HMC as Main Memory in Embedded Systems

Carlos Michel Betemps[†‡], Bruno Zatt[†], Mauricio Lima Pilla[†]
[†]Federal University of Pelotas (UFPel) - Graduate Program in Computing (PPGC) - Pelotas, RS, Brazil
[‡]Federal University of Pampa (UNIPAMPA) - Campus Bagé - Bagé, RS, Brazil
{cm.betemps, zatt, pilla}@inf.ufpel.edu.br

***Abstract*—Paper abstract [Problem. Solution. Methodology. Results.].**

I. Introduction

[Hybrid Memory Cube. Embedded Systems. Paper's Objective. Methodology. Paper's structure.]

Objetivos:

- Realizar um estudo de revisão sobre memórias HMC ([11], [19], [3], [15]).
- An evaluation of HMC memories usage in embedded systems domain, in replacement of DDR3 memories as system main memory;
- Analysis of the experiments results pointing out trends and learned lessons.

Hybrid Memory Cube Consortium [6] embraces a number of partners dedicated to the development of HMC technology. HMC memories highlights are the improved latency, bandwidth, and density [11]. Embedded Systems usually have restrictions in area and energy consumption, and yet stringent constraints about execution time and processing capacity (bandwidth). We evaluate HMC memories as the main memory technology in embedded systems domain. The evaluation uses simulated experiments.

II. Related Works

Some works had used HMC and related memories. Focusing on a broader scope, especifically on 3D technology, Zou et al. [19] 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 context is presented in [17], using hybrid architectures with volatile and non-volatile memories.

Santos et al. [16] 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 [8] 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 HMC memories extensions to make possible processing-in-memory of vector operations, aiming mitigate communication channel contention and cache polution. *Active Memory Cube* (AMC) is a processing-in-memory architecture presented by Nair et al. [14] that uses a set of processing units implemented at the HMC's logic layer. In this work we uses HMC as main memory and evaluates the L2 cache need in Embedded Systems domain.

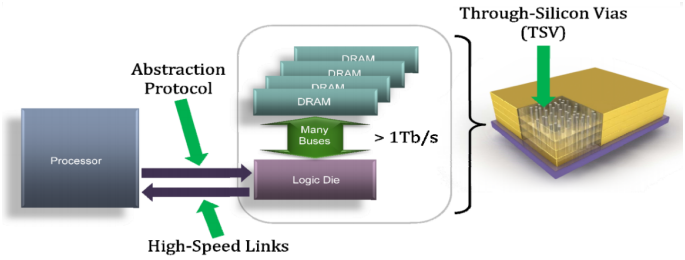

Figure 1. HMC System [11]

III. Hybrid Memory Cube Review

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 [6]. This three-dimensional DRAM architecture effectively reduce the distance traveled by signals, increasing the density of the memory and significantly increasing the performance achieved [17]. The stacking of many dense DRAM devices produces a very high-density footprint. Thus, HMC improves latency, bandwidth, power, and density [11].

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 (in the logic die). The through-silicon via (TSV) technology and fine pitch copper pillar are used to interconnect the dies [11]. HMC is connected to the CPU or the GPU through high speed serial links [12]. 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) [11].

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 [12].

The HMC DRAM is a die segmented into multiple autonomous partitions. Each partition includes two independent

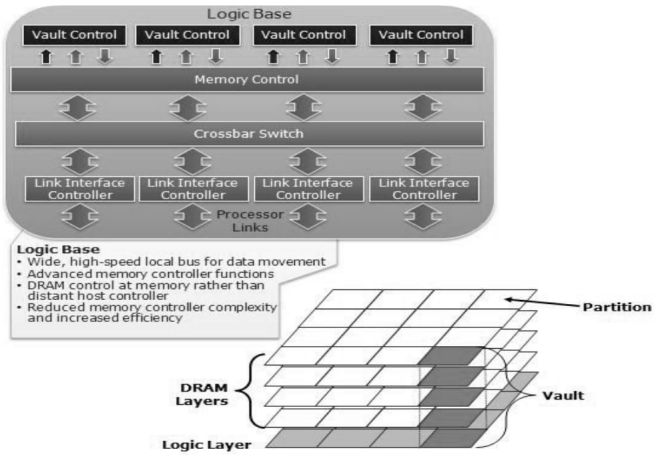

Figure 2. HMC Block Diagram

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 [11] (see Fig. 2). 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 [6].

IV. Background

[Concepts, Tools, Benchmarks, Standards, etc. used in the work.]

V. Methodology

This paper envisages evaluate the HMC memory use as a main memory of Embedded Systems, and the possibility of eliminate L2 cache from memory hierarchy. The work's research question can be defined as follows:

- As main memory, can be the HMC memories usage advantageous on embedded systems domain?
- Can be cache L2 cache level be eliminated from a memory hierarchy that uses HMC as main memory?

