Shielding STT-RAM Based Register files on GPUs Against Read Disturbance

Hang Zhang, Key State Laboratory of High Performance Computing, College of Computer, National Univeristy of Defense Technology, China
Xuhao Chen, College of Computer, National Univeristy of Defense Technology, China
Nong Xiao, Key State Laboratory of High Performance Computing, College of Computer, National Univeristy of Defense Technology & School of Data and Computer Science, Sun Yat-sen University, China
Lei Wang, Fang Liu, Wei Chen, College of Computer, National Univeristy of Defense Technology, China
Zhiguang Chen, Key State Laboratory of High Performance Computing, College of Computer, National Univeristy of Defense Technology, China

To address the high energy consumption issue of SRAM on GPUs, emerging Spin-Transfer Torque (STT-RAM) memory technology has been intensively studied to build GPU register files for better energy-efficiency, thanks to its benefits of low leakage power, high density, and good scalability. However, STT-RAM suffers from the read disturbance issue, which stems from the fact that the voltage difference between read current and write current becomes smaller as technology scales. The read disturbance leads to high error rates for read operations, which cannot be effectively protected by the SEC-DED ECC on large-capacity register files of GPUs.

Prior schemes (e.g. read-restore) to mitigate the read disturbance usually incur either non-trivial performance loss or excessive energy overhead, thus not applicable for the GPU register file design which aims to achieve both high performance and energy-efficiency. To combat the read disturbance, we propose a novel software-hardware co-designed solution, i.e. *Red-Shield*, which consists of three optimizations to overcome the limitations of the existing solutions. First, we identify dead reads at compiling stage and augment instructions to avoid unnecessary restores. Second, we employ a small read buffer to accommodate register reads with high access locality to further reduce restores. Third, we propose an adaptive restore mechanism to selectively pick the suitable restore scheme, according to the busy status of corresponding register banks. Experimental results show that our proposed design can effectively mitigate the performance loss and energy overhead caused by restore operations, while still maintaining the reliability of reads.

CCS Concepts: •**Hardware** → **Spintronics and magnetic technologies; Emerging technologies;**

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: GPU, STT-RAM, Register File, Read Disturbance

ACM Reference Format:
Hang Zhang, Xuhao Chen, Nong Xiao, Lei Wang, Liu Fang, Wei Chen, Zhiguang Chen. Shielding STT-RAM Based Register files on GPUs against Read Disturbance. *ACM J. Emerg. Technol. Comput. Syst.* 10, 5, Article 39 (March 2016), 17 pages.
DOI: 0000001.0000001

This work is supported by National Natural Science Foundation of China, under grant Nos. 61433019, U1435217, 61232003, 61502514, 61202121, 61402503, 61402501, 61120106005 and 61303073; National High Technology Research and Development 863 Program of China, under grant Nos. 2015AA015305; Research Fund for the Doctoral Program of Higher Education of China, under grant Nos. 20114307120013.
Author's addresses: H. Zhang, X. Chen, N. Xiao, L. Wang, F. Liu, W. Chen, Z. Chen, College of Computer, National University of Defense Technology, Changsha, Hunan 410073, Changsha, China.
The Corresponding authors are H. Zhang (zhanghanghit@gmail.com) and N. Xiao (xiao-n@vip.sina.com).

DOI: 0000001.0000001

1. INTRODUCTION

General-purpose graphics processing units (GPGPUs) have been widely used for high performance computing during the last decade, owing to its high throughput and energy-efficiency [Schulte et al. 2015]. Its single instruction multiple thread (SIMT) architecture enables thousands of threads running concurrently to hide long latency of memory operations and hence achieve high throughput. To maintain hardware contexts of a huge number of threads, GPGPUs usually employ a large-capacity register file for seamlessly context switching. Unfortunately, large-capacity register files also consume a large portion of power that usually accounts for 15-20% of total power in modern GPUs [Leng et al. 2013], as the register files are currently built with SRAM technology that has high leakage power in sub-micron technology nodes.

To combat the high energy consumption of SRAM, there have been extensive studies on replacing SRAM with emerging STT-RAM to build energy-efficient register files of GPUs, due to its benefits of low leakage power, high density, and good scalability [Goswami et al. 2013; Li et al. 2015; Liu et al. 2015]. However, `read disturbance` issue has become a major design challenge for STT-RAM based on-chip memory [Sun et al. 2012; Wang et al. 2015] recently. The read disturbance lies in the decreasing difference between write current and read current that are applied to read/write STT-RAM cells as technology scales. Due to performance concerns, GPU register files usually employ simple ECC protections against SRAM errors [Palframan et al. 2014], and it is not sufficient to shield the high error rate caused by the read disturbance issue for STT-RAM based register file [Wang et al. 2015]. Therefore, how to cope with the read disturbance issue is a key concern that decides whether the emerging STT-RAM can be employed to build energy-efficient register files on GPUs, and it has not been addressed well by prior works.

Restoring data after each read can be a simple yet effective solution to suppress the read disturbance on GPU register files [Sun et al. 2012; Wang et al. 2015], which we referred as **RD** (Restoring Data) scheme. However, these restore operations lead to unaccepted performance loss. Fig. 1 shows the performance impact on read disturbance when **RD** is employed to combat read disturbance issue. Here, **Base** represents a SRAM based register file design as a baseline. **WB** denotes a STT-RAM register file design that replaces SRAM with STT-RAM, which is enhanced with a SRAM write buffer as previous design [Liu et al. 2015]. The read disturbance issue is not handled with this design, and hence it results in a high error rate. The detailed configurations are shown in Section 5.

Nearly 27.3% of GPU throughput is reduced under this restore scheme (RD) on average, which means these restore operations lead to significant performance loss that is unacceptable for high-throughput-demand GPUs. Besides, these excessive restores also consume non-trivial dynamic energy. What make things even worse is that the read disturbance issue will even be aggravated by the shrinking difference between read current and write current as technology scales. Consequently, it is imperative to judiciously tackle the read disturbance for STT-RAM based register file while preserving the high throughput and energy-efficiency of GPUs. To this end, we propose a novel software-hardware co-design, namely ***Red-shield*** (***Re***ad ***D***isturbance ***Shield***), to alleviate the performance loss and extra energy consumption incurred by the read disturbance issue on GPUs. In summary, we make the following contributions in this work.

— We devise a compiler assisted dead read identification scheme to identify dead reads that do not need restores, and remove unnecessary restore operations related to these reads.

