THE CURRENT STATUS OF MICROCOMPUTER ARRAYS

John C. Knight
NASA Langley Research Center

Robert G. Voigt
Institute for Computer Applications in Science and Engineering*

INTRODUCTION

In this paper we survey some current research on microcomputer arrays. There are a large number of projects currently under development, and some are at the stage of constructing hardware, while others are at the design and planning stages. We will not attempt to present an exhaustive discussion of these efforts, but rather a general overview of the topic followed by a more detailed discussion of three projects whose purposes are specifically scientific computation.

MOTIVATION

A microprocessor is the central processor of a small computer in the form of a single integrated circuit. By combining a microprocessor with a memory and a few other components, a complete computer of rather limited power can be constructed. At the present time, microprocessors typically use a word size of 8 or 16 bits, use 16-bit word addressing, and have a 1-word integer add time of a few microseconds. Often they do not provide multiply or divide instructions, and floating point operations must be explicitly programmed. The remarkable thing about microprocessors which makes them so attractive is their cost. At the time this paper was written, a microprocessor could be purchased for a few tens of dollars, and a complete microcomputer could be built for a few hundred dollars.

As the general level of semiconductor technology advances, microprocessors will become even less expensive, faster, and more sophisticated. In addition, a complete microcomputer on a single integrated circuit, a few of which are produced now, will become more widely available. Many projections from the semiconductor industry have appeared in the literature and indicate the probable availability of 32-bit microprocessors with built-in floating point hardware by the early 1980's.

---

*Research by the second author was performed under NASA Contract Number NAS1-14101 while he was in residence at ICASE, NASA Langley Research Center, Hampton, Virginia.

Despite the limited capabilities of a single microprocessor, it is clear that very significant computing power could be achieved by linking many of them (perhaps thousands) together in some way, and their very low cost now makes this financially feasible even though there remain technical problems.

From the point of view of the problem to be solved, there appear to be two clear motivations for considering arrays of microprocessors. For problems of moderate size, those that tax the resources of present day computers (but can be solved), the motivation is one of reducing both the cost and the time for obtaining a solution. For example, in structural analysis one routinely solves design problems that require on the order of 10 minutes of CDC 6600 computer time. However, to optimize that design, the engineer might like to solve the basic problem hundreds of times. The conflict is clear, and a compromise on the number of runs is usually made.

At the other end of the scale there are problems of significant importance that are simply too large for any present-day computers or those that will be available by 1980. For example, to model most of the details of the flow of a gas over a complex body, such as an aircraft moving through the atmosphere, would require a grid in excess of 10^6 points over which the three-dimensional Navier-Stokes equations would be discretized. It is predicted (ref. 1) that in order to model this flow in a few hours of computer time, a computer capable of performing at a sustained rate of 1 billion floating point operations per second would be required. The current super computers are capable of peak rates on the order of 100 million floating point operations per second. It is conceivable that the required raw performance could be obtained with a large array of microprocessors. Whether or not such an array could be used efficiently for scientific problems is another question with no clear answer at this point. A discussion of this problem is beyond the scope of this paper, but it may ultimately be the most difficult to solve.

PROCESSOR ORGANIZATIONS

If many microprocessors are going to be used in parallel in the solution of a large problem, then each processor will be working on a very small part of the problem at the same time. This leads to a requirement for some form of communication between the processors since in general each processor will need the results generated by others as it proceeds. For example, if many microprocessors are used to solve a discretized differential equation by Jacobi's method, they could be organized so that each processor corresponds to a node of the grid and is able to exchange data with its nearest row and column neighbors. The communications requirement is the most critical issue in the design of microprocessor networks. The expense of any completely general high performance interconnection scheme, such as direct connection of every processor to every other, becomes prohibitive as the number of processors increases. An inadequate interconnection scheme on the other hand either limits performance of the network or limits the class of problems which can be successfully tackled. A detailed discussion of interconnection schemes may be found in reference 2.

