electronics

MDPI

Article

Fast and Accurate Memory Simulation by Integrating DRAMSim2 into McSimA+

Konstantin Bick [1], Duy Thanh Nguyen [1], Hyuk-Jae Lee [1] and Hyun Kim [2,*]

[1] Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, Korea; kon.bick@capp.snu.ac.kr (K.B.); thanhnd@capp.snu.ac.kr (D.T.N.); hyuk_jae_lee@capp.snu.ac.kr (H.-J.L.)

[2] Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea

* Correspondence: hyunkim@seoultech.ac.kr

Received: 2 July 2018; Accepted: 17 August 2018; Published: 20 August 2018

check for updates

Abstract: Computer architecture simulators play a crucial role in the verification of a new system's design. However, a single simulator may not be sufficient in covering detailed modeling of the entire system, thereby lacking in the simulation of a specific functionality under investigation. In this case, combining two simulators is necessary to compensate for the drawbacks of a single simulator. This paper proposes the integration of DRAMSim2, a simulator that thoroughly models DDR-SDRAM main memory architecture, into the application-level+ simulator McSimA+. The challenges of achieving an efficient integration, especially the integration of a cycle-accurate simulator into an event-driven environment, are addressed. The combined simulator achieves high accuracy due to cycle-accurate simulation while maintaining high speed and flexibility of the event-driven application-level+ simulator. The new simulator's overall system performance and the accuracy of the newly-integrated power model are verified against the gem5 simulator.

Keywords: computer architecture; simulator integration; DRAMSim2; McSimA+

1. Introduction and Motivation

The use of computer architecture simulators simplifies the verification process of a new system's design, compared to building a real system for testing. Thus, typical simulator requirements include reasonable simulation times while maintaining a high result accuracy. Numerous computer architecture simulators are available, differing in their functionality, speed and degree of detail. In [1], distinctive categories of computer architecture simulators have been discussed, and they will be referred to in this paper.

For research related to main memory architecture, DRAMSim2 is a renowned simulator that uses Micron's DDR-SDRAM power model for accurate power consumption estimates [2]. DRAMSim2 was designed as a dedicated main memory simulator because CPU and full-system simulators often lack an accurate model of the main memory system and therefore provide only rough main memory power consumption estimates [3]. DRAMSim2 can be driven with trace files or a full-system simulator. Driving DRAMSim2 with trace files, however, proves to be inconvenient because simulation times are extended owing to the fact that two separate steps are required: one in which trace files have to be generated and another in which they have to be simulated. The size of a trace file is proportional to the simulation's runtime, and thus, file sizes may quickly become impractically large. More crucially, memory transactions cannot be analyzed against their corresponding CPU instructions in real time. To compensate for these drawbacks, the more efficient solution is to integrate DRAMSim2 into a full-system simulation environment, executing memory transactions as they occur in the system. Previously, DRAMSim2 has been integrated into the MARSSx86 full-system simulator [3]. It is rather straightforward to integrate DRAMSim2 into a cycle-accurate full-system simulator. Cycle-accurate full-system simulators like MARSSx86 maintain a clock cycle that may have to be downscaled, but can be used directly to clock DRAMSim2. However, MARSSx86's main drawback is that the simulation speed lags behind event-driven simulators [4].

Event-driven full-system simulators such as McSimA+, do not simulate clock cycles when there is no activity in the system. Such simulators run faster than cycle-accurate simulators, while only slightly sacrificing overall system accuracy. However, because the focus of McSimA+ is on multicore research, accuracy is sacrificed in the simplified main memory system. Thus, for main memory research, the integration proves advantageous: it combines fast simulation speeds of the computer system with a cycle-accurate model of the main memory system. While the integration of two computer architecture simulators is not uncommon, this is especially true when both of them are cycle-accurate or rely on a similar source of clock information. For a general simulator integration, the main challenges include finding the right spot in the source code to cut off one simulator and let another take over, but also the synchronization of the information needed by both simulators to process requests. This requires a thorough understanding of both simulators' source code. If, however, a cycle-accurate simulator is integrated into an event-driven simulator, additional challenges apply: first, the problem of a missing clock cycle in one simulator has to be solved, and second, an analysis of the event management in the event-driven simulator is necessary to adjust its stopping condition in case additional events are generated. This paper discusses how to overcome all of these problems, while its focus is on the challenges that only apply to cycle-accurate and event-driven simulator integrations. The missing clock cycle is solved using self-calling events. These events are added to McSimA+'s global event queue with a frequency matching the main memory's I/O (input/output) bus clock. Thus, starvation of the combined simulator is prevented, which would otherwise have occurred if memory transactions had been added to DRAMSim2, but could not be processed because of the lack of a clock.

