DOI:10.1145/2209249.2209272

Looking Back and Looking Forward: Power, Performance, and Upheaval

By Hadi Esmaeilzadeh, Ting Cao, Xi Yang, Stephen M. Blackburn, and Kathryn S. McKinley

Abstract

The past 10 years have delivered two significant revolutions. (1) Microprocessor design has been transformed by the limits of chip power, wire latency, and Dennard scaling—leading to multicore processors and heterogeneity. (2) Managed languages and an entirely new software landscape emerged—revolutionizing how software is deployed, is sold, and interacts with hardware. Researchers most often examine these changes in isolation. Architects mostly grapple with microarchitecture design through the narrow software context of native sequential SPEC CPU benchmarks, while language researchers mostly consider microarchitecture in terms of performance alone. This work explores the clash of these two revolutions over the past decade by measuring power, performance, energy, and scaling, and considers what the results may mean for the future. Our diverse findings include the following: (a) native sequential workloads do not approximate managed workloads or even native parallel workloads; (b) diverse application power profiles suggest that future applications and system software will need to participate in power optimization and management; and (c) software and hardware researchers need access to real measurements to optimize for power and energy.

1. INTRODUCTION

Quantitative performance analysis is the foundation for computer system design and innovation. In their classic paper, Emer and Clark noted that "A lack of detailed timing information impairs efforts to improve performance."[5] They pioneered the quantitative approach by characterizing instruction mix and cycles per instruction on time-sharing workloads. They surprised expert reviewers by demonstrating a gap between the theoretical 1 MIPS peak of the VAX-11/780 and the 0.5 MIPS it delivered on real workloads. Industry and academic researchers in software and hardware all use and extend this principled performance analysis methodology. Our research applies this quantitative approach to measured power. This work is timely because the past decade heralded the era of power- and energy-constrained hardware design.[a] Furthermore, demand for energy efficiency has intensified in large-scale systems, in which energy began to dominate costs, and in mobile systems, which are limited by battery life. A lack of detailed energy measurements is impairing efforts to reduce energy consumption on modern workloads.

[a] Energy = power × execution time.

Society has benefited enormously from exponential hardware performance improvements. Moore observed that transistors will be smaller and more numerous in each new generation.[15] For a long time, this simple rule of integrated circuit fabrication came with an exponential and transparent performance dividend. Shrinking a transistor lowers its gate delay, which raises the processor's theoretical clock speed (Dennard scaling[3]). Until recently, shrinking transistors delivered corresponding clock speed increases and more transistors in the same chip area. Architects used the transistor bounty to add memory, prefetching, branch prediction, multiple instruction issue, and deeper pipelines. The result was *exponential* single-threaded performance improvements.

Unfortunately, physical power and wire-delay limits will derail the clock speed bounty of Moore's law in current and future technologies. Power is now a first-order hardware design constraint in all market segments. Power constraints now severely limit clock scaling and prevent using all transistors simultaneously.[6, 8, 16] In addition, the physical limitations of wires prevent single cycle access to a growing number of the transistors on a chip.[9] To effectively use more transistors at smaller technologies, these limits forced manufacturers to turn to chip multiprocessors (CMPs) and recently to heterogeneous parallel systems that seek power efficiency through specialization. Parallel heterogeneous hardware requires parallel software and exposes software developers to ongoing hardware upheaval. Unfortunately, most software today is not parallel, nor is it designed to modularly decompose onto a heterogeneous substrate.

Moore's transistor bounty also drove *orthogonal* and *disruptive* changes in how software is deployed, is sold, and interacts with hardware over this same decade. Demands for correctness, complexity management, programmer productivity, time-to-market, reliability, security, and portability pushed developers away from low-level compiled ahead-of-time (*native*) programming languages. Developers increasingly

Original version: "*Looking Back on the Language and Hardware Revolutions: Measured Power, Performance, and Scaling*," ACM Conference on Architecture Support for Programming Languages and Operating Systems, pp. 319–332, Newport Beach, CA, 2011. Also published as "*What Is Happening to Power, Performance and Software?*," IEEE Micro Top Picks from the Computer Architecture Conferences of 2011, May/June 2012, IEEE Computer Society.

choose high-level *managed* programming languages with a selection of safe pointer disciplines, garbage collection (automatic memory management), extensive standard libraries, and dynamic just-in-time compilation for hardware portability. For example, modern Web services combine managed languages, such as PHP on the server side and JavaScript on the client side. In markets as diverse as financial software and cell phone applications, Java and .NET are the dominant choices. The exponential performance improvements provided by hardware hid many of the costs of high-level languages and helped create a virtuous cycle with ever more capable and high-level software. This ecosystem is resulting in an explosion of developers, software, and devices that continue to change how we live and learn.

Unfortunately, a lack of power measurements is impairing efforts to reduce energy consumption on traditional and modern software.

Figure 1. Eight findings from an analysis of measured chip power, performance, and energy on 61 workloads and eight processors. The ASPLOS paper includes more findings and analysis.

Findings

Power consumption is highly application dependent and is poorly correlated to TDP.

Power per transistor is relatively consistent within microarchitecture family, independent of process technology.

Energy-efficient architecture design is very sensitive to workload. Configurations in the native non-scalable Pareto Frontier substantially differ from all the other workloads.

Comparing one core to two, enabling a core is not consistently energy efficient.

The Java Virtual Machine induces parallelism into the execution of single-threaded Java benchmarks.

Simultaneous multithreading delivers substantial energy savings for recent hardware and for in-order processors.

Two recent die shrinks deliver similar and surprising reductions in energy, even when controlling for clock frequency.

Controlling for technology, hardware parallelism, and clock speed, the out-of-order architectures have similar energy efficiency as the in-order ones.

2. OVERVIEW

Our work quantitatively examines power, performance, and scaling during this period of disruptive software and hardware changes (2003–2011). Voluminous research explores performance analysis and a growing body of work explores power (see Section 6), but our work is the first to systematically measure the power, performance, and energy characteristics of software and hardware across a range of processors, technologies, and workloads.

We execute 61 diverse sequential and parallel benchmarks written in three native languages and one managed language, all widely used: C, C++, Fortran, and Java. We choose Java because it has mature virtual machine technology and substantial open source benchmarks. We choose eight representative Intel IA32 processors from five technology generations (130 nm to 32 nm). Each processor has an isolated processor power supply with stable voltage on the motherboard, to which we attach a Hall effect sensor that measures power supply current, and hence processor power. We calibrate and validate our sensor data. We find that power consumption varies widely among benchmarks. Furthermore, relative performance, power, and energy are not well predicted by core count, clock speed, or reported Thermal Design Power (TDP). TDP is the nominal amount of power the chip is designed to dissipate (i.e., without exceeding the maximum transistor junction temperature).

