Energy-efficient GPU Design with Reconfigurable In-package Graphics Memory

Jishen Zhao[1], Guangyu Sun[2], Gabriel H. Loh[3], and Yuan Xie[1,3]

[1]The Pennsylvania State University
Dept. of Computer Science and Engineering

{juz138, yuanxie}@cse.psu.edu

[2]Peking University
Center for Energy-efficient
Computing and Applications

gsun@pku.edu.cn

[3]Advanced Micro Devices, Inc.
AMD Research

gabriel.loh@amd.com

ABSTRACT

We propose an energy-efficient reconfigurable in-package graphics memory design that integrates wide-interface graphics DRAMs with GPU on a silicon interposer. We reduce the memory power consumption by scaling down the supply voltage and frequency while maintaining the same or higher peak bandwidth. Furthermore, we design a reconfigurable memory interface and propose two reconfiguration mechanisms to optimize system energy efficiency and throughput. The proposed memory architecture can reduce memory power consumption up to 54%, without reconfiguration. The reconfigurable interface can improve system energy efficiency by 23% and throughput by 30% under a power budget of 240W.*

Categories and Subject Descriptors

B.3.1 [**Memory Structures**]: Semiconductor Memories

Keywords

Graphics Memory, 3D Packaging, Reconfigurable Interface

1. INTRODUCTION

Modern GPU systems have become an attractive solution for both graphics and general-purpose workloads that demand high computational performance. The graphics processing unit (GPU) exploit extreme multi-threading to target high-throughput [1, 15]. For example, the AMD Radeon[TM] HD 5870 employs 31,000 threads interleaved across 1600 stream processors [1]. To accommodate the 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 off-chip memory (from hundreds of megabytes to gigabytes). Consequently, off-chip memory consumes a significant portion of power in a GPU system. Figure 1 evaluates the maximum power consumption of two GPU systems, the AMD Radeon[TM] HD 6990 [1] and NVIDIA Quadro® FX5800 [15] with the memory power model described in Section 5. We considered the
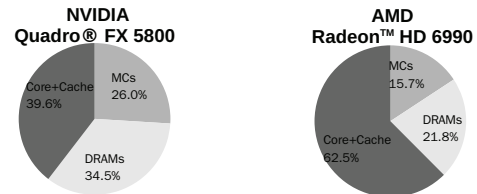
Figure 1: Power breakdown of NVIDIA and AMD GPUs.

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 GPU systems, the off-chip memory consumes from 20% to over 30% of the total GPU system power. Note that for the workloads evaluated in this paper, the highest average bandwidth utilization observed was 35%. At this bandwidth level, the memory power consumption is 21.8% and 34.5% for the two GPU systems, respectively. These results match well with results provided by an industry partner, which indicated that the graphics memory consumes ∼25% of the total GPU power. 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 quite significant. For example, the maximum power consumption of the AMD Radeon[TM] HD 6990 is 375W; therefore a 12.5% power reduction saves 47W. Consequently, techniques that reduce the graphics memory power can be very effective at reducing the total system power.

In this paper, 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 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 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 in-package 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 prefers better energy efficiency.

2. BACKGROUND AND RELATED WORK

Energy-efficient GPU Architectures: Various existing work explores how to address the GPU power challenge. Most of the existing studies explore either GPU shader cores and caches architecture [7,17], or software optimization [13], and require both hardware and software modifications to current GPU processor design. In our work, we explore power reduction techniques by limiting the archi-
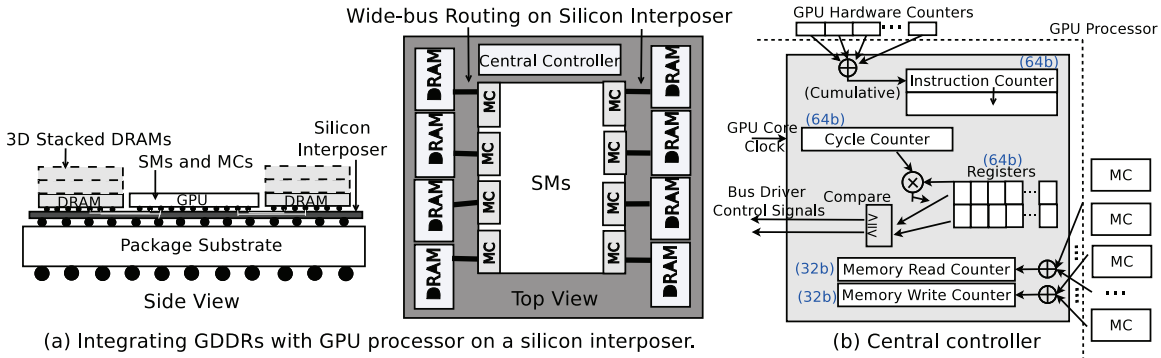
Figure 2: Overview of the proposed in-package graphics memory design.

