

# acmqueue CPU DB: Recording Microprocessor History

**With this open database, you can mine microprocessor trends over the past 40 years.**

**Andrew Danowitz, Kyle Kelley, James Mao, John P. Stevenson, Mark Horowitz, Stanford University**

In November 1971, Intel introduced the world's first single-chip microprocessor, the Intel 4004. It had 2,300 transistors, ran at a clock speed of up to 740 KHz, and delivered 60,000 instructions per second while dissipating 0.5 watts. The following four decades witnessed exponential growth in compute power, a trend that has enabled applications as diverse as climate modeling, protein folding, and computing real-time ballistic trajectories of angry birds. Today's microprocessor chips employ billions of transistors, include multiple processor cores on a single silicon die, run at clock speeds measured in gigahertz, and deliver more than 4 million times the performance of the original 4004.

Where did these incredible gains come from? This article sheds some light on this question by introducing CPU DB ([cpudb.stanford.edu](http://cpudb.stanford.edu)), an open and extensible database collected by Stanford's VLSI (very large-scale integration) Research Group over several generations of processors (and students). We gathered information on commercial processors from 17 manufacturers and placed it in CPU DB, which now contains data on 790 processors spanning the past 40 years.

In addition, we provide a methodology to separate the effect of technology scaling from improvements on other frontiers (e.g., architecture and software), allowing the comparison of machines built in different technologies. To demonstrate the utility of this data and analysis, we use it to decompose processor improvements into contributions from the physical scaling of devices, and from improvements in microarchitecture, compiler, and software technologies.

## AN OPEN REPOSITORY OF PROCESSOR SPECS

While information about current processors is easy to find, it is rarely arranged in a manner that is useful to the research community. For example, the data sheet may contain the processor's power, voltage, frequency, and cache size, but not the pipeline depth or the technology minimum feature size. Even then, these specifications often fail to tell the full story: a laptop processor operates over a range of frequencies and voltages, not just the 2 GHz shown on the box label.

Not surprisingly, specification data gets harder to find the older the processor becomes, especially for those that are no longer made, or worse, whose manufacturers no longer exist. We have been collecting this type of data for three decades and are now releasing it in the form of an open repository of processor specifications. The goal of CPU DB is to aggregate detailed processor specifications into a convenient form and to encourage community participation, both to leverage this information and to keep it accurate and current. CPU DB ([cpudb.stanford.edu](http://cpudb.stanford.edu)) is populated with desktop, laptop, and server processors, for which we use SPEC<sup>13</sup> as our performance-measuring tool. In addition, the database contains limited data on embedded cores, for which we are using the CoreMark benchmark for performance.<sup>5</sup> With time and help from the community, we hope to extend the coverage of embedded processors in the database.

TABLE 1: Categories used to organize per-processor specifications in CPU DB.

| Category                                     |                                |                                    |                          |
|--|--------------------------------|------------------------------------|--------------------------|
| Processor architecture and microarchitecture | Memory system                  | Physical characteristics           | Technology               |
| Summary Parameter                            |                                |                                    |                          |
| Architecture family                          | Last level cache               | Vdd nominal<br>Clock frequency TDP | Process size             |
| Parameters                                   |                                |                                    |                          |
| Manufacturer                                 | L1 data size                   | Vdd high                           | Process name             |
| Family name                                  | L1 instruction size            | Vdd low                            | Process type             |
| Code name                                    | L2 size                        | Nominal frequency                  | Feature size             |
| Model name                                   | L3 size                        | Turbo frequency                    | Effective channel length |
| Date released                                | Memory bandwidth               | Low power frequency                | Number of metal layers   |
| Number of cores                              | FSB pins                       | TDP                                | Metal type               |
| Threads per core                             | Memory pins                    | Die size                           | F04 delay                |
| Word size                                    | Power and ground pins I/O pins | Number of transistors              |                          |

For users to analyze different processor features, CPU DB contains many data entries for each CPU, ranging from physical parameters such as number of metal layers, to overall performance metrics such as SPEC scores. To make viewing relevant data easier, the database includes summary fields, such as nominal clock frequency, that try to represent more detailed scaling data. Table 1 shows the current list of CPU DB parameters. Table 2 summarizes the “microarchitecture” specifications.

All high-performance processors today tell the system what supply voltage they need within a range of allowable values. This makes it difficult to track how power-supply voltage has scaled over time. Instead of relying on the specified worst-case behavior, researchers are free to analyze the power, frequency, and voltage that a processor actually uses while running an application, and then add it to the CPU DB repository. Table 3 is a summary of the measured parameters tracked in CPU DB.

TABLE 2: Microarchitectural parameters contained in CPU DB.

| Manufacturer                 | Microarchitecture                  | Revision                    | ISA                             |
|------------------------------|------------------------------------|-----------------------------|---------------------------------|
| ISA version                  | ISA extensions                     | Floating point pipe stages  | Integer pipe stages             |
| Max uOps issued per cycle    | Integer functional units           | Load store functional units | Floating point functional units |
| Total functional units       | Max instructions decoded per cycle | Reorder buffer              | Instruction window size         |
| Instruction fetch queue size | Branch history table               | Branch target buffer        | Branch predictor accuracy       |
| Integer registers            | Floating point registers           | Total registers             | Floating point coproc.          |
| TLB entries                  | Out of order                       | Integrated mem. controller  | ---                             |

TABLE 3: Measured parameters in CPU DB.\*

|             |   |
|-------------|---|
| Power       | Power for specified load, Idle power, Max operating power       |
| Voltage     | Vdd for specified load, Vdd idle, Vdd at max power              |
| Performance | SPECrate 2006, SPEC 2006, SPEC 2000, SPEC 1995, SPEC 1992, MIPs |

\* Note: Spec benchmarks also include comprehensive fields for performance on individual spec subtests.

While CPU DB includes a large set of processor data fields, certain members of the architecture community will likely want to explore data fields that we did not think to include. To handle such situations, users are encouraged to suggest new data columns. These suggestions will be reviewed and then entered in the database.

A similar system helps keep CPU DB accurate and up to date. Users can submit data for new processors and architectures, and suggest corrections to data entries. We understand that users may not have data for all of the specifications, and we encourage users to submit any subsets of the data fields. New data and corrections will be reviewed before being applied to the database.

With these mechanisms for adding and vetting data, CPU DB will be a powerful tool for architects who wish to incorporate processor data into their studies. Because many database users will probably want to perform analyses on the raw CPU DB data, the full database is downloadable in comma-separated value format.

#### TECHNOLOGY NORMALIZATION METHODOLOGY

CPU DB allows side-by-side access to performance data for relatively simple in-order processors (up to the mid-1990s) and modern out-of-order processors. One could ask if, at the cost of lower performance, the simplicity of the older designs conferred an efficiency advantage. Unfortunately, direct comparisons using the raw data are difficult because, over the years, manufacturing technologies have improved significantly. A fair comparison would be possible if both processors were manufactured using the same process; but since porting all of these older processors to modern technologies is not feasible, we need another approach. To enable such comparisons, we instead estimate how processor performance and power would scale with technology.

Our main performance metric is based on industry-standard SPEC CPU2006 scores.<sup>13</sup> Unfortunately, most older processors did not run SPEC 2006 and instead measured performance in MIPS (million instructions per second) and, later, in terms of SPEC 1989, SPEC 1992, SPEC 1995, and SPEC 2000. In those cases we estimate SPEC 2006 numbers by converting old scores into a *SPEC 2006 equivalent* score using a conversion factor. The conversion values are determined by examining systems that have scores for two versions of SPEC and then taking the geometric mean of the set of ratios between overlapping scores. This method was used to create the summary performance scores in the database. We also provide the raw scores so that users can develop better conversion methods over time.