A two-step verification process is undergone to prove the combined simulator's validity: nine programs of the PARSEC 3.0 benchmark suite, differing in memory intensity, are simulated, and core system parameters like IPC (instructions per cycle) are compared against standalone McSimA+. To point out improved memory power model accuracy, again, smaller programs differing in memory intensity are simulated with the combined simulator, and results are validated against gem5 SE (System Call Emulation Mode) [5] and its default memory system simulator, DRAMPower [6]. Regarding the overall system performance, it is found that the new combined simulator does not reduce the simulated system performance accuracy already achieved by McSimA+. Accuracy improvements are achieved in power statistics, where especially the dynamic power is far overestimated by standalone McSimA+. Moreover, simulation speed is found to be higher than gem5 SE for all applications and almost on-par with standalone McSimA+.

The remainder of this paper is organized as follows: In Section 2, both McSimA+ and DRAMSim2 are briefly introduced. The integration of DRAMSim2 into the McSimA+ simulator and the solution to the clocking problem are discussed in Section 3. Section 4 describes the two-step verification of the combined simulator, and Section 5 concludes this article.

2. Simulators

McSimA+ is a multicore application-level+ simulator with detailed micro-architecture modeling [4]. While it was developed to offer a level of functionality that lies between an application and a full-system simulator (hence application-level+), its main purpose is to achieve higher simulation speeds in comparison with other common full-system simulators. It relies on Intel's Pin [7], a dynamic binary instrumentation tool, to extract and interpret the instructions of any application running on the host machine. This information is then used as input for a so-called PinToolSimulator (PTS). McSimA+, as an instance of such a PTS, models an entire multicore computer system comprising the processor cores, L1 and L2 caches, directory and memory controller. McSimA+ is an event-driven timing simulator, and thus, not every clock cycle is simulated when there is no activity in the system.

Events are processed whenever they occur, and fixed timing parameters represent delays in the system. As the focus of McSimA+ lies on multicore system research, its model of the main memory is simplified.

DRAMSim2 is a cycle-accurate memory system simulator [3]. It supports the simulation of the DDR2/3-SDRAM memory architecture, and result accuracy has been verified against memory manufacturer Micron's Verilog models. Not only can the timing and energy parameters of the DRAM chip be modified by the user, but also a variety of scheduling and row buffer policies is available. DRAMSim2 may simulate the memory transactions based on trace files that include crucial information such as timestamp, address and transaction type (i.e., READ or WRITE). However, DRAMSim2 natively includes a shared library interface to enable the integration into another full-system simulator, handling the information otherwise extracted from trace files.

3. Integration

The shared library interface of DRAMSim2 makes it possible to integrate the simulator into a full-system simulator environment. If the system simulator is event-driven, like McSimA+, higher simulation speeds can be achieved in comparison to simulators that simulate every clock cycle. However, DRAMSim2 requires to be clocked in order to process memory transactions. Therefore, to achieve the high accuracy of the cycle-accurate simulator while maintaining the speed and flexibility of the event-driven application-level+ simulator, DRAMSim2 should be well-integrated into the McSimA+ simulation environment. The integration process described in this section is summarized in a block diagram in Figure 1. It shows the memory controller interface of McSimA+ (PTSMemoryController) and the functions used to integrate DRAMSim2.

The shared library interface of DRAMSim2, which is called by the full-system simulator, McSimA+, comprises two functions: addTransaction() and update(). Additionally, a callback() function is implemented in McSimA+ to keep track of returned memory transactions. Requests from the directory to the memory controller in McSimA+ are forwarded to DRAMSim2 by the addTransaction() function. The update() function represents the clock input and has to be frequently called by the full-system simulator. When memory transactions are completed, the callback() function returns the physical address, transaction type and clock cycle of completion to McSimA+.

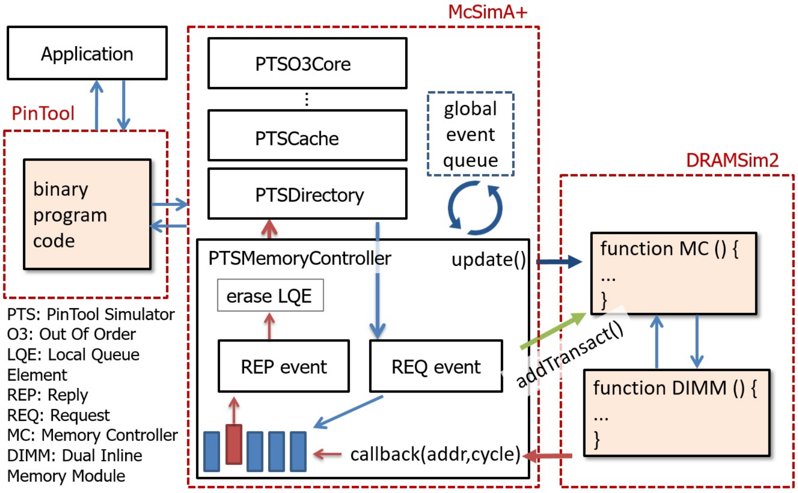


Figure 1. Block diagram of DRAMSim2 integration into the McSimA+ simulation environment.