tectural modifications to the graphics memory interface with only minor changes to GPU compute-unit architecture.

Graphics memory: Graphics double data rate (GDDR) memories are specifically designed for graphics cards, game consoles, and high-performance computing systems. GDDR memories usually employ high frequencies to achieve very high bandwidths. Unfortunately, power consumption dramatically increases as well. A large power supply is required, and usually comes with high cost. Conventional GDDRs have employed VF scaling techniques to reduce power by adapting the memory interface to the memory bandwidth requirement of an application [6]. However, the power reduction comes at the expense of memory bandwidth degradation. For future high-performance GPGPU and advanced video processing (e.g., 3D HD multi-screen video games), existing power saving techniques may not suffice.

3D Package Using Silicon Interposer Technology: The new packaging approaches with silicon interposer attract much attention in high-performance multi-chip module (MCM) and SiP designs. A variety of work from academia and industry explores the development of MCM and SiP with silicon interposers [4, 5, 16]. A silicon interposer provides high-density interconnections, and is capable of providing a line width of less than $10\mu m$ and thousands of I/O pads per square centimeter. The coarse-pitch through-silicon vias (TSVs) provide the external connections between the package and individual dies/chips for the parallel and serial I/O, power/ground, clocking, configuration signals, etc. The length of the interconnection from the chip to the substrate is reduced. In our work, we leverage silicon interposer-based 3D packaging technology to explore the energy efficiency benefits for GPU systems.

3. IN-PACKAGE GRAPHICS MEMORY ARCHITECTURE

Many power-saving techniques for GPU memory come with the undesirable side effect of a degradation of the provided memory bandwidth [6]. In this section, we present the feasibility results and energy efficiency benefits of in-package 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.

Figure 2 depicts an overview of our GPU system architecture with integrated graphics memory. The data I/Os of conventional graphics memory standards are 16 or 32 bits wide per channel, limited by the package pin count. Our in-package memory does not suffer from such limitations since the DRAMs are directly integrated with the GPU processor in the same package. In addition, the number of power/ground pins can be reduced at the much lower DRAM power consumption than conventional GDDRs. While we fix the DRAM

capacity in this work, the capacity of graphics memory can be extended by stacking multiple DRAM dies together.

3.1 Feasible Design with 3D Packaging

DRAMs are integrated on the GPU processor package with silicon interposer technology. Our thermal analysis results showed that such integration does not incur significant temperature increase to the system. We studied a GPU system configuration based on NVIDIA Quadro® FX5800 [15]. 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® FX5800 [15], resulting in 124W. The power of 4GB DRAM is calculated as 60W, based on Hynix's GDDR5 memory [8]. The areas of different GPU components are obtained from the GPU die photo, which has a $470\,mm^2$ die area. We assume the ambient temperature to be $40\,°C$. We used the HotSpot thermal simulation tool [14] to conduct our analysis. The maximum steady-state temperature of the GPU (without DRAMs) is $72.6\,°C$. With 4GB interposer-mounted DRAMs placed beside the GPU processor, the maximum temperature is $78.4\,°C$. Thus, it is feasible to employ interposer-based memory integration.

3.2 Energy Efficiency Benefits

We examine the peak-memory bandwidth (PBW) [2] and maximum total power consumption of 2GB graphics memories with different memory interface configurations, including per-channel bus widths (abbreviated as bus width in the following), memory frequency, and supply voltage (Figure 3). The memories considered are electrically similar to GDDR DRAMs except for the wider buses. The supply voltage is scaled appropriately to support the given memory clock frequency. DRAMs with bus widths of 16 and 32 bits per channel are evaluated as off-chip GDDR memory. DRAMs with wider buses are evaluated as in-package memory. The GDDR memory power model that we used will be described in detail in Section 5.

