

RZ 3883
Computer Science

(# Z1209-002)
24 pages

09/04/2012

Research Report

ExaBounds—Better-than-back-of-the-envelope Analysis for Large-Scale Computing Systems

Phillip Stanley-Marbell‡

IBM Research – Zurich
8803 Rüschlikon
Switzerland

‡Author is now at the Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

LIMITED DISTRIBUTION NOTICE

This report will be distributed outside of IBM up to one year after the IBM publication date.
Some reports are available at <http://domino.watson.ibm.com/library/Cyberdig.nsf/home>.



Research

Africa • Almaden • Austin • Australia • Brazil • China • Haifa • India • Ireland • Tokyo • Watson • Zurich

ExaBounds—Better-than-back-of-the-envelope Analysis for Large-Scale Computing Systems

Phillip Stanley-Marbell*

Abstract

Large-scale computing systems such as supercomputers and commercial data centers pose many unique challenges in terms of power delivery and heat removal, performance at-scale, monetary cost of purchase and operation, and reliability during operation. Architectures to address these challenges must make appropriate hardware-level choices, in the context of the demands of target applications. Given the huge combinatorial space of choices of *algorithms* for solving various computational problems, *hardware architectures*, and *technology* for system implementation, it is necessary to have a mechanism to identify subsets of the algorithm, architecture, and technology design space worth more detailed study.

EXABOUNDS is an analytic framework being developed for efficiently estimating coarse-grained bounds on, and growth rates of, compute performance, power dissipation, monetary cost, and reliability, of large-scale computing systems. While not intended to enable precise performance prediction as detailed processor or interconnect simulators do, it enables *insight* into the interaction between performance, power, cost, and reliability, by providing a meaningful yet simple model with complete *visibility* into the causal relations between system parameters and resulting system behavior. The framework incorporates a large body of empirical technology data, and its utility is demonstrated with a design study for a real future large-scale computing system.

1 Introduction

This draft document provides a very brief and preliminary overview of the EXABOUNDS analysis tool, the preliminary version of which is due to be delivered in month 5 of the Dome project (item 9.6 in version 4 of the planning Gantt charts); the EXABOUNDS analysis tool is being developed over the course of the five-year Dome project. This document also serves as the deliverable for the literature survey of the state-of-the-art (item 1.2 in version 4 of the planning Gantt charts).

Recent semiconductor technology generations have yielded diminishing returns from utilizing silicon area to achieve improved performance via aggressive pipelining

*This work was done in 2012, while the author was affiliated with IBM Research — Zürich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland. The author is currently with the Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

and sophisticated wide out-of-order issue architectures. Across the spectrum of computing applications, there has therefore been a growing interest in eschewing the previous norm of processors with complex microarchitectures, for large numbers of simpler processors.

This trend raises many questions. For example, which point in the design spectrum—with embedded processors on one end, and server processors on the other—is the appropriate choice, in the contexts of *energy-efficiency*, *compute performance*, *reliability*, and *monetary cost*? Should 3D integration be employed in preference to on-die (2D) integration of larger numbers of cores?, and so on. Such questions must ideally be contemplated in the context of complete systems incorporating components such as power supplies, cooling, etc. To illustrate, Figure 1 plots the purchase price, versus 1-year operation costs for just the processors, for achieving a *peak* performance of 10^{18} FLOP/s (top) and 10^{18} INT OP/s (bottom), based on a range of commercially-available processors (see Table 1). From the figure, the processor designs on the Pareto frontier (MX5 and OMP) are clearly desirable over the other processor choices with respect to purchase price and operating cost for workload sizes of 10^{18} FLOPs / 10^{18} INTOPs. However, the power dissipation for real systems will comprise much more than just the processor power. Furthermore, this power will scale with process generations, achieved performance will be dependent on application properties, and the appropriate system architecture will depend on the properties of applications to be executed on it.

1.1 Contributions

EXABOUNDS is being designed to provide an analytic framework to enable the qualitative investigation of the above questions for future exaflop-scale (exa-scale) systems having millions of hardware compute cores and tens of millions of software threads or contexts. Systems of such scale cannot, by definition, be analyzed using conventional queuing-theoretic [41], network [64] or microarchitectural simulation tools [63], and require new tools to enable the efficient exploration of their design space. EXABOUNDS is a new analytic framework being designed with the goal of providing:

- lower bounds on power dissipation, based on lower limits imposed by CMOS device physics and interconnect and packaging technology properties;
- upper bounds on performance imposed by available hardware concurrency and hardware data movement capability, application data movement requirements, available parallelism at different application granulari-

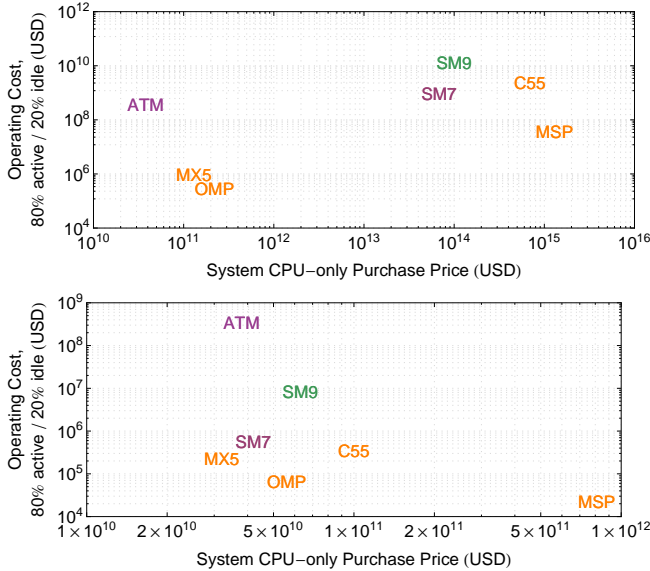


Figure 1. From measured performance, peak and idle power, price, and cost of energy delivery, it is possible to calculate purchase price versus 1-year operation costs for servers, for exa-FLOP (top) and exa-INTOP systems (bottom). This however yields no insight into the tradeoffs for different applications with varying fine- and coarse-grained parallelism, good or poor data locality, the effects microarchitecture, or of technology progression.

ties, and workload size;

- upper bounds on reliability across components of a large-scale system, as a function of, among other things, hardware concurrency (number of system components);
- lower bounds on cost imposed by design, manufacturing, hardware, and power delivery costs, over the running time of an application;
- precise analytic formalization of the *strong-* and *weak-scaling* properties of applications paired to hardware architectures, and an analytic treatment of their relation to *work and span parallelism* [21], *Amdahl's law* [2], *isoefficiency* [36], and the *roofline model* [85].

Where applicable, the analytic models are supported with empirical models of technology scaling trends, built from a large collection of data from real designs, spanning the last three decades, and augmented by ITRS roadmap data. This enables prediction of trends and changes in system-level tradeoffs across future generations of systems. For algorithm-dependent properties in the model, we present characterizations of real-world algorithms, as input to the analytic framework.

Coarse estimates of the bounds on performance, power, reliability, and cost, while not providing the precision of microarchitecture simulation studies or analytic models based on microarchitectural parameter design space exploration [59], are still useful to system designers. Such bounds are complementary to fine-grained modeling tools such as McPAT [60], CACTI [81], Graphite [63], Wattch [12], and Mambo [8], which would find use in more detailed model-

ing of some fraction (e.g., the processor socket granularity) of a complete data center or supercomputer. We use McPAT and CACTI to obtain die-level power and area estimates, and build on top of them an analytic model for compute performance at the core-level, and power, performance, reliability, and cost models for entire computing clusters. By considering performance, power, reliability, and cost within a unified analytic framework, we are able to capture important potentials for design-time tradeoffs. For example, this enables co-consideration of how constraints, on, say, power delivery can influence performance, or how reliability and cost can be affected by architectures which require large numbers of system sockets. Unfortunately, however, there has hitherto been little research in such integrated whole-system bounds.

1.2 Structure of the framework

EXABOUNDS captures the interaction between application properties such as instruction-level parallelism (ILP), thread-level parallelism (TLP), data-level parallelism (DLP), data movement, and memory reference locality, and hardware properties. These hardware properties include available instruction-, thread- and task-level concurrency, available memory and interconnect bandwidth, and the influence of clock frequencies, operating voltages, and transistor counts. Interactions between hardware and software are captured from the die level, through the socket, card, mid-plane (rack-unit), rack and aisles of racks, to a complete system. The central themes of *parallelism* and *data movement* at all levels of the hierarchy, are used to constrain the complexity of the models. Because the models are constructed in a bottom-up fashion, as opposed to using generic regression fits of empirical data, the analytic formulations capture the intuition behind the trends observed.

The effectiveness of the bounds predictions is evaluated by comparing its predictions to the power, performance, and cost of two real prototype systems. The real-world validation on a diverse set of platforms, ranging from two ARM-based platforms to a multi-core PowerPC platform, is achieved using a combination of performance measurements, power measurements, and thermal measurements to estimate power dissipation apportionment for whole computing systems (including aspects such as fans and power supplies), as captured by the model.

Following an overview of related research in Section 2, Section 4 presents the integrated power, performance, reliability and cost framework, along with empirical data, drawn from information obtained for more than 130 programmable processor designs over the last three decades, that provides part of the device-related input to the analysis. Section 5 presents the implementation of EXABOUNDS as a software tool, and a preliminary analysis of the trends predicted by the framework, using both a state of the art high-performance server configuration, as well as a state-of-the-art low-power ARM processor implementation. The article concludes in Section 6 with a summary.

2 Related Research

The analysis and prediction of computation- and power-performance of systems encompasses a broad range of techniques. Traditionally, different approaches have been employed at different levels of system detail, and for different system scales or sizes.

EXABOUNDS may be compared to existing research along five dimensions: ① *modeling approach*, ② *efficient design-space exploration*, ③ *empirical semiconductor- and circuit-*

Table 1. Energy per integer (INT-OP) and double-precision floating-point operation (DP-FLOP) at peak system utilization, average leakage power, and silicon (packaged die) cost for achieving a peak capability of 10^{18} DP-FLOP per second. For processors without hardware floating-point units, the cost of software emulation is used. Energy delivery operational expenses (E-OPEX) per processor assumes 80% of time at peak performance, over a time period of 1 year, and an energy cost of 0.15 USD per kWh.

Processor		Energy for 1 DP-FLOP (μ J)	Idle Power for 10^{18} DP-FLOP/s system (MW)	Device cost for 10^{18} DP-FLOP/s system (Trillion USD) and 1 yr. E-OPEX	Energy for 1 INT-OP (nJ)	Idle Power for 10^{18} INT-OP/s system (KW)	Device cost for 10^{18} INT-OP/s system (Billion USD) and 1 yr. E-OPEX
No hardware FPU							
❶ TI MSP430F2274	(MSP430)	7.9	130	1,300 ; 34 E-6	4.9	82	810 ; 21 E-6
❷ Marvell PXA32x	(ARM)	3.8	230	■ ; 62 E-6	2.4	140	■ ; 39 E-6
❸ Atmel AT91SAM7S16	(ARM)	3.1	3,200	68 ; 860 E-6	1.9	2,000	42 ; 540 E-6
❹ TI TMS320C5515	(C55x)	1.4	8,000	690 ; 2,100 E-6	0.22	1,200	100 ; 320 E-6
❺ Atmel AT91SAM9M10	(ARM)	0.52	48,000	100 ; 12,000 E-6	0.32	30,000	63 ; 7,800 E-6
Hardware IEEE754-compliant DP-FPU							
❻ Freescale i.MX515	(ARM)	0.0025	3.3	0.13 ; 0.86 E-6	0.63	820	32 ; 210 E-6
❼ Intel Atom N450	(x86)	0.0033	1,200	0.038 ; 320 E-6	3.3	1,200,000	38 ; 320,000 E-6
Ⓣ TI OMAP3503	(ARM)	0.0084	0.96	0.22 ; 0.25 E-6	2.1	240,000	56 ; 63 E-6
❸ IBM BG/L CPU	(Power™)	0.0046	71	■ ; 18 E-6	9.2	140,000	■ ; 37,000 E-6
❹ IBM PowerEN	(Power™)	0.0017	540	■ ; 140 E-6	1.7	540,000	■ ; 140,000 E-6

level models, ❶ modeling of algorithm-hardware interaction, and ❺ modeling total cost of ownership.

2.1 Modeling approach

The challenge of design space exploration may be approached from two extremes. A *top-down empirical characterization* of whole-system performance may be performed, possibly using techniques from statistical experiment design [74] to significantly reduce the number of measurements needed. Alternatively, a *bottom-up first-principles* approach may be employed, in which the known interactions between system parameters are used to construct a model for system performance. Such a model may then be used without calibration for comparative studies (*relative performance*), or may be calibrated with real-system measurements to enable prediction for *absolute performance*.

A variety of methods to characterize parallel system performance in a bottom-up manner, with a small set of closed-form relations have been previously explored in the research literature. They include models such as various PRAM models, LogP [23] (which improves upon PRAM models by capturing the computation and communication latency, overhead, bandwidth and processing constraints of real-world parallel computing systems), and parallel performance metrics such as the isoefficiency [36].

Like Hoefler et al. [43], our objective is to enable early-design-stage exploration of the interaction between algorithm properties and compute architectures, in the context of whole-system metrics such as performance and power dissipation. Hoefler et al. however, like Kerbyson et al. [53], illustrate these concepts with hand-constructed analytic models for specific applications (what they term a *semi-empirical* approach), while we provide a set of broadly-applicable models, and tools for automation of application characterization to complement them.

To enable the estimation of application performance without the need for time-consuming simulation or hand-crafted analytic models of program behavior, Karkhanis and Smith developed a *first-order superscalar processor model* [51], which, like the model of Azizi et al. which follows it [5], is at the microarchitecture level. Like the *mechanistic*

models of Karkhanis and Smith, and Eyerman [51, 52, 32, 33] (but unlike the work of Noonburg and Shen [67]), for the processor-level component of our models, we start from a baseline performance model formulated from empirically determined ideal-case instruction-, thread-, and data-level parallelism, and data reuse distance distributions, and add penalties due to non-ideal components of the memory hierarchy. This is similar in spirit to the *lost cycle analysis (lca)* approach of Crovella and LeBlanc [22], as well as to techniques employed in performance analysis to build *CPI stacks*; indeed, we use the unique detailed CPI stacks that can be constructed from the IBM Power7 platform’s hardware performance counters to validate our performance predictions later in this article. Lca, however, requires the user to build analytic models of the performance of each algorithm, in order to fit these models to empirically measured properties.

Like Boyd et al. [11], Czechowski et al. [24] and DeBenedictis [27], our intentions are to provide upper bounds on performance, to enable system designers to estimate the best performance achievable for a given algorithm implementation on a target machine; the MACS (machine, application, compiler, and schedule) bounds from Boyd et al. are, in essence, best-case instruction execution rates coupled with instruction mix counts. While both their work and ours involve the concept of “hierarchies”, the MACS hierarchy refers to levels of refinement of the model, while in our case, a hierarchy is used to capture the physical composition of large-scale computing systems. DeBenedictis estimates system performance using a simple set of machine balance targets combined with “derated” processor and technology characteristics. The “balance principles” of Czechowski et al. [24], can be seen as a subset of our bounds relations, with a focus only on the best-case instruction throughput and memory bandwidth, not taking into account the interactions between various hardware constraints (e.g., cost, packaging, and power delivery).

In analyzing whole-system performance, one popular approach is to model system behavior as a Markov chain or queuing network. Queuing models permit the analysis

of whole-system behavior when some dynamic properties of the hardware and applications are known. These properties might be, e.g., the access duration and inter-arrival time distributions for specific hardware resources, resulting from application or end-user properties. Given these characteristics, discrete event simulation is typically used to determine steady-state resource utilization, and hence system performance. Under certain restrictions, it is possible to obtain direct analytic results without resorting to such simulation.

