Optimizing Bandwidth and Power of Graphics Memory with Hybrid Memory Technologies and Adaptive Data Migration

Jishen Zhao[1] and Yuan Xie[1,2]

[1]Department of Computer Science and Engineering, The Pennsylvania State University

[2]Advanced Micro Devices, Inc.

juz138@cse.psu.edu

ABSTRACT

While GPUs are designed to hide memory latency with massive multithreading, the tremendous demands for memory bandwidth and power consumption constrain the system performance scaling. In this paper, we propose a hybrid graphics memory architecture with different memory technologies (DRAM, STT-RAM, and RRAM), to improve the memory bandwidth and reduce the power consumption. In additionation, we present an adaptive data migration mechanism that exploits various memory access patterns of GPGPU applications for further memory power reduction. We evaluate our design with a set of multithreaded GPU workloads. Compared to traditional GDDR5 memory, our design leads to 16% of GPU system power reduction, and improves the system throughput and energy efficiency by 12% and 33%.*

1. INTRODUCTION

The increasing computational power of modern graphics processing units (GPU) makes it a commonly used solution for high-performance computing by employing hundreds of processing units and thousands of in-flight threads [12]. Although memory bandwidth has been increasing in recent years, the ratio of bandwidth to computing capability is decreasing, and thus it will become a bottleneck to scalable performance for GPU systems. Furthermore, the power consumption of GPU systems continues to increase to accommodate the high throughput demands. Consequently, maintaining high memory bandwidth and system energy efficiency is a critical challenge for future GPU systems.

Conventional graphics DRAM (GDDR) adopted various techniques to reduce memory power consumption, which may incur bandwidth penalty. For example, Elpida's GDDR5 memories are specified to operate over a large contiguous voltage and frequency (VF) ranges to support data rates starting from as low as 800MB/s per channel to a maximum rate of 20GB/s [5]. While 1.6GB/s per channel is sufficient for displaying static images, a data rate of 6GB/s is required for playing high-definition (HD) video, and the maximum data rate may be fully utilized by high-end gaming applications. The existing power saving methods cannot satisfy the increasing bandwidth demand of future high-performance computing and advanced video processing applications.

The emergence of various non-volatile memory (NVM) technologies, such as spin-transfer torque memory (STT-RAM), phase-change memory (PCM), and resistive memory (RRAM), provides promising memory system solutions with non-volatility and low power consumption. Our study of various GPU workloads shows that only a portion of data (less than 50% for 10 out of 20 studied application) is frequently accessed during run-time. Therefore, the data that is infrequently accessed and of low write intensity can be stored in NVM and managed in standby mode with near-zero power consumption. In this paper, we propose a hybrid graphics memory system, mixing DRAM, STT-RAM, and RRAM, that can provide higher memory bandwidth and consumes less power than the traditional GDDR5 memory. Replacing part of the DRAM with a NVM partition, the hybrid graphics memory can run at a higher frequency and thus provide higher peak memory bandwidth. By migrating the read-only and infrequently-accessed data in the NVM partition, the hybrid memory system also consumes less power than conventional GDDR5 memory. Although NVM has a longer write latency than DRAM, our study indicates that the memory access patterns of GPU workloads can naturally hide such latency. In order to save the memory power without much performance degradation, we propose an adaptive data migration mechanism, leveraging different memory access patterns of different GPU workloads.

2. BACKGROUND AND RELATED WORK

Graphics memory and power saving techniques 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

*This work is supported in part by SRC grants, and NSF 0903432, 1218867, and 1213052.

Permission to make digital or hard copies of all or part 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 bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2012, November 5–8, 2012, San Jose, CA, USA.
Copyright 2012 ACM 978-1-4503-1573-9/12/11 ...$15.00.

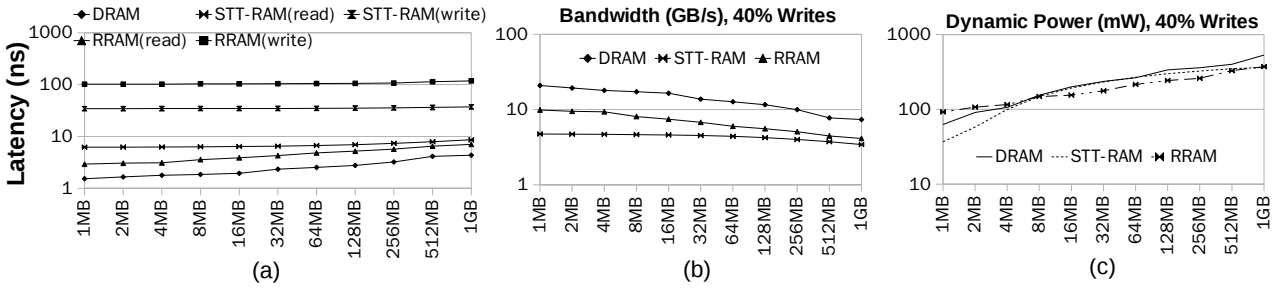

Figure 1: (a) Latency, (b) provided bandwidth (PBW), and (c) dynamic power of different memory technologies with respect to different capacities.