To estimate the performance of a processor if it were manufactured using a newer process, we calculate the clock frequency in that technology using gate-delay data. While the speed of the cache memory on the processor scales with technology, the delay going to main memory has scaled only slowly with time. As a result, doubling the clock frequency generally does not double the processor's performance. We finesse this issue the same way the microprocessor industry does: by scaling the on-chip cache so the percentage memory stall time remains constant. Using the empirical rule that miss rates are proportional to the square root of the cache size,<sup>9,14</sup> we expand the last-level cache by four times for each doubling of clock frequency. Thus, we assume that the processor performance scales with clock frequency, but we penalize the energy and area of the processor by growing its cache.

For the clock-cycle time estimate, we need to know how the delays of the gates and wires will scale. Fortunately, the delay scaling of different logic gates is similar, so it is sufficient to measure

how the delay of a single gate scales. Our analysis uses the delay of an inverter driving four equivalent inverters (a fanout of four, or FO4) as the gate-speed metric. Inverters are the most common gate type, and their delay is often published in technology papers. For wire delay it is important to remember that a design's area will shrink with scaling, so its wire delay will, in general, reduce slowly or, at worst, stay constant. Its effect on cycle time depends on the internal circuit design. Designers generally pipeline long wires, so they tend not to limit the critical path. Thus, we ignore wire delay and make the slightly optimistic assumption that a processor's frequency in the new technology will be greater by the ratio of FO4s from old to new:

$$f_2 = f_1 \frac{FO4_1}{FO4_2}$$

Using FO4 as a basic metric has an additional advantage: it cleanly covers the performance/energy variation that comes from changing the supply voltage. Two processors, even built in the same technology, might be operated at different supply voltages. The energy difference between the two can be calculated directly from the supply voltage, but the voltage's effect on performance is harder to estimate. Using FO4 data for these designs at two different voltages provides all the information that is needed.

Having accounted for the effect of the scaled memory systems, we find that estimating the power of a processor with scaled technology is fairly straightforward. Processor power has two components: dynamic and leakage. In an optimized design, the leakage power is around 30 percent of the dynamic power, and the leakage power will scale as the dynamic power scales.<sup>16</sup>

Dynamic power is given by the product of the processor's average activity factor,  $\lambda$  (the probability that a node will switch each cycle), the processor frequency, and the energy to switch the transistors:

$$\text{Energy} = C(V_{dd})^2$$

The processor's average activity factor depends on the logic and not the technology, so it is constant with scaling. Since capacitance per unit length is roughly constant with scaling,  $C$  should be proportional to the feature size  $\lambda$ . We have already estimated how the frequency will scale, so the estimated power and performance scaling for technology is:

$$P_2 = P_1 \frac{\lambda_2 V_2^2 FO4_1}{\lambda_1 V_1^2 FO4_2} + P_{\text{cache}}$$

$$Perf_2 = Perf_1 \frac{FO4_1}{FO4_2}$$

For analyzing processor efficiency, it is often better to look at energy per operation rather than power. Energy/op factors out the linear relationship that both performance and power have with

frequency (FO4). Lowering the frequency changes the power but does not change the energy/op. Since energy/op is proportional to the ratio of power over performance, we derive the next next equation by dividing the previous two:

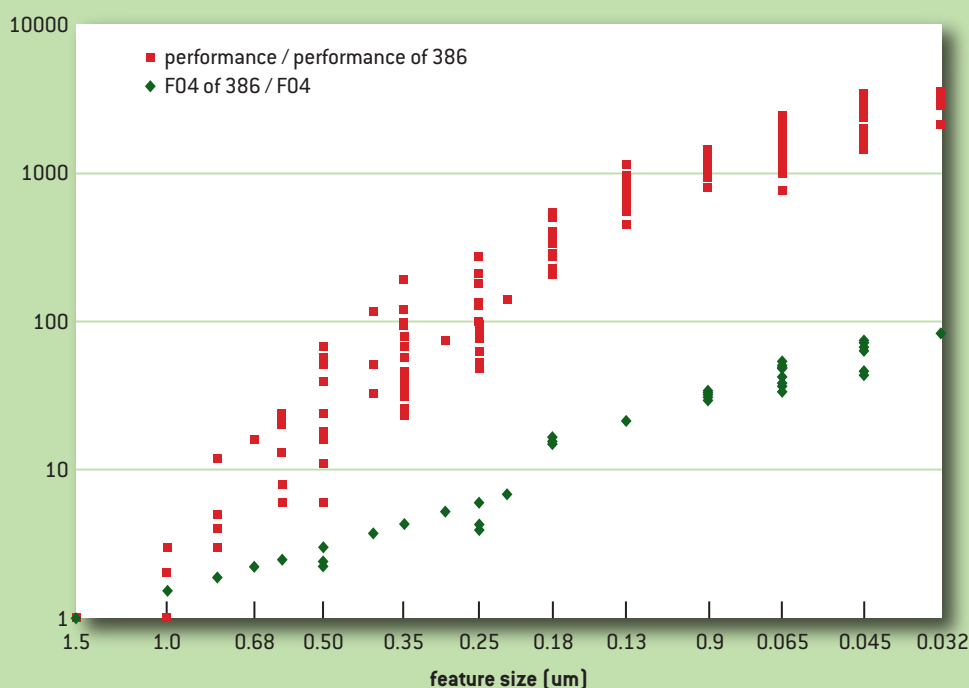
$$\frac{\text{energy}}{\text{op}} \propto \frac{P_1}{\text{Perf}_1} \frac{\lambda_2 V_2^2}{\lambda_1 V_1^2} + \frac{P_{\text{cache}}}{\text{Perf}_1} \frac{\text{FO4}_2}{\text{FO4}_1}$$

With these expressions, it is possible to normalize CPU DB processors' performance and energy into a single process technology. While Intel's Shekhar Borkar *et al.* gave a rough sketch of how technology scaling and architectural improvement contributed to processor performance over the years,<sup>2</sup> our data and normalization method can be used to generate an actual scatter plot showing the breakdown between the two factors: faster transistors (resulting from technology scaling) and architectural improvement. As seen in figure 1, process scaling and microarchitectural scaling each contribute nearly the same amount to processor performance gains.

As a quick sanity check for our normalization results, we plot normalized performance versus transistor count and normalized area in figures 2 and 3. These plots look at Pollack's rule, which states that performance scales as the square root of design complexity.<sup>1</sup> Pollack's rule has been used in numerous published studies to compare performance against processor die resource usage.<sup>2,4,10,15</sup>

## FIGURE 1

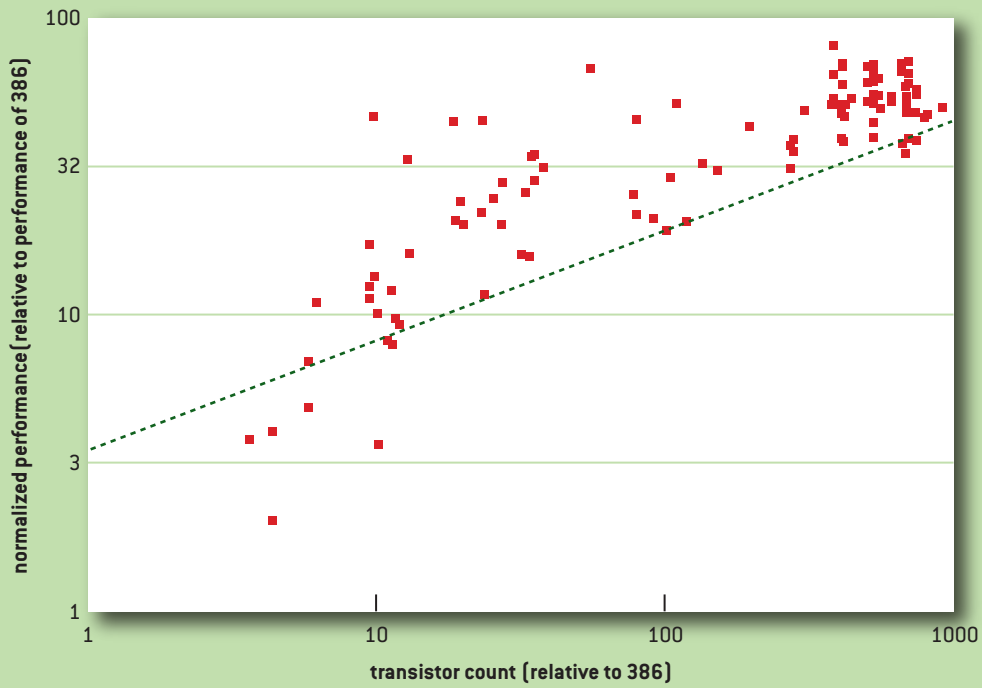
### Processor Performance Improvements Over Time



All processors are normalized to the performance of the Intel 386. The squares indicate how processor performance actually scaled with time, while the diamonds denote how much speedup came from improving the manufacturing process.