Using controlled hardware configurations, we explore the energy impact of hardware features and workload. We perform historical and Pareto analyses that identify the most power- and performance-efficient designs in our architecture configuration space. We make all of our data publicly available in the ACM Digital Library as a companion to our original ASPLOS 2011 paper. Our data quantifies a large number of workload and hardware trends with precision and depth, some known and many previously unreported. This paper highlights eight findings, which we list in Figure 1. Two themes emerge from our analysis: *workload* and *architecture*.

Workload. The power, performance, and energy trends of native workloads substantially differ from managed and parallel native workloads. For example, (a) the SPEC CPU2006 native benchmarks draw significantly less power than parallel benchmarks and (b) managed runtimes exploit parallelism even when executing single-threaded applications. The results recommend that systems researchers include managed and native, sequential and parallel workloads when designing and evaluating energy-efficient systems.

Architecture. Hardware features such as clock scaling, gross microarchitecture, simultaneous multithreading, and chip multiprocessors each elicit a huge variety of power, performance, and energy responses. This variety and the difficulty of obtaining power measurements recommend exposing on-chip power meters and, when possible, power meters for individual structures, such as cores and caches. Modern processors include power management techniques that monitor power sensors to minimize power usage and boost performance. However, only in 2011 (after our original paper) did Intel first expose energy counters, in their production Sandy Bridge processors. Just as hardware event counters provide a quantitative grounding for performance innovations, future architectures should include power and/or energy meters to drive innovation in the power-constrained computer systems era.

Measurement is key to understanding and optimization.

3. METHODOLOGY

This section presents an overview of essential elements of our methodology. We refer the reader to the original paper for a more detailed treatment.

3.1. Software

We systematically explore workload selection and show that it is a critical component for analyzing power and performance. Native and managed applications embody different trade-offs between performance, reliability, portability, and deployment. It is impossible to meaningfully separate language from workload and we offer no commentary on the virtue of language choice. We create four *workloads* from 61 benchmarks.

Native non-scalable: C, C++, and Fortran single-threaded compute-intensive benchmarks from SPEC CPU2006.

Native scalable: Multithreaded C and C++ benchmarks from PARSEC.

Java non-scalable: Single and multithreaded benchmarks that do not scale well from SPECjvm, DaCapo 06-10-MR2, DaCapo 9.12, and pjbb2005.

Java scalable: Multithreaded Java benchmarks from DaCapo 9.12 that scale in performance similarly to native scalable on the i7 (45).

We execute the Java benchmarks on the Oracle HotSpot 1.6.0 virtual machine because it is a mature high-performance virtual machine. The virtual machine dynamically optimizes each benchmark on each architecture. We use best practices for virtual machine measurement of steady state performance.[2] We compile the native non-scalable workload with *icc* at `-o3`. We use *gcc* at `-o3` for the native scalable workload because *icc* did not correctly compile all benchmarks. The *icc* compiler generates better performing code than *gcc*. We execute the same native binaries on all machines. All the parallel native benchmarks scale up to eight hardware contexts. The Java scalable workload is the subset of Java benchmarks that scale well.

3.2. Hardware

Table 1 lists our eight Intel IA32 processors which cover four process technologies (130 nm, 65 nm, 45 nm, and 32 nm) and four microarchitectures (NetBurst, Core, Bonnell, and Nehalem). The release price and date give context regarding Intel's market placement. The Atoms and the Core 2Q (65) *Kentsfield* are extreme market points. These processors are only examples of many processors in each family. For example, Intel sells over 60 Nehalems at 45 nm, ranging in price from around $190 to over $3700. We believe that these samples are representative because they were sold at similar price points.

To explore the influence of architectural features, we selectively down-clock the processors, disable cores on these chip multiprocessors (CMP), disable simultaneous multithreading (SMT), and disable Turbo Boost using BIOS configuration.

3.3. Power, performance, and energy measurement

We isolate the direct current (DC) power supply to the processor on the motherboard and measure its current with *Pololu's ACS714* current sensor board. The supply voltage is very stable, varying by less than 1%, which enables us to correctly calculate power. Prior work used a clamp ammeter, which can only measure alternating current (AC), and is therefore limited to measuring the whole system.[10, 12] After publishing the original paper, Intel made chip-level and core-level energy measurements available on Sandy Bridge processors.[4] Our methodology should slightly overstate chip power because it includes losses due to the motherboard's voltage regulator. Validating against the Sandy Bridge energy counter shows that our power measurements consistently measure about 5% more current.

We execute each benchmark multiple times on every architecture, log its power values, and then compute average power consumption. The aggregate 95% confidence intervals of execution time and power range from 0.7% to 4%. The measurement error in time and power for all processors and benchmarks is low. We compute arithmetic means over the four workloads, weighting each workload equally. To avoid biasing performance measurements to any one architecture, we compute a *reference performance* for each benchmark by averaging the execution time on four architectures: Pentium 4 (130), Core 2D (65), Atom (45), and i5 (32). These choices capture four microarchitectures and four technology generations. We also normalize energy to a reference, since energy = power × time. The reference energy is the average benchmark power on the four processors multiplied by their average execution time.

We measure the 45 processor configurations (8 stock and 37 BIOS configurations) and produce power and

Table 1. Specifications for the eight processors used in the experiments.

Processor	µArch	Processor	sSpec	Release date	Price (USD)	CMP SMT	LLC (B)	Clock (GHz)	nm	Trans M	Die (mm²)	VID Range (V)	TDP (W)	FSB (MHz)	B/W (GB/s)	DRAM Model
Pentium 4	NetBurst	Northwood	SL6WF	May '03	–	1C2T	512K	2.4	130	55	131	–	66	800	–	DDR-400
Core 2 Duo E6600	Core	Conroe	SL9S8	Jul '06	316	2C1T	4M	2.4	65	291	143	0.85–1.50	65	1066	–	DDR2-800
Core 2 Quad Q6600	Core	Kentsfield	SL9UM	Jan '07	851	4C1T	8M	2.4	65	582	286	0.85–1.50	105	1066	–	DDR2-800
Core i7 920	Nehalem	Bloomfield	SLBCH	Nov '08	284	4C2T	8M	2.7	45	731	263	0.80–1.38	130	–	25.6	DDR3-1066
Atom 230	Bonnell	Diamondville	SLB6Z	Jun '08	29	1C2T	512K	1.7	45	47	26	0.90–1.16	4	533	–	DDR2-800
Core 2 Duo E7600	Core	Wolfdale	SLGTD	May '09	133	2C1T	3M	3.1	45	228	82	0.85–1.36	65	1066	–	DDR2-800
Atom D510	Bonnell	Pineview	SLBLA	Dec '09	63	2C2T	1M	1.7	45	176	87	0.80–1.17	13	665	–	DDR2-800
Core i5 670	Nehalem	Clarkdale	SLBLT	Jan '10	284	2C2T	4M	3.4	32	382	81	0.65–1.40	73	–	21.0	DDR3-1333

performance data for each benchmark and processor. Figure 2 shows an example of this data, plotting the power versus performance characteristics for one of the 45 processor configurations, the stock i7 (45).

