Characterizing Chip-Multiprocessor Variability-Tolerance*

Sebastian Herbert
Electrical and Computer Engineering
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
sherbert@ece.cmu.edu

Diana Marculescu
Electrical and Computer Engineering
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
dianam@ece.cmu.edu

ABSTRACT

Spatially-correlated intra-die process variations result in significant core-to-core frequency variations in chip-multiprocessors. An analytical model for frequency island chip-multiprocessor throughput is introduced. The improved variability-tolerance of FI-CMPs over their globally-clocked counterparts is quantified across a range of core counts and sizes under constant die area. The benefits are highest for designs consisting of many small cores, with the throughput of a globally-clocked design with 70 small cores increasing by 8.8% when per-core frequency islands are used. The small-core FI-CMP also loses only 7.2% of its nominal performance to process variations, the least among any of the designs.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design studies

General Terms

Design, Performance

Keywords

Process variability, chip-multiprocessor, frequency islands

1. INTRODUCTION

Process variability from a range of sources is growing as technology scales, resulting in increasingly nonuniform transistor delays within each die. Meanwhile, scaling has allowed designers to integrate a growing number of processor cores, giving rise to true chip-multiprocessors (CMPs). Individual cores are thus becoming small enough that spatially-correlated intra-die process variations manifest themselves as core-to-core (C2C) power and performance variations [6].

Prior studies of C2C variations were limited to globally-clocked (GC) designs [3][6]. However, commercial CMPs such as AMD's quad-core Barcelona now offer per-core frequency scaling to enable fine-grained power control. Such frequency island (FI) designs have increased variability-tolerance because the impact of any slow path can be limited to the core containing it. Designs with a larger number of smaller cores are better able to isolate variations. However, the throughput of such designs is less sensitive to core frequency than that of designs with fewer, larger cores due to higher contention for the memory system [3].

To quantify the magnitude of these effects, an analytical model for frequency island chip-multiprocessor (FI-CMP) throughput is developed. Monte Carlo analysis is performed over generated variation maps for four constant-area CMP configurations with different core counts and sizes.

It is demonstrated that per-core frequency control yields the greatest performance improvements for CMPs consisting of many small cores. When running a large number of application threads, moving to frequency islands increases the throughput of a design with 70 small cores by 8.8%, while a design with seven large cores in the same die area only sees an improvement of 5.4%. Moreover, the small-core FI design displays the highest variability-tolerance, losing the smallest amount (7.2%) of its nominal performance to variations.

Sensitivity to the number of application threads is examined and it is shown that the small-core frequency island design displays the best variability-tolerance across thread counts. The qualitative results are demonstrated to be insensitive to the magnitude and spatial correlation of the process variation.

The main contribution of this paper is an analytical model for the throughput of FI-CMPs and its use it to quantify the performance benefits of the FI design style across a range of CMP designs. Results show that a small-core FI-CMP displays the best variation-tolerance across application thread counts and variation parameters.

Section 2 presents an overview of related work. Section 3 details the model used for chip-multiprocessor throughput, while Section 4 explains the process variation model. Sections 5 and 6 present the experimental methodology and results. Section 7 concludes the paper.

2. RELATED WORK

Marculescu and Talpes considered using frequency islands to address variability in single-core designs and reported the mean frequency increases that could be obtained for each clock domain [7]. However, CMP designs are better suited to being partitioned into FIs because synchronization only needs to be performed between the cores and shared cache, reducing the frequency of interdomain communication.

Humenay *et al.* modeled the impact of intra-die variation due to lens aberrations on C2C frequency variation in a CMP [6]. They examined using adaptive body-biasing or adaptive supply voltage to speed up slower cores, but both

* This research has been funded in part by Semiconductor Research Corporation contract No. 2005-HJ-1314. Sebastian Herbert is supported by an Intel Foundation PhD Fellowship.

DAC 2008, June 8–13, 2008, Anaheim, California, USA.

had significant power overhead. Teodorescu *et al.* evaluated using ABB to improve both power and performance [10].

Real variation patterns are noisier than the smooth one used by Humenay *et al.* Masuda *et al.* examined the characteristics of intra-die variation from fabricated wafers [8]. Work by Sarangi *et al.* focused on generating artificial intra-die variation maps with realistic spatial correlation [9].

