PISTON Cache: Shared DRAMCache Management for Integrated Heterogeneous Systems

ABSTRACT

This document is intended to serve as a sample for submissions to ISCA 2016. We provide some guidelines that authors should follow when submitting papers to the conference. In an effort to respect the efforts of reviewers and in the interest of fairness to all prospective authors, we request that all submissions follow the formatting and submission rules detailed below.

1. INTRODUCTION

The remarkable advances in computing power of the modern microprocessor over the last few decades can predominantly be attributed to Moore's Law and advances in manufacturing technology that has allowed shrinking of transistor sizes. Miniaturization of transistors has allowed addition of specialized on-chip hardware circuitry for acceleration units. A study in 2010 by Koomey et. al [1] found that the amount of computation that could be done per unit of energy doubled about every 18 months. However, to reach exascale and beyond requires a thousand fold decrease in energy consumed per flop computed. Graphics processing units (GPUs) have evolved from being fixed-function pipelines and are being used to accelerate data parallel code for general purpose (GP) applications. Compared with multi-core CPUs, GPGPUs offer the potential for better performance at lower energy. Traditionally these discrete processors have had their own independent memory systems. To take advantage of the discrete GPUs, the CPU must copy data to the GPUs memory and back. This data movement is wasteful and expends energy while also adding latency as the transfer happens over a slower PCIe bus. The separate address spaces and complex programming models that need to manage 2 sets of data further impede expansion of the workloads that benefit from GPGPU computing. Modern processor chips have had lower capacity GPUs on-die allowing for graphics rendering only.

However in view of the widespread use of the GPU for general purpose applications, processor manufacturers including AMD[2], Intel[3], and NVIDIA[4] are beginning to allow general purpose OpenCL/CUDA programs to run on their Integrated Heterogeneous System (IHS) platform which were so far restricted to graphics rendering. To this effect the HSA Foundation [5] was setup to develop and define cross-vendor hardware specifications and software development tools needed to allow application software to better use this architecture. This architecture provides a shared virtual address space making pointer sharing semantics possible on CPU and GPU which simplifies programming. Further a shared physical address space reduces GPU initialization time and enables several high level languages to also take advantage of the parallel processing synergistically with the CPU. Programmers can now write applications that seamlessly integrate CPUs with GPUs while benefiting from best attributes of each. Finer-grain data parallel sections in applications like garbage collection [6] of virtual machines and parallel stream processing like face detection, compression, encryption-decryption etc. can now use the integrated GPGPU to deliver better performance. This integrated architecture also has the added advantage of being able to run programs whose dataset sizes are not constrained by the size of memory on the GPU. The GPU can invoke traditional operating systems paging mechanisms and hardware MMUs to fault pages.

Parallely, DRAM memory speeds have not kept pace commensurate to CPU speeds and this coupled with a limited growth in pin counts has led to memory [7] and bandwidth wall [8] for off-chip DRAM systems. The advent of die-stacking technology [9] provides a way to integrate disparate silicon die of NMOS DRAM chips and CMOS logic chips with better interconnects. The implementation is accomplished either by 3D vertical stacking of DRAM chips using through-silicon vias (TSV) interconnects or horizontally/2.5D stacking on a interposer chip as depicted in Figure 1. This allows the addition of a sizable DRAM chip close to processing cores. The onchip DRAM memory can provide anywhere from a couple of hundreds of megabytes to a few gigabytes of storage at high bandwidths of 400GB/s compared to the 90GB/s of DDR4 bandwidth. The better interconnect also lowers the latency of access by around 20-25% compared to off-chip memory. This on-chip DRAM capacity has been advocated to be used as a large last level cache which is transparent to software in several works in literature. In this context, throughput oriented GPUs with high MLP and large bandwidth requirement can benefit from the bandwidth capabilities while latency oriented CPU applications can benefit from reduced latency of data access from the DRAM Cache thus improving the overall system performance. The stacked DRAM Cache also reduces energy consumed per access for the overall system in line with the goals of IHS.

