VARIUS-NTV: A Microarchitectural Model to Capture the Increased Sensitivity of Manycores to Process Variations at Near-Threshold Voltages [‡]

Ulya R. Karpuzcu*, Krishna B. Kolluru†, Nam Sung Kim†, and Josep Torrellas*
*University of Illinois Urbana-Champaign †University of Wisconsin Madison
{rkarpu2,torrella}@illinois.edu {kkolluru,nskim3}@wisc.edu

***Abstract*—Near-Threshold Computing (NTC), where the supply voltage is only slightly higher than the threshold voltage of transistors, is a promising approach to attain energy-efficient computing. Unfortunately, compared to the conventional Super-Threshold Computing (STC), NTC is more sensitive to process variations, which results in higher power consumption and lower frequencies than would otherwise be possible, and potentially a non-negligible fault rate.**

To help address variations at NTC at the architecture level, this paper presents the first microarchitectural model of process variations for NTC. The model, called *VARIUS-NTV*, extends the existing VARIUS variation model. Its key aspects include: (i) adopting a gate-delay model and an SRAM cell type that are tailored to NTC, (ii) modeling SRAM failure modes emerging at NTC, and (iii) accounting for the impact of leakage in SRAM models. We evaluate a simulated 11nm, 288-core tiled manycore at both NTC and STC. The results show higher frequency and power variations within the NTC chip. For example, the maximum difference in on-chip tile frequency is ≈2.3x at STC and ≈3.7x at NTC. We also validate our model against an experimental chip.

***Keywords*-Process variations, Near-threshold voltage, Many-core architectures, SRAM fault models, Power constraints.**

I. Introduction

Power or energy consumption is typically the primary concern in today's computer platforms, ranging from datacenters to handhelds. The main reason for their importance is that CMOS technology has long ago stopped scaling close to perfectly and, as a result, power density increases significantly with each technology generation. If we are to continue delivering scalable computing performance, we need to find new ways to compute more energy- and power-efficiently.

One way to attain higher energy efficiency is to reduce the supply voltage (V_{dd}) to a value only slightly higher than a transistor's threshold voltage (V_{th}). This environment is called Near-Threshold Computing (NTC) [7], [13], [28] — as opposed to conventional Super-Threshold Computing (STC). V_{dd} is a most powerful knob because it impacts both dynamic and static energy super-linearly. Current indications suggest that NTC can decrease the energy per operation by several times over STC [7], [13]. A drawback is that it imposes a frequency reduction, which may be tolerable through more parallelism in the application. For parallel loads, since more cores can be running concurrently within the chip's power envelope, the result is a higher throughput.

A roadblock for NTC is its higher sensitivity to process variations — i.e., the deviation of device parameters from their nominal values. Already in current-technology STC multicores, process variations result in noticeable differences in power and performance across the different cores of a chip [11]. At NTC, due to the low operating V_{dd} [28], the same amount of process variations causes a substantially larger impact on transistor speed and power consumption variations. Process variations are undesirable because they result in chips that consume more static power, cycle at lower frequencies, and can even be faulty.

Process variations should be addressed at multiple levels, including at the computer architecture level. To confront variations at the architecture level, we first need models of process variations and how they affect frequency and power, at a level of abstraction that is useful to microarchitects. Such models exist for STC (e.g., [20], [25], [27], [35], [37]). Unfortunately, none of them is applicable to NTC — NTC uses new memory structures and requires new delay and power models.

This paper presents the first microarchitectural model of process variations for NTC. The model, called *VARIUS-NTV*, extends the existing VARIUS variation model [37]. It models how variation affects the frequency attained and power consumed by cores and memories in an NTC manycore, and the timing and stability faults in SRAM cells at NTC. The key aspects include: (i) adopting a gate-delay model and an SRAM cell type that are tailored to NTC, (ii) modeling SRAM failure modes emerging at NTC, and (iii) accounting for the impact of leakage current in SRAM timing and stability models.

We evaluate a simulated 11nm, 288-core tiled manycore at both NTC and STC. Our results show that the expected process variations induce higher differences in frequency (*f*) and power at NTC than at STC. For example, the maximum difference in tile *f* within a chip is ≈3.7x at NTC and only ≈2.3x at STC. We evaluate different core-

[‡] This work was supported in part by the National Science Foundation under grant CCF-1012759 and CAREER Award CCF-0953603; DARPA under UHPC Contract Number HR0011-10-3-0007; DOE ASCR under Award Number DE-FC02-10ER2599; an IBM Faculty Award; and a generous gift from AMD.

tiling organizations in the chip and different configurations of on-chip V_{dd}- and f-domains. Our experiments show that variation management is especially important at NTC. Finally, we validate our model against an experimental 80-core prototype chip [11].

This paper is organized as follows: Section II provides a background; Section III presents our VARIUS-NTV variation model; Section IV describes the manycore architecture evaluated; Sections V and VI evaluate VARIUS-NTV for the architecture; Section VII outlines our initial validation of VARIUS-NTV; and Section VIII discusses related work.

II. Background

A. Near-Threshold Computing (NTC) Basics

NTC refers to an environment where V_{dd} is set to a value only slightly higher than the transistors' V_{th} [7], [13], [28]. For current technologies, this roughly corresponds to V_{dd} ≈500mV, while the V_{dd} of conventional (or STC) environments is ≈1V.

NTC pushes back the manycore power wall by reducing the energy per operation several times compared to STC — at the expense of degrading the frequency of operation [13]. The result is that the power is expected to reduce by about an order of magnitude, allowing more cores to operate simultaneously for the same manycore power envelope. If the application has parallelism, this is a major advantage.

Figure 1 compares the scaling of three parameters under NTC, STC, and as imposed by classical CMOS theory [10]: supply voltage, transistor delay and power density. The X axis shows gate length to characterize each technology generation. Classical scaling relies on scaling V_{dd} down at every technology generation by a constant scaling factor κ. Both V_{dd} and transistor delay reduce at each generation, giving rise to a constant power density. Conventional STC scaling deviates from classical scaling in that the decrease of the transistor's V_{th} has practically stopped to keep subthreshold leakage under control, which in turn has prevented V_{dd} from scaling [19]. A consequence of this fact is that power density now keeps increasing. As we go from STC to NTC scaling, the curves experience vertical shifts. Specifically, as V_{dd} decreases (Figure 1(a)), power density goes down and transistor delay increases (Figure 1(b)).

In terms of energy and delay, NTC is close to a sweet spot. Figure 2 shows the inverse of energy per operation (labeled as energy efficiency) in MIPS/Watt (left Y axis) and the transistor delay (right Y axis) as a function of V_{dd}. In the NTC region, the energy efficiency is high and the transistor delay is relatively low. Away from this region, higher V_{dd} quickly results in substantially lower energy efficiency. Lower V_{dd}, on the other hand, quickly results in slower transistors.

