Uncore RPD: Rapid Design Space Exploration of the Uncore via Regression Modeling

Abstract

An accurate and efficient regression-based design space exploration methodology is proposed that models the impacts of the memory hierarchy and the network-on-chip (NoC) on the overall chip multiprocessor (CMP) performance. Designers cannot explore all possible designs for a NoC without considering interactions with the rest of the uncore, in particular the cache configuration and memory hierarchy which determine the amount and pattern of the traffic on the NoC. The proposed regression model is able to capture the salient design points of the uncore for a comprehensive design space exploration by designing memory and NoC-specific regression models and leveraging recent advances in uncore simulation. To show the utility of our methodology, two case studies are presented: i) Analyzing and refining regression models for an 8-core CMP and ii) performing a rapid design space exploration to find best performing designs of a NoC-based CMP given area-constraints for CMPs of up to 64 cores. Through these case studies, it is shown that i) Simultaneous consideration of the memory and NoC parameters in the NoC design space exploration can refine uncore-based regression models, ii) sampling techniques must consider the dynamic design space of the uncore, and iii) overall, the proposed regression models reduce the amount of simulations required to characterize the NoC design space by up to four orders of magnitude.

1. Introduction

Given the current trend of an increasing number of cores in chip multi-processors (CMPs), some CMPs have the potential to have over 100 cores in the near term future. Intel has already produced the 80-core Intel Teraflops processor [25], while Tilera has developed a 100-core CMP [24]. As a means to scale performance within power and area budgets for a large number of cores, designers have increasingly relied upon using network-on-chip (NoC) topologies to manage communication traffic between cores. However, efficiently designing NoCs for CMPs with large number of cores is an ongoing problem for system designers [18]. Thus, as a solution to this growing need for effective NoC design techniques, a rapid design space exploration methodology, Uncore RPD, is proposed to pinpoint the best use of hardware resources for NoC-based CMPs.

System designers cannot explore all possible designs for a NoC without considering interactions with the rest of the uncore, in particular the cache configuration and memory hierarchy, which determine the amount of pattern of the traffic on the NoC. A particular measure of the co-dependent impacts of the memory hierarchy and NoC on overall performance is shown in Figure 1. In this motivational experiment on four arbitrarily selected SPLASH-2 benchmarks, the parameters of the NoC, shared caches and application data size are fixed, while the private L1 cache sizes are varied from 32kB to 128kB. In all four of these benchmarks, as the private caches increase, the NoC traffic generated from the application is reduced. Hence, it is essential to model the memory hierarchy in the context of NoC design, as NoC utilization can vary drastically based on application and the memory hierarchy.

The memory hierarchy and NoC design space has grown to over 100 million configurations for homogeneous CMPs. Traditional cycle-accurate simulators, such as `gem5` [4], suffer from large simulation time, which makes it prohibitive to explore the CMP core-uncore design space with millions of configurations. In an effort to quickly simulate designs in this growing design space, researchers have recently discovered efficient techniques to speed up uncore-based simulation [6, 21]. Despite the significant speedup in simulation time, it is computationally infeasible to exhaustively simulate every design with simulation. Thus, it is necessary to apply a technique, such as regression modeling, to characterize the design space with a reduced sample of fast simulations. Statistical regression models have been applied to the single core microarchitecture [9, 16], multiprocessor model [17], shared caches [8, 9, 13], accelerators [20], and GPUs [14]. This paper is the first to accurately and efficiently apply statistical regression modeling [16] to design the memory as well as the NoC design space of the "uncore" in a CMP system. By leveraging advances in the NoC-based CMP simulation through SynchroTrace [21], the proposed statistical regression methods can simulate CMPs up to 64 cores in the order of minutes to 24 hours. By utilizing regression modeling [16], the exploration time of the design space is reduced by up to four orders of magnitude compared to fast, contemporary uncore simulators, with median prediction errors of CMP performance as low as 1.4%. As an added novelty, a low-discrepancy sampling technique, SOBOL, is used in sampling the uncore design space to improve the systematic biases within the model by 4×.

1.1 Contributions

This paper makes the following contributions:

1. Regression modeling strategies are assessed for the memory and NoC design space.
2. Domain knowledge of the memory and NoC co-dependent impacts on performance are applied into the model.
3. An 8-core NoC-based CMP regression model is assessed for accuracy and modeling biases.