Bowman *et al.* presented an analytical throughput model for globally-clocked CMPs (GC-CMPs) and examined the sensitivity of several CMP designs' throughputs to process variations [3]. However, they assumed completely parallel workloads and only analyzed performance *variation* rather than the relative performance of the designs. Starting from their work, this paper develops a model for FI-CMPs running workloads composed of serial and parallel execution.

3. CMP THROUGHPUT MODEL

3.1 Globally-Clocked CMPs

Bowman *et al.* developed an analytical model for the throughput of a GC-CMP with N_{cores} single-threaded cores running $N_{threads}$ completely parallel threads [3]. The cycles per instruction for a single thread are modeled as:

$$CPI = CPI_{com} + M_{L2}\left(S_{L2}\right) \times L_{miss}\left(F_{clk}\right) \quad (1)$$

CPI_{com} is the computation component of CPI and assumes a perfect L2 cache. $L_{miss}\left(F_{clk}\right)$ represents the total latency to service an L2 miss in terms of processor cycles at frequency F_{clk}. S_{L2} is the *effective* L2 cache capacity available to a single thread and $M_{L2}\left(S_{L2}\right)$ is that thread's miss rate in misses per instruction. Multiple threads may share code and/or data in the L2 cache, so S_{L2} is calculated as:

$$S_{L2} = \frac{S_{L2,total}}{N_{threads} - N_{share}\left(N_{threads}\right) + 1} \quad (2)$$

Here, $N_{share}\left(N_{threads}\right)$ is the average number of sharers for a single L2 cache line and ranges from 1 to $N_{threads}$. The miss rate is assumed to scale with the square root of the effective cache size, a common approximation.

The throughput of one of the threads is:

$$TP\left(1\right) = IPC\left(1\right) \times F_{clk} = \frac{1}{\frac{CPI_{com}}{F_{clk}} + t_{dram} + t_{link}} \quad (3)$$

t_{dram} represents the memory DRAM component of stall time (in seconds per instruction) and is given by:

$$t_{dram} = M_{L2}\left(S_{L2}\right) \times \frac{L_{dram}}{N_{pr}} \quad (4)$$

L_{dram} is the average time (in seconds) to service a request in the main memory DRAM array, while N_{pr} is the average number of parallel memory requests for an application thread. t_{link} represents the memory link component of stall time (in seconds per instruction) and is given by:

$$t_{link} = M_{L2}\left(S_{L2}\right) \times L_s \times \left(1 + \frac{U}{2\left(1-U\right)}\right) \quad (5)$$

L_s is the unloaded service latency (in seconds) of the L2 miss path, excluding the DRAM array. This resource is modeled as an M/D/1 queue, and U is its utilization:

$$U = IPC(N_{threads}) \times M_{L2}\left(S_{L2}\right) \times L_s\left(F_{clk}\right) \quad (6)$$

The dependence between $IPC\left(N_{threads}\right)$ and U results in a quadratic equation for the aggregate IPC. $IPC\left(N_{threads}\right)$ is given by the root that yields $0 < U < 1$ and the throughput is then calculated as $IPC\left(N_{threads}\right) \times F_{clk}$.

This model was validated for single- and multi-threaded applications and shown to have a mean error of 4% [3]. However, it has two serious shortcomings. First, it is restricted to GC-CMPs, despite the fact that high-performance FI-CMPs have already become available. Second, it cannot handle parallel applications with some serial portion.

3.2 A Throughput Model for FI-CMPs

To model FI-CMP throughput, the single F_{clk} needs to be replaced with one F_{clk_i} for each core. $IPC\left(N_{threads}\right)$ no longer has any meaning because there is no global cycle time. Thus, utilization is redefined with respect to wall time:

$$U = TP\left(N_{threads}\right) \times M_{L2}\left(S_{L2}\right) \times L_s \quad (7)$$

$TP\left(N_{threads}\right)$ replaces $IPC(N_{threads})$, so the L2 miss service latency is now specified in seconds (*i.e.*, L_s replaces $L_s\left(F_{clk}\right)$). Because each thread has a different throughput, $TP(N_{threads})$ can no longer be defined as $N_{threads}$ times the throughput of a single thread. Instead,

$$TP\left(N_{threads}\right) = \sum_{i=0}^{N_{threads}} TP_i \quad (8)$$

The dependence between utilization and throughput is significantly complicated by these extensions. U is a function of the aggregate throughput of the application threads, now each running at a different frequency and with a different throughput. Rather than solving for throughput, it is simpler to solve for the utilization. U can then be used to compute the throughputs of the individual threads using Equations 3, 4, and 5 with the per-core F_{clk_i} values.

