Exploring Applications of STT-RAM in GPU Architectures

Xiaoxiao Liu, Mengjie Mao, Xiuyuan Bi, Hai Li, *Fellow, IEEE*, and Yiran Chen, *Fellow, IEEE*

***Abstract*— Use of modern GPUs has been extended from traditional 3D graphic processing to computing acceleration of many scientific, engineering, and enterprise applications. In modern GPUs, on-chip memory capacity keeps increasing to support thousands of chip-resident threads. For example, a large register file is needed in order to efficiently process highly-parallel threads in *single instruction multiple thread* (SIMT) fashion, and a large shared memory is often implemented to allow data sharing among the threads on the chip. On-chip memory capacity of GPUs, however, is highly constrained by large memory cell area and high static power consumption of conventional SRAM implementation. In this work, we propose to utilize the emerging *multi-level cell* (MLC) *spin-transfer torque RAM* (STT-RAM) technology to implement register file and shared memory in GPUs. Compared to SRAM, MLC STT-RAM (or MLC-STT) has a much smaller cell area as well as ultra-low standby power, thanks to the non-volatility of MLC-STT technology. Hence, the footprint and leakage power of the implemented memory components are substantially reduced. Moreover, in light of asymmetric performance of soft and hard bits of a MLC-STT cell, we propose a dynamic data remapping strategy in register file and shared memory implementations that allows a flexible tradeoff between the memory access time and the available capacity: frequently-accessed data is always mapped to the fast rows built with the soft bits of the MLC-STT cells while the slow rows composed of the hard bits are used only when a larger capacity is critically needed. We also develop a novel rescheduling scheme to minimize the waiting time of the issued warps to access register banks in the register file, which is induced by the long writeback operations through the reordering of the issued warps. Finally, an early termination technology is also applied to save the write energy of the shared memory if the bits of the memory do not flip. Experimental results on benchmarks of ISPASS2009, Rodinia, Parboil, and CUDA show that on average, MLC-STT register file can achieve 3.28% system performance improvement, 9.48% energy reduction, and 38.9% energy efficiency improvement compared to conventional SRAM-based design. Meanwhile, MLC-STT shared memory leads to 3.45% system performance improvement, 49.3% energy reduction, and 116% energy efficiency improvement.**

***Index Terms*— STT-RAM, multi-level cell, GPGPU, register file, shared memory.**

I. Introduction

Graphic processing units (GPUs) have been widely adopted in both graphic processing and general parallel computing. Besides stand-alone GPU products (such as AMD's *Radeon* [1] and Nvidia's *Tegra* [2]), GPU cores are often integrated with CPU cores on the same die in heterogeneous systems to accelerate the processing of streaming data [3]. Thousands of parallel threads are simultaneously processed to achieve high throughput and computation efficiency. On the one hand, the extremely high execution parallelism requires a big register file in every GPU core to retain a large volume of active threads. Once a thread is suspended by memory accesses, the GPU can immediately switch to another context with virtually zero latency as long as the required information is held in the register file. On the other hand, GPU employs a on-chip scratchpad memory, called shared memory (Nvidia's terminology) to reduce the costly round trip access to global memory. This design offers short latency and high bandwidth of the data sharing between the threads in a thread block. In Nvidia's Fermi GPU architecture, for example, a highly banked shared memory is implemented with the same access latency as that of the register file if there is no bank conflict [4].

Fig. 1 summarizes the deployment of on-chip memories in four recent Nvidia GPU products [5]–[7]. All on-chip memory sizes increase rapidly. Especially register file capacity almost doubles per generation, significantly exceeding the growing speed of any other memory types. Since Nvidia's Fermi architecture, a maximum 48KB shared memory has been implement to support a large number of active threads that are processed in parallel. Moreover, as the number of active blocks depends on the number of registers totally requested and the amount of shared memory per thread block required for a given kernel [6], it is desirable to increase the number of active blocks on an streaming multiprocessor (SM) as it would increase the system performance by masking latency. As such, the conventional on-chip memory implemented with large and leaky SRAM cells results in significant area and power consumption: previous research showed that register file and shared memory in a GPU occupy 13.4% and 4.9% of chip area [8], [9] and accounts for 18% and 6.7% of system power consumption [10], [11], respectively. The capacity of register

Manuscript received February 17, 2020; revised July 9, 2020 and September 3, 2020; accepted October 5, 2020. Date of publication November 16, 2020; date of current version December 21, 2020. This work was supported in part by the National Science Foundation (NSF) under Grant 1955246, Grant 1725456, and Grant 1910299 and in part by the U.S. Department of Energy (DOE) under Grant DE-SC0018064. This article was recommended by Associate Editor Y. Zhang. *(Corresponding author: Xiaoxiao Liu.)*

Xiaoxiao Liu is with AMD Inc., Santa Clara, CA 95054 USA (e-mail: xiaoxiao.liu@amd.com).

Mengjie Mao is with Waymo LLC, Mountain View, CA 94043 USA (e-mail: meng.j.mao@gmail.com).

Xiuyuan Bi is with Black Sesame Technologies, Santa Clara, CA 95050 USA (e-mail: xiuyuan.bi@bst.ai).

Hai Li and Yiran Chen are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: hai.li@duke.edu; yiran.chen@duke.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2020.3031895.

Digital Object Identifier 10.1109/TCSI.2020.3031895

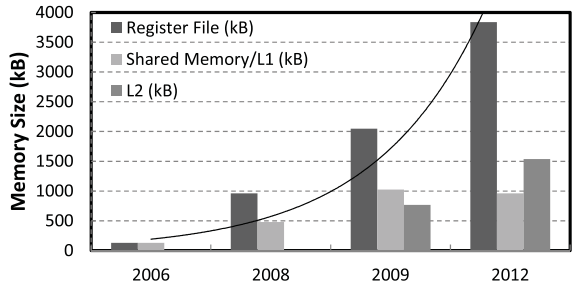

Fig. 1. Deployment trend of register file and other on-chip memories in GPUs.

TABLE I
CHARACTERISTICS AND REGISTER FILE USAGE STATISTICS OF THE SELECTED GPU BENCHMARKS (SHM:SHARED MEMORY)

Application	Abbr.	Regs/ Thread	Max Warps	Reg Usage	SHM Byte/thread	SHM Usage
Matrix Q Computation	MRI-Q	12	48	56.2%	0	0
Hotspot	HOT	34	24	79.7%	12	35%
BlackScholes	BLK	25	32	78.1%	0	0
Neural Network	NN	21	8	16.4%	0	0
Lattice-Boltzmann Method	LBM	36	27	94.9%	0	0
Breadth First Search	BFS	12	48	56.2%	0	0
Path Finder	PF	13	48	60.9%	8	25%
Streamcluster	SC	12	48	56.3%	0	0
Discrete Cosine Transform	DCT	32	16	50.0%	8	8.3%
Back Propagation	BP	11	48	51.5%	4	96%
StoreGPU	STO	33	12	38.4%	127	99%
N-Queens solver	NQU	16	9	14.0%	164	96%

file and shared memory is anticipated to continue increasing given by the rapidly growing active thread number in each *streaming multiprocessor* (SM) [12].

Although the needed capacity of register file and shared memory in GPU generally increases, the actual usage varies significantly in different applications. We selected and simulated ten popular GPU workloads from [10], [13], in which the average register file usage ratio ranges from 27.2% to 92.9%, and the shared memory usage ratio ranges from 0% to 99%, as shown in Table I. A low usage ratio potentially leads to a big waste of the capacity and leakage power.

Spin-transfer torque random access memory (STT-RAM) is a promising technology to replace SRAM in on-chip memory implementation. Its simple cell structure and non-volatility enable much smaller cell area than SRAM and almost zero standby power [14], [15]. Moreover, the recent invention of *multi-level cell* (MLC) design [16] doubles the storage density of STT-RAM. The use of MLC STT-RAM (MLC-STT) in last level cache of CPUs has been widely investigated for energy efficiency enhancement [17]–[19].