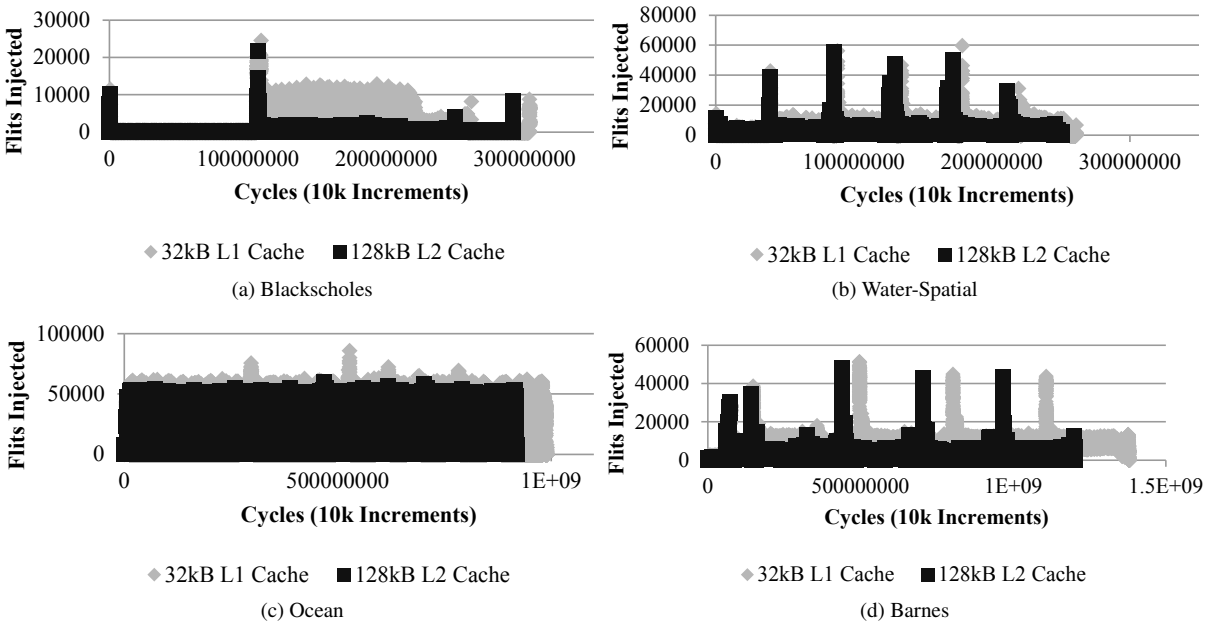


Figure 1. Comparison of NoC traffic, while varying private caches

4. A low-discrepancy sampling technique, SOBOL, is investigated for a more accurate memory and NoC design space exploration.
5. A large scope design space exploration area-constrained case study of up to a 64-core NoC-based CMP is examined, to prove the efficiency of the proposed design space exploration methodology.

2. Background and Related Work

The proposed NoC-based CMP design space exploration methodology includes i) fast simulation of memory and NoC-focused CMP simulation through using SynchroTrace [21], ii) a reduction in the number of simulations required to characterize the design space by leveraging regression modeling strategies by Harrell [10] (introduced to the computer architecture literature by Lee et al [15]), and iii) a low discrepancy sampling technique, SOBOL [23], to aid in the fidelity of the statistical regression models in characterizing the uncore design space of memory and NoC.

2.1 NoC-based CMP Simulation

Enhancements to traditional execution-based simulators to model NoC-based CMPs [4, 6, 11, 12, 21] are a well-tread research area, but these enhancements are not efficient enough for design space exploration. Sniper [6] is a multi-threaded execution-driven simulator that obtains speedup through interval simulation, achieving simulation performance by an order of magnitude over full-system simulators such as Gem5 [4]. SynchroTrace [21] is a trace-driven simulation framework comprised of synchronization-aware traces that drive a timing model, interfaced with the cache and NoC simulators from Gem5 [4], Ruby and Garnet, respectively. The synchronization-aware traces are generated from the native execution of multi-threaded applications. Both Attackboard [12] and Netrace [11] create dependency-tracking network traces for NoC simulators. Although they are fast for benchmarking single design NoCs, these network traces are unreliable for design space exploration of NoCs, as they require multiple full-system simulation runs for their methodology. Additionally, Synfull [2] generates network traces based application and cache coherence behavior. However, the traces are required to be generated per memory configuration, and generating the traces require full-system simulation. In the context of a memory hierarchy and NoC design space of over 100 million configurations in homogeneous CMPs, these execution- and trace-based simulation techniques alone are not scalable. For the proposed design space exploration methodology, SynchroTrace is leveraged for its speedup (up to an 18× speedup over the Gem5 full-system simulator) while maintaining 97% accuracy in design space exploration of the uncore.

2.2 Regression and Computational Modeling Techniques

A recent hot research area has been in modeling techniques to reduce the total number of simulations required to estimate the power and performance of large design spaces. These techniques include statistical regression models [9, 16, 20], 2D wavelet models [8], artificial neural networks [8, 13], and ranking algorithms [7]. Statistical regression models have been applied to the single core microarchitecture [9, 16], multiprocessor model [17], bus-based CMPs [13], shared caches [8, 9, 13], accelerators [20], and GPUs [14]. SVR-NoC [22] leverages a learning-based support vector regression model to compute NoC-specific performance metrics, such as the channel average waiting time and the traffic flow latency. However, this model does not attribute the affects of the memory hierarchy on the NoC. ArchRanker [7] applies a ranking algorithm to find the best design in a design space. Overall, ArchRanker reduces the total number of execution driven simulation samples required to rank the design space in terms of performance. Additionally, ArchRanker only models a design space of up to 8 cores with a fixed NoC interconnect. Varying the NoC parameters, which is a necessity in uncore design space exploration, was not performed in ArchRanker [7].

Predictor	Design Space Predictor Values
Core Count	[2,4,8,16,32,64]
L1 Hit Latency (Cycles)	[2,3,4,5,6]
L1 Cache Size (kB)	[8,16,32,64,128]
L1 Associativity	[2,4,8,16]
L2 Cache Size (kB)	[128,256,512,1024,2048]
L2 Associativity	[2,4,8,16]
NoC Buffer Depth (Flits)	[1,2,3,4,5,6,7,8]
NoC Virtual Channels	[1,2,3,4,5]
NoC Channel Bandwidth (Flits)	[2,4,8,16,32]
NoC 2D Mesh Rows	[1,2,4,8]

Table 1. Memory and NoC Predictors

3. Restricted Cubic Spline Model for NoC-based Design Space Exploration

The overall goal of using the restricted cubic spline models is to characterize the performance of the overall memory and NoC design space using a small set of samples. To create the models for NoC-based CMPs, it is necessary to determine i) the relevant memory and NoC design predictors, ii) the co-dependent relationship between the predictors, iii) the number of samples required to generate the model, and iv) the sampling technique. For reference, the design predictors used in the memory and NoC case studies of Section 5 are shown in Table 1. Prior to presenting a NoC-based CMP restricted cubic spline model, a general restricted cubic spline model and its motivation are described in the next section.