4. PERSPECTIVE

We organize our analysis into eight findings, as summarized in Figure 1. The original paper contains additional analyses and findings. We begin with broad trends. We show that applications exhibit a large range of power and performance characteristics that are not well summarized by a single number. This section conducts a Pareto energy efficiency analysis for all of the 45 nm processor configurations. Even with this modest exploration of architectural features, the results indicate that each workload prefers a different hardware configuration for energy efficiency.

4.1. Power is application dependent

The nominal thermal design power (TDP) for a processor is the amount of power the chip may dissipate without exceeding the maximum transistor junction temperature. Table 1 lists TDP for each processor. Because measuring real processor power is difficult and TDP is readily available, TDP is often substituted for real measured power. Figure 3 shows that this substitution is problematic. It plots measured power on a logarithmic scale for each benchmark on each stock processor as a function of TDP and indicates TDP with an "✗." TDP is strictly higher than actual power. The gap between peak measured power and TDP varies from processor to processor and TDP is up to a factor of four higher than measured power. The variation among benchmarks is highest on the i7 (45) and i5 (32), likely reflecting their advanced power management. For example, on the i7 (45), measured power varies between 23 W for 471.omnetpp and 89 W for fluidanimate! The smallest variation between maximum and minimum is on the Atom (45) at 30%. This trend is not new. All the processors exhibit a range of benchmark-specific power variation. TDP loosely correlates with power consumption, but it *does not* provide a good estimate for (1) maximum power consumption of individual processors, (2) comparing among processors, or (3) approximating benchmark-specific power consumption.

Finding: *Power consumption is highly application dependent and is poorly correlated to TDP.*

Figure 2 plots power versus relative performance for each benchmark on the i7 (45), which has eight hardware contexts and is the most recent of the 45 nm processors. Native (red) and managed (green) are differentiated by color, whereas scalable (triangle) and non-scalable (circle) are differentiated by shape. Unsurprisingly, the scalable benchmarks (triangles) tend to perform the best and consume the most power. More unexpected is the range of power and performance characteristics of the non-scalable benchmarks. Power is not strongly correlated with performance across workload or benchmarks. The points would form a straight line if the correlation were strong. For example, the point on the bottom right of the figure achieves almost the best relative performance and lowest power.

4.2. Historical overview

Figure 4(a) plots the average power and performance for each processor in their stock configuration relative to the reference performance, using a log/log scale. For example, the i7 (45) points are the average of the workloads derived from the points in Figure 2. Both graphs use the same color for all of the experimental processors in the same family. The shapes encode release age: a square is the oldest, the diamond is next, and the triangle is the youngest, smallest technology in the family.

Figure 2. Power/performance distribution on the i7 (45). Each point represents one of the 61 benchmarks. Power consumption is highly variable among the benchmarks, spanning from 23 W to 89 W. The wide spectrum of power responses from different applications points to power saving opportunities in software.

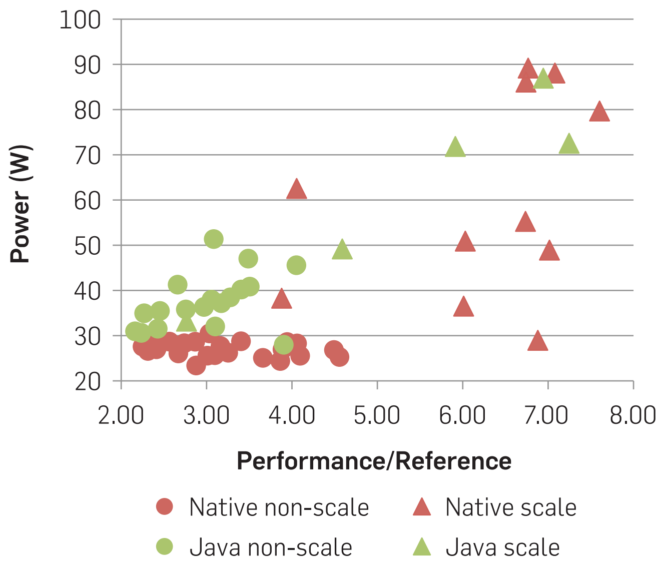


Figure 3. Measured power for each processor running 61 benchmarks. Each point represents measured power for one benchmark. The "✗"s are the reported TDP for each processor. Power is application dependent and does not strongly correlate with TDP.

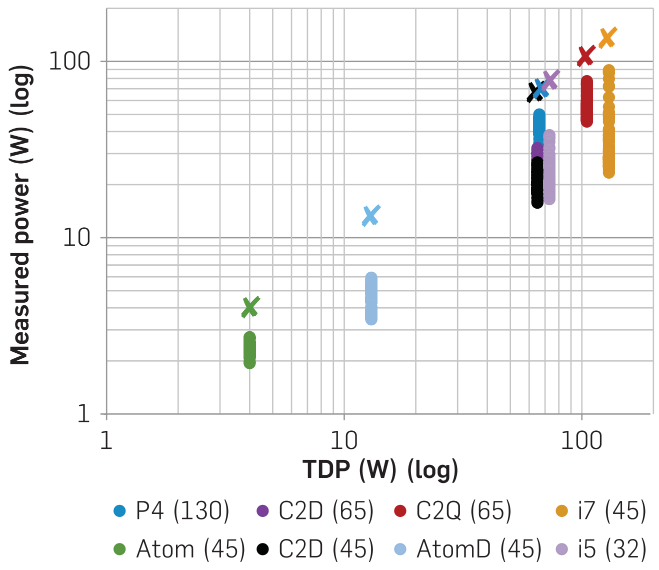
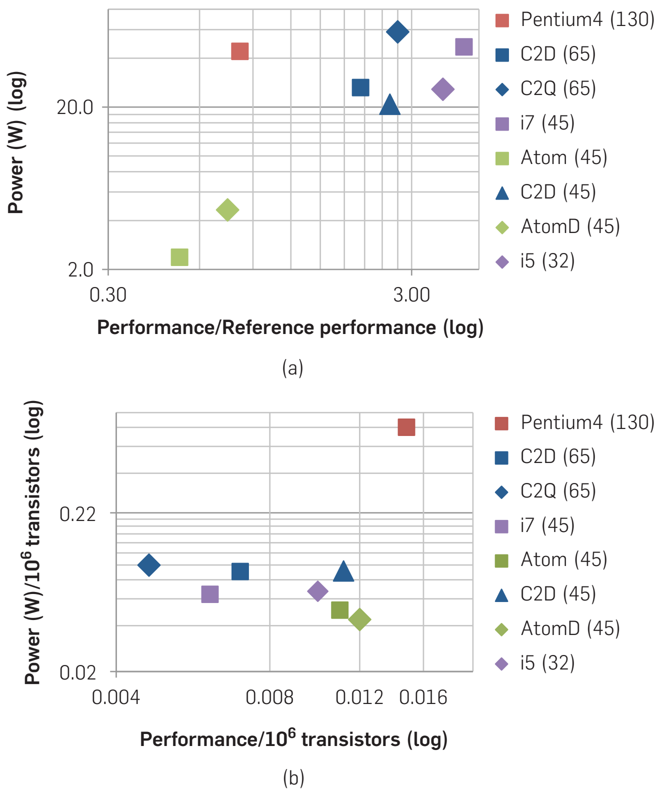


