Variation-Aware Speed Binning of Multi-core Processors

John Sartori*, Aashish Pant†, Rakesh Kumar*, Puneet Gupta†
*ECE Department, University of Illinois at Urbana Champaign
†EE Department, University of California at Los Angeles
*E-mail:sartori2@illinois.edu
†E-mail:apant@ee.ucla.edu

***Abstract*—Number of cores per multi-core processor die, as well as variation between the maximum operating frequency of individual cores, is rapidly increasing. This makes performance binning of multi-core processors a non-trivial task. In this paper, we study, for the first time, multi-core binning metrics and strategies to evaluate them efficiently. We discuss two multi-core binning metrics with high correlation to processor throughput for different types of workloads and different process variation scenarios. More importantly, we demonstrate the importance of leveraging variation model data in the binning process to significantly reduce the binning overhead with a negligible loss in binning quality. For example, we demonstrate that the performance binning overhead of a 64-core processor can be decreased by 51% and 36% using the proposed variation-aware core clustering and curve fitting strategies respectively. Experiments were performed using a manufacturing variation model based on real 65nm silicon data.**

***Index Terms*—Multi-core, Binning, Performance, Process Variations**

I. Introduction

Performance (or speed) binning refers to test procedures to determine the maximum operating frequency of a processor. It is common practice to speed bin processors for graded pricing. As a result, even in the presence of manufacturing process variation, processors can be designed at the typical "corners", unlike ASICs, which are designed at the worst-case corners. Binning a processor also sets the expectations for the consumer about the performance that should be expected from the processor chip.

In the case of uniprocessors, the performance of a processor is strongly correlated with its frequency of operation. As a result, processors have traditionally been binned according to frequency [8]. However, for chip multiprocessors, the appropriate binning metrics are much less clear due to two main considerations.

1) If binning is done according to the highest common operating frequency of all cores (one obvious extension to the uniprocessor binning metric), good performance correlation of the binning metric would only be observed when the maximum operating frequencies of all cores are very similar. We speculate that this assumption will not hold true in the future based on the following observations.
   - The transition from multi-core to many-core would mean several tens to hundreds of cores on a single die. In this case, all the cores are unlikely to have similar maximum safe operating frequencies.
   - With scaling, technology process variation is increasing. There is no obvious process solution to variability in sight. ITRS [16] predicts that circuit performance variability will increase from 48% to 66% in the next ten years. Moreover, many-core die sizes may scale faster than geometric technology scaling [10], facilitated by future adoption of 450mm wafers and 3D integration. As a result, core-to-core frequency variation is likely to increase in coming technology generations.
2) The second reason why binning metrics may need to be re-evaluated for multi-core processors is that a good binning metric should not only correlate well with the maximum performance of the chip (in order to maximize producer profits and consumer satisfaction), but should also have acceptable time overhead for the binning process. As we show in this paper, different binning metrics have different binning overheads, and therefore, the tradeoff between correlation to performance and timing overhead should be evaluated carefully.

In the simplest and most general form of speed binning, speed tests are applied and outputs are checked for failure at different frequencies [29]. The testing may be structural or functional in nature [5], [8], [9]. The total test time depends on the search procedure, the number of speed bins, and the frequency distribution of the processor. To the best of our knowledge, this is the first work discussing speed binning in the context of multi-core processors.

In this paper, we make the following contributions.

- We explore, for the first time, speed binning in the context of multi-core processors.
- We propose two multi-core binning metrics and quantify their correlation with absolute performance as well as their testing time overheads for various kinds of workloads.
- We demonstrate that leveraging data from the process variation model can have a significant impact on binning efficiency and propose several variation-aware binning strategies.

Our results show that variation-aware binning strategies can reduce testing time significantly with little or no degradation

in performance correlation.

II. Modeling Variation

An accurate, physically justifiable model of spatial variability is critical in reliably predicting and leveraging core-to-core variation in the binning process. Though most design-end efforts to model spatial variation have concentrated on spatial correlation (e.g., [14], [15]), recent silicon results indicate that spatial dependence largely stems from across-wafer and across-field trends [12]. [6] assumes the source of core-to-core variation to be lithography-dependent across-field variation. Though a contributor, across-field variation is smaller compared to across wafer variation [11] (even more so with strong RET and advanced scanners). In light of these facts, we use a polynomial variation model [4] for chip delay, similar to those proposed in [11], [12], [13], having three components: (1) systematic (bowl-shaped) across wafer variation[1]; (2) random core-to-core variation (arising from random within-die variation); and (3) random die-to-die variation (e.g., from wafer-to-wafer or lot-to-lot variation).

$$V_d(x,y) = A(X_c+x)^2 + B(Y_c+y)^2 + C(X_c+x) + D(Y_c+y) + E(X_c+x)(Y_c+y) + F + R + M \quad (1)$$

where $V_d(x,y)$ is the variation of chip delay at die location x,y; X_c, Y_c are the wafer coordinates of the center of the die ($(0,0)$ is center of wafer); x,y are die coordinates of a point within the die; M is the die-to-die variation and R is the random core-to-core variation. A,B,C,D,E,F are fitted coefficients for systematic across-wafer variation. **We use a fitted model as above based on real silicon data from a 65nm industrial process [4]**[2]. The goal of the binning process is to accurately classify a chip into one of n bins (where n is decided based on business/economic reasons) in light of the above variation model.

