ARGO: Aging-aware GPGPU Register File Allocation

Majid Namaki-Shoushtari[*], Abbas Rahimi[†], Nikil Dutt[*], Puneet Gupta[‡], Rajesh K. Gupta[†]

[*]Department of Computer Science
University of California, Irvine
Irvine, CA 92697-3435
{anamakis, dutt}@ics.uci.edu

[†]Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404
{abbas, gupta}@cs.ucsd.edu

[‡]Electrical Engineering Department
University of California, Los Angeles
Los Angeles, CA 90095-1594
puneet@ee.ucla.edu

ABSTRACT

State-of-the-art general-purpose graphic processing units (GPGPUs) implemented in nanoscale CMOS technologies offer very high computational throughput for highly-parallel applications using hundreds of integrated on-chip resources. These resources are stressed during application execution, subjecting them to degradation mechanisms such as negative bias temperature instability (NBTI) that adversely affect their reliability. To support highly parallel execution, GPGPUs contain large register files (RFs) that are among the most highly stressed GPGPU components; however we observe heavy underutilization of RFs (on average only 46%) for typical general-purpose kernels. We present ARGO, an Aging-awaRe GPGPU RF allOcator that opportunistically exploits this RF underutilization by distributing the stress throughout RF. ARGO achieves proper *leveling* of RF banks through deliberated power-gating of stressful banks. We demonstrate our technique on the AMD Evergreen GPGPU architecture and show that ARGO improves the NBTI-induced threshold voltage degradation by up to 43% (on average 27%), that yields improving RFs static noise margin up to 46% (on average 30%). Furthermore, we estimate a simultaneous reduction in leakage power of 54% by providing sleep states for unused banks.

Categories and Subject Descriptors

C.1.2 [**Multiple Data Stream Architectures (Multiprocessors)**]: Single-instruction-stream, multiple-data-stream processors (SIMD)

General Terms

Reliability, Design, Performance.

Keywords

NBTI, GPGPU, Aging, Register File, Power-gating.

1. INTRODUCTION

General-purpose graphic processing units (GPGPUs) have become popular platforms for executing applications that exhibit a high degree of thread-level parallelism. Nanoscale CMOS technologies permit fabrication of GPGPUs with hundreds of integrated on-chip resources and large RFs, allowing very high computational throughput for a wide range of highly parallel applications. High performance GPGPU execution greatly stresses on-chip resources, compromising system reliability due to aging degradation mechanisms such as negative bias temperature instability (NBTI) [1].

NBTI manifests itself via an increase in the threshold voltage of PMOS transistor when a logic '0' at the gate is applied. This V_{th} drift strongly depends on the amount of time during which a PMOS transistor is *stressed*. On the other hand, when the stress condition is *relaxed*, the aging can be recovered partially, and the V_{th} decreases toward the nominal value [5],[6]. In SRAMs, the value dependence of NBTI is weaker than in logic circuits: given the symmetric structure of a 6T-SRAM cell, one PMOS is always under stress as long as it stores any value so the SRAM cell structure degrades continually [39],[23]. This makes SRAM RFs the most likely candidates for stress and hence reliability failures. On the other hand, to support massive parallelism, GPGPUs feature *very large* RFs to hold the state of each thread making the RF reliability a major concern. But many GPGPU applications exhibit imbalanced utilization of these very large RFs.

To ensure necessary observability of the non-uniform aging degradation, traditionally compact *in situ* NBTI and oxide degradation sensors have been proposed [28]. These sensors enable high-volume data collection to guide dynamic NBTI management schemes and warn of impending device failure. Using NBTI sensors, adaptive guardbanding has been proposed earlier to reduce the otherwise conservative guardbands due to better than worst-case operating conditions [9]. However, these post-silicon reliability techniques rely on *reactive* measures (e.g., error detection and error correction) with inevitable overhead. For controllability, power-gating is deployed as an effective technique to mitigate NBTI-induced aging [10],[11],[17]; power-gating achieves intrinsic protection against NBTI by providing sleep states that spare gates from stress that induce NBTI effects.

The lifetime of the chip is limited by the component that ages the most. Since RFs are stressed continually during execution, they exhibit steady growth in aging and could compromise the lifetime of the chip. Furthermore, since GPGPUs have large RFs, it becomes critical to devise strategies that *heal* aging of RFs. Accordingly, this paper makes the following contributions:

- We observe that *RFs are not uniformly utilized*: unlike uniformly-exercised GPGPU compute units, there is a large kernel-dependent variation in RF utilization. Indeed our studies on a set of fifteen general-purpose kernels show significant kernel-to-kernel variations, with only a 46% average RF utilization. This presents an opportunity to exploit this imbalanced RF utilization for intelligent RF allocation to extend RF (and therefore chip) lifetimes.

- We propose ARGO, an adaptive architectural technique for GPGPU RF allocation that exploits imbalanced RF utilization to ameliorate lifetime degradation by uniformly distributing the stress throughout the RFs, without performance penalty. Unlike traditional *reactive* approaches, ARGO is a *proactive* technique that opportunistically exploits the underutilized portion of RF by proper *leveling*, accomplished through light-weight *virtual sensing* in conjunction with deliberated power-gating of stressful banks.
- We apply ARGO on the AMD Evergreen GPGPU architecture with general-purpose applications written in OpenCL. Experimental results for fifteen general-purpose kernels show the efficacy of ARGO through deliberated power-gating without throughput penalty: ARGO improves NBTI-induced V_{th} degradation by up to 43% (on average 27%), improves RFs static noise margin by up to 46% (on average 30%), and estimates a 54% reduction in leakage power.

The rest of the paper is organized as follows. Section 2 surveys prior work in this specific topic area. Section 3 describes GPGPU architecture. Section 4 covers an overview of NBTI degradation. ARGO is presented in Section 5. In Section 6, we present experimental results followed by conclusions in Section 7.

2. RELATED WORK

Various techniques have been proposed to slow down the aging of processors, using microarchitectural, circuit-level and power gating strategies [12]−[14]. At the microarchitectural level, Colt [12] equalizes the duty cycle ratio and the usage frequency of the functional units in a microprocessor. To mitigate aging effects, it uses a number of measures such as complement mode execution, cache set rotation, and operand identifier swapping schemes. These measures are intrusive and fairly complicated: the complement mode is applied to the whole data path, control path, and storage hierarchy. Wearout-aware compiler-directed register assignment techniques are proposed in [13] to distribute the stress-induced wearout throughout the RF. Even though [13] does not impose architectural overheads and modification, its compiler-based approach is only limited to single-threaded applications. Another aging-aware assignment of RF is also proposed to balance the duty cycle ratio of the internal bits in RF [14]; however, with balanced signal probabilities, there is still an inevitable static noise margin degradation [35],[36]. Kang *et al.* [23] show two orders of magnitude increase in read failure probability even with balanced signal probabilities. Further this technique results in high power and area overhead for very large RFs. Single-core techniques also exploit the underutilization of RFs by putting unused registers into a low-power state or shut-off state through inserting explicit instructions in the application's binary [41],[42]. These techniques not only require intrusive modification of the Instruction Set Architecture (ISA) and compiler, but also still result in uneven stress distribution across the RFs. In contrast, ARGO specifically deploys an unobtrusive aging management technique (that balances RF stress) for GPGPUs executing multi-threaded applications.

