GPU NTC Process Variation Compensation with Voltage Stacking

Rafael Trapani Possignolo, Student Member, IEEE Elnaz Ebrahimi, Ehsan Khish Ardestani, Alamelu Sankaranarayanan, Jose Luis Briz, Member, IEEE, Jose Renau

Abstract—Near Threshold Computing (NTC) has the potential to significantly improve efficiency in high throughput architectures like GPGPUs. Nevertheless, NTC is more sensitive to process variation (PV) as it complicates power delivery. We propose GPU Stacking, a novel method based on voltage stacking, to manage the effects of PV and improve the power delivery simultaneously. To evaluate our methodology, we first explore the design space of GPGPUs in the NTC to find a suitable baseline configuration and then apply GPU Stacking to mitigate the effects of PV. When comparing with an equivalent NTC GPGPU without process variation management, we achieve 37% more performance on average. When considering high production volume, our approach shifts all the chips closer to the nominal non-process variation case, delivering on average (across chips) $\approx$ 80% of the performance of nominal NTC GPGPU, whereas when not using our technique, chips would have $\approx$ 50% of the nominal performance. We also show that our approach can be applied on top of multi-frequency domain designs, improving the overall performance.

Index Terms—Process Variation, Voltage Stacking, Near-Threshold Computing, GPGPUs

I. Introduction

Near Threshold Computing (NTC) is a circuit design technique used to reduce the power envelop used by a design and thus improve energy efficiency by reducing the operating voltage to near the threshold voltage of the transistors employed [1]. Although more energy-efficient, NTC devices usually do not present as high performance as their non-near threshold counterparts. However, it has been shown that the performance impact resulting from NTC can be mitigated through parallelism. An ideal candidate for such operation is a GPU [1], [2], [3]. Nonetheless, NTC makes the system more sensitive to process variation [4].

To manage the additional sensitivity to process variation (PV) introduced by NTC, some researchers have proposed frequency scaling [5] or having multiple voltage domains [6]. Having multiple voltage domains requires additional power rails which will further exacerbate the current delivery problem. Multi-frequency domains [7] have been shown to moderately mitigate the problem of PV effects on energy and performance in many-core NTC designs.

Voltage stacking improves the efficiency of power delivery [8]. When n units are stacked, they are placed in a series fashion, rather than the conventional parallel scheme. Thus, the current in the power delivery network is reduced by a factor of n in a system. This allows voltage regulators (VRs) with increased efficiency, smaller areas and fewer package pins dedicated to power [9]. Voltage stacking has been applied to CPUs [8], GPUs [10], and SRAMs [11].

However, in this work we look at voltage stacking from a different perspective. We note that voltage stacking can compensate for PV effects. The proposed GPU Stacking methodology lets the voltage node between the stacked elements (V_{MID}) float[1]. This floating node is the key to PV compensation. GPU Stacking alleviates the current delivery challenges, and intrinsically mitigates PV effects without requiring multiple voltage domains. GPU Stacking automatically creates a voltage domain per level in the stack without the cost of multiple power rails. We build on top of this premise, and discuss how it can be leveraged for managing PV.

Voltage stacking of many cores has its own challenges, among which is the load mismatch between the stacked cores [9]. Cores go through different phases while running applications, which can result in transient impedance mismatch of the stacked cores, and yield timing failures. As a result, stacking is successful when the cores have a matching workload. GPGPUs are instances of such designs. Not only are the cores identical, but the applications running on them are roughly homogeneous[2].

The evaluation of GPU Stacking is carried out in the near threshold region. Although the use of this method in the near threshold region is proposed, it is not a requirement. The use of NTC in this research study is justified by the increased sensibility of NTC to process variation effects. The first part of this experiment consists of finding the ideal GPU configuration for the NTC region. By carefully sizing the GPU to NTC, power consumption is reduced by 43% with only 4.8% performance degradation compared to the baseline.

Based on our experiments, there is a potential for self balancing in the stacked configuration. We observe that stacking of cores with opposing PV trends is a better

Part of this study has been supported by the National Science Foundation under grants CNS-1059442-003, CNS-1318943-001, CCF-1337278, and CCF-1514284. This work was supported in part by grants TIN2016-76635-C2-1-R (AEI/FEDER, UE), gaZ: T48 research group (Aragón Gov. and European ESF), and HiPEAC4 (European H2020/687698). Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the funding institutions.

R.T. Possignolo, E. Ebrahimi, E.K. Ardestani, A. Sankaranarayanan and J. Renau are with the University of California, Santa Cruz, Santa Cruz, CA, 95064 and J.L. Briz is with the Universidad Zaragoza, Pedro Cerbuna, 12, 50009 Zaragoza, Spain.

[1]For safeguarding reasons, a few voltage regulators are used to cap the maximum and minimum levels of voltage but the voltage does float between maximum and minimum values.

[2]Even though divergence exists in modern GPUs, the amount of load mismatch observed in traditional GPGPUs benchmarks can be handled by our technique, as it will be shown in our evaluation.

choice to gain the best self balancing results. This study proposes the stacking of SIMD Lanes, because having a large number of lanes provides more opportunities for PV compensation. Analyzing several PV maps, GPU Stacking shows that it is able to deliver, on average, $\approx 80\%$ of the nominal performance in a multi-frequency domain scenario (as opposed to $\approx 60\%$ in a non-stacked configuration). This represents a reduction of about 30% in the effects of process variation even after using a multi-frequency domain based compensation of process variation effects. When not considering the multi-frequency domain scenario, the reduction in process variation impact is even more important, around 37% of the process variation effects are compensated. This compensation is mainly due to an uneven voltage distribution that considers the PV of cluster of *lane*s. For a nominal voltage of 0.6V, the observed voltage was in the range of 0.602V to 0.65V for *lane*s negatively affected by PV, and in the range of 0.55V to 0.598V for *lane*s positively affected by PV.

The floating V_{mid} node could be a source of problems in cases of extreme load mismatch, or high temperatures. Thus, we propose the use of Dummy Activity (DA), and a small voltage regulator to keep the voltage within safe operational margins (i.e., to avoid voltage starvation in one of the levels). We use SPICE simulations to verify the reliability of the power delivery network (PDN), and show that GPU Stacking does not incur extra voltage noise. In our simulations, DA and the additional VR were not required, given the stability of the operational voltage.

The main contributions of this paper are:

- The first study to show how voltage stacking alleviates PV effects.
- The first study proposing stacking with an uneven voltage division (V_{MID}).
- A study to propose a technique to make the post-silicon configuration of the design feasible.

The remainder of this paper is organized as follows. In Section II, we discuss related work in a few different topics: voltage stacking, PV mitigation effects, NTC and energy efficiency in GPUs. Then, in Section III, we briefly present background information needed to the understanding of this work. We present the GPU Stacking model in Section IV. Finally, we discuss the experimental setup in Section V and results in Section VI. We wrap-up on Section VII.

II. Related Work

We divide our related work into a few categories. We start discussing prior approaches for voltage staking. Then, we follow presenting other techniques to mitigate PV effects. Finally, we discuss work on NTC and energy efficiency for GPUs.

Voltage Stacking: As supply voltage decreases, the efficiency of power delivery components degrades [9]. On-chip voltage regulators [12] have been proposed to increase the PDN efficiency, as well as the series configurations of units rather than parallel [13], [14], [8]. Such configurations are known as Multi-Story Power Delivery [13], charge recycling [14], or voltage stacked systems [8]. Our proposed technique, GPU Stacking, depends on a series configuration of cores, yet for a different purpose. Note that no previous research on voltage stacking exploits the stacking method to control or neutralize PV effect.

Process Variation mitigation: PV increases as feature size shrinks. And lowering V_{dd}, as a power management technique, further exacerbates the PV effects. Lee et al. study the impact of frequency variation on the throughput of a GPGPU [15]. Adaptive Body Bias (ABB) leverages the power-performance trade-off to manage PV effects [16]. Slower devices due to PV, can run faster by consuming more power, and vice versa. Adaptive Supply Voltage (ASV) is another technique, where supply voltage of a region in the design is adjusted to compensate for performance loss due to variation. ABB and GPU Stacking are orthogonal techniques, and could be used together, although ABB efficiency is expected to reduce with technology scaling [7]. Similarly to ASV, our technique provides a custom supply voltage to each stacking cluster to compensate for process variation.

NTC: An extensive amount of research targets increasing the power/energy efficiency of processors. Apart from the many proposed techniques of how to reduce the utilization of resources [17], [18], or how to promote the use of more power efficient structures [19], [20], there are a significant number of proposals that attempt to utilize the power-performance trade-off. DVFS and Power Gating are among the techniques widely studied at an architectural level, for the same purpose [21], [22]. Intel Turbo Boost technology is another example of utilizing voltage and frequency scaling to adapt to runtime conditions.

Dreslinski et al. [1] study devices for near threshold operation, and Chang et al. propose the optimization of device parameters for NTC [2]. They propose a slightly modified SRAM cell to address the stability challenges introduced in near threshold regions. Lower V_{dd} exacerbates the effects of process variation. Another study of NTC in many cores where they argued in favor of fine-grained core assignment and DVFS [7].

Energy efficiency in GPGPUs: With the rising popularity of GPGPUs, several research groups discuss strategies to make them more energy efficient [23], [24], [25], [26]. Lee et al. study the impact of frequency variation on the throughput of a GPGPU [15]. This methodology, however, is the first to extend the evaluation to NTC trade-offs and it is an extension of previous research addressing NTC challenges. Massive data parallelism and extremely repetitive nature of the GPGPU applications is leveraged to adapt the operational region and configuration to the runtime application demand.

