Illustrative Design Space Studies with Microarchitectural Regression Models

Benjamin C. Lee and David M. Brooks
Division of Engineering and Applied Sciences
Harvard University
Cambridge, Massachusetts
{bclee, dbrooks}@eecs.harvard.edu

Abstract

We apply a scalable approach for practical, comprehensive design space evaluation and optimization. This approach combines design space sampling and statistical inference to identify trends from a sparse simulation of the space. The computational efficiency of sampling and inference enables new capabilities in design space exploration. We illustrate these capabilities using performance and power models for three studies of a 260,000 point design space: (1) pareto frontier analysis, (2) pipeline depth analysis, and (3) multiprocessor heterogeneity analysis. For each study, we provide an assessment of predictive error and sensitivity of observed trends to such error.

We construct pareto frontiers and find predictions for pareto optima are no less accurate than those for the broader design space. We reproduce and enhance prior pipeline depth studies, demonstrating constrained sensitivity studies may not generalize when many other design parameters are held at constant values. Lastly, we identify efficient heterogeneous core designs by clustering per benchmark optimal architectures. Collectively, these studies motivate the application of techniques in statistical inference for more effective use of modern simulator infrastructure.

1 Introduction

Microarchitectural design space exploration is often inefficient and ad hoc due to the significant computational costs of current simulator infrastructure. While simulators provide insight into application performance for a broad range of microarchitectural designs, the inherent costs of modeling microprocessor execution result in long simulation times and, in trace-driven simulators, non-trivial storage costs. Designers circumvent these challenges by constraining the design space considered (often using intuition or experience) and/or reducing the size of simulator inputs via trace sampling. However, by pruning the design space with intuition before a study, the designer risks obtaining conclusions that simply reinforce prior intuition and may not generalize to the broader space. Trace sampling, while effective in reducing the simulator input size by orders of magnitude, only impacts per simulation costs and does not address the number of simulations required in a comprehensive design space study. Trace sampling alone is insufficient as per simulations costs decrease linearly, albeit by a large factor, while the number of potential simulation points increase exponentially with the number of design parameters. This exponential increase is currently driven by the design of multicore, multi-threaded microprocessors targeting several different metrics including single-thread latency, throughput for emerging parallel workloads, and energy. These trends will also lead to more variety in the set of viable and interesting designs (*e.g.*, simpler, less aggressive cores), thereby requiring a more thorough exploration of a comprehensive design space.

Techniques in statistical inference are necessary for a scalable simulation approach that addresses these fundamental challenges, modestly reducing detail for substantial gains in speed and tractability. Even for applications in which obtaining extensive measurement data is feasible, efficient analysis of this data often lends itself to statistical modeling. Such an approach typically requires an initial data set for model formulation or training. The model responds to predictive queries by leveraging correlations in the original data for inference. Regression follows this predictive paradigm in a relatively cost effective manner, formulating models from observed data by numerically solving a system of linear equations. Predictions are obtained by evaluating a linear system. Well optimized numerical linear algebra libraries lead to computationally efficient models, enabling thousands of predictions in a few seconds.

Design space sampling and statistical inference enables the designer to (1) perform a tractable number of simulations independent of design space size or resolution and (2) use simulator data efficiently by inferring trends without explicit and exhaustive simulation. To achieve the first objective, we sample points uniformly at random from the design space for simulation (Section 2). Prior work has found 1,000 samples sufficient for a space with 1 billion designs and we similarly obtain 1,000 samples from a space of 375,000 designs [14]. To achieve the second objective given these samples, we formulate non-linear regression models for microarchitectural performance and power prediction (Section 3), achieving median error rates of 7.2 and 5.4 percent relative to simulation. Given their accuracy, we apply regression models to comprehensively explore a design space for three optimization problems:

1. **Pareto Frontier Analysis:** We comprehensively characterize the design space, constructing a regression predicted pareto frontier in the power-delay space. We find predictions for pareto optima are as accurate as those for the broader space (Section 4).

2. **Pipeline Depth Analysis:** We compare a constrained pipeline depth study against an enhanced study that varies all parameters simultaneously via regression modeling. We find constrained sensitivity studies may not generalize when many other design parameters are held at constant values (Section 5).

3. **Multiprocessor Heterogeneity Analysis:** We identify efficiency maximizing architectures for each benchmark via regression modeling and cluster these architectures to identify design compromises. We quantify the power-performance benefits from varying degrees of core heterogeneity, quantifying a theoretical upper bound on $bips^3/w$ efficiency gains. We find modest heterogeneity may provide substantial efficiency benefits relative to homogeneity (Section 6).

For each case study, we provide an assessment of predictive error and sensitivity of observed trends to such error. Collectively these studies demonstrate the applicability of regression models for performance and power prediction in practical design space optimization.

2 Experimental Methodology

2.1 Simulation Framework

We use Turandot, a generic and parameterized, out-of-order, superscalar processor simulator [16]. Turandot is enhanced with PowerTimer to obtain power estimates based on circuit-level power analyses and resource utilization statistics [1]. The modeled baseline architecture is similar to the current POWER4/POWER5. The simulator has been validated against both a POWER4 RTL model and a hardware implementation. Power scales superlinearly as pipeline width increases, using scaling factors derived for an architecture with clustered functional units [25]. Cache power and latencies scale with array size according to CACTI [21]. We do not leverage any particular feature of the simulator and our framework may be generally applied to other simulation frameworks with similar accuracy. We evaluate performance in billions of instructions per second (bips) and power in watts (w).

	Set	Parameters	Measure	Range	$\|S_i\|$
S_1	Depth	depth	FO4	9::3::36	10
S_2	Width	width	decode b/w	2,4,8	3
		L/S queue	entries	15::15::45	
		store queue	entries	14::14::42	
		functional units	count	1,2,4	
S_3	Physical Registers	general purpose	count	40::10::130	10
		floating-point	count	40::8::112	
		special purpose	count	42::6::96	
S_4	Reservation Stations	branch	entries	6::1::15	10
		fixed-point	entries	10::2::28	
		floating-point	entries	5::1::14	
S_5	I-L1 Cache	i-L1 cache size	KB	16::2x::256	5
S_6	D-L1 Cache	d-L1 sache size	KB	8::2x::128	5
S_7	L2 Cache	L2 cache size	MB	0.25::2x::4	5

Table 1. Design space; i::j::k denotes values from i to k in steps of j.

We use R, an open-source software environment for statistical computing, to script and automate statistical analyses [23]. Within this environment, we use the Hmisc and Design packages implemented by Harrell [7].