III. Binning Metrics

Traditional uniprocessor binning strategies, which sort chips according to maximum operating frequency, may fail to adequately characterize multicore processors, in which within die process variation given by Equation 1 can be substantial. In this section, we propose and discuss two simple binning metrics that recognize the frequency effects of process variation. We assume that individual cores are testable and runnable at independent operating frequencies [22], [23], [24], [25], [26], [27] though our discussion and analysis would continue to hold in other scenarios.

A. Min-Max and Σf

Min-Max stands for the minimum of the maximum safe operating frequencies for various cores of a chip multiprocessor. The *min-max* metric is computed using equation 2, where n represents the number of frequency bins, m represents the number of processor cores, and f_{ij} is a successful test frequency or 0 if core j fails the i^{th} test.

$$min\text{-}max = \min[\max[f_{ij}|_{i=1}^{n}]|_{j=1}^{m}] \quad (2)$$

The second binning metric that we evaluate is Σf. While frequency represents the primary means of increasing the performance of uniprocessors, new conventional wisdom dictates that the performance of multiprocessors depends on increasing parallelism [17]. Thus, ranking processors according to maximum attainable aggregate throughput represents a fitting binning strategy. Ideally, aggregate throughput should be maximized when every core operates at its maximum frequency. Consequently, we calculate the Σf metric using equation 3.

$$\Sigma f = \sum_{j=1}^{m} \max[f_{ij}|_{i=1}^{n}] \quad (3)$$

B. Correlation to Throughput

In terms of correlation of the metric with the throughput of the chip, *min-max* is conservative and therefore, should demonstrate good correlation only for workloads with regular partitioning (parallel or multi-threaded workloads) in which the load is distributed evenly between all cores. For other workloads that have inherent heterogeneity (multi-programmed workloads), Σf should demonstrate good correlation, especially when runtimes are designed to take advantage of the heterogeneity inherent in systems and thread characteristics. In fact, for multi-programmed workloads, the magnitude of miscorrelation between actual throughput and Σf depends on the extent of disparity between the workloads that run on various cores. One drawback of Σf is that it may increase the binning overhead, although we show in this paper that utilizing knowledge of variation trends can help to keep the overhead in check.

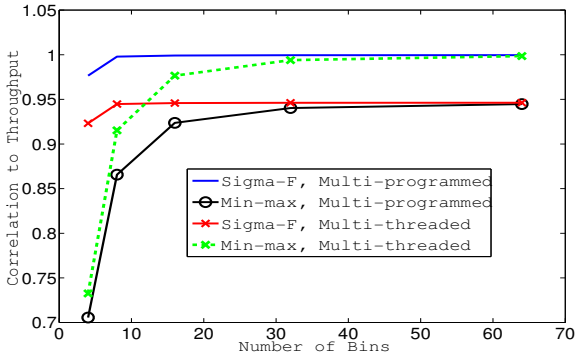


Fig. 1. Correlation of *min-max* and Σf to throughput for multi-programmed and multi-threaded workloads.

Figure 1 compares the correlation of *min-max* and Σf to actual throughput for multi-programmed and multi-threaded workloads using Monte-carlo simulations on 100,000 dice, each die being a 64 core processor in 65nm technology on a 300mm wafer (please refer to section V for further details on experimental setup). It is evident that Σf is a better metric for multi-programmed workloads while *min-max* performs better

[1]Example physical sources of across-wafer bowl-shaped variation include plasma etch, resist spin coat, post exposure bake [4].

[2]For this model, mean = 4GHz, $\sigma_{bowl} = 0.128$GHz, $\sigma_R = 0.121$GHz, $\sigma_M = 0.09$GHz.

for multi-threaded workloads for moderate to large number of bins. This is because the performance of multi-threaded benchmarks depends on the speed of the slowest executing thread (because of thread synchronizations in the benchmarks) which is nicely captured by *min-max*. Also, the correlation of Σf and *min-max* to the throughput of multi-programmed and multi-threaded workloads respectively, converges to 1 asymptotically with the number of bins. This is because, finer binning granularity leads to more precise estimation of maximum core frequencies. Conversely, when the number of bins is small, we observe rather poor performance correlation for the metrics.

To compare the two metrics, consider the asymptotic case of very large n and m and completely random core-to-core variation (i.e., A, B, C, D, E, F, M all equal zero in equation 1). In this simplified case, Σf converges to $m \times mean\_frequency$ while *min-max* converges to $(E(Min_{i=1...\infty} f_i) = 0$, i.e., for multi-programmed workloads, we expect the *min-max* to be a progressively worse metric as the number of cores in a die increases or the variation increases.

C. Binning Overhead

The binning overhead depends on the specific testing methodology that is used. On one extreme lies the case where individual cores are tested one at a time and on the other extreme is the case where all cores are tested simultaneuosly in parallel. While the latter reduces test time compared to the former, it results in higher power consumption of the circuit during test. With ever increasing number of cores within a multiprocessor, parallel testing of all cores leads to very high test power. Hence, testing is usually performed by partitioning the design into blocks and testing them one at a time [2], [1], [3]. For our work, we assume that cores are tested one at a time. Note that the analysis is also extensible to cases where a group of cores are tested together in parallel.

To calculate the binning overhead for *min-max* on a processor with n frequency bins and m cores, we use binary search[3] (i.e. frequency tests are applied in a binary search fashion) to find f_{max} for every core. However, the search range will reduce progressively. The worst case arises when f_{max} for every core is 1 bin size less than the f_{max} found for the previous core. In this case, the worst-case number of tests that need to be performed can be computed as $(log(n!) + m - n)$ (assuming $m \geq n$). The best case binning overhead for *min-max* would be m tests.

