Parameter Variation at Near Threshold Voltage: The Power Efficiency versus Resilience Tradeoff

Josep Torrellas Nam Sung Kim Radu Teodorescu
University of Illinois University of Wisconsin Ohio State University

Technical Report, Department of Computer Science, University of Illinois
March 2012

A Executive Summary

A.1 Problem Addressed

The strongest lever that we have to improve the power-efficiency of CMOS devices is to reduce their supply voltage (V_{DD}). When V_{DD} is only a bit higher than the threshold voltage (V_{TH}), we attain a fairly optimal operation point, where energy per operation is low and switching delay is not too high. This region, known as *Near Threshold Voltage* (NTV), is attractive because it can offer a 1-2 orders of magnitude reduction in power relative to the conventional Super Threshold Voltage (STV) operation [6, 10, 32].

While NTV is appealing, it has three major limitations: lower speed, higher share of static power, and higher impact of process variations. The first two are well known and likely to be worked around — especially for highly-parallel workloads. The third one, however, is less known and harder to tame. Process variations are the deviation of the values of device parameters (such as a transistor's V_{TH}) from their nominal specifications. They lead to circuits and chips with lower speed, higher power consumption and lower resilience. The negative impact of process variations is much higher at NTV: small changes in V_{TH} induce large changes in transistor speed and power consumption due to an intrinsic effect coming from having V_{DD} so close to V_{TH}.

Variations also include parametric changes that occur dynamically, such as variations in V_{DD} or temperature, or due to wearout effects. An NTV environment is also more vulnerable to these because, to attain high power-efficiency, its devices have been designed with tiny guard-bands.

Clearly, if NTV operation is to be mastered, parameter variations at NTV must be understood, mitigated, and tolerated. This is the purpose of our work.

A.2 Why Conventional Approaches are Insufficient

Parameter variations already exist and are handled in current STV technologies. However, current approaches to tackle variations are too timid and lack scalability to be truly effective for a 7nm 1000-core NTV chip. These approaches mainly include Adaptive Voltage Scaling (AVS) and Adaptive Body Biasing (ABB) (e.g., [18, 27, 43, 44] among many others). AVS requires multiple V_{DD} domains using widely-used switching Voltage Regulators (VRs). Unfortunately, in a 1000-core NTV chip, the application of fine-grain AVS would have an exceedingly high design, package, and power cost. On the other hand, the future applicability of ABB is highly questionable, given current technology trends.

In addition, current approaches to tackle variations largely focus on a single layer of the computing stack. By construction, such approaches are conservative because they have to assume worst-case conditions for the other layers. To tackle variation cost-effectively at NTV, we need robust solutions that span multiple layers.

Finally, a further difficulty in coping with variations at NTV is that we lack basic models of how parameter variations impact an NTV chip: all the existing microarchitectural models of variations apply only to STV [19, 29, 31, 39, 40].

A.3 Proposed Approach

This proposal describes an *integrated* approach to mitigate and tolerate parameter variations at NTV, and attain over 10x improvements in power efficiency — hence reaching the

program goal of 75 GFLOPS/watt in 7nm technology generation. Our approach consists of the synergistic operation of several novel, aggressive techniques that cut across the technology, circuits, architecture, and runtime layers. The work also includes a robust variation modeling effort, and the *fabrication of a simple NTV chip* for model validation.

The work is done by three experts with a many-year track record of contributions in process variations, who have complementary expertise and a history of joint publications.

The unique aspects of the proposed work are as follows (Figure 1):

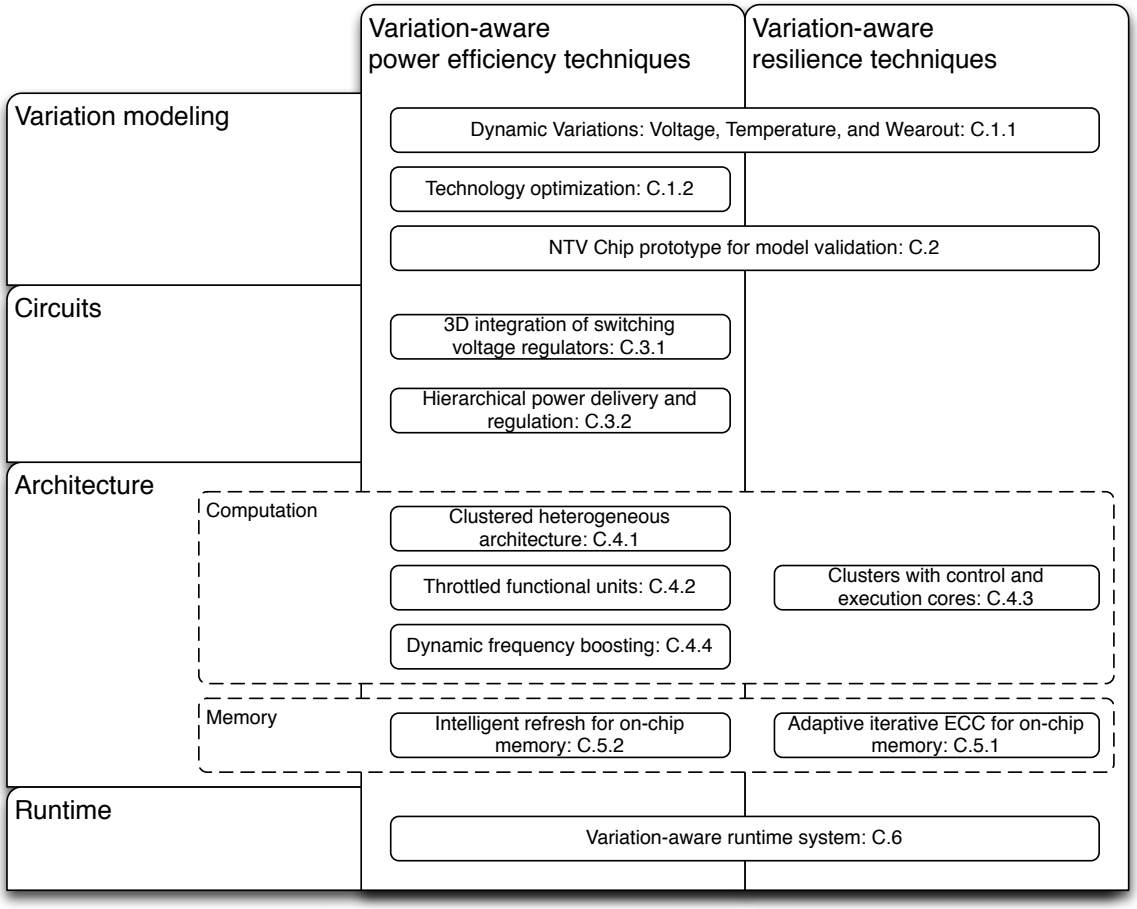


Figure 1: High-level overview of the proposed effort.

