FINAL REPORT
OF THE FIRST PHASE

ADVANCED NAVAL TACTICAL COMMAND AND CONTROL STUDY

Prepared for
The Advanced Warfare Systems Division
Naval Analysis Group
OFFICE OF NAVAL RESEARCH

VOLUME-V
TECHNOLOGY
PART TWO

Prepared by
informatics inc.

Informatics Report
TR-65-58-2

Final Report of the First Phase of the

ADVANCED NAVAL TACTICAL COMMAND AND CONTROL STUDY (U)

VOLUME V - TECHNOLOGY

(Part Two)

15 January 1965

Prepared for
Advanced Warfare Systems Division
Naval Analysis Group
Office of Naval Research

under
Contract Nonr-4388(00)

by

Informatics Inc.

GENERAL PREFACE TO ALL VOLUMES OF THE FINAL REPORT OF THE FIRST PHASE OF ANTACCS

The first phase of the Advanced Naval Tactical Command and Control Study (ANTACCS) is complete. A final report of the first year's work is presented in five volumes of which this is Volume V. This Volume is presented in two parts. Part One consists of Sections 1 through 6. Part Two contains Sections 7 through 10 and Appendices A, B and C. The five volumes are:

Volume 1	Summary Report; a review of the total study to date, summarizing study findings and giving principal conclusions and recommendations. Provides an introduction to all other volumes.
Volume II	General System Requirements; develops for system planners, details of command and control needed to meet the anticipated threat with the anticipated Naval force posture of the 1970-1980 period.
Volume III	Integration; uses system concepts developed in Volume II to give a planning example by analyzing command and control needs of a Task Force Commander, showing how technology (Volume V) and methodology (Volume IV) can be applied to meet his needs.
Volume IV	Methodology; analyzes planning tools for system design and evaluation and interprets their use in planning tactical command and control systems.
Volume V	Technology; collects for system planners basic information on current and projected electronic data processing and display technology of importance to the improvement of tactical command and control.

ANTACCS is a continuing study to assist planners of the Navy's tactical command and control system of 1970-1980. It is sponsored and directed by the Office of Naval Research and is supported by the Bureau of Ships and the U.S. Marine Corps.

The overall program is directed by Mr. Ralph G. Tuttle, the ONR Scientific Officer. The program benefitted from the assistance of a Study Monitor Panel consisting of representatives from:

Bureau of Ships
Bureau of Weapons
Naval Command System Support Activity
Office of the Chief of Naval Operations
Office of Naval Research, and
United States Marine Corps

The first phase of the study was carried out by Booz Allen Applied Research, Inc. and Informatics Inc. from January 1964 through January 1965. Booz Allen Applied Research Inc. prepared Volume II and supplied parts of Volume I. Informatics Inc. prepared Volumes III, IV, and V, and the rest of Volume I.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont.)

LIST OF ILLUSTRATIONS (Cont.)

LIST OF ILLUSTRATIONS (Cont.)

LIST OF TABLES

LIST OF TABLES (Cont.)

Section 7
MACHINE SYSTEMS ORGANIZATIONS

7.1 INTRODUCTION

The most significant advances in computer usage currently and in the future will result from the interaction of:

1) Improved componentry
2) Formulation of improved methods of problem solution, and
3) Improved organization of the physical component elements.

Previous sections have treated the first subject in detail. The second area of investigation is not within the scope of this report. This chapter is concerned with the item No. 3.

Computer system organization involves the topological configuration of larger components as well as the internal organization of smaller elements. It deals with the design of the components of a computer system and their relationship to each other with respect to capability, communication and synchronization. It is an extremely important subject in computer technology, since it is the opinion of many experts in the field that greater design advances will come through logic organization improvements rather than through hardware component improvements. Computer system organization is an especially important consideration to designers of advanced command systems because of the stringent requirements and inherent complexities of such systems.

In the evolving field of computer system organization, two trends are evident-- the trend toward modularity and the trend toward parallelism. The case for modularity is a strong one. It is so nearly self evident that modularity is a desirable characteristic in system design that to suggest otherwise would border on heresy. Modularity provides flexibility and structural freedom for incremental extension in case of an expanding requirement and it is an important feature to provide reliability (limiting the need for redundancy) and maintainability. A system with modular characteristics tends to optimize more easily and resists obsolescence.

The advantages of using parallelism in a system are less clear. There are attendant difficulties relating to increased complexity and coordination, and precedence problems which are sometimes imposed as a result of parallel processing. However, it appears that research effort is shifting from emphasis on monolithic concepts. There are inherent limitations of flexibility, reliability and capability in this approach. A degree of physical decentralization and functional disbursement is increasingly evident in system design.

It is assumed that the trends toward modularity and parallel processing are well established and valid, but not fully developed. The resultant advantage of modularity and parallelism may be summarized as those of flexibility and adaptability. Changing tactical situations for ACDS will call for increased speed of response and ability to reconfigure flexibly.

The first subject to be discussed is that of multi-computer systems. The advantages of multi-computers are seen in the areas of reliability, cost, maintenance, and parallel processing capability. Definitions are presented and existing hardware configurations are described. These definitions are sometimes arbitrary but are in line with current usage. It is noted that the discussion of highly parallel computers to be considered in a later section is an extension of the multi-computer system topic.

Future plans for the acquisition of computer systems by the Navy for ACDS should be heavily influenced by multicomputing considerations. The use of multicomputers represents one of the most promising approaches to achieving more processing capacity by introducing parallelism.

Command and control systems are especially amenable to the application of multi-computers. The changing patterns of the data processing load related to the threat or the military situation is especially suitable for multi-computer approaches. An important advantage of multi-computers is the reliability which can be achieved through the module redundancy rather than total system redundancy. Improved performance is also obtained by the automatic scheduling and sharing of peripheral equipment among the processing units.

An important aspect of the multi-computers is the design of executive control programs. Executive systems for achieving program control should be given priority consideration in the design of multi-computer systems.

The next subsection discusses the development of associative memories. Considerable emphasis is placed on this topic since it seems likely that this development will prove very useful in the time period of interest. The use of associative memories as the main memory element in a large system is a doubtful prospect, but for use as a peripheral component, associative memories appear to be promising. Here again the interrelationship between subtopics is noted. The inherent parallelism provided by the parallel search characteristics of associative memories may, by logical extension of their capabilities, relate to the study of highly parallel computers. Further, it is seen that the capability for parallel processing may be obtained by application of microprogramming techniques similar to those used with stored logic machines.

The associative memory may find a place of special usefulness in the future ACDS environment. In general, the associative memory is useful when fast or immediate access to data is needed for which comparison criteria are available. The technique of retrieving data associatively, based on one or more search criteria, is intended to replace the costly function of table search.

The system planner should be alert to the unique logical capability afforded by associative memory techniques. Associative memories have potential application for several areas of particular interest for future tactical systems. Among these are: data correlation, sorting, data retrieval and systems programming. Although the associative memory is not without drawbacks, (primarily of cost), the possibilities for limited use in conjunction with other memory elements is worthy of consideration.

Stored logic computers and microprogramming techniques are discussed next. The stored logic design has fallen from favor in recent years due to a number of disadvantages which became apparent in the course of events. However, there appear to be a number of valuable design concepts which were associated with early designs and which are currently being exploited in the design of new computers. Stored logic may be viewed from the standpoint of the programmer and from the standpoint of the logic designer. It is the latter viewpoint which is receiving continuing emphasis and attention.

Stored logic should not be the foremost criterion in the selection of a machine for future Navy use. Further requests for quotation on future computers should neither demand it nor exclude it. Rather, they should encourage flexibility and adaptability from the logic and programming points of view and let the designer-manufacturer develop an approach best suited.

There is an increasing use of memory hierarchies in computers and computer system design. The usual method of memory classification is according to access speed and storage capacity. The ideal hierarchy is usually described as one with a fine gradation of speed and capacity characteristics from small amounts of very high speed storage (register, scratchpad storage) through high speed memory main storage, through successively larger amounts of lower speed storage, and finally, to large amounts of bulk storage such as that provided by discs or drums.

The trend toward use of multiple memories of various speeds and capabilities may be expected to continue. One could anticipate a system design encompassing a wide variety of such memory types each employed according to the special capacities afforded.

The system planner must not only be aware of the various kinds of memory at his disposal, but must be alert to the special capabilities afforded by the combinative properties of hierarchies of memories. Also included are the topics of read-only memories and the fixed plus variable computer organizational concept.

The next subsection deals with novel computer organizations which are usually described as highly parallel computers. The highly parallel computer organization represents an almost complete departure from conventional computer design. In these computers, the arithmetic and control logic is decentralized to the extent that it exists at nodes of a network. All of the arithmetic and control units at the nodes may operate in parallel to provide, in theory, a high speed operation. There is some question whether the significant capabilities of this organization can be effectively used for command systems in the future. The description of these novel organizations is limited in scope according to evaluations of potential applicability to advanced command systems.

The final subsection outlines important computer design parameters for ACDS applications, indicates present trends and specifies areas for further study. A consideration of ACDS general computer requirements is followed by a review of present NTDS compatible computers.

One of the central objectives of computer system organization is to arrive at a series of recommendations on the characteristics of a possible future NTDS family of computers. The basic computer is considered, as a member of a family of computers, in a time sharing environment, and in a multi-computer configuration.

Note that in choosing systems for description in this section the choice has been made on the basis of reflecting different forms of multi-processing or multi-computer organization rather than to present the latest information on new systems. For example, it is not possible to include the very latest information on developing systems such as the IBM System/360.

7.2 MULTI-COMPUTERS

7.2.1 Introduction

One of the major areas of interest in system organization is the subject of multi-computers. The use of multi-computers represents one of the most promising approaches to achieving more processing capacity by using parallelism in system design. Recent years have seen a significant trend from the concept of a single large central computer as the main system element toward a decentralization of computer elements. This trend is consistent with the increasing emphasis on modularity and parallelism noted earlier.

Among applications for which the use of multi-computers have been found particularly advantageous are those with these characteristics:

1) Those systems in which the I/O demands of the system measured as a percentage of overall system capacity are unusually high .
2) Those in which there is a geographic separation of locations at which information is to be fed to or received from the system.
3) Those in which the system must respond to varying traffic rates of input and output messages.
4) Those in which the system must continue to accomplish high priority tasks even though it is not completely functional.
5) Those in which a number of diverse problems or tasks must be handled simultaneously.
6) Those in which real time response capability is an important requirement.
7) On line systems.

Multi-computers represent an increasing segment of total installed hardware. This is particularly the case in command and control systems, both tactical and strategic.

Designing multi-computer systems is not an exact science. The possible interrelationships of elements are highly complex. Such efforts call for the traditional disciplines of digital logic design, communications, switching theory, programming, and related skills.

7.2.1.1 Definitions

A multi-computer system is one which is composed of a number of computing elements including two or more separate processors capable of operating programs simultaneously. The term multi-processing is used to describe application of a number of computers all executing different jobs, or parts of a single job, under control of a single executive program. It is noted that these definitions suggest more than simply duplexing for reliability (as in SAGE). Also, computers operating independently on separate tasks (even though colocated) do not qualify under the definition.

Multi-programming is the process of using one computer for different processing tasks on a time shared basis. Time sharing is a special case of multi-programming wherein the several jobs being run concurrently are originated by different users. It is getting a great deal of attention, especially among scientific users who wish to have many problem solvers using the machine simultaneously. Applications in command and control could involve many analysts using the machine simultaneously. To some extent, multi-programming supplies a motivation for multi-computing. There are advantages associated with the use of each, and there are reinforcing advantages to the use of a combination of both concepts.

7.2.1.2 Characteristics of Command and Control System

The trend toward multi-computer systems is particularly evident in military systems. In command and control systems, the concept is so well established that it is not a question of whether multi-computers are used but rather the extent to which they are used.

There are a number of characteristics and requirements of command and control systems which motivate toward the selection of a multi-computer design. Some of these are:

1) Reliability - Duplication of equipment is a characteristic of command and control systems. However, the use of multi-computers permits reliability to be achieved through module redundancy rather than total system redundancy. Failure of a component must result in a reduced capability rather than a catastrophic failure of the entire system. The characteristic of continued operation despite component failure is sometimes referred to as graceful degradation.

2) Expansibility - It is frequently found necessary to increase the scope of system tasks or to expand the capacity of the system as a whole. The multi-computer approach provides for incrementally modular expansibility.

3) Need for Parallel Processing - It is often a system requirement that multiple programs be run simultaneously without interference. The use of multi-computers provides the ability of processing multiple tasks simultaneously.

4) Flexibility and Adaptability - Flexibility and adaptability can be achieved on a millisecond basis by realigning tasks for the available equipment, or can be achieved on a month or year basis by adding components to the system.

5) Operational Compatability - The use of multi-computers is compatible with tactical deployment concepts, and provides the degree of functional or physical decentralization desired.

7.2.1.3 Shipboard Use of Multi-Computers

The use of multi-computers aboard ship presents special problems of communication, physical placement, reliability and backup, processing load distribution, and task assignment. The computers selected may be general purpose or highly specialized, similar or identical to each other, or disparate in size or capacity, (e.g., large primary computers used in conjunction with satellite or peripheral computers). The system design may call for the computer to be assigned individually to accommodate separate subsystems or for shared task assignments and dynamic reassignment of available processing capacity as a function of current task priorities. The computers may be required to be highly interrelated, sharing common inputs, and servicing the same display devices and peripheral equipment. The number of computers and the optimum size to provide the proper degree of granularity for reliability and backup, is, of course, of primary importance. These considerations, and many others, must be examined in detail in the context of system requirements with cognizance of the most advanced multi-computer usages.

7.2.2 Classifications of Multi-computers

Multi-computers can be classified according to a number of criteria. Categorization by assignment of function and topology of communication is described by Bauer.[1]* Other classifications could be made on the basis of machine capabilities or by the number of processors in the system. Comfort[2] selects a classification of multi-processors by order. From 2 to 9 processors constitutes a multi-processor of Order 1; from 10-99 processors, Order 2; 100-999 processors, Order 3, etc. A multi-processor of Order 3 or more he classifies a "highly parallel machine."

