Hybrid Memory Cube in Embedded Systems

ABSTRACT

This paper presents an evaluation of the Hybrid Memory Cube (HMC) use in the embedded systems context. HMC can provide better performance with less use of energy and area. Embedded systems normally have constraints in energy consumption and area usage, but yet with stringent perfomance requirements. Thus, HMC can be fit in embedded systems to provide a main memory with the needed features. In the experiments, modest configurations of L1 and L2 caches were used in conjunction with a main memory of DDR3 or HMC types. MiBench applications were executed in a 4-CPU architecture simulated at gem5. CasHMC and CACTI were used to make estimations about time, energy, and area. The HMC use allows better estimate for execution time and consumed energy, including situations where L2 cache is dispensable from the memory hierarchy. EDP and EDAP metrics show the higher HMC energy efficiency and higher density compared to DDR3. In average, EDP and EDAP values for HMC configurations were nearly 73% and 85% lower, respectively, than DDR3 configurations.

CCS Concepts

•Computer systems organization → Embedded systems; •Hardware → Memory and dense storage;

Keywords

Hybrid Memory Cube, Embedded Systems, DDR3

1. INTRODUCTION

Hybrid Memory Cube (HMC) combines high-speed logic process technology with a stack of through-silicon-via (TSV) bonded memory die. It's an innovation in DRAM memory architecture that sets a new standard for memory performance, power consumption, and cost [12]. HMC memories highlights are the improved latency, bandwidth, and density [14]. A single HMC can provide more than 15x the performance of a DDR3 module, utilizing 70% less energy per bit than DDR3 DRAM technologies, and using nearly 90% less space than today's RDIMMs [12]. *Hybrid Memory Cube Consortium* [12] embraces a number of partners dedicated to the development of HMC technology.

Embedded Systems can be viewed as information processing systems embedded in a larger product normally not visible to the users [18]. These systems are the fastest-growing portion of the computer market [5]. Embedded systems have a serie of functional requirements, but equally important are the nonfunctional ones. Typical nonfunctional requirements include performance (e.g. execution time), physical size (area), and power or energy consumption - to embedded systems energy consumption usually is more important since battery life depends on it [27]. Many factors can influence the system performance, energy consumption and area, including the system memory configuration.

This paper envisages the evaluation of HMC as a main memory and the L2 cache influence on the memory hierarchy in embedded systems domain. Embedded Systems usually have restrictions in area and energy consumption, and yet stringent constraints about execution time. Thus, we evaluate HMC as the main memory technology in embedded systems domain using the execution time, energy consumption, and area as evaluation parameters. Considering the defined evaluation parameters and derived parameters like EDP (Energy Delay Product) and EDAP (Energy Delay Area Product), the work's research questions can be defined as follows:

- RQ1) Can be advantageous to use HMC, comparing to DDR3 one, as main memory in embedded systems domain?
- RQ2) Can be L2 cache level eliminated from a memory hierarchy that uses HMC as main memory?

To the evaluation we use results from performed experiments, as follows. The execution of four aplications of MiBench [8] benchmark were simuated on *gem5* [4]. The simulated *gem5* ISA (Instruction Set Architecture) was ARM [2]. The *gem5* produce simulation stats and memory access

SAC 2018,April 09-13, 2018, Pau, France

©2016 ACM. ISBN 978-1-4503-5191-1/18/04…$15.00

DOI: DOI:http://dx.doi.org/10.1145/2851613.2851735

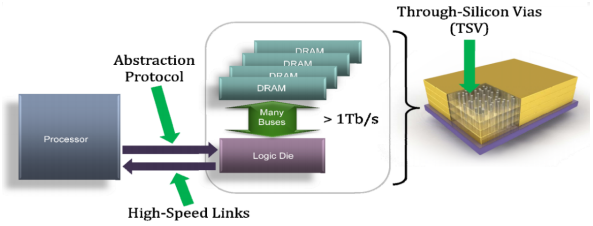


Figure 1: HMC System [14].

traces. CACTI [26] were used to produce estimates about area, access times, and energy for the caches, as well access time and area for DDR3 main memory. CasHMC [15] was used to estimate access time for the HMC main memory. For main memory energy consumption estimates, for both DDR3 and HMC, we use literature values [22, 21, 17, 14].

The rest of the paper is organized as follows. Section 2 briefly describes the HMC technology. Section 3 presents some related works. The performed experiments are presented in Section 4, including de used tool chain (4.1), the arquitecture and cache configurations (4.2) and the execution time (4.3) and energy (4.4) estimations. Section 5 presents the experiments results for execution time, energy consumption, and area, as well as the results of the derived metrics EDP and EDAP. Section 6 concludes the paper.