Sharapov et al. [76] use a combination of queuing-theoretic modeling and hardware constraints to capture the behavior of applications on petascale-class systems, focusing on performance prediction, and not capturing the interactions between performance, power, reliability, and cost. Even though they outline a detailed infrastructure for manual performance characterization of individual applications, they employ a simple argument based on Amdahl’s law for performance scaling predictions to large (petascale) system sizes.

The PHANTOM framework [87], like the performance prediction sub-component of EXABOUNDS, is relevant to the task of predicting behavior of applications at scale. It however inherently depends on access to a single node instance of the target full system, and it provides no facilities for a system architect or algorithm designer to gain insight into how algorithm properties might influence performance, power, reliability, or cost, across a range of architectural choices and implementation technologies; it simply achieves performance prediction by scaled simulation and replay of measured compute times and communication traffic.

For performance and power estimation of small-scale systems (e.g., a single processor socket) microarchitectural simulators such as Simics [61], Mambo [9], SimpleScalar [13], Wattch [12], and Graphite [63] may be used, in conjunction with memory subsystem timing and power-performance predictors such as CACTI [81], or network simulators [84, 64].

2.2 Efficient design-space exploration

Several prior research efforts have focused on enabling rapid exploration of architectural choices for microprocessor designs, using techniques ranging from regression modeling [59] and neural networks [47], to various deterministic and probabilistic first-principles and mechanistic models [5, 51, 52, 67]. These efforts have however focused almost entirely on modeling or estimating performance, with limited coverage of the interaction between performance, packaging, power consumption, reliability, and cost. Unlike the few prior attempts to estimate the effects of device, materials, and circuit parameters on computing performance for a single processor socket [60, 38, 35], the EXABOUNDS framework is targeted at coarse-grained analysis of complete computing clusters; this means that the analytic framework takes into account properties ranging from transistor switching speeds and logic depth per pipeline stage, to number of racks and machine room cooling efficiency. Like the HLS framework [68], our goal is to enable fast early-stage design exploration; HLS, however, only targets performance prediction for a microprocessor core.

2.3 Empirical semiconductor data and circuit-level models

One commonly-used rule of thumb for the relation between hardware complexity (in die area or transistor count), and achievable performance, is Pollack’s rule, which postulates that performance grows as the square root of the num-

ber of transistors [10]. A number of research efforts have provided more quantitative bases for relating achieved performance to transistor-level cost, starting with the work of Palacharla et al. [69].

McPAT [60] enables analytic modeling of complete microprocessors, from their microarchitectures and on-chip interconnect networks, to the influence of circuit and historical (ITRS) CMOS technology trends. Its use of timing estimation is primarily in the context of using timing targets as constraints for its power and area estimation. Area estimation is performed using analytic modeling in the style of CACTI for on-chip array structures such as SRAMs and CAMs, and using historical data for estimating the area costs of structures such as on-chip memory controllers.

The empirical models of Eble [30] relate cycles per instruction (CPI) to the number of transistors for on-chip logic (i.e., excluding transistors for caches). Eble however implicitly assumes that CPI can continue to increase as long as more hardware resources are used. While this was true for low-issue-width architectures as empirically demonstrated by Eble, it ceases to be true for high issue widths, since many applications have limited *usable* instruction-level parallelism [17]; the diminishing returns to performance from increasing transistor counts in recent years is illustrated in Figure 2.

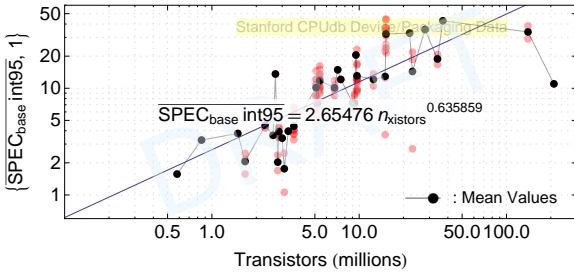
Timing- and power-performance of circuit-level designs are typically performed with tools such as SPICE [66], possibly using technology models such as the Berkeley predictive technology model and its derivatives [88] to predict properties of future semiconductor process generations. Similarly, tools such as BACPAC [79], SUSPENS, and GTX [16] enable timing and power prediction for an entire design or die—clock distribution, wiring, I/O, etc.

To support the analytic models in EXABOUNDS, we built a database of design properties for a large collection of digital designs (microprocessors, DSPs, and other digital ASICs), from publications in the solid state circuits literature (e.g., ISSC and JSSC). The Stanford CPU database (CPUdb) curated by Horowitz et al. [25], in contrast, is based on datasheet values, and is restricted to microprocessors. Like them, we have published the provenance of our data points [78], enabling the community to duplicate or verify our observed insights. An advantage of our use of data from ISSC and JSSC publications is however that the values used in our corpus are typically measured values of, e.g., power dissipation at a stated supply voltage and clock frequency as opposed to CPUdb’s reliance on the reported supply operating range, and their use of *thermal design point (TDP)* values as estimates of power consumption.

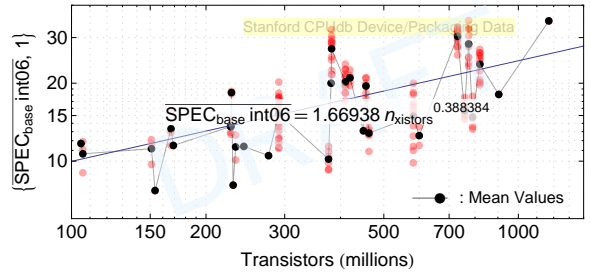
2.4 Algorithm-hardware interaction

Like Jouppi [49], and Noonburg and Shen [67], the performance sub-component of our models are structured by decoupling an application’s available parallelism from machine-available parallelism. Our application-parallelism characterization captures not just mean values like prior work [58, 72, 83, 4, 71], but rather, the entire *distribution* of ideal-case ILP/TLP values over the program’s execution, a superset of the *smoothability* information of Theobald and Gao [80].

Hill and Marty [42], and, more recently, Borkar and Chien [10], investigated the tradeoff between homogeneous and heterogeneous mixes of core sizes. Our work enables such discussions to be placed on a more quantitative footing by capturing application properties that map to large



(a) SPEC INT 95 rating versus transistors per die.



(b) SPEC INT 2006 rating versus transistors per die.

Figure 2. Diminishing benefits to performance from increasing transistor counts (from $n_{\text{xistors}}^{0.64}$ to $n_{\text{xistors}}^{0.39}$).

single-thread-performance cores (i.e., ILP), versus application properties that map better to many smaller cores (more DLP or TLP).

There have been a number of previous attempts at characterizing the interrelation between algorithm parallelism, performance, power dissipation, and energy-to-completion [57, 19]. These efforts have focused on manually analyzing the parallelism in an algorithm (e.g., via manual work and span analysis [21]), and then subsequently combining these hand-analysis results with relations between supply voltage, power/energy, and clock frequency. In our work, we make the significant step of being able to perform such analysis on compiled program binaries, automating the work and span analysis to estimate ideal-case parallelism of real applications. More importantly, we use the results of these analyses at the scale of complete compute clusters, and not for an isolated microprocessor.

Like Marin and Mellor-Crummey [62], we employ architecture-agnostic characterization of applications, and the subsequent independent pairing of that characterization with machine properties to enable performance prediction. Unlike their work however, which characterizes only application locality, we characterize temporal and spatial locality, as well as instruction- and thread-level parallelism. Characterizing parallelism inherent in serial applications enables us to start with serial applications just like Marin and Mellor-Crummey, but to then make realistic predictions not just for candidate serial platforms, but also for parallel ones.

Our goals of enabling fast early-design-stage modeling are similar to those of the PMAc project [1], which focused primarily on *memory bound kernels*. Like them, we also adopt the approach of separately characterizing architecture-independent application/algorithm properties, and hardware properties, before combining the two to predict performance; thus, the PMAc approach at the cluster level is similar to the approach of Noonburg and Shen [67] and by Jouppi [49] at the microarchitecture level, and to our approach at the level of a complete supercomputing cluster or datacenter. Unlike PMAc, which estimates the effect of the memory system based on an empirical probe program (MAPS), we rigorously characterize the temporal and spatial reuse distances of applications, which enables us to accurately estimate cache and TLB hit rates for a variety of hardware configurations after the fact, from a single application characterization.

2.5 Modeling costs in large-scale systems

Koomey [56] describes a tool for estimating the *true total cost of ownership* for datacenters, capturing both the capital and operational expenses. Our cost models, presented in Section 4.7, are focused on operating expenses, in particular, energy delivery costs for computing and communication, while treating other operation costs such as cooling and power regulation using efficiency factors. For the computing and communication energy costs however, we provide significantly more fine-grained modeling, as functions of application properties, workload size, and hardware implementation properties.

Patel et al. [77] have investigated modeling the thermodynamics and electrical energy for driving cooling equipment in datacenters. We purposefully thus sidestep duplicating such work, and focus on modeling more interaction between applications and hardware, and components of the hardware at the board level.

3 Illustrative Example

As an illustrative running example in the remainder of the article, we consider the task of designing a computational backend for a radiotelescope. This challenge is one of significant importance to large scientific instruments such as the preexisting *low-frequency array (LOFAR)* [26], and the planned *square-kilometer array (SKA)* [28].

Two aspects of the computational workload of these systems are the *per-station processing* for aperture arrays (beamforming of signals from multiple antenna elements), and *central data processing* (correlation of the signals between multiple stations). The core *computational problems* (Figure 2.5, top) behind the latter component can be addressed with the simple *algorithmic solution* listed in Figure 2.5(bottom). Although the core algorithms solving the problems of interest may be simple, the data rates that must be processed are substantial.

The current LOFAR deployment sustains data rates of 230 Gb/s going into the per-station beamformer, requiring approximately 36×10^{12} multiplies per second, per station. Given the inherent requirement that stations are in remote locations, this processing should be achievable at a minimum power budget, such as that which can be provided by a local solar panel.

At an even larger scale, the central data processor must cross-correlate signal streams from all antennas in an instrument. For example, for the high-frequency (< 25 GHz) component of the final SKA system, this will correspond to up to 3300 antennas, each handling up to 17,000 channels, each of two polarizations. At 16 bits per sample, and a sam-

```

CORRELATE(samples :  $\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rightarrow$  visibilities :  $\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ )
1   $t$       :  $\mathbb{N} = \{0, \dots, n_{\text{integ}} - 1\}$ .
2   $ch$      :  $\mathbb{N} = \{0, \dots, n_{\text{chans}} - 1\}$ .
3   $bl$      :  $\mathbb{N} = \{0, \dots, \frac{1}{2}(n_{\text{chans}}^2 - n_{\text{chans}}) + n_{\text{chans}} - 1\}$ .
4   $stn_A, stn_B$  :  $\mathbb{N} = \{0, \dots, n_{\text{stns}} - 1\}$ .
5   $pol_1, pol_2$  :  $\mathbb{N} = \{0, \dots, n_{\text{pols}} - 1\}$ .
6
7  ▷
8  ▷      Computational problem definition for correlation. The following property is satisfied by a valid visibilities array:
9  ▷
10  $\forall bl \forall ch \forall pol_1 \forall pol_2 \forall stn_A \forall stn_B ((stn_B \geq stn_A) \cap (bl = \frac{1}{2}stn_A(2n_{\text{stns}} - stn_A - 1))) \Rightarrow$ 
11 visibilities $[bl][ch][pol_1][pol_2] = \sum_{t=0}^{n_{\text{integ}}-1} (\mathbf{samples}[stn_A][ch][t][pol_1] \cdot \mathbf{samples}[stn_B][ch][t][pol_2])$ .

SIMPLECORRELATOR(SamplesType  $samples[n_{\text{stns}}][n_{\text{chans}}][n_{\text{integ}}][n_{\text{pols}}]$ )
1  for  $ch \leftarrow 0$  to  $n_{\text{chans}}$ 
2     $bl \leftarrow 0$ 
3    for  $stat1 \leftarrow 0$  to  $n_{\text{stns}}$ 
4      for  $stat2 \leftarrow stat1$  to  $n_{\text{stns}}$ 
5         $bl \leftarrow bl + 1$ 
6        for  $pol1 \leftarrow 0$  to  $n_{\text{pols}}$ 
7          for  $pol2 \leftarrow 0$  to  $n_{\text{pols}}$ 
8             $sum \leftarrow 0$ 
9            for  $pol2 \leftarrow 0$  to  $n_{\text{pols}}$ 
10              $sum \leftarrow sum + samples[stat1][ch][time][pol1] * conj(samples[stat2][ch][time][pol2])$ 
11
12              $visibilities[bl][ch][pol1][pol2] \leftarrow sum$ 
13
14 return  $visibilities$ 

```

Figure 3. A computational problem of interest and a simple algorithm that solves this computational problem.

ple rate of 1 Gs/s, this yields over 1×10^{17} samples per second, which must be cross correlated, or 112 M correlations per sample group, at a rate of 1 Gs groups per second.

For the above subsystems, we would like to be able to answer certain kinds of questions:

- What are the properties of various algorithmic solutions to the beamforming and correlation computational problems?
- At the microarchitecture level, do these algorithms in consideration have available instruction- or thread-level parallelism?
- Do the different algorithms solving the same computational problem have differing amounts of data reuse?
- What is the interplay between exploited parallelism and the pressure on the memory system?
- What is the tradeoff in terms of performance, power, and die area of larger caches?
- Should multiple cores be integrated on a die, or should concurrency integration occur at the level of packaged dies?
- What fraction of system power dissipation will be lost to point-of-load voltage regulation?
- What will be the monetary power delivery costs for a system of a given size, over the course of a year?

The EXABOUNDS analytic framework is designed to provide insight into such questions, by providing bounds on

compute performance, power dissipation, monetary cost, and reliability. These bounds are not tight, but rather coarse estimates intended to guide more detailed (and time-consuming) analysis, early in the design process.

4 Hierarchical analysis of large-scale computing systems

A typical large-scale computing system may be viewed from at least eight layers of detail—the *core*, *die*, *socket*, *card*, *rack unit*, *aisle*, and *whole system* layers, as illustrated in Figure 4. When the behavior of such a hierarchy is captured in a set of analytic relations, the equations corresponding to lower levels in the hierarchy may be seen as *forward predictive equations*, and predict values of properties at higher levels; the *metrics* or outputs of the relations at one level are often *parameters* or inputs of the next level. An example is the formulation of the performance per die (level 1 in Figure 4) in terms of the performance per core (level 0 in Figure 4), number of cores per die, available memory bandwidth per die, and application properties. Similarly, a set of *backward constraint equations* may restrict the values of parameters and metrics at lower levels of detail, based on constraints defined at higher levels of detail; an example of such a constraint is the core-level power dissipation limit imposing constraints on the core design parameter values.

4.1 Terminology and notation

Tables 2 and 3 list a representative subset of the parameters that appear in the analytic relations defined in the remainder of this article, across multiple levels of the hierarchy of Figure 4. In the table, the parameters at a given

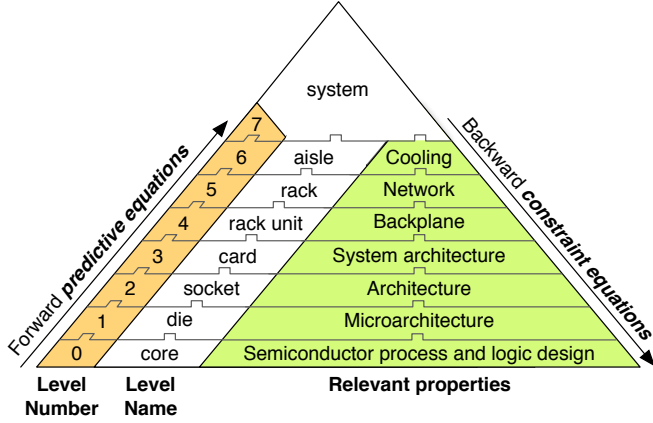


Figure 4. A computing cluster may be viewed at several hierarchical levels, which may be referred to as *tiers*, *scales*, *strata*, etc. This article considers eight such levels.

level have subscripts corresponding to which (of eight) levels they pertain to. Thus, e.g., n_1 denotes the number of programmable processor cores per die (level 1). Superscripts are used to distinguish between possibly-multiple parameters of the same type at a given level, and will be omitted when the intended meaning should be clear from the context. In the remainder of this article, the basic granularity of time employed will be a processor clock cycle, clk .