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 [15]. The peak power consumption of GDDR3 is over 100W with less than 128GB/s bandwidth [15]. GDDR5, the latest generation of graphics DRAM [15], employs VF scaling to reduce power consumption. Unfortunately, the memory bandwidth is degraded as well, leading to the performance loss of GPU systems. In our work, we exploit hybrid graphics memory designs targeting at system energy efficiency. Both memory bandwidth and power consumption are improved with our hybrid graphics memory design.

Emerging memory technologies STT-RAM is the latest generation of magnetic RAM (MRAM) [17]. STT-RAM employs Magnetic Tunnel Junction (MTJ), which contains two ferromagnetic layers and one tunnel barrier layer, as its binary storage. RRAM is another emerging non-volatile memory [17]. In RRAM, a normally insulating dielectric is conducted through a filament or conduction path generated by applying a sufficiently high voltage. The filament may be reset or set by applying an appropriate voltage. PCM [14] uses chalcogenide-based material to storage information. The chalcogenide-based material can be switched between a crystalline phase (SET or "1" state) and an amorphous phase (RESET or "0" state) with the application of heat. Compared to DRAM, NVMs of STT-RAM, RRAM, and PCM [14] show significant power benefits. Due to the non-volatility, NVMs do not require refresh operations, and have near-zero standby power. The drawbacks of NVMs are their long write latency, high write energy, and endurance problem. The long write latency will reduce the memory bandwidth, a critical resource of a GPU. In this work, we argue that only a small portion of data is frequently accessed at each execution period with GPU applications. By maintaining this portion of data in DRAM, the overall provided bandwidth will not be reduced. Furthermore, we improve the memory bandwidth by reducing the DRAM capacity and scale up the memory clock frequency. Compared to STT-RAM and RRAM, PCM incur significant latency overhead. Therefore, we only consider STT-RAM and RRAM in our hybrid memory design. The endurance of STT-RAM is over 10^{15} [16]. The endurance of RRAM is in the range of 10^5 to 10^{10} [7]. Both are feasible to be used as main memory. While the endurance of our hybrid memory is not discussed in detail, we limit the frequency of data migration to extend the lifetime.

Hybrid memory Most existing work on hybrid memory focuses on main memory of CPUs and only studies general-purpose applications [3, 4, 10]. We evaluated various GPU applications, and observe that the specific characteristics of memory access patterns can be utilized to fully leverage the power benefits of NVMs with low performance overhead. Yu *et al.* proposed hybrid register file design in GPUs [19]. They employ SRAM and DRAM memory cells to reduce the power consumption of register file. Compared to their work, our work explores graphics memory design and uses NVMs for memory power reduction. *To the best of our knowledge, this is the first work to study hybrid memory architecture that leverages emerging NVMs for GPU systems.*

3. MOTIVATION

We evaluate the provided bandwidth (PBW) and power consumption of DRAM, STT-RAM, and RRAM. NVMs appear to have significant power benefits, but lower PBW than DRAM. Our hybrid memory hardware configuration is built based on these bandwidth and power characteristics. We also study the memory access patterns of various GPU applications, based on which we develop our energy-efficient adaptive data migration mechanism.

3.1 Characteristics of Memory Technologies

We use NVSim [18], a circuit level memory model, to evaluate the performance, bandwidth, and power of the three memory technologies of DRAM, STT-RAM, and RRAM. Figure 1 and 2 illustrate the results. At each memory capacity, the memory bank and mat organizations are optimized for read latency to minimize the dynamic power consumption. We only show bandwidth and dynamic power at 40% write intensity, the maximum write intensity observed from the GPU applications we studied.

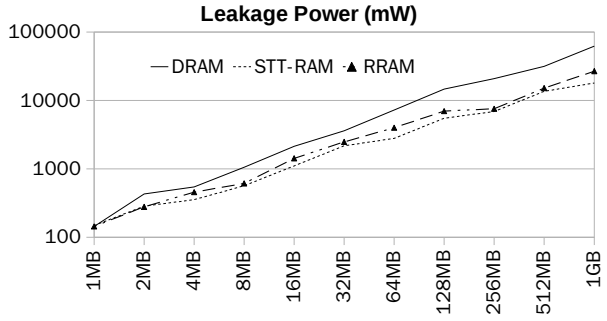

Figure 2: Leakage power of different memory technologies with respect to various capacities.

