Optimizing GPU Energy Efficiency with 3D Die-Stacking Graphics Memory and Reconfigurable Memory Interface

JISHEN ZHAO, Pennsylvania State University

GUANGYU SUN, Peking University

GABRIEL H. LOH, Advanced Micro Devices, Inc.

YUAN XIE, Pennsylvania State University

The performance of graphics processing unit (GPU) systems is improving rapidly to accommodate the increasing demands of graphics and high-performance computing applications. With such a performance improvement, however, power consumption of GPU systems is dramatically increased. Up to 30% of the total power of a GPU system is consumed by the graphic memory itself. Therefore, reducing graphics memory power consumption is critical to mitigate the power challenge.

In this article, we propose an energy-efficient reconfigurable 3D die-stacking graphics memory design that integrates wide-interface graphics DRAMs side-by-side with a GPU processor on a silicon interposer. The proposed architecture is a “3D+2.5D” system, where the DRAM memory itself is 3D stacked memory with through-silicon via (TSV), whereas the integration of DRAM and the GPU processor is through the interposer solution (2.5D). Since GPU computing units, memory controllers, and memory are all integrated in the same package, the number of memory I/Os is no longer constrained by the package’s pin count. We can reduce the memory power consumption by scaling down the supply voltage and frequency of memory interface while maintaining the same or even higher peak memory bandwidth. In addition, we design a reconfigurable memory interface that can dynamically adapt to the requirements of various applications. We propose two reconfiguration mechanisms to optimize the GPU system energy efficiency and throughput, respectively, and thus benefit both memory-intensive and compute-intensive applications. The experimental results show that the proposed GPU memory architecture can effectively improve GPU system energy efficiency by 21%, without reconfiguration. The reconfigurable memory interface can further improve the system energy efficiency by 26%, and system throughput by 31% under a capped system power budget of 240W.

Categories and Subject Descriptors: B.3.1 [Memory Structures]: Semiconductor Memories

General Terms: Design, Performance

Additional Key Words and Phrases: Graphics memory, 3D packaging, reconfigurable interface, 3D ICs

ACM Reference Format:

Zhao, Z., Sun, G., Loh, G. H., and Xie, Y. 2013. Optimizing GPU energy efficiency with 3D die-stacking graphics memory and reconfigurable memory interface. *ACM Trans. Architec. Code Optim.* 10, 4, Article 24 (December 2013), 25 pages.

DOI: <http://dx.doi.org/10.1145/2541228.2541231>

This paper is an extension of the conference paper entitled “Energy-efficient GPU Design with Reconfigurable In-package Graphics Memory” [Zhao et al. 2012] in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, 2012.

Authors’ addresses: J. Zhao, CSE Department, Pennsylvania State University; email: juz138@cse.psu.edu; G. Sun, Center for Energy-efficient Computing and Applications, Peking University; email: gsun@pku.edu.cn; G. H. Loh, AMD Research, Advanced Micro Devices, Inc.; email: gabe.loh@amd.com; Y. Xie, CSE Department, Pennsylvania State University; email: yuanxie@cse.psu.edu.

This work is supported in part by SRC grants, and NSF 0903432, 1017277, and 1213052.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701, USA, fax +1 (212) 869-0481 or permission@acm.org.

© 2013 ACM 1544-3566/2013/12-ART24 \$15.00

DOI: <http://dx.doi.org/10.1145/2541228.2541231>

1. INTRODUCTION

Modern Graphics Processing Unit (GPU) systems have become an attractive solution for both graphics and general purpose workloads that demand high computational performance. The GPU exploits extreme multithreading to target high-throughput [AMD 2012; NVIDIA 2010]. For example, AMD Radeon™ HD 7970 employs 20,480 threads interleaved across 32 compute units [AMD 2012]. To accommodate such high-throughput demands, the power consumption of GPU systems continues to increase. As existing and future integrated systems become power limited, reducing system power consumption while maintaining high energy efficiency is a critical challenge for GPU system design.

To satisfy the demands of high-throughput computing, the GPUs require substantial amounts of memory (from hundreds of megabytes to gigabytes, which are usually off-chip memory) that can support a very large number of read and write accesses. Consequently, the off-chip memory consumes a significant portion of power in a GPU system. We evaluated the maximum power consumption of two GPU systems, AMD Radeon™ HD 7970 [AMD 2012], and NVIDIA Quadro® 6000 [NVIDIA 2010], with the memory power model described in Section 5. Figure 1 presents the evaluated power distributions of GPU cores and caches, memory controllers, and off-chip memory. We considered the memory bandwidth utilization (the fraction of all cycles when the data-bus is busy transferring data for reads or writes) from 10% to 50%. For both studied GPU systems, the off-chip memory consumes from 20% to more than 30% of the total GPU system power. Note that for the workloads evaluated in this article, the highest average bandwidth utilization observed was 35%. At this bandwidth level, the memory power consumption is 30.1% and 27.7% for the two GPU systems, respectively. If we can reduce the memory power by half, 12.5% of system power can be saved; this may seem like a relatively small amount, but it is in fact quite significant. For example, the maximum power consumption of AMD Radeon™ HD 7970 [AMD 2012] is 230W; therefore, a 12.5% power reduction saves 29W. Therefore, techniques that reduce the graphics memory power requirements can be very effective at reducing the total system power.

Conventional Graphics DRAMs (GDDR) have employed several techniques to reduce memory power consumption. Using the Pseudo-Open Drain (POD) signaling scheme [Elpida 2010] with On-Die Termination (ODT), static power is only consumed when driving a “low” signal and thus reduces the power of the memory interface. GDDR5, the latest generation of commercial graphics memory, provides further power savings compared to its predecessors with lower supply voltages; DVFS; and independent ODT strength control of address, command, and data [Elpida 2010]. Voltage/Frequency (VF) scaling techniques employed by conventional GDDRs reduce power by adapting the memory interface to the memory bandwidth requirements of an application. However, the power reduction comes at the expense of memory bandwidth degradation. For example, Elpida’s GDDR5 memories are specified to operate over a large contiguous VF range to support data rates starting from as low as 800MB/s per channel to the maximum rate of 20GB/s. Although 1.6GB/s per channel may be sufficient for displaying static images, a data rate of 6GB/s is required for playing high-definition (HD) video, and the maximum data rate of 20GB/s may be fully utilized by high-end gaming applications. For future high-performance GPGPU and advanced video processing (e.g., 3D HD multiscreen video games), existing power-saving techniques may not suffice.

In this article, we propose to integrate GDDR-like graphics memory with the GPU processor and memory controllers in a single package with silicon interposer system-in-package (SiP) technology so that the number of memory I/Os are not constrained by the package pins. By using the significantly larger number of data I/Os, we can provide a given amount of bandwidth while reducing the power consumption of memory

Fig. 1. Power breakdown of both NVIDIA and AMD GPUs. Power consumptions of GPU cores and caches, memory controllers, and DRAMs are examined to show that off-chip DRAM accesses consume a significant portion of the total system power.

by scaling down its supply voltage and frequency. Furthermore, we design a reconfigurable memory interface and propose two reconfiguration mechanisms (EOpt and PerfOpt) to optimize (1) GPU system energy efficiency and (2) system throughput under a given power budget. With minor changes to the memory interface, our wide-bus 3D die-stacking memory design is practical and easy to implement. With the flexibility of optimizing for either power or performance, our design can adapt to both high-performance computing, which prefers high throughput, and power-constrained applications, which prefer better energy efficiency. The contributions we present in this article include:

- A 3D die-stacking wide-I/O interface graphics memory design with lower power consumption than conventional graphics memories while providing better peak memory bandwidth.
- A reconfigurable memory interface, which can adapt the bus width, frequency, and supply voltage to the application's runtime performance and memory access behavior.
- Two reconfiguration mechanisms with different optimization targets. One mechanism targets at optimizing GPU system energy efficiency by reconfiguring the memory interface alone. The other simultaneously co-optimizes both the reconfigurable memory interface and the GPU clock frequency to increase system throughput under a given power budget.

2. BACKGROUND AND RELATED WORK

2.1. Energy-Efficient GPU Architectures

Various existing work explores how to address the GPU power challenge [Gebhart et al. 2011; Yu et al. 2011; Ren and Suda 2010; Al Maashri et al. 2009; Galal and Horowitz 2011; Wang et al. 2009]. Gebhart et al. [2011] investigated register file caching and multilevel thread scheduling to reduce the number of accesses to large register files to reduce power [Gebhart et al. 2011]. SRAM-DRAM hybrid memory technology was exploited by Yu et al. [2011] to reduce the area and power consumption of GPU register files [Yu et al. 2011]. In their work, embedded DRAM (eDRAM) with a higher density than SRAM is used to store multiple copies of register file data. Wang et al. [2009] proposed the predictive shader shutdown technique to exploit workload variations across

frames for leakage reduction of GPU shader processors [Wang et al. 2009]. Software optimization was studied by Ren et al. [2010] to improve the GPU power efficiency by modifying matrix multiplication algorithms [Ren and Suda 2010]. Most of the existing studies explore either GPU shader cores and caches architecture, or software optimization, and require both hardware and software modifications to current GPU processor design. In our work, we explore power reduction techniques by limiting the architectural modifications to the graphics memory interface with only minor changes to GPU compute-unit architecture.

