CasHMC: A Cycle-Accurate Simulator for Hybrid Memory Cube

Dong-Ik Jeon, *Student Member, IEEE* and
Ki-Seok Chung, *Member, IEEE*

Abstract—3D-stacked DRAM has been actively studied to overcome the limits of conventional DRAM. The Hybrid Memory Cube (HMC) is a type of 3D-stacked DRAM that has drawn great attention because of its usability for server systems and processing-in-memory (PIM) architecture. Since HMC is not directly stacked on the processor die where the central processing units (CPUs) and graphic processing units (GPUs) are integrated, HMC has to be linked to other processor components through high speed serial links. Therefore, the communication bandwidth and latency should be carefully estimated to evaluate the performance of HMC. However, most existing HMC simulators employ only simple HMC modeling. In this paper, we propose a cycle-accurate simulator for hybrid memory cube called CasHMC. It provides a cycle-by-cycle simulation of every module in an HMC and generates analysis results including a bandwidth graph and statistical data. Furthermore, CasHMC is implemented in C++ as a single wrapped object that includes an HMC controller, communication links, and HMC memory. Instantiating this single wrapped object facilitates simultaneous simulation in parallel with other simulators that generate memory access patterns such as a processor simulator or a memory trace generator.

Index Terms—Memory control and access, memory design, modeling of computer architecture, simulation

1 Introduction

DYNAMIC random access memory (DRAM), which is widely employed as the main memory, has been steadily developed for several decades. As semiconductor manufacturing technology advances, DRAM capacity, bandwidth, and power consumption have all been improved. In addition, many performance enhancement techniques such as double data rate (DDR) I/O signaling, prefetching, posted column address strobe (CAS), and the power-down mode have been proposed. Despite all these efforts, DRAMs may cause a performance bottleneck and excessive power consumption. In order to overcome these concerns, 3D-stacked DRAM, which combines a few stacked DRAM dies using through silicon via (TSV), has been actively studied.

Emerging technologies based on the 3D-stacked DRAM are wide I/O, high bandwidth memory (HBM), and the hybrid memory cube (HMC). Wide I/O is suitable for mobile embedded systems because of the low power consumption due to a stacked DRAM with many I/O channels. HBM employs a 2.5D stacking architecture via an interposer (interposer stacking), and it mainly focuses on maximizing the bandwidth. HBM widens the data channel width in the interposer layer to overcome the limits of the data channel width when many DRAM chips are composed on a single board. Unlike Wide I/O and HBM, the HMC architecture is composed of stacked DRAM dies and a logic die at the bottom [1]. The logic die is used to control the DRAM. Therefore, a high capacity memory can be implemented by chaining several HMC devices, which is advantageous to be adopted in a server system. Moreover, since the logic die supports arithmetic and logic operations with internal or external memory data, HMC has been employed in the processing-in-memory (PIM) architecture.

• *The authors are with the Department of Electronic and Computer Engineering, Hanyang University, Seoul 04763, Korea.*
E-mail: estwingz@naver.com, kchung@hanyang.ac.kr.

Manuscript Accepted 19 May 2016received 21 June 2016; revised 12 Aug. 2016. Date of publication 15 Aug. 2016; date of current version 26 June 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/LCA.2016.2600601

Wide I/O and HBM are fundamentally similar to conventional DRAM architecture, which means that the same operating method as the conventional read/write memory request is employed. However, HMC is connected to the CPU or the GPU through high speed serial links with the SerDes interface, as shown in Fig. 1. Memory requests from a link are delivered to the corresponding vault controller through a crossbar switch. The vault controller re-arranges requests by a memory arbitration scheme just as a conventional memory controller does. Accordingly, a vault in an HMC is roughly equivalent to a channel in a traditional DRAM system, so is a vault controller to a memory controller, and so are DRAM dies to ranks. By utilizing this hierarchical resemblance, many existing HMC simulators have been built by modifying traditional DRAM simulators. However, existing simulators are not sufficiently accurate to simulate HMC-specific behaviors in accordance with the HMC specification [2]. Therefore, this paper introduces a cycle-accurate model of HMC, and a simulator called CasHMC is implemented with the model for exact and fast HMC performance analysis (CasHMC is available at https://github.com/estwings57/CasHMC). The proposed CasHMC supports unique features of the HMC, such as the high speed serial link, the packet-based interface, and the vault-based DRAM structure. Since CasHMC simulates all the HMC modules at a cycle-accurate level, it is capable of deriving precise simulation results and analyses.

