NEW TECHNOLOGY, INC.

4811 Bradford Boulevard
Huntsville, Alabama 35805

FR1021
12 October 1984

Final Report

ATMOSPHERIC MODELING AND SENSOR SIMULATION (AMASS) STUDY

Contract NAS8-35189

Prepared for:
National Aeronautics and Space Administration
George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

This final report provides a summary of the tasks performed on Contract NAS8-35189 for the Atmospheric Sciences Division of the System Dynamics Laboratory of the Marshall Space Flight Center.

1.0 BACKGROUND

New Technology, Inc. (NTI) began this 12-month study with the goal of enhancing the capabilities of the Atmospheric Modeling and Sensor Simulation (AMASS) system. This system is used in processing atmospheric measurements which are utilized in the evaluation of sensor performance, conducting design-concept simulation studies, and also in the modeling of the physical and dynamical nature of atmospheric processes. These results may then be evaluated in order to furnish inputs into the final design specifications for new space sensors intended for future Spacelab, Space Platform and free-flying missions. In addition, data gathered from these missions may subsequently be analyzed to provide a better understanding of the requirements for atmospheric modeling.

The following study tasks were proposed in order to both enhance the AMASS system utilization and attempt to integrate the AMASS system with other existing equipment to facilitate the analysis of data for modeling and image processing.

1.1 COMMON GRAPHICS PACKAGE

Perform a study to determine the feasibility of establishing a common graphics package (e.g., NCAR graphics package) for the systems used in atmospheric modeling and sensor simulation studies. This commonality of software could lead to significant simplification in setting up modeling software as well as possibly reducing the need to transfer data files between systems to access graphics software.

1.2 RJE METHODOLOGY

Investigate the possibility of connecting the systems together via RJE methodology in order to provide better utilization of computer resources available to the scientific user community in a cost-effective manner. Use of RJE techniques is preferred over interactive terminals for several reasons,

including reducing the need to learn extensive JCL and allowing the convenience of batch mode for large modeling programs.

1.3 PROTOTYPE DEVELOPMENT

Determine techniques which might improve system performance, select the most promising, specify required material for development of the prototype, make recommendations, and subsequently test prototypes to assure feasibility of use in a normal operating environment.

1.4 WRITEABLE CONTROL STORE

Since the sensor area of image processing involves gathering data from sensor simulation of mosaic arrays as well as the analysis of real data which actually is transmitted from the sensor, both rapid CPU speed and high bandwidth input/output (I/O) are of the utmost importance. Therefore, a study was recommended to determine benefits which may be derived by coding selected portions of software into Writeable Control Store (WCS) in an attempt to decrease program execution time.

1.5 HARDWARE OPTIMIZATION

It was proposed that a study be undertaken in order to improve execution performance of atmospheric models. Since raw computer speed is the driving factor in this environment, an evaluation of the means in which hardware optimization could be accomplished would seem most desirable. A prime area of interest here was the evaluation of individual modeling programs to determine where the areas of intense execution concentration reside and resolution of techniques for improving general system performance.

1.6 ARRAY PROCESSOR

In a similar manner as described in Section 1.5, it appeared that the addition of an array processor to the AMASS might permit a significant improvement in the execution speed of models. Selection of a candidate program and use of it as a benchmark to run on a similarly configured system equipped with an array processor could provide valuable comparative results for the evaluation followed by appropriate recommendation based on that study.

1.7 LINK TO HIGH-SPEED COMPUTER FACILITIES

A study was recommended to determine how the AMASS could be linked to other existing or projected NASA high-speed computer facilities (e.g., Class 6 system proposed for MSFC). Important factors for consideration included communications, ease of operation from the standpoint of the user scientist, and providing as much transparency to the user as possible to minimize the necessity of learning extensive job control languages and utilities.

2.0 SUMMARY OF ACCOMPLISHMENTS

The following subsections reflect status of each of the individual work tasks. Figure 1 shows the milestone chart for all tasks. Due to the lengthy period required to procure equipment, work on hardware optimization and the array processor was deferred until items arrived. Attention was directed instead toward RJE methodology and the high-speed computer link.

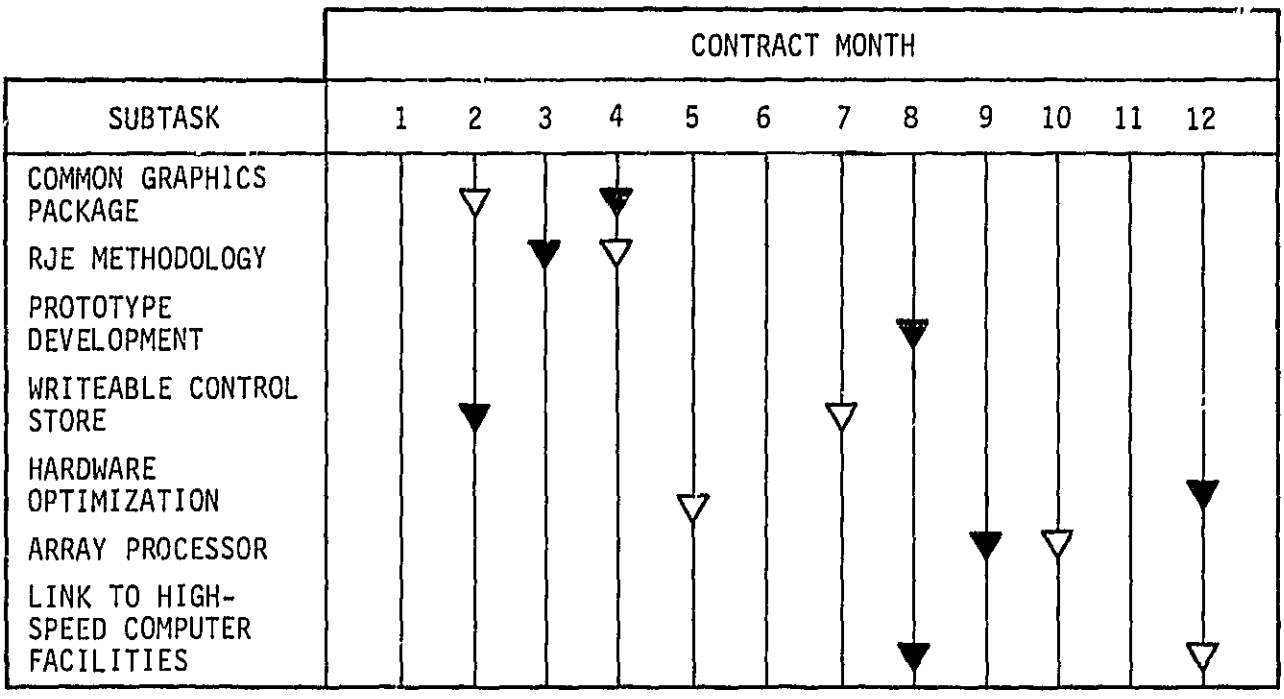

Fig. 1 Milestones

2.1 COMMON GRAPHICS PACKAGE

The National Center for Atmospheric Research (NCAR) graphics package was implemented and checked out on the AMASS system. It was determined that capability to support FR-80 microfiche output could also be added relatively easy. Although the NCAR code used at the Goddard NASA High Speed Computing Facility (NHSCF) differs somewhat from the version at MSFC, the basic features of their package offer substantial compatibility. A Tektronix 4115B color graphics terminal was installed and checked out on the AMASS system. The existing NCAR metacode translator presented no problems.

2.2 REMOTE JOB ENTRY (RJE) METHODOLOGY

The possibility of connecting to the Class 6 Cyber 205 at the NHSFC was investigated. Protocols evaluated included RJE 2780/3780 and HASP emulator packages. HASP was selected due to its versatility and compatibility with RJE software at Goddard. Equipment necessary for accomplishing the connection was specified and ordered as prototype items. The system was reconfigured, and the connection established and checked out.

A user meeting at NHSFC was attended by K. Parker of NTI and L. MacLean of MSFC. Other remote users included JPL, MIT, University of Wisconsin and University of Maryland. This meeting provided an opportunity for remote users to alert NHSFC to their problems as well as offer a chance for users to exchange information.

2.3 PROTOTYPE DEVELOPMENT

The RJE/HASP interface described in the previous section was developed in this manner to determine if the interface would perform satisfactorily. The following items were procured under this contract:

Selector Channel M32-010
Quad Synchronous Adapter M47-002
Line Conditioning Module M47-005
Cable (LCM to Modem) 25 ft M47-007
HASP Emulator S70-015-ABC
UDS 208/AB Modem

2.4 WRITEABLE CONTROL STORE

The Writeable Control Store with associated FORTRAN enhancement software package was installed on the AMASS. Numerous benchmarks were run to evaluate the relative improvement in execution speed over the standard configuration.

