Mitigating NBTI Degradation on FinFET GPUs through Exploiting Device Heterogeneity

Ying Zhang, Sui Chen, Lu Peng, Shaoming Chen
Division of Electrical and Computer Engineering
School of Electrical Engineering and Computer Science
Louisiana State University
{yzhan29, csui1, lpeng, schen26}@lsu.edu

***Abstract*—Recent experimental studies reveal that FinFET devices commercialized in recent years tend to suffer from more severe NBTI degradation compared to planar transistors, necessitating effective techniques on processors built with FinFET for endurable operations. We propose to address this problem by exploiting the device heterogeneity and leveraging the slower NBTI aging rate manifested on the planar devices. We focus on modern graphics processing units in this study due to their wide usage in the current community. We validate the effectiveness of the technique by applying it to the warp scheduler and demonstrate NBTI degradation is considerably alleviated with slight performance overhead.**

Keywords— NBTI, FinFET, reliability, heterogeneity

I. INTRODUCTION

As we shift into the deep submicron era, innovative materials and device architectures is becoming ever demanding to continue the trend toward smaller and faster transistors. Among all candidates in investigation, the Fin field-effect-transistor (FinFET) stands as one of the most promising substitutes for traditional devices at the ensuing technology nodes, since it presents several key advantages over its planar counterpart [1][13][24][26]. By wrapping the conducting channel with a thin vertical "fin" which forms the body of the device, the gate is coupled tighter with the channel, increasing the surface area of the gate-channel interface and allowing much stronger control over the conducting channel [1]. This effectively relieve the so-called short channel effects (SCE) that are observed on planar transistors manufactured with sub-32nm technology, which in turn implies that FinFET device can provide superior scalability in the deep submicron regime [1].

Another cornerstone motivating the realization of FinFET is the potential performance gain. FinFET transistors can be designed with lower threshold voltage (Vt) and operate with higher drive current, leading to faster switching speed compared to conventional planar devices [1]. Released documents from industry demonstrate that the FinFET transistor persistently demonstrates shorter delay than the planar one while the support voltage is varying, enabling the design and manufacturing of faster processors. Public documents from leading manufacturers also show that the FinFET structure is capable of largely decreasing leakage when the transistor is off [1]. Recently, the Ivy Bridge [2] and Haswell central processing units [3] released by Intel have commercialized this structure (i.e., referred to as "Tri-gate transistor" by Intel), which is also expected to be adopted by other semiconductor manufacturers on their upcoming products [7].

Nonetheless, FinFET is not an impeccable replacement of traditional devices as it raises many challenges to the current industry. One of the most daunting conundrums is the increasing aging rate caused by negative bias temperature instability (NBTI). Recent experimental studies demonstrate that FinFET transistors are more vulnerable to NBTI, leading to a shorter lifetime than a planar device [21][41]. The NBTI aging rate is evaluated by the increase of delay on the critical path after a certain amount of service time. A chip is considered as failed when the delay increment exceeds a pre-defined value after which the timing logic of the processor cannot function correctly. Under the same operation condition, the FinFET device is observed to degrade much faster than the planar counterpart, implying a significantly reduced service lifespan of the target processor. This clearly spurs the development of new techniques to circumvent this problem and prolong the lifetime of FinFET-made processors.

Fortunately, a brief comparison between the main features of FinFET and planar devices sheds some light on alleviating the NBTI effect on future processors. By effectively exploiting the device heterogeneity and leveraging the higher NBTI immunity of planar transistors, the aging of the FinFET structures can be largely suppressed. In this paper, we propose a technique built on top of this principle to improve the durability of FinFET processors. In general, our technique is implemented by replacing an existing structure with a planar-device equivalent. Along with minor modifications at the architectural level, our proposed technique is essentially transferring the "aging stress" from the vulnerable FinFET components to the more NBTI-tolerable planar structures, which in turn lower down the temperature on the structure in study, and thus considerably mitigate the NBTI degradation. Note that the proposed scheme is practically feasible because of the good compatibility between the FinFET and planar process technology [12][18][20].

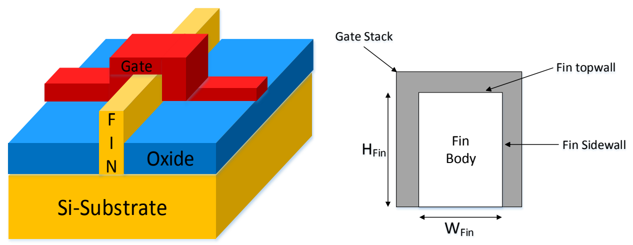
Considering that the general-purpose graphics processing unit is becoming an increasingly important component in a wide spectrum of computing platforms, we choose a modern GPU as the target architecture to evaluate the effectiveness of our proposed strategy. In this paper, we mainly concentrate on optimizing the reliability of the warp scheduler because of its importance. However, the technique described in this paper can be simply applied to CPU for NBTI mitigation as well. In general, the main contributions of this work are as follows:

- To the best of our knowledge, this paper is the first attempt to address the NBTI alleviation at the architectural level for future GPUs manufactured with FinFET.
- We propose a hybrid-device warp scheduler for reliable operation. By decoupling the warp scheduling into two steps of operations and conducting the prerequisites evaluation in a planar-device structure, we eliminate a large amount of read accesses to the FinFET scheduler hardware and considerably alleviate the NBTI effect.