- **Variation Modeling**

  We have recently released the first model of process variations for logic and memory at NTV that is usable by microarchitects. In this work, we extend it to include *dynamic variations*, namely voltage, temperature, and wearout-induced variations. The voltage variation model will include support for detecting dangerous voltage droops (*Voltage Emergencies*). As part of the modeling effort, we will identify and select technology parameter values optimized for NTV operation. The models will be integrated, interfaced to an architecture simulator, and widely distributed to the community.

- **NTV Chip Prototype**

  To validate and calibrate our NTV models, we will design, fabricate, and measure a tiled NTV chip. Each tile will contain: (i) a power-gating device to turn on/off the tile, (ii)

chains of logic gates to mimic a core's critical paths, (iii) many 8-transistor SRAM arrays, and (iv) a controller of the tile's switching activity. The chip will allow us to validate our variation models for delay and leakage power for logic and SRAM, voltage, temperature, and wearout.

- **Circuits**
  - **3D-Integration of Switching Voltage Regulators (VRs).** To attain power-efficient operation in a large chip with frequency and power variations, we need to support many V_{DD} and frequency domains. To ensure high VR efficiency, we propose the 3D integration of a die that contains switching VRs, using a higher V_{DD} and a technology suitable for their inductors and analog circuitry.
  - **Hierarchical Power Delivery and Regulation.** Simply supporting many V_{DD} domains is inefficient because a small domain is vulnerable to V_{DD} droops and its VR has to be sized up to provide substantial power when other domains are turned off. To provide scalable, *truly power-efficient* V_{DD} regulation, we propose a hierarchical power delivery structure composed of several 3D-stacked switching VRs and many linear (or low-drop-out (LDO)) VRs. Each 3D-stacked switching VR supports a global V_{DD} domain, where per-core LDO VRs provide per-core local V_{DD} domains. This approach enables fine-grain modulation of each core's V_{DD} and can attain over 90% power efficiency.
- **Architecture: Computation**
  - **Clustered Heterogeneous Architecture.** For power efficiency, the chip exploits the strong spatial correlation of process variations and organizes in clusters. A cluster is the smallest possible frequency and V_{DD} domain. It contains a heterogeneous set of compute engines and memory. To run serial sections fast, one of the cores per cluster is a wide superscalar that can run either at NTV or at high V_{DD}.
  - **Throttled Functional Units.** When a program requires multiple clusters, running it at the frequency of the slowest cluster in the group results in a very power-inefficient execution. Hence, we propose a novel, reconfigurable design of cores that allows their frequency to be raised significantly, while reducing their own throughput. We are sacrificing the performance of an individual core for the benefit of the entire multi-cluster group. The result is increased power efficiency.
  - **Clusters with Control and Execution Cores.** Some Recognition, Mining, and Synthesis (RMS) programs can tolerate some errors, while producing satisfactory results, as long as their control flow is guaranteed correct. To support such inexact codes power-efficiently, our chip can be trivially reconfigured into clusters with control cores (CCs) and clusters with execution cores (ECs). CCs execute the control code at a safe V_{DD} and suffer no errors, while ECs execute the rest at aggressively-low V_{DDs} and can suffer errors. Firewalls ensure CC-EC separation.
  - **Dynamic Frequency Boosting.** The large frequency variation across the chip's clusters may cause imbalance in a program that uses multiple clusters. To improve the program's power-efficiency, we propose to dynamically re-balance performance heterogeneity. The idea is to reconfigure the power-delivery network to support two power-supply rails set at two different NT voltages: a higher one and a lower one. Each cluster can be dynamically connected to either rail, where it runs at a different frequency. By scheduling

clusters that are inherently slow to spend more time on the faster, higher-V_{DD} rail, while those that are fast spend more time on the slower, lower-V_{DD} rail, we eliminate the appearance of frequency variation.

- **Architecture: On-Chip Memory**
  - **Adaptive Iterative ECC for On-Chip Memory.** Given the variations across a large chip, the strength of the error detection and correction codes assigned to on-chip memory lines has to be malleable: slower chip sections must have stronger protection codes, while faster sections can have weaker codes. Hence, we propose a novel scheme where the size of the codes is fixed, and error correction uses an iterative decoding process. Lines in slow chip sections simply *perform more decoding iterations on the codes* than lines in fast sections of the chip. Protection is gradually strengthened as the V_{DD} is lowered.
  - **Intelligent Refresh for On-Chip Memory.** Since, in a large NTV chip, much of the power is due to memory leakage, future implementations will likely use on-chip memories that do not leak, although they may need refresh. Hence, we propose Intelligent Refresh where we aim to *only refresh those locations that we expect to use*. Since, at any time, much of the on-chip memory contains useless data, *the opportunity is great*. Intelligent Refresh refreshes lines on demand, triggered by a per-line Sentry Bit (SB) that interrupts the cache controller before the line decays. We propose various selective refresh algorithms, based on the line's state (e.g., do not refresh clean lines replicated by the cache coherence protocol).

    We attain even *higher power-efficiency by combining* this technique with previous ones. Specifically, we can decrease the refresh rate at the cost of increasing the iterations of the iterative ECC codes. Additionally, we can reduce the refresh rate for clusters with Execution Cores running inexact codes.

- **Runtime**

  While all the previous techniques are hardware-driven by default, they can also be managed with software algorithms. In this case, we can exploit more accurate, higher-level information. In the proposal, we outline how software algorithms can be used to: avoid voltage emergencies; improve the efficiency of switching VRs; divide the work between clusters of Control and Execution cores; determine when the Dynamic Frequency Boosting technique should switch individual clusters from one power rail to the other; and improve the efficiency of Intelligent Refresh by providing information on on-chip memory lines that contain useless data.

A.4 How the Pieces Fit Together

As shown in the vertical axis of Figure 1, we can classify our variation-aware techniques into those that target power efficiency and those that target resilience. Those in the first group either provide V_{DD} and frequency levels that allow the application to run more power-efficiently (despite the large variations across the chip) or reduce the leakage energy. This group includes: (i) the circuits techniques, (ii) the clustered architecture organization, (iii) the throttled functional units, (iv) the dynamic frequency boosting, (v) the intelligent refresh, and (vi) the runtime actuators of these techniques. The group that targets resilience

includes: (i) the clusters with control and execution cores, (ii) the adaptive iterative ECC, and (iii) the runtime actuators for these techniques.

