HMC and DDR Performance Trade-offs

Paulo C. Santos(✉), Marco A. Z. Alves, and Luigi Carro

Informatics Institute - Federal University of Rio Grande Do Sul, Porto Alegre, Brazil
{pcssjunior,mazalves,carro}@inf.ufrgs.br

Abstract. The evolution of main memories, from SDR to the current DDR, presents multiple technological breakthroughs, but still far from the requirements of the processors. With the advent of Hybrid Memory Cube (HMC), a promise of high bandwidth with low energy consumption and less area may provide better efficiency than the traditional DDR modules. This is especially attractive for embedded systems. In this paper, we perform a comprehensive performance comparison between HMC and DDR memories, to understand the capabilities and limitations of both. Simulation results running SPEC-CPU2006 and SPEC-OMP2001 benchmarks show that applications with low memory pressure behave similarly with HMC or DDR. We make the new observation that HMC performs better than DDR specially for applications with a high memory pressure and low spatial data locality. However, for applications with a streaming behavior, commonly present in the embedded system domain, our experiments show that current HMC row-buffer specifications do not take advantage of the spatial locality present in those applications.

Keywords: HMC · DDR · Main memory · Performance evaluation

1 Introduction

Due to increasing requirements from embedded applications, the architectures of embedded systems are becoming similar to high performance computers in the sense that performance techniques are being adapted to this new context. Following this trend embedded systems are commonly applying Double Data Rate (DDR) memories. The evolution of DDR systems brought benefits in terms of performance, while keeping power consumption levels constant. However, for the new Hybrid Memory Cubes (HMCs), the trade-off between energy consumption and performance is more interesting. The industry is predicting that HMC will provide both higher performance and considerably lower energy consumption in comparison to the current memory systems [1,20]. However, simulation platforms and evaluation experiments are required to understand the new trade-offs.

In this paper, we aim to understand the performance difference between the HMC and traditional DDR 3 memories. In addition, we intend to evaluate

P. C. Santos—We acknowledge the support of CNPq and CAPES.

M. Götz et al. (Eds.): IESS 2015, IFIP AICT 523, pp. 159–171, 2017.
https://doi.org/10.1007/978-3-319-90023-0_13

what type of application exploits more efficiently each memory architecture. We also adapt a cycle-accurate simulator to model both memory systems to perform detailed experiments, which are capable of explaining the sources of performance differences between these two memory systems.

The main contributions of this paper are the following:

HMC simulator: We extended a cycle-accurate simulator to implement a detailed HMC model, considering the internal vaults, the DRAM signal latencies and the link bandwidth.

DDR and HMC comparison: Using 29 single-threaded and 7 multi-threaded benchmark applications, we performed an analysis comparing HMC to DDR 3 memories, in order to understand the limits imposed by the number of DDR 3 channels and HMC links, as well as the characteristics that allow each application to benefit most from each memory system.

Application behavior correlation: In our experiments we show, as expected, that applications with low memory pressure keep the performance at the same level when changing between HMC and DDR 3 memories. We make the new observation that applications with high memory pressure (i.e., memory pressure higher than 0.5 GB/s) with low spatial memory locality benefits more from the HMC, because of the closed-row policy and the high bank parallelism. However, applications with a high memory locality performs better with DDR 3 memories mainly because of its 8 KB row buffer.

We evaluate the HMC and DDR 3 memories modeling an Atom-inspired embedded system consisting of 8 cores with 32 KB L1 and 256 KB L2 caches. Simulations executing the SPEC-CPU2006 and SPEC-OMP2001 benchmarks show that HMC with 4 links improves the performance by up to 27% for SPEC-CPU2006 and up to 109% for SPEC-OMP2001 when compared to the DDR 3 with 4 channels, and improves up to 27% and 50% respectively when compared to the DDR 3 with 8 channels. Such improvements are observed for those applications with high memory pressure and non-contiguous data access behavior. However, for applications with high memory pressure and a contiguous data access behavior, DDR 3 with 8 channels performed up to 26% better.