II. BACKGROUND

A. NBTI Degradation Mechanism

Negative Bias Temperature Instability is becoming one of dominant reliability concerns for nanoscale P-MOSFETs. It is caused by the interaction of silicon-hydrogen (Si-H) and the inversion charge at the Si/oxide interface [10]. When a negative voltage is applied at the gate of PMOS transistors, the Si-H bonds



(a) (b)

Figure 1. FinFET transistor structure: (a) overview (b) side view

are progressively dissociated and H atoms diffuse into the gate oxide. This process eventually breaks the interface between the gate oxide and the conducting channel, leaving positive traps behind. As a consequence, the threshold voltage of the PMOS transistor is increased, which in turn elongates the switching delay of the device through the alpha power law [34]:

$$T_s \propto \frac{V_{dd} L_{eff}}{\mu (V_{dd}-V_t)^{\alpha}} \quad \text{(1)}$$

Where μ is the mobility of carriers, α is the velocity saturation index and approximates to 1.3. L_{eff} denotes the channel length. The process described above is termed the "stress" phase where the threshold voltage is persistently increasing with the service time, modeled by the following equation [40].

$$\Delta V_{tstress} = \left(\frac{qT_{ox}}{E_{ox}}\right)^{1.5} . K . \sqrt{C_{ox}(V_{gs}-V_t)} . e^{\frac{-E_a}{4kT}+\frac{2(V_{gs}-V_t)}{T_{ox}E_{01}}} . T_0^{-0.25} . T_{stress}^{0.25} \quad \text{(2)}$$

However, when the stress voltage is removed from the gate, H atoms in the traps can diffuse back to the interface and repair the broken bond. This results in a decrease in the threshold voltage, thus termed the "recovery" stage. This iterative stress-recovery processes lead to a saw-tooth variation of the threshold voltage throughout the device's lifespan. The final V_t increase taking both stress and recovery into account can be computed as:

$$\Delta V_t = \Delta V_{tstress} \cdot \left(1 - \frac{2\xi_1 T_{ox} + \sqrt{\xi_2 e^{\frac{-E_a}{kT}} T_0 T_{stress}}}{(1+\delta)T_{ox} + \sqrt{e^{\frac{-E_a}{kT}}(T_{stress}+T_{recovery})}}\right) \quad \text{(3)}$$

Note that in equations (2) and (3), T_{stress} and $T_{recovery}$ respectively denote the time under stress and recovery. Other parameters are either constants or material-dependent variables and are listed in Section 4.

That FinFET devices are more vulnerable to NBTI is generally attributed to its unique non-planar architecture, which is visualized by Figure 1. As can be seen, compared to a traditional planar transistor, the FinFET structure is designed with additional fin sidewall surface with higher availability of Si-H bonds [21][41], implying larger chances of forming interface trap and consequently expediting the device degradation.

The NBTI aging rate depends on multiple factors including both circuit parameters and workload execution patterns. In general, it is acknowledged that voltage, temperature, and the stress/recovery time have strong impact on the aging rate [10][38]. In this work, we mainly focus on the impact of temperature. Specifically, our proposed techniques significantly reduce the accesses to the target structures, thus lowering down the localized activity and temperature, which in turn enhances the structure durability.

B. *Target GPU Architecture*

The prevalence of unified programming language (e.g., CUDA, OpenCL) has made the general-purpose graphics processing unit a core component in a large variety of systems ranging from personal computers to high-performance computing clusters. Therefore, it is highly important to alleviate the NBTI degradation on this ever increasingly important platform.

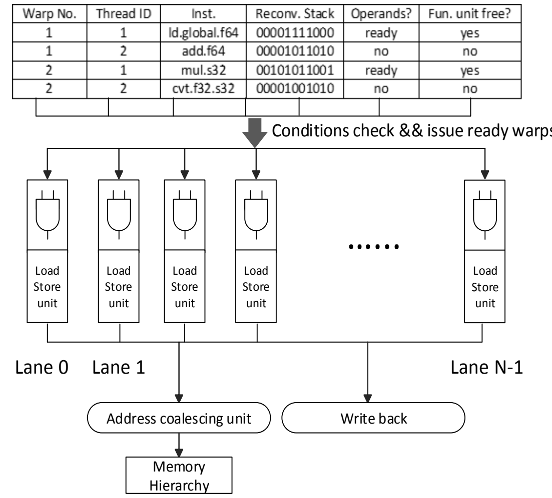


Figure 2. The architecture of the warp scheduler

In this section, we follow the Nvidia terminology to depict the architecture of a representative GPU. The major component of a modern GPU is an array of Streaming Multiprocessors (SMs), each of which contains an amount of CUDA cores (SPs), load/store units and special function units (SFUs). A CUDA core is responsible for performing integer ALU and floating point operations while the SFUs are devoted to conducting transcendental operations such as sine, cosine, and square root. Each stream multiprocessor also contains a register file, a shared memory and a level 1 cache (usually including the instruction, data, constant, and texture caches) that are shared among all threads assigned to the SM. All stream multiprocessors connect to an interconnection network, which transfers the memory requests/services between the SMs and the shared L2 cache.