Figure 2 shows our estimated impact of these techniques on the power efficiency of a large embedded NTV chip. If no variation-aware techniques are applied, the power efficiency is likely to be low, possibly around 8 GFLOPS/watt for 7nm. If we apply our variation-aware techniques that target power efficiency, we expect to get a major boost in power efficiency, bringing us to around 50 GFLOPS/watt.

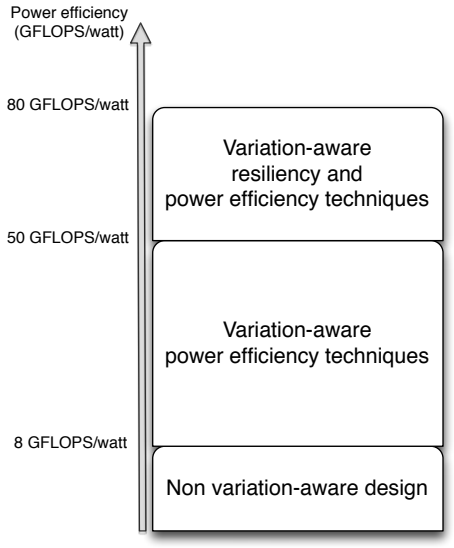


Figure 2: Estimated power efficiency improvements from our proposed solutions.

Most interestingly, if we *combine our techniques* that target power efficiency and those that target resilience, we expect another boost in power efficiency. The reason is that the resilience techniques provide a *safety net* that allows the power-efficiency techniques to aggressively move into an overall more power-efficient operating point. Specifically, the power-efficiency techniques can reduce the V_{DD} (or, in the case of Intelligent Refresh, increase the refresh period) *beyond safe levels* and risk faults if the resilience techniques can guarantee they will tolerate them. Specifically, the iterative ECC can perform additional iterations, and the clusters with Execution Cores can still manage acceptable execution. This combined operation will take us to around 80 GFLOPS/watt.

B Related Work

Previous work has demonstrated the energy efficiency of NTV designs [6, 10, 9, 32, 50]. Architectures designed specifically to take advantage of low NTV properties such as fast caches relative to logic have been proposed by Zhai et al. [50] and Dreslinski et al. [9]. Other work has focused on improving the reliability of large caches in low voltage processors [13, 34]. While significant progress has been made in bringing this technology to market, including a prototype processor from Intel [47], many challenges remain, including reliability and high variation.

Process variation modeling: VARIUS [40] relies on the spherical correlation function, matching empirical measurements at the cost of more computation time than the Agarwal's model [1]. VARIUS extended Mukhopadhyay's access time failure model by adding the impact of systematic variation along with variation in channel length, and replacing the gate delay model by alpha-power law. However, neither the conventional 6-transistor cell, nor the alpha-power law should be used at NTV. Moreover, VARIUS only models access time failures, where other failure modes such as read/write/hold failures become critical at NTV. Finally, VARIUS does not account for leakage currents during timing analysis, and the impact of leakage increases at NTV.

Technology optimization: Bol et al. [4] explored which technology flavor should be selected for sub-threshold voltage logic among existing technologies. However, there is practically no study about what is the general direction of key technology parameter optimization for processors operating at NTV. Furthermore, all previous technology optimizations were performed by analyzing performance and power consumption at the circuit level [11]. However, to maximize power efficiency, we observe that a technology optimization must be performed after considering performance and power consumption at both circuit and architecture levels. Otherwise, the optimization using circuit-level performance and power will lead to sub-optimal power efficiency at the architecture level.

On-chip voltage regulators: Kim et al. [24] analyzed the potential benefit of supporting per-core voltage domains using on-chip VRs for a processor running embedded applications. To analyze the efficiency of the on-chip VRs, they assume that the inductors used for the VRs are mounted on the processor package, while the remaining VR circuits are on the same chip with the processor. However, this approach can only provide a limited number of voltage domains due to the physical constraint associated with the package-mounted inductors. Intel demonstrated an in-silicon VR (ISVR), which includes the on-chip inductors, but it suffers from low efficiency due to the poor quality of the inductors [8]. On the other hand, our approach takes advantage of 3D-stacking technology and existing per-core power-gating devices to provide low-cost and high-efficiency per-core VRs.

Voltage variability and emergencies: Prior work has focused on addressing voltage emergencies in low core count systems. Reddi et al. [38] proposed a solution for eliminating emergencies in single-core CPUs. They employ heuristics and a learning mechanism to predict voltage emergencies from architectural events. When an emergency is predicted, execution rate is throttled, reducing the slope of current changes. Gupta et al. [16] proposed an event guided adaptive voltage emergency avoidance scheme: Recurring emergencies are avoided by initiating various operations such as *pseudo-nops*, prefetching, and a hardware throttling mechanism on events that cause emergencies. Gupta et al. also proposed DeCoR [15], a

checkpoint/rollback solution which allows voltage emergencies but delays the commit of instructions until they are considered safe. A low voltage sensor, of known delay, signals that an emergency is likely to have occurred and the pipeline is flushed and rolled back to a safe state.

Very few previous studies have examined voltage emergencies in multicore chips. Gupta et al. [14] characterize within-die voltage variation using a detailed distributed model of the on-chip power-supply grid. They model a 4-core CMP and use a multi-programmed workload consisting of SPEC applications in their evaluation. We are not aware of any previous work that examines voltage emergencies in chips with more than 4 cores. PI Teodorescu's previous work [35] examined voltage emergencies in CMPs with up to 32 cores. In this work we will focus on modeling voltage variability and eliminating voltage emergencies in chips with very large numbers of cores at NTV.

Error correction algorithms for on-chip memory resilience: Prior approaches to cache error correction have generally focused on much lower error rates than those expected in near-threshold. Some existing microprocessors use simple SECDED-based ECC techniques [2, 36, 37], mostly intended to deal with soft errors at high supply voltages. Researchers have proposed a few cache-based error correction approaches targeted at higher error rates. Sun et al. [42] developed a cache protection solution based on multi-bit ECC (DECTED, Dual Error Correction Triple Error Detection). Overall, the error rate that can be tolerated is about 0.5%, significantly below the error rates in near-threshold. Wilkerson et al. [48] developed a technique for coping with embedded DRAM errors that occur as a result of variation-induced cell leakage. They use both SECDED and BCH codes to protect data lines. At NTV, the multi-bit error rate may be so high that the high-latency BCH correction would add far too much to the average cache access latency.