2.5 ARRAY PROCESSOR

It immediately became obvious that the CPU power offered by the AMASS would not satisfy requirements of running multiple modeling/simulation programs simultaneously. A study was undertaken (see attachment) of off-the-shelf AP's available which would offer standard interface to the Perkin-Elmer 3250 with reasonable cost ($70 - $90k). During this time, an FPS AP-120B array processor was borrowed from another lab for feasibility studies. It was determined that although the currently available AP's meeting these specifications did not offer much more than the borrowed AP, at least three vendors indicated cost-competitive, standard-interface units would be available within months. The recommendation was made to continue experimentation with the AP-120B and keep monitoring the AP market.

2.6 SOFTWARE OPTIMIZATION

Modifications were made to the P-E operating system in an attempt to speed up the overall system performance. The multiterminal monitor system was modified to disable the heuristic scheduling algorithm. This allows manual modification of the time slice granted each user. This becomes more important as the job mix changes from CPU-bound to I/O-bound. A recommendation was made to obtain the FORTRAN Z compiler to provide universal optimization capability. This software was received; however, the upgrade is not compatible with the existing revision level. The correct revision should arrive when the P-E extended software maintenance contract goes into effect.

2.7 HARDWARE OPTIMIZATION

The original configuration of the AMASS is shown in Fig. 2. Figure 3 depicts the current hardware configuration. The dotted lines indicate items under procurement. The following equipment was recommended and subsequently procured and installed:

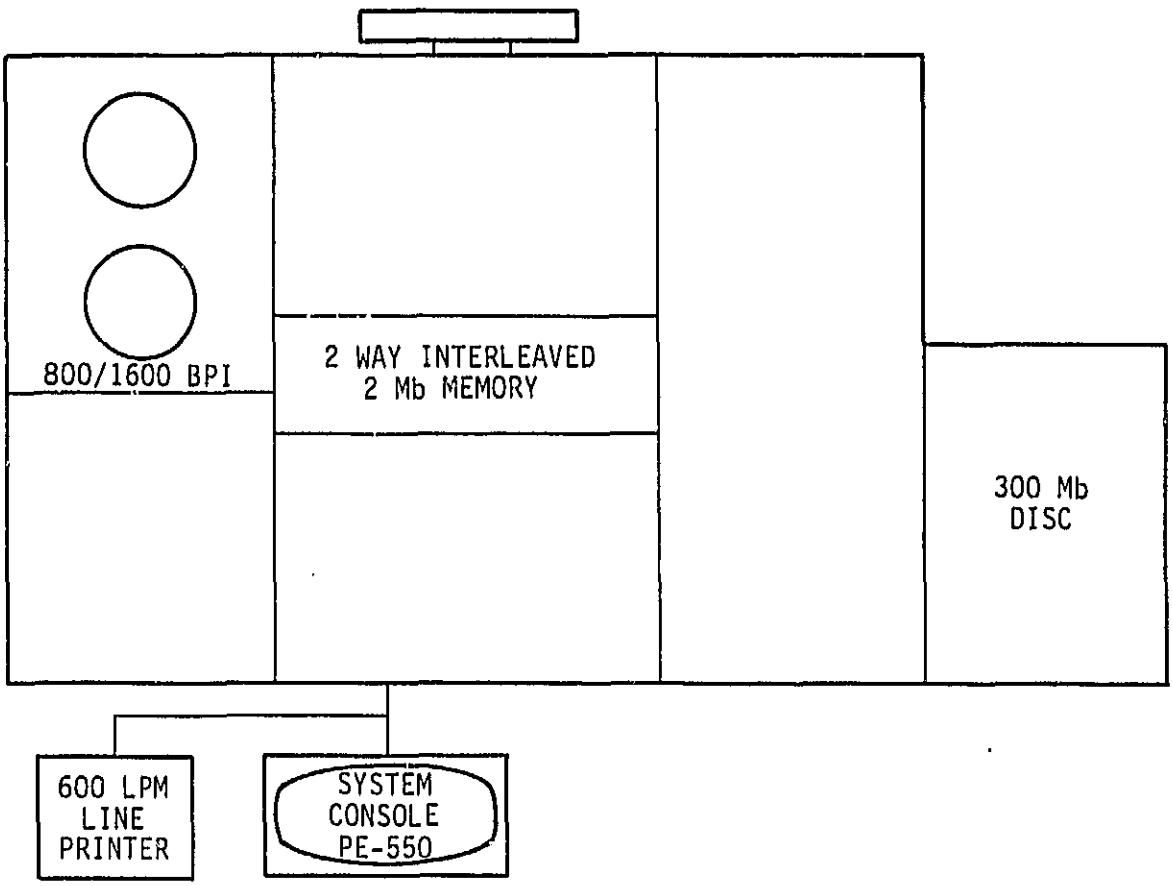

Fig. 2 AMASS Configuration

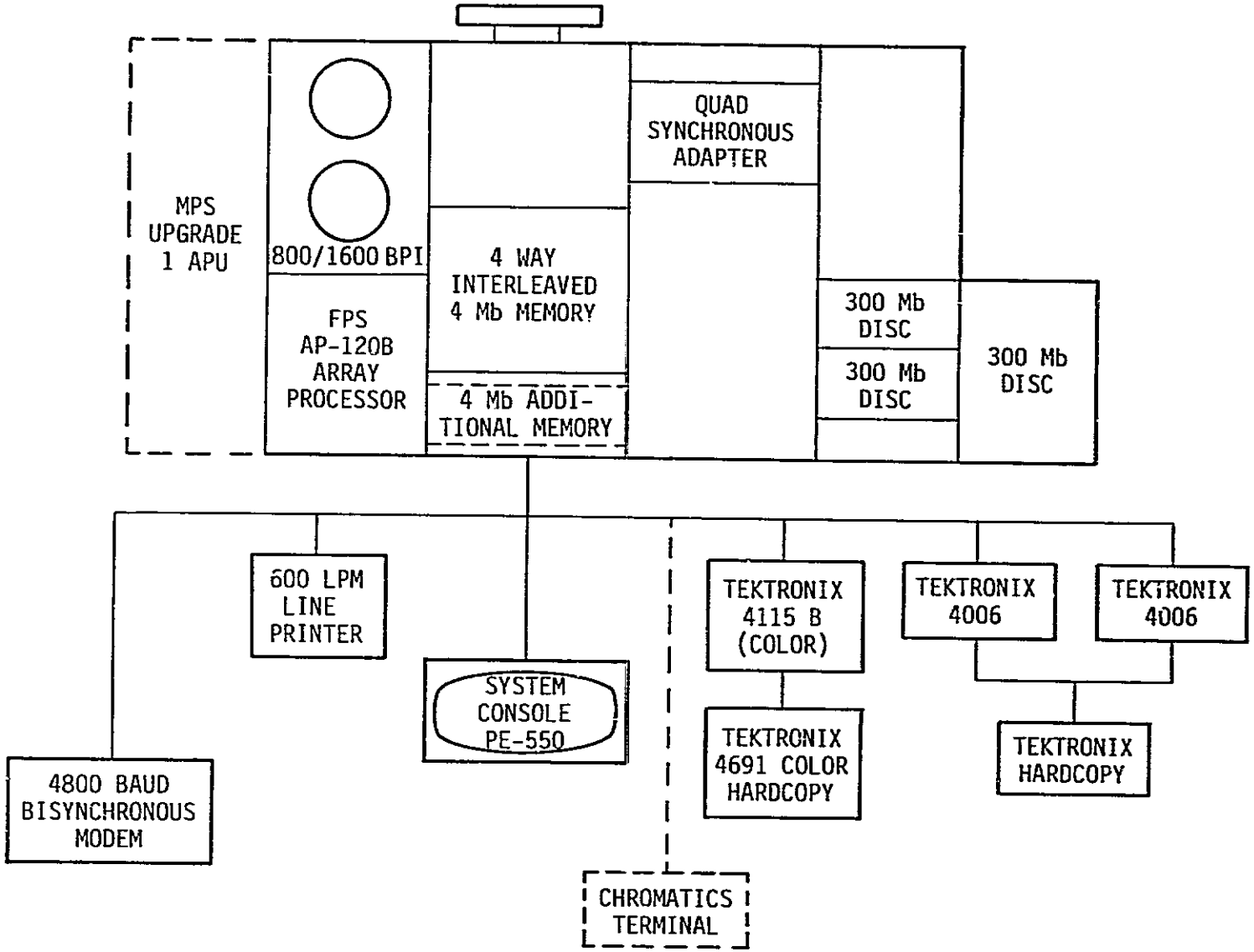

Fig. 3 Hardware Configuration

- Four-way interleaved memory
- Additional 2 Mbytes Memory (total of 4 Mb)
- Array Processor (on loan)
- Tektronix 4115 color terminal
- Tektronix 4691 color hardcopy
- Tektronix 4006 terminal (2)
- Tektronix monochrome hardcopy
- 600-Mb disc storage
- Selector channels
- 4800-baud bisynchronous model/quad synchronous adaptor (QSA).