An application developed in CUDA (or OpenCL) contains at least one kernel running on the GPU. A typical kernel includes several blocks composed of substantial threads. During a kernel execution, multiple blocks are assigned to an SM according to the resource requirement. A group of threads from the same block form a warp treated as the smallest scheduling unit to be run on the hardware function units in an SIMT fashion.

III. HYBRID-DEVICE WARP SCHEDULER

As an emerging platform targeting for massively parallel computing domains, a modern GPU is designed with several unique characteristics different from a regular CPU. In this section, we concentrate on the warp scheduler because it is an important structure that is frequently accessed during program execution. By observing representative execution behaviors of a large collection of GPU applications, we propose a technique exploiting the device heterogeneity to alleviate the NBTI degradation. As we will demonstrate shortly, the proposed technique does not introduce any additional component to the existing GPU architecture, thus minimizing the hardware cost for the implementation.

A. *Opportunity for Improvement*

To improve the thread-level parallelism (TLP) and maximize the execution throughput, a modern GPU usually allows multiple warps to reside on the same streaming multiprocessor and hide the execution latencies by switching among those resident warps. At any instant, a warp is considered as ready for execution only when several constraints are simultaneously satisfied.

A first-order prerequisite is the functional correctness, which is secured by ensuring data dependencies between warp instructions. When a warp cannot be dispatched because of unsatisfied data dependency, it should wait until all of its operands are ready. A scoreboard hardware structure is responsible for keeping track of

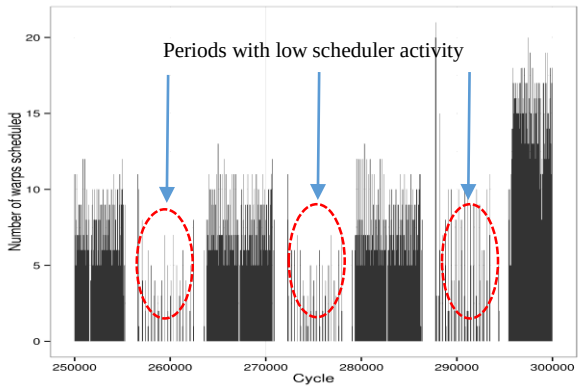


Figure 3. A snapshot of the scheduler activity while running *WP*

data dependencies in a modern GPU. In addition, warps on a streaming multiprocessor contend for limited functional units. When the dispatch port of the functional unit that a warp needs to use is not vacant, the warp cannot be issued even when its data dependencies have been satisfied.

The warp scheduler is an SRAM hardware structure in charge of selecting candidates from all resident warps to dispatch. For the purpose of high performance, a warp scheduler is capable of dispatching one warp per clock cycle, requiring that scanning through all the scoreboard entries and querying the dispatch ports of all functional units should be performed at each cycle [23][25]. Figure 2 illustrates the high-level organization of a warp scheduler equipped in an SM to elaborate the scheduling process. As shown in the figure, all entries, each of which stores complete information of a warp instruction, are going through the conditions checking in parallel in order to identify the candidates ready for execution. Note that to minimize the delay, the scheduler must read the detailed information of a warp (warp ID, opcode, etc) while evaluating the constraints so that it can dispatch warps as soon as they are ready. Selected warps are sent to the appropriate function units according to the instruction opcode afterwards.

This particular design naturally inspires a technique to mitigate the NBTI degradation on the scheduler. If the readiness of all warp instructions are known ahead via a certain "predicate", then only the entries with all constraints met are accessed, which in turn decreases the localized activity and temperature, and improves the structure durability.

To justify the potential effectiveness of this strategy, we run a wide spectrum of GPU applications, aiming to observe typical behaviors on the warp scheduler. Figure 3 plots a snapshot of the warp scheduler's behavior when *WP* is running on a GPU in order to exemplify the activity on the scheduler. The horizontal axis corresponds to the elapsed time and the vertical axis represents the accumulative number of ready warps at each time interval. The number is collected every 50 cycles. With this setting, the maximum number of ready warps cannot exceed 100 on each sampling point considering that two warp instructions can be issued at each cycle. As can be seen from the figure, there are a large amount of execution periods with number of ready warps far less than the theoretical peak, implying a significant reduction in accesses to the scheduler entries in potential. We generally observe that, at any given instant, less than 35% of all the warps have the two prerequisites satisfied for all the tested benchmarks. This observation confirms that there is large headroom for us to optimize the reliability on the warp scheduler.

