ISSN 2278-3091
Volume 9, No.1, January – February 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse99912020.pdf
https://doi.org/10.30534/ijatcse/2020/99912020

Integration of Gem5 And Dramsim2 For DDR4 Simulation

P. Karunamurthy[1], S.S.N Alhady[2*], A.A.A Wahab[3], W.A.F.W Othman[4]
[1]School of Electrical & Electronic Engineering, Universiti Sains Malaysia, Malaysia, prithadevi.karunamurthy@intel.com
[2*] School of Electrical & Electronic Engineering, Universiti Sains Malaysia, Malaysia, sahal@usm.my
[3] School of Electrical & Electronic Engineering, Universiti Sains Malaysia, Malaysia, aeizaal@usm.my
[4] School of Electrical & Electronic Engineering, Universiti Sains Malaysia, Malaysia, wafw_othman@usm.my

ABSTRACT

Simulators have drawbacks due to the total time taken for the simulation. Therefore, two simulators should be integrated to produce a robust simulator to overcome this hurdle. The main aim of this research which is to integrate two open-source simulators to study the simulation of DDR4 is achieved. GEM5 and DRAMSim2 are integrated for DDR4 simulation using ISA x86. DRAM Controller codes and codes to access the bank or bank groups in DDR4 are modified for DDR4. The parameters of DDR4 – 24000 passed into the simulation. Based on the simulation results, GEM5 DRAMSim2 has verified its correctness and legal to be used for DDR4 simulations. The modified DRAM Controller codes for DDR4 from DDR3 is proven working when 100% pass. GEM5 DRAMSim2 is 99.2% faster than previous work done with GEM5 – NVMain. Moreover, GEM5 DRAMSim2 used only 23% power from the overall power to perform ACT/PRE activities during the execution of 20 000 instructions. Furthermore, simulation of DDR4 using GEM5 DRAMSim2 used 40% less background power compared to previous GEM5 – NVMain work. The performance of DDR4 using GEM5 DRAMSim2 is fast because the correlation between the average bandwidth and average latency is 0.9975. This research proved that the integrated GEM5 DRAMSim2 is an effective and efficient simulator for DRAM simulations.

Key words: computer system simulator, DRAMSim2, DDR4, Gem5, memory system simulator

1.INTRODUCTION

The need for larger data storage is growing rapidly. Therefore, advancing in-memory performance is very crucial in these days. To serve this purpose DDR is being improved in terms of performance and execution time from one generation to another [1]. However, DDR research in real life would cost a large amount of money and would be tedious, hence open-source simulator would be a better alternative.

Nevertheless, memory system simulators are encountering some drawbacks due to the expanded memory trend. For instance, exploring memory system performance by simulations consumes ample time compared to the execution in a real system due to the complexity of memory architecture.

This is a great disadvantage for memory technology to be advanced in the fast-moving world. To overcome this hurdle, two simulators must be combined to achieve better efficiency thus reducing the time taken for memory simulation.

Therefore, in this research, two open-source simulators have been integrated to explore DDR behavior and performance. Hence, the ultimate goal of this research is to integrate GEM5 and DRAMSim2 for DDR4 simulation. GEM5 is a simulator used for evaluating performance and analysis for computer architecture [2] whereas DRAMSim2 is a simulator dedicated to memory system simulator. Up-to-date, there is no integration of the GEM5 DRAMSim2 simulation for DDR4. Since DDR4 is the latest and widely used high-speed memory in many applications [3], it would be reasonable to explore its architecture for betterment in this research.

2. METHODOLOGY

In this research, the methodology is divided into four main phases. Figure 1 summarizes the research flow implemented in a flowchart form for better understanding. The first phase focuses on building GEM5 and running the source code for computer system architecture. In parallel, building and running DRAMSim2 for the memory system model is also done. The second phase is the integration of GEM5 and DRAMSim2 to be one powerful simulator. The third phase is divided into two parts. The first part focuses on modifying DRAM Controller codes. It is done by modifying the existing codes of the DDR3 controller in DRAMSim2 such that it is valid for the DDR4 controller. Then DDR4 parameters are taken from the vendor's datasheet to create a new device.ini file. The second part of the third phase is DDR4 simulation. In the final phase, the evaluation of the simulator performance and DDR4 performance is done. The evaluation is compared with other simulators from previous work to benchmark this research.