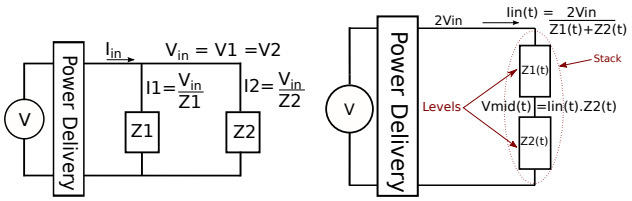
III. Background

This section briefly describes background concepts related to Voltage Stacking, PV and NTC that are needed to better understand this work. We also go over the micro-architecture of a GPU, giving special attention to the details relevant to this paper.

Figure 1 depicts two power delivery schemes, the conventional power delivery scheme (Figure 1-a) has all the elements in parallel. The elements can be interpreted as individual gates, functional units within a core, a whole core, and so forth. The stacked power delivery scheme has elements in series, in the case show in the figure,

the number of stacking levels is two, since there are two elements stacked. In general, for voltage stacking, this is not a requirement, but in this paper we only consider two levels. In a voltage stacked system, the power delivered is the same as in the conventional case, but the delivery voltage is multiplied by n, the number of stack levels, and the current is divided by n, on average [8].

Assuming that the power consumed by each element is the same, the voltage across each element is equal to the nominal V_{dd}. Because roughly half the current flows through the system, the power delivery subsystem could operate more efficiently [9]. However, this is not always the case. When a full core is stacked on top of another, the activity on each core will depend on the program running, and will create a mismatch between the stack levels and make V_{mid} shift from V_{dd}. This problem has been solved in different ways in the literature, for instance by inserting an extra voltage regulator [8], partitioning a core in units that have correlated power consumption [9], or by stacking memory arrays for SRAM application [11].



(a) Conventional (b) Stacked

Fig. 1. Conventional vs. stacked power delivery mode

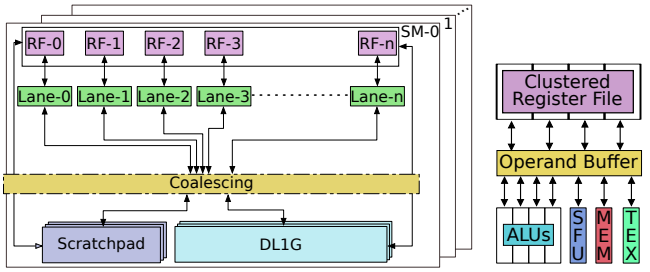
In this work, we use GPGPUs, which contain a large number of identical cores operating in lock-step, running the same program. This provides a very good starting point for voltage stacking. We note that "divergence" can occur and we address divergence later in this paper.

A. General Purpose GPU

In this section, we explore the micro-architecture of a GPU. General purpose computing on graphic processing units (GPGPU) is becoming pervasive as it provides excellent computing power for massively parallel applications. GPGPUs are mainly designed as a cluster of simple processors, depicted in Figure 2. Identical simple processors (*lane*[3]) operate in lock-step inside a stream multiprocessor cluster (SM), running identical threads (though processing different data).

The homogeneous structure of GPGPUs, both in hardware and application, makes them suitable for voltage stacking in order to manage process variation. GPU Stacking stacks *lane*s inside SMs. All the other structures remain in a conventional configuration. The choice of stacking *lane* provides room for more configurability, due to the larger number of *lane*s. It also provides a fine-grained mitigation of process variation, while at the SM level, techniques like multi-clock and multi-voltage domain are possible.

[3]Some authors use the term *lane* to refer to each of the small execution cores within what we call a SIMD lane. We use the same definition as Gebhart et al. [27], and thus count one lane per load/store unit. This leads to a *lane* count 4× smaller than what is advertised for commercial GPUs.



(a) GPGPU (b) SIMD Lanes

Fig. 2. GPGPUs present a large number of identical SIMD lanes, ideal for stacking.

B. Process Variation Model and Effects

So far, we have mentioned PV without specifying how PV can affect a chip. Process variation is the deviation from nominal specifications of a chip due to the difficulty of precisely control the fabrication process of a chip, especially at small feature sizes, e.g., lithographic lens, dopant density fluctuations, and others. Variation can be divided in two categories: systematic and random. From a design point of view, they differ on the granularity at which they occur. Random variation occurs at the transistor level. Systematic variation occurs at a much coarser granularity: within-die (WID) and die-to-die (D2D). Significant variation can be seen on the order of half the chip length [4]. Overall, random variation ends up being averaged out across a few gates in a path since statistically, positive and negative variation will be observed in that path.

D2D variation cannot be addressed with on die techniques, since the whole die is biased towards a process corner. Therefore, this work aims to address WID variation. WID affects blocks in different regions of the design differently. This makes that each clock region is limited by the slowest block within it.

For a MOS transistor gate delay can be formulated as follows, where V_{dd} is the supply voltage to the core, L_{eff} is the effective channel length, K and M are fabrication constants [4].

$$delay \propto \frac{V_{dd} \times L_{eff}}{K \times ln^2(e^{M \times (V_{dd} - V_{th})} + 1)} \quad (1)$$

Due to the exponential component with the $(V_{dd} - V_{th})$ term, the gate delay exhibits more sensitivity to V_{th} variation at supply voltages close to V_{th}. The delay variation with regard to L_{eff} however is linear. This is particularly important for NTC applications, since V_{th} will have more effect in those cases.

C. NTC and Power Performance Trade-off

The main argument for the adoption of NTC is its energy-efficiency in terms of energy per-operation [4]. This is because, frequency has a linear relationship with the voltage down to the near threshold region, while power is reduced in a cubic relationship with voltage. Thus, for instance, going from 1V to 0.5V, the device delay increases by factor of 2, while the power is lowered to $(1/2)^3$ of the original value which results in a reduction of energy

consumption, as a product of power and delay, to $(1/2)^2$ of its original value.

The performance loss can be compensated with extra resources to support more parallelism. If we consider 2× more resources to compensate for the 2× increase in delay, the power would increase by a factor of 2, and the delay would decrease by half, leaving the energy reduction unchanged. So ideally, without considering the impact of PV and faults, energy consumption can be cut to $(1/2)^2$ for the same performance target.

However, as mentioned, the effects of PV, in particular of V_{th} variation, are exponential with $(V_{dd} - V_{th})$, meaning that the effect of variation is more significant at lower voltages. Therefore, our approach is particularly useful in NTC applications.

IV. GPU Stacking

In this section, we take a bottom-up approach to construct the GPU Stacking model. We start by analysis the fundamentals of how stacking can help mitigate PV effects, and build upon that to get to the stacked GPU.

A. Process Variation Compensation with Voltage Stacking

GPU Stacking provides a unique opportunity to manage the effects of process variation. An increased channel length (L_{eff}) or an increased threshold voltage (V_{th}) due to process variation will result in higher impedance of the channel and slower device.

In a conventional parallel power delivery system, the adverse process variation results in a lowered current Iin_i through the core$_i$, as $Iin_i = V_{dd}/Z_i$[4]. Since gate delay is inversely proportional to the Iin_i, it will result in a higher delay and a slower core. To compensate for the lower Iin_i, higher voltage can be applied to the core, or the body bias could be adjusted to reduce the gate delay. In short, the adverse effect of process variation can be compensated by delivering higher voltage.

In a stacked configuration, the same current I_c passes through different stack levels. Therefore, higher impedance of core$_i$, due to adverse process variation, results in a higher voltage across it ($V_i = I_c Z_i$).

Equation 2 shows the voltage across each core. Index i is the core or *lane* number. Depending on the switching activity of the circuit, the equivalent supply to ground impedance of a core changes during execution. This will be referred to as $Z(t)$. Process variation will bias Z according to the magnitude of the variation. Switching activity (or the running application) will change the transient aspect of Z. Using a performance and power simulator, we can obtain the core power traces and when physical dimensions of the PDN are given, using a circuit simulator, we can analyze the impedance change over time [28]. In our experiments, since the power consumption changes over time, the impedance of the circuit has also varied over time.

[4] $Z(t) = R(t) + jX(t)$. The real component R defines the relationship of the magnitudes of I and V, and the complex component X affects the phase of V and I. In VLSI circuits the switching of the core will cause complex effects and thus phase change. To avoid impedance effects we discuss adding extra coupling capacitors.

$$V_i(t) = V_{in}(t) \times \frac{Z_i(t)}{\sum Z(t)} \quad (2)$$

Utilizing the inherent feature of stacking is a key contribution of this work. The core with higher impedance due to adverse process variation will have a higher voltage drop across its power terminals. This results in a core speed up, relative to its speed without the higher voltage and with respect to a conventional power supply system, as delay is inversely proportional to the voltage. This, of course, comes at the cost of a lower speed for the other core in the stack. So stacking enables the slower core to run faster, relative to its speed in a conventional configuration, at the cost of the faster core running slower. In other words, the effects of process variation are intrinsically balanced in a stacked configuration.

Ideally, the variation effects in a stacked configuration converge to an average variation. For this simplistic example, let us assume linear effects of PV[5], then one can expect the frequency of the stacked cores to converge to the average of the two cores in a conventional configuration. For example, a core with 10% variation compared to the nominal value runs at $0.9f$ in a conventional configuration, and a core with -10% variation runs at $1.1f$. Stacking these cores would result in both cores exhibiting delay properties similar to the nominal values and run at about the nominal frequency ($\frac{+1.1f+0.9f}{2} = f$). After all, the nominal value is nothing more than the mean properties across all the samples. A non-symmetrical example, would have non-ideal compensation, e.g., $c_1 = 0.8f$ and $c_2 = 1.1f$ would result in an average of $0.9f$, which is better since the cores would have to run at $0.8f$ if stacking was not applied.