B. Two-stage Scheduling

Our proposed technique to enhance the durability of the warp scheduler stems from the aforementioned fact at the first place. In order to identify the ready warps, the baseline scheduler is decoupled into two components as visualized in Figure 4. By doing

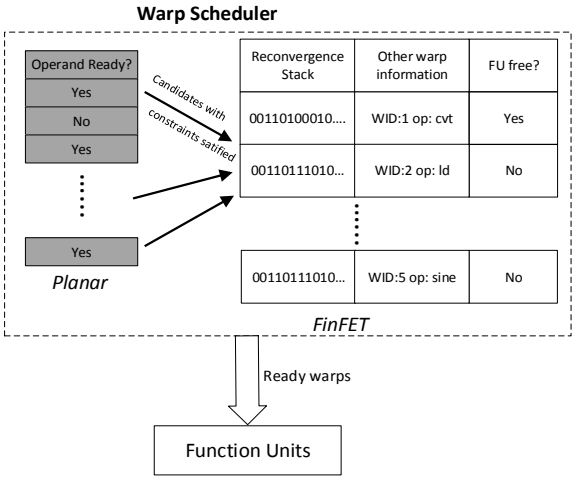


Figure 4. The architecture of hybrid-device 2-stage scheduler

so, the prerequisites checking is extracted from the original parallel accesses and is performed prior to obtaining the detailed information of warp instructions. This checking operation outputs the ID of all available candidates resided on the SM, triggering the consequent accesses to the hardware structure which stores all necessary information to dispatch ready warps based on the specific scheduling policy. If a large amount of resident warps are eliminated from the candidate list due to the violation of scheduling constraints, substantial accesses to the scheduler hardware (i.e., the structure at the right side in Figure 7) can be avoided.

A non-trivial issue requiring careful consideration in this particular scheduler design is what information should be checked in the first stage. Theoretically, evaluating more scheduling prerequisites would filter larger number of accesses since only the common set of candidates that satisfy each individual constraints are allowed to continue the second stage. However, for certain conditions, checking them in the first stage would lead to undesirable execution behavior because their evaluation results might be changed in the following cycle. The checking on function units' (FU) availability falls into this category. This is because that the FU status is updated every cycle and a function unit that appears to be free in the current cycle is not necessarily available in the following cycle, if it is assigned to another warp instruction. Therefore in this work, we only check the data dependency in the first stage. As we will demonstrate in section 6, this still results in sufficiently high filter rate for most benchmarks and largely alleviate the NBTI degradation.

On the other hand, considering that the failure of any structure located on the critical path will prevent the entire chip from working correctly, the component where the condition evaluations are conducted tends to become the bottleneck from the perspective of reliability, since all of its entries still needs to be scanned every cycle. To overcome this problem, we propose to manufacture this component with the more NBTI-tolerable planar devices. This hybrid-device design effectively leverages the benefits of both devices, aiming to enhance the processor durability. Note that the planar-transistor-made component recording the data dependency and function unit availability is unlikely to suffer from early failure because it only requires one bit for each entry and thus consume negligible power. Also recall that this design is technically feasible due to the good compatibility between FinFET and planar processes as demonstrated in patents [12][20].

Another naturally arising concern with this design is the performance degradation resulted from the sequential scheduler access. Nevertheless, as we will demonstrate in section 5, the performance overhead for most applications are fairly small because only actual accesses to the FinFET part of the scheduler

Table 1. Architectural parameters for the GPU in study.

Parameter	Values
#SM	15
#SP	32/SM
LDST units	16/SM
Shared memory	32KB/SM
L1 data cache	16KB/SM
Scheduler	Greedy than oldest (GTO)
Core frequency	1400MHz
Interconnection	1 crossbar/direction
L2 cache	768KB: 128 cache line size, 16-way associativity. Access latency 5 cycles
L2 frequency	700MHz
Memory	FR-FCFS scheduling, 64 max. requests/MC
SIMD lane width	16
Threads/warp	32
Technology	22nm

Table 2. Benchmarks used in this work.

#	Application	Domains
1	B+tree	Search
2	Backprop	Pattern Recognition
3	Blackscholes	Financial Engineering
4	Gaussian	Linear Algebra
5	Heartwall	Medical Imaging
6	LPS	3D Laplace Solver
7	Myocyte	Biological Simulation
8	NN	Neural Network
9	NW	Bioinformatics
10	WP	Weather Prediction

introduces an extra cycle delay. In scenarios where none of the resident warps pass the constraints checking, the execution latency is not impacted.

IV. Experimental Setup

We validate the proposed techniques using a modified GPGPU-Sim 3.1 [14], a cycle-accurate GPGPU simulator. GPUWattch [29] and HotSpot 5.0 [8] are integrated in the simulator for power and temperature calculation, respectively. The chip floorplan required by HotSpot is calibrated against the one used in a recent paper focusing on GPU thermal management [31]. The target architecture is configured based on a Fermi GTX 480 [6] that is widely used in many high-performance computers. Table 1 lists the architectural parameters for our simulation.

