ADVCOMP 2014

The Eighth International Conference on Advanced Engineering Computing and Applications in Sciences

ISBN: 978-1-61208-354-4

August 24 - 28, 2014

Rome, Italy

ADVCOMP 2014 Editors

Jaime Lloret Mauri, Polytechnic University of Valencia, Spain

Sigeru Omatu, Osaka Institute of Technology, Japan

home

top

<<

>>

ADVCOMP 4: Development of computing support

Modeling and Visualization of Cataract Ontologies

Klaus Peter Scherer, Karlsruhe Institute of Technology, Germany
Constantin Rieder, Karlsruhe Institute of Technology, Germany
Christian Henninger, Karlsruhe Institute of Technology, Germany
Markus Germann, Karlsruhe Institute of Technology, Germany
Joachim Baumeister, denkbares GmbH, Germany
Jochen Reutelshöfer, denkbares GmbH, Germany

Towards a Decision Support System for Automated Selection of Optimal Neural Network Instance for Research and Engineering

Rok Tavcar, Cosylab, d.d., Slovenia
Jože Dedic, Cosylab, d.d., Slovenia
Drago Bokal, Faculty of Natural Sciences and Mathematics, University of Maribor, Slovenia
Andrej Žemva, Faculty of Electrical Engineering, University of Ljubljana, Slovenia

A Memory Controller for the DIMM Tree Architecture

Young-Jong Jang, School of Electronics Engineering, Kyungpook National University, Republic of Korea
Young-Kyu Kim, School of Electronics Engineering, Kyungpook National University, Republic of Korea
Taewoong Ahn, School of Electronics Engineering, Kyungpook National University, Republic of Korea
Byungin Moon, School of Electronics Engineering, Kyungpook National University, Republic of Korea

Classification of Pattern using Support Vector Machines: An Application for Automatic Speech Recognition

Gracieth Batista, Federal Institute of Maranhão, Brazil
Washington Silva, Federal Institute of Maranhão, Brazil
Orlando Filho, Federal Institute of Maranhão, Brazil

Searching Source Code Using Code Patterns

Ken Nakayama, Tsuda College, Japan
Eko Sakai, Otani University, Japan

foreword
committee
sessions
authors

home
top
<<
>>

A Memory Controller for the DIMM Tree Architecture

Young-Jong Jang, Young-Kyu Kim, Taewoong Ahn, and Byungin Moon
School of Electronics Engineering
Kyungpook National University
Daegu, Republic of Korea
e-mail: youngjong25@ee.knu.ac.kr, kyk79@ee.knu.ac.kr, myannet11@gmail.com, bihmoon@knu.ac.kr

***Abstract*—The dual in-line memory module (DIMM) tree architecture was proposed to solve signal integrity and data access latency problems of many-DIMM system. Although the DIMM tree demands a memory controller specific to it, there has been little research on the memory controller for the DIMM tree. For this reason, this paper proposes a new memory controller architecture for the DIMM tree. This architecture was modeled using DRAMSim2 for its verification and analysis, and the experimental results show that the proposed DIMM tree memory controller works properly and efficiently.**

Keywords-DIMM tree; many-DIMM system; DIMM to DIMM transfer; memory controller

I. INTRODUCTION

In the traditional computer systems, there are two memory access methods: one is the multi-drop bus based memory access and the other is the point-to-point link-based memory access. The former method is mainly used in the traditional computer systems [1], but it is not appropriate for implementation of large-capacity main memory systems due to its signal integrity issues [2]. In the latter method, DIMM modules are connected through daisy chains. The fully buffered DIMM (FB-DIMM) is an example that uses point-to-point links [3]. In this method, the control signals are buffered and repeated at each DIMM. This method has the disadvantage that as the number of connected DIMM modules increases, the transmission time of control signals also increases. In order to resolve these problems, the DIMM tree architecture has been proposed. The DIMM tree architecture overcomes the transmission delay time and signal integrity issues by a tree structure connection of DIMM modules [4].

