CR-152177

The ILLIAC IV Memory System: Current Status and Future Possibilities

David Stevenson

May 8, 1978

Revised August 7, 1978

Institute for Advanced Computation

NASA-Ames Research Center

Moffett Field, California

CR-152177

THE ILLIAC IV MEMORY SYSTEM:
CURRENT STATUS AND FUTURE POSSIBILITIES

NAS2-8870	Technology Development Corporation
NAS2-8728	Lear Siegler Incorporated, Management Services Division
NAS2-9359	Computer Sciences Corporation

1

C O N T E N T S

FIGURES

T A B L E S

✓

Acknowledgements

This report represents the work of many members of the Institute. The following individuals gathered information, designed the proposed systems, or otherwise contributed to this report: Jim Bennett, Hal Brown, Hsiau Chang, Gary Feierbach, Al Goodrich, Rich Guyon, Mo Gunn, Stu Hopkins, Chuck Johnson, Glenn Lewis, Ray Lim, Bill Malloy and Steve Tulloh. In addition, the following researchers at NASA-Ames were helpful in supplying information about their Illiac codes and their use of the system: Harry Bailey, Margaret Covert, Harvard Lomax, Walter Reinhardt, Bob Rogallo, Ken Stevens and Alan Wray.

David Stevenson
Project Leader

I. Introduction

The Illiac IV was conceived in the mid-1960s; design began in 1967 and fabrication two years later. Since that time, memory technology has changed dramatically. For example, the fast local processor memory of the Illiac uses the first commercially available high speed bipolar memory devices (developed especially for the project), each chip having a storage capacity of 256 bits. Today, chips of comparable access speed have up to sixteen times as much storage capacity. A magnetic disk technology was selected for the extended memory of the Illiac. The years since then have seen the introduction of charged-coupled devices and magnetic bubble technology as alternative storage media, and the cost per bit of slow, random access electronic memory devices has decreased to the point that they are economically feasible for billion bit computer memories.

Not only has memory technology progressed, but the requirements being placed on the Illiac have also evolved in the intervening years. Now that considerable experience has been accumulated with two-dimensional models of physical phenomena, scientists in such areas as seismic and aerodynamic research have turned their attention to three-dimensional models, and this change increases memory requirements by at least a factor of ten; also the sheer amount of data being gathered by satellites has significantly increased the volume of data in signal and image processing. Finally, the inevitable deterioration of rotating mechanical devices due to age is beginning to impact the reliability and availability of the current memory system. Fortunately, the advances in memory technology can be brought to bear in relieving the two major pressures on the current memory system: replacements significantly different with respect to size and reliability are now feasible. This report examines several different enhancements.

I.A. Background of the Study

This study was charged with examining the future needs of researchers who will use the Illiac and estimating the requirements they will place on the memory system. The staff of the Institute who participated in the study were drawn from a broad base of disciplines: project designers, application programmers, systems specialists, and hardware engineers. Thus the study was capable of investigating all relevant issues concerning the Illiac memory system.

The underlying philosophy of the study was that memory enhancements to the system should be evolutionary rather than revolutionary in nature and that they should be aimed at meeting the forseeable needs of the Illiac user community during the next five years. The consequences of this philosophy were two-fold. First it was necessary to review the system architecture to determine its inherent limitations and current state. This determined the components which must be replaced and the limitations due to interfacing the replacements with the remaining system. The second consequence was the necessity to review the ways current users manage the system resources in solving their problems. This indicated the characteristics which would be desirable in the replaced components. Determining the characteristics of the current system and its users constituted the first phase of the study.

The second phase of the study considered various alternatives to replacing critical memory components. Three different approaches were explored to the point that their merits and limitations could be reasonably ascertained. The main issues which were considered for each alternative were cost, risk, system impact, software requirements and implementation schedules.

Two major areas concerning the memory system were not included in this study. The first was the user interface: the job control issues of the way users acquire, describe and manipulate the various memory resources in the system. The second area was the performance evaluation of the system: the hardware and software tools used to collect and analyze system performance characteristics for system management and accounting purposes.

I.B. Structure of the Report

The next chapter provides an executive survey of the results of this study. The remainder of the report is devoted to a detailed presentation of the material which was gathered during the course of the investigation.

The third chapter presents the findings of the first phase. It begins with a description of the current system, its performance and status, and the limitations it places on possible enhancements. The chapter includes a description of the planned enhancements to the Illiac processor and the probable resulting increase in its speed. There follows a description of representative codes on the Illiac with respect to their memory usage characteristics; these are useful in projecting future computational demands that might be placed on the memory system. A model of the Illiac memory system is developed, and using the characteristics of the example codes, several conclusions about desirable characteristics for the future memory system are drawn. Finally, several less easily quantifiable influences on the memory system are discussed: the relationship between local memory size and the cost of designing user application codes, the need for error correction ability on various memory components, the feasibility of a virtual memory system on the Illiac, and the requirements for an on-line high performance file storage facility.

The final chapter presents the material developed during the second phase of the study. After a brief technology survey, different implementations are presented for each system memory component. Next, three different memory systems are proposed to meet the identified needs of the Illiac user community. These three alternatives differ considerably with respect to storage capacity and accessing capabilities, but they all offer significant improvements over the current system. Each proposed system is analyzed in detail, and their relative merits are discussed. The chapter concludes with a review of the current system weaknesses, and how these weaknesses will be addressed by the proposed memory systems.

II. Executive Summary

The memory system of a modern, large scale computer serves three different purposes: from the user's point of view, 1) it holds instructions and data for rapid accessing by the processor, 2) it contains the large data base for the computer to process, and 3) it provides storage for files when they are not being actively processed by the computer. In the Illiac IV computer's memory system, these functions are performed respectively by different memory devices: 1) a fast semiconductor local memory, 2) a large, high performance disk extended memory, and 3) a commercial disk and tape file system. In addition, the memory system has other components for buffering the data transfers between these devices.

The current Illiac memory system has enabled scientists to demonstrate the applicability of parallel processing in such diverse and important areas as computational aerodynamics, climate modelling, satellite image processing and seismic research. These past successes have generated demands for even greater capabilities in the memory system. Unfortunately, the current system, with some components nearing ten years in age, is showing signs of physical deterioration. It is now time to consider replacing some components and adding new facilities in order to insure that the Illiac system will remain responsive to the needs of its research community over the next five years.

II.A. Current System Configuration

The IAC computing facility consists of the Illiac IV computer, a Tenex based support system, and various other systems, such as a B6700 computer. The ILLIAC IV is an array processor consisting of sixty-four processing elements under explicit control of a central control unit; the control unit broadcasts instructions for the processing elements to execute. The Illiac processor has two memory components associated with it: a local memory and an extended memory. The file storage memory for the Illiac is currently part of the Tenex file system. Each of these components is generally regarded as inadequate to meet the expected demands on the Illiac during the next five years.

The Illiac has a one million byte (8.4 million bits) fast access local memory called the PEM -- the Processing Element Memory -- which is directly addressable by the processor. Most users have found that this memory is too small for efficient utilization of the machine's computational power (up to 40 million 64-bit floating point operations per second). The major reason for the inefficiency lies in the fact that the large program data bases reside on the extended memory, with the local memory serving as a buffer area, and the buffer area available in the local memory is usually not large enough to mask all the delay incurred in accessing the extended memory. In addition, there are some programs for which a natural partitioning of the data base (for example a plane of a three-dimensional grid) will not fit in the available storage in local memory, and so an artificial sub-partitioning is employed with the attendant increase in program development time and in program execution time.

The current extended memory of the Illiac is a fixed-head disk which has been specially modified in order to support a transfer rate of 500 million bits per second. The capacity of the disk is around half a billion bits, which for several applications is too small. In addition, the disks themselves are showing signs of increasing deterioration which manifests itself in increasing maintenance costs and decreasing reliability and availability.

Currently, there is no special file system for the Illiac; Illiac files are stored in the Tenex file system in the central support system. This file system has a present capacity of about three billion bits. Since the immense speed of the Illiac makes the processing of very large files feasible, it is estimated that the on-line storage needed to support the Illiac is around eighteen billion bits; thus the storage capacity of the current system is severly limiting. The present system supports a transfer rate from this file system to the Illiac of slightly over one million bits per second. Even with the presently planned enhancement to this file system to increase the transfer rate to five million bits per second, the Tenex file system is clearly inadequate to support the processing of large data sets on the Illiac.

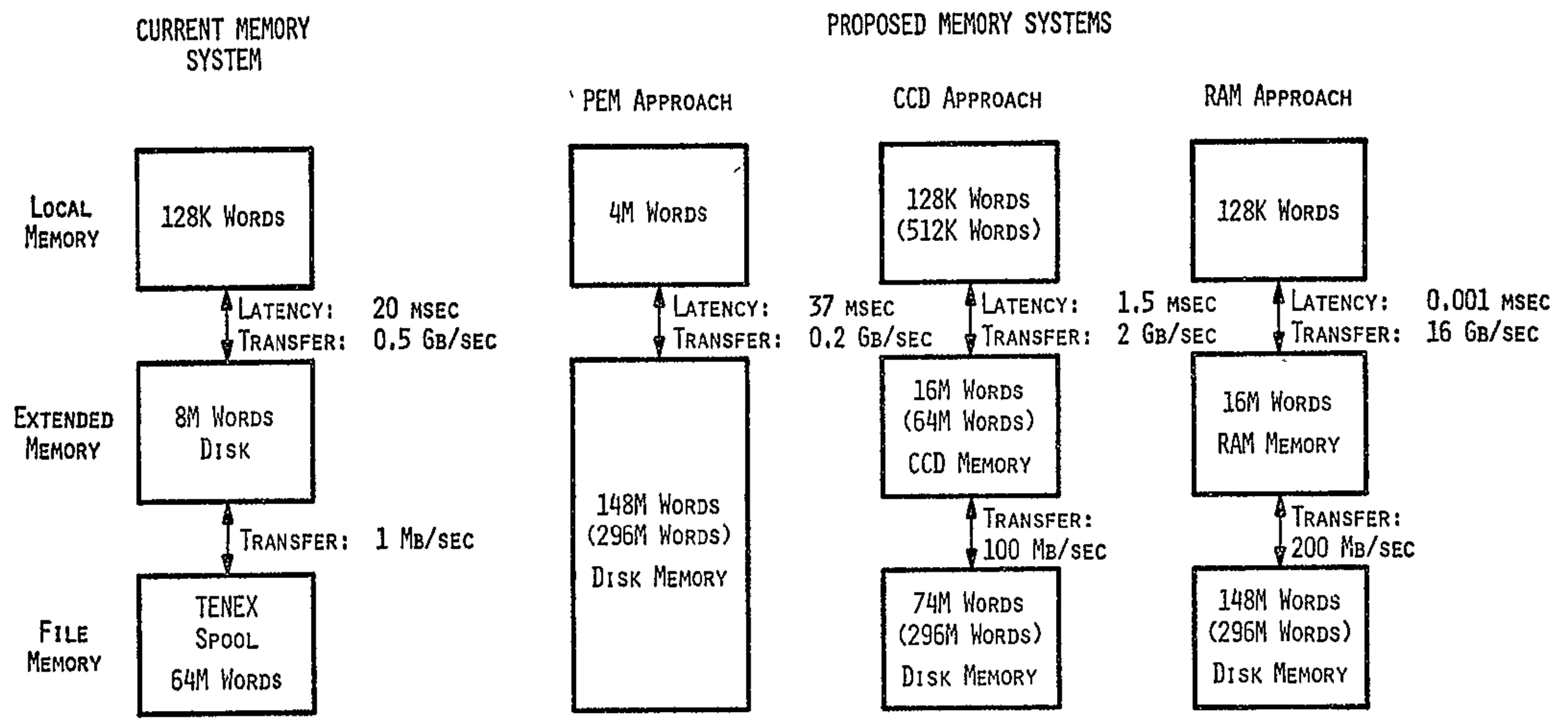
In summary, the Illiac memory system has five areas of weakness: the low bandwidth between the system and the outside world, the lack of adequate file storage space in the system, the inadequate bandwidth between the file system and the array, the transfer characteristics between the extended memory and the local memory (which result in either inefficient use of the machine or significant program development time to reduce this inefficiency), and the declining amount of extended memory available for user programs. For the current users, two of these weaknesses receive most attention: the size of the extended memory and the difficulty in using this memory efficiently. Because of its deteriorating state, the current extended memory will be replaced, and a satisfactory replacement will accentuate the current limitations of the file system.

By examining current codes on the Illiac and considering the planned development of these programs, it is possible to generate guidelines as to necessary or desirable characteristics of future memory systems to support these research projects. Briefly, the major characteristics are size of local and extended memory and the transfer rate between them. A transfer rate of 200 million bits per second is required, with a rate closer to one billion bits per second preferable. The local memory should be large enough to permit adequate buffers to compensate for the latency involved in accessing the extended memory; a ratio of at least 800 thousand bits of local memory for each millisecond of latency of the extended memory is dictated by the Illiac's current application programs. The capacity of the extended memory is governed by the size of the problems to be solved on the Illiac; a minimum size for the extended memory is one billion bits (sixteen million 64-bit words) with four billion bits preferred (sixty-six million words). This study identified three different approaches to memory systems enhancement which would meet these requirements.

II.B. Alternative Approaches to System Enhancement

A review of the currently available memory technologies indicates that there are three approaches to enhancing the Illiac system which are technologically and economically feasible and still meet the performance characteristics demanded of the Illiac memory system. Basically there are two philosophies to enhancing the Illiac memory: concentrate on the local memory or on the extended memory. Emphasizing the local memory leads to one design; namely, a large local memory and a mundane (by Illiac standards) extended memory. Concentrating on the extended memory instead, two solutions are considered: a memory using charged-coupled devices for a high performance disk-like replacement or a random access memory using high integration semiconductor components. Both of these extended memory replacements also require a staging memory to buffer transfers between the support system and the Illiac, thus an on-line file system dedicated to Illiac files is included in each proposed memory enhancement. In the first approach, the extended memory can also serve these functions. These three approaches and their relative merits will be briefly described; they are summarized in figure II.1.

The first approach consists of a four million word local memory and a disk-based extended memory. The extended memory has an initial capacity of nine billion bits and can be increased in increments of 2.25 billion bits; a second phase calls for doubling the initial capacity. The transfer rate between local and extended memory is a maximum of 350 million bits per second, but the sustained rate would probably be closer to 200 million bits per second. The extended memory would also be used for buffering file transfers from the Illiac system to the external world, and it would also maintain Illiac files on-line for periods of one to two weeks (such as temporary files created for re-start purposes, or between runs in a multiple-run experiment). The cost of the initial configuration is approximately 1.6 million dollars and the second phase is about 200 to 500 thousand dollars, the later estimate depending upon whether error correction is found to be needed in the local memory. One would expect that most of the programs for such a system would restrict their data base so as to reside entirely in the local memory, with the extended memory being used to hold output and re-start data. Only a few programs would be expected to use the extended memory to hold the active data base of a program, and the large local memory insures that there would be adequate



COST:	PEM APPROACH	CCD APPROACH	RAM APPROACH
1ST PHASE 1979	$ 1.6M	$ 1.5M	$ 2.4M
(2ND PHASE 1980-81)	($ 0.4M)	($ 1.6M)	($ 0.2M)
TOTAL	$ 2M	$ 3.1M	$ 2.6M

LEGEND: K = 1024, M = MILLION, MSEC: MILLISECOND,
WORDS = 64 BITS, MB = MEGABIT = MILLION BIT,
GB = GIGABIT = BILLION BIT

Figure II.1.

Current and Proposed Memory Systems

memory available for buffer areas. The transfer rate is less than half the current local to extended memory transfer rate, although this degradation would have a relatively minor effect on the total execution of most current programs. This approach is called the PEM approach since it concentrates on the local memory of the Illiac.

The second approach consists of a sixty-six million word extended memory, a half million word local memory, and a combined buffer memory and file memory. The first phase calls for a sixteen million word extended memory using charged-coupled devices and a five billion bit buffer/file memory using disk technology. The second phase consists of increasing the extended memory to the full sixty-six million word capacity and increasing the buffer/file memory to eighteen billion bits. It also includes enhancing the local memory to half a million words. The performance of the extended memory is such that problems users currently have using the disk-based extended memory will be reduced by an order of magnitude during the first phase and further alleviated during the second. The cost of this approach is approximately one and a half million dollars for the first phase and 1.6 million dollars for the second. This approach is called the CCD approach since this is the memory technology chosen for the extended memory.

The third approach is a sixteen million word extended memory which has the same characteristics as the present local memory. It also has the same buffer/file memory as the PEM approach. This extended memory would be constructed out of high density random access semiconductor memory chips. The second phase of this approach would be limited to increasing the buffer/file memory from nine to eighteen billion bits. By using a random-access extended memory, this approach virtually eliminates user objections to the current extended memory. The cost of this approach is approximately 2.4 million dollars for the first phase and 200 thousand dollars for the second. This approach is called the RAM approach since this is the memory technology chosen for the extended memory.

The schedule for all three approachs are virtually identical; the first phase being completed during 1979, and the second phase proceeding during the 1980-81 time frame.

II.C Relative Merits of the Enhancements

The three approaches differ primarily with respect to initial cost, final memory , size, system impact, effect on program development, and demand on the Institute's resources for hardware development. The technological risk involved in the three approaches are acceptable: there is essentially no risk involved in the RAM or PEM approaches. The CCD approach has the greatest amount of technological risk associated with it, since the technology chosen for the extended memory has not yet been used in large scale memories. With respect to implementation risk, the RAM approach is significantly greater than the other two since it involves a much more complex coupling with the control unit.

The amount of in-house development required for these three approaches varies considerably. All approaches use the same system for the buffer/file memory, and the interface, although complex, is straight-forward. Ignoring this common task, the CCD approach requires no in-house development aside from specification, the PEM approach requires a modest involvement, and the RAM approach requires a considerable amount of in-house development.

The three approaches have different impacts on the current system software and on user's program development efforts. Again, the impact of the buffer/file memory on the current support system is roughly constant for the three approaches, and will be ignored in this discussion. The PEM approach requires negligible changes to the current system software. For the CCD approach, the rapid response time of the disk replacement will necessitate the current control mechanism being replaced by an I/O executive dedicated exclusively to managing transfers between local and extended memory. This will require the development of the appropriate system software. The RAM approach also requires considerable system software changes, since it involves adding an instruction to the current Illiac repetoire; changes must be made in the assembler and probably in the programming languages as well. As far as the impact of these three approaches on user code development is concerned, the PEM approach will be of most help to users whose problems fit in the four million word memory; for larger problems, about the same level of effort will be required as is now being expended. However, the capability to have much larger buffer areas may mitigate the amount of time which must now be spent during program design in order for a program to be reasonably efficient under the current configuration. The CCD approach, by significantly improving the performance characteristics of the extended memory, will greatly reduce the effort required to use the machine efficiently. The RAM approach should virtually eliminate the problems users experience with the current extended memory, except for those few larger problems which will not fit in the extended memory and must use the buffer/file memory for part of their active data bases.

As a final point of comparison, the relative performances of the three memory systems varies depending upon the size of the active data base being processed. The target Illiac applications for the fully enhanced Illiac system can be grouped into four categories according to the size of their data base: those problems with less than one half million words, those problems with one half to sixteen million words, those problems with sixteen to sixty-six million words and those problems with greater than sixty-six million words. Problems in the first category are best served by either the RAM or CCD approach with a 5 to 50% degradation expected going to the PEM approach. This is due to the fast local memory provided with these two approaches. Since the fast local memory in the CCD approach is one half verses the one eighth million words of the PEM approachs about 10 to 50% better performance would be expected of the CCD approach in this range. Problem data bases in the second category, those in the one half to sixteen million word range, would best be served by a RAM memory system. A 5 to 10% degradation is expected going to CCD since this is the amount of time most programs will lose moving data between the local and extended memories. A degradation of up to 50% should be expected with the PEM approach for problems under four million words; above four million words the degradation would be 100% since part of the data base must then be stored on disk. Problem data bases in the third category, in the sixteen to sixty-six million words range, would perform 100% better with the CCD approach over RAM and PEM since data bases in the latter cases would be stored on disk. For the last category, over sixty-six million words, the picture reverses with both the RAM and PEM approaches performing 25 to 30% better than the CCD approach. This is due to the fact that in the CCD memory system the extended memory has only two high performance disk controllers instead of the four used in both the PEM and RAM approaches.

In summary, all three approaches increase the reliability and availability of the Illiac system while significantly reducing a major cause of the inefficient use of the Illiac and also reducing the high cost of program development. With respect to cost and technological risk, the three approaches are comparable although their differences are perceptible. With respect to the amount of system software changes, the CCD and RAM approaches require significantly more work than the PEM approach. There are four major differences in the three approaches. First, in respect to in-house development and required modifications to the Illiac (and the resultant unavailability of the Illiac), the RAM approach is significantly more demanding on the Institutes resources than the other two. Second, the RAM approach is significantly more expensive for a minimal configuration (the first phase) and the CCD approach considerably more expensive for the complete system. Third, with respect to the size of the data base which can be processed without having to resort to the complexities of using a disk base memory efficiently, the PEM approach is the most restrictive, the RAM next most restrictive, while CCD is least restrictive. Finally, with respect to the complexity of program design, PEM and RAM approaches are roughly comparable, with the CCD approach being significantly greater, and the PEM approach significantly greater still for large programs using its buffer/file memory to hold active data sets. All three approaches are feasible and significantly improve the memory system characteristics; the selection of one approach over the others should depend upon the relative importance of these last four differences.

III. Current System Status and Use

This chapter presents the technical background used in evaluating the various memory enhancements to be discussed in the next chapter. It is the product of a multi-disciplinary approach by the Institute to examine the current memory system and the way the memory is currently being used. The critical factors involved in system performance are identified, and an attempt is made to anticipate the potential bottlenecks of future use of the system. The organization of this chapter and the major results are summarized as follows.

The first section gives an overview of the system organization, both the hardware and the software. The current status of the hardware is briefly discussed: the basic reliability of the processor and local memory continues to improve, but incipient failure modes of the local memory indicate that replacement of the printed circuit cards is likely to be required within a few years. The extended memory, being a disk, is showing the inevitable effects of wear on physical devices, and an estimated lifetime of more than two years would be optimistic. The design of the system software for managing data movement in the system is such that alterations to the various physical components of the memory system are expected to have only localized impact on the remainder of the system software. Currently, data movement into the system and within the system is rather limited. Faster input devices are available; however, data movement within the system will probably be a major bottleneck unless steps are taken to replace the hardware and re-organize the components. The section concludes with a discussion of the scheduled enhancements to the Illiac processor to increase its computing power: an increase in processing speed of up to a factor of two is expected during the next year.

The second section surveys some of the major codes which have been developed for the Illiac, plus some possible future developments of these and other codes. From the point of their memory usage characteristics, these codes could process their data-base at rates of 10 to 100 million bits per second, with 60 million bits per second being typical. However, the latency of the disk, and the transfer rate between disk and local memory, cut this rate by a factor of two or more. Data bases for current Illiac problems range from half a million to ten million 64-bit words; this is often due to the size of available memory (for the data base and storage of output data). Removing this limitation would, of course, increase the demands on the file transfer capability of the system.

The third section presents a model of the Illiac memory system. Using this model and the surveyed codes, guidelines can be developed for the characteristics of acceptable memory replacements. The bandwidth between local and extended memory should be at least 200 million bits per second, and preferably between 600 and 1000 million bits per second. And it should be possible to load the extended memory from the support system at a rate greater than 6 million bits per second; this requirement indicates that a high bandwidth, on-line file storage facility directly connected to the extended memory is desirable.

The fourth section discusses four miscellaneous issues which affect the evaluation of memory system enhancements. The first concerns the sizing of the array memory. The local memory should be large enough to reduce the complexity of user codes (and the attendant high development costs) which results from a local memory too small for the natural formulation of user problems. The best estimate of a lower bound for local memory size is half a million words. The extended memory should have a capacity of at least one billion bits, although applications exist which are well suited to the Illiac architecture except that they require four billion bits of storage for their data bases. Second, semiconductor memories of more than half a million words have been shown by field experience to require the capability of detecting and correcting random errors, and this capability may be needed in an enhanced memory system for the Illiac unless significantly more reliable memory chips are used. Third, the concept of a virtual memory for the Illiac is discussed, together with a possible implementation and an estimate of its performance. A virtual memory is the popular solution to reducing the cost of program development since it relieves much of the burden from the programmer to explicitly move data around in a hierarchic memory system, such as the Illiac's; however, this has usually been accompanied by considerable degradation of processor execution capability. And fourth, in order to alleviate the problems of manual handling of large files in the current system, a reliable, on-line file storage facility is needed. Estimates of the capacity of such a storage facility indicates that at least eighteen billion bits are required.

The last section reviews the major weaknesses of the present system and indicates the various guidelines developed in the preceeding four sections for improving the system performance. This final section, then, presents a summary of the major conclusions of this chapter.

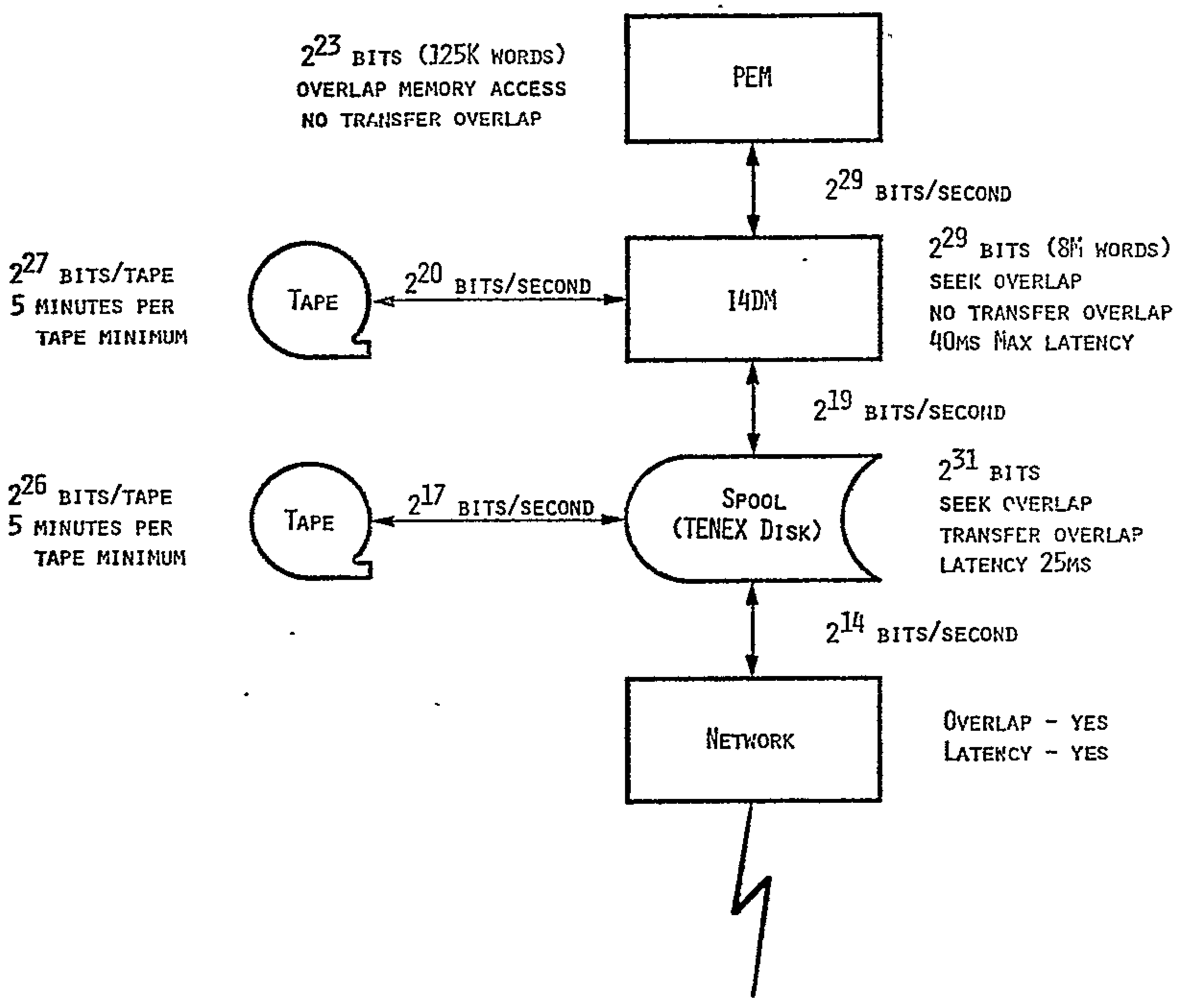
III.A. Present System Configuration

The Illiac IV memory system includes all memory in the IAC system used by Illiac jobs. As such, it consists of both the array memory itself and the support system memory. This section presents a description of the hardware and system software involved in the memory system. It begins with an overview of the system and its current status, then proceeds with a more detailed discussion of the array memory and the support system. It concludes with a description of the Illiac processor and the possibilities of increasing its execution speed since a faster processor will place greater demands on the memory system.

III.A.1. System Overview

The Illiac memory system can best be understood by consulting Figure III.1. This diagram shows the components used for data storage, and specifically does not show data transfer media. It depicts the Illiac memory system as it appears to the users. The current memory capacities and the maximum data rates which the hardware provides are indicated in the figure; sustained ransfer rates are somewhat less and depend upon system load. For a user, the ultimate consideration is that the Illiac system throughput is currently limited by an input/output rate of about one megabit (million bits) per second if the data go ultimately to tape; for strictly network users the rate is even less. This could be increased to five megabits per second by using higher performance disks and tapes now commercially available. On-line file storage in the IAC system is currently Tenex disks (called SPOOL in the figure); there is no on-line mass storage device in the Illiac memory system. There is some question whether, in the absence of such a device, increases in memory size and processor speed will significantly increase the effectiveness of IAC's support of the research community.

When a data set is to be processed by the Illiac, it is moved onto the Illiac's memory, either from the Tenex disks or directly from tape. Large data sets are placed on the extended memory (Illiac IV Disk Memory -- I4DM); the program and small data sets are placed in the processor's local memory (Processing Element Memory -- PEM). Data on the extended memory must be brought into local memory for processing; this movement is under the explicit control of the executing program. The Illiac is currently operated in a dedicated fashion: the array memory resources (PEM and I4DM) are assigned to one user at a time; there is no multi-programming use of the Illiac.



NOTE: TRANSFER RATES LISTED ARE MAXIMUM RATES

Figure III.1.

Current Illiac Memory System Characteristics

The philosophy behind the implementation of IAC's data movement system is the central memory concept (see Figure III.2). The central memory is used to buffer data between storage devices. Each device has an associated device control process, which can cause the device to perform input or output to or from pages in the central memory. Thus, a device to device transfer is decomposed into two phases: a source device to central memory transfer, followed by a central memory to destination device transfer. In this way, each device control process requires only the ability to manipulate the device and interface to the central memory. This provides a general mechanism for establishing a data transfer path between any two storage devices in the system with a fraction of the complexity required in a conventional architecture. Also, the installation or removal of a device is greatly simplified by the central memory concept.

Transfer of data occurs through buffer pages in the central memory; these pages are shared by the source and destination devices. The source device control process causes the buffers to be filled and then it signals the destination device that the transfer has been completed. When signaled, the destination device control process causes the buffer contents to be emptied into its storage device and it then signals that this phase of the transfer has completed.

The user process which requires data movement must set up the two device control proceses before the transfer can begin. This involves sending appropriate device addressing information to each device control process, acquiring buffer pages to be shared by the two devices, and specifying the communication path between the two control processes. After this initialization, the transfer is started for both device control processes and continues until all pages have been transferred.

III.A.1.a. Current Configuration

The preceding paragraphs describe the concept underlying the IAC data movement system; however, the actual configuration has not evolved to this idealized state. This is primarily due to the fact that a suitable central memory is not yet operational, and a generalized buffer management system has not been developed.

The current implementation of the central memory (see Figure III.3) consists of a number of smaller memory units. For each memory module, only a subset of the system devices can access it. This means that many transfer paths are not available, and some paths require an intermediate transfer to move data from a buffer in one memory to a buffer in another memory. The device for transforming a processor's logical address in the central memory into a physical location in one of the memory modules is called a Q1011.

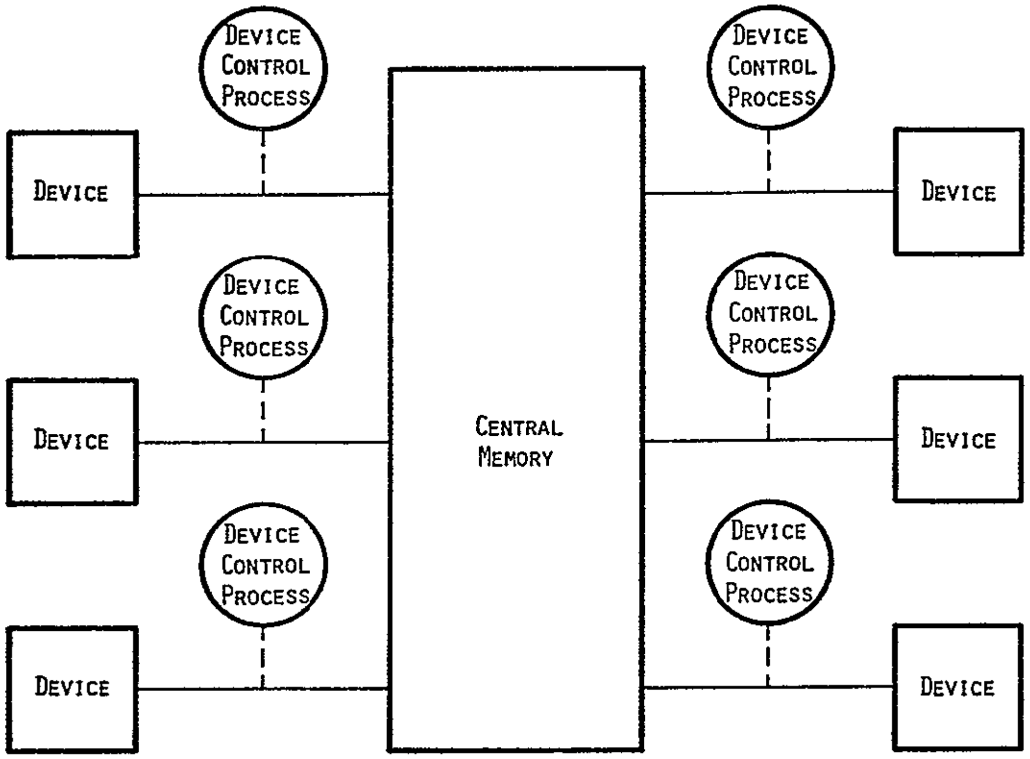


Figure III.2.

The Central Memory Concept

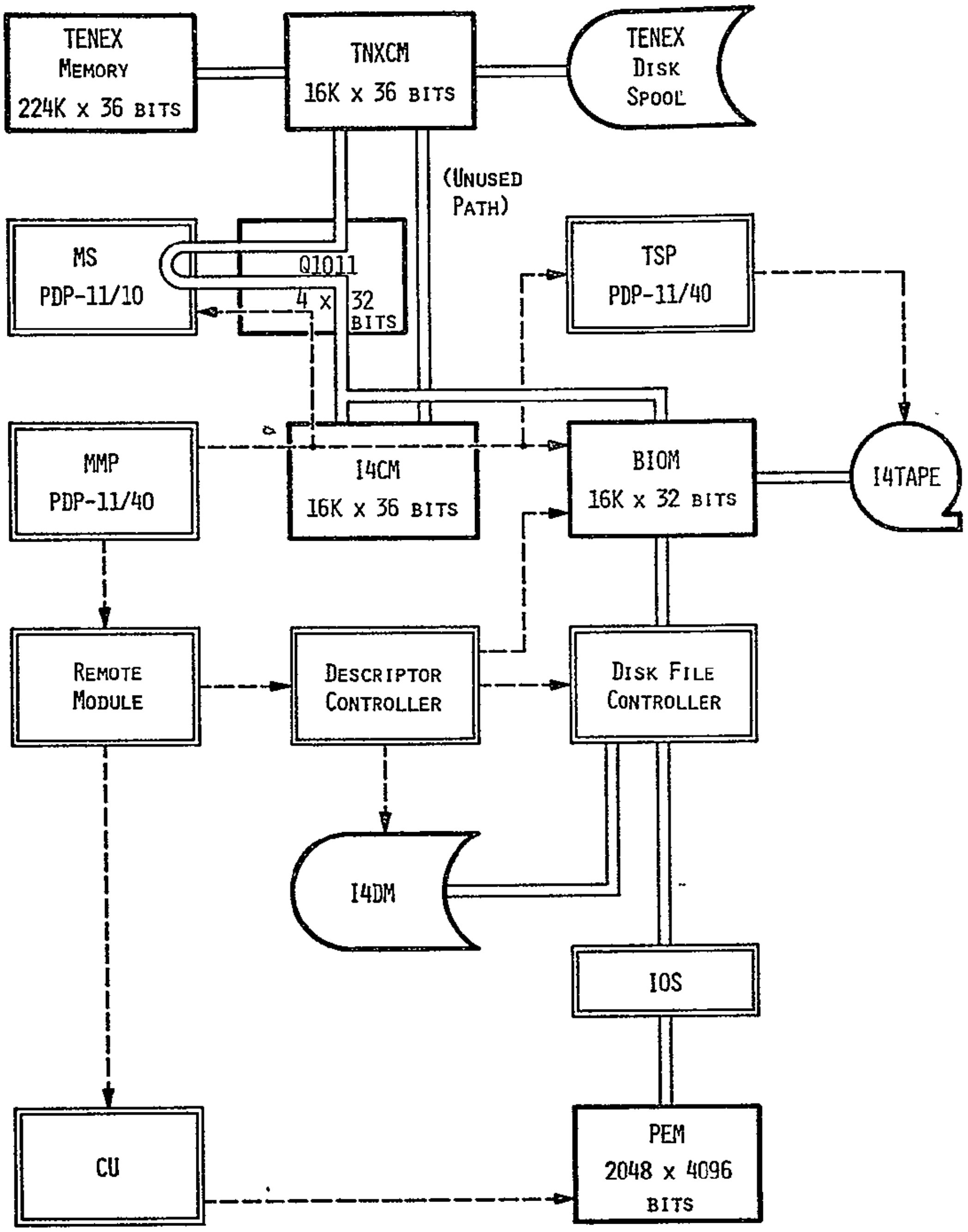


Figure III.3.