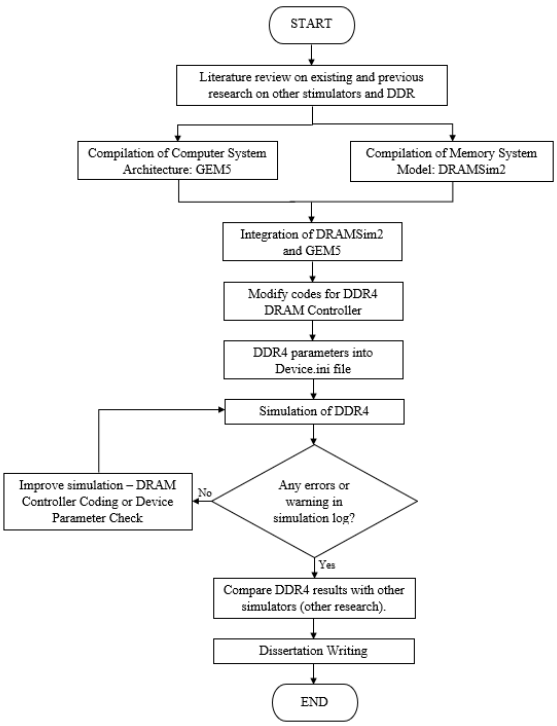

Figure 1: Summary of research flow.

The dependencies of GEM5 are downloaded and installed in Ubuntu using the Sudo command, sudo apt install build-essential git m4 scons zlib1g zlib1g-dev libprotobuf-dev protobuf-compiler libprotoc-dev libgoogle-perftools-dev python-dev python

The main source code of GEM5 is downloaded from the active support group, https://gem5.googlesource.com/public/gem5 using git command. Next, the GEM5 source code is downloaded and installed. Finally, the GEM5 is compiled using x86 ISA. The command line to build GEM5 using SCons is scons build/X86/gem5.opt -j3. GEM5 build is a success when a directory called build is automatically created after the Scons command.

Once the GEM5 is built, a configuration file is created to run the GEM5. The configuration file is a python script in which a system to simulate is created along with the system's components and all the parameters are specified. The script is user-defined and almost all options on the command lines are allowed to be defined by the user. The script is used to execute the simulation by the following command build/X86/gem5.opt configs/example/se.py.

The DRAMSim2 source is downloaded from https://github.com/umd-memsys/DRAMSim2 using git clone. DRAMSim2 is built by using the command make. Then, on the gzipped traces, the preprocessor is executed, ./traceParse.py k6_aoe_02_short.trc.gz. This is ensured that the DRAMSim2 can be simulated successfully as a standalone simulator.

Next, the trace-based simulator is executed in the main DRAMSim2 directory using ./DRAMSim -t traces/k6_aoe_02_short.trc -s system.ini -d ini/DDR3_micron_64M_8B_x4_sg15.ini -c 1000. This command will run 1 000 simulations of the existing k6_aoe_02_short trace using the DDR3 part. Since DRAMSim2 supports up to DDR3 up-to-date, any DDR3.ini file in the /ini folder can be used to ensure the DRAMSim2 built successfully.

The second phase of methodology in this dissertation is to integrate DRAMSim2 into GEM5 and to ensure the integration is a success. The entire DRAMSim2 is copied in /gem5/ext/dramsim folder. Command-line build/X86/gem5.opt configs/example/se.py --mem-type=DRAMSim2 is used to integrate both the simulators. By executing this command, the DRAMSim2 memory controller model replaces the redundant functions in GEM5 memory controller files. Thus, leaving a lightweight yet sophisticated and powerful memory controller interface. Hence, requests from and to the GEM5 directory memory controller forwarded to DRAMSim2. This creates an organized simulation environment.

