2013 IEEE 21st International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems

A Versatile Performance and Energy Simulation Tool for Composite GPU Global Memory

Bin Wang* Yizheng Jiao* Weikuan Yu* Xipeng Shen† Dong Li‡ Jeffrey S. Vetter‡

Auburn University* College of William & Mary† Oak Ridge National Lab‡

{bwang,yzj0018,wkyu}@auburn.edu xshen@cs.wm.edu {lid1,vetter}@ornl.gov

ABSTRACT

As a cost-effective compute device, Graphic Processing Unit (GPU) has been widely embraced in the field of high performance computing. GPU is characterized by its massive thread-level parallelism and high memory bandwidth. Although GPU has exhibited tremendous potential, recent GPU architecture researches mainly focus on GPU compute units and full system exploration is rare due to the lack of accurate simulators that can reveal hardware organization of both GPU compute units and its memory system. In order to fill this void, we build a GPU simulator called VxGPUSim that can support the simulation with detailed performance, timing and power consumption statistics. Our experimental evaluation demonstrates that VxGPUSim can faithfully reveal the internal execution details of GPU global memory of various memory configurations. It can enable further research on the design of GPU global memory for performance and energy tradeoffs.

1. INTRODUCTION

Advances in computer technologies and the need to leverage computation accelerators have propelled the wide adoption of General Purpose Graphics Processing Unit (GPGPU). Since memory plays a critical role in determining the computing efficiency of many GPU applications, a series of efforts have been undertaken to create software optimization tools for enhancing GPU memory performance [22]. However, there has been little work in exploring the hardware design space for GPU global memory in the research community. One of the main barriers is the lack of an open and versatile simulation tool that can faithfully simulate data accesses of GPU global memory in detail. Existing GPU simulators such as GPGPU-Sim have all focused on the simulation of GPU execution units, but not the memory system. In order to include a subsystem for off-chip global memory, they either adopt static memory models that use a mean value for memory latency, or build non-modular memory modules that are infeasible to be extended.

A desirable simulator that supports full GPU simulation should be capable of modeling several low-level memory management and configuration details including parallelism support in memory system organization, request queue organization, row buffer management policy and even hybridization of multiple memories for GPU memory. Hybrid memory systems have received a lot of attention recently [17, 21, 23, 19, 11]. Exploring composite configurations for GPU global memory by combining DRAM with other non-volatile random access memory may help leverage the best characteristics of multiple types of memory, and alleviate the power and density-scaling limitations faced by the traditional DRAM system.

We build a simulator called VxGPUSim that can accurately model composite GPU global memory and accordingly report the performance and power consumption. It will enable memory system architects and researchers to explore a rich memory design space that includes issues on memory parallelism, row buffer locality, request queue management and memory hybridization. A unified memory address space is introduced to integrate different memory devices into a single GPU global memory, which facilitates studying the applicability and utilization of NVRAM in GPU systems. We have conducted a comprehensive experimental evaluation to demonstrate the functionalities of our simulator and the sensitivity of applications to different memory parameters, such as the number of ranks, the request queue size, the open/close page policy, etc. These experiments can lend useful examples for researchers to create GPU global memory with customizable configurations and design assessment experiments that meet their research needs. This simulator will help researchers conduct more exploratory research in GPU memory design, and assist them in exploiting the use of NVRAM in GPU memory hierarchy.

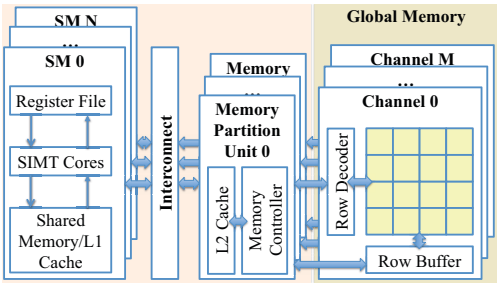

Figure 1: GPU Memory Hierarchy

The rest of this paper is organized as follows. In Section 2, we present the background of GPU memory hierarchy and non-volatile memories. The simulator implementation is described in Section 3. We then present the evaluation in Section 4. Section 5 describes the related work and Section 6 the conclusion.