Combining Equations 7 and 8 yields:

$$\frac{U}{M_{L2}\left(S_{L2}\right) \times L_s} = \sum_{i=0}^{N_{threads}} TP_i \quad (9)$$

Thus, the steady-state utilization is the $U \in (0,1)$ such that

$$0 = \sum_{i=0}^{N_{threads}} \frac{1}{\frac{CPI_{com}}{F_{clk_i}} + t_{dram} + t_{link}\left(U\right)} - \frac{U}{M_{L2}\left(S_{L2}\right) \times L_s} \quad (10)$$

t_{link} has been written as $t_{link}\left(U\right)$ to explicitly denote its dependence on U. When $U = 0$, the sum term is the processor throughput if memory system contention is ignored and the second term is zero. As U increases, the sum term decreases due to rising contention, while the second term increases linearly. Thus, the right-hand side of Equation 10 is monotonically decreasing in U on $(0,1)$.

Rather than calculating an analytical solution (as is done for the globally-clocked model), a numerical root-finding algorithm based on Brent's Method is used to find the solution. Because the function is continuous and monotonically decreasing, convergence is guaranteed. For the most complex configuration evaluated, the algorithm took an average of approximately seven iterations to converge.

3.3 Modeling Serialization

The model in Section 3.1 only handles multithreaded applications with completely parallel threads [3]. A new workload parameter, s, is used to account for serialization. It can easily be added to both the GC-CMP and FI-CMP models. The overall throughput is derived from Amdahl's Law:

$$TP = \left(\frac{s}{TP\left(1\right)} + \frac{1-s}{TP\left(N_{threads}\right)}\right)^{-1} \quad (11)$$

Serialization does not need to be between all application threads. s models slowdown because of contention for application resources, as opposed to contention for the memory system. It is obtained by curve-fitting the throughput as

a function of the number of threads after other workload parameters have been obtained. Thus, its particular value does not necessarily indicate that $s \times 100\%$ of the workload is completely serial and $(1-s) \times 100\%$ is completely parallel.

4. PROCESS VARIATION MODEL

One input to the throughput model is the set of core frequencies. These are generated through a model of spatially-correlated intra-die process variation in threshold voltage V_{TH} and effective channel length L_{eff}, developed by Sarangi *et al.* [9]. The spatially-correlated L_{eff} variation is strongly dependent on the spatially-correlated V_{TH} variation [4], and ITRS gives $\frac{\sigma_{L_{eff}}}{\mu_{L_{eff}}}$ as approximately half of $\frac{\sigma_{V_{TH}}}{\mu_{V_{TH}}}$. Thus,

$$L_{eff} = L_{eff,0} \times \left(1 + \frac{1}{2} \times \frac{V_{TH} - V_{TH,0}}{V_{TH,0}}\right) \quad (12)$$

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

$$\rho(r) = \begin{cases} 1 - \frac{3r}{2\phi} + \frac{1}{2}\left(\frac{r}{\phi}\right)^3 & r \leq \phi \\ 0 & r > \phi \end{cases} \quad (13)$$

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. Sarangi *et al.* determined ϕ to be 0.5 for a typical microprocessor die.

The final parameter is the variation magnitude. As in previous work, we assume the spatially-correlated component of intra-die V_{TH} variation to have $\sigma^{sys}_{V_{TH}} = 6.4\%$ (and thus $\sigma^{sys}_{L_{eff}} = 3.2\%$) [9]. Inter-die variation is modeled as normally-distributed with zero mean and $\sigma^{d2d}_{V_{TH}} = 5.0\%$.