2 Technological Constraints of Memory Designs

In this section, we present the architecture of DDR devices and the HMC internals. For the rest of this paper, we focus on the DDR 3 and HMC specification version 2.0. The DDR and the HMC systems are presented at a level of abstraction that is sufficient to understand the terminology and key concepts of this paper. For a detailed description, we refer the reader to [5,6,9].

2.1 DRAM and DDR Architectures

Traditional main memory modules are formed by multiple devices that act in a coordinated way [9]. The highest memory structure level is the module, which

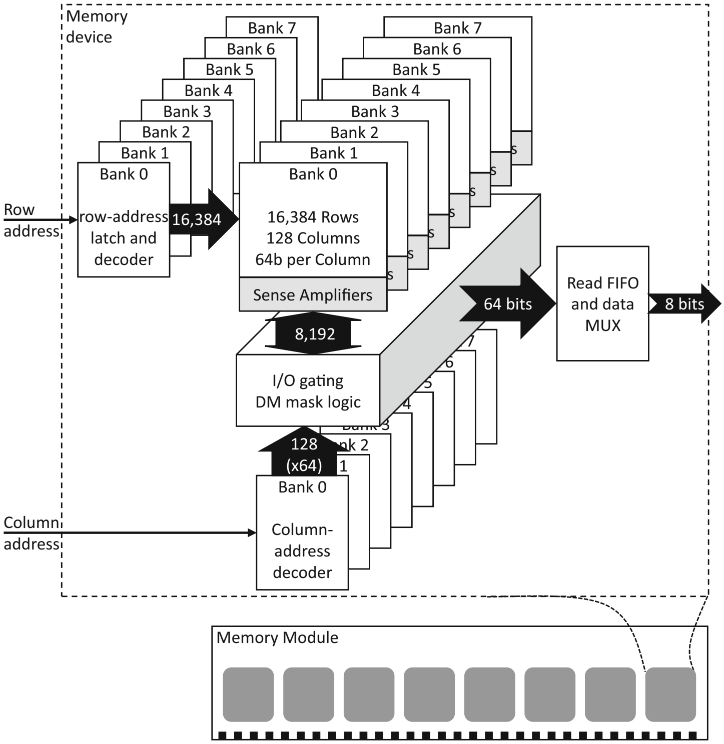

Fig. 1. DDR 3 × 8 functional block diagram of a single device. Adapted from [12].

consists of a set of devices. A module may have multiple ranks, each rank consisting of multiple devices, which will operate in synchrony. The devices are composed of a set of banks, and all the devices in a given rank react to an operation signal, always operating in the same bank for a given signal. These banks are composed of sub-arrays, formed by rows that are accessed per column. Figure 1 shows a basic schematic of the DDR 3 × 8 device.

The DRAM protocol manages these arrays using these 5 basic, simplified operations: *precharge* (prepares the arrays and sense amplifiers to read a new row), *row access strobe* (reads a specific row using the sense amplifiers into a SRAM row buffer, with 1 buffer for each bank), *column access strobe* (bursts data of a specific column of the row buffer from the DRAM devices to the bus), *column write* (receives data from the bus and overwrites the addressed column of an opened row) and *refresh* (refresh capacitor charges a row, usually done automatically by each device).

Since processors have been increasing their throughput demand, DRAM-based memories have evolved trying to meet the requirements of modern processors. DDR memories emerged as a major technological breakthrough, providing the ability of transmitting data at both clock edges. However, its evolutions as

DDR 2, DDR 3, DDR 4 and so forth generally increased the I/O frequency by increasing the data burst capability and bus operating frequency. However, the organization of a DDR device in all versions experienced few architectural modifications. Despite these advancements in memory technology, the operating frequency of the basic devices to a certain data width is limited, providing a lower throughput than what is required by modern processors. Thus, besides the burst technique, sets of devices are deployed in a module to increase parallelism and increase data throughput.

