By David A. Patterson

LATENCY LAGS BANDWITH

Recognizing the chronic imbalance between bandwidth and latency, and how to cope with it.

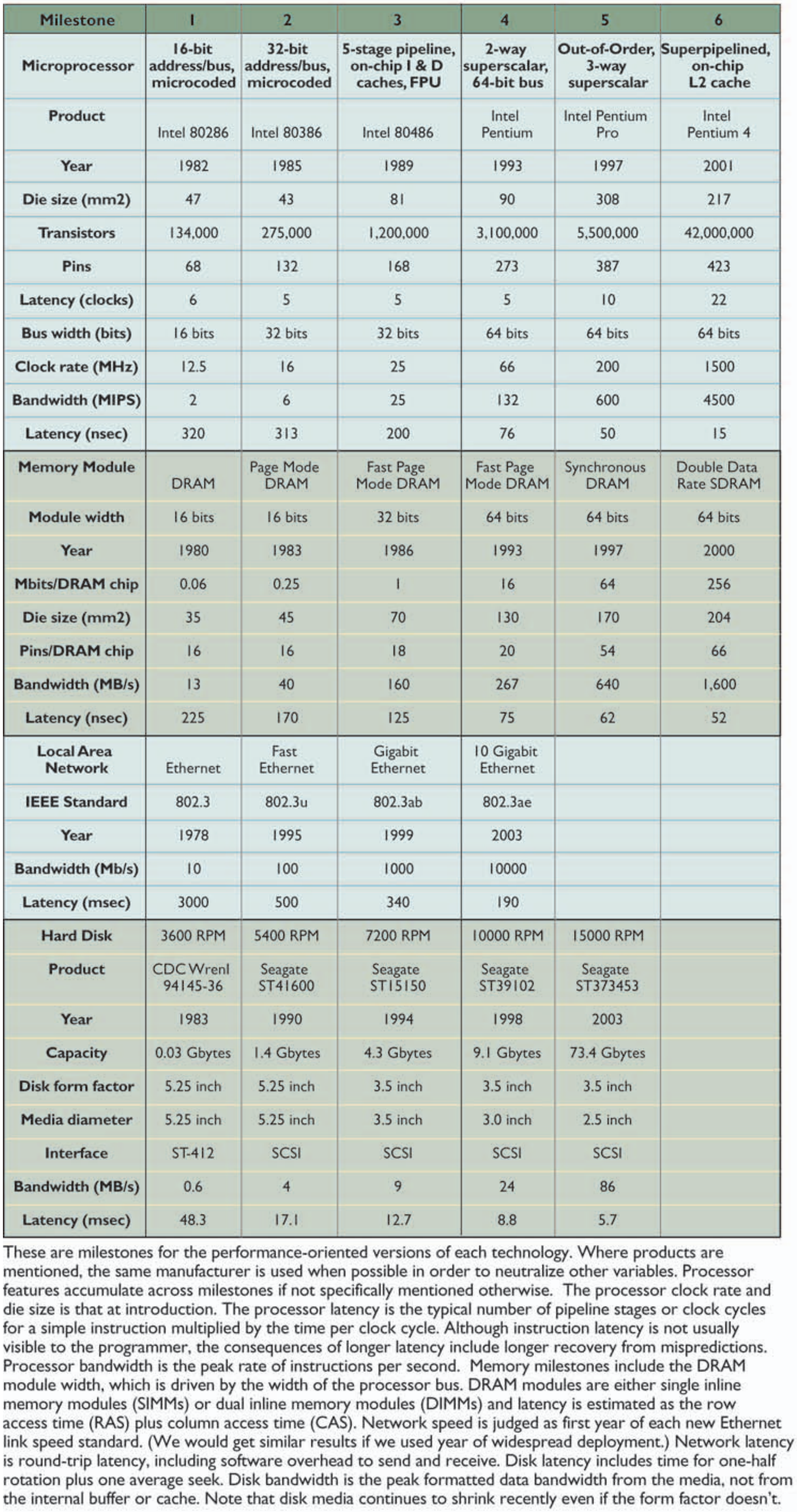
As I review performance trends, I am struck by a consistent theme across many technologies: bandwidth improves much more quickly than latency. Here, I list a half-dozen performance milestones to document this observation, many reasons why it happens, a few ways to cope with it, a rule of thumb to quantify it, plus an example of how to design systems differently based on this observation.

