ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.11, No.14, 1–8

An improved memory system simulator based on DRAMSim2

Zhen Xie[a)], **Yang Zhang, Jun Yang, and Longxing Shi**

National ASIC System Engineering Research Center, Southeast University, Nanjing, 210096, China

a) *xz@seu.edu.cn*

Abstract: An improved memory system simulator based on DRAMSim2 is presented. The memory system simulator has been widely used in the design space exploration of the SoC (System on Chip). DRAMSim2 is one of the more common DDR2/3 memory system simulators. A memory system consists of the memory controller and the DRAM device. DRAMSim2 models the memory controller in a general way, which makes it impractical to faithfully track the behavior of a specific memory controller. In this work, we present an improved memory system simulator based on DRAMSim2. In response to the differences between the memory controller DRAMSim2 modeled and the practical controller, part of DRAMSim2 is modified and improved, considering the structure characterizations of the practical controller. Experiments show that the improved simulator properly matches the practical memory system.

Keywords: SoC, memory system simulator, DRAMSim2

Classification: Electron devices, circuits, and systems

References

[1] V. Cuppu and B. Jacob: ISCA (2001) 62.

[2] Y. Kim, V. Seshadri, D. Lee, J. Liu and O. Mutlu: ISCA (2012) 368. DOI:10.1109/ISCA.2012.6237032

[3] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill and D. A. Wood: ACM SIGARCH CAN **33** [4] (2005) 92. DOI:10.1145/1105734.1105747

[4] L. Zhao, R. Iyer, J. Moses, R. Illikkal, S. Makineni and D. Newell: IEEE Micro **27** [4] (2007) 21. DOI:10.1109/MM.2007.66

[5] S. Rixner: MICRO (2004) 355. DOI:10.1109/MICRO.2004.22

[6] H. Choi, J. Lee and W. Sung: ISPASS (2011) 66. DOI:10.1109/ISPASS.2011.5762716

[7] P. Rosenfeld, E. Cooper-Balis and B. Jacob: IEEE CAL **10** [1] (2011) 16. DOI:10.1109/L-CA.2011.4

[8] B. Jacob, S. Ng and D. Wang: *Memory Systems — Cache, DRAM, Disk* (Morgan Kaufmann, Burlington, 2010) 497.

[9] A. Hansson, N. Agarwal, A. Kolli, T. Wenisch and A. N. Udipi: ISPASS (2014) 201. DOI:10.1109/ISPASS.2014.6844484

DOI: 10.1587/elex.11.20140466
Received May 14, 2014
Accepted June 26, 2014
Publicized July 11, 2014
Copyedited July 25, 2014

[10] D. Wang, B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel and B. Jacob: ACM SIGARCH CAN **33** [4] (2005) 100. DOI:10.1145/1105734.1105748

1 Introduction

A wide variety of simulators have been used for exploring the design space of the SoC (System on Chip). The memory system simulator is a key component for the full system simulation [1, 2]. Compared to the simplistic models of the memory system, which attach fixed latency or throughput to memory accesses [3, 4], the detailed memory system simulator can account for the highly complex behavior of modern memory systems [5, 6]. Among them, one of the more common memory system simulators is DRAMSim2 [7]. DRAMSim2 is a cycle-accurate DDR2/3 simulator, providing a detailed model for the memory systems.

The memory system consists of two components, the memory controller and the DRAM device. The DRAM device is standardized, and operations are constrained by dozens of timing parameters specified by the manufacturer. A typical memory controller schedules requests while keeping track of the parameters of the DRAM device. Of course, memory controllers designed by different vendors are diverse. However, DRAMSim2 models the memory controller in a general way, which makes it impractical to faithfully track the behavior of a specific memory controller. The inaccuracy will be described later.

In this paper, we aim to provide an improved memory system simulator based on DRAMSim2 for the practical memory system. The experimental results show that compared to DRAMSim2, the improved memory system simulator matches the practical memory system better. To the best of our knowledge, this is the first study that has ever been attempted.

