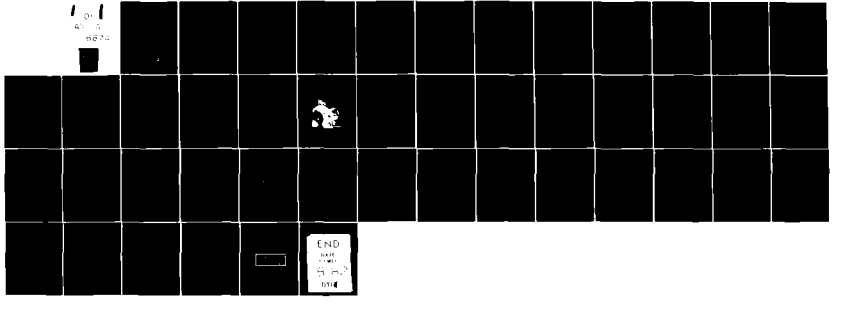
AD-A116 874 RAND CORP SANTA MONICA CA F/G 9/2
FUTURE COMPUTING,(U)
DEC 81 D LEINWEBER
UNCLASSIFIED RAND/P-6680 NL



(2)

FUTURE COMPUTING

David Leinweber

December 1981

P-6680

The Rand Corporation
Santa Monica, California 90406

I. FUTURE COMPUTING

This is a paper based on a talk given at the Litton Software Technology Management Conference held at the Adam's Mark Hotel, Houston, Texas, September 21-23, 1981. The paper takes the form of a briefing note. The author would like to thank Charlie Bridge and John Scudder of Litton Industries and Danny Hillis of MIT for encouragement and assistance.

Colleges Faltering in Effort to Ease Critical Shortage of Programmers

By George Anders
Staff Reporter of The Wall Street Journal

At some computer schools, says an educator in the field, they "don't even let employers into the classroom." The reason is that corporate recruiters many times bypass the students and try to lure the instructors.

That is just one facet of the continuing bottleneck in the booming computer industry: the shortage of qualified graduates to fill jobs. Estimates of jobs available run as high as 50,000, but the nation's colleges only graduated 11,000 people with bachelor's degrees in the field last year.

That was up 37% from a year earlier, but the Labor Department predicts that demand for programmers will double by the end of the decade. So the crisis seems likely to worsen and educational shortcomings are at the heart of it.

Expensive Machinery

New York University's computer-related adult education program is increasing enrollments by 20% to 25% a year, but, says director Stewart Fink, "We can't accommodate everyone we'd like to take." Money is a big problem. Strained college budgets won't provide for many of the $1,000 terminals and $1 million mainframe computers needed for teaching.

Many prospective programmers are finding that one of the biggest hurdles in their careers is getting educated in the first place. "I spent three weeks haggling with deans just to get into two courses I needed for my major," says a junior at the University of Michigan.

Capable instructors, naturally, are besieged with high-paying offers to work in the industry rather than teach. "We pay a lot more" to get instructors to stay, says Hyman Sharoff, president of the New York School of Computer Technology. Salaries, he says, reach $40,000 a year.

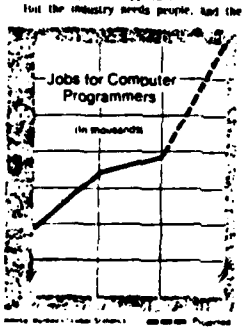
Being Helpful

In a few instances industry is helping by lending employes as instructors. At Northern Iowa Area Community College in Mason City, programmers from a nearby Sperry Corp. facility serve as adjunct instructors for evening classes. "Sperry has been very helpful to us," says Bill McKeown, the dean of curriculum.

But spokesmen at several other major computer firms said their financial support for universities is directed more at research than at teaching. The prestigious big firms like IBM, Sperry and Control Data Corp. in any case are most likely to get first crack at the available programmers.

Despite shortages, "We haven't been stripped," says a Control Data spokesman. IBM says the 600 or so programmers it hires out of school each year need and get "a lot of in-house training" to grasp the IBM system. Hewlett-Packard Co. says publicity about its need for programmers brought a *stream of unqualified applicants.*

But the industry needs people, and the



smaller firms in particular have to take what they can get. "Just about all our graduates get snapped up within four months, unless they have a personality disorder or only know English as a second language," says NYU's Mr. Fink.

Many Motivations

NYU's programming students include a number of school teachers, copywriters and "people in their late 40s planning a career switch," he adds. "In some cases, their company went bankrupt or moved, and they didn't want to move."

The better educated a programmer, the greater the demand there is for his services. "Demand probably matches production of students at the two-year level," says Robert Gillespie, vice provost for computing at the University of Washington. Students in that category can do chores like helping to write a payroll program.

"The real mismatch is at four years and above," says Mr. Gillespie. That is where graduates know the languages and computer structure to handle the nuances of, say, an airline ticketing program.

The National Center for Education Statistics says 3,587 persons earned master's degrees in computer science last year. The figure is believed to be about 10 times that figure.

Robust salaries for applicants with less education may be dampening desire for advanced degrees. At Rensselaer Polytechnic *Institute in Troy, N.Y.*, a bachelor's degree in computer sciences brings jobs paying at least $23,000, officials say.