FIGURE 2

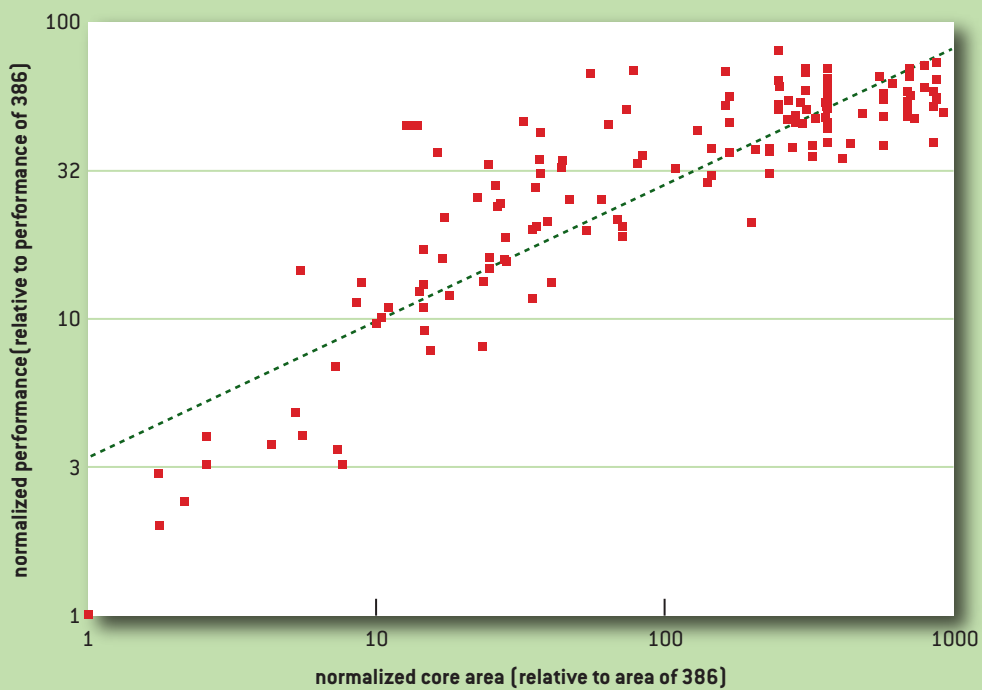
Pollack's Rule Using CPU-DB: Performance vs. Transistor Count \*



\* The regression yields  $Perf_{norm} \propto n_{trans}^{0.37}$

FIGURE 3

Pollack's Rule Using CPU-DB: Performance vs. Normalized Area \*



\* The regression yields  $Perf_{norm} \propto n_{trans}^{0.46}$

Figures 2 and 3 show that our normalized data is in close agreement with Pollack's rule, suggesting that our normalization method accurately represents design performance.

#### PHYSICAL SCALING

One of the nice side benefits of collecting this database is that it allows one to see how chip complexity, voltage, and power have scaled over time, and how well scaling predictions compare with reality. The rate of feature scaling has accelerated in recent years (figure 4). Up through the 130 nm (nanometer) process generation, feature size scaled down by a factor of

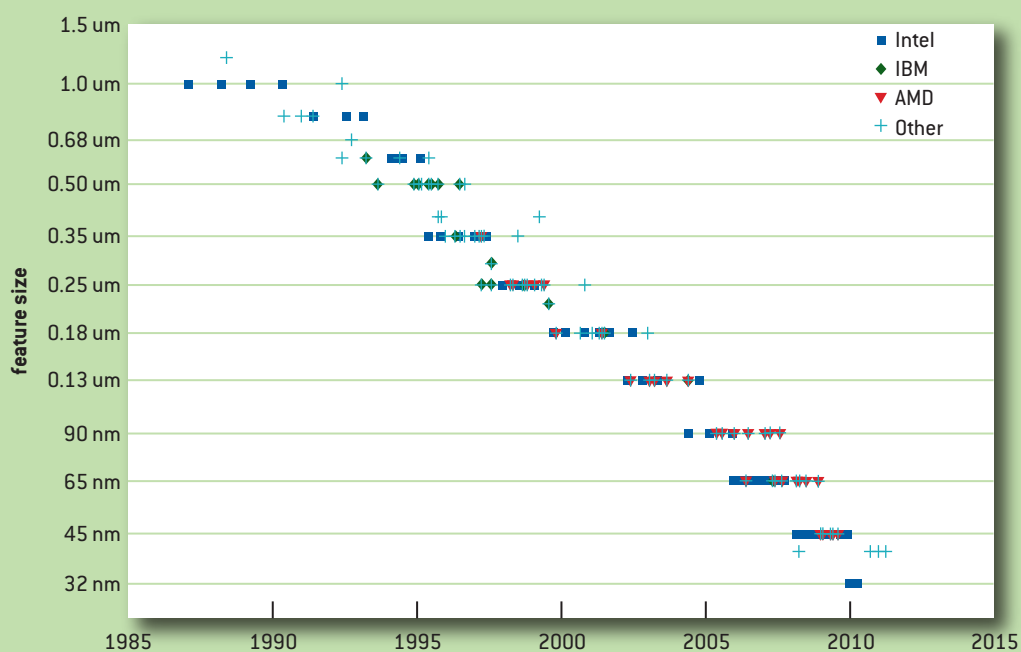
$$\alpha = \frac{1}{\sqrt{2}}$$

roughly every two to three years. Since the 90 nm generation, however, a new process has been introduced roughly every two years. Intel appears to be driving this intense schedule and has been one of the first to market for each process since the 180 nm generation.

As a result of this exponential scaling, in the 25 years since the release of the Intel 80386, transistor area has shrunk by a factor of almost 4,000. If feature size scaling were all that were driving processor density, then transistor counts would have scaled by the same rate. An analysis of commercial microprocessors, however, shows that transistor count has actually grown by a factor of 16,000.

One simple reason why transistor growth has outpaced feature size is that processor dies have

