Title

CONTROLLING A RADIAliY-CONNECTED ARRAY OF MINICOMPUTERS

Author

Meng, J.

Publication Date

1982-05-01

Lawrence Berkeley Laboratory

UNIVERSITY OF CALIFORNIA

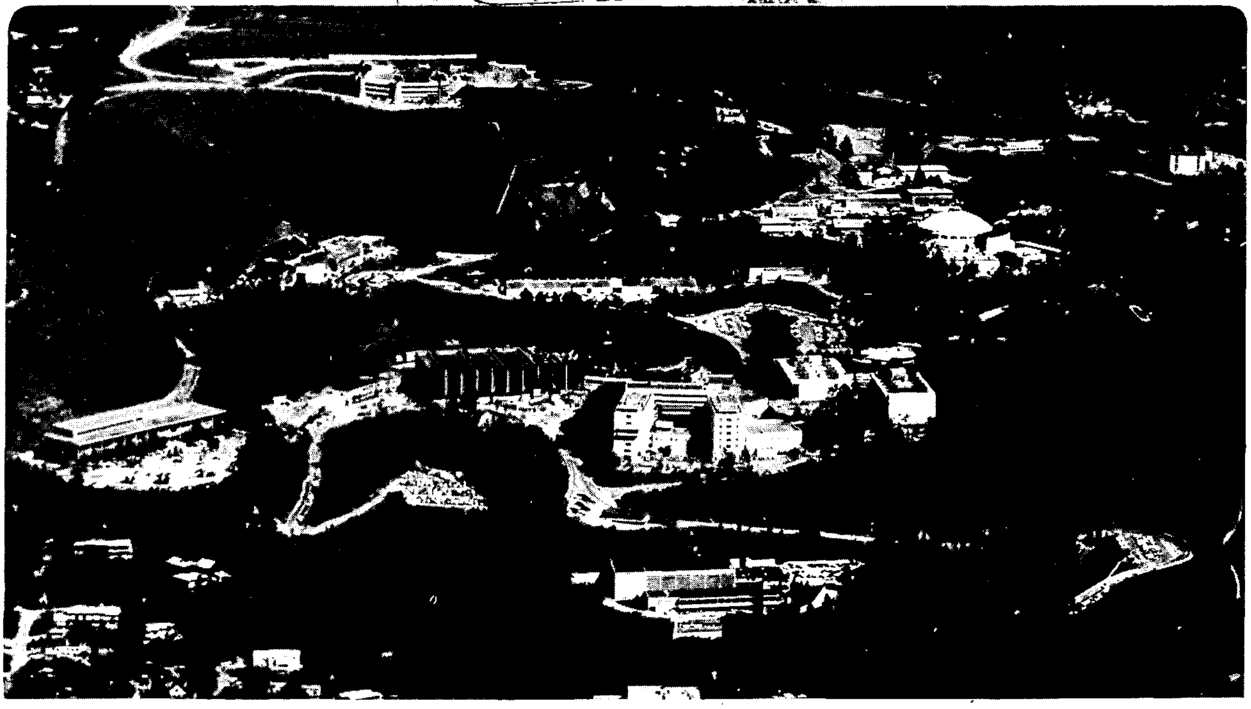
Engineering & Technical Services Division

Presented at the Sixteenth Asilomar Conference on Circuits, Systems, and Computers, Pacific Grove, CA November 8-10, 1982

CONTROLLING A RADIALLY-CONNECTED ARRAY OF MINICOMPUTERS

John Meng

May 1982



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

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

LBL-14471

CONTROLLING A RADIALLY-CONNECTED ARRAY OF MINICOMPUTERS

John Meng

Lawrence Berkeley Laboratory, University of California
Berkeley, California 94720 USA

Abstract

To connect a radial array of eight minicomputers to a single master minicomputer we have simulated front panel controls, in which switch contacts are electronically made or broken under program control in the master, and data is returned to the master via what normally would be register displays. Experience so far indicates that this means of connecting to an array of minicomputers has many powerful advantages over other more conventional computer array control schemes.