Lure of a Job

And at Carnegie Mellon University, "We are people we know would make good Ph.D students who go off and get industry jobs," says John McDermott, assistant head of the Pittsburgh university's computer science department.

A leading aim of Carnegie Mellon's Ph.D program *is to provide the faculty to teach* the next generation of programmers, but Mr. McDermott says only a third of these students go on to faculty positions with more working for business or research institutions.

Mr. Gillespie looks to Japan and France for indications of how to remedy the U.S. programmer bottleneck. France, he says, has declared "a national commitment to put computers in high schools, and the earlier you provide them, the more effective it's going to be."

Japan began planning in 1972 for an information society, he says. Its multibillion-dollar investment plan is yielding as many engineers as the U.S. has, with half the population, he states.

Yet American institutions making the deepest commitment to computer education say they haven't been able to escape faculty shortages and crowded classrooms. Rensselaer thought it was in good shape after a major upgrading about three years ago that raised yearly spending on computer education to as much as $1,000 per graduate. (General universities may spend $200 to $300 per person.)

But this fall, Rensselaer is getting 220 freshmen who plan to major in computer science, up from 95 two years ago. Can the school comfortably handle this surge? "No," says an administrator.

Slide 1

Slide 1 Commentary: This article is typical of thousands that have appeared in the technical and popular press on the subject of computer technology in the last couple of years. It shows an almost exponential growth in the demand for computer programmers.

Others like it cite statistics such as five computers per programmer within the next 10 years. This kind of figure is virtually meaningless. When microprocessors are being installed in small consumer-oriented systems like electric razors, stoves, and blenders, it doesn't make a lot of sense to talk about the number of programmers per computer. Such figures can be likened to the extrapolation of TV sales figures in the late 50s and 60s, which, if continued, would leave us up to our knees in television sets. The real question is seen in Slide 2.

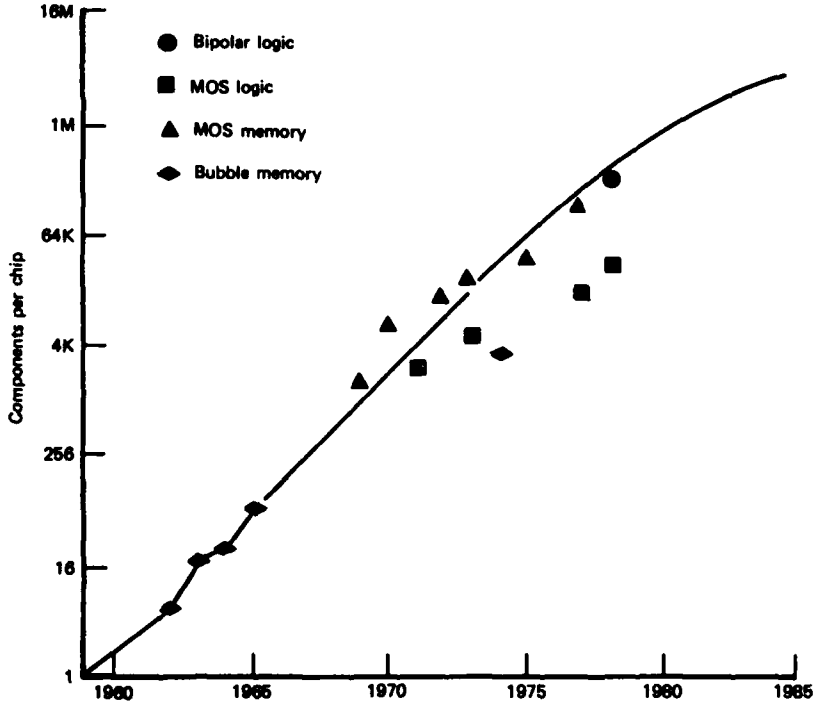
". . . I haven't the slightest idea on how to take advantage of VLSI"

Gordon Moore,
President/Founder
Intel Corporation,
at the Caltech
VLSI Conference,
January, 1979

Slide 2

Slide 2 Commentary: This quote is taken from a paper presented by Gordon Moore, President and founder of the Intel Corporation at the Cal Tech VLSI Conference in Pasadena in January 1979. It's quite a remarkable thing for him to say, having had as much responsibility as anyone for the microcircuit revolution.

