Computer Sciences Department

A Statistical Exploration of the CMP Design Space

Samuel A. Koblenski

Technical Report #1510

July 2004

UNIVERSITY OF WISCONSIN MADISON

A Statistical Exploration of the CMP Design Space

Samuel A. Koblenski

University of Wisconsin-Madison
Department of Electrical and Computer Engineering
1415 Engineering Drive
Madison, WI 53706

Submitted in partial fulfillment of the M.S. Degree in Electrical and Computer Engineering
Project Option

May 2004

A Statistical Exploration of the CMP Design Space

Abstract

Computer architects use simulation extensively to design computer systems, and they primarily rely on their intuition and detailed knowledge of a system to guide the simulation experiments that they perform. However, using a statistical design to construct an experiment and using statistical analysis to interpret the results will reduce the simulation runs necessary to optimize the system, enhance understanding of the main factors and interactions of the system, and increase confidence in the attained optimal design point. Moreover, the advent of Chip-Multiprocessors (CMPs) is challenging established ideas about multiprocessor design. The architect can use a statistical design to quickly explore the novel design space of CMPs and evaluate which factors are more important to improve performance or to reduce cost. This paper describes the method of constructing fractional factorial designs and analyzing the results to build a statistical model of the CMP design space.

1 Introduction

Simulators have proven invaluable to the design of next-generation microprocessors, and with good reason. Simulation is an extremely flexible and cost-effective way to evaluate architectural features and tradeoffs before committing to a design. It enables computer architects to evaluate design points while the system is not fully specified, greatly enhancing the performance potential of the end product.

Such a powerful tool comes with tradeoffs, the most apparent being the large amount of time and computational resources required to run detailed simulations. This constraint compels the designer to collect the maximum amount of information from a minimal number of simulation runs. However, most simulation methodologies fall into one of two sub-optimal strategies: the best-guess approach and the one-factor-at-a-time approach [28].

The best-guess approach relies heavily on the designer's intuition and knowledge of the system under simulation. The designer decides on initial parameter values that are thought to give close to optimal performance, runs the simulation, and reasons about the results. New values are chosen for the parameters thought to have the greatest impact on performance; simulations are run using these new values; and the process is repeated until a desired performance level is reached. This method can be thought of as a depth-first search of the design space, because the designer decides at each point where to go next based on current and previous performance data. As long as the designer's intuition is correct, this approach works reasonably well. An overall view of the design space is never developed, but that may not be necessary to produce good results. On the other hand, this approach has no guarantee of success. The process could be repeated indefinitely without finding the desired performance level, and if a local minimum in the design space is found, the designer may stop searching for more gains prematurely.

The one-factor-at-a-time approach also starts with a baseline configuration simulation, similar to the best-guess approach. Then each parameter is varied individually against the baseline and the value producing the maximum performance for each parameter is selected. This approach has intuitive appeal due to its relative simplicity. The behavior of this type of search more closely follows a breadth-first search of the design space, although just like the

best-guess approach, the entire design space is not covered. It also suffers from the inability to observe any *interaction* effects between parameters. Two parameters interact when one parameter's effect on performance does not stay constant as the other parameter is varied. This type of information is lost in the one-factor-at-a-time approach. Additionally, a more powerful design will be much more efficient in simulation time than this approach.

Montgomery [28] provides an excellent treatment of some more powerful designs with the one chosen for this work being the fractional factorial design. The theory behind these and other statistical designs is based in combinatorics [33, 37, 38]. A factorial design allows an experimenter to study the parameters of a design space. These parameters are called *factors*, and for this work each parameter will assume two values within its range. These designs are called two-level factorial designs because the factors take on two different levels. To fully study the design space, every permutation of the factor levels must be run. This full factorial design will require 2^k simulations for a system of k factors. This design produces independent information for all interactions, where a one-factor interaction, or *main effect*, is the effect of a single factor, a two-factor interaction quantifies the effect that two factors have on each other, and so on. Since the number of simulations in a full factorial design becomes prohibitively expensive for anything but a small number of factors, a fractional factorial design can be constructed that reduces the interaction information available in the results somewhat, but requires only a fraction of the simulations.

The benefits of a fractional factorial design are three-fold:

1) The limited number of simulations necessary to gather the desired information makes the design efficient. The designer can decide ahead of time what level of information should be produced regarding main effects and interactions. Then the exact number of simulations required to produce that level of information is run.

2) The information on specific main effects and interactions can be isolated, giving the designer invaluable knowledge about subtle interactions in the system.

3) The rigorous design and analysis of the experiment instills confidence that the simulations have provided accurate information about the system, and that the design point ultimately chosen is robust, i.e. it is likely to be a global maximum and it is resilient to the effects of experimental uncertainty.

In this paper we apply a fractional factorial design to the relatively new design space of Chip-Multiprocessors (CMPs) to discover the dominant main effects and interactions in the space, and to develop an analytical model of the main effects to provide designers with a solid intuition about the tradeoffs inherent to a CMP.

The rest of this paper is organized as follows: Section 2 describes the construction of a fractional factorial design and discusses some of its important properties. Section 3 explains how to effectively analyze the experimental results and build an analytic model using the results. Section 4 describes our simulation environment. Section 5 develops a design for the CMP space, Section 6 presents the results of the CMP experiment, Section 7 discusses related work, and Section 8 concludes.

2 Fractional Factorial Designs

A 2^k factorial design is the most powerful of the two level designs since all main effects and interactions are independently measurable. However, the measurement of all interactions, especially higher interactions involving 3 or more factors, is not usually required because they are normally insignificant. Fractional factorial designs reduce the number of simulations required to estimate the main effects and lower interactions by some fraction $1/2^p$, producing a 2^{k-p} design. The number p will determine how many higher interactions will be associated, or *aliased,* with lower interactions. Constructing a design in a way that controls the aliasing structure is relatively straightforward. The following sections describe this construction procedure, the resulting aliasing structure, and another useful property of this design called projection.

2.1 Design Construction

The description of constructing a fractional factorial design is best done by example, so let us assume that a system has five factors (A, B, C, D, and E) where each factor has a high (+) and a low (-) value. Now suppose we cannot afford to run the full factorial design of 32 simulations, and we are willing to consider most two-factor and higher interactions as insignificant. We can construct a 2^{5-2} design of eight simulations by first constructing the *basic design* of the first k-p factors as a full factorial design. Table 1 shows that the basic design consists of every permutation of the factors A, B, and C where each permutation has each factor set at its high (+) or low (-) value. Now we must decide on the values that D and E will assume for each run. No matter how the values are assigned, interactions will be aliased with each other because only a full factorial design produces enough information to independently measure each interaction. However, the aliasing structure can be controlled by choosing appropriate design generators for the last p factors.

	Basic Design				
Run	A	B	C	D = AB	E = AC
1	-	-	-	+	+
2	-	-	+	+	-
3	-	+	-	-	+
4	-	+	+	-	-
5	+	-	-	-	-
6	+	-	+	-	+
7	+	+	-	+	-
8	+	+	+	+	+

Table 1: A 2^{5-2} fractional factorial design

A generator is an interaction involving factors from the basic design that is intentionally aliased with one of the leftover factors. In Table 1, the interaction AB denotes the effect of the interaction between factors A and B. This interaction is intentionally aliased with factor D, and interaction AC is aliased with factor E. The signs for each of the p remaining factors are found by multiplying the signs of the factors from the basic design for each run. For example, the sign that factor D should assume for run 1 is found by multiplying the negatives for A and B together

to get a positive sign. Choosing good design generators is essential to controlling the aliasing structure of the design, and in most cases, more than one set of generators will produce the desired aliasing structure. This concept will be explained further in the next section.

2.2 Aliasing

The aliasing structure of a design specifies the main effects and interactions that cannot be disambiguated in the results, and it can easily be found from the generators used to construct the design. The generators impose a *defining relation* on the design, which is used to find the interactions that are aliased with each other. The defining relation is found by first listing each generator appended with its main effect. These sets of factors are called *generating relations*, and they make up part of the defining relation. Each set of factors in the defining relation is called a *word.* In our example, the generators D = AB and E = AC, where '=' denotes aliasing, become the generating relations ABD and ACE. Then each combination of these generating relations are multiplied together to produce the rest of the words in the complete defining relation. Factors that are squared in a word are removed, so by multiplying ABD and ACE we get A^2BCDE, which becomes BCDE. Thus, the defining relation is I = ABD = ACE = BCDE. The I factor is an identity factor. The result of the product of the identity factor with any other factor is that factor. Now multiplying each word of the defining relation with a main effect or interaction will produce a set of factors and interactions that are aliased together. Repeating this procedure with every factor or interaction not already part of a previous alias set will produce a complete aliasing structure for the design. In our example the aliasing structure is

A = BD = CE = ABCDE	D = AB = BCE = ACDE	BC = DE = ACD = ABE
B = AD = CDE = ABCE	E = AC = BCD = ABDE	BE = CD = ADE = ABC
C = AE = BDE = ABCD		

From this aliasing structure, we can see that if only two two-factor interactions are known to be important, both having one factor in common, this design will be adequate. The factors making up the two interactions can be assigned to B, C, and E with C being the common factor. Then the interactions of interest, BC and BE, are not aliased. As long as the designer assigns factors to the correct design columns, the simulation results will contain the desired information. This approach requires the designer to have some prior knowledge about which two-factor interactions are likely to affect performance and which interactions can be neglected.

If such knowledge is not available or more interactions exist, a stronger design must be constructed. The strength of a design can be characterized by its *resolution.* A resolution III design has no main effects aliased with each other. Our example design is of resolution III and is denoted as a 2^{5-2}_{III} design. A resolution IV design adds that no main effects are aliased with any two-factor interactions. A resolution V design adds that no two-factor interactions are aliased with each other, and so on. In general, a design has a resolution equivalent to the smallest number of letters found in any of the words in its defining relation [28].

One simple way to strengthen a design is to replicate the entire fractional factorial design with all signs in the design matrix reversed. This procedure is called *folding over* a design, and it converts any resolution III design into a

resolution IV design. Folding over a design has the effect of negating all words consisting of an odd number of factors. To find the new defining relation, all words with an even number of factors are left unchanged, and all even products of relations with an odd number of factors are also included. An even product is a product of an even number of relations. Taking the products of an even number of the odd words causes the sign of the resulting word to flip again so that it is positive. The end result is a defining relation with all positive words.

