Reduction of Variation-Induced Energy Overhead in Multi-Core Processors

Nigel Drego, *Member, IEEE,* Anantha Chandrakasan, *Fellow, IEEE,* Duane Boning, *Fellow, IEEE,* and Devavrat Shah, *Member, IEEE*

***Abstract*—Core-to-core variability in future many-core chip multi-processors (CMPs) negatively impacts energy. Underperforming cores necessitate increasing the system voltage to maintain homogeneous core performance, introducing an energy overhead. Multiple supply voltages can be used to mitigate the impact of delay variation in CMPs. In this paper, we carefully analyze the use of a local search algorithm to pick near-optimal supply voltages while meeting a fixed performance target. With two system voltages, we prove our algorithm selects the global optimum and in the more general multiple voltage case we develop quantitative bounds. Using a custom simulation methodology on a real processor core, we show that two system voltages provide the most incremental benefit, reducing the energy overhead relative to a single voltage by 59–75% and total energy by 6–16%. Additionally, the worst 5–15% of cores in such systems necessitate increasingly larger amounts of incremental energy for a constant incremental performance gain. Therefore, turning off or disabling these cores is beneficial to a joint performance-energy metric.**

***Index Terms*—Delay measurement, digital circuits, spatial correlation, variation.**

I. Introduction

THE tradeoffs between mitigating performance variability and other key product metrics are increasingly complex. In general, most proposed and implemented variation mitigation techniques involve tradeoffs between die area, design complexity, power, cost and yield, making evaluation of any technique a difficult, multi-dimensional problem. In the context of multi-core processors, mitigating variation becomes non-trivial in scope and complexity. Nevertheless, effective management of variation at this level is critical, as high-performance multi-core processors, in which power and variation are intricately linked, are expected to scale to many tens if not hundreds or thousands of cores per die. In such systems, core-to-core frequency variations will arise due to underlying process variation.

Manuscript received February 19, 2010; revised August 26, 2010 and November 30, 2010; accepted December 17, 2010. Date of current version May 18, 2011. This paper was recommended by Associate Editor S. Vrudhula.

N. Drego is with PDF Solutions, San Jose, CA 95110 USA (e-mail: ndrego@mtl.mit.edu).

A. Chandrakasan, D. Boning, and D. Shah are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: anantha@mtl.mit.edu; boning@mtl.mit.edu; devavrat@mit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2010.2102431

Discussions with computer architects reveal that both architects and operating system designers value operating frequency homogeneity at the system level unless the cost of ensuring it is too high. If this is the case, solutions such as factored operating systems, where each core runs an OS servlet and shields software from underlying core differences [1], and self-aware software capable of detecting a core's power and performance state through a variety of hardware sensors, have been proposed. However, the value of homogeneity is believed to be greater than software solutions, if core-to-core performance variation lies in the range of 20–50%.

To ensure homogenous core frequencies, a number of techniques might be employed, including increasing device sizes, error detection and correction (e.g., error-correcting codes used in SRAMs, razor flip-flops [2]), asynchronous architectures, lowering clock frequencies, increasing the system voltage level [Fig. 1(b)] and providing each core its own voltage [Fig. 1(d)]. However, all have tradeoffs that must be considered.

The above list is by no means exhaustive, but any of these solutions includes significant undesirable components, most often in the form of increased power dissipation. Bowman *et al.* showed that 31–53% additional power/energy is necessary to overcome the impact of process variation at the 50 nm technology node, if voltage scaling is used to maintain performance over the nominal case of no variation [3]. In justifying the push to thousand-core processors, Borkar acknowledges that fine-grain power management is necessary to fit these processors within the desired power envelopes [4]. For design and power delivery simplicity, Borkar suggests using two voltage supplies such that a core operates at either a frequency, f, or $f/2$ and uses the lower voltage when operating at $f/2$.

This paper undertakes reduction in the energy required to cope with variability in massively parallel multi-core processors by introducing additional, optimally-chosen, power-supply voltages. We begin in Section II by evaluating related work in the field. In Section III we mathematically define the concept of "variation-induced energy overhead," formulate the problem to be solved, and provide an algorithm capable of efficiently solving this problem. We next describe our simulation methodology, in Section V, to efficiently simulate energy savings on a hypothetical one-thousand core processor. Section VI demonstrates reduction in the energy overhead of this processor by 59–75% and further shows that turning off some number (5–15%) of the worst performing cores

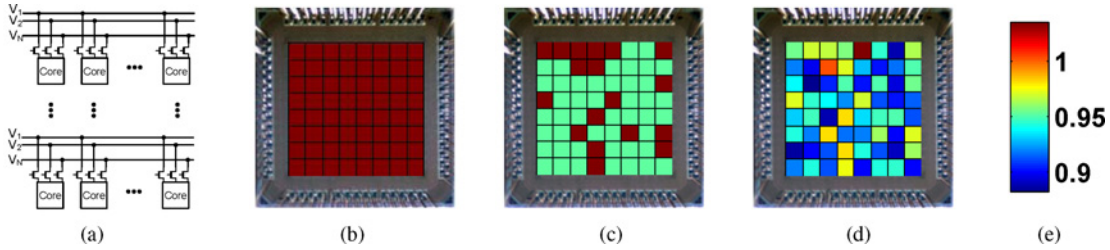

Fig. 1. Block diagram of a CMP with each core able to select from N voltages and example core voltages, with normalized energies, to meet a performance constraint. (a) CMP block diagram. (b) $N = 1$, $E = 1.23$. (c) $N = 2$, $E = 1.083$. (d) $N = N_{\text{core}}$, $E = 1$. (e) Voltage scale.

benefits a joint performance-energy metric. Finally, we address practical considerations and future enhancements to this paper in Section VII and conclude in Section VIII.

II. Related Work

Most prior work in this area has focused on various aspects of power reduction under latency constraints through static voltage scheduling at the gate level [5], [6], [7], [8], [9], [10]. Dynamic voltage scheduling has also been studied heavily as evidenced by the works of [11], [12], [13]. However, none of these works consider process variation in their formulations. Most recently, Liang *et al.* used voltage interpolation at the gate or pipeline level to mitigate process variability [14]. Voltage interpolation at this level has the potential to considerably reduce energy consumption, even more so than at the core level, but requires many power-multiplexers (as many as one per gate) and selection of the best combination of power-mux settings from possibly hundreds of combinations.

Marculescu *et al.* included process variability in the context of heterogeneous blocks in an embedded application that together must meet some latency constraint [15]. Each block is a voltage/frequency island and the optimal voltages and frequencies for islands are solved for. This is not readily applied to the problem of homogeneous cores in a CMP. A similar approach is taken by Stefano *et al.* in [16] in dividing a single core, pipelined design into multiple voltage islands to mitigate process variation.

Humenay *et al.* explored the impact of variation on core-to-core frequencies and showed that adaptive voltage scaling can aid in reducing core-to-core frequency scaling but only consider the case of each core having its own unique frequency [17]. Donald *et al.* also explored core-to-core variation in frequency and proposed allowing the system to turn off cores if the additional power consumed by the core is higher than a proposed metric [18]. In addition to adding system power-supply voltages, we also investigate turning off or disabling cores in Section VI-C, based on a joint performance-energy metric.

III. Mitigation Strategy: Multiple Power-Supply Voltages

We tackle the combined power and variability problem in generic multi-core processors with the introduction of one or more additional power-supply voltages to the system [Fig. 1(c)]. Specifically, we go beyond Borkar's suggestion of two system voltages, as we focus on efficient selection of the optimal value of a vector of power-supply voltages whereby *each core* of a chip multi-processor (CMP) is assigned a single voltage from within the vector in order to minimize total chip energy while meeting performance (frequency) *and* yield constraints. This vector is unique to an individual CMP and is computed during test after each core has been characterized to determine the core distribution for that CMP.

Given the near certainty with which we can expect variation to impact large multi-core processors, we will define the "variation-induced energy overhead" as the energy required over and above that when an ideal mitigation solution, such as individual core voltages, is used (E_{ideal}). Mathematically, this can be defined as $\frac{E-E_{\text{ideal}}}{E_{\text{ideal}}}$. Simulations on a RAW processor core [19] ported to the 45 nm technology node show this can be 20% or more, depending on the amount of variation, if a single system voltage is simply scaled upward to account for the worst performing core. Though not as pessimistic as predicted by Bowman, this magnitude of power/energy overhead is large enough to warrant more efficient solutions.

In this section, we formulate an analytic approach and provide an efficient iterative algorithm to find good power-supply vector values. When the vector is composed of only two voltages ($N = 2$), we prove there is only a single optimum, and we provide an efficient mathematical formulation for finding the optimum. When the vector contains more than two voltages ($N > 2$), the algorithm utilizes the $N = 2$ case to find local optima. Quantitative bounds on the performance are also formulated, and we qualitatively show that our algorithm behaves according to the derived bounds.

A. Problem Formulation

Energy in a multicore processor is computed as the sum of the individual core energies plus any shared resources, as shown in (1), where N_{Core} is the total number of cores on chip. The individual energy/operation of each core is shown in (2), where C_i is the effective switched capacitance in the core and T is the cycle time required to complete an operation

$$E = \sum_{i=1}^{N_{\text{Core}}} E_i + E_{\text{shared}} \tag{1}$$

$$E_i = E_{\text{dyn}_i} + E_{\text{leak}_i} = C_i V_{DD}^2 + I_{\text{leak}_i} V_{DD} T. \tag{2}$$

The minimum cycle-time achievable by a core is a function of power-supply voltage and can be expressed as in (3), where K and V_T are parameters determined by the critical path in

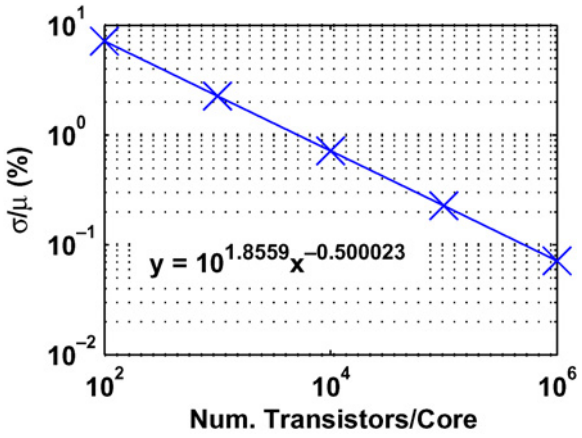

Fig. 2. Variation in core leakage currents reduces to << 1% with even moderate core size.

the design and are subject to variation. α is a technology-dependent parameter

$$T = \frac{KV_{DD}}{(V_{DD} - V_T)^{\alpha}}. \tag{3}$$

Before continuing with further formulation of the problem, we list some assumptions. Specifically, we assume a high-performance CMP, leading to the following assumptions.