2.2. Graphics Memory

Graphics Double Data Rate (GDDR) memories are specifically designed for graphics cards, game consoles, and high-performance computing systems. The operation and interface of GDDR memories are similar to DDR memories. Similar to other DRAM technologies, GDDR memories are composed of a number of memory banks. Each bank consists of a 2D array that is addressed with a row address and a column address, both of which share the same address pins. In a typical memory access, a row address is first provided to a bank using a command that activates all of the memory cells in the row. Memory cells are connected to the sense amplifiers using long wires and are subsequently connected to the data pins. Once the sense amplifiers detect the values of the memory cells in a row, they latch the values in a row buffer so that subsequent accesses can be serviced directly from the row buffer and also for the eventual write-back to the DRAM cells. Graphics and multimedia applications require high memory bandwidth to render 3D images and buffer the large amount of frame data for image and video processing. To satisfy these needs, GDDR memories usually employ high frequencies in order to achieve very high bandwidths. Unfortunately, power consumption dramatically increases as well. A large power supply is required and usually comes with high cost. It has been reported that the GDDR power consumption is growing linearly with bandwidth [Samsung 2010]. The peak power consumption of GDDR3 consumes more than 100W with less than 128GB/s bandwidth [Samsung 2010]. GDDR5 is the latest generation of graphics DRAM architecture that improves upon conventional DRAM technologies, such as DDR3 SDRAM [Samsung 2010], by increasing clock speeds and decreasing power requirements. Although GDDR5 employs VF scaling to reduce power consumption, the provided memory bandwidth is degraded at the same time. In our work, we exploit the interconnect bandwidth of die-stacking technologies (silicon interposer specifically) to reduce the graphics memory power consumption while maintaining the required bandwidth.

2.3. 3D Package Using Silicon Interposer Technology

3D packaging approaches that employ through-silicon vias (TSVs) and silicon interposer have been widely adopted in high-performance multichip module (MCM) and SiP designs [Dorsey 2010; Dong et al. 2010; Khan et al. 2006; Kim et al. 2011b]. Various TSV techniques have been developed for next-generation packaging technologies [Akazawa et al. 2003; Andry et al. 2006]. A silicon interposer provides high-density interconnections and is capable of providing a line width of less than $10\mu\text{m}$ and hence hundreds of thousands of I/O pads per square centimeter. It has thousands of TSVs per square centimeter for high-density routing. The coarse-pitch TSVs provide the external connections between the package and individual dies/chips for the parallel and serial I/O, power/ground, clocking, configuration signals, and so forth. The length of the interconnection from the chip to the substrate is reduced. A variety of work from academia and industry explores the development of MCM and SiP with silicon interposers. Sunohara et al. [2008] proposed a design with a microchannel substrateless package that can be used to fabricate highly integrated silicon modules

such as systems on chip (SoC). Xilinx uses silicon interposers with microbumps and TSVs to combine multiple FPGA die slices in a single package [Dorsey 2010]. Their stacked-silicon interconnect design provides tens of thousands of I/Os that interconnect multiple FPGA chips. The latency of signals passing between FPGAs is also reduced to $\frac{1}{5}$ of that of standard I/Os. A heterogeneous main memory was proposed by Dong et al. [2010] to implement in-package memories providing fast and high-bandwidth data access. Their design employs high-speed in-package interconnects and implements a small fast memory space to work with a conventional off-package main memory. In this article, we leverage silicon interposer-based 3D packaging technology to explore the energy efficiency benefits for GPU systems.

2.4. Integrated/In-Package DRAMs

Most previous studies on integrated/in-package memory systems employ vertical (3D) memory stacking and place the DRAMs directly on top of the processor cores [Kgil et al. 2006; Liu et al. 2005; Loi et al. 2006; Loh 2008; Gu et al. 2008; Woo et al. 2010; Kim et al. 2011a; Micron 2013; Tezzaron 2010]. Samsung recently announced a 3D-stacked wide-I/O DRAM targeting mobile system [Kim et al. 2011a]. The presented two-layer DRAM with four 128-bit wide buses has 12.8GB/s peak bandwidth, 2Gb of capacity, and only 330.6mW of power consumption. Unfortunately, graphics and high-performance computing systems require much higher memory bandwidth of several hundred GB/s; therefore, this mobile-optimized part would be difficult to directly apply to GPUs. Woo et al. [2010] rearchitected the memory hierarchy, including the L2 cache and DRAM interface, and take full advantage of the massive bandwidth provided by stacking the DRAMs on top of processor cores. Tezzaron Corporation [2010] has implemented true 3D DRAMs, where the individual bitcell arrays are stacked in a 3D fashion. The peripheral control logic and circuitry are placed on a separate, dedicated layer, incorporated with different process technologies. Recently, developed the Hybrid Memory Cube (HMC) [Micron 2013], which combines high-speed logic process technology with a stack of TSV bonded memory die. The HMC increases density per bit and reduces the overall package form factor. Different from prior work, our memory interface and reconfiguration mechanisms are specifically designed for GPU systems with massive multithreading. Although the integrated graphics memory can potentially provide very high bandwidths by exploiting the huge number of connections afforded by interposer-based integration, the overall memory interface still needs to be carefully designed and balanced because the power consumption also increases with the bus widths.

2.5. Dynamic Power Management Techniques in GPU and CPU Systems

Our work explores power management mechanisms with dynamic voltage/frequency scaling (DVFS) technique applied to (1) graphics memory only (EOpt in Section 4.3) and (2) the entire GPU system consisting of the GPU processor and graphics memory (PerfOpt in Section 4.4). Most existing mechanisms seeking to save power consumption of GPU systems focused on leveraging idle states of GPU cores [NVIDIA 2008; Jiao et al. 2010] and graphics memory [Samsung 2010]. How to actively tune the VF states of GPU systems remains an open question.

A large body of previous work has studied CPU system power management with DVFS techniques. Most of these studies focused on DVFS of only CPU cores [Herbert and Marculescu 2007; Kaxiras and Martonosi 2009; Isci et al. 2006; Wu et al. 2005]. Recent work [David et al. 2011; Deng et al. 2011] showed that DVFS on memory provides substantial energy savings. One of our reconfiguration mechanisms, EOpt, also employs DVFS of graphics memories. Compared to prior work [David et al. 2011; Deng et al. 2011], EOpt can sustain peak memory bandwidth during reconfiguration with

Fig. 2. Conventional GPU system with off-chip GDDR5s. The number of data I/Os is limited by package pin count.

modifications restrained to memory controller and memory interface. Very few works studied coordinated DVFS for power management of the entire system including CPU processor and memory subsystem. CoScale [Deng et al. 2012] explored power management mechanisms by coordinating DVFS of CPU and memory subsystem under performance constraints. However, CoScale [Deng et al. 2012] implemented its DVFS algorithm in the operating system (OS) with a typical reconfiguration interval corresponding to an OS time quantum (5ms). Directly adopting the CoScale [Deng et al. 2012] method in GPU systems will require even longer reconfiguration intervals, because the DVFS algorithm that executes on the host CPU will need to communicate with the GPU through the PCIe interface with a very long turnaround latency. Profiling results shows that our benchmarks (described in Section 5) require reconfiguration interval ranges from $3\mu s$ to $60\mu s$, much smaller than OS time quanta. Consequently, we provide a hardware-based mechanism to perform DVFS to GPU processor and graphics memory with smaller reconfiguration intervals than OS-based mechanisms.

3. 3D DIE-STACKING GRAPHICS MEMORY ARCHITECTURE

Many power-saving techniques for graphics memory come with the undesirable side effect of a degradation of the provided memory bandwidth. In this section, we present the feasibility results and energy efficiency benefits of 3D die-stacking wide-interface graphics memory, integrated side-by-side with a GPU processor. We show that VF scaling can be employed to reduce power consumption without affecting memory bandwidth by leveraging silicon interposer-based 3D packaging technology.

3.1. Feasible 3D Packaging Technology for Our Design

Figure 3 depicts an overview of our GPU system architecture with integrated graphics memory. Different from conventional GPU systems (shown in Figure 2), which employ off-chip DRAMs as graphics memory, we integrate GPU processor and DRAMs with silicon interposer technology. Multiple layers of memory cells can be stacked on top of a logic layer with TSV technology. We estimate the silicon interposer area required by integrating DRAMs based on DRAM density. DRAM density at 50nm is 27.9Mb/mm^2 [Loh 2008]. The density becomes 43.6Mb/mm^2 if we scale the technology node to 40nm (we assume 40nm technology node in our evaluations in Section 6). In this case, we can integrate 6GB of DRAMs within 281.8mm^2 of silicon interposer area by stacking the DRAMs into four layers of memory cells and one logic layer.

