NASA Contractor Report 172250

NASA-CR-172250
19840004709

ICASE

DESIGN, DEVELOPMENT AND USE OF THE FINITE ELEMENT MACHINE

Loyce M. Adams

and

Robert G. Voigt

Contract Nos. NAS1-17070, NAS1-17130
October 1983

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center, Hampton, Virginia 23665

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

DESIGN, DEVELOPMENT AND USE OF THE FINITE ELEMENT MACHINE

Loyce M. Adams
Institute for Computer Application in Science and Engineering

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

Abstract

In this paper we describe some of the considerations that went into the design of the Finite Element Machine, a research asynchronous parallel computer under development at the NASA Langley Research Center. The present status of the system is also discussed along with some indication of the type of results that have been obtained to date.

Research was supported by the National Aeonautics and Space Administration under NASA Contracts No. NAS1-17070 and No. NASA1-17130 while the authors were in residence at ICASE, NASA Langley Research Center, Hampton, VA 23665.

INTRODUCTION

During the summer of 1976 a weekly seminar was held at ICASE to study developments in parallel computing. The regular participants were Richard Brice, Griffith Hamlin, Harry Jordan, John Knight, David Loendorf, Jerry Tucker, and Robert Voigt with managerial support provided by James Ortega (ICASE) and Robert Fulton (NASA). Prior to that time David Loendorf had begun to investigate ways to speed up the solution of structural analysis problems by introducing parallelism into the finite element process utilizing microprocessor technology. It was therefore natural that the group used problems in structural analysis as a focal point for discussions.

This emphasis on an application area was unique. At that time only two parallel systems were under development: the Illiac IV eventually installed at the NASA Ames Research Center and the C.mmp at Carnegie-Mellon University. Both of these systems were essentially general purpose devices; the Illiac IV was to be used for a variety of large scale scientific problems and the C.mmp was primarily a vehicle for research into a variety of computer science issues arising in parallelism. The group was interested in how an application area might drive a design and whether such a narrow focus might lead to major simplifications in both hardware and software. The influence of the application will be discussed further in the next section.

Another central theme of the discussions was the role of microprocessors. At the time such devices were in their infancy. Simple eight-bit processors were readily available but the more powerful sixteen-bit versions were not. Nevertheless, it was clear that microprocessors were going to grow rapidly in capability, and it was reasonable to consider what could be accomplished by developing a system out of many such devices.

Thus, the activity of the group focused on ways to utilize microprocessors in a system for solving problems in structural analysis via the finite element method. The ideas and concepts developed were organized into an initial hardware design done by Harry Jordan and reported in Jordan [1978]. The eventual manifestation of the design is known as the Finite Element Machine (FEM) and is discussed more thoroughly in Section 3.

The FEM has had a long development period and the way the machine is to be used has undergone numerous changes. Some of the reasons for the extended development time are discussed in Section 4. Section 5 contains a brief discussion of the type of results that have been obtained using FEM and the paper concludes with some observations about developing research computers. Finally, the bibliography contains all work that has been published on FEM as of this writing. Many of these papers are not cited but are included here for completeness.

2. MACHINE DESIGN ISSUES

We will now discuss some of the issues considered by the research group which influenced the design of FEM. In its simplest form the finite element method for the case of static stress analysis may be described as follows:

1. subdivide the region of interest into elements,
2. choose basis functions spanning the space in which the approximate solution lies,
3. integrate the basis functions over each element to determine its contribution,
4. assemble the contributions of all the elements into a single system

$$K\,x = f \tag{2.1}$$

5. solve (2.1) for the approximate solution x.

For more details the reader is referred to the finite element literature, for example, Strang and Fix [1973].

When the above process is implemented on a serial computer the majority of the time is consumed by steps 3 and 5. In addition certain solution techniques for Eq. (2.1) do not require the actual formation of the stiffness matrix K. Thus the activity of the research group focused on steps 3 and 5.

In order to have a focal point for discussion consider the simplified planar structure in Figure 1.

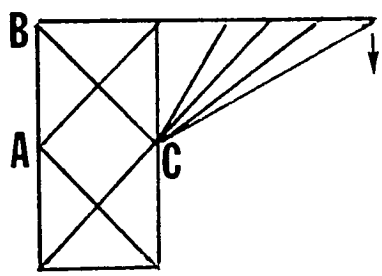

Figure 1. Example Structure

Assume we are interested in determining the stresses in the structure if a force is applied as indicated by the arrow. Further assume that the structure is modeled by different elements such as beams and plates and that an appropriate set of basis functions has been chosen. Then from step 3 the basis functions must be integrated over each element. These integrations may be done in parallel for each element. However since the elements may be different or since similar elements may have different material properties, it is not possible to execute the same instruction sequence across all of the

elements. Thus in order to achieve the maximum degree of parallelism it was considered desirable for FEM to be a parallel system of multiple-instruction-multiple-data (MIMD) type in the classification of Flynn [1966].