To achieve high bandwidth using the DDR memories, the multi-channel technique is widely adopted. This approach allows accessing multiple memory modules in parallel and independently, enabling data transfers from more than one row buffer at same time. Thus, if a system requires large bandwidth, for example, four channels may be required, using 256 wires in a half-duplex fashion. However, such a large number of wires in a bus is prohibitive for embedded systems, mainly because of area and power consumption constraints. To control all the devices of each channel, the memory controller must issue all the signals to the devices's bank. This control also increases with the number of channels [21].

2.2 HMC Architecture

Breaking the traditional way of DDR evolution, HMC is not concerned with increasing I/O frequency by using burst techniques. Despite using the same DRAM cell and its restrict accessing times, HMC changes the paradigm by hiding its device latencies internally, mostly with the aid of 3D integration and Through-Silicon Via (TSV) technology [13], which enables the integration of a massive bank parallelism [5,10,15].

Basically HMC memories are composed of up to 8 layers of DRAM memory and one logical layer per vault, all integrated in the same device. Figure 2 illustrates the main HMC architectural details. The three main components inside the HMC are the following:

Memory vaults: HMC memories store data on contiguous row buffers, interleaving throw the memory vaults and then to memory banks inside the vaults. HMC may have up to 32 vaults, each one can be composed by up to 16 banks, where each row in the bank has up to 256 bytes. In theory, the HMC can fetch data from 32 different banks (one per vault) in parallel and copy it to the internal read buffers. Only after the data is ready in the read buffers, the links may send it to the processor's memory controller.

Memory controller: The 3D integration technology enabled the integration of memory and logic in the same chip on different layers. Thus, an HMC has a dedicated memory controller attached to each vault, providing great data access parallelism. In this way, the processor's memory controller can be simplified to work with simple data request commands, reducing its complexity, area and energy consumption.

Serial links: Unlike DDR 3 memories, which transmit 64 bits per channel, HMC memory uses serialization, to transmit data through 16 full-duplex lanes

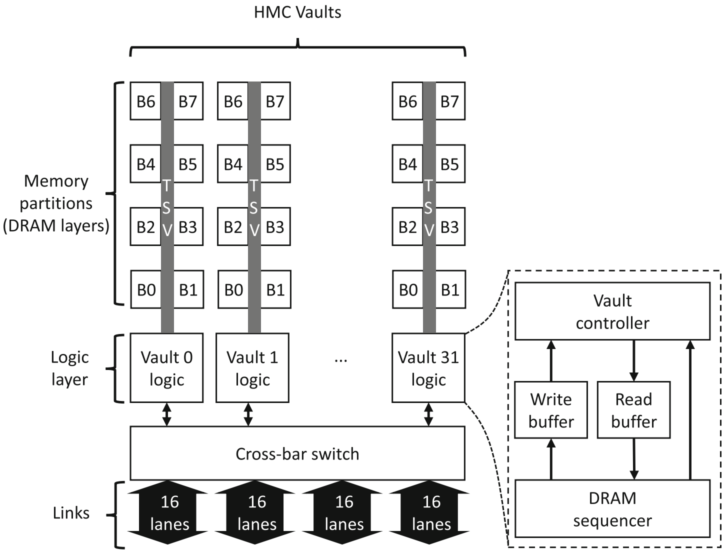

Fig. 2. HMC block diagram formed by 32 vaults with 8 banks each. Adapted from [5].

per link (each lane is a pair of differential signal lines). However, each link is not strongly attached to a specific portion of the memory, which means that any link can be used to transfer data from/to any HMC vault. The adoption of this communication technique leads to a smaller area for buses. Moreover, these links are capable of achieving higher frequencies with less interference during the transmissions [5,19].

As mentioned, other differences between HMC and DDR memories are the row buffer size and the bank parallelism. Typically, DDR memories have row buffers of 8192 bytes per bank (split among the devices), while HMC specifies row buffers of up to 256 bytes per bank. Meanwhile, the DDR modules provides only 8 memory banks which can act in parallel, while the HMC has up to 512 DRAM banks.

Previous work mentions that these architectural differences in the HMC architecture may result in up to 70% less energy consumption than DDR3-1333, and a 15 times theoretical speedup of the system [5,10,15]. However, it is not clear if all applications can benefit from the HMC. In the remainder of this paper, we present the methodology for our experiments and the results showing the most important aspects of an application to perform better in the HMC compared to DDR memories.