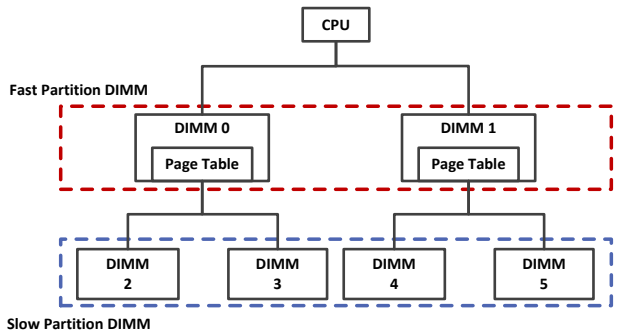

Figure 1. Partitioned DIMM tree structure.

The structure and operation of the DIMM tree architecture were presented for the implementation of the DIMM tree architecture, and the concept of the partitioned DIMM tree architecture and the direct DIMM-to-DIMM transfer was established in order to efficiently manage large-capacity memory of the DIMM tree architecture [5]. The memory controller for the DIMM tree architecture is also very important for the implementation of the DIMM tree architecture. However, there was little concrete discussion on the memory controller for the DIMM tree architecture. Thus, this paper proposes a hardware architecture of a memory controller for the management of the DIMM tree architecture.

The rest of the paper is organized as follows. Section 2 describes the background of the DIMM tree architecture. Section 3 presents the hardware architecture of the proposed DIMM tree memory controller. In Section 4, we present the experimental environments, and analyze the results of the experiments. Finally, Section 5 concludes the paper.

II. DIMM TREE ARCHITECTURE

Therdsteerasukdi et al. [4] presented the DIMM tree architecture of better scalability by connecting DIMM modules in a tree structure to solve signal integrity and access latency problems. Then, they improved their research by proposing the partitioned DIMM tree and the direct DIMM-to-DIMM transfer for efficient memory management.

A. Partitioned DIMM Tree

The partitioned DIMM tree architecture consists of a fast partition and a slow partition, as in shown Fig. 1 [5]. The fast partition corresponds to the DIMMs at a level closer to the CPU. Thus, it takes one DIMM access latency to access the fast partition. The slow partition contains the remaining DIMMs. The fast partition is used as a cache of the slow partition, which works as the main memory. The relation between the fast partition and the slow partition is similar to that between main memory and hard disk in a virtual memory system, so data transfer between the fast partition and slow partition is performed by page units. In addition, the page fault handler to process page faults is necessary in the memory controller [5][6].

B. Page Fault Handler

The partitioned DIMM tree architecture uses a small space in the fast partition for a fast partition page table to manage page translation between the fast and slow partitions. When memory access is requested from the CPU, the fast partition page table in the fast partition is checked to find the

 ISBN: 978-1-61208-354-4

fast partition page corresponding to the requested page of the slow partition. When a page fault occurs, the page fault handler updates the requested page to the fast partition from the slow partition, and writes back the replaced page to the slow partition from the fast partition. These update and write back are processed with the proposed direct DIMM-to-DIMM transfer [5]. This direct DIMM-to-DIMM transfer reduces the overhead of page fault processing.

III. PROPOSED DIMM TREE MEMORY CONTROLLER

In this section, a hardware architecture of a memory controller is proposed for the DIMM tree architecture described in Section 2. The proposed hardware architecture of the DIMM tree memory controller is shown in Fig. 2. The organization and operation of the DIMM tree memory controller is as follows.

When a read or write request is entered from the CPU to the bus, and the request is loaded into the Transaction Buffer, and then four fast partition page table entries, which are accessed using the least significant bits of the requested slow partition page number as the index, as shown in Fig. 3, are loaded into the Page Table Buffer. The Hit/Miss Control Unit is used to check whether a page fault occurs or not for the request. This module compares the tag of the slow partition address with each of the slow partition page number tags of four page table entries loaded on the Page Table Buffer. When one of the four page table entry tags matches the tag of the slow partition address and the valid bit of the corresponding page table entry is 1, the state becomes 'hit'. Otherwise the state is 'miss' meaning that a page fault occurs. In case of 'hit', the hit/miss signal of the Hit/Miss Control