The bars in Figure 3(a)-(e) show the maximum power consumption for configurations with different bus widths and clock speeds, but maintaining the same PBW (i.e., bus-width × clock speed is kept constant). The opportunity for memory power reduction is exploited by scaling down the memory's supply voltage corresponding to the frequency reduction. It can be observed that the power consumption follows U-shaped bathtub curves. 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 and on-die termination 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. Figure 3(f) shows the potential benefit of achieving higher PBW with the in-package graphics memory at the same or lower memory power consump-
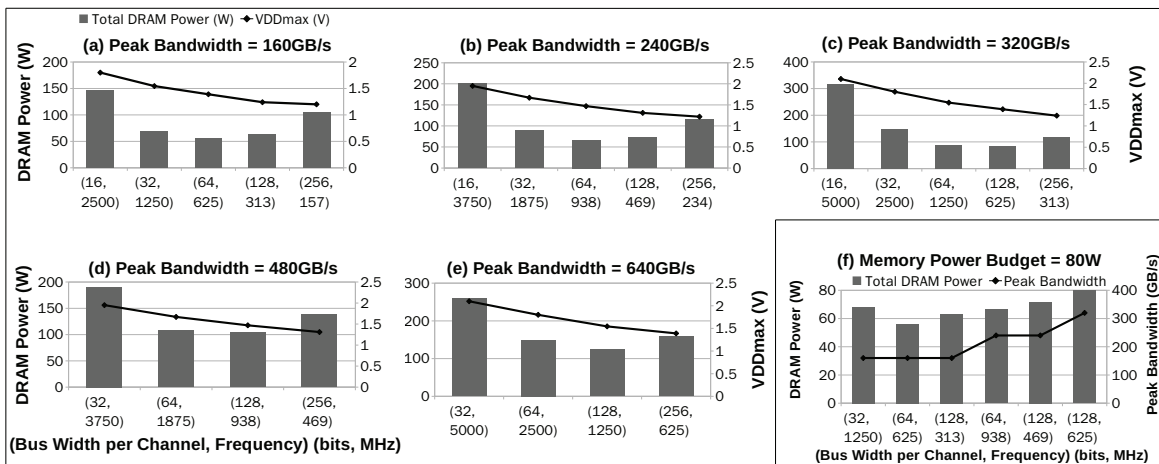
Figure 3: The PBW and maximum total memory power consumption of 16×1Gb GDDRs with various interface configurations of bus width and memory clock frequency. (a)-(e) show memory power at the same PBW. (f) shows different PBW that can be achieved at a fixed memory power consumption.

tion. With a fixed memory power consumption of 80W, the standard GDDR5 with 32-bit interfaces can only provide 160GB/s of PBW. However, the wide interface in-package memories can achieve up to 320GB/s bandwidth. In addition, the (64, 938) configuration can even provide higher bandwidth than the standard GDDR5 while consuming less power.

4. RECONFIGURABLE ARCHITECTURE

In this section, we present a reconfigurable memory interface that can dynamically adapt to the demands of various applications based on dynamically observed memory access and performance information. On top of the reconfigurable memory interface hardware, we further propose two reconfiguration mechanisms, EOpt and PerfOpt, that optimize system energy efficiency and performance (in terms of throughput), respectively.