Several recent efforts have focused on circuit-level measures to mitigate process variability and aging. To combat the impact of core-to-core frequency variations on GPGPU throughput, Lee *et al.* [15] propose two techniques: run each core at its maximum frequency independently; and disable the slowest cores. These solutions impose non-negligible performance penalty: the first directly diminishes the throughput of a cluster, and the second imposes extra latency for synchronization of cores with different frequencies. Furthermore, these techniques only consider the effects of static process variation on computing cores, and *do not address aging* of GPGPUs which is dynamic in nature. In response to these deficiencies, Rahimi *et al.* [17] propose NBTI-aware compiler-directed scheme that uniformly distributes the stress of instructions with the aim of minimizing aging of GPGPU without any performance penalty. Wang *et al.* [16] propose to compensate process variation-induced SRAM mismatch by exploiting BTI to partially offset variation that can be applied during burn-in test. These techniques are complementary to ARGO, and our aging-aware RF allocation policy with virtual sensing can be superposed on top of these resilient circuit techniques.

Recent approaches have also proposed NBTI-aware power-gating [10],[11],[17] that exploits a circuit's sleep state that is intrinsically immune to aging. It has been used for inactive portion of the cache to meet a given lifetime target [11]. An orthogonal problem is aging of PMOS sleep-transistors which is critical to lifetime of whole system. Calimera *et al.* [18] propose static and dynamic strategies to compensate the aging effects on the sleep transistors by means of body biasing, sleep-transistor oversizing, and equivalent stress-time reduction. Although power-gating can significantly compensate the V_{th} shift due to aging, the benefit is sensitive to the fraction of time that a circuit spends in sleep mode, and thus significant performance degradation must be tolerated to achieve high power-gating factors. In contrast, ARGO arranges GPGPU RF accesses to exploit power-gating without any performance penalty. The objective of our work is to develop an *iso-throughput* dynamic RF allocation policy for compute units in GPGPUs that increases RF expected lifetime without changing the flow of execution.

3. GPGPU ARCHITECTURES AND RFs

Extreme multithreading with fast thread switching in GPGPUs is supported by a large register file that is much larger than the cache holding the execution state of each thread. For example the AMD Radeon HD 5870 GPU has 5 MB of on-chip RF while it only has 160 KB of L1-cache. Similarly, the NVIDIA GTX480 GPU has a 2MB RF, while the shared L1-cache size is only 512 KB. The inversion in sizing between the GPU cache and register file, compared to a traditional CPU memory hierarchy, is a critical GPU microarchitectural feature that is needed for supporting massively parallel execution of multiple threads [27].

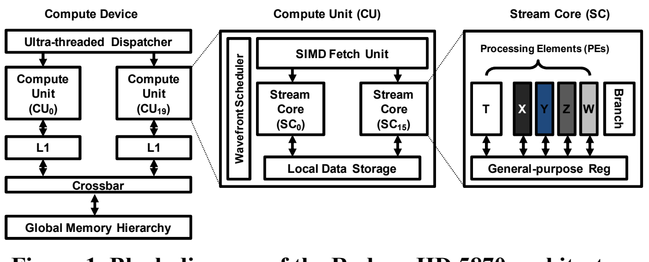

Figure 1. Block diagram of the Radeon HD 5870 architecture

Here we focus on the Evergreen family of AMD GPUs (a.k.a., Radeon HD 5000 series), designed to target not only graphics applications, but also general-purpose data-intensive applications. The Radeon HD 5870 GPU consists of 20 compute units, a global front-end ultra-thread dispatcher, and a crossbar to connect the global memory to the L1-caches [19]. The block diagram of architecture is shown in Figure 1.

Every compute unit has access to a global memory, implemented as a hierarchy of private 8KB L1-caches, and 4 shared 512KB L2-caches. Each compute unit contains a set of 16 stream cores that have access to a shared 32KB local data storage, ***as well as a 256KB general-purpose RF***. Within a compute unit, a shared instruction fetch unit provides the same machine instruction for all stream cores to execute in a single-instruction, multiple-data (SIMD) fashion. Finally, each stream core is a five-way very long instruction word (VLIW) processor capable of issuing up to five floating point scalar operations from a single VLIW that consists of five slots ($slot_X$, $slot_Y$, $slot_Z$, $slot_W$, $slot_T$). Because of this VLIW structure, the general-purpose registers are actually 128-bit wide and hold four 32-bit values, described as the X, Y, Z and W elements.

Applications for HD 5870 are written in OpenCL, comprising a host program and one or more device kernels executing on the GPU device. In the OpenCL programming model, an instance of the OpenCL kernel is called a work-item. Work-items are arranged into work-groups for SIMD execution. The Evergreen architecture supports work-groups containing between 1 to 256 work-items. Since the number of stream cores in a compute unit (16) is lower than the maximum number of work-items in a work-group, Evergreen supports the notion of a wavefront as the quantum unit of scheduling. A wavefront is composed of 64 work-items virtually executing at the same time on the 16 stream cores of a compute unit. Thus a work-group contains up to 4 wavefronts that share execution resources. To manage these resources, a wavefront scheduler dynamically selects wavefronts for execution.

3.1 Register File Utilization

Several factors limit the maximum number of concurrent OpenCL threads that can execute on a compute unit of an AMD GPGPU (called "compute unit occupancy"). Considering the limited physical registers within a compute unit, if a work-item uses many registers, it eventually prevents other wavefronts from being executed concurrently. The number of registers per work-item is exclusively decided at compile time. The compute unit occupancy is also a function of the local memory capacity. When the compute unit allocates a work-group that increases the usage of the local memory, it reduces the total number of wavefronts that can be allocated to the same compute unit. The local memory used by a work-group could be determined at compile-time or run-time [25]. The work-group size also determines the total number of wavefronts that can be allocated. Note that the allocation unit is an entire work-group (set of wavefronts). The work-group size is exclusively decided at run-time by the OpenCL host program [25].

Given a fixed die area, there is an upper bound for all the aforementioned parameters that yields limited compute unit occupancy. Applications are typically limited by registers, local memory usage, or available state preservation capacity in the architecture and as a result not all RF space is utilized for every application. Table 1 shows the variation in utilization of RF space for the Radeon HD 5870 GPGPU while executing different applications listed in the first column. The second column, *#of Registers,* shows the required number of registers for every kernel, while *#of Concurrent WF* shows the maximum possible number of interleaved wavefronts for every kernel. A HD 5870 compute unit cannot preserve state of more than 32 wavefronts at a time. As shown, even for very large set of input parameters, on average 54% of RF is not utilized during the execution of these kernels.