Thus the even word of our example 2_{IV}^{5-2} design with fold-over is BCDE, and the only even product is ABD * ACE = BCDE. Since these words are equivalent, the defining relation is BCDE and the aliasing structure is free of aliasing between main effects and two-factor interactions. In this case a better design with the same number of runs would be a 2_{V}^{5-2} design with the defining relation ABCDE. Using fold-over to increase the resolution of a design must be done with care, because usually it will not produce a design with the best aliasing structure. It is mainly useful when a design has already been executed, and the experimenter wants to remove some aliases without using a completely new design and wasting the data already gathered.

If the designer wants to de-alias specific interactions after a design has been simulated, it is possible to augment the design with one additional run for each pair of aliased interactions that need to be separated. The reason that two interactions are aliased is because their columns in the design matrix are equal, just as the columns for D and AB are equal in the example 2_{III}^{5-2} design. Adding another run with values assigned to the main factors such that the two interactions have different signs effectively de-aliases them. Using this technique, a designer can simulate a small fractional factorial design, analyze the results, and decide which additional runs to simulate to separate the important aliased interactions, making fractional factorial designs an extremely powerful time-saving simulation tool.

2.3 Projection

The fractional factorial design method is beginning to look fairly powerful. But suppose the designer needs to measure higher interactions that are still aliased in a design containing a reasonable number of runs. If the designer believes that at least some of the main effects will be negligible, the fractional factorial design can be *projected* onto a stronger design or even a full factorial design. Then the results can be analyzed as if this stronger design was executed. Care must be taken when using this technique since the stronger design was not actually utilized, and ignored higher interactions could have an effect on the results. Using intuition about the system is imperative when justifying the projection. Running the stronger design as a check would strengthen any conclusions made about the projected design.

To project one design onto another, factors with insignificant effects must be selected and eliminated from the design. Any words in the defining relation that contain the eliminated factors are removed, reducing or eliminating the aliasing structure of the design. In our example, if factors C and D are deemed insignificant then all words in the defining relation (ABD = ACE = BCDE) are eliminated because one or both of the insignificant factors appears in each word. Thus, the design becomes a full factorial design in A, B, and E. Now all interactions involving A, B, and E can be observed without having to run additional simulations. If simulation parameters are assigned to factors judiciously during construction, the design will more likely project onto a stronger design. It is possible for a design

to be projected onto another design with every run duplicated some number of times. When this duplication occurs, the projected design is said to be replicated, and the extra runs can be used to better approximate experimental error. Unfortunately, interactions that are aliased in the original design and consist of only significant factors cannot be de-aliased using projection because the relations that define the alias will not be eliminated.

3 Analysis of Results

The results from a fractional factorial design allow the designer to estimate the main effects and their interactions and to construct and validate a statistical model of the design space. A careful analysis that separates significant effects from negligible effects involves evaluating experimental error through the analysis of variance (ANOVA). Validating a model requires an examination of the model's residuals and a test of fit between the model and results. The following sections focus on the procedures for using these analyses. For the derivations justifying these procedures, refer to the literature [9, 10, 28].

3.1 Analysis of Variance

The effect of each factor or interaction is estimated by taking the difference of the average response when the factor is high and the average response when the factor is low. This calculation is equivalent to the following procedure:

1) Add columns to the original design matrix for the interactions not aliased to the main effects. The signs for the column entries are found in the same way as the generated main effects.
2) Add a column listing the responses for each run.
3) Multiply each column by the response column and find the sum of each column.
4) Divide the sum of each column by 2^{k-p-1} to obtain the effect of each factor or interaction.

Run	A	B	C	D	E	BC	BE	Metric
1	-	-	-	+	+	+	-	45
2	-	-	+	+	-	-	+	100
3	-	+	-	-	+	-	+	45
4	-	+	+	-	-	+	-	65
5	+	-	-	-	-	+	+	75
6	+	-	+	-	+	-	-	60
7	+	+	-	+	-	-	-	80
8	+	+	+	+	+	+	+	96
Effect	14	1.5	19	19	-18.5	-1	16.5	

Table 2: Estimated effects of main effects and interactions, adapted from Montgomery [28]

Table 2 shows the example 2^{5-2}_{III} design with effects estimated. Since effects near zero indicate that the effect is likely to be insignificant, the metric should be chosen so that a larger response indicates higher performance. ANOVA is used to determine which factors are significant. Normally, a statistics software package is used to run the calculations, but seeing an example is still instructive.

First, the sum of squares of each factor must be calculated. The factor columns are summed in the same way as they were previously, except that the metric is squared for each run first. The resulting effects are divided by the number

of runs executed. The sums of squares for the factors that are suspected of being significant are left independent, and the remaining sums of squares are added together to approximate the experimental error. Then, each sum of squares is divided by the degrees of freedom for that factor to obtain the mean squares. In fractional factorial designs each factor has one degree of freedom, and the degree of freedom of the error is equivalent to the number of factors that make up the error. Next, the F ratio is found by dividing each factor's mean square by the error's mean square. Finally, the probability value that each insignificant factor is looked up in an F-distribution table [28].

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F ratio	Probability
A	392	1	392	120.6	<0.01
C	722	1	722	222.2	<0.01
D	722	1	722	222.2	<0.01
E	684.5	1	684.5	210.6	<0.01
BE	544.5	1	544.5	167.5	<0.01
Error	6.5	2	3.25		
Total	3071.5	7			

Table 3: Analysis of Variance for the 2^{5-2}_{III} design example

These steps are shown in Table 3. The probability values in the rightmost column specify the probability that the variation due to each factor is insignificant. Thus, the factors thought to be significant are significant with a probability greater than 99%. A software package would compute the probabilities more precisely. Another useful observation from this data is found by adding up the sum of squares of A, C, D, E, and BE and comparing the result to the total sum of squares, which is the total variation of the simulations. The selected factors account for 99.8% of the variation in the response so the result that they are significant is intuitively satisfying.

3.2 Regression Models

Following the completion of ANOVA, constructing a linear regression model is quite easy. Since each factor is varied between two values, each estimated effect for the significant factors is divided in half to get the midpoint of the effect. These midpoints are assigned to the coefficients for each factor, and averaging the responses for all runs gives the offset for the model. The regression model for our example would be

$$y = 70.75 + 7A + 9.5C + 9.5D - 9.25E + 8.25BE$$

The residuals of the regression model are found by plugging the factor values for each run into the model and finding the difference between the response in simulation and the response of the model. A plot of the residuals versus the simulation response should show no patterns or trends, i.e. they should have a normal distribution.

How can the designer be sure that a linear regression model is appropriate? One straightforward solution is to test for lack of fit. To perform this test, some number of additional runs must be simulated at the design center-point, i.e. with all factors set halfway between their high and low values. Multiple runs are required at the center-point because the variance of the response at this point will be used to determine the *pure error* associated with the simulations. This pure error quantifies the variance inherent in the simulation environment, while the *total error* found above using ANOVA includes variations between simulation runs of different configurations. If these two errors are significantly different, it means that either higher order model terms or other interactions are contributing to the total

error. The lack of fit error is the difference between the total error and the pure error. The same F-distribution test as was used for testing the significance of selected factors is used to calculate the lack of fit significance. The F ratio is the ratio of lack of fit error to pure error and the resulting probability value specifies the probability that a linear regression model fits the data.

If a linear model is not adequate, the data should be fit to a higher order model. If the design is augmented with runs in which each factor appears at its high and low value and all other factors are at their center-points, a second-order model can be fit to the data. This design is called a face-centered central composite design, and it can add a significant number of runs to the original design. If a majority of the main effects of the original design are insignificant, this design can be useful since only the additional runs corresponding to the significant main effects need to be simulated. Since fitting data to a second-order model is much more complicated than for a linear model, a statistical software package is normally used to run the analysis. However, fitting to a second-order model, if necessary, provides much more information about the design space, possibly revealing minimum and maximum points in the space and generally giving the designer a better view of the space.

4 Simulation

To run the simulations required for the fractional factorial design, we use the Simics full system simulator from Virtutech AB [26]. Simics can boot and run unmodified Solaris 9 code for a SPARC V9 system with up to 16 processors, enabling the simulation of commercial workloads that require operating system calls. Since Simics only has a simple timing model in which each instruction executes in a single cycle, we use the interface that Simics provides to plug in our own cycle-accurate processor timing model [27] similar to that of a MIPS R10000 and a detailed memory hierarchy model that simulates all forms of latency and network contention [3].

Normally simulators are deterministic in that successive runs of the same configuration will produce exactly the same results. This property is completely unlike that of a real system, which will not exactly reproduce results for successive runs of the same workload due to variations in the timing of interrupts, I/O operations, context switches and other non-deterministic events. Estimating the pure error of the simulated system for the lack of fit test would not work in a deterministic simulator because the simulation response of the design's center point would have zero variance. We approximate the non-deterministic behavior of a real system by adding a slight random perturbation to the cache hit latencies, which results in unique executions to the extent that different processes may be resident in the processors at the end of each run [5]. With the simulated systems exhibiting some variation, the pure error of the system can be estimated, and the lack of fit test can be used.

The commercial workloads used for this study are listed in Table 4, and include an online transaction processing workload (OLTP), a Java middleware workload (SPECjbb), and two web server applications (Apache and Zeus). Each workload was simulated in Simics for a large number of transactions in order to reach a steady state before taking a checkpoint. Then simulations were were continued from each checkpoint with the memory timing model enabled to generate memory instruction traces that are used to warm up the caches in detailed simulation. Finally, the workloads were run forward from the warm up phase with the memory and processor timing models to gather the performance results for each configuration of the statistical design.