2.2 Benchmark Suite

We consider SPECjbb, a Java server benchmark, and eight compute intensive benchmarks from SPEC2000 (ammp, applu, equake, gcc, gzip, mcf, mesa, twolf). We report experimental results based on PowerPC traces of these benchmarks. The SPEC2k traces used in this study were sampled from the full reference input set to obtain 100 million instructions per benchmark program. Systematic validation was performed to compare the sampled traces against the full traces to ensure accurate representation [11]. Our benchmark suite is representative of larger suites frequently used in the microarchitectural research community [18]. Although specific conclusions of our design space studies may differ with different benchmarks, we do not leverage any particular benchmark feature in model formulation and our framework may be generally applied to other workloads with similar accuracy.

2.3 Design Space Sampling

The approach for obtaining observations from a large microarchitectural design space is critical to efficient formulation of regression models. Table 1 identifies seven groups of parameters varied simultaneously. The range of values considered are specified by sets, $S_1, \ldots, S_7$. The Cartesian product of these sets, $S = \prod_{i=1}^{7} S_i$, defines the design space that contains $|S| = \prod_{i=1}^{7} |S_i| = 375,000$ points. Models are formulated with $n = 1,000$ samples from the space and each sampled design is simulated for all workloads in the benchmark suite.

Techniques that sweep design parameter values to consider all design points in S is impractical despite continuing research to reduce per simulation costs. In contrast to prior research that emphasizes trace sampling [20, 24], we sample *uniformly at random* (UAR) from the design space S to control the exponentially increasing number of design points as parameter count and resolution increases [14]. This approach provides observations from the full range of parameter values and enables identification of trade-offs between parameter sets. An arbitrarily large number of values may be included in a set S_i, thereby achieving greater parameter space resolution, since the number of simulations is decoupled from set cardinality via random sampling. While design space studies that consider points around a baseline configuration may be biased toward the baseline, sampling UAR provides unbiased observations.

3 Regression Modeling

We build on our prior work that derived regression models for the microarchitectural design space and validated for randomly selected designs [14, 15]. This statistically robust derivation applied statistical analyses including variable clustering, association and correlation analysis, residual analysis, and significance testing. We further this prior work by applying performance and power regression models to practical design space optimization.

3.1 Model Formulation

For a large universe of interest, suppose we have a subset of n observations for which values of the response and predictor variables are known. Let $\vec{y} = y_1, \ldots, y_n$ denote observed responses. For a particular point i in this universe, let y_i denote its response and $\vec{x_i} = x_{i,1}, \ldots, x_{i,p}$ denote its p predictors. Let $\vec{\beta} = \beta_0, \ldots, \beta_p$ denote regression coefficients used in describing the response as a linear function of predictors plus a random error e_i as shown in Equation (1). Transformations f and $\vec{g} = g_1, \ldots, g_p$ may be applied to the response and predictors, respectively, to improve model fit. We fit a regression model to observations by determining $\vec{\beta}$ with the *method of least squares*.

$$f(y_i) = \beta_0 + \sum_{j=1}^{p} \beta_j g_j(x_{ij}) + e_i \qquad (1)$$

In the context of microprocessor design, the response y represents a metric of interest (*e.g.*, performance or power) and the predictors x represent design parameter values (*e.g.*, pipeline depth or L2 cache size).

3.2 Predictor Interaction

In some cases, the effect of two predictors $x_{i,1}$ and $x_{i,2}$ on the response cannot be separated; the effect of $x_{i,1}$ on y_i depends on the value of $x_{i,2}$ and vice versa. The interaction between two predictors may be modeled by constructing a third predictor $x_{i,3} = x_{i,1}x_{i,2}$ to obtain $y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \beta_3 x_{i,1} x_{i,2} + e_i$. We draw on domain-specific knowledge to specify such interactions. Pipeline depth likely interacts with cache sizes. As the L2 cache size decreases, memory stalls per instruction will increase and instruction throughput gains from pipelining will be constrained. Pipeline width is expected to interact with register file and queue sizes. We also specify interactions between sizes of adjacent cache levels in the memory hierarchy (*e.g.*, L1 and L2 cache size interaction).

3.3 Non-Linearity

Linearity assumptions are often too restrictive as non-linear transformations may reduce error and capture non-linear effects. A square-root transformation on the response ($f(y_i) = \sqrt{y}$) is particularly effective for reducing error variance in our performance models. Similarly, a log transformation ($f(y_i) = log(y)$) more effectively captures exponential trends in our power model. We also consider *restricted cubic splines* on the predictors. Splines divide the predictor domain into intervals with endpoints called *knots* and different cubic polynomials are fit to observations within each interval to obtain a piecewise cubic polynomial. Cubic splines have several advantages over simpler polynomial transformations and lower order splines [14].

The position and number of knots are tunable when specifying non-linearity with splines. Knots at fixed quantiles of a predictor's distribution ensure a sufficient number of points in each interval and is effective in most datasets [22]. As the number of knots increases, flexibility improves at the risk of over-fitting the data. The strength of a predictor's correlation with the response will determine the number of knots in the transformation. A lack of fit for predictors highly correlated with the response will have a greater negative impact on accuracy. For example, predictors with stronger performance relationships will use 4 knots (*e.g.*, pipeline depth and register file size) and those with weaker relationships will use 3 knots (*e.g.*, latencies, cache sizes, reservation stations).

3.4 Prediction

Figure 1 presents boxplots of the error distributions from performance and power predictions of 100 validation points

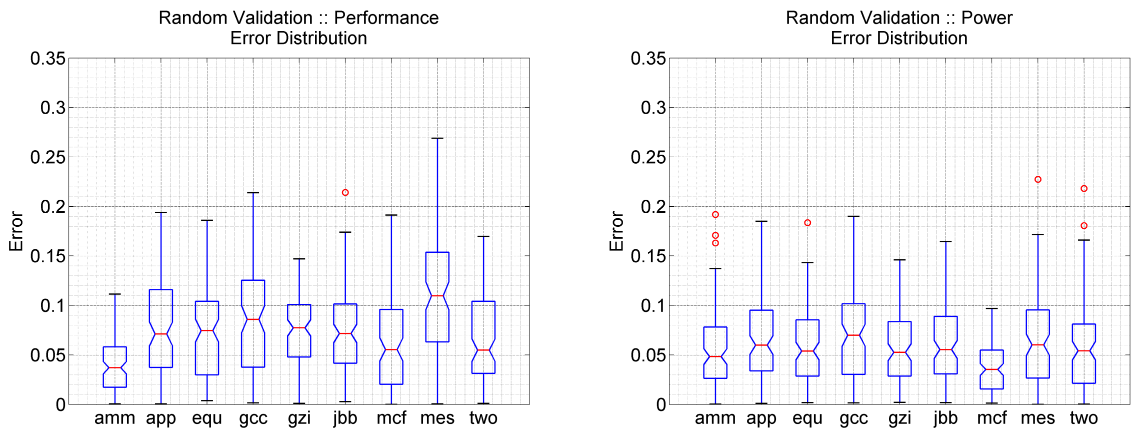


Figure 1. Distribution of prediction errors for 100 random validation designs.

sampled UAR from the design space. The error is expressed as $|obs - pred|/pred$. Boxplots are graphical displays of data that measure location (median) and dispersion (interquartile range), identify possible outliers, and indicate the symmetry or skewness of the distribution. Boxplots are constructed by

1. horizontal lines at median and upper, lower quartiles
2. vertical lines drawn up/down from upper/lower quartile to most extreme data point within 1.5 of the IQR (interquartile range - the difference between first and third quartile) of the upper/lower quartile with short horizontal lines to mark the end of the vertical lines
3. circles denote outliers

Figure 1 indicates the performance model achieves median errors ranging from 3.7 percent (ammp) to 11.0 percent (mesa) with an overall median error across all benchmarks of 7.2 percent. Power models are slightly more accurate with median errors ranging from 3.5 percent (mcf) to 7 percent (gcc) and an overall median of 5.4 percent. Although such model validation is statistically representative, applications of regression modeling will likely predict metrics within a structured, coherent design space study.

