Architecting Energy-efficient STT-RAM Based Register File on GPGPUs via Delta Compression

Hang Zhang[†‡], Xuhao Chen[‡], Nong Xiao[†‡*], Fang Liu[‡]
[†]State Key Laboratory of High Performance Computing, College of Computer, National University of Defense Technology, China
[‡]College of Computer, National University of Defense Technology, China
[*]School of Data and Computer Science, Sun Yat-sen University, China
{hangzhang, chenxuhao, nongxiao, liufang}@nudt.edu.cn

ABSTRACT

To facilitate efficient context switches, GPUs usually employ a large-capacity register file to accommodate a massive amount of context information. However, the large register file introduces high power consumption, owing to high leakage power SRAM cells. Emerging non-volatile STT-RAM memory has recently been studied as a potential replacement to alleviate the leakage challenge when constructing register files on GPUs. Unfortunately, due to the long write latency and high energy consumption associated with write operations in STT-RAM, simply replacing SRAM with STT-RAM for register files would incur non-trivial performance overhead and only bring marginal energy benefits.

In this paper, we propose to optimize STT-RAM based GPU register files for better energy-efficiency and performance via two techniques. First, we employ a light-weight compression framework with awareness of register value similarity. It is coupled with a group-based write driver control to mitigate the high energy overhead caused by STT-RAM writes. Second, to address the long write latency overhead of STT-RAM, we propose a centralized SRAM-based write buffer design to efficiently absorb STT-RAM writes with better buffer utilization, rather than the conventional design with distributed per-bank based write buffers. The experimental results show that our STT-RAM based register file design consumes only 37.4% energy over the SRAM baseline, while incurring only negligible performance degradation.

1 Introduction

General-purpose graphics processing units (GPGPUs) have been used for high performance computing during last decades, owing to its high throughput and energy-efficiency [1]. Its single instruction multiple thread (SIMT) architecture enables thousands of threads running concurrently to hide memory operations with long latency. To maintain hardware contexts of a huge number of threads, GPGPUs usually employ a large-capacity register file for seamlessly context switching. Typically, this size of register files on GPUs is much larger than that on CPUs. For instance, the NVIDIA Tesla K80 GPU employs 512KB 32-bit registers for each streaming multiprocessors (SM) [2], while each core of Intel Haswell architecture only has 336 registers [3].

DAC '16, June 05-09, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2897937.2897989

However, such large-capacity register file consumes huge power which usually accounts for 15-20% power consumption of the total power in modern GPUs [4]. This is because register files are currently built with the SRAM technology that has high leakage power in sub-micron technology nodes. Furthermore, the leakage power of SRAM keeps increasing as technology scales in the future [5][6]. Meanwhile, the size of GPU register files is expected to be continually increased as scaling up the number of co-running threads, to provide higher throughput for emerging applications, such as big data and deep learning applications [7][8]. Consequently, it is important to explore new design solutions for building energy-efficient regiser files for future GPUs.

Recently, an emerging non-volatile memory technology, STT-RAM (Spin Transfer Torque MRAM), has been explored to be a substitution for SRAM as on-chip storage (such as L1 and last level cache on CMP platforms) [9][10], due to its lower leakage power, higher density and better scalability. Prior works have studied the benefits of using single level cell (SLC) STT-RAM [11][12][13] or multi-level cell (MLC) STT-RAM [14] to build the register file for GPUs. Although STT-RAM has been shown to be able to reduce leakage power consumption, naively adopting STT-RAM to construct register files on GPUs suffers from performance degradation and low energy-efficiency, because STT-RAM exhibits higher write latency compared to SRAM [11][12][14] and the high dynamic write power consumption of STT-RAM may offset the benefits of lower leakage power of STT-RAM.

Despite the fact that prior works have shown initial benefits of designing GPU register file with STT-RAM, they have not explored effective optimizations to further reduce the high dynamic energy consumption incurred by employing STT-RAM. Such optimizations are necessary because the dynamic energy consumption starts to dominate when adopting STT-RAM on register files, which are frequently accessed. In addition, the write buffer also needs to be carefully designed, otherwise the large size of SRAM may countervail the benefit of employing STT-RAM. Consequently, we propose several techniques in this paper to optimize the energy consumption of STT-RAM based register file design.