Now, we apply this reasoning to a SPICE simulation of a toy circuit, to test how this compensation works in a simple circuit. SPICE simulations (at 45nm technology [29]) are performed for an example case where inverters are configured in conventional and series with two stack levels. The test circuit consists of an inverter driving $4FO4$, to exacerbate the delay effects with such a small circuit. The stacked configuration is supplied with $1.2V$. We test three configurations: nominal (ref), not stacked with PV (PV), stacked with PV (s-xxxx). For the stacked configuration, the variation is set to affect the header inverter positively (i.e., shorter L_{eff}, thus faster), and the footer negatively (i.e., longer L_{eff}, thus slower).

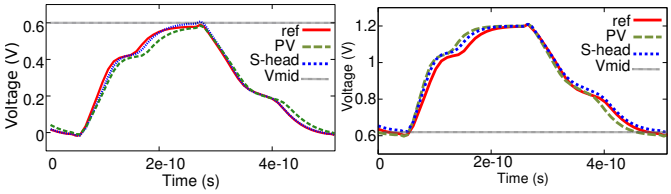


Fig. 3. Stacked configuration ($S - foot$ and $S - head$) intrinsically mitigates the variation effects. These signals are closer to the case without process variation ref, than the non-stacked baseline PV. Since V_{mid} has shifted, there is more voltage available for the slower part of the design.

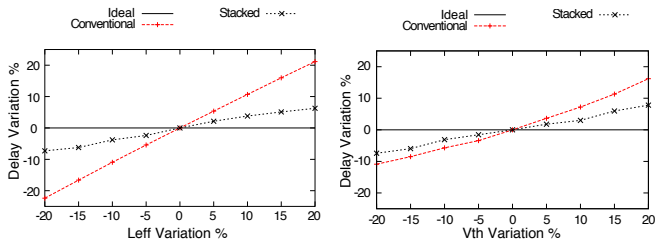
In this example, there is a 20% process variation in effective channel length (L_{eff}). If the channel length

[5] This is not considered in our evaluation.

increases, the response time will be slower than the nominal non-variation case (lines PV and ref at the bottom of Figure 3). And if the channel shortens, the opposite effect will be observed (PV will be faster than ref at the top of Figure 3). However, this also implies different power consumption for each, as seen above.

The simulation results show that with voltage stacking, the voltage rail between the two levels of the stack (V_{mid-pv}) settles around 0.63V. This is because the header transistors have less resistance due to the shorter L_{eff}. The header transistors are effectively supplied with 1.2V - 0.63V = 0.57V, and the footer transistors with 0.63V. In this case, there is a mitigation of the delay variation, shown by $S-head$ and $S-foot$, which are closer to the scenario without process variation. This is a reduction of more than half the delay variation introduced by process variation. Next, we evaluate the effects of V_{th} variation on the inverter delays. The same scenarios are simulated, except that this time the variation is on V_{th}. The transient response for this case is similar to that of Figure 3, and thus the graph is not included.

Figure 4 summarizes all the experiments for different variation values from -20% to +20%. We evaluate both L_{eff} and V_{th}. Note how the delay variation is smaller with the use of stacking. Only one line is presented for the stacked configuration, since the stack position (header or footer) does not change the result.



(a) L_{eff} (b) V_{th}

Fig. 4. Mitigation of the variation effects compared to conventional configuration.

The SPICE model confirms the premise of intrinsic mitigation of process variation in stacked configuration. Throughout the text, "variation" refers to the combination of the effects of different sources of process variation, and its total effect on the frequency is measured, unless otherwise specified. However, as could be expected, the compensation is not ideal, being specially suboptimal for V_{th}, due to the non-linearity dependency between delay and V_{th}.

B. Which Lanes to Stack?

Now that we established how PV effects can be compensated by voltage stacking, we look into the problem of how to decide which GPU *lane*s should be stacked for optimal results. Stacking neighboring *lane*s can mitigate the process variation effects regionally. For example, consider the die shown in Figure 5. The figure shows four *lane*s with differing amounts of variation compared to the nominal properties. Stacking the *lane*s based on their adjacency (i.e., *lane* 1 and 2 as one stack, and *lane* 3 and 4 as another) will help mitigate the worst case variation (i.e., 20%). The 1-2 stack would operate at about -15% of the nominal frequency ($(1.1f + 1.2f)/2 = 1.15f$, in practice the attenuation would be less). The 3-4 stack would operate at about 15% of the nominal frequency ($(0.9f + 0.8f)/2 = 0.85f$). The actual frequency must be that of the slowest stack.

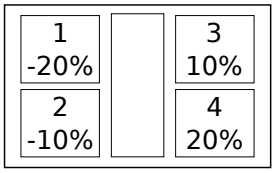


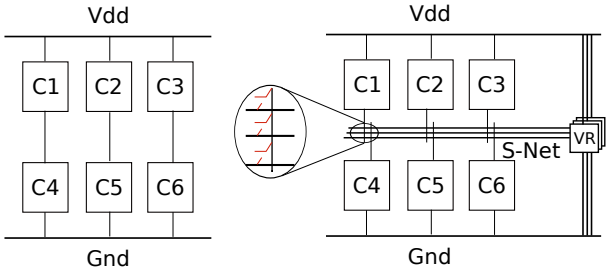
Fig. 5. Sample die with 4 *lane*s and different variations.

For the best results, however, the stacking configuration has to be determined based on the observed variation, i.e., to stack a *lane* adversely affected by variation with a *lane* positively affected. For example, stacking *lane* 1 and *lane* 4 together, and *lane* 2 and *lane* 3 as another stack would result in each running at about the nominal frequency. The best strategy is to cluster, in each side of the stack, *lane*s minimizing the standard deviation of variation (the rationale is that *lane*s with similar process variation require approximately the same compensation). Then, the cluster with maximum process variation average in the header should be stacked with the one with minimum negative variation in the footer. However, this might not be trivial to find, since the number of possible combinations is large. A simpler approach, used in this study, is to have the same number (for instance N) of *lane*s in all the clusters. The clustering is made by simply picking the N *lane*s with maximum variation in the header and clustering them with the N *lane*s with negative variation in the footer. The process is repeated for the remaining *lane*s. This simpler approach works fine in the level of lanes, since their spatial proximity causes similar variations. Note that although this example shows an optimal case with opposite variations having the same magnitude, this is not a requirement.

Since the variation is not known until after fabrication, a configurable fabric is needed to group and stack the *lane*s based on the observed variation. We adapt the idea of a configurable power delivery fabric [14], [30] to allow the connection of *lane*s that are not neighbors, because in terms of variation management, neighboring *lane*s are most likely, affected similarly. Our proposed fabric capitalizes on the fact that GPU Stacking is actually better suited for the connections of logic with opposite variation effects. Note that this may cause a problem, given that it increases the path from V_{dd} to Gnd. Thus, there is a design trade-off here: on one hand, the compensation would be better if the stacked logic were farther apart in the chip, on the other hand, if this distance is larger, the voltage droop due to voltage rails and switches is increased. That is another reason why stacking lanes is a good design choice as opposed to stacking SMs. Lanes are well constricted in space, within the SM, while SMs will be farther apart in the chip, still an SM is large enough to be used for the purposes of GPU Stacking, i.e., there is enough variation in the SM to allow for the type of

compensation we aim, as observed in our evaluation.

To simplify the design, we propose to cluster the stacking of *lane*s. The clustering is a trade-off between cost and complexity. To cluster *lane*s for stacking, we define Shared Net. Shared Net is a common net that connects a number of *lane*s. For example, V_{dd} is conceptually a shared net. However, we specifically use the term Shared Net for an intermediate net that connects a number of *lane*s in a stacked configuration (V_{mid}). The number of Shared Nets is a design parameter and changes the trade-off between area and compensation granularity (more Shared Nets result in a more fine-grained compensation).



(a) Stacked (b) 3 Shared Nets

Fig. 6. Shared Net configuration simplifies stacking and supports post-fabrication configurability.

Figure 6a shows a two-level stacked configuration of six cores with no shared nets. The stacking configuration is static, and is determined at the design time. As the variation is not known at the design time, and there is a spatial correlation in the variability, this scheme is not likely to provide the compensation opportunities. Figure 6b shows the design with three Shared Nets. To simplify the proposed design, each *lane* is fixed to either header or footer at design time. This choice certainly reduces the freedom of the system, but because there are several *lane*s per SM, the loss is minimal. The resistance in Figure 6b Shared Nets is not modeled in this study. The scheme is similar to a multi-power domain case, where multiple voltage rails are present and each part of the circuit connects to a different rail. The configurability only applies to the intermediate nets.

The number of Shared Nets is determined at design time based on the expected variation of the technology, or the severity of variation effects on the design metrics and to the level they need to be managed. This decision has to be based on the variation profile for the fabrication technology. Given the knowledge on the variation, it is possible to calculate the expected power consumption for a block (in relative terms), e.g., using Varius-NTV models [4], the power information can then be used to calculate the expected V_{mid} voltage for a given stacking configuration, which can finally be used to calculate the expected performance. The decision on the number of Shared Nets is then a trade-off between the cost of adding an extra Shared Net and the extra performance boost gained. This is explored in our evaluation.

During post-silicon testing, each *lane* can be tested to characterize the observed variation for that particular *lane*. This increases binning time, but current chips already undergo this type of testing for speedgrade purposes. Once the effect of variation is known for each die, the *lane*s can join a cluster based on their observed variation. Note that the clustering is static and done once in the lifetime of the chip in a calibration step, right after fabrication. To allow for the post-silicon configurability, an array of power transistors or fuses can be used between each *lane* and each Shared Net. Power transistors are present in modern designs for power gating purposes. Although we did not consider them in this design, the reduced current passing through them (due to voltage stacking) will largely reduce their impact on the circuit.

When multiple Shared Nets are present, the clustering is done by stacking the same number of *lane*s (namely n) on each side (foot or head) of the Shared Net. The *lane*s are sorted by variation (minimum delay), for both head and foot groups. The n first *lane*s in the head group are stacked with the n last *lane*s in the foot group. The next n *lane*s in each group are stacked together, and so forth. This configuration will have the maximum compensation within each SM, as *lane*s with opposed variation trends tend to be in opposite sides of the same stack, replicating the behavior observed in Section IV-A.

