LBL-16194 c.2

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

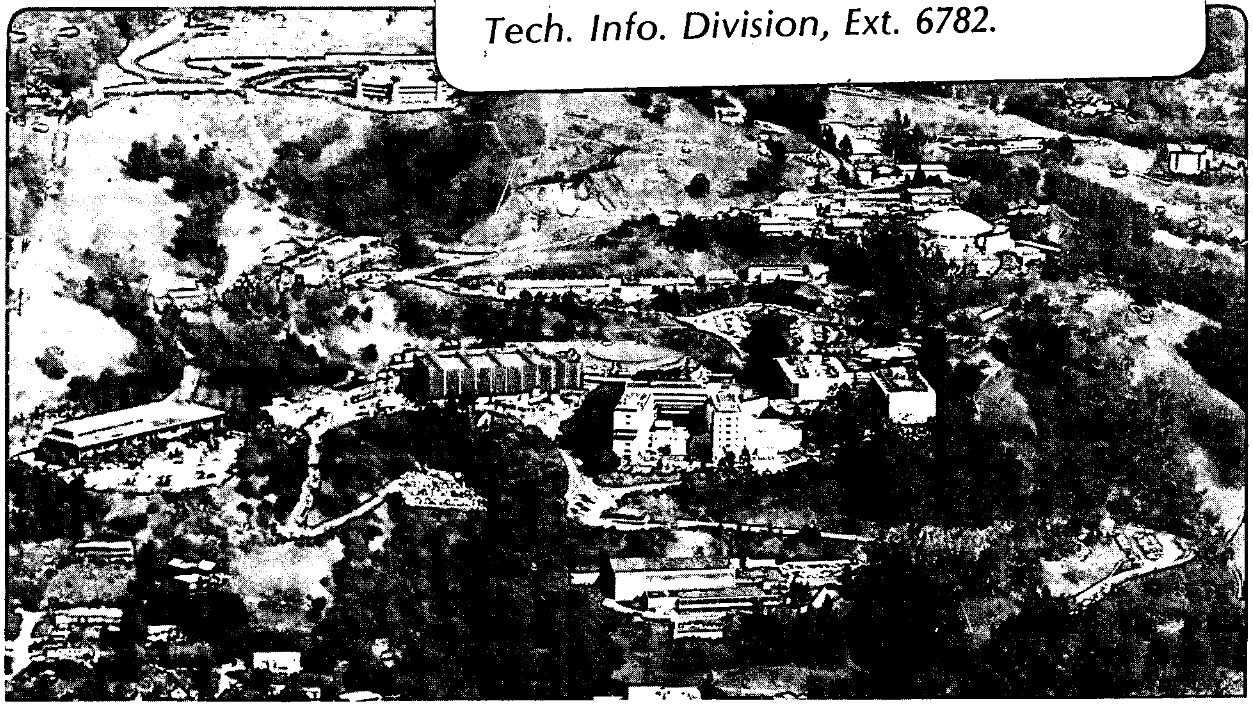
Engineering & Technical Services Division

Presented at the 17th Conference on Circuits, Systems, and Computers, Pacific Grove, CA, October 30 - November 2, 1983

MULTIPLICATION OF PROCESSING CAPACITY WITH A PARALLEL PROCESSOR ARRAY

J. Meng

October 1983



LBL-16194 c.2

Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

MULTIPLICATION OF PROCESSING CAPACITY WITH A PARALLEL PROCESSOR ARRAY

John Meng

Lawrence Berkeley Laboratory, University of California
Berkeley, California 94720

Abstract

A parallel connection of eight minicomputers is under test in an attempt to deal interactively with arrays of experimentally-generated kiloparameter data events. We have been able to achieve computer amplification factors linearly proportional to the number of executing processors. By replacing our minicomputers with single chip processor arrays, we expect to observe additional multiplication of data processing capacity.