The selector channels were installed to create a separate data path for (1) the original 300-Mb disc, (2) tape drive, (3) array processor, and (4) 600-Mbyte (actually two 300 Mbyte discs on a shared controller) disc drive.

The 4800-baud modem and QSA work with HASP software to permit RJE connection with other sites such as GSFC. Four-way interleaving was installed in an attempt to increase overall speed as were the selector channels described above. Equipment recommended but not acquired includes:

- MPS upgrade for P-E (with one APU)
- Chromatics color graphics terminal (to support doppler radar/lidar)
- Additional 4 Mbytes memory (total 8 Mb).

2.8 OTHER

Classes in modems and image processing offered by UAH were attended by Ms. Parker as well as training at the University of Wisconsin for the Man Computer Interactive Data Acquisition System (McIDAS). General consulting was also occasionally offered to various users to assist them in optimizing use of the AMASS.

Currently a study is underway to determine the best means of implementing an overall data management plan for the facility. This will also facilitate transportability of data between the HP, Harris and Perkin-Elmer.

2.9 TRAVEL

The following trips were required for the purposes listed:

Destination	Duration	Purpose
University of Wisconsin Madison, Wisconsin	1 week	Study and evaluation of methodology of ingesting image data into the AMASS for use in image processing which may be quasi-real time
Goddard Space Flight Center Greenbelt, Maryland	1 day	Meeting regarding RJE between AMASS and Class 6 equipment

Attachment

ARRAY PROCESSOR EVALUATION

FOR

ATMOSPHERIC MODELING AND SENSOR SIMULATION SYSTEM

NAS8-35189

May 1984

Prepared by:

Karen G. Parker
New Technology, Inc.

Prepared for:

ATMOSPHERIC SCIENCES DIVISION
George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama

Section I. INTRODUCTION

The Atmospheric Science Division (ASD) is faced with the problem of maintaining adequate throughput for its users at reasonable cost. In general, it has been estimated that the throughput requirements of a typical university or industrial computer center are escalating at a rate of 15 to 20 percent per year, based on either Central Processing Unit (CPU) seconds or connect time. This growth can perhaps be attributed to two basic causes. The first is an increase in the number of applications amenable to analysis or modeling by computers. The second is the use of increasingly sophisticated analytical techniques and models. Certainly the field of atmospheric science reflects such growth in both areas as demonstrated by the increasing dependence on computers to perform the intense "number crunching" required by many of the algorithms for such items as cloud modeling, weather forecasting, satellite and radar data reduction, and image processing.

Although these throughput needs have in some cases been satisfied by using centrally-located large-scale vector processing machines such as the Cyber and Cray, there frequently exists a need to locate the computer resources within the confines of the laboratory or research facility. Minicomputers are often used in performing these research tasks; however, the numerical calculation requirements can overwhelm such a machine, and processing time to accomplish the manipulation of the data may exceed all requirements. Often computationally intensive problems have been relegated to be run after hours in order to avoid interference with interactive user activity during normal working hours.

Currently, one project in which the ASD is involved is the development of a cloud modeling program, which is predominatly CPU-bound on the Perkin-Elmer (P-E) 3250 system. Not only does the software require on the order of days to execute; other users are excluded for all practical purposes from being allowed to run while the model is executing. One obvious approach to solve this problem is to offload the code to a Cray or Cyber class machine, particularly since the code seems well-suited to vectorization. This approach is being followed; hardware and software required to utilize the Houston Automatic Spooling Program (HASP) on the P-E to allow connection to the Cyber 205 at Goddard Space Flight Center (GSFC) have been installed. However, this will not necessarily solve the overall problem because requirements for other somewhat similar models will certainly appear later. This approach also involves coordination with GSFC for resources such as CPU time, disk space, etc.

Another approach to increase and maintain throughput, which appears to be a cost-effective alternative, is to attach an array processor as a peripheral to the P-E computer.

The array processor may be considered to be a computer, optimized in architecture and instruction set, which accepts blocks of data and instructions from a host mini- or large-scale computer and performs computations at speeds many times those possible by the host alone.

In a typical application, a program is separated into two parts. One, basically input/output (I/O), is executed by the host computer while the other,

compute-bound, is executed by the array processor. For discussion purposes, compute-bound will imply that the routine or program actually expends 80 percent or more of the total execution time of the program in performing numerical calculations.

Usually, an array processor is interfaced to a front-end host computer that handles communications with users, file manipulation operations, and most peripheral I/O operations (Figure 1). The array processor concentrates on the computationally intensive operations and effectively acts as a processing unit for the system. This arrangement imposes a twofold impact on computer technology:

- A compact, economical minicomputer-based array processor system could possibly offer computer power comparable in some ways to that of a multimillion dollar mainframe system.

- Time consuming (and thus costly) calculations may be offloaded from a host to an array processor as an economical move to increase the capabilities of the host system to handle other users.

The same specialized architecture that causes the array processor to be super efficient for "number crunching" applications make it ill-suited for other applications that do not involve intensive arithmetic. Such areas as signal processing, image processing, speech recognition, and structural analysis are very well-suited to using array processors. Due to the compute-intensive nature of the atmospheric models and extensive use of arrays, it appears that addition of an array processor could be most beneficial in offloading the P-E and consequently freeing the CPU resource to enhance throughput for other users.

In the past, due to data and program source limitations, most array processors were used for special, dedicated applications such as signal or image processing. In many cases, this meant using modestly sized programs for which speed was critical. There has also been a general reluctance on the part of data processing managers to attach "foreign" equipment to their systems. However, improvements in array processor architecture, including the incorporation of hardware to handle much larger programs, larger quantities of data, and more accurate data representation as well as the development of enhanced software to facilitate programming have combined to make the array processor a feasible alternative solution to the throughput problem.

Within this document, the following array processors will be evaluated for anticipated effectiveness and/or improvements in throughput by attachment of the device to the P-E:

Floating Point Systems (FPS) AP-120B
Floating Point Systems 5000
CSP, Inc. MAP-400
Analogic AP500
Numerix MARS-432
Star Technologies, Inc. ST-100

These vendors either currently offer a standard, off-the-shelf interface to the P-E host, or they plan to have a prototype available. Other vendors which were omitted, such as Sky have indicated no intention to offer a P-E interface at this time, primarily due to a limited market demand. Although NUMERIX was included in this study, they are not committed to providing a P-E interface. However, they have expressed interest in the possibility and have contacted P-E corporation regarding the matter.

Section II. BACKGROUND

Array processors have achieved their speed with state-of-the-art logic elements combined into functional units that display some combination of the two keys toward increased throughput: parallelism and pipeline processing. Just as Direct Memory Access (DMA) transfer may be overlapped with ongoing processing, architectures that incorporate parallelism allow such operations as floating point addition, floating point multiplication, integer arithmetic, and data fetches to occur simultaneously. This parallel processing becomes more efficient as the number of things that can be done simultaneously increases. For example, an algorithm performed in 8 seconds using in-line code might require 2 seconds if four distinct processor elements could be used efficiently in parallel.

The array processor has multiple data paths to eliminate communication bottlenecks that might prevent the various hardware elements from operating in parallel. The use of multiple data paths is particularly helpful for a number of reasons. By permitting each element of the machine to have access to the others, bus constraint problems are removed. The device thus does not need to wait for another particular device to become available in order to transfer data. An important feature of this architecture is that it permits ease of programming and debugging since no element has to rely on the state of other elements to perform its function. The net result of using this approach with the Floating Point Systems architecture, for example, is that up to ten floating point addition, floating point multiplication, data fetch, deposits of results, address calculation, etc. can be processed simultaneously during each cycle.

Normally, floating point calculations are not performed at one time; instead, several intermediate steps may be required. This "pipeline" process may be used on an array processor to increase hardware utilization by segmenting time-consuming operations like floating point addition and multiplication. Consequently, each functional unit can output every machine cycle, once set up, even if it takes more than one cycle to perform the entire operation for a segmented unit.