1) E_{shared} is the energy of shared caches, I/O and other peripheral circuit blocks surrounding the processor cores. Many, if not all, of these blocks have their own power-supplies separate from the processor cores. As a result, modifications in how the cores are powered do not, to first order, affect this component and will be omitted in the following analysis.
2) For simplicity, we treat C_i as a constant. While the workload may vary from core to core, the total capacitance in a core is the sum of millions or more individual capacitances. When accounting for aggregate variation, summation typically reduces variation in C_i due to averaging of random variation.
3) Similarly, due to the large number of transistors per core, the variance in leakage current (I_{leak_i}) per core is small and can be treated as a constant across all cores. Fig. 2 illustrates this, showing that the relative variation in core leakage currents is inversely proportional to the square root of the number of transistors in the core.

However, E_{leak_i} does have a strong dependence on voltage and this must be captured as different voltages are used for different cores. Rather than modeling I_{leak_i} with the typical exponential function ($I_{\text{leak}_i} = I_o e^{\frac{V_{DD}-V_T}{nV_{\text{th}}}}$), which would complicate the ensuing math, we use the simpler assumption that the leakage energy is a constant proportion of the total energy [20]: $E_{\text{leak}_i} \approx \beta E_{\text{dyn}_i}$, where β is constant across all cores—consistent with the above assumption that the leakage current variance is small. Modeling leakage energy in this manner amounts to a linearization of the exponential model, so $I_{\text{leak}} \propto V_{DD} \Rightarrow E_{\text{leak}} \propto V_{DD}^2$. Since the optimal voltages for each core do not deviate much more than 50–100 mV from the nominal voltage, this linearization is acceptable. Furthermore, with technology improvements and innovation in power-saving techniques, leakage energy as a fraction of total energy has reduced from 30% to 10–15% in state-of-the-art multi-core processors [21], [22], [23], [24]. Simulations of individual gates and paths with the 45 nm PTM models used in this work (Section V) show β values as high as 7% for the frequencies of interest.

With these assumptions, (2) is now simplified to

$$E_i = E_{\text{dyn}_i} + \beta E_{\text{dyn}_i} = (1+\beta)\, CV_{DD}^2. \tag{4}$$

Our goal is to *minimize* E subject to both yield and performance constraints. In particular, we wish to minimize E such that some fraction of the cores, $0 \le y_o \le 1$, in a CMP achieve a certain minimum frequency (maximum delay) of operation, $f_{\min}$. Mathematically, for any CMP, independent of other CMPs, we wish to achieve the following minimization:

$$\begin{array}{ll} \text{minimize} & E \\ \text{subject to} & y = y_o \end{array}$$

where the core yield, y, is defined as follows. A core is labeled "acceptable" if its frequency of operation for a given voltage, $f_i(V)$, is greater than or equal to the constraint, $f_{\min}$. Otherwise, it is unacceptable

$$A_i \overset{\text{def}}{=} \begin{cases} 1, & \text{if } f_i(V) \ge f_{\min} \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The summation of A_i over all cores gives the number of acceptable cores

$$N_{\text{Acc}} = \sum_{i=1}^{N_{\text{Core}}} A_i. \tag{6}$$

And finally, the core yield, y, of an individual CMP is the number of acceptable cores divided by the total number of cores, which must be equal to the yield constraint

$$y = \frac{N_{\text{Acc}}}{N_{\text{Core}}} = y_o. \tag{7}$$

These constraints can be achieved by allowing each core to select its own minimum power-supply voltage, denoted by $V_{i,\min}$, so that $f_i(V_{i,\min}) = f_{\min}$, as shown in Fig. 1(d). This is the case where we have as many supply voltages as we have cores, $N = N_{\text{Core}}$. However, this solution introduces the substantial overhead of having an on-chip DC-DC converter for each core. Instead, we can use a smaller number of power-supply voltages ($N << N_{\text{Core}}$), as shown in Fig. 1(c), and attempt to minimize the energy in such a case.

B. Incorporating Process Variation

We will now describe how process variation is accounted for in this problem formulation. As defined above, $V_{i,\min}$ is the minimum voltage required for core i such that all cores operate at the desired frequency, $f_{\min}$, and provide a homogeneous view at the system level ($f_i = f_{\min}, \forall i$). Core-to-core frequency variation, a symptom of underlying process variation, will result in a distribution of $V_{i,\min}$'s, represented by $f(V_{\min})$ as illustrated in Fig. 3 (grey region in background, left axis), which describes the probability that a core requires $V_{\min}$ to operate at the desired frequency. The cumulative distribution

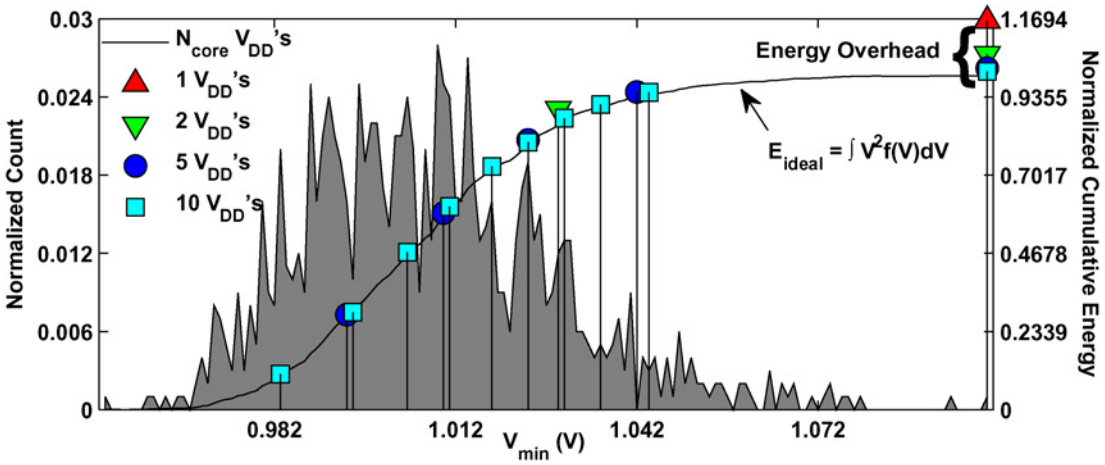

Fig. 3. Example $V_{\min}$ distribution and discretization for 1K-core RAW processor based CMP. The variability-induced energy overhead is drastically reduced as more system voltages are added.

function (CDF), $F(V_{\min})$, is the fraction of cores that can successfully use $V_{\min}$. As such, using only N voltages and combined with the assumptions mentioned above, the total energy we wish to minimize is

$$E_N = (1+\beta)\,CN_{\text{Core}}V_1^2F(V_1) + (1+\beta)\,CN_{\text{Core}}\sum_{i=2}^{N}V_i^2\left[F(V_i)-F(V_{i-1})\right] \quad (8)$$

where the first term accounts for the energy of all cores that can successfully use V_1 and the term under summation accounts for the energy of cores that can successfully use V_i since $F(V_i) - F(V_{i-1})$ is the fraction of cores using V_i.

The equation above amounts to a discretization of the second moment of the distribution, because as $N \to \infty$, and the voltages are spaced infinitesimally close to each other, E_N reduces to $E_{\text{ideal}} = (1+\beta)\,CN_{\text{Core}}\int V^2 f(V)dV$. As we do not have an infinite number of voltages, we depict this in Fig. 3 with the solid black curve representing E_{ideal} for $N = N_{\text{Core}}$, the cumulative energy[1] that would be required if every core were assigned its associated $V_{i,\min}$. By using fewer than N_{Core} voltages, the energy curve is discretized, with voltage placements (computed using the algorithm presented in the following section) at the positions indicated by the dashed lines. This is similar to approximating a continuous integral by partitioning the interval and using Reimann sums of finite subintervals (distance between dashed lines of the same color). However, in this case the E_{ideal} curve is a lower bound, as each core must be provided a voltage greater than or equal to its minimum required voltage. Performing this discretization, we see that using only one voltage (red upper-pointing triangle) results in the greatest overhead (nearly 17% in this example), or worst approximation of the E_{ideal} curve. Increasing the number of voltages results in successively better approximations, and with ten voltages (light blue squares), the energy overhead is tiny.

[1] $(1+\beta)$, C and N_{Core} are normalized out due to normalizing all energies to the case of E_{ideal}.

Using both (8) and Fig. 3, we see that to meet a core yield constraint, $y = y_o$, we should pick V_N such that it satisfies $V_N = F^{-1}(y_o)$, where $F^{-1}(y)$ is the inverse cumulative distribution function. In this paper, we use a Gaussian distribution function and its associated CDF due to relative ease of analysis compared to other distribution functions. Furthermore, despite $N_{\text{Core}} < \infty$, we use a continuous rather than discrete distribution, as the ensuing math is made more tractable. Both approximations introduce only small error, as will be shown in Section VI-A.

C. Energy Minimization

Although Fig. 3 included voltage placements that minimized total energy for the given number of voltages being used, we did not discuss how a particular vector of voltages is chosen. To choose a vector of voltages that minimizes energy, we use a very simple, but highly efficient iterative algorithm. We first present the algorithm and then qualitatively discuss the performance of the algorithm.

