Predict; Don’t React for Enabling Efficient Fine-Grain DVFS in GPUs

Srikant Bharadwaj†
Microsoft
Redmond, WA
srikant.bharadwaj@microsoft.com

Shomit Das†
Samsung
Austin, TX
shomit.das@samsung.com

Kaushik Mazumdar
Advanced Micro Devices
Fort Collins, CO
kaushik.mazumdar@amd.com

Bradford Beckmann
Advanced Micro Devices
Bellevue, WA
brad.beckmann@amd.com

Stephen Kosonocky†
Uhnder
Fort Collins, CO
stephen.kosonocky@uhnder.com

ABSTRACT
With the continuous improvement of on-chip integrated voltage regulators (IVRs) and fast, adaptive frequency control, dynamic voltage-frequency scaling (DVFS) transition times have shrunken from the microsecond to the nanosecond regime, providing immense opportunity to improve energy efficiency. The key to unlocking the continued improvement in V/f circuit technology is the creation of new, smarter DVFS mechanisms that better adapt to rapid fluctuations in workload demand.

It is particularly important to optimize fine-grain DVFS mechanisms for graphics processing units (GPUs) as the chips become ever more important workhorses in the datacenter. However, GPU’s massive amount of thread-level parallelism makes it uniquely difficult to determine the optimal V/f state at run-time. Existing solutions—mostly designed for single-threaded CPUs and longer time scales—fail to consider the seemingly chaotic, highly varying nature of GPU workloads at short time scales.

This paper proposes a novel prediction mechanism, PCSTALL, that is tailored for emerging DVFS capabilities in GPUs and achieves near-optimal energy efficiency. Using the insights from our fine-grained workload analysis, we propose a wavefront-level program counter (PC) based DVFS mechanism that improves program behavior prediction accuracy by 32% on average as compared to the best performing prior predictor for a wide set of GPU applications at 1µs DVFS time epochs. Compared to the current state-of-art, our PC-based technique achieves 19% average improvement when optimized for Energy-Delay² Product (ED²P) at 50µs time epochs, reaching 32% when operated with 1µs DVFS technologies.

CCS CONCEPTS
• Hardware → Power and energy → Power estimation and optimization → Chip-level power issues; • Computer systems organization → Architectures → Parallel architectures → Single instruction, multiple data.

†The work described in this paper was conducted while the authors were affiliated with Advanced Micro Devices, Inc.

© 2024 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ASPDAC ’24, April 27–May 1, 2024, San Diego, CA, United States
© 2024 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 979-8-4007-0394-2/23/03...
https://doi.org/10.1145/3623278.3624756

KEYWORDS
Dynamic Voltage Frequency Scaling, Graphics Processing Unit.

1 Introduction
Dynamic Voltage Frequency Scaling (DVFS) techniques continue to improve the energy efficiency of modern computing architectures generation after generation [1]. The expansive benefits of DVFS come from the cubic relationship of voltage to power consumption owing to the basic dynamic power equation \( P = CV^2f \), where frequency \( f \) also reduces with voltage \( V \). The insight behind DVFS is that systems exhibit phased behavior where their performance has varying dependence on per-component operating frequencies, which in modern systems can be modified dynamically with supply voltage. Dynamically adjusting frequencies and voltages can minimize unnecessary power consumption, resulting in more power-efficient architectures when managed properly.

The recent advances in the design of on-chip voltage regulators [2, 3, 4, 5, 6] open new opportunities for finer frequency control and fundamentally push the power efficiency of future systems to higher limits [2]. Unlike earlier off-chip voltage regulator designs, modern circuit technology enables the integration of regulators within the chip. These integrated voltage regulators (IVR) were introduced just a few years ago [6] and continue to improve, providing ever-decreasing response latencies [7]. In particular, digital low dropout (LDO) regulators and switched regulators have emerged as an efficient new class of IVRs because of their low area footprints and small quiescent current requirements [8, 9, 10, 11]. Thus, they are increasingly being adopted in modern multicore DVFS systems [12, 13, 14, 15].

Complementary to IVR improvements, clock generation units have also undergone significant recent advances. IVRs can be combined with Phase-Locked Loops (PLLs) and digital frequency synthesizer (DFS) solutions where integer/fractional dividers are used to generate more traditional coarse-grain power/clock (V/f) domains. However, an alternative emerging clocking scheme gaining popularity is voltage-adaptive Frequency-Locked Loops (FLLs) [6, 7, 8]. With these FLLs, V/f domains can be created by providing a clock frequency proportionate to the domain’s supply voltage, which in turn is controlled by IVRs. When supply voltage is adjusted, these new FLLs facilitate frequency adjustment transition times within few nanoseconds [6, 7, 8, 16, 17, 18] primarily due to their shorter settling times, allowing for iso-frequency time epochs of only a few microseconds or less. As a result, these faster IVRs enable more energy efficiency and thereby...
Sensitivity Prediction

Figure 1: (a) Opportunity for improving GPU ED²P at different DVFS time epochs. (b) Program behavior prediction accuracy of the state-of-the-art sensitivity estimation model [20] compared to an even more accurate reactive (theoretical) estimation model and the predictive mechanism proposed in this paper.

Maximizing the efficiency improvement with fine-grain DVFS requires an accurate estimation of frequencies’ impact on program behavior of elapsed time epochs and prediction of future time epochs. Most of the prior DVFS techniques use analytical [24, 25] or machine learning estimation models [4, 31, 32] to determine a heuristic relationship between performance and the optimal operating frequency or phase. These models [1], designed for single-threaded CPUs, rely on estimating the execution time of any given workload or work segment at different frequencies before selecting an optimal frequency.

Such techniques have also been extended to GPU Compute Units (CUs) [20], but lead to lower accuracies because they do not account for the many threads simultaneously executing in the CU (more in Section 4.1). Figure 2(a) shows how models created for CPUs have been previously applied to CUs in GPUs leading to high inaccuracies in optimal frequency prediction of V/f domains for finer-grain time scales (more in Section 4.1).

Beyond the problem of estimation, existing mechanisms [10, 20, 24, 33, 34] use the estimation of the current epoch to set the frequencies of subsequent time epochs and are thus reactive in nature. To put it another way, they estimate a work segment's performance after it has been completed, and then they immediately apply that assessment to the subsequent work segment or time epoch. However, but even with the most accurate estimates in a loss of at least 16% loss of potential efficiency when used reactively at 50µs (Figure 1(b)). Furthermore, the inability of reactive policies to achieve maximum power savings amplifies as we approach finer-grain DVFS time epochs, as seen in Figure 1(a). Thus, leveraging the improvement in circuit voltage and frequency control technologies requires an understanding of workloads at these fine-grain time epochs as well as the development of a new mechanism which can provide better prediction accuracy than a traditional reactive system.

To that end, we analyze GPU workloads at these fine-grain time scales and observe that GPUs showcase varying phase behavior, thus reducing the predictability of the system. For this, we define a frequency sensitivity metric and adopt a methodology that measures it at fine-grain time epochs (Section 3.1). We then formally define the sensitivity metric (Section 3.2) and employ it to identify the fine-
grain phase patterns. We observe that GPUs show highly varying phase behavior (Section 3.3) during kernel execution. Further, we identify that the massive multithreading inherent to GPUs prevent the direct application of CPU-based models for determining phase behavior at fine-grain time scales (Section 4.1).

Based on these observations, we propose a novel prediction mechanism (as opposed to reaction), PCSTALL, that predicts the frequency sensitivity of future fine-grain time epochs in GPUs resulting in near optimal energy efficiency. PCSTALL (Section 4.4) adopts the simple models created for CPUs to wavefronts instead of CUs as shown in Figure 2(b). Further, we design a novel way to utilize Program Counter (PC) value of wavefronts (or warps) to predict the phase instead of reacting, thus delivering higher frequency sensitivity prediction accuracy (up to 81%). Consequently, our approach provides 19% average improvement in the ED-P compared to state-of-art techniques at 50µs time epochs and reaches up to 32% improvement with 1µs DVFS time epoch technology.

2 Related Work and Background

DVFS is a widely used technique for improving energy efficiency. The general idea of DVFS is to dynamically change the voltage and frequency of operation to match the current requirements of a processing element (or group of processing elements). While the classical objective of DVFS has been to improve power savings [1], recent works have also proposed DVFS for improving performance under a power constraint [25, 35, 36]. Exploiting DVFS requires a mechanism that can estimate performance at different frequencies and then adjusts the clock frequency and voltage supply to optimize an objective function. Several previous works have proposed a wide variety of mechanisms for estimating workload phase behavior and then adjusts the clock frequency and voltage supply to optimize an objective function. The following subsections summarize these works.

