THE INSTITUTE FOR SYSTEMS RESEARCH

ISR TECHNICAL REPORT 2015-06

Physically Constrained Design Space Modeling for 3D CPUs

Caleb Serafy

Physically Constrained Design Space Modeling for 3D CPUs

Caleb Serafy, Ankur Srivastava and Donald Yeung
University of Maryland
College Park, MD, USA
{cserafy1, ankurs, yeung}@umd.edu

ABSTRACT

Design space exploration (DSE) is becoming increasingly complex as the number of tunable design parameters increases in cutting edge CPU designs. The advent of 3D integration compounds the problem by expanding the architectural design space, causing intricate links between memory and logic behavior and increasing the interdependence between physical and architectural design. Exhaustive simulation of an architectural design space has become computationally infeasible, and previous work has proposed fast DSE methodologies using modeling or pseudo-simulation.

Modeling techniques can be used to predict design space properties by regression fitting. However in the past such techniques have only been applied to optimization metrics such as performance or energy efficiency while physical constraints have been ignored. We propose a technique to apply spline modeling on a 3D CPU design space to predict optimization metrics and physical design properties (*e.g.*, power, area and temperature). We use these models to identify optimal 3D CPU architectures subject to physical constraints while drastically reducing simulation time compared to exhaustive simulation. We show that our technique is able to identify design points within 0.5% of the global optimal while simulating less than 5% of the design space.

1. INTRODUCTION

Design space exploration (DSE) involves the evaluation of a multitude of design choices before detailed implementation. Such a technique is necessary to make optimal design choices and perform educated tradeoff analysis of conflicting objectives. In its simplest form DSE can be performed by exhaustively simulating the entire design space and choosing the optimal design. However as CPU designs become ever more complex in the pursuit of Moore's law performance scaling, the DSE problem has become increasingly intractable as the design space becomes exponentially large. It is simply no longer computationally feasible to exhaustively simulate the cross product of all design variables.

Past work has attempted to overcome the computational infeasibility of exhaustive simulation in two ways. One is to reduce simulation time by orders of magnitude using techniques such as host-compiled simulation [1] or statistical simulation [2]. Although these approaches can make exhaustive simulation possible, the accuracy of such fast simulation techniques is reduced, and the applicability of the techniques is limited in scope. Another approach to the DSE problem is to simulate only a small subset of the full design space and use modeling techniques to predict the properties of unsimulated designs. Modeling approaches [3, 4, 5, 6, 7] have shown promising results on large architectural design spaces.

Vertical integration of circuits (3D ICs) is an up-and-coming technology that shows great potential towards improving circuit power and performance as well as facilitating new CPU architecture paradigms such as stacked memory and highly connected on-chip networks. However 3D ICs also bring new challenges, chief among them thermal management. Moreover, past work [8, 9] has shown that 3D-CPU architectural design choices have a profound impact on physical properties such as power, area and temperature and significant portions of the 3D CPU design space can be infeasible due to physical constraint violations. 3D integration significantly complicates the DSE problem as follows:

- 3D integration brings many new architectural opportunities that significantly compound the intractability of exhaustive simulation.
- 3D ICs are more thermally sensitive to architectural changes than equivalent 2D chips due to their physical structure.
- 3D ICs can eliminate communication bottlenecks that are inherent in 2D ICs, making performance and power more sensitive to architectural changes.
- *Ad hoc* fixes late in the design cycle to fix infeasible or suboptimal architectural design choices choices can be more costly in 3D ICs due to a higher interconnectivity and density of circuit components.

Physical awareness during DSE is becoming more important, especially in the context of 3D ICs. Past work [10, 9] has examined the effect of physical constraints

on DSE, but has only done so with exhaustive simulation of a small design space. Previous work on design space modeling [3, 4, 5] has simply attempted to predict high performance or energy efficient architectural designs with no consideration as to whether such designs were physically feasible. In this paper we introduce a spline regression modeling technique for 3D CPUs which models physical properties (*e.g.*, power, area and temperature) along with traditional optimization metrics such as instructions per second or energy-delay-product. Our models are able to predict the performance and temperature of a diverse 3D CPU design space and leads us to identify a thermally feasible solution within 0.5% of the optimal solution while simulating less than 5% of the design space.

The rest of this paper is organized as follows. In the remainder of this section we give a detailed overview of related work and enumerate the contributions this work makes to the research effort. Section 2 explains our simulation framework which is used to evaluate the performance, power, temperature and area of 3D CPU. Section 3 introduces our modeling approach for estimating optimization metrics and physical properties of the entire design space while only simulating a small subsection of the space. Such an approach is necessary to make the physically constrained DSE problem computationally feasible. Section 4 explains the experimental setup of our study, and Section 5 presents the results which demonstrate the effectiveness and accuracy of our DSE modeling algorithm. Finally, Section 6 concludes the paper.

1.1 Previous Work

As CPU design space has become increasingly large, exhaustive simulation has become computationally infeasible. Methodologies to facilitate large scale DSE have taken two orthogonal approaches: drastically reduce simulation time or produce models of unsimulated design points using simulation data from a small subset of the design space. The work by Genbrugge and Eeckhout [2] attempts to massively reduce simulation time with statistical simulation, which entails constructing a short code sequence that is representative of a full workload. Other work by Gandhi *et al.* [1] uses host-compiled simulation, which natively executes workloads that have been annotated with performance and power data generated offline using system models. Both techniques massively reduce simulation time, but at the cost of reduced accuracy and limited applicability.

Design space modeling likewise trades off accuracy for increased simulation time by omitting simulation of certain design points and estimating those points using modeling techniques. However, a significant advantage of modeling approaches over statistical or host compiled simulation is the ability to maintain accuracy in regions of interest in the design space (*e.g.*, close to the optimal design) by clustering more simulation data in that region. This is important because it is often the case that accuracy of the simulations is only important in a small subset of the design space: the region around the optimal points. Different modeling techniques have been proposed to accurately estimate the properties of a design. Early work by Joseph *et al.* used linear regression to model instructions per cycle (IPC) across a 23-variable CPU design space. However only two factors of each variable were considered, and the accuracy of the generated models was not reported. Later that year two similar works by Lee and Brooks [6] and İpek *et al.* [5] applied spline regression and artificial neural network models to similar problems, yielding average errors less than 10%, although maximum error was around 50%. More recent work by Jia *et al.* [4] and Wang [3] applied spline regression to GPUs and multi-core CPUs respectively. These techniques reduced maximum error to around 15% and had average errors in the single-digits range.

A limitation on most of the previous work on design space modeling is that it only presents models for performance. Exceptions to this trend are [6] and [3] which also model power and energy respectively. However the power and energy estimates in those papers are simply used to generate Pareto optimal tradeoff curves and are not used to model physical constraints on the system.