For the solution of Eq. (2.1) both direct and iterative methods were considered. For most applications of interest the matrix K is symmetric, positive definite and banded with bandwidth β as indicated in Figure 2.

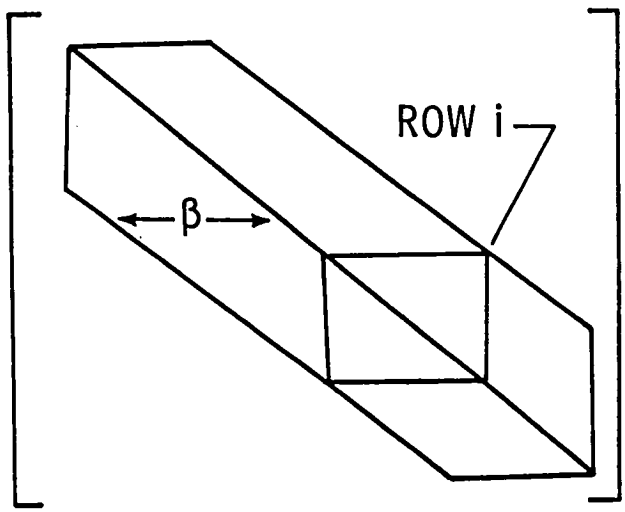

Figure 2. Form of the Stiffness Matrix

In a direct method such as Cholesky factorization, at the i^{th} step conceptually rows $i + 1$ through $i + \beta$ are modified using the pivot row i. It is possible for these modifications to be done in parallel; however, row $i + \beta + 1$ can not be modified until computation on row $i + 1$ has been completed. Thus the degree of parallelism in the sense of Hockney and Jesshope [1981] is limited to β unless one is prepared to consider parallelism at the operation level within each row. The latter is possible but raises serious questions about interprocessor communication for the

element in the i^{th} column of the pivot row must be made available to all processors containing elements of the i^{th} column that are due to be modified by the pivot row.

Finally there is the usual problem of fill associated with direct methods. In general all elements within the band will become non-zero during the factorization. This destroys the sparsity of the matrix and greatly increases the storage requirements.

Iterative methods do not suffer from the fill associated with direct methods. In addition it is easier to obtain a higher degree of parallelism. For example, if we consider the iterative method

$$x^{k+1} = Bx^k + d, \tag{2.2}$$

where x^k represents the approximate solution vector, the degree of parallelism is N, the number of nodes of the discretization. This leads to the concept of a node per processor.

In addition to the increased parallelism this approach also offers the advantage of requiring primarily only local communication. Writing equation (2.2) as

$$x_i^{k+1} = \sum_{j \in I_i} b_{ij} x_j^k + d_i,$$

we see that x_i^{k+1} depends only on a relatively small number of values of x^k as indicated by the index set I_i which consists of those nodes which are physically connected to node x_i. Thus it is desirable to have communication paths between the processor containing x_i and the other processors containing x_j for all $j \in I_i$. Therefore it was decided that each processor should be connected to its eight nearest neighbors in the plane so as to

support the communication required by triangles, an important part of many real structures, see Figure 3.

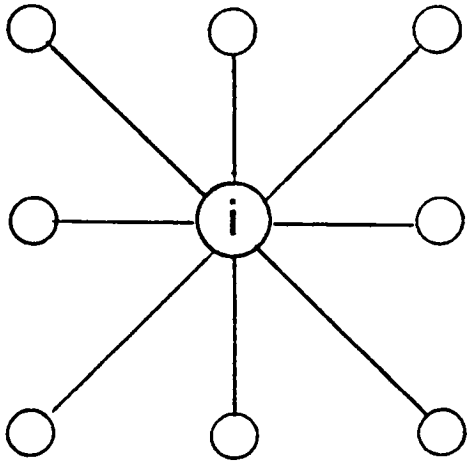

Figure 3. Eight nearest neighbor communication paths for processor i.

It should be noted that the connectivity sets I_i do not all represent the same pattern or number of connections. For example contrast the connectivity of nodes A and B in Figure 1. This means that the computations required for updating each node will not be the same and hence reinforces the requirement that the system be of MIMD type.

Equation (2.2) is the prototype of the classicial Jacobi iteration which exhibits the maximum degree of parallelism but does not have as desirable convergence characteristics as methods like Gauss-Seidel. In Gauss-Seidel like methods, x_i^{k+1} depends on other values at the (k+1) step and thus was not thought of as a parallel method. However, many authors have pointed out that Gauss-Seidel can be turned into a parallel method by employing the so-called red-black or checkerboard ordering, see for example, Ortega and Voigt [1977]. Thus the eight nearest neighbor connection would support the use of modern iterative methods on FEM.