In practice, three different approaches to this problem have been used. They are

1. P processors, M memories, and a PxM electronic crossbar switch allowing any processor to access any memory.

2. P processors each with a local memory and all processors connected to a bus structure allowing transfer of data.

3. P processors each with a local memory and arranged into some form of regular lattice. Each processor is then permanently connected to a small subset of the others (usually its neighbors in the lattice).

Figures 1, 2, and 3 show examples of these organizations. Most microprocessor networks do not conform exactly to one of these but add to the basic structure based on application-specific requirements.

Within the basic structure, it is possible to arrange for all processors to be executing the same instruction at the same time in the sense of the Illiac IV, known as Single Instruction Multiple Data (SIMD), or for each processor to be working on its own program and then synchronizing with other processors as necessary, known as Multiple Instruction Multiple Data (MIMD). The choice of which scheme to use is very involved since each has its advantages and the intended application must influence the decision.

NAVIER-STOKES COMPUTER

A study by the RAND Corporation (ref. 3) has proposed a design for an array of microprocessors intended to be used specifically for the numerical solution of the Navier-Stokes equations on very large problems which cannot presently be tackled in realistic running times.

The proposed number of processors is 10 000 arranged as a 100x100 square and operating in the SIMD mode. Each processor is connected to its four nearest neighbors only. Processors on the edges are considered as neighbors of appropriate edge processors on the opposite side, forming what amounts to a "wrap-around" connection. This simple communication scheme reduces the complexity of the hardware and has been shown to be feasible for the intended application (ref. 4). The organization of the individual processors is unusual since a single integrated circuit microprocessor is not used. The intended application does not require all of the facilities of a microprocessor and so the individual processors will consist of an adder, some registers, and a small amount of memory. These may be combined in a single special integrated circuit or constructed from a small number of commercially available components. Since no sequencing capability is included, control information will be broadcast to all of the processing elements from a central control unit. If necessary, individual processors can be selected by a set of row and column lines. The extreme simplicity of the processors means that individual arithmetic operations may be quite slow, but since 10 000 will be performed in parallel, the

overall performance will be substantial.

The amount of memory used is determined by the intended application. It is proposed to solve problems in three dimensions by using the array on a series of two-dimensional planes in sequence. Thus, data for one grid point from each plane have to be stored by each processor, and it is necessary to maintain several variables for each point. The memory will be large enough to ensure that no data transfers to or from peripheral equipment are necessary during problem solution.

The performance expected on the application of interest is very great. Assuming a 32-bit word, 500 nanosecond add time, and a total grid size of between 1/2 million and 10 million points, the solution time is predicted to be between two and three orders of magnitude faster than for a general purpose computer of the CDC 7600 class.

An interesting proposal in this design is the incorporation of a program accessible light-emitting diode (LED) on each processor. The purpose is to allow visual examination of certain aspects of the solution as it proceeds by arranging for each processor to switch on or off its LED based on local solution characteristics.

WISPAC

A group in the Department of Electrical and Computer Engineering at the University of Wisconsin has proposed an MIMD array for the solution of a wide range of partial differential equations (ref. 5). As with the previous design, the basic idea is to have computing power associated with each node of the discretized equations; however, the Wisconsin Parallel Array Computer (WISPAC) is considerably more general in its design. Figure 4 shows the overall structure.

WISPAC consists of a three-dimensional array of as many as 100x100x20 microcomputers, each connected to its six nearest neighbors and with edge nodes making "wrap-around" connections. The array is logically subdivided into sectors with from 5x5x5 to 25x25x20 nodes per sector. All of the nodes in a given sector are connected to a sector control computer, and all of these are connected to a master processor. A sector control computer is primarily responsible for overall program control, control of communication among microcomputers, and input/output to the outside through the master processor.

The microcomputer at each node of the array contains a full microprocessor and local data and program memory, making it possible for each processor to execute different instructions simultaneously. This also alleviates the possible bottleneck of a single controller attempting to distribute instructions to the entire array.