The term *application-level parallelism (ALP)* at a given level in a hierarchy will be used to denote the maximum number of units of work, for a given application, that may simultaneously be in progress at a particular moment, restricted only by the application’s innate dependencies. At level 0 (within a processor core), this corresponds to the instruction-level parallelism (ILP); at level 1, it is the thread-level parallelism (TLP). In the remainder of this article, when the parallelism is characterized from serial applications [17], this TLP will be *basic-block-level TLP*, since that is the granularity at which thread-level parallelism can be meaningfully machine-extractable from serial applications. At higher levels in the hierarchy, the parallelism may be coarse-grained TLP (e.g., as manifest in explicitly parallel programming models such as Cilk, OpenMP, or Pthreads), multiple-program-multiple-data (MPMD) parallelism, or data-level parallelism (DLP).

The bounds we consider fall into four groups:

- **bounds on computation throughput** (C_0 to C_7), the reciprocal of the time to solution for a hierarchy of problem subdivisions L_i at level i ;
- **bounds on power dissipation** at all the levels in the system hierarchy (P_0 to P_7);
- **bounds on monetary cost** $\$$; at a given hierarchy level;
- **bounds on mean time to failure**, MTBF $_i$.

The compute throughput is in principle defined for each of integer (henceforth abbreviated INT), single- (SPFP) and double-precision floating point (DPFP). In what follows, to simplify the exposition, the type distinction is however omitted unless necessary; instruction mixes can however be used in practice as input to the link between application parallelism and machine parallelism models.

Table 2. Notation for application-/algorithm-specific application parameters. Only this small set of application properties are required, and tools for their automatic characterization are detailed in Section 5.

Description	Notation	Working Example
Overall workload size (instructions)	L_{sys}	120E12 instrs.
Workload subdivision at level i	L_i	10E9 instrs.
Fraction of instrs. that perform	F_i^{dmo}	0.5
Data movement operations at level i		
Data movement accesses per cycle, level i	$B_{app,i}^{dmo}$	0.5 B/pclk
Innate parallelism at level i	ALP_i^{type}	4 at $i = 0$
Data address reuse distribution at level i	$D_i^{data-reuse}$	(distribution)
Instruction address reuse at level 0	$D_0^{instr-reuse}$	(distribution)
Instruction type mix at level 0	F_0^{type}	$F_0^{ctrl} = 0.15$

The relations that follow feature a number of constants of proportionality, which are all denoted K , with distinguishing super- and subscripts.

4.2 Performance bounds

Computation may be characterized, broadly, as consisting of *arithmetic operations* and *data accesses*. Memory is used as the mechanism through which dependent instruction streams communicate, e.g., via read-after-write (RAW) true dependencies. When applications are parallelized, whether across multiple cores on a die sharing a single memory, or across multiple processors connected via a network, these (formerly serialized) dependencies must be satisfied through memory (in the former case) or via network communication (in the latter). In what follows, the term *data movement* is used to refer to both communication to memory and communication over an interconnect.

The raw data movement requirements of applications (and the temporal and spatial reuse in those access patterns) will limit the benefits of added hardware concurrency in the presence of limited scaling of memory and interconnect bandwidths and latencies. Similarly, application parallelism (at the instruction-, thread- or task-level) will limit the benefits that can be gained from hardware-provided concurrency.

In practice, a system will execute a large collection of algorithms both concurrently and in series. In the remainder of this article, the workload is described as though it were a single algorithm with properties representative of the overall workload mix.

4.2.1 Hierarchical workload division

Given an algorithm and its input, there is a *total amount of work*, L_{sys} , in instructions of a given type, that needs to be completed. Small problem sizes correspond to the dominant case in so-called *capacity machines*, while large problem sizes correspond to the typical situation in *capability machines*. We formulate the computational work, L_i , to be performed *per computation unit* at level i in the hierarchy of Figure 4, by an application with amount ALP_i of parallelism available, as

$$L_i = \frac{L_{i+1}}{\min(n_i, ALP_i)} \quad \text{instructions, } \forall i < 7,$$

with the boundary condition of

$$L_7 = \frac{L_{sys}}{\min(n_7, ALP_7)} \quad \text{instructions.}$$

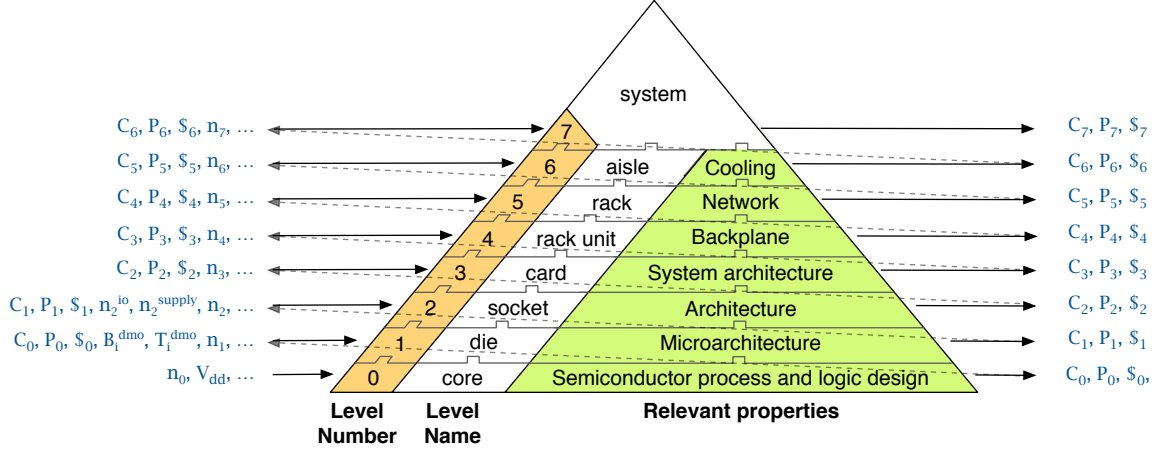


Figure 5. Inputs to, and outputs from, the different layers of the hierarchical EXABOUNDS model. Only a small set of application properties are needed for application-driven analyses. The majority of the parameters in the model are technology dependent quantities that change over technology generations and need not be specified by architects.

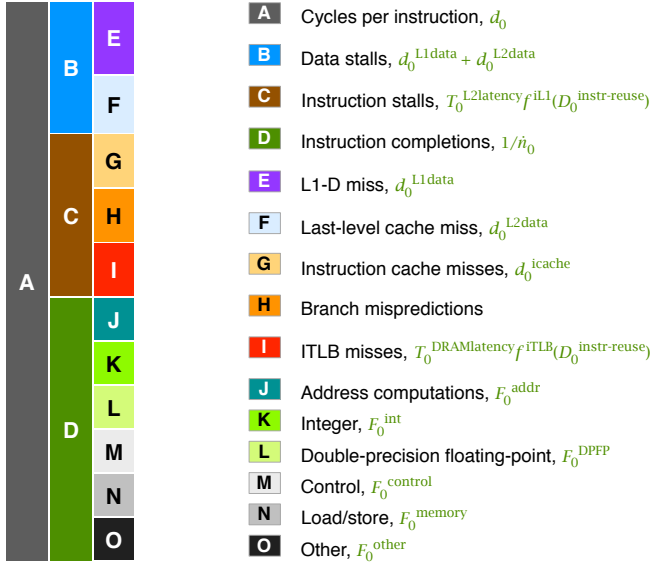


Figure 9. The EXABOUNDS analytic framework captures the contributors to execution delay. The cycle delays per instruction can be represented as a CPI stack breakdown diagram.

That is, the actual workload per each compute component at a given level is determined by how much parallelism in hardware at that level (n_i) can be exploited by an application having available parallelism ALP_i . These relations succinctly capture the recursive division of a workload (i.e., a given algorithm paired with an input dataset) over a system's compute components; this formulation will be refined in Section 4.6 to account for the additional work, per compute element, that must be carried when some parts of the system fail. An example is shown in Figure 6.

4.2.2 Computation throughput

The computation throughput achieved by a system is a combination of properties of applications and that of

the hardware (micro) architectures on which they execute. This is most intuitively captured by the cycles per instruction (CPI), expressed as the product of program properties (events per instruction for event type i , F_i) and hardware properties (cycles per event, for event type i , MP_i) [31]:

$$CPI = \sum F_i \cdot MP_i. \quad (1)$$

The CPI for an application is broken down further into multiple components, as illustrated in Figure 9. The CPI stack estimated from EXABOUNDS can be compared to that obtained from hardware performance counters read from actual applications running on the IBM Power architecture (Figure 7 and Figure 8). These components to the instruction delay, shown in the Figure, are detailed further in what follows.

The instruction cache and I-TLB miss components of CPI are computed as the product of I-cache/I-TLB miss penalty (cycles to access L2 or memory) and the probability of a miss, which is in turn computed from the instruction address reuse histogram (i.e., the instruction stream's "LRU stack distance", as detailed further in Section 5), i.e.,

$$d_0^{icache} = T_0^{L2latency} \cdot f^{iL1}(D_0^{instr-reuse}) + T_0^{DRAMlatency} \cdot f^{iTLB}(D_0^{instr-reuse}), \quad (2)$$

where $T_0^{L2latency}$ and $T_0^{DRAMlatency}$ are the L2 and DRAM access latencies in cycles, $D_0^{instr-reuse}$ is the application's instruction reuse distribution, characterized with tools such as those described in Section 5, and f^{iL1} and f^{iTLB} are functions from the distribution values to the i-L1 and i-TLB miss rates.

The branch misprediction delay, d_0^{branch} , is computed as the sum of the pipeline length, n_0^{pipe} , and the expected time between a branch misprediction and its resolution; the latter is shorter if there are shorter dependence chains [33, 51], and is modeled as the sum of the frontend pipeline's capacity (length times width), n_0^{front} , and the reorder buffer size, n_0^{ROB} , divided by the application's harmonic mean ILP,

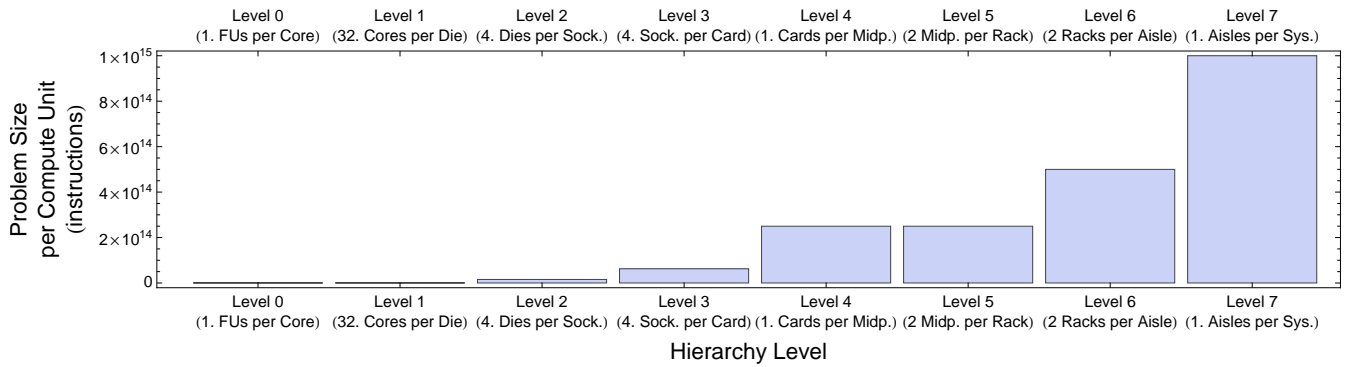


Figure 6. Illustrative example of work per system hierarchy level, for the correlator working example of Section 3, mapped to one example system (a single IBM BlueGene/P rack). It is a function of the hardware concurrency at each level, coupled to the application's available parallelism at different levels.

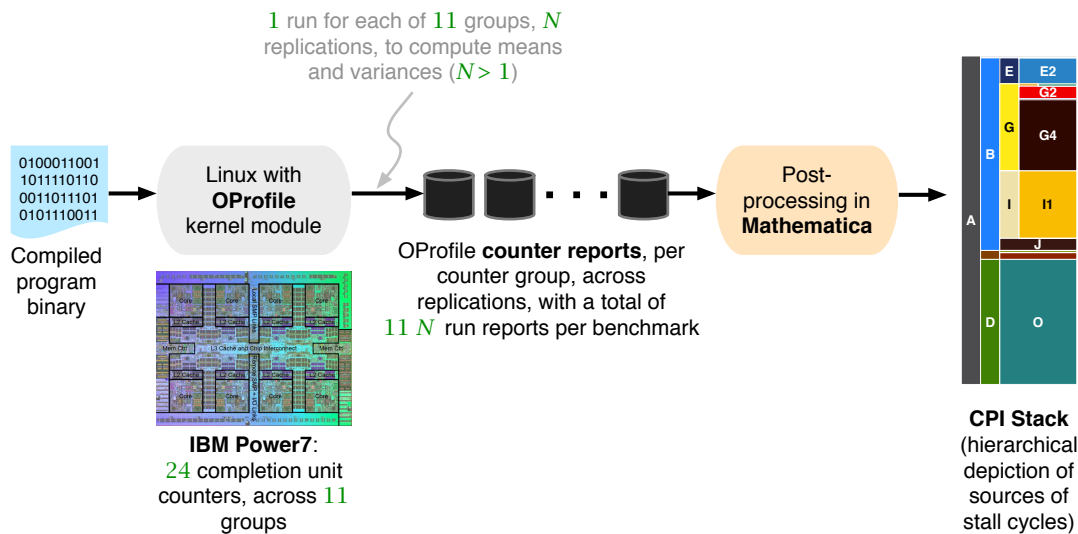


Figure 7. Applications are executed on an IBM Power7 system to obtain hardware performance counter data, which is then post-processed to obtain the CPI stack visualizations.

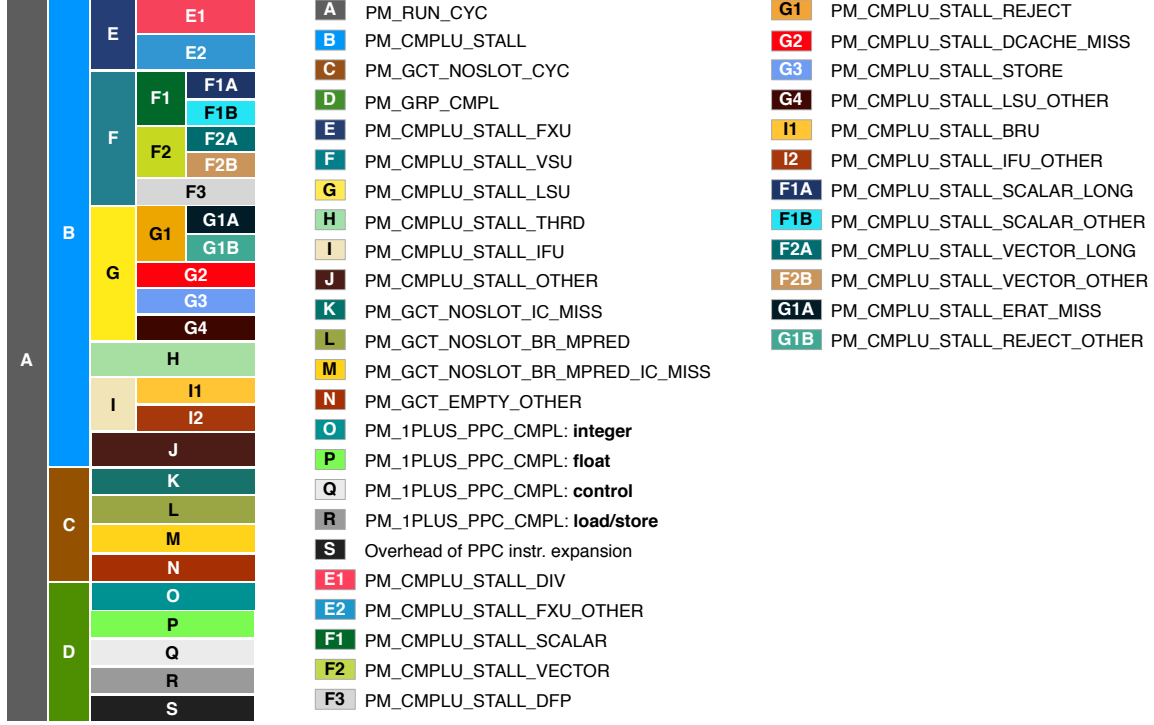


Figure 8. The Power architecture provides a comprehensive set of hardware performance counters (counter names shown in the legend of the figure), designed specifically to enable the construction of a detailed CPI stack. Such actual CPI stacks can be compared to the CPI stacks estimated from EXABOUNDS (Figure 9).