McSimA+ maintains two queues: one queue for Local Queue Elements (LQEs) and another to keep track of requests between components, called the Global Event Queue (GEQ). The LQE stores information such as pointers to the address field and the request type. Whenever requests are added to a certain component's local queue, a timestamp in the form of a global event is added to the GEQ.

Local events are processed based on the timestamps stored in the GEQ. Once the GEQ is empty, the simulation ends.

As DRAMSim2 already comprises a sophisticated memory controller model, redundant functions in the memory controller file of McSimA+ were removed, leaving a lightweight memory controller interface (PTSMemoryController). To further improve result accuracy, the user space inherent virtual address used throughout McSimA+ is translated to its corresponding physical address. This is done by accessing the pagemap information stored in the Linux directory /proc/[pid]/pagemap, where pid refers to the process ID [8]. Once memory transactions containing the physical address and Boolean information of transaction type (true: WRITE) have been transferred to DRAMSim2, they are processed according to the queuing and scheduling policies of the main memory simulator.

Meanwhile, the LQEs of requests added by the directory must be stored in the memory controller interface of McSimA+, as they contain information that has not been passed to DRAMSim2, but is required in the ongoing simulation in McSimA+. Upon completion of the memory transactions in DRAMSim2, the shared library will call the callback() function to return the memory address and the transaction type. Using this information, the memory controller interface of McSimA+ will then be searched for the corresponding LQE of the transaction, which will then be returned to the directory.

In order for transactions to be processed in DRAMSim2, the main memory simulator requires a permanent call of the update() function, representing a clock's rising edge. As clock cycles may be skipped in McSimA+ when no events are occurring, the solution is to push clock cycle events frequently onto the GEQ. Upon construction of the lightweight memory controller interface, an initial global event is added to the GEQ to ensure that the processing function of the memory controller interface is called again. Inside the processing function, the update() function of DRAMSim2 is called, and transactions are transferred to the memory system simulator. Hereafter, a global event is added to the GEQ whenever the processing function in the memory controller is called, with the delay set according to the I/O bus frequency of the memory module.

These events, which are constantly being added to the GEQ to emulate the clock cycle of the memory simulator, cause another problem in that the previous break condition of McSimA+, which is to terminate the simulation once the GEQ is completely empty, is never met. To address this, a Boolean "program done" flag is implemented in the memory controller interface and is set to true once the program code of the simulated program has ended. When the "program done" flag is set, the memory controller interface investigates the elements in the GEQ to determine from which component they originate. If only elements added by the memory controller interface remain, then all other system processes have been completed. The interface will then stop adding new events, and the simulation will terminate correctly. The combined simulator makes it possible to track memory transactions in real time instead of relying on the fixed latency and bandwidth parameters that most full-system or application-level+ simulators, including McSimA+, utilize. Moreover, DRAMSim2 allows for a more thorough modification of the memory system and thus completes the detailed microarchitecture modeling of other components in McSimA+. Meanwhile, the combined simulator just marginally sacrifices the speed advantage of the standalone event-driven application-level+ simulator.