The contributions are summarized as follows:

- We present a lightweight compression framework with the awareness of the register value similarity to reduce the dynamic energy consumption of STT-RAM based register file. A novel group-based write driver control is proposed along with the compression to enable effective energy saving.
- We propose a centralized write buffer design instead of previously proposed distributed per-bank design, to improve the

utilization of the SRAM write buffer and therefore mitigate the performance degradation due to the long latency overhead of STT-RAM write operations.

- We implement our design in a cycle-accurate GPGPU simulator. The evaluation results show that our proposed STT-RAM register file design consumes only 37.4% energy compared to the baseline SRAM design with negligible performance loss.

2 Preliminary

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

2.1 GPGPU Architecture and Register Files

Our baseline GPU architecture is similar to NVidia's Fermi GTX480, which is composed of 16 streaming multiprocessors (SMs). Each SM 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.

In each SM there are 32,768 32-bit registers. These registers 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 an access request of a warp.

2.2 STT-RAM Memory Technology

Non-volatile memory technologies, such as STT-RAM (Spin Transfer Torque MRAM), PCM (Phase Change Memory) and ReRAM (Resistive Memory), are widely explored to replace existing SRAM or DRAM technologies, due to their advantages of near-zero leakage power, high density, good scalability and non-volatility. 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 [15, 16].

To store a bit of data, an STT-RAM cell uses a magnetic tunneling junction (MTJ) structure. 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, 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 1 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 [17] and is configured similarly as prior work [12]. It shows that STT-RAM has much lower leakage power consumption but longer write operation latency and higher write energy consumption than that of SRAM. 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.

Table 1: 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

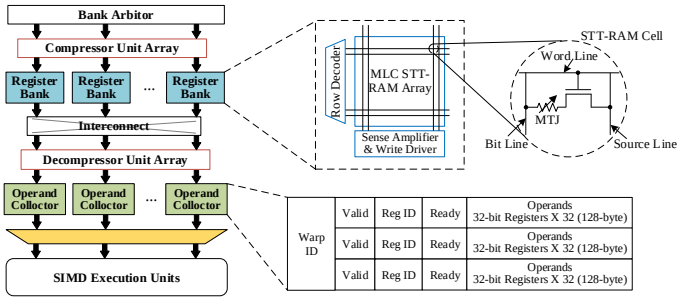

Figure 1: The STT-RAM based register file design.

3 Energy-efficient Register File Design

With much lower leakage power, STT-RAM is therefore studied to replace SRAM for building on-chip storage. To overcome the limitations (long write latency and high write energy) of STT-RAM, we propose propose several techniques to optimize the STT-RAM based register file on GPUs.

3.1 Design Overview

We build the 128KB register file with STT-RAM cells and also keep the banked structure. Thanks to the high density, the total area of the STT-RAM array is only 19.5% of the SRAM array as shown in Table 1. The saved area can be utilized for further optimization.

The architecture of our proposed STT-RAM based register file is shown in Fig. 1. Each of the register bank has a 8KB capacity. There are 64 register entries in each bank. Each register entry consists of 1024 STT-RAM cells, which contribute to 1024 data bits, and serves 32 read/writes of 32-bit registers. Therefore, each register bank contains 2048 32-bit registers in total. Note that our design is different from the distributed register file design [12] where a register access request is serviced by 32 register banks, which inevitably increases the possibility of bank conflicts.

Despite the advantages of low leakage and high density, STT-RAM still has two downsides compared with SRAM. First, the dynamic power consumption of STT-RAM writes is higher than that of SRAM. Second, STT-RAM has relatively long write latency. Such long write latency increases the probability of postponing successive register reads from obtaining data from a register bank that is busy on servicing a write request. Given the drawbacks, it is imperative to devise architectural optimizations to alleviate the overheads, while preserving the advantages of STT-RAM.