$\hat{n}_0 = \min(n_0, ALP_0)$, i.e.,

$$d_0^{\text{branch}} = n_0^{\text{pipe}} + \frac{n_0^{\text{front}} + n_0^{\text{ROB}}}{\hat{n}_0}. \quad (3)$$

The miss penalty for L1 data cache misses is computed from the data reuse histogram distribution and the L1-to-L2 latency, i.e.,

$$d_0^{\text{L1data}} = T_0^{\text{L2latency}} \cdot f^{\text{L1D}}(D_0^{\text{data-reuse}}). \quad (4)$$

The miss penalty for L2 data cache misses is computed as the product of the L2 miss probability, and the difference between the main memory latency and reorder buffer size divided by the program ILP; the latter term captures the fact that applications with longer dependence chains (smaller ILP), will take longer filling the reorder buffer, and, until that happens, the mean IPC would not have fallen below its non-miss-event equilibrium:

$$d_0^{\text{L2data}} = (T_0^{\text{DRAMlatency}} - \frac{n_0^{\text{ROB}}}{\hat{n}_0}) \cdot f^{\text{L2D}}(D_0^{\text{data-reuse}}). \quad (5)$$

The above relations do not capture the effects of prefetching or of software-managed memories. While such hardware optimizations are important, they are not pervasively implemented in microprocessors. Thus, in arguing for generality and simplicity, we currently exclude them from EXABOUNDS. These bounds build on prior work [33, 51], but we extend them in a number of important ways described below.

The rate of completion of a given workload is not only dependent on the compute operations to be completed, but

also on the data motion bandwidth needs of an application being met. The components of instruction delay due to the memory hierarchy presented in Equations 2 to 5 however implicitly assume that while accessing DRAM has a latency $T_0^{\text{DRAMlatency}}$, an unlimited number of such accesses may be initiated in aggregate by the concurrent hardware units at a given level in the hierarchy. In practice, at any level of the hierarchy, there will be an additional queuing delay component, due to real systems having limited memory bandwidth.

For an application with data motion bandwidth *requirement* of $B_{app,i}^{\text{dmo}}$, a hardware platform that *provides* B_i^{dmo} data motion bandwidth per pin, and degree of concurrency constrained by both application parallelism and hardware resources $\hat{n}_i = \min(n_i, ALP_i)$, the queuing delay can be approximated by Little's law [54]:

$$d_0^{\text{BWq}} = \frac{B_{app,0}^{\text{dmo}}}{\min\left(B_{app,0}^{\text{dmo}}, \frac{B_0^{\text{dmo}} \cdot \hat{n}_0}{\hat{n}_0}\right)} \cdot \frac{1}{F_0^{\text{dmo}}}. \quad (6)$$

The numerator on the right hand side of the above expression represents queue length, and the denominator represents the arrival rate of memory requests to the (bandwidth-limited) memory channel.

The overall instruction throughput at the processor core

Table 3. Notation for hardware parameters. The example values are based on observations of actual systems; specific references are omitted for brevity.

Description	Notation	Working Example
Data motion bandwidth per signal line at level i	B_i^{dmo}	1 Gb/s per pin at $i = 0$
Data motion latency at level i	T_i^{dmo}	100 ns at $i = 0$
Volatile Memory at level i	M_i	16 GB equiv. DRAM per core, at $i = 0$
Datapath width in bits	n_0^{bits}	64
Pipeline depth	n_0^{pipe}	10
Func. units per core (integer)	n_0^{INT}	8
Func. units per core (single float)	n_0^{SFP}	8
Func. units per core (double float)	n_0^{DFP}	8
Transistors per core (or, Programmable cores per die)	n_0^{xcore}	40 million
Dies per socket	n_1^{die}	8
Resistance, power supply pin + pad	R_2^{supply}	$1\mu\Omega$
I/O interfaces per card	n_3	8
Sockets per card	n_3	2
Cards per rack unit	n_4	10
Rack units per rack	n_5	6
Racks per aisle	n_6	16
Aisles per system	n_7	8
Clock frequency	f_i	4×10^9 Hz at $i = 0$
Supply voltage	V_i	1.2 V at $i = 0$
Threshold voltage	V_t	0.4 V
Parameters based on technology improvements over time		
Transistors per die	n_0^{xpc}	200 million/mm ²
Area per die	n_1^{apd}	200 mm ²
Transistors per unit area	n_0^{xpc}/n_1^{apd}	2 million/mm ² at 90 nm node
Power supply pins per socket	n_2^{supply}	150
I/O pins per socket	n_2^{io}	64

level is thus:

$$d_0 = \frac{1}{\hat{n}_0} + d_0^{icache} + d_0^{branch} \cdot F_0^{control} + (d_0^{L1data} + d_0^{L2data}) \cdot F_0^{memory} + d_0^{BWq}, \quad (7)$$

where F_0^{type} is the instruction mix fraction for each instruction of type *type* (e.g., integer, floating-point)¹.

From the foregoing relation for total amount of work (in instructions executed), needed to solve a given problem, and from the above relation for delay per instruction, rate of problem solutions per second for a given problem size, C_i when seen from level i , is thus

$$C_i = \frac{L_i + L_1^{ovhd}}{d_0} \text{ instructions/second.} \quad (8)$$

L_1^{ovhd} is the overhead incurred per computing unit, in parallelization at level i . The delay per instruction is only formulated for level 0, and for all higher layers, the additional computational overheads are captured by the term, L_1^{ovhd} , representing overhead work at a given layer. Equation 8 expresses the compute performance of a system as a closed-form hierarchical expression, as d_i is a function of

¹This division of the delay per operation into components due to latency, bandwidth and parallelism is similar to the approach of Burger et al. [14].

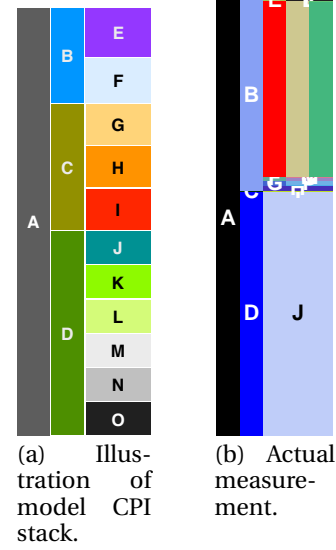


Figure 10. Comparison of EXABOUNDS performance breakdowns (left) to that measured using hardware performance counters on the IBM Power7 (right), for the simple algorithm corresponding to the correlator computational problem in Figure 2.5.

C_{i-1} . It succinctly captures the interaction between hardware properties (B_i^{dmo} , $n_i^{i/o}$ pins, n_i , and T_i), application properties (L_{sys} , $B_{app,i}^{dmo}$, F_i^{dmo} , and ALP_i) and system software overheads (L_i^{ovhd}). When best-case (i.e., smallest) values of F_i^{dmo} and T_i^{dmo} are used in the relation for d_i , one obtains an upper bound on C_i , and conversely, worst-case values provide lower bounds on C_i .

For the working example application, whose instruction mix breakdown is shown in Figure 10(a). Figure 10(b, c) show the estimated C_0 (core-level compute throughput, i.e., CPI), based on these models, compared to the measured CPI from hardware performance counter measurements on the IBM Power7, an 8-core/32-thread system whose relevant properties in the context of the EXABOUNDS model are listed in Table 4, and illustrated in Figure 11. Figure 10(d, e) show the estimated C_2 (socket-level compute throughput), based on these models, compared to the measured socket-level instruction throughput from hardware performance counter measurements on the IBM Power7, for the working example application.

4.3 Bandwidth versus cache size tradeoffs

At each level, i , of a computing system hierarchy, there is an inherent tradeoff between the effective available bandwidth (B_1^{dmo}), and the size of buffer storage. These buffers range from the registers within a core microarchitecture, to the cache hierarchy (L1, L2, and deeper). For a given buffer size at a given level of the system (e.g., L1 cache size), an application reuse distance distribution can be used to estimate the miss rate and hence the required memory bandwidth. These reuse distance distributions can be collected using techniques described further in Section 5. In the absence of application-specific reuse distance distributions, a reasonable rule of thumb that captures the empirically observed relation between cache size (S_i^{level}) and the effective

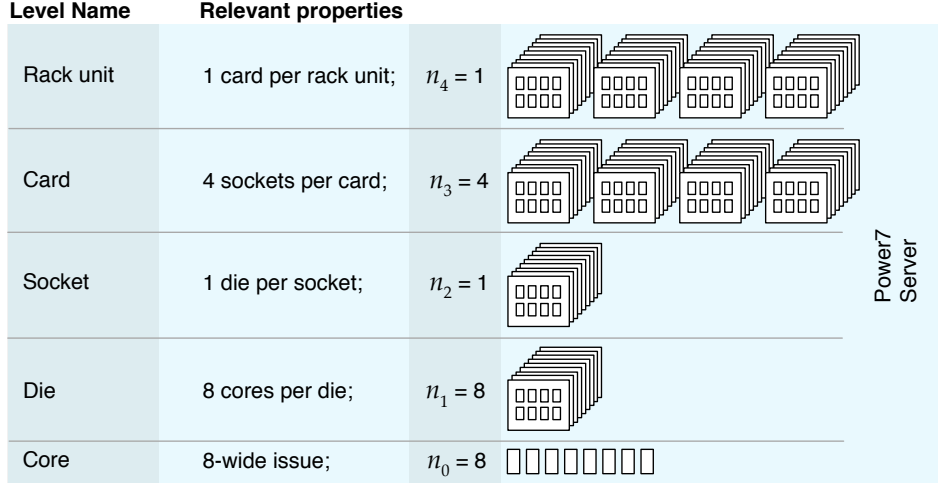


Figure 11. Illustration of a system built out of Power7 processors, in the context of the hierarchy presented in this article.

Table 4. Values for parameters of the EXABOUNDS model, corresponding to a single-socket IBM Power7 system. The L3 bandwidth and latency values in parentheses are for the case where the L3 latency is taken to be that of the non-local adaptive victim L3.

Parameter	Description	Value for IBM Power7
$B_0^{dmo} \cdot n_0^{i/o}$	L2 bandwidth	256 GB/s
$B_1^{dmo} \cdot n_1^{i/o}$	L3 bandwidth	128 (512) GB/s
$B_2^{dmo} \cdot n_2^{i/o}$	DRAM bandwidth	100 GB/s
T_0^{dmo}	L2 latency	8 clks
T_1^{dmo}	L3 latency	25 (125) clk
T_2^{dmo}	DRAM latency	400 clk
M_0	L2 size	256 kB
M_1	L3 size	32 MB
n_0^{bits}	Datapath width	64
n_0^{pipe}	Pipeline capacity	120
n_0^{front}	Frontend capacity	6 · 9
n_0^{ROB}	Reorder buffer capacity	48
n_0^{INT}	# integer units	4
n_0^{DPFP}	# DP FP units	4
n_1^{cor}	Cores per die	8
n_2^{die}	Dies per package	1
f_0	Core clock frequency	4×10^9 Hz
n_1^{apd}	Area per die	567mm^2
n_0^{tpc}	Transistors per die	1.2×10^9

available bandwidth [39], is:

$$B_1^{dmo\text{-effective}} = \frac{K_1^S}{\sqrt{S_1^{\text{level}}}}. \quad (9)$$

4.4 Unification of existing models and terminology

The foregoing relations for compute performance as a function of parallelism, workload size, and data movement bandwidth encompass several existing bounds, rules of

thumb, and terminology, of application scaling on parallel systems.

4.4.1 Strong and weak scaling

The term *strong scaling* is used in the computing systems literature to refer to improvement in performance of a system as a function of hardware concurrency, at a fixed problem size [82]. This corresponds to the increase in C_i (from the viewpoint of any level, i) as a function of the total available hardware concurrency at level i and below ($\prod_{l=0}^i n_l$), at fixed problem size L_i . To capture strong scaling properties, we introduce the notion of the *strong scaling coefficient*

$$\frac{dC_i}{dn_z}, \quad (10)$$

where $z \leq i$ is the level at which hardware concurrency is scaled. Similarly, the term *weak scaling* is used to refer to improvement in performance of a system with increasing hardware parallelism, if the problem size is permitted to increase, and may be analyzed both numerically and analytically as in the case of strong scaling above. In an extreme case of weak scaling, where the degree of hardware concurrency remains fixed and problem size increases, weak scaling can be represented by the *weak scaling coefficient*

$$\frac{dC_i}{dL_i}. \quad (11)$$

This makes the relation between weak scaling and data-level parallelism apparent, as both relate to increases in performance as problem size is increased.

4.4.2 Isoefficiency and Gustafson's law

The rate at which a system's input size must grow with increasing processor count in order to maintain the speedup observed at a smaller (baseline) number of processors, is captured in the concept of the *isoefficiency* [36] (scalability) for a given pairing of an algorithm to a platform. For a given algorithm and hardware, the isoefficiency function can be formulated from the expressions for C_i , as

$$\frac{dL_i}{dn_z} \quad \text{such that} \quad \frac{C_i(L_i, n_z)}{C_i(L_{i\min}, n_{z\min})} \geq \frac{C_i(L_{i^0}, n_{z^0})}{C_i(L_{i\min}, n_{z\min})}.$$

In the above, $z \leq i$ is the level at which hardware concurrency is scaled, n_{z^0} and L_{i^0} are the baseline hardware con-

currency and problem size to which scaled speedup is being compared, and $n_{z,\min}$ and $L_{i,\min}$ are the minimum hardware concurrency and problem size. The term $\frac{dL_i}{dn_z}$ can be expressed in terms of the expression for computation throughput (Equation 8) and that for strong and weak scaling (Equations 10 and 11), as

$$\frac{dL_i}{dn_z} = \frac{dL_i}{dC_i} \cdot \frac{dC_i}{dn_z}, \quad (12)$$

i.e., the isoefficiency is related to the ratio of the strong scaling coefficient to the weak scaling coefficient. Gustafson's law [37], like isoefficiency analysis, captures the fact that the fraction of an algorithm's execution that is not parallelizable may effectively decrease with increasing problem size.

4.4.3 Amdahl's, work, and span laws

Amdahl's law [2] captures the bound on performance improvement as a result of some portion (AMDAHL_p) of an algorithm's dynamic execution stream being (infinitely) parallelizable (ALP = ∞), and the remainder being completely serial (ALP = 1), *assuming problem sizes remain fixed*.