A significant problem remains: one must be able to map the discretized structure of interest onto the processors of the FEM so that all nodes that are connected lie on processors that are connected. This turns out to be a nontrivial problem even if the degree of connectivity of every node is eight or less, see for example, Bokhari [1979]. However many structures contain nodes that are connected to more than eight other nodes as node C in Figure 1. Communication required by such nodes obviously cannot be supported directly by the eight nearest neighbor connections. Thus it was decided to augment the so-called local processor connections with a global bus which provides a connection between any two processors. The work of Bokhari focused on finding mappings of the nodes onto the processors so as to minimize the use of the global bus which was viewed as a resource that could be easily saturated.

At this point the design appeared to hold considerable promise for the classical iterative methods, but there was also interest in studying the more modern accelerations of these methods, as well as the conjugate gradient method and its many variants. A key step in these methods requires parameters which are obtained by computing inner products involving the approximate residual and direction vectors. In the scenario described above the approximate solution is distributed across the processors and it requires $O(n)$ steps to accumulate an inner product using local connections on an $n \times n$ array. To overcome this delay a separate circuit was designed that connects the processors in a classical binary tree. This made it possible to find the maximum element of a vector or to sum the elements of a vector in $O(\log n)$ time when the elements were distributed across the $n \times n$ array. For additional details see Jordan et al. [1979].

At this point the basic concepts of the FEM were fixed and a preliminary design for 1024 processors was done by Harry Jordan (see Jordan [1978]) under support from the Structures Division at the NASA Langley Research Center (LaRC). In 1979 LaRC began fabrication of an experimental system under the leadership of David Loendorf with hardware integration support provided by Frank Mewszel; in 1981 David Loendorf left Langley and Olaf Storaasli assumed responsibility for system development. The prototype presently contains eight processors with expansion continuing; a 36 processor version is shown in Figure 4. The system is discussed in more detail in the next section.