This work extends the application of MLC-STT to register file and shared memory in GPUs. Thanks to the simple and dense cell structure of STT-RAM, the memory area can be cut down to 27.7% of SRAM implementation. Moreover, the doubled storage capacity can effectively relax the limitation on the number of active thread blocks due to insufficient register file and shared memory size in memory-hungry applications and hence improve system performance, as we shall show in Section V. However, the hard and the soft bits in a MLC-STT cell demonstrate very different access time requirements. Thus, we propose a remapping strategy to relocate data based on its access frequency and the memory usage requirement of the application. Particularly, the frequently-accessed data is always mapped to the fast rows built with the soft bits of the MLC-STT cells while the slow rows composed of the hard bits are used only when a larger capacity of the register file or shared memory is needed. Moreover, for MLC-based register file, a run-time warp rescheduling scheme is developed to reorder the issuing of the ready warps to minimize the access stall induced by the long write operations of MLC-STT. Finally, a write termination scheme is designed to reduce the useless write operation with high energy in the MLC-based shared memory. Compared with the existing research on the application of STT-RAM in micro-architecture, our major contributions include:

1) We propose using MLC-STT to implement register files and shared memory in GPUs – a highly scalable design in terms of storage capacity (density) and area/power cost;
2) A remapping strategy is designed to perform a flexible tradeoff between the access time and the capacity of the register file and shared memory based on the access time difference between the soft and the hard bits of a MLC-STT cell and the run-time register file access patterns;
3) A novel warp rescheduling scheme is also developed to minimize the access stall introduced by the slow write access to the MLC-STT register banks.
4) An early write termination scheme is introduced to reduce the redundant write to MLC-STT based shared memory to save power and latency.

Our simulation results show that compared to conventional SRAM-based register file and shared memory, our MLC-STT designs respectively achieve on average 3.28% and 3.45% system performance improvements on system performance across selected benchmarks. The average energy efficiencies of register file and shared memory are greatly enhanced about 38.9% and 116%, respectively.

The rest of our paper is organized as follows: Section II introduces the basics of STT-RAM technology and major design challenges of GPU register file and shared memory; Section III presents the design details on the MLC-STT on-chip memories with different configurations as well as the remapping strategy, warp rescheduling scheme, and early write termination; Section IV describes the experimental methodology and system setup of our evaluations; Section V gives our simulation results and discussions; Section VI summarizes the related works; Section VII concludes our work.

II. PRELIMINARY

A. STT-RAM and MLC STT-RAM

In a *single-level cell* (SLC) STT-RAM cell, binary data is represented by two resistive states of a *magnetic tunning junction* (MTJ) device. As shown in Fig. 2(a-b), a MTJ

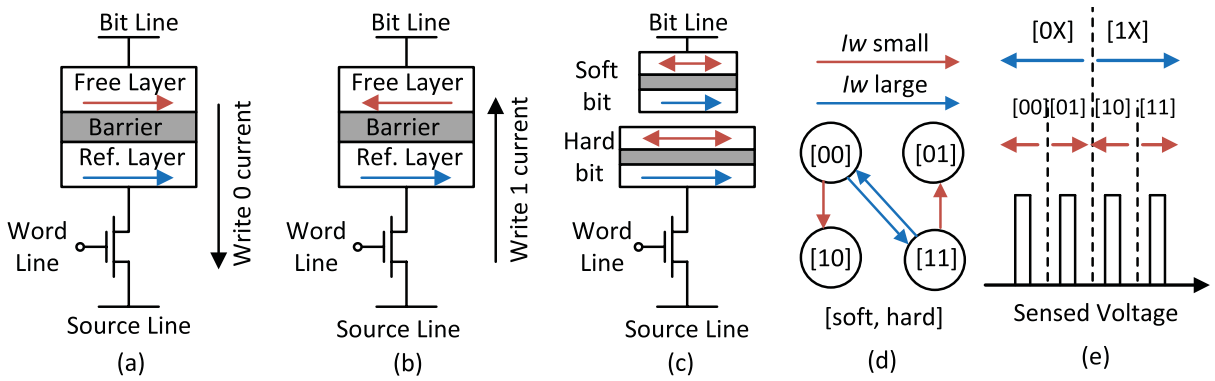

Fig. 2. STT-RAM cell structure and operation diagram. (a,b) single level cell (SLC) represents "0" and "1". (c) multi-level cell (MLC) using serial stacking structure. (d) 2-step write transition diagram. (e) 2-step read operation.

consists of two ferromagnetic layers and one metal oxide barrier (e.g., MgO) in between. One ferromagnetic layer has a fixed magnetization direction and hence is referred to as *reference layer* [20]. The magnetization direction of the other ferromagnetic layer (named as *free layer*) can be flipped by a spin-polarized current. When the magnetization directions of the two ferromagnetic layers are parallel, the MTJ is at low resistance state, representing logic "0"; when the magnetization directions of the two layers are anti-parallel, the MTJ is at high resistance state, corresponding to logic "1". The data stored in a STT-RAM cell can be read out by injecting a small read current and comparing the generated sensing voltage on the bit-line (BL) with a reference.

To further enhance the data storage density of STT-RAM, MLC-STT technology was proposed by integrating two MTJs in a cell [21], [22]. Fig. 2(c) illustrates a popular serial MLC-STT cell structure which is composed of a small MTJ and a larger MTJ. The combination of the resistance states of the two MTJs constructs four different resistance levels, representing a 2-bit data. The bits determined by the resistance states of the large and small MTJs are normally denoted as "hard bit" and "soft bit", respectively.

As the two MTJs are connected in serial and driven by the same NMOS access transistor, the same current passes through both MTJs during read and write operations. When applying a large switching current to program the large MTJ, the small MTJ encounters an even larger switching current density and hence will turn to the same resistance state as the large MTJ. In other words, programming the hard bit enforces both the hard and soft bits to be aligned and eventually switch to the destination value of the hard bit, no matter what is the original value of the soft bit. If the target value of the soft-bit is different from the one of the hard-bit during a write operation, an additional programming step with a small write current is required to program only the soft-bit to the target value. During the programming of soft-bit, however, a small current shall be applied in order to flip only the resistance state of the small MTJ. Such a 2-step write operation of MLC-STT cells is depicted in Fig. 2(d). Similarly, a 2-step read operation is required to detect the soft and hard bits in sequence, as shown in Fig. 2(e). Notably, programming or detecting only the soft bit can be completed in one step.

When implementing memory with MLC-STT cells, the hard and soft bits can be separated into different data sets [19]. For instance, a row of MLC-STT cells in a physical array can be regarded as two logic rows that respectively consist of the hard bits and soft bits of these MLC-STT cells. As such, reading or writing the soft-bit row requires only one step. Accessing a hard-bit row, in contrast, need to take two-step operations, resulting in much longer access latency and higher energy consumption. Such performance difference between hard-bit and soft-bit rows motivates our remapping strategy in the designs of MLC-STT GPU register file and shared memory.

B. Modern GPU Architecture

Threads are executed in *single instruction multiple thread* (SIMT) fashion in GPUs to achieve high computing parallelism and throughput. Without loss of generality, in this work, we adopt the terminologies of Nvidia and use GTX480 [23] in Fermi family as our baseline model.

A GPU is composed of 16 *streaming multiprocessors* (SMs), each of which includes one GPU processing pipeline. *GPU kernel* is instantiated with a grid of parallel thread blocks. Each SM in Fermi architecture can support up to 8 parallel thread blocks. The maximum number of the threads that can be concurrently executed in one thread block is 1024 with maximum 63 32-bit registers assigned to each thread. 32 threads in one block are grouped as a *warp* to issue. As illustrated in Fig. 3(a), a GPU processing pipeline includes *Instruction Unit*, *Register Address Unit*, *Register File Unit*, and *Execution Unit* [24]. Instruction Unit decodes instructions, and schedules the execution of the instructions based on additional information like priority and date dependency. The instructions selected in a round-robin fashion are sent to Register Address Unit at the end of each clock cycle. Accordingly, Register Address Unit accesses the registers in Register File Unit.

Register File Unit in a SM contains 32,768 32-bit registers to hold instruction operands. Highly banked architecture is usually utilized to realize multi-port access in Register File Unit implementation. In Fermi architecture, one SM integrates 16 banks, each of which contains 64 entries of 1,024 single-port SRAM cells (i.e., 32 registers). As shown in Fig. 3(b), an operand collector in Register File Unit is used

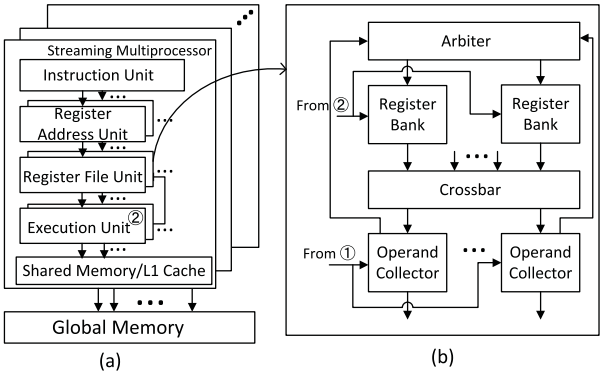

Fig. 3. GPU architecture and register file design in GPUs.

to collect and dispatch the active warps to available banks. When all the operands have been acquired, instructions and operands are sent to Execution Unit.