4.1 Application Classification

We analyze the unique characteristics of GPGPU applications, and classify the applications into two types, memory-intensive and non-memory-intensive. Figure 5 illustrates the typical GPU system performance, power, and energy efficiency (applications *nw* and *sd2* are used as examples). Here we define energy efficiency as the performance per Watt. It is shown that higher PBW 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 with the two types of applications. The IPC of type-M (memory-intensive) is sensitive to the change of memory interface configuration, as shown in Figure 5(a). 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 PBW. Furthermore, the IPC of these applications typically stays low, due to the continuous memory demands that significantly slow down the instruction execution. As shown in Figure 5(b), the IPC of type-C (non-memory-intensive or compute-intensive) applications is much higher than that of type-M. The IPC curve remains stable with different memory interface configurations at the same PBW. 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 energy efficiency. With type-C applications, we can improve the system energy efficiency by reducing the DRAM power without significant performance degradation. If we take a closer look at the details of each
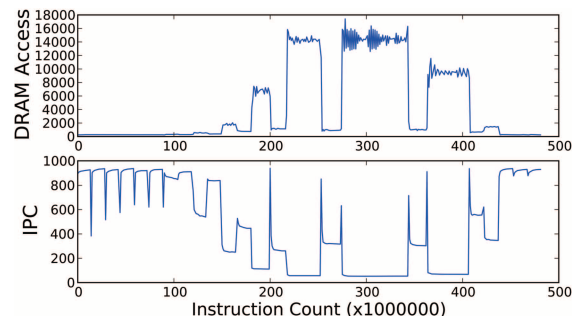

Figure 4: A single application can have different memory intensities during the execution.

application execution, both type-M and type-C periods can be observed in a single application (e.g., *bfs*) as illustrated in Figure 4. The rough analysis shows that this application is memory-intensive, due to the total amount of DRAM accesses. However, a non-trivial portion of the instruction execution is actually non-memory-intensive. For the best system energy efficiency or performance, different memory configurations should be adopted with the two levels of memory intensities.

4.2 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.

Hardware implementation: At the interface between the GPU processor and the in-package graphics memories, we add a central controller, control signals to the bus drivers, and controls for dynamic VF scaling (Figure 2(b)). The central controller is used to collect global information of GPU performance and memory accesses. A vector of counters are maintained in the controller 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 reconfiguration mechanism. The calculator module calculates system energy efficiency based on the collected performance information and the estimated power consumption. The memory intensity is calculated as *the memory accesses per 1000 instructions*.

Overhead: The reconfigurable memory interface incurs both area and performance overheads. The total storage overhead is 128B, including various counters, registers, and arithmetic logic compo-
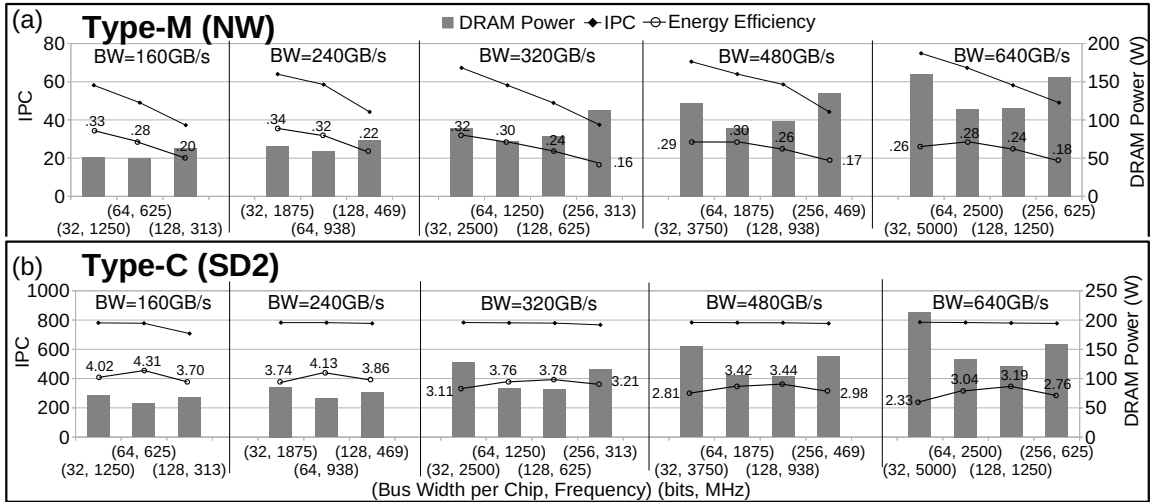
Figure 5: Performance, power, and energy efficiency of two types of applications, which are (a) memory-intensive (Type-M) and (b) non-memory-intensive (or compute-intensive) (Type-C).

nents shown in Figure 2(b). The bus transmission lines are routed on the silicon interposer, which has sufficient space for the buses. To estimate the performance overhead, we evaluated the total latency of memory access pattern change detection and reconfiguration from 100 to 1000 cycles, i.e., 300ns to 3μs at 325MHz GPU core clock frequency in our baseline. We measured the execution time of various applications, and the maximum and minimum values are 6,000,000 cycles (18ms) and 40,000 cycles (120μs), respectively. Therefore, the maximum performance overhead is estimated as 3μs / 120μs, which is less than 2.5% of total execution time.