Figure 4. Power/performance trade-off by processor. Each point is an average of the four workloads. (a) Power/performance trade-offs have changed from Pentium 4 (130) to i5 (32). (b) Power and performance per million transistors. Power per million transistors is consistent across different microarchitectures regardless of the technology node. On average, Intel processors burn around 1 W for every 20 million transistors.



While historically mobile devices have been extensively optimized for power, general-purpose processor design has not. Several results stand out that illustrate that power is now a first-order design goal and trumps performance in some cases. (1) The Atom (45) and Atom D (45) are designed as low-power processors for a different market; however, they successfully execute all these benchmarks and are the most power-efficient processors. Compared to the Pentium 4 (130), they degrade performance modestly and reduce power enormously, consuming as little as one-twentieth the power. Device scaling from 130 nm to 45 nm contributes significantly to the power reduction from Pentium to Atom. (2) A comparison between 65 nm and 45 nm generations using the Core 2D (65) and Core 2D (45) shows only a 25% increase in performance, but a 35% drop in power. (3) A comparison of the two most recent 45 nm and 32 nm generations using the i7 (45) and i5 (32) shows that the i5 (32) delivers about 15% less performance, while consuming about 40% less power. This result has three root causes: (a) the i7 (45) has four cores instead of two on the i5 (32); (b) since half the benchmarks are scalable multithreaded benchmarks, the software parallelism benefits more from the additional two cores, increasing the advantage to the i7 (45); and (c) the i7 (45) has significantly better memory performance. Comparing the Core 2D (45) to the i5 (32) where the number of processors are matched, we find that the i5 (32) delivers 50% better performance, while consuming around 25% more power than the Core 2D (45).

Contemporaneous comparisons also reveal the tension between power and performance. For example, the contrast between the Core 2D (45) and i7 (45) shows that the i7 (45) delivers 75% more performance than the Core 2D (45), but this performance is very costly in power, with an increase of nearly 100%. These processors thus span a wide range of energy trade-offs within and across the generations. Overall, these results indicate that optimizing for both power and performance is proving a lot more challenging than optimizing for performance alone.

Figure 4(b) explores the effect of transistors on power and performance by dividing them by the number of transistors in the *package* for each processor. We include all transistors because our power measurements occur at the level of the package, not the die. This measure is rough and will downplay results for the i5 (32) and Atom D (45), each of which have a Graphics Processing Unit (GPU) in their package. Even though the benchmarks do not exercise the GPUs, we cannot discount them because the GPU transistor counts on the Atom D (45) are undocumented. Note the similarity between the Atom (45), AtomD (45), Core 2D (45), and i5 (32), which at the bottom right of the graph are the most efficient processors by the transistor metric. Even though the i5 (32) and Core 2D (45) have five to eight times more transistors than the Atom (45), they all eke out very similar performance and power per transistor. There are likely bigger differences to be found in power efficiency per transistor between chips from different manufacturers.

Finding: *Power per transistor is relatively consistent within microarchitecture family, independent of process technology.*

The leftmost processors in the graph yield the smallest amount of performance per transistor. Among these processors, the Core 2Q (65) and i7 (45) yield the least performance per transistor and use the largest caches among our set. The large 8MB caches are not effective. The Pentium 4 (130) is perhaps most remarkable—it yields the most performance per transistor and consumes the most power per transistor by a considerable margin. In summary, performance per transistor is inconsistent across microarchitectures, but power per transistor is more consistent. Power per transistor correlates well with microarchitecture, regardless of technology generation.

4.3. Pareto analysis at 45 nm

The Pareto optimal frontier defines a set of choices that are most efficient in a trade-off space. Prior research uses the Pareto frontier to explore power versus performance using *models* to derive potential architectural designs on the frontier.[1] We present a Pareto frontier derived from *measured performance and power*. We hold the process technology constant by using the four 45 nm processors: Atom (45), Atom D (45), Core 2D (45), and i7 (45). We expand the number of processor configurations from 4 to 29 by configuring the number of hardware contexts (SMT and CMP), by clock scaling, and disabling/enabling Turbo Boost. The 25 non-stock configurations represent alternative design points. For each configuration, we compute the averages for each workload and their average to produce an energy/performance scatter plot (not shown here). We next pick off the frontier—the points that are not dominated in performance or energy efficiency by any other point—and fit them with a polynomial curve. Figure 5 plots these polynomial curves for each workload and the average. The rightmost curve delivers the best performance for the least energy.

Each row of Figure 6 corresponds to one of the five curves in Figure 5. The check marks identify the Pareto-efficient configurations that define the bounding curve and include 15 of 29 configurations. Somewhat surprising is that none of the Atom D (45) configurations are Pareto efficient. Notice the following: (1) Native non-scalable shares only one choice with any other workload. (2) Java scalable and the average share all the same choices. (3) Only two of eleven choices for Java non-scalable and Java scalable are common to both. (4) Native non-scalable does not include the Atom (45) in its frontier. This last finding contradicts prior simulation work, which concluded that dual-issue in-order cores and dual-issue out-of-order cores are Pareto optimal for native non-scalable.[1] Instead, we find that all of the Pareto-efficient points for native non-scalable in this design space are quad-issue out-of-order i7 (45) configurations.

Figure 5 starkly shows that each workload deviates substantially from the average. Even when the workloads share points, the points fall in different places on the curves because each workload exhibits a different energy/performance trade-off. Compare the scalable and non-scalable benchmarks at 0.40 normalized energy on the *y*-axis. It is impressive how well these architectures effectively exploit software parallelism, pushing the curves to the right and increasing performance from about 3 to 7 while holding energy constant. This measured behavior confirms prior model-based observations about the role of software parallelism in extending the energy/performance curve to the right.[1]

Finding: *Energy-efficient architecture design is very sensitive to workload. Configurations in the native non-scalable Pareto frontier differ substantially from all other workloads.*

In summary, architects should use a variety of workloads, and in particular, should avoid only using native non-scalable workloads.