Table 1. Register requirement of different applications

Kernel	#of Registers	#of Concurrent WFs	RF Utilization
Reduction	4	32	50%
BinarySearch	2	32	25%
DwtHaar1D	4	32	50%
BitonicSort	4	8	13%
FastWalshTransform	4	32	50%
FloydWarshall	6	32	75%
BinomialOption	13	16	81%
DiscreteCosineTransform	7	8	22%
MatrixTranspose	3	32	38%
MatrixMultiplication	22	8	69%
SobelFilter	9	28	99%
URNG	6	8	19%
RadixSort	16	1	6%
Histogram	16	2	13%
BlackScholes	19	12	89%

This RF underutilization exists mainly for two categories of kernels:

- Kernels with high demands of local memory: e.g., Histogram, RadixSort, BitonicSort.
- Kernels requiring few registers: e.g., Reduction, BinarySearch, FastWalshTransform. This low demand of registers is achieved by the effective usage of the temporary registers resided in the processing elements.

We note that the kernels preserve these characteristics across a standard set of OpenCL compiler options[1]. We utilize options for math intrinsic, and options that control optimization of performance and code size; however, *#of Registers* is changed slightly by 13% across all kernels. *These observations motivate us to opportunistically exploit the RF underutilization during runtime allocation to improve the aging reliability of GPGPUs.*

4. DEVICE LEVEL NBTI MODELING

We now briefly describe the dynamic NBTI model to show how our architectural approach can combat the NBTI effects. When logic input '0' is applied to the gate of a PMOS transistor ($V_{gs} = -V_{dd}$), the dissociation of $Si-H$ bonds along the silicon oxide interface, causes the generation of interface traps, [1]−[4]. The rate of generation of these traps is accelerated by temperature, and the time of applied stress. The increase in voltage threshold due to this phenomena, $\Delta V_{th\text{-}stress}$, could be modeled with:

$$\Delta V_{th-stress} = (K_v \sqrt{t_{stress}} + \sqrt[2n]{\Delta V_{th-t0}})^{2n} \qquad (1)$$

where t_{stress} is the time that PMOS is under stress; K_v has dependence on electrical field, temperature (T), and V_{dd}; n is the time exponent parameter, and for H_2 diffusion is 1/6; and ΔV_{th-t0} is the initial V_{th} variation of PMOS at time zero due to process variation [5]. When logic input '1' is applied to the gate ($V_{gs} = 0$), the transistor turns off, and H atoms diffuse back, eliminating some of the traps. This process is called the recovery phase that can recover part of the V_{th} shift:

[1] including, '*-cl-single-precision-constant*', '*-cl-opt-disable*', '*-cl-mad-enable*', '*-O1*', '*-O2*', '*-O3*', etc.

$$\Delta V_{th-re\mathrm{cov}} = \Delta V_{th-stress}(1 - \frac{2\xi_1 t_e + \sqrt{\xi_2 C t_{re\mathrm{cov}}}}{(1+\delta)t_{ox} + \sqrt{Ct}}) \quad (2)$$

where t_{recov} is the time under recovery; t_{ox} is the oxide thickness; t_e is the effective oxide thickness; t is the total time; C has temperature dependence; ζ_1, ζ_2, δ constants are defined in [5]. The cycle-by-cycle V_{th} variation at clock cycle i-th is defined in (3), and (4). Let $\Delta V_{th\text{-stress},i}$ and $\Delta V_{th\text{-recov},i}$ be changes in the V_{th} at the end of i-th stress and recovery cycles, respectively:

$$\Delta V_{th-stress-i} = (K_v \sqrt{\alpha T_{clk}} + \sqrt[2n]{\Delta V_{th-re\mathrm{cov},i-1}})^{2n} \quad (3)$$

$$\Delta V_{th-re\mathrm{cov}-i} = \Delta V_{th-stress-i}(1 - \frac{2\xi_1 t_e + \sqrt{\xi_2 C(1-\alpha)T_{clk}}}{2t_{ox} + \sqrt{CiT_{clk}}}) \quad (4)$$

where i = t/T_{clk}; T_{clk} is the time period of one stress-recovery cycle, duty cycle (α) is the ratio of the time spent in stress to the period time [5]. [20] shows that ΔV_{th} is a monotonically increasing function of higher duty cycle (α), t, V_{dd}, T. The NBTI-induced V_{th} shift is also a function of process-dependent parameters, and relatively insensitive to the switching frequency (f) when it is above 100Hz [21]. Stress distribution can tune duty cycle (α) and hence decrease NBTI-induced effects.

4.1 NBTI-induced SNM Degradation

The RF memory cells of GPGPUs are typically built as 6T-SRAM-type cells [27]. As a result of NBTI, the threshold voltage of PMOS transistors inside a memory cell increases with time. The amount of increase can be calculated using (3) and (4). Because of the cell's symmetric structure, regardless of what value the cell holds, one of the PMOS transistors is always under stress.

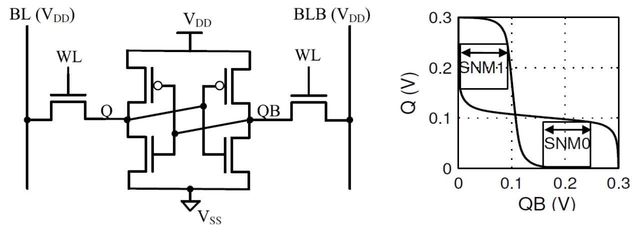

Figure 2. Graphical representation of the SNM for 6T SRAM

The stability of SRAM cells is measured as static noise margin (SNM), and is defined as the minimum dc noise voltage required to change the state of SRAM cell [22]. Typically READ SNM is considered as the factor determining the life-time of the SRAM-type memories which is measured when the word line is on and the cell is being read [23]. SNM can be computed as the side length of a maximum square enclosed between the two static characteristics curves of a SRAM cells (Figure 2). During the time, either of PMOS transistors of the back-to-back inverters is becoming weaker due to the V_{th} shift. Both imbalanced V_{th} shift and the V_{th} shift itself contribute to SNM degradation. Even with balanced signal probabilities, 12 to 16 percent of degradation in the SNM is unavoidable after 5 years [35],[36], thereby increasing reliability failures. This necessitates the proper stress leveling of the SRAM-type cells.

BTI could be exploited during burn-in to compensate initial V_{th} mismatch due to the manufacturing variations [16],[45]. Also, write-margin may improve with BTI as well [44]. This work assumes that BTI worsens the SRAM VCC_{min} (i.e., it is dominated by nominal SNM as opposed to variation induced SNM or write margin).

