

Strategic Directions in Computer Architecture

TREVOR MUDGE¹

Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI, 48109-2122 <tnm@eecs.umich.edu>; <http://www.eecs.umich.edu/~utnm/>

1. INTRODUCTION

Looking back on the last 30 years, we have seen the remarkable developments in semiconductor technology enabling the implementation of ideas that were previously beyond the computer architect's grasp. Ideas like multilevel memory hierarchies, pipelining, multiple instruction issue, and speculative execution are just a few examples of architectural innovations that have become commonplace in high-end computers. We expect that processor performances will continue to double every two to three years, memory densities (DRAM) will continue to quadruple every three years, and disk densities will also quadruple every three years. We do not see any decrease in this rate of progress for the next decade—the time frame we have taken as “strategic.” However, beyond that we expect semiconductor technology, the driving force behind these de-

velopments, to reach physical limits, at which point we may start to see computers based on different primitives. For example, the stochastic nature of the gates may become important.

Several important books have appeared in the past few years that are pertinent to this discussion because they have placed the study of computer architecture on a quantitative footing. Two of the most notable, those by Hennessey and Patterson [1996] and Flynn [1995], are recommended to the reader who wishes to get a deeper understanding of the subject and who may also wish to find out more about some of the ideas that space allows only brief mention of in this discussion. A recent workshop sponsored by the NSF on Critical Issues in Computer Architecture Research is also relevant [NSF 1996].

What Is Computer Architecture?

The term “computer architecture” was first used by IBM to describe the programming interface to their System 360 series of computers, specifically the instructions, registers, and flags seen by the assembly programmer. This interface has become known as the instruction set architecture (ISA) of a computer. We take a broader view for the purposes of this article that includes not only the instruction set but its implementation with hardware components such as memories, registers, buses, and arithmetic units. A specific computer

¹The author would like to express his deep thanks to the members of the Computer Architecture Group for their assistance and support, as well as for permission to borrow from their position statements. Any errors or omissions are entirely the responsibility of the author.

Group members include Arvind, MIT; Tom Conte, North Carolina State University; Joel Emer, DEC; Matt Farrens, University of California, Davis; Dirk Grunwald, University of Colorado; Kai Li, Princeton University; Jack Mills, Intel Corp.; David Nagle, Carnegie Mellon University; Yale Patt, University of Michigan; Jim Smith, University of Wisconsin; Wen-Hann Wang, Intel Corp.; David Wood, University of Wisconsin; and Robert Yung, Sun Microsystems.

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.

© 1996 ACM 0360-0300/96/1200-0671 \$03.50

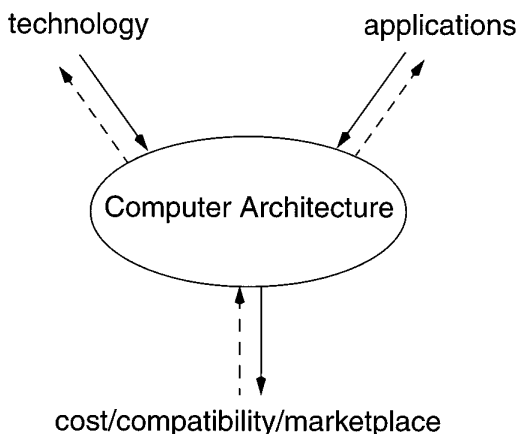


Figure 1. Computer architecture.

architecture is usually created to run a particular set of applications, or workload, at some performance level cost-effectively. Figure 1 illustrates the interaction of cost, applications, and architecture. Of course, the problem is more complex than the figure depicts; in particular, the ability to run particular applications often constrains the ISA through the need to use legacy code (the Intel ISA being the most obvious example) and the availability of applications is heavily influenced by the marketplace. One overriding factor should be clear: computer architecture is to a significant degree an engineering activity of providing a solution to a (computing) problem within a cost constraint.

2. TRENDS

Trends in computer architecture fall into two classes: those that result from the progress in the implementation technologies, and those that result from popular applications.

Technology-Driven Trends

As already asserted, the computer “revolution” has been driven by the remarkable growth in semiconductor technologies. The common denominator has been the constant reduction in the size of electronic and magnetic devices that

can be manufactured inexpensively. If we take our time horizon to be the next decade as an appropriate strategic point in the future, we expect to see feature sizes of 0.05 microns and chips with 100 million devices operating at several gigahertz. Technical obstacles need to be overcome for this progress to occur, but it is our expectation that the overwhelming resources in industry and academia that are deployed in the computer industry will solve them.