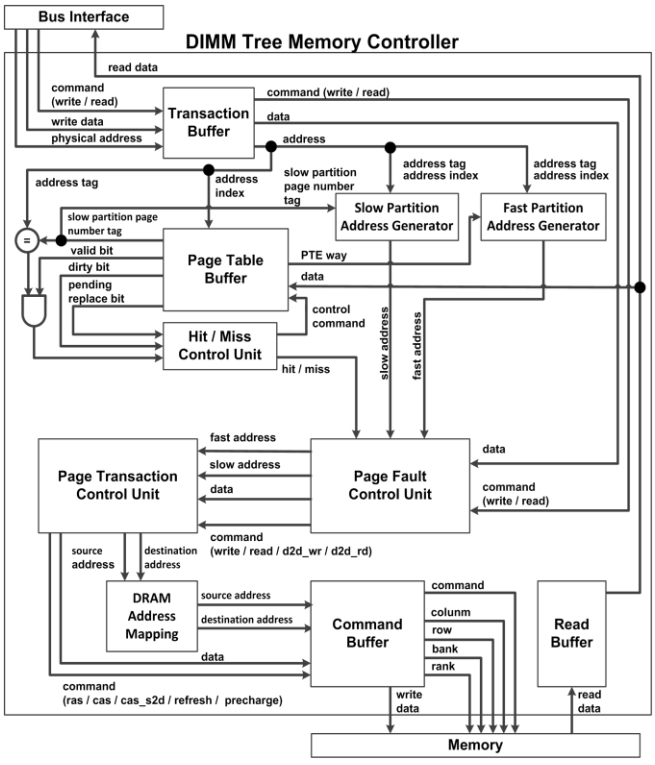

Figure 2. Architecture of the proposed DIMM tree memory controller.

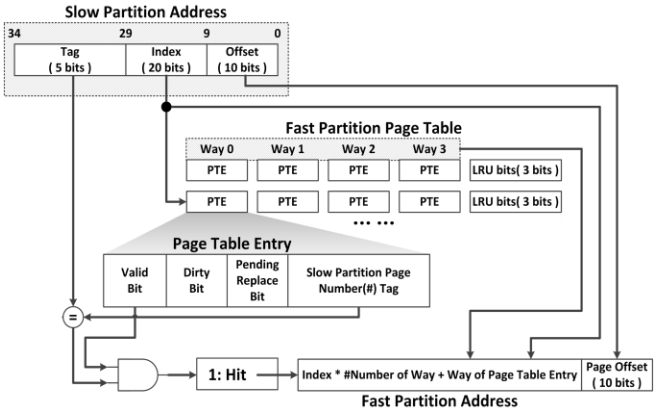

Figure 3. Address translation from the slow partition to fast partition.

Unit is set to 1, and the Page Table Buffer outputs the way number of the 'hit' page table entry.

The Page Fault Control Unit generates the DIMM-to-DIMM transfer command. When the hit/miss signal from the Hit/Miss Control Unit is "1", the Page Fault Control Unit directly transfer the request command in the Transaction Buffer to the Page Transaction Control Unit. However, when a page fault occurs, Some DIMM-to-DIMM commands are inserted. When the page-faulted request command is a write, the Page Fault Control Unit inserts the DIMM-to-DIMM update command (D2D_RD) to transfer the faulted page from the slow partition to the fast partition. After the write request command, the DIMM-to-DIMM writeback command (D2D_WR) is issued to write the corresponding page of the fast partition to the slow partition. When the request command is a read, the Page Fault Control Unit inserts the DIMM-to-DIMM update command (D2D_RD) for getting the faulted page of the slow partition to the fast partition before issuing the read request command to read data from the fast partition. These generated command streams are transferred to the Page Transaction Control Unit.

The Page Transaction Control Unit processes the commands from the Page Fault Control Unit, and makes DRAM control commands such as row active (RAS), column read (CAS), CAS source to destination (CAS_S2D), REFRESH, and PRECHARGE. The CAS_S2D is a special command for the DIMM tree architecture [5], but the others are common commands for controlling DRAM. When the command from the Page Fault Control Unit is either a write or read, the Page Transaction Control Unit generates RAS and CAS commands. Since the CPU can only access the fast partition, the RAS and CAS from the Page Transaction Control Unit are associated with the fast partition. On the other hand, if the command from the Page Fault Control Unit is either D2D_WR or D2D_RD, the Page Transaction Control Unit generates RAS and CAS_S2D commands. The D2D_WR and D2D_RD have two addresses, one is a source address and the other is a destination address. The source address and destination address are determined depending on the type of the command. If the command is D2D_WR, the source address is a fast partition address, and the destination address is a slow partition address. It is vice versa in case of the D2D_RD command.