Much past work has demonstrated the interaction between performance and physical feasibility. An early work by Cong *et al.* [11] demonstrated the need for simultaneous consideration of both architectural (*e.g.*, cycles per instruction) and physical (*e.g.*, critical path delay) design metrics to accurately judge performance. Subsequent work by Li *et al.* [10] and Serafy *et al.* [9] showed that the imposition of physical constraints on a CPU design space significantly affected feasible performance and substantially changed the nature of the optimal architecture.

However, these studies illustrated the need for physically aware DSE using exhaustive simulation of a small design space. To the best of our knowledge no previous work has presented a technique for modeling the physical feasibility region in a realistic many-dimensional design space. Moreover to the best of our knowledge no previous work has done modeling-based DSE for 3D CPUs. In this paper we present a modeling approach to estimate optimization metrics and the physical properties of a 3D CPU design space, and show that only with such models can one identify architectural design points that are optimal subject to physical constraints.

1.2 Contributions

This paper makes the following contributions:

- We propose a design space modeling technique that builds regression models to predict optimization metrics and physical properties of an entire architectural design space while only simulating a small subset of the space.
- To the best of our knowledge our work is the first to apply design space modeling techniques to 3D CPUs.
- To the best of our knowledge our work is the first to apply design space modeling to physical proper-

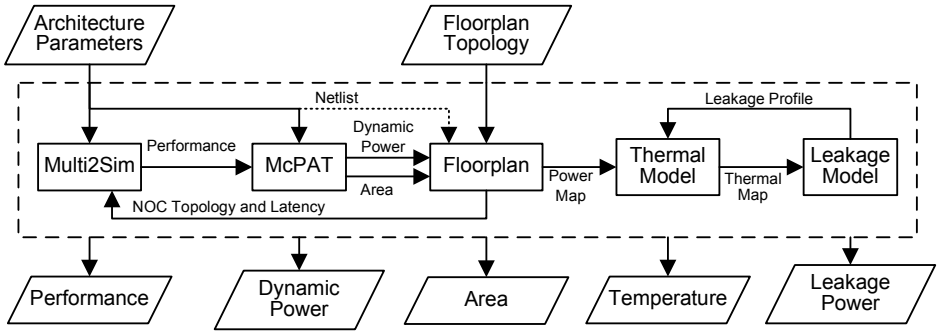


Figure 1: 3D CPU simulation flow

ties such as temperature to predict the feasibility region of a design space.

- We show that physical constraints fundamentally change the nature of the optimal design point, and without proper models of these constraints infeasible designs would be deemed optimal.

2. 3D CPU SIMULATION FLOW

The simulation flow used to evaluate power, performance, area and temperature of a specific 3D CPU architecture is shown in Figure 1. The details of each component of the simulation flow are explained in the following subsections.

2.1 Performance Simulation

Performance simulation is performed by Multi2Sim (M2S) [12], a cycle accurate CPU simulator. Architectural definitions are passed to the simulator through configuration files that describe parameters such as: number of cores, number of function units within cores, pipeline width, buffer and queue lengths, register file size, cache size and associativity, cache latency, network-on-chip (NOC) topology and latency, branch predictor size and type *etc.* Cache and table (*e.g.*, register file, register alias table (RAT) and branch target buffer) latencies are determined using CACTI [13] to provide realistic architectural setups to the simulator. DRAM latency is calculated using the model proposed in [8]. NOC topology and latency is calculated as explained in Section 2.3.1. M2S simulates the execution of a provided x86 binary file on the described CPU. The simulator outputs a list of performance statistics such as IPC, DRAM and cache reads, writes, hits and misses, branch prediction rate, number of instructions that access each type of execution unit, reads and writes to buffers, queues and RAT *etc.* The specific architectural configurations and software workloads simulated for this study are discussed in Section 4.

2.2 Power and Area Estimation

Power and area estimation is performed by McPAT [14], a multi-core power, area and timing estimation tool. Based on the architectural definition, the area and energy per access of each CPU component is estimated

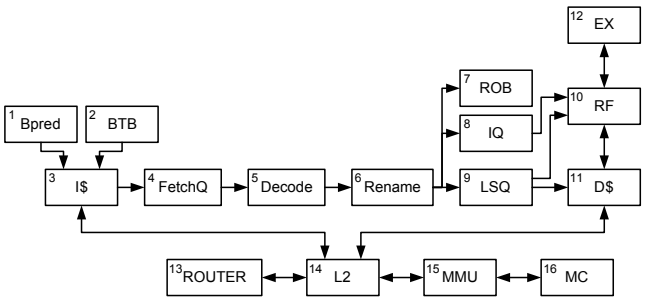


Figure 2: Core component netlist

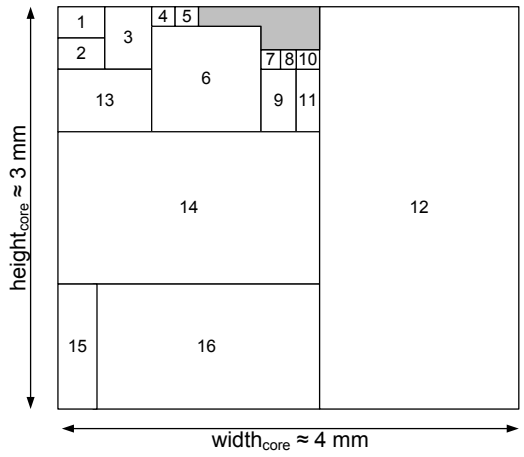
for a selected technology node. Using the performance statistics generated by M2S, the power dissipation of each CPU component can be estimated by multiplying energy per access by number of accesses.

Note: CPU components such as cache may have different types of accesses (*e.g.*, read and write) which use different amounts of energy. Different types of accesses are given unique energy estimates by McPAT and reported separately by M2S.

2.3 Floorplan

Each core is partitioned into a set of components (*e.g.*, data cache, RAT, branch predictor, instruction queue and execution unit) which are interconnected by the abstract netlist shown in Figure 2. Detailed descriptions of each component can be found in [14]. A general floorplan (FP) topology was generated offline which minimizes the wirelength between connected components. The general FP topology is shown in figure Figure 3.

FP topologies are represented using transitive closure graphs (TCGs) [15] which define a unique floorplan when combined with area and aspect ratio data for each component (components are assumed to be rectangular). Areas of each component are unique to a given architectural design point, and are generated by McPAT (Section 2.2). Aspect ratio is optimized for each design using a simulated annealing scheme that minimizes core area and wirelength. Core width and height are denotated as $width_{core}$ and $height_{core}$ respectively.