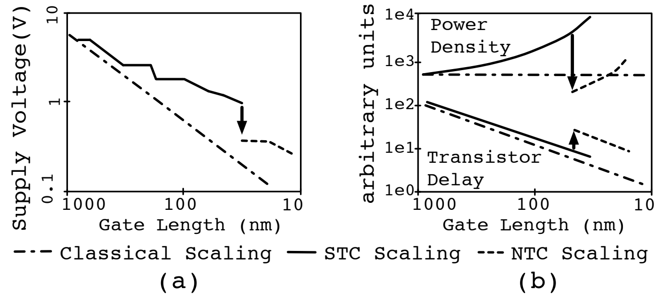


Figure 1. Parameter scaling under three scenarios [7].

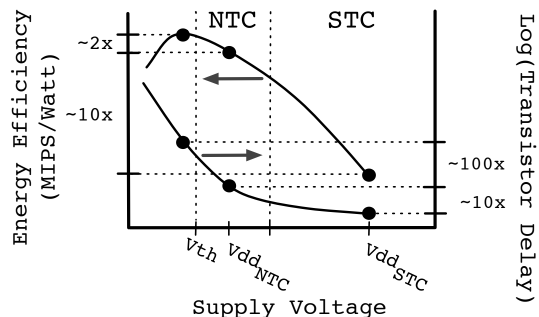


Figure 2. Impact of V_{dd} on energy efficiency and delay [13].

B. The Impact of Process Variations at NTC

Each technology generation becomes increasingly vulnerable to process variations, which manifest across the chip as static, spatial fluctuations in transistor parameters around the nominal values [2], [3]. Within-die (WID) process variations are caused by systematic effects (e.g., due to lithographic irregularities) and random effects (e.g., due to varying dopant concentrations) [38]. Two key process parameters affected by variations are V_{th} and the effective channel length (L_{eff}). The higher the V_{th} and L_{eff} variations are, the higher the variations in transistor switching speed and static power consumption are. This results in chips with increased variation in frequency and power consumption across cores and memories. Note that, in an environment with variation, the average core has lower frequency than before. This is because the slower transistors determine the frequency of the whole core. Moreover, the average core consumes more static power. The reason is that low-V_{th} transistors consume more additional power than high-V_{th} ones save.

Unfortunately, transistor delay and power consumption are more sensitive to variations in V_{th} and L_{eff} at NTC than at STC. Consider transistor delay first. At low V_{dd}, transistor delay is experimentally found to be more sensitive to changes in V_{th} [14]. For example, Figure 3 shows the transistor delay from the model of Markovic *et al.* [28] as V_{th} varies. For V_{dd}=0.6V, the difference in delay between

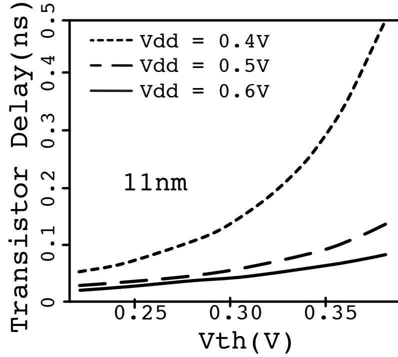


Figure 3. Transistor delay for different V_{th}.

transistors of V_{th}=0.25V and 0.35V is around 30ps, while for V_{dd}=0.4V, it jumps to over 200ps.

Dynamic power is also more sensitive to process variations at NTC than at STC. The reason is that dynamic power depends on the frequency and, as we have seen, at low V_{dd}, transistor delay (and hence frequency) is more sensitive to changes in V_{th}.

C. Modeling Process Variations at STC: VARIUS

There are several microarchitectural models that analyze the impact of process variations on processors and memories at a level that is useful to microarchitects (e.g., [20], [25], [27], [35], [37]). However, these works only apply to STC, and not to NTC. In this paper, we take one of these models, namely VARIUS [37], and substantially extend it so that it applies to NTC. To understand our contributions, we briefly describe VARIUS.

VARIUS models variations in V_{th} and L_{eff}. It models their systematic component by dividing the die into a grid and assigning to each grid point a ΔV_{th} and ΔL_{eff} value as sampled from a multivariate normal distribution with μ=0 and σ_{sys}. Moreover, these values have a spatial correlation that follows a spherical function. With this function, the correlation between two points only depends on their Euclidean distance. At a distance equal to zero, the correlation is one. The correlation then decreases with distance and, at a distance called *Correlation Range* (ϕ), the correlation becomes zero. VARIUS models the random component of variation with a normal distribution with μ=0 and σ_{ran}.

VARIUS plugs the V_{th} and L_{eff} variations in the alpha-power law (Equation 1) and in the equation for static power [9]. It then finds the variation in transistor (and gate) delay and transistor static power, respectively.

$$t_g \propto \frac{V_{dd} \times L_{eff}}{\mu(V_{dd} - V_{th})^{\alpha}} \tag{1}$$

To find the distribution of delay of a pipeline stage, VARIUS proceeds differently depending on whether the stage has only logic, only an SRAM memory access, or a combination of both. For logic, it assumes that wire delays do not suffer from variations and, knowing the number of gates in a logic path, it uses the gate delay variation computed above to compute the path delay variation. If VARIUS knows the distribution of the logic path delays in the stage (e.g., from Razor data [16]), it can estimate the distribution of variation-afflicted logic path delays.

For a stage with a memory access, VARIUS models the 6-transistor SRAM cell of Figure 4(a). Using the variation in transistor delay, it computes the variation in cell read access time. It assumes that the read access time is more critical than the write access time. Then, using the cell access time, it computes the memory line access time. Note that the pipeline stage also contains some logic, namely the decoder, the logic at the intersection of word- and bit-line, and the logic at the sense amplifier. The delay through all this logic is modeled using the previous logic-stage model and is added to the memory access delay to find the distribution of total path delay in the stage.

For pipeline stages that combine both logic and memory access, VARIUS estimates the delay distribution by appropriately weighting the delay of a logic stage and a memory stage. Finally, the pipeline stage with the longest delays determines the safe frequency of the processor.

The static power (P_{sta}) in the processor (or memory module) is found by integrating the P_{sta} of all of its transistors. VARIUS uses statistical principles to find a normal distribution for the processor's P_{sta} as a function of the normal distributions of the transistors' P_{sta}.

III. VARIUS-NTV: A Microarchitectural Model of Process Variations for NTC

VARIUS-NTV builds on VARIUS [37] to develop a microarchitectural model of process variations and resulting timing errors that is valid at NTC. Much of the general approach that VARIUS uses still applies to NTC — although the values of most parameters change. However, there are several important aspects that require complete redesign. This is where VARIUS-NTV contributes.

The main contributions of VARIUS-NTV are in four dimensions, which address four major limitations of VARIUS: (i) the VARIUS model for gate delay is based on the alpha-power law, which is only accurate for V_{dd} much larger than V_{th}; (ii) the VARIUS memory model uses a 6-transistor SRAM cell, which cannot reliably operate at NTC; (iii) for SRAM cells, the VARIUS model only considers read access (or timing) failures, while other memory failure modes dominate at NTC; and (iv) in the SRAM failure analysis, VARIUS neglects the impact of leakage while, at NTC, the impact of leakage is substantial.