Figure 5. Energy/performance Pareto frontiers (45 nm). The energy/performance optimal designs are application dependent and significantly deviate from the average case.

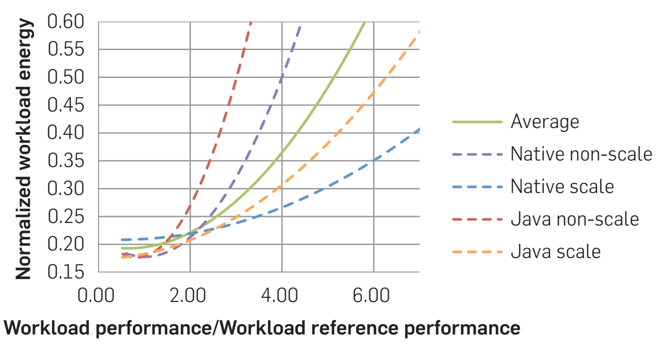


Figure 6. Pareto-efficient processor configurations for each workload. Stock configurations are bold. Each "✔" indicates that the configuration is on the energy/performance Pareto-optimal curve. Native non-scalable has almost no overlap with any other workload.

	Atom(45)1C2T@1.7GHz	Core2D(45)2C1T@1.6GHz	**Core2D(45)2C1T@3.1GHz**	i7(45)1C1T@2.7GHz NoTB	i7(45)1C1T@2.7GHz	i7(45)1C2T@1.6GHz	i7(45)1C2T@2.4GHz	i7(45)2C1T@1.6GHz	i7(45)2C2T@1.6GHz	i7(45)4C1T@2.7GHz NoTB	i7(45)4C1T@2.7GHz	i7(45)4C2T@1.6GHz	i7(45)4C2T@2.1GHz	i7(45)4C2T@2.7GHz NoTB	**i7(45)4C2T@2.7GHz**
Native non-scalable				✔	✔	✔	✔								
Native scalable	✔	✔										✔	✔	✔	✔
Java non-scalable	✔		✔				✔	✔		✔	✔				✔
Java scalable	✔								✔			✔	✔	✔	✔
Average	✔								✔			✔	✔	✔	✔

5. FEATURE ANALYSIS

Our original paper evaluates the energy effect of a range of hardware features: clock frequency, die shrink, memory hierarchy, hardware parallelism, and gross microarchitecture. This analysis resulted in a large number of findings and insights. Reader and reviewer feedback yielded a diversity of opinions as to which findings were most surprising and interesting. This section presents results exploring chip multiprocessing (CMP), simultaneous multithreading (SMT), technology scaling with a die shrink, and gross microarchitecture, to give a flavor of our analysis.

5.1. Chip multiprocessors

Figure 7 shows the average power, performance, and energy effects of chip multiprocessors (CMPs) by comparing one core to two cores for the two most recent processors in our study. We disable Turbo Boost in these analyses because it adjusts power dynamically based on the number of idle cores. We disable Simultaneous Multithreading (SMT) to maximally expose thread-level parallelism to the CMP hardware feature. Figure 7(a) compares relative power, performance, and energy as a weighted average of the workloads. Figure 7(b) shows a break down of the energy as a function of workload. While average energy is reduced by 9% when adding a core to the i5 (32), it is increased by 12% when adding a core to the i7 (45). Figure 7(a) shows that the source of this difference is that the i7 (45) experiences twice the power overhead for enabling a core as the i5 (32), while producing roughly the same performance improvement.

Finding: *Comparing one core to two, enabling a core is not consistently energy efficient.*

Figure 7(b) shows that native non-scalable and Java non-scalable suffer the most energy overhead with the addition of another core on the i7 (45). As expected, performance for native non-scalable is unaffected. However, turning on an additional core for native non-scalable leads to a power increase of 4% and 14%, respectively, for the i5 (32) and

Figure 7. CMP: Comparing two cores to one core. (a) Impact of doubling the number of cores on performance, power, and energy, averaged over all four workloads. (b) Energy impact of doubling the number of cores for each workload. Doubling the cores is not consistently energy efficient among processors or workloads.

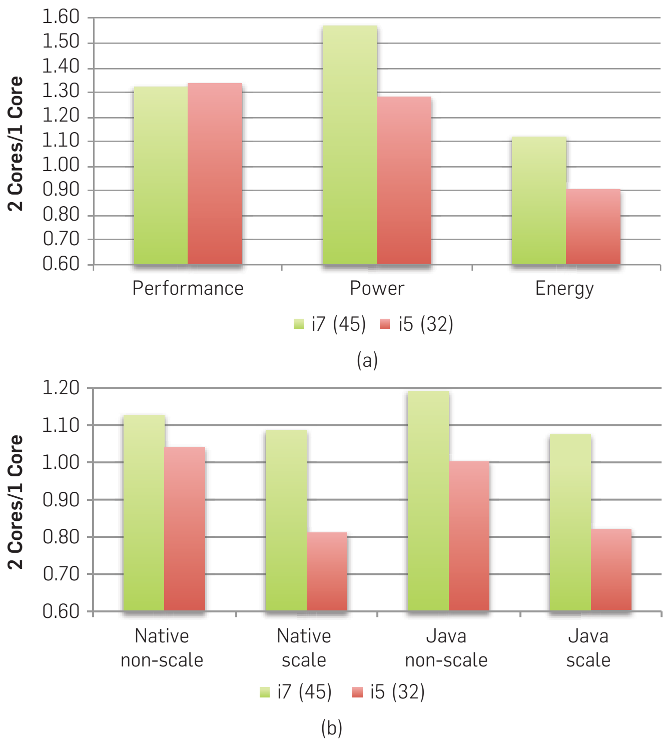


Figure 8. Scalability of single-threaded Java benchmarks. Counterintuitively, some single-threaded Java benchmarks scale well. This is because the underlying JVM exploits parallelism for compilation, profiling, and garbage collection.