3.5 Design Space Studies

Given the accuracy of regression models, we present applications of performance and power regression modeling to three representative design space studies:

- **Pareto Frontier Analysis:** Comprehensively characterize the design space, constructing a regression predicted pareto frontier in the power-delay space.
- **Pipeline Depth Analysis:** Combine regression and the framework of prior pipeline depth studies to identify $bips^3/w$ maximizing depths. Enhance prior studies by varying all design parameters simultaneously instead of fixing most non-depth parameters.
- **Multiprocessor Heterogeneity Analysis:** Identify $bips^3/w$ maximizing architectures for each benchmark via regression. Cluster these architectures to identify compromise designs and power-performance benefits from varying degrees of core heterogeneity.

We explore a design space of 262,500 points ranging that includes depths from 12 to 30 FO4. We formulate the models using samples from the design space of Table 1. The design space for sampling and model formulation should be larger than the space for exploration to mitigate errors from extrapolation and we increase the sample space to include 9, 33, and 36 FO4 designs as well. For each case study, we provide an assessment of predictive error and sensitivity of observed trends to such error. Collectively, these studies demonstrate the applicability of regression models for performance and power prediction within practical design space optimization problems.

4 Pareto Frontier Analysis

Pareto optimality is an economic concept with broad applications to engineering. Given a set of design parameters and a set of design metrics, a pareto optimization changes the parameters to improve at least one metric without negatively impacting any other metric. A design is pareto optimal when no further pareto optimizations can be implemented. For the microarchitectural design space, pareto optima are designs that minimize delay for a given level of power consumption. A pareto frontier is defined by a set of delay minimizing optima across a range of power budgets.

Regression models enable a complete characterization of the microarchitectural design space. We leverage the com-

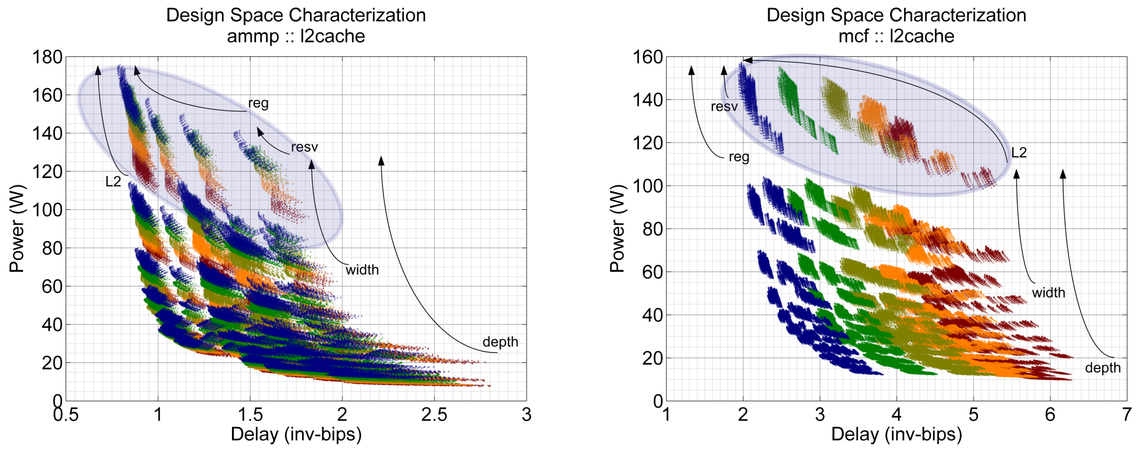


Figure 2. Regression predicted delay, power of all designs for representative benchmarks. Arrows indicate trends as particular resource sizes increase. Colors map to L2 cache sizes.

putational efficiency of regression to perform an exhaustive evaluation of the design space containing more than 260,000 points, requiring fewer than four hours per benchmark.[1] Such a characterization reveals all trade-offs between a large number of design parameters simultaneously compared to an approach that relies on per parameter sensitivity analyses. Given this characterization, we construct pareto frontiers. While we cannot explicitly validate the regression identified pareto frontier against a hypothetical frontier found by exhaustive simulation, the former is likely close to the latter given the accuracy observed in validation.

4.1 Design Space Characterization

Figure 2 plots the predicted delay (inverse throughput) and power of the design space by exhaustively evaluating the regression models for representative benchmarks. The design space is characterized by several overlapping clusters of similar designs. Each cluster contains designs with a particular pipeline depth-width combination. For example, the shaded mcf cluster with delay ranging from 1.9 to 5.3 seconds and power ranging from 100 to 160 watts delivers the lowest delay at the greatest power with depth of 12FO4 and decode bandwidth of 8 instructions per cycle.

The arrows of Figure 2 identify power-delay trends as a particular resource size increases. Consider the shaded 12FO4, 8-wide design clusters for ammp and mcf. Mcf experiences substantial performance benefits from larger caches with delay shifting from 5.3 to 1.9 seconds as L2 cache size shifts from 0.25 to 4MB. In contrast, ammp sees increasing power costs with limited performance benefits of 1.0 to 0.8 seconds as L2 cache size increases by the same amount. Ammp also appears to exhibit greater instruction level parallelism, effectively utilizing additional physical registers and reservation stations to reduce delay from approximately 1.8 to 0.8 seconds compared to mcf's reduction of 2.5 to 2.0 seconds.

4.2 Pareto Optima Identification

Given a design space characterization, Figure 3 plots regression predicted pareto optima. These optima minimize delay for a given power budget. Given regression models and exhaustively predicted power and delay characteristics, the frontier is constructed by discretizing the range of delays and identifying the design that minimizes power for each delay in a number of delay targets. These designs are pareto optimal with respect to the regression models, but may not be the same optima obtained via a hypothetical exhaustive simulation of the space.