Yoon et al. [49] devised a method for correcting SRAM errors without storing ECC in dedicated SRAM cells. Instead, ECC bits are stored in cacheable DRAM memory space. In the place of ECC, a much less expensive error detection code is stored in dedicated SRAM. This approach is very efficient because only in the rare event that a line is both dirty and suffers an error must the ECC code be fetched from DRAM or elsewhere in the cache. However, the potentially very high error rates at NTV could impose too high of a demand for DRAM access for this to be applicable to our needs.

Previous work by PI Teodorescu [34] provides an initial evaluation of the strength of iterative decoding when applied to NTV caches. We will build on that work to provide stronger ECCs by experimenting with multiple, stronger base codes to be used in the proposed iterative ECC.

Dual-Vdd designs: Previous work has proposed dual and multi-V_{DD} designs with the goal of improving energy efficiency. Most previous work has focused on tuning the delay vs. power-consumption of paths at fine granularity within the processor. For instance, Kulkarni et al. [25] propose a solution for assigning circuit blocks along critical paths to the higher power supply, while blocks along non-critical paths are assigned to a lower power supply. Liang et al. proposed Revival [30], which uses voltage selection at pipeline stage granularity to reduce the effects of delay variation. Calhoun and Chandrakasan proposed local voltage dithering [5] to achieve very fast dynamic voltage scaling in subthreshold chips. These solutions assign multiple voltages at much finer granularity than in our design, incurring a higher design and verification complexity.

C Technical Approach

C.1 Parameter Variation Models and Technology Optimization

Process variations lead to slower, more power-hungry, and less resilient cores and memory modules. To develop variation-mitigation solutions at NTV, it is essential that we first construct accurate models of parameter variations at NTV. While there are several microarchitectural models of process variations (e.g., [19, 29, 31, 39, 40]), they do not accurately capture the NTV environment.

We have recently developed models of process variations at NTV (*VARIUS-NTV* [21] by Kim and Torrellas) and of memory design at NTV (*CACTINTV* [45], by Teodorescu) that are usable by microarchitects. These models follow our previous releases of variation models and tools at conventional voltages (*VARIUS* [40] by Teodorescu and Torrellas, and *Facelift* [46] by Tiwari and Torrellas). In this work, we will: (i) extend these models to include *dynamic variations* (voltage, temperature, and wearout), (ii) identify and select parameter values optimized for NTV operation, and (iii) validate these models with measurements of a simple chip that we will fabricate (Section C.2).

C.1.1 Dynamic Variations: Voltage, Temperature, and Wearout

To attain high energy and power efficiency, NTV circuits will have small guard-bands. Consequently, dynamic variations in voltage, temperature, and wearout will have a significant impact on power efficiency, resilience, or both.

We will extend our framework to model voltage variations due to changes in manycore load. Unfortunately, it takes too long to simulate a detailed circuit-level power-grid netlist using SPICE. Hence, we will build an architectural power-grid model that captures voltage variations. It will be modeled as a linear system, and will interface to an architectural simulator that gives us the cycle-accurate power consumption of the manycore. The power-distribution model will be coupled with voltage regulator (VR) models that will provide data on VR response under load changes. The VR models will be based on existing VR designs or on our own Low-Drop-Out (LDO) VR designs (Section C.3). The power-distribution model will be validated with SPICE simulations.

To model temperature variations, we will modify the existing Hotspot thermal simulator [41] to be compatible with our NTV parameters. We will integrate it into our framework to capture temporal and spatial temperature variations during the execution of the workload. Finally, to model wearout, we will extend our Facelift aging model [46] to be applicable to NTV environments.

All these models will be integrated together into the architectural simulator. This is because dynamic variations can be captured only through execution of a workload on a specific chip implementation. These models will be usable by architects and made public domain.

C.1.2 Selecting Parameter Values Optimized for NTV Operation

To attain high energy efficiency at NTV, we will optimize the technology parameters (e.g., V_{TH}, L_{EFF}, and T_{OX}) in two ways. First, we will optimize their values to yield the best performance and power trade-off at NTV — rather than at STV, as current systems do. This is because a technology is optimal for a relatively narrow range of V_{DD}. To see the importance of such choice, consider two common technology choices for a given technology

node, namely high-performance or low-power. At 16nm, the minimum energy point (power delay product) of the low-power technology is 1.65x lower than the minimum energy point of the high-performance one. The V_{DD} values that attain these minimums are 15% apart.

Second, we will optimize parameter values to yield their optimal delay/power trade-off point *at the architecture level*, rather than at the device level. We showed in [26] that such points are different because frequency increases do not always translate directly into throughput increases at the architecture level. Hence, we will optimize technology parameters with an iterative process requiring both device- and architecture-level evaluations. For this, we will develop a novel framework that integrates the device-level delay and power models into our simulator for manycore architecture performance.

C.2 NTV Chip Prototype for Model Validation

To validate and calibrate our NTV variation models, we plan to design, fabricate, and measure a simple chip. The chip will be composed of "dummy core" tiles (e.g., 10x10 tiles). Each tile will contain (i) a power-gating device that can turn on/off the tile, (ii) chains of logic gates that mimic critical paths in a core, (iii) many 8-transistor SRAM arrays, and (iv) a controller for the switching activity in the tile.

The power-gating device will be used to turn on and off the tile, forcing dynamic voltage variations in the power grid. This will allow us to validate the voltage variation model. The chains of logic gates will model the logic in a processor pipeline. They will allow us to validate our delay and leakage power variation models. Including a fully-functional core for validation would be a time-consuming task with little value added for the model validation. Using similar chains of gates has been used to detect and mitigate the impact of variations in the IBM Power 7 processor [28].

The conventional 6-transistor SRAM cell does not operate reliably at NTV and, therefore, for caches and other structures, we must use new cells such as the 8-transistor one. These are the SRAM cells modeled by our VARIUS-NTV [21]. Each tile will include a large number of 8-transistor SRAM arrays. They will allow us to validate our SRAM variation model, especially the distribution of the critical V_{NTMIN} parameter, which is the cell's minimum operating voltage at NTV.

Finally, the controller for switching activity in the tile will allow us to change the dynamic power consumption. This will affect voltage variations, temperature across the chip, and wearout. As a result, we can validate the voltage, temperature, and wearout models.

C.3 Circuit Design

C.3.1 3D-Integration of Switching Voltage Regulators

Given the range of frequency and power consumption variations observed across a large NTV chip, it is necessary to apply different V_{DDs} and frequencies to different parts of the chip. The conventional approach to supporting several V_{DD} domains is to use several on-chip switching Voltage regulators (VR). This is because, as we point out in [12], supporting several off-chip switching VRs is not feasible: the current capacity of individual domains has to be sized quite high, for the case when the domain is the only one turned on in the chip. As a result, the pins are unlikely to support such current.