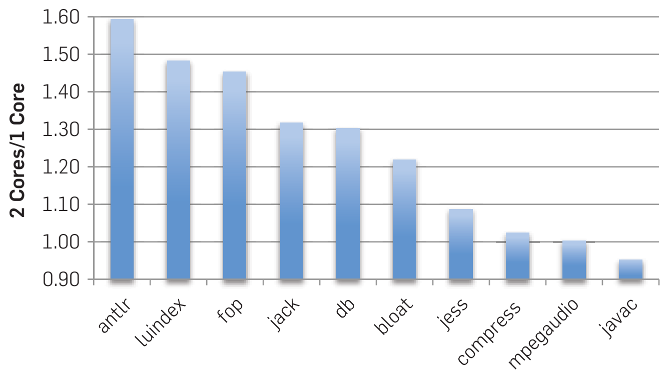
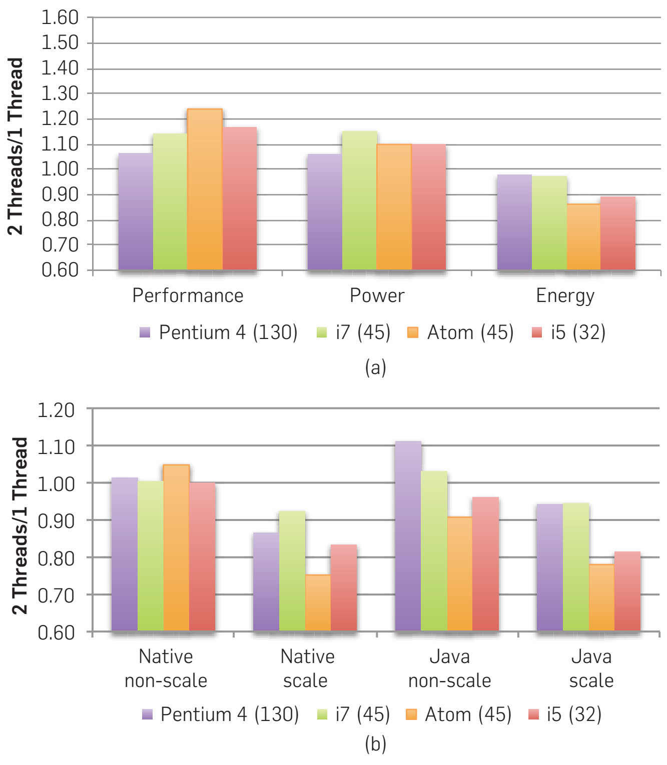


Figure 9. SMT: one core with and without SMT. (a) Impact of enabling two-way SMT on a single-core with respect to performance, power, and energy, averaged over all four workloads. (b) Energy impact of enabling two-way SMT on a single core for each workload. Enabling SMT delivers significant energy savings on the recent i5 (32) and the in-order Atom (45).



i7 (45), translating to energy overheads.

More interesting is that Java non-scalable does not incur energy overhead when enabling another core on the i5 (32). In fact, we were surprised to find that the reason for this is that the *single-threaded* Java non-scalable workload runs faster with two processors! Figure 8 shows the scalability of the single-threaded subset of Java non-scalable on the i7 (45), with SMT disabled, comparing one and two cores. Although these Java benchmarks are single-threaded, the JVMs on which they execute are not.

Finding: *The JVM induces parallelism into the execution of single-threaded Java benchmarks.*

Since virtual machine runtime services for managed languages, such as just-in-time (JIT) compilation, profiling, and garbage collection, are often concurrent and parallel, they provide substantial scope for parallelization, even within ostensibly sequential applications. We instrumented the HotSpot JVM and found that its JIT compilation and garbage collection are parallel. Detailed performance counter measurements revealed that the garbage collector induced memory system improvements with more cores by reducing the collector's displacement effect on the application thread.

5.2. Simultaneous multithreading

Figure 9 shows the effect of disabling simultaneous multithreading (SMT)[19] on the Pentium 4 (130), Atom (45), i5 (32), and i7 (45). Each processor supports two-way SMT. SMT provides fine-grain parallelism to distinct threads in the processors' issue logic and in modern implementations; threads share all processor components (e.g., execution units and caches). Singhal states that the small amount of logic exclusive to SMT consumes very little power.[18] Nonetheless, this logic is integrated, so SMT contributes a small amount to total power even when disabled. Our results therefore slightly underestimate the power cost of SMT. We use only one core, ensuring that SMT is the sole opportunity for thread-level parallelism. Figure 9(a) shows that the performance advantage of SMT is significant. Notably, on the i5 (32) and Atom (45), SMT improves average performance significantly without much cost in power, leading to net energy savings.

Finding: *SMT delivers substantial energy savings for recent hardware and for in-order processors.*

Given that SMT was and continues to be motivated by the challenge of filling issue slots and hiding latency in wide issue superscalars, it may appear counterintuitive that performance on the dual-issue in-order Atom (45) should benefit so much more from SMT than the quad-issue i7 (45) and i5 (32) benefit. One explanation is that the in-order pipelined Atom (45) is more restricted in its capacity to fill issue slots. Compared to other processors in this study, the Atom (45) has much smaller caches. These features accentuate the need to hide latency, and therefore the value of SMT. The performance improvements on the Pentium 4 (130) due to SMT are half to one-third that of more recent processors, and consequently, there is no net energy advantage. This result is not so surprising given that the Pentium 4 (130) is the first commercial implementation of SMT.

Figure 9(b) shows that, as expected, the native non-scalable workload experiences very little energy overhead due to enabling SMT, whereas Figure 7(b) shows that enabling a core incurs a significant power and thus energy penalty. The scalable workloads unsurprisingly benefit most from SMT.

The excellent energy efficiency of SMT is impressive on recent processors as compared to CMP, particularly given its very low die footprint. Compare Figures 7 and 9. SMT provides less performance improvement than CMP—SMT adds about half as much performance as CMP on average but incurs much less power cost. The results on the modern processors show that SMT in a much more favorable light than in Sasanka et al.'s model-based comparative study of the energy efficiency of SMT and CMP.[17]

5.3. Die shrink

We use processor pairs from the Core (Core 2D (65)/Core 2D (45)) and Nehalem (i7 (45)/i5 (32)) microarchitectures to explore die shrink effects. These hardware comparisons are imperfect because they are not straightforward die shrinks. To limit the differences, we control for hardware parallelism by limiting the i7 (45) to two cores. The tools and processors at our disposal do not let us control the cache size, nor do they let us control for other microarchitecture changes that accompany a die shrink. We compare at stock clock speeds and control for clock speed by running both Cores at 2.4 GHz and both Nehalems at 2.66 GHz. We do not directly control for core voltage, which differs across technology nodes for the same frequency. Although imperfect, these are the first published comparisons of measured energy efficiency across technology nodes.

Finding: *Two recent die shrinks deliver similar and surprising reductions in energy, even when controlling for clock frequency.*

Figure 10. Die shrink: microarchitectures compared across technology nodes. "Core" shows Core 2D (65)/Core 2D (45) while "Nehalem" shows i7 (45)/i5 (32) when two cores are enabled. (a) Each processor uses its native clock speed. (b) Clock speeds are matched in each comparison. (c) Energy impact with matched clocks, as a function of workload. Both die shrinks deliver substantial energy reductions.

