Core-level DVFS for Spatial Multitasking GPUs

Jehee Cha
Electronic and Electric Engineering Department
Hongik University
Seoul, Korea
carjehee@gmail.com

Jiho Kim
Electronic and Electric Engineering Department
Hongik University
Seoul, Korea
jihokimhi@gmail.com

Yongjun Park
Department of Computer science
Hanyang University
Seoul, Korea
yongjunpark@hanyang.ac.kr

***Abstract*—DVFS (Dynamic voltage frequency scaling) is one of the most widely used power management technologies employed to improve the performance or minimize the power consumption by controlling voltages and frequencies in real time. When applying device-level DVFS in graphics processing units (GPUs) that support spatial multitasking, it is difficult to determine the optimal DVFS status when multiple running kernels have different characteristics. To solve the problem, we created a GPU simulator that can operate at different streaming multiprocessor frequencies according to the characteristics of the assigned kernel and compared it with a single-clock-based spatial multitasking GPU simulator.**

Keywords—GPU, Multitasking, DVFS

I. INTRODUCTION

With the growth of multimedia and information technology, the amount of data to be processed by mobile APs has increased. Modern mobile APs are equipped with shared graphics processing units (GPUs) and multiple central processing unit (CPU) cores. For example, Samsung's mobile AP Exynos embeds 8-core CPUs and one shared GPU. GPU is designed to perform simple operations for massive input data, and it has many simple processors called streaming multiprocessors (SMs). GPU was originally an accelerator that simply accelerated graphic operations, but it has been recently used to handle a large amount of input data and has become a general computation unit in big-data fields such as data mining and deep learning.

When a GPU runs several tasks sequentially, if it first runs a very long task, the subsequent tasks will have to wait until the long task is finished. Therefore, latency of the following tasks become too high and they might miss the latency deadline. However, for GPUs that support spatial multitasking, subsequent tasks do not have to wait until the first long-running task is finished. Therefore, a spatial multitasking environment in which multiple applications can be performed simultaneously is highly preferred.

Dynamic voltage frequency scaling (DVFS) is a well-known technique that accelerates program execution or minimizes unnecessary power consumption by changing voltage and frequency in real time. DVFS in GPUs has recently been studied. For example, NVIDIA's GPUs are designed to increase the frequency of SM to a predetermined level of power consumption or temperature when high performance is required [1]. NVIDIA's Kepler architecture GPU model supports a frequency boost of approximately 15% [2]. Equalizer has shown that DVFS can be applied to memory, L2 cache, and interconnection network as well as core, thereby improving performance and reducing power consumption [3].

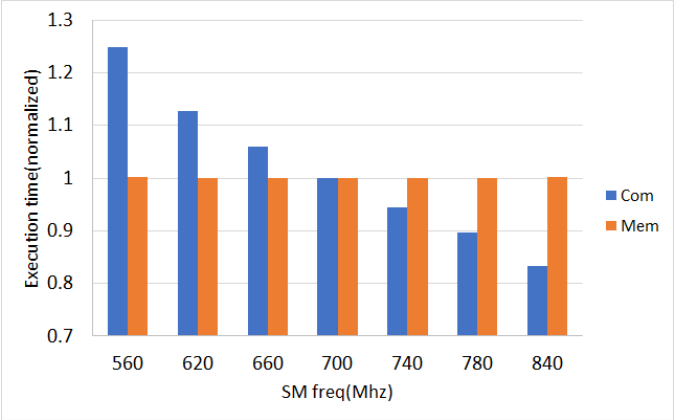


Fig. 1. Effect of SM frequency on execution time.

Figure 1 shows the result of execution of a compute-intensive application (Mri-q) and a memory-intensive application (Lattice Boltzmann method) from the parboil benchmark suite [5] with varying SM frequencies. In compute-intensive applications, SM is a main performance bottleneck, and therefore, an increase in the SM frequency results in high performance enhancement. However, as the performance of a memory-intensive application depends mainly on cache or memory performance and the SM frequency is not a main performance bottleneck, the performance will not change even when the SM frequency is reduced.