To fully evaluate the Σf metric, the maximum operating frequency of each core must be learned. Using binary search, this process performs, at worst, $m \times logn$ tests[4]. The best case is still m tests. We will show the average case runtime results of both these testing strategies using monte-carlo analysis.

[3]In this work, we assume that if a core works at a certain frequency, it is guaranteed to work at all lower frequencies. This stems from the specific case of using binary search in conjunction with the minmax metric. The constraint can be easily avoided by adding one more test per core (i.e., testing it at the minmax frequency)

[4]Note that this expression and the expressions corresponding to min-max ignore the bias introduced in binary search by the probability distribution of the frequencies themselves.

It should be noted from the above discussion that the binning overhead for Σf is always equal to or higher than that of *min-max* and this remains true even when simple linear search (i.e. frequency tests are applied in a simple linear fashion, which is the case with most industrial testing schemes) is used instead of binary search. Moreover, the disparity between binning times for min-max and Σf is never higher for binary search than for linear search. For *min-max*, the worst case overhead is on the order of n^2 and the best case is m tests. For Σf, the worst case number of tests is on the order of $m \times n$ and the best case is m tests. This is also shown in Figure 2 by performing Monte-Carlo simulations on a 64 core multi-processor in 65nm technology with a 300mm wafer. In this work, we use binary search for comparing test time overheads of various binning strategies but as explained above, our proposed analysis and results will hold for linear search as well.

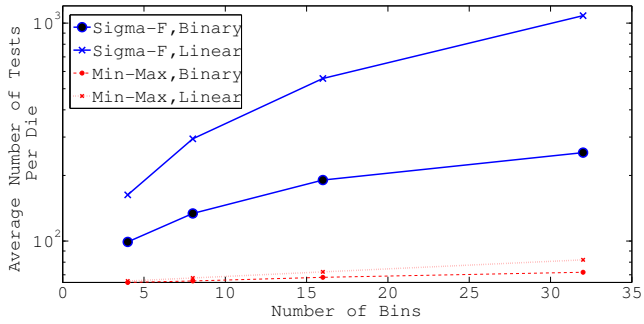


Fig. 2. The increase in overhead because of linear search for frequency binning is higher for Σf than *min-max*.

IV. Using the Variation Model to Reduce Binning Overhead

The binning metrics described above, as well as the binning strategies for those metrics, are agnostic of the process variation model. The overhead of binning using those metrics, however, depends strongly on the process variation model. In this section, we advocate the use of variation-aware binning strategies. We argue that the overhead of binning can be considerably reduced by making the binning strategies variation model-aware. The maximum safe operating frequency (f_{max}) of a core can be strongly predicted (i.e. mean with standard deviation around it) based on the process variation model. Therefore, the process variation model can give a smaller frequency range within which the search should be performed.

A. Curve Fitting

We propose curve fitting as a technique for reducing testing time overhead by trimming the range of frequencies at which a core must be tested. The curve-fitting strategy involves using the variation model (equation 1) to approximate the expected frequency (in GHz) as well as the standard deviation $(= \sqrt{(\sigma_M^2 + \sigma_R^2)})$ of a core, given its location within a die and die location within the wafer. Therefore, we can identify the center (= mean) as well as the corners (= +/-$k\sigma$) of a new,

tighter search range. If the core falls outside of this range (decided by k), we assign the core to the lowest frequency bin. Curve fitting reduces both the average and worst-case testing time for each core.

B. Clustering

Another strategy for reducing the binning overhead can be to create a hybrid metric which incorporates the advantages of each of the original metrics – namely, the low testing overhead of min-max and the high performance correlation of Σf. This behavior can be achieved by clustering the cores in a chip multiprocessor and then using min-max within the clusters (low binning overhead advantage) while using Σf over all clusters (high correlation to maximum throughput advantage). To further reduce the overhead of binning, a process like curve fitting can be applied, where the process variation model is used to identify the search range for f_{max} of a core. We refer to this combination of clustering and curve fitting as *smart clustering*

In order to improve the performance correlation within the cluster and minimize the binning overhead (especially when across-wafer variations are high), clusters can be chosen intelligently to minimize frequency variation (and hence loss of correlation) within a cluster. To this end, the cluster size can be set to be inversely proportional to the spread of frequency mean (calculated from the bowl-shape in equation 1) within the cluster. In general, the dice close to the center of the bowl (typically close to the center of the wafer) will see large cluster sizes, while clusters are smaller for the dice closer to the edge of the wafer. We do not evaluate variable clustering in this paper due to the relatively low across-wafer variations that our current process variation models suggest.

V. Methodology

We model chip multiprocessors with various numbers of cores on the die for different technologies. Each core is a dual-issue Alpha 21064-like in-order core with 16KB, 2-way set-associative instruction cache and data cache. Each core (1 mm^2 at 65nm) on a multiprocessor has a private 1MB L2 cache ($0.33MB/mm^2$ at 65nm). We assumed a gshare branch-predictor [7] with 8k entries for all the cores. The various miss penalties and L2 cache access latencies for the simulated cores were determined using CACTI [18]. We model the area consumption of the processors for different technologies using the methodology in [19].