Summary

Arrays of processors are useful tools in the solution of several types of problems. For example, Monte Carlo calculations lend themselves to the truly simultaneous multiple processing available from a radially-controlled array of processors. Exhaustive search or processing operations on large data bases are expedited by having a group of processors work simultaneously on data chunks.

Feeding and control of such an array of processors usually takes the form of a data link either into the input/output bus structure or into and out of the memory associated with each element of the array of processors. Linkage is dependent on a program executing in the master processor communicating with programs executing in the slave processors. In order to eliminate the many problems inherent in such an approach, we have connected the master processor to the slave processors by building electronic equivalents of the operator control panel (OCP) of each slave processor. The master processor is able to "close" and to "open" any of the switch contacts normally available at the slave processor's control panel. It can read the data and status normally available on the operator's display. This approach gives the master computer total control over each slave. It can down-load programs, initialize registers and start execution. It can spoon-feed parameters and data to any one or to all of the slaves. It can trace program execution and detect problems quickly.

Background

Earlier discussions resulted in the 1978 proposal[1,2] to build a multi-user data analysis facility using radially-connected arrays of minicomputer central processors as the computational engine instead of using a single conventional large-scale central processor. The rationale behind such a proposal was in part economic and in part based on matching the processor to the logic common to many raw-data based analysis problems. The system is segmented in such a way that selected pieces can be replaced as they become obsolete without necessitating the replacement or modification of other components. For example, the minicomputer arrays, the 'supermicros' of Fig. 1, can be upgraded to improve data throughput without requiring replacement or even program modifications of the large interactive system at the peak of the pyramid. Thus the system defies total obsolescence and this makes it attractive from an ecomonic standpoint.

Many classes of raw data, in particular experimental data from the nuclear sciences, are naturally segmented into finite events and consequently lend themselves to genuine parallel processing. Operating several independent processors in parallel on a common data base is more nearly an optimum solution for this task than is depending on a single processor which must handle all the data sequentially.

These thoughts, among others, led us to develop a scheme for actually connecting multiple minicomputer-class processors to a large data base.[3,4,5] Figure 1 is a rough conceptualization of the scheme, known as the Modular Interactive Data Analysis Systems, or MIDAS. In general, control flows downward from the peak of the pyramid. The level to which we are addressing ourselves is the connection between each mini and its group of 'supermicros.' This connection is required to handle program downloading, control signals and status from the supermicros, and in some instances data and/or parameter loads.

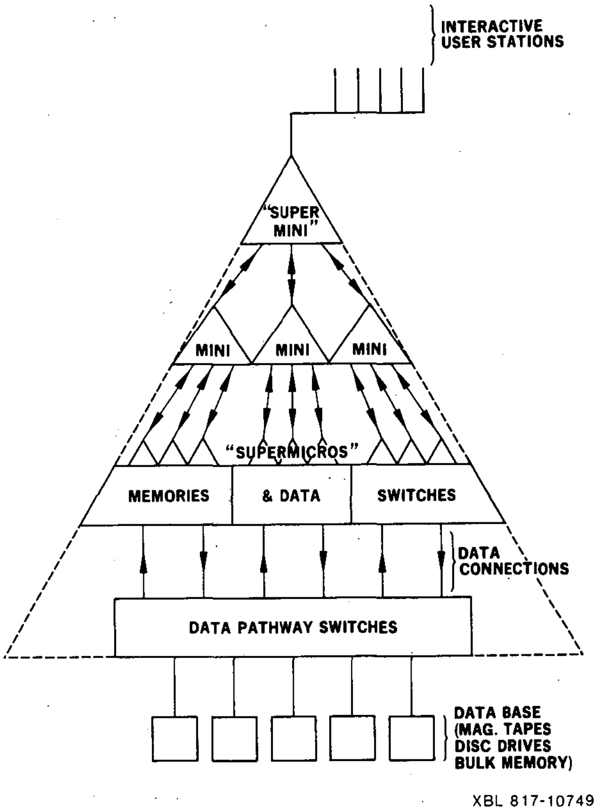


Fig. 1. MIDAS - Computational modules reside at the 'supermicro' level.

