A Case for Near Memory Computation Inside the Smart Memory Cube

Erfan Azarkhish,
Davide Rossi, and Igor Loi
DEI, University of Bologna, Bologna, Italy
Emails: erfan.azarkhish@unibo.it,
davide.rossi@unibo.it, and igor.loi@unibo.it

Luca Benini
ITET, Swiss Federal Institute of Technology,
Zurich, Switzerland,
DEI, University of Bologna, Bologna, Italy
Email: lbenini@iis.ee.ethz.ch

***Abstract*—3D integration of solid-state memories and logic, as demonstrated by the Hybrid Memory Cube (HMC), offers major opportunities for revisiting near-memory computation and gives new hope to mitigate the power and performance losses caused by the "memory wall". In this paper we present the first exploration steps towards design of the Smart Memory Cube (SMC), a new Processor-in-Memory (PIM) architecture that enhances the capabilities of the logic-base (LoB) in HMC. An accurate simulation environment has been developed, along with a full featured software stack. All offloading and dynamic overheads caused by the operating system, cache coherence, and memory management are considered, as well. Benchmarking results demonstrate up to 2X performance improvement in comparison with the host SoC, and around 1.5X against a similar host-side accelerator. Moreover, by scaling down the voltage and frequency of PIM's processor it is possible to reduce energy by around 70% and 55% in comparison with the host and the accelerator, respectively.**

I. Introduction and Related Works

Heterogeneous 3D integration has provided another opportunity for revisiting near memory computation to fill the gap between the processors and memories. Several research efforts in the past years demonstrate the renewed interest in moving part of the computation to where the data resides ([1][2][3]), specifically, in a 3D stacked memory context. Near memory computation can provide two main opportunities: (1) reduction in data movement by vicinity to the main storage resulting in reduced memory access latency and energy, (2) higher bandwidth provided by Through Silicon Vias (TSVs) in comparison with the interface to the host limited by the pins. Most recent works exploit the second opportunity by trying to accelerate data-intensive applications with large bandwidth demands ([4][2]). In [3] and [1] also, networks of 3D stacked memories are formed and host processors are attached to their peripheries, providing even more hospitality for processing-in-memory (PIM) due to huge bandwidth internal to the memory-centric network. These platforms, however, are highly costly and suitable for high-end products with extremely high performance goals [1]. A look at the latest specification of the Hybrid Memory Cube (HMC) [5], as the most recent technological breakthrough in memory manufacturing, reveals that its ultra-fast serial interface is able to deliver as much bandwidth as is available inside the 3D stack (Four serial links each with 32 lanes operating from 12.5GB/s to 30Gb/s). For this reason, the same bandwidth available to a PIM on the logic die is also theoretically available to the external host, and high-performance processing clusters or GPU architectures executing highly parallel and optimized applications can demand and exploit this huge bandwidth [6]. This puts PIM in a difficult but realistic position with its main obvious advantage over the external world being vicinity to the memory (lower access latency and energy) and not an increased memory bandwidth.

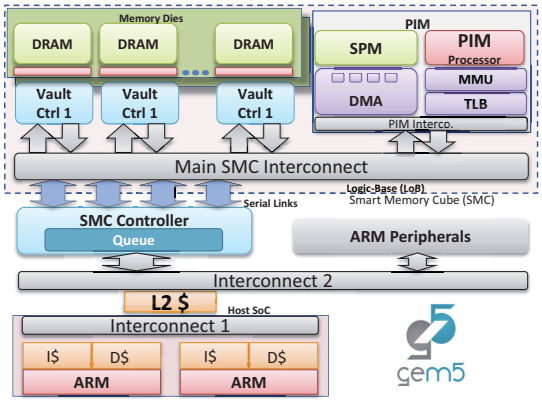

Fig. 1. An overview of the SMCSim Environment for Design Space Exploration of the Smart Memory Cube (SMC)