ID#	Name	Description
1	Bpred	Branch Predictor
2	BTB	Branch Target Buffer
3	I$	L1 Instruction Cache
4	FetchQ	Fetch Queue
5	Decode	Decoder
6	Rename	Rename Unit
7	ROB	Reorder Buffer
8	IQ	Instruction Queue
9	LSQ	Load-Store Queue
10	RF	Register File
11	D$	L1 Data Cache
12	EX	Execution Unit
13	ROUTER	NOC Router
14	L2	L2 Cache
15	MMU	Memory Management Unit
16	MC	Memory Controller[1]

Figure 3: Core floorplan topology

2.3.1 Core Tiling and NOC Design

The core floorplan is replicated on an $i \times j \times k$ grid such that $i \cdot j \cdot k = n$ where n is the total number of cores. The values i, j and k are chosen such that total area per layer (*i.e.* $i \cdot width_{core} \cdot j \cdot height_{core}$) is less than A_{max} and layer aspect ratio (*i.e.* $i \cdot width_{core} / j \cdot height_{core}$) is close to one. NOC topology is defined as an $i \times j \times k$ 3D mesh [16] and NOC latency is defined as the wire delay of length $\max[width_{core}, height_{core}]$. Wire delay is calculated using the wire delay model from [17] with technology parameters extracted from McPAT source code. NOC topology and latency are fed back into the performance simulator to get accurate inter-core communication simulations.

Note: Since floorplan generation and NOC design only require area and not power estimates we first run McPAT before M2S with all performance statistics set to zero to get area estimations. Then we determine the tiling topology and NOC latency and feed these estimates into the M2S architectural description file. We run McPAT again using the correct performance statistics after running M2S to get power estimates.

2.4 Thermal Model

Once the chip floorplan has been constructed and component power estimation is complete, we have a power density map for each tier. Power density maps are converted into thermal maps using our compact thermal model [18]. A 3D grid is constructed representing the physical structure of the 3D IC. Each tier in the chip stack is divided into four sub-layers: silicon substrate, active silicon, interconnect and oxide cap. Likewise the power map is discretized into a 3D grid and the total power of each power grid is assigned to the respective physical grid in the active silicon sub-layer (all other sub-layers have zero power). Then each physical grid is converted to an electrical circuit representation as shown in Figure 4. Power is modeled as a current source and thermal resistance is modeled as electrical resistance. The voltage at the center of each circuit grid represents the temperature of the respective physical grid. Thermal resistances are evaluated based on material properties and dimensions of the respective physical grid using the technique in [18]. Material properties and dimensions of different sub-layers are listed in Table 1.

[1] A single memory controller can be shared amongst multiple cores

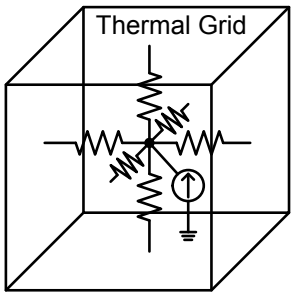


Figure 4: Thermal grid unit cell

2.4.1 Leakage Model

McPAT reports a base leakage value for each CPU component which is estimated at a fixed temperature T_0. To obtain more accurate leakage power estimates which take into account leakage power's strong dependence on temperature we iteratively solve our thermal model and then scale leakage estimates at each grid based on the estimated temperature of that grid after the previous iteration. We repeat this process until the change in temperature between two iterations is less than some threshold (*e.g.*, 1 °C). Equation (1) gives the thermal leakage scaling equation which is extracted from McPAT source code [14] at $T_0 = 360\,\text{K}$.

$$P_{leak}(T) = P_{leak}(T_0) \cdot (5.121(\tfrac{T}{T_0})^2 - 6.013\tfrac{T}{T_0} + 1.892) \quad (1)$$

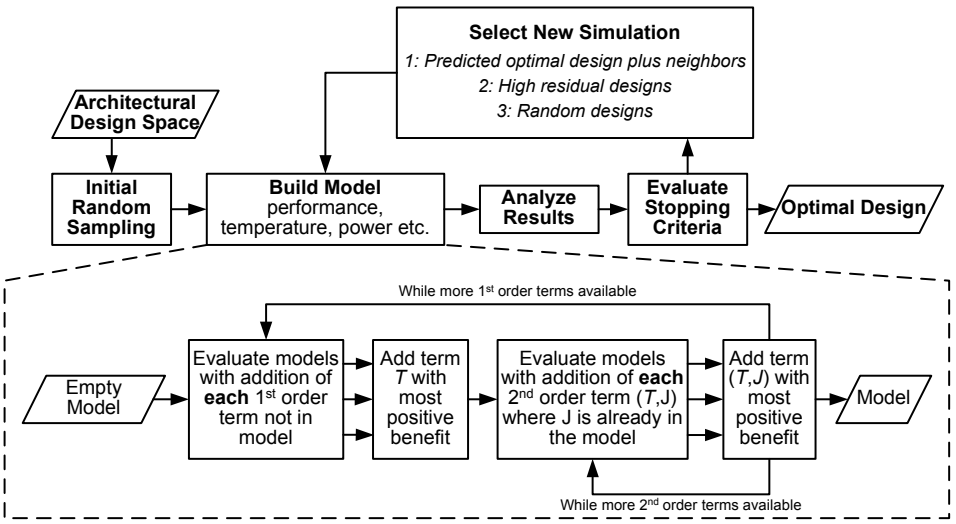


Figure 5: Modeling algorithm

Table 1: Thermal model material properties

sub-layer	Thickness (µm)	Material	Conductivity ($W\,m^{-1}\,K^{-1}$)
Top Substrate	995	Si	148
Thinned Substrate	55	Si	148
Active Silicon	5	Si	148
Interconnect	15	SiO_2+Cu	2.25
Oxide Cap	15	SiO_2	1.4

3. MODELING ALGORITHM

In this section we introduce our modeling technique for 3D CPU design space subject to physical constraints. We use the smoothing spline analysis of variance (SS-ANOVA) [19] modeling technique to build models for each design parameters of interest (*e.g.*, performance, temperature and power) as a composition of cubic spline functions evaluated on a combination of design variables. First we give some background on SS-ANOVA modeling and then describe our algorithm for building models of the 3D CPU architectural design space with a limited number of simulations. Figure 5 illustrates the overall flow of our modeling algorithm, and details are given in the subsections below.

