A $\mu-$architectural Model of Process Variation for Near-Threshold Computing

Ulya R. Karpuzcu* Krishna Kolluru† Nam Sung Kim† Josep Torrellas*
† Department of Electrical and Computer Engineering, University of Wisconsin-Madison
* Department of Computer Science and Electrical and Computer Engineering, University of Illinois at Urbana-Champaign

Abstract

Near-Threshold Voltage Computing (NTC), where the supply voltage is only slightly higher than the transistors' threshold voltage, is a promising approach to push back the many-core power wall. A key NTC shortcoming is the higher sensitivity to parameter (process, supply voltage, temperature) variation. This report introduces a $\mu-$architectural model to characterize increased sensitivity to process (static) variation.

1 Introduction

Ideal CMOS device scaling [11] relies on scaling voltages down with lithographic dimensions at every technology generation by a constant scaling factor κ (Table 1). In this case, power density stays constant.

Transistor/circuit Parameter	Scaling Factor
Vdd	$1/\kappa$
Vth	$1/\kappa$
C	$1/\kappa$
I	$1/\kappa$
Area A (of same functionality)	$1/\kappa^2$
Transistor delay $C \times Vdd/I$	$1/\kappa$
Power dissipation $Vdd \times I$	$1/\kappa^2$
Power density $Vdd \times I/A$	1

Table 1: Classical scaling theory basics [11].

In recent generations, however, to keep leakage current under control, the decrease in the transistor's threshold voltage (Vth) has stopped, which in turn has prevented the supply voltage (Vdd) from scaling [17]. A direct consequence of this fact is that power density no more stays constant: If the chip area is fixed (due to system cooling constraints and the associated cost), per transistor power consumption no more decreases enough to compensate for the increased power of having more transistors integrated in the same area over technology generations. If the the chip power budget is fixed[1], we easily realize that there is a growing gap between what can be placed on a chip and what can be powered-on simultaneously. For example, Figure 1 shows data computed from the ITRS 2008 update [20] assuming beefy cores and a 100W chip power budget. The figure compares the number of cores that can be placed on a chip at a given year and the number of those that can be powered-on simultaneously. The growing gap between the two curves shows the *Many-core Power Wall*. Clearly, we urgently need new ways to operate more energy-efficiently.

An effective way to attain energy-efficient execution is to substantially reduce *Vdd*, to a value only slightly higher than

[1] A viable scaling scenario due to system cooling constraints and the associated cost.

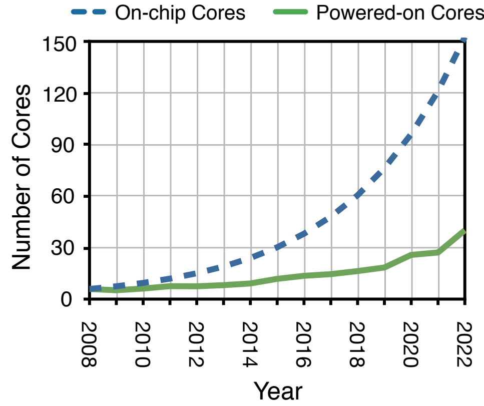


Figure 1: The Many-core Power Wall, based on data from ITRS projections.

Vth. This environment is known as Near-Threshold Voltage Computing (NTC) [9, 13, 26] — as opposed to the conventional, Super-Threshold Voltage Computing (STC). *Vdd* is a powerful lever because it has quadratic and approximately linear impact on dynamic and static energy, respectively. The drawback is a degradation in frequency. However, if lower frequencies can be tolerated through more parallelism in the execution of the application, NTC is very attractive. According to experimental data [9, 13], NTC can attain up to 8–10x higher energy efficiency than STC (measured, for example, in MIPS/Watt). If this comes with a frequency reduction of approximately 10x, the power consumption decreases by 80–100x [13]. Under NTC, therefore, many more cores can be powered-on simultaneously.

A major roadblock in NTC is increased sensitivity to process variation, namely the deviation of device parameters from their nominal values. Already in STC many-cores, process variation results in substantial power and performance degradation [12, 16, 18, 38]. Unfortunately, under NTC, the resulting variation in transistor speed and power consumption intensify, hence it is crucial to develop NT-specific holistic techniques. However, existing $\mu-$architectural models of process variation apply only to ST [18, 23, 25, 33, 35]. There is no model of variation at NT. To capture the increased sensitivity to process variation at NT voltages, we modified the VARIUS [35] model applicable at ST to accommodate

- A NT gate delay model
- A robust SRAM cell architecture to ensure reliable operation at NT voltages
- NT-specific SRAM failure modes
- Leakage models for timing and stability analysis

This report is organized as follows: Section 2 provides a background; Section 3 introduces the variation model; and Sections 4 and 5 evaluate the model.

2 Background

2.1 Process Variation Basics

Each technology generation, manufacturing restrictions exacerbate vulnerability to process variation, which manifests across the chip as static, spatial fluctuations in transistor parameters around the nominal values [2, 3]. Process variation is usually analyzed at global (*die-to-die, D2D*) and local scopes (*within-die, WID*). D2D variation can be approximately captured by a global offset per die, on a per-parameter basis. WID variation, on the other hand, is responsible for the heterogeneity in transistor parameters across the building blocks of any given die, and forms the focus of this report[2].

WID variation is caused by *systematic* effects due to lithographic irregularities and *random* effects due to, e. g., varying dopant concentrations [37]. Two important vulnerable parameters are the threshold voltage (*Vth*) and the effective channel length (L_{eff}), which directly determine a transistor's switching speed and its leakage power.

Since systematic and random effects are triggered by different physical phenomena, their impact can be captured by two independent random variables per vulnerable parameter. The end impact is additive as given by Equation 1, where Vth_0 and L_{eff0} depict the nominal values of Vth and L_{eff}; Vth_{SYS} and L_{effSYS}, the parametric shift due to systematic effects; and Vth_{RAND} and $L_{effRAND}$, the parametric shift due to random effects, respectively.

$$\begin{aligned} Vth &= Vth_0 + \Delta Vth_{WID} = Vth_0 + \Delta Vth_{SYS} + \Delta Vth_{RAND} \\ L_{eff} &= L_{eff0} + \Delta L_{effWID} = L_{eff0} + \Delta L_{effSYS} + \Delta L_{effRAND} \end{aligned} \tag{1}$$

The higher the *Vth* and L_{eff} variation is, the higher the variation in transistor speed, hence in path delay and frequency across the chip is. This results in slower processors, since variation-afflicted path delay generally follows a symmetric distribution, and the slowest path ends up determining the frequency of the whole processor [3]. In addition, the difference between the frequencies of different cores on a many-core increases. Also, as *Vth* varies, transistor leakage varies across the chip. However, low-*Vth* transistors consume more power than high-*Vth* ones save, due to the exponential dependence of leakage power on -*Vth* (the BSIM3v3.2 MOSFET transistor model equation [10]). As a result, with variation, chips consume substantially more leakage power. In a many-core, different cores leak different amounts.

When compared to logic, on-chip memory blocks are more prone to variation, as triggered by aggressive transistor sizing to satisfy high density requirements, and the higher sensitivity to transistor mismatch. With technology scaling, the minimum operating voltage for a conventional SRAM array tends to increase to accommodate higher safety margins as induced by exacerbated variability.

[2]Note that D2D and WID variation may be correlated.

[3]Under process variation, some paths get faster, while others become slower. Theoretically, it could be the case that none of the slower paths exhibit a delay higher than the *nominal* critical path (i.e. the ideal critical path were there no variation); then, the nominal operating frequency would be saved.
Assume that for a given variation-afflicted path delay distribution, a standard deviation of σ_0 results in some slower paths when compared to the nominal critical path. More variation translates to a larger standard deviation. What is the impact of a larger deviation $\sigma > \sigma_0$? Slower paths would be slower, and faster paths would be faster. Hence, the operating frequency would be lower.

2.2 Modeling Process Variation

2.2.1 Modeling Variation in Vth and L_{eff}

While there are several microarchitectural models of process variation (e.g., [18, 23, 25, 33, 35]), this report builds on VARIUS [35].

VARIUS captures *systematic* WID variation by dividing the die into a grid, where each grid point is assigned a Vth and L_{eff} value as sampled from a multivariate Gaussian distribution. Systematic variation shows spatial correlation, in that parameters of transistors in immediate vicinity show less diversity. VARIUS captures spatial correlation by a position and direction independent (*spherical*) correlation function. The correlation between two points on die only depends on their Euclidean distance; decreases first linearly, then sub-linearly, as the distance increases. This function converges to zero – to demarcate negligible correlation – beyond the *correlation distance*, ϕ. The grid granularity should be tailored as a function of ϕ. Due to spatial correlation, specific regions of the die exhibit a systematic shift of the same quantity and in the same direction on a per parameter basis; the grid-granularity should be set to sample at least one point from each such region.

