Hybrid CMOS-TFET based Register Files for Energy-Efficient GPGPUs

Zhi Li, Jingweijia Tan, Xin Fu
EECS Department, University of Kansas, Lawrence, KS 66045, USA
zli@ku.edu, jtan@ittc.ku.edu, xinfu@ittc.ku.edu

Abstract

State-of-the-art General-Purpose computing on Graphics Processing Unit (GPGPU) is facing severe power challenge due to the increasing number of cores placed on a chip with decreasing feature size. In order to hide the long latency operations, GPGPU employs the fine-grained multi-threading among numerous active threads, leading to the sizeable register files with massive power consumption. Exploring the optimal power savings in register files becomes the critical and first step towards the energy-efficient GPGPUs. The conventional method to reduce dynamic power consumption is the supply voltage scaling, and the inter-bank tunneling FETs (TFETs) are the promising candidates compared to CMOS for low voltage operations regarding to both leakage and performance. However, always executing at the low voltage (so that low frequency) will result in significant performance degradation. In this study, we propose the hybrid CMOS-TFET based register files. To optimize the register power consumption, we allocate TFET-based registers to threads whose execution progress can be delayed to some degree to avoid the memory contentions with other threads, and the CMOS-based registers are still used for threads requiring normal execution speed. Our experimental results show that the proposed technique achieves 30% energy (including both dynamic and leakage) reduction in register files with little performance degradation compared to the baseline case equipped with naive power optimization technique.

Keywords

Tunneling field effect transistors, general-purpose computing on graphics processing units, energy efficiency

1. Introduction

Modern graphics processing unit (GPU) supports tens of thousands of parallel threads and delivers remarkably high computing throughput. General-Purpose Computing on GPUs (GPGPUs) are becoming the attractive platform for general-purpose applications that request high computational performance such as scientific computing, financial applications, medical data processing, and so on. However, GPGPU is facing severe power challenge due to the increasing number of cores placed on a single chip with decreasing feature size [1].

In order to hide the latency induced by the function unit computation and off-chip memory accesses, GPU employs the fine-grained multi-threading that quickly switches among a large number of simultaneously active threads. As a result, substantial register files are required to keep the register context of each thread. For example, Nvidia Fermi GPU supports more than 20,000 parallel threads and contains 2MB register files [7]. Accessing such sizeable register files leads to massive power consumption [2-6]. It has been reported that the register files consume 15%-20% of the GPU stream multiprocessor's power [8]. Effectively optimizing the register files power consumption is critical and the first step towards the energy-efficient GPUs.

Supply voltage scaling is the fundamental technique to reduce the dynamic power consumption, but it is limited by the leakage constraints in CMOS digital circuits. Recently, Inter-bank Tunneling Field Effect Transistors (TFETs) have shown to be the attractive candidates to operate at low supply voltages (e.g. 0.3V) with ultra low leakage and higher frequency than CMOS [9, 10]. However, at higher supply voltage, CMOS devices are able to achieve much better performance than TFETs. The unique characteristics of CMOS and TFETs at different voltage levels provide great opportunity in GPU power savings without hurting the performance.

In the GPGPU applications, all threads in a kernel execute the same code [11], and exhibit similar execution progress in the fine-grained multi-threading environment. When one thread encounters an off-chip memory access, other threads are likely to issue the requests at approximately the same time, leading to severe memory contentions which extend the memory access time. And the pipeline in the GPU stream multiprocessor stalls when all threads stall due to the long-latency memory accesses. It has been found that the performance of numerous GPGPU workloads are still bounded by the memory resources even modern GPUs provide very high memory bandwidth [30]. In order to alleviate the memory contentions and efficiently utilize the memory bandwidth, threads can run at different paces which effectively avoid the interferences among memory requests. It enables the implementation of the TFET-based registers in GPGPUs for a number of threads so that they can run at a lower frequency without any performance degradation, and meanwhile, both the dynamic and leakage power of the registers reduces substantially. On the other hand, applying TFETs for all registers in the GPGPU will cause significant performance penalty since many threads still need to execute at high frequency to achieve the high computational throughput.

In this paper, we propose the hybrid CMOS-TFET based registers in GPUs to obtain optimal energy reduction with negligible performance penalty. The contributions of this study are as follows:

(1) We observe that threads in GPGPU workloads can be seriously delayed while executing in the GPU streaming multiprocessors due to the memory access interference with others. Instead of stalling in the pipeline on the occurrence of serious memory contentions, threads can execute at a low speed by using TFET-based registers to postpone their memory requests. It helps to achieve the win-win scenario: preventing the interferences and achieving the attractive power savings.

(2) We propose to build the hybrid TFET-based and CMOS-based registers, and perform the memory contention-

aware register allocation. Based on the access latency of previous memory transaction, we predict the thread stall time during its following memory access, and allocate TFET-based registers to that thread to postpone its execution progress to the maximum degree without performance loss. By doing this, we maximize the utilization of the TFET-based registers, thus, optimize the energy consumption while maintaining the performance.

(3) Our evaluation results show that the proposed register allocation technique in the hybrid register design exhibits the strong capability of reducing the register energy consumption (including both dynamic and static energy) by 30% compared to the case with naive power optimization technique (i.e. power gating the unused registers [2]). Especially, it achieves 42% energy reduction (16% dynamic saving and 26% leakage saving) in memory-intensive benchmarks with only 2.5% performance degradation.

The rest of the paper is organized as follows: Section 2 provides the background of GPGPU and TFETs. Section 3 proposes the hybrid CMOS-TFET based registers and the memory contention-aware TFET-based register allocation. Section 4 describes our experimental methodologies and evaluates the proposed mechanism. We discuss the related work in Section 5, and conclude with Section 6.