Table 1 identifies four to six performance milestones over the last 20 to 25 years for microprocessors, memory, networks, and disks. The microprocessor milestones are six generations of IA-32 processors, going from a 16-bit bus, microcoded 80286 to a 64-bit bus, superscalar, out-of-order execution, superpipelined Pentium 4. Memory module milestones go from 16-bit wide, plain DRAM to 64-bit wide double data rate synchronous DRAM. Ethernet advanced from 10Mbps to 10Gbps. Disk milestones are based on rotation speed, improving from 3600RPM to 15000RPM. In each case we record best-case bandwidth and record latency as the time for a simple operation assuming there is no contention.

Figure 1 plots the relative improvement in bandwidth and latency for each of those milestones. Performance is the primary differentiator for microprocessors and networks, so they have seen the greatest gains: 1000–2000X in bandwidth and 20–40X in latency. Capacity is generally more important than performance for memory and disks, so capacity has improved most, yet their bandwidth advances are still much greater than their gains in latency. Clearly, bandwidth has outpaced latency across these technologies.

To find a rule of thumb to quantify the trends, Table 2 summarizes the average improvement for each of these technologies. The first three rows give the annual rate of improvement in latency, capacity, and bandwidth. The next row shows how many years it

Table 1. Performance milestones in bandwidth and latency for processors, memory modules, local area networks, and disks [3, 5].



Milestone	1	2	3	4	5	6
Microprocessor	16-bit address/bus, microcoded	32-bit address/bus, microcoded	5-stage pipeline, on-chip I & D caches, FPU	2-way superscalar, 64-bit bus	Out-of-Order, 3-way superscalar	Superpipelined, on-chip L2 cache
Product	Intel 80286	Intel 80386	Intel 80486	Intel Pentium	Intel Pentium Pro	Intel Pentium 4
Year	1982	1985	1989	1993	1997	2001
Die size (mm2)	47	43	81	90	308	217
Transistors	134,000	275,000	1,200,000	3,100,000	5,500,000	42,000,000
Pins	68	132	168	273	387	423
Latency (clocks)	6	5	5	5	10	22
Bus width (bits)	16 bits	32 bits	32 bits	64 bits	64 bits	64 bits
Clock rate (MHz)	12.5	16	25	66	200	1500
Bandwidth (MIPS)	2	6	25	132	600	4500
Latency (nsec)	320	313	200	76	50	15
Memory Module	DRAM	Page Mode DRAM	Fast Page Mode DRAM	Fast Page Mode DRAM	Synchronous DRAM	Double Data Rate SDRAM
Module width	16 bits	16 bits	32 bits	64 bits	64 bits	64 bits
Year	1980	1983	1986	1993	1997	2000
Mbits/DRAM chip	0.06	0.25	1	16	64	256
Die size (mm2)	35	45	70	130	170	204
Pins/DRAM chip	16	16	18	20	54	66
Bandwidth (MB/s)	13	40	160	267	640	1,600
Latency (nsec)	225	170	125	75	62	52
Local Area Network	Ethernet	Fast Ethernet	Gigabit Ethernet	10 Gigabit Ethernet		
IEEE Standard	802.3	802.3u	802.3ab	802.3ae		
Year	1978	1995	1999	2003		
Bandwidth (Mb/s)	10	100	1000	10000		
Latency (msec)	3000	500	340	190		
Hard Disk	3600 RPM	5400 RPM	7200 RPM	10000 RPM	15000 RPM	
Product	CDC Wrenl 94145-36	Seagate ST41600	Seagate ST15150	Seagate ST39102	Seagate ST373453	
Year	1983	1990	1994	1998	2003	
Capacity	0.03 Gbytes	1.4 Gbytes	4.3 Gbytes	9.1 Gbytes	73.4 Gbytes	
Disk form factor	5.25 inch	5.25 inch	3.5 inch	3.5 inch	3.5 inch	
Media diameter	5.25 inch	5.25 inch	3.5 inch	3.0 inch	2.5 inch	
Interface	ST-412	SCSI	SCSI	SCSI	SCSI	
Bandwidth (MB/s)	0.6	4	9	24	86	
Latency (msec)	48.3	17.1	12.7	8.8	5.7	

