An Energy-Efficient and Scalable eDRAM-Based Register File Architecture for GPGPU

Naifeng Jing [1,3], Yao Shen [1], Yao Lu [1], Shrikanth Ganapathy [4], Zhigang Mao [3], Minyi Guo [1,2], Ramon Canal [4], Xiaoyao Liang* [1,2]

[1] Department of Computer Science and Engineering
[2] Shanghai Key Laboratory of Scalable Computing and Systems
[3] School of Microelectronics
Shanghai Jiao Tong University, Shanghai, China
{sjtuj, liang-xy}@sjtu.edu.cn

[4] Universitat Politecnica de Catalunya, Barcelona, Spain
{sg, rcanal}@ac.upc.edu

ABSTRACT

The heavily-threaded data processing demands of streaming multiprocessors (SM) in a GPGPU require a large register file (RF). The fast increasing size of the RF makes the area cost and power consumption unaffordable for traditional SRAM designs in the future technologies. In this paper, we propose to use embedded-DRAM (eDRAM) as an alternative in future GPGPUs. Compared with SRAM, eDRAM provides higher density and lower leakage power. However, the limited data retention time in eDRAM poses new challenges. Periodic refresh operations are needed to maintain data integrity. This is exacerbated with the scaling of eDRAM density, process variations and temperature. Unlike conventional CPUs which make use of multi-ported RF, most of the RFs in modern GPGPU are heavily banked but not multi-ported to reduce the hardware cost. This provides a unique opportunity to hide the refresh overhead. We propose two different eDRAM implementations based on 3T1D and 1T1C memory cells. To mitigate the impact of periodic refresh, we propose two novel refresh solutions using bank bubble and bank walk-through. Plus, for the 1T1C RF, we design an interleaved bank organization together with an intelligent warp scheduling strategy to reduce the impact of the destructive reads. The analysis shows that our schemes present better energy efficiency, scalability and variation tolerance than traditional SRAM-based designs.

1. INTRODUCTION

General-Purpose computing on Graphics Processing Unit (GPGPU) has enabled extraordinary acceleration on parallel data processing in the many-core era. It explicitly exploits the data parallelism through the ever-increasing on-chip computing power. For instance, the recent NVIDIA GTX680 GPU provides 3090 GFLOPS/s with 1536 cores [15]. This core count is dramatically larger than that of the conventional CPUs. The massive data parallelism comes from the higher memory bandwidth and the larger number of parallel threads executed in a SIMD (Sigle-Instruction Multiple-Data) fashion. In NVIDIA GPGPUs, threads are organized and issued in warps. GPGPUs typically implement a zero overhead context switch between warps to hide the long latency memory operations. To achieve that, GPGPUs need a large number of registers to hold the contents and states of all active warps. In the NVIDIA Fermi architecture, each SM is equipped with a 128KB RF that caters for 1,536 active threads [8].

The default option to build RF is SRAM. However, such a large-sized RF can easily occupy significant amount of chip area since each SRAM cell requires at least six transistors (plus some wiring). It is also very power hungry mostly due to the high leakage current in SRAMs. In addition, the size of the RF is still far from enough when considering scalability as we see the total thread count grows quickly in every product generation. Therefore, the problem of low density and high leakage of SRAM-based RF will soon become the bottleneck in the future GPGPUs. In contrast, modern CPUs have begun to replace SRAM with eDRAM as the primary storage component [24]. eDRAM features higher density and lower leakage in comparison to SRAM. Given the fact that the majority of commercial GPGPUs are operating at a relatively low frequency (600MHz-1GHz), modern eDRAM cells can easily meet the speed requirement [13, 14] while providing doubled density and one order of magnitude lower leakage per bit [22, 4]. Therefore, eDRAM potentially offers new opportunity in the design of on-chip memory structures on GPGPU.

Despite the potential benefits of using eDRAM, it also poses new challenges to processor architects. The challenge arises from the weakness of eDRAM cells: the well-known data retention problem. In order to maintain the data integrity of the memory cells, the eDRAM requires periodic refresh operations. Refreshes occupy the normal operating cycles and they will block the memory for normal data service. Furthermore, temperature and process variations adversely impact the retention time. In a word, the refresh operations are the major impediment for the wide adoption of eDRAM as the on-chip storage.

In this paper, we propose to use eDRAM as an alternative for building the RF in a state-of-art GPGPU. Our study evaluates the impact on system performance and energy consumption, including dynamic, leakage and refresh

ISCA '13 Tel-Aviv, Israel

energy across several technology nodes for both 3T1D and 1T1C eDRAM cells. We propose several optimization techniques to reduce the performance penalty incurred by refresh operations. We find that the heavily banked RF architecture in modern GPGPUs provide a unique opportunity to hide the refresh overhead if a smart refresh policy is put in place. In summary, we make three contributions in this paper:

- We quantitatively demonstrate the performance and power impact of replacing the SRAM cells with two types of eDRAM cells. We also analyze the cons and pros for such replacement.
- We explicitly exploit the particularities in the GPGPU instruction flow and propose novel solutions to reduce the penalty caused by refresh operations in eDRAM.
- We study the impact of process variations on memory structures and show that our proposed schemes present better variation tolerance and scalability than traditional SRAM-based RF designs.

This paper is organized as follows. Section 2 reviews the background of on-chip memory technology and process variations. It describes the GPPGU execution pipeline principle and the banked RF structure with operand collector. In the 3rd and 4th sections, we introduce our RF implementation details with eDRAMs together with the novel refresh schemes and warp scheduling strategies. Section 5 presents our evaluation methodology. The detailed performance, power and area results are presented in section 6. Section 7 introduces the related work and finally, section 8 concludes this paper.

2. BACKGROUND

In this section, we first review different memory technologies for on-chip data storage. After that, we briefly introduce the impact of process variations on memory behavior. Finally, we discuss the general RF architecture and working principle in current GPGPUs.

2.1 On-chip Memory Technology

2.1.1 SRAM

The state-of-art CPU designs are highly pipelined to boost the operating frequency. In order to keep up with the fast speed, on-chip memory structures must have fast data access. Consequently, high performance SRAM is currently the default option for RF or cache implementations. As shown in Fig. 1(a), a typical SRAM cell is made of six transistors. Four transistors ($T_1 - T_4$) form two cross-coupled inverters to store a single bit of data. This storage cell has two stable states, i.e. 0 and 1. Two access transistors (T_5, T_6) serve to control the access to the storage cell when the cell is being read or written. The value stored in the cell is accessed when the word line (WL in the figure) is enabled on T_5 and T_6. These transistors connect the storage cell to the bit lines $\overline{BL}$ and BL. The symmetric structure of SRAM cell allows for differential signaling, which eases the detection of small voltage swings by a sense amplifier. Such an SRAM cell is designed for single read or write operation at a time. To provide multiple data access at the same time, one can insert multiple ports with extra transistors added to the basic cell incurring much area penalty. Since the RF size of CPU is not overwhelmingly large, it is still affordable to implement multi-ported RF structures with SRAM.

2.1.2 eDRAM

eDRAM technology originates from discrete DRAM technology, but is adopted to be compatible with the CMOS logic fabrication processes. Recently, there is a growing interest in the studies of applying eDRAM for on-chip memory storage due to its higher density and lower leakage in the general purpose processors [14, 26]. For instance, IBM Power4 and Power5 processors implement their L2 caches with the logic-based eDRAM technology [21, 18]. The Power7 microprocessor also includes an eDRAM-based L3 cache [24]. Besides, IBM also uses eDRAM in the network data router [9] and Microsoft uses eDRAM in their game console XBox360 [19, 5].