Another classification is according to whether the processors are all identical (e.g., D-825) or incorporate a gradation of capability (e.g., IBM 7090/1401, CDC 1604/160A Systems). Yet other distinctions involve executive concepts, principles of load sharing, and scheduling control. Multi-processors may be discussed according to how they interrelate with peripheral equipment and memory units.

Typically, multi-computer system organization involves:

(1) Interaction of processors with each other

(2) Interaction of processors with peripheral or satellite equipment

(3) Interaction of processors with memory units

*References are collected in Section 10.9

7.2.2.1 Interaction between Processors

Two basic approaches to the organization of multicomputers based on assignment of functions are described in the Bauer paper (See Figure 7-1). The hierarchy arrangement is that in which one or more central processors carry out the main processing load, and a number of special processors carry out subordinate tasks. A distributed arrangement has two or more processors of approximately equal power and responsibility in the system.

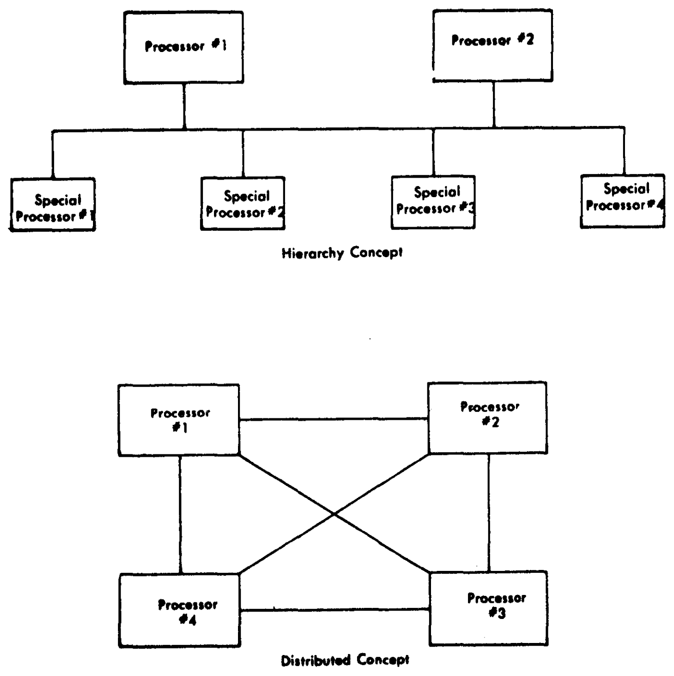


Figure 7-1. Assignment of Functions

7.2.2.2 Interaction of Processors with Peripheral Equipment

A categorization according to topology of communication is discussed and illustrated in the Bauer paper. The primary types of communication are the substation approach and the centralized communication approach.

Substation communication uses a level of principal processing and a level of switching and communication. All communication with peripheral units is handled through communication with the substation. No highly-specialized or unusual devices are required for switching. Usually switching can be done by small satellite computers. But if the switching requirements become high, these satellite computers become expensive and awkward to handle.

The centralized communication approach routes all communication through one central switching unit (See Figure 7-2). It results in a flexible system and provides high total reliability, since only the switching unit is critical to system operation.

Switching can be by information bus or by central switch. In the information bus method, communication must rely on specifying, coding, and addressing, or on dedicated time slot techniques. All techniques are conventional and relatively inexpensive for a small number of modules.

The central switch is a crossbar type similar to those used in telephone central. For many modules the cost is low, and high communication rates through the switch are possible since communication paths are independent. However, few electromechanical switches of this type have actually been made for computer use. The original crossbar switches developed for telephone purposes were electromechanical. Those used in computers have employed transfluxor magnetic cores or diodes and gates.

The information bus can be further classified as a multiple bus or a time shared bus. Separate buses connect the processor to one or more memories in the multiple bus system. This is a fairly inflexible system. In the time-shared bus system, each module time shares a bus. This is the lowest-cost system but it suffers from control complexity problems.

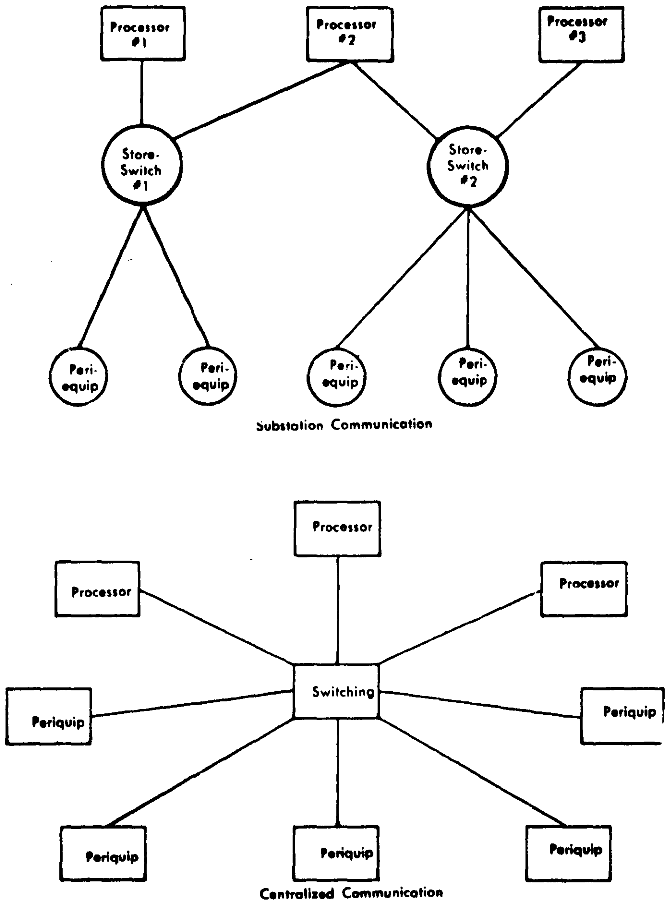


Figure 7-2. Topology of Communication

7.2.2.3 Interaction of Processors with Memory Units

The tendency to separate the primary computer components into distinct functional categories provides numerous combinative possibilities for system manipulation. In particular, the relation of computer memories to associated processors is varied. The question of shared vs. private storage, the types of memory divisibility, and memory protection features, are all important considerations.

The shared memory concept is illustrated in Figure 7-3(A). One of the early memory sharing systems is the GAMMA 60 system. It consists of arithmetic processors, I/O units, logical processors, and a common memory.

The configurations shown in Figure 7-3 (B and C) are logically similar but suggest slightly different implementations. When a complex switching network is employed, it is necessary to have a control unit (perhaps a small computer) control the switching functions. It is sometimes suggested that the capability to switch memory units from one processor to another amounts to an extremely high rate of data transfer. This capability also has reliability significance since it is possible to switch an entire memory from a failing processor to an operational one. Figure 7-3 represents schematically the concept of shared and private memory contained in the same system. It should be noted that the question of single vs. multiple memories is to some extent arbitrary since a single large memory may be regarded as being composed of separate logical memories. The basic functional distinction of private vs. shared memory involves the ability or inability for a given processor to address a particular segment or unit of memory in the system.

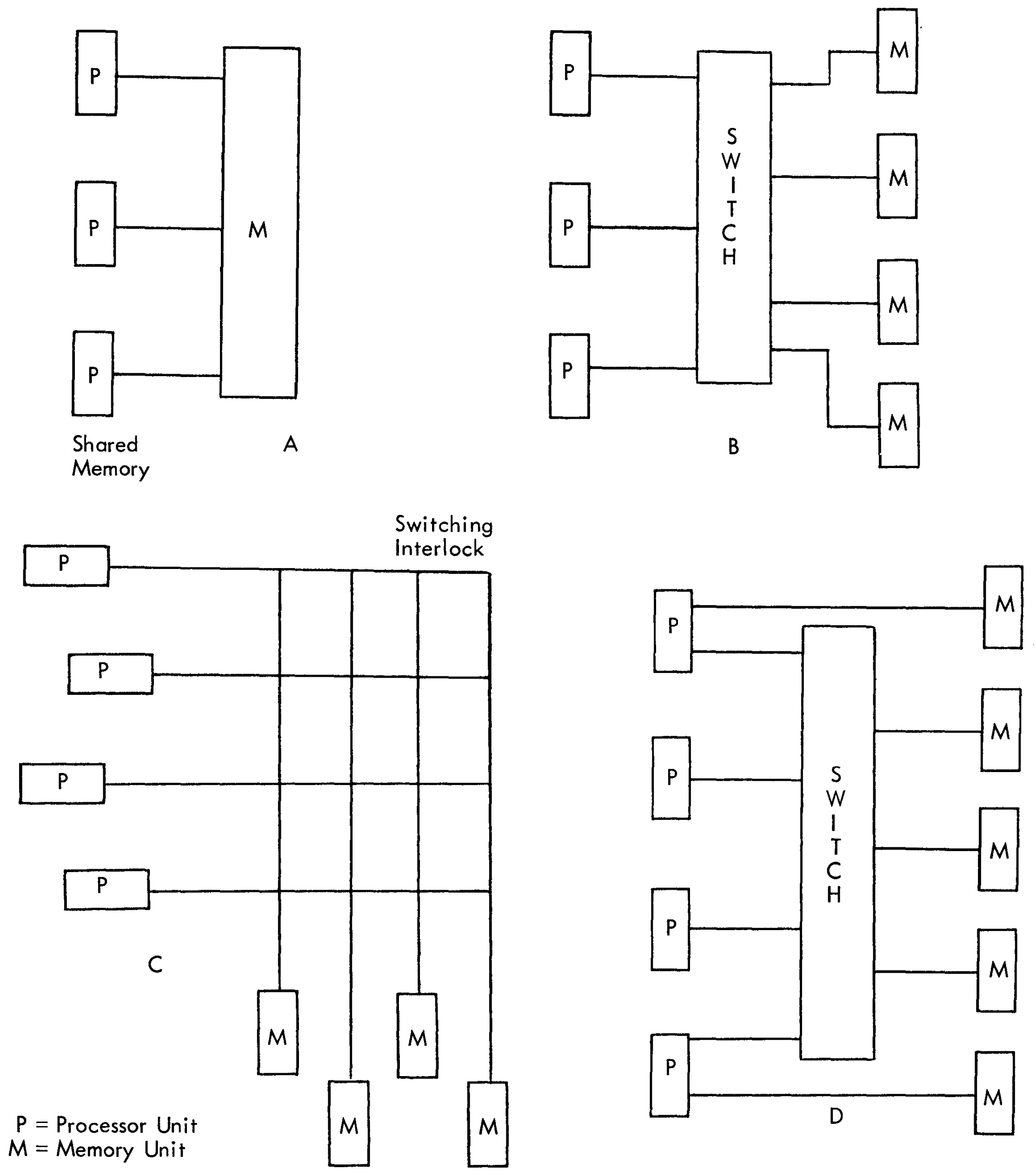


Figure 7-3. Processor and Memory Organizations

7.2.3 Multi-Computer Systems

7.2.3.1 RW-400 System

The RW-400 multi-computer system is built around an expandable central exchange to which a number of primary modules are attached. The central exchange is a switching center. Computing and buffering modules provide control for the system. These consist of computer modules and self-instructed buffer modules. They are self-controlled, and allow completely independent processing of two or more problems.

The system operational concept is that, although each of the computer modules is identical, one is designated the controlling module. In this designated module the master control program resides. The master control program initiates all operations on the basis of priority. It gives jobs to computer modules and assigns certain slave modules in groups to the computer modules. An alert-interrupt network allows coordinated system action. This makes it possible for the system to change on a microsecond basis to provide a programmable self-organizing aspect to the system. The intention is that the most efficient and economical complement of equipment should be applied to the problem at all times.

The RW-400 computer is centered around a high speed cross bar type switch. The Y axis of the switch represents "spigots" onto which controlling modules are tied. The X axis represents spigots onto which slave units are tied. The kinds of slave units tied to this system are printers, readers, plotters, tapes, input buffers and output buffers.

Another configuration of the RW-400 uses a CDC 1604. This system is shown schematically in Figure 7-4. A master control program written for the 1604 controls the system.

Perhaps the most interesting part of the system is the switch. The switch is 16 x 64 (controlling axis and slave axis respectively). Connections can be made by any one of the controlling modules to any one of the slave modules in approximately 70 microseconds.