4. Result Comparison and Verification

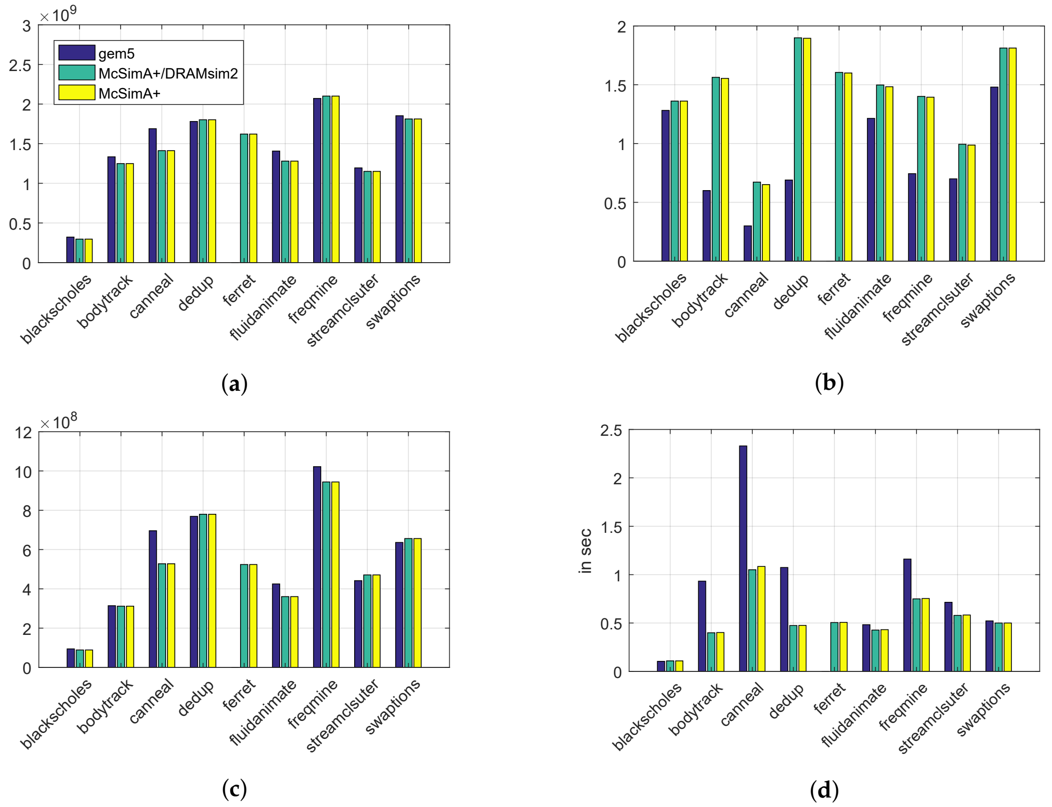
In this section, the combined simulator is compared against standalone McSimA+ and gem5 with its respective memory simulator DRAMPower. Three aspects are evaluated in order to verify the combined simulator's results. Firstly, the simulated performance of the application is thoroughly discussed in Section 4.1. Next, the simulation runtime is compared in Section 4.2. Finally, the power model and power consumption is thoroughly discussed (Section 4.3).

4.1. Simulated System Performance

McSimA+ has been verified to execute diverse benchmark programs with high IPC (instruction per cycle) accuracy in comparison to real server execution [4]. Simulations in the combined simulator, however, will differ in IPC owing to following reasons: Firstly, the virtual-to-physical address translation described in Section 3 might improve the memory performance as different virtual addresses may point to the same physical address. Secondly, congestion in DRAMSim2 is simulated more accurately because real bus packets are sent through the memory system. This might result in decreased memory performance compared to the standalone McSimA+ simulator in memory-intensive programs. The PARSEC 3.0 [9] benchmark suite has been simulated in all three simulators, gem5, McSimA+/DRAMSim2 and McSimA+, to verify that the combined simulator's overall system performance remained accurate. The benchmark packages **facesim**, **ferret**, **raytrace**, **vips** and **x264** failed due to unsupported system calls in gem5 SE mode. Out of those five applications, McSimA+ results for **facesim**, **raytrace**, **vips** and **x264** are omitted because their correct execution in the simulator could not be verified. The results for **ferret**, however, show an instruction count that fits well in the profile of the other PARSEC packages in "simsmall" configuration. Thus, the results of **ferret** are only shown for McSimA+ and the combined simulator. The remaining applications, however, are sufficient to represent different memory intensities. The same memory timings, based on Micron's "MT41J512M8" DDR3-SDRAM module [10], were adopted for all three simulators where applicable. However, McSimA+ does not offer the plethora of memory timing parameters found in DRAMSim2 and DRAMPower, the memory simulator inside gem5. The memory's I/O bus frequency of 800 MHz, which is emulated through reoccurring GEQ events in McSimA+, is set by modifying the memory controller's process interval parameter in McSimA+. The remaining system is set up as shown in Table 1.