An interesting feature of the array design is an intra-array communication scheme in addition to the six nearest neighbor connections. Each processor can

function as a switch by accepting data from any one of its six nearest neighbors and passing the data on to any other. The logic for handling this communication could be established for one particular problem by the sector control computer, or it could be changed during the solution of a problem by the individual node computers. This capability greatly increases the set of algorithms that one might expect to run efficiently on the array. Communication between non-neighboring nodes using this mechanism does represent an overhead which increases significantly if data must be moved over long distances or if a large number of nodes must communicate over short distances. In addition, for nonuniform problems, selection of efficient communication paths is a nontrivial problem. Reducing this overhead is perhaps the key to the effectiveness of the design.

DISTRIBUTED ARRAY PROCESSOR

International Computers Limited (ICL) has developed a machine called the Distributed Array Processor (DAP) (ref. 6) which is currently being marketed. The DAP consists of a large number of processors which are organized as a square array, with each processor connected to its four nearest neighbors. It is an SIMD machine and there is a control unit which broadcasts instructions to the processors in the array. Each processor contains an enable/disable bit which can be set under program control and determines whether that processor will execute the current instruction or no operation.

The processors are unusual in that they operate on single bits. A processor consists of a 1-bit adder, three 1-bit registers, and typically, a 4096 bit memory. The memories of all the processors taken together also constitute part of the memory of a conventional computer known as the host. This sharing of memory allows for very effective communication between the two systems and permits the host to operate on the data with no data transmission, for example, to set up the initial problem and to extract results. The host and the DAP may operate concurrently on separate tasks provided the host does not use the part of its memory which is shared. For example, the host may be compiling a program to be run on the DAP while the DAP is executing a different program.

As well as being able to communicate with its four nearest neighbors, each DAP processor is also connected to row and column "highways". These are data paths which allow data to be shared by all the processors in a row or column.

Since each processor can only perform single bit addition, floating point operations have to be explicitly programmed in terms of 1-bit operations. Multi-bit arithmetic can either be performed by one processor in bit serial mode or by a group of processors in parallel. In the first case, all of the bits of an operand are stored in one processor's memory, and in the second case, one bit from an operand is stored in each processor's memory. The precision of arithmetic operations and all the details of the arithmetic algorithm are therefore under program control. Thus, if a 29-bit floating

point format is most appropriate for a particular problem, it can be used and will be more efficient in both speed and use of memory than a longer format. As well as flexibility in determining arithmetic, the bit level operations give the DAP considerable diversity in problem solving. For example, the maximum element of an array which has been stored with all of the bits of an element in one memory can be found by a bit level algorithm whose time is proportional to the number of bits in the number format, not the size of the array.

The manufacturer reports (ref. 7) the construction of a prototype DAP with 1024 processors organized as a 32x32 square. Each is equipped with 1024 bits of storage. The time reported for a matrix multiplication involving 32x32 matrices and using 32-bit floating point numbers was 16 milliseconds. This corresponds to approximately 4 million floating point operations per second. Inversion of such a matrix was found to take 29 milliseconds. Typical production models are expected to be arrays of 32x32, 64x64, or 128x128 processors.

PROJECTIONS

Although microcomputer arrays are in their infancy, it is possible to identify some areas where significant problems must be overcome in order to achieve any degree of success in solving real scientific problems. In this section, we will outline some of these problems and indicate some possible solutions that are currently under consideration.

Processor communication is still the major issue. No present design involving microprocessors has attempted to use the layout shown in figure 1. Some use the bus technique shown in figure 2, and most use a combination of the schemes shown in figures 2 and 3. The difficulty of connecting every processor to every other is that the complexity, and hence the cost, of the switch increases as the square of the number of processors. The 16x16 switch on the Carnegie Mellon University C.mmp computer (ref. 8) proved to be one of the most difficult problems in that design. There have been recent proposals (for example, refs. 9 and 10) which use less than full cross bar switches in a tree structure. This idea reduces complexity at the cost of reduced generality. Decisions must be made regarding the routing of information which may prove to be an expensive overhead.

At the other extreme, although the cost is minimal, the limitations of a nearest neighbor connection are clear, and it may be advantageous to augment this in some way, for example, with a global bus (ref. 11). In the end, experience with some configurations will be needed in order to determine the best approach for a given subset of problems.