We considered two types of workloads – multi-programmed workloads and multi-threaded workloads. Table I lists the ten benchmarks used for constructing multi-programmed workloads and the three multi-threaded benchmarks. The benchmarks are chosen from different suites (SPEC, IBS, OOCSB, and Mediabench) for diversity. The parallel applications (CG, FT, MG) are chosen from the NAS benchmark suite and run to completion. The class B implementations have been used.

Multi-programmed workloads are created using the sliding window methodology in [21]. For multi-programmed workloads, the performance of a multiprocessor is assumed to be the sum of the performance of each core of the multiprocessor,

TABLE I
Benchmarks used

Program	Description
ammp	Computational Chemistry (SPEC)
crafty	Game Playing: Chess (SPEC)
eon	Computer Visualization (SPEC)
mcf	Combinatorial Optimization (SPEC)
twolf	Place and Route Simulator (SPEC)
mgrid	Multi-grid Solver: 3D Potential Field (SPEC)
mesa	3-D Graphics Library (SPEC)
groff	Typesetting Package (IBS)
deltablue	Constraint Hierarchy Solver (OOCSB)
adpcmc	Adaptive Differential PCM (MediaBench)
CG	Parallel Conjugate Gradient (NAS)
FT	Parallel Fast Fourier Transform (NAS)
MG	Parallel Multigrid Solver (NAS)

derated by a constant factor. The methodology is accurate for our case, where each core is assumed to have a private L2 cache and a memory controller [19]. The methodology was shown to be reasonable for our benchmarks even for processors with shared L2 [19], due to the derating factor.

After fast-forwarding an appropriate number of instructions [20], multi-programmed simulations are run for 250 million cycles. As mentioned before, parallel applications are run to completion. The frequency of each core is determined by the variation model. Simulations use a modified version of SMTSIM [21].

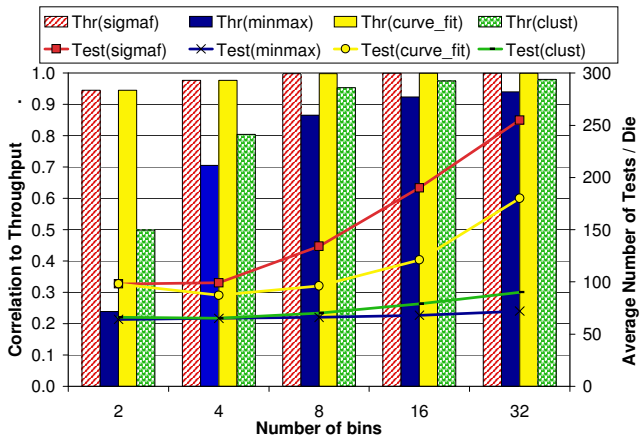
VI. Analysis of Results

In this section, we compare the binning metrics and the various evaluation strategies in terms of their overheads as well as their correlation to throughput. We run Monte-Carlo simulations using 100,000 dice. Unless specified otherwise, each die is a 64-core processor (256 mm^2) in a 65nm technology 300mm wafer, binned using 8 frequency bins. *Curve fitting* and *smart clustering* use a search range of $\pm 3\sigma$ (where σ accounts for the random die to die and within die variations), while Σf and the baseline *clustering* approach search the entire frequency range for f_{max}. We use the process variation model as described by Equation 1, with $\sigma_{bowl} = 0.128$GHz, $\sigma_R = 0.121$GHz, $\sigma_M = 0.09$GHz, based on a fitted model from a 65nm industrial process.

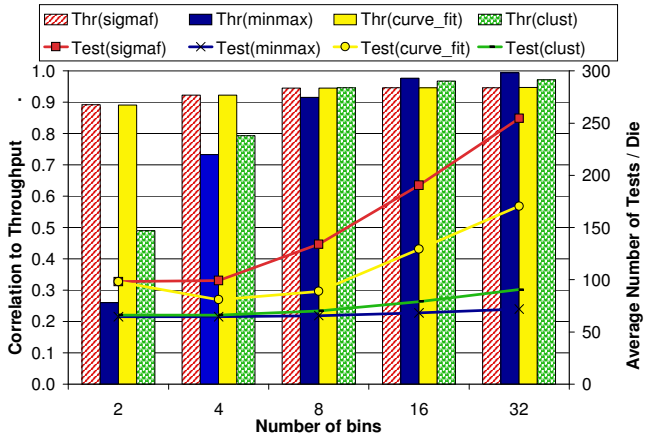
A. Dependence on Number of Bins

Figure 3 shows how binning overhead and throughput correlation vary with the number of frequency bins for multi-programmed (Fig. 3(a)) and multi-threaded (Fig. 3(b)) workloads. Using 100,000 data points (processor dice), we calculate correlation between the average of the maximum throughput of the various workloads on a processor (where cores run at different frequencies dictated by the variation model) and the value of the metric when following a given binning strategy. Note that performance of a thread often does not vary linearly with frequency due to pipeline hazards, memory accesses, etc., so it is unlikely that correlation will be 1 for any binning metric.

There are several things to note in these graphs.