To evaluate the effectiveness of our techniques in practice, we choose a set of programs from several benchmark suites [5][14][16], representing typical HPC applications derived from different domains. A full list of applications used in this work is given in Table 2. For each program, we run them till completion and use the execution statistics to mimic distinct workload patterns. In specific, to model the NBTI degradation after a 7-year lifespan, we extrapolate the collected activity to represent the load in 7 years under the steady temperature. Note that our technique does not explicitly introduce recovery stages to the structure in study, so the recovery time is set to zero. We report the final increase in the critical path delay as a measurement of the NBTI aging on the hardware. Equations (2) and (3) described in section 2.1 are used to compute the variation in the threshold voltage, which in turn translates to the delay increase via equation (1). We set the parameters referred by the equations according to recent studies on device features [9][15][36]. Table 3 lists the specific parameter values used in this paper.

V. Result Analysis

A. Improvement on Reliability

Figure 5 demonstrates the NBTI degradation in terms of the increase in scheduler delay on both the baseline GPU and the one

Table 3. Parameter values for computing NBTI.

Parameters	FinFET	Planar	Description
T_{ox}	1.2nm	1nm	Effective oxide thickness
V_t	0.179v	0.3v	Threshold voltage
E_o	0.335v/nm	0.12v/nm	Electrical field
Fixed parameters			
q	1.602×10^{-19}		Electron charge
V_{dd}	0.9v		Operating voltage
ε_{ox}	1.26×10^{-19}F/m		Permittivity of gate oxide
ξ_1	0.9		Other constants
ξ_2	0.5		
k	8.6174×10^{-5} ev/K		
δ	0.5		
T0	10^{-8} s/nm^2		

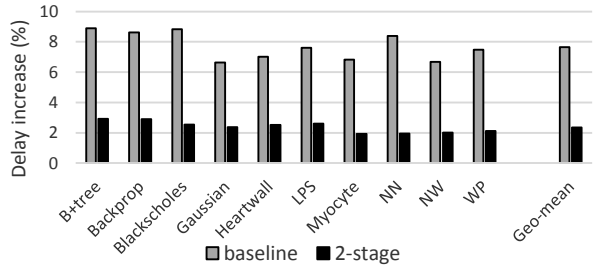


Figure 5. The NBTI degradation on the warp scheduler

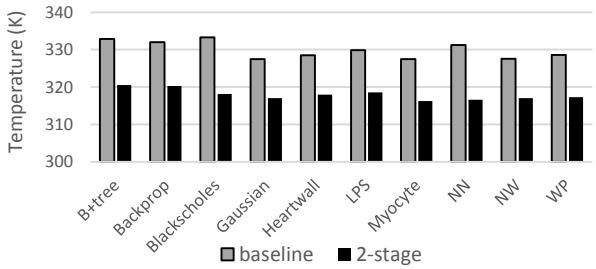


Figure 6. The steady temperature on the warp scheduler

with hybrid-device 2-stage warp scheduler. Note that in the figure, the bars marked by "2-stage" refer to the proposed design. A higher delay increase indicates more severe NBTI degradation. As can be observed, the aging due to NBTI on the scheduler hardware is largely suppressed for all benchmarks under investigation when the proposed technique is applied. On average, the hybrid-device 2-stage scheduler presents merely 2.36% longer delay after the designed service life, reduced from 7.7% on the baseline GPU.

While the general improvement on the durability is significant, however, it is notable that the benefits corresponding to different workloads are obviously distinct. For example, the load represented by *NN* causes the scheduler delay to be prolonged by around 8.4% after 7 years services on the baseline GPU. With the adoption of the proposed technique, this degradation can be reduced to 1.96%. On the other hand, an execution pattern similar to *Backprop* prevents the scheduler obtaining the same amount of benefit from the technique. Specifically, the scheduler still suffers from 2.9% longer delay after employing the hybrid-device design, while the baseline platform shows 8.6% longer delay that is similar to the degradation corresponding to *NN*.

Considering the exponential relationship between temperature and NBTI degradation, we collect the localized temperature on the scheduler hardware and demonstrate it in Figure 6 for further analysis. Not surprisingly, although the proposed technique can significantly cool down the scheduler in most cases, we note that the temperature reductions are apparently different among the evaluated programs, which is similar to the observation made from Figure 5. When executing *NN*, the temperature on the scheduler is reduced by up to 15°C, whereas the temperature reduction for *Backprop* is about 11°C. To gain more insights into the reason

Table 4. Filter rate on the first stage of warp scheduler.

Application	Filter Rate
B+tree	75.82%
Backprop	76.93%
Blackscholes	88.74%
Gaussian	98.82%
Heartwall	88.46%
LPS	90.59%
Myocyte	99.85%
NN	97.41%
NW	97.70%
WP	99.49%
Geo-mean	90.96%