Floating point arithmetic takes longer than accumulator or integer arithmetic operations. Therefore, pipelining a floating point multiplier and adder frees array processor designers from basing cycle time on the relatively long interval required for floating point arithmetic. Since a sum or product can be initiated every cycle, pipelining increases the number of floating point operations that can be performed in a given period of time. For instance, Figure 2 depicts the use of pipelined calculation in the floating point multiplier of the Floating Point System processor. Note that three separate steps are required to complete the multiplication of two 38-bit floating point numbers. If the value of X1 times Y1 were desired, the product would be available after three machine cycles, since each stage requires one machine cycle. Using the FPS machine cycle time of 167 nanoseconds, approximately 0.5 microseconds will be required to obtain the product of X1 and Y1.

With this pipelined multiplier, it is possible to begin the next product, say X2 times Y2, when the first product reaches stage 2. Similarly, a third

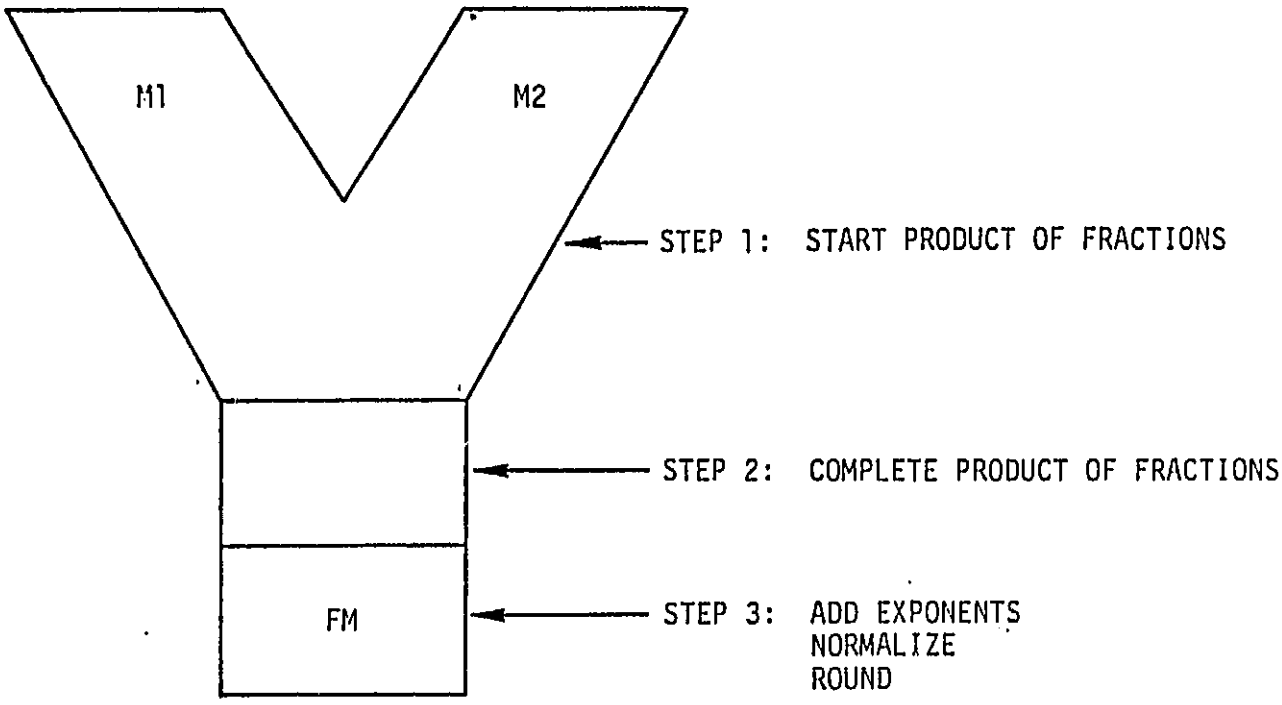

Figure 2. Floating Point Multiplier

product X3 times Y3 may begin when the second product is transferred to stage 2 and the first product is transferred to stage 3. This means that a result is made available on every machine cycle, in this case at a rate of one every 167 nanoseconds.

Array processors differ primarily in how closely they are coupled to their host computer and in their use of synchronous or asynchronous functional units. Tightly coupled processors use the host memory as their own, and consequently can be connected only to the host for which they are designed. Loosely coupled systems, in contrast, have their own memory and can be connected to different host computers. The disadvantage of this design is that it requires data transfers from the host to the processor which may be very time-consuming. Memory size for loosely coupled systems may vary from a few thousand to several hundred thousand locations for data ranging from 16 to 64 bits in length. In most array processors, 32 or 38 bits are used for floating point numbers, while 16 bits are commonly used for integers.

Processors incorporating synchronous timing contain functional units that operate at the same speed, providing a predictable flow of data. On the other hand, in asynchronous designs the functional units operate at different speeds, so control modules are required to ensure that the data flow between units in a correct, timely fashion.

Elements of a synchronous processor operate according to fixed time interrelationships, thus largely avoiding the need for a programmer to coordinate the various subsystems. In addition, the synchronous array processor has a finite number of machine states and can thus be modeled in software by a simulator, which enables a user to debug a program interactively, line by line.

Although the asynchronous array processor tends to be characterized by a capacity for relatively high throughput, it may be arduous to program since its architecture requires the programmer to assume the burden of coordinating various asynchronous elements. In addition, program debugging may be quite difficult due to the unlimited number of machine states.

Both synchronous and asynchronous machines are evaluated in this document. Individual processor architectures are described for each machine.

Section III. EVALUATION CRITERIA

Requirements of the enhanced P-E or P-E/array processor configuration include the following:

- Capability to run Fast Fourier Transforms for up to 128 x 128 x 128 arrays.
- Some programmability to permit setup of a sequence of commands for subsequent execution by the array processor independent of the host.
- FORTRAN compiler for the array processor.
- Minimum of 1 Mb memory, directly addressable.
- Library of image processing routines.
- Reasonable complement of diagnostic routines to permit isolation of faults in hardware.

Both internal array processor architecture and host minicomputer system architecture affect the performance and efficiency of the combined system.

In order to accomplish an effective interface with a host minicomputer such as the P-E, an array processor should have the following general capabilities:

- Perform rapid calculations in conjunction with readily available, standard, relatively inexpensive host minicomputers.
- Interface directly to the host with minimal downtime.
- Have the capability to work independently of the host, which is required only to transfer computationally intensive problems to the attached processor and retrieve results when available.
- Offer several specialized processors operating in parallel mode since multiple parallel processors, under effective control, yield high speed operation.
- Feature software adaptability which allows program deletions, changes, or modifications to existing algorithms and enables new routines to satisfy altered application requirements.
- Provide floating point data format since rounding hardware in floating point arithmetic units, which is compatible with fixed-point formats, promotes improved accuracy.
- Provide a means for directly connecting external devices such as disk drive, analog-to-digital (A/D) devices, etc. to the

array processor to prevent burdening the host with future additional I/O requirements. In this manner, interaction between the host memory and the array processor is limited to transferring final results to the host.

Several items may be considered to evaluate array processor performance:

- Million Floating-Point Operations Per Second (MFLOPS)
- Function execution times
- Peak memory access rate
- Standardized benchmark programs (e.g., application code or Whetstone).

Of these, the most appropriate measure is the application program since it provides immediate and direct estimation of anticipated performance improvement.

Use of MFLOPS as a measurement tool tends to be misleading. For example, suppose it is desired to estimate the vector addition function execution time on a synchronous array processor with an advertised speed of 12 MFLOPS. The equation

$$y(i) = u(i) + v(i) \quad , i = 1,2,\ldots N$$

requires N floating point additions. Merely dividing N by 12 to yield a time of 83 nanoseconds times N is incorrect because three memory references per addition are required at a time of 167 nanoseconds per reference. Specifically, the references are

fetch u(i), fetch v(i), store y(i)

and the time to accomplish these memory references for each addition operation is 167 x 3 = 501 nanoseconds. Therefore, the estimated time per operation is actually 584 nanoseconds, which differs from the original estimate by 501/83, which is roughly a factor of six.

Suppose instead that function execution times are used. Assume that the time required for the vector addition operation is advertized at 0.5 microseconds per point vector add time. If vectors u and v are read from host to array processing memory, added to form a result vector y, and written back into host memory; I/O time dominates arithmetic processing time. Assuming that the host/array processor operates at a 1-Mb/second memory access rate, 4 microseconds are required to read the 32-bit sample from host memory and 4 microseconds to write it back, totaling 8 microseconds I/O time per sample. The 0.5 microseconds per point vector add time is insignificant compared to the 8-microseconds transfer time.

Since all of these measurements tend to be rather misleading when utilized on an individual basis coupled with the fact that each user's application typically requires a different mix of operations, timing a specific application program or some subset of it appears to provide the best direct measure of array processor performance. Also, by having competing array processor vendors

implement either all or part of an application program, the user can measure both vendor responsiveness as well as product performance at the same time. It is important to note, however, that although a limited number of vendors offer a P-E interface for their array processor, it may not be possible to run a user benchmark on a P-E array processor configuration. Perhaps it will be necessary to benchmark using a different CPU host such as the VAX computer in order to obtain baseline results for comparison.