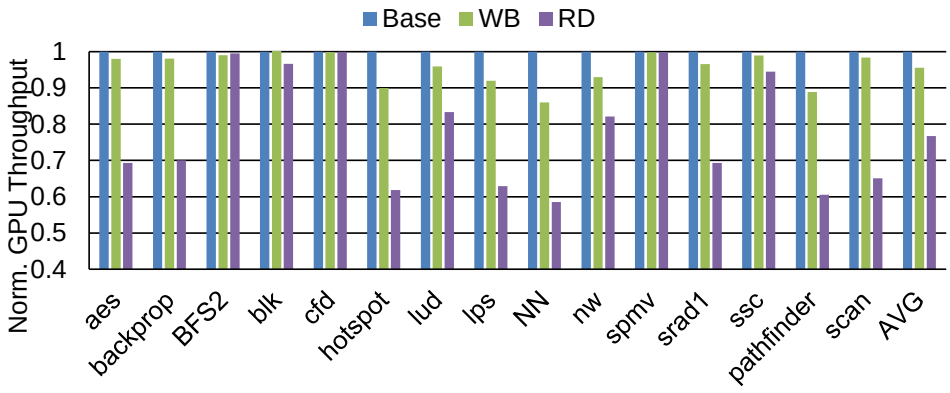

Fig. 1. The GPU performance slowdown under read disturbance with the selective restore scheme.

— Since successive reads to the same register lead to temporal locality, we employ a small SRAM read buffer to accommodate the reads with high locality.
— We propose an adaptive restore mechanism to selectively pick the optimal restore scheme, according to the status of register banks.

To the best of our knowledge, this is the first work to address the read disturbance issue for large-scale STT-RAM based register files on GPUs. The structure of this paper is organized as follows: Section 2 introduces the background of GPU register file architectures and the basics of STT-RAM memory technology. Section 3 demonstrates the read disturbance issue of STT-RAM based register file as the motivation. Section 4 elaborates the detailed design of our proposed schemes. The experimental evaluation and analysis is shown in Section 5. We review the related work on STT-RAM based GPU register file design and the STT-RAM read disturbance issue on Section 6. Then Section 7 gives the conclusion of our work.

2. BACKGROUND

In this section, we briefly introduce the GPU architecture and its register file structure, as well as the basics of STT-RAM memory technology.

2.1. GPGPU Architecture and Register Files

Our baseline GPU architecture is similar to NVIDIA Fermi architecture. It is composed of 16 streaming multiprocessors (SMs), each of which contains a 5-stage pipeline. An SM consists of 32 single structured CUDA cores. All the CUDA cores inside an SM share the same instruction fetch and issue logic.

In CUDA programs, a *kernel* is a grid of parallel thread blocks. Each thread block can have at most 1,024 threads. A warp which contains 32 threads is the minimum scheduling unit in GPUs and executes in a lock-step manner. The overview of the baseline architecture is shown in Fig. 2. Each SM has 32,768 32-bit registers which constitute a total 128KB register file. To enable simultaneous accesses to the register file from multiple warps, the register file is divided into 16 banks, each of which has 8KB capacity. Each bank is partitioned into 64 register entries, and each entry has 1024-bit data width. A register entry can be read/written to fulfill access requests of warps.

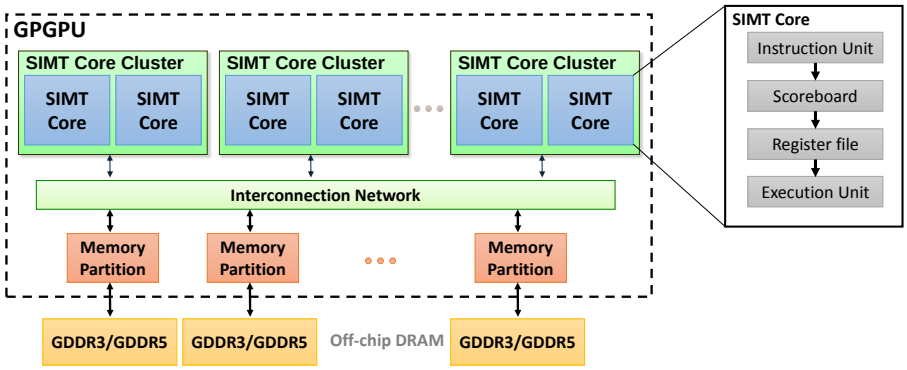

Fig. 2. The GPGPU architecture explored in this work.

2.2. STT-RAM Memory Technology

As technology scales into submicron, the leakage power of transistors becomes dominant for total power consumption, which imposes great challenges of building SRAM based on-chip resources. As a result, researchers start to explore the emerging Emerging non-volatile memory (NVM) technologies, such as STT-RAM (Spin Transfer Torque MRAM), PCM (Phase Change Memory) and ReRAM (Resistive Memory), are widely explored to replace conventional SRAM/DRAM technologies, due to the advantages of near-zero leakage power, high density, good scalability and non-volatility[Zhou et al. 2009; Xu et al. 2015; Sun et al. 2011]. Compared with PCM and ReRAM, STT-RAM has a relatively low read/write latency and high endurance ($> 10^{12}$), and thus is intensively studied as a promising alternative to replace SRAM [Sun et al. 2009; Sun et al. 2011].

STT-RAM cells uses a magnetic tunneling junction (MTJ) structure to store a bit of data. whose accessibility is controlled by an NMOS access transistor. shown in Fig. 3. The MTJ is composed of two ferromagnetic layers separated by a dielectric layer (usually built in MgO). The magnetization of one ferromagnetic layer is fixed (referred as the reference layer), whereas the magnetization of the other is changeable (referred as the free layer). The magnetization of the free layer switches to the other direction when the applied current exceeds the critical threshold by injecting spin polarized electrons. If the free layer is parallel with the reference layer, the MTJ exhibits low resistance,

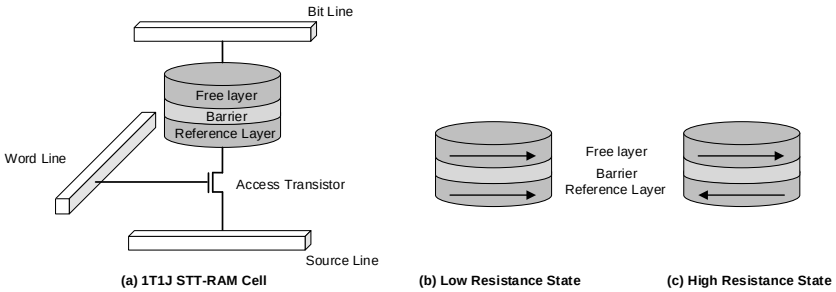

Fig. 3. (a) The 1T1MTJ STT-RAM cell structure. (b) Low resistance state: the magnetization directions of the free layer and the reference layer are parallel. (c) High resistance state: the magnetization directions of the free layer and the reference layer are anti-parallel.

which denotes the logic value as ”0”; If the free layer is anti-parallel with the reference layer, the MTJ exhibits high resistance, which denotes the logic value as ”1”.

