International Journal of Control and Automation
Vol. 8, No. 2 (2015), pp. 43-54
http://dx.doi.org/10.14257/ijca.2015.8.2.05

A T-DIMM ID Based Command Routing Method for the DIMM Tree Architecture

Young-Kyu Kim and Byungin Moon[*]

School of Electronics Engineering, Kyungpook National University, Daegu, Korea
kyk79@ee.knu.ac.kr, bihmoon@knu.ac.kr

Abstract

The DIMM tree architecture was proposed to overcome the drawbacks of the traditional DIMM interface methods such as multi-drop bus and point-to-point links. As the DIMM tree architecture has a tree topology, the most important function of the DIMM tree architecture is the command routing function between upper-level and lower-level DIMMs. In order to implement this function, it uses a special type of DIMMs, called tree DIMMs (T-DIMMs). However, because the pre-proposed T-DIMM uses the command routing method based on routing tables, some problems occur such as hardware cost increase due to the routing table, routing table initialization, and limitation in the number of ranks in one T-DIMM. To solve such problems, this paper proposes a command routing method based on a T-DIMM ID without the routing table. The proposed method was modeled with a register transfer model (RTL). The model of the proposed method has 2 ranks per T-DIMM, and doesn't use the routing table for command. This RTL model was synthesized into the gate-level model. The experimental results confirmed that hardware cost of the proposed method is reduced dramatically while its performance is the same, as compared with the previous routing method in the DIMM tree architecture.

Keywords: *DIMM tree architecture, main memory, in-memory computing, DRAM*

1. Introduction

Traditional dual inline memory module (DIMM) interface methods such as multi-drop bus and point-to-point links have a limited number of connectable DIMMs because of the signal integrity issue and added latency [1-2]. DIMM tree architecture was proposed to overcome the drawbacks of the traditional DIMM interface methods, and to implement the many-DIMM system [3]. This tree architecture uses a special type of DIMMs called tree DIMMs (T-DIMMs) connected by a tree topology [4], as shown in Figure 1. Level 2 T-DIMMs in Figure 1 receive T-DIMM access commands from the memory controller, and sometimes it must route the received commands to level 3 T-DIMMs in order to access level 3 T-DIMMs, which are not directly connected with the memory controller [5-7].

To implement the command routing function, which is one of the most important functions of the T-DIMM, a T-DIMM must be able to choose between aborting the received command, executing that command, or forwarding it to lower level T-DIMMs. As the previous studies about the DIMM tree architecture are based on the assumption that all T-DIMMs in the DIMM tree architecture have only one rank per one T-DIMM, each T-DIMM decides where to route a command depending on the value of the routing table entry corresponding to the rank number [3]. This function is performed by the DIMM interface router (DIR) block in the T-DIMM. Particularly the Router module in the DIR implements detailed functions related command routing. However, we found

[*] Corresponding author

following problems about using the routing Table based on the unique rank numbers for command routing.

- **T-DIMM Hardware Cost Overhead**: This routing method requires high hardware cost because the hardware cost overhead increases geometrically with the growth of the DIMM tree depth.
- **Initializing the Routing Table**: The routing Table has to be initialized by the initialization data at least once. The initialization data is decided by the location of the concerned T-DIMM in the DIMM tree, because the neighboring rank numbers of the each of T-DIMMs in the DIMM tree is different each other. Therefore, the number of initialization data should have to be same the number of T-DIMM in the DIMM tree, but the initialization data management policy has not considered. Furthermore, the access method for the routing table in the T-DIMM for initializing has never been discussed.
- **Limited Number of Ranks in the T-DIMM**: Each T-DIMM of the DIMM tree architecture needs to have only one rank number to use routing Table based on the rank number. The number of ranks is obviously one of influential factors for DIMM performance.

The paper proposes a new method that assigns identification (ID) into each of T-DIMMs, and routes the command based on T-DIMM ID without the routing table in order to overcome the drawbacks of using the routing Table.