Therefore, when applications of different characteristics are running simultaneously in a spatial multitasking GPU, an increase in the SM frequency increases only the performance of compute-intensive applications and the performance of

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on March 04,2026 at 01:58:53 UTC from IEEE Xplore. Restrictions apply.
978-1-5386-5457-6/18/$31.00 ©2018 IEEE

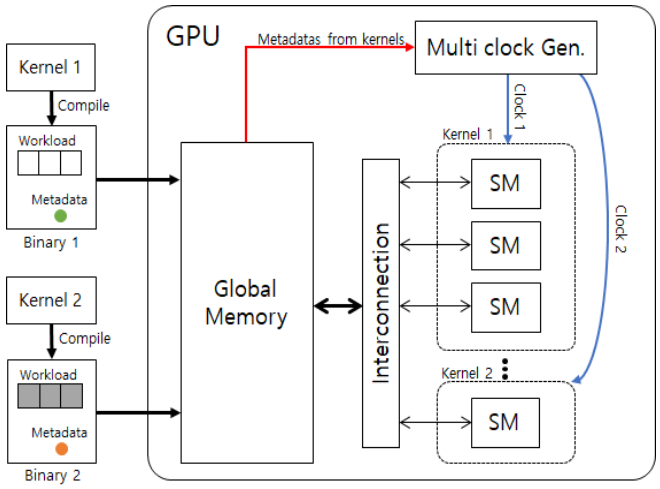


Fig. 2. Multi-clock spatial multitasking GPU simulator overview

Algorithm 1 Decision algorithm of SM frequency

nSM, SM[m] are the number of SMs in GPU and m-th SM
metaReg[m] is m-th bit of the n-bit metadata register

```
1:  for Each SM do
2:      if SM[m]_has_kernel() then
3:          if metaReg[m] is 1 then          › SM[m] is mem
4:              SM[m]'s CLK = slow
5:          else                             › SM[m] is com
6:              SM[m]'s CLK = fast
7:          end if
8:      else                                 › SM[m] is empty
9:          SM[m]'s CLK = base
10:     end if
11: end for
```

memory-intensive applications is not affected. This causes inefficient power performance ratios when the system supports device-level DVFS. To solve this problem, we propose a core-level DVFS system to set the operating clock frequency in core granularity based on the characteristics of the application allocated to the SM in the GPU supporting the space-multitasking.

Therefore, we modified GPGPU-Sim v3.2.2[4] to create a space-multitasking GPU simulator that has multiple clock generators and allows them to operate at different frequencies depending on the characteristics of the application assigned to the core. We also compared it with a space multitasking GPU simulator that operates at a single frequency.

II. Multi-clock-based Spatial Multitasking GPUs

In this section, we introduce a GPU simulator to support core-level DVFS. First, we explain the high-level architecture of the simulator, and subsequently explain the multi-clock generator with the algorithm.

A. Overview

Figure 2 shows the overview of the multi-clock spatial multitasking GPU simulator. First, metadata about kernel characteristics are added to the execution binary file in the kernel compilation process. Metadata about the characteristics of the kernel can be obtained either by the programmer (to add the information directly to the original OpenCL/CUDA code), or by the compiler (based on static/dynamic profiling process). If the compiler writes metadata through profiling, it has some compilation time overhead, but it can minimize the programmer efforts to determine the application characteristics and manually add the information to the original source code. These metadata are stored in a special register for each SM when loading the kernel binary to a target GPU and target SMs for the kernel are assigned. Subsequently, in the next cycle, the multi-clock-generator assigns the appropriate frequency to each SM according to the characteristics of the kernel based on the saved metadata.

B. Multi-clock Generator

The multi-clock generator consists of a register, two clock generators, and two 2-input multiplexers. Algorithm 1 shows how the multi-clock generator determines the frequency of a target SM. Metadata for multiple kernel properties are stored in a register in the multi-clock generator. The register has the same number of bits as the number of SMs in the GPU. When allocating a thread block to an empty SM, it stores the metadata in bits of the register that match the SM number. The bit "1" in the register indicates that the assigned kernel is memory-intensive, and "0" indicates that it is compute-intensive. If there is no thread block assigned in the SM, it operates at the baseline frequency. Therefore, in the idle state, it operates at the baseline frequency.

