RC25458 (WAT1404-038) April 14, 2014
Electrical Engineering

IBM Research Report

GPUVolt: Modeling and Characterizing Voltage Noise in GPU Architectures

Jingwen Leng, Yazhou Zu, Minsoo Rhu
University of Texas at Austin

Meeta Gupta
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
USA

Vijay Janapa Reddi
University of Texas at Austin

GPUVolt: Modeling and Characterizing Voltage Noise in GPU Architectures

Jingwen Leng, Yazhou Zu, Minsoo Rhu
Dept. of Eletrical and Computer Engineering
University of Texas at Austin
{jingwen, yazhou.zu, minsoo.rhu}@utexas.edu

Meeta Gupta
IBM T.J. Watson
msgupta@us.ibm.com

Vijay Janapa Reddi
Dept. of ECE
University of Texas at Austin
vj@ece.utexas.edu

Abstract

Voltage noise is a major obstacle in improving processor energy efficiency because it necessitates large operating voltage guardbands that increase overall power consumption and limit peak performance. Identifying the leading root causes of voltage noise is essential to minimize the unnecessary guardband and maximize the overall energy efficiency. We provide the first-ever characterization and modeling of voltage noise in GPUs based on a new simulation infrastructure called GPUVolt. *Using it, we identify the key intracore microarchitectural components (e.g., the register file, special functional units) that significantly impact the GPU's voltage noise. We also demonstrate that intercore-aligned microarchitectural activity detrimentally impacts the chip-wide worst-case voltage droops. On the basis of these findings, we propose a combined register-file/execution-unit throttling mechanism that smooths GPU voltage noise and reduces the guardband requirement by as much as 29%.*

1. Introduction

Voltage guardbands [1–3] have been a long-standing and established mechanism to ensure robust execution. By raising the voltage regulator's output from its nominal operating voltage (e.g., 20% in IBM POWER6 [4]), the processor is guaranteed to meet its frequency target under the worst-case operating conditions such as process, temperature and voltage variations, aging, etc. However, an over-provisioned guardband consumes additional power and limits peak performance [5].

Prior measurement results show that throttling the processor's frequency and voltage according to its runtime activity can on average reduce power consumption by 24% without violating program correctness [3], simply because worst-case conditions occur infrequently in the real world [6]. On the basis of such insightful characterization, several throttling mechanisms have been proposed that intelligently mitigate the worst-case voltage guardband requirement [1–3, 6–11]. A majority of these studies concluded that rapid current changes and resonant current behavior (e.g., the $L\frac{di}{dt}$ effect caused by quick increases in microarchitectural activities after pipeline stalls) are the major causes of voltage noise in CPUs [9–11].

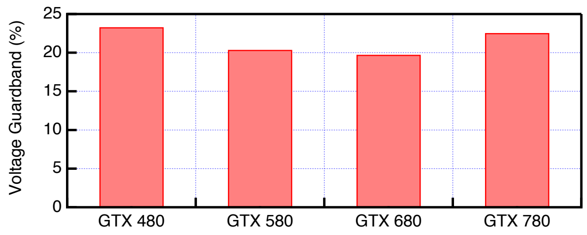


Fig. 1: Measured worst-case voltage guardbands across four generations of NVIDIA GPU architectures indicate the guardband required is large. The details of our measurement setup are described in Sec. 2.3, specifically see "*critical voltage*."

No such prior work exists for GPUs, even though our measurements of the GPU voltage guardband shown in Fig. 1 indicate that they can be as large as the CPUs' voltage guardband. A fundamental reason is the lack of infrastructure support along with critical insights. Thus, the goals of this paper are to demonstrate new insights that uniquely pertain to the GPU and to provide a platform to support new work. Architectural differences between CPUs and GPUs motivate us to conduct such a study. For instance, a GPU has a much larger register file, supports thousands of threads, and has a large number of cores. Such differences alter the voltage noise *root causes* in a GPU architecture versus a CPU architecture.

We provide the first detailed, quantitative modeling and characterization of GPU voltage noise and the leading causes of voltage droops in GPUs. First, we propose *GPUVolt*, a new simulation framework that we developed based on prior work that models the GPU on-die voltage noise behavior accurately. It has 0.9 correlation with hardware measurements. GPUVolt is integrated with GPGPU-Sim [12] and GPUWattch [13], which are robustly validated GPU performance and power simulators, respectively. GPUVolt adds a new dimension that allows researchers to perform a configurable study of the trade-offs between GPU performance, power, and voltage guardband. The infrastructure will be released to the public.