Latency and bandwidth: Figure 1(a) shows that DRAM has the

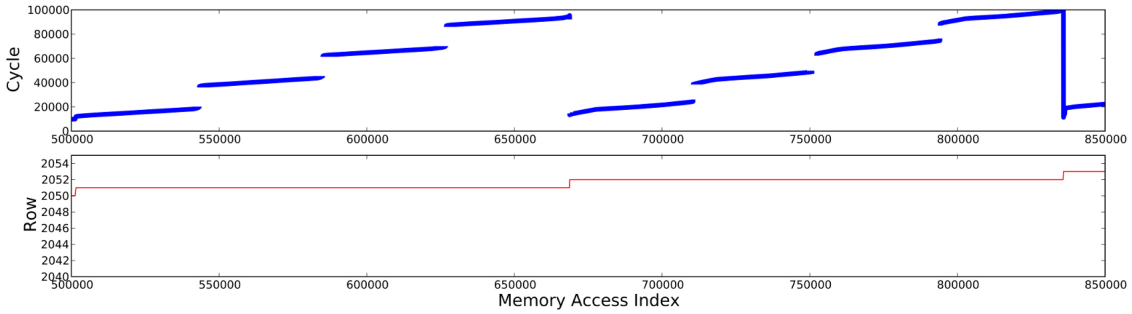

Figure 3: The pattern of "interleaved access".

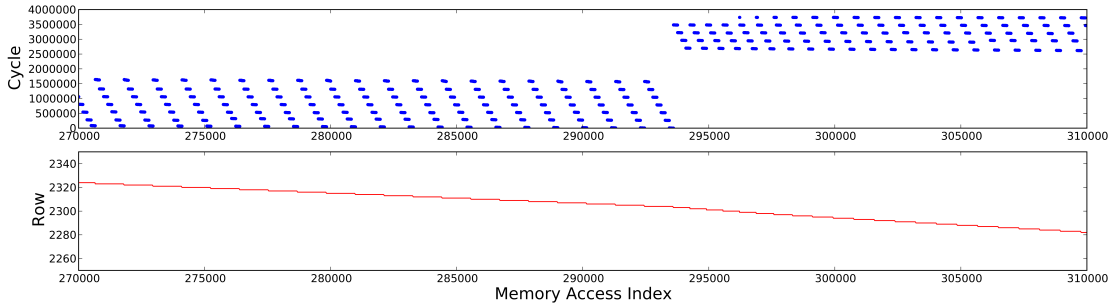

Figure 4: The pattern of "access then idle". Can also observe the "burst" pattern during the access period.

lowest read and write latencies among the three memory technologies. The read latency of RRAM is lower than STT-RAM, and is comparable with DRAM at large capacities. For example, RRAM only incurs 2ns additional read latency than DRAM at 128MB. Although with higher read latency, STT-RAM has lower write latency than RRAM across all capacities. Figure 1(b) shows the PBW curves. Due to long write latency, the average PBW of RRAM and STT-RAM is only 50% and 30% that of DRAM.

Power: Due to the low access current, the dynamic power of RRAM is lower than DRAM. As shown in Figure 1(c), RRAM consumes 17% dynamic power less than that of DRAM at the capacities larger than 128MB. Figure 2 illustrates the leakage power of different memory technologies. On average, the leakage power of STT-RAM and RRAM is only 37% and 48% that of DRAM. In addition, due to the non-volatile nature, we can obtain near-zero standby power by power gating the idle portions STT-RAM and RRAM. Overall, we can not find a single winner from the perspective of both performance and power. DRAM has the best PBW among the three memory technologies. However, the two NVMs have significant power benefits.

Based on such observation, we adopt hybrid memory design to combine the benefits of high PBW of DRAM and low power consumption of NVMs.

3.2 Memory Access Patterns of GPU Workloads

We examined various GPU workloads on the baseline GPU system with GDDR5 graphics memory (Section 5.1), and observe three memory access patterns, namely "interleaved access", "access then idle", and "burst".

Interleaved access: Figure 3 demonstrates the pattern of "interleaved access". Here we sort the memory accesses based on ascendant order of DRAM row address. The x-axis is the index of memory accesses. The figure on the first row represents the cycle of each memory access. The figure on the second row represents the row that is accessed by each memory request. For example, the 550000th memory access is from the *row 2051* at *cycle 40000*. Figure 3 shows that row 2051 is accessed during the entire application execution. However, we can observe three idle periods that are twice as long as the time when the row being accessed. Although not shown in the figure, we also observe that a significant $\frac{5}{6}$ of memory access is read-only. This portion of data may be corresponding to constant or texture data in GPU applications. Since RRAM only incur small read latency penalty, we can maintain this portion of data into RRAM. We can reduce the memory power by powering off the memory space during the idle periods.