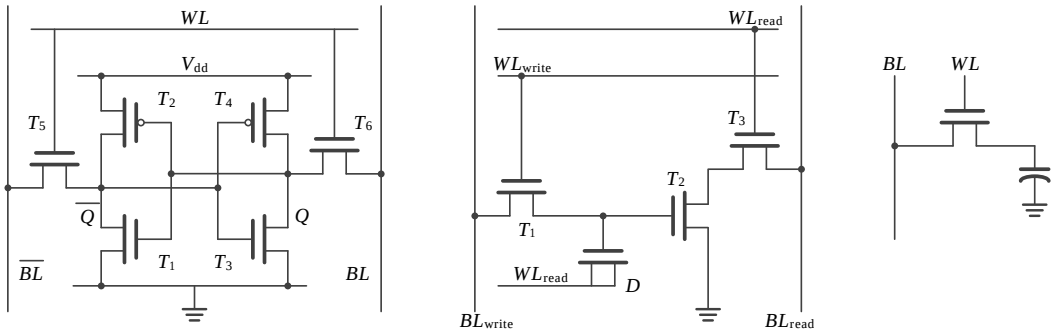
In contrast to SRAM which uses six transistors for the storage of one bit data, the simplest eDRAM cell (1T1C) consists of only one transistor for access and a dedicated capacitor to store the data, as shown in Fig. 1(c). Due to charge sharing between the storage capacitor and the bitline, reading of the 1T1C cell is destructive, meaning that each read of the memory must be followed immediately by a write back to prevent data damage. Another popular eDRAM structure is the 3T1D cell as shown in Fig. 1(b), which leverages the gate capacitance of a diode for data storage and also for enhanced access speed. This is also called gain cell eDRAM. The read operation in a 3T1D cell is not destructive because there is no charge sharing as is the case in 1T1C. Nevertheless, it is not as dense as the 1T1C cell since it incorporates more transistors and the leakage power is slightly higher. In addition, the 3T1D eDRAM cell is logic-compatible and does not require extra masks and processing steps. Although the 1T1C eDRAM requires extra masks and processing steps involving adding the DRAM deep trench and other processes to the logic process, the logic libraries do not have to be remapped to new ground rules, which is a significant extra advantage [14]. As a result, these two types of cells are commonly used in current designs.

Unlike SRAM cells, eDRAMs need periodic refresh operations to maintain data integrity because data values are stored on capacitors and the charge gradually leaks away. For a two dimensional eDRAM array, the periodic refresh scheduler initiates the refresh operations row by row. A refresh operation includes a read of the stored value and a write back. Each eDRAM cell has to be refreshed before reaching the retention time, otherwise the data stored has the risk of being lost. During the refresh operation, the memory is freezed and any normal read/write operation to the bank is stalled until the refresh completes.

2.2 Process Variation Impact on Memory

Variations arise from manufacturing process shifts and other environmental effects. It has been acknowledged that process variations greatly impact the circuit performance. Circuit designers and computer architects have already taken the variation effect into consideration at the design stage to ensure that the final product runs as expected.

Variations in memory circuits will affect SRAM performance and power which are critical to high-performance microprocessors. Current microprocessors heavily employ SRAM for on-chip data storage. However, SRAMs are facing increasing read/write stability issues with the continu-



(a) 6T-SRAM (b) 3T1D-eDRAM (c) 1T1C-eDRAM

Figure 1: Schematic of three on-chip memory cells

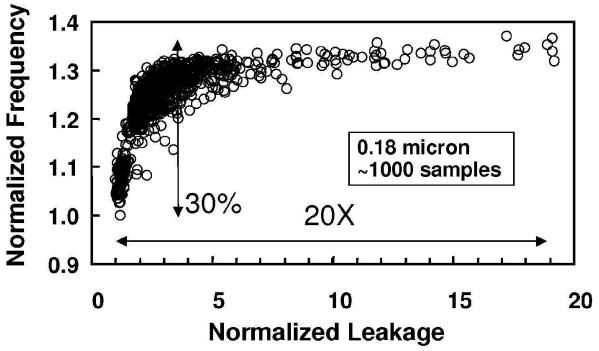


Figure 2: Impact of process variations on frequency and leakage [7]

ous shrinking in transistor sizes. The parameters affected by process variations include gate length, gate width, oxide thickness and the dopant concentration. Fig. 2 illustrates the experimental results from Intel. It shows that the impact of the oxide thickness variations can cause 30% frequency deviation with 20X leakage variations in the massive chip production. As technology scales, the variation problem is becoming even more critical. In the SRAM, process variations can cause significant access speed degradation because a bunch of slow cells determine the final operable frequency.

Process variations also affect the eDRAM cell. In the e-DRAM cell, if the driving capability of the access transistor is reduced due to variations, the access speed of the cell will reduce. However, this effect can otherwise be viewed as a shorter retention time with unchanged access speed, as discussed in detail in [11]. The speed of an eDRAM cell gradually slows down after refresh. We define the retention time as the time point where the speed of the eDRAM cell falls below the nominal operating frequency. In this sense, process variations do not necessarily affect the operating frequency as the SRAM. In the presence of process variations, the cells in eDRAM array present a distribution in retention time rather than the access speed. In addition, the eDRAM does not suffer from the cell stability problem – as seen in SRAM – since it does not rely on the stability of the cross-coupled inverters. As a result, eDRAM is an attractive and viable option for constructing on-chip processor memories except for the finite data retention time which needs to be taken into consideration.

2.3 GPGPU Pipeline and the Banked Register File Architecture

Typically, modern GPGPUs consist of many small cores called stream multiprocessors. In the NVIDIA Fermi architecture, each SM has 32 data processing lanes, one instruction cache, one scheduler, one RF, one fast L1 cache/shared memory and a large number of computing units as shown in Fig. 3. The execution of computing tasks is highly parallelized. For instance, threads are often issued in groups of 32, called a warp. At any given clock cycle, a warp that is ready for execution is selected and issued to the datapath by the instruction scheduler. The RF is designed for fast data access and it is shared by all threads in the SM.

In order to fully use the data parallelism, thousands of threads are concurrently present in the SM as long as there are sufficient hardware resources (RF, shared memory, etc). The scheduler works at a warp granularity to issue ready warps. All 32 threads in the same warp are executed in a SIMD manner. A distinctive feature of GPGPU hardware design is that the warps can be switched with zero cycle penalty. Consequently, GPGPUs can keep the execution units busy even if some warps are stalled for long memory operations.

To support the zero cost warp switching in the GPGPU, each thread in the SM is allocated with some dedicated physical registers in the RF. As a result, the RF in a GPGPU is huge. For instance, the recent NVIDIA Fermi architecture is equipped with a total of 32,768 registers of 32-bit wide each, which adds up to 128KB per SM and 2MB for the whole chip. This number is far beyond the RF size in a state-of-art CPU. The RF size is still increasing for each product generation as it has to support a larger number of concurrent threads for larger parallelism in execution.