Figure 2 shows a comparison of simulation results of gem5, McSimA+/DRAMSim2 and standalone McSimA+. The PARSEC benchmarks are compiled for single-thread execution, using the **gcc-serial** build configuration. This configuration is chosen to increase compatibility with the simulators. Owing to the lack of pthread library support, running multithreaded applications in gem5 SE mode proves to be challenging. McSimA+, on the other hand, supports the pthread library, and the combined simulator does not affect multithreaded executions. Shared library instances of DRAMSim2 are created according to the number of memory controllers specified in the McSimA+ system configuration file. Specifically, Figure 2a shows that the overall instruction count for the simulated benchmark packages only slightly differs between all the simulators. In the case of **canneal**, the difference is the greatest: gem5 simulated approximately 19.6 % additional instructions compared to McSimA+. The integration proposed in this paper does not alter the instruction count, and thus, it is identical for McSimA+ and McSimA+/DRAMSim2. Although the simulated systems were matched according to Table 1, the IPC count shows some more derivation. While it matches very well in the case of **blackscholes**, gem5 reveals slightly lower IPCs for **fluidanimate**, **streamcluster** and **swaptions** and considerably lower IPC for **bodytrack**, **canneal**, **dedup** and **freqmine**. However, because the overall IPC profile of gem5 and McSimA+ is matching in both cases, the variation is unlikely due to the inaccuracy of either simulator. Rather, it is explained by the differing memory architecture between gem5 and McSimA+. McSimA+, for example, integrates a directory-based cache coherence protocol that is not available inside gem5. As for the IPC difference between McSimA+ and the combined simulator shown in Figure 2b, the difference is negligible and proves that the combined simulator retains the level of accuracy achieved by standalone McSimA+. The different memory architectures make it infeasible to compare off-chip memory transaction counters directly. However, the amount of memory instruction reflected in L1 cache accesses (READ and WRITES) again matches well in gem5 and McSimA+, as can be seen in Figure 2c. This shows that McSimA+ does not skip memory instruction on a great scale, even though there are fewer L1 data cache accesses prominently in the case of **canneal**. Lastly, the runtime of the simulated applications is shown in Figure 2d. While gem5 reports the application's simulated runtime directly, it is not reported by McSimA+, and thus, it is calculated by the instruction count, the IPC and the clock period. As expected, the runtime of gem5, McSimA+ and the combined simulator matches well in the case of **blackscholes**, **fluidanimate**, **streamcluster** and **swaptions**, similar to the IPC figure. The largest derivation of the runtime in the three simulators is in the case of **canneal**. This might be due to a certain type of instruction taking longer to be executed in gem5's processor model compared to McSimA+.

More importantly, the simulated runtime difference between McSimA+ and the proposed simulator is marginal. However, in the case of **canneal**, the combined simulator shows a slightly increased performance. This small performance difference can be explained by the virtual-to-physical address translation implemented in the combined simulator, as described in Section 3. Since virtual addresses may point to the same physical address, their corresponding memory transactions in DRAMSim2 may be accelerated by the open-page row-buffer policy enabled in DRAMSim2. Virtual-to-physical address translation is not implemented in standalone McSimA+ because it lacks an accurate memory system, and thus, such a translation is superfluous.



(**a**) (**b**)

(**c**) (**d**)

Figure 2. PARSEC 3.0 simulation results of gem5, McSimA+/DRAMSim2 and McSimA+. The input size is 'simsmall' and the build-configuration **gcc-serial**. gem5 results for the packages **facesim**, **ferret**, **raytrace**, **vips** and **x264** are omitted because simulations failed due to unsupported system calls. For McSimA+, the correct execution could not be verified for **facesime**, **raytrace**, **vips** and **x264**; however, the results for ferret are shown. (**a**) Simulated instructions; (**b**) instructions per cycle (IPC); (**c**) L1 data cache accesses; (**d**) simulated program runtime.

Table 1. Simulated system parameters.

Parameter	Simulated System
CPU model	Out Of Order (OOO)
CPU frequency	2.4 GHz
Software threads	1
Memory controller	1
Memory modules (DIMM)	1
Main memory frequency	800 MHz
Main memory size	2 GHz
L1 data cache size	32 kB
L2 cache size	256 kB

Note: gem5 simulations were executed on a virtual machine running Ubuntu 16.04, while McSimA+ and McSimA+/DRAMSim2 simulations were executed on a virtual machine running CentOS 7. The host machine runs on an Intel Core i5-4590.

4.2. Simulation Performance

The simulation times of all three simulators are compared in Figure 3. McSimA+ shows the shortest simulation times for all simulated benchmarks, closely followed by the proposed combined simulator, which is 9.3% slower on average. The simulation runtime difference between those two simulators peaks for **dedup**, where the combined simulator runs approximately 19% longer. Due to the cycle-accuracy, simulations take longer when run on gem5. Even if IPC matches closely, like in the case of **blackscholes**, it takes gem5 approximately 66% longer to complete. The combined simulator is up to four-times faster in the PARSEC simulations. The biggest difference of the simulation time can be seen in the case of **canneal**, where gem5 runs for more than six hours, while McSimA+ and the combined simulator both can complete the simulation in less than two hours. The simulation time overhead, however, varies with different factors, like the type of instructions that are simulated. Thus, even though the simulation time of **canneal** and **dedup** is almost identical in McSimA+ and the combined simulator, it greatly differs in gem5.