5. AGING-AWARE REGISTER FILE MANAGEMENT

The key idea of ARGO, our aging-aware RF allocation technique, is to uniformly distribute the stress across the entire physical RF by exploiting the imbalanced utilization. Thus, it needs a means for aging instrumentation. Based on this instrumentation, we can have an estimate of aging while we are allocating (i.e., stressing) various portions of the RF. ARGO requires minor architectural modifications to keep track of the RF allocation (described in Section 5.4). We now describe our aging-aware RF allocation technique.

5.1 Virtual Sensing

ARGO's RF allocation technique requires continual assessment of the impact of NBTI-induced aging on RF. Traditional post-silicon reactive dynamic NBTI management techniques typically employ NBTI sensors [28],[29], that provide ΔV_{th} measurement with 3σ accuracy of 1.23 mV for a wide range of temperatures. However, these sensors incur area as well as power overheads. Thus, our technique relies on a concept of *virtual sensing* where our virtual sensing capability estimates the aging profile of different portions of the RF in a relative manner. It is worth noting that our lightweight sensing ignores manufacturing variability within register file that could be compensated using the proposed technique in [16] during burn-in.

When an OpenCL kernel is launched on the GPGPU, the ultra-threaded dispatcher, as the work-group scheduler, assigns work-groups to any of the 20 available compute units. The ultra-threaded dispatcher does not allocate simultaneous execution of multiple kernels to a compute unit. Consequently, all of the interleaved wavefronts on a compute unit, execute the same type of kernels with different data sets. Figure 3 shows the total execution cycles per work-group for 20 work-groups of three benchmarks. Our observation shows that variation in the execution time of different work-groups of a kernel is less than 8% for a wide range of kernels listed in Table 1.

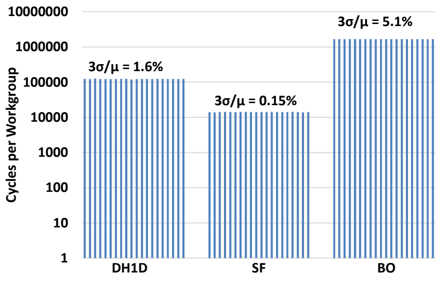

Figure 3. Cycles per work-group

Two architectural features justify this observation: (i) the round robin wavefront scheduler [25], and (ii) the strategy that GPGPUs follow for handling thread divergence[2]. As a result, every work-

[2] It consists in bringing all work-items together through all possible execution paths, while keeping active only those work-items whose conditional execution matches the currently fetched instruction flow.

group allocates a portion of the RF for a fixed amount of time during the execution of a kernel. Because all the cells inside the allocated portion of RF are holding the state of work-groups, they are being aged. The rate of this aging across different allocated portions of RF is similar due to the almost same amount of stress time. This provides a simple virtual sensing metric: *at the granularity of work-groups, various RF portions are aged at the same rate*. Our allocation policy will be defined based on this observation later in Section 5.3.

5.2 RF Organization

Figure 4 shows the block diagram of the large RF inside one compute unit along with the RF allocator and the additional power gaters. Note that the baseline RF does not include power gaters.

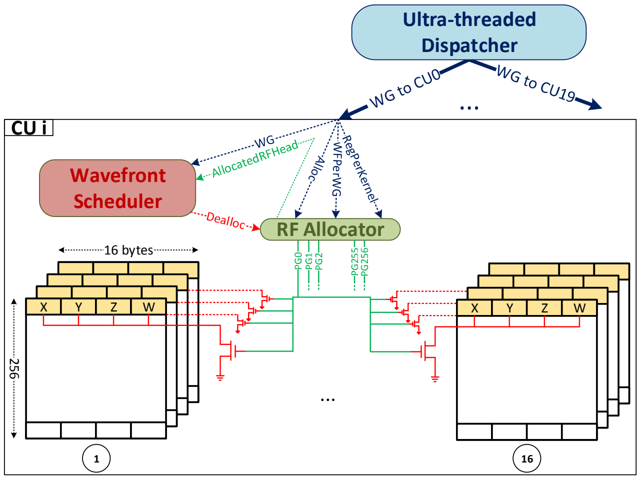

Figure 4. Sliced RF organization

The RF is logically partitioned into 16 slices where each slice serves one stream core inside a compute unit. For higher throughput, GPGPUs normally have register buffers attached to their stream cores. This enables a wavefront to read all of its registers ahead of time, while another wavefront is being executed, and then starts its execution. There are 64 work-items inside a wavefront, with groups of four executed on the same stream core in a time multiplexed fashion. Every slice of the RF is horizontally banked into 256 banks, with each 1-KB on a separate power domain. Each bank can be independently power gated providing sleep states that spare its cells from stress.

5.3 Allocation Policy

Typically, the RF is allocated at the granularity of an entire work-group. As Figure 4 shows, the ultra-threaded dispatcher maps a work-group to an available compute unit. Meanwhile, it asks the RF allocator for a portion of the RF pool. Then, the work-group data along with the head of allocated space will be inserted into the wavefront scheduler's queue. From then on, each work-item inside the new work-group has its own share of the RF. Given the head of RF space for the work-group and index of a work-item in that work-group, the logical to physical register mapping will be performed during execution time. However, the manner in which the RF allocator works greatly impacts on durability of RF banks. We describe the existing aging-oblivious allocator, followed by our ARGO scheme.

5.3.1 Aging-oblivious Allocation Scheme

Figure 5 outlines the existing aging-oblivious RF allocator that, allocates the first free portion of the RF for the incoming work-group without considering the aging profile of each RF bank.

The RF allocator receives two commands: *Allocate,* which comes from the ultra-threaded dispatcher, and *Deallocate,* which is driven by the wavefront scheduler. The number of registers per kernel (*RegPerKernel*), which has been determined offline by the compiler as a *metadata*, is given to the RF allocator by the ultra-threaded dispatcher. In addition, the RF allocator also needs to know the number of wavefronts inside the work-group (*WFPerWG*) [Line 1]. The RF allocator returns as output the head of the allocated RF portion via *AllocatedRFHead* [Line 2]. To keep track of allocations, the RF allocator saves the head of allocated portions in *HeadForWGs* [Line 3] and status of each portion in *PortionStatus* [Line 4]. Because the maximum number of work-groups per compute unit, *Max#WGperCU*, is limited to 8 on HD 5870 (it is a constant number for all GPGPUs in the Evergreen family), and also work-groups are from the same type, the RF allocator in this scheme can statically determine 8 possible portions and save their head indexes in *HeadForWGs* [Lines 5-7].

```
00: AGING-OBLIVIOUS-RF-ALLOCATOR ()
01: INPUTS: Allocate, Deallocate, HeadDealloc, RegPerKernel, WFPerWG
02: OUTPUTS: AllocatedRFHead

03: VARIABLE: HeadForWGs[1…Max#WGperCU] /* Holds WG heads */
04: VARIABLE: PortionStatus [1…Max#WGperCU] /* Initially all portions are free */

05: LOOP N: 1…Max#WGperCU
06:    HeadForWGs = RegPerKernel × WFPerWG × (N-1);
07: ENDLOOP

08: IF (Allocate)
09:    AllocatedPortion = FindFirstFreeRFPortion();
10:    AllocatedRFHead = HeadForWGs [AllocatedPortion];
11:    PortionStatus [AllocatedPortion] = 1;
12: ENDIF

13: IF (Deallocate)
14:    DeallocatedPortion = FindPortionIndex(HeadDealloc);
15:    PortionStatus [DeallocatedPortion] = 0;
16: ENDIF
```
Figure 5. Aging-oblivious RF allocation procedure