Workload	Transactions			Description
	Fast Forward	Warm-up	Executed	
Apache	500000	2000	500	A static web server configured with a repository of 80,000 files serving 400 clients [4]
SPECjbb	1000000	15000	10000	A server-side 3-tier Java system that models the middleware server business logic [4]
OLTP	100000	300	100	A TPC-C benchmark modeling the database of a wholesale supplier using IBM's DB2 v7.2 EEE database management system [4]
Zeus	500000	2000	500	Another static web server configured similarly to Apache, but exhibits different characteristics

Table 4: Workloads run on the design

Name	Parameter	Low Value (-)	High Value (+)	Area Cost (CBE)
A	Pipeline Organization	In-Order	Out-of-Order	D * 1.25
B	Pipeline Width	1	4	D * $Width^2$
C	Outstanding Requests	1	4	(D/4) * Req^2
D	Instruction Window	32	128	32 * $IW_{ENTRIES}$
E	Branch Predictor [14]	2kB YAGS	16kB YAGS	$BPred_{SIZE}$
F	Number of Cores	2	16	Core Area * Cores
G	L1 Cache Size	16 kB	128 kB	$L1_{SIZE}$
H	Integer ALUs	2	6	7.5kB * ALUs
J	L1 Associativity	1-Way	8-Way	IF($L1_{SIZE}$=16kB) G * 2.4
K	Memory Latency	150 Cycles	450 Cycles	256kB
L	L2 Associativity	1-Way	8-Way	N/A
M	L2 Latency	16 Cycles	48 Cycles	N * 16kB
N	L2 Banks	4	32	$L2_{SIZE}$ * 1.1
O	L2 Link Bandwidth	16 Bytes	128 Bytes	(F * N) * $L2_{BW}$
P	Memory Bandwidth	16 GB/s	128 GB/s	1024kB

[A] The in-order cores used a certain area, and out-of-order cores used 25% more area than the in-order cores.

[C] Increasing the number of outstanding requests only affects the load-store queue, which makes up one fourth the area cost of the instruction window.

[D] The instruction window cost includes the reorder buffer and load-store queue costs with the reorder buffer having twice the entries and the load-store queue having the same number of entries as the instruction window.

[J] The high L1 associativity only affects the area of the small L1 cache. The cost of high associativity in the large L1 cache is amortized over the cache size and is negligible.

[K] The area cost only applies to shorter memory latency, assuming a memory controller is integrated on chip to reduce latency [23]. The memory itself would also need to be faster to produce the simulated difference in latencies.

[M] The area cost only applies to shorter L2 cache latency, assuming wire delays are reduced using a technique like transmission lines to reduce latency [8].

[N] The area cost only applies to 32 L2 banks, and was found using ECACTI.

[O] The L2 link bandwidth cost captures the cost of additional switches necessary for more wires at higher bandwidths and more connectivity with more cores and L2 banks.

[P] The area cost only applies to the higher bandwidth, assuming more pads and larger pin drivers are necessary to achieve the higher bandwidth.

Table 5: Parameters and Low/High Values

Fifteen parameters were selected for the CMP design space, with the majority of the parameters dealing with the memory hierarchy architecture. For this design, we selected most memory system parameters that we could vary in simulation. Two notable exceptions are the cache coherence protocol and interconnect structure. These two parameters do not have well defined high and low values because they are categorical in nature. Statistical designs that accommodate multiple levels of parameters are much better at addressing the issues inherent to these types of parameters. Certain processor parameters such as branch predictor size and issue width were included based on their significant impact on uniprocessor performance [40]. The parameters used to explore the CMP design space along with their associated high and low values are listed in Table 5 with a letter name assigned to each parameter for easy reference. Other parameters that were not varied are listed in Table 6.

Fixed Parameters	Value	Area (CBE)
Indirect Branch Predictor	256 entries	2 kB
Return Address Stack	64 entries	0.5 kB
Integer Register File (logical + rename)	160 + 64	16 kB
Floating Point Register File (logical + rename)	64 + 128	16 kB
Integer Multipliers	1	6 kB
FPU	1	14 kB
Cache Block Size	64 Bytes	N/A
Memory Size	4 GB	N/A

Table 6: Fixed Processor Parameters

To realistically restrict the design space, the die size was fixed at 300mm^2 and the area of each component at the various parameter values was approximated assuming a 65nm process. The area cost associated with each parameter is also given in Tables 5 and 6. Footnotes are provided in the table to explain the meanings and reasonings behind the area costs that are not immediately obvious. The total die area consumed by each configuration was calculated, and all leftover area was allocated to the L2 cache. When the L2 cache is direct mapped, the cache size can only be varied in powers of two. For the 8-way set associative L2 cache, we allowed the associativity to vary between 6- and 10-way so that more L2 cache could fit on the die for a given configuration. This relaxation of the high associativity value gives the configurations with a high associativity an additional advantage of flexibility. In either case, since the size of the L2 cache can only be adjusted at a coarse granularity in our simulator, most configurations did not fill the die. This property is acceptable since the area model is only a coarse approximation, and we were primarily interested in bounding what could reasonably fit on a single die in a next generation process. In this way a certain cost was associated with the various parameter values so that the effects of parameters that have a large area penalty were not artificially amplified.

The area and latency impact of various cache parameters was approximated using ECACTI [2] and the area estimates of other structures were gathered using die area data collected for research on the CMP design space at the University of Texas at Austin [18, 20], as well as a multitude of die photographs taken from commercial processors [11, 12, 13, 21, 24, 36, 39]. We made use of the Cache Byte Equivalent (CBE) measurement unit described in Huh et al. [20] to simplify the area calculations. The sizes of various structures were measured in the equivalent amount of cache that would fill the area of the structure, and we calculated that a 300mm^2 die can hold a total of 16MB of

cache. It is assumed that an in-order instruction window is implemented as a simple FIFO queue while the out-of-order instruction window will grow quadratically in area as it grows in issue width. Both organizations do register renaming.

In addition to the area penalties, the length of the pipeline is increased as the core complexity, L1 cache size and L1 associativity are increased. The base pipeline for an in-order scalar core with a 32 entry instruction window and a 16kB direct mapped cache is seven stages long, consisting of one fetch stage, two decode stages, one issue stage, one execute stage and two retire stages. Increasing either the L1 size or associativity adds a fetch stage and a data cache access stage. Making the cores either out-of-order or superscalar adds a decode stage. Finally, increasing the instruction window increases the size of the reorder buffer, adding one more retire stage. Thus, a large out-of-order superscalar processor will have 12 stages and a three cycle data cache access latency.

Initially it would appear that we should use ECACTI to estimate the L2 access latency for different L2 cache configurations in a similar manner to what was done for the L1 cache. However, doing so would make the L2 latency a dependent parameter instead of a parameter varied in isolation as required by a factorial design. Making a parameter dependent tends to corrupt the results of these designs [28]. Instead, we assume that if nothing is done to reduce wire delays in future processors and processor frequencies continue to scale, L2 access latencies will increase substantially. It has been known for some time that wire delays will not scale well with future technologies [1, 32]. According to Beckmann et al. [8], if current trends continue, the L2 cache uncontended latency could reach 48 cycles or more for processors with aggressive clock rates. Implementing a novel interconnect such as transmission line caches could achieve access latencies around 16 cycles or lower, depending on L2 bank sizes.

The memory bandwidth numbers are derived from the ITRS roadmap [22], and assume an area penalty for implementing high bandwidth pins because they will require much larger serializing/deserializing circuits to drive the high frequency signals. Additional pins are also required for higher bandwidth, further increasing the area penalty of an aggressive memory interface. Approximate areas for the structures behind the L2 cache latency and memory bandwidth parameters are very difficult to come by, but a reasonable approximation used to restrict these parameters is better than none at all. The primary concern is to restrict what can reasonably fit on a single die rather than perform a detailed area-accurate design of a CMP system.

5 A CMP Fractional Factorial Design

The fractional factorial design used in this study is shown in Table 7. The letters across the first row correspond to the letter names in Table 5 for each factor, and the L2 cache sizes for each run are shown in the rightmost column. As designs become larger than about five factors, manual design generation and result analysis becomes error-prone and unmanageable. This 2_{IV}^{15-10} design was generated using a statistical design of experiment software package from Stat-Ease called Design-Expert [6]. With this program the user can generate designs and analyze results of designs with up to 15 factors and 256 runs. If larger designs are required, Statistica from StatSoft [35] handles designs of up to 100 factors.

Run	A	B	C	D	E	F	G	H	J	K	L	M	N	O	P	L2 $ (MB)
01	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8.0
02	-	-	-	-	+	-	-	-	-	+	+	+	+	+	+	10.0
03	-	-	-	+	-	-	+	+	+	-	-	+	-	+	+	8.0
04	-	-	-	+	+	-	+	+	+	+	+	-	+	-	-	12.0
05	-	-	+	-	-	+	-	+	+	-	+	-	+	-	+	10.0
06	-	-	+	-	+	+	-	+	+	+	-	+	-	+	-	8.0
07	-	-	+	+	-	+	+	-	-	-	+	+	+	+	-	8.0
08	-	-	+	+	+	+	+	-	-	+	-	-	-	-	+	8.0
09	-	+	-	-	-	+	+	-	+	+	-	-	+	+	-	8.0
10	-	+	-	-	+	+	+	-	+	-	+	+	-	-	+	7.0
11	-	+	-	+	-	+	-	+	-	+	-	+	+	-	+	8.0
12	-	+	-	+	+	+	-	+	-	-	+	-	-	+	-	10.0
13	-	+	+	-	-	-	+	+	-	+	+	-	-	+	+	12.0
14	-	+	+	-	+	-	+	+	-	-	-	+	+	-	-	8.0
15	-	+	+	+	-	-	-	-	+	+	+	+	-	-	-	12.0
16	-	+	+	+	+	-	-	-	+	-	-	-	+	+	+	8.0
17	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	4.5
18	+	+	+	+	-	+	+	+	+	-	-	-	-	-	-	4.0
19	+	+	+	-	+	+	-	-	-	+	+	-	+	-	-	10.0
20	+	+	+	-	-	+	-	-	-	-	-	+	-	+	+	8.0
21	+	+	-	+	+	-	+	-	-	+	-	+	-	+	-	8.0
22	+	+	-	+	-	-	+	-	-	-	+	-	+	-	+	12.0
23	+	+	-	-	+	-	-	+	+	+	-	-	-	-	+	8.0
24	+	+	-	-	-	-	-	+	+	-	+	+	+	+	-	12.0
25	+	-	+	+	+	-	-	+	-	-	+	+	-	-	+	12.0
26	+	-	+	+	-	-	-	+	-	+	-	-	+	+	-	8.0
27	+	-	+	-	+	-	+	-	+	-	+	-	-	+	-	12.0
28	+	-	+	-	-	-	+	-	+	+	-	+	+	-	+	8.0
29	+	-	-	+	+	+	-	-	+	-	-	+	+	-	-	8.0
30	+	-	-	+	-	+	-	-	+	+	+	-	-	+	+	10.0
31	+	-	-	-	+	+	+	+	-	-	-	-	+	+	+	4.0
32	+	-	-	-	-	+	+	+	-	+	+	+	-	-	-	7.0

Table 7: 2_{IV}^{15-10} Fractional Factorial Design Matrix

The first five factors in Table 7 are the basic design, and the rest of the factors are derived from the basic design factors using the following design generators:

F = ABC	G = ABD	H = ACD	J = BCD	K = ABE
L = ACE	M = ADE	N = BCE	O = BDE	P = CDE

The parameters were assigned to each factor in Table 5 such that the interactions AB, AC, AD, AH, BC, BD, BE, BK, BP, CK, CM, CP, FK, FP, and KP were not aliased with each other. We believed that the largest interactions would involve the number of cores, pipeline organization, pipeline width, outstanding requests, memory latency, and memory bandwidth. It is highly unlikely that an important interaction will involve factors that are not important, and since these parameters are likely to have a large effect on performance, interactions between them are also likely to affect performance. Due to the difficulty of arranging the parameters so that all potentially important interactions are not aliased, a couple sets of aliased interactions only involve factors believed to be less important.