Another critical issue that must be faced is that of control of the array. A control processor may be used to broadcast instructions to the array, or the computers in the array may execute their own sequence of instructions. The former approach requires that less hardware be invested at each node but reduces the algorithmic possibilities since any reasonably sized controller could support only a relatively small number of instruction streams. Again,

the particular application should probably be a significant factor in the final design decision.

The question of reliability must be faced also. Although individual integrated circuits may have mean times to failure in the millions of hours, for a 100 000 node array with perhaps ten circuits per node, the failure rate may not be acceptable. Hardware will undoubtedly become more reliable, but array designers may have to supply error detection schemes which reduce the impact of a failure. This might involve switching a standby processor into the array where the failure occurs. The implication of this approach on intra-processor communication is profound.

Providing software for array computers is also a significant problem. Constructing an operating system is probably tractable. One solution would be to simply treat the array as a slave device to a host, allowing relatively routine modifications to the host operating system. The question of programming languages is more difficult. Potential users of array computers would prefer simply to run existing FORTRAN programs and to write new programs in standard FORTRAN. Experience with earlier parallel processors has shown that this usually produces very inefficient use of the array. The problem is that automatic detection and exploitation of parallelism are only marginally successful on real programs. New programming languages or modified versions of existing languages which give ready access to the unique features of the new hardware seem to be required.

The performance of a computer in the solution of a problem depends heavily on the particular algorithm being used. Algorithm development in the recent past has been dominated by the need to make effective use of serial computers. The result has been significant improvements in serial algorithm performance. Unfortunately, efficient algorithms which exploit the capabilities of array processors have received little attention. Considerable development in this area is needed if this class of machines is to reach its full potential.

REFERENCES

1. Peterson, Victor L.: Computational Aerodynamics and the Numerical Aerodynamic Simulation Facility. Proceedings of NASA Ames Research Center Workshop on Future Computer Requirements for Computational Aerodynamics, Oct. 4-6, 1977, pp. 5-30.

2. Sullivan, H.; and Bashkow, T. R.: A Large Scale Homogeneous Fully Distributed Parallel Machine. 4th Annual Symposium on Computer Architecture Proceedings, 1977, pp. 105-117.

3. Gritton, E. C.; et. al.: Feasibility of a Special Purpose Computer To Solve the Navier-Stokes Equations. Report No. R-2183-RC, RAND Corp., 1977.

4. Weiman, C.; and Grosch, C.: Parallel Processing Research in Computer Science: Relevance to the Design of a Navier-Stokes Computer. Proceedings of the 1977 International Conference on Parallel Processing, 1977, pp. 175-182.

5. Cyre, W. R.; et. al.: WISPAC: A Parallel Array Computer for Large-Scale System Simulation. Simulation, vol. 29, no. 5, Nov. 1977, pp. 165-172.

6. Parkinson, D.: Technical Description of the Distributed Array Processor. Document No. AP-2, International Computers, Ltd., 1976.

7. Flanders, P. M.; et. al.: Efficient High Speed Computing With the Distributed Array Processor. Proceedings of the Symposium on High Speed Computer and Algorithm Organization, 1977, pp. 113-128.

8. Wulf, W. A.; and Bell, C. G.: C.mmp - A Multi-Mini-Processor. Proceedings of AFIPS 1972 Fall Joint Computer Conference, 1972, pp. 765-777.

9. Lipovski, G. J.: On a Varistructured Array of Microprocessors. IEEE Trans. Comput., vol. C26, no. 2, Feb. 1977, pp. 125-138.

10. Nutt, G. J.: Memory and Bus Conflict in an Array Processor. IEEE Trans. Comput., vol. C26, no. 6, June 1977, pp. 514-521.

11. Jordan, H. F.: A Special Purpose Architecture for Finite Element Analysis. Report Number 78-9 (Contract NAS1-14101), Institute for Computer Applications in Science and Engineering, Mar. 29, 1978. (Available as NASA CR-158918.)