The first part of the third phase in this methodology is to develop controller codes. In GEM5 DRAMSim2, the existing DRAM controller supports up to DDR3 controller only. Any newer controller, such as DDR4 is not openly available. The main difference in DDR3 and DDR4 is that DDR3 has no bank groups, only banks whereas DDR4 has bank groups that consist of banks. Hence, the flow to access banks in DDR3 and DDR4 is different.

Figure 2 is a flowchart that simplifies the controller code to differentiate DDR3 and DDR4. To differentiate DDR3 and DDR4 the bank selection is done. The codes in DRAM Controller to identify the bank is edited for this. Since DDR3 has only banks, the banks' addresses (BA) are three bits, BA0, BA1, and BA2. For instance, BA2BA1BA0: 101 chooses bank 6. Whereas in DDR4, the banks' addresses are four bits BG0, BG1, BA0, and BA1. BG0BG1 denotes the bank group (BG) and, BA0and BA1 denote the bank address (BA). For instance, BG1BG0BA1BA0 0110 chooses bank 7 from bank group 1.

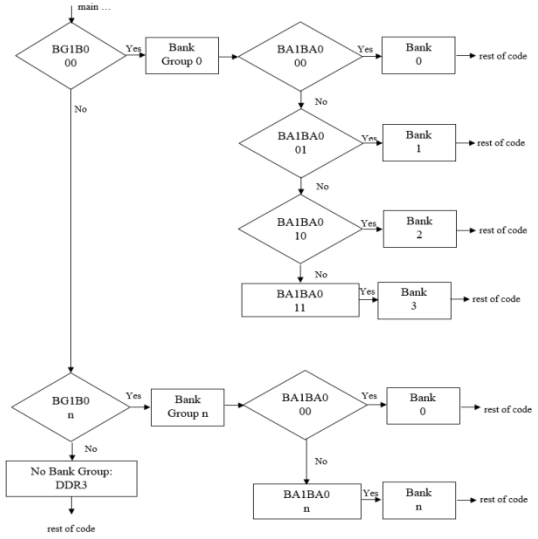

Figure 2: DRAM Controller code flowchart to identify bank.

In DDR3, banks will be accessed individually whilst in DDR4, the bank groups will be accessed first before accessing the banks. Therefore, DRAM Controller codes in GEM5 is edited such that the read/write or any other commands queue up and executed accordingly either in the same bank group or in other bank groups in DDR4. The DRAM Controller codes in GEM5 can be found in */gem5/src/mem/dram_ctrl.cc*. Figure 3 simplifies the DRAM controller codes flow for DDR4 bank access timings.

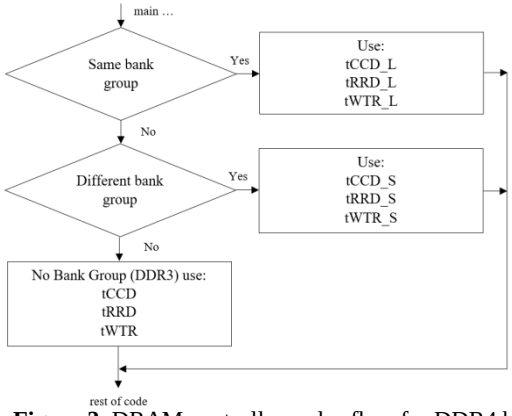

Figure 3: DRAM controller codes flow for DDR4 bank access timings.

The first part of the third phase is to give the DDR4 parameters into DRAMSim2. The parameters of DDR4 can be found in any DRAM vendors specification data. The DRAM used in this dissertation is Micron MT40A2G4, a single DDR4 – 2400 (16 x 4 configuration). The existing device.ini of DDR3 is edited with the DDR4 parameter. Then the device.ini is declared in */gem5/src/mem/Dramsim2.py.*

Before the DDR4 simulation is done, the command ***scons build/X86/gem5.opt -j3*** is executed again to build GEM5 again. This is to ensure there is no coding error in GEM5 DRAMSim2 after developing the DRAM Controller codes for DDR4. If the GEM5 build is not successful, the modified codes must be corrected referring to the output error.