1) *Minimum-Energy Voltage Selection Algorithm:* When choosing a vector, V^*, of N voltages ($V_1 < V_2 < \ldots < V_{N-1} < V_N$), V_N is chosen to meet a core yield constraint as discussed above, so we need only choose the other $N-1$ voltages. We use the minimum-energy voltage selection (MEVS) algorithm shown in Algorithm 1—effectively a local hill-climbing approach. The algorithm begins with all N voltages spaced uniformly, and iteratively solves for the optimal V_i given that all other V_j, $j \neq i$ are equal to their previous values, until none of the V_i's change by more than ϵ (1–5 mV) from one iteration to the next. Solving for a single V_i while keeping all others constant [V_i = FindOptimal(V^*, i)] is equivalent to solving the simpler case of two voltages ($N = 2$), detailed next.

With only two voltages in the system, (8) reduces to

$$E_2 = (1+\beta)\,CN_{\text{Core}}\left(V_1^2F(V_1) + V_2^2\left[F(V_2)-F(V_1)\right]\right). \quad (9)$$

V_2 is picked *a priori* to meet the core yield constraint as discussed in the previous section, so the problem is reduced to optimally choosing V_1, which may only take values $0 \leq V_1 <$

Algorithm 1: MEVS algorithm

V^* = distribute V_i's uniformly;
Initialize V^*_{old} to 0;
while $\left(V_i - V_{i,old} > \epsilon\right) \forall i$ **do**
  $V^*_{old} = V^*$;
  foreach *Voltage* V_i **do**
    /* Solve for local optimal V_i holding all $V_{j \neq i}$ constant */
    V_i = FindOptimal(V^*, i);
  end
end

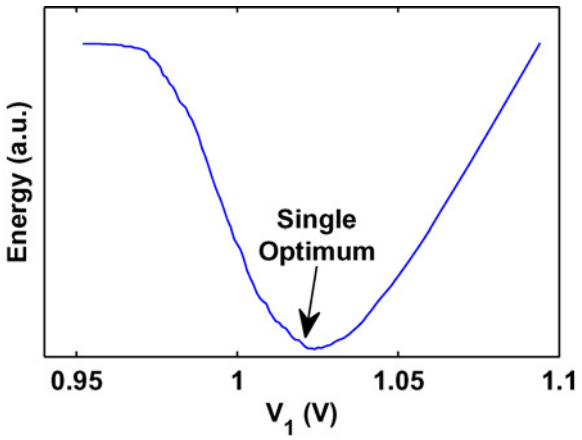

Fig. 4. Example of a single optimal set of system voltages to reduce energy per operation for $N = 2$.

V_2. There is only one optimal choice for V_1 and the proof of this is shown in Appendix A. This single optimum is also seen in Fig. 4 where the total energy is plotted versus V_1. When V_1 is smaller than $min(V_{i,\min})$ all cores must use V_2 to meet the performance constraint and hence the energy is maximum. However, as V_1 increases, more cores utilize V_1 rather than V_2 and the energy decreases. As V_1 continues increasing, the cumulative energy of the cores utilizing V_1 grows faster than the decrease in energy resulting from switching from V_2 to V_1, resulting in a minimum energy point.

Since there is no closed-form for the Normal CDF, it is written in terms of the error function, expressed as a Maclaurin series

$$F(x) = \frac{1}{2}\left[1 + erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right] \tag{10}$$

$$erf(z) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n z^{2n+1}}{n!(2n+1)} \tag{11}$$

$$F(x) = \frac{1}{2}\left[1 + \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n \left(\frac{x-\mu}{\sigma\sqrt{2}}\right)^{2n+1}}{n!(2n+1)}\right]. \tag{12}$$

We substitute (12) into (9) using a finite number of terms from the Maclaurin series (in practice, three or four terms achieves good accuracy). The derivative of the resulting polynomial is taken, resulting in another polynomial. The roots of the latter are determined using the *roots* function in MATLAB. All but one root are invalid, lying outside the range of interest or being complex.

In the multiple voltage case, (8) is used but only a single voltage is solved for at a time, and the derivative with respect to that voltage is used; all other terms are constant. While (8) is non-convex in general, we have empirically observed that there is only a single global optimum. We have also observed that though the proposed algorithm finds local optima, they are exactly (or very close to) the global optimum (see Fig. 9).

2) *Algorithm Performance:* Since we are unable to prove global optimality in the general case, we attempt to bound the energy overhead. This overhead is the difference between the "ideal" energy when each core has its own voltage (13) and the energy when we have discretized (13) using $N < N_{\text{Core}}$ voltages (14)

$$E_{\text{ideal}} = \int_{-\infty}^{\infty} V^2 f(V) dV \tag{13}$$

$$E_{\text{discrete}} = \sum_{i=1}^{N} \int_{V_{i-1}}^{V_i} V_i^2 f(V) dV. \tag{14}$$

The bounds on the overhead, and thus algorithm performance, are determined by how the N voltages are distributed. In the simple case of uniform intervals between the voltages, the overhead can be bounded by $O(\frac{\sigma}{N}\sqrt{\mu^2+\sigma^2})$ (not shown here due to space constraints). However, with more intelligent spacings, the bounds can be tightened to $O(\frac{\mu\sigma}{N})$ as shown in Appendix B. Intuitively, the spacings chosen are such that the size of each interval balances out the voltage cost [V^2 in (13)] across all intervals. One could also attempt to balance the entire energy over all of the intervals (i.e., balance $V^2 f(V)dV$). However, since the ratio $\frac{f(\mu)}{f(\mu+\sigma)}$ is constant (or the area underneath $f(V)$ remains constant despite changes in σ), this will only change the value of the constants in the bound.

As seen in Fig. 5, the actual performance of the algorithm closely fits the derived bounds. In the case of Fig. 5(a), the fit is quadratic rather than linear as was derived. Conceptually, this makes sense as we did not handle mean shifts in the derivation in Appendix B. If the mean shifts, both E_{ideal} and E_{discrete} increase in a quadratic fashion due to the V^2 term, and so the bound should also include a μ^2 term as the actual performance indicates.

IV. Analytic Energy Reduction

The power of the MEVS algorithm is in both the analytic formulation and the computational efficiency it offers. These characteristics allow for a fully analytic characterization of the energy overhead and the potential energy reduction based only on variation statistics (μ and σ). With an analytic framework in place, the "variability-induced energy overhead" can be characterized. In Fig. 6(a), the energy overhead for $+3\sigma$ core yield is plotted versus the magnitude of variation present in the $V_{\min}$ distribution ($\frac{\sigma}{\mu}$) and the number of voltages in the system (N).[2] The linear dependence on σ is again noticed: focusing on

[2]For this analysis $\mu = 1.0$ V as this is the nominal voltage found in most state-of-the-art systems.

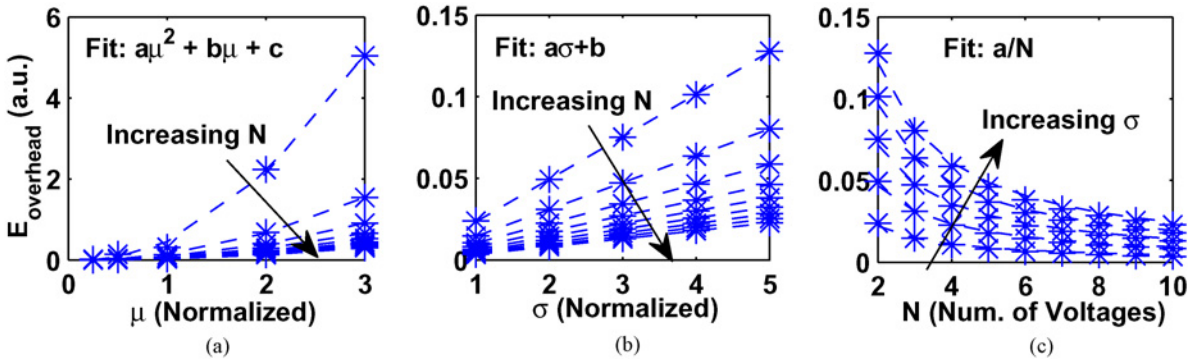

Fig. 5. Actual algorithm performance matches the derived bounds. The derivation is found in Appendix B (stars = simulation data points, dashed line = fit line). (a) Overhead versus μ. (b) Overhead versus σ. (c) Overhead versus N.

$N = 1$, it is apparent that even for a modest amount of variation ($\sim$5%) the energy overhead is significant, approaching 30%. Given 90 nm test-chip measurements from our prior work [25] and the roughly linear dependence between delay and power-supply voltage observed in those measurements, at the 90 nm node within-die variation results in $\frac{\sigma}{\mu} \approx 1\%$ and consequently a 5% energy overhead. While fairly manageable at the 90 nm node, the magnitude of variation typically increases and as the number of cores increases, the core yield constraint will increase to perhaps 4σ or 5σ, resulting in much larger energy overheads; the effects of changing the core yield constraints are discussed and quantified below.

Looking at Fig. 6(b), we notice that the amount of reduction in energy overhead is constant with the magnitude of variation. However, since the magnitude of the overhead is linearly increasing with the magnitude of variation, the energy reduction relative to the total energy will also increase linearly with variation. Furthermore, the addition of only a single new power-supply voltage ($N = 2$) provides the largest incremental energy savings no matter the magnitude of variation, with asymptotic energy reduction afterward.

V. Simulation Methodology

To test the MEVS algorithm and demonstrate the energy savings of using multiple system voltages in a real design, we use the RAW core, developed at MIT, as it is specifically developed for multicore applications [19]. However, the 64-core RAW processor is implemented in a mature 0.18 μm technology node, requiring that the core be ported to a more leading-edge process before simulations could be carried out. This involves re-synthesis in Synopsys Design Compiler with a non-optimized predictive 45 nm technology (PTM [26]) using FreePDK45, in combination with Nangate's OpenCell standard cell library [27]. As neither the FreePDK45 nor Nangate's standard cell library includes a memory compiler, SRAMs are not implemented or included in any of the subsequent simulations; however, this is consistent with all of the above analysis where SRAMs are also excluded. Although not optimal, processor register files are synthesized using available standard cell flip-flops.

