ABSTRACT

In this paper, having investigated the behavior of GPGPU applications, we present an efficient L2 cache architecture for GPUs based on STT-RAM technology. With the increase of processing cores count, larger on-chip memories are required. Due to its high density and low power characteristics, STT-RAM technology can be utilized in GPUs where numerous cores leave a limited area for on-chip memory banks. They have however two important issues, high energy and latency of write operations, that have to be addressed. Low data retention time STT-RAMs can reduce the energy and delay of write operations. However, employing STT-RAMs with lower retention time in GPUs requires a thorough investigation on the behavior of GPGPU applications based on which the STT-RAM based L2 cache is architected. The STT-RAM L2 cache architecture proposed in this paper, can improve IPC by more than 100% (16% on average) while reducing the average consumed power by 20% compared to a conventional L2 cache architecture with equal on-chip area.

Categories and Subject Descriptors
C.1.4 [Processor Architectures]: Multiple Data Stream Architectures (Multiprocessors); B.7.1 [Types and Design Styles]: Memory technologies

General Terms
Design, Performance, Power

Keywords
GPU, GPGPU Application, STT-RAM, Retention Time

1. INTRODUCTION

During last decade, Graphic Processor Units (GPUs) with hundreds of simple cores have become one of the main stream technologies in many-core high-performance systems. GPUs are composed of multiple Stream Multiprocessors (SMs) as well as a rich memory hierarchy including register files, shared memories and cache arrays. Unlike conventional graphics co-processors, General Purpose GPUs (GPGPUs) use SMs as vector processors to achieve a massive Thread-Level Parallelism (TLP). Furthermore, modern GPUs form logical groups (named warps) of parallel threads belonging to the same instruction pack and schedule a number of warps for interleaved execution on a SM; this can lead to better memory performance and relax the problem of branch divergence.

While warp size is rather fixed through successive GPU generations (due to the overhead of branch divergence for larger warps), the number of warps assigned to one SM has been steadily increasing in recent products (e.g. 64 warps per SM in Kepler compared to 24 in G80 [1]). Studies on GPU’s cache hierarchy reveal that most L1 data cache hits are resulted from intra-warp locality [11]. Hence, for a fixed L1 cache size of 32KB, the gained performance is maximized when a specific number of warps are assigned to each SM (i.e. any less or more warps than such a specific number of warps per SM will result in lower performance). Also, increasing the number of warps necessitates a larger L1 cache to sustain performance. Designers are also handicapped for increasing L1 cache size, therefore the pressure is put on L2 cache (next level cache). So, the L2 cache size has to be increased when the number of warps per SM is increased (e.g. L2 cache size increased from 786KB in Fermi to 1536KB in Kepler, both having maximum 48KB L1 D-cache). In recent GPUs, the SRAM L2 cache size kept pace with this capacity demand. With current chips’ maximum power budget being used, further increase of the number of warps (requiring further growth of L2 cache size) will be very difficult. This is also true for increasing the number of cores in future GPUs.

Entering deep nanometer technology era where leakage current increases by 10 times per technology node, SRAM arrays confront serious scalability and power limitations [2]. Among the known technologies to replace SRAMs is Spin Torque Transfer RAM (STT-RAM) [5]. The STT-RAM cell has near zero leakage power, and is about 4× denser than the SRAM cell. The promising characteristics of STT-RAM (especially its higher density and lower power) make it a good candidate for building the L2 cache of future GPUs. Construction of the STT-RAM cell requires deriving a large current for a long time period when writing into it. So, realization of such a promising non-volatile memory structure can face problems with memory access latency and energy.

As our analysis for a wide range of GPGPU applications show, there is a large variation of write intensity to different blocks of the L2 cache. In addition, Write Working Set (WWS) of the L2 cache has two specific features; First, WWS of typical applications within a specific time period is small and second, rewrite interval time of the blocks in WWS is usually lower than 100μs. Consequently, the large data retention of 10 years in STT-RAM is not required for most of the blocks in WWS. This gives us the opportunity of device level optimization for GPUs. It is also shown that physical and magnetization properties of a STT-RAM cell brings a tradeoff between device non-volatility and latency/energy of...
write operations [12]. So, there is an opportunity for device level optimization of STT-RAM L2 caches of modern GPUs.