Although pareto optima may be preferred for particular delay or power targets, not all pareto optima are power-performance efficient with respect to $bips^3/w$, the inverse energy delay-squared product.[2] We compute the efficiency metric for each design on the pareto frontier and identify the most efficient designs in Table 2. The $bips^3/w$ optimal design for ammp is located at 1.0 seconds and 35.9 watts in the delay-power space, the knee of the pareto optimal curve. Similarly, the mcf $bips^3/w$ optimal design is located at 3.5 seconds and 12.9 watts. Overall, these optima are drawn from diverse regions of the design space motivating comprehensive space exploration.

[1]Based on wall clock time of 15 seconds for 800 predictions on 1.8 GHz Pentium M extrapolated to more powerful compute clusters and optimized numerical linear algebra libraries.

[2]$bips^3/w$ is a voltage invariant power-performance metric derived from the cubic relationship between power and voltage [2].

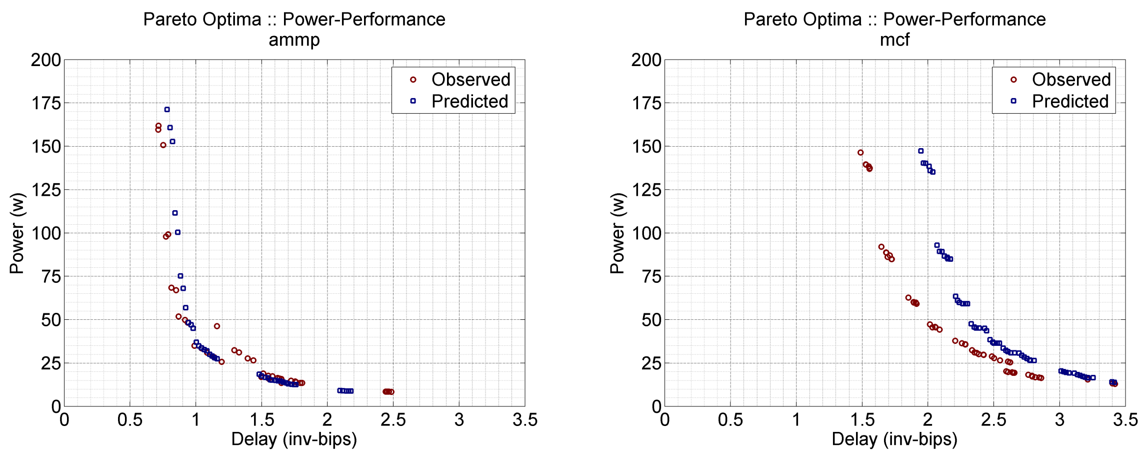


Figure 3. Modeled and simulated pareto optima for representative benchmarks.

	Depth	Width	Reg	Resv	I-$ (KB)	D-$ (KB)	L2-$ (MB)	Delay Model	Error	Power Model	Error
ammp	27	8	130	12	32	128	2	1.0	0.2%	35.9	-3.9%
applu	27	8	130	15	16	8	0.25	0.8	-0.8%	39.6	0.1%
equake	27	8	130	15	64	8	0.25	1.2	-0.8%	41.5	-3.0%
gcc	15	2	70	9	16	8	1	1.2	5.2%	44.1	-6.0%
gzip	15	2	70	6	16	8	0.25	0.8	8.8%	24.2	0.0%
jbb	15	8	80	12	16	128	1	0.6	-4.7%	80.9	1.6%
mcf	30	2	70	6	256	8	4	3.5	2.4%	12.9	-3.0%
mesa	15	8	80	13	256	32	0.25	0.4	5.2%	86.9	-7.1%
twolf	27	8	130	15	128	128	2	1.1	-1.2%	34.5	-3.6%

Table 2. $bips^3/w$ maximizing per benchmark architectures.

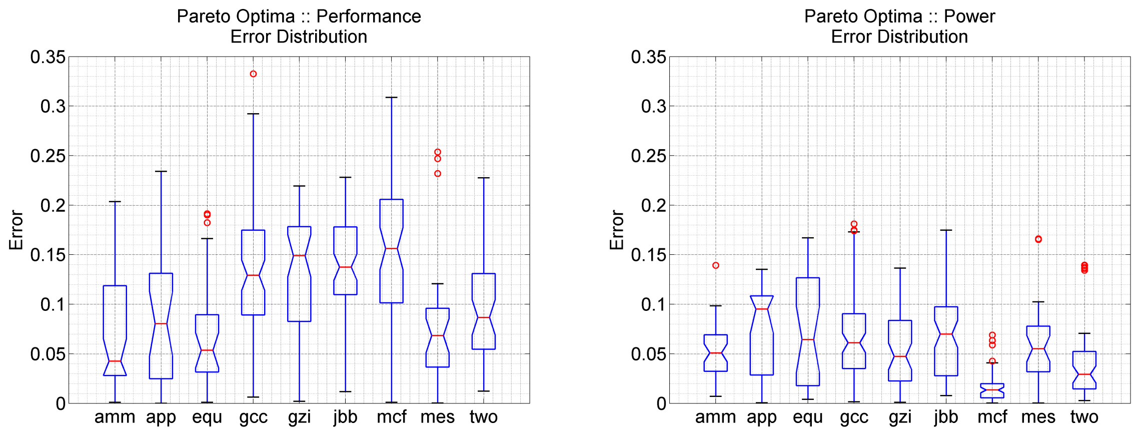


Figure 4. Distribution of prediction errors for pareto frontier.

4.3 Pareto Optima Validation

Figure 3 superimposes simulated and predicted pareto frontiers, suggesting good relative accuracy. Regression effectively captures the delay-power trends of the pareto frontier. As performance prediction is less accurate than power prediction, however, differences between are characterized by horizontal shifts in delay. Performance model accuracy is the limiting factor for more accurate pareto frontier prediction across all benchmarks in our suite.

Figure 4 presents the error distributions for the performance and power prediction of points on the pareto frontier. The median performance error ranges from 4.3 percent (ammp) to 15.6 percent (mcf) with an overall median of 8.7 percent. Similarly, the median power error ranges from 1.4 percent (mcf) to 9.5 percent (applu) with an overall median of 5.5 percent. These error rates are consistent with the performance and power median error rates of 7.2 and 5.4 percent observed in the validation of random designs (Figure 1), suggesting predictions for pareto optima are no less accurate than those for the overall design space. As shown in Table 2, errors associated with $bips^3/w$ optimal predictions are also consistent with those for the broader space. Delay errors range from 0.2 to 8.8 percent while power errors range from 0.1 to 7.1 percent.

5 Pipeline Depth Analysis

Prior pipeline studies considered various depths while holding most other design parameters at constant values, in part, to control the simulation costs of varying multiple parameters simultaneously [8, 9, 26]. Thus constraining the space may lead to narrowly defined studies with conclusions that may not generalize. Regression models enable a more complete characterization of pipeline depth trends by allowing other design parameters to vary simultaneously. A more comprehensive depth analysis ensures observed trends are not an artifact of the constant baseline values to which other parameters are held.