Thermal tolerance can be another concern of processor-memory integration. We studied the integration of GPU processor and DRAMs with both vertical 3D memory stacking and silicon interposer technologies. We performed thermal analysis with a GPU system configuration based on NVIDIA Quadro[®] 6000 [NVIDIA 2010]. We computed the maximum power consumption of GPU processors and memory controllers by subtracting the DRAM power from the reported maximum power consumption of Quadro[®] 6000 [NVIDIA 2010], resulting in 136W. The power of 6GB

Fig. 3. Overview of the proposed 3D die-stacking graphics memory design. The proposed architecture, which is a 3D+2.5D system, where the DRAM memory itself is 3D stacked memory with TSV, whereas the integration of DRAM and GPU processor is through the interposer solution (2.5D). The number of memory I/Os is greatly increased. Multiple DRAM die slices can be stacked on top of a controller/interface circuitry layer with TSV technology.

DRAM is calculated as 68W based on Hynix's GDDR5 memory [Hynix 2009]. The areas of different GPU components are obtained from the GPU die photo, which has a 529mm² die area.¹ We assume the ambient temperature to be 40°C. We used the HotSpot thermal simulation tool [Skadron et al. 2004] to conduct our analysis. The maximum steady-state temperature of the GPU (without DRAMs) is 71.2°C. With 6GB interposer-mounted DRAMs (four layers of memory cells plus one layer of logic) placed beside the GPU processor as shown in Figure 3, the maximum temperature is 76.6°C. Thus, it is feasible to employ interposer-based memory integration. Vertically stacking memories on top of the GPU incurs much greater temperature increases than a silicon interposer-based approach. By stacking the same DRAMs on top of the GPU processor, the temperature rises to 83.8°C, a 7.2°C increase compared to the interposer-mounted DRAMs. Moreover, the temperature rise can further increase system-wide power consumption due to the temperature dependence of leakage power. Therefore, our design employs interposer-based memory integration.

3.2. Energy Efficiency Benefits of 3D Die-Stacking Graphics Memory

The key advantage of integrated DRAMs is the wide interface. Our primary approach is therefore to explore the system energy efficiency benefits provided by a wide-interface graphics memory. The data I/Os of conventional graphics memory standards (e.g., GDDR3 and GDDR5) are 16 or 32 bits wide per channel, limited by the package pin count. Our 3D die-stacking memory does not suffer from such limitations since the DRAMs are directly integrated with the GPU processor in the same package. Each memory channel can be connected to the corresponding memory controller via wide buses. We propose two different approaches to take advantage of the 3D die-stacking memory. First, we can leverage the VF scaling to reduce memory power consumption but still maintain peak memory bandwidth equivalent to conventional (2D/off-chip) graphics memories. Second, we can improve GPU system performance by using the higher memory bandwidth while respecting power constraints.

Before demonstrating these benefits, we first introduce the definition of peak memory bandwidth, which is the theoretical maximum data transfer rate. It is typically defined in units of bytes/second (B/s) and is computed by the following equation:

$$\text{Peak Memory Bandwidth} = N_{mem} \times W \times f_{mem} \times 2, \quad (1)$$

¹Quadro[®] 6000 consists of a single GF100 core, <http://www.anandtech.com/show/2918>.

Fig. 4. The peak memory bandwidth and maximum total DRAM power consumption of 12-channel×1Gb GDDR memory with various interface configurations. The clock frequency and supply voltage (VDDmax) of the memory interface are scaled according to per-channel bus width. DRAMs with 16-bit and 32-bit bus widths are evaluated as off-chip GDDR memory. The rest of configurations are evaluated as 3D die-stacking GDDR memory. Note that off-chip GDDR memory with 720GB/s bandwidth is not practical with today's memory technology, because the clock frequency will be as high as 7.5GHz. The figure shows memory power at various peak memory bandwidths.

where N_{mem} is the total number of memory channels, W is the number of I/Os per channel, and f_{mem} is the memory clock frequency. The final multiplier of $\times 2$ accounts for double-data rate signaling used by GDDR memories.

We examine the peak memory bandwidth and maximum total power consumption of graphics memory with different memory interface configurations, including per-channel bus widths, memory frequency, and supply voltage (Figure 4). The graphics memories considered are electrically similar to GDDR5 memory except for the wider buses.² The memory interface configuration is defined as a set of parameters (*bus width, frequency, Vdd*). The supply voltage (*Vdd*) is scaled appropriately to support the given memory interface clock frequency. DRAMs with per-channel bus widths of 16 and 32 bits are evaluated as off-chip GDDR memory. DRAMs with per-channel bus widths of 64 bits or wider are evaluated as 3D die-stacking memory. When computing the maximum power consumption, we use a memory bandwidth utilization of 0.35, which is the maximum observed ratio from our GPU applications. The GDDR memory power model that we used will be described in detail in Section 5.

The bars in Figure 4(a)–(e) show the maximum power consumption for several configurations with different bus widths and clock speeds, but maintaining the same peak memory bandwidth (i.e., bus width $\times$ clock speed is kept constant). Our 3D die-stacking graphics memory has a bus width several times wider than conventional GDDRs. Therefore, we can reduce the memory frequency and still achieve equivalent peak memory bandwidth as conventional GDDR memories. For example, we can reduce f_{mem} by one half and still maintain the same peak memory bandwidth by using a bus twice as wide as conventional GDDR. One opportunity for memory power reduction is

²Note that these memories are different from conventional CPU memories (e.g., DDR3). Graphics memories are designed with much tighter electrical specifications (e.g., directly soldered to boards, no DIMMs, no sockets) and therefore can run at significantly higher clock speeds than CPU memories.

Fig. 5. Different peak memory bandwidths that can be achieved at a fixed memory power budget. The standard GDDR5 with 32-bit interfaces can only provide 144GB/s of peak memory bandwidth. 3D die-stacking graphics memory can even provide higher bandwidth than the standard GDDR5 while consuming less power.

exploited by scaling down the memory's supply voltage corresponding to the frequency reduction. In addition, 3D die-stacking DRAMs do not require ODT [Vick et al. 2012]. Power consumption can be further reduced by eliminating the ODT resistors. As illustrated in Figure 4(a)–(e), the power consumption follows U-shaped bathtub curves at low peak memory bandwidths of 144GB/s, 180GB/s, and 288GB/s. With wider memory buses, lower frequency allows us to scale down the supply voltage, which directly results in a power reduction. However, the power consumed by I/O output drivers keeps increasing with the bus width. When the bus width is increased to 256 bits, the I/O power component starts to dominate the total memory power and finally overwhelms the power benefits of VF scaling. The optimal bus configuration is 128 bits wide. With a higher peak memory bandwidth of 360GB/s and 720GB/s, the memory power consumption will continue to decrease with 256-bit bus width. The optimal bus width is now achieved by even wider bus configurations.

Figure 5 shows the potential benefit of achieving higher peak memory bandwidth with 3D die-stacking graphics memory at the same or lower memory power consumption. With a fixed memory power budget of 35W, the standard GDDR5 with 32-bit interfaces can only provide 144GB/s of peak memory bandwidth. However, the wide-interface 3D die-stacking memories can achieve up to 360GB/s bandwidth. In addition, 128-bit and 256-bit bus width configurations can even provide higher bandwidth than the standard GDDR5 while consuming less power.

4. ENERGY AND PERFORMANCE OPTIMIZATIONS WITH RECONFIGURABLE MEMORY INTERFACE

Different applications have different memory access patterns, as well as different power and performance demands. The previous section demonstrated how given different bandwidth requirements, the optimal memory interface configuration varies. We now present a reconfigurable memory interface that can dynamically adapt to the demands of various applications based on dynamically observed memory access and performance information. We propose two reconfiguration mechanisms, EOpt and PerfOpt, that optimize the system energy efficiency and performance (in terms of throughput), respectively. To maximize the system energy efficiency, we configure the memory interface to minimize the DRAM power and maintain the system instructions per cycle (IPC) rate. To improve the system throughput under a given power budget, we co-optimize the memory configuration and the GPU clock frequency by shifting power saved from the memory interface over the GPU. Our study is conducted on two types of GPGPU applications: memory intensive and compute intensive. Our memory interface design can dynamically detect the different memory access patterns of the two types and apply different strategies to achieve our design goals.