3.2 Value Similarity Aware Compression

Recent works have explored the value similarity that exists in the SIMT execution model on GPUs [3, 18]. As mentioned, threads of a warp execute the same instruction in the lock-step way, but may access 32 different registers or memory locations. Typically the data accessed by the 32 threads are likely to have little difference with each other due to the SIMT model. Motivated by this observation, we propose to employ compression techniques to optimize the dynamic write energy consumptions of the STT-RAM

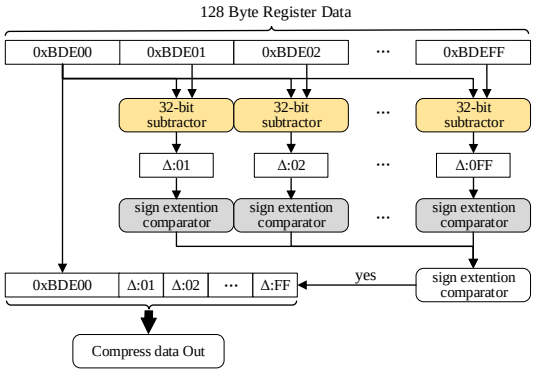

Figure 2: The compress process example on BDI compression.

based register file by leveraging the value similarity of GPUs.

The basic idea is to reduce the number of writing bytes of a register entry requested from a single warp via compression. Without compression, writing 32 32-bit registers from one warp is usually consolidated into a total 128-byte data. The 128-byte data is then formed into one register write request for updating one register entry in the target register bank. If this 128-byte data can be compressed as fewer data bytes, e.g. 40-byte, then we can save the energy of writing 128-40=68 bytes. In this case, more than half of the consumed dynamic energy is reduced. The higher the compression ratio is, the more energy can be saved via compression.

Since the access latency of the register file plays a vital role for overall GPU throughput, only light-weight compression algorithms are considered for the STT-RAM based register file. Complicated compression algorithms that involve high compress/decompress latency are not suitable for our design. In this work, we employ the base-delta-intermediate (BDI) compression algorithm by virtue of its low compress/decompress latency and high compression ratio [3, 19]. However, the compression does not take effect without fine-grained access control. To enable effective energy saving, we propose a *group-based write driver control* on STT-RAM memory arrays, which will be elaborated later.

To perform compression, BDI algorithm uses a data representation of a base and delta values against the base to present the compressed data. If the size of BDI data representation is smaller than the original data, the original data is compressed as a BDI representation. If the size is unchanged or larger, the original data is left uncompressed. In BDI algorithm, the original data is first divided into smaller chunks. The first chunk is usually selected as the base value. The difference between the *base* and each of other chunks is computed by subtraction, and stored as the *delta*. If there exists value similarity between chunks, the size of computed delta is small, and storing the delta requires less storage capacity than storing the original chunk. The original BDI compression iterates the compressing process to figure out the optimal length of the base and delta, ranging from 2, 4, to 8 byte. Since the compression latency is critical to the performance of the register file, only three fixed representations of the compressed data is allowed [3] to mitigate the overhead of the original BDI. Otherwise, the register value is just left as uncompressed. Since both compression and decompression operations only involve simple subtraction and addition operations, compression latency or decompression latency incur negligible overhead (details in Section 4). The architectural design of BDI compression algorithm engine is shown in Fig. 2.

Although the data can be compressed, the dynamic energy consumption is not reduced directly, because 1024 columns are accessed and sensed altogether in the STT-RAM array, after the corresponding wordline is activated. In order to write fewer bytes when the register write request can be compressed, the structure of a STT-RAM array should also be redesigned to allow a fine-grained control on accessing STT-RAM data arrays. Therefore, we introduce the group-based write driver control.

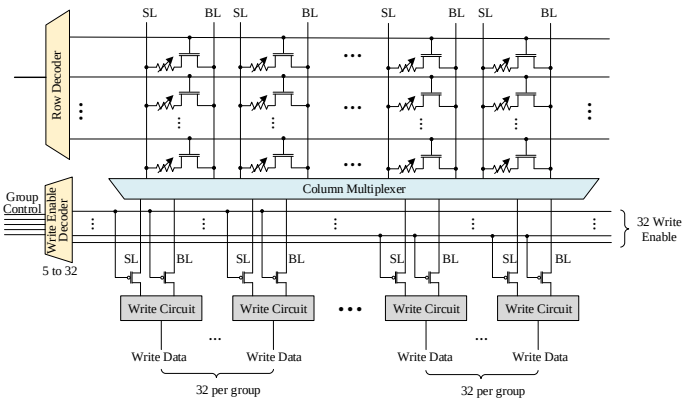

Figure 3: The group-based write driver control design.