3.1 SS-ANOVA Modeling

A spline is a piecewise polynomial function [19]. In this work we consider cubic splines, which are piecewise cubic functions. Splines are both differentiable and continuous at the piecewise boundaries which are called knots [19]. The smoothing spline is a technique to smooth noisy data by fitting a spline function to the data. Analysis of variance (ANOVA) is a statistical technique for analyzing the underlying source of variations in a population [19]. Multi-factor ANOVA can be used to generate models of an observed data set as a function of some underlying properties of each observation. An observation f can be modeled as a function of the variables $\mathbf{x} = x_1, x_2, \ldots, x_n$ as shown in Equation (2) [3]. SS-ANOVA limits the functions $\{f_1, \ldots, f_n, f_{1,2}, \ldots, f_{1,2,\ldots,n}\}$ to be spline functions which operate on some subset of the variables in $\mathbf{x}$. Each unique subset of input variables is called a term, and the order of a term is the number of members in the subset. c is the trivial function on the 0^{th} order term (*i.e.* a scalar value).

$$f(\mathbf{x}) = c + \sum_{j=1}^{n} f_j(x_j) + \sum_{j=1}^{n} \sum_{k=j+1}^{n} f_{j,k}(x_j, x_k) + \ldots + f_{1,2,\ldots,n}(x_1, x_2, \ldots, x_n) \quad (2)$$

In this work we use the `gss` [20] package for the statistical computing environment `R` [21] to generate a unique smoothing spline model for each design property of interest. To generate each model, `gss` takes a set of simulation data for the design property to be modeled and a set of model terms. However, choosing the appropriate simulation points and model terms is nontrivial problem because that choice strongly affects the quality of the model and suboptimal choices have a high cost in terms of total simulation time and model complexity. Our model building technique is explained in detail in the following subsections.

3.2 Choosing Model Terms

The number of terms (*i.e.* unique subsets of all model variables) associated with n variables is 2^n. However as a rule of thumb a model is unreliable when the number of terms is greater than $s/20$ [22] where s is the number of simulated points. If too many model terms are used, the model can suffer from over-fitting, making it very accurate with respect to the observed data, but a poor predictor of the unsimulated data we wish to predict. Thus the number of model terms must be kept relatively small in order to maintain model accuracy with a small

number of simulations (which is of course the intended goal of the modeling in the first place).

The coefficient of determination (R^2) is a commonly used metric to evaluate how well a model fits the fitting data [23]. However R^2 monotonically increases as new terms are added to a model [4]. Adjusted R^2 ($\bar{R}^2$) [24] is defined in Equation (3) and scales R^2 relative to the number of model terms, m, and the number of data points, s. Thus if an additional model term is added that only marginally improves R^2, $\bar{R}^2$ will decrease, indicating that the added term has reduced the quality of the model. Separate models are built for each design property of interest, so a separate $\bar{R}^2$ is calculated for each model.

$$\bar{R}^2 = 1 - (1 - R^2) \cdot \frac{m-1}{m-s-1} \quad (3)$$

We use a forward selection $\bar{R}^2$ based technique to select the terms in the model. The model building technique is roughly the same as the one used in [4], and is shown in the bottom half of Figure 5. Starting with an empty model we consider each model consisting of one first order term. We evaluate the $\bar{R}^2$ metric for each model and accept the one with the largest value. We then consider adding each remaining first order term and accept the terms that increases the quality of the model by at least θ. Model terms are added starting with the one that offers the largest improvement to model quality. Every time a new first order term is added to the model, we consider all second order interaction terms created by combining the new first order term with any other first order terms already in the model. Amongst all new second order terms generated this way we add any that cause the model quality to improve by at least θ. Terms are added to the model one at a time, and each time any term is added to the model the $\bar{R}^2$ values are reevaluated. So second order terms are added to the model starting with the term that offers the most improvement and ending when no more improvement is available. The model is complete once all first order terms have been added to the model, or when adding any new first order terms causes an improvement model quality less than θ. We limit our model to terms of order two and below, although the proposed model building approach could easily be extended to include terms of arbitrary order. High order interactions are seldom significant [19] so limiting the order of our model is expected to reduce the complexity of the model and the model building procedure without incurring significant losses in accuracy.

3.3 Adding Simulation Points

Initial models are built using a random sampling of η simulation points from the design space. After the optimal model terms are selected for the initial simulation data, the error of the model is analyzed and the optimal design point is predicted using a user specified objective function (*e.g.*, performance or energy efficiency) and set of constraints (*e.g.*, thermal limit, power limit and/or area constraint) evaluated on the model estimates. Unsimulated design points are then chosen to be simulated and added to the data set which is used to rebuild the model. This procedure repeats as shown in Figure 5 until the maximum number of simulations has been performed.

During each iteration of the modeling algorithm estimates of each design property are created for the entire 3D CPU design space. The design points that have already been simulated use the result of the simulation as an estimate whereas the unsimulated points use the predictions from the model. Based on these estimates, one can estimate the optimal design point subject to the constraints. We also identify each design point that is a neighbor of the optimal design, where design i is a neighbor of design j if i and j have the same value for every design variable except one, and the difference is at most ϕ levels (ϕ is referred to as the neighborhood radius). If the predicted optimal point or any of its neighbors have not yet been simulated, they are designated for simulation, and the model is rebuilt with the new simulation data. The idea here is to get better model fidelity around the predicted optimal point, and to get real simulation data in this region to reduce the error of our optimality prediction.

If the predicted optimal point and all its neighbors have already been simulated, we identify the design point with the highest error. If that point or any of its neighbors have not yet been simulated, we simulate them and rebuild the model. Otherwise we simulate χ random unsimulated designs from the design space and rebuild the model. The idea here is to improve the model fidelity in regions with high error, and thus reduce the maximum error of the model. If that isn't possible we resort to random sampling. The three cases for choosing new simulation points is enumerated at the top of Figure 5 in priority order.

3.4 Stopping Criteria

Our model building algorithm terminates when the total number of simulations reaches ζ. We investigate the tradeoff between number of simulations and optimality of our selected design in Section 5.

4. EXPERIMENTAL SETUP

In this section we describe the experimental setup to evaluate the effectiveness of the modeling technique introduced in Section 3. In the following subsections we introduce the 3D CPU design space, our objectives and constraints for choosing an architectural design point and the metrics we use to measure the success of our approach. Results are presented and discussed in Section 5.

4.1 Architectural Design Space

Out study searches the architectural design space in Table 2. Variables with values in brackets can take on any of the bracketed values, and the cross product of all variable values represents the complete design space. The architectural design space in Table 2 contains 4374 unique design points.

Table 2: Architectural design space (baseline architectural values shown in bold).