Current System Implementation

The general method of communication between device control processes discussed above is not implemented in the current system. Most device control processors communicate with only one other device control process, using a specialized communication protocol. The basic techniques are common to all of the methods used for interprocess communication. Because most devices are involved in only one transfer path, the buffer page assignments are fixed instead of dynamically allocated as indicated in the above discussion.

The current central memory interface to the Illiac subsystem consists of three 16K word memory units (where K stands for 1024): the Buffer Input/Output Memory (BIOM), the Tenex Buffer Memory (TNXCM), and the Illiac Central Memory (I4CM). No storage device has a direct access path to all of these memory units. TNXCM contains the Tenex buffers for data passage between the Tenex file system and the Illiac subsystem. The BIOM, considered as part of the Illiac subsystem, contains another set of buffers for data passage between Tenex, the Illiac and the Illiac Tape facility (I4TAPE). Control of these data movement processes is communicated through the I4CM unit. The Tenex system controls data movement between the TNXCM and the Tenex disk; the Memory Server Process (MS) controls transfers between TNXCM and the BIOM. Figure III.3 shows the interconnection of these devices in the current system implementation.

Within the Illiac subsystem, data movement is controlled by the Memory Management Process (MMP). It can exert control over the Illiac Disk File Controller (DFC) via the Remote Module and the Descriptor Controller (DC) in order to transfer data between the Illiac Disk Memory (I4DM) and the Processing Element Memory (PEM). The I4DM-PEM transfer proceeds through the DFC and the Input/Output Switch (IOS). The MMP also has the capability of delegating control to the Tape System Process (TSP) to transfer data between the I4DM and I4TAPE; data in this transfer moves through the DFC and the BIOM. The MMP has the further capability of moving data between either the I4DM or the PEM and the central system by using the BIOM. This is accomplished by passing control to the MS processor to handle the BIOM to Tenex transfers; it retains control over the DFC through the Remote Module and the DC. Note that the DFC, in addition to providing control over the I4DM, acts to establish a data path between any pair of the I4DM, PEM or BIOM units. In particular the I4DM to PEM path provides the very high transfer rate necessary for efficient Illiac utilization.

A transfer between Tenex Disk and either the I4DM or the PEM must be made through the intermediate TNXCM memory because Tenex disk cannot directly access the BIOM. The MS processor is used to bridge this gap in central memory because it can access both TNXCM and the BIOM.

There are three control processes which direct the movement of data within the Illiac System. The Memory Management Process (MMP) controls transfers between the BIOM, PEM and I4DM. The MMP Server process (MS) controls transfers of data between Tenex disk and TNXCM. MS also performs the actual movement of data between TNXCM and the BIOM. The Tape System Process (TSP) controls transfers between the Illiac Tape System (I4TAPE) and the BIOM. Each of the three device control processes, MMP, TSP, and MS, runs in a separate dedicated minicomputer.

III.A.1.b. Interprocess Communication

In the general central memory concept, a user process sets up a source control process and a destination control process, and establishes the communication between them. In the current configuration, the communication paths are predefined. The task of initializing the two device control processes is usually handled by one of the control processes. A Tenex user process communicates with the dominant device control process, which in turn initializes itself and the other device control process. In the case of the Illiac system, the MMP is always the dominant device control process.

The basic communication mechanism used in the current configuration is a pair of memory locations called slots. The slots are in the I4CM and each slot provides a full duplex communication path; requests are placed in one of the locations, and results are returned in the other. Slots are used for comunication between Tenex user processes and the MMP, between MMP and MS, and between MMP and TSP. A special slot mechanism for communication between an Illiac process and the MMP is implemented with Illiac hardware registers, rather than central memory locations.

The transfer between I4TAPE and I4DM is initiated by a user process in Tenex. This process communicates the specification to the MMP, which initializes both TSP and itself for a transfer using the BIOM as the central memory buffer.

Data transfer beween Tenex Disk and the I4DM is a bit more complicated. The Tenex Disk can transfer data only to TNXCM; an intermediate process, MS, is used to move the data between TNXCM and BIOM. The Tenex user process communicates with the MMP, which initializes the MS process and itself for a transfer using the BIOM as the central memory buffer. A transfer between Tenex Disk and PEM is similar, the only difference being that the MMP directs data to the PEM instead of to the I4DM.

The transfer between PEM and I4DM is initiated by a process running on the Illiac. For efficiency, the transfer is made directly between the devices without using a buffer. The MMP is the device control process for both the PEM and the I4DM; its role in the transfer is mainly to initiate the transfer. After converting the logical area name to a physical location on the I4DM, the MMP constructs a file descriptor for the Descriptor Controller (DC). The DC is the part of the array's I/O Sub-System (IOSS) which actually controls data movement between I4DM, PEM and BIOM.

Except for the PEM-I4DM transfer, all data movement must be initiated by a Tenex process. This implementation maintains Tenex as the central resource allocation facility. To allow an Illiac process to initiate data movement, a companion process to the Illiac (CMOVE) is available during an Illiac run. The Illiac process communicates with CMOVE through a message system managed by the MMP. The CMOVE process uses the data movement control described previously for Tenex initiated transfers.

III.A.1.c. Current Hardware Status

The current state of the hardware components will have an effect on the planing of enhancements. These can be briefly summarized as follows.

Local Memory (PEM)

An increasing number of problems are being isolated to micro-cracks in the printed circuit boards and this may require their replacement en masse at some future date. The chip failures are at a tolerable rate; however, they are no longer manufactured.

Processing Elements (PEs) and the Control Unit (CU)

Although the processing elements are the most failure prone devices, simply because of numbers, their reliability record has been constantly improving. From July 5 to 31, 1977, for example, only 17 of the 214,256 chips in the PEs failed. The chips used in the PEs are no longer manufactured, but in general the spare level is high. It may be necessary to go into prototype chip production for the few chips whose supplies are low. The record of reliability for the Control Unit has also been constantly improving; during the same period in July, for example, only 5 of the 17,136 chips in the CU failed.

Disk Memory (I4DM)

The extended memory of the Illiac is currently implemented using disk technology. It is the least controllable resource in the system. The disks are old but are being refurbished. This component has an estimated remaining life of from two to three years but has the disadvantage that it is not expandable at reasonable cost. It is also probable that as this mechanical system ages, its reliability will decrease and its maintenance costs will increase. The available capacity of the disks has been declining during the recent years. Of the originally implemented 16 million word memory, only 8 million words were ever on-line, and the last year has seen the capacity continue to decline. By the first of this year, the capacity seemed to have stabilized at around 5 million 64-bit words being routinely available, although this is slowly increasing.

About two years ago the disks were synchronized and the electronics were modified and adjusted. Since that time the number of disks available to users increased to a high of seven disks; this process took about a year. In the last year the disks which had accumulated 50 to 60 thousand operational hours began to require bearing replacement. The number of available disks deteriorated to two or three before the required maintenance tools could be designed and fabricated. Two of the four disks currently operational are reconditioned ones. Four more disk units will be re-installed during the first half of 1978.

Central System

The PDP-11's that are used for control processors have a very good reliability record. As the system throughput improves, however, it may turn out that their speed must be augmented by special hardware or newer devices. However, the bulk of the processors will adequately perform their functions. The PDP-10 also has a good record. The only consideration for it is throughput. As demands on the IAC system increase, enhancements may be required. The current PDP-10's at IAC are KI models, which are about two and a half times as slow as the fastest PDP-10 (the KL model). But since the IAC system uses a software paging device and relatively slow memories, the difference in available computing power may be closer to a factor of five. Using a faster memory, for example, would increase the available processing power by approximately fifty percent.

The Tenex file storage devices, both tapes and disk, have the worst record of reliability of the central system components, requiring two to four hours a week for preventive maintenance and emergency repair. Overall, their record is good, however. They were adequate for the environment of the Institute a year ago: in-house development and friendly Illiac users. But as the Institute moves toward general availability of Illiac service, an improvement is appropriate, with the present hardware remaining for the in-house development system.

Overall, the hardware in the Illiac system is in fair to good condition. However, the effects of age on the mechanical devices such as disks (especially I4DM) are beginning to impact the availability and usefulness of the system to an appreciable extent. Also, as the speed of the Illiac itself improves, the demands on the system (especially the peak demands associated with moving Illiac data files around: the demands most visible to the user community) will increasingly reveal the already noticeable weaknesses of the system. The remainder of this section will treat the individual components of the memory system in greater detail, discussing the limitations they place on possible enhancements.

III.A.2. Illiac Array Memory System

There are three major components to the memory system of the array itself (called the IOSS -- the I/O Sub-System). These are the PE memory (PEM), the extended memory (I4DM), and a buffering memory to the support system

(BIOM). Controlling the transfer of data among these memories is the Descriptor Controller (DC) and the Memory Management Processor (MMP). These devices are described in more detail in this section, their performance characteristics are given, and the limitations their current implementations place on future enhancements are indicated.

Data can be transferred between any two of the three memories at a rate of 500 megabits per second. These transfers are initiated by the MMP, a PDP-11 model 40. There are request ports to the MMP for these transfers. Both an Illiac program and processes running on the PDP-10 or other PDP-11s can request these transfers.

Processor Local Memory (PEM)

The Illiac's local memory is 2K rows of 4096 bits. The present cycle time and access time are ten clock cycles; the present clock rate is 12.5 megahertz. It is likely that the cycle time will be reduced to two clocks while the access will be reduced to four clocks. This implies that only operations which can be explicitly overlapped will be able to use the two clock cycle, other operations will be limited by the four clock access. The clock rate will also be increased to 16 megahertz. The PEM proper can be increased in capacity while retaining these enhancements; various possibilities are considered in Chapter IV.

The PEM has a number of requesting ports. A request on any port will use 4096 bits of PEM bandwidth regardless of the number of bits used per reference. For example, the I4DM is specified as being a 500 megabit per second path -- that is, it requests 1024 bits every 2 microseconds. Actually, each 1024 bit reference accesses 4096 bits of PEM. Each requesting port to the PEM is similar, only the bandwidth utilization varies.

The present PEM has 32 thousand chips of 256 bits each. A new PEM design could use up to the same number of chips and still have enough extra room for error correction. Using 32 thousand chips (plus 4 thousand for error correction) the memory could be 4096 bits by up to 32K rows. This is the limit if the current architecture is kept. The Illiac instruction format allows for 16 bits to specify an address with PE local memory, and this places a natural limit on the addressable size of PEM, namely 64K words per PE or four million words for the entire local memory. In addition, the address lines from the PE to its memory are currently 11 bits wide, although provision has been made to accommodate a 16-bit address cable. This means that a local memory larger than the current memory will require additional, although minor, modifications to the hardware. Finally, it should be noted that there is adequate power and cooling capacity to support up to 40 thousand of the memory chips that are now commercially available and meet the speed requirements.

Another PEM limitation is the bus to the I/O sub-system (IOSS) which connects the PEM to the I4DM and BIOM. It is 1024 bits out of a 4096 bit data register. The multiplexing required limits the IOSS bandwidth ultimately to 1024 bits every Illiac clock. The best possible transfer rate is therefore 16 gigabits (billion bits) per second with a 16 megahertz clock. However, this limitation could be circumvented if the bus were jettisoned entirely, for example if each PE had its own slice of extended memory directly associated with it.

Extended Memory (I4DM)

By the early part of 1978 there were four disks in operation in the I4DM, for a maximum available memory of five million words. With the refurbishing program under way, this may increase by as much as a factor of two for the remainder of the year.

The disks require 40 milliseconds per revolution, so that on the average 20 milliseconds is required between the time that a random I/O request is made and the requested data is available. This latency factor can be significant in determining the total run time for some user programs; section III.B treats this issue in more detail. In general, disk latency is a major issue for most of the users of the Illiac, and it is often cited as a major hinderance in developing codes for the machine.

The Illiac disk is addressable to a quarter of an Illiac page (a page is 64K bits). Each band of Illiac disk has 300 Illiac pages. It is a limitation of the disk hardware that between any two transfer requests at least 3/4 of an Illiac page must be skipped. However each request may be for a large but contiguous portion of a disk. The current I4DM places no restrictions on a possible replacement, other than that the communication protocol with the DC remain the same. By moving the MMP/DC functions into the CU/extended memory enhancement, even this limitation is removed. This report assumes that there is no significant restriction on replacing I4DM.

BIOM Operation

The BIOM is a true multiport memory. It has a capacity of eight Illiac pages, or 512K bits. A fast port, capable of the IOSS 500 megabit transfer rate, accesses the memory over a 128-bit path. The other four ports access the memory as 16K words of 32 bits each. An access on these slower ports takes 5 clocks of 200 nanoseconds each. The BIOM has three devices connected on these slow ports: the B6700, the Descriptor Controller (IOSS), and an interface from a PDP-10 Memory (commonly called the ME-10). The connection to the B6700 is used for data transfer between the PDP-10 and the B6700; very soon an alternate path between these two devices will be provided, so the interference this causes to current array performance need not be considered in evaluating enhancements.

The Descriptor Controller, which controls all IOSS data paths, stores its 'result descriptors' in the BIOM via its memory port. The current software permanently allocates one Illiac page of the BIOM for the storage of these descriptors. An additional page is reserved for transfers to the B6700. Therefore only six Illiac pages are available for buffering the transfer of data between I4DM and the central system. The control path of the Descriptor Controller is the only control path which causes enough conflict to merit mentioning in detail in this report. The dedication of one quarter of the BIOM to the descriptors described above is a significant factor in determining the maximum throughput rate through the BIOM and limits the system to I4DM transfer rate to ten megabits per second. The current system has a peak transfer rate of two megabits per second, so the BIOM is quite adequate for the present configuration. However, the BIOM must be replaced for transfer rates over ten megabits per second to be possible.

The ME-10 is a modified PDP-10 memory module which allows standard PDP-10 requesting devices to access the BIOM. The BIOM appears to be the upper 16K words of a 32K word ME-10. The word size of the BIOM is 32-bits rather than the 36-bits of the PDP-10; thus for the BIOM half of the module, only the most significant 32 bits of data respond; the least significant bits must be all zero for parity considerations. The PDP-10 processor, as well as other processors described below, can access the BIOM through this modified ME-10.

MMP Operation

The MMP provides the operating system of the Illiac (it replaces the B6700 in the original design of the Illiac IV). As such it does no explicit data transfers. It can request IOSS transfers, access the communications sections of memory, operate the Illiac console, and respond to Illiac interrupts from user code (most often these interrupts are also IOSS transfer requests). The MMP is request driven; in the communication area of memory (I4CM) there is a set of request ports (called slots) with which a Tenex or PDP-11 process can make a request for action by the MMP. These ports have a response portion for the MMP to signal that the request has been completed; these response ports must be accessed by the requestor to determine the state of the operation. The Illiac program can also request MMP operations and obtain results through a separate interface. The performance of these interfaces in the present configuration is not a major bottleneck. Performance measurements of the I4DM-MMP operation indicate that overhead due to the MMP accounts for two to four milliseconds, which is usually only ten to twenty percent of the data movement time.

The major limitation of the MMP is that it is a PDP-11 and can only poll memory and respond to interrupts at minicomputer speeds. Presumably at some point the overhead here may become significant, especially if advances are made in other areas. The alternatives are to put the MMP functions into the Illiac control unit itself, or to use a fast microprocessor.

In order to make the interface between an Illiac program and the I4DM understandable and to increase the availability of the disk, a mechanism was invented to allow the Illiac user to access the Illiac disk using file-like area names. The conversion between these names and the physical disk is made by the MMP based on a file kept current on Tenex. This file is initialized by the ALLOC subsystem, which is in charge of assigning the physical areas on the disk. The file, called the I4D-file, can be larger than can be contained in the local storage of the MMP; this arises when complex maps of the disk are employed. The current MMP memory will hold a description of 13K pages of I4DM, which is about three times the number of I4DM pages currently available. Thus each page can appear in three different logical areas. For a larger disk area, such complex maps will no longer fit within the MMP local memory, and this local memory is almost as large as the PDP-11 can accomodate.

The high bandwidth among the memories in the IOSS is a major feature of the Illiac system: it permits the efficient manipulation of the very large data bases which occur in large scale scientific computing. However, the latency of the I4DM tends to make it difficult for many Illiac programs to realize this efficiency. The path to the outside world, the BIOM, is currently limited less by the size of its buffer memory than by the rate at which the support system can transfer data.

III.A.3. Support System

The Illiac support system consists of a PDP-10 processor with a Tenex operating system, together with several PDP-11 satellite processors to handle file transfers. The interface to the ARPA network is the standard IMP and will not be discussed in this report.

MS

The MMP Server (MS) is a process which runs on a PDP-11 model 10. It will transfer data between a Tenex System Memory and the BIOM. It manages the buffers in both memories, making disk requests to both responsible processors (PDP-10 and MMP). MS also provides enough added efficiency that the transfer rate between the two disks is now at about one megabit per second, which is an increase of an order of magnitude over the previous implementation in which Tenex controlled the transfer. The key to the improvement is an optimized transfer capability installed at the interrupt level of the PDP-10 disk routine. The details and timing of this interface are still under development, but MS is ultimately limited by its memory copy mechanism. The hardware used to move a block of data from one memory to another was not designed for this purpose originally and for this reason MS has a bandwidth limit of about 1.9 megabits per second.

I4TAPE

The I4TAPE process is very much like MS; it buffers transfers between the tape units dedicated to the Illiac and the BIOM. However, since there is no contention for the tape units as there is for the Tenex disks in MS, the data rate is limited only by buffering and the synchronization with the Illiac disk rotation. Realistic rates of 1.33 megabits have been achieved. The tape hardware transfers at 1.92 megabits per second, but gaps, start/stops, and disk latency reduce the efficiency. The overhead of the Illiac program making the request has not been measured, but it is estimated at about 10 seconds to initiate the process. The tape service is limited by the implementation that requires a Tenex process to handle each transfer request. This implementation was chosen so that the interface to the Illiac user would be simple and flexible.

PDP-10

The Tenex memory system is ultimately limited to 256K words by the addressability of the peripherals. The age of the drum and disk systems are the cause of this limitation; newer devices have a greater addressing range. The Tenex memory was recently increased by more than 100K words from the original 128K words. This more than doubled the amount available for users and decreased the idle time from the 30 to 50% level to the 0 to 5% level (measured during peak times with load averages in excess of 10.0).

There are three sub-systems which run in the Tenex environment and control data movement within the Illiac memory system: MOVE, RUN and ALLOC/DALLOC. MOVE controls data movement between the I4DM and either a Tenex file or I4TAPE. While the data movement is initiated by MOVE, direct control is maintained by the appropriate PDP-11 process: MMP, TSP, or MS. RUN controls the loading of the Illiac. Overall control is maintained by RUN, but direct control of program movement and status communication is maintained by the MMP. In addition, post-run dump is controlled by RUN, as is elapsed execution time monitoring. CMOVE, a sub-process of RUN, is maintained as a communications terminal for an Illiac program to control data movement involving the Tenex system. ALLOC/DALLOC controls the allocation and de-allocation of specific I4DM bands to area designators so that Illiac or Tenex processes can access these areas on a "by name" basis, rather than by physical location.

Modifications to the Illiac memory system would directly affect only ALLOC/DALLOC, provided the MMP, TSP, and MS interface protocols remain fixed. The various proposed memory enhancements in this report will have little impact on these Tenex sub-systems, with the possible exceptions of the introduction of an on-line file storage device and a large staging memory adjunct to the extended memory system. Either of these will require additional system control modules. However, the impact of memory enhancements on existing Tenex software is not of significant scale to warrant consideration in evaluating the proposed modifications.

The present disk and tape hardware could be replaced by hardware which has peak data rates of ten megabits per second. Having these devices, would afford the capability of approaching the peak data rates of which the system is capable. Thus it is possible that I/O into the system could improve by almost a factor of ten in a dedicated mode. However, in a shared memory multiprocessing system such as Tenex and the central memory concept, the average data rate will be lower than ten megabits per second, and a system improvement by a factor of about five is anticipated.

Summary

The file transfer rate within the support system is currently slightly over one megabit per second, thus movement of large Illiac files takes about one minute per million words (64-bit words), and this is well within a factor of two of the limitations of the current hardware and system design. One megabit per second is also the I/O rate between the support system and the outside world. Faster tape and disk units could improve the peak I/O rate by almost a factor of ten. To increase the transfer rate within the system itself requires faster processors (within the current implementation) or additions of special purpose hardware for the explicit purpose of supporting high block transfer rates for moving very large files.

III.A.4. Illiac Speed Enhancements

The processing speed of the Illiac determines the demands that will be placed on the memory system. In considering enhancements to the memory, it is therefore necessary to take into considerations the improvements that may be made in the execution rate of the Illiac. The following discusses the improvements that are being considered; the conclusion is that an increase in the observed speed of the processor could be as much as a factor of two, or better.

III.A.4.a. Description

The Illiac processor itself consists of a central Control Unit (CU) which fetches and decodes instructions and generates enable signals to control the execution of sixty-four processing elements (PEs). The CU is divided into two major units: ADVAST (advanced station) and FINST (final station). The role of ADVAST is to process instructions which deal with program control and to prepare instructions to be executed by the PEs. The role of FINST is to take the instructions intended for parallel execution by the array and to convert them into sequences of enable signals which actually control the PEs.

The CU was originaly designed to run at 25 megahertz (40 nanosecond cycle), but early in the design the speed was reduced to 20 megahertz. The PEM was designed to run at a 10 megahertz rate and be pipelined with respect to responding to memory requests. It can probably be run at 8 megahertz now.

The PEM rate is additionally limited by the fact that the operand select gating in the PE is used both for sending the address to the PEM from the PE and for receiving the data from the PEM. This constrains the fetch rate to once per two cycles (which fits well with the megahertz PEM rate). The current PEM access time is about 200 nanoseconds; with the various delays that are added in the PE by sending the address and receiving the data into the same register, the PEM access time is probably limited to four cycles at 16 megahertz.

The floating-point add and multiply instructions seem to be the most frequently occurring PE instructions, and indeed the PEs were designed mainly to be floating point add and multiply functional units. Consequently, PE arithmetic has little potential for increased speed-up. The original design called for a seven cycle add and a nine cycle multiply for 64-bit floating-point arithmetic. At this time the add also takes nine cycles because of some long signal paths in the PE. The two extra cycles were added to allow more time where the long paths are used. There is a possibility that at least one of the cycles in the seven cycle add can be eliminated and this would reduce the number of cycles to six, thus limiting the arithmetic bandwidth for this particular instruction to 170 million floating-point results per second (at 16 megahertz). This is an upper bound since it includes no time for fetching operands and saving results.

III.A.4.b. Speed-up of Simple Loops

It is instructive at this point to consider some simple code examples; these can indicate what processor speed-up is possible. Consider the following loop of code:

```
LOOP: LDA   X(ACO)    ; load accumulator from X(indexed by ACO)
      STA   Y(ACO)    ; store accumulator
      TXLTM ACO,LOOP  ; increment address pointer and branch
```

The machine currently takes sixteen cycles to execute this loop (per time through the loop) which consists of nine or ten cycles for the load and four or five for the store, with a dead cycle (which is included in all FINST instructions) between each. This amounts to a transfer rate of 4096 bits (one row) per 1.28 microseconds, or 3.2 gigabits per second.

Were the memory fetch to be overlapped in FINST so that a new memory cycle could be prepared before the old one finished, the same loop could take as few as six cycles, which is a speed-up of a factor of 2.5 times. In this case an example of how to take advantage of such overlap of memory requests is as follows:

```
LOOP: LDA   X(ACO)
      LDB   X+1(ACO)     ; load B register
      STA   Y(ACO)
      STB   Y+1(ACO)
      TXLTM ACO,LOOP
```

The effective rate is increased because the PE can fetch the next word while the first word is arriving. The new rate is two rows per eight cycles or a speed-up of a factor of four. Such programming currently results in no program speed-up because the data must be in the receiving register before FINST will begin execution of another instruction. The new transfer rate would be (using 16 megahertz clock rate) almost 16.4 gigabits per second or a factor of 5.12 increase. Incidentally, ADVAST now takes 15 cycles in the former loop (2 for each indexed PE instruction and 11 for the TXLTM) and 19 cycles in the second loop, so quite a few enhancements to ADVAST would have to be installed in order for FINST to be used effectively in these cases. For example, a one cycle indexed PE instruction and a four cycle TXLTM would be sufficient. Both of these enhancements are feasible.

Consider another loop:

```
STARTUP: LDS   W(ACO)    ; pre-load S register
   LOOP: LDA   S         ; load accumulator from S
         ADRN  X(ACO)    ; add X to accumulator
         ADRN  Y(ACO)
         LDS   W+1(ACO)
         STA   Z(ACO)
         TXLTM ACO,LOOP
```

This is a rather trivial Z=W+X+Y and contains a pre-fetch of the next W to take advantage of full overlap of memory operations with arithmetic. There are four memory operations (three fetches and a store) which account for most of the 38 clocks that this loop currently takes -- the 18 cycles in the two additions are hidden. At 12.5 megahertz (the current frequency) this amounts to about 42 million floating-point results per second. The best possible performance for this loop is 14 cycles at 16 megahertz, or 146 million floating-point results per second, for a speed-up of a factor of 3.5.

Thus considering just the throughput gains to be made from increasing the memory access rate, and a faster clock rate, it appears that an increase in processing speed of upwards of a factor of two is possible. The rest of this discussion concentrates on enhancements to the processor itself.

III.A.4.c. Additional Processor Enhancements

The processing of instructions in the control unit starts in the instruction look ahead portion, where blocks of sixteen instructions are fetched from the PEM. Prefetching occurs simultaneously with other ADVAST activities, but JUMP/SKIP instructions cause ADVAST activities to wait. It is not known how much interference such waiting introduces into programs, or how long ADVAST waits for FINST queue positions to become available. It is assumed that this interference has little effect on determining the processor speed.

Once the instructions arrive in ADVAST, there are many extra cycles in the most common instructions. For all simple operations such as integer addition, leading ones detection, logical operations, and shifts, one clock should be enough. References to ADVAST local memory should take two cycles. Of course, instructions which use functional units in an iterative fashion must take more than one clock. A few example instructions should indicate how much is to be gained in removing extra clock cycles from these instructions. The instruction to load an accumulator from local memory takes three cycles when not indexed and five cycles when indexed. They should take two and three clocks, respectively. This kind of speed-up approaches a factor of two and, since these instructions occur frequently in most programs, this should have a noticeable effect on program execution speed.

There are other ADVAST instructions whose execution time can be reduced. Fetching data from PEM is not now overlapped, and instructions which manipulate processor enable bits currently wait longer than necessary for PE data. Finally, if the local memory of ADVAST is enhanced, references to this memory could take only one cycle if a sufficiently fast technology is employed. Enhancements to the instruction set for managing extended memory improvements would not significantly affect ADVAST except for the obvious changes to the control logic for those activities that ADVAST may perform to prepare the instruction for FINST.

In the FINST section of the control unit, several possibilities for processor enhancement arise. Considering the importance of the PEM resource, an effort is now underway to streamline the use of PEM by FINST. At present FINST overlaps a memory operation with a previous arithmetic operation, but it will not overlap two memory operations. Successive memory operations now take ten clocks each. The target is to reduce each memory operation to four clocks and overlap them if possible. This was discussed earlier with respect to PEM usage.

In an attempt to make the Illiac operational, certain modifications were installed which made the machine work correctly but also degraded the performance. In an attempt to increase performance, these modifications are being systematically removed and the design errors corrected. Examples of this process are removing the extra cycle between all FINST instructions, reducing the route steps from two clocks to one, and permitting routing to be overlapped with computing. The present route instruction is not overlapped because it inhibits the processing elements' clocks. In an effort to produce a two clock period data pulse with no spikes, the route instruction inhibits the intermediate spike-producing clock signal. An overlapping instruction executed during this inhibited clock would have no effect in the PE. To prevent errors of this type, no PE instruction execution is permitted to occur during a route instruction. However, a one clock route would not need to inhibit PE clocks and hence could be overlapped with other PE instructions.

III.A.4.d. Summary of Processor Enhancements

All of the logic modifications described in this section are scheduled for implementation during the next year. For the most part, they consist of adapting the present implementation to fulfill the original design specifications, or redefining these specifications in line with the present hardware's potential. As such they are design intensive and require a minimum expenditure in hardware. Thus it is difficult to predict to what extent each modification will be successful in increasing the maximum speed of the machine, much less how well they will together improve the realized execution speed of user programs (the latter is to some extent dependent upon optimizing the order in which instructions are issued, for example to exploit pre-fetching of operands into scratch registers in the PE). However, an increase in observed performance of a factor of two seems reasonable, and this factor is assumed for the purposes of this report.

III.A.5. Conclusion

As the speed of the Illiac continues to increase, the already noticeable weaknesses of the system will become more pronounced. These weaknesses are:

1. the limited bandwidth between the system and the outside world and the resulting time required to move data onto and off of the system;

2. the lack of a large on-line storage facility to keep active data bases between runs over a period of a few weeks;

3. the low bandwidth between the file system and the array, and the resulting time to load and unload the array memory;

4. the bandwidth between local and extended memory and the latency in accessing the extended memory, which together lead to inefficient use of the machine or to extended program development time to reduce these inefficiencies; and

5. the declining amount of extended memory.

Each of these weaknesses affect different users to different extents, but it is fair to say that improvements in any area will generally help most users. Of the weaknesses, the last two are generally regarded by the user population as the most serious and have the highest priority to be corrected. The remaining sections of this chapter treat these issues in more detail, attempting to quantify the current situation and, through analysis of usage patterns, to evaluate the direction and extent of memory enhancements needed to meet the developing needs of the scientific community which will use the Illiac in the future.

III.B. Application Program Examples

In order to evaluate the various alternatives for enhancing both the local memory (PEM) and the extended memory (I4DM), several application codes were examined with respect to memory usage patterns. In addition, the memory usage characteristics of several proposed codes were estimated on the basis of experience with similar, existing codes. This section describes these codes. The next section develops a model of a hierarchic memory system, and these codes will be used to supply values to some of the parameters of that model.

III.B.1. Twelve Representative Illiac Codes.

Twelve codes that were written for the Illiac, or are currently being designed, are described in this sub-section. They are representative of the application areas currently being supported by the Illiac system; these major application areas are computational aerodynamics, seismic research, climate modelling and satellite image processing. Since these areas are expected to include the major users of the system over the next five years, an understanding of the way these codes use the current memory will be useful in developing guidelines for memory enhancements.

Although these programs could be used to construct a job mix to evaluate the various memory systems proposed in the next chapter, that is not their primary intention here, since such an exercise would require a policy decision outside the scope of this report: it would require a weighting of the relative importance between memory capacity and accessing flexibility.

AIRCRAFT/SAR

The Illiac AIRCRAFT/SAR code is an experimental code for processing raw, uncorrelated digital air-borne Synthetic Aperature Radar (SAR) video data. The input to the program is a 512x6144 array of 8-bit binary integers. Each 8-bit word of input is converted to a 64-bit Illiac floating-point complex number (32-bit real and imaginary parts). The data is then processed using Fast Fourier Transform algorithms to produce a correlated radar power image array corresponding to an area of approximately 1.8 by 2.5 kilometers. The output is again a 512x6144 array of 8-bit binary integers.

GISS

A weather model formulated by Goddard Institute for Space Studies (GISS) simulates the global circulation of the atmosphere over a two week period. The grid is "split" with latitudes nearer the poles having fewer grid points. The model consists of 46 latitudes, each latitude composed of nine atmospheric layers; the equatorial latitudes have sixty-four grid point resolution. Each grid point has four variables associated with it; surface grid points have an additional six variables. In addition, about 250 rows are set aside for temporary variables generated in processing a latitude.

A natural formulation of the problem processes the data base by latitudes, sweeping through the grid from south pole to north pole. The basic integration algorithm requires points at neighboring latitudes for updating a given latitude, but since a predictor-corrector scheme is used, four latitudes must be present for processing. Each sweep through the grid requires approximately 5.36 seconds; some sweeps require more time than others since different routines are used in processing some of the time steps. Each run will generally execute 2016 time steps. The input to the program is 123,648 words; the amount of output generated by the program varies under user control, but a typical run might produce two million words of output.

A refined model would have 72 latitudes, with maximum resolution of 128 grid points at the equator. It would use 32-bit arithmetic, so each latitude would remain one Illiac row in length. One would expect a computing degradation of about 1.5 in going to 32-bit arithmetic -- the operations take longer and routing is slightly more complex. The number of time steps would remain the same. Thus for a larger version of the GISS model, the total computing time would increase by a factor of about 2.3 on a data base 50% larger.

I4TRES and ACTION

I4TRES is an Illiac adaptation of a code developed by Systems, Science and Software to perform a seismic simulation using finite-difference methods. The finite difference grid is 80x80x160 nodes (typically representing 0.1 kilometers in the physical situation). The simulation is usually carried out for 250 time steps and involves six variables at each node for each time step. Thus the total number of variables in a simulation is about ten million. In addition, fewer than 100 rows of Illiac memory are reserved for the temporaries generated in processing each plane.

No significant input is required unless the problem is being restarted in which case a restart dump of about ten million words must be brought in. Output typically consists of about forty variables at each of 2000 nodes at every time step. Since these are packed as 36-bit words, this results in about ten million words.

In the current implementation a run takes about six hours (90 seconds per time step). About half of this time is in I4DM-PEM moves (about 1300 moves per time step of about ten thousand words).

For this problem there is a fairly natural hierarchy of data sets. The algorithm requires that neighboring points be used to update a node; thus to process one line requires three lines in each of three planes. The natural data set the inner most level is 80x3x3x6 (one line of the grid). The second level is 80x80x3x6 (three planes). The third level is 80x80x160x6 (the grid) and the total simulation is 40x2000x250 (the total output). Since I4DM cannot at present contain the full simulation, the time steps are grouped with only twenty to fifty being executed in a run. Similarly, since the second level set cannot be contained in PEM, it is overlayed in PEM with only a quarter being treated at any one time in PEM.

The ACTION code is a second generation version of I4TRES which is being developed at the Institute. Whereas I4TRES is based on a finite difference formulation, ACTION will use a finite element mathematical model. The data set for ACTION is expected to be about three times larger than I4TRES and the execution time per step is expected to increase proportionately.

LANDSAT

The current version of the LANDSAT code involves two processes: classification and color assignment. The basic process involves a frame consisting of about 7.5 million pixels (picture—elements) in which each pixel represents about 1.4 acres in the total ten million acre picture. The pixel information is the digitized color intensity at four wavelengths. In the classification process, each pixel is assigned a cluster number. The color process assigns the color of each pixel in the final image to be produced; this assignment is made on the basis of the pixel's cluster number.

The input to the classification process is about 7.5 million 32-bit pixels (eight bits for intensity for four wavelengths). The output is 7.5 million eight bit cluster identifiers and 7.5 million eight bit statistical threshholds. The classification output is input to the color process. The color output is three 7.5 million eight bit intensity files.

The classification takes about three minutes of computational time and with I4TAPE will take about six minutes of transfer time into the system. Currently the transfer time is about nine minutes. The color process involves only trivial amounts of processing and is thus entirely dominated by transfer time.