Introduction

Since about 1964 we have been designing and building custom minicomputer-based systems for use in high-speed data acquisition and analysis. Figure 1 uses these nearly two decades of experience as its data source, starting with a single PDP-5 based system in 1964-1965 and presently with a ModComp-based parallel processor system. The near future could easily be microprocessor-based hyperparallel systems. The principle subjects of this paper are the last two data points on the system hardware capacity line in Fig. 1. The last point represents the projected capacity of a hyperparallel system. The next to last point represents the measured capacity of the prototype parallel processing unit we now have operating. To properly introduce these two, we need to briefly explore their time context as well as their environmental context.

"System hardware capacity" (Fig. 1) is a contrived number derived empirically from our experiences with what constitutes computer power applied to our needs. It is proportional to instructions executable per second, byte-length of the data word and the log (base 2) of memory capacity in words. From about 1967 until 1980, hardware capacity has increased exponentially. Minicomputers running as single processors are, in the 1980's, facing limits beyond which it will be difficult to go. Major improvements in the individual processor are unlikely, and particularly in the case of the minicomputer promise to be uneconomic.

Data acquisition and analysis in our research environment tends to be the driving force for system development. The second curve on Fig. 1, cost per unit of value, illustrates this dramatically. Points on the curve are simply system

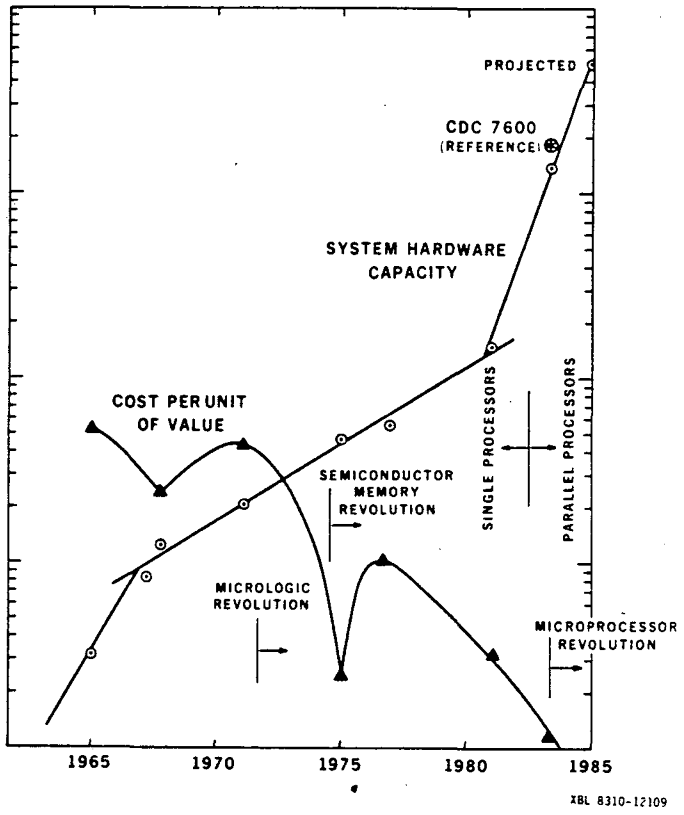


Fig. 1 Processing power of the computer systems we assembled increased exponentially from the late 1960's through 1980. For the near future, developments in parallel processor architecture simultaneously with still another hardware revolution--the microprocessor revolution--have steepened the curve. Per unit of value, costs of very powerful systems, in spite of earlier bumps, are plunging.

costs divided by system hardware capacity. When the first system was installed, it connected to an external data-taking device; a pulse height-analyzer. The second group of systems demonstrated the feasibility of using computer program memory to do the histogramming, the feasibility of storing an incoming data stream verbatim on magnetic tape for later replay and analysis and the ability to allow more than one person simultaneous access to the computer. The next system was expected to have these capabilities, but with greatly expanded capacity, explaining its high cost. The introduction of micrologic and later of inexpensive semiconductor memories and finally of microprocessor technology accounts for the later improved cost per unit of value.