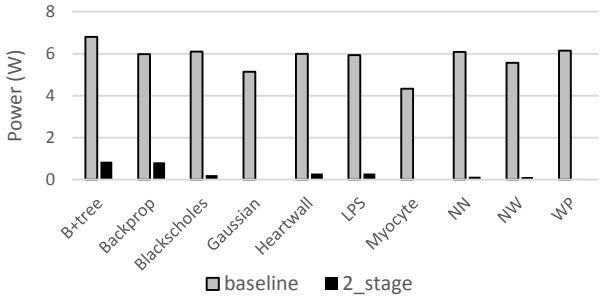


Figure 7. The power consumed by the warp scheduler

behind this phenomenon, let us recall the rationale of the 2-stage scheduler that is described in section 3.2. The essential reason for the reduced scheduler accesses is that a large amount of prerequisite evaluations turn out to be false, thus the unnecessary operations on the "unready warps" are avoided. In other words, how much benefit can be obtained from the proposed technique largely depends on the amount of accesses that can be filtered. Table 4 lists the percentage of accesses saved by the constraint checking stage. As can be seen, the data dependency checking stage can generally filter out more than 92% of accesses to the scheduler, thus considerably enhancing the durability of the hardware. In particular, we note that 76.9% of scheduler accesses when executing *Backprop* are dispensable, while for *NN* this ratio rises up to 97.4%, implying higher possibilities to lower the power and temperature on the scheduler.

We also plot the power consumption of the scheduler in Figure 7 to visualize the changes on the scheduler activity. Clearly, the hybrid-device 2-stage scheduler significantly reduces the scheduler power for all evaluated benchmarks, which in turn lowers the local temperature and improves the hardware durability.

B. Performance Overhead

The extra cycle introduced by the 2-stage scheduler is likely to result in undesirable performance overhead for the program execution. Figure 8 shows the performance in terms of normalized IPC (normalized to the baseline GPU) of all benchmarks running on a GPU with the 2-stage scheduler. It is straightforward to note that the performance degradation is distinct among the program collection. In this subsection, we briefly analyze the possible impact on the performance due to the extra cycle and explain the different performance degradation.

The GPU's massive parallelism may be able to hide part of the extra latency during the execution depending on the features of applications. We use the terms "longest warp" and "longest-warp chain" to help explain the latency manifested in the results. We define "longest warp" as the warp with the longest running time during a kernel launch and "longest-warp-chain" as the set of longest warps in each of the sequence of kernel launches in the lifetime of an application. In a typical GPU application, the running time of a longest-warp chain is the sum of execution latencies of all warps in the chain because a) when a kernel is launched, all its warps are started simultaneously and b) a kernel

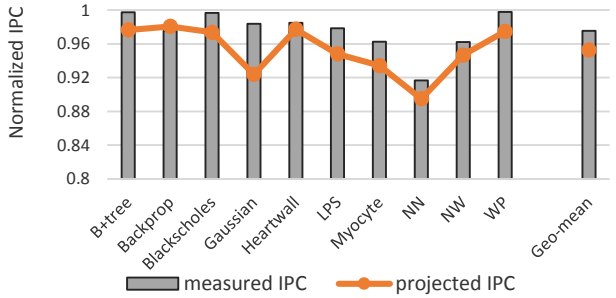


Figure 8. Normalized IPC on the GPU with 2-stage scheduler

is not launched until all warps of the previous kernel launch complete. In other words, latency on the longest warp could not be hidden as easily as that on other warps. Longest warps also do not overlap temporally. For each longest warp we can compute its average latency as:

$$AvgLatency = \frac{\sum_{n=1}^{N} C_n}{\sum_{n=1}^{N} I_n} \quad (4)$$

Where N is the number of kernel launches and C_n and I_n are respectively the number of cycles and warp instructions of the longest warp in each kernel launch.

The I_n instructions in a kernel launch are the instructions issued to and executed by a warp. The extra cycle introduced to the scheduler will be added before each of the instructions is executed. Since the instructions are executed in-order, this is equivalent to adding extra cycles to the entire longest-warp chain. The average latency of the warp after adding the extra cycles should become:

$$AvgLatency_{delayed} = \frac{\sum_{n=1}^{N} C_n + \sum_{n=1}^{N} I_n}{\sum_{n=1}^{N} I_n} \quad (5)$$

The overhead indicators can be deducted from the two latencies shown above:

$$OverheadInd = \frac{\Delta latency}{AvgLatency} = \frac{AvgLatency_{delayed} - AvgLatency}{AvgLatency} = \frac{\frac{\sum_{n=1}^{N} C_n + \sum_{n=1}^{N} I_n}{\sum_{n=1}^{N} I_n} - \frac{\sum_{n=1}^{N} C_n}{\sum_{n=1}^{N} I_n}}{\frac{\sum_{n=1}^{N} C_n}{\sum_{n=1}^{N} I_n}} = \frac{\sum_{i=1}^{N} C_n + \sum_{i=1}^{N} I_n - \sum_{i=1}^{N} C_n}{\sum_{i=1}^{N} C_n} = \frac{\sum_{i=1}^{N} I_n}{\sum_{i=1}^{N} C_n} = \frac{1}{AvgLatency} \quad (6)$$

The normalized IPC (measured) and the one derived from the overhead indicator (projected) are both plotted in Figure 8. As the figure shows, they are closely correlated. The average latencies and the overheads are determined by the behaviors of the longest warps which are in turn closely related to the characteristics of individual applications. For example, *B+tree* involves a kernel launch with 48 warps on each SM and initiates many global memory transactions (159.26 per cycle). Its longest warp has an average delay of more than 100 cycles. *NN*, on the other hand, has a much smaller average delay (smaller than 10), because it generates much fewer global memory transactions (only 0.06 per cycle) and each SM executes only 8 warps. With such few memory transactions and fewer warps, each of the warps, including the longest warp, does not have to wait for long-delay memory operations while sharing more computational resources. This different memory request intensities result in average latencies of the longest warp chains as 41.7 and 8.53 cycles for *B+tree* and *NN*, respectively. Consequently, we observe apparently different performance losses for these two benchmarks.