Random variation, on the other hand, does not exhibit spatial correlation, hence each transistor should be considered individually. Unless each grid point corresponds to a transistor, random variation cannot be captured at grid granularity. VARIUS models the random component analytically per variation-afflicted parameter.

2.2.2 Modeling Variation-Induced SRAM Failure Modes

When compared to logic, on-chip memory blocks are more prone to variation, as triggered by aggressive transistor sizing to satisfy high density requirements, and the higher sensitivity to transistor mismatch.

For SRAM, VARIUS models the variation in a conventional 6-transistor cell (Figure 2(a)). The cell consists of two inverters, formed by transistors *PR-NR* and *PL-NL*, connected in a positive feedback loop, and two access transistors, *AXR* and *AXL*. V_R stores the cell's value and V_L its complement. To read from or write to the cell, word-line *WL* is driven high to connect the cell to the bit-lines *BL* and *BR*. To read, the bit-lines are pre-charged to logic high. To write, *BR* is pre-conditioned to the value to be written and *BL* to its complement. This cell is liable to five types of failures [28], of which VARIUS only models one. In the following discussion, assume that V_R=0 without loss of generality [4].

A *Read Upset* failure occurs if, during a read operation, the cell flips its contents. Reading starts by setting $WL = BR = BL = 1$. While reading $V_R = 0$ ($V_L = 1$), BL keeps its value and BR discharges over AXR and NR, giving rise to a voltage difference between the bitlines, which is captured by the sense amplifiers to extract the cell content. Sense amplifiers accelerate the read by detecting small voltage differences between the bitlines.

While BR discharges, due to charge sharing between AXR and NR, V_R increases. NR should be stronger [5] than

[4] Since the cell is symmetric, the discussion applies directly to the case where V_R=1.
[5] If NR is stronger than AXR, V_R would be driven closer to logic low than logic high.

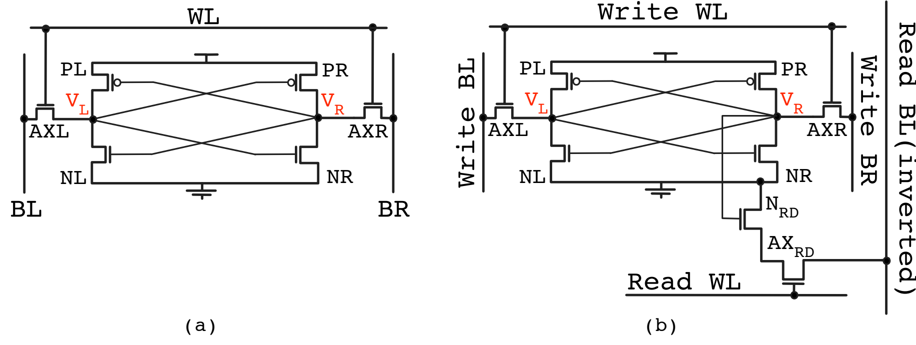


Figure 2: SRAM cell architecture: conventional 6-transistor cell (a) and 8-transistor cell (b). In the figure, V_R and V_L are the voltages at the points indicated, which are referred to as nodes *R* and *L*, respectively.

AXR such that V_R does not reach the switching threshold of the PL/NL inverter during the read, otherwise a read upset flips the cell content [28]. This can be due to variation leading to an increase in Vth_{NR} while Vth_{AXR} decreases (i.e. AXR becomes stronger than NR), and/or causing shifts in Vth_{PL} and/or Vth_{NL} such that the switching threshold of the PL/NL inverter decreases.

A *Read Access* failure occurs if, during a read, the time needed to produce a voltage difference between the two bit-lines exceeds the period that *WL* stays high. This failure occurs when the discharge transistors (*AXR* and *NR*) are too slow: Access failures are triggered by increasing Vth of AXR and/or NR in the presence of variation, such that the transistors become slower. This is the only SRAM failure modeled by VARIUS.

A *Write Stability* failure occurs if, during a write, the cell cannot change its logic state even if the write duration is extended to infinity. To write a 1 to a cell storing 0, i.e. $V_R = 0$ ($V_L = 1$), BL is driven low while BR is pre-conditioned high. Once WL gets triggered, BR again discharges over $AXR - NR$ as it was the case for the read, however, AXR cannot drive V_R high enough to reach the switching threshold of the inverter, to ensure read stability. Hence, V_L can only be driven low through AXL. Since *BL* is 0, current flows through *PL* and *AXL*. Once V_L decreases enough to reach the the switching threshold of the inverter PR/NR, the positive feedback loop completes the write by pulling up V_R high. This is opposed by PL, pulling V_L to Vdd. Depending on the strength of these transistors, and that of *PR* and *NR*, V_L may never reach a value low enough to trigger the switching of the *PR*-*NR* inverter and flip the cell contents. To ensure write stability, PL should be weaker than AXL.

In a cell without the write stability failure, a *Write Timing* failure occurs if the write is unable to change the logic state of the cell by the end of the designated write duration. Write failures are triggered by variation-induced shifts in the switching threshold of the PR/NR inverter and/or PL becoming stronger than AXL.

Finally, a *Hold* failure occurs if the logic value held by the cell is distorted by excessive leakage of the transistors forming the core of the cell, while the cell is not being accessed. Hold failures are triggered by variation-induced shifts in Vth of the transistors forming the core of the storage cell.

While write stability enhances the weaker the access transistors are, read stability demands stronger access transistors.

Due to the circuit symmetry, the analogous semantics apply to read from and write to a cell storing 1, i.e. $V_R = 1$ ($V_L = 0$).

2.3 Near-Threshold Computing Basics

Near-Threshold Computing (NTC) refers to an environment where the supply voltage *Vdd* is set to a value only slightly higher than the transistors' threshold voltage *Vth* [9, 13]. For current technologies [6], this roughly corresponds to *Vdd* $\approx$ 500mV, compared to conventional (or Super-Threshold Computing (STC)) environments, where *Vdd* $\approx$ 1V.

NTC pushes back the many-core power wall by reducing the energy per operation[7] about 10x compared to STC — at the expense of reducing the frequency of operation about 10x [13]. As a result, the power reduces by approximately 100x, allowing more cores to operate simultaneously for the same many-core power envelope. If the application has parallelism, this is a major advantage.

Figure 3 compares the scaling of three key parameters under NTC, STC, and as imposed by classical CMOS theory [11]: Supply voltage, transistor delay and power density. The x-axis denotes gate length to characterize each technology generation. **Classical scaling** relies on scaling voltages down with lithographic dimensions at every technology generation by a constant scaling factor κ (Table 1). Both Vdd and transistor delay reduce by $1/\kappa$ each generation, giving rise to a constant power density [8]. **Conventional STC scaling** deviates from classical scaling in that the decrease in the transistor's threshold voltage (Vth) has recently stopped to keep subthreshold leakage under control, which in turn has prevented the supply voltage (Vdd) from scaling [17]. A direct consequence of this fact is that power density no more stays constant. The curves experience substantial vertical shifts as we go from STC to **NTC scaling**. Specifically, supply voltage (Chart (a)) and power density (Chart (b)) reduce significantly, while transistor delay increases.

The NTC range of operation is close to a sweet-spot in energy efficiency: Figure 4 characterizes the energy efficiency as MIPS/Watt (left Y axis) and the transistor delay (right Y axis) as a function of *Vdd*. In a narrow range of voltages around *Vth*, the energy efficiency is highest. Out of this range, higher voltages quickly result in substantially lower energy efficiency, since power consumption increases more than the delay reduces. Lower voltages, on the other hand, quickly result in excessively slower transistors than acceptable while power consumption decreases significantly. The operating voltage cannot be lowered indefinitely due to reliability constraints.

2.4 Impact of Leakage on on Near-Threshold Computing

At NT voltages, the magnitude of leakage current decreases when compared to ST, however, the on-current, I_{ON}, decreases more due to lower Vdd. Hence, the relative impact of the leakage current, I_{OFF}, increases. Figure 5 points to the drastic decrease in I_{ON}/I_{OFF} in a NTC environment of $Vdd = 0.5V$, when compared to a STC environment of $Vdd = 0.8V$. Consequently, VARIUS-NTC takes into account the impact of the leakage current on SRAM timing and

[6] 45 or 32nm

[7] $C \times Vdd^2$

[8] Subthreshold leakage (current) tends to increase each generation even under classical scaling, due to the exponential dependence on $-Vth$ (Equation 25). However, in early generations the magnitude remained negligible.