3.1 Overview of Regression Modeling Methodology

Assume there are P predictors (design configuration parameters) describing the uncore design space, with X as the set of independent predictors (also called a vector of predictors). Then, $X = X_1, X_2, \ldots X_P$. Let Y be defined as the response variable to the set of predictors X. In the context of design space exploration, a predictor X_2, for instance, can represent L2 cache size, while Y represents overall system performance. The dependent variable Y is assumed to be constant for a given set of predictors X. β represents a set of regression coefficients (weights) with $\beta = \beta_0, \beta_1, \ldots, \beta_P$.

Thus, for a given observation i, using an initial weighted sum linear regression of the predictor values $X_{i,P}$ of observation i, the predicted value $\hat{y}$ is :

$$\hat{y} = \beta_0 + \beta_1 X_{i,1} + \beta_2 X_{i,2} \ldots \beta_P X_{i,P} \quad (1)$$

$$= \beta^T X_i \quad (2)$$

To fit a linear regression model to all design configuration observations, N, the best-fitting approach can be used by minimizing the sum of squared errors of the actual performance values Y to the predicted performance values Y_i:

$$Error(\beta_0, \beta_1, \ldots, \beta_P) = \sum_{i=1}^{N} (Y_i - \beta_0 - \sum_{j=1}^{P} \beta_j X_{i,P})^2 \quad (3)$$

However, this linear regression model cannot model predictors with nonlinear relationships with the response (i.e. NoC bandwidth to system performance). Polynomial models can be considered but have an undesirable property: an established fit in one region may impact the fit in another region. Thus, to address these issues, restricted cubic spline models [10] are used. Generally, splines are used as a flexible modeling technique to capture piecewise functions throughout the prediction space. The piecewise functions can be linear or model polynomials. The prediction space is divided into intervals by *knots*, k knots $t_1, \ldots, t_k$, which are the endpoints that connect the piecewise functions. The number of knots, k, can be tuned based on the number of available configurations for a given design parameter X; an increase in knots typically leads to a better overall fit to the model. Since the cubic spline models have low modeling accuracy near the tails of the model, restricted cubic spline models, with linear tails, are used for better overall accuracy.

With the regression coefficients $\beta_0, \ldots, \beta_{k-1}$ are estimated, the restricted cubic spline model takes the form of:

$$\hat{y} = \beta_0 + \beta_1 X + \beta_2 (X - t_1)_+^3 + \ldots + \beta_{k+1} (X - t_k)_+^3 \quad (4)$$

Once the regression coefficients are computed, the predicted response (i.e. performance) for a set of design configurations, X, can be computed easily by evaluating the linear Equation 4.

3.2 Co-dependent Relationship between Predictors

The memory and NoC predictor have co-dependent impact on performance, as motivated in the example shown in Figure 1. An example of this includes the interaction of the L1 cache size on the NoC channel bandwidth, for NoC-intensive benchmarks. If these predictors impact on the response cannot be separated, an additional regression coefficient and predictor term must be added. This scenario is defined as an *interaction* [10], with a newly constructed predictor term $X_3 = X_1 X_2$.

The added complexity to the regression model is as follows:

$$\hat{y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$$

To discover the potential memory and NoC co-dependent impacts, a Spearman correlation algorithm is run across the sampled simulation data. This algorithm produces the strength of each predictor in its relationship to performance. If the top 3 strongest overall predictors are a mix of the memory hierarchy and the NoC, these predictors are assigned the co-dependent impacts of the memory hierarchy and NoC in the model creation.

Figure 2 is an example of the Spearman coefficient response of the predictors for the Ocean benchmark. In this example, it is seen that out of the top 3 highest responses, the highest predictor strength to performance relationship is a NoC predictor (NoC channel bandwidth), and the other two of the top three are the L1 cache size, and the number of NoC virtual channels. This high-level memory/NoC relationship is maintained in the proposed restricted cubic spline model by accounting for these top 3 interactions in the model creation for Ocean. The rest of the benchmarks analyzed and presented in Table 2.

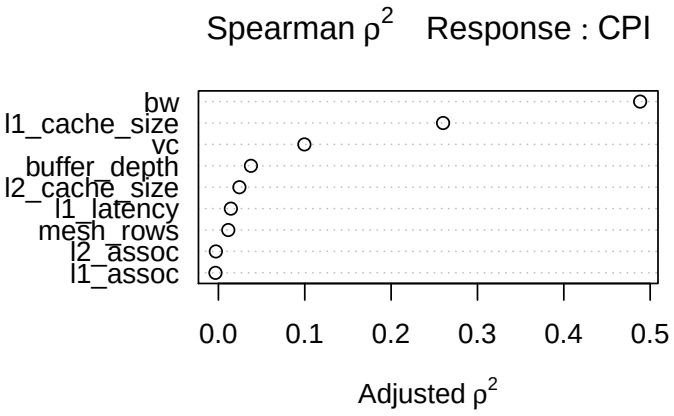


Figure 2. Spearman Correlation of Cache and NoC Predictors of Ocean

Benchmark	Top 3 Predictors and Spearman Coefficient		
Barnes	L1 Latency: 0.554	L1 Cache Size: 0.179	NoC Bandwidth: 0.112
Blackscholes	L1 Latency: 0.833	L1 Cache Size: 0.023	# Virtual Channels: 0.015
Cholesky	L1 Latency: 0.478	NoC Bandwidth: 0.196	L1 Cache Size: 0.100
FFT	L1 Latency: 0.671	NoC Bandwidth: 0.117	# Virtual Channels: 0.033
FMM	L1 Latency: 0.655	L1 Cache Size: 0.117	NoC Bandwidth: 0.078
LU	L1 Cache Size: 0.250	L1 Latency: 0.234	NoC Bandwidth: 0.234
Ocean	NoC Bandwidth: 0.492	L1 Cache Size: 0.264	# Virtual Channels: 0.105
Water-NSquared	L1 Latency: 0.520	NoC Bandwidth: 0.056	L2 Cache Size: 0.049
Water-Spatial	L1 Latency: 0.527	NoC Bandwidth: 0.055	L2 Associativity: 0.047

