Automating Design of Voltage Interpolation to Address Process Variations

Kevin M. Brownell, *Student Member, IEEE*, Ali Durlov Khan, *Member, IEEE*, Gu-Yeon Wei, *Member, IEEE*, and David Brooks, *Member, IEEE*

***Abstract*—Post-fabrication tuning provides a promising design approach to mitigate the performance and power overheads of process variation in advanced fabrication technologies. This paper explores design considerations and VLSI-CAD support for a recently proposed post-fabrication tuning knob called *voltage interpolation*. Successful implementation of this technique requires examination of the design tradeoffs between circuit tuning range and static power overheads within the synthesis flow of the design process, in addition to the implications of place and route. Results from the exploration of the scheme for a 64-core chip-multiprocessor machine using industrial-grade design blocks show that the scheme can be used to mitigate overhead arising from random and correlated within-die process variations. A design using voltage interpolation can match the nominal delay target with a 16% power cost, or for the same power budget, incur only a 13% delay overhead after variations.**

***Index Terms*—Post fabrication tuning, process variations, voltage interpolation.**

I. Introduction

TREMENDOUS gains in system integration and performance have been achieved by advances in CMOS device technology, as transistors have become both smaller and faster. However, in fabrication technologies beyond 65 nm, difficulties in the manufacturing process manifest as potentially large variations in the device features of fabricated transistors [1]. Worsening variations in semiconductor process technology will greatly impact the power and performance of future microprocessors and complex digital systems. Design level solutions to address process variation will be increasingly important, and to be cost-effective, such solutions must not drastically alter existing design flows.

Process variations occur at multiple spatial scales. Variations at the chip-level lead to performance differences between individual dies, but increasingly, process variations are becoming more fine-grained. Uneven mask exposure due to lithography limitations leads to correlated variations in transistor gate length and width at the granularity of units within cores. Random dopant fluctuations can change the threshold voltage of individual devices. Unlike die-to-die (D2D) variations, within-die (WID) correlated, and random variations cannot easily be solved by traditional speed-binning techniques, because a handful of slow transistors can potentially lead to slow-speed paths that affect overall processor clock frequency.

Post-fabrication tuning through body bias is a potential design solution that seeks to tune the performance of individual blocks within a chip to smooth out the impact of variation [2], [3]. However, due to the lack of body control in SOI and future triple-gate devices, new post-fabrication tuning knobs have recently been proposed. This paper explores *voltage interpolation*, one such tuning knob [4]. Voltage interpolation offers localized and fine-grained voltage tuning with two global supply voltages (VHIGH and VLOW). The post-fabrication setting of the supply voltages enables local logic blocks to operate off of different effective voltages, interpolated between VHIGH and VLOW. In turn, the delay of each block can be tuned with *effective voltages* to overcome variability effects.

Although voltage interpolation (VI) has been validated with a prototype chip implementing a simple floating-point adder [4], the technique has not been fully explored in the context of a standard VLSI-CAD design flow. This paper explores the following design issues related to VI.

- Tradeoffs in the VI synthesis flow related to the number and distribution of voltage domains impact the delay tuning range of the technique and static power overheads at domain boundaries.
- This paper explores the implications of place and route for a voltage interpolated design by evaluating three different placement strategies: delay, energy, and area overheads associated with these three strategies are examined.
- This paper investigates the concept for several logic blocks within the Sun UltraSPARC T1 core, using a standard flip-flop based design style [5].
- Implementation of VI shows the scheme's ability to compensate for random and correlated variations, by enabling better performance and power for a given design yield. The variation analysis is performed within the context of a 64-core chip-multiprocessor (CMP) scenario.

In Section II, we discuss background information and related work, in addition to the basic concept of voltage interpolation, elucidating its potential for fine grained voltage tuning. Next, a modified CAD flow for evaluating and implementing voltage interpolation is presented in Section III. This modified flow provides a framework for preforming a case study on the effective-

Manuscript received December 18, 2008; revised May 04, 2009. First published November 10, 2009; current version published February 24, 2011. This work was supported by NSF Grant CCF-0429782, NSF grant CCF-0702344, and a gift from Intel Corp. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or Intel.

K. M. Brownell, G.-Y. Wei, and D. Brooks are with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 USA (e-mail: brownell@eecs.harvard.edu; guyeon@eecs.harvard.edu; dbrooks@eecs.harvard.edu).

A. D. Khan was with Harvard University, SEAS, Cambridge, MA 02138 USA. He is now with Cadence Design Systems, Chelmsford, MA 01824 USA (e-mail: adkhan@eecs.harvard.edu).

Digital Object Identifier 10.1109/TVLSI.2009.2034457

ness of VI. Section IV describes our architecture and circuit simulation platform, which allows us to preform logic clustering and evaluate VI in the presence of process variations. Using our modified flow, Section V presents a case study of voltage interpolation, as applied to several logic blocks from the Sun UltraSPARC T1 processor, discussing the design decisions necessary for successful implementation. Using the results of the case study, Section VI explores how voltage interpolation mitigates the effects of process variations in the context of a 64-core CMP.

II. Background

Several techniques have recently been proposed to tune power/performance requirements after fabrication. These techniques often make decisions about optimizing the design based on statistical analysis of the design features, in contrast to design-time-only optimization techniques that make the best decision to improve the expected yield [6]–[10]. Post-fabrication tuning techniques can loosely be grouped into three categories: those targeting supply voltage, transistor body bias, and clock frequency.

Global voltage and frequency scaling can address variability by adaptively tuning the supply voltage for a given frequency target or by fine-grained frequency adjustment. A major drawback of the technique is that the approach must be applied at the level of chips or cores (for CMPs). Fine-grained voltage scaling with a large number of voltage domains incurs the high cost of supplying multiple voltages and the voltage regulators that generate the voltages.

In contrast, adaptive body biasing (ABB) provides a fine-grained method to control threshold voltages and, therefore, leakage and performance [3]. Recently, individual well biasing schemes with locally generated body biases have been proposed providing fine-grained control [2]. The decision of how to group the devices is made at design time, but the tuning decision occurs after fabrication during test. For example, Kulkarni, *et al.* propose a variability-aware method that clusters gates at design time into a handful of groups [11]. After fabrication, observed variations determine the body biasing per group. The results show significant improvement in reducing leakage power when compared to a fixed design-time based dual threshold voltage assignment method. Mani *et al.* recently proposed an optimization framework that combines design-time decisions (gate sizing) with tuning decisions (ABB) [12]. This framework, formulated as an adjustable optimization problem, allows the decision-maker a chance to update the optimization strategy upon learning additional information. Other researchers have considered body-bias placement based on microarchitectural blocks [13]. Unfortunately, ABB cannot be applied to technologies that lack the ability to control the body bias such as SOI and triple-gate CMOS. Thus, designers will need to explore additional tuning knobs.

Clock frequency provides another adjustable tuning knob. Tsai *et al.* propose a flow that uses statistical timing analysis to synthesize a post-silicon-tunable clock tree that can combat timing uncertainty and yield degradation [14]. Other system-level design solutions have also been considered, mainly globally asynchronous locally synchronous (GALS) [15]. This technique however can only address some of the variations that occur at coarse spatial and temporal scales.

The basic concept of utilizing two supply voltages has been considered in the past to reduce power consumption. Usami and Horowitz describe a design-time methodology called clustered voltage scale that connects logic gates to one of two supply voltages in order to reduce power while minimizing performance degradation [16]. Agarwal and Nowka suggest a voltage-selection technique that enables power reduction in a simple adder [17]. All of these works require level shifters when crossing different voltage boundaries, which introduce delay and power overheads. The voltage interpolation technique discussed in this paper obviates level shifters and avoids the related overheads. In addition to employing multiple voltages, there is a large body of work that propose design-time selection of devices with high and low threshold voltages to selectively speed up or slow down different circuit paths. Again, such design-time tuning cannot efficiently deal with increasing localized random fluctuations of device parameters set by the fabrication process and time-varying aging effects.

Previously, we explored the ability of voltage interpolation to combat process variations, only considering the technique at the synthesis stage [18]. Also, we have considered the place and route implications for two circuit blocks [19]. In this work, we consider the place and route implications of a third circuit block, in addition to preforming the variation analysis throughout the place and route stage of the CAD flow.

A. Voltage Interpolation

VI offers a fine grained, voltage tuning technique to combat the effect of process variations. A typical design must accommodate the slowest paths in the design, which may be much slower than expected due to process variations. In accommodating these slow paths, the supply voltage may need to be raised, or the frequency may need to be lowered. While this voltage or frequency shift is necessary for the proper operation of the design, a large portion of the design is now using a much higher voltage, or much lower frequency, than would otherwise be necessary. VI is a technique that enables combinational logic within single flip-flop (FF) bounded stages to operate off of a fine-grain "effective voltage, by providing two different voltage potentials on either VDD or GND.

Fig. 1(a) illustrates a basic implementation of VDD interpolation. Within a flip-flop (FF)-bounded stage, the combinational logic is split into several substages, each of which can choose between either a high voltage (VDDH) or a low voltage (VDDL). A slow logic block, resulting from process variations, can select VDDH to speed up, while a faster logic block can save power by selecting VDDL. By choosing a supply voltage appropriate to each particular block of logic, the entire pipe stage can be viewed as operating off of a single "effective voltage." In the single voltage supply case, all logic blocks must run off of the voltage required by the worst case path, whereas with voltage interpolation, varying the "effective voltage" can guarantee timing.