Pipeline depth is specified by the number of fan-out-of-four (FO4) inverter delays per pipeline stage.[3] When logic and latch overhead per pipeline stage is measured in terms of FO4 delay, deeper pipelines have smaller FO4 delays. We consider pipeline depths ranging from 12 to 30FO4 to compare and contrast the following approaches:

- **Original Analysis:** Consider the POWER4-like baseline architecture of Table 3, predicting power-performance efficiency as depth varies and all other design parameters are held constant at baseline values.

[3]FO4 delay is defined as the delay of one inverter driving four copies of an equally sized inverter.

Processor Core	
Decode Rate	4 non-branch insns/cy
Dispatch Rate	9 insns/cy
Reservation Stations	FXU(40),FPU(10),LSU(36),BR(12)
Functional Units	2 FXU, 2 FPU, 2 LSU, 2 BR
Physical Registers	80 GPR, 72 FPR
Branch Predictor	16k 1-bit entry BHT
Memory Hierarchy	
L1 DCache Size	32KB, 2-way, 128B blocks, 1-cy lat
L1 ICache Size	64KB, 1-way, 128B blocks, 1-cy lat
L2 Cache Size	2MB, 4-way, 128B blocks, 9-cy lat
Memory	77-cy lat
Pipeline Dimensions	
Pipeline Depth	19 FO4 delays per stage
Pipeline Width	4-decode

Table 3. Baseline Architecture

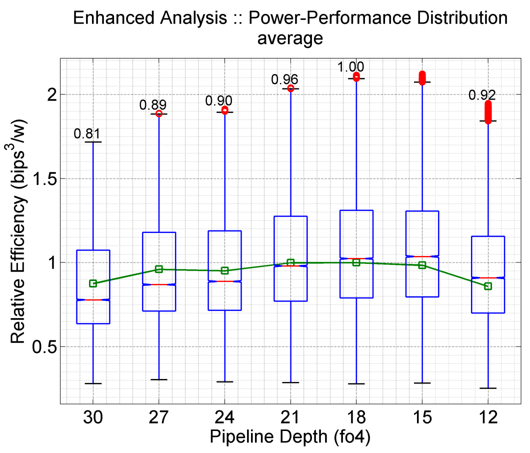
- **Enhanced Analysis:** Consider the design space of Table 1, predicting efficiency as parameters vary simultaneously.

5.1 Pipeline Depth Trends

The line plot of Figure 5(a) presents predicted efficiency relative to the $bips^3/w$ maximizing baseline design in the constrained original analysis. 18 FO4 delays per stage is optimal for an average of the benchmark suite. Although choosing the deepest or shallowest pipeline will achieve only 85.9 or 87.6 percent of the optimal efficiency, respectively, the models suggest a plateau around the optimum and not a sharp peak. The superimposed boxplots of Figure 5(a) show the efficiency distribution of the 37,500 designs for each pipeline depth in the enhanced analysis. By graphically presenting efficiency quartiles, the boxplot for 18 FO4 designs indicate 75, 50, and 25 percent of these designs achieve efficiency of at least 79, 102, and 131 percent of the original $bips^3/w$ optimum.

The maxima of these boxplots constitute a potential bound on $bips^3/w$ efficiency achievable in this design space with up to 2.1x improvements at the optimal 18 FO4 pipeline depth. These bounding architectures are characterized by wide pipelines as well as larger queue and register file sizings. The efficiency of wide pipelines are likely a result of the energy-efficient functional unit clustering modeled by the simulator, which enables near linear power increases as width increases [19, 25]. However, our power models also account for superlinear width power scaling for structures such as the multi-ported register file, memory units, rename table, and forwarding logic [25]. Larger queue and reservation resources result from deeper pipelines and more instructions in flight.

The points at which the line plot intersect the boxplots indicate unexploited efficiency. Intersection at a lower point in the boxplot indicates a larger number of configurations are predicted more efficient than baseline at a particular



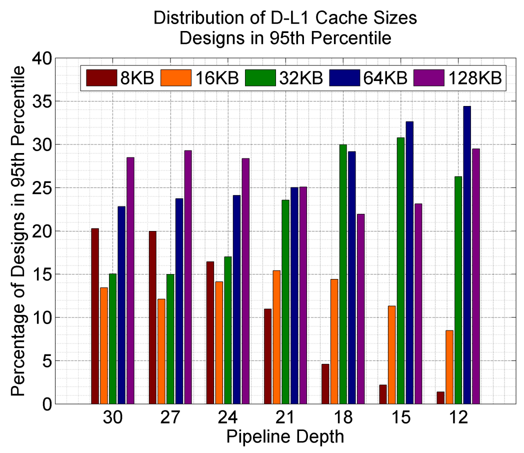


Figure 5. (a) Efficiency for original (line plot) and enhanced (boxplots) analyses relative to original $bips^3/w$ optimum. (b) Distribution of d-L1 cache sizes for designs in 95th percentile.

depth. More than 58 percent of 12 FO4 and 39 percent of 30 FO4 designs are predicted more efficient than baseline, corresponding to more than 21,000 and 14,000 designs, respectively. Such a large number of more efficient designs is not surprising, however, since the baseline resembles designs for server workloads with less emphasis on energy efficiency. Less efficient designs may be pruned from further study enabling more judicious use of detailed simulators should additional simulation be necessary.

Efficiency penalties for sub-optimal depths are also more significant for the bounding architectures. The $bips^3/w$ maximizing depth is 15-18 FO4 and the sub-optimal 30 FO4 design achieves 88 percent of the optimal efficiency, incurring a 12 percent efficiency penalty. The numbers above each boxplot in Figure 5(a) quantify each bound architecture's efficiency relative to that of the $bips^3/w$ maximizing bound architecture. While the bound architectures are also most efficient at 15 to 18 FO4, the sub-optimal 30 FO4 design achieves only 81 percent of the optimal efficiency and incurs a 19 percent penalty. This trend is observed for all depths shallower than the optimal 18 FO4. Since bound architectures are characterized by wider pipelines, choice of depth becomes more significant. For the average across our benchmark suite, wide pipelines with shallow depths will result in greater design imbalances and power-performance inefficiencies.