Unfortunately, on-chip switching VRs incur a large power loss (25-30%) and consume a large chip area ($\approx 110mm^2$) [8]. The reason is that they require high-quality on-chip in-

ductors, which are hard to integrate with cores in the same chip. This large power loss is unacceptable in a highly energy-efficient NTV environment, while the area consumption is also undesirable.

In this work, we propose to explore the 3D integration of a die that contains switching VRs with the processor die. This 3D integration allows us to use a technology that is different in each die. The die with VRs will be built using a higher V_{DD} and a technology suitable for implementing analog circuits and their inductors. The result will be high VR efficiency — as if we had off-chip switching VRs.

C.3.2 Hierarchical Power Delivery and Regulation

Simply supporting many V_{DD} domains using 3D-integrated switching VRs is not enough at NTV. There are two main challenges. The first one is that a small V_{DD} domain has little capacitance and changes in load can create major voltage droops. To tolerate this effect, we need to add more V_{DD} guard-band, which is inefficient. The second challenge is the fact that the switching VR assigned to one of these domains still has to be sized up to provide substantial power — in the cases when many of the other V_{DD} domains are shut off [12, 20]. Hence, the system is not very cost-effective.

To solve this problem, in this work we propose to explore a novel *hierarchical* power delivery structure. It is composed of several 3D-stacked switching VRs and several linear (or *low-drop-out (LDO)*) VRs. We envision 3D-stacked switching VRs to provide 4-8 *global* V_{DD} domains. Each global V_{DD} domain is connected to a group (cluster) of cores. Inside one such group, per-core LDO VRs provide per-core local V_{DD} domains, and can modulate the core's V_{DD}. Note that these local V_{DD} domains are not completely isolated electrically because they are connected through LDO VRs to other local domains. As a result they do not suffer the small-capacitance problem of switching VR-regulated V_{DD} domains. Our idea is illustrated by Figure 3.

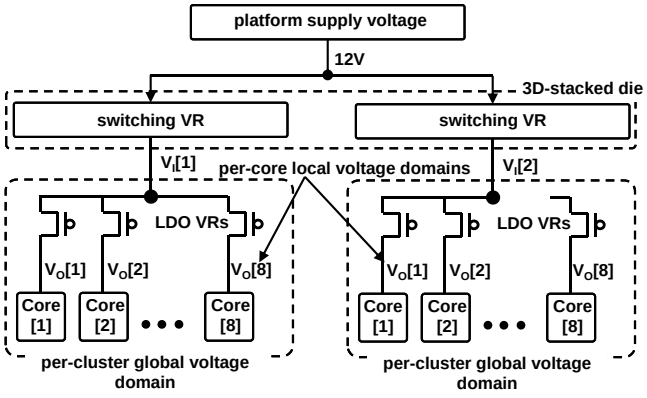


Figure 3: Proposed hierarchical VR scheme.

Since each core in a chip already has a per-core power-gating (PCPG) device for power management, we can implement LDO VRs at a very low cost by reusing that logic. Specifically, the output device of an LDO and its buffers can be shared with a PCPG device. We showed in [12] that these two components account for 83% of the total LDO VR area. In addition, the output voltage provided by an LDO can be easily modulated dynamically by changing the resistance of the device [23]. Finally, and most importantly, our analysis shows

that LDOs in a manycore processor executing multithreaded workloads attain a very high power efficiency ($> 90\%$) [12].

While our design of Figure 3 enables enormous power-efficiency opportunities in an NTV chip, it opens up multiple research questions that we will address in our work:

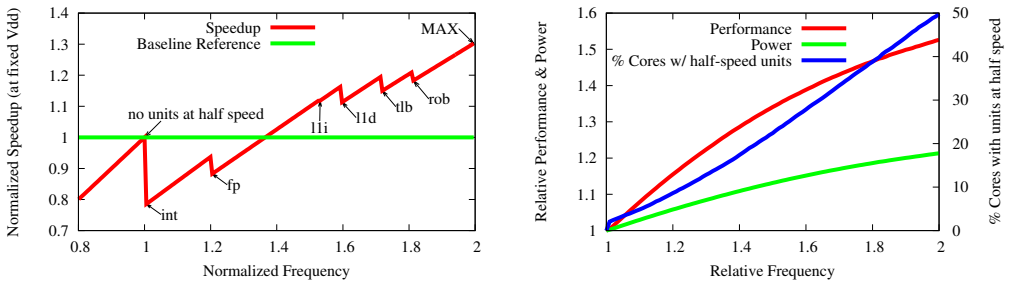
- What is the most effective hierarchy in terms of number of global V_{DD} domains? There is a trade-off between (i) the power-efficiency of the LDOs and (ii) the V_{DD} guard-band size and the implementation cost. Specifically, with more global V_{DD} domains, each of them will contain fewer local V_{DD} domains. This leads to smaller difference across local V_{DD}s and thus higher efficiency for LDO VRs. On the other hand, there is less capacitance per global V_{DD} domain and thus we need more V_{DD} guard-band. In addition, there is a higher implementation cost. The opposite occurs with fewer global V_{DD} domains.
- How do we maximize the overall performance and energy-efficiency of the cores, while minimizing the power loss due to global and local VRs? This requires jointly optimizing the values of the global and local V_{DD}s.
- What is the most effective way to control the resistance of the output device of LDO VRs? LDOs regulate voltage by modulating the resistance of the output devices using analog feedback circuitry. Such analog circuitry complicates the design and manufacturing-testing process at NTV, as well as impacting the yield.
- What is the most cost-effective granularity for V_{DD} modulation? Modulating the V_{DD} of each core individually with an LDO can maximize the benefit of multiple local V_{DD} domains. However, controlling the V_{DD} of a group of cores at a time is less expensive in terms of testing and tuning. We will also explore how the optimal design point is impacted by the decap placement.
- How do we optimize the distribution of the V_{DDs} in a 3-D die structure? This is a complicated problem that requires that we jointly optimize the placement of the VRs, LDOs, and decaps in the multi-plane structure.

C.4 Chip Architecture: Computation

C.4.1 Clustered, Heterogeneous Architecture

The large number of cores in a variation-afflicted NTV chip can only be organized energy-efficiently in clusters. A cluster exploits the strong spatial correlation of within-die process variations by running all the cores at the same frequency and V_{DD}. A cluster is the smallest possible frequency and V_{DD} domain. It contains a heterogeneous set of compute engines (general-purpose cores, semi-specialized cores and accelerators) plus memory (both a cluster memory and per-core local memory).