C. Divergence and Extreme Conditions

The last component that needs to be addressed is the presence of diverging code executing in each stack level, which has the potential to cause power mismatch and make V_{mid} to diverge too much from safe levels.

However, V_{mid} cannot be fixed by the means of an extra traditional voltage regulator, as it is usually the case in voltage stacking designs. The process variation compensation comes from the fact that V_{mid} is "floating", i.e., is not at a fixed frequency. Footer and header groups have different voltages instead. Nevertheless, it is possible that due to load unbalance (caused by GPU divergence), or e.g., , extreme temperature conditions, the voltage difference is such that either level has not enough voltage to guarantee correctness. We call this "voltage starvation". We propose different mechanisms to handle such scenarios.

Dummy-activity is inserted through the activation of parts of the lane that are not being used. For instance, if divergence is observed for long periods of time, this could shift the voltage towards the most active *lane*s. This shift can be canceled by adding activity in inactive *lane*s, or inactive parts of *lane*s. When Dummy-activity is inserted, the *lane* does not commit any change to the architectural state, nor does it execute stores, for obvious reasons.

"Lane turn-off" is a more drastic measure for extreme cases. In this case, there is no scheduling for one of the lanes in the level consuming more power than expected. This can only be done in architectures where each *lane* within the SM can execute different code, and would require awareness in the scheduler to be able to maintain correctness. The actual "turning-off" may be done in terms of power-gating, which requires *lane* level power-gating or in terms of scheduling/clock-gating. If after the first *lane* is turned off, there would still be deviation, a new *lane* is then turned-off. This could lead to big impact on performance, thus Dummy-activity is a preferable solution whenever possible.

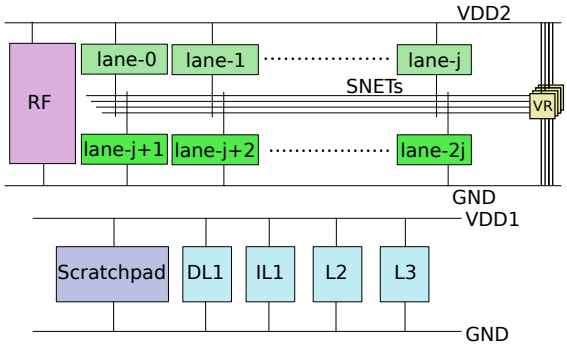


Fig. 7. GPU Stacking allows a more fine-grained voltage adjustment per lane.

Additional VRs are used for extreme cases. In the stacked logic, it is expected to have V_{mid} floating, however, within a range that guarantees its functionality. Two small integrated voltage regulators are used, one pull-up and one pull-down, which would be activated when V_{mid} falls below 0.4V or rises above 0.8V, to guarantee correct behavior and avoid bit flips. Since these regulators only cap extreme cases, they are not used in regular operations, therefore, they can be small and their potential inefficiency is not problematic in the overall design. Since dummy-activity and lane turn-off have to be implemented in the micro-architecture level, they are naturally slower, and thus extra VRs are only used to give enough time for those to be activated.

The natural candidate for triggering these mechanisms is V_{mid}, so in the case of the architectural mechanisms, the scheduler needs to be aware of the voltage during the regular operation of the GPU. VR is always connected and is triggered without any architectural intervention. In our experiments (details in Section VI), those mechanisms were never activated for any of the benchmarks tested, even when divergence was observed (for instance in BFS).

D. Final Design

In this section, we present our GPU Stacking final design. GPU Stacking does not change GPU organization (Figure 2), nor does it affect the placement. GPU Stacking divides the GPU into different power domains, one for non-stacked structures (e.g., caches, shared memory), and one "super" power domain with the stacked *lane*s (and associated register files). It is a "super" power domain because, to be precise, each stack is a power domain of its own, but this is not known before fabrication. Figure 7 shows the proposed micro-architecture, from a power delivery perspective. The figure shows a solution with four Shared Nets.

We adopt the stacked SRAMs previously proposed [11] (RF in Figure 7). Note that the SRAM stacked design has read and write ports at nominal voltage, and thus level shifters need to be added for read and write to the register file. There are multiple options available for level shifters suitable for voltage stacking [31], with different trade-offs between area, power and performance.

One concern about GPU Stacking is the area overhead due to Shared Nets. Introducing extra Shared Nets could increase the total amount of metal dedicated to power rails. On the other hand, GPU Stacking decreases the overall current of the chip and metal from V_{dd} and *Gnd* rails could be reduced.

Let m_b be the total metal budget for power rails in a chip. For a 2-level stacked system (as the one proposed here), the overall current is reduced by half $I_{stack} = I_{base}/2$. Thus, V_{dd} and *Gnd* could have roughly half the metal budget as needed in the baseline[6] ($m_b/4$ for each). For a system with n Shared Nets, each Shared Net takes I_{stack}/n current. Thus, each Shared Net needs $\approx n$ times less metal than V_{dd} ($m_b/4/n$). Consequently in GPU Stacking, the total metal budget for power rails is $m_{vdd}+m_{gnd}+m_{snets} = m_b/4+m_b/4+n\cdot m_b/4n = 3/4\cdot m_b$. In other words, in a GPU Stacking configuration it is possible to reduce the amount of metal dedicated to power rails. In our evaluation, we assume that the metal budget is kept constant instead (i.e., V_{dd} and *Gnd* metal is reduced by 1/3 only), which in turns helps improving the PDN by reducing the resistance.

This solution is suitable for systems not bound by power density. However, if power density becomes an issue, a more elaborated power delivery network would be needed. It could be achieved by the observation that each *lane* only taps either V_{dd} and Shared Net or *Gnd* and Shared Net. Thus, it is possible to reduce the amount of tracks dedicated to the unused rail on top of each *lane* and increase the density of the relevant rails.

In any case, a more thorough analysis of the area overhead would be required when implementing a design that uses GPU Stacking. However, since area overhead is usually not a major concern in modern chip designs, we do not put much emphasis in this analyse and leave it as future work.

V. Experimental Setup

To evaluate GPU Stacking, we start by determining what is the optimal configuration in terms of SMs and *lane*s for a NTC GPU. Non-NTC commercial GPUs vary in size from 2 to 16 SMs, and can have between 16 and 64 *lane*s per SM. After deciding on the baseline GPU for our experiments, we evaluate the potential of GPU Stacking to compensate for process variation both in performance and power. Then, we evaluate the reliability of the PDN, as it is a main concern in voltage stacking proposals.

A. Baseline GPU Choice

A modified version of ESESC [32] is used to simulate a GPGPU. For power estimation, we use a GPGPU model developed based on McPAT [33], very similar to GPUSimPow [34]. McPAT, takes the microarchitectural activity statistics from ESESC, and calculates the power consumption of each component. This simulation setup provides both dynamic and leakage power as well as performance for the applications. The temperature dependency of leakage is also taken into account by ESESC.

We simulate GPGPUs with a range of configurations, created either by varying the number of SMs or the

[6]For clarity, we are referring to the amount of tracks dedicated to rails, not their pitch.

structures within each (e.g., number of *lane*s in each SM). This is summarized in Table I. McPAT [33] tool estimates the power consumption of the GPGPU model and only the on chip structures are modeled for this experiment. Since the stacking is applied within the SMs, the number of SMs does not affect the result per SM.

Benchmarks used are from popular suites Rodinia [35], Parboil [36] and CUDA SDK (bfs, cfd, convolution, hotspot, backprop, lbm, transpose, srad and sgemm).

TABLE I
Simulation parameters

Parameter	1x	1.5x	2x
*lane*s per SM	32	32	64
RFs per SM	32K	64K	64K
DL1G-Scratchpad memory per SM	32KB	32KB	64KB
Maximum Frequency	1.5 GHz		
Streaming Multiprocessors (SM)	up to 8		
Threads per warp	32		
Maximum Warps per SM	24		
L2	256KB 16w		
L3	4MB 32w		
Memory access latency	180 cyc		
V_{dd}	0.4-1.0 V		
V_{th}	0.30 V		
δV	0.1		
Ambient Temperature	25C		

B. Process Variation Modeling

To evaluate the impact of GPU Stacking to compensate for the performance loss due to PV, we model PV following an existing methodology [37]. Briefly, VARIUS-NTV [4] (planar) and VARIUS-TC [38] (FinFET) are used to generate process variation maps. VARIUS models both fine grained (Within Die) and coarser grained (Die-to-Die) variation – both systematic and random components [39]. The systematic component is modeled using a multivariate distribution with a spherical spatial correlation structure and the random component, which occurs at the transistor level, is modeled analytically. VARIUS divides the chip into n small equally sized rectangles. Each grid point has a systematic variation of L_{eff} and V_{th} which are assumed to have normal distribution. The random variation of L_{eff} and V_{th} is treated differently because of the level of granularity at which it occurs and it is assumed to be distributed normally and without any correlation [39].

Given the GPU floorplan as the input, VARIUS-NTV provides die maps each with a specific process variation case. The goal is to use VARIUS-NTV to consider the worst process variation over maps or die maps and understand how V_{mid} behaves in extreme cases of process variation effects or an application. VARIUS-NTV also outputs normalized delay, normalized L_{eff}, and effective V_{th} for each component of the GPU die (lanes, caches, register files, etc.). The information from VARIUS-NTV is then used to calculate the expected power for each element in the stack and then the expected voltage on each Shared Net. The calculated V_{mid} is then fed back into VARIUS-NTV to calculate the delay and power after compensation in the stacked configurations.

This experiment is performed for different number of Shared Nets, where we compare our scheme with a conventional non-stacked baseline and against multi-frequency domain, which has been shown to have promising results in mitigating process variation effects [7].