2. Background

2.1. General-Purpose Computing on Graphics Processing Units (GPGPUs) Architecture

A typical GPU consists of a scalable number of in-order streaming multiprocessors (SM) that can access to multiple on-chip memory controllers via an on-chip interconnection network [11]. In GPU programming models, highly-parallel kernel functions are launched to the GPU for execution. The kernel is composed of a grid of light-weighted threads; a grid is divided into a set of blocks; each block is composed of hundreds of threads. Threads in the kernel are assigned to the SMs at the granularity of blocks.

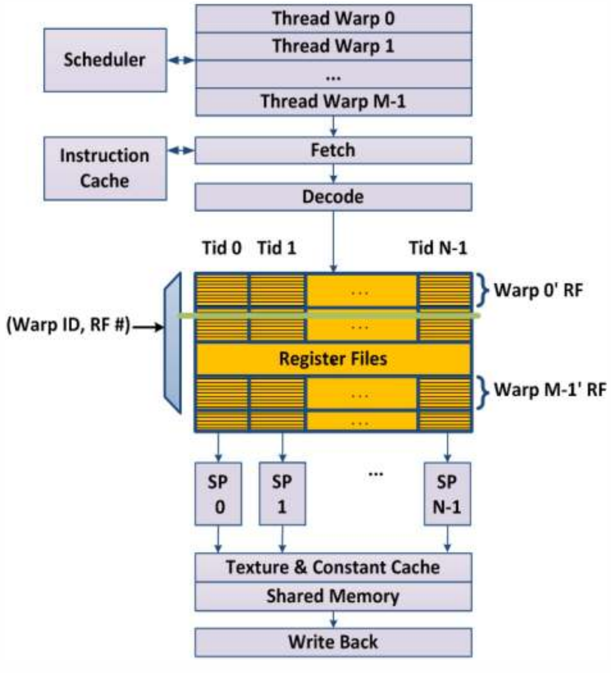


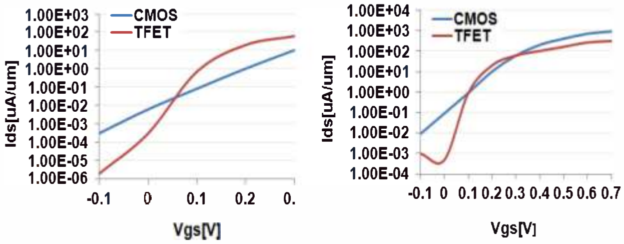
Figure 1. Streaming Multiprocessor microarchitecture

Figure 1 illustrates the SM microarchitecture. Threads in the SM execute on the single-program multiple-data (SPMD) model. A number of individual threads (e.g. 32 threads) from the same block are grouped together, called warp. In the pipeline, threads within a warp execute the same instruction but with different data values. As Figure 1 shows, each warp has a dedicated slot in the warp scheduler. At every cycle, a ready warp is selected by the scheduler to feed the pipeline. The instruction is then fetched from the instruction cache based on the PC of the issued warp, and further decoded. In the SM, a number of registers are statically allocated to each warp when the block is distributed. All threads in the warp access a number of registers (i.e. the register vector) simultaneously based on the warp ID and the register number, the register values are processed in parallel across the streaming processors (SP).

GPU is usually equipped with its own off-chip external memory (e.g. global memory) connected to the on-chip memory controllers. The off-chip memory access can last hundreds of cycles, and a long latency memory transaction from one thread would stall all threads within a warp. In other words, the warp cannot proceed until all the memory accesses from its threads complete.

2.2.Tunneling Field Effect Transistors (TFETs)

The sub-threshold slope of the transistor is the key factor in leakage power consumption, and a steep sub-threshold device achieves low leakage current. Traditional CMOS devices are limited to 60mV/decade sub-threshold slope which induces high leakage current during the voltage scaling [12]. While TFETs [9] exhibit sub-60mV/decade sub-threshold slope and achieve very low leakage power consumption at low supply voltage. Figure 2(a) compares the OFF-state leakage current (I_{OFF}) and ON current (I_{ON}) of the two kinds of devices when VCC is 0.3V. As it shows, TFETs are able to obtain much lower leakage current and stronger driven current, therefore, ultra low leakage with high frequency. They are promising for energy-efficient computing. On the other hand, as Figure 2(b) exhibits, although TFETs are still able to achieve low I_{OFF} at high supply voltage (e.g. 0.7V), CMOS devices have larger driven current and better performance than TFETs.



(a) (b)

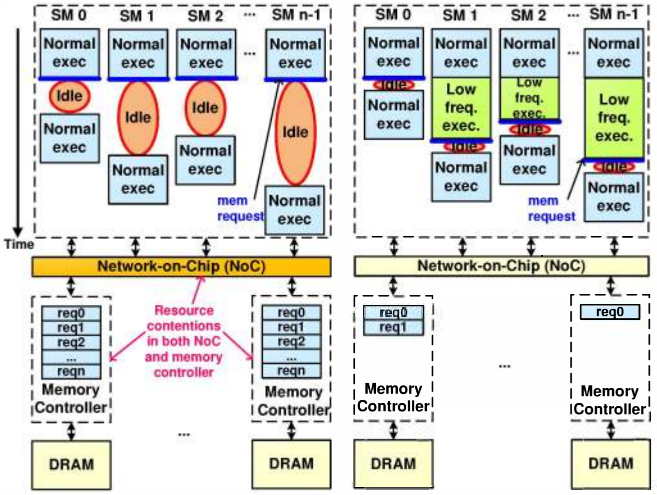
Figure 2. OFF-state leakage current and ON current of TFET and CMOS when VCC is (a) 0.3V and (b) 0.7V (adapted from [13])

TFETs have the characteristic of uni-direction conduction which causes a challenge on designing the SRAM storage cell. Recently, many different TFET SRAMs have been explored to overcome this limitation [14-17]. By comparing those designs on several aspects (e.g. frequency, noise margins, power, and area), in this study, we apply the 6T TFET SRAM proposed by Singh et al. [14] to implement the TFET-based register files.