As shown in Fig. 3(a), shared memory is implemented with L1 Cache together and shares 64KB memory capacity based on configuration. Shared memory is composed of 16 banks, and each bank is designed to support 4 byte read/write per cycle. The data shared by threads in the same block can be stored in share memory for quick access, and programmers are in charge of the use of shared memory and also the avoidance of bank conflict. Before launching a thread block, the CUDA compiler checks the number of available registers and also the usage of shared memory: if the the total usable memory resource is smaller than memory request, the launching will suspend.

III. MLC STT-RAM Based Register File and Shared Memory

In this section, we will describe the proposed MLC-STT register file and shared memory designs in GPU architecture. Besides replacing SRAM cell with MLC-STT cell, we also propose *address remapping* strategy for register file and shared memory, and *warp rescheduling* scheme for register file and *MLC early write termination* for shared memory separately to mitigate the impacts of the slow read and write accesses of MLC-STT and hence improve the performance and energy efficiency.

A. Overview of MLC STT-RAM Based Memory Bank

The memory banks of register file and shared memory in Fermi GPUs share a similar design [4], [25]. As the main difference is only the size of the memory bank, we use MLC STT-RAM based register file as an example to introduce our design.

Fig. 4 illustrates a memory bank of the proposed *MLC-STT based register file* (MLC-RF) in Fermi architecture. Each register file array has 64 physical entries which are supported by a programmable row decoder with 6-bit address. 1,024 MLC-STT cells are contained in each row, corresponding to 2,048 data bits. The capacity of each row is doubled from that of the SRAM or SLC-STT design, allowing more registers to be accessed at a time. Considering the fact that having inactive

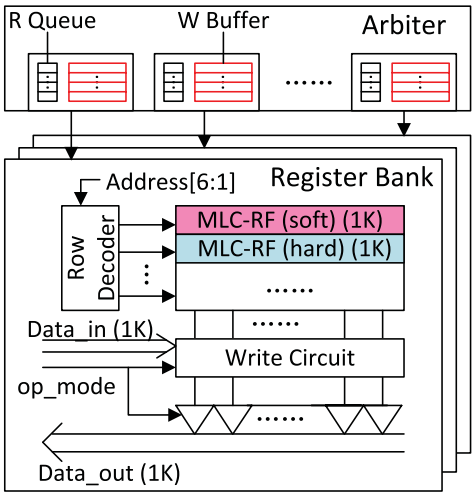

Fig. 4. MLC STT-RAM-based register file.

TABLE II
Comparison of SRAM and STT-RAM Based Register Banks

Technology	SRAM	STT-RAM	
		SLC	MLC
	(64Kb)	(64Kb)	(128Kb)
Memory Area (mm^2)	0.350	0.097	0.097
Total Read time (ns)	0.710	1.32	S: 1.38, H: 1.77
Total Write time (ns)	0.708	8.37	S: 7.68, H: 15.43
Read energy (nJ)	0.044	0.018	S: 0.019, H: 0.025
Write energy (nJ)	0.044	0.088	S: 0.092, H: 0.15
Leakage power per SM (mW)	100.416	1.1875	1.1875

threads in one warp is considerably high [26], a wider row may lower the efficiency of register bank accesses. In this work, we tend to keep the same capacity per row to maintain the same effectiveness of bank access as the SRAM register file design. Hence, we divide a row of 1,024 MLC-STT cells into a fast row with 1,024 soft-bits and a slow row with 1,204 hard-bits. Such a design leads to a register bank with total 128 logic rows, which requires 7-bit address control. Here, address bits [5 : 0] are used for address decoding while address bit [6] decides the access to the soft or the hard bits of the physical entry.

Table II summarizes the design parameters of register banks built with SRAM, SLC-STT, and MLC-STT at 32nm technology node. The results are simulated by CACTI [27] and HSpice with PTM [28] under 32nm technology. In the MLC-STT design, we assume the area of the big MTJ (hard bit) is twice as that of the small MTJ (soft bit), which achieves the best robustness in read and write operations [16]. The timing information of sense amplifier is derived from SPICE simulations. The adopted register file configuration and area information are obtained through a modified NVsim [29]. The areas of the SLC-STT and MLC-STT designs are only about 27% of that of SRAM design while the bit capacity of the MLC-STT design is 2× that of the SLC-STT and SRAM designs. The read and write access times of the SLC-STT and MLC-STT designs are longer than that of the SRAM

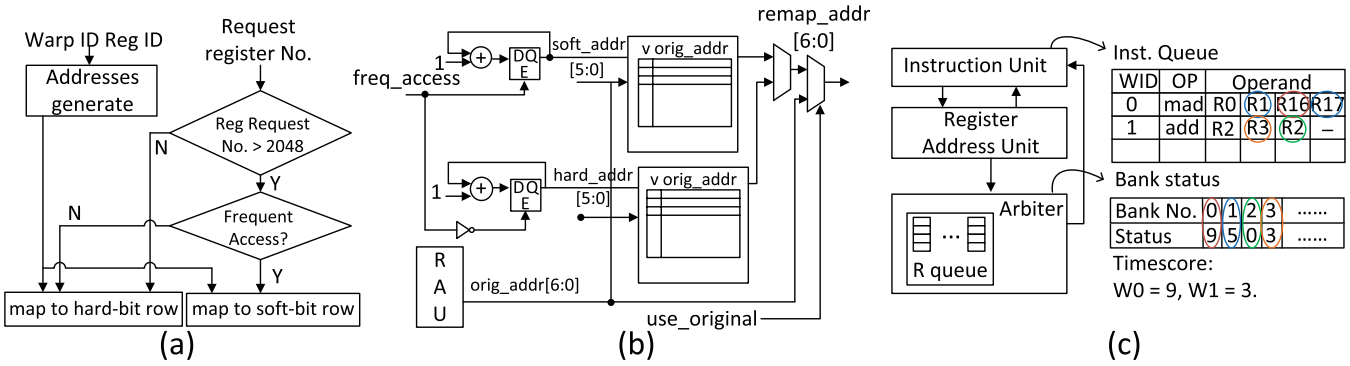

Fig. 5. (a) The register bank address remapping algorithm. (b) The hardware implementation diagram of remapping. (c) Warp rescheduling.

design. Due to the 2-step operation, the hard-bit rows require much longer time to complete a read or write access than the SLC-STT design. In contrast, accessing the soft-bit rows is very similar to the SLC-STT design and their read and write performances are also very close.

Write Buffers are adopted in Arbiters to alleviate the impact of slow access time of the MLC-STT design. Each register bank requires one Write Buffer. The Write Buffer temporally holds the writeback data until the former write is completed and releases write port. The needed number of entries in a write buffer is determined by the MLC-STT write access cycles and the access frequency. In this work, we set the operating frequency of the GPU core to 700MHz. As a result, the write accesses to the soft- and hard-bit rows of the MLC-STT register files take 6 and 12 cycles, respectively. To find the minimum required write buffer size, we carefully increase the number of write buffer entries and rerun the selected benchmarks until overflow disappears. Based on the evaluation, a 4-entry Write Buffer is sufficient to remove the overflow since writebacks occur infrequently and the register file is highly banked.

In the original arbiter design of Fermi architecture, a read queue is assigned to each register bank to arrange all the read requests from different Operand Collectors in a row. The Read Queue structure remains unchanged in our proposed MLC-RF, holding read requests when awaiting the slow MLC write operations to finish. The introduction of Write Buffer incurs 15% increase in MLC-RF bank area. Nonetheless, the overall area of a MLC-RF bank is still less than 30% of the one of SRAM implementation, mainly benefiting from the high data storage density of MLC-STT.

B. MLC-Aware Remapping Strategy

Write Buffers shorten the response time of register files but can not help on reducing the long read and write latency's of MLC-STT banks. As aforementioned in Section II-A, MLC-STT contains soft- and hard-bit rows which have very distinctive read and write performance. It can be beneficial if the GPU maximizes the usage of soft-bit rows for sake of system performance and energy consumption. In other words, the data (especially the one being accessed frequently) shall be mapped to the soft-bit rows before considering the hard-bit rows. In the implementation, run-time information such as how many registers to be accessed and how many times they will be accessed can be obtained from the compiler. During compilation, we first perform profiling on the access frequency of all the registers involved during the execution of a kernel. A bitmap vector is then generated to record whether a register will be placed in a soft-bit row or a hard-bit row. Once the kernel is offloaded to SMs, such a bitmap vector is copied to a special register and guide the register mapping.