Second, we perform an in-depth analysis of voltage droops for both single-core and chip-wide GPU-specific microarchitectural activities. We demonstrate that global synchronous activity across multiple cores at the second-order droop fre-

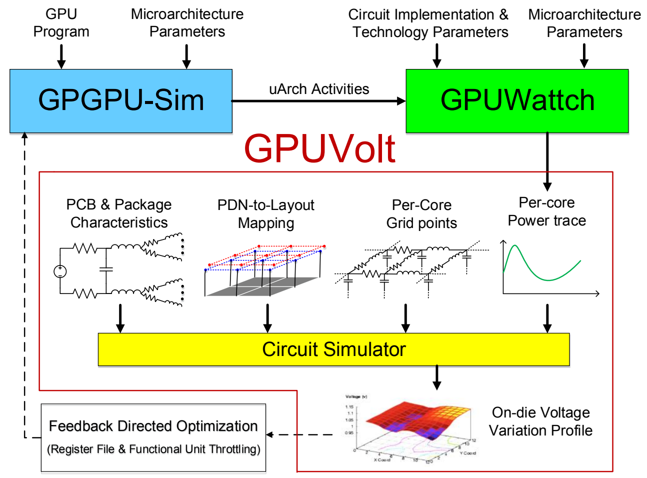

GPU Program
Microarchitecture Parameters
Circuit Implementation & Technology Parameters
Microarchitecture Parameters
GPGPU-Sim
uArch Activities
GPUWattch
GPUVolt
PCB & Package Characteristics
PDN-to-Layout Mapping
Per-Core Grid points
Per-core Power trace
Circuit Simulator
Feedback Directed Optimization
(Register File & Functional Unit Throttling)
On-die Voltage Variation Profile

Fig. 2: An integrated and configurable voltage-noise simulation framework for the GPU many-core architecture.

quency and the core-level register file activity at first-order droop frequency are the root causes of large voltage droops in the GPU architecture. The global synchronous activity is caused by activity occurring in specific microarchitectural units, such as special functions and floating-point units.

Third, we propose a throttling mechanism to reduce the GPU's worst-case voltage guardband. Our mechanism, which throttles the register-file and functional units, reduces the guardband by up to 29%. The key insight, however, is the identification of voltage noise root causes and the ability to throttle them effectively with minimal performance loss.

The paper is organized as follows: Sec. 2 describes the GPUVolt modeling methodology. Sec. 3 focuses on the in-depth characterization of GPU voltage noise root causes, both at the individual core level and chip-wide activity. Sec. 4 demonstrates a use case of GPUVolt, discussing our proposed register-file and functional-unit throttling mechanism. Sec. 5 discusses the related work. We conclude the paper in Sec. 6.

2. GPU Voltage-Noise Modeling

In this section, we describe the voltage noise modeling methodology of GPUVolt. We start by providing an overview of the necessary GPU cosimulation infrastructure, with which GPUVolt is tightly integrated to create a robust and flexible voltage noise simulation infrastructure. Next, we provide the details of the voltage noise simulation framework. Finally, we validate GPUVolt against hardware measurements, showing that it has strong 0.9 correlation across a range of applications.

2.1. Simulation Framework Overview

GPUVolt simulates the voltage noise behavior by calculating the time domain response of the power (voltage) delivery model under current input profiles of each core (Fig. 2). We use GPUWattch [13], a cycle-level GPU power simulator, to approximate the current variation profile of each GPU core under a certain supply voltage level. GPUWattch takes the microarchitectural activity statistics from GPGPU-Sim [12], a cycle-level performance simulator, and calculates the power consumption of each microarchitectural component.

We assume the widely established GTX 480 architecture for our study. We tested and evaluated the accuracy of both GPGPU-Sim and GPUWattch to simulate this architecture. Both tools simulate the architecture with high accuracy. GPGPU-Sim has a strong 97% correlation with the hardware, whereas GPUWattch has a modest 10% modeling error.

We omit a table listing all the simulated architecture details due to space constraints and also, because we do not modify the architecture's default configuration. But, briefly, the GTX 480 consists of many cores that are called streaming multiprocessors (SMs) in NVIDIA terms. The GTX 480 has 15 such SMs. Each SM contains a 64 KB L1 cache/scratchpad, and all SMs share a large 768 KB L2 cache that is backed by six high-bandwidth memory channels. In addition, each SM has a large 131 KB register file and a set of SIMD pipelines to support the execution of a large number of logically independent scalar threads (i.e., 1536 threads).