**FIGURE 4** Scaling of Transistor Feature Sizes Over Time

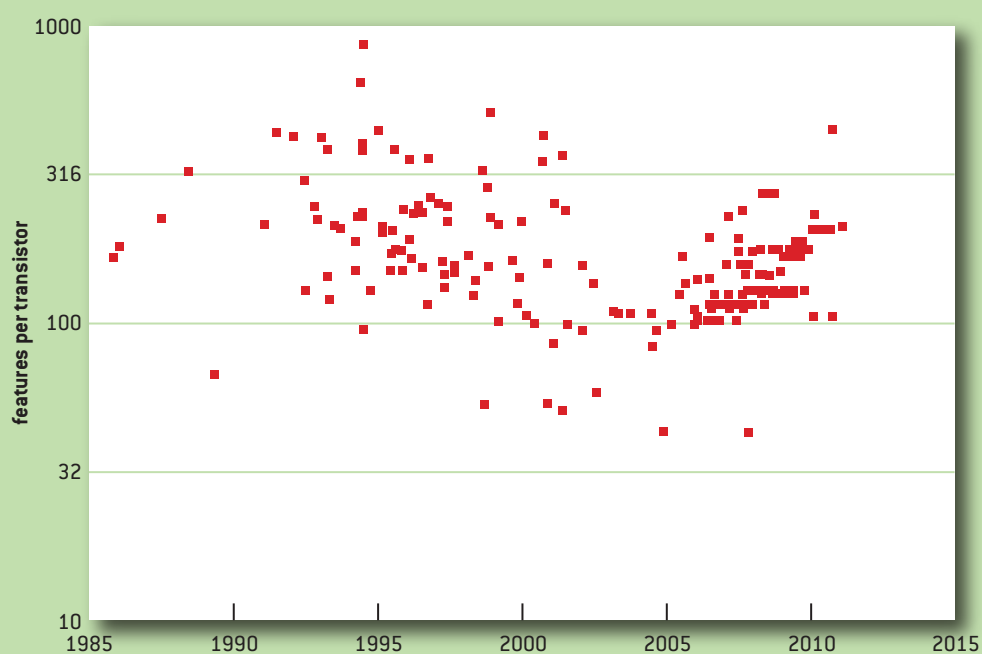


grown. While the 80386 microprocessor had a die size of 103 mm<sup>2</sup>, modern Intel Core i7 dies have an area of up to 296 mm<sup>2</sup>. This is not the whole story behind transistor scaling, however. Figure 5 shows technology-independent transistor density by plotting how many square minimum features an average processor transistor occupies. We generated this data by taking the die area, dividing by the feature size squared, and then dividing by the number of transistors. From 1985 to 2005 increasing metal layers and larger cache structures (with their high transistor densities) had decreased the average size of a transistor by four times. Interestingly, since 2005, transistor density actually dropped by roughly a factor of two. While our data does not indicate a reason for this change, we suspect it results from a combination of stricter design rules for sub-wavelength lithography, using more robust logic styles in the processor, and a shrinking percentage of the processor area used for cache in chip multiprocessors.

Our data also provides some interesting insight into how supply voltages have scaled over time. Most people know voltage scales with technology feature size, so many assume that this scaling is proportional to feature size as originally proposed in Robert Dennard's 1974 article.<sup>6</sup> As he and others have noted, however, and as shown in figure 6, voltage has not scaled at the same pace as feature size.<sup>3,12</sup> Until roughly the 0.6 μm node, processors maintained an operating voltage of 5 volts, since that was the common supply voltage for popular logic families of the day, and processor power dissipation was not an issue. It was not until manufacturers went to 3.3 volts in the 0.6 μm generation that voltage began to scale with feature size. Fitting a curve on the voltage data from the

## FIGURE 5

### Number of Square Features Per Transistor Over Time



*Note that features per transistor fell until about 2004, indicating a growth in technology-independent transistor density. In modern chips, transistors have started to grow.*



half-micron to the 0.13  $\mu\text{m}$  process generations, our data indicates that, even when voltage scaled, it did so with roughly the square root of feature size. This slower scaling has been attributed to reaping a dual benefit of faster gates and better immunity to noise and process variations at the cost of higher chip-power density.

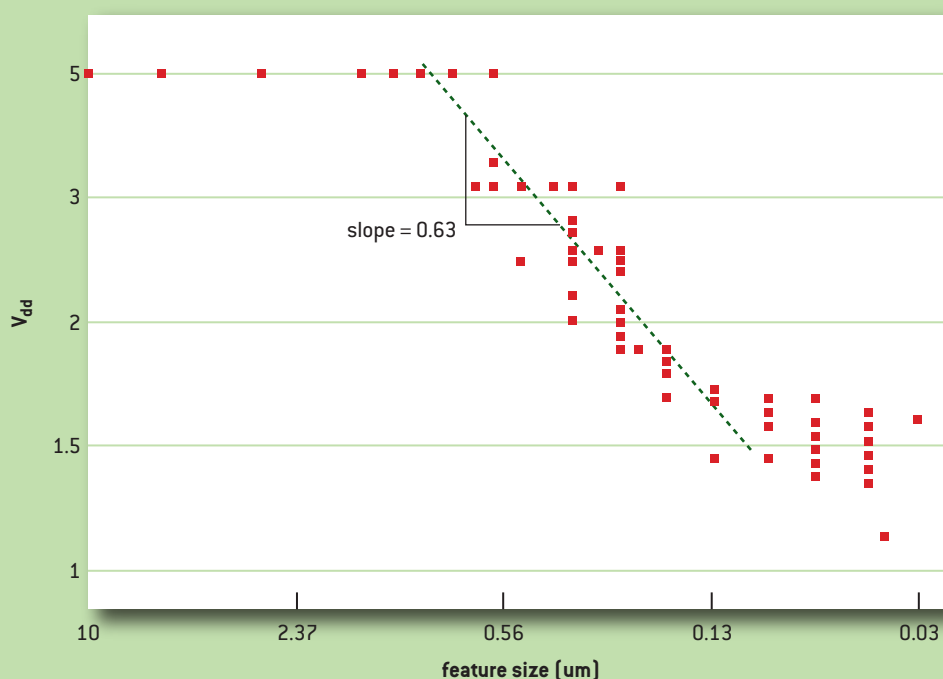
From the 0.13  $\mu\text{m}$  generation on, voltage scaling seems to have slowed. At the same time, however, trends in voltage have become much harder to estimate from our data. As mentioned earlier, today almost all processors define their own operating voltage. The data sheets have only the operating range. Figure 6 plots the maximum specified voltage. More user data should provide insight on how supply voltages are really scaling.

#### CIRCUITS AND PIPELINING

Circuit designers and microarchitects were not content to scale frequency with gate speed—if they had been, then microprocessors would be running at only around 500 MHz today. As figure 7 shows, frequencies scaled much faster than simple gate speed. The reason for this discrepancy is largely because of architectural decisions that decreased the logic depth in each processor pipeline stage and increased the number of stages. From 1985 to around 2000, the frequency rapidly increased as a result of faster, more parallel circuit implementations of adders, branch units, and caches, and the use of aggressive pipelining. These trends are evident in the contrast between the two-stage fetch/execute pipeline of the Intel 80386, and the 30-plus pipeline stages in the Prescott Pentium IV.

## FIGURE 6

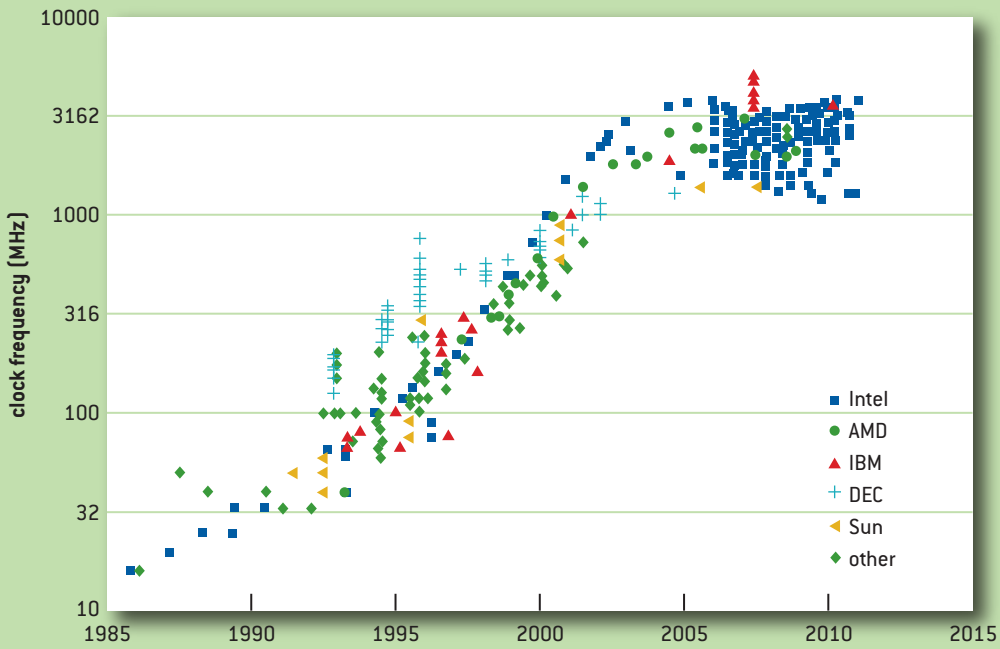
### Voltage Versus Feature Size



*It is clear that voltage scaling did not follow one simple rule. First, by convention, it was maintained at 5 volts. Once voltage reductions were required, a new convention was established at 3.3 volts. Then voltage was reduced in proportion to feature size until the 130 nm node. Log-space regression reveals that voltage scaled roughly as the square root of feature size between the 0.6  $\mu\text{m}$  and 130 nm nodes.*