III. Experimental Result

A. Methodology

We used a modified GPGPU-Sim v3.2.2[4] for concurrent kernel execution. The GPU configuration parameters for evaluation are listed in Table I. We used target benchmark applications from Rodinia suite[6], Parboil suite[5] and Polybench suite [7]. The characteristics of the applications used in this section are shown in Table II. Simulation was performed with multiple difference cases such as (Com + Com), (Mem + Mem), and (Mem + Com) according to the possible combinations with different kernel characteristics. We used the number of executed instructions per second as a metric for performance. IPC cannot be a metric for performance because the clock period of one cycle varies when the frequency changes.

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on March 04,2026 at 01:58:53 UTC from IEEE Xplore. Restrictions apply.

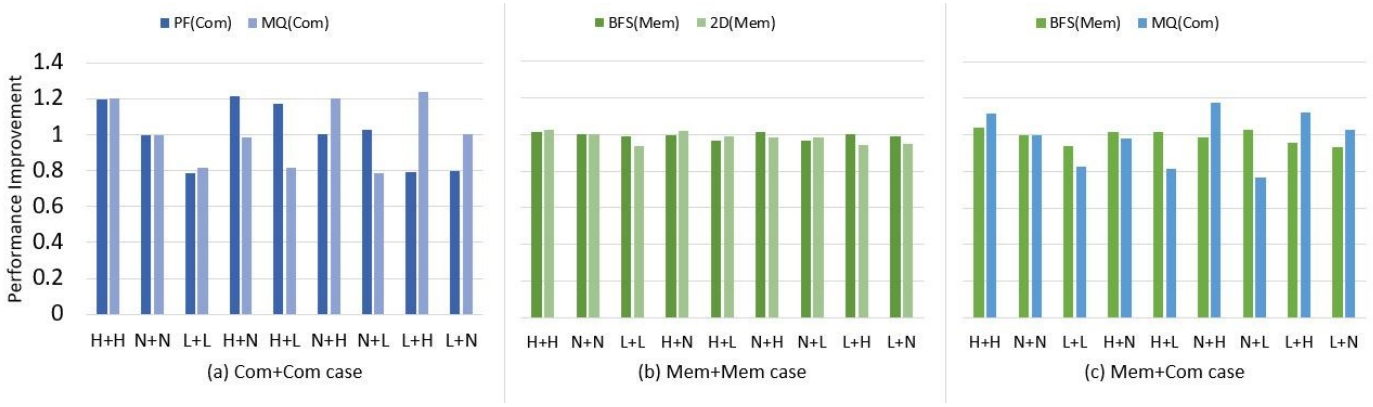


Fig. 3. Multi-clock spatial multitasking GPU simulator overview

TABLE I. GPU CONFIGURATION PARAMETERS

Architecture	Fermi
Number of SMs	15
SM Frequency	High (H) : 840MHz Normal (N) : 700MHz Low (L) : 560MHz
No. of Registers	32,768
Warp Size	32

TABLE II. CHARACTERISTICS OF APPLICATION

Name	Type	Suite
2Dconv (2D)	Mem	Polybench
Breadth First Search (BFS)	Mem	Rodinia
Mri-q (MQ)	Com	Parboil
Pathfinder (PF)	Com	Rodinia

B. Result

1) Com+Com Cases

Figure 3-(a) shows the total performance for all possible combinations of frequencies when executing two compute-intensive kernels simultaneously. Executing two compute-intensive kernels with a 20 % increase in the SM frequency (in the case of H+H) resulted in a 20.4% increase in the performance compared with the baseline (N+N). Conversely, when the SM frequency was reduced by 20% (L + L), the performance was reduced by 19.3% compared with the baseline. Based on all the possible combinations, it is observed that the performance has been changed linearly by the frequency of the SM where the kernel is running.

2) Mem+Mem Cases

Figure 3-(b) shows the performance of (Mem+Mem) cases. In this case, unlike the (Com + Com) case, the performance did not change significantly depending on the SM frequency. The biggest performance difference is shown in the case where the performance of 2DConvolution decreased by 6.5% compared with the baseline when the SM frequency was reduced by 20% (L + L). In the other cases, the performance change was less than 5%.