Group-Based Write Driver Control: Choosing the width of a register entry for a register bank is important, because it determines the number of banks that needs to be accessed when servicing one register request from a warp. Generally, the more banks a register request needs to access, the higher possibility of bank conflicts will be. For STT-RAM based register file design, high bank conflicts can lead to longer delays for the operator collector to receive requested register values. This long delay may even result in pipeline stalls, thereby degrading the performance. In this work, different from prior work [3][12], we choose a wider register entry of 1024-bits for our proposed STT-RAM based register file design to minimize the possibility of bank conflicts and thus achieve better performance.

We redesign the write driver circuit to enable fine-grained control on how many columns to be written for compressed register entries under the BDI compression framework. The redesigned driver circuity is shown in Fig. 3. To reduce overheads of routing wires, the control granularity is managed in a group manner, instead of using a per-column granularity. For a 1024-bit register entry, all the 1024 write drivers from the STT-RAM array is divided into 32 groups, and each group contains 32 columns.

The 32-group control signals are generated from the compression engine, depending on the data size of the compressed register entry. Then the control signals are fed with a 5-to-32 *Write Enable Decoder* as the write enable signals. Each write enable signal controls whether the 32 pairs of the source line and the bitline in the corresponding group need to be activated or not. If the register entry to be written can be compressed, then only a small portion of the group select lines that indicates the valid compressed data are enabled. By doing this, the dynamic energy consumption of writing an STT-RAM register entry is saved. Note that the group-based write driver control design incurs negligible hardware overhead, since only 64 transistors and a 5-32 decoder are added for the write driver circuit.

3.3 Centralized Write Buffer

The long write latency is another major concern when designing the STT-RAM based register file on GPUs, given the register file is tightly coupled with the SM pipeline. Any extra delays incurred by STT-RAM writes may postpone register read requests and even cause the pipeline stall.

Table 2: The Retention and Write Latencies for STT-RAM

Retention Time	10 years	10 ms
Write Latency @ 700MHz	4 cycle	2 cycle

Prior works choose to design the write buffer in a per-bank manner. That is each STT-RAM bank is equipped with a dedicated write buffer to absorb the write requests. When a write request is sent to a targeting bank that is busy with a previous write, the write buffer temporarily holds this pending write request. Such a write buffer design can mitigate the performance loss when a register read request is delayed by the long latency write.

However, such per-bank design cannot make full use of write buffers when write traffics are unbalanced among register file banks. For instance, when the write buffer of Bank A is fully occupied by pending write requests, the pipeline stalls if another write request arrives at Bank A. In this situation, even though there are still available write buffers at Bank B, Bank A cannot use that. One possible solution is to increase the number of write buffer entries for each register bank as the prior strategy [12]. However, more high leakage SRAM buffers introduce higher area and energy overhead and thus may offset the benefits of employing STT-RAM.

To address this issue, we propose a simple yet effective write buffer design for the hybrid register file on GPGPUs. In this design, the write buffer is constructed in a centralized way, and is deployed at the arbiter and shared by all the register banks together. Each of the register bank can access all the entries of this centralized write buffers. By doing this, the utilization of write buffers is effectively improved even under the same capacity as before. This design is more effective especially when the write traffic is unbalanced.

Since the register value stored in the write buffer is finally written into the register file, we also compress the register value before storing them into the write buffer. This can also reduces the number of writing bits of data so as to save the dynamic energy consumption of writing SRAM. In this way, the compress/decompress overheads are only paid once, while the energy consumption of the write buffer is optimized. To minimize access contention from different warps to this centralized buffer, the write buffer is also architected in a bank-based structure, while the total capacity of the write buffer is still shared by all the register banks.