Static timing analysis using Synopsys PrimeTime is then performed to choose 20 independent critical paths for detailed further analysis, as depicted in Fig. 5. The number of paths is limited to 20 as Borkar *et al.* showed that beyond 14 critical paths the frequency distribution does not materially change [28]. Furthermore, repeated paths (such as bits of busses) are eliminated, minimizing the likelihood of significant circuit-level correlation between the chosen paths. A 1K-point Monte Carlo voltage-sweep analysis is done in HSPICE for each critical path, in order to analyze the effects of within-die random variation on each path. Sweeps for a single critical path are shown in Fig. 8(a) with the associated probability distribution in Fig. 8(b), showing increased delay variability as V_{DD} decreases. Within-die systematic variation is not modeled, as our previous work in [25] indicated that random variation dominates in regular, arrayed structures.

Each of the 1K voltage-sweeps for each path is fit to the delay model in (3), and variation in the delay model parameters (K and V_T) is characterized. In particular, for each voltage sweep of each path, the appropriate values of K and V_T are determined by achieving the best fit[3] to (3). Doing this for all 1000 sweeps results in a distribution of K and V_T for each path. The statistics of each distribution as well as any cross-correlation between K and V_T (generally low) are subsequently computed and saved for further simulation. For this process, the fit values of K and V_T had between 2–4% $\frac{\sigma}{\mu}$ variation for each path.

Once this characterization is complete, the statistics of the K and V_T distributions are used to model 10K one-thousand core CMPs using (3). Analytic modeling allowed efficient simulation of many more paths, ten million in this case, than would otherwise be achievable using time-domain simulation. For each core, i, random values for K and V_T are generated from the above distributions and used to generate the delay versus V_{DD} curves for each critical path, j, $1 \leq j \leq 20$, as shown in (15). The curves for each CMP are also subjected to the same zero-mean, normally distributed mean (μ) shift, denoted by $D_0(V_{DD})$, to simulate die-to-die variation

$$D_i(V_{DD}) = D_0(V_{DD}) + \max_{\forall j} \left(\frac{K_{i,j} V_{DD}}{\left(V_{DD} - V_{T_{i,j}}\right)^{\alpha}} \right). \quad (15)$$

[3]The common minimum mean square error methodology is used for fitting non-linear data.

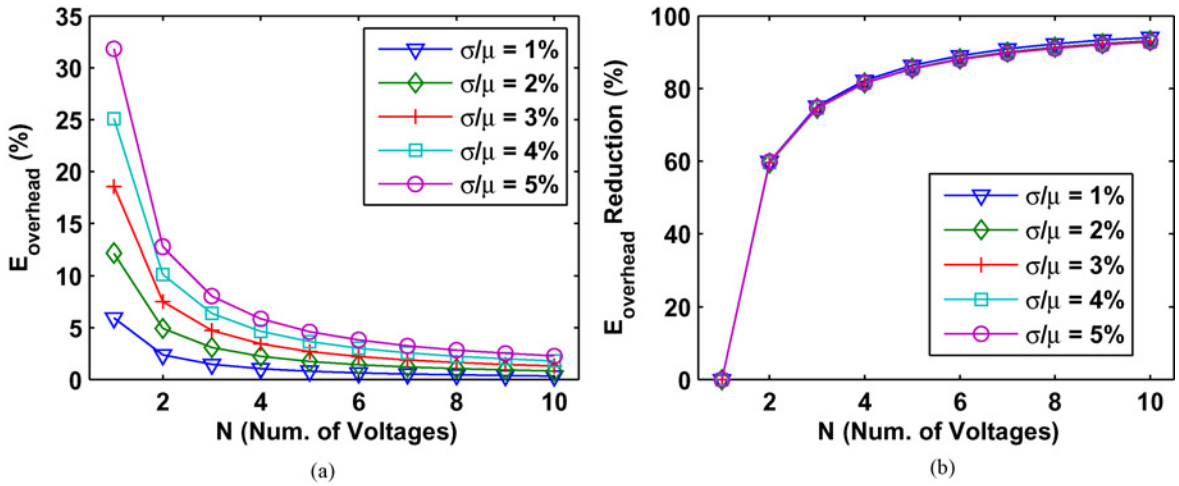

Fig. 6. Analytic computation of E_{overhead} and reduction of E_{overhead} when using additional voltages selected by the MEVS algorithm. (a) E_{overhead}. (b) Reduction in E_{overhead}.

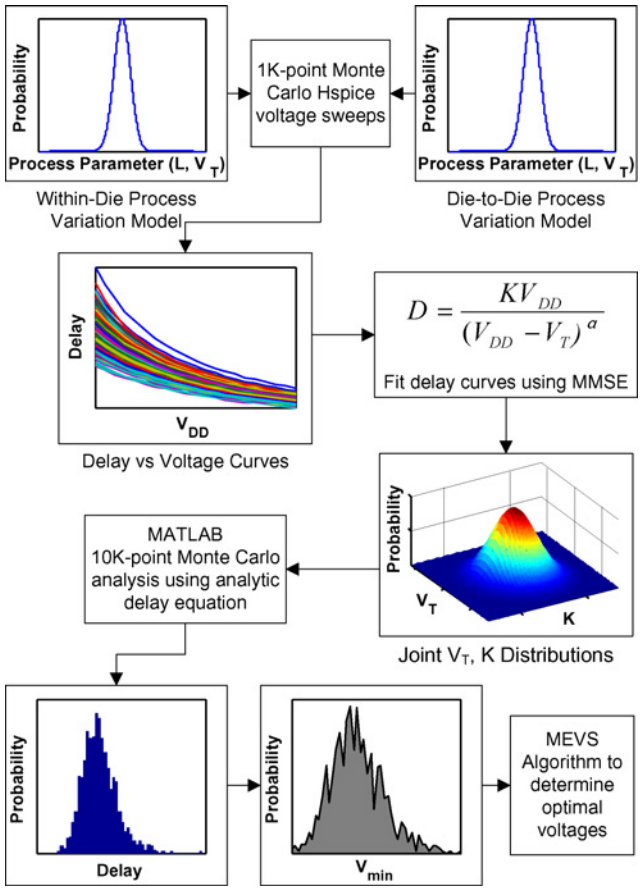

Fig. 7. Simulation methodology to efficiently evaluate MEVS algorithm and resultant energy savings.

The max of these curves for a given V_{DD} plus the die offset gives the delay versus voltage curve for core i. This is used to determine the minimum supply voltage, $V_{i,\min}$, required to meet a user-defined delay constraint, d [i.e., find $V_{i,\min}$ such that $D_i(V_{i,\min}) = d$]. The collection of $V_{i,\min} \forall i$ results in the $f(V_{\min})$ distribution for each CMP to which the MEVS algorithm is applied.

VI. Results and Analysis

The above simulation methodology, applied on 10K samples of a one-thousand core multicore processor based on the RAW core, allowed efficient evaluation of the energy savings as a result of adding voltages to the system.

A. Effect of Approximations

To understand how the approximations mentioned above affect optimal voltage selection and the resultant energy reduction, we compared a subset of the 10K voltage vectors selected by the MEVS algorithm to the actual optimal vector in each case. Since finding the actual optimal vector requires exhaustive search over a large multi-dimensional space, time and computational constraints limited this comparison to no more than six voltages.

First, Fig. 3 shows an example $V_{\min}$ distribution in a single CMP. Despite not matching any common distribution, the bulk (80%) of the distribution can be approximated with the Gaussian distribution, resulting in only small error. More importantly, Fig. 9 shows that the MEVS-selected vector is very close to the globally optimal vector despite using: 1) a Gaussian distribution as an approximation to the actual distribution; 2) the truncated Maclaurin series approximation for the Gaussian CDF; 3) continuous rather than discrete math; and 4) slightly larger ϵ as N increases to aid in convergence time.[4]

Even with six voltages in the system, the average vector distance from optimal is 10 mV, within the ripple of any power-supply voltage (typically no less than 10 mV). As the number of voltages increase, an increase in the distance from optimal is expected simply due to adding the error of each voltage. However, even in the worst-case of a 25 mV distance, this implies that the five additional voltages are on average no more than 5 mV away from their optimal values. More importantly, these small distances from the optimal

[4]Using many voltages in a system would likely not be practical, and for $N \leq 5$ keeping ϵ small ($\sim$1 mV) has no effect on convergence time.

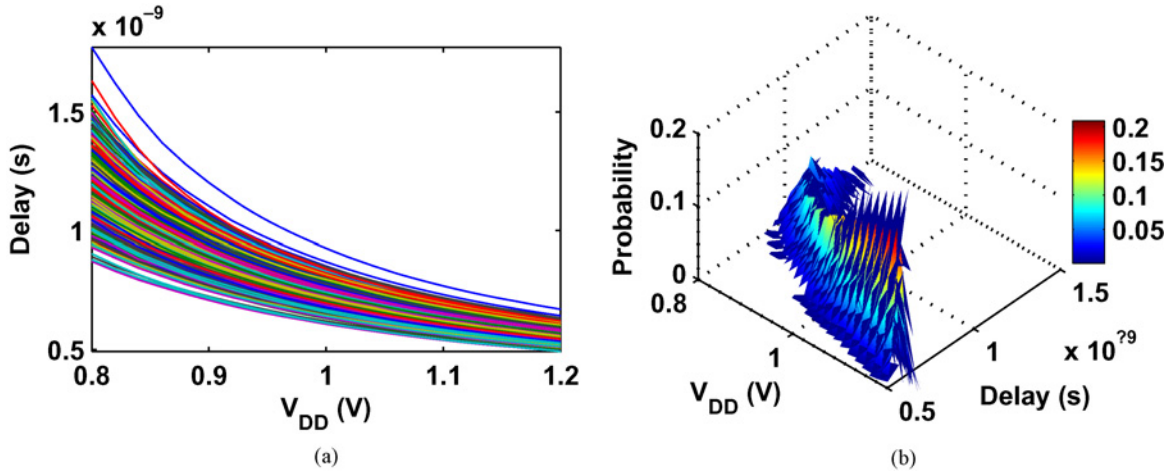

Fig. 8. 1K-point Monte-Carlo voltage sweeps for a single critical path. (a) Critical path delay versus V_{DD}. (b) Probability distribution of critical path delay versus V_{DD}.