The rest of this paper is organized as follows. In Section 2, we discuss our motivations by analyzing DRAMSim2. The design and implementation process of the improved memory system simulator is described in Section 3, with implementation results presented in Section 4. Finally, Section 5 concludes this paper.

2 Motivations

DRAMSim2 is implemented in C++ as an object oriented model of a DDR2/3 memory system that includes a detailed, cycle-accurate model of a memory controller that issues commands to a set of DRAM devices attached [7]. A block diagram of the components of DRAMSim2 is illustrated in Fig. 1. The DRAMSim2 core is wrapped in a single object called *MemorySystem*, which consists of two components: the memory controller and the DRAM device.

As mentioned, DRAMSim2 models the memory controller in a general way. Requests from the driver (any module that issues requests, such as CPU and GPU) are buffered into a transaction queue. These transactions are

DOI: 10.1587/elex.11.20140466
Received May 14, 2014
Accepted June 26, 2014
Publicized July 11, 2014
Copyedited July 25, 2014

converted into DRAM commands and placed into a command queue and then be issued to the DRAM device. The memory controller maintains the state of every memory bank and uses this information to decide which request should be issued to the DRAM device. Reads and writes are processed by the DRAM device and responses are returned at a later time [7].

The DRAM device is described by DRAM parameters, i.e. the timing constraints of a specific DRAM device, such as row access strobe time (t_RAS) and column access strobe time (t_CAS). These parameters can be found on the data sheet provided by the manufacturer. The memory controller should keep track of dozens of these parameters and issue the commands under the constraints.

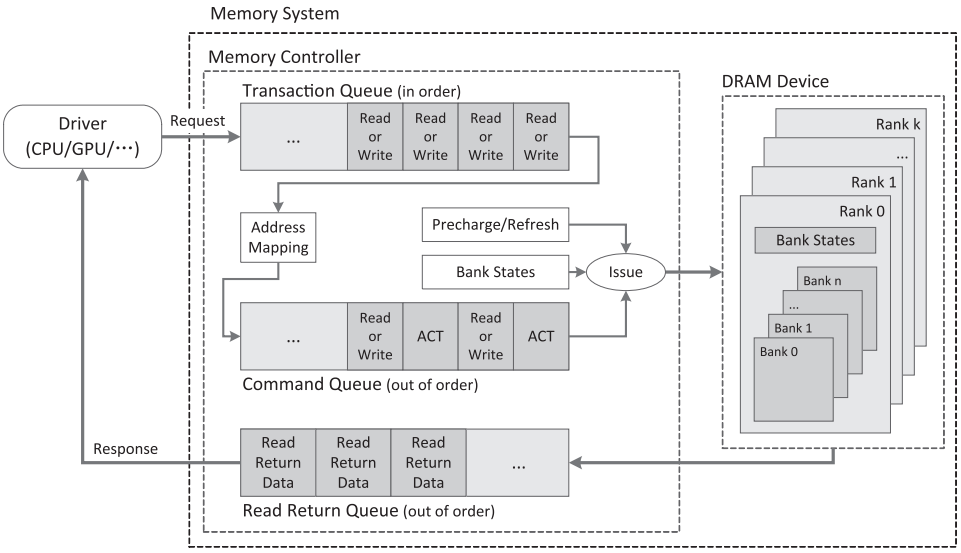

Fig. 1. Block diagram of the components of DRAMSim2.

The memory controller described by DRAMSim2 can be configured by some parameters that are independent of the DRAM device. These controller parameters include the address mapping scheme, the row buffer policy, the structure and scheduling policy of the command queue. However these parameters are not enough to describe a memory controller accurately (discussed in detail below). Therefore the memory controller DRAMSim2 modeled cannot match a specific memory controller properly.

For example, two commercial memory controllers we have and the memory controller of DRAMSim2 are configured with the same parameters. Statistics including the average bandwidth and detailed latency states are used to compare the three memory controllers. In Table I and Fig. 4, simulation-based results show that the two practical memory controllers with the same configurations are different with each other, and apparently the controller DRAMsim2 modeled matches neither of the two controllers. The comparison method is described later.