Figure 5(b) presents the distribution of data cache sizes in the most efficient designs at each depth. In particular, we take the 37,500 designs at each depth and consider designs in the 95-th percentile (*i.e.*, 1,875 designs in the top 5 percent of each depth's boxplot). Small 8KB data caches are observed for 20.3 percent of top designs at 30FO4 while such caches are optimal for only 1.4 percent of top designs at 12FO4. The percentage of top designs with larger 64KB

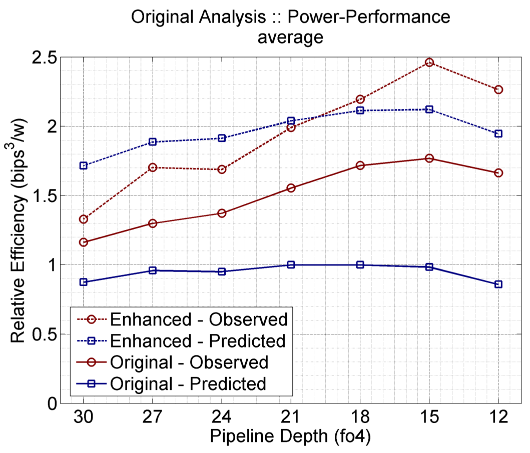
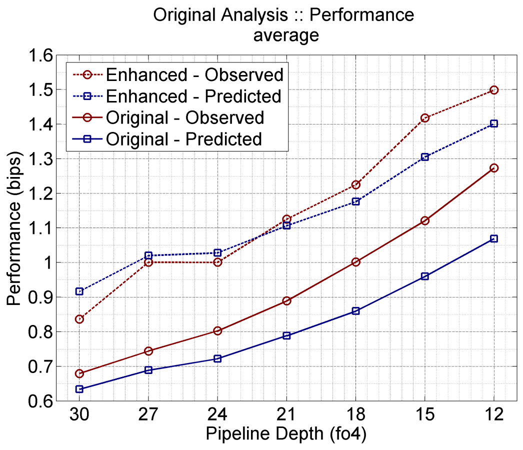


Figure 6. Predicted, simulated efficiency for original, enhanced analyses relative to original $bips^3/w$ optimum.

caches increases from 22.8 to 34.4 percent with deeper pipelines. Thus, smaller caches are increasingly viable at shallow pipelines while top designs often have larger caches at deep pipelines. This frequentist approach confirms our intuition that deeper pipelines favor larger caches to mitigate the increased costs of cache misses. This analysis also illustrates variability in the most efficient designs and the effect of parameter interactions on optimization.

5.2 Pipeline Depth Validation

Figure 6 validates the $bips^3/w$ predictions and suggests regression captures high-level trends in both analyses. The models correctly identify the most efficient depths to within 3 FO4 and capture the difference in efficiency penal-



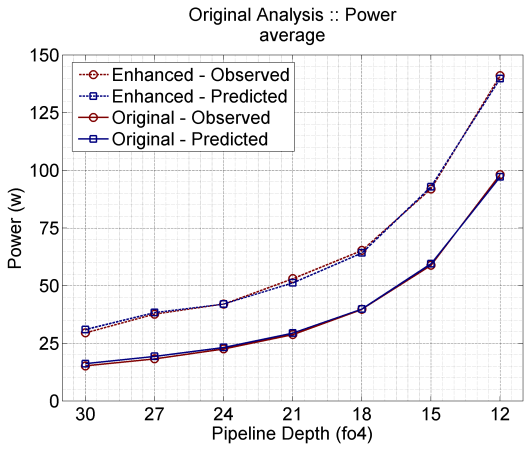


Figure 7. Predicted and simulated (a) performance, (b) power for original and enhanced analyses.

ties from sub-optimal depths between the two analyses. Whereas models predict 12 and 19 percent penalties, simulation identifies 52 and 67 percent penalties relative to 15 FO4 for the original and enhanced analyses, respectively. Thus, the significance of an optimal depth and penalties for sub-optimal designs are more pronounced in simulation.

Although the models are accurate for capturing high-level trends, $bips^3/w$ error rates are larger than those for performance and power. However, the $bips^3/w$ validation obscures underlying performance and power accuracy. By decomposing the validation of $bips^3/w$ in Figure 7, we find the underlying models exhibit good relative accuracy, effectively capturing performance and power trends. Since predictions from less accurate performance models must be cubed to compute $bips^3/w$, performance model errors are also cubed and negatively impact $bips^3/w$ accuracy. Countering these effects is continuing work.

6 Multiprocessor Heterogeneity Analysis

As shown in Table 2, regression models may be used to identify the $bips^3/w$ optimal architectures for each benchmark. In a uniprocessor or homogeneous multiprocessor design, the core is designed as an approximate compromise between these per benchmark optima to accommodate a range of workloads. Heterogeneous multiprocessor core design mitigates the efficiency penalties of this compromise [13]. However, prior work considered limited design spaces due to simulation costs. We combine regression modeling and clustering analyses to enable a more general exploration of core designs in heterogeneous architectures. This study identifies design compromises for the $bips^3/w$ design metric and quantifies a theoretical upper bound on the potential efficiency gains from high-performance heterogeneity, neglecting any associated multiprocessor overhead.

In particular, we combine our regression models with K-*means clustering*. A K-clustering of a set S is a partition of the set into K subsets which optimizes some clustering criterion, usually a similarity metric. Well defined clusters are such that all objects in a cluster are very similar and any two objects from distinct clusters are very dissimilar. General K-clustering is NP-hard and K-means clustering is a heuristic approximation.

6.1 Clustering Methodology

We first completely characterize the design space via regression to identify *benchmark architectures*, the $bips^3/w$ maximizing architectures for each benchmark in our suite (Table 2). These designs constitute the set to be partitioned into K subsets when clustering. The optimal design parameters exhibit significant spread across benchmarks with depth ranging from 15 to 30FO4, width ranging from 2 to 8 instructions decoded per cycle, and L2 caches ranging from 0.25 to 4MB. Each benchmark's execution characteristics are reflected in its optimal architecture. For example, compute-intensive gzip has the smallest L2 cache while memory-intensive mcf has the largest.

We perform K-means clustering for these nine benchmark architectures to identify *compromise architectures*. The heuristic for K clusters consists of the following:

1. Define K centroids, one for each cluster, and place randomly at initial locations in space containing objects to be clustered.

2. Assign each object to cluster with closest centroid.

3. When all objects have been assigned, recompute placement of K centroids such that its distance to objects in its cluster is minimized.

4. Since centroids may have moved in step 3, object assignment to clusters may change. Thus, steps 2 and 3 are repeated until centroid placement is stable.

Cluster	Depth	Width	Reg	Resv	I-$ (KB)	D-$ (KB)	L2-$ (MB)	Avg Delay Model	Avg Power Model	Benchmarks
1	15	8	80	12	64	64	0.5	2.26	82.17	jbb, mesa
2	27	8	130	14	32	32	0.5	1.05	32.53	ammp, applu, equake, twolf
3	15	2	70	8	16	8	0.5	0.93	37.55	gcc, gzip
4	30	2	70	6	256	8	4	0.29	12.91	mcf

Table 4. K=4 Compromise Architectures

In the microarchitectural context with p design parameters, we wish to cluster architectures occupying a p dimensional space. The Euclidean distance between two normalized and weighted vectors of parameter values quantifies similarity in steps 2 and 3. Each cluster corresponds to a grouping of similar architectures and each centroid represents its cluster's compromise architecture. We take the number of clusters as the number of distinct compromise designs and, thus, a measure of heterogeneity.