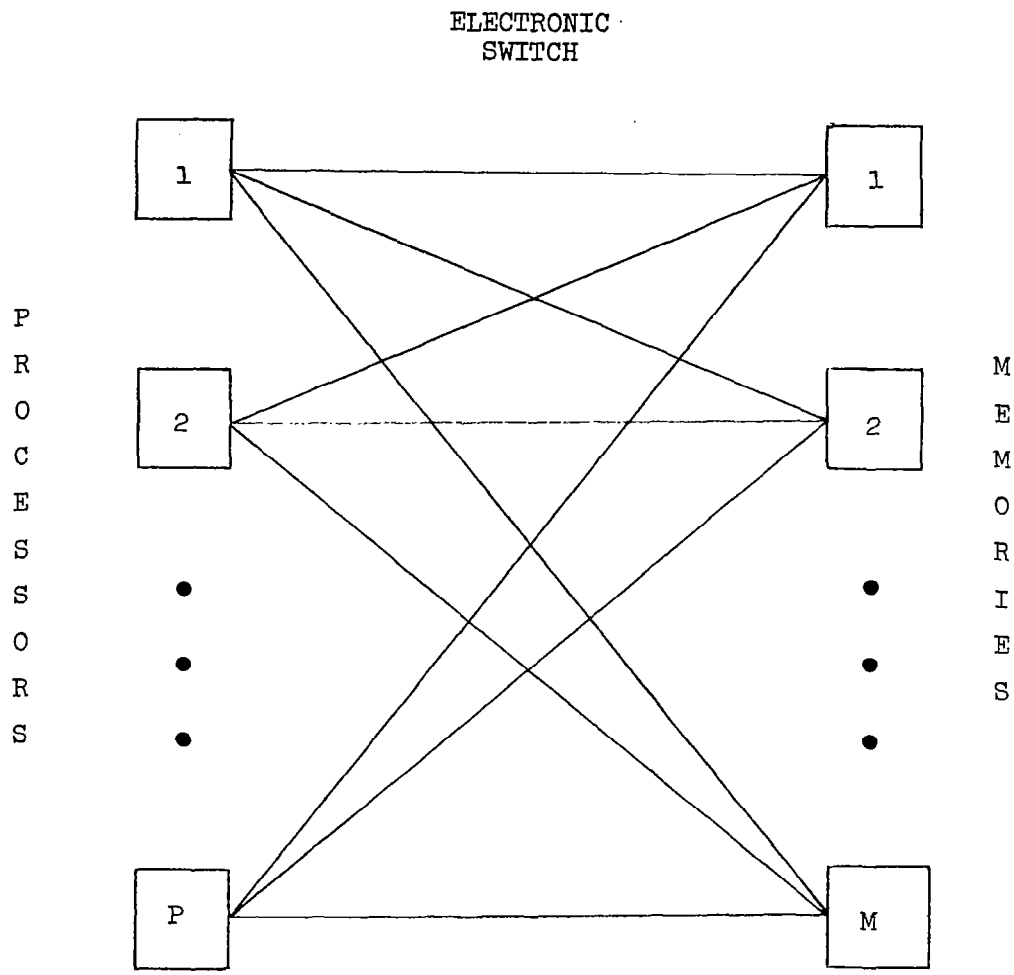

Figure 1.- General connection of processors and memories.