2.2. Modeling Methodology

GPUVolt's power delivery model consists of three parts (Fig. 4a): the printed circuit board (PCB), the package, and the on-die power delivery network (PDN). We abstract the PCB and package circuit characteristics into a lumped model, while for the on-die PDN we use a distributed model that can capture the on-die voltage fluctuations accurately across the chip. A distributed model can reflect both intra-SM voltage noise as well as inter-SM voltage noise interference [14].

Accurately modeling the GTX 480 PDN characteristics is challenging because there is no public information on its actual PDN design. Therefore, we derive our initial model from the original Pentium 4 model developed by Gupta et al. [14]. However, we scale its PDN parameters in accordance to the GPU's peak thermal design power (TDP) because designers must design the PDN to match the target processor architecture's peak current draw [1,2]. The GTX 480 has a high TDP of over 200 W whereas the Pentium 4 model has a TDP of only 60-70 W [14]. Because high-performance processor package

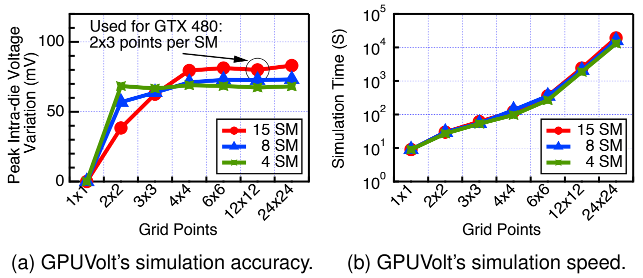
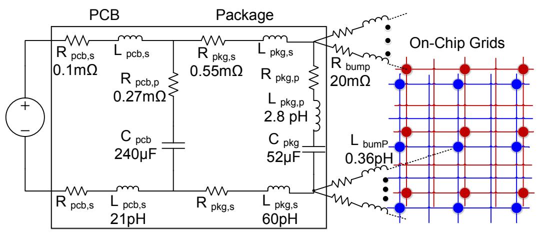
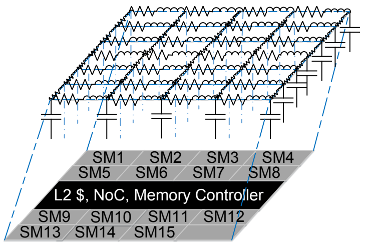


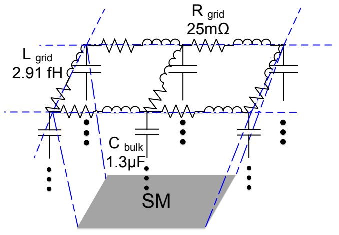
Fig. 3: GPUVolt's simulation accuracy versus simulation speed trade-off (without GPGPU-Sim and GPUWattch overheads).



(a) Overview of the power delivery model.



(b) Mapping the on-chip model to the GPU.



(c) PDN mapping at the SM level.

Fig. 4: Simulated voltage model in GPUVolt. (a) Global view of the power delivery model, including PCB, package, and on-chip PDN. (b) Mapping between the on-chip model and the GPU layout. (c) The on-chip PDN model for each SM.

impedance is no longer scaling linearly [2], we scale GPUVolt's grid parameters by 2× (compared to the 4× TDP ratio between two processors). The parameters and their values are shown in Fig. 4a. Other scaling factors (e.g., 1.5× and 3×) are also possible, which simply result in different PDN characteristics, that are in fact valid configurations (Sec. 2.3).

We lay out the SMs, L2 caches, network on chip (NoC), and memory controllers into the PDN grid based on publicly available die photos of GTX 480 (Fig. 4b); the die photos show an aspect ratio of each SM not being 1, so we use 2×3 grid points to model each SM (Fig. 4c) and 4×6 grid points to model the L2 cache, NoC, and memory controllers.

We do not model the intra-SM floorplan in detail, for two main reasons. First, the goal of GPUVolt is to focus on inter-SM voltage variations and to study the impact of such variations on other SMs in a many-core GPU architecture; the intra-SM variations are relatively small, and therefore adding more detail does not necessarily provide us with additional insights at the chip level. Second, there is no publicly available intra-SM floorplan information for any of the contemporary general-purpose GPU architectures. Having said that, it is entirely feasible to extend GPUVolt with floorplan-specific details. We leave this as future work.