2 Related Work

One of the most widely used DRAM simulators is DRAMSim2 [3]. DRAMSim2 employs a cycle-accurate model of a DDR2/3 memory system, which includes the DRAM storage, the memory controller, and the standard memory bus. Although DRAMSim2 is widely used in the DRAM simulation, due to the HMC's unique structure, it is not sufficiently accurate to simulate HMC.

Another simulator called BOBSim employs a model for a buffer-on-board (BOB) memory system [4]. The BOB memory system has an independent controller for its respective channels, and the BOB controller in a CPU is connected via high speed serial links. Since the overall BOB memory system structure is similar to an HMC, BOBSim was employed for the HMC simulation in [5]. However, because all of the links in the BOB memory system are fixed to their respective channels depending on the address mapping, it is not suitable for HMC simulation.

There is a simulator specialized for HMC called HMC-Sim [6]. This is an HMC simulation framework that provides the infrastructure to flexibly simulate one or more of the HMC devices attached to a processor core. HMC-Sim was developed specifically for the HMC 1.0 and 2.0 configurations. HMC-Sim supports most HMC features, such as multiple HMC chaining and a packet-based interface, but it is not based on a cycle-accurate model. That is, it is difficult to simulate the exact HMC behavior. Therefore, CasHMC is proposed with cycle-accurate HMC modeling.

3 CasHMC

Fig. 2 shows the overall architecture of CasHMC. Each box represents a C++ object class composing CasHMC. Since each updates its state every clock cycle, CasHMC is capable of simulating the HMC behavior at a cycle-accurate level. In addition, the object composition that is similar to the actual HMC hardware architecture leads to more meaningful analysis. The entire CasHMC is wrapped in an encapsulated object called CasHMCWrapper. This facilitates simultaneous simulation in parallel with other simulators that generate memory access patterns such as a processor simulator or a memory trace generator by instantiating the encapsulated CasHMCWrapper object. Therefore, CasHMC is highly useful in simulating HMC operations. CasHMC includes a trace

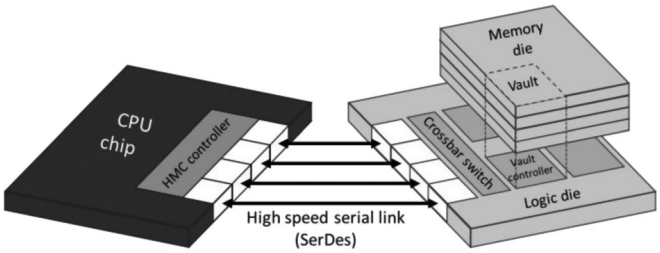

Fig. 1. The structure composed of CPU and HMC with a high speed serial link (SerDes) connection.

generator that generates read/write memory requests according to simulation parameters such as the memory utilization rate and the read/write ratio. If pre-generated memory request sequences are stored in a trace file, it can be loaded by the trace loader included in CasHMC.

As mentioned, packetized memory requests are delivered through high speed serial links with the SerDes interface. Since a high speed serial link for one way data transmission is employed, the link flow control plays an essential role. In order to control the link flow, the HMC specification defines flow packets such as Retry Pointer Return (PRET), Token Return (TRET), and Init Retry (IRTRY) [2]. These packets determine when a transmission takes place, how many packets will be transmitted, and whether a link retry sequence is to be initiated. The flow control by these flow packets is implemented in CasHMC, as shown in Fig. 3. The same component names as defined in the HMC specification are used in CasHMC [2]. CasHMC implements data request packets with various widths ranging from 16 bytes to 256 bytes, and it also implements all atomic commands for arithmetic, Boolean, comparison, and bitwise operations in accordance with the HMC specification [2]. In addition, CasHMC supports various link settings, which include the number of links, the lane speed, and the link width. Fig. 3 shows the detailed link master and slave structure and the flow packet control mechanism in CasHMC.