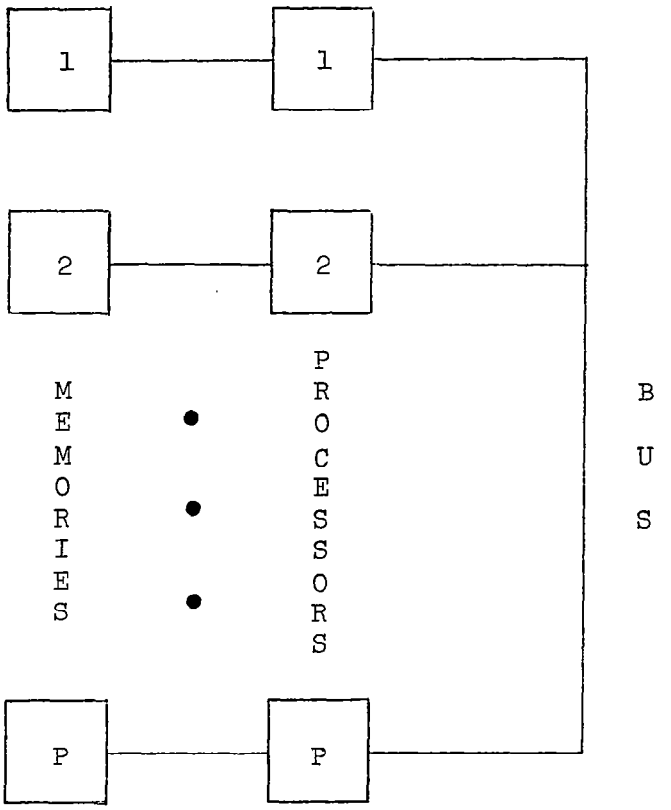

Figure 2.- Processors connected via a bus.

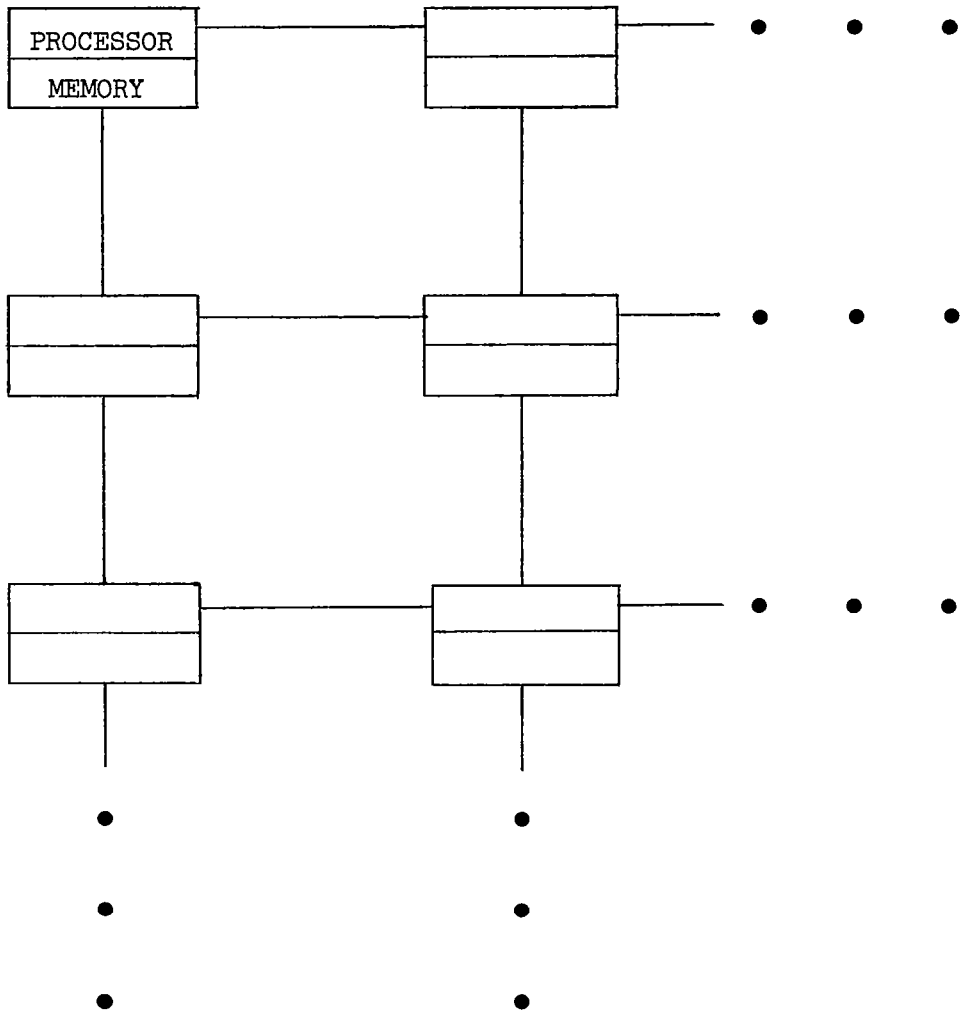

Figure 3.- Processors connected as a lattice.