2.1 Technology Improvements in Fast-Fine-Grain DVFS

The voltage regulator transition latency significantly influences the overall efficiency achieved by a DVFS-enabled system. Thus, we first summarize the history of voltage regulator technology.

**Challenges in Traditional DVFS.** Traditional V/f scaling using off-chip regulators exhibited voltage transition latencies between V/f states in the order of hundreds of microseconds [37]. This allowed for DVFS management to be performed by a firmware [38, 39, 40]. In addition, traditional PLLs have long latency re-lock times on the order of multiple microseconds which creates a restrictive upper bound on efficient DVFS times for off-chip regulation. In contrast, the higher speed DVFS solutions targeted in this paper require a much lower latency frequency adjustment mechanism.

**Integrated Voltage Regulators.** With the advent of different types of integrated voltage regulators (IVRs) [2, 3, 4, 6, 41, 42, 43], it has now become possible to significantly reduce transition times between voltage-frequency states. This is mainly because they can provide a fast transient response to dynamically varying loads. There are multiple families of IVRs based on the methodology used for voltage regulation: linear resistive LDOs, switched capacitor regulators, and inductor-based buck converters.

Several efforts have been made to improve the design of these IVRs allowing transition times in the order of few nanoseconds. However, with the advancement of packaging technologies, such as using substrate layers for air core inductors [6] or embedding magnetic arrays inside package [44], die area is no longer a large concern and switched regulators have since been shown to provide fast responses [6, 16, 45]. LDOs on the other hand, operate on feedback-based resistive voltage division and thus can be implemented with entirely digital components. Recent works have shown LDOs provide fast settling times in the range of nanoseconds [8, 17, 18] and they have been proven to be beneficial in commercial designs [12, 13, 14, 23].

**Phase-Locked Loops.** Alternatively, high-speed digital frequency synthesis (DFS) techniques, developed for adaptive clock stretching [7, 13], can also be used to quickly select a derivative frequency relative to a locked maximum reference frequency generated from a PLL. Recent commercial products [13] have been shown to have a clocking system that can stretch a reference clock by a programmable amount in the nanosecond time frame. Several studies [2, 19, 46] have used the technology to show that fine-grain DVFS mechanisms can improve the overall energy efficiency of a system.

Earlier DVFS works from a decade ago proposed DVFS policies in the order of hundreds of milliseconds [25], which reduced to a single millisecond [47] policies a few years later. Recent works [20] on GPU DVFS techniques have described time epochs as low as tens of microseconds. While technology improvements have made commercial GPU products and prototypes showcase frequency transition times within nanoseconds [12, 15, 23], there has been minimal work done in supporting the reduced DVFS time epochs. The trend of technological improvement in IVRs, PLLs, and FLLs promises DVFS iso-frequency time epoch duration times to reduce from the current status of hundreds of microseconds to a few microseconds in the near future in commercial products.

Apart from the temporal aspect, on-chip regulators also impact the spatial effectiveness of a DVFS-based system by enabling multiple V/f islands within a chip [7]. With multi-core CPUs becoming commonplace, there has been a lot of CPU-related work managing multiple clock domains [48, 49]. Several studies have concluded that managing multiple separate clock domains provides improved energy efficiency as compared to a single time domain [35, 49, 50, 51]. Thus, our work focuses on fast as well as multiple clock domain support in GPUs. In the remainder of this paper, we use the word fine-grain to describe these targeted fast-transitioning, finely tunable domains.

2.2 DVFS Control Mechanisms

There has been extensive literature [24, 33, 34, 52, 53] on controlling DVFS systems for both CPUs and GPUs [20, 26, 27, 28, 54]. DVFS mechanisms have also been extended to work for memory subsystem and CPU + Memory combined management [32, 47, 55, 56]. Such control policies require monitoring dynamic system behavior, estimating performance at other possible V/f states, and adjusting the operating V/f states for future time epochs to meet power-performance targets. Thus, exploiting the DVFS mechanism to its maximum potential requires the ability to predict the future performance of a system at different operating V/f states. If we know the performance at each operating state, we can transition to an optimal frequency given an objective function. The general approach towards this challenge is to adopt heuristic methods to predict the performance of a future time epoch at different frequencies. Solutions usually involve two key challenges. First, when executing a work segment at one operating frequency for a time epoch, it is difficult to estimate the performance at other
frequencies for the same work segment. This is a major challenge because each workload scales differently at different frequencies. Second, even with accurate estimation of an elapsed time epoch, it is a challenge to predict what the workload behavior will be for a future time epoch, especially for workloads with highly varying behavior. Prior solutions to these two challenges are described next.

2.3 Estimating Frequency Sensitivity

Estimating frequency sensitivity to performance of an elapsed work segment has been studied in-depth by several previous works. While earlier works studied the estimation of frequency sensitivity using a linear scaling approach [57], performance counters [10, 24, 33, 34, 58] have been shown to be more accurate. These approaches mainly fall into two major categories: (a) sampling, and (b) analytical. The sampling models estimate a scaling factor (sensitivity) by executing a workload at different V/f points. The general idea is that the compute performance scales with frequency, while asynchronous memory phases remain constant. Most importantly, the intuition behind this generalization is that any CPU workloads (even multithreaded) can be approximated by a single in-order thread of execution. Although CPUs can execute out-of-order, memory stalls can still dominate portions of execution time by measuring the latency of leading loads.

Critical Path. The Critical Path [10] model was proposed to consider a realistic memory subsystem and DRAM. The model book-keeps the timestamps such that loads on the critical path of execution are taken into consideration. The latencies of these critical path loads are then combined for estimating the asynchronous time spent by the core.

CRISP. The CRISP model [20] extended the Critical Path model by considering the high number of store stalls observed in GPU, as well as the high memory-computation overlap. They calculate both core and memory time by selectively measuring store stalls and computation overlap. It is important to note that, CRISP assumes a CU to be equivalent to a CPU core and approximates execution within a CU to a single-threaded workload as shown in Figure 2(a).

To our knowledge, none of these prior CPU approach accounts for multiple threads executing on the same core, which is common in GPUs. The CRISP model extends the CPU model to GPU CU leading to low accuracies in estimation at fine-grain time scales as we will discuss later.

2.4 Predicting Sensitivity

The second part of the challenge, which has often been overlooked by previous works, is predicting the sensitivity of future time epochs. The performance modeling techniques discussed above mostly restrict themselves to reactive approaches. In other words, they estimate the performance of a work segment after executing it and then immediately apply the estimation to the next work segment or time epoch, as shown in Figure 3(a). Such reactive approaches are generally referred to as last-value predictors [25]. Longer-term value predictions, which predict the duration of a continuous phase, were also proposed [60]. Such reactive systems rely on the assumption that workloads exhibit similar behavior over consecutive work segments or time epochs [10, 20]. A few efforts that improve upon this assumption use a global phase history table to predict the variation across consecutive time epochs [25, 61]. A recent work [4] utilizes a Q-learning mechanism to predict V/f state directly using a set of attributes. As we will see later in Section 3.3, these reactive policies are not sufficient for obtaining maximum power efficiency gains in the case of fine-grain DVFS in GPUs.

3 Fine-Grain Voltage-Frequency Islands

To explore fine-grain DVFS, we consider an AMD MI200TM GPU [62] with multiple clock domains across the CUs as shown in Figure 4. Specifically, for most of our evaluations, we assume a fine-grain clock domain that comprises of one CU along with its L1 caches. Later, Section 6.5 shows our single-CU observations apply to systems with multiple CUs in a clock domain as well.

3.1 Fixed-Time Epoch

For DVFS domains tunable to different frequencies, even as low as 1 µs, it is imperative to manage the domains with a purely hardware-mechanism. A software-managed mechanism would be unable to react fast enough at such short timescales [32]. At these short time scales, we advocate for a fixed time-window-based approach for managing DVFS because the system can consistently adapt to the
minor changes in the workload behavior with minimal overhead to IVRs and PLLs. In contrast, a fixed-instruction-window approach could either miss productive transition possibilities or encounter frequent unproductive transitions leading to resonance noise [63]. This effect is especially magnified for GPUs where there is high variation in instructions committed over time (as discussed later in this section). Thus, a fixed-time epoch control ensures that transitions occur quickly while amortizing the transition power overhead.

3.2 Characterization of Fine-Grain Phases

DVFS requires a quantitative characterization of the phase so that an optimal frequency can be chosen. Earlier works characterized workloads based on the sensitivity of execution time to operating frequency [25, 64]. Such characterization helps optimize DVFS at a thread-granularity to meet performance deadlines and minimize EDP. Later works utilizing finer granularities of DVFS have focused on a fixed number of instructions [10, 20, 65, 66] and characterized execution time sensitivity in a similar manner. However, a different metric would be required to analyze the frequency sensitivity of fixed-time epochs. The general approach to measuring frequency sensitivity of different phases is to characterize them as compute-intensive—if the amount of work done increases substantially with frequency—or memory-bound—if otherwise.