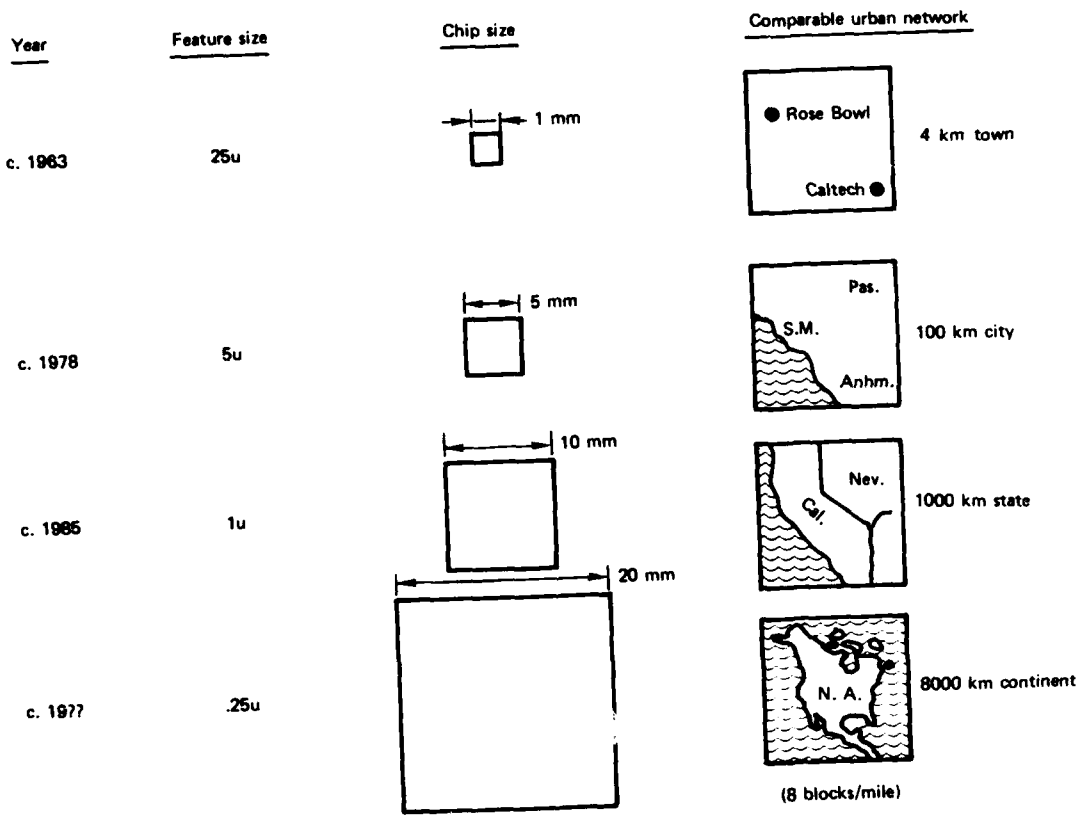
This is the central problem facing the computer industry today, and Moore had good reason to emphasize the point. In the intervening year or two, a number of developments have intervened that may have altered his perspective, but the magnitude of the adjustment that the computer industry must make, and the new economics introduced by VLSI, are still enormous. The entire cost structure underlying the use of computers is turning upside down, things have changed very fast.


16M
1M
64K
4K
256
16
1
Components per chip
Bipolar logic
MOS logic
MOS memory
Bubble memory
1960
1965
1970
1975
1980
1985

Slide 3

Slide 3 Commentary: This chart shows one of the standard ways of presenting the increase in complexity we have seen (and expect to continue to see) in microelectronic components in the period between 1960 and 1985. An increase in complexity by a factor of 16 million or more is forecast, with costs going down as rapidly as the number of components go up. While doubts have been raised regarding the ability of the semiconductor industry to implement devices with this complexity, I do not believe that these will be found to be warranted. The development of advanced CAD/CAM (computer-aided design and manufacturing) systems for the production of VLSI circuits is bringing devices with the complexity shown even at the far edges of this chart within reach. These "silicon compilers" have the ability to generate VLSI layouts compatible with extensive design rules and restrictions, and do it in a manner that has even enabled first-time student users to successfully implement their own VLSI systems. This technology has progressed to the point where advanced undergraduate students can implement their projects on remote computers, transmit the files over a network for evaluation, and when they are satisfied, transmit these files again over a network for mass generation and fabrication and receive test quantities of the VLSI components within a matter of weeks or days. The devices built in this way are really integrated systems, not integrated circuits. They include a high-level language (LISP) microprocessor, graphic manipulation devices, virtual memory pagers, database subsystems, and a number of signal processing applications.

Remarkably, most of these components work as designed. The errors found are not due to the silicon compilers but are rather of the logical sort at the highest level of abstraction that were made at the time of the initial design.



Microcircuit complexity explosion

Slide 4

Slide 4 Commentary: Here's a different way of looking at the microcircuit complexity explosion that brings the point closer to home than the previous chart. A factor of 16 million is difficult to appreciate. We see very little in our lifetime that approaches this degree of change. You could find a way of comparing the altitude and speed of hot air balloons with the space shuttle that would get you up close to it, but that change took a much longer time. One can appreciate the consequences of the reduction in feature size in integrated circuits and the simultaneous increase in the maximum chip size by drawing an analogy with a dense urban network, as seen in our larger cities. Usually there are eight city blocks per mile. We can ask the question, "What is the equivalent size of an urban network comparable in complexity to the chips we will soon be producing?" Early integrated circuits, circa 1963, with 25 micron features on 1 millimeter squares were comparable to a small town, 4 kilometers on a side, such as Pasadena. Each of the "streets" is 25 microns wide. In 1978, 5 micron features on 5 millimeter chips had a complexity comparable to the Los Angeles area. The 1985, 1 micron devices will be equivalent to an urban network covering all of California and Nevada, parts of Arizona and a good deal of the Pacific coastal waters. By the end of the century, with 0.25 micron devices, 20 millimeters on a side, we will be dealing with devices having the complexity of a dense urban network 8000 kilometers on a side, larger than the entire North American continent.