3. Hybrid CMOS-TFET based Register Files

3.1. The Observation on Memory Contentions

In GPUs, the off-chip memory requests from SMs need to go through the on-chip network routing to certain memory controller and wait there to be served. When numerous requests are issued at similar time by multiple SMs, both on-chip network and memory controllers will be severely congested which significantly increases the memory access time. Unfortunately, such congestion issue occurs frequently in GPUs due to the unique characteristic of the GPGPU applications: all threads in the kernel across SMs execute the same instructions and proceed at similar rate in the fine-grained multithreading environment. Although there are up to thousands of active threads running in each SM, they are unlikely to fully hide the extremely long-latency memory transaction caused by the memory contentions. As a result, the SM suffers long-time pipeline stall. The GPU memory bandwidth is already considered as one of the resource constraints for many GPGPU workloads even modern GPUs provide pretty high memory bandwidth [30].



(a) (b)

Figure 3. (a) SMs suffer long pipeline stall due to the severe memory contentions. (b) Leveraging TFETs to absorb the long pipeline stall and alleviate memory contentions.

Figure 3(a) shows an example of the memory resource contentions among SMs. Several SMs encounter the global memory access instructions and send out memory requests simultaneously. The buffers in network-on-chip (NoC) and memory controllers are quickly filled up by those requests and they have to be served sequentially to access the DRAM buffers (Figure 3(a) takes a snapshot on the NoC and memory controllers). Therefore, the memory transactions spend longer time to finish, and the pipeline in SMs quickly turns to be idle (highlighted as red circles in Figure 3(a)) since other active threads in the SM will stall at the memory instructions in the near future as well.

The thread throttling mechanism has been proposed recently to alleviate the memory contentions and shrink the pipeline idle time [18]. It dynamically stalls certain threads to restrict the number of concurrent memory tasks and avoid the interferences among memory requests. As can be seen, appropriately slowing down the threads before their memory accesses can even introduce positive effect on performance. Allocating the TFET-based registers to those threads and managing them to execute at low frequency during the register read/write operations provides the perfect approach to control the thread progress. Figure 3(b) demonstrates the example of intelligently leveraging the low frequency operations on TFETs to absorb the pipeline stall time (shown in green rectangles) and meanwhile, separate the memory requests from SMs. As it shows, both NoC and memory controllers have few queued requests, and the off-chip memory access time reduces significantly. More importantly, the benefit of TFETs on reducing both dynamic and leakage energy is effectively explored. Obviously, CMOS-based registers are essential during the normal execution. In this work, we propose the hybrid CMOS-TFET based registers, and use TFET-based registers to delay threads execution speed to the maximal degree so that achieve the goal of maximizing the energy savings without hurting the performance.

3.2. Memory Contention-Aware TFET Register Allocation

As described in Section 2.1., when launching threads to the SM, a number of registers are statically designated to them according to their resource requirements. The register ID encoded in the instruction is used as the index to the physical register being read/written. In other words, the mapping between the register ID and the physical register is fixed all the time. However, when applying the same mapping mechanism in the hybrid register, the use of the TFET-based registers cannot be managed at the run time.

In this work, a register renaming table is applied to record the physical register number corresponding to the register ID encoded in the instruction. A register renaming stage is inserted into the SM pipeline following the decode stage. Note that this additional stage does not affect back-to-back instruction latencies. It only induces 1.5% performance overhead based on our evaluation across a large set of GPGPU benchmarks (detailed experimental methodologies are described in Section 4.1.), which also matches the observation made in [4]. During the register renaming stage, the destination register ID is renamed to a free physical register. The renaming table also provides the information of physical registers to be read according to the source register IDs. Therefore, the thread has the flexibility to map a register to either CMOS or TFET based physical register. A register in the renaming table is released after its last read, and the register lifetime information can be simply obtained by the compiler which indicates the last instruction reading the register. Since threads in a single warp execute the same instruction, they share the same renaming information. The execution of a branch instruction may cause warp divergence, threads in a diverged warp execute in serial fashion. A physical register will not be released until the last read finishes across all threads in the warp.

The critical challenge in the hybrid register design becomes the runtime CMOS/TFET physical register allocation to the destination register ID in the warp. Aggressively utilizing the TFET registers degrades the performance significantly; on the other hand, too conservatively using the TFET registers fails to achieve the goal of maximizing the registers power savings. Moreover, the TFET utilization among warps needs to be different to well control the warp execution progress and avoid the interferences. As can be seen, it is crucial that the TFET-based register allocation adapts to the memory access pattern of the workloads. For example, randomly or periodically renaming the destination registers to TFET registers can easily hurt the performance as they are blind to the memory accesses. It is highly possible that the TFET registers are improperly used when there are few memory transactions and the high throughput is expected during that period of the workload execution. We propose the MEMory contention-aware TFET Register Allocation (named as MEM_RA as abbreviation) to achieve the optimal power savings with little performance penalty.

Recall that SM supports the SPMD execution model, threads from a warp exhibit the same progress and stall for the same amount of time, therefore, the stall time at warp level is the finest granularity can be considered. The warp stall time due to the off-chip memory access implies the severity of the memory contentions. A long waiting time means the occurrence of serious contentions, and if the memory request from the warp had been postponed by using the TFET registers, such contentions may be removed successfully. Unfortunately, the waiting time is not available until the request has already been serviced and the contentions already take place. We use the last value prediction mechanism to predict the warp stall time in its next global memory transaction based on the previous memory access latency, and utilize TFET registers to absorb that predicted stall time before the warp sends out its memory request.

Note that the warp has already been slowed down to some degree in previous memory transaction, its following memory request might not interfere with others and it is unnecessary to further delay its progress. This happens in kernels with heavy computation tasks which help to separate the memory transactions and relief the memory contentions. However, the case is different in memory-intensive workloads. Even the warp has been delayed before, its following memory access can get involved with memory transactions from other warps due to the frequently issued memory requests, and further postponing its execution progress is desired.