SEASAT/SAR

A proposed code to produce an image from Synthetic Aperture Radar data is SEASAT/SAR. A frame represents 100 kilometers by 119 kilometers of surface area and is collected by the satellite in fourteen seconds. The desired resolution is 25 meters. However, the input is oversampled and actually consists of about 13K by 30K pixels of eight bit intensities. The input is about 400 million eight bit pixels per frame, or 50 million words. The output is 4000x4000 pixels or about two million words. The intermediate precision required in processing is 64-bits.

The processing involved is extensive. For each 13K length column, the program executes a Fast Fourier Transform (FFT), filters the result, and performs an inverse FFT. Due to range migration, the image must be straightened out. Finally, for each 30K length row, the program does an FFT, filters the result, and does an inverse FFT. The row is then truncated and resampled to obtain a 4K length.

The basic computational time is estimated at one hour assuming the data could be core contained. With the current architecture it may take over six hours. Since the problem will not fit in primary or secondary storage, I/O time may also be significant.

SHUTTLE CODE

The SHUTTLE code was developed at NASA-Ames to simulate the non-equilibrium flow field of the space shuttle on re-entry. The model uses a 10x17x12 grid, with three problems being solved at one time. There are 24 variables per grid point. The data base is divided into 48 sections and is buffered through memory once each time step. There is one second per time step and 600 to 800 iterations per case.

2D-STRATOSPHERIC

The 2D-STRATOSPHERIC code, developed at NASA-Ames, models the effects of certain chemical species in the stratosphere after injection of foreign chemical compounds. The current version has an input of 78144 words, which is also the size of its internal data base during execution. A new version increases the size to 86592 words. The output is variable. Typically it is about fifty times the size of the input. One run models one year of real time and takes approximately sixteen minutes. The time step is one day and takes approximately three seconds in the current version and an estimated seven seconds for the new model. For the present system configuration, each run requires approximately 700 seconds to load and unload the data base.

2D-TRANSONIC

The 2D-TRANSONIC code was developed at NASA-Ames to solve the time-averaged Navier-Stokes equations with a turbulence model; it simulates the transonic flow over a two dimensional airfoil at various angles of attack. The model consists of a 60x128 grid, with each grid point having some fifteen variables. Sixteen temporaries are generated for PEM contained grid points.

As currently implemented, the grid is in two 60x64 parts with only one part in PEM at a time. It takes one second per iteration and four passes over the data per iteration. On the average there are two thousand iterations per case. If the data base must be dumped for restarting another run, then one variable per grid point node is required.

3D-COMPRESSIBLE TURBULENCE

The 3D-COMPRESSIBLE TURBULENCE code was developed at NASA-Ames to compute the velocity field for turbulent compressible flows. The algorithm is based on a finite difference method, and the model also includes density and temperature. The code has a data base of 20x64x64x64 words. Actual run time is 20 seconds per time step, of which approximately half is I/O time. Each run is approximately 200 steps for a total run time of 4000 seconds. The output data base is 256 words per time step. Two points are of interest with respect to this code's use of the current memory system. The first is that because of the small PEM, the main data base must be segmented, and the segmentation chosen requires that some data from I4DM be read more than once in order to process the data base. As a result, the total amount of data which passes between PEM and I4DM is 14 million words to process a data-base that is 5.24 million words. The second point is that because of the latency of the disk an effort was made to reduce access time in reading data. This leads to reading non-contiguous pages from the disk and also results in a complicated arrangement of data on the disk. If the data base must be dumped for restarting another run, then five variables per grid point node are required.

3D-GALACTIC

The 3D-GALACTIC code was developed at the University of Chicago and NASA-Ames. It simulates the formation and evolution of galactic structure due to gravitation. The code simulates a galaxy, currently using about 116,000 point masses to represent stars. The three dimensional n-body problem is solved numerically for sixteen time steps per run, and the results are accumulated for up to 128 consecutive time steps. Each run takes approximately 2400 seconds, and an additional six minutes are required to move the problem on and off the Illiac. The basic input consists of 266K words and the output of each run is approximately 299K words.

3D-INCOMPRESSIBLE TURBULENCE

The 3D-INCOMPRESSIBLE TURBULENCE program was developed at NASA-Ames to compute the velocity field for turbulent incompressible flow. The main algorithm is a Fast Fourier Transform. The current code has a data base of ten 32-bit variables per grid point on a 64x64x64 grid. Data is accessed in two ways. The data base logically consists of a matrix having 40 rows 32 Illiac pages long and 32 columns 40 pages long. Each step requires that portions of the data base be read and written four times row-wise and four times column-wise. The data is stored physically by column, that is, all pages in a given column are stored contiguously on a single band. Data for consecutive pages ácross a row are stored physically on contiguous bands separated by three page gaps. The entire data base is accessed by column, but only the first 16 rows are accessed row-wise. Slightly under 100 thousand words are set aside in local memory for buffer areas.

Each time step takes about eight seconds of computation to process, and slightly under two seconds are required for data transfer. Currently, an additional ten seconds are spent in transferring data due to the effects of disk latency or data access re-tries. A significant amount of this access time can be eliminated when enough extended memory is available for the primary data base, however, since a more sophisticated lay-out, requiring about twice as much physical room for optimal placing of the data base, is possible.

A typical run is about 50 time steps, for a total run time of 1000 seconds. The output data base is around ten thousand words. If the data base must be dumped for restarting another run, then three words per node are required.

A future version of this code would have double the resolution in each dimension, thus increasing the demands on memory size by a factor of eight. The amount of computing required for such a problem would be slightly more than eight times as much since the basic algorithm of this program, the Fast Fourier Transform, requires time proportional to NxLog(N) to process a data set of size N.

3D-TRANSONIC

Similar to the two dimensional transonic code above, the 3D-TRANSONIC code is being developed at NASA-Ames to solve the time-averaged Navier-Stokes equations with a turbulence model in order to simulate the transonic flow over a three dimensional body of revolution at various angles of attack. The model is an 80x40x40 grid with 23 variables per grid point. There are 25 temporaries generated for PEM-contained grid points. Currently the program works on a sub-grid of size 8x8x40 or 8x8x80 within PEM. The grid is moved into PEM buffers each of whose maximum size is 8x8x80. The code is expected to take thirty seconds per iteration. It will require six passes over the entire grid per iteration. There are an average of between 1500 and 3000 iterations per case, thus requiring up to twenty-five hours of Illiac computing time. One of the goals of this research project is to reduce the number of iterations required for problem solution, so the above estimate is probable conservative. If the data base must be dumped for restarting another run, then five words per node are required.

These programs indicate how the current configuration of the Illiac is being used. They are useful in indicating the directions memory enhancements to the Illiac system should tend. And they can be used as a sample job mix for evaluating the effectiveness of different memory configurations.

The main characteristics of these codes are collected in two tables. The first, Table III.i, consists of those codes currently running on the Illiac. The programs marked with an asterisk are those for which the main algorithm is an in-place algorithm, that is, the program can update the data base by over-writing the data as it resides in local memory. The Fast Fourier Transform is an example of this type of algorithm. This type of classification is important to the study since quantitively different results are derived for in-place algorithms than for other types of algorithms.

The other table, Table III.ii, gives the expected performance for codes being developed, or proposed, for the Illiac. It also includes the extrapolated performance for future versions of some of the codes in Table III.i. It should be noted, however, that this assumes the current processor speed and disk latency, as well as present system throughput characteristics. From this last table, a number of conclusions about the future requirements on the Illiac system can be inferred. These are listed in the concluding part of this section.

III.B.2. Conclusions

The most noticeable aspect of Table III.i is that the latency time runs from 50 to 125 percent of the compute time. For most of the codes, the transfer time is well under ten percent of the compute time. Although it would be desirable for codes to take only two or three percent of their compute time to be loaded and unloaded from the array, most of the current programs far exceed this goal. If the observable processing speed of the Illiac doubles, then these percentages also double. This means that latency will increase as a concern for the users, as will the transfer rate between the support system and the processor.

Table III.i.

Characteristics of some Major Illiac Programs

Applications	Compute Time	I4DM Latency Time	PEM-I4DM Transfer Time	TOTAL Run Time	System-I4DM Transfer Time
AIRCRAFT/SAR	28	22 (79)	2 (7)	52 (186)	768 (1396)
GISS	10800	930 (8)	60 (0)	10900 (101)	150 (1)
I4TRES	10800	10800 (100)	500 (5)	22100 (204)	900 (8)
LANDSAT	210	3 (1)	3 (1)	216 (103)	540 (257)
SHUTTLE	790	300 (37)	10 (1)	800 (101)	60 (5)
2D-STRATO	950	480 (51)	15 (1)	965 (101)	700 (74)
2D-TRANSONIC*	1110	640 (57)	250 (22)	2000 (180)	40 (6)
3D-COMPRESS	3630	1630 (45)	370 (10)	4000 (110)	25 (0)
3D-GALAXY	2000	740 (37)	45 (2)	2045 (102)	360 (18)
3D-INCOMPRESS*	400	500 (125)	100 (25)	10000 (250)	20 (2)

Notes: Time in seconds; parenthetical numbers are percentage with respect to compute time. Asterisk indicates that the program is based on in-place algorithms (see text for definition of in-place algorithms).

Table III.ii.

Expected Characteristics of Some Future Illiac Progams

Applications	Compute Time	I4DM Latency Time	PEM-I4DM Transfer Time	Total Time
ACTION	32000	32000	1500	67500
GISS	27000	1400	60	27100
SEASAT/SAR*	10700	340	190	12230
2D-STRATO	2560	480	10	3050
3D-TRANSONIC*	62500	37500	7500	108000

Notes: Time in seconds.
Asterisk denotes in-place algorithm.

III.C. A Memory Model and its Implications

The model discussed in this section is based on the current Illiac memory organization. It is a hierarchical memory system with three levels. The first level is the local memory (PEM) and holds the operands which Illiac instructions directly reference. The second level is the extended memory (I4DM) and holds the data base of the program during its execution. The third level is the external memory (presently Tenex file storage and magnetic tapes) and holds the program's data when the program is not executing. The following parameters of this system are of particular interest, and will be considered in this report: the size of the first two levels of memory, the bandwidths between levels, and the average latency involved in moving data sets between levels.

III.C.1. Parameters of the Model

Programs which use the Illiac usually have several input and output files which reside on the external memory. The total size of these files is referred to as the throughput of the program. A program has a maximum amount of memory required on the second level (the extended memory), and the program accesses this storage at a given rate, the DATA RATE, which is measured in bits per second of compute time. Formally, the data rate is defined to be the number of bits to be processed divided by the time required for processing. The data rate can be thought of as the rate at which a program processes its data. This quantity is particularly important for this study and a brief explanation of its significance is appropriate here. Consider the following example. A one million bit file resides in extended memory. It is brought into local memory and half a second is spent computing on that file; therefore, the data rate in this example is two megabits per second. This is independent of the latency of waiting for the transfers (waiting for the current disk to begin the transfer) and of the actual transfer time. Thus, even if these two parameters change, the data rate remains constant and can be used to estimate system throughput for various extended memory characteristics. It might also be noted that computing time is simply the size of the data set divided by the data rate.

For a given program, the data rate may differ for various portions of the program, or during a simulation it may change from one time step to another. Also, different formulations of the same program may have different data rates, especially if the local memory is increased substantially. Representative values for the programs in the previous section are given in Table III.iii. As the reader can see from the table, the data rate for current Illiac programs generally ranges from 2 megabits per second to over 100 megabits per second with 30 megabits per second being an average. There are enough codes that are much higher, so that 60 megabits per second is more representative of these programs as a whole. In view of the expected speed improvements to the processor, a generic data rate of 100 megabits per second will be used in this report in an effort to balance the memory system.

Programs use the local memory for two purposes: to hold the program and to hold the program's working data base. The amount of memory not devoted to the program can be partitioned into buffers with the program working on data in the buffers. There are two general buffering schemes which are of importance in this context. The most primitive technique, called simple buffering, involves bringing data into local memory, operating on it, returning it to extended memory, and then fetching the next batch of data from memory. This technique essentially does not allow computing to be performed while waiting for the read and write requests to be satisfied since between the writing of the buffer and the reading of the next data set, there is no information in the local memory on which to compute. Single buffering is used when the amount of computing is very small (the data rate is quite high relative to the size of PEM) and the bulk of the compute time is inevitably lost to latency. In such a case it is beneficial to reduce the number of read/write requests (thereby reducing total latency), and this means operating on as much of the data base at one time as possible, i.e. having as large a buffer area in the local memory as possible.

The other buffering scheme involves using several buffers and is often called multiple buffering. In this scheme one buffer is used to write out the results of the computation while the program computes on information in other buffers. When the output buffer is empty, the next data to be processed are read into that buffer. If the amount of computation on the data buffers is such that the write and read of the I/O buffer are completed before the computing is finished, then the latency effects of the read/write process have no effect on the run-time of the program.

There is a point at which it is computationally more efficient to use a simple buffer scheme than a multiple buffer scheme. The following argument demonstrates how to determine this point for a given problem and memory system. Some general remarks will follow the example. Simple buffering allows for larger buffers, and thus requires fewer reads to access the entire data base. Therefore simple buffering requires less latency, whereas multiple buffering, with greater latency time, requires more computing to hide the latency. The size of a buffer in simple buffering varies, but half the available memory area for a buffer is about average (in-place algorithms, such as FFTs, can use the entire available memory for the buffer, however). On the other hand, a multiple buffering scheme could require five buffers (three to hold the old data base -- three planes of the grid, for example -- plus one for the newly computed plane of data, and one to be used for input and output), thus only a fifth of the available memory can be used for each buffer.

Table III.iii.

I/O Characteristics of Some Major Illiac Programs

Applications	Data Rate	Input	Output	Throughput
GISS	1.7	0.1	2	3.2
I4TRES	28	10	10	120
LANDSAT	7.7	4	3	2100
SEASAT/SAR	37	50	8	350
2D-STRATO	6	0.1	4.3	290
2D-TRANSONIC	110	0.05	0.13	13
3D-COMPRESS	90	0.002	0.0	0.6
3D-GALAXY	11	0.3	0.3	19
3D-INCOMPRESS	130	0.01	0.01	3.2
3D-TRANSONIC	57	0.08	0.2	0.3

Notes: Data Rate in megabits/sec, input and output in megawords, throughput in kilobits/sec.

If M is the amount of the local memory that can be devoted to buffers and S is the size of the problem's data base to be processed, then simple buffering requires 2xS/M reads whereas multiple buffering requires 5xS/M reads (assuming one read is required to fill a buffer). Assuming the same number of writes are required as reads, this means that the difference in latency time between the two methods is 6xS/M times the average latency. This is also the minimum amount of computing time which must be overlapped with latency before multiple buffering is more efficient. Thus the total time to process a data base once during the problem must be at least this great to warrant considering a multiple buffering scheme. For the current memory system, 6/M times an average latency of 20 milliseconds comes to 19 nanoseconds per bit, assuming three quarters of the current memory can be used for buffering (the rest of memory being used for program and temporary variables). This corresponds to a data rate of 52 megabits per second. Programs with a data rate lower than this could probably profit from using a multiple buffering scheme. About three fourths of the surveyed programs have lower data rates now; with an increase in processor speed of a factor of two, about half of them would.

For in-place algorithms, simple buffering requires one buffer and multiple buffering two, thus the difference is S/M. Again assuming one read and one write per buffer, the break-even point is 2xM times the average latency. For the present system, this is about 6 nanoseconds per bit or a data rate of over 160 megabits per second (again assuming three quarters of the memory is available for buffering). None of the programs currently exhibit this data rate.

A few comments are in order before leaving this issue. Generally, multiple buffering schemes are more complex to program, which adds to the cost of developing application codes and subsequently modifying or otherwise maintaining them. It can also be the case that using smaller buffers can impact the algorithm by reducing the amount that can be done with each buffer of data. For example, if each segment of a data base must be processed twice, as in solving a tri-diagonal system, then when the entire segment can fit in one buffer, only half as many reads will be required.

As the speed of the processor increases, the data rates of the programs increase, assuming no changes are made to them. In addition, increasing the memory size or decreasing the latency causes the break-even point between the two buffering schemes to increase with respect to data rate. Thus, one would expect that in the future fewer codes will benefit from the simplicity of simple buffering schemes. In addition, if programs are developed which require several reads and writes to construct a buffer (pulling information from different files on the extended memory) or if more complex models are used (thus decreasing the data rate), simple buffering becomes less efficient. The remainder of this section is in terms of multiple buffering schemes, the modifications required due to simple buffering will be noted at the end of each sub-section, as appropriate.

III.C.2. Memory Size and Data Access Rate

The inefficiency due to latency can be lessened or eliminated by multiple buffering if computing can continue while waiting for the transfer of data to begin. For this to be effective, the amount of computing must approach or exceed the expected delay due to latency. The expected latency time may be estimated as half of the maximum latency time (on the average), or it may be less if the data set is carefully arranged on the extended memory (assuming that the extended memory is a physically rotating device, such as the current disk memory, or is an electronically rotating device, such as charged-coupled devices or magnetic bubbles). In the current memory system, optimizing the layout of data on disk in order to minimize latency is generally complicated since what may be a good map for one phase of a program may be a very poor map for another phase. In many cases, the total map of the disk must be adjusted in order to achieve a reduction in latency for all phases taken together. Generally, the difficulty of mapping the disk decreases with the increase of the ratio of total space available to the total space required, that is, more space on the disk gives added flexibility for data layout.

The amount of computing to be done on each buffer can be increased by increasing the buffer size. For a fixed data rate, processing time increases linearly with the size of the buffer to be processed. Thus the effects of latency in accessing extended memory can be reduced whenever the total latency associated with the read/write requests required to construct a buffer of information is less than the the time it takes to process the buffer (the buffer size divided by the data rate).

The minimum number of read/write requests to process a buffer is two, one to read the buffer in and one to write it out again. However, there may be several reads and/or several writes involved if several different files are used to construct a buffer. For the purposes of this study, two read/write requests per buffer are assumed throughout. Thus it is of interest to consider when twice the average latency is less than the computing time. The computing time is the buffer size divided by the data rate. Figure III.4 presents a graph relating latency and buffer size for various data rates. For a fixed data rate, the graph indicates for what values of memory and latency further improvements would have no effect on run time. If the available memory and average latency lie above the line, then run time cannot be improved. If the available memory and average latency lie below the line, then increasing memory or decreasing latency will improve run time.

There are three regions on the latency axis that are of practical interest; these correspond to the characteristic latency times for disk technology, charged-coupled device technology, and semiconductor random access memory technology. The Illiac programs in this study require from one to two megabits for program memory, thus leaving around five megabits for buffer area. The lines drawn on the graph include the effects of two read/write requests multiplying the latency factor and the number of buffer areas for different buffering schemes. Thus the graph can be examined without these adjustments being performed.

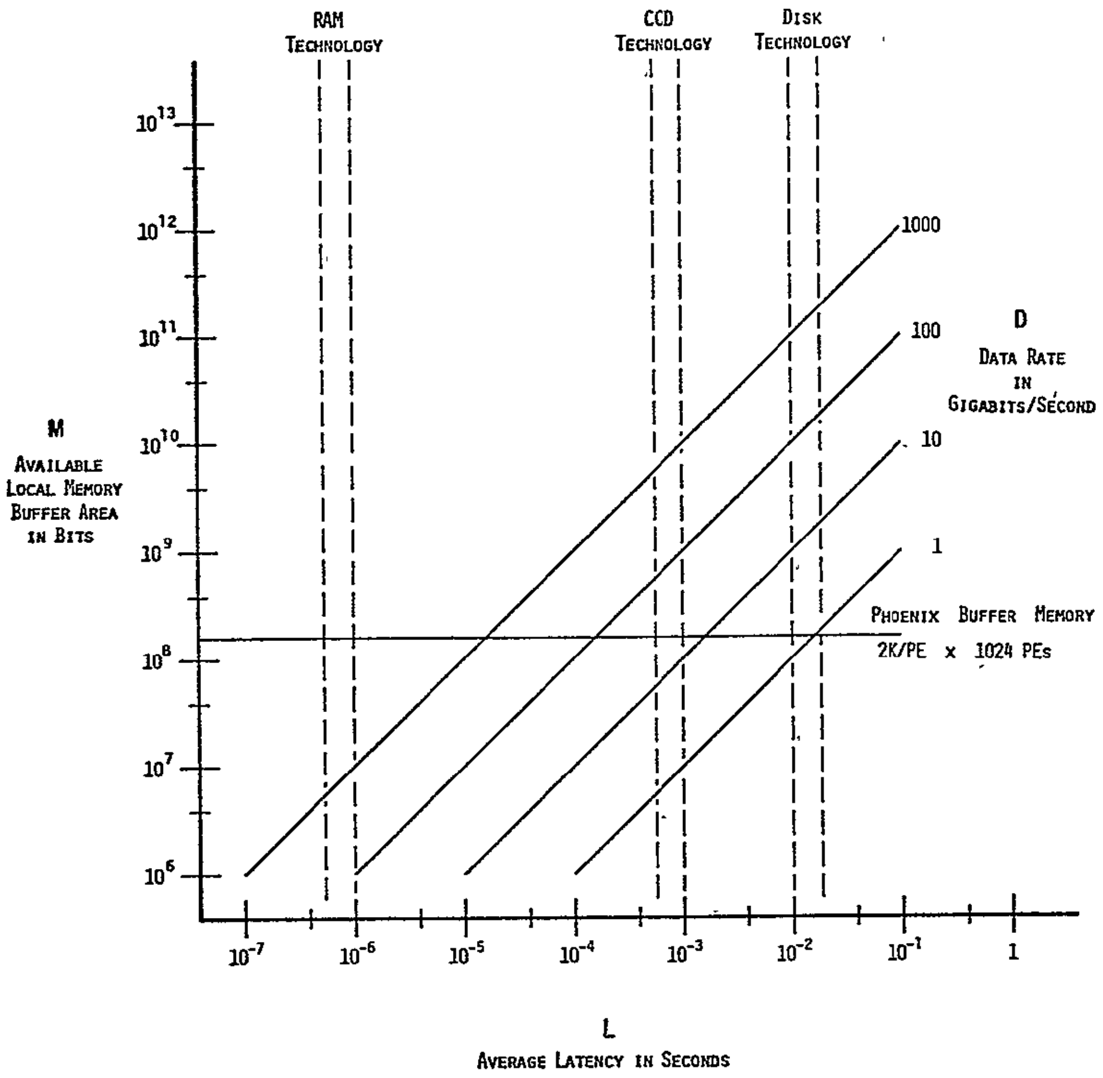


Figure III.4.

Graphs of $2L = M/DS$

For multiple buffering, the major conclusion to be drawn from examining the graph is that the current size of PEM creates major problems for Illiac programs with data rates higher than ten megabits; this is with respect to disk latency only. To continue using a disk technology for extended storage, and taking into consideration that the potential speed-up of the Illiac will approximately double the data rate of existing programs, an increase in the size of local memory by at least a factor of ten seems to be indicated. On the other hand, changing the underlying technology of extended memory reduces the latency by more than a factor of ten and hence latency becomes less of a factor, if at all, in determining a suitable size for the local memory. The above discussion was for multiple buffering, since multiple buffering is the only buffering scheme which decreases the contribution of latency time to the total run time. For simple buffering, one generally wants latency to be a small fraction of the compute time; this report uses 10 percent of compute time as an acceptable amount of latency time in simple buffering.

In summary of this discussion, examination of the data rate of current Illiac codes, together with the anticipated increase in the speed of the processor, suggest the ratio of local memory size to extended memory latency. This ratio is in some sense a figure of merit for a two level memory system, since it reflects the ability to have buffer areas large enough to mitigate the effects of latency: the ratio increases with an increase in local memory (indicating greater ability to mask the effects of latency) or with a decrease in latency (indicating less of a need to mask the effects of latency). The ratio of memory to latency, M/L, should be greater than some multiple of the data rate. If one excludes in-place algorithms, this multiple is either 10 or 40, depending upon whether multiple or simple buffering is used, respectively. The observed data rate of Illiac codes range from 1.7 to 90 with 30 being average; this implies a ratio of memory to latency of upwards of 300 megabits of local memory per second of latency. For in-place algorithms, the multiples are 4 and 20, again for multiple or simple buffering. The observed data rates for in-place algorithms range from 36 to 134 with an average of 80; these indicate a ratio of upwards of 800 megabits per second. For the generic 100 megabit data rate, the ratio should be upwards of a gigabit per second.

III.C.3. Bandwidth between Local and Extended Memory

Further implications of the observed and anticipated data rate of Illiac programs can be drawn. Perhaps the simplest is the requisite bandwidth between the local memory and the extended memory. To a large extent, the bandwidth available from the extended memory is a matter of architectual design -- more parallel read heads on a disk or more memory banks on an electronic memory, for example. The Illiac I/O port will, as seen in sub-section III.A.1, accept up to sixteen gigabits per second, although for most application problems the current 500 megabits per second seems to be adequate. The reason for the satisfaction with the transfer rate can be traced to the data rates most frequently observed. Ignoring latency, programs consist entirely of computing and transferring information. If the data rate is 20 megabits per second, only 1/25 the transfer rate, then the time spent transferring information accounts for less than four per cent of the program's total run time. Even for programs with data rates of 100 megabits per second, the transfer rate would account for less than 17 per cent of the run time. While 17 per cent is beginning to become significant, if the transfer rate were 50 megabits per second (which for example is the sustained transfer rate for the STAR-100 extended memory disk system at NASA Langley Research Center), the transfer time would account for two-thirds of the total execution time.

The above discussion, of course, assumes that transfer time cannot be overlapped with computing. If it is possible to overlap both latency and transfer with computing, then the following inequality is obtained. NLet B represent the hardware bandwidth in megabits, D the program's data rate in megabits per second, L the latency in seconds for accessing extended memory, and W the size of a buffer in local memory measured in megabits. Then the amount of computing performed is W/D.

It is necessary that the buffer size W be large enough so that W/D is greater than the sum of both the total latency time and the transfer time. The latency time is 2L (for both a read and a write). The transfer time is 2xW/B. Thus the total data movement time is 2(L+W/B). Assuming that the bandwidth B is greater than twice the data rate D, and solving for the buffer size W, one sees that:

For the current memory system on the Illiac, B is 500 megabits and 2L is about 40 milliseconds, thus the buffer size W should be greater than 1000xDx4x10E-2/(500-2D). For multiple buffering on the current Illiac memory (so W is 7.3/5 or about 1.46) this expression indicates that the memory is suited for problems for which D satisfies 1.46>40xD/(500-2D): that is for programs with a data rate less than 17 megabits. If one desires that the Illiac be well matched for a data rate equal to 100 megabits per second, then using disk technology, the above analysis and assumptions on B and L indicate that the memory buffer be greater than 13.3 megabits, or that the local memory be 67 megabits (one million words) or larger.

Rewriting the above derived inequality, the bandwidth B should be greater than 2xWxD/(W-2xDxL). Using a buffer size of 100K words, a generic data rate D of 100 megabits per second, and a latency of 1 millisecond (easily realized using charged coupled devices), the expression indicates that a bandwidth of 200 megabits per second is desirable. Actually, one would probably want the transfer time to be significantly greater, since there is no clever way of arranging data so as to minimize the transfer time. By having an overly comfortable bandwidth, the transfer of data between extended memory and local memory ceases to be an issue; only the arrangements of data in extended memory remains, and that has a chance of being successfully dealt with given enough time and programmer ingenuity.

III.C.4. Extended Memory Size and System Bandwidth

As was seen earlier, the minimum size of the local memory is dictated largely by the latency of the extended memory. The key to the relationship between size and latency is the data rate of the programs the memory system is expected to support. In determining the minimum extended memory size, different considerations usually arise. This is because extended memory is designed to be the main store for an executing problem's data, rather than a buffering area between the local memory and the support system memory. Thus it is loaded only once at the beginning of program execution, and any latency involved in beginning the data movement is suffered only once. But since very large files are involved, the effective bandwidth between the support system and extended memory becomes a dominant consideration. Of course the effective bandwidth will be less than the maximum bandwidth due to latency of memory and staging devices and due to software overhead. For the purposes of discussing the capacity of extended memory, however, the effective bandwidth is the main determinant, and again the key will be the rate at which executing programs deal with their data. Here the issue is not the program's data rate, but its THROUGHPUT RATE, which is defined to be the sum of its input and output data divided by its compute time.

What is at issue in determining the size of the extended memory, aside from the obvious requirement that it hold the entire data base for reasonable problems, is how many different problem sets should reside simultaneously in the extended memory. That is, whether the extended memory is used in a simple buffered fashion or in a multiply buffered fashion: whether a program is loaded into the extended memory, executed, unloaded and another program loaded, or whether a program is loaded while another program is being executed, the intent being to overlap computing with program set-up so as to mitigate the delay caused by an insufficient system bandwidth. Thus, for example, a program requiring a throughput higher than the system can deliver could be staged onto extended memory while a low throughput program is computing.

Clearly, if the time to load and unload a program on extended memory is only a few per cent of the total run time, then simple buffering (or dedication of extended memory to one program at a time) is acceptable, and the simplicity of simple buffering makes it irresistibly attractive. Multiple buffering entails at a minimum protecting one user's program and data from other users, and this capability adds to the cost and complexity of the extended memory.

Estimating the throughput characteristics of future Illiac programs is relatively straight-forward, based on the throughput characteristics of the example programs. One expects the computing time to be reduced, by anywhere from ten to fifty or sixty per cent. The time lost to extended memory latency can similarly be estimated, dropping by a factor of 60 by going from disk based technology to Charged-Coupled Devices. One complication does arise, and that is that some computational processes do not require time linearly proportional to the size of their input; for example, the Fast Fourier transform takes time proportional to NxLog(N) where N is the size of the input. The result is that as problem sizes increase, their throughputs decrease and the demands on the system decrease as well. For the purposes of this study, however, linearity (or constant throughput requirements) will be assumed for simplicity.

As can be seen from Table III.iii, current programs on the Illiac exhibit a throughput rate of from under 100 to over 1800 kilobits per second with three hundred kilobits per second being a representative number. Thus to keep system loading and unloading under ten percent of the program execution time, a six megabit per second rate is indicated. To illustrate the potential impact of an enhanced Illiac on the support system, consider increasing the processing rate of the Illiac by a factor of two and reducing the latency of the extended memory by a factor of one hundred. For one example problem, the Compressible Turbulence program, the throughput rate would go from its current 340 kilobits per second to over one megabit per second. Thus the impact could be substantial.

III.C.5. Summary

In summary, the main implications of this memory model are as follows. First, the latency of the extended memory has a significant impact on the size of the local memory when its latency time is comparable to disk technology. The characteristics of the current programs suggest that to mitigate the effects of latency, the ratio of the local memory to the latency of the extended memory should be greater than 800 thousand bits of storage for each milisecond of latency. Second, a transfer rate between local and extended memory in excess of 200 megabits per second seems warranted by the current applications. And third, to dedicate the extended memory to one user at a time and to expect no more than ten percent execution time degradation requires the system to support a transfer rate of at least six megabits per second (currently) and possibly up to twenty megabits per second in an enhanced system.

III.D. Miscellaneous Topics

This section treats four topics which affect the memory system enhancements. The first involves the size of the local memory and the extended memory. The second deals with reliability of the local memory and the third topic involves a virtual memory organization for the array's memory system. The final topic discusses the need for and size of a large on-line file storage facility in the Illiac system.

III.D.1. Size of Local and Extended Memory

Recognizing that the more complex a program is, the harder it is to debug and to modify, it is important to consider what size of local memory would be necessary in order for all of the application codes surveyed for this report to be programmed in a fairly straight-forward way. Many of these applications involve three dimensional grids. For such applications, it is natural to have a local memory large enough so that entire two-dimensional planes could be computed at one time, with all needed data residing in local memory. For most of the applications using three dimensional grids whose algorithms are based on finite difference approximations to the partial differential equations, it usually takes three planes to produce an updated plane. Thus it is natural to assume that the local memory is adequate if six planes of data can be held in local memory at one time. This allows three planes of data for computing a new plane of data and two planes of data for I/O buffering. Using this criterion, a local memory of a quarter million words seems adequate for all of the above applications except 3D-Compressible Turbulence. A local memory of half a million words would be of ample size for all of the above applications.

Only if significantly larger models are going to be employed on the Illiac would the local memory need be larger than half a million words, based on a "natural" accommodation of the problem formulation, and this would in turn necessitate a significantly larger extended memory accessible to the user.

When one considers how large the extended memory should be, a simple rule of thumb would be that with a machine twice as fast, problems twice as large can be solved, so an increase in extended memory size of at least a factor of two should be considered. The "large" problems now being run (or proposed) on the Illiac have outputs of five to ten million words, and data bases of three to ten million words. If one assumes that output can be written to the support system while the program is running, then the underlying basic data base is the determining factor of the size of extended memory. Considering the current applications surveyed in this report, it seems that a ten million word extended memory should be available for user programs. On the other hand, three dimensional codes usually require doubling the resolution in each dimension in order for the problem to be interesting. This indicates that an extended memory eight times the current I4DM would be of interest to some users: this calls for a memory capacity of about sixty million words.

In an effort to mitigate the time required to load and unload user problems, the extended memory should have a high bandwidth out of the memory. Whether this is to the support system or to a staging device is largely a function of the capability of the central system to accept large data bases at a high transfer rate. If this is not the case, then a buffering mechanism is needed. Without a separate buffering mechanism, then loading and unloading problems should be overlapped with running other user jobs. This means allowing multiple users to share the extended memory, which would require a larger extended memory and the capability to protect an executing program from inadvertently accessing other users' files on the extended memory. If the extended memory is to be shared by different users, then a minimum size would be fifteen million words, thus allowing a large program to alternate with a smaller one. In this case, the bandwidth to the support system could be less since the time to load and unload a problem would be hidden from the user.

III.D.2. Error Correction on Memory

With the present electronic technology, bipolar static random access memory chips are the basic building blocks for high-speed memory systems, such as the local memory for the Illiac. If a memory system is small, and its application is intended for conventional systems, the problem of reliability is not a major issue. On the other hand, if the memory system is large and its application is intended for a large computing system, the problem of reliability is a critical issue and it should receive serious consideration. The following discussion addresses the need for error correction and a system implementation of error correction for the Illiac.

The need for error correction for a memory system depends on several factors. The major factors are the type of memory chips used, the number of chips needed, and the memory system fabrication. In this discussion, the issue of system fabrication is not considered and it is assumed that the memory system will be fabricated with the best available commercial technology and practice. The type of memory chips used will be commercially available with a known reliability record. With these remarks, consider a hypothetical memory consisting of sixteen thousand chips. Memory chip failure rate is generally 0.02% per thousand hours, thus the mean time between failures for this system would be 312 hours.

This mean time between failures includes the memory chip failure only. It does not include failures caused by fabrication, power supplies, or control-and-address logic. To account for these failures, a computed mean time between failures can be degraded by a factor, usually two or greater, depending upon the manufacturing and operating environment. Using a factor of two gives a system mean time between failures of 156 hours, which is slightly less than a week. A standard error correction method can improve this memory system reliability by a factor of about 100 times. The effect of using error correction would be to eliminate the memory system as a contributor to system unavailability.

One major issue with respect to error correction on the Illiac arises from the fact that while the memory is organized into 64-bit words, it is possible to write only a 32-bit half word. This creates a problem for error correction circuitry, since to calculate the appropriate encoding, the full 64 bits of information must be known if each 64-bit word is to be protected. There are two solutions: the first is to protect each 32-bit half word of memory; this is expensive in both memory storage requirements (requiring ten additional check bits per word instead of six) and in coding/decoding circuitry. The other alternative is to protect the memory in 64-bit words, but to require that a write in 32-bit mode take longer than in 64-bit mode: a 32-bit write would actually be a read of the entire 64-bit word, followed by a modification of that word (by the 32 bits to be stored), and then the new 64-bit word encoded and stored in the memory.

Ideally, any correction of a memory system error should be performed at the site of use, so that transmission errors can be corrected as well. Thus if a processor is the user of the data, then the processor should perform the error correction. The memory word width would simply be wide enough to store the error-correction bits along with the information bits. In the case of the Illiac PEM system, this concept is not feasible because the PEM system has three data paths: a CU data path, a PE data path, and an I/O data path. These data paths were designed and hard-wired, thus it would be a major task to modify the cabling to carry the error correction bits and to add encoders and decoders to the CU, the PEs, and the I/O system. Thus, the implementation of the error correction capability should be restricted to reside within each PEM module.

