021): 2001181.

[5]. Brand, Marcel, Michael Witterauf, Éricles Sousa, Alexandru Tanase, Frank Hannig, and Jürgen Teich. "*-Predictable MPSoC execution of real-time control applications using invasive computing." Concurrency and Computation: Practice and Experience 33, no. 14 (2021): e5149.

[6]. Suresh, K. S. ., & Kamalakannan, T. . (2023). Digital Image Steganography in the Spatial Domain Using Block-Chain Technology to Provide Double-Layered Protection to Confidential Data Without Transferring the Stego-Object. International Journal of Intelligent Systems and Applications in Engineering, 11(2s), 61–68. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2508

[7]. Garzón, Esteban, Yosi Greenblatt, Odem Harel, Marco Lanuzza, and Adam Teman. "Gain-cell embedded DRAM under cryogenic operation—A first study." IEEE Transactions on Very Large-Scale Integration (VLSI) Systems 29, no. 7 (2021): 1319-1324.

[8]. Shrivastava, Anurag, and Sudhir Kumar Sharma. "Various arbitration algorithm for on-chip (AMBA) shared bus multi-processor SoC." In 2016 IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pp. 1-7. IEEE, 2016.

[9]. Noy, Tzachi, and Adam Teman. "Design of a refresh-controller for GC-eDRAM based FIFOs." IEEE Transactions on Circuits and Systems I: Regular Papers 67, no. 12 (2020): 4804-4817.

[10]. Narinx, Jonathan, Robert Giterman, Andrea Bonetti, Nicolas Frigerio, Cosimo Aprile, Andreas Burg, and Yusuf Leblebici. "A 24 kb single-well mixed 3T gain-cell eDRAM with body-bias in 28 nm FD-SOI for refresh-free DSP applications." In 2019 IEEE Asian Solid-State Circuits Conference (A-SSCC), pp. 219-222. IEEE, 2019.

[11]. Widmer, Marco, Andrea Bonetti, and Andreas Burg. "FPGA-based emulation of embedded DRAMs for statistical error resilience evaluation of approximate computing systems." In Proceedings of the 56th Annual Design Automation Conference 2019, pp. 1-6. 2019.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7214
***Article Received**: 28 April 2023 **Revised:** 14 June 2023 **Accepted:** 30 June 2023*

[12]. Seyedzadeh Sany, Bahareh, and Behzad Ebrahimi. "A 1-GHz GC-eDRAM in 7-nm FinFET with static retention time at 700 mV for ultra-low power on-chip memory applications." International Journal of Circuit Theory and Applications 50, no. 2 (2022): 417-426.

[13]. Lee, Changmin, Wonjae Shin, Dae Jeong Kim, Yongjun Yu, Sung-Joon Kim, Taekyeong Ko, Deokho Seo et al. "Nvdimm-c: A byte-addressable non-volatile memory module for compatibility with standard ddr memory interfaces." In 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 502-514. IEEE, 2020.

[14]. Cui, Jinhua, Youtao Zhang, Liang Shi, Chun Jason Xue, Jun Yang, Weiguang Liu, and Laurence T. Yang. "Leveraging partial-refresh for performance and lifetime improvement of 3D NAND flash memory in cyber-physical systems." Journal of Systems Architecture 103 (2020): 101685.

[15]. Brian Moore, Peter Thomas, Giovanni Rossi, Anna Kowalska, Manuel López. Machine Learning for Fraud Detection and Decision Making in Financial Systems. Kuwait Journal of Machine Learning, 2(4). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/216

[16]. Steiner, Lukas, Matthias Jung, Felipe S. Prado, Kirill Bykov, and Norbert Wehn. "DRAMSys4. 0: a fast and cycle-accurate systemC/TLM-based DRAM simulator." In Embedded Computer Systems: Architectures, Modeling, and Simulation: 20th International Conference, SAMOS 2020, Samos, Greece, July 5–9, 2020, Proceedings 20, pp. 110-126. Springer International Publishing, 2020.