Access then idle: Figure 4 illustrates the pattern of "access then idle". We can observe that the memory rows between 2300 and 2320 are only accessed during the initial time to the cycle 1,800,000. Afterward, this portion of memory becomes idle, and never accessed again. Potentially, we can turn off this portion of memory to save power. This is impossible with pure DRAM based memory. With NVMs, however, it is feasible to standby some portions of memory space.

Burst: The "burst" pattern represents frequently accesses during the entire execution time. It can be observed during the access period of both "interleaved access" and "access then idle" patterns. For example, in Figure 4 row 2300 is in "burst" state before cycle 2,400,000 is achieved. This portion of data needs to be maintained in DRAM to minimize performance degradation.

4. HYBRID MEMORY ARCHITECTURE

Figure 5 depicts an overview of our hybrid graphics memory design. We replace half of the DRAM capacity with RRAM and STT-

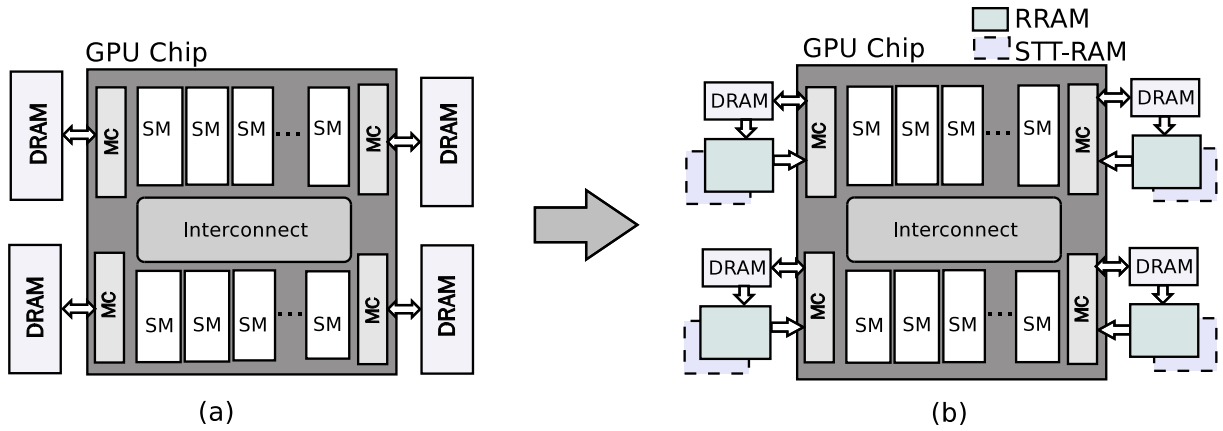

Figure 5: Overview of GPU system with hybrid memory. (a) Conventional GPU system with off-chip GDDRs. (b) GPU system with hybrid memory.

RAM. With half the capacity, DRAM can provide up to 25% higher memory bandwidth by scaling up the clock frequency. We migrate the read-only and infrequently accessed data to NVMs. Due to the non-volatility, we can significantly reduce the memory power consumption by powering off the idle NVM space. The hardware modification is limited to the memory interface and controllers and no modification is required to the internal structures of the GPU processors and the memory arrays.

4.1 Hardware Configuration

Partitions of different memory technologies: Our hybrid graphics memory consists of DRAM, RRAM, and STT-RAM partitions. Frequently accessed data is maintained in the DRAM that has the highest PBW among the three memory technologies. The two NVM partitions are used to store the data with low access frequency. The idle rows in the NVM partitions are powered off. Between the two NVMs, RRAM is used to store data that is read-only or with extremely low write intensity. Since STT-RAM has better write performance than RRAM, it is used to store the infrequently accessed data with higher write intensity.

Capacity of each partition: Our baseline GPU system (Section 5.1) employs 256MB DRAM per channel. In our hybrid graphics memory, we reduce the DRAM capacity to 128MB per channel and replace the rest 128MB with RRAM. This is based on the observation that the dynamic power of DRAM is higher than RRAM at the capacities of higher than 128MB. In addition to power benefits, reducing the DRAM capacity can also improve PBW. In Figure 1(b), we observe that the PBW of DRAM at 128MB is 1.25× that of 256MB. We adopt 8MB STT-RAM per channel. This is based on the fact that the dynamic power of STT-RAM is the lowest among the three memory technologies at small capacities. The row buffer size of different memory partitions are configured to be 256-bit, so that data migration incurs minimum overhead in address mapping. The hybrid memory may incur a small increased area ($9mm^2$) compared to the baseline (45nm technology). However, the increased memory area is only limited to the off-chip memory and we do not expect high cost increase to the GPU system.