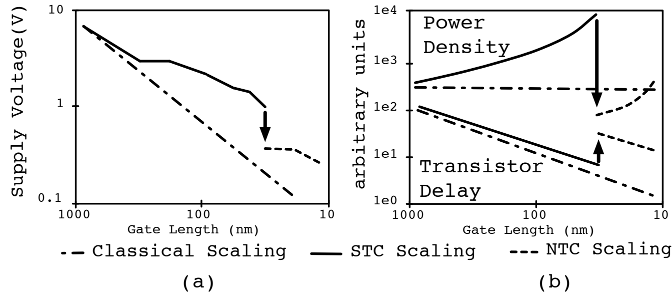


Figure 3: Parameter scaling under three scenarios [9].

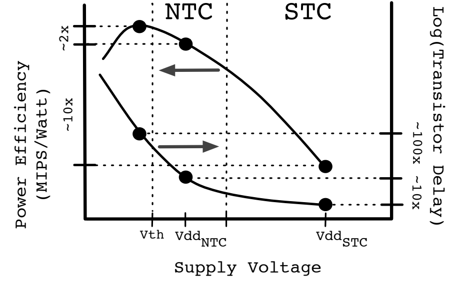


Figure 4: Impact of *Vdd* on energy efficiency and delay [13].

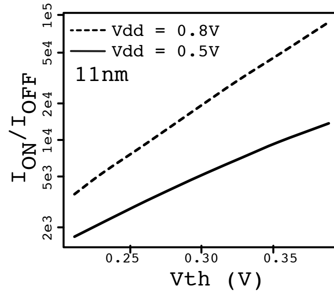


Figure 5: Impact of leakage.

stability, as we will see in later sections. As part of leakage current, we only consider sub-threshold leakage; gate leakage is excluded because we assume high-K metal gate devices like the ones currently in use.

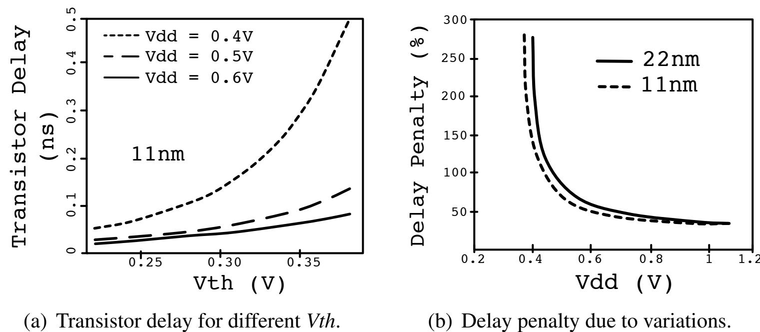
2.5 Impact of Process Variation on Near-Threshold Computing

At NT voltages, the sensitivity of circuit timing and power consumption to variations in *Vth* and L_{eff} drastically increases when compared to ST. The impact of a given ΔVth_{WID} and ΔL_{effWID} on an environment with low voltage (i.e., NT) is much higher than on one with high voltage (i.e., ST).

2.5.1 Impact of Process Variation on Circuit Timing at NT Voltages

Severe frequency variation at NT stems from increased sensitivity of transistor delay to Vth at NT voltages [14], as depicted in Figure 6(a). The transistor delay is obtained from the model of Markovic *et al.* [26] as *Vth* is varied.

The *Vth* range is obtained as follows: First, we assume that Vth follows a Gaussian distribution [37]. The nominal *Vth*



(a) Transistor delay for different *Vth*. (b) Delay penalty due to variations.

Figure 6: Increased sensitivity of circuit timing to process variation at NT voltages.

suggested by ITRS [21] at 11nm constitutes the mean, $\mu = Vth_0 = 0.3V$. For the sake of analysis, we set total (systematic and random) $(\sigma/\mu)_{Vth} = 10\%$, hence the variation-induced $Vth_0 \times (1 \pm 3\sigma)$ covers the range of $[0.21, 0.39]V$. We see that, for *Vdd*=0.6V, the difference in delay between transistors of *Vth*=0.25V and 0.35V is only 18.8ps, while for *Vdd*=0.4V, it jumps to 62.5ps, increasing by $\approx 3.3\times$. As a result of this larger delay variation, the frequency that a processor can support at near-threshold voltages varies more. Figure 6(b) from Chang *et al.* [9] shows the guard-banding of the critical path due to variations for different *Vdd*. As *Vdd* decreases toward *Vth*, the guard-banding required quickly increases, diminishing the potential benefit of NTC. Overall, therefore, it is crucial to develop techniques to cope with process variations in an NTC environment.

2.5.2 Impact of Process Variation on Power Consumption at NT voltages

Variation in power consumption at NT increases since (1) variation in dynamic power increases due to the increase in variation in frequency, as covered in the previous section; (2) the share of static power increases at NT voltages.

3 VARIUS-NTC: A μ-architectural Model for Process Variation at NT Voltages

Since NTC does not impose radical changes in the manufacturing process, the VARIUS methodology to extract Vth and L_{eff} variation maps, still applies – although the values of parameters such as σ may change. These parameters are directly affected by technology scaling trends. Technology scaling came along with frequency improvement – though at a progressively slower rate each generation, while the share of static power has been rapidly increasing. NTC is most effective for throughput-critical environments as opposed to latency-, hence frequency-critical. Also, the significant reduction in dynamic power due to NTC is *not* accompanied by a corresponding reduction in the static power, thus NTC induces a much higher share of static power when compared to STC. Based on these observations, it can be argued that older technology generations – as characterized by a lower frequency and lower leakage – conform better to NTC, however, in [36] it is demonstrated that for performance-critical application spaces, new technologies are still favorable, where throughput represents a stronger function of the operating frequency.

On the other hand, VARIUS performance model and timing error infrastructure cannot be directly deployed at NT, for the following reasons : (1) The performance (frequency) model is based on the alpha-power law [34], which does not accurately capture operation near the threshold voltage. (2) Memory timing model relies on a conventional 6T SRAM cell, which cannot reliably operate at near-threshold voltages in an efficient manner. (3) VARIUS timing model neglects the impact of (subthreshold) leakage. At NT, the magnitude of off (leakage) current decreases when compared to ST due to lower Vdd, however, the on current decreases more. Hence, the relative impact of leakage current (i.e. the ratio of off current to on current) increases. (4) The only memory failure mode considered is (read) access failures, however, at NT, write stability and write timing becomes critical along with hold failures. The following details the four new aspects.

3.1 Gate Delay Model at NT

To model the gate delay (t_g), VARIUS uses the alpha-power law (Equation 3), 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 region near the threshold voltage accurately. There exists alpha-power law variants [4, 7, 19, 29] that attempt to extend the model to sub-threshold region. Usually, these come along with an increased number of fitting parameters that have no direct physical interpretation. Further, covering sub-threshold region does not necessarily imply that the near-threshold region is properly modeled.

Consequently, in VARIUS-NTC, we use the EKV-based [15] model proposed by Markovic *et al.* [26]. The formula for the on-current is given in Equation 2, where vt depicts thermal voltage and n a process dependent parameter determined by subthreshold characteristics.

$$\begin{aligned} I &= \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{Vgs-Vth}{2\times n\times vt}}+1) \quad \text{where} \quad \mu \propto T^{-1.5} \\ Vth &= Vth_0 + k_{DIBL} \times (Vds - Vdd_0) + k_T \times (T - T_0) \end{aligned} \tag{2}$$

The resulting gate delay from CV/I is shown in Equation 4. Since the EKV model covers all regions of operation, Equation 4 remains equally valid at ST and at NT voltages. In all cases, *Vth* is a function of *Vdd* and temperature as per Equation 5, where Vth_0, Vdd_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 Vth, respectively.

$$t_g \propto \frac{Vdd \times L_{eff}}{\mu(Vdd - Vth)^{\alpha}} \quad \text{where} \quad \mu \propto T^{-1.5} \tag{3}$$

$$t_g \propto \frac{Vdd \times L_{eff}}{\mu \times n \times vt^2 \times ln^2(e^{\frac{Vdd-Vth}{2\times n\times vt}}+1)} \quad \text{where} \quad \mu \propto T^{-1.5} \tag{4}$$

$$Vth = Vth_0 + k_{DIBL}(Vdd - Vdd_0) + k_T(T - T_0) \tag{5}$$

VARIUS-NTC captures variation in gate delay by Equation 4 as a function of the underlying Vth and L_{eff} variation maps, process parameters, and operating conditions.