Additionally, we leverage the technique of relaxing the retention time of STT-RAM cells [11][20] to further mitigate this effect of long write latency of STT-RAM. By making the thickness of the free layer thinner, we reconstruct the STT-RAM cell with retention time of 10 ms, which corresponds to 7000000 cycles on GPUs and is far more than 789 cycles that is reported as the average inter-access distance to a register [5].

4 Evaluation

In this section, we present the simulation configuration, and evaluate our proposed design in terms of performance, energy and hardware overhead.

4.1 Simulation Configuration

We use a cycle-accurate simulator GPGPU-Sim [21] to model the detailed GPU architecture. We augment the simulator with the STT-RAM register file design, 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 3. The power and energy parameters for compress/decompress unit and wires are shown in Table 4, which is derived from previous work [3].

Table 3: 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
Compression Latency	2 cycle
Decompression Latency	1 cycle

Table 4: Estimated energy and power values

Parameter	Value
Operating Voltage (V)	1.0
Wire Capacitance (fF/mm)	300
Wire Energy (32-bit, pJ/mm)	2.6
Compression unit energy/activation (pJ)	23
Compression unit leakage power (mW)	0.12
Decompression unit energy/activation (pJ)	21
Decompression unit leakage power (mW)	0.08

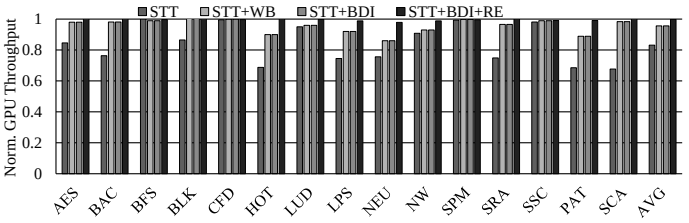

Figure 4: The system throughput normalized to the baseline.

We select 15 representative benchmarks from Rodina [22], Parboil [23] and NVIDIA CUDA SDK [24] benchmark suites to evaluate our proposed design. These benchmarks cover a wide range of applications with various characteristics, including AES Encryption (AES), Back Propagation (BAK), Breadth-First Search (BFS), Blackscholes (BLK), CFD Solver (CFD), Hotspot (HOT), LU Decomposition (LUD), Laplace Solver (LPS), Neural Network (NEU), Needleman-Wunsch (NW), Sparse-Matrix Vector Product (SPM), Srad (SRA), Similarity Score (SSC), Pathfinder (PAT) and Scan (SCA).

We compare the experimental results for various configurations as follows:

- *Base*: the conventional SRAM based register file design.
- *SRAM+BDI*: the SRAM based register file design with BDI compression as in the previous work [3].
- *STT*: the STT-RAM based register file design without the write buffer.
- *STT+WB (Write Buffer)*: the STT-RAM based register file design with the centralized write buffer, configured as similar as in the previous work [12].
- *STT+BDI*: the optimized STT-RAM based register file with BDI compression technique and the write buffer, enhanced with the group-based write driver control.
- *STT+BDI+RE*: the same as BDI configuration, but with the retention time of STT-RAM cells relaxed.

4.2 Performance Impact

Fig. 4 demonstrates the normalized GPU system throughput for different configurations. It shows that directly building the register file with STT-RAM incurs significant performance overhead, which loses 17% performance on average compared to the baseline. Some benchmarks, such as HOT and SCA, even suffer a significant

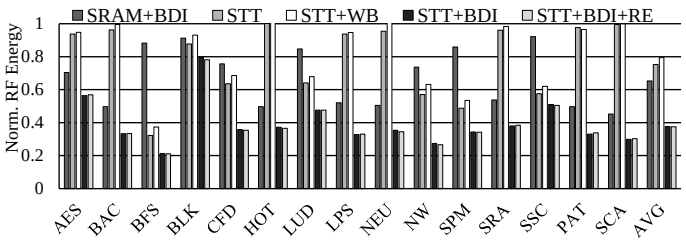

Figure 5: The register file energy consumption normalized to the baseline.