Next, the simulation is done by using command, ***build/X86/gem5.opt configs/example/se.py --mem-type=DRAMSim2 --cmd=/home/pk/gem5/tests/test-progs/hello/bin/x86/linux/hello -I 1000000 -n 2***. In general, this command is using memory simulator DRAMSim2 into ISA x86 to execute the test file for 1 000 000 instructions. Figure 3.7 illustrates the codes for DDR4 simulation flowchart in GEM5 DRAMSim2.

The method to evaluate the performance of GEM5 DRAMSim2 is divided into four parts. The first part is to validate the correctness of GEM5 DRAMSim2 for the DDR4 simulation. For this purpose, the simulator is stress-tested with 10M random requests in the ratio of 4:4:2, read:write:ACT/Prefetch/Refreshes. The timestamped log is collected for every half an hour to check on any warning or error during the execution of 10M requests.

The next part is to evaluate the performance of GEM5 DRAMSim2 for DDR4 simulation. To calculate the performance of the GEM5 DRAMSim2 in this part, the total number of instructions simulated (ints) in a random simulation is divided by the total time taken (s). The performance of GEM5 DRAMSim2 will be in terms of inst/s.

To make a benchmark for this research, this research is compared to previous research of DDR4 using GEM5 – NVMain (Farrell, Tsulaia, Dotti, Calafiura and Leggett, 2017). For apple-to-apple comparison with previous work, the same amount and type of instructions, which is random 20 000 instructions in the ratio of 4:4:2 (read:write:ACT/Prefetch/Refreshes) is used. The computer specifications and platform are the same in this research and previous research. This information can be found in Chapter 1.4: Research Scope of this dissertation. The performance of GEM5 DRAMSim2 for 20 000 is calculated and compared to previous work, GEM5 – NVMain.

The third part is to compare the power consumed during the execution of 20 000 instructions in DDR4 using GEM5 DRAMSim2 with GEM5 – NVMain. The simulator power consumption information in this research can be found in the simulation results log. Mainly, three power is given focus, they are background power, burst power, and ACT/PRE power.

Background power indicates the power consumed by the simulator to do overall simulation. Burst power specifies the power used to send the address to the memory and ACT/PRE power is a power used to initiate and perform read and write accesses. Only background power is compared with the previous research. This is because background power consumes the majority power during the simulation.

The fourth and final part is to evaluate DDR4 timing in GEM5 DRAMSim2. This is done by running simulation three times to calculate the average latencies and bandwidth. The latency and bandwidth information can be obtained from the simulation results log. Simple linear regression and correlation are done to study the trend of average latency and average bandwidth of DDR4. Pearson tool is used for this purpose.

The average latency is an independent variable while the average bandwidth depends on the average latency. Average latency influences the trend of bandwidth. The numerical measure of the degree of connection between the two parameters is associated with the correlation coefficient (r). Therefore, the Pearson correlation is used to evaluate the linear relationships between these two parameters. The formula to obtain the coefficient correlation, r [4];

$$r = \frac{\sum(x_i - x_{\text{average}})\,(y_i - y_{\text{average}})}{\sqrt{\sum(x_i - x_{\text{average}})^2 * \sum\,(y_i - y_{\text{average}})^2}} \qquad (1)$$

From equation (1), x being the latency while y being the bandwidth. Theoretically, the average bandwidth must be directly proportional to the average latency for better efficiency. A scatter graph is plotted with a best fit line to

observe the overall direction of the average bandwidth. The more linear the plot is, the faster the timing parameter of DDR4 in simulator. Additionally, the coefficient of determination or R-squared, R^2 value will be discussed.

3.RESULTS AND DISCUSSIONS

To validate the correctness of GEM5 DRAMSim2 as one simulator, the simulator must simulate a stream of memory requests using a valid sequence of DRAM commands. It has to be with respect to the status transitions and the timing parameters of a standard, DDR4 in this dissertation. To fulfill that, the GEM5 DRAMSim2 simulator is stress-tested with a trace that contains 10M of memory requests. Those requests are mainly made of reads and writes and a combination of sequential and random addresses and the minority of the request consists of refreshes, power-downs, and self-refreshes.