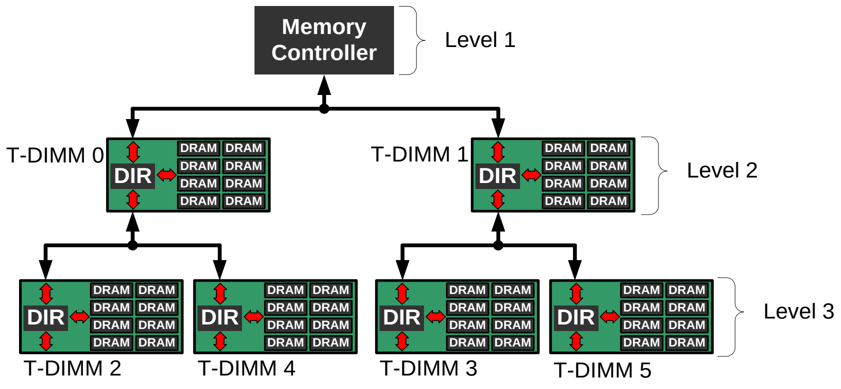


Figure 1. DIMM Tree Architecture

The remainder of this paper is organized as follows. Section II gives background for the latest memory management policies in the DIMM tree architecture and T-DIMM hardware architecture. Section III describes the proposed T-DIMM ID based command routing method. Section IV and V describes our experimental framework and result for the T-DIMM ID based command routing method. Section VI concludes this paper.

2. Background

In this section, we give a brief overview of the latest pre-proposed memory management policies in the DIMM tree architecture, and describe hardware architecture of the T-DIMM.

2.1. Memory Management Policies in the DIMM Tree Architecture

The DIMM tree architecture used the partitioned DIMM tree policy to reduce the memory access time when accessing the T-DIMM of level 3 and over. Partitioned DIMM tree classifies the DIMM tree into the fast partition and slow partition, as shown in Figure 2. The fast partition consists of the T-DIMMs in the fast access level that is able to fast access such as level 2, and the slow partition consists of the remaining T-DIMMs. The fast partition is used as a cache of the slow partition, which works as the actual main memory in the system. Also, data transfer method between two partitions is the direct DIMM to DIMM transfer without the supporting processor [8].

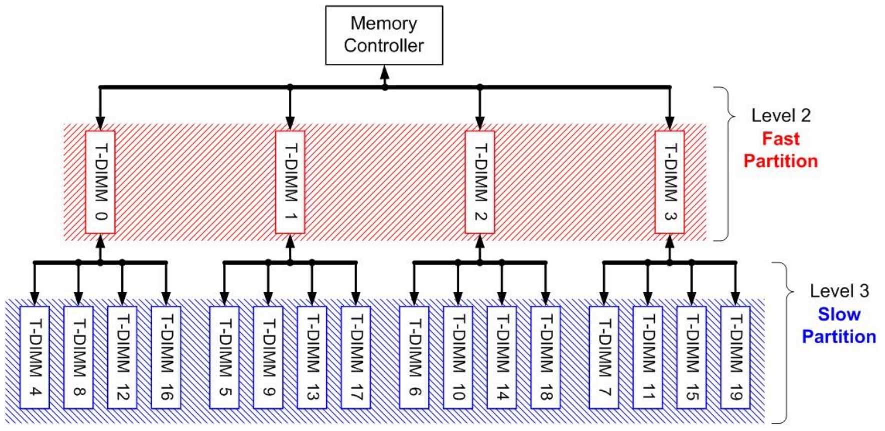


Figure 2. Fast and Slow Partition of the DIMM Tree Architecture

2.2. T-DIMM Hardware Architecture Overview

The T-DIMM needs to have two external channels to connect the upper level T-DIMM and lower level T-DIMMs. The previous studies about the DIMM tree architecture proposed the use of the multiband radio-frequency interconnect (MRF-I) to reduce T-DIMM pin overhead [9-10]. Since the MRF-I is the signal transmission technique based on amplitude shift keying (ASK), the MRF-I signals have not digital characteristics but sinusoidal characteristics. Accordingly, the MRF-I signals are converted into the digital signals to process the coming commands based on the MRF-I. The DIR in the T-DIMM performs such conversion function. The block diagram about the hardware architecture of the T-DIMM is as shown in Figure 3. The DIR consists of the Parent DIMM MRF-I Transceiver, the Child DIMM MRF-I Transceiver, the Data Rate Converter, and the Router. The Parent DIMM MRF-I Transceiver is used to interface with an upper level T-DIMM, and the Child T-DIMM MRF-I Transceiver is used to interface with lower level T-DIMMs. The Data Rate Converter is used to change data rate and bus width for internal channel that is used to access DRAM chips in the T-DIMM. The Router decides how to process the received commands.