3.2 Logic Timing Model at NT

VARIUS-NTC characterizes variation-induced shift in the path delay distribution of a pipeline stage following VARIUS' framework. Variation-afflicted path delay distribution, D_{Var}, can serve extraction of either (1) the maximum path delay, from $max(D_{Var})$, to designate a proper clock period; or (2) the timing error rate per cycle at a designated clock period, t_{CLK}, from $1 - cdf_{D_{Var}}(t_{CLK})$. A path causes a timing error iff it is exercised and its delay exceeds the designated clock period. It is assumed that there exists at least one path that has a delay equal to the clock period t_{CLK} if there was no variation.

A purely-logic pipeline stage constitutes of a multitude of paths of various delays. Even if there was no variation, there

exist a specific distribution of path delays, D_{Logic}. The question is how this distribution changes under process variation, to give rise to $D_{VarLogic}$. By fixing the share of wire delay in the path delay by the multiplier k_W,

$$D_{Logic} = D_{Gates} + D_{Wire} \quad \text{and} \quad D_{Wire} = k_W \times D_{Logic} \implies D_{Gates} = (1 - k_W) \times D_{Logic} \tag{6}$$

applies, where D_{Gates} corresponds to the delay of a sequence of gates along a path modulo wires. Variation in interconnect is neglected.

The path delay distribution of a logic stage under variation is given in Equation 7. If the path delay distribution, D_{Logic}, is normalized to the clock period, it would be clustered at 1 due to state-of-the-art design optimization. In this case, Equation 7 can be approximated by Equation 8.

$$\begin{aligned}
& D_{VarLogic} = D_{VarGates} + D_{Wire} = D_{SysGates} + D_{RandGates} + D_{Wire} \\
& D_{SysGates} = k_{Sys} \times D_{Gates} = k_{Sys} \times (1 - k_W) D_{Logic} \\
& D_{RandGates} = D_{Rand} \times D_{Gates} = D_{Rand} \times (1 - k_W) D_{Logic} \\
\implies & D_{VarLogic} = (1 - k_W)(k_{Sys} + D_{Rand}) \times D_{Logic} + k_W D_{Logic}
\end{aligned} \tag{7}$$

$$D_{VarLogic} \approx (1 - k_W)(k_{Sys} \times D_{Logic} + D_{Rand}) + k_W D_{Logic} \tag{8}$$

The systematic variation in D_{Gates}, $D_{SysGates}$, consists of inter- and intra-stage components. The inter-stage systematic component represents the *stage systematic mean*, the average shift in delay over all the paths in the stage, lumped into the multiplier k_{Sys} of D_{Gates}. The intra-stage systematic deviation, on the other hand, remains much smaller due to the high degree of spatial correlation. Already at 45nm, the length of a typical pipeline stage remains less than $0.1\times$ the chip length for a 4-core design [35]. Hence, for typical values of ϕ around $0.5\times$ the chip length, intra-stage systematic deviation can be neglected. VARIUS-NTC does not model intra-stage systematic deviation. Under this assumption, Vth and L_{eff} per transistor – hence gate – along each path in a pipeline stage change in the same direction, moreover, by the same quantity. From this respect, a pipeline stage is assumed atomic – each pipeline stage represents a grid point.

The random variation in D_{Gates}, $D_{RandGates}$, on the other hand, stems from non-correlated, *independent* shifts in transistor – hence gate – delay distribution. The random shift in transistor delay distribution is not necessarily by the same quantity and in the same direction over all transistors. Hence, the random shift in path delay distribution cannot be lumped into a constant coefficient such as k_{Sys} of the systematic variation. If each grid point corresponded to a transistor, each point per sampled die would be characterized by a specific value of the random shift superimposed on the systematic shift. However, operation at transistor granularity is not feasible. Hence, while systematic variation is captured by a *constant* coefficient of D_{Gates} per grid point per die, the coefficient of D_{Gates} corresponding to the random component, D_{Rand}, is modeled analytically as a random variable of a specific distribution.

Putting It All Together

According to Equation 7, to arrive at the variation-afflicted path delay distribution, $D_{VarLogic}$, k_W, D_{Logic}, k_{Sys} and D_{Rand} should be known. k_W, the (average) share of wire delay in the path delay (were there no variation) represents a design-specific constant. It is assumed that D_{Logic} is normalized by t_{CLK} and follows a Gaussian distribution. This is justified by data collected for representative circuitry. Due to state-of-the-art design optimization, the distribution is impulse-like, with majority of path delays accumulated at 1.

Extracting k_{Sys}: Following VARIUS, VARIUS-NTC works on normalized gate delay, t_{norm} as captured by Equation 9.

$$\begin{aligned} t_{norm} &\propto t_g/t_0 \\ t_g &\propto \frac{Vdd \times L_{eff}}{\mu \times n \times vt^2 \times ln^2(e^{\frac{Vdd - Vth}{2 \times n \times vt}} + 1)} \\ t_0 &\propto \frac{Vdd_0 \times L_{eff0}}{\mu \times n \times vt^2 \times ln^2(e^{\frac{Vdd_0 - Vth_0}{2 \times n \times vt}} + 1)} \end{aligned} \tag{9}$$

k_{Sys} is determined by t_{norm}, where Vth and L_{eff} represent variation afflicted parameters of the grid point (i.e. the center of the pipeline stage). This is only (an apparently) "gate" delay formula, however, $D_{VarLogic}$ corresponds to path delay. Recall that k_{Sys} represents systematic variation, and being highly correlated, Vth and L_{eff} of all gates along a path in a pipeline stage would change in the same direction, moreover, by the same quantity; a pipeline stage is assumed atomic from this respect. Hence, the stage (path) delay can be simply represented by $l \times t_g$, with l being the length of the critical path. In normalizing path delays, l disappears (but the quantity of interest still corresponds to paths as opposed to gates).

Extracting D_{Rand}: To determine D_{Rand}, the distribution of this random variable [9] should be known along with its parameters (mean μ and standard deviation σ). Mathematically, the parameters of the distribution can be estimated independent of the underlying distribution by relying on the principles of *functions of iid – independent, identically distributed random variables*.

In this case, each gate's L_{eff} and Vth is affected independently. Hence, the path delay should be formed by composition of l independent gate delays; path delay = $l \times$ gate delay does not hold any more. Instead, path delay should be determined from the sum of l iid random variables.

Assume that G represents the *absolute* delay of a FO4 gate – the random variable corresponding to t_g. Let P be the absolute path delay as composed by the sum of l independent Gs. In this case, $\sigma(P) = \sqrt{l} \times \sigma(G)$, $\mu(P) = l \times \mu(G)$. However, VARIUS-NTC is after the normalized path delay, P_N. The parameters of $P_N = P/(l \times t_0)$ would be $\sigma(P_N) =$

[9] A random variable X is a function that assigns a real number $X(\psi)$ to each outcome ψ in the sample space of a random experiment.

$\sigma(P)/(l \times t_0)$, $\mu(P_N) = \mu(P)/(l \times t_0)$. Hence,

$$\sigma(P_N) = \sqrt{l} \times \sigma(G)/(l \times t_0) \quad \Longrightarrow \quad \sigma(P_N) = \sigma(G/t_0)/\sqrt{l} \tag{10}$$

$$\mu(P_N) = l \times \mu(G)/(l \times t_0) \quad \Longrightarrow \quad \mu(P_N) = \mu(G/t_0) \tag{11}$$

G/t_0 corresponds to the distribution of normalized gate delay, as captured by t_{norm}. G/t_0 represents a function of the random variables L_{eff} and Vth. Orthogonal to the determination of its distribution, the mean and standard deviation can be estimated from formulae for functions of random variables. If, for example, L_{eff} and Vth are assumed to follow a Gaussian distribution [37], Equation 5[10] in [35] can be deployed to extract $\sigma(G/t_0)$ and $\mu(G/t_0)$. For Gaussian Vth and L_{eff}, $\mu(G/t_0)$ can be calculated from systematically shifted Vth_0 and L_{eff0} plugged-in as means, along with σ_{Rand}s of Vth and L_{eff} as standard deviations, into Equation 5 from [35]. D_{Rand} represents an additive term to be superimposed on systematic-variation afflicted D_{Logic}, i.e. $k_{Sys} \times D_{Logic}$, according to Equation 8. The shift in delay due to systematic variation should not be doubly accounted for; this is why once $\mu(G/t_0)$ is extracted, to arrive at the the mean of D_{Rand}, $\mu(D_{Rand})$, the stage systematic mean should be subtracted out [11]. $\sigma(D_{Rand})$, on the other hand, corresponds to $\sigma(G/t_0)$:

$$\begin{aligned} \mu(D_{Rand}) &= \mu(P_N) - k_{Sys} = \mu(G/t_0) - k_{Sys} \\ \sigma(D_{Rand}) &= \sigma(P_N) = \sigma(G/t_0) \end{aligned} \tag{12}$$

How does the picture change in extraction of the stage systematic mean, k_{Sys}***?*** $k_{Sys} = \mu(P_N) = \mu(P)/(l \times t_0)$. For systematic variation, $\mu(P) = l \times \mu(G)$. Hence, $\mu(P_N) = l \times \mu(G)/(l \times t_0) = \mu(G/t_0)$. Since systematic deviation in Vth and L_{eff} is neglected, this $\mu(G/t_0)$ can be directly extracted from t_{norm}, by plugging in systematically shifted Vth_0 and L_{eff0} as obtained from physical variation maps; there is no need to deploy Equation 5 from [35].