Sources of uncorrelated variation, such as random dopant fluctuations, are not modeled. Such variations average over the transistors in a path, and thus with $\sigma_{rand} \approx \sigma_{sys}$ the core-to-core frequency variations will be dominated by the spatially-correlated component [1][2].

5. EXPERIMENTAL METHODOLOGY

5.1 Designs Evaluated

Four core sizes are considered: small, medium, large, and monolithic. They are based on the Intel P54C, Intel Pentium III, Intel Core 2 Duo, and an extrapolation to a future monolithic single-core design, all scaled to the 22 nm technology node by Bowman *et al.* [3]. The small core is an in-order design, while the other three are out-of-order. The experiments use a constant die size of 70 mm^2, with half of the die allocated to the cores and their private L1 caches and the other half to 16 MB of shared L2 cache and the I/O circuitry. The four designs are enumerated in Table 1.

The nominal frequencies of the designs are assumed identical at 4.0 GHz. The frequency of a core is limited by its slowest path, with delay calculated as:

$$d \propto \frac{L_{eff} V_{DD}}{(V_{DD} - V_{TH})^{\alpha}} \quad (14)$$

For short-channel MOSFETs, $\alpha = 1.3$. 2007 ITRS projections for the 22 nm node are used for other parameters.

The L2 cache is assumed to be moved off the critical path through pipelining, so the frequency of a globally-clocked design is set by its slowest core. For both GC and FI designs, a discrete set of operating frequencies is used, with bins spaced every 10% of the nominal frequency (400 MHz).

Core type	Area (mm^2)	Cores per chip
Small	0.5	70
Medium	1.5	23
Large	5.0	7
Monolithic	35.0	1

Table 1: Chip-multiprocessor designs considered

5.2 Workloads Evaluated

Six multithreaded commercial workloads are considered. 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 workloads are two queries from TPC-H, running on DB2. Finally, *Apache HTTP Server v2.0* and *Zeus Web Server v4.3* are evaluated on SPECweb99 under saturation by requests.

The workload-dependent throughput model parameters are obtained through microarchitectural simulations with the Flexus CMPFlex.OoO full-system simulator [5]. The performance of these workloads is highly sensitive to their being tuned to a particular system, so the workloads are retuned as the size and number of cores are varied. Thus, trends in the model parameters capture the fact that the same workload configuration would not be used on a system with a few large cores and one with many small cores.

The performance of each CMP design is evaluated across application thread counts from one to 70, sufficient to occupy all of the cores in any of the designs. When there are fewer threads than cores, threads are scheduled onto randomly chosen cores. Always using the fastest cores only has a significant performance impact when the number of cores far exceeds the number of threads, in which case performance will be uncompetitive under either assumption.

5.3 Model Validation

The globally-clocked throughput model in Section 3.1 was validated for both single- and multi-threaded applications and found to have a mean error of 4% [3]. The model described in Sections 3.2 and 3.3 was validated across the workloads in Section 5.2 on a moderately sized 16-core design. Random uniform $(0.5, 1.5)$ frequencies (relative to nominal) were assigned to each core and model results were compared with those of a version of Flexus implementing a detailed model of FI-CMPs, including synchronization overhead.

