# Core Tunneling: Variation-Aware Voltage Noise Mitigation in GPUs*

Renji Thomas Kristin Barber Naser Sedaghati Li Zhou
Radu Teodorescu

Department of Computer Science and Engineering
The Ohio State University
{thomasr, barberk, sedaghat, zholi, teodores}@cse.ohio-state.edu

## ABSTRACT

Voltage noise and manufacturing process variation represent significant reliability challenges for modern microprocessors. Voltage noise is caused by rapid changes in processor activity that can lead to timing violations and errors. Process variation is caused by manufacturing challenges in low-nanometer technologies and can lead to significant heterogeneity in performance and reliability across the chip. To ensure correct execution under worst-case conditions, chip designers generally add operating margins that are often unnecessarily conservative for most use cases, which results in wasted energy.

This paper investigates the combined effects of process variation and voltage noise on modern GPU architectures. A distributed power delivery and process variation model at functional unit granularity was developed and used to simulate supply voltage behavior in a multicore GPU system. We observed that, just like in CPUs, large changes in power demand can lead to significant voltage droops. We also note that process variation makes some cores much more vulnerable to noise than others in the same GPU. Therefore, protecting the chip against large voltage droops by using fixed and uniform voltage guardbands is costly and inefficient.

This paper presents *core tunneling*, a variation-aware solution for dynamically reducing voltage margins. The system relies on hardware critical path monitors to detect voltage noise conditions and quickly reacts by clock-gating vulnerable cores to prevent timing violations. This allows a substantial reduction in voltage margins. Since clock gating is enabled infrequently and only on the most vulnerable cores, the performance impact of core tunneling is very low. On average, core tunneling reduces energy consumption by 15%.

## 1. INTRODUCTION

Graphics processing units (GPUs) are increasingly used as high-performance and energy-efficient accelerators for general purpose computing. However, hard constraints on power consumption are now limiting their performance growth. Lowering the chip supply voltage ($V_{\mathrm{dd}}$) is one of the most effective techniques for improving energy efficiency. Unfortunately, multiple technological challenges are making supply voltage reduction very difficult going forward.

One of these challenges is voltage noise caused by abrupt fluctuations in current demand ($\delta I/\delta t$). These fluctuations generally occur because of rapid changes in workload intensity. If the voltage deviates too much from its nominal value, it can lead to so-called "voltage emergencies," which can cause timing errors. To prevent these emergencies, chip designers add voltage margins that in modern processors can be as high as 20% [1, 2, 3, 4] wasting substantial amounts of energy.

Another important challenge is process variation caused by extreme difficulties in manufacturing chips with very small feature sizes. Variation affects crucial transistor parameters such as threshold voltage ($V_{\mathrm{th}}$) and effective gate length ($L_{\mathrm{eff}}$) leading to heterogeneity in transistor delay and power consumption. Process variation is also generally mitigated by adding conservative guardbands to accomodate the slowest regions of the chip, resulting in additional energy costs.

Previous work has addressed voltage noise in CPUs [2, 5, 6, 7, 8, 9, 10, 11, 12] and more recently GPUs [3, 4, 13] with a variety of hardware and software-based solutions that help reduce guardbands and save energy. Process variation has also received significant attention mainly targeting CPUs [14, 15, 16, 17, 18, 19, 20] and to a lesser extent GPUs [21]. The combined effects of process, voltage and temperature variation on single-core CPUs were studied in [22].

Virtually all prior work on GPUs has addressed either voltage noise or process variation independently. This paper investigates their combined effects on modern GPU architectures and presents concerted mitigation solutions. One of the barriers to conducting research in this space has been the difficulty of jointly modeling process variation and voltage noise. To make this work possible, we developed a distributed power delivery model integrated with a process variation model for GPUs. The model captures the effects of processor activity on supply voltage as well as the combined effects

*This work was supported in part by the Defense Advanced Research Projects Agency under the PERFECT (DARPA-BAA-12-24) program and the National Science Foundation under grant CCF-1253933.

of $V_{dd}$ and $V_{th}$ variations on circuit delay distribution at fine granularity within the chip.

We observed that GPUs are especially vulnerable to voltage noise. Their highly parallel design leads to workload execution patterns that are often coordinated and highly variable, leading to large $\delta I/\delta t$. For instance lower-power memory accesses are often followed by high-power floating point instructions executed by all compute lanes in a GPU core (SM). In addition, activity is often synchronized across SMs, particularly after new kernels are launched. These large $\delta I/\delta t$ events cause large $V_{dd}$ droops that can lead to voltage emergencies.

Moreover, we observe that process variation can lead to important core-to-core differences in sensitivity to voltage noise. Since virtually all GPUs run all their SMs at the same frequency, this results in some SMs having lower effective margins than others, even though they are design-identical. This means that any solution which targets voltage noise and is applied indiscriminately to all SMs will very likely waste energy.

In this paper we present a variation-aware, voltage noise mitigation solution that addresses the challenge of voltage noise in GPUs in an energy-efficient way. Our solution relies on delay monitoring circuits (also called critical path monitors) to detect when the voltage droops to dangerous levels. When that happens, we deploy a mechanism we call "core tunneling" that essentially gates the clock of the entire SM to prevent errors. Since no computation occurs while clock-gated, the core can safely "tunnel" through a region of low voltage that would be otherwise unsafe. This mechanism allows the GPU to operate reliably with substantially lower voltage margins. A power management governor dynamically adjusts the supply voltage of the chip based on feedback from the critical path monitors. The goal is to balance the reduction in supply voltage with the performance overhead of core tunneling.

To reduce the number of tunneling events we implement activity smoothing solutions aimed at reducing $\delta I/\delta t$. We observed that some of the worst voltage droops occur when new application kernels are launched. Since launching a new kernel, which activates many thread blocks, generally follows a low-power execution phase, the resulting $\delta I/\delta t$ can be very large. To address this problem we developed a noise-aware thread block scheduler that staggers the launching of new threads across the GPU to smooth out $\delta I/\delta t$ and reduce the worst voltage droops.

Our solutions reduce voltage margins with little performance impact, lowering energy consumption by an average of 15%. In addition, we show core tunneling is robust to the worst case voltage noise induced by a "voltage virus" that excites the chip's resonance frequency.

In summary, this paper makes the following contributions:

- First work to model the combined effects of voltage noise and process variation in GPUs.
- Core tuneling, a simple, process variation-aware mechanism that reduces voltage margins with low performance impact.
- A noise-aware thread block scheduler that further mitigates the largest observed voltage droops.
- Demonstrates core tunneling robustness against worst-case voltage virus.

The rest of this paper is organized as follows: Section 2 characterizes voltage noise and process variation in GPU systems. Section 3 presents core tunneling, our variation-aware noise mitigation solution. Section 4 details the variation and voltage noise modeling framework. Sections 5 and 6 present the methodology and experimental evaluation. Section 7 discusses related work and section 8 concludes.

## 2. VOLTAGE NOISE AND VARIATION

A modeling infrastructure was developed to capture the joint effects of voltage noise and process variation on GPUs. It integrates the GPGPU-Sim [23] simulator and the GPUWattch [24] power model with a detailed power-delivery model that captures $V_{dd}$ response to $\delta I/\delta t$ as a function of workload. The framework is also integrated with the `VARIUS` process variation model [25]. This modeling infrastructure is described in detail in Sections 4 and 5.

### 2.1 Voltage Noise

Runtime variation in workload intensity is a key factor in voltage noise. Large and rapid changes in current demand ($\delta I/\delta t$) from the various functional blocks across the chip can lead to large droops in the supply voltage. These effects are illustrated in Figure 1 using a section of the breadth-first search (BFS) benchmark running on a 16-SM GPU. The top graph shows total chip power consumption and the bottom graph shows the lowest $V_{dd}$ across the GPU. The figure shows the execution of two kernels that correspond to two loop iterations in the algorithm. We observe that inside each kernel power consumption is relatively constant, which corresponds to low $V_{dd}$ noise. However, when the kernels are first scheduled for execution we see large power spikes as multiple SMs become active simultaneously. This leads to large voltage droops around 50, 400 and 800$\mu$s. Lower voltage is associated with an increase in circuit delay that can lead to timing failures.