The "fraction of an application's workload that is parallelizable", as often attributed to Amdahl's law, is not always a meaningful quantity. It is meaningful in applications which have only two phases or components—one which has no parallelism, and a second which has unlimited parallelism (more than any degree of hardware concurrency that will ever be paired with it). Most applications however have a variety of phases or components, each with varying amounts of parallelism, with application parallelism not always exceeding the available hardware concurrency. The EXABOUNDS model however enables us to define a proxy parameter for the Amdahl parallelism, AMDAHL_{pz}, since speedup can also be expressed in terms of the relations derived in the foregoing sections for C_i , at fixed L , i.e.,

$$\begin{aligned} \text{speedup} &= \frac{C_i(L_i, n'_z)}{C_i(L_i, n_z = 1)} \\ &= \frac{1}{(1 - \text{AMDAHL}_{pz}) + \frac{\text{AMDAHL}_{pz}}{n'_z}}. \end{aligned} \quad (13)$$

Thus

$$\text{AMDAHL}_{pz} = \frac{n'_z (C_i(L_i, n'_z) - C_i(L_i, n_z = 1))}{(n'_z - 1) C_i(L_i, n'_z)}. \quad (14)$$

It is therefore possible to calculate the effective "Amdahl parallel fraction", given an observed improvement in compute throughput from $C_i(L_i, n_z = 1)$ to $C_i(L_i, n'_z)$. The calculated fraction will however depend not only on the change in performance ($C_i(L_i, n'_z) - C_i(L_i, n_z = 1)$) with increasing hardware concurrency, but also on the baseline system size (n_z).

Contrary to the (useful) simplifying assumptions made by Amdahl's law analysis, many applications cannot be split into two well-defined phases, with one having unlimited parallelism and the other being entirely serial. When ALP is greater than n_i at any level i , then the parallel execution time is limited by the ratio of the problem size to the number of processors, i.e., $\hat{n}_i = \min(n_i, ALP_i)$. This corresponds to Leiserson's *work law* [21, 40], when ignoring the constraints on data movement captured in the expression for d_i above. Similarly, when n is greater than ALP at any level (i.e., $\hat{n}_i = ALP_i$), execution time is limited by the ratio of the

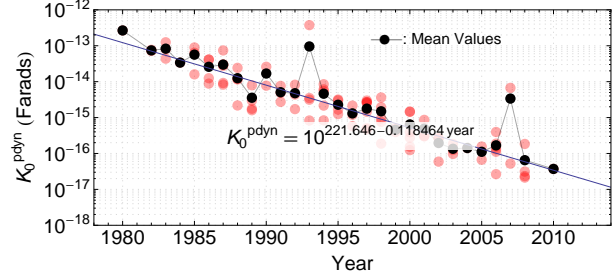


Figure 12. Empirical values of $K_0^{\text{p dyn}}$, based on reported measurements in the solid state circuits literature.

problem size to the amount of ALP, corresponding to the *span law* [21, 40].

4.4.4 The roofline model

The roofline model [85] bounds the maximum attainable performance (FLOPS_{peak}) of an application as a function of the peak memory bandwidth (BW_{peak}) and operational intensity (OI):

$$\min(\text{FLOPS}_{\text{peak}}, \text{BW}_{\text{peak}} \times \text{OI}). \quad (15)$$

The operational intensity at which the two terms in the min expression are equal corresponds to the "ridge point" between the flat and slanted parts of the "roof" in the roofline model.

The *operational intensity*, *peak memory bandwidth*, and *peak floating-point performance* correspond, respectively, in EXABOUNDS, to F_i^{dmo} , B_i^{dmo} , and $n_i \cdot f_i$. The coupling of parameters to implementation constraints in EXABOUNDS therefore enables the investigation of new tradeoffs, such as the effect of power delivery constraints (which limit the number of package pins left over for memory bandwidth) on the position of the roofline ridge point.

The ridge point represents the point (in terms of the operational intensity) where performance shifts from being memory bandwidth dominated, to being compute dominated. An application operating with operating intensity at the ridge point is therefore "balanced", being equally constrained by memory bandwidth and computation. Thus, a ridge point at lower operating intensity values means an application on the platform has a smaller need to reach a high flops per DRAM access.

The ridge point can be formulated analytically in the context of EXABOUNDS as the point when

$$n_i \cdot f_i = B_i^{\text{dmo}} \cdot F_i^{\text{dmo}} \quad (16)$$

Applying the constraint imposed by Equation 16 to any of the components of the EXABOUNDS model (e.g., compute throughput or power dissipation) enables the formulation of an expression for that quantity in a balanced system, potentially yielding new insight into the properties of balanced systems.

4.5 Hierarchical power dissipation and technology trends

The performance bounds presented in the previous section capture the dependence of execution time on application and hardware properties, and can be linked to bounds on average power dissipation, energy usage, and resulting subsystem temperatures for a given workload. In what follows, we focus on power and energy estimates, excluding

temperature estimation from the analysis as it will depend significantly on the nature of packaging and cooling solutions employed; in the presence of models for the thermal resistance of system packaging, the analyses we provide may be used as input to estimating system temperature.

4.5.1 Core dynamic and leakage power

Contemporary computing system integrated circuits (ICs) are implemented almost exclusively in CMOS technology. For (bulk) CMOS, the power dissipated is a function, among other things, of the supply voltage, which we denote as V_0 , the threshold voltage, V_t , clock frequency, f_0 , and the switching activity, s_0 . The switching activity, which is the fraction of the maximum logic transitions achieved per cycle, is a function of the application or algorithm being executed. The power dissipation of core logic (ALUs, etc.) and on-chip memory structures (pipeline latches, register file, CAM structures, cache SRAMs) can thus be modeled as [50, 44]:

$$P_0^{cor} = \underbrace{K_0^{pdyn} \cdot V_0^2 \cdot f_0 \cdot s_0}_{\text{dynamic power}} + \underbrace{V_0 \cdot K_0^{pleak} \left(e^{\frac{q(V_{gs}-V_t)}{k_{leak} \cdot k \cdot T}} \right)}_{\text{static/leakage power}}. \quad (17)$$

V_{gs} is the gate voltage, k is Boltzmann's constant, q is the electron charge constant, and T is temperature in Kelvin; K_0^{pdyn} , K_a^{leak} , and K_0^{pleak} are functions of the number of transistor properties and transistor technology. K_0^{pdyn} is estimated empirically by fitting the first term on the right hand side of Equation 17 to measurements of die power dissipation over a range of clock frequencies or voltages, under an assumption of a fixed value for the activity factor. Figure 12 shows the K_0^{pdyn} , estimated empirically in this manner, from reported measurements for a collection of 130 digital designs. The resulting empirical model for K_0^{pdyn} can be used to guide predictions of values for future systems, and similar empirical data sets can be used to guide predictions for K_0^{pleak} and K_a^{leak} . System architects employing this framework need only supply values of the independent variables (operating temperature, operating voltage, operating clock frequency).

The supply voltage (V_0) and clock frequency (f_0) in Equation 17 are however not independent: f_0 is limited to a maximum value of f_0^{max} , by the Sakurai alpha-power-law voltage-frequency dependence [73],

$$f_0^{max} = K_0^V \cdot \frac{(V_0 - V_t)^{K_\alpha}}{V_0}. \quad (18)$$

K_0^V and K_α are process technology and design- and transistor-sizing-dependent constants, to which we do not attach a strict physical interpretation in what follows. V_t is similarly treated without a strict physical interpretation; for designs which operate in the super-threshold regime, it corresponds to the device threshold voltage.

Figure 13(a) and Figure 13(b) plot values of K_0^V and K_α for several digital designs across several years of published data from the solid state circuits research literature. Each point in the plot was obtained by curve-fitting the data from a "Shmoo" plot to Equation 18; a subset of those Shmoo plots is shown in Figures 13(c) and 13(d). From Figure 13,

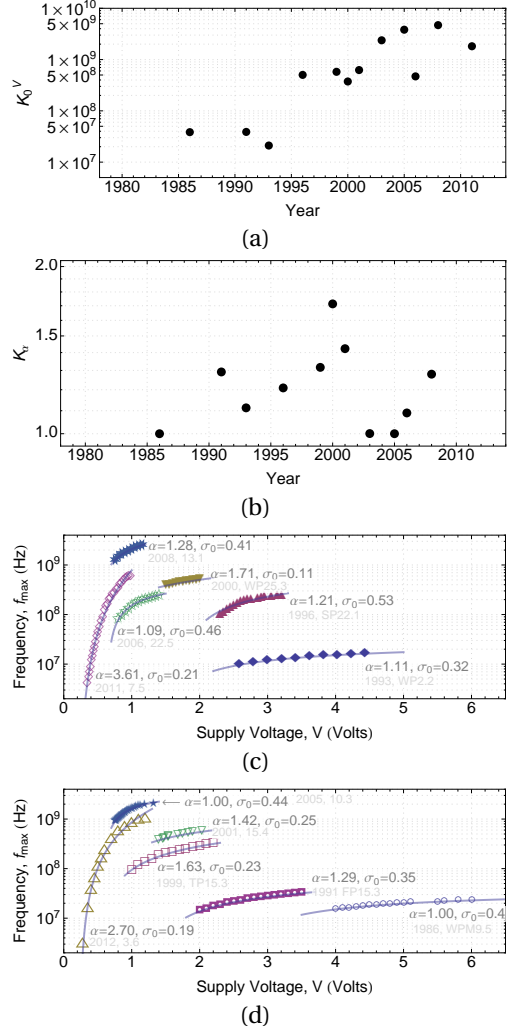


Figure 13. Empirical values of (a) K_0^V and (b) K_α , and a subset of the frequency-versus-voltage curves from which they are derived (c-d). In the latter, the points denote measurements, and the lines represent the best-fit to the model of Equation 18.

it is evident that while the value of K_0^V is closely correlated to the system's year of introduction (shown in figure), implementation process and peak clock frequency (not shown for brevity), the value of K_α is more design dependent, particularly for sub-90 nm designs (after year 2000 in the figure). The ranges and trends from Figure 13 can be used as estimates; the values necessary to achieve desired whole-system properties can serve as design goals in the eventual system implementation.

4.5.2 Resistive losses between die and socket

In many processor designs, the use of lower operating voltages across process generations is coupled with designs employing larger numbers of transistors. This is illustrated empirically for the same collection of microprocessor and other digital designs discussed in prior sections, in Figure 14.

When the overall power consumption stays the same (e.g., due to the desire to use the increased transistor budget

$$P_i = \frac{n_i \cdot P_{i-1} + P_i^{glu} + P_i^{ohmic} + K_i^{dmoPwr} \cdot B_i^{dmo} + K_i^{MemPwr} \cdot M_i}{\eta_i}, \quad (19)$$

$$\forall i \geq 1,$$

Figure 16. Hierarchical power model.

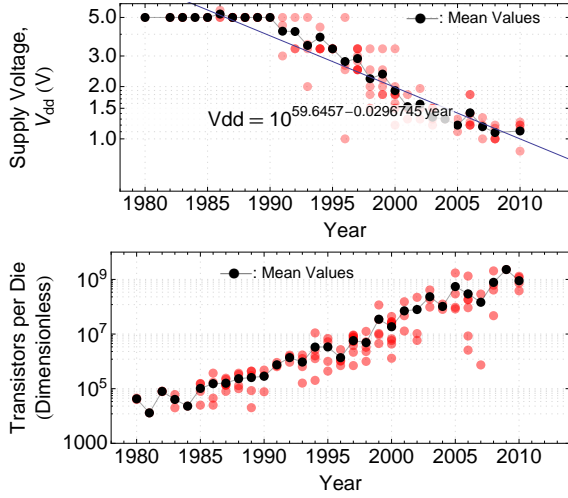


Figure 14. Empirical transistor density trends.

to pack more features into an implementation), a reduction in the operating voltage leads to higher power supply currents (Figure 15), resulting in larger resistive losses in the power supply pins. These losses are typically mitigated by using a larger number of power supply pins (i.e., increasing n_2^{supply}) to reduce the current per pin. When the number of pins that are available per package are limited, this leads to restrictions on the number of pins available for other purposes, e.g., for I/O and memory. There is therefore a direct link between power supply currents and restrictions on data movement bandwidth for a packaged die [78]. Integrating an increasing number of cores per die, as is the often assumed trend for future performance growth of computing systems [3, 10], will thus place increasing pressure on packaging (pins).

An alternative to increased integration at the die level, is the design of systems with integration occurring more at the system-level rather than at the die-level. For example, Chang et al. [20] assume that increased resistive power loss will not be an issue for deeply-voltage-scaled devices, and assume that such systems will achieve parallelism via system-level scaling and not scaling number of devices per die. However, due to the need to mitigate packaging cost in large-scale systems, there will be an ever increasing desire to maximize the number of devices per die, at least to the limit permitted by memory bandwidth limits and manufacturing yield losses. For all of these reasons, it is important to capture the interaction between power delivery and resistive loss constraints, and package-level I/O bandwidth requirements.

The resistive power losses incurred, P_i^{ohmic} , whether at *controlled-collapse chip connection (C4)* solder bumps on a die or in bond wires in a package (within level $i = 2$), or in current-carrying conductors at the rack level (within level

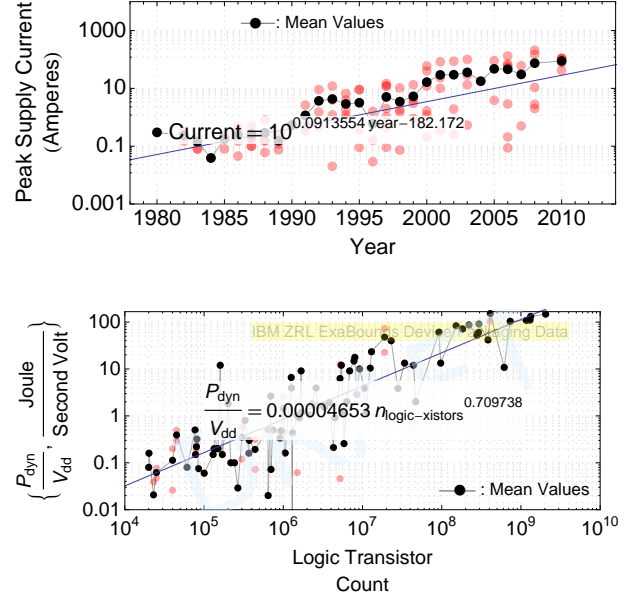


Figure 15. Empirical supply current trends.

$i = 5$), are formulated in generalized form as

$$P_i^{ohmic} = \left(\frac{I_i^{supply}}{n_i^{supply}} \right)^2 \cdot R_i^{supply} \cdot n_i^{supply} \quad (20)$$

$$= \left(\frac{n_i \cdot P_{i-1}}{n_i^{supply} \cdot V_{i-1}} \right)^2 \cdot R_i^{supply} \cdot n_i^{supply}.$$

For example, for P_2^{ohmic} , with pad resistances of upwards of 40 m Ω [55], assuming combined pad, C4 and pin resistances of 300 m Ω , and n_2^{supply} of near 600 in high-performance designs [46], the data from Figure 15 implies excessive resistive power losses in packages of up to 5 W.

4.5.3 Data movement power

Table 4.5.2 lists cost assumptions for data movement on-die, off-die, at the board level, and at coarser granularities, along with the projected achievable signaling rates, for both electrical and optical interconnects. The data in the table is based on the 2011 ITRS roadmap [34] as well as data reported in the research literature [7].

The power efficiency of electrical interconnect links typically decreases with increasing bit rates, making it sometimes desirable to use many I/O pins at a lower signalling rate, rather than one at a high rate. We capture this trade-off by modeling bandwidths in terms of the number of pins available for signaling, the bit rate per pin, and the power dissipation as a function of bit rate.

Dunning et al. [29] cite power usage at between 40 and 200 mW per Gb/s, with performance on current and planned state-of-the-art implementations ranging from as

Table 5. Projected properties of electrical and optical global on-chip interconnects as a function of year, 2012–2022. For optical interconnects, the table below assumes the die can always be crossed in a single hop.