Fig. 3 justifies our grid point allocation scheme (i.e., 2×3 grid points for each SM and 4×6 grid points for the rest). It captures the trade-off between simulation accuracy and speed as the number of total on-chip PDN grid points varies. We inspect the *peak intra-die voltage variation* under maximum SM current variation, which reflects the highest voltage minus the lowest voltage on the die at the same cycle. In effect, it lets us quantify the impact of voltage noise on one core in response to another core's activity, which may be adjacent or located elsewhere on the chip. If we assume a lumped model with only 1 grid point, the intra-die voltage variation in Fig. 3a is nonobservable, which can lead to incorrect conclusions. However, the model begins to capture peak intra-die voltage variation as the grid size increases. With a total of 12×12 grid points, we can achieve a reasonable balance between simulation accuracy and simulation time. The peak intra-die variation starts saturating as the grid size exceeds our choice while the simulation time continues to increase (Fig. 3b).

Fig. 3 also shows how the intra-die variation magnitude varies with the number of GPU SMs. We show this primarily to emphasize the point that the modeling methodology is configurable. GPUVolt can readily support a varying number of SMs, depending on what is assumed of the target architecture.

2.3. Model Validation

We start the validation by showing the impedance-frequency profile of our PDN, which establishes consistency with prior modeling work. Fig. 5 shows the impedance profile, extracted using GPUVolt's modeled PDN. As expected, the impedance profile shows two peak values due to the RLC effects of the PDN. Among the two peak values, the higher peak corresponds to voltage droops that occur at the order of tens-of-cycles, which is commonly referred to as the *first-order droop* (around 100 MHz). The lower peak impedance corresponds to voltage droops that occur at the order of hundreds-of-cycles, known as *second-order droop* (around 1 MHz). Our results are in line with previous studies [14, 15] and validate GPUVolt's PDN modeling methodology. We include other scaling factor results to demonstrate the ability to correctly model cheaper (i.e., high impedance) or costlier (i.e., low impedance) PDNs.

To further validate the PDN, we compare it against measurement results. Ideally, one would measure and compare the hardware's impedance-frequency profile with that of the simulator's. Unfortunately, we do not have access to the required hardware V_{sense} pins [16]. Therefore, we perform a best-effort validation of GPUVolt by comparing the simulated worst-case

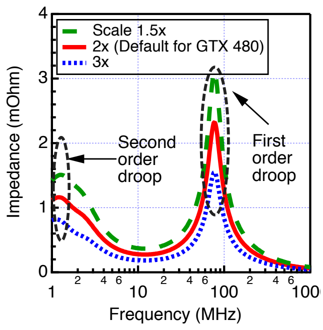


Fig. 5: Our PDN model's impedance-frequency profile.

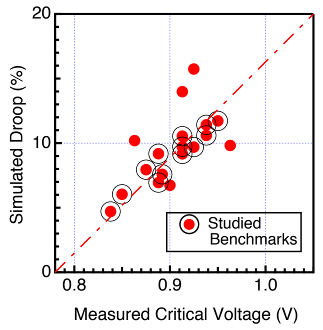


Fig. 6: Simulated droop versus measured critical voltage.

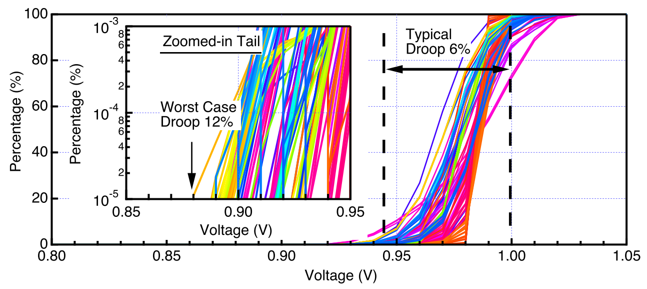


Fig. 7: Cumulative distribution of voltage droops. The typical droop is about 6%. The inset plot zooms into the tail portion.

voltage droops against the *critical voltage* measured on real hardware, using a variety of GPU applications. We measure an application's critical voltage by progressively reducing the GTX 480's supply voltage until the application crashes (i.e., produces a segmentation fault or wrong output compared to the reference run at nominal voltage). We decrement the processor's supply voltage from its default value (1.063 V, 700 MHz) in 10 mV steps, checking the program's correctness after each step. The first voltage at which the application produces an incorrect result is recorded as its critical voltage.

For robust validation, we include applications from a diverse set of benchmark suites, which have a large range of worst-case voltage droops. The application set includes five large programs from CUDA SDK [17], BlackScholes (BLS), convolutionSeparable (CVLS), convolutionTexture (CVLS), dct8x8 (DCT), binomialOptions (BO); seven programs from Rodinia [18]: BACKP, KMN, SSSP, NNC, CFD, MGST, and NDL; and one DMR program from LoneStarGPU [19]. The worst-case droop ranges from 5% to 12%. Because of measurement limitations, we can only validate the whole program's worst-case droop, although kernel-level droops can be analyzed using GPUVolt (Sec. 3).