C. Power Delivery Simulation and Technology Node

To evaluate the V_{mid} noise and voltage noise behavior in the GPGPU PDN, we adapt the methodology proposed by Leng et al. [28]. In short, a off-chip and on-chip power delivery network is simulated with cores modeled as current-controlled current sources, where the transient current is estimated by cycle-accurate micro-architectural simulation for each core. This type of approach has been used in multiple studies of this sort [40], [37], [41]. Thus, we use the power traces from ESESC for each *lane*.

Our SPICE simulations model the printed circuit board (PCB), the package [28], and the on-chip PDN, using the IBM Power Grid benchmark (ibmpg1t) [42]. Figure 8 summarizes the complete PDN with the simulation parameters. The grid is represented by the four resistors in the box named "PDN", but the simulation is performed using the full grid. On-package capacitors (C_p) are used to stabilize the voltage on Shared Nets and smaller on-die capacitors (C_d) are used to eliminate fast transient response due to mismatch. Although the figure only shows one set of on-package caps with respective C4 bump, we use one set per Shared Net. The cores are modeled as variable resistances based on the power traces from ESESC. This methodology is compatible with current industry practices, and short of fabricating a chip, it is the best available method for this type of low-level analysis. For technology node, we use planar CMOS at 45nm [29], and FinFETs devices at 15nm [43].

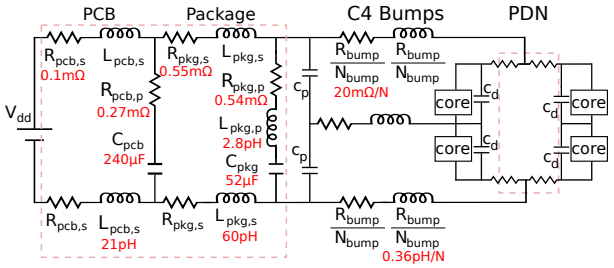


Fig. 8. The complete power delivery model used for simulations including on small on-die decoupling caps ($C_d = 100fF$ per Shared Net) and larger on-package decoupling caps ($C_p = 5uF$ per Shared Net).

For between level communication, we use the level shifter proposed in [13] which has been shown to provide good performance/energy/area trade-offs [31]. Our SPICE simulations show that this circuit has one FO4 delay overhead at the NTC voltage, when communicating between different stacks, however these level shifters can substitute buffers that were present in the design, minimizing the performance impact. Memory and caches are not stacked in this study, and since the process variation is being mitigated due to the stacking in the cores, it is expected that they will end up achieving a higher frequency than memory. To keep the improvement on the logic side, we consider an increase in the number of access cycles rather than reducing the frequency of the core.

We consider both single-clock domain, and multi-clock domain, in which case, each SM runs at a single frequency. In order to meet timing, this will be the frequency of the slowest *lane* (after compensation). It is possible to have different frequencies for different SMs, but in this study only one frequency is considered for the whole GPU.

VI. Evaluation

We first look into the baseline selection results, since all other results will be built upon this GPU. Then, we present the main results on how GPU Stacking can mitigate PV effects and present other advantages of our approach. Then, we evaluate different aspects related to the stability of our approach to get a comprehensive idea of how GPU Stacking affects the power delivery of a chip. Finally, we investigate how the trade-offs presented by GPU Stacking are changed when FinFET are used and finish our evaluation discussing design and fabrication aspects of this new approach.

A. GPU Sizing and Baseline

We start our evaluation with a careful analysis of performance, energy and area trade-offs of GPUs using NTC. We consider different voltages and number of SMs. Figure 9 summarizes energy-delay (ED) and energy-delay-area (EDA) products for $1x$ with different number of SMs. The y-axis shows the normalized value for each metric with reference to the $1x$/4SM configuration at 1V supply. The x-axis is V_{dd}. In general, energy decreases as the V_{dd} approaches near threshold region. Then the delay starts to degrade more rapidly, increasing the energy consumption, mainly due to clocked logics and leakage.

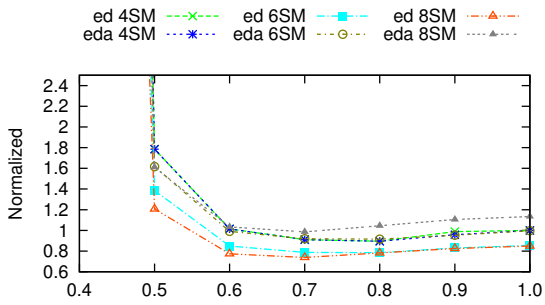
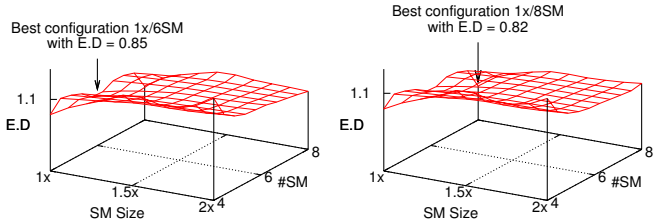


Fig. 9. Designs with more SMs become more efficient in terms of energy-delay product, as well as energy-delay-area product as V_{dd} decreases.

To understand the effect of a NTC GPU, let us examine the 4SM and 8SM configurations at both $1.0V$ and at the NTV voltage of 0.6V. Clearly, 8SM has a higher EDA than 4SM at $1.0V$. However, as the V_{dd} approaches the near threshold, EDA of both configuration decreases. Moreover, at around $0.6V$, 8SM configuration has lower EDA than 4SM. This means that the 8SM configuration is more efficient at lower voltages. Also the delay metric for 8SM configuration at this point is the same as the delay of the baseline 4SM running at $1.0V$. This shows that investment in extra resources pays off as the V_{dd} approaches the near threshold region by maintaining the performance within 4.8% of $1x$/4SM configuration operating at $1.0V$, while reducing the power consumption to about 43% of baseline. At the cost of more area.

Another observation is that the optimal configuration for different metrics changes by changing the V_{dd}. For example Figure 10a shows the design space for ED at $1.0V$. The optimal configuration is $1x$/6SM. Bigger structure sizes for cache or number of *lane*s could increase the performance, but the increased power makes such a trade off less desirable due to power budget constraints and possible thermal issues. Figure 10b shows the design space for the same metrics at $0.6V$. At this condition, the optimal configuration is 8SM. Also the relative efficiency of bigger structure sizes (e.g., 1.5×) increases. This implies that the architectural parameters, such as cache or RF size, should be reconsidered for maximum efficiency as the operating voltage changes.



(a) Typical (b) Near Threshold

Fig. 10. The optimized baseline in near threshold region is different from the typical super threshold region. Larger structure sizes for cache or register file or number of lanes could become more desirable in near threshold. This demands reconsidering the architectural parameters to obtain the best energy efficiency, rather than just lowering the voltage.

B. Benefits of GPU Stacking on PV

Now that we established the baseline configuration of the NTC GPU, we look into the positive effect of GPU Stacking on PV. We used the variation maps generate by VARIUS-NTV, where each map correspond to a die. Then we estimated performance and power for each die. Each die presents a unique PV, and thus GPU Stacking will have different effects on each. We also look at the effect of different numbers of Shared Nets in the design.

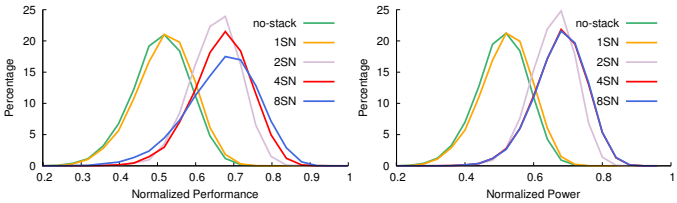


Fig. 11. The proposed techniques shift the performance and power towards the ideal scaling with no process variation.

Figure 11 shows histograms for performance and power, i.e., the y-axis is the percentage of chips (out of 10K chips) and the x-axis is the performance/power. Performance and power are normalized to the value obtained in the case with no process variation. Both Non-Stacked and Stacked (xSN, where x is the number of Shared Nets) methods

are shown. The nominal case, i.e., with no PV, would be in the $x = 1$ mark. We notice that without stacking and with PV, the average performance is around 50% of the nominal. Using one Shared Net per SM does not improve the non-stacked case much, since all the *lane*s share the same Shared Net. Two, four and eight Shared Nets provide good design solutions, with eight being pretty close to four in terms of power, but with slightly lower power. The best configuration is arguably eight Shared Nets in this case, but two Shared Nets present a good trade-off between design complexity and results.

Overall, GPU Stacking delivers about 75% of the performance, with 75% power, compared to the no variation conditions. This represents a reduction in the degradation due to process variation: 37% in performance, and 39% in power compared to the conventional configuration. The increase in power is due to the increased frequency, but Energy-per-instruction remains roughly the same. This may seem like a no-gain approach, but means that GPU Stacking is able to reduce the effects of process variation, delivering a chip that is closer to ideal scaling. For the sake of comparison, when using ASV with four power domains, it would be possible to deliver only about $\approx 62\%$ of nominal performance, on average, while with 4 Shared Nets, it would be possible to deliver $\approx 70\%$ of nominal performance, on average.

Another way of mitigating the effects of PV is to use multi-clock domain [15]. Since each SM in a GPUs operate in lock-step, the maximum number of clock domains possible is equals to the number of SMs. However, our technique is orthogonal to multi-clock domains and could be applied in combination to it. Thus, we repeat the previous experiment, but considering multi-clock domains. Our results (Figure 12) show that, by only using multi-clock domains, it is only possible to slightly improve the performance to close to 60% of the nominal performance. When also applying GPU Stacking, there is a small improvement with $2SN$. $4SN$ and $8SN$ have very similar performance, but four Shared Nets have better power consumption. Thus, $4SN$ seems to be the best configuration, with two Shared Nets providing a good trade-off point. In summary, combining multi-clock domain and GPU Stacking delivers about 80% of the nominal performance, with 70% of the power, which is a 20% improvement over multi-clock domain only.