Another architectural challenge brought by the SIMD execution is the simultaneous multiple read and write operations to the RF. In the NVIDA PTX standard, each instruction can read up to 4 registers and write 1 register at the

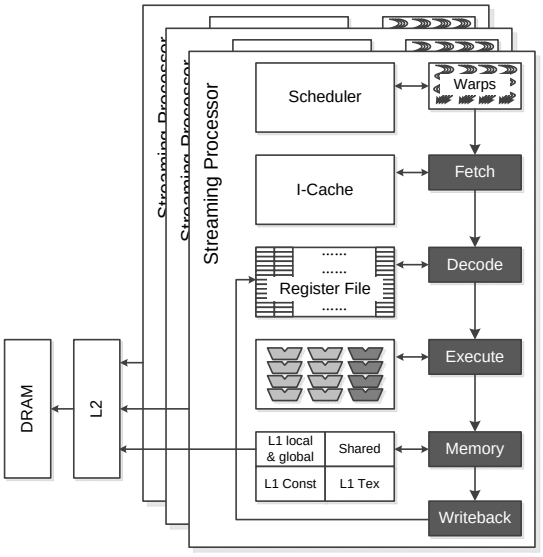


Figure 3: Block diagram of a GPGPU pipeline

same time. One solution is to use a multi-ported SRAM, which will require two more transistors for an additional port. To construct such a heavily-ported and large-sized SRAM is typically not feasible or economical. To solve this problem, NVIDIA has proposed a banked RF architecture together with an operand collector to avoid the huge area and power overhead caused by multi-ported RFs [25, 17]. A banked architecture combines multiple single-ported RF banks and distributes physical registers across the banks. Each bank has its own decoder, sense amplifiers and it can operate independently. The purpose is to mimic a multi-ported RF to sustain the demanding needs of concurrent reads and writes, while not paying the prohibitive hardware cost.

The operation of a banked RF is shown in Fig 4. There are multiple single-ported RF banks for data storage, a crossbar unit for data transfer and several operand collectors for collecting register values. When the arbiter receives the RF access requests, it tries to distribute the accesses to different banks but it only allows a single access to a bank at a time. If there are bank conflicts, the requests are queued and serialized. For an instruction reading 4 source registers, a single cycle RF access is possible only if the 4 registers are allocated in different banks. Otherwise, the pipeline can stall up to 3 cycles in the worst case if all the 4 source registers are located in the same bank. Also, if read and write operations conflict in the same bank, they have to be serialized as well since the memory bank can host only one operation at a time. Once all the register values are collected for an instruction, the warp can be issued to the execution units, freeing a collector for the next instruction.

3. RF IMPLEMENTATION WITH EDRAM

As discussed above, the SRAM-based RF faces scaling issue and it might incur an unacceptable hardware cost due to the ever-growing thread count. In this section, we propose to use eDRAM for implementing the RF in a GPGPU architecture.

3.1 Basic Behavior of a 3T1D Cell

The basic structure of a 3T1D cell is shown in Fig. 1(b). Due to its compatibility with standard CMOS process technology, it has been widely used as the SRAM alternative for on-chip memory. A 3T1D cell consists of a write access transistor (T_1), a read access transistor (T_3) and a storage transistor (T_2) where the charge (data value) is stored at the gate capacitor. During a write operation, the write bit line (BL_{write}) is driven to the desired voltage, and the transistor T_1 is activated through the write word line (WL_{write}) passing charges to the gate capacitor. For a read operation, the read bit line (BL_{read}) has to be precharged at V_{dd}, and then T_3 is activated (i.e. WL_{read}). If a high charge level is stored, T_2 turns on and discharges the bit line. Since there is no charge sharing in the read access, the 3T1D cell features non-destructive characteristics.

The 3T1D eDRAM cell has comparable access speed to a 6T SRAM cell as has been verified by published results from fabricated chips [13]. It can easily meet the sub-1GHz frequency requirement in modern GPGPUs. Beyond that, it has the advantage of smaller silicon area, lower leakage and non-destructive read operations. These benefits make it a promising candidate in building RF for modern GPGPU.

3.2 Basic Behavior of a 1T1C Cell

The 1T1C cell has even smaller footprint and leakage power than the 3T1D cell, and thus can be leveraged to accommodate even larger number of concurrent threads. But the destructive read in 1T1C cell poses new challenges to the architecture design.

A 1T1C cell consists of an access transistor and a capacitor as shown in Fig. 1(c). Fewer number of transistors makes the 1T1C cell the densest memory structure which is largely demanded for GPGPU RF design. However the additional concern of using 1T1C cell is that the read access is destructive. Whenever the access transistor is turned on, the charge stored on the capacitor is shared with the bit line and the stored value is destroyed. An immediate write-back operation after every read is required to restore the capacitor to the original value. In a conventional architecture, write-backs consume extra clock cycles and they almost cut the RF throughput by half.

3.3 RF Design with eDRAM Cells and the Feasibility

The eDRAM array is quite similar to a conventional SRAM array. It is relatively straight-forward to replace the SRAM cells with eDRAM cells to reform the memory array in the RF. To keep up with the high data bandwidth, our eDRAM-based RF uses the same banking organization and peripheral circuits as in the SRAM. The sense amplifier design is slightly different since eDRAM can only perform single-ended probing.

The eDRAM is conventionally believed to be slower than SRAM, but is fast enough to meet the GPGPU requirement. The high performance of GPGPU is achieved through massive parallelism rather than absolute frequency. Public results [13] shows that the CMOS-compatible 3T1D cell has much shortened retention time typically at the magnitude of a couple of *us*, much smaller than the traditional off-chip DRAM with the retention time of hundreds of *us*. This poses new challenges when applied to our proposed GPGPU RF designs.

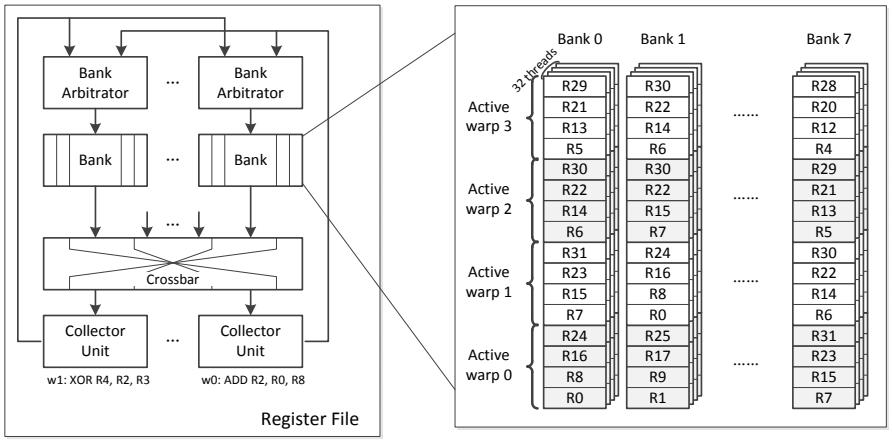


Figure 4: The banking scheme and operand collector in GPGPU RF

4. REFRESH SCHEMES

To ensure data integrity, eDRAM arrays must embody a refresh mechanism. This mechanism should enforce *correctness*, i.e., guarantee that data stored in the eDRAM cells are refreshed before they reach the retention time. On top of that, the refresh scheme should be designed with high efficiency and low complexity to provide better *serviceability* in terms of performance and energy consumption for the GPGPUs.

4.1 Full RF Refresh (B-All)

In this paper, we perform the refresh at a register entry level. A register entry is the single named register for each of the 32 threads in a warp. As shown in Fig 4, register entry $R0$ stands for a group of 32 registers containing a total of 1024 bits since each register is 32-bit long in the GPGPU. The RF is then built upon a large number of register entries, organized in multiple banks to provide parallel data access capability.