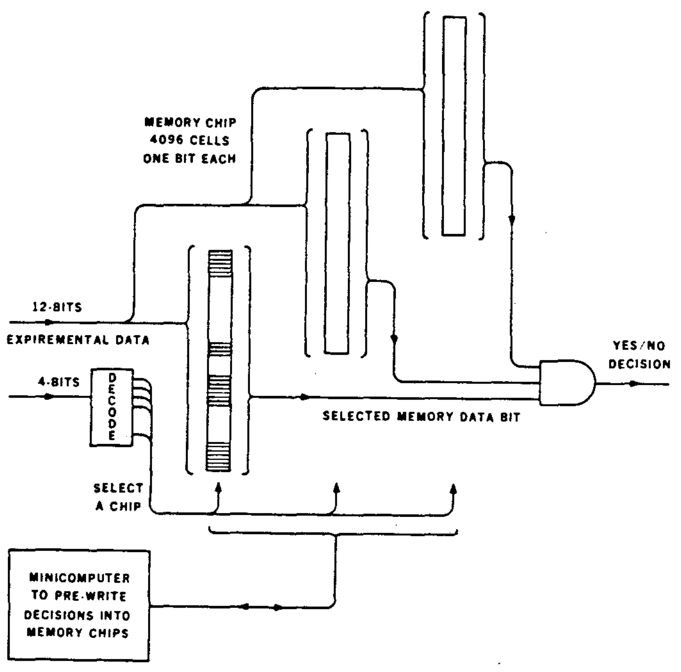
The change in data acquisition was a general change in philosophy. Originally, data was partially processed as it was taken. Monotonically, the trend has been to acquire and store all available data, relying on subsequent replay for analysis and reanalysis. Such data normally consists of sequential discrete events, either independent or correlated, and each represented by a group of parameters. Whereas in the 1960's and early 1970's, eight parameters per event usually was adequate, today's detectors are spewing out hundreds or thousands of parameters per event. Data acquisition today is often limited only by the speed and density of affordable tape machines. Data analysis has become a major bottleneck because of the volume of data, and the value of human judgement and iterative procedures in the analysis. Interaction requires fast replays, but high data volume makes fast replays difficult.

Our initial response to this pressing demand was the conceptualization of high-speed special processors controlled by a minicomputer. For example, the sorting module in Fig. 2 was once proposed. Yes-no decisions were preprogrammed into memories. Data then drove the memory address lines and a decision appeared within half a microsecond. To make it more general, plans were made to have a 'stable' of special-purpose processors, all controlled by one minicomputer. This eventually evolved into our existing parallel-processor, where eight general-purpose processors are controlled by a single minicomputer.

Parallel Processing Array

Our initial objective was to build special-purpose processors, each to perform a limited set of tasks at very high speed, and to run them as a group under the control of a separate minicomputer. Driven by practical and technological realities, we decided to use minicomputer central processors. As general-purpose devices, they would be able to do the various special-purpose jobs required. Using a mincomputer from the same manufacturer who built the system controller let us make more efficient use of our software support. In order to achieve our objectives, it would be necessary to be able to 'amplify' the data throughput of one processor by the number of processors we planned to run. The phase I prototype, completed in late 1982, ran five processors. Phase II, now operational, is running eight processors (Fig. 3).

Not starting from a conventional processor design orientation or with the general purpose goals of normal processor design, our configuration has several unique features. Philosophically, we felt it to be of prime importance to separate data processing channels from control channels. Thus we began by prohibiting data-processing central processors (CPU's) from performing any control functions. Processors are interconnected within the data stream by what is in effect a crossbar switch (Fig. 3).



XBL 8310-12077

Fig. 2 An early proposal for a specialized sorting processor under independent minicomputer control. The minicomputer, operating under direct control of an experimenter, prewrites bits into memory locations. High speed sorting occurs when data is used as addresses and the pre-written memory bits are read to produce an accept/reject decision. This idea led to the concept of many special processors under minicomputer control; precursor of our prototype parallel processor configuration.

Control of the system resides in a separate minicomputer system, and is exercised over the multiple processors via emulated front panel controls[1]. Input and output for the data processors is done by switching memory blocks. Memories are filled from an input pipeline processor, switched into the address space of an available minicomputer CPU, and finally switched onto another pipeline processor to be emptied. Memory switching requires less than 50 ns. The input pipeline has the responsibility for presorting to separate data from comments or labels. It also contains high speed RAM-driven logic used to separate events from each other and stack them into predetermined fixed locations in the memories.