2. HYBRID MEMORY CUBE

A Hybrid Memory Cube (HMC) is a single package containing either four or eight DRAM die and one logic die, all stacked together using through-silicon via (TSV) technology [11]. This three-dimensional DRAM architecture effectively reduce the distance traveled by signals, increasing the density of the memory and significantly increasing the performance achieved [24]. The stacking of many dense DRAM devices produces a high-density footprint. Thus, HMC improves latency, bandwidth, power, and density when compared to DDR3 memories [12, 14]. Despite the promising advantages of 3D technology, there are significant concerns for the thermal impact. The increased power density can result from placing one power hungry block over another in the multi-layered 3D stacks [28]. Besides, the high static power of an HMC device compromises power efficiency when the device is lightly utilized [25].

Figure 1 shows the HMC system diagram. The HMC is a stack of heterogeneous die, with a standard DRAM as a building block, which can be combined with various versions of application-specific logic. The through-silicon via (TSV) technology and fine pitch copper pillar are used to interconnect the dies [14]. HMC is connected to the CPU or the GPU through high speed serial links [15]. HMC uses a simple abstracted protocol versus a traditional DRAM. The host sends read and write commands versus the traditional RAS (Row Access Strobe) and CAS (Column Access Strobe) [14].

The logic die is used to control the DRAM. Therefore, a high capacity memory can be implemented by chaining several HMC devices. 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 [15].

The HMC DRAM is a die segmented into multiple autonomous partitions. Each partition includes two independent memory banks. Within an HMC, memory is organized into vaults. Memory vaults are vertical stacks of DRAM partitions. Each partition consists of 32 data TSV connections and additional command/address/ECC connections [14]. Each vault has a memory controller (called a vault controller) in the logic base that manages all memory reference operations within that vault. Each vault controller determines its own timing requirements. Refresh operations are controlled by the vault controller, eliminating this function from the host memory controller [11].

3. RELATED WORKS

Focusing on a broader scope, especifically on 3D technology, Zou et al. [29] presents the 3D memory integration in heterogeneous architectures, allowing the integration of disparate technologies on the same chip. Beica [3] presents a review of 3D technologies with TSV integration, presenting market trends and applications. An evaluation of applying the emergent memory technologies on data-intensive applications and HPC (high performance computing) context is presented in [24], using hybrid architectures with volatile and non-volatile memories.

Santos et al. [23] explore the use of the reduced latency HMC memories to streaming aplications and point out situations where the use of L3 cache is not necessary. Other work [7] deals with performance and energy consumption issues of using a Gen2 HMC memory in the running of data-centered applications - emulation and execution are combined in a FPGA board. Alves et al. [1] proposes the HIVE architecture, a HMC memory extension to make possible processing-in-memory (PIM) of vector operations, aiming mitigate communication channel contention and cache pollution (caused by ineffective prefetches). *Active Memory Cube* (AMC) is a processing-in-memory architecture presented by Nair et al. [20] that uses a set of processing units implemented at the HMC's logic layer.

The works about HMC memories focuses, mainly, on HPC and PIM systems. However, given the HMC features, there's room for research about HMC, like the use of this type of memory in embedded systems. In this work we use HMC as main memory in the embedded systems context and compare its behaviour with DDR3 memories, evaluating the presence (or not) of L2 cache in the memory system hierarchy. The used evaluation parameters are execution time, consumed energy, area, and the derived ones (EDP and EDAP).

4. HMC EVALUATION METHODOLOGY

The work's methodology can be visualised in the Fig. 2 and consists of some steps described in this section. The methodology begins with the setup of the CACTI configuration files, the simulation environment, and the applications to be executed (Linux image file) in the simulated ISA. To the applications setup we use *MiBench* benchmark [8] cross-compiled applications to run at the ARM ISA. The

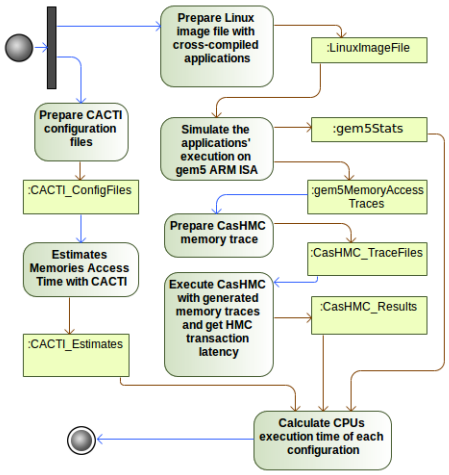


Figure 2: The performed methodology flow.

Table 1: Applications' Allocation on CPUs

CPU0	CPU1	CPU2	CPU3
basicmath	patricia	typeset	blowfish enc.

`gcc-arm-gnueabihf` (*cross-compiler*) was used to build four *MiBench* applications, according to Tab. 1. In all applications the small data set was used to save simulation time and trace file size.