Therefore, in this paper an improved memory system simulator for the practical memory system is proposed, and the goal is to accurately model the

DOI: 10.1587/elex.11.20140466
Received May 14, 2014
Accepted June 26, 2014
Publicized July 11, 2014
Copyedited July 25, 2014

behavior of the practical memory controller. The design and implement process is described in the next section.

3 Design and implementation process

After researching the practical memory controllers in detail, we find that structures of the both two practical memory controllers are similar. An abstracted block diagram of the practical memory controller is illustrated in Fig. 2. Indeed, the block diagram is similar to that of DRAMSim2, a memory controller can be abstracted as a composition of the transaction unit, the command unit (including the command scheduler and the command execution) and the read return unit [8]. However, there are some problems in the way that DRAMSim2 models these units.

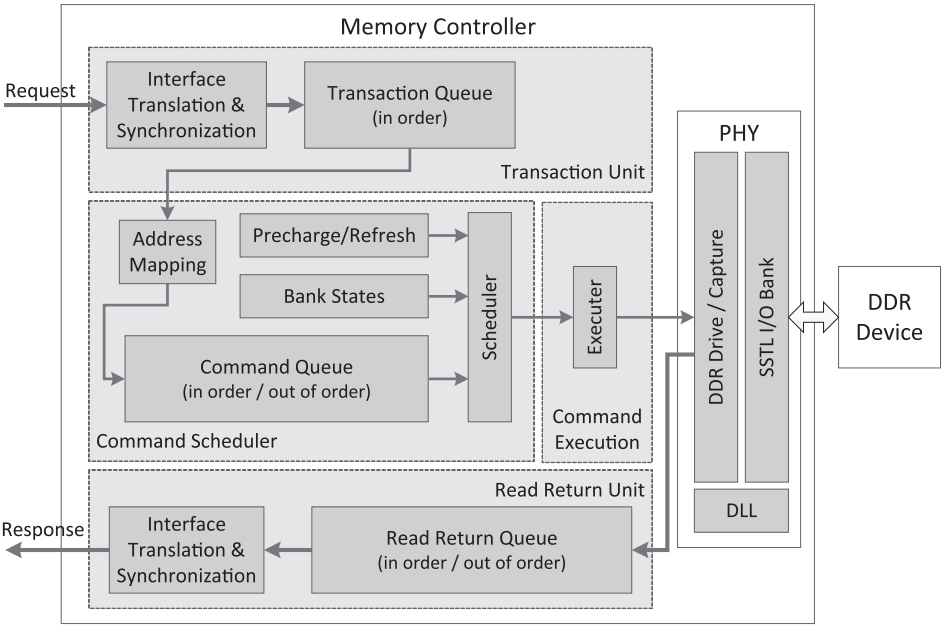

Fig. 2. Abstracted block diagram of the practical memory controller.

First, in practice the interface protocol between the driver and the memory controller (e.g. Advanced High-performance Bus (AHB), Advanced eXtensible Interface (AXI)) is different from the intermediate interface protocol of the memory controller (e.g. Packet Memory Interface (PMI) and Host Memory Interface (HMI) custom-defined by Synopsys and used in its memory controllers). Besides, since commonly the running frequency of the memory controller is different from that of the driver, the transaction unit should implement the interface translation and synchronization. In DRAMSim2, requests from the driver are buffered into the transaction queue, and at the same clock cycle directly converted into DRAM commands and placed into the command queue. DRAMSim2 ignores the latency introduced by the interface translation and synchronization.

DOI: 10.1587/elex.11.20140466
Received May 14, 2014
Accepted June 26, 2014
Publicized July 11, 2014
Copyedited July 25, 2014

Second, correspondingly the read return unit also should implement the interface translation and synchronization. DRAMSim2 also ignores the latency introduced, since responses are returned immediately when read return data are available in the read return queue.

Third, the command scheduler of the practical memory controller is pipelined, and the latency is inevitable. The latency is also ignored in DRAMSim2. Besides, DRAMSim2 is free to issue requests from the command queue out of order as long as it doesn't schedule writes ahead of dependent reads or violate timing constraints. This function may not be supported by some practical memory controllers, which means DRAMSim2 is not suitable for these memory controllers.