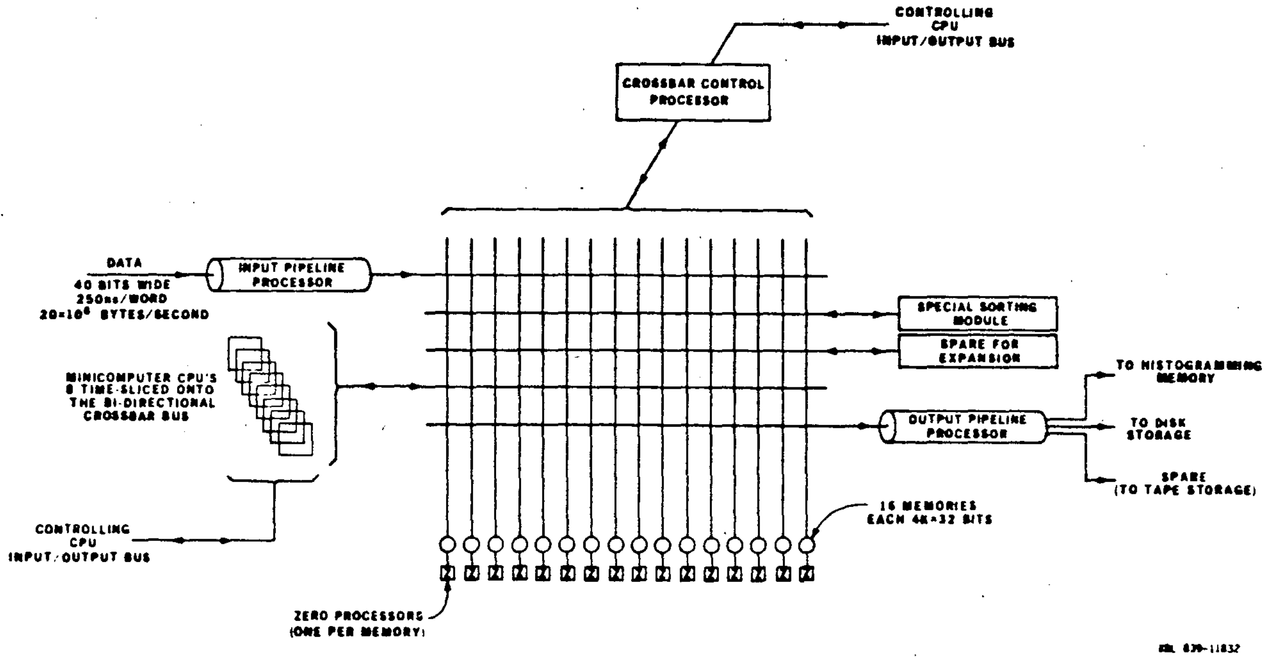

CONTROLLING CPU INPUT/OUTPUT BUS
CROSSBAR CONTROL PROCESSOR
DATA
40 BITS WIDE
250ns/WORD
20×10⁶ BYTES/SECOND
INPUT PIPELINE PROCESSOR
SPECIAL SORTING MODULE
SPARE FOR EXPANSION
MINICOMPUTER CPU'S 8 TIME-SLICED ONTO THE BI-DIRECTIONAL CROSSBAR BUS
TO HISTOGRAMMING MEMORY
OUTPUT PIPELINE PROCESSOR
TO DISK STORAGE
SPARE (TO TAPE STORAGE)
16 MEMORIES EACH 4K×32 BITS
CONTROLLING CPU INPUT/OUTPUT BUS
Z
ZERO PROCESSORS (ONE PER MEMORY)
XBL 839-11832

Fig. 3 Our parallel processor configuration employs a crossbar to interconnect memories to special-purpose processors. Control functions are not allowed in the high-speed data stream. They are exported to attached dedicated minicomputers and microcomputers.