4.3 EOpt: Optimizing Energy Efficiency

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. Sophisticated prediction schemes could be incorporated into the reconfiguration mechanism, although most GPU applications do not frequently change their memory access patterns. Therefore, we use a simple 1-bit prediction scheme that yields sufficient accuracy without much performance overhead.

4.4 PerfOpt: Optimizing System Performance Under a 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. Therefore, we explore GPU system performance optimization under a given power budget. Our performance optimization target is the instruction throughput (the executed instructions per second).

Figure 6 shows the flow chart of our reconfiguration mechanism. Our reconfiguration mechanism addresses the type-M and -C execution periods of a workload with different strategies. During type-C periods, we always employ the memory interface configuration that minimizes the DRAM power consumption. Any saved power is transferred to scale up the GPU core clock frequency/supply voltage to improve system performance. During type-M periods, we consider two possible 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.

Note that short-duration violations can be tolerated if there is sufficient thermal headroom. Existing CPU systems (e.g., Intel's Turbo Boost technology) already "overclock" beyond their nominal power limits for short itervals. If violations last too long, on-chip thermal monitors can always trigger more aggressive VF scaling to prevent catastrophic damage to the chip [9]. Another possible issue with this reconfiguration mechanism is that the power budget may be violated within an instruction interval. In fact, it is a trade-off between the strictness of following the power budget constraint and the amount performance overhead. We evaluated various lengths of instruction intervals from 1000 to 1 billion, and find that 1 million instruction intervals can provide sufficient guarantee of power budget constraint with below 5% of reconfiguration overhead.

5. EXPERIMENTAL SETUP

Simulations are performed on GPGPU-Sim [2], a cycle accurate PTX-ISA simulator. The shader cores, caches, and interconnection network of the baseline GPU are configured based on NVIDIA Quadro® FX5800 [15]. We modify the simulator to implement our reconfiguration mechanisms. The instruction interval N is set to be 1 million. The baseline graphics memory is off-chip GDDR5 memory, reported as can support up to 3.5GHz frequency [8]. We
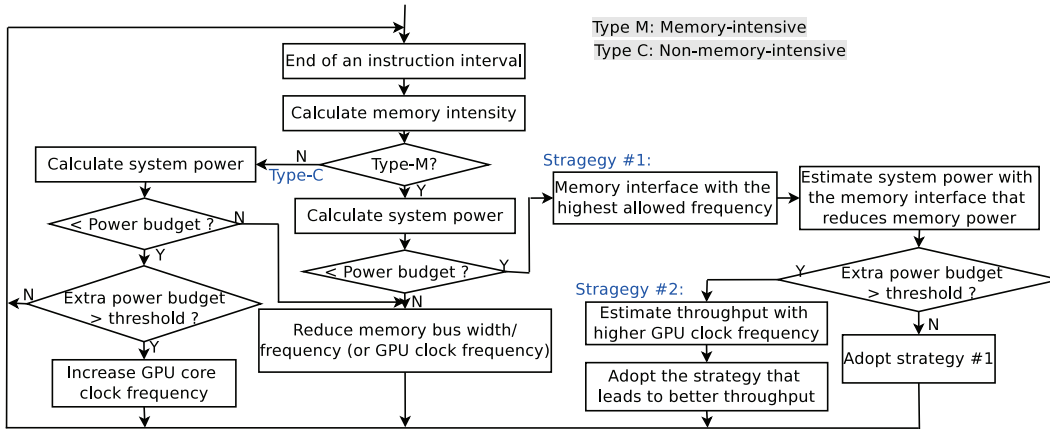
Figure 6: Flow chart for PerfOp.

Table 1: DRAM configurations.