Variable	Value(s)
Number of cores (*core*)	{**8**, 16, 32}
Memory controllers	*core*·{**1/2**, 1/4, 1/8}
Clock frequency	{**2.4**, 3.0} GHz
NOC width	128 bits
L2 cache size (per core)	{**256**, 512, 1024} kB
L2 cache associativity	{**4**, 8, 16}
L1 cache size (per core)	{**16**, 32, 64} kB
L1 cache associativity	1
Pipeline width	{**2**, 4, 6}
Branch predictor	Tournament
Local history table	1024 8-bit entries
Global predictor	4096 2-bit entries
BTB size	32 kB
BTB associativity	1
Reorder buffer length (*rob*)	{**96**, 128, 160}
Issue queue length	$0.4 \cdot rob$
Load-store queue length	$0.5 \cdot rob$
Fetch queue length	64
Int architectural registers	$0.67 \cdot rob$
FP architectural registers	$0.33 \cdot rob$
RAT size	*rob* 8-bit entries
DRAM size	4 GB
Cache line size	64 B
DRAM bus width	64 B

4.1.1 Stacked DRAM Architecture

This study considers 3D CPUs with stacked DRAM, illustrated in Figure 6. By integrating the DRAM on chip with TSVs, the core-memory bandwidth can be increased drastically. Instead of having just a few memory controllers (MCs) each with an 8-byte DRAM bus, stacked DRAM can accommodate 10s of MCs each with a 64-byte DRAM bus (*i.e.* one full cache line). Larger bus size reduces data transfer delays, and more memory controllers reduces memory queuing delay and facilitates parallel memory access [25, 8]. Stacked DRAM is considered to be one of the primary advantages of 3D CPUs [26, 27].

4.1.2 Software Benchmarks

Each architectural design point is simulated using a set of software workloads from the SPLASH-2 [28] and PARSEC [29] benchmark suites. The performance of each design point is defined as the average normalized performance (Section 4.1.3) across all benchmarks and the maximum temperature for each design point is the maximum temperature across all benchmarks. The specific benchmark programs used for this study are given in Table 3. The inputs and parameters used for each benchmark are the default settings recommended on the Multi2Sim website [12].

4.1.3 Performance Normalization

Since direct averaging of raw performance gives more weight to benchmarks with higher performance, we instead normalized the performance of each design point/benchmark pair to the performance of a baseline architecture (See Table 2) running the same benchmark. Normalized performance is then averaged across all benchmarks to provide average performance values for each design point. This normalization approach is reasonable because the raw performance (*i.e.* instructions per unit time) is not important, but rather the relative ordering of design points with respect to performance. Furthermore all benchmarks should contribute equally to average performance estimates, and normalization is the only way to achieve this.

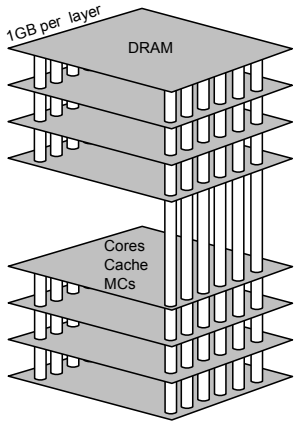


Figure 6: Stacked DRAM architecture

Table 3: Simulated Workloads

SPLASH-2	PARSEC
`water-nsquared`	`blackscholes`
`fft`	`fluidanimate`
	`dedup`

4.2 Modeling Algorithm Parameters

The modeling algorithm introduced in Section 3 can be parametrized to make tradeoffs between simulation time and optimality of the selected design point. In this study we use the following parameters:

- We sample $\eta = 40$ simulation points at random from the design space to build the initial model.
- The threshold for accepting new model terms is $\bar{R}^2_{new} - \bar{R}^2_{current} > \theta = 0$. θ could be increased to reduce the model complexity while only marginally reducing the quality of the model.
- We use a neighborhood radius of $\phi = 1$.
- We iteratively simulate random design points in increments of $\chi = 20$.
- We use a nominal stopping criteria of $\zeta = 200$ simulations. The tradeoff of optimality vs. number of simulations is investigated in Section 5.

4.3 Optimization Objectives and Constraints

The goal of our DSE study is to identify the 3D CPU architectural design point within the design space which has the highest performance (instructions per unit time) subject to thermal and area constraints. An architecture is considered feasible if:

- The maximum temperature amongst all grids in the generated thermal map (and across all software benchmarks as explained in Section 4.1.2) is less than $T_{violation} = 85\,^{\circ}\mathrm{C}$.
- The maximum area of each core layer is less than $A_{max} = 400\,\mathrm{mm}^2$ and the total number of logic layers does not exceed four[2].

Our modeling technique builds separate models to predict the performance and temperature of each architectural design point and identifies the highest performance thermally feasible design.

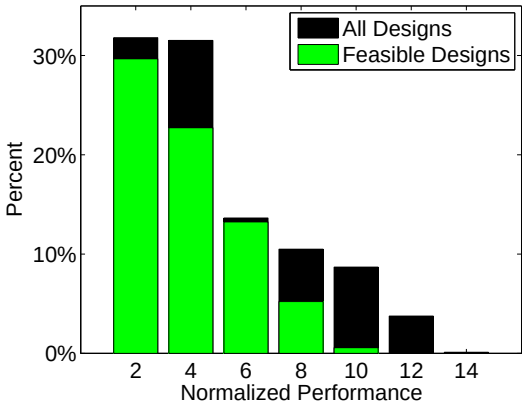
Note: All power area and timing estimations in this experiment are calculated for the 32 nm technology node.

4.4 Evaluation Metrics

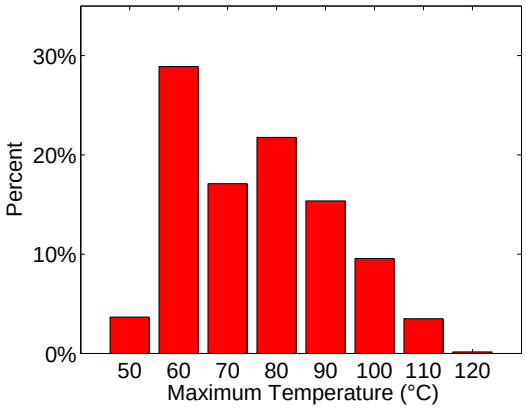
The goal of the experiment is to identify the optimal design point subject to a set of physical constraints while minimizing the total number of simulations performed. Thus the primary metrics used to evaluate the quality of our technique will be the optimality of the solution, the number of simulations performed and the runtime overhead of the modeling technique. The optimality of a chosen design point is defined as the normalized performance of that design divided by the normalized performance of the design that is truly optimal (obtained by exhaustive simulation solely for the purpose of evaluation). In general the optimality of the solution will increase as more simulations are performed, eventually degenerating into the exhaustive simulation where the optimal solution is known. The net speedup of our technique consists of the reduction in total number of simulations minus the runtime overhead of building the models. However we will show in Section 5 that the modeling overhead is negligible compared to the reduction in necessary simulations due to application of our approach.

A secondary evaluation metric for our technique is the overall accuracy of the generated models. This is considered a secondary metric because the goal is to identify optimal design points, and not to propose an accurate model of the entire design space *per se*. Inaccuracy that does not change the relative ordering of design points (*e.g.*, offset bias) does not affect the quality of the DSE. Likewise inaccurate estimates of design points that are far from optimal or far from the edge of the feasibility region are less important since these errors are less likely to change the prediction of optimal design point. For this reason we evaluate error statistics across the subset of design points whose performance is within 20% of the truly optimal design point and whose temperature is within 10 °C of the thermal violation temperature $T_{violation}$.