Control of the memories is done with a hybrid microcomputer/hardware controller. It is possible (though not in our existing prototype) to run different code in different CPU's (or groups of CPU's) and to 'thread' a data memory through first one CPU group and then another and another, etc.[3] Because control is exercised via front-panel emulation, the controlling minicomputer is able to monitor the operations of the data processors without slowing execution. This monitoring operation could be used to select more or fewer processors to run a code, thus dynamically optimizing the system for any particular problem. More details are presented in references.[2,3,4]

Results of Prototype Tests

The input pipeline is 40-bits wide (5 bytes) and shifts one word every 250 ns (4×10^6 H_3). Data passes through each stage at $5 \times 4 \times 10^6 = 20 \times 10^6$ bytes per second. In reality, our 300 MByte disk can only supply 0.64×10^6 bytes per second. Figure 4 illustrates the data path. Results appear graphically in the lower part of the figure. Data transport memories are zeroed, switched to the input pipeline processor to be filled, switched to an available minicomputer central processor for processing and finally connected to the output pipeline to be emptied. From there, the cycle starts again by the memory being zeroed.

The left graph (Fig. 4) is the result of measuring data throughput with different numbers of minicomputers switched into the available pool. Except for the last two points (seven and eight processors) the system is compute-bound. With seven or eight CPU's running, the entire system has to wait for data from the disk controller.

The processors are running identical programs compiled in FORTRAN and down-loaded from the download and control processor before the test began. The FORTRAN code was taken unchanged from other computers (CDC7600 PDP-11 and Mod Comp Classic). The right-hand graph (Fig. 4) plots results for a completely compute-bound case. The vertical axis is a ratio of system performance to the performance of a single stand-alone minicomputer (with the same type of CPU as those in the system). The lower limit of each bar is the result of running identical programs in the system and in the stand-alone computer. The relationship is very nearly linear with a slope of one. We then went back to the program and tried to optimize it for use on our multiprocessor system, producing the results illustrated by the tops of the bars.

Hyperparallelism

Several different scenarios have been proposed for future developments.[2,3] From a hardware standpoint, we are in the midst of a microprocessor revolution and an ongoing semiconductor memory revolution. As microprocessors approach discrete systems in speed and function, we are being given the option of replacing our two large, 125-watt-each minicomputer CPU boards with a few chips dissipating nearly two orders of magnitude less power. A third large board, also a candidate for miniaturization houses the minicomputer's memory.

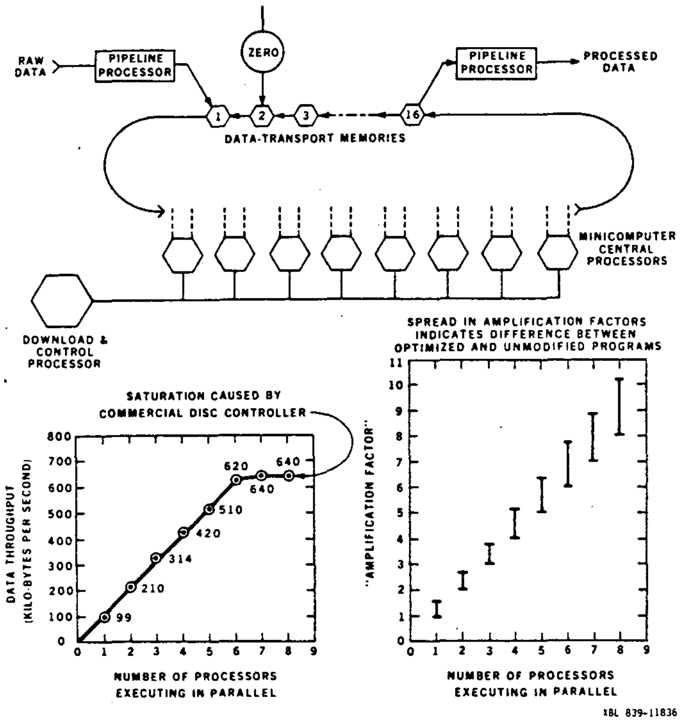

RAW DATA
PIPELINE PROCESSOR
ZERO
PIPELINE PROCESSOR
PROCESSED DATA
DATA-TRANSPORT MEMORIES
MINICOMPUTER CENTRAL PROCESSORS
DOWNLOAD & CONTROL PROCESSOR
SATURATION CAUSED BY COMMERCIAL DISC CONTROLLER
DATA THROUGHPUT (KILO-BYTES PER SECOND)
99
210
314
420
510
620
640
640
NUMBER OF PROCESSORS EXECUTING IN PARALLEL
SPREAD IN AMPLIFICATION FACTORS INDICATES DIFFERENCE BETWEEN OPTIMIZED AND UNMODIFIED PROGRAMS
"AMPLIFICATION FACTOR"
NUMBER OF PROCESSORS EXECUTING IN PARALLEL
XBL 839-11836