Based on the above observation, we propose a STT-RAM L2 cache architecture with two portions: a small-sized low-retention array (LR) and a large-sized high-retention array (HR). The LR cache is responsible for holding temporal WWS of the application. The HR cache, on the other hand, provides a high-capacity array holding a large portion of L1-touched data which are most expected to be read-only or less-frequently written. To increase the STT-RAM cache efficiency, the proposed architecture has 2 key differences with the the traditional on-chip GPU cache: 1) a monitoring logic determines write-intensive data blocks forming temporal WWS of the running applications. The main features of this logic are its low overhead and fast reaction to the changes in WWS; 2) regarding cache access types and the elapsed time of the data rewrites in LR cache, the cache search mechanism is modified for better energy efficiency. For write efficiency, we use a simple buffering logic to take care of data expiration in LR cache and postpone refresh of data blocks to the last cycles of retention period.

2. BACKGROUND

GPU and its Memory Hierarchy. A GPU application comprises of one or more kernels. Each kernel launches a hierarchy of threads (a grid of blocks of warps of scalar threads executing same kernel) on Stream Processors (SP) which are clustered in SMs. The thread blocks are allocated as a single unit of work to a SM and warps are the smallest fetched units within this hierarchy. A thread commonly accesses local, shared and global data through a rich memory hierarchy shown in Figure 1-a. The local data is usually placed in a dense register file array which is private to each SM. A SM is also associated with a software-managed local memory for shared data accesses by threads within a block. Global data refers to the data shared among all threads within a grid. When a thread requests data from off-chip main memory, the accesses pass through a two-level cache hierarchy. The L1 caches are private to SMs with no coherency among them. The L2 cache is a banked cache array that is shared by all SMs and use write back policy with respect to main memory. Each L2 bank communicates with L1 caches of different cores through an interconnection network. As L1 caches are not coherent, any write operation of a global data is handled as a write-allocate (on miss) or write-no-allocate (on miss). Local data of a thread are written into the associated L1 cache following a write-back mechanism (figure 1-b). A more complete description of GPU memory system is available in [3].

The L1 cache and shared memory in NVIDIA GPUs are reconfigurable and have a total (L1-Shared memory) capacity of 64KB per SM which can be configured as (16KB-48KB), (32KB-32KB) or (48KB-16KB). This gives a flexibility to programmers to set cache and shared memory sizes based on the requirements of non-shared and shared data, respectively. A GPU also has constant and texture caches for specific data that are backed by L2. At the highest level of hierarchy, each L2 bank has a point-to-point connection with an off-chip DRAM by a dedicated memory controller.

Spin-Torque Transfer RAM. STT-RAM technology is a scalable Magnetic Random Access Memory (MRAM). The basic structure of an STT-RAM cell is composed of a standard NMOS access transistor and a Magnetic Tunnel Junction (MTJ) as the information carrier (Figure 2).

MTJ consists of 2 ferromagnetic layers and an oxide barrier layer (Figure 2). One of the ferromagnetic layers (i.e. the reference layer) has a fixed magnetic direction while the magnetic direction of the other layer (i.e. the free layer) can be changed by directly passing a spin-polarized current. If these 2 ferromagnetic layers are on anti-parallel (parallel) directions, the MTJ resistance is high (low), indicating a logical ‘1’ (‘0’) state. Compared to the conventional MRAM, STT-RAM exhibits superior scalability since the threshold current required to make status changes decreases as the MTJ size shrinks. More details on STT-RAM and the model used in this paper are in [14].

3. RELATED WORK

The body of knowledge on STT-RAM memory technology as well as GPU is extensive and reviewing the state-of-the-art is beyond the scope of this work. Interested reader is referred to [5, 16]. Here, we focus on the most important studies alleviating cache hierarchy shortcomings in GPU by using STT-RAM.

Al Maashri et al. [9] focus on using STT-RAM technology for 3D stacking of texture memory. They do not really address the use of GPUs for general purpose high-performance computing and consider graphics applications. In a recent research by Goswami et al. [6], a single low retention time STT-RAM is used for the register file, constant cache and texture cache, and a hybrid memory (STT-RAM and SRAM) is employed for the shared memory of GPUs. To remedy the high energy consumption of write operations, early write termination mechanism [17] with a higher granularity...
is used. In this paper, we concentrate on studying trade-offs between STT-RAM retention time and write latency/energy and propose an architecture to surpass the cache shortcomings using this promising feature in GPU’s last level cache.