Finally, in practical memory system there is a PHY (physical layer) module connecting the memory controller and the DRAM device. DRAMSim2 ignores the latency introduced by the PHY.

In response to these differences, and considering the structure characterizations of the practical memory controller pipeline, we present an improved memory system simulator based on DRAMSim2. The DRAM device model and part of the command unit in DRAMSim2 are reserved, since they can issue requests from the command queue and return responses correctly under the timing constraints of the DRAM device. The remaining part of DRAMSim2 is modified and improved.

In order to realize general-utility, the modification provides configurability. The pipeline structure is modeled and three configurable parameters are designed to describe the latencies introduced by the transaction unit, the command unit and the read return unit. These parameters capture the frontend latency, which describe the pipeline stages of the memory controller, allowing us to reflect the controller design complexity. Andreas Hansson et al. use a static timing parameter to capture the frontend latency [9]. Obviously, our model is more accurate.

Another parameter is added to capture the static backend latency [9]. The backend latency describes the PHY design, thus allowing us to study the impact of the interconnection between the memory controller and the DRAM device. All of these parameters can be configured by the user according to the design of practical memory controllers.

4 Implementation results

To validate our memory system simulator, we compare it to DRAMSim2 and two practical memory systems (MS1 and MS2). DRAMSim2 and most other memory system simulators keep track of the bandwidth and latency of the requests [9, 10]. The simulator outputs detailed bandwidth and latency, which enables us to compare the effects of memory systems on these key performance metrics. The overall verification process is shown in Fig. 3.

DRAMSim2 and the improved simulator can be compiled in standalone mode, simulating requests recorded in a trace. Two sample traces (K6 and MASE) provided with DRAMSim2 are used to stimulate the memory system.

DOI: 10.1587/elex.11.20140466
Received May 14, 2014
Accepted June 26, 2014
Publicized July 11, 2014
Copyedited July 25, 2014

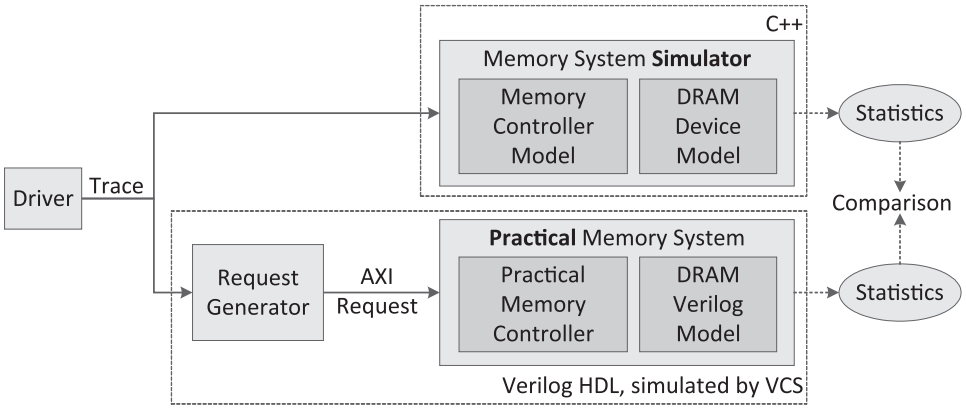

Fig. 3. An overview of the verification process.

Since the trace cannot be loaded directly by the practical memory controller, the trace is processed automatically by a request generator in Verilog to issue requests under the corresponding interface protocol. The request generator and the practical memory controller IP (Intellectual Property) are included along with the DRAM Verilog model (DRAM manufacturers such as Micron supply Verilog timing models for their DRAM devices, which can be used to simulate the transaction to the DRAM device) and executed by the VCS simulator.

In our experiments, the driver runs at 1 GHz, DDR2-800 is used and the running frequency of the memory system is 400 MHz. The DRAM parameters and the controller parameters except those new added parameters, e.g. the address mapping scheme (rank-bank-row-col), the row buffer policy (open page), the structure and scheduling policy of the command queue (per rank and round robin), are configured as the same.