After, we use *gem5* [4][6] to simulate the ARMv7 ISA [2] with a 4-core system architecture. *gem5* generates execution stats and memory access traces. We used the `Atomic-Simple` CPU model, the `Full-System` simulation mode, and the `Classic` memory model. For the memory access traces we use a command line parameter to debug memory accesses (`--debug-flags=MemoryAccess`). The CACTI [26, 16] was used to get power, area, and time (access latency) estimations of cache memories. For DDR3 we use CACTI to estimate access latency and area. For HMC we use area from a HMC memory product from Micron [19]. The CasHMC [15] was applied to get latency data (transaction latency) of HMC memory using the memory traces generated by *gem5*. The traces were adjusted to the CasHMC trace format. The transaction latency was considered as the access time of HMC memory, since the number of transactions corresponds to the sum of the traces READ and WRITE operations. Energy consumption estimations from academic literature was used for HMC [22] and DDR3 [17] main memories.

The equations 1, 2, and 3 were used to calculate the Execution Time of each CPU (and application) and the total Execution Time. And the equations 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13 were applied to obtain estimates for energy consumption, Energy Delay Product (EDP), and Energy Delay Area Product (EDAP). The equations are detailed in the Subsections 4.3 and 4.4.

4.1 Experimental Tool Chain

This subsection presents the used tools and benchmark. The *gem5* simulator supports the following ISAs: ARM, ALPHA, MIPS, Power, SPARC, and x86; including Linux boot in three of them (ARM, ALPHA, and x86) [4]. Three key dimensions provide the flexibility of *gem5* [4]: (*i*) CPU Model: with the `AtomicSimple`, `TimingSimple`, `InOrder`, and `O3` alternatives; (*ii*) System Modes: `System-call Emulation` (SE) and `Full-System` (FS) are supported; and (*iii*) Memory System: two different memory system models, `Classic` and `Ruby`, are provided.

The CACTI tool implements an analytical model for the access time and cache cycle. The main CACTI input parameters are: cache size, cache line (block) size, and associativity; and information related to the cache organization and technology parameters [13]. The CACTI was extended to deal with the off-chip characteristics of DRAM memories [16].

The *MiBench* benchmark [8] is composed of thirty-five applications, distributed in six categories, specially defined according to the embedded systems market/domain. It also defines a small and a large data set for the applications. The small data set represents a light-weight and yet useful workload, while the large data set provides a more stressful and real-world workload for embedded applications.

CasHMC [15] is a cycle-accurate simulator for hybrid memory cube (HMC). It provides a cycle-by-cycle simulation of every module in an HMC and generates analysis results including af bandwidth graph and statistical data. Memory traces can be provided as input parameter to CasHMC. The CasHMC memory traces must contain information about the execution cycle, the memory address accessed, and the respective performed operation (READ or WRITE).

Table 2: Caches Configurations

(#DDR3,#HMC)	L1i&d	L2	(#DDR3,#HMC)	L1i&d	L2
(1,9)	8KB	512KB	(5,13)	8KB	NO
(2,10)	8KB	256KB	(6,14)	16KB	NO
(3,11)	8KB	128KB	(7,15)	32KB	NO
(4,12)	8KB	64KB	(8,16)	64KB	NO

4.2 Architecture and Cache Configurations

The performed simulations have used several configurations to L1i&d caches (L1 instruction cache and L1 data cache) size and L2 cache size, according to Tab. 2. We use modest caches settings and some configurations do not use L2 cache to put more access pressure on the main memory. The main memory type was varied between DDR3 and HMC in the configurations (the first eigth configurations with DDR3 and the last ones with HMC as main memory). Some cache parameters were fixed, as follows: 64B (bytes) cache line size, 2-way L1i&d associativity, 16-way L2 associativity, and 2GB(bytes) main memory size.

The configurations aiming is to evaluate the HMC as main memory, considering the use or not of L2 cache. The base configuration use only L1i&d caches and DDR3 main memory (configuration #5 in the Tab. 2).

The used architecture on the experiments is composed of four processors. The Alg. 1 presents an excerpt from the simulation script used to put the applications in execution on each architecture CPU. The `nohup` allows to run a command ignoring hangup signals, the `taskset` is used to launch a new

command with a given CPU affinity, and the `&` instructs the command to run in background.

4.3 Execution Time Calculation

We calculate the execution time of each application on each configuration based on the *gem5* stats and memory access traces, CACTI estimates, and CasHMC results, as follows. The *Miss Penalty* (MP) represents a penalty due a cache miss an is calculated by Eq. 1. For a configuration with L2 cache, the MP for a L1 cache corresponds to the access time in L2 cache. In the case of a NO L2 configuration, the MP is the access time in the main memory (DDR3 or HMC). For convenience, the system $CycleTime$ was set to `1ns`, corresponding to a system clock frequency of `1GHz`.The MP was calculated in cycles with the ceiling operator.