2. BACKGROUND

GPU: Generally, a discrete GPU consists of several highly multithreaded and pipelined *Stream Multiprocessors* (SM), one interconnection network, several memory partition units, and an off-chip global memory, as shown in Figure 1. One SM is composed of a cluster of *Single Instruction Multiple Threads* (SIMT) cores. SIMT cores execute distinct thread, operate on scalar registers and progress in lockstep. SIMT cores in an SM share the per-SM register file as well as the configurable shared memory and L1 cache. A large off-chip global memory is connected via the on-chip memory channels and cached by the last-level L2 cache if available. The L2 cache interacts with SMs via the interconnection network. For old generations of GPU cards that do not have L2 cache, the memory channels are directly connected with the interconnection network.

1526-7539/13 $26.00 © 2013 IEEE
DOI 10.1109/MASCOTS.2013.39

GPU global memory, as the focal point of interest in this study, can utilize either GDDR3/5 or DDR3 SDRAM. GDDR3/5 is similar to DDR3 in circuit organization, but GDDR3/5 can offer higher peak bandwidth than DDR3 because of its higher data transfer rate per pin and its prefetch buffers.

Non-Volatile Memories: Compared with DRAM, non-volatile memory technologies have several advantages. First, due to the non-volatility, NVRAM cells do not incur leakage, eliminating refreshing logic and power. Second, the peripheral circuits (e.g., row and column decoders, input/output network and sense amplifiers) can be powered down without loosing the contents in memory during idle time [25], which saves background energy. Third, due to non-destructive nature of NVRAM operations, only the dirty lines in a row buffer need to be written back to the NVRAM array [9]. This saves dynamic energy. Lastly, NVRAM have small cell sizes, which makes it scalable to even smaller cell sizes and higher density. However, NVRAM suffers from limited write endurance and higher write latency and energy consumption.

Among various NVRAM technologies, we mainly focus on two representative cases, PCM and STT-RAM. In general, STT-RAM is faster and incurs lower energy consumption; however, PCM is slower and has much higher write latency and energy consumption.

3. IMPLEMENTATION

VxGPUSim is designed by incorporating two existing popular simulators — GPGPU-Sim and DRAMSim2. We use the former to faithfully simulate the compute unit and process pipelining of GPU device and the latter GPU global memory system. We also derive the timing and energy model of NVRAM by selectively adopting the documented DRAM access protocol but applying the NVRAM specific access parameters to it. The NVRAM access parameters such as read, write latency are calculated by using NVMain. More details about the design space our simulator offers and NVRAM models are present as follows for the purpose of providing clear view of VxGPUSim.

3.1 Memory System Design Space

Several design issues, such as memory organization and memory controller, have significant impact on the system performance. A good simulator needs to provide flexibility to explore various memory system designs. To achieve a given amount of GPU global memory, we can use multiple ranks of chips with low density or one rank of dense chips. The key difference is based on the observation that multiple ranks typically work in parallel, resulting in higher parallelism. Though high parallelism is helpful in retaining high throughput, the utilization of multiple ranks might cause higher power consumption partially due to the increased background power of more memory chips. Therefore, there is a trade-off between rank level parallelism and energy efficiency.

Memory scheduler design is another big concern for memory system designer. In this work, we do not explore different scheduling algorithms (the default is FR-FCFS), but mainly investigate how memory controller parameters can impact the effectiveness of different scheduling algorithms. We examine configuration parameters such as memory queue sizes, queuing structure, address translation scheme and row buffer management policy, all of which are the critical design concerns for GPU memory request scheduling.