Incidentally, due to the effort necessary to convert code to run on an array processor (without using an array processor oriented FORTRAN compiler), some of the vendors have been rather hesitant to run any user-selected benchmark code.

In view of these points, the following items will be used as basic guidelines in evaluation:

- Level of satisfaction of user requirements
- Interface compatibility with host
- Array processor architecture (especially speed and memory configuration)
- Benchmark results (where available)
- Software flexibility and programmability
- I/O requirements (possibility of direct connection to external devices)
- Cost

Section IV. TECHNICAL CHARACTERISTICS

The purpose of this section is to provide a brief overview of each of the array processors undergoing evaluation, namely:

Floating Point Systems (FPS) AP-120B
Floating Point Systems 5000
CSP, Inc. (CSPI) MAP-400
Analogic AP500
Numerix MARS-432
Star Technologies, Inc. ST-100

Introduced in 1976, the Floating Point Systems AP-120B array processor allows computational speeds of up to 12 MFLOPS. The data word length and table memory word length are each 38 bits; the instruction word length is 64 bits. Figure 3 depicts the architecture of the AP-120B.

A fairly recent addition to the FPS product line is the FPS-5000 series. The general architecture design of this series is shown in Figure 4. Note that although the overall system is synchronous, the compute processors operate in an asychronous nature in order to achieve maximum throughput. Another feature of this series is that these multiple compute processors, termed "co-processors," may be used in a configuration to enhance performance. The 5200 and 5300 series are designed to interface with a P-E host and operate in a manner similar to the AP-120B. Figure 5 shows the FPS-5000 system and architecture.

Although the currently offered FPS FORTRAN compiler does not support the co-processor architecture, a new version offering this support is supposedly soon to be released. The new version should also provide ANSI-77 level FORTRAN. Currently another problem exists in programming the FPS--specifically the 64KW page limit. The new compiler is also supposed to correct this problem.

The architecture of the CSPI MAP-400, which was introduced in 1981 is depicted in Figure 6. The MAP-400 is basically an extension of the MAP-300. The system actually consists of two MAP-300 arithmetic processing units and a single Control and Supervisory Processor Unit (CSPU). It is advertized to be capable of executing 24 MFLOPS.

The Host Interface Module (HIM) performs all Host-MAP transfers of commands and data. Program Memory (Bus 1) contains the SNAP-II Executive and library. Each bank of data memory may contain up to 256 K bytes of memory in any combination of 170-, 300-, or 500-nanosecond access speed. The CSPU is a 16-bit single-board minicomputer which executes the Snap II Executive Operating System and provides control over all MAP processors.

Since the CSPI is based on asychronous architecture, it tends to be somewhat tedious to program. Another consideration is that it was extremely difficult to get information from the vendor. If this lack of vendor responsiveness provides any indication of corporate product support after purchase, the performance features of the MAP-400 seem eclipsed by fear of subsequent maintenance and programming problems.

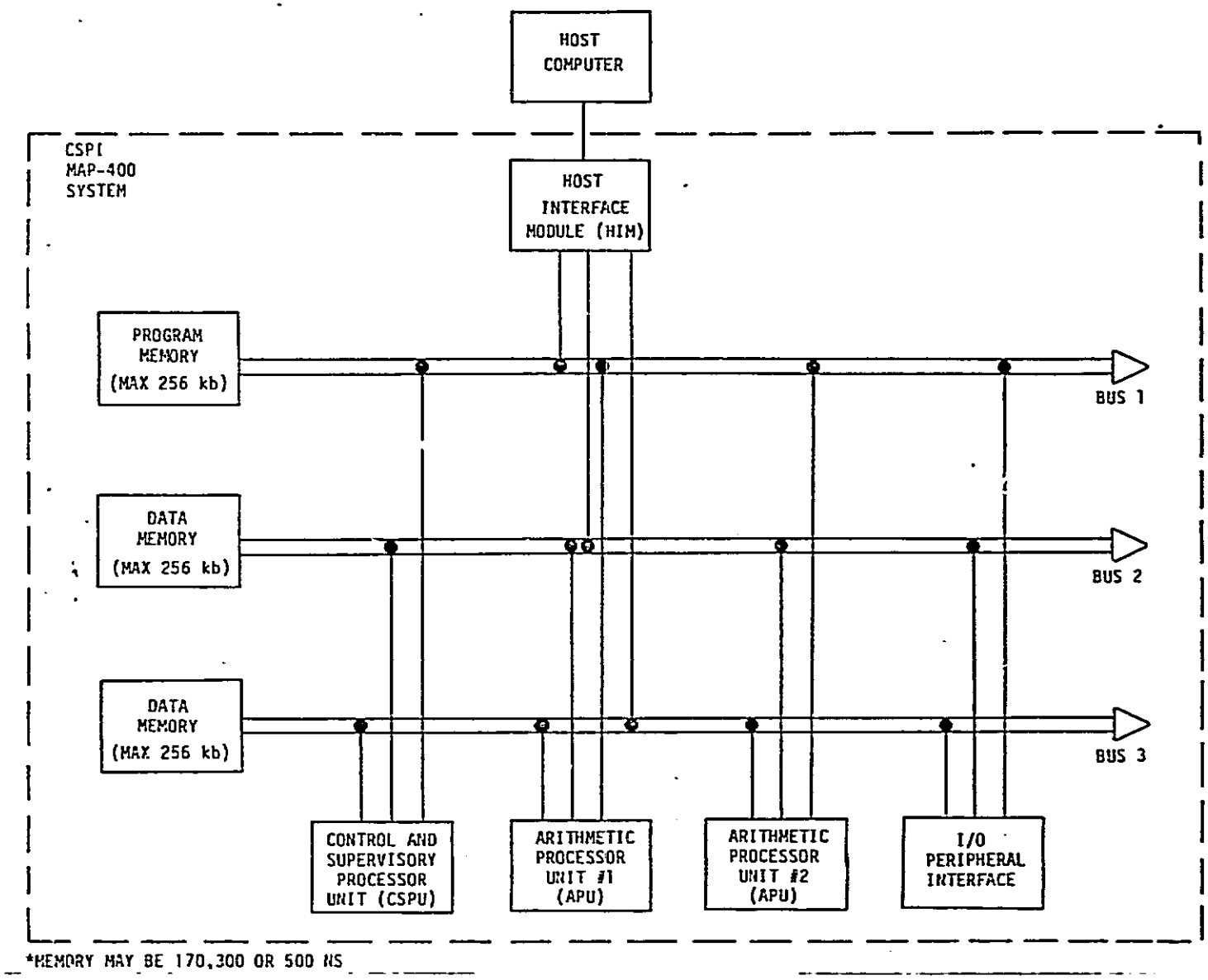

Figure 6. CSPI MAP-400 Architecture

In contrast, contacts with Analogic proved most favorable. All requests for information and/or technical questions were answered without delay, and frequent "check-up" calls were received. A week of tuition-free training was also offered.

The Analogic AP500, which was released in 1982, offers a relatively low-cost array processor with an advertised maximum of 9.4 MFLOPS. The accuracy internal to the pipeline is 40 bits, with a 32-bit mantissa and 8-bit characteristic. Thus, the precision of the 32-bit mantissa exceeds that of the 28-bit mantissa of the 38-bit array processor and similarly, the 24-bit mantissa of the 32-bit array processor.

The system architecture of the AP500 is shown in Figure 7. It is based on a Motorola MC 68000 supermicro and supports up to 256 K bytes of program memory and up to 1 M (32 bit) words of data memory. Although Analogic does not currently market an interface to the P-E, one is currently under development and is due to be released within months.

The Numerix MARS-432 is advertized as having computational power of 30 MFLOPS. As in the case of Analogic, Numerix currently does not offer a P-E interface; however, they have been in negotiation with P-E to develop one. According to a P-E representative, this Numerix project is still in its "infant" stage. Analogic and Numerix are developing their respective interfaces in response to customer pressure, and estimates for having these items commercially available vary from a few months to possibly a year.

A FORTRAN Development System is available for Numerix which consists of a FORTRAN compiler, linker, and trace/monitor. In addition, a Microcode Development System (MDS) provides an offline development package, which includes the macroassembler, microcode debugger, and a utility to provide automatic microcode optimization.