Table I illustrates the circuit level parameters comparison between SRAM and STT-RAM at 32nm technology node for 700MHz clock rate. All these parameters are derived from NVsim [Dong et al. 2012] and configured similarly as previous work [Li et al. 2015]. It shows that STT-RAM has much lower leakage power consumption but longer write operation latency and higher write energy consumption than SRAM. The STT-RAM cell also only has nearly 1/3 feature size of SRAM, which results in denser STT-RAM memory arrays. Note that the energy consumption of a write operation can be roughly estimated as the production of write pulse width, current, and supply voltage. It is expected to decrease as the MTJ size scales down.

3. MOTIVATION

In this section, we review the challenges of the read disturbance issue for STT-RAM and analyze its impact on STT-RAM based register file design.

3.1. The Read Disturbance Problem

With good scalability, STT-RAM can even reduce the cell size beyond 22nm with the write current scaling proportionally with MTJ area [Chun et al. 2013]. It can be modeled by Equation 1 [Wang et al. 2015].

$$I_\omega = A * (J_{c0} + \frac{C}{T_\omega^\alpha}) \tag{1}$$

where I_ω is the write current; A denotes the MTJ area; J_{c0} denotes the critical current density at zero temperature; T_ω denotes the write current duration; C and α are fitting parameters.

STT-RAM read operations have the same operating mechanism as write operations but with smaller voltage. Fig. 4 illustrates the amplitude comparison between write current and read current at different technology nodes. For larger technology nodes, e.g. 130nm, the read current is much smaller than the write current, and hence the reads are reliable. While write current scales well as mentioned, read current cannot scale and stays as the same at sub-micron regime, because it is hard to build sense amplifiers that can sense the correct data below $20\mu A$ current [Weisheng Zhao et al. 2009]. When read current is close to write current, the read operation may change

Table I. Parameters of SRAM and STT-RAM

Parameter	SRAM	STT-RAM
Cell Factor (F^2)	146	57.5
Area (mm^2)	0.194	0.038
Read latency (cycle)	1	1
Write latency (cycle)	1	4
Read energy (pJ/bit)	0.203	0.239
Write energy (pJ/bit)	0.191	0.300
Leakage power (mW)	248.7	16.2

Table II. Read Disturbance Error Rate

Tech(nm)	45	32	22	15	11
BER	1.38E-8	3.38E-7	3.07E-6	2.16E-5	1.2E-4
LER	1.42E-5	3.50E-4	3.17E-3	2.21E-2	1.17E-1
LER SEC-DED	1.02E-10	6.12E-8	5.04E-6	2.46E-4	7.11E-3

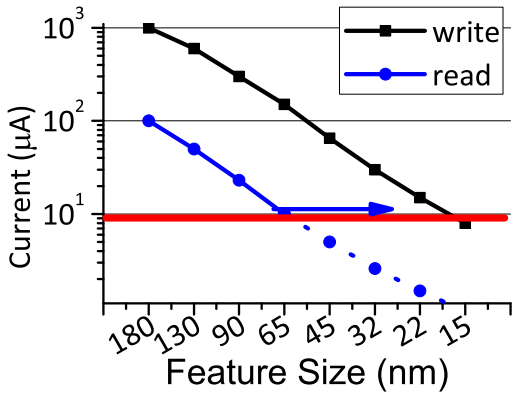

Fig. 4. The scaling of read and write currents [Wang et al. 2015]

the stored value, which is referred as the *read disturbance* for STT-RAM. We can use Equation 2 to model the read disturbance [Wang et al. 2015; Smullen et al. 2011].

$$P = 1 - exp\{-exp[-\frac{t}{\tau}\Delta(1 - \frac{I}{I_{c0}})]\} \tag{2}$$

where P is the read disturbance rate; I denotes the read current; t denotes the read pulse width; τ denotes the inverse of the attempt frequency; Δ_0 denotes the magnetic memorizing energy without any current and magnetic field; I_{c0} denotes the critical switching current at 0K.

3.2. Impact of the Read Disturbance

We use the model from [Wang et al. 2015] to calcuate the read disturbance error rate. The 1024-bit register entry is evaluated as a whole block. Due to performance concerns, GPUs usually employ SEC-DED ECC to protect register files [Rossi et al. 2011]. The modelled error rate for read disturbance on STT-RAM register files is shown in Table II. The BER denotes the raw bit error rate, the LER denotes the line error rate, and the LER SEC-DED denotes the line error rate after adopting SEC-DED ECC. We can see that LER SEC-DED is so high that it cannot guarantee the reliability of reading a register entry even under ECC protection. For instance, the LER SEC-DED at 22mm is about 5.04E-6, which is much greater than 2.5E-11, the acceptable error rate of DRAM [Wang et al. 2015; Schroeder et al. 2009]. This implies that we cannot rely on the SEC-DED ECC code to protect STT-RAM based register files on GPUs, when the read disturbance issue is taken into consideration, because the error rate is so high that we cannot build reliable STT-RAM register files.

Additionally, the state-of-the-art design [Wang et al. 2015] that aims to suppress the read disturbance issue is not applicable to GPU register file design, because it is specifically designed for STT-RAM based LLC (L2 cache) whose features are quite different from those of the register file, for example, it requires help information from the upper-level memory hierarchy (L1 cache), and however there is no such upper-level layer above register files on GPUs. Therefore, it is necessary to devise effective mechanisms to obviate the read disturbance issue on STT-RAM based GPU register files while not incurring much performance and energy overhead.

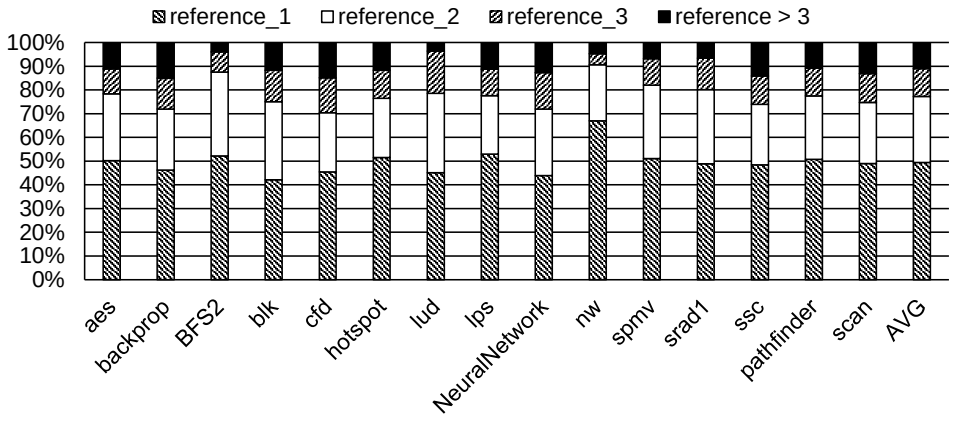

Fig. 5. Register read access count distribution. The number of read accesses to registers is shown.