GPGPU kernels often have to juggle between different performance demands for best system tradeoffs. For example, coalesced global memory accesses tend to have high regularity while uncoalesced accesses exacerbate the speed gap between fast SIMT cores and slow off-chip global memory. Beyond the difference in the amount of memory traffic, regularity also helps to reserve good row buffer locality. Open-page policy guarantees row buffer locality, but unlimited accesses to an open row might starve the requests to other rows. Under such constraint, a GPU simulator with detailed memory system should be able to provide design choices to compromise throughput and fairness. In our implementation, we provide the parameter *Maximum Access Count* to balance the row buffer locality and fairness.

3.2 NVRAM Timing and Energy Models

Access protocol: VxGPUSim uses the conventional memory access protocol to simulate DRAM. But when simulating NVRAM it uses only four commands from DRAM access protocol: ACTIVATE, READ, WRITE and PRECHARGE. NVRAM needs no REFRESH command because of its non-volatility. Similar to DRAM memory access protocol, *ACTIVATE* is used to select, sense and load a row from NVRAM array into the row buffer; *READ* and *WRITE* commands are used to transfer data bits between row buffer and I/O pins. Note that *PRECHARGE* does not mean the same thing as it does for DRAM. In DRAM system, *PRECHARGE* is used to reset row buffers and bitlines and prepare them for upcoming *ACTIVATE* command to a different row within the same bank [8]; however, in our NVRAM access protocol, *PRECHARGE* also carries out the actual NVRAM array writes, because we assume that, in NVRAM, the row buffer serves as a write-back cache such that energy- and latency-expensive array writes occur only when a dirty row is evicted and written back into the NVRAM array. The reduced write traffic is also essential to improve NVRAM's limited lifetime.

Timing Model: Due to the destructive behavior of DRAM array reads, the data in row buffer must be restored back to the DRAM array before precharging current row and activating another new row. In order to overlap the restoration time, the memory device actively restores data from the row buffers to the DRAM arrays while the memory controller continuously issues column access commands. Thus, data accesses in typical DRAM devices could be simply divided into two steps, accessing memory arrays and accessing row buffers. Accessing row buffers incur similar latency and energy costs for both DRAM and NVRAM since the peripheral circuits of row buffers are independent of memory technology [9].

Though the main parameters in our timing model have slightly different meanings, we still follow the naming conventions in current DRAM timing models. In modern DRAM devices, the row cycle time (tRC) represents the minimum delay for consecutive accesses to different rows within the same bank and is limited by the duration of the write cycle [8]. Basically, tRC must account for the row access time ($tRCD$), the column access time ($tCAS$), the data burst duration ($tBURST$), the write data restore time (tWR) and the precharge time (tRP) [8]. As a result, $tRC \geqslant tRCD + tCAS + tBURST + tWR + tRP$, where $tRCD$ and tRP quantify the memory array read and write latencies, respectively. In order to meet the power budget on a single memory device, the issuing frequency of NVRAM array access commands (*ACTIVATE* and *PRECHARGE*) are constrained by $tRRD_{act}$ and $tRRD_{pre}$ [9]. Globally, $tFAW$ (Four bank Activation Window) limits the minimal time interval that four *ACTIVATE* commands can be issued in order to limit the peak current profile in NVRAM devices. Other parameters are independent of memory technology, such as $tCAS$, $tCCD$ (Column-to-Column Delay), $tBURST$, and $tWTR$ (Write To Read delay time) that constrain consecutive buffer commands, and tWR and $tRTP$ that ensure data stability on peripheral circuits. Considering that NVRAM needs no refresh, we nullify timing parameters that control the refresh operations in DRAM's timing model, including *Refresh time*, $tRFC$ (Auto Refresh Command Period) and $tREFI$ (Average Peri-

odic Refresh Interval).

Energy Model: The energy consumption of DRAM chips can be easily modeled using the DRAM power calculators [14] from MICRON, which take as input parameters current values from available DDR3 DRAM datasheets. However, due to the unknown manufacture details of practical NVRAM chips, it becomes infeasible to continue using this model to simulate the energy consumption of NVRAM chips. Thus we develop a dual-mode energy model for both DRAM and NVRAM as in [16]. The "current" mode follows the same way by which DRAM power calculators work, while the "energy" mode takes as input energy values from third-party tools such as CACTI [15] and NVSim [3]. Another purpose of this dual-mode is to maintain the flexibility of simulating various emerging memory technologies.