Figure 8 shows the architecture of the MARS-432. Although an explicit separation exists between Data Memory (DM) and Program Memory (PM), a data path between the two does exist. This enables the DM to be used as a bulk storage area, if necessary, for large PM segments. A second bus (CBUS) exists to connect all system elements within the machine and the host computer. It supports reading and writing various control and data registers in the processing system and may be used to examine the state of the complete machine. It is also utilized in dynamic control functions such as starting and stopping the array processor and signaling significant error conditions.

Perhaps the Cadillac of all array processors, the Star Technologies ST-100 array processor offers an advertised capability of 100 MFLOPS. STAR was formed in 1981 primarily by ex-FPS employees who recognized array processor limitations such as memory size, addressing constraints, and host overhead time. Another point of concern was that as host minicomuters improved performance, the array processors did not substantially enhance their capabilities over those offered during the mid-70's. The original goal of STAR was to produce the first second-generation machine with adequate speed to increase the system improvement ratio at 10:1 when used in conjunction with a host and to introduce features to eliminate or reduce to the greatest extent possible the known disadvantages.

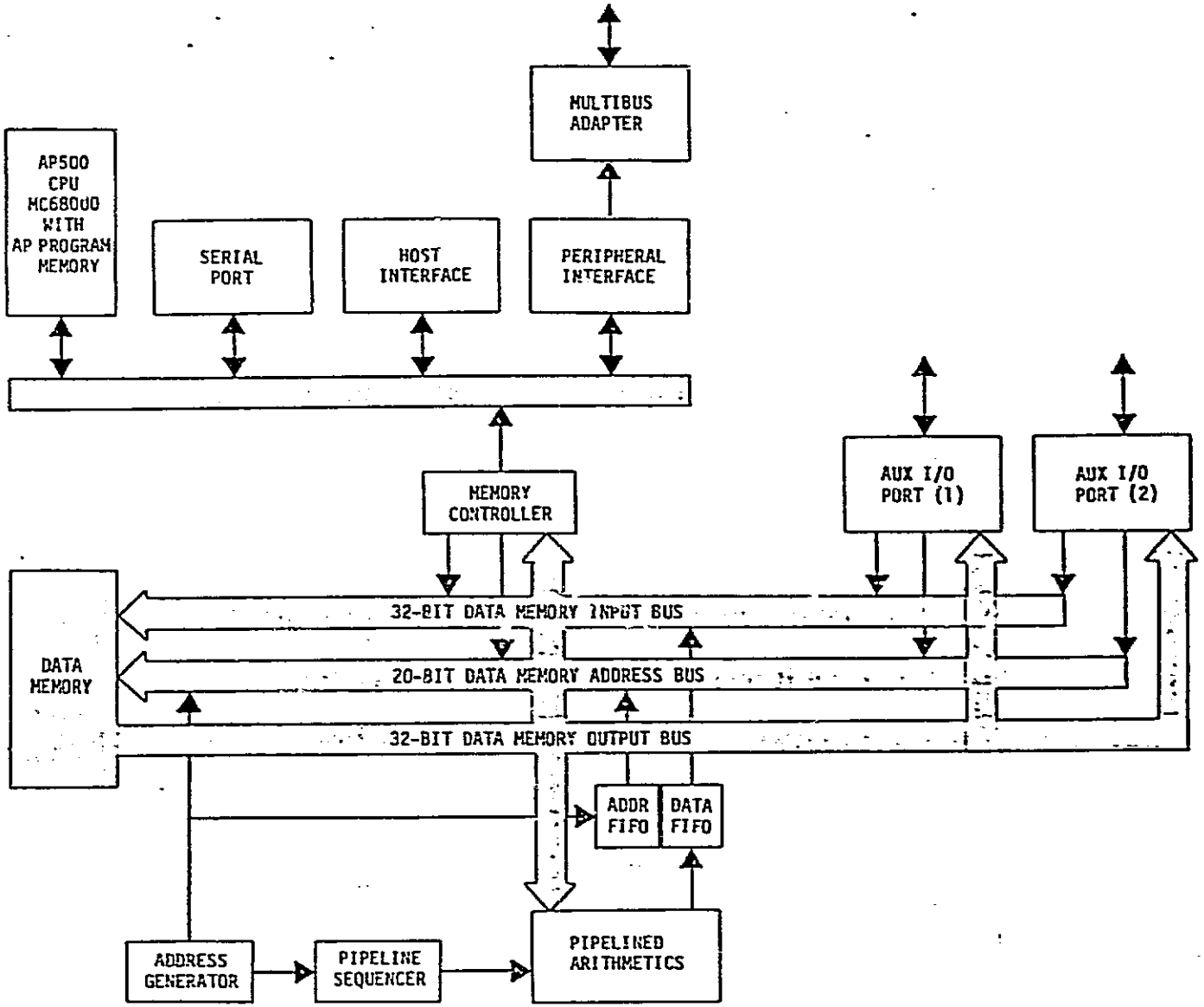

Figure 7. Analog AP500 System Diagram

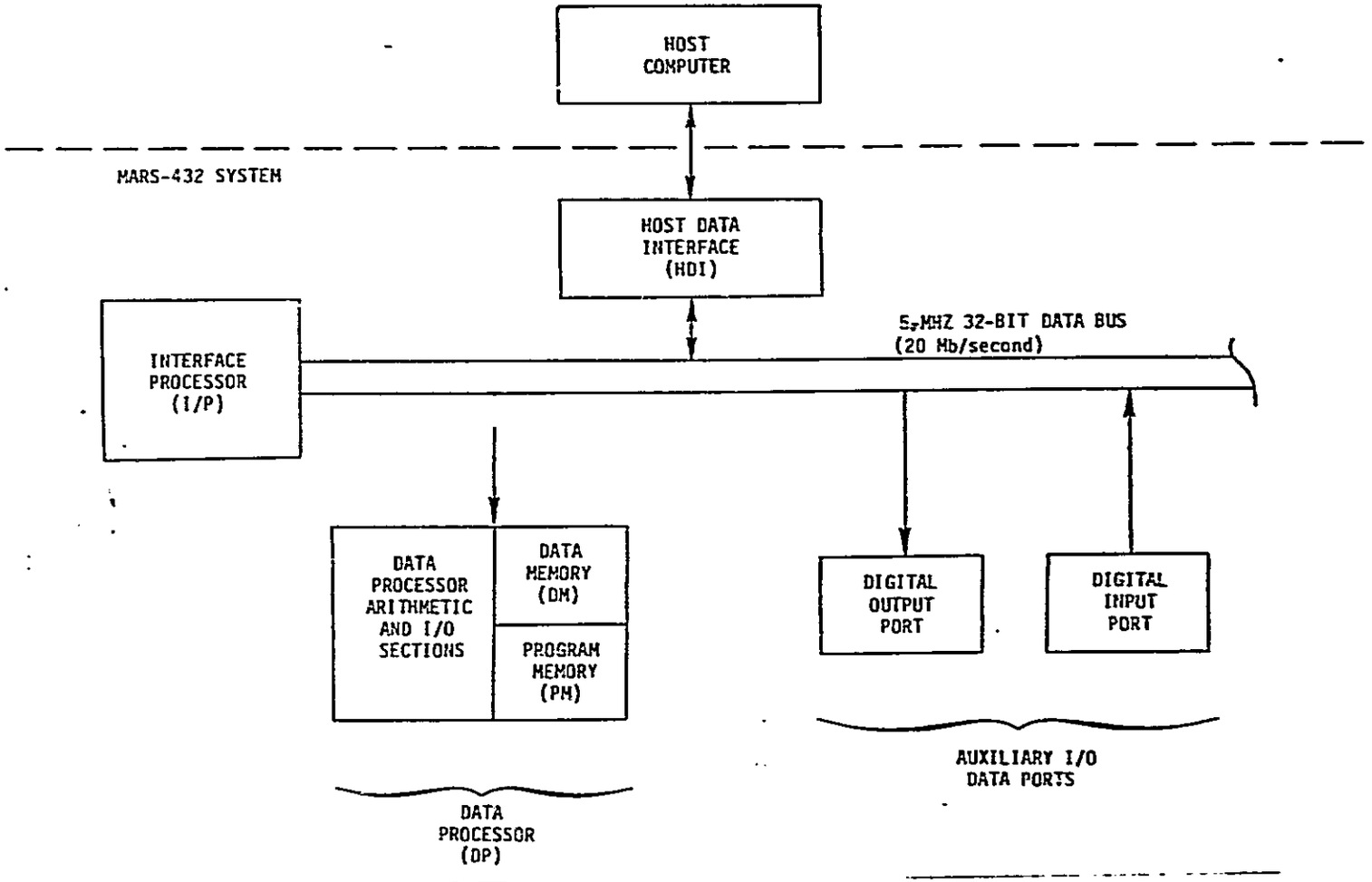

Figure 8. MARS-432 System Level Architecture

Figure 9 shows the system architecture of the ST-100. Note that a hardware division/square root module is included. The standard configuration has 512 K (32-bit) words of contiguous main memory, expandable to 8 M words. This essentially eliminates the program size and addressing limitations of the first generation of array processors. An additional 48 K words of data cache allows buffering of the arithmetic units from main memory, thus allowing simultaneous I/O with computation and obtaining a higher percentage of the maximum speed of the array processor. Therefore, simultaneous communication of the host computing system with arithmetic computation is permitted. This feature serves to reduce host overhead time substantially; this is one of the primary disadvantages of first generation array processors.