However, this introduces novel challenges in managing contention for shared memory resources where the DRAMCache is the first level of shared capacity in the memory

hierarchy of the two heterogeneous processor architectures. When a GPU kernel is launched it creates large number of concurrently running threads in lock-step in a SIMD execution model and injects large number of requests to DRAMCache. The GPU can also switch execution between groups of threads to hide this memory access latency and exploit available parallelism which further exacerbates the problem. This causes bottlenecks in request queues and contention for sets in the DRAMCache thus severally impacting CPU performance. Although the GPU can sustain longer memory latencies, as capabilities of execution units in GPUs of IHS chips increase, threads will be able issue larger number of memory requests more rapidly and available parallelism to hide memory latency decreases, thus reducing throughput of the entire system. Since the GPU can execute in a burst mode reserving shared resources will lead to idling and under-utilization of resources.

The primary contributions of this work are summarized as follows:

- We propose the addition of a large capacity stacked DRAM for IHS processors. This first level shared memory capacity between CPU and GPU would be used a last-level cache to contain the majority of the large working set of GPU which otherwise will not fit in a on-chip SRAM cache. The considerable bandwidth provided by the DRAMCache should benefit GPU programs and the lower latency of access will assist latency sensitive CPU applications.
- We study the effects of sharing this resource between the processors and show that conventional DRAMCache architectures do not perform well when applied naively for IHS processors. This is the first of the kind evaluation for a IHS setup.
- We propose an intelligent IHS aware scheme for managing occupancy levels for a direct mapped DRAMCache called *PISTON*. *PISTON* is lightweight and at the same time avoids explicit cache partitioning which would result in unused capacity when either cores are idle. *PISTON* does not compromise on .
- The predictor is central to extracting performance from a DRAMCache by avoiding high latency tags-in-DRAM lookups by predicting a miss ahead of time and starting an early fill request to memory in parallel. We adapt the DRAMCache predictor to be IHS aware in concordance with our occupancy control mechanism. This allows each type of processor to maximize the use of bandwidth and latency benefits provided by the DRAMCache.
- Finally we enable *PISTON* to dynamically adjust occupancy at runtime for varying cache requirements of heterogeneous cores to maximize the overall system throughput that can be got by using this DRAMCache.

To the best of our knowledge, this is the first study on the interactions of IHS with a shared stacked DRAMCache. The rest of the paper is organized as follows. Section 2 demonstrates the performance that can be gained by adding a DRAMCache to a IHS chip and motivates the need for architecting an IHS aware approach to DRAMCache organization. We present the principles of PISTON Cache and its descibe its working in Section 3 . Next, in Section 4 we show the experimental setup and methodology followed by evaluation results in Section 5. Section 6 presents related work in this area and Section 7 concludes the paper.

2. MOTIVATION

As seen in section 1 Incorporate GPU into the same die as CMPs has become widespread and their use as GPGPU is gaining traction as well.

3. CHAINING MECHANISM

4. EXPERIMENTAL SETUP & METHODOLOGY

We evaluate the performance of chaining using multi-programmed SPEC 2006 applications coupled with a Rodinia application that contains the GPU phase of execution. We use Rodinia [10] applications that are modified to elide the memcpy api calls to run with unified virtual and physical address spaces. These workloads are run on a cycle accurate simulator gem5-gpu [11] which is configured to simulate cache coherent unified CPUs and GPUs using the VI_Hammer protocol. The cache hierarchy has per SM private GPU L1 that are non-inclusive of the shared GPU L2 cache and can hold stale data. However, GPU L2 cache is coherent with all levels of the CPU hierarchy. The DRAM Cache we evaluate here is the first level shared cache between the 2 split cache hierarchies of CPUs and GPUs while they have a shared level of cache within themselves. We fast-forward the initialization phase of the workloads up until just before the launch of the first kernel of the GPU program. We ensure that each core executes atleast 2 Billion instructions and each 4 core workload executes 20 billion instructions and each 16 core workload executes x Billion on average in the fast-forward phase. We do this by letting adding no-ops to the Rodinia benchmarks for the duration until the initialization of the SPEC programs is complete.

Multiple memory controllers accesses are almost equally divided by clever XOR mechanism. Each mem ctrl is responsible for a 4GB memory DDR3 1600MHz DRAM and a 128MB HMC 2500MHz DRAM Cache device. The 128MB DRAM Cache caches only data from the 4GB memory chunk memory controller is responsible for. So there are no cross bus requests between controllers.