Fig. 4 Configured as shown here, the architecture of Fig. 3 is running with processing capabilities directly proportional to the number of processors executing. Foldover of data throughput has been observed at limits of system input devices and system output devices. The greater-than-one slope in the right-hand graph is the result of taking advantage of specialized input, zeroing and output processors.

Using new high-density memory chips and new microprocessor chips, it becomes practical to compress our entire system. The resulting one or two large logic cards can then be assembled into a parallel array, giving us a parallel array of parallel processors - the hyperparallel array. (See Fig. 5.) It seems clear that for certain problems - those involving sequences of independent data blocks where processing speed limits data throughput - this array can significantly multiply processing capacity once again.

To prevent the crossbar structure from becoming unwieldy, we propose filling a large segmented memory from the input pipeline. Segmentation can be accomplished by proper address decoding and memory chip size selection. The segmented memory is then given as a block to a hyperparallel processor array where the segments 'separate' and each associates with one of the microprocessors. Finally is the 'reassembly' of the memory array for transport to the output pipeline. Degradation of performance can occur within a hyperarray because the subarray must wait until all its processors are finished before releasing the segmented memory for emptying. It would be expected that all processors in a hyperarray would run the same program code during execution, so downloading is no more of a problem with this system than it is with the original system. However, it will be desirable for an external processor to be able to monitor the performance of individual processors within each hyperarray.

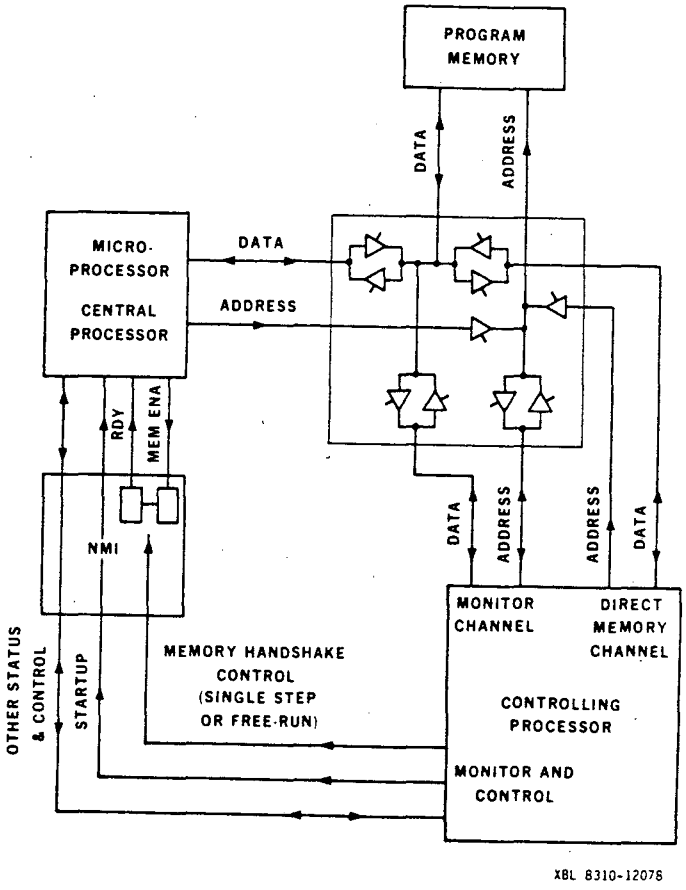


Fig. 5 Microcomputer central processor chips may be controlled and monitored at the chip-memory interface. When we replace minicomputer processor cards with single chip devices, we can use this connection instead of the operator control panel emulators we are using on our prototype system.