To quantitatively measure the work done by a system, we consider the number of instructions committed for any given time epoch. For a sampled set of unique time epochs (Section 5.1) in the application coed, Figure 5 shows the number of instructions committed at different operating frequencies. As expected for the compute-intensive epochs, the number of instructions committed increases with increases in frequency (high slope). Meanwhile for the memory-bound epochs, the increase is low or negligible (low slope).

The key observation in Figure 5 is that the number of instructions committed has a mostly linear relationship to operating frequencies for the range most appealing to fine-grain DVFS. Performing linear regression over the limited value range obtained for different time epochs and workloads corroborated this observation with an average $R^2$ value of 0.82. This strong correlation lets us model the performance at the targeted frequency range using a linear model. In other words, we could model the number of instructions executed $I_f$ at frequency $f$ as:

$$I_f = I_0 + Sf$$

The term $I_0$ signifies the minimum number of instructions that would be executed. The term $S$ here quantifies the performance sensitivity of the fine-grain time epoch to the operating frequency. We define Sensitivity of a time epoch assuming a given starting condition. Sensitivity signifies the potential increase in committed instructions for an increase in unit frequency. This term, therefore, quantifies the phase of the fine-grain workload; higher sensitivity values indicate a more compute-intensive work segment, while a lower value signifies a memory-intensive work segment.

$$Sensitivity = Sens_x = \frac{\Delta \text{Instructions Committed in X domain}}{\text{Frequency of X domain}}$$

It is important to note that the linear model is empirical and is suitable for our targeted range of GPU DVFS frequencies (1.0-3.0 GHz), but not all frequencies. We observed that frequencies as low as 0.8 GHz and as high as 3.5 GHz exhibit a considerably similar
understand the phased behavior at the fine-grain time scales. The sensitivity frequencies (e.g., 200 MHz), we expect this empirical model to be insufficient.

Across a variety of workloads, Figure 7(a) quantifies the average sensitivity of future time epochs. Figure 8 shows how wavefront-level sensitivity to estimate the total sensitivity of the CU. the architecture involved. We take these challenges into account when designing our fine-grain DVFS prediction mechanism.

4 Fine-Grain Sensitivity Prediction for GPUs

4.1 GPU Execution Hierarchy

The underlying execution mechanism of a CU is, however, vastly different than the single-thread model assumed by prior GPU DVFS works. A CU executes many wavefronts (sometimes called warps) in a lock-step mechanism with individual program counters (PC). At any single point of time, the CU can host a fixed number of wavefront contexts which are called wavefront slots as shown in Figure 4 (black boxes). This results in an arbitrary mix of instructions being executed in any given time epoch. The mix of instructions is dependent on the individual progress of the wavefronts and could potentially be executed in any order (subject to individual wavefront ordering). This arbitrary mixture of instructions from different wavefronts thus determines the work segment of a CU in any given time epoch. Such an execution mechanism results in three levels of variation.

First, the exact sequence of instructions committed by the CU in any given time epoch can highly vary for different operating frequencies. Second, because individual wavefronts progress at different rates, there is a high variation in instruction mixture and frequency sensitivity across consecutive time epochs. Last, each wavefront goes through different phases depending on the nature of their instructions and other environmental factors (memory traffic). These characteristics pose a challenge to the prediction of sensitivity for future time epochs. Figure 8 shows how wavefront-level variations within the sampled application BwdBN affect a CU’s sensitivity across time epochs. Although some of these characteristics may exist in a hyper-threaded CPU core, their effect is considerably lower because of the relatively low degree of hyper-threading (~2 threads) compared to a CU (~40 wavefronts).

4.2 Wavefront-Level Estimation

We propose utilizing wavefront-level sensitivity to estimate the total sensitivity experienced by a clock domain. The execution of wavefronts is comparative to the execution of in-order CPU threads. Each wavefront has a PC denoting the next instruction to be executed and wavefronts execute and commit instructions in-order. Also, similar to multi-threaded CPU execution, when a wavefront stalls waiting for load dependencies, other ready wavefronts consume execution resources and progress. Due to their similarities
to CPU threads, we apply the prior CPU DVFS models described in Section 2.3, to estimate per-wavefront sensitivity for any time epoch.

The next step is to aggregate the sensitivity obtained at the wavefront-level to the clock-domain level. Because of the commutative nature of the sensitivity metric defined in Section 3.2, the sensitivity of a clock domain would just be the sum of the sensitivities of the constituent CUs. The sensitivity of a CU itself would involve the combined sensitivities of the constituent wavefronts. Formally, the previously stated equation can be generalized to a clock domain as:

$$\text{Sens}_{\text{domain}} = \frac{\Delta \text{Ins}}{\Delta \text{Freq}} = \frac{1}{\sum_{i=1}^{n \text{CU}_i} \sum_{j=1}^{n \text{WF}_{ij}} \Delta \text{Ins}_{WF_{ij}}} = \frac{1}{\sum_{i=1}^{n \text{CU}_i} \sum_{j=1}^{n \text{WF}_{ij}} \text{Sens}_{WF_{ij}}}$$

Thus, having a scalable sensitivity metric helps us determine the performance of a clock domain without losing detail from wavefront-level variations.

### 4.3 Wavefront-Level Prediction

The main challenge for predicting GPU frequency sensitivity assuming fine-grain DVFS is their high variation, as discussed in Section 3.3. Due to the high variability, it is not sufficient to utilize a simple reactive mechanism. Instead, the mechanism must anticipate the variation in sensitivity beforehand.

**Wavefront Phase.** One of the key reasons for the high variation in frequency sensitivity is the independent progress of each wavefront. Moreover, the wavefronts themselves exhibit phases depending on the nature of instructions and environmental conditions as shown in Figure 9. To observe the predictability of wavefronts, we measure the difference in wavefront-level sensitivity in consecutive iterations starting from the same starting PC address.

Figure 9: PC-based phased prediction model leverages the repetitive nature of kernel execution. This example shows how later iterations leverage information from prior iterations.

Figure 10: Average sensitivity change in CU sensitivity across consecutive iterations (1µs) starting from same respective wavefront PC address.

Figure 11: (a) Change in sensitivity (1 µs) observed for different wavefront slots in quickS (highest inter-wavefront variation). (b) Average relative change in sensitivity when different index offset values for PC-table with a CU-level granularity. (Epoch 0 vs Epoch 2 in Figure 9). This study was designed to identify whether wavefronts would exhibit similar sensitivity when executing the same sequence of instructions.

Figure 10 shows the change in sensitivity of wavefronts starting from any given PC over consecutive iterations. We observe that the change in sensitivity over consecutive iterations of the same sequence of instructions for a given wavefront (1WF) is only 25 ΔIns/ΔGHz on average. This is much lower than the 105 ΔIns/ΔGHz average change observed for consecutive time epochs (Figure 7) and shows that the inherent sensitivity of a time epoch in any wavefront is primarily determined by the nature and order of the instructions executed. The different granularities (64CU, CU, WF) represent the cases when iterations within the respective boundaries were considered. We observe that the variation does not change by a lot when sensitivities of different wavefronts executing within a single CU (1CU) or a single GPU (64CU) are compared to each other at identical starting PC addresses. This shows that a PC address can be used to predict sensitivity of a wavefront if the same wavefront or any other wavefront has reported its sensitivity starting from the same PC address.

**Contention Across Wavefronts.** To understand the effect of contention across wavefronts, we compared the change in sensitivity for a wavefront to the highest priority wavefront. Figure 11(a) shows the average relative difference in sensitivity observed for different wavefront slots for quickS application. This shows that most of the difference in performance is because of contention between wavefronts within a CU; with the highest priority wavefront experiencing no impact on sensitivity and the lower priority wavefronts experiencing an increased relative change in sensitivity. We attribute this variation to the GPU’s ‘oldest-first’ wavefront scheduling policy which prioritizes the execution of the oldest ready wavefront.