Figure 2 shows the same data as a histogram of the lowest voltage reached across the GPU in each cycle. We can see a significant spread in voltages ranging from 1.1V to 1V, indicating that the worst droops bring the $V_{dd}$ at least 15% (150mV) below the nominal voltage of 1.15V. We also note that the $V_{dd}$ distribution exhibits a long and shallow tail towards lower voltages, indicating that large droops occur rarely. However, static design margins have to be large enough to tolerate the worst droops the chip can experience, even though these are generally rare events.

### 2.2 Process Variation

Process variation affects the speed of transistors across the GPU die, making some regions slower than others. In order to run the chip safely at its target frequency, a higher supply voltage is needed to boost the slower

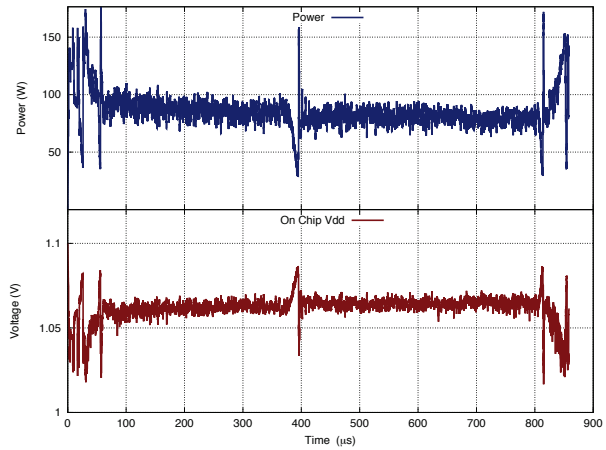


Figure 1: Power consumption (top) and minimum die $V_{dd}$ (bottom) traces for the BFS algorithm.

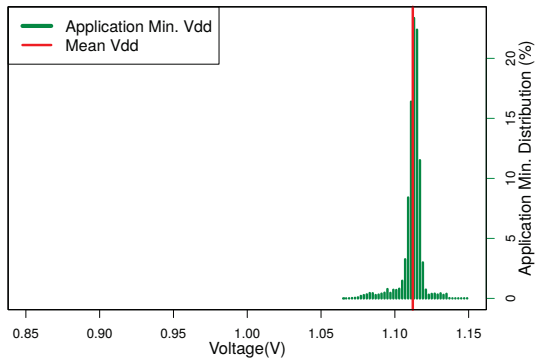


Figure 2: Minimum voltage histogram for BFS.

sections. Figure 3 shows an outline of a GPU floorplan showing the minimum voltages required by different regions of the chip. This distribution models a typical $V_{th}$ variation of 5% $\sigma/\mu$ (standard deviation/chip average). The regions shown in blue are more resilient and can safely run at around 0.8V. The regions shown in red are the most vulnerable, requiring a $V_{dd}$ of 0.95-1V. SMs that happen to be located in slower regions of the die are more vulnerable and therefore require either a higher supply voltage or lower chip frequency. This profile is consistent with prior work that has characterized process variation effects on GPUs [21].

## 2.3 Combined Effects of Noise and Variation

Design margins that account for both process variation and voltage noise are very inefficient because they have to consider the worst-case droop *and* the most vulnerable SMs. Most of the time the GPU's voltage will be well above the lowest allowed. Moreover, even when large voltage droops occur, only a few SMs will be affected.

Figure 4 illustrates the combined effects of voltage noise and process variation on benchmark BFS. On the same plot we show both the histogram of minimum

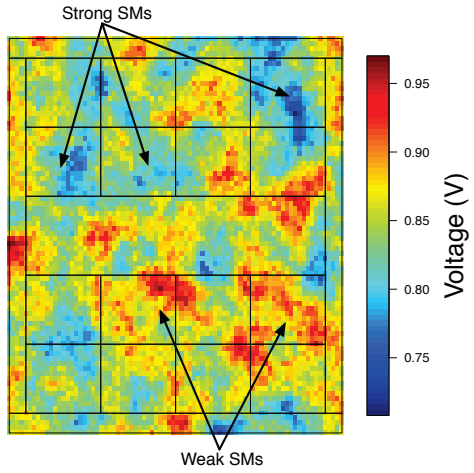


Figure 3: GPU floorplan overlaid on variation map showing the lowest safe $V_{dd}$ at which regions of the die can operate safely.

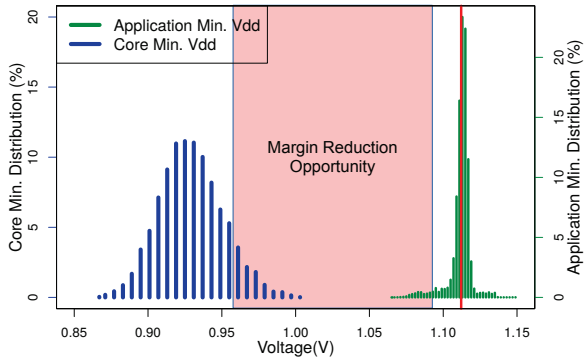


Figure 4: Histograms of minimum SM safe voltage (left) and the lowest on-die $V_{dd}$ reached while running BFS (right).

voltages while running BFS (right-hand side of the figure) and the histogram representing the minimum safe $V_{dd}$ allowed by the SMs (left-hand side). The SM-level distribution is the result of Monte Carlo simulations of hundreds of dies and shows the percentage of SMs that need a certain $V_{dd}$ level to function correctly. All threshold voltage distributions we generate have the same average $V_{th}$ in order to capture only within-die variation and exclude die-to-die effects.

We can see that the most resilient SMs can safely tolerate $V_{dd}$s as low as 0.87V, while the most vulnerable ones require at least 1V. The area highlighted in red represents the opportunity for margin reduction in this benchmark. Lowering the margin essentially implies shifting the benchmark's runtime $V_{dd}$ distribution towards the SM-level distribution. Margin reduction might cause the two distributions to overlap, which means that the voltage will *occasionally* drop below the minimum

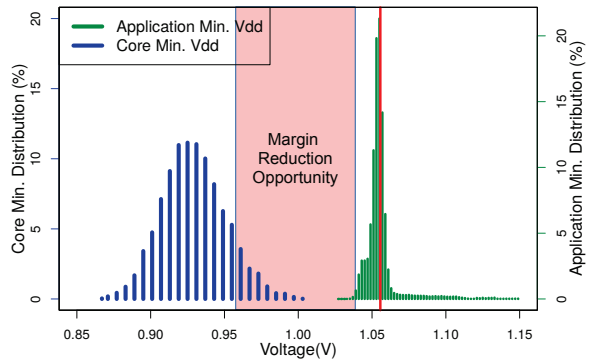


Figure 5: Histograms of minimum SM safe voltage (left) and the lowest on-die $V_{dd}$ reached while running CP (right).

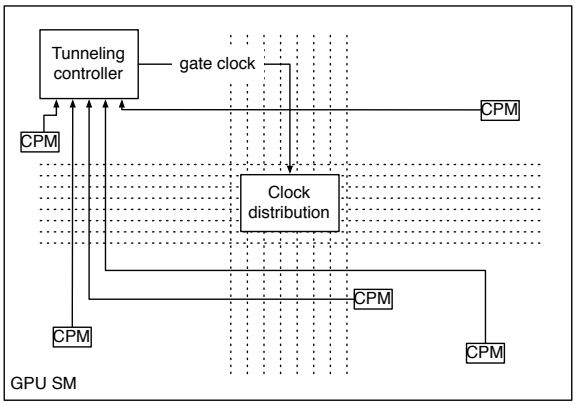


Figure 6: Core tunneling hardware in a GPU SM.

safe level for *some* SMs. To prevent errors, noise mitigation mechanisms will have to be in place, but they will only need to be deployed on the most vulnerable SMs. These generally account for about 10% of the total number of SMs in a chip.