In this section, we present the main contributions of VARIUS-NTV.

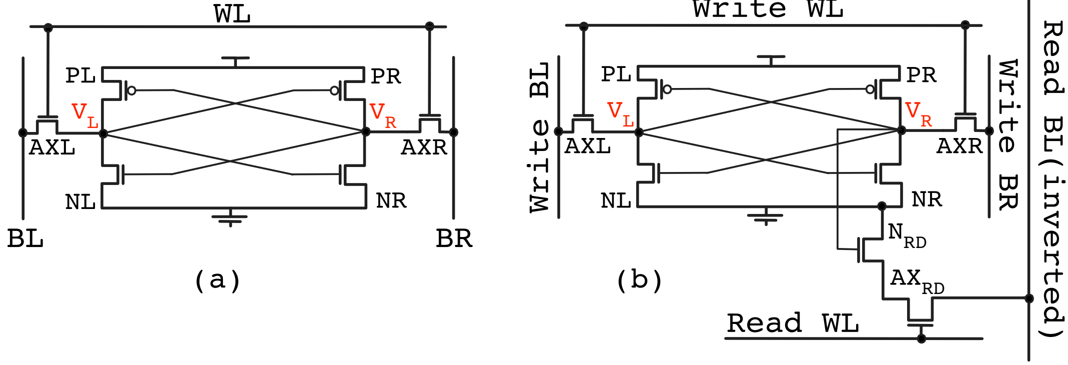


Figure 4. SRAM cell architecture: conventional 6-transistor cell (a) and 8-transistor cell (b). V_R and V_L are the voltages at the nodes indicated, which are referred to as nodes *R* and *L*, respectively.

A. Gate Delay

To model the gate delay (t_g), VARIUS uses the alpha-power law (Equation 1), where α is a process parameter capturing carrier velocity saturation, and μ identifies the carrier mobility as a function of the temperature (T). This equation does not model the NTC region accurately. There are alpha-power law variants [4], [6], [21], [31] that attempt to extend the model to the subthreshold region. Usually, they come with an increased number of fitting parameters that have no direct physical interpretation. Furthermore, that they cover the subthreshold region does not necessarily imply that they model NTC properly.

Consequently, in VARIUS-NTV, we use the EKV-based [15] model proposed by Markovic *et al.* [28]. The formula for the on-current is given in Equation 2, where v_t is the thermal voltage and n a process-dependent parameter determined by subthreshold characteristics. The carrier mobility's *T* dependence is $\mu \propto T^{-1.5}$.

$$I \propto \mu/L_{eff} \times n \times v_t^2 \times ln^2(e^{\frac{Vgs-Vth}{2\times n\times v_t}}+1) \qquad (2)$$

The resulting gate delay, obtained from *CV/I*, is shown in Equation 3. The equation captures the variation in gate delay as a function of the variation in V_{th} and L_{eff}. Since the EKV model covers all regions of operation, Equation 3 is equally valid at STC and NTC. In all cases, V_{th} is a function of V_{dd} and temperature as per Equation 4, where V_{th_0}, V_{dd_0} and T_0 are the nominal values of these parameters, and k_T and k_{DIBL} represent constants of proportionality capturing the impact of T and DIBL (Drain Induced Barrier Lowering) on V_{th}, respectively.

$$t_g \propto \frac{V_{dd} \times L_{eff}}{\mu \times n \times v_t^2 \times ln^2(e^{\frac{Vdd-Vth}{2\times n\times v_t}}+1)} \qquad (3)$$

$$V_{th} = V_{th_0} + k_{DIBL}(V_{dd} - V_{dd_0}) + k_T(T - T_0) \qquad (4)$$

B. SRAM Cell

VARIUS uses the conventional 6-transistor cell shown in Figure 4(a). This cell requires careful sizing of the transistors, since it poses conflicting requirements on the AXR and AXL access transistors to prevent both read and write failures. While such a design is typical at STC, it becomes very hard to use at NTC, where transistors are more sensitive to process variations. One way to address this problem is to power SRAMs at a higher V_{dd} than the logic. Unfortunately, this approach is costly, since cache memory and logic blocks are often highly interleaved in the layout. Moreover, it requires extra voltage regulators in the platform, and results in additional design, validation, and testing issues. Finally, it is hardly scalable: as we move to smaller technologies, the relative difference between the safe SRAM and logic voltages increases, diminishing the power reduction benefit of NTC.

Consequently, VARIUS-NTV uses the 8-transistor cell of Figure 4(b) [8], [29]. This cell is easier to design reliably because it decouples the transistors used for reading (AX_{RD} and N_{RD}) and those for writing (the rest). Compared to the 6-transistor cell, read and write timing margins can be independently optimized with marginal increase in cell area [8]. In addition, of the five types of SRAM failure modes (read timing, read upset, write stability, write timing, and hold) [30], this cell eliminates read upset failures because the cell's internal nodes are decoupled from the read bit-line (BL).

C. Memory Failure Modes

While VARIUS only considers read timing failures, VARIUS-NTV models all of the SRAM failure modes (except read upsets, which cannot occur in the 8-transistor cell because a read cannot flip the cell contents by construction). We now describe how VARIUS-NTV models them.

1) Hold Failure: In a cell storing 0 ($V_R = 0,\ V_L = 1$), at low V_{dd}, the voltage V_L decreases by construction. This is

because, when the cell is not accessed, although NL, PR, and the access transistors are off, there is leakage through NL and AXL. A hold failure occurs when the leakage current through the NL and AXL transistors in Figure 4(b) reduces V_L below the V_{SWITCH} of the $PR-NR$ inverter while the cell is not being accessed. At that point, the cell's state is lost.

To model these failures at a given V_{dd}, VARIUS-NTV uses Kirchhoff's current law to compute V_L and V_{SWITCH} at V_{dd}. V_L is extracted from $I_{PL}(V_L) - I_{NL}(V_L) - I_{AXL}(V_L) = 0$, where

$$\begin{aligned} I_{PL}(V_L) &\propto \mu/L_{eff} \times n \times v_t^2 \times ln^2(e^{\frac{Vdd-Vth}{2\times n\times v_t}}+1) \\ I_{NL}(V_L) &\propto \mu/L_{eff} \times T^2 \times e^{-\frac{Vth}{n\times v_t}} \\ I_{AXL}(V_L) &\propto \mu/L_{eff} \times T^2 \times e^{-\frac{Vth}{n\times v_t}} \end{aligned} \quad (5)$$

and where V_{th} in each equation is expressed as a (different) function of V_L.

Similarly, V_{SWITCH} is extracted from $I_{PR}(V_{SWITCH}) - I_{NR}(V_{SWITCH}) + I_{AXR}(V_{SWITCH}) = 0$ for the $PR-NR$ inverter when V_{IN} = V_{OUT} [30].

