2. Challenges in Generating the Next Computer Generation

GORDON BELL

A leading designer of computer structures traces the evolution of past and present generations of computers and speculates about possible future courses of development for the next generations.

WHEN the Japanese told us about the Fifth Generation, we knew we were in a race. Ed Feigenbaum and Pamela McCorduck characterized it in their book, *The Fifth Generation*, but did not chart the racetrack. In the 1970s, while writing *Computer Structures* with Allen Newell, and recently in organizing the collection of The Computer Museum, I identified the pattern repeated by each generation. In order to win, or even to be in the race, knowing the course is essential.

The Cycle of Generations

A computer generation is produced when a specific cycle of events occurs. The cycle proceeds when

- motivation (e.g. the threat of industrial annihilation) frees resources;

- technology and science provide ideas for building new machines;
- organizations arise to build the new computing structures; and (after the fact)
- use of the resulting products confirm a generation's existence.

A generation results from at least two trips around this cycle, each succeeding trip an accelerated version of the previous one, as in a cyclotron. A newly perceived need (e.g. the "need" for personal computing) injects into this figurative accelerator the intention to "build a machine." New technology is then applied (e.g. in the case of the personal computer: the first microprocessors, higher-density memory and floppy disks), followed first by new contributions in system architecture and design, then by actual construction and manufacture. System software development further boosts the whirling "particle," and experimental use with relevant algorithms (e.g. Visicalc) completes its first cycle. Before it goes around again, critical evaluation is crucial. The particle is then accelerated again, until it attains the energy level necessary for use in practical applications and in industry.

In the past, two trips around this cycle tended to produce a generation of computers. The first trip generally resulted in a new structure, the second in a product with market acceptance. The personal computer (PC) followed something like this pattern, but required three cycles to reach the high energy level characteristic of a generation. The very first PC, the LINC, now in The Computer Museum, was built in 1962 and cost about $40K—the price of current engineering workstations. The PC concept was not viable from a market perspective until 1975, when the 4K memory chip and reasonably powerful microprocessors became available. The Apple II (circa 1978) using the 16K memory chip, and the IBM PC (circa 1981) using the 64K chip comprise the second and third trips around the accelerator.

Today one subgenerational cycle takes about 3 years to complete—the time it takes to develop a new technology—

with both industry and academe providing the push.

The Japanese Approach to the Next Generation

> "If a computer understands English, it must be Japanese." (A pearl from Alan Perlis, speaking at The Computer Museum, September 1983)

The Japanese Fifth Generation plan, formulated in 1980, is based on worldwide research. Because the Japanese understand large-scale, long-term interactive processes, this research effort appears to be 3 to 5 years ahead of any such American effort.

The Japanese evolutionary approach to engineering and their leverage of the world's research have both been impressive. For example, they have chosen to develop a practical product from the concept of the Artificial Intelligence (AI) workstation—a concept long familiar in the lab and offered by Symbolics in 1980. They began with a Digital Equipment Corporation DECsystem 20 and are working to produce workstation hardware capable of executing Lisp and Prolog, 10 and 20 times faster than the DECsystem 20. In the process they hope to develop significant applications, which will allow them to use the workstation, evaluate it, and whirl around the cycle again at a higher performance level. Thus working with the natural pattern—starting with the criterion of usefulness, rather than devising arbitrary, revolutionary, and perhaps useless architectures—they envision completing two more cycles by 1990. Evolution has allowed them to gain supremacy in semiconductors and even supercomputers.

The United States Approach to the Next Generation

Because our efforts in the United States have until now involved a multiplicity of independent, uncoordinated inventions (many of them games), the Japanese may already have won the race to produce the next computer generation. As a guerilla army, so to speak, we have been drawn into the contest, lacking any notion of how this game should be played, with what combination of resources, and whether by fielding individuals or teams.