3 Evaluation Methodology

This section presents the simulation details, the application kernels and the evaluation methodology, showing how we compare our mechanism to the baseline embedded system and previous work.

3.1 Modeling DDR and HMC Simulation

To evaluate DDR and HMC memories, we used an in-house cycle-accurate simulator [2,3]. Simulator of Non-Uniform Cache Architectures (SiNUCA) is a trace-driven simulator, thus it executes traces generated on a real machine with a real workload without the influence from the OS or other processes. The traces are simulated in a cycle-accurate way, where each component is modeled to execute its operations on a clock cycle basis. SiNUCA currently focuses on the x86_32 and x86_64 architectures.

SiNUCA originally offered support only for modeling DDRx memories with an open-row policy. However, few modifications in the source code were necessary to model HMC, due to SiNUCA's high parameterization. To model a HMC, the memory controllers are used as HMC controllers, which are located inside the HMC device. The channels can be configured to act as memory vaults. However, changes were made to provide support on closed-row policy and new HMC instructions. Although HMC-aware compilers are not available yet, synthetic codes can be evaluated with this version of SiNUCA.

3.2 Configuration Parameters and Workload

The simulation parameters are inspired by Intel's Atom processor with the Silvermont Out-of-Order (OoO) micro-architecture [8]. Table 1 shows the simulation

Table 1. Baseline system configuration.

OoO Execution Cores - 2 GHz; 8 cores; Front-end 2-wide; 16 B fetch block size
14 stages (3-fetch, 3-decode, 3-rename, 2-dispatch, 3-commit);
24-entry fetch buffer, 32-entry decode buffer, 32-entry ROB; 16-entry BOB;
INT: 2-alu, 1-mul. and 1-div.; FP: 1-alu, 1-mul. and 1-div. (1-3-20; 5-5-20 cycle);
1-load and 1-store functional units (1-1 cycle); MOB entries: 10-read and 10-write;

Branch Predictor - 1 branch per fetch; 4 K-entry 4-way set-associative BTB;
Two-Level PAs predictor; 16 K-entry BHT, 2-bits prediction;

L1 Data + Inst. Cache - 32 KB, 8-way, 2-cycle; 64 bytes line; LRU policy;
MSHR entries: 10-request, 8-write-back; Stride Prefetcher: 1-degree, 16-strides table;

L2 Cache - 256 KB shared for every 2 cores; 8-way, 4-cycle; 64 bytes line; LRU policy;
MSHR entries: 10-request, 6-write-back; Inclusive LLC; MOESI coherence;
Stream Prefetcher: 2-degree, 16 prefetch distance, 32-streams;

Low Power DDR3-1600 Controller and Interconnection - Bi-directional ring, 1∼8-channels; 8 LP-DRAM banks, 8 KB row buffer per bank (1 KB per device), 8 burst length; Open-row first policy; CAS, RP, RCD, RAS and CWD latency (12-17-14-34-6 cycles);
HMC Module and Interconnection - Bi-directional ring, 1∼4-links @ 8GHz; 32 Vaults, 16 LP-DRAM banks per Vault @ 800 MHz, 256 B row buffer per bank, 2 burst length; Closed-row policy; CAS, RP, RCD, RAS and CWD latency (12-17-14-34-6 cycles);

parameters used for our tests. The Silvermont micro-architecture only supports 2 memory channels. In order to build a possible future scenario for comparison, we also extrapolate the baseline configuration with up to 8 memory channels. We apply the same extrapolation idea to the HMC, in order to evaluate the influence of the number of links to the performance.

As the workloads for our experiments, we chose the 29 serial applications from the SPEC-CPU2006 [7] and 7 parallel applications from the SPEC-OMP2001 [18] benchmark suite. The SPEC-CPU2006 benchmark suites (integer and floating point) were executed using the *reference* input set, executing a representative slice of 200 million instructions selected by PinPoints [14]. The SPEC-OMP2001 benchmarks were executed using the *reference* input set as well, executing up to one time step from its parallel region.