Figure 5 shows the same type of minimum voltage histogram for the Coulombic Potential (CP) benchmark. We notice a similar spread in voltages as in the case of BFS. However, unlike BFS, CP exhibits almost no tail at low $V_{dd}$, with the lowest voltages reached frequently throughout the benchmark. We expect margin reduction to be more difficult for CP since it's effective voltage is already low and lowering it further would push a large number of cycles below the safe voltage.

# 3. CORE TUNNELING

We propose replacing the conservative static voltage guardband with a lower dynamic margin, that adjusts to the chip's operating conditions. To ensure reliable operation under reduced margins, we developed a variation-aware voltage noise mitigation solution. The system includes noise detection capabilities, clock gating of select cores to avoid errors when noise is detected, and a dynamic guardband management system for energy optimization.

## 3.1 Voltage Noise Detection

The system relies on delay monitoring circuits called Critical Path Monitors (CPMs) [26, 27] to detect when voltage droops to dangerous levels. CPMs are simple devices that measure signal propagation delay through a chain of logic gates chosen to be representative of the various logic paths inside a processor. When voltage droops, propagation delay through the affected sections of the chip increases. This change in delay is detected by the CPM and can be used as a trigger for noise mitigation solutions. CPMs have been investigated extensively in the research community and have been deployed commercially in IBM POWER7 processors [26]. They can detect voltage changes that are as small as 10mV within a single clock cycle at multi-gigahertz frequencies [27].

Process variation makes the critical $V_{dd}$ for each SM different. However, since CPMs measure circuit delay and not $V_{dd}$, there is no need to calibrate each CPM individually to the characteristics of the SM. The CPMs can be jointly calibrated to measure deviations from the processor cycle time. Propagation delay is a function of both $V_{dd}$ and $V_{th}$. The CPMs will therefore capture the joint effects of both process variation and voltage noise. As an additional benefit, delay changes due to temperature variation or aging should also be captured by the CPMs.

## 3.2 Tunneling to Avoid Errors

When excessive noise is detected, the system deploys a protection mechanism we call "core tunneling" that gates the clock of the entire core to prevent timing errors. Since all activity stops while the SM is clock-gated, the core can safely "tunnel" through a region of low voltage that would be otherwise unsafe.

Figure 6 shows a diagram of the tunneling hardware in a GPU SM. Multiple CPMs are placed close to functional units that consume the most power and will experience the worst voltage droops. An SM-level tunneling controller unit collects delay information from all CPMs and uses the one showing the highest delay to decide when clock gating should be activated.

Two thresholds are established to control the activation (*tunnel entry*) and deactivation (*tunnel exit*) of clock gating. When the core tunneling controller determines that the CPM delay exceeds the *tunnel entry* threshold the SM is clock gated starting from the following cycle. The *tunnel entry* threshold is a function of the processor's frequency and is chosen such that tunneling is activated before the lowest safe $V_{dd}$ can be breached under worst case $\delta I/\delta t$. The SM will remain clock gated as long as the measured delay is higher than the *tunnel exit* threshold. When the delay from all CPMs drops below that level, the controller disables clock-gating and the SM resumes regular activity.

## 3.3 Runtime Voltage Management

In order to reduce voltage margins, a GPU-level power governor implements a form of voltage speculation [28] that dynamically adjusts $V_{dd}$ at constant frequency to

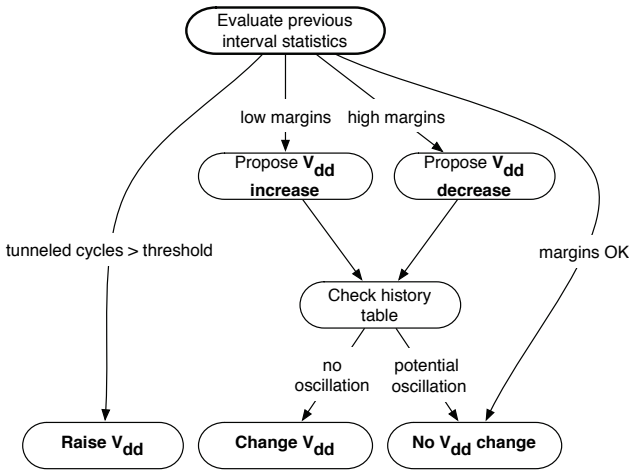


Figure 7: The power governor's control flow diagram for $V_{dd}$ management.

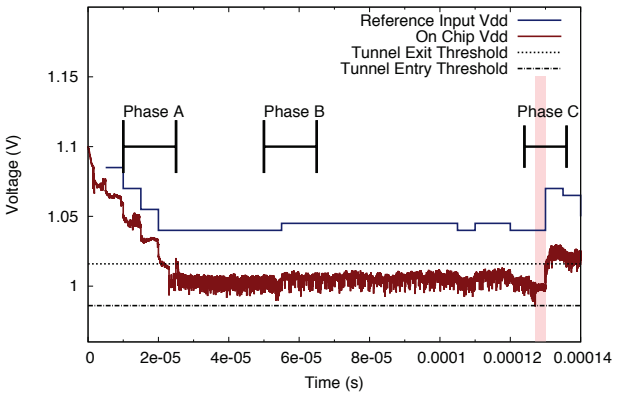


Figure 8: Dynamic $V_{dd}$ adaptation and core tunneling for a section of the WP benchmark.

adapt to operating conditions. Since $V_{dd}$ is adjusted by small increments and the clock frequency remains the same, there is no performance penalty associated with these changes. The governor is invoked at regular intervals (e.g. every 5$\mu$s) to re-evaluate the GPU's supply voltage based on lightweight feedback from the SMs. The governor algorithm runs on an on-chip microcontroller similar to those used to manage power in the Intel Itanium [29] or Core i7 [30] processors.

Figure 7 shows the control flow diagram that defines the governor's operation. At the end of each evaluation interval the governor collects some lightweight information from the tunneling controllers in each SM. The governor uses this information to decide whether to raise, lower or keep the supply voltage constant. The goal is to optimize energy consumption by balancing low $V_{dd}$ margins (which saves power) with the frequency and duration of core tunneling events (which reduces performance penalty).

The governor examines the number of cycles each SM has been tunneled over the previous interval. If any SM has been tunneled for longer than a predetermined threshold (e.g. >50% of the time) the governor raises the supply voltage immediately by a $\Delta V_{dd}$. This is done to reduce the number and duration of future tunneling events and ensure the performance impact of core tunneling is minimal.

Each tunneling controller also monitors the amount of available timing margin relative to the GPU's cycle time. This information is collected and evaluated by the governor. If the timing margin of the slowest SM is below a threshold, the governor proposes an increase in $V_{dd}$. If, on the other hand, the margin is sufficiently large, the governor proposes a decrease in $V_{dd}$ by a variable increment that is proportional to the available margins. If margins are within the target delay range, the $V_{dd}$ will remain unchanged over the next interval.

Certain workload patterns could cause the governor to oscillate between two neighboring voltage levels. To avoid unnecessary voltage changes, the governor uses a history table to record previous $V_{dd}$ states. If a change in $V_{dd}$ is the result of a low or high delay margin, the governor checks the history table to ensure that proposed $V_{dd}$ level has not been recorded in the recent past (5-10 evaluation intervals). If the proposed $V_{dd}$ is found, the potential for oscillation exists and the $V_{dd}$ is not changed; otherwise the proposed $V_{dd}$ is put in place. Voltage changes triggered by the tunneling threshold indicate delay margins have been exceeded. As a result, the voltage is raised immediately regardless of history, as Figure 7 shows.