Consider, for example, the suggestions by MIT's Mike Dertouzos of ways we might still win in this competition with Japan:

1. Spend $100–200M to develop high-speed computers with AI functions.
2. Encourage increased openness toward foreign workers in U.S. industry and academia.
3. Provide tax credits for long-range investments that support a national technology policy.
4. Reexamine our antitrust policies with an eye toward permitting consortia in relevant industries.
5. Emphasize long-term research and development rather than our traditional short-term gains.

Each of Dertouzos's points raises questions:

1. Does the United States have a plan by which to coordinate the expenditure of several hundred million dollars? Our postwar university research has proceeded until now by means of small, decoupled projects. Can we quickly link such efforts up into large, interdependent, directed projects? Won't a larger budget simply raise salaries and swap a fixed set of people from place to place, because capital cannot be traded off instantaneously for labor?
2. Clearly we have been successful by free interchange of scientific knowledge. Would an open door policy toward foreign researchers and research results, although important for other reasons, necessarily increase our success in this immediate race?
3. Does the United States possess a long-range national computer technology policy that R&D tax credits might further? If not, tax credits could do little besides increase the earnings of the corporations enjoying them. Furthermore, regardless of tax considerations, few U.S. corporations are presently equipped to do research of the kind needed. Few U.S. corporate managers understand the differences between advanced product development and mere product enhancement, let alone the best techniques in basic and applied research. While U.S. managers have trouble guiding the flow of ideas through the stages of product development within a single company, the Japanese have mastered techniques for transforming worldwide research into successful processes and products.
4. Are antitrust laws really a significant issue? The long gestation times of the several consortia that have been created in the United States were not due to obstructive FTC rules.
5. Are we prepared to abandon our usual emphasis on revolutionary machines in favor of a directed, evolutionary approach like that of the Japanese? The United States has tended to fund university projects based on the commitment of a single researcher or research group to a fascinating architecture. Such projects, which are really 10-year high-risk experiments not based on the need/use cycle, are especially vulnerable to loss of interest in midstream. We must learn from the Japanese how to define, establish, and execute projects in a way that does not violate the historically proven evolution.

The fate of the Fifth Generation is already cast. Can DARPA (Defense Advanced Research Projects Agency) provide leadership through the funding of univeristy research to enable the United States to reach the next generation successfully? In the past this agency has provided a "guerilla army" of researchers with funds to pursue timesharing, computerized speech understanding, graphics, packet switching, and, most recently, Very Large Scale Integration (VLSI). VLSI technology permits large primary memories, powerful microprocessors, Local Area Networking of personal computers, and architectures involving multiple processors to enhance performance and increase fault tolerance. DARPA's focus has always been on the funding of revolutionary new machines, some of which have aided progress

toward new generations and some of which have not. Perhaps if we can clarify the successes and failures of the past we can draw useful conclusions about our potential for the future.

What Can Be Learned from Experimental Machines of the Past?

> "[Building] experimental equipment merely for demonstration of [a] principle and without inherent possibility of transformation to designs of value to others does not [facilitate good] systems engineering." (Such was Jay Forrester's opinion when he headed MIT's Project Whirlwind, which produced an experimental computer that was eventually used as the prototype for the air defense system, SAGE.)

Table 1 displays information on the lifespans and usefulness of several university-based computers from the First to Fourth Generations. Development of the first four, from the Harvard Mark I/IBM ASCC to the Whirlwind, was driven by need: These machines saw use in the context of the critical manpower shortage following World War II. The Mark I, the first modern programmable machine, was put to work computing for the Navy immediately upon completion. Columbia's SSEC, derived from the Mark I, was a landmark in the history of computing because its development resulted in IBM's entry into computer building.

The University of Pennsylvania's ENIAC, built with backing from the Department of the Army, was the next revolutionary machine. By using electronic circuits instead of relays, ENIAC provided several orders of magnitude more performance than had Mark I or the Bell Labs relay-operated machines. The concept of the electronically stored program was original with ENIAC, which then led through the University of Pennsylvania EDVAC, Princeton's Institute for Advanced Studies IAS, and the University of Illinois ILLIAC I to the establishment of the computer industry.