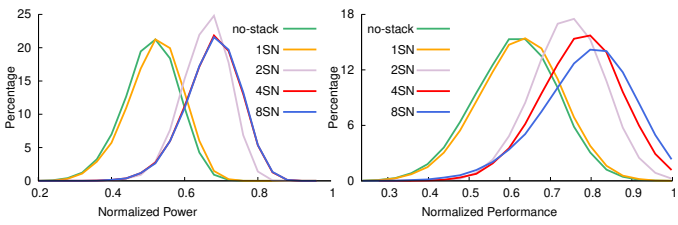


Fig. 12. When multi-clock domain architecture is used, GPU Stacking shifts the performance further towards the ideal scaling with no process variation.

C. Other advantages of GPU Stacking

Besides compensating PV effects, using GPU Stacking has other advantages. The increase in the supply voltage has some indirect advantages, some of which are quantified in this section. They share two main causes: voltage regulators are more efficient at higher voltages and the overall current in the circuit is reduced to roughly half [13], [8]. Using results from Hong et al. research [44], we estimate memory power as 17% of the total GPU power. The memory power is not affected by GPU Stacking.

TABLE II
GPU Stacking delivers better performance and power than a non-stacked configuration under process variation. It also allows better VR efficiency, and reduced number of power pins.

Parameter	Expected Variation
Performance vs. PV not-stacked	+37%
Power vs. PV not-stacked	+39%
VR area	-41%
VR efficiency	+10%
Supply pins	-41%

Some of the advantages of GPU Stacking are related to voltage regulators: VR area is reduced by roughly 2× with half the current [45]. In the Intel Haswell integrated VR, each $2.8mm^2$ cell can delivery a 25A maximum current. A modern low-end GPU consumes $\approx 55W$ [46]. VR efficiency is a function of both V_{dd} and output current [47]. Considering both the efficiency from increasing V_{dd} from $\approx 0.6V$ (at the near-threshold region) to $\approx 1.2V$, and reducing by half the total current, we expect an improvement in VR efficiency of 12%. Also, there is a $\approx 50\%$ reduction in current drawn by stacked logic, that can yield a $\approx 41\%$ reduction in VR area.

The number of pins and pads is mainly determined by the total amount of current flowing through them. To keep the current per pin constant, it is now possible to reduce the number of pins. Again the current related to the logic decreases to roughly half, but the current in memories is the same. Once more, this yields a reduction of 41% in the total number of power pins. Note that the number of pads dedicated to V_{dd} can be also decreased, but pads are now needed for Shared Nets, since on-package decaps are used. The overall number of pads is not expected to change. Table II summarizes the expected variation in multiple chip parameters due to GPU Stacking.

D. Analysis of PDN Stability

We also looked into the PDN stability under different optics to make sure that GPU Stacking can reliably execute different applications keeping V_{mid} stable and by looking at power distributions concerns across the chip.

The main concern is that divergence between application threads could cause V_{mid} to diverge from acceptable levels. In typical GPGPU applications, threads exhibit very similar activity rates. This minimizes the possibility of a load mismatch in the stack. In our experiments, we observe that the power consumption of *lane*s are within 5% of each other 98% of the time, and within 10%, 99.2% of the time. $Power_{lane1}/Power_{lane2}$ averages 1.000024, with standard deviation of 0.070. The sampling rate for our measurement is on the order of 1-10 MHz.

One source of concern is peak difference, which is as high as 40% in our experiments. We observe that mismatches higher than 30% only occurred for the backprop benchmark, but only 0.1% of the time for that benchmark.

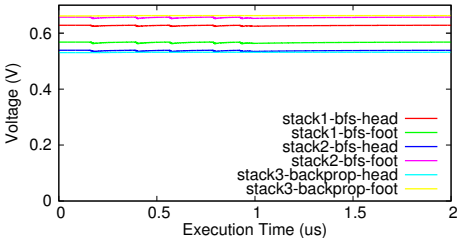
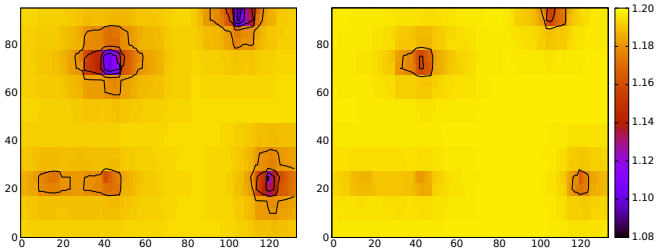


Fig. 13. It is possible to maintain the voltage in each level of the stacks with on package decoupling capacitors .



(a) Conventional (b) GPU Stacking

Fig. 14. GPU Stacking reduces the IR drop by reducing the total current flowing through V_{dd}. The stacked configuration is feed with $1.2V$, while the conventional was scaled up for better comparison.

Since the elevated mismatches are observed in a very short interval, they can be handled by decoupling capacitors.

To examine whether the achievable range of decoupling capacitance in the design is enough for the observed mismatches, we ran a SM-like design through synthesis and back-end design down to GDS. We then extracted the capacitance of the supply nets. Our experiments show that such a design using standard cell decoupling capacitance cells (dcaps) would have total capacitance of $1.4nF/mm^2$. At the super threshold region, with a power density of about $1W/mm^2$, the time constant for the power supply RC circuit would be on the order of a couple of nanoseconds. Such a small time constant is not enough to sustain the transient mismatches that appear on the order of $100ns$ to $1us$. At the near threshold region, with smaller power density, the time constant would be on the order of 10 nanoseconds which is still not enough. Therefore, we use of on-package capacitors, which are commonly use for PDN stabilization. Adding additional capacitors for the Shared Nets will not increase the package pins [9].

In the PV case, V_{mid} is expected to deviate from the nominal voltage, and this behavior is desired, since it is the source of the PV compensation. We evaluate how the load mismatch affects the voltage available for both levels of the stack by carrying a SPICE simulation of the model presented in Figure 8. Each stack contains 16 lanes, 8 in the header and 8 in the footer, a total of 4 stacks (2 SMs with 32 *lane*s each) are hooked in the grid, equally distributed, each V_{mid} has two $5\mu F$ on package decoupling capacitor (one between V_{mid} and Gnd and the other between V_{mid} and V_{dd}). BFS and backprop are run, one in each SM (those where the two benchmarks with higher mismatch between lanes).

Figure 13 shows the on-chip transient voltage for each stack level during execution, one of the stacks is omitted for clarity. The voltage source is $1.2V$. Instead of plotting the voltage with relation to the global Gnd, we plot the local voltage difference, which is more meaningful. Voltage for each stack level stays within 10% the expected voltage for compensation, showing a very good balance. This also implies that the minimum voltage for our technique to work is V_{th} +10%.

Another concern regarding the PDN is lateral current and IR drop. The main concern here is that if stacked *lane*s are further apart within the chip, there will be increased resistance for current to traverse that distance.

The total current for V_{dd} in GPU Stacking is roughly half the current than the conventional case, therefore, one would expect decreased voltage droop. On the other hand, given a fixed budget for power delivery, the insertion of new power rails requires that some of the resources used by V_{dd} and Gnd in the conventional scenario be used to Shared Nets instead. Since the current in this rails is expected to be much smaller, the resource reduction in V_{dd} and Gnd is small. We consider that 1/3 of the metal used to power delivery is used to Shared Nets. Figure 14 shows a 2D color map of the IR drop for V_{dd}, both in the conventional configuration and in GPU Stacking for the BFS benchmark from our SPICE simulations, already considering the reduced resources for V_{dd}, the contour lines are traced every $0.02V$. The conventional configuration was scaled up from $0.6V$ to $1.2V$ for better comparison.

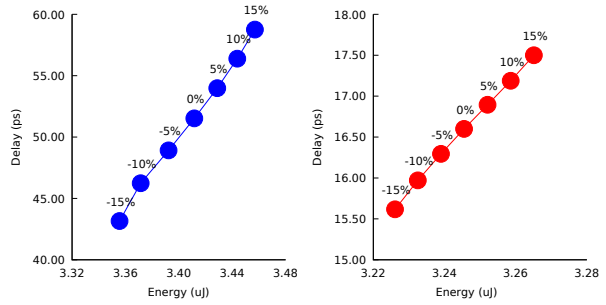
For V_{mid}, we start by finding the worst case from the variation maps, i.e., the case with maximum power difference between stack levels. We note that, since we are stacking lanes, within an SM, and given the spatial correlation of the variation, in most cases the variation is well below this maximum. To estimate the resistance, we use the IBM Power Grid Benchmark (ibmpg1t), which is properly scaled to estimate the equivalent grid of one Shared Net. The transient simulation considers the maximum variation case. The maximum observed IR drop in V_{mid} was $65mV$ in our simulations for all benchmarks, which is 5% of the whole supply voltage ($1.2V$), or 10% of the NTC voltage of $0.6V$, since we are using over $200mV$ from V_{th}, this is well within acceptable margins.

E. Stacking FinFETs vs. Planar CMOS

Now that we looked into the benefits of GPU Stacking and have analyzed the stability and the impacts on the PDN, we look into how FinFETs affect the trade-offs observed for GPU Stacking. We want to make sure that our proposal is still useful in newer technology nodes.

The first observation is that FinFETs seem to be less sensitive to PV effects than CMOS devices. Figure 15 shows the Energy vs. Delay for both CMOS and FinFETs, with L_{eff} variation from -15% to 15%. As we increase L_{eff} variation in FinFETs, the effect on energy-delay is much smaller than planar CMOS, which indicates that the effect of PV in energy consumption of FinFETs is not as significant as it is in planar CMOS. Since GPU Stacking relies on the difference of power to shift V_{mid} and mitigate PV, it cannot be directly applied in that case. Also, there is intrinsically less PV to compensate in this case.