This design was augmented with eight runs at the design centerpoint and an L2 cache size of 10 MB to estimate pure error and test for lack of fit for the linear regression model. The L2 cache was set to be 5-way set associative to

achieve this cache size. One issue in constructing centerpoints for this design is that a midpoint does not exist between an in-order pipeline and an out-of-order pipeline. The way to resolve this is to run half of the centerpoints with an in-order pipeline and the other half with an out-of-order pipeline. The results are then averaged to determine the true centerpoint. If other categorical factors are present in a design, the centerpoints must be further split to include each permutation of the categorical factors. If too many such factors are present, the centerpoint begins to look like its own full factorial design. At that point it may be more efficient to replicate the design to estimate pure error. One final issue is that a number of factors are restricted to be a power of two, so the exact midpoint cannot be simulated. If the midpoint of these factors is set to be as close to the true midpoint as possible, it will not introduce a significant amount of error to the lack of fit test. The test primarily relies on a replicated design point different from those in the fractional factorial design so that pure error can be estimated.

6 Results

Special care must be taken to select the important factors from the results and verify that the results are meaningfully interpreted. First, a systematic method of selecting and verifying important factors is presented. Second, the results for the simulated workloads are discussed. Third, the results from phase two are used to construct and run a refined statistical design that will be used to generate a regression model of the important factors in the CMP design space.

6.1 Results of the Fractional Factorial Design

The results for each workload were analyzed using Design-Expert. To facilitate the selection of the most important factors, the program computes the responses of each factor and interaction as described in Section 3.1, and displays the responses on a half normal plot. The plots of the responses for the four workloads studied are shown in Figure 1. These plots are constructed by ranking the absolute values of the responses from least to greatest and then plotting them on a half normal probability scale. The x-axis of the plot corresponds to the magnitude of the responses. The y-axis corresponds to the percentile of the response's rank. The appealing property of this plot is that any responses due to simulation noise will appear along a straight line starting at zero magnitude and zero percentile. All of the design centerpoints are also plotted as triangles and should appear along that same line. Any responses that fall off of the line have magnitudes that cannot be explained by noise alone and such responses have significant effects on performance. These responses are labeled with their factor names.

Once the important factors are selected, ANOVA is used to verify that the factors are indeed significant. A linear regression model should be constructed to verify that the responses do not exhibit undesired trends. Table 8 shows an example ANOVA for the Zeus benchmark computed by Design-Expert. All probabilities of insignificance are less than 0.3%, and the selected factors account for 92% of the variation in the results. (Note that all probabilities of insignificance are less than 3% across the selected factors for all workloads.) However, the curvature in the results, found by comparing the average of the design centerpoints to the average of the rest of the simulation runs, is also significant. The curvature quantifies the difference between the centerpoint of the linear regression model and the simulated centerpoint of the design. Because of this large curvature a linear model will not adequately fit the results, as borne out by the high significance of the lack of fit test. Regardless of this lack of fit, the linear regression model can still be used to perform some diagnostics on the data.

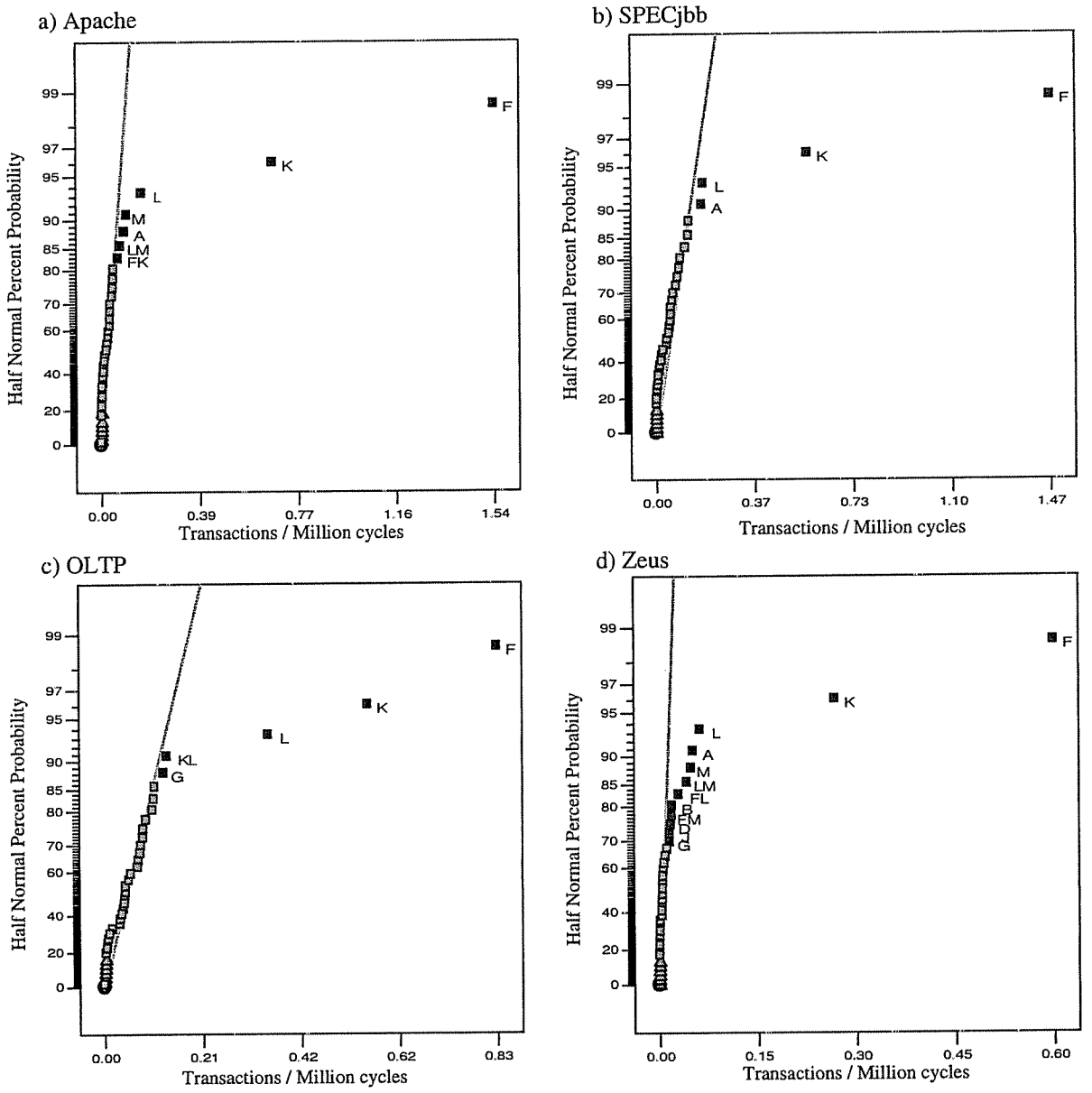

Figure 1: Half Normal Plots of simulated workloads:
Apache (a), SPECjbb (b), OLTP (c), and Zeus (d)

Two diagnostic tests that can be performed on the model are shown in Figure 2. Figure 2(a) shows a normal plot of the residuals for Apache. A normal plot differs from a half normal plot in that the residuals can be positive or negative, and the percentile scale is changed so that the residuals are clustered near the center of the plot. Residuals are found by taking the difference between the predicted response of the model and the actual response from simulation for each run. Residuals should all be due to noise so they should all appear along a straight line. That is clearly not true for the Apache residuals.

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F ratio	Probability
Model	4.4419	12	0.37016	1515.450	< 0.0001
A	0.0270	1	0.02699	110.485	< 0.0001
B	0.0041	1	0.00406	16.617	0.0004
D	0.0033	1	0.00333	13.632	0.0010
F	3.5934	1	3.59345	14711.636	< 0.0001
G	0.0028	1	0.00276	11.288	0.0024
J	0.0028	1	0.00277	11.346	0.0024
K	0.7140	1	0.71400	2923.111	< 0.0001
L	0.0393	1	0.03935	161.092	< 0.0001
M	0.0241	1	0.02415	98.857	< 0.0001
FL	0.0090	1	0.00901	36.886	< 0.0001
FM	0.0039	1	0.00391	16.026	0.0005
LM	0.0182	1	0.01818	74.427	< 0.0001
Curvature	0.3682	1	0.36820	1507.425	< 0.0001
Residual	0.0064	26	0.00024		
Lack of Fit	0.0061	20	0.00031	7.449	0.0098
Pure Error	0.0002	6	0.00004		
Cor Total	4.8165	39			