The simplest way to refresh the eDRAM RF is to use a refresh counter. The counter records the retention time of the RF and resets the counter after every refresh. If the counter is about to expire, the entire RF is frozen and any normal traffic to the RF is completely blocked. This will cause stalls to the execution pipeline for the entire period of the refresh. Inside the RF, all banks are refreshed in parallel but register entries in the same bank are refreshed in a sequential order. Once all entries in the RF are refreshed, the RF is released and back to normal instruction accesses.

4.2 Refresh Using Bank Bubble (BB)

In this section, we propose a novel refresh strategy to mitigate the performance penalty introduced by refresh operations. It is clear that eDRAM-based RF has to pause normal operation for periodic refresh in any retention time interval and this can possibly cause performance penalty. To reduce the penalty introduced by the unavoidable refresh, we propose a fine-grained refresh strategy that takes advantage of RF bank bubbles. Recall the structure in Fig 4, where the multi-banked RF is designed to mimic a multi-ported RF by distributing simultaneous RF accesses into different memory banks. Although effective, the multi-banked RF does not behave 100% like a multi-ported RF and there are always chances of bank conflicts. Due to the inevitable bank conflicts, RF accesses have to be serialized and thereby cause pipeline stalls. Like the instruction of warp 0 ($w0$) in Fig 4, reading of $R0$ and $R8$ happens to be in the same $Bank0$ causing a bank conflict. This bank conflict causes the instruction to be stalled for one cycle to accommodate the consecutive access to the same bank. During the stall cycle, all other banks remain unoccupied and this provides perfect opportunity for refresh. We call this a bank bubble. Bank bubbles may also occur when the RF is not fully loaded. NVIDIA PTX support 4 register reads for every instruction but many instructions only read two or less registers while the banked RF can support six more accesses (considering a total of eight banks as an example) if there are no conflicts.

To hide the refresh cost under the presence of opportunistic bank bubbles, we associate each register entry with a refresh counter. The counter is reset after every refresh. Otherwise it keeps counting till it reaches the retention time. The refresh action must be performed before the counter expires or data loss might occur. Whenever there is a bank bubble detected by a bubble checker as in Fig. 5, we will opportunistically refresh one of the register entries by the refresh generator. The strategy is to refresh the entry with the counter close to expiration. We select an entry for refresh if its counter value falls below a predefined threshold. A simple comparing logic detecting the leading zeros of a vector is used to determine if the counter value is smaller than the threshold. Multiple entries from different banks can be refreshed at the same time. But if multiple entries from the same bank are eligible, the generator randomly picks one. Using the opportunistic scheme, most of the refresh operations are hidden behind bank bubbles, greatly reducing the RF blocking time for normal service. Nevertheless, the opportunistic scheme cannot guarantee that every entry gets refreshed in time. If there is no bank bubble for a long time and the refresh counter is about to expire (determined by another smaller threshold), a mandatory refresh is enforced,

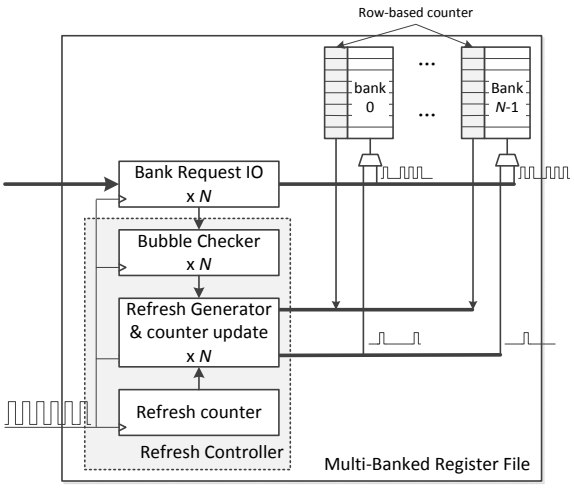


Figure 5: The BB scheme in a refresh controller

blocking that RF bank from normal access.

The BB scheme is fine-grained and requires a refresh counter for each register entry. These counters are driven by a system clock much slower than the core frequency. This helps reduce the number of bits required in the counter so that the hardware cost is reduced. We use 4-bit counters in this paper. Considering the area reduction by replacing SRAM with eDRAM, adding these counters will not increase the total RF area.

4.3 Refresh Using Bank Walk-Through (BW)

One issue in the BB scheme is the extra hardware overhead to implement the per-entry refresh counters. In a Fermi style RF, there could be 1024 register entries requiring the same number of refresh counters in a SM. Although the area cost is affordable because the eDRAM array inherently saves area, we still want to look for other schemes that can further reduce the hardware cost. We propose another efficient refreshing strategy called bank walk-through. This scheme only refreshes one RF bank at a time and different banks are staggered for refresh. The register entries in different banks take turns to perform the refresh operations. We only need one global refresh counter for this scheme. If the counter is going to expire, we start to walk through the banks one by one, each time refreshing one entry in the bank. The process is repeated until all the entries in all the RF banks are refreshed and after that the refresh counter is reset.

The biggest disadvantage of B-All scheme is that the normal RF access is completely blocked during the refresh period. BW scheme solves this problem by staggering the refresh operations permitting only one bank for refresh at a time. While that bank is blocked, other banks are still up for normal service. The refresh counter counts in the time for walking through all the banks and guarantees the retention time limitation. If a ready instruction accesses registers not in the bank being refreshed, it can just proceed as normal. On the other hand, even if the bank is under refresh, the instruction is still likely to be served in a few cycles. This is not the case in the B-ALL scheme where all instructions have to wait for hundreds of cycles until the entire RF is refreshed.

4.4 RF Reorganization with Odd-Even Warp Rescheduling

The 1T1C cell has smallest cell area and lower leakage power consumption making it an attractive option. But the destructive read in 1T1C cell poses a new challenge. In this paper, we propose a novel RF organization together with a simple warp scheduler to mitigate the impact of destructive reads in the 1T1C eDRAM. Firstly, we need to double the number of RF banks. In the Fermi style GPGPU, this increases the total number of RF banks from 16 to 32. To maintain the same number of registers, we cut the total entries in each bank by half. The banks are divided into two groups as shown in Fig. 6. $Bank0-7$ forms $group0$ and $Bank8-15$ forms $group1$. The warps are also divided into even and odd warps based on the warp ID. Then, even warps are assigned to bank $group0$ and odd warps to bank $group1$.

To hide the penalty of the destructive reads, we also need to change the warp scheduling policy. The warp scheduler will now take turns to issue from even and odd warps at consecutive cycles. When an even warp instruction is issued to bank $group0$, it takes one cycle to perform the read access and the next cycle to perform the write-back. The scheduler will force the issue of an odd warp instruction to bank $group1$ following the even warp issue slot so that there is no bank conflict. By interleaving the issue of odd and even warps to two different bank groups, the performance impact of the destructive reads is greatly reduced. Note that the write-back can be coalesced with the normal RF write if there is no bank conflict. Like in the SRAM case, if read, write or write-back conflict in the same bank, they have to be serialized giving write-back the highest priority. The downside of this strategy is that it cuts by half the available candidate warps for issue at any given cycle. To reduce the performance impact, we allow to issue warps with the same polarity in consecutive slots if there are no ready warps to issue in the other group. In this case, bank conflicts can occur when read operations from the second instruction access to the same bank as the write-back from the first instruction. This situation will cause a pipeline stall.

5. EVALUATION METHODOLOGY

In our evaluation, we focus on the performance, power and area associated with the SRAM-based RF implementation and our proposed eDRAM-based RF architectures implemented in either 3T1D or 1T1C eDRAM cells.