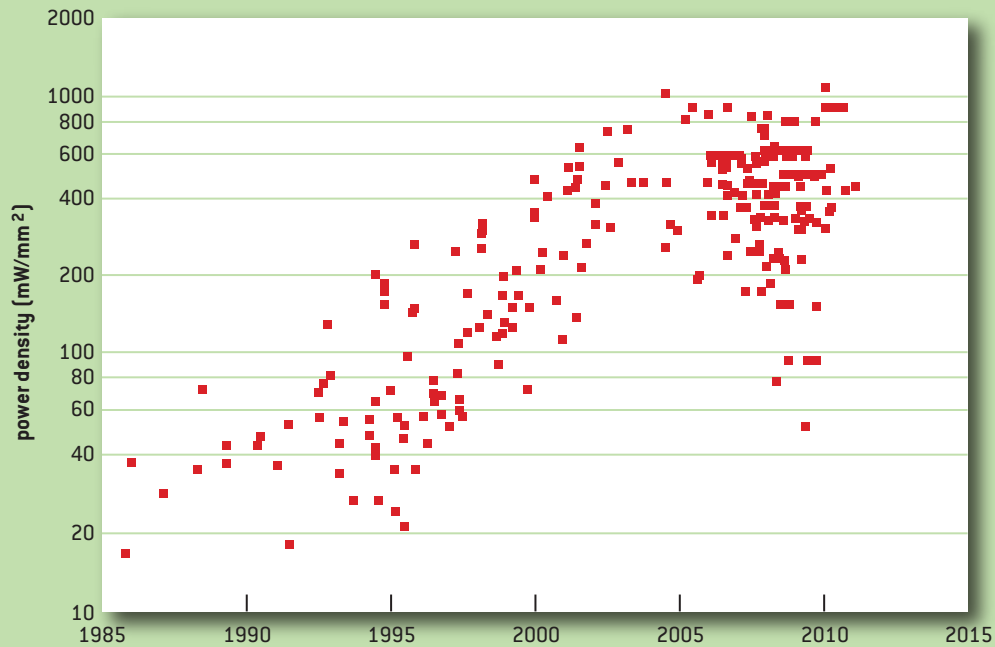
## FIGURE 7

Processor Frequency Scaling Over Time



## FIGURE 8

Power Density Over Time



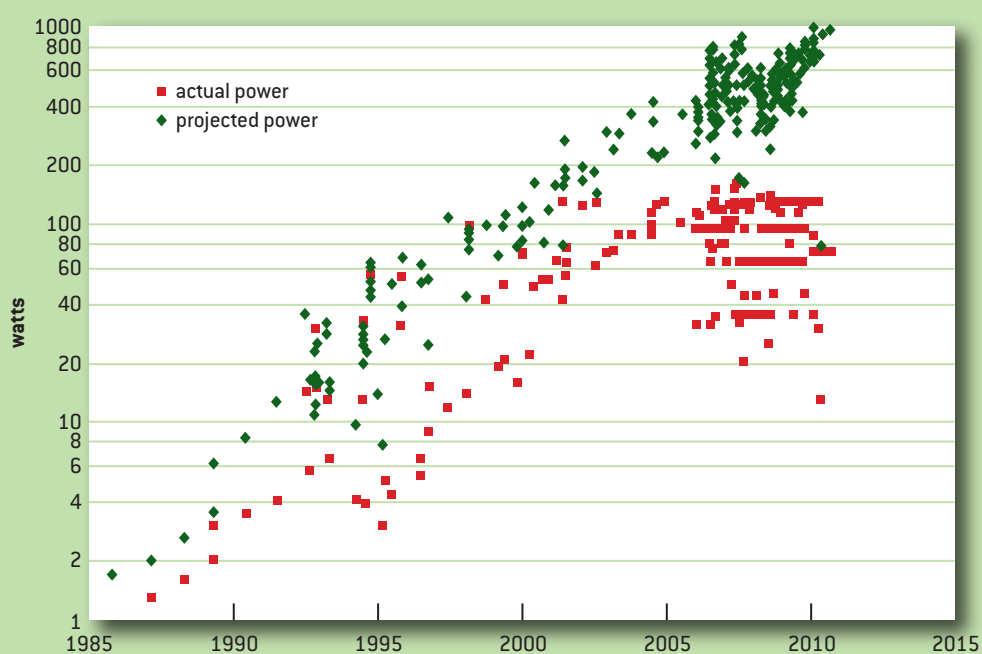
Since 2000, processor frequencies have stagnated, but this is not the whole story. Our data confirms that gate speeds have continued to improve with technology. What is different now, though, is that the industry has moved away from deeply pipelined machines and is now designing machines that do more work per pipeline stage. The reason for this change is simple: power. While short-tick machines are possible and might be optimal from a performance perspective,<sup>7,11,14</sup> they are not energy efficient.<sup>8</sup>

In light of slower voltage scaling and faster frequency scaling, it comes as no surprise that processor power has increased over time. As illustrated in figure 8, processor power density has increased by more than a factor of 32 from the release of the 80386 through 2005, although it has recently started to decrease as energy-efficient computing has grown in importance.

Interestingly, scaling rules say power should be much worse. From the Intel 80386 to a Pentium 4, feature size scaled by 16 times, supply voltage scaled by around four times, and frequency scaled by 200 times. This means that the power density should have increased by a factor of  $16 \cdot 200/4^2 = 200$ , which is much larger than the power density increase of 32 times shown in figure 8. Figure 9 compares observed power with how power should have scaled if we just scaled up an Intel 386 architecture to match the performance of new processors. The eightfold savings represents circuit and microarchitectural optimizations—such as clock gating—that have been done during this period to keep power under control. The energy savings of these techniques had initially been growing, but, unfortunately, recently seem to have stabilized at around the eightfold mark. This is

## FIGURE 9

### How Power Should Have Scaled



not a good sign if we hope to continue to scale performance, since technology scaling of energy is slowing down.

#### MICROARCHITECTURE AND SOFTWARE

While process technologists were finding ways to scale transistors, processor architects were working equally hard in advancing and innovating at the microarchitecture level. Indeed, this effect can be seen in CPU DB where, after normalizing for technology, we observe a hundredfold improvement in microarchitecture/software performance since the Intel 80386 days. Historically, as the number of transistors per chip increased with technology scaling, architects found ways to use those transistors to create faster, more advanced uniprocessors. In addition to aggressive clock scaling, architects implemented features such as speculative execution, parallel instruction issue, out-of-order processing, and larger caches—all of which contributed to improved single-threaded performance.

By roughly 2005, increasingly complex processors, along with slowed voltage scaling, caused processors to hit a new constraint: the power wall. This resulted in a significant shift in the industry. Moore's law meant that processor designers could still expect an ever-increasing number of transistors, but they had to use these transistors in energy-efficient ways; increasing performance now meant decreasing energy/instruction to keep power constant. As a response to this challenge, the industry transitioned toward CMP (chip multiprocessor) designs that use many simple processors to increase the aggregate performance of the chip.