Table 8: ANOVA for Zeus benchmark

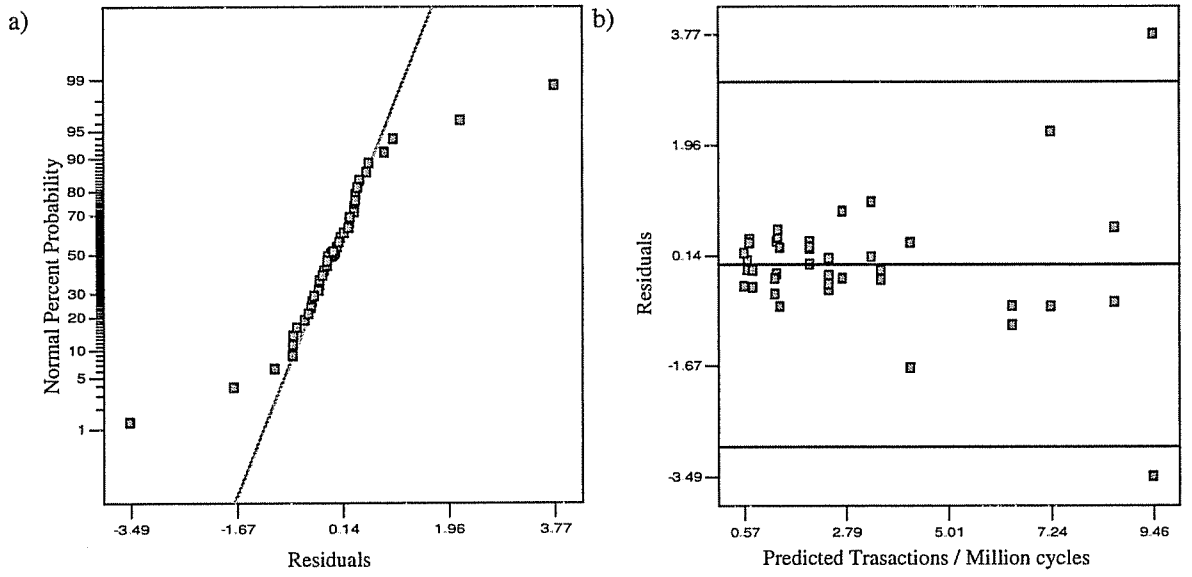

Figure 2: Normal Plot of Residuals (a) and plot of residuals vs. the predicted response of the model (b) for Apache

To get a better understanding of why this is happening, each residual can be plotted against its predicted response from the model as is done in Figure 2(b). The lines at the top and bottom of the plot denote the expected maximum and minimum residuals for the model. Any points falling outside this range are called outliers, due to their uncharacteristically large values compared to the rest of the residuals. Two outliers exist for the Apache residuals, and the residual plot has a funnel shape to it: residuals are smaller when the predicted response is smaller, and larger

when the response is larger. This characteristic suggests that the data should be compressed using a transform operation that condenses the points [28]. In other cases, data may need to be expanded by squaring each data point. This type of transformation is usually required when the variance of the simulations is unusually high. The appropriate transform is normally found by intuition and trial and error. For Apache, taking the natural logarithm of each data point scaled the data appropriately. The resulting diagnostic plots are shown in Figure 3. The residuals now lie close to a straight line, and the residuals vs. predicted response plot shows a more normal distribution with no outliers. All half normal plots shown in Figure 1 use data that has been properly transformed.

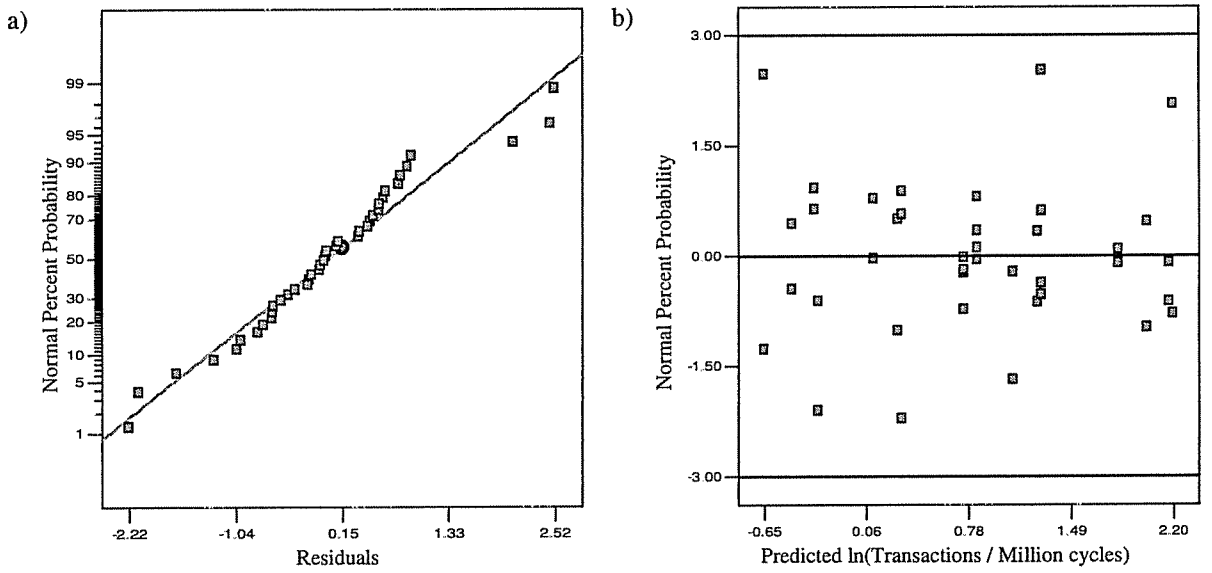

Figure 3: Transformed Normal Plot of Residuals (a) and plot of residuals vs. the predicted response of the model (b) for Apache

Now we can return to the half normal plots of Figure 1 with confidence that the results are meaningful. The first thing to notice is that all of the workloads have the same three most important factors in the same order. These factors also have similar relative magnitudes to the other factors in the plots. The number of cores is by far the most significant contributor to performance for these workloads, which suggests that for our assumptions, putting 16 cores on a single chip will not saturate the available memory bandwidth. Memory bandwidth is not even among the important factors in any of the workloads. Varying the memory bandwidth by a factor of eight did not result in any appreciable gain in performance, so we can conclude that a modest amount of memory bandwidth will adequately support up to 16 processors in a CMP in next generation processes. Part of this result is likely due to the large L2 cache of at least 4 MB in all of the simulations. The L2 cache acts as a filter for the memory interface, smoothing out the bursty behavior of the processor memory accesses and satisfying a majority of the requests.

On the other hand, memory latency comes in second on the list of important factors, and it contributes two to three times as much to performance as the rest of the factors even though it was only varied by a factor of three. Clearly, as processors in a CMP reach higher and higher clock speeds, the memory hierarchy must be designed to keep pace with the processors. Whether this is done by integrating the memory controller on-chip, by including a board level memory-side cache, by designing new and faster memories, or by a combination of these techniques, memory

latency must be addressed. Otherwise, if clock rates continue to scale, CMP performance is likely to suffer the same diminishing returns that are plaguing uniprocessors.

Other significant factors include the L2 associativity and latency, and the pipeline organization. The L2 latency is important for reasons similar to those for memory latency, although the magnitude of its importance is much smaller, making it less critical than memory latency. The L2 associativity is important because a direct-mapped L2 cache will experience contention even when it is only supporting two cores. For the Zeus benchmark, the interaction between the L2 associativity and the number of cores is significant, meaning that as the number of cores increases, the contention in a direct-mapped cache worsens. It is interesting to see that out-of-order cores significantly improve performance for all workloads except OLTP. This result is in agreement with other work showing that OLTP experiences only minor speedups with out-of-order processors [34]. Conversely, the pipeline width, instruction window size, and branch predictor size have an insignificant effect for all workloads. Because these enhancements consume a much larger die area than their performance contribution would justify, allocating the die area to the L2 cache is just as effective at increasing performance with much less design effort. The other insignificant factors would follow similar reasoning.

One final observation pertaining to these results is that the designer can qualitatively and quantitatively assess the variability of the workloads using the half normal plots and ANOVA, respectively. Looking at the plots, a workload exhibits less variability compared to other workloads when the line denoting insignificance has a larger slope relative to the magnitude of its most significant factor, i.e. the line is more vertical on the plot. Thus, Zeus has the smallest variability and OLTP has the largest variability among these workloads.

ANOVA directly computes the variability of a given workload as the pure error of that workload using the design centerpoints. However, it is not necessary to simulate centerpoints to get an approximation of variability. Insignificant factors will still lie along a straight line in the half normal plot so significant parameters can still be selected and used in ANOVA. Then the variability of the workload is computed in ANOVA using the insignificant factors as pure error terms. Because all simulated configurations are used to compute each factor's response, it is unnecessary to run multiple simulations at each data point to measure simulation error. If it is decided that the resulting error is too large, replicating the design will further bound the error by increasing the number of data points used to approximate the simulation error. If the responses of some factors or interactions exhibit less variance relative to the new variance approximation of the design, the the number of significant factors could increase. However, the ranking and relative magnitudes of factors will not change appreciably.

6.2 Refining the Results

Unfortunately, some of the factors that were predicted to be significant turned out to be insignificant. As a result, some of the interactions involving the factors that are significant are aliased with each other. Most notably, the pipeline organization-memory latency interaction (AK) is aliased with the cores-L2 associativity interaction (FL) and the pipeline organization-cores interaction (AF) is aliased with the memory latency-L2 associativity interaction (KL). We could run two additional simulations to remove these aliases, but we would like to investigate two other questions as well. First, we would like to know if there is a major performance improvement between 4-way and 32-

way associativity in the L2 since it is obvious that a direct-mapped L2 is not good enough. Second, we would like to have a more certain interpretation of the significant factors and their interactions. To accomplish these objectives, we can pick the five most important factors for a full factorial design. The results of this design are analyzed in the same way as fractional factorial designs.