Table 2. Spearman Correlation: Splash2 and Parsec Benchmarks

3.3 Sampling Techniques

A general sampling strategy posited by Harrell [10] is to sample $20\times$ the number of predictors. The results of the case studies of Section 5 show that a sample size of $20\times$ the number of predictors is largely sufficient for design space exploration of the uncore, as well. In regards to the sampling strategy, the uniform-at-random sampling technique is recommended by Harrell (and adopted by Lee et al. [16]) due to its computational ease and lack of user-induced bias in the model.

However, in the context of memory and NoC design space exploration, uniform-at-random sampling neglects to represent the low cache / varying NoC region with enough samples. Consequently, the regression models constructed with uniform-at-random sampling all have a systematic bias. Hypothetically, this systematic bias will manifest itself as overestimated performance for low performing designs (i.e. low cache / low NoC), as the uniformly sampled space will feature higher cache sizes where sensitivity to NoCs are lower. In effect, this bias renders these uniformly sampled models unusable for design space exploration, as low performing designs may be predicted as having a desirable performance.

To resolve this systematic bias, Uncore RPD leverages a low-discrepancy sampling technique, SOBOL [5, 23]. SOBOL is a deterministic quasirandom sequence that can sample "more uniformly" than uniform-at-random sampling and can be constructed for N-dimensions. The overall benefit with this technique is that the sequence will produce a sample set that can guarantee sampling of the more critical design spaces. Thus, this low-discrepancy sampling technique will resolve the issue of systematic bias in over-predicting performance for poor designs.

4. Experimental Methodology

The simulation platform, experimental methodology, and case studies are introduced in this section. Using the proposed design space exploration methodology, Uncore RPD, 8-core CMP models are constructed and assessed for accuracy and modeling biases. Following this assessment, a regression model is constructed for varying a number of cores to rapidly explore the large uncore design space of CMPs of up to 64 cores.

4.1 Memory/NoC Simulation Platform and Design Space

A number of L1 and L2 cache sizes, associativities, and NoC parameters that include buffer depth, number of virtual channels, channel bandwidth, and 2D mesh dimensions are explored in the design space exploration experiments. The quantitative values of the design space are listed in Table 2. The regression models are generated using the statistical computing tool R. Uncore simulation is performed with SynchroTrace [21]. These predictors model a subset of the wide memory and NoC design space including varying configurations of private and shared caches, as well as the NoC. For design spaces with a fixed core count, there are 1.6 million design configurations available, and 9.6 million design configurations in the design space when cores are varied. The area of these design configurations is calculated with Cacti 6.5 [19] for the caches and Orion 2.0 [15] for the NoC using the 65nm technology. Multi-threaded applications from the Splash-2 [26] and PARSEC-2.1 [3] benchmark suites were used in the design space exploration case studies. The synchronization-aware traces (for SynchroTrace) of these multi-threaded applications were captured on the Linux Kernel 2.6 in CentOS 6 with the standard POSIX Thread API.

4.2 Case Studies Descriptions

Memory and NoC-focused restricted cubic spline models are generated for 8-core NoC-based CMPs and assessed for model accuracy and bias in Section 5.1. Regression models that incorporate a core count as a configuration parameter (predictor) are evaluated in Section 5.2. These regression models are then used to extrapolate a high performing design region, given area constraints from a design space of a CMP of up to 64 cores.

A single model may not be accurate for all benchmarks in each case study as applications individually affect how the memory hierarchy or the NoC may be used. Specifically, each individual application may have a high variety in application characteristics; some benchmarks may have a larger working set than other, contributing pressure onto local cache sizes, while other benchmarks may have poor spatial locality and thus contribute L1 cache misses and NoC traffic. Thus, individual benchmark-specific models are necessary.

5. Experimental Results

In this section, the results of the design space exploration case studies are detailed.

5.1 Regression Model for 8-Core Chip

Regression models are designed and refined for nine 8-thread benchmarks from Splash-2 [26] and Parsec 2.1 [3] across the design space of nine memory hierarchy and NoC predictors. Regression models are created using existing sampling strategies of Harrell [10]. Specifically, the number of samples required for the models are $20\times$ the number of predictors (i.e. 180 samples for the nine predictors examined), and the sampling strategy is uniform-at-random sampling over these predictors. The 180 samples per benchmark are then simulated through SynchroTrace, and individuals models are created for each of the Splash-2 and PARSEC 2.1 benchmarks. The models are assessed for accuracy, in comparison to a validation set of 18 samples and evaluated for systematic model biases.