We have no experience in dealing with this sort of complexity. It is not appropriate to call such a device an integrated circuit. This will be a truly integrated system consisting of components previously thought to be of a cost and size which would prohibit their inclusion in small, inexpensive packages. Designers of the future will be able to assemble these systems, consisting of banks of communicating high-speed computers running a combination of new and existing software, choosing the components in much the same way that today's designers choose power supplies, I/O interfaces, microprocessors or smaller-scale logic elements.

This example is originally due to Carver Mead of the California Institute of Technology.

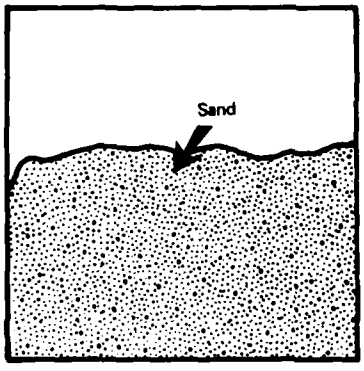
Assets

- VLSI
- Existing hardware
- New programmers/software
- Existing software
- 30 years of experience

Slide 5

Slide 5 Commentary: What assets do we have to cope with the coming changes in the computer industry? These are seen in the chart above. First is the inexpensive new VLSI hardware, which brought on the problem in the first place. Second is existing hardware. This is nothing new. Computers have been instrumental in the design of computers for years. We should not expect to see any change in this. The third asset we bring to the problem is new programmers and new software. This, as we have noted, is a particularly sore point. Another asset we bring to address this issue is the existing software base, developed at great cost over many years. Finally, we have the 30 years of experience we have gained in the development and use of computer systems. But there are strong new economic forces which influence the combination of these assets that the industry will ultimately choose.

Future computers



- Raw material: sand
- Production rate: thousands/day
- Delivery: immediate
- Price minimal
- O & M costs: very low

Slide 6

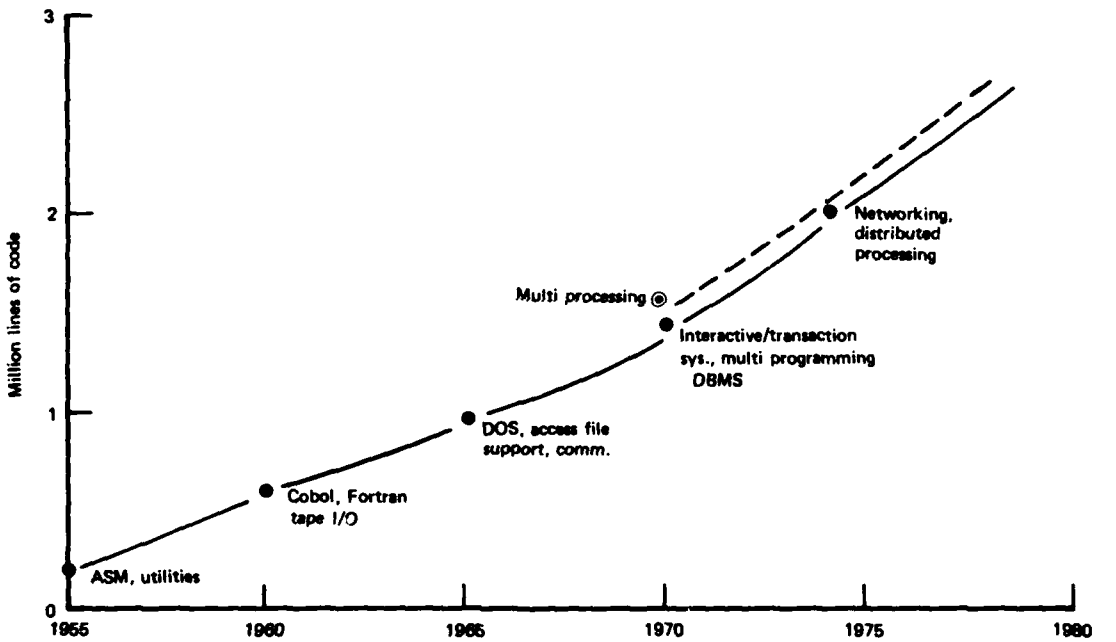
Slide 6 Commentary: Slide 6 shows the raw material for future computers: sand. Thousands of computers can be produced in an hour at minimal cost. Once the computers are installed, the operating and maintenance costs are relatively low.

Future programmers

Slide 7

Slide 7 Commentary: On the other hand, this chart shows the raw material for future computer programmers. There is a minimum nine-month delivery time and then it takes 20 years to get them working. While the buy-in price is relatively low, the operating and maintenance expenses are staggering.