Limited by the RTL simulation speed of the practical memory system, we run the simulation for 1 ms (1 million clock cycles of the driver), the average bandwidth and latency are shown in Table I. When the memory access rate meets the requirement of driver requests, the bandwidth is decided by the number of transactions recorded in the trace. Therefore, the bandwidths of the memory systems are the same. However, the average latency of DRAMSim2 is different from the practical memory systems as mentioned

Table I. Average bandwidth and latency of the memory systems.

	Trace	DRAMSim2 configured for MS1 & MS2	Practical memory system #1 (MS1)	Improved simulator configured for MS1	Practical memory system #2 (MS2)	Improved simulator configured for MS2
Average Bandwidth	K6	175 MB/s				
	MASE	64 MB/s				
Average Latency	K6	35.33 ns	110.75 ns	107 ns	80.51 ns	80 ns
	MASE	44.35 ns	128.42 ns	127 ns	97.60 ns	100 ns

DOI: 10.1587/elex.11.20140466
Received May 14, 2014
Accepted June 26, 2014
Publicized July 11, 2014
Copyedited July 25, 2014

before, and the latency distribution is also different as shown in Fig. 4. The improved memory system simulator matches the practical memory system properly, as the error of the average latency simulated is less than 3.4% and the latency distribution corresponds with that of the practical memory system.

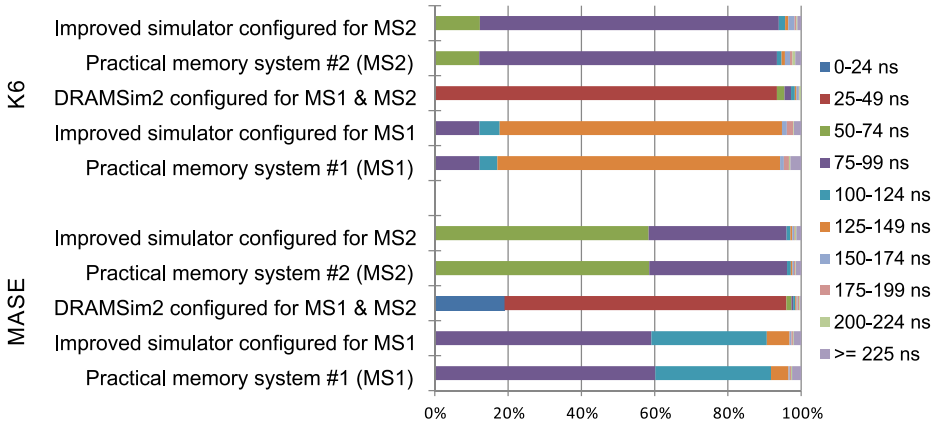

Fig. 4. Latency distribution of the memory systems.

Furthermore, 500 consecutive transactions are tracked and the latency of each transaction is recorded as illustrated in Fig. 5. The error of each the latency between the simulator and the practical memory system is shown in Fig. 6. The improved simulator properly matches the practical memory system. Some mismatches happen when a refresh is in progress. Requests that are issued while a refresh is in progress have to wait much longer than other requests. Since the memory controller of the simulator cannot issue a refresh at the same time with the practical memory controller, the mismatch is inevitable. As the refresh period is long (7800 ns), the mismatch is acceptable.

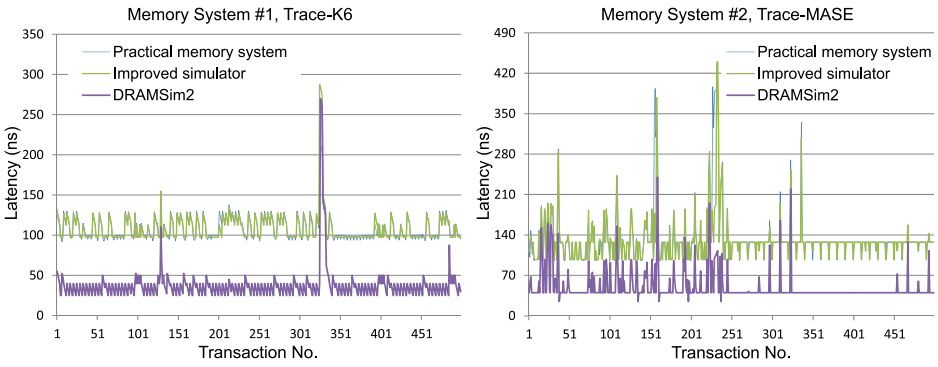

Fig. 5. Recorded latencies of 500 consecutive transactions.