## 3.4 Core Tunneling Example

The process of dynamically adjusting the $V_{dd}$ to workload characteristics is illustrated with a sample trace from the Weather Prediction (WP) benchmark shown in Figure 8. The figure shows the reference supply voltage as it enters the GPU package at the level dictated by the power governor, as well as the lowest on-chip $V_{dd}$ for one SM. The difference between the reference input $V_{dd}$ and the on-die $V_{dd}$ is caused by the combined effects of IR drop and voltage noise. The figure also shows the voltage equivalents of the tunnel entry and tunnel exit thresholds.

This example reveals three distinct phases that help demonstrate the dynamic $V_{dd}$ adaptation process:

In `Phase A` voltage noise is subdued because power consumption is low as the benchmark is ramping up. The SMs report that high timing margins are available. As a result, the governor drops the $V_{dd}$ in aggressive 30mV increments. At the end of phase A, the tunneling controllers are reporting that available margins are on taget. The governor stops lowering the $V_{dd}$ and holds it constant.

`Phase B` demonstrates how this system compensates for an overall increase in power consumption. As the workload intensity picks up and the power demand increases, the $V_{dd}$ droops as a result. This leads to the SMs reporting reduced magins. To avoid excessive tun-

neling, the governor raises the $V_{dd}$ by 5mV. This state is then held for the remainder of phase B.

`Phase C` demonstrates how the system handles a significant $\delta I/\delta t$ event. At the beginning of phase C, a rapid increase in power demand causes the supply voltage to drop rapidly before the governor has a chance to intervene. The core tunneling controller of the most vulnerable SMs detect that their safety margins are about to be exceeded. When the tunnel entry threshold is breached the SMs are immediately tunneled to avoid errors (light-red shaded area in Figure 8). Since activity is high and the $V_{dd}$ remains below the tunnel exit threshold for an extended period of time, the governor has to raise the chip's supply voltage to bring the SMs back from tunneling.

### 3.5 Noise-Aware Thread Block Scheduler

Some of the largest $\delta I/\delta t$ events observed in our experiments are caused by the launching of application kernels, which activate many thread blocks across the GPU. Kernel launches are generally preceded by relatively idle phases as the computation for the previous kernel wraps up and data for the new kernel is being transferred to the GPU. As this large group of threads begins computation simultaneously, activity and power consumption increase rapidly. The resulting $\delta I/\delta t$ can therefore be very large. This behavior has also been observed in prior work by Leng et al. [3].

To reduce the impact of a new kernel launch on voltage noise, we develop a variation and noise-aware thread block scheduler that takes the characteristics of the power delivery network as well as SM vulnerability to noise into account when making thread block assignments. In order to smooth out the increase in activity, the noise-aware block scheduler controls the SM activation by staggering the assignment of thread blocks to SMs over a longer period of time. The scheduler will first assign the maximum number of thread blocks to the first SM in its queue which will allow that SM to begin execution. Instead of immediately assigning work to the other SMs, the thread block scheduler will wait for a short period of time before assigning threads to the next SM.

The block scheduler is also process variation-aware and takes into account each SM's vulnerability to noise. The scheduler assigns work to the less vulnerable SMs first because $\delta I/\delta t$ is larger when the system is under light load. This improves the performance of the voltage speculation system because it allows it to more aggressively lower $V_{dd}$ when the weak SMs are not active.

Figure 9 shows the effect of a kernel launch from benchmark BFS on supply voltage. The figure shows the lowest $V_{dd}$ across the chip and the number of active thread blocks over time. The top plot shows the baseline block scheduler. We can clearly see that as most thread blocks become active over a short period of time, the voltage droops rapidly by more than 80mV. The bottom plot shows the noise-aware scheduler that staggers the thread block launch over time. We can see that the noise-aware scheduler is very effective at reducing $\delta I/\delta t$ at kernel launch, resulting in a much smoother voltage profile, with no significant droops.

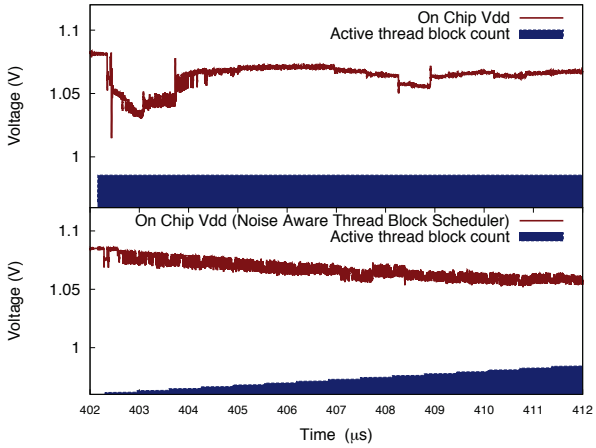


Figure 9: Baseline (top) and noise-aware (bottom) thread block scheduler effects on supply voltage.

## 4. JOINT MODELING OF VOLTAGE NOISE AND PROCESS VARIATION

A modeling infrastructure that captures both voltage noise and process variation in a joint simulation was developed for this work. A distributed power delivery model captures cycle-level voltage fluctuations at thousands of points across the GPU. A process variation model captures the distribution of threshold voltages across the chip. The two models are integrated by computing the distribution of circuit delays across the chip as a function of both the $V_{dd}$ and $V_{th}$ distributions.

### 4.1 Power Delivery Modeling

A chip's power delivery system consists of off-chip and on-chip components. The off-chip network includes a voltage regulator, capacitors used to stabilize supply voltage, wires and other components. Our off-chip power delivery model follows the design layout and component characteristics of the Intel Xeon [31, 32], a system with a similar thermal design power and chip size as the Fermi GPU we model. The circuit layout is shown in Figure 10. Table 1 summarizes the component values.

On the chip, power is delivered through a set of pins and C4 pads. These connect to a network of wires that deliver the required voltage to the various chip components. We model the on-chip power grid using a distributed `RLC` network similar to those used by prior work [5, 32]. Figure 11 shows the circuit layout of the network. Wiring is modeled as an `RL` network with two planes – one for the $V_{dd}$ and one for the $V_{ss}$ – connected by capacitors. Current sinks across the capacitors are used to model the current drawn by the various functional units, as in work by Herrell and Beker [33]. C4 bumps are placed uniformly throughout the entire chip.

The inputs to the model consist of current traces at cycle granularity for each functional unit of the GPU. The

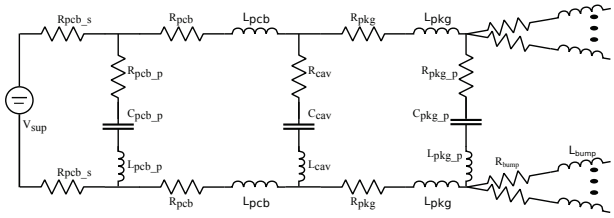


Figure 10: Off-chip component of the power delivery network.

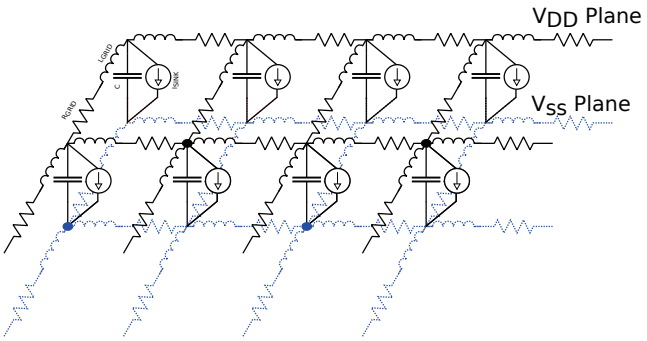


Figure 11: Distributed model of the on-chip power delivery network.

model output is a trace of on-chip $V_{\text{dd}}$ distributions at cycle granularity. A circuit simulator such as SPICE can be used to resolve the power delivery network for each set of inputs. However, because the network is large and it includes many inductive elements, a traditional circuit solver like SPICE would be prohibitively expensive, especially given the need to simulate full benchmarks at cycle granularity. As an alternative, we use a specialized `RLC` solver based on the preconditioned Krylov-subspace iterative method that we developed based on models by Chen and Chen [34]. Our implementation is orders of magnitude faster than SPICE, allowing the simulation of longer benchmarks. We validate our optimized solver against SPICE simulations.