The hold failure probability of a cell is $P_{Cell,Hold} = P[V_L(V_{dd}) - V_{SWITCH}(V_{dd}) < 0]$. Then, the hold failure probability of a line is $P_{Line,Hold} = 1 - (1 - P_{Cell,Hold})^{line_size}$, where *line_size* is the number of cells per line, and $1-(1-P_{Cell,Hold})^{line_size}$ gives the probability that at least one cell fails. A line is faulty if at least one of its cells is faulty. The failure probability of cells is assumed independent in this case.

To avoid hold failures, the minimum allowable supply voltage, $V_{ddMIN,Cell}$, is obtained by solving $V_L(V_{ddMIN,Cell}) = V_{SWITCH}(V_{ddMIN,Cell})$ under variation. Then, $V_{ddMIN,Line} = max(V_{ddMIN,Cell})$ for all the cells in the line.

2) Write Stability Failure: Without loss of generality, we focus on a cell that stores a 0 (V_R=0 and V_L=1). VARIUS-NTV computes the voltage (V_{LW}) that node L reaches when the write BL is set to 0 (where BR = 1) and the write duration is extended to infinity. If the value is above the switching threshold of the $PR-NR$ inverter (V_{SWITCH}), then a write failure occurs.

The V_{LW} distribution is computed using Kirchhoff's current law at node L, from $I_{PL}(V_{LW}) - I_{NL}(V_{LW}) - I_{AXL}(V_{LW}) = 0$. The V_{SWITCH} distribution is extracted as explained above in the hold failure analysis.

In all cases, transistor parameters are subjected to the variation model. Finally, the per-cell probability of write stability failure becomes $P_{Cell,WStab} = P[V_{LW} - V_{SWITCH} > 0]$. A memory line suffers from write stability failure if there is at least one cell in the line suffering from it.

3) Read Timing Failure: VARIUS-NTV computes the random variable that captures the time taken to generate a detectable voltage drop on the read bit-line as

$$D_{VarReadCell} \propto \frac{1}{I_{AX_{RD}} + \sum I_{STA}} \quad (6)$$

where $I_{AX_{RD}}$ is the bit-line discharge current through the AX_{RD} transistor in Figure 4(b), and $\sum I_{STA}$ is the leakage over all of the cells attached to the bit-line. To calculate the distribution of $1/I_{AX_{RD}}$, first, the source voltage of AX_{RD}, V_{RD}, is extracted by solving the Kirchhoff's law at this node, from $I_{AX_{RD}}(V_{RD}) = I_{N_{RD}}(V_{RD})$. When reading from a cell storing 1 (V_R=1 and V_L=0), the transistor currents follow from:

$$\begin{aligned} I_{AX_{RD}} &\propto \mu/L_{eff} \times n \times v_t^2 \times ln^2(e^{\frac{Vdd-V_{RD}-Vth}{2\times n\times v_t}}+1) \\ I_{N_{RD}} &\propto \mu/L_{eff} \times n \times v_t^2 \times ln^2(e^{\frac{Vdd-Vth}{2\times n\times v_t}}+1) \end{aligned} \quad (7)$$

where V_{th} in each equation is expressed as a (different) function of V_{RD}.

Then, the probability distribution of $D_{VarReadCell}$ can be attained by applying those of V_{th} and L_{eff} given by the variation model to $\frac{1}{I_{AX_{RD}(V_{RD})}+\sum I_{STA}}$. Following the VARIUS methodology, the maximum of $D_{VarReadCell}$ over all of the cells in a line is the time to read an entire memory line $D_{VarReadLine}$. Finally, the probability of read access failure ($P_{ReadAccess}$) is $P[D_{VarReadLine} > t_{READ}]$, where t_{READ} is the designated read duration.

4) Write Timing Failure: Given a cell without write stability failure, VARIUS-NTV models a write timing failure by computing $D_{VarWriteCell}$. This is the time that node *L* takes to reach the switching threshold (V_{SWITCH}) of the *PR-NR* inverter. It is:

$$\begin{aligned} D_{VarWriteCell} \propto \frac{1}{I_L} &= \int_{Vdd}^{V_{SWITCH}} dv_L/i_L(v_L) \\ i_L(v_L) &= i_{PL}(v_L) - i_{NL}(v_L) - i_{AXL}(v_L) \end{aligned} \quad (8)$$

where I_L is the discharge current at node *L* during the write, obtained following [30]. $i_L(v_L)$ is a function of Gaussian random variables V_{th} and L_{eff} under process variation. It is obtained with Kirchhoff's current law.

After obtaining the probability distribution for $D_{VarWriteCell}$, we compute the distribution of the maximum of $D_{VarWriteCell}$ over all of the cells in a line. Finally, the probability of write timing failure ($P_{WriteTiming}$) is $P[D_{VarWriteLine} > t_{WRITE}]$, where t_{WRITE} is the designated write duration.

D. Impact of Leakage

At NTC, the magnitude of the leakage current (I_{off}), decreases when compared to STC. However, the on-current (I_{on}), decreases even more due to lower V_{dd}. Hence, the relative impact of I_{off} increases. Consequently, unlike VARIUS, VARIUS-NTV takes into account the impact of the leakage current on SRAM timing and stability, as we have seen in previous sections. As part of I_{off}, we only consider

subthreshold leakage; we exclude gate leakage because we assume high-K metal gate devices like the ones currently in use.

IV. Manycore Architecture Modeled

To evaluate VARIUS-NTV, we model an 11nm manycore architecture that operates at NTC. The manycore is organized in tiles (36 in our default configuration) for ease of design (Figure 5). Each tile has a tile memory and several cores (8 in our default configuration), each with a per-core memory. Each core is a single-issue engine where memory accesses can be overlapped with each other and with computation. Each tile memory is a bank of a shared L2 cache, while the per-core memories are L1 caches. Data in the L1 caches is kept coherent with a directory-based MESI coherence protocol where each pointer corresponds to one tile. The cores are connected with a bus inside each tile and with a 2D torus across tiles. Table I shows the default architecture and technology parameters. In the table, all of the parameters that are not labeled with STC refer to the NTC environment.

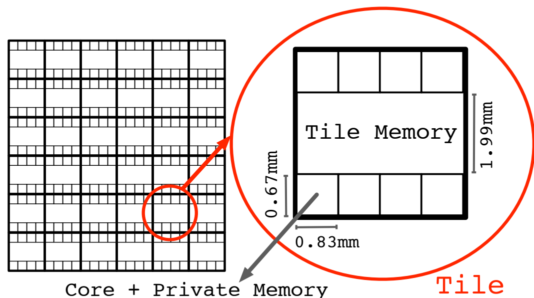


Figure 5. Manycore architecture used to evaluate VARIUS-NTV.

We evaluate an STC version of the manycore and three NTC versions of it. The three NTC versions differ based on the use of voltage and frequency domains, as listed in Table II.

The technology parameters used in Table I are derived from ITRS [22] and projected trends from industry. Every single experiment is repeated for 100 chips with different variation profiles, and we present the average. More samples beyond 100 do not change the results noticeably.