Memory stall cycles (MSC) refers to the number of cycles during which the processor is locked waiting for a memory access [9]. The MSC is given by Eq. 2, its value is used to compute the CPU execution time, given by Eq. 3 [9]. The terms $\#Misses$ and $\#Cycles$ corresponds to number of the cache misses and the executed cycles (of a given CPU), respectively, both obtained from the *gem5* stats. The MSC was calculated using the misses number of each cache memory and its respective miss penalty. To determine the CPU time, the number of executed cycles in each CPU was used and the Sum of the MSC values ($SMSC$) of each cache (L1i, L1d, and, when it's the case, L2) were used, according to Eq. 3. Thus, the execution time of each CPU (and, therefore, of each application described in Tab. 1) was calculated.

```
#!/bin/sh
sleep 10          # Wait for system to calm down
cd mibench        # go to applications' folder
m5 resetstats     # Reset the gem5 stats
nohup taskset -c 0 ./basicmath_small ... &
nohup taskset -c 1 ./patricia small.udp ... &
nohup taskset -c 2 ./lout-3.24/lout ... &
nohup taskset -c 3 ./bf e input_small.asc ... &
wait              # wait applications finish its work
m5 dumpstats      # save gem5 stats
m5 exit           # exit the simulation
```

Algorithm 1: Execution Script

4.4 Energy Estimation

gem5 statistics and CACTI estimates were also used to calculate the energy consumed by the caches (L1i&d and L2). Basically, the *gem5* information is about the number of *read* and *write* operations in the cache memories. The total energy in *read* operations ($TERO$) consumed by caches is given by the number of *Read Operations* $\#RO$ multiplied by the *Read Energy*RE (Eq. 4). In similar fashion, the total energy in *write* operations ($TEWO$) is calculated by Eq. 5, where $\#WO$ is the number of *Write Operations* and WE is the *Write Energy*. Both RE and WE are provided by the CACTI tool. For the dynamic energy estimation, the total energy in misses (TEM) in each cache was calculated by Eq. 6, where $\#M$ is the total number of the cache misses and AE is the value of cache Access Energy. The used AE value is the *total dynamic read energy per access* estimate provided by CACTI. Finally, the total dynamic energy (TDE) is calculated by Eq. 7.

Although dynamic power is the primary source of power dissipation, static power is becoming an important issue because leakage current flows even when a transistor is off [9]. CACTI estimates the values of *total leakage power of a bank (mW)* and *total gate leakage power of a bank (mW)*. For our exploration, the sum of these two values is the *total leakage power* (TLP) of a bank, in mW (milliwatts). We compute the static energy for each cache memory (L1i&d and L2) using the Eq. 8. The used $CPU\,Ex\,Time$ is the maximum one between the CPUs execution times. The total static energy (TSE) is given by the sum of the $SECC$ values of each cache (L1i, L1d, and L2).

For the main memory energy estimations, for both HMC and DDR3, we used values from the literature. HMC consumes 13.7 pJ/bit and DDR3 consumes 70 pJ/bit [22]. The HMC energy consumption ($HMCec$) estimation is given by Eq. 9, and for DDR3 ($DDR3ec$) by Eq. 10. The estimations use the sum of READ and WRITE operations in the main memory and the cache line size (64 bytes), which corresponds to the size of each transaction. The total energy (TE) is calculated by Eq. 11, considering static and dynamic ones, and the energy consumed by the main memory (MME - main memory energy), conforming the memory type ($HMCec$ or $DDR3ec$).

Aiming evaluate the configurations using multiple parameters, we use the Energy Delay Product (EDP) and Energy Delay Area Product (EDAP). EDP (Energy Delay Product) for an application is defined as product of energy consumed multiplied by time taken for the application, and represents the overall gain by taking both performance and energy into account [24]. The EDP value was calculated using the total energy (TE) used by all executed applications and the maximum CPU execution time ($CPU\,Ex\,Time$) between the CPUs. The EDP value is given by Eq. 12. As a complementary measure we use Energy Delay Area Product (EDAP), given by Eq. 13. The used value for area corresponds to the sum of area estimations for the cache memories (provided by CACTI) - L2 and L1i&d - and the main memory area - for DDR3 we use CACTI estimations and for HMC we use the area ($31mm \times 31mm$) of a Micron product [19].

$$MP = \lceil {}^{Memory\,Access\,Time}/_{Cycle\,Time} \rceil \quad (1)$$

$$MSC = \#Misses \times MP \quad (2)$$

$$CPU\,Ex\,Time = (\#Cycles + SMSC) \times Cycle\,Time \quad (3)$$

$$TERO = \#RO \times RE \quad (4)$$

$$TEWO = \#WO \times WE \quad (5)$$

$$TEM = \#M \times AE \quad (6)$$

$$TDE = TERO + TEWO + TEM \quad (7)$$

$$SECC = CPU\,Ex\,Time \times TLP \times 10^{-3}(J) \quad (8)$$

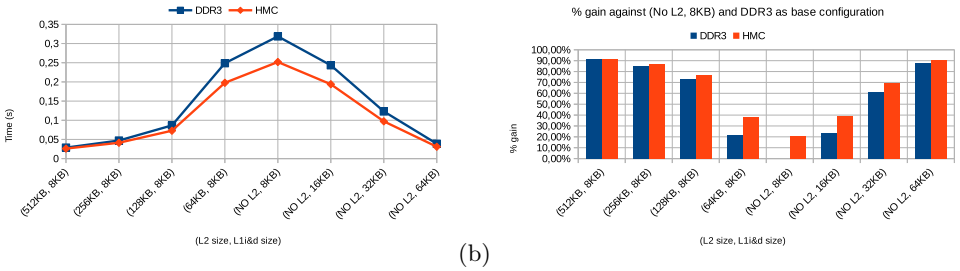
$$HMCec = (\#RO + \#WO) \times 64 \times 8 \times 13.7 \times 10^{-12}(J) \quad (9)$$