## 4.2 Process Variation Integration

We use `VARIUS` [25] to model within-die process variation effects on threshold voltage ($V_{\text{th}}$). We generate multiple $V_{\text{th}}$ distribution maps of the GPU die. To integrate the theshold voltage and supply voltage distributions we use the Alpha Power Law to model circuit delay as a function of $V_{\text{dd}}$ and $V_{\text{th}}$ (1).

$$Delay = C\frac{V_{\text{dd}}}{(V_{\text{dd}} - V_{\text{th}})^{\alpha}} \quad (1)$$

| Resistance | | Inductance | | Capacitance | |
|---|---|---|---|---|---|
| $R_{pcb,s}$ | $100\mu\Omega$ | $L_{pcb}$ | $45pH$ | | |
| $R_{pkg}$ | $200\mu\Omega$ | $L_{pkg}$ | $6pH$ | | |
| $R_{pcb_p}$ | $300\mu\Omega$ | $L_{pcb_p}$ | $40pH$ | $C_{pcb_p}$ | $1256\mu F$ |
| $R_{pkg_p}$ | $540\mu\Omega$ | $L_{pkg_p}$ | $2.5pH$ | $C_{pkg_p}$ | $120\mu F$ |
| $R_{cav}$ | $150\mu\Omega$ | $L_{cav}$ | $20pH$ | $C_{cav}$ | $1222\mu F$ |
| $R_{bump}$ | $10m\Omega$ | $L_{bump}$ | $50pH$ | | |

Table 1: Off-chip network RLC parameter values.

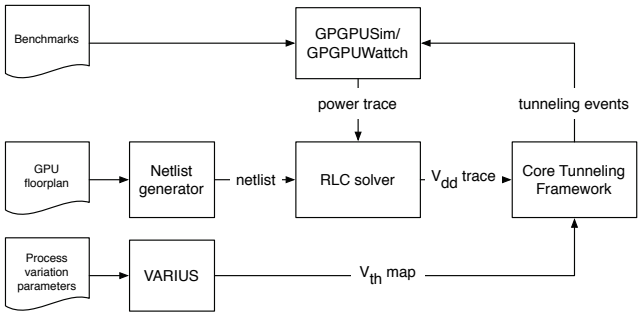


Figure 12: Overview diagram of the evaluation infrastructure.

where $C$ and $\alpha$ are constants used to fit to a technology node. The resulting delay distribution captures the effects of process variation and voltage noise across the die at clock cycle granularity.

Given a fixed clock frequency and a $V_{\text{th}}$ distribution, the model can also compute the lowest safe supply voltage in each point of the die. Figure 3 is an example of such a distribution of minimum safe voltages for a given target frequency. In this work we assume a 5% $\sigma/\mu$ $V_{\text{th}}$ variation.

# 5. EVALUATION METHODOLOGY

Our modeling frameworks includes a GPU simulator with a modified power model, an `RLC` solver, netlist generator and core tunneling framework. Figure 12 shows a high-level diagram of the interaction between the various components of our infrastructure.

## 5.1 GPU Architecture

We use GPGPU-Sim [23] to model a mid-size GPGPU architecture with 512 SIMD lanes (16 SMs with 32 compute lanes each). The SM design and floorplan is inspired by NVIDIA Fermi [35].

Each SM contains an L1 cache, read-only constant and texture caches, and a software-managed shared memory (scratch-pad). There are 32 standard integer/floating-point ALUs (i.e. SPU) and 8 special functional units. All the caches inside the SM are backed up by eight shared L2 banks. The front-end contains instruction fetch (fetch unit plus instruction cache), a highly-banked register file with an associated operand-collector logic, a thread scheduling and branching unit, instruction issue, decode, and scoreboarding logic. Table 2 summarizes the configuration parameters for this architecture.

To estimate per-unit dynamic and leakage power consumption, we use GPUWattch [24] (with a modified McPAT [37]) for functional units and other pipeline internals. For the memory modules (e.g. caches and register file), we use CACTI [38].

Significant changes are needed to existing tools to accurately model cycle-by-cycle changes in power. Most power models, including GPUWattch, compute energy per operation. If the operation is multi-cycle, the energy has to be spread over the entire duration of the operation. This will accurately model the power consumed

| GPU Specs | |
|---|---|
| SMs | 16 |
| L2 Cache | 8 Banks, 128KB/bank |
| DRAM-C | FR-FCFS, 4 MCs |
| Clock | 1440 MHz |
| **SM Specs** | |
| Core | 1440 MHz, in-order pipeline |
| SIMD lanes | SPU/SFU/LSU : 32/8/16 |
| Const-Cache | 8KB |
| Inst-Cache | 2KB |
| L1 Data Cache | 16KB |
| Shared memory | 48 KB , 32 banks |
| Scheduling | GTO [36] |
| Front-end | Max IPC=1, mem. coalescing |
| Max Threads / SM | 1536 |
| Max Blocks / SM | 8 |
| Max Warps / SM | 48 |
| Reg-file | 32684 × 32-bit, 16 banks |

Table 2: GPU architecture parameter details.

| Application | Abbr. | Inst. Count |
|---|---|---|
| AES Cryptography | AES | 28M |
| BFS Graph Traversal | BFS | 17M |
| Coulombic Potential | CP | 126M |
| LIBOR Monte Carlo | LIB | 907M |
| 3D Laplace Solver | LPS | 82M |
| MUMmerGPU | MUM | 77M |
| Neural Network | NN | 68M |
| N-Queens Solver | NQU | 2M |
| Ray tracing | RAY | 71M |
| StoreGPU | STO | 134M |
| Weather Prediction | WP | 215M |

Table 3: Benchmarks used in the evaluation.

in each cycle as the aggregate of multiple (potentially multi-cycle) instructions. Leakage power is computed separately as a function of unit area. A total power trace is generated and converted to a piece-wise linear current trace that is used as an input to the `RLC` solver.

The power delivery model relies on an `RLC` netlist that is die and package specific. Generating this netlist and assigning various parameters requires a detailed floorplan of the GPU. We base our floorplan on the NVIDIA Fermi die using publicly-available information about component locations and relative sizes. Some design choices are based on educated estimates.

## 5.2 Benchmarks

Table 3 summarizes the set of benchmarks we use. These represent a diverse set of CUDA applications, chosen from different suites (Rodinia, Parboil and NVIDIA Compute SDK). The benchmarks are compiled with NVIDIA C Compiler (nvcc) version 4.0.

# 6. EVALUATION

This section characterizes the effects of voltage noise and process variation on the modeled GPU. We evaluate the reduction in $V_{dd}$ margins made possible by our core tunneling framework and the power and energy savings derived from the reduced voltage margins. Finally, we evaluate the robustness of core tunneling under worst-case noise conditions using a synthetic voltage "virus" workload.

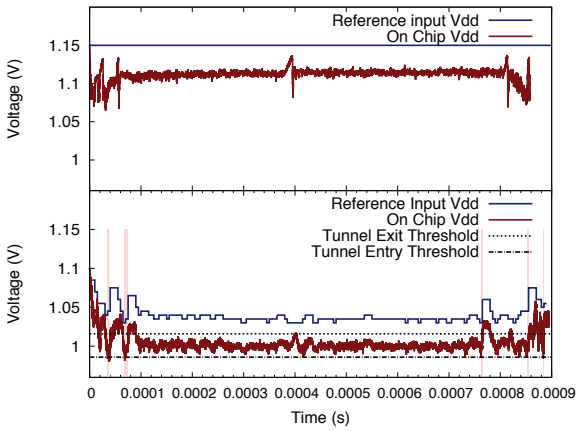


Figure 13: Minimum $V_{dd}$ trace of BFS for the baseline (top) and the system with dynamic $V_{dd}$ adjustment and core tunneling (bottom).