Figure 10 depicts the major flow paths of control and data for the ST-100. The ST-100 can interface to a maximum of seven host computers and may run in dedicated mode to one or in multi-user mode with all. Thus not only current systems, but future acquisitions as well, are accommodated.

Table 1 shows a comparative summary of selected characteristics for each of the array processors under evaluation.

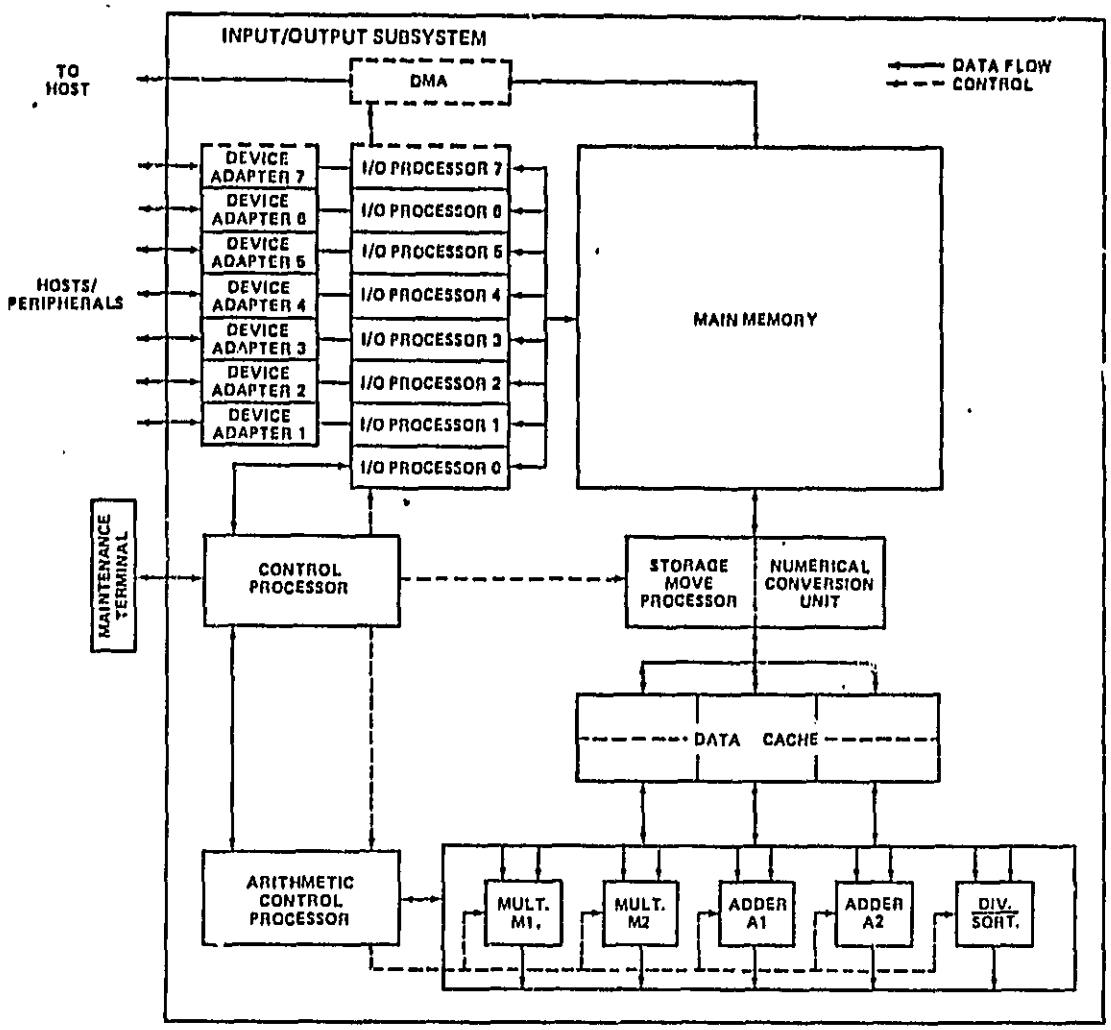

Figure 9. ST-100 System Architecture

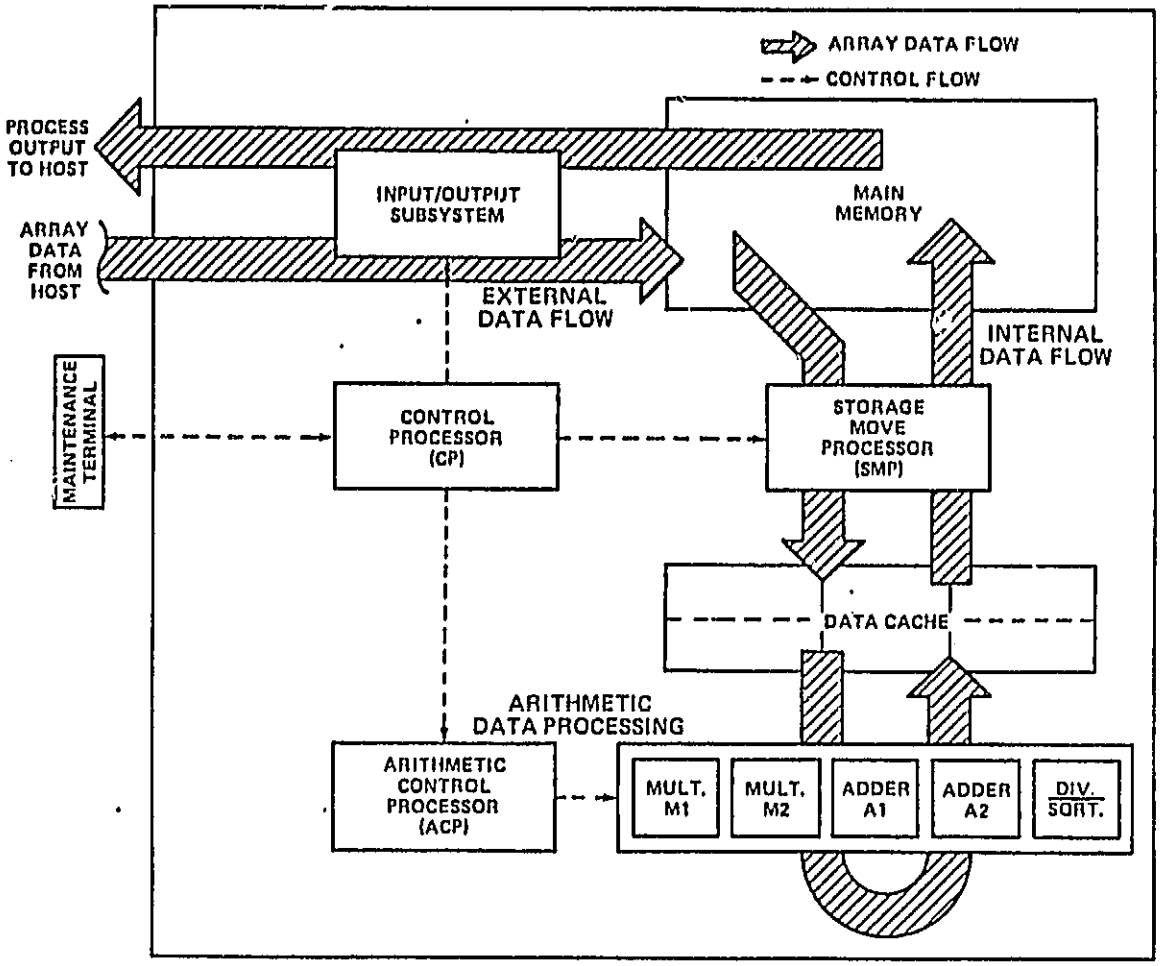

Figure 10. ST-100 Major Flows

Section V. SOFTWARE

The increased complexity of array processor hardware (as compared to serial computer hardware) and the emergence of general-purpose designs combined with a growing acceptance of array processors among users have had a major impact on the software.