Fig. 6. GPU system performance, power, and energy efficiency of two types of applications, with various peak memory bandwidths (from 144GB/s to 720GB/s). DRAMs with 32-bit per-channel bus width is evaluated as off-chip GDDR memory. The rest configurations are evaluated as 3D die-stacking GDDR memory. We study a GPU system with six memory controllers, each with two DRAM channels. (a) Type-C is nonmemory intensive (or compute intensive). (b) Type-M is memory intensive. The IPC of type-M is sensitive to the change of memory interface configuration. The IPC of type-C applications is much higher than type-M and is not sensitive to the change of memory interface configuration. Multithreading of GPU cores can hide the memory access latency very well.

4.1. Motivation: Application Classification

The memory interface with fixed bus width and frequency cannot satisfy different memory utilization requirements of various applications. Even a single application can have variable memory access patterns during execution. First of all, we analyze the unique characteristics of GPGPU applications so that our reconfigurable memory interface can be designed accordingly.

Figure 6 illustrates GPU system performance, power, and energy efficiency of memory-intensive and compute-intensive applications (the detailed results of the two benchmarks, *Merge Sort* [CUDASDK 2010] and *Needleman Wunsch* [Che et al. 2009], will be demonstrated in Section 6). Here, we define energy efficiency as the performance per Watt:

$$\text{Energy Efficiency} = \frac{IPC}{P_{system}} \quad (2)$$

As shown in Figure 6, higher peak memory bandwidth can directly lead to higher DRAM power consumption with both types of applications. The relationship between the system performance and memory interface configuration appears to be different

Fig. 7. A single application can have variable memory intensity during the execution. Even a memory-intensive application can have some compute-intensive periods. (An example application *BFS*.)

with the two types of applications. Figure 6(a) illustrates type-C (compute-intensive) applications. At the same peak memory bandwidth, the IPC curve remains stable with different memory interface configurations. With type-C applications, the GPU multithreading can hide the memory access latency and the instruction execution is not slowed down. Overall, varying the memory interface configuration will only affect the system power consumption. In this case, we can improve the system energy efficiency by reducing the DRAM power without significant performance degradation.

The type-M (memory-intensive) applications are shown in Figure 6(b). The IPC of type-M is sensitive to the change of memory interface configuration. Decreasing the memory frequency (and consequently increasing the memory access latency) results in significant IPC degradation, even though we provide wider buses to keep the same peak memory bandwidth. Furthermore, the IPC of these applications typically stays much lower than that of type-C, due to the continuous memory demands that significantly slow down the instruction execution. Overall, varying the memory interface configuration will affect both IPC and DRAM power consumption in type-M applications. Trade-offs between the two must be considered to optimize the system performance and energy efficiency.

If we take a closer look at the details of each application execution, both type-M and type-C periods can be observed in a single application (e.g., *BFS*), as illustrated in Figure 7. For example, the application appears to be memory intensive with frequent DRAM accesses during the instruction count between 220 million and 250 million. IPC is lower than 100 in this period. During the instruction count of 250 million to 270 million, the memory intensity is reduced and the IPC achieves 350. The rough analysis shows that this application is memory intensive due to the total amount of DRAM accesses. However, a nontrivial portion of the instruction execution is actually compute intensive. For the best system energy efficiency or performance, different memory configuration policies should be adopted with the two levels of memory intensities.

4.2. Reconfigurable Memory Interface Hardware Design

Our reconfigurable memory interface is aware of these different application behaviors, such that the bus width and memory frequency can be dynamically tuned according to the observed performance and memory access information of each workload. The hardware implementation of our reconfigurable memory interface and the overhead are discussed next.

Fig. 8. Hardware implementation of reconfigurable memory interface. (a) A central controller is used to collect performance and memory access information. (b) Connection between memory controller and 3D die-stacking graphics memory (one channel). At each channel, data bus width is initially set to 256 bits. (c) Data bus can be reconfigured to be narrower by switching off a subset of I/O drivers. Control signals are connected to the central controller.

Table I. Storage Overhead of Various Counters and Registers in Central Controller

Component	Instruction Counter	Cycle Counter	Read Counter	Write Counter	Register Vector	Arithmetic Storage
Width	64-bit	32-bit	32-bit	32-bit	64-bit	64-bit
Length	2	2	1	1	8	4
Storage	16B	8B	4B	4B	64B	32B

Hardware Implementation. Our energy-efficient reconfigurable memory interface hardware design is illustrated in Figure 8. We make several modifications to the interface between the GPU processor and the 3D die-stacking graphics memories, including adding a central controller, control signals to the bus drivers, and controls for DVFS. The central controller in Figure 8(a) is used to collect global information of both GPU performance and memory accesses. A vector of counters are maintained in the controller, including instruction counter, cycle counter, and memory access counters, to collect performance and memory access information from either GPU hardware performance counters or memory controllers. A threshold register vector is used to store various thresholds and initial values described in the reconfiguration mechanism. The calculator module calculates the system energy efficiency based on the collected performance information and the estimated power consumption. The results are stored in result registers for comparison. Figures 8(b) and (c) illustrate our data bus implementation. The basic topology of a bidirectional point-to-point data bus is a set of transmission lines, with transmitter and receiving devices at both ends of each bit. Control signals of I/O drivers are connected to the central controller. These control signals switch the drivers in the transmitters on and off to change the bus width.

Overhead. The reconfigurable memory interface incurs both area and performance overheads. The storage overhead is listed in Table I, including various counters, registers, and arithmetic logic components. The bus transmission lines are routed on the silicon interposer, which has sufficient space for the buses. To estimate the performance overhead of reconfiguration, we sweep the reconfiguration latency (the total latency of memory access pattern change detection and reconfiguration regardless of detailed reconfiguration mechanisms used) from 100 to 1K cycles. We find that performance

Table II. Performance Overhead of Reconfigurable Memory Interface with 1K-Cycle Latency of Each Reconfiguration

Benchmark	PF	HS	BP	MC	DWT	GS	CFD	MS	SP	LUD
Total cycles	759K	808K	498K	2.0M	2.4M	3.0M	26.5M	4.0M	4.7M	38M
Overhead	2.2%	2.0%	1.6%	0.8%	0.7%	0.5%	0.5%	0.4%	0.3%	0.3%

(Here, the total cycle counts include the reconfiguration latency.)

overhead of reconfiguration is lower than 2.5% of total execution time. Table II shows the estimated performance overheads of 10 benchmarks with 1K-cycle latency of each reconfiguration (our experimental configuration and benchmarks will be described in Section 5). With other benchmarks used in our evaluation, performance overheads of reconfiguration are even lower.

4.3. EOpt: Optimizing the System Energy Efficiency

As shown in Figure 6, the memory interface configuration will affect the system energy efficiency in both type-M and type-C applications. Therefore, a direct way of utilizing our reconfigurable memory interface is to optimize the system energy efficiency. Specifically, we strive to maintain performance that is competitive to a static memory interface approach but dynamically choose different memory configurations to save power when possible.

During type-M execution periods, both IPC and memory power consumption will be affected by the change of memory interface. Therefore, we choose configurations that maintain high memory clock frequencies to minimize the IPC degradation. Given the memory frequency constraint, the bus width is then configured to minimize the memory power consumption. During type-C execution periods, IPC is stable when we change the memory interface configuration. Consequently, we tend to adopt the memory frequency and bus width configuration that minimizes the memory power consumption.

Our reconfiguration mechanism for system energy efficiency optimization is composed of three steps: detection, comparison, and reconfiguration. During the execution of an application, we sample both IPC and the memory access count. If we detect a change of memory intensity, we compare the estimated system energy efficiency with different (*bus width, frequency*) configurations. The configuration that results in the highest system energy efficiency will be adopted.