These are milestones for the performance-oriented versions of each technology. Where products are mentioned, the same manufacturer is used when possible in order to neutralize other variables. Processor features accumulate across milestones if not specifically mentioned otherwise. The processor clock rate and die size is that at introduction. The processor latency is the typical number of pipeline stages or clock cycles for a simple instruction multiplied by the time per clock cycle. Although instruction latency is not usually visible to the programmer, the consequences of longer latency include longer recovery from mispredictions. Processor bandwidth is the peak rate of instructions per second. Memory milestones include the DRAM module width, which is driven by the width of the processor bus. DRAM modules are either single inline memory modules (SIMMs) or dual inline memory modules (DIMMs) and latency is estimated as the row access time (RAS) plus column access time (CAS). Network speed is judged as first year of each new Ethernet link speed standard. (We would get similar results if we used year of widespread deployment.) Network latency is round-trip latency, including software overhead to send and receive. Disk latency includes time for one-half rotation plus one average seek. Disk bandwidth is the peak formatted data bandwidth from the media, not from the internal buffer or cache. Note that disk media continues to shrink recently even if the form factor doesn't.

takes bandwidth to double at that annual rate of improvement. The last two rows show the advance in capacity and latency in that bandwidth doubling time.

To see if the results hold when performance improves rapidly, Table 3 illustrates a more recent version of same information, this time limited to the three latest milestones. The time for bandwidth to double in the last decade has indeed shrunk, especially for networks. However, the relative improvements in latency over that shorter time are still similar to the latency improvements over the longer time of Table 2.

Thus, using either the last two decades, or just the more recent period, performance advances of these four disparate technologies is captured by the following rule of thumb:

In the time that bandwidth doubles, latency improves by no more than a factor of 1.2 to 1.4.

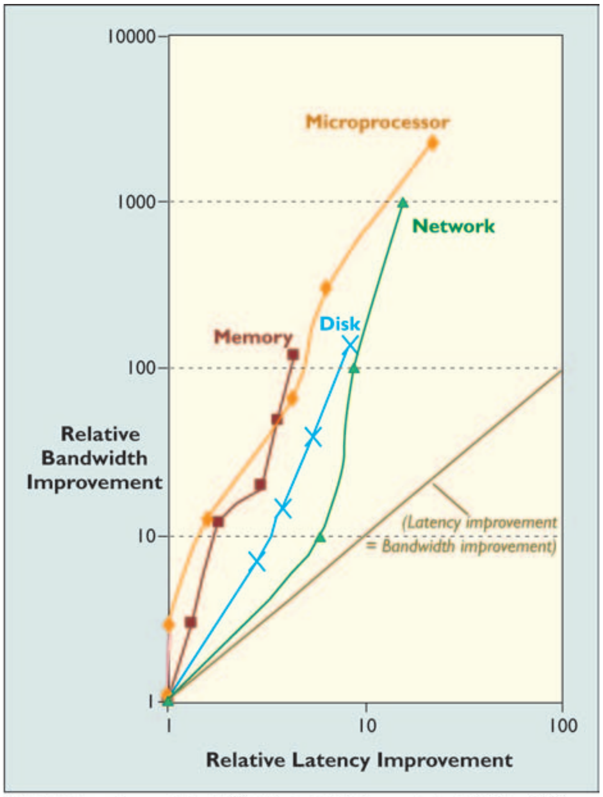
A more dramatic statement is that *bandwidth improves by at least the square of the improvement in latency.* Note that even in these performance-oriented versions of memory and storage, capacity improves more rapidly than bandwidth.

Reasons for Bountiful Bandwidth but Lagging Latency

"There is an old network saying: Bandwidth problems can be cured with money. Latency problems are harder because the speed of light is fixed—you can't bribe God."
—*Anonymous*

Given the record of advances in bandwidth versus latency, the logical question is why? Here are five technical reasons and one marketing reason.