Fig. 6 shows the correlation between the measured critical voltage and the simulated worst-case voltage droop. GPUVolt faithfully captures the expected critical voltage behavior. The Pearson's correlation is 0.9 assuming the default 2× scaling factor for the GTX 480 architecture and minus the four outliers. As expected, programs with a high measured critical voltage show a large simulated voltage droop, and vice versa.

3. GPU Voltage-Noise Characterization

We use GPUVolt to characterize GPU voltage noise at the program, SM-component, and global inter-SM interference level. Our analysis reveals that large voltage droops occur rarely in the GPU, and as such the GPU voltage guardband is overprovisioned. Although this insight has been observed in CPUs, we are the first to report such analysis on GPUs.

We show that key microarchitecture components, such as the large register file and functional units, are the main contributors of voltage droops in the GPU architecture. Furthermore, we show that activity at the intra-SM level when in sync with other SMs' activity can lead to global synchronous microarchitectural activity that can cause large chip-wide droops.

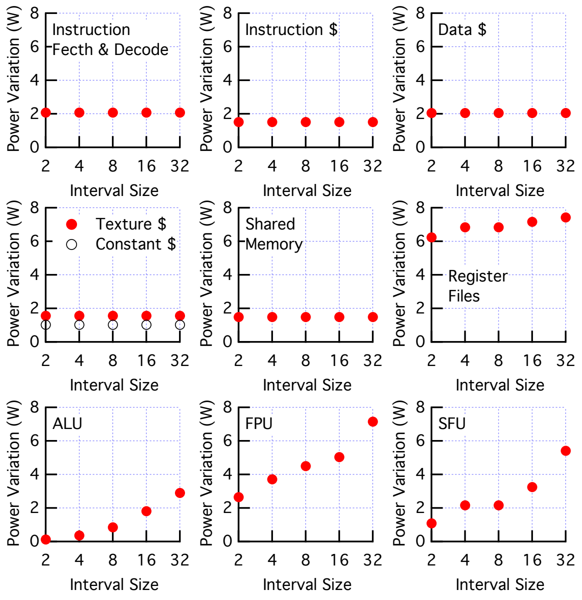


Fig. 8: Power variation for all the major GPU components over several different interval sizes, ranging from 2 to 32 cycles.

3.1. Program Voltage Droop Distribution

To understand the typical voltage noise profile on GPUs, we gathered the voltage traces of all the programs mentioned previously in Sec. 2.3. Fig. 7 shows a cumulative distribution profile of the voltage droops for the different GPU programs. Each GPU program consists of one or more kernels, where a kernel is defined as a single unit of execution. Each line in Fig. 7 corresponds to a distinct program kernel. We analyze the data from over 200 kernels executed across all the programs.

We observe that the vast majority of the voltage samples (over 99.9% of the time) are greater than 0.94 V. We refer to these droops as the typical voltage droops, which are half the magnitude of the worst-case droop (i.e., 0.88 V) indicated by the zoomed-in tail portion. The large voltage droops rarely occur, with a cumulative frequency that is less than 0.02%.

It is also important to note that both typical- and worst-case voltage droop behaviors are very much program- or kernel-dependent. On one hand, the lines in Fig. 7 are not overlapping, which indicates that the typical droop behavior varies across the programs and their kernels. On the other hand, as the inset plot shows, the worst-case droop of some kernels is as small as 5% (i.e, 0.95 V), whereas the worst-case droop of other kernels is as large as 12% (i.e, 0.88 V).

The differences between typical- and worst-case droop motivate us to understand the GPU's voltage-noise root causes in detail. We focus mainly on characterizing and, to a lesser extent, mitigating the worst-case voltage droop at the architecture level since the first step is to uncover the microarchitectural components that are responsible for the large voltage droops.

3.2. Component Current Variation

The first step to identify the voltage noise root causes is to characterize each component's contribution to the total $L\frac{di}{dt}$ effect. We approximate each microarchitectural component's per-cycle current draw using the per-cycle power consumption results from GPUWattch [13]. A large power variation in a short time period would lead to a large voltage droop.