Algorithm 1 describes our reconfiguration mechanism. In each instruction interval- i , we take a sample of cycle count $C(i)$ and compare it with $C(i - 1)$. If the difference between them is higher than a predefined threshold σ , we mark the current interval as a potential execution pattern change. To avoid frequent reconfigurations, the reconfigurable memory interface only reacts to a “real” execution pattern change identified by continuously marking K potential execution pattern changes. Upon a real execution pattern change, we calculate the memory intensity as the memory accesses per 1K instructions:

$$\text{Memory Intensity}(i) = \frac{R(i)}{I(i)} \times \frac{1}{1000}, \quad (3)$$

where $R(i)$ is the number of memory accesses in interval- i , and $I(i)$ is the number of executed instructions. If it is a compute-intensive interval, we simply configure the memory interface to minimize the estimated DRAM power at the same peak memory bandwidth. Otherwise, system energy efficiency is estimated and compared with different memory interface configurations. The energy efficiency of the current configuration $\eta(i, 0)$ is calculated as $N/(C(i)P(i, 0))$. Here $P(i, 0)$ is the system power, which is calculated using the power model described in Section 5.3. Since we cannot obtain the

ALGORITHM 1: EOpt—reconfiguration mechanism to optimize system energy efficiency.**Input:** The cumulative instruction count I , cycle count C , and memory access count R .**Output:** The reconfiguration signal s , indicating the switch of memory interface configuration.Initialization: $s \leftarrow 0$, $I(0) \leftarrow 0$, $I(temp) \leftarrow 0$, $C(0) \leftarrow 0$, $C(temp, 0) \leftarrow 0$, $C(temp, 1) \leftarrow 0$, $R(0) \leftarrow 0$, $R(temp) \leftarrow 0$, $\eta(a, 0) \leftarrow 0$, $\eta(b, 0) \leftarrow 0$.**repeat** **if** $C - C(temp, 0) \geq n$ **then** $I(i) \leftarrow I - I(temp)$; **if** $I(i) \geq N$ **then** $C(i) \leftarrow C - C(temp, 1)$; $C(temp, 1) \leftarrow C$; $R(i) \leftarrow R - R(temp)$; **if** $|C(i) - C(i-1)| > \sigma_1$ **and** $|R(i) - R(i-1)| > \sigma_2$ **then** **if** $k < K$ **then** $k \leftarrow k+1$; **else** **if** $R(i) < \sigma_m$ **then** **if** $P_m(0) > P_m(j)$ **then** $s \leftarrow \text{Config}(j)$; **end** **else** $\eta(i, 0) \leftarrow C(i)P(i, 0)$; **if** $\text{freq}(i, j) < \text{freq}(i, 0)$ **then** $\eta(i, j) \leftarrow (C(i)\Delta(j))P(i, j)$; **end** $\eta(i, j) \leftarrow (C(i)/\Delta(j))P(i, j)$; **if** $\eta(i, 0) < \eta(i, j)$ **then** $s \leftarrow \text{Config}(j)$; **end** **end** **end** **end** $I(temp) \leftarrow I$; **end** **end** $C(temp, 0) \leftarrow C$; $R(temp) \leftarrow R$;**until** $\text{ProgramEnd} == \text{true}$;

runtime performance and power information of other configurations $\eta(i, j)$, we adopt a simple but effective way to estimate the system energy efficiency. We find that the change of IPCs is nearly a constant with each memory-intensive application when we double the bus width or reduce the memory frequency by half. The constant varies with different applications. With the same application, the maximum observed deviation of this constant is 9.5% with all the evaluated applications. Therefore, we estimate $\eta(i, j)$ by $(I(i)/(C(i, 0)\Delta(j)))/P(i, 0)$ or $(I(i)/(C(i, 0)/\Delta(j)))/P(i, 0)$, based on whether we reduce or double the memory frequency in the configuration of $\text{Config}(i, j)$. If $\eta(i, j)$ is higher than current $\eta(i, 0)$ by more than a predefined threshold δ , we determine that switching to $\text{Config}(i, j)$ is beneficial in terms of energy efficiency, and so we reconfigure the memory interface. In our evaluation, we obtain different Δ of different applications by profiling each application for two instruction intervals after initialization with original and doubled memory frequencies, respectively. We use a simple 1-bit prediction scheme that yields sufficient accuracy without much performance overhead. More sophisticated prediction schemes could be incorporated into the reconfiguration mechanism, although most GPU applications do not frequently change their memory access patterns.

Fig. 9. PerfOpt—reconfiguration mechanism to optimize system performance under the given power budget. Type-M (memory-intensive) and type-C (compute-intensive) execution periods are addressed in different manners.

4.4. PerfOpt: Optimizing System Performance Under Given Power Budget

As long as the power consumption is affordable, the primary demand from the graphics and high-performance computing markets is high performance in terms of throughput. In this section, we explore GPU system performance optimization under a given power budget. Our performance optimization target is the instruction throughput—that is, the executed instructions per second.

Again, our reconfiguration mechanism addresses the type-M and -C execution periods of a workload with different strategies. During type-C (compute-intensive) execution periods, we always employ the memory interface configuration that minimizes the DRAM power consumption. Any power saved is transferred to scale up the GPU core clock frequency/supply voltage to improve the system performance. During type-M (memory-intensive) periods, we consider two strategies. First, because the memory interface configuration directly affects system performance (during type-M phases), we choose the memory configuration that delivers the highest system performance while staying within the system power budget. Second, sometimes an application can be a relatively memory-intensive phase while still having significant compute needs as well. In these cases, reconfiguring the memory interface to free up more power for the GPU can still result in a net performance benefit despite the reduced raw memory performance. Based on the predicted benefit, our system will choose the better of these two strategies.

Figure 9 shows the flow chart of our reconfiguration mechanism. At the end of each instruction interval, we evaluate the system power consumption and the memory intensity (Equation (3)). If the power budget constraint is violated, we reduce either memory interface bus width/frequency or GPU clock frequency. During type-M periods, decreasing the GPU clock frequency has priority, since both GPU and memory interface configurations are closely related to system throughput and the GPU processor contributes to the higher portion of power consumption. During type-C periods, we reduce the memory frequency and bus width and leave the GPU configuration to be stable. This policy will not affect performance significantly for compute-intensive periods. If we have sufficient extra power budget, we attempt to improve the system throughput. During type-C execution period, we simply scale up the GPU core clock frequency according to the amount of the extra power. If we detect a type-M period, one of the two strategies is to be applied. One strategy is to pick a memory interface configuration with the highest frequency allowed by the power budget. The other strategy is to

Table III. Baseline GPU Configuration
The parameters of streaming multiprocessors (SMs), caches, and memory controllers.

Number of SMs	16 (574MHz clock)	Warp size	32
CTAs per SM	8	Threads per SM	1536
Shared memory per SM	48KB	Registers per SM	32768
L1 caches per SM	Texture: 12KB; Data: 16KB	DRAM request queue size	16
Shared L2 cache	768KB	Memory controllers	6 (FR-FCFS)

Table IV. DRAM Configurations

Baseline is off-chip GDDR5 memory with 32-bit bus width per chip. The maximum bus width of 3D die-stacking DRAM is 256 bits per chip. The maximum clock frequency is 1.5GHz. The peak memory bandwidth of 3D die-stacking DRAM can be changed by scaling down the bus width and clock frequency.

Memory interface	Baseline GDDR5	3D die-stacking DRAM
Clock frequency	1.5GHz	375MHz–1.5GHz
Bus width per chip	32 bits	Maximum 256 bits
Bandwidth	144GB/s	Maximum 720GB/s
Memory timing	$t_{RAS}=28\text{ns}$, $t_{RP}=12\text{ns}$, $t_{RC}=40\text{ns}$, $t_{RCD}=12\text{ns}$, $t_{RRD}=5.5\text{ns}$	$t_{RAS}=22.4\text{ns}$, $t_{RP}=9.6\text{ns}$, $t_{RC}=32\text{ns}$, $t_{RCD}=9.6\text{ns}$, $t_{RRD}=4.4\text{ns}$

reduce the DRAM power consumption and reuse the extra power budget to increase the GPU clock frequency. We estimate the throughput improvement as $\frac{f_{new}}{f_{old}}$. The strategy with the better estimated throughput is adopted.

A possible issue with this reconfiguration mechanism is that the power budget may be violated within an instruction interval. Short-duration violations can be tolerated if there is sufficient thermal headroom. Existing CPU systems already “overclock” beyond their nominal power limits for short intervals [Intel 2012]. If violations last too long, on-chip thermal monitors can always trigger more aggressive VF scaling to prevent catastrophic damage to the chip [Intel 2005].

5. EXPERIMENTAL SETUP

5.1. Simulation Method

We use GPGPU-sim v.3.0 [Bakhoda et al. 2009], a cycle-accurate PTX-ISA simulator, to run our experiments. The simulator models shader cores, texture caches, constant caches, L1 caches, interconnection network, memory controllers, and DRAM memories. Tables III and IV specify the configurations and parameters used in our simulation. We assume that the system is implemented with 40nm process technology. We evaluate a GPU processor with 16 streaming multiprocessors (SMs). The SMs, caches, and memories are configured based on NVIDIA Quadro[®] 6000 [NVIDIA 2010]. We model a perfect crossbar interconnection network in the GPU processor with one-cycle latency so that the bandwidth demand at the processor-memory bus is not limited by the network bandwidth. We modify the simulator to implement our reconfiguration mechanisms.

Table IV lists the DRAM configuration parameters used in our simulation. Each memory controller has two DRAM channels. Therefore, we model a system with 12 DRAM channels. The baseline is off-chip GDDR5 graphics memory with 32-bit bus width per chip and 1.5GHz clock frequency. This is in line with the GDDR5 memory used by NVIDIA Quadro[®] 6000 [NVIDIA 2010]. The low-level memory timing of the baseline is obtained from the datasheet [Hynix 2009]. 3D die-stacking memory latency is the sum of DRAM core access latency, silicon interposer pin delay, intrapackage wiring delay, and memory controller traversal delay. Our 3D die-stacking memory employs 3D stacked DRAM dies. As reported in Tezzaron’s datasheets [Tezzaron 2010], the access latency (tRC) of a five-layer DRAM is only 67.5% of that of conventional 2D

Table V. Characteristics of Selected GPGPU Benchmarks
IC represents instruction count; MI represents memory intensity.