For the performance and power evaluation, we use the cycle accurate architectural simulator GPGUP-Sim v3.1 [6]. Table 1 shows the configuration for the simulator. Each shader core has 32 execution units and a 128KB RF. The RF is composed of 16 banks to serve at most 2 warp instructions at a time. These settings are common in the Fermi style architecture.

In our evaluation, the SRAM-based RF is used as our baseline design for comparison. It uses single-ported SRAM cells with multiple banks to provide the analogy of a multi-ported SRAM RF with operand collectors. For the eDRAM-based RF, we modify the architectural design to support our proposed new organizations and features such as the back pressure scheme due to the refreshing operations. The simulator is also augmented to model the two proposed eDRAM cells, including their read and write behaviors, refreshing

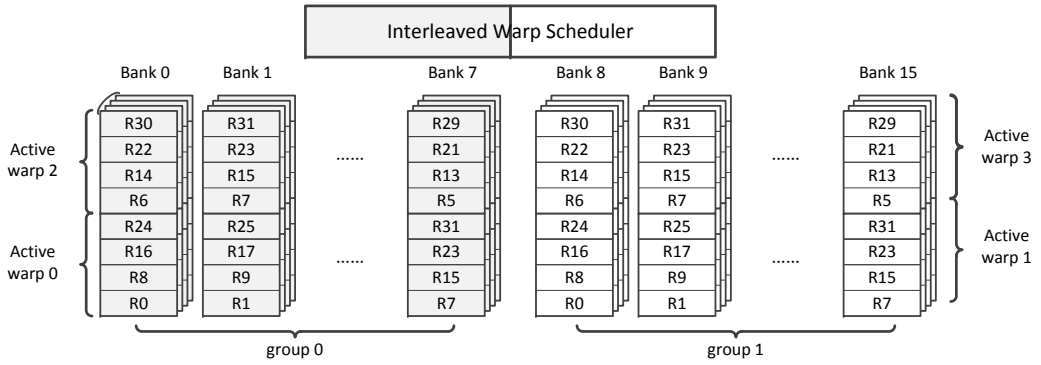


Figure 6: Bank reorganization for 1T1C RF

Table 1: Key parameter settings in the GPGPU-Sim simulator

Parameters	Setting
Number of Core (SM)s	15
Core (SM) Frequency (MHz)	700 MHz
Warp Size	32
Number of Threads / Core	1,536
Number of active CTAs / Core	8
Number of Registers / Core	32,768
Number of Collecter Units / Core	4
Warp Scheduling Policy	Round Robin

operations and induced latency.

Table 2 shows the benchmarks used in our simulation. Eight benchmarks are taken from the widely used GPGPU-Sim benchmark suite [6] and CUDA SDK [16], which cover a wide range of applications and various register file stressing conditions. We use CUDA SDK 4.2 for compilation, and the simulator is configured to use PTXPlus syntax for simulation, which is a hardware instruction set extended by GPGPU-Sim to provide more accurate simulation results as reported in [23].

Table 2: GPGPU benchmark characteristics

Name	Description	Regs per Thread
AES	AES Cryptography	14
IMG	Image Denoising	24
LIB	LIBOR Monte Carlo	18
MAT	Matrix Multipliy	13
MUM	MUMmerGPU	16
NN	Neural Network	13
SAD	Sum of Absolute Differences	29
WP	WRF microphysics	60

In the experiments, different refreshing strategies and RF organizations are evaluated and compared with the SRAM baseline design for various retention time values (t_{RET}). Constrained by the RF speed requirement, the values of t_{RET} are selected based on the HSPICE simulation using PTM CMOS model [1] on 45nm, 32nm and 22nm technology nodes, which range from about 11.74us down to 3.01us with shrinking geometry sizes. The retention times are translated to equivalent retention cycles from 8K to 2K given the SM operating frequency. Furthermore, we also scale down the retention cycles to predict the future smaller feature sizes, such as 16nm and 11nm technology nodes with extrapolated values of 1K and 512 cycles, respectively.

Table 3: Simulated data of three memory cells at 45nm technology node

	SRAM	3T1D	1T1C
Read Energy (fJ/access)	422	340	281
Write Energy (fJ/access)	170	134	108
Leakage Power (uW/bank)	28.6	17.2	4.08
Cell Size (mm^2)	80	37	18

For the evaluation, we use the execution time reported by the simulator to quantify the GPGPU performance under different conditions. The energy consumption of the RF is estimated by the statistics from the architectural simulator and HSPICE simulated energy primitives considering both of the memory array and the peripherals circuitry such as sensor amplifiers, pre-charge, decoder and wordline drivers similar to the modeling method described in [21]. We document our HSPICE simulated power primitives such as read, write and leakage for the three kinds of memory cells in Table 3 using the 45nm node. Combined with the statistics collected from the simulator, we can calculate the actual dynamic, leakage and refresh power numbers. We also incorporate the refresh counter power into the refresh power. We assume an environment temperature of 70°C for all of our simulations.

The eDRAM saves significant memory array area, e.g. over 50% for 3T1D cells and over 70% for 1T1C cells. The saved area is sufficient to accommodate the extra refresh counters with associated logics (less than 3% area overhead), doubled decoders and sense amplifiers (less than 10% overhead) in the odd-even rescheduling scheme introduced in our eDRAM-based RF designs. For a fair comparison, we assume the same amount of total RF registers for both the SRAM and eDRAM cases. Nevertheless, eDRAM designs have smaller overall footprint and the saved chip area can be used for other purpose such as adding more registers. We do not consider such possibility in this paper.

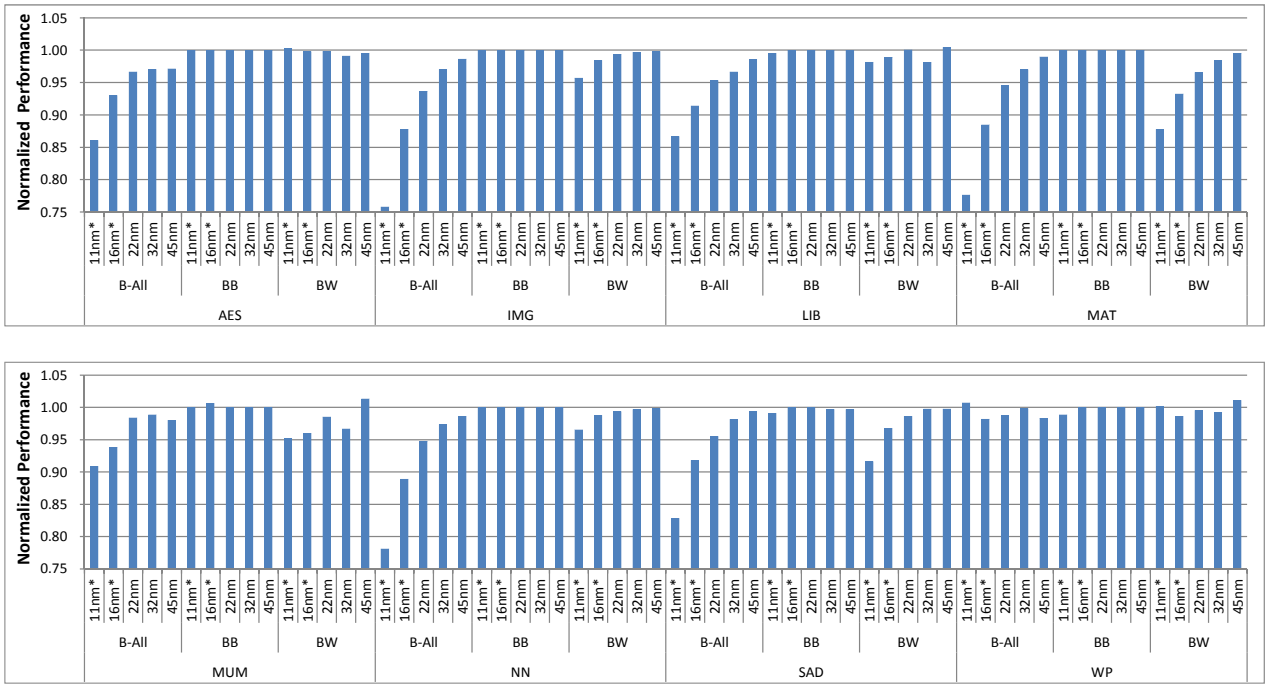


Figure 7: Normalized performance of the 3T1D RF (* represents predicted technology nodes)

6. EXPERIMENTAL RESULTS

In this section, we present our simulation results in terms of energy and performance. Most results will be normalized to the SRAM counterpart for easy comparison. We will also look into the variation tolerance of the proposed schemes.