### 4.4 Novel PC-Based Phase Prediction Unit

The observations above motivated us to build PCSTALL, a wavefront-level PC-based sensitivity predictor as shown in Figure 13. The predictor leverages the repetitive behavior of instruction sequences across wavefronts or iterations to accurately predict sensitivity and is composed of two mechanisms: update and lookup. At the end of each epoch, each wavefront estimates their epoch’s sensitivity and stores the estimate in a table (update mechanism). For the next epoch, the predictor accesses the same table using each wavefront’s next PC to estimate the overall sensitivity (lookup mechanism). Such a PC-based predictor is more feasible in GPU kernels where the code size is often limited and iteratively executed by many wavefronts. These characteristics ensure that the table is quickly populated for successful retrievals.
The update (top dotted lines) and lookup (bottom solid lines) mechanisms are also depicted.}

Figure 13: PCSTALL: Microarchitecture of PC-indexed sensitivity table. PC of wavefronts are used to index into a table that stores the information about the sensitivity of the time epoch starting from the PC. The update (top dotted lines) and lookup (bottom solid lines) mechanisms are also depicted.}

workload, wavefront slot #7 with a 40% higher value of relative change in sensitivity would get a value of 1.40. A small subset of epochs from a single CU from each workload was used to determine this calibration value and then applied to all workloads in our evaluations.

Lookup Mechanism. Wavefronts index into a table using some bits of their current PC address to index and retrieve wavefront-level sensitivity of the upcoming time epoch. The individual sensitivities are then summed up to calculate the overall sensitivity of a CU. The CU sensitivity is then used to predict instructions committed at different frequencies. We model the table such that wavefronts index into them one by one at a fixed cycle before the start of a time epoch. Thus, some latency is incurred in the lookup mechanism. This can be improved by further optimizations.

Update Mechanism. After the execution of the time epoch, each wavefront’s sensitivity is calculated using the previously discussed estimation model. The estimated sensitivities are stored into the table for future reference. The update mechanism happens in a non-critical path and has no latency impact on future predictions.

Hardware Design. The PC-based predictor requires one or more tables per GPU to store the sensitivities and are shared by multiple wavefronts as shown in Figure 13. The negligible reduction in accuracy (inferred from Figure 10) when the table is shared across a different number of wavefronts provides flexibility on where the tables are placed. The table is initialized to 0 on boot up and after every epoch, the table is updated with the new values calculated by the estimation model for each wavefront.

Each wavefront needs to be lookup into the table by indexing the starting PC-address. The table is fixed in size and is indexed using bits from the PC-address. For tuning the offset bits, we calculated the relative change in consecutive iterations for different offsets as shown in Figure 11(b). We observe that the relative change starts increasing when the PC-address offset is greater than 4 bits (~ 4 instructions per entry). For tuning the number of entries in PC-table we calculated the hit ratio at different sizes and observed that 128 entries were sufficient for achieving 95%+ hit ratio. Because most workloads involve loops of a few hundred instructions we set the PC table to 128 entries (covering 512 instructions). The offset and the entries define the bits of PC table which will be used to index
respectively. The same PC addresses (0x2A, 0x3B, 0x4C) are then used to index into the table. For our assumptions of 4 bits offset and 128 entries, bits 10-4 of the PC will be used to index into the table.

Table 1 presents the hardware storage overhead of our PC-based predictor per instance. PCSTALL consumes less storage and thus less power compared to the state-of-art CRISP model. The PC table could either be instantiated one per CU or shared among many CUs.

Example Workflow of PCSTALL. Consider three wavefronts (WF0, WF1, WF2) within a CU0 executing instructions starting from PC addresses 0xAAA, 0xBBB, 0xCCD for time epoch t1 (1µs) optimizing for EDP. Also consider that WF0 > WF1 > WF2, in terms of scheduling age. At the beginning of time epoch t1, all wavefronts would use bits 10-4 of their PCs (0x2A, 0x3B, 0x4C) to index into the table and retrieve the sensitivity. The wavefront sensitivities would be summed up to predict the overall sensitivity of CU0 for the time epoch t1. This sensitivity is then utilized to choose a frequency depending on the objective function, for example 1GHz. At the end of time epoch t1, wavefront sensitivity is estimated using counters for total instructions that have been executed by each wavefront as well as the time spent waiting for memory. If the wavefronts executed, 100, 150, and 200 instructions and spent 500ns, 400ns, and 300ns respectively waiting for memory. Then their calculated sensitivities would be 50, 60, and 60 respectively. This is then scaled up relative to the age by multiplying it by the normalized factor for the wavefront slot (1.00, 1.05, 1.10) to result in 50, 63, and 66 respectively. The same PC addresses (0x2A, 0x3B, 0x4C) are then used to index into the table and update these values of sensitivity into the table before executing time epoch t2.

5 Evaluation Methodology

We utilize the publicly available AMD GCN3 based GPU simulation model [68] within the gem5 simulator [69] for evaluating the performance of the system. We assume a 64 CU GPU with 16 L2 banks shared among all CUs. The clock domain of the memory subsystem, along with L2 cache, is fixed at 1.6 GHz for all the evaluations. Clock synchronization across the CUs has been modeled using the clock domain crossing units available in HeteroGarnet [70]. For most of our evaluations, we assume a single clock domain for each CU, unless otherwise specified. Each clock domain is DVFS-enabled [15] with an integrated voltage regulator and frequency modulator capable of transitioning to a frequency between 1.3 GHz – 2.2 GHz at steps of 100 MHz (10 V/f states). The transition latency is assumed to be 4ns for 1µs time epochs, 40ns for 10µs, 200ns for 200µs, and 400ns for 100µs time epochs. At the end of each epoch, the local DVFS manager uses the associated control mechanism to assign an operating frequency for each clock domain for the next epoch subject to an objective function (EDP, EDP, etc.).

Power Model. We utilize an in-house power model based on detailed hardware measurements. The power model takes in performance counter data and estimates energy consumed for any given interval similar to previous works [71]. The model projects into the table. For our assumptions of 4 bits offset and 128 entries, bits 10-4 of the PC will be used to index into the table.

Table 1 presents the hardware storage overhead of our PC-based predictor per instance. PCSTALL consumes less storage and thus less power compared to the state-of-art CRISP model. The PC table could either be instantiated one per CU or shared among many CUs.

| PCSTALL | Sensitivity Table | 128 entries | 128 | 328 |
|---------|------------------|-------------|-----|-----|
| CRISP   |                  |             |     |     |
| CRIT    |                  |             |     |     |
| LEAD    |                  |             |     |     |
| STALL   |                  |             |     |     |

Figure 14: Fork-Pre-Execute Methodology: Simulator process is forked at each time epoch and sample processes are executed allotting a unique shuffled frequency (e.g. f0, f1, and f2) for each clock domain. Performance data is sent to the original process where optimal frequencies are then selected for each clock domain and the time epoch is re-executed.
Average CRISP ORACLE otherwise specified.

Some system-level objective functions include minimizing EDnP designs and then normalizing it over a static run of 1.7 GHz, unless summing up the energy consumed by each workload with different value (minimum for ED2P) is then selected. suitable frequency which enables an optimized objective function has been calculated by normalizing the runtime of all workloads, geometric mean (GeoMean) across the workloads. The GeoMean We present evaluations for individual workloads as well as a objective function (such as ED2P) of our fixed time epochs. A

solution reaches 97.6% accuracy with only 10 processes (one for each frequency state). We use this mechanism to generate near-accurate estimations of any time epoch for a given clock domain.

5.2 Objective Functions
The exact objective function, that DVFS optimizes for, varies depending on the product, use-case, workload, and environment. Some system-level objective functions include minimizing EDnP metrics or maintaining operation within certain power bounds. Our evaluations consider minimizing EDP and ED2P because EDP is often important for battery-constrained environments while ED2P is important for performance-oriented servers. Our DVFS prediction mechanism could easily be extended to other objective functions such as meeting per-job quality-of-service (QoS) deadlines.

Voltage/Frequency Selection. We require both a performance model and an energy model to make decisions about DVFS settings to improve EDP or related measures. The performance model is obtained from the sensitivity value using the PC-index. The energy model we use assumes that the phase of the next interval will be computationally similar to the current interval, which is typically a safe assumption for GPUs as has been shown in several previous works [10, 20]. We use the energy and the inverse of number of instructions committed per time epoch as a proxy for calculating the objective function (such as ED2P) of our fixed time epochs. A suitable frequency which enables an optimized objective function value (minimum for ED2P) is then selected.

We present evaluations for individual workloads as well as a geometric mean (GeoMean) across the workloads. The GeoMean has been calculated by normalizing the runtime of all workloads, summing up the energy consumed by each workload with different designs and then normalizing it over a static run of 1.7 GHz, unless otherwise specified.

5.3 Workloads and Baseline Models
We evaluate our approach using HPC and machine intelligence GPU workloads. Specifically, for HPC applications, we consider the ECP proxy applications [72] and for machine intelligence applications, we evaluate the DeepBench [73] and DNNMark [74] benchmark suites.