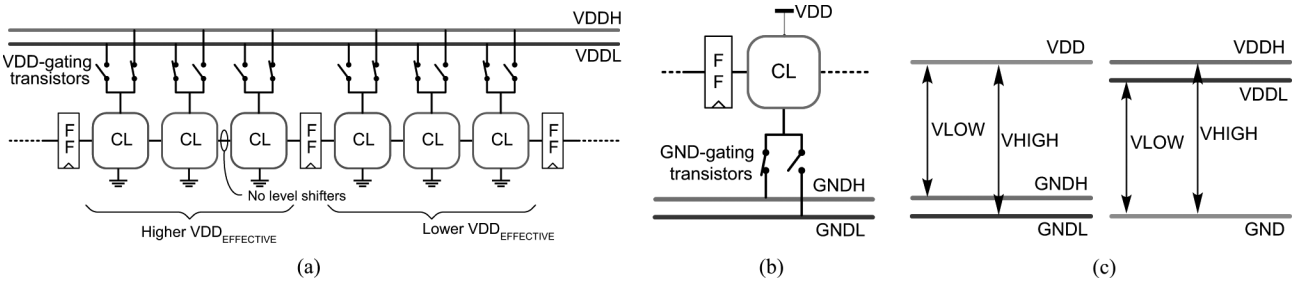


Fig. 1. Overview of voltage interpolation. (a) Block diagram of VDD interpolation. (b) Example of GND interpolation. (c) Equivalence of VDD interpolation and GND interpolation.

Note that while Fig. 1(a) implements voltage interpolation using VDDH and VDDL, the same principle can be applied by using a fixed VDD, a low ground voltage (GNDL) and a high ground voltage (GNDH), as seen in Fig. 1(b). Fig. 1(c) illustrates how VDD and GND interpolation are interchangeable, and regardless of which method of interpolation is implemented, any given substage can choose a high voltage (VHIGH) or a low voltage (VLOW). Since we assume a common bulk node, ground interpolation can introduce body effect to nMOS devices, which slightly increases tuning range.

An important design parameter for voltage interpolation is the size of the voltage difference between VHIGH and VLOW. While a larger voltage difference allows for a larger tuning range, there is a static power penalty at the boundary between a VLOW and a VHIGH stage. In the case of VDD interpolation, this is due to the weak "1" from the VLOW stage failing to completely turn off the pMOS devices of the VHIGH stage. In GND interpolation, this is due to the weak "0" failing to turn off the MOS devices. While level shifters could be used to alleviate this problem, they introduce unacceptable overheads of their own in terms of both delay and power. Section III-A elaborates upon and quantifies this static power penalty.

Another important design parameter for voltage interpolation is the number of times to cut each pipe stage. For N cuts, there are 2^N possible effective voltages achievable. An increase in the number of cuts provides an increase in the tuning resolution of a pipe stage, allowing for finer tuning of a particular block of logic. However, an increase in the number of cuts also results in an increase in the number of possible VLOW/VHIGH substage boundaries, thus exacerbating the static power problem. Additionally, a larger number of cuts suffers from an increase in the area overhead due to the power switches and routing of control signals.

A third consideration is the balance of the cuts. If stages are split in order to balance delay between substages, they may suffer severe power imbalances. Likewise, if stages are split to balance power, substages may experience delay imbalances. Moreover, perfectly-balanced delays between substages may not yield the best result.

In order to place and route an implementation of VI, the placement tool must insert power switches in close proximity to the blocks they gate. Additionally, a scannable latch must hold the current voltage setting (VHIGH or VLOW) of each VI substage, e.g., four latches for a three cut design. The outputs of these latches must be routed to the control inputs of the power switches. These control bits can be set once after fabrication and do not switch frequently, due to the mostly static nature of process variations.

Since VI uses an additional supply voltage compared to a traditional design, implementing VI may introduce some energy overhead. We assume that all supply voltages, including this extra supply, are provided by off-chip voltage regulators, similar to AMD's Griffin processor family [20]. Depending on the particular efficiency versus current load characteristic of the regulators, different combinations of VHIGH and VLOW settings will result in different amounts of energy overhead. We do not directly explore the energy overhead resulting from different current loads on different voltage supplies. The goal of this paper is to explore tradeoffs in these design considerations for a set of highly-optimized design blocks. These tradeoffs will depend on the system architecture and amount and type of variation experienced by the system. The paper addresses the VLSI-CAD support that is necessary to exploit voltage interpolation within a standard synthesis flow, taking into consideration the implications of place and route.

B. Power Switching

Since VI requires two power switches for each substage, VI incurs an area overhead, in addition to a nonzero impedance between the global supply net and the local supply nodes. Larger power switches can minimize this impedance, but incur higher area overhead. The requirements and design decisions of these power switches are similar to those of power gating.

Power gating is a technique that has been extensively applied to substantially decrease the idle power consumption of inactive CMOS circuits [21]. Different circuit size granularities have been explored for power gating, although primarily coarse-grain techniques have been applied in industry. Many works have examined issues surrounding power gate sizing, and implemented schemes to accurately assess how much area is required for a given circuit block [22]–[24], by taking into account input patterns and timing criticality of the cells being power gated. In the context of VI, we explore the tradeoffs between circuit size granularities for power gating.

We quantify this overhead via circuit simulations of the setup presented in Fig. 2. The framework includes multiple copies of a block of combinational logic. Each copy consists of a number of gates configured as a delay chain, and fed by a pseudo random bit sequence (PRBS). Each circuit block's PRBS is initialized with a different seed, and configured to conservatively have a

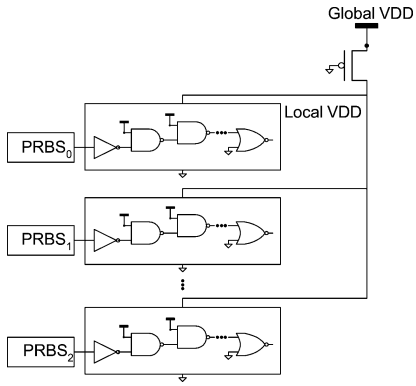


Fig. 2. Simulation setup to assess VDD-gating overheads.

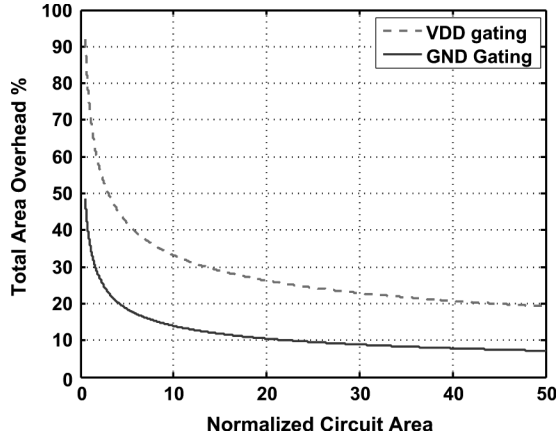


Fig. 3. Relationship between circuit size and area of power switches for a worst case VDD or GND droop of 5%.

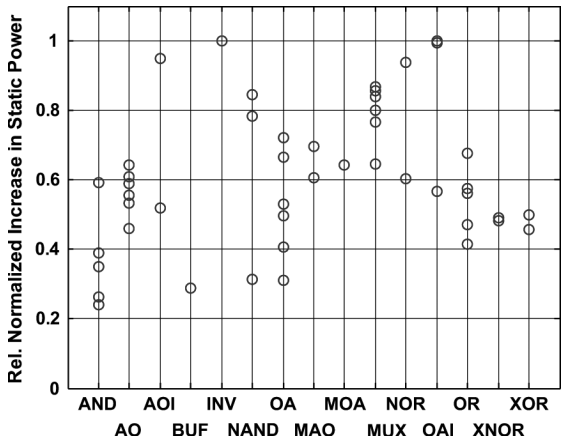


Fig. 4. Relative increase in static power, normalized to an inverter, for 3× drive strength foundry-provided standard cells in UMC 130-nm CMOS process.

20% transition density, in order to model the typically less than 20% activity rate of typical microprocessors [25]. Each block consists of the most common gates seen along critical paths of the UltraSPARC T1 floating point adder, in order to model real-life circuits, synthesized using a 130-nm UMC standard cell library. A high threshold voltage power switching device connects the global supply to the shared local supply node shared by all of the blocks. This power gating device, although depicted by a pMOS connecting to VDD in Fig. 2, can also be realized with an nMOS device connecting the local GND nodes to the global GND. The simulations vary the number of circuit blocks from 10 to 100, and the area of the power switching device from 1% to 50% of the circuit area. Each simulation runs for 5000 cycles, while recording the worst-case voltage droops. The resulting voltage droops combine to construct Fig. 3, which shows the area overhead required by circuit blocks of a certain size for a maximum voltage droop of 5%. For a given power switch size, larger current draw leads to larger average voltage droop in the local supply nodes, which degrades margins. Hence, power switches must be sized sufficiently large to minimize this droop. Interestingly, simulation results show that the relative area overhead of the power switches increases as circuit area decreases. This is because, in addition to average droop, the power switches can exacerbate local voltage noise. While large blocks can benefit from current averaging across the large number of switching logic gates, smaller blocks are more susceptible to worst-case switching scenarios. Our simulations also show that increasing power switch size (and reducing impedance) is more effective than adding bypass capacitors to the local supply nodes, for equal area overhead.