6.1 Evaluation of RF Design with 3T1D Cell

Figure 7 demonstrates the simulated performance results for the eight benchmarks using 3T1D eDRAM-based RF. The results are normalized to the SRAM-based design at the same technology node to show the performance degradation due to refresh operations. Three refresh strategies – the basic full RF refresh (B-All), the bank bubble refresh (BB) and the bank walk-through (BW) are evaluated. They are further categorized into the five technology nodes – 45nm, 32nm, 22nm, and the predicted 16nm and 11nm which represent much tighter refresh requirements

From the figure, we can see that for the basic B-All refresh scheme, the degradation in execution performance is significant with shrinking geometry sizes and tighter refresh requirements. For instance, more that 30% penalty can be observed for "IMG" and "NN" under the B-All refresh strategy. In this case, the small retention time causes frequent refresh operations which then block the normal RF traffic for relative longer time. When the retention time gets longer, the performance penalty for 3T1D RF reduces notably. For instance, with the technology nodes at 45nm, even the basic B-All refresh strategy can deliver comparable performance as SRAM for most cases.

The B-All strategy with a simple refresh counter generally presents acceptable performance when the retention time is not too tight. In contrast, our proposed BB refresh strategy intelligently hides the refresh cost into the RF bank bubbles Fig. 7 shows that it can deliver SRAM-comparable performance even for small retention times for all the investigated applications. It applies per register entry refresh counter to enable independent and opportunistic refresh. The hardware cost is higher than the B-ALL scheme but the significant performance benefit justifies the investment. However, if the hardware cost is highly appreciated, the BW strategy provides a compromise with less hardware cost than the BB scheme but better performance than the B-ALL scheme.

Figure 8 demonstrates the energy consumption of the eight benchmarks using the 3T1D RF. The energy consumption is further broken down to dynamic, leakage and refresh energy. From the figure, we can see that the 3T1D RF generally delivers much less energy consumption than the SRAM except for the 11nm node, where retention time is so short that too many refresh operations are triggered. In memory arrays, leakage generally contributes to a large portion of the total power consumption. The 3T1D RF delivers smaller leakage energy than the SRAM-based RF and the reduction of leakage energy compensates for the additional refresh energy in most cases, making the 3T1D RF a viable design choice. Our simulation results show more than 20% energy savings above the 16nm node.

For a clearer view of the energy efficiency of the 3T1D RF design with different refreshing strategies, Fig. 9 calculates the averaged ratio of normalized performance over energy for the eight benchmarks, normalized to SRAM efficiency. A higher bar means better efficiency. For the three proposed refreshing strategies, it is clear that the BB strategy is the most energy-efficient one with higher bars under every technology node. It exhibits better efficiency than SRAM even under 11nm nodes where retention time is extremely tight. The B-All scheme is the worst in terms of energy efficiency especially at smaller technology nodes. When

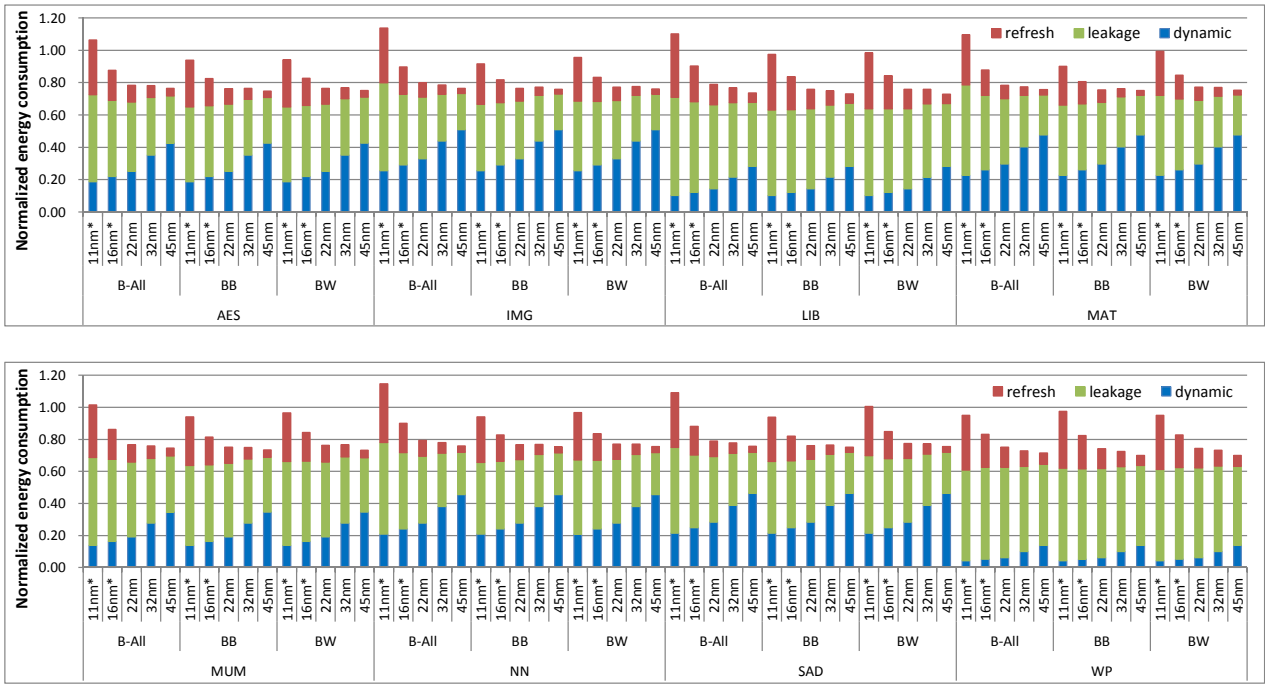


Figure 8: Normalized energy consumption of the 3T1D RF

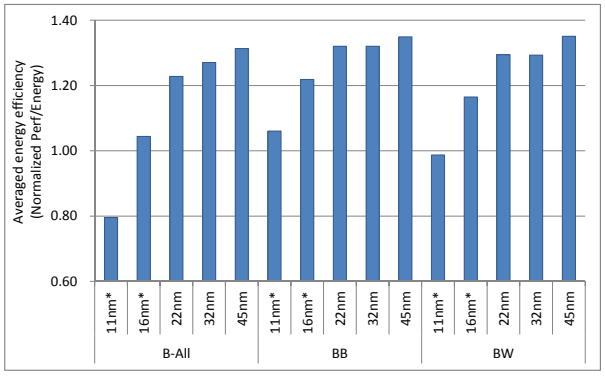


Figure 9: Normalized energy efficiency for the 3T1D RF

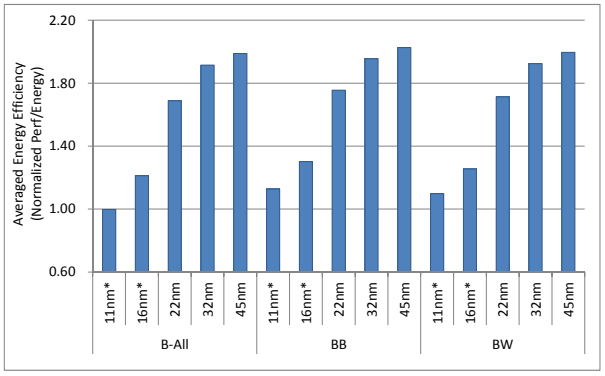


Figure 10: Normalized energy efficiency for the 1T1C RF

comparing with SRAM-based RF, the proposed 3T1D RF generally yields better energy efficiency for most of the cases with up to 1.3X gains for the BB scheme.