V. Experimental Setup

We evaluate VARIUS-NTV by using it to estimate the performance and power consumption of the manycore architecture of Section IV. We interface Pin [26] over a user-level pthreads library to the SESC [34] cycle-level architectural simulator. SESC estimates both execution time and energy consumed. The energy analysis relies on McPAT [24] scaled to 11nm. An updated version of HotSpot takes the detailed

System Parameters	
Technology node: 11nm Num. Cores: 288 Num. Tiles: 36 (8 cores/tile)	P_{MAX} = 100W T_{MAX} = 100°C Chip area ≈ 20mm x 20mm
Variation Parameters	
Correlation range: $\phi = 0.1$ Total $(\sigma/\mu)_{Vth} = 20\%$; equal contrib. systematic & random	Sample size: 100 chips Total $(\sigma/\mu)_{Leff} = 10\%$; equal contrib. systematic & random
Technology Parameters	
Vdd_{NOM} at STC = 0.77V Vth_{NOM} at STC = 0.30V f_{NOM} at STC = 3.3GHz $f_{interconnect}$ at STC = 2.5GHz $k_T = -1.5mV/K$; $n = 1.5$	Vdd_{NOM} at NTC = 0.54V Vth_{NOM} at NTC = 0.33V f_{NOM} at NTC = 1.0GHz $f_{interconnect}$ at NTC = 0.8GHz $k_{DIBL} = -150mV/V$
Architectural Parameters	
Per-core memory: 64KB WT, 4-way, 2ns access, 64B line On-chip network: bus inside tile and 2D-torus across tiles Crossing a f domain boundary: 2ns	Tile memory: 2MB WB, 16-way, 10ns access, 64B line Directory-based MESI Avg. memory round-trip access time (before contention): ≈80ns

Table I
Architecture and technology parameters.

Name	NTC Manycore Configuration
MVMF	Multiple V_{dd} and multiple f domains (one per tile).
SVMF	Single chip-wide V_{dd} domain and one f domain per tile.
SVSF	Single chip-wide V_{dd} and f domains.

Table II
Configurations for the NTC manycore.

layout of the chip and models the temperature, in turn affecting the leakage energy in a feedback loop. VARIUS-NTV is implemented in R [40].

In our experiments, we run multi-programmed workloads that contain some or all of the following 8 PARSEC applications: blackscholes, ferret, fluidanimate, raytrace, swaptions, canneal, dedup, and streamcluster. Each application can run with 4, 8 or 16 threads. For each application, we measure the complete parallel section (called Region of Interest or ROI) running the *simsmall* input data set.

VI. Evaluation

In our evaluation, we first describe how we set the operating voltages and frequencies of the manycore, then assess the impact of process variations in NTC and STC environments, and then explore some design parameters.

A. Computing the Operating Point

To determine the operating V_{dd} and f at NTC, our model starts with SRAM blocks. Our goal is to estimate V_{ddMIN}, the minimum sustainable V_{dd}. It is set by hold and write stability failure analyses.

Our model first finds the minimum V_{dd} needed to avoid hold failures, namely $V_{dd,hold}$. The $V_{dd,hold}$ distribution is attained by solving $V_L(V_{dd,hold}) = V_{SWITCH}(V_{dd,hold})$, where the former is the voltage at node L (Figure 4(b)), while

the latter is the switching threshold of the *PR-NR* inverter. The chosen $V_{dd,hold}$ value is obtained at the 3σ of the distribution — after approximating to a normal distribution. Our model then proceeds with write stability failure analysis, to guarantee that the chosen $V_{dd,hold}$ also avoids write stability failures. At this step, a higher V_{dd} may emerge, if the write stability failure rate at $V_{dd,hold}$ remains higher than the target tolerable error rate. The resulting V_{dd} is V_{ddMIN}.

Once V_{ddMIN} is picked, VARIUS-NTV considers timing issues in order to set the *f*. The selected *f* is determined by the slowest component of the chip, based on our model's analysis of path delay distributions at V_{ddMIN}. For logic blocks, the analysis follows that of VARIUS [37]. For SRAMs, it can be shown that, for the parameters considered, write timing requires longer delays than read timing for the same V_{dd}. This is consistent with the work of Abella *et al.* [1]. Hence, write timing analysis determines the path delays in each SRAM block. To determine the maximum path delay, VARIUS-NTV approximates the path delay distributions to normal ones and picks the 3σ cut-off point. This maximum delay determines the *f* at V_{ddMIN}.

B. Impact of Process Variations at NTC and STC

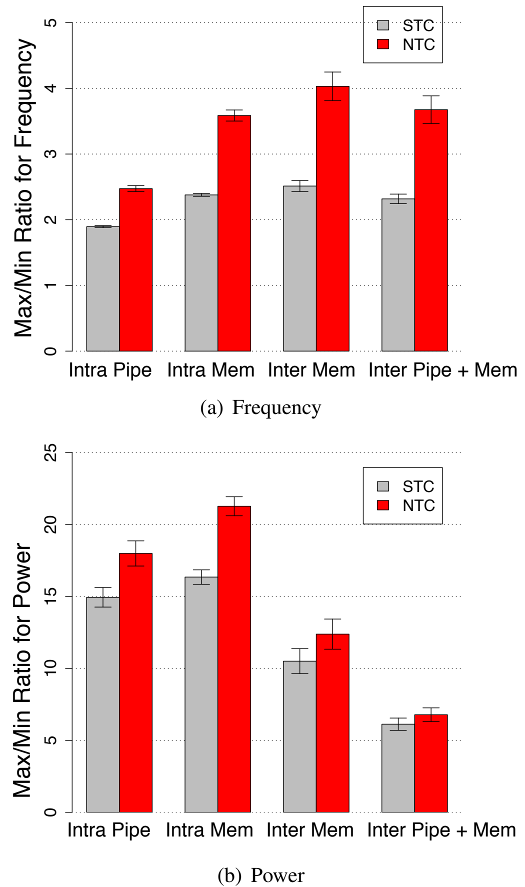
To examine the impact of WID process variations on the *f* and power consumption at NTC and STC, we consider three types of on-chip blocks separately: logic (the core pipelines), small memories (the per-core local memories) and large memories (the per-tile memories). We do this because they have different critical path distributions. In all cases, the *f* for a block is determined by finding the distribution of the path delays in the block at V_{ddNOM} and then picking, as the period for the block, the delay at the 3σ of the distribution. The power of the block is the sum of the static and dynamic components.

We consider intra-tile variations first. In each tile, we compute the ratio of the frequencies of the fastest and slowest pipelines in the tile. We then take the average of the ratios across all tiles (*Intra Pipe*). We repeat the same process for local memories in the tile to calculate *Intra Mem*. Finally, for the power consumption, we take the power ratio of highest to lowest consuming pipelines, and highest to lowest consuming local memories, to compute *Intra Pipe* and *Intra Mem*, respectively.

For inter-tile variations, we measure the ratio of the frequencies of the fastest and slowest tile memories on chip (*Inter Mem*). We then consider the frequency that each tile can support (the lowest frequency of its pipelines, local memories and tile memory), and compute the ratio of the frequencies of the fastest and slowest tiles (*Inter Pipe+Mem*). Finally, we repeat the computations for power (*Inter Mem* and *Inter Pipe+Mem*). We report the mean of the experiments for 100 chips.