Name	Parameter	Low Value (-)	High Value (+)
A	Pipeline Organization	In-Order	Out-of-Order
B	Number of Cores	2	16
C	Memory Latency	150 Cycles	450 Cycles
D	L2 Associativity	4-Way	32-Way
E	L2 Latency	16 Cycles	48 Cycles

Table 9: Full Factorial Design Parameters with high and low values

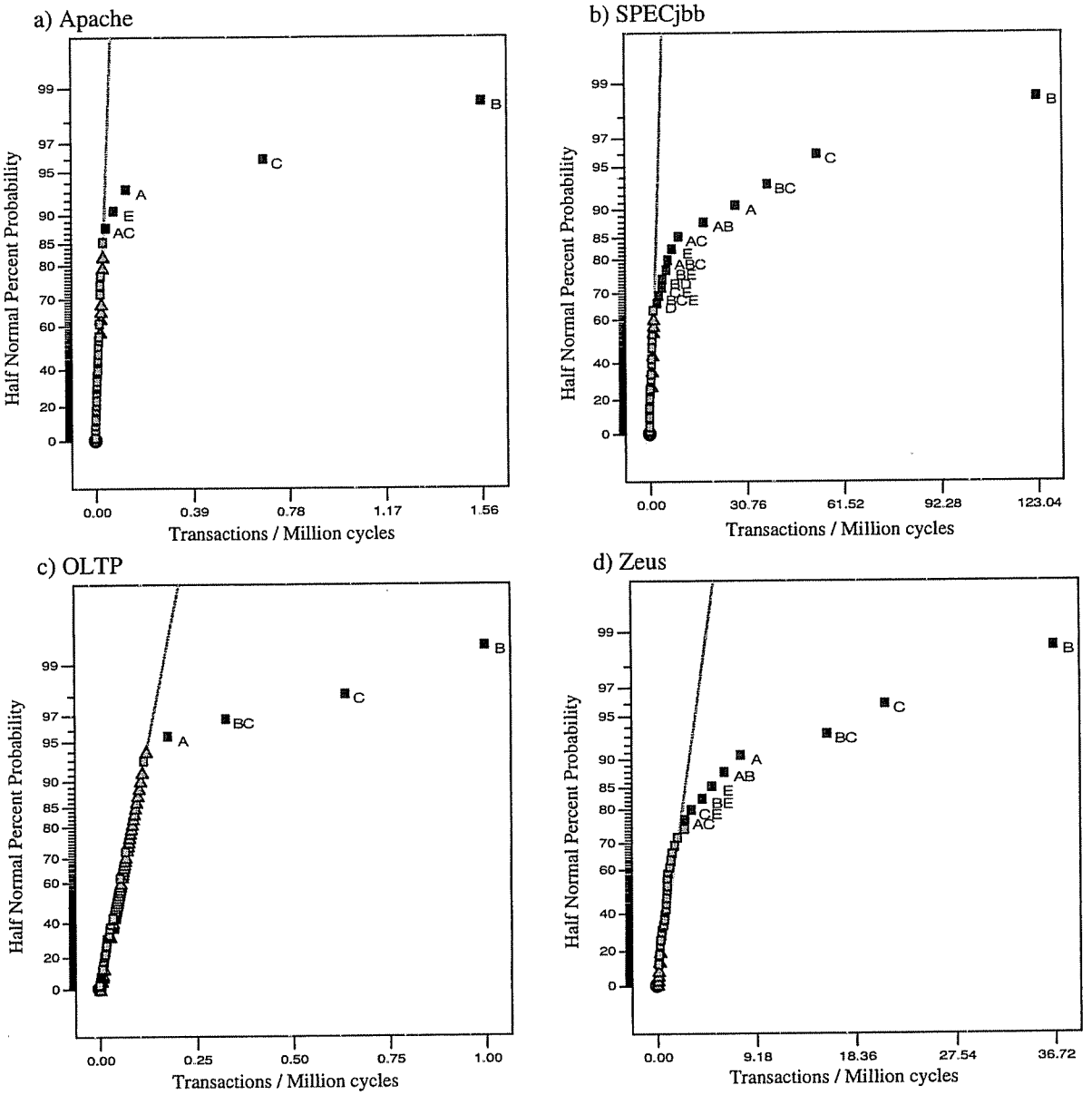

Figure 4: Full Factorial Half Normal Plots of
Apache (a), SPECjbb (b), OLTP (c), and Zeus (d)

The parameters are shown in Table 9 with their associated factor names. All of the other parameters from the previous design are now fixed at the midpoints between their high and low values. Area costs and L2 cache sizes are computed the same way as they were for the previous design, and all configuration permutations were simulated as required for a full factorial design. This design has 32 design runs and 8 centerpoint runs, the same as the previous design, but now all interactions are separated for the factors studied, right up to the five factor interaction ABCDE.

The resulting half normal plots are shown in Figure 4. The OLTP benchmark had such high variability that it was replicated once to produce the plot in Figure 4(c). We can quickly notice that L2 associativity is missing from the significant factors except for SPECjbb where it comes in last among the significant factors. This result implies that a 4-way set associative L2 cache reduces contention enough for CMPs consisting of up to 16 processors, which is not particularly intuitive. The size of the L2 cache may be contributing to the lack of contention seen in the results.

Overall, the results are similar to the fractional factorial design, as they should be. The same factors are significant with similar magnitudes. SPECjbb and Zeus do have a considerable number of significant interactions that were not easily seen in the previous results. Now they are all isolated and it becomes clear that these two workloads are sensitive to subtle interactions between the significant factors. We can use the regression models generated from the significant factors to further explore these interactions.

The best regression models that can be generated from this design are second order models with one second order term. This restriction is due to the choice of only simulating a replicated centerpoint with the fractional factorial design. We do not have enough information to derive multiple second order terms because an intermediate data point is needed for each factor that will have a second order term in the model. Other statistical designs exist to support these stronger models [28], but they are beyond the scope of this work. Nevertheless, insight can still be gained from the restricted models. In particular, the regression model for OLTP produces a good fit for the data.

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F ratio	Probability
Model	35.227	9	3.914	325.70	< 0.0001
A	0.011	1	0.011	0.94	0.3355
B	20.513	1	20.513	1706.94	< 0.0001
C	8.165	1	8.165	679.45	< 0.0001
E	0.095	1	0.095	7.91	0.0064
B^2	3.173	1	3.173	264.02	< 0.0001
AB	0.270	1	0.270	22.45	< 0.0001
AC	0.059	1	0.059	4.95	0.0293
BC	2.231	1	2.231	185.63	< 0.0001
AB^2	0.079	1	0.079	6.60	0.0123
Residual	0.841	70	0.012		
Lack of Fit	0.168	24	0.007	0.48	0.9728
Pure Error	0.673	46	0.015		
Cor Total	36.068	79			
Model Equation:	$1.73 + 0.027*A + 0.566*B - 0.357*C - 0.039*E - 0.498*B^2$ $+ 0.065*A*B - 0.030*A*C - 0.187*B*C + 0.079*A*B^2$				

Table 10: ANOVA results and regression model for OLTP

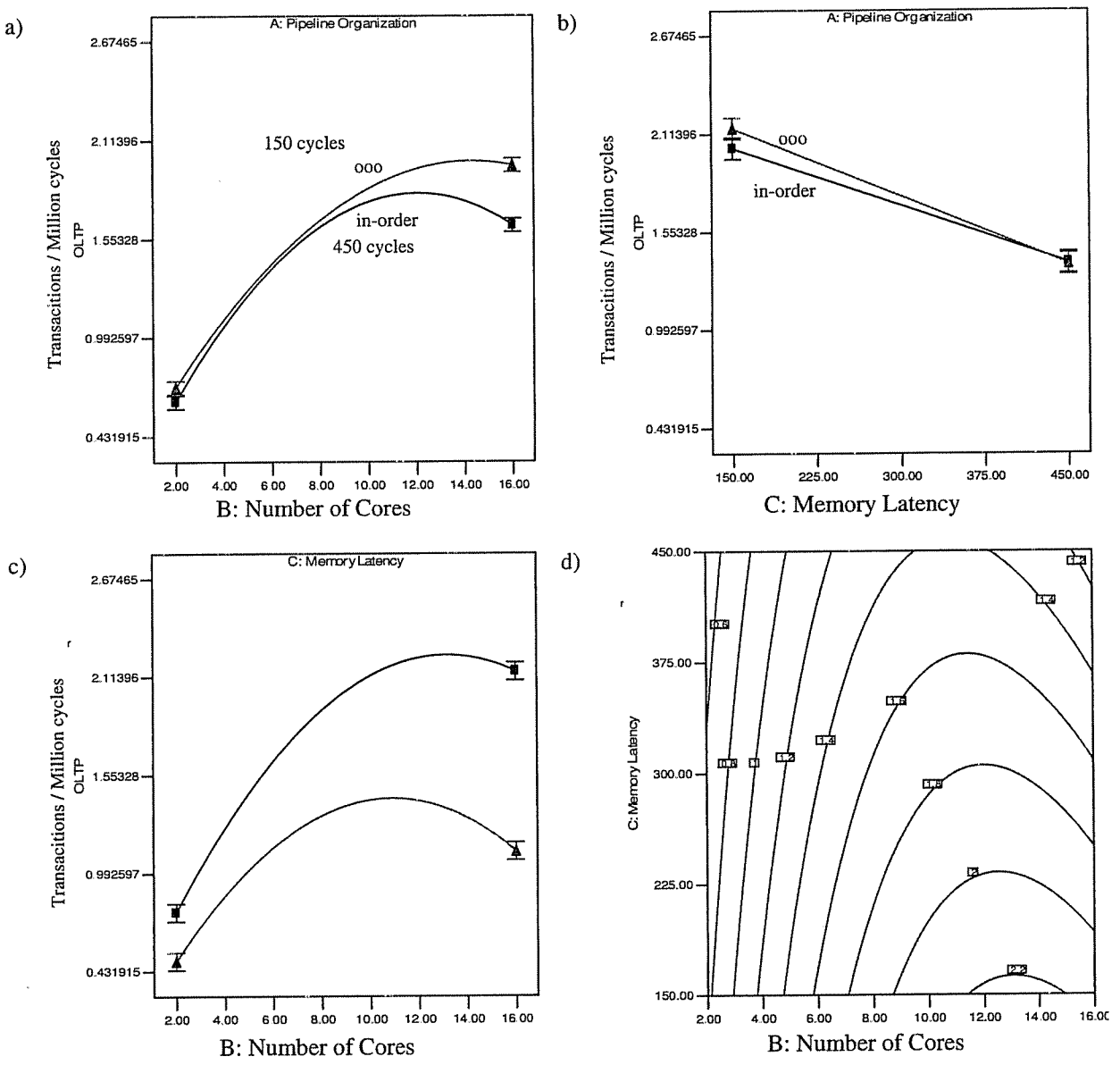

Figure 5: Interaction Plots for pipeline organization vs. number of cores (a), pipeline organization vs. memory latency (b), memory latency vs. number of cores (c), and a contour plot of memory latency vs. number of cores (d) for OLTP.