3. Proposed Command Routing Method Based on the T-DIMM ID

The proposed method for the command routing assigns identification (ID) to each of T-DIMMs in the DIMM tree architecture, and routes received commands using this assigned T-DIMM ID instead of the routing Table. In this chapter, we should explain T-DIMM ID fields, an ID allocation method, and a command processing method.

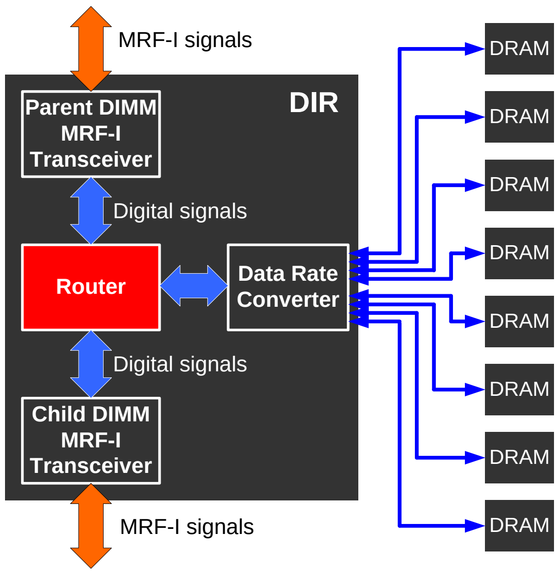


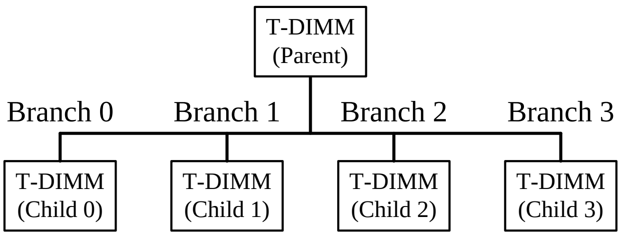
Figure 3. The Block Diagram of the T-DIMM

3.1. T-DIMM ID Structure

T-DIMM ID length is decided by the two factors, which are degree and depth of the DIMM tree architecture. Also, since T-DIMM ID will be mapped to a part of physical address for the physical address, T-DIMM IDs must be sequential and unique in order to provide a contiguous address space. T-DIMM ID needs enough length in order to allocate contiguous IDs to all of T-DIMMs in the DIMM tree architecture, and it is calculated by using follow equation (1), where N is the number of T-DIMMs in the DIMM tree architecture.

$$T\text{-}DIMM\ ID\ Length = \lceil log_2^{\ N} \rceil\ bits \qquad (1)$$

The proposed T-DIMM ID is divided into level and node fields. A level field value is the number of descendant nodes. The level field of T-DIMM ID can be used to estimate the level of concerned T-DIMM in the DIMM tree because T-DIMMs in the same level have the same number of descendant nodes. Consequently, the level fields are the symbolized value as each level, and therefore a level of concerned T-DIMM can be identified by checking the level field of its ID. The Node field of a T-DIMM is used to indicate its parent T-DIMM and its branch number beneath the parent. As one node field is made based on the node field value of the parent T-DIMM ID, which can reason the subordinate relationship through the node field value. To make a node field is classified according to the level with or without a parent T-DIMM. The node field of a T-DIMM without any parent such as level 2 T-DIMMs is decided according to its branch number as shown in Figure 4(a) and (b). The node field of a T-DIMM with a parent such as level 3 and 4 T-DIMMs is made by concatenating the node field of parent T-DIMM and its branch number as shown in Figure 4(b).



(a) Branch number in the DIMM tree

• Node Field of the Level without Parent T-DIMM

Node Field = Branch Number

• Node Field of the Level with Parent T-DIMM

Node Field = (Parent Node Field * Branch Factor) + Branch Number

(b) Node fields according to the level with parent or without

Figure 4. Node Fields According to the Level with Parent or Without

The complete T-DIMM ID consists of level and node fields as shown in Figure 5, and Figure 6 is a practical demonstration of the proposed method in the DIMM tree architecture with branch factor and depth four, shows contiguous allocated T-DIMM IDs from level 2 to level 4.