5.1.1 Regression with Existing Sampling Strategies

The performance prediction error of the nine Splash-2 and PARSEC 2.1 benchmarks are presented in Figure 3. As shown in Figure 3, the model for the blackscholes benchmark is the most accurate, with a median performance prediction error of 1.4%. All of the benchmarks have a median performance prediction of less

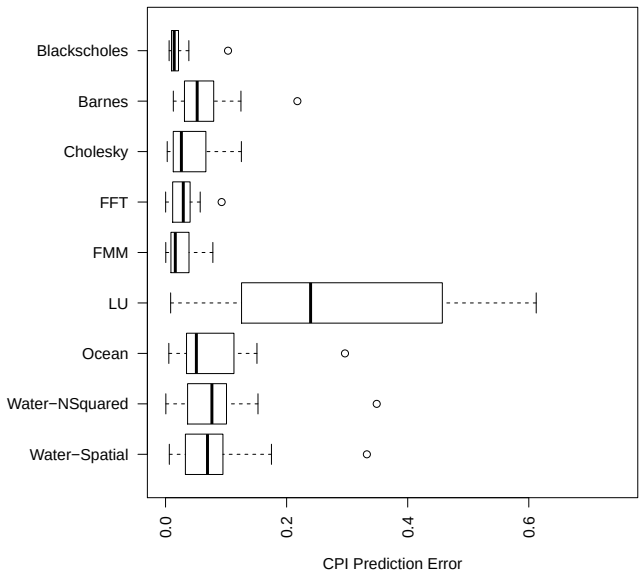


Figure 3. 8-Core Prediction Error without Memory and NoC Interactions

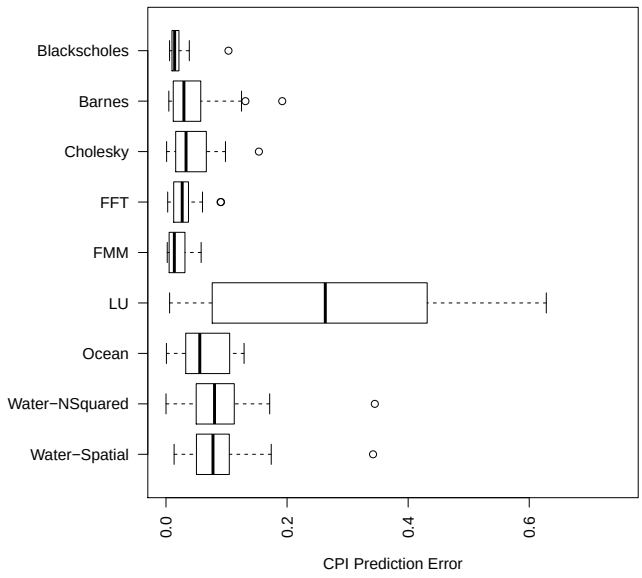


Figure 4. 8-Core Prediction Error with Memory and NoC Interactions

than 6.9%, with the exception of LU. It is evident that the model for the LU benchmark, which has a median performance prediction of 24.0%, has a much higher prediction error than the other eight benchmarks. During the simulations of the LU benchmark, it was discovered that the LU benchmark injects NoC traffic at a much higher rate than the other eight benchmarks: an order of magnitude more than Barnes, FFT, and FMM. Thus, LU has NoC parameters that are more sensitive (compared to the other benchmarks) with respect to performance, and this sensitivity is prevalent only for the small cache design spaces. Thus, the error may justify increasing the number of samples for this benchmark to better represent this critical region. An investigation on the impact of the number of samples on model performance is left to future work.

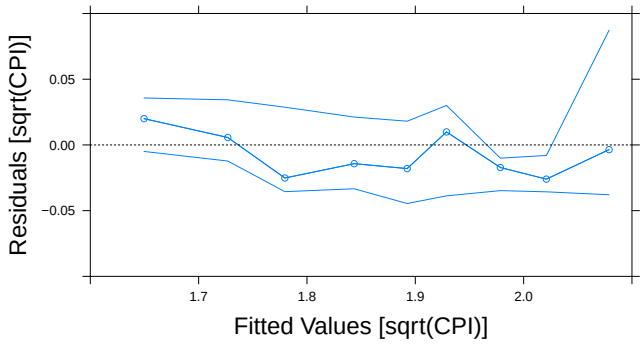
For the two most NoC-intensive benchmarks, Ocean, and LU, modeling the co-dependent impact of the cache and NoC predictors on performance further increases the accuracy of the regression model. As the cache and NoC can have a co-dependent on performance, an additional coefficient must be made explicit to account for this interaction. For reference, the strategies to model the memory and NoC co-dependent impacts on performance is presented in Section 3.2. The prediction error for LU, and Ocean are improved by a notable margin. For LU model, the worst performing model, the median performance prediction error for the LU barely improves to 23.8%. However, the first quartile and third quartile prediction errors reduce from 12.9% to 9.6% and 45.0% to 40.7%, respectively. For the Ocean model, the maximum error reduces from 29.6% to 12.9%, while the median error reduces from 7.5% to 6.1%.

5.1.2 Correcting Model Bias through Low Discrepancy Sampling

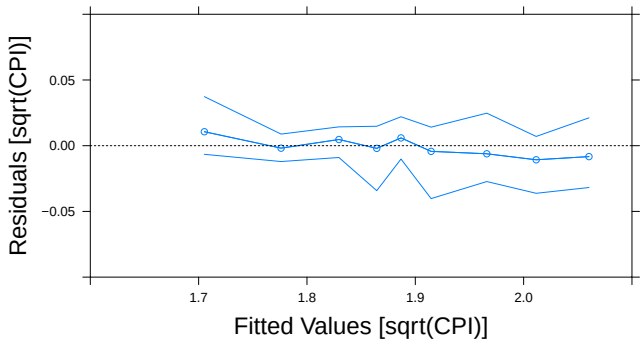
The modeling error of the eight benchmarks, illustrated Figure 4, is sufficient given the range of overall system performance in the total design space. However, a systematic bias in the model must be addressed prior to a design space exploration using the regression models. For all of the benchmarks, the highest variety in outliers or in the 3rd Quartile–max value range are from the same design configurations in the validation set. This error is attributed to over-predicting the performance of the low performing designs (high CPI). It is necessary to remove this model bias, as these models will predict higher performance for low performing design configurations, and as a result, incorrect design configurations may be in the desired high performing subset design space.