Table 4 identifies compromise architectures and their average power-delay characteristics when executing their associated benchmarks in a $K = 4$ clustering. The four compromise architectures capture all combinations of pipeline depths and widths. Cluster 1 contains the aggressive deep, wide pipeline for jbb and mesa. Cluster 4, containing the memory-intensive mcf, is characterized by a large L2 cache and shallow, narrow pipeline. Clusters 2 and 3 trade-off pipeline depth and width depending on application-specific opportunities for instruction level parallelism.

Figure 8 plots the delay and power characteristics of the nine benchmark architectures executing their corresponding benchmarks (radial points). Aggressive architectures with deep, wide pipelines are located in the upper left quadrant and the less aggressive cores with shallow, narrow pipelines are located in the lower right quadrant. Deep,narrow and shallow,wide architectures both occupy the moderate center. The four compromise architectures executing their benchmark clusters are also plotted (circles) to demonstrate the delay-power compromises with associated per benchmark optima. Although we cluster in a p-dimensional microarchitectural space, the strong relationship between an architecture and its delay-power characteristics means we also observe clustering in the 2-dimensional delay-power space. Spatial locality between a centroid and its cluster's objects suggest modest delay and power penalties from architectural compromises. Thus, the delay-power characteristics of the benchmark suite executing on a heterogeneous multiprocessor with these four cores are similar to those when executing on the nine benchmark architectures. As a corollary, the benchmarks could achieve close to ideal $bips^3/w$ efficiency on this heterogeneous design.

6.2 Heterogeneity Trends and Validation

Figure 9(a) plots predicted $bips^3/w$ efficiency gains for the nine benchmarks and the benchmark average as the number of clusters increases in the K-means algorithm. Recall cluster count quantifies the degree of heterogeneity. Efficiency is presented relative to the POWER4-like baseline (cluster count 0). The homogeneous architecture identified by K-means clustering (cluster count 1) is predicted to improve average efficiency by 1.46x with the largest gains for mesa (4.6x) at the expense of mcf (0.46x). For three cores, all benchmarks see benefits from heterogeneity resulting in an average gain of 1.9x. We observe diminishing marginal returns in heterogeneity beyond 4 cores. The four cores in Table 4 are predicted to benefit efficiency by 2.2x, 8 percent less than the theoretical upper bound of 2.4x that is achievable only from the much greater heterogeneity of 7 to 9 cores. The benefits for nine different cores is the theoretical upper bound on heterogeneity benefits as each benchmark executes on its $bips^3/w$ maximizing core.

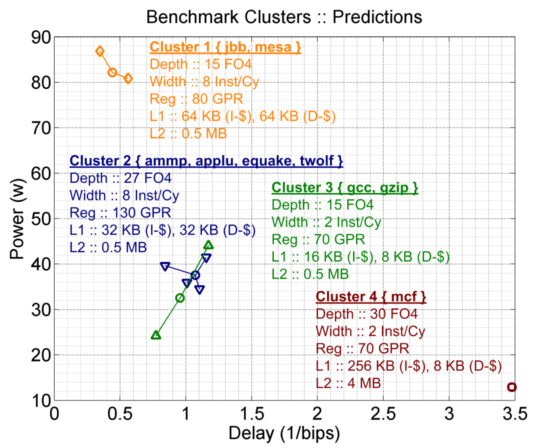
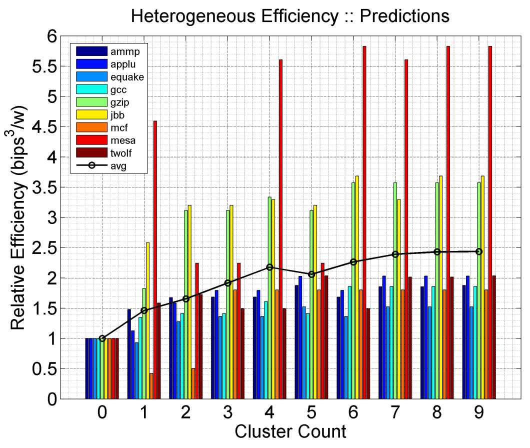


Figure 8. Delay and power for per benchmark optima of Table 2 (radial points) and resulting compromises of Table 4 (circles).

Figure 9(b) presents efficiency gains observed when simulating compromise architectures from the clustering analysis. The models capture application-specific effects such as the significant benefits for mesa as cluster count increases and the efficiency sacrifices of mcf for 2 or 3 clusters to benefit the overall benchmark average. Although the models over-estimate efficiency benefits, they capture the relative benefits across benchmarks. The simulated four core average benefit is 1.5x versus the modeled benefit of 2.2x and the upper bound of heterogeneity benefits is simulated



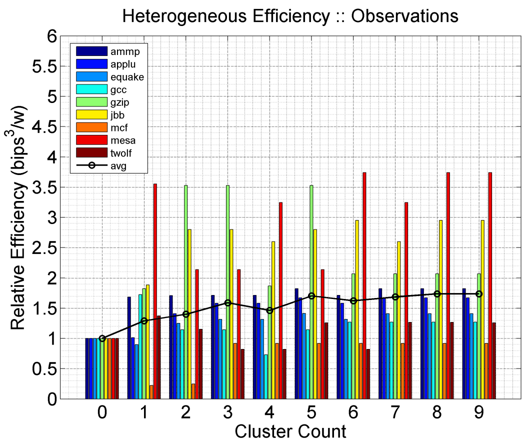


Figure 9. (a) Predicted and (b) simulated efficiency gains. Cluster 0 is baseline, cluster 1 is homogeneous multicore from K-means, cluster 9 is heterogeneous multicore of benchmark architectures.

at 1.7x versus the modeled bound of 2.4x. Note, however, that relative gains are consistent with four cores achieving 92 and 88 percent of the theoretical maximum in regression and simulation, respectively.

7 Related Work

Zyuban, *et al.*, examined power and performance effects of varying pipeline depths [26]. Hartstein, *et al.*, and Hrishikesh, *et al.*, also studied pipeline depth optimality [8, 9]. These studies held the majority of design parameters at constant values while our pipeline study simultaneously varies a large number of additional parameters.

Kumar, *et al.*, identify heterogeneous cores from a modest design space. Design alternatives were evaluated with exhaustive simulation [13]. For homogeneous multiprocessors, Davis, *et al.*, suggest less aggressive in-order cores are performance optimal [3], and Huh, *et al.*, suggest larger out-of-order cores maximize throughput [10]. Both design spaces are relatively modest as experience and intuition were used to prune the space. In contrast, we consider the entire design space, enabling the discovery of potentially unexpected optima.

Eeckhout, *et al.*, study statistical simulation for simplifying workloads in architectural simulation [4]. Nussbaum, *et al.*, examine similar statistical superscalar and symmetric multiprocessor simulation [17]. Both profile benchmarks to construct smaller, synthetic benchmarks with similar characteristics. Introducing sampling and statistics into simulation frameworks reduces accuracy in return for gains in speed and tractability. While Eeckhout and Nussbaum suggest this trade-off for simulator workload inputs to reduce per simulation costs, we propose this trade-off for resulting outputs to reduce the number of required simulations.