III.D.3. Virtual Memory

The major advantage of a virtual memory mechanism is that it relieves the programmer from explicitly moving data among the levels of a memory hierarchy. The major disadvantage is that it is usually associated with increased inefficiency in processor utilization, since much time is spent moving data around unproductively unless the programmer is very careful in structuring the way the program accesses the data structures. There is an additional difficulty associated with a virtual memory for an array processor like the Illiac. That is, each processor can generate an independent address into its local memory; thus each such instruction could generate a page fault (or up to 64 page faults on the Illiac). There are two ways to treat this within the current framework of the Illiac. The first is to assume that PE-indexed instructions will always produce page faults and simply wait for the longest possible time required to satisfy page faults. For this to be feasible on the Illiac, each PEM would have to have its own slice of extended memory, and be able to manage it independently of the other PEs. Also, page replacement would have to be extremely rapid; such would be the case if, for example, either a large random access chip is used for the extended memory. The second way to treat PE-indexed instructions is to limit the extent of changes that a PE can do to the address, for example limiting PE modification to the least significant 6 bits in the address. Thus the CU could determine in advance which page each instruction references and can arrange for that page to be accessible to the PEs before the instruction is executed.

The implementation of the second alternative would require modifying the PE to use only the low order six bits of the index registers and to inhibit carries beyond six bits. This would be a small PE modification. The CU memory service unit would need a table containing virtual addresses associated with the real pages in PEM. If ADVAST or the instruction look ahead unit process an instruction which references memory and the CU is in virtual mode, an interrupt could be generated if the page is not present, else the location of the page could be inserted into the page address portion of the address. Access to this table would have to be provided, as would protection from unauthorized manipulation (by the user). Other operations required for preventing faults and returning to the program could be left to the system. All this would involve noticeable modification to the CU, but not an excessive amount.

Simulations of the performance of a virtual memory on the Illiac have been performed. Two will be described here in detail. A Fast Fourier Transform (FFT) was chosen as the program to illustrate the virtual memory performance: this was because it is a major portion of several application codes (see section III.B.1.) and because the memory reference patterns of the transform exhibit both best-case accessing (many accesses to the same page) and worst-case accessing (each access to different pages). The memory that was simulâted had 64 pages of 128 rows each (for a half-million word local memory); the paging scheme used a least recently used algorithm.

The program performs a one million point FFT (actually a two-dimensional FFT on 1024 by 1024 data). There are six steps to the FFT program: a pre-processing step to arrange the data for efficient parallel processing, a FFT transform for the first dimension, a transpose, a second FFT pre-processing step, a second FFT for the second dimension and a post-processing step to arrange the data in a standard format. The data layout for the two transforms are different - the first has a page containing an entire row of the data whereas the second has each row of data spread over sixteen pages. The paging characteristics of these two arrangements vary considerably.

The percentage of memory references which generated page faults for each of the six steps were 1, 0.1, 2, 0.01, 0.0005 and 0.8 respectively. The paging characteristics of the two different FFTs are strikingly different:

0.1 percent page faults versus 0.0005 percent. The overall page fault rate was about .05 percent; if both FFTs were as efficient as the second there would have been virtually no page faults in the program (less than 0.0005 percent of the memory references). As a final note, all of the page faults in this program involved paging out modified pages before a new page could be brought in (except for the initial 64 page faults).

A second experiment in assessing the performance of a virtual memory was performed using the TRES code (see section III.B.1. for a description). The virtual memory configuration used in this analysis was the same as above. Each sweep through the data base required approximately 14,500 page faults, of which about 6,500 required a write to backing store. Since there are more than 16 million memory accesses per time step, less than 0.17% of all memory accesses will require a page fault.

Both examples suggest that paging faults on an ILLIAC virtual memory should average less that 0.1%, thus a ratio of no more than 0.1% to 1 for accessing backing store versus accessing primary memory would be tolerable: this would no more than double the run times of programs such as those examined above. (A degradation of a factor of two is an accepted standard expectation for using a higher level language instead of assembly language and the reason for using a virtual memory is the same as for using a higher level language: reduce program complexity and improve programmer efficiency.)

There has been little thought given to what the impact of restricting PE-indexing would be on program design and coding. Since all of the above modifications could be emulated in software with the current hardware, experiments along these lines should be continued to determine the page size, or number of modifiable bits, before deciding on the implementation.

III.D.4. On-Line File Storage

This subsection presents an initial attempt at estimating the amount of on-line file storage capacity that will be needed to support users of the Illiac system. The approach taken is to extrapolate from the total memory requirements of anticipated application programs to obtain an estimate of the amount of storage that could be required by sixty hours of Illiac processing time. It Assumes a speed improvement of a factor of two over the current execution times of these codes.

Table III.iv gives the relative frequency with which seven of the surveyed Illiac programs are expected to be run during the next year. Five of these have compute times of over two hours (and therefore even longer total run times); this implies that they will likely be run as several smaller steps with the machine being validated between steps as having no detectable hardware faults. For the programs which run for several hours, it is

Table III.iv.

Storage Requirements for Illiac Jobs

Applications	Runs week each	I/O Data per week	Temp Data per run	Run Time
ACTION	1	4200	1100	16200
I4TRES	1	1400	1400	5400
LANDSAT	1	2500	0	820
2D-STRATO	12	3700	0	6000
3D-COMPRESS	12	45	380	12000
3D-INCOMPRESS	12	70	450	19200
3D-TRANSONIC	1	200	210	31000

Notes: Data size in megabits; time in seconds.

likely that between steps the machine will be made available for other users. Thus one of the functions of the file system will be to hold check-point or restart dumps of these programs between steps. In addition, the Landsat programs may have several different users operating on a particular frame (data set), and it would be convenient to have this frame on-line between the different runs. Thus one of the functions of the on-line file system will be to hold very large Illiac files for short periods of time (about a week).

The total amount of file storage that would be required to keep all input, output and temporary data for these programs is about fourteen billion bits and the amount of Illiac processing time required by these programs is about twenty-five hours. If one assumes that the entire job mix would have the same ratio of storage requirements to compute time, then it is a simple matter to extrapolate these numbers to estimate the amount of storage needed to keep all data on-line for a week of Illiac processing. Sixty hours of Illiac processing time per week was chosen for the purpose of sizing the file storage memory. This gives a storage requirement of about thirty-six billion bits.

Since this projection is based on a processor speed improvement of a factor of two, which may be optimistic, and since many of the files generated during a run will be put onto tape to be examined at remote sites, the study specifies an on-line file capability of expanding to thirty-six billion bits, but the initial goal is an on-line file system with a capacity of eighteen billion bits. This facility could be implemented in two phases, with the first)phase being somewhat smaller. In any event, the size of the file memory should be at least four times the size of the extended memory in order to hold a re-start memory dump as well as have adequate room for staging transfers onto and off of the extended memory.

III.E. The Scope and Characteristics of Future Developments

This section summarizes the weaknesses of the current system, the future requirements placed on the system by forseeable applications, and the characteristics which are desirable to develop while making necessary replacements to the existing system. Taken together they suggest guidelines for specifying replacements and additions to the Illiac system to meet the computing demands of the scientific research community during the next five years. They also form a basis for evaluating various proposed alternatives.

As a background for this discussion, two points must be kept in mind. The first is that several components of the current system are in an advanced state of mechanical fatigue and will have to be replaced within the next few years; the most noticeable of these is the Illiac extended memory (I4DM). The local memory (PEM), while not being in such a precarious state now, nevertheless could become a critical problem near the end of the time frame under consideration. The second point is that the speed of the Illiac processor itself will increase over the next year or two, possibly by up to a factor of two. This will siginificantly increase the demands already being placed on the system, both in the size of the problems which become feasible for the Illiac and in the data movement rate required by the support system to move the large data files for Illiac processing.

With these two expected near-term developments in the Illiac system, it is instructive to review the current weaknesses of the present system implementation. There are five major areas of concern. These are:

1. The limited bandwidth into the system and the resulting time required to get data into and out of the system.

   Data movement between the Illiac system and the outside world currently ranges from sixteen kilobits (via the network) to 1.3--1.5 megabits per second (using the I4TAPE system). Currently available commercial disks and tapes can support up to eight times these peak transfers rates.

2. The lack of an on-line storage facility to keep active data bases between runs over a period of a few weeks.

   Currently, the Tenex file system is used not only for the support system utilities, but also for on-line storage of active Illiac data. As an indication of the seriousness of this problem, the current capacity of the Tenex system is 3 gigabits -- this is only six times the current (low) capacity of the I4DM. The lack of adequate on-line file storage is a serious problem which needs to be addressed if the entire system is to function effectively.

3. The low bandwidth between the file system and the array, and the resulting time to load and unload the array memory.

The current system to I4DM transfer rate is around one megabit per second. This leads to current programs requiring five to ten per cent of their compute time just to move data on and off the Illiac, and some codes spend half or more or their total run time performing such data movement. With a processor twice as fast, these percentages will double. Commercial disks are available with peak transfer rates of up to 87 megabits per second; such disks, used as staging devices, would significantly alleviate this problem.

4. The bandwidth between local and extended memory and the latency in accessing the extended memory.

The major factor in lost time during computing is the current disk latency which averages 20 milliseconds per read or write request. Non-disk technologies can reduce this by a factor of upwards of 100; this would effectively eliminate the software problems which users encounter in trying to minimize the deleterious effects of disk latency. The other alternative is to increase the local memory sufficiently so that fewer reads to the extended memory are needed. The analysis of the Illiac memory system concluded that the ratio of the size of local memory to the latency of extended memory should be upwards of 800 megabits of storage per second of extended memory latency. The bandwidth between the local and extended memory is required to be considerably greater than 200 megabits per second.

5. The declining amount of extended memory (I4DM).

The current disk memory on the Illiac is approximately eight million words. By reconditioning existing, non-functional disks, the capacity could be as much as 9.6 million words when all eight disks are operational. Current and projected application codes for the Illiac have data bases up to twenty million words and a few could use sixty-four million words. Allowing for collection of intermediate results during the course of a computation, an extended memory of twenty-four to sixty-four million words (four gigabits) should be adequate. More than sixty-four million words would probably not be useful to single users, however.

There are several other issues which are important to the Illiac system but which lie outside the scope of this study. These will be mentioned briefly. The first is the overall reliability of the Illiac, since clearly without a sufficiently reliable computing engine it makes little difference what the memory characteristics. The reliability record of the Illiac has been steadily improving during the last two years, and it is roughly comparable to the reliability records of other large-scale scientific computers. Replacing the memories and including error correction capabilities will also improve the reliability and availability of the machine.

The second issue involves the support system. There are two aspects to this issue: the inaccessibility of the entire system when the central system computer (Tenex) is unavailable and the increasing burden on Tenex for pre-processing and post-processing of data for the Illiac. The solution to the first problem is a dual system of some form -- either a dual-processor configuration or two separate processors of which either can be assigned by the operator to handle critical system functions when the other mal-functions or is required for other purposes. The solution to the second problem is either the capability to off-load computing requirements to a second processor (such as the B6700 or a second Tenex) or an enhanced central computing capability (such as a KL model PDP-10). The entire issue of the support system capabilities (apart from data movement and data storage considered in detail in this report) is important and will need to be addressed as the memory enhancements to the Illiac system are made. The next chapter considers various alternatives to enhancing the Illiac memory system.

IV. Future Alternatives

This chapter presents different memory enhancement alternatives which meet the memory needs of the Illiac as described in the preceding chapter. It is the product of several hardware and software system designers at the Institute. The three proposed memory systems each offer a balanced system to support the computing needs of the Illiac user community, although their individual characteristics differ considerably with respect to performance and philosophy of programming the Illiac. The organization of this chapter and the major results are summarized as follows.

The first section gives a survey of current memory technology. The most promising technologies for the local memory are high speed bipolar and NMOS random access semiconductor memory chips. The NMOS chips are slower, but their greater integration permits a local memory with greater capacity to fit into the space occupied by the current memory. Three memory technologies can be used for the extended memory: Charged-Coupled Devices (CCDs), high density random access semiconductor memories (RAMs), and high performance disks. CCDs are a relatively new technology for large scale computer memories and therefore represent some technological risk in being used in the extended memory. The random access memory chips are the most expensive on a cost-per-bit basis, and therefore will probably limit the size of the extended memory to sixteen million words. The performance of disk technology is such as to require a considerably large local memory to compensate for its access latency. For the on-line file storage memory, high performance disk technology is the most attractive alternative.

The second section describes various alternatives to replace the local memory and the extended memory of the present system, and it also discusses the capabilities of a proposed on-line file system for the Illiac. There are two possibilities for the local memory: one which is as fast as the current processor could possibly use and the other which is as large as the processor could possibly address. Because of space limitations, the first alternative is limited to be less than two million words; because the second requires the use of high integration chips, the access speed of this alternative is limited to be about the same as the current memory as it is now being used. For the extended memory, three technologies are appropriate: CCD, RAM and high performance disks; with these technologies, four alternatives are possible. The first two involve CCD -- one as a straight-forward disk replacement (using much of the current I4DM switching mechanism) and the other as a specially designed memory system. Although RAM technology could also be used as a disk replacement, its high performance access capabilities are better exploited by a special memory design which gives the extended memory the same access characteristics as the local memory. An extended memory based on disk technology would have slightly worse latency and significantly worse transfer capabilities in comparison to the current extended memory, thus requiring a significantly larger local memory than the Illiac currently has, but the cost per bit would allow for a significant increase over the current capacity of the I4DM. The conceptual design presented for a disk based extended memory can also be used for the on-line file memory. The Illiac file system will manage this file memory; in addition, since the file memory will be used for staging jobs onto and off of the array, the file system will automatically manage this task as well.

The third section describes and analyzes three balanced memory systems composed of the system components presented in the second section. The first system is characterized by a four million word local memory and a quarter billion word extended memory (which uses high performance disks). For users requiring more than four million words of memory, program design will require the same considerations as programs for the current memory configuration, but since this proposal has overall performance characteristics about ten times better than the present system, this improvement should be reflected in eliminating the need for careful program tuning and data arrangement on the disk which currently characterizes the development of efficient Illiac codes. The second proposed memory system consists of a half million word local memory, a sixty-six million word extended memory (using CCD technology) and a special buffer/file memory. Although again the extended memory is similar to a disk, its latency and transfer characteristics are significantly better than the present disk memory. This will mean that even programs with very simple data movement between local and extended memory and simple data arrangement on this extended memory will still make efficient use of the computing resources of the machine. The third proposed system consists of a sixteen million word extended memory (using RAM technology) and a buffer/file memory. The extended memory of this proposal supports the same accessing capabilities as the local memory, so that the entire array memory can be treated as homogeneous when designing Illiac programs; this is expected to reduce the time required to design Illiac codes by a significant extent. Aside from programming considerations and memory capacity, the three proposed systems differ significantly with respect to the amount of in-house development required, initial cost for a minimum configuration and impact on the existing system hardware and software.

The final section reviews the major weaknesses of the present system and indicates how these limitations are corrected by the proposed memory system alternatives. This section, then, presents a summary of the entire report.

IV.A. Technology Survey

Four memory technologies are briefly reviewed in this section: disk technology, charge coupled devices (CCD), magnetic bubble memories and random access memories (RAM). The information is provided on devices as of the first quarter of 1978. Manufacturers are suggested merely to demonstrate the availability and typical characteristics of each type of device. Listed characteristics of each technology are based on what appears to be the best overall characteristics with respect to the Illiac memory system; particular devices can be obtained with some characteristics better, but at the expense of other characteristics. Pricing information is provided for bugetary purposes only. Estimates tend to reflect the quantities necessary to build very large systems but an actual bid will, of course, reflect the marketing considerations of the manufacturer. One might also note that systems houses associated with chip manufacturers can frequently use selected partials (less than 100% perfect chips with only the known good storage circuits being used by the system) for a less expensive memory system with the same performance characteristics and only a slight degradation in reliability (due to more chips being used).

The technologies which are leading candidates for implementing an extended memory are CCD, NMOS RAM and high performance disks. The greater in-the-field experience with 16k dynamic RAM gives some preference to an NMOS RAM selection. The use of a high performance disk would require a very large local memory in order to balance the total system.

Replacement of the current PEM is more straight-forward since one need only consider RAM devices. Because of the processing speed requirements, static devices with access times less than 200 nanoseconds are considered. The use of a high density, dynamic RAM chip for the local memory is an interesting alternative if memory size is more important than access speed.

For the Illiac file storage system, disk technology is the preferred medium because of the attractive cost per bit. The major question to be resolved in this area is the transfer rate between the file storage system and the array; since a high transfer rate is required (unless multiple users can reside in the extended memory simultaneously), either a high performance buffer is needed or the entire file storage system must have sufficient performance to act as an on-line buffer storage between the array memory and the archival memory system of the installation.

IV.A.1. Disk Technology

There are two types of disks: moving head and fixed head. The main differences between the two are that moving head disks have a more complex mechanical assembly (and hence a poorer reliability record), significantly less cost per bit, and a higher density of data per unit volume; however, fixed head disks, with a greater number of read/write heads, have a greater potential for much higher bandwidth (although it would require modifications to the commercially available disks to realize this potential). The high latency time and low transfer rate of disk technology make most commercially available disks poor choices for the extended memory, but special high performance disks could be used in connection with a very large local memory.

If disk technology is used for the extended memory, the drives would have to be synchronized as are the current disk drives. This would require special hardware for this purpose, as the need for synchronized disks rarely arises in the commercial market. An alternative would be to supply each disk with a track buffer capable of holding an entire track of a disk. The disadvantage of this approach is that changing tracks incurs the full latency time of the device. In addition, this approach implies a page size equal to the track capacity, which in turn implies the need for a larger local memory for the Illiac.

IV.A.1.a. Moving Head Disks

Two types of moving head disks will be considered. The first type is the familiar serial transfer IBM style drives which provide transfer rates up to 10 megabits per second. The second type is a more specialized unit with multiple read/write amplifiers to effect a parallel transfer capability up to 87 megabits per second.

IV.A.1.a.i. Serial Transfer Disks

Manufacturers:	CDC 844-21
	AMPEX DM9300
	CalComp Trident series
	ISS 7833
	STC 8350 mod A2
Read/Write Data Rate:	10 megabits/sec (for 330 compatible disks)
Maximum Latency:	16.7 milliseconds rotation
	55 milliseconds maximum seek time
Power Consumption:	300 watts/drive
System Bandwidth:	0.160 gigabits/sec (16 drives)
	0.64 gigabits/sec (64 drives)
System Price:	0.0015 cents per bit
	$35,000 per disk, drive and controller
Reliability:	3000 hours mean time between failures

The major advantages of moving head disk technology is the low cost per bit and the availability of a standard (IBM 3330 or IBM 3333 compatible). If these disks were used for the extended memory, the moving head drive could significantly complicate disk mapping problems by introducing an additional mapping dimension with a different latency behavior. This additional latency and complexity could be counteracted by a local memory large enough to reduce the number of required accesses (and hence the total latency time). The bandwidth of using sixteen 300 megabyte drives is only 160 megabits per second where a higher bandwidth is more appropriate. Given the reliability of moving head disks, it is inadvisable to increase the number of drives in order to increase the bandwidth.

IV.A.1.a.ii. Parallel Transfer Disks

Manufacturers:	CDC 819-2
	Ampex DMP-9309
Read/Write Data Rate:	87 megabits/sec (for DM930X)
Maximum Latency:	16.7 milliseconds rotation
	55 milliseconds maximum seek time
Power Consumption:	400 watts/drive
System Bandwidth:	0.340 gigabits/sec (4 drives)
System Price:	0.004 cents per bit
	$85,000 per drive and controller
Reliability:	2500 hours mean time between failures

An alternative to the serial transfer disks would be to use the more expensive CDC 819-2 with a 38 megabit per second data rate per drive. Sixteen of these drives would give a more respectable 608 megabits per second system bandwidth. The penalty here is a more than doubling of the price per bit.

An even more attractive alternative is the Ampex DM930X Parallel Transfer Drive. This recently announced product is a single spindle, direct access storage device which uses a standard IBM 3336 Mod II-type removable disk pack and contains 815 cylinders of 19 data tracks. Each data track has a capacity of 161,000 bits. When formatted, the capacity is about 144,000 bits. Data transfer is accomplished using nine heads simultaneously (optionally six or four) for a transfer rate of 87 megabits/second. The maximum data skew is 128 bits. The maximum rotational latency is 16.7 milliseconds (8.35 milliseconds average) and the maximum seek latency is 55 milliseconds (28 milliseconds average). Drives can be synchronized by providing a clocked AC power source from the controller which synchronizes the drives on the index pulse. The base price for each Ampex DMP-9309 drive is projected to be $45,000. Deliveries are scheduled to begin in the fall of 1978.

IV.A.1.b. Fixed Head Disks

Manufacturer:	Amcomp 8500
Read/write Data Rate:	8.7 megabits per second
Maximum Latency:	17.2 milliseconds
Power Cosumption:	350 watts per drive
System Bandwidth:	0.139 gigabits per second (16 drives)
	0.556 gigabits per second (64 drives)
System price:	0.2 cents per bit
	$35,000 per disk, drive and controller
Reliability:	10,000 hours mean time between failures per disk

Fixed head disk technology has not made great strides in recent years due in part to diminished market demand and to technological limitations. It is still price competitive with solid state technology (such as CCD and RAM) and will remain so for the near-term future. Recently, however, moving head disk technology has eroded the fixed head disk market as users found ways to circumvent seek latences with overlapped seek and space allocation algorithms. There is no standard for fixed head disk drives (de facto or otherwise) so procuring fixed head disks as a system component entails commitment to a single supplier.

IV.A.2. Solid State Rotating Devices

There are two types of solid state memory technologies which have access characteristics similar to disks in that accessing requires a latency period while the requested data is positioned under a read/write sensing device. These memory technologies are magnetic bubbles and charged coupled devices (CCDs). Compared with disk latency, the dramatic decrease in latency of these two types of memory devices make them outstanding choices for the extended memory. Their cost per bit precludes them from serving as the media for the file system. CCD has preference over magnetic bubbles on a performance basis.

IV.A.2.a. Charged Coupled Devices

Manufacturers:	Intel 2464
	Fairchild CCD464
	Texas Instruments TMS3064
Organization:	64 Kbits as 2 loop structure
Read/Write Data Rate:	2.5 megabits per second per chip
Maximum Latency:	260 microseconds (2464 only)
Power consumption:	75-300 milliwatt per device
Natural Bandwidth:	2.56 gigabits per second
System price:	0.05 cents per bit

CCD offers a system bandwidth comparable to a random access memory at a price significantly less. The power consumption per bit is also much less. The latency, however, is still two orders of magnitude greater than a random access memory, limiting its usefulness to multi-word transfer situations (such as the extended memory).

In mid-1978 CCD will be available with TTL compatible inputs and outputs (the Intel 2464, for example) and with simple clocking schemes. The availability of CCD devices should be good in comparison to other solid state memory devices due to the greater tolerance CCD has to crystallographic faults and the fewer masking steps than are required to fabricate RAM devices. A distinct disadvantage to CCD technology as it exists now is the lack of industry-wide standardization. The three CCD chips listed above, for example, are non-interchangable, although each of these manufacturers has designated a secondary source for its device. To date, secondary sources for CCD devices have not been particularly active, but this is expected to change with the anticipated market demand for 64K CCD devices.

CCD can be thought of as a rotating drum memory with 256 bits per track and 256 tracks (the organization of Intel's 2464, for example). A read or write can be accomplished in a page mode at one position across tracks at a rate of 2.5 megabits per second or in a serial mode by track at the rate of one megabit per second. Page mode I/O operation consumes the least power and is the recommended method for using CCD devices in general. Of the three chips listed above, only Intel and Fairchild chips have page mode. Spurious error rates in CCD have been very high, and industry sources suggest that a memory system using CCD components should be periodically re-written using one bit error correction circuitry to correct any bits that might have failed. Even if spurious error rates do not necessitate this action, error correction logic should be used for all I/O operation to maintain a high system mean time between failure.

IV.A.2.b. Magnetic Bubble Memories

Manufacturer:	Texas Instruments TBM0103
Organization:	92K bit as 157 by 144
Read/Write Data Rate:	50 kilobits per second per device
Maximum Latency:	8 milliseconds
Power Consumption:	0 - 1.5 watts per device
System Bandwidth:	51.2 megabits per second
System Price:	0.05 cents per bit

Magnetic bubble memories are a non-volatile storage medium which, like CCD, appear to have a drum-like organization. The data rate and latency are more than an order of magnitude slower than CCD. The price per bit is roughly comparable to CCD at the present time but is expected to be somewhat less in the near future.

The primary attractiveness of magnetic bubble memory is its non-volatility; this characteristic is not relevant for the extended memory since this memory is not required to retain information in the absence of power. The slow transfer rate of the device means it would be necessary to multiplex the device by a large factor in order to meet the projected bandwidth requirements of the extended memory. Magnetic bubble memories are still in pilot production with limited availability from the manufacturers. No standard exists so devices will probably not have a second source in the time frame of this study.

In summation, magnetic bubble memory is not a viable choice for the memory technology in any of the memory components of the Illiac system.

IV.A.3. Random Access Memories

Semiconductor technology is also used to make randomly accessible memory devices (RAMs). For the purpose of this study there are two (mutually exclusive) characteristics of memory devices which make RAMs of interest for use in the Illiac memory system: high speed memory (bipolar memory technology) and high density memory (NMOS technology). The high speed memory is of interest to support the high speed of the Illiac processor; the high density memory generally is the least expensive on a cost per bit basis and is therefore attractive for large memories.

IV.A.3.a. Static Bipolar and Short Channel NMOS Memories

Manufacturer:	Fairchild 93470 (bipolar) Intel 2147 (Short Channel NMOS)
Organization:	4096 bits by 1
Read/Write Data Time:	50 nanoseconds
Power Consumption:	800 milliwats per device
System Price:	0.4 cents per bit

Bipolar and short channel NMOS memory chips offer about twice the speed of standard NMOS memories of comparable size, at about four times the cost. The only memory component which requires such speed, and can tolerate such cost because of its small size, is the local processor memory. The speeds of the above devices, for example, are sufficient to meet the performance constraints of a 16 megahertz Illiac clock. Faster devices are available (with lower density), but the additional speed cannot be used to advantage by the processor.

IV.A.3.b. Dynamic NMOS Memory Chips

Manufacturers:	Mostek MK4116-2 Intel 2116-2 Texas Instruments 4116
Organization:	16384 by 1
Read/Write Data Time:	150/375 nanoseconds
Power Consumption:	20-462 milliwatts per device
System Bandwidth: non-interleaved)	2.7 gigabits per second (16 modules,
System price:	0.1 cents per bit

Manufacturers have been producing 16K NMOS dynamic RAMs for nearly one year. Production volume has lagged demand by a significant margin and this is expected to continue until 1979. This means that a dramatic decrease in price per bit should not be expected over the next year. Like CCD, NMOS memories are subject to spurious bit errors so that error correction circuitry must be incorporated in any large NMOS system to ensure reliable performance. Periodically rewriting data through the error circuitry would probably not be necessary due to the short life-time of the data (a maximum of two hours on the extended memory).

Dynamic RAM provides a moderately priced solution to large memory systems that must have both high bandwidth and very low latency. If access to such a memory system is expected to be sixty-four or fewer words per access, then dynamic RAM is a very attractive choice. Dynamic RAM has also become standardized with several manufacturers making compatible devices, thus insuring availability of vendors for system components and spares. This also ensures vigorous price competition after manufacturers have saturated market demand.

If 16K dynamic RAMs are used to replace the local memory of the processor, page mode accessing can produce a significant savings in cycle time. This can be illustrated with the Mostek chip. This device has a read access time of 150 nanoseconds and a cycle time of 375 nanoseconds. This means that the first access can be made in 150 nanoseconds but that subsequent accesses to the same device would take 375 nanoseconds. In order to improve subsequent accesses, a special page mode is provided. Page mode allows subsequent reads to take place in only 170 nanoseconds if the addresses are in the same row (128 bit segment) of the device. Similarly, the first write in page mode takes 95 nanoseconds with a subsequent writes into the same row taking 170 nanoseconds each. Another advantage of page mode is a thirty per cent decrease in power consumption. A disadvantage to using dynamic devices for the local memory is the refresh requirement. This must either be done on a piece-meal basis, say every 16 clocks, or by having the control unit stop processing periodically (every 2 milliseconds) and wait for refresh (which takes 48 microseconds). Block refresh time thus accounts for about 2.5 per cent of the time (which cannot be overlapped with computing unless a separate memory is available which does not require refreshing). Interspersing refresh with other instructions which do not access memory requires modifications to the control unit which are more complex than those required for block refresh.

IV.A.3.c. Core Memories

Manufacturer:	Ampex Megastore
Organization:	250K by 20 bit word modules
Read/Write Data Rate:	500K 20-bit words/second
Maximum Latency:	2.5 microseconds
System Bandwidth:	1.6 gigabits/second (4-way interleaved, 16 modules)
System Price:	0.1 cents bit or less
System MTBF:	290 hours without SEC/DED

The Ampex Megastore is a magnetic core based technology where the "cores" are punched out of a magnetic tape-like medium. The basic building block is a 250Kx20-bit module which measures 17x19x1.5 inches. Two hundred and forty of these modules would be requires for 16M words. The current speed is 2 microseconds access and 2.5 microseconds cycle but systems are usually 4 to 8-way interleaved to effect a higher transfer rate.

Modules have a calculated mean time between failures (MTBF) of 14,000 hours but current field experience indicates 5 to 7 times this figure may be more appropriate. A system of 240 modules with 5 times the 14,000 hour/module failure rate would have a system MTBF of 240 hours. This indicates that error correction might not be required. If the reliability is closer to the calculated value then some error correction capability would definitely be required. Error correction in this instance is complicated by the fact that these memories commonly have a 4 adjacent bit failure mode rather than the single bit failure mechanism usually associated with random access memories. This means that a system module would have to contain four error correction circuits accross a 288-bit word (four 72-bit words) where each group of four adjacent bits from the same board would be mapped into four separate 72-bit words.

The circuit cards cost about $5,000 each but a system of the size envisioned might be significantly less than the calculated 0.1 cents per bit. Systems have been in the field for over one year and no significant problems are anticipated in the construction of a sixteen million word system. A six to twelve month delivery schedule (depending on whether the Institute is willing to wait for faster units) should be expected.

Such a memory might be considered as an alternative to the use of dynamic NMOS memory chips. This technology avoids the complexities of refresh and the power consumption is less. The system cost would be comparable. The major drawback to its use is the higher latency in accessing this memory, but this may not be significant.

IV.A.4. Concluding Remarks

Either 4K static RAMs or 16K dynamic RAMs are well suited for use in the local memory: the former would give the maximum speed while increasing the capacity while the latter would, within the current board space, allow the local memory to reach its limit of four million words. For the extended memory three technologies are attractive: high performance (parallel transfer) disks give the greatest capacity, RAM devices give the greatest flexibility and performance, and CCDs give an attractive intermediate solution between these two extremes. For the on-line file memory, high performance disks provide the greatest capability for efficient utilization of the computing resources of the Illiac system.

The choice of a memory technology for a particular system component rests not only on performance characteristcs and price per bit, but also on availability in the marketplace. This is true not only with respect to initial procurement, but also in regard to the expected lifetime of the memory component. There are several aspects to this. For example, the current Illiac extended memory is no longer supported by its original manufacturer and this has exacerbated its maintenance problems. Thus one clearly wants to choose a memory technology which will be commercially viable. It appears that all of the technologies considered above will be commercially supported during the expected lifetime of the memory system so that maintenance of the memory system will not be complicated by the lack of support by the industry.

IV.B. System Component Alternatives

This section presents various architectures and implementations for major system components in the Illiac memory system. The components are the processor's local memory and extended memory and the Illiac file system. Two alternatives for the local memory are discussed: one using a 4K static memory chip and the other using a 16K dynamic memory chip. Both alternatives represent the current state-of-the-art for their respective technologies.

For the extended memory, three approaches meet the performance demands required of this system component. Two of the technologies are solid state technologies: Charged-Coupled Devices and random access memory chips. The other is based on a high performance disk recently announced. The solid state memories have the advantage of being easy to repair (basically chip or card replacement); the disk is based on a standard commercial family of disks and so should be easier to maintain than the current I4DM, which is no longer commercially supported. All three memories have error detection and correction so that reliability should not be an issue.

The final system component to be dicussed is the on-line file system. This memory will use disks because of the advantageous cost per bit for this storage medium. For the purposes of discussion, the high performance disk used in the above proposed extended memory will also be used in this component, largely because this memory may be required to hold the active data base of a few of the problems which will be run on the Illiac, and hence high performance will be necessary. The main function of this memory, however, would be to buffer transfers of Illiac files between the processor and the support system. In addition, it could also be used to hold back-up or re-start dumps for programs and intermediate results to be used in multiple-run Illiac jobs. Thus this memory will have a file mechanism associated with it, and a description of a proposed Illiac file system constitutes the bulk of the discussion about the on-line file memory.

IV.B.1. Local Memory Replacements

The major options involved in the replacement of the PE memories are the choice of memory chips and the inclusion of error correction circuitry. Since the PE uses the same register to communicate both address and data to the memory, the memory response time is limited to two cycles -- the first for the address and the second for the data. Given the possibility of a sixteen megahertz clock cycle, this means that memory chips with a system cycle time of under 120 nanoseconds would provide as high performance memory as the processor will be capable of using. Currently a few bipolar and specially selected NMOS memory chips meet this requirement with an integration level of 4096 bits per chip at a cost of around $20 per chip. The next two years should show an increase in the number of devices meeting the speed requirements as well as a drop in prices. Of course, slower chips could be used -- the current Illiac memory is treated as an ten-cycle memory, for example. The result in using slower chips would be less performance, but the cost per bit would be less and the level of integration (and therefore total size of the memory) could be greater.

Systems constructed from many components exhibit failures, and the larger the number of components, the more frequent and hence more noticable these failures become. For memory systems, techniques have been developed to detect and correct a high percentage of the most frequent types of memory failures; the major question is deciding the size of memory which requires that these techniques be employed. Initial field experience with the CRAY-1 computer suggests that for memories constructed out of solid-state RAMs it may be necessary to have error correction in systems with as few as 32 thousand chips. More recent field experience with the Cray-1 memory shows that semiconductor memory can be very reliable, to the point of not requiring error correction; the publicity accorded to the initial Los Alamos experience, however, makes it unlikely that another Cray-1 will be built without error correction. On the other hand, 32 thousand is the number of chips in the present Illiac local memory, and the current memory is not a major source of system errors. In addition, as the semiconductor industry develops its expertise in designing and fabricating memory chips, the reliability of successive products tend to reflect this experience in improved reliability, although this trend has in the past been counter-balanced whenever new fabrication techniques are adopted and when the geometries of the devices reach critical dimensions. In short, it appears that the 16K RAM chips are dramatically more reliabile than the preceding 4K and lesser density chips, even on a chip to chip comparison basis (rather than an error per bit comparison basis). In any event, this section assumes that a local memory of eight to sixteen thousand 4K chips will not require error correction, but larger systems should have this capability. All memory replacements will have the excess storage capacity for error correction, however, so that if this capability is added later, only the error correction logic circuitry will have to be added (in the Memory Service Unit). The cost of this extra storage is about 12 percent since this is the amount of redundancy that is required for error correction.

Two types of PE memory replacements are described. The first uses 4K static RAM chips and behaves as the current PE memory does; it is therefore viewed as a simple memory replacement, and the only change visible to the user is a change in the size of memory. The other type uses 16K dynamic RAMs, and since this type of memory chip behaves differently in this context, both from the user's point of view and the system's point of view, more detail will be given to its implementation and mode of use.

Although the electronics are in place to support a 16-bit PE addressing capability, only 11-bit addresses are currently implemented. To increase the addressing capability, 15 wires and 5 terminators must be added to the Memory Service Unit. This requires about three hours per PE. Since the cabinet power must be shut off to make these modifications and since such an action is quite disruptive of the normal array operation, even after power is restored, this task should be done only on a spare basis or during scheduled power shut-down of the array (during the annual year end shut-down, for example).

The following contribute to the cost of replacing PE memory: design cost, chip cost, memory board cost, back-plane cost, and error correction cost. Design cost is estimated at $35,000 per board type; chip costs vary with quantity purchased as well as date of purchase: the costs used here are for bulk quantity procurred during late 1978. The costs of memory boards, including fabrication and mounting chips, also vary with design and are estimated at $500 per fully loaded board, $350 for a half-loaded board, and $250 for boards which are less than half-loaded. The cost for error correction is estimated at $60,000 for design and $200,000 for fabrication.

For all alternatives described below, it is expected that the current air conditioning is sufficient to dissipate the additional heat generated by the PE memory replacements and that current power supplies are adequate.

IV.B.1.a. Local Memory Using 4K Static RAM Chips

The current 4K static RAM chips are organized as 4K by 1 bit, thus using a chip for a bit slice of a PEM yields a minimum sized replacement of 4K words per PE or a total PEM size of 256 thousand words. Such a replacement would require 5000 chips at about $25 per chip and 300 memory boards at approximately $250 per board. Allowing $35,000 for design costs, this memory would cost approximately $235,000.