## 6.1 Voltage Margin Reduction

Figure 13 shows the traces of reference input as well as minimum on-die $V_{dd}$ for benchmark BFS. The top graph shows the baseline and the bottom one the core tunneling system. For the baseline system we use a static guardband of 10% added to the lowest safe $V_{dd}$ of the weakest SM. This brings the reference $V_{dd}$ to a fixed 1.15V.

For the core tunneling system (bottom) we can see the dynamic adaptation in reference $V_{dd}$ (dark blue line). The continuous voltage adjustment keeps the on-die $V_{dd}$ closer to the minimun safe $V_{dd}$, reducing the effective voltage margin. The voltage droops below the tunnel entry threshold when $\delta I/\delta t$ activity is high around the $50\mu s$, $80\mu s$ and $750\mu s$ markers. The offending SMs are briefly tunneled as a result during that time (light red shaded areas). In all these cases the power governor raises the reference $V_{dd}$ to help the weak SMs exit from their tunnels.

Figure 14 shows the histogram plots of the minimum on-die $V_{dd}$ in each cycle while running BFS. The top plot is the baseline and the bottom represents the core tunneling system. The plots also show the distribution of the lowest safe SM voltages.

We can see that on the system with core tunneling the runtime $V_{dd}$ distribution has shifted to the left, closer to the distribution of the lowest safe voltages. Moreover, the low-$V_{dd}$ tail is now much shorter, indicating that the benchmark operates towards the energy-efficient low-$V_{dd}$ region most of the time. The slight overlap between the two histograms represents the probability of tunneling events for the SMs whose lowest safe $V_{dd}$ is higher than the tail of the runtime $V_{dd}$ distribution. Since the overlap is small we expect SM tunneling to be relatively rare and performance impact small.

Figure 15 shows the traces of minimum $V_{dd}$ across the die for benchmark CP for the baseline system (top) and the system with core tunneling (bottom). CP is

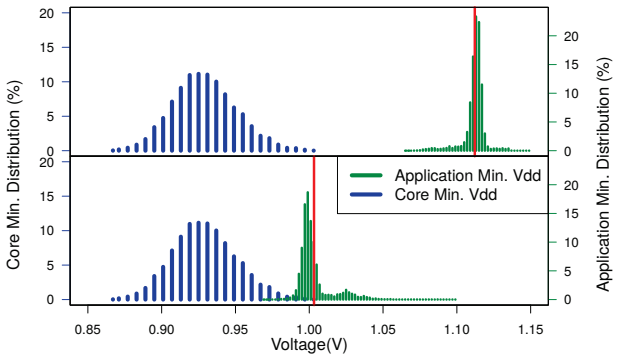


Figure 14: Histograms of minimum $V_{dd}$ for the BFS benchmark for the baseline (top) and the core tunneling system (bottom).

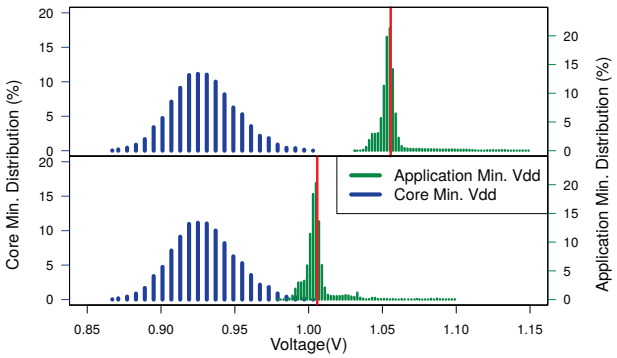


Figure 16: Histograms of minimum $V_{dd}$ for the CP benchmark for the baseline (top) and the core tunneling system (bottom).

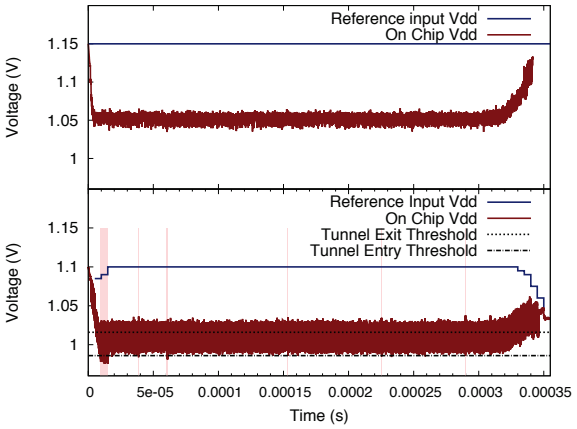


Figure 15: Minimum $V_{dd}$ trace for CP for the baseline (top) and the system with core tunneling (bottom).

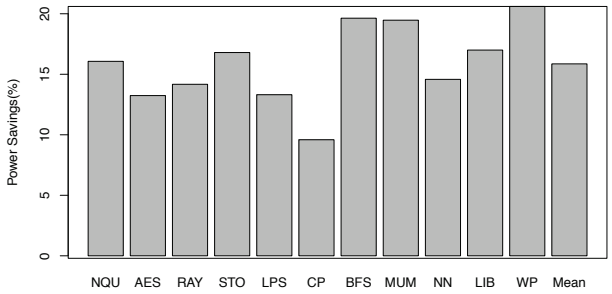


Figure 17: Power savings from dynamic $V_{dd}$ adjustment with core tunneling.

a compute intensive benchmark with very high power consumption pulling the die's $V_{dd}$ low throughout its execution. Given the benchmark's high average power consumption, normal activity leads to relatively large $\delta I/\delta t$ and high peak-to-peak voltage noise.

Figure 16 shows the histogram view for CP. The fact that this benchmark, on average, runs much closer to the lowest safe $V_{dd}$ than BFS can be seen from the short tail towards low voltages and tight distribution on the top chart. As the bottom chart shows, core tunneling lowers CP's average voltage, shifting the on-chip $V_{dd}$ distribution to the left to an average that is close to that of BFS. However, since CP's voltage was already low, the margin reduction is only about half of that achieved for BFS.

## 6.2 Power and Energy Savings

Figure 17 shows the power savings resulting from the lower $V_{dd}$ margins enabled by core tunneling. We see substantial power savings that are as high as 20% for some benchmarks and average around 15%. Benchmarks like CP, which consistently exhibit high power consumption that pushes the $V_{dd}$ close to the safety margin have lower, but still non-trivial power savings of around 10%.

The performance overhead of core tunneling is shown in Figure 18. On average, performance overhead is less than 0.5%. The performance impact is kept low primarily through the variation-aware aspect of our design. Since only SMs that are margin-critical need to be tunneled, and tunneling is a relatively rare event, the overall impact on runtime is minimal.

Theres is little variation in overhead across benchmarks. The benchmarks with the largest performance overhead, namely Raytrace (RAY), StoreGPU (STO) and BFS are characterized by occasional activity coordination across SMs, which leads to large $\delta I/\delta t$. These are so large and fast that the only way our system can handle them is through extended tunneling of the sensitive SMs. This brings a slightly higher performance impact.

CP is a notable case because it is a benchmark with high power consumption and low average voltage, so one might expect its overhead to be high. However, the benchmark has a uniform behavior that allows the power governor to quickly find an ideal $V_{dd}$ level that can be maintained throughout the execution with minimal

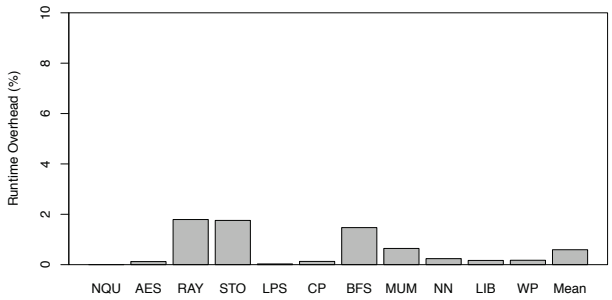


Figure 18: Performance overhead of core tunneling.