Memory interface: Figure 6 shows our memory interface configuration. As shown in Figure 6, an additional bus of 32 bits is adopted to accommodate data migration and NVM reads (data will be read out directly from the NVMs, once it is migrated to the NVM partitions). GPU can read and write from the DRAM partition. GPU can only read from the NVM partition. In our experiments, we consider the additional I/O termination power incurred by this modification and show the total system power is still reduced with such I/O overhead. Furthermore, multiplexing is required to switch the reads between DRAM and NVMs.

Memory controller: A NVM controller is integrated in the memory controller to accommodate the data migration and manage the power mode of NVMs. Components of the NVM controller include data migration buffers, read buffers, and registers to store the memory idle states. We modify the memory controller to facilitate data migration. Timers are employed to manage the idle state of each DRAM row. Each row also has counters to collect memory access data. We evaluated the area and power overhead of these timers and counters. For 32-bit timers (counters), the total storage size will be 32.3KB (assume 2KB page size (row-buffer size) as same as the baseline GDDR5). This is negligible in a memory system of 256MB. An address mapping table is used to store the new address after a row is migrated from DRAM to NVMs.

4.2 Data Migration Mechanism

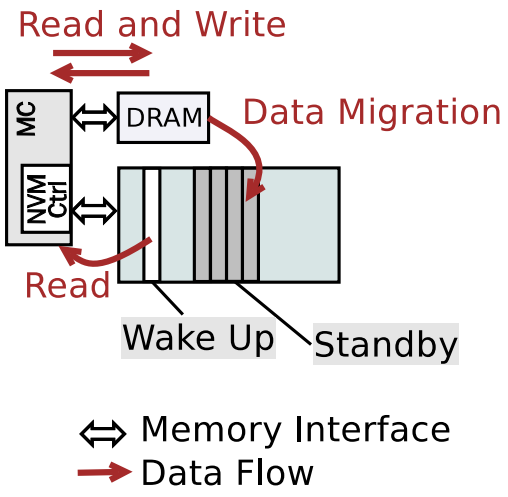

Figure 6: Memory interface configuration and data flow between memory controller and hybrid memory.

Our data migration algorithm is implemented in the memory controllers. The goal of our data migration mechanism is to improve the system energy efficiency, i.e., to reduce the memory power consumption with low performance degradation. Figure 6 illustrates the data flow between memory controller and hybrid memory. The overall idea is to store the idle data in NVMs in powered-off state, and maintain the frequently accessed data in DRAM. Based on our GPU workload characterization, we find the opportunity of data migration in both "access and idle" and "interleaved access" patterns. With "access then idle", it is straightforward to start migration once a row becomes idle. With "interleaved access", however, the start point of data migration needs to be carefully determined. To obtain sufficient energy benefit from data migration, the idle time need to be sufficiently long. Otherwise, the GPU system may suffer significant performance loss due to the low PBW of NVMs and NVM lifetime will be reduced. Our data migration mechanism is illustrated in Figure 7 and 8.

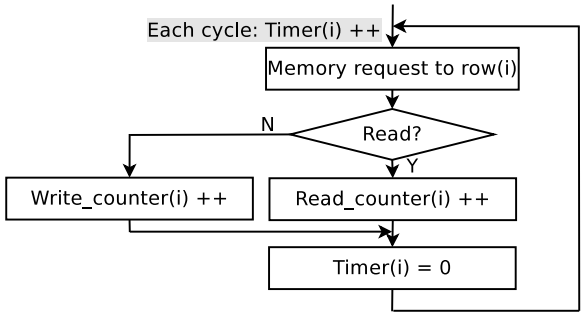

Figure 7: The loop of DRAM access management.

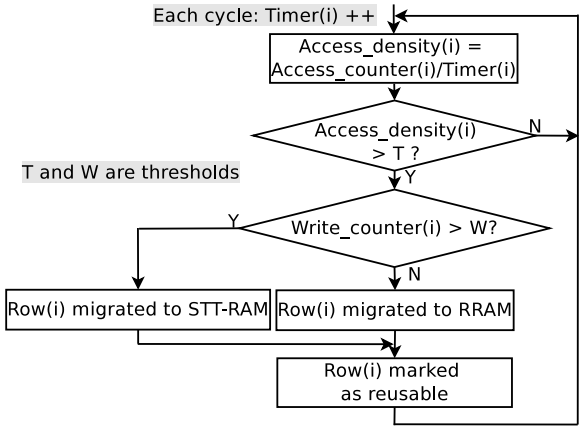

Figure 8: Control of data migration.