O.S. software evolution

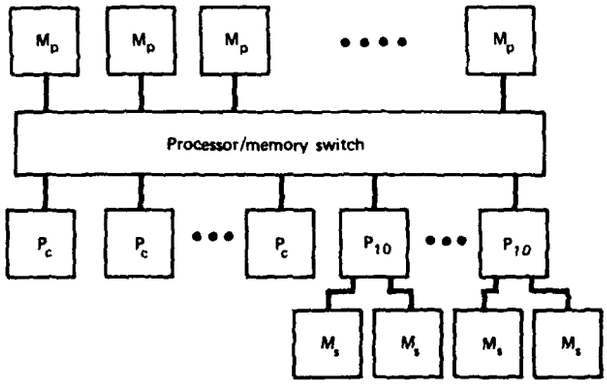


Slide 8

Slide 8 Commentary: This chart traces the operating system software evolution for the last 25 years. It shows the growth in complexity of software that has been developed to support the growing computing capability. We can already see many parallels in the development of microcomputer software. Growing from simple assemblers and primitive I/O utilities, systems have evolved to include high-level languages such as COBOL and FORTRAN in the first evolution (and PASCAL and BASIC in the second), and later, software for disk file access, programmer support, and communication. We can see that this path has been followed again by commercial microcomputer products. The rapidity of that evolution is in part due to the freedom from the complication of attempting to share resources between a number of users. When you have a $2 million system, it is worth it to spend $200,000 to share it among several people. A $2000 system, on the other hand, is better replicated. What this does as a side benefit for the microsystem user is remove all the interuser overhead that we are used to on multiuser time-sharing systems, providing even more computing power to the microcomputer user than one might expect based on the processor speed and memory size alone. Continuing along the curve in Slide 8, we see that the past evolution of computing systems has emphasized multiprogramming. Multiprogramming was especially difficult to develop. It took a longer time than anyone had anticipated and there were more ignominious failures than had been anticipated. We all know the stories about buried tapes, and once such an investment was made it

was widely promulgated and was in part responsible for the remarkable spread of computing in the late 60s and early 70s. The VLSI revolution undercuts the basic economics which led to the development of multiprogramming. It drives us more in the direction of multiprocessing--the use of several processors to perform tasks rather than the sharing of one processor. In an important way, multiprocessing also allows us to take advantage of existing software if we can use a processor that will emulate the appropriate host machine.

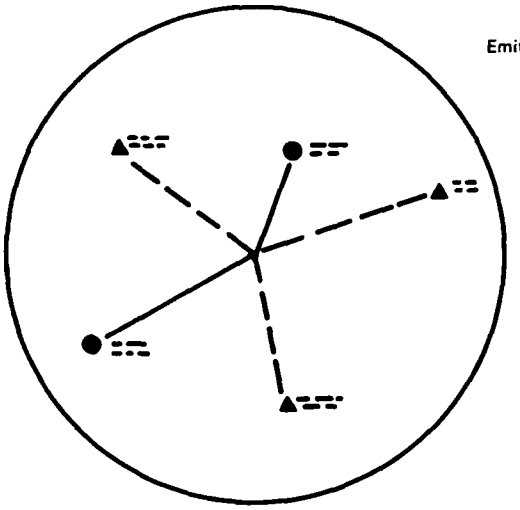
Multiprocessor architecture



Slide 9

Slide 9 Commentary: This chart shows the architecture of a general multiprocessor. A set of primary memories labeled M_p are connected through a switch to a set of central and I/O processors that are P_c and P_{IO}. The I/O processors can be connected to secondary memories or devices denoted M_s. This type of architecture is seen commercially in machines like the TANDEM-16 or in research machines such as the Cm* or the C.mmp. While structuring a process for execution in this kind of an architecture is not fully understood, people have been doing it anyway, in the spirit of those who threw rocks long before understanding ballistics. The processors in the multiprocessor need not be identical. In many cases, if maximum advantage is to be taken from the existing body of software, one would not desire the processors to be the same, rather optimized to execute the code most appropriate to the task at hand.

Current ECM application

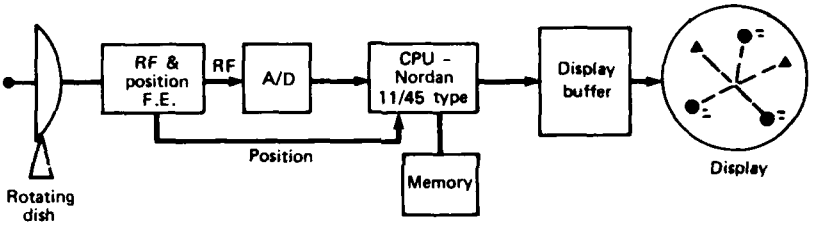


Emitter locator —
10 MHz bandwidth
up to 100 emitters

Slide 10

Slide 10 Commentary: This chart illustrates a typical electronic countermeasures application as can be found in the AWACS or any number of electronic warfare systems. This is an emitter locator, which tracks the sources and intensities of radio transmitters in its field of view. As an illustrative specification, consider a system with a 10 MHz bandwidth and the ability to track up to 100 emitters.

Current implementation



10 MHz, 100 source E.L.S.

Slide 11

Slide 11 Commentary: A simplified representation of the implementation for this kind of system is seen in tnis chart. A rotating dish antenna would be connected to a radio frequency front-end and antenna-positioning system. The RF information derived from the antenna would be sent through an analog-to-digital converter and processed by a CPU, perhaps a militarized 11/45 type or equivalent. The processor would also receive position information from the antenna front end so it would know where the signals it was dealing with were emitted. The processor would generate data for a display buffer, which would be shown to the operator and relayed through appropriate interfaces to other ECM subsystems.