Alternatives

Figure 2 is an enlargement of the region under scrutiny. During our deliberations on the exact structure of the first supermicros to be used in MIDAS, several alternatives were considered. These ranged from the in-house construction of a processor to the purchase of complete minicomputer systems. In-house construction seemed too ambitious a project for our limited staff and resources. Purchase of complete minicomputer systems was prohibitively expensive for our prototype budget. Our compromise was to purchase, untested, CPU (Central Processor Unit), EAU (Extended Arithmetic Unit) and memory cards from a computer manufacturer. There being no conventional input/output connections to the supermicro level, we did not purchase input/output cards. Our available options for communications between a master computer and a slave processor would normally be either via the input/output bus or via transfers into and out of memory. Having elected to purchase the processors without input/output capability, our remaining option would seem to be communication by memory block.

Either technique--memory block or input/output communication--poises problems when the range of control functions is considered. Each slave processor must contain an executing program capable of handling program down-loading and startup, passage of operating parameters and occasional data, and passage of condition codes and status information. The system executing in the slave processor must be tolerant of the foibles of amateur programmers trying to make use of the system. Possible though it may be, having a controlling program in the slave processor represents the use of a substantial amount of program memory. It also represents a substantial investment which must be sacrificed when slave processors are changed.

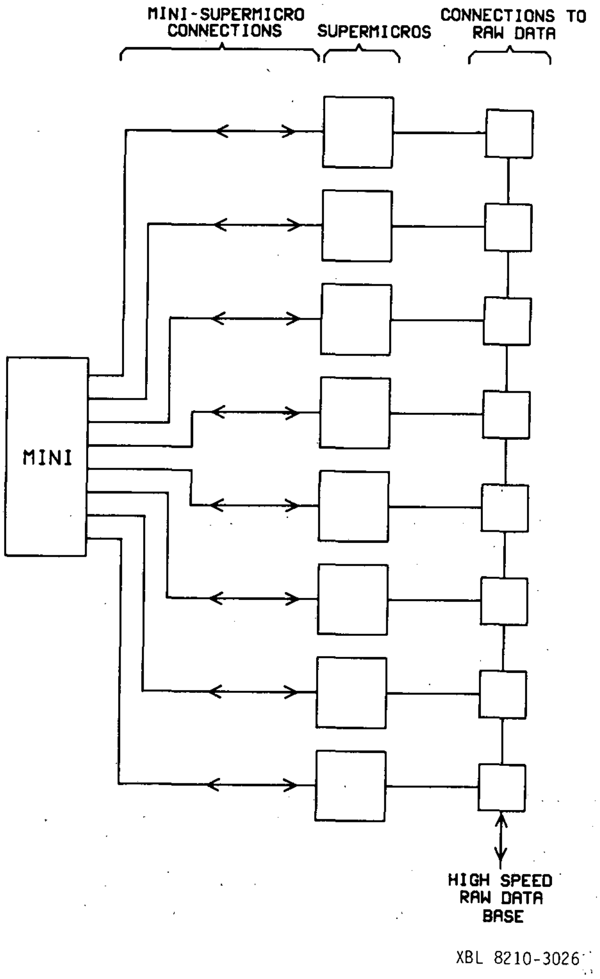
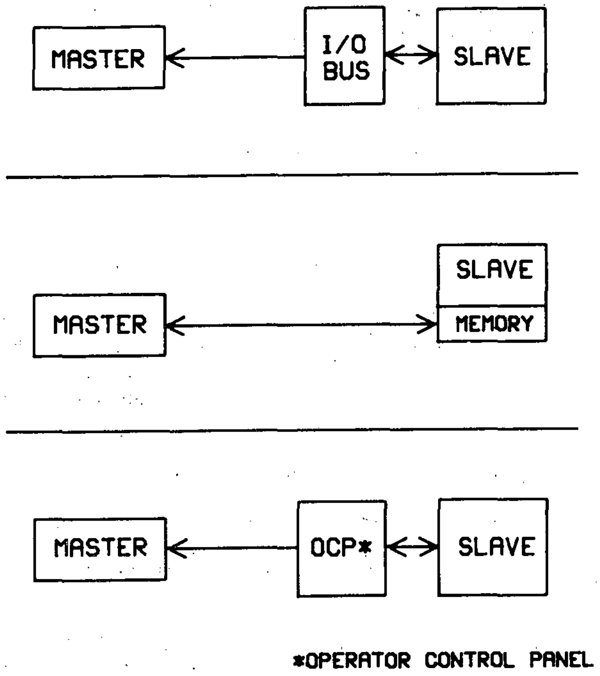