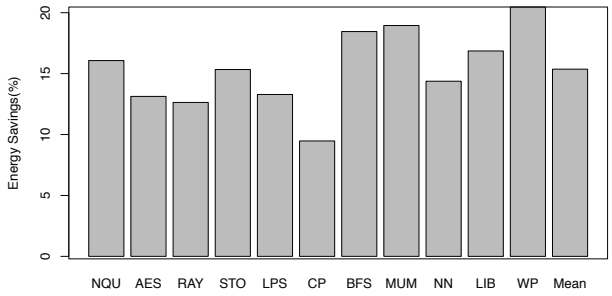


Figure 19: Energy savings from dynamic $V_{\text{dd}}$ adjustment with core tunneling.

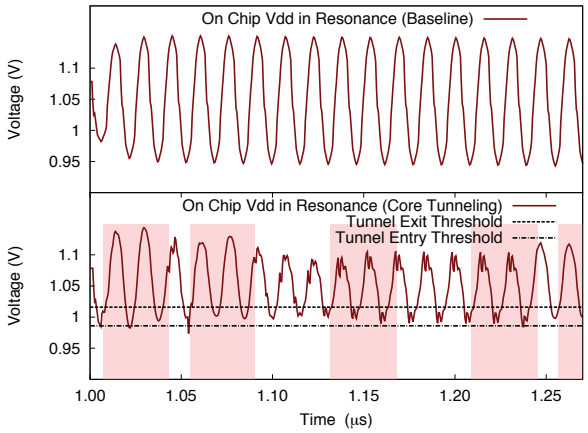


Figure 20: System response to the resonating workload.

| Application | Noise Reduction |
|---|---|
| AES | 40mV |
| BFS | 80mV |
| CP | - |
| LIB | 32mV |
| LPS | 44mV |
| MUM | 35mV |
| NN | 34mV |
| NQU | 50mV |
| RAY | 30mV |
| STO | - |
| WP | 25mV |

Table 4: Peak-to-peak noise reduction from the variation-aware thread block scheduler.

core tunneling activity. As a result, the performance overhead of CP is close to zero.

The large overall power savings and low performance overhead translate into substantial energy savings that average about 15%, as Figure 19 shows.

## 6.3 Resilience to Resonating Voltage Virus

In order to test the robustness of core tunneling beyond the available benchmarks, we construct a worst case "noise virus" consisting of a workload that induces min-to-max oscillations in power, synchronized across the GPU. This is achieved by activating all functional units simultaneously across all SMs. In addition, we tune the frequency of the oscillation to match the resonance frequency of the die obtained from its impedance profile. At resonance, the PDN's impedance reaches its peak, which translates into the worst voltage droops. Note that this is an artificial power trace designed to test the limits of the system, not an actual application.

The top of Figure 20 shows the response of the baseline GPU to the voltage virus. We can see very large voltage oscillations which indicate the power delivery network is resonating. The supply voltage droops in excess of 200mV which would quickly render the chip unusable.

The bottom of Figure 20 shows the response of the GPU to the same workload when core tunneling is enabled. Core tunneling protects the system against the virus in two ways: (1) it tunnels vulnerable SMs when safety margins are exceeded (red shaded areas); (2) it disrupts the resonant pattern because not all SMs are tunneled at the same time and for the same duration due to their variable sensitivities to low $V_{\text{dd}}$. Variable tunneling times across the GPU lead to phase shifts that disrupt the resonance.

As Figure 20 shows, the resonance disruption is so effective that is allows all SMs to exit their tunnels for some sections of the execution, in spite of the very aggressive workload.

## 6.4 Variation-Aware Block Scheduler Effects

Our noise-aware thread block scheduler reduces the magnitude of voltage droops associated with kernel launches. This is achieved by staggering the assignment of thread blocks to SMs, which lowers $\delta I/\delta t$. The largest savings come from benchmarks like BFS which have high initial IPC and experience the largest droops. Workloads such as CP and STO have low initial IPC at kernel launch and therefore benefit the least from the staggered thread block launch.

Table 4 summarizes the amount of peak-to-peak voltage droop reduction for each benchmark.

## 7. RELATED WORK

Previous work [2, 5, 6, 7, 8, 9, 10, 11] has proposed several hardware and software mechanisms for reducing the slope of current changes ($\delta I/\delta t$), which reduces voltage noise. This allows the use of smaller voltage guardbands, saving substantial amounts of power. Most previous work, however, has focused on single-core CPUs [5, 7, 8, 9, 10, 11] or low core-count systems [2, 5, 39]. Gupta et al. [5] characterize within-die voltage variation using a detailed distributed model of the on-chip power-supply grid. Other work [12] has shown that, as the number of cores in future CMPs increases, the effects of chip-wide activity variation overshadow the effects of within-core workload variability. Work by Kim et al. [39] presents a rigorous testing framework for generating kernels that stress the power delivery network and cause worst-case voltage noise. They test and evaluate their framework on real hardware. Recent work has measured voltage noise on production IBM CPUs systems [27].

Voltage noise in GPUs has received relatively little attention. Recent work by Leng et al. [4] has characterized voltage noise in multiple real GPU systems. They found voltage noise to be a significant problem that leads designers to add voltage margins that are as high as 20%. Prior work by Leng et al. [3] has analyzed the issue of voltage noise in simulation with new modeling tools [13] designed for GPUs. They present insights into controlling coordinated activity across SMs. They also evaluate the effect of the large Register File present in GPUs on voltage noise. They propose methods of disrupting these patterns of activity across and within SMs to mitigate voltage noise

Work by Lefurgy et al. [26] uses CPMs to measure timing margins and adjust local DPLL (clock frequency) at core granularity to avoid timing margin errors. Their solution does not apply easily to GPUs that use a single frequency (and clocking) domain for all SMs.

A significant body of work has examined techniques to mitigate or tolerate process variation. These include optimizations of register files and execution units [14], data caches [15], pipeline balancing [16], intelligent floorplanning [17], and core-to-core variation in power [18]. Other work proposed variation-aware thread scheduling [20] to exploit core-to-core variability. Recent work has examined the effects of process variation on GPUs [21].

The combined effects of process, voltage and temperature variation on CPUs were studied by Gupta et al. [22]. They present a local recovery mechanism for dealing with noise-induced timing errors and a framework for dynamically tuning clock frequency and supply voltage at fine spatial granularity to alleviate process and temperature variation effects.

This is the first work we are aware of that jointly models process variation and voltage noise and proposes variation-aware noise mitigation solutions in GPUs.

## 8. CONCLUSION AND FUTURE WORK

This paper demonstrates the importance of accounting for process variation when addressing voltage noise in GPUs. We show that margins added to protect against both process variation and voltage noise are too conservative. We present a new variation-aware framework for reducing these margins by leveraging critical path monitors for noise detection and core tunneling for noise mitigation. Our solution takes advantage of typical GPU workload variability to dynamically lower the $V_{\mathrm{dd}}$ and reduce margins. We find an average of 15% power reduction with very little performance impact, resulting in 15% savings in energy.

Core tunneling could benefit from compiler or system-driven hints about phases of execution that are expected to have high power consumption, especially if this involves coordinated activity across many cores. This could help the power governor preemptively raise the voltage to avoid tunneling. Other solutions that achieve voltage smoothing could reduce the probability of tunneling events and allow $V_{\mathrm{dd}}$ to be lowered further. We hope this work will inspire further research into the joint effects of voltage noise and process variation on modern chips.

## Acknowledgements

The authors would like to thank the anonymous reviewers for their feedback. We also extend special thanks to Vijay Janapa Reddi for suggestions on the camera ready and the members of The Ohio State Computer Architecture Research Lab for discussions and insights on this work. This work was supported in part by the Defense Advanced Research Projects Agency under the PERFECT (DARPA-BAA-12-24) program and the National Science Foundation under grant CCF-1253933.