In order to delay the warp appropriately across various types of workloads, we sample the memory access latency periodically at run time and introduce it into the warp stall time prediction. Eq.1 describes the analytical model to predict the stall cycles (represented as *SC*) of a warp based on its previous memory access latency (represented by *prev_acc*) and the latest sampled memory access latency (represented by *sample_acc*),

$$SC=\begin{cases} 0, & \text{if } prev\_acc \le thr\_acc \\ \left[\dfrac{sample\_acc}{ref\_acc}\times(prev\_acc-thr\_acc)\right], & \\ & \text{if } sample\_acc<ref\_acc, prev\_acc>thr\_acc \\ prev\_acc-thr\_acc, & \text{if } sample\_acc\ge ref\_acc, prev\_acc>thr\_acc \end{cases} \qquad \text{Eq.1}$$

where *thr_acc* is the threshold latency to determine whether the warp should be delayed in the near future. It is set as the memory access cycles under perfect memory system (e.g. 10 core cycles in our GPU machine configuration). When the *prev_acc* is no longer than the *thr_acc*, it implies that the previous memory transaction does not run into any congestion and the warp proceeds at good speed rate, so no delay is required. *ref_acc* is the referred memory access latency describing the memory access time with moderate resource contentions. When *sample_acc* is longer than *ref_acc*, it implies that the kernel currently exhibits the memory-intensive characteristic, the aggressive delay on the warp execution is preferred. The stall cycle is directly set as the extra waiting time in the previous memory access (i.e. *prev_acc* minus *thr_acc*). To the contrary, a short *sample_acc* compared to *ref_acc* means that the kernel involves heavier computation tasks, the predicted stall time is scaled down according to the ratio of *sample_acc* to *ref_acc*.

Once the stall time is calculated by using the analytical model above, the warp starts to allocate TFET-based registers to the destination register IDs in its following execution. Generally, the read/write time to TFET-based SRAM operating at low supply voltage is as twice as that of the CMOS-based SRAM at normal voltage [19]. The access time to TFET registers is modeled as 2 cycles in our study. In other words, one extra cycle is required to finish the TFET register read/write operation. The TFET register allocation is disabled when the predicted stall time is expected to be fully absorbed. Note that the register read time lasts 2 cycles as long as there is one TFET-based source register. When a warp diverges at a branch instruction, the extra delay is also modeled for all the sequentially executed threads if they use TFET registers. A warp issues multiple memory transactions when a load instruction is executed and the load requests from threads belonging to that warp fail to get coalesced. Those transactions may complete at different time, as a result, the register write back cannot be performed concurrently. Writing values to TFET registers in a load instruction is likely to induce quite long delay which easily makes the warp over-postponed. The TFET register allocation for load instructions are skipped in MEM_RA.

3.3. Implementation

(1) The Number of the TFET-Based Registers

Since CMOS- and TFET-based SRAMs have similar size [29], we set the total amount of hybrid registers in each SM as the same as that (i.e. 16K) in the baseline case with default GPU configuration for the fair comparison. The partition of CMOS- and TFET-based registers is important to the effectiveness of our proposed MEM_RA mechanism. Fabricating the sizeable TFET registers forces the use of TFET registers when there are insufficient CMOS registers, it reduces power by sacrificing the high computational throughput; while the small TFET registers cannot provide enough TFETs for the energy saving purpose. In the ideal case, the number of CMOS-based registers should perfectly matches their utilization under the impact of MEM_RA, which is largely determined by the warp waiting time during the off-chip memory accesses. The quantity of TFET registers may be more than required in the ideal case, it is better to have idled TFET-based instead of CMOS-based registers considering the extremely low leakage power consumed by TFET circuits.

Figure 4 shows the percentage of the warp stall time to its total execution time in various types of GPGPU benchmarks, the detailed experimental setup is described in Section 4. In the computation-intensive benchmarks (e.g. CP, LPS, MM, and RAY), there are few memory accesses, and the warp stall time is very close to zero. While the numerous memory transactions in the memory-intensive benchmarks (e.g. BFS, BP, MT, NE, and NW) causes much longer warp stall time. On average across all the benchmarks, the stall time is around 22%. In other words, CMOS registers should be applied in the remaining 78% of the execution time. Therefore, the CMOS registers are designed to account for 78% of the total registers, and the remains are TFET-based registers. Our 16K hybrid registers are composed of 12.5K CMOS-based and 3.5K TFET-based registers. We also performed detailed sensitivity analysis on varying the size of CMOS registers (e.g. 6K, 10K, and 14K) in the total 16K hybrid design, and found that 12.5K CMOS register is the optimal design regarding to the total energy saving and performance overhead.

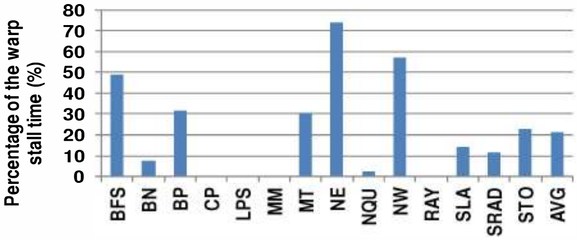


Figure 4. The percentage of the warp stall time caused by the off-chip memory accesses. It is near zero in CP, LPS, MM, and RAY.

In GPU SM, the per-block resources (e.g. registers, shared memory) are not released until all the threads in the block finish execution, they limit the number of blocks that can simultaneously run in the SM. Different per-block resources become the bottleneck for kernels that have different resource requirements. The bottleneck structure is prone to be fully utilized while other structures are usually underutilized. Therefore, a portion of CMOS registers may be free through the entire kernel execution, leading to the considerable leakage power consumption. In [2], the power gating technique has been introduced into GPU SM to remove leakage. We apply it to power off the unused CMOS registers in SMs. Information such as the maximum number of threads allocated to each SM, and the quantity of physical registers required per thread can be easily obtained during the kernel launch process. Hence, the total register utilization would not exceed the product of those two factors. The requirement on CMOS registers can be estimated by scaling down the total register utilization to 78%, and the power gating is enabled on the remaining idled CMOS registers for a long time until the kernel completes. The energy and time overhead caused by the power gating is negligible with regard to the large power reduction by keeping those registers in the power-gated mode during the entire kernel execution period.