Hypothetical new system requirement:

- 100 MHz bandwidth
- 2000 source capacity

Design alternatives:

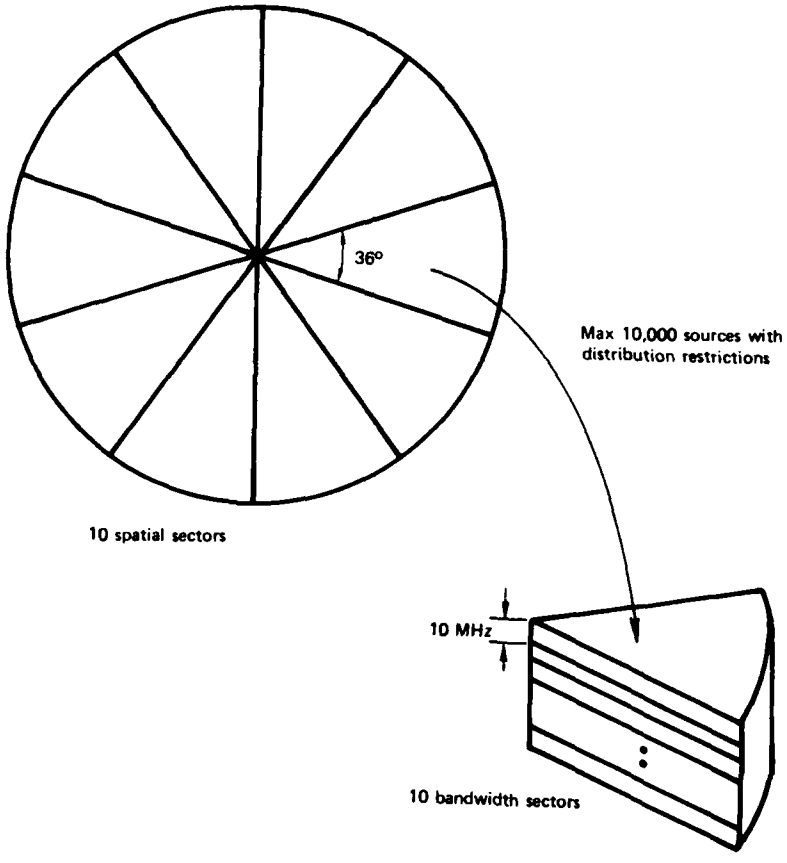
- Re-implementation with higher speed processor
- Multiprocessor with new code
- Multiprocessor with emulation

Slide 12

Slide 12 Commentary: An engineer of the future might be faced with the requirements shown in this chart. The old ECM system has become the export model and the hypothetical new requirement for U.S. forces would be an emitter locator system with a 100 MHz bandwidth and a 2000 source capacity. These are performance increases over the previous system by factors of 10 and 20 respectively.

The design alternatives facing this engineer would be reimplementation on a single high-speed processor large enough and fast enough to handle the task, implementation on a multiprocessor architecture with new code, or the use of a multiprocessor architecture emulating the CPU in the previous system.

Partition of new ECM task for multiprocessing

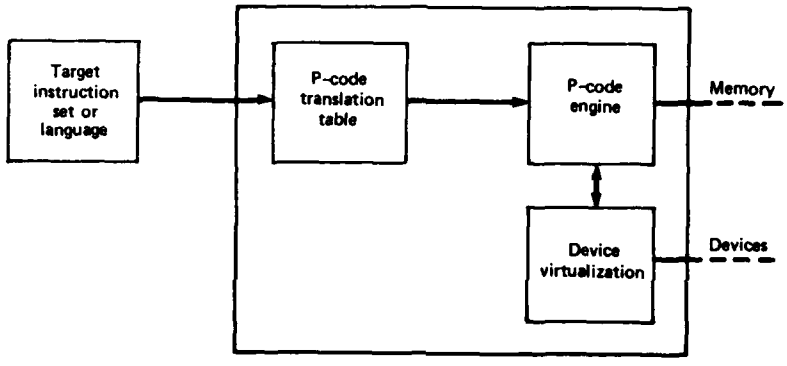


Slide 13

Slide 13 Commentary: The first step in any multiprocessor implementation is the partitioning of the task. This can be done on functional lines, by the creation of similar subtasks, or both. For this particular example, a useful initial partition is shown in this chart, in which the area of coverage for the emitter locator system has been divided into 10 spatial sectors, each 36 degrees wide. Each of these wedges is then divided into 10 bandwidth sectors, each 10 MHz wide. If each sector is serviced by a processor emulating the design original, then this system will provide a 100 MHz bandwidth and allow a maximum of 10,000 sources to be observed, provided no more than 100 are found in any particular sector. Quite a wide range of spatial and frequency distributions would be admissible and still allow the system to track the 2000 sources specified in the design requirement.

In the next three charts we will examine at different levels of detail the implementation of the new ECM system in a multiprocessor emulator system.

