A Research Platform for Custom Memory Cube

Gongyu Wang, Herman Lam, Yu Zou, Riju Xavier, Shefali Gundecha, Alan George
NSF Center for High-Performance Reconfigurable Computing (CHREC),
Department of Electrical and Computer Engineering,
University of Florida, Gainesville, FL
{wangg, hlam, zou, rxavier, sgundecha, george}@chrec.org

1. INTRODUCTION

With the continuing and escalating demands from big data and extreme-scale computation as we head towards the Exascale realm, memory technology with higher bandwidth and power efficiency have become major considerations in the design of HPC architectures. The emergence of 3D stacked-memory technologies such as hybrid memory cube (HMC) [6] is driving new research on processor-in-memory (PIM), processor-near-memory (PNM), and computational RAM (C-RAM) [2][4][8] for modern data-intensive computing architectures. In HMC architecture, a base logic layer controls multiple layers of DRAM arrays, which can provide much higher memory bandwidth and power efficiency than existing memory modules. Custom memory cube (CMC) is a novel and promising extension to HMC, in which customized data-processing capabilities can be embedded in the logic layer of the HMC to take advantage of the high bandwidth and low latency in the package, enabling new forms of PIM, PNM, and C-RAM.

The concept of CMC has already been studied in several recent works [1][3][7]. Ahn et al. [1] presented a CMC-like architecture to facilitate large-scale, graph-processing apps, which has shown an order of magnitude performance improvement and 87% average energy savings over conventional systems. DRAMA [3] and AMC [7] are two forms of research on CMC, both of which reported much higher performance and power efficiency over conventional systems across multiple high-performance computing kernels and apps. Since CMC devices do not exist, the aforementioned work relied upon simulators to evaluate performance and power consumption of their notional CMC architectures. However, as new HMC parts become available, a cost-effective hardware-based platform becomes an attractive alternative to simulators for CMC researchers and designers to explore and evaluate the design space of CMC architectures and apps.

In this extended abstract, we present the design, implementation, and evaluation of a research platform for the emulation and study of CMC architectures and apps based on an FPGA-HMC board (MA-100 board developed by Micron). Details on this platform are presented in Section 2, with operations of the Data-Rearrangement Engine (DRE) [5] kernel from LLNL as an initial CMC prototype. In Section 3, we report progress and preliminary results from a page-rank app running on a DRE on the research platform and discuss future plans.

2. RESEARCH PLATFORM FOR CMC

For hardware-based, design-space exploration of CMC architectures, the research platform is required to support (1) development of custom data-processing logic, (2) design and execution of apps on CMC, (3) performance and power measurement of the execution, and (4) customization of key platform parameters (e.g., frequency, HMC packet size, etc.) for accurate emulation of CMC architectures. In this section, we describe key aspects of the CMC research platform that fulfill these requirements.

Architecture and programming. The architecture of the research platform with the DRE example is shown in Fig. 1. The host is connected to the MA-100 board through an 8-lane PCIe (gen 3) interface. On the board resides an Altera Arria-10 GX1150 FPGA connected to a 4GB HMC device through two serial 16-lane links. Infrastructure logic components (gray blocks in Fig. 1) are provided by Micron within the FPGA to enable communication among the host, FPGA, and HMC. The FPGA can be programmed by designers to house data-processing logic of the CMC, conceptually serving as *an extension to the logic layer of the HMC*. For example, we implemented DRE logic on the FPGA, with the DRE view buffer in the HMC for a CMC prototype.

The research platform supports a high-level programming language and toolset (i.e., hybrid-threading (HT) toolset) for efficient implementation of the CMC data-processing logic onto the FPGA. HT has a C/C++ form of syntax and a thread-based programming model with intrinsic support from the infrastructure components. Using HT, we were able to implement DRE as a CMC prototype on the platform more quickly than using the traditional HDL-based design flow.

Further customization of the platform parameters is possible by modifying the infrastructure components on the FPGA. Micron provided us with access to the source code (in Verilog) of these components. With their help, we are actively building new features for the platform while enhancing existing ones.

Performance and power measurement. Performance and power monitoring logic is available on the research platform for detailed inspection of the CMC architecture. As shown in Fig. 1, there are three types of performance monitors: clock-cycle counters for the CMC logic (ClkCount); performance monitors for CMC memory accesses (CMC_PERFMON); and performance monitors for the memory controllers (MC_PERFMON). The latter two types of monitors can produce a statistical summary of the total and average clock cycles associated with memory loads and stores. These monitors have a separate clock signal and control path and thus can be accessed independently from the CMC logic.