Determining the Distribution of D_{Var}: Since, (1) the underlying zero-variation path delay, D_{Logic}; (2) D_{Rand}, to capture random variation, is assumed to follow a Gaussian distribution without loss of generality, $D_{VarLogic}$ follows a Gaussian distribution.

3.3 SRAM Cell at NT

As indicated in Section 2.2.2, the 6-transistor cell of Figure 2(a) requires careful sizing of the transistors, with conflicting requirements to prevent both, read and write failures. In an environment where the impact of process variations on transistors is more considerable such as in NTC, this is difficult. To address this problem, one approach is to power SRAMs at a higher *Vdd* than the processor logic. Unfortunately, this approach is costly, since i) cache memory and logic blocks

[10] A formulae to extract σ and μ of a function of Gaussian random variables. The random variables here correspond to Vth and L_{eff}, hence not necessarily independent and identically distributed. See [22] p.130 for a more general discussion.

[11] Note that, since systematic and random components are independent, they give rise to independent random variables. The cumulative impact is additive. Orthogonal to the underlying distributions, the cumulative mean/variance is the sum of the means/variances.

are often highly interleaved in the layout, ii) extra voltage regulators are required in the platform, and iii) many design, validation, and testing issues can be incurred by the voltage-domain crossing. Moreover, this approach is not 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, since SRAM power does not scale.

Consequently, VARIUS-NTC follows other work [1, 9] in using the 8-transistor cell of Figure 2(b) [8, 27]. This cell is easier to design reliably because it decouples the transistors used for reading from those for writing. Specifically, the two additional transistors N_{RD} and AX_{RD} are only responsible for reads, while the rest supports the writes as in the 6-transistor cell. To read, the read bitline is pre-charged and the read wordline is driven high. If the cell stores a 0, the bitline keeps its value; otherwise, it gets discharged to 0 through N_{RD} and AX_{RD}. The writes proceed as in the 6-transistor cell. In this manner, compared to 6T cells, read and write timing margins can be independently optimized, enhancing the reliability margin significantly with marginal increase in cell area [8].

Of the five failure modes of Section 2.2.2, this design eliminates the read stability (i.e., read upset) failure incurred during read operations, because the cell internal nodes are decoupled from the read bit-line. The rest of the failure modes are possible, but they can be mitigated at a reasonable cost by optimizing some transistors for writes and others for reads, separately.

3.4 SRAM Stability Analysis at NT

For an SRAM cell to be functional, stability failure probability should remain negligible. If the operating Vdd is fixed for a given design, while timing failures can be mitigated by relaxing timing constraints (i.e. by increasing the designated clock period, t_{CLK}), there is no such remedy for stability failures.

3.4.1 Hold Failure Analysis at NT

For a cell storing 0 ($V_R = 0,\ V_L = 1$), at low Vdd, the voltage V_L reduces by construction. When the cell is not accessed, the access transistors, NL and PR are off. Due to leakage through NL and AXL, if V_L reduces enough to reach the V_{SWITCH} of *PR-NR* inverter, the cell content would be distorted. A hold failure occurs when the leakage current through the *NL* and *AXL* transistors in Figure 2(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 gets lost. To model these failures at a given Vdd, VARIUS-NTC uses KCL to compute V_L and V_{SWITCH} at Vdd. V_L is extracted from $I_{PL}(V_L) - I_{NL}(V_L) - I_{AXL}(V_L) = 0$, where

$$
\begin{aligned}
I_{PL}(V_L) &\propto \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{Vdd-Vth}{2\times n\times vt}} + 1) \quad \text{where} \quad Vds = Vdd - V_L \\
I_{NL}(V_L) &\propto \mu/L_{eff} \times T^2 \times e^{-\frac{Vth}{n\times vt}} \quad \text{where} \quad Vds = V_L \\
I_{AXL}(V_L) &\propto \mu/L_{eff} \times T^2 \times e^{-\frac{Vth}{n\times vt}} \quad \text{where} \quad Vds = V_L
\end{aligned}
\tag{13}
$$

V_{SWITCH} is extracted from $I_{PR}(V_{SWITCH}) - I_{NR}(V_{SWITCH}) + I_{AXR}(V_{SWITCH}) = 0$, where

$$\begin{aligned} I_{PR}(V_{SWITCH}) &\propto \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{Vdd-V_{SWITCH}-Vth}{2\times n\times vt}}+1) \quad \text{where} \quad Vds = Vdd - V_{SWITCH} \\ I_{NR}(V_{SWITCH}) &\propto \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{V_{SWITCH}-Vth}{2\times n\times vt}}+1) \quad \text{where} \quad Vds = V_{SWITCH} \\ I_{AXR}(V_{SWITCH}) &\propto \mu/L_{eff} \times T^2 \times e^{-\frac{Vth}{n\times vt}} \quad \text{where} \quad Vds = Vdd - V_{SWITCH} \end{aligned} \tag{14}$$

The threshold voltage is derived from:

$$Vth = Vth_0 + k_{DIBL} \times (Vds - Vdd_0) + k_T \times (T - T_0) \tag{15}$$

The hold failure probability per cell constitutes of $P_{Cell,Hold} = P[V_L(Vdd) - V_{SWITCH}(Vdd) < 0]$. If no redundant cells are provided, the hold failure probability of a line would be $P_{Line,Hold} = 1 - (1 - P_{Cell,Hold})^{line\ size}$, where *line size* denotes 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 deems faulty. The failure probability of cells is assumed independent in this case. This approximation holds valid if systematic deviation within a line can be neglected, since random variation per transistor (hence per cell) is independent.

$$P_{Mem,Hold} = \sum_{i=1}^{number\ of\ lines} \binom{number\ of\ lines}{i} \times P_{Line,Hold}{}^{i} \times (1 - P_{Line,Hold})^{number\ of\ lines-i} \tag{16}$$

$$P_{Mem,Hold} = 1 - (1 - P_{Line,Hold})^{number\ of\ lines} \tag{17}$$

To avoid hold failures, the minimum allowable supply voltage, $Vdd_{MIN,Cell}$, can be obtained by solving $V_L(Vdd_{MIN,Cell}) = V_{SWITCH}(Vdd_{MIN,Cell})$ under variation. Then, $Vdd_{MIN,Line} = max(Vdd_{MIN,Cell})$ applies.

3.4.2 Write Stability Failure Analysis at NT

Write failures are of symmetric nature. Without loss of generality, we focus on a cell that stores a 0 (V_R=0 and V_L=1). To model a write stability failure, VARIUS-NTC 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 such value is above the switching threshold of the *PR-NR* inverter (V_{SWITCH}), then a write stability failure occurs. V_{LW} distribution is computed using Kirchoff's Current Law (KCL) at node L, from $I_{PL}(V_{LW}) - I_{NL}(V_{LW}) - I_{AXL}(V_{LW}) = 0$, where

$$\begin{aligned} I_{PL}(V_{LW}) &\propto \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{Vdd-Vth}{2\times n\times vt}}+1) \quad \text{where} \quad Vds = Vdd - V_{LW} \\ I_{NL}(V_{LW}) &\propto \mu/L_{eff} \times T^2 \times e^{-\frac{Vth}{n\times vt}} \quad \text{where} \quad Vds = V_{LW} \\ I_{AXL}(V_{LW}) &\propto \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{Vdd-Vth}{2\times n\times vt}}+1) \quad \text{where} \quad Vds = V_{LW} \end{aligned} \tag{18}$$

On the other hand, V_{SWITCH} distribution is extracted by using KCL for the *PR-NR* inverter when V_{IN} = V_{OUT} [28], from $I_{PR}(V_{SWITCH}) - I_{NR}(V_{SWITCH}) + I_{AXR}(V_{SWITCH}) = 0$, where

$$\begin{aligned} I_{PR}(V_{SWITCH}) &\propto \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{Vdd-V_{SWITCH}-Vth}{2\times n\times vt}}+1) \quad \text{where} \quad Vds = Vdd - V_{SWITCH} \\ I_{NR}(V_{SWITCH}) &\propto \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{V_{SWITCH}-Vth}{2\times n\times vt}}+1) \quad \text{where} \quad Vds = V_{SWITCH} \\ I_{AXR}(V_{SWITCH}) &\propto \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{Vdd-V_{SWITCH}-Vth}{2\times n\times vt}}+1) \quad \text{where} \quad Vds = Vdd - V_{SWITCH} \end{aligned} \quad (19)$$

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

3.5 SRAM Timing Analysis at NT

Memory access time is extracted in a hierarchical way, starting from cell access time. The random variable to represent variation-afflicted cell access time, $D_{VarCell}$ is usually proportional to the reciprocal of a discharge current. Expressed as a function of Vth and L_{eff}, the discharge current evolves with the type of access (read or write) and the cell content. The SRAM cell remains very small when compared to the correlation range, ϕ [35], hence systematic variance within a cell is neglected. In extracting $\sigma(D_{VarCell})$ and $\mu(D_{VarCell})$, principles of functions of random variables are deployed (as captured by Equation 5 from [35], e.g., if Vth and L_{eff} follow Gaussian distribution): $\sigma(D_{VarCell})$ and $\mu(D_{VarCell})$ are calculated from the closed form expression of the discharge current by plugging in the systematically shifted Vth_0 and L_{eff0} as the means along with the variances of the underlying random Vth and L_{eff} distributions.