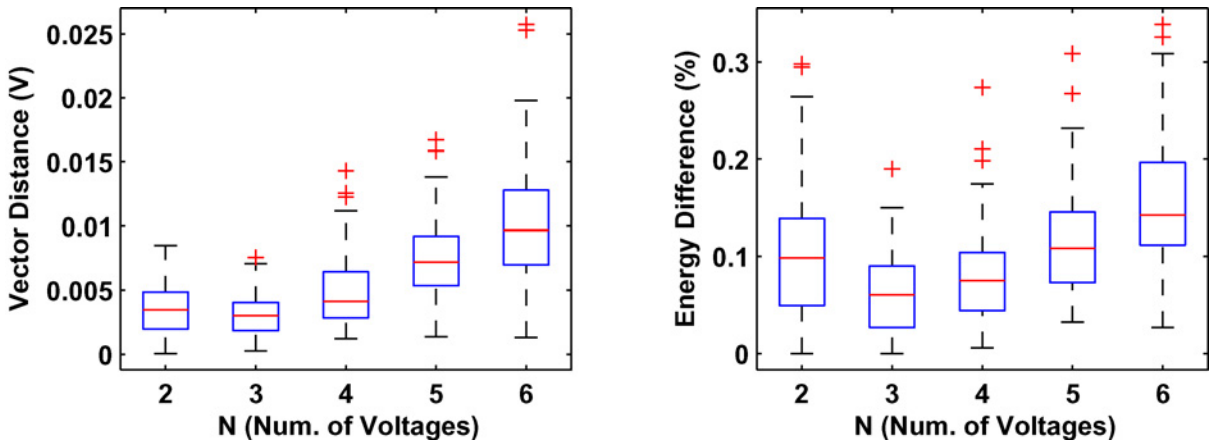

Fig. 9. Vector distance and energy difference between MEVS-selected and globally optimal vectors.

values result in energy differences between the MEVS-selected vectors and the globally optimal vector of much less than 1%.

B. Energy Reduction

With the effect of approximations shown to be small, we next analyze energy reduction in the RAW core as a result of using multiple voltages. Fig. 10(a) plots both the total energy reduction (energy difference between using a single voltage and multiple voltages), as well as the reduction in the amount of variability-induced energy overhead (energy over and above each core having its own voltage $V_{i,\min}$) for a 1000-core CMP. By adding just a single additional voltage ($N = 2$), anywhere between 59–75% of the variability-induced energy overhead is eliminated, resulting in a total energy savings of 6–16%, with an average savings of 9%. These results are in-line with the analytic results above, and are expected from observed variability in the delay and $V_{\min}$ distributions of ∼3%. The large range of energy savings noticed is due primarily to die-to-die variation, which results in mean shifts of the $V_{\min}$ distribution as opposed to greater magnitudes of within-die variation.

The addition of more voltages does increase the energy savings but at diminishing returns: with five voltages, roughly 90% of the energy overhead is eliminated, and it would take 995 additional voltages to reach E_{ideal}. Since the absolute energy savings are dependent on the magnitude of variation, as CMPs are scaled to smaller processes where variation is expected to increase, the energy overhead of using only a single voltage for all cores will also increase, as seen in Fig. 6(a). Use of our voltage selection algorithm will result in larger absolute energy savings as the relative magnitude of variation ($\frac{\sigma}{\mu}$) increases.

To understand the energy reduction on CMPs with fewer cores, we undertook an identical Monte Carlo analysis for a 100-core CMP. The results of this analysis, shown in Fig. 10(b), are that the mean energy reduction and reduction in energy overhead are reduced. Total energy reduction is approximately half that of the 1000-core CMP but energy overhead is reduced only by a few percent, indicating there is naturally a smaller variation-induced overhead in the 100-core CMP. This is expected, as a CMP with fewer cores will typically have a smaller *range* in core-to-core variation than a chip with more cores.

C. Effect of Core Yield-Constraint Choice

In the above analysis, the core yield constraint was such that the last voltage in the system, V_N, had to accom-

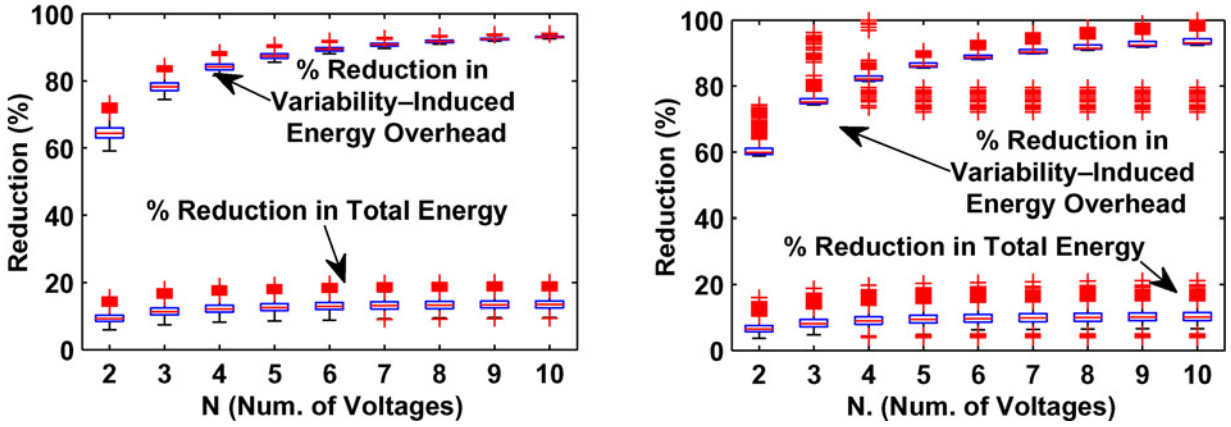

Fig. 10. Energy reduction for 1000 and 100 core CMPs as a function of number of system voltages. (a) 1000 cores. (b) 1000 cores.

modate the worst-performing core, or stated differently, all cores had to function at the required frequency. However, in such a massively parallel system, having a 100% core yield may not always be necessary, nor may it be efficient from a performance-energy perspective. To quantitatively analyze this, an appropriate metric is required. Since performance and energy are both individually important, a metric that includes both is used: for this analysis, our metric is the ratio of the total performance of the system to the total energy in the system.

The total performance in the system is proportional to the product of clock frequency (f), number of instructions completed per clock (IPC), and number of operational (or yielding) cores ($y_o \times N_{\text{Core}}$). The total energy in the system is given by (8) multiplied by the core yield, y_o, and the frequency, f. Since two system voltages provide the most incremental benefit, this analysis is only performed for $N = 2$, and so (9) is used in place of (8) resulting in

$$\frac{\text{Tot. Perf.}}{\text{Tot. Energy}} = \frac{y_o N_{\text{Core}} IPC f}{y_o E_2 f} \tag{16}$$

$$\frac{\text{Tot. Perf.}}{\text{Tot. Energy}} = \frac{IPC}{C\left(V_1^2 F(V_1) + V_2^2 \left[F(V_2) - F(V_1)\right]\right)}. \tag{17}$$

Although it would seem that (17) has no dependence on number of yielding cores, recall that there is an implicit dependence, as V_2 is selected *a priori* such that it meets the core yield constraint ($V_2 = F^{-1}(y_o)$), and so there is still a dependence on core yield. Furthermore, for the purposes of this analysis we assume that both IPC and C are constant (or do not change significantly per core), and are simply scaling factors that can be removed from the analysis.

The analysis shows that the combined performance-energy metric is roughly constant, as seen in Fig. 11(a). This is expected, as both performance and energy scale linearly with the number of operating cores ($y_o N_{\text{Core}}$). However, there is a slight decreasing trend as the energy does not strictly scale linearly with core yield, reflecting the necessary increase in both V_1 and V_2 to support additional poor-performing cores. As the core yield constraint increases to roughly 85–90%, the increase in V_2 accelerates due to the exponential nature of the tails of the distribution as seen in Fig. 11(b), resulting in faster decreases in the performance-energy metric as shown in the inset of Fig. 11(a).

Looking at the incremental change in performance relative to the incremental change in energy with increasing core yield constraint, the right axis (green plot) of Fig. 11(a) shows that the change is relatively constant until $y_o \geq 85\%$, where it sharply decreases. Intuitively, this means that for a constant increase in energy, a constant increase in performance is achieved until $y_o \geq 85\%$, at which point the incremental increase in energy required for the same incremental gain in performance becomes increasingly large. This result suggests that there may be an upper-bound on practical core yield constraints unless total computational throughput is of the essence; a similar conclusion was reached in [18] by turning off power-hungry cores.

Another way to arrive at the same conclusion is to explore the energy overhead as a function of the desired core yield. Fig. 12(a) shows an analytic computation of E_{overhead} as a function of the core yield for a fixed amount of variation in the $V_{\min}$ distribution[5] ($\frac{\sigma}{\mu} = 3\%$ in this case, according to the observed variation in the RAW core). The energy overhead increases exponentially with the desired core yield constraint, and so reducing the core yield constraint by a few percent can have a large impact on energy. Nevertheless, even if the core yield constraint is reduced to 90%, $E_{\text{overhead}} \approx 8-10\%$, which is still significant enough to warrant adding voltages to the system. Fig. 12(b) shows that a single additional voltage ($N = 2$) still provides the most incremental reduction in E_{overhead}; however, the benefit is somewhat reduced as the core yield constraint is decreased.

VII. Practical Considerations

Modern microprocessors have many power/performance modes and other design constraints that must be considered when attempting to implement a multiple voltage system. The

[5]The number of voltages is intentionally limited to $N \leq 3$ for both plots in Fig. 12, as the MEVS algorithm has difficulty assigning voltages due to the limited distance between minimum and maximum $V_{\min}$, especially as the core yield constraint is reduced. In this regime, the Maclaurin approximation is not sufficient to properly model the very narrow minimum (i.e., the derivatives change too rapidly in the vicinity of the minimum).