VLSI emulator

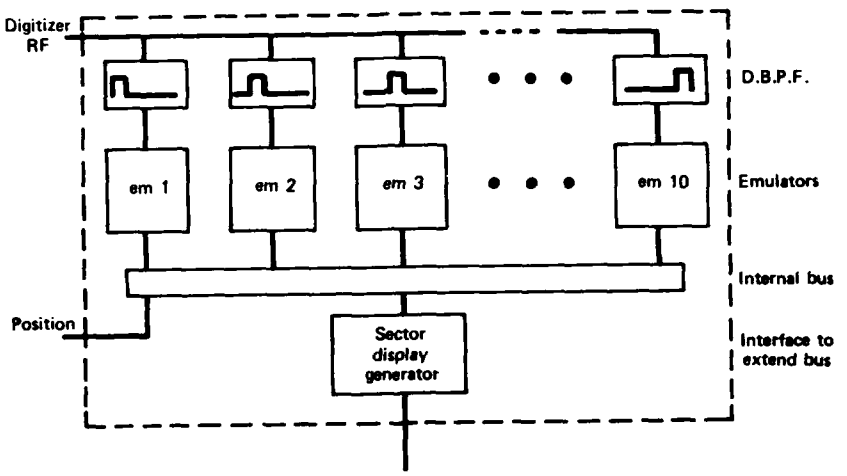


Slide 14

Slide 14 Commentary: This chart shows the simplified-representation basic structure of a VLSI emulator, i.e., a VLSI device which can run the instruction set originally designed for a larger computer, in this case the PDP-11 used in the first ECM example. The target instruction set, or language, would be translated to an intermediate form, such as P-Code. A P-Code engine would execute the instructions. Some additional work would be required to provide device virtualization, i.e., provide an environment for the operating program such that I/O commands will be serviced in what appears to be a normal manner. The device virtualization is really a mapping transformation from the devices the program expects to those that are actually available on the new system.

Emulators are fairly common in research and development context today. Many systems running on minicomputers are used to develop code for microprocessors. Perhaps the most well known single emulator is the Amdahl V-6 which runs IBM software. National Semiconductor is preparing a smaller-scale chip set to emulate the 370 series of computers, and more can be expected. Use of emulators allows the designers to reuse existing debugged software components.

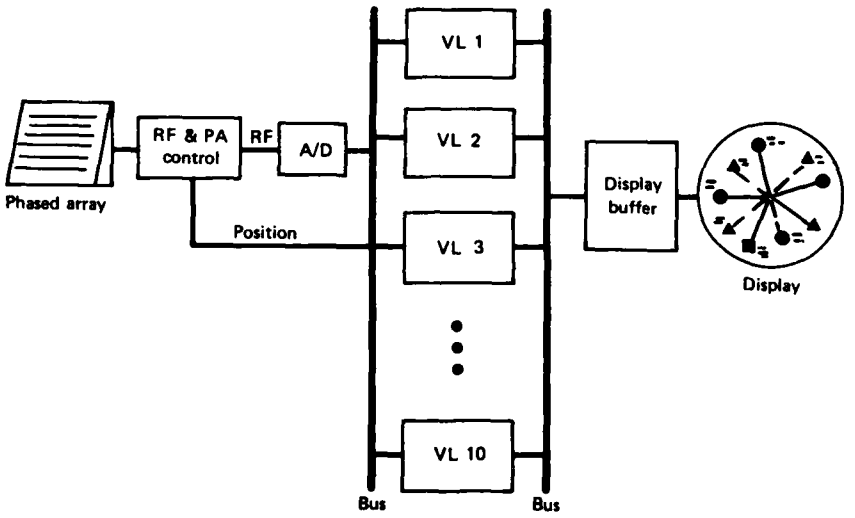
Single sector ECM VLSI component



Slide 15

Slide 15 Commentary: This chart shows how to combine these emulators with other logic in a VLSI component that will form the basis for the new high-performance emitter locator system. Digitized RF and position information enter as shown. The 100 MHz RF signal is partitioned by a series of digital band pass filters into 10 bands as shown, the first going from zero to 10 MHz, the second from 10 to 20 MHz, and so on down to the last covering the band from 90 to 100 MHz. Each of the emulators running the code from the original application is then looking at what it believes to be a 360 degree, 10 MHz problem, just as they were designed to deal with. The sector display generator then takes the collective output of the emulators and puts it together to form the union of the results of the subproblems. Additional spatial resolution comes for free here, subject to RF and propagation limitations, since, while the emulators "think" they are working on 360 degree sectors, they are actually working on 36 degrees, giving 10 times the angular resolution of the old system.

Multiprocessor implementation



Slide 16

Slide 16 Commentary: Finally, these VLSI components can be put together as shown in this chart to implement the high-performance, 100 MHz, 2000 source emitter locator system. The antenna in this implementation might be a phased array with a front end configured to control such a device. The position information would again be on the bus and available to each of the processors within the VLSI system components. A display buffer would integrate the various sector displays within the components to form the display shown on the right or to interface to other systems. The performance of this system could be increased still further by allowing a dynamic allocation of the mapping of the task onto the processor arrays. Thus, the sectors would need not be a constant 36 degrees and the bandwidth sectors would need not be a constant 10 MHz wide. Dynamic allocation based on the distribution of sources in the zone of coverage could change these parameters and allow the system to focus its attention in the most active areas of the spectrum. This kind of dynamic control illustrates part of the motivation for object-oriented languages, such as Smalltalk or MESA. One would merely assert the consistency of the partition assigned by means of adjusting the band pass filters and the spatial allocation to each component with the partition known to the display processor, and allow the detailed code to maintain this consistency to be generated by a compiler.