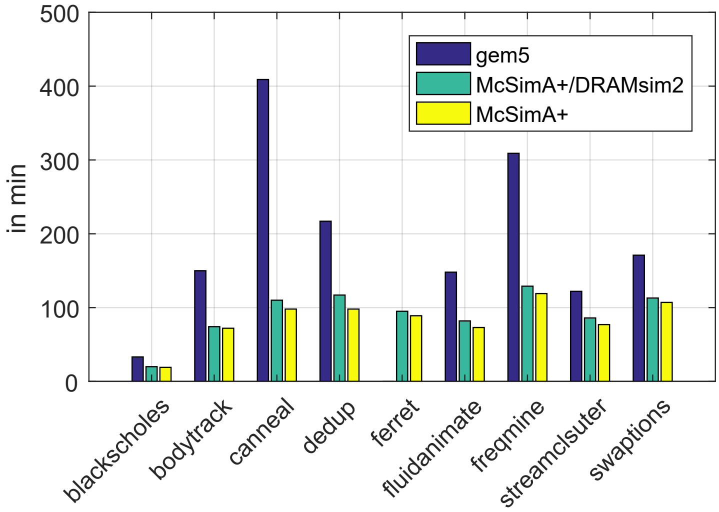
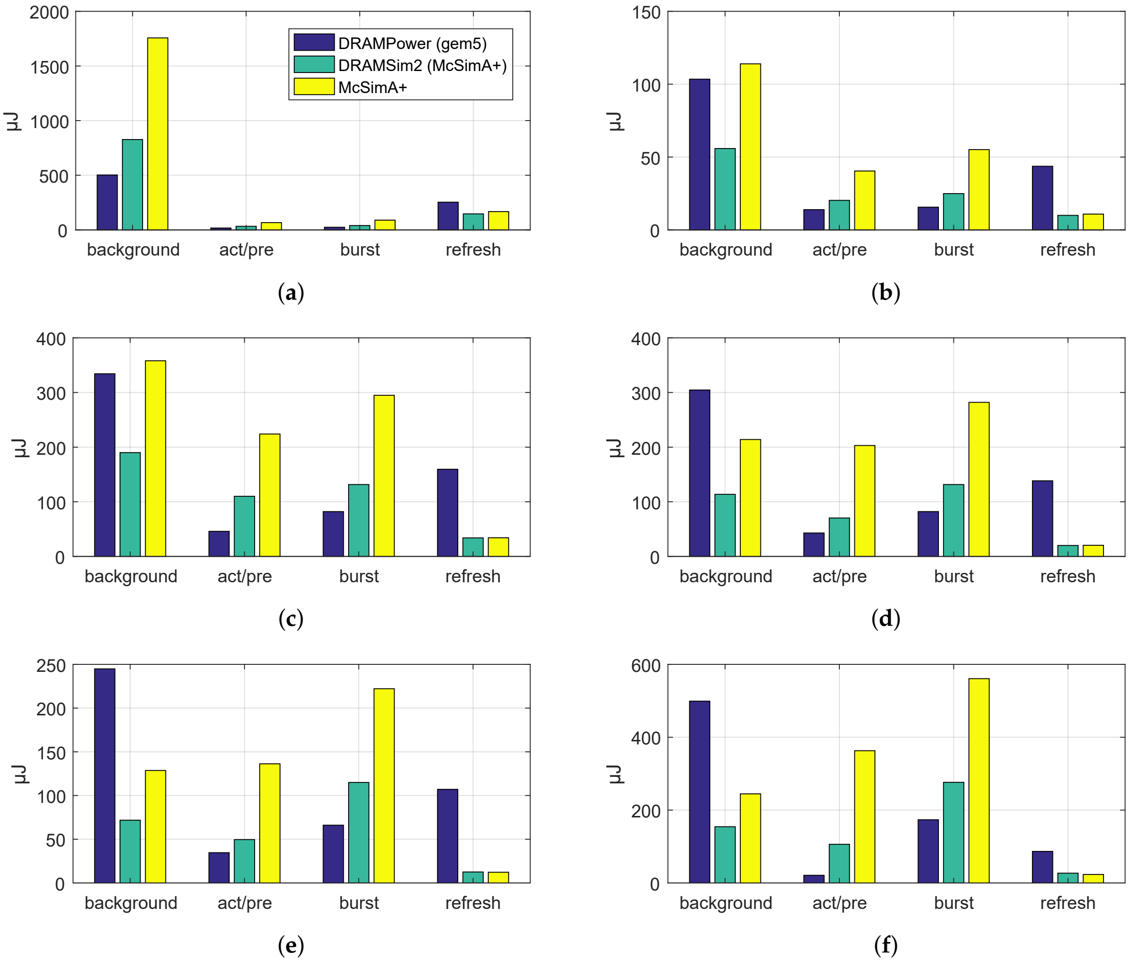


Figure 3. PARSEC 3.0 simulation times of gem5, McSimA+/DRAMSim2 and McSimA+.