Abbrev.	Benchmarks	IC	MI	IPC	Power
MC	Monte Carlo [CUDASDK 2010]	1G	0.1	526	158W
MS	Merge Sort [CUDASDK 2010]	2G	0.5	522	191W
BN	Binomial Options [CUDASDK 2010]	11.8G	0.6	549	196W
CT	Texture Convolution [CUDASDK 2010]	5.4G	1.0	565	217W
MM	Matrix Multiplication [CUDASDK 2010]	836M	1.4	397	169W
SN	Sorting Networks [CUDASDK 2010]	5.6G	1.5	484	158W
HS	Hot Spot [Che et al. 2009]	297M	1.7	375	127W
NE	Nearest Neighbor [Che et al. 2009]	47M	1.8	109	148W
PF	Path Finder [Che et al. 2009]	72M	2.3	196	183W
SLA	Scan of Large Arrays [CUDASDK 2010]	15.4G	3.3	437	174W
DWT	Discrete Wavelet Transform [Che et al. 2009]	20M	3.4	411	152W
GS	Gaussian Elimination [Che et al. 2009]	9M	3.6	143	149W
HG	Histogram [CUDASDK 2010]	5G	3.7	491	217W
64H	64bin Histogram [CUDASDK 2010]	25.8G	3.9	423	204W
CS	Separable Convolution [CUDASDK 2010]	8G	4.1	428	212W
LUD	LU Decomposition [Che et al. 2009]	5.6G	4.8	150	161W
SD	Speckle Reducing Anisotropic Diffusion [Che et al. 2009]	2.4G	5.7	379	186W
BP	Back Propagation [Che et al. 2009]	190M	6.2	388	213W
CFD	CFD Solver [Che et al. 2009]	4.9G	8.5	187	197W
BLK	BlackScholes Option Pricing [Che et al. 2009]	9.5G	9.0	203	154W
FWT	Fast Walsh Transform [CUDASDK 2010]	4.4G	12.2	209	129W
SC	Streamcluster [Che et al. 2009]	2.8G	15.2	191	199W
SP	Scalar Product [CUDASDK 2010]	24M	19.0	259	163W
KM	K-means [Che et al. 2009]	6.1G	23.9	260	117W
NW	Needleman Wunsch [Che et al. 2009]	211M	27.6	29	173W
BFS	Breadth First Search [Che et al. 2009]	454M	81.9	62	140W

GDDR5 memories. Furthermore, the latency of signals passing through silicon interposer can be reduced to 1/5 of that with standard I/Os [Dorsey 2010]. Therefore, with 3D die-stacking DRAMs, we conservatively assume 20% memory latency reduction compared to off-chip GDDR5 memory. The refresh period of off-chip DRAM is 64ms. To account for higher leakage rates due to higher temperature operation, we assume a 32ms refresh period with 3D die-stacking DRAM. The maximum bus width of each 3D die-stacking DRAM chip is 256 bits. We evaluated the system energy efficiency with various peak memory bandwidths by varying the configuration of memory interface (bus width, clock frequency, and supply voltage). With the maximum clock frequency of 1.5GHz, 3D die-stacking graphics memory can provide 1152GB/s peak memory bandwidth. In this article, we only evaluated up to 720GB/s peak memory bandwidth. This is sufficient to show the benefit of our design.

5.2. Workloads

We evaluate various available GPU workloads from the NVIDIA CUDA SDK [NVIDIA CUDASDK 2010] and Rodinia Benchmarks [Che et al. 2009]. Table V lists the characteristics of our 26 workloads. The memory intensity of some applications, such as *MC*, *MS*, and *BN*, is lower than 1.0. The three most memory-intensive benchmarks are *KM*, *NW*, and *BFS*. We profiled our benchmarks by sweeping the instruction interval N from 1000 to 10 million. We found that we can catch the changes of memory intensity of all the benchmarks with 1 million instruction intervals. Further reducing the instruction interval can incur performance overhead by frequently invoking the reconfiguration

algorithms without really reconfiguring the memory interface. Therefore, we show the results with the 1 million instruction interval in Section 6.

5.3. Power Model

We model system power consumption of three subcomponents, including GPU cores and caches, memory controllers, and DRAMs. We calculate the power of GPU cores, caches, and memory controllers based on the power model from McPAT [Li et al. 2009]. We modified the power model to adapt to the configuration of GPU processor. We add GPU-specific power components to the power model, including warp schedulers and instruction buffers, the large register file, different types of caches, and the shared memory. The dynamic instruction execution and memory access information is fed into the power model to calculate the runtime power consumption of each component. We calculate the DRAM power based on the power model from Micron [Janzen 2010]. First, we calculate the maximum DRAM power with different interface configurations offline, assuming 100% memory bandwidth utility. The values are stored in the central controller. During the execution of an application, we obtain its runtime memory access statistics and calculate the real-time power consumption. With 3D die-stacking DRAM, ODT resistors can be eliminated [Vick et al. 2012]. Therefore, we only model the power consumption of I/O drivers.

6. RESULTS

In this section, we show our experimental results and discuss the reasons leading to these results.

6.1. Static Interface

Figure 10 shows the performance, power, and energy efficiency of the 3D die-stacking graphics memory with fixed bus widths during application execution. The different shaded bars correspond to the memory interface configuration that results in the maximum system energy efficiency at different peak memory bandwidths.

Our results of GPU system throughput (executed instructions per second), power consumption, and energy efficiency of the integrated graphics memories are normalized to the baseline of off-chip GDDR5 memory that supports a peak bandwidth of 144GB/s. At low peak memory bandwidths, some applications show system throughput losses compared to the baseline off-chip GDDR5 memory. This is due to the fact that at low-memory bandwidths, the wider buses provided by the 3D die-stacking graphics memory are offset by lowering their clock speeds. At a system bandwidth allocation of 144GB/s (i.e., providing no more bandwidth than the baseline off-chip memory solution), memory-intensive applications (e.g., *KM*, *NW*, and *BFS*) do incur some significant IPC degradations when using the 3D die-stacking graphics memory. Not surprisingly, the compute-intensive applications (e.g., *MC* and *MS*) do not suffer from the reduction in memory interface clock speed. The 3D die-stacking memory, however, provides a much more power-efficient implementation (middle plots), which in turn leads to better energy efficiency (performance per Watt, shown in bottom plots).

Of course, fixing the system bandwidth to be equal to the off-chip solution does not really take advantage of the wide interface provided by the 3D die-stacking memory. By increasing the memory interface clock speed to provide bandwidths of 360GB/s and 720GB/s (note even at these higher bandwidths, the power consumption of 3D die-stacking memory can still be lower than the off-chip GDDR5 due to the lower clock frequency), performance on the memory-intensive applications can be brought back up. The DRAM power consumption also increases slightly, but the overall energy efficiency increases when the peak memory bandwidth is increased from 288GB/s to 360GB/s, and then goes back down as the highest bandwidth configurations do not provide

Fig. 10. Results of 3D die-stacking graphics memory without reconfiguration, normalized to the baseline off-chip GDDR5 (represented by the dash lines). (a) System throughput (executed instructions per second). (b) System power. (c) System energy efficiency. (d) The bus width configurations that provide the highest system energy efficiency at each peak memory bandwidth. The clock frequency and supply voltage of the memory interface are scaled according to the bus width per chip. The mean improvement of system energy efficiency is up to 73%.

much additional performance while continuing to increase DRAM power. Overall, the results show that even with a static (no reconfiguration) 3D die-stacking graphics memory solution, the energy efficiency (performance per Watt) of the GPU system can be improved by up to 21%.

6.2. Reconfiguring for Energy Efficiency

Figure 11 shows GPU system throughput, power, and energy efficiency results of reconfigurable memory interface optimized for energy efficiency. Since we do not have any initial information about an application, the initial memory interface configuration is always set to a 256-bit bus width. All results are normalized to the case of 3D die-stacking graphics memory using static 256-bit interfaces to demonstrate the additional benefit of dynamic reconfiguration on top of the benefits of integrated memories. The different shaded bars represent the results at different peak memory bandwidths. Here, we only plot the results of 144GB/s and 180GB/s due to limited space. The last bar shows the results of reconfiguration across various peak memory bandwidths.