While copying the work-group data to the wavefront queue, the ultra-threaded dispatcher also requests a portion of the RF by issuing *Allocate* for the RF allocator. The aging-oblivious allocator responds to this command [Lines 8-12] by examining *PortionStatus* for the first-indexed free portion [Line 9]. The head of that portion will be returned as the head of the allocated space [Line 10]. Then, that portion will be marked as occupied [Line 11]. Once the allocation is completed, the wavefront scheduler marks the work-group as a ready work-group and schedules it for execution. Finally, when the work-group finishes its execution, the wavefront scheduler asks the RF allocator for deallocating the work-group's RF portion [Lines 13-16].

For a large category of applications that utilize only part of RF (as discussed in Section 3.1), the simple aging-oblivious allocator results in more accesses to lower-indexed portions of the RF rather than higher-indexed portions.

5.3.2 ARGO Scheme

In contrast to the aging-oblivious scheme, ARGO responds to the allocation requests by greedily choosing portions of RF with lower aging based on the virtual sensing as described in Section 5.1.

As shown in Figure 6, ARGO requires the same architectural interface [Lines 1-2] as the aging-oblivious allocator. However, in contrast to the aging-oblivious scheme, ARGO always returns the *least recently used* portion for allocation [Line 5], instead of returning the first free portion of the RF. ARGO assumes all the manufacturing variation induced SRAM mismatches are compensated during burn-in using the technique in [16]. Therefore, to simplify tracking of the least recently used RF region without requiring the aging sensors and searching the sensor data, we *rotate* the allocated space at a constant rate (once per work-group). Because of this constant rate rotation, the next candidate is always the portion right after recently allocated space of RF [Line 6]. Once the request is received, ARGO wakes up the newly allocated portion [Line 7]. When all of the wavefronts of a work-group have finished their execution, the wavefront scheduler informs the RF allocator by issuing the *Deallocate* command and providing the head of allocated RF space for that specific work-group via *HeadDealloc*. The RF allocator then shuts off the corresponding memory blocks through *PG* signals and puts them into recovery mode [Lines 9-11].

```
00: ARGO ()
01: INPUTS: Allocate, Deallocate, HeadDealloc, RegPerKernel, WFPerWG
02: OUTPUTS: AllocatedRFHead, PG[1…#ofBanks]

03: VARIABLE: NextAllocCandidate /* Initially all blocks are power-gated */

04: IF (Allocate)
05:     AllocatedRFHead = NextAllocCandidate;
06:     NextAllocCandidate = NextAllocCandidate + RegPerKernel × WFPerWG;
07:     Wake-up register space [AllocatedRFHead…NextAllocCandidate - 1];
08: ENDIF

09: IF (Deallocate)
10:     Turn-off register space [HeadDealloc…HeadDealloc + (RegPerKernel ×
         WFPerWG) -1];
11: ENDIF
```

Figure 6. ARGO procedure

5.4 ARGO Overhead

We now discuss the overheads incurred for the hardware implementation of ARGO. To quantify ARGO's performance, area, and power overheads, we implemented both ARGO and the baseline aging-oblivious allocators using Verilog RTL. Then both allocator modules were synthesized using *Synopsys Design Compiler* with TSMC 45-nm general purpose technology.

Performance Overhead: ARGO does not impose any performance penalty thanks to its careful single-cycle implementation. ARGO and the aging-oblivious allocator meet the target frequency of 800MHz at the worst-case corner. Regarding the wake-up latency from the sleep state, recent efforts [32],[33],[43] have shown the latency is less than ten cycles for a power-gated cell. Due to availability of other ready wavefronts in the queue, these ten cycles are effectively hidden. Note that these ten cycles are not in the critical access path; the allocation portions are rotated at the mapping stage and once a work-group is mapped to a compute unit, the logical to physical mapping does not change for the whole execution time of that work-group.

Area Overhead: Hardware implementation of the ARGO module incurs a relative area overhead of 2.1× compared to the aging-oblivious module. The other area overhead is due to addition of the sleep transistors which is < 1% based on an analysis with an industrial memory compiler [40]. Note that we need an address translator for each compute unit. This address translator takes head of allocated space for work-group, index of the current wavefront, and the register number and does the job of logical to physical mapping. However this address translator is not an overhead of our approach: it is also required in the aging-oblivious scheme for the logical to physical address mapping [37]. To hide this latency, two separate wavefronts execute in an interleaved fashion: the first wavefront accesses the RF, while the other executes; and then they switch.

Power Overhead: ARGO consumes 0.92mW at the operating temperature of 125°C with 800MHz clock frequency. This incurs a power overhead of 74% in comparison with the aging-oblivious module (0.56mW). The power overhead of power-gater transistors is negligible. However, these power overheads could be efficiently compensated by the side benefit of power savings in ARGO as presented in Section 6.2.6.

6. EXPERIMENTAL RESULTS

Our goal is to show to what extent ARGO's opportunistic actions can improve reliability and extend the lifetime of the RF for a wide range of applications and different operational conditions.

6.1 Experimental Setup

We have enhanced Multi2Sim [25][26], a cycle-accurate simulation framework − a CPU-GPU model for heterogeneous computing targeting AMD Evergreen ISA – with the logical to physical mapping of register accesses in order to collect statistical data of physical RF accesses. We have also modified the framework to support the microarchitectural modeling proposed in this work. The kernels of AMD APP SDK 2.5 [24], with the large parameters listed in Table 2, are executed on the simulator. To be conservative, we choose the parameters so as to put the highest possible load on compute devices of HD 5870 while executing the corresponding benchmark.

For V_{th} and SNM measurements, we use Predictive Technology Model (PTM) 65nm, and 45nm Bulk transistor models [30] and the NBTI modeling discussed in Section 4. The SNM is numerically extracted by using the graphical method described in [22]. Using *Synopsys HSPICE*, the characteristics of the two inverters are plotted into a two coordinate systems where the axes are rotated of 45º relative to each other. The maximum and minimum values of the resulting curve represent the side length of the maximum and minimum squares nested between the two static characteristics, namely the SNM values. The SNM with the minimum absolute value represents the actual SNM of the SRAM cell.