Determining NVRAM Parameters: Based on a wide-range survey of emerging NVRAM memories, Meza et al. [12] summarized the energy and latency ranges of NVRAM that are proportional to DRAM. Using those coefficients, we can easily derive the needed parameters for NVRAM to set up a fair comparison. Besides, the method in [16] is also applicable to derive timing/energy parameters from the results of NVSim [3].

4. EXPERIMENTAL EVALUATION

4.1 Methodology

In order to validate our simulator and to demonstrate its ability to explore Memory Controller design, we model a baseline memory system with GDDR5 following the parameters in Hynix GDDR5 SGRAM H5GQ1H24AFR [7]. In this memory system, each memory partition is equipped with 512MB memory, resulting in 3GB global memory in total. We also modeled DDR3 memory device according to Micron DDR3 MT41J256M8HX-15E specifications in [13]. Based on this model, we derived the timing and current parameters for PCM and STT-RAM using the the ranges from [12]. The SM is based on the NVIDIA's Fermi Architecture. The characteristics of simulated GPU cores are listed in Table 1. For the simulation of hybrid memory, we include 2 DRAM ranks and 4 PCM ranks in each memory partition, the resulting size is 3GB. The same partition size is used for the memory systems with only DRAM or PCM.

Table 1: Core Parameters

Architecture Configuration	15 SM clusters (30 SMs), Butterfly network (1400Mhz), 6 Memory Partitions
SM Pipeline	1400Mhz, Pipeline Width:16, threads per Warp:32, Maximum threads per SM:1024
On-chip Memory (per-SM)	Constant: 8KB/64B-line/24-way, Texture: 12KB/128B-line/2-way, DL1: 16KB/128B-line/8-way, Registers: 32768, Shared Memory: 48KB
Unified L2$	768KB, 256B line, 8-way
DDR3 Device	DDR3-667Mhz, x8, 1.5V, Cacheline interleaving
Memory Controller	FR-FCFS, Channel:2, Transaction Queue:128/256 entries, Command Queue:16/32/64/96/128 entries, Open/Close-page
Timing(cycles)	DRAM - tRCD: 12, tRP: 8, tRRDact: 5, tRRDpre: 5, Refresh time: 64ms, tRFC/tREFI: 64/7.8μs PCM - tRCD: 37, tRP: 100, tRRDact: 3, tRRDpre: 18, Refresh time: N/A, tRFC/tREFI: N/A STT-RAM - tRCD: 37, tRP: 20, tRRDact: 5, tRRDpre: 8, Refresh time: N/A, tRFC/tREFI: N/A

4.2 Exploring Memory System Design

In order to showcase the features that are provided by VxGPUSim, we select a set of memory-intensive programs from the benchmarks used in the previous GPU research work. Table 2 summarizes the main characteristics of these applications we use.

In this section, we mainly report the results of evaluating row buffer locality. For all experiments, we use per_rank_per_bank queuing structure and the partition-channel-row-bank-rank-col address mapping scheme, because of their consistent performance.

4.2.1 Row Buffer Locality

Row buffer locality is defined as $Num_{accesses}/Num_{acts} - 1$, where $Num_{accesses}$ is the total number of memory requests and Num_{acts} is the total number of *ACT* commands that are used by the memory controller to activate a row in the memory chips. In our simulator, a parameter *Maximum Access Count* is provided to explore how performance and fairness are effected by row buffer locality .