4. GPGPU APPLICATION CHARACTERIZATION

In this section, we characterize GPGPU applications and then use these results to figure out the proper STT-RAM L2 structure. GPGPU applications have different write patterns on cache blocks. In [15], the coefficient of variance (COV) parameter was used to indicate inter and intra cache set write variations in CMP applications. Figure 3 shows inter and intra cache writes COV for GPGPU workloads. Applications like myocyte, bfs and backprop have more writes on some cache blocks and other applications such as stencil, cfd and LIB have even writes over L2 blocks. This justifies having a STT-RAM region which favors write operations using a reduced retention time of STT-RAM cells.

Two-Part L2 cache. Reducing the retention time of STT-RAM cells increases the error rate because of early data bit collapse (with respect to high-retention cells). Table 1 shows different magnetization stability height (Δ) and their corresponding retention times (R.T), write operation energy (W.E) and latency due to the decrement in the current which is needed for writing, but more refreshing is required to prevent data loss. This leads us to using a non-uniform STT-RAM cache structure with different retention times. Therefore, we propose 2-parted L2 cache with different data retention times to manage write operation energy and data refreshment problems by migrating rewritten data blocks from high-retention (HR) part to low retention (LR) part and vise versa.

Maximizing Utilization of L2 Cache Parts. The maximum performance and power improvement is obtained by fully utilizing the LR part hosting the frequently-written blocks; so, it is necessary to determine a write threshold on HR part above which the corresponding block is moved to the LR part. The threshold value is so important as unnecessary migrations of data blocks may cause performance and power overheads. Figure 4 reports the effect of write thresholds on the LR/HR writes ratio and total number of write ratio with respect to TH1 case (the architecture using a threshold value of 1) for different GPGPU workloads. As can be seen in the figure (down), decreasing the threshold value results in better LR utilization while not imposing noticeable write overhead, so the threshold 1 is a justified value as most GPGPU applications are divided into grids which run sequential; each grid uses the results of the previous grid, hence private data of the previous grid is not needed when running the current grid. In other words grids have a small amount of writes happening usually at the end of their execution and before the next grid starts running.

As ideally, we want the LR part to keep WWS of the workload and the HR part to keep read-only blocks, the best write threshold is determined by a fully-associative LR structure (where any evicted block from HR part can move into any block in the LR part). Figure 5 shows the effect of associativity degree on the write utilization of LR part with respect to a fully-associative LR structure. It is obvious that a fully-associative structure provides the highest utilization for LR part, but it is not an appropriate architecture for large caches imposing more complexity. On the other hand, direct-mapped structures have the least complexity, but the least utilization, too. Using a set-associative structure could have the advantages of both fully-associative and direct-mapped structures in a balance. Thus, the LR and HR parts of our cache with different retention times and sizes may have different associativities to gain more utilization with a proper data migration policy between them. Although, different Associativity degrees do not have considerable difference in average but there is some gaps between associativity degree 2 and 1 for some applications like backprop, bfs and mri-gridding, so we use associativity degree of 2 for the LR cache part and explain it in more details throughout the next section.

L2 cache Retention Time. The retention time of the LR part is reduced in order to reduce its write energy and latency, but its refreshing is a new issue to be addressed. Error prevention or data recovery are the common solutions which are not applicable here because of numerous bit collapses when a cache block with low retention time expires. To address this issue, the intervals between successive writes in the LR part are monitored and then used to set the retention time of LR blocks. Figure 6 shows the distribu-

![Figure 3: Inter and intra set write variations](image)

**Table 1**: STT-RAM parameters for different data retention times.

<table>
<thead>
<tr>
<th>Δ</th>
<th>R.T</th>
<th>W.L(ns)</th>
<th>W.E(nJ)</th>
<th>Refreshing (block)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40.29</td>
<td>10 year</td>
<td>10</td>
<td>1.158</td>
<td>0</td>
</tr>
<tr>
<td>17.50</td>
<td>40ms</td>
<td>2</td>
<td>0.161</td>
<td>1867</td>
</tr>
<tr>
<td>11.51</td>
<td>100µs</td>
<td>0.8</td>
<td>0.070</td>
<td>2750755</td>
</tr>
</tbody>
</table>

![Figure 4: HR write threshold analysis](image)
tions of rewrite intervals. Most of data blocks in LR part are rewritten within 100µs as we expected that frequently-written blocks would be migrated to the LR part. To minimize power and latency of write operations, a similar analysis is performed for HR part to set its retention time. Our analysis shows that a 40ms retention time for HR part can handle more than 90% of cache blocks writes.