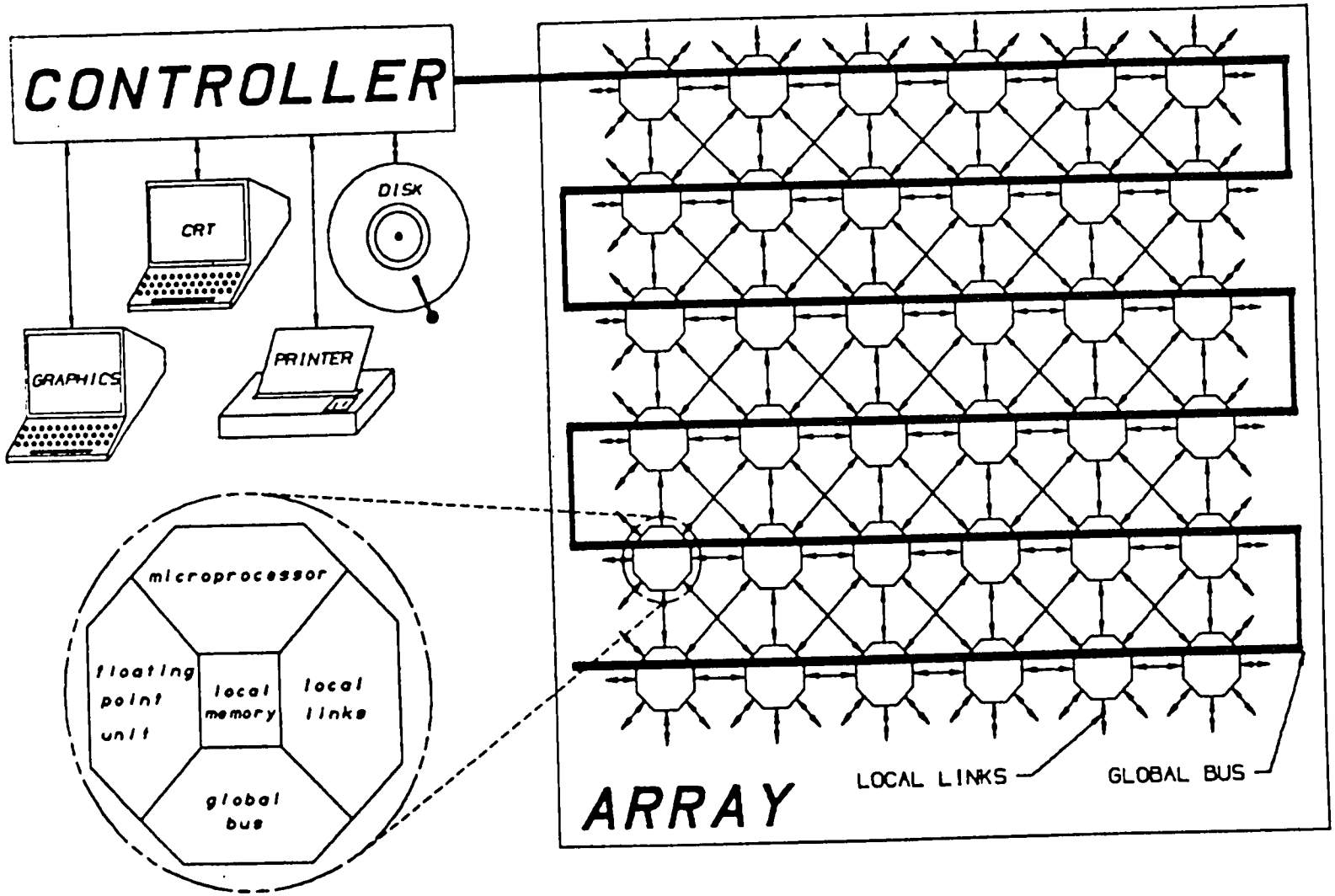

Figure 4. Finite Element Machine System

3. THE CURRENT FINITE ELEMENT MACHINE

In this section we describe the hardware and system software for both the controller and the nodal processors. More detailed descriptions of the hardware may be found in Jordan [1978], Jordan, et al. [1979], and Loendorf [1983]. A summary of the current system software may be found in Storassli, et al. [1982] and detailed descriptions of the controller support software and array software may be found in Knott [1983a] and Crockett [1984] respectively.

Hardware

The controller consists of a TI 990/10 minicomputer with 128K words of memory, four 5-megabyte disk drives, a Kennedy 9000 tape drive, and a line printer. The purposes of the controller are to serve as the user interface to the FEM array by providing program development and problem definition tools, to provide mass storage for programs and nodal processor input and output data, and to initiate and monitor activity on the array.

Each nodal processor in the array is comprised of three hardware boards: the CPU board, the IO-1 board, and the IO-2 board. The CPU board contains a Texas Instrument 9900 16-bit microprocessor, 16K bytes of erasable programmable read only memory (EPROM), 32K bytes of random access memory (RAM), and an Advanced Micro Devices AM9512 floating point chip. The EPROM and 4K of RAM are reserved for system software. The remaining 28K RAM is available for program code, run-time data structures, and input data. The AM9512 floating point chip with a clock frequency of 2 MHz provides single precision (32-bit, 25-bit mantissa) and double precision (64-bit, 57-bit mantissa) add, subtract, multiply, and divide operations. To use this capability, the operands must be loaded by the nodal processors' system software which requires approximately 360 microseconds for two single

precision numbers (this number was obtained through private discussions with Tom Crockett). Once the operands are loaded, a single precision floating point multiply can be performed in approximately 100 microseconds.

The IO-1 board contains twelve local communication links and the summation/maximum hardware. Each link is a 1.5 MHz bit serial interface with an associated hardware FIFO buffer capable of storing 16 16-bit words of input data from a neighboring processor. The links are normally configured in an eight nearest neighbor with torroidal wrap around scheme but may be changed before each program execution to support other strategies. Likewise, an output register holds values that have been transferred from the memory on the CPU board for transmission to neighboring processor(s). The summation/maximum hardware allows p values, one per processor, to be added in $\log_2 p$ time by providing a binary tree structure with the processors initially at the leaves of the tree. This hardware works independent of the other communication networks of the machine and was designed specifically to perform summations (needed by inner products) and determine maximum values (desirable for norm calculations).

The IO-2 board contains the global bus connections, the signal flag networks, and the processor's self-identification tag. The global bus is a 1.25 MHz time-multiplexed 16-bit parallel bus that connects all processors to each other and to the controller. The bus has hardware FIFO buffers on both the input and output lines capable of storing 64 words of data for buffering purposes. The bus serves as the vehicle for transmitting the program code and data from the controller disk to each nodal processor. The bus is also used during the execution of application programs to transmit data between non-neighboring processors with each processor having equal priority for the bus on a first-come, first-serve basis, see Knott and Crockett [1982]. The bus

can be used in the broadcast mode to send information to a set of processors from another processor or the controller. The signal flag hardware connects a processor to eight separate binary flag networks which span all processors. Any or all of these hardware flags can be enabled or disabled during program execution and allow for synchronization and decision making. A processor's physical self-identification number is hardwired on the IO-2 board and is matched to the logical processor number of a particular application code by the system software for use in interprocessor communication and decision making.

At present, eight processors, all connected via local links to each other, have been running application codes. Currently, another eight processor system is being installed providing one system for hardware and any additional software development and one for application users. The next step will be to add eight processors to one system for a 4×4 FEM array. Eventually a larger array may be built if studies performed on the 4×4 array indicate that such an effort is warranted.