The two major obstacles that we see are the issues of power and memory latency. Low-power design is important because portable computing will become ubiquitous in the next five years. Even in the realm of high-performance computing, power issues are important because power consumption is linearly related to clock frequency. The common figure of doubling clock rates every two years thus corresponds to doubling power consumption every two years. This is not sustainable from a power perspective. The solution will require solutions that combine circuit and device technology advances with complementary developments in computer architecture.

The second obstacle is memory latency. Intrachip latencies and bandwidths are decreasing and increasing, respectively, at much greater rates than the corresponding interchip figures. In conventional processor and memory designs, this trend leads to substantial increases in memory latency relative to instruction execution, and corresponding decreases in bandwidth. New techniques that address this fundamental problem are needed. A concrete example of this can be seen with DRAMs. While the CPU speed curve has increased at about 50% per year, DRAM speed has improved only at a rate of about 10% per year over the last decade—DRAM improvement has focused on density which, as noted earlier, has quadrupled every three years. The cost of a DRAM access is equivalent to over a hundred CPU instructions today instead of the few instructions that it was only a de-

cade ago. Furthermore, disk access time improvement is at an even lower rate than the DRAM's. To bridge the performance gap of the storage hierarchy, architects will have to understand better how to design write-buffers, caches, memory systems, disk I/O subsystems, and interconnects.

Application-Driven Trends

We expect to see a blossoming of new applications, particularly those that make computers easier to use and those that allow computers to communicate with one another. They have already been widely discussed in the trade and popular press in the context of multimedia, virtual reality, and the Internet. Examples will include:

- web servers that can provide HTML service,
- network computers to replace the low cost X-terminal while giving the user web access,
- multimedia databases (probably object-oriented) capable of handling complex queries,
- virtual reality with 3D motion graphics, and
- network routers and low-latency, low-overhead network interfaces.

It is interesting to observe how these kinds of applications compare to those that still dominate much of academic research, which has had a heavy bias towards scientific and engineering workloads. This is not surprising given that early computer developers were from the scientific and military communities. Indeed, ENIAC, one of the earliest computers, was built to compute ballistic trajectories. In the intervening years the use of computers has shifted into nonscientific areas. This movement has been dramatic in the microprocessor/personal computer era. The thinking in academia is still heavily influenced by this early history and has led to the development of research areas that may not be the most relevant point of depar-

ture for future work in computer architecture.

3. CHALLENGES

If we combine the trends in applications with the trends in technology over the next decade, several challenges emerge:

- (1) *Power and size.* Are there architectural strategies for reducing power consumption or is it purely a circuit-design/technology issue?
- (2) *Performance.* Will the improvement due to technology of 50% per year satisfy the rate of growth that we expect (as has been true in the past) in application computing requirements, or will architecture play a stronger role than in the past? An obvious candidate is the introduction of parallelism.
- (3) *Complexity.* How can we cope with design complexity, design time, and correctness (testing, fault tolerance)? This activity consumes nearly 40% of the development costs of today's high-end microprocessors. Perhaps this is a subject that more correctly belongs in the ECAD area; however, there may be architectural styles that simplify validation.

4. FUTURE RESEARCH DIRECTIONS

We believe that research in computer architecture research should span a spectrum of activities from the incremental to the revolutionary. Furthermore, it is important for the computer architecture community to pursue projects throughout this spectrum, because experimentation offers the surest way to identify the important issues. The group had somewhat differing views on how experimental architectural research can best be carried out. Some regarded prototype construction as essential, whereas others felt that simulation was sufficient and more cost-effective. After all, it was noted that modern passenger jets (the Boeing 777) have been designed and *flown* com-

pletely in simulation; why cannot the same approach be taken for computers? Indeed, this paradigm from the aerospace world prompted one group of researchers at the University of Wisconsin to call their experimental testbed the Wisconsin Wind Tunnel. A counterargument to pure simulation is that the science of flight is better understood than the science of computers. However, as our understanding increases, it is likely that hardware prototyping will be slowly replaced by simulation activity.