5. PROPOSED ARCHITECTURE

In this section, we describe our proposed L2 architecture for replacing SRAM L2 with STT-RAM L2 in GPUs. As justified in previous section, we need to have two parallel cache parts, LR and HR, to keep more frequently and less frequently written blocks, respectively.

Figure 7 depicts the block diagram of the proposed STT-RAM L2 cache architecture. Two parallel structures with different retention times and swap buffers are employed. Write counter (WC) is a saturating counter for determining whether a data block is a frequently-written block to migrate it to LR. However, based on our observation on write thresholds, a write threshold of 1 suffices our design) is negligible. After analyses, we carefully determined the buffer size to hold 20 cache lines and found RCs in LR side is due to the lower retention time of blocks in LR part which requires a more accurate refresh mechanism. On buffer full, dirty lines are forced to be written back in main memory, in order to avoid data loss. Fortunately, this is rare and worst-case happens for mri-q with 1% write-back overhead.

As we have two parallel cache arrays at L2, different approaches can be considered in order to handle cache accesses. Two possible approaches include parallel and sequential searches to the LR and HR parts. Parallel searches in LR and HR parts result in better performance but is more power consuming. In contrast, sequential searches may cause more latency but save power. The sequential search mechanism can be handled by a cache search selector shown in figure 7. The cache search selector determines which part of the L2 cache has to be searched first, regarding the access type (read or write). If the requested data was not found in the first cache, second cache will search next. Obviously, as frequently written data are kept in LR part if there is a write request first LR part is searched and then HR part. For read accesses this action happens in reverse.

The main overheads of the proposed architecture include RCs of cache blocks in HR and LR parts and the buffers between LR and HR parts. Note that the overhead of these 2 and 4 bit counters with respect to typical block sizes (256B in our design) is negligible. After analyses, we carefully determined the buffer size to hold 20 cache lines and found mined based on simulation experiments). The use of larger RCs in LR side is due to the lower retention time of blocks in LR part which requires a more accurate refresh mechanism. On buffer full, dirty lines are forced to be written back in main memory, in order to avoid data loss. Fortunately, this is rare and worst-case happens for mri-q with 1% write-back overhead. No refreshing mechanism is considered for the few blocks which violate 40ms rewrite period in HR part. Such blocks can be simply invalidated or forced to write back if blocks are dirty to guarantee correct execution of applications.

Figure 6: Rewrite interval time distribution in LR cache.

Figure 7: Block diagram of the proposed L2 architecture.
that such storage is enough to handle movements in almost all workloads. Synthesis shows that the area overhead of added RCs and buffers (each capable of holding 10 blocks) in our design is less than 6KB (lower than 1%).

6. EVALUATION RESULT

To evaluate the proposed architecture, GPGPU-Sim3.2.1 simulator [9] was used to simulate PTX commands at cycle level. This simulator mimics the behavior of SMs, L2, interconnection network, and off chip main memory. For simulations, 3 groups of benchmarks were considered: 1) benchmarks provided with the simulator [3], 2) Rodinia [4] and 3) Parboil [13]; we used CUDA version of benchmarks in our experiments. For calculating power, area and latency values, we used CACTI 6.5 [10] slightly modified for STT-RAM.