[2]Area constraints are imposed during floorplanning. If area constraints can not be met during floorplanning, that design point is omitted from the design space.



(a)



(b)

Figure 7: Distribution of design properties in design space (a) normalized performance (b) temperature (°C). Performance plot shows distribution of all designs and distribution of thermally feasible designs.

4.5 Comparison to Other Techniques

The rudimentary technique to which our technique shall be compared is exhaustive simulation. This entails simply comparing the number of simulations performed and the quality of the selected design point to the simulation of all design points and the true optimal design point. However one can conceive of a less rigorous approach to DSE in which some portion of the solution space is sampled at random and the best design amongst the sampled designs is selected. We call this approach the random sampling technique. Both our proposed technique and the random sampling technique present a tradeoff between number of simulations and quality of chosen design. Exhaustive simulation is simply a degenerative case of the random sampling technique where the entire solution space is sampled. We compare the trade off curves of simulation count vs. quality for our proposed technique and the random

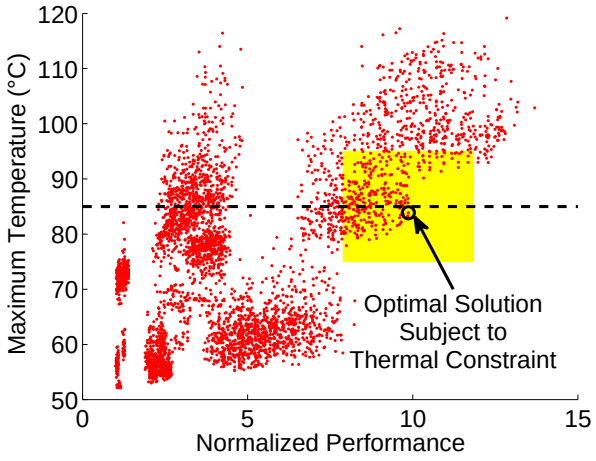


Figure 8: Temperature vs. performance of design space. Optimal solution circled in black. Yellow [shaded] region contains all points within 20% of optimal performance and within 10 °C of $T_{violation}$.

sampling technique and demonstrate the superiority of our technique.

4.5.1 *Dealing with Randomness*

Both our proposed technique and the random sampling technique involve randomness. In some cases the optimal solution could be chosen as the first point to simulate, leading to 100% optimality with only one simulation. Alternatively it is possible that the optimal solution is not chosen until the very last simulation, requiring exhaustive simulation in order to gain 100% optimality (however this is much less likely using our proposed modeling technique than using the random sampling technique). In order to characterize the quality of both techniques we replicate them multiple times and investigate the mean and standard deviations of the results. We show that our technique has significantly better average behavior and significantly less variance. We replicate the random sampling technique 100 times on our data set. Due to the runtime overhead (See Section 5.3.1) of our proposed modeling technique, we replicate it only 8 times.

5. RESULTS

5.1 Design Space Characterization

We begin by examining the properties of the design space. Exhaustive simulation was performed for the purpose of evaluation, as the optimal solution must be identified before the quality of either the proposed modeling technique or the random sampling technique can be evaluated. Exhaustive simulation took weeks to perform using university servers, further motivating the strong need for techniques such as the one proposed in this paper in order to reduce simulation time significantly below that of exhaustive design space simulation.

Table 4: Truly-optimal and predicted-optimal 3D CPU design parameters.

Variable	Value	
	Optimal	Our Method
Number of cores	32	
Memory controllers	8	
Clock frequency	3.0 GHz	
NOC width	128 bits	
L2 cache size (per core)	1024 kB	
L2 cache associativity	16	4
L1 cache size (per core)	32 kB	64 kB
L1 cache associativity	1	
Pipeline width	6	
Branch predictor	Tournament	
Local history table	1024 8-bit entries	
Global predictor	4096 2-bit entries	
BTB size	32 kB	
BTB associativity	1	
Reorder buffer length	128	160
Issue queue length	51	64
Load-store queue length	64	80
Fetch queue length	64	
Int architectural registers	85	107
FP architectural registers	43	54
RAT sets	128	160
RAT entry size	8-bit	
DRAM size	4 GB	
Cache line size	64 B	
DRAM bus width	64 B	
NOC topology	$4 \times 4 \times 2$	
NOC delay	3 cycles	
Normalized performance	9.87	9.85
Max temperature	84 °C	83 °C
Area per layer	298.6 mm^2	299.2 mm^2
Number of core layers	2	

We provide some statistics of the design space properties in order to give context for the results of this study.

Figure 7a shows the distribution of normalized performance across all architectural design points. We can see that the design space is biased heavily towards the low-performance region and the subset of only thermally feasible designs is even more heavily biased. This implies that random sampling is not a very good technique since the probability of randomly sampling a high-performance design point is low. The more biased the performance distribution in a design space is towards low-performance design points, the less effective random sampling will be. Since our proposed modeling technique builds models to predict the high-performance design points and direct simulations towards that region, we expect the solution quality produced by our technique to be much less dependent on the distribution of the design space.

Note: Performance was the design objective for this work, however other objective functions such as energy efficiency could be used equally well since our modeling