$$DDR3ec = (\#RO + \#WO) \times 64 \times 8 \times 70 \times 10^{-12}(J) \quad (10)$$

$$TE = TDE + TSE + MME \quad (11)$$

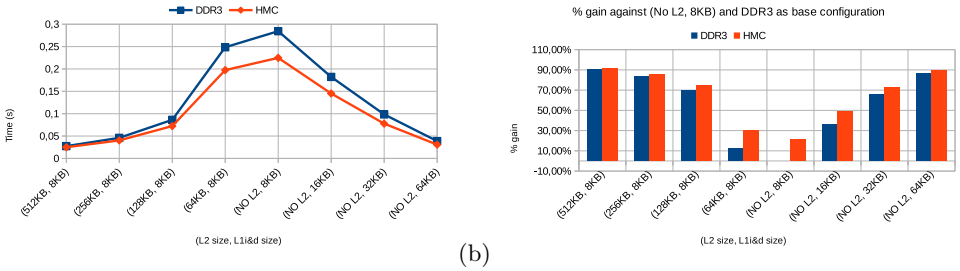
$$EDP = TE \times CPU\,Ex\,Time \quad (12)$$

$$EDAP = TE \times CPU\,Ex\,Time \times Area \quad (13)$$



(a) (b)

Figure 3: (a) Total execution times. (b) Percentage gain against (No L2, 8KB) and DDR3 configuration.



(a) (b)

Figure 4: (a) *basicmath* execution times. (b) Percentage gain against (No L2, 8KB) and DDR3 configuration.

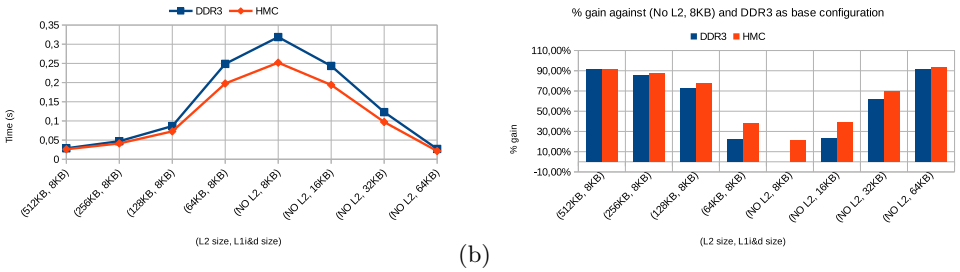
5. RESULTS AND ANALYSIS

The total execution time in the experiment is the maximum execution time between the four applications executed in the architecture CPUs, since they executed in parallel. Fig. 3 (a) shows the total execution times of each configuration and (b) shows the percentage gains of each configuration (L1i&d size, L2 size) using the `(NO L2,8KB)` and DDR3 configuration as base (configuration #5 in Tab. 2). Figures 4, 5, 6, and 7 shows the similar information (execution times and percentage gains) for each application separately. Fig. 3 (a) shows that the HMC memory provides a better execution time in all cases, but the larger L2 or L1i&d caches, lower is the diference between the DDR3 and HMC memory configurations. Fig. 3 (b) show that even without L2 cache a HMC memory can provide a comparable execution time with only L1i&d caches. Fig. 7 (b) presents negative percentage gains in the configurations with L2 cache. Since the architecture has four CPUs and four applications were executed, the CPU3 (used to run the blowfish encoding application) also executed the OS operations, since the cycles number of it corresponds to the total simulation cycles. A L2 cache level do not help in the memory loading of the programs executable files. In configurations without L2 cache the number of READ operations in main memory is significantly increased (Fig. 8(a)), even when considering the increasing in L1i&d cache size between the settings without L2 cache - the relation between READ and WRITE operations is less affected by the increasing of the L1i&d cache size (Fig. 8(b)). On the other way, the number of WRITE operations is little affected by the absence (or not) of L2 cache. The increased number in READ operations makes more direct accesses to the main memory and the HMC memory responds better to this situation (with a lower access time mean).

Fig. 9 shows the consumed energy by all applications execution. DDR3 memory causes a most intense increase in energy consumption when the L2 level cache is not present. For HMC memory, the increment in this case is less intense and comparable with configurations with the presence of L2 cache. Only with the presence of larger L2 cache or larger L1i&d caches that the use of DDR3 memory allows a closest energy consumption between the memories configurations. In this context, the cache memories can maintain almost all the applications data and instructions, so the main memory is little used. Comparing Fig. 9 and Fig. 8(a) we can see the relation between the increased number of READ operations and the increased energy consumption in the settings without L2 cache. Energy estimations for configurations with HMC memories were between 46% and 78% lower than DDR3 ones (68% in average).