4.3. Memory Power Model

As discussed in the previous subsection, the total number of off-chip memory transactions is not identical in gem5 and McSimA+ (McSimA+/DRAMSim2) for the PARSEC 3.0 applications. However, to accurately compare the memory power consumption, identical numbers of memory transactions are crucial. Thus, the power model comparison is analyzed using a different set of short applications, where memory reads and memory writes are perfectly matched between all the simulators (see Table 2). In order to achieve matching transaction numbers, cache sizes need to be adjusted in gem5 and McSimA+ accordingly. Moreover, the programs analyzed in this subsection are carefully selected to each show a different memory intensity (cf. "mem transact./instr" in Table 2).

Besides accurate overall system performance, modifiable and cycle-accurate power simulations are the key feature of the combined simulator. While DRAMSim2 adopts Micron's widely-acknowledged power model with minor changes, the model is improved in DRAMPower to achieve higher accuracy regarding power state transitions, command timings and refresh power calculation [6]. Thus, in Figure 4 memory power results of the combined simulator and standalone McSimA+ are compared with DRAMPower results, obtained through gem5 simulations. Table 3 summarizes the grouping of power statistic of the three simulators used in the comparison. In DRAMSim2 and McSimA+ power statistics had to be converted to energies for a comparison with DRAMPower. Six programs with different memory intensities (see Table 2) are compared using timing and current specifications of Micron's "MT41J512M8" DDR3-SDRAM module, which is also the default main memory device in gem5. RADIX, a sorting algorithm from the SPLASH-2 benchmark suite, has a low memory intensity. STREAM, included in the McSimA+ source files, represents a very high memory intensity. Two additional programs, ARRAY and SWAP , were written to represent memory intensities between RADIX and STREAM; see Table 2. ARRAY and SWAP are simple programs including the initialization, writing and reading of data arrays in a fashion that results in the desired memory intensity. Cache sizes in gem5 and McSimA+ are set in a way that read and write counts in gem5 and McSimA+ match almost exactly. Besides the power statistics, also the simulated instructions per second are compared.



(**a**) (**b**) (**c**) (**d**) (**e**) (**f**)

Figure 4. Energy statistics of programs with different memory intensities simulated in gem5/DRAMPower, DRAMSim2/McSimA+ and standalone McSimA+. (**a**) RADIX ; (**b**) ARRAY ; (**c**) SWAP -2; (**d**) SWAP-5; (**e**) SWAP-10; (**f**) STREAM.

Table 2. Applications used for power model comparison.

	RADIX	ARRAY	SWAP-2	SWAP-5	SWAP-10	STREAM
total instructions	28 m	3.4 m	5.6 m	2.3 m	1.2 m	1.25 m
memory reads	52 k	34 k	163 k	163 k	131 k	395 k
memory writes	23 k	13 k	92 k	92 k	73 k	125 k
mem transact./instr.	0.026%	1.4%	4.5%	11.3%	17.3%	42%
instr./sec (McSimA+)	210 k	173 k	188 k	140 k	117 k	104 k
instr./sec (combined)	207 k	161 k	161 k	118 k	90 k	66 k
instr./sec (gem5)	178 k	139 k	115 k	72 k	53 k	36 k

Note: mem transact./instr.: average memory transactions per instruction, instr./sec: instructions per second that the host machine is capable of simulating.

Table 3. Power statistics grouping.

DRAMSim2	DRAMPower	McSimA+
background	ACT bg + PRE bg + ACT pdn + PRE pdn	standby
ACT/PRE	ACT + PRE	dynamic
burst	write + read	i/o
refresh	refresh + self-refresh	refresh

Note: ACT: Activate, PRE: Precharge, bg: background, pdn: power-down, i/o: input/output.

The energy statistics of McSimA+ are straightforward. While background and refresh power are calculated with rank granularity, only dynamic and i/o power take memory command counters into consideration. Figure 4 shows that ACT/PRE and burst energies occupy a small portion of the total power in low memory intensity programs like RADIX, but account for the dominant proportions in memory-intensive programs such as STREAM. Although a memory power-down state in McSimA+ is mentioned in the source files, the memory did not enter this state during any simulation. Thus, background energy seems overestimated in RADIX and underestimated in STREAM. Activate (ACT)/Precharge (PRE) and burst energies are overestimated in all four simulated programs compared to DRAMPower, while refresh power is underestimated. As for DRAMSim2, all power statistics are based on counters that keep track of power states and commands. Power statistics are calculated based on timing and current parameters obtained from a DDR-SDRAM datasheet. Refresh power is found to be underestimated because it does not include required precharges before the execution of actual refresh commands [6]. ACT/PRE and burst power, however, are far more accurately estimated compared to McSimA+. As for background power, it is evident that state transitions' energy is lacking in DRAMSim2 when compared to DRAMPower: for a program with low memory intensity like RADIX, background power is slightly overestimated, but underestimated for medium to high memory intensity programs. Overall, if DRAMPower is assumed to be totally accurate, the error of the combined simulator as a percentage is as follows: background 38 %, ACT/PRE 130 %, burst 63 %, refresh 73 %. For McSimA+, it is: background 23 %, ACT/PRE 524 %, burst 247 %, refresh 72 % (see Figure 4). However, being modifiable and cycle-accurate, the current inaccuracies may be improved to match DRAMPower estimations at a later stage. This, however, is impossible for McSimA+ because power estimates are partly based on rank granularity, and thus, memory architecture research is hardly possible.