To be able to run sequential applications and serial sections fast, one of the cores in each cluster is a wide superscalar that is able able to run either at NTV or at high voltage (i.e., STV). In the latter case, the rest of the cores in the cluster are turned off. In sequential applications or in serial sections, the active or critical threads are scheduled on the wide cores running at STV. Finally, to save energy and to eliminate the slowest parts of the chip under process variations, we are able to disable individual cores or parts of cores.



(a) Single core with throttling (b) Average across 600 many-cores

Figure 4: Performance versus frequency under functional unit throttling.

C.4.2 Throttled Functional Units

Variation at NTV creates a wide distribution of core frequencies across the chip. Since clusters or groups of clusters are set to run at the frequency of the slowest core in the group, low-frequency cores have a big impact on the frequency of the group. Hence, improving the frequency of these critical cores can have a big impact on frequency.

We propose an aggressive design of cores that allows their frequency to be raised significantly, while trading off other metrics such as throughput, functional integrity or resilience. In one design, functional units (FUs) have two possible speeds: full speed (running at the core's frequency) and half speed (running at half the core's frequency). When a core is the slowest in the cluster, its slowest FUs can switch to half speed (double the cycles per operation), allowing the frequency of the core (and cluster) to be increased substantially.

Figure 4(a) shows the effect of halving the frequency of individual FUs in a core on the performance of a SPEC benchmark. Configuring some slow FUs (int and fp) to run at half-speed allows core (and cluster) frequency to be raised substantially, resulting in performance improvement. Figure 4(b) shows the performance increase of a 64-core cluster as we raise the frequency of its slow cores. Even if the performance of one or more cores is reduced, the loss is more than offset by the overall frequency increase. We are willing to sacrifice the performance of individual cores for the benefit of the entire cluster.

C.4.3 Clusters with Control and Execution Cores

A large NTV chip is an ideal platform to run highly-parallel throughput applications. An important class of such codes are those in Recognition, Mining, and Synthesis (RMS). RMS codes are data intensive, with kernels that mostly execute probabilistic mathematical models. Much of their computation can tolerate some errors, while producing satisfactory outcomes, as long as the correctness of the control flow is guaranteed.

To support such inexact codes energy-efficiently, our chip can be trivially reconfigured into clusters with control cores (CC) and clusters with execution cores (EC). Clusters with CCs execute the control code. They do not suffer errors because they run at a safe V_{DD} level. Clusters with ECs execute the computation code and, since they run at an aggressively low V_{DD}, can suffer errors. In this work, we will explore the architecture and protection mechanisms needed to support these EC and CC clusters.

C.4.4 Dynamic Frequency Boosting

The large frequency variation across the clusters of an NTV chip may cause imbalance in a parallel application that uses multiple clusters. To avoid this problem, we propose a simple, low-overhead framework for dynamically re-balancing performance heterogeneity. The idea is to reconfigure the power delivery network to support two power supply rails set at two different NT voltages: a higher one and a lower one. Each cluster can be dynamically connected to either of the two power rails using a power gating circuit. This allows each cluster to rapidly switch between two different maximum frequencies. An on-chip controller determines when individual clusters are switched from one rail to the other and how much time they spend on each rail. A "boost budget" restricts how many clusters can be assigned to the high V_{DD} rail at the same time, subject to power constraints.

This framework can eliminate the appearance of frequency variation by scheduling clusters that are inherently slow to spend more time on the faster, higher-V_{DD} rail, while those that are fast spend more time on the slower, lower-V_{DD} rail. A schedule can be chosen such that frequencies for all clusters executing the application average the same value over a finite interval. The result is a high-variation chip that attains performance homogeneity.

An initial evaluation of this framework [33] shows that it can virtually eliminate core-to-core frequency variation in NTV chips. In this work we will apply boosting at the granularity of clusters, which have many times the capacitance of single cores. This might increase the time required to safely switch between voltage rails. We will examine approaches to reducing this overhead.

C.5 Chip Architecture: On-Chip Memory

Ensuring power-efficient resilience for on-chip memory structures such as caches and local memories is a key issue at NTV. The reason is two-fold: memory cells are particularly vulnerable to parameter variations and memory accounts for a large fraction of chip area. Hence, we propose two novel techniques: Adaptive Iterative ECC and Intelligent Refresh.

C.5.1 Adaptive Iterative ECC for On-Chip Memory

At NTV, on-chip memory needs to be protected with error detection and correction codes because the low voltage makes memory cells much more error-prone. However, given the variations across chip, the strength of the codes has to be adaptable. Specifically, sections of the chip that have more errors require stronger error correction cores while sections with fewer or no errors can have weaker codes. Finally, codes should take little space, since they consume leakage power.

We propose to devote some small section of the cache capacity to storing error correction information, and use an iterative decoding process to perform error correction when needed. The technique relies on a generalization of turbo product codes [17]. The idea is to apply a simple error correction code *repeatedly* to a data block to achieve an exponential increase in error correction capability. Our adaptation is simple and requires no storage reconfiguration: it simply means applying more decoding iterations to lines that suffer larger numbers of errors and fewer decoding iterations to the others. It needs no a priori knowledge of the vulnerability of a line. While applying this iterative decoding process to the protection information takes longer than traditional schemes, we pay-as-we-need (depending on the variation parameters), while keeping the storage overhead low.

Our initial evaluation of the iterative approach using very simple codes [34] shows that we can provide significantly stronger error correction than existing solutions targeting high error rates [7, 22], for the same or lower parity storage overhead. We will explore more advanced error correction codes as our building blocks. We will perform an analysis to find the optimal tradeoff between correction strength, protection storage overhead and decoding speed. We will examine efficient hardware implementations that take little die area and minimize decoding and encoding time.

C.5.2 Intelligent Refresh for On-Chip Memory

In a large NTV chip, much of the power consumed is due to leakage. Morever, since most of the chip area is devoted to memory structures, most of the leakage comes from memories. Hence, we expect that future NTV chips will use memory technologies that do not leak. Any such technology has to be compatible with processor logic, and will likely need refresh – e.g., a form of advanced eDRAM. In this proposal, we assume that such low-leakage technology exists and show how to optimize the energy spent on refresh.

We only need to spend power refreshing those locations that we will use. It is well known that, at any time, much of the data in on-chip memory locations is useless. Hence, eliminating the refresh to all such locations presents a major opportunity. It helps that, as the program references a cache line, the hardware automatically refreshes the line. In this proposal, we propose *Intelligent Refresh*, to refresh only the lines that we are likely to use and bring power consumption in on-chip memories to the very minimum.