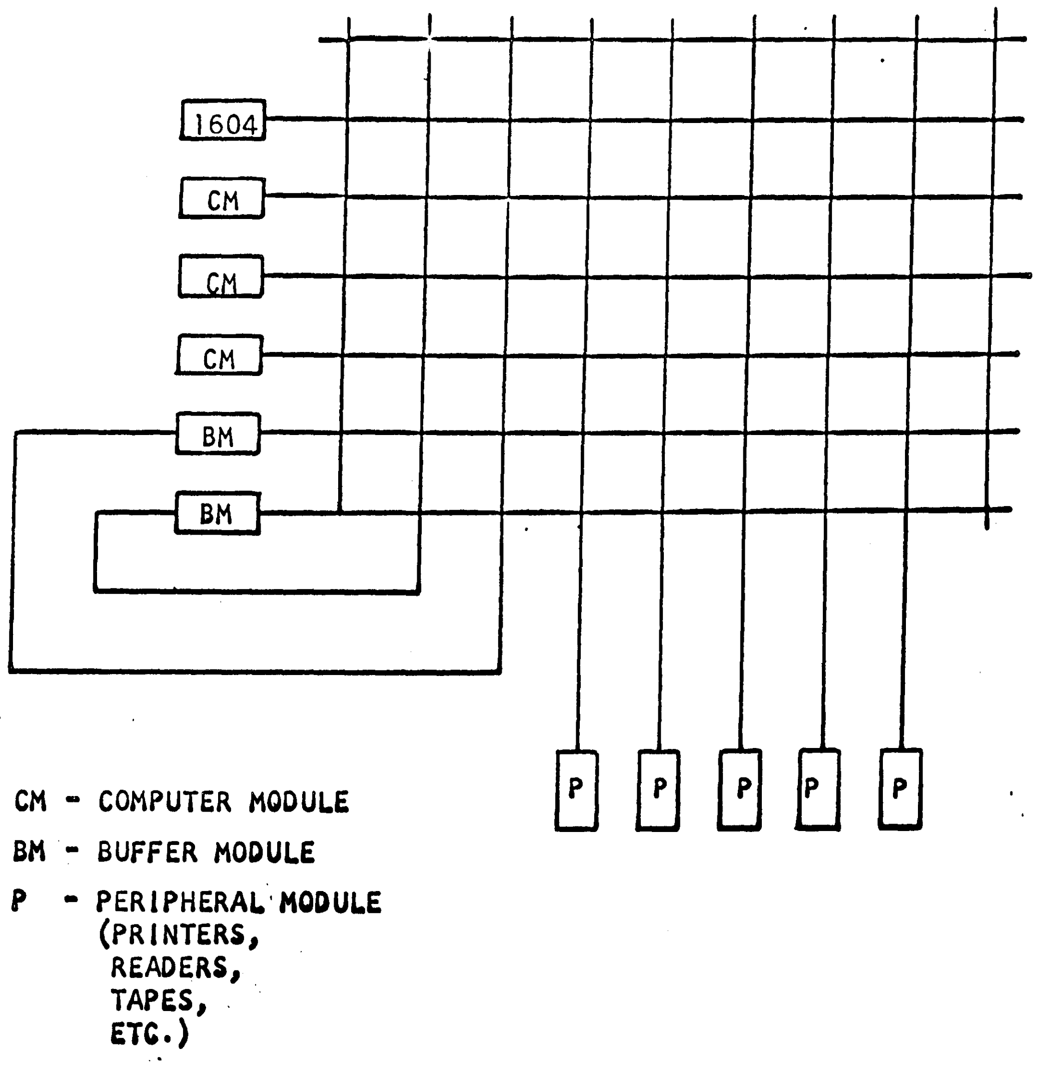


Figure 7-4. Ramo-Wooldridge RW-400

Two switches were made for this system, one uses transfluxors and the other diodes. The switch has some interesting and advanced features. The switch itself has a core memory of some 2,000 words. The core memory stores data on the connections which are made or could be made. The programmer can disallow certain switch points under program control, thus facilitating the master control of the system. Another feature is the "symbolic address file" which allows symbolic addresses to be assigned rather arbitrarily to the various hardware spigots. A comprehensive set of alerts and interrupt lines is also possible through the switch.

An important aspect of this system is that each module is self-contained. That is, each module has its own power supply and may be tied to the switch with little difficulty. This has the advantage of expansibility but the disadvantage of the high cost of each of the modules. Although not part of the first RW system, a group of peripheral equipment can be attached to one spigot.

Connected to the switching center are magnetic tape modules, magnetic drum modules, peripheral buffer modules, and console communication and display buffer modules. There are also modules for punched cards, punched tape, high speed printing, and control consoles to handle the input/output requirements of the nominal system. Additional man/machine communication devices for interrogation, display, and control operations can be added as required.

The functioning of an RW-400 multi-computer system depends on the number and type of the modules, and the number of modules in a given system is governed by the system application. Initially it may consist of the minimum number and types of modules, to do a small problem solution. In this configuration it would work very much like a conventional computer. Later this system could be expanded by the addition of modules as needed. Modules may be assigned to work together on a problem in proportion to its needs. As soon as a module's function has been completed, it may be released for assignment to another task. The system is thus self-controlled, matching processing capacity to each problem for the time necessary to do the job. Full system capability can be brought to bear upon very large problems when the need arises.

The RW-400 data system allows the expansion of the system from a modest initial installation into a powerful and comprehensive information processing center. System obsolescence is reduced by the expandability in both the number and types of processing modules. The ultra-high systems reliability is a feature of the system because of the multiplicity of common elements. Parallel processing and parallel information handling modules increase the system's speed and adaptability when handling complex computing workloads.

The system was designed for a specialized military application and is now in operation at the Rome Air Development Center Experimental Computer Complex.

7.2.3.2 Burroughs D825

The D 825 is a military system oriented toward the command/control problem.[4] (See Figure 7-5). Its modular design allows a building block approach to applications design. The simplest configuration is a single computer module, one memory module, one input/output control module, and peripheral devices. A fully expanded system would consist of four computer modules, 16 memory modules, ten input/output control modules, and 64 peripheral devices. The processor or computer modules share access to the executive program, to each other, and to all memory and input outputs.

The interconnection switching is termed a switching interlock by Burroughs. This switching function is distributed among the various modules which are then interconnected by cabling. Accordingly, a failure in the switch hardware can be handled in the same way as a failure in a module.

The executive routine used with this multi-computing system was developed to provide an automatic control framework for efficiently and effectively running multipath, parallel, real-time programs. The executive program maintains a job file for all programs currently in the system. One element on the file is an "image" of the job, which is transferred to the thin-film registers of a processor when assignment of that job to that processor is made by the executive routine. This image is either the initial data needed by the processor to begin the job or it is data retrieved from the thin-film registers of an interrupted processor, giving the status of the job at that instant.

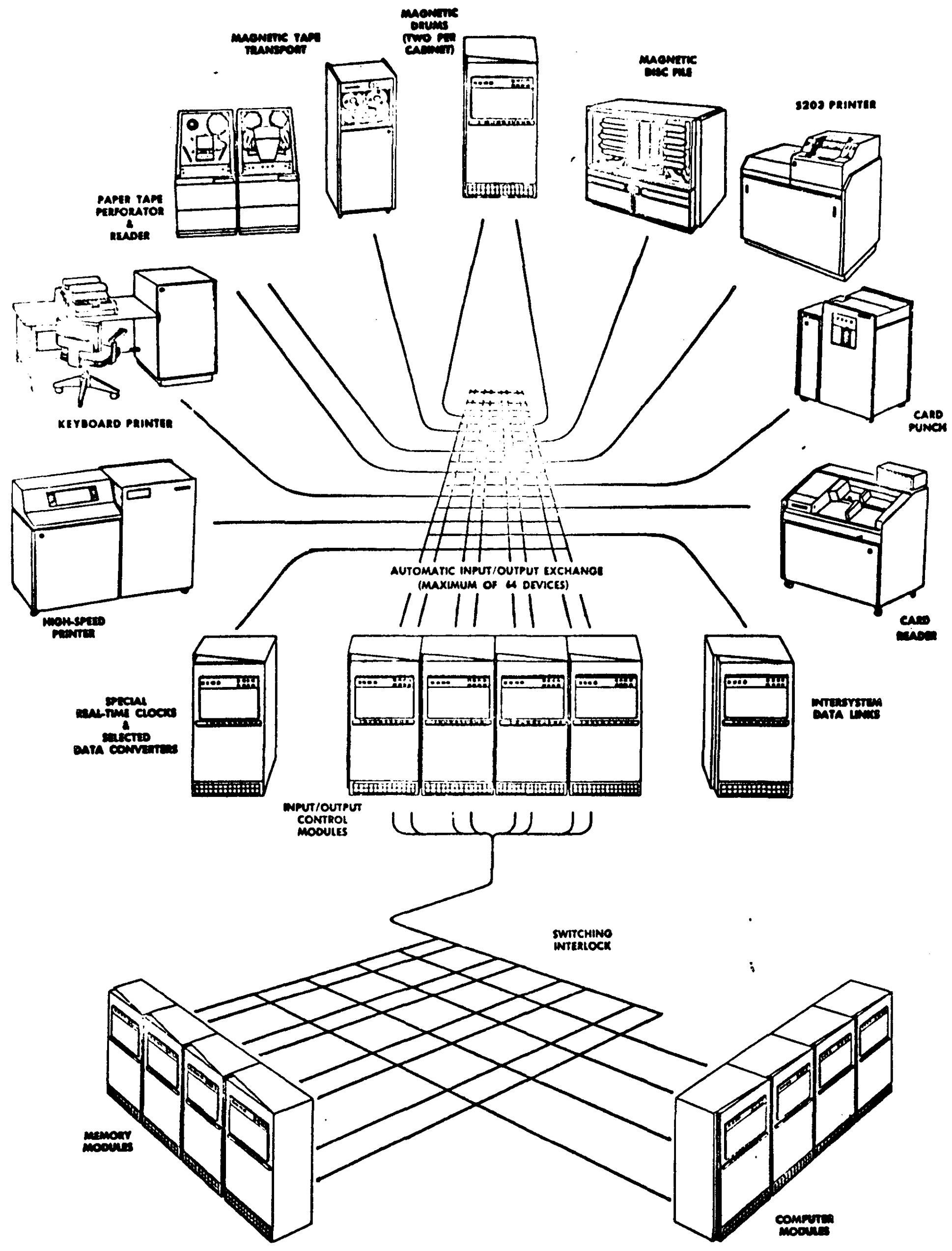

MAGNETIC TAPE TRANSPORT
MAGNETIC DRUMS (TWO PER CABINET)
MAGNETIC DISC FILE
5203 PRINTER
PAPER TAPE PERFORATOR & READER
KEYBOARD PRINTER
CARD PUNCH
HIGH-SPEED PRINTER
CARD READER
AUTOMATIC INPUT/OUTPUT EXCHANGE (MAXIMUM OF 64 DEVICES)
SPECIAL REAL-TIME CLOCKS & SELECTED DATA CONVERTERS
INTERSYSTEM DATA LINKS
INPUT/OUTPUT CONTROL MODULES
SWITCHING INTERLOCK
MEMORY MODULES
COMPUTER MODULES

Figure 7-5. System Organization, Burroughs D825
Modular Data Processing System

When a processor seeks a new assignment because it has completed a program or has been interrupted, it runs the executive routine. If it is assigned to that program it transfers the image of the program from the main memory to its own thin film registers. If this program is again interrupted the processor transfers the program, revised to the new status, back to the executive routine to be drawn out again as another processor becomes available. Thus, programs or program segments are shifted about from processor to processor to permit direct response to the various interrupts in the system. Programs are never associated directly with any individual processor. Moreover, processors need not even be aware that they are picking up a partially completed job.

The executive routine may also be used by the programmer to divide a program among several processors for simultaneous parallel solution of the separate elements.

The D825 system has a comprehensive interrupt system with eight internal processor generated interrupts and four external interrupts. The automatic interrupt capability is fairly extensive and includes protection features such as reflecting (by interrupt) when an attempt is made to write out of bounds, and a real-time clock overflow. The control structure permits immediate and automatic entry of a priority program without damage to interrupt programs. It also permits rapid, simple addition to or substitution of hardware or executive routines within the system structure.

The D825 is the expanded and militarized forerunner of the Burroughs B 5000 Computer, and is to be used as the real time computer for the Back-Up Interceptor Control (BUIC) System of SAGE.

7.2.3.3 Burroughs B-5000

The B-5000 (See Figure 7-6) achieves physical and operational modularity through the use of electronic switches which function logically like telephone crossbar switches.[5] The Burroughs B-5000, in a dual configuration, is a standard B-5000 computer complex to which is added a second processor, usually designated Processor B. Both processors are connected directly into the memory exchange and hence have equal access to memory modules or to I/O channels. Since the B-5000 uses independent memory modules, both processors may work continuously provided they are working from different modules.

Processor A (and either physical processor may be logically so designated) is the only one that may work in control state. This is a state determined by hardware in which Processor A executes the master control program functions. In this machine the master control program is kept permanently stored on drum, and is called into core at the initiation of processing.

Processor B may not work in control state, but may halt and interrupt Processor A at completion of any operation, need for I/O, etc. Hence, Processor A schedules and controls Processor B, endeavoring to keep it and itself busy.

Design criteria of the B-5000 call for separateness of input/output operations from processing operations and generalized handling of indexing and subroutines. Multi-programming and parallel processing by the multiple processors are both featured. The design is said to have been conceived from a joint hardware/software standpoint and assumes that the user will use higher level (compiler) language to the virtual exclusion of machine language programming.

7.2.3.4 CDC-3600

The Control Data 3600 system incorporates a modular approach which provides many possibilities for multicomputer usages.[6] The design maintains

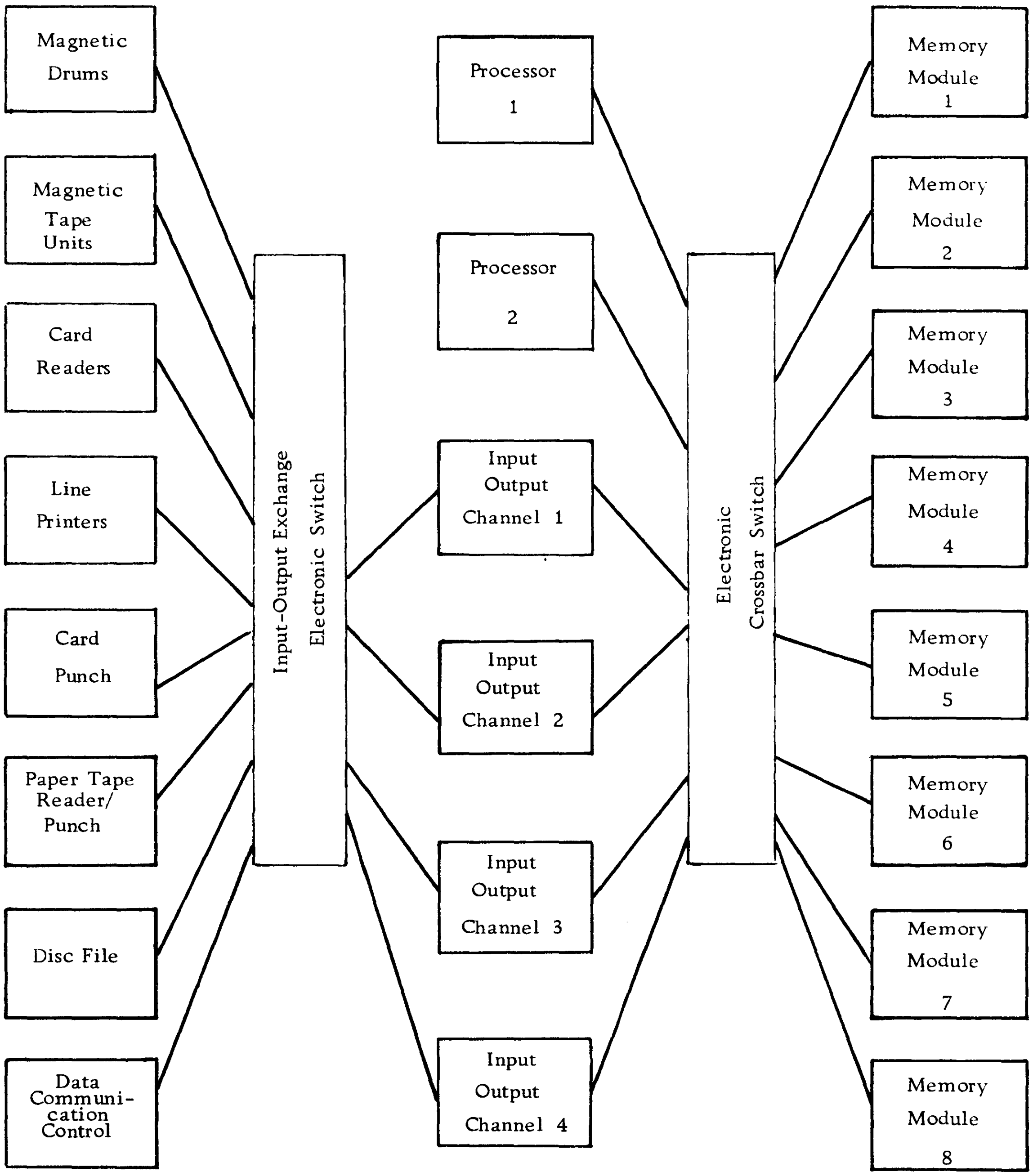


Figure 7-6. Organization of the B 5000 System

compatibility with the CDC-1604 (programs written for the 1604 may be executed on the 3600 system) and inter-computer communication may be achieved between these two computers.