Figure 5a illustrates the residuals of the uniform-at-random sampling-based regression model for FFT by grouping the CPI prediction for each of the 180 design points into sets of 20. At each of the fitted values, the residuals are presented in as values: first quartile, median, and third quartile. The residuals of this model show that the designs with the highest CPIs (lowest performance) have a large positive error; in other words, the design configurations with high CPI are overpredicted in performance using the initial regression models. This error can be attributed to the sampling technique leveraged from the work of Harrell [10]. Although uniformly-at-random sampling reduces the chance of a user-generated bias for the model, this assertion does not hold true for general model bias when modeling a memory and NoC. Specifically, for the design space in which the memory is small, the resulting NoC traffic will be increased. This design space will feature very sensitive NoC predictors, whereas if the memory subsystem were of high resources, the NoC would be under-utilized. It is important to sample this region adequately, as a model may not be able to predict the performance of this design space purely from the samples of design space medium to larger memory systems. As a whole uniformly-at-random sampling may have higher densities of designs of different regions of the design space but may miss sampling the "small-cache" region fully. Thus, to reduce this potential model bias, it is essential to use a sampling technique that covers the whole design space: low-discrepancy sampling.

To reduce the model bias, a form of low-discrepancy sampling, SOBOL, is leveraged for the model creation in order to "more uniformly" sample the design space as described in Section 3.3. As shown in Figure 5b, the high-CPI model bias in the uniform-at-random sampling is corrected with SOBOL sampling for FFT, with a 4× improvement in residual error for the highest CPI designs sampled.



(a) FFT - UAR



(b) FFT - SOBOL

Figure 5. CPI Residuals of FFT with Uniform-at-Random and SOBOL Sampling [Top Line - 3rd Quartile, Middle Line - Median, Bottom Line - 1st Quartile]

5.2 Area-Constrained Design Space Exploration of Large Multi-threaded Programs

In this section, a case study on area-constrained design space exploration of a wide memory and NoC space with a varied number of cores is explored. The overall goal of this experiment is to determine a design subspace that encompasses the highest performing designs given uncore area constraints. Within this design subspace, there are designs that sacrifice area and/or power for increased performance. This case study explores the flexibility of varying the number of cores, and thus seeks to discover if fewer cores with a larger uncore or a large number of cores with a smaller uncore is optimal.

This case study examines the design configurations in Table 1, including a varying number of cores from two to 64. The area constraint is based on the area of the Intel SCC [1]: 576mm^2. To assess the uncore, a constraint of 50% of the total chip area is designated for the uncore.

Regression models are developed sampling 180 designs over the entire design space, including the varied number of cores. For this experiment, the number of rows on the 2D mesh of the NoC are kept fixed to a respective core count, i.e. 64 cores has an 8 mesh rows, while 4 cores have 2 mesh rows. Thus, nine design parameters are investigated, with a total number of design configurations reaching 2.4 million. The prediction accuracy of the regression model is detailed in Figure 6. It is notable that the median error of all of the benchmarks are less than 10% with a third quartile error of approximately 20% or less.

Benchmark	Size of Optimal Performing Design Subspace
Water-Spatial	1260
FMM	3063
LU	145
FFT	1418
Cholesky	917

Table 3. Size of Optimal Design Subspace of Selected Benchmarks

5.2.1 Optimal Design Space with an Error Margin

To discover the optimal design in the design space, all of the area values of the 2.4 million design configurations are computed and filtered through the area constraint (50% of 576mm^2). The remaining designs are processed through the regression model as to predict the performance of each design configuration, including the highest performing design configuration. A design space of interest can be constructed by discovering the optimal performing design and the design configurations that fall within the regression model error margin of the best design. Thus, the amount of prediction error for each model affects the size of the targeted subspace containing the optimal design configurations under area constraints. For example, the model for LU has a third quartile error of 18.2%. The best predicted design for LU in this design space is a 16-core CMP with a 128kB L1 cache, 256kB L2 cache, and a relatively large NoC with a 16 byte channel length. The performance of the optimal design is predicted to be 1.5 CPI. The designs that are within 18.2% of 1.5 CPI are added to the design space of interest. Overall, the proposed regression modeling methodology enables a filtered optimal design space, on the order of 100s to 1000s of design configurations, to be discovered out of the total 2.4 million design configurations. Table 3 summarizes the sizes of the design space of interest for five benchmarks.

5.2.2 Analyzing the Area-Constrained Design Space

For this case study, the prediction results of the LU and Cholesky benchmarks are examined in the context of this area-constrained large design space exploration. The proposed methodology, Uncore RPD, enables designers to rapidly visualize the uncore design space. Figure 7 illustrates the design space for LU given the area constraint; every design under the area constraint is evaluated for predicted performance and is categorized by number of cores of the simulation. Figure 8 illustrates the design space for LU with a categorization of total cache area. Through these two figures, it is evident that i) all of the optimal designs (i.e. within the error margin of the highest performing design) are of 16 cores for a design space of up to 64 cores, and ii) the optimal designs contain cache sizes varying between 100mm^2 and 200mm^2 with varying NoC resources. This rapid design space exploration describes a relatively heterogeneous result for Cholesky: i) as shown in Figure 9 the optimal design region is dominated by a mix of 8-core CMPs and 16-core CMPs, and ii) by correlating the cache size to the optimal performing designs, Figure 10 depicts that highest performing designs have total cache sizes between 100mm^2 and 200mm^2 with varying NoC resources.

In the context of 2.4 million design configurations in a subset of the uncore design space, it is infeasible to characterize the entire design space solely using simulation. Overall, as presented in this case study, the proposed regression modeling methodology, Uncore RPD, enables designers to rapidly investigate the large uncore subspace by reducing the amount of simulations required to characterize the design space by over four orders of magnitude.