Without overflowing the controller's request buffer, this 10M of memory requests are fed into GEM5 DRAMSim2 as quickly as possible. This simulation took about seven hours and no violations were reported in the collected timestamped log of every command issued by GEM5 DRAMSim2. This is an indication that the simulation of DDR4 using GEM5 DRAMSim2 is legal. In spite of this, the simulator gained confidence in its correctness.

The performance of GEM5 DRAMSim2 as a simulator for DDR4 is crucial to be noted in order to find out the efficiency of the simulator. The total time taken for a complete simulation is 6µs for 5 954 total number of instructions (read and write. Hence, the performance of GEM5 DRAMSim2 for DDR4 simulation is 992 333 333.3 inst/s.

Comparing with the previous research for DDR4 simulation using GEM5 – NVMain [5], DDR4 simulation takes about 0.006s for almost 20 000 instructions of read and write. On the other hand, the simulated time in this dissertation is 6µs for 5 954 instructions of read and write. An unbiased comparison is made by multiplying the simulated instructions by 3.36 times so that 20 000 instructions are fed into GEM5 DRAMSim2 to find out the total time taken for a complete simulation. The GEM5 DRAMSim2 took about 0.00005s to execute 20 000 instructions.

All the comparison information of GEM5 DRAMSim2 versus GEM5 – NVMain are tabulated in Table 1 for a clearer picture. From Table 1, it is clearly seen that the performance of GEM5 DRAMSim2 is better than GEM5 – NVMain because GEM5 DRAMSim2 takes lesser time to execute 20 000 instructions. Simply put, the performance of GEM5 DRAMSim2 is better than GEM5 – NVMain.

Table 1: GEM DRAMSim2 versus GEM5 – NVMain for total time taken for execution and performance of simulator.

	GEM5 DRAMSim2	GEM5 – NVMain
Instructions fed (read: write, 9:1) (inst)	20 000	20 000
Total time taken to execute (s)	0.00005	0.006
Performance of simulator (inst/s) (x 10 ^ 6)	400	3

From the simulation results, the rate at which works are performed by the GEM5 DRAMSim2 is tabulated in Table 2. The background power consumed by GEM5 DRAMSim2 to simulate DDR4 is 0.03W, the burst power is 0.004W and the Active/Precharge (ACT/PRE) power is 0.01W. A pie – chart, Figure 2, is plotted to have a clearer picture of the power consumed by GEM5 DRAMSim2 for DDR4 simulation.

Table 2: Power consumption by GEM5 DRAMSim2 for DDR4 simulation.

Power	W	%
Background	0.03	68
Burst	0.004	9
ACT/PRE	0.01	23

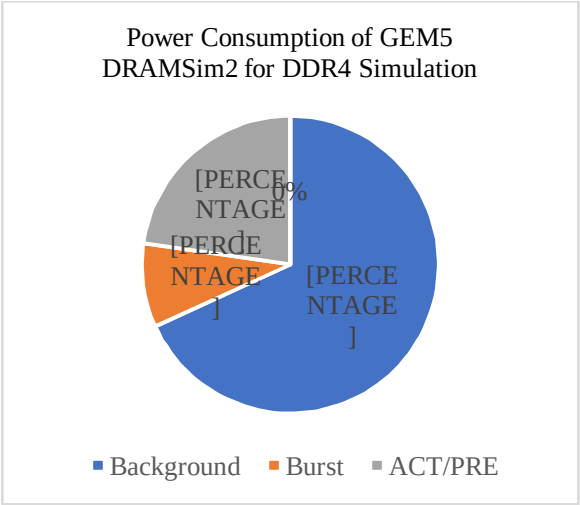

Figure 4: A pie – chart to display the energies consumed by GEM5 DRAMSim2 for DDR4 simulation activity.