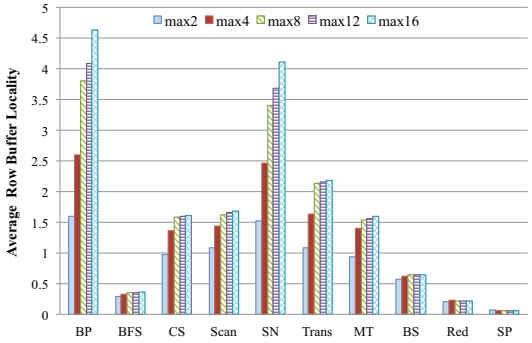

Figure 2: The impact of Maximum Access Count on locality

Figure 2 shows the average row buffer locality when using different *Maximum Access Count*. It can be seen that *BFS, BS, Red* and *SP* exhibit limited sensitivity to this parameter due to their uncoalesced access patterns. In addition, *CS, Scan, Trans, MT* and *BS* become insensitive once *Maximum Access Count* increases beyond 8. This means that 8 is a good choice for Maximum Access Count in order to capture all available concurrent accesses for these four benchmarks. Meanwhile, average data localities of *BP* and SN monotonically increase when Maximum Access Count changes from 2 to 16. This indicates that their access patterns have good regularity and can benefit from higher *Maximum Access Count*.

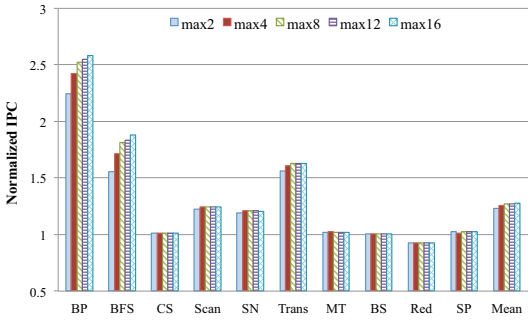

Figure 3: The impact of Row Buffer Locality on IPC

Figure 3 shows the correlation between row buffer locality and IPC. The baseline configuration is the *close-page* policy. On average, using 2, 4, 8, 12, 16 as the Maximum Access Count increases IPC by 25%, 27.7%, 28%, 28%, 28.3% respectively. This adequately demonstrates the advantage of *open-page* policy over *close-page* policy and the strong correlation between row buffer locality and the performance of GPGPU computing kernels.

Figure 4 shows the impact of row buffer locality on power consumption. According to the currently employed power model, the total power consumption can be broken down into four parts, including *Background*, *Activation/Precharge (ACT/PRE)*, *Burst R/W*, and *Refresh*. Only *ACT/PRE* is directly related to row buffer locality because good locality reduces precharge/activation operations (i.e., memory commands). Meanwhile, row buffer locality

Table 2: Benchmark Characteristics (BP:Backprop, BFS:Breadth First Search, CS: ConvolutionSeparable, SN: SortingNetworks, Trans: Transpose, MT: MersenneTwister, BS: BlackScholes, Red: Reduction, SP: ScalarProd, VD: VectorAddition)

	BP	BFS	CS	Scan	SN	Trans	MT	BS	Red	SP	VD
Suite	Rodinia	Rodinia	SDK	SDK	SDK	SDK	SDK	SDK	SDK	SDK	SDK
#Total blocks	8192	46896	55296	819728	73984	212992	384	7680	389	128	196
#Total warps	65536	750336	147456	6557824	692224	1703936	1536	30720	3077	1024	1568
Million Instr.	190	460	2369	16119	4472	2458	4311	6237	407	24	1