4. RED-SHIELD DESIGN DETAILS

In this section, we present our proposed design, ***Red-shield***, which incorporates several software and hardware optimizations to address the read disturbance issue of STT-RAM based register files on GPUs.

4.1. Dead Read Identification

To understand the characteristic of accessed register values on GPUs, we conduct profiling on different benchmarks, and the result is shown in Fig. 5. The detailed experiment configuration is shown in Section 5. Most of these benchmarks have similar distributions on how many of the register values are referenced. On average, nearly 49.3% of the generated register values are referenced (read) only once, while nearly 27.9% and 11.6% of them are referenced twice and three times, respectively. Those are referenced more than threes times only constitute 11.2% of total references. A similar result has been observed in previous work [Gebhart et al. 2011]. In fact, it implies that we do not need to restore those register values that will not be re-referenced in the future.

Furthermore, for a register value that will be re-referenced several times, the last reference also does not need to be restored, since there is no more read after the last reference. For instance, if we know the register value A will be referenced three times in total after it is generated, the 3rd read operation on A does not need a restore operation. Here, we refer to the last read operation as the ***dead read***, since the register value is not useful anymore after the last read operation. Apparently, the only read operation to the register value that has only one reference is also a ***dead read***.

If we can identify these dead reads before run-time, then the unnecessary restore operations to these dead reads can be avoided and both performance and energy-efficiency can be improved. To this end, we propose a dead read identification scheme that can effectively recognize dead reads at the compiling stage. The standard register analysis algorithm [Appel 1997] is employed to analyze register liveness coverage, and the pseudo code is shown in Algorithm 1.

Since the PTX specification [NVIDA 2009] only allows four read operands for one instruction, we add four ***dead bits*** into each instruction to represent the liveness of the four operands. The 4-bits are set according to the register liveness analysis, to indicate whether the reads of their corresponding operands are dead or not. At run-time, the dead bits that hold the register liveness information are passed to the register

ALGORITHM 1: Register Analysis Algorithm

for *each n in CFG* **do**
  in[n]=∅, out[n]=∅;
end
repeat
  for *each n in CFG* **do**
    in'[n]=in[n], out'[n]=out[n];
    in[n]=use[n] ∪ (out[n] - def[n]);
    out[n]= $\bigcup_{s \subset succ[n]}$ in[s];
  end
until *in'[n]=in[n] and out'[n]=out[n] for all n*;

file (Fig. 8). Based on this information, the register arbitrator who is responsible for read restoring can make dicision on whether to perform a restore operation or not. Since the register liveness analysis is a standard technique in modern compilers, the overhead is mostly the instruction extension and hence is trivial, which is consistent with the previous work [Jing et al. 2013]. The hardware logic for passing the dead bits is also negligible.

Our scheme can guarantee the correctness of our proposed scheme. We can make sure that we do not wrongly skip the necessary restore operation to a read operations that the read disturbance happens. This correctness of the proposed dead read identification scheme relies on the correctness of the standard algorithm of register liveness analysis. That algorithm aims to identify the liveness of registers, and make two logical registers that do not have overlapped liveness share a same physical register, so as to alleviate the pressure on register capacity. Their scheme can guarantee the correctness that these two register do not have overlapped liveness indeed. By having the correct liveness information of a register, we can easily identify the last read operation to that register as the dead read.

```
                          register number-> 0 1 2 3 4 5 6

1     cvt.u32.u16 $r2, $r0.hi;               R L W L L
2     mul.half.lo.u16 $r5,$r3.lo,$r2.hi;     L L R R L W
3     mul.half.lo.u16 $r6,$r4.lo,$r1.hi;     L R L L R L W
4     mad.wide.u16 $r5,$r3.hi,$r2.lo,$r5;    L L R R L W L
5     mad.wide.u16 $r6,$r4.hi,$r1.lo,$r6;    L R L L R L W
6     shl.u32 $r5,$r5,0x00000010;            L L L L L W L
7     shl.u32 $r6,$r6,0x00000010;            L L L L L L W
8     mad.wide.u16 $r3,$r3.lo,$r2.lo,$r5;    L L R W L R L
```

Fig. 6. A snippet of the PTX source code for the backprop benchmark

4.2. Read Buffer Promotion

Although the dead read identification scheme can effectively reduce the number of unnecessary restore operations, for register values that have multiple references, it is still necessary to restore the data after each read operation (except for the last one). An example of the register liveness analysis result of `backprop` is shown in Fig. 6. 'R' denotes a register read; 'W' denotes a register write; 'L' indicates a register is alive. For instance, r_2 is referenced three times (line 2,4,8) in this snippet code. Though the last read operation can be identified as a dead read, there is still a need to restore the first two read operations after reading.

For the register values that have many references (e.g. more than 3 times), if the reference count information is available, we can devise smart schemes to leverage this temporal locality. To further reduce the number of restore operations, we propose a read buffer promotion scheme to make full use of this observed high temporal locality. To be specific, we build a reference tracking algorithm (Algorithm 2) by leveraging the register liveness analysis. Based on the liveness information, this algorithm collects the reference count for each read operand of each instruction. The reference count for a specific register is re-collected only if the corresponding operand is dead or written. This reference count is also encoded into instructions. We then deploy a small SRAM read buffer to store these register values with high temporal locality (e.g. reference count > 3). The hardware details and how the read buffer at maintain in the register file architecture are shown in Fig. 8.

ALGORITHM 2: Calculate read reference count

```
Allocate and initialize: queue[max_register_num];
for i ⊂ set_instructions do
    for op ⊂ i_read_op do
        if op.dead == false then
            queue[op.register].push(i);
        else
            i_leader = queue[op.register].front();
            i_leader.ref_count[op] = queue[op.register].size();
            while queue[op.register].empty()==false do
                queue[op.register].pop();
            end
        end
    end
end
```

4.3. Adaptive Restore Design

A STT-RAM read operation is similar to a write operation except that the read voltage is smaller than write voltage. To restore a data value in STT-RAM cells in an efficient way, researchers have proposed the selective restore (SR) [Wang et al. 2015] that involves two read operations and one write operation. After first read operation, SR requires another read that employs a current with an opposite direction compared to the first read. This second read operation aims to identify which specific cells have been disturbed. Then a successive write operation can write only the cells that have been destructed at the first read. The SR scheme is shown in Fig. 7(a). The SR can effectively reduce the number of restored data bits, and therefore save energy consumption caused by unnecessary restores. However, since SR requires extra operation and causes non-trivial performance overhead, it can not be naively applied to the STT-RAM based register file design on which performance is more critical.