We compare our DVFS approach to the baseline models described in Section 2.3, three static frequencies, as well as the oracle discussed earlier in Section 5.1. TABLE II lists all evaluated designs including their estimation models and prediction mechanisms. We include three models that use accurate oracular estimates. The first uses the accurate estimates of the prior time epoch to create a reactive predictor (ACCREAC) and the second uses the accurate estimates to fill in a table for the accurate, but not practical, PC-based predictor (ACCPC). Finally, we directly use the accurate estimates for the next time epochs in the ORACLE model for near-optimal comparison. Our PC-based predictor could potentially be combined with any of the previously described wavefront-level estimation models, but we chose the STALL model because of its relatively simplicity.

5.4 Hierarchical Power Management
The hardware based DVFS system described in this paper has been designed with a commercial hierarchical power management system in mind. Within such a scheme, higher-level power management policies set power objectives at millisecond scales, which then impact the internal frequency range used by the hardware DVFS controller. The CUs may also need to coordinate with a global agent to ensure power delivery integrity is not compromised [43, 75]. For our evaluations, we chose a small range of frequencies (1.3GHz-2.2GHz) to simulate the power constraint set by a higher-level power manager above.

6 Evaluations
Measuring the advantages achieved by our DVFS prediction approach involves assessing the accuracy of the predictor as well as the overall power efficiency improvement. In this section, we first evaluate the accuracy of the predictor by comparing it to the oracle. Then we present results optimizing for minimal EDnP and EDP for individual workloads as well as the overall GeoMean.

6.1 Prediction Accuracy
We calculate prediction accuracy by comparing the number of predicted instructions committed to the number of actual instructions committed. Note that the prediction accuracy is a power-model-agnostic metric and only focuses on the prediction algorithm itself. Figure 15 shows the prediction accuracy of different models compared to an oracle reported sensitivity at 1µs. The hardware based DVFS system described in this paper has been designed with a commercial hierarchical power management system in mind. Within such a scheme, higher-level power management policies set power objectives at millisecond scales, which then impact the internal frequency range used by the hardware DVFS controller. The CUs may also need to coordinate with a global agent to ensure power delivery integrity is not compromised [43, 75]. For our evaluations, we chose a small range of frequencies (1.3GHz-2.2GHz) to simulate the power constraint set by a higher-level power manager above.

We observe that complex models such as CRIT and CRISP outperform simpler models such as STALL and LEAD yet still have a relatively low prediction accuracy of ~60%. Most of this

TABLE II. DVFS PREDICTION DESIGNS EVALUATED

| Name   | Estimation Model          | Control Mechanism |
|--------|---------------------------|-------------------|
| STALL  | Stall Model [24]          | Reactive          |
| LEAD   | Leading Load [24, 33, 34] | Reactive          |
| CRIT   | Critical Path [10]        | Reactive          |
| CRISP  | CRISP GPU Model [20]      | Reactive          |
| ACCREAC| Accurate Estimate         | Reactive          |
| PCSTALL| Stall – Wavefront         | PC-Based          |
| ACCPC  | Accurate Estimate         | PC-Based          |
| ORACLE | Accurate Estimate         | Oracle            |

Figure 15: Accuracy of prediction models compared to an ORACLE prediction (100% accurate) at 1µs time epochs. A practical implementation of a PC-based predictor (PCSTALL) outperforms even perfect reactive model (ACCREAC) in most models.
inaccuracy is due to the reactive nature of prior approaches. Even with an accurate sensitivity estimation, a reactive model (ACCREAC) is only able to reach 63% accuracy in prediction on an average. On the other hand, a PC predictor has the potential (ACCPC) of reaching 90% prediction accuracy. Even with a practical sensitivity estimation model, our PC-predictor (PCSTALL) can deliver up to 81% prediction accuracy, outperforming the accurately estimating reactive model (ACCREAC). The PC-based predictors show high improvements for memory-bound applications such as xsbench and FwdPool. These results show that high prediction accuracies can be achieved even with a simple prediction mechanism. These prediction accuracies directly translate into better frequency selection and result in improved power efficiencies as shown in the next subsections.

Figure 1(b) shows the trend of accuracy with different time epoch durations. Although PCSTALL delivers the best accuracies at finer time epochs, PCSTALL also results in improvement at longer time epochs. Our results show that PCSTALL can be beneficial even at 50µs time epochs, showing that our policy can be adopted in current generation hardware as well.

6.2 Minimizing ED²P

Next, we evaluate how the different models minimize overall ED²P. Figure 16 shows the ED²P values normalized to a static 1.7 GHz operation. The ORACLE improves power efficiency by up to 54%, whereas the reactive models provide just a 34% potential improvement with CRISP delivering 23% improvement in ED²P compared to 1.7 GHz operation. On the other hand, our PC-based predictive model showcases potential savings in ED²P of 48% while ACCPC can reach up to 51% improvement.

Figure 17 shows the trend of improvement in ED²P when the DVFS time epoch is reduced from 100 µs to 1 µs. We see that the advantage of PCSTALL improves as we approach finer time epochs. Even at a 50µs time epoch, PCSTALL delivers a 36% improvement compared to a 21% improvement by CRISP. However, for some workloads such as comd, hpgmg, pennant, BwdPool, FwdSoft CRISP performs better than PCSTALL at coarse time epochs such as 100µs. Further, a small subset of workloads such as hpgmg and lulesh do not see much benefit in going to the finer grain time scales offering only 10% improvement in ED²P. However, for other workloads the benefits outweigh the overhead of performing DVFS at finer time scales.

Figure 18 shows the improvement in normalized energy and delay from CRISP → PCSTALL (black border) for all the workloads when minimizing ED²P at 1µs. Note the x-axis and y-axis limits. Values have been normalized to Static-1.7 GHz operation. The data point at (1,1) reflects those normalized points.

Figure 16: ED²P of workloads normalized to a static 1.7GHz execution at 1µs time epochs.

Figure 17: ED²P of workloads comparing CRISP and PCSTALL against ORACLE designs at coarser time epochs (100µs, 50µs 10µs).

Figure 18: Scatter plot showing the improvement in normalized energy and delay from CRISP → PCSTALL (black border) for all the workloads when minimizing ED²P at 1µs. Note the x-axis and y-axis limits. Values have been normalized to Static-1.7 GHz operation. The data point at (1,1) reflects those normalized points.
These results reinforce the intuition that a future-looking EDP-based optimization will lead to more improvement in energy reduction in latency compared to the reduction in energy. Some workloads, such as CompM and hacc, improve performance by a large magnitude. The drop is more in the vertical direction for most workloads because of the higher weightage given to the delay value. EDP-based optimization will lead to more improvement in energy as well as some of the results will be shown in the next paragraphs.

Frequency Time Share. Figure 19 shows the percentage of time CUs spend at each frequency state using the PCSTALL mechanism for EDP optimization. As expected, the CUs frequently select the higher operating frequencies for the compute-intensive applications (like dgemm and hacc), while the CUs mostly stay within the lower frequency ranges for the memory-intensive applications. (like hpgmg and xsbench). Workload dgemm also has a highly heterogeneous behavior, leading to comparatively lower accuracies. BwdPool shows an interesting behavior where it adopts a single frequency (1.5 GHz) during its training period. This is mainly because of the constant rate of instruction executed by the CUs during its execution. FwdSoft behaves in a special way where a static operation at 1.7 GHz is beneficial over both 1.3GHz and 2.2GHz. Our detailed analysis showed that this is because of a second-order effect where higher frequency operation of multiple CUs leads to thrashing at L2 cache. This leads to performance degradation at 2.2 GHz.

Figure 19: Average time share of each frequency state in CUs while optimizing for EDP using PCSTALL at 1µs.

Figure 20: Geomean EDP of workloads normalized to a static 1.7 GHz execution at different time epoch durations.

6.3 Minimizing EDP

Figure 20 shows the trend in EDP improvement delivered by PCSTALL. The results show a similar trend as EDP. While PCSTALL is able to take advantage of the finer-grain time epochs, reactive policies fail to show a considerable advantage. However, the difference in reactive and predictive policies is lower compared to EDP optimizing executions.

6.4 Fixed-Performance Energy Savings

In addition to the EDP and EDP objective functions, we studied the overall energy savings within different performance degradation limits. Figure 21(a) shows the energy savings with 5% and 10% performance degradation limits. We see that PCSTALL can deliver up to 9.6% energy savings when restricted to 5% performance degradation compared to 2.1% savings from CRISP. The difference between PCSTALL and CRISP increases when the degradation limit is increased to 10% where PCSTALL results in 19.86% energy savings compared to a 4.7% energy saving from CRISP.