Table 2. AMD APP SDK 2.5 kernels with selected parameters

Kernel		Parameter
Reduction	*Rdn*	N=4000000
BinarySearch	*BSe*	N=50000000
DwtHaar1D	*DH1D*	N=500000
BitonicSort	*BSo*	N=10000
FastWalshTransform	*FWT*	N=100000
FloydWarshall	*FW*	N=1000
BinomialOption	*BO*	N=600
DiscreteCosineTransform	*DCT*	X=600, Y=600
MatrixTranspose	*MT*	X=800, Y=800
MatrixMultiplication	*MM*	X=1000, Y=1000, Z=1000
SobelFilter	*SF*	default input file
URNG	*URNG*	default input file
RadixSort	*RS*	N=600000
Histogram	*HS*	X=2500, Y=2500
BlackScholes	*BSc*	N=3000

6.2 Simulation Results

6.2.1 *Improvement in ΔV_{th}*

Figure 7 shows the improvements in V_{th} drift with regard to the aging-oblivious scheme after five years of execution. We observe worst-case V_{th} degradation of 22% (19%) for 45nm (65nm) technology, when the aging-oblivious allocator is used. Our approach can improve shift in V_{th} up to 43% for 65nm technology and up to 46% for 45nm technology. The average improvement is 26% (27%) for 45nm (65nm) technology. There is no opportunity for the SobelFilter benchmark since the limiting parameter in the occupancy calculator for this benchmark is the number of available registers. Therefore almost all the RF space (98.88%) is being used when the kernel puts the highest possible load on the HD 5870. Because of that, our approach is not effective in this case, although it does not incur any performance overhead. From now on, we consider SobelFilter as the representative of baseline for comparison purpose.

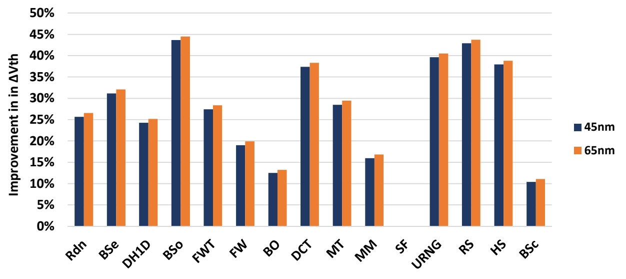

Figure 7. ΔV_{th} improvement after 5 years of execution

6.2.2 *Improvement in SNM Degradation*

Figure 8 shows the SNM improvements after five years of execution. As the comparison of Figure 7 and Figure 8 suggests, improvements in SNM and V_{th} show the same trend. Indeed, this has been analytically proved in [23] that SNM sensitivity to V_{th} degradation (dSNM/dV_{th}) retains near a constant value throughout the whole lifetime.

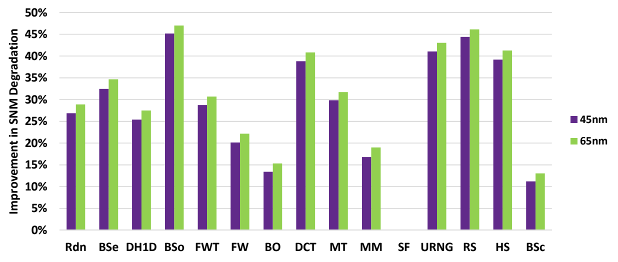

Figure 8. Read SNM improvement after 5 years of execution

6.2.3 *Trend of SNM Degradation*

The lifetime of an SRAM is typically defined as the time after which the SNM of a cell has decreased by 20% [38]. Figure 9 shows the trend of SNM degradation over the course of five years. The baseline (SF) will fail after 3.2 years in 45nm technology. Similar aging rate has been reported in [38]. ARGO facilitates all other benchmarks to reliably execute for at least 5 years. ARGO can improve absolute value of SNM up to 9.8%. This is significant compared to the 20% SNM guardband for NBTI that is needed if no adaptation is done.

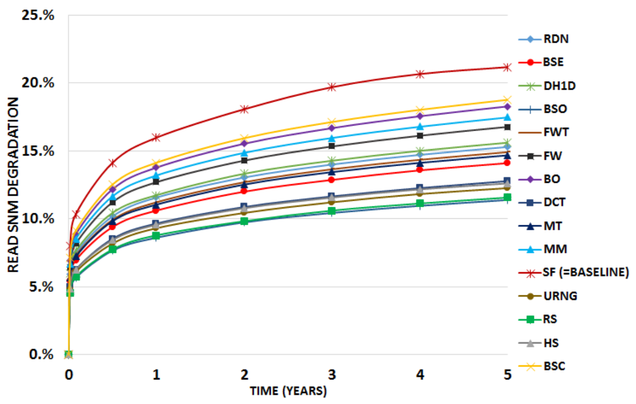

Figure 9. Trend of SNM degradation (45nm)

6.2.4 *Various Operational Conditions*

To quantify achievable gain of employing ARGO in different operational conditions, we have conducted experiments where the cell voltage and temperature are varied. Figure 10 shows the percentage of improvement in SNM degradation in four different operational conditions, summarized below:

- Keeping the temperature constant, achievable gain is the same while varying the cell voltage from 0.9 to 1.
- Changing the operating temperature from 25°C to 110°C, the achievable gain will approximately double.
- The achievable gain is consistent across a wide range of timing scales thanks to the adaptiveness of ARGO.

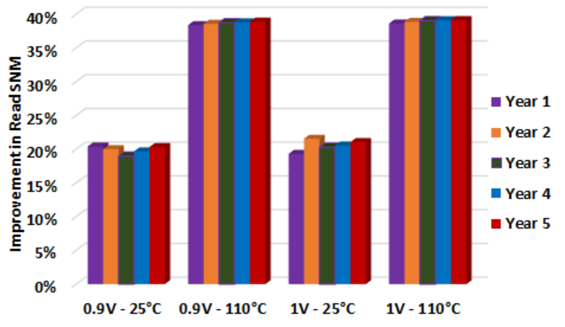

Figure 10. Achievable SNM improvement in various conditions for BSe benchmark

6.2.5 *Recovery Time vs. Bank Size*

If the overhead of power gating logic is of concern, we can increase the bank size, thereby reducing the power gater addressing logic power and area overhead. The trade-off is illustrated in Table 3 by showing worst case recovery time (i.e., the time that bank remain power gated) for different bank sizes. Remarkably, in most cases we can achieve the same recovery percentage without any performance overhead with regard to case of 1K banks. The same recovery percentage comes from the well-organized RF; each row of RF corresponds to one register for whole work-items inside a wavefront. In many cases, *wavefronts per work-group* × *number of required registers* is already a multiple of bank size. Therefore there is no difference between the power-gating at that granularity or at a lower granularity. The results suggest to bank up the RF at the granularity of 2K or 4K where the recovery percentage is still comparable with case of 1K banks.