VI. Related Work

NBTI Mitigation: NBTI has been recognized as a major reliability concern as the semiconductor industry shifts into the deep submicron era. Abella et al. [10] develop a set of techniques to relieve the NBTI aging for typical structures in a modern CPU.

For combinational logics, they insert desired vectors as inputs to the structures for recovery. To alleviate the aging for memory-like components, they propose a strategy to avoid the bias on different bits. Ramakrishnan et al. [33] introduce a similar approach to reduce the NBTI wearout in FPGAs by loading the reversing bit patterns in idle periods. Gunadi et al. [22] introduce a scheme called Colt to balance the utilization of devices in a processor for reliability improvement. Specifically focusing on the storage components, Shin et al. [35] propose to proactively set the PMOS transistors to recovery mode, and moving data around free cache arrays during operation.

Converse to these works which attempt to manipulate the time under stress and recovery, Tiwari et al. [38] propose a framework named facelift to combat NBTI degradation by adjusting higher level parameters including operating voltage, threshold voltage and the application scheduling policy. To enhance the reliability of storage cells, Abella [11] proposes to use NAND gates instead of inverters to reduce the average degradation on each PMOS. On the other hand, Rahimi et al. [32] focus on the GPUs designed in VLIW fashion and present a compiler-based technique to slow down the NBTI aging for this particular architecture.

Characterization of FinFET Reliability: As FinFET is widely considered as an attractive replacement of planar transistors for the next few technology nodes, studies focusing on the reliability of this new structure is becoming fairly important. Lee et al. [28] investigate the NBTI characteristics on SOI and body-tied FinFETs and observe that a narrow fin width leads to more severe degradation than a wider fin width. Crupi et al. [19] compare the reliability of triple-gate and planar FETs. The author show that the behavior of time-dependent dielectric breakdown (TDDB) is not changed on the triple-gate architecture under different gate voltages and temperatures. This is also corroborated in the work conducted by Groeseneken et al. [21], which further demonstrate that FinFET devices tend to suffer from more severe NBTI degradation. In [39], Wang et al. analyze the soft-error resilience of FinFET devices and conclude that FinFET circuit is more reliable than bulk CMOS circuit in terms of soft-error immunity.

VII. Conclusion

FinFET technology is recognized as a promising substitute of conventional planar devices for building processors in the next decade due to its better scalability. However, recent experimental studies demonstrate that FinFET tends to suffer from more severe NBTI degradation compared to the planar counterpart. In this work, we focus on the NBTI reliability issue of a modern GPU made of FinFET and propose to address this problem by exploiting the device heterogeneity. We introduce a technique that merely involves minor modifications to the existing GPU architectures. The proposed technique leverages planar devices' higher immunity to NBTI and is effective in slowing down the aging rate of the device. Our evaluation results demonstrate that the minor changes to the warp scheduler can considerably alleviate the degradation due to NBTI with slight performance overhead.

References

[1] Intel Corporation. Intel's revolutionary 22nm transistor technology. May 2011.

[2] Intel Corporation. 3rd Generation of Intel Core i7 Processor. http://ark.intel.com/products/family/65505.

[3] Intel Corporation. 4th Generation of Intel Core i7 Processor. http://ark.intel.com/products/family/75023.

[4] Nvidia Corporation. CUDA C Programming Guide.

[5] Nvidia Corporation. CUDA Computing SDK 4.2.

[6] GTX 480 Specifications. http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-480/specifications

[7] http://www.eetimes.com/document.asp?doc_id=1264668.

[8] Hotspot 5.0 Temperature Modeling Tool. http://lava.cs.virginia.edu/HotSpot

[9] Predictive Technology Model. http://ptm.asu.edu

[10] J. Abella, X. Vera, and A. Gonzalez. Penelope: The NBTI-aware processor. In MICRO'07.

[11] J. Abellea, X. Vera, O. Unsal, and A. Gonzalez. NBTI-resilient memory cells with NAND gates for highly-ported structures. In Workshop on Dependable and Secure Nanocomputing, Jun. 2007.

[12] B. A. Anderson, A. J. Joseph, and E. J. Nowak. Integrated circuit including FinFET RF switch angled relative to planar MOSFET and related design structure. U.S. Patent 8125007 B2, Feb. 2012.

[13] A. Asenov, C. Alexander, C. Riddet, and E. Towie. Predicting future technology performance. In DAC'13.

[14] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt. Analyzing CUDA Workloads Using a Detailed GPU Simulator, in ISPASS'09.