1. Moore's Law helps bandwidth more than latency. The scaling of semiconductor processes provides both faster transistors and many more on a chip. Moore's Law predicts a periodic doubling in the number of transistors per chip, due to scaling and in part to larger chips; recently, that rate has been 22–24 months [6]. Bandwidth is helped by faster transistors, more transistors, and more pins operating in parallel. The faster transistors help latency, but the larger number of transistors and the relatively longer distances on the actually larger chips limit the benefits of scaling to latency. For example, processors in Table 1 grew by more than a factor of 300 in transistors, and by more than a factor of 6 in pins, but area increased by almost a factor of 5. Since distance grows by the square root of the area, distance in Table 1 doubled.



Note that latency improved about 10X while bandwidth improved about 100X to 1000X.

Figure 1. Log-log plot of bandwidth and latency milestones from Table 1 relative to the first milestone.

2. Distance limits latency. Distance sets a lower bound to latency. The delay on the long word lines and bit lines are the largest part of the row access time of a DRAM. The speed of light tells us that if the other computer on the network is 300 meters away, its latency can never be less than one microsecond.

3. Bandwidth is generally easier to sell. The nontechnical reason that latency lags bandwidth is the marketing of performance: it is easier to sell higher bandwidth than to sell lower latency. For example, the benefits of a 10Gbps bandwidth Ethernet are likely easier to explain to customers today than a 10-microsecond latency Ethernet, no matter which actually provides better value. One can argue that greater advances in bandwidth led to marketing techniques to sell bandwidth that in turn trained customers to desire it. No matter what the real chain of events, unquestionably higher bandwidth for processors, memories, or the networks is easier to sell today than latency. Since bandwidth sells, engineering resources tend to be thrown at bandwidth, which further tips the balance.

	Processor	Memory Module	LAN	Disk
Annual Latency Improvement (all milestones)	1.17	1.07	1.12	1.09
Annual Capacity Improvement (all milestones)	--	1.52	--	1.48
Annual Bandwidth Improvement (all milestones)	1.50	1.27	1.39	1.28
Time for Bandwidth to Double in Years	1.7	2.9	2.1	2.8
Capacity Improvement in Time for Bandwidth to Double	--	3.4	--	3.0
Latency Improvement in Time for Bandwidth to Double	1.3	1.2	1.3	1.3

The fourth row shows the time for bandwidth to double, the next row shows how much capacity improves in that time, and the last row shows latency improvement in that time. The small difference per year results in large difference per decade; using the smallest imbalance as an example, disk bandwidth improved by 5.2 per decade but disk latency improved only by 2.4. Capacity improvement for cost-sensitive disks such as ATA is higher than 1.48 per year, but we focus on performance-oriented technologies such as SCSI.

Table 2. Summary of annual improvements in latency, capacity, and bandwidth in Table 1.

4. Latency helps bandwidth. Technology improvements that help latency usually also help bandwidth, but not vice versa. For example, DRAM latency determines the number of accesses per second, so lower latency means more accesses per second and hence higher bandwidth. Also, spinning disks faster reduces the rotational latency, but the read head must read data at the new faster rate as well. Thus, spinning the disk faster improves both bandwidth and rotational latency. However, increasing the linear density of bits per inch on a track helps bandwidth but offers no help to latency.

5. Bandwidth hurts latency. It is often easy to improve bandwidth *at the expense* of latency. Queuing theory quantifies how buffers help bandwidth but hurt latency. As a second example, adding chips to widen a memory module increases bandwidth but the higher fan-out on address lines may increase latency.

6. Operating system overhead hurts latency. A user program that wants to send a message invokes the operating system, which then invokes a network driver before hardware is accessed. This overhead is amortized over large messages, and thus plays a smaller role in bandwidth, but it can be a large part of the latency for short messages.

These six reasons help explain why bandwidth has outstripped latency since 1980, and why I expect it to continue. Although there may well be innovations that make a one-time reduction in latency in a given technology, I believe they will occur within the context of unrelenting improvement in bandwidth.

Coping with Bountiful Bandwidth but Lagging Latency

"If a problem has no solution, it may not be a problem, but a fact not to be solved, but to be coped with over time."