Figure 7 is the loop of DRAM access management. At each DRAM access, we update the access counters and the row timer, based on the types of accesses. Figure 8 shows the mechanism to determine the start point of data migration. We do not immediately initialize a data migration operation at a row time-out event. Instead, data migration is determined by the memory access density within a period of execution time. In this way, we avoid the GPU system performance degradation incurred by unnecessary data migration and extend the NVM lifetime. Furthermore, we evaluate the write intensity to determine which NVM space will be used. STT-RAM is used when the write intensity is higher than a predefined threshold to avoid high performance degradation incurred by frequent data migration. Some applications may have extremely low memory intensity. The power consumed by data migration may be higher than the DRAM dynamic power. In this case, we disable data migration when the memory intensity is lower than a threshold. To determine the power modes (standby or wake-up), we maintain a timer for each row in NVMs. Once a time-out event is detected, we power off the row. A read or migration request to the row will wake it up.

5. EVALUATION

In this section, we show experimental results of system performance, power, and energy efficiency with the our hybrid graphics memory design and explain the reasons leading to these results.

5.1 Experimental Setup

Simulation platform: Table 1 specifies the configurations and parameters of the baseline GPU processor, the baseline pure GDDR5 graphics memory, and our hybrid graphics memory. The shader cores, caches, and interconnection network are configured based on NVIDIA's Quadro FX5800 [13]. The DRAM configuration and timing are based on Hynix GDDR5 datasheet [6]. We use GPGPU-sim [1], a cycle accurate PTX-ISA simulator, to run our experiments. We modify the simulator to implement our hybrid graphics memory and data migration mechanism.

Benchmark: We evaluate twenty GPU applications from the NVIDIA CUDA SDK [11], Rodinia Benchmarks [2], and applications distributed with GPGPU-sim [1]. Table 2 lists the characteristics of these applications. The applications have various average demanded bandwidth (DBW).

Power model: Our power model considers various GPU system components, including GPU cores and caches, memory controllers, and graphics memories. We calculate the power of GPU cores, caches and memory controllers based on the power model from McPAT [8]. We modify the power model to adapt to the configuration of GPU shader cores. The memory power, DRAM, STT-RAM, and RRAM, is calculated with NVSim [18]. We also evaluate the I/O power of graphics memory with a modified DRAM power model from Micron [9]. Micron's power model is based on DDR3 standard that employs the stub series terminated logic (SSTL) signaling scheme. GDDR5 employs the pseudo open drain (POD) signaling scheme. The POD drivers uses a 40Ω or 60Ω impedance that drives into a 60Ω equivalent on-die-terminator (ODT) tied to Vddq. Our DRAM power model adopts the POD signaling scheme.

5.2 Results

Figure 9 to 12 show the power, performance, and energy efficiency with our hybrid graphics memory design.

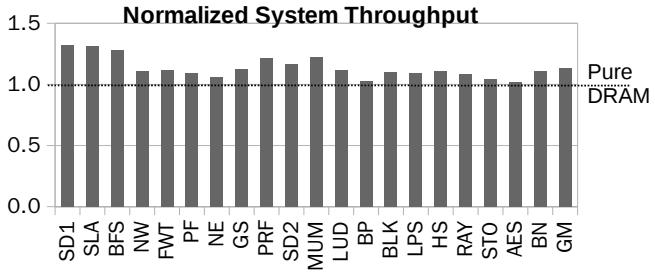

Figure 9: System throughput with hybrid graphics memory, normalized to baseline.

Table 1: Baseline GPU and hybrid graphics memory configuration.

Shader Cores	30	Core Clock Frequency	325 MHz
SIMD Pipeline Width	32	Warp Size	32
Threads per Core	1024	CTAs per Core	8
Registers per Core	16384	Shared Memory per Core	16KB
DRAM Request Queue	32	Memory Controller	FR-FCFS
Caches per Core	64B Lines; Constant Cache: 8KB; Texture Cache: 8KB; L1 Cache: 16KB		
Baseline Graphics Memory	GDDR5, 1GHz, 256MB per Channel		
Hybrid Graphics Memory	Per channel: 128MB GDDR5, 1.25GHz; 128MB RRAM; 8MB STT-RAM (32-bit timer and counter per memory row)		

Table 2: Characteristics of 20 GPU benchmarks. DBW is the demanded memory bandwidth in the unit of GB/s.