Fig. 5(a) depicts the corresponding *register file remapping algorithm*, which requires the support of two mapping tables and one *register address unit* (RAU), as shown in Fig. 5(b). Remapping is activated only when the register demand exceeds the half capacity of the MLC-STT register bank. Otherwise, hard-bit rows will not be utilized so that the register bank works virtually with the same capacity as the SRAM/SLC-STT design, offering an uniformed fast access time of soft-bit rows. When more than half capacity (i.e., 2,048 registers) of the MLC-STT register bank is requested, the first 2,048 registers with frequency accesses will be mapped to the soft-bit rows while the rest will go to the hard-bit rows. The relationship between the original address and the mapped physical address is retained in a remapping table. Each remapping table entry corresponds to one row in the register bank, including 1 valid bit and 7-bit mapped address. Hence, the size of the remapping table is small and the incurred area overhead is negligible.

C. Warp Rescheduling for MLC STT-RAM Based Register File

The long write latency of STT cells (even in the soft-bit rows of MLC-STT) may severely degrade the system performance by blocking other accesses to the same register bank. Fig. 5(c) describes such an example, where two warps – $W0$ and $W1$, are waiting in the instruction buffer in a serial Round Robin manner. Their register requests are mapped to different banks and they do not have data dependency with the current warps being executed. When the access to the register bank that $W0$ requests is held by a writeback from the previous warp, the operand fetching of $W0$ has to wait. Since all the available entries in Scoreboard, Operand Collector, and Read Queue have been held by $W0$, the processing of $W1$ is stalled at the issue stage even the register bank to be accessed by $W1$

is free. In such a case, we may swap the issue sequence of $W0$ and $W1$ to avoid the stalling of $W1$.

Based on the above observation, we propose a *warp rescheduling* scheme to minimize the waiting time of the issued warps for register bank access. The warp rescheduling tends to rearrange the issue order of the ready warps by prioritizing the accesses to the free register banks. The effectiveness of rescheduling is mainly determined by the availability's of the free register banks and the warps ready to be issued, which are generally large in GPU execution. A parameter named $timescore$ for each warp is defined to guide warp rescheduling as:

$$timescore = Max[busy\_cycle[bank\_id_i]\&valid[bank\_id_i] \quad (i = 0, .., 15)]. \quad (1)$$

Here $busy\_cycle$ indicates the remaining cycles to complete the current write operation of the register bank, and $valid$ denotes whether the warp has a register request from this bank. $bank\_id_i$ labels the register bank and is determined by the identifier of the register (reg_id) by:

$$bank\_id = reg\_id \ \% \ bank\_number. \quad (2)$$

Here $bank\_number$ is the total number of register banks in one SM. In Fermi architecture, a SM includes 16 register banks. As shown in Fig. 5(c), a bank status table is added in Instruction Unit to assist acquiring the $busy\_cycle$ of each register bank. $busy\_cycle$ is updated by Register File Unit whenever new write requests arrive and automatically counts down every cycle.

During scheduling, $timescore$ will be checked whenever warps are going to scoreboard for data dependency check. As shown in Eq. (1), the maximum $busy\_cycle$ will be chosen as the $timescore$ of a warp. The warp will be managed to issue only when its $timescore$ is smaller than a threshold. If a warp failed to pass the $timescore$ checking, it will remain in Instruction Buffer and wait for the next round check. The scheduler will continue to check the next available warp until one is issued successfully. Note that data dependency checking and $timescore$ checking can be performed simultaneously. Hence, no timing overhead on scheduling step is introduced. We run extensive experiments and found that 5 is the optimal value of the $timescore$ threshold which maximizes the system performance and energy efficiency of our design.

D. MLC STT-RAM Based Shared Memory

1) Overview: Shared memory per SM in Fermi architecture is composed of 32 banks of memories with 4 bytes bit-width in each bank. The capacity of shared memory is configurable as 16KB or 48KB in total in SRAM implementation. We keep the same memory bank width and peripheral circuit design as MLC-RF shown in Fig. 4. Similar to MLC-RF, a Write Buffer is implemented in each memory bank to hold the write data while waiting for the former write completes. Unlike register file, which already has a read queue corresponding to each register bank, a Read Buffer is also applied for read requests while waiting slow write process to complete.

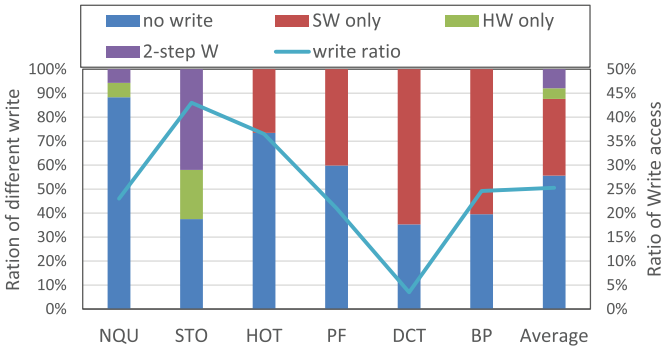

Fig. 6. Actual write bit flip in MLC-STT based shared memory and write access ratio.

2) MLC Remapping Strategy: Like in MLC-RF, we also implement the MLC remapping strategy with the same hardware implementation as shown in Fig. 5(b) in MLC-based shared memory to maximize the utilization of soft-bit row to obtain better performance and efficiency. As most of the read/write to shared memory is tile-based [25], the read/write access to each row is more balanced than register file which has warp-based request. So we only use the required memory capacity of shared memory to determine whether the remapping is activated. If more than half of the MLC shared memory is required, the hard-bit row will be accessed. Otherwise, only soft-bit row will be used.

3) MLC Early Write Termination: The performance of MLC-STT based shared memory is benefiting from its high density, but the high write energy still serves as a major challenge for STT-RAM, especially in MLC-STT which requests 2-step write in its regular write operation. The 2-step write operation consumes about 4× more energy than the read operation in MLC-STT in our evaluation (see Table II). Ping *el at.* [30] designed an early write termination for SLC-STT to shut down the redundant write operation if a same value is going to be written in the STT-RAM cell. Inspired by this work, we propose an early write termination to reduce the costly and unnecessary writes for MLC-STT shared memory(MLC-EWT). After the remapping of soft bit rows and hard bit rows is applied, either only soft bit rows or both soft bit and hard bit rows are used. If only the soft bit of a MLC cell is used, the operation of MLC-EWT is similar to [30]: the programming current of soft bit is cut when the stored data is detected as the same as the data being written. But if both soft bit and hard bit of the MLC cell are used, it becomes much more complex as the 2 bits new data could be 2 bits different, only 1 bit different or 2 bits the same compared to the stored data. We note that soft bit is detected first but hard bit is programmed first during the 2-step read and write access of MLC. Hence, to shorten the latency of decision making and simplify the control logic, we limit read operation to only soft bit to the greatest extent possible, then choose the proper operation from continuing 2-step write, only hard bit write, or no write at all. The details of MLC-EWT cases are summarized in Table III, and the design of MLC-EWT circuit is shown in Fig. 7. The MLC-EWT circuit generates 2 cut signals for turning off programming current

TABLE III

WRITE OPERATIONS OF MLC SHARED MEMORY WHEN BOTH SOFT BIT AND HARD BIT ARE USED. HW–ONLY WRITE HARD BIT, AND 2W – 2-STEP WRITE

		Write data			
		00	01	10	11
Data in cell	00	–	2W	2W	HW
	01	HW	–	2W	HW
	10	HW	2W	–	HW
	11	HW	2W	2W	–

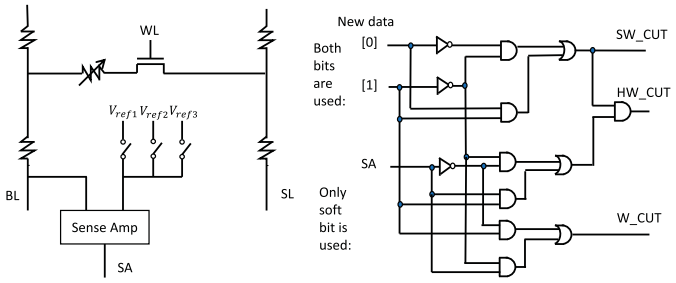

Fig. 7. MLC-EWT circuit.

of hard bit (HW_CUT) and soft bit (SW_CUT) separately during programming of hard bit. The MLC-EWT circuit first detects if the 2 bits of the new data are the same, then uses the result as SW_CUT to turn off soft bit write. Hard bit write is only cut when the stored 2-bit data is the same as the new value. We estimated the area and energy overheads of the MLC-EWT circuit using PTM 32nm HP models in HSPICE. As this circuit is added on each column (e.g., 256 memory cells per column), the average energy overhead of generating the write cut signals is about $2.12fJ$ per column and the area overhead per column is only $16.32um^2$, which are both marginal compared to the write energy of MLC-STT and the total area of memory (Table II).