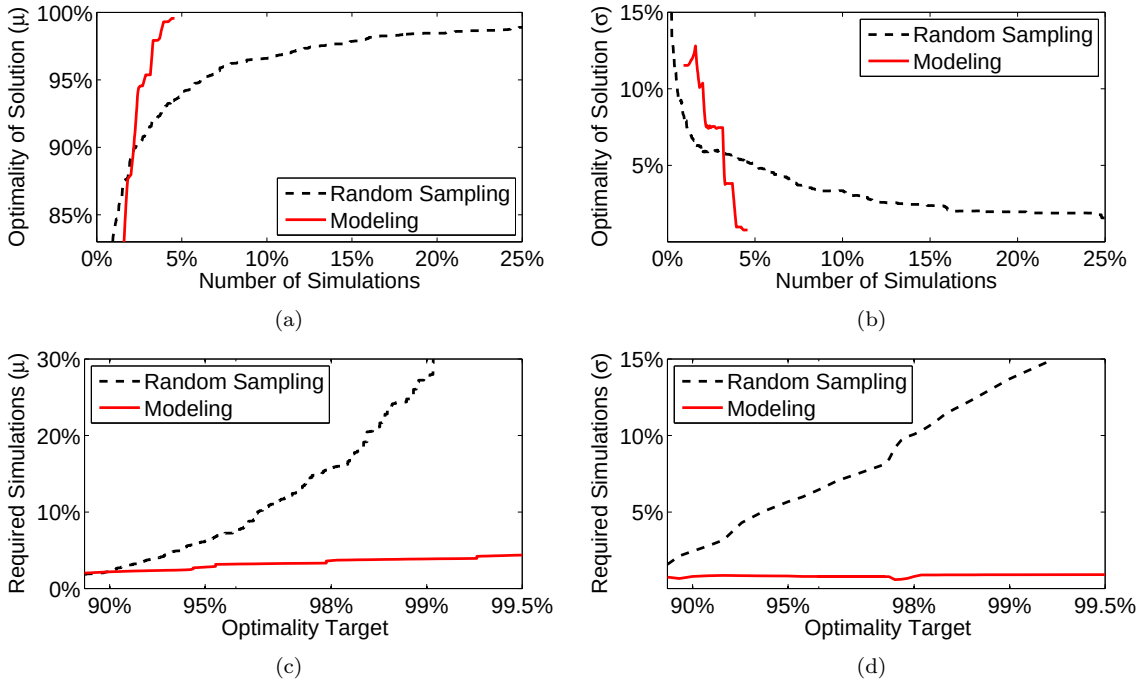


Figure 9: (a,b) Optimality of predicted solution vs. number of simulations (c,d) Required simulations vs. target solution optimality. Plots show mean (μ) and standard deviation (σ) across repeated applications of the techniques.

techniques can be applied to any quantifiable design property.

Likewise Figure 7b shows the distribution of temperature in the design space. Similar to performance, our technique uses modeling to predict the temperature of each design point, and directs simulations towards those points predicted to be both high performance and thermally feasible.

Finally, Figure 8 shows a scatter plot of the performance and temperature of each design point in the design space. The thermal feasibility constraint $T_{violation}$ is shown as a dotted line and the optimal solution point is circled. The region within 20% of the performance of the optimal solution and within 10 °C of the thermal constraint is highlighted in yellow [shaded]. Design points in this region are used to evaluate the accuracy of the model in the region of interest (Section 5.4). We can see that identification of the optimal solution without exhaustive simulation is non-trivial as there are many other design points that have similar performance but vary heavily in temperature, or likewise that have similar temperature and vary heavily in performance. Moreover the correlation between performance and temperature is weak, motivating the need for independent models of each design property.

5.2 Quality of Chosen Solution

We compare the architectural design point selected by our proposed technique to the true optimal solution for a representative run of our algorithm using simulation constraint $\zeta = 200$. The comparison is shown in Table 4 with the optimal solution on the left and the solution predicted by our proposed DSE methodology on the right. The design point chosen by our approach is identical to the optimal design in most variables. The main difference between the chosen solution and the optimal is the L2 cache associativity, L1 cache size and reorder buffer (ROB) length (along with the RF size and queue lengths which are a fixed multiple of the ROB length). The optimal solution scales down the size of the L1 cache and the length of the ROB in order to accommodate a more highly associative L2 cache. The performance of the chosen design is within 0.2% of the optimal performance, and the difference in temperature and area is negligible. Both designs use the same NOC topology and latency.

5.3 Simulation Time and Design Quality

There exists a fundamental tradeoff between the number of simulations and the quality of the identified solution. We compare the random sampling technique to our proposed modeling technique and show that our technique is far better both in terms of quality of tradeoff and the reliability of the approach.

We track the optimality of the proposed solution as the algorithm iteratively adds additional simulation points and compare this to the solution quality produced by the random sampling technique as number of random samples is increased. Both techniques are repeatedly evaluated on the data set and the average trends and standard deviation in solution quality are reported in Figures 9a and 9b respectively. We observe that on average our technique does much better than the random sampling technique when the number of samples

is greater than 2% (roughly 90 simulations). When the number of simulations are less there is not enough data to build proper models, and overfitting occurs causing bad predictions of the optimal design point. Simulating around 2% of the design space allows both techniques to predict a solution within 10% of the optimal solution, but both techniques have very large variance. However as more simulations are added and our proposed technique is able to build proper models, the optimality of our predicted solution quickly approaches 100% and the variance reduces tremendously. Alternatively the quality and variance of the random sampling technique improves much more slowly.

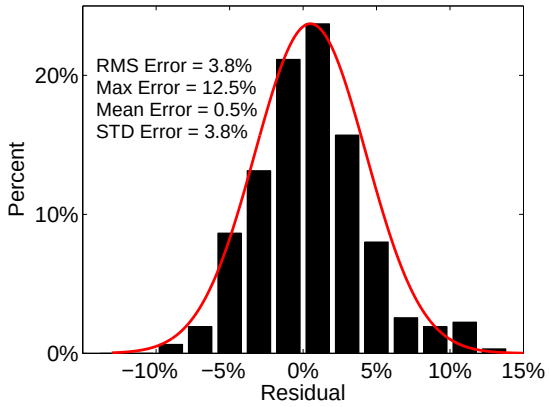
We also examine the data from the perspective of the number of simulations required to reach an optimality target. The mean and standard deviation in number of required simulations to meet a given optimality target are show in Figures 9c and 9d respectively. An interesting result is that if 90% optimality is acceptable for the application at hand the modeling approach is unnecessary as random sampling will find a sufficient solution just as quickly. However as the optimality target is increased the number of solutions required for the random sampling technique increases very quickly towards exhaustive search, whereas our proposed technique requires only marginal increases in number of samples. Moreover the variance in number of simulations required to meet a given optimality target is independent of that target when applying our proposed modeling technique, whereas the variance of the random sampling technique increases exponentially as the optimality target is increased. Our proposed technique is on average able to find a solution within 0.5% of the optimal while simulating less than 5% of the solution space.

5.3.1 *Overhead of modeling approach*

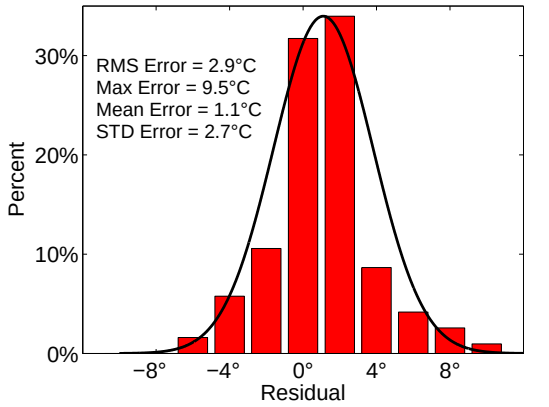
There is obviously some runtime overhead for building the model in the proposed modeling approach. We observed that the time consumed building models was less than the time consumed to simulate a single design point ($< 0.025\%$ of the design space). Figure 9c clearly shows that for high optimality targets ($> 90\%$) this overhead is negligible compared to the savings in number of required simulations.