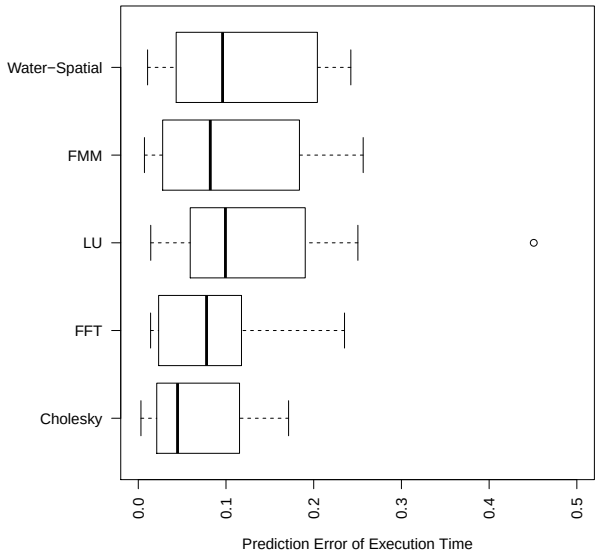


Figure 6. Regression Modeling Error for Varying Number of Cores

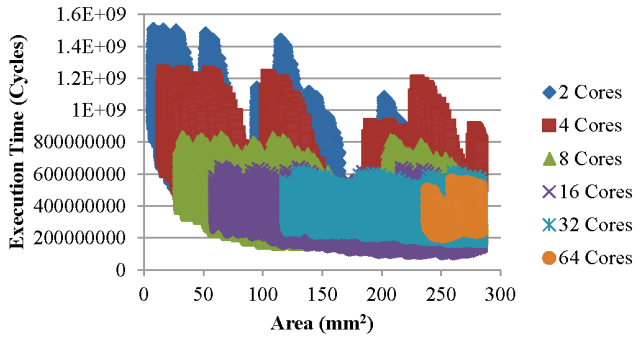


Figure 7. Filtered Design Space for LU using Regression Modeling

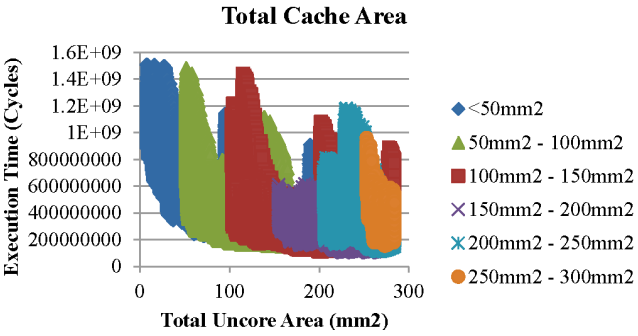


Figure 8. Cache Area of Filtered Design Space for LU using Regression Modeling

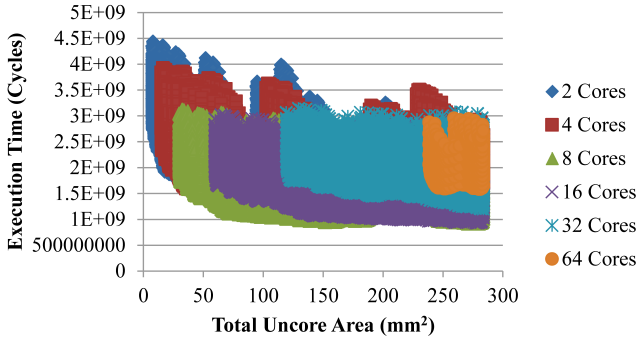


Figure 9. Filtered Design Space for Cholesky using Regression Modeling

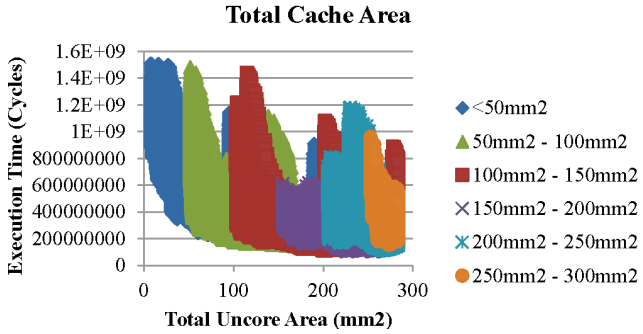


Figure 10. Cache Area of Filtered Design Space for Cholesky using Regression Modeling

5.2.3 Simulation Statistics

The compute time required to perform the area-constrained design space exploration case study is presented in Table 4 for Gem5 [4], SynchroTrace [21], and the proposed Uncore RPD. This table aggregates the total compute time for capturing traces, simulation, and time associated with regression modeling to characterize the design space of 2.4 million uncore design configurations. This time can be divided up by the number of simulations and compute nodes available in a research computing cluster. Design space exploration with Gem5 does not require traces or further modeling, but the total compute time is on the order of 1000 years. SynchroTrace requires generating traces once for each thread count (i.e. a trace for each 2, 4, 8, 16, 32, and 64 threads per benchmark) and has a total compute time on the order of 100 years. Uncore RPD requires the same trace generation step to leverage SynchroTrace, but the proposed methodology can characterize the design space sufficiently with only 180 designs and minimal compute time for model creation and prediction. In practice, using the previously generated traces, the Uncore RPD methodology was executed fully on LU benchmark in approximately an hour to simulate the samples and generate the model using a cluster with 2432 compute cores.