From Figure 4, it is seen that most of power consumption is majorly used in the background whereas least power is used for burst and ACT/PRE activities. This shows GEM5 DRAMSim2 does not require much power to perform ACT/PRE activities. Hence, the read access and write access in GEM5 DRAMSim2 needs lesser power to be performed.

Due to majority power is consumed in background and background power determines the performance of simulator, only the background power consumption is compared with previous research. By using GEM5 DRAMSim2 for DDR4 simulation, the total background power consumed is only 0.03W whereas GEM5 – NVMain used about 0.05W. Figure 5 depicts a power graph to differentiate the power consumption by GEM5 DRAM and GEM5 – NVMain. Simply put, simulation of DDR4 using GEM5 DRAMSim2 uses 40% less background power compared to using GEM5 – NVMain. This

means DDR4 simulation using GEM5 DRAMSim2 is more efficient than using GEM5 – NVMain.

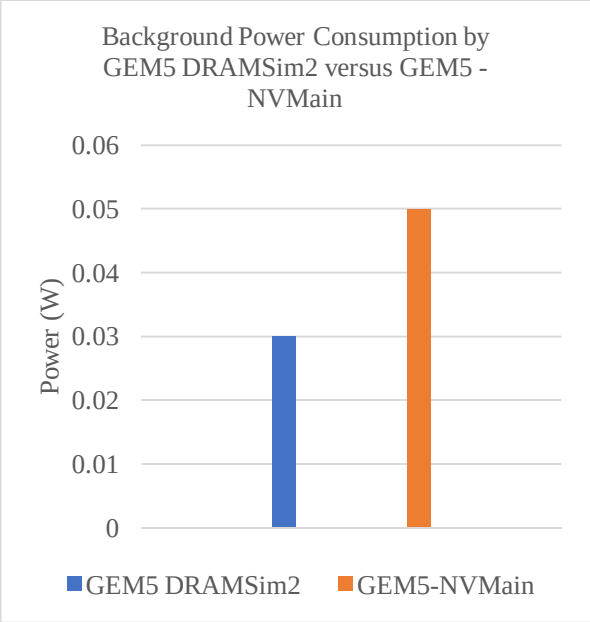

Figure 5: Background power consumption by GEM5 DRAMSim2 versus GEM5 – NVMain.

Table 3 explains the correlation between the average latency and average bandwidth of DDR4 simulated is a positive 0.9975 coefficient. As the average latency increases, the average bandwidth increases proportionally. A scattered graph, Figure 6, of average bandwidth versus average latency is plotted with best fit line to observe the trend of average bandwidth depending on average latency.

The best fit line (orange-dashed line) is plotted using the least-square method to find the best fit for a line though data points. R2 value, 0.9951 closer to the value of 1, indicates that the best fit is 99.51% and it is a good fit of the regression analysis model. This conveys that appropriate time is used in GEM5 DRAMSim2 for DDR4 simulation to initiate a request for a byte in memory until it retrieved by processor and the rate to process it is very quick.

Table 3: Average latency and average bandwidth correlation table.

	Average Latency (ns)	Average Bandwidth (MB/s)
Average Latency (ns)	1	
Average Bandwidth (MB/s)	0.9975	1

The summary of the regression output is tabulated in Table 4. The standard error of 0.2364 (23.64%), shows the precision of the regression analysis – the smaller the number, the more certain is the regression equation. The larger value of F statistics, 2 839, explains the significant relationship between the average latency and average bandwidth. The Significance F or p-value of 1.4356 x 10^-17 which is less than 0.05 (5%) suggests that the results are very reliable.