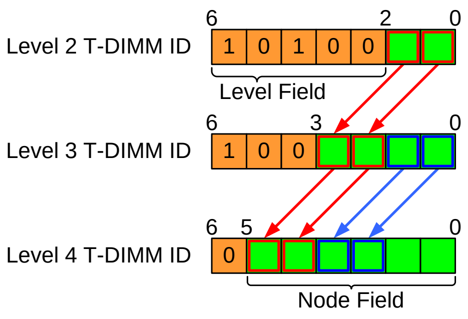


Figure 5. Structures of T-DIMM ID for DIMM Tree Branch Factor and Depth of Four

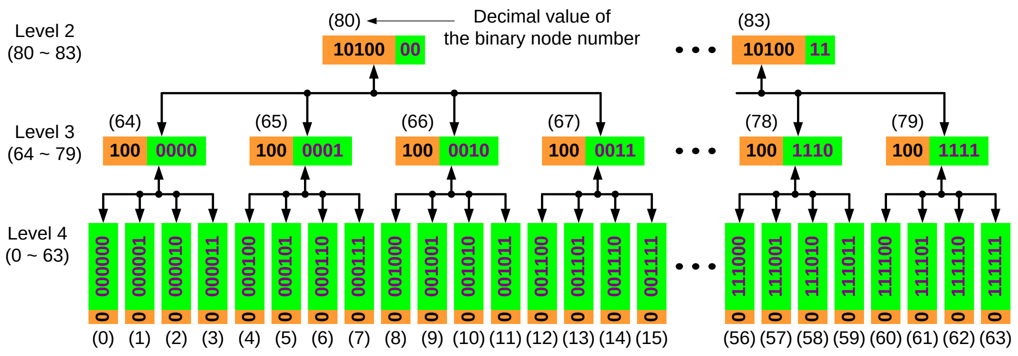


Figure 6. The Practical Demonstration for Allocating T-DIMM ID

3.2. T-DIMM ID Based Command Routing Method

A T-DIMM must decide what to do as soon as it receives a command. It can choose between aborting the command, forwarding the command to the lower level, or executing the command. A T-DIMM compares its T-DIMM ID and the T-DIMM ID field of the address of the received command. If they are equal, the T-DIMM executes the received command to access its DRAM modules. If not, it has to check whether the target T-DIMM of the command is a child by comparing its node field and that of the target T-DIMM. If the target T-DIMM is its child, it forwards the received command to the lower level. Otherwise, it aborts the command.

Analyzing the level field and node field for handling the received command could be implemented by various algorithms. In this paper, we propose one of methods as an example, which was used modeling the T-DIMM, for analyzing a level field and node field, sees Figure 7.

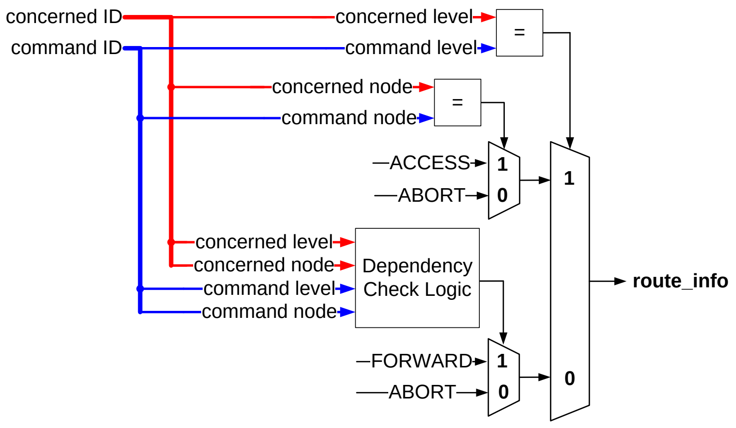


Figure 7. An Example of the Algorithm for Analyzing Level Fields and Node Fields from the Concerned T-DIMM ID and Received T-DIMM ID

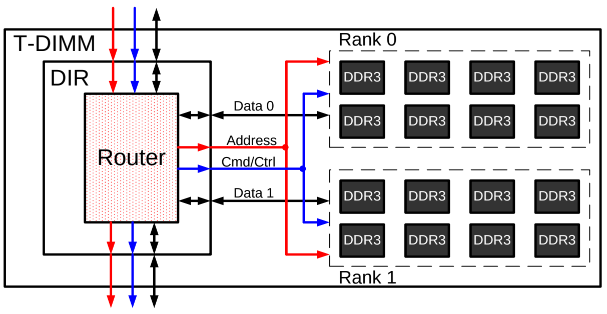


Figure 8. Block Diagram of the T-DIMM Modeling

4. Experiments

In this chapter, we explain the experiments for the function verification and hardware cost estimation of the proposed command routing method. The proposed T-DIMM and DIMM tree architecture are modeled with a register transfer level (RTL) using Verilog-HDL. Since DIMM tree with a branch factor and depth of four is modeled, there are 84 T-DIMMs in the modeled DIMM tree architecture. Also, the modeled DIMM tree architecture employs the partitioned DIMM tree and direct DIMM-to-DIMM transfer, and level 2 is used as a fast partition.