6.2 Evaluation of RF Design with 1T1C Cell

In addition to refresh, the 1T1C RF has different behavior due to the destructive read. To mitigate the write-back penalty, we always try to interleave the register bank groups and apply the odd-even warp rescheduling as proposed in section 4.4.

We investigate the 1T1C RF design following the same evaluation metrics used in 3T1D RF, and very similar observations can be made. We plot the energy efficiency for 1T1C RF in Fig. 10, where we find that the 1T1C RF can yield up to 2X better energy efficiency than the SRAM-based RF, even larger than 3T1D design. The superior efficiency comes from the smaller dynamic power due to the smaller memory array size and the much-reduced leakage power as there are fewer leaky transistors in a 1T1C cell.

6.3 Impact of Process Variation

In this section, we study the impact of process variations on the performance and energy in the SRAM and eDRAM implementations. The process variations on the threshold voltage (V_{th}) will greatly impact the RF performance implemented with 6T SRAM. That is because in the convectional SRAM-based RF, the access speed of each cell can deviate from the nominal value due to the threshold variation. Plus,

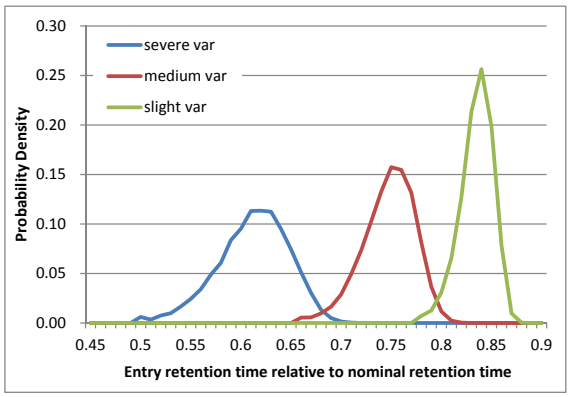


Figure 11: The distribution of retention time for register entries considering process variation

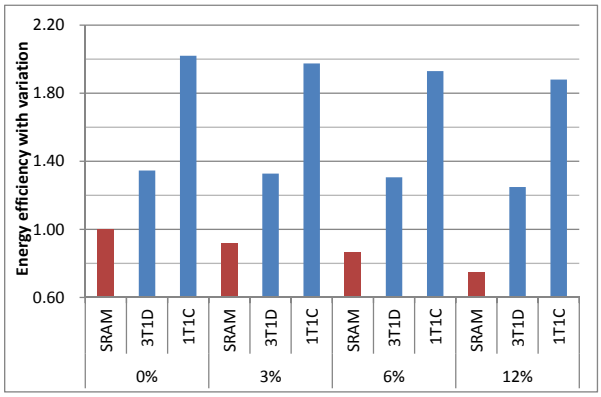


Figure 12: Impact of process variation on the energy efficiency

the operating frequency of the RF after chip fabrication will be determined by the slowest SRAM cell in the memory. If any cell in the bank deviates negatively from the expected operating frequency, the overall frequency and, in turn, the performance will drop.

Process variations also affect the eDRAM-based RF. However, the impact can be viewed as a variation in the retention time as discussed before. For the RF in the presence of process variation, eDRAM cells present varying retention time under the same nominal speed, and the retention time of each register entry will be determined by the worst cell in that entry. It means that the worst cell in the memory array with shortest retention time will not globally degrade the entire RF, but only the register entry it resides. Every register entry can set their refresh counter individually according to their local retention time. Of course in the B-ALL scheme, there is a single refresh counter and the retention time is determined by the worst cell.

In our study, we model the process variation by applying the widely used Monte-Carlo simulation method similar to [3, 11, 2]. We consider the threshold voltage variation as the only source of process variation, which is modeled as a random variable. Three types of variation assumptions are considered in our study. For instance, the medium variation assumes $3\sigma V_{th}/V_{th} = 6\%$ and severe variation assumes $3\sigma V_{th}/V_{th} = 12\%$. For a slight variation, we assume $3\sigma V_{th}/V_{th} = 3\%$. By applying Monte-Carlo HSPICE simulations, the access speed distribution of SRAM cells and the retention time distribution of 3T1D eDRAM cells can be derived. Fig. 11 illustrates how the three V_{th} variation assumptions affect the distribution of retention time for each register entry under the 32nm node, normalized to the nominal retention time. It clearly shows that the retention time of almost all the register entries are shortened from the nominal, and the cells with the worst negative variation can reduce the retention time significantly.

In our study, we investigate the impact of process variation by comparing the energy efficiency of the baseline SRAM-based design against 3T1D and 1T1C eDRAM-based designs under the same amount of threshold variations. We use the BB refresh scheme in this case as it provides the best serviceability. For briefness, we only provide the energy efficiency results at 32nm node in Fig. 12. Note that all the evaluated results are normalized to that of the SRAM-based design without any variations.

From the figure, we can see that our eDRAM-based RF implementation provides much better tolerance to process variations than its SRAM counterpart. The efficiency of SRAM-based RF decreases largely with the increasing degradation on operating frequency under strong variations. This is due to the fact that the speed of SRAM is limited to the slowest cell in the memory. In contrast, the efficiency of eDRAM-based RF also degrades with stronger variations but at a much milder pace. In conclusion, in the presence of the expected strong process variation for the future technology nodes, we envision our proposed eDRAM-based RF can generally provide better energy efficiency and scalability than the conventional SRAM-based designs.

7. RELATED WORK

As an emerging technology, the eDRAM technology has drawn growing interest both from academic and industry. For example, due to its logic compatibility, the 3T1D eDRAM has been considered as one of the most promising alternatives to 6T SRAM cell since its invention by Luk et al. [13]. It can provide lower leakage, higher density and faster memory access, which can meet the demands from large on-chip memory storage in a high performance processor. The potential has been verified by fabricated chips and commercial products. For instance, IBM Power4 and Power5 have used the logic-based eDRAM to implement their L2 caches [21, 18]. An eDRAM L3 cache is also included in Power7 microprocessor [24].

However, in the current GPGPU architectures, the cache is tiny as opposed to its CPU counterparts. The GPGPU L2 cache, typically with a size of 768KByte [10], is relative small when compared to the RF in a GPGPU. The huge amount of concurrent threads and zero-cost context switching request the RF to be as large as possible and the demand is still growing in future GPGPUs. This situation has made the 6T SRAM cell inadequate for on-chip RF implementation. To mitigate this problem, Gebhart et al. propose a register file caching with a two-level thread scheduler to hide memory access latency [8]. Yu proposed a SRAM-DRAM hybrid memory structure for larger RF in GPGPU saving area and power with small performance loss [27]. However, the hybrid

memory cell is custom designed and requires dedicated context switch between SRAM and DRAM. A context-aware warp scheduler is needed to hide the context switching latency. In contrast, our proposed eDRAM replacement uses standard cells, and applies relative simple scheduling and easy implementations to hide the refreshing cost.