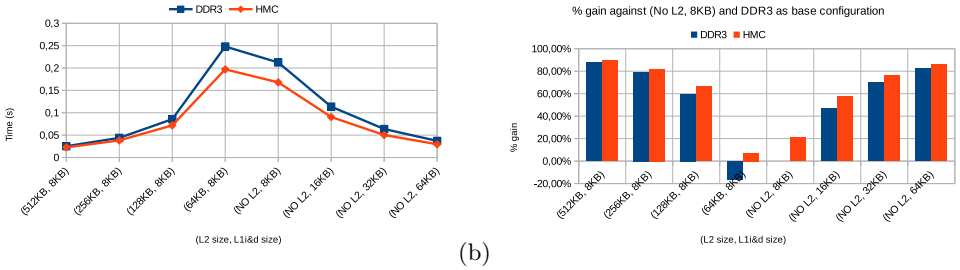
Regarding time and energy together, Fig. 10 show a energy vs. time plot detailing each configuration presented in Tab. 2. Only DDR3 settings with greater L2 or L1i&d caches can obtain similar times to the HMC configurations, but all with higher energy consumption. Fig. 11 shows the EDP values for the experimented configurations. Configurations with lower L2 or L1i&d caches (central configurations in Fig. 11) provide larger values for EDP, however the values for HMC configurations are significantly lower. EDP values for configurations with HMC memories were between 51% and 83% lower than DDR3 ones (73% in average).

Fig. 12 present the EDAP values for each configuration and the behaviour is similar to EDP graph (Fig. 11), the rigth and left side settings are comparable, however the central configurations shows the significant difference between the DDR3 and HMC memories. Fig. 13 shows a 3D plot for energy, time, and area. EDAP values for configurations with HMC memories were between 72% and 90% lower than DDR3 ones (85% in average). The 3D stacked DRAMs of HMC memories allows high density with reduced area, an



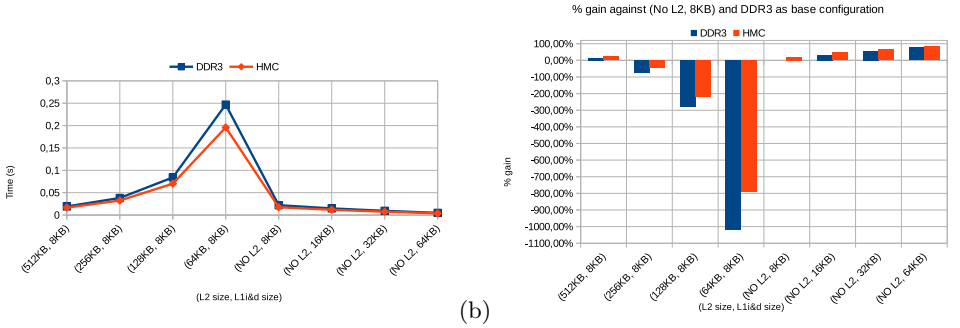
(a) (b)

Figure 5: (a) *patricia* execution times. (b) Percentage gain against (No L2, 8KB) and DDR3 configuration.



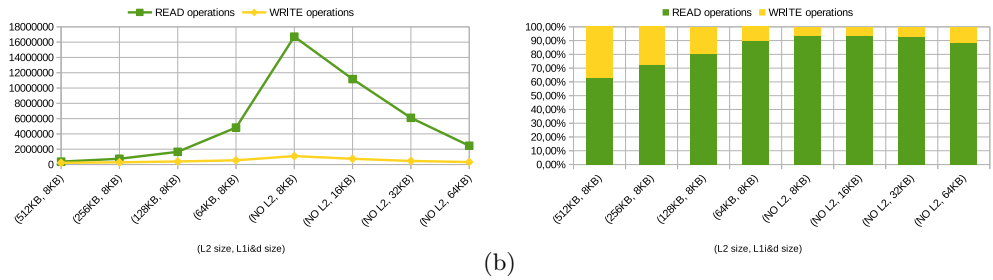
(a) (b)

Figure 6: (a) *typeset* execution times. (b) Percentage gain against (No L2, 8KB) and DDR3 configuration.



(a) (b)

Figure 7: (a) *blowfish* execution times. (b) Percentage gain against (No L2, 8KB) and DDR3 configuration.



(a) (b)

Figure 8: READ and WRITE Operations. (a) Absolute Number. (b) Percentage between Operations.

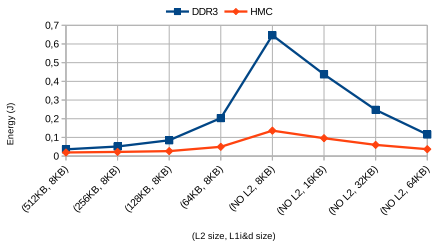


Figure 9: Energy Consumption for All Applications.