Power consumption of the FPGA and HMC devices is measured and recorded periodically by an on-board device. The measurement data can be read back through the system driver of the MA-100 board, using a pre-installed utility executable or programmatically within the app code.

Execution modes. Two execution modes of apps on the emulated CMC platform are supported: the baseline mode and the CMC mode. The former has the app running on the host, accessing the data structures allocated in the HMC, which is allowed by the unique capability of the MA-100 board for direct HMC memory accesses from the CPU (marked by blue arrows in Fig. 1). The latter has the data-intensive part of the app running on the emulated CMC platform (in the FPGA) and the other part on the host. For an app, designers can evaluate the benefits of running on the CMC architecture by comparing the measured performance and/or power

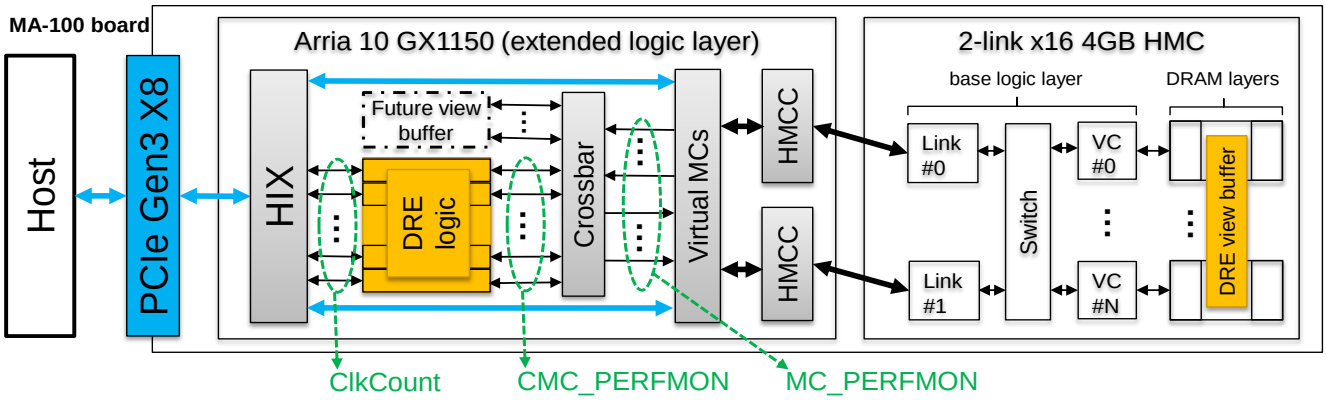


Fig. 1: Concept diagram of the CMC research platform based on MA-100 board with DRE as an example

between the CMC execution mode and the baseline mode. Moreover, various CMC designs can be evaluated by executing the app with each design in CMC mode and comparing the measured performance and/or power.

Using this research platform, we can quickly implement CMC architecture ideas, develop and execute test apps in hardware, and evaluate the ideas through in-hardware performance and power measurements. We have implemented DRE as a CMC prototype on the platform and gathered some preliminary runtime results for both execution modes.

3. PROGRESS & FUTURE WORK

This work is still in its early stages. The progress we have made includes: (1) development and tests of performance and power measurement methods for the research platform; (2) development of a CMC prototype for DRE on the platform; and (3) initial validation and tests of the DRE. Our tests with the performance monitors have shown inconsistent results, which are being inspected and validated. Initial test results of our DRE implementation using a page-rank app running on the research platform is shown in Table 1. The app is first executed in baseline mode, in which the DRE is bypassed and the host directly reads data from the HMC device. Then, the app is run in CMC mode with the DRE activated. The results show that the app runs nearly two times faster in CMC mode than baseline mode. Further breakdown of the CMC-mode runtime shows that the DRE runtime (including runtime of DRE commands: setup and fill) takes less than 1% of the total app runtime.