Fig. 2. The area of particular interest is the mini-supermicro connection; required to control eight slaves from a single master.

Another communication channel into and out of the slave processors does exist. This is the primative human-processor interface--the operator control panel (OCP). The normal mode of communication using the OCP takes the form of switch closures sensed by the processor and of visible lamps turned on or off by the processor in response. The range of functions available via this route is normally extensive; including the examination and alteration of registers, examination and alteration of memory contents and complete control over the machine's execution. We decided to replace the various control panel switches with flip-flops set or reset by input/output commands executed on the master computer. Slave processor lights are connected to be electrically read by the master computer. The master computer becomes a super-fast operator to the slave, exercising absolute control.

Of the three alternatives (Fig. 3), this last scheme won because of its inherent simplicity, both in construction and operation; and because of the extent to which it awards control to the master computer.



XBL 8210-3025

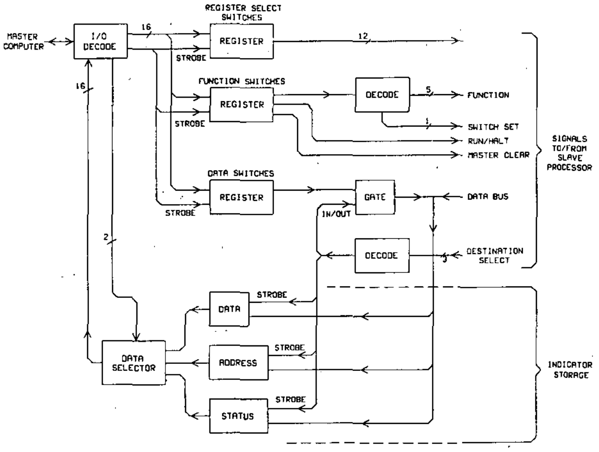
Fig. 3. Three available means of communicating with a slave processor. Only communication via the operator control panel does not require the slave to be executing a program.

Implementation

Figure 4 is a sketch of the logic required to simulate the OCP of a supermicro in MIDAS. The details of such an implementation are uniquely peculiar to the computers being used as slaves and to that being used as a master. In MIDAS, both master and slave are currently ModComp Classic types. Ironically, the master is the less powerful of the two models. Each slave processor is given its own OCP simulation logic.

Slave processors can be controlled individually. Each OCP is assigned a unique master computer input/output device code. In addition, one device code is assigned to simultaneously control all slave processors. It is possible to load or to control processors individually, but it is also possible to load or to control processors in gang mode--all at once.

The ModComp Classic operator control panel uses the signals shown on the right side of Fig. 4. Register select switches are generally used to select the source of data being displayed on the data indicators. The function decode from a second register enables functions such as 'load memory address,' 'load memory data,' or 'load memory data and step program count.' The 'switch set' line strobes the enabled function. Run (not-halt) and master clear appear as unique bits in the function register. The third register loaded from the master computer emulates data switches; a source for functions needing data.



XBL 8210-3027

Fig. 4. Connection to one slave processor. Each slave processor has a duplicate of this logic.

Three destination-select lines are decoded to provide strobes for three registers which normally would drive only display lamps. Data for these registers comes off a data bus out of the slave processor. A data selector connected to the master computer input/output bus selects and reads any of these display registers on command. When the destination select lines are not generating strobes for the indicator registers, a signal is generated which gates data from the data-switch register onto the bus to be used by the slave processor.