Most of the methods now being used in high-end microprocessors were originally proposed many years ago. And although impressive gains have been made since the late 1970s, many of these gains have come from adopting innovations that were proposed for, and used in, large systems many years before. Today, microprocessors have “caught up” with most innovations of mainframes and supercomputers, and the leading edge of innovation can now be found in microprocessor designs. A question that immediately comes to mind is: what fundamentally new techniques of the same importance as pipelining, superscalar, and very long instruction word (VLIW) are being proposed today for use in processors 10 years from now? For the most part, researchers investigating high-performance processors are looking at evolutionary variations on already known techniques for increasing instruction-level parallelism, for example, ways to increase superscalar issue by some small factor, more elaborate branch prediction methods, ways of increasing cache hit rates by more sophisticated buffering methods, more elaborate cache coherence methods, and so on. These kinds of incremental research are important, but the research community should also be looking at bigger, more revolutionary approaches whose value will only become widely known many years from now.

This revolutionary research attempts to strike out in new directions and make fundamental changes in structures and

design assumptions. Typically this sort of research adopts a view that one of the assumptions about technological trends will no longer hold. This type of research provides insights and opportunities for creativity that were not always obvious when following the standard line of progress. The current investigations of processor-in-memory (PIM) is an example of changing assumptions: specifically, that processor and memory technologies are unsuited to the same die. Similarly, the work in multiscalar research at the University of Wisconsin was largely inspired by the opportunities that arise when multiple processors fit on a single chip.

We expect that the ideas of the future, both incremental and revolutionary, will continue to exploit the three key properties of:

- locality,
- parallelism, and
- predictability

that have been used to such great advantage in successful computer designs of the past. The following research directions will be particularly important.

Parallelism

This area of research has already been explored widely, with mixed success. The area of massively parallel processing (MPP) with machines with hundreds or thousands of processors has had the most limited success. The idea goes back to the Solomon computer of the 1960s. Notable successors have been the Illiac IV (1970s), Goodyear’s Massively Parallel Processor (1970s), the Hypercube machines of the 1980s, and Thinking Machine’s Connection Machine (1980s). These machines became progressively easier to build, particularly in the 1980s when high-performance commodity microprocessors appeared. However, difficulty of programming presented a formidable barrier to the widespread use of massively parallel processors. They have found an important niche for very

high-performance applications where machines and a team of specialist programmers can be dedicated to the application.

Networks of workstations (NOWs) appear to offer more promising avenues for MPP research. They have the advantage that they exist primarily to satisfy another established need. However, research into networks of workstations, like that into MPPs, does not address the fundamental question of whether there is a high degree of parallelism in enough application domains to support widespread use of massive parallelism.

Perhaps one of the most promising areas of future research in parallel processing will be in small-scale shared-memory multiprocessors (MPs) with anywhere from 2 to 100 processors. Their widespread use as servers means that they already have an established market. Increasing their application base to include parallel processing on single tasks is the key to enabling their wider acceptance. In common with MPPs, it is primarily a software problem. However, the presence of a shared memory greatly simplifies the barrier to programming.

The possibility of extending the shared-memory paradigm has led to research into NUMA (nonuniform memory access) architectures that interconnect groups of MPs into larger systems. The work at Stanford on the DASH and more recently the FLASH multiprocessors is exploring such systems. Indeed, the researchers on NOWs and NUMA MPs see an opportunity for producing a unified parallel processing API (application programmer interface).

Another promising area of future research in parallel processing is simultaneous multithreading (SMT). The goal here is to interleave concurrent threads of execution through a processor that is capable of retaining copies of each context of the respective threads. The idea goes back to the HEP computer developed by Denelcor in the late 1970s and early 1980s. It has renewed potential because many common applications are

being written to support multithreading.

Of course, the most successful application of parallel processing has been at the instruction level. Most high-performance processors today employ some level of instruction-level processing; notable examples are the Pentium Pro, the DEC Alpha, and the MIPS R10000. These machines are pipelined superscalar architectures and the detection of parallelism is done by the hardware and so they are able to execute existing binaries. This line of research is more evolutionary than revolutionary, but because of its importance we can expect considerable resources will be expended to increase the degree of instruction-level parallelism from the current figure of about four instructions at a time to 3–4 times that number. To reach these levels compiler support will be needed. Thus computer architecture will start to encompass compiler technology too. Indeed, it has been a feature of a significant amount of architecture research for some time.

A more revolutionary approach to instruction-level parallelism are the VLIW architectures. Several commercial machines have been produced in the past by Cydrome and Multiflow. They were not commercial successes. One drawback is that this class of machines requires extensive compiler support and does not easily support legacy code; however, considerable work is focusing on them and these difficulties may very well be solved.