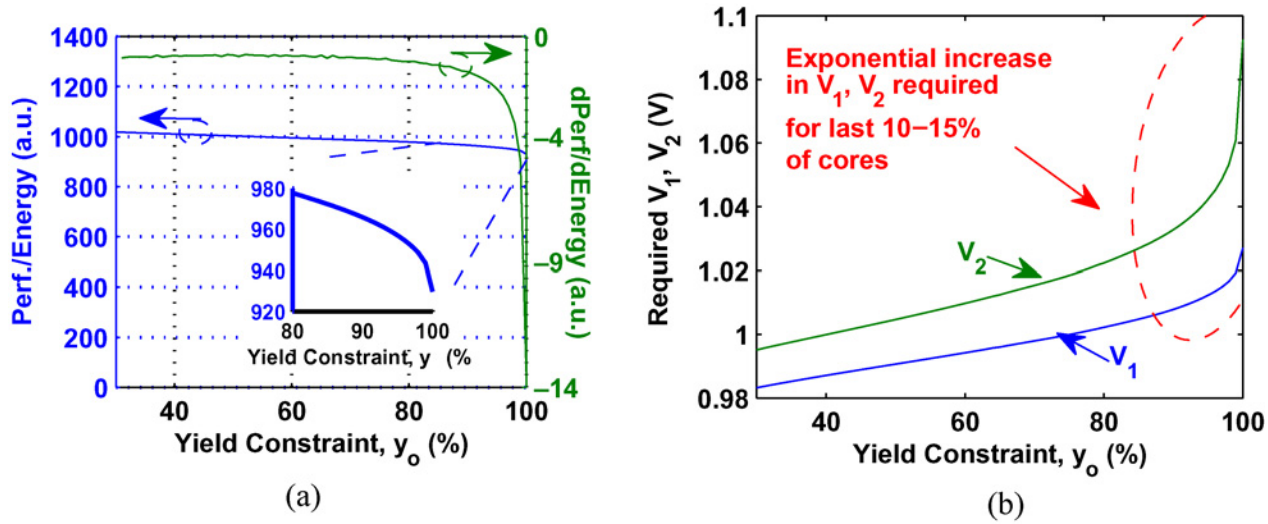

Fig. 11. Joint performance-energy metric versus core yield. (a) Performance-energy metric. (b) Required V_1, V_2.

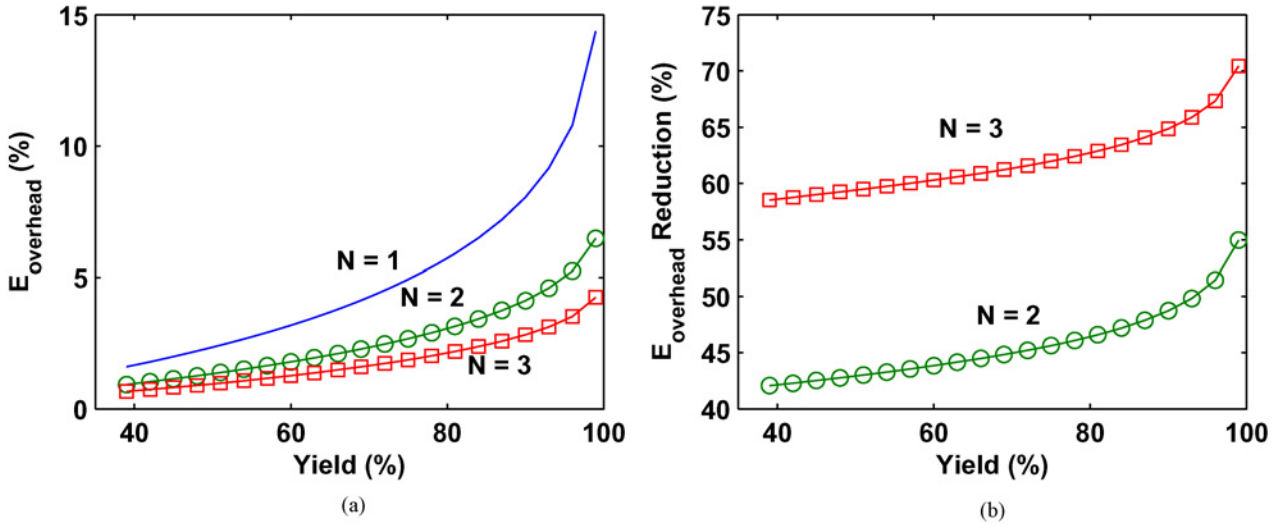

Fig. 12. Analytic computation of E_{overhead} and reduction of E_{overhead} versus core yield constraint. Reducing the yield constraint can drastically reduce E_{overhead}. (a) E_{overhead}. (b) Reduction in E_{overhead}.

following are salient aspects of physical systems, with a brief analysis of how each affects the framework and results above.

A. Partitioning of Cores

The above analysis assumes that the physical layout allows each core to select the best voltage from the supplied vector of optimal voltages. For the purposes of analytic demonstration of the greatest possible energy reduction, this is appropriate. However, such a layout would result in substantial area and routing resource overhead. A more practical layout requiring reduced overhead might group cores together, sharing a common voltage amongst the group. This will necessarily reduce the energy reduction potential as each group will be limited by the worst-case core in that group. To understand how this affects the energy reduction we have rerun the Monte-Carlo simulation for 1000-core processors where the cores have been partitioned into groups of 4, 8, 16, and 32. For simplicity, the MEVS algorithm is executed just as before but each group is now assigned a voltage from the optimal vector that is closest to, but greater than, the voltage required for the worst core in that partition.

Assigning voltages to groups of cores has the effect of decreasing the energy reduction potential, as shown in Fig. 13. With increasing partition size there is increasing variance between the best and worst cores in each partition. Consequently, assigning only a single voltage to the partition has diminishing returns as the partition size grows.

Though this method of choosing voltages for groups of cores may no longer be optimal, it nevertheless provides insight into the dependence of energy reduction on partition size. Refactoring the algorithm to explicitly include partitions—perhaps by using the MEVS algorithm for each group on its own and then globally selecting a vector of voltages—should be a subject of future work.

Another partitioning strategy is to ensure each voltage has an equal number of cores ($P = \frac{N_{\text{Core}}}{N}$). In such a strategy, the

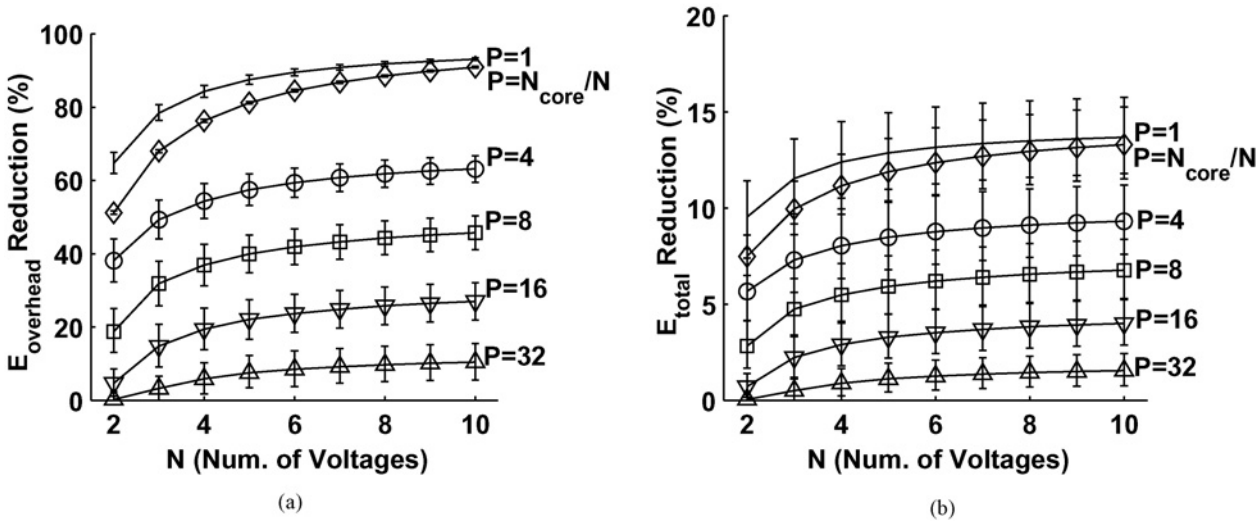

Fig. 13. Partitioning the cores into groups of 4, 8, 16 and 32 cores each results in substantial decreases in energy reduction. The worst core in each group will limit the potential energy reduction for that group of cores. Similarly, requiring that each partition contain an equal number ($P = \frac{N_{\text{Core}}}{N}$) of cores results in reduced energy savings, e.g. 10% for P = 1, N = 2 versus 7% mean E_{total} reduction for $P = \frac{N_{\text{Core}}}{N}$, N=2. Note the y-axis scale change on the right figure. (a) Reduction in E_{overhead} versus partition size. (b) Reduction in energy versus partition size.

voltage for each partition is determined by sorting all cores from minimum to maximum, dividing them into partitions of size $\frac{N_{\text{Core}}}{N}$, and setting $V_i = max(V_{i,\min})$ where the max is taken over all cores in that partition. As shown on the curve labeled $P = \frac{N_{\text{Core}}}{N}$ in Fig. 13, this strategy suffers due to limiting the number of cores assigned to each voltage. Doing so necessarily causes cores to use higher voltages than would be necessary in the optimal case, especially in the tails.

In practice, it is likely easier to have regions of the die where a subset of voltages can be selected by each core in the region. This would limit the distance current has to travel from the chip boundary (e.g. C4 bumps) to each core. Appropriately selecting the correct voltage values for each region in such a scheme should be a subject of future work.

B. Leakage and Temperature

The analytic framework above utilizes a simplified linear leakage model resulting in modifications only to the constants in the equations in Section III; in our comparison framework, these constants drop out, resulting in no impact of leakage on the results. Due to the decreasing values of β in recent commercial multi-core processors, such a model is sufficient to illustrate the energy reduction potential of multiple core voltages. However, a future analysis could include the impact of temperature as suggested below.