Doubling the number of chips would give a half-million word PEM. The only difference between this alternative and the previous one is the number of chips. Ten thousand chips at a cost of about $20 per chip would mean that this memory would cost about $310,000.

One and two million word memories using 4K static RAM chips would require error correction for an additional cost of $260,000. This is offset somewhat by the declining cost per chip when purchased in substantial quantities. Twenty thousand chips at $16 per chip would be required for a one million word memory, and forty thousand chips at about $12 per chip would be required for a two million word memory. Also, the boards themselves will be more costly to manufacture because of the additional number of chips on each board; it is estimated that for the one million word memory this fabrication cost will be $350 per board and for the two million word memory about $500 per board. In addition, new PEM backplanes would be required at a cost of about $35,000. Thus the total cost for a one million word PEM is estimated at approximately $755,000 and for a two million word PEM about $960,000.

In the 1980 time frame, the chip cost would be about half that used above. Thus a 256 thousand word PEM would cost about $160,000, the half million word PEM replacement about $300,000, the million word PEM about $600,000 and the two million word PEM approximately $725,000.

IV.B.1.b. Local Memory Using 16K Dynamic RAM Chips

The use of dynamic memory chips introduces several characteristics which are absent in memories which use static chips. Most obvious is the need to refresh the dynamic memories and the more complex control circuitry to support refresh. In addition, dynamic memories are generally slower. On the other hand, greater integration is possible with dynamic memories, and the cost per bit is lower than for static memories.

For the Illiac application, the refresh requirement can be met in several ways. One approach is to assign every sixteenth clock cycle to memory refresh. Thus if there is an instruction being executed during such a cycle which does not reference the memory, then it can be executed simultaneously with the refreshing mechanism, otherwise it will remain in the FINST queue until the memory is available. There is already in place in the CU the mechanism which will support such scheduling: the scoreboard unit. By suitable modification of the CU to keep track of the appropriate cycle to refresh and the memory address to be refreshed, this method of memory refresh would have the least impact on processor performance. A simpler method would be to interrupt the control unit periodically to refresh the entire memory en masse. Although this would have noticeable impact on array performance (degrading computing rate by about five percent), it is simpler to implement since it requires only that the interrupt mechanism already in the control unit be brought to an operational status.

Replacing the current 256-bit chips with 16K chips allows for a sixteen million word PEM, however the address register being only 16 bits limits the memory to four million words. For a memory of four million words, error correction circuiry may be needed, at a cost of $160,000. Design of the memory boards is estimated at $35,000 and the cost of the PEM backplanes is approximately $35,000. A four million word memory requires 20,000 chips at a cost of about $20 each, plus 300 memory boards which cost about $250 to fabricate and load with memory chips. Thus a four million word memory would cost approximately $800,000.

The dynamic memory chips are slower than the current memory chips now in the Illiac, thus if the Illiac memory is operating at its maximum speed (presumably by the end of this year), installing a memory based on dynamic chips will cause a decrease in performance unless suitable steps are taken. One possibility is to use the special page mode feature of some dynamic RAM chips. This can be described briefly as follows. RAM chips are organized as a matrix of bits. To access a memory bit, one first selects a row and this row is loaded into an internal memory buffer. Next one selects a bit in this internal buffer to be accessed. If the next memory reference is to this same row, then the first part of the sequence can be omitted, and therefore references require less time.

For the user, this process would appear as follows. The array memory consists of a collection of memory banks or modules. Access to any module brings a page into a most favored page status. Subsequent accesses to that same page can be made at a much faster rate than to a different page in that same module, provided the access is made within 64 memory cycles. Keeping track of which memory addresses can be accessed rapidly is a function which could be automated by a compiler (using the rule of thumb of when in doubt, assume not). In order for this feature to be transparent to previously compiled programs, the hardware implementation must be compatible with the current instruction set. The method chosen for this study was to add twelve instructions to the existing instruction set which take only four cycles to fetch operands from memory in page mode--all the existing instructions which reference PEM are assumed to require eight cycles to fetch operands from the memory. (this is, of course, the current memory performance.) The twelve new instructions would be floating-point add, subtract, multiply and divide (normalized and rounded), and the four load and four store instructions: to the A, B, X and R registers.

Simulations have been performed on the expected behavior of such a two-access speed memory. One simulation was a million-point, two dimensional Fast Fourier Transform. The result of this simulation was that 59 percent of the memory references could have been made in the fast access mode. This would represent a savings of up to 12 seconds (depending upon how much memory accessing could be overlapped with computing) out of a total run time between 60 and 120 seconds (extrapolating from current Illiac performance and allowing for processor speed improvements). The percentage of fast accesses was greater in the data re-configuration phases of the algorithm (up to 72 per cent), but this accounts for less than one percent of the compute time of the algorithm. As another example, an examination of the TRES code (described in section III.B.1) showed that all but 0.1% of the memory accesses could be performed in the fast mode. Thus the speed improvement for this particular application could be as much as 25%. This points up how application-dependent the performance of this type of memory would be. In both cases, it is interesting to note the relatively modest increase in processor speed which can be expected from this approach. Thus the additional complexity of adding new instructions and keeping track of page activity (either in software or by the programmer) does not seem to be justified on the basis of an expected speed improvement of only ten to twenty percent.

The schedule for implementing an enhanced PEM contans three tasks in addition to modifying the PE addressing lines. The first is the design of the memory boards and support facilities. This is estimated to take six months. Next there follows a procurement cycle of from four to nine months. The final phase involves the actual fabrication and is estimated to take a minimum of nine months. Thus the total schedule requires from nineteen to twenty four months before the memory is ready for installation and testing. Installation could be performed incrementally, but the entire modification to the control unit for timing memory references will have to be complete before the first PEM modification. Testing and installation is expected to extend over six months.

IV.B.1.c. Discussion

The major advantage of using a 4K chip in the PEM replacement is that the speed is sufficient to match the processor requirements. In addition, for half a million or one million word memory, the cost is $300,000 and $600,000, respectively, and the system impact is negligible -- essentially this approach is a board for board replacement and the addition of some PE backplane wires. It has the further advantage of accomplishing what will probably be done for general maintenance purposes during the early 1980s anyway, namely the replacement of the PE memory boards themselves. However, it is still necessary to address the I4DM replacement and the size of local memory suggest that an extended memory using either CCD or RAM will be required.

Using a 16K dynamic chip for the PE memory makes a four million word memory feasible. Many of the current codes could run effectively core-contained, and such a large memory could supply buffer areas large enough to hide the latency of a disk based extended memory for those programs requiring much larger memories. In addition, provision must be made in the control unit to accomodate memory refresh, either implementing the interrupt mechanism or using a more elaborate mechanism. Dynamic chips are slower (but still slightly faster than the current PEM is now being used) and require more support electronics for their use than static RAMS. The basic chip performance could be improved by a factor of two by using page mode, but this would entail altering the control unit for decoding additinal instructions and modifying compilers to exploit these new features. Since simulation results suggest that in practice page mode would increase processor performance by less than twenty percent, this additional complexity seems unwarranted.

Both of these approaches are well within the state-of-the-art for memory technology and at least one vendor has expressed an interest in bidding on a procurement should the Institute elect one of these alternatives.

IV.B.2. Extended Memory Replacements

Three technologies can be used to implement a replacement for the I4DM: Charged-Coupled Devices, random access semiconductor devices and high performance disks. This sub-section presents conceptual designs of several memory systems in order to indicate the feasibility of using such devices and to indicate the expected system performance characteristics that are possible. Each memory alternative has the minimum performance characteristics specified in Chapter III as required of the extended memory of the Illiac system.

IV.B.2.a. Charged-Coupled Devices for Extended Memory

Charged-Coupled Devices (CCDs) are an attractive choice for implementing the extended memory of the Illiac memory system since their maximum latency, typically one millisecond, is a factor of forty better than the current I4DM, and thus a CCD extended memory would significantly reduce the major programming difficulty in using the Illiac. In addition, its cost per bit, in delivered memory systems from commercial vendors, makes a sixteen million word extended memory fiscally attractive in the 1979 time frame, and a sixty-six million word memory fiscally possible in the 1980 time frame. On the other hand, CCD memories have not yet been delivered using the 64K chip that this application would require, although this situation will change during this year.

This sub-section describes two approaches in using CCD memories for the extended memory. The first is a simple disk replacement: the philosophy behind this alternative is to have the least system impact and simplest possible implementation of a replacement for the current I4DM. The second approach is to design a CCD-based external memory without the artificial constraint of compatibility with the current I/O sub-system. These two alternatives are presented in turn. One might note that the modules acquired for the first alternative could be specified so that they could be used in the second alternative at a later date. This approach may be attractive if the Institute elects to perform the bulk of the interfacing. On the other hand, at least one vendor has expressed an interest in supplying both memory and interface for the second approach, thereby relieving the Institute of a major design and fabrication task.

IV.B.2.a.i. CCD as a Disk Replacement

An I4DM replacement with the least system impact would be one that directly plugs into the Disk File Controller (DFC) and replaces the Electronics Unit (EU) and its associated disk drives (see Figure IV.1). The EUs contain the analog electronics and the data de-skewing registers for the disk drives. Each EU handles up to eight drives. Replacement of only the disk drives would involve creation and reception of 128 analog signals per unit. This approach would reduce the generality of the replacement module and be expensive to implement; therefore it is not recommended.

Replacement of the DFC and interfacing the replacement at the PE cabinet will be considered in sub-section IV.B.2.a.ii. Replacement of the DFC will have significant software impact and if one is to take full advantage of the CCD memory characteristics it will have a major hardware impact as well.

The EU delivers and receives data on 384 (3x128) bi-directional data lines to the DFC. A CCD replacement for the EU and the disk drives would have to provide a compatible data pathway unless the DFC were modified to accept a smaller word width. The DFC interface to the EU is a three-way multiplexor/demultiplexor which reduces to a 128-bit word width; thus the DFC logic could be modified so that the multiplexor/demultiplexor is bypassed if it were found to be inconvenient to design the CCD memory system for a 384-bit word width or if a commercial system is available which delivers a 128-bit word. Since this possibility involves DFC changes which would severely impact current system hardware, it should not be considered as a simple plug-compatible replacement.

The current SU (disk) organization is 512 tracks per drive (256 tracks on each side). It is organized as four bands of 128-bit parallel words. Each band in turn is made up of 1200 sectors, each sector containing 128-bit words. It currently takes 32 microseconds to read a sector with an average access time due to rotational latency of 19.6 milliseconds (if storage layout on the disk is not taken into account in issuing read/write requests). The current address space allows four bits for drive selection, two bits for band selection and eleven bits for sector selection; thus a

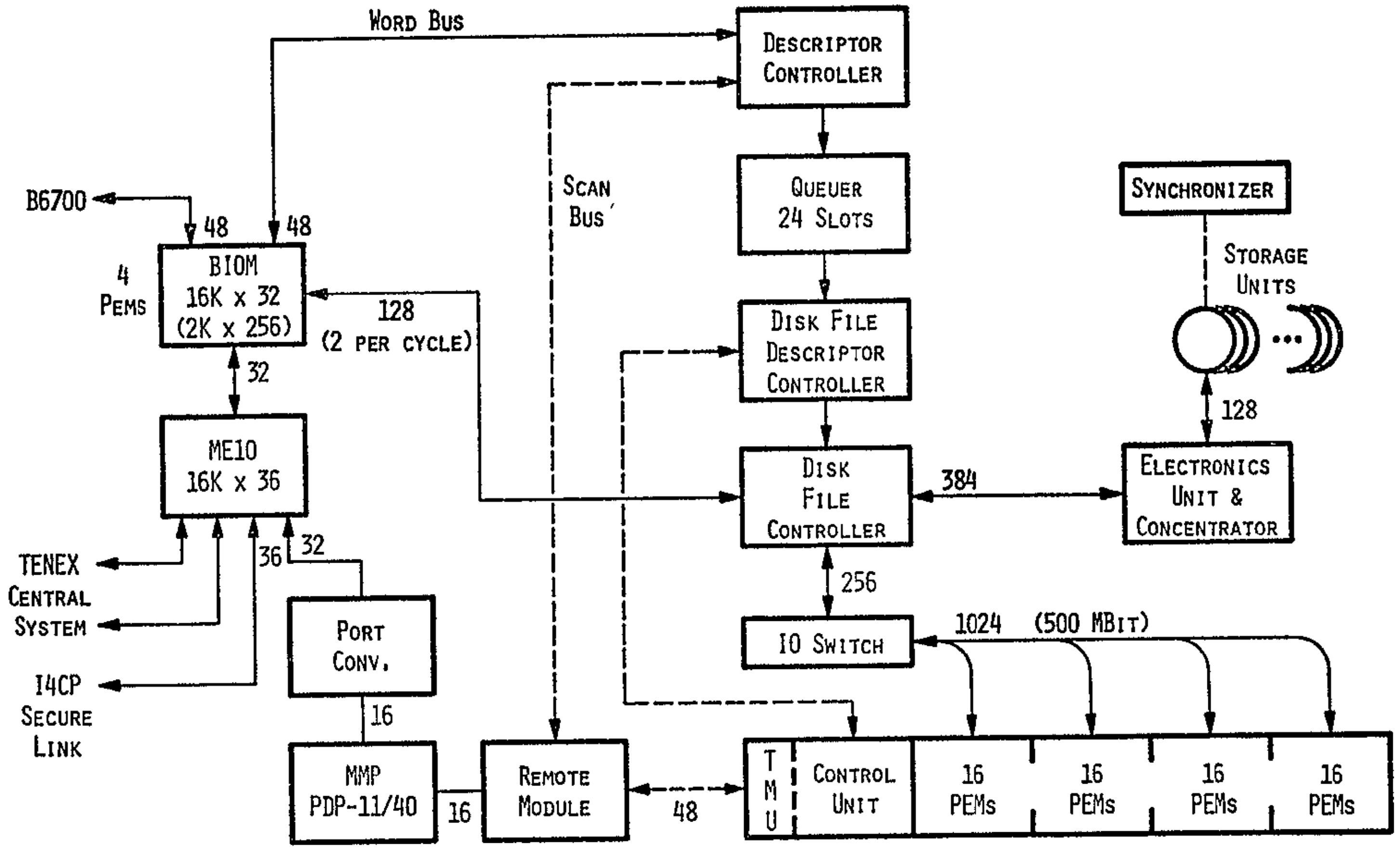


Figure IV.1.

Current Illiac IOSS

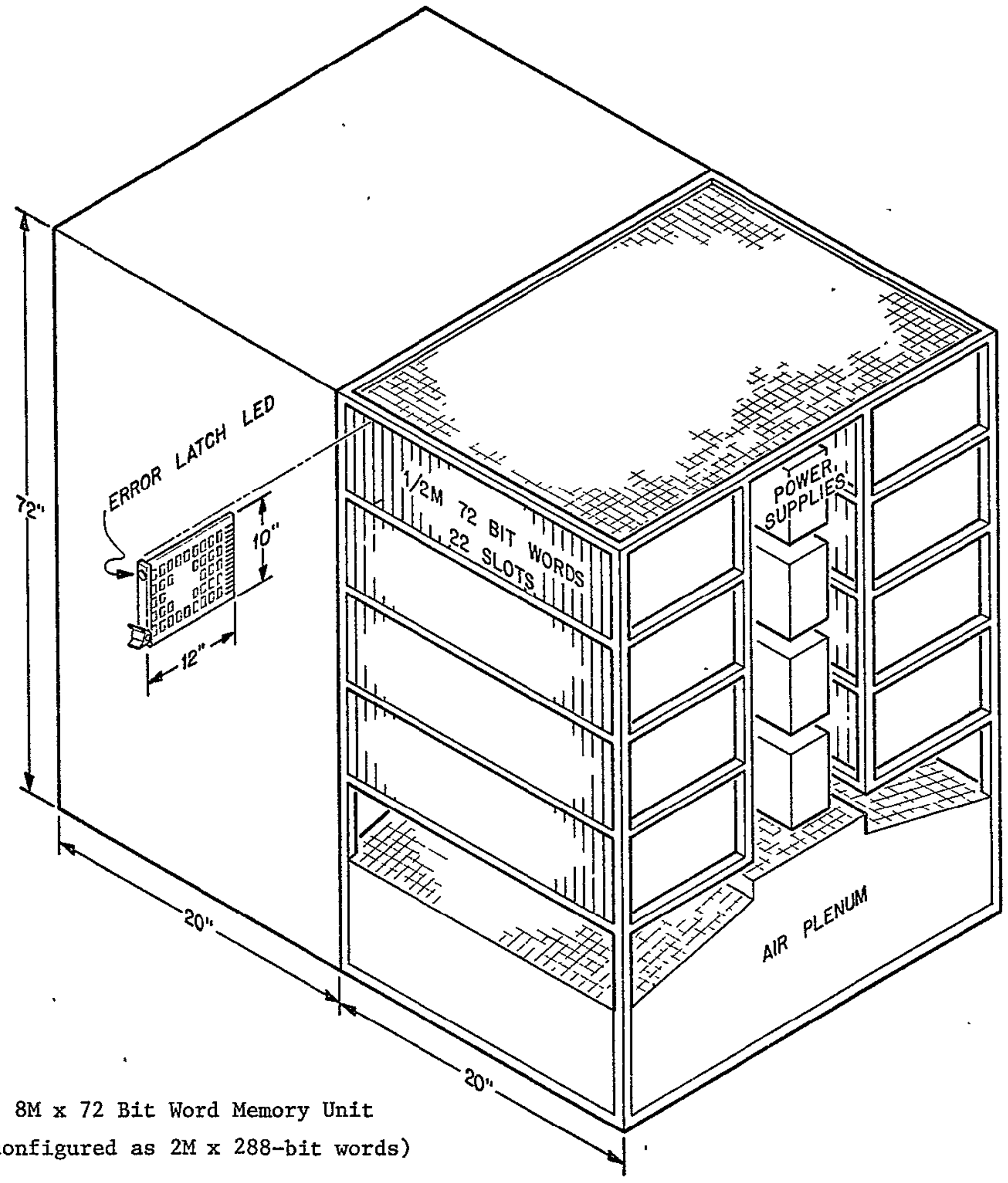
total of seventeen bits are available to address any sector (and therefore 128K sectors can be addressed). The current hardware does not utilize this address space since only 1200 rather than 2048 sectors are available per band. A relatively minor modification to the DFC should allow both formats so that the EU replacement may be expanded to 33.5 million words. This assumes that the current sector size of 128 words is maintained and that the already implemented DFC select bit is used to extend the number of addressable sectors to 256K.

A Fairchild CCD464 64K-bit CCD memory chip is currently available and will be used for the purpose of this discussion as a building block for a representative disk replacement system. This device is organized as 16 addressable 4096-bit loops. It can be read or written at a rate of from one to four megahertz; this could handle the current disk data rate of four megahertz per track although multiplexing may be required if the high clock rate cannot be achieved in the delivered system. The 4096-bit loop can be sub-divided into 31 sectors of 129 bits each (128 plus parity). Four chips of 16 addressable lines each would make one band of one bit width or 1984 sectors. Straight-forward software and hardware changes would be required to address more than 1200 sectors per band but the required eleven-bit registers are already in place.

Ideally a memory card which is in a system with single-bit error detection and double-bit error correction (SEC/DED) will have only one bit of a word on it so that if any chip on the card fails the system will continue to perform properly. In fact, the card (if designed properly) could be replaced while the system is operating without affecting system performance. Commercially available modules are not usually designed in this manner since it is desirable to be able to use memory boards to build small systems as well as large system. One such commercially viable module might be 64K by 36 bits and such a module will be used for subsequent discussion. To make a four megaword unit, 128 such cards are necessary along with the appropriate control logic and SEC/DED (see Figure IV.2). A cabinet 72 inches high by 40 inches wide by 22 inches deep would provide 3968K words of 128 bits each (including longitudinal points on every 128 words); this could be used as a unit in the extended memory. Thus each module would provide 128-bit words to either an EU replacement unit or to a Memory Unit Controller (MUC).

Internally each cabinet or Memory Unit (MU) contains 128 modules of 64K by 36 bit memories (individual circuit cards), each with its own chip and loop selection logic (see Figure IV.3). A selected loop is read continuously unless a write operation commences. All devices in a module are enabled so that they recirculate and refresh in place with the active loop.

A write into an MU would begin upon an address match. Data would be received on 128 bi-directional data lines which pass through a Hamming code and parity generator (SEC/DED code) into a data register of 128 data bits, 14 Hamming code bits and 2 parity bits (144 bits in all). Each 32-bit halfword in the data register would be routed to a separate memory module where it would be clocked into the active loop of the selected chip. Each succeeding word on the data lines would be similarly handled.



8M x 72 Bit Word Memory Unit
(configured as 2M x 288-bit words)

Figure IV.2.

2M Word Memory Unit Physical Layout

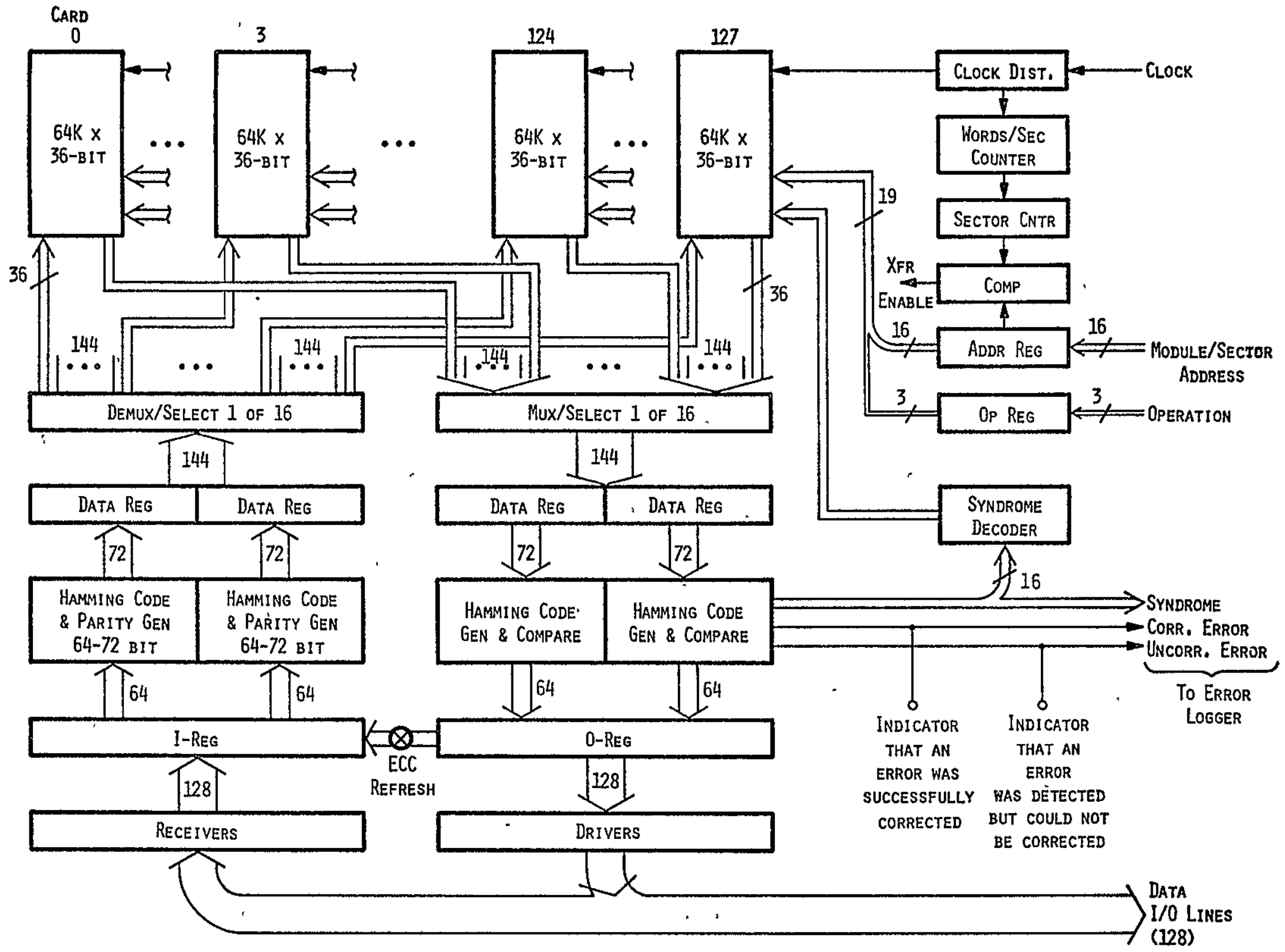


Figure IV.3.

2M 128-bit Word Memory Unit

A read from a memory module would, on an address match, send out 36 bits of data per clock into its respective position in a 144-bit register. This data is checked by a Hamming Code checker. When errors are corrected, the module in which the failed bit occurred is logged and an indicator on that module is activated to indicate to maintenance personnel that an error occurred on that module. A one-shot timer could be started so that the occurrance of a second error in the timer window would be treated as prima facie evidence of a chip malfunction. This would catch all hard errors and soft errors of high frequency. If SEC/DED fails, the uncorrectable error line is activated and the relevant information is sent to the MUC.

The Memory Unit Controller (MUC) is needed to provide the interface to the Disk File Controller (DFC), the Disk File Descriptor Controller (DFDC) and the MUs (see Figure IV.4). The MUs must provide a 128-bit data pathway of 512 megabits per second transfer rate. Any one of four MUs may be selected for a given I/O operation. Only one MU need be operational for the system to be working. A read operation would take successive three word groups from the selected MU and demultiplex them across the 384-bit data register to the DFC. A write operation would multiplex the three 128-bit words of the 384 bit register onto one 128-bit wide bus. This bus would provide a data pathway to a selected MU.

When the MUC is in read address mode (signal DRA from DFDC) the 12 bit sector address is selected instead of the data and this is presented on DFC data lines D000-D011, D016-D027, etc., up to D304-D316 to represent addresses on SU0 through SU15. Since a master clock in the MUC is used by all MUs the sector address will be identical for all emulated SUs. Since the CCD replacement for the SU has only 31 sectors worth of storage in a loop, there are two alternatives for accessing a sector. The first is that several loops must be stepped through to reach the desired sector. A second alternative is to take the information from the DFDC queuer 1) to establish the remainder of the sector address, 2) to select the appropriate loop or chip and 3) to present that address to the DFC. The latter course is chosen since the first course would entail latency equivalent to the current I4DM. Hardware changes are necessary to the DFDC querer to obtain this information, but they should not be difficult to accomplish. Chip, loop and MU selection may be developed from the DFC select bits, TS1 and TS2 band selection, and the querer, or perhaps all the information may be taken directly from the querer.

The cost for the initial MU module (including SEC/DED) will be about $350,000. The MUC will cost about $200,000. The total system of 32 million words would cost about $1.4 million but an initial 8 million word system, expandable to 32 million words, would cost only $550,000 and an initial 16 million word system would cost about $880,000.

The system reliability of a 16 million word memory can be approximated by calculating the probability of a double error in a given word of an MU. Since hard CCD failures are usually catastrophic (that is, a loop or an entire chip fails rather than a single bit fails), calculation will not be refined to consider partial device failures. Considering a combined hard and spurious failure rate of 1 per thousand hours per device (soft error

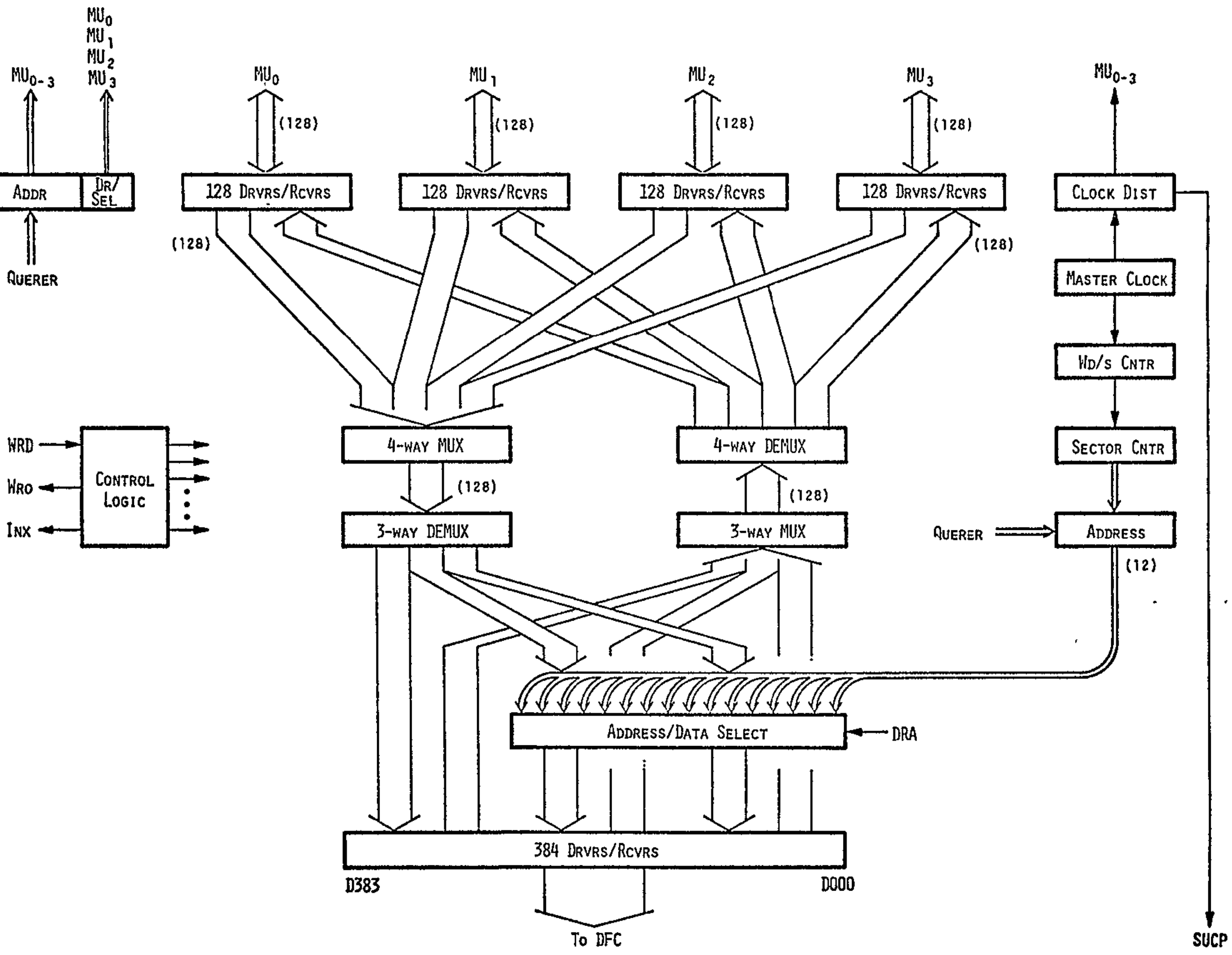


Figure IV.4.

EU Replacement Memory Controller

rates are expected to be extremely high), the mean time before failure of an MU without SEC/DED is approximately 0.1 hours.

SEC/DED applied in the normal fashion would not alleviate this problem. If, however, the entire memory is refreshed through the SEC/DED circuitry every few minutes, a different picture emerges. If an entire memory of sixteen million words is "scrubbed" every 10 minutes, the system reliability due to memory components only would be one catastrophic failure every 8,800 hours (see Table IV.1 for calculations).

These reliability calculations assume that 1) failure rates in chips per hours can be treated as the probabilities that a given chip will fail; 2) the system contains 16 million 72-bit words; 3) refresh is through SEC/DED for the entire memory at a rate of once every 10 minutes (refresh takes approximately one second); 4) the probability of a soft failure for a device is 0.1 percent per hour; 5) the probability of a hard failure for a device is 0.0001 percent per hour; 6) soft failures affect only one bit; 7) hard failures affect an entire device; and 8) hard failures are fixed every two hours.

If there are 25,000 non-memory components with a failure rate of 0.5x10**-6 hours the system mean time between failures due to non-memory components could be 80 hours. Obviously the non-memory components will effectively determine the system reliability, although it should be noted that some of these non-memory components will also be protected by the error correction circuitry.

The system impact of the SU-EU replacement approach would be minimal since this system would be a nearly direct replacement of the existing EU with its associated SUs. A minor modification would be necessary to the DFDC querer to get the address word and the Burroughs CTL cable device/receivers would have to be obtained to communicate with the DFC 384-bit data bus. No software changes would be required, although much of the MAP subsystem would become extraneous.

The risk of this approach depends on the availability of the CCD devices, the Burroughs cable drivers and the IAC resources available to design and construct these units. Although the logical complexity is not great, it is still a large amount of hardware. Since it is an extensible design, only one MU need be constructed at the start, thus lowering the overall risk.

There are seven tasks in implementing this CCD disk replacement. They can be grouped into those relating to the memory control unit and those relating to the memory units. Aside from interacting during the initial design phase, these two groups are independent until the final test stage. The first three tasks deal with the memory control unit, and must largely be done within the Institute since this unit is unique to the installation. Task one is the design of the MUC to replace the EU; this task is estimated to take eight months, but could with intensive effort be reduced to six. Task two is the procurement of the MUC boards and cabinet; the procurement cycle is four to nine months. The third task is the fabrication of the

Table IV.i.

MTBF Calculations for a 16M Word CCD Memory

Calculation of a double soft failure MTBF:

```
P(soft failure in word/hr)         = P(device failure/hr)/(words/device)
                                   = .001/(65536/72)
                                   = 1.1x10**-6 soft error prob./hr/word
P(word error/10 min) = P(word failure/hr)/6
                     = 1.83x10**-7
P(2 errors in a given word/10 min) = P(word error/10 min)**2
                                   = 3.36x10**-14
P(double soft failure in system/hr) = P(2 errors in word/10 min)x6x(#words)
                                    = 3.36x10**-14x6x16x(1024)
                                    = 3.38x10**-6
```

This calculation means that the mean time between soft errors is 1/(3.38x10**-5) hours, or about 296,000 hours.

Calculation of hard/soft failure MTBF: (Hard failure MTTR = 2 hours)

```
P(hard failure & soft failure/device/2 hr) = P(hard failure/2 hr) x
 Device-soft-failure-likelyhood
                                           = 2x10**-6x0.001x(12/2)
                                           = 1.2x10**-8
```

Note that the Device-Soft-failure-likelihood will be half the soft device failure rate for 10 minutes since the error mean life will be 5 minutes.

```
P(hard/soft system failure/2 hr) = P(hard & soft/device/2 hr)x(#devices)
                                 = 1.2x10**-8x18432
                                 = 2.2x10**-4 failures/2 hours
```

This calculation indicates that the mean time between hard/soft error combination is 1/(3.1x10**-6) hours, or about 9000 hours.

Calculation of hard/hard failure MTBF:

```
P(hard/hard device failure/2 hr) = P(hard device failure/2 hr)**2
                                 = (2x10**-6)**2 = 4x10**-12
P(hard/hard System failure/2 hr) = P(hard/hard device failure/2 hr) x
 (#devices)
                                 = 4x10**-12x18432
                                 = 7.4x10**-8 failures/2 hours
```

This calculation indicates that the mean time between catastrophic hard errors is 2/(7.4x10**-8) hours, or about 27 million hours.

System MTBF considering only the memory components would then be:

```
MTBF(system) = 1/(P(soft/soft) + P(hard/soft) + P(hard/hard))
             = 1/(3.38x10**-6 + 2.2x10**-4/2 + 7.4x10**-8/2)
             = 8,800 hours
```

MUC; one should allow six months for delivery. Thus the MUC could be in place ready for acceptance testing in eighteen to twenty-three months.

The tasks involved with the memory units are similar, but their durations are different. Task four is the specification of the MU; this should take two months. Task five is the procurement of the MUs; again this is four to nine months. The sixth task is the fabrication of the MUs; at the present time, manufacturers estimate this requires six months, including their engineering time for the cabinets and backplanes to meet our specifications and to include error correction. Subsequent acquisitions of MUs should therefore take less time (from two to four months). Thus the MUs could be delivered in one year to eighteen months.

The last task involves testing and integrating the system. One should allow four months for testing the MUC with one MU, three months for adding the second MU, and two months for subsequently acquired MUs.

IV.B.2.a.ii. Complete IOSS replacement

The prior approach involved replacing only the SUs and the EU with a CCD memory system. The objective was to realize some of the benefits of CCD memory while minimizing the impact on current hardware and software. The full potential benefit of CCD would be masked by the intervening hardware. In particular, the Memory Management Processor (MMP) has a response time of 2.4 milliseconds which must be added to the CCD latency for each request to calculate the total latency. A re-examination of the entire IOSS with a complete CCD memory system replacement as a design objective will now be described. The purpose of this description is to demonstrate the feasibility of such an approach and to indicate the specifications to be used in acquiring such a memory from a vendor.