Currently, we are conducting experiments to validate the performance monitors using established memory benchmarks. Next, we will use DRE for further testing of the research platform: (1) applying the validated performance monitors to the DRE to gather and study the results; and (2) optimizing our DRE implementation then checking if the platform shows expected performance improvement. Based on the results and lessons learned, we will continue to enhance the platform and explore how best to use it. For example, we plan to add the following features:

Table 1: Runtime results of DRE app (page rank) running in baseline and CMC modes on the research platform

Graph scale	2^19	2^20	2^21	2^22	2^23	2^24
Baseline mode	33.08	72.62	148.60	280.20	617.04	1274.39
CMC mode	19.14	41.57	81.84	156.32	357.40	666.50
↳ DRE time	1.36	2.86	6.50	7.23	23.84	33.18
↳ non-DRE time	17.78	38.71	75.34	149.09	333.56	633.32

(1) enabling access to internal FPGA buffers using virtual addresses from the host that can reduce the latency for CMC apps to get access to the computed results of the processing logic (e.g., lower latency to access DRE view buffer as shown in Fig. 1); (2) creating a software library that allows CMC designers to use the research platform without extensive expertise in FPGA programming; and (3) calibrating the research platform for timing-accurate emulation of CMC using the statistical latency data gathered by the validated performance monitors.

4. ACKNOWLEDGMENTS

This research was supported by CHREC members and by the I/UCRC Program at NSF under Grant No. IIP-1161022. We are grateful for the feedback, collaboration, and support provided by Laboratory of Physical Sciences, Micron, and Lawrence Livermore National Laboratory.

5. REFERENCES

[1] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, “A scalable processing-in-memory accelerator for parallel graph processing,” SIGARCH Comput. Archit. News 43, 3 (June 2015), 105-117.

[2] J. Draper, J. T. Barrett, J. Sondeen, S. Mediratta, C. W. Kang, I. Kim, and G. Daglikoca, “A Prototype Processing-In-Memory (PIM) Chip for the Data-Intensive Architecture (DIVA) System,” J. VLSI Signal Process. Syst. 40, 1 (May 2005), 73-84.

[3] A. Farmahini-Farahani, J. Ho Ahn, K. Morrow and N. Sung Kim, "DRAMA: An Architecture for Accelerated Processing Near Memory," in IEEE Computer Architecture Letters, vol. 14, no. 1, pp. 26-29, Jan.-June 1 2015.

[4] M. Gokhale, B. Holmes, and K. Iobst, “Processing in Memory: The Terasys Massively Parallel PIM Array,” Computer 28, 4 (April 1995), 23-31.

[5] M. Gokhale, S. Lloyd, and C. Hajas, “Near memory data structure rearrangement,” In Proceedings of the 2015 International Symposium on Memory Systems (MEMSYS '15). ACM, New York, NY, USA, 283-290.

[6] Micron. Hybrid Memory Cube. http://www.hybridmemorycube.org/, 2011.

[7] R. Nair et al., "Active Memory Cube: A processing-in-memory architecture for exascale systems," in IBM Journal of Research and Development, vol. 59, no. 2/3, pp. 17:1-17:14, March-May 2015.

[8] H. S. Stone, “A logic-in-memory computer,” Computers, IEEE Transactions on, C-19(1):73-78, Jan 197.

Research Platform for Custom Memory Cube (CMC)