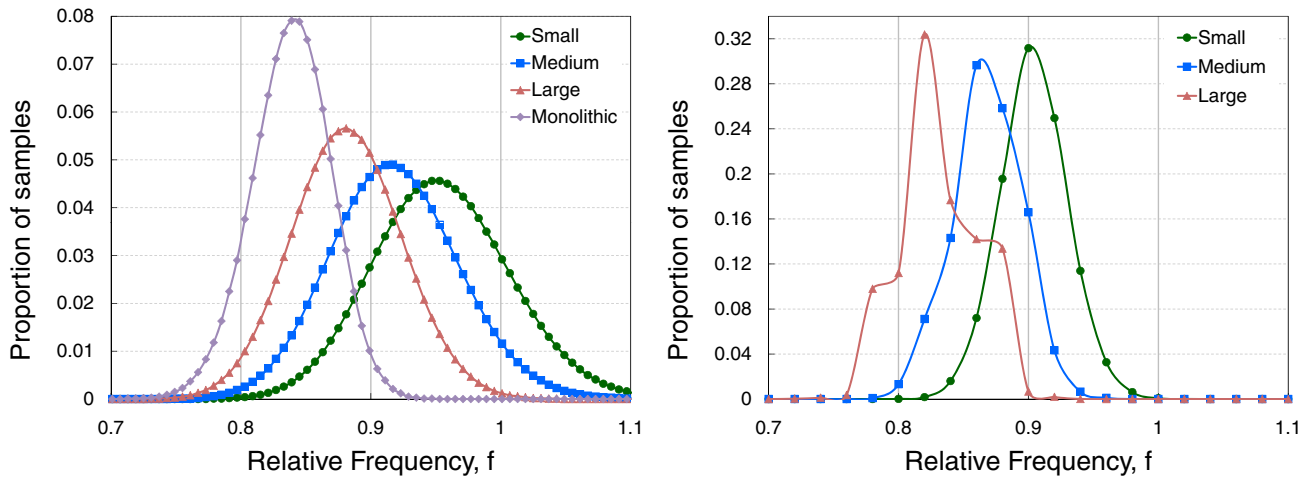
The mean absolute error in the model is approximately 5%, and no systematic under- or over-estimation is observed (on average, the model overestimates throughput by about 1%). Thus, over a large number of Monte Carlo runs the mean results from the throughput model converge to a value close to the mean that would be obtained from simulation.

5.4 Variation Parameters Evaluated

The variability parameters are set as outlined in Section 4, with default values of $\phi = 0.5$ and $\sigma^{sys}_{V_{TH}} = 6.4\%$. Sensitivities to these parameters are investigated by running the entire set of experiments over all combinations of $\phi \in \{0.25, 0.5, 0.75\}$ and $\sigma^{sys}_{V_{TH}} \in \{3.2\%, 6.4\%, 9.6\%\}$. Because inter-die variation cannot induce core-to-core variations, it is ignored by default. Its impact is examined in Section 6.4 by rerunning the experiments with $\sigma^{d2d}_{V_{TH}} = 5.0\%$.

5.5 Monte Carlo Flow

Monte Carlo analysis is performed over 100,000 variation maps. The die is broken into a 100×100 grid and the correlation function in Equation 13 is used to compute the covari-



(a) Individual core frequencies

(b) Chip mean frequencies

Figure 1: Frequency distributions show that the frequency benefit of FIs increases as core size decreases

ance of V_{TH} (and L_{eff}) values at every pair of grid points. The Cholesky decomposition of the covariance matrix Σ (C such that $C^T C = \Sigma$) is used to create variation maps with the given correlation structure. Variation map values are computed by generating a vector N of independent standard normal random variables, calculating $M = C^T N$, and then scaling to the appropriate mean and standard deviation.

A delay map is generated from the variation map according to Equation 14. For each CMP design in Table 1, the core area of the die is partitioned into the correct number of cores and the maximum frequency of each core is computed as the inverse of its longest path delay. The throughput model described in Section 3 is run with these core frequencies over all workloads and thread counts to yield the FI design throughputs as a function of $N_{threads}$. This is repeated with all frequencies set equal to the lowest one to obtain the throughputs of the corresponding globally-clocked designs.

6. RESULTS

6.1 Impact on Clock Frequency

The impact of core size on clock frequency is shown in Figure 1 for the default variation values. Figure 1(a) shows the distribution of the individual core frequencies. The curves are generated by connecting the points in a histogram of the Monte Carlo data; the y-axis values represent the proportion of samples that fall into a particular histogram bin. The histogram bins are not related to the actual frequency bins, which are shown as solid vertical lines in the plot.

Variation in the core frequency distribution increases as core size decreases, because each core's frequency is being set by a `max` over fewer path delays. Because a globally-clocked design is limited by its slowest core, any GC-CMP will draw its frequency from the monolithic distribution, regardless of the actual number or size of the cores. Due to spatially-correlated intra-die variations, the mean frequency of the globally-clocked designs is 21.6% below nominal.

Figure 1(b) shows the distribution of the FI-CMP mean frequencies after binning of the cores. Each sample point is the mean of the frequencies of the cores on a particular chip. The odd shape of the curve for the large-core design comes from the fact that with only seven cores, binning results in a relatively limited set of possible chip mean frequencies. Using per-core FIs increases mean frequency from the globally-clocked mean by 6.1%, 10.8%, and 15.4% for the large-, medium-, and small-core configurations.