Figure 10 plots the technology-normalized energy/op versus the normalized performance. For

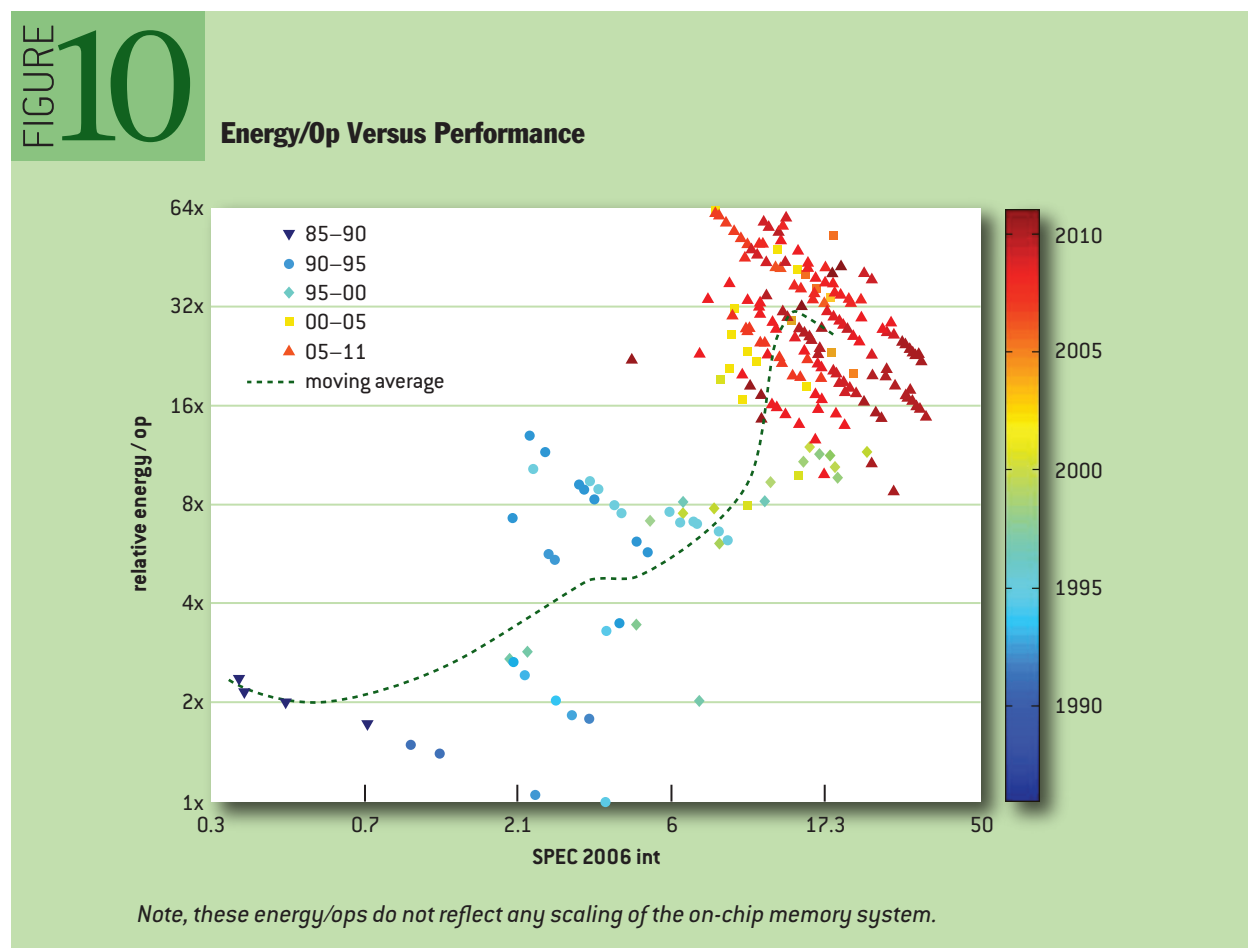


FIGURE 11

Energy/Op Versus Performance, Modified to Scale Up the Memory System of Older Cores

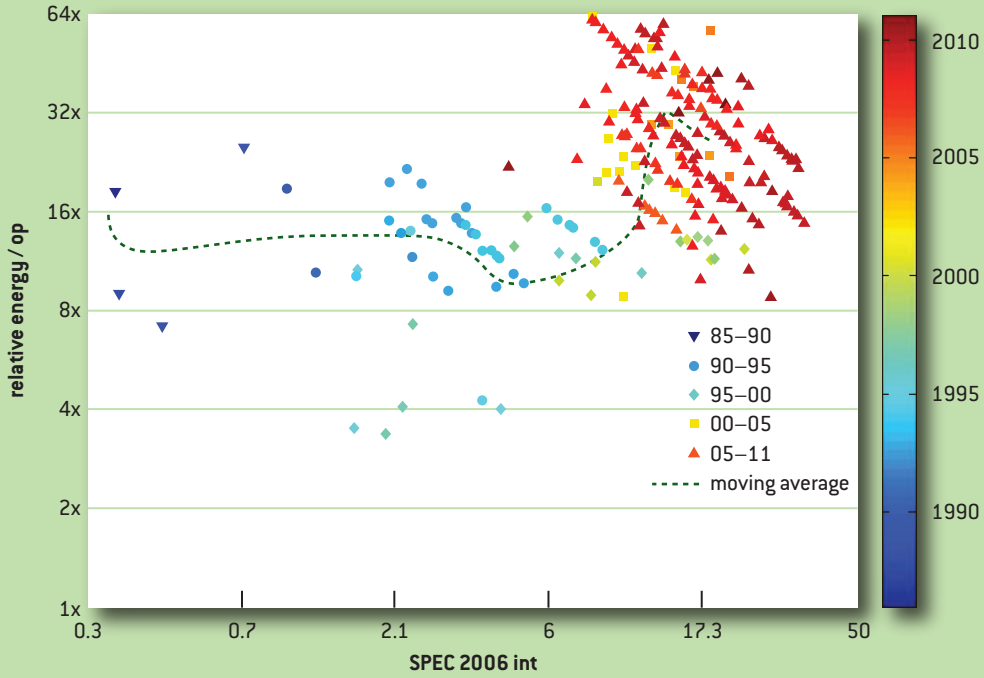
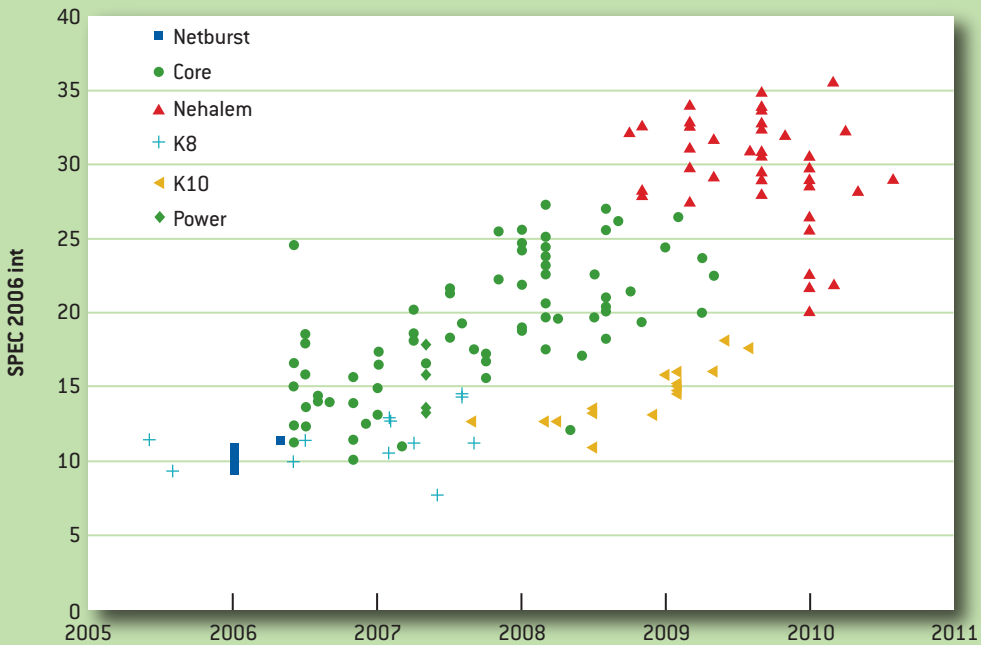


FIGURE 12

Performance Versus Year Since 2005



this plot, we assume that the power needed to scale up the cache size is small compared with the processor power, providing an optimistic assumption of the efficiency of these early machines. This plot indicates that, for early processor designs, energy/op remains relatively constant while performance scales up.