4 Experimental Results

This section presents the results for SPEC-CPU2006 and SPEC-OMP2001 benchmark suites when simulating HMC and DDR memories.

4.1 SPEC-CPU2006 Results

The first result regarding the SPEC-CPU2006 benchmark suite shows the average performance when executing all the applications for each one of the systems, with DDR 3 varying the number of channels and the HMC varying the number of links. Figure 3 presents the speedup results over the DDR 3 with 1 channel.

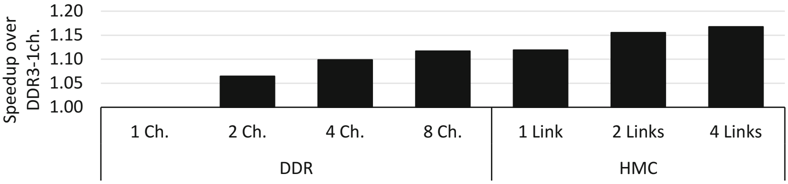

Fig. 3. Performance results for SPEC-CPU2006.

Memory Usage and Pressure: In order to understand the behavior of the different applications executing on HMC 4 links and DDR 3 8 channels, we correlate performance compared to the memory footprint and the memory pressure.

Figure 4 presents the SPEC-CPU2006 applications sorted by their performance. It shows the speedup of the HMC over the DDR 3, also showing in the secondary axis the amount of requests per second (pressure) the HMC serviced on average. On the top of the figure, the memory footprint is presented.

The plot shows some performance difference between DDR 3 and HMC only for those applications with pressure higher than 0.50 GB/s. This also correlates

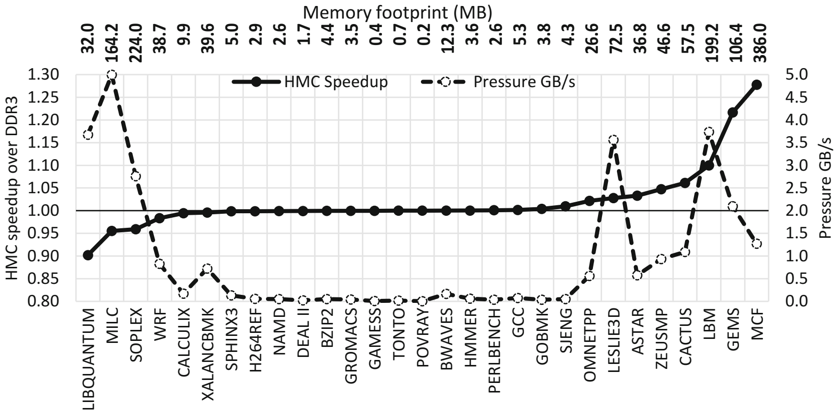

Fig. 4. HMC speedup over DDR 3 and the memory pressure (GB/s) for SPEC-CPU2006 applications.

with the memory footprint, where applications with more than 32 MB tends to have some performance difference between DDR 3 and HMC.

Both metrics show insignificant performance change for applications with low memory footprint and consequently low memory pressure. However, both metrics (memory footprint and pressure) cannot explain alone if a specific application will benefit or not from the HMC.

Memory Contiguity: Selecting only the SPEC-CPU2006 benchmarks with pressure higher than 0.50 GB/s, we obtained the list of applications with a reasonable memory pressure. For those applications, Fig. 5 correlates the HMC speedup with the memory access contiguity observed. The contiguity was