Memory Clock	t_{RAS}	t_{CL}	t_{RP}	t_{RC}	t_{RCD}	t_{RRD}
Baseline: Off-chip GDDR5, 2GB, Bandwidth = 320GB/s						
2.5GHz	22ns	8ns	8ns	30ns	8ns	5ns
In-package Graphics Memory, 2GB,						
Bandwidth = 160GB/s, 240GB/s, 320GB/s, 480GB/s, 640GB/s						
3.75GHz	16ns	5ns	5ns	20ns	5ns	4ns
2.5GHz	18ns	7ns	7ns	24ns	7ns	4ns
1.875GHz	21ns	8ns	8ns	29ns	8ns	4ns
1.25GHz	23ns	10ns	10ns	32ns	10ns	5ns
938MHz	27ns	12ns	12ns	36ns	12ns	6ns
625MHz	31ns	14ns	14ns	38ns	14ns	7ns
469MHz	37ns	16ns	16ns	40ns	16ns	11ns
313MHz	47ns	20ns	20ns	50ns	20ns	14ns

scale the PBW to 320GB/s by increasing the memory frequency from 1.25GHz to 2.5GHz. We use curve fitting based on GDDR5 to obtain the low-level DRAM timing as shown in Table 1. The latency of signals passing through silicon interposer can be reduced to 1/5 of that with standard I/Os [5]. We conservatively assume 20% memory latency improvement compared to off-chip memories.

We evaluate various available GPU workloads from the NVIDIA CUDA SDK [12], Rodinia benchmarks [3], and applications distributed with GPGPU-sim [2]. Table 2 lists the characteristics of our 16 workloads. The memory intensity (MI) of some applications, such as *aes* and *sto*, is lower than 1.0. The three most memory-intensive benchmarks are *mum*, *bfs*, and *nw*.

We calculate the DRAM power based on the power model from Micron [10] and modify it to calculate the I/O power of GDDR memory with the pseudo-open drain (POD) signaling scheme. We calculate the power of GPU cores, caches and memory controllers based on the power model from McPAT [11], and modify the model to adapt to the configuration of GPU shader cores. The performance and memory-access information is fed into the power model to calculate the run-time system power consumption.

6. RESULTS

Static Interface: Figures 7 shows the performance and power of the in-package graphics memories with fixed bus widths of 64-bit (other bus width configurations also show the same trend in energy efficiency improvements). Our results of the integrated graphics memories are normalized to the baseline of off-chip GDDR5 memory that supports a peak bandwidth of 320GB/s. Some applications show IPC losses compared to the baseline. This is due to the fact that for a given fixed bandwidth, the larger buses provided by the in-package graphics memories are offset by lowering their clock speeds. At a system bandwidth allocation of 160GB/s, memory-intensive applications (e.g., *mum*, *bfs*, and *nw*), do incur some significant IPC
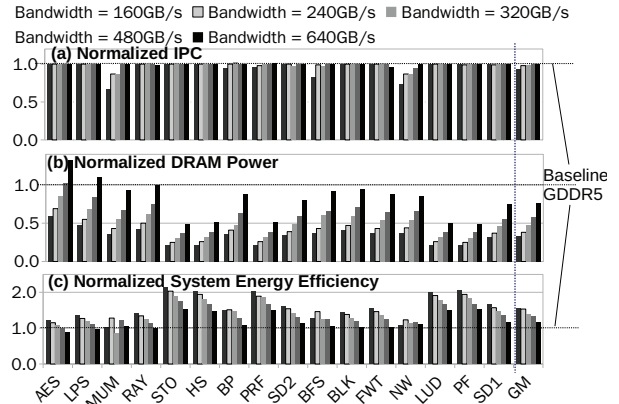

Figure 7: (a) IPC, (b) DRAM power, and (c) energy efficiency with in-package graphics memory (no reconfiguration).

degradations when using the in-package graphics memory. Not surprisingly, the non-memory-intensive applications (e.g., *aes* and *sto*), do not suffer from the reduction in memory clock speed. The in-package memory, however, provides a much more power-efficient implementation (middle plots), which in turn leads to better energy efficiency. Of course, fixing system bandwidth to be equal to the off-chip solution does not really take advantage of the wide interface provided by the in-package memory. By increasing the memory interface clock speed to provide bandwidths of 480GB/s and 640GB/s, performance on the memory-intensive applications can be brought back up. Even without reconfiguration, the in-package graphics memory solution significantly improves the overall energy efficiency of the GPU system by up to 54% on average.