Whirlwind, unlike the other machines listed, was built as a prototype for production. Its simple, straightforward, fast, 16-bit word, parallel design led directly to that of the SAGE computer. The SAGE real-time, interactive vacuum tube machines ran for 25 years without failure. Whirlwind's

Table 1 — Selected U.S. University-Based Computers

Machine	First Use	Concept-Use	Project-Use	Use	Use to Total Life	Results (in addition to engineer and scientist training)
Harvard Mark I — IBM ASCC	8/44	7	5	15	.7	Use, separate data and program memories
IBM SSEC	1/48	-	2.5	4.5	.6	Use, commitment to computers
ENIAC	6/46	4	3	9	.7	Use, stored program, Electronic Computer (program and data in one memory) design for EDVAC
MIT Whirlwind	6/50	5.5	3.5	9	.6	Use, circuits, core memory, real time and interactive computation, proto for SAGE system
ILLIAC I	9/52	4	3	10	.7	Use, proto for six others
MIT/Lincoln Lab TX-0	57	3	2	8	.8	Use, transistor circuits, large core memory
ILLIAC II	6/63	5.5	3	3	.3	Asynchronous logic, design too conservative
ILLIAC IV	11/75	12	8.5-10.5	6.5	.3	Use, parallelism (algorithms), accelerate bipolar memory development, stimulated competitive approaches
CMU C.mmp	5/75	5	4.5	6	.7	Parallelism, Intel 432 proto
CMU Cm*	9/76	4	2	>6	>.6	Parallelism, multibus-type structures
U. of Texas TRAC	83	7	5	>1	>.1	—

transistorized successor for the SAGE Project, the TX-0, took about a year to design and then remained in use for over 10 years. The Digital Equipment Corporation was based on the TX-0's well-enginered state-of-the-art circuits. The LINC computer, introduced in 1962, was so simply designed that in many cases it was assembled by its final users—the first build-your-own, interactive, personal computer with keyboard, CRT, and personal LINCtape filing system. It cost about $40,000, the price of modern workstations and could be easily moved from lab to lab. It was used by individuals running their own applications. And the MIT community during the 1950s and 1960s was a hotbed of users, who put together such machines to satisfy their desires to compute.

University of Illinois Computers

Ever since ILLIAC I, built on the IAS and von Neumann architecture, the University of Illinois has been a center for the building of new machines. Work at Illinois based on the circuitry and logic of the prototype IAS machine enabled such machines to be built at six other laboratories. The long-lived ILLIAC I's made significant contributions to our understanding of software and specific computer applications.

ILLIAC II, a transistor circuit based machine, went into operation three years after significantly better machines had become available (e.g. the IBM 1401 and 7090, the CDC 160/1604, and the DEC PDP-1). Its designers, aiming to produce a very-high-performance computer yet not faced with problems of mass production, selected conservative—even obsolete—technologies (e.g. germanium instead of silicon transistors, discrete wiring instead of printed circuits). The unwieldy result did not meet expectations. Consequently, the building of experimental machines at universities was squelched for some time.

ILLIAC IV developed from the Westinghouse Solomon

Project in 1962 but was not put into service until 1975.[1] A truly revolutionary machine, it operated at 250 million operations per second, its 64 parallel processing elements controlled by a single instruction stream. Its memory hierarchy for the processing elements—1 megabyte of RAM, 2 megabytes of core memory, and 139 megabytes of storage on a fixed-head disk—clearly violated "Amdahl's constant," which suggests 1 byte of memory is needed for each instruction per second executed.

