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 ([9], [15], [4], [12]).
- An evaluation of HMC memories usage in embedded systems domain, in replacement of DDR 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 [9]. 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

[Works that present and/or use HMC memories.]

Some works had used HMC and related memories. Focusing on a broader scope, especifically on 3D technology, Zou et al. [15] presents the 3D memory integration in heterogeneous architectures, allowing the integration of disparate technologies on the same chip. Beica [4] 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 [14], using hybrid architectures with volatile and non-volatile memories.

Santos et al. [13] 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 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. [11] that uses a set of processing units implemented at the HMC's logic layer. This work uses HMC as main memory and evaluates the L2 cache need in Embedded Systems domain.

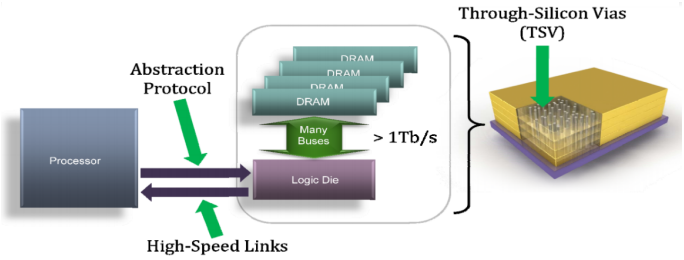

Figure 1. HMC System [9]

III. Hybrid Memory Cube Review

[A review about HMC memories.]

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 [14]. The stacking of many dense DRAM devices produces a very high-density footprint. Thus, HMC improves latency, bandwidth, power, and density [9].

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 (logic die). The through-silicon via (TSV) technology and fine pitch copper pillar are used to interconnect the dies [9]. HMC is connected to the CPU or the GPU through high speed serial links [10]. 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) [9].

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 [10].

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

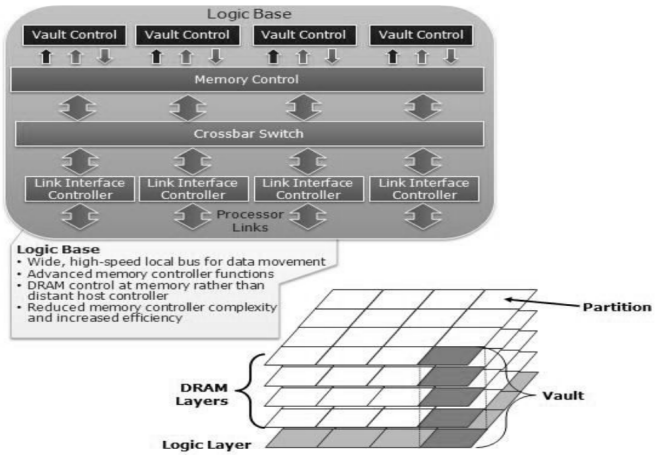

Figure 2. HMC Block Diagram

memory banks. Memory vaults are vertical stacks of DRAM partitions. Each partition consists of 32 data TSV connections and additional command/address/ECC connections [9] (see Fig. 2). Within an HMC, memory is organized into vaults. 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

[Presents the detailed steps applied in the work's development, mainly the ones related to the experiments.]

The methodology consist of the follow steps:

- Use *gem5* to simulate the ARM architecure [3], considering the needed simulation setup;
- Use *MiBench* benchmark [8] cross-compiled applications to run at the simulated ARM architecture. The used *MiBench* applications are presented on Tab. I;
- Use CACTI-3DD [5] to get power, area, and time estimations of HMC and DDR memories;
- Use CasHMC [10] to get latency and bandwidth data. The CasHMC receives as input memory traces;
- The simulation have used the following memory hierarchy settings, inspired by the characteristics of Cortex-A7 processor [2]:
  - L1i: 32kB size, 2-way associativity, 64B cache line size;
  - L1d: 32kB size, 4-way associativity, 64B cache line size;
  - L2: 512kB size, 16-way associativity, 64B cache line size (if present);
  - Main Memory: 2GB size, considering DDR and HMC technologies;

Table I
Applications' Allocation on each CPU

Core0	Core1
basicmath	bitcnts
qsort	dijkstra
patricia	stringsearch
jpeg	typeset
blowfish enc.	sha
GSM enc.	

Algorithm 1 Execution Script

```
#!/bin/sh
# Wait for system to calm down
sleep 10
cd mibench # go to applications' folder
m5 resetstats # Reset the gem5 stats
nohup taskset -c 0 ./basicmath_small ... &
nohup taskset -c 1 ./bitcnts 75000 ... &
nohup taskset -c 0 ./qsort_small qs_input_small.dat ... &
nohup taskset -c 1 ./dijkstra_small input.dat ... &
nohup taskset -c 0 ./patricia small.udp ... &
nohup taskset -c 1 ./search_small ... &
nohup taskset -c 0 ./jpeg_small.sh ... &
nohup taskset -c 1 ./lout-3.24/lout ... &
nohup taskset -c 0 ./bf e input_small.asc ... &
nohup taskset -c 1 ./sha input_small.asc ... &
nohup taskset -c 0 ./toast -fps -c data/small.au ... &
wait # wait all applications finish its work
m5 dumpstats # save gem5 stats
m5 exit # exit the simulation
```

  - Experimentation' Settings:
    - L1 + DDR (ddr) - base setting
    - L1 + HMC (hmc) - to evaluate the gain of HMC memories against DDR;
    - L1 + L2 + DDR (l2+ddr) - to evaluate the gain of L2 cache insertion in the memory hierarchy;
    - L1 + L2 + HMC (l2+hmc) - to evaluate the gain of L2 cache insertion and the use of HMC memory as main memory.
- The used architecture on the experiments is composed of two 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.

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. Cortex - A7 MPCore. Technical Reference Manual. https://static.docs.arm.com/ddi0464/f/DDI0464.pdf, 2013. [Online. Accessed 24-Jul-2017].

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

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

[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] 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.

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

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.