[15] S. Chaudhuri, and N. K. Jha. 3D vs. 2D analysis of FinFET logic gates under process variations. In ICCD'11.

[16] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron. Rodinia: A Benchmark Suite for Heterogeneous Computing. In IISWC'09.

[17] Z. Chishti, M. D. Powell, and T. N. Vijaykumar. Distance associativity for high performance energy efficient non-uniform cache architectures. In MICRO'03.

[18] J. P. Colinge, "Multiple-gate SOI MOSFETs", Solid-State Electronics, vol. 48, no. 6, June 2004.

[19] F. Crupi, B. Kaczer, R. Degraeve, V. Subramanian, P. Srinivasan, E. Simoen, A. Dixit, M. Jurczak, and G. Groeseneken. Reliability comparison of triple-gate versus planar SOI FETs. In IEEE Transactions on electron devices, vol. 53, no. 9, Sept. 2006.

[20] B. B. Doris, D. C. Boyd, M. Leong, T. S. Kanarsky, J. T. Kedzierski, M. Yang. Hybrid planar and FinFET CMOS devices. U.S. Patent 7250658 B2, Jun. 2007.

[21] G. Groeseneken, F. Crupi, A. Shickova, S. Thijs, D. Linten, B. Kaczer, N. Collaert, and M. Jurczak. Reliability issues in MUGFET nanodevices. In IEEE 46th Annual International Reliability Physics Symposium (IRPS), April 2008.

[22] E. Gunadi, A. A. Sinkar, N. S. Kim, and M. H. Lipasti. Combating aging with the colt duty cycle equalizer. In MICRO'10.

[23] J. Hennessy, D. A. Patterson. Computer architecture: a quantitative approach. 5th edition.

[24] A. B. Kahng. The ITRS design technology and system drivers roadmap: process and status. In DAC'13.

[25] H. Kim, R. Vuduc, S. Baghsorkhi, J. Choi, and W. Hu. Performance analysis and tuning for general purpose graphics processing units (GPGPU). DOI: 10.2200/S00451ED1V01Y201209CAC020

[26] V. B. Kleeberger, H. Graeb, and U. Schlichtmann. Predicting future product performance: modeling and evaluation of standard cells in FinFET technologies. In DAC'13.

[27] E. Kultursay, J, Swaminathan, V. Saripalli, V. Narayanan, M. Kandemir, and S. Datta. Performance enhancement under power con-straints using heterogeneous CMOS-TFET multicores. In CODES+ISSS'12.

[28] H. Lee, C-H. Lee, D. Park, and Y-K. Choi. A study of negative-bias temperature instability of SOI and body-tied FinFETs. In IEEE Electron Device Letters, vol. 26, no.5, May 2005.

[29] J. Leng, T. Hetherington, A. Eltantawy, S. Gilani, N. S. Kim, T. M. Aamodt, V. J. Reddi. GPUWattch: enabling energy optimizations in GPGPUs. In ISCA'13.

[30] S. Mahapatra, P. B. Kumar, and M. A. Alam. Investigation and modeling of interface and bulk trap generation during negative bias temperature instability of p-MOSFETs. In IEEE Transactions on Electron Devices, vol. 51, no.9, Sept. 2004.

[31] R. Nath, R. Ayoub, and T. S. Rosing. Temperature aware thread block scheduling in GPGPUs. In DAC'13.

[32] A. Rahimi, L. Benini, R. K. Gupta. Aging-aware compiler-directed VLIW assignment for GPGPU architectures. In DAC'13.

[33] K. Ramakrishnan, S. Suresh, N. Vijaykrishnan, M. J. Irwin, and V. Degalahal. Impact of NBTI on FPGAs. In VLSI'07.

[34] T. Sakurai and R. Newton. Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas. IEEE Journal of Solid-State Circuits, 1990.

[35] J. Shin, V. Zyuban, P. Bose, and T. M. Pinkston. A proactive wearout recovery approach for exploiting microarchitectural redundancy to extend cache SRAM lifetime. In ISCA'08.

[36] S. Sinha, G. Yeric, V. Chandra, B. Cline, and Y. Cao. Exploring sub-20nm FinFET design with predictive technology models. In DAC'12.

[37] B. Swahn and S. Hassoun. Gate sizing: FinFETs vs 32nm Bulk MOSFETS. In DAC'06.

[38] A. Tiwari and J. Torrellas. Facelift: Hiding and slowing down aging in multicores. In MICRO'08.

[39] F. Wang, Y. Xie, K. Bernstein, and Y. Luo. Dependability analysis of nano-scale FinFET circuits. In ISVLSI'06.

[40] W. Wang, V. Reddy and A. Krishnan, "Compact Modeling and Simulation of Circuit Reliability for 65-nm CMOS Technology", IEEE Transactions on Device and Materials and Reliability, 7(4):509-517, December 2007.

[41] Y. Wang, S.D. Cotofana, and L. Fang. Statistical reliability analysis of NBTI impact on FinFET SRAMs and mitigation technique using independent-gate devices. In NANOARCH'12.