Access time to a memory line, $D_{VarLine}$, can then be determined from $max(D_{VarCell})$. A memory access not only involves array components, but also logic components such as decoders, muxes and sense amplifiers. Hence, the random variable to characterize memory access time, $D_{VarAccess}$, represents a linear combination of $D_{VarLine}$ and $D_{VarLogic}$, which captures the variation in the periphery.

The distribution for $D_{VarAccess}$ can serve determination of the probability that the access time to an individual memory line remains less than a designated duration, t_{CLK}, from $cdf_{D_{VarAccess}(t_{CLK})}$; but not the probability of timing error-free access to a given number of memory lines, in other words, the probability of timing error-free execution for a given SRAM block. In [35], it is shown, however, that these probabilities converge for large SRAM blocks.

Observe that the number of error-free lines at a given clock period t_{CLK} follows a binomial distribution [12]: A line is error-free if its access time remains less than t_{CLK}, and suffers from a timing error otherwise. The probability of an individual line being error-free is known as $p = cdf_{D_{VarAccess}(t_{CLK})}$. Let L be the number of lines with access time $< t_{CLK}$, in other words, the number of error-free lines. $cdf_L(l)$, gives the probability of having number of error-free lines

[12] If a random experiment is repeated l times, and L denotes the number of times event E occurs, L is a binomial random variable with range 0,...,l. E in this context denotes the number of error free accesses.

$< l$. $P[low < L < up] = c$ defines a confidence interval for L with c depicting the confidence level. Let $c = 0.9$. Then, $P[low < L < up] = cdf_L(up) - cdf_L(low) = 0.9$. If $cdf_L(up) = 0.95$ and $cdf_L(low) = 0.05$, $up = cdf_L^{-1}(0.95)$ and $low = cdf_L^{-1}(0.05)$. In this case, *a* 90% confidence interval for L constitutes of $[cdf_L^{-1}(0.05), cdf_L^{-1}(0.95)]$. Under the assumption that each line is uniformly accessed, normalizing this interval by the total number of lines leads to a bound for the corresponding cdf – the probability that the access time for a fraction of lines remains less than a given clock period.

3.5.1 SRAM Read Timing at NT

To model a read access failure, VARIUS-NTC computes $D_{VarReadCell}$, which is the random variable capturing time taken to generate a detectable voltage drop on the read bit-line

$$D_{VarReadCell} \propto \frac{1}{I_{AX_{RD}} + \sum I_{STA}} \tag{20}$$

where $I_{AX_{RD}}$ is the bit-line discharge current through the AX_{RD} transistor in Figure 2(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}, should be extracted by solving KCL 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), transistor currents follow from Equation 2:

$$\begin{aligned} I_{AX_{RD}} &= \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{Vdd - V_{RD} - Vth}{2 \times n \times vt}} + 1) \quad \text{where} \quad Vds = Vdd - V_{RD} \\ I_{N_{RD}} &= \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{Vdd - Vth}{2 \times n \times vt}} + 1) \quad \text{where} \quad Vds = V_{RD} \end{aligned} \tag{21}$$

When reading from a cell storing 0 (V_R=0 and V_L=1), N_{RD} is cut-off:

$$\begin{aligned} I_{AX_{RD}} &\propto \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{Vdd - V_{RD} - Vth}{2 \times n \times vt}} + 1) \quad \text{where} \quad Vds = Vdd - V_{RD} \\ I_{N_{RD}} &\propto \mu/L_{eff} \times T^2 \times e^{-\frac{Vth}{n \times vt}} \quad \text{where} \quad Vds = V_{RD} \end{aligned} \tag{22}$$

Then, the probability distribution of $D_{VarReadCell}$ can be attained by applying those of *Vth* and L_{eff} given by the variation model to $\frac{1}{I_{AX_{RD}(V_{RD})} + \sum I_{STA}}$. $D_{VarReadCell}$ can be conservatively set to $max(D_{VarReadCell}(V_R = 0), D_{VarReadCell}(V_R = 1))$. Alternatively, if signal probabilities are known, $D_{VarReadCell}$ would constitute a weighted sum of the underlying probabilities. Following the VARIUS methodology, the maximum of $D_{VarReadCell}$ over all 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.

3.5.2 SRAM Write Timing at NT

Given a cell without write stability failure, VARIUS-NTC 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.

$$\begin{aligned} D_{VarWriteCell} \propto \frac{1}{I_L} &= \int_{Vdd}^{V_{SWITCH}} dv_L/i_L(v_L) \\ i_L(v_L) &= i_{PL}(v_L) - i_{NL}(v_L) - i_{AXL}(v_L) \end{aligned} \tag{23}$$

where I_L is the discharge current at node L during the write, obtained following [28]. With $i_L(v_L)$ representing a function of Gaussian random variables Vth and L_{eff} under process variation,

$$
\begin{aligned}
i_{PL}(v_L) &\propto \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{Vdd-Vth}{2\times n\times vt}}+1) \quad \text{where} \quad Vds = Vdd - v_L \\
i_{NL}(v_L) &\propto \mu/L_{eff} \times T^2 \times e^{-\frac{Vth}{n\times vt}} \quad \text{where} \quad Vds = v_L \\
i_{AXL}(v_L) &\propto \mu/L_{eff} \times n \times vt^2 \times ln^2(e^{\frac{Vdd-Vth}{2\times n\times vt}}+1) \quad \text{where} \quad Vds = v_L
\end{aligned}
\tag{24}
$$

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

4 Evaluation Setup

4.1 Many-core Architecture and Technology

For operation at the nominal voltage, technology scaling brings along frequency benefits, while for NTC the frequency requirement remains much modest, but the share of static power rapidly increases with lower Vdd. Static power, on the other hand, increases each technology generation. It can be argued that an older generation conforms better to NTC operation, however, in [36] it is demonstrated that for performance-critical application spaces, new technologies are favorable due to the reduced power consumption when active, due to the scaled Vdd. Relying on this observation, we evaluate the model using an 11nm NTC chip with 288 cores. Each core is a single-issue in-order engine. The chip is organized in clusters, where each cluster has 8 cores, per-core private L1 caches in the core-private memories, and a shared L2 cache in the cluster memory. The technology parameters and density scaling estimation are derived from ITRS data [21]. Table 2 shows the architectural and technological details. All the parameters in the table that are not labeled with STC refer to the NTC environment.

System Parameters	
Technology node: 11nm Num. Cores: 288 Num. Clusters: 36 Num. Cores/Cluster:8	P_{MAX}: 100W Share of static power: 25% at $Vdd_{NOM}(STC)$ $T_{NOM} = 85$°C Chip area: 24mm x 24mm
Variation Parameters	
Correlation range: $\phi = 0.5$ Total $(\sigma/\mu)_{Vth} = 25\%$ Systematic $(\sigma/\mu)_{Vth} = 17.68\%$ Random $(\sigma/\mu)_{Vth} = 17.68\%$	Sample size: 100 Total $(\sigma/\mu)_{Leff} = 12.5\%$ Systematic $(\sigma/\mu)_{Leff} = 8.84\%$ Random $(\sigma/\mu)_{Leff} = 8.84\%$
Technology Parameters	
Vdd_{NOM} at STC = 0.8V Vth_{NOM} at STC = 0.3V f_{NOM} at STC = 5GHz $k_T = -1.5mV/K$	Vdd_{NOM} at NTC = 0.497V Vth_{NOM} at NTC = 0.338mV f_{NOM} at NTC = 1.2 GHz $k_{DIBL} = -150mV/V; n = 1.5$
Memory Specification	
Core-private memory: 32KB WT, 4-way, 2 cyc access, 64B line	Cluster memory: 2MB WB,16-way, 8 cyc access, 64B line

Table 2: Architectural and technological parameters.