The power dissipation associated with adjusting voltages impacts the local temperature on the die. Leakage energy can be significantly affected as the leakage current is exponentially dependent on threshold voltage (V_T) and the thermal voltage ($V_{\text{th}} = \frac{kT}{q}$), both of which are temperature dependent.

Humenay *et al.* showed that if voltage-scaling alone is used to compensate for the impacts of variation, increases in both leakage and temperature necessitate either more expensive cooling solutions or thermal throttling, leading to dynamic performance asymmetries [17]. Compared to this worst-case of scaling a single voltage system, adding voltages significantly reduces active power as demonstrated above, but also has a more marked impact on leakage power above than when considering DIBL alone: local temperatures will also decrease due to decreased active power dissipation resulting in lower leakage power as well. To quantify this, it is necessary to incorporate the thermal impedances of the package and cooling solution into leakage models which fit into the revised energy calculation above. This can then be used to better select the optimal voltages based on temperature as well as duty cycles.

C. Impact of Memory Subsystems

Up to this point, we have implicitly ignored the impact of memory subsystems, as many techniques have been applied to reduce their active power, including putting large portions of the cache into sleep modes. For example, the Intel Dual Core Xeon processor features 18 MB of L2 and L3 caches, but only 0.08% of the caches are actively powered for a given cache access. The large caches utilized on this processor allows the cache to be organized into many smaller arrays and sub-arrays, which can individually be put into sleep states. This results in 0.75 W/MB average power for the caches, leading to the caches contributing less than 10% of the overall power budget [29].

However, in massively parallel multicore systems, each core will likely have much smaller caches, in which a larger fraction of the cache will be actively powered for a cache access. More significantly, leakage of the many un-accessed lines in the active portion of the cache will result in an increase of the fraction of total power associated with the memory subsystems. In the case that there are additional globally-shared caches, these memory systems will increase the shared energy component relative to the total energy [see (1)]. Though this must be incorporated, caches are typically operated on separate power-supplies from the computational cores and will likely not affect the optimization of the core power-supply

voltages. The shared energy may affect how many and which cores are turned off; some preliminary work evaluating this has been done in [18], in which a core is turned off if the incremental power required to operate it is greater than the power of the shared blocks, amortized over all operating cores, if that core were not turned on.

D. Power Transistors and Routing Resources

Lastly, each voltage requires global routing resources for power distribution, but this can be mitigated in the case of two voltages to a large degree by reducing the width and density of each voltage's power grid since power will be divided roughly equally over the voltages. More importantly, power-multiplexing transistors are required. These transistors are necessarily large to handle the relatively large currents necessary to power each core. However, they can also be used to power-gate the entire core, as Intel has done in their most recent Nehalem architecture to eliminate leakage power [30].

VIII. Conclusion

Future massively parallel multi-core architectures are highly susceptible to process variation in highly scaled processes. While maintaining core performance homogeneity may not be critical long-term, software and system design in the short-term to mid-term necessitate identical behavior across cores. Ensuring this using typical variation mitigation solutions is often expensive and complex across many dimensions. We proposed using a small number of additional system voltages and have presented an efficient algorithm for optimal selection of a vector of voltages to reduce variation-induced energy overhead. When two voltages are used, our algorithm is provably optimal and in the more general case we showed that it selects very close to optimal vectors. We demonstrated analytically and through simulation that a single additional voltage provides the most incremental benefit. Simulations of a hypothetical one-thousand core processor based on the RAW core showed reduction of energy overhead by 59–90% for 2–10 voltages, corresponding to 6–21% total energy reduction. Furthermore, turning off or disabling the worst performing cores on a CMP is beneficial to a simple joint performance-energy metric.

Acknowledgment

The authors acknowledge the support of the Focus Center for Circuit and System Solutions, one of five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation Program. They would also like to thank N. Verma, Y. Ramadass, and K. Balakrishnan for valuable discussions in formulating the problem as well as the Computer Architecture Group, led by Prof. A. Agarwal, at the Massachusetts Institute of Technology, Cambridge, for discussion and direction related to future multi-core processors and help in porting the RAW core to a 45 nm process.

Appendix A
Proof of Optimality for $N = 2$

Theorem 1: There exists only one solution to the $N = 2$ case and that solution must lie in $0 \leq V_1 < V_2$.

Proof: We take the derivative of (9) with respect to V_1 and set it equal to 0, giving

$$\frac{f(V_1)}{F(V_1)} = \frac{2V_1}{V_2^2 - V_1^2}. \tag{18}$$

The LHS of (18) is shown to be a positive, strictly decreasing function by Pechtl in [31]. Pechtl also showed that it is asymptotic to $-V_1$ as $V_1 \to -\infty$ and goes to 0 as $V_1 \to \infty$. On the right side of (18), the numerator is strictly increasing and the denominator is strictly decreasing over $0 \leq V_1 < V_2$, starting at V_2^2 when $V_1 = 0$ and reaching 0 at $V_1 = V_2$. Thus, the LHS is monotonically decreasing while the RHS is monotonically increasing, so there is at most one intersection point, and it must be located within the range of interest since the RHS ranges from 0 (at $V_1 = 0$) to ∞ (at $V_1 = V_2$). ■

Appendix B
Bounds on MEVS Algorithm

Let the N voltages be distributed with intervals of $\delta_k := g(k)$. We define $g(x)$ to be the "continuous" form of $g(k)$ and $G(x) = \int_0^x g(y)dy$. We also define $V_k = V_L + \sum_{j=1}^{k} \delta_j$. Over any one interval, the energy cost in the discretized case is given by

$$E_{\text{int}} = \int_{V_k}^{V_{k+1}} V_{k+1}^2 f(V)dV. \tag{19}$$

Similarly, in the ideal, continuous case it is (13) over a single interval

$$E_{\text{int}} = \int_{V_k}^{V_{k+1}} V^2 f(V)dV. \tag{20}$$

Subtracting the two and using a change of variables ($V = V_k + t, => dv = dt$), we get

$$E_{\text{overhead,int}} = \int_0^{\delta_k} \left[2V_k(\delta_k - t) + \delta_k^2 - t^2\right] f(V_k + t)dt. \tag{21}$$

However, in the limit of large N and small δ_k, $\delta_k^2 - t^2$ is small enough to be ignored (this introduces an error on the order of $O(\frac{1}{N^2})$ which is smaller than the overall overhead, see below) and $f(V_k + t) \approx f(V_k)$. So, (21) reduces to

$$E_{\text{overhead,int}} \leq \delta_k^2 V_k f(V_k). \tag{22}$$

The total overhead is then the summation of the individual interval overheads

$$E_{\text{overhead,TOT}} \leq \sum_{k=1}^{N} \delta_k^2 V_k f(V_k). \tag{23}$$

We choose $g(k)$ such that the size of each interval results in the voltage "cost," V^2, from (13) being balanced across all intervals. Choosing $g(k)$ as in (24) will achieve this

$$g(k) \propto \frac{1}{V_L + \sum_{j=1}^{k-1} g(j)} = \frac{1}{V_{k-1}}. \tag{24}$$

In the continuous case, this becomes

$$\frac{dG(x)}{dx} = \frac{C}{V_L + G(x)} \tag{25}$$

where C is a normalization constant. To be self-consistent, we also require that $\sum_{k=1}^{N} g(k) = |V_U - V_L|$ in the discrete case and $G(N) = |V_U - V_L|$ in the continuous case.

Integrating (25) we get

$$G(x) = \sqrt{V_L^2 + 2Cx} - V_L \tag{26}$$

and using the boundary conditions, C can be computed to be

$$C = \frac{1}{2N}\left(V_U^2 - V_L^2\right). \tag{27}$$

We can also make the following approximation:

$$\delta_k := g(k) \simeq G(k) - G(k-1) \simeq G'(k) \tag{28}$$

$$\simeq \frac{C}{V_L + G(k)} \tag{29}$$

which gives $C = G'(k)\,[V_L + G(k)]$.

Since $\delta_k \simeq G'(k)$, we can reformulate (23) as follows:

$$E_{\text{overhead,TOT}} = \sum_{k=1}^{N} G'(k)^2\,(V_L + G(k))\,f\,(V_L + G(k)) \tag{30}$$

and using (28) this becomes

$$E_{\text{overhead,TOT}} = C\left[\sum_{k=1}^{N} G'(k) f\,(V_L + G(k))\right] \tag{31}$$

which, in the limit of large N, is a Riemann integral

$$E_{\text{overhead,TOT}} \simeq C \int_{V_L}^{V_U} f(V)dV. \tag{32}$$

Since the integral in the above equation is simply $F(V_U) - F(V_L) < 1$, we can say

$$E_{\text{overhead,TOT}} \leq C = \frac{1}{2N}(V_U^2 - V_L^2) \tag{33}$$

$$\leq \frac{1}{2N}(V_U - V_L)(V_U + V_L). \tag{34}$$

Finally, in general $V_U - V_L \propto \sigma$ and $V_U + V_L \propto \mu$, so $E_{\text{overhead,TOT}} \leq O\left(\frac{\mu\sigma}{2N}\right)$.

REFERENCES

[1] D. Wentzlaff and A. Agarwal, "The case for a factored operating system (FOS)," Comput. Sci. Artif. Intell. Lab., Massachusetts Instit. Technol., Cambridge, Tech. Rep. MIT-CSAIL-TR-2008-060, 2008.

[2] D. Blaauw, S. Kalaiselvan, K. Lai, W.-H. Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull, "Razor I I: In situ error detection and correction for P V T and S E R tolerance," in *Proc. IEEE ISSCC*, Feb. 2008, pp. 400–622.

[3] K. Bowman, X. Tang, J. Eble, and J. Meindl, "Impact of extrinsic and intrinsic parameter variations on CMOS system on a chip performance," in *Proc. IEEE Int. ASIC/SoC Conf.*, Sep. 1999, pp. 267–271.

[4] S. Borkar, "Thousand core chips: A technology perspective," in *Proc. DAC*, 2007, pp. 746–749.