Gongyu Wang, Herman Lam, Yu Zou, Riju Xavier, Shefali Gundecha, Alan George

CHREC NSF Center for High-Performance Reconfigurable Computing

UF UNIVERSITY of FLORIDA

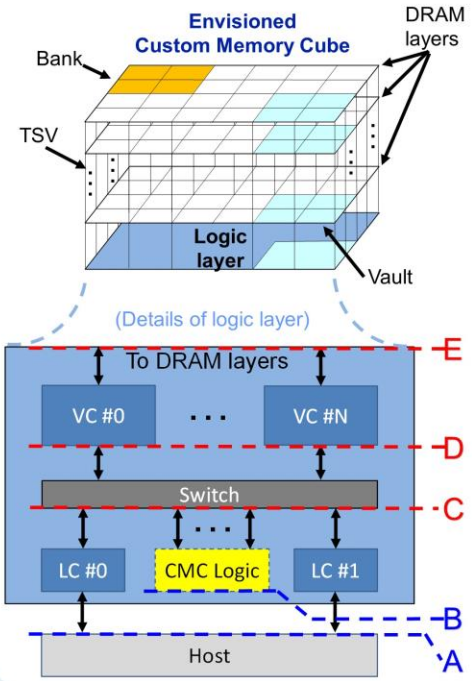
MOTIVATION

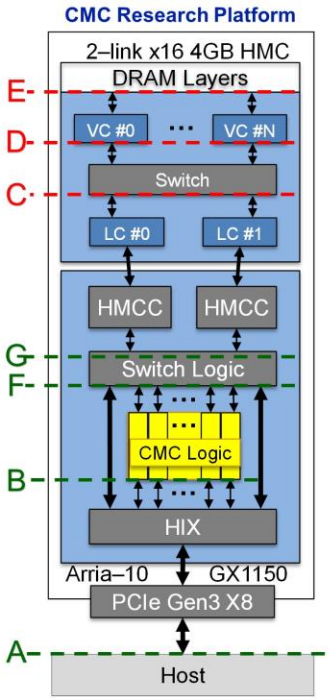
- Memory-intensive apps: memory bottleneck & high energy consumption
- Availability of 3D stacked DRAMs such as Hybrid Memory Cube (HMC)
  - Higher bandwidth and lower power than current memories
  - Substantially increased internal bandwidth and reduced latency
  - Potential for Processing Near Memory (PNM) or Processing In Memory (PIM)
- Lack of hardware-based CMC research platform
  - Benefit of such platform: hardware-in-the-loop emulation of CMC
  - Existing work on CMC arch. & app are based on simulators [1][2][3]

GOALS

- Create hardware-based research platform to perform design-space exploration (DSE) of future CMC architectures & applications
- Research platform requirements:
  - Supports DSE of alternative CMC data-processing logic
  - Able to execute CMC apps & measure performance and power of such apps
  - Platform itself is customizable (e.g., key parameters including frequency, HMC packet size, etc.)

APPROACH





1 FPGA-HMC board as basis for CMC research platform
- FPGA houses custom data-processing logic, conceptually serving as extension to logic layer
- Host conceptually connected to CMC device

2 Compute and report perf. and power of conceptual CMC
- Based on measurement of performance and power on research platform

3 Validate and show capability of platform for design-space exploration of CMC arch. & apps
- Data-Rearrangement Engine (DRE) from LLNL as initial CMC prototype

TSV: through-silicon via
Ctrl: controller
HIX: host interface crossbar
LLNL: Lawrence Livermore National Laboratory
HMCC: HMC controller
LC: link controller
VC: vault controller
FPGA: field-programmable gate array

1 Platform Architecture & Programming

- MA-100 board from Micron/Convey
  - 2-link x16 4GB HMC, Arria-10 GX1150 FPGA, and 8x PCIe Gen3 host interface
- Programming via high-level language and tools
  - Hybrid-threading (HT) toolset from Micron/Convey
  - Simple example code on right
- Customization of platform (e.g., key parameters)
  - Via modifying infrastructure/driver code of MA-100
  - Micron/Convey provided source & technical support

2 Performance & Power Measurement

- Performance measurement points: **A, B, F, G**
  - **A, B**: exec. time of app & clock count for CMC logic
  - **F, G**: perf. monitors for CMC logic & memory control
  - **C, D, E**: round-trip latencies of HMC internal parts
- Power measurement
  - Hardware part available on MA-100 to measure power of FPGA and HMC separately
- Programmable measurement in app code via infrastructure or driver of MA-100 board

3 Preliminary Results on CMC Arch. & Apps

- Host-only mode: app running on CPU directly accessing data stored in HMC
  - Supported by MA-100's direct HMC accesses
- CMC mode: data-intensive part of app running on CMC logic in FPGA and the other part on CPU
- Enable testing CMC arch. & apps in one or both modes

Graph scale	2^19	2^20	2^21	2^22	2^23	2^24
Baseline mode	33.08	72.62	148.60	280.20	617.04	1274.39
CMC mode	19.14	41.57	81.84	156.32	357.40	666.50
DRE time	1.36	2.86	6.50	7.23	23.84	33.18
non-DRE time	17.78	38.71	75.34	149.09	333.56	633.32

- Case study of DRE with page rank app
  - Running in CMC mode **2x** faster than host-only mode (i.e., baseline mode)
  - Runtime in DRE: **<1%** of total runtime

Future Work

- Perf. report of **C, D, E** based on **A, B, F, G** measurements
- Calibration of research platform for timing-accurate emulation of CMC arch. & apps
- Validate research platform by implementing known DRE designs and compare with LLNL's published results

[1] J. Ahn, et al, "A scalable processing-in-memory accelerator for parallel graph processing," SIGARCH Comput. Archit. News 43, 3 (June 2015), 105-117.
[2] A. Farmahini-Farahani, et al, "DRAMA: An Architecture for Accelerated Processing Near Memory," in IEEE Computer Architecture Letters, vol. 14, no. 1, pp. 26-29, Jan.-June 1 2015.
[3] R. Nair et al., "Active Memory Cube: A processing-in-memory architecture for exascale systems," in IBM Journal of R & D, vol. 59, no. 2/3, pp. 17:1- 17:14, March-May 2015.