We noticed from this plot, however, that some of the early processors (e.g., the Pentium) appear far more energy efficient than modern processor designs. To estimate the scaled energy of these processors more fairly, we scale the caches by the square of the improvement in frequency to keep the memory stall percentage constant, and we estimate the power of a 45 nm low-power SRAM at around 0.5 W/MB. Including this cache energy-correction factor yields the results in figure 11. Comparing these two plots demonstrates how critical the memory system is for low-energy processors. The leakage power of our estimated large on-chip cache increases the energy cost of an instruction by four to eight times for simple processors.

Surprisingly, however, the original Pentium designs are still substantially more energy efficient than other designs in the plot. Clearly, more analysis is warranted to understand whether this apparent efficiency can be leveraged in future machines.

In recent years, desktop processors have shifted toward high-throughput parallel machines. With this shift, it was unclear whether processor designers would be able to scale single-core performance. A brief analysis of the data in figure 12 shows that single-core performance continues to scale with each new architecture. Within an architecture, performance depends largely on the part's frequency

## FIGURE 13

Performance Versus Clock Rate and Cache Size Since 2005

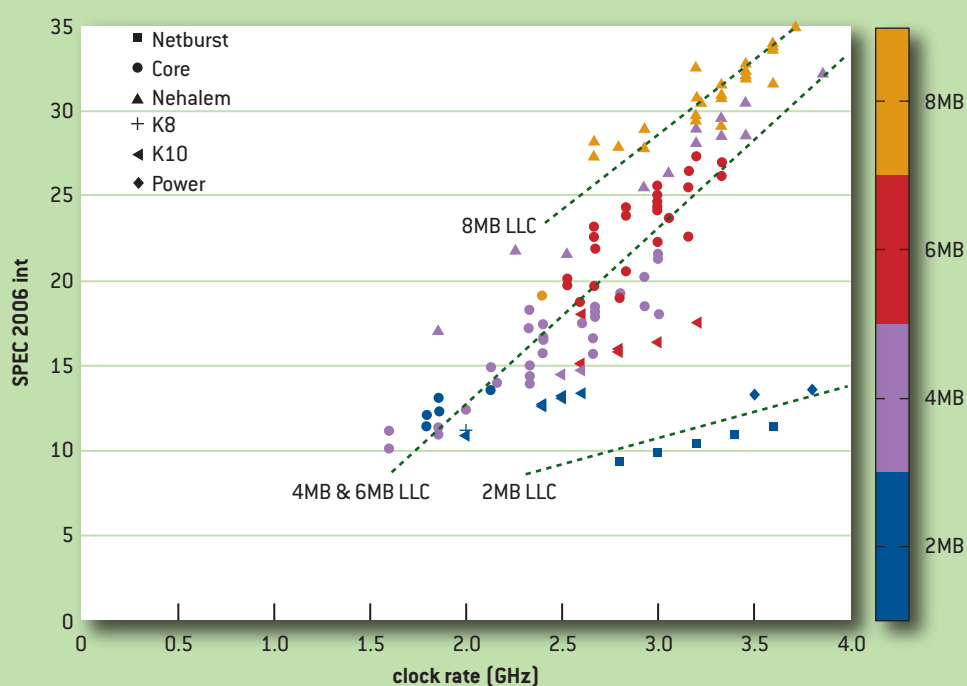
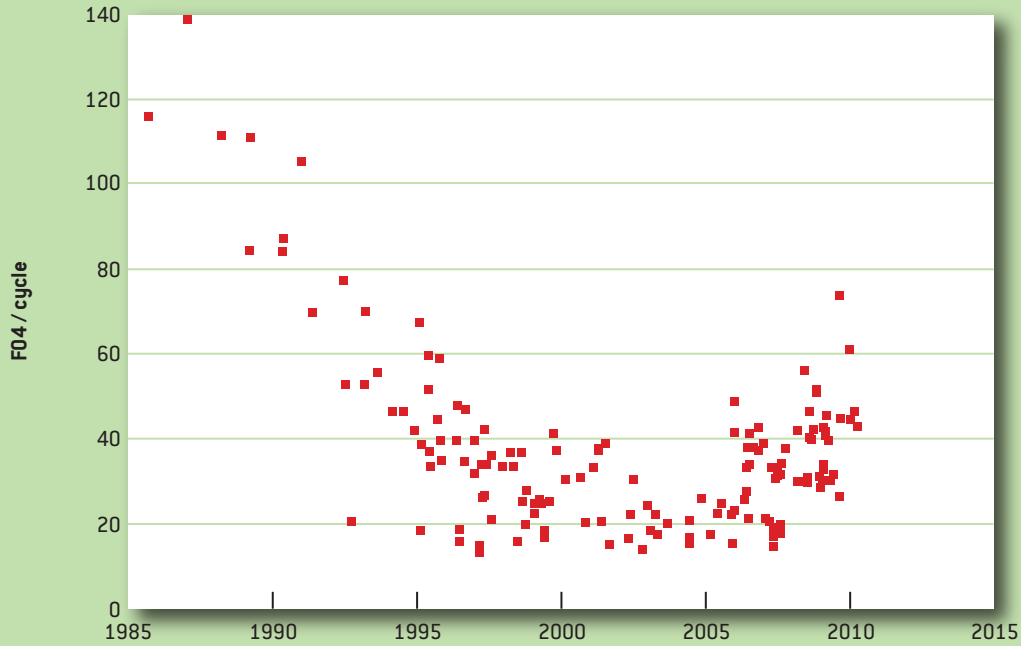


FIGURE 14

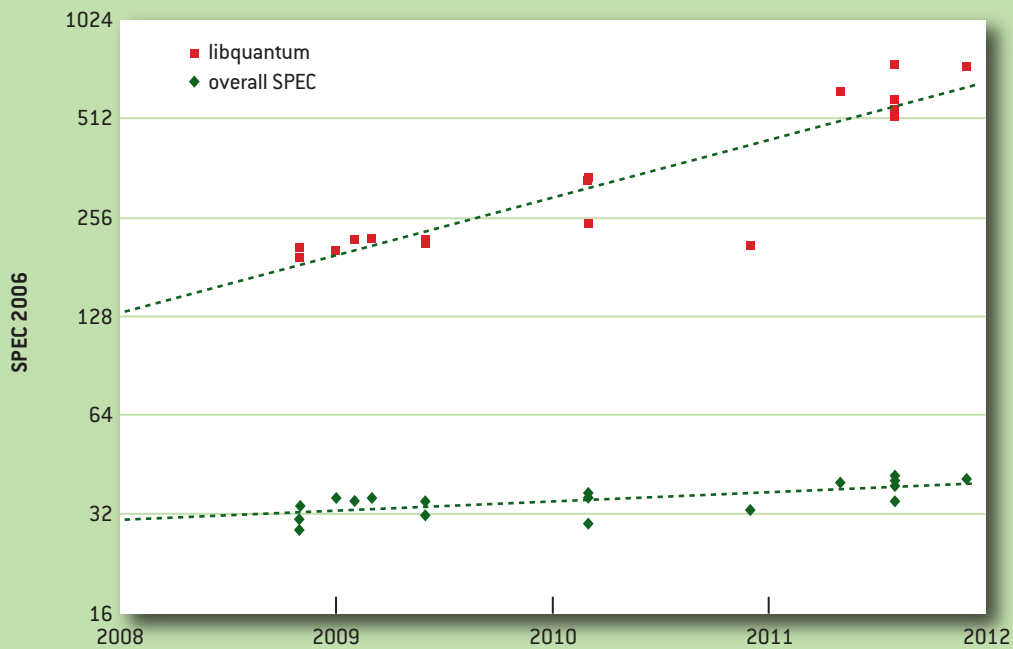
F04 Delays Per Cycle for Processor Designs



F04 delay per cycle is roughly proportional to the amount of computation completed per cycle.