Nevertheless, it is still possible to use GPU Stacking to compensate for PV effects on FinFET devices. To do so, we propose the use of the extra VRs to force the V_{mid}



(a) Planar CMOS 45nm (b) FinFET 15nm

Fig. 15. Delay increases more rapidly for Planar CMOS than FinFET as we vary L_{eff}.

voltage to a beneficial level. GPU Stacking is designed and evaluated in the same way, but during the binning process, a voltage level is chosen for each Shared Net based on the variation of the *lane*s assigned to it.

Now, we look into how much PV compensation can be obtained by applying GPU Stacking to a FinFET based GPU design. We start from the variation maps and calculate the expected performance and power for each chip, considering the best forced voltage for each Shared Net. Figure 16 shows histograms for performance and power. The y-axis is the percentage of chips (out of 50K) and the x-axis is performance/power.

In Figure 16 performance and power are normalized to the values of the no-process-variation case. We note that in this case, the no stacking version already presents performance and power numbers very close to nominal ($\approx$ 90%), still GPU Stacking is able to shift the curve above to 95% performance, on average. Although these results are less impressive than the CMOS counterpart, the compensation effect is still observed with FinFET devices. The multi-clock region experiment yielded very similar results and was omitted for the sake of space.

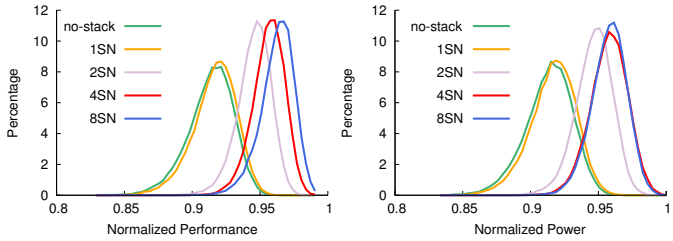


Fig. 16. The proposed techniques shift the performance and power towards the ideal scaling with no process variation.

F. GPU Stacking Practicality Issues

Finally, we finish our evaluation by looking into some practical aspects regarding fabrication, design, power-gating. We briefly discuss those issues in this section.

1) Extreme variation cases: In some extreme cases, the difference in variation between the stack levels may be significantly elevated, reducing the voltage in one of the levels to non-acceptable levels (compared to V_{TH}). After fabrication, chips are divided in performance/power grades or discarded according to the variation. This should also be the case in the GPU Stacking case. In practice, this is done during binning, where timing characteristics of the chip are assessed. In this paper, to evaluate how GPU Stacking affects yield, we use VARIUS-NTV [4] to calculate the expected V_{mid} for each stack. We then compare this value to the V_{th} of each lane in the stack. If the supply voltage in the lane is smaller than $1.2{\cdot}V_{th}$ for at least one of the lanes, we consider the chip discarded. Our simulations show that, in such scenario, 1.8% of the chips would be discarded. A more aggressive approach would be to disable lanes which do not meet such requirement, but we do not consider this case.

2) In-Rush current and power gating: GPU Stacking reduces the current drawn from the PDN by roughly half during the regular operation and start-up phase. Our SPICE models were simulated for both stacked and non-stacked configurations, and showed that in-rush is indeed smaller in GPU Stacking (data is not included).

Power gating can be applied in addition to the GPU Stacking. The only requirement is that pairs of lanes need to be powered off together (one in the head and one in the foot group), but there is no need to power gate one entire SM at a time.

3) Implementation flow: GPU Stacking operates in the physical level only and does not make changes in the RTL level. But the physical implementation flow needs to be altered. GPU Stacking creates voltage rails that will be connected to the "local" GND rail of some portions of logic and to the "local" V_{dd} rail of others. The idea of having multiple power rails is similar to multiple power domains. Power transistors can then be used to open or close the circuit and choose which Shared Net each core will tap. Another option is to crack open some connections, since the clustering decision do not change after fabrication.

GPU Stacking also requires isolated wells to avoid unwanted body bias effects between stack levels, thus either Fully-Depleted SOI or triple-well technology are needed [8]. Although this is restrictive, FD-SOI has high availability and triple-well only requires an extra mask during fabrication.

Sign-off of the stacked configuration may also be a concern. However, we note that a corner based approach can be used to determine the possible ranges of power and timing of a stacked system. The procedure should take into account what happens when stacking clusters of different types, i.e., FF and SS, FF and FF, SS and SS. Although this increases the effort for sign-off, it can be trivially implemented in EDA flows. This may be a bit of a pessimistic approach, since it will show worst-case configurations that will likely not happen.

VII. Conclusion

We present GPU Stacking as a method to manage the effects of process variation. We show that the stacking of cores with opposite variations tend to balance the variation effects that would have appeared in a conventional configuration. To maximize the balancing effect, cores with opposite variations should be stacked, which requires post-silicon configurability. We propose a clustering technique to make such a configurability feasible. The homogeneous nature of GPGPU applications make them suitable candidates for GPU stacking. Previous

voltage stacking publications only analyzed multicores and required complex circuitry to stabilize the voltage. This work is the first to use stacking in the context of process variation, and to propose a floating middle rail. We show that the stable nature of GPGPUs allows for the use of only decoupling capacitors to stabilize the power delivery.

This research provides a detailed evaluation of NTC with GPGPUs, and the idea of GPU stacking. We first carefully size a GPU for NTC operation, achieving 43% power savings, with only 4.8% performance degradation. We then apply GPU Stacking to manage process variation, which impacts NTC circuits more than circuit in the super-threshold region. The homogeneous nature of GPGPU architectures and applications, make them a very interesting candidate for exploration in the extreme domains, with both low voltages and small feature sizes. We show that stacking can increase performance under process variation at near threshold, on average, by 37% compared to the traditional (not stacked) configuration, delivering 80% of the performance compared to the no variation (ideal) conditions. Even when using multi-frequency domain, GPU Stacking is able to further improve PV compensation by about 30%. Although this technique is more suited to GPUs due to their homogeneous nature, it could be adapted to use in a scheme similar to CoreUnfolding [9], where authors leverage power consumption correlation among parts of a single core.

References

[1] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudgi, "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits," February 2010.

[2] L. Chang, D. Frank, R. Montoye, S. Koester, B. Ji, P. Coteus, R. Dennard, and W. Haensch, "Practical strategies for power-efficient computing technologies," Proceedings of the IEEE, vol. 98, no. 2, pp. 215–236, 2010.

[3] P. Basu, H. Chen, S. Saha, K. Chakraborty, and S. Roy, "Swiftgpu: Fostering energy efficiency in a near-threshold gpu through a tactical performance boost," in 2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC), June 2016, pp. 1–6.

[4] U. Karpuzcu, K. Kolluru, N. Kim, and J. Torrellas, "Varius-ntv: A microarchitectural model to capture the increased sensitivity of manycores to process variations at near-threshold voltages," in Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on. IEEE, 2012, pp. 1–11.

[5] T. N. Miller, X. Pan, R. Thomas, N. Sedaghati, and R. Teodorescu, "Booster: Reactive core acceleration for mitigating the effects of process variation and application imbalance in low-voltage chips," in High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on. IEEE, 2012, pp. 1–12.

[6] C. Silvano, G. Palermo, S. Xydis, and I. Stamelakos, "Voltage island management in near threshold manycore architectures to mitigate dark silicon," in Proceedings of the conference on Design, Automation & Test in Europe. European Design and Automation Association, 2014, p. 201.

[7] U. R. Karpuzcu, A. Sinkar, N. Sung Kim, , and T. J., "EnergySmart: Toward Energy-Efficient Many cores for Near-Threshold Computing," in Proceedings of the 2013 international symposium on high performance computer architecture. IEEE Computer Society, 2013.

[8] S. Lee, D. Brooks, and G. Wei, "Evaluation of Voltage Stacking for Near-threshold Multicore Computing," in Low Power Electronics and Design (ISLPED), 2012 IEEE International Symposium on. ACM, 2012, pp. 373–378.

[9] E. K. Ardestani, R. T. Possignolo, J. L. Briz, and J. Renau, "Managing mismatches in voltage stacking with coreUnfolding," ACM Trans. Archit. Code Optim., vol. 12, no. 4, pp. 43:1–43:26, Nov. 2015.

[10] Q. Zhang, L. Lai, M. Gottscho, and P. Gupta, "Multi-story power distribution networks for GPUs," in Design, Automation, and Test in Europe (DATE), Proceedings of, Mar 2016.

[11] E. Ebrahimi, R. T. Possignolo, and J. Renau, "SRAM voltage stacking," in 2016 IEEE International Symposium on Circuits and Systems (ISCAS), May 2016, pp. 1634–1637.

[12] W. Kim, D. Brooks, and G. Wei, "A fully-integrated 3-level dc/dc converter for nanosecond-scale dvs with fast shunt regulation," in Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International. IEEE, 2011, pp. 268–270.

[13] J. Gu and C. H. Kim, "Multi-story Power Delivery for Supply Noise Reduction and Low Voltage Operation," in Proceedings of the 2005 International Symposium on Low Power Electronics and Design, ser. ISLPED '05. New York, NY, USA: ACM, 2005, pp. 192–197.

[14] S. Rajapandian, K. Shepard, P. Hazucha, and T. Karnik, "High-voltage Power Delivery Through Charge Recycling," Solid-State Circuits, IEEE Journal of, vol. 41, no. 6, pp. 1400–1410, 2006.

[15] J. Lee, P. Ajgaonkar, and N. Kim, "Analyzing throughput of GPGPUs exploiting within-die core-to-core frequency variation," in Performance Analysis of Systems and Software (ISPASS), 2011 IEEE International Symposium on. IEEE, 2011, pp. 237–246.