It should be noted that detecting communication errors on these serial links is crucial for high performance and reliability. Therefore, HMC detects a packet error by cyclic redundancy checks (Koopman CRC-32K) and sequence number checks (SEQ), and it enters the error abort mode when an error is detected [2]. CasHMC supports link error detection for packets with an automatic retry. In the case of Koopman CRC-32K, the determination of an error depends on the packet data. Since the CasHMC link delivers the packet on the same composition as the actual HMC, precise link error modeling is implemented.

In summary, CasHMC has various advantages that are not available in other existing HMC simulators.

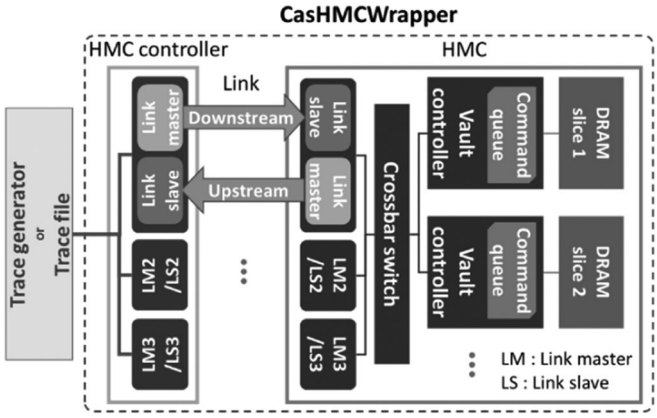

Fig. 2. Overview of the CasHMC structure.

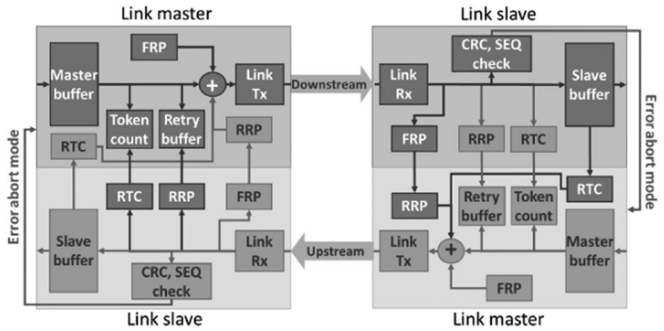

Fig. 3. Detailed link master and link slave composition of CasHMC in the downstream and upstream directions.

- CasHMC is the first cycle-accurate level HMC simulator.
- Memory access patterns are generated by three difference sources.
- The link flow control by flow packets is provided.
- All data requests/responses and atomic commands in accordance with the HMC specification 2.0 is implemented.
- Various link settings, which include the number of links, the lane speed, and the link width, are available.
- Packet errors can be detected by cyclic redundancy checks and sequence number checks.
- Various simulation result files will be generated (log, performance summary, and graph).

However, these features may cause longer simulation time. Selective activation of these features can be configured by two configuration files: SimConfig.h, DRAMConfig.h. Therefore, the users can select the operation mode of CasHMC in favor of either simulation accuracy or simulation speed.

4 Validation

It is important that the simulation truly matches the actual behavior of the memory system. To prove that it is actually cycle-accurate, we need cycle-by-cycle timing parameters and timing models from manufactures. Unlike DRAM simulators where major manufacturers provide the detailed information, unfortunately, such detailed HMC timing information is not available yet. Therefore, it is difficult to validate the cycle-level accuracy. Instead, CasHMC provides debug facility by two log files (debug, state) generated after each simulation run is over. The debug log file keeps track of all the debug information. Fig. 4a shows an example log file where the request flow, the link transmission, and the command issue are shown for each clock cycle. The state log file shows which request is ready to be transmitted in the respective class object, as shown in Fig. 4b. A memory transaction flow is investigated with considering latency by employing these two log files one by one for every clock cycle.

```
--------------[ CPU clk:59 / HMC clk:15 ]--------------
 (BUS)         [T6-Read] Down) SENDING transaction to HMC controller (HC)
   (LM_D0)     [F5-RD64] Down) Decreased TOKEN : 1 (remaining : 62/64)
   (LM_D0)     [F5-RD64] Down) RETRY BUFFER[1] (FRP : 2)
   (LK_D0)     [F5-RD64] Down) START transmission packet
--------------[ CPU clk:60 / HMC clk:16 ]--------------
 (HC)          [F6-RD64] Down) SENDING packet to link mater 1 (LM_D1)
   (LK_D0)     [F5-RD64] Down) DONE transmission packet
     (CQ_5-1) [C3-READ] Down) POP command from command queue
```

(a)

```
--------------[ CPU clk:59 / HMC clk:15 ]--------------
 (HC)          Down [T6-Read][ - ]...
   (LM_D0)     RTRY (1)[F5-RD64]
    (CQ_5-1)   Que [C3-READ][ - ]...
--------------[ CPU clk:60 / HMC clk:16 ]--------------
   (LM_D1)     Down [F6-RD64][ - ]...
   (LS_D1)     Down [F5-RD64][ - ]...
```

(b)

Fig. 4. Example log files after a CasHMC simulation run is over (a) debug (b) state.