(2) The Implementation of MEM_RA

Figure 5 demonstrates the implementation of our proposed memory contention-aware TFET-based register allocation in the hybrid register design. A counter is attached to each warp slot in the warp scheduler. When a warp encounters an off-chip memory access, its counter is re-set as zero and starts the auto increase every cycle to record the memory access latency. Upon the completeness of the memory transaction, the cycle number stored in the counter is sent to an ALU for the warp stall time prediction, meanwhile, the sampled memory access cycles with the static information (i.e. threshold latency, referred access latency) also input to the ALU. The output is written back to the counter, it will be read when the warp enters into the pipeline, and a larger-than-one value in the counter implies the necessity of writing to the TFET-based register. In [4], Gebhart et al. found that 70% of the register values are read only once in GPGPU workloads. It implies that most TFET register values are read once, therefore, renaming the destination register to the TFET register usually causes 2-cycle extra delay: one additional cycle during the value write back, and another one when it is read by a subsequent instruction. As can be seen, one TFET register allocation takes two cycles of the warp stall time in most cases. And the counter value will decrease by two upon a successful TFET register allocation. Note that the counter decrease is just used to estimate the possible delay to the warp when renaming to the TFET registers. For TFET registers being read multiple times, the warp stall time will be taken more than two cycles. Moreover, the counter auto-increases occurs at warp waiting time while its value decrease is performed at the normal execution time, there is no overlap between the counter auto-increase and decrease processes.

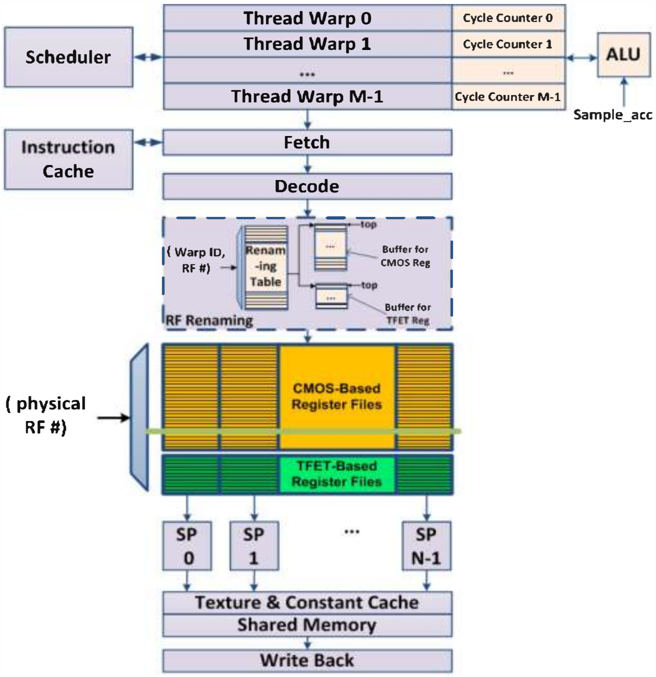


Figure 5. Memory contention-aware TFET-based register allocation

Considering that the ALU is used once a warp completes a memory instruction, and the major computation in it is division (as shown in Eq.1.) lasting for tens of cycles, we set the referred access latency as 2 to power of n and translate the division into logical shift. It will operate based on the product of the sampled memory access latency and the previous stall time. We performed the detailed sensitivity analysis on the referred access latency, and found that MEM_RA achieves optimal trade-off between power and performance when setting it as 2^7=128 cycles. Note that the warp stall time estimation occurs in parallel with the write back stage, it does not introduce any extra delay to the critical path in the pipeline.

As Figure 5 shows, register files are partitioned into CMOS-based and TFET-based registers, and each physical register vector has a unique identification number. Two power supply lines are used to support the high (low) voltage operations on CMOS (TFETs) registers. The register renaming stage is added into the SM pipeline (shown as the dotted rectangle), during which the register renaming table is accessed. It is indexed by the warp ID and register number encoded in the instruction, and each entry holds the corresponding physical register vector number which will be used for register access in the following stage. Two FIFO buffers are attached to the renaming table to keep the released CMOS and TFET register vectors, respectively. The top register in each buffer is consumed for the renaming, while the bottom is filled by the newly released register. In the case that a CMOS register is requested while the buffer for CMOS registers is empty, the buffer for TFET registers will provide a free TFET register instead, and vice versa. Note that there is always at least one free CMOS/TFET register available for renaming since the required resources have already been well estimated when the block is assigned to the SM.

(3) Hardware and Power Overhead

The major hardware added into the SM is the register renaming pipeline stage including the register renaming table, two buffers for the released CMOS and TFET register vectors, and some simple combinational logics. In order to keep the renaming information for all physical registers, the number of entries in the renaming table is equal to the amount of register vectors which is 512 in our default GPU configuration. Similarly, the total size of the two buffers is 512 as well. Each entry in those three structures contains 9 bits. The hardware in the renaming stage causes around 2% area overhead to the register files in the SM. In addition, to predict the warp stall time, thirty-two 11-bit counters (we set the maximum memory access time as 2048 cycles), and the unit performing simple integer arithmetic and logic operations are added in the SM. The overall hardware overhead to the SM register files is 3%. We develop the power model (including both dynamic and leakage power) for the added hardware, and find that it induces around 2.9% power overhead to the register files by running a large set of GPGPU benchmarks.