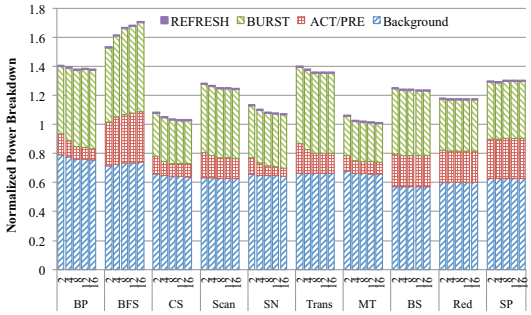

Figure 4: The impact of Row Buffer Locality on Power

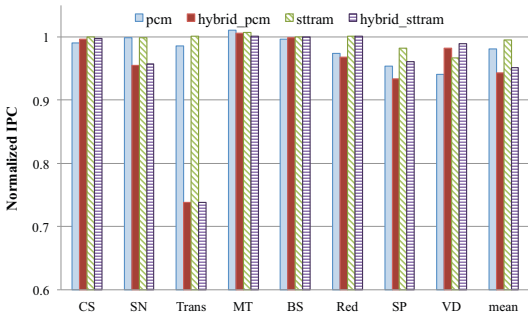

Figure 5: IPC Comparison among DDR3, PCM and STT-RAM

indirectly decreases the background power by making the memory ranks busy, thus avoiding the transition to the low power mode. Among the 10 benchmarks, all except *CS* and *MT* incur higher power consumption with *open-page* policy compared to *close-page*. This means that the idle time of global memory is effectively reduced. By increasing the *Maximum Access Count*, we observe obvious power reduction of *ACT/PRE* in *BP, CS, Scan, SN, Trans, MT* and *BS*, which reflects their better locality as shown in Figure 2.

4.3 Exploratory Memory Hybridization for GPU Global Memory

To demonstrate the capability of our memory simulator in enabling non-volatile memory-based GPU global memory for general purpose computing kernels, we have studied two popular non-volatile memory technologies PCM and STT-RAM, with four configurations: *PCM only*, *PCM+DDR3*, *STT-RAM only*, and *STT-RAM+DDR3*. In each case, the total memory size is 3GB, among which 33% is DDR3 for the hybrid cases.

4.3.1 Performance, Power and Energy

We use 8 benchmarks (*CS, SN, Trans, MT, BS, Red, SP* and *VD*) to evaluate how our simulator can reveal the performance impact of these memory configurations. Given the fact that NVRAM read normally has DRAM-like performance and energy consumption but NVRAM write is much slower and more energy-hungry. We exploit this characteristic in re-designing those benchmarks by allocating write-friendly arrays into DRAM and read-friendly arrays into NVRAM. Figure 5 shows the normalized IPCs achieved by different memory configurations. On average, both pure-PCM and hybrid PCM+DDR3 have 2% performance loss; pure STT-RAM has 1% performance loss, but hybrid DDR3+STT-RAM has no performance loss. These results show that our simulator can reveal the impact of different memory compositions and the sensitivity of different kernels to such compositions.

Figure 6 shows that the power consumption (normalized against DDR3) can vary a lot depending on both the memory system design and the kernel characteristics. Unsurprisingly, *pure-PCM* system constantly incurs higher power consumption because PCM's write operations cost roughly 50 times more energy than DRAM. As a more promising DRAM alternative, *STT-RAMs* exhibit better power performance than pure DRAM in both *STT-RAM only* and *STT-RAM+DDR3* configurations. Both *PCM+DDR3* and *STT-RAM+DDR3* configurations exhibit great potential in improving energy efficiency. Figure 7 shows the normalized energy consumption comparison among DDR3, pure PCM, PCM+DDR3, pure STT-RAM and STT-RAM+DDR3 configurations. Unsurprisingly, pure-PCM system constantly incurs the highest energy consumption because of its expensive write accesses. However, PCM+DDR3 configuration on average saves 17% energy. Though pure STT-RAM system can save 11% energy (27% less than pure PCM), the STT-RAM+DDR3 configuration (plus effective utilization) can save another 17% energy.