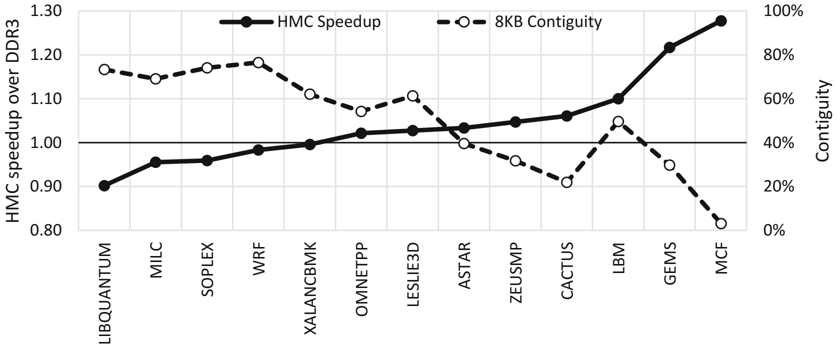

Fig. 5. HMC speedup and contiguity results for SPEC-CPU2006.

obtained for the 8 KB row buffer hit ratio of the DDR 3. Thus, the higher contiguity indicates that more contiguous accesses happened in a short period of time.

The main aspects from the DRAM architecture that influence the performance of contiguous accesses are the row buffer size, and the row buffer policy. In the case of the DDR 3, the row buffer contains 8 KB of contiguous data, while for HMC it holds only 256 B. Regarding the row buffer policy, the DDR 3 usually adopts the open-row policy, while the HMC specification describes the use of closed-row policy.

The HMC uses smaller row buffers compared to DDR 3 mainly to reduce the energy DRAM array consumption while it also increases the parallelism between the vaults. Open-row policies makes more sense with large row buffers. Considering that smaller row buffers will service less cache misses, the closed-row policy will close the row buffer as soon as the actual request is serviced. Performing the early row precharge command improves the performance for future accesses to different rows, while it can hurt performance if future accesses map to the recently closed row.

In our experiments, we implemented a smart closed-row policy, which identifies if multiple requests inside the read/write buffer map to the same row, and just close the actual row after all the requests have been serviced. However, even with such scheme, the performance is reduced when contiguous accesses happen.

4.2 SPEC-OMP2001 Results

In order to understand if scenarios with higher memory pressure would change our conclusions, we evaluated the multi-threaded applications from SPEC-OMP2001. Figure 6 presents the results when executing the applications with a different number of threads. The speedup results are normalized to the DDR 3 with 1 channel.

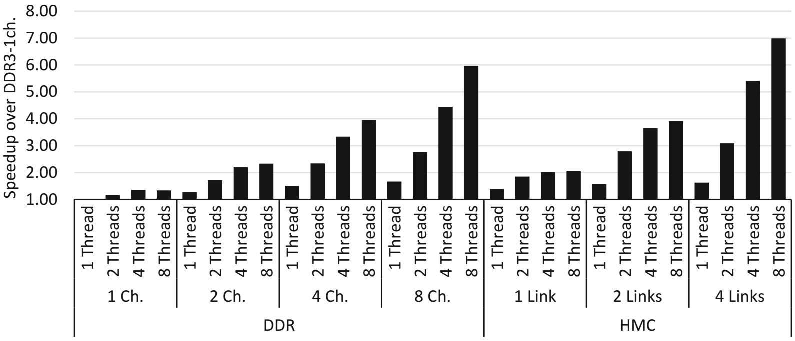

Fig. 6. Performance results for SPEC-OMP2001.

We can observe that HMC with a single link performed 50% better than DDR 3 with a single channel. Comparisons considering the maximum commercial products, HMC with 4 links and DDR 3 with 4 channels, show and average speedup of 1.75 for HMC. Comparing the best version of the DDR 3 and the HMC, we can observe that HMC performed 15% better.

Memory Contiguity: To evaluate the speedup change between the HMC and DDR memories, we took the DDR 3 with 8 channels and HMC with 4 links execution cycles in order to calculate the speedup between the HMC and its counterpart with DDR. Figure 7 presents the results with different numbers of threads, showing the HMC speedup for each application, compared to the same application with same amount of threads using DDR. In order to observe how contiguous the applications from the SPEC-OMP2001 benchmark suite are, the figure also presents in the secondary axis the contiguity ratio (8 KB row buffer hit ratio) for each application with different numbers of threads.