Table 3. Worst case recovery time vs. bank size

Kernel	Recovery Time (%)			
	1K	2K	4K	8K
Rdn	48%	48%	48%	48%
BSe	63%	63%	63%	63%
DH1D	44%	44%	44%	44%
BSo	87%	87%	87%	87%
FWT	53%	53%	53%	53%
FW	29%	29%	29%	29%
BO	13%	13%	13%	8%
DCT	77%	73%	73%	73%
MT	56%	56%	56%	42%
MM	21%	21%	14%	14%
SF	0%	0%	0%	*
URNG	81%	81%	75%	75%
RS	86%	86%	86%	86%
HS	78%	78%	78%	78%
BSc	9%	9%	9%	4%

* Bank size=8K cannot be selected since it results in performance degradation

6.2.6 *RF Leakage Power Reduction Estimation*

At a first order of approximation, SRAM leakage power is proportional to its size [31]. Figure 11 shows the achievable reduction in leakage power of RF for different benchmarks: an average of 54% and up to 94%. This power saving compensates the power overheads, except for the SF benchmark.

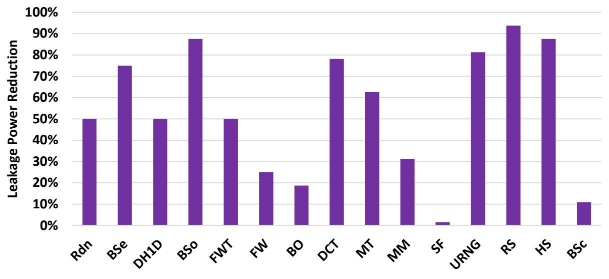

Figure 11. Estimated reduction in register file leakage power

6.3 Discussion on Wakeup Latency

When a RF portion is coming out of the sleep mode, the switch cells are turned on to supply power to the cells. The simultaneous switching current of charging the design to a full power-on state might be too high to tolerate in some cases. A practical method to control the wakeup rush current is to implement the switch cells in a daisy chain where the switch cells are driven by a buffer chain and turned on gradually in the delayed sequence [34]. Our analysis on a commercial 45nm memory compiler shows that it could be up to 100 cycles depending of the rush current that can be tolerated. Our technique is still applicable in that scenario. For that purpose we modify RF allocator in a manner that performs the rotation in intervals of 10's of seconds (which can completely amortize 100 cycles wakeup latency) instead of rotating the allocations once per work-group. Since in long term the fraction of time that cells are in recovery mode does not change, this leads to same V_{th} improvement. Also it simplifies RF allocator and power gating logic, therefore reduces their area and power overheads. Instead we need a watch-dog timer per each compute unit to count for intervals that incur area and power overheads.

7. CONCLUSION AND FUTURE WORK

We proposed ARGO, an architectural technique suitable for GPGPUs parallel applications, to uniformly distribute the stress within RFs, with the aim of improving the NBTI-induced degradation. ARGO not only improves the RF's static noise margin up to 46% (on average 30%), but also incurs no throughput penalty over fifteen general-purpose kernels execution. This is achieved by leveraging the underutilized portion of RFs, and deliberated power-gating of stressful banks. Moreover, leakage power is reduced by 54% thanks to power-gating of unused banks.

Ongoing work is focused on generalizing the proposed approach on other memory subsystems, and exposing the aging characterization to OpenCL runtime software for obtaining a favorable tradeoff between performance and lifetime.

8. ACKNOWLEDGMENTS

This research was partially supported by NSF Variability Expedition Grant Numbers CCF-1029783 and CCF-1029030.

9. REFERENCES

[1] G. Chen, M. F. Li, C. H. Ang, J. Z. Zheng, and D. L. Kwong, "Dynamic NBTI of p-MOS Transistors and its Impact on MOSFET Scaling," *IEEE Electron Device Letters*, vol. 23, no. 12, pp. 734-736, December 2002.

[2] G. Chen, K. Y. Chuah, M. F. Li, D. S. Chan, C. H. Ang, J. Z. Zheng, Y. Jin, and D. L. Kwong, "Dynamic NBTI of PMOS Transistors and its Impact on Device Lifetime," Proc. *IEEE IRPS*, pp. 196-202, 2003.

[3] K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, S. R. Nassif, E. J. Nowak, D. J. Pearson, and N. J. Rohrer, "High-performance CMOS Variability in the 65-nm Regime and Beyond," *IBM Journal of Research and Development*, vol. 50, no. 4.5, pp.433-449, July 2006.

[4] S. Chakravarthi, A.T. Krishnan, V. Reddy, C.F. Machala, and S. Krishnan, "A Comprehensive Framework for Predictive Modeling of Negative Bias Temperature Instability," Proc. *IEEE IRPS*, pp. 273-282, 2004.

[5] W. Wang, S. Yang, S. Bhardwaj, S. Vrudhula, F. Liu, and Y. Cao, "The Impact of NBTI Effect on Combinational Circuit: Modeling, Simulation, and Analysis," *IEEE Trans. on VLSI Systems*, vol. 18, no. 2, pp. 173-183, February 2010.

[6] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive Modeling of the NBTI Effect for Reliable Design," Proc. *IEEE CICC*, pp. 189-192, 2006.

[7] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "An Analytical Model for Negative Bias Temperature Instability," Proc. *IEEE/ACM ICCAD*, pp. 493-496, 2006.

[8] M. Gebhart, S. W. Keckler, B. Khailany, R. Krashinsky, and W. J. Dally, "Unifying Primary Cache, Scratch, and Register File Memories in a Throughput Processor," Proc. *IEEE/ACM MICRO*, pp. 96-106, 2012.

[9] A. Rahimi, L. Benini, and R. K. Gupta, "Hierarchically Focused Guardbanding: An Adaptive Approach to Mitigate PVT Variations and Aging," Proc. *IEEE/ACM DATE*, pp. 1695-1700, 2013.

[10] A. Calimera, E. Macii, and M. Poncino, "NBTI-aware Power Gating for Concurrent Leakage and Aging Optimization," Proc. *IEEE/ACM ISLPED*, pp. 127-132, 2009.

[11] M. Loghi, H. Mahmood, A. Calimera, M. Poncino, and E. Macii, "Energy-optimal Caches with Guaranteed Lifetime," Proc. *IEEE/ACM ISLPED*, pp. 141-146, 2012.

[12] E. Gunadi, E. Gunadi, A. A. Sinkar, N. S. Kim, and M. H. Lipasti, "Combating Aging with the Colt Duty Cycle Equalizer," Proc. *IEEE/ACM MICRO*, pp. 103-114, 2010.

[13] F. Ahmed, M. M. Sabry, D. Atienza, and L. Milor, "Wearout-aware Compiler-directed Register Assignment for Embedded Systems," Proc. *IEEE ISQED*, pp.33-40, 2012.