According to the investigations carried out in [8], GPGPU applications are divided into 2 general categories based on their behavior respect to cache: 1) cache insensitive and 2) cache sensitive (cache friendly). Note that increasing the cache size can increase the performance of some applications, while some other applications do not benefit from larger cache sizes. Of course, the saved area as a result of replacing the SRAM L2 by STT-RAM L2 can be used to add new components to favor an application to run faster. In this paper, for a fair comparison, we have considered using the saved area for additional resources in the GPUs in two ways: 1) The saved area is used for having a larger L2 cache, or 2) The saved area is used for adding other useful resources (here we considered larger register file). Since the STT-RAM cell is 4× denser than the SRAM cell, the simulations were done for 3 configurations: C1) Replacing L2 with a STT-RAM of the same area, i.e. a 4× larger L2 cache, C2) Replacing L2 with a same-sized STT-RAM where the saved area is used for having larger register file in SMs, and C3) A combination of C1 and C2, that doubles the size of L2 and uses the remaining area for enlarging register files in SMs. Note that as at least data array area is 80× than tag array area in our configuration, we keep tag array SRAM so it is fast and its area overhead remains insignificant. Table 2 shows these configurations based on GTX480 structure. Simulation results are compared to the baseline GPU with SRAM L2 and a GPU using a normal STT-RAM L2 with 4× capacity.

Figure 8-a shows the achieved speedup values with respect to the SRAM baseline system in 3 different regions: the region 1 includes applications which cannot benefit either from larger caches or register files. In regions 2 and 3, the applications are grouped whose kernels (or at least one of the kernels) suffer from register file shortage. In this region, no speedup is gained for some benchmarks since the enhanced register file could not help the GPU to assign more thread blocks to a SM. This is even more visible in C3, as with doubling the size of STT-RAM L2 cache, a smaller register file, in comparison with C2, can be formed; so, applications like histo could not benefit from the added resources and no IPC improvement is resulted. Regions 3 and 4 include the benchmarks that can benefit from larger cache sizes so region 3 contains applications which are cache friendly and suffer from shortage of register file.

Using the STT-RAM baseline system improves the average Instruction Per Cycle (IPC) by 5%, while exploiting C1 architecture has resulted in 16% improvement (and up to more than 100%). It must be noted that no performance degradation is observed for applications like MUM, mgridder and lavaMD, in C1, C2, and C3 systems, while the STT-RAM baseline system shows performance degradation for these benchmarks.

Figure 8-b shows the dynamic power consumed by different architectures normalized to that of the SRAM baseline system. Dynamic energy consumption in STT-RAM architectures is high due to the write energy of MTJ cells. The write energy in STT-RAM cells is more than SRAM cells even after reducing the retention time of STT-RAM cells. As can be seen in the figure, the average dynamic power consumed in C1, C2, and C3 are 1.69×, 1.67×, and 1.94× that of the SRAM baseline, respectively. The benchmark suit composes of various applications with near zero (streamcluster) to 63% (sad) of write operations that shows the proposed architecture efficiency in handling write operation in L2 cache. However, for some applications, our proposed architectures consume higher amount of dynamic energy than the SRAM baseline as a result of their single write traffic into HR part of the cache. Note that the dynamic power consumption in the STT-RAM baseline system has become 5× when compared to C1; this is mainly due to the efficient mapping of the blocks onto LR and HR parts in the proposed architecture.

As the leakage power of magnetic memory cells are negligible, the average total consumption power of the whole L2 cache has reduced by 20%, 63.5% and 42% in C1, C2, C3, respectively, compared to the SRAM baseline system. It is noteworthy that the STT-RAM baseline system, however, has consumed 19% more energy with respect to the SRAM baseline system, despite its lower leakage power.

7. CONCLUSION

Current trends in VLSI technology and GPU architectures show that future GPUs should have larger L2 caches with a proportionally larger power consumption. Due to the desirable properties of STT-RAM devices (low leakage power and high density), STT-RAM can replace SRAM L2 caches in future GPU architectures. However, STT-RAM high write latency and energy are two main drawbacks that have to be dealt with before it can replace SRAM. STT-RAM cells with lower retention times can be employed to remedy the latency and energy problems of write operations but they require refreshing. This paper proposed a two-part STT-RAM L2 cache architecture for GPUs based on a detailed workload investigation of GPGPU applications. It tries to keep frequently-written blocks in a small low-retention part.
of the cache while less-frequently written high-frequently read blocks are mapped onto the high-retention part of the cache. Our evaluations showed that our architecture could effectively improve IPC by up to more than 100% (16%, on average) while the average L2 cache energy consumption is improved by 20%, when compared to the conventional SRAM based L2 architecture in current GPUs.

8. REFERENCES