The block diagram of the modeled T-DIMM is illustrated in Figure 8. To simplify modeling, we assume that all of interfaces in the DIMM tree architecture are digital signals, and we omitted the blocks for supporting the MRF-I in the DIR, because the MRF-I does not affect our experiment results. The reference for setting up the DRAM parameters, which is adopted to model the T-DIMM, is a DDR3 datasheet [11-12], we reduced the memory capacity in the modeling because the experiments in this paper do not require large memory capacity. The detailed parameters about the T-DIMM modeling is given in Table 1.

Table 1. Parameters of the T-DIMM Modeling

Module	Parameter	Value
DDR3	Data Width	8 bits
	Burst Size	8
	Additional Latency	0 clocks
	Column Latency	6 clocks
	Capacity	16 Kbytes
T-DIMM	Data Width	64 bits
	Rank	2
	Capacity	256 Kbytes

Figure 9 shows the block diagram of the modeled Router. The Concerned T-DIMM ID is a register for saving the own (concerned T-DIMM) ID, and the Routing Block creates a routing information for processing the received command. The Command Routing State Machine handles the Parent Routing Buffer, Access Buffer, and Child Routing Buffer for routing the received command to the upper level, internal DRAMs, or lower level based on the received address, command, control, and routing information.

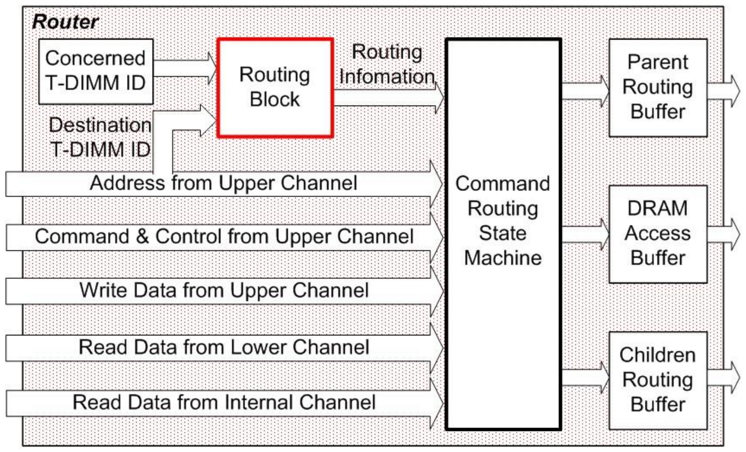

Router
Concerned T-DIMM ID
Routing Block
Routing Infomation
Destination T-DIMM ID
Address from Upper Channel
Command & Control from Upper Channel
Write Data from Upper Channel
Read Data from Lower Channel
Read Data from Internal Channel
Command Routing State Machine
Parent Routing Buffer
DRAM Access Buffer
Children Routing Buffer

Figure 9. Block Diagram of the Routing Modeling

Routing Block in the Router modeling, which handles a received command, is used to investigate hardware cost for between the proposed method and existing methods. The Routing Blocks of three types are modeled for analyzing required hardware cost about the proposed method and existing methods. One of the three Routing Block models is made based on T-DIMM ID, and is modeled as a hardware module with reference to the block diagram of the Figure 7. Other models are as existing methods based on routing Table; we implemented such Routing Blocks with two general methods for implementing the routing Table, because the detailed hardware architecture of the T-DIMM router based on a routing Table has not been still published. The general methods for implementing the routing Table are an Indexed Table method [13] and a content addressable memory (CAM) Table method [14]. The Indexed Table method manages only one Table for the rank numbers of all of T-DIMMs in the DIMM tree architecture as shown in Figure 10(a). The CAM table method manages an optimized Table for only requiring minimum number of the rank numbers to route the commands, and hardware logics for a fast Table searching technique, which is a similar CAM, are needed, otherwise indexed Table method, like as Figure 10(b).