We faithfully model Fill Queue for fill requests for insertion into cache on the return path from main memory. [12]

5. RESULTS

6. RELATED WORK

7. CONCLUSION

8. REFERENCES

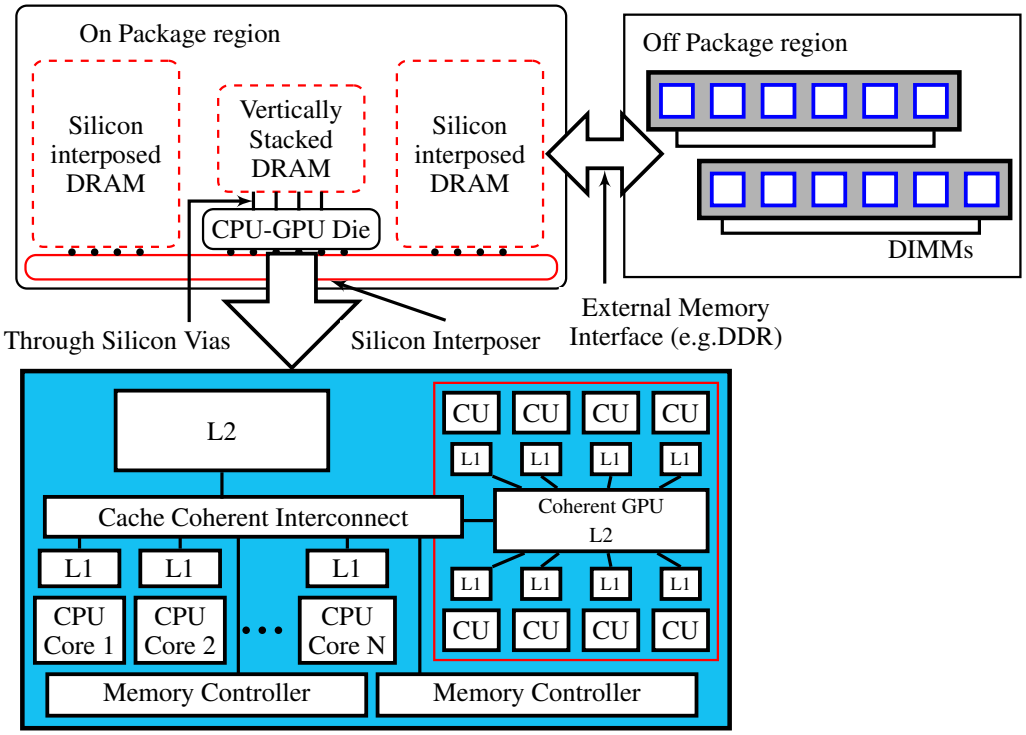

Figure 1: Architecture of a Integrated Heterogeneous System

[1] J. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of historical trends in the electrical efficiency of computing," *IEEE Annals of the History of Computing*, vol. 33, pp. 46–54, March 2011.

[2] "The future of the apu - braided parallelism." `http://developer.amd.com/wordpress/media/2013/06/2901_final.pdf`.

[3] "Intel graphics opencl." `https://software.intel.com/en-us/node/540387`.

[4] "Fastest processors, smartphones, and tablets - nvidia tegra." `http://www.nvidia.com/object/tegra.html`.

[5] "Hsa foundation standards." `http://www.hsafoundation.com/standards/`.

[6] "Java sumatra." `http://openjdk.java.net/projects/sumatra/`.

[7] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *SIGARCH Comput. Archit. News*, vol. 23, pp. 20–24, Mar. 1995.

[8] B. M. Rogers, A. Krishna, G. B. Bell, K. Vu, X. Jiang, and Y. Solihin, "Scaling the bandwidth wall: Challenges in and avenues for cmp scaling," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, (New York, NY, USA), pp. 371–382, ACM, 2009.

[9] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCaule, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb, "Die stacking (3d) microarchitecture," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, (Washington, DC, USA), pp. 469–479, IEEE Computer Society, 2006.

[10] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*, IISWC '09, (Washington, DC, USA), pp. 44–54, IEEE Computer Society, 2009.

[11] J. Power, J. Hestness, M. Orr, M. Hill, and D. Wood, "gem5-gpu: A heterogeneous cpu-gpu simulator," *Computer Architecture Letters*, vol. 13, Jan 2014.

[12] A. J. Cheng-Chieh Huang, Vijay Nagarajan, "Dca: a dram-cache-aware dram controller," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, November. 2016.