In this work, we focus on this dark corner of the PIM research, and try to demonstrate that even if delivered bandwidth to the host can be as high as the internal bandwidth of the memory, PIM's vicinity to memory itself can provide interesting opportunities for energy and performance optimization. We focus on a worst-case scenario where a single PIM processor is trying to compete with a single thread on host. In our experiments caches are not thrashed, the memory interface is not saturated, and the host can demand as much bandwidth as it requires. Our PIM proposal (called the Smart Memory Cube) is built on top of the existing HMC standard with full compatibility with its IO interface specification. We have developed a full-system simulation environment called SMCSim and verified its accuracy against Cycle-Accurate (CA) models [7]. SMCSim models all software and hardware layers and takes into account the offloading and dynamic overheads caused by the operating system, cache coherence, and memory management. We devised an optimized memory virtualization scheme for zero-copy data sharing between host and PIM; enhanced PIM's operations by the aid from atomic in-memory operations, as well as, a flexible Direct Memory Access (DMA) engine. Here we present our preliminary results.

II. SMCSim and PIM Design

SMCSim is a simulation environment developed based on gem5 [8], capable of modeling an SMC device attached to a complete host System on Chip (SoC) (See Fig.1). A processor-in-memory (PIM) is placed on the LoB layer of SMC with flexible and generic computational capabilities. All models have been calibrated based on the state of the art and their accuracy has been compared and verified against a previously developed

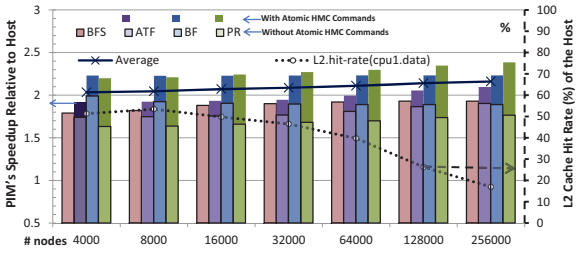

Fig. 2. PIM's speed-up with/without SMC Atomic Commands (left axis), LLC hit-rate associated with the data port of the executing CPU (right axis)

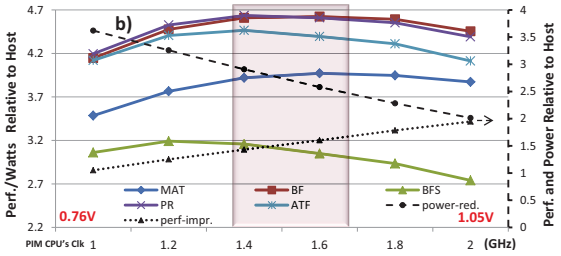

Fig. 3. PIM's energy efficiency versus its clock frequency

cycle-accurate model [7]. PIM features an ARM Cortex-A15 core without caches or prefetchers, and is augmented with a Scratchpad Memory (SPM), a Direct Memory Access (DMA) engine, a Translation Look-aside Buffer (TLB), and a Memory Management Unit (MMU). Software assisted virtual memory support has been implemented to improve its programmability and scalability. The DMA engine of PIM is capable of bulk data transfers between the DRAM vaults and its SPM, allowing for latency hiding. On the other hand, in-memory (atomic) operations can reduce data movement when computation is local to one DRAM row [2]. We have augmented our vault controllers with three types of atomic commands suitable for the benchmarks under our study. A software-stack (including a light-weight device driver and a user level API) has been developed for the user applications to view PIM as a standard accelerator. The code running on PIM is sent to it using a binary offloading mechanism and zero-copy virtual pointer sharing among host and PIM is provided, as well.