4. Evaluations

4.1. Experimental Methodology

We implement our MEM_RA technique on the cycle-accurate, open-source, and publicly available simulator GPGPU-Sim [20] to obtain the GPGPU performance statistics. We build our power model based on the energy analysis tool CACTI [21]. We set the high supply voltage as 0.7V and low supply voltage as 0.3V. The read/write times to CMOS- and TFET-based registers and the total execution time are collected from the modified GPGPU-Sim to evaluate both RF dynamic and leakage energy consumption. Our energy estimation is consistent with previous studies [3, 4, 8].

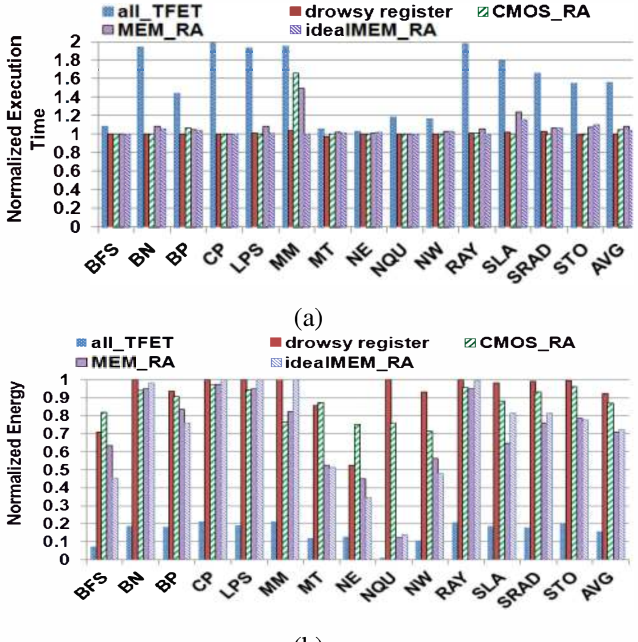
Our baseline GPGPU configuration is set as follows: there are 28 SMs in the GPU, SM pipeline width is 32, warp size is 32, each SM supports 1024 threads and 8 blocks at most, each SM contains 16K 32-bit registers, 16KB shared memory, 8KB constant cache, and 64KB texture cache, the warp scheduler applies the round robin scheduling policy, the immediate post-dominator reconvergence [22] is used to handle the branch divergences; the GPU includes 8 DRAM controllers, each controller has a 32-entry input buffer, and applies out-of-order first-ready first-come first-serve scheduling policy [20]; the interconnect topologies is Mesh, and the dimension order routing algorithm is used in the interconnect, the interconnect router contains two virtual channels, and flit size is 16B. We collect a large set of available GPGPU workloads from Nvidia CUDA SDK [23], Rodinia Benchmark [24], Parboil Benchmark [25] and some third party applications. The workloads show significant diversity according to their kernel characteristics, branch divergence characteristics, memory access patterns, and so on.

4.2. Results

In order to justify the effectiveness of MEM_RA, we compare it with several power reduction techniques. The baseline case studied in this paper is employing only CMOS-based registers and power gating the unused registers during the kernel execution. Another naïve mechanism for power saving is simply applying TFETs to all SM registers, it is named as all_TFET. In previous work, the drowsy cache has been proposed to reduce the cache leakage power [26]. Similarly, registers belonging to a warp can be put into the sleep mode when the warp stalls in the pipeline, but it takes couple of cycles to wake them up for further accesses. We also investigate the effect of drowsy register from the performance and power perspectives. In the hybrid register design, the long access time to TFET registers may largely degrade the performance when they are randomly used. A straightforward technique to maintain performance is to avoid the allocation of TFET registers if possible. In other words, the CMOS register is selected for renaming as long as there is any one free. We name this technique as CMOS_RA, it is applied on the hybrid 12.5K CMOS registers and 3.5K TFET registers. Note that the power gating technique is integrated into drowsy register and CMOS_RA, respectively, for the fair comparison. Since TFET has extremely low leakage power, the power gating is not triggered in all_TFET mechanism.

As discussed in Section 3.3., ideally, the size of CMOS registers would exactly match their usage. We further investigate the effectiveness of MEM_RA when the CMOS registers size is set ideally, called idealMEM_RA. (We name the MEM_RA using 12.5K CMOS and 3.5K TFET registers as MEM_RA for short.) Since benchmarks exhibit different memory access patterns, their requirements on CMOS registers vary greatly. Although designers rarely fabricate a GPU with certain number of CMOS(TFET) registers to specifically satisfy a single benchmark's requirement, the results of idealMEM_RA provide a more accurate evaluation on the capability of MEM_RA on power optimizations while maintaining the performance.

Figures 6 describes (a) the execution time and (b) the overall energy when running the investigated benchmarks under the impact of several power reduction techniques described above. The results are normalized to the baseline case. Note that the performance and energy overhead caused by each technique is also included in the results. As Figure 6(a) shows, all_TFET hurts the GPU performance significantly, the execution time is almost doubled in several benchmarks (e.g. BN, CP, MM, and RAY). On average, all_TFET degrades the performance by 56%. Although it reduces the energy consumption significantly (total energy decreases to 16% as shown in Figure 6(b)), it is not worth to scarify such large portion of throughput to achieve the low energy consumption. Interestingly, the kernel execution time under drowsy register mechanism remains the same although there is time overhead to wake up registers staying in the sleep mode, because the wake up time is trivial with regard to the hundred-cycle long memory access. Moreover, the energy reduction achieved by drowsy register is small, only around 7%.



(a)

(b)

Figure 6. The (a) normalized execution time (b) normalized energy consumption when running benchmarks under several power optimization techniques

CMOS_RA is performance friendly which causes 5% performance penalty on average. Because it uses the CMOS registers in majority of the time, and there is no performance loss when the benchmark needs less than 12.5K registers. However, the performance penalty is high for benchmarks requesting a large amount of registers, as TFET registers are consumed in that case. For example, the execution time for MM increases 65% under CMOS_RA because the RF utilization in that benchmark is 100%.