On the other hand, the direct restore (DR) [Sun et al. 2012] can be also employed in our STT-RAM register file design. The idea is to skip the second read operation and restore all the ”1”s, regardless whether each of these ”1”s is disturbed or not. Thus we can save the time for performing one read operation for each restore. The downside of this scheme is that it might write the ”1”s that are not disturbed and might consume unnecessary energy. However, this downside is not a problem because of two reasons: 1) Our previous dead read identification and read buffer promotion schemes can effectively reduce the number of restores. Thus, the number of required direct restores

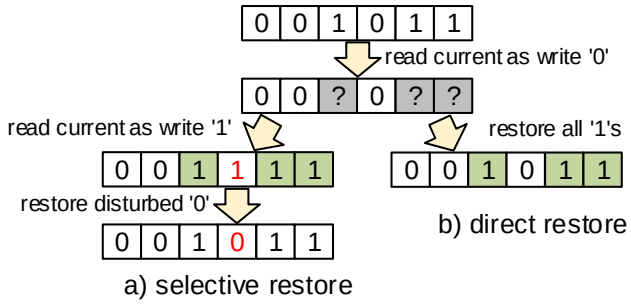

Fig. 7. Comparison between the previous restore operation and our fast-restore operation

are relatively smaller compared to the baseline that employs SR scheme. 2) as technology scales, the read current and write current are getting closer, and therefore the direct restore (under the inversion optimization) only consumes a reasonable amount of energy close to the second read operation in the SR scheme.

Furthermore, we introduce an adaptive restore scheme that combines the benefits of the selective restore and the direct restore together, and can selectively choose to use SR or DR, according to the serving state of register banks. To be specific, the operand collector only performs DR when there is a bank conflict, since in this situation the latency of read operations has a dominant impact on performance; If there is no bank conflict, SR is preferred to minimize the energy consumption of restore operations.

Fig. 8 shows the schematic view of the overall architecture that contains these three techniques we have proposed. When a read instruction enters the GPU pipeline, it has already been tagged with the dead read information and the read reference count, which are generated by the complier as we proposed. When the instruction enters into the accessing register stage, the arbiter determines which read mode (SR or DR) will be used to read the STT-RAM register entry. After the data is fetched from the register file, the arbiter determines whether this data needs to be placed into the read buffer, according to the reference count information.

5. EXPERIMENTAL EVLUATION

In this section, we present the simulation configuration, and evaluate our proposed design in terms of performance and energy. Then we give a brief estimation on software and hardware overheads.

Table III. Simulation 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	GTO

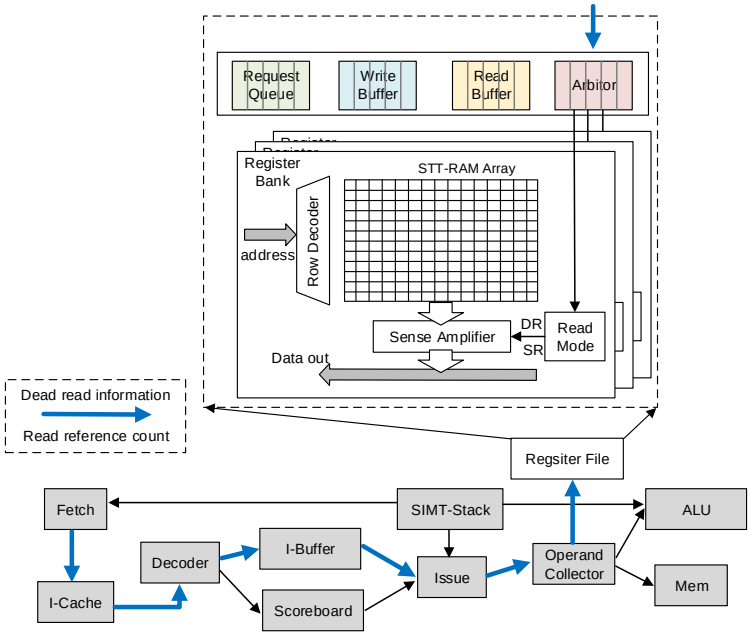

Fig. 8. The overall architecture of our proposed design

5.1. Simulation Configuration

We use a cycle accurate GPGPU simulator, GPGPU-Sim v3.2 [Bakhoda et al. 2009], to model the detailed GPU architecture. We augment the simulator with the STT-RAM hybrid register file, with respect to the circuit level parameters of both SRAM and STT-RAM. Our baseline GPU is configured as NVIDIA Fermi GTX480. The detailed configuration is shown in Table III. We modify the PTX parser to perform dead read identification.

We select 18 representative benchmarks from Rodina [Che et al. 2010], Parboil [Stratton et al. 2012], Mars [Fang et al. 2011] and NVIDIA CUDA SDK [NVIDA 2012] benchmark suites to evaluate our proposed design. These benchmarks cover a wide range of applications with various characteristics, and are shown in Table IV.

We compare different configurations as follows.

— base: the baseline register file built with SRAM.
— STT: the register file design built with STT-RAM.
— WB: the STT-RAM based register file with a SRAM write buffer, configured as in previous work [Liu et al. 2015]. Since there is no read disturbance protection in this scheme, the line error rate is so high that it is not a feasible for GPUs.
— RD: the STT-RAM based register file design with the selective restore scheme [Wang et al. 2015].
— CO: Configured similar with RD, but with the dead read identification technique.
— CORB: Configured similar with CO, but with the read buffer promotion design.
— CORBAR: Configured similar with CORB, but with the adaptive restore scheme.