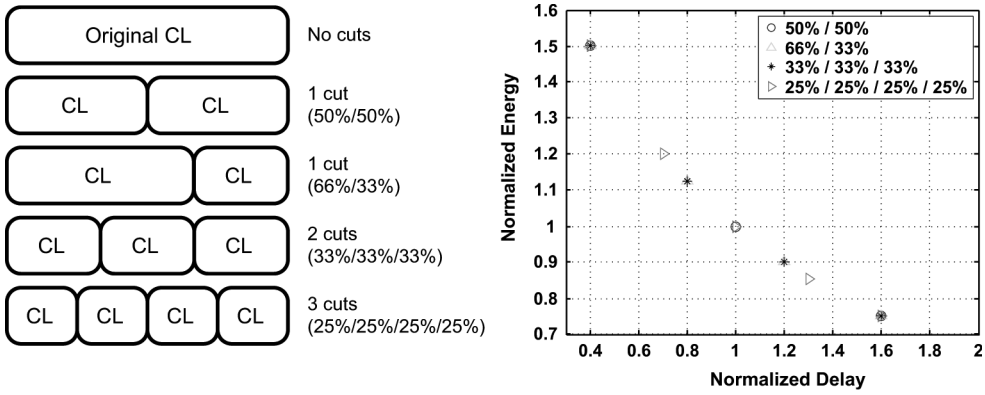
As expected, the higher mobility of GND switches results in lower resistance for a given device area and introduce lower area overhead with GND interpolation. However, there may be some concern that GND interpolation could result in higher static power leakage at VLOW to VHIGH substage boundaries than VDD switches for a given ΔV. Our simulations show that this increase in static power is negligible and the significantly lower area overhead of using GND switches outweighs this penalty. Hence, we choose to use GND interpolation for the remainder of this paper.

To implement these GND switches required by voltage interpolation, we propose a strategy of populating empty space in the layout with unit-sized GND switching cells. This empty space is a consequence of using a placement density of less than 100%, which is necessary to allow proper routing of the design. Ordinarily, MOS capacitor filler cells occupy this empty space.

This approach necessitates a small modification to the place and route flow. The global supply nets are distributed in a manner typical of an ordinary design. However, the tool must connect the local supply nets of each substage to the global supply nets through the power switching devices. The power switching cell consists of a pair of GND-gating devices of predetermined width, which also implies a minimum quantized area cost even for the smallest circuit block sizes that may require smaller GND-gating transistors. This minimum cost further motivates us to closely consider the size of the blocks being gated, and the number of power gating cells required. Once all of the overheads have been calculated, the final comparison of different strategies only considers solutions that allows all of the power switches to fit within empty spaces.

III. Modified CAD Flow

Implementing VI requires a few insertions and alterations to a standard CAD flow. First, logic synthesis is preformed on the Verilog or VHDL description of the design. Next, a substage



(a) (b)

Fig. 5. Investigation of different cutting strategies for an ideal block of logic. (a) Four combinational logic cutting scenarios. (b) Normalized energy versus normalized delay.

clustering algorithm produces an annotated netlist. Then, a substage-aware algorithm preforms placement, to group substages in the layout. Finally, power switching cells are inserted into the layout, and the design is routed.

In order to successfully investigate VI's ability to mitigate the effects of process variations, a number of design decisions must be made to achieve the best outcome.

A. Substage Clustering

First, we consider tradeoffs in the substage clustering process, starting with the static power penalty at stage boundaries. After synthesizing a netlist, a substage clustering algorithm must cut the combinational logic. The following two sections address two vital issues involved with substage clustering—the static power penalty at the boundary of VLOW and VHIGH substages, and the number and delay balance of substages.

1) Static Power: One of the primary concerns associated with VI is the potential for static power at the interface between a VLOW stage and a VHIGH stage, as previously discussed. In order to take this static power increase into account, exhaustive HSPICE simulations of every possible input combination, for each gate in the standard cell library, were preformed. Assuming each input combination is equally likely, simulations obtain an average static power for each cell. For these simulations, we assume a worst-case ΔV of 300 mV. Lower ΔV's lead to lower static power.

Fig. 4 presents the relative increase in static power at a VLOW/VHIGH boundary for a representative collection of 3$\times$ drive strength standard cells, assuming all input combinations to a gate are equally likely, normalized to the relative increase observed for an inverter. Each type of standard cell has one or more points, depending on the different numbers of inputs. For example, the library has 2-, 3-, and 4-input NAND gates, corresponding to three points in the plot. The set of 3$\times$ drive strength cells was chosen to be representative because most of the gate types in our library had a 3$\times$ cell. The relative increase in static power does not change substantially when considering other drive strengths. The inverter clearly suffers the highest relative static power increase, with most other cells suffering static power increase at much lower levels. An inverter at a VLOW/VHIGH boundary has two possible inputs for VDD interpolation- a weak "1" and a normal "0," thus experiencing a worst case static current draw 50% of the time. Complex gates have multiple transistors between VDD and GND, so require most or all of their inputs to hold weak levels before experiencing their worst case current draw, making the average static power increase less likely. These results suggest that the substage clustering algorithm should avoid inverters and OAI for the first gate at the boundaries between blocks operating off of different voltages. Our later analysis of VI carefully takes into account the impact of this static power penalty between VLOW and VHIGH stages.

2) Substage Balance: Voltage interpolation relies on the splitting of a block of combinational logic into multiple subdivisions in order to offer fine-grained voltage tuning. There are multiple ways one can implement the cuts—depending on the number of cuts and division of delay between cuts. Fig. 5(a) illustrates different cutting possibilities assuming an ideal block of combinational logic where all delay paths are perfectly balanced and consume the same amount of energy. The figure considers four scenarios: one cut, delays split 50%/50%; one cut, split 66%/33%; two cuts, split 33%/33%/33%; and three cuts, split 25%/25%/25%/25%. Three out of the four scenarios implement the cuts with delays evenly distributed across each subdivision. One scenario (1 cut, 66%/33%) considers the case where the delay through the first group of logic is twice as long as the second group. Fig. 5(b) plots the normalized energy versus normalized delay scatter plot as a result of implementing VI. The two extreme ends correspond to cases where all of the logic groups, regardless of the cutting strategy, operate off of VHIGH or VLOW. The one cut scenario with delays split 50%/50% yields three distinct delay versus energy points while the 66%/33% delay split scenario yields four distinct points. The delay imbalance offers a richer set of possible delays since VHIGH/VLOW and VLOW/VHIGH configurations lead to different delays. For larger numbers of cuts, Fig. 5(b) shows that the number of energy/delay points does not increase substantially, because many of the configurations overlap if the delays and energy consumption of each stage are perfectly balanced. This leads one to conclude that for an ideal block of logic, a more intelligent cutting strategy may offer a finer

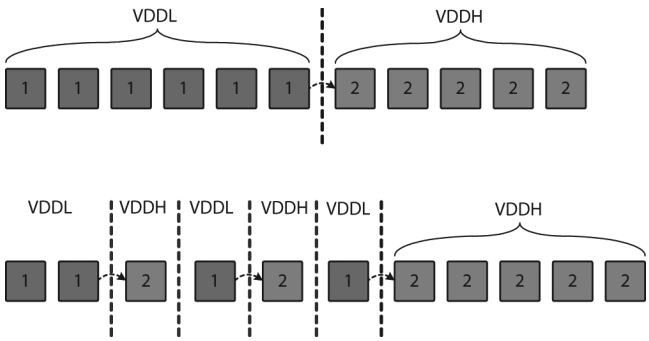


Fig. 6. Static power overhead resulting from cell reassignment.

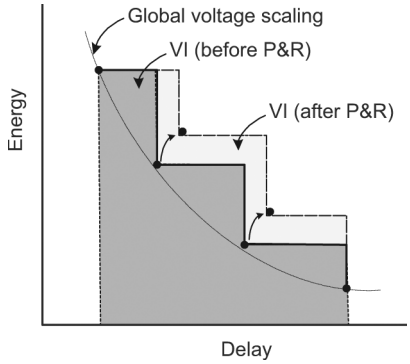


Fig. 7. Calculation of energy overhead.

tuning resolution, and that moving to a larger number of cuts can yield a large number of effective voltage settings.

B. Place and Route Overheads

In addition to the area overheard of the power switching cells discussed in Section II-B, the place and route flow for a VI design may introduce two additional sources of overhead—delay and energy.

1) Delay Overhead: Any constraints or limitations imposed upon the place and route flow may create the potential for an increase in critical path delays compared to a layout which lacks constraints. This delay overhead results from the decrease in flexibility that the placement tool has to find optimal locations for each cell and related groups of cells, leading to increases in wire length and routing congestion. We can quantify this delay overhead by comparing the worst-case delays reported after place and route.

2) Energy Overhead: If during place and route the cells of a particular substage cannot all be placed together, the static power loss may be increased by unintentionally introducing additional VLOW to VHIGH boundaries. As depicted in Fig. 6, all of the cells of each substage are contiguous and there in a single boundary, before placement (top). However, depending on the place and route strategy, the tool may place some cells separate from their original grouping (group 1) and reassign them to the local supply node of a different group (group 2), after placement (bottom). We can evaluate this potential energy overhead by analyzing the energy versus delay curve associated with VI. For any given VI design, there are a number of different possible tuning points, depending on the number of VI substage cuts. Fig. 7 illustrates the stepwise relationship between energy and delay due to quantized tuning points. If more substages connect to VHIGH, delay is lower at the expense of higher energy. The smooth curve represents the energy versus delay relationship if local voltage scaling were possible. Cell reassignment can cause a shift in the stair steps and increase energy for the same delay. Throughout the remainder of this paper, we represent energy overhead as the average increase in energy across the delay tuning range, calculated via detailed power analysis.

C. Substage Aware Placement Strategies

In order to realize an implementation of a design instrumented with VI, each substage needs to have its own local supply net. This motivates the need for regular well-defined

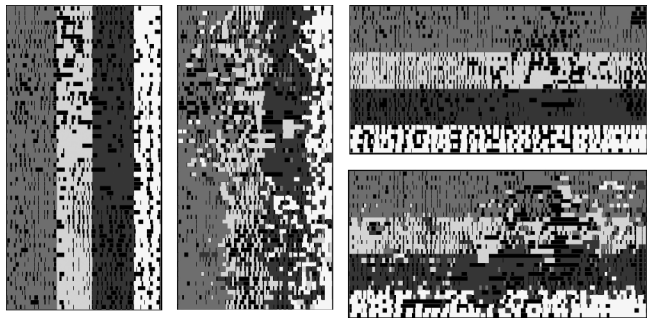


Fig. 8. Examples of forced placements with strict (left and right-upper) and relaxed (middle and right-lower) boundaries for two aspect ratios ($\mathrm{h}/\mathrm{w} = 2$ and $\mathrm{h}/\mathrm{w} = 0.5$).

stage layout regions. We explore three layout placement strategies to achieve regular layout regions: forced placement, cluster boxing, and a hybridization of the two. After these well-defined regions are created, power MUX cells can be inserted into empty space in the layout, and the design can be routed.