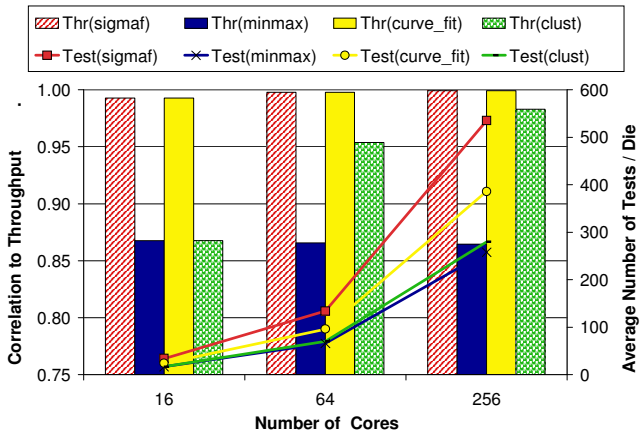


(a) multi-programmed

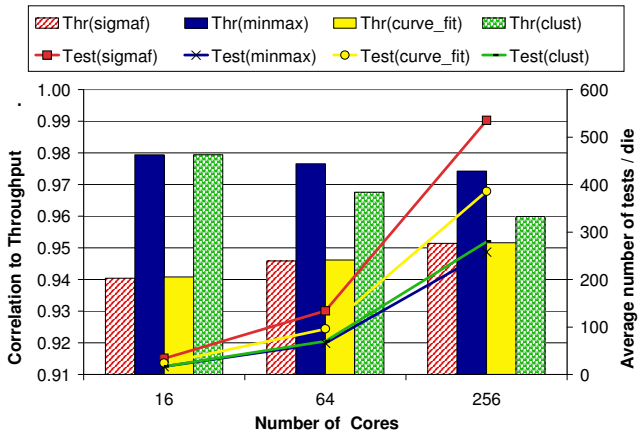


(b) multi-threaded

Fig. 3. Correlation of various binning metrics to actual throughput and their binning overhead for (a), multi-programmed benchmarks and, (b) multi-threaded benchmarks, with varying number of bins.



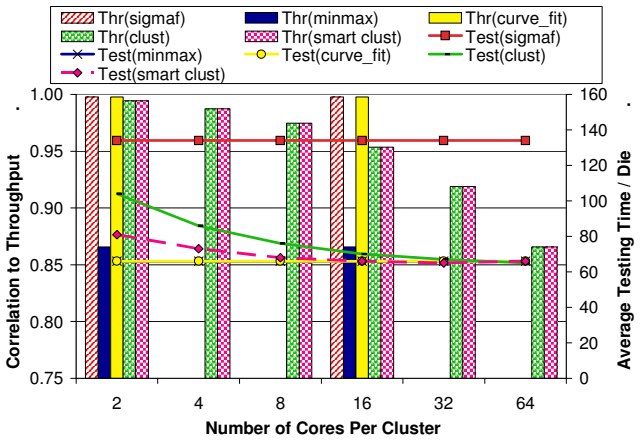
(a) multi-programmed



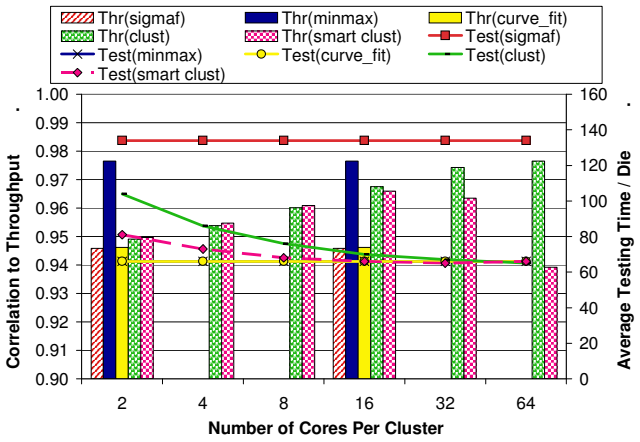
(b) multi-threaded

Fig. 4. Correlation of various binning metrics to actual throughput and their binning overhead for (a), multi-programmed benchmarks and, (b) multi-threaded benchmarks, with varying number of cores in the multi-processor.

- *First,* Σf *achieves significantly better correlation to throughput than* min-max *for multi-programmed workloads*. This is not surprising, considering that the throughput of a thread often depends on the frequency of the core it is running on, and for multi-programmed workloads, every thread execution is independent. *min-max* fails to account for variation in frequency (and therefore, average throughput) between individual cores.
- While the correlation of min-max to throughput suffers for multi-programmed workloads, *min-max actually surpasses* Σf *for multi-threaded benchmarks as the number of bins increases*. This is due to the fact that synchronization in the parallel benchmarks causes performance to be constrained by the slowest thread, since faster threads must wait at synchronization points until all threads have arrived.
- Correlation is especially low for a small number of frequency bins. This is because the binning process picks an overly conservative frequency as f_{max} for a die in that case. Even the relative performance of *min-max* (as compared to Σf) worsens as the number of frequency bins is decreased.
- *In terms of binning overhead,* min-max *is significantly faster than* Σf*, especially for large number of bins (70% faster for 32 bins)*. This is because while Σf involves doing binary search over the full frequency range (over all frequency bins) for every core, *min-max* progressively reduces the search range and requires very few tests per core, on average. *minmax* and Σf have comparable overheads for small number of bins since the search range is reduced.
- The graph also shows that *curve_fit (the approach of using variation model aware curve fitting to approximate* Σf*) has performance correlation to throughput that is equivalent to that of* Σf. This is because a range of 6σ ($\pm 3\sigma$) is searched for *curve_fit*, which is often big