To set the voltages and frequencies at 11nm under STC and NTC, we proceed as follows: Guided by ITRS data, we first

determine the supply and threshold voltages at STC as Vdd_{NOM}=0.8V and Vth_{NOM}=0.3V. We set the STC frequency to be a conservative f_{NOM}=5GHz. We then target an NTC environment that is 10 times slower than the baseline STC, as suggested by Dreslinski *et al* [13]. To ensure that the operating *Vdd* for NTC renders a negligible timing error rate, we rely on VARIUS-NTC. With these parameters, when all the clusters are active, the NTC chip consumes a maximum of P_{MAX}=100W.

4.2 Simulation Infrastructure

We interfaced Pin [24] over a user-level pthreads library [30] to SESC [32]. The power analysis relies on Wattch [5] and Cacti [39] for array structures. The scheduler is implemented in R and consumes (1) activity factors to estimate dynamic power, (2) per-application IPC as a function of frequency from micro-architectural simulation. For static power estimation, we use Equation 25, which applies to both STC and NTC.

$$P_{STA} = Vdd \times I_{STA}, \quad \text{where} \quad I_{STA} \propto \mu/L_{eff} \times n \times vt^2 \times e^{-\frac{Vth}{n \times vt}} \tag{25}$$

5 Evaluation

5.1 Model Validation

To validate VARIUS-NTC's t_g, Vth, and I_{STA} models (i.e. Equation 4, 5 and 25), we begin with the validation of the Vth model, since t_g and I_{STA} are strongly dependent on Vth. For a 12nm Predictive Technology Model (PTM) [13], the Vth values generated by VARIUS-NTC, the BSIM analytical model [6], and HSPICE are compared Vth values from VARIUS-NTC closely track the values from either HSPICE or BSIM with less than 0.4% error over the designated Vdd range. The main source of discrepancy between HSPICE and VARIUS-NTC is the accuracy of modeling the DIBL effect.

By deploying Vth values for the given voltage range from both, VARIUS-NTC and BSIM models, t_g and I_{STA} are compared against HSPICE measurements of a FO4 inverter chain. VARIUS-NTC follows HSPICE analysis within 10% of error for the given Vdd range. Since the leakage is exponentially dependent on Vth, the small error in Vth results in a large difference between the model and measurement.

5.2 Impact of Process Variations in an NTC Environment

We examine the impact of process variations on the *f* and power at NTC and provide a comparison to STC. Logic blocks (the core pipelines), small memories (the per-core, local memories) and large memories (the cluster memories) are separately considered, since logic and memory blocks have different critical path structures, and small and large memories have different critical path timing distributions. The frequency of a block is given by the path delay at 3σ of its distribution – the maximum frequency that the block can support at the designated Vdd_{NOM}; the static power is given by the sum of the static power of all the components within the block.

To estimate the variation of a parameter, we compute the ratio between the maximum and minimum values of the

[13] 12nm PTM is not publicly available yet, but the authors obtained the model in advance by the courtesy of Yu Cao, developer of the PTM models at Arizona State University [31].

parameter on a given chip. We consider five variation scopes: *pipeline intra-cluster, pipeline inter-cluster, local memory intra-cluster, local memory inter-cluster*, and *cluster memory inter-cluster*. For example, the pipeline intra-cluster variation is computed by measuring the frequency of each core pipeline, then calculating, in each cluster *i*, the ratio r_i between the highest and lowest pipeline core frequency in the cluster, and finally computing the mean of the r_i for all *i* in the chip. The pipeline inter-cluster variation is computed by taking the r_i generated above and computing, for the chip max(r_i)/min(r_i). The local-memory intra-cluster and inter-cluster variations are computed in a similar way. Finally, the cluster-memory inter-cluster variation is the ratio between the highest and lowest frequency of all the cluster memories. We sample 100 chips to estimate the variation over each scope.

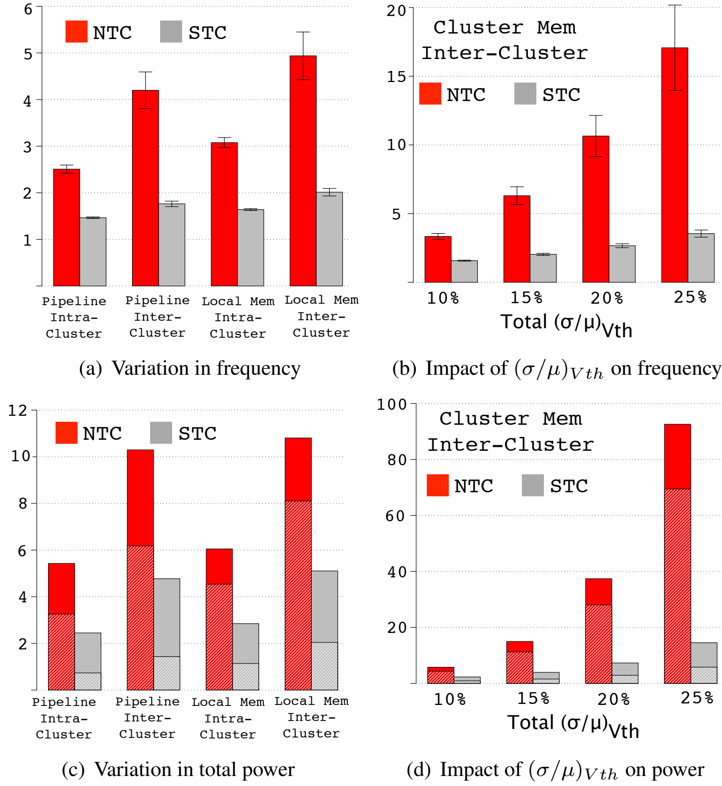
5.2.1 Frequency Variation

Figure 7(a) shows frequency variation for four of the five variation scopes, both, under NTC and STC, along with 99% confidence intervals. Total $(\sigma/\mu)_{Vth}$ is set to 25%. Starting with NTC, the pipeline intra-cluster variation is about 2.5×, hence there is a cluster where one core can run 2.5× faster than another. The pipeline inter-cluster variation is about 4.15×, which indicates that the cluster with the highest frequency imbalance has an imbalance that is 4.15× higher than the one with the least imbalance. This data shows that the system is highly heterogeneous. A similar picture is attained for the local memory frequencies, with a higher imbalance than observed for the pipeline. This is due to the memory cells being more prone to variation, as induced by the smaller size transistors to satisfy density requirements.

The frequency variation for cluster memory inter-cluster is depicted in Figure 7(b), over a representative range of total $(\sigma/\mu)_{Vth}$. For $(\sigma/\mu)_{Vth} = 25\%$, there is one cluster memory that is $\approx$ 17 times faster than another. The large size of these memories causes those that fall in a slow section of the chip to be quite slower than the others. As the most sensitive scope to process variation, cluster memory frequency variation increases exponentially with increasing total $(\sigma/\mu)_{Vth}$. At $(\sigma/\mu)_{Vth} = 25\%$, the average cluster memory inter-cluster variation is about 4× in STC as opposed to about 17× in NTC. This is because a fixed amount of *Vth* and *Leff* variation affects transistor delay more at low voltages (namely, at NTC) than at STC [14].

5.2.2 Power Variation

Figure 7(c) shows variation in total power for four of the five variation scopes, both, under NTC and STC. Total $(\sigma/\mu)_{Vth}$ is set to 25%. At NTC, the pipeline intra-cluster variation is about 5.7×, where pipeline inter-cluster variation is more than 10×. The imbalance is exacerbated for local memories. The shaded stacks point to the fraction of variation that stems from static power. Variation in dynamic power closely tracks the variation in frequency, since at a constant *Vdd*, dynamic power is directly proportional to *f*. On the other hand, the variation in static power is even larger than variation in frequency. This is because the static power of a block is given by the sum of the static power of all the transistors in the block, while the frequency is given by its slowest path.



(a) Variation in frequency (b) Impact of $(\sigma/\mu)_{Vth}$ on frequency

(c) Variation in total power (d) Impact of $(\sigma/\mu)_{Vth}$ on power

Figure 7: Variation profile for frequency (a), for power (c); the impact of $(\sigma/\mu)_{Vth}$ on frequency variation (b), on total power variation (d).