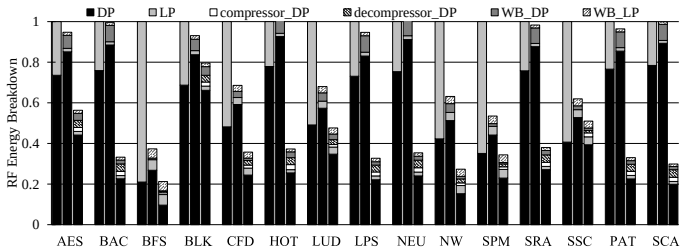

Figure 6: The energy breakdown of the register file for different configurations: Bar 1 is Base, Bar 2 is WB, Bar 3 is STT+BDI.

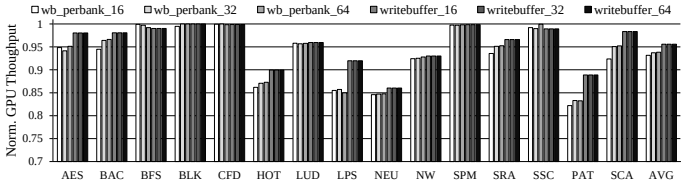

Figure 7: The GPU throughput comparison for different write buffer configurations normalized to the baseline.

slowdown of more than 30%. However, our proposed centralized write buffer can effectively mitigate the performance loss, achieving nearly 95% performance of the baseline on average. The difference of centralized and distributed write buffer design will be discussed in Fig. 7. Meanwhile, BDI compression does not incur significant performance degradation, due to its lightweight algorithmic property, though we increase the pipeline depth to accommodate compress/decompress operations, which is consistent with previous observations [3]. Finally, after applying the relaxed retention scheme, the STT-RAM based register file can achieve nearly 99% performance of the baseline. Combining all the optimizing techniques together makes the STT-RAM based design competitive with the SRAM based design in terms of performance.

4.3 Energy Saving

The energy consumption of register files consists of static and dynamic energy consumption. Fig. 5 compares energy consumption for different configurations, normalized to the SRAM baseline. It shows that directly constructing the register file with STT-RAM can save 24.8% energy consumption, since STT-RAM can effectively reduce the static energy consumption, despite its non-trivial performance degradation as shown in Fig. 4. Although the centralized (SRAM) write buffer design has substantial benefits for performance, it also incurs extra energy consumption, leading to 79.3% energy consumption of the baseline on average. As for the BDI compression technique, it can significantly reduce the total energy consumption of the register file, consuming only 37.4% energy compared to the baseline on average. This is because the BDI compression technique can effectively reduce the number of written bytes to the STT-RAM based register file, when the compression ratio of the written register entries is high. Note that for the STT-RAM based register file, the dynamic energy consumption dominates the total energy consumption, due to the high per-bit energy consumption of STT-RAM. By reducing the number of bits that are written, the BDI compression design can effectively decrease the dynamic energy consumption of the STT-RAM based design. Adopting the BDI compression to the SRAM based register file design can also bring down energy consumption as in previous work [3], achieving 65.2% energy consumption of the SRAM baseline. However, it is not as effective as our proposed design, since 1) STT-RAM has lower leakage energy consumption. 2) STT-RAM has higher per-bit dynamic energy consumption, and hence BDI can effectively cut off this high dynamic energy consumption of STT-RAM based design.

4.4 Energy Consumption Breakdown

To understand the detailed reason why our design is effective for reducing the dynamic energy consumption, Fig. 6 shows the energy consumption breakdown for different configurations, with a few interesting observations. DP denotes dynamic power consumption, whereas LP denotes leakage power consumptions. First, the ratio between the dynamic energy consumption and the static energy consumption for different applications differ at a wide variety. Some benchmarks have high ratios of static energy consumption, such as BFS, NW and SSC, whereas some other have high ratios of dynamic energy consumption, such as AES, CFD and LPS. For applications with high static power consumption, such as BFS and SPM, STT-RAM is able to effectively decrease the total energy consumption by reducing the static energy consumption. However, for applications with high dynamic power consumption, the STT-RAM design is less effective to reduce the total energy. It even increases the dynamic energy consumption in some cases due to the high per-bit dynamic access energy.