When writing a line of shared memory, however, it is less likely that the overall processing time of the write can be terminated early because most likely at least one cell requires both steps. Nonetheless, the early terminated write process still substantially reduces the total energy consumption of MLC write operations. Our execution analysis shows that the flipping bits are not the majority of all the bits during write operations, as shown in Fig. 6. On average, only 6% MLC-STT cells actually need 2-step write. Benefit from remapping scheme, some applications only need soft bit write which use less than half of shared memory size, like `HOT`, `PF`, `DCT`, and `BP`. Thus, we designed an early write termination circuit in the MLC shared memory to avoid useless writes.

IV. EVALUATION SETUP

We implement all the designs on GPGPU-sim [10], a cycle-accurate GPU performance simulator, for system functionality verification and performance evaluation. The baseline GPU is configured as Nvidia GTX480 [23]. Table IV summarizes the parameters of our system configuration. *Loose round-robin* (LRR) scheduler, which is widely adopted in GPU warp scheduling, is selected as the basic scheduler in the simulated system. The SM operating frequency is set to 700MHz. We assume a fixed capacity of 64Kb for both SRAM and SLC-STT register banks. When MLC-STT is applied for same-area policy, the bank capacity is raised to 128Kb. The parameters of MLC memory bank are generated from a modified NVsim [29] at 32nm technology node. The STT device parameters from [14] are adopted for cell area estimation. The detailed configurations of register banks can be found in Table II.

TABLE IV

GPGPU-SIM CONFIGURATION

Parameter	Value
Number of SMs	16
Core/Shader/DRAM Frequency	700/1400/924MHz
Register File/SM	128KB
Max Warps/SM	48
Max Threads/SM	1536
Max Thread Blocks/SM	8
Max Registers/Thread	63
Max Threads/Thread Block	1024
L1/Shared Memory	16KB/48KB
Warp Scheduler	Round Robin

Ten GPU workloads from [7], [10], [13] [31] are selected in our performance evaluation. Table I shows that the register file usage of the selected workloads varies from 27.2% to 92.9% and the shared memory usage varies from 0% to 99%, offering a good coverage on all representative cases.

V. EVALUATION RESULTS AND DISCUSSION

In this section, we evaluate the effectiveness of MLC-STT register file and shared memory implementations in improving GPU runtime performance and energy consumption, and compare it with SRAM and SLC-STT options.

A. MLC-STT Based Register File

1) System Performance: Fig. 8 summarizes the GPU performance with different register file configurations over the selected benchmarks. All the results are normalized to that of the baseline with SRAM register file ("SRAM"). It shows that simply replacing SRAM with SLC-STT or MLC-STT without any optimizations ("SLC" and "MLC", respectively) results in on average 7.23% and 6.89% performance degradation, respectively. In particular, significant system performance degradation (>14%) was observed in the applications that have intensive register file accesses, e.g., `MRI-Q`, `STO` and `BLK`, due to the long read and write access latencies of SLC-STT and MLC-STT. However, some benchmarks with high register file usage (i.e., `HOT`, `LBM`, and `BP`) outperform the SRAM baseline, thanks to the doubled register file capacity. Interestingly, `NN` also shows performance improvement even its register file usage is only 27.2% across all kernels. Our detailed analysis found that the register file usage of `NN` is

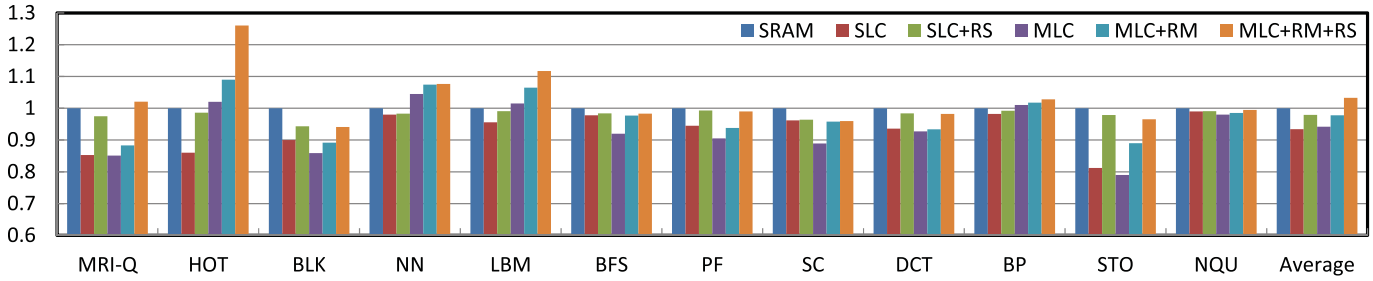

Fig. 8. System performance comparison under different register file configurations. All the results are normalized to that of SRAM baseline design.

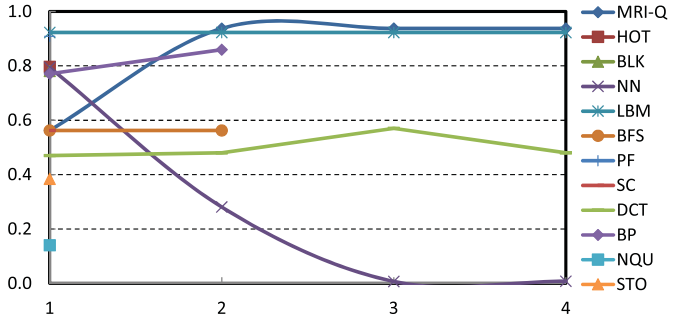

Fig. 9. Register usage of the first 4 kernels in each benchmark.

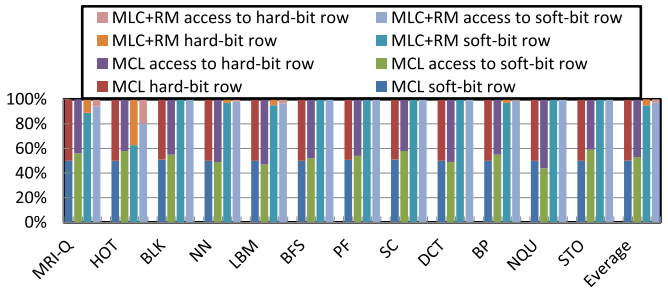

Fig. 10. The statistics of soft-/hard-bit row accesses with/without remapping.

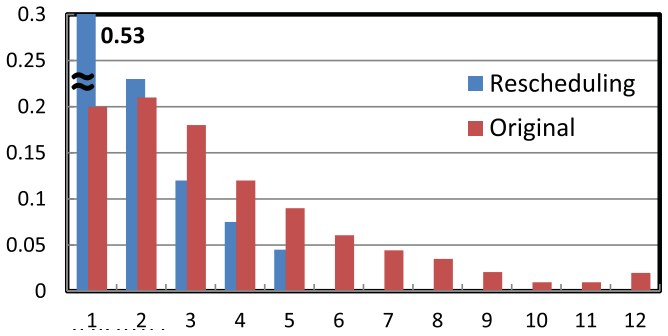

Fig. 11. Rescheduling influence on timescore of issued warps (x-axis: the number of timescore, y-axis:percentage of issued warps).

concentrated on only one kernel, as shown in Fig. 9. Hence, the increased register file capacity greatly raises the number of the threads that can be issued on that kernel, resulting in considerable performance improvement.

When remapping strategy is applied ("MLC+RM"), the two-step read and write operations of MLC-STT register file are minimized, introducing on average 4.17% enhancement in system performance compared to "MLC". Remapping is particularly effective in the benchmarks that are sensitive to register file access latency: In SC and HOT, for example, the performance improvements w.r.t. "MLC" are as high as 7.5% and 7.01%, respectively.

Note that rescheduling scheme can be applied to both SLC-STT and MLC-STT register file designs to alleviate the impact of the long access latency on GPU performance. Hence, we simulated the performances of the GPUs that have 1) the SLC-STT register file with rescheduling scheme ("SLC+RS") and 2) the MLC-STT register file with both remapping strategy and rescheduling scheme ("MLC+RM+RS"), respectively. Simulation results show that the GPU performance of "SLC+RS" or "MLC+RM+RS" substantially improves by 4.47% or 5.89% compared to "SLC" or "MLC+RM", respectively. Compared to SLC-STT, the benefit of rescheduling scheme is more prominent in MLC-STT because of the longer access latency of MLC-STT cells. Among five register-hungry benchmarks (MRI-Q, HOT, NN, LBM, and BP), MLC+RM+RS even outperforms SRAM baseline significantly, say, on average 10.4% speedup. Across all benchmarks, MLC+RM+RS still outperforms SRAM baseline by 2.51%.