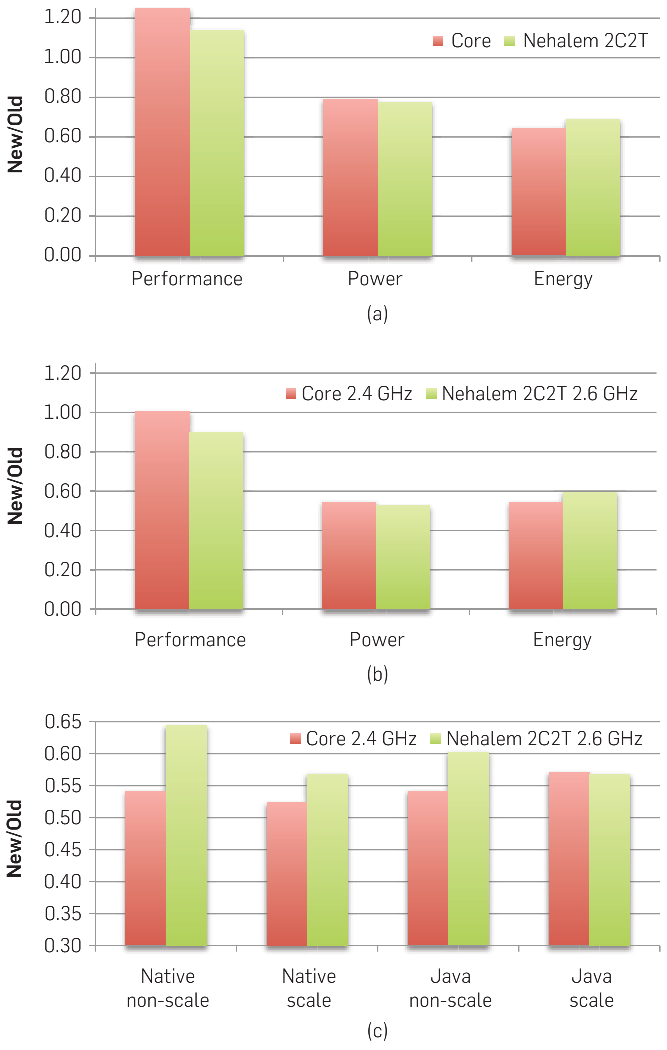


Figure 10(a) shows the power and performance effects of the die shrinks with the stock clock speeds for all the processors. Figure 10(b) shows the same comparison with matched clock speeds, and Figure 10(c) breaks down the workloads for the matched clock speeds. The newer processors are significantly faster at their higher stock clock speeds and significantly more power efficient. Figure 10(b) shows the same experiment, but down-clocking the newer processors to match the frequency of their older peers. Down-clocking the new processors improves their relative power and energy advantage even further. Note that as expected, the die-shrunk processors offer no performance advantage once the clocks are matched; indeed, the i5 (32) performs 10% slower than the i7 (45). However, power consumption is reduced by 47%. This result is consistent with expectations, given the lower voltage and reduced capacitance at the smaller feature size.

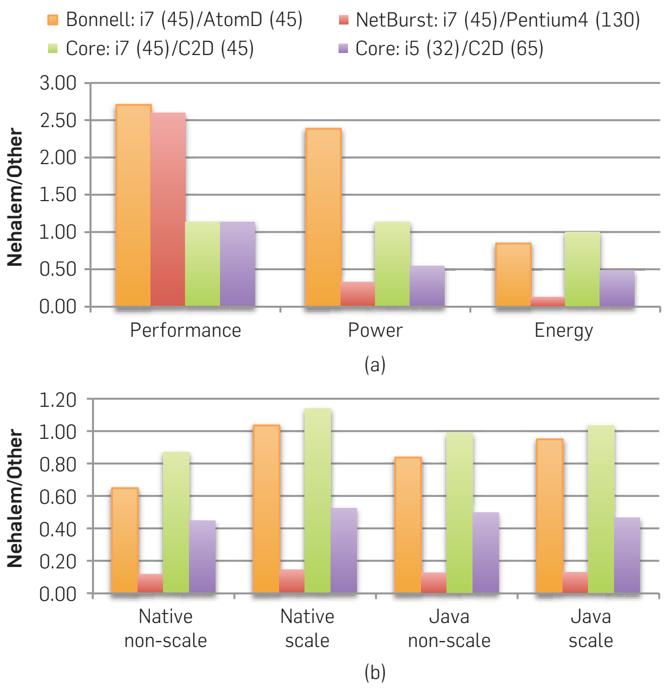
Figures 10(a) and (b) reveal a striking similarity in power and energy savings between the Core (65 nm/45 nm) and Nehalem (45 nm/32 nm) die shrinks. This data suggests that Intel maintained the same rate of energy reduction across the two most recent generations. As a point of comparison, the models used by the International Technology Roadmap for Semiconductors (ITRS) predicted a 9% increase in frequency and a 34% reduction in power from 45 nm to 32 nm.[11] Figure 10(a) is both more and less encouraging. Clock speed increased by 26% in the stock configurations of the i7 (45) to the i5 (32) with an accompanying 14% increase in performance, but power reduced by 23%, less than the 34% predicted. To more deeply understand die shrink efficiency on modern processors, one requires measuring more processors in each technology node.

5.4. Gross microarchitecture change

This section explores the power and performance effect of gross microarchitectural change by comparing microarchitectures while matching features such as processor clock, degree of hardware parallelism, process technology, and cache size.

Figure 11 compares the Nehalem i7 (45) with the NetBurst Pentium 4 (130), Bonnell Atom D (45), and Core 2D (45) microarchitectures, and it compares the Nehalem i5 (32) with the Core 2D (65). Each comparison configures the Nehalems to match the clock speed, number of cores, and hardware threads of the other architecture. Both the i7 (45) and i5 (32) comparisons to the Core show that the move from Core to Nehalem yields a small 14% performance improvement. This finding is not inconsistent with Nehalem's stated primary design goals, that is, delivering scalability and memory performance.

Figure 11. Gross microarchitecture: a comparison of Nehalem with four other microarchitectures. In each comparison, the Nehalem is configured to match the other processor as closely as possible. (a) Impact of microarchitecture change with respect to performance, power, and energy, averaged over all four workloads. (b) Energy impact of microarchitecture for each workload. The most recent microarchitecture, Nehalem, is more energy efficient than the others, including the low-power Bonnell (Atom).