As Figure 6 shows, the energy reduction under CMOS_RA is 13%, while MEM_RA is able to achieve 30% energy savings with similar performance loss (i.e. 8%). Such 30% energy reduction contributes to around 5% energy savings to the entire SM, which is already considered as the noticeable energy optimization as discussed in [4]. Different from CMOS_RA, MEM_RA intelligently migrates the resource usage from CMOS to TFET registers which reduces the total dynamic energy. Meanwhile, the extra access delay in TFETs absorbs the warp waiting time and prevents the interferences among memory requests which minimizes the impact on performance. Especially for the memory-intensive benchmarks, such as the BFS, BP, MT, NE, and NW, MEM_RA generally reduces the energy by 42% with only 2.5% performance loss.

One may notice that MEM_RA introduces the long execution time in MM as well. Because MM fully utilizes the RF resources and contains quite few memory accesses to trigger the memory-contention aware TFET register allocation. As Figure 6(a) demonstrates, the performance of MM maintains the same under idealMEM_RA since the GPU will be equipped with all CMOS registers if running such type of benchmarks, and it cannot reduce the energy. On average, idealMEM_RA slightly outperforms MEM_RA on performance but meanwhile, obtains less energy savings. In summary, MEM_RA successfully explores the energy-efficient GPGPUs and its effectiveness is quite close to that of idealMEM_RA.

The performance degradation in SLA is noticeable under MEM_RA and idealMEM_RA. Because they use the last memory access latency to predict the warp waiting time and enable the TFET register allocation correspondingly, the prediction accuracy is affected when the next memory access pattern differs greatly from the last one. As a result, the TFET registers are excessive utilized which hurts the performance. Generally, the last value prediction mechanism achieves pretty high accuracy for most benchmarks and helps MEM_RA to minimize the performance penalty.

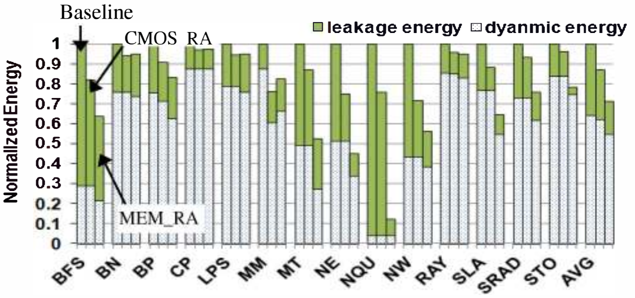


Figure 7. Dynamic and leakage energy consumptions under baseline case, CMOS_RA, and MEM_RA.

We further split the normalized overall energy obtained by MEM_RA into the dynamic and leakage portions and present them in Figure 7. The energy partition under the baseline case and CMOS_RA is also included in the figure. As it shows, CMOS_RA can barely optimize the dynamic power since the CMOS register are frequently accessed. MEM_RA exhibits strong capability in reducing not only leakage but also dynamic energy. On average, the dynamic energy reduction compared to the baseline case is 10%, while the leakage decreases 20%. In addition, the dynamic (leakage) energy savings in memory-intensive benchmarks is 16% (26%).

5. Related Work

There have been several studies on building hybrid storage-cell based structure and furthermore, heterogeneous multi-core processors based on CMOS and TFETs to achieve the good trade-off between performance and power [13, 19, 27, 28]. For instance, Narayanan et al. [19] developed the hybrid cache architecture that uses a mix of TFET and the non-volatile memory. Swaminathan et al. [13] proposed to replace some of the CMOS cores with TFET alternatives, and dynamically migrate threads between CMOS and TFET cores to achieve significant energy savings with negligible performance loss. We build the hybrid registers in GPGPU and leverage its unique characteristics to fully explore the benefit of TFETs for the energy-efficient GPGPU design.

Many methodologies have been proposed recently to reduce the GPGPU registers dynamic power. Gebhart et al. [4] proposed register file caching and two-level thread scheduler to reduce the number of reads and writes to the large main register file and save its dynamic energy. The authors further extended their work to the compiler level and explored register allocation algorithms to improve register energy efficiency [5]. Yu et al. integrated embedded DRAM and SRAM cells to reduce area and energy [3]. In addition, several works have been done on GPGPU register leakage power optimization. Chu et al. [6] explored the fine granularity clock gating scheme for registers. Wang et al. [2] adopted the power gating technique at architecture level for leakage reduction on GPGPUs. Our technique targets on both dynamic and leakage savings and it is orthogonal to the techniques discussed above.

6. Conclusions

Modern GPGPU employs the fine-grained multi-threading among numerous active threads which leads to the large register files consuming massive dynamic and leakage power. Exploring the optimal power savings in register files become the critical and first step towards the energy-efficient GPGPU. The conventional method to reduce dynamic power is to scale down the supply voltage which causes substantial leakage in CMOS circuits. The TFETs are the promising candidates for low voltage operations regarding to both leakage and performance. However, always executing at the low voltage (so that low frequency) will result in significant performance degradation. In this study, we propose the hybrid CMOS-TFET based register files. We leverage the unique characteristics of GPUs during the off-chip memory accesses, and explore the memory contention-aware TFET register allocation (MEM_RA) to make use of TFET registers in alleviating the memory contentions, and meanwhile gaining the attractive energy optimization. Our experiment results show that MEM_RA obtains 30% energy (including both dynamic and leakage) reduction in register files compared to the baseline case with power gating technique. Especially, it achieves 42% energy savings in memory-intensive benchmarks with only 2.5% performance loss.

Acknowledgement

This work is supported by the National Science Foundation under Award No. EPS - 0903806 and matching support from the State of Kansas through the Kansas Board of Regents.

References

[1] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, and D. Glasco. GPUs and the Future of Parallel Computing. IEEE Micro, 31:7–17, September 2011.

[2] P. Wang, C. Yang, Y. Chen, and Y. Cheng. Power Gating Strageties on GPUs, ACM Transaction on Architecture and Code Optimization (TACO), Volume 8 Issue 3, October 2011.