System Software

The system software consists of the vendor's standard software for the TI 990 controller, the FEM Array Control Software (FACS) that runs on the TI 990 and provides the user interface to the array, the NODAL executive operating system that runs on each TI 9900 microprocessor in the array, and the PASCAL Library extensions (PASLIB) that support access to the architectural features of the array like communication and synchronization. A short description of each of these software components and how they work together to implement applications programs follow.

The vendor's software for the TI 990 controller includes a screen editor, an assembler, a reverse assembler, a Pascal compiler, and a link editor. The applications programmer uses this software to edit, compile, and link his program to be run on the array. Typically this program will be executed by all the processors in the array with different data. The programmer can use an interactive graphics interface or the text editor to model his problem and partition this data to separate data files (stored on the controller) for each processor in the array. Alternatively, a heuristic utility program may be written to partition the data for the processors, Bokhari [1979].

After the program and data files for each processor have been stored on the 990 disk, the FACS software is used in conjunction with the nodal EXEC operating system on each of the nodal processors to initialize the Array, select the array configuration, define the size of the data areas (memory on the nodal processors that contains either initial data or intermediate data between job runs), load any or all of these data areas from the data files on the controller, and download (broadcast) the program linked code. All these FEM commands are implemented as control language procedures in FACS which is a natural extension of the menu-driven command interpreter of the vendor software. The programmmer must therefore create a command program which describes which program(s) and data are to be down loaded and the appropriate sequence for that downloading and execution. This command program in turn is invoked by a single controller command. After the program begins execution on the array, the controller enters an interactive execute mode and receives all messages/errors from all array processors but displays on the user's terminal information from only one preselected processor. During execution, the FACS software maintains a file of all output data received from the array, errors encountered by all the processors, and a log of the events during the job run

which can be post processed by utility programs at the end of the job session. FACS also provides interactive debugging commands that allow the user to single step, halt, kill, resume, dump memory, set program breakpoints, and inspect and change memory, status, and registers.

The two components of the system software that run on each TI 9900 microprocessor are the NODAL EXEC operating system and the PASLIB routines. NODAL EXEC is stored in EPROM on each TI 9900 and provides interrupt handling, basic I/O, timing, memory allocation, task management, and a command monitor. In addition, NODAL EXEC contains a package of command routines which implement all functions the Controller commands the TI 9900 to perform. Typical functions include loading object code, loading data into the data areas, establishing processor connectivity (local and global neighbors), executing programs, performing debugging, and uploading results.

Perhaps of more interest to the applications programmer are the PASLIB routines. PASLIB is a library of Pascal subroutines that allow the programmer to use the local links, global bus, signal flag network, sum/max circuit, and the AM9512 floating point unit as well as communicate with the controller. The most commonly used routines are written in assembly language and stored in EPROM. A few of these will now be described.

To synchronize using flag i, processors must first call the ENABLE (flag i) routine to add this flag to the network, after which a call to the BARRIER (flag i) routine will cause all processors with flag i enabled to synchronize. Note that these routines must be called by all processors wishing to synchronize. The BARRIER routine may be used in iterative algorithms to synchronize before a call to the ALL (flag i) routine is executed to check for global convergence.

To send n words of data that are stored in memory starting at location ℓ to processor p the programmer would call the PASLIB routine SEND(p,ℓ,n) or SEND2(p,ℓ,i,n) if data is distinguished by an index tag i. Data may also be broadcast to all local and global neighbors by SENDALL(ℓ,n) or SENDAL2(ℓ,i,n).

Data is received from another processor in either a synchronous or asynchronous mode (which has to be defined by the programmer in the command file on the controller). For the synchronous mode, input from the sending processor is queued in the order it is received and must be read by the receiver in this order. For the asynchronous mode, only the most recently received record (for each index tag) is saved. By providing these two modes of communication, the system software must necessarily be more general and therefore more expensive; however, they provide a mechanism for studying both synchronous and asynchronous algorithms.

To use the AM9512 floating point unit, the operands must be loaded via PASLIB routines. For example, to multiply two numbers x and y and store in z, the appropriate statement would be z := MULT(x,y). This adds a cost of 358 μs to execute the MULT procedure compared to the 99 μs for actually performing the multiply on the AM9512. (This tremendous overhead is due to the incompatibility of the AM9512 and the TI 9900 that could not be avoided at the time the hardware selections were made.)

4. FEM DEVELOPMENT EXPERIENCES

FEM development to date has provided a number of learning experiences which may be useful to share. Progress has been slower than anticipated for a variety of reasons involving both hardware and software issues.