The 3600 system is constructed of functional modules: The Computation Module, Communication Modules, and Storage Modules. These modules may be arranged in a variety of systems configurations.

A simple system consists of a computational module, one or more communication modules, and one or more storage modules. Such a system is shown in Figure 7-7. This figure illustrates a maximum simple configuration.

A five-way switch is built into each storage module. Processors access the storage modules on a first come first serve basis. The storage modules contain priority scanners which scan the five access channels for storage results. It is noted that each storage module consists of two separate banks which may be individually accessed.

The multiple bus arrangement is somewhat less flexible than the cross-bar switch arrangement but results in a lower cost due to the fewer number of switch points.

The modular approach allows for the expansion of a simple system to one with more than one computational module and the construction of multi-computer systems. Various arrangements of shared storage are possible. It is possible to have all storage modules made available to the computational modules as in Figure 7-8 ,(A) or it may be desired that each computational module has direct communication to private storage in addition to sharing common storage (Figure 7-8, (B)).

With the common storage feature of the system design, reliability is enhanced. For example, one computer could be assigned to the higher priority tasks and another could be operating on lower priority tasks or in standby status. The primary computer could turn control over to a standby computer to finish the task in case of an operational difficulty.

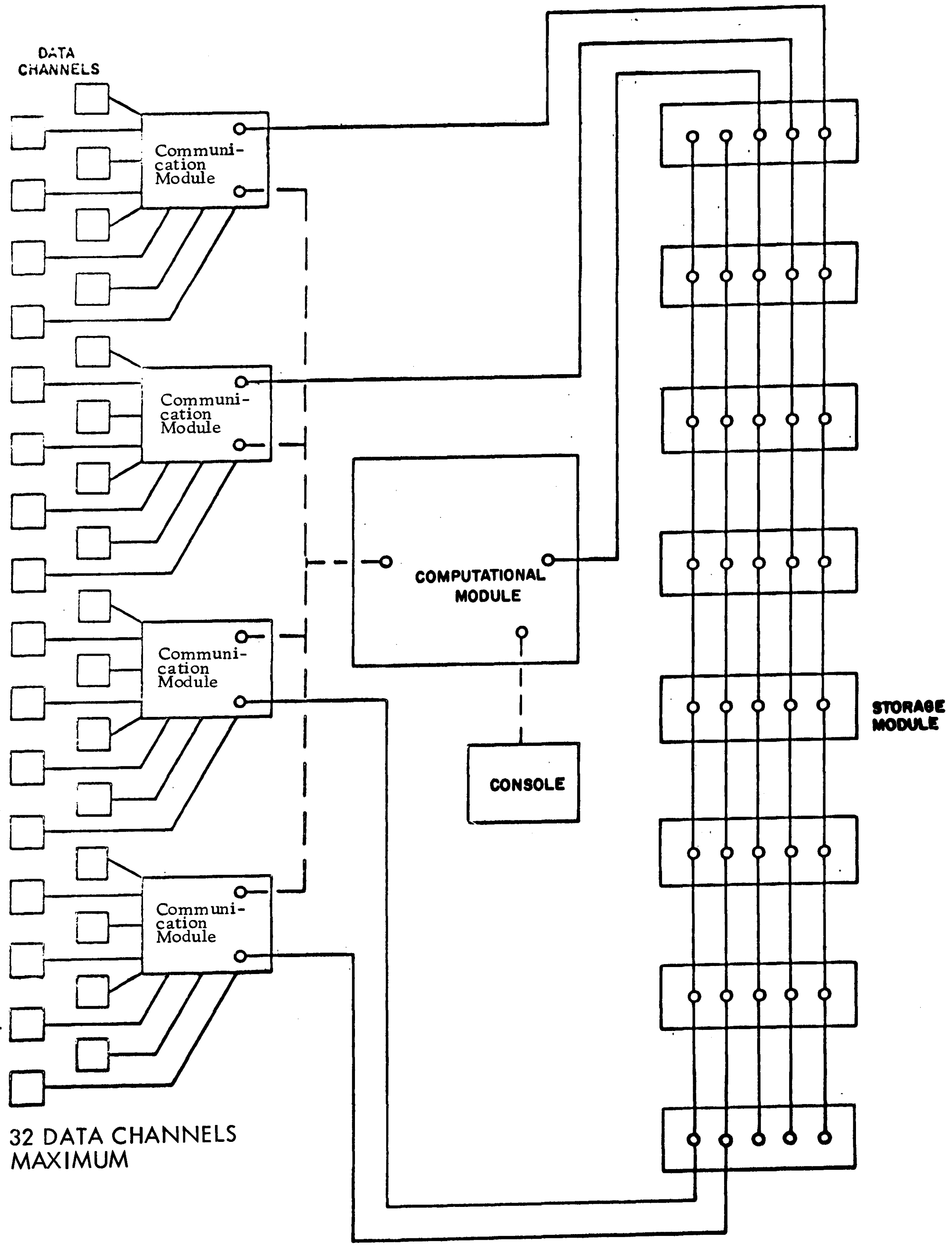


Figure 7-7. Maximum 3600 Simple System

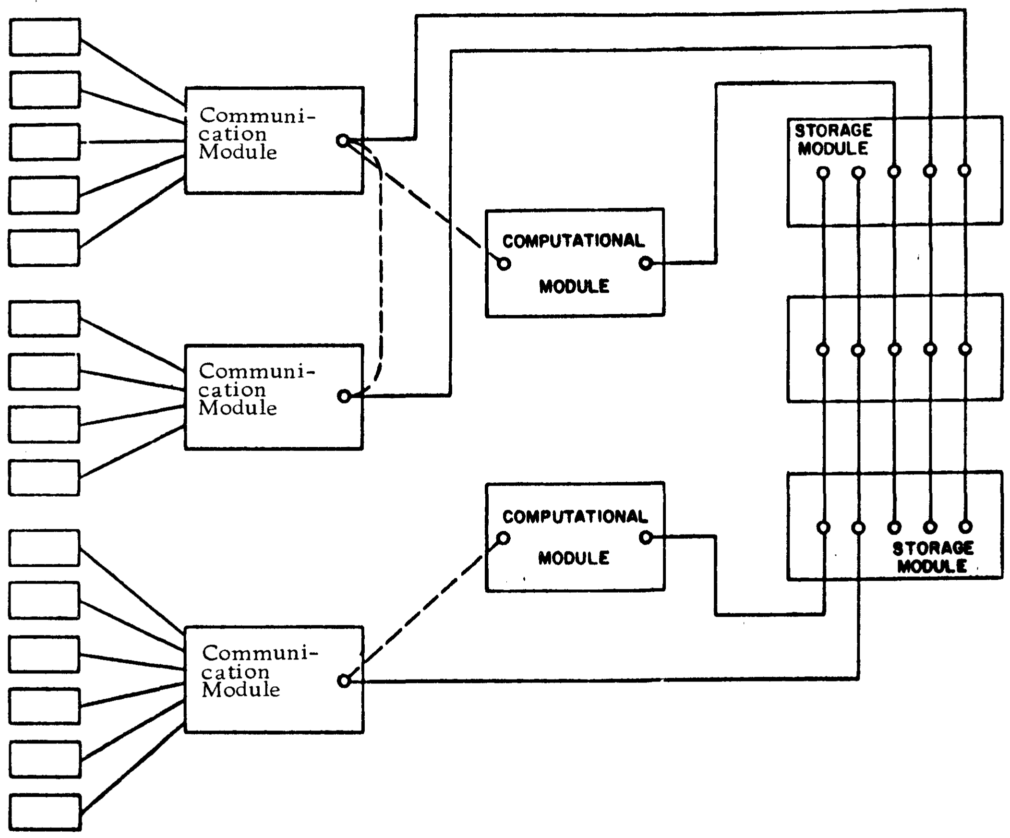


Figure 7-8(A). Two 3600 Systems Sharing Common Storage

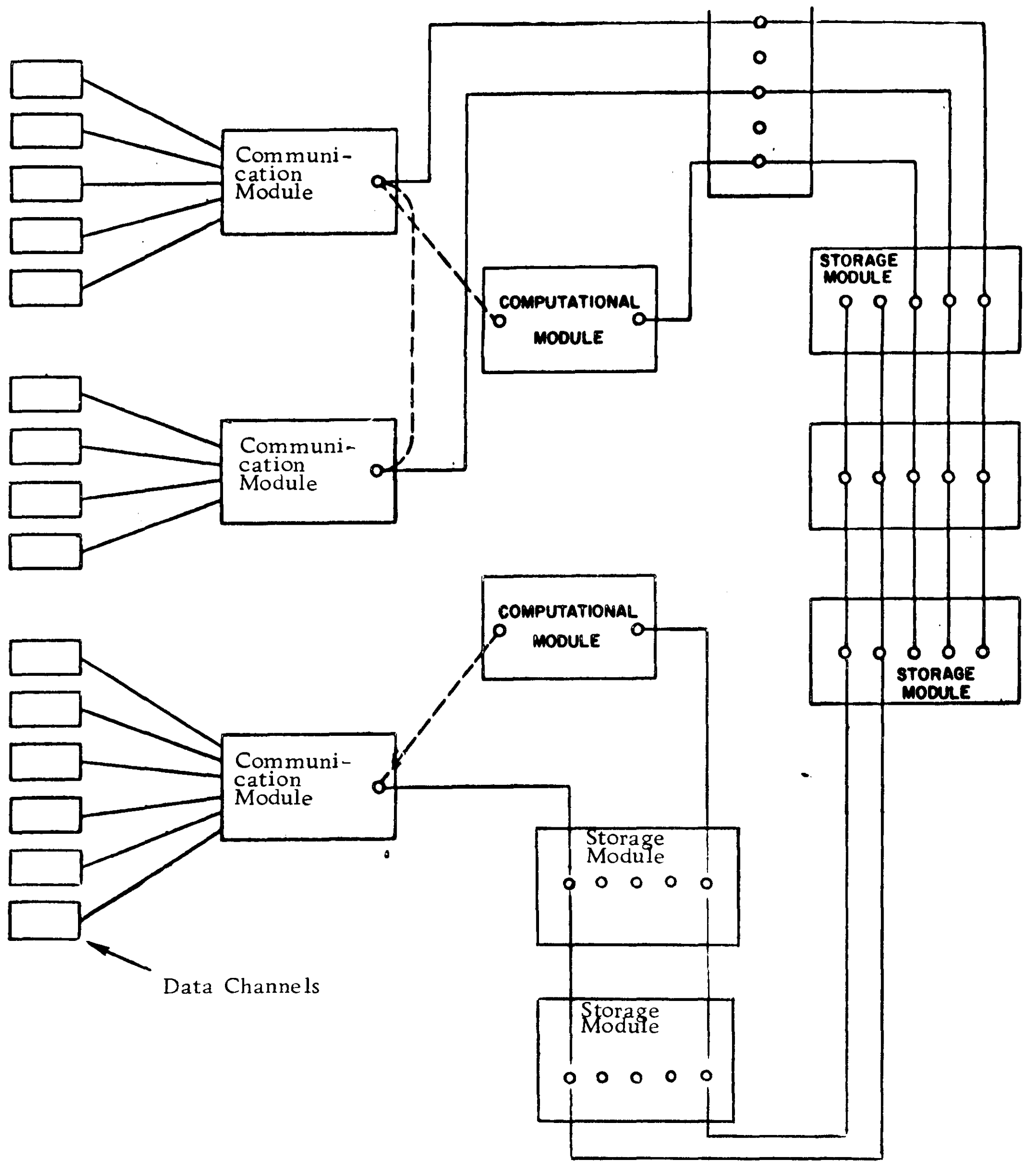


Figure 7-8(B). Two Computer System with Private and Common Storage

7.2.3.5 CDC-1604/160A

A configuration of Control Data Corporation computers on a smaller scale is the 1604/160A system.

Other configurations include the 160A as a satellite to any 3000 Series computer. Also the 3200 computer may be used as a satellite to another 3200 or to a 3400 or a 3600. Or a 3400 may be used as a satellite to another 3400 or to a 3600.

The 1604/160A configuration uses a magnetic tape system to link the 160A to the 1604. Either computer has independent and simultaneous access to the tape system. In the various modes of operation, the computers may be used independently or together in a single system.