TABLE I. SIMPLESCALAR PARAMETERS

Simulation Tool	SIM-cache
Processor ISA	PISA instruction
L1 D-cache	256 sets, 32 bytes cache line, 4-way set-associative, LRU policy
L1 I-cache	
L2 D-cache	1024 sets, 64 bytes cache line, 4-way set-associative, LRU policy
L2 I-cache	
I-TLB	16 sets, 4096 bytes cache line, 4-way set-associative, LRU policy
D-TLB	32 sets, 4096 bytes cache line, 4-way set-associative, LRU policy

TABLE II. DRAMSIM2 PARAMETERS

DRAM type	DDR3-1600[a]
DRAM Data Width	8 bytes
Row buffer policy	close page policy
Bank per rank	8
Row count	16384
Column count	1024
DIMM size	1 GB
Page size	1 KB

a. Micron, x8 DDR3-1600 DIMM [12]

The command generated by the Page Transaction Control Unit, is transferred to the Command Buffer. The Command Buffer sends the DIMM tree commands (RAS, CAS and CAS_S2D) and addresses (rank, bank, row and column) to DIMM modules, which process those commands.

IV. EXPERIMENTS

For the verification of the proposed DIMM tree memory controller hardware architecture, we modeled the proposed hardware architecture by DRAMSim2 [7] and generated six workloads each of which has 500 thousand requests for the simulation.

A. Workload

For the performance verification of the proposed DIMM tree memory controller hardware architecture, we need workload of large size with a wide range of memory addresses. For this reason, we extracted the memory transaction traces by running bzip2, gromacs, hmmer, lbm, mcf and milc in the SPEC CPU 2006 benchmark [8] in the SimpleScalar simulator [9]. Bzip2, gromacs and hmmer use a wide range of memory addresses, but have high locality. Lbm, mcf and milc have high locality and use a small range of memory addresses [10]. Experiments were carried out using the SimpleScalar with the parameters describes in Table Ⅰ. Also, the PIN [11] was used to attach the time stamp to the memory transaction traces extracted from the SimpleScalar.

B. Experimental Environments

The DRAMSim2 was used for modeling the proposed DIMM tree memory controller. It is assumed that capacity of each DIMM is 1GB and the page size is 1 KB. The DIMM tree architecture has a tree structure with the depth of 2 and the degree of 4, as shown in Fig. 4. The fast partition has the size of 4 GB, and the slow partition is composed of 16 GB. The block diagram of the proposed DIMM tree memory controller model is shown in Fig. 5. The detailed parameters applied to DRAMSim2 is shown in Table Ⅱ.

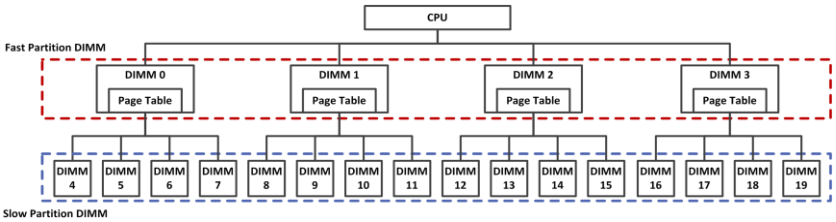

Figure 4. DIMM tree structure used in experiments.

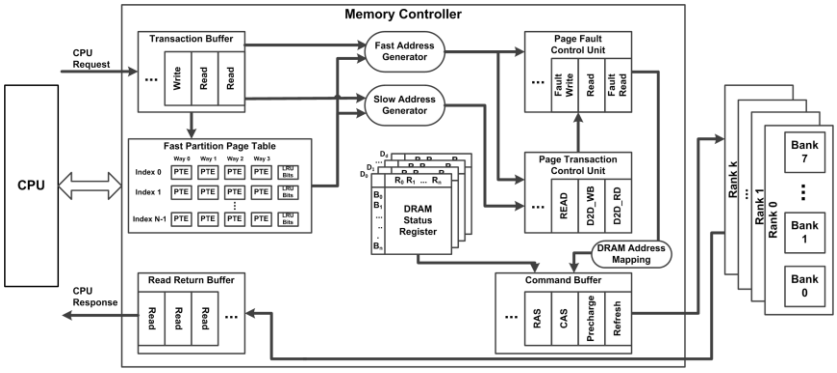

Figure 5. Simulation environment of the proposed DIMM tree memory controller.