The biggest impedient for the wide use of eDRAM in high-performance processors is the periodic refreshing operations. Many researchers have focused on this topic. Generally, these methods try to reduce the number of refresh operations through various solutions. Liang et al. proposed variation-aware refresh operations for eDRAM cells in the L1 cache [11]. Liu et al. exploited memory access patterns to reduce the refresh operations [12]. Stuecheli et al. looked into deferring the refresh operations by an elastic refresh [20]. Alizade et al. proposed to reschedule the refresh operations via dedicated scheduling algorithms [4]. However, these methods are general applicable for CPUs but not targeting at GPGPUs, and thus cannot take the advantage of GPGPU architectures. For example, the heavily banked RF provide unique opportunities for hiding refresh operations.

8. CONCLUSIONS

This paper studies the feasibility of replacing the conventional SRAM with eDRAM in constructing the GPGPU RF. Our study quantitatively shows that the eDRAM with 3T1D and 1T1C structures are both promising alternatives, in terms of performance, energy consumption and area density. To address the refresh issue, we explicitly exploit the distinctive characteristics in GPGPU RF architecture and the real applications. We propose several refresh strategies with intelligent warp rescheduling to minimize the performance penalty introduced by eDRAM refresh operations. Our experiment results demonstrate that even with moderate retention time, the eDRAM replacement can deliver comparable performance to SRAM, lower overall energy consumption, better variation tolerance and significant area reduction.

9. ACKNOWLEDGMENTS

This work is partly supported by the National Natural Science Foundation of China (Grant No. 61202026, 61272099 and 61261160502); 863 Program of China (No. 2011AA01A202); Program of China National 1000 Young Talent Plan; Changjiang Scholars and Innovative Research Team in University (IRT1158, PCSIRT), China.

10. REFERENCES

[1] Predictive Technology Model (PTM). http://ptm.asu.edu/.

[2] B. Cheng, S. Roy and A. Asenov. CMOS 6-T SRAM cell design subject to atomistic fluctuations. *Solid-State Electronics*, 51(4):565 – 571, 2007.

[3] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. In *International Conference on Computer Aided Design.*, pages 900 – 907, Nov. 2003.

[4] M. Alizadeh, A. Javanmard, S.-T. Chuang, S. Iyer, and Y. Lu. Versatile refresh: low complexity refresh scheduling for high-throughput multi-banked eDRAM. In *Proceedings of the 12th ACM SIGMETRICS / PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, pages 247–258, 2012.

[5] J. Andrews and N. Baker. Xbox 360 system architecture. *IEEE Micro*, 26(2):25–37, Mar. 2006.

[6] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt. Analyzing CUDA workloads using a detailed GPU simulator. In *International Symposium on Performance Analysis of Systems and Software*, pages 163–174, 2009.

[7] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De. Parameter variations and impact on circuits and microarchitecture. In *Proceedings of the 40th annual Design Automation Conference*, pages 338–342, 2003.

[8] M. Gebhart, D. R. Johnson, D. Tarjan, S. W. Keckler, W. J. Dally, E. Lindholm, and K. Skadron. Energy-efficient mechanisms for managing thread context in throughput processors. In *Proceedings of the 38th annual international symposium on Computer architecture*, pages 235–246, 2011.

[9] H. M. Haynie, J. M. Turner, J. C. Hanscom, M. Cadigan, N. Hadzic, D. Di Genova, J. Aylward, S. W. Salisbury, P. Sciuto, T. D. Needham, C. E. Bubb, and R. B. Tremaine. IBM system z10 open systems adapter ethernet data router. *IBM Journal of Research and Development*, 53(1):8:1 –8:12, Jan. 2009.

[10] W. Jia, K. A. Shaw, and M. Martonosi. Characterizing and improving the use of demand-fetched caches in GPUs. In *Proceedings of the 26th ACM international conference on Supercomputing*, pages 15–24, 2012.

[11] X. Liang, R. Canal, G.-Y. Wei, and D. Brooks. Process variation tolerant 3T1D-based cache architectures. In *Proceedings of the 40th International Symposium on Microarchitecture*, pages 15–26, 2007.

[12] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu. RADIR: Retention-aware intelligent DRAM refresh. In *Proceedings of the 39th International Symposium on Computer Architecture*, pages 1–12, 2012.

[13] W. Luk, J. Cai, R. Dennard, M. Immediato, and S. Kosonocky. A 3-transistor DRAM cell with gated diode for enhanced speed and retention time. In *VLSI Circuits, 2006 Symposium on*, pages 184 –185, 2006.

[14] R. E. Matick and S. E. Schuster. Logic-based eDRAM: origins and rationale for use. *IBM J. Res. Dev.*, 49(1):145–165, Jan. 2005.

[15] NVIDIA. http://www.geforce.com/hardware/desktop-gpus.

[16] NVIDIA. https://developer.nvidia.com/cuda-toolkit.

[17] Brett W. Coon, John Erik Lindholm, Samuel Liu, Stuart F. Oberman, Ming Y Siu. Operand collector architecture, 2006.

[18] B. Sinharoy, R. N. Kalla, J. M. Tendler, R. J. Eickemeyer, and J. B. Joyner. Power5 system microarchitecture. *IBM Journal of Research and Development*, 49(4.5):505 –521, July 2005.

[19] J. Stokes. AMD to market with CPU/GPU combo chip. http://arstechnica.com/gaming/news/2010/08/microsoft-beatsintel-amd-to-market-with-cpugpu-combo-chip.ars.

[20] J. Stuecheli, D. Kaseridis, H. C.Hunter, and L. K. John. Elastic refresh: Techniques to mitigate refresh

penalties in high density memory. In *Proceedings of the 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 375–384, 2010.

[21] J. M. Tendler, J. S. Dodson, J. S. Fields, H. Le, and B. Sinharoy. Power4 system microarchitecture. *IBM Journal of Research and Development*, 46(1):5 –25, Jan. 2002.

[22] S. Thoziyoor, J. H. Ahn, M. Monchiero, J. B. Brockman, and N. P. Jouppi. A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, pages 51–62, 2008.

[23] W. W. F. Tor M. Aamodt. http://gpgpu-sim.org/manual/.

[24] D. Wendel, R. Kalla, R. Cargoni, J. Clables, J. Friedrich, R. Frech, J. Kahle, B. Sinharoy, W. Starke, S. Taylor, S. Weitzel, S. Chu, S. Islam, and V. Zyuban. The implementation of Power7tm: A highly parallel and scalable multi-core high-end server processor. In *IEEE International Solid-State Circuits Conference*, pages 102 –103, 2010.

[25] N. Whitepaper. Nvidia's next generation CUDA compute architecture: Fermi.

[26] C. Wilkerson, A. R. Alameldeen, Z. Chishti, W. Wu, D. Somasekhar, and S.-l. Lu. Reducing cache power with low-cost, multi-bit error-correcting codes. In *Proceedings of the 37th annual international symposium on Computer architecture*, pages 83–93, 2010.

[27] 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 *Proceedings of the 38th annual international symposium on Computer architecture*, pages 247–258, 2011.