The 1607 magnetic tape system consists of four digital tape handlers and a synchronizer control unit. The synchronizer control unit makes use of independent read and write channels. Any tape handler may be operated through the synchronizer read channel, while any other tape handler is being operated through the synchronizer write channel. Synchronizer read and write channels may be assigned to either computer. A direct line from the synchronizer read channel to the synchronizer write channel permits direct transmission of information between the two computers.

The 160A uses a single 12-bit channel for both read and write. The 1604 uses separate 48-bit channels and is capable of reading and writing simultaneously.

7.2.3.6 704X/709X Direct Coupled System

The IBM 704X/709X Direct Coupled System presents an approach to multiprocessing aimed at achieving faster throughput and reduced turn-around time. Division of function between the computer is segregated by relegating one computer to the role of input/output handling thus relieving the other computer from input/output delays. The Direct Coupled System combines the IBM 7090/7094 (704X) and the IBM 7090/7094/7094II (709X) systems.

Direct coupling allows the 709X to be operated with no data channels or peripheral equipment. Each computer is modified slightly to provide the direct coupling capability. However, when not operating as a direct coupled system, both computers may be operated as stand-alone independent systems. The system is intended to operate in conjunction with a mass storage device such as a disc file or a core file. A simplified system configuration is illustrated in Figure 7-9. Each computer has the capability to interrupt the other.

The 704X is provided direct access to the 709X memory and may specify block core-to-core transfers between the 704X and the 709X. It may also request direct transfers from the disc file to the 709X memory (or to the 704X).

In addition to performing buffering and input/output functions, the 704X is usually called on to handle all utility functions, sequencing of jobs, system program loading and other administrative tasks.

Executive functions are performed in the 704X and the executive routine for the system resides entirely in the 704X. This program uses 16K words of storage and is contained also on the disc. The remaining 16K are allotted for I/O buffer purposes.

The Direct Coupled System may be regarded as achieving its purpose in increasing machine utilization and reducing turn-around time. As a multi-computer system, it is not a modular design, and incremental system expansion is possible only on the grossest level. Another difficulty is sometimes encountered in trying to equalize the load between the two computers.

7.2.3.7 IBM/System 360

The design of the IBM/System 360 family of computers permits a wide variety of multi-computer systems. A rather even gradation of computer capability is available with increases of performance of approximately 2.5 to 3 from one model to the next. The largest computer is approximately fifty times as powerful as the smallest one. Even finer gradations are available through combining models in various configurations.

Communication between central processing units in a multi-computer system may be accomplished in a number of ways and at several different transmission rates. There are basically four methods of communication: 1) via shared control units, 2) through a list channel connector, 3) by a direct control feature, and 4) through shared storage. These methods are illustrated schematically in Figure 7-10.

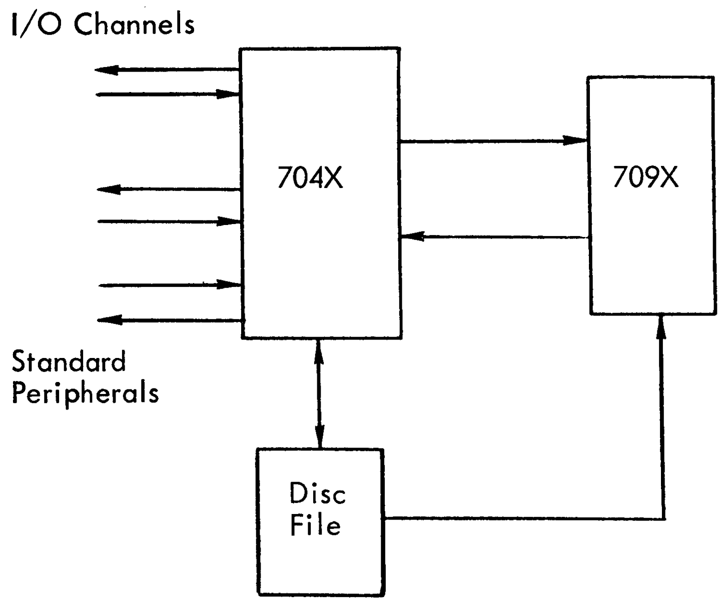
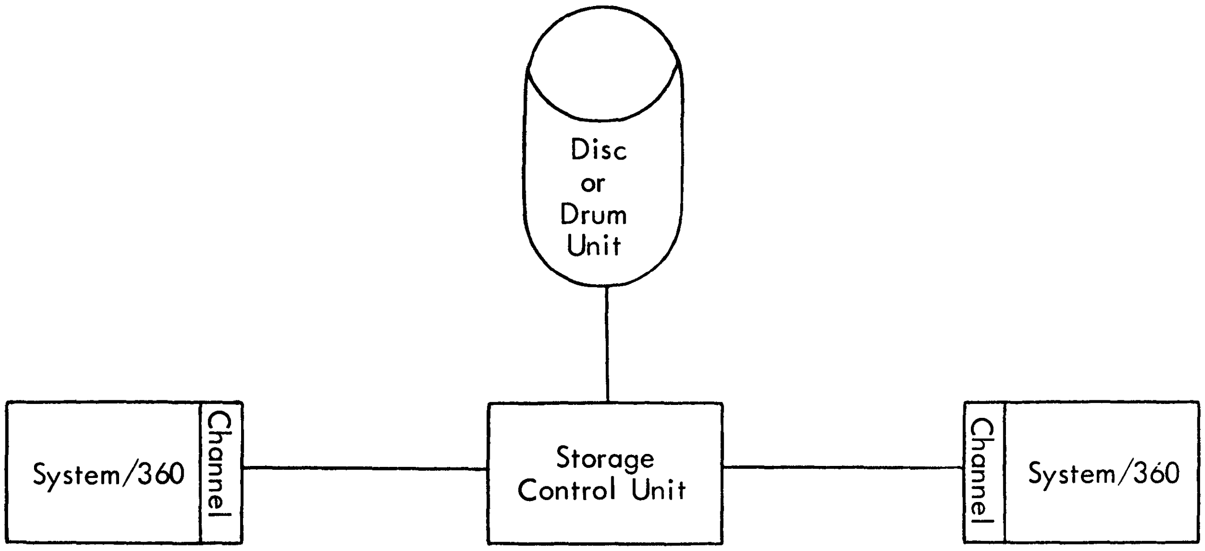
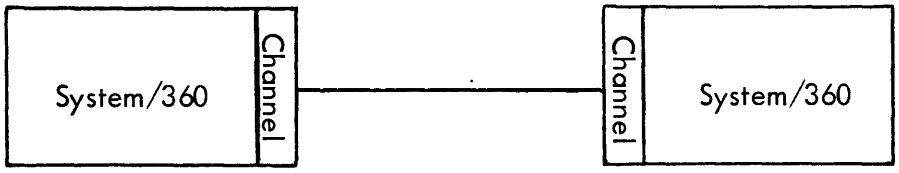


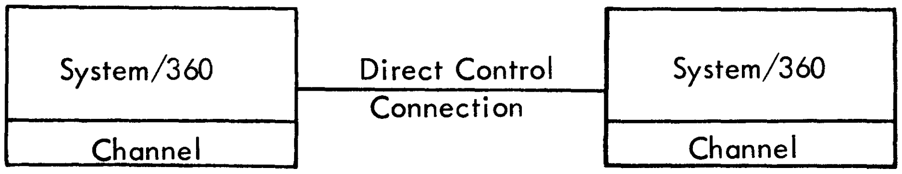
Figure 7-9. IBM 704x/709x Direct Coupled System



Shared Control Unit



Channel-to-Channel Connection



Direct Control Interconnection

Figure 7-10. IBM System/360 Communication Methods

Of the various levels of communication, the largest in capacity, (but only moderately fast in response), is by means of shared I/O device, for example, a disc file. Faster transmission is obtained by direct connection between the channels of two individual systems.

The channel-to-channel adapter allows any channel on any model of System/360 to appear as an I/O device to any other channel of any model of System/360. These channels can be on the same machine or on different machines.

The direct control features eight input and eight output lines which may be used in a manner analogous to the sense lines of the direct data feature of the 7090. This feature also has external interrupt capabilities and may be used to transfer control signals in a multi-system complex.

Finally, it is possible for storage to be shared between processing units, making information exchange possible at storage speeds.

The IBM 2361 Core Storage (See Figure 7-11) provides from 1 to 8 million characters of core storage in addition to the regular core storage on the Models 50, 60, 62 and 70. It may be shared between any two of these models of System/360. It is also possible for multiple 2361 boxes to be available to one processor while some of the boxes are available to a second processor and the remainder to a third processor.

A design feature of the System/360 Model 50 allows two Model 50 processors to be connected so that the full complement of addressable core storage of each processor is available to either, both for processing or input/output.

The various modes of communication are supplemented by an interrupt feature which permits one processing unit to be interrupted by another, and makes available status information from one processor to another. To facilitate control of multi-system configurations, the operator control panel of each processor may be remoted to a common master console.

IBM has selected no single multi-computer configuration for special attention. The flexibility possible allows each user to tailor the equipment to his specific needs.

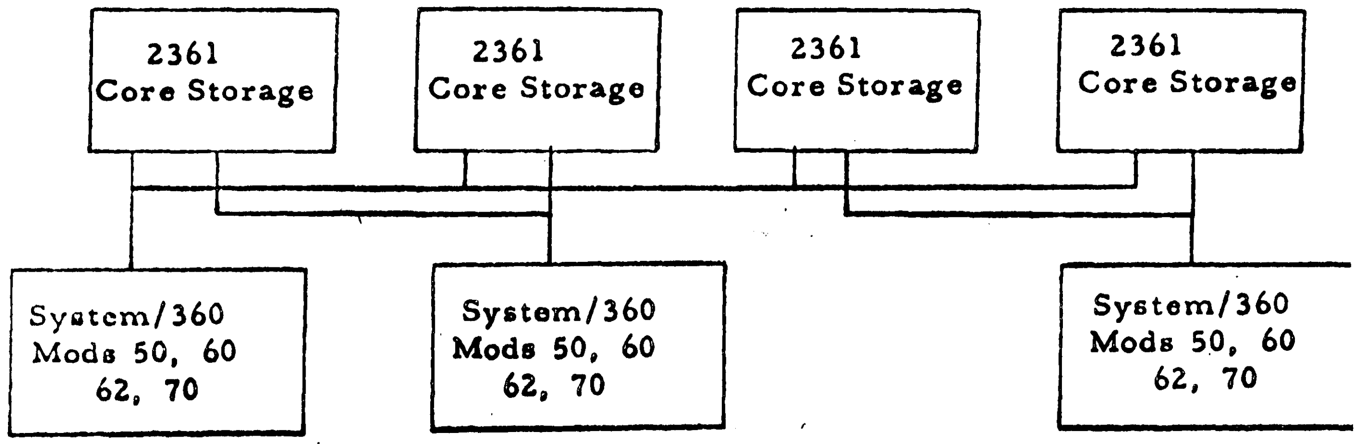

2361 Core Storage
2361 Core Storage
2361 Core Storage
2361 Core Storage
System/360 Mods 50, 60 62, 70
System/360 Mods 50, 60 62, 70
System/360 Mods 50, 60 62, 70

Figure 7-11. Shared 2361 Core Storage

7.2.4 Existing System Usage of Multi-computers

7.2.4.1 Navy Tactical Data System (NTDS)

The Navy Tactical Data System is a multi-computer combat direction system operating aboard ships, in real-time. It is used to process, correlate, and evaluate tactical data. Inputs to the system come from sensors and from other systems. Outputs are provided to other systems both internally on the ships and external to them.

The NTDS system uses a standard computer, the AN/USQ-20 (UNIVAC), and achieves needed capacity by the use of multiple units of this standard item. At least two unit computers are used in each installation to satisfy requirements for high reliability and twenty-four hours' operation. Changes in the system to satisfy additional or modified requirements are met by changing the programming and by the addition of more unit computers.

A set of operating modes was designed to provide ability to handle the most important functions in all but the most extreme conditions of damage or equipment malfunction.

A typical configuration for the NTDS system would consist of three unit computers, three data links (high-speed, teletype, and surface-to-air), a video processor for radar data, three display units, a keyset general, a magnetic tape unit, a symbol generating unit for the displays, and a system monitoring console.

All tasks for the system are classified into three categories: service, tracking, and user. Service tasks are those common to more than one function, and service tasks associated with inputs are usually the highest priority tasks. Tracking tasks have next highest priority. Service tasks associated with outputs and user tasks are lowest priority. An exception is that subprograms connected with inter-computer data transfer, which is classified as a service task, are actually contained in each computer.

In the NTDS, techniques have been developed which allow computer changeover to take place without serious interruption to system operations. Unless no system recovery is involved, computer changeover can be accomplished in a period of a few seconds to several minutes. When computer changeover is required for system recovery the problem becomes involved, especially with regard to preserving data when a computer has malfunctioned. However, critical information can still be obtained from magnetic tape,and other data must be reacquired by starting the computing processes again or by getting the data from other ships.

7.2.4.2 Navy OPCON Center

The OPCON Center configuration of multi-computers uses three CDC 1604's and one CDC 160A which act as a controlling unit. Developed by BuShips, this large-scale multi-computer system is for a classified command/control application. The basic configuration is represented schematically in Figure 7-12.

The system is akin to the Ramo-Wooldridge system, having a crossbar type switch as a central element. 160A and 1604 computers serve as the controlling units. The system differs, however, from the Ramo-Wooldridge approach in a number of important ways. First, the 160A computer - a smaller computer - is designated the controlling computer for the system. Another important difference is in the hardware embodiment of the switch. Rather than a solid state high-speed switch, this switch uses a dry reed type switching element. Presumably, this switching time, instead of being on the order of microseconds, requires milliseconds.

Another important difference between the BuShips systems and the Ramo-Wooldridge system is that there is no assignment table or symbolic file internal storage with the CDC switch. The equipment provides the basic switching function under program control and it facilitates an alert and interrupt system as well. This switch is of a size 16 x 64.