We quantify the power variation "speed" of each microarchitecture component by recording its peak power variation within a timing window. Using various window lengths of size N, we capture the peak current draw characteristics of the different components accurately. We sweep N over 2, 4, 8, 16, and 32 cycles, enough to cover the first-order droop impedance (Fig. 5). We find that power variation plateaus for all components with a time scope larger than 32 cycles; therefore, we do not increase N beyond 32. The microarchitecture components include front-end (i.e., fetch & decode); various on-chip caches (i.e., texture, constant, and data); shared memory; register file; and integer, floating-point, and special-function units (ALU/FPU/SFU). The list is comprehensive and includes all the major components.

Fig. 8 shows the characterization results. We make three important observations. First, power variation of the front-end and various caches is stable and low across different window sizes. For example, power variation of the instruction cache is constantly 2 watts across different cycles. We expect this because instruction cache access is a single-cycle operation. Other caches (data/constant/texture) and shared memory have similar power variation with slightly different magnitudes.

Second, the register-file has the most rapid power variation among all components. Its behavior is closely tied to the unique characteristics of the GPU architecture. Modern GPUs require a large register file to hold the architectural states of thousands of threads in each SM core. In our simulated GTX 480 architecture, the register file size is 131 KB, which is much larger than the 16 KB to 48 KB L1 cache sizes. Consequently, the register file access rate and power consumption are much higher compared to the RF in CPUs [13, 20].

Third, the integer unit (ALU), floating-point unit (FPU), and special function unit (SFU) also have large power variation. As compared to the register file, these components exhibit large variation at the window size of 32 cycles, which is due to the units' multicycle execution latencies.

3.3. Intra-SM Voltage Droop Analysis

We must quantify each components's contribution to an SM's voltage-noise profile over its execution duration because even though specific components may experience high power variation, it does not automatically imply that they are the leading contributors of large voltage droops in the GPU. Their impact may vary depending on their utilization frequency.

We leverage the linear property of our voltage model to quantify each component's contribution to a single SM's voltage noise. The linear property of GPUVolt's RLC circuit model implies that the temporal response of the PDN's on-chip voltage noise is the sum of the individual parts over time. Therefore, we can establish each component's contribution to the SM's total voltage noise by feeding the individual component's current profile separately into GPUVolt.

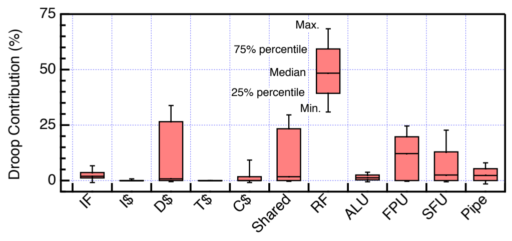


Fig. 9: Component contribution to any voltage droop greater than 3% (i.e. greater than typical droop) at the single-SM level.

Fig. 9 shows the contribution of the major components to voltage droops in a single SM. We perform a cycle-level comparison of each component's contribution to the magnitude of voltage droops that are larger than 3% of the nominal supply voltage. We pick 3% as the threshold because the maximum droop at the intra-SM level is about 5%. Therefore, a 3% threshold filters out the typical intra-SM droop behavior, letting us isolate and focus on the large intra-SM droops.

Fig. 9, shown as a box plot, captures the maximum, 75%, and 25% quartiles, and the minimum contribution of each component for the cycle-by-cycle voltage samples gathered during a run. Even at the intra-SM level, the register file remains the single most dominant source of voltage droops, with a maximum of 70% and median of 50% contribution to the droops. Other components, such as FPU, SFU, shared memory, and data cache, also contribute to large droops, but their influence is smaller as compared to the register file.

3.4. Chip-Wide Voltage Droop Analysis

We expand our analysis to chip-wide voltage droops to understand how intra-SM component activity combined with activity from all SMs can lead to large voltage droops with magnitudes larger than 8%. We find that aligned activity and second-order droop effects are the dominant root causes.

Chip-wide droops are caused by aligned component activity across different SMs because GPUVolt assumes a *shared* PDN (i.e., all SMs are connected to the same power grid); prior work demonstrates that a shared PDN is more robust to voltage noise than a split power grid where cores are connected to separate power grids [4]. An unfortunate side effect of a shared PDN is that one SM's aggregate component activity can impact another SM's voltage; such behavior has been studied in CPUs [6, 9], but the root causes are unknown in GPUs.

Unlike in the intra-SM scenario, where rapid power variation occurs at the first-order droop frequency, the aligned chip-wide power variation occurs at the second-order droop

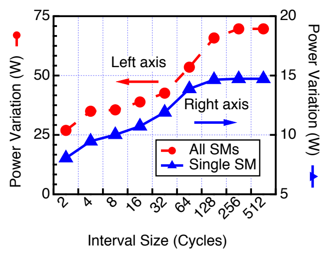


Fig. 10: SM power variation at different interval sizes.