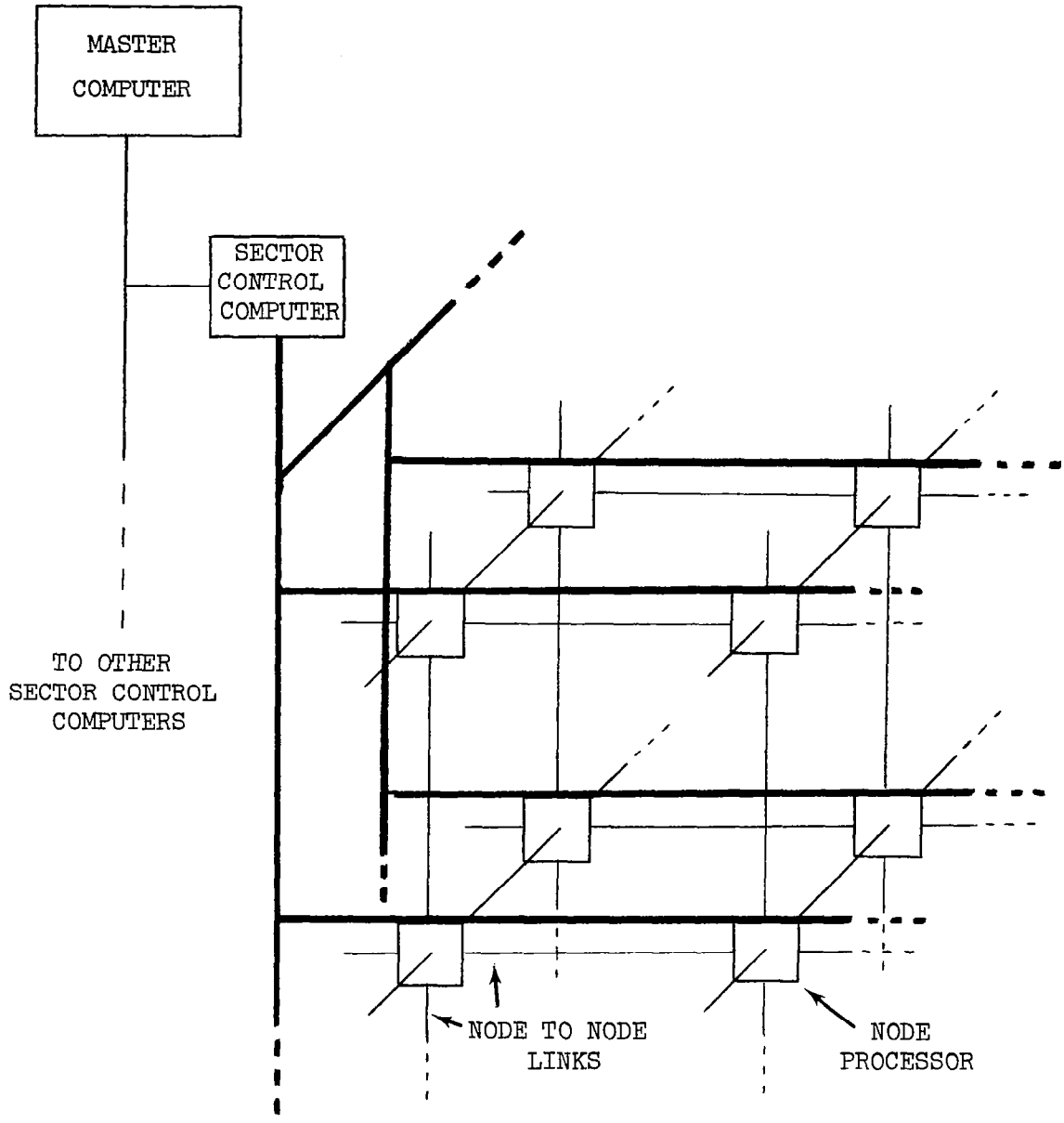

Figure 4.- WISPAC structure.