Memory Hierarchy Design

We noted earlier that intrachip latencies and bandwidths are decreasing and increasing, respectively, at much greater rates than the corresponding interchip figures. Consequently the architecture of memory hierarchies is likely to become more important than processor architecture. Some of the problems are going to be partially solved by developments in architecture coupled with developments in technology.

Technological solutions are beginning to appear, particularly among DRAM manufacturers who, in the past few years, have introduced the synchronous DRAM, the RAMBUS, and a variety of other variations on DRAMs that are capable of gigabyte/second data-transfer rates when consecutive memory references are made. The internal organization of DRAMs is highly suitable for streaming data transfers, and it is not costly to change the memory chip's system interface to permit such transfers. In applications where video streams are called for, these memories have already proved critical. It remains to be seen how much architects can use this new high-bandwidth capability to reduce memory latency in more general purpose settings.

A key technique that is emerging in memory hierarchy design is prefetching. In order to use bandwidth to mask latency, some method of fetch prediction is necessary between the memory levels. The idea of using prediction in computer architecture is not new; it has already found important application in branch prediction and speculative execution in general. We expect that prefetching research will continue to grow in importance.

New Benchmarks

Given what was said about the nature of the applications in the next decade, it is probably not surprising that we believe that the benchmark programs used for evaluating new computer designs should be changed. We have become complacent in our study of the specmarks and large scientific problems. The computer architecture community has almost completely ignored the increasing importance of multimedia processing, including speech, voice recognition, and video encode and decode. It should be an embarrassment to the research community that the addition of multimedia instructions has been, to our knowledge, an entirely industrial action, with no published eval-

uation of architectural or algorithmic alternatives by the computer architecture research community.

It is already clear that the behavioral characteristics of large databases, heavily loaded Web servers, object-oriented languages, interpreters, for example, Java, and instruction set emulators are all significantly different from the standard computer architecture paper benchmarks. Examination of the characteristics of these workloads, along with methods for looking at them should be a fertile area for new architectural innovation.

REFERENCES

- FLYNN, M. J. 1995. *Computer Architecture*. Jones and Bartlett, Boston, MA.
- HENNESSY, J., AND PATTERSON, D. 1996. *Computer Architecture: A Quantitative Approach (2nd ed.)*, Morgan-Kaufmann, San Mateo, CA.
- NSF 1996. *NSF Workshop on Critical Issues in Computer Architecture Research*, Sponsored by the Microelectronic Systems Architecture Program of the Division of Microelectronic Information Processing Systems at NSF (May 21). <http://www.cise.nsf.gov/mips/MSAWorkshop96/index2.html>

BIBLIOGRAPHY

General and Historical

- AGERWALA, T., AND COCKE, J. 1987. High performance reduced instruction set processors. IBM Tech. Rep. (March).
- AMDAHL, G. M., BLAAUW, G. A., AND BROOKS, F. P., JR. 1964. Architecture of the IBM System 360. *IBM J. Res. Dev.* 8, 2 (April), 87–101.
- ATANASOFF, J. V. 1940. Computing machine for the solution of large systems of linear equations. Internal Rep., Iowa State University, Ames.
- BHANDARKAR, D., AND CLARK, D. W. 1991. Performance from architecture: Comparing a RISC and a CISC with similar hardware organizations. In *Proceedings of the Fourth Conference on Architectural Support for Programming Languages and Operating Systems*, (Palo Alto, CA, April) IEEE/ACM, 310–319.
- BELL, G., CADY, R., MCFARLAND, H., DELAGI, B., O'LAUGHLIN, J., NOONAN, R., AND WULF, W. 1970. A new architecture for mini-computers: The DEC PDP-11. In *Proceedings of AFIPS SJCC*, 657–675.