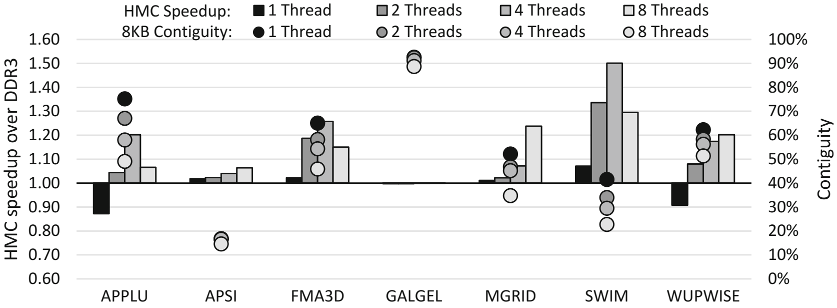

Fig. 7. HMC speedup over DDR 3 and the contiguity ratio for SPEC-OMP2001 applications.

We can observe for SPEC-OMP2001 the same behavior reported previously, that HMC can perform better than DDR for applications with lower contiguity. In this case, the number of parallel threads influences the contiguity of the accesses arriving to the main memory. Another factor that influences the performance of the multi-threading applications is the higher memory pressure present in the main memory.

4.3 Summary of Evaluations

As we could observe in the results with single and multi-threaded applications, the performance improvements when using HMC are for the applications with high memory pressure and low spatial data locality. For those applications with low memory pressure, the performance differences between HMC and DDR memories are negligible. Due to its smaller row buffers and closed-row policy, the HMC can hurt the performance of applications with contiguous data access behavior.

We can observe that when executing parallel applications, the spatial locality of the addresses arriving to the main memory reduces. It can be explained by the domain division per thread, where each thread tends to work in a different range of addresses. Moreover, with multi-threads generating requests, the memory pressure rises leading to a better exploration of the multiple HMC vaults.

In our experiments, we showed that applications with low memory pressure do not benefit from a different memory architecture. In this sense, low-end systems with very low performance processors may not benefit from HMC memories. The same can be said for streaming applications with high spatial locality, which benefit more from the DDR row buffer size. Furthermore, for systems with higher processing capabilities, the HMC can become more interesting when executing multi-threaded or multi-programmed workloads, which may create enough memory pressure with low spatial locality.

5 Related Work

The work presented in [16,17] verifies the maximum HMC memory bandwidth achievable, evaluating different high performance computing systems. The authors used the HMC-Sim simulator [11] varying HMC configurations coupled to a 16-core x86 processor. The authors only performed the experiments based on the HMC specification version 1.0 [4], which states a maximum theoretical bandwidth of 160 GB/s through 256 memory banks.

Fujitsu XIfx [20] introduces the usage of 8 HMC memories coupled to a set of SPARC64 processors. The work shows the performance achieved when HMC memory is adopted replacing DDR 3 memories. The processor coupled with HMC memories is 3.5 times faster on average, than previous DDR 3 memories with 4 channels. Due to the great reduction of the main memory bottleneck with the adoption of HMC, processor performance is dramatically expanded.

The HMC-Sim [11] is a cycle accurate simulator for HMC memories only, not modeling the cache hierarchy, interconnections and the processor. We choose to adapt the SiNUCA [3] memory model to support HMC, due to its capabilities of simulating the full system. Thus, we provide another tool capable to model HMC, while we can compare the findings with the other simulator.

Previous work explore only high performance processors, leaving it unclear whether the same behavior presents itself in the embedded system domain. In this work, we model an embedded system environment comparing the HMC 2.0 specification which supports up to 512 DRAM banks. In our evaluations we show the performance difference between HMC (with 1∼4 links) and DDR 3 (with 1∼8 channels), explaining the application behavior behind the performance results.

6 Conclusions and Future Work

Following the trend to move embedded systems closer to the high performance system domain, several high-end embedded systems are adapting multiple high performance mechanisms, such as DDR memories. However, the new HMC memories lack evaluations regarding their trade-offs when compared to DDR memories, especially in the embedded domain.