C. Results

The generated workloads, each of which has a total of 500 thousand requests, are used for the CPU Requests of the DIMM tree memory controller model of DRAMSim2. Through simulations with workloads, it was verified that the proposed DIMM tree memory controller operates properly. In addition, performance analysis of the proposed DIMM tree memory controller was carried out as follows.

Fig. 6 shows hit rate per 1,000 traces when a total of 10 thousand traces of each workload are processed. Lbm, mcf and milc are memory-intensive workload, thus have high hit rate. After 5 thousands of traces have been processed, the hit rates of lbm, mcf and milc was maintained at about 73%. On the other hand, hit rates of bzip2, gromacs and hmmer were continuously increased. As shown in Fig. 7, the hit rates of gromacs and hmmer increased rapidly after 50 thousands of traces were processed. This is mainly because that the number of compulsory misses, which occur intensively at the early time of simulation, decreases after running 50 thousand traces. Workloads such as bzip2, gromacs and hmmer have a wide address range, so they need a longer time to fill the fast partition pages. The number of cycles taken to run all the traces is shown in Table Ⅲ. The performance for the workload hmmer is measured best because the hit rate for hmmer is best.

TABLE III. CLOCK CYCLES TAKEN TO PROCESS EACH WORKLOAD

Workloads	10000 traces (cycles)	100,000 traces (cycles)	500,000 traces (cycles)
bzip2	1,263,919	12,720,015	39,481,566
gromacs	538,966	3,409,885	11,668,567
hmmer	508,589	1,999,400	6,265,317
lbm	522,217	5,168,498	25,814,382
mcf	535,822	5,276,435	26,298,294
milc	535,397	5,279,990	26329965

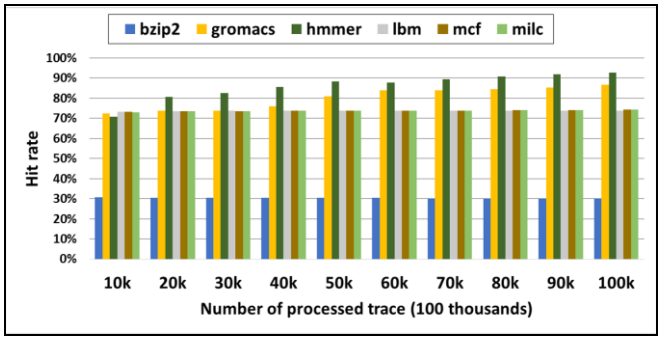

Figure 7. Hit rate per 10,000 traces when running 100,000 traces of each workload.

V. CONCLUSIONS AND FUTURE WORK

This paper proposed a hardware architecture of the DIMM tree memory controller. The proposed architecture was modeled using the DRAMSim2, Workloads were generated using the SPEC CPU 2006 to verify the functionality and performance. The experimental results show that the proposed DIMM tree memory controller works properly. This paper presented a DIMM tree memory controller which can be used as a reference model in DIMM tree based large memory systems. The study of this paper assumes that he DIMM tree memory controller operates on the single-processor environment. However, recently, the multiprocessor environment is used widely. So, our future work is to run multiple loads in parallel on the DIMM tree memory controller for multiprocessor systems.

ACKNOWLEDGMENT

This investigation was financially supported by Semiconductor Industry Collaborative Project between Kyungpook National University and Samsung Electronics Co. Ltd. This study was supported by the BK21 Plus project funded by the Ministry of Education, Korea (21A20131600011). This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the C-ITRC (Convergence In-formation Technology Research Center) support program (NIPA-2014-H0401-14-1004) supervised by the NIPA (National IT Industry Promotion Agency).

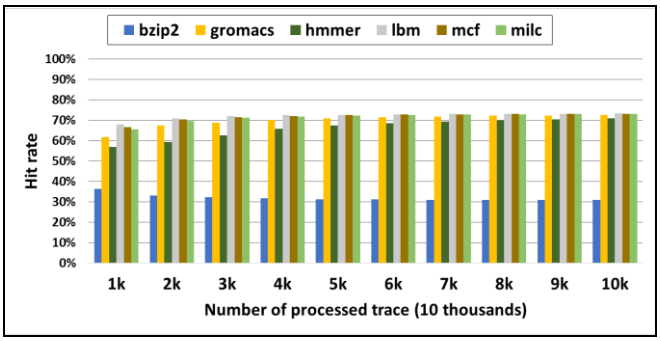

Figure 6. Hit rate per 1,000 traces when running 10,000 traces of each workload.