EOpt: Figure 8 shows performance and power results of reconfigurable memory interface optimized for energy efficiency. Because we do not have any initial information about an application, the initial memory interface configuration is always set to a 128-bit bus width. All results are normalized to the case of in-package graphics memory using static 128-bit interfaces to demonstrate the additional benefit of dynamic reconfiguration on top of the benefits of integrated memories. As shown in Figure 8(a), the reconfiguration mechanism yields the greatest IPC improvement for the three most memory-intensive applications. The IPCs of non-memory-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 IPC remains stable on the non-memory-intensive applications. Figure 8(b) shows that DRAM power with almost all applications is reduced, and by an average of 14%. Figure 8(c) il-

Table 2: Characteristics of selected GPGPU benchmarks (the instruction count (IC) and the memory intensity (MI)).

| Abbrev.	Benchmarks	IC	MI	Abbrev.	Benchmarks	IC	Memory
HS	Hot Spot	80M	1.5	BP	Back Propagation	193M	4.3
PRF	Particle Filter	3.9G	4.4	BFS	Breadth First Search	484M	37.0
NW	Needleman Wunsch	218M	8.0	LUD	LU Decomposition	40M	2.5
PF	Path Finder	76M	1.9	FWT	Fast Walsh Transform	4.5G	5.2
AES	AES Encryption	30M	0.3	BLK	BlackScholes Option Pricing	196M	3.2
LPS	3D Laplace Solver	82M	3.8	MUM	MUMmerGPU	75M	43.7
RAY	Ray Tracing	65M	3.0	STO	StoreGPU	123M	0.6
SD1	Speckle Reducing Anisotropic Diffusion	8.4G	5.8	SD2	Speckle Reducing Anisotropic Diffusion	2.4G	3.8
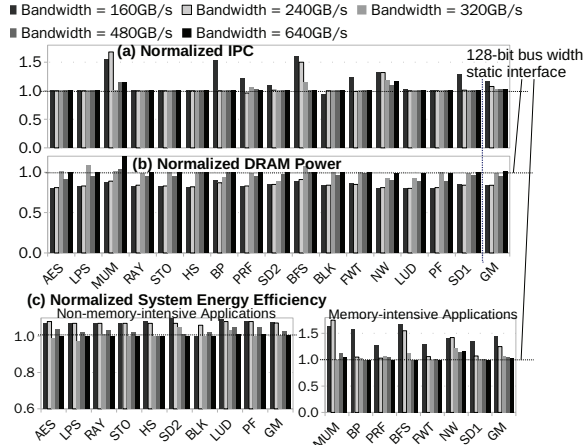
Figure 8: Results of using EOpt to all benchmarks. (a) IPC. (b) DRAM power. (c) System energy efficiency improvement.
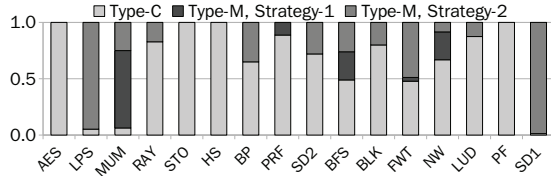

Figure 9: The fraction of instructions spent on different configuration modes, using PerfOpt.

lustrates that the overall system energy efficiency is improved for all the benchmarks. Across all low- and high-intensity applications, the energy efficiency improves by 23% on average.

PerfOpt: Figure 9 illustrates the fraction of instructions spent on each configuration mode under the power budget of 220W (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, *mum*, spends about 68% 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. In the rest 7% of instructions, the memory power is minimized as type-C periods to further improve system performance. It is also shown that the configuration modes change from one to another for most applications. The exceptions are those non-memory-intensive applications, such as *aes*, *sto*, *hs*, and *pf*, which stay in type-C configuration all the time. Table 3 shows the results of the reconfiguration for improving overall GPU throughput given various system power budgets. All the performance results are normalized to the mean throughput of the static configuration that leads to the highest average performance with all the applications under the given power budget. The results demonstrate that the reconfiguration mechanism can adjust the memory power consumption to fit the application memory needs, and that

Table 3: Mean Throughput improvement of all tested benchmarks for each given GPU system power budget.