3) Mem+Com Cases

Figure 3-(c) shows the performance when two kernels with different characteristics are executed together. By increasing the frequency of all SMs by 20% (H + H), the performance of the memory-intensive kernels increased only by 3.7% compared with the baseline but the performance of the compute-intensive kernel increased by 11.5%. For the (L + L) case, the performance of the memory-intensive kernel and the compute-intensive kernel decreased by 5.8% and 17.3%, respectively. The performance of the (H + N) case was similar to the baseline. The performance of the (H + L) case was similar to that of the (N + L) case. In the (Mem + Com) configuration, the performance of compute-intensive kernel in the (N + H) case was the best, which improved the performance by 17.4% compared with the baseline. This performance gain is higher than that of the (H + H) case and power consumption is also expected to be less.

4) Power Efficiency Considation

In (Mem+Mem) and (Mem+Com) cases, the difference between the high and low SM frequencies of the memory-intensive kernel is 280 MHz, which corresponds to 1/4th of the baseline frequency. Compared with SM frequency difference of 1/4th of the baseline frequency, the performance difference between the best case and the worst case is only 10%. This is very inefficient in terms of power performance ratio. Therefore, when running the memory-intensive kernel, it is better not to increase the frequency of the SM in which the memory-intensive kernel is running.

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on March 04,2026 at 01:58:53 UTC from IEEE Xplore. Restrictions apply.

IV. Conclusion

In GPUs that support spatial multitasking, it was inefficient to increase the frequency of all SMs to improve the performance when running multiple applications with different characteristics because memory-intensive kernels are hardly affected by the SM frequency. To solve this problem, we proposed a core-level DVFS simulator that adjusts the frequency of the SM according to the characteristics of the assigned kernel. In the proposed simulator, the performance is measured with nine possible SM frequency combinations for various kernel combinations with different characteristics. Based on the evaluation, a compute-intensive kernel shows the highest performance gain up to 17% when its SM frequency was increased by 20% without changing the frequency of the memory-intensive kernel.

V. Future Work

Our GPU simulator requires an additional clock generator consisting of two 2-input multiplexers and several registers. Therefore, compared with typical device-level DVFS GPUs, the additional hardware overhead must be minimized. We will investigate methods to improve the performance and reduce the power consumption in more general cases using our simulator.

Acknowledgement

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP)(No.NRF-2015R1C1A1A01053844), ICT R&D program of MSIP/IITP (No.2017-0-00142), and the R&D program of MOTIE/KEIT (No.10077609).

References

[1] NVIDIA, GPU Boost 2.0. http://www.nvidia.in/object/nvidia-gpu-boost-2-in.html

[2] X. Mei, L. S. Yung et al., “A Measurement Study of GPU DVFS on Energy Conservation,”; in Proceedings of the Workshop on Power-Aware Computing and Systems, No.10, 2013

[3] A. Sethia and S. Mahlke, “Equalizer: Dynamic Tuning of GPU Resources for Efficient Execution,”; International Symposium on Mircoarchitecture (MICRO), 2014.

[4] A. Bakhoda, G. L. Yuan et al., “Analyzing CUDA Workloads Using a Detailed GPU Simulator,”; IEEE Symposium on Performance Analysis of Systems and Software, pp. 163—174, 2009.

[5] J. A. Strantton, C. Rodrigues et al., “Parboil: A Revised Benchmark Suite for Scientific and Commercial Throughput Computing,”; Center for Reliable and High-Performance Computing, 2012.

[6] S. Che et al. “Rodinia: A benchmark suite for heterogeneous computing,”; in Proceeding of the IEEE Symposium on Workload Characterization, pp. 44—54, 2009.

[7] L.-N. Pouchet, “Polybench: the polyhedral benchmark suite,”; URL: http://www.cs.ucla.edu/pouchet/software/polybench, 2012.

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on March 04,2026 at 01:58:53 UTC from IEEE Xplore. Restrictions apply.