The current IOSS (see Figure IV.1 again) contains the following components: 1) the IO Switch (IOS), 2) the Disk File Controller (DFC), 3) an Electronic Unit (EU), 4) Several Storage Units (SUs), 5) Synchronizer, 6) a Disk File Descriptor Controller (DFDC), 7) a Queuer, 8) a Descriptor Controller (DC), 9) a Remote Module (RM), and 10) the MMP. Currently a program on the Illiac will make a disk request through the Test and Maintenance Unit (TMU) of the Illiac Control Unit (CU). This request will be passed on through the RM to the software queue in the MMP. These user commands are then broken down into hardware subcommands which are passed through the RM to the DC and finally to the Queuer. Commands remain in the Queuer until a matching sector address occurs. At that time, they are executed by the DFDC and DFC. Results (either acknowledgement of successful transfers or of error conditions) are sent back to the MMP and stored in the result queue. From there they are passed through the RM to the TMU and thence to the user program.

A CCD memory system could attach to the system at the same point as the current IOS. By placing a switch at this point, either the current IOSS or the CCD memory could be accessed. This would reduce hardware impact and allow a phased incorporation into the system without an extended shut down. The envisioned switch would be a simple redesign of the current cabinet I/O logic (see Figure IV.5). These units could connect to a global multiplexor/demultiplexor which would handle from four to sixteen 4 million word CCD memory units of the type described above in the simple EU-SU replacement approach. Alternatively, the cabinet I/O could directly connect to four 16 million word CCD memory units of the type described above in the simple EU-SU replacement approach. Alternatively, the cabinet I/O could directly connect to four 16 million word CCD memory units that are only one-fourth loaded (see Figure IV.6.). The disadvantage of the latter approach is that the cabinetry and back-plane wiring costs for the entire system may have to be born by the first phase.

Control of the memory would be exerted through the TMU's TRO and TRI registers in somewhat the same manner as control is currently exerted over the I4DM. A request will be sent out TRO and a reply will be received in TRI. Several simplifying characteristics will make the CCD control module much easier to implement. First, only one request will be handled at a time, so no request or result queues are necessary. This is possible since the reduced latency of CCD reduces the need for this feature. Second, pages of the CCD memory will be directly addressed with no remapping or run time symbolic reference. The reason for this is that with SEC/DED, it is expected that the entire memory will be available for user programs, much as it is assumed that all 64 processors are available; the CCD memory will be easy to repair (this is in contrast with the current disk memory, which requires the ability to be remapped in order to have sufficient availability); and third, the entire CCD memory system will be clocked uniformly with the master clock and synchronized with the CU clock.

Focusing first on the Global MUX/DEMUX Selector (GMDS) (Figure IV.7) and using the previously defined MU, this unit is simply a 4-1-4 selector-multiplexor demultiplexor (from four buses, select one and distribute successive words over four buses). Other alternatives include a 1-1-4 or 2-1-4 selector-multiplexor/demultiplexor for a smaller 4 and 8 MU configuration respectively. Several modes of operation are possible. An initial configuration of two MUs would transfer data in the form of sixteen 256-bit words; these would be multiplexed/demultiplexed four ways to make four 1024-bit words or sixteen groups of four 64-bit words. These sixteen groups of four words will in turn be multiplexed/demultiplexed by four in the cabinet electronics to transfer sixty-four 64-bit words, one to each PE. The transfer rate for this configuration could be one gigahertz assuming a four megahertz system clock.

A four MU configuration could operate by selecting a group of two MUs at a time for a given Illiac page transfer -- this produces a one gigahertz bandwidth. If a higher bandwidth were desired, each set of two could be alternatively selected for each 256-bit word transferred, thereby giving a bandwidth of two gigahertz. An eight MU configuration could attain four gigahertz by multiplexing four groups of two MUs together.

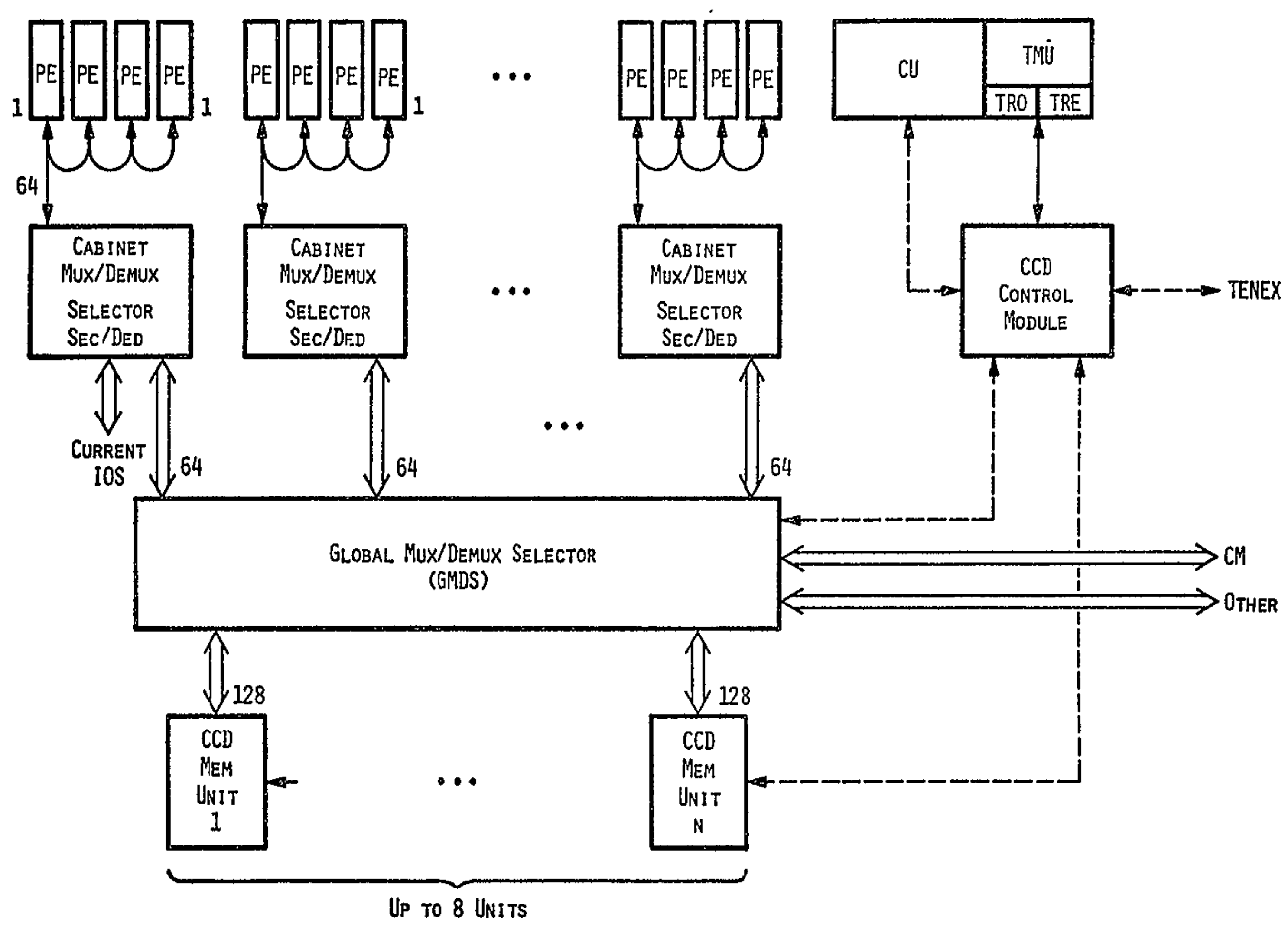


Figure IV.5.

CCD Memory System Block Diagram

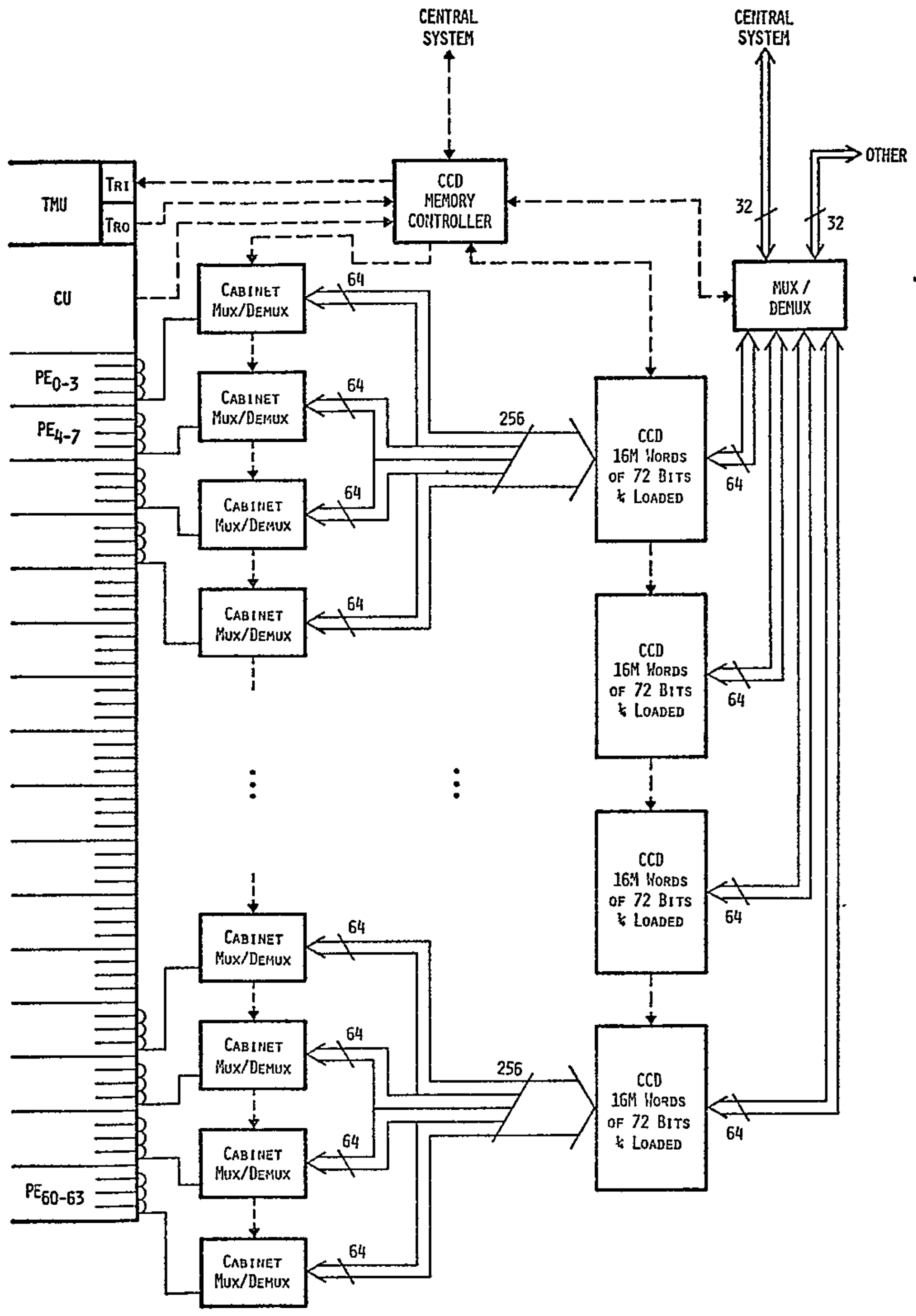


Figure IV.6.

CCD Memory System Using Quarter Loaded Modules

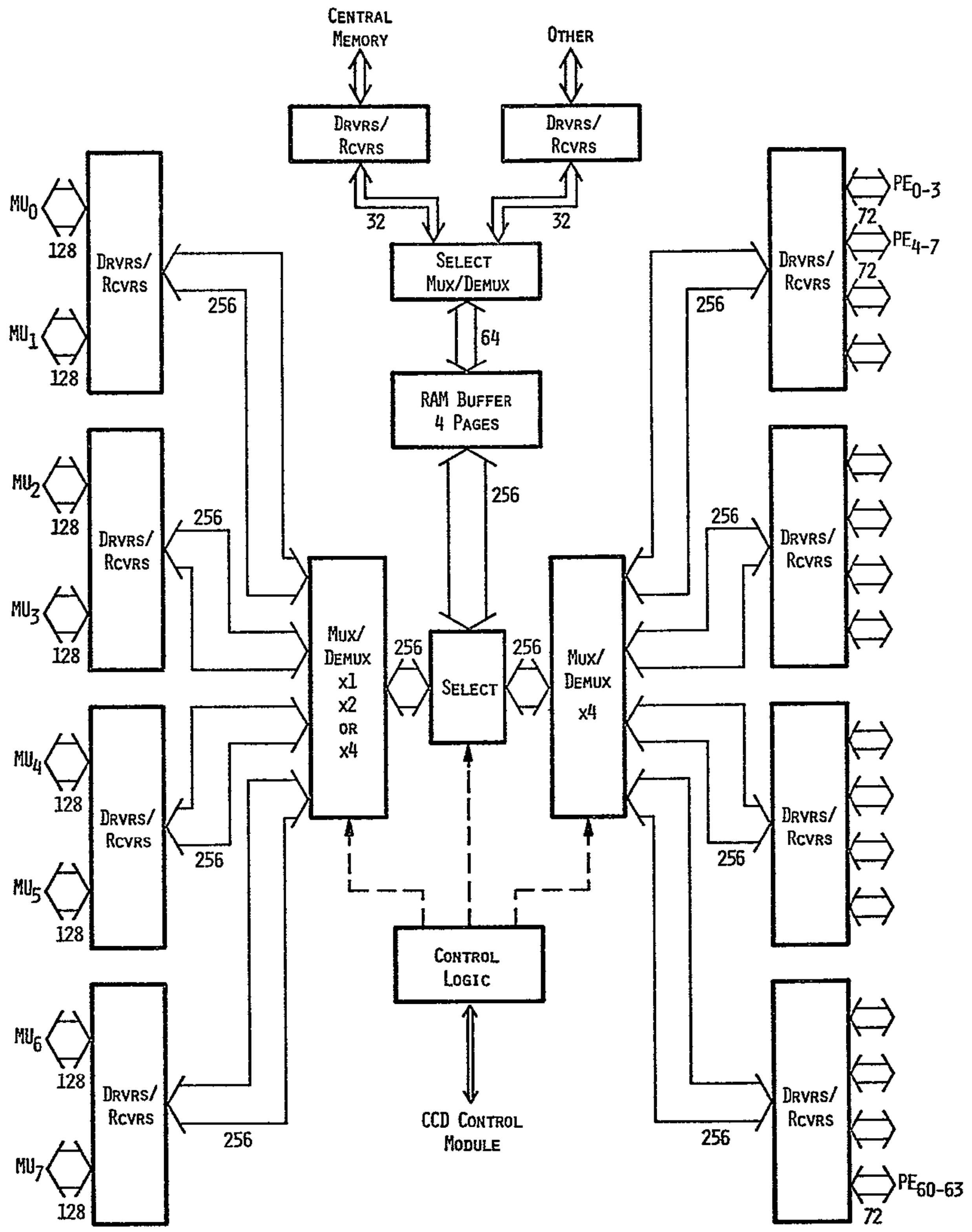


Figure IV.7.

Global Mux/Demux Selector

The GMDS unit must also transfer data between the support system and the CCD memory for loading and unloading Illiac data and for general file transferring. A 256x1024 RAM buffer (also organized as 64x4096) is used to buffer transfers to the slower central memory (or to other possible devices such as a staging disk memory system or a high speed tape system).

Control of the GMDS unit and the CCD memory modules would reside in one module, the CCD control module (CCM) (see Figure IV.8). This module would handle only one CCD to PEM request at a time from the CU. This would necessitate a rudimentary I/O executive on the Illiac if request queueing were desired. Without request queueing, a second request issued before the first request is completed would cause the CU to pause until the completion signal of the first request is received. CCM will also handle file movement requests originating from both the CU and the control system. The file movement requests, unlike the CCD to PEM requests, would be queued in the CCM and acted on as a programmed process (in contrast to the "hardwired" CCD to PEM transfers). A CCD to PEM transfer would interrupt a file transfer. On completion of the CCD to RAM transfer, the file transfer would resume until interrupted again or until the file transfer was accomplished.

If the approach of eliminating the GMDS is elected, then a means must be provided for 1) a total of 1024 data lines (64x16) with only sixteen million words in place and 2) a simple method of adding more memory (up to a total configuration of sixty-four million words). Several methods can be imagined: for example using the 1024 data lines as a bus or partially loading memory boards. Both of these extremes are probably unsatisfactory. The first limits the maximum bandwidth and the latter introduces production hazards. A reasonable approach would be to have the cabinet capacity for sixty-six million words but have the units designed initially to accommodate 16M words with empty card slots for the remaining fifty million words. Alternatively, one could design the first phase to use the memory cards as they will be used in the second phase even though the cabinetry may be inappropriate for the second phase. Thus, the memory boards could be used in the second phase but a different back-plane organization, and hence a new cabinet, would be required. This would increase the cost of phase two by about 100 thousand dollars.

The direct approach shown in Figure IV.6 uses four 16M word modules which are only one-fourth loaded. If the cards are assumed to have 64Kx36 bit capacity then 512 cards are required initially, incidently providing 1024 bit parallel data lines . Adding the remaining 1536 cards will boost the capacity to the system to sixty-six million words. This approach could require the cabinetry, back-plane and control logic for the full 64M word configuration. This would be a sizeable part of the total expenditure for 64M words. Expansion would require only the addition of the memory cards and power supplies. Provision should be made for multiplexing the additional cards so that 4-way multiplexing may be achieved for a bandwidth of eight gigahertz or more. Another port will have to be provided which multiplexes the 256 lines down to 32 for file transferring to the external system.

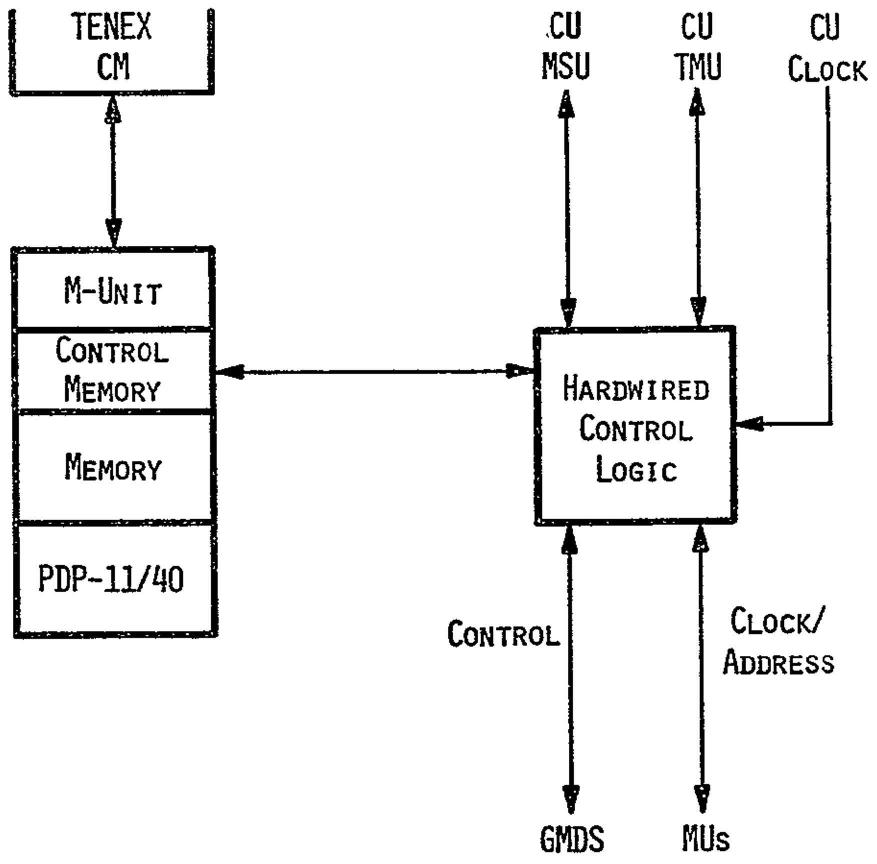


Figure IV.8.

CCD Control Module

The control of this system would be similar to the prior CCD Control Module (CCM) except that data pathway switching is now performed in the memory modules themselves instead of in the GMDS. CCM control lines that formerly went to the GMDS will now be broadcast to the memory modules and to the MUX/DEMUX unit which ports to either the control memory or to other devices like I4TAPE or a disk staging area. The MUX/DEMUX unit will have to have a few pages of memory for buffer due to the slower data rate outside the Illiac system.

If the entire memory and interface are supplied by a vendor, the system could be delivered about a year after the award of the contract. There is the possibility that the interface and a memory module could be delivered earlier to begin integration into the system and acceptance testing. One should note that the second approach connects directly to the I/O switch, so that hardware integration requires only switching cables between the exsting disk memory and this replacement. Thus system integration should have negligible impact on the availability of the Illiac, with acceptance testing being performed during scheduled maintenance times.

The cost of this CCD extended memory would be as follows. For an initial sixteen million word memory, plus sufficient cabinetry to house an eventual sixty-six million word memory, the cost is about $880,000. The interface would be between $50,000 and $100,000. During the 1980-81 time frame, additional increments of sixteen million words of CCD memory, plus power supplies, would cost about $490,000 each. Thus an initial sixteen million word extended memory using this approach would cost about $980,000, and the complementing fifty million words could be acquired for one and a half million dollars.

IV.B.2.a.iii. Discussion

As a disk replacement, a CCD memory significantly reduces the major difficulty users encounter in programming the Illiac: latency in accessing this memory is more than a factor of twenty less than for a disk based memory (the exact factor depends upon the rate at which the memory is clocked). Thus it is expected that most users would be able to take a casual approach toward arranging data in this extended memory and still not incur major performance degradation. But the latency will still be there; although this memory will support one-row transfers, users will still transfer data in blocks. The lower latency will permit smaller block sizes than are now efficient to use, and it is expected that the greater latitude in selection of block sizes will reduce the criticality of this issue in program design. Memory system divisions of two large semiconductor manufacturers are currently developing CCD memory systems and could supply memory modules suitable for use in the Illiac memory system. This minimizes the amount of internal development on the part of Institute in replacing the I4DM. The second approach, where the vendor supplies the interface as well, would require no development within the Institute. Using the memory boards being developed for other products also allows this project to benefit from whatever experience the vendors have already acquired with these devices (which in this case will be minimal, based almost entirely upon mock-ups or prototypes due to the newness of this level of integration).

The price of this memory is very attractive and makes a sixty-six million word memory economically feasible in the 1980 time frame. This is the most compelling advantage in considering this technology as a replacement for the I4DM.

Although CCD is an established technology, the level of integration in the memory system which is considered here is new and untried. One manufacturer's experience to date indicates that the transient error rate in this type of memory may be substantial, and that periodic scrubbing of the memory may be required (much as memory refresh is required of dynamic memories). The impact on system performance will be negligible, and as the frequency of scrubbing can be varied even with the system in place, it can be adjusted depending upon whether the actual system error rate is greater or less than that observed in current prototypes. In any event, this approach is better than relying on device re-design or packaging innovations to improve the situation.

IV.B.2.b. RAM Based Extended Memory

The design goals of the extended memory proposed in this sub-section are to decrease proogramming overhead and to increase the availability of the Illiac resource. A RAM implementation of the extended memory allows taking the former goal to its ultimate conclusion. The result is an extended memory which appears to have the major characteristics of the local memory: PE indexing, individual memory enables, and essentially no latency in accessing. The use of error correction circuitry and the ease of repairing semiconductor memories accomplishes the second goal.

IV.B.2.b.i. System Characteristics

To the user, each PE will have an associated 256K word extended memory module. The communication between the PEM and this memory (called the XMEM) is accomplished via a new instruction: MOVE. This new instruction is a two address instruction (the first two address instruction in the Illiac), with both addresses capable of being indexed by the control unit and by the local PEs. The simplest MOVE instruction from the point of view of program design would be one which moves data as fast as the PEM will allow, thus eliminating 1) the need to overlap computing with data movement, 2) the need to post the completion of the transfer for program inspection and 3) the need to buffer data. For these reasons the proposed memory has a very high bandwidth. This high bandwidth incurs extra cost in the implementation, but the expected savings in terms of hardware design (overlap and posting) and software design (posting and buffering) are expected to offset the extra hardware cost.

The MOVE instruction (see Figure IV.9) has nine fields. This instruction is an ADVAST instruction and hence bit 0 is zero; bit 19 is the instruction parity bit and bit 18 is the GLOBAL bit which is unused. Bits 1-4 and 20-23 contain the operation code for this instruction. Two fields in the instruction, bits 8-15 and 24-31, indicate multiplexing order. Each field is divided into four 2-bit fields, one field for each cycle of multiplexing on the memory bus. The first field is associated with the source and the second field with the destination. The purpose of this option is to allow a user to specify a restricted amount of re-ordering of data as it moves between local memory and an associated part of this extended memory (this will be explained with the implementation description below). This mechanism is a generalization of the current capability to transfer data from I4DM to a quarter or a half of the current PEM, instead of across all the PEM (admittedly, this current capability has yet to find significant application in user codes). Bit 5 of the instruction indicates ADVAST indexing and multiple row moves (see Figure IV.10). Bits 6-7 are unused in this case and the ACAR designators are taken from the descriptor syllables as needed. If either of the index operations reaches completion when moving multiple rows, the whole operation is stopped and the instruction is ended. The last row transfer performed will be the one where the index field of the accumulator is equal to its limit field. Bits 16-17 designate the accumulator which contains the transfer descriptor.

The source and destination transfer descriptors are 32-bit descriptors which reside in the same accumulator as indicated in bits 16-17 of the MOVE instruction; their formats are identical. Bit 0 of each syllable indicates whether the PEM or XMEM is described (0 for PEM and 1 for XMEM). Bits 1-2 indicate PE mode bit protection. If the destination is PEM the transfer is not made when the protection indicator is 1 and the individual PE mode bit is zero. If the destination is XMEM and the machine is in 32 bit mode, the transfer is similarly protected with some loss in bandwidth as fetching and restoring are required. In 64-bit mode the protection is supplied if either function is true. Bit 3 indicates that the PE X register is used for indexing and bit 4 indicates that the S register is used. Both cannot be set. Bit 5 is not used. Bits 6-7 of each syllable indicate the index register to be added to the displacement if the instruction calls for indexing. This field caannot indicate the descriptor register (i.e. point to itself). The format of the index register is the standard ACAR index format.

The index word format has three fields. The first field, bits 1-15, is the magnitude of the increment; bit 1 is the sign bit. Bits 16-39 is the limit, and bits 40-63 hold the current index.

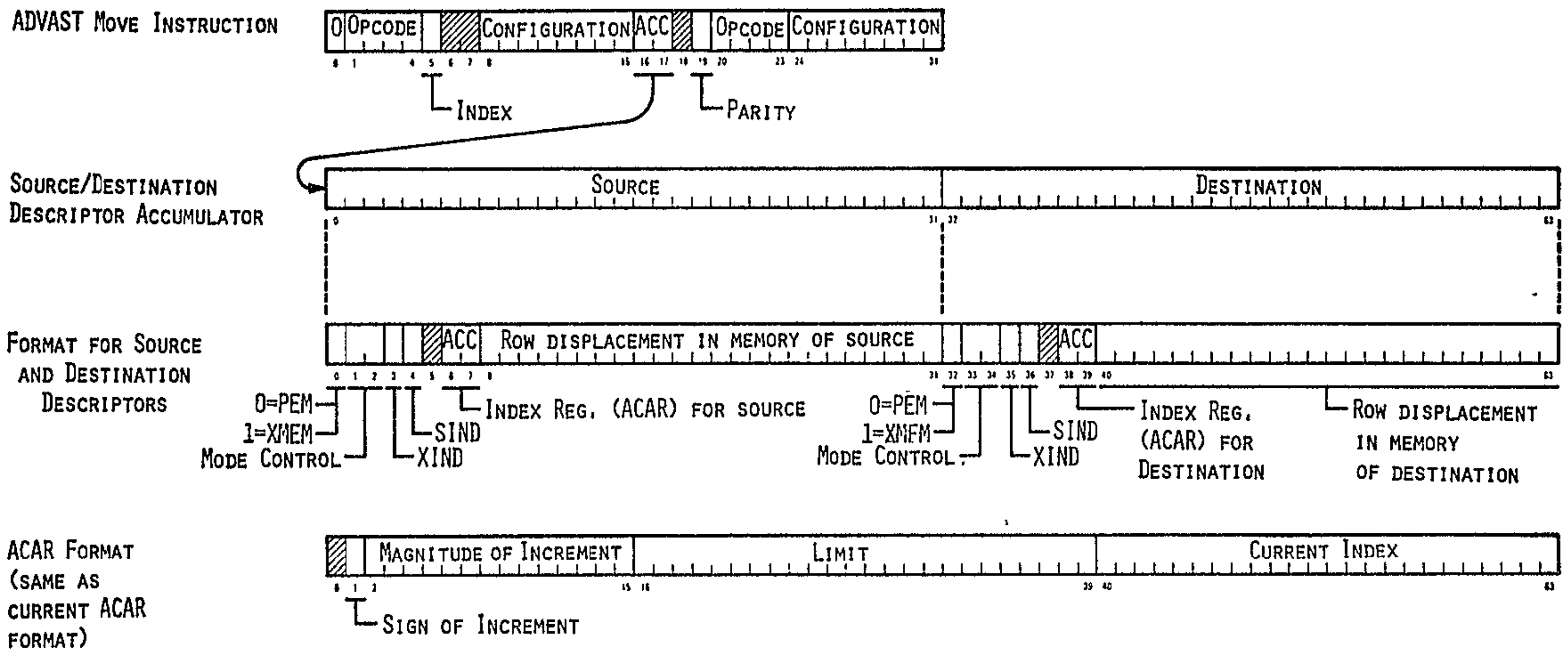

ADVAST Move Instruction
0
Opcode
Configuration
ACC
Opcode
Configuration
Index
Parity
Source/Destination Descriptor Accumulator
Source
Destination
Format for Source and Destination Descriptors
ACC
Row displacement in memory of source
ACC
0=PEM
1=XMEM
Mode Control
SIND
XIND
Index Reg. (ACAR) for source
Index Reg. (ACAR) for Destination
Row displacement in memory of destination
ACAR Format (same as current ACAR format)
Magnitude of Increment
Limit
Current Index
Sign of Increment

Figure IV.9.

Format Specification for Move Instruction

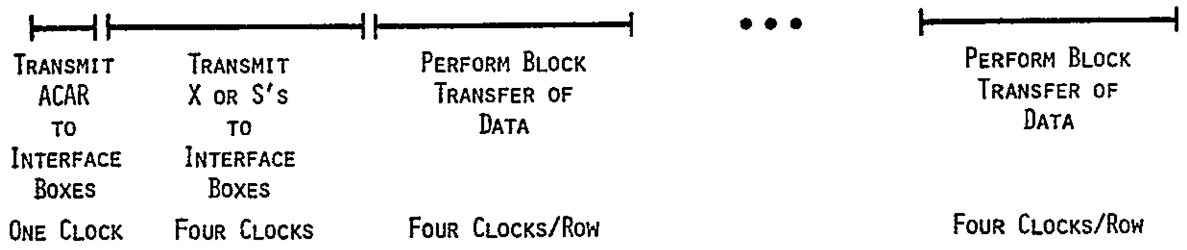

TRANSMIT ACAR TO INTERFACE BOXES
ONE CLOCK
TRANSMIT X OR S'S TO INTERFACE BOXES
FOUR CLOCKS
PERFORM BLOCK TRANSFER OF DATA
FOUR CLOCKS/ROW
• • •
PERFORM BLOCK TRANSFER OF DATA
FOUR CLOCKS/ROW

Figure IV.10.

Move Instruction Timing

There are several exceptional conditions for this instruction. An undefined instruction is one where the instruction and/or its descriptor is somehow invalid. A non-existant memory error occurs when the PE generates an address to the XMEM which does not exist. An uncorrectable data or address error occurs when the error checking circuit for the XMEM detects and uncorrectable error. These errors are catastrophic and cause prgram termination. A correctable data error occurs when the error checking circuitry detects a single bit data error and corrects it. The error is logged for system maintenance purposes, but the program is not notified that error correction has been automatically performed; thus the program continues processing with the correct data.

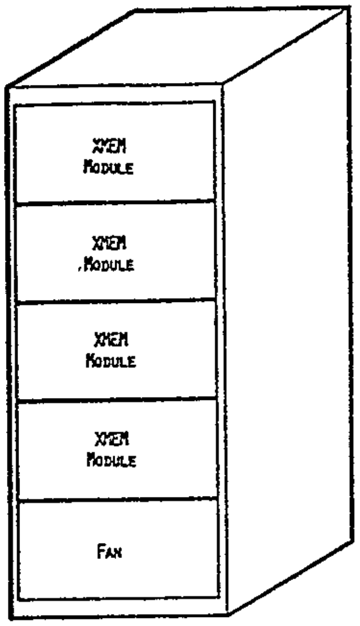
Other instructions could be modified to use XMEM, and it is possible that the program could reside in XMEM as well. The current Illiac instructions LOAD, BIN and STORE could access XMEM if bits 8-15 of these instructions are non-zero. The value of this field would indicate the segment referenced (here segment would be PEM or XMEM). A segment register in ADVAST exists to indicate to the Instruction Look Ahead mechanism (ILA) the appropriate segment of memory from which to get the next program instructions. ADVAST references to this register which change its value cause ILA to clear the instruction buffer when the next JUMP instruction is encountered and take a new instruction from the new segment. In addition, if an interrupt is encountered, the program switches to the interrupt segment and could on subsequent interrupts switch back to the original program segment. Thus by exploiting the current ILA mechanism, it should be possible to allow the program to reside in XMEM.

IV.B.2.b.ii. Implementation

The primary purposes of this extended memory replacement are to enhance the memory system of the Illiac with respect to size, reliability and speed. In view of complaints about the limited size of the local memory, the uncertain availability of the machine and the difficulty of using the current I4DM, the use of high speed, high density, low cost semiconductor RAM for an extended memory replacement is especially attractive. Complaints about the limited size of PEM are directly related to the drawbacks of the I4DM. Therefore an adequate solution to the problems of using the I4DM should also assuage the difficulties of using the PEM.

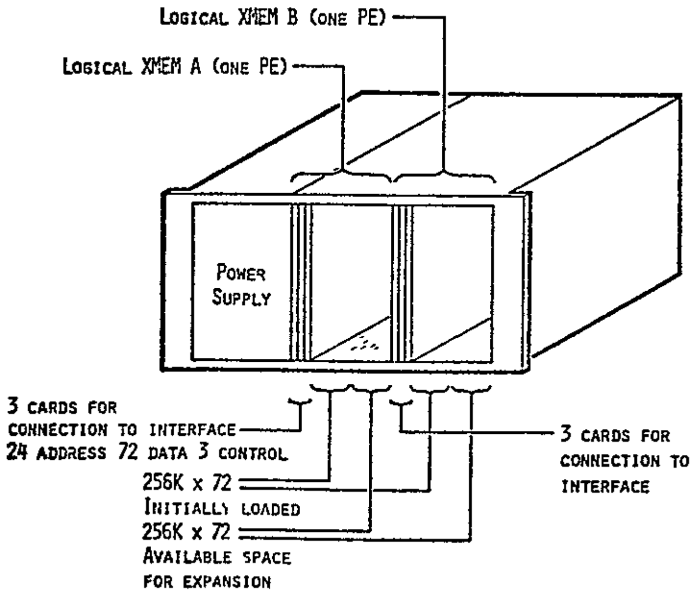
This extended memory design includes four major sections: the external memory hardware, the interface between the Illiac and this external memory, the Illiac modifications required to access this type of external memory and the interface to the support system.

The external memory hardware consists of a set of memory modules organized in 72-bit words (see figure IV.11). These modules use 16K semiconductor RAM chips and have a bandwidth of from two to eight million words per



XMEM Cabinet
8 Required

EACH MODULE WILL INTERFACE TO TWO PE'S
STANDARD 6' HIGH 24" MOUNT CABINET
30" DEEP



XMEM Module
32 Required

Figure IV.11.

RAM Memory Unit Physical Layout

second; these memories will be interleaved sufficiently to support as much as eight million words per second transfer rate per PE.