At the time the microprocessor was selected, the TI 9900 was the only 16-bit processor available. As development progressed a number of unexpected small hardware purchases were required. Significant delays in procurement were encountered due both to delay in manufacturer delivery and to federal procurement policies. In hindsight the low-bidder competitive procurement process of the government was often not the most effective strategy to purchase small quantities of scarce parts to meet the requirements of an evolving research system. Any cost benefits from competitive procurements were negated by delays in system development incurred while waiting for deliveries. A better strategy might have been a master contract for all parts with the specifics to be determined as work progressed.

The design itself required the usual modifications but a serious weakness was the omission of any hardware error detection. The latter situation caused significant delays in the debugging process that was already complicated by the presence of several processors functioning independently.

As with most research projects funding was limited and staffing levels were barely adequate to encompass the hardware, software, numerical analysis and applications disciplines. Furthermore when some initial hardware became operational, it was difficult to satisfy the needs of both those doing hardware enhancement and those doing systems/applications development -- activities equally important for such research projects. The competition for access was finally resolved by establishing a dual system, presently consisting of eight processors for hardware development and another eight for software.

Not surprisingly there were also difficulties in the software development. The original idea of choosing a controller with the same instruction set as the processors in the array and thereby using that software as a basis for the

array software seemed sound. However, the software underwent such signficant changes that it might have been better to develop an all new software system. Major issues revolved around the adaptation of Pascal to the operating environment of the array. For example floating point arithmetic had to be adapted to account for the presence of the AM9512. This involved facilities for moving data to that device as well as converting the data to the appropriate format.

Perhaps the biggest issue was to provide support for the variety of communication mechanisms available. Since this is a major issue in any parallel computing system we will discuss it in more detail below. It should be noted first however that the fact that the FEM software provides an effective environment for the user is a credit to the efforts of Tom Crockett and Judson Knott. Their task was further complicated by changes in the way users expected to utilize the array, an example of which follows.

As discussed earlier, the original concept of the FEM involved considering a node, or possibly an element, of the discretization per processor. The implementation of a standard iterative method would then require frequent communication in which a processor would send a value to each of its neighbors and receive a value from each. As it turned out such frequent bursts of communication involving only a few words of data were inefficient.

This inefficiency can be better understood by considering the steps a processor must complete to actually send data:

1. interrupt the processor,
2. copy the data to be sent to an output buffer,
3. place the output buffer in a queue for either local or global transmission,
4. generate a send interrupt,

5. execute the send interrupt,

6. transmit the data.

Executing these steps in order to send two words of data requires approximately 1.8 milliseconds with the actual transmission contributing only a few microseconds. When one compares this time with the approximately .46 milliseconds required by a floating point operation, one is led to try to organize the computation so that the frequency of transmission is decreased while the amount of data per transmission is increased.

One way to accomplish this change in transmission style is to place several nodes or elements in each processor. In fact, note that if a two-dimensional region is stored in each processor then for many iterative methods only the boundary data must be exchanged between the processors. If the region is q×q then the amount of data transmitted is $O(4q)$ while the computation per processor is $O(q^2)$. This provides a mechanism for balancing the communication time with the computation time. However the array is now being used differently than was originally intended further complicating the software development.

5. USE OF THE FEM

The results obtained on FEM so far can be put into two separate categories; namely, parameter results and numerical algorithm results especially appropriate for static stress analysis.

Tom Crockett and Judd Knott have timed the system software routines that send and receive data and perform synchronization using the flag network. These parameters are given in Adams [1982] and were used there as input to a model to predict performance of some numerical algorithms as the number of

processors increased. Smith and Loendorf [1982] have also obtained FEM parameters for use in performance evaluation. For the most recent version of the operating system software, Knott [1983b] has timed the PASLIB routines that send and receive a package of numbers between neighbor processors as a function of the package size. The results of his study have given guidance to application programmers as to the tradeoffs in package size selection.

Execution times of several iterative algorithms for fixed sized problems with the number of processors varying from one to five have been obtained. These times were used to compare the algorithms as a function of the number of processors and to obtain speedup results for a given algorithm. The iterative algorithms include multi-color SOR as described in Adams and Ortega [1982] with results reported in Adams [1982], conjugate gradient (Adams [1982]), and m-step preconditioned conjugate gradient (Adams [1983a] and [1983b]). In addition, execution time and speedup results for assembling the finite element stiffness matrix K of (2.1) in parallel is given in Adams [1982].

Additional studies are underway to determine execution time and speed up results for a variety of algorithms and applications. Direct methods for solving the system (2.1) are being compared particularly for multiple right hand sides. Direct and iterative methods for tridiagonal systems are also being compared. This problem is particularly important as a building block for other more general iterative methods.

A new application under study involves computing the nonlinear dynamic response of a structure under a prescribed load condition. The technique is highly parallel and preliminary results indicate that excellent speedups are possible.