At first, users had to code their array processors in assembly language. Because many users resisted this, most manufacturers now provide large libraries of mathematical functions. These routines are very tightly coded in assembly language and may be executed via FORTRAN-like calls. Most of these routines involve repetitive, nonrecursive, floating point calculations. They are repetitive in the sense that the same mathematical operations are performed on large arrays of data, but nonrecursive in that operations on the next set of data can begin before the final results of a previous data set are ready. Examples of repetitive, nonrecursive calculations are matrix operations, smoothing and filtering of signal processing data, and image enhancement techniques.

Occasionally as the user attempts to replace arithmetically intensive portions of a program with calls to array processor routines, a needed function may not appear with those provided in the library. Development software for an array processor is most helpful in this event. This software typically includes the following:

- Cross assembler - Assembles programs written in the array processor assembly language on the host machine.
- Simulator - Acts as an interactive program which runs on the host to simulate execution of the array processor.
- FORTRAN compiler - Converts FORTRAN subroutines into array processor assembly language code, which is subsequently assembled by a cross assembler. For large FORTRAN application programs, a compiler can expedite array processor program implementation; often, however, this is done at the expense of degraded execution speed.

FORTRAN compilers are commercially available for only a few array processors. Presently, these existing compilers can handle a subset of the ANSI 66 Standard for FORTRAN. These compilers have been constrained to modest-sized problems (less than 500 to 1,000 lines of FORTRAN), primarily due to limitations on the size of program source memory.

In general, the advantage of using a FORTRAN compiler over hand coding includes the following:

- Source code remains transportable
- Conversion effort is reduced

- Debugging is easier (standard FORTRAN on host)
- Source code remains readable

Compability with a FORTRAN environment will probably be far more important for the new generation of attached processors than it was for the previous array processors. Coding very large scientific or engineering programs in assembly language tends to be impractical, and library routines are not always available to duplicate all the mathematical operations of a typical user. Most users simply run their code through the attached processor FORTRAN compiler. Some users, of course, will take the additional step of replacing very small, numerically intensive sections of code with calls to library routines or with assembly language code of their own.

It is more complicated to obtain an executable program from FORTRAN source code for a peripheral array processor than for a stand-alone host computer. In addition to application routines, the attached processor is usually supported by a compiler, an assembler, and a linker. All of these program development tools reside in and run on the host computer.

Figure 11 shows that input for the FPS array processor linker consists of the outputs of the array processor FORTRAN compiler and assembler, together with any FPS library routines desired by the user. The linker will in turn generate two files, a Host-Attached Software Interface (HASI) and an array processor load module. The HASI performs the communication function between the host and the array processor. It must be run through the host FORTRAN compiler and linked with the host resident portion of the program and host library routines to create a file that the host can execute. The load module contains the code that is run on the attached processor and transferred from the host during program execution.

In all probability, the FORTRAN compiler is the software module that will be one of the most crucial areas in determining the future success of the general purpose attached array processor. Generating efficient code for a computer that is pipelined and capable of parallel operations is certainly a challenging problem for the array processor vendor.

It is anticipated that it will be some time before compiler code reaches an efficiency level comparable to that generated by an experienced assembly language programmer. However, some of the vendors indicated that their companies are currently concentrating on improving their compilers. Two also indicated that ANSI-77 FORTRAN compilers should be announced shortly as standard products of their respective companies.

Diagnostics are also frequently provided to enable the user to identify a hardware problem, isolate the defective board, and exchange it for a factory or depot-supplied replacement.

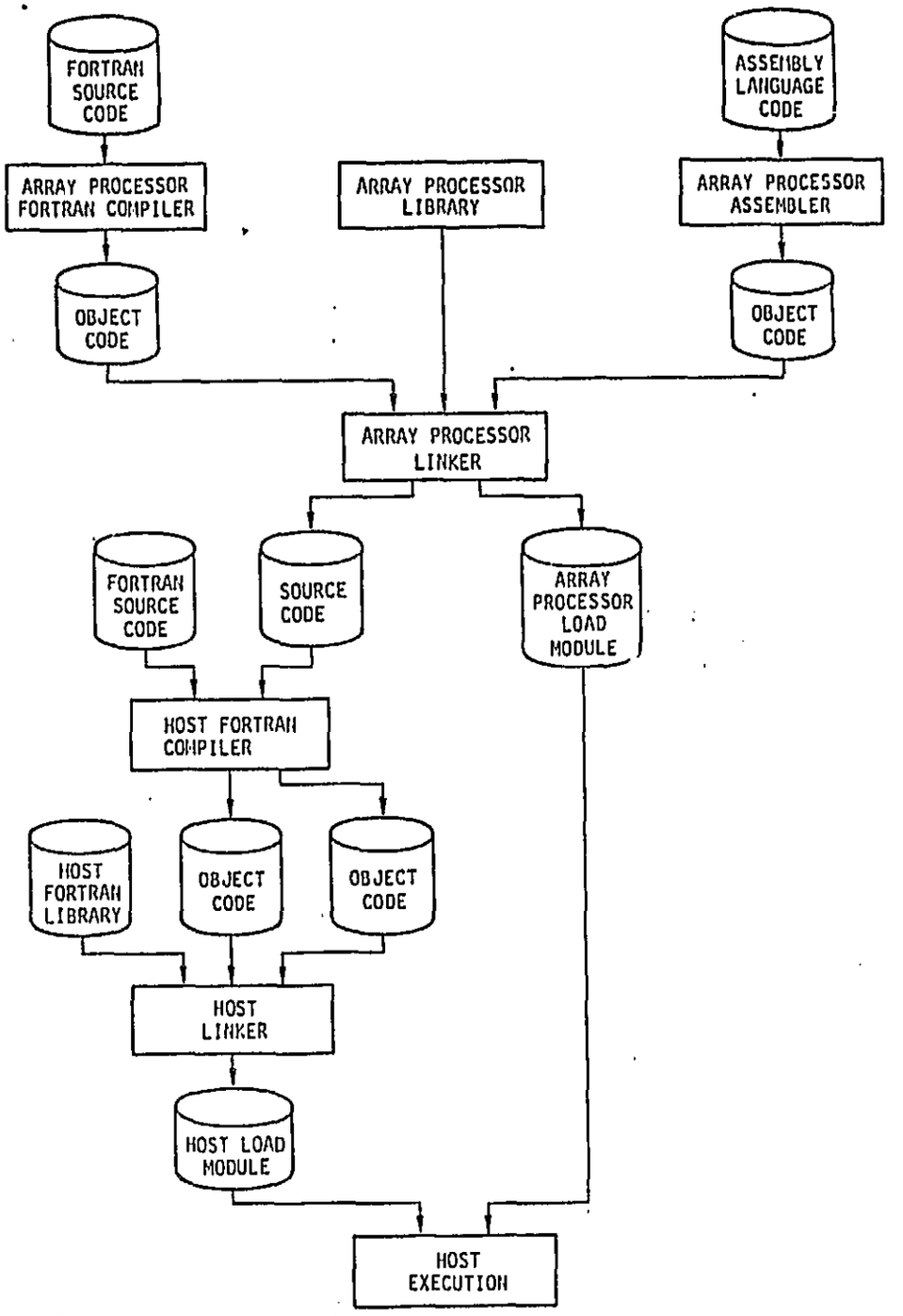

Figure 11. FPS Array Processor Linker Setup

Section VI. COST DATA

Example configurations (Tables 2 through 5) were priced for purposes of comparison. Guidelines included 1 Mb of data memory, host interface hardware and software, libraries, and (if available) the FORTRAN compiler. Configurations were done to match one another as closely as possible to facilitate realistic evaluation.

Section VII. SUMMARY AND CONCLUSION

None of the array processors under study seemed to perfectly meet requirements. Currently, a Model FPS AP/20-B is being used on a temporary basis to evaluate performance of an array processor interfaced to the P-E for selected program models. It is recommended that this study continue and that the market be monitored for updates in the status of all array processor vendors who offer or introduce P-E interfaces. Perhaps within a year or so, an array processor may become available which satisfies all requirements, offers a standard off-the-shelf P-E interface, and is of a more reasonable cost. Thus, by postponing this purchase for a while, a better solution may be available that is also more cost-effective.

During the course of this study, however, a very interesting approach appeared. P-E offers an upgrade of the P-E 3250 to a Multi Processor System (MPS) for a cost comparable to that of the average array processor. This approach involves modification of the hardware to support one or more Auxiliary Processor Units (APUs) under direction of the CPU. This method will permit multiple CPU-bound jobs to execute concurrently while sharing central memory. Currently this approach seems most appropriate to enhance overall system performance and throughput. Figure 12 is a quote for this upgrade. It is also recommended that an additional 4 Mb of memory be purchased at a cost of approximately $32,000 to support running multiple jobs in an efficient manner.