Finding: *Controlling for technology, hardware parallelism, and clock speed, the out-of-order architectures have similar energy efficiency as the in-order ones.*

The comparisons between the i7 (45) and Atom D (45) and Core 2D (45) hold process technology constant at 45 nm. All three processors are remarkably similar in energy consumption. This outcome is all the more interesting because the i7 (45) is disadvantaged since it uses fewer hardware contexts here than in its stock configuration. Furthermore, the i7 (45) integrates more services on-die, such as the memory controller, that are off-die on the other processors, and thus outside the scope of the power meters. The i7 (45) improves upon the Core 2D (45) and Atom D (45) with a more scalable, much higher bandwidth on-chip interconnect, which is not exercised heavily by our workloads. It is impressive that, despite all of these factors, the i7 (45) delivers similar energy efficiency to its two 45 nm peers, particularly when compared to the low-power in-order Atom D (45). It is unsurprising that the i7 (45) performs 2.6× faster than the Pentium 4 (130), while consuming one-third the power, when controlling for clock speed and hardware parallelism (but not for factors such as memory speed). Much of the 50% power improvement is attributable to process technology advances. This speedup of 2.6 over 7 years is however *substantially* less than the historical factor of 8 improvement experienced in every prior 7-year time interval between 1970 through the early 2000s. This difference in improvements marks the beginning of the power-constrained architecture design era.

6. RELATED WORK

The processor design literature is full of performance measurement and analysis. Despite power's growing importance, power measurements are still relatively rare.[7,10,12] Here, we summarize related power measurement and simulation work. Our original paper contains a fuller treatment.

Power measurement. Isci and Martonosi combine a clamp ammeter with performance counters for per unit power estimation of the Intel Pentium 4 on SPEC CPU2000.[10] Fan et al. estimate whole system power for large-scale data centers.[7] They find that even the most power-consuming workloads draw less than 60% of peak possible power consumption. We measure chip power and support their results by showing that TDP does not predict measured chip power. Our work is the first to compare microarchitectures, technology generations, individual benchmarks, and workloads in the context of power and performance.

Power modeling. Power modeling is necessary to thoroughly explore architecture design.[1, 13, 14] Measurement complements simulation by providing validation. For example, some prior simulators used TDP, but our measurements show that it is not accurate. As we look to the future, we believe that programmers will need to tune their applications for power and energy, not only performance. Just as hardware performance counters provide insight into applications, so will power and energy measurements.

Methodology. Although the results show conclusively that managed and native workloads have different responses to architectural variations, perhaps this result should not be surprising. Unfortunately, few architecture or operating system publications with processor measurements or simulated designs use Java or any other managed workloads, even though the evaluation methodologies we use here for real processors and those for simulators are well developed.[2]

7. CONCLUSION

These extensive experiments and analyses yield a wide range of findings. On this basis, we offer the following recommendations in this critical time period of hardware and software upheaval. *Manufacturers* should expose on-chip power meters to the community. *System software and application developers* should understand and optimize power. *Researchers* should use both managed and native workloads to quantitatively examine their innovations. *Researchers* should measure power and performance to understand and optimize power, performance, and energy.

Acknowledgments

We thank Bob Edwards, Daniel Frampton, Katherine Coons, Pradeep Dubey, Jungwoo Ha, Laurence Hellyer, Daniel Jiménez, Bert Maher, Norm Jouppi, David Patterson, and John Hennessy. This work is supported by ARC DP0666059 and NSF CSR-0917191.

References

1. Azizi, O., Mahesri, A., Lee, B.C., Patel, S.J., Horowitz, M. Energy-performance tradeoffs in processor architecture and circuit design: a marginal cost analysis. In *ISCA* (2010).
2. Blackburn, S.M. et al. Wake up and smell the coffee: Evaluation methodologies for the 21st century. *CACM 51*, 8 (2008), 83–89.
3. Bohr, M. A 30 year retrospective on Dennard's MOSFET scaling paper. *IEEE SSCS Newsletter 12*, 1 (2007), 11–13 (http://dx.doi.org/10.1109/N-SSC.2007.4785534).
4. David, H., Gorbatov, E., Hanebutte, U.R., Khanna, R., Le, C. RAPL: memory power estimation and capping. In *ISLPED* (2010).
5. Emer, J.S., Clark, D.W. A characterization of processor performance in the VAX-11/780. In *ISCA* (1984).
6. Esmaeilzadeh, H., Blem, E., St. Amant, R., Sankaralingam, K., Burger, D. Dark silicon and the end of multicore scaling. In *ISCA* (2011).
7. Fan, X., Weber, W.D., Barroso, L.A. Power provisioning for a warehouse-sized computer. In *ISCA* (2007).
8. Hardavellas, N., Ferdman, M., Falsafi, B., Ailamaki, A. Toward dark silicon in servers. *IEEE Micro 31*, 4 (2011), 6–15.
9. Hrishikesh, M.S., Burger, D., Jouppi, N.P., Keckler, S.W., Farkas, K.I., Shivakumar, P. The optimal logic depth per pipeline stage is 6 to 8 FO4 inverter delays. In *International Symposium on Computer Architecture* (2002).
10. Isci, C., Martonosi, M. Runtime power monitoring in high-end processors: Methodology and empirical data. In *MICRO* (2003).
11. ITRS Working Group. International technology roadmap for semiconductors, 2011.
12. Le Sueur, E., Heiser, G. Dynamic voltage and frequency scaling: the laws of diminishing returns. In *HotPower* (2010).
13. Li, S., Ahn, J.H., Strong, R.D., Brockman, J.B., Tullsen, D.M., Jouppi, N.P. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *MICRO* (2009).
14. Li, Y., Lee, B., Brooks, D., Hu, Z., Skadron, K. CMP design space exploration subject to physical contraints. In *HPCA* (2006).
15. Moore, G.E. Cramming more components onto integrated circuits. *Electronics 38*, 8 (19 Apr 1965), 114–117.
16. Mudge, T. Power: a first-class architectural design constraint. *Computer 34*, 4 (Apr. 2001), 52–58.
17. Sasanka, R., Adve, S.V., Chen, Y.K., Debes, E. The energy efficiency of CMP vs. SMT for multimedia workloads. In *ICS* (2004).
18. Singhal, R. Inside Intel next generation Nehalem microarchitecture. Intel Developer Forum (IDF) presentation (August 2008), 2011.
19. Tullsen, D.M., Eggers, S.J., Levy, H.M. Simultaneous multithreading: maximizing on-chip parallelism. In *ISCA* (1995).

Hadi Esmaeilzadeh (hadianeh@cs.washington.edu), Department of Computer Science and Engineering, University of Washington.

Cao Ting, Stephen M Blackburn, Xi Yang ([Cao.Ting, Steve.Blackburn, Xi.Yang]@anu.edu.au), Research School of Computer Science, Australian National University.

Kathryn S McKinley (mckinley@microsoft.com), Microsoft Research, Redmond, WA; The University of Texas at Austin, Department of Computer Science, Austin, TX.