As shown in Figure IV.12 each PE has a complement of XMEM. In order to use the existing cables, the memories will be cabled together in groups of four, as the PE's are presently connected to the I4DM, and the cable will be time multiplexed for the transmission of address and the transmission and reception of data (see Figure IV.13). The MOVE instruction has two eight bit fields to indicate the multiplexing order. Through suitable encoding, any of the 256 possible associations of four PE's to their four XMEM modules can be accomplished; the simplest, where each PE is associated with its own memory module, is denoted by all zeros in the two fields. The connection between the interface section and the memory modules is simple: 72 data lines, 24 address lines and three control lines (for read, write and refresh). All error detection and correction is done in the interface hardware and refresh is controlled by dedicated logic under CU control so that synchronization is possible.

A set of high speed bipolar logic is used for interfacing the external memory hardware with the processing unit. This logic will enable the Illiac to perform address calculation, error checking, and extended memory refresh without extensive modifications to the processing units and without the need for extensive special purpose hardware in the vendor supplied memory modules. The primary functions of the interface (see Figure IV.14) are to compute addresses during block transfers and to provide error correction. The address calculation requires copies of the components required to perform the calculations, that is, the CU index field (base address and ACAR index field), the increment field and a copy of the X register or the S register for each of the PEs. By making the MOVE instruction sufficiently fast, overlapping of data movement and computing will not be necessary; thus greatly simplifying the design and eliminating potential timing problems. The error correction circuit is a normal SEC/DED circuit which includes a logger for errors. The logger contents can be read into the PE and will probably be included in the Illiac dump file on the support system.

The modifications to the Illiac itself are primarily to the control unit for the purpose of allowing the processor to reference and maintain the extended memory. These center around adding the MOVE instruction, which requires modifications primarily to the ADVAST section of the control unit. Both FINST and ADVAST must have procedures to implement the new MOVE instruction. In ADVAST the ACAR incrementing must be modified to increment both the source and destination. FINST must be able to control two memories a row at a time. Both modifications will require new states for the two units. In addition, storage for the segment address bits must be added since current CU addressing is by syllable. This is easily accomplished: a CU register is currently appended to a CU address to form the memory address for the LOAD, BIN and STORE instructions; since this register can be manipulated under program control, the segment address can be handled in this fashion.

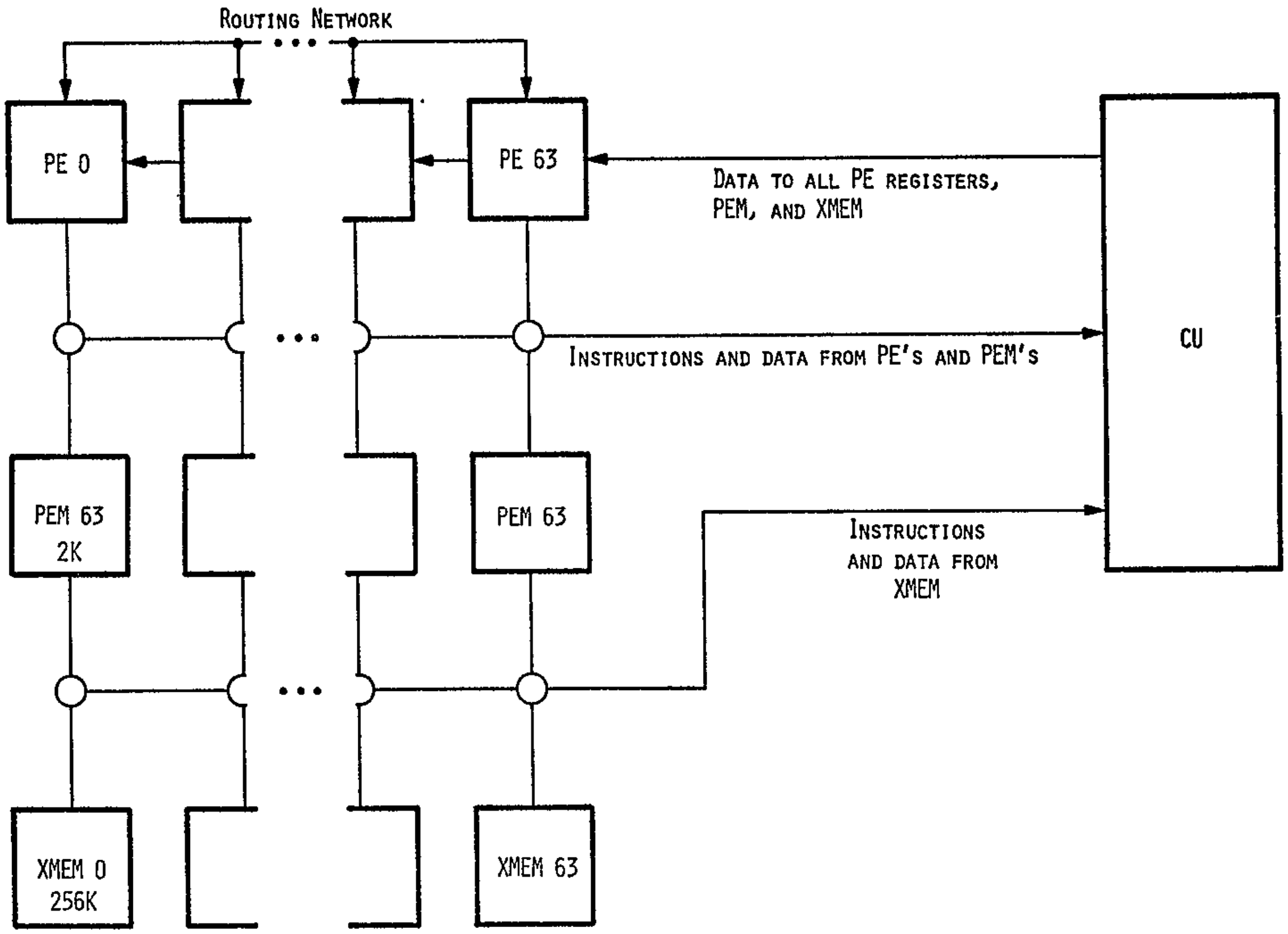

Routing Network
PE 0
PE 63
Data to all PE registers, PEM, and XMEM
CU
Instructions and data from PE's and PEM's
PEM 63
2K
PEM 63
Instructions and data from XMEM
XMEM 0
256K
XMEM 63

Figure IV.12.

System Block Diagram for RAM Extended Memory

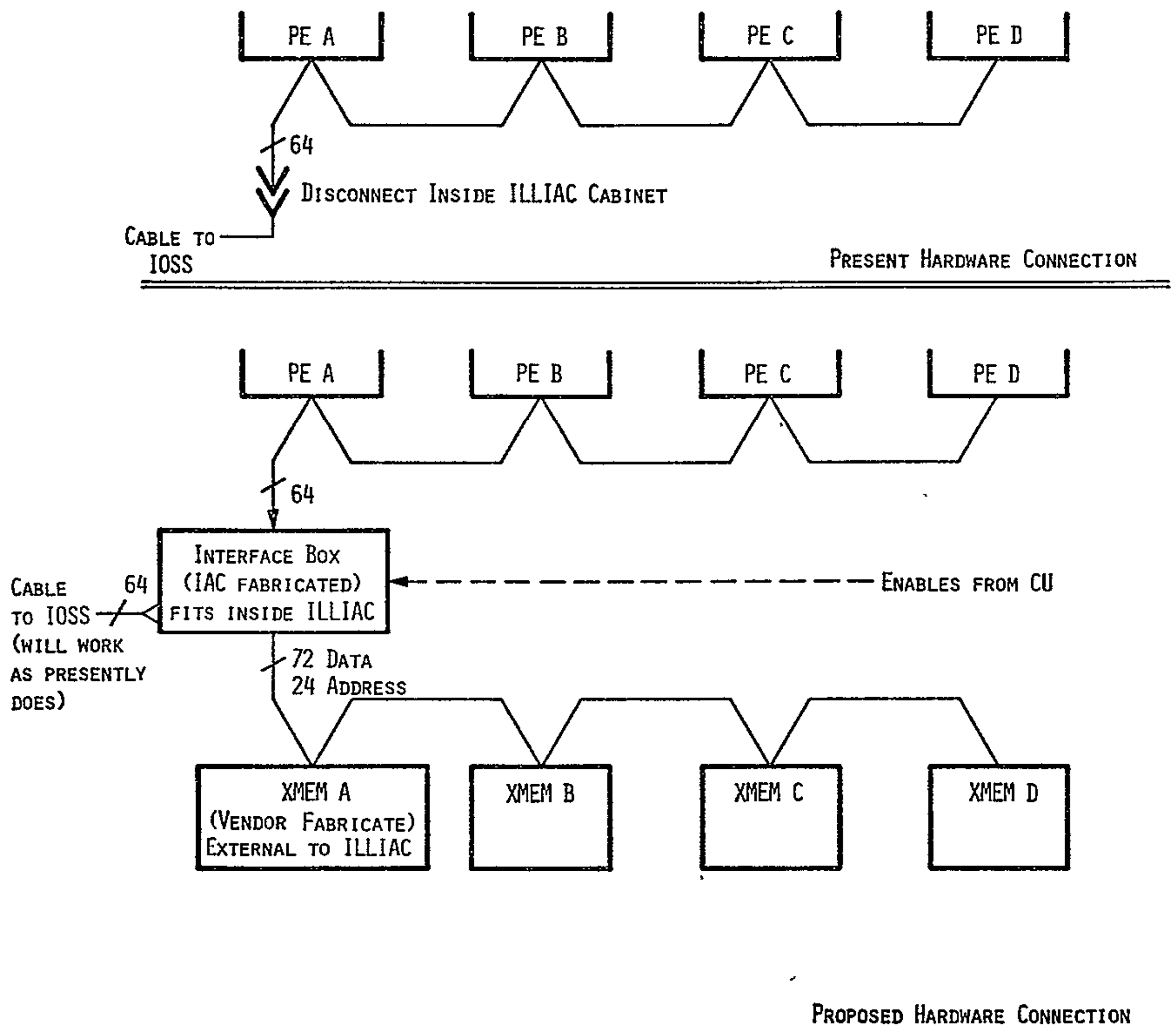

PE A
PE B
PE C
PE D
64
DISCONNECT INSIDE ILLIAC CABINET
CABLE TO IOSS
PRESENT HARDWARE CONNECTION
PE A
PE B
PE C
PE D
64
INTERFACE BOX (IAC FABRICATED) FITS INSIDE ILLIAC
CABLE TO IOSS (WILL WORK AS PRESENTLY DOES)
64
ENABLES FROM CU
72 DATA
24 ADDRESS
XMEM A (VENDOR FABRICATE) EXTERNAL TO ILLIAC
XMEM B
XMEM C
XMEM D
PROPOSED HARDWARE CONNECTION

Figure IV.13.

Connection of RAM Extended Memory to Illiac

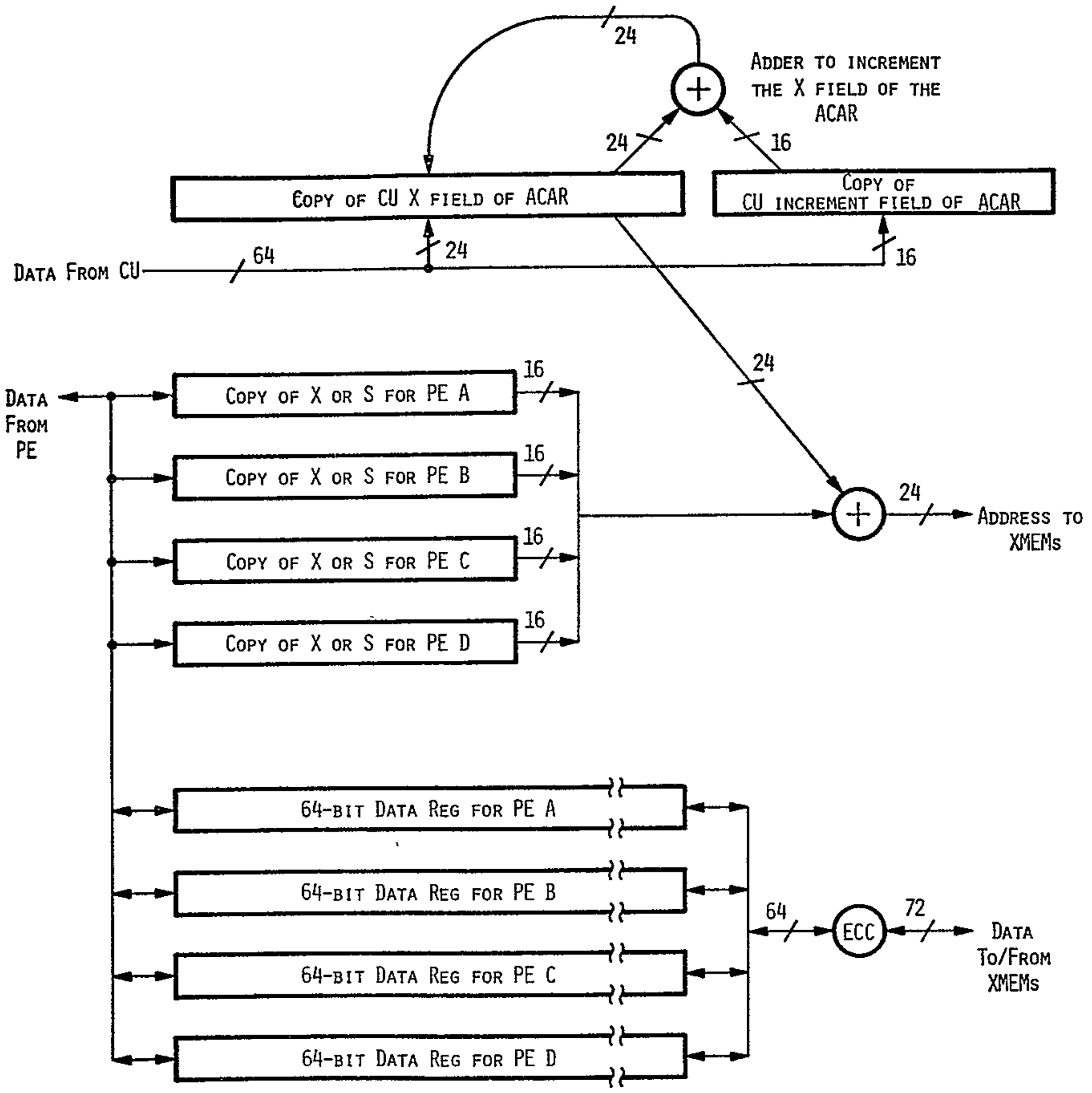


Figure IV.14.

Interface Block Diagram

Interfacing to the support system requires a set of modifications to the Illiac and support system to enhance the useability of the extended memory by allowing high speed transfers between the extended memory and the Illiac file system. Currently the MMP can access all Illiac memory through the TMU - Direct Memory Access (DMA) port into the Illiac memories, registers, ADB and PEM. By manufacturing a new interface to include a DMA port into the support system memory and allowing transfers between a support system memory and an Illiac memory, efficient data transfers can be accomplished. This interface exploits features of the Illiac (TMU's DMA transfers) in order to simplify both its own design and that of the XMEM; the TMU also provides synchronization with the Illiac function, thus further simplifying the design of this interface. Some modifications may be required for the DMA transfers into the Illiac in order to overlap transfers with computation. This interface is relatively simple and is similar to other interfaces fabricated by the Institute.

The implementation of the XMEM would be in four phases. The first phase includes the design of the PEM-XMEM interface, the fabrication of an initial section (which would handle four PEs), its installation and check-out. This phase should take from six to nine months. The second phase would be the installation and check-out of one extended memory module; this should take three months. The next nine months would constitute the third phase: mainly installing and testing the remaining memory modules. During this time the following capabilities of the extended memory should be realized: the basic transfer capability which consists of moving data addresses between the source/destination descriptor and the MAR or XMEM address register, moving data from PEM to the I/O bus and from the I/O bus to the MIR, and refreshing the extended memory. The final phase consists of changes to the control unit and would extend over several months. These CU changes would be necessary in order to implement the PE indexing capability and the ADVAST indexing capability, to overlap the transfers of multiple rows with computing, and to protect individual PEs during a move instruction.

The cost of this approach to extended memory replacement would be about 1.2 million dollars for the memory modules and an additional 0.6 million dollars for designing and fabricating the interface.

IV.B.2.b.iii. Possible Extensions

There are two notable extensions possible to the RAM extended memory. The first would be the ability to load PE registers (A, B, S, and X registers) directly from the XMEM. This would allow the XMEM to be treated exactly as PEM, only it would be slower. The second extension would be to increase the size of XMEM. This would allow both the staging of user jobs and the capability to run larger problems efficiently.

Loading the PE registers directly is fairly straight-forward. This involves replacing one card type in the PEs. There are eight of these cards in each PE so that for the entire system, plus spares, 560 cards are required. The cards will cost less than $100,000 to fabricate. This capability is considered as an extension rather than as part of the initial design since it requires more changes to the control unit as well as to the processing elements, and the present proposal does not require the modification of the processing elements themselves. It is also possible that when this particular card is designed, other modifications to that card may be appropriate.

Expanding the size of the XMEM has several possible solutions. There will be panel space for up to thirty-two million words using twice as many boards. There is some concern about the reliability of that number of chips, and experience with the smaller XMEM should prove decisive in determining this issue. It is also possible to replace the 16K RAM chips with 64K RAM chips. This is a real possibility in the next three years and may at that time have the same cost as the present 16K RAM chip expansion proposed here.

IV.B.2.b.iv. Discussion

The main advantage of the RAM approach for the extended memory is the random accessability of the memory and the consequences of this feature for the programmer. The negligible latency, random access to single rows and independent indexing into this memory mean that the programmer can treat the extended memory as a homogeneous extension of the local memory. In addition, using XMEM for program storage would remove the need to implement program overlays and also eliminate instruction storage from contending for PEM space. The high data rate has an additional advantage for the programmer in that only simple buffering is required. The negligible latency means that no user specified synchronization between computing and data transfer need be considered since this can be handled at an instruction level by the control unit.

With respect to the hardware, using existing memory boards in extremely simple modules available from vendors is a definite advantage since qualification has already been done and reliability data are known. On the other hand, the interface between these modules and the Illiac does place a demand on the resources of the Institute, and the new instruction requires modifications to the control unit. This modification has some element of risk to regular Illiac service as does all CU work. As a fall-back position the XMEM could be interfaced with the array in the same fashion as the CCD-based extended memory proposals. This would simplify the interface while providing a "zero latency disk."

Finally, this approach requires software modifications, both in the system area to improve file transfer capabilities and in the language areas to incorporate the move instruction into the existing compilers and assembler. Both of these tasks are significant, but not inordinate, burdens on the resources of the Institute.

IV.B.2.c. Disk-Based Memory System

A third and final memory technology to consider for the extended memory is magnetic disks. Since the major disadvantages of disk technology are its latency and low transfer rate, this approach will consider using only the disk with the lowest latency and highest transfer rate, namely the Ampex DMP-9309 disk drives. The description of a memory system using these devices will also be used in discussing the on-line file system for the Illiac, although the demands on the performance of such a file system will not be as stringent unless that system is frequently used to contain the active data base of problems which are too large for the extended memory (which may be the case if either the RAM or PEM alternative is chosen for the extended memory).

A configuration using four synchronized Ampex DMP-9309 drives will supply a collective peak bandwidth of 348 megabits per second, and while that is somewhat lower than the current I4DM to PEM transfer rate, it is still adequate for most of the problems surveyed in this study. The four drives can be expanded to eight or more drives and this is easily accomplished by daisy chaining additional units behind the initial configuration of four. For reliability the system should be operational with fewer than four drives. Figure IV.15 shows a block diagram of an extended memory system using four or more of these drives. The four sets of drives are operated by four disk controllers which may function either independently or in a grouped fashion for full bandwidth transfers. A selection and multiplexing mechanism on the computer side of the Disk File Controllers (DFCs) could select one to four data streams from the DFCs and combine them to provide a 256-bit wide data stream for either the local memory of the array or for the support system. If the PEM path is selected, the data must be de-multiplexed to produce 1024 bits to go directly to the PE cabinets. Alternatively, if the IOS is used, the 256 paths may be taken directly to the IOS.

If the support system path is selected, the output from this memory must be buffered to match the bandwidth of the central memory. The size of the buffer should be at least six to ten Illiac pages in order to hold the contents of one or two DFC buffers and the support system communication region (much as the BIOM currently does in performing the identical function).

File transfer would be controlled by a PDP 11/40 which would be temporarily interrupted by the CU's TMU if PEM transfers are to take place. Requests for PEM transfers as well as file transfers would be queued. The PEM transfer queue would have priority over the file transfer queue. As long as any tasks are in the PEM transfer queue, all file activity to the DFCs is suspended. The reason for this is the expected arm contention from

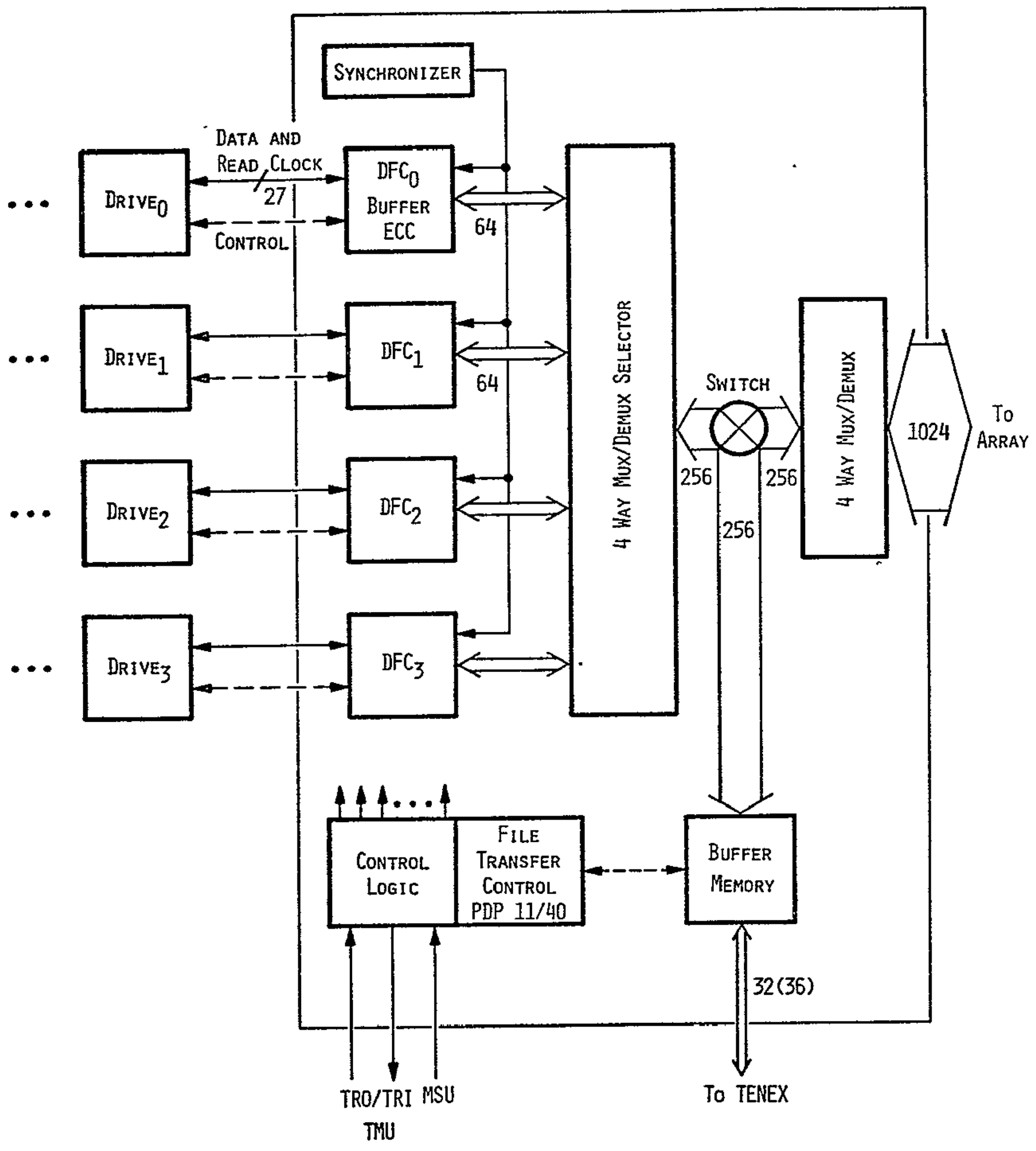


Figure IV.15.

I4 Extended Memory Using Ampex Disks

servicing two types of requests in an interleaved fashion. Although requests are handled as a unit, actual transfers occur in smaller bursts of activity. Transfers between the disk and PEM proceed one sector at a time, whereas the unit of transfer between this memory and the central memory is the size of the buffer.

Each DFC (see Figure IV.16) is composed of a master controller which controls the read/write functions of the disk and nine slave controllers. On a write operation, nine data streams are synchronously and simultaneously sent to nine disk write heads. On a read operation, data is read from nine read heads simultaneously, but the data is usually not synchronous so that nine clock lines are also provided for independent data clocking. Each data stream to be read or written is clocked serially into or out of sixteen 1024-bit buffers. Each 1024-bit buffer when full/empty would cause the controller to sequence the buffer selector to the other buffer. A convenient sector size (per head) would be 8192 bits; this would fill exactly eight buffers. The sector gap time would be larger than the data skew between heads which must be resolved before proceeding to the next set of eight buffers.

When a set of eight buffers per head is full for all nine heads, 72-bit word transfers may commence through the SEC/DED logic to provide 64-bit data words to the Illiac array memory. While this transfer is taking place, the DFC will continue to the next sector and proceed to fill the next buffer. A similar sequence occurs for the write transfer.

The reason for buffering in this manner is the use of a Hamming code error correction technique whereby a burst error may be corrected as data passes through this buffer. This method of error correction, essentially an interlaced code, is robust for extremely long burst errors and multiple short burst errors (as long as the multiple bursts do not occupy the same relative buffer positions) and allows a regular implementation using RAM memory. The penalty of this method is that the entire buffer must be read before error correction can be performed, hence the need for two buffers.

The cost of the DMP-9309 drives is about $45,000 per drive with another $60,000 for each controller. The MUX/DEMUX circitry, including cable drivers and receivers and a buffer memory for the support system should run about $150,000. The control logic and PDP 11/40 file processor should cost about $100,000. Thus the entire memory subsystem would total about $700,000.

The Ampex disk units and controllers will be available in the first quarter of 1979. Delivery of four drives could take place in the second quarter. Another four to twelve drives could be purchased later to expand the data staging capability in nine billion bit (four drive) increments.

As an extended memory, this approach has the advantage of having a large capacity; indeed, its capacity is so great that excess storage can be used for staging subsequent Illiac jobs and for holding re-start data for multiple-run jobs. It is also attractive on a cost-per-bit basis. It is, however, a moving head disk, and this additional complexity over the

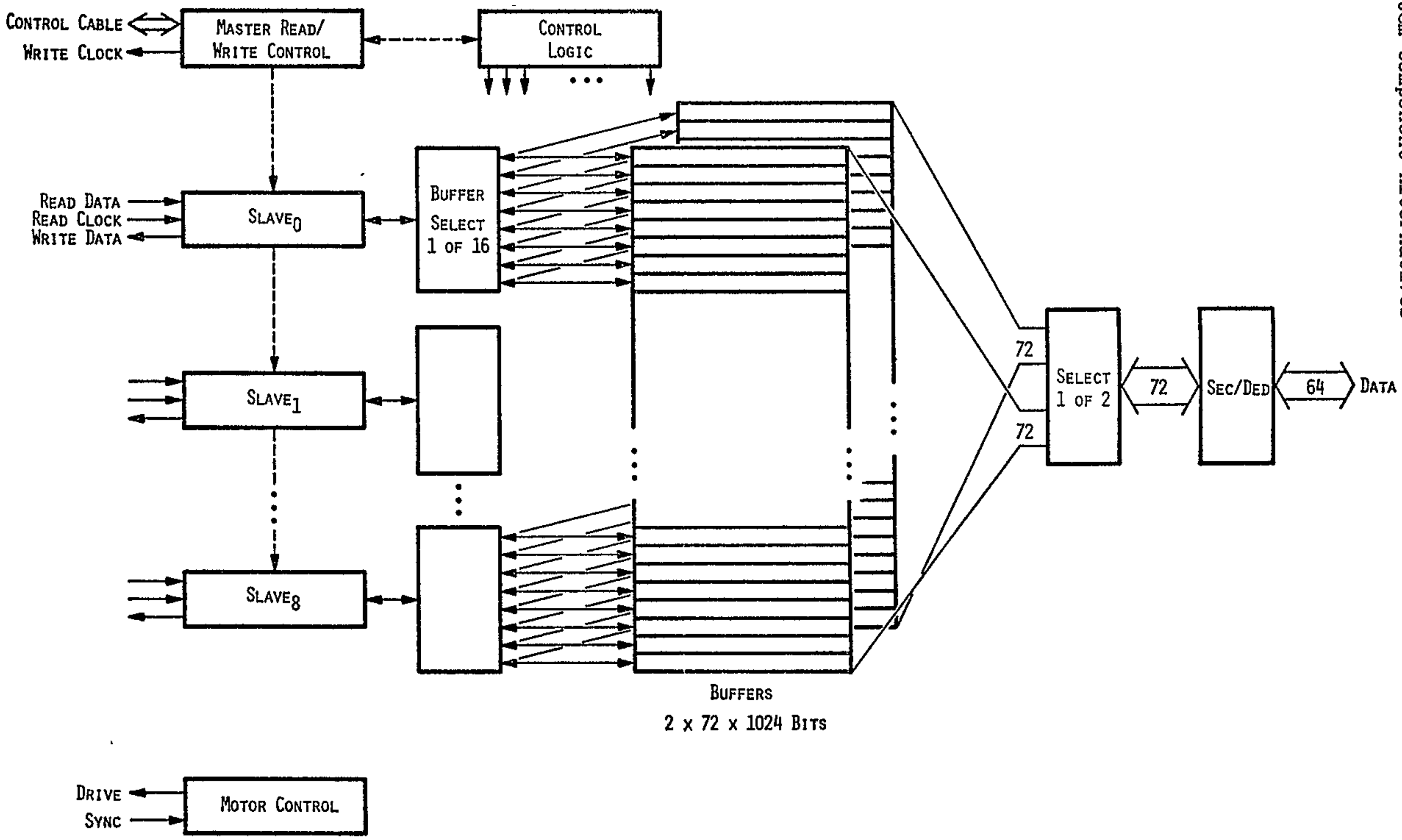


Figure IV.16.

Disk File Controller for Ampex Disk Memory present fixed-head disk system is a distinct disadvantage. If this memory can process file requests while an Illiac job is running, then the position of the arm cannot be guaranteed and it makes little sense for a user to arrange data on this system to any extent more than to guarantee that accesses, once initiated, are performed efficiently (i.e. the level of mapping required is merely with respect to reading and skipping disk areas, not arranging data so that subsequent read areas, after some intervening computing, will be readily accessed). This suggests that a very large local memory must be provided with such an extended memory, and that a simple file structuring mechanism be provided for those users who must rely on this extended memory to hold the active data sets for their programs.

The technological risk associated with this type of extended memory is negligible, despite the fact that the disk under consideration is not yet in production. Disk technology is well understood and the parameters of this disk are within the state-of-the-art capabilities of disks. In addition, this particular disk is designed as an extension of a standard product line at Ampex, so that delivery and future supply is reasonably well assured. In addition, this particular application is not expected to be a major drain on Ampex's production capability for this device.

IV.B.3. On-line File System

The Illiac file system will perform many types of data movement for users; it will be general enough to handle basic services, but since the Illiac system is a restricted environment there is no need for the sophistication of a fully developed operating system. The file system will manage the buffer/file memory; this memory will be implemented using standard disk technology and will be structured similar to the disk memory described in sub-section IV.B.2.c. Depending upon the performance required of this system, several different types of disks could be used, but because of the large files which are characteristic of Illiac jobs, a high transfer rate will probably be needed and this indicates one of the high perormance disks will be used.

The buffer/file memory will contain two types of files; one will be user files and the other temporary or scratch files. The former type will be created by special users with each such user having a maximum amount of space available for such files. These files have a guaranteed minimum life-time, both in calendar time and in hours of available Illiac processor time. After this guaranteed life-time, files will be subject either to migration to tape or to automatic deletion. Scratch files can be created by Illiac jobs provided there is sufficient space; their life-time is guaranteed only during the run of the program, but they will remain in the system until the space is needed.

IV.B.3.a. Purpose of the File System

The Illiac file system will be used to manage collections of information which fall into three classes. First is buffered data: information passing through the Illiac file system on its way to its ultimate destination. Second is temporary storage for Illiac jobs; such files arise when the active data base of an Illiac job exceeds the size of the Illiac memory. Third is user storage; this is data which is owned by a specific user and which remains in the Illiac file system for an extended period. The information in this system will be organized as files, and the distinction as to which class a file belongs is useful only when deciding if a file should be removed from the system.

The Illiac file system consists of various storage components. These components have different characteristics which are described as follows. The primary memory of the file system is the buffer/file memory which will be called I4DSK in this discussion. This storage component will hold the on-line Illiac file system, including buffered data, Illiac temporary files and Illiac user files. All files are stored with 32-bit word lengths in fixed length blocks of 2048 words. The Illiac extended memory (I4EM) holds active data files for Illiac runs in progress. The PE memory (PEM) is a storage component which is not organized into files, however it is a part of the overall data movement system and is accessed using absolute addressing. The Illiac tape facility (I4TAPE) is used to store files off-line on magnetic tape. Files on I4TAPE are recorded sequentially in blocks of 8192 8-bit bytes, equivalent to an Illiac page. The Tenex A File System (SYSA) is independent of the Illiac file system, but it plays an important part in the overall data movement system with which the Illiac system must interact. The Tenex system supports 36-bit word lengths in blocks of 512 words, called Tenex pages. When a Tenex file is used in the Illiac system, it is always converted to 32-bit word length by truncating each Tenex word and using only the most significant 32 bits.

The organization of a file is handled differently depending upon the storage component. I4DSK files are sequentially allocated files with a static size, as are I4EM files. I4TAPE files are sequentially allocated records on tape, but the size of the file is not determined until the completion of a file write. SYSA files consist of randomly allocated pages which are located through a multi-level index structure.

Files are managed in the Illiac file system with the use of a simple directory structure. This directory indicates the file storage location on the buffer/file memory and in the extended memory. This directory is stored on the buffer/file memory for efficiency and backed-up on the Tenex disk for reliability. Files of I4TAPE are managed with a tape catalog service provided by Tenex; a mechanism is provided by which the file transfer control process can query and up-date the tape catalog.

In the Illiac file system, file names are composed of three parts: the storage component designator, the directory name of the file owner, and the file name. For example, I4DSK:<Mahler>Input1 would specify a file owned by

Mahler which is stored in the Illiac file system. Either of the first two parts may be defaulted to system supplied arguments. If the directory name is omitted, then the default is the directory to which the job using the file system is connected on Tenex. If the storage componenet designator is omitted, the default is the Tenex file system (SYSA).

IV.B.3.b. File System Usage

The Illiac file system can be used in interactive, batch or system modes. The interface to the file system functions at each of these levels will be described in turn.

In the interactive user environment, moving data between any two storage components for the authorized user is accomplished by the MOVE subsystem. This subsystem accepts a source file name and a destination file name; for example,

```
MOVE I4EM:Area1,SYSA:Output.File
```

will move the designated file from the extended memory to Tenex memory. In interactive use, the MOVE subsystem will allocate space to the destination file if it is new. Users are restricted from moving files into the Illiac file system unless access has been granted by the appropriate authority.

The batch environment requires more sophistication in performing data movement, yet at the user level there should be little difference in the procedure or appearance from the interactive level. A file moved during batch may be moved directly to the storage component, or the file may be moved to a staging file on the Illiac buffer/file memory, with the de-staging transfer queued for later execution by the file system. For instance, moves between I4DSK and I4EM would be performed immediately, whereas moves from I4EM to SYSA or I4TAPE would be staged to a file on the buffer/file memory.

In the batch environment, once a batch job begins execution, it must be guaranteed that the required resources will be available to finish the job. In regards to data movement, this means that file space must be pre-allocated before the job begins. The user must therefore place a declaration of required resources at the beginning of the batch control file. This declaration is the ALLOC statement. For example,

```
ALLOC I4DSK:500,TEMP:25000,I4TAPE:2,SYSA:1000
```

insures that at least 500 Illiac pages will be available in the user's directory in the buffer/file memory, that 25,000 pages are available in the buffer area for scratch files for Illiac jobs and that two scratch tapes and 1000 Tenex pages are available for file storage before the batch sequence is started.