6.2 Impact on Application Throughput

While designs using a larger number of smaller cores see greater frequency benefit from the FI design style, the dependence of application throughput on frequency decreases as the number of threads competing for memory resources increases [3]. Figure 2 shows how the average throughput scales with the number of threads. Results are normalized to the monolithic core. Curves are labeled as being for small (S), medium (Me), large (L), or monolithic (Mo) cores in the nominal (Nom), globally-clocked (GC), or frequency island (FI) designs. There is no data for Mo/FI because the monolithic core cannot be divided into per-core FIs. The throughput of each design increases sublinearly until $N_{threads}$ equals N_{cores}, at which point throughput becomes constant.

Table 2 details throughput at the saturation points where $N_{threads} = N_{cores}$ for each design. Dividing the throughput by the number of *active* threads (*i.e.*, the `max` of $N_{threads}$ and N_{cores}) will yield the per-core throughput at each design point. Figure 2 and Table 2 allow variability-tolerance results to be placed in a larger performance context.

The small FI cores provide the best variation-tolerance across thread counts. With only a single thread, contention for the memory system is not an issue. The medium-, large-, and monolithic-core GC-CMPs all lose approximately 18.7% of their nominal throughput to variability. Because the in-order small core spends a greater fraction of its time stalled on the memory system and unable to proceed, it loses only 16.5% of its (already low) nominal throughput. Using FIs increases throughput over that of the corresponding GC design by 10.1% for the small core, 9.3% for the medium core, and 5.5% for the large core. Thus, with the frequency island design style the small core loses only 8.1% of its nominal throughput to variations, while the medium core loses 11.2%, the large core 14.2%, and the monolithic core 18.6%.

With enough threads to saturate any of the designs, the throughput degradation from nominal experienced by the GC-CMPs depends on the number of threads contending for the memory system. The small-, medium-, large-, and monolithic-core GC designs lose 14.7%, 16.7%, 18.0%, and 18.6%, with the frequency sensitivity of throughput decreas-

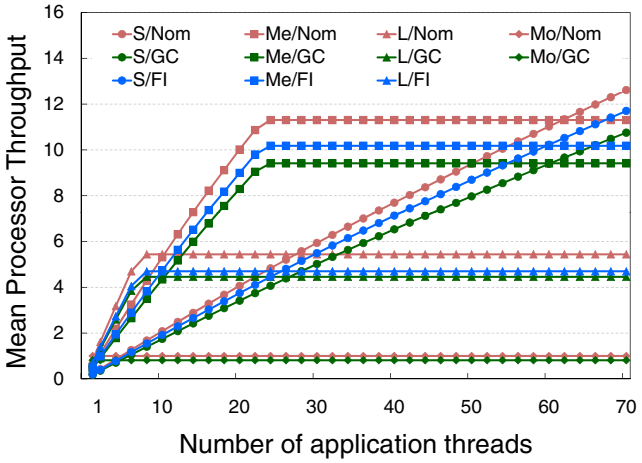


Figure 2: Throughput versus number of threads. (S)mall, (Me)dium, (L)arge, (Mo)nolithic cores / Nominal, globally-clocked, frequency island designs

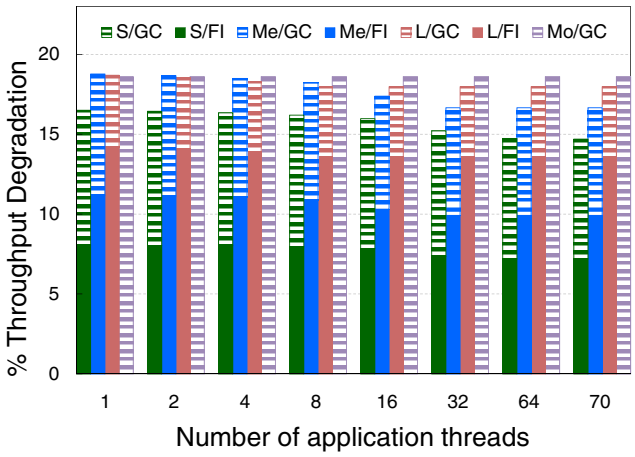


Figure 3: Throughput degradation from nominal for various thread counts

ing as the number of active threads increases. Using FIs increases throughput by 8.8%, 8.1%, and 5.4% for the small-, medium-, and large-core designs. Again, the small-core FI-CMP displays the best variability-tolerance. It loses 7.2% of its nominal throughput to variations, compared to 9.9% for the medium-core FI-CMP, 13.6% for the large-core FI-CMP, and 18.6% for the monolithic core.