[5] S. Raje and M. Sarrafzadeh, "Variable voltage scheduling," in *Proc. ISLPED*, 1995, pp. 9–14.

[6] K. Usami and M. Horowitz, "Clustered voltage scaling technique for low-power design," in *Proc. ISLPED*, Apr. 1995, pp. 3–8.

[7] J.-M. Chang and M. Pedram, "Energy minimization using multiple supply voltages," *IEEE Trans. Very Large Scale Integr.*, vol. 5, no. 4, pp. 436–443, Dec. 1997.

[8] M. C. Johnson and K. Roy, "Optimal selection of supply voltages and level conversions during data path scheduling under resource constraints," in *Proc. ICCD*, 1996, pp. 72–77.

[9] C. Yeh, M.-C. Chang, S.-C. Chang, and W.-B. Jone, "Gate-level design exploiting dual supply voltages for power-driven applications," in *Proc. DAC*, 1999, pp. 68–71.

[10] C. Chen and M. Sarrafzadeh, "Provably good algorithm for low power consumption with dual supply voltages," in *Proc. ICCAD*, 1999, pp. 76–79.

[11] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," in *Proc. ISLPED*, Aug. 1998, pp. 197–202.

[12] G. Qu, "What is the limit of energy saving by dynamic voltage scaling," in *Proc. ICCAD*, 2001, pp. 560–563.

[13] S. Hua and G. Qu, "Approaching the maximum energy saving on embedded systems with multiple voltages," in *Proc. ICCAD*, 2003, p. 26.

[14] X. Liang, G.-Y. Wei, and D. Brooks, "ReVIVaL: A variation-tolerant architecture using voltage interpolation and variable latency," *SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 191–202, 2008.

[15] D. Marculescu and S. Garg, "Process-driven variability analysis of single and multiple voltage-frequency island latency-constrained systems," *IEEE Trans. Comput.-Aided Design*, vol. 27, no. 5, pp. 893–905, May 2008.

[16] B. Stefano, D. Bertozzi, L. Benini, and E. Macii, "Process variation tolerant pipeline design through a placement-aware multiple voltage island design style," in *Proc. DATE*, 2008, pp. 967–972.

[17] E. Humenay, D. Tarjan, and K. Skadron, "Impact of process variations on multicore performance symmetry," in *Proc. DATE*, Apr. 2007, pp. 1–6.

[18] J. Donald and M. Martonosi, "Power efficiency for variation-tolerant multicore processors," in *Proc. ISLPED*, 2006, pp. 304–309.

[19] M. Taylor, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, A. Agarwal, W. Lee, J. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, and J. Kim, "Evaluation of the RAW microprocessor: An exposed-wire-delay architecture for ILP and streams," in *Proc. ISCA*, Jun. 2004, pp. 2–13.

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

[21] S. Rusu, S. Tam, H. Muljono, D. Ayers, J. Chang, B. Cherkauer, J. Stinson, J. Benoit, R. Varada, J. Leung, R. D. Limaye, and S. Vora, "A 65-nm dual-core multithreaded Xeon processor with 16 MB L3 cache," *IEEE J. Solid-State Circuits*, vol. 42, no. 1, pp. 17–25, Jan. 2007.

[22] U. Nawathe, M. Hassan, K. Yen, A. Kumar, A. Ramachandran, and D. Greenhill, "Implementation of an 8-core, 64-thread, power-efficient SPARC server on a chip," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 6–20, Jan. 2008.

[23] G. Konstadinidis, M. Tremblay, S. Chaudhry, M. Rashid, P. Lai, Y. Otaguro, Y. Orginos, S. Parampalli, M. Steigerwald, S. Gundala, R. Pyapali, L. Rarick, I. Elkin, Y. Ge, and I. Parulkar, "Architecture and physical implementation of a third generation 65 nm, 16 core, 32 thread chip-multithreading SPARC processor," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 7–17, Jan. 2009.

[24] S. Rusu, S. Tam, H. Muljono, J. Stinson, D. Ayers, J. Chang, R. Varada, M. Ratta, S. Kottapalli, and S. Vora, "A 45 nm 8-core enterprise Xeon processor," *IEEE J. Solid-State Circuits*, vol. 45, no. 1, pp. 7–14, Jan. 2010.

[25] N. Drego, A. Chandrakasan, and D. Boning, "An all-digital, highly scalable architecture for measurement of spatial variation in digital circuits," in *Proc. IEEE ASSCC*, Nov. 2008, pp. 640–651.
[26] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," in *Proc. CICC*, 2000, pp. 201–204.
[27] J. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. Davis, P. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "FreePDK: An open-source variation-aware design kit," in *Proc. IEEE Int. Conf. Microelectron. Syst. Educ.*, 2007, pp. 173–174.
[28] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proc. DAC*, 2003, pp. 338–342.
[29] J. Chang, M. Huang, J. Shoemaker, J. Benoit, S.-L. Chen, W. Chen, S. Chiu, R. Ganesan, G. Leong, V. Lukka, S. Rusu, and D. Srivastava, "The 65-nm 16-MB shared on-die L3 cache for the dual-core Intel Xeon processor 7100 series," *IEEE J. Solid-State Circuits*, vol. 42, no. 4, pp. 846–852, Apr. 2007.
[30] S. Gunther and R. Singhal, "Next generation Intel microarchitecture (nehalem) family: Architectural insights and power management," in *Proc. Intel Developer Forum*, Aug. 2008.
[31] A. Pechtl, "A note on the derivative of the normal distribution's logarithm," *Arch. Math.*, vol. 70, no. 1, pp. 83–88, Jan. 1998.

Nigel Drego (M'03) received the B.S. degree in computer engineering from the University of California, Irvine, in 2001, and the M.S. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 2003 and 2009, respectively.

From 2003 to 2005, he was a Design Engineer with Intel Corporation, Santa Clara, CA, where he was involved in the design of a high-performance microprocessor. He is currently a Senior Research and Development Engineer with PDF Solutions, San Jose, CA, where his research involves development of new test structures for state-of-the-art process technologies.

Anantha Chandrakasan (M'95–SM'01–F'04) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, in 1989, 1990, and 1994, respectively.

Since September 1994, he has been with the Massachusetts Institute of Technology (MIT), Cambridge, where he is currently the Joseph F. and Nancy P. Keithley Professor of electrical engineering. He is the Director of MIT Microsystems Technology Laboratories, Cambridge. He is a co-author of *Low Power Digital CMOS Design* (Dordrecht, The Netherlands: Kluwer, 1995), *Digital Integrated Circuits* (Pearson Prentice-Hall, 2003, 2nd ed.), and *Sub-Threshold Design for Ultra-Low Power Systems* (Berlin, Germany: Springer, 2006). He is a co-editor of *Low Power CMOS Design* (Piscataway, NJ: IEEE Press, 1998), *Design of High-Performance Microprocessor Circuits* (Piscataway, NJ: IEEE Press, 2000), and *Leakage in Nanometer CMOS Technologies* (Berlin, Germany: Springer, 2005). His current research interests include low-power digital integrated circuit designs, wireless microsensors, ultrawideband radios, and emerging technologies.

Dr. Chandrakasan has been a co-recipient of several awards, including the 1993 IEEE Communications Society's Best Tutorial Paper Award, the IEEE Electron Devices Society's 1997 Paul Rappaport Award for the Best Paper in an EDS publication during 1997, the 1999 DAC Design Contest Award, the 2004 DAC/ISSCC Student Design Contest Award, and the ISSCC 2007 Beatrice Winner Award for Editorial Excellence. He served as a Technical Program Co-Chair for the 1997 International Symposium on Low Power Electronics and Design, VLSI Design'98, and the 1998 IEEE Workshop on Signal Processing Systems. He was the Signal Processing Sub-Committee Chair for ISSCC 1999–2001, the Program Vice-Chair for ISSCC 2002, the Program Chair for ISSCC 2003, and the Technology Directions Sub-Committee Chair for ISSCC 2004–2007. He was an Associate Editor for the IEEE JOURNAL OF SOLID-STATE CIRCUITS from 1998 to 2001. He served on the SSCS AdCom from 2000 to 2007 and was the Meetings Committee Chair from 2004 to 2007. He was the Conference Chair for ISSCC 2010.

Duane Boning (S'90–M'91–SM'00–F'05) received the B.S., M.S., and Ph.D. degrees from the Massachusetts Institute of Technology (MIT), Cambridge.

Currently, he is a Professor of electrical engineering and computer science and the Associate Head for electrical engineering with the Department of Electrical Engineering and Computer Science, MIT. From 1991 to 1993, he was a Technical Staff Member with Texas Instruments, Dallas. He served as the Associate Director for MIT Microsystems Technology Laboratories, Cambridge, from 1998 to 2004. His current research interests include variation modeling, control, and environmental issues in semiconductor and microelectromechanical systems manufacturing with emphasis on chemical–mechanical polishing and plasma etch, and computer-aided design tools for statistical process, device, and circuit design. He has written over 120 papers and conference presentations in these areas of research.

Dr. Boning is the Editor-in-Chief of the IEEE TRANSACTIONS ON SEMICONDUCTOR MANUFACTURING.

Devavrat Shah (M'04) received the B.Tech. degree from the Indian Institute of Technology Bombay, Mumbai, India, and the Ph.D. degree from Stanford University, Stanford, CA, in 1999 and 2004, respectively.

He is currently a Jamieson Career Development Associate Professor with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge. He is a member of the Laboratory for Information and Decision Systems and Operations Research Center. His current research interests include theory of large complex networks, including network algorithms, stochastic networks, network information theory, and large scale statistical inference.

Dr. Shah has received a number of best paper awards from networking, machine learning, and operations management conferences. He received the 2005 George B. Dantzig Best Dissertation Award from INFORMS and the First ACM SIGMETRICS Rising Star Award 2008 for his work on network scheduling algorithms. He was recently awarded the 2010 Erlang Prize from INFORMS given to a young researcher for outstanding contributions to applied probability. He is currently an Associate Editor of *Operations Research*.