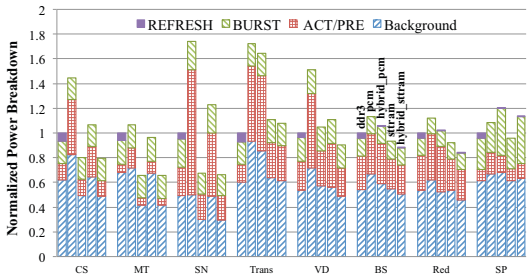

Figure 6: Power comparison among DDR, PCM and STT-RAM

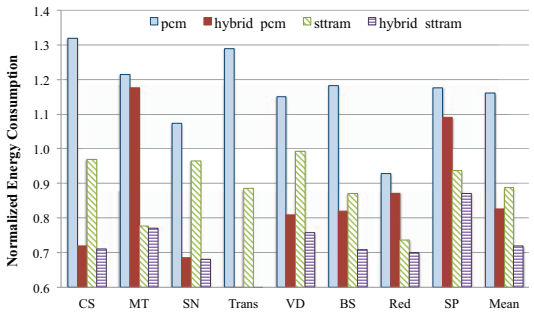

Figure 7: Energy consumption comparison among DDR, PCM and STT-RAM

These results demonstrate that our memory simulator can support different memory configurations and enable further studies on

the tradeoff between memory performance and energy consumption in the design of GPU global memory.

4.3.2 Endurance

Another major concern for applying NVRAM to GPU is its limited write endurance. PCM and STT-RAM might be quickly worn out due to massive parallelism of GPU. Using the average memory traffic summarized in Table 3 and the endurance analytical model in [17], we find that the baseline pure PCM based global memory that has a capacity of 3GB can only last 0.2 year, when assuming a cell endurance of 10^8 writes [9]. By exploring a balanced wear-leveling scheme that can distribute writes, we show that write traffic to PCM can be significantly reduced in the PCM+DDR3 configuration (hy_pcm), from 27.6 to 3.4 bytes/cycle as shown in Table 3. STT-RAM has an endurance (4×10^{12} cycles [6]), that is orders of magnitude higher than that of PCM. Our simulator also reflects this strength of STT-RAM and shows that the STT-RAM+DDR3 configuration (hy_sttram) can bear tremendous write traffic.

Table 3: Average Write traffic to PCM/STT-RAM (bytes per cycle)

config.	CS	MT	SN	Trans	VD	BS	Red	SP	Mean
pcm	25.4	27.5	30.7	53.8	43.7	39.7	0.011	0.2	27.6
hy_pcm	0	27.3	0	0	0	0	0	0.2	3.4
sttram	25.6	27.4	30.6	54.6	44.9	39.9	0.012	0.2	27.9
hy_sttram	0	27.2	0	0	0	0	0	0.2	3.4

5. RELATED WORK

Various GPU simulators have been developed for architectural simulation of general purpose applications. GPGPU-Sim [2] models detailed microarchitectural features of NVIDIA-like GPU devices and executes original CUDA and OpenCL code; MacSim [5] simulates x86 and NVIDIA PTX instructions in detailed heterogeneous/homogeneous micro-architectural behaviors. GPUWattch [10] enables power simulation inside GPGPU-Sim via a configurable cycle-level power model. There are also some simulators for GPU graphical processing. For example, TEAPOT [1] is a full system GPU simulator for the mobile GPUs and supports the OpenGL ES 1.1/2.0 API. However, none of the simulators mentioned above focuses on the intricacy of GPU global memory, nor do they explore GPU global memory that is composed of different memory devices.

Moreover, a lot of work tried to improve GPU efficiency by incorporating new memory organizations. For instance, Satyamoorthy [18] evaluated the potentials of employing STT-RAM based GPU shared memory in terms of performance, area and energy; Goswami et al. [4] proposed differential memory update based STT-MRAM register file and hybrid shared memory for GPUs to optimize both power and performance; Zhao et al. [24] proposed the graphics memory organization of hybridizing DRAM, STT-RAM and RRAM as well as an adaptive data migration mechanism to improve memory bandwidth and reduce power consumption. However, this work mainly focuses on the hybrid graphics memory organization of DRAM and NVRAM. Our previous work [20, 19] has much relevance to this paper but does not provide as rich set of functionalties and as detailed NVRAM models.