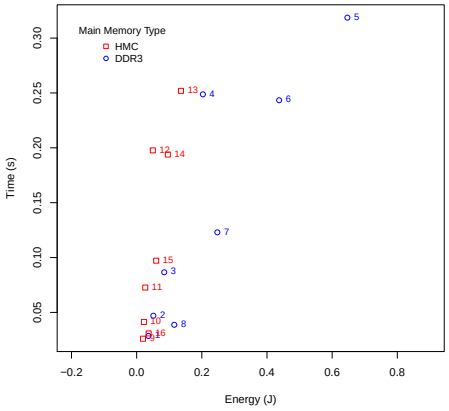


Figure 10: Energy vs Time Plot (configuration number according to Tab. 2).

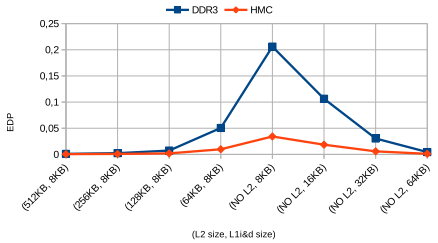


Figure 11: Energy Delay Product (EDP).

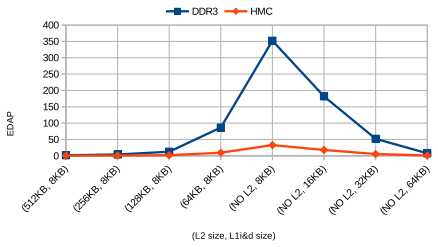


Figure 12: Energy Delay Area Product (EDAP).

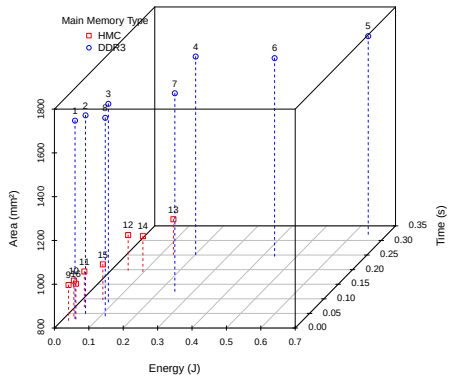


Figure 13: Energy, Time, and Area Plot (configuration number according to Tab. 2).

important aspect in a domain like Embedded Systems.

Based on the performed experiments, the two research questions presented in Section 1 can be answered:

- RQ1 answer: *Yes.* The estimates of configurations with HMC memory show better tradeoffs regarding execution time, consumed energy, and area.
- RQ2 answer: *Yes, in some situations.* The HMC configuration without L2 cache and with 8KB L1i&d cache size can obtain a similar perfomance against the DDR3 configurations with double sized (16KB) L1i&d caches or 64KB L2 cache size.

6. CONCLUSION AND FUTURE WORK

The HMC memories are presented as an innovation in DRAM memory architecture [12] and as indicated to server systems and to processing-in-memory (PIM) architectures [15]. In this work, experiments were performed to evaluate the use of HMC memories in the context of embedded systems. We had used *gem5* to simulate the ARMv7 ISA in the execution of four MiBench applications. The used configurations were varied in the size of L2 and L1i&d caches and in the memory type (DDR3 and HMC). CACTI, CasHMC and estimates from literature were used to calculate the applications' execution time and consumed energy. Also, we calculated the used area for the caches and main memory. EDP and EDAP were used to evaluate the configurations considering the execution time, consumed energy, and area in a grouped fashion.

The configurations without L2 cache force a significant increased number of, mainly, READ operations in the main memory. This operations cause higher execution times and consumed energy values, but with less impact when HMC memory is used. The configuration #13 from Tab. 2 (no L2 cache, 8KB L1i&d caches, and HMC main memory) reaches a similar performance to the configurations #4 and #6, which both uses DDR3 as main memory and a 64KB

L2 cache (#4) or a double sized (16KB) L1i&d cache (#6). This shows a situation where the L2 cache is dispensable. Also shows that HMC can be used in the Embedded System context where modest cache system configurations can be an obligation, and with gains in execution time and consumed energy. EDP and EDAP graphs (Fig. 11 and Fig. 12) show that HMC features provides configurations with a better tradeoffs regarding execution time, consumed energy, and area. The HMC energy efficiency and the high density allow these tradeoffs.

The HMC Consortium presents some boards with HMC memory inside [12]. Also, some products are available to purchase, like the HMC Module from HiTech Global [10]. As a future work, we plan to use a board to evaluate the estimated values for time and energy in the experimented configurations. Besides, the issues of thermal impact and high static power can be evaluated with a HMC equipped board.

7. REFERENCES

[1] M. A. Alves, M. Diener, P. C. Santos, and L. Carro. Large vector extensions inside the hmc. In *DATE 2016*, pages 1249–1254. IEEE, 2016.