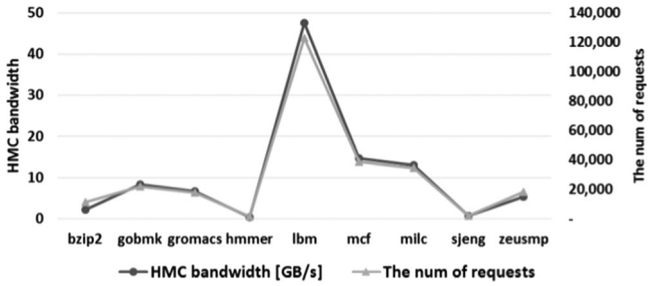

Fig. 5. The comparison between the number of memory requests and HMC bandwidth corresponding to SPEC CPU2006 benchmarks.

In order to validate the simulation accuracy, the various SPEC CPU2006 benchmarks [7] are executed by CasHMC. The trace files of benchmarks are extracted by a CPU simulator called SimpleScalar [8]. Fig. 5 shows the comparison between the number of memory requests in the trace file and the HMC bandwidth from the CasHMC simulation result corresponding to benchmarks. As shown in the figure, the number of memory requests is quite similar to the HMC bandwidth which is one of the results from CasHMC. Therefore, it can be said that CasHMC simulates the HMC operation truly to the memory access characteristic. Furthermore, all memory requests are represented by a transaction class object. An object of the class is generated by dynamic memory allocation, when CasHMC receives a memory request. Then, the allocated memory is freed, when the response is returned. That is, if CasHMC simulates all memory requests without missing any, memory leaks should not occur. An instrumentation framework for dynamic analysis tools called Valgrind was used to test whether there is any memory leaks [9]. It is reported that CasHMC has no memory leaks and any other errors. Thus, the simulation results from CasHMC is claimed to be quite reliable.

5 Simulation Results

After a CasHMC simulation run is over, four log files (debug, state, setting, result) are generated in the result folder. The debug and state file are the above mentioned log files. The setting file has all the parameter information on the simulation run. The result file shows the simulation results, which include the number of requests, reads, writes, flow packets, bandwidth, and average latency. These results are also provided per link as well. Therefore, this detailed log information on every clock cycle provides very useful information to accurately understand the HMC operation and analyze the HMC performance.

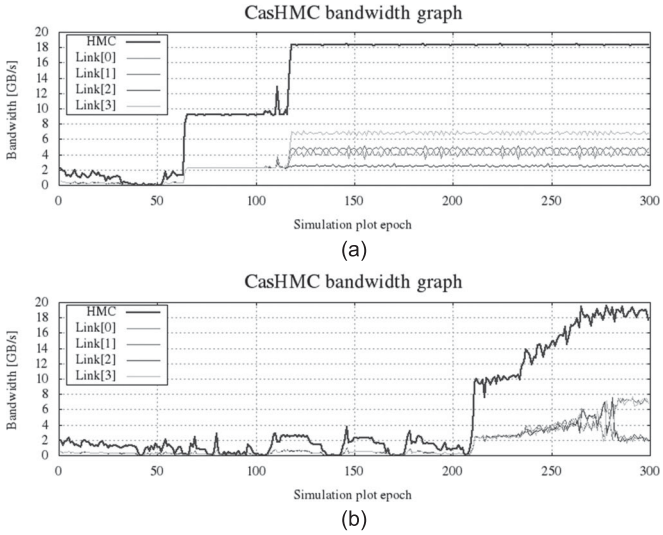

Fig. 6. Example bandwidth graph of the HMC and link by the generated by Gunplot graphing utility: (a) milc (b) zeusmp benchmark.