Abbrev.	Benchmarks	IC	DBW	Abbrev.	Benchmarks	IC	DBW
SD1	Speckle Reducing Anisotropic Diffusion	8.4G	206.9	LUD	LU Decomposition	40M	1.6
SLA	Scan of Large Arrays	15.8G	178.0	MUM	MUMmerGPU	75M	1.5
NW	Needleman Wunsch	218M	15.4	BP	Back Propagation	193M	1.0
BFS	Breadth First Search	484M	13.4	BLK	BlackScholes Option Pricing	196M	0.8
FWT	Fast Walsh Transform	4.5G	12.5	RAY	Ray Tracing	65M	0.7
GS	Gaussian Elimination	8.8M	5.1	LPS	3D Laplace Solver	82M	0.7
PF	Path Finder	76M	4.6	HS	Hot Spot	80M	0.4
NE	Nearest Neighbor	6.1M	4.0	STO	StoreGPU	123M	0.2
PRF	Particle Filter	3.9G	3.2	AES	AES Encryption	30M	0.1
SD2	Speckle Reducing Anisotropic Diffusion	2.4G	2.5	BN	Binomial Options	18G	0.1

Throughput improvement: We have shown that with half the capacity of the baseline DRAM, we can obtain up to 25% PBW improvement by scaling up the DRAM clock frequency (Section 3). We evaluate the system throughput, which is the number of executed instructions per second, with our hybrid GPU memory architecture. As shown in Figure 9, the most significant throughput improvement is obtained by applications with high DBWs, such as *SD1*, *SLA*, and *BFS*. Although the average DBW of *PRF* and *MUM* is lower than 10 GB/s, the two applications suffer very heavy memory accesses with high DBW during a period of execution time. Therefore, increasing the PBW can also improve the throughput with these two applications. The mean throughput improvement with all the applications is 12%.

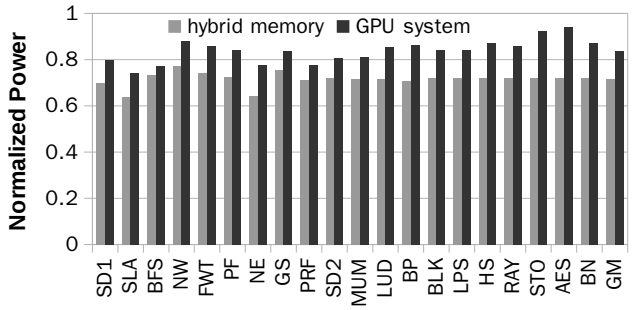

Figure 10: Memory and system power consumption with hybrid graphics memory, normalized to the baseline pure DRAM based graphics memory.

Power saving: Increasing the DRAM clock incurs power overhead, since the supply voltage is scaled up as well. However, the reduced DRAM capacity results in significant leakage power reduction. The NVM partitions also reduces the memory power consumption with low dynamic power and near-zero standby power. We observe that both memory and system power consumption is reduced with all evaluated applications. Figure 10 shows the power consumption of hybrid graphics memory and GPU system, normalized to the baseline with pure DRAM based graphics memory. Our proposed design is more effective on savings system power with applications that have higher bandwidth demands, such as *SD1*, *SLA*, and *BFS*, because memory consumes a large portion of total system power for these applications. The mean savings of memory and system power consumptions are 31% and 16%, respectively.

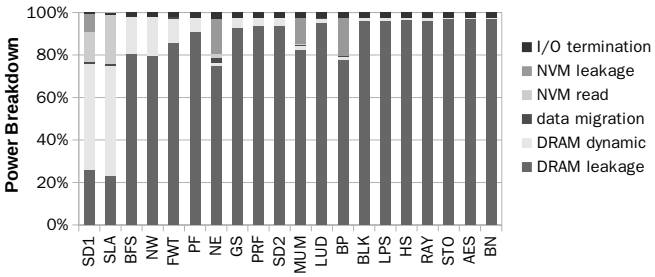

Figure 11: Power breakdown of hybrid graphics memory.

Power breakdown: We also studied the power breakdown of our hybrid graphics memory design. The results are demonstrated in Figure 11. Here we do not show DRAM refresh power, since the maximum of two refreshes are observed with various applications and incur negligible (less than 1% of total memory power) power consumption. With applications that have high DBWs, such as *SD1*

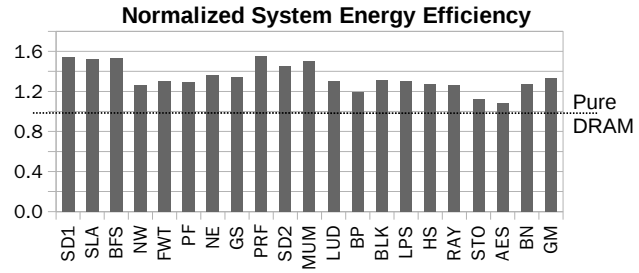

Figure 12: System energy efficiency with hybrid graphics memory, normalized to baseline.

and *SLA*, the dynamic power covers a significant portion of total memory power consumption due to the high memory access intensity of these workloads. In contrast, DRAM leakage power dominates the total memory power consumption of applications with low DBWs, such as *STO*, *AES*, and *BN*. In this case, we cannot afford the dynamic power consumption of data migration and it is therefore disabled with these three applications to avoid the power overhead incurred by data migration.