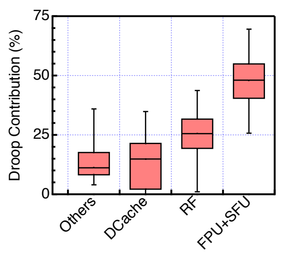


Fig. 11: Component impact on chip-wide voltage droops.

frequency. Fig. 10 shows the total peak power variation for a single SM and all the 15 SMs. We study interval ranges between 2 cycles and 512 cycles. The wide interval captures both the first- and second-order droop frequencies. The single SM's power variation begins to saturate at the 64-cycle interval with a peak of 14 watts, which corresponds to a single SM's maximum power consumption at any given point. In contrast, the total SM power variation for all SMs reaches a peak between the 256- or 512-cycle interval, which matches with the second-order droop frequency. The peak value is about 70 watts, which indicates that there are at least six SMs whose activities are in strong alignment to cause large droops.

To understand global component activity impact on chip-wide voltage droops, we carry out a characterization study as in Sec. 3.3. We feed GPUVolt with components' currents from all SMs to expose each component's droop contribution to all droops that are larger than 8% of the nominal supply voltage. Fig. 11 presents our results, and it shows that the global aligned activities are from the execution units across SMs. The execution units (mainly FPU and SFU) contribute most to the chip-wide droops (maximum 75% and median 50%). Compared to the single or intra-SM case, the register file only accounts for 25% to 45% of the total chip-wide droops.

Our insights emphasize that it is important to understand both chip-wide and intra-SM activity in a combined fashion to comprehensively identify voltage noise root causes in GPUs.

4. GPU Voltage Noise Mitigation: A Case Study

We conduct a proof-of-concept study to demonstrate that it is possible to mitigate the GPU's worst-case guardband on the basis of our intra-SM and chip-wide inter-SM voltage droop characterization. Our goal is *not* to comprehensively evaluate a wide variety of mechanisms and demonstrate which is best; rather, it is to demonstrate that our root-cause analysis is sound and that throttling the key components (i.e., execution units and the register file) will reduce the worst-case voltage droop.

We evaluate a throttling solution that is similar to "Pipeline Damping" [8], which limits the key components' activity increase over an interval of consecutive cycles. In our work, we set the interval size such that it matches the components' droop-impact characteristics. For example, the power variation of the register file (RF) causes large voltage droop at

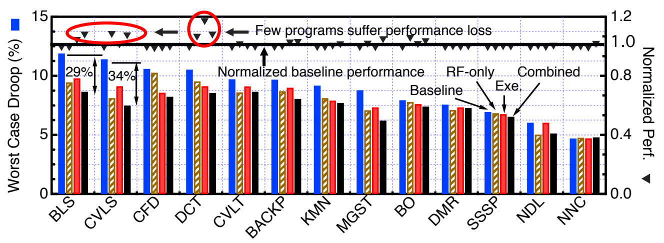


Fig. 12: Worst-case voltage droop reduction caused by throttling components identified to cause the most voltage droop.

the first-order droop frequency. Similarly, the execution units (Exe.) cause large voltage droops at the second-order droop frequency. Consequently, we set 8 and 800 cycles as the throttling interval size for the RF and Exe., respectively.

Fig. 12 shows the throttling results in terms of the worst-case droop with and without our throttling evaluation. The key insight is that we have to perform a combination of RF and execution unit throttling because the root cause of a large voltage droop can be due to either component. Combined throttling can effectively mitigate the worst-case droop. In BLS, the droop reduces from 12% to 8.5%, which is a 29% improvement. However, RF-only throttling barely reduces the droop to 10.25% in CFD from its maximum droop of 10.6%. The geometric-average performance overhead of throttling both components is 4.1% for all of the evaluated programs.

5. Related Work

Gupta et al. were the first to use a distributed PDN model to model on-die voltage noise [14]. GPUVolt is a natural but GPU-specific extension of the prior work. GPUVolt is configurable and useful to study GPU voltage-noise characteristics with different SMs (e.g., Fig. 3a), package characteristics (e.g., Fig. 5), microarchitecture configurations (e.g., Fig. 2), etc.

At the single-core level, prior work concluded that rapid current increases and resonant current behavior caused by microarchitectural activities–e.g., pipeline flushing and cache misses–are the root causes of voltage droops [1, 2, 7, 8, 10, 11]. In contrast, our GPU component-level characterization shows that the GPU's throughput-architecture design causes new sources of problems, such as its large register file.