	Electrical	Optical
Signaling voltage	0.8 V – 1.8 V	N/A
Signaling rate	2 Gb/s – 6 Gb/s	TBD
Resistivity	aluminum: 3.3 mΩ·cm; thin-film copper: 2.2 mΩ·cm	N/A
Capacitance	TBD	N/A
Energy per bit per cycle	$\max(2, -1.32E8 \cdot (y - 2006)^{4.13E-7} + 1.32E8)$	90 aJ [7]
Cycles to cross die	$0.17 \cdot y^{2.85} + 35.45$	1

low as 1 mW per Gb/s, to 25 mW per Gb/s. Based on I/O designs from Intel [18, 6], which span the 5–20 Gb/s range, we employ the model of increase in power with bit rate of

$$\text{Power dissipation per Gb/s} = 0.0170 \cdot \text{rate}^{1.86} + 2.36 \text{mW}. \quad (21)$$

Dunning et al. predict an eventual limit to pin bandwidth for DDR DRAM interfaces of between 2.5 and 3.5 Gb/s per pin, given the nature of DDR mechanical and electrical connections (sockets, multi-drop channel, etc.). If using board-soldered GDDR-like DRAM ICs, they cite an achievable upper limit of between 5 and 6 Gb/s per pin. The above are all for single-ended signaling; a move to differential (current-mode) signaling will permit additional per-pair signal rates, but requires a doubling of the number of pins per logical signal channel.

In addition to estimating the growth rate of power dissipation with signaling rate, it is also useful to capture how signaling rate and power dissipation vary as a function of process geometry. Signaling rate increases (approximately linearly) with decreasing capacitance relative to the strength of gate drivers, while the energy per bit transmission is a function of the product of capacitance and the square of signaling voltage. Horowitz et al. [45] state a trend in signaling delay with technology trend of 500 ps for a FO-4 gate delay, per micron of gate length. These effects are captured by the models listed in Table 4.5.2, from the 2011 ITRS report, which also take into account floorplanning constraints, such as the need to pipeline long global wires.

The energy cost for communication may also be expressed in terms of the energy per bit, per millimeter of trace distance; contemporary values are on the order of 18 fJ/bit-mm for on-chip electrical links, approximately 2 pJ per bit for PCB traces, and approximately 10 pJ per bit for optical communications, but not including the required serialization/deserialization (SerDes) and clock data recovery (CDR) [70].

4.5.4 Overall power model

The power dissipation at levels above the processor core are thus formulated as shown in Equation 19, in Figure 16. In Equation 19, η_i captures the loss in power delivery efficiency due to power supply regulation and cooling overheads. The term involving the memory capacity at a given level captures the fact that for modern large-scale systems, a significant component of the system power dissipation is contributed by the memory devices (DRAM), for both the memory cell storage energy (e.g., in DRAM cells) and in the memory access logic, and data and control signaling. P_i^{glu} captures the power dissipation by peripheral circuits required at a given hierarchy level, and the product $K_i^{dmoPwr} \cdot B_i^{dmo}$ captures the power associated with commu-

nication.

4.6 Whole-system reliability and failover model, and device reliability trends

At scales of millions of processor sockets, system reliability, as captured by the mean time between failures (MTBF), will be of great interest in future large-scale systems. In what follows, we assume that faults in a given level are independent and identically distributed, with exponentially distributed inter-arrival times T , having mean failure rate λ_i for the dominant source of failures at level i . The distribution of inter-arrival times, which take on instance values t , is thus given by

$$\Pr(T = t) = \lambda_i e^{-\lambda_i t}. \quad (22)$$

The mean failure rate (λ_i) in practice equates to the empirically measured *failures in time* (FIT), with 1 FIT corresponding to one failure every billion hours ($\lambda = 10^{-9}$). Individual integrated circuits, whether DRAM or compute ASICs, when designed with resilience in mind, can have FIT rate of less than 100, while components with moving parts, such as power supplies and fans may have fit rates an order of magnitude higher.

Treating time as continuous, the mean time to failure of an *entire* level, i , of a system (e.g., an entire core, socket, rack, etc.), is related to the performance and power models through the variable n_i , and is given by

$$\begin{aligned} MTBF_i &= \int_0^{\infty} t \cdot \lambda_i e^{-\lambda_i t} \cdot (1 - \lambda_i e^{-\lambda_i t})^{n_i-1} dt \\ &= \frac{H_{PFQ}(1, 1, 1 - n_i, 2, 2, \lambda)}{\lambda}. \end{aligned} \quad (23)$$

H_{PFQ} is the generalized hypergeometric function, and results from the expression of the integral in terms of standard closed-form special functions. Using the closed-form expression in terms of the hypergeometric special function enables efficient evaluation of the MTBF at level i . The above relation captures the fact that $n_i - 1$ devices must simultaneously be in working condition in the case where *only* one device fails. Intuitively, larger per-device failure rates, and larger numbers of devices n_i , lead to exponentially smaller times between failures. Faults in lower layers are assumed to be mitigated by design techniques, and do not propagate to lead to a fault in all devices at the layer above in the system hierarchy. The translation of this per-hierarchy-layer MTBF into a system-level MTBF will depend on whether the machine in question is used as a capability or capacity machine, and is thus not discussed further here.

4.6.1 Failover

When failures are mitigated by using *failover*, the workload of a failing device must be taken over by the remaining

devices in a system. Thus, the finer the granularity (larger n_i) of concurrency, the smaller the impact of a single failing device on the system's performance. This is captured by refining the expression for the work, L_i per component at level i , to include the additional work from failed components that must be carried, yielding

$$L_i = \frac{L_{i+1}}{\min(n_i - \text{fail}_{\text{ovhd}^i}, \text{ALP}_i)}, \forall i < 7. \quad (24)$$

The parameter $\text{fail}_{\text{ovhd}^i}$ is the ratio of the MTTR to MTBF for a given level i , assuming at most singly-occurring failures, and corresponds to the fraction of compute unit equivalents lost.

However, smaller granularities, as shown above, lead to poorer MTBF performance. There is thus a tradeoff between decreased MTBF with more cores and an associated potential for increased performance in the presence of available application-level parallelism, and the benefit of lower overheads in the presence of an occurred failure. Reddi et al. [48] give quantitative evidence of such a tradeoffs in the Microsoft Bing server, in investigating the tradeoff between using a larger number of lower-performance Intel Atom processors, versus a smaller number of higher-performance Intel Xeon processors for a large-scale search system.

Another example of a real-world reliability study is presented by Shroeder, Pinheiro, and Weber [75], who give empirical measurements of DRAM failure rates (both correctable errors (CEs) and uncorrectable errors (UEs)) in the wild. They show that:

- Error rates were not correlated to temperature *for the range of temperatures occurring within the server deployments studied*, after accounting for the dependence on utilization.
- Error rates increased with age.
- Error rates seemed to follow a power law, with DIMMs that have witnessed an error being more likely to witness an error in the future. This, along with the utilization-dependence was postulated to be due to the dominant failure mode being permanent hard errors, as opposed to soft errors: if errors are hard, then the more memory is used, the more likely it is that an error will be detected. Likewise once an error in a bit is hard, the more likely an error in the bit will be detected again.

Reliability may also degrade with system age, due to age-related effects such as oxide traps and negative bias temperature instability (NBTI), hot carrier injection (HCI), and electromigration. ExaBounds does not however capture such aging effects, and extension to do so is a subject of future research.

4.7 Hierarchical cost bounds and trends

For large-scale computing systems, costs of purchase and operation play a role of equal importance to the traditional concern of computation performance. The *cost of purchase* includes compute hardware cost, as well as building infrastructure costs needed for running such large systems, and often include the cost of power distribution, redundant power, generators, cooling infrastructure, and so on. The dimensioning of such infrastructure is dependent on the power dissipation profile of the computation hardware it encloses. The *cost of operation*, while including components such as wages for operators and software license

costs, is dominated by energy supply costs. A holistic model of large-scale systems must thus capture both the total cost of acquisition (TCA), and the total cost of ownership (TCO) of a system. Such a formulation, in the context of the framework introduced thus far, is presented in Equation 25.

In Equation 25, $K_i^{\$NRE}$ and $K_i^{\$pkg}$ are the costs associated with non-recurring engineering costs and hardware/packaging respectively, and make up the TCA; at the die level, $K_i^{\$pkg}$ is interpreted as the cost per transistor. The TCO comprises the costs associated with power dissipation in the interconnect, computation, memory, and cooling, and are a function of the time, $\frac{L_{\text{sys}}}{C_T}$, needed for execution of an application with input L_{sys} . Examples of the constants in Equation 25 at various hierarchy levels include $K^{\$pwr}$ (4.14×10^{-8} USD per Joule), $K_0^{\$Si}$ ($\approx 8 \times 10^{-19}$ USD per Hz, and predicted to drop to $\approx 2 \times 10^{-19}$ USD per Hz in the next five years [65]).

For a system to be cost-effective, the rate of increase of cost with added hardware resources (the *costup* [86]) should be smaller than the rate of increase of performance (speedup) with the same increase of resources. From Equations 8 and 25, taking the derivatives with respect to a system growth parameter (e.g., number of cores at the die level), one may formulate constraints on cost-effective systems.

4.8 Constraints: power delivery, cooling, and area

Constraints play an important role in the early stages of most computing system designs, since they are often one of the few aspects which are pre-determined. Based on the hierarchical EXABOUNDS analytic framework, any derived value or metric at a given layer in the hierarchy may also be treated instead as a constraint on its constituent parameters; this is detailed further in the description of the tool implementation in Section 5.

4.8.1 Power delivery and cooling constraints

At a fixed supply voltage, power dissipation (and hence the cooling requirements) grows approximately linearly with clock frequency (Equation 17), and hence with performance. However, achieving higher *ranges* of clock frequency (and performance) require operation at higher supply voltages as dictated by Equation 18 and shown empirically for many designs in Figure 13. In particular, for a system that always operates at the maximum permitted frequency for a given supply voltage, power dissipation grows approximately quadratically with increasing voltage/frequency operating point, and hence with performance. These trends of performance and power usage are illustrated in Figure 18.

A limit on the amount of heat that can be removed from a system (thick horizontal line in the figure) will impose a limit on the achievable performance (dotted line in the figure). However, even below this limit, it may still be undesirable to increase performance, when the marginal benefit of higher operating points is outweighed by the marginal cost of increased power dissipation. At operating points below a certain “performance-power/thermal balance” point, θ_{balance} , it is always beneficial, in the context of a combined view of performance and cooling requirements, to move to a higher operating point if the performance is required by an application. Above θ_{balance} however, heat generation grows at a faster rate than performance. As illustrated in Figure 18, EXABOUNDS enables these tradeoffs and

$$\$i = K_i^{\$NRE} + K_i^{\$pkg} + K_i^{\$pwr} \cdot K_i^{\$cool} \cdot P_{i-1} \cdot \frac{L_{sys}}{C_7} + n_i \cdot \$i_{-1}, \quad \forall i \geq 1, \quad (25)$$

and

$$\$0 = K_0^{\$NRE} + K_i^{\$pkg} + K_0^{\$Si} \cdot f_0 \cdot n_0^{xpc} + K^{\$pwr} \cdot P_0^{cor} \cdot \frac{L_{sys}}{C_7}. \quad (26)$$

Figure 17. Hierarchical cost model.

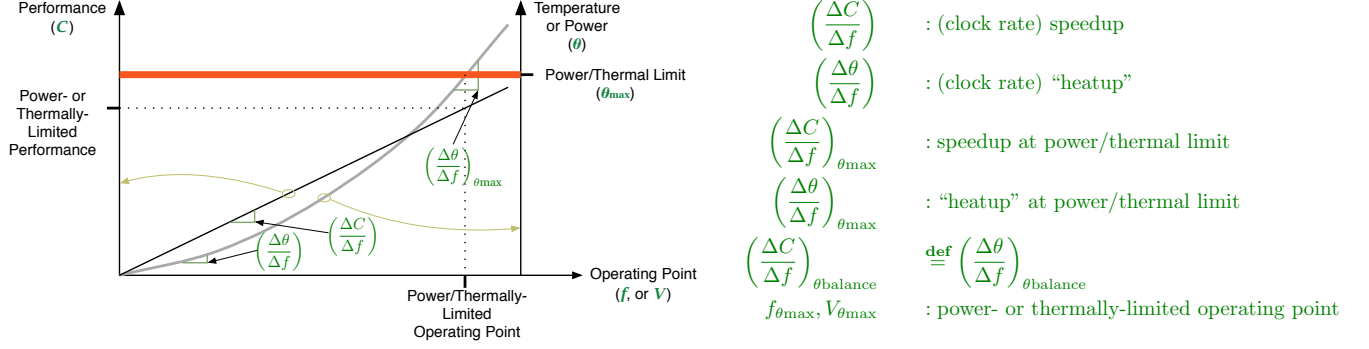


Figure 18. Tradeoffs exist between gains in performance versus both the absolute thermal limits, and the marginal thermal cost associated with a given increase in performance.

optimal operating points to be captured analytically, and to be related to application performance, C (and thence to reliability and cost as well).

4.8.2 Area constraints

Die area models can be constructed by associating a transistor count with each of the parameters in levels 0 (core) and 1 (die) of the EXABOUNDS model. These transistor counts can then be used in conjunction with the transistor areal density trend model shown in Figure 14 to estimate required die area at a given target implementation technology generation. The EXABOUNDS framework calls upon CACTI to obtain area, access time, and power estimates for on-die memories (caches).

One avenue for dealing with area constraints is the utilization of stacked dies with three-dimensional integration [55], with connections between dies in a stack implemented using *through-silicon vias* (TSVs). The effective die area available can be approximated by

$$\text{Effective die area} = A_1^{\text{die}} \cdot n_2^{\text{stack}} \cdot \left(1 - F_2^{\text{PwrTSV}} - F_2^{\text{IoTSV}}\right), \quad (27)$$

where A_1^{die} is the area per die in the stack, n_2^{stack} is the number of layers stacked, F_2^{PwrTSV} is the fraction of area devoted to power TSVs, and F_2^{IoTSV} is the area fraction dedicated to signaling I/O TSVs.

5 EXABOUNDS—A Framework for Better-than-back-of-the-envelope Analysis of Large-scale systems

Figure 19 shows the overall architecture of the framework. This document focuses only on the final stage; the characterization framework is described elsewhere [15, 17].

5.1 Application of The Framework: Cost-effective processor choices for scale-out systems

As an illustrative practical application, when completed, the EXABOUNDS framework should enable addressing questions such as the two sets of questions listed below, for the station-level beamforming application described previously in Section 3.