Table 4: Regression summary output

Regression Statistics	
Coefficient correlation, r	0.9975
Coefficient of determination, R^2	0.9951
Standard Error	0.2364
F statistics	2839.7513
Significance F, p-value	1.4356 x 10^-17

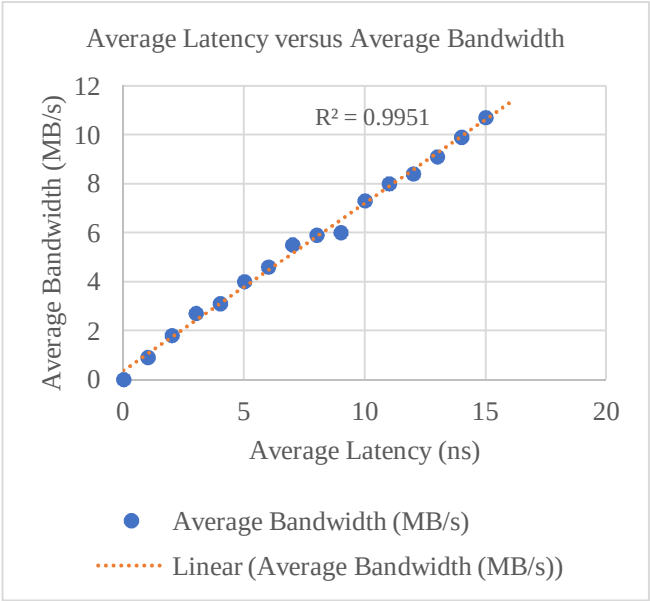

Figure 6:: Graph of average latency versus average bandwidth.

4.CONCLUSIONS AND DISCUSSIONS

In conclusion, the main aim of this research which is to integrate two open-source simulators to study the simulation of DDR4 is achieved. The three objectives of this research are also achieved. First and foremost, the integration of GEM5 and DRAMSim2 as one simulator is accomplished without any violations reported in the collected timestamped log after sending 10M requests. This proves that there is 0% error in the integration of these two simulators and simulation for DDR4 is legal. Secondly, the existing DRAM Controller codes for DDR3 in DRAMSim2 is modified for DDR4 is also accomplished in this research. This is proven when 100% pass was seen when DDR4 parameters are taken into the simulation. Finally, the third objective which is to identify the performance of the GEM5 DRAMSim2 simulator for DDR4 and simultaneously to analyze DDR4 timing in GEM5 DRAMSim2 is also achieved. GEM5 DRAMSim2 executed 20 000 instructions in 0.00005s only, which translates to the performance of GEM5 DRAMSim2 is 99.2% faster than previous work done with GEM5 – NVMain. Moreover, GEM5 DRAMSim2 used only 23% power from the overall power to perform ACT/PRE activities during the execution of 20 000 instructions. Furthermore, simulation of DDR4 using GEM5 DRAMSim2 used 40% less background power compared to

previous GEM5 – NVMain work. The performance of DDR4 using GEM5 DRAMSim2 is fast because the correlation between the average bandwidth and average latency is 0.9975. This research proved that the integrated GEM5 DRAMSim2 is an effective and efficient simulator for DRAM simulations.

REFERENCES

[1] Huang, N. K., Hsieh, C. Y., Tseng, B. C., & Shih, L. Y. (2018). **Comprehensive signal and power co-investigation on DDR4 simulation and measurement**. In *2018 IEEE International Symposium on Electromagnetic Compatibility and 2018 IEEE Asia-Pacific Symposium on Electromagnetic Compatibility (EMC/APEMC)*, p. 1041-1044. https://doi.org/10.1109/ISEMC.2018.8393943

[2] Abudaqa, A. A., Al-Kharoubi, T. M., Mudawar, M. F., & Kobilica, A. (2018) **Simulation of ARM and x86 microprocessors using in-order and out-of-order CPU models with Gem5 simulator**, *5th International Conference on Electrical and Electronic Engineering (ICEEE)*, p.317-332. doi: 10.1109/ICEEE2.2018.8391354

[3] Schmitz, T. (2015). **The rise of serial memory and the future of DDR**. *WP456*.

[4] Boughorbel, S., Jarray, F., & El-Anbari, M. (2017). **Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric**. *PloS one*, *12*(6), e0177678.

[5] Farrell, S., Tsulaia, V., Dotti, A., Calafiura, P., & Leggett, C. (2017). **Multi-threaded ATLAS simulation on Intel Knights Landing processors**. In *J. Phys. Conf. Ser.* https://doi.org/10.1088/1742-6596/898/4/042012