The regression models for each workload were generated and fit to the data using Design-Expert, and only terms that significantly contributed to the fit of the model were included in the final model for each workload. Table 10 shows the ANOVA results and the corresponding model for OLTP. The model accounts for 98% of the variability in the simulations, and the lack of fit is 97% likely to be insignificant, meaning the model fit is excellent. The pipeline organization does not have a significant effect on performance as shown by its high probability of insignificance, but interaction terms involving this factor are significant so the pipeline organization should be included in the model.

To evaluate the model equation, the value for each factor is cast into the range [0,1] for values selected between the factor's low and high values. However, it is much easier to plot slices of the design space to gain further intuition about the model, instead of evaluating the model as a whole. Figure 5 shows interaction and contour plots for OLTP for number of cores, pipeline organization, and memory latency interactions.

An interaction plot consists of two lines that trace the response of the model as one factor is varied and the other is set at its two extremes. All other factors are held constant. These plots show how much the response of the model changes between the two levels of one factor at different values of the factor it interacts with. Figure 5(a) shows that when there is only a small number of cores on a chip, the choice of in-order cores versus out-of-order cores has little effect on performance, but as more cores are put on chip, it becomes more important to use out-of-order cores. This interaction is intriguing. It is possible that as contention increases in the on-chip interconnect as more cores are added, the in-order cores cannot tolerate the added latency to the L2 cache, something out-of-order processors are much better at handling. Figure 5(b) shows that as the memory latency is increased, out-of-order cores lose their performance advantage over in-order cores. Even though out-of-order cores can handle the added latency to the L2 cache, the magnitude of the memory latency is too great for these cores to tolerate. From Figure 5(c) it is clear that memory latency also has an effect on the optimum number of cores. If the memory latency is high, performance peaks at about 10 cores on chip. Adding more cores actually hurts performance because the L2 cache size is reduced. The smaller L2 cache has a higher miss rate, which increases the average latency to memory. On the other hand, a lower memory latency allows for more cores to be added before performance starts to drop off, and the performance gains from adding more cores is much greater than it was for longer latencies.

Contour plots give another view of a slice of the design space by plotting lines of constant performance, called contours, as two factors are varied between their low and high values. If the contours are close to vertical, the factor on the y-axis does not contribute significantly to performance, and if they are close to horizontal the same is true for the factor on the x-axis. Areas in the contour plot where contours are spaced closer together point to areas in the design space where the performance is changing more drastically with changes in factor values. Figure 5(d) shows a contour plot of memory latency versus the number of cores on chip. Major performance gains are made in moving from two to six cores almost regardless of the memory latency, but as the number of cores continues to increase, memory latency plays a larger role in performance. The same conclusions can be made from interaction and contour plots such as those in Figures 5(c) and (d), yet contour plots allow for more of the design space to be viewed at once since both factors are varying across their ranges.

The interaction and contour plots for the SPECjbb, Apache, and Zeus models all looked similar, so only the most interesting of the SPECjbb plots are presented in Figure 6. The first plot shows the interaction of the memory latency with the number of cores. Having a lower memory latency improves the gains resulting from adding additional cores, and unlike for OLTP, the performance gains do not roll off for a large number of cores. Contention may not be as much of an issue with these workloads as it was with OLTP. We can also compare the contour plots of memory latency versus number of cores for in-order and out-of-order cores. The contour plots in Figures 6(c) and (d) show these results for a fixed contour step size. It is clear that performance increases more rapidly as cores are

added if they are out-of-order cores as opposed to in-order cores, and again, performance gains are not as great for higher memory latencies. Finally, Figure 6(b) shows the interaction between the L2 latency and the number of cores for SPECjbb. The interaction is small and the number of cores has a much greater effect on performance as seen by the almost vertical lines in the plot. However, L2 latency does begin to have a larger effect when 12 to 16 cores are put on-chip, which is likely due to the additional contention in the on-chip network.

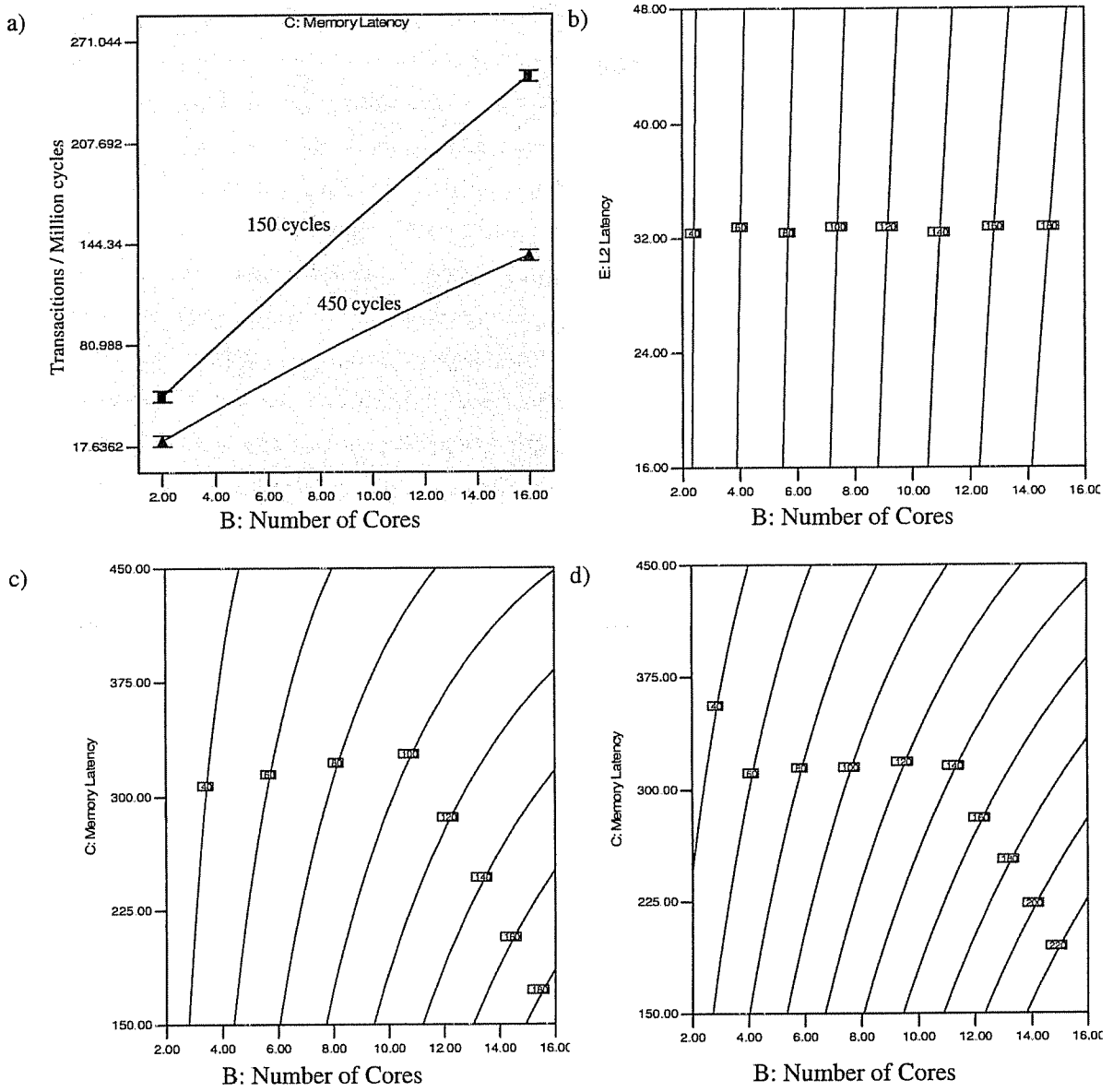

Figure 6: Interaction Plot for memory latency vs. number of cores (a), and contour plots for L2 latency vs number of cores (b), memory latency vs. number of cores for in-order cores (c), and memory latency vs. number of cores for out-of-order cores (d) for SPECjbb

6.3 Summary of Design Space

Through this discussion it has become apparent that only a few parameters and their interactions have a disproportionately large impact on the performance of a CMP. The number of cores that are placed on a single

chip stands out as the most important parameter. The memory latency also has a great impact on performance, and it cannot be ignored even in a CMP. Putting multiple cores on a chip may produce large gains in throughput up to a certain point, but the interaction between the number of cores and the memory latency quickly throttles the performance gains of adding even more cores.

Using out-of-order cores will enhance the throughput of CMP designs utilizing a large number of cores by tolerating the additional latency to the L2 cache banks caused by increased contention. However, going beyond modest out-of-order cores may not be worth the design costs and power consumption because allocating that die area to the L2 cache is just as effective in boosting throughput. The memory bandwidth also appears to be unimportant. Current memory bandwidths can easily support the additional cores that will be added to future CMPs. It appears that the design focus should be on maintaining or reducing the memory latency in order to enable continued scaling of CMPs in the future.

7 Related Work

This work is most closely related to the statistical study done by Yi, et al. [40] for the uniprocessor parameters in the SimpleScalar Toolset. They used a Plackett and Burman (PB) statistical design to rank 43 parameters in order of importance for 13 of the SPEC 2000 benchmarks. The PB design that they used only required 44 simulation runs per benchmark instead of the minimum of 64 that would be required with a fractional factorial design. Unfortunately, the PB design has an extremely complex aliasing structure in which factors can be *partially* aliased with other factors, making the results analysis and design refinement difficult at best. The ANOVA technique used in the fractional factorial analysis could be used on a PB design, although it was not used in their work. Interactions have also been studied previously by Fields, et al. [17]. They develop a framework for quantifying the interaction costs inherent in modern out-of-order superscalar processors to better understand which structures in the processor are responsible for each cycle spent on an executing program.