Figure 11(a) shows the system throughput of each benchmark. The reconfiguration mechanism yields the greatest throughput improvement for the most memory-intensive applications. In terms of performance, the initial configuration is not optimal for memory-intensive execution periods. During these periods, the memory interface is

Fig. 11. Results of using EOpt to all benchmarks, normalized to the static memory interface configuration with 256-bit bus width at different peak memory bandwidths. The last shaded bar represents that the memory interface can be configured to provide any peak memory bandwidth. The results of this bar is normalized to the 256-bit bus configuration that results in the highest average system energy efficiency at 144GB/s peak memory bandwidth across all of the benchmarks. (a) System throughput (executed instructions per second). (b) System power. (c) System energy efficiency. Overall, the system energy efficiency is improved by up to 26%, on average.

reconfigured to narrower buses and higher frequencies. Therefore, the throughput of memory-intensive applications are significantly improved. The throughput of compute-intensive applications are not affected by the change of memory interface, and execution pattern detection may cause performance overhead. Fortunately, most applications do not incur frequent execution pattern changes, and the throughput remains stable on the compute-intensive applications. Figure 11(a) also demonstrates that reconfiguration across various bandwidths can lead to the highest throughput improvement, although reconfiguring among larger configuration space can incur higher performance overhead than among smaller configuration space. With memory-intensive applications, reconfiguring among various peak memory bandwidth configurations can lead to 5% and 12% higher throughput improvement compared to 144GB/s and 180GB/s configurations, respectively.

Figure 11(b) shows that system power with almost all applications is reduced by an average of up to 12%. Figure 11(c) illustrates that the overall system performance-per-Watt rate is improved for all the benchmarks. The improvement of the compute-intensive applications (16%) is not as great as for the memory-intensive applications (44%). The reason is that the system throughput is significantly improved with memory-intensive applications but almost stays the same with compute-intensive applications. We observed that EOpt improves system energy efficiency of all baseline configurations, including those peak memory bandwidth configurations that are not plotted in the figure. Across all low- and high-intensity applications, the performance per Watt improves by 26%, on average.

Fig. 12. The fraction of instructions spent on different configuration modes, using PerfOpt under the power budget of 220W. Even the most memory-intensive applications (such as *KM*, *NW*, and *BFS*) have a portion of less memory-intensive periods, which we can utilize to improve system performance.

Fig. 13. Throughput improvement under various GPU system power budgets, using PerfOpt. The baseline (represented by the dash line) is the static configuration that provides the best average throughput of all benchmarks under each fixed power budget.

We also compared EOpt to an oracle reconfiguration algorithm, in which application activities are known at the beginning of execution and memory interface frequency is set to the optimum value accordingly. Due to the moderate performance overhead of reconfiguration as shown in Table II, we found that EOpt incurs a maximum of 2.2% overhead of system throughput compared to the oracle algorithm across all our benchmarks with variable peak memory bandwidths. EOpt leads to a maximum of 1.8% increase in system power consumption. As a result, EOpt leads to at most a 4% energy efficiency reduction.

6.3. Reconfiguring to Increase Performance Under a Fixed Power Budget

Figure 12 illustrates the fraction of instructions spent on each configuration mode under the power budget of 220W (a similar trend can be observed with other power budgets). Even the most memory-intensive applications have a portion of less memory-intensive periods, which we can utilize to improve system performance. For example, *KM* spends about 49% of its instructions with high-memory clock speed to avoid system performance degradation (strategy-1). In 25% of the instructions, the memory clock is slowed down by strategy-2 to increase the GPU clock frequency. The rest (26%) of instructions intervals are type-C periods, in which memory power is minimized so that we can use the saved power budget to improve system throughput by scaling up GPU processor clock frequency. It is also shown that the configuration modes change from one to another for most applications. The exceptions are those compute-intensive applications, such as *MC*, *MS*, *BN*, *CT*, *MM*, *SN*, *HS*, *NE*, and *PF*, which stay in type-C configuration all of the time.

Figure 13 and Table VI show the results of reconfiguration for improving overall GPU throughput given various system power budgets. All of the performance results are normalized to the mean throughput of the static configuration that leads to the highest average performance with all applications under the given power budget. It can be observed that the amount of improvement is different for the applications with different memory intensities. For compute-intensive applications, a higher power budget directly leads to more performance improvement. Since we always configure the

Table VI. Mean Throughput Improvement of All the Tested Benchmarks for Each Given GPU System Power Budget

Budget (W)	180	190	200	210	220	230	240
Improvement	15%	17%	20%	23%	25%	28%	31%
Ratio of Ideal Performance	99.7%	99.7%	99.2%	98.8%	98.5%	98%	97.7%

memory interface to minimize the DRAM power, extra power is available to increase GPU core clock frequency. Although the cores may request memory accesses at a higher rate, the multithreading can still handle memory latency well without much performance degradation. The throughput improvement of the three most memory-intensive applications—*KM*, *NW*, and *BFS*—is not as significant as the compute-intensive applications. This is because a significant portion of the power budget is consumed by the DRAM for these memory-intensive applications. However, we still observe an average of 8% throughput improvement with these three most memory-intensive applications under the power budget of 220W and more improvement with higher power budgets. Table VI shows the mean throughput improvement under various power budgets that we have evaluated. The results show that the reconfiguration mechanism can adjust the memory power consumption to fit the application memory needs, and that the saved power can be effectively redeployed to improve the GPU core performance up to 31%. The last row of Table VI shows our PerfOpt compared to the oracle algorithm with known application activities at the beginning of execution. It is shown that PerfOpt can provide a close to ideal performance due to the moderate performance overhead of our reconfigurable memory interface.

In addition, we can use even reduced power budgets for some of applications. This is not possible with static memory interface, because some intervals will violate the low power budget. With PerfOpt, the memory interface can be reconfigured to consume lower power than the given budget in these intervals. For example, we can apply a power budget as low as 130W for *HG* and *64H*, using PerfOpt. With static memory interface, the lowest power budget that can be used is 160W with both applications.

7. CONCLUSION

We have designed a reconfigurable 3D die-stacking wide-interface graphics memory, integrated with a high-performance GPU on a silicon interposer. Two reconfiguration mechanisms are proposed to effectively optimize system energy efficiency and performance for both memory-intensive and compute-intensive applications. Our design is feasible and easy to implement. Almost all hardware modification is limited to the memory interface and controllers, and no modification is required to the internal structures of the processors and the graphics memory arrays. Reconfiguration is only applied to the memory interfaces, and internal bus widths are fixed. Therefore, the main extra manufacturing cost is the packaging cost. Another merit of our design is the flexibility of optimizing either power or performance. With high-performance computing systems, performance may be the primary design consideration. Power-constrained applications, however, will prefer high energy efficiency. Our two reconfiguration mechanisms can be employed by both types of systems.

We see several interesting avenues for continuing study related to this work. First of all, we fixed the graphics memory capacity in this study. With 3D memory stacking, we can potentially extend the memory capacity. The impact of larger capacities and the structure of 3D stacking can be studied. Furthermore, it might be interesting to explore the benefits and limitations of employing shared wide-interface DRAMs in CPU-GPU heterogeneous systems. To fully utilize the bandwidth provided by the wide

buses, both memory interface and memory hierarchy designs should be aware of the different memory access patterns of CPU and GPU workloads.