5. Conclusions

This paper discussed the advantages of integrating (and the actual integration of) a cycle-accurate main memory simulator into an event-driven application-level+ simulation environment. The combined simulator makes it possible to track memory transactions in real time instead of relying on the fixed latency and bandwidth parameters that most full-system or application-level+ simulators, including McSimA+, implement. A two-step verification process showed that the overall system accuracy is only slightly sacrificed while the simulation runtime is still shorter compared to other system emulation simulators like gem5. Moreover, the integration allows for a more thorough modification of the memory system and, thus, completes the detailed microarchitecture modeling of other components in McSimA+. Overall, an ideal balance of simulation runtime, main memory result accuracy and flexibility has been achieved. Most importantly, power estimates of custom DDRx-SDRAM designs, which the standalone McSimA+ simulator does not support, can be simulated and verified with the integrated simulator. Therefore, the combined simulator is ideal for memory research, especially the verification of custom main memory designs and architectures based on the DDR2/DDR3-SDRAM standard.

Author Contributions: K.B.: conceptualization, investigation, methodology, validation, visualization and writing the original draft. D.T.N.: methodology and validation. H.-J.L.: project administration and supervision. H.K.: conceptualization, project administration, supervision and writing review and editing.

Funding: This work received funding from Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-IT1602-03.

Acknowledgments: This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-IT1602-03.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Akram, A.; Sawalha, L. *A Comparison of x86 Computer Architecture Simulators*; Technical Report TR-CASRL-1-2016; Western Michigan University: Kalamazoo, MI, USA, 2016.
2. Calculating Memory System Power for DDR. 2001. Available online: https://www.micron.com/~/media/documents/products/technical-note/dram/tn4603.pdf (accessed on 15 March 2018).
3. Rosenfeld, P.; Cooper-Balis, E.; Jacob, B. DRAMSim2: A Cycle Accurate Memory System Simulator. *IEEE Comput. Archit. Lett.* **2011**, *10*, 16–19. [CrossRef]
4. Ahn, J.H.; Li, S.O.; Jouppi, N.P. McSimA+: A manycore simulator with application-level+ simulation and detailed microarchitecture modeling. In Proceedings of the 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Austin, TX, USA, 21–23 April 2013; pp. 74–85.
5. Binkert, N.; Beckmann, B.; Black, G.; Reinhardt, S.K.; Saidi, A.; Basu, A.; Hestness, J.; Hower, D.R.; Krishna, T.; Sardashti, S.; et al. The gem5 simulator. *SIGARCH Comput. Archit. News* **2011**, *39*, 1–7. [CrossRef]
6. Chandrasekar, K.; Akesson, B.; Goossens, K. Improved Power Modeling of DDR SDRAMs. In Proceedings of the 2011 14th Euromicro Conference on Digital System Design, Oulu, Finland, 31 August–2 September 2011; pp. 99–108.
7. Luk, C.-K.; Cohn, R.; Muth, R.; Patil, H.; Klauser, A.; Lowney, G.; Wallace, S.; Reddi, V.J.; Hazelwood, K. Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation. In Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), Chicago, IL, USA, 12–15 June 2005.
8. Pagemap, from the Userspace Perspective. Available online: https://www.kernel.org/doc/Documentation/vm/pagemap.txt (accessed on 5 March 2018).
9. Bienia, C.; Kumar, S.; Li, K. PARSEC vs. SPLASH-2: A Quantitative Comparison of Two Multithreaded Benchmark Suites on Chip-Multiprocessors. In Proceedings of the IEEE International Symposium on Workload Characterization, Seattle, WA, USA, 14–16 Septmeber 2008; pp. 47–56.
10. DDR3 SDRAM Features. 2009. Available online: https://micron.com/~/media/documents/products/data-sheet/dram/ddr3/4gb_ddr3_sdram.pdf (accessed on 5 March 2018).

© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).