- BUCHOLTZ, W. 1962. *Planning a Computer System: Project Stretch*. McGraw-Hill, New York.
- BURKS, A. W., GOLDSTINE, H. H., AND VON NEUMANN, J. 1946. Preliminary discussion of the logical design of an electronic computing instrument. Rep. to the US Army Ordnance Department, p. 1; also appears in *Papers of John von Neumann*, W. Aspray and A. Burks, Eds., MIT Press, Cambridge, MA, and Tomash Publishers, Los Angeles, 1987, 97–146.
- DITZEL, D. R., AND CLARK, D. W. 1980. Retrospective on high-level language computer architecture. In *Proceedings of the Seventh Annual Symposium on Computer Architecture* (La Baule, France, June), 97–104.
- GOLDSTINE, H. H. 1972. *The Computer: From Pascal to von Neumann*. Princeton University Press, Princeton, NJ.
- HAUCK, E. A., AND DENT, B. A. 1968. Burroughs B6500-B7500 stack mechanism. In *Proceedings of AFIPS SJCC*, 245–251.
- HENNESSY, J. 1984. VLSI processor architecture. *IEEE Trans. Comput. C-33*, 11 (Dec.), 1221–1246.
- MENABREA, L. F. 1842. *Sketch of The Analytical Engine Invented by Charles Babbage*. Bibliotheque Universelle de Geneve (Oct.).
- PATTERSON, D. 1985. Reduced instruction set computers. *Commun. ACM* 28, 1 (Jan.), 8–21.
- THORNTON, J. E. 1964. Parallel operation in Control Data 6600. In *Proceedings of the AFIPS Fall Joint Computer Conference 26, part 2*, 33–40.
- TOUMA, W. R. 1993. *The Dynamics of the Computer Industry: Modeling the Supply of Workstations and Their Components*. Kluwer Academic, Boston.
- UPTON, M., HUFF, T., MUDGE, T., AND BROWN, R. 1994. Resource allocation in a high clock rate microprocessor. In *Sixth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VI)*, 98–109.
- WILKES, M. V. 1985. *Memoirs of a Computer Pioneer*. MIT Press, Cambridge, MA.
- WILKES, M. V. 1995. *Computing Perspectives*. Morgan-Kaufmann, San Francisco.

Pipelining and Instruction-Level Parallelism

- DAVIDSON, E. S., THOMAS, A. T., SHAR, L. E., AND PATEL, J. H. 1975. Effective control for pipelined processors. In *COMPCON*, (San Francisco, March), IEEE 181–184.
- ELLIS, J. R. 1986. *Bulldog: A Compiler for VLIW Architectures*. MIT Press, Cambridge, MA.
- FISHER, J. A. 1983. Very long instruction word architectures and ELI-512. In *Proceedings of*

the Tenth Symposium on Computer Architecture (Stockholm, June), 140–150.

- GOLDEN, M., AND MUDGE, T. 1996. A comparison of two common pipeline structures. In *Institution of Electrical Engineers Proceedings - E, Computers and Digital Techniques*.
- HWU, W.-M., AND PATT, Y. 1986. HPSm, a high performance restricted data flow architecture having minimum functionality. In *Proceedings of the Thirteenth Symposium on Computer Architecture* (Tokyo, June), 297–307.
- JOHNSON, M. 1990. *Superscalar Microprocessor Design*, Prentice-Hall, Englewood Cliffs, NJ.
- JOUPPI, N. P., AND WALL, D. W. 1989. Available instruction-level parallelism for superscalar and superpipelined processors. In *Proceedings of the Third Conference on Architectural Support for Programming Languages and Operating Systems*, (Boston, April), IEEE/ACM 272–282.
- KOGGE, P. M. 1981. *The Architecture of Pipelined Computers*. McGraw-Hill, New York.
- SMITH, J. E. 1981. A study of branch prediction strategies. In *Proceedings of the Eighth Symposium on Computer Architecture* (Minneapolis, May), 135–148.
- SMITH, M. D., HOROWITZ, M., AND LAM, M. S. 1992. Efficient superscalar performance through boosting. In *Proceedings of the Fifth Conference on Architectural Support for Programming Languages and Operating Systems* (Boston, Oct.), IEEE/ACM, 248–259.
- TJADEN, G. S., AND FLYNN, M. J. 1970. Detection and parallel execution of independent instructions. *IEEE Trans. Comput. C-19*, 10 (Oct.), 889–895.
- TOMASULO, R. M. 1967. An efficient algorithm for exploiting multiple arithmetic units. *IBM J. Res. Dev.* 11, 1 (Jan.), 25–33.

Multiprocessors and Massively Parallel Processing

- ANDERSON, T. E., CULLER, D. E., AND PATTERSON, D. 1995. A case for NOW (networks of workstations). *IEEE Micro* 15, 1 (Feb.), 54–64.
- ARCHIBALD, J., AND BAER, J.-L. 1986. Cache coherence protocols: Evaluation using a multiprocessor simulation model. *ACM Trans. Comput. Syst.* 4, 4 (Nov.), 273–298.
- BOUKNIGHT, W. J., DENEBOURG, S. A., MCINTYRE, D. E., RANDALL, J. M., SAMEH, A. H., AND SLOTNICK, D. L. 1972. The Illiac IV system. *Proc. IEEE* 60, 4, 369–379. Also appears in *Computer Structures: Principles and Examples*, D. P. Siewiorek, C. G. Bell, and A. Newell, Eds., McGraw-Hill, New York (1982), 306–316.
- HAYES, J. P., AND MUDGE, T. N. 1989.