Dan Slotnik, designer of the ILLIAC IV, recently commented to me:

> Most machines come about through evolution, and that's counter to the notion of original research which is supposedly the basis of university rewards. . . . I'm convinced that universities can't and shouldn't build machines. There are too many ideas, too much democracy, and too little discipline. I used to have to stop the flow of ideas on interconnection every week when we were designing ILLIAC IV. There is also too much bureaucracy. In a state university it takes 90 days to get an IC.

Larry Roberts, who headed DARPA while ILLIAC IV was being built, claimed the machine should have been executed with TTL (transistor-transistor logic) and not ECL (emitter coupled logic) technology. "People complain bitterly," he said, "but in the end conservative technology seems to work out better." The point is that a sacrifice of processing speed with the use of conservative technology—a sacrifice in instructions per second—may sometimes pay off in months of useful machine life if it gets the machine built significantly earlier than would be the case if more revolutionary technologies were used. Taking too long to get a machine operational limits its subsequent useful life and delays what is its essential purpose—i.e. to demonstrate whether its structure will advance computing.

Applied to certain problems, the ILLIAC IV was the world's fastest machine until the Cray 1 came into production. Its major contributions, however, were by-products: its design advanced our understanding of parallelism for single instruction, multiple data machines; it demonstrated

[1] R. Michael Hord, *The Illiac IV, The First Supercomputer.* Computer Science Press, Rockville, Md. 1981.

the use of fast semiconductor memories; and it stimulated commercial production of Texas Instruments ASC, Control Data Corporation's STAR, and the Cray 1.

Carnegie-Mellon University Multiprocessors

Carnegie-Mellon University's experimental machines, designed to obtain information about parallelism, were more evolutionary than the ILLIAC IV. The useful by-products of their construction also cost less than those of the University of Illinois machine by almost two orders of magnitude.

The idea that processors can be harnessed together in parallel to form powerful composite machines is intriguing because it means that high performance might be attainable not through massive designs but through repeated use of a relatively simple design. In the multiprocessor design, multiple instruction streams operate in parallel on multiple data streams. Multiprocessors were studied at Carnegie-Mellon in the late 1960s. Bill Strecker's 1970 thesis computed the performance for p processors accessing a common memory of m modules. This seminal work on multiprocessor structure was rejected for publication at the time because of relevance and originality; but during the last 10 years dozens of theses and papers have embellished Strecker's model, all referring to his early work.

The Japanese Fifth Generation Project is predicated on successful use of multiprocessor and dataflow parallelism. One researcher at the University of Illinois recently told me he wouldn't work on a multiprocessor project involving 32 processors unless it could be extended to 1000. Yet we have no evidence to date that more than a few processors can operate effectively in parallel on a single problem! Current research focused on exotic switching structures among thousands of processors and memories distracts scientists from the more difficult job of building even a small version of such a machine, and from what may be the impossible task of using one. Using a combination of new architecture, system software, language, and algorithm design, someone must first demonstrate that we can build a useful

production machine involving 10 processors, before we extend this to a large-scale multiprocessor involving 100 or 1000 units.

C.ai and C.mmp In May of 1971 a multiprocessor comprising 16 processors was proposed as the first of the Carnegie-Mellon machines. The C.ai would have had one gigabyte of very high bandwidth memory and was intended for use in artificial intelligence research.

A much simpler design than this, using 16 Digital Equipment Corporation PDP-11 processor modules, was in design by August of the same year. Called C.mmp, this machine was used to examine the feasibility of multiprocessing and to develop a new "capability-based" operating system using a modification of the PDP-11. What was learned from the project is well documented in Professor W. A. Wulf's book, *Hydra*. The inability of this design to address all the memory available to it limited its usefulness. Furthermore, it was difficult actually to attain the maximum processing speeds theoretically possible. In order to be attactive to users a machine must offer more computational power than is easily available from other designs. By 1978 the Carnegie-Mellon University computing environment included other designs that were larger and easier to use than the C.mmp, which consequently was not used in production applications.