5.2. Performance and Energy Evaluation

Fig. 9 shows the GPU throughput under different configurations, all normalized to the baseline. It is clear to see that employing the selective restore scheme (RD) to suppress

Table IV. Benchmarks

Benchmark	Description
AES	AES Encryption
BAC	Back Propagation
BFS	Breadth-First Search
CFD	CFD Solver
LAV	lavaMD
LPS	Laplace Solver
LUD	LU Decomposition
MUM	MUMmerGPU
NEU	Neural Network
NN	k-Nearest Neighbors
NW	Needleman-Wunsch
PVC	Page View Count
SRA	Structured Grid
SSC	Similarity Score
PAT	Pathfinder
MON	Monte Carlo
SAD	Sum of Absolute Differences
SCP	ScalarProd

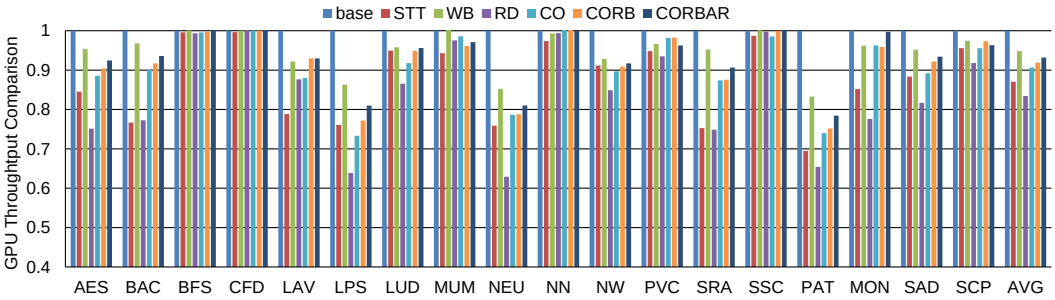

Fig. 9. GPU throughput comparison between different configurations

read disturbance incurs unacceptable performance loss, achieving only 83.4% of the SRAM baseline performance on average. This is because longer read operations (plus restore time) significantly increases the probability of stalling pipelines. Fortunately, our proposed dead read identification scheme (CO) can effectively improve throughput and achieve 90.6% of the baseline performance, due to the capability of avoiding unnecessary restore operations. For some applications, such as BAC, NEU and MON, CO can even alleviate the performance loss to achieve 96.26% of the baseline performance. Moreover, CORB further improves performance to 91.9% of the baseline on average, due to the capability of accommodating read request with multiple accesses. Some benchmarks like LAV, LPS and LUD can significantly benefit from the high temporal locality. Finally, CORBAR achieves 93.1% of the baseline performance on average, since it can adaptively choose when to use DR or SR according to whether the corresponding bank is busy or not. This adaptive feature is especially helpful for benchmarks like LPS, SRA and PAT, which have higher number of bank conflicts.

Fig. 10 shows the register file energy consumption under different configurations. It is not surprising that RD leads to excessive energy consumption overhead, which even offsets the low energy benefit of STT-RAM technology. For some benchmarks,

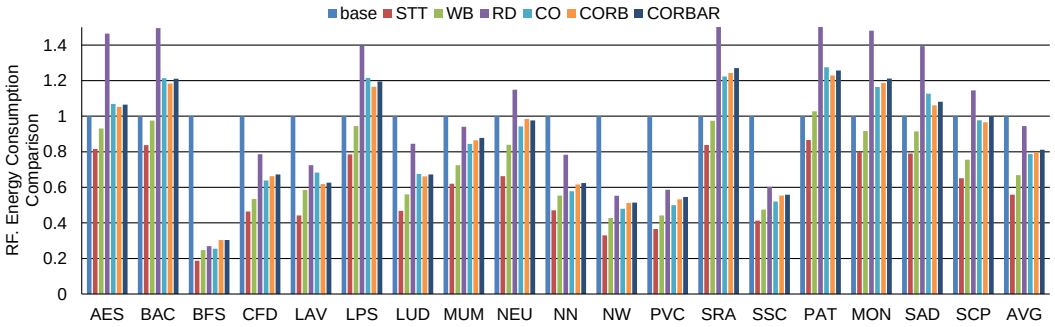

Fig. 10. Energy consumption comparison between different configurations.

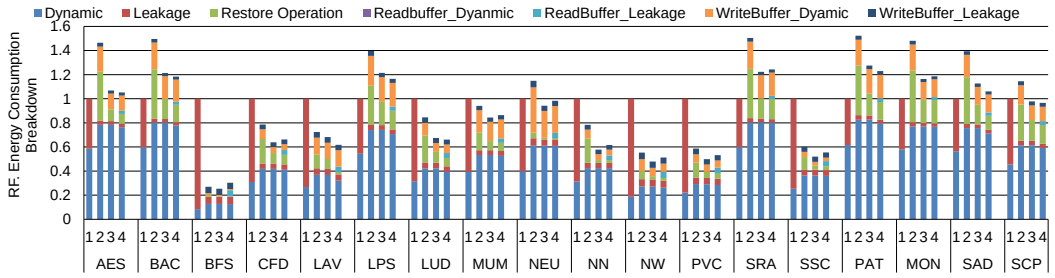

Fig. 11. Energy consumption breakdown under different configurations. (1:base, 2:RD, 3:CO, 4:CORB)

such as LPS, NEU and PAT, high restore energy consumption makes the total energy consumption of STT-RAM 40% greater than the SRAM baseline. Fortunately, the proposed CO scheme can effectively mitigate the restore energy consumption on average and achieve 78.7% energy consumption of the baseline, even under read disturbance. CORB consumes 79.8% of the baseline energy on average, meaning the energy overhead of the small read buffer is trivial. CORBAR consumes 81.2% of the baseline energy on average. Though CORBAR consumes a little bit higher energy than CORB, the energy-efficiency of the register file is not sacrificed thanks to the low leakage power of STT-RAM.

We find an interesting observation. For benchmarks on which leakage energy consumption dominates, such as BFS, CFD, NN and SSC, although the performance almost does not change when read disturbance occurs, the STT-RAM based design achieves high energy-efficiency by reducing leakage energy consumption and our proposed design can further cut off the restore energy consumption. On the other hand, for benchmarks on which dynamic energy consumption dominants, such as AES, BAC and LPS, our proposed design can effectively alleviate performance loss.

To understand the reason why our propose scheme can effectively reduce the energy consumption of restore operations, we breakdown the energy consumption of the register file in Fig. 11. It is clear that employing STT-RAM for the register file design successfully cuts down high leakage energy consumption. However, due to the read disturbance issue, the restore energy contributes a large portion energy consumption. By skiping these unnecessary restore operations, our proposed comprehensive design, *Red-shield*, is able to mitigate the restore energy overhead.

5.3. Overhead Analysis

For the dead read identification, the compiling time is trivial, because it is based on the standard register analysis that has been widely employed for GPUs. Thanks to the small feature size of STT-RAM, the area overhead for building a 128KB STT-RAM register file per SM is only 19.5% of the SRAM based design, according to Table I. We also adopt a SRAM writer buffer as previous work [Li et al. 2015], which is modelled as associative SRAM caches with 8KB capacity using NVsim [Dong et al. 2012], and its area incurs 5.6% hardware overhead. Based on our experiments, a 4KB SRAM read buffer is enough for our design, and its area incurs 3.2% hardware overhead. In summary, the area of our proposed STT-RAM register file design is only 28.3% of the SRAM-based register file design.