Ipek, *et al.*, predict the performance of design spaces with automated artificial neural networks (ANN) trained by gradient descent and predicted by nested weighted sums [5]. Our approach requires greater statistical analysis, but is more computationally efficient, numerically solving and evaluating linear systems for training and prediction.

Eyerman, *et al.*, combine synthetic trace simulation with heuristics to search for global optima within a design space [6]. The most effective heuristics, variants of steepest descent and genetic search, require between 900 and 1,000 simulations *per optimization problem*. However, these simulations are specific to a given optimization problem since they simulate design points along a particular path taken to the estimate of a particular metric's optimum. In contrast, our regression models require 1,000 simulations *per design space* since they may be formulated once and used in multiple studies. Furthermore, our models could also be applied within heuristics to significantly reduce search time.

Joseph, *et al.*, derive performance models using stepwise regression, an automatic iterative approach for adding and dropping predictors from a model depending on measures of significance [12]. However, stepwise regression produces significant biases [7] and this prior work does not predict performance, using the models only for significance testing. In contrast, we derive and apply predictive performance and power models for design space exploration.

8 Conclusions and Future Directions

We present a series of diverse design space studies to motivate the use of techniques in statistical inference in microarchitectural research. In particular, we apply microarchitectural performance and power regression models to pareto frontier, pipeline depth, and multiprocessor heterogeneity analyses. We find pareto optima predictions are no less accurate than those for the broader design space, pipeline depth studies may not generalize when the majority of design parameters are held at constant values, and multiprocessor heterogeneity has significant potential for improving power-performance efficiency. In each study, we demonstrate regression modeling's ability to comprehensively capture trends in a large design space while controlling simulation costs. The computational efficiency of obtaining predictions enable much more aggressive studies previously not possible via simulation.

We intend to expand our models to support other parameters such as cache-associativity and in-order execution. For larger design spaces, we may apply the models in heuristic search instead of exhaustive prediction. Because regression models produce analytical equations, symbolic optimization may also be feasible.

References

[1] D. Brooks, P. Bose, V. Srinivasan, M. Gschwind, P. G. Emma, and M. G. Rosenfield. New methodology for early-stage, microarchitecture-level power-performance analysis of microprocessors. *IBM Journal of Research and Development*, 47(5/6), Oct/Nov 2003.

[2] D. Brooks and et. al. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6):26–44, Nov/Dec 2000.

[3] J. Davis, J. Laudon, and K. Olukotun. Maximizing cmp throughput with mediocre cores. In *PACT05: International Conference on Parallel Architectures and Compilation Techniques*, September 2005.

[4] L. Eeckhout, S. Nussbaum, J. Smith, and K. DeBosschere. Statistical simulation: Adding efficiency to the computer designer's toolbox. *IEEE Micro*, Sept/Oct 2003.

[5] E.Ipek, S.A.McKee, B. de Supinski, M. Schulz, and R. Caruana. Efficiently exploring architectural design spaces via predictive modeling. In *ASPLOS-XII: Architectural support for programming languages and operating systems*, October 2006.

[6] S. Eyerman, L. Eeckhout, and K. D. Bosschere. Efficient design space exploration of high performance embedded out-of-order processors. In *Design, Automation, and Test in Europe*, March 2006.

[7] F. Harrell. *Regression modeling strategies*. Springer, New York, NY, 2001.

[8] A. Hartstein and T. Puzak. The optimum pipeline depth for a microprocessor. In *International Symposium on Computer Architecture*, May 2002.

[9] M. Hrishikesh, K. Farkas, N. Jouppi, D. Burger, S. Keckler, and P. Sivakumar. The optimal logic depth per pipeline stage is 6 to 8 fo4 inverter delays. In *International Symposium on Computer Architecture*, May 2002.

[10] J. Huh, D. Burger, and S. Keckler. Exploring the design space of future cmps. In *PACT01: International Conference on Parallel Architectures and Compilation Techniques*, September 2001.

[11] V. Iyengar, L. Trevillyan, and P. Bose. Representative traces for processor models with infinite cache. In *Proceedings of the 2nd Symposium on High Performance Computer Architecture*, February 1996.

[12] P. Joseph, K. Vaswani, and M. J. Thazhuthaveetil. Construction and use of linear regression models for processor performance analysis. In *Proceedings of the 12th Symposium on High Performance Computer Architecture*, Austin, Texas, February 2006.

[13] R. Kumar, D. Tullsen, and N. Jouppi. Core architecture optimization for heterogeneous chip multiprocessors. In *PACT'06: International Conference on Parallel Architectures and Compilation Techniques*, April 2006.

[14] B. Lee and D. Brooks. Accurate and efficient regression modeling for microarchitectural performance and power prediction. In *ASPLOS-XII: International Conference on Architectural Support for Programming Languages and Operating Systems*, October 2006.

[15] B. Lee and D. Brooks. Statistically rigorous regression modeling for the microprocessor design space. In *ISCA-33: Workshop on Modeling, Benchmarking, and Simulation*, June 2006.

[16] M. Moudgill, J. Wellman, and J. Moreno. Environment for powerpc microarchitecture exploration. *IEEE Micro*, 19(3):9–14, May/June 1999.

[17] S. Nussbaum and J. Smith. Modeling superscalar processors via statistical simulation. In *PACT2001: International Conference on Parallel Architectures and Compilation Techniques*, Barcelona, Sept 2001.

[18] A. Phansalkar, A. Joshi, L. Eeckhout, and L. John. Measuring program similarity: experiments with spec cpu benchmark suites. In *ISPASS05: International Symposium on Performance Analysis of Systems and Software*, March 2005.

[19] K. Ramani, N. Muralimanohar, and R. Balasubramonian. Microarchitectural techniques to reduce interconnect power in clustered architectures. In *ISCA-31: Proceedings of the Workshop on Complexity Effective Design*, June 2004.

[20] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *Tenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, October 2002.

[21] P. Shivakumar and N. Jouppi. An integrated cache timing, power, and area model. In *Technical Report 2001/2, Compaq Computer Corporation*, August 2001.

[22] C. Stone. Comment: Generalized additive models. *Statistical Science*, 1:312–314, 1986.

[23] R. D. Team. *R Language Definition*.

[24] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe. SMARTS: Accelerating microarchitecture simulation via rigorous statistical sampling. In *International Symposium on Computer Architecture*, June 2003.

[25] V. Zyuban. Inherently lower-power high-performance superscalar architectures. In *Ph.D. Thesis, University of Notre Dame*, March 2000.

[26] V. Zyuban, D. Brooks, V. Srinivasan, M. Gschwind, P. Bose, P. Strenski, and P. Emma. Integrated analysis of power and performance for pipelined microprocessors. *IEEE Transactions on Computers*, Aug 2004.