## 9. REFERENCES

[1] N. James, P. Restle, J. Friedrich, B. Huott, and B. McCredie, "Comparison of split-versus connected-core supplies in the POWER6 microprocessor," in *International Solid-State Circuits Conference (ISSCC)*, pp. 298–604, February 2007.

[2] V. J. Reddi, S. Kanev, W. Kim, S. Campanoni, M. D. Smith, G.-Y. Wei, and D. Brooks, "Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling," in *International Symposium on Microarchitecture (MICRO)*, pp. 77–88, December 2010.

[3] J. Leng, Y. Zu, and V. Reddi, "GPU voltage noise: Characterization and hierarchical smoothing of spatial and temporal voltage noise interference in GPU architectures," in *International Symposium on High Performance Computer Architecture (HPCA)*, pp. 161–173, February 2015.

[4] J. Leng, A. Buyuktosunoglu, R. Bertran, P. Bose, and V. J. Reddi, "Safe limits on voltage reduction efficiency in GPUs: A direct measurement approach," in *International Symposium on Microarchitecture (MICRO)*, pp. 294–307, 2015.

[5] M. S. Gupta, J. Oatley, R. Joseph, G.-Y. Wei, and D. Brooks, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *Design Automation and Test in Europe (DATE)*, pp. 624–629, April 2007.

[6] M. S. Gupta, K. K. Rangan, M. D. Smith, G.-Y. Wei, and D. Brooks, "Towards a software approach to mitigate voltage emergencies," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 123–128, August 2007.

[7] M. S. Gupta, K. K. Rangan, M. D. Smith, G.-Y. Wei, and D. Brooks, "DeCoR: A delayed commit and rollback mechanism for handling inductive noise in processors," in *International Symposium on High Performance Computer Architecture (HPCA)*, pp. 381–392, February 2008.

[8] M. S. Gupta, V. Reddi, G. Holloway, G.-Y. Wei, and D. Brooks, "An event-guided approach to reducing voltage noise in processors," in *Design Automation and Test in Europe (DATE)*, pp. 160–165, April 2009.

[9] R. Joseph, D. Brooks, and M. Martonosi, "Control techniques to eliminate voltage emergencies in high performance processors," in *International Symposium on High Performance Computer Architecture (HPCA)*, pp. 79–90, February 2003.

[10] M. D. Powell and T. N. Vijaykumar, "Pipeline muffling and a priori current ramping: architectural techniques to reduce high-frequency inductive noise," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 223–228, August 2003.

[11] V. J. Reddi, M. S. Gupta, G. Holloway, G. yeon Wei, M. D. Smith, and D. Brooks, "Voltage emergency prediction: Using signatures to reduce operating margins," in *International Symposium on High Performance Computer Architecture (HPCA)*, February 2009.

[12] T. N. Miller, R. Thomas, X. Pan, and R. Teodorescu, "VRSync: Characterizing and eliminating synchronization-induced voltage emergencies in many-core processors," in *International Symposium on Computer Architecture (ISCA)*, pp. 249–260, June 2012.

[13] J. Leng, Y. Zu, M. Rhu, M. Gupta, and V. J. Reddi, "GPUVolt: Modeling and characterizing voltage noise in GPU architectures," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 141–146, ACM, 2014.

[14] X. Liang and D. Brooks, "Mitigating the impact of process variations on processor register files and execution units," in *International Symposium on Microarchitecture (MICRO)*, pp. 504–514, IEEE Computer Society, December 2006.

[15] S. Ozdemir, D. Sinha, G. Memik, J. Adams, and H. Zhou, "Yield-aware cache architectures," in *International Symposium on Microarchitecture (MICRO)*, December 2006.

[16] A. Tiwari, S. R. Sarangi, and J. Torrellas, "ReCycle: Pipeline adaptation to tolerate process variation," in *International Symposium on Computer Architecture (ISCA)*, June 2007.

[17] E. Humenay, D. Tarjan, and K. Skadron, "The impact of systematic process variations on symmetrical performance in chip multi-processors," in *Design Automation and Test in Europe (DATE)*, April 2007.

[18] J. Donald and M. Martonosi, "Power efficiency for variation-tolerant multicore processors," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 304–309, October 2006.

[19] R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "Mitigating parameter variation with dynamic fine-grain body biasing," in *International Symposium on Microarchitecture (MICRO)*, pp. 27–39, December 2007.

[20] R. Teodorescu and J. Torrellas, "Variation-Aware Application Scheduling and Power Management for Chip Multiprocessors," in *International Symposium on Computer Architecture (ISCA)*, pp. 363–374, June 2008.

[21] P. Aguilera, J. Lee, A. Farmahini-Farahani, K. Morrow, M. Schulte, and N. S. Kim, "Process variation-aware workload partitioning algorithms for GPUs supporting spatial-multitasking," in *Design Automation and Test in Europe (DATE)*, pp. 1–6, March 2014.

[22] M. Gupta, J. Rivers, P. Bose, G.-Y. Wei, and D. Brooks, "Tribeca: Design for PVT variations with local recovery and fine-grained adaptation," in *International Symposium on Microarchitecture (MICRO)*, pp. 435–446, December 2009.

[23] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 163–174, April 2009.

[24] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "GPUWattch: Enabling energy optimizations in GPGPUs," in *International Symposium on Computer Architecture (ISCA)*, June 2013.

[25] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "VARIUS: A Model of Parameter Variation and Resulting Timing Errors for Microarchitects," *IEEE Transactions on Semiconductor Manufacturing*, vol. 21, pp. 3–13, February 2008.

[26] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno, and J. B. Carter, "Active management of timing guardband to save energy in POWER7," in *International Symposium on Microarchitecture (MICRO)*, pp. 1–11, December 2011.

[27] R. Bertran, A. Buyuktosunoglu, P. Bose, T. Slegel, G. Salem, S. Carey, R. Rizzolo, and T. Strach, "Voltage noise in multi-core processors: Empirical characterization and optimization opportunities," in *International Symposium on Microarchitecture (MICRO)*, pp. 368–380, December 2014.

[28] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *International Symposium on Microarchitecture (MICRO)*, pp. 7–18, December 2003.

[29] R. McGowen, C. A. Poirier, C. Bostak, J. Ignowski, M. Millican, W. H. Parks, and S. Naffziger, "Power and temperature control on a 90-nm Itanium family processor," *IEEE Journal of Solid-State Circuits*, vol. 41, pp. 229–237, January 2006.

[30] "Intel Core™ i7 Processor." `http://www.intel.com`.

[31] "Voltage regulator module (VRM) and enterprise voltage regulator-down (EVRD) 11.1 design guidelines," tech. rep., Intel Corp., September 2009.

[32] X. Zhang, T. Tong, S. Kanev, S. K. Lee, G.-Y. Wei, and D. Brooks, "Characterizing and evaluating voltage noise in multi-core near-threshold processors," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 82–87, 2013.

[33] D. Herrell and B. Beker, "Modeling of power distribution systems for high-performance microprocessors," *IEEE Transactions on Advanced Packaging*, vol. 22, no. 3, pp. 240–248, 1999.

[34] T.-H. Chen and C. Chen, "Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods," in *Design Automation Conference (DAC)*, pp. 559–562, June 2001.

[35] P. N. Glaskowsky, "NVIDIA's Fermi: The first complete GPU computing architecture," September 2009. White Paper.

[36] T. G. Rogers, M. O'Connor, and T. M. Aamodt, "Cache-conscious wavefront scheduling," in *International Symposium on Microarchitecture (MICRO)*, pp. 72–83, December 2012.

[37] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," in *International Symposium on Microarchitecture (MICRO)*, pp. 469–480, December 2009.

[38] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A Tool to Model Large Caches," Tech. Rep. HPL-2009-85, HP Labs, 2009.

[39] Y. Kim, L. K. John, S. Pant, S. Manne, M. Schulte, W. L. Bircher, and M. S. S. Govindan, "AUDIT: Stress testing the automatic way," in *International Symposium on Microarchitecture (MICRO)*, pp. 212–223, December 2012.