This project spawned the group that went on to design the Intel 432 processor. Clearly not everyone involved with C.mmp learned the lesson in limited memory addressing inherent in using the PDP-11 because the 432, too, suffered from the small address and excessive overhead that came out of the operating system approach.

*Cm** An evolutionary descendant of C.mmp, Cm* is a set of computer modules that permits construction of a medium-scale multiprocessor (50 processors) in a two-level hierarchy, making the construction of a multiprocessor of over a 100 processors possible. Cm* uses concepts from C.mmp's operating system. From the point of view of any one processor, memory is conceived as three-fold: memory

local to that processor; memory "pooled" by the cluster of 10 processors to which that processor belongs; and memory in another cluster. Any component processor in the multiprocessor computer can access any memory available to the system, but each is designed to "prefer" local to cluster memory and intra-cluster to inter-cluster memory. The Cm* is thus problem-idiosyncratic, all access times varying with whether data are in local, in the cluster, or in another cluster's memory. Cm* is thus enabling researchers to study the relationships between hardware and software structures in parallel processing. Although significant work remains before the individual processors can work harmoniously together without extensive hand tuning of programs to suit particular interactions, the evolution of Cm* from C.mmp has already paid off. Now, a machine is needed that combines all the lessons learned in C.mmp and Cm*.

The Role of Future Research at Universities

Given our need for university research in the drive toward the next computer generation, what form shall the contribution of the universities take? Shall the research be done (a) by graduate students, (b) by nonacademic professionals within the university, or (c) by joint ventures between universities and outside companies? If the latter, shall funding come from the world of venture capital or from DARPA?

(a) Graduate students provide cheap, brilliant, but unpredictable labor. Reliance on students is not to be recommended unless the machine involved can be assembled easily from well-defined industry-standard modules. A major university infrastructure will be required if experimental machines are to be designed and hardware for them is to be fabricated within our universities.

(b) The presence of nonacademic computer professionals creates a second culture inside the university. Such a twin-cultured structure, being unstable, is somewhat difficult to manage, but it is essential to building a machine within the university. Carnegie-Mellon University was successful with this approach. MIT, too, established laboratories combining academic and nonacademic professional staffs, which

produced such successful systems as Whirlwind, TX-0, TX-2, LINC, and the Multics™ timesharing system.

(c) Joint projects between universities and companies from the private sector often result in a hardware/software split, the university doing software. Digital Equipment pioneered in this form of interaction during development of timesharing on the DECsystem 10/20. For example, MIT and Stanford developed initial editors (e.g. SOS), assemblers, debugging tools (DDT) and compilers (e.g. LISP, SAIL). The most advanced joint project of this kind today is that involving Carnegie-Mellon University and IBM, who are working together on the creation of an educational computing environment in an IBM-provided industrial setting located on the CMU campus. Japanese companies build machines for their universities, especially the University of Tokyo. This arrangement has been used in developing previous generations and seems viable still.

We also see consortia forming among companies and universities for funding semiconductor facilities and for writing VLSI design software.

A related approach would use people from both academe and established industries, drawn into companies outside the university. These, funded by venture capital, would produce low-tech, low-risk products that might then fund projects to advance the state of the art. Many believe that such start-up company entrepreneurism is the way to beat the Japanese because it generates highly focused energy (one high-tech company, for example, recently developed and produced a UNIX product based on the Motorola 68000 chip in a period of 9 months); but I wonder whether the Japanese will be much daunted by our production of 123 kinds of 68000-based workstations? On the other hand, this arrangement—e.g. in the case of Amdahl's Trilogy Corporation—has funded really creative and advanced technology that no large corporation could fund because of the great risk involved.

Funding of computer science by the military often acts simply to churn a limited supply of researchers, moving them from place to place and raising salaries. The projects

involved are typically large, requiring professors to become good managers in a university environment designed to employ them as good teachers. After a few years of work as a major-project manager at the salary of a teacher and in an environment lacking adequate engineering resources a professor may become an easy target for industry recruitment. In this way industry scoops up kernals of the nation's "seed corn."