1) Forced-Placement Strategy: Forced placement specifies well-defined regions assigned to different substages prior to placement. Then, the placement tool places cells into regions that match their substage number. These regions can either be relaxed or strict. A strict region only allows cells assigned to it to be placed within the region. A relaxed region allows some flexibility at the boundaries, such that some cells that are not originally assigned to it may still be placed within the region. Fig. 8 shows forced placement applied in four ways, using two different aspect ratios, each of which uses strict or relaxed regions. The advantage of the forced-placement strategy is that it maintains substage assignments originally made for the cells (during synthesis) and it avoids energy overheads. Unfortunately, this method is susceptible to delay overheads when compared to an unconstrained place-and-route of the netlist due to routing congestion. Relaxed forced placement can alleviate routing congestion, but does not produce a layout which can easily connect local substage supply nets, due to a significant number of non-contiguous substage regions in the layout.

2) Cluster-Boxing Strategy: Cluster boxing is a placement strategy that starts with a baseline, unconstrained placement of cells and overlays a predetermined grid on the layout, where each rectangle within the grid is identical in size. Then, each grid rectangle is reassigned to be a substage with respect to the

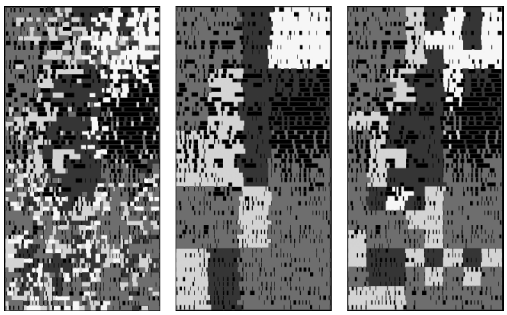


Fig. 9. Examples of cluster boxing with unconstrained (left), 5×5 (middle), and 8×15 (right) grids.

substage that the majority of the cells belong to. Fig. 9 shows two example results of cluster boxing applied to the original placement (left) with two different grid sizes. The advantage of this method is that it does not incur a delay overhead as it retains the original unconstrained placement. On the other hand, arbitrarily changing the substage assignments of cells may increase the number of substage boundary crossings along any given path, introducing energy overheads.

3) Hybrid Strategy: The forced-placement and cluster-boxing strategies represent two ends of a delay-energy tradeoff continuum. The former enjoys no energy overhead, but potentially high delay overhead, while the latter incurs a high energy overhead, but no delay overhead. This motivates the consideration of a hybrid strategy that starts with a relaxed forced placement and applies cluster boxing to it. Since relaxed forced placement offers looser constraints there is a smaller impact on the critical path; additionally, there is a smaller power penalty since fewer cells are reassigned during the cluster boxing phase.

IV. Simulation Framework

Having defined a modified CAD flow and framework for analyzing VI, we now discuss our simulation framework for characterizing our standard cell library, applying random and correlated variations, and preforming substage clustering, to enable our case study of blocks from the UltraSPARC T1 processor in Section V, and our analysis of the ability of VI to combat process variations in Section VI.

In order to quantify the impact of random process variations, exhaustive HSPICE Monte Carlo simulations were preformed on each cell in the Faraday standard cell library for the UMC 130-nm process [26]. Effective gate length, oxide thickness, and threshold voltage (Vt) were varied. Sigma over mean numbers were chosen to be representative of modern 65-nm technologies, and were chosen to be 3%, 3%, and 8%, respectively.

The decision to perform our analysis using a commercial 130-nm technology is motivated by our eventual plan to fabricate a test chip using VI and validate this simulation-based study. However, since more aggressive technologies will exhibit greater amounts of process variations, we should evaluate the effectiveness of VI for modern chips and, therefore, choose variations appropriate for 65-nm technologies. Additional simulations show that for a given tuning range, the static power seen using 65-nm technology models is actually less than that for the 130-nm technology node. This is because the slight decrease in the threshold voltage from 130 to 65 nm allows for a lower difference between VHIGH and VLOW to achieve the same tuning range. This lower voltage difference in turn reduces the static power penalty. Hence, post-fabrication tuning with VI ought to readily scale to more advanced technology nodes.

We characterized our standard cell library using the simulation setup. All inputs are shaped to resemble transitions of fanout-out-of-4 (FO4) inverters and all outputs see FO4 loading. Two hundred and fifty Monte Carlo simulations were performed for each set of inputs and input transitions that caused a transition on the output of each cell, across a range of supply voltages (0.9, 1.05, and 1.2 V). Assuming each input combination is equally likely, the propagation delays resulting from each input combination were averaged. Then, the 250 different average propagation delays resulting from different values for gate length, oxide thickness, and Vt were processed to obtain a standard deviation (σ) and mean (μ) values for the delay of each cell, representing the spread of delays for each gate caused by random process variations. Average energy numbers were also obtained for each cell.

We use a multi-level quad tree to model the effects of correlated within-die variations [27]. The standard deviation of the total delay variation seen across four levels of the quad tree was set roughly to be 8.33% and equally distributed across the levels. This value was chosen such that the maximum delay shift possible due to correlated variations is equal to the maximum possible delay shift of an inverter resulting from our modeled random variations. This approach is consistent with the approach from [4].

The VI concept relies on cutting combinational logic blocks into several pieces. Our approach utilizes the Synopsys Design Compiler [28] pipelining package, which is invoked by the *pipeline_design* command. This surrogate cutting algorithm enables us to rapidly evaluate different cut strategies and better understand different constraints and limitations for an eventual VI cutting algorithm. The dc pipeline package is used to pipeline a block of combinational logic, with the goal of minimizing clock speed for a given number of pipe stages and, thus, balancing the delay of each stage. Since actual repipelining of the designs was not our goal, we set the tool to ignore any delays due to flip flop clock-to-Q and setup time when pipelining. For example, the command *pipeline_design -no_clock_correction -stages 4* can be used on a synthesized design loaded into Design Compiler. The resulting netlist would be pipelined into four stages, with the delay though the combinational portion of each stage roughly balanced. In order to create a design with substages of different delays, *pipeline_design* is invoked with more substages than necessary. Then, adjacent substages are merged into one.

After running through the pipelining tool to obtain cut points, flip flops added by the tool are removed. By instructing the tool to avoid adding or removing any cells, the resulting design, with delay-balanced substages, exactly matches the original design. For the netlists we investigated, *pipeline_design* was only able to split the logic into four or fewer stages. Attempting to create five stages often resulted in the fifth stage being almost entirely empty. As a result, we only investigate up to three cuts in our analysis.

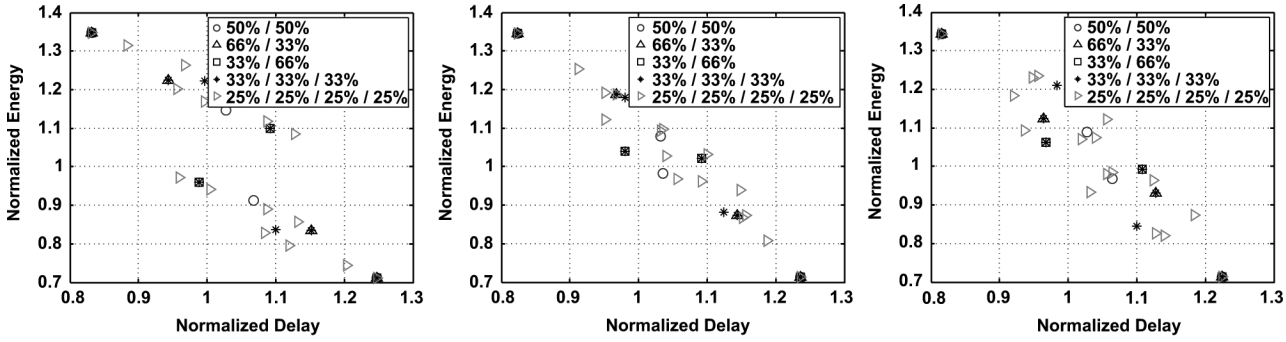


Fig. 10. Impact of different cutting strategies for three blocks of combinational logic from UltraSPARC T1: (a) ALU; (b) FADD2; (c) FADD3.

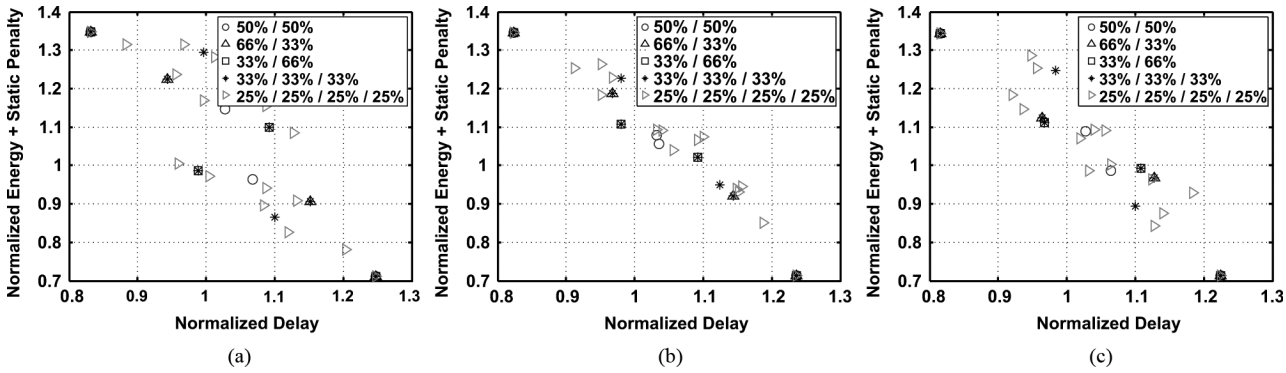


Fig. 11. Impact of static power penalties added to the energy vs. delay points in Fig. 10: (a) ALU; (b) FADD2; (c) FADD3.

V. Case Study: UltraSPARC T1

Based on the simulation framework described in Section IV, and following the modified CAD flow detailed in Section III, this section investigates the potential drawbacks and benefits of applying VI to the arithmetic logic unite (ALU) and two floating point adder stages from the UltraSPARC T1. We first consider various strategies to divide up or cut a block of combinational logic for voltage interpolation. After verifying that three cuts offer the best tradeoff in terms of overall energy efficiency despite higher static power costs, we evaluate the overheads of place and route for our three different strategies, and conclude that the hybrid and forced placement strategies offer the lowest energy delay squared product (ED^2).

A. Substage Clustering

In contrast to the ideal block of logic considered in Section III-A, real implementations exhibit a wide range of delay path imbalances. Fig. 10 presents normalized energy versus normalized delay scatter plots for three blocks of logic found in the UltraSPARC T1 RTL code. These three blocks were synthesized aggressively for the same frequency target and the different cut scenarios were obtained using the approach described in Section IV. Delay and energy for each gate is modeled using the mean values calculated in our HSPICE simulations described in Section IV. Despite implementing cut strategies that strive to meet balanced delay targets, a wider variety of effective voltages are available due to inherent imbalances.

Since there is a discrete number of cells along each path, each with a different delay, it is impossible to split the paths into perfectly equal stages. This can be advantageous for voltage interpolation, as it can increase the number of effective voltages available. However, not every configuration is usable since some configurations exhibit higher energy and higher delay compared to others. The ALU, as depicted in Fig. 10(a), exhibits two sets of energy versus delay trends. Although the ALU is balanced relatively well in terms of delay, it is not balanced very well for power. The first subdivision of logic in the ALU consumes a disproportionate fraction of power, nearly 60% of the total, thus causing the observed shift in overall energy consumption whenever it switches between VLOW and VHIGH. Hence, only a subset of the configurations that offer the best energy-delay tradeoff ought to be used. This power imbalance introduces a large energy cost to meet normalized delay targets below 0.95 in the ALU. The other two blocks, FADD2 and FADD3, do not suffer as much from power imbalances and exhibit smoother trends.

Fig. 11 presents the same set of relationships as Fig. 10, but includes the static power penalty when a VLOW stage precedes a VHIGH stage, as discussed in Section III-A. While some points have shifted up slightly, the overall change in the achievable energy/delay points is relatively small. While Figs. 10 and 11 show that more cuts yield a richer set of possible voltage settings, it is not clear which cut strategy would be best. While more cuts offer more voltage settings, static power penalties are higher.

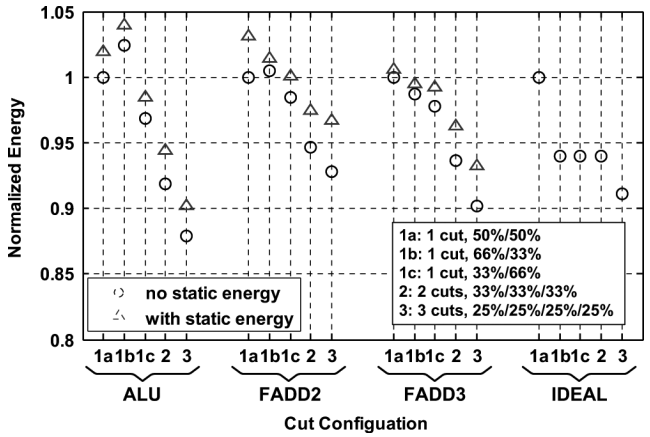


Fig. 12. Normalized energy versus cut configurations with and without static energy penalties.

Fig. 12 quantifies the relationship between the different cutting strategies, before and after the static power penalty is considered. The y-axis represents the average energy of a circuit whose target delay is equally distributed between the all VHIGH and all VLOW settings, normalized to the energy of the one cut, equally split case for each circuit. This average energy is calculated by taking the integral of the usable configurations from Figs. 10 and 11.

Each of the three logic blocks are shown, in addition to the case for an ideal circuit that can be perfectly cut for delay and energy. Fig. 12 shows that as one moves to increasing numbers of balanced cuts, the average energy falls, even when static power is taken into account. In the case of the ALU, three cuts achieves $\sim 12\%$ less average energy than one cut both when considering and ignoring static energy, despite the per-stage power imbalance noted earlier. The FADD2, FADD3, and ideal blocks use 7%, 10%, and 9% less energy respectively by moving from one to three cuts. The unbalanced one cut strategies are sometimes better, and sometimes worse than the balanced one cut strategies, depending on the circuit. Since the balanced three cut strategy provides a lower average energy for each block than all of the other options, it is the cut strategy of choice, before considering the overheads of place and route.

B. Substage Aware Placement Analysis

We explore three layout placement strategies to achieve regular layout regions using the built-in capabilities of Cadence SoC Encounter [29]: forced placement, cluster boxing, and a hybridization of the two, as described in Section III-C.

To start, we use Encounter to perform unconstrained placement on the ALU, FADD2, and FADD3 blocks. This provides the baseline critical paths to which to compare the results of different strategies. These unforced placements also serve as a basis for cell reassignment in the cluster boxing strategy.

We assume a placement density of 70% and allow four layers of metal for signal routing. High effort timing driven placement is employed in addition to high effort timing driven routing. We choose a core density of 70% to allow fair comparisons among the layouts resulting from the various placement strategies. If a higher placement density is used, a large number of the design points fail to route and hamper our evaluation of the relative effectiveness of the different strategies. We assume that the empty space that would otherwise go to filler cells can be used to hold unit power MUX cells. Since we chose 70% core density, this leaves 30% of the area for power switches.

In order to evaluate the different placement strategies, we consider the tradeoffs involved in balancing among the following variables: original stage assignments for the cells, which corresponds to tunability of the system; overall critical path; power implications, particularly at the stage boundaries; area of power control MUXes; and the simplicity of the place-and-route schemes. The number of VI substages for each block can vary as function of the number of cuts implemented, where one cut results in two substages, and we consider up to three cuts. While more cuts typically improve circuit tunability (finer-grain effective voltages), they also increase static power overhead. Although Section V-A concluded that three cut voltage interpolation was optimal, this may not hold true when considering the additional overheads of place and route.

1) Forced Placement Results: There are two types of overhead associated with the forced-placement strategy, delay and area. The energy for each layout is unchanged from the post-synthesis average energy. The delay overhead corresponds to the negative slack that results from place and route compared to the original delay target during synthesis. Fig. 13 presents the delay overhead for a variety of aspect ratios, cut directions, and number of cuts, for ALU, FADD2, and FADD3 with strict and relaxed forced placement. The baseline results correspond to unforced placement with no cuts. While there should be a total of 10 points per column (except baseline), not all configurations routed successfully. The plot shows that in general, relaxed regions result in less delay overhead than strict regions, as expected. Although a large number of designs imposing strict regions failed to route (and omitted), there are a significant number of design points that suffer less than 10% delay overhead. It is important to note that since place and route is not deterministic, some results even come out better than the baseline. Moreover, imposing placement regions may provide the place and route tool with a better starting point.

Fig. 14 plots the corresponding area overhead introduced by the power MUX cells versus given delay overhead for forced placement with strict regions. Using the results of Section II-B, the area overhead required by the two power switches can be calculated for a worst-case voltage droop of 5%. Due to the large block sizes resulting from forced placement, area overhead is modest. The fewer the number of cuts, the larger the size of the blocks, and therefore the lower the area impact. All of the design points for this strategy incur less than 30% area overhead.

2) Cluster Boxing Results: Since unforced placements are used as the basis for this strategy, there is no increase in critical path over the baseline designs. The grid size was varied from 1×5 boxes to 15×15 boxes. Fig. 15 presents the energy overhead incurred by applying this strategy. The baseline design represents the average energy without considering any effects of cell reassignment. In general, any cell reassignment results in an increase in average energy due to an increase in static power. Nevertheless, there are several cluster boxed design points which lie between 5% and 10% energy overhead.

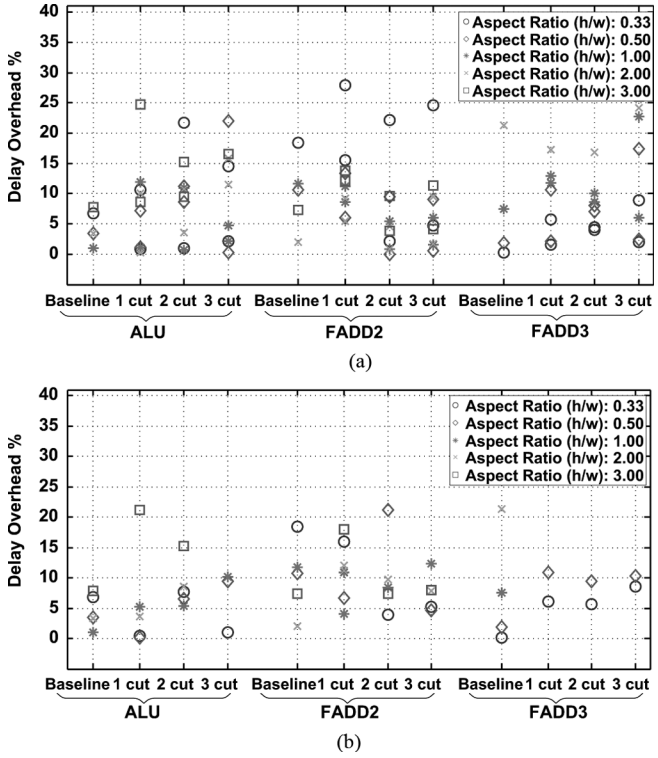


Fig. 13. Delay overheads resulting from forced placement. (a) Relaxed regions. (b) Strict regions.

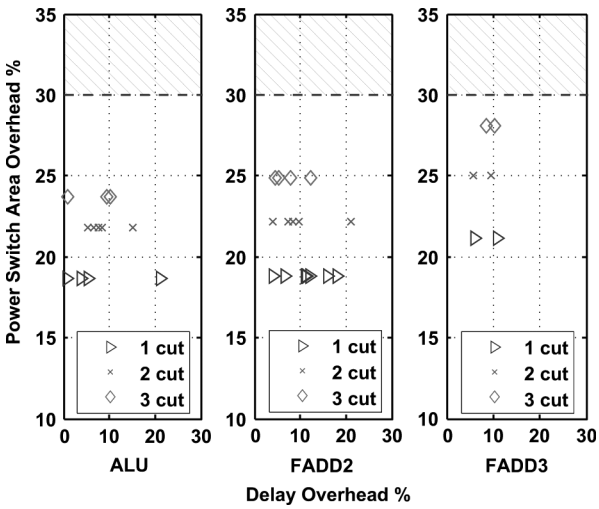


Fig. 14. Delay overhead versus area overhead for forced placement with strict regions.

The ALU and FADD3 tended to incur higher energy penalties than FADD2, due to the fact that these blocks have imbalanced substages. Since cluster boxing will reassign cells to the majority substage in a given grid block, if a substage is overrepresented, the resulting cluster boxed layout will make the previously large substages larger, and the previously small substages smaller. This creates an imbalance in the energy/delay tuning points of the design, leading to higher average energy.

Fig. 16 presents the corresponding area overhead with respect to the set of energy overhead design points. While some points have relatively low energy and area impacts, a significant

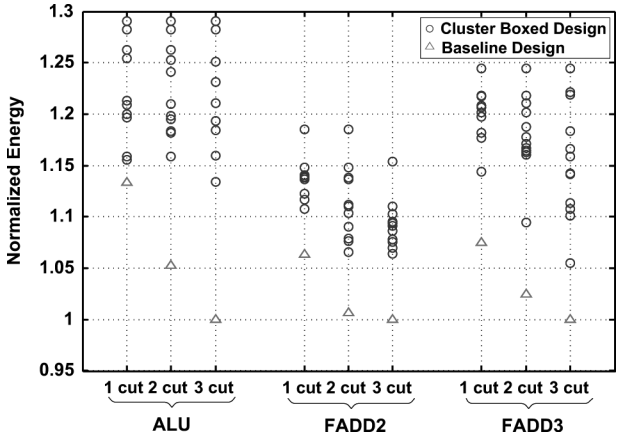


Fig. 15. Energy overhead with cluster boxing.

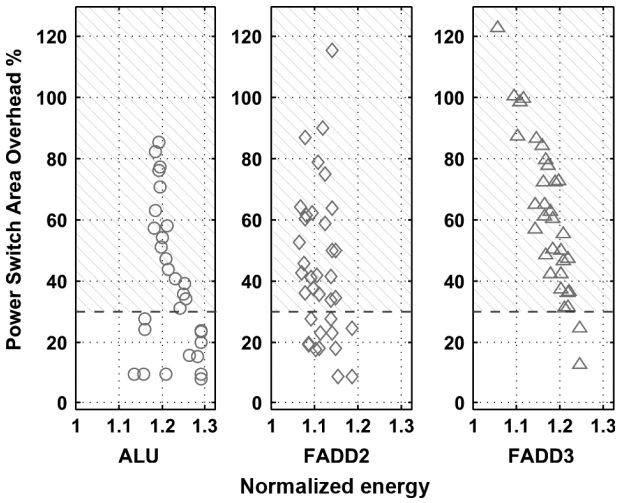


Fig. 16. Energy overhead versus area overhead for cluster boxing.

number have unacceptably high area overhead for a given energy overhead. Note that any design points lying above 30% (or any design points where individual grid blocks had area overhead larger than the grid block size) will not be able to be implemented due to lack of room for the power switches.

3) Hybrid Results: For the hybrid strategy, we apply cluster boxing to the relaxed region forced placements that have the shortest critical paths among the one, two, and three voltage interpolation cuts for each block, giving us a total of six different layouts. We note that there is a low average energy increase after cell reassignment.

To evaluate this hybrid strategy, we compare it to the forced-placement and cluster-boxing strategies by plotting the area overhead versus the energy delay squared product (ED^2). This metric is used because it is a measure of circuit performance independent of any voltage or frequency scaling techniques. Fig. 17 plots all of the design points which had low enough area overhead, such that all of their power switches could be placed within what would otherwise be used for filler cells. For both FADD2 and FADD3, the hybrid strategy with three cuts leads to a layout with the lowest ED^2. For the ALU, while the hybrid strategy with three cuts does very well, a three cut forced placement design does somewhat better. In general,

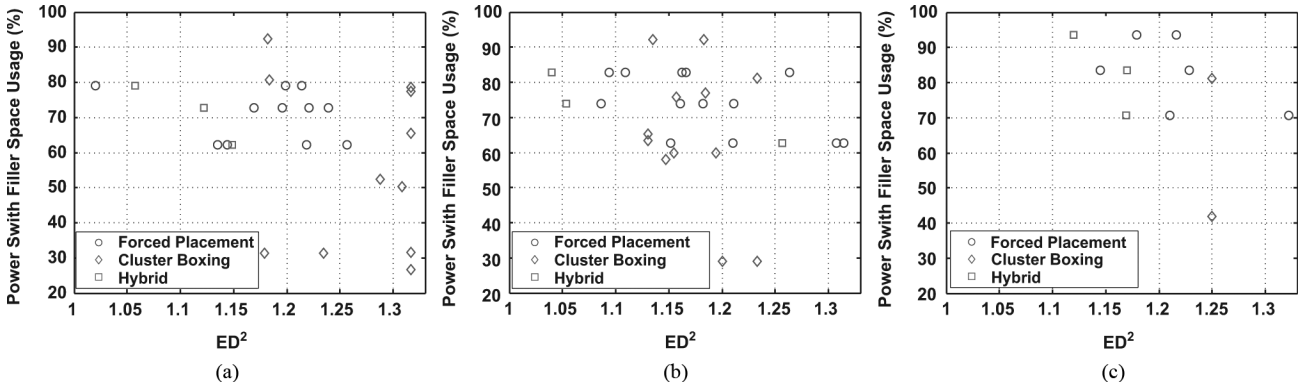


Fig. 17. Comparison of three placement strategies: (a) ALU; (b) FADD2; (c) FADD3.

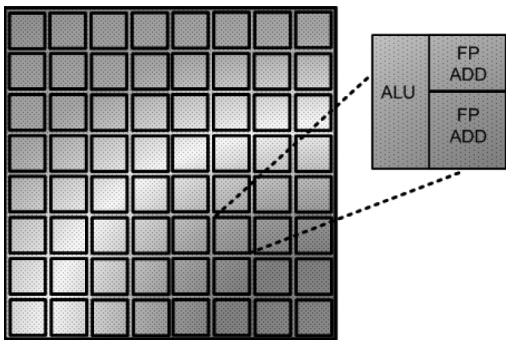


Fig. 18. 64-core CMP.

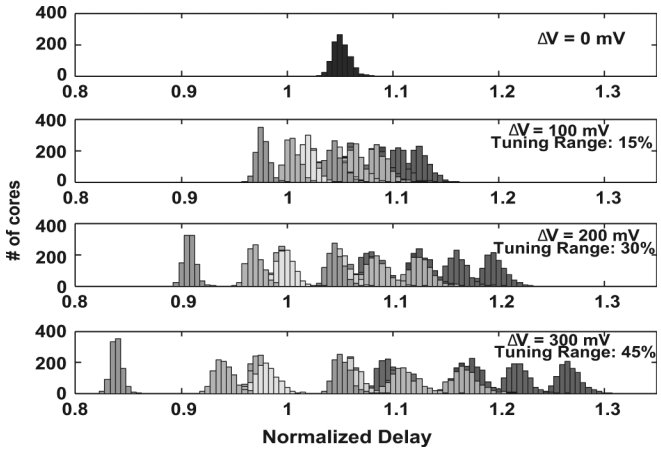


Fig. 19. Impact of random variations on 1000 cores without VI ($\Delta \mathrm{V} = 0$ mV) and with voltage interpolation.

the hybrid solutions have lower ED^2 than most, but not all, of the forced placement and cluster boxing designs for a given block.

VI. Impact of Variations

Having considered the static power penalty, cutting strategies, and implications of place and route, we now shift gears to investigate how VI can combat the impact of random and correlated variations. To demonstrate the effects of both types of variation on a full system, we consider a CMP-like scenario consisting of 64 cores arranged in an 8×8 grid, as shown in Fig. 18. Each core contains one of each of the three circuit blocks (ALU, FADD2, and FADD3). The critical path of any one block limits the frequency achievable by the entire core. This is a fair consideration, as all three blocks were synthesized aggressively for the same frequency target, and all represent typical blocks which could set the frequency of a microprocessor. As we assume a global voltage, the slowest core limits the frequency achievable by the entire CMP.

First, we carefully model every path and every instance delay in each block, taking into account factors such as loading, fast/slow inputs, and rising edge versus falling edge when calculating the delay of each instance in each path. Since each block is synthesized assuming cells operating off of 1.2 V, we can scale the delay of each cell with respect to HSPICE simulations of the cell at different voltages. By using the σ/μ delay numbers of each standard cell, gathered from Monte Carlo simulations, random variations can be applied to every cell for 1000 different cores (each containing an ALU, a FADD2, and a FADD3). For each core, the worst resulting critical path sets the frequency of that core.

To demonstrate VI's delay-tuning capabilities, Fig. 19 compares histograms of the 1000 cores with random variations, with and without voltage interpolation. The x-axis is normalized to the target delay (nominal delay) of the design without any variations. Without voltage interpolation ($\Delta \mathrm{V} = 0$ mV) and when only random variations are considered, the worst-case delay distribution of the cores is relatively tight. However, there is a shift in the mean and none of the cores can meet the nominal delay target. The three subsequent subplots show the resulting delay distributions for three different settings of $\Delta \mathrm{V}$ (VHIGH—VLOW). Each subplot contains histograms for of all 1000 cores and all 16 voltage configurations, corresponding to the three-cut strategy.

While a larger $\Delta \mathrm{V}$ allows for wider tuning range, a core only suffering random variations does not require a high tuning range, and can subsist with a $\Delta \mathrm{V}$ of 100 mV. On the other hand, a large percentage of the cores require all logic groups to operate off of VHIGH, thus being no different from simply raising the global supply voltage. In contrast, with $\Delta \mathrm{V} = 300$ mV, more voltage configurations that use a mix of VHIGH and VLOW stages can meet the nominal delay target, thus providing an advantage over simple voltage scaling. Moreover, larger tuning

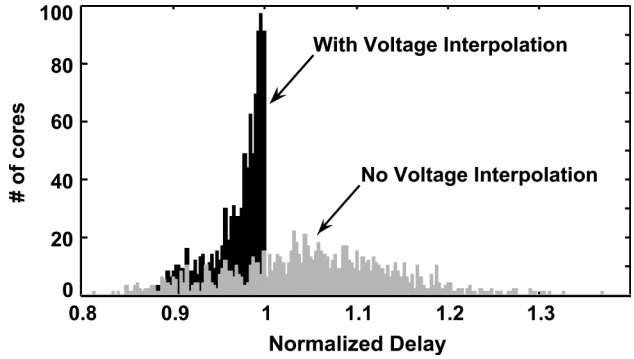


Fig. 20. Effectiveness of VI for 1000 cores with random and correlated WID variations.

range is needed to also combat the effects of correlated variations in large chips.

While a small ΔV is sufficient for random variations, large chips also suffer from D2D and WID variations. We use a multilevel quad-tree method to model correlated WID variations as described earlier in Section IV. We assume the total correlated variation has roughly equal magnitude to cell-level random variations, applied evenly across four levels of the quad tree. Fig. 20 presents the effect of adding in correlated variations to the 1000 cores considered in Fig. 19, assuming the cores are a part of a much larger multi- or many-core CMP. Correlated variations significantly increase the spread of critical path delays in cores operating off of a single nominal voltage (no VI, nominal $\mathrm{VDD} = 1.05$ V) and a large ΔV is required to provide adequate tuning range to compensate for both random and correlated variations. When VI is applied to this scenario with $\Delta\mathrm{V} = 300$ mV ($\mathrm{VHIGH} = 1.2$ V and $\mathrm{VLOW} = 0.9$ V), over 99% of the cores can be configured to meet the nominal delay target and often choose configurations with lower energy than merely using a single higher global voltage.

A. 64-Core CMP

To examine the effects of within-die correlated variations on a full system, we consider a CMP consisting of 64 of our previously analyzed cores, arranged in an 8×8 grid. Having evaluated the implications of place and route, we assess the benefits of VI in the presence of delay and energy overheads. We choose the most successful placement strategy for each block—three cut forced placement for the ALU and three cut hybrid placement for FADD2 and FADD3. These designs are only subject to energy and delay overheads, since the necessary power switches are able to be placed within areas that would otherwise be used for filler cells.

We investigate 1000 chips, by randomly choosing cores with random variations from our previous analysis and applying the multilevel quad tree to model correlated variations. The worst-case core sets the frequency of the entire chip. Fig. 21(a) shows the delay distribution of the single-voltage chips operating off of a nominal $\mathrm{VDD} = 1.05$ V, with the x-axis again normalized to a target delay without variations. With the addition of correlated variations, not a single chip using the nominal voltage can meet the original timing target. Limited by the worst-case core out of 64 per chip, the distribution tightens up and the mean shifts to longer delays when compared to the results in Fig. 20, which considers the impact of random and correlated variations on a core-by-core basis.

We first investigate the benefits of VI ($\mathrm{VHIGH} = 1.2$ V, $\mathrm{VLOW} = 0.9$ V) by comparing yields. Fig. 21(b) plots the yield versus delay for both the voltage interpolated and the single-voltage chips (at two voltage settings) for a range of target delays, normalized to the no variations target delay. At the original target delay, with both random and correlated WID variations, 72% of the VI chips meet timing, whereas none of the single-voltage chips meet timing when operating off of the nominal voltage ($\mathrm{VDD} = 1.05$ V). Raising the single global VDD to 1.2 V leads to yields marginally higher than implementing VI with $\mathrm{VHIGH} = 1.2$ V. The yields are not equivalent due to the delay overheads associated with the placement of the VI design. In either case, this maximum voltage limit caps the yield at the nominal delay target. By relaxing the timing target by 8%, 98.3% of the VI chips can meet timing, whereas only 1.1% of the chips operating off of $\mathrm{VDD} = 1.05$ V meet timing. Further relaxing the original timing target by 12% improves yield to 100% for the VI chips, whereas only 7.2% of the 1.05 V chips meet timing.

While results thus far have shown that VI can improve timing, it is important to also consider energy. Otherwise, simply meeting timing with VI is no better than scaling the global voltage with respect to the worst-case critical path in each chip to meet timing. Fig. 21(c) presents boxplots[1] for the 1000 CMP chips with VI ($\mathrm{VHIGH} = 1.2$ V and $\mathrm{VLOW} = 0.9$ V) across a number of different chip frequency targets, but again presented in terms of delay. The x-axis is normalized to the nominal, no variations delay, and the y-axis is normalized to the nominal voltage chip energy (nominal $\mathrm{VDD} = 1.05$ V). At the nominal delay target, 72% of the VI chips can meet timing and require higher than nominal energy for most of the functional chips. The median chip requires 16% more energy compared to the nominal. If we relax the timing target to the mean of the delay distribution (22% slow down) in Fig. 21(a), all of the VI chips meet timing, and the median chip consumes 13% less energy than the nominal chip energy. If the delay target is slowed down further to allow 99% of the nominal voltage chips to meet timing (33% slow down), all VI chips operate at lower than nominal energy, with the median chip consuming 25% less than nominal energy. The flattening out of the minimum energy for longer timing targets arises because VHIGH and VLOW are fixed throughout the plot.

In contrast to using VI, Fig. 21(d) presents boxplots for the same 1000 chips but using global voltage scaling. Across the range of delays shown for 1000 chips, voltage scales from a minimum of 0.9 V to a maximum limited to 1.2 V. This voltage limit again leads to yields marginally higher to that of using VI above. At the nominal delay target, there is a $>42\%$ energy penalty for the median chip using chip-wide global voltage scaling, whereas the median chips with VI suffered a 16% energy penalty. While global voltage scaling must choose a voltage that accommodates the worst-case delay path in the worst-case core, VI offers the

[1]Boxplots are graphical displays of data that measure location (median) and dispersion (interquartile range), identify possible outliers, and indicate the symmetry or skewness of the distribution.

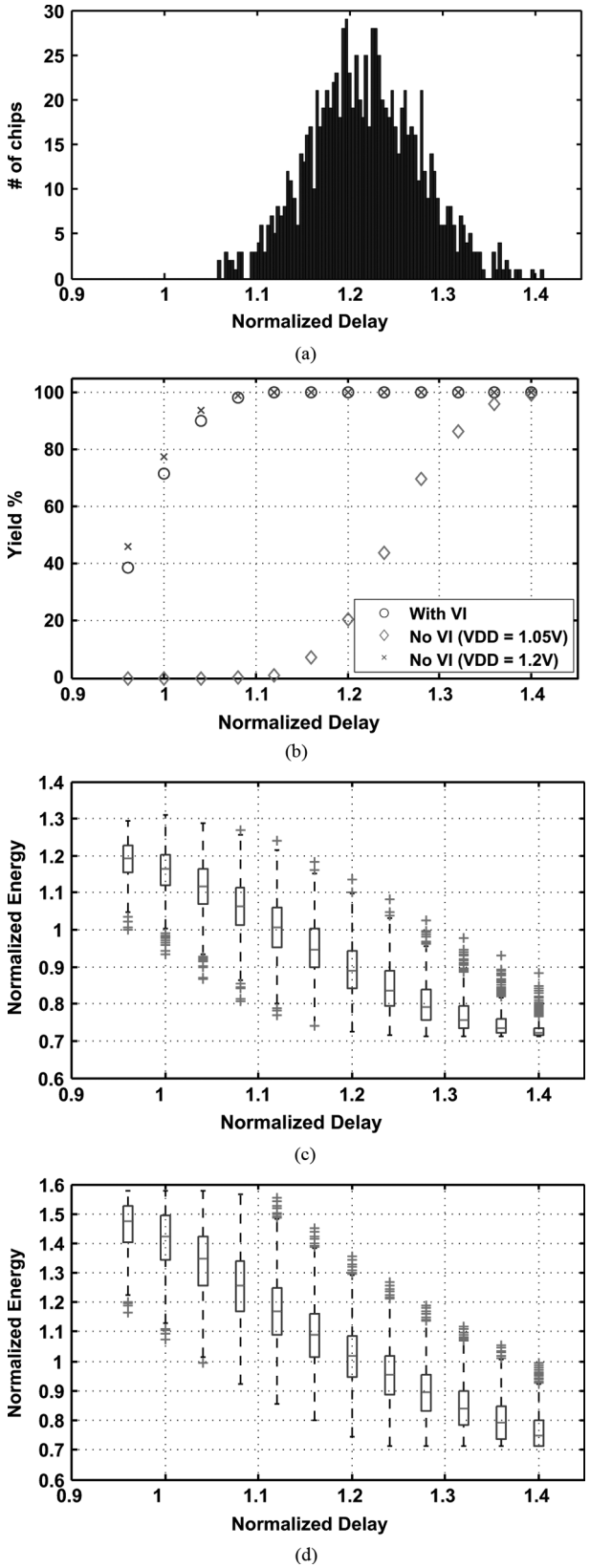


Fig. 21. Evaluation of VI applied to 1000 CMP chips suffering random and correlated variations, after place and route. (a) Delay distribution of CMP chips operating off of a single fixed nominal VDD (without voltage interpolation). (b) Performance yield with and without VI (c) Normalized energy versus normalized delay for 1000 CMPs with VI ($\mathrm{VHIGH} = 1.2$ V, $\mathrm{VLOW} = 0.9$ V). (d) Normalized energy versus normalized delay for 1000 CMPs with global voltage scaling (maximum $\mathrm{VDD} = 1.2$ V).

effect of providing fine-grained voltages to each logic block in each core to maximize energy efficiency.

VII. Conclusion

This paper has explored a number of design issues related to voltage interpolation, which can combat the deleterious effects of process variations. Tradeoffs related to the number of stage cuts and the magnitude of $\Delta\mathrm{V}$ were considered. The study centered on a number of blocks from an UltraSPARC T1 core, considering both random and correlated within-die process variations for a 130-nm CMOS process with foundry-provided standard cells. We show that for these blocks, a higher number of cuts provides benefits that outweigh the static power cost associated with more cuts. Our work also examined the costs introduced by place and route within the context of three different placement strategies. Of them, the hybrid placement and the forced placement strategies produce the best balance of delay and energy overhead. Additionally, in the context of a 64-core CMP, a $\Delta\mathrm{V} = 300$ mV is required to cover the delay spread seen by random and correlated variations. We show that, by using voltage interpolation, the median chip can hit the original timing target with only a 16% increase in energy consumption, whereas the median single-voltage chip requires a 42% increase in energy. Additionally, the median VI chip can hit the original energy budget with only a 13% delay overhead, whereas the median single-voltage domain chip suffers a 22% delay overhead.

References

[1] K. A. Bowman, S. G. Duvall, and J. D. Meindl, "Impact of die-to- die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration," *IEEE J. Solid-State Circuits*, vol. 37, no. 2, pp. 183–190, Feb. 2002.

[2] J. Gregg and T. Chen, "Post silicon power/performance optimization in the presence of process variations using individual well-adaptive body biasing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 3, pp. 366–376, Mar. 2007.

[3] J. W. Tschanz, J. T. Kao, and S. G. Narendra, "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage," *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1396–1402, Nov. 2002.

[4] X. Liang, D. Brooks, and G.-Y. Wei, "A process-variation-tolerant floating-point unit with voltage interpolation and variable latency," in *Proc. Int. Solid-State Circuits Conf.*, Feb. 2008, pp. 404–623.

[5] Sun Microsystems Inc., Santa Clara, CA, "OpenSPARC T1 chip design," 2008. [Online]. Available: http://www.opensparc.net/

[6] A. Davoodi and A. Srivastava, "Variability driven gate sizing for binning yield optimization," in *Proc. 43rd Des. Autom. Conf.*, Jun. 2006, pp. 959–964.

[7] Y. Lu and V. D. Agrawal, "Statistical leakage and timing optimization for submicron process variation," in *Proc. 20th Int. Conf. VLSI Des.*, Jan. 2007, pp. 439–444.

[8] Y. Xu, X. Li, K. Hsiung, S. Boyd, and I. Nausieda, "Opera: Optimization with ellipsoidal uncertainty for robust analog IC design," in *Proc. 42nd Des. Autom. Conf.*, Jun. 2005, pp. 632–637.

[9] A. Srivastava, T. Kachru, and D. Sylvester, "Low-power-design space exploration considering process variation using robust optimization," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, no. 1, pp. 67–79, Jan. 2007.

[10] D. Sinha, N. V. Shenoy, and H. Zhou, "Statistical timing yield optimization by gate sizing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 10, pp. 1140–1146, Oct. 2006.

[11] S. Kulkarni, D. Sylvester, and D. Blaauw, "A statistical framework for post-silicon tuning through body bias clustering," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2006, pp. 39–46.

[12] M. Mani, A. Singh, and M. Orshansky, "Joint design-time and postsilicon minimization of parametric yield loss using adjustable robust optimization," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2006, pp. 19–26.

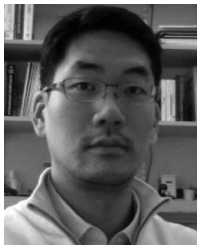
[13] R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "Mitigating parameter variation with dynamic fine-grain body biasing," in *Proc. 40th Int. Symp. Microarch. (MICRO-40)*, Dec. 2007, pp. 27–42.
[14] J. Tsai, L. Zhang, and C. Chen, "Statistical timing analysis driven post-silicon-tunable clock-tree synthesis," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2005, pp. 575–581.
[15] D. Marculescu and E. Talpes, "Variability and energy awareness: A microarchitecture-level perspective," in *Proc. 42nd Ann. Conf. Des. Autom.*, Jun. 2005, pp. 11–16.
[16] K. Usami and M. Horowitz, "Clustered voltage scaling technique for low-power design," in *Proc. Int. Workshop Low Power Des.*, Apr. 1995, pp. 3–8.
[17] K. Agarwal and K. Nowka, "Dynamic power management by combination of dual static supply voltages," in *Proc. Int. Symp. Quality Electron. Des.*, Mar. 2007, pp. 85–92.
[18] K. Brownell, G.-Y. Wei, and D. Brooks, "Evaluation of voltage interpolation to address process variations," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)* , Piscataway, NJ, 2008, pp. 529–536.
[19] K. Brownell, "Place and route considerations for voltage interpolated designs," in *Proc. Int. Symp. Quality Electron. Des.*, Mar. 2009, pp. 594–600.
[20] J. Owen and M. Steinman, "Northbridge architecture of AMD's Griffin microprocessor family," *IEEE Micro*, vol. 28, no. 2, pp. 10–18, Mar. 2008.
[21] F. Fallah and M. Pedram, "Standby and active leakage current control and minimization in CMOS VLSI circuits," *IEICE Trans.*, vol. 88-C, no. 4, pp. 509–519, 2005.
[22] J. Kao, S. Narendra, and A. Chandrakasan, "MTCMOS hierarchical sizing based on mutual exclusive discharge patterns," in *Proc. 35th Ann. Conf. Des. Autom. (DAC)*, New York, 1998, pp. 495–500.
[23] J. Kao, A. Chandrakasan, and D. Antoniadis, "Transistor sizing issues and tool for multi-threshold CMOS technology," in *Proc. 34th Ann. Conf. Des. Autom. (DAC)*, New York, 1997, pp. 409–414.
[24] A. Ramalingam, B. Zhang, A. Devgan, and D. Z. Pan, "Sleep transistor sizing using timing criticality and temporal currents," in *Proc. Conf. Asia South Pac. Des. Autom. (ASP-DAC)*, New York, 2005, pp. 1094–1097.
[25] N. Weste and D. Harris, *CMOS VLSI Design*, 3rd ed. New York: Addison Wesley, 2004.
[26] Faraday Technology Corporation, Hsin-chu City, Taiwan, "UMC 0.13 μm logic core cell library," 2008. [Online]. Available: http://www.faraday-tech.com/
[27] A. Agarwal, V. Zolotov, and D. Blaauw, "Statistical timing analysis using bounds and selective enumeration," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, no. 9, pp. 16–21, Sep. 2003.
[28] Synopsys, Mountain View, CA, "Synopsys design compiler," 2008. [Online]. Available: www.synopsys.com
[29] Cadence, San Jose, CA, "Cadence SoC encounter," 2008. [Online]. Available: www.cadence.com

Kevin M. Brownell (S'08) received the B.S. degree in computer engineering from Lehigh University, Bethlehem, PA, in 2007, and the S.M. degree in engineering sciences from Harvard University, Cambridge, MA, in 2009, where he is currently pursuing the Ph.D. degree in computer engineering.

His research interests include 3-D graphics architectures, in addition to power and variability aware design.

Ali Durlov Khan (M'97) received the B.S. degree from Worcester Polytechnic Institute, Worcester, MA, and the S.M. degree in computer science from Harvard University, Cambridge, MA, in 2008.

He spent five years as a Consulting Engineer for Paradigm Works, where he worked on projects at Avici, Mercury Computer, Agere, and Intel. He worked in the Power-Efficient Computing and Mixed-Signal VLSI Research Group. He was a Senior Design Engineer with MIT startup Tilera Corporation, working on the Tile Processor. Currently, he is a Solution Architect with Cadence Design Systems, Chelmsford, MA.



Gu-Yeon Wei (S'97–M'01) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1994, 1997, and 2001, respectively.

He is currently an Associate Professor with the Department of Electrical Engineering, School of Engineering and Applied Sciences, Harvard University, Cambridge, MA. His research interests span several areas: high-speed, low-power link design; ultra low power hardware for flapping-wing microrobotic bees; and codesign of circuits and computer architecture for high-performance and embedded processors to address PVT variability and power consumption.

David Brooks (M'02) received the B.S. degree from the University of Southern California, Los Angeles, and the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ, all in electrical engineering.

He is a Gordon McKay Professor of computer science with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA. He joined Harvard in 2002 after spending one year as a Research Staff Member at IBM T. J. Watson Research Center. His research interests include technology-aware computer design, with an emphasis on power efficient computer architectures for high performance and embedded systems.