6.5 Scalability Study

In addition to the single CU clock domains, we study the scalability of PCSTALL for larger clock domains. Figure 21(b) shows the normalized EDP improvement delivered by PCSTALL at different clock domain granularity in comparison to CRISP and ORACLE. A 1CU clock domain granularity means that a single table (with 128 entries in our evaluation) is only shared by 40 workfronts (40 wavefronthash/4CU), 2CU clock domain would mean that a table (with the same 128 entries) is shared by 80 wavefronts, 40 each from 2 CUs.

In general, as the clock domain granularity increases, the opportunity in reducing the EDP via DVFS decreases because of the decrease in opportunities to customize V/f state for the instructions being executed. In comparison to ORACLE, CRISP does not extract much ED P improvement, resulting in only a 4% improvement over a static operation at 32CU granularity. On the other hand, PCSTALL delivers high improvement in EDP even at the 32CU clock domain granularity reaching 18% improvement compared to 24% improvement possible with ORACLE.

This shows that PCSTALL could be beneficial not just for smaller clock domains but for larger domains as well. The scalable nature of PCSTALL enables high power efficiency gains leveraging the continuous improvement in circuit technologies.
7 Conclusion

In this paper, we presented prediction-based techniques to unlock the fine-grain DVFS opportunities provided by the improving IVR technology. Our key observation was, at fine-grain time scales, GPU workloads exhibit high variation in sensitivity with respect to the operating clock frequency. Further, we observed that existing performance estimation models work more accurately at a wavefront level than at a CU or GPU level. Combining these observations, we advocated for a predictive model more suited to rapidly adjust to highly varying workload behavior and designed a scalable wavefront-level PC-based predictor. Our analysis showed that our prediction-based mechanism significantly outperforms previously proposed reactive techniques and comes close to achieving oracular prediction accuracies resulting in high energy efficiencies.

Acknowledgements

The authors would like to thank several reviewers for their constructive feedback and suggestions for this paper. We also thank Gabriel Loh, Tushar Krishna, and Karen Prairie for their feedback on improving early drafts of this work. The authors would like to thank several reviewers for their constructive feedback and suggestions for this paper. We also thank Gabriel Loh, Tushar Krishna, and Karen Prairie for their feedback on improving early drafts of this work. We also thank Gabriel Loh, Tushar Krishna, and Karen Prairie for their feedback on improving early drafts of this work.

References

[1] S. Kaxiras and M. Martinson, “Computer Architecture Techniques for Power-Efficiency,” Synthesis Lectures on Computer Architecture, vol. 3, no. 1, pp. 1-207, 2008.
[2] S. Eyerman and L. Eckhout, “Fine-grained DVFS using on-chip regulators,” ACM Transactions on Architecture and Code Optimization, vol. 8, no. 1, pp. 1-24, 2011.
[3] W. Kim, M. S. Gupta, G. Y. Wei and D. M. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," 2008. [Online]. Available: http://eecs.harvard.edu/~dbrooks/kin2008_hpca.pdf. [Accessed 25 6 2019].
[4] Y. Bai, V. W. Lee and E. Ipek, "Voltage Regulator Efficiency Aware Power Management," Sigplan Notices, vol. 45, no. 1, pp. 825-838, 2017.
[5] A. Paul, S. P. Park, S. Dinesh, Y. M. Kim, N. Borkar, U. R. Karpuzcu and C. H. Kim, "System-Level Power Analysis of a Multicore Multiprocessor Processor With On-Chip Voltage Regulators," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, pp. 3468-3476, 2016.
[6] E. A. Burton, G. Schrom, F. Paillet, J. P. Douglas, W. J. Lambert, K. Radhakrishnan and M. J. Hill, "FIVR — Fully integrated voltage regulators on 4th generation Intel® Core™ SoCs," 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6803344. [Accessed 21 11 2019].
[7] M. Fojtik, B. Keller, A. Klimek, B. Linckert, P. Nkpinney, S. G. Tell, B. Zimmer, T. Raja, K. Zhou, W. J. Dally and B. Kailhany, "A Fine-Grained GALS SoC with Pausible Adaptive Clocking in 16 nm FinFET," in 2019 25th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2019.
[8] X. Sun, A. Boora, W. Zhang, V. R. Pamula and V. Sathe, "14.5 A 0.6-to-1.1V Computationally Regulated Digital LDO with 2.79-Cycle Mean Settling Time and Autonomous Runtime Gain Tracking in 65nm CMOS," IEEE International Solid-State Circuits Conference (ISSCC), 2019.
[9] Y. Okuma, K. Ishida, Y. Ryo, X. Zhang, P. H. Chen, K. Watanebe, M. Takamiya and T. Sakurai, "0.5-V input digital LDO with 98.7% current efficiency and 2.7-µA quiescent current in 65nm CMOS," 2010. [Online]. Available: http://lowpower.iiis.u-tokyo.ac.jp/paper/2010_22.pdf. [Accessed 25 11 2019].
[10] R. Mitakahudinov, E. Ebrahimim and Y. N. Patt, "Predicting Performance Impact of DVFS for Realistic Memory Systems," IEEE Micro, pp. 155-165, 2012.
[11] P. Patermoster, A. Maki, A. Hernandez, M. Grossman, M. Lau, D. Sutherland and A. Mathad, "XBOX Series X: A Next-Generation Gaming Console SoC," in 2021 IEEE International Solid-State Circuits Conference (ISSCC), 2021.
[12] P. A. Meinerzhagen, C. Tokunaga, A. Malavasi, V. Vaidya, A. Mendon, D. A. Mathaiukity, J. P. Kulkarni, C. Augustine, M. Cho, S. T. Kim, G. E. Mathew, R. Jain, J. F. Ryan, C.-C. Peng, S. Paul, S. R. Vangal, B. P. Esperzar, L. Cuelhar, M. Woodman, B. Iyer, S. Mayuran, G. N. Chinya, C. Zou, Y. Liao, K. Ravichandran, H. Wang, M. M. Khellah, J. W. Tschanz and V. De, "An energy-efficient graphics processor featuring fine-grain DVFS with integrated voltage regulators, execution-unit turbo, and retentive sleep in 14nm tri-gate CMOS," 2018. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8310172. [Accessed 16 8 2019].
[13] T. Singh, A. Schufer, S. Rangarajan, D. John, C. Hentzen, R. Schreiber, M. Rodriguez, S. Kosonocky, S. D. Nafligzer and A. Novak, "Zen: An Energy-Efficient High-Performance X86 Core," IEEE Journal of Solid-state Circuits, vol. 53, no. 1, pp. 102-114, 2018.
[14] T. Burd, N. Beck, S. White, M. Paraschou, N. Kalyanansundaram, G. Donley, A. D. Smith, L. D. Hewitt and S. D. Nafligzer, “Zeppelin” An SoC for Multichip Architectures,” IEEE Journal of Solid-state Circuits, vol. 54, no. 1, pp. 133-143, 2019.
[15] D. Bouvier, J. Gibney, A. Banerjee and S. Arora, "AMD Raven-Ridge APU: Delivering a new level of visual performance in an SoC," [Online]. Available: https://www.hotchips.org/hc50/Icon1 Ef05_AMD_APU_AMD_Raven_HotCh ips30_Final.pdf. [Accessed 22 2 2020].
[16] B. Zimmer, Y. Lee, A. Paggelli, J. Kwak, R. Jevtic, B. Keller, S. Bailey, M. Bilogujevic, P.-F. Chiu, H.-P. Le, P.-H. Chen, N. Saturaja, R. Avizienis, A. Waterman, B. Richards, P. Flatresse, E. Alon, K. Asanovic and B. Nikolic, "A RISC-V vector processor with tightly-integrated switched-capacitor DC-DC converters in 28nm FDSOI," 2015. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7231305. [Accessed 12 2 2020].
[17] R. Muthukuruppan, T. Mahajan, H. Krishnamurthy, S. Mangal, A. Dhanshekar, R. Ghayal and V. De, "A digitally controlled linear regulator for per-core wide-range DVFS of atom™ cores in 14nm tri-gate CMOS featuring non-linear control, adaptive gain and code mapping," in ESSCIRC 2017 - 43rd IEEE European Solid State Circuits Conference, Leuven, 2017.
[18] S. J. Kim, D. Kim, H. Ham, J. Kim and M. Seok, “A 67.1-ps FOM, 0.5-V Hybrid Digital LDO With Asynchronous Feedforward Control Via Slope Detection and Synchronous PI With State-Synchronous PI With State-Shift Switching,” 2018. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8490730. [Accessed 12 2 2020].
[19] H. Li, X. Wang, J. Xu, Z. Wang, R. K. V. Maeda, Z. Wang, P. Yang, L. H. K. Duong and Z. W. Wang, ”Energy-Efficient Power Delivery System Paradigms for Many-Core Processors,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 3, pp. 449-462, 2017.
[20] R. Nath and D. M. Tullsen, "The CRISP performance model for dynamic voltage and frequency scaling in a GPGPU," in 2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2015.
[21] X. Mei, L. S. Yung, K. Zhao and X. Chu, "A measurement study of GPU DVFS on energy conservation," 2013. [Online]. Available: http://comp.hkbu.edu.hk/~chxw/papers/hotpower_2013.pdf. [Accessed 15 8 2019].
[22] J. a. B. A. Leng, R. Bertran, P. Bose, Y. Zu and V. J. Reddi, "Predictive Guardbanding: Program-Driven Timing Margin Reduction for GPUs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 1, pp. 171-184, 2021.
[23] S. T. Kim, Y.-C. Shih, K. Mazumdar, R. Jain, J. F. Ryan, C. Tokunaga, C. Augustine, J. P. Kulkarni, K. Ravichandran, J. W. Tschanz, M. M. Khellah and
V. De, "Enabling Wide Autonomous DVFS in a 22 nm Graphics Execution Core Using a Digitally Controlled Fully Integrated Voltage Regulator," IEEE Journal of Solid-state Circuits, vol. 51, no. 1, pp. 18-30, 2016.