DOI: 10.1587/elex.11.20140466
Received May 14, 2014
Accepted June 26, 2014
Publicized July 11, 2014
Copyedited July 25, 2014

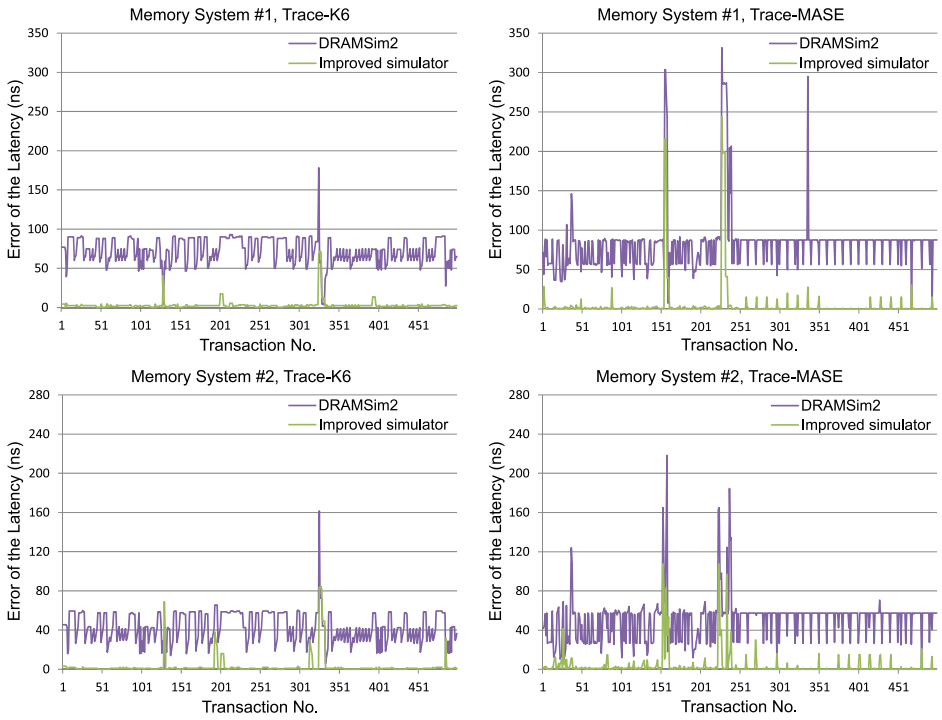

Fig. 6. The error of each the transaction latency between the simulator and the practical memory system.

5 Conclusion

In this paper, an improved memory system simulator based on DRAMSim2 is presented. A memory system consists of the memory controller and the DRAM device. DRAMSim2 models the memory controller in a general way, which makes it impractical to faithfully track the behavior of a specific memory controller. To address this problem, we present an improved memory system simulator based on DRAMSim2 considering the structure characterizations of the practical memory controller pipeline. The pipeline structure is modeled and some configurable parameters are designed to describe the frontend latency and the backend latency, allowing us to reflect the controller design complexity.

The improved memory system simulator is compared to DRAMSim2 and two practical memory systems to validate the accuracy. The results show that the improved simulator is much more accurate than DRAMSim2, matching the practical memory system properly. Overall, we think that the improved memory system simulator is more helpful for designers to explore the design space of the SoC and make the optimized design decisions.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61204023), the National High Technology Research and Development Program of China (863 Program) (2012AA012703) and the Natural Science Foundation of Jiangsu Province (BK2011334).

DOI: 10.1587/elex.11.20140466
Received May 14, 2014
Accepted June 26, 2014
Publicized July 11, 2014
Copyedited July 25, 2014