Use of this connection by the master computer amounts to having a super-fast operator standing at the control panel of each slave processor. Slaves need no special start-up hardware or software. They need no resident system to handle the loading, monitoring and control functions associated with down-loading and execution of software tasks.

Uses

Downloading and Execution

Each program to be executed in the slave processors becomes a 'thumb-in;' loaded with a rapid sequence of 'store next' switch closures. To execute down-loaded programs, the program counter is set with another switch closure and the RUN switch is turned on. During normal operation, slave processors typically get the attention of the master computer by halting. Codes in data, address or status indicators signal the nature of the request.

Program Debugging

Debugging slave processor programs is enhanced by having the master computer 'single-step' through the down-loaded program. The master computer can perform this operation at high speed until interesting areas are found. The master can then print register and/or memory and status data as the stepping proceeds through interesting or questionable code. Whereas this technique is most often impractical for a human operator, the master computer steps and monitors at microsecond speeds, rapidly skipping through uninteresting blocks of code; pausing only for examinations of areas of particular interest.

Hardware Maintenance and Debugging

The master computer can run memory diagnostics on slave computer memory by way of a program running in the master which is exercising slave memory through slave front-panel operations.[6] Another way of running memory diagnostics is by down-loading a program into the slave and then monitoring slave operations. However, this second technique requires the slave to be nearly operational. The first technique only requires access to the memory by front panel operations. This comparison of techniques applies to many slave-CPU maintenance operations. Our front-panel connection lets us debug defective hardware without depending on the defective hardware being able to execute diagnostics.

Conclusions

Where it is desirable to gang arrays of computers into a slaved parallel operating block, controlling the slaves can be done effectively and efficiently by replacing slave front panel switch closures and indicators with flip-flops. The master computer is then connected to control switch closures and to monitor slave processor indicators. Benefits are complete, effective control of the slaves by the master, enhanced hardware debugging and maintenance capabilities, and greatly enhanced program debugging capability.

Acknowledgment

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.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U. S. Department of Energy to the exclusion of others that may be suitable.

References

1. Maples, C. C., "A Specialized, Multi-User Computer Facility for the High-Speed, Interactive Processing of Experimental Data," Proc. of Computerized Data Acquisition Systems in Particle and Nuclear Physics Conf. Santa Fe, NM, May 14-17, (1979).

2. Maples, C. C., "Proposal for a High-Speed Interactive Facility for the Reduction and Analysis of Scientific Data." Presented at the American Physical Society, Asilomar, CA, Nov. 1-3, 1978. Lawrence Berkeley Laboratory report LBL-7196 (1978)

3. Maples, C., Rathbun, W., Meng, J. and Weaver, D., "A Fast Time-Sliced Multiple Data Bus Structure for Overlapping Data Transfers and Transformations," Presented at the Topical Conference on Computerized Data Acquisition in Particle and Nuclear Physics, Oak Ridge, Tn., May 28-30, 1981. Lawrence Berkeley Laboratory report LBL-12498.

4. Maples, C., Rathbun, W., Weaver, D., and Meng, J., "A Design of MIDAS - A Modular Interactive Data Analysis System," Presented at the Topical Conference on Computerized Data Acquisition in Particle and Nuclear Physics, Oak Ridge, Tn., May 28-30, 1981. Lawrence Berkeley Laboratory report LBL-12504.

5. Maples, C., Weaver, D., Rathbun, W. and Meng, J., "The Utilization of Parallel Processors in a Data Analysis Environment," Presented at the Topical Conference on Computerized Data Acquisition in Particle and Nuclear Physics, Oak Ridge, Tn., May 28-30, 1981. Lawrence Berkeley Laboratory report LBL-12505.

6. Meng, J. and Weaver, D., "Use of Embedded Microcomputers in System Debugging and Maintenance," Presented at the IEEE Conference, Applications of Mini and Microcomputers, San Francisco, Ca., Nov. 9-12, 1981. Lawrence Berkeley Laboratory report LBL-12313.

TECHNICAL INFORMATION DEPARTMENT
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720