[3] W. Yu, R. Huang, S. Xu, S.-E. Wang, E. Kan, and G. E. Suh. SRAM-DRAM Hybrid Memory with Applications to Efficient Register Files in Fine-Grained Multi-Threading. In Proceedings of ISCA, 2011.

[4] M. Gebhart, D. R. Johnson, D. Tarjan, S. W. Keckler, W. J. Dally, E. Lindholm, and K. Skadron. Energy-Efficient Mechanisms for Managing Thread Context in Throughput Processors, In Proceedings of ISCA, 2011.

[5] M. Gebhart, S. W. Keckler, and W. J. Dally. A Compile-Time Managed Multi-Level register File hierarchy. In Proceedings of MICRO, 2011.

[6] S. Chu, C. Hsiao, and C. Hsieh. An Energy-Efficient Unified Register File for Mobile GPUs. In Proceedings of IFIP 9th International Conference on Embedded and Ubiquitous Computing, October 2011.

[7] D. Kanter. Inside Fermi: Nvidia's HPC Push, 2009. http://www.realworldtech.com/page.cfm?ArticleID=RWT093009110932.

[8] S. Hong and H. Kim. An Integrated GPU Power and Performance Model. In Proceedings of ISCA, 2010.

[9] S. Mookerjea, D. Mohata, R. Krishnan, J. Singh, A. Vallett, A. Ali, T. Mayer, V. Narayanan, D. Schlom, A. Liu, and S. Datta. Experimental Demonstration of 100nm Channel Length In0.53ga0.47as-based Vertical Inter-Band Tunnel Field Effect Transistors (TFETs) for Ultra Low-Power Logic and Sram Applications. In Proc. IEEE Int. Electron Devices Meeting (IEDM), 2009, pp. 1–3.

[10] N. N. Mojumder and K. Roy. Band-to-Band Tunneling Ballistic Nanowire FET: Circuit-Compatible Device Modeling and Design of Ultra-Low-Power Digital Circuits and Memories. IEEE Transactions On Electron Devices, vol. 56, pp. 2193–2201, 2009.

[11] NVIDIA. CUDA Programming Guide Version 3.0., Nvidia Corporation, 2010.

[12] Y. Taur and T. H. Ning. Fundamentals of Modern VLSI Devices. Cambridge University Press, 2009.

[13] K. Swaminathan, E. Kultursay, V. Saripalli, V. Narayanan, M. Kandemir, and S. Datta. Improving Energy Efficiety of Multi-Threaded Applications Using Heterogeneous CMOS-TFET Multicores. In Proceedings of ISLPED, 2011.

[14] J. Singh, K. Ramakrishnan, S. Mookerjea, S. Datta, N. Vijaykrishman and D. Pradhan. A Novel Si Tunnel FET based SRAM Design for Ultra Low-Power 0.3V Vdd Applications. In Proceedings of ASPDAC, 2010.

[15] V. Saripalli, S. Datta, V. Narayanan and J. P. Kulkarni. Variation-Tolerant Ultra Low-Power Heterojunction Tunnel FET SRAM Design. In Proceedings of International Symposium on Nanoscal Architectures, 2011.

[16] D. Kim, Y. Lee, J. Cai, I. Lauer and L. Chang, S. J. Koester, D. Sylvester and D. Blaauw. Low Power Circuit Design Based on Heterojunction Tunneling Transistors (HETTs). In Proceedings of ISLPED, 2009.

[17] X. Yang and K. Mohanram. Robust 6T Si tunneling transistor SRAM design. In proceedings of DATE, 2011.

[18] H. Cheng, C. Lin, J. Li, and C. Yang. Memory Latency Reduction via Thread Throttling. In Proceedings of MICRO, 2010.

[19] V. Narayanan, V. Saripalli, K. Swaminathan, R. Mukundrajan, G. Sun, Y. Xie, and S. Datta. Enabling Architectual Innovations Using Non-Volatile Memory. In Proceedings of GLSVLSI, 2011.

[20] A. Bakhoda, G.L. Yuan, W. W. L. Fung, H. Wong, Tor M. Aamodt, Analyzing CUDA Workloads Using a Detailed GPU Simulator, In Proceedings of ISPASS, 2009.

[21] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi. Cacti 5.1. HP Labs, Tech. Rep. 2008.

[22] S. S.Muchnick. Advanced Compiler Design and Implementation. Morgan Kaufmanns, 1997.

[23] http://www.nvidia.com/object/cuda_sdks.html

[24] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, S. Lee, and K. Skadron. Rodinia: A Benchmark Suite for Heterogeneous Computing, In Proceedings of IISWC, 2009.

[25] Parboil Benchmark suite. URL: http://impact.crhc.illinois.edu/parboil.php

[26] K. Flautner, N.S. Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy Caches: Simple Techniques for Reducing Leakage Power. In Proceedings of ISCA, 2002.

[27] V. Saripalli, A. Mishra, S. Datta and V. Narayanan. An Energy-Efficient Heterogeneous CMP based on Hybrid TFET-CMOS Cores. In Proceedings of DAC, 2011.

[28] V. Saripalli, G. Sun, A. Misha, Y. Xie, S. Datta and V. Narayanan. Exploiting Heterogeneity for Energy Efficiency in Chip Multiprocessors. IEEE Trans on Emerging and Selected Topics in Circuits and Systems, vol. 1, pp. 109-119, June 2011.

[29] A. Pal, A. B. Sachid, H. Gossner, and V. R. Rao. Insights Into the Design and Optimization of Tunnel-FET Devices and Circuits. IEEE Trans on Electron Devices, vol. 58, NO. 4, April 2011.

[30] V. Sathish, M. J. Schulte, and N. S. Kim. Lossless and Lossy Memory I/O Link Compression for Improving Performance of GPGPU Workloads, In Proceedings of PACT, 2012.