- Hypercube supercomputers. *Proc. IEEE* 77, 12 (Dec.), 1829–1841.
- HOLLAND, J. H. 1959. A universal computer capable of executing an arbitrary number of subprograms simultaneously. In *Proceedings of the East Joint Computer Conference 16*, 108–113.
- LENOSKI, D., LAUDON, J., GHARACHORLOO, K., GUPTA, A., AND HENNESSY, J. L. 1990. The Stanford DASH multiprocessor. In *Proceedings of the Seventh International Symposium on Computer Architecture* (Seattle, June), 148–159.
- LOVETT, T., AND THAKKAR, S. 1988. The Symmetry multiprocessor system. In *Proceedings of the 1988 International Conference of Parallel Processing* (University Park, PA.), 303–310.
- SCHWARTZ, J. T. 1980. Ultracomputers. *ACM Trans. Program. Lang. Syst.* 4, 2, 484–521.
- SEITZ, D. 1985. The cosmic cube. *Commun. ACM* 28, 1 (Jan.), 22–31.
- SLOTNICK, D. L., BORCK, W. C., AND MCREYNOLDS, R. C. 1962. The Solomon computer. In *Proceedings of the Fall Joint Computer Conference* (Philadelphia, Dec.), 97–107.
- SWAN, R. J., FULLER, S. H., AND SIEWIOREK, D. P. 1977. Cm*—A modular, multi-microprocessor. In *Proceedings AFIPS National Computer Conference 46*, 637–644.
- WOOD, D. A., AND HILL, M. D. 1995. Cost-effective parallel computing. *IEEE Comput.* 28, 2 (Feb.).
- CHEN, P. M., LEE, E. K., GIBSON, G. A., KATZ, R. H., AND PATTERSON, D. A. 1994. RAID: High-performance, reliable secondary storage. *ACM Comput. Surv.* 26, 2 (June), 145–188.
- HOAGLAND, A. S. 1963. *Digital Magnetic Recording*, Wiley, New York.
- HOSPODOR, A. D., AND HOAGLAND, A. S. 1993. The changing nature of disk controllers. *Proc. IEEE* 81, 4 (April), 586–594.
- JACOB, B., CHEN, P., SILVERMAN, S., AND MUDGE, T. 1996. An analytical model for designing memory hierarchies. *IEEE Trans. Comput.*
- KILBURN, T., EDWARDS, D. B. G., LANIGAN, M. J., AND SUMNER, F. H. 1962. One-level storage system. *IRE Trans. Electr. Comput. EC-11* (April), 223–235. Also appears in *Computer Structures: Principles and Examples* (1982), D. P. Siewiorek, C. G. Bell, and A. Newell, Eds., McGraw-Hill, New York, 135–148.
- OLUKOTUN, O. A., MUDGE, T. N., AND BROWN, R. B. 1992. Performance optimization of pipelined primary caches. In *Proceedings of the Nineteenth Annual International Symposium on Computer Architecture*, 181–190.
- PATTERSON, D. A., GIBSON, G. A., AND KATZ, R. H. 1987. A case for redundant arrays of inexpensive disks (RAID). Tech. Rep. UCB/CSD 87/391, Univ. of Calif. Also appeared in *ACM SIGMOD Conference Proceedings*, (Chicago, June 1–3, 1988), 109–116.
- PRZYBYLSKI, S. A. 1990. *Cache Design: A Performance-Directed Approach*. Morgan-Kaufmann Publishers, San Mateo, CA.
- SMITH, A. J. 1982. Cache memories. *Comput. Surv.* 14, 3 (Sept.), 473–530.
- SMITH, A. J. 1985. Disk cache-miss ratio analysis and design considerations. *ACM Trans. Comput. Syst.* 3, 3 (Aug.), 161–203.

Caches and Memory Systems

- BUCHER, I. V., AND HAYES, A. H. 1980. I/O performance measurement on Cray-1 and CDC 7000 computers. In *Proceedings of the Computer Performance Evaluation Users Group, 16th Meeting*, NBS 500–65, 245–254.