Figure 6 compares these ratios for NTC and STC. Figure 6(a) shows the *f* ratios. We observe that the frequency ratio of the fastest to the slowest blocks is substantially higher at NTC than at STC — for the same process variation profile. For example, *Inter Pipe+Mem* at NTC is 3.7, while it is only 2.3 at STC (Figure 6(a)). This is because a low V_{dd} amplifies the effect of process variations on delay.

Figure 6(b) shows the power ratios. The variation in total power also increases at NTC. However, the relative difference in power ratios between NTC and STC is generally smaller than the relative difference in frequency ratios. The reason is that power includes both dynamic and static power, and the ratios for static power are the same for NTC and STC. Consequently, the relative difference in power ratios is smaller. Still, the absolute difference is significant. Consequently, the chip is more heterogeneous at NTC.



(a) Frequency

(b) Power

Figure 6. Impact of variations at NTC and STC.

These experiments have used a fixed, safe V_{ddNOM} for the whole chip. In reality, process variations in the SRAM cells result in each tile supporting a different V_{ddMIN}, the minimum sustainable V_{dd} to avoid failures. Such V_{ddMIN} values are lower than V_{ddNOM} for many tiles. Figure 7 shows the distribution of the V_{ddMIN} values for all the tiles in a sample chip at NTC. The data is shown as a histogram. We can see that the V_{ddMIN} values of tiles in a chip vary

along a significant 0.46-0.58V range.

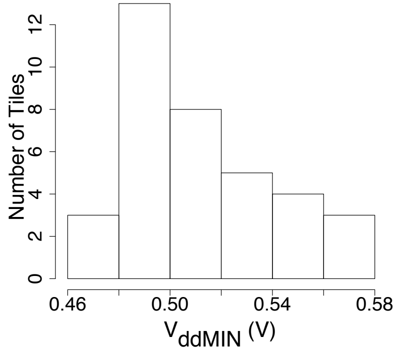


Figure 7. Values of V_{ddMIN} for all the tiles of a representative chip at NTC.

C. Design Space Exploration

A promising way to combat the increased impact of process variations is to rely on fine grain, per-tile V_{dd} and *f* tuning. To quantify the effect, we compare the manycore configurations of Table II across different tile granularities ranging from 4 cores per tile to 16 cores per tile. *MVMF* is an environment with a V_{dd} and an *f* domain per tile; *SVMF* has a single V_{dd} domain in the chip but one *f* domain per tile; finally, *SVSF* characterizes a variation-oblivious environment, with a single V_{dd} and *f* domain per chip.

Figure 8 compares the performance (in normalized *MIPS*) of our 288-core NTC chip for the different environments. We consider two workload scenarios: one where we use all the tiles in the chip (Figure 8(a)) and one where we only use about half of the tiles (Figure 8(b)). Specifically, we use 128 out of the 288 cores and leave the others idle. Figure 9 repeats the analysis for STC.

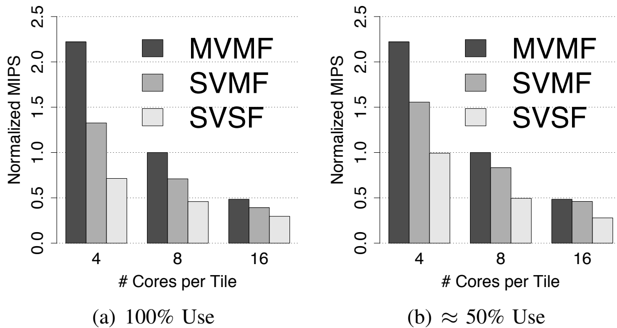


Figure 8. Performance of our 288-core chip at NTC with different tile sizes and configurations. The charts correspond to using all the tiles (a) and using approximately only half (b).

In each figure, we keep the total number of cores in the chip constant, and perform a sensitivity analysis of different tile granularities: 4, 8 or 16 cores per tile. In each case, the workload consists of 4-threaded, 8-threaded, or 16-threaded parallel applications, respectively, from PARSEC. Each application uses one tile, and we report the average performance of the workload in *MIPS*. In each plot, to make the comparison fair, the power consumed by all of the environments is kept constant. In *MVMF*, the per-domain V_{dd} and *f* are set as per Section VI-A. Specifically, each tile runs at the tile-specific V_{ddMIN}, and at the maximum *f* that it can support at this voltage. In *SVMF*, all the tiles in the chip run at the maximum of the V_{ddMIN}s across all tiles. The per-tile frequencies are increased accordingly. Finally, in *SVSF*, the chip uses the same voltage as *SFMV* but it runs at the chip-wide minimum of per-tile frequencies. Recall that the V_{ddMIN} of a tile represents the maximum V_{ddMIN} across its components, where the *f* of a tile corresponds to the minimum *f* across its components at the designated tile V_{dd}. The applications are assigned to tiles according to highest average IPC application to highest *f* tile. After the *MIPS* of each environment is computed, it is normalized to that of *MVMF* for an 8-core tile in each plot.

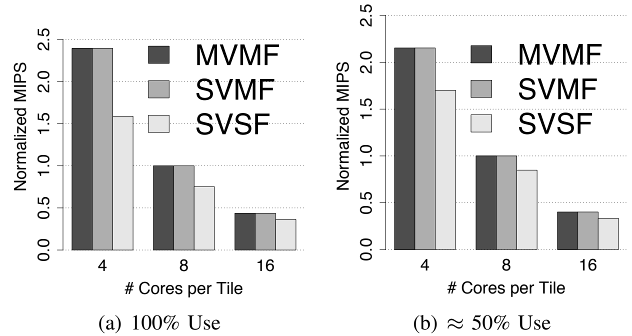


Figure 9. Performance of our 288-core chip at STC with different tile sizes and configurations. The charts correspond to using all the tiles (a) and using approximately only half (b).

Starting with the fully-utilized chip (Figure 8(a)), we observe that *SVMF* only attains 59%, 71%, and 81% *MIPS* of *MVMF*, for 4-core, 8-core, and 16-core tiles, respectively. This is because it does not exploit the multiple V_{dd} domains of *MVMF*. The difference between the two bars gets larger as the tile granularity becomes finer, as *MVMF* tracks core-to-core variations closer. *SVSF* in this case only reaches 32%, 46%, and 61% *MIPS* of *MVMF*, for 4-core, 8-core, and 16-core tiles, respectively. As the tile granularity increases, the differences between the different configurations diminish.

Figure 8(b) repeats the experiment when only ≈ half of the tiles are busy. For *MVMF*, we pick the 32, 16, and 8 most *MIPS/W*-efficient tiles for 4-, 8-, and 16-cores per tile granularity, respectively, and then assign the applications of higher IPC to the faster tiles in turn. The resulting power

consumption is the power budget that we allow to the other environments. The other environments pick their 32, 16 or 8 most *MIPS/W*-efficient tiles that satisfy the budget. We see similar trends as in Figure 8(a) except that the drop in *MIPS* is not as large. The reason is that each environment now picks a subset of energy-efficient tiles — leaving energy-inefficient ones idle.