[14] S. Wang, T. Jin, C. Zheng, and G. Duan, "Low Power Aging-Aware Register File Design by Duty Cycle Balancing," Proc. *IEEE/ACM DATE*, pp. 546-549, 2012.

[15] J. Lee, P. P. Ajgaonkar, and N. S. Kim, "Analyzing Throughput of GPGPUs Exploiting Within-die Core-to-core Frequency Variation," Proc. *IEEE ISPASS*, pp.237-246, 2011.

[16] J. Wang, S. Nalam, Z. Qi, R. W. Mann, M. Stan, and B. H. Calhoun, "Improving SRAM Vmin and Yield by Using Variation-aware BTI Stress," Proc. *IEEE CICC*, pp.1-4, 2010.

[17] A. Rahimi, L. Benini, and R. K. Gupta, "Aging-aware Compiler-Directed VLIW Assignment for GPGPU Architectures," Proc. *IEEE/ACM DAC*, 2013.

[18] A. Calimera, E. Macii, and M. Poncino, "Design Techniques for NBTI-Tolerant Power-Gating Architectures," *IEEE Trans. on Circuits and Systems II*, vol.59, no.4, pp.249-253, April 2012.

[19] AMD Corporation. ATI Radeon HD 5870 Graphics

[20] W. Wang, Z. Wei, S. Yang, and Y. Cao, "An Efficient Method to Identify Critical Gates under Circuit Aging," Proc. *IEEE/ACM ICCAD*, pp.735-740, 2007.

[21] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive Modeling of the NBTI Effect for Reliable Design," Proc. *IEEE CICC*, pp. 189-192, 2006.

[22] E. Seevinck, F. List, and J. Lohstroh, "Static-noise Margin Analysis of MOS SRAM Cells," *IEEE J. Solid-State Circuits*, vol. SSC-22, no. 5, pp. 748–754, October 1987.

[23] K. Kang, H. Kufluoglu, K. Roy, and M. A. Alam, "Impact of Negative-Bias Temperature Instability in Nanoscale SRAM Array: Modeling and Analysis," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 10, pp. 1770- 1781, 2007.

[24] AMD Accelerated Parallel Processing (APP) SDK 2.5. Available: *http://developer.amd.com/gpu/AMDAPPSDK*

[25] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. R. Kaeli, "Multi2Sim: a Simulation Framework for CPU-GPU Computing," Proc. *ACM PACT*, pp. 335-344, 2012.

[26] Multi2Sim [Online]. Available*: http://www.multi2sim.org*

[27] M. Abdel-Majeed and M. Annavaram, "Warped Register File: A Power Efficient Register File for GPGPUs," Proc. *IEEE HPCA*, 2013.

[28] P. Singh, E. Karl, D. Blaauw, and D. Sylvester, "Compact Degradation Sensors for Monitoring NBTI and Oxide Degradation," *IEEE Trans. on VLSI Systems*, vol. 20, no. 9, pp. 1645-1655, September 2011.

[29] P. Singh, E. Karl, D. Sylvester, and D. Blaauw, "Dynamic NBTI Management Using a 45 nm Multi-Degradation Sensor," *IEEE Trans. on Circuits and Systems*, vol. 58, no. 9, pp. 2026-2037, September 2011.

[30] PTM, "Predictive Technology Model," in Nanoscale Integration and Modeling (NIMO) Group at ASU, Available: *http://ptm.asu.edu*

[31] S. Kaxiras, S. Kaxiras, Z. Hu, and M. Martonosi, "Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power," Proc. *IEEE/ACM ISCA*, pp. 240-251, 2001.

[32] Y. Wang, S. Roy, and N. Ranganathan, "Run-time Power-gating in Caches of GPUs for Leakage Energy Savings," Proc. *IEEE/ACM DATE*, pp. 300-303, 2012.

[33] A. Kahng, S. Kang, T. Rosing, and R. Strong, "TAP: Token-based Adaptive Power Gating," Proc. *IEEE/ACM ISLPED*, pp. 203-208, 2012.

[34] K. Shi and J. Li, "A Wakeup Rush Current and Charge-up Time Analysis Method for Programmable Power-gating Designs," Proc. *IEEE SoCC*, pp. 163-165, 2006.

[35] A. Calimera, E. Macii, and M. Poncino, "Analysis of NBTI-induced SNM Degradation in Power-gated SRAM Cells," Proc. *IEEE ISCAS*, pp. 785-788, 2010.

[36] N. Gong, S. Jiang, A. Challapalli, M. Panesar, and R. Sridhar, "Variation-and-aging Aware Low Power Embedded SRAM for Multimedia Applications," Proc. *IEEE SoCC*, pp. 21-26, 2012.

[37] P. Xiang, Y. Yang, M. Mantor, N. Rubin, and H. Zhou, "Many-Thread Aware Instruction-level Parallelism: Architecting Shader Cores for GPU Computing," Proc. *ACM PACT,* pp. 449-450, 2012.

[38] A. Calimera, M. Loghi, E. Macii, and M. Poncino, "Partitioned Cache Architectures for Reduced NBTI-induced Aging," Proc. *IEEE/ACM DATE*, pp. 938-943, 2011.

[39] H. Yang, S.Yang, W. Hwang, and C. Chuang, "Impacts of NBTI/PBTI on Timing Control Circuits and Degradation Tolerant Design in Nanoscale CMOS SRAM," *IEEE Trans. on Circuits and Systems*, vol. 58, no. 6, pp. 1239-1251, June 2011.

[40] ARM Memory IP: *www.arm.com*

[41] H. Tabkhi and G. Schirner, "AFReP: Application-guided Function-level Register file Power-gating for Embedded Processors," Proc. *IEEE/ACM ICCAD*, pp. 302-308, 2012.

[42] J. L. Ayala, A. Veidenbaum, and M. López-Vallejo, "Power-aware Compilation for Register File Energy Reduction," *Springer International Journal of Parallel Programming*, vol. 31, no. 6, pp. 449-465, December 2003.

[43] E. Pakbaznia and M. Pedram, "Design of a Tri-modal Multi-threshold CMOS Switch with Application to Data Retentive Power Gating," *IEEE Trans. on VLSI Systems*, vol. 20, no. 2, pp. 380-385, February 2012.

[44] J.C. Lin, A. Oates, H. Tseng, Y. Liao, T. Chung, K. Huang, P. Tong, S. Yau, and Y. Wang, "Prediction and Control of NBTI – Induced SRAM V_{ccmin} Drift," *Proc. IEEE IEDM,* pp. 1-4, 2006.

[45] C. T. Boon, V. Balakrishnan, Y. Cao, and P. Gupta, "Extended Burn-in for V_{th} Variation Reduction Using NBTI," *Proc. IEEE DFM&Y*, pp. 25-28, 2009.