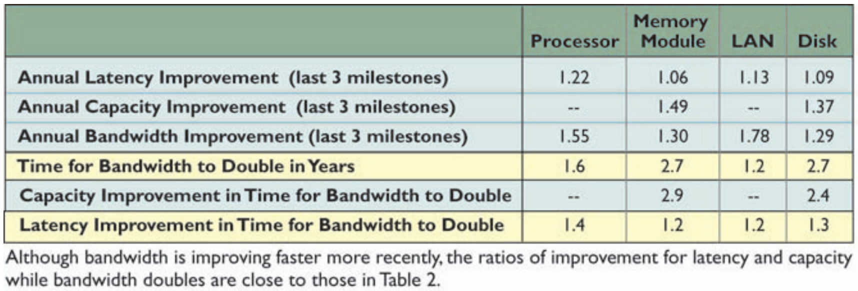
—Shimon Peres ("Peres's Law")

Despite this imbalance, latency remains important for the interactive applications either on the desktop or across a network: a quick and predictable user interaction time is critical to productivity [3].

The bandwidth-latency imbalance may not be a problem that can be solved in each technology; it may instead be a fact with which we must cope in bringing about more balanced systems. Here are three techniques developed over the years to cope with this imbalance. In fact, yet another reason for this trend may be the relative difficulty of hiding bandwidth shortages versus the plausibility of hiding some latency using techniques like caching.

- *Caching: Leveraging capacity to help latency.* Processors first added caches to overcome the long latency to memory, relying on locality of reference to allow a small fast memory to capture most accesses. About half of the area of existing large microprocessor chips is for caches. File systems routinely use a large fraction of current large main memory to act as a file cache to try to avoid going to disk, and disks today include caches of many megabytes to try to avoid accessing the disk surface.
- *Replication: Leveraging capacity to again help latency.* The increase in capacity of memory and storage makes affordable copies of data to reduce latency. Thus, ISPs often use multiple sites spread across the country in part to reduce the network latency to users. Storage systems that duplicate data for dependability may also read the copy that is closest to the disk read head to reduce latency. Processors replicate registers in clusters of functional units to reduce their latency.
- *Prediction: Leveraging bandwidth to again help latency.* Rather than wait until the computer knows what it wants, designers are increasingly using techniques that guess in advance and continue with high performance when they guess correctly. Thus, processors predict early whether a branch is taken, and processors, caches, memory, and disk controllers prefetch data.

Note that these three approaches are not panaceas, as they all have their drawbacks. Caches can have high miss rates, keeping replicas coherent is complicated and may need a great deal of communication, and predictions require recovery mechanisms and can interfere with performance when predictions are wrong. Nevertheless, they often work.



	Processor	Memory Module	LAN	Disk
Annual Latency Improvement (last 3 milestones)	1.22	1.06	1.13	1.09
Annual Capacity Improvement (last 3 milestones)	--	1.49	--	1.37
Annual Bandwidth Improvement (last 3 milestones)	1.55	1.30	1.78	1.29
Time for Bandwidth to Double in Years	1.6	2.7	1.2	2.7
Capacity Improvement in Time for Bandwidth to Double	--	2.9	--	2.4
Latency Improvement in Time for Bandwidth to Double	1.4	1.2	1.2	1.3

Although bandwidth is improving faster more recently, the ratios of improvement for latency and capacity while bandwidth doubles are close to those in Table 2.

Table 3. Summary of annual improvement in latency, capacity, and bandwidth for the three most recent milestones in Table 1.

Optimistically speaking, these three techniques should increase in popularity to cope with performance advances that emphasize bandwidth over latency. Pessimistically speaking, such techniques are now fully deployed and it may be more difficult to find the next set of tricks to help cope. By this line of argument, the bandwidth-latency imbalance may be even more evident in the future.

Thus, in addition to coping with relatively higher latency, we should consider new ways to engineer systems with lower latency. The imbalance in these four technologies is a result in part of design decisions and engineering investments. For each of these technologies, it may be possible to improve latency at either higher costs or by lowering bandwidth a little, which we have in excess. For example, DRAM blocks and interfaces could be redesigned to emphasize latency at higher cost per Mb, and SCSI disks already demonstrate lower latency than ATA disks at higher cost per GB. The difficulty of marketing latency innovations is one of the reasons latency has received less attention thus far, and this obstacle must be addressed if we are to feature latency. Perhaps we can draw inspiration from the more mature automotive industry, which advertises time to accelerate from 0-to-60 miles per hour in addition to peak horsepower and top speed.