[2] ARM Ltd. http://www.arm.com/, 2017. [Online. Accessed 10-Jul-2017].

[3] R. Beica. 3d integration: Applications and market trends. In *3DIC 2015*, pages TS5–1. IEEE, 2015.

[4] N. Binkert et al. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, Aug. 2011.

[5] T. M. Conte. *Embedded Systems*. Appendix E in [9].

[6] gem5. The gem5 Simulator - A modular platform for computer-system architecture research. http://gem5.org/Main_Page, 2017. [Online. Accessed 10-Jul-2017].

[7] M. Gokhale, S. Lloyd, and C. Macaraeg. Hybrid memory cube performance characterization on data-centric workloads. In *Irregular Applications: Architectures and Algorithms, 5th Workshop on*, page 7. ACM, 2015.

[8] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. Mibench: A free, commercially representative embedded benchmark suite. In *WWC 2001*, pages 3–14, Washington, DC, USA, 2001. IEEE Computer Society.

[9] J. L. Hennessy and D. A. Patterson. *Computer architecture: a quantitative approach*. Elsevier, 5th edition, 2012.

[10] HiTech Global, LLC. Hybrid Memory Cube (HMC) Module. http://www.hitechglobal.com/Accessories/HybridMemoryCube-HMC.htm, 2017. [Online. Accessed 09-Sept-2017].

[11] HMCC. Hybrid memory cube specification rev. 2.1, 2014. [Online. Accessed 10-Jul-2017].

[12] HMCC. Hybrid memory cube consortium. http://www.hybridmemorycube.org/, 2017. [Online. Accessed 10-Jul-2017].

[13] HP. Cacti - an integrated cache and memory access time, cycle time, area, leakage, and dynamic power model. http://www.hpl.hp.com/research/cacti/, 2008. [Online. Accessed 07-Jul-2017].

[14] J. Jeddeloh and B. Keeth. Hybrid memory cube new dram architecture increases density and performance. In *VLSIT 2012*, pages 87–88. IEEE, 2012.

[15] D.-I. Jeon and K.-S. Chung. Cashmc: A cycle-accurate simulator for hybrid memory cube. *Computer Archit. Letters*, 16(1):10–13, 2017.

[16] N. P. Jouppi, A. B. Kahng, N. Muralimanohar, and V. Srinivas. Cacti-io: Cacti with off-chip power-area-timing models. *VLSI Systems*, 23(7):1254–1267, 2015.

[17] K. T. Malladi, B. C. Lee, F. A. Nothaft, C. Kozyrakis, K. Periyathambi, and M. Horowitz. Towards energy-proportional datacenter memory with mobile dram. In *ISCA 2012*, pages 37–48, Washington, DC, USA, 2012. IEEE Computer Society.

[18] P. Marwedel. *Embedded system design*, volume 1. Springer, 2006.

[19] Micron. Hmc gen2 (hmc-15g-sr) data sheet. https://www.micron.com/products/hybrid-memory-cube/short-reach-hmc, 2010. [Online. Accessed 21-Aug-2017].

[20] R. Nair et al. Active memory cube: A processing-in-memory architecture for exascale systems. *IBM Journal of Research and Development*, 59(2/3):17–1, 2015.

[21] J. T. Pawlowski. Hybrid memory cube (hmc). In *Hot Chips 23 Symposium*, pages 1–24. IEEE, 2011.

[22] P. Rosenfeld. *Performance exploration of the hybrid memory cube*. PhD thesis, University of Maryland (College Park, Md.), 2014.

[23] P. C. Santos, M. A. Alves, M. Diener, L. Carro, and P. O. Navaux. Exploring cache size and core count tradeoffs in systems with reduced memory access latency. In *PDP 2016*, pages 388–392. IEEE, 2016.

[24] A. Suresh, P. Cicotti, and L. Carrington. Evaluation of emerging memory technologies for hpc, data intensive applications. In *CLUSTER 2014*, pages 239–247. IEEE, 2014.

[25] S. Wang, Y. Song, M. N. Bojnordi, and E. Ipek. Enabling energy efficient hybrid memory cube systems with erasure codes. In *ISLPED 2015*, pages 67–72. IEEE, 2015.

[26] S. J. E. Wilton and N. P. Jouppi. Cacti: an enhanced cache access and cycle time model. *IEEE Journal of Solid-State Circuits*, 31(5):677–688, May 1996.

[27] M. Wolf. *Computers as Components: Principles of Embedded Computing System Design*. Morgan Kaufmann, 2012.

[28] Y. Zhang, L. Li, Z. Lu, A. Jantsch, M. Gao, H. Pan, and F. Han. A survey of memory architecture for 3d chip multi-processors. *Microprocessors and Microsystems*, 38(5):415–430, 2014.

[29] Q. Zou, M. Poremba, R. He, W. Yang, J. Zhao, and Y. Xie. Heterogeneous architecture design with emerging 3d and non-volatile memory technologies. In *ASP-DAC 2015*, pages 785–790. IEEE, 2015.