[24] G. Keramidas, V. Spiliopoulos and S. Kaxiras, "Interval-based models for runtime DVFS orchestration in superscalar processors," 2010. [Online]. Available: http://pages.cs.wisc.edu/~kaxiras/papers/cl10_dvfs_model.pdf. [Accessed 25 6 2019].

[25] C. Isci, G. Contreras and M. Martonos, Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management, 2006, p. 359-370.

[26] K. Straube, J. Lowe-Power, C. Nitta, M. Farrens and V. Akella, "Improving Provisioned Power Efficiency in HPC Systems with GPU-CAPP," 2018 IEEE 25th International Conference on High Performance Computing (HPC), 2018.

[27] C. C. Nugteren, G. G. J. v. d. Braak and H. Corporaal, "Roofline-aware DVFS for GPUs," 2014. [Online]. Available: http://ece.neu.edu/groups/nucar/nucartalks/roofline-aware_dvfs_for_gpus.pdf. [Accessed 15 8 2019].

[28] A. Mishra and N. Khare, "Analysis of DVFS Techniques for Improving the GPU Energy Efficiency," 2015. [Online]. Available: http://file.sciprint.org/pdf/ojece.2015121415504865.pdf. [Accessed 15 8 2019].

[29] Z. Tang, Y. Wang, Q. Wang and X. Chu, "The Impact of GPU DVFS on the Energy and Performance of Deep Learning: An Empirical Study," in Proceedings of the Tenth ACM International Conference on Future Energy Systems, Phoenix, 2019.

[30] G. Antoniou, H. Volos, D. B. Bartolini, T. Rollet, Y. Sazeides and J. H. Yahya, "AgilePkgC: An Agile System Idle State Architecture for Energy Proportional Datacenter Servers," arXiv, 2022.

[31] B. Dutta, V. Adhinarayan and W.-c. Feng, "GPU power prediction via ensemble machine learning for DVFS space exploration," 2018. [Online]. Available: https://vtechworks.lib.vt.edu/handle/10919/81997. [Accessed 15 8 2019].

[32] K. Fan, B. Cosenza and B. Juurlink, "Predictable GPUs Frequency Scaling for Energy and Performance," in Proceedings of the 48th International Conference on Parallel Processing, Kyoto, Japan, 2019.

[33] S. Eyerman and L. Eeckhout, "A Counter Architecture for Online DVFS Profitability Estimation," IEEE Transactions on Computers, vol. 59, no. 11, pp. 1576-1583, 2010.

[34] B. Rountree, D. K. Lowenthal, M. Schulz and B. R. d. Supinski, "Practical performance prediction under Dynamic Voltage Frequency Scaling," 2011. [Online]. Available: https://yadda.icm.edu.pl/yadda/element/bwmeta1.element.iiiae-000006008533. [Accessed 25 6 2019].

[35] C. Isci, A. Buyuktosunoglu, C.-Y. Chen, P. Bose and M. Martonosi, "An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget," IEEE Micro, pp. 347-358, 2006.

[36] H. Hoffmann and M. Maggio, "PCP: A Generalized Approach to Optimizing Performance Under Power Constraints through Resource Management," 2014. [Online]. Available: https://usenix.org/system/files/conference/ics14-icac14-paper-hoffmann.pdf. [Accessed 23 7 2019].

[37] S. Park, J. Park, D. Shin, Y. Wang, Q. Xie, M. Pedram and N. Chang, "Accurate Modeling of the Delay and Energy Overhead of Dynamic Voltage and Frequency Scaling in Modern Microprocessors," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 32, no. 5, pp. 695-708, 2013.

[38] V. Spiliopoulos, S. Kaxiras and G. Keramidas, "Green governors: A framework for Continuously Adaptive DVFS," 2011. [Online]. Available: http://diva-portal.org/smash/record.jsf?pid=diva2:474791. [Accessed 25 6 2019].

[39] M. Weiser, B. B. Welch, A. J. Demers and S. Shenker, "Scheduling for reduced CPU energy," 1994. [Online]. Available: https://link.springer.com/chapter/10.1007/978-0-585-29603-6_17. [Accessed 25 6 2019].

[40] P. Zou, A. Li, K. Barker and R. Ge, "Inductor-Directed Dynamic Power Management for Iterative Workloads on GPU-Accelerated Systems," in 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), 2020.

[41] A. Zou, J. Leng, Y. Xu, T. Tong, V. J. Reddi, D. M. Brooks, G.-Y. Wei and X. Zhang, "Tory: Early-Stage Design Space Exploration Tool for Integrated Voltage Regulators," 2017. [Online]. Available: https://dl.acm.org/citation.cfm?id=3602268. [Accessed 23 7 2019].

[42] A. Paul, S. P. Park, D. Sonasekhar, Y. M. Kim, N. Bokar, U. R. Karpuecu and C. H. Kim, "System-Level Power Analysis of a Multicon: Multipower Domain Processor With ON-Chip Voltage Regulators," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2016.

[43] J. Haj-Yahya, M. Alser, J. S. Kim, L. Orosa, E. Rotem, A. Mendelson, A. Chattopadhyay and O. Mutlu, "FlexWatts: A Power- and Workload-Aware Hybrid Power Delivery Network for Energy-Efficient Microprocessors," arXiv, 2020.

[44] M. Sankarasubramanian, K. Radhakrishnan, Y. Min, W. Lambert, M. J. Hill, R. Dani, R. Mesch, L. Wojewoda, J. Chavarria and A. Augustine, "Magnetic Inductor Arrays for Intel® Fully Integrated Voltage Regulator (FIVR) on 10th generation Intel® Core™ SoCs," in 2020 IEEE 70th Electronic Components and Technology Conference (ECTC), Orlando, 2020.

[45] C. Schaef, K. Radhakrishnan, K. Ravichandran, J. W. Tschanz, V. De, N. Desai, H. K. Krishnamurthy, X. Liu, K. Z. Ahmed, S. Kim, S. Weng, H. T. Do and W. J. Lambert, "A Light-Load Efficient Fully Integrated Voltage Regulator in 14-nm CMOS With 2.5-nH Package-Embedded Air-Core Inductors," IEEE Journal of Solid-state Circuits, vol. 54, no. 12, pp. 3316-3325, 2019.

[46] G. Yan, Y. Li, Y. Han, X. Li, M. Guo and X. Liang, "AgileRegulator: A hybrid voltage regulator scheme redeeming dark silicon for power efficiency in a multicore architecture," 2012. [Online]. Available: http://cs.sjtu.edu.cn/~guomy/pdf/conferences/c130.pdf. [Accessed 5 7 2019].

[47] Q. Deng, D. Meister, A. Bhattacharjee, T. F. Wensisch and R. Bianchini, "CoScale: Coordinating CPU and Memory System DVFS in Server Systems," IEEE Micro, pp. 143-154, 2012.

[48] S. Akram, J. B. Sartor and L. Eeckhout, "DVFS performance prediction for managed multithreaded applications," 2016. [Online]. Available: https://biblio.ugent.be/publication/7245653. [Accessed 23 7 2019].

[49] H. Zhang and H. Hoffmann, "Maximizing Performance Under a Power Cap: A Comparison of Hardware, Software, and Hybrid Techniques," Sigplan Notices, vol. 44, no. 2, pp. 545-559, 2016.