FIGURE 15

Libquantum Score Versus SPEC Score



This figure shows how compiler optimizations have led to performance boosts in Libquantum.

and cache size. Figure 13 illustrates this point by plotting the performance versus frequency and cache size for several modern processor designs. Frequency scaling with each new architecture is slower than before, and peak frequencies are now often used only when the other processor cores are idle. Figure 14 plots cycle time measured in gate delays and shows why processor clock frequency seems to have stalled: processors moved to shorter pipelines, and the resulting slower frequency has taken some time to catch up to the older hyperpipelined rates.

More interesting is that even when controlling for the effects of frequency and cache size, single-core microarchitectural performance is still being improved with each generation of chips (figure 13). Improvements such as on-chip memory controllers and extra execution units all play a role in determining overall system efficiency, and architects are still finding improvements to make.

Our results, however, come with the caveat that some portion of the performance improvement in modern single-core performance comes from compiler optimizations. Figure 15 shows how performance of the SPEC 2006 benchmark Libquantum scales over time on the Intel Bloomfield architecture. Libquantum concentrates a large amount of computation in an inner for loop that can be optimized. As a result, Libquantum scores have risen 18 times without any improvement to the underlying hardware. Also, many SPEC scores for modern processors are measured with the Auto Parallel flag turned on, indicating that the measured “single-core” performance might still be benefiting from multicore computing.

## CONCLUSIONS

Over the past 40 years, VLSI designers have used an incredible amount of engineering expertise to create and improve these amazing devices we call microprocessors. As a result, performance has improved and the energy/op has decreased by many orders of magnitude, making these devices the engines that power our information technology infrastructure. CPU DB is designed to help explore this area. Using the data in CPU DB and some simple scaling rules, we have conducted some preliminary studies to show the kinds of analyses that are possible. We encourage readers to explore and contribute to the processor data in CPU DB, and we look forward to learning more about processors from the insights they develop.

*For more information about the online CPU database and how to contribute data, please visit <http://cpudb.stanford.edu>.*

## CONTRIBUTING AUTHORS:

Omid Azizi (Hicamp Systems, Inc.)  
John S. Brunhaver II (Stanford University)  
Ron Ho (Oracle)  
Stephen Richardson (Stanford University)  
Ofer Shacham (Stanford University)  
Alex Solomatnikov (Hicamp Systems, Inc.)

## REFERENCES

1. Borkar, S. 2007. Thousand core chips: a technology perspective. In *Proceedings of the 44th annual Design Automation Conference*: 746-749; <http://doi.acm.org/10.1145/1278480.1278667>.



2. Borkar, S., Chien, A. A. 2011. The future of microprocessors. *Communications of the ACM* 54(5): 67-77; <http://doi.acm.org/10.1145/1941487.1941507>.
3. Chang, L., Frank, D., Montoye, R., Koester, S., Ji, B., Coteus, P., Dennard, R., Haensch, W. 2010. Practical strategies for power-efficient computing technologies. *Proceedings of the IEEE* 98(2): 215-236.
4. Chung, E. S., Milder, P. A., Hoe, J. C., Mai, K. 2010. Single-chip heterogeneous computing: does the future include custom logic, FPGAs, and GPGPUs? In *Proceedings of the 43rd Annual IEEE/ACM International Symposium on Microarchitecture*: 225-236; <http://dx.doi.org/10.1109/MICRO.2010.36>.
5. CoreMark, an EEMBC Benchmark. CoreMark scores for embedded and desktop CPUs; <http://www.coremark.org/home.php>.
6. Dennard, R., Gaensslen, F., Yu, H., Rideout, V., Bassous, E., LeBlanc, A. 1999. Design of ion-implanted MOSFETs with very small physical dimensions. *Proceedings of the IEEE* 87(4): 668-678 (reprinted from *IEEE Journal of Solid-State Circuits*, 1974).
7. Hartstein, A., Puzak, T. 2002. The optimum pipeline depth for a microprocessor. *Proceedings of the 29th Annual International Symposium on Computer Architecture*: 7-13.
8. Hartstein, A., Puzak, T. 2003. Optimum power/performance pipeline depth. *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture* (Dec.): 117-125.
9. Hartstein, A., Srinivasan, V., Puzak, T. R., Emma, P. G. 2006. Cache miss behavior: is it  $\sqrt{}$ ? *Proceedings of the 3rd Conference on Computing Frontiers*: 313-320; <http://doi.acm.org/10.1145/1128022.1128064>.
10. Hill, M., Marty, M. 2008. Amdahl's law in the multicore era. *Computer* 41(7): 33-38.
11. Hrishikesh, M., Jouppi, N., Farkas, K., Burger, D., Keckler, S., Shivakumar, P. 2002. The optimal logic depth per pipeline stage is 6 to 8 FO4 inverter delays. *Proceedings of the 29th Annual International Symposium on Computer Architecture*: 14-24.
12. Nowak, E. J. 2002. Maintaining the benefits of CMOS scaling when scaling bogs down. *IBM Journal of Research and Development* 46(2.3):169-180.
13. SPEC (Standard Performance Evaluation Corporation). 2006. SPEC CPU2006 results; <http://www.spec.org/cpu2006/results/>.
14. Sprangle, E., Carmean, D. 2002. Increasing processor performance by implementing deeper pipelines. *Proceedings of the 29th Annual International Symposium on Computer Architecture*: 25-34.
15. Woo, D. H., Lee, H.-H. 2008. Extending Amdahl's law for energy-efficient computing in the many-core era. *Computer* 41(12): 24-31.
16. Zhang, X. 2003. High-performance low-leakage design using power compiler and multi-Vt libraries. SNUG (Synopsys Users Group) Europe.

### LOVE IT, HATE IT? LET US KNOW

[feedback@queue.acm.org](mailto:feedback@queue.acm.org)

**Andrew Danowitz** received his B.S. in General Engineering from Harvey Mudd College, and an M.S. in Electrical Engineering from Stanford University, and he is currently a Ph.D. candidate in Electrical Engineering at Stanford University. Andrew's research focuses on reconfigurable hardware design methodologies and energy efficient architectures.

**Kyle Kelley** is a Ph.D candidate in Electrical Engineering at Stanford University. His research interests include energy-efficient VLSI design and parallel computing. Kelley has a BS in engineering and a BA in economics from Harvey Mudd College, and an MS in electrical engineering from Stanford University.

**James Mao** is a Ph.D. candidate in the VLSI Research Group at Stanford University, working on verification tools for mixed-signal designs. He holds a BS in electrical engineering from the California Institute of Technology and a MS in electrical engineering from Stanford University.

**John P. Stevenson** is a graduate of the U.S. Naval Academy and served as an officer on board the *USS Los Angeles* (SSN-688) prior to entering the Ph.D. program at Stanford University. His interests include computer architecture and digital circuit design. He is investigating novel memory-system architectures. When not working, he enjoys surfing.

**Mark Horowitz** is the Chair of the Electrical Engineering Department and the Yahoo! Founders Professor at Stanford University, and a founder of Rambus, Inc. He is a fellow of IEEE and ACM and is a member of the National Academy of Engineering and the American Academy of Arts and Science. Dr. Horowitz's research interests are quite broad and range from using EE and CS analysis methods on problems in molecular biology to creating new design methodologies for analog and digital VLSI circuits.

© 2012 ACM 1542-7730/12/0200 \$10.00