Except for the inclusion of the ALLOC statement, a batch control file contains statements which could be executed interactively with no change. While a batch job is waiting to be started, a staging process scans the batch control file for MOVE statements. Files which are to be read into I4EM or PEM are moved into the buffer/file memory and given the same name. When a job is executed in batch, the MOVE and RUN subsystems first attempt to locate the files in I4DSK. Files would be found on I4DSK if the staging process had sufficient time to move the files, otherwise the files are retrieved from the specified storage component (SYSA or I4TAPE). In the batch sequence, a MOVE which writes data from I4EM or PEM to SYSA or I4TAPE causes the source file to be moved to a temporary I4DSK file which has the same name as the destination file. The file on I4DSK is later moved to its ultimate destination by the de-stage process.

In the run-time environment, Illiac programs are able to access the PEM and the I4EM directly, but must request the file transfer control process to perform file transfers between I4DSK and I4EM or PEM. A file on I4EM or PEM is just a convenient way of referencing a block of storage without knowing the orgin of that block at compile time. The Illiac programming languages will provide functions to deal with the I4EM or PEM files. There are three commands of interest: MAP, READ and WRITE. When a MAP(file,array) statement is executed, this function associates a file's location with a structure in the program. In this way, references to the array will access locations in the block of storage assigned to the file. To re-map the array to another file, the MAP function can be called again. The READ(file1,file2) command is equivalent to a MOVE batch control statement moving file1 on I4DSK to file2 on I4EM. This function allows a program to overlay its active data base under program control. The WRITE command is similar to the READ command except that it moves data from I4EM to I4DSK. Instead of the default buffer/file memory designation of I4DSK, a user may specify TEMP instead.

In the system environment, the file transfer control process provides functions to be used by the Illiac programming languages and the data movement subsystems. These functions handle management duties and transfer operations. The OPEN(filename,size) command opens a file of maximum extent 'size' for transfer; it checks to verify that the user has access privileges or it assigns storage if the file does not exist at the time of the call. The MOVEPAGES(file1(offset1),file2(offset2),pages) command causes the specified number of pages to be moved from file1 to file2; the optional offsets allow transfers to begin at locations other than the beginning of the file. The LOCATE(file) command returns the origin address and size of a file if the file is located in I4EM or PEM; this command is used to map a file to an array in an Illiac program. The STAGE(file1,file2) command queues a transfer to be handled by the stage/de-stage process. This is equivalent to a MOVE statement except that control is returned immediately rather than after the transfer has been completed. The CLOSE(file) command closes an open file and updates the necessary system tables.

IV.B.3.c. Implementation Considerations

Specific problems require more consideration to arrive at a feasible design. Some of the issues which deserve attention are the complexity of the file structure, file protection, restriction of resource use, temporary versus permanent files, deletion of files, confidentiality of data, Tenex file access interlock, staging files and the general run environment save files. These issues will be discussed in turn.

The Illiac file system is used in a limited environment and does not need many of the features commonly found in a file system to support a full operating system. Therefore the file system structure can be very straight-forward. Since the files in the Illiac system are not subject to relocation in a virtual memory, a simple sequential allocation scheme can be used. To define a file, the only information required is a start address and a page count. It may, however, prove necessary to implement a structure for segmented files if fragmentation occurs so frequently that space reclaimation costs become prohibitive.

Because of the simple nature of the file system, storage assignment should also be simple. Assignment starts at one end of the logical address space and proceeds towards the other end, with storage being assigned by a best-fit or first-fit algorithm. When space becomes scarce due to fragmentation, a storage compaction operation can be performed. This operation can occur during one of the scheduled periods of Illiac unavailability (such as during confidence checks), or it may be automatic when a batch job requires space to run.

File access protection can be implemented in the Illiac system in the same fashion as in Tenex. When a file is moved from Tenex to the Illiac file system, the protection key stays with the file. When a file is created in the Illiac file system, it receives the same protection key it would if it were created in the user's Tenex directory. The file transfer control process interfaces to Tenex to request the necessary information to implement a consistent file protection scheme.

To insure that a user does not consume too much storage space and thereby prohibit another user's batch job from running, storage allocation must be limited to some maximum for each user who has been granted the privilege of creating files to be stored on I4DSK. This allocation is guaranteed to the user, and files created while the user is under allocation will remain in the system until explicitly deleted (although they may migrate to the tape system if they are not accessed within a given time limit). Temporary files and buffer files may be created using storage space which has not been assigned to any user. Such files have no guaranteed existence and may be deleted by the system when space becomes critical. Restrictions may also be placed on the number of tapes consumed by a user. Since files have a fixed size, the system checks file limits when I/O requests are processed to insure that a user does not use storage which has not been allocated.

A temporary file is created from storage space allocated from the general free space; this area is not charged to the user's personal directory. However, the price for this free allocation is an indeterminant existence. In a batch run, the ALLOC statement can be used to reserve temporary storage during the run, so a temporary file can be used throughout the run without fear of being reclaimed by the system. In this way, a user can minimize personal allocation requirements by using temporary scratch files. Permanent files are used to save results which will be used in subsequent batch jobs, and these files are available for processing tasks which have a special need to keep files on I4DSK instead of the Tenex disk or I4TAPE. The mechanism for implementing temporary and permanent files requires careful design to insure a consistent handling of such files by all system programs. The mechanism must allow a user to convert any file to be either permanent or temporary as necessary. This mechanism can either be implemented by a file attribute scheme or by dividing the I4DSK into a temporary component and a permanent component. In the latter strategy, temporary files could be accessed with an I4BUF: storage component designation.

A user may delete temporary or permanent files whenever such a file is not opened for a transfer. Deleted file space is then available for general allocation or for allocation only to the specific user, depending upon whether the file was temporary or permanent, respectively. The system may delete temporary files if the file space is needed for creation of a new temporary file, or if space is needed to increment a user's personal allocation. A selection algorithm must be developed which basically eliminates the least valuable file; an obvious criteria is to delete the least recently used files. In some cases, users may be requested to release file space so that a job which requires a significant amount of storage can be run. After deleting a file or releasing I4EM space, it may be desirable to zero the released storage to protect against accidental breach of confidentiality when the file space is re-assigned to another user.

A file which has been transferred to I4DSK from a Tenex file may be viewed either as a copy of the original or as the relocated original. The former approach, although simpler to implement, requires users to be aware of the possibility of multiple copies of a particular file and that the file they access may not be an up-to-date copy. The latter approach treats the Illiac and Tenex file system as a storage hierarchy. In such a scheme, the file which remains on Tenex disk is considered invalid until the copy on I4DSK is removed by writing it back to Tenex or by deleting it. During the time the file is on I4DSK, access to the copy in the Tenex file system must be restricted.

The implementation of staging/de-staging could pose the most difficult questions in the Illiac file system. The staging process should be as automatic and transparent as possible, and yet a user should be able to control the environment to make maximum use of the Illiac system. Intermediate files used in the staging process must be permenent files until they have been moved to their ultimate destinations, at which time they can be converted to temporary files subject to deletion. Of course, immediate deletion could be adopted, but this could result in redundant file staging if the file is needed twice in the same batch run (or even in different runs close in time). Implementing a hierarchical approach to file management requires solving the question of regulating access to a file which is scheduled to be staged or de-staged. File name contention between automatic staging operations and user operations require some thought to insure that a user's files are not over-written, as might happen if a file is staged for a write to Tenex disk, modified, and then staged for a write to I4TAPE. The selection of files for staging should be studied to determine if an optimum algorithm can be achieved.

Due to an operational policy that Illiac hardware verification tasks must be performed periodically, long running jobs must be run as several successive jobs and therefore the run environment must be saved. This is done by the SAVE subsystem which creates a structured file containing the run image, all status information, and copies of the I4EM files used by the job. The SAVE file is stored as a temporary file when created by the system because of a verify interruption. The SAVE file may also be created explicitly by the user to back up processing stages, in which case the SAVE file can be moved to any of the storage components: SYSA, I4TAPE, or permanent I4DSK file. A RESTORE subsystem is required to re-instate the run environment so that a program is unaffected by the interruption.

In summary, although there are several alternatives to be considered in implementing the Illiac file system, the basic outline of the system presented here suggests the characteristics of file manipulation that will be available to users. These involve the creation of permanent and scratch files and the automatic staging of files to and from the processor. In addition, users with special requirements will be able to manage the staging of their files explicitly and will also be able to specify the retention of Illiac files on the buffer/file memory for use in subsequent jobs. This file system is flexible enough to satisfy the requirements of the Illiac user community and simple enough to insure a simple and efficient implementation.

IV.C. Alternative Memory Systems and their Evaluation

The memory of the Illiac array consists of a fast, local memory and a slower, extended memory. Enhancing the array memory can thus proceed along two lines depending upon whether the local memory or the extended memory receives the major enhancement. One approach would be to increase the size of the local memory to the maximum which can be supported by the processor, i.e. to four million words. This still does not address the problem of a deteriorating I4DM. The machine must still be capable of solving problems which require considerably more storage than four million words, so a replacement for the I4DM must also be considered. Since a large local memory relaxes the performance demands on the extended memory, a high performance disk system can be used for the extended memory. This disk system can also serve as a staging memory between the Illiac and the central system; it can also function as a file system for re-start dumps of Illiac jobs.

For programs which require only four million words of memory, this approach would eliminate some of the problems which currently arrise in designin Illiac programs, since programs with large storage requirements must now use the I4DM. However, programs which would require more than four million words of storage would have to use the new extended memory. Although the characteristics of this new memory system (memory size and the ratio between local memory size and extended memory latency) are more than an order of magnitude better than the current system, large programs will still have the complexities which arise from partitioned data bases and buffer management.

Concentrating instead on replacing the extended memory with a very high performance memory, one can further improve the overall characteristics of the memory system. The two most attractive memory technologies for a high performance extended memory are CCD and RAM. The second proposed system enhancement consists of a half million word local memory and a sixty-six million word CCD extended memory. It also includes a buffer and file memory similar to the extended memory of the first alternative. For large problems, the overall system performance of this second alternative is about a factor of two over the first alternative, although for small problems (problems requiring between half a million and four million words of memory) its performance is only slightly better. On the other hand, almost all problems would be required to partition their data bases and sequence through them, keeping only a part in a buffer in the local memory. This complicates program design and adds to the time and cost of program development, but since this system's characteristics are much more than an order of magnitude better than the present system, this improvement should significantly reduce the current demands on programmer time and ingenuity.

The third proposed memory system is built around a RAM based implementation of the extended memory. It consists of the current local memory and a sixteen million word extended memory, together with the same buffer/file memory as the first approach. Because of the minimal latency and very high transfer rate associated with this type of extended memory, the performance of the memory system should be about five to ten percent better than the CCD approach for problems which require under sixteen million words of memory. The real advantage of this system lies in the ability to access this extended memory randomly, which virually eliminates the need for partitioning data bases and buffering their use. For problems requiring more than sixteen million words of memory, however, these data management problems return and the performance of the system is about two times worse than the CCD approach.

The first three sub-sections treat each of these memory systems in more detail, comparing their implementation, schedules, costs, risks, system impact, and usability. A final sub-section discusses their relative merits.

IV.C.1. Four Million Word Local Memory

The first proposed memory system is oriented around a four million word local memory. In addition, a disk-based memory is used for the purposes of 1) storing the active data sets of Illiac jobs which are larger than will fit in the local memory, 2) storing check-point or re-start dumps of Illiac jobs between runs on the processor, and 3) staging or buffering data transfers between the array and the support system. This buffer/file memory initially consists of four high performance disks with a total capacity of nine billion bits, average access time of 37 milliseconds, and a sustained transfer rate of more than 200 million bits per second. A later phase calls for increasing the capacity of the buffer/file to eighteen billion bits.

The local memory is fabricated using 16K dynamic RAM chips, and it replaces the current local memory inside the Illiac cabinetry. It uses the same power and cooling equipment as the current memory does. A straight-forward use of this new memory treats its system access time of 400 nanoseconds as six clock cycles (versus the current memory access time of ten clock cycles). A more complicated use of this memory would exploit a fast access mode of these chips, but since simulations suggest that the performance improvements are slight, this option will not be considered further.

Since the memory is dynamic, it must be refreshed periodically. The simplest mechanism for performing this task is a timed interrupt which causes the entire memory to be refreshed. This would probably be implemented by a counter in the control unit periodically issuing the appropriate interrupt to perform this refresh.

The extended memory in this configuration consists of high performance moving head disks. These disks can be synchronized so that four may operate in parallel, thereby increasing the transfer rate of the system over the 87 million bits per second available from one disk. Four disk controllers were chosen for this application to give a maximum transfer capability of 350 million bits per second, but a sustained transfer rate would probably be closer to 200 million bits per second. This is the minimum rate which the analysis of Chapter III indicates would be acceptable for current and forseen Illiac jobs. The average latency of these disks due to rotational delays is nine milliseconds, half that of the current I4DM, and the average delay due to arm movement is 28 milliseconds, and therefor the average access latency is 37 milliseconds, which is about twice the current I4DM. Since the disk system will be able to respond to system file transfers while an Illiac job is executing, the average delay Illiac jobs will experience may be closer to the latter figure. On the other hand, with a local memory thirty-two times larger than the present PEM, larger buffers can be used for non-PEM resident data bases, and using buffer areas sixteen times larger than is possible in the current memory will decrease the number of requests to the extended memory by a factor of sixteen. Thus the time a program loses due to the current I4DM latency will be reduced by a factor of eight for this disk external memory (although each access has twice the latency, there are fewer than one-sixteenth as many). Of course, solving larger problems than is now done on the Illiac will cause the absolute time lost to latency to increase, but if the amount of computing increases at least linearily with problem size, the relative amount of latency lost can only decrease from a maximum of less than an eighth of the current latency time (which, from Chapter III, varies from about half to slightly more than the compute time for current Illiac programs). The transfer rate of this new extended memory is about two and a half times less than the current I4DM; thus the transfer time of Illiac programs will increase by about this amount. This may become significant for a few of the very large codes; if they transfer large blocks at a time would have a burst transfer rate of approximately 240 million bits per second (this is significantly below the peak instantaneous transfer rate because of formatting conventions and head movement between cylinders).

Thus the system performance of this memory system will be significantly better than the current memory system with respect to latency and absolute memory size, which are the two major criteria of the Illiac memory system. The speed of the local memory is about a third faster than is presently being used in the machine, but about twice as slow as the machine could utilize; the effect of this would probably be to limit the increase in machine speed to about thirty percent. The transfer time between local and extended memory is about half the current transfer rate; for most codes their transfer times would still be under five percent of the compute time, but for large codes (such as 3D-TRANSONIC) they may exceed ten or even twenty percent of the compute times.

The initial cost of this system would be about a million dollars for the local memory, $200,000 for the disks and $400,000 for the controllers, the buffer to the support system, and the switching mechanism, for a total of

1.6 million dollars. The cost of the second phase would depend upon the amount of disk memory added to the buffer/file memory and whether or not error correction is added to the local memory. Because this local memory uses only half the chips as the present memory, and because the 16K chips are considered to be exceptionally reliable, error correction may not be needed. The local memory in the first phase has storage for the error correction bits already in the acquired memory boards, but the circuitry to generate the error correction code and to correct memory failures would reside in the memory service unit. The memory service unit would not be altered, beyond adding additional address wires, until the second phase. The cost of an additional four disks would be $200,000 and the cost of error correction would be about $250,000, for a total cost for the second phase of about half a million dollars.

The amount of Institute involvement required by this approach is basically the design of the memory boards for the local memory and the specification and design of the interface between the Ampex drives and the system; this latter task is simplified if commercial controllers for the Ampex drives can be used in this application. The technological risk in this approach is minimal: the RAM chips used in the local memory have been on the market in production quantities for about nine months, and disk technology has developed to the point that the characteristics of the Ampex drives appear well within the capabilities of the industry. The enhancement to the local memory will probably be scheduled for completion in mid 1980; the extended memory could be available in late 1979 or early 1980. The major impact of this approach on the current system software would be limited to the required file management system for the extended memory system, since it would also serve as the buffer/file memory system.

In summary, this proposed memory system offers a significant improvement over the current memory system; it greatly enhances the size and latency characteristics of the extended memory so as to attract large programs to the Illiac, and the four million word local memory should make the machine extremely attractive for smaller problems as well. The demands on the design and fabrication resources of the Institute are minimal, unless suitable disk controllers for the extended memory cannot be procured from industry sources. The one and a half million dollar cost for the initial configuration makes this approach fiscally attractive, and by using proven technology, the technological risk of using this approach is modest.

IV.C.2. Sixty-six Million Word CCD Extended Memory

The second proposed memory system consists of a sixty-six million word extended memory implemented in CCDs, a half million word local memory implemented in high speed bipolar logic, and a nine billion bit on-line file system implemented by high performance disks. The on-line file system is similar to the buffer/file system of the previous memory system, serving the same functions except that few if any Illiac jobs would use it to hold the active data base of the computation; the major difference in the two systems is that this buffer/file memory uses only two controllers since the performance demands will be less. This gives a saving of about $100,000 at the expense of half the transfer rate.

The extended memory in this configuration consists of sixteen modules of CCD memory boards; these boards would be a standard vendor product and the initial phase calls for each module to have only a quarter of its final capacity. The initial sixteen million word configuration could then be incrementally increased to the full sixty-six million word capacity merely be adding memory boards and power supplies. This extended memory, including the interface to the Illiac, would be supplied by a vendor; the cost of the initial configuration is about one million dollars and the cost of increasing the memory to its full capacity in the 1980-81 time frame is another million dollars. The initial memory would be delivered about a year after award of the contract.

The local memory in this system would be a replacement of the current local memory and would be scheduled for the second phase. A half million word memory made from 4K static chips would require only a quarter of the number of chips now in the PE memory, so that error correction capability would probably not be needed. The cost of this replacment in the 1980-81 time frame would be around $300,000.

The cost of the initial configuration of sixteen million word extended memory and the buffer/file memory would be about one and a half million dollars. The second phase, which doubles the capacity of the buffer/file, increases the size of the extended memory to sixty-six million words and replaces the local memory with half a million words would cost about 1.6 million dollars. The involvement of the Institute would consist of specifying the CCD memory for procurement and designing and constructing the interface for the buffer/file memory during the first phase. For the second phase, specification and possibly design of the local memory replacement boards would be required.

Although experience to date with 64K CCD devices is not as extensive as for other technologies under consideration, the technological risk involved in this approach does not appear to be unacceptable. Fairchild, for example, will be delivering a CCD system to Burroughs well before they would bid on this application; thus by the time a CCD memory would be acquired for an extended memory, there will be field experience with the technology. Even if CCDs have a significant transient error rate, a periodic refresh through error correction circuitry should provide an exceptionally reliable system. The controllers and control logic for a CCD extended memory can be quite simple with only a modest amount of hardware development necessary on the part of the Institute.

Since this configuration closely models the current system, it is fairly simple to predict the expected performance of this memory system. To begin with, the latency of the CCD memory is about a factor of twenty-five better than that of the I4DM. With a local memory four times the current one, buffers could be three tor four times their current size which in turn reduces the number of accesses to the extended memory. However, rather than seeing the time lost to latency of the extended memory being reduced by a factor of seventy-five or more, the factor will probably be closer to ten. The discrepency would be due to simpler buffering schemes being used, and "inefficient" lay-outs of data on the disk. Reduction by a factor of ten would bring latency time to within five to ten percent of program execution time. Most users feel this percentage is acceptable, especially when such performance results from a simple data management scheme. Such simplicity is possible with the high performance of the CCD memory system. This situation should significantly reduce the demands on programer time and ingenuity in writing Illiac codes, and hence meet one of the major objectives of enhancing the Illiac memory system.

There are two areas of system software that this proposal will significantly affect. The first area is the file system for the buffer/file memory, and is the same as for the other proposals. The second area is in the control of the extended memory. Currently this task is performed by the MMP which takes more than two milliseconds to respond to a request for access to the I4DM. This is clearly unacceptable for a memory whose maximum access time is one millisecond; hence a new I/O executive will be required, either running on the Illiac itself or on a dedicated processor with adequate performance. Part of the function of the MMP is to map the logical address which a program issues into the physical address of the I4DM. This capability increases the availability of the I4DM since damaged areas of the disk can be programmed out of the user address space without affecting the user's program. This capability will not be needed by the new extended memory, so the I/O executive will be simpler and faster than the current MMP.

In summary, this proposed memory system offers significant improvements over the current memory system; it gives the Illiac the capability of efficiently processing problems requiring up to sixty-six million words of memory while significantly reducing the complexity of using the machine. The one and a half million dollar cost for the initial configuration makes this approach fiscally attractive, and the demands on the resources of the Institute are minimal.

IV.C.3. Sixteen Million Word RAM Extended Memory

The third proposed memory system consists of a sixteen million word extended memory which uses 16K dynamic RAM chips and a nine billion bit buffer/file memory which uses high performance disks. The buffer/file memory in this configuration is identical to the one in the first proposal, since this memory will have to support the users whose data bases will not reside in the sixteen million word extended memory.

The extended memory is basically the memory described in section IV.B.2.b. Of the three proposals, this extended memory has the greatest flexibility and performance. Processors will be able to access the memory with the same flexibility they now access their local memory. This capability will virtually eliminate one of the major difficulties in using the machine with its current memory configuration.

In particular, the RAM extended memory may be independently addressed by the processing elements. Codes with sophisticated data structures could easily take advantage of this feature, for example in managing independent queues in separate processing elements. Particle codes such as those used in plasma and nuclear physics and physical and quantum chemistry usually contain sections that are naturally formulated in this manner. However, all users should benifit from RAM extended memory by not having to partition their data bases into small working sets for PEM. This should significantly reduce the complexity of writing programs for the Illiac system. Both of these advantages should attract new applications that previously would have been considered impractical on the current Illiac memory architecture.

Of the three extended memories proposed to replace the I4DM, the RAM extended memory has the highest implementation risk, and this risk is significantly greater than for the other two. This is because it is closely coupled with the control unit whereas the others have a simple interface, similar to he current CU-I4DM interface. The control unit is a complex device with many interacting components; even simple changes can have unforseen ramifications. In addition, only a small number of personnel understand this unit to the point of being able to make the additions to the CU which are required by this approach. These two factors combine to make the implementation of this approach a significantly greater risk than the other two proposals. However, there is a fall-back position for implementing a RAM extended memory. This would be to use a simple interface, such as one described for the CCD extended memory. This would in effect provide users with a "zero latency disk" which could be used with almost the same amount of flexibility as the proposed RAM memory (the major exception being PE indexing).

The cost of this memory system would be about 1.8 million dollars for the extended memory and $600,000 for the buffer/file memory. Since both the extended memory and the buffer/file memory require design and fabrication of interfaces by the Institute, the availability of personnel to perform these tasks will dictate the scheduling of when these two resources become available for use. With the extended memory receiving the initial attention, this part of the system should be available, even with limited capabilities, by late 1980. The buffer/file memory should be available about a year later.

The performance of programs whose data base is larger than sixteen million words can be estimated by a simple calculation. Assume a sixty-six million word data base which resides on the buffer/file memory. If the program must sweep through this data for each step of the program, and if the computing is performed at the generic rate of 100 million bits per second, then the compute time to process the data base is 40 seconds. To process the data, it must be brought from the buffer/file memory to the extended memory and then written out again. Since this can be done in four blocks, the latency of the buffer/file is negligible. The transfer time, however, is considerable: it takes 40 seconds just to move the data. There are two points to this example. First, for programs which do not fit within the extended memory, data movement time will be comparable to the compute time, as it is in the current system (but because of the transfer rate in the proposed system rather than because of latency in the current system). Second, despite this fact, programs with very large data bases are still possible with this memory system.

The implications of this memory system with respect to system software fall into two areas. The first is again the file system for the buffer/file memory. The second area involves a new instruction. Both the assembler and the higher level languages must be modified in order to provide access to this instruction. Although the addition to the assembler is conceptually straight-forward, it is complicated by the fact that the assembler has not been altered in several years and there is no one at the Institute who has worked on the assembler. There are several options with regard to incorporating the new instruction into the existing higher level languages. The simplest is merely to allow access to this instruction the same way users gain access to any other assembler instruction. A more elegant way would be to provide an intrinsic or library function which performs the move function. Finally, the most sophisticated option, and the one requiring the most work, would be to automate the movement of data from the extended memory to the local memory with the compiler detecting when data must be moved to the local memory and generating the appropriate instructions (as well as managing the temporary storage in the local memory for extended memory variables).

In summary, this third memory system offers the most flexible extended memory which should significantly improve the system performance and reduce the programming complexities of the present system. The implementation of this system places a significant burden on the resources of the Institute. The cost of this system, even at two and a half million dollars, is attractive from the point of view of the additional flexibility which it offers, and problems larger than the sixteen million word extended memory can still be solved by using the buffer/file memory as third level in the memory hierarchy.

IV.C.4. Relative Merits of the Systems

To aid in the exposition, the three approaches described above will be referred to as the PEM, CCD and RAM approaches, respectively. The three approaches differ primarily with respect to initial cost, size, system impact, effect on program development, and demand on the Institute's resources for hardware development. The technological risk involved in the three approaches are acceptable: there is essentially no risk involved in the RAM and PEM approaches. The CCD approach has the greatest amount of technological risk associated with it, since the technology chosen for the extended memory has not yet been used in large scale memories; this risk, however, is not inordinate. With respect to implementation risk, the RAM approach is significantly greater than the other two since it involves a much more complex coupling with the control unit; this risk, however, can be avoided by using a simple interface, if necessary.

The amount of in-house development required for these three approaches varies considerably. All approaches use the same system for the buffer/file memory, and the interface, although complex, is straight-forward. The only issue is the design of the disk controller and whether the design and fabrication is performed by the Institute or is acquired from an outside vendor. Ignoring this common task, the CCD approach requires no in-house development aside from specification, the PEM approach requires a modest involvement, and the RAM approach requires a considerable amount of in-house development.

The three approaches have different impacts on the current system software and on user's program development efforts. Again, the impact of the buffer/file memory is roughly constant for the three approaches, and will be ignored in this discussion. The PEM approach requires negligible changes to the current system software. For the CCD approach, the rapid response time of the disk replacement will necessitate that the current control mechanism be replaced by an I/O executive dedicated exclusively to managing transfers between local and extended memory. This will require the development of the appropriate system software. The RAM approach also requires considerable system software changes; since it involves adding an instruction to the current Illiac repetoire, changes must be made in the assembler and probably in the programming languages as well. As far as the impact of these three approaches on user code development is concerned, the PEM approach will be of most help to users whose problems fit in the four million word memory; for larger problems, about the same level of effort will be required as is now being expended. The capability to have much larger buffer areas may mitigate the amount of time which must be spent during program design in order for a program to be reasonably efficient. The CCD approach, by significantly improving the performance characteristics of the extended memory, will greatly reduce the effort required to use the machine efficiently, and the RAM approach, for problems requiring less than sixteen million words, should virtually eliminate the problems users experience with the current extended memory.

As a final point of comparison, the relative performances of the three memory systems varies depending upon the size of the active data base being processed. The target Illiac applications for the fully enhanced Illiac system can be grouped into four categories according to the size of their data base: those problems with less than one-half million words, those problems with one-half to sixteen million words, those problems with sixteen to sixty-six million words and those problems with greater than sixty-six million words. Problems in the first category are best served by either the RAM or CCD approach with a 5 to 50 percent degradation expected going to the PEM approach. This is due to the fast local memory provided with these two approaches. Since the fast local memory in the CCD approach is one-half versus the one-eighth million words of the PEM approach, about 10 to 50 percent better performance would be expected of the CCD approach in this range. Problem data bases in the one-half to sixteen million word range would best be served by a RAM memory system with a 5 to 10 percent degradation expected going to CCD since most programs will lose only 5 to 10 percent of their execution time due to latency and transfer time. A degradation of up to 50 percent should be expected with the PEM approach for problems under four million words; above four million words the degradation would be 100 percent since part of the data base must then be stored on disk. Problem data bases in the sixteen to sixty-six million word range would perform 100 percent better with the CCD approach over RAM and PEM since data bases in the latter two cases would be stored on disk. Over sixty-six million words the picture reverses with both the RAM and PEM approaches performing 25 to 30 percent better than the CCD approach. This is due to the fact that in the CCD memory system the extended memory has only two high performance disk controllers instead of the four used in both the PEM and RAM approaches.

In summary, all three approaches increase the reliability and availability of the Illiac system while significantly reducing a major cause of the inefficient use of the Illiac and a major cause of the high cost of program development. With respect to cost and technological risk, the three approaches are comparable although their differences are perceptible. With respect to the amount of system software changes, the CCD and RAM approaches require significantly more work than the PEM approach.

There are four major differences in the three approaches. First, with respect to in-house development and required modifications to the Illiac (and the resulting unavailability of the Illiac), the RAM approach is significantly more demanding on the Institute's resources than the other two. Second, the RAM approach is significantly more expensive for a minimal configuration (the first phase). Third, with respect to the size of the data base which can be processed without having to resort to the complexities of using a disk base memory efficiently, the PEM approach is the most restrictive, being limited to 200 million bits, the RAM next most restrictive, limited to one billion bits, while CCD is least restrictive with a limit of four billion bits. Finally, with respect to the complexity of program design, PEM and RAM approaches are roughly comparable, with the CCD approach being significantly greater, and the PEM and RAM approaches significantly greater for large programs using the buffer/file memory to hold their active data sets. Despite their differences, all three approaches are feasible and significantly improve the memory system characteristics.

IV.D. Ability of Proposed System to Meet Future Demands

This section reviews the necessary and desirable characteristics for future memory enhancements which were derived during the first phase of the memory study and were summarized in section III.E. The capabilities of current memory technology are represented in the three proposed systems, and each proposed memory is discussed with respect to how well it meets the specified requirements. Five major areas of weakness were identified in the present system. These are:

1. The limited bandwidth between the system and the outside world and the resulting time required to move data onto and off of the system.

The solution to this problem involves upgrading the support system's disk and file storage media. This is currently being pursued, but the solution of using commercial IBM compatible components will still limit the transfer rate of the support system to about five megabits per second. However, since the Institute supports external users, this weakness can be improved only as the computing community at large develops faster I/O devices.

2. The lack of a large on-line storage facility to keep active data bases between runs over a period of a few weeks.

This weakness is met by the buffer/file memory in each of the three proposals. Of the eventual capacity of 18 billion bits, approximately 12 billion should be available for on-line file storage. This would increase the absolute file storage by a factor of six, and since most of the buffer/file memory will be devoted to the storage of user files, this should substantially improve the current situation. As designed, the maximum capacity of this memory is 36 billion bits, although for the CCD approach, two additional controllers would be needed, in addition to more disk drives, to achieve this expansion.

3. The low bandwidth between the file system and the array, and the resulting time to load and unload the array memory.

The effective transfer rates of the buffer/file memory is 200 million bits per second for the PEM and RAM proposals and 100 milion bits per second for the CCD approach. This represents factors of 100 and 50, respectively, in improved transfer times over the present system. This is, of course, over-design, but it has the additional benefit of permitting large problems (those over 4 million words in the proposed PEM memory system and over 16 million words in the proposed RAM memory system) to be run with reasonable efficiency (about 50%) using the buffer/file memory to store the main data base.

4. The bandwidth between local and extended memory and the latency in accessing the extended memory, which together lead either to inefficient use of the machine or to extended program development time to reduce these inefficiencies.

The size of the local memory is important in the efficient use of the machine: its major function is to hold the data to be accessed by the processor. If there is an associated delay in accessing the extended memory then part of the local memory must be set aside to buffer accesses to the local memory. The more area that can be reserved for buffering, the more data that can be accessed with each request and hense the fewer requests required and the less time lost to latency in servicing those requests. Thus a figure of merit to be applied to a two level memory system is the ratio of the local memory capacity to the average delay in accessing the extended memory. Clearly, the larger the ratio, the easier such a system will be to use efficiently. The current system has a ratio of about 419 million bits of storage per second of latency. The PEM approach has a ratio of 9.6 billion bits per second, the CCD has a ratio of about 66 billion bits per second, and the RAM has a ratio of about 25 trillion bit per second. There are two implications to these numbers. The first is that all proposed memory systems offer at least an order of magnitude improvement over the current memory system. The fact that both CCD and RAM are significantly more than and order of magnitude improvement has little impact since after an order of magnitude improvement, subsequent improvement can be at most 10 percent of the original situation. The second implication is more important, and that is the way users approach designing their programs can change with two of these three proposed systems. With the CCD proposal, more flexibility is possible due to the greater efficiency of accessing the extended memory for small transfers. With the RAM proposal, the flexibility is so great that little if any consideration in using the two level memory will probably be required in designing programs with data bases smaller than sixteen million words (although for larger problems, the program design process will be similar to the current process).

5. The declining amount of extended memory.

All of the proposed memory systems offer larger, reliable memories for their extended memory. In contrast to the current I4DM which has a half half billion bits, the RAM offers a billion bit extended memory, the CCD approach a four billion bit extended memory, and the PEM proposal an eighteen billion bit extended memory (which also serves as its buffer/file memory).

Thus each of the proposed memory systems directly responds to four of the five areas of weakness in the current system. The three systems differ primarily in the size of their extended memories and in their approaches to solving the problem of a low effective bandwidth between the local and extended memory. Any of these three systems should significantly improve the Illiac's memory system and continue to maintain the system as a major computing resource for the scientific research community.

Glossary of Terms

The following is a list of abbreviations and terms used in this report. Terms which are defined and used only within a local context have not been included in this list.

ACAR: one of four 64-bit general purpose registers in the Illiac processor's control unit.

ADB: one of sixty-four 64-bit scratch pad registers in the processor's control unit.

ADVAST: the Illiac Control Unit's ADVAnced STation (see page 24).

BIOM: the Buffer I/O Memory (see page 14).

Buffer/file memory: a high performance, large capacity memory to be used to buffer transfers between the support system and the Illiac; this memory is a proposed addition to the system.

CCD: Charged-Coupled Devices are a serial solid-state memory device.

CCD approach: a proposed memory system based on a sixty-six million word memory made of CCDs (see page 97).

CU: the Control Unit of the Illiac (see page 24).

DC: the Descriptor Controller in the current Illiac disk memory (see pages 14 and 75).

DFC: the Disk File Controller of the current Illiac disk memory (see pages 14 and 70).

DFDC: the Disk File Descriptor Controller in the current Illiac disk memory (see pages 14 and 75).

DMA: Direct Memory Access without interrupting a central control unit.

Data Rate: the rate at which a program processes its data base, considering only computing and not memory accessing delays (see page 36).

Extended memory: the memory of a large computer which holds the bulk of the active data base during a computation.

FINST: the FINal STation of the Illiac control unit (see page 24).

Gigabit: one billion bits.

ILA: the Instruction Look Ahead unit of the Illiac control unit.

IOS: the I/O Switch of the current Illiac disk memory (see pages 14 and 75).

IOSS: the I/O Sub-System of the Illiac (see page 15).

I4CM: the Illiac IV Central Memory (see page 14).

I4DM: the Illiac IV Disk Memory (see page 16).

I4TAPE: the Illiac IV TAPE system (see page 22).

K: 1024.

Local Memory: the fast memory local to the processor in a large scientific computer (the PEM on the Illiac).

M: 1048576; thus 64M is slightly over sixty-six million.

MAP: a software system for users to arrange their data on the I4DM.

MAR: the PE's Memory Address Register.

Megabit: one million bits.

Megabyte: eight million bits.

MIR: the PE's Memory Information Registers, used for communication between the PE or I4DM and the PEM.

MMP: the Memory Management Processor (see page 14).

MOVE: a sub-system for moving data among memory components of the Illiac memory system (see page 89); also a proposed Illiac instruction to move data between the extended memory and the local memory (see page 79).

MS: the MMP Server (see pages 14 and 21).

MTBF: Mean Time Between Failures.

MU: the Memory Unit of the proposed CCD extended memory (see page 72).

M/L: the ratio of local memory capacity to the latency of the extended memory, used as a figure of merit for hierarchical memories (see page 40).

NMOS: N-channel Metal Oxide on Silicon, a semiconductor process for fabricating memories and logic.

PE: a Processing Element of the Illiac.

PEM: the PE Memory.

PEM approach: a proposed memory system based on a major enhancement to the PEM (see page 95).

RAM: Random Access Memory.

RAM approach: a proposed memory system based on a RAM extended memory (see page 99).

SEC/DED: Single Error Correction, Double Error Detection, usually used on memories.

SU: the Storage Unit in the current Illiac Disk Memory.

Tenex: the operating system for the support system (processor A). also called Tenex A.

TMU: the Test and Maintenance Unit of the Illiac.

Throughput rate: the rate at which programs process their data, from input to the system until output from the system (see page 42).

TNXCM: Tenex Buffer Memory (see page 14).

TRO: the control unit's TMU Output Register.

TRI: the control unit's TMU Input Register.

TSP: Tape System Process for Illiac I4TAPE facility (see page 14).