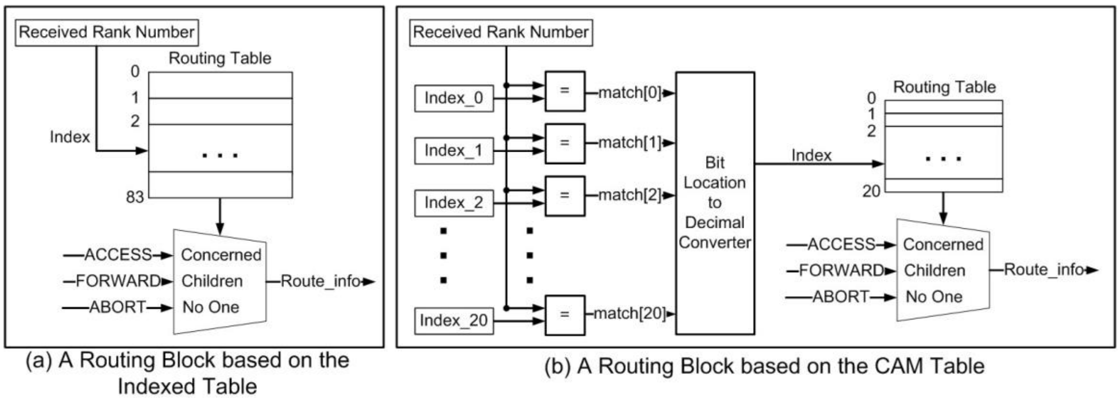


Figure 10. Routing Block Based on Existing Methods

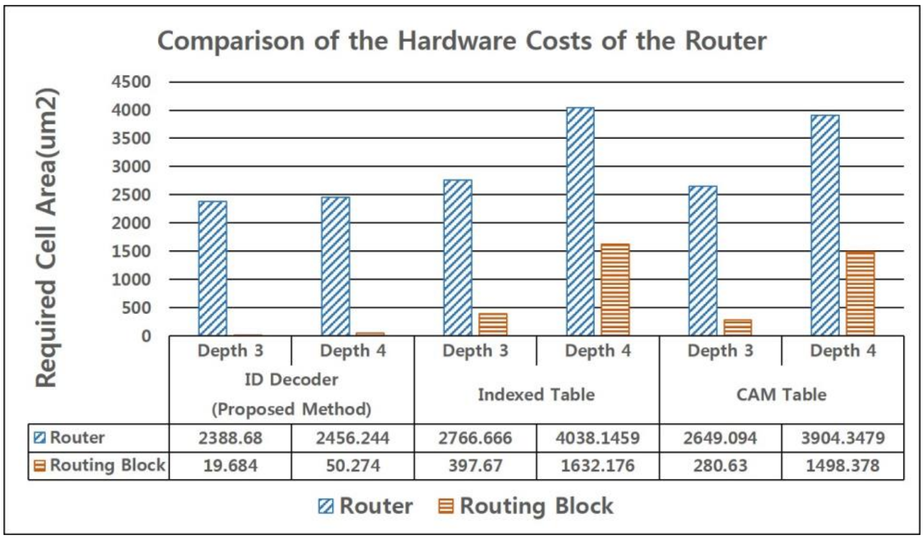
Through such modeling, we simulated and analyzed the hardware cost of the proposed method and existing method. The modeling is implemented with Verilog-HDL, and is synthesized in the front-end level with the Synopsys Design Complier [15] based on NanGate PDK45 Open Cell library [16]. In order to verify the hardware costs of the modeling, we analyzed the area reports from synthesize result.

5. Experiment Results

Figure 11 shows the required cell area of the T-DIMM Routers about the proposed method and existing methods. The experimental results are presented separately as for each of the Router and Routing Block.

The hardware cost of the proposed method is reduced drastically than existing command methods based on routing Table. With a DIMM tree of depth 3, the required cell area for Routing Block of the proposed method (ID decoder) is degraded on average 95% than the Routing Block of the Indexed Table, and Router is degraded 14% than the Router of the Indexed Table. In the comparison of the CAM Table, The required cell area for Routing Block and Router of the ID decoder is respectively degraded on average 93% and 10% than CAM Table.

With a DIMM tree of depth 4, the required cell area for Routing Block of the ID decoder is degraded on average 97% than the Routing Block of the Indexed Table, and Router is degraded 39% than the Router of the Indexed Table. In the comparison of the CAM Table, The required cell area for Routing Block and Router of the ID decoder is respectively degraded on average 97% and 37% than CAM Table.