Statistics has more commonly been used to attempt to build a faster simulator while still retaining as much simulation accuracy as possible. Noonburg and Shen proposed using Markov chains to model the state transitions of a machine model consisting of buffers, pipelines, and other components [29]. The HLS simulator, developed by Oskin, et al. [31] uses runtime statistics gathered from benchmarks to generate synthetic program traces which are then run on a symbolic simulator. The HLS simulator quickly converges on a characteristic IPC for the given benchmark. Eeckhout, et al. [15] expand on the HLS simulator by introducing the use of a statistical flow graph to more accurately capture the dependencies present in a program execution. A related work argues for the use of statistical simulation [16] and quantifies the errors involved in using this tool. All of the above techniques attempt to reduce simulation time by using statistics to reduce the execution time of each simulation run. This work instead proposes a way to reduce the number of simulation runs necessary to make decisions about a design, and in the process increase the designer's intuition about the design space. Also, one disadvantage of statistical simulation is that only closely related architectures can be simulated accurately. Using a statistical design to guide simulation allows the designer to explore a much larger design space.

Numerous studies have also been done on the CMP design space. The oldest on-going research in CMPs is being done in the Hydra group at Stanford [30]. They have focused mainly on a four processor CMP with in-order cores as a platform for exploring thread-level speculation in CMPs [19]. Huh, et al. [20] attempted a more complete exploration of the CMP design space by varying the pipeline organization, L2 cache size and associativity, and number of cores on chip. Die area was restricted in their study, and the implications of finite memory bandwidth were explored, but a statistical methodology was not used so the results were somewhat harder to interpret. The Piranha project [7] took a closer look at one design point in the CMP space with eight in-order cores and a shared, banked L2 cache on-chip. A network interface and router were also integrated on chip to allow larger systems of CMPs to be built in a glueless fashion. This CMP was compared with a monolithic out-of-order processor using the same die area, and the CMP showed greater throughput for commercial workloads in simulation. Kumar, et al. [25] explore the design space of heterogeneous CMPs that integrate some more aggressive and some less aggressive cores on one chip to handle a mixed workload in a computational server environment.

8 Conclusion

Due to the large and growing resource requirements of detailed simulation studies, it has become apparent that more careful design of experiments using simulation will save significant amounts of time in conducting these studies. Fractional factorial designs have been proposed as an existing, powerful methodology for accomplishing this goal while also strengthening the execution and analysis of architectural experiments. Using a rigorous statistical design greatly improves the validity of conclusions reached, enhances intuition about the design space under study, and facilitates a firmer understanding of the interactions in the system that will influence design decisions.

We have applied these principles to the emerging CMP design space to gain a better understanding of the tradeoffs involved in designing a CMP. In the process we have found that memory latency is as much of a factor in CMPs as it has been for uniprocessor design, and we cannot expect the advantages of CMPs to outweigh the necessity of a well-designed memory interface. It is also clear that CMPs produced on 65nm and later processes will easily support up to 16 processor cores on one chip without saturating available memory bandwidth. Finally, we have seen that interactions between the memory latency, number of cores, and pipeline organization have a significant impact on the performance of CMPs, and understanding these interactions will result in more efficient high-performance CMP designs.

References

[1] V. Agarwal, M.S. Hrishikesh, S. W. Keckler, and D. Burger. Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures. In *Proceedings of the 27th International Symposium on Computer Architecture,* 2000.

[2] V. Agarwal, S. W. Keckler, and D. Burger. The Effect of Technology Scaling on Microarchitectural Structures. *UT-Austin Computer Sciences Technical Report TR-00-02*, May, 2001

[3] A. R. Alameldeen, M. M. K. Martin, C. J. Mauer, K. E. Moore, M. Xu, M. D. Hill, D. A. Wood, D. J. Sorin. Simulating a $2M Commercial Server on a $2K PC. *IEEE Computer*, 36(2):50-57, Feb. 2003.

[4] A. R. Alameldeen, C. J. Mauer, M. Xu, P. J. Harper, M. M. K. Martin, D. J. Sorin, M. D. Hill, and D. A. Wood. Evaluating Non-deterministic Multi-threaded Commercial Workloads. In *Proceedings of the Computer Architecture Evaluation using Commercial Workloads,* Feb, 2002.

[5] A. R. Alameldeen and D. A. Wood. Variability in Architectural Simulations of Multithreaded Workloads. In *Proceedings of the 9th International Symposium on High-Performance Computer Architecture,* pages 7-18, 2003.

[6] M. J. Anderson and P. J. Whitcomb. *DOE Simplified: Practical Tools for Effective Experimentation.* Productivity, Inc, 2000.

[7] L. A. Barroso, K. Charachorloo, R. Mcnamara, A. Nowatzyk, S. Qadeer, B. Sano, S. Smith, R. Stets, and B. Verghese. Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing. In *Proceedings of the 27th International Symposium on Computer Architecture,* 2000.

[8] B. M. Beckmann and D. A. Wood. TLC: Transmission Line Caches. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture,* pages 43-54, 2003.

[9] G. E. P. Box, W. G. Hunter, and J. S. Hunter. *Statistics for Experimenters.* John Wiley & Sons, Inc., 1978.

[10] J. L. Devore. *Probability and Statistics for Engineering and the Sciences.* Duxbury, 2000.

[11] H. de Vries. Looking at Intel's Prescott Die, Part II. http://www.chip-architect.com/news/2003_04_20_Looking_at_Intels_Prescott_part2.html, April, 2003.

[12] H. de Vries. Understanding the detailed Architecture of AMD's 64 bit Core. http://www.chip-architect.com/news/2003_09_21_Detailed_Architecture_of_AMDs_64bit_Core.html, Sept, 2003.

[13] Die Photos of Various Processors. http://bwrc.eecs.berkeley.edu/CIC/die_photos/, 2001.

[14] A.N. Eden and T. Mudge. The YAGS Branch Prediction Scheme. In *Proceedings of the 31st Annual IEEE/ACM International Symposium on Microarchitecture,* pages 69-77, 1998.

[15] L. Eeckhout, R. H. Bell Jr., B. Stougie, K. D. Bosschere, and L. K. John. Control Flow Modeling in Statistical Simulation for Accurate and Efficient Processor Design Studies. In *Proceedings of the International Symposium on Computer Architecture,* 2004.

[16] L. Eeckhout, S. Nussbaum, J. E. Smith, and K. De Bosschere. Statistical Simulation: Adding Efficiency to the Computer Designer's Toolbox. *IEEE Micro,* pages 26-38, Sept-Oct 2003.

[17] B. A. Fields, R. Bodik, M. D. Hill, and C. J. Newburn. Using Interaction Costs for Microarchitectural Bottleneck Analysis. In *Proceedings of the 36th International Symposium on Microarchitecture,* 2003.

[18] S. Gupta, S. W. Keckler, and D. Burger. Technology Independent Area and Delay Estimates for Microprocessor Building Blocks. Technical Report 2000-5, Department of Computer Sciences, University of Texas at Austin, April 2000.

[19] L. Hammond, M. Willey, and K. Olukotun. Data Speculation Support for a Chip Multiprocessor. In *Proceedings of the 8th International Converence on Architectural Support for Programming Languages and Operating Systems,* pages 58-69, 1998.

[20] J. Huh, S. W. Keckler and D. Burger. Exploring the Design Space of Future CMPs. In *Proceedings of the 2001 International Conference on Parallel Architectures and Compilation Techniques,* pages 199-210, 20001.

[21] Intel Pentium III Processor Die Photo Fact Sheet. http://www.tomshardware.com/cpu/19990810/, 1999.

[22] The International Technology Roadmap for Semiconductors. http://public.itrs.net, 2003.

[23] C. N. Keltcher, K. J. McGrath, A. Ahmed, and P. Conway. The AMD Opteron Processor for Multiprocessor Servers. *IEEE Micro,* pages 66-76, March-April, 2003.

[24] R. E. Kessler. The Alpha 21264 Microprocessor. *IEEE Micro,* pages 24-36, March-April 1999.

[25] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas. Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance. In *Proceedings of the 31st International Symposium on Computer Architecture,* 2004.

[26] P. S. Magnusson et al. Simics: A Full System Simulation Platform. *IEEE Computer*, 35(2):50-58, Feb. 2002.

[27] C. J. Mauer, M.D. Hill, and D. A. Wood. Full System Timing-First Simulation. In *Proceedings 2002 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems,* pages 108-116, 2002.

[28] D. C. Montgomery. *Design and Analysis of Experiments.* John Wiley & Sons, Inc., 1997.

[29] D. B. Noonburg and J.P. Shen. A Framework for Statistical Modeling of Superscalar Processor Performance. In *Proceedings of the 3rd International Symposium on High Performance Computer Architecture,* 1997.

[30] K. Olukotun, B. A. Nayfeh, L. Hammond, K. Wilson, and K. Chang. The Case for a Single-Chip Multiprocessor. In *Proceedings of the 7th International Symposium on Architectural Support for Programming Languages and Operating Systems,* 1996.

[31] M. Oskin, F. T. Chong, and M Farrens. HLS: Combining Statistical and Symbolic Simulation to Guide Microprocessor Designs. In *Proceedings of the 27th International Symposium on Computer Architecture,* 2000.

[32] S. Palacharla, N. P. Jouppi, and J. E. Smith. Complexity-Effective Superscalar Processors. In *Proceedings of the 24th International Symposium on Computer Architecture,* pages 206-218, 1997.

[33] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice Hall, Inc., 1982.

[34] P. Ranganathan, K. Gharachorloo, S. V. Adve, and L. A. Barroso. Performance of Database Workloads on Shared-Memory Systems with Out-of-Order Processors. In *Proceedings of the 7th International Symposium on Architectural Support for Programming Languages and Operating Systems,* 1998.

[35] STATISTICA: Design of Experiments. http://www.statsoftinc.com/products/doe.html.

[36] J. M. Tendler, J. S. Dodson, J. S. Fields Jr., H. Le, and B. Sinharoy. Power4 System Microarchitecture. *IBM Whitepaper,* 2002.

[37] A. Tucker. *Applied Combinatorics.* John Wiley & Sons, Inc., 2002.

[38] J. H. van Lint. *Introduction to Coding Theory.* Springer Verlag, 1999.

[39] F. Völkel and B Töpelt. Barton's Here: Athlon XP 3000+ vs. P4 3.06 GHz. http://www17.tomshardware.com/cpu/20030210/barton-02.html, Feb, 2003.

[40] J. J. Yi, D. J. Lilja, and D. M. Hawkins. A Statistically Rigorous Approach for Improving Simulation Methodology. In *Proceedings of the 9th International Symposium on High Performance Computer Architecture,* 2003.