In this paper, we present a performance evaluation comparing HMC and DDR memories, in order to understand the possible speedup scenarios of this new memory architecture. We point the new finding that applications with streaming behavior that have high memory access locality perform better on DDR 3 memories, while sparse accesses are better serviced by the HMC. However, applications with low memory pressure keep the same performance in both systems.

As future work, we intend to extend the evaluation for the energy consumption domain. We also plan stress different processor parameters that most affect the memory pressure, such as the number of entries in the Memory Order Buffer (MOB), and cache Miss-Status Handling Registerss (MSHRs), as well as memory disambiguation techniques.

References

1. Altera: Hybrid memory cube controller IP core user guide (2015). https://www.altera.com/solutions/technology.html
2. Alves, M.A.Z.: Increasing energy efficiency of processor caches via line usage predictors. Ph.D. thesis, Universidade Federal do Rio Grande do Sul (2014)
3. Alves, M.A.Z., Diener, M., Moreira, F.B., et al.: SiNUCA: a validated micro-architecture simulator. In: High Performance Computation Conference (2015)
4. Hybrid Memory Cube Consortium: Hybrid memory cube specification rev. 1.0 (2011). http://www.hybridmemorycube.org/
5. Hybrid Memory Cube Consortium: Hybrid memory cube specification rev. 2.0 (2013). http://www.hybridmemorycube.org/
6. Davis, B.T.: Modern DRAM architectures. Ph.D. thesis, University of Michigan (2001)
7. Henning, J.L.: SPEC CPU2006 benchmark descriptions. ACM SIGARCH Comput. Archit. News **34**(4), 1–17 (2006)
8. Intel: Intel Atom Processor E3800 Product Family. Technical report (2015)
9. Jacob, B., Ng, S., Wang, D.: Memory Systems: Cache, DRAM, Disk. Morgan Kaufmann, Burlington (2008)
10. Jeddeloh, J., Keeth, B.: Hybrid memory cube new DRAM architecture increases density and performance. In: Symposium on VLSI Technology, pp. 87–88, June 2012
11. Leidel, J., Chen, Y.: HMC-sim: a simulation framework for hybrid memory cube devices. In: International Parallel Distributed Processing Symposium Workshops, pp. 1465–1474, May 2014
12. Micron: 1Gb: x4, x8, x16 DDR3 SDRAM features, 1Gb_DDR3_SDRAM - Rev. N 11/14 EN (2006)
13. Olmen, J.V., Mercha, A., Katti, G., et al.: 3D stacked IC demonstration using a through silicon via first approach. In: International Electronic Devices Meeting (2008)
14. Patil, H., Cohn, R., Charney, M., et al.: Pinpointing representative portions of large Intel Itanium programs with dynamic instrumentation. In: International Symposium on Microarchitecture, pp. 81–92, December 2004
15. Pawlowski, J.: Hybrid memory cube (HMC). In: Hot Chips 23 (2011)
16. Rosenfeld, P.: Performance exploration of the hybrid memory cube. Ph.D. thesis, University of Maryland (2014)

17. Rosenfeld, P., Cooper-Balis, E., Farrell, T., Resnick, D., Jacob, B.: Peering over the memory wall: design space and performance analysis of the hybrid memory cube. Technical report UMD-SCA-2012-10-01, University of Maryland (2012)
18. Saito, H., Gaertner, G., Jones, W., et al.: Large system performance of SPEC OMP2001 benchmarks. In: International Symposium on High Performance Computing, pp. 370–379 (2006)
19. Thanh-Hoang, T., Shambayati, A., Deutschbein, C., Hoffmann, H., Chien, A.: Performance and energy limits of a processor-integrated FFT accelerator. In: High Performance Extreme Computing Conference, pp. 1–6, September 2014
20. Yoshida, T., Hondou, M., Tabata, T., et al.: SPARC64 XIfx: Fujitsu's next generation processor for HPC. IEEE Micro **35**(2), 6–14 (2015)
21. Zhu, Z., Zhang, Z., Zhang, X.: Fine-grain priority scheduling on multi-channel memory systems. In: International Symposium on High-Performance Computer Architecture, pp. 107–116, February 2002