Finally, in Figure 9, the experiments are repeated for STC. For STC, *MVMF* and *SVMF* become equivalent, since the nominal STC V_{dd} is high enough to produce a safe operating point across all of the tiles. There is no need to set the V_{dd} of some tiles higher or lower depending on their V_{ddMIN}. Apart from this, while generally the same trends apply as under NTC operation, the MIPS loss as incurred by *SVSF* operation is much less.

VII. Model Validation

Our initial validation of VARIUS-NTV involves a validation of the parameters used and a comparison to the results reported in an experimental chip.

1) Validation of Model Parameters: VARIUS-NTV builds on the VARIUS variation and timing error model which, as explained in [37], was calibrated with experimental data from Friedberg *et al.* [17] and Razor [16], and validated with error rates in logic and memory [37]. To validate the new VARIUS-NTV formulas, we start with V_{th}, which is a complex function of V_{dd}, *Leff*, and other technology parameters. We obtained a version of the 12nm Predictive Technology Model (PTM) from Yu Cao from Arizona State University [32]. We compared the V_{th} values generated by VARIUS-NTV to those generated by the BSIM analytical model [5], and HSPICE. The V_{th} values from VARIUS-NTV closely track those from both HSPICE and BSIM with less than 1% error over the designated V_{dd} range. The main source of discrepancy is the accuracy of modeling the DIBL effect.

We then used V_{th} values from VARIUS-NTV to generate values for gate delay and static power. We compared the values to HSPICE measurements of a FO4 inverter chain. The delay and static power scaling trends of VARIUS-NTV follow HSPICE within a 10% of error for our V_{dd} range.

2) Comparison to Silicon Measurements: To further validate VARIUS-NTV, we compare its outputs to the variation measurements from Intel's 80-Core TeraFLOPS processor [11]. To this end, we experimented with a 12mm×20mm chip that mimicks the TeraFLOPS processor, where each core (which they call tile) has 2 floating point units, a 3KB instruction memory, and a 2KB data memory. According to the chip micrograph, the chip organizes the 80 cores into 10 rows and 8 columns. To match their technology parameters, we adapted VARIUS-NTV to a 65nm CMOS technology with a V_{ddNOM} of 1.2V.

Figure 8 in [11] depicts the *measured* variation in core frequency (f_{MAX}) for the 80 cores of a single die at 50°C and V_{dd}=0.8V. At 0.8V, the authors report a ratio of highest core frequency to lowest core frequency equal to 1.62.

We repeat the conditions in which these measurements were taken to the extent that we can. We generate VARIUS-NTV frequency maps for 100 sample dies, assuming $(\sigma/\mu)_{Vth} = 5\%$ for the 65nm technology, with an equal contribution of random and systematic variation. The histogram of the resulting ratios of highest core frequency to lowest core frequency as generated by VARIUS-NTV is shown in Figure 10(a). As shown in the histogram, VARIUS-NTV produces an average value of ≈1.48 for the ratio of frequencies, with a 95% confidence interval of (1.452, 1.483).

Further, Figure 10(b) shows the frequency distribution of the cores in one of the dies, as generated by VARIUS-NTV at 0.8V. For this particular die, the ratio of highest core frequency to lowest core frequency is ≈1.4. This figure is very similar to Figure 8 in [11].

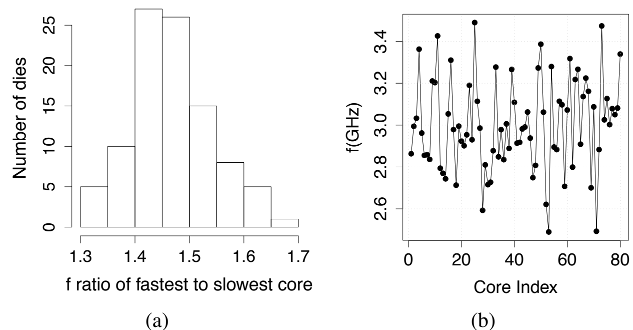


Figure 10. Data generated by VARIUS-NTV that replicates the data presented in [11]. Chart (a) shows a histogram of the ratios of highest core frequency to lowest core frequency over 100 dies. Chart (b) shows the frequency map for one of the sample dies.

Recall that the 80-Core processor does not represent an NTC design. However, our validation experiments are run at the relatively low 0.8V (where the nominal V_{dd} at 65nm is 1.2V). No further measured data is provided in [11] below 0.8V. To our knowledge, there is no detailed variation characterization of any NTC chip that is available.

VIII. Related Work

There are several microarchitectural models that analyze the impact of process variations on the frequency and power of processors and memories at a level that is useful to microarchitects. They include the work of Humenay *et al.* [20], Liang and Brooks [25], Marculescu and Talpes [27], Romanescu *et al.* [35], and Sarangi *et al.* [37] (on which this work builds) among others. As indicated before, these works only apply to STC, and not to NTC.

A few papers include a good description of the challenges and issues at NTC [7], [13], [28].

There are many other works that are related to evaluating the impact of process variation, mostly in STC environments. We list some of the most relevant here. Humenay *et al.* demonstrate that WID process variations lead to considerable performance and power consumption asymmetry among the cores in a CMP [20]. To minimize such asymmetry, they propose per-core ABB and ASV. Donald and Martonosi analyze core-to-core power variations in a CMP due to WID variation [12]. They propose to turn off cores when they consume excessive leakage power in order to maximize the chip-wide performance/power. Herbert and Marculescu examine the impact of core size on the throughput of a fixed area chip in the presence of WID variations [18]. They find that smaller cores (thus more cores per chip) running at independent f lead to higher throughput than larger ones. Li and Martinez propose to optimize the number of active cores and their *Vdd*s and *f*s jointly while running a workload on a CMP [23] where they apply DVFS chip-wide rather than independently per core. In [33], Rangan *et al.* propose a throughput driven scheduling scheme to guarantee that a variation-afflicted chip performs very close to a perfect chip operating at the average frequency of the former. Rotem *et al.* [36] analyze the impact of single and multiple voltage and frequency domains in a CMP environment, considering power delivery limitations. They propose a clustered topology to maximize performance. The authors ignore the impact of variation. Finally, Teodorescu and Torrellas [39] examine the impact of process scheduling in the context of a manycore with variation. They provide heuristics to schedule the workload for performance or for power efficiency. It would be interesting to reproduce these works in the context of NTC.

IX. Conclusion

To help confront process variations at the architecture level at NTC, this paper has presented the first microarchitectural model of process variations for NTC. The model, called *VARIUS-NTV*, extends an existing variation model for STC. It models how variation affects the frequency attained and power consumed by cores and memories in an NTC manycore, and the timing and stability faults in SRAM cells at NTC. The key aspects include: (i) adopting a gate-delay model and an SRAM cell type that are tailored to NTC, (ii) modeling SRAM failure modes emerging at NTC, and (iii) accounting for the impact of leakage in SRAM failure models.