In contrast, the BDI compression technique can effectively lower the dynamic power consumption of the STT-RAM based design on applications that are dynamic energy consumption dominated, such as LPS and SCA. Overall, our design has the capability to mitigate the increasing dynamic energy consumption incurred by employing STT-RAM memory, and does not incur significant performance loss. Note that the compressor and decompressor only consume negligible dynamic energy and static energy, due to their simple circuit design.

4.5 Centralized Write Buffer Evaluation

The performance evaluation of our proposed centralized write buffer is shown in Fig. 7. The wb_perbank_XX denotes the write buffer constructed in a per-bank manner with total XX entries, while write-buffer_XX denotes the write buffer constructed in a centralized way with total XX entries. For per-bank configurations, the performance is correlated to the total write buffer entries, and the 16-entry configuration always has the best performance. This is because fewer per-bank write buffer entry leads to higher contention of the corresponding register bank. For the centralized write buffer design, the performance remains nearly unchanged, thanks to the sharing of write buffer entries among register banks.

4.6 Compression Ratio Analysis

The effectiveness of the BDI compression technique on reducing the total energy consumption depends on the compression ratio for register values. The compression ratio for different benchmarks are show in Fig. 8. Most of the benchmarks have shown relatively high

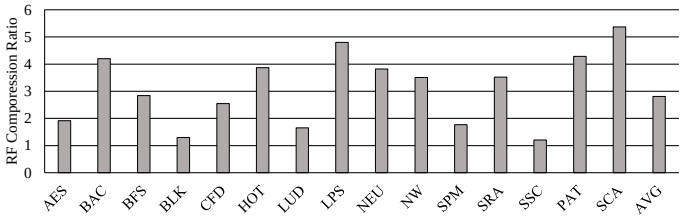

Figure 8: The compression ratio comparison for different applications.

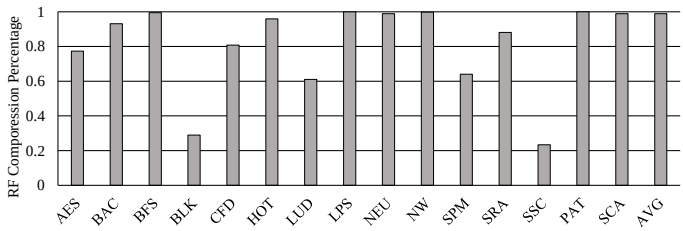

Figure 9: The percentage of the register entries that can be compressed by BDI compression.

compression ratios, and the average compression ratio is around 2.81. As mentioned, the high compression ratio stems from the value similarity that exists in the SIMT execution model of GPUs', which has also been observed by previous work [3, 18]. To get the overall understanding of the application characteristics, we also evaluate the existence of value similarities in details. The percentage of the written register entries that can be compressed over the total written register entires is shown in Fig. 9. All the benchmarks have shown the existence of value similarity. For some benchmarks, such as BFS and LPS, it is surprising to see that nearly all the register written entries can be compressed.

4.7 Hardware Area Analysis

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 1, thanks to the small feature size of STT-RAM. For the centralized write buffer design, 16 buffer entries are enough to avoid pipeline stalls incurred by the long write latency of STT-RAM from our experimental results. The writer buffer is modelled as 8-way associative SRAM cache with 8KB capacity using NVsim [17], and its area incurs 5.6% hardware overhead. We use the compressor and decompressor design in previous work [3], and their total area is $0.07mm^2$, which incurs 36.0% hardware overhead. In summary, the area of our proposed STT-RAM register file is only 61.2% of the SRAM-based register file design.

5 Conclusion

As technology scales, the high leakage power of SRAM hampers the deployment of larger size register files for future GPUs. STT-RAM is a promising technology to replace SRAM, but needs careful consideration to mitigate its write latency and high write energy issue. In this work we propose an STT-RAM based register file design to improve energy efficiency for GPUs via compression. To overcome the high write energy issue of STT-RAM, the light-weight BDI compression technique is employed to reduce dynamic energy consumption of the register file. Meanwhile, a centralized write buffer design is proposed to mitigate the performance gap caused by the long write latency. Experimental results show that our proposed design is able to substantially reduce energy consumption and area of the register file while preserving the performance. This work demonstrates that the STT-RAM based register file design is a promising solution for future GPU architectures.