We propose to refresh lines on demand. We add a Sentry Bit (SB) to each line. We design the SB so that it decays before the data in its associated line decays. When an SB is about to decay, it interrupts the cache controller, which refreshes the line and the SB. When a line is accessed, its SB is also refreshed. Note that process variations affect the rate at which different lines decay. However, since the SB is physically located with the data it protects, both will be affected by process variations in a similar way. As a result, some parts of the chip will refresh more frequently that others. We expect major power reductions through selective refresh as follows:

- Selective refresh based on the state of the line. A simple approach is to mask interrupts from lines where the Valid bit is zero. The Valid bit can be cleared in hardware (by the cache coherence protocol) or in software (if the program knows what data is stale). More advanced algorithms involve not refreshing any clean line that the cache coherence protocol has replicated on multiple L1 caches, or not to refresh any non-dirty line.
- Trade-off the refresh rate for the strength of the ECC codes. We can refresh less frequently and save power, but we will have more errors and will need more iterations on the protection codes (Section C.5.1).
- Use different refresh rates for clusters with Control or Execution Cores. Since EC clusters are tolerant to errors, we can reduce their refresh rate and still keep weak protection codes.

C.6 Variation-aware Runtime System

The circuit and architecture knobs described in the previous sections should be driven by intelligent algorithms, so that the chip strikes the optimal balance between power efficiency and resilience. By default, we have assumed that such algorithms are hardware-based. However, software-driven actuation of these knobs opens up the opportunity of

utilizing more accurate, higher-level information. Consequently, as part of our work, we will explore software support for controlling the following five aspects described before:

- One of the great challenges of NTV operation is the increased sensitivity to voltage fluctuations. These fluctuations are caused by abrupt changes in power demand triggered by load changes. If the voltage deviates too much from its nominal value, it can lead to so-called "voltage emergencies," which can cause timing and memory retention errors. To prevent these emergencies, we will design software-based algorithms that stagger workload scheduling and power management activities, in such a way as to avoid large power spikes. By smoothing out the power demand across cores in a voltage domain, we will reduce the amplitude of the voltage fluctuations, allowing much smaller V_{DD} guard-bands.
- The power efficiency of switching VRs varies significantly (35%-85%) with the magnitude of the current drawn, which is proportional to the workload activity in the voltage domain. The load at which the VR attains the maximum power efficiency is specified in the VR data sheets. Both higher and lower loads result in lower power efficiency. Consequently, we will develop workload scheduling algorithms to be incorporated in a runtime to keep the power efficiency of VRs high. For example, they may involve consolidating several workloads from lightly-loaded domains into a single domain that can work at the optimal load.
- Another runtime support involves assigning the parts of an application (or full applications) that cannot tolerate errors to clusters with Control Cores (CCs), while those that can are assigned to clusters with Execution Cores (ECs). Too many errors in the ECs may impact the quality of the final results (i.e., the Quality of Computing (QoC)). The runtime system has to monitor that the QoC is above a given threshold, and increase the V_{DD} if it is not. For these experiments, we will use the PARSEC applications [3], which are descendants from RMS applications initially developed at Intel. When the Test and Verification contractor (TAV) provides the embedded DoD applications of interest in this program, we will use them.
- Runtime support is also useful to control the Dynamic Frequency Boosting architecture. Runtime system software will determine when individual clusters are switched from one power rail (and frequency) to the other and how much time they spend on each rail. A "boost budget" restricts how many clusters can be assigned to the high voltage rail at the same time, subject to power constraints and optimization objectives. The goal is that the average frequency of all clusters that participate in a parallel job is the same over a fixed time interval.
- Intelligent refresh can also benefit from runtime support. Specifically, the application can provide information about on-chip memory lines that contain data that will not be used. These lines do not need to be refreshed. For example, the application knows about temporary data that is now stale, or structures that need to be initialized first.

D Bibliography

References

[1] A. Agarwal et al. Path-based statistical timing analysis considering inter- and intra-die correlations. In *ACM Int. Workshop Timing Issues Specification Synthesis Digital Systems (TAU)*, pages 16–21, June 2002.

[2] H. Ando, K. Seki, S. Sakashita, M. Aihara, Kan, and K. Imada. Accelerated testing of a 90nm sparc64 v microprocessor for neutron ser. *IEEE Workshop on Silicon Errors in Logic - System Effects (SELSE)*, 2007.

[3] C. Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, January 2011.

[4] D. Bol, D. Flandre, and J.-D. Legat. Technology flavor selection and adaptive techniques for timing-constrained 45nm subthreshold circuits. In *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, ISLPED '09, pages 21–26, 2009.

[5] B. Calhoun and A. Chandrakasan. Ultra-dynamic voltage scaling (UDVS) using subthreshold operation and local voltage dithering. 41(1):238–245, January 2006.

[6] L. Chang, D. J. Frank, R. K. Montoye, S. J. Koester, B. L. Ji, P. W. Coteus, R. H. Dennard, and W. Haensch. Practical strategies for power-efficient computing technologies. *Proceedings of the IEEE*, 98(2):215–236, February 2010.

[7] Z. Chishti, A. R. Alameldeen, C. Wilkerson, W. Wu, and S.-L. Lu. Improving cache lifetime reliability at ultra-low voltages. In *International Symposium on Microarchitecture*, 2009.

[8] J. T. DiBene, P. R. Morrow, C. M. Park, H. W. Koertzen, P. Zou, F. Thenus, X. Li, S. W. Montgomery, E. Stanford, R. Fite, and P. Fisher. A 400 amp fully integrated silicon voltage regulator with in-die magnetically coupled embedded inductors. In *Proceedings of The Applied Power Electronics Conferences and Exposition*, 2010.

[9] R. G. Dreslinski, G. K. Chen, T. Mudge, D. Blaauw, D. Sylvester, and K. Flautner. Reconfigurable energy efficient near threshold cache architectures. In *International Symposium on Microarchitecture*, pages 459–470, December 2008.

[10] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge. Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits. *Proceedings of the IEEE*, 98(2):253–266, February 2010.

[11] D. Frank, L. Chang, and W. Haensch. Improving the energy/power constraint for technology optimization. In *Electron Devices Meeting (IEDM), 2011 IEEE International*, pages 15.2.1 –15.2.4, 2011.

[12] H. Ghasemi, A. Sinkar, M. Schulte, and N. S. Kim. Cost-effective power delivery for supporting per-core voltage domains for power-constrained processors. In *Proceedings of the 49th ACM/IEEE Design Automation Conference*, page To appear, 2012.

[13] H. R. Ghasemi, S. Draper, and N. S. Kim. Low-voltage on-chip cache architecture using heterogeneous cell sizes for multi-core processors. In *International Symposium on High-Performance Computer Architecture*, pages 38–49, February 2011.