Budget(W)	200	210	220	230	240
Imprv.	10.0%	14.8%	21.8%	27.5%	30.3%

the saved power can be effectively redeployed to improve the GPU core performance.

7. CONCLUSION

We have presented a reconfigurable in-package wide interface graphics memory, integrated with a high-performance GPU on a silicon interposer. Our design is feasible and easy to implement. Almost all the hardware modification is limited to the memory interface and controllers and no modifications are required to the internal structures of the processors and the memory arrays. Reconfiguration is only applied to the memory interfaces, and internal bus widths are fixed. Therefore, the main extra manufacturing costs is the packaging cost. Another merit of our design is the flexibility of optimizing either power or performance. For high performance computing systems, performance is the primary design consideration. Yet, power constrained applications prefer high energy efficiency. Our reconfiguration mechanisms can be employed by both types of systems.

8. REFERENCES

[1] AMD. Radeon specifications. http://www.amd.com/us/products/desktop/graphics/.

[2] A. Bakhoda, G. Yuan, and W. Fung et al. Analyzing cuda workloads using a detailed gpu simulator. In *Proc. of Intl. Symp. on Performance Analysis of Systems and Software*, pages 163–174, 2009.

[3] S. Che, M. Boyer, and J. Meng et al. Rodinia: a benchmark suite for heterogeneous computing. In *Proc. of Intl. Symp. on Workload Characterization*, pages 44–54, 2009.

[4] X. Dong, Y. Xie, and N. Muralimanohar et al. Simple but effective heterogeneous main memory with on-chip memory controller support. In *Proc. of the International Conference for High Performance Computing*, pages 1–11, 2010.

[5] P. Dorsey. Xilinx stacked silicon interconnect technology delivers breakthrough FPGA capacity, bandwidth, and power efficiency. In *Xilinx White Papers*, 2010.

[6] Elpida. Introduction to GDDR5 SGRAM. http://www.elpida.com/pdfs/e1600e10.pdf. 2010.

[7] M. Gebhart, D. R. Johnson, and D. Tarjan et al. Energy-efficient mechanisms for managing thread context in throughput processors. In *Proc. of the Intl. Symp. on Computer architecture*, pages 235–246, 2011.

[8] Hynix. GDDR5 SGRAM datasheet. http://http://www.hynix.com/products/graphics/.

[9] Intel. Thermal protection and monitoring features: a software perspective. In *Intel Software Network*, pages 1–6.

[10] J. Janzen. The Micron power calculator, http://www.micron.com/products/dram/syscalc.html.

[11] S. Li, J. H. Ahn, and R. D. Strong et al. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proc. of the Intl. Symp. on Microarchitecture*, 2009.

[12] NVIDIA. CUDA SDK. http://www.nvidia.com/object/cudasdks.html.

[13] D. Q. Ren and R. Suda. Modeling and optimizing the power performance of large matrices multiplication on multi-core and GPU platform with CUDA. In *Proc. of the International Conf. on Parallel Processing and Applied Mathematics*, pages 421–428, 2010.

[14] K. Skadron, M. R. Stan, and K. Sankaranarayanan et al. Temperature-aware microarchitecture: modeling and implementation. *ACM Trans. on Architecture and Code Optimization*, 1(1):94–125, 2004.

[15] N. specifications. http://www.nvidia.com/object/productquadrofx5800us.html.

[16] M. Sunohara, T. Tokunaga, T. Kurihara, and M. Higashi. Silicon interposer with TSVs (through silicon vias) and fine multilayer wiring. In *Proc. of the Electronic Components and Technology Conference*, pages 847–852, 2008.

[17] W.-K. S. Yu, R. Huang, and S. Q. Xu et al. SRAM-DRAM hybrid memory with applications to efficient register files in fine-grained multi-threading. In *Proc. of the Intl. Symp. on Computer Architecture*, pages 247–258, 2011.