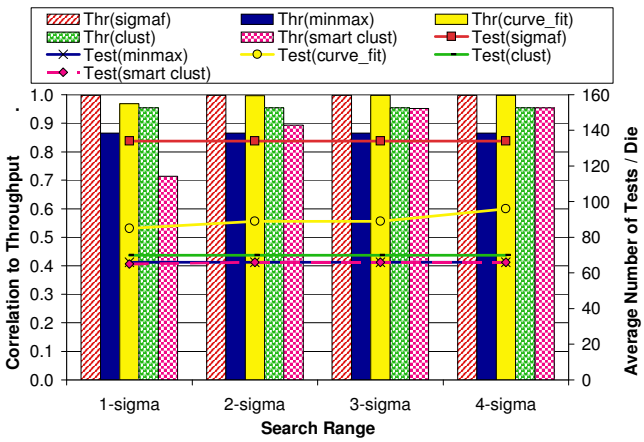


(a) multi-programmed

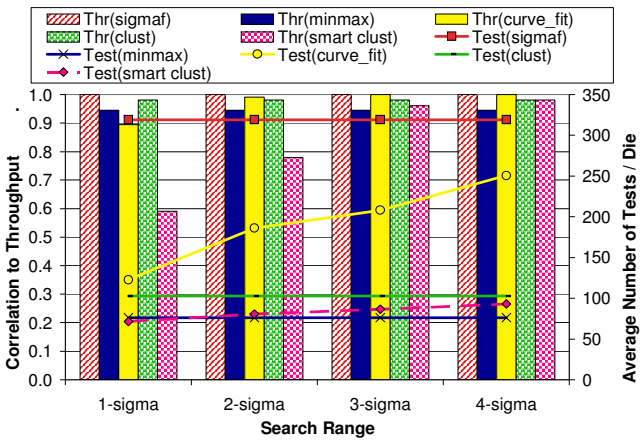


(b) multi-threaded

Fig. 5. Correlation of various binning metrics to actual throughput and their binning overhead for (a), multi-programmed benchmarks and, (b) multi-threaded benchmarks, with varying number of cores per cluster. This just affects clustering and other plots are shown for reference.



(a) 8 frequency bins



(b) 64 frequency bins

Fig. 6. Correlation of various binning metrics to actual throughput and their binning overhead for (a), 8 bins and, (b) 64 bins, with varying search range. Here, σ refers to total standard deviation of die-to-die and core-to-core variation. This just affects variation-aware binning strategies and other plots are shown for reference.

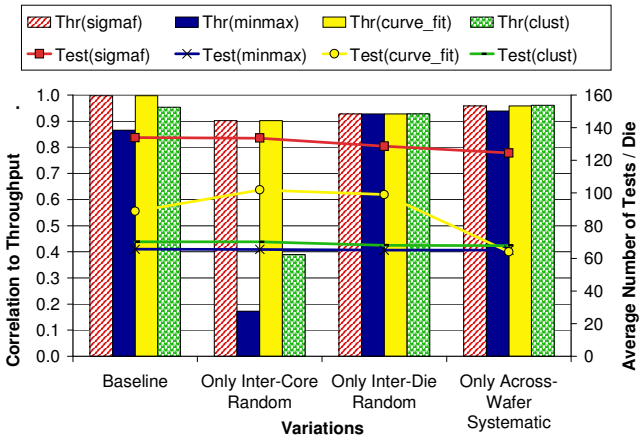
enough to allow the discovery of the true f_{max} of a core. *In terms of binning overhead, curve_fit is significantly faster than Σf (36% for our baseline architecture)*. This is because the range of frequencies that are searched for *curve_fit* is directed by the variation model and is therefore, relatively small. Overhead is greater than that for *min-max* because of the need to estimate the f_{max} for every core.

- *Clustering-based strategies (the approach of using clustering to approximate Σf) result in a smaller binning overhead than curve_fit (26% for the baseline, results are shown for a cluster size of 16)*. Clustering that relies on the variation model to reduce the search range for f_{max} of the cores (*smart clust*) is faster than the naive approach that performs search over the full range for all cores (6% improvement in test time for the baseline case). In terms of correlation to throughput, clustering-based strategies lie between Σf and *min-max* for both types of workloads. This is not surprising, considering that clustering represents a hybrid between the two schemes.

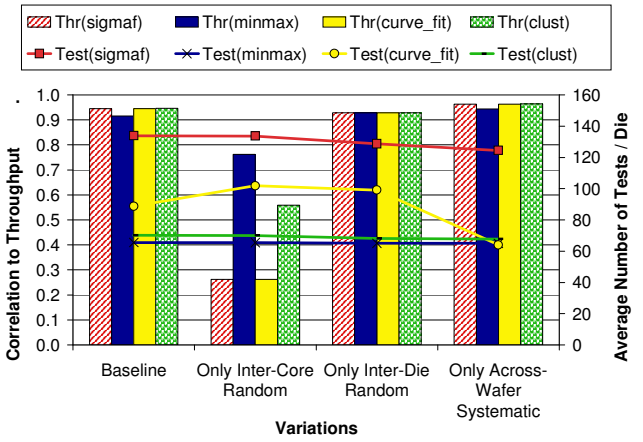
B. Dependence on Number of Cores

Figure 4 shows how correlation and binning overhead change with the number of cores on the processor dice. The results are shown for 16 frequency bins. There are several things to note from these graphs.