2) Effectiveness of Remapping and Rescheduling: To further evaluate the effectiveness of the proposed remapping strategy, we compare the ratio of the accesses to the hard-bit and soft-bit rows and the usage of these two types of rows in different benchmarks before and after the remapping strategy is applied. Fig. 10 shows the statistical results. Without remapping, on average 53% of accesses fall on soft-bit rows, accounting for 50.2% of register bank entries in use. The observation indicates the register file accesses are evenly distributed to the soft-bit and hard-bit rows. The situation changes dramatically after applying the remapping strategy: the accesses to soft-bit rows are greatly promoted to 97.3%. Accordingly, averagely 95.1% of overall soft-bit rows are occupied.

Moreover, for the five benchmarks with less register usage (BLK, BFS, PF, SC, DCT, STO, and NQU), the soft-bit rows are sufficient to supply all the register demand, leading to 100% of access and usage ratios. The other five register-hungry benchmarks, however, need to activate more thread blocks. We attempt to maximize the use of the soft-bit rows and allocate the rest to the hard-bit rows. On average, only 7.5% of hard-bit rows are taken by 11.8% of total register accesses among the five register-hungry benchmarks.

The effectiveness of the proposed rescheduling scheme can be represented by timescore of each warp after it is issued. The statistical results before and after applying the rescheduling

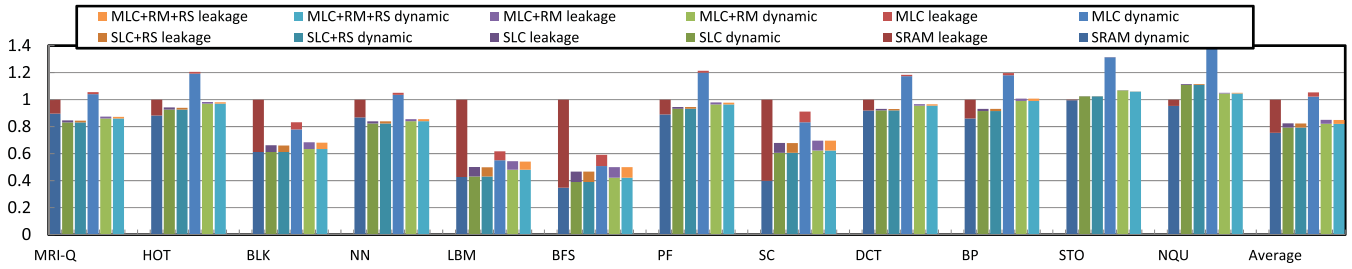

Fig. 12. Register Energy consumption under different register file configurations. All the results are normalized to that of SRAM baseline design.

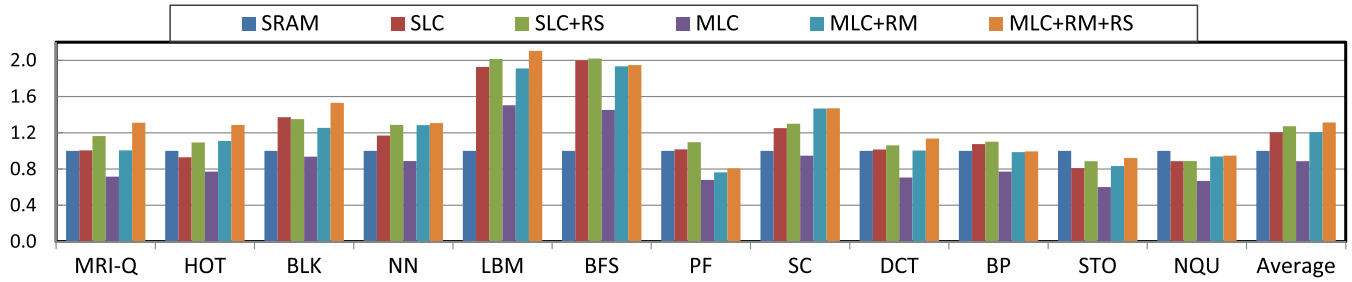

Fig. 13. Normalized energy efficiency.

scheme are compared in Fig. 11. Before scheduling is applied, majority (>80%) of issued warps need to wait for more than one cycle before successfully retrieving the operands from register file. After applying the scheduling, more than 50% warps can immediately access the register file while the maximum timescore of all the warps reduces from more than 12 cycles down to 5 cycles.

3) Energy Consumption and Energy Efficiency: Fig. 12 shows the energy consumption of register files with different configurations, including both dynamic and leakage energies. Again, all the results are normalized to the SRAM baseline. Thanks to the non-volatility of STT-RAM, the leakage energy consumption of register files dramatically reduces when SLC-STT and MLC-STT are applied. However, directly applying SLC-STT and MLC-STT without any optimizations incurs considerable increase in dynamic energy consumption. The higher dynamic power consumption in MLC is due to the complex 2-step write and read operations: On average, SLC achieves approximately 17.6% of energy reduction compared to SRAM baseline while MLC achieves 5.3% total energy saving. In general, the dynamic energy of both SLC-STT and MLC-STT register files dramatically reduces when remapping strategy is applied: On average, MLC+RM consumes 15.0% less energy than SRAM, which is very close to that of SLC. In the benchmarks which have less execution time and require a small amount of register file (e.g., NQU and STO), less energy saving is achieved in both SLC and MLC RF implementations. Note that rescheduling scheme does not reduce the number of write operations. Hence, it does not affect energy dissipation visibly. All the simulations above have taken into account the energy consumed on the additional control circuits.

In this work, we use the ratio of the normalized performance over energy from [32] to measure the energy efficiency of our proposed MLC-STT register file design. Fig. 13 shows the normalized energy efficiency of various register file configurations with and without optimizations. Our design, i.e., MLC-STT register file with remapping strategy and rescheduling scheme ("MLC+RM+RS"), achieves the best energy efficiency in 7 out of 10 benchmarks. On average, it is 40.0% more efficient than SRAM baseline and 14.9% better than the design of SLC+RS.

B. Evaluation of MLC-STT Shared Memory

Our evaluation also shows that SLC-based shared memory design introduces on average 15.4% performance degradation due to the its long write latency, as shown in Fig. 14. The memory write dominant workloads suffer from significant performance loss: NQU and STO show more than 30% performance degradation in the SLC implementation. The naïve MLC implementation even show larger performance degradation in the workloads that do not need a large shared memory, such as PF, BP, HOT, and DCT. However, in STO and NQU which request large shared memory space to enable frequent on-chip data accesses, the system performance benefits from the increased shared memory capacity introduced by MLC. The introduced average performance improvement is 19.4% compared to SLC. Similar to MLC-RF, remapping strategy (MLC+RM) helps to minimize the costly two-step MLC read and write operations. The performance of 6 benchmarks averagely increases by 14.3%, and is close to that of SRAM baseline. When write termination is applied, the performance will be further improved by 3.45%.

Fig. 14(b) compares the energy consumption of various configurations. Over all 6 GPGPU workloads, SLC-STT based shared memory design averagely saves 49.3% total energy compared to SRAM baseline. MLC-STT based shared memory averagely saves only 24.4% total energy compared to SRAM

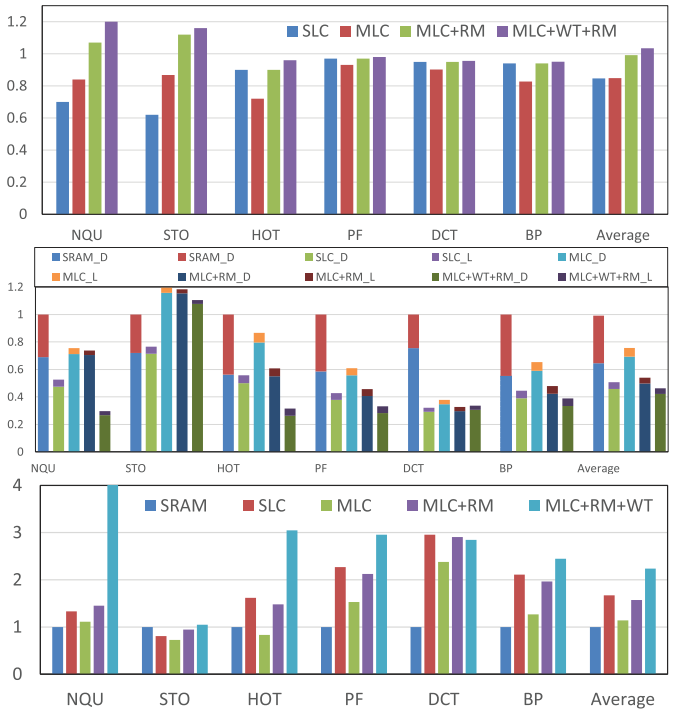

Fig. 14. (a) Performance, (b) Energy consumption, and (c) Normalized energy efficiency of MLC-based shared memory(_D means dynamic energy consumption, _L meas leakage energy consumption).

baseline due to the costly 2-step read/write accesses. Non-optimized MLC implementation introduces significant energy overhead in the workloads that are write intensive: In STO, MLC-based shared memory does not save any energy and indeed consumes 19.5% more energy than the SRAM baseline. When applying remapping algorithm, the power consumption is reduced a lot for HOT, PF, DCT, and BP as they require less than half of the available shared memory capacity and all their requests are mapped into soft-bit rows. The application of write termination further reduces the power consumption of all the benchmarks. NQU, which has the highest ratio of the unchanged-data write accesses, achieves the largest energy reduction (43.7%) among all 6 benchmarks.