6. RELATED WORK

In section, we review the related research work on building STT-RAM based register file on GPUs and techniques that cope with the read disturbance issue for STT-RAM.

6.1. STT-RAM Based Register file on GPUs

Thanks to these advantages of STT-RAM, researchers have explored STT-RAM based register file design on GPUs for high energy-efficiency, thereby alleviating the high leakage power consumption from SRAM.

Goswami *el al.* first propose to build the register fils and shared memory with STT-RAM technology [Goswami et al. 2013]. In order to mitigate the overhead of long write latency of STT-RAM, they re-architect STT-RAM cells with thinner MTJ to trade off retention time for better performance. To further compensate the performance loss of STT-RAM, Li *et al.* propose to design a hybrid register file architecture augmented with two SRAM write buffer and a warp-aware write back strategy [Li et al. 2015]. Liu *et al.* propose to use multi-level cell (MLC) STT-RAM [Liu et al. 2015] to build the register file for GPUs due by leveraging the high density of MLC STT-RAM cells. Wang *et al.* proposes a write-aware STT-RAM based register file design on GPUs, by splitting the bank write and make the arbitrator to distribute nonconflicting reads and writes to the same RF bank to increase the access parallelism [Wang and Xie 2015]. Zhang *et al.* propose to leverage a light-weight compression algorithm to mitigate the STT-RAM writes that are associated with high energy consumption [Zhang et al. 2016].

Although these work have demonstrated efficient designs to mitigate the high power consumption of SRAM, they do not take the read disturbance issue into consideration, and the read distrubance issue can either result in high error rate making computed results incorrect, or easily offset the energy benefits of employing STT-RAM, as we have illustrated. As technology scales, the read disturbance becomes a so critical concern for the STT-RAM based GPU register file design that engineers should carefully cope with.

6.2. Read Disturbance Issue

Li *et al.* discuss the reliability issues on STT-RAM memory technology, including write/read disturbance and write / read failure [Li et al. 2008]. Zhang *et al.* demonstrate that the read current must scale accordingly to keep the disturbance on the MTJ resistance state at a minimum level and keep the safe read margin, otherwise read disturbance errors shall manifest at small technology nodes [Zhang et al. 2012]. A recent chip demonstration [Takemura et al. 2010] (45nm) of STT-MRAM treats reads as destructive and restores the original data after each read. Sun *et al.* proposes two write modes for read disturbance and choose the suitable one according to the capability of error tolerance: One performs restore-after-read to ensure data integrity if the application

is sensitive to high error rate, while the other does not if having high error tolerance [Sun et al. 2012]. Wang *et al.* postpones the restore operation to a read till the time that this line is evicted from the upper level cache L1. Based on the status of the line at the eviction time, they selectively restore the disturbed cells to achieve energy efficiency [Wang et al. 2015]. As we have analyzed, these already proposed techniques either incur high performance overhead or high energy consumption overhead if they are directly applied to the GPU architecture without careful optimization. On the other hand, our scheme leverages the GPU architecture feature to further eliminate these unnecessary restore operations, including the register liveness feature, of which prior work fail to makes good use.

7. CONCLUSION

STT-RAM has been explored as a promising alternative for SRAM to build the large-capacity register file on GPUs due to the advantage of high energy-efficiency. However, the read disturbance issue of STT-RAM imposes great challenges for register file design as technology further scales. To address the read disturbance issue, we present a novel design, *Red-shield*. It employs complier optimization to filter out short-lifetime reads so as to mitigate the performance and energy overhead incurred by the read-restore operations. Coupled with the read buffer promotion design as well as the optimized read-restore scheme, *Red-shield* can effectively alleviate pipeline stalls caused by read disturbance and improve the energy-efficiency. In total, our design can promote STT-RAM based GPU register files to become a feasible and effective solution for future GPU architectures.

ACKNOWLEDGMENTS

We are grateful to our anonymous reviewers for their suggestions to improve this paper.

REFERENCES

Andrew W Appel. 1997. *Modern compiler implementation in C*. Cambridge university press.

Ali Bakhoda, George L. Yuan, Wilson W L Fung, Henry Wong, and T. M. Aamodt. 2009. Analyzing CUDA workloads using a detailed GPU simulator. In *Proceedings of the International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 163–174. DOI: http://dx.doi.org/10.1109/ISPASS.2009.4919648

Shuai Che, Jeremy W. Sheaffer, Michael Boyer, Lukasz G. Szafaryn, Liang Wang, and Kevin Skadron. 2010. A characterization of the Rodinia benchmark suite with comparison to contemporary CMP workloads. In *Proceedings of IEEE International Symposium on Workload Characterization (IISWC)*. DOI: http://dx.doi.org/10.1109/IISWC.2010.5650274

Ki Chul Chun, Hui Zhao, J D Harms, Tae-Hyoung Kim, Jian-Ping Wang, and C H Kim. 2013. A Scaling Roadmap and Performance Evaluation of In-Plane and Perpendicular MTJ Based STT-MRAMs for High-Density Cache Memory. *IEEE Journal of Solid-State Circuits* 48, 2 (2013), 598–610. DOI: http://dx.doi.org/10.1109/JSSC.2012.2224256

Xiangyu Dong, Cong Xu, Yuan Xie, and Norman P. Jouppi. 2012. NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 7 (2012), 994–1007. DOI: http://dx.doi.org/10.1109/TCAD.2012.2185930

Wenbin Fang, Bingsheng He, Qiong Luo, and Naga K. Govindaraju. 2011. Mars: Accelerating MapReduce with graphics processors. *IEEE Transactions on Parallel and Distributed Systems* 22, 4 (2011), 608–620. DOI: http://dx.doi.org/10.1109/TPDS.2010.158

Mark Gebhart, Stephen W. Keckler, and William J. Dally. 2011. A compile-time managed multi-level register file hierarchy. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. DOI: http://dx.doi.org/10.1145/2155620.2155675

Nilanjan Goswami, Bingyi Cao, and Tao Li. 2013. Power-performance co-optimization of throughput core architecture using resistive memory. In *Proceedings of the IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. DOI: http://dx.doi.org/10.1109/HPCA.2013.6522331

Naifeng Jing, Yao Shen, Yao Lu, Shrikanth Ganapathy, Zhigang Mao, Minyi Guo, Ramon Canal, and Xiaoyao Liang. 2013. An energy-efficient and scalable eDRAM-based register file architecture for GPGPU. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA)*. DOI:http://dx.doi.org/10.1145/2508148.2485952

Jingwen Leng, Tayler Hetherington, Ahmed ElTantawy, Syed Gilani, Nam Sung Kim, Tor M Aamodt, and Vijay Janapa Reddi. 2013. GPUWattch: enabling energy optimizations in GPGPUs. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA)*. 487. DOI:http://dx.doi.org/10.1145/2485922.2485964