We evaluated a simulated 11nm manycore at both NTC and STC. Our results showed that the expected process variations induce higher differences in f and power at NTC than at STC. For example, the maximum difference in tile f within a chip is ≈3.7x at NTC and only ≈2.3x at STC. We evaluated different core-tiling organizations in the chip and different configurations of on-chip V_{dd}- and f-domains. Our experiments showed that variation management is more crucial at NTC. Finally, we validated our model against an experimental 80-core prototype chip. We are ready to release the model and all its software to the public domain.

References

[1] J. Abella, P. Chaparro, X. Vera, J. Carretero, and A. Gonzalez. High-Performance Low-Vcc In-Order Core. In *International Symposium on High Performance Computer Architecture*, January 2010.

[2] K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, S. R. Nassif, E. J. Nowak, D. J. Pearson, and N. J. Rohrer. High-Performance CMOS Variability in the 65-nm Regime and Beyond. In *IBM Journal of Research and Development*, July/September 2006.

[3] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De. Parameter Variations and Impact on Circuits and Microarchitecture. In *Design Automation Conference*, June 2003.

[4] K. A. Bowman, B. L. Austin, J. C. Eble, X. Tang, and J. D. Meindl. A Physical Alpha-power Law MOSFET model. In *International Symposium on Low Power Electronics and Design*, August 1999.

[5] BSIM. *http://www-device.eecs.berkeley.edu/~bsim/BSIM4*.

[6] Y. Cao and L. T. Clark. Mapping Statistical Process Variations Toward Circuit Performance Variability: An Analytical Modeling Approach. In *Design Automation Conference*, June 2005.

[7] 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*, February 2010.

[8] L. Chang, R. Montoye, Y. Nakamura, K. Batson, R. Eickemeyer, R. Dennard, W. Haensch, and D. Jamsek. An 8T-SRAM for Variability Tolerance and Low-Voltage Operation in High-Performance Caches. *Journal of Solid-State Circuits*, April 2008.

[9] Y. Cheng and C. Hu. *MOSFET Modeling and Bsim3 User's Guide*. Kluwer Academic Publishers, 1999.

[10] R. Dennard, F. Gaensslen, V. Rideout, E. Bassous, and A. LeBlanc. Design of Ion-Implanted MOSFETs with Very Small Physical Dimensions. In *Journal of Solid-State Circuits*, October 1974.

[11] S. Dighe, S. Vangal, P. Aseron, S. Kumar, T. Jacob, K. Bowman, J. Howard, J. Tschanz, V. Erraguntla, N. Borkar, V. De, and S. Borkar. Within-Die Variation-Aware Dynamic-Voltage-Frequency-Scaling With Optimal Core Allocation and Thread Hopping for the 80-Core TeraFLOPS Processor. *Journal of Solid-State Circuits*, January 2011.

[12] J. Donald and M. Martonosi. Power Efficiency for Variation-tolerant Multicore Processors. In *International Symposium on Low power Electronics and Design*, October 2006.

[13] 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*, February 2010.

[14] M. Eisele, J. Berthold, D. Schmitt-Landsiedel, and R. Mahnkopf. The Impact of Intra-die Device Parameter Variations on Path Delays and on the Design for Yield of Low Voltage Digital Circuits. *Transactions on VLSI Systems*, December 1997.

[15] C. C. Enz, F. Krummenacher, and E. A. Vittoz. An Analytical MOS Transistor Model Valid in All Regions of Operation and Dedicated to Low-voltage and Low-current Applications. *Analog Integrated Circuits Signal Processing*, 1995.

[16] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Zeisler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation. In *International Symposium on Microarchitecture*, December 2003.

[17] P. Friedberg, Y. Cao, J. Cain, R. Wang, J. Rabaey, and C. Spanos. Modeling Within-die Spatial Correlation Effects for Process-design Co-optimization. In *International Symposium on Quality of Electronic Design*, March 2005.

[18] S. Herbert and D. Marculescu. Characterizing Chip-multiprocessor Variability-tolerance. In *Design Automation Conference*, June 2008.

[19] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein. Scaling, Power, and the Future of CMOS. In *International Electron Devices Meeting*, December 2005.

[20] E. Humenay, D. Tarjan, and K. Skadron. Impact of Process Variations on Multicore Performance Symmetry. In *Conference on Design, Automation and Test in Europe*, April 2007.

[21] H. Im. Physical insight into fractional power dependence of saturation current on gate voltage in advanced short channel MOSFETS (alpha-power law model). In *International Symposium on Low Power Electronics and Design*, August 2002.

[22] International Technology Roadmap for Semiconductors (ITRS). *2009 Update*.

[23] J. Li and J. Martinez. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In *International Symposium on High-Performance Computer Architecture*, February 2006.

[24] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *International Symposium on Microarchitecture*, December 2009.

[25] X. Liang and D. Brooks. Mitigating the impact of process variations on processor register files and execution units. In *International Symposium on Microarchitecture*, December 2006.

[26] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood. Pin: Building customized program analysis tools with dynamic instrumentation. In *Conference on Programming Language Design and Implementation*, June 2005.

[27] D. Marculescu and E. Talpes. Variability and energy awareness: A microarchitecture-level perspective. In *Design Automation Conference*, June 2005.

[28] 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*, February 2010.

[29] Y. Morita, H. Fujiwara, H. Noguchi, Y. Iguchi, K. Nii, H. Kawaguchi, and M. Yoshimoto. An Area-Conscious Low-Voltage-Oriented 8T-SRAM Design under DVS Environment. In *Symposium on VLSI Circuits*, June 2007.

[30] S. Mukhopadhyay, H. Mahmoodi, and K. Roy. Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, December 2005.

[31] M. Orshansky, J. Chen, and C. Hu. Direct sampling methodology for statistical analysis of scaled CMOS technologies. *Transactions on Semiconductor Manufacturing*, November 1999.

[32] Predictive Technology Model (PTM). *http://ptm.asu.edu/*.

[33] K. Rangan, M. Powell, G.-Y. Wei, and D. Brooks. Achieving uniform performance and maximizing throughput in the presence of heterogeneity. In *International Symposium on High Performance Computer Architecture*, February 2011.

[34] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos. SESC simulator, January 2005. http://sesc.sourceforge.net.

[35] 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.

[36] E. Rotem, R. Ginosar, A. Mendelson, and U. Weiser. Multiple clock and voltage domains for chip multi processors. In *International Symposium on Microarchitecture*, December 2009.

[37] 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. *Transactions on Semiconductor Manufacturing*, February 2008.

[38] A. Srivastava, D. Sylvester, and D. Blaauw. *Statistical Analysis and Optimization for VLSI: Timing and Power*. Springer, 2005.

[39] R. Teodorescu and J. Torrellas. Variation-Aware Application Scheduling and Power Management for Chip Multiprocessors. In *International Symposium on Computer Architecture*, June 2008.

[40] The R Project for Statistical Computing. *http://www.r-project.org/*.