Multiprocessor emulation system architectures

- Maximize use of existing software resources
- Cost-effective use of new technology - " what to do with VLSI"
- Provided a suitable partition can be found
  - Identical sub-problems
  - Task segmentation by function

Slide 17

Slide 17 Commentary: This chart summarizes some of the properties of a multiprocessor emulation system architecture. We can maximize the use of existing software resources, while meeting the design requirements. In the example chosen, the software for the original system would be of a complexity that several hundred thousand dollars or more would have been spent in its development, debugging and maintainence. Second, we have made a cost effective use of new technology, in part answering Gordon Moore's initial question of what to do with VLSI. However, all of this is provided a suitable partition can be found, either with nearly identical subproblems or task segmentation by function or both. In this sense the deck was stacked for the preceding example because of the easy partition into subproblems. But many other applications can be envisioned which will be amenable to this kind of design methodology.

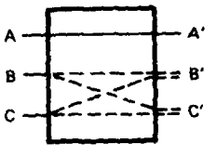
For instance, consider a retail system. A company might desire to bring out a product that integrated point-of-sale electronic cash registers with inventory management. A sophisticated inventory management process might run on a large 370 class machine, while point-of-sale terminals which read the universal product codes seen in grocery stores are generally implemented on mini- or micro computers. One could integrate the software for these two processes on a three-processor system: one processor emulating the old point-of-sale terminals to capture the data, another emulating a 370 and running the inventory system, and a third mediating and providing the forma'

changes necessary to make the data output by the first system acceptable to the second, and providing interfaces to the real I/O devices. This is a much simpler programming task than reimplementing what has already been done at great effort and expense. It makes sense in an environment where "bits" are almost free.

We can ask the question, Is this the end for VLSI technology? Will we multiprocess and emulate our way through the near-term crisis until we eventually discover what we really want to do with software, then develop the right kind of systems to run on the right kind of new machines once we figure it out, but continue producing 0.1 micron VLSI? Maybe, but this may not be the last word.

MIT/IBM physics of computation conference

- Universal and reversible logic element



Can have zero energy dissipation
Obeys conservation laws

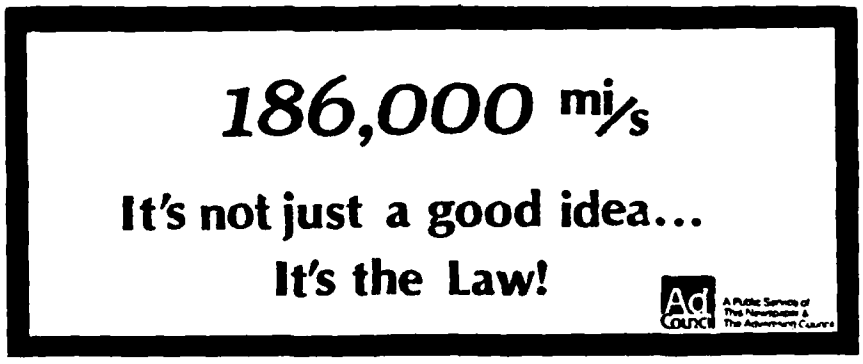
- So What? R. P. Feynman, Keynote Address —
  Derived parts of non-relativistic Q.M.
  based on arrays of these reversible automata
  ⇒ may be able to model all of physics digitally
  understanding → control → possibility of quantum computation!

Slide 18

Slide 18 Commentary: A recent conference held in Massachusetts, the MIT-IBM Physics of Computation Conference, revealed some surprising new results. The first of these was a universal and reversible conservative logic element. Reversibility is of interest because reversible processes have zero energy dissipation. Many phenomena in physics obey conservation laws, which allows one to understand them by means of their invariants, but application of conservation laws to computing is something new.

It is first useful to explain what we mean by universality before discussing the universal, conservative and reversible logic. A universal element is one that can realize any Boolean function. All of our logic is based on the set of OR, NOT and FANOUT, which provides the basis for all logic design. These designs are clearly irreversible since one element of the set, OR, has two inputs and one output. The universal and reversible logic element developed by Professor Ed Fredkin of MIT has three inputs, A, B, and C, and three outputs, A', B', and C', as shown. Output A' is always input A, and outputs B' and C' will either replicate inputs B and C or interchange them, depending on the value of A. If A is 1, B' and C' replicate B and C; if A is zero, they interchange. It is also conservative in that the number of 1s and 0s in the input set is the same as in the output. So what does all this mean? The answer probably will not be known for years, but we have a hint in the keynote address delivered by Professor Richard Feynman, winner of the 1960

Nobel prize in physics for his derivation of quantum electrodynamics. In his keynote address at the conference, Professor Feynman demonstrated a derivation of portions of nonrelativistic quantum theory based on arrays of these reversible automata. Others have speculated that we may in fact be able to model all of physics including relativistic quantum mechanics with ensembles of this kind of computing element. One can further speculate that an understanding of physics on such a quantum automata level will lead to means for control of these processes, perhaps even quantum computation. I don't pretend to know what this means or where it will lead, but a glimmer of an answer can be found on a bumpersticker seen at the conference and on this last chart (slide 19): "186,000 miles per second--it's not just a good idea, it's the law."


186,000 mi/s
It's not just a good idea...
It's the Law!
Ad Council
A Public Service of
This Newspaper &
The Advertising Council

Slide 19

END

DATE
FILMED

8 82

DTIC