Power variation for cluster memory inter-cluster is depicted in Figure 7(d), over total $(\sigma/\mu)_{Vth}$. For $(\sigma/\mu)_{Vth} = 25\%$, there is one cluster memory that is ≈ 90 times more power hungry than another. Cluster memory power variation increases exponentially with increasing total $(\sigma/\mu)_{Vth}$, at a faster pace than the frequency variation. Under STC, the same amount of process variation affects the power less than in NTC. At $(\sigma/\mu)_{Vth} = 25\%$, the average cluster memory inter-cluster variation is about $18\times$ in STC as opposed to about $90\times$ in NTC. Variation under NTC is not only exacerbated due to higher variation in frequency, hence dynamic power, when compared to STC. It is the share of static power significantly increasing at NTC. Otherwise, both, at NTC and STC, variation in static power is higher than the variation in dynamic power. According to Figure 7(c), under STC, static power remains less than $\approx 30\%$ for logic, and $\approx 40\%$ for local memories, of the total power. At NTC, however, the share of static power jumps to $\approx 63\%$ for logic, and $\approx 77\%$ for local memories.

6 Conclusions

To cope with process variations at NT voltages, this report introduced a ($\mu-$) architectural model of process variation to work under NTC. The model is based on an existing model of process variation tailored for STC. The main extensions involved using a different gate-delay model and SRAM cell, modeling new SRAM cell failure modes, and including leakage currents in the timing/stability analysis. Our results show that the expected process variations at 11nm induce significantly larger changes in core frequency in an NTC environment than in an STC one.

References

[1] J. Abella et al. High-Performance Low-Vcc In-Order Core. In *Int. Symp. on High Performance Computer Architecture*, January 2010.

[2] K. Bernstein et al. High-Performance CMOS Variability in the 65-nm Regime and Beyond. In *IBM Journal of Research and Development*, July/September 2006.

[3] S. Borkar et al. Parameter variations and impact on circuits and microarchitecture. In *Design Automation Conference*, 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*, 1999.

[5] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. *SIGARCH Comput. Archit. News*, 28(2), 2000.

[6] BSIM:. *http://www-device.eecs.berkeley.edu/˜bsim/BSIM4/BSIM460*.

[7] Y. Cao and L. T. Clark. Mapping statistical process variations toward circuit performance variability: an analytical modeling approach. In *Conference on Design Automation*, 2005.

[8] L. Chang et al. An 8T-SRAM for Variability Tolerance and Low-Voltage Operation in High-Performance Caches. *Journal of Solid-State Circuits*, 43(4), April 2008.

[9] L. Chang et al. Practical strategies for power-efficient computing technologies. *Proceedings of the IEEE*, 98(2):215–236, February 2010.

[10] Y. Cheng et al. *MOSFET Modeling and Bsim3 User's Guide*. Kluwer Academic Publishers, 1999.

[11] R. Dennard et al. Design of Ion-Implanted MOSFETs with Very Small Physical Dimensions. In *Journal of Solid-State Circuits*, October 1974.

[12] J. Donald and M. Martonosi. Power efficiency for variation-tolerant multicore processors. In *Int. Symp. on Low power Electronics and Design*, pages 304–309, 2006.

[13] R. G. Dreslinski et al. Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits. *Proceedings of the IEEE*, 98(2):253–266, February 2010.

[14] M. Eisele et al. The impact of intra-die device parameter variations on path delays and on the design for yield of low voltage digital circuits. *IEEE Transactions on VLSI Systems*, 5(4), dec. 1997.

[15] C. C. Enz et al. An analytical mos transistor model valid in all regions of operation and dedicated to low-voltage and low-current applications. *Analog Integr. Circuits Signal Process.*, 8(1):83–114, 1995.

[16] S. Herbert and D. Marculescu. Characterizing chip-multiprocessor variability-tolerance. In *Conf. on Design Automation*, 2008.

[17] M. Horowitz et al. Scaling, Power, and the Future of CMOS. In *Int. Electron Devices Meeting*, December 2005.

[18] E. Humenay et al. Impact of Process Variations on Multicore Performance Symmetry. In *Conf. on Design, Automation and Test in Europe*, 2007.

[19] 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*, 2002.

[20] International Technology Roadmap for Semiconductors. 2008 Update.

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

[22] A. Leon-Garcia. *Probability and Random Processes for Electrical Engineering*. Addison Wesley, 2nd edition, 1994.

[23] X. Liang and D. Brooks. Mitigating the impact of process variations on processor register files and execution units. In *Int. Symp. on Microarchitecture*, pages 504–514, 2006.

[24] C.-K. Luk et al. Pin: building customized program analysis tools with dynamic instrumentation. In *Conf. on Programming Language Design and Implementation*, pages 190–200, 2005.

[25] D. Marculescu and E. Talpes. Variability and energy awareness: a microarchitecture-level perspective. In *Design Automation Conference*, pages 11–16, 2005.

[26] D. Markovic et al. Ultralow-power design in near-threshold region. *Proceedings of the IEEE*, 98(2):237–252, Feb. 2010.

[27] Y. Morita et al. An Area-Conscious Low-Voltage-Oriented 8T-SRAM Design under DVS Environment. In *Symposium on VLSI Circuits*, 14-16 2007.

[28] S. Mukhopadhyay et al. Modeling of failure probability and statistical design of sram array for yield enhancement in nanoscaled cmos. *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 24(12):1859 – 1880, December 2005.

[29] M. Orshansky, J. Chen, and C. Hu. Direct sampling methodology for statistical analysis of scaled cmos technologies. *Transactions on Semiconductor Manufacturing*, 12(4):403 –408, Nov 1999.

[30] H. Pan, K. Asanović, R. Cohn, and C.-K. Luk. Controlling program execution through binary instrumentation. *SIGARCH Comput. Archit. News*, 33(5):45–50, 2005.

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

[32] J. Renau et al. SESC simulator, January 2005. http://sesc.sourceforge.net.

[33] B. F. Romanescu et al. Quantifying the impact of process variability on microprocessor behavior. In *Workshop on Architectural Reliability*, December 2006.

[34] T. Sakurai and A. Newton. Alpha-Power Law MOSFET Model and Its Applications to CMOS Inverter Delay and Other Formulas. In *Journal of Solid-State Circuits*, April 1990.

[35] S. Sarangi et al. Varius: A model of process variation and resulting timing errors for microarchitects. *Trans. on Semiconductor Manufacturing*, 21(1):3–13, February 2008.

[36] M. Seok, D. Sylvester, and D. Blaauw. Optimal technology selection for minimizing energy and variability in low voltage applications. In *International symposium on low power electronics and design*, pages 9–14, 2008.

[37] A. Srivastava et al. *Statistical Analysis and Optimization for VLSI: Timing and Power*. Springer, 2005.

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

[39] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. Jouppi. CACTI 5.1. Technical Report. Technical Report HPL-2008-20, Hewlett Packard Labs, April 2008.

Appendix: A Closer Look into the Logic Timing Model

Extraction of D_{Rand}***'s parameters***: The standard deviation of this distribution, $\sigma(D_{Rand})$, can be estimated from the random standard deviation per gate. Each gate's random variation is independent, since the underlying random Vth/L_{eff} variation is independent and gates do not share transistors. For a path of length l, D_{Rand} corresponds to the sum of l iid random variables – FO4-gate delays. The variance of the sum of l iid random variables is the sum of their variances [22]. Hence, for the random component, $\sigma(D_{Rand}) = \sigma(Rand, FO4Gate) \times \sqrt{l}$ applies. To extract $\sigma(Rand, FO4Gate)$, random-variation induced gate delay distribution, $D_{Rand,FO4Gate}$ can be calculated from Equation 4.

1. To extract the σ and μ of the random-variation induced FO4-gate delay distribution per pipeline stage, $D_{Rand,FO4Gate}$, principles on functions of random variables is deployed. To achieve this, Equation 4 – representing a function of the random variables Vth and L_{eff} – is used, with the zero-mean Gaussian random component of Vth and L_{eff} superimposed on the systematically shifted Vth_0 and L_{eff0} (recall that random and systematic variation are independent). This is because, a pipeline stage represents an atomic grid point, across which a constant systematically shifted Vth and a constant systematically shifted L_{eff} applies. On top of this, the random variation in Vth and L_{eff} is superimposed as a zero-mean Gaussian distribution.

2. The resulting gate delay mean from 1. does not reflect $\mu_{D_{Rand,FO4Gate}}$; in fact it is the gate delay random mean superimposed on the gate delay systematic mean. The resulting variance, on the other hand, is purely random, since the systematic Vth/L_{eff} variance across a pipeline stage is neglected. (If this was not the case, we should have used the sum of random and systematic variances of Vth and L_{eff} in step 1., instead of the random variances.)

3. Finally, extraction of the (random-variation induced) path delay distribution D_{Rand}'s parameters follows from the observation that path delay's random component represents the sum of iid gate delays – $D_{Rand,FO4Gate}$s – as characterized in step 2.