III. Preliminary Results

Our baseline host system is composed of two Cortex-A15 CPU cores @2GHz with 32KB of instruction cache, 64KB of data cache, and a shared 2MB L2 cache with associativity of 8 as the last-level cache (LLC).The memory cube model provides 512MB of memory with 16 vaults, 4 stacked memory dies, and 2 banks per partition [7]. PIM has a single core processor similar to the host processors running at the same frequency, with the possibility of voltage and frequency scaling by means of dedicated clock and voltage domains on the LoB.We have chosen four large-scale graph processing applications to accelerate on PIM: *Average Teenage Follower (ATF)*, *Breadth-First Search (BFS)*, *PageRank (PR)*, and *Bellman-Ford Shortest Path (BF)* [1][2]. We have implemented *atomic-increment*, floating-point *atomic-add-immediate*, and *atomic-min* command inside the vault controllers for the sake of these benchmarks. Randomly generated sparse graphs ranging from 4K node to 512K nodes with characteristics obtained from real world data sets [9] have been represented using List of Lists (LIL) format. Several experiments on different graph sizes demonstrated that the offloading overheads decrease with the size of the graphs and are always below 5% of the total execution time. The speed-up achieved by PIM in comparison with the host is shown in Fig.2 for different number of graph nodes. Lightly shaded columns represent PIM's speedup without aid from the atomic HMC commands, while the highly shaded ones use them. PIM's frequency is equal to the host (2GHz). An average speed-up of 2X is observable increasing with the graph size. Also, the average benefit of using *atomic-increment*, *atomic-min*, *atomic-float-add* can is obtained as 10%, 18%, and 35%, respectively.

Energy consumption of each component type has been modeled differently using logic synthesis for the interconnects, CACTI [10] for the caches, DRAMPower [11] for the DRAM devices and the available data on HMC, memory controllers, and ARM processors. We considered the energy consumed in the "used" DRAM pages, as well as, all other components shown in Fig.1. The voltage and frequency of PIM's processor (on LoB) were scaled down from 2GHz@1.05V to 1GHz@0.76V. Under these circumstances, PIM can reduce power consumption by 3X, and the optimal point in terms of energy efficiency has been plotted in Fig.3 at around 1.5GHz. Finally, comparing PIM with a similar host-side accelerator revealed that in all four graph traversal benchmarks PIM beats it by a factor of 1.4X to 1.6X. This can be explained by the latency sensitivity of the graph traversal benchmarks. Also, since the host side accelerator needs the serial links and the SMC Controller to be active, under the same conditions as the previous experiments, PIM achieves an energy reduction of 55% compared to a similar accelerator located on the host.

IV. Conclusions and Ongoing Work

We presented the first exploration steps towards design of SMC, a new PIM architecture enhancing the capabilities of the LoB die in HMC. Full-system simulation results demonstrated that even in a case where the only benefit of using PIM is latency reduction, up to 2X performance improvement in comparison with the host SoC, and around 1.5X against a similar host-side accelerator is achievable. Also, by scaling down the voltage and frequency of the proposed PIM it is possible to reduce energy by about 70% and 55% in comparison with the host and the accelerator, respectively. As an ongoing work, PIM is being extended to a cluster of processors executing parallel applications.

Acknowledgement

We would like to thank Samsung Electronics for their support and fundings.

References

[1] J. Ahn *et al.*, "A scalable processing-in-memory accelerator for parallel graph processing," in *ISCA '15*.

[2] J. Ahn, S. Yoo *et al.*, "PIM-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture," in *ISCA '15*.

[3] Z. Sura *et al.*, "Data access optimization in a processing-in-memory system," in *CF '15*.

[4] D. Zhang *et al.*, "TOP-PIM: Throughput-oriented programmable processing in memory," in *HPDC '14*.

[5] *Hybrid Memory Cube Specification 2.0*, Hybrid Memory Cube Consortium Std., 2014.

[6] J. Zhong and B. He, "Towards GPU-accelerated large-scale graph processing in the cloud," in *CLOUDCOM '13*.

[7] E. Azarkhish *et al.*, "High performance AXI-4.0 based interconnect for extensible smart memory cubes," in *DATE '15*.

[8] A. Hansson *et al.*, "Simulating DRAM controllers for future system architecture exploration," in *ISPASS '14*.

[9] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," Jun. 2014. [Online]. Available: http://snap.stanford.edu/data

[10] S. Wilton and N. Jouppi, "CACTI: an enhanced cache access and cycle time model," *JSSC*, 1996.

[11] K. Chandrasekar *et al.*, "Improved power modeling of DDR SDRAMs," in *Digital System Design (DSD), 14th Euromicro Conference on*, 2011.