- For multi-programmed workloads, the correlation to throughput increases with the number of cores for both clustering-based strategies. Better correlation with more cores is a result of having a fixed cluster size, which results in a larger number of clusters per chip (note that with more clusters, the granularity of clustering becomes finer). To confirm this, we also performed experiments to see how the correlation and binning overhead change



(a) multi-programmed



(b) multi-threaded

Fig. 7. Correlation of various binning metrics to actual throughput and their binning overhead for (a), multi-programmed benchmarks and, (b) multi-threaded benchmarks, for different process variation scenarios.

when the number of cores per cluster (and, therefore, the number of clusters) is changed for a fixed sized chip (with 64 cores). Figure 5 shows the results. We indeed observe that the binning overhead of clustering decreases with increasing number of cores per cluster. Similarly, the correlation to throughput decreases for multi-programmed workloads with increasing cores per clusters.

- Interestingly, the roles of the metrics are reversed for multi-programmed and multi-threaded workloads. While Σf and curve fitting do well for multi-programmed workloads, min-max and clustering do better for multi-threaded workloads. This reversal can be explained by the fact that Σf and curve fitting (a close approximation) characterize the maximum throughput of a die, which is strongly correlated to performance for multi-programmed workloads. However, when workload performance correlates more strongly to the performance of the weakest core, min-max wins out. Since clustering uses the min-max metric as its backbone, trends for clustering are similar to those for min-max.
- As the number of cores per cluster increases, we see an interesting difference between the two types of clustering for multi-threaded benchmarks. *For clustering that bounds the search range based on perceived variation (smart clust), throughput correlation levels off and begins to decrease as the number of cores per cluster becomes large*. This is because the limited search range may not be wide enough to capture the variation range in a large cluster. However, when the entire search range is considered, correlation continues to increase even as the number of cores per cluster increases. This is because performance is correlated to the performance of the slowest core on the die for our multi-threaded benchmarks, and larger clusters result in less over-estimation of performance for a processor running such benchmarks.

C. Dependence on Search Range for Variation-Model Aware Approaches

Figure 6 shows how performance correlation and binning overhead change as the search range is varied for 8 and 64 frequency bins (we only show the results for multi-programmed workloads as multi-threaded benchmarks behave similarly). Both techniques that rely on the variation model to come up with aggressive search ranges (*curve_fit* and *smart clust*) have better correlation as the search range is increased. The improvement is higher for larger number of frequency bins. For example, when moving from 2σ to 3σ, correlation to throughput for curve fitting improves by 30% for 64 bins but just by 6% for 8 bins. However, the increase in binning overhead is also higher for a larger number of bins. *Therefore, unless the variation is large enough to justify an increase in the bin count, fixed search range of* 2σ *or* 3σ *is good enough.*

D. Dependence on Nature of Variations

In Figure 7, we show the effect that the nature of variations has on binning metrics and their evaluation. The four cases: *baseline* (incorporates all variation model components), *only inter-core random*, *only inter-die random*, and *only across-wafer systematic* (i.e., the bowl-shaped variation) all have the same variance. As within-die (i.e. core-to-core) variation increases, the correlation of *min-max* to the throughput of multi-programmed workloads decreases, since it grossly underestimates throughput (because it takes the minimum f_{max} of all cores). However, for multi-threaded workloads, Σf shows poor performance correlation when inter-core variation dominates, since it overestimates the throughput of the processor. *Therefore, increase in random core to core variation magnifies the difference between the two metrics with the workload types. This implies that in such a variation scenario, choice of metric will strongly depend on the expected workload type*. Note that variation-aware binning strategies that use the variation model for prediction (i.e., curve fitting) achieve maximum reduction of binning overhead in cases where there is systematic variation (*baseline* and *only across-wafer systematic*).

VII. Conclusion

In this paper, we have studied for the first time, speed binning for multi-core processors. We have compared two intuitive metrics – *min-max* and Σf – in terms of their correlation to actual throughput for various kinds of workloads as well as their testing overheads. Furthermore, we have proposed binning strategies which leverage the extent of variation (clustering) as well as the partially systematic nature of variation (curve fitting). From our analysis, we conclude the following

- In terms of correlation to actual throughput, Σf is an overall better metric except for two cases where *min-max* performs well: 1) multi-threaded benchmarks, with large number of bins (larger than 8) and, 2) multi-threaded benchmarks when within-die variations are dominant. However, *min-max* has a significantly lower binning overhead than Σf (lower by as much as 70%).
- Clustering based strategies which are a hybrid of Σf and *min-max* reduce the binning overhead by as much as 51% with a small loss (5% points for 8 bins) in correlation to throughput.
- Variation-model aware strategies help in reducing the binning overhead significantly with the same correlation to throughput as Σf. Variation aware curve fitting reduces the binning overhead by as much as 36%.

Our overall conclusion is that uniprocessor binning methods do not scale well for multi-core processors in the presence of variations. Multi-core binning metrics and testing strategies should be carefully chosen to strike a good balance between goodness of the metric and time required to evaluate it. Most importantly, the efficiency of speed binning can be improved significantly by leveraging process variation knowledge to optimize the binning procedure.

In some cases, power and memory/cache size are also important binning metrics. For low power embedded applications where power is an equally important metric as performance, the same notion of binning can be employed to categorize processors. The variation model can be used to bin processors based on power dissipation. The concept of voltage binning [28] [29] can be extended for multicore processors by making use of similar techniques as suggested in this paper. This is part of our ongoing work on efficient characterization of multicore processors.