[14] M. S. Gupta, J. Oatley, R. Joseph, G.-Y. Wei, and D. Brooks. Understanding voltage variations in chip multiprocessors using a distributed power-delivery network. In *Design Automation and Test in Europe*, pages 1–6, April 2007.

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

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

[17] Y. He and P. Ching. Performance evaluation of adaptive two-dimensional turbo product codes composed of hamming codes. In *International Conference on Integration Technology*, pages 103–107, March 2007.

[18] S. Herbert and D. Marculescu. Characterizing chip-multiprocessor variability-tolerance. In *IEEE/ACM Design Automation Conf. (DAC)*, 2008.

[19] E. Humenay, D. Tarjan, and K. Skadron. Impact of process variations on multicore performance symmetry. In *IEEE Conf. on Design, Automation and Test in Europe (DATE)*, pages 1653–1658, 2007.

[20] N. James, P. Restle, J. Friedrich, B. Huott, and B. McCredie. Comparison of split-versus connected-core supplies in the power6 microprocessor. In *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pages 298 –604, feb. 2007.

[21] U. Karpuzcu, K. Kolluru, N. Kim, and J. Torrellas. Varius-ntc: Microarchitectural model of process variation for neat-threshold voltage computing. In *Dependable Systems Networks (DSN), 2012 IEEE/IFIP 42st International Conference on*, jun. 2010.

[22] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. Hoe. Multi-bit error tolerant caches using two-dimensional error coding. In *International Symposium on Microarchitecture*, pages 197–209. IEEE Computer Society, 2007.

[23] N. S. Kim, J. Seomun, A. Sinkar, J. Lee, T. H. Han, K. Choi, and Y. Shin. Frequency and yield optimization using power gates in power-constrained designs. In *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, pages 121–126, 2009.

[24] 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 *International Symposium on High-Performance Computer Architecture*, pages 123–134, February 2008.

[25] S. Kulkarni, A. Srivastava, and D. Sylvester. A new algorithm for improved VDD assignment in low power dual VDD systems. In *International Symposium on Low Power Electronics and Design*, pages 200–205, May 2004.

[26] J. Lee et al. Workload-adaptive process tuning strategy for power-efficient multi-core processors. In *IEEE/ACM Int. Symp on Low Power Electronics and Design (ISLPED)*, pages 103–114, August 2010.

[27] J. Lee and N. S. Kim. Optimizing total power of many-core processors considering voltage scaling limit and process variations. In *IEEE/ACM Int. Symp. on Low Power Electronics and Design (ISLPED)*, pages 201–206, 2009.

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

[29] X. Liang and D. Brooks. Mitigating the impact of process variations on processor register files and execution units. In *IEEE/ACM Int. Symp on Microarchitecture (MICRO)*, pages 504–514, 2006.

[30] X. Liang, G.-Y. Wei, and D. Brooks. ReVIVaL: A variation-tolerant architecture using voltage interpolation and variable latency. *IEEE Micro*, 29(1):127–138, 2009.

[31] D. Marculescu and E. Talpes. Variability and energy awareness: a microarchitecture-level perspective. In *IEEE/ACM Design Automation Conf. (DAC)*, pages 11–16, 2005.

[32] D. Markovic, C. C. Wang, L. P. Alarcon, T.-T. Liu, and J. M. Rabaey. Ultralow-power design in near-threshold region. *Proceedings of the IEEE*, 98(2):237–252, February 2010.

[33] T. 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 *International Symposium on High-Performance Computer Architecture*, pages 27–38, February 2012.

[34] T. Miller, R. Thomas, J. Dinan, B. Adcock, and R. Teodorescu. Parichute: Generalized turbocode-based error correction for near-threshold caches. In *International Symposium on Microarchitecture*, pages 351–362, December 2010.

[35] T. Miller, R. Thomas, X. Pan, and R. Teodorescu. VRSync: Characterizing and eliminating synchronization-induced voltage emergencies in many-core processors. In *International Symposium on Computer Architecture*, June 2012.

[36] J. Mitchell, D. Henderson, and G. Ahrens. IBM POWER5 processor-based servers: A highly available design for business-critical applications. *IBM Technical Report*, 2006.

[37] N. Quach. High availability and reliability in the itanium processor. *IEEE Micro*, 20(5):61–69, 2000.

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

[39] B. F. Romanescu, S. Ozev, and D. J. Sorin. Quantifying the impact of process variability on microprocessor behavior. In *Workshop on Architectural Reliability*, December 2006.

[40] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. VARIUS: A model of parameter variation and resulting timing errors for microarchitects. *IEEE Transactions on Semiconductor Manufacturing*, 21(1):3–13, February 2008.

[41] K. Skadron et al. Temperature-aware microarchitecture. In *IEEE/ACM Int. Symp. on Computer Architecture (ISCA)*, pages 2–13, 2003.

[42] H. Sun, N. Zheng, and T. Zhang. Realization of l2 cache defect tolerance using multi-bit ecc. In *Defect and Fault Tolerance of VLSI Systems*, pages 254–262, October 2008.

[43] R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. Mitigating parameter variation with dynamic fine-grain body biasing. In *International Symposium on Microarchitecture*, pages 27–39, December 2007.

[44] R. Teodorescu and J. Torrellas. Variation-aware application scheduling and power management for chip multiprocessors. In *IEEE/ACM Int. Symp on Computer Architecture (ISCA)*, June 2008.

[45] R. Thomas and R. Teodorescu. CACTINTV: A tool for modeling access delay and energy of near-threshold SRAM memory. Submitted for publication.

[46] A. Tiwari and J. Torrellas. Facelift: Hiding and slowing down aging in multicores. In *International Symposium on Microarchitecture*, pages 129–140. IEEE Computer Society, 2008.

[47] S. Vangal. A solar powered IA core? No way! Research@Intel, September 2011. http://blogs.intel.com/research/2011/09/ntvp.php.

[48] C. Wilkerson, A. R. Alameldeen, Z. Chishti, W. Wu, D. Somasekhar, and S.-L. Lu. Reducing cache power with low-cost, multi-bit error-correcting codes. In *International Symposium on Computer Architecture*, 2010.

[49] D. Yoon and M. Erez. Memory mapped ECC: Low-cost error protection for last level caches. *ACM SIGARCH Computer Architecture News*, 37(3):116–127, 2009.

[50] B. Zhai, R. G. Dreslinski, D. Blaauw, T. Mudge, and D. Sylvester. Energy efficient near-threshold chip multi-processing. In *International Symposium on Low Power Electronics and Design*, pages 32–37, August 2007.