Recent DARPA-funded research has tended to produce an algorithm, a program, a chip, or a system, whose development yields enough basic knowledge and a product idea promising enough to support a start-up company. Clark's Geometry Engine, for example, forms the basis for Silicon Graphics; the Timing Verify of Widdoes and McWilliams provides the basis for Valid Logic; and the Stanford University Network workstation is the basis of SUN Microsystems.

It is finally possible for people in the computer sciences to progress rapidly through the cycle from freedom to fame and riches by identifying and solving a problem while working in a research setting and then establishing a company to exploit their discoveries by developing the products of the next computer generation. Contrary to intuition, however, I believe that a surge in research funding will only move researchers from place to place and perpetuate further the erroneous notion that more money can instantaneously buy better scientific ideas and increase the available talent.

Breakthrough into the Next Generation

Leading computer scientists have proclaimed that the next developmental cycle, the one that will produce the Next Generation, will be driven by the demand among nonprogrammers for natural language communications capabilities based on breakthroughs in artificial intelligence. Since few genuine AI applications (e.g. "expert systems") are even now in operation, however, this view seems to me to be historically improbable. Nevertheless, the popular (and funded) belief is that revolutionary structures, implemented with VLSI and ULSI technologies and predicated

on a high degree of processing parallelism, will be the key to these applications.

Past results suggest that basing the future on parallelism is risky, especially before a model of use for such systems exists as a design target. The only demonstrable parallelism today outside of vector machines (e.g. Cray 1) involves multiprocessors, which seem to be looked upon with renewed optimism in every generation. In the mid-1960s with large computers and in the mid-1970s with minicomputers, I felt that designing for multiprocessors was the best way to provide more computational power. Now, in the mid-1980s with their separate units becoming ever smaller, faster, and more powerful, use of multiprocessors must be an important way to increase performance—a fact reflected in the marketing of multiprocessors in all product ranges: supercomputers (Cray X-MP, Dennelcor), superminicomputers (ELEXSI), and microcomputers (Synapse).

With the advent of several commercial examples, it has become crucial for our universities to become involved in the use and further understanding of multiprocessors.

Why have multiprocessors not been used appreciably before now? It may be that in previous generations we have always found simpler ways, using technology or instruction set (e.g. vectors) to increase performance. Engineering may until now have been too conservative. Certainly operating systems and programming languages have failed to support or encourage multiprocessor designs. And conversely, there may have been no market for such machines because users have not been prepared to program them without language and operating support.

Human organization theory—the science of how human "processors" function (or malfunction) together—doesn't seem to contribute much to our understanding of parallelism, except anecdotally. More than a decade ago, for example, Melvin Conway speculated that people fashion computer structures to resemble the human organizations they know best. This may suggest why n individuals build n-pass compilers; why IBM builds hierarchically structured protocols like their System Network Architecture; why

DARPA maintains a store-and-forward net to have isolation between network and host computers; and why Digital builds democratic (anarchic) structures like Unibus, Ethernet, DECnet, and multiprocessors that can be employed very flexibly. Beyond such interesting notions, however, we lack an organization theory that can shed light on why it is as difficult to get more than six computer processors working together as it is to harmonize the work of six human beings, unless either group is wholly "top down directed" and given clear goals. Research should therefore concentrate for the moment on the general case of multiprocessors, because it has the greatest connectivity via primary memory. Slow or restricted networks such as Local Area Networks, trees, grids, hypercubes, etc., can be considered later, once we have understood the general case and the communications techniques it involves.

A glance at computer history reveals this pattern: New technologies lead to new computer structures (e.g. minicomputers, personal computers), which in turn create new companies. But only IBM and the large Japanese companies have so far demonstrated the basic understanding, the long-term commitment, and the marketing capabilities necessary to make the transition from one generation to the next. In both instances, however, someone else has had to establish the paths.

Where will the pathfinders arise today?