6. Conclusions

In this work, Uncore RPD: a rapid regression-based uncore design space exploration methodology is presented. Effective regression modeling reduces the number of simulations required to describe the design space of the memory and NoC. The proposed regression model is able to capture the salient design points of the uncore for a comprehensive design space exploration by extending previous work on microarchitecture-focused regression models and recent

Simulation Methodology	Trace Capture Time	Simulation Time	Model Creation and Prediction Time	Total Time
Gem5	N/A	1370 years	N/A	1370 years
SynchroTrace	21 hours	274 years	N/A	274 years
Uncore RPD	21 hours	180 hours	5 seconds	207 hours

Table 4. Design Space Exploration Total Compute Time for LU

advances in uncore simulation. SOBOL, a low-discrepancy sampling technique, is implemented to reduce systematic bias, and the model is further refined through modeling the co-dependent impact of the memory and NoC on overall system performance. To show the utility of our methodology, two case studies are presented: i) Analyzing and refining regression models for an 8-core CMP and ii) Performing a rapid design space exploration to find best performing designs of a NoC-based CMP given area-constraints for CMPs up to 64 cores. Through these case studies, it is shown that i) Simultaneous consideration of the memory and NoC parameters in the NoC design space exploration can refine uncore-based regression models, ii) Sampling techniques must consider the dynamic design space of the uncore, and iii) Overall, the proposed regression models reduce the amount of simulations required to characterize the NoC design space by up to four orders of magnitude.

References

[1] Intel single-chip cloud computer. http://www.intel.com/.

[2] M. Badr and N.E. Jerger. Synfull: Synthetic traffic models capturing cache coherent behaviour. In *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pages 109–120, June 2014.

[3] C. Bienia and K. Li. Parsec 2.0: A new benchmark suite for chip-multiprocessors. In *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, June 2009.

[4] N. Binkert et al. The gem5 simulator. In *The ACM SIGARCH Computer Architecture Newsletter*, August 2011.

[5] Paul Bratley and Bennett L. Fox. Algorithm 659: Implementing sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 14(1):88–100, March 1988.

[6] T. E. Carlson, W. Heirman, and L. Eeckhout. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, November 2011.

[7] Tianshi Chen, Qi Guo, Ke Tang, O. Temam, Zhiwei Xu, Zhi-Hua Zhou, and Yunji Chen. Archranker: A ranking approach to design space exploration. In *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pages 85–96, June 2014.

[8] Chang-Burm Cho, J. Poe, Tao Li, and Jingling Yuan. Accurate, scalable and informative design space exploration for large and sophisticated multi-core oriented architectures. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2009. MASCOTS '09. IEEE International Symposium on*, pages 1–10, Sept 2009.

[9] P. Hallschmid and R. Saleh. Fast design space exploration using local regression modeling with application to asips. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(3):508–515, March 2008.

[10] Frank E. Harrell, Jr. *Regression Modeling Strategies*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[11] J. Hestness, B. Grot, and S. W. Keckler. Netrace: dependency-driven trace-based network-on-chip simulation. In *Proceedings of the International Wokshop on Network on Chip Architectures (NoCArc)*, pages 31–36, 2010.

[12] Y. S.-C. Huang, Y.-C. Chang, T.-C Tsai, Y.-Y Chang, and C.-T King. Attackboard: A novel dependency-aware traffic generator for exploring NoC design space. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 376–381, 2012.

[13] Engin İpek, Sally A. McKee, Rich Caruana, Bronis R. de Supinski, and Martin Schulz. Efficiently exploring architectural design spaces via predictive modeling. *SIGPLAN Not.*, 41(11):195–206, October 2006.

[14] Wenhao Jia, K.A. Shaw, and M. Martonosi. Stargazer: Automated regression-based gpu design space exploration. In *Performance Analysis of Systems and Software (ISPASS), 2012 IEEE International Symposium on*, pages 2–13, April 2012.

[15] A. B. Kahng, B. Li, L. Peh, and K. Samadi. ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration. In *Proceedings of the Design, Automation Test in Europe (DATE)*, April 2009.

[16] Benjamin C. Lee and David M. Brooks. Statistically rigorous regression modeling for the microprocessor design space. In *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2006.

[17] Benjamin C. Lee, Jamison Collins, Hong Wang, and David Brooks. Cpr: Composable performance regression for scalable multiprocessor models. In *Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 41, pages 270–281, Washington, DC, USA, 2008. IEEE Computer Society.

[18] R. Marculescu, U. Y. Ogras, P. Li-Shiuan, N. E. Jerger, and Y. Hoskote. Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives. *The IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,*, 28(1):3–21, January 2009.

[19] N. Muralimanohar, R. Balasubramonian, and N. Jouppi. Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2007.

[20] K. Nepal, O. Ulusel, R.I. Bahar, and S. Reda. Fast multi-objective algorithmic design co-exploration for fpga-based accelerators. In *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*, pages 65–68, April 2012.

[21] S. Nilakantan, K. Sangaiah, A. More, G. Salvador, B. Taskin, and M. Hempstead. Synchrotrace: Synchronization-aware architecture-agnostic traces for light-weight multi-core simulation. In *To Appear in Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2015.

[22] Zhiliang Qian, Da-Cheng Juan, Paul Bogdan, Chi-Ying Tsui, Diana Marculescu, and Radu Marculescu. Svr-noc: A performance analysis tool for network-on-chips using learning-based support vector regression model. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, pages 354–357, March 2013.

[23] I. M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *Computational Mathematics and Mathematical Physics*, 7(4):86+, 1967.

[24] Tilera Tile Gx-100. online. http://www.tilera.com.

[25] S. Vangal et al. An 80-tile 1.28tflops network-on-chip in 65nm cmos. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, pages 98–589, February 2007.

[26] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 programs: characterization and methodological considerations. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 1995.