REFERENCES

- AKAZAWA, M., KURAMOCCHI, S., MARUYAMA, T., NAKAYAMA, K., TAKANO, A., YAMAGUCHI, M., AND FUKUOKA, Y. 2003. High-density packaging technologies on silicon substrates. In *Proceedings of the Electronic Components and Technology Conference*. 647–651.
- AL MAASHRI, A., SUN, G., DONG, X., NARAYANAN, V., AND XIE, Y. 2009. 3D GPU Architecture Using Cache Stacking: Performance, Cost, Power and Thermal Analysis. In *Proceedings of the International Conference on Computer Design*. 254–259.
- AMD. 2012. AMD Radeon™ HD 7970 Graphics. <http://www.amd.com/us/products/desktop/graphics/7000/7970/Pages/radeon-7970.aspx>.
- ANDRY, P. S., TSANG, C., SPROGIS, E., PATEL, C., WRIGHT, S. L., WEBB, B. C., BUCHWALTER, L. P., MANZER, D., HORTON, R., POLASTRE, R., AND KNICKERBOCKER, J. 2006. A CMOS-compatible process for fabricating electrical through-vias in silicon. In *Proceedings of the Electronic Components and Technology Conference*. 1–7.
- BAKHODA, A., YUAN, G. L., FUNG, W. W. L., WONG, H., AND AAMODT, T. M. 2009. Analyzing CUDA workloads using a detailed GPU simulator. In *Proceedings of the International Symposium on Performance Analysis of Systems and Software*. 163–174.
- CHE, S., BOYER, M., MENG, J., TARJAN, D., SHEAFFER, J. W., LEE, S., AND SKADRON, K. 2009. Rodinia: A benchmark suite for heterogeneous computing. In *Proceedings of the International Symposium on Workload Characterization*. 44–54.
- CUDASDK. 2010. GPU Computing SDK. (2010). <https://developer.nvidia.com/gpu-computing-sdk>.
- DAVID, H., FALLIN, C., GORBATOV, E., HANE BUTTE, U. R., AND MUTLU, O. 2011. Memory power management via dynamic voltage/frequency scaling. In *Proceedings of the International Conference on Autonomic Computing*. 31–40.
- DENG, D., MEISNER, D., BHATTACHARJEE, A., WENISCH, T. F., AND BIANCHINI, R. 2012. CoScale: Coordinating CPU and memory system DVFS in server systems. In *Proceedings of the International Symposium on Microarchitecture*. 143–154.
- DENG, Q., MEISNER, D., RAMOS, L., WENISCH, T. F., AND BIANCHINI, R. 2011. MemScale: Active low-power modes for main memory. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*. 225–238.
- DONG, X., XIE, Y., MURALIMANOHAR, N., AND JOUPPI, N. P. 2010. Simple but effective heterogeneous main memory with on-chip memory controller support. In *Proceedings of the International Conference for High Performance Computing*. 1–11.
- DORSEY, P. 2010. Xilinx stacked silicon interconnect technology delivers breakthrough FPGA capacity, bandwidth, and power efficiency. Xilinx White Paper: Virtex-7 FPGAs, WP 380, October 27, 2010, 1–10.
- ELPIDA. 2010. Introduction to GDDR5 SGRAM. <http://www.elpida.com/pdfs/E1600E10.pdf>.
- GALAL, S. AND HOROWITZ, M. 2011. Energy-efficient floating-point unit design. *IEEE Trans. Comput.* 60, 7, 913–922.
- GEBHART, M., JOHNSON, D. R., TARJAN, D., KECKLER, S. W., DALLY, W. J., LINDHOLM, E., AND SKADRON, K. 2011. Energy-efficient mechanisms for managing thread context in throughput processors. In *Proceedings of the International Symposium on Computer Architecture*. 235–246.
- GU, S. Q., MARCHAL, P., FACCHINI, M., WANG, F., SUH, M., LISK, D., AND NOWAK, M. 2008. Stackable memory of 3D chip integration for mobile applications. In *Proceedings of the International Electron Devices Meeting*. 1–4.
- HERBERT, S. AND MARCULESCU, D. 2007. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 38–43.
- HYNIX. 2009. Hynix GDDR5 SGRAM datasheet, [http://www.hynix.com/datasheet/pdf/graphics/H5GQ1H24AFR\(Rev1.0\).pdf](http://www.hynix.com/datasheet/pdf/graphics/H5GQ1H24AFR(Rev1.0).pdf).
- INTEL. 2005. Thermal protection and monitoring features: A software perspective. In *Intel Software Network*. 1–6.
- INTEL. 2012. Turbo boost technology 2.0. <http://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost>.
- ISCI, C., CONTRERAS, G., AND MARTONOSI, M. 2006. Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *Proceedings of the International Symposium on Microarchitecture*. 359–370.

- JANZEN, J. 2010. The Micron power calculator. <http://www.micron.com/products/dram/syscalc.html>.
- JIAO, Y., LIN, H., BALAJI, P., AND FENG, W. 2010. Power and performance characterization of computational kernels on the GPU. In *Proceedings of the International Conference on Green Computing and Communications*. 221–228.
- KAXIRAS, S. AND MARTONOSI, M. 2009. *Computer Architecture Techniques for Power-Efficiency*, Morgan and Claypool.
- KGIL, T., D'SOUZA, S., SAIDI, A., BINKERT, N., DRESLINSKI, R., MUDGE, T., REINHARDT, S., AND FLAUTNER, K. 2006. PicoServer: Using 3D stacking technology to enable a compact energy efficient chip multiprocessor. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*. 117–128.
- KHAN, N., YOON, S. W., VISWANATH, A. G. K., GANESH, V. P., RANGANATHAN, D. W., LIM, S., AND VAIDYANATHAN, K. 2006. Development of 3D stack package using silicon interposer for high power application. In *Proceedings of the Electronic Components and Technology Conference*. 1–5.
- KIM, J., OH, C. S., LEE, H., LEE, D., HWANG, H., HWANG, S., NA, B., MOON, J., KIM, J., PARK, H., RYU, J., PARK, K., KANG, S., KIM, S., KIM, H., BANG, J., CHO, H., JANG, M., HAN, C., LEE, J., KYUNG, K., CHOI, J., AND JUN, Y. 2011a. A 1.2V 12.8GB/s 2Gb mobile Wide-I/O DRAM with 4×128 I/Os using TSV-based stacking. In *Proceedings of the IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 496–498.
- KIM, N., WU, D., KIM, D., RAHMAN, A., AND WU, P. 2011b. Interposer design optimization for high frequency signal transmission in passive and active interposer using through silicon via (TSV). In *Proceedings of the Electronic Components and Technology Conference*. 1160–1167.
- LI, S., AHN, J. H., STRONG, R. D., BROCKMAN, J. B., TULLSEN, D. M., AND JOUPPI, N. P. 2009. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the International Symposium on Microarchitecture*. 469–480.
- LIU, C. C., BURTSCHER, M., AND TIWARI, S. 2005. Bridging the processor-memory performance gap with 3D IC technology. *IEEE Des. Test* 22, 6, 556–564.
- LOH, G. H. 2008. 3D-stacked memory architectures for multi-core processors. In *Proceedings of the International Symposium on Computer Architecture*. 453–464.
- LOI, G. L., AGRAWAL, B., SRIVASTAVA, N., LIN, S., SHERWOOD, T., AND BANERJEE, K. 2006. A thermally-aware performance analysis of vertically integrated (3-D) processor-memory hierarchy. In *Proceedings of the Design Automation Conference*. 991–996.
- MICRON. 2013. Hybrid Memory Cube specification 1.0. http://hybridmemorycube.org/files/SiteDownloads/HMC_Specification%201_0.pdf.
- NVIDIA. 2008. PowerMizer 8.0 Intelligent Power Management Technology. Tech. brief. http://www.nvidia.com/object/feature_powermizer.html.
- NVIDIA. 2010. Quadro 6000—Workstation graphics card for 3D design, styling, visualization, CAD, and more. <http://www.nvidia.com/object/product-quadro-6000-us.html>.
- REN, D. Q. AND SUDA, R. 2010. Modeling and optimizing the power performance of large matrices multiplication on multi-core and GPU platform with CUDA. In *Proceedings of the International Conference on Parallel Processing and Applied Mathematics*. 421–428.
- SAMSUNG. 2010. DDR3 and GDDR5. <http://www.samsung.com/global/business/semiconductor/products/Products.html>.
- SKADRON, K., STAN, M. R., SANKARANARAYANAN, K., HUANG, W., VELUSAMY, S., AND TARJAN, D. 2004. Temperature-aware microarchitecture: Modeling and implementation. *ACM T. Archit. Code Op.* 1, 1, 94–125.
- SUNOHARA, M., TOKUNAGA, T., KURIHARA, T., AND HIGASHI, M. 2008. Silicon interposer with TSVs (through silicon vias) and fine multilayer wiring. In *Proceedings of the Electronic Components and Technology Conference*. 847–852.
- TEZZARON. 2010. FaStack 3D stackable DRAM. http://www.tezzaron.com/memory/FaStack_memory.html.
- VICK, E., GOODWIN, S., CUNNINGHAM, G., AND TEMPLE, D. S. 2012. Vias-last process technology for thick 2.5D Si interposers. In *Proceedings of the 3D Systems Integration Conference*. 1–4.
- WANG, P., CHENG, Y., YANG, C., AND CHENG, Y. 2009. A predictive shutdown technique for GPU Shader processors. *IEEE Comput. Archit. Lett.* 8, 1, 9–12.
- WOO, D. H., SEONG, N. H., LEWIS, D. L., AND LEE, H. S. 2010. An optimized 3D-stacked memory architecture by exploiting excessive, high-density TSV bandwidth. In *Proceedings of the International Conference for High Performance Computing*. 1–12.
- WU, Q., MARTONOSI, M., CLARK, D. W., REDDI, V. J., CONNORS, D., WU, Y., LEE, J., AND BROOKS, D. 2005. A dynamic compilation framework for controlling microprocessor energy and performance. In *Proceedings of the International Symposium on Microarchitecture*. 271–282.

- YU, W. S., HUANG, S. Q., WANG, S., KAN, E., AND SUH, G. E. 2011. SRAM-DRAM hybrid memory with applications to efficient register files in fine-grained multi-threading. In *Proceedings of the International Symposium on Computer Architecture*. 247–258.
- ZHAO, J., SUN, G., LOH, G. H., AND XIE, Y. 2012. Energy-efficient GPU design with reconfigurable in-package graphics memory. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 403–408.

Received April 2013; revised July 2013; accepted August 2013