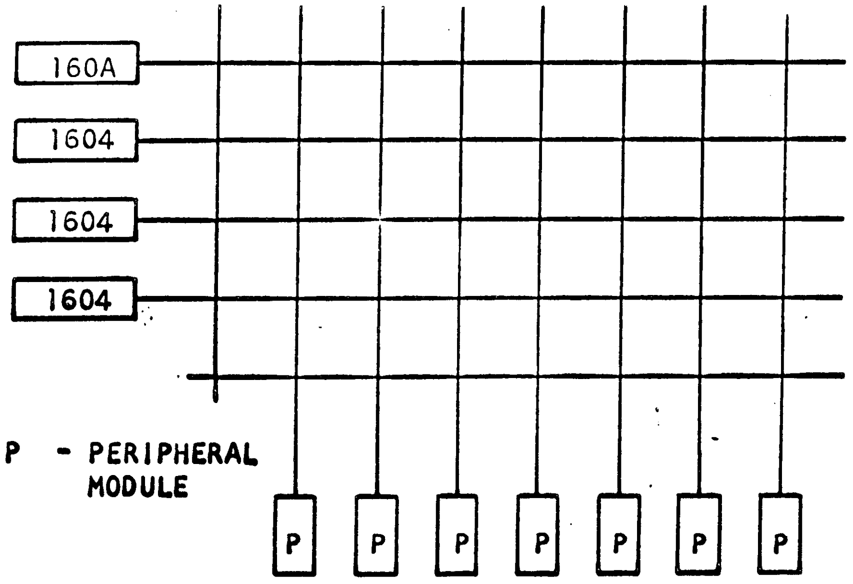


Figure 7-12. Navy OPCON Center Configuration

7.2.4.3 Burroughs D 825 Applications

This system is now being applied to a number of command and control problems. These include air defense and communications processing. The prototype system has been in use at the U.S. Naval Research Laboratory for two years. Two other systems are also being used by the Government. Burroughs is under contract to develop an Automatic Message Processing System for the Signal Corps which is of the store-and-forward type. It will use the D 825 including data demand and data buffer modules and a data demand exchange. Thirty-four systems are installed or in production for the SAGE back-up intercepter control network (BUIC).

7.2.4.4 Real Time Data Handling System - Pacific Missile Range

The real time data handling system of the Pacific Missile Range, RTDHS, provides data handling support in four areas:[7]

1) Ballistic impact prediction for range safety.
2) Flight testing involving air breathing, air-to-air and ground-to-air missiles.
3) Ballistic or perturbed (controlled) impact predictions to aid recovery.
4) Orbital vehicle control to alter free body path as for injection into orbit, rendezvous of orbiting vehicles, and control of re-entry.

The system is heavily communication oriented. The computers communicate with each other and with range instrumentation. Computers used are the CP642B/USQ-20 computers. These computers are placed at various instrumentation sites and at central operations areas.

The basic concept of the RTDHS includes two types of sites; primary and peripheral. Peripheral sites use the RTDH-PC computer. Primary sites have the Q-20B computer. Peripheral sites receive local radar and human control inputs, and provide local radar control, data recording, and data monitoring facilities. Tracking data is sent from the peripheral sites to the primary sites. In addition to the inputs and outputs to the peripheral sites, the primary sites handle range safety and automatic aircraft vectoring operations. Control data and acquisition data are sent from primary sites to peripheral sites.

The system is, therefore, a real-time system consisting of multi-computer installations which communicate through high speed digital data links via wire or microwave. Local automatic control is provided by the various computers, and overall system control is provided by the primary site computers.

The computers are tied together with a large number of special purpose devices for receiving data from local radars, transmission of control information, receiving and transmitting communications data, and digital transmission to display devices. In addition to the special purpose equipment, normal peripherals such as magnetic tapes, printers, and control consoles are provided.

This system illustrates several interesting features. The primary site computers act in a multicomputer environment and at the same time provide limited control for the remotely located peripheral computers. At the peripheral sites, the peripheral computers are capable of almost autonomous control but may also feed into the primary site, primarily in the capacity of data gathering. One of the peripheral sites is also a multicomputer system.

7.2.4.5 NASA Ground Operational Support System (GOSS)

This system includes all world-wide tracking, telemetry, and communication networks. All information gathered is sent to the IMCC (Integrated Mission Control Center) at Houston, Texas.

The IMCC is the focal point for control of Apollo/Gemini missions. Flight controllers integrate all flight operations and provide all necessary support to the mission. The major functions of the IMCC are to analyze all critical flight performance data from the vehicle or the tracking network, display all pertinent information to operations personnel, generate commands and communicate with the vehicle, display pertinent GOSS status and performance information, control the GOSS, recovery-related units, and alternate recovery sites, and communicate with other appropriate agencies.

The computing complex consists of a duplexed UNIVAC-490/IBM 7094 multicomputer system. There are four 7094's used interchangeably for system test, standby, simulated mission, and program test. During system test, one 7090 is designated as the primary computer, and a second one is in standby status. Two 490's act largely in the capacity of input/output processors and message switching.

7.2.4.6 Satellite Test Center

The Satellite Test Center (STC) presents an interesting example of several multi-computer usages.[8] The function of this system is to provide control of satellite vehicles with multiple satellites in orbit at the same time.

Each tracking station has at least two, and, typically, four CDC 160-A computers. A division of the computing task between two 160-A's is according to function. One 160-A receives, and formats input data, the other handles communications data, antenna pointing data, command data, etc. With four computers, two vehicles may be supported.

The Satellite Test Center receives data from the tracking station by telephone lines which lead to a crossbar switch. This switch allows the input data to go to any one of sixteen 160-A computers. These computers act primarily in the capacity of buffers (and are in fact, called Bird Buffers).

Another crossbar switch is used for connections to sixteen printer displays and four CDC-1604's. Both crossbar switches are operated under control of a scheduling computer (160-A).

The arrangement of computers (See Figure 7-13) is flexible for communication and segregates computer tasks according to function. The primary problem of control in this system is one of scheduling. It was found that the controlling 160-A was only marginally adequate for this task.

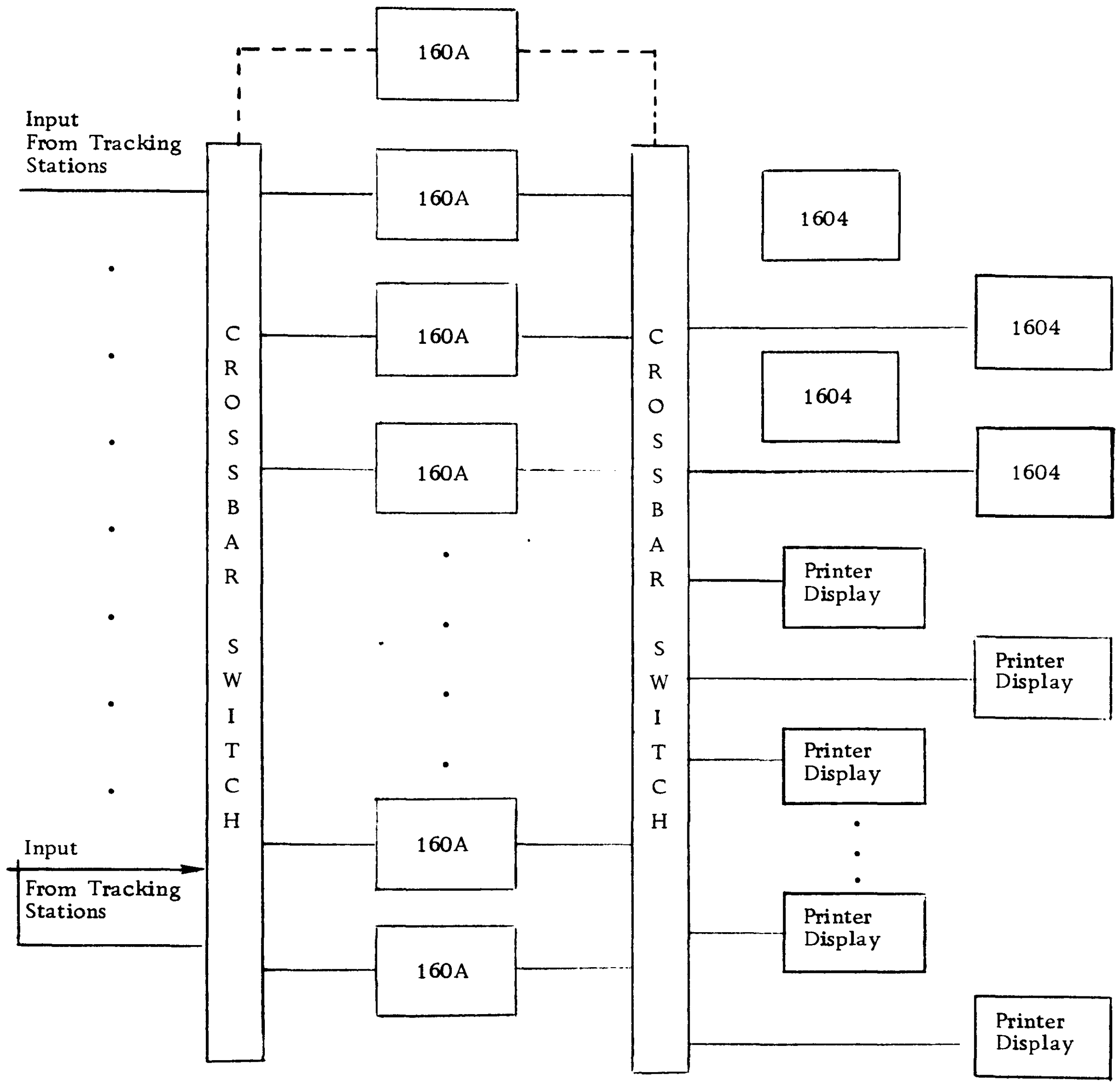


Figure 7-13. Satellite Test Center STC

7.2.5 Hardware Requirements

Certain hardware requirements are essential to achieving desirable system characteristics in a multi-computer complex. Several of these are listed and described briefly to follow:

1) Priority Interrupt Logic - It must be possible for processors to interrupt each other and to be interrupted by other system devices. The logic design may call for a wired-in priority scheme ; it may be entirely a function of software logic; or it may be a combination of the two.

2) Expansibility - The ability to add modules, either in real or extended time, permits the system to access or release equipment (or to grow with requirements which may change with time). This capability exercised in systems real-time permits functioning as a single large system or a number of smaller ones.

3) Switching and Communication - There must be a switching scheme, controllable from the master processors to permit reconfiguration of the system from the hardware and inter-connection standpoint. More than one processor should be able to control, access, and execute the master control program. Extensive ability for data communication must be built into the system with provision for alternate paths.

4) Memory Protection - Provision must be made to prevent memory addressing errors. This is particularly the case when more than one processor is accessing the same memory bank. Several methods of memory protection are described by Critchlow. These include the use of limit registers, mask registers, the hardware lockout feature, use of a check routine and use of fixed (read-only memories). The latter two methods are useful only for special cases.

7.2.6 Programming Considerations

7.2.6.1 Cyclic Operation

In real-time operations, there are periodic functions to be performed. Unfortunately, the periodicity of these functions are likely to vary and sometimes the frequency of operations are unpredictable. This may present severe scheduling and load balancing problems in a multi-computer system. One approach to this problem is to select a computation cycle which approximates the frequency of the most significant periodic occurrences. This cycle may be initiated by a computer clock signal or by some regular event such as a regularly expected input request interrupt. The cycle is of particular importance to the executive control program which may use the cycle interval as a framework for scheduling required processing. Typically, certain functions must operate every cycle, some operations occur at multiples of the cycle interval and others operate asynchronously in respect to the cycle.

7.2.6.2 Interrupt Processing

An important trend in real-time systems is the use of interrupt methods for data input. A characteristic of such systems is the requirement that input/output demands be serviced rapidly. In particular, a sensitive and timely response to command/control information input by human or electronic intervention is needed. In some cases, the computers must respond to reconfiguration commands and alter the priority of operations in real-time.

The importance of servicing external devices before possible loss of data suggests that interrupt control assumes a prominent position in processing priority. Viewed in this way, the aggregate of interrupt processors constitutes a high (perhaps preeminent) level of executive control. By means of the interrupt technique, events external to the computer are registered in the computer program,and the computer is able to respond to new situations in a predetermined and appropriate manner.

The interrupt feature consists of the ability to impose a hardware signal into the computing sequence in order to initiate a required action. The interrupt signal will ordinarily cause a branching of control to an interrupt routine. After the interrupt routine is completed, control is usually reverted to the sequence which was suspended.

Typically, the interrupt signals that data is ready for input into computer memory. It may also, however, simply signal an external event. Another use of the interrupt is to signal the end of a previously initiated data transfer. In some cases, this notification of a completed transfer triggers a new transfer. The interrupt is also used to indicate malfunctions (e.g., power failure, parity error, etc.) Interrupt signals should be identifiable, both as to the interrupt source (which device), and as to the reason for the interrupt.

In a multi-computer system, the ability for one processor to interrupt other processors is a desirable, if not necessary, characteristic. One of the most significant tasks involves the updating of common data bases to maintain currency. An important consideration in this connection is the extent to which data input is to be duplicated.

Interrupts will ordinarily cause interrupted routines (that is, the routine that was operating is interrupted). This may result in a re-entrant condition. If the routine is re-entered prior to completion, previously computed interim results (usually in temporary storage) may be destroyed. This is a problem that has been solved in a number of ways, usually by the individual user programmer. However, standard re-entrant procedures should be adopted, and an easily specified method provided, as a function of system support.

When various equipment is competing for attention, problems of priority result. These priorities relate not only to I/O requirements but to the computations underway. In some systems, interrupt signals may be selectively enabled or inhibited under program control. This feature provides flexibility in providing for an operational sequence based on a priority logic which may be preset or dynamically alterable.

7.2.6.3 Multi-Level Control

The priorities of varying processing and interrupt requirements will ordinarily result in a multi-level hierarchy of control. In a fixed inflexible program sequence, tasks may be arranged in a preset order. This characteristic is not typical of a real time cyclic command system in which the operation sequence is likely to vary from cycle to cycle. To facilitate this type of operation, various levels of executive control may be recognized.