As shown in Fig. 14(c), MLC-STT shared memory with remapping strategy and write termination ("MLC+RM+WT") achieves the best energy efficiency in all 6 benchmarks. It is 116% more energy-efficient than SRAM baseline and 48% than the SLC design.

C. Overall Energy and Performance Impacts

We also evaluate the overall performance and power saving of a streaming multiprocessor having both MLC-RF and MLC-SM in the 6 benchmarks. As shown in Fig. 15, on average, the MLC-based streaming multiprocessor design achieves 5.47% performance improvement and 3.2% energy saving across all 6 benchmarks. Positive energy saving is achieved in all the benchmarks compared to SRAM baseline because of the near-zero leakage current of STT-RAM. NQU, STO, and HOT achieve large performance improvement (i.e., 12.1%, 14.7% and 19.5%, respectively) by using MLC implementation. Such

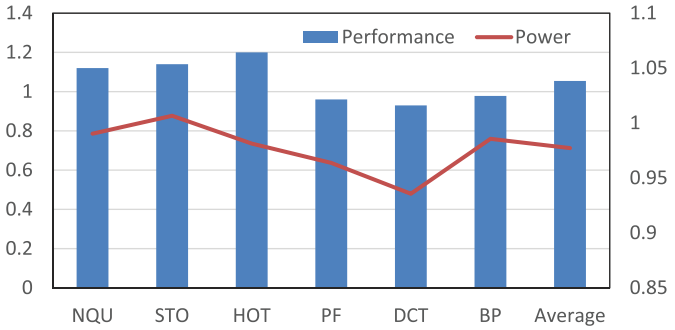

Fig. 15. Overall performance and energy consumption of MLC-based GPU streaming multiprocessor.

improvement comes from the increased parallel thread blocks introduced by the large capacity of MLC-STT. On the contrary, the workloads that do not need large register file or shared memory show even slight performance degradation (i.e., 4.2%, 6.83%, and 2.2%, respectively) due to the long write latency of MLC-STT.

VI. Related Works

MLC-STT was invented to replace SRAM in the implementation of last level cache in general-purpose microprocessors for capacity and energy improvement [17], [21], [33], [34]. To alleviate the impact of the long read/write latency of MLC-STT, line pairing and line swapping schemes were proposed by Jiang *et al.* to optimize the usage of the read-fast-write-slow and read-slow-write-fast regions in parallel MLC-STT [18]. In [19], Bi *et al.* improved the access performance of the cache implemented with serial MLC-STT that is composed of the reversely connected MTJs by grouping the bits of the cache cells into fast and slow entries.

In modern GPU designs, on-chip memory size rapidly grows. Register file quickly takes the place of cache and becomes the largest on-chip memory component to support large volume of parallel threads and fast context switching; the capacity of shared memory has been increased generation by generation to provide enough memory space for data sharing among the threads in one thread block [4], [6], [35], [36]. Many studies have been conducted to enhance the efficacy of register file and shared memory, such as unifying the memory space of register file, shared memory and L1 cache to provide dynamic re-configurable memory size based on application requirement [25], adding an extra cache to reduce the access number of register file [37], and turning off unallocated register file entries by a tri-modal access control unit for leakage power reduction [26].

As the on-chip memory capacity is severely constrained by the large cell area of SRAM, embedded DRAM (eDRAM) [32] and SRAM-DRAM hybrid structure [38] were introduced into the register file design in GPGPUs. Recently, Goswami *et al.* proposed using SLC-STT to implement register file and shared memory in GPUs for power and performance improvements [9]. To mitigate the read disturbance of STT-RAM, Hang *et al.* applied a software-hardware co-designed solution to reduce the performance loss and energy overhead caused by restore operations while maintaining the reliability of

read [39]. Some studies have been focused on addressing the costly write of STT-RAM for GPU register file, such as [40] and [41]. As GPU becomes the primary compute processor for AI and tensor core is added in SM for matrix multiplication running with other math units together in parallel [42], more register file and shared memory capacity are desired. Our MLC-STT-based design further explores the improvement space of performance and power through higher integration density and more flexible tradeoff between access performance and effective storage capacity. We note that MLC-STT is also facing the endurance and read disturbance as SLC, some even more serious. As we focus on mitigating the impact of long latency and high energy of hard bit write in this work, we plan to address the bit-level disturbance and endurance issues at architecture level next, as well as consider other types of MTJ devices have less program current, like perpendicular MTJ [43].

VII. Conclusion

In this work, we propose a MLC-STT register file design to overcome the limited scalability of SRAM-based register file and shared memory in GPU architecture. By leveraging high integration density and non-volatility of MLC-STT, our design dramatically reduces the area and leakage power of GPU register file and shared memory. To overcome the performance degradation due to the slow access to the hard bits of MLC-STT cells, we propose a remapping strategy that allocates frequently-accessed registers into soft-bit rows with faster access speed and less energy consumption whenever it is possible. A bank-status-aware-scheduling scheme is also invented to reorder the warp issuing to minimize the access stalls induced by the long write accesses. Experimental results show that the MLC-STT designs deliver better system performance than that of conventional SRAM baseline while achieving significant area reduction and system energy efficiency improvement.

References

[1] AMD. *Radeon*. Accessed: 2000. [Online]. Available: http://www.amd.com/us/products/notebook/graphics/Pa-ges/notebook-graphics.aspx

[2] Nvidia. *Tegra*. Accessed: 2008. [Online]. Available: http://www.nvidia.com/object/tegra.html

[3] AMD. *Heterogeneous Computing*. Accessed: 2012. [Online]. Available: http://developer.amd.com/resources/he-terogeneous-computing

[4] Nvidia. *Fermi*. Accessed: 2010. [Online]. Available: http://www.nvidia.com/object/fermi-architecture.html

[5] M. Arora, "The architecture and evolution of CPU-GPU systems for general purpose computing," Res. Surv., Dept. Comput. Sci. Eng., Univ. California, San Diego, CA, USA, Tech. Rep., 2012.

[6] Nvidia. *Kepler*. Accessed: 2012. [Online]. Available: https://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf

[7] S. Che *et al.*, "Rodinia: A benchmark suite for heterogeneous computing," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Oct. 2009, pp. 44–54.

[8] W. W. L. Fung, I. Sham, G. Yuan, and T. M. Aamodt, "Dynamic warp formation: Efficient MIMD control flow on SIMD graphics hardware," *ACM Trans. Archit. Code Optim.*, vol. 6, no. 2, pp. 1–37, Jun. 2009.

[9] N. Goswami, B. Cao, and T. Li, "Power-performance co-optimization of throughput core architecture using resistive memory," in *Proc. IEEE 19th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2013, pp. 342–353.

[10] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, Apr. 2009, pp. 163–174.

[11] J. Leng *et al.*, "GPUWattch: Enabling energy optimizations in GPGPUS," in *Proc. 40th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2013, pp. 487–498.

[12] K. L. Spafford, J. S. Meredith, S. Lee, D. Li, P. C. Roth, and J. S. Vetter, "The tradeoffs of fused memory hierarchies in heterogeneous computing architectures," in *Proc. 9th Conf. Comput. Frontiers (CF)*, 2012, pp. 103–112.

[13] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, and W.-M.-W. Hwu, "Optimization principles and application performance evaluation of a multithreaded GPU using CUDA," in *Proc. 13th ACM SIGPLAN Symp. Princ. Pract. Parallel Program. (PPoPP)*, 2008, pp. 73–82.

[14] R. Dorrance, F. Ren, Y. Toriyama, A. A. Hafez, C.-K.-K. Yang, and D. Markovic, "Scalability and design-space analysis of a 1T-1MTJ memory cell for STT-RAMs," *IEEE Trans. Electron Devices*, vol. 59, no. 4, pp. 878–887, Apr. 2012.

[15] Y. Zhang *et al.*, "Compact model of subvolume MTJ and its design application at nanoscale technology nodes," *IEEE Trans. Electron Devices*, vol. 62, no. 6, pp. 2048–2055, Jun. 2015.