We can interject control between the microprocessor and its external memory, as shown in Fig. 6. The controlling processor controls the memory 'ready' line and can force a pause between memory cycles. During the pause, it can read (or supply) memory data and address as well as reading any available status signals. By controlling non-maskable interrupt lines and vectors, it can control startup.

The controlling computer can supply the microprocessor with program steps allowing it the same absolute control and access it had via the minicomputer operator control panel connection. Since the interjection is between micro and memory, the controlling processor can easily download programs directly into micro memory. The controlling processor has non-interfering monitoring capability over microprocessor address and data lines, allowing for optimizations similar to those available in the prototype system.

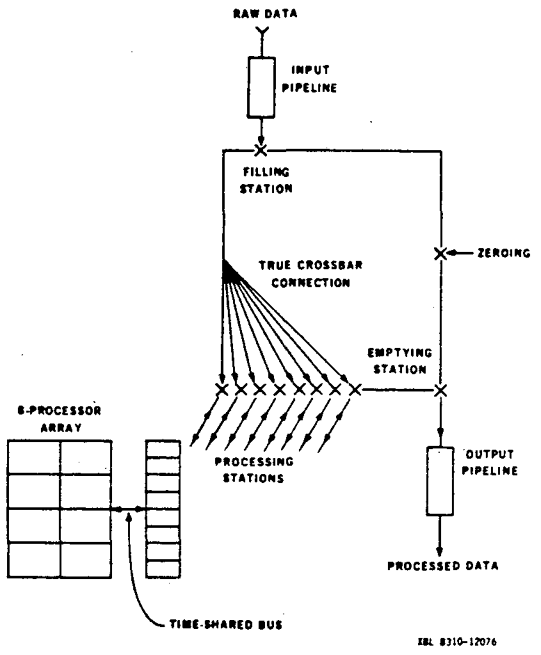


Fig. 6 The microprocessor revolution in concert with the ongoing semiconductor memory revolution is making it physically and economically practical to think about building the equivalent of our prototype parallel processor system onto a set of two or three processor boards which could be included as elements in the same general structure we are now using. The predicted result is a further amplification of processing power.

Conclusions

Historical development in demand for data taking and analysis led us to develop a unique parallel processor configuration. The resulting system runs processing codes nearly unchanged from single stand-alone systems, and with eight processors running in parallel executes the code about eight times faster than the stand-alone system. Optimizing programs to take advantage of our multiple processor architecture can result in additional processing speed enhancements of as much as 25%. A newer system, using a parallel array of parallel microprocessors has the potential of further dramatic enhancement of system performance for certain data sets.

Acknowledgments

I wish to acknowledge the unwavering help of F. Gin, H. Jackson and T. Shimizu during the development of this system and during a prior decade of other developments. D. Weaver, C. Maples, W. Rathbun and D. Logan were instrumental in the development of the prototype parallel processor systems.

This work was supported by the Director's Office of Energy Research, Office of High Energy and Nuclear Physics, Division of Nuclear Physics, and by Nuclear Sciences of the Basic Energy Program of the U. S. Department of Energy under Contract Number DE-AC03-76SF00098.

References

1. Meng, J., "Controlling a Radially-Connected Array of Minicomputers." Conference Record, Sixteenth Asilomar Conference on Circuits, Systems and Computers, Nov., 1982, pp. 280-284. Also Lawrence Berkeley Laboratory report LBL-14471.

2. Weaver, D., Creve Maples, John Meng and William Rathbun, "Operating System Considerations in the Multiprocessing MIDAS Environment." Lawrence Berkeley Laboratory Report, Oct., 1983.

3. Maples, C., Daniel Weaver, John Meng, William Rathbun and Douglas Logan, "Utilizing a Multiprocessor Architecture - The Performance of MIDAS." Lawrence Berkeley Laboratory Report, Oct., 1983.

4. Maples, C., William Rathbun, Daniel Weaver and John Meng, "The Design of MIDAS - A Modular Interactive Data Analysis System," IEEE Trans. on Nuclear Science, NS-28, 3880 (1981).