[50] H. Hoffmann and M. Maggio, "PCP: A Generalized Approach to Optimizing Performance Under Power Constraints through Resource Management," 2014. [Online]. Available: https://hug.hhi.de/search/publication/44544051-23aa-4beb2-904e-b780181c3910. [Accessed 5 7 2019].

[51] C. C. Lin, C. J. Chang, Y. C. Syu, J. J. Wu, P. Liu, P. W. Cheng and W. T. Hsu, "An Energy-Efficient Task Scheduler for Multi-core Platforms with Per-core DVFS Based on Task Characteristics," 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6957247. [Accessed 5 7 2019].

[52] M. Curtis-Maury, D. Dziwera, C. D. Antonopoulos and D. S. Nikolopoulos, "Online power-performance adaptation of multithreaded programs using hardware event-based prediction," 2006. [Online]. Available: http://people.cs.vt.edu/~dsn/papers/ics06.pdf. [Accessed 5 7 2019].

[53] B. Su, J. Gu, L. Shen, W. Huang, J. L. Greathouse and Z. Wang, "PPEP: Online Performance, Power, and Energy Prediction Framework and DVFS Space Exploration," IEEE Micro, pp. 445-457, 2014.

[54] A. Gendler, E. Knoll and Y. Sazeides, "i-DVFS: Instantaneous Frequency Switch During Dynamic Voltage and Frequency Scaling," in IEEE Micro, 2021.

[55] J. Haj-Yahya, M. Alser, J. Kim, A. G. Yaglıkçı, N. Vijaykumar, E. Rotem and O. Mutlu, "SysScale: Exploiting Multi-domain Dynamic Voltage and Frequency Scaling for Energy Efficient Mobile Processors," in Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture, Virtual Event, 2020.

[56] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony and R. Rajkumar, "Critical Power Slope: Understanding the Runtime Effects of Frequency
Scaling," in Proceedings of the 16th International Conference on Supercomputing, New York, New York, USA, 2002.

[57] Q. Wu, M. Martonosi, D. W. Clark, V. J. Reddi, D. A. Connors, Y. Wu, J. Lee and D. M. Brooks, "A Dynamic Compilation Framework for Controlling Microprocessor Energy and Performance," IEEE Micro, pp. 271-282, 2005.

[58] Y. Saezides, R. Kumar, D. Tullsen and T. Constantinou, "The Danger of Interval-Based Power Efficiency Metrics: When Worst Is Best," in IEEE Computer Architecture Letters, 2005.

[59] S. Akram, J. B. Sartor and L. Eckhout, "DEP-BURST: Online DVFS Performance Prediction for Energy-Efficient Managed Language Execution," IEEE Transactions on Computers, vol. 66, no. 4, pp. 601-615, 1 April 2017.

[60] C. Isci, A. Buyuktosunoglu and M. Martonosi, "Long-term workload phases: duration predictions and applications to DVFS," IEEE Micro, vol. 25, no. 5, pp. 39-51, 2005.

[61] W. L. Birtcher and L. K. John, "Predictive power management for multi-core processors," , 2010. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-24322-6_21. [Accessed 21 11 2019].

[62] A. Smith and N. James, "AMD Instinct™ MI200 Series Accelerator and Node Architectures," in 2022 IEEE Hot Chips 34 Symposium (HCS), 2022.

[63] R. Thomas, N. Sedaghati and R. Teodorescu, "EmerGPU: Understanding and mitigating resonance-induced voltage noise in GPU architectures," 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7482076. [Accessed 24 11 2021].

[64] C. Isci and M. Martonosi, "Phase characterization for power: evaluating control-flow-based and event-counter-based techniques," 2006. [Online]. Available: http://parapet.ee.princeton.edu/papers/canturk-bpc2006.pdf. [Accessed 15 8 2019].

[65] O. Khan and S. Kundu, "Microvisor: a runtime architecture for thermal management in chip multiprocessors," 2011. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-24568-8_5. [Accessed 20 11 2019].

[66] S. Srinivasan, R. Kumar and S. Kundu, "Program phase duration prediction and its application to fine-grain power management," 2013. [Online]. Available: https://doi.org/10.1109/isvlsi.2013.6654634. [Accessed 20 11 2019].

[67] F. Xie, M. Martonosi and S. Malik, "Compile-time dynamic voltage scaling settings: opportunities and limits," Sigplan Notices, vol. 38, no. 5, pp. 49-62, 2003.

[68] A. Gutierrez, B. M. Beckmann, A. Dutu, J. Gross, M. LeBeau, J. Kalamatianos, O. Kayiran, M. Poremba, B. Potter, S. Puthoor, M. D. Sinclair, M. Wyse, J. Yin, X. Zhang, A. Jain and T. G. Rogers, "Lost in Abstraction: Pitfalls of Analyzing GPUs at the Intermediate Language Level," 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8327041. [Accessed 30 7 2019].

[69] N. L. Binkert, B. M. Beckmann, G. Black, S. K. Reinhardt, A. G. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill and D. A. Wood, "The gem5 simulator," ACM Sigarch Computer Architecture News, vol. 39, no. 2, pp. 1-7, 2011.

[70] S. Bharadwaj, J. Yin, B. Beckmann and T. Krishna, "Kite: A Family of Heterogeneous Interposer Topologies Enabled via Accurate Interconnect Modeling," in 57th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, US, 2020.

[71] J. L. Greathouse and G. H. Loh, "Machine learning for performance and power modeling of heterogeneous systems," In Proceedings of the International Conference on Computer-Aided Design (ICCAD ’18). ACM, New York, NY, USA, Article 47, 6 pages, 2018.

[72] I. Karlin, J. Keasler and J. R. Neely, "LULESH 2.0 Updates and Changes," 2013. [Online]. Available: https://codesign.llnl.gov/pdfs/lulesh2.0_changes.pdf. [Accessed 2 8 2019].

[73] BAIDU Research, "DeepBench," [Online]. Available: https://github.com/baidu-research/DeepBench. [Accessed 2 8 2019].

[74] S. Dong and D. R. Kaeli, DNNMark: A Deep Neural Network Benchmark Suite for GPUs," 2017. [Online]. Available: https://dl.acm.org/citation.cfm?id=3038239. [Accessed 2 8 2019].

[75] P. Shah, R. G. Shenoy, V. Srinivasan, P. Bose and A. Buyuktosunoglu, "TokenSmart: Distributed, Scalable Power Management in the Many-Core Era," in IEEE Computer Architecture Letters, 2021.

[76] C. Isci, G. Contreras and M. Martonosi, "Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management," IEEE Micro, pp. 359-370, 2006.

[77] K. Choi, R. Soma and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance trade-off based on the ratio of off-chip access to on-chip computation times," 2004. [Online]. Available: http://apoftab.usc.edu/~khwan/g-dvfs.pdf. [Accessed 5 7 2019].

[78] V. Spiliopoulos, A. Bagdia, A. Hansson, P. Aldworth and S. Kaxiras, "Introducing DVFS-Management in a Full-System Simulator," 2013. [Online]. Available: http://it.ue.se/katalog/vassp447/gem5_dvfs.pdf. [Accessed 30 7 2019].

[79] Q. Wu, V. J. Reddi, Y. Wu, J. Lee, D. Connors, D. Brooks, M. Martonosi, D. W. Clark and yes, A Dynamic Compilation Framework for Controlling Microprocessor Energy and Performance, 2005, p. 271–282.

[80] Z. Topnak-Deniz, M. A. Sperling, J. F. Bulzacchelli, G. S. Still, R. Kruse, S. Kim, D. W. Boerstler, T. Glocokler, R. Robertazzi, K. Stawiasz, T. Diemoz, G. English, D. T. Hui, P. H. Maenchen and J. Friedrich, "5.2 Distributed system of digitally controlled microregulators enabling per-core DVFS for the POWER8 TM microprocessor," 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6757354. [Accessed 25 11 2019].

[81] H. Li, J. Xu, Z. Wang, R. K. V. Maeda, P. Yang and Z. Tian, "Workload-Aware Adaptive Power Delivery System Management for Many-Core Processors," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 10, pp. 2076-2086, 2018.

[82] A. Jog, O. Kayiran, A. Pattnaik, M. Kandemir, O. Mutlu, R. Iyer and C. R. Das, "Exploiting Core Criticality for Enhanced GPU Performance," Sigmetrics Performance Evaluation Review, vol. 44, no. 1, pp. 351-363, 2016.

[83] AMD, "Polaris Whitepaper".

[84] L. Wang, M. Jahre, A. Adileho and L. Eckhout, "MDM: The GPU Memory Divergence Model," in International Symposium on Microarchitecture(MICRO), 2020.