VIII. **Acknowledgement**

We would like to thank Dr. Lerong Cheng for discussions on the variability model. Work at UIUC was supported in part by Intel, NSF, GSRC, and an Arnold O Beckman Research Award. Work at UCLA was partly supported by NSF.

References

[1] Girard, P., "Survey of low-power testing of VLSI circuits", *Design & Test of Computers, IEEE*, 2002.

[2] Y. Bonhomme, P. Girard, C. Landrault and S. Pravossoudovitch, "Test Power: a Big Issue in Large SOC Designs", *Electronic Design, Test and Applications, IEEE International Workshop on*, 2002.

[3] Nicolici, Nicola, Al-Hashimi and Bashir M., "Power-Constrained Testing of VLSI Circuits", *Series: Frontiers in Electronic Testing*, 2003.

[4] L. Cheng, P. Gupta, K. Qian, C. Spanos, and L. He, "Physically Justifiable Die-Level Modeling of Spatial Variation in View of Systematic Across Wafer Variability", *IEEE/ACM DAC*, 2009.

[5] B.D. Cory, R. Kapur and B. Underwood, "Speed Binning with Path Delay Test in 150nm Technology", *IEEE Design & Test of Computers*, 2003, pp. 41-45.

[6] E. Humenay, D. Tarjan and K. Skadron, "Impact of Process Variations on Multicore Performance Symmetry", *Proc. IEEE/ACM DATE*, 2007, pp. 1653-1658.

[7] S. McFarling, "Combining branch predictors", *Technical Report TN-36m, Digital Western Research Laboratory*, June 1993.

[8] D. Belete, A. Razdan, W. Schwarz, R. Raina, C. Hawkins and J. Morehead, "Use of DFT Techniques in Speed Grading a 1GHz+ Microprocessor", *Proc. IEEE ITC*, 2002, pp. 1111-1118.

[9] J. Zeng, M. Abadir, G. Vandling, L. Wang, A. Kolhatkar and J. Abraham, "On Correlating Structural Tests with Functional Tests for Speed Binning of High Performance Design", *Proc. IEEE ITC*, 2004, pp. 31-37.

[10] S. Borkar, "Design Challenges of Technology Scaling", *IEEE Micro*, 1999.

[11] K. Qian and C.J. Spanos, "A Comprehensive Model of Process Variability for Statistical Timing Optimization", *Proc. SPIE Design for Manufacturability through Design-Process Integration*, 2008.

[12] P. Friedberg, W. Cheung and C.J. Spanos, "Spatial Modeling of Micron-Scale Gate Length Variation", *Proc. SPIE Data Analysis and Modeling for Process Control*, 2006.

[13] B.E. Stine, D.S. Boning and J.E. Chung, "Analysis and Decomposition of Spatial Variation in Integrated Circuit Processes and Devices", *IEEE Trans. Semiconductor Manufacturing*, 10(1), 1997.

[14] J. Xiong, V. Zolotov and L. He, "Robust Extraction of Spatial Correlation", *Proc. ACM ISPD*, 2006.

[15] F. Liu, "A General Framework for Spatial Correlation Modeling in VLSI Design", *Proc. IEEE/ACM DAC*, 2007.

[16] ITRS, 2007, http://public.itrs.net.

[17] Asanovic, Krste and Bodik, Ras and Catanzaro, Bryan Christopher and Gebis, Joseph James and Husbands, Parry and Keutzer, Kurt and Patterson, David A. and Plishker, William Lester and Shalf, John and Williams, Samuel Webb and Yelick, Katherine A., "The Landscape of Parallel Computing Research: A View from Berkeley", EECS Department, University of California, Berkeley, 2006.

[18] S. J. E. Wilton and N. P. Jouppi, "CACTI: an enhanced cache access and cycle time model," *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 677–688, May 1996.

[19] Rakesh Kumar and Dean M. Tullsen, "Core architecture optimization for heterogeneous chip multiprocessors", *International Conference on Parallel Architectures and Compilation Techniques, PACT*, 2006.

[20] Timothy Sherwood and Erez Perelman and Greg Hamerly and Brad Calder, "Automatically Characterizing Large Scale Program Behavior", ASPLOS, 2002.

[21] D.M. Tullsen, "Simulation and Modeling of a Simultaneous Multithreading Processor", 1996, *22nd Annual Computer Measurement Group Conference*.

[22] J. Dorsey, et al., "An Integrated Quad-core Opteron Processor", *ISSCC 07*, 2007.

[23] Herbert, S. and Marculescu, D. "Analysis of Dynamic Voltage/Frequency Scaling in Chip-Multiprocessors", *ISLPED '07*. 2007.

[24] C. Isci, et al, "An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget", *MICRO*, 2006.

[25] Juang, et al., "Coordinated, Distributed, Formal Energy Management of Chip Multiprocessors", *ISLPED '05*, 2005.

[26] J. Sartori and R. Kumar, "Distributed Peak Power Management for Many-core Architectures", *DATE '09*, 2009.

[27] J. Sartori and R. Kumar, "Three Scalable Approaches to Improving Many-core Throughput for a Given Peak Power Budget", *HiPC '09*, 2009.

[28] J. Tschanz, K. Bowman, and V. De, "Variation-tolerant circuits: circuit solutions and techniques", *DAC ACM*, 2005.

[29] Paul, S., Krishnamurthy, S., Mahmoodi, H., and Bhunia, S, "Low-overhead design technique for calibration of maximum frequency at multiple operating points", *ICCAD*, 2007.