Multicore CPU voltage noise studies focused on thread interference and how to mitigate the effect at the global level by scheduling threads [6, 9]. We took a different approach by studying the contribution of various components and their combined effect on voltage noise across the different SMs. We find that synchronized global activity of the SMs' execution units and register files can lead to large chip-wide voltage droops that we can mitigate by throttling these units.

6. Conclusion

GPUVolt is an integrated voltage-noise simulation framework that is specifically targeted at GPU architectures. We validated it against hardware measurements, and it shows a 0.9 correlation for a range of programs. Using GPUVolt, we demonstrate that the register file and aligned execution unit (i.e., ALU/FPU/SFU) activity at the second-order droop frequency are the main sources of voltage noise. Controlling their utilization can reduce the worst-case voltage droop magnitude by as much as a 29% with a marginal impact on performance.

Acknowledgement

This work is sponsored, in part, by Defense Advanced Research Projects Agency, Microsystems Technology Office (MTO), under contract no. HR0011-13-C-0022. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This document is: Approved for Public Release, Distribution Unlimited.

References

[1] E. Grochowski, D. Ayers, and V. Tiwari, “Microarchitectural simulation and control of di/dt-induced power supply voltage variation,” in *Proc. of HPCA*, 2002.

[2] R. Joseph *et al.*, “Control techniques to eliminate voltage emergencies in high performance processors,” in *Proc. of HPCA*, 2003.

[3] C. R. Lefurgy *et al.*, “Active management of timing guardband to save energy in power7,” in *Proc. of MICRO*, 2011.

[4] N. James *et al.*, “Comparison of Split-Versus Connected-Core Supplies in the POWER6 Microprocessor,” in *Proc. of ISSCC*, 2007.

[5] D. Ernst *et al.*, “Razor: a low-power pipeline based on circuit-level timing speculation,” in *Proc. of MICRO*, 2003.

[6] V. Reddi *et al.*, “Voltage Smoothing: Characterizing and Mitigating Voltage Noise in Production Processors via Software-Guided Thread Scheduling,” in *Proc. of MICRO*, 2010.

[7] M. D. Powell and T. N. Vijaykumar, “Pipeline Muffling and a Priori Current Ramping: Architectural Techniques to Reduce High-frequency Inductive Noise,” in *Proc. of ISLPED*, 2003.

[8] M. D. Powell and T. Vijaykumar, “Pipeline damping: a microarchitectural technique to reduce inductive noise in supply voltage,” in *Proc. of ISCA*, 2003.

[9] T. N. Miller *et al.*, “VRSync: Characterizing and Eliminating Synchronization-induced Voltage Emergencies in Many-core Processors,” in *Proc. of ISCA*, 2012.

[10] M. S. Gupta *et al.*, “An event-guided approach to handling inductive noise in processors,” in *Proc. of DATE*, 2009.

[11] V. Reddi *et al.*, “Voltage emergency prediction: Using signatures to reduce operating margins,” in *Proc. of HPCA*, 2009.

[12] A. Bakhoda *et al.*, “Analyzing CUDA Workloads Using a Detailed GPU Simulator,” in *Proc. of ISPASS*, 2009.

[13] J. Leng *et al.*, “GPUWattch: Enabling Energy Optimizations in GPGPUs,” in *Proc. of ISCA*, 2013.

[14] M. S. Gupta *et al.*, “Understanding Voltage Variations in Chip Multiprocessors Using a Distributed Power-delivery Network,” in *Proc. of DATE*, 2007.

[15] K. Aygun *et al.*, “Power Delivery for High-Performance Microprocessors,” in *Intel Technology Journal, Nov. 2005.*

[16] M. Saint-Laurent and M. Swaminathan, “Impact of power-supply noise on timing in high-frequency microprocessors,” *IEEE Transactions on Advanced Packaging*, 2004.

[17] NVIDIA Corporation, “CUDA C/C++ SDK CODE Samples,” 2011.

[18] S. Che *et al.*, “Rodinia: A benchmark suite for heterogeneous computing,” in *Proc. of IISWC*, 2009.

[19] M. Burtscher, R. Nasre, and K. Pingali, “A Quantitative Study of Irregular Programs on GPUs,” in *Proc. of IISWC*, 2012.

[20] M. Gebhart *et al.*, “Energy-efficient mechanisms for managing thread context in throughput processors,” in *Proc. of ISCA*, 2011.