In general, it is not possible to arbitrarily parallelize an application. While increasing the thread count of the web server and online transaction processing workloads merely requires providing more independent transactions to process (*i.e.*, raising the request rate), the decision support systems workloads consist of single queries and are thus more difficult to parallelize. Many other types of workloads also have limited parallelism.

Figure 3 shows the mean throughput degradation from nominal experienced by each design across thread counts. The degradations for the FI designs are shown as solid bars placed in front of striped bars representing the degradations for the globally-clocked designs. Thus, the visible portion of the striped bars represents the throughput that is *gained* by moving from a globally-clocked design to a frequency island one. The results change smoothly between the $N_{threads} = 1$ and $N_{threads} = 70$ cases discussed earlier, with the amount of throughput lost decreasing slightly as the thread count rises due to increasing contention. The qualitative results are consistent across thread counts and the small-core FI design always displays the best variation-tolerance.

Several factors contribute to the superior variability-tolerance of the small cores. First, the globally-clocked design loses the least throughput when it uses small cores because these in-order cores experience more frequent memory stalls. Second, as the thread count increases past the saturation points of the other designs, the 70 small cores see the most contention for memory. Due to these two factors, the small cores are less sensitive to frequency variations. However, because the small cores are better at isolating spatially-correlated variations, they actually experience higher frequency increases than larger cores and thus display the largest throughput gains in moving to the FI design.

FIs improve the performance of designs with smaller cores the most and thus provide bias towards such designs. For example, with ten threads L/GC outperforms M/GC, but M/FI outperforms L/FI. The same effect is seen at the point where small cores become better than medium cores.

6.3 Impact of Frequency Island Granularity

Because designing with a large number of FIs might not be feasible, the experiments are also run under a constant FI count. The monolithic core still uses a single frequency, and the other designs are all broken into seven FIs. Evaluations are performed with 70 application threads and the results are shown in Figure 4.

Combining multiple FIs degrades their ability to isolate intra-die variations. The small-, medium-, and large-core FI designs now all see the same chip mean frequency increase over their GC counterparts of approximately 6.1%. Because a greater number of active threads implies that throughput is less sensitive to frequency, this means that the FI design style now gives the largest throughput gain to the large-core design. The FI small-, medium-, and large-core designs see throughput improvements of 3.9%, 4.6%, and 5.4% and now lose 11.4%, 12.8%, and 13.6% of their nominal throughput to process variations. Despite the fact that the large-core design saw the greatest percent throughput increase, the small-core FI design still displays the highest performance and highest variability-tolerance.

CMP Design		Average Application Throughput Relative to Monolithic											
Core Type	N_{cores}	$N_{threads} = 1$			$N_{threads} = 7$			$N_{threads} = 23$			$N_{threads} = 70$		
		Nom	GC	FI	Nom	GC	FI	Nom	GC	FI	Nom	GC	FI
Small	70	0.22	0.18	0.20	1.47	1.23	1.35	4.64	3.90	4.28	12.61	10.75	11.70
Medium	23	0.56	0.45	0.50	3.77	3.08	3.36	11.30	9.41	10.18	11.30	9.41	10.18
Large	7	0.82	0.66	0.70	5.44	4.46	4.70	5.44	4.46	4.70	5.44	4.46	4.70
Monolithic	1	1.00	0.81	N/A	1.00	0.81	N/A	1.00	0.81	N/A	1.00	0.81	N/A