Energy efficiency: Figure 12 illustrates the results of system energy efficiency, defined as the executed instructions per second per Watt. The system energy efficiency is improved with all the evaluated applications. The most significant improvement can be observed with the applications having high DBWs. The mean improvement of system energy efficiency is 33%.

6. CONCLUSION AND FUTURE WORK

Memory bandwidth and power are the two critical resources of GPU systems and significantly constrain the performance scaling. In this work, we propose a hybrid graphics memory that improves both memory bandwidth and system energy efficiency. The key insight in our work is that hybrid graphics memory design is especially suitable for GPU applications. The memory access patterns of these applications are naturally used to hide the latency issue of NVMs. Our initial results are very promising for future GPU systems, improving 33% in system energy efficiency. Our migration mechanism limits the frequency of write operations, and therefore we do not expect significantly degradation of the lifetime.

Several directions can be explored in our future work. First of all, this work only studies off-chip graphics memory. It may be interesting to explore hybrid memory hierarchy design of the entire memory system with both on-chip caches, shared memories, and off-chip memories. Another interesting research direction is to improve the memory lifetime. In the hybrid graphics memory, the STT-RAM can be employed as the replacement of portions of RRAM with error bits. How to balance between performance and power of such design needs to be studied.

7. REFERENCES

[1] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt. Analyzing CUDA workloads using a detailed GPU simulator. In *Proc. of Intl. Symp. on Performance Analysis of Systems and Software*, pages 163–174, 2009.

[2] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron. Rodinia: a benchmark suite for heterogeneous computing. In *Proc. of Intl. Symp. on Workload Characterization*, pages 44–54, 2009.

[3] G. Dhiman, R. Ayoub, and T. Rosing. PDRAM: a hybrid PRAM and DRAM main memory system. In *Proceedings of the Design Automation Conference*, pages 664–469, 2009.

[4] X. Dong, Y. Xie, N. Muralimanohar, and N. P. Jouppi. Hybrid checkpointing using emerging nonvolatile memories for future exascale systems. *ACM Trans. on Architecture and Code Optimization*, 8:1–29, 2011.

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

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

[7] K.-H. Kim, S. Hyun Jo, S. Gaba, and W. Lu. Nanoscale resistive memory with intrinsic diode characteristics and long endurance. *Applied Physics Letters*, 96(5):1–3, 2010.

[8] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proc. of the Intl. Symp. on Microarchitecture.*, 2009.

[9] Micron. System-power calculator, http://www.micron.com/products/dram/syscalc.html.

[10] J. C. Mogul, E. Argollo, M. Shah, and P. Faraboschi. Operating system support for NVM+DRAM hybrid main memory. In *Proceedings of the Conference on Hot Topics in Operating Systems*, pages 14–14, 2009.

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

[12] NVIDIA. GeForce GTX 480. http://www.nvidia.com/object/productgeforcegtx480us.html.

[13] NVIDIA. Quadro FX5800 spec. http://www.nvidia.com/object/productquadrofx5800us.html.

[14] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y.-C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S.-H. Chen, H.-L. Lung, and C. H. Lam. Phase-change random access memory: a scalable technology. *IBM Journal of Research and Development*, 52:465–479, 2008.

[15] Samsung. Green memory solution. http://www.samsung.com/global/business/semiconductor.

[16] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie. Hybrid cache architecture with disparate memory technologies. In *Proc. of the International Symposium on Computer Architecture*, pages 34–45, 2009.

[17] Y. Xie. Modeling, architecture, and applications for emerging non-volatile memory technologies. *IEEE Computer Design and Test*, 28:44–51, 2011.

[18] C. Xu, X. Dong, N. P. Jouppi, and Y. Xie. Design implications of memristor-based RRAM cross-point structures. In *Proceedings of the Design, Automation and Test in Europe Conference*, pages 1–6, 2011.

[19] W.-K. S. Yu, R. Huang, S. Q. Xu, S.-E. Wang, E. Kan, and G. E. Suh. SRAM-DRAM hybrid memory with applications to efficient register files in fine-grained multi-threading. In *Proceeding of the International symposium on Computer architecture*, pages 247–258, 2011.