6. CONCLUSIONS

In this paper, we have undertaken an effort to design and develop a versatile tool called VxGPUSim that can enable GPU global memory with a variety of different compositions and support accurate simulation on the global memory's performance, timing and power consumption levels. Accordingly, we introduce a unified memory address space for integrating NVRAM with DRAM into the same GPU global memory. A comprehensive set of experiments have been conducted to validate VxGPUSim and study the sensitivity of applications to different memory parameters, such as the number of rank, the request queue size and the open/close page policy, etc. Our experimental evaluation demonstrates that VxGPUSim can faithfully reveal the internal execution details of GPU global memory and that it can be used to facilitate future research on the applicability and benefits of composite memory for GPU systems.

Acknowledgments

This work is funded in part by an Alabama Innovation Award, by an NSF award CNS-1059376 and by DOE Early Career Award. We are very thankful for GPU equipment donated from NVIDIA to Auburn University.

7. REFERENCES

[1] J. M. Arnau et al. TEAPOT: a toolset for evaluating performance, power and image quality on mobile graphics systems. In *ICS*, 2013.

[2] A. Bakhoda et al. Analyzing CUDA workloads using a detailed GPU simulator. In *ISPASS*, 2009.

[3] X. Dong et al. NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 31(7):994–1007, 2012.

[4] N. Goswami et al. Power-performance co-optimization of throughput core architecture using resistive memory. In *HPCA*, 2013.

[5] HPArch, Georgia Institute of Technology. Macsim simulator. http://code.google.com/p/macsim.

[6] Y. Huai. Spin-transfer torque MRAM (STT-MRAM): Challenges and prospects. *AAPPS Bulletin*, 18(6), 2008.

[7] Hynix. 1Gb (32Mx32) GDDR5 SGRAM H5GQ1H24AFR (Rev1.0). http://www.hynix.com.

[8] B. L. Jacob et al. *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann, 2008.

[9] B. C. Lee et al. Architecting Phase Change Memory as a Scalable DRAM Alternative. In *ISCA*, 2009.

[10] J. Leng et al. GPUWattch: Enabling Energy Optimizations in GPGPUs. In *ISCA*, 2013.

[11] Z. Liu et al. PCM-Based Durable Write Cache for Fast Disk I/O. In *MASCOTS*, 2012.

[12] J. Meza et al. Evaluating Row Buffer Locality in Future Non-Volatile Main Memories. Technical report, Carnegie Mellon University, 2012.

[13] Micron. MICRON DDR3 SDRAM MT41J256M8. http://www.micron.com/products/dram/ddr3-sdram.

[14] Micron. Technical Note TN-41-01: Calculating Memory System Power for DDR3, 2007.

[15] N. Muralimanohar et al. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0. In *MICRO*, 2007.

[16] M. Poremba et al. NVMain: An Architectural-Level Main Memory Simulator for Emerging Non-volatile Memories. In *ISVLSI*, 2012.

[17] M. K. Qureshi et al. Scalable High-Performance Main Memory System Using Phase-Change Memory Technology. In *ISCA*, 2009.

[18] P. Satyamoorthy. STT-RAM for Shared Memory in GPUs. Master's thesis, University of Virginia, 2011.

[19] B. Wang et al. Exploring Hybrid Memory for GPU Energy Efficiency through Software-Hardware Co-Design. In *PACT*, 2013.

[20] B. Wang and W. Yu. Performance and Power Simulation for Versatile GPGPU Global Memory. In *IPDPS (PhD forum)*, 2013.

[21] H. Yoon et al. Row buffer locality aware caching policies for hybrid memories. In *ICCD*, 2012.

[22] E. Zhang et al. On-the-Fly Elimination of Dynamic Irregularities for GPU Computing. In *ASPLOS*, 2011.

[23] W. Zhang and T. Li. Exploring Phase Change Memory and 3D Die-Stacking for Power/Thermal Friendly, Fast and Durable Memory Architecture. In *PACT*, 2009.

[24] J. Zhao and Y. Xie. Optimizing bandwidth and power of graphics memory with hybrid memory technologies and adaptive data migration. In *ICCAD*, 2012.

[25] P. Zhou et al. A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology. In *ISCA*, 2009.