Table 2: Comparison of CMP design throughput

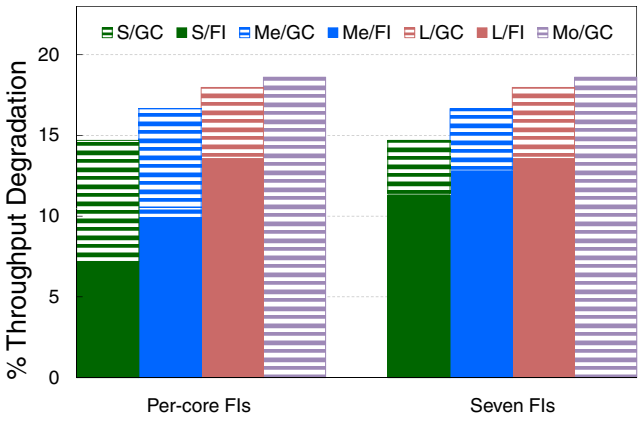


Figure 4: Throughput degradation from nominal with per-core FIs and only seven FIs

6.4 Sensitivity to Variation Parameters

Because little is known about what the variation characteristics will be at the 22 nm node, these experiments are rerun with higher and lower levels of spatial correlation and variation. In all cases, $N_{threads} = 70$. Figure 5 shows the throughput degradation from nominal experienced by each design under low (L), medium (M), and high (H) degrees of spatial correlation ($\phi \in \{0.25, 0.5, 0.75\}$) and variability magnitude ($\sigma^{sys}_{V_{TH}} \in \{3.2\%, 6.4\%, 9.6\%\}$).

As expected, the throughput degradation of all designs worsens with increasing variability magnitude. Moreover, it also worsens with decreasing spatial correlation because lower spatial correlation implies more cross-die variability.

The variability-tolerance results are qualitatively the same across all variation configurations. The FI design style always enables higher variability-tolerance than the globally-clocked style. Moreover, the small-core FI-CMP always sees the most benefit from the FI design style and has the best variability-tolerance. The same set of experiments was also run assuming $\sigma^{d2d}_{V_{TH}} = 5.0\%$ and the resulting mean throughputs changed by less than 1% in all cases.

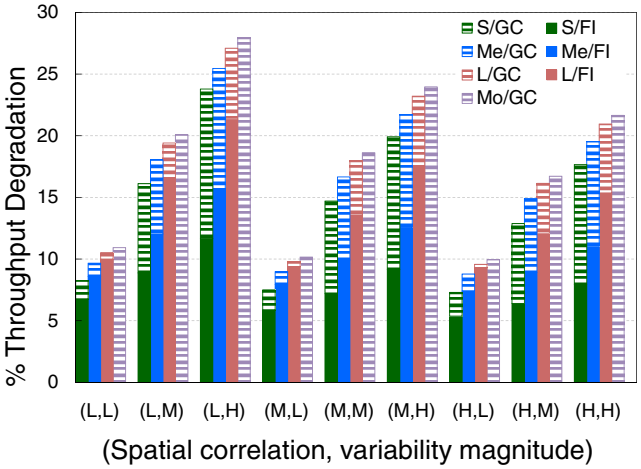


Figure 5: Throughput degradation from nominal as a function of variation parameters with $\sigma^{d2d}_{V_{TH}} = 0.0\%$

7. CONCLUSION

Current chip-multiprocessor designs implement per-core frequency scaling to enable fine-grained power control. Designers can use per-core frequency control to isolate the impact of slow paths to the cores they occur in and thus reduce the amount of performance lost to variability.

The throughput of globally-clocked and frequency island CMPs was compared across a range of core counts, core sizes, thread counts, and variability parameters. Designs with more, smaller cores were found to have higher variability-tolerance than those with fewer, larger cores, and the FI design style always increased variability-tolerance. A design consisting of many small cores in a frequency island organization was found to consistently show the greatest throughput improvement over its globally-clocked counterpart and was found to have the best variability-tolerance overall.

8. REFERENCES

[1] Y. Abulafia and A. Kornfeld. Estimation of FMAX and ISB in microprocessors. *IEEE Transactions on VLSI Systems*, 13(10):1205–1209, Oct 2006.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] H. Masuda, S. Ohkawa, A. Kurokawa, and M. Aoki. Challenge: Variability characterization and modeling for 65- to 90-nm processes. In *CICC'2005: Proceedings of the IEEE 2005 Custom Integrated Circuits Conference*, 2005.

[9] 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.

[10] 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.