[16] Y. Zhang, L. Zhang, W. Wen, G. Sun, and Y. Chen, "Multi-level cell STT-RAM: Is it realistic or just a dream?" in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2012, pp. 526–532.

[17] Y. Chen, W.-F. Wong, H. Li, and C.-K. Koh, "Processor caches built using multi-level spin-transfer torque RAM cells," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design*, Aug. 2011, pp. 73–78.

[18] L. Jiang, B. Zhao, Y. Zhang, and J. Yang, "Constructing large and fast multi-level cell STT-MRAM based cache for embedded processors," in *Proc. 49th Annu. Design Autom. Conf. (DAC)*, 2012, pp. 907–912.

[19] X. Bi, M. Mao, D. Wang, and H. Li, "Unleashing the potential of MLC STT-RAM caches," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2013, pp. 429–436.

[20] M. Hosomi *et al.*, "A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-RAM," in *IEDM Tech. Dig.*, Dec. 2005, pp. 459–462.

[21] T. Ishigaki *et al.*, "A multi-level-cell spin-transfer torque memory with series-stacked magnetotunnel junctions," in *Proc. Symp. VLSI Technol.*, Jun. 2010, pp. 47–48.

[22] Y. Chen *et al.*, "Access scheme of multi-level cell spin-transfer torque random access memory and its optimization," in *Proc. 53rd IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2010, pp. 1109–1112.

[23] Nvidia. *GeForce GTX 480*. Accessed: 2010. [Online]. Available: http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-480/specifications

[24] B. W. Coon, J. E. Lindholm, S. Liu, S. F. Oberman, and M. Y. Siu, "Operand collector architecture," U.S. Patent 7 834 881, Nov. 16, 2010.

[25] M. Gebhart, S. W. Keckler, B. Khailany, R. Krashinsky, and W. J. Dally, "Unifying primary cache, scratch, and register file memories in a throughput processor," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2012, pp. 96–106.

[26] M. Abdel-Majeed and M. Annavaram, "Warped register file: A power efficient register file for GPGPUs," in *Proc. IEEE 19th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2013, pp. 412–423.

[27] P. Shivakumar and N. P. Jouppi, "Cacti 3.0: An integrated cache timing, power, and area model," Compaq Comput. Corp., Palo Alto, CA, USA, Tech. Rep. 2001/2, 2001.

[28] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," *IEEE Trans. Electron Devices*, vol. 53, no. 11, pp. 2816–2823, Nov. 2006.

[29] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 994–1007, Jul. 2012.

[30] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "Energy reduction for STT-RAM using early write termination," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2009, pp. 264–268.

[31] J. A. Stratton *et al.*, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," *Center Reliable High-Perform. Comput.*, vol. 127, pp. 1–12, 2012.

[32] N. Jing *et al.*, "An energy-efficient and scalable eDRAM-based register file architecture for GPGPU," in *Proc. 40th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2013, pp. 344–355.

[33] Y. Chen, W.-F. Wong, H. Li, C.-K. Koh, Y. Zhang, and W. Wen, "On-chip caches built on multilevel spin-transfer torque RAM cells and its optimizations," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 9, no. 2, pp. 1–22, May 2013.

[34] W. Xu, Y. Chen, X. Wang, and T. Zhang, "Improving STT MRAM storage density through smaller-than-worst-case transistor sizing," in *Proc. 46th Annu. Design Autom. Conf. DAC*, 2009, pp. 87–90.
[35] Nvidia. *Maxwell*. Accessed: 2014. [Online]. Available: http://international.download.nvidia.com/geforce-com/international/pdfs/GeForce-GTX-750-Ti-Whitepaper.pdf
[36] Nvidia. *Pascal*. Accessed: 2016. [Online]. Available: http://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf
[37] M. Gebhart, S. W. Keckler, and W. J. Dally, "A compile-time managed multi-level register file hierarchy," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2011, pp. 465–476.
[38] 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 *Proc. 38th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2011, pp. 247–258.
[39] H. Zhang *et al.*, "Shielding STT-RAM based register files on GPUs against read disturbance," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 2, pp. 1–17, Mar. 2017.
[40] J. Wang and Y. Xie, "A write-aware STTRAM-based register file architecture for GPGPU," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 12, no. 1, pp. 1–12, 2015.
[41] H. Zhang, X. Chen, N. Xiao, and F. Liu, "Architecting energy-efficient STT-RAM based register file on GPGPUs via delta compression," in *Proc. 53rd Annu. Design Autom. Conf. (DAC)*, 2016, pp. 1–6.
[42] Nvidia. *Volta*. Accessed: 2017. [Online]. Available: https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf
[43] G. Wang *et al.*, "Compact modeling of perpendicular-magnetic-anisotropy double-barrier magnetic tunnel junction with enhanced thermal stability recording structure," *IEEE Trans. Electron Devices*, vol. 66, no. 5, pp. 2431–2436, May 2019.

Xiaoxiao Liu received the B.S. and M.S. degrees in electrical engineering and software engineering from Beihang University, Beijing, China, in 2005 and 2009, respectively, and the Ph.D. degree from the University of Pittsburgh in 2017. She worked with Dr. Y. Chen on emerging memory and heterogeneous system architecture during her Ph.D. study. She currently is working as a Senior Technical Staff at GPU Architecture Team, AMD.

Mengjie Mao received the B.S. degree in computer science from the South China University of Technology, the M.S. degree in computer science from the University of Science and Technology of China, and the Ph.D. degree from the University of Pittsburgh in 2016. He is currently a Software Engineer with Waymo LLC.

Xiuyuan Bi received the B.E. and M.E. degrees in microelectronics from Tsinghua University, Beijing, China, and the Ph.D. degree in electrical engineering from the University of Pittsburgh, Pittsburgh, PA, USA. He is currently a Principal Design Engineer with Black Sesame Technology. His research interests include emerging nonvolatile memory technology and hardware acceleration for deep neural networks.

Hai (Helen) Li (Fellow, IEEE) received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, and the Ph.D. degree from the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, in 2004. She is currently the Clare Boothe Luce Associate Professor with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA. Before that, she was with Qualcomm Inc., San Diego, CA, USA, Intel Corporation, Santa Clara, CA, Seagate Technology, Bloomington, MN, USA, the Polytechnic Institute of New York University, Brooklyn, NY, USA, and the University of Pittsburgh, Pittsburgh, PA, USA. She has authored or coauthored more than 200 technical papers in peer-reviewed journals and conferences and a book titled *Nonvolatile Memory Design: Magnetic, Resistive, and Phase Change* (CRC Press, 2011). Her current research interests include neuromorphic architecture for brain-inspired computing systems, machine learning and deep neural network, memory design and architecture, and architecture/circuit/device cross-layer optimization for low power and high performance. She is currently a Distinguished Lecturer of the IEEE CAS Society and a Distinguished Speaker of ACM. She is a Distinguished Member of ACM. She was a recipient of the NSF CAREER Award, the DARPA Young Faculty Award (YFA), and the TUM-IAS Hans Fisher Fellowship from Germany. She received seven best paper awards and additional seven best paper nominations from international conferences. She was the General Chair or Technical Program Chair of multiple IEEE/ACM conferences and a technical program committee member of over 30 international conference series. She serves as an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, the IEEE TRANSACTIONS ON MULTI-SCALE COMPUTING SYSTEMS, ACM TECS, *IEEE Consumer Electronics Magazine*, ACM TODAES, and IET-CPS.

Yiran Chen (Fellow, IEEE) received the B.S. and M.S. degrees from Tsinghua University and the Ph.D. degree from Purdue University in 2005. After five years in industry, he joined the University of Pittsburgh in 2010, as an Assistant Professor, and then promoted to Associate Professor with tenure in 2014, held a Bicentennial Alumni Faculty Fellowship. He is currently a Professor with the Department of Electrical and Computer Engineering, Duke University, and serving as the Director of the NSF Industry–University Cooperative Research Center (IUCRC) for Alternative Sustainable and Intelligent Computing (ASIC) and the Co-Director of the Center for Computational Evolutionary Intelligence (CEI), Duke University, focusing on the research of new memory and storage systems, machine learning and neuromorphic computing, and mobile computing systems. He has published one book and more than 350 technical publications and holds 93 U.S. patents. He received six best paper awards and 13 best paper nominations from international conferences. He was a recipient of the NSF CAREER Award and the ACM SIGDA Outstanding New Faculty Award. He is a Distinguished Member of ACM, a Distinguished Lecturer of the IEEE CEDA, and a recipient of the Humboldt Research Fellowship for Experienced Researchers. He serves or served as an associate editor for several IEEE and ACM transactions/journals and served on the technical and organization committees of more than 50 international conferences.