6. CONCLUSIONS

The FEM is important as a model of the kind of research system that is required to understand the issues in parallel computing and to evaluate various techniques for making such systems useful. While the development of the FEM took longer than anticipated, a great deal has been learned during that process. A collection of researchers have been able to evaluate techniques for programming MIMD systems. They have developed an understanding of how to use communication among the processors and have actually measured the performance of that communication in solving real problems. They are beginning to learn just how hard it can be to debug a program for a system of asynchronous processors. This kind of experience may lead to improved techniques for tracking and isolating errors in MIMD systems.

There is no scarcity of ideas in the computer science community about how one should design an MIMD system. There is a scarcity of systems on which to evaluate these ideas. Such systems are absolutely essential if we are to develop the knowledge required to produce useful MIMD systems. Despite some shortcomings, the FEM is an excellent example of how important such a test bed can be.

Acknowledgement

The authors are indebted to Robert Fulton for many useful discussions and for suggestions for improving the manuscript.

REFERENCES

[1] Adams, G., D. Mullens, and A. Shah [1979], "Software Design Project: Finite Element System Executive," Report No. CSGD-79-1, Computer Systems Design Group, Univeristy of Colorado.

[2] Adams, L. M. [1982], "Iterative Algorithms for Large Sparse Linear Systems on Parallel Computers, Ph.D. Thesis, University of Viriginia; also published as NASA CR-166027, NASA Langley Research Center, November.

[3] Adams, L. M. [1983a], "M-Step Preconditioned Conjugate Gradient Methods," NASA CR-172130, NASA Langley Research Center, Hampton, VA, April.

[4] Adams, L. M. [1983b], "An M-Step Preconditioned Conjugate Gradient Method for Parallel Computation," Proc. of the 1983 Intl. Conference on Parallel Processing, IEEE Catalog No. 83CH1922-4, August, pp. 36-43.

[5] Adams, L. M. and J. M. Ortega [1982], "A Multi-Color SOR Method for Parallel Computation," Proc. of the 1982 Intl. Conference on Parallel Processing, IEEE Catalog No. 82CH1794-7, August, pp. 53-56.

[6] Bokhari, S. H. [1979], "On the Mapping Problem for the Finite Element Machine, Proc. of the 1979 Intl. Conference on Parallel Processing, IEEE Catalog No. 79CH1433-2C, August, pp. 239-248.

[7] Bokhari, S. H. [1981], "On the Mapping Problem," IEEE Trans. Computers, Vol. C-30, March, pp. 207-214.

[8] Bokhari, S. H. [1981], "MAX: An Algorithm for Finding Maximum in an Array Processor with a Global Bus," Proc. of the 1981 Intl. Conference on Parallel Processing, IEEE Catalog No. 81CH1634-5, August, pp. 302-303.

[9] Bokhari, S. H. and A. D. Raza [1983], "Reducing the Diameters of Array," Report No. EECE-83-01, Department of Electrical Engineering, University of Engineering and Technology, Lahore, Pakistan, June.

[10] Bostic, S. W. [1983], "TEKLIB Graphics Library," NASA TM-84633, NASA Langley Research Center, Hampton, VA, March.

[11] Crockett, T. W. [1984], PASLIB Programmer's Guide for the Finite Element Machine, preliminary draft, to appear as NASA Contractor Report, NASA Langley Research Center, Hampton, VA.

[12] Chughtai, M. Ashraf [1982], "Complete Binary Spanning Trees of the Eight Nearest Neighbor Array," Report No. EECE-82-01, Department of Electrical Engineering, University of Engineering and Technology, Lahore, Pakistan, November.

[13] Gannon, D. [1980], "A Note on Pipelining a Mesh Connected Multiprocesor for Finite Element Problems by Nested Dissection," Proc. of the 1980 Intl. Conference on Parallel Processing, IEEE Catalog No. 80CH1569-3, August, pp. 197-204.

[14] Hockney, R. W. and C. R. Jesshope [1981], "Parallel Computers," Adam Hilger Ltd., Bristol, Great Britain.

[15] Iqbal, M. Ashraf and S. H. Bokhari [1983], "New Heuristics for the Mapping Problem," Report No. EECE-83-02, Department of Electrical Engineering, University of Engineering and Technology, Lahore, Pakistan, June.

[16] Jordan, H. F. [1978], "A Special Purpose Architecture for Finite Element Analysis," Proc. of the 1978 Intl. Conference on Parallel Processing, IEEE Catalog No. 78CH1321-9C, August, pp. 263-266.

[17] Jordan, H. F., ed. [1979], "The Finite Element Machine Programmer's Reference Manual," Report No. CSDG-79-2, Computer Systems Design Group, University of Colorado, Boulder, CO.