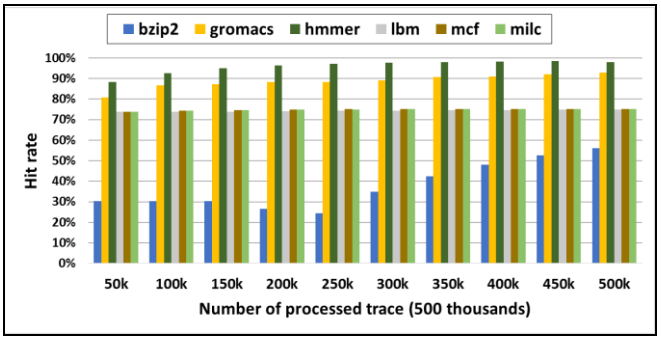

Figure 8. Hit rate per 50,000 traces when running 500,000 traces of each workload.

REFERENCES

[1] J. H. Kim, et al., "Challenges and Solutions for Next Generation Main Memory Systems," Proc. IEEE 18th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS 09), Oct. 2009, pp. 93-96, doi: 10.1109/EPEPS.2009.5338468.

[2] B. Jacob, S. Ng, and D. Wang, "Memory Systems: Cache, Dram, Disk," Morgan Kaufmann, 2008, pp. 377-391.

[3] B. Ganesh, A. Jaleel, D. Wang, and B. Jacob, "Fully-Buffered DIMM Memory Architectures: Understanding Mechanisms, Overheads and Scaling," Proc. IEEE 13th International Symposium on High Performance Computer Architecture (HPCA 07), Feb. 2007, pp. 109-120, doi: 10.1109/HPCA.2007.346190.

[4] K. Therdsteerasukdi, et al., "The DIMM Tree Architecture: A High Bandwidth and Scalable Memory System," Proc. IEEE 29th International Conference on Computer Design (ICCD 11), Oct. 2011, pp. 388–395, doi: 10.1109/ICCD.2011.6081428.

[5] K. Therdsteerasukdi, et al., "Utilizing Radio-Frequency Interconnect for a Many-DIMM DRAM System," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 2, no. 2, June 2012, pp. 210–227, doi: 10.1109/JETCAS.2012.2193843.

[6] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable High Performance Main Memory System Using Phase-Change Memory Technology," Proc. 36th annual international symposium on Computer architecture (ISCA 09), June 2009, pp. 24–33, doi: 10.1145/1555754.1555760.

[7] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMsim2: A Cycle Accurate Memory System Simulator," IEEE Computer Architecture Letters, vol. 10, no. 1, June 2011, pp. 16-19, doi: 10.1109/L-CA.2011.4.

[8] J. L. Henning, "Spec CPU2006 Benchmark Descriptions," ACM SIGARCH Computer Architecture News, vol. 34. no. 4, Aug. 2006, pp. 1–17, doi: 10.1145/1186736.1186737.

[9] T. Austin, L. Eric, and D. Ernst, "SimpleScalar: An Infrastructure for Computer System Modeling," IEEE Computer, vol. 35, no. 2, Feb. 2002, pp. 59-67, doi: 10.1109/2.982917.

[10] X. Mingli, D. Tong, Y. Feng, K. Huang, and X. Cheng, "Page policy control with memory partitioning for DRAM performance and power efficiency," IEEE International Symposium on Low Power Electronics and Design (ISLPED 13), Sept. 2013, pp. 298-303, doi: 10.1109/ISLPED.2013.6629312.

[11] V. Reddi, A. M.Settle, D. A. Connors, and R. S. Cohn. "PIN: A Binary Instrumentation Tool for Computer Architecture Research and Education," Proc. ACM Workshop on Computer Architecture Education (WCAE 04): held in conjunction with the 31st International Symposium on Computer Architecture, June. 2004, p. 22, doi: 10.1145/1275571.1275600.

[12] Micron. 1 Gb: x4,x8,x16 DDR3 SDRAM Features 2006.

 ISBN: 978-1-61208-354-4