In the Real Time Data Handling System (RTDHS) an executive concept is employed which recognizes a division of executive control function into several levels: the executive, the cycle supervisor, and the interrupt processors.

"Multi-level Programming for a Real-Time System" is described in a paper by Shafritz et al, which makes similar distinctions between executive levels.[10] These levels are called the base control level; which controls operations which must be done each cycle; the iterative control level, which controls non-cycle bound tasks of lower priority; and the peripheral device control level.

The implications of multi-level programming on multi-computer systems is of particular significance in systems which share the executive function between processors. In such systems, communication between processors is required when tasks are undertaken and when tasks are completed in order that the overall systems priorities of operations be maintained. For processors employing shared memories the need for such communication is eliminated.

7.2.6.4 Priority Considerations

The dynamic entrance of priority tasks into a system may require that other tasks must be deferred or suspended. A great deal of complex analysis may be required to determine which tasks should be deferred or suspended in order to cause the minimum disruption of established schedules. Saturation of the computing system with job requests also creates a requirement for analysis and judgment in scheduling.

For example, a decision is required as to whether all tasks should be a little bit late or one task should be very late in order that the other tasks can be on time. However, relatively simple rules can be invoked which state how the executive programs should handle the scheduling and equipment allocation tasks, allowing for empirical optimization by the programmer through his priority and task definition, and at the same time provide for future improvements to be made which reflect more mathematical or sophisticated treatments.

There are a number of criteria for determining priority. Input requests assume a high priority because of possible loss of data. High priority is also accorded failure or malfunction condition signals. One priority criterion suggested by Bauer in 1958 is important to note.[11] He stated that on the basis of the user desires, a priority should be assigned to a problem and a rough estimate made of the time required to perform the computation in the problem multiplexed environment. As the time grew nearer for the completion of the problem solution, the computer would upgrade the priority if it appeared unlikely that the schedule could be met on the existing operation. Thus the computer through its supervisory control would automatically monitor the status of current tasks and would dynamically change the priorities as necessary.

Thus priority may be alterable as a function of elapsed time. An example of this need is seen in several cases. For instance, it is possible that a low priority task could be delayed for minutes (indefinitely) because of its low priority status, when to service the task would require only a matter of microseconds.

Ordering the priority of system tasks is normally the function of the executive program. A convenient method used in several systems (NTDS, RTDHS), and implied in others is to consider all programs to be ordered by priority and referenced in a single task list. It is then possible that each computer will contain an executive program which would scan the list sequentially and select the highest task then pending.

The requisites for a workable multi-computer task assignment concept based on a common task list are:

1) Each computer must contain or have access to all programs referenced in this common task list and all required data.

2) The list must be contained in each computer and continuously updated to insure currency.

3) Timely communication to the other computers must follow. when each task is selected in order to avoid duplicate assignment.

The use of this concept in priority control achieves a high degree of flexibility but has certain disadvantages. The duplicated storage of programs and data may be somewhat wasteful; this may be avoided, however, in systems with memory-sharing processors. Several problems must be solved using this method (e.g., precedence and reentrant standardization problems), but these are not necessarily unique to this concept of priority determination. It is noted that back up and reliability are served by this design since, in case of computer or other failure, highest priority tasks are automatically assumed by the remaining processors.

Another method of task assignment according to priority is to segregate tasks among processors in order of priority; i.e., the highest priority tasks to one computer, and lower priority tasks to the other computers. In case of failure, a switchover may be required to assure that highest priority tasks are accomplished.

7.2.6.5 Precedence Considerations

The scheduling problem resolves to the requirement to operate a given set of programs with available processors in such a way as to optimize one or more operational functions, e.g., to minimize execution time or program steps.

A thorough treatment of sequencing and scheduling from the standpoint of precedence relationships is given by Swartz.[12] An automatic sequencing procedure is proposed. A precedence diagram is first prepared from which is derived a precedence matrix. An assignment matrix is then developed which designates the unit in which each task is to be performed. Six generalized rules are derived which obviate the need for testing all permutations.

Sequencing procedures in a multi-computer system are highly dependent upon precedence relationships. The primary precedence relationship derives from the dependence on the completion of one task for the successful operation of another.

A primary advantage of multi-computers is that greater computing capacity can be brought to bear on high priority tasks and response time is thus shortened. However, the problem must be formulated in such a way as to be solvable by parallel computation. The use of precedence diagrams is very helpful in determining the degree of potential parallelism for a given problem.

In a multi-computer system, precedence relationships must be preserved. These relationships are not changed materially by reason of being operated in a multiprocessing system rather than by a single computer. However, the method of assignment of tasks will vary according to the executive concept. One method of assignment involves a fixed ordering of the tasks under control of a master executive program.

If the duration of each task is known in advance, this method results in an efficient computer use. However, if the duration of the tasks may not be accurately anticipated, idle time may result. An alternate method consistent with the use of a common task list as discussed in the previous section allows each processor to select the highest priority task then pending.

7.2.6.6 Multi-Programming

Multi-programming is defined as the time sharing of a processor or of several processors to operate concurrently on several independent program tasks. Multi-processing refers to the application of more than one processor to the solution of one or more program tasks. The two terms are quite distinct and can even be thought of as two extremes. The two concepts are often considered together,[9] and are applied in the same systems to good effect. They are by no means incompatible, and to some extent, multiprogramming supplies a motivation for multi-computing. There are advantages associated with the use of each, and there are reinforcing advantages to the use of a combination of both concepts.

The purpose of multi-programming is to minimize delays caused by processors having to wait for I/O service. In addition to input/output/processing load smoothing, multi-programming may be said to exhibit advantages of greater throughput, reduction of turn around time, improved machine utilization, and less set-up time (as amortized over the greater number of users served).

Several of the purported advantages are somewhat doubtful and in practice, the added complexities may, to a large extent, negate the advantages. There are scheduling, space allocation, and I/O conflict problems to be resolved. Accounting for machine time used poses another difficult problem.

There are a number of features, both programming and hardware, to be considered in connection with multi-programming. Amdahl lists five requirements which must be provided for in hardware and/or in the programming system:

1) Memory protection,
2) Program and data relocatability,
3) Supervisory program for I/O control and interrupt processing,
4) Interrupt System,
5) Symbolic addressing of I/O.[13]

To these could be added a real-time clock interrupt to provide a basis for distribution of computer time among time sharing programs (hardware), and a clean subroutine linkage method for standard use (programming).

A goal of multi-programming systems is to allow each user to consider that his program is operating continuously; and to relieve him of consideration of I/O details. Therefore, contact with I/O is handled by the supervisor program, and user references should be symbolic (as noted above).

Scheduling remains the principal problem for multi-computer systems. One method, used by the Honeywell 800/1800 system is to treat scheduling as a separate pre-process. This simplifies the operations required of the monitor-executive (control program) at execute time.

A division of scheduling and allocation control functions is suggested by Critchlow.[9] He outlines the functions of three programs: the Peripheral Control Program, the Scheduler, and the Memory Allocator. The hierarchy of control for this scheme is shown in Figure 7-14.

The use of PERT techniques for solution of scheduling problems was suggested by Wegner in a discussion at the Symposium on Advanced Computer Organizations at the 1963 IFIPS Munich Conference. "The critical path of the network would indicate the principal bottlenecks limiting the efficiency of the computer, and would suggest components to be replaced or duplicated. Using this approach, a balanced multi-programming computer design could be developed which utilized adequately each group of computer components."

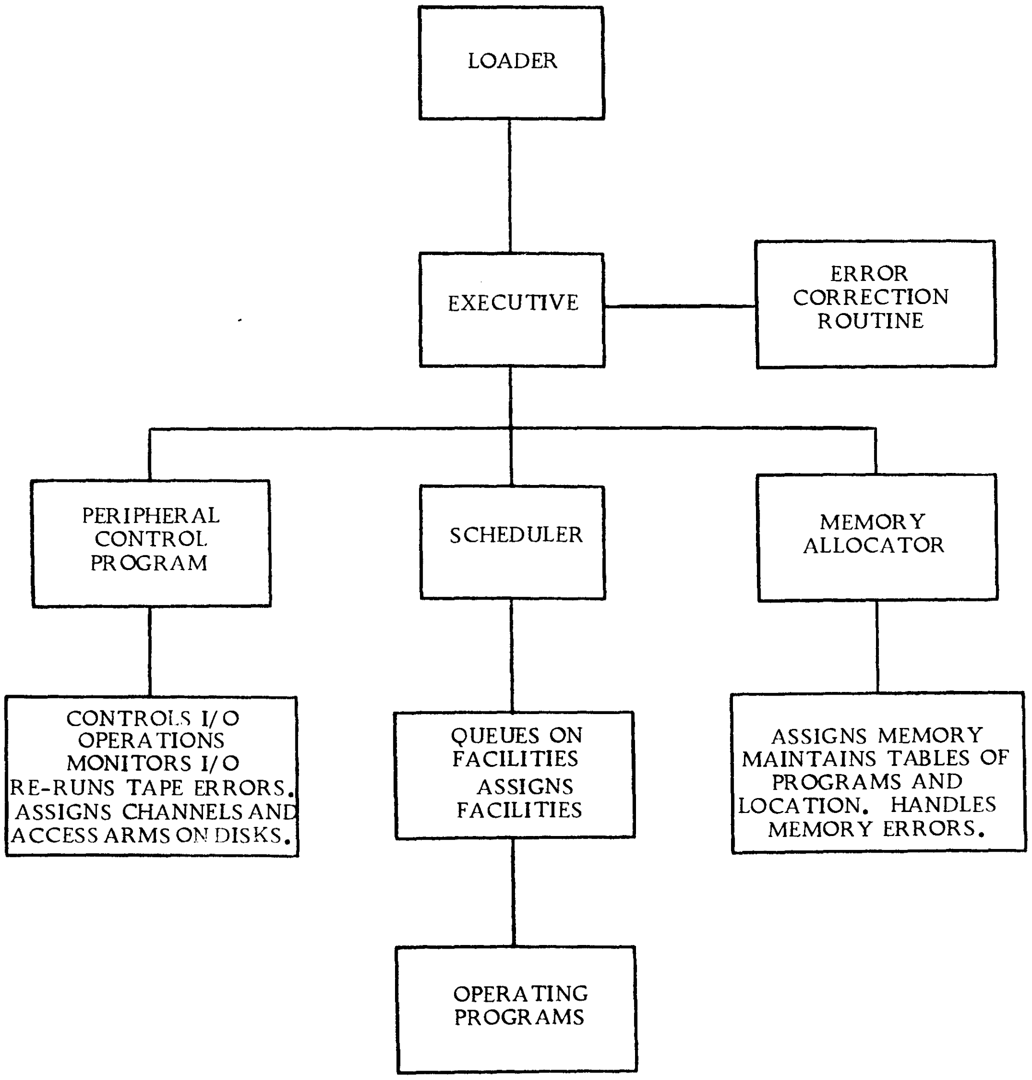

LOADER
EXECUTIVE
ERROR CORRECTION ROUTINE
PERIPHERAL CONTROL PROGRAM
SCHEDULER
MEMORY ALLOCATOR
CONTROLS I/O OPERATIONS MONITORS I/O RE-RUNS TAPE ERRORS. ASSIGNS CHANNELS AND ACCESS ARMS ON DISKS.
QUEUES ON FACILITIES ASSIGNS FACILITIES
ASSIGNS MEMORY MAINTAINS TABLES OF PROGRAMS AND LOCATION. HANDLES MEMORY ERRORS.
OPERATING PROGRAMS

Figure 7-14. Hierarchy of Control for Multiprogramming System.

7.2.6.7 Load Sharing

If the length of computer tasks may be clearly anticipated, the problem of optimum allocation of tasks to processors, and optimum sequencing of tasks within individual processors is greatly simplified. Unfortunately, this is usually not the case. It is not always possible to divide the tasks between processors in such a way as to equalize the load.

Even in cases in which tasks are of fixed and predictable duration, the occurrence of external events, (e.g., input request interrupts, receipt of command information, etc.) may cause computing load imbalance, and possibly enforced idle time. This problem may become particularly serious if the output of one processor is required by another processor for performance of a subsequent task.

In systems for which the processors interrelate to a high degree in the performance of common tasks, intermediate results are often passed between the processors. Frequently, one machine must be idle while waiting for needed data.

One approach to this problem involves periodic monitoring of the program progress in each processor. For this scheme, processors signal each other, reporting status information, and make periodic decisions whether to cause a shift of functions from one computer to another. It is quickly seen that the logic for such decisions is not necessarily trivial and that such a monitoring scheme, if made too complex, becomes so cumbersome as to negate the advantages of load redistribution. In general, the pre-empting of one processor from continuation of its assigned tasks to accommodate higher priority tasks should be minimized.

A preferable treatment of load sharing may be obtained with the use of common function lists as described in a foregoing subsection. With this approach, the processors are able to schedule themselves by selecting tasks on an availability basis. This scheme introduces a self-organizing characteristic to the system, but is subject to precedence limitations as was noted.

"An Analysis of Computing-Load Assignment in a Multi-Processor Computer" is the title to a paper by Aoki, Estrin, and Mandell which presents one of the few analytical treatments of this subject.[14] The problem was analyzed with the use of three 2-computer models. One interesting conclusion derived is that, "assuming two processors with uniformly distributed computation times, it was shown that the ratio of computing time in a one-processor system is a function of the ratio of the standard deviation of the computing time distribution to the expected computing time for the single computer."

7.2.6.8 Programming

The basic limitation in computer technology is sometimes seen as the difficulty in describing problems in an operational language that is both unambiguous and precise. However, multi-computer systems are probably not significantly more difficult to program than other systems. The primary complexities imposed by reason of the multi-computing aspects of the system are those of scheduling and load sharing as discussed in the previous section. To a large extent, executive control for any system deals mainly with considerations which are independent of whether the system is a multi-computer system or not. Executive control in a multi-computer system involves different kinds of complexities (not necessarily more) than for a single unit processing system.