[16] H. Wann, C. Hu, K. Noda, D. Sinitsky, F. Assaderaghi, and J. Bokor, "Channel doping engineering of mosfet with adaptable threshold voltage using body effect for low voltage and low power applications," in VLSI Technology, Systems, and Applications, 1995. Proceedings of Technical Papers. 1995 International Symposium on. IEEE, 1995, pp. 159–163.

[17] M. J. Lyons and D. Brooks, "The design of a bloom filter hardware accelerator for ultra low power systems," in Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design, ser. ISLPED '09, 2009, pp. 371–376.

[18] E. Donkoh, T. S. Ong, Y. N. Too, and P. Chiang, "Register file write data gating techniques and break-even analysis model," in Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design, ser. ISLPED '12, 2012, pp. 149–154.

[19] J. Cong, M. A. Ghodrat, M. Gill, B. Grigorian, and G. Reinman, "Charm: a composable heterogeneous accelerator-rich microprocessor," in Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design, ser. ISLPED '12, 2012, pp. 379–384.

[20] J.-H. Choi, J.-H. Lee, S.-W. Jeong, S.-D. Kim, and C. Weems, "A low power tlb structure for embedded systems," IEEE Comput. Archit. Lett., pp. 3–3, 2002.

[21] M. Huang, J. Renau, and J. Torrellas, "A framework for dynamic energy efficiency and temperature management," in Journal on Instruction Level Parallelism, Aug 2002.

[22] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, ser. MICRO 39, 2006, pp. 347–358.

[23] J. Lee, V. Sathisha, M. Schulte, K. Compton, and N. S. Kim, "Improving throughput of power-constrained gpus using dynamic voltage/frequency and core scaling," in Proceedings of the 2011 International Conference on Parallel Architectures and Compilation Techniques, ser. PACT '11, 2011, pp. 111–120.

[24] J. Zhao, G. Sun, G. H. Loh, and Y. Xie, "Energy-efficient gpu design with reconfigurable in-package graphics memory," in Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design, ser. ISLPED '12, 2012, pp. 403–408.

[25] A. M. Gebhart, "Energy-Efficient Mechanisms for Managing On-Chip Storage in Throughput Processors," Ph.D. dissertation, The University of Texas at Austin, May. 2012.

[26] A. Sankaranarayanan, E. K. Ardestani, J. L. Briz, and J. Renau, "An energy efficient gpgpu memory hierarchy with tiny incoherent caches," in International Symposium on Low-Power Electronics and Design, Beijing, China, Sept. 2013.

[27] M. Gebhart, D. R. Johnson, D. Tarjan, S. W. Keckler, W. J. Dally, E. Lindholm, and K. Skadron, "A hierarchical thread scheduler and register file for energy-efficient throughput processors," ACM Trans. Comput. Syst., vol. 30, no. 2, pp. 8:1–8:38, Apr. 2012.

[28] J. Leng, Y. Zu, M. Rhu, M. Gupta, and V. J. Reddi, "Gpuvolt: Modeling and characterizing voltage noise in gpu architectures," in Proceedings of the 2014 International Symposium on Low Power Electronics and Design, ser. ISLPED '14. New York, NY, USA: ACM, 2014, pp. 141–146. [Online]. Available: http://doi.acm.org/10.1145/2627369.2627605

[29] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm early design exploration," Electron Devices, IEEE Transactions on, vol. 53, no. 11, pp. 2816–2823, 2006.

[30] S. Kanev, "Motivating software-driven current balancing in flexible voltage-stacked multicore processors," Ph.D. dissertation, Harvard University Cambridge, Massachusetts, 2012.

[31] E. Ebrahimi, R. T. Possignolo, and J. Renau, "Level shifter design for voltage stacking," in Circuits and Systems (ISCAS), Proceedings of the 2017 IEEE International Symposium on, May 2017.

[32] E. K. Ardestani and J. Renau, "ESESC: A Fast Multicore Simulator Using Time-Based Sampling," in International Symposium on High Performance Computer Architecture, ser. HPCA'19, 2013.

[33] S. Li, J. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in Microarchitecture. 42nd IEEE/ACM Int'l Symp. on. IEEE, 2009, pp. 469–480.

[34] J. Lucas, S. Lal, M. Andersch, M. Alvarez-Mesa, and B. Juurlink, "How a Single Chip Causes Massive Power Bills GPUSimPow: A GPGPU Power Simulator," in IEEE International Symposium on Performance Analysis of Systems and Software, 2013.

[35] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC), ser. IISWC '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 44–54.

[36] J. Stratton, C. Rodrigues, I. Sung, N. Obeid, L. Chang, N. Anssari, G. Liu, and W. Hwu, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," Center for Reliable and High-Performance Computing, 2012.

[37] R. Thomas, K. Barber, N. Sedaghati, L. Zhou, and R. Teodorescu, "Core tunneling: Variation-aware voltage noise mitigation in gpus," in 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA), March 2016, pp. 151–162.

[38] S. K. Khatamifard, M. Resch, N. S. Kim, and U. R. Karpuzcu, "VARIUS-TC: A Modular Architecture-Level Model of Parametric Variation for Thin-Channel Switches," in 2016 IEEE 34th International Conference on Computer Design (ICCD), vol. 00. Los Alamitos, CA, USA: IEEE Computer Society, 2016, pp. 654–661.

[39] S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "VARIUS: A model of process variation and resulting timing errors for microarchitects," Semiconductor Manufacturing, IEEE Transactions on, vol. 21, no. 1, pp. 3–13, 2008.

[40] J. Leng, Y. Zu, and V. J. Reddi, "GPU Voltage Noise: Characterization and Hierarchical Smoothing of Spatial and Temporal Voltage Noise Interference in GPU Architectures," in 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), Feb 2015, pp. 161–173.

[41] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System Level Analysis of Fast, per-Core DVFS Using on-Chip Switching Regulators," in 2008 IEEE 14th International Symposium on High Performance Computer Architecture, Feb 2008, pp. 123–134.

[42] S. Nassif, "Power grid analysis benchmarks," in Design Automation Conference, 2008. ASPDAC 2008. Asia and South Pacific, 2008, pp. 376–381.

[43] S. Khandelwal, J. Duarte, N. Paydavosi, D. Lu, C.-H. Lin, Mohan, S. Yao, T. Morshed, A. Niknejad, and C. Hu, "BSIM-CMG 108.0.0 Multi-Gate MOSFET Compact Model," 2014.

[44] S. Hong and H. Kim, "An integrated GPU power and performance model," SIGARCH Comput. Archit. News, vol. 38, no. 3, pp. 280–289, Jun. 2010. [Online]. Available: http://doi.acm.org/10.1145/1816038.1815998

[45] I. Corporation. (2010) A 400 amp fully integrated silicon voltage regulator with in-die magnetically coupled embedded inductors. Palm Spring, CA.

[46] (2017) NVIDIA GeForce GTX750 Specifications. NVIDIA Corporation. Available on http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-750/specifications.

[47] I. Technologies. (2013) High performance DrMos TDA21220. Available on http://www.infineon.com/cms/en/product/power/dc-dc-converter/dc-dc-integrated-power-stage/drmos-integrated-power-stage/TDA21220/productType.html?productType=db3a3044243b532e0124de3165386adc.

Rafael Trapani Possignolo is a Ph.D. candidate at UC Santa Cruz, working with computer architecture and VLSI design automation. His main research interest lies in techniques and tools to improve design productivity, such as automated pipelining and fast synthesis methods. Rafael is also interested in architecture and circuit level techniques that enable better power and performance tradeoffs, such as voltage stacking. Rafael holds a B.S. in Computer Engineering an M.S. in Electrical Engineering from University of Sao Paulo, an M.S. in Embedded Systems Engineering, and an Engineering degree from École Nationale Supérieure des Techniques Avancees (ENSTA-ParisTech).

Elnaz Ebrahimi is a Ph.D. candidate at UC Santa Cruz, part of Santa Cruz MicroArchitecture (MASC) laboratory. Her primary research is in architecture- and circuit-level SRAM designs that ease the conventional design paradigms. She is also interested in power modeling and power-efficient design of processor microarchitecutres. Elnaz holds an M.S. in Computer Architecture form UC Santa Cruz and a B.S. in Computer Engineering from San Jose State University.

Ehsan K.Ardestani is a CPU Architect and PhD Alumni of Computer Engineering at the University of California, Santa Cruz. His research interests include computer architecture, and simulation methodology. He received his B.S. and M.S. degree in Computer Engineering and Computer Architecture from Isfahan University and Amirkabir University of Technology, Iran, respectively.

Alamelu Sankaranarayanan obtained her Ph.D. in Computer Engineering from UC Santa Cruz in 2016. Her areas of interest include processor microarchitectures, energy efficient design and architectural simulation methodologies.

José Luis Briz is an Associate Professor at the University of Zaragoza (UZ). His research interests include memory hierarchy and processor microarchitecture, embedded systems and real-time scheduling on MPSoCs. He as also contributed to the application of computers to Structural Geology. Briz has a combined BS/MS degree in Geology, a MS degree in Computer Science, and a Ph.D. in Computer Engineering, all of them from UZ. He is a member of the gaZ group, the I3A Research Institute, and Affiliate of the HiPEAC European Network of Excellence. He is also a member of the ACM, the IEEE, and of the Spanish Society of Computer Architecture (SARTECO)

Jose Renau is a professor of computer engineering at the University of California, Santa Cruz (http://masc.soe.ucsc.edu/). His research focuses on computer architecture, including design effort metrics and models, infrared thermal measurements, low-power and thermal-aware designs, process variability, thread level speculation, FPGA/ASIC design, Fluid Pipelines, and Pyrope (a modern hardware description language) and Live flows that aim at productivity in hardware design. Renau has a Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign.