Variation-Aware Dynamic Voltage/Frequency Scaling

Sebastian Herbert and Diana Marculescu
Department of Electrical and Computer Engineering
Carnegie Mellon University
{sherbert,dianam}@ece.cmu.edu

Abstract

Fine-grained dynamic voltage/frequency scaling (DVFS) is an important tool in managing the balance between power and performance in chip-multiprocessors. Although manufacturing process variations are giving rise to significant core-to-core variations in power and performance, traditional DVFS controllers are unaware of these variations.

Exploiting the different power/performance profiles of the cores can significantly improve energy-efficiency. Two hardware DVFS control algorithms are considered and the gains enabled by incorporating variability-awareness are demonstrated on multithreaded commercial workloads. For a design with per-core voltage/frequency islands (VFIs), the mean power per unit throughput for a simple threshold-based controller is reduced by 8.0% when variability-awareness is added. A complex greedy-search controller sees an even larger reduction of 15.4%. The variability-aware versions of the two controllers achieve power/throughput reductions of 2.1% and 9.9% relative to LinOpt*, a recent software variability-aware DVFS scheme.*

Designs which apply DVFS at a coarser granularity are also considered, and the variability-aware schemes maintain significant improvement over the -unaware ones. With four cores per VFI, variability-awareness reduces power/throughput by 6.5% and 9.2% for the threshold-based and greedy-search controllers, respectively.

1. Introduction

Dynamic voltage/frequency scaling (DVFS) is a popular method for improving microprocessor energy-efficiency. By lowering clock speed and supply voltage during frequency-insensitive application phases, large reductions in power can be achieved with modest performance loss.

This research has been funded in part by National Science Foundation Award No. CNS 00720529. Sebastian Herbert is supported by an Intel Foundation PhD Fellowship.

Technology scaling allows designers to integrate increasing numbers of cores onto a single die. Microprocessor designs are moving away from full-chip voltage/frequency control towards finer-grained methods which allow increased energy-efficiency. For example, AMD's recent quad-core Opteron allows independent frequency control of all four cores, the shared L3 cache and on-chip northbridge, the DDR interface, and four HyperTransport links [10].

However, scaling also results in worsening manufacturing process variations in key physical and electrical parameters. Variations can result in nominally correct designs failing to meet frequency or power targets. Traditionally, the interdie component of process variation dominated and was addressed through speed-binning. However, intradie variations have become increasingly important at recent technology nodes, resulting in core-to-core variations in power and performance [16]. Despite this, traditional DVFS control algorithms treat all cores as nominal.

Exposing variability information to DVFS controllers significantly improves energy-efficiency. Two hardware DVFS controllers are considered: a simple threshold-based controller (*Threshold*) and a more complex greedy-search controller (*Greedy*). Methods for exploiting core-to-core variability are proposed and the benefits on multithreaded commercial workloads are quantified. For *Threshold*, a reduction of 8.0% in power per unit throughput is obtained, while *Greedy* sees a larger power/throughput reduction of 15.4%. When tuning the variability-unaware and -aware versions of an algorithm to iso-performance, variability-awareness maintains most of its power/throughput advantage (7.0% for *Threshold* and 10.8% for *Greedy*). *Threshold* and *Greedy* are compared with *LinOpt*, a recently proposed software variability-aware DVFS scheme [28]. Variability-aware *Threshold* achieves power/throughput 2.1% lower than *LinOpt*, while variability-aware *Greedy* exhibits power/throughput 9.9% lower than *LinOpt*.

The proposed schemes are also evaluated on a design which aggregates clusters of four cores into single voltage/frequency islands (VFIs), sacrificing flexibility for reduced design complexity. Here, variability-awareness re-

sults in reductions in power/throughput of 6.5% and 9.2% for the *Threshold* and *Greedy* controllers, respectively.

The overhead of adding variability-awareness to these DVFS controllers is limited. In the case of *Threshold*, the only additional hardware is a set of fuses to be programmed at test time describing each VFI's variations. If the complex variability-unaware *Greedy* scheme is implemented in a power management microcontroller, then the cost of adding variability-awareness is once again limited to fuses. If *Greedy* is built in custom hardware, variability-awareness requires four additional multiplexers.

This study uses new numerical models for how frequency and leakage current scale with supply voltage, temperature, and process variations. These models achieve worst-case errors with respect to SPICE of 0.22% for frequency and 0.49% for leakage current, significantly outperforming analytical models used in prior microarchitecture-level studies [16, 21, 27]. Variability information is exposed to DVFS hardware via an effective process variation parameter, p_{eff}, which aggregates the impact of a VFI's variations into a single value.

The remainder of this paper is organized as follows. Section 2 details the target architecture and Section 3 the variability models. Section 4 explains the specifics of the DVFS algorithms considered in both their variability-unaware and -aware forms. Section 5 presents the experimental methodology and Section 6 the results. Section 7 gives an overview of related work and Section 8 concludes the paper.

2. Target Architecture

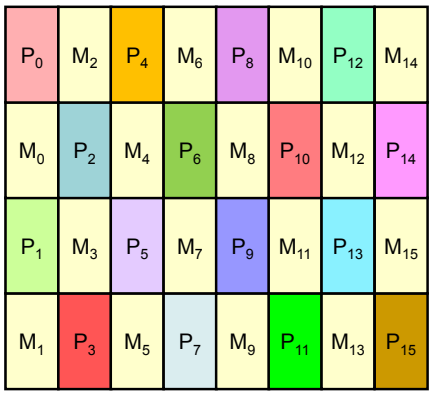
A chip-multiprocessor with 16 out-of-order SPARCv9 cores is considered, with the parameters shown in Table 1. The cores have private L1 instruction and data caches, while a single logical L2 cache is shared among the cores using a Piranha-style design [2]. The L2 is implemented as 16 independent banks distributed across the chip. Each core and L2 cache tile is a node in an 8x4 mesh on-chip network.

Table 1. Processor parameters

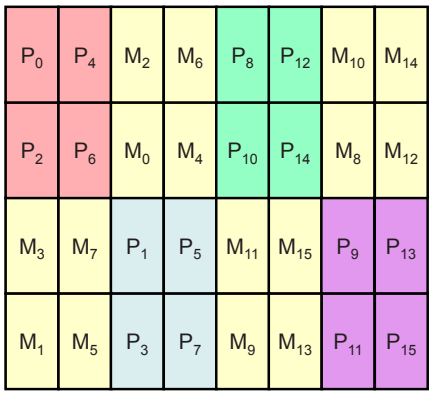
Parameter	Value
Nominal frequency	3.0 GHz
Technology	22 nm node with V_{dd} = 0.9 V
DVFS interval	50 μs
Number of tiles	16 core tiles, 16 L2 tiles
On-chip network	8 $\times$ 4 mesh
L1-I/D caches	64 KB, 64 B blocks, 2-way SA, 2-cycle load-to-use, LRU, 4R/W
L2 cache	16 $\times$ 1 MB, 64 B blocks, 16-way SA, 20-cycle hit, LRU, 1R+1W per bank
Main memory	60 ns for random access, 64 GB/s peak bandwidth
Pipeline	8 stages deep, 4 instructions wide
ROB/LSQ size	128
Store buffer size	64

The chip is divided into voltage/frequency islands (VFIs). The L2 cache, network, and memory controller always run at the nominal V_{dd}. Due to process variations, individual tiles have different maximum operating frequencies at identical supply voltages. There is a single clock domain for the L2 cache and memory controller, which is run at the maximum frequency of the slowest L2 tile, and a second one for the network, which is run at the maximum frequency of the slowest tile overall (core or L2).

Finally, there are either four or 16 clock domains for the cores. In the fine-grained VFI design, each core has its own clock and V_{dd}. The core and L2 cache tiles are interleaved at the tile granularity, giving rise to the fine-grained floorplan



(a) Fine-grained design



(b) Coarse-grained design

Figure 1. Floorplans for each design. "P" is a processor tile, "M" is a memory (L2) tile.

shown in Figure 1a. With coarse-grained VFIs, clusters of four cores share a clock and supply voltage. The floorplan is altered to make cores in a cluster physically adjacent, which makes each core VFI contiguous at a cost in the quality of the on-chip network. The floorplan for the coarse-grained design is shown in Figure 1b.

Three PLLs are used for clocking. One clocks the L2 and memory controller, one clocks the on-chip network, and the third is used to clock the cores. Each core VFI contains a clock divider to create its own local clock signal from the output of this shared PLL, as in Intel's Montecito [11]. This approach allows single-cycle frequency transitions.

Due to the narrowing gap between supply and threshold voltages, a limited set of four supply voltages is available. Supply voltages of 0.9 V, 0.8 V, 0.7 V, and 0.6 V result in nominal frequencies of 3 GHz, 2.5 GHz, 2 GHz, and 1.4 GHz. On-chip voltage regulators [14] enable fast voltage transitions taking only 5 ns. An interval of 50 μs is assumed for making DVFS decisions.

The cores on a single die support different frequencies due to intradie process variations. In the variability-unaware designs, the slowest core determines the frequency levels available to all of the core VFIs. The variability-aware designs, on the other hand, can exploit frequency asymmetry by running each core VFI at its own maximum frequency.

The only interdomain communication is between the tiles and the network. As in AMD's quad-core Opteron, asynchronous queues provide interfacing between clock domains [10], with the buffers between the tiles and their routers implemented as dual-clock FIFOs [8].

3. Variability Modeling

3.1. Parameter Variation Modeling

Variations are considered in two key parameters: threshold voltage V_{th} and effective channel length L_{eff}. A model of spatially-correlated intradie process variation developed by Sarangi *et al.* is used to generate their values [23]. Spatially-correlated L_{eff} variation is the main source of spatially-correlated V_{th} variation [7], so perfect correlation is assumed between the spatially-correlated components of the two parameters. The two are lumped into a single process variation parameter p, which gives the value of variations in standard deviations from the mean. Sarangi *et al.* determined the spatially-correlated component of intradie V_{th} variation to have $\frac{\sigma}{\mu} = 6.4\%$ and the spatially-correlated component of intradie L_{eff} variation to have $\frac{\sigma}{\mu} = 3.2\%$.

The processor die is divided into a uniform 88×88 grid, yielding 242 points per tile. Using a multiple of eight for the grid dimensions ensures the same number of grid points in every tile, given the 8x4 floorplans shown in Figure 1.

Parameter values are assumed to follow a multivariate normal distribution. The correlation between parameter values at two points is position- and direction-independent and given by a spherical function:

$$\rho(r) = \begin{cases} 1 - \frac{3r}{2\phi} + \frac{1}{2}\left(\frac{r}{\phi}\right)^3 & r \leq \phi \\ 0 & r > \phi \end{cases} \quad (1)$$

r is the distance between the points and ϕ is the distance past which the correlation is zero; these are given relative to the die size. This function decreases roughly linearly as r increases from zero and then decreases more slowly, reaching zero at $r = \phi$. Sarangi *et al.* determined $\phi = 0.5$ for a typical microprocessor die.

Interdie variations do not induce core-to-core variations and are ignored. Uncorrelated intradie variations average over the transistors in a path, and thus with $\sigma_{rand} \approx \sigma_{sys}$ (as in a typical manufacturing process [1]) the core-to-core frequency variations will be dominated by the spatially-correlated component [3]. Aggregating leakage over all the transistors within a single core ensures that uncorrelated variation will not have a large core-to-core leakage effect.

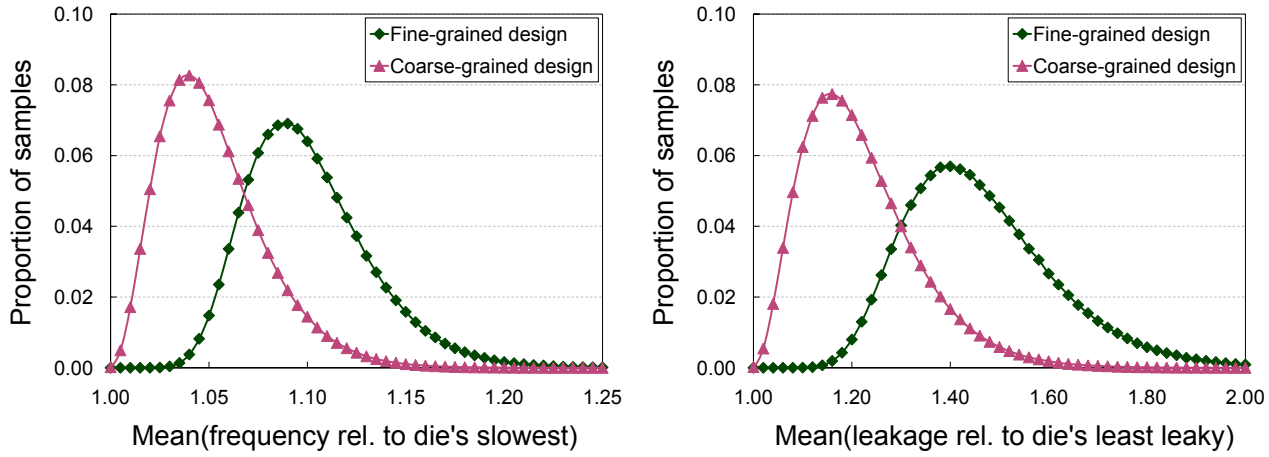
3.2. Frequency and Leakage Modeling

V_{th} and L_{eff} values are used to determine the maximum frequency and subthreshold leakage of each core across V_{dd} and temperature. SPICE simulation data obtained using the 22 nm hi-K metal gate BSIM4 predictive technology models were used to fit the model coefficients [31].

Fit data were generated on a grid of uniformly-spaced (V_{dd}, T, p) tuples. V_{dd} values were spaced between the lowest and highest levels in the processor, 0.6 V and 0.9 V. The processor has a worst-case temperature of 100 °C, enforced by dynamic thermal management. Frequency simulations were run assuming this worst case, and thus the frequency model conservatively underestimates maximum frequency. For leakage simulations, T values were spaced between 45 °C, the ambient temperature used in microarchitecture-level simulations, and 100 °C. Finally, p values were evenly spaced between -4 and 4, an 8σ range.

Fitting a response surface model generally yields a model that is accurate for parameter values within the range that was fit, but which loses accuracy outside that range. V_{dd} and T are deterministic parameters and can be bounded. However, p is a random variable. Due to spatial correlation in process variations there are entire cores which exhibit p values far from nominal when simulating a large number of dies. In order to avoid significant error for such cores, equal importance was given to all p values.

Both models are of the form $e^{P(V_{dd},T,p)}$, where $P(V_{dd}, T, p)$ is some polynomial function (for the frequency model, all the terms containing T are dropped because a constant worst-case temperature is assumed). This



(a) Distribution of mean VFI frequencies on a die

(b) Distribution of mean VFI leakages on a die

Figure 2. Distributions for the amount of inter-VFI variation

yields better accuracy than models of the form $P\,(V_{dd}, T, p)$ with the same number of coefficients. The models were fit by minimizing the maximum absolute percent error (MAPE).

V_{dd}, T, and p are scaled to $[-1, 1]$, so the value of every term which is multiplied by a model coefficient (*e.g.*, $V_{dd}^2 \cdot T \cdot p$) falls in $[-1, 1]$. Thus, the absolute values of the coefficients give some indication of the importance of the corresponding terms. Fitting starts with a full n^{th} order model and proceeds iteratively. In each iteration, the term with the coefficient with the smallest absolute value is dropped and the model is refit. The process terminates when the removal of a term would more than double the error.

The frequency model tracks the frequency of a 13-stage ring oscillator built from FO4 inverters of eight times minimum size. While the ring oscillator may not accurately track wire-dominated paths, the vast majority of microprocessor critical paths are gate-dominated [4]. The design process avoids creating wire-dominated critical paths because such paths would severely limit the benefit that a design would obtain from being scaled to a new technology [24].

Simulations were performed on a grid of 24,311 uniformly-spaced pairs of $V_{dd} \in [0.6\,\text{V}, 0.9\,\text{V}]$ and $p \in [-4, 4]$. A constant worst-case temperature of 100 °C was used. Initially, the full third-order model was fit, which contains ten coefficients. After the iterative pruning described above, the final model contains six terms.

24,000 test data points were generated on another grid of uniformly spaced (V_{dd}, p) pairs, with no values identical to those in the fitting data. Across this test data, the model's root-mean-square percent error (RMSPE) is 0.09% and its MAPE is 0.22% relative to SPICE.

The leakage model predicts the source-to-drain subthreshold leakage current in uA/um, averaged across NFETs and PFETs, for a given V_{dd}, T, and p. In the 22 nm technology used, gate leakage is controlled such that source-to-drain leakage is the dominant form of static power. Simulations were performed on a grid of 274,625 (V_{dd}, T, p) triples, with $V_{dd} \in [0.6\,\text{V}, 0.9\,\text{V}]$, $T \in [45\,^\circ\text{C}, 100\,^\circ\text{C}]$, and $p \in [-4, 4]$. Because subthreshold leakage is more nonlinear than frequency, model fitting started with the full fourth-order model. Iterative pruning reduced the model from 30 coefficients to 14 coefficients.

262,144 test data points were generated on another uniformly-spaced grid over the same V_{dd}, T, and p ranges. No parameter value in the test data was identical to one in the fitting data. On the test data, the model achieves a RMSPE of 0.21% and MAPE of 0.49% relative to SPICE.

3.3. Model Results

Inter-VFI variation was examined over 2.5 million generated dies. V_{dd} was set to the nominal 0.9 V. The leakage model was run assuming $T = 75\,^\circ\text{C}$, as this is more typical than the worst-case temperature of 100 °C assumed by the frequency model. In the microarchitecture-level simulations used to generate the performance results, the leakage model is run with thermal data dynamically generated by the HotSpot thermal simulator [25]. Figure 2 shows the results; each sample point is the average of the frequencies/leakages of the VFIs on a die relative to the frequency/leakage of the slowest, least leaky VFI on that die.

There is significantly more variation in leakage than in frequency. For fine-grained VFIs, the mean VFI leakage is 45% higher than the lowest VFI leakage on average. For coarse-grained VFIs, this is reduced to around 20%. It is primarily this large inter-VFI leakage variation that variability-aware DVFS exploits.

3.4. Exposing Variation Information

For each die, the full grid data must be reduced to a manageable size before being exposed at the microarchitecture level. Frequency variations can be accounted for using the greatest p value within each VFI, corresponding to its slowest part. For the leakage model, the p values at every grid point within a VFI must be aggregated together. This is done by finding the single effective p value, p_{eff}, which most closely tracks a VFI's leakage across V_{dd} and T.

p_{eff} is computed from four measurements. Leakages for each grid point at $(V_{dd}, T) \in \{0.6\,\mathrm{V}, 0.9\,\mathrm{V}\} \times \{45\,^{\circ}\mathrm{C}, 100\,^{\circ}\mathrm{C}\}$ are aggregated into the leakages for each VFI at each (V_{dd}, T). Each VFI's p_{eff} is then found as the single p value for that VFI that will minimize the MAPE with respect to the aggregated data across the four points.

This process is extremely effective. Due to spatial correlation, a VFI's p_{eff} value extremely rarely has to account for large regions of both very positive p values and very negative p values. Across 1,000 dies and 13,456 pairs of uniformly spaced $V_{dd} \in [0.6\,\mathrm{V}, 0.9\,\mathrm{V}]$ and $T \in [45\,^{\circ}\mathrm{C}, 100\,^{\circ}\mathrm{C}]$ values, the worst single-VFI RMSPE and MAPE are 0.006% and 0.011% for fine-grained VFIs and 0.007% and 0.014% for coarse-grained VFIs.

Figure 3 shows the p_{eff} distributions for the two chip designs across the 1,000 dies. For fine-grained VFIs, the mean p_{eff} value is -0.012, the median is -0.010, and 52.3% of VFIs have $p_{eff} < 0$. For coarse-grained VFIs, the corresponding figures are -0.021, -0.017, and 53.8%. A slight bias towards negative p_{eff} is observed, which is expected because leakage is roughly exponential in p.

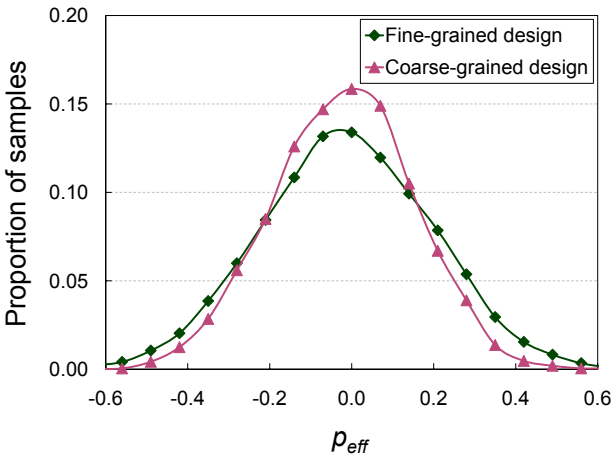


Figure 3. p_{eff} distributions

4. DVFS Algorithms

Two DVFS controllers are considered. *Threshold* is a simple threshold-based algorithm, while *Greedy* is a more complex greedy-search controller which attempts to find the operating point minimizing power/throughput. The energy-efficiency metric used to judge DVFS algorithms is power/throughput (P/T).

Variability-aware DVFS primarily addresses leakage power disparity to improve energy-efficiency. Subthreshold leakage current is strongly dependent on V_{dd} due to drain-induced barrier lowering (DIBL), which results in a transistor's V_{th} decreasing as its V_{ds} increases. For example, for relatively leaky transistors with a process variation parameter of $p = -2$ operating at $T = 75\,^{\circ}\mathrm{C}$ and $V_{dd} = 0.9\,\mathrm{V}$, the subthreshold leakage current falls by 29% for an 11% reduction in V_{dd}, 51% for a 22% reduction in V_{dd}, and 66% for a 33% reduction in V_{dd}. Leakage *power* is given by $P_{leak} = I_{leak} \cdot V_{dd}$, so the reduction in power is even greater.

To implement variability-aware DVFS, each VFI's process variations are characterized at test time via leakage current measurements and the p_{eff} describing that VFI's variations is computed. As shown in Section 3.4, p_{eff} can be accurately determined through IDDQ testing of bare die by measuring the leakage current of each VFI at only four (V_{dd}, T) points. p_{eff} values are used at runtime to bias leaky VFIs towards lower V/F levels and less leaky VFIs towards higher V/F levels. As was shown in Section 3.4, the p_{eff} distribution is roughly symmetric about 0, so approximately the same number of VFIs will be biased in each direction. The reduction in leakage from the leaky VFIs will more than compensate for the increase from the less leaky VFIs, both lowering total power and evening out the power profile across the cores.

More throughput is lost on the VFIs which are slowed down than is gained on the ones which are sped up. Frequency asymmetry is exploited to recover lost performance. Leaky cores, which are biased to lower V/F levels, also tend to have higher maximum frequencies. In the baseline design, all core VFIs run with the same set of frequency levels, set by the slowest VFI. In the variability-aware designs, fast VFIs with $p_{eff} < 0$ run at their own maximum frequencies.

Frequency asymmetry can be exploited by extending the standard testflow used during fabrication. Normally, the frequency of a fabricated chip is determined by sweeping a series of test vectors through it, increasing frequency until the slowest core fails. To exploit asymmetry, this process must continue past the failure of the slowest core to the failure of the fastest, resulting in somewhat longer test time.

4.1. *Threshold*

Threshold-unaware is a simple controller which attempts to keep some structure's utilization within given bounds. The original algorithm due to Talpes and Marculescu was applied to single-core multiple clock domain architectures

and worked by monitoring the queues between clock domains [26]. The CMP version proposed by Herbert and Marculescu considers the utilization of each VFI, computed as the number of non-spin instructions retired divided by the number of retire slots [15].

Every interval, the utilization U is compared to the up threshold T_{up} and the down threshold T_{down}. If $U > T_{up}$, the VFI does not have sufficient frequency headroom and its V/F level is raised. If $U < T_{down}$, the VFI has too much excess capacity and its V/F level is lowered. While $U \in [T_{down}, T_{up}]$, the VFI's voltage and frequency are held unchanged. T_{down} is set to 0.2 and T_{up} to 0.4, lower than the values proposed by Herbert and Marculescu and resulting in a greater bias towards preserving performance. This will be compared to *Greedy*, which considers only maximizing energy-efficiency.

Threshold-aware assigns each VFI its own thresholds based on process variations. The target utilization needs to be raised for leaky cores to bias them towards lower V/F levels and lowered for non-leaky ones to bias them towards higher V/F levels. Leakage current is roughly exponential in p, so an exponential scaling is applied. The target utilization range for VFI i is set to $[e^{-p_{eff,i}} \cdot T_{down}, e^{-p_{eff,i}} \cdot T_{up}]$. For VFIs which are leakier than nominal, $p_{eff,i} < 0$ and thus $e^{-p_{eff,i}} > 1$, while the opposite is true for the VFIs which are less leaky than nominal. This scaling results in a target utilization band of $[0.279, 0.558]$ for the single-core VFI in the 5$^{\text{th}}$ percentile in Figure 3 (a very leaky core) and $[0.139, 0.278]$ for the single-core VFI in the 95$^{\text{th}}$ percentile (a core with very low leakage).

The extra hardware needed to make *Threshold* variability-aware is minimal. If *Threshold-unaware* uses hardwired thresholds, those thresholds must be made single-writable so that they can be set at test time. If the original algorithm uses thresholds stored in some special register set by the operating system, then *Threshold-aware* can be implemented by exposing the per-VFI p_{eff} values from the hardware to the operating system so that the threshold can be adjusted by the OS before it is set. The same effect could be achieved with some hardware cost by scaling the OS-programmed thresholds in hardware.

4.2. *Greedy*

Greedy-unaware is based on the CMP extension by Herbert and Marculescu [15] of the *greedy-search* scheme proposed by Magklis *et al.* [20]. The controller attempts to operate its VFI at the V/F level which minimizes power/throughput, assuming that P/T is a convex function of V/F level. During each interval, the controller counts the number of non-spin instructions retired, as in *Threshold*. Current sensors are used to approximate the energy consumed by the VFI over the interval, allowing the computation of energy per instruction (EPI), which differs from power/throughput only in units.

Each interval, the controller compares the current and previous intervals' EPI values. If EPI has improved, the controller makes another move in the same direction as the last one. If EPI has degraded, the controller assumes that it has overshot the optimal level. It makes a transition in the opposite direction of the last one to the suspected optimal level and stays there for $N = 5$ intervals. The controller then continues exploration by making a move in the direction opposite the V/F level which preceded the hold.

A simple optimization improves the algorithm's performance relative to Herbert and Marculescu's version. When their controller reached either the highest or lowest V/F level it would stay there until the energy-efficiency metric worsened, then move away and hold. Noise in the metric values resulted in the original controller spending relatively little time at the optimal level if it was one of the two extremes. The optimization triggers a hold whenever either extreme is reached and the metric has improved, guaranteeing that at least $N + 1$ intervals will be spent there.

Although not intentionally designed to do so, *Greedy-unaware* implicitly adapts to variability because each each VFI's V/F level is chosen based on its measured power consumption. Thus, two VFIs running identical threads but with different leakage characteristics might be run at different V/F levels. However, *Greedy-unaware* attempts to minimize each VFI's local P/T. Global P/T may be improved by explicitly exposing variability information and sacrificing some throughput on a leaky VFI if a similar amount of throughput can be gained on a less leaky VFI.

Greedy-aware scales metric values exponentially to bias core V/F levels as proposed. Each interval, the scaled EPI for VFI i is computed as $SEPI_{n,i} = (e^{p_{eff,i}})^{L_{n,i}} \cdot EPI_{n,i}$. $L_{n,i}$ is the V/F level of VFI i on interval n. The V/F levels are indexed by consecutive integers, starting from 0 and with a higher index indicating a lower V_{dd}. For VFIs which are leakier than nominal, $p_{eff,i} < 0$ and $(e^{p_{eff,i}})^{L_{n,i}} < 1$ and decreases as the V/F level decreases, resulting in better SEPI values at lower V/F levels. The opposite is true for VFIs which are less leaky than nominal. For the single-core VFI in the 5$^{\text{th}}$ percentile in Figure 3, this results in the metric values being scaled by 1.00 at $V_{dd} = 0.9$ V, 0.70 at $V_{dd} = 0.8$ V, 0.49 at $V_{dd} = 0.7$ V, and 0.34 at $V_{dd} = 0.6$ V. For the single-core VFI in the 95$^{\text{th}}$ percentile, the scaling factors are 1.00 at $V_{dd} = 0.9$ V, 1.39 at $V_{dd} = 0.8$ V, 1.94 at $V_{dd} = 0.7$ V, and 2.71 at $V_{dd} = 0.6$ V.

Some processors incorporate power-management microcontrollers, such as the Foxton unit in Intel's Montecito [22]. If *Greedy-unaware* is implemented in this way, the hardware support required to make it variability-aware consists of test-time programming of measured variation parameters. If this is not the case, the overhead is still

low compared to the initial complexity of *Greedy-unaware*, consisting primarily of four multiplexers. Examining the range of scaling factors above, a resolution of 12 bits will be more than sufficient for performing variability-aware scaling. Rather than multiply the current interval's EPI by $(e^{p_{eff,i}})^{L_{n,i}}$, the previous interval's EPI is divided by $(e^{p_{eff,i}})^{L_{n,i}}$. This allows the scaling to be done during the interval by the same hardware used to compute energy per instruction at the end of the interval, requiring additional multiplexers for the two inputs and one output of this block. There are three non-trivial values of $(e^{p_{eff,i}})^{L_{n,i}}$, because for $L_{n,i} = 3$ this expression is identically one. The other three values are computed statically at test time and programmed into the chip. The controller then uses a fourth multiplexer to choose the correct scaling factor at runtime.

4.3. *LinOpt*

Teodorescu and Torrellas proposed *LinOpt*, a software-based variability-aware DVFS controller using linear programming [28]. The effectiveness of exposing variability-awareness in the manner proposed is demonstrated by comparing the *Threshold* and *Greedy* controllers to *LinOpt*.

To allow the selection of optimal voltage levels to be posed as a linear programming problem, *LinOpt* assumes that **1)** a core's frequency is directly proportional to its supply voltage, **2)** a core's throughput is directly proportional to its frequency, **3)** a core's total power draw (static plus dynamic) is a linear function of its supply voltage, and **4)** that the available supply voltages form a continuous interval $[V_{dd,min}, V_{dd,max}]$. The first two assumptions lead to a core's throughput being expressed as $T_i = a_i \cdot V_{dd,i}$ and the third leads to its power draw being written as $P_i = b_i \cdot V_{dd,i} + c_i$. The constants a_i, b_i, and c_i are recomputed at every DVFS interval, minimizing the maximum absolute error across data from the last three voltage levels used. The last assumption means that the solution to the linear programming problem will yield core supply voltages which must be rounded to the nearest available ones.

The objective is to choose the $V_{dd,i}$ to maximize the total throughput:

$$T = \sum_{i=1}^{N_{cores}} a_i \cdot V_{dd,i} \quad (2)$$

Each core's V_{dd} must be in the allowable range:

$$\forall i \in 1, 2, ..., N_{cores} : V_{dd,min} \leq V_{dd,i} \leq V_{dd,max} \quad (3)$$

Each core's power must be below its safe maximum:

$$\forall i \in 1, 2, ..., N_{cores} : P_i = b_i \cdot V_{dd,i} + c_i \leq P_{core,max} \quad (4)$$

The total power drawn by all cores must be below the target:

$$P = \sum_{i=1}^{N_{cores}} (b_i \cdot V_{dd,i} + c_i) \leq P_{total,max} \quad (5)$$

LinOpt is compared against both *Threshold* and *Greedy* in their variability-unaware and variability-aware forms. Because the *Threshold* and *Greedy* controllers allow a core to run at peak power, *LinOpt* is given additional freedom by removing the per-core power budgets from Equation 4. *LinOpt* attempts to maximize throughput given a fixed power budget, so the total power budget for the cores in Equation 5 is set equal to the power used by the cores in the scheme being compared against on a per-checkpoint basis. Thus, the comparison tests whether *LinOpt* can provide higher performance at iso-power.

5. Experimental Methodology

5.1. Microarchitecture-level Simulator

Evaluations are performed using Flexus CMPFlex.OoO, a microarchitecture-level full-system simulator for chip-multiprocessors [13]. Flexus runs real workloads on real operating systems and models processors, main memory, disks, and all other components of a computer system. It supports the use of statistical sampling using checkpointed workloads to speed up experiments and address simulation variability. Each scheme was evaluated across 10 generated dies, representing many CPU-years of simulation time.

The base CMPFlex.OoO models an implicit zero-latency crossbar between the cores and shared L2, which was replaced with the tiled architecture and on-chip network described in Section 2. The cycle-accurate network model includes routers, buffers, and links. CMPFlex.OoO also only models fully-synchronous designs. It was rewritten to model VFI-based designs with arbitrary tile frequencies.

Detailed power and thermal modeling were also added. The hybrid instruction-/microarchitecture-level dynamic power model is based on Wattch [5], with 22 nm parameters taken from ITRS predictions and observed scaling trends. The static power model is based on that proposed by Butts and Sohi [6]. The base leakage current is computed dynamically during simulator execution using the leakage current model from Section 3.2 with runtime temperature data.

These power estimates are used as inputs to the HotSpot thermal simulator developed by Skadron *et al.* [25]. The core floorplan is adapted from that used by Skadron *et al.*, merging the separate integer and FP instruction windows, moving the L1 caches, and scaling the design to the 22 nm node. The chip floorplans from Figure 1 are used. Dynamic thermal management is performed by reducing a VFI's V/F level if it approaches the 100 °C maximum temperature. Iterated simulations are used to accurately account for the effects of temperature on leakage power and power on temperature, terminating when the difference in power values from one run to the next is less than 0.01%.

5.2. Workloads Evaluated

The multithreaded workloads in Table 2 are evaluated. The online transaction processing workloads consist of TPC-C v3.0 on both *IBM DB2 v8 ESE* and *Oracle 10g Enterprise Database Server*. The Decision Support Systems (DSS) workloads are two queries from TPC-H, both on DB2. *Apache HTTP Server v2.0* and *Zeus Web Server v4.3* are evaluated on SPECweb99 under request saturation.

Each workload is simulated at 40 to 80 points, with the number determined by workload variability. For each checkpoint, non-transient state (cache, branch predictor, memory, and disk contents) is loaded and detailed simulation is performed for 500 μs. The first 400 μs are used for warmup and statistics are gathered over the last 100 μs. Non-spin user-mode throughput serves as the performance metric. This corresponds to the amount of useful work done for workloads such as TPC-H, while it has been shown for transaction-oriented workloads such as SPECweb and TPC-C that the number of user instructions per transaction is relatively constant [12] [30] (and thus user-mode throughput is proportional to transactions per minute).

5.3. Configurations Evaluated

Configuration names take the form x-y-z. x indicates the DVFS scheme used and can be "T" for *Threshold* or "G" for *Greedy*. y indicates the VFI granularity and is either "F" (fine) or "C" (coarse). z indicates the variability-awareness of the DVFS controller. "U" indicates unaware, "VA" variability-aware with frequency asymmetry, and "VS" variability-aware with symmetric frequencies.

Table 2. Workloads used

Workload	Notes
Online Transaction Processing (TPC-C)	
tpcc_db2	DB2, 100 warehouses, 64 clients, 450 MB buffer pool
tpcc_oracle	Oracle, 100 warehouses, 16 clients, 1.4 GB SGA
Decision Support Systems (TPC-H on DB2)	
tpch_qry1	450 MB buffer pool, scan-dominated
tpch_qry2	450 MB buffer pool, join-dominated
Web Server (SPECweb99)	
apache	16K connections, FastCGI, worker threading model
zeus	16K connections, FastCGI

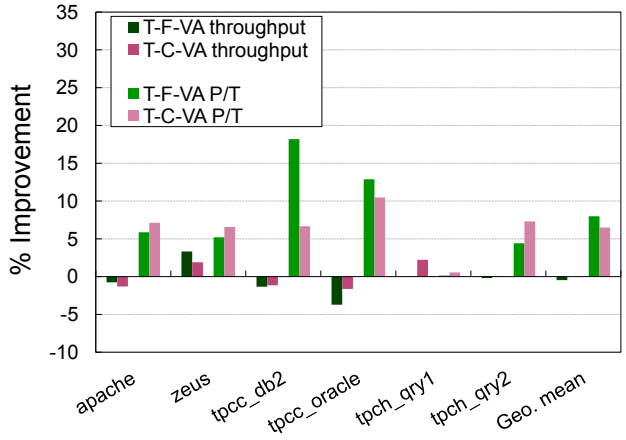
6. Results

6.1. Effectiveness of Variability-Awareness

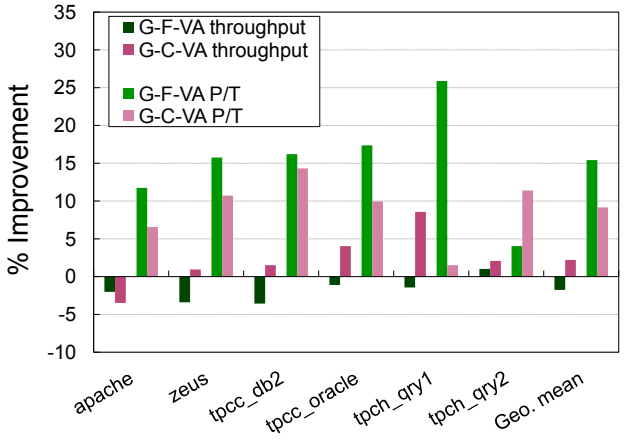
Figure 4 compares the effectiveness of variability-aware DVFS for the two chip designs. Each data point is normalized to variability-unaware DVFS on the same hardware, isolating the impact of variability-awareness.

For *Threshold*, the improvement in mean P/T over the variability-unaware design is 8.0% when using fine-grained VFIs, reduced to 6.5% when using coarse-grained VFIs. Exploiting frequency asymmetry is seen to be generally effective in removing the performance loss due to the biasing of the VFI V/F levels, with mean throughput changed by less than 0.5%.

For *Greedy*, variability-awareness brings even larger energy-efficiency gains. It improves P/T by 15.4% for fine-grained VFIs and 9.2% for coarse-grained VFIs. Again,



(a) Improvements for *Threshold-aware* relative to *Threshold-unaware*



(b) Improvements for *Greedy-aware* relative to *Greedy-unaware*

Figure 4. Throughput and P/T results for variability-aware DVFS with asymmetric frequencies

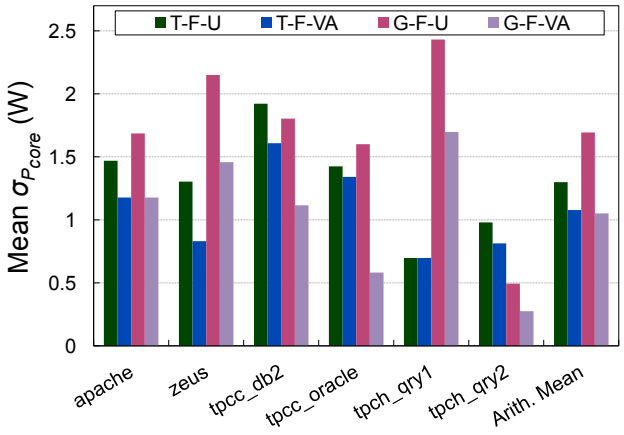


Figure 5. Power disparity metric $\sigma_{P_{core}}$

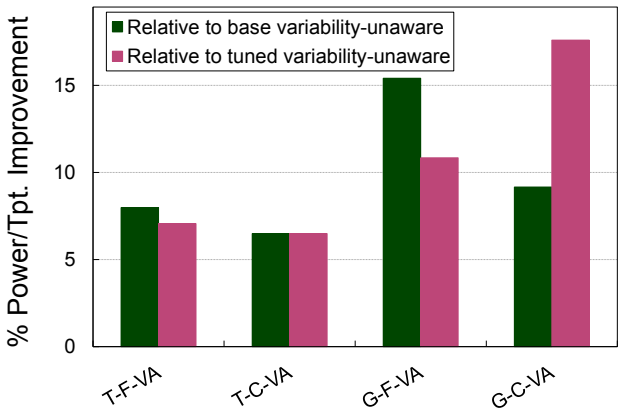


Figure 6. P/T improvement over base and tuned variability-unaware algorithms

the mean throughput changes are moderate, with a 1.8% loss for the fine-grained VFI design to a 2.2% gain for the coarse-grained VFI design.

The change in throughput when adding variability-awareness to *Greedy* is somewhat unpredictable due to interactions between the DVFS controller and the higher frequency levels set for the faster VFIs. In the case where inter-VFI frequency asymmetry is not exploited, adding variability-awareness always lowers throughput, as expected. This can be seen in Figure 8b. However, changing a domain's available frequency levels may move the optimal V/F level found by the controller. This is the effect observed in *tpch_qry1* for G-C-VA. Exposing frequency asymmetry results in higher optimal V/F levels, so the net frequency increase is much larger than that from asymmetry alone.

Besides improving energy-efficiency, variability-aware DVFS also smoothes out differences in power between the cores by biasing leaky cores towards lower V/F levels and less leaky cores towards higher V/F levels. One metric of the intercore power variation is the standard deviation of the power draws of the 16 cores, $\sigma_{P_{core}}$. Figure 5 shows mean $\sigma_{P_{core}}$ values, taken across the checkpoints within a workload and all 10 simulated dies. Variability-awareness reduces the mean $\sigma_{P_{core}}$ by 17.0% for *Threshold* and 38.0% for *Greedy*. The power drawn by a single core is on the order of several Watts, so these represent reductions in mean $\frac{\sigma_{P_{core}}}{\mu_{P_{core}}}$ of 18.0% for *Threshold* and 33.3% for *Greedy*.

6.2. Comparison at Iso-performance

Enabling variability-awareness impacts throughput to some extent. In order to examine what portion of the energy-efficiency gain brought about by variability-awareness remains after throughput changes are accounted for, tuned versions of the variability-unaware algorithms are simulated. The tuned variability-unaware controller uses the variability-aware control scheme, but with a single p_{eff} value for all VFIs on all dies. This p_{eff} value is unrelated to variations, but is chosen such that the mean throughput of the tuned variability-unaware controller matches that of the variability-aware one. The difference in mean throughput between a tuned variability-unaware algorithm and its variability-aware counterpart was under 0.05% in all cases.

Figure 6 shows the P/T improvement that the variability-aware controller exhibits over both the base variability-unaware controller (as in Figure 4) and the tuned variability-unaware version. The data show that the energy-efficiency advantage of variability-awareness does not simply stem from modulating throughput. When using fine-grained VFIs, variability-awareness degrades throughput slightly. Thus, allowing the variability-unaware controllers the same throughput degradation does narrow the advantages of variability-awareness (T-F-VA's power/throughput improvement drops from 8.0% to 7.1% and G-F-VA's drops from 15.4% to 10.8%). Variability-awareness has no impact on the mean throughput of T-C-VA. For G-C-VA, variability-awareness increases mean throughput. Matching the same throughput increase through tuning merely increases the energy-efficiency advantage of variability-awareness.

6.3. Comparison with *LinOpt*

The P/T improvements of the proposed schemes over *LinOpt* are shown in Figure 7. Fitting model coefficients and solving the linear programming problem in *LinOpt* was assumed to take no time. As described in Section 4.3, *LinOpt* is run targeting iso-power with the scheme it is being compared against. Due to the error inherent in *LinOpt*'s assumptions, it does not precisely meet its power target, with

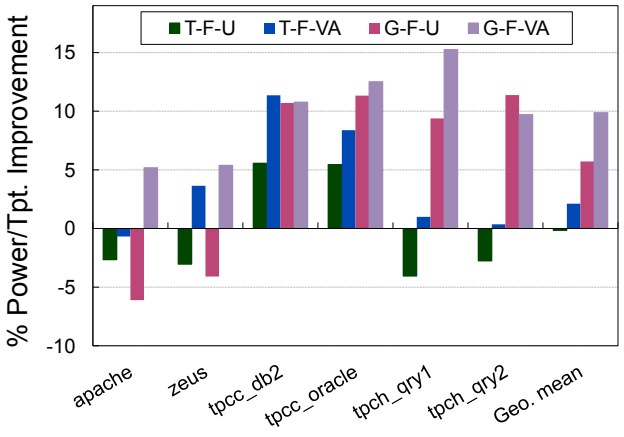


Figure 7. P/T improvement relative to *LinOpt*

LinOpt's mean power deviating from the target mean power by up to 2.5%. Thus, even though the intent was to reach iso-power, results are presented as power/throughput to ensure a fair comparison. Tweaking the power targets so that iso-power is actually achieved was found to have a negligible impact (less than 0.5%) on the P/T results.

All four of the other DVFS controllers are superior to *LinOpt* on the TPC-C workloads, while *LinOpt* outperforms *Threshold-unaware* and *Greedy-unaware* on the webserver workloads and *Threshold-unaware* on the TPC-H workloads. *LinOpt* assumes that throughput is directly proportional to core frequency. The original *LinOpt* evaluation used multiprogrammed SPEC2000 workloads [28], for which this assumption may be somewhat valid. However, it is much less accurate for multithreaded workloads with contention between threads and especially on those with large memory stall components, such as TPC-C.

LinOpt makes three linearizing assumptions. The assumption that core power is a linear function of supply voltage is fairly accurate. Across all workloads and the four configurations *LinOpt* is compared against, the RMSPE in *LinOpt*'s estimate of the power that will be used by the chosen assignment of supply voltages to cores is 9.1% for the individual core powers and 6.1% for the aggregated core power. This error is calculated with the actual voltage levels used (*i.e.*, after rounding the voltage levels chosen by the algorithm to available ones). However, the assumption that throughput is directly proportional to supply voltage is poor. The RMSPEs in the throughput estimates are 132% for the individual core throughputs and 55% for the aggregate core throughput. The TPC-C workloads display the highest RMSPE in the throughput estimation at 160% for the individual core throughputs and 71% for the aggregate throughput. The web server workloads, on which *LinOpt* does best, have the lowest RMSPE in the throughput estimation at 87% for the individual core throughputs and 34% for the aggregate throughput. The assumption that core frequency is directly proportional to supply voltage is accurate, yielding an RMSPE of 2.7% when fit to simulation data from $(V_{dd}, p) \in \{0.6\,\mathrm{V}, 0.7\,\mathrm{V}, 0.8\,\mathrm{V}, 0.9\,\mathrm{V}\} \times \{-2, -1, 0, 1, 2\}$. Thus, the source of error in the assumption that throughput is directly proportional to supply voltage is assuming that throughput is directly proportional to frequency.

LinOpt is on par with *Threshold-unaware* in mean P/T, although there is great variation between workloads depending on the accuracy of *LinOpt*'s throughput estimates. *Threshold-aware* slightly outperforms *LinOpt*, improving mean power/throughput by 2.1%. However, both versions of *Threshold* are much less complex than *LinOpt* and can be easily implemented in hardware. *LinOpt* requires a software implementation due to the complexity of fitting the model coefficients and solving the LP problem.

Comparing *LinOpt* to *Greedy-unaware* is particularly instructive because the two operate using the same inputs - runtime measurements of each core's power and throughput. *Greedy-unaware* does better on the database workloads, while *LinOpt* does better on the web server workloads. *Greedy-unaware* has 5.7% lower mean power/throughput. Finally, adding variability-awareness to *Greedy-unaware* allows it to outperform *LinOpt* on all workloads with a 9.9% lower mean power/throughput.

6.4. Effectiveness of Frequency Asymmetry

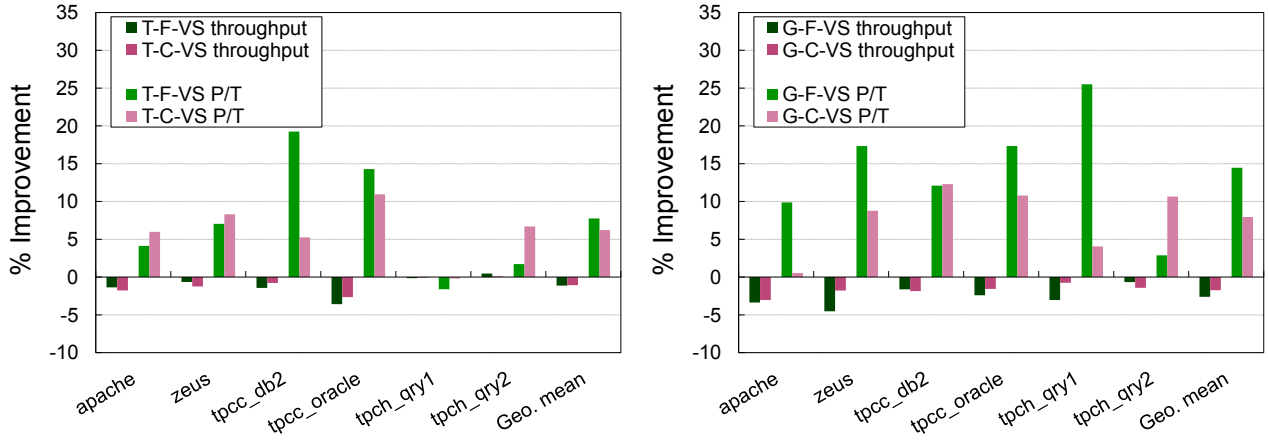
Variability-aware DVFS consists of two parts: the biasing of a VFI's V/F level in a particular direction based on its leakiness and the increase in the frequencies of fast, leaky VFIs. To isolate the impact of the two parts, experiments were performed in which the latter was disabled. Results are shown in Figure 8, to be compared to Figure 4.

While not implementing frequency asymmetry does reduce the performance of *Threshold-aware*, the difference is minimal (around a 1% difference in mean throughput regardless of the chip design). Exploiting frequency asymmetry has a larger performance impact for *Greedy-aware*, as explained in Section 6.1. The largest impact is for coarse-grained VFIs, for which throughput drops by 3.9%.

For both controllers, the difference in energy-efficiency between the symmetric frequency level and asymmetric frequency level cases is low. Thus, if the sole target of variability-awareness is improving energy-efficiency, symmetric frequency levels can yield a simpler implementation. Frequency asymmetry only needs to be exploited if it is desirable that throughput be maintained.

7. Related Work

Several researchers have investigated DVFS for chip-multiprocessors. Isci *et al.* examined global controllers op-



(a) Improvements for *Threshold-aware* relative to *Threshold-unaware*

(b) Improvements for *Greedy-aware* relative to *Greedy-unaware*

Figure 8. Throughput and P/T results for variability-aware DVFS with symmetric frequencies

erating under fixed power budgets [17] while Juang *et al.* proposed a distributed, control-theoretic scheme [18]. Li and Martínez [19] examined the performance of full-chip DVFS for CMPs. Herbert and Marculescu examined the tradeoffs involved in choosing the VFI granularity for a variety of different DVFS algorithms [15].

Schemes proposed to mitigate the impact of process and temperature variations include the use of global asynchrony and local clocking due to Marculescu and Talpes [21] and the per-core adaptive body-biasing/adaptive supply voltage scheme proposed by Humenay *et al.* [16]. Teodorescu *et al.* proposed and evaluated an implementation of dynamic body-biasing [27]. Tiwari *et al.* proposed *ReCycle*, a flexible pipeline clocking scheme which uses time-borrowing between stages to increase clock frequency [29].

There has been relatively little work on the interactions between process variations and energy-efficiency (rather than performance). Donald and Martonosi used an analytical model to evaluate the impact of process variations on energy-efficiency in chip multiprocessors [9], examining how the energy-efficiency of a parallel application varied with the number of active cores. Teodorescu and Torrellas proposed the variation-aware *LinOpt* DVFS algorithm for CMPs [28], which was evaluated earlier.

8. Conclusion

Due to spatially-correlated intradie process variations, nominally identical cores on the same chip-multiprocessor die differ in power and performance. Traditional schemes for dynamic voltage/frequency scaling have neglected these variations, but exploiting them significantly improves energy-efficiency.

A new method of expressing the variations within a single voltage/frequency island by means of an effective process variation parameter p_{eff} was proposed. The benefits of exposing this variability information to existing DVFS control algorithms were demonstrated for two hardware DVFS controllers: a simple threshold-based controller and a more complex greedy-search controller. For the former, variability-aware DVFS is able to achieve an improvement of 8.0% in power per unit throughput over the variability-unaware scheme. For the latter, the benefit is even larger at 15.4%. The proposed schemes were found to outperform *LinOpt*, a previously proposed software variability-aware DVFS scheme. The proposed schemes were also compared on a design which aggregates clusters of four cores into single VFIs. Exposing variability information to the DVFS controller was still effective: energy-efficiency improved by 6.5% for the threshold-based algorithm and 9.2% for the greedy-search algorithm.

References

[1] Y. Abulafia and A. Kornfeld. Estimation of FMAX and ISB in microprocessors. *IEEE Transactions on VLSI Systems*, 13(10), Oct 2006.

[2] L. Barroso, K. Gharachorloo, R. McNamara, A. Nowatzyk, S. Qadeer, B. Sano, S. Smith, R. Stets, and B. Verghese. Piranha: a scalable architecture based on single-chip multiprocessing. In *ISCA '00: Proceedings of the 27th annual International Symposium on Computer Architecture*, 2000.

[3] K. Bowman, S. Duvall, and J. Meindl. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *IEEE Journal of Solid-State Circuits*, 37(2), Feb 2002.

[4] K. A. Bowman, A. R. Alameldeen, S. T. Srinivasan, and C. B. Wilkerson. Impact of die-to-die and within-die parameter variations on the throughput distribution of multi-core

processors. In *ISLPED '07: Proceedings of the 2007 International Symposium on Low Power Electronics and Design*, 2007.

[5] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *ISCA '03: Proceedings of the 27th annual International Symposium on Computer Architecture*, June 2000.

[6] J. Butts and G. Sohi. A static power model for architects. In *MICRO 33: Proceedings of the 33rd annual ACM/IEEE International Symposium on Microarchitecture*, 2000.

[7] Y. Cao and L. T. Clark. Mapping statistical process variations toward circuit performance variability: an analytical modeling approach. In *DAC '05: Proceedings of the 42nd annual Design Automation Conference*, 2005.

[8] T. Chelcea and S. Nowick. Robust interfaces for mixed systems with application to latency-insensitive protocols. In *DAC '01: Proceedings of the 38th annual Design Automation Conference*, Jun 2001.

[9] J. Donald and M. Martonosi. Power efficiency for variation-tolerant multicore processors. In *ISLPED'06: Proceedings of the 2006 International Symposium on Low Power Electronics and Design*, 2006.

[10] J. Dorsey, S. Searles, M. Ciraula, E. Fang, S. Johnson, N. Bujanos, R. Kumar, D. Wu, M. Braganza, and S. Meyers. An integrated quad-core Opteron processor. In *ISSCC '07: IEEE International Solid-State Circuits Conference Digest of Technical Papers*, 2007.

[11] T. Fischer, J. Desai, B. Doyle, S. Naffziger, and B. Patella. A 90-nm variable frequency clock system for a power-managed Itanium architecture processor. *IEEE Journal of Solid-State Circuits*, 41(1), Jan 2006.

[12] R. Hankins, T. Diep, M. Annavaram, B. Hirano, H. Eri, H. Nueckel, and J. Shen. Scaling and characterizing database workloads: Bridging the gap between research and practice. In *MICRO '03: Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, 2006.

[13] N. Hardavellas, S. Somogyi, T. Wenisch, R. Wunderlich, S. Chen, J. Kim, B. Falsafi, J. Hoe, and A. Nowatzyk. SimFlex: a fast, accurate, flexible full-system simulation framework for performance evaluation of server architecture. *SIGMETRICS Performance Evaluation Review*, 31(4), April 2004.

[14] P. Hazucha, T. Karnik, B. A. Bloechel, C. Parsons, D. Finan, and S. Borkar. Area-efficient linear regulator with ultra-fast load regulation. *IEEE Journal of Solid-State Circuits*, 40(4), Apr 2005.

[15] S. Herbert and D. Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *ISLPED '07: Proceedings of the 2007 international symposium on Low power electronics and design*, 2007.

[16] E. Humenay, D. Tarjan, and K. Skadron. Impact of process variations on multicore performance symmetry. In *DATE '07: Proceedings of the Conference on Design, Automation, and Test in Europe*, 2007.

[17] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In *MICRO '06: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, 2006.

[18] P. Juang, Q. Wu, L.-S. Peh, M. Martonosi, and D. W. Clark. Coordinated, distributed, formal energy management of chip multiprocessors. In *ISLPED '05: Proceedings of the 2005 International Symposium on Low Power Electronics and Design*, 2005.

[19] J. Li and J. F. Martínez. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In *HPCA '06: Proceedings of the 12th International Symposium on High-Performance Computer Architecture*, 2006.

[20] G. Magklis, P. Chaparro, J. González, and A. González. Independent front-end and back-end dynamic voltage scaling for a GALS microarchitecture. In *ISLPED '06: Proceedings of the 2006 International Symposium on Low Power Electronics and Design*, 2006.

[21] D. Marculescu and E. Talpes. Variability and energy awareness: a microarchitecture-level perspective. In *DAC '05: Proceedings of the 42nd annual Design Automation Conference*, 2005.

[22] R. McGowen, C. Poirier, C. Bostak, J. Ignowski, M. Millican, W. Parks, and S. Naffziger. Power and temperature control on a 90-nm Itanium family processor. In *ISSCC '06: IEEE International Solid-State Circuits Conference Digest of Technical Papers*, 2006.

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

[24] J. Schutz and C. Webb. A scalable x86 cpu design for 90 nm process. In *ISSCC '04: IEEE International Solid-State Circuits Conference Digest of Technical Papers*, 2004.

[25] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *ISCA'03: Proceedings of the 30th annual International Symposium on Computer Architecture*, 2003.

[26] E. Talpes and D. Marculescu. Toward a multiple clock/voltage island design style for power-aware processors. *IEEE Trans. Very Large Scale Integr. Syst.*, 13(5), 2005.

[27] R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. Mitigating parameter variation with dynamic fine-grain body biasing. In *MICRO 40: Proceedings of the 40th annual ACM/IEEE International Symposium on Microarchitecture*, 2007.

[28] R. Teodorescu and J. Torrellas. Variation-aware application scheduling and power management for chip multiprocessors. In *ISCA'08: Proceedings of the 35th annual International Symposium on Computer Architecture*, 2008.

[29] A. Tiwari, S. R. Sarangi, and J. Torrellas. ReCycle: pipeline adaptation to tolerate process variation. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, 2007.

[30] T. Wenisch, R. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. Hoe. SimFlex: Statistical sampling of computer system simulation. *IEEE Micro*, 26(4), 2006.

[31] W. Zhao and Y. Cao. New generation of predictive technology model for sub-45nm design exploration. In *ISQED '06: Proceedings of the 7th International Symposium on Quality Electronic Design*, 2006.