Gushu Li, Xiaoming Chen, Guangyu Sun, Henry Hoffmann, Yongpan Liu, Yu Wang, and Huazhong Yang. 2015. A STT-RAM-based low-power hybrid register file for GPGPUs. In *Proceedings of the 52nd Annual Design Automation Conference (DAC)*. 1–6. DOI:http://dx.doi.org/10.1145/2744769.2744785

Jing Li, Haixin Liu, Sayeef Salahuddin, and Kaushik Roy. 2008. Variation-tolerant Spin-Torque Transfer (STT) MRAM array for yield enhancement. In *Proceedings of the Custom Integrated Circuits Conference*. 193–196. DOI:http://dx.doi.org/10.1109/CICC.2008.4672056

Xiaoxiao Liu, Mengjie Mao, Xiuyuan Bi, Hai Li, and Yiran Chen. 2015. An efficient STT-RAM-based register file in GPU architectures. In *Proceedings of the 20th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 490–495. DOI:http://dx.doi.org/10.1109/ASPDAC.2015.7059054

NVIDA. 2009. NVIDIA next generation CUDA compute architecture: Fermi. (2009). DOI:http://dx.doi.org/10.1016/j.immuni.2005.11.006

NVIDA. 2012. GPU Computing SDK. (2012). https://developer.nvidia.com

David J. Palframan, Nam Sung Kim, and Mikko H. Lipasti. 2014. Precision-aware soft error protection for GPUs. In *The IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*. 49–59. DOI:http://dx.doi.org/10.1109/HPCA.2014.6835966

Daniele Rossi, Nicola Timoncini, Michael Spica, and Cecilia Metra. 2011. Error Correcting Code Analysis for Cache Memory High Reliability and Performance. In *Design, Automation, and Test in Europe (DATE)*. 1–6. DOI:http://dx.doi.org/10.1109/DATE.2011.5763257

Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. 2009. DRAM errors in the wild: a large-scale field study. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems (SIGMETRICS)*. DOI:http://dx.doi.org/10.1145/1555349.1555372

Michael J. Schulte, Mike Ignatowski, Gabriel H. Loh, Bradford M. Beckmann, William C. Brantley, Sudhanva Gurumurthi, Nuwan Jayasena, Indrani Paul, Steven K. Reinhardt, and Gregory Rodgers. 2015. Achieving Exascale Capabilities through Heterogeneous Computing. *IEEE Micro* 35, 4 (jul 2015), 26–36. DOI:http://dx.doi.org/10.1109/MM.2015.71

Clinton W. Smullen, Anurag Nigam, Sudhanva Gurumurthi, and Mircea R. Stan. 2011. The STeTSiMS STT-RAM simulation and modeling system. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD*. DOI:http://dx.doi.org/10.1109/ICCAD.2011.6105348

John A Stratton, Christopher Rodrigues, I-Jui Sung, Nady Obeid, Li-Wen Chang, Nasser Anssari, Geng Daniel Liu, and Wen-Mei W Hwu. 2012. Parboil: A revised benchmark suite for scientific and commercial throughput computing. *IMPACT Technical Report* (2012).

Guangyu Sun, Xiangyu Dong, Yuan Xie, Jian Li, and Yiran Chen. 2009. A novel architecture of the 3D stacked MRAM L2 Cache for CMPs. In *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)*. 239–249. DOI:http://dx.doi.org/10.1109/HPCA.2009.4798259

Zhenyu Sun, Xiuyuan Bi, Hai Helen Li, Weng-Fai Wong, Zhong-Liang Ong, Xiaochun Zhu, and Wenqing Wu. 2011. Multi retention level STT-RAM cache designs with a dynamic refresh scheme. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 329–338. DOI:http://dx.doi.org/10.1145/2155620.2155659

Zhenyu Sun, Hai Li, and Wenqing Wu. 2012. A dual-mode architecture for fast-switching STT-RAM. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design (ISLPED)*. 45–50. DOI:http://dx.doi.org/10.1145/2333660.2333673

R. Takemura, T. Kawahara, K. Ono, K. Miura, H. Matsuoka, and H. Ohno. 2010. Highly-scalable disruptive reading and restoring scheme for Gb-scale SPRAM and beyond. In *Proceedings of the IEEE International Memory Workshop (IMW)*. DOI:http://dx.doi.org/10.1016/j.sse.2010.11.032

Jue Wang and Yuan Xie. 2015. A Write-Aware STTRAM-Based Register File Architecture for GPGPU. *ACM Journal on Emerging Technologies in Computing Systems* 12, 1 (2015), 1–12. DOI:http://dx.doi.org/10.1145/2700230

Rujia Wang, Lei Jiang, Youtao Zhang, Linzhang Wang, and Jun Yang. 2015. Selective restore: an energy efficient read disturbance mitigation scheme for future STT-MRAM. In *Proceedings of the 52nd Annual Design Automation Conference (DAC)*. 1–6. DOI:http://dx.doi.org/10.1145/2744769.2744908

Weisheng Zhao, C. Chappert, V. Javerliac, and J.-P. Noziere. 2009. High Speed, High Stability and Low Power Sensing Amplifier for MTJ/CMOS Hybrid Logic Circuits. *IEEE Transactions on Magnetics* 45, 10 (oct 2009), 3784–3787. DOI:http://dx.doi.org/10.1109/TMAG.2009.2024325

Cong Xu, Dimin Niu, Naveen Muralimanohar, Rajeev Balasubramonian, Tao Zhang, Shimeng Yu, and Yuan Xie. 2015. Overcoming the challenges of crossbar resistive memory architectures. In *Proceedings of the IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. 476–488. DOI:http://dx.doi.org/10.1109/HPCA.2015.7056056

Hang Zhang, Xuhao Chen, Nong Xiao, and Fang Liu. 2016. Architecting energy-efficient STT-RAM based register file on GPGPUs via delta compression. In *Proceedings of the 53rd Annual Design Automation Conference (DAC)*. ACM Press, New York, New York, USA. DOI:http://dx.doi.org/10.1145/2897937.2897989

Yaojun Zhang, Wujie Wen, and Yiran Chen. 2012. The prospect of STT-RAM scaling from readability perspective. *IEEE Transactions on Magnetics* 48, 11 (2012), 3035–3038. DOI:http://dx.doi.org/10.1109/TMAG.2012.2203589

Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. 2009. A durable and energy efficient main memory using phase change memory technology. In *Proceedings of the 36th annual international symposium on Computer architecture (ISCA)*. 14–23. DOI:http://dx.doi.org/10.1145/1555815.1555759