The first set of questions concerns statements that can be made about desirable hardware properties, given the algorithm characteristics and technology trends:

- ❶ **Power budget available for processors**, given power delivery constraints.
- ❷ **Memory bandwidth requirements per core**, given application's raw memory bandwidth and temporal/spatial reuse characteristics.
- ❸ **Core count limitation**, based on expected package pin count and memory bandwidth requirements.
- ❹ **The ratio of functional unit area to cache area.**

The second set of questions involves a comparison between a range of candidate architectures (Figure 20): a GPU-based platform, three platforms based on low-power embedded processor architectures, and two platforms based on high-performance server processors. For each platform, we investigate (relative to a common baseline), for the station-level beamforming algorithm:

- ❶ *Mean time to failure (MTTF)* under assumptions of socket lifetime being the typical deployment duration of the processor type in question, as well as with a fixed per-processor failure rate across all processor types.
- ❷ *Performance*, under a whole-system power constraint of 10 kW.
- ❸ *Scalability of performance* with technology genera-

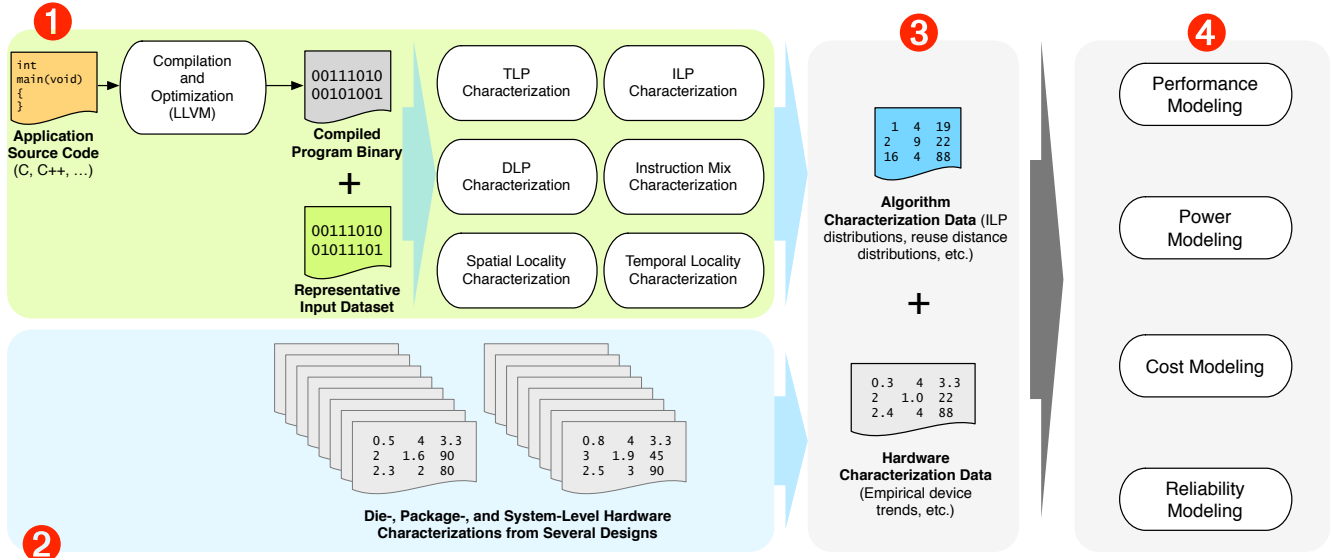


Figure 19. EXABOUNDS and application characterization tool flow.

tions. Some architectures may not be able to take advantage of improvements in transistor density due to, e.g., being memory bandwidth limited.

- 4 *Power dissipation* under the performance requirements of the station-level beamforming.
- 5 *Total cost of acquisition and operation* over a fifteen year period.

6 Summary of Contributions

This article provided an overview of the ideas and implementation behind the EXABOUNDS framework, being developed as part of the five year Dome project, a collaboration between IBM, ASTRON, and the South-African square kilometer array organization (SKA-SA).

A Data Sources

To obtain the data presented in this article, approximately 250 publications were analyzed, primarily from the *International Solid-State Circuits Conference (ISSCC)*, pertaining mostly to programmable processor designs (ranging from microcontrollers to microprocessors, digital signal processors (DSPs), and other digital ASICs). The selected publications were restricted to those which provided information on most or all of: die size, number of transistors (split into logic versus latches/memory where known), measured peak power dissipation, nominal operating voltage, operating frequency, gate length, total package pins, and number of signal versus supply pins. From the initial set of design publications analyzed, 130 (some of which described multiple designs) were retained which provided most of the required data, or for which there was an associated *IEEE Journal of Solid-State Circuits (JSSC)* publication with the missing detail.

The publications corresponding to the data points in the analysis are listed in Table 6; the format of entries is *Year:paper-ID*. In the PDF version of this article, each entry encodes a web search URL for the corresponding paper's title.

Table 6. Publications corresponding to the ISSCC device technology data in this article.

1980:THAM9.1	1980:WAM3.4	1981:THAM9.3	1982:THAM10.2
1982:WPM6.1	1983:FAM18.4	1983:THAM10.2	1983:THAM10.4
1984:WPM8.4	1985:THPM14.4	1985:WAM1.3	1985:WPM8.4
1985:WPM8.5	1986:THAM10.2	1986:THAM12.1	1986:WPM8.2
1986:WPM9.5	1987:FAM20.2	1987:THPM16.2	1987:THPM16.3
1987:THPM16.4	1987:WAM2.4	1988:THAM11.6	1988:THAM12.3
1988:THPM12.6	1988:THPM12.7	1989:THPM12.8	1989:WAM3.1
1989:WAM3.2	1989:WAM3.5	1989:WPM7.2	1990:TAM7.6
1990:TPM9.4	1990:WPM3.3	1990:WPM3.7	1991:FPM15.15
1991:FP15.3	1991:TAM5.3	1992:TA6.5	1992:TA6.2
1992:WP4.6	1993:WP2.2	1993:WP2.6	1993:TA5.2
1993:TA5.5	1994:TP12.5	1994:FA15.2	1994:TP12.7
1994:TP12.6	1994:TP12.2	1995:TA6.6	1995:TP10.3
1995:TP10.6	1996:FA10.4	1996:FA8.6	1996:FP13.5
1996:SP22.1	1997:FA10.2	1997:FA10.3	1997:FA10.4
1997:SP25.1	1997:FA10.5	1997:FP16.3	1997:SP25.2
1998:FA7.6	1998:FP15.1	1998:FP15.3	1998:SA18.3
1998:SA18.6	1999:MP5.1	1999:MP5.6	1999:MP5.7
1999:TP15.3	1999:TP15.4	2000:14.6	2000:MP4.2
2000:MP5.2	2000:MP5.4	2000:MP5.5	2000:MP5.7
2000:TP14.5	2000:WP25.2	2000:WP25.3	2000:WP25.6
2001:15.4	2001:15.1	2001:15.5	2001:15.6
2002:20.4	2002:22.5	2003:14.1	2003:14.4
2003:14.5	2003:14.6	2004:3.1	2004:3.2
2004:3.4	2005:10.1	2005:10.3	2005:10.4
2005:10.7	2006:22.4	2006:22.5	2006:23.6
2006:29.5	2006:5.1	2006:5.3	2006:5.5
2007:15.1	2007:5.3	2007:5.6	2008:13.1
2008:4.2	2008:4.4	2008:4.5	2008:4.6
2009:3.1	2010:5.1	2010:5.5	2010:5.7

B References

- [1] Modeling Application Performance by Convolving Machine Signatures with Application Profiles. (Cited on page 5.)
- [2] G. Amdahl. Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS Conference Proceedings vol. 30*, pages 483–485, Reston, VA, April 18–20 1967. AFIPS Press. (Cited on pages 2 and 13.)

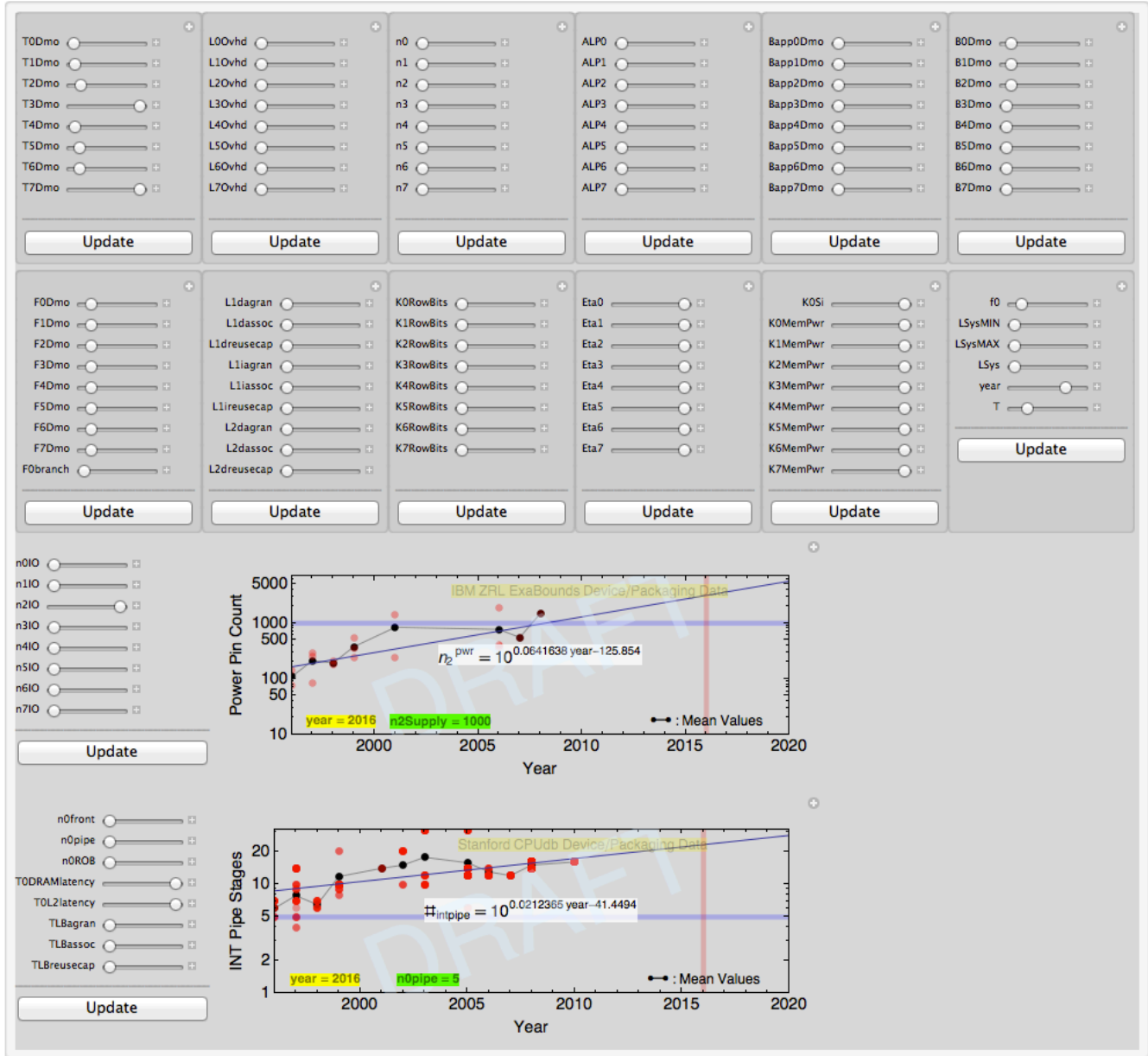


Figure 21. Screenshot of a portion of the EXABOUNDS implementation interface in Mathematica. The implementation in Mathematica encapsulates all the relations presented in this article, as well as the enabling the visualization of parameter values in the context of historical hardware and processor performance data.

[3] K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, D. Wessel, and K. Yelick. A view of the parallel computing landscape. *Commun. ACM*, 52(10):56–67, Oct. 2009. (Cited on page 15.)






[4] T. M. Austin and G. S. Sohi. Dynamic dependency analysis of ordinary programs. *SIGARCH Comput. Archit. News*, 20(2):342–351, 1992. (Cited on page 4.)

[5] O. Azizi, A. Mahesri, B. C. Lee, S. J. Patel, and M. Horowitz. Energy-performance tradeoffs in processor architecture and circuit design: a marginal cost






analysis. In *Proceedings of the 37th annual international symposium on Computer architecture, ISCA '10*, pages 26–36, New York, NY, USA, 2010. ACM. (Cited on pages 3 and 4.)

[6] G. Balamurugan, J. Kennedy, G. Banerjee, J. Jaussi, M. Mansuri, F. O’Mahony, B. Casper, and R. Mooney. A scalable 5–15 gbps, 14–75 mw low-power i/o transceiver in 65 nm cmos. *Solid-State Circuits, IEEE Journal of*, 43(4):1010–1019, april 2008. (Cited on page 16.)

[7] R. Beusoleil, P. Kuekes, G. Snider, S.-Y. Wang, and

Level Name	Relevant properties			
Rack unit	1 card per rack unit;	$n_4 = 1$		Hypothetical System Extension
Card	24 sockets per card;	$n_3 = 24$		
Socket	1 die per socket;	$n_2 = 1$		ARM Cortex A8 System
Die	1 core per die;	$n_1 = 1$		
Core	2-wide issue;	$n_0 = 2$		

(a)

Level Name	Relevant properties			
Rack unit	1 card per rack unit;	$n_4 = 1$		Intel Core i7 Server
Card	2 sockets per card;	$n_3 = 2$		
Socket	1 die per socket;	$n_2 = 1$		Intel Core i7 Server
Die	4 cores per die;	$n_1 = 4$		
Core	4-wide issue;	$n_0 = 4$		

(b)

Figure 20. Illustration of a (a) “wimpy”, and (b) “brawny” system. in the context of the hierarchy presented in this article. Actual instances of the “wimpy” and “brawny” systems, corresponding to an ARM implementation and an Intel Nehalem are used in our validation analysis.

- R. Williams. Nanoelectronic and nanophotonic interconnect. *Proceedings of the IEEE*, 96(2):230–247, feb. 2008. (Cited on pages 15 and 16.)
- [8] P. Bohrer, J. Peterson, M. Elnozahy, R. Rajamony, A. Gheith, R. Rockhold, C. Lefurgy, H. Shafi, T. Nakra, R. Simpson, E. Speight, K. Sudeep, E. Van Hensbergen, and L. Zhang. Mambo: a full system simulator for the powerpc architecture. *SIGMETRICS Perform. Eval. Rev.*, 31:8–12, March 2004. (Cited on page 2.)
- [9] P. J. Bohrer, J. L. Peterson, E. N. Elnozahy, R. Rajamony, A. Gheith, R. L. Rockhold, C. Lefurgy, H. Shafi, T. Nakra, R. O. Simpson, E. Speight, K. Sudeep, E. V. Hensbergen, and L. Zhang. Mambo: a full system simulator for the powerPC architecture. *SIGMETRICS Performance Evaluation Review*, 31(4):8–12, 2004. (Cited on page 4.)
- [10] S. Borkar and A. A. Chien. The future of microprocessors. *Commun. ACM*, 54:67–77, May 2011. (Cited on pages 4 and 15.)
- [11] E. Boyd, W. Azeem, H. Lee, T. Shih, S. Hung, and E. Davidson. A hierarchical approach to modeling and improving the performance of scientific applications on the ksr1. In *Parallel Processing, 1994. ICPP 1994. International Conference on*, volume 3, pages 188–192. IEEE, 1994. (Cited on page 3.)
- [12] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th annual international symposium on Computer architecture*, ISCA '00, pages 83–94, New York, NY, USA, 2000. ACM. (Cited on pages 2 and 4.)
- [13] D. Burger, T. M. Austin, and S. Bennett. Evaluating future microprocessors: The simplescalar tool set. Technical Report CS-TR-1996-1308, University of Wisconsin, Madison, July 1996. (Cited on page 4.)
- [14] D. Burger, J. R. Goodman, and A. Kägi. Memory bandwidth limitations of future microprocessors. In *Proceedings of the 23rd annual international symposium on Computer architecture*, ISCA '96, pages 78–89, New York, NY, USA, 1996. ACM. (Cited on page 11.)
- [15] V. C. Cabezas and P. Stanley-Marbell. Quantitative analysis of parallelism and data movement properties across the berkeley computational motifs. In *Proceedings of the 8th ACM International Conference on Computing Frontiers*, CF '11, pages 17:1–17:2, New York, NY, USA, 2011. ACM. (Cited on page 18.)
- [16] A. E. Caldwell, Y. Cao, A. B. Kahng, F. Koushanfar, H. Lu, I. L. Markov, M. Oliver, D. Stroobandt, and D. Sylvester. Gtx: the marco gsrc technology extrapolation system. In *DAC: Proceedings of the 37th conference on Design automation*, pages 693–698, New York, NY, USA, 2000. ACM. (Cited on page 4.)
- [17] V. Caparrós Cabezas and P. Stanley-Marbell. Parallelism and data movement characterization of contemporary application classes. In *Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures*, SPAA '11, pages 95–104, New York, NY, USA, 2011. ACM. (Cited on pages 4, 7 and 18.)
- [18] B. Casper, J. Jaussi, F. O'Mahony, M. Mansuri, K. Canagasaby, J. Kennedy, E. Yeung, and R. Mooney. A 20gb/s forwarded clock transceiver in 90nm cmos b. In *Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers. IEEE International*, pages 263–272, feb. 2006. (Cited on page 16.)
- [19] A. Chandrakasan, S. Sheng, and R. Brodersen. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, 1992. (Cited on page 5.)
- [20] L. Chang, D. Frank, R. Montoye, S. Koester, B. Ji, P. Co-teus, R. Dennard, and W. Haensch. Practical strategies for power-efficient computing technologies. *Proceedings of the IEEE*, 98(2):215–236, feb. 2010. (Cited on page 15.)
- [21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 2009. (Cited on pages 2, 5 and 13.)
- [22] M. E. Crovella and T. J. LeBlanc. Parallel performance prediction using lost cycles analysis. In *Proceedings of the 1994 ACM/IEEE conference on Supercomputing*, Supercomputing '94, pages 600–609, New York, NY, USA, 1994. ACM. (Cited on page 3.)
- [23] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a realistic model of parallel computation. In M. Chen, editor, *Proceedings of the 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 1–12, San Diego, CA, May 1993. ACM Press. (Cited on page 3.)
- [24] K. Czechowski, C. Battaglini, C. McClanahan, A. Chan-