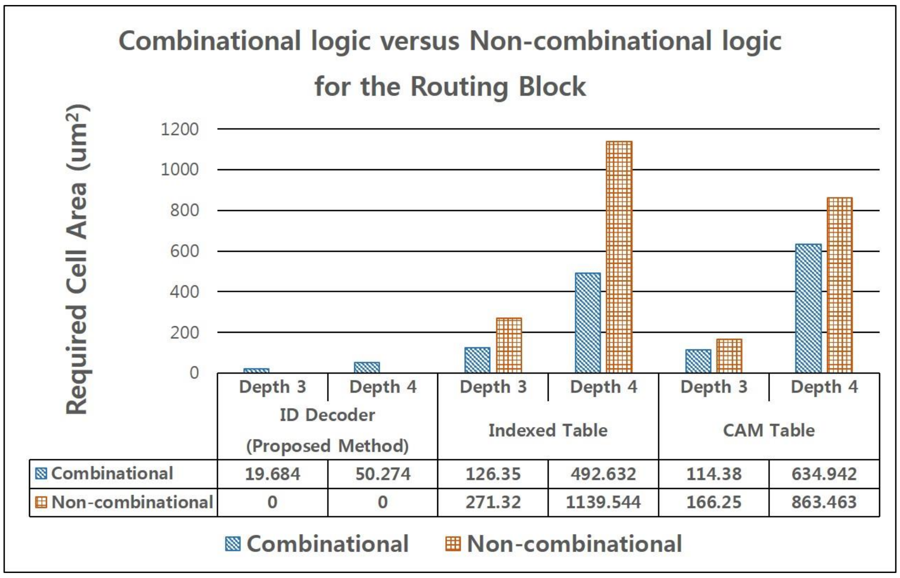


	Depth 3	Depth 4	Depth 3	Depth 4	Depth 3	Depth 4
	ID Decoder (Proposed Method)		Indexed Table		CAM Table	
▨ Router	2388.68	2456.244	2766.666	4038.1459	2649.094	3904.3479
☰ Routing Block	19.684	50.274	397.67	1632.176	280.63	1498.378

Figure 11. Comparison of the Hardware Costs of the Routers and Routing Blocks

The reason for these results is that since Indexed Table has only one large table storing all the rank numbers in the DIMM tree architecture, the required cell area for non-combinational logics is greater than the ID Decoder, see Figure 12. On the other hand, since CAM Table has an optimized table for minimum number of rank numbers in the DIMM tree architecture, the required cell area for non-combinational logics reduced than Indexed Table, but it needs additional combinational logics for fast search the table, see Figure 12. In contrast, ID Decoder, which is the proposed method, did not needs non-combinational logics, and was synthesized by only combinational logics.

In the last analysis, according to change the depth of DIMM tree from 3 to 4, see Figure 11, the required cell areas are shown that Routing Block and Router are around 155% and 3% of the increase of rate in the ID Decoder, whereas Indexed Table was about 310% and 46% of the increase of rate, and CAM Table was 434% and 47% of the increase of rate. Additionally, the performance of three Routers was same in our simulation, because each of three models for Routing Block can process a received command in one clock cycle. We used memory transaction traces by running bzip2, gromacs, hmmer, lbm, mcf, and milc in the SPEC CPU 2006 [17] benchmark in the SimpleScalar simulator [18].



	ID Decoder (Proposed Method)		Indexed Table		CAM Table	
	Depth 3	Depth 4	Depth 3	Depth 4	Depth 3	Depth 4
▨ Combinational	19.684	50.274	126.35	492.632	114.38	634.942
⊞ Non-combinational	0	0	271.32	1139.544	166.25	863.463

Figure 12. Combinational Logic Versus Non-Combinational Logic for the Routing Blocks

6. Conclusions

This paper proposes a new command routing method based on the T-DIMM ID without the routing table, in order to solve the problems caused by using the rank number based routing table in the previous T-DIMM architecture. Particularly, this paper presents how to allocate IDs to all the T-DIMMs in the DIMM tree, and proposes a command routing method using these T-DIMM IDs. The paper modeled and synthesized the proposed and previous methods to compare their hardware costs. The experimental results showed that our proposed method solves problems of the previous methods such as initializing the routing table and limitation in the number of ranks in one T-DIMM, and that the proposed method reduces hardware costs of the Routing Block and Router maximally by 97% and 37%, respectively, compared with those of the routing table based command routing methods. Finally, our future work is to study on practical implementation of the DIMM tree and T-DIMM.

ACKNOWLEDGEMENTS

This research was supported by Kyungpook National University Research Fund, 2013(2014). This investigation was financially supported by Semiconductor Industry Collaborative Project between Kyungpook National University and Samsung Electronics Co. Ltd.