The work's methodology consist of the next steps:

- Use *gem5* [4][7] to simulate the ARM architecure [2], considering the needed simulation setup. The system architecture have 4 cores. *gem5* generates execution stats and memory access traces;
- Use *MiBench* benchmark [9] cross-compiled applications running at the simulated ARM architecture. We use `gcc-arm-gnueabihf` (*cross-compiler*) to build four *MiBench* applications and run each one on a respective CPU, according to Tab. II;
- Use CACTI [18], [5], [13] to get power, area, and time estimations of HMC and DDR memories;
- Use CasHMC [12] to get latency and bandwidth data. The CasHMC receives as input memory traces;

Table I
Caches and Memory Configurations

#	L1i&d Size (KB)	L2 Size (KB)	MM Type	#	L1i&d Size (KB)	L2 Size (KB)	MM Type
1	8	512	DDR	9	8	512	HMC
2	8	256	DDR	10	8	256	HMC
3	8	128	DDR	11	8	128	HMC
4	8	64	DDR	12	8	64	HMC
5	8	-	DDR	13	8	-	HMC
6	16	-	DDR	14	16	-	HMC
7	32	-	DDR	15	32	-	HMC
8	64	-	DDR	16	64	-	HMC

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. I. Some configurations do not use L2 cache. The main memory (MM) type was varied between DDR3 and HMC in the configurations. Some cache parameters were fixed, as follows:

- Cache line size of 64B (bytes);
- 2-way L1i&d associativity;
- 16-way L2 associativity;
- 2GB 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.

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

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 in this work). For convenience, the system $Cycle\,Time$ was set to `1ns`, corresponding to a system clock frequency of `1GHz`. The MP was calculated in cycles with the ceiling operator.

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

Memory stall cycles (MSC) refers to the number of cycles during which the processor is locked waiting for a memory access [10]. The MSC is given by Eq. 2, its value is used to compute the CPU execution time, given by Eq. 3 [10]. The terms $\#\,Misses$ and $\#Cycles$ corresponds to number of the cache misses and the executed cycles (of a CPU), respectively, both obtained from the *gem5* stats. The MSC was calculated using the misses number of each cache memory

Table II
APPLICATIONS' ALLOCATION ON CPUS

CPU0	CPU1	CPU2	CPU3
basicmath	patricia	typeset	blowfish enc.

Algorithm 1 Execution Script

```
#!/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
```

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 L2, when present) were used, according to Eq. 3. Thus, the execution time of each CPU (and, therefore, of each application described in Tab. II) was calculated.

$$MSC = \#Misses \times MP \tag{2}$$

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

VI. RESULTS AND ANALYSIS

[Presents the results and its analysis.]

VII. CONCLUSION AND FUTURE WORK

[Present the learned lessons, conclusions and possibilities of enhancement and future works.]

•

REFERENCES

[1] Marco AZ Alves, Matthias Diener, Paulo C Santos, and Luigi Carro. Large vector extensions inside the hmc. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*, pages 1249–1254. IEEE, 2016.

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

[3] Rozalia Beica. 3d integration: Applications and market trends. In *3D Systems Integration Conference (3DIC), 2015 International*, pages TS5–1. IEEE, 2015.

[4] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7, 2011.

[5] Ke Chen, Sheng Li, Naveen Muralimanohar, Jung Ho Ahn, Jay B Brockman, and Norman P Jouppi. Cacti-3dd: Architecture-level modeling for 3d die-stacked dram main memory. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*, pages 33–38. IEEE, 2012.

[6] Hybrid Memory Cube Consortium. Hybrid memory cube specification rev. 2.1. http://www.hybridmemorycube.org/, 2014. [Online. Accessed 10-Jul-2017].

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

[8] Maya Gokhale, Scott Lloyd, and Chris Macaraeg. Hybrid memory cube performance characterization on data-centric workloads. In *Proceedings of the 5th Workshop on Irregular Applications: Architectures and Algorithms*, page 7. ACM, 2015.

[9] 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 *IEEE International Workshop on Workload Characterization, 2001. WWC-4.*, WWC '01, pages 3–14, Washington, DC, USA, 2001. IEEE Computer Society.

[10] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. 5 edition, 2012.

[11] Joe Jeddeloh and Brent Keeth. Hybrid memory cube new dram architecture increases density and performance. In *VLSI Technology (VLSIT), 2012 Symposium on*, pages 87–88. IEEE, 2012.

[12] Dong-Ik Jeon and Ki-Seok Chung. Cashmc: A cycle-accurate simulator for hybrid memory cube. *IEEE Computer Architecture Letters*, 16(1):10–13, 2017.

[13] Norman P Jouppi, Andrew B Kahng, Naveen Muralimanohar, and Vaishnav Srinivas. Cacti-io: Cacti with off-chip power-area-timing models. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(7):1254–1267, 2015.

[14] Ravi Nair, Samuel F Antao, Carlo Bertolli, Pradip Bose, Jose R Brunheroto, Tong Chen, C-Y Cher, Carlos HA Costa, Jun Doi, Constantinos Evangelinos, 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.

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

[16] Paulo C Santos, Marco AZ Alves, Matthias Diener, Luigi Carro, and Philippe OA Navaux. Exploring cache size and core count tradeoffs in systems with reduced memory access latency. In *Parallel, Distributed, and Network-Based Processing (PDP), 2016 24th Euromicro International Conference on*, pages 388–392. IEEE, 2016.

[17] Amoghavarsha Suresh, Pietro Cicotti, and Laura Carrington. Evaluation of emerging memory technologies for hpc, data intensive applications. In *Cluster Computing (CLUSTER), 2014 IEEE International Conference on*, pages 239–247. IEEE, 2014.

[18] 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.

[19] Qiaosha Zou, Matthew Poremba, Rui He, Wei Yang, Junfeng Zhao, and Yuan Xie. Heterogeneous architecture design with emerging 3d and non-volatile memory technologies. In *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, pages 785–790. IEEE, 2015.