So What?

If everything improves at the same rate, then nothing really changes. When rates vary, we see real dislocation that in turn requires real innovation.

If you agree with this observation, how should it affect what you build? Table 1 shows performance milestones recently occurred every three or four years. As successful systems last much longer than three or four years, your design decisions should work well across several new milestones.

Let's pick a hypothetical example. Suppose you are designing a storage system that keeps multiple replicas of data at different remote sites to ensure data dependability. Caching is an obvious choice to reduce latency. Your design could take advantage of the replicas by making requests from the one that tends to be fastest or by making multiple requests to multiple copies and just using the fastest reply. Prefetching should also be considered, as the downside to prefetching will decline over time. You might have a choice between a chatty protocol that uses many small messages or one that uses few very large ones. Bandwidth advances favors the latter, even if you ultimately send more bytes. Depending on the mixture of reads and writes, you might select a log-structured file system for each remote site since its performance is tied to disk bandwidth whereas the performance traditional update-in-place file systems are tied to disk latency. Finally, you might also want to accumulate data into large block sizes, as these would be a good match to the increasing bandwidth of the memory and the disks.

Thus, new designs should consider caching, replication, and prediction, as these techniques have worked well in the past. In addition, there will be design decisions where latency advances would lean one way while bandwidth advances would lean another. I would lean toward bandwidth.

Hardware and software innovators may look across disciplines to find techniques that can turn bandwidth into useful performance, as the increasing imbalance requires invention. Since processors are often the first to face these obstacles, they have been a rich source of inspiration for others. To try to cope with further imbalance, processor designers have considered predicting the *value* of an operand before it is calculated in order to let computation proceed. Latency to memory is approaching the time to execute a thousand instructions, so we may see techniques like recalculation instead of going to memory to fetch a value. Perhaps such exotic solutions will inspire novel approaches to join caching, replication, and prediction as important techniques to help cope with the bountiful bandwidth but limited latency.

Conclusion

We have seen rapid performance advances in processors, memories, networks, and disks since 1980. Figure 1 shows that bandwidth has improved much more than latency. This article discusses six reasons why this imbalance occurs, and notes that caching, replication, and prediction help cope with it.

From a system's perspective, you can argue that Figure 1 understates the case, as using multiple components multiplies bandwidth without helping latency. For example, there are greater bandwidth gains using multiple disks in a disk array, simultaneous communication in a switched network, multiple memory modules in a large memory, or multiple processors in a cluster or shared memory multiprocessor.

This article offers a rule of thumb to set expectations of products in four disparate technologies: in the time that bandwidth doubles, latency improves by no more than a factor of 1.2 to 1.4; stated more simply, bandwidth grows by at least the square of the improvement in latency. I expect this ratio of bandwidth-latency advances will hold for the foreseeable future. Hardware and software developers should plan accordingly. C

References

1. Gries, M. A survey of synchronous RAM architectures. *Computer Engineering and Networks Laboratory* (TIK). Zurich, Germany, (Apr. 1999).
2. Grochowski, E. and Halem, R. Technological impact of magnetic hard disk drives on storage systems. *IBM Systems J. 42,* 2 (July 2003), 338-346.
3. Hennessy, J. and Patterson, D. *Computer Architecture: A Quantitative Approach*. Morgan Kauffman, San Francisco, CA, 1990, 1996, 2003. (Most of the historical data in Table 1 comes from the three editions of this book.)
4. IC Knowledge. *History of the Integrated Circuit;* www.icknowledge.com/history/history.html (2003)
5. Patterson, D. and Hennessy, J. *Computer Organization and Design: The Hardware/Software Interface.* Morgan Kauffman San Francisco, CA, 1994, 1998, 2004. (Some historical data in Table 1 comes from the three editions of this book.)
6. Ross, P. 5 commandments of engineering. *IEEE Spectrum* (Dec. 2003).

David A. Patterson (patterson@cs.berkeley.edu) is the Pardee Professor of Computer Science at the University of California at Berkeley. He is also the president of ACM.