6 Acknowledgements

We are grateful to our anonymous reviewers for their suggestions to improve this paper. This work is supported by National Natural Science Foundation of China, under grant Nos. 61433019, U1435217, 61232003, 61502514, 61402503, 61402501, 61120106005 and 61303073; National High Technology Research and Development 863 Program of China, under grant 2015AA015305.

References

[1] "Top500 Supercomputers List," 2015. [Online]. Available: http://www.top500.org/lists/2015/06/

[2] NVIDIA, "NVIDIA Kepler GK110/GK210 Architecture Whitepaper."

[3] S. Lee, K. Kim, G. Koo, H. Jeon, W. W. Ro, and M. Annavaram, "Warped-compression: enabling power efficient GPUs through register compression," in *ISCA*, 2015.

[4] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt *et al.*, "GPUWattch: enabling energy optimizations in GPGPUs," in *ISCA*, 2013.

[5] M. Abdel-Majeed and M. Annavaram, "Warped register file: A power efficient register file for GPGPUs," in *HPCA*, 2013.

[6] 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 *ISCA*. New York, New York, USA: ACM Press, 2011.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances In Neural Information Processing Systems*, pp. 1–9, 2012.

[8] K. Wang, K. Zhang, Y. Yuan, S. Ma, R. Lee, X. Ding *et al.*, "Concurrent Analytical Query Processing with GPUs," in *VLDB*, 2014.

[9] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM L2 Cache for CMPs," in *HPCA*, 2009.

[10] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient STT-RAM caches," in *HPCA*, feb 2011.

[11] N. Goswami, B. Cao, and T. Li, "Power-performance co-optimization of throughput core architecture using resistive memory," in *HPCA*, 2013.

[12] G. Li, X. Chen, G. Sun, H. Hoffmann, Y. Liu, Y. Wang *et al.*, "A STT-RAM-based low-power hybrid register file for GPGPUs," in *DAC*, 2015.

[13] J. Wang and Y. Xie, "A Write-Aware STTRAM-Based Register File Architecture for GPGPU," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 12, no. 1, pp. 1–12, aug 2015.

[14] X. Liu, M. Mao, X. Bi, H. Li, and Y. Chen, "An efficient STT-RAM-based register file in GPU architectures," in *ASP-DAC*, 2015.

[15] J. Zhou, X. J. Yang, J. J. Wu, X. Zhu, X. D. Fang, and D. Huang, "A memristor-based architecture combining memory and image processing," *Science China Information Sciences*, vol. 57, no. 5, pp. 1–12, 2014.

[16] L. Wang, J. L. Xue, and X. J. Yang, "Acyclic orientation graph coloring for software-managed memory allocation," *Science China Information Sciences*, vol. 57, no. 9, pp. 1–18, sep 2014.

[17] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, 2012.

[18] J. Kim, C. Torng, S. Srinath, D. Lockhart, and C. Batten, "Microarchitectural mechanisms to exploit value structure in SIMT architectures," in *ISCA*, 2013.

[19] G. Pekhimenko, V. Seshadri, O. Mutlu, M. Kozuch, P. Gibbons, and T. Mowry, "Base-delta-immediate compression: Practical data compression for on-chip caches," in *PACT*, 2012.

[20] A. Jog, A. K. Mishra, C. Xu, Y. Xie, V. Narayanan, R. Iyer *et al.*, "Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs," in *DAC*, 2012.

[21] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *ISPASS*, 2009.

[22] S. Che, J. W. Sheaffer, M. Boyer, L. G. Szafaryn, L. Wang, and K. Skadron, "A characterization of the Rodinia benchmark suite with comparison to contemporary CMP workloads," in *IISWC*, 2010.

[23] J. A. Stratton, C. Rodrigues, I.-J. Sung, N. Obeid, L.-W. Chang, N. Anssari *et al.*, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," *IMPACT Technical Report*, 2012.

[24] NVIDA, "GPU Computing SDK." [Online]. Available: https://developer.nvidia.com