[18] Jordan, H. F. and D. A. Podsiadlo [1980], "A Conjugate Gradient Program for the Finite Element Machine," Computer Systems Design Group, University of Colorado, Boulder, CO.

[19] Jordan, H. F. and D. A. Podsiadlo [1981], "Operating Systems Support for the Finite Element Machine," Report No. CSDG-81-2, Computer Systems Design Group, University of Colorado, Boulder, CO.

[20] Jordan, H. F. and P. L. Sawyer [1978], "A Multi-Microprocessor System for Finite Element Structural Analysis," Trends in Computerized Structural Analysis and Synthesis, A. K. Noor and H. G. McComb, Jr., eds., Pergamon Press, Oxford, pp. 21-29.

[21] Jordan, H. F., M. Scalabrin, and W. Calvert [1979], "A Comparison of Three Types of Multiprocessor Algorithms," Proc. of the 1979 Intl. Conference on Parallel Processing, IEEE Catalog No. 79CH1433-2C, August, pp. 231-238.

[22] Knott, J. D. [1983a], "FEM Array Control Software User's Guide," NASA CR 172189, NASA Langley Research Center, Hampton, VA.

[23] Knott, J. D. [1983b], "A Performance Analysis of the PASLIB Version 2.1X SEND and RECV Routines on the Finite Element Machine," NASA CR 172205, NASA Langley Research Center, Hampton, VA.

[24] Knott, J. D. and T. W. Crockett [1982], "Fair Dynamic Arbitration for a Multiprocessor Communication Bus," Computer Architecture News, Vol. 10, No. 5, September, pp. 4-9.

[25] Loendorf, D. [1983], "Advanced Computer Architecture for Engineering Analysis and Design," Ph.D. Thesis, University of Michigan, Aerospace Engineering Department.

[26] McBride, W. E. [1980], "Simulations Solution Algorithms for the FEM," Modeling and Simulation, Vol. 11, Part 2, Proc. of the 11th Annual Pittsburgh Conference on Modeling and Simulation, pp. 595-599.

[27] McBride, W. E. [1981], "Simulating the PAM (Purely Asynchronous Method) on the FEM (Finite Element Machine)," Modeling and Simulation, Vol. 12, Proc. of the 12th Annual Pittsburgh Conference on Modeling and Simulation, pp. 413-416.

[28] Mehrotra, P. and T. W. Pratt [1982], "Language Concepts for Distributed Processing of Large Arrays," ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, August, pp. 19-28.

[29] Mehrotra, P. [1982], "Parallel Computation on Large Arrays," Ph.D. Thesis, University of Virginia.

[30] Ortega, J. and R. Voigt [1977], "Solutions of Partial Differential Equations on Vector Computers," Proc. 1977 Army Numerical Analysis Conference, pp. 475-526.

[31] Podsiadlo, D. A. [1981], "An Operating System for the Finite Element Machine," Report No. CSDS-81-3, Computer Systems Design Group, University of Colorado, Boulder, CO.

[32] Rea, G. C. [1983], "A Software Debugging Aid for the Finite Element Machine," Report No. CSDG-83-2, Computer Systems Design Group, University of Colorado, Boulder, CO.

[33] Shah, A. K. [1980], "'Group Broadcast' Mode of Interprocessor Communication for the Finite Element Machine," Report No. CSDG-80-1, Computer Systems Design Group, University of Colorado, Boulder, CO.

[34] Smith, C. and D. D. Loendorf [1982], "Performance Analysis of Software for an MIMD Computer," Performance Evaluation Review, Vol. 11, No. 4, Proc. of the 1982 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, pp. 151-162.

[35] Storaasli, O. O., S. W. Peebles, T. W. Crockett, J. D. Knott, and L. M. Adams [1982], "The Finite Element Machine: An Experiment in Parallel Processing," NASA TM-84514, NASA Langley Research Center, July.

[36] Voitus, R. F. [1981], "MPSP: A Multiple Process Software Package for the Finite Element Machine," Report No. CSDG-81-4, Computer Systems Design Group, University of Colorado, Boulder, CO.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-172250		

4. Title and Subtitle	5. Report Date
Design, Development and Use of the Finite Element Machine	October 1983
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Loyce M. Adams and Robert G. Voigt	83-56
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center	NAS1-17070, NAS1-17130
	13. Type of Report and Period Covered Contractor Report
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor: Robert H. Tolson
Final Report

16. Abstract

In this paper we describe some of the considerations that went into the design of the Finite Element Machine, a research asynchronous parallel computer under development at the NASA Langley Research Center. The present status of the system is also discussed along with some indication of the type of results that have been obtained to date.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
parallel processing finite element method	62 Computer Systems 64 Numerical Analysis Unclassified-Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	28	A03