5.4 Model Accuracy

We examine the distribution of model errors in the region around the optimal solution (yellow [shaded] region in Figure 8) for our performance (Figure 10a) and temperature (Figure 10b) model. We examine percent error in the performance model, and absolute error in the temperature model. The error distributions fit well to a normal distribution with a distribution mean close to zero, which implies an unbiased model. Both root-mean-square (RMS) and maximum error are reasonably low. We reiterate that model accuracy is not a primary goal of the proposed modeling technique. Section 5.3 slows the high quality of our approach at achieving its primary objective: discovering good design solutions subject to physical constraints while using very few simulations.



(a)



(b)

Figure 10: Model error distribution (a) normalized performance (b) temperature (°C).

6. CONCLUSIONS

In this paper we propose a technique for design space exploration of a 3D CPU architectural design space subject to physical constraints. We apply smoothing spline regression modeling to direct our simulations by predicting optimization metrics and physical design properties of unsimulated architectures. We perform an experiment which searches for the design point with maximum performance subject to a thermal constraint. Our technique is able to select a thermally feasible design point within 0.5% of the globally optimal solution while simulating less than 5% of the design space.

7. REFERENCES

[1] D. Gandhi, A. Gerstlauer, and L. John, "Fastspot: Host-compiled thermal estimation for early design space exploration," in *Quality Electronic Design (ISQED), 2014 15th International Symposium on*, pp. 625–632, IEEE, 2014.

[2] D. Genbrugge and L. Eeckhout, "Chip multiprocessor design space exploration through statistical simulation," *Computers, IEEE Transactions on*, vol. 58, no. 12, pp. 1668–1681, 2009.

[3] H. Wang, Z. Zhu, J. Shi, and Y. Su, "An accurate acosso metamodeling technique for processor architecture design space exploration," in *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, pp. 689–694, IEEE, 2015.

[4] W. Jia, K. Shaw, M. Martonosi, *et al.*, "Stargazer: Automated regression-based gpu design space exploration," in *Performance Analysis of Systems and Software (ISPASS), 2012 IEEE International Symposium on*, pp. 2–13, IEEE, 2012.

[5] E. İpek, S. A. McKee, R. Caruana, B. R. de Supinski, and M. Schulz, *Efficiently exploring architectural design spaces via predictive modeling*, vol. 40. ACM, 2006.

[6] B. C. Lee and D. M. Brooks, "Accurate and efficient regression modeling for microarchitectural performance and power prediction," in *ACM SIGPLAN Notices*, vol. 41, pp. 185–194, ACM, 2006.

[7] P. Joseph, K. Vaswani, and M. J. Thazhuthaveetil, "Construction and use of linear regression models for processor performance analysis," in *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*, pp. 99–108, IEEE, 2006.

[8] C. Serafy, A. Srivastava, and D. Yeung, "Unlocking the true potential of 3d cpus with micro-fluidic cooling," in *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*, ISLPED '14, (New York, NY, USA), pp. 323–326, ACM, 2014.

[9] C. Serafy, A. Srivastava, A. Bar-Cohen, and D. Yeung, "Design space exploration of 3d cpus and micro-fluidic heatsinks with thermo-electrical-physical co-optimization," in *Proceedings of the ASME 2015 International Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Microsystems*, ASME, 2015.

[10] Y. Li, B. Lee, D. Brooks, Z. Hu, and K. Skadron, "Cmp design space exploration subject to physical constraints," in *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*, pp. 17–28, IEEE, 2006.

[11] J. Cong, A. Jagannathan, G. Reinman, and M. Romesis, "Microarchitecture evaluation with physical planning," in *Proceedings of the 40th annual Design Automation Conference*, pp. 32–35, ACM, 2003.

[12] R. Ubal, J. Sahuquillo, S. Petit, and P. Lopez, "Multi2sim: A simulation framework to evaluate multicore-multithreaded processors," in *Computer Architecture and High Performance Computing, 2007. SBAC-PAD 2007. 19th International Symposium on*, pp. 62–68, 2007.

[13] P. Shivakumar and N. P. Jouppi, "Cacti 3.0: An integrated cache timing, power, and area model," tech. rep., Technical Report 2001/2, Compaq Computer Corporation, 2001.

[14] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pp. 469–480, IEEE, 2009.

[15] J.-M. Lin and Y.-W. Chang, "Tcg: a transitive closure graph-based representation for non-slicing floorplans," in *Design Automation Conference, 2001. Proceedings*, pp. 764–769, 2001.

[16] B. Feero and P. Pande, "Performance evaluation for three-dimensional networks-on-chip," in *VLSI, 2007. ISVLSI '07. IEEE Computer Society Annual Symposium on*, pp. 305–310, March 2007.

[17] R. H. J. M. Otten and R. K. Brayton, "Planning for performance," in *Proceedings of the 35th Annual Design Automation Conference*, DAC '98, (New York, NY, USA), pp. 122–127, ACM, 1998.

[18] B. Shi, A. Srivastava, and P. Wang, "Non-uniform micro-channel design for stacked 3d-ics," in *Proceedings of the 48th Design Automation Conference*, DAC '11, (New York, NY, USA), pp. 658–663, ACM, 2011.

[19] C. Gu, *Smoothing spline ANOVA models*, vol. 297. Springer Science & Business Media, 2013.

[20] C. Gu, "Smoothing spline anova models: R package gss,"

[21] B. D. Ripley, "The r project in statistical computing," *MSOR Connections*, vol. 1, no. 1, pp. 23–25, 2001.

[22] F. E. Harrell, *Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis*. Springer Science & Business Media, 2013.

[23] M. H. Kutner, C. Nachtsheim, and J. Neter, *Applied linear regression models*. McGraw-Hill/Irwin, 2004.

[24] H. Theil, "Economic forecasts and policy," 1958.

[25] J. Meng, K. Kawakami, and A. Coskun, "Optimizing energy efficiency of 3-d multicore systems with stacked dram under power and thermal constraints," in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pp. 648–655, June 2012.

[26] D. H. Kim, K. Athikulwongse, M. Healy, M. Hossain, M. Jung, I. Khorosh, G. Kumar, Y.-J. Lee, D. Lewis, T.-W. Lin, C. Liu, S. Panth, M. Pathak, M. Ren, G. Shen, T. Song, D. H. Woo, X. Zhao, J. Kim, H. Choi, G. Loh, H.-H. Lee, and S. K. Lim, "3d-maps: 3d massively parallel processor with stacked memory," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pp. 188–190, Feb 2012.

[27] G. H. Loh, "3d-stacked memory architectures for multi-core processors," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ISCA '08, (Washington, DC, USA), pp. 453–464, IEEE Computer Society, 2008.

[28] 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 22Nd Annual International Symposium on Computer Architecture*, ISCA '95, (New York, NY, USA), pp. 24–36, ACM, 1995.

[29] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, PACT '08, (New York, NY, USA), pp. 72–81, ACM, 2008.