TABLE 1
Bandwidth and Transaction Latency Summary: (a) Milc (b) Zeusmp Benchmark

(a)

HMC Bandwidth [GB/s]	Request count	Flow Packet count	Transaction latency			
			Mean [ns]	StDev [ns]	Max [ns]	Min [ns]
13.06	34747	117336	46.97	16.57	129	27

(b)

HMC Bandwidth [GB/s]	Request count	Flow Packet count	Transaction latency			
			Mean [ns]	StDev [ns]	Max [ns]	Min [ns]
5.31	18479	96855	44.28	16.43	167	24

Additionally, CasHMC generates a Gnuplot script file and a graph data file in a graph folder when a simulation run is over. Gnuplot is a command-line driven graphing utility for Linux, MS Windows, and many other platforms [10]. If a Gnuplot script file is executed with the Gnuplot program, a bandwidth graph is generated. Fig. 6 shows examples of milc and zeusmp SPEC CPU2006 benchmark simulation results. As shown in the figure, the bandwidth change is derived from not only the total HMC, but also each individual link. Furthermore, the graph data file has bandwidth values in terms of the unit time defined as the *PLOT_SAMPLING* parameter. You may observe clearly different HMC behaviors depending on the benchmark program. Table 1 shows a partial summary of simulations of the two benchmark programs derived from the result log file. Therefore, it is possible to analyze the simulation results easily and accurately.

6 Conclusions and Future Work

DRAM has been constantly improved for several decades, but the development is often claimed to have reached an impasse due to the limit of semiconductor process and circuit design technology. In order to overcome this limit, 3D-stacked DRAM has drawn much attention. HMC is a promising 3D-stacked DRAM technique to implement a large-scale memory block and processing-in-memory. In this paper, we propose an HMC simulator called CasHMC. The proposed CasHMC simulates the cycle-accurate behavior of the HMC with all data requests/responses and atomic commands which cover HMC specification 1.0 and 2.0. CasHMC generates various log files to keep track of the simulation operations with text and graphical result files.

Several advanced features of the simulator are currently being implemented to support power consumption estimation and HMC chained topology. In particular, high speed serial links consume a significant amount of power. Therefore, power modeling of HMC will be useful for power-aware HMC memory scheduling.

Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (R7119-16-1009, Development of Intelligent Semiconductor Core Technologies for IoT Devices based on Harvest Energy). Ki-Seok Chung is the corresponding author.

References

[1] J. Jeddeloh and B. Keeth, "Hybrid memory cube new DRAM architecture increases density and performance," *Symp. VLSI Technol.*, Jun. 2012, pp. 87–88.

[2] Hybrid Memory Cube Consortium (HMCC), Hybrid Memory Cube Specification 2.1, 2016. [Online]. Available: http://www.hybridmemorycube.org/files/SiteDownloads/HMC-30G-VSR_HMCC_Specification_Rev2.1_20151105.pdf
[3] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A cycle accurate memory system simulator," *IEEE Comput. Archit. Lett.*, vol. 10, no. 1, pp. 16–19, Jan.-Jun. 2011.
[4] E. Cooper-Balis, P. Rosenfeld, and B. Jacob, "Buffer-on-board memory system," in *Proc. Int. Symp. Comput. Archit.*, Jun. 2012, pp. 392–403.
[5] M. J. Khurshid and M. Lipasti, "Data compression for thermal mitigation in the hybrid memory cube," in *Proc. Int. Conf. Comput. Des.*, Oct. 2013, pp. 185–192.
[6] J. D. Leidel and Y. Chen, "HMC-Sim: A simulation framework for hybrid memory cube devices," in *Proc. Int. Parallel Distrib. Process. Symp. Workshops*, 2014, pp. 1465–1474.
[7] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Comput. Archit. News*, vol. 34, pp. 1–17, Sep. 2006.
[8] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An infrastructure for computer system modeling," *IEEE Comput.*, vol. 35, no. 2, pp. 59–67, Feb. 2002.
[9] N. Nethercote and J. Seward, "Valgrind: A framework for heavyweight dynamic binary instrumentation," in *Proc. Conf. Program. Lang. Des. Implementation*, vol. 42, pp. 89–100, Jun. 2007.
[10] T. Williams, "Gnuplot," 2016. [Online]. Available: http://www.gnuplot.info