References

[1]. B. Jacob, S. W. Ng and D. T. Wang, Editors, MEMORY SYSTEMS: Cache, DRAM, Disk, Morgan Kaufmann publishers, Burlington, **(2008).**
[2]. B. Ganesh, A. Jaleel, D. Wang and J. Jacob, Editors. Fully-Buffered DIMM Memory Architectures: Understanding Mechanisms, Overheads and Scaling. Proceedings of the 13th International Symposium on High Performance Computer Architecture, **(2007)** February 10-14, Phoenix, Arizona, USA.
[3]. K. Therdsteerasukdi, G. Byrn, J. Ir, G. Reinman, J. Cong and M. F. Chang, Editors, “The DIMM Tree Architecture: A High Bandwidth and Scalable Memory System”, Proceedings of IEEE 29th International Conference on Computer Design, **(2011)** October 6-9, Asheville, NC, USA.
[4]. E. Horowitz, S. Sahni, D. P. Mehta, Editor, “Fundamentals of Data Structures in C++ -2nd ed”, Silicon Press Publishers, New Jersey, **(2007).**
[5]. Z. Gao, J. Fan, Y. Jia and L. Zhang, Int. J. Security and its Applications, vol. 1, no. 2, **(2007).**
[6]. M. Zeynali, L. M. Khanli and A. Mollanejad, Int. J. Grid and Distributed Computing, vol. 2, no. 4, **(2009).**

[7]. D. Virmani, Int. J. Advanced Science and Technology, vol. 55, **(2013).**
[8]. K. Therdsteerasukdi, G. Byrn, J. Ir, G. Reinman, J. Cong and M. F. Chang, IEEE Trans. Emerg. Sel. Topics Circuits Syst., vol. 2, no. 2, **(2012).**
[9]. M. F. Chang, Z. Xu, J. Kim, J. Ko, Q. Gu and B. Lai, IEEE Trans. Electron Devices, vol. 52, no. 7, **(2005).**
[10]. J. Ko, J. Kim, Z. Xu, Q, Gu, C. Chien and M. F. Chang, Editors, "An RF/Baseband FDMA-Interconnect Transceiver for Reconfigurable Multiple Access Chip-to-Chip Communication", Proceeding of IEEE International Solid-State Circuits Conference Digest of Technical Paper. **(2005)** February 6-10, San Francisco, USA.
[11]. "DDR3 Datasheet, Micron", 4GB: x4, x8, x16 DDR3 SDRAM Features, **(2006).**
[12]. "DDR3 UDIMM Datasheet, Micron", 1GB, 2GB (x72, ECC, SR) 240-Pin DDR UDIMM Features, **(2008).**
[13]. K. J. Nesbit and J. E. Smith, "Micro", IEEE Micro, vol. 25, no. 1, **(2005).**
[14]. K. Pagiamtzis and A. Sheikholeslami, IEEE J. Solid-State Circuits, vol. 41, no. 3, **(2006).**
[15]. "Synopsys Design Complier", http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DesignCompiler/Pages/default.aspx.
[16]. "NanGate PDK45 Open Cell library", http://www.nangate.com/?page_id=2325.
[17]. J. L. Henning, SIGARCH Comput. Archit. News, vol. 34, no. 4, **(2006).**
[18]. T. Austin, L. Eric and D. Ernst, Computer, vol. 35, no. 2, **(2002).**

Authors

Young-Kyu Kim, is a Ph.D. student in the School of Electrical Engineering, Kyungpook National University at Daegu in Korea. He received the B.S degree in Department of Electronic Engineering in Gyeongju University in 2005 and M.S. degree in Electrical Engineering & Computer Science in Kyungpook National University in 2010.

His research interests are in the following areas: SoC (system on a chip), computer architecture, and multiprocessor system.

Byungin Moon, is currently a professor in the School of Electronics Engineering, Kyungpook National University at Daegu in Korea. He received the B.S. degree and M.S. degree in Electronic Engineering in Yonsei University in 1995 and 1997 respectively. He received a Ph.D. degree in Electrical & Electronic Engineering in the same university in 2002. He spent two years as a Senior Researcher in Hynix Semiconductor Inc., and also worked as a Senior Researcher in Yonsei University for one year.

His research interests are in the following areas: SoC, computer architecture, and vision processor.

* Corresponding Author: Byungin Moon.