- dramowlshwaran, and R. Vuduc. Balance principles for algorithm-architecture co-design. In *Proceedings of the 3rd USENIX conference on Hot topic in parallelism, HotPar'11*, pages 9–9, Berkeley, CA, USA, 2011. USENIX Association. (Cited on page 3.)
- [25] A. Danowitz, K. Kelley, J. Mao, J. P. Stevenson, and M. Horowitz. Cpu db: Recording microprocessor history. *Queue*, 10(4):10:10–10:27, Apr. 2012. (Cited on page 4.)
- [26] M. de Vos, A. Gunst, and R. Nijboer. The lofar telescope: System architecture and signal processing. *Proceedings of the IEEE*, 97(8):1431–1437, 2009. (Cited on page 5.)
- [27] E. P. DeBenedictis. The Sandia Petaflops Planner, 2003. SAND REPORT, SAND2003-3609. (Cited on page 3.)
- [28] P. Dewdney, P. Hall, R. Schilizzi, and T. Lazio. The square kilometre array. *Proceedings of the IEEE*, 97(8):1482–1496, 2009. (Cited on page 5.)
- [29] D. Dunning, R. Mooney, P. Stolt, and B. Casper. *Intel Technology Journal*, 13(4), 2009. (Cited on page 15.)
- [30] J. Eble III. *A generic system simulator with novel on-chip cache and throughput models for gigascale integration*. PhD thesis, Georgia Institute of Technology, 1998. (Cited on page 4.)
- [31] P. G. Emma. Understanding some simple processor-performance limits. *IBM J. Res. Dev.*, 41:215–232, May 1997. (Cited on page 8.)
- [32] S. Eyerman, L. Eeckhout, T. Karkhanis, and J. E. Smith. A performance counter architecture for computing accurate cpi components. In *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems, ASPLOS-XII*, pages 175–184, New York, NY, USA, 2006. ACM. (Cited on page 3.)
- [33] S. Eyerman, L. Eeckhout, T. Karkhanis, and J. E. Smith. A mechanistic performance model for superscalar out-of-order processors. *ACM Trans. Comput. Syst.*, 27:3:1–3:37, May 2009. (Cited on pages 3, 8 and 10.)
- [34] I. T. R. for Semiconductors. International technology roadmap for semiconductors, 2008 update. http://www.itrs.net/Links/2008ITRS/Update/2008_Update.pdf (accessed May 2009), 2008. (Cited on page 15.)
- [35] D. J. Frank, W. Haensch, G. Shahidi, and O. H. Dokumaci. Optimizing cmos technology for maximum performance. *IBM Journal of Research and Development*, 50(4.5):419–431, July 2006. (Cited on page 4.)
- [36] A. Y. Grama, A. Gupta, and V. Kumar. Isoefficiency: Measuring the scalability of parallel algorithms and architectures. *IEEE Parallel Distrib. Technol.*, 1(3):12–21, 1993. (Cited on pages 2, 3 and 12.)
- [37] J. L. Gustafson. Reevaluating amdahl's law. *Commun. ACM*, 31(5):532–533, 1988. (Cited on page 13.)
- [38] W. Haensch, E. J. Nowak, R. H. Dennard, P. M. Solomon, A. Bryant, O. H. Dokumaci, A. Kumar, X. Wang, J. B. Johnson, and M. V. Fischetti. Silicon cmos devices beyond scaling. *IBM J. Res. Dev.*, 50:339–361, July 2006. (Cited on page 4.)
- [39] A. Hartstein, V. Srinivasan, T. R. Puzak, and P. G. Emma. Cache miss behavior: is it $\sqrt{2}$? In *Proceedings of the 3rd conference on Computing frontiers, CF '06*, pages 313–320, New York, NY, USA, 2006. ACM. (Cited on page 12.)
- [40] Y. He, C. E. Leiserson, and W. M. Leiserson. The cilkview scalability analyzer. In *SPAA '10: Proceedings of the 22nd ACM symposium on Parallelism in algorithms and architectures*, pages 145–156, New York, NY, USA, 2010. ACM. (Cited on page 13.)
- [41] P. Heidelberger and K. S. Trivedi. Queueing network models for parallel processing with asynchronous tasks. *IEEE Trans. Comput.*, 31:1099–1109, November 1982. (Cited on page 1.)
- [42] M. D. Hill and M. R. Marty. Amdahl's law in the multicore era. *Computer*, 41:33–38, July 2008. (Cited on page 4.)
- [43] T. Hoefler, W. Gropp, W. Kramer, and M. Snir. Performance modeling for systematic performance tuning. In *State of the Practice Reports, SC '11*, pages 6:1–6:12, New York, NY, USA, 2011. ACM. (Cited on page 3.)
- [44] M. Horowitz, T. Indermaur, and R. Gonzalez. Low-power digital design. In *Low Power Electronics, 1994. Digest of Technical Papers., IEEE Symposium*, pages 8–11, Oct 1994. (Cited on page 14.)
- [45] M. Horowitz, C.-K. K. Yang, and S. Sidiropoulos. High-speed electrical signaling: overview and limitations. *Micro, IEEE*, 18(1):12–24, Jan/feb 1998. (Cited on page 16.)
- [46] Intel Corporation. Xeon 5500. (Cited on page 15.)
- [47] E. İpek, S. A. McKee, R. Caruana, B. R. de Supinski, and M. Schulz. Efficiently exploring architectural design spaces via predictive modeling. In *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems, ASPLOS-XII*, pages 195–206, New York, NY, USA, 2006. ACM. (Cited on page 4.)
- [48] V. Janapa Reddi, B. C. Lee, T. Chilimbi, and K. Vaid. Web search using mobile cores: quantifying and mitigating the price of efficiency. In *ISCA '10: Proceedings of the 37th annual international symposium on Computer architecture*, pages 314–325, New York, NY, USA, 2010. ACM. (Cited on page 17.)
- [49] N. Jouppi. The nonuniform distribution of instruction-level and machine parallelism and its effect on performance. *Computers, IEEE Transactions on*, 38(12):1645–1658, Dec 1989. (Cited on pages 4 and 5.)
- [50] H. Kaeslin. *Digital integrated circuit design: from VLSI architectures to CMOS fabrication*. Cambridge University Press, 2008. (Cited on page 14.)
- [51] T. S. Karkhanis and J. E. Smith. A first-order superscalar processor model. In *Proceedings of the 31st annual international symposium on Computer architecture, ISCA '04*, pages 338–, Washington, DC, USA, 2004. IEEE Computer Society. (Cited on pages 3, 4, 8 and 10.)
- [52] T. S. Karkhanis and J. E. Smith. Automated design of application specific superscalar processors: an analytical approach. In *Proceedings of the 34th annual international symposium on Computer architecture, ISCA '07*, pages 402–411, New York, NY, USA, 2007. ACM. (Cited on pages 3 and 4.)

- [53] D. J. Kerbyson, H. J. Alme, A. Hoisie, F. Petrini, H. J. Wasserman, and M. Gittings. Predictive performance and scalability modeling of a large-scale application. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*, Supercomputing '01, pages 37–37, New York, NY, USA, 2001. ACM. (Cited on page 3.)
- [54] L. Kleinrock. *Queueing systems. volume 1: Theory*. 1975. (Cited on page 10.)
- [55] J. Knickerbocker, P. Andry, B. Dang, R. Horton, M. Interante, C. Patel, R. Polastre, K. Sakuma, R. Sirdeshmukh, E. Sprogis, et al. Three-dimensional silicon integration. *IBM Journal of Research and Development*, 52(6):553–569, 2008. (Cited on pages 15 and 18.)
- [56] J. Koomey, K. Brill, P. Turner, J. Stanley, and B. Taylor. A Simple Model for Determining True Total Cost of Ownership for Data Centers. 2007. (Cited on page 5.)
- [57] V. A. Korthikanti and G. Agha. Analysis of parallel algorithms for energy conservation in scalable multicore architectures. In *Proceedings of the 2009 International Conference on Parallel Processing, ICPP '09*, pages 212–219, Washington, DC, USA, 2009. IEEE Computer Society. (Cited on page 5.)
- [58] M. S. Lam and R. P. Wilson. Limits of control flow on parallelism. In *Proceedings of the 19th annual international symposium on Computer architecture, ISCA '92*, pages 46–57, New York, NY, USA, 1992. ACM. (Cited on page 4.)
- [59] B. C. Lee and D. M. Brooks. Accurate and efficient regression modeling for microarchitectural performance and power prediction. In *ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 185–194, New York, NY, USA, 2006. ACM. (Cited on pages 2 and 4.)
- [60] 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 *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 42*, pages 469–480, New York, NY, USA, 2009. ACM. (Cited on pages 2 and 4.)
- [61] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hällberg, J. Högberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A full system simulation platform. *Computer*, 35(2):50–58, Feb. 2002. (Cited on page 4.)
- [62] G. Marin and J. Mellor-Crummey. Cross-architecture performance predictions for scientific applications using parameterized models. In *Proceedings of the joint international conference on Measurement and modeling of computer systems, SIGMETRICS '04/Performance '04*, pages 2–13, New York, NY, USA, 2004. ACM. (Cited on page 5.)
- [63] J. Miller, H. Kasture, G. Kurian, C. Gruenwald, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal. Graphite: A distributed parallel simulator for multicores. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1–12, jan. 2010. (Cited on pages 1, 2 and 4.)
- [64] C. Minkenbergh and G. Rodriguez. Trace-driven co-simulation of high-performance computing systems using omnet++. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Simutools '09*, pages 65:1–65:8, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). (Cited on pages 1 and 4.)
- [65] G. Moore. Our revolution, 2002. <http://www.singularity.com/charts/page62.html>. (Cited on page 17.)
- [66] L. W. Nagel and D. O. Pederson. Simulation Program with Integrated Circuit Emphasis. In *Proceedings of the 16th Midwest Symposium Circuit Theory*, Waterloo, Canada, 1973. (Cited on page 4.)
- [67] D. B. Noonburg and J. P. Shen. Theoretical modeling of superscalar processor performance. In *Proceedings of the 27th annual international symposium on Microarchitecture, MICRO 27*, pages 52–62, New York, NY, USA, 1994. ACM. (Cited on pages 3, 4 and 5.)
- [68] M. Oskin, F. T. Chong, and M. Farrens. Hls: combining statistical and symbolic simulation to guide microprocessor designs. In *Proceedings of the 27th annual international symposium on Computer architecture, ISCA '00*, pages 71–82, New York, NY, USA, 2000. ACM. (Cited on page 4.)
- [69] S. Palacharla, N. P. Jouppi, and J. E. Smith. Complexity-effective superscalar processors. In *Proceedings of the 24th annual international symposium on Computer architecture, ISCA '97*, pages 206–218, New York, NY, USA, 1997. ACM. (Cited on page 4.)
- [70] Peter M. Kogge (editor). Exascale computing study: Technology challenges in achieving exascale systems, univ. of notre dame, cse dept. tech. report tr-2008-13, 2008. (Cited on page 16.)
- [71] B. R. Rau and J. A. Fisher. Instruction-level parallel processing: history, overview, and perspective. *J. Supercomput.*, 7(1-2):9–50, 1993. (Cited on page 4.)
- [72] E. M. Riseman and C. C. Foster. The inhibition of potential parallelism by conditional jumps. *IEEE Trans. Comput.*, 21:1405–1411, December 1972. (Cited on page 4.)
- [73] T. Sakurai and A. Newton. Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas. *IEEE Journal of Solid-State Circuits*, 25(2):584–594, 1990. (Cited on page 14.)
- [74] S. M. Sanchez. Better than a petaflop: the power of efficient experimental design. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 73–84. Winter Simulation Conference, 2008. (Cited on page 3.)
- [75] B. Schroeder, E. Pinheiro, and W.-D. Weber. Dram errors in the wild: a large-scale field study. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems, SIGMETRICS '09*, pages 193–204, New York, NY, USA, 2009. ACM. (Cited on page 17.)
- [76] I. Sharapov, R. Kroeger, G. Delamarter, R. Cheveresan, and M. Ramsay. A case study in top-down per-

- formance estimation for a large-scale parallel application. In *Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming*, PPOPP '06, pages 81–89, New York, NY, USA, 2006. ACM. (Cited on page 4.)
- [77] R. K. Sharma, R. Shih, C. Bash, C. Patel, P. Varghese, M. Mekanapurath, S. Velayudhan, and M. Kumar, V. On building next generation data centers: energy flow in the information technology stack. In *Proceedings of the 1st Bangalore Annual Compute Conference, COMPUTE '08*, pages 8:1–8:7, New York, NY, USA, 2008. ACM. (Cited on page 5.)
- [78] P. Stanley-Marbell, V. Caparros Cabezas, and R. Luijten. Pinned to the walls: impact of packaging and application properties on the memory and power walls. In *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design, ISLPED '11*, pages 51–56, Piscataway, NJ, USA, 2011. IEEE Press. (Cited on pages 4 and 15.)
- [79] D. Sylvester and K. Keutzer. System-level performance modeling with BACPAC – berkeley advanced chip performance calculator, Sept. 1999. (Cited on page 4.)
- [80] K. B. Theobald, G. R. Gao, and L. J. Hendren. On the limits of program parallelism and its smoothability. In *Proceedings of the 25th annual international symposium on Microarchitecture, MICRO 25*, pages 10–19, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press. (Cited on page 4.)
- [81] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi. CACTI 5.1. Technical Report HPL-2008-20, Hewlett Packard Laboratories, November 2008. (Cited on pages 2 and 4.)
- [82] Vivek Sarkar (editor). Exascale Software Study: Software Challenges in Extreme Scale Systems, September 2009. (Cited on page 12.)
- [83] D. W. Wall. Limits of instruction-level parallelism. In *Proceedings of the fourth international conference on Architectural support for programming languages and operating systems, ASPLOS-IV*, pages 176–188, New York, NY, USA, 1991. ACM. (Cited on page 4.)
- [84] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik. Orion: a power-performance simulator for interconnection networks. In *MICRO 35: Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*, pages 294–305, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press. (Cited on page 4.)
- [85] S. Williams, A. Waterman, and D. Patterson. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, Apr. 2009. (Cited on pages 2 and 13.)
- [86] D. A. Wood and M. D. Hill. Cost-effective parallel computing. *Computer*, 28:69–72, February 1995. (Cited on page 17.)
- [87] J. Zhai, W. Chen, and W. Zheng. Phantom: predicting performance of parallel applications on large-scale parallel machines using a single node. In *Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPOPP '10*, pages 305–314, New York, NY, USA, 2010. ACM. (Cited on page 4.)
- [88] W. Zhao and Y. Cao. Predictive technology model for nano-CMOS design exploration. *JETC*, 3(1), 2007. (Cited on page 4.)