Design of control programs for multi-computers is a pre-eminent consideration, nonetheless, and much attention should be devoted to this problem. The preparation of the executive programs is a one time expense which pays dividends in the system support afforded. The control programs provide the framework within which the operations programs are executed. If properly designed, the system user programmer may be relieved of much of the concern about the details of the system operation. In general, the user programmer should not have to be unduly aware of the multi-computer aspect of the system.

The programmer should be provided with the necessary tools to specify parallelism without the necessity of being concerned with assignment of individual processors to the tasks. This subject is explored in a paper by Conway. He suggests the use of the terms "FORK" and "JOIN" as instructions for specifying parallelism. These are particularly useful commands, since, from a programming standpoint, all parallelism is the result of forks in flowchart paths. Similarly, parallel paths may be rejoined by a JOIN command.

The incorporation of special language features and the addition of new compiler language statements for multi-computer usages will be an important consideration in developing programming facility. The language design should be flexible and should incorporate software modularity; and should be as nearly parametric to the anticipated applications as is possible.

In this connection, the development of a special command/control language is recommended. This language should include special multi-computer macros to treat precedence, scheduling, and task allocation problems. In addition, it should be sufficiently generalized to be to some extent translatable through switch action and should also have display feedback characteristics.

7.2.7 Executive Control

Executive control philosophy has been described in a paper by Pickering et al as "a recognition of the system's tasks in order of system priority; distributing them among the system's computers; and then controlling them in the individual computers of the system by an executive routine within each computer."[15]

There are several kinds of executive control, and within each executive program is usually a hierarchy of control to provide supervisory, scheduling, monitoring, and peripheral control. In addition, as was pointed out earlier, the aggregate of external interrupt signals together with the awaiting interrupt routine logic constitutes another form of executive control which is able to superimpose environmental conditions and occurrences into the system operation.

Several basic concepts of executive control are characteristic in current multi-computer systems. Some of these are:

1) Master-Slave Control - This is typical in systems with centralized control. This has the advantage of flexibility in the assignment to the various subprocessors.

2) Reciprocal Control - This type of control prevails in systems wherein both (or several) computers all have equal control capability, but at any one time only one processor is in charge.

3) Autonomous Control - This type of control may be appropriate for systems which divide tasks largely by function (e.g., one for I/O and one for processing). The executive programs may be relatively independent and interrelate only occasionally.

4) Shared Control - The shared task concept was discussed in previous sections. Here, as in the reciprocal control concept, the processors have equal executive control capability; but no one computer is in absolute control, and the executive concept is one of self-servicing in respect to task selection.

There has been relatively little done in terms of formal or rigorous investigation of program control of multi-computer systems. This subject is a challenging one, deserving much design consideration; preferably early in the development of the total system design.

The determination of executive philosophy is usually derived after other system elements are determined and is, therefore, largely a pragmatic consideration, an after-the-fact recognition of the system structure.

The full and complete optimization of all units of a multi-computer system is a multi-queue and scheduling problem of great proportions. However, much can be implemented by the executive control program to improve the overall efficiency of the system. Although the executive control program cannot yet be expected to optimize the system according to a complex mathematical formula, it can nevertheless accomplish considerable system optimization. It can also provide the very first and important step in the research and development process which must take place to allow full use of multi-computer systems.

7.2.7.1 Burroughs D 825

Since all memory is totally shared between all processors in this system, programs or program segments may be shifted about from processor to processor. Also, since programs need not be associated with any particular processor, it is possible to regard programs as controlling processors instead of processors executing programs. The result of this concept is an executive routine that is executed by each processor when its services are needed for obtaining a new job. This program is called Automatic Operating and Scheduling Program (AOSP).

The AOSP maintains a job file for all programs currently in the system. When a processor seeks a new assignment, it runs the AOSP, and if it is assigned to the program it transfers the program from the main memory to its own memory. If the program is interrupted for any reason, the processor transfers it back to the main memory suitably revised to take account of the work done to that time by the processor. Subsequently, the program is again taken up by another processor or the same processor depending on the priorities and the loadings.

The AOSP may also divide a single program between several processors since program branches may be specified by programmers. In this way, simultaneous parallel solution of the separate segments can be achieved.

The AOSP resides in the main memory of the D 825. The actual memory space required varies considerably with the application because, for any given application, the AOSP is a selection of routines drawn from many options.

The AOSP initiates all programs that are part of the system application. It allocates space for these programs in the system's memory. The AOSP schedules all jobs and determines the equipment configurations to be used. Additionally, it coordinates all input/output operations and obtains buffer storage, updates lists of availability of input/output peripherals, and sequences multiple requests for the same data.

7.2.7.2 UNIVAC NTDS System

The unit computer of the Naval Tactical Data System is the UNIVAC AN/USQ-20 A typical multi-computer installation may use three of these basic units. The executive control philosophy for the NTDS was based on the requirement that in a real-time system the delay in responding to an input is negligible in the time reference of the user equipment or operator. Hence, scheduling of the functions to be performed by the system must ensure that peak loads are smoothed out and all tasks must be performed within their real-time requirements.

The executive control philosophy recognizes systems tasks in order of system priority, distributes these tasks among the computers, and controls the tasks in the individual computers through an executive routine within each computer.

The executive routine consists of an eight instruction main control loop, some interrupt processing, and "flag" setting routines, together with three tables per computer.

An operational program is organized into component subprograms of which there might be ten to twenty-five typically. These subprograms in each computer vary in complexity up to six subroutine levels.

The executive routine calls on one of the subprograms based on the need for execution. When the subprogram has been executed control is returned to the executive.

One of the many interrupts may be activated at any time. Each interrupt triggers a subroutine in the executive thus setting a "flag" to cause a later entry to that subprogram.

The executive control philosophy is complemented through a method of table control. There are three tables, which contain entries for each assigned subprogram, ordered by system priority. The executive flag table contains the time at which the subprogram is to be executed. The executive time table contains the automatic delay before the subprogram can be repeated, and the executive jump table contains the address of the subprogram. In operation, the executive routine sequentially compares the real-time clock of the computer with the entries in the executive flag table. It starts with the task of highest priority. When the clock is greater than or equal to the executive flag table, the corresponding time table entry is added to the current time, the sum is stored in the executive flag table, and control is transferred to the subprogram as specified in the executive jump table. When control is returned to the executive routine, or if no correspondence occurs, the search of the executive flag table is repeated starting again with the task of highest priority.

7.2.7.3 Real Time Data Handling System Primary Site Executive Concept

The RTDHS Primary Site is a two-computer sytem which illustrates the shared executive concept of control. The RTDHS Primary Site executive programs provide the framework within which the operations programs are executed. In general, the control functions divide conveniently between those functions required for system operation and support, and those which control and sequence the operations programs. Imposed upon the operation defined by these two systems of control is a third set of controls, exerted external to the computer by the many interrupt control lines of the system.

Accordingly, the master control design of the primary site computer program incorporates interrelating control programs which are divided functionally into three categories: the executive, the cycle supervisor,and the interrupt processors. The executive program (EXEC) controls the system operation, sequences and schedules system functions, responds to new input,by setting communication items and control information, and controls porgram output. These operations may be performed synchronously in respect to the data cycle. The functions of EXEC are classified into two categories:

1) Class 1 - The higher priority executive functions which must operate every data cycle.
2) Class 2 - Those functions of lower priority which are performed on a time available basis.

The cycle supervisor program (CYSUP) controls the sequence of the operations programs. It accomplishes this by interrogating a common task list, called the common function table, selecting the highest priority task and branching control to the operations program required to perform the task. If the system is operating in a two computer mode, the second computer is notified when the task is undertaken and the program results are transmitted to the other computer when the task is completed. It is noted that the cycle supervisor program is completely dependent upon the common function table; that is, it makes no independent decisions regarding the operations programs sequence other than those explicitly derived from the logic associated with the common function table. Imposed on the system control program (EXEC) and the operations control program or cycle supervisor (CYSUP) are the interrupt processors. By this means, events external to the computer are registered in the computer program in a timely way, and the computer is able to respond to new situations in a predetermined manner.

In addition to the external interrupts which are expected regularly as a function of time, there are many other external interrupt sources from which events external to the computer may be registered in the computer program almost instantaneously. Thus, a sensitive and timely response to command/control information input by human or electronic intervention is possible. The computer can also respond quickly to reconfiguration commands and is able to alter quickly the priority of operations when necessary.

All real-time operations in the system are carried out within the framework of the cycle concept. The cycle is initiated regularly by the input of data (Input Data Request Interrupt). The transfer of control between the several levels of control programs and the operations and utilities programs for three data cycles is illustrated in Figure 7-15. This illustration is for one computer only. The control transfers for the other computer would be similar but they vary in detail except in respect to the common input data request interrupt which initiates the cycles.

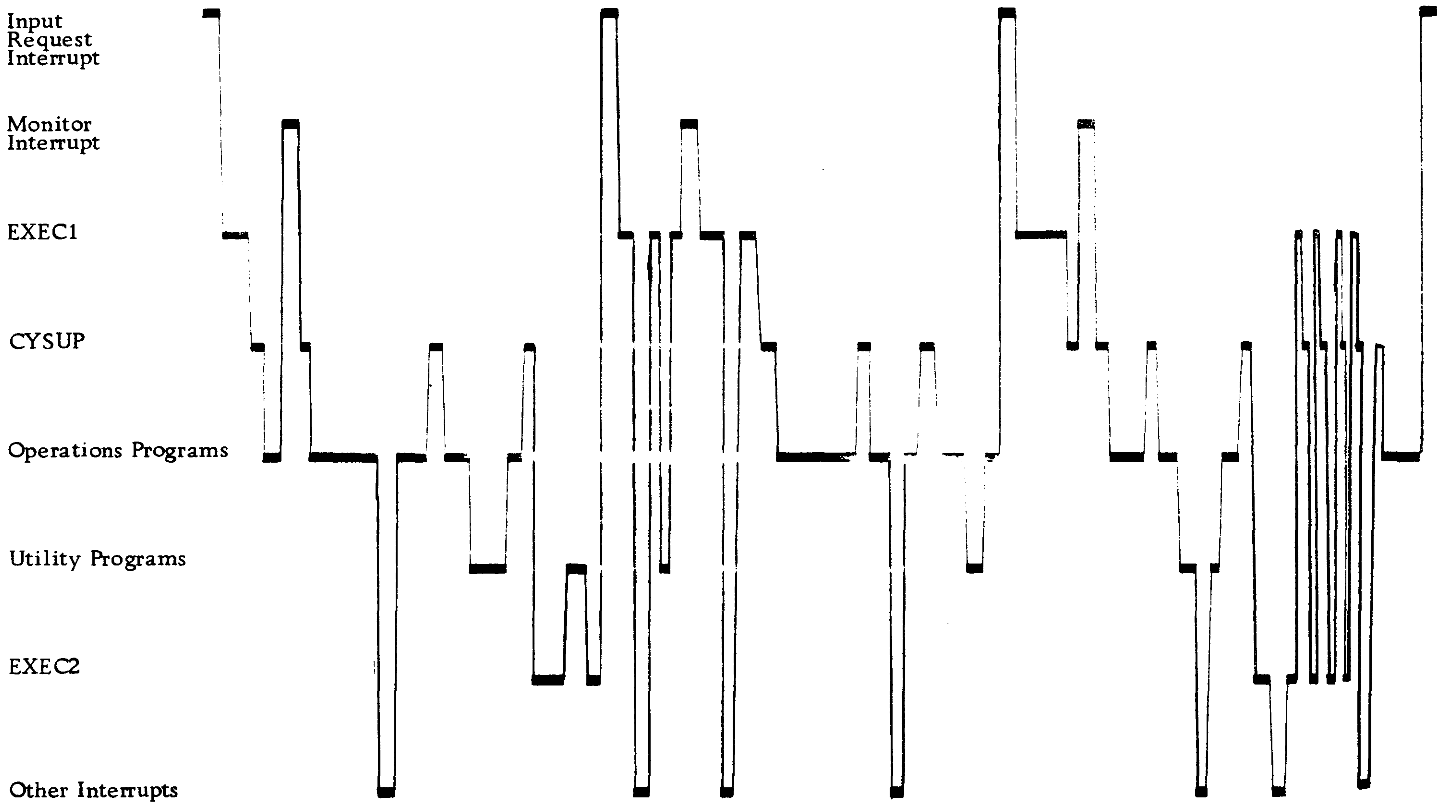


Figure 7-15. Illustration of Data Cycle Control Transfers

7.2.8 Advantages of Multi-processing

The advantages of multi-processing are noted in foregoing sections. A brief summary of these advantages follow. Several potential disadvantages are also listed. In general, they are the contra of the characteristics cited as advantages, i.e., the items enumerated to follow are not all necessarily unmixed blessings.

7.2.8.1 Flexibility

Multi-computer systems are expansible and flexible. Additional modules can be added easily to modular multi-computer systems. Flexibility is gained by using the various larger elements in different ways at different times. Even in a multi-computer system that is not modular, system capability can be expanded by replacing one of the computers with a more capable computer. For this purpose, it is important to provide interfaces among the computers that do not lead to unduly complicated programming for the replacement.

7.2.8.2 Reliability

Multi-computer systems give increased reliability over a system which relies on a single central computer. In the multi-computer system one computer can perform high priority tasks while the other is down for preventive maintenance. Moreover, in a highly decentralized, modular system, a few extra modules are sufficient to give the system the same reliability as that achieved by duplexing a large computer and at a mere fraction of the cost.

7.2.8.3 Equipment Cost

Multi-computer systems can be less expensive than central computer systems. However, the smaller computer usually has a lower capability/cost ratio than the larger. If the cost of equipment were the only consideration the multi-computer system would not show to advantage. But the economics of a system are often improved by a decentralized computer system. The question of equipment cost is a complex one. It requires careful analysis on a case-by-case basis, with due regard to the somewhat intangible cost items, such as tactical compatibility and obsolescence factors. As mentioned earlier, the reliability of the multi-computer complex can be increased at less cost than the duplication of computers in the central computer system.

7.2.8.4 Programming

Programming considerations were discussed in an earlier section. Added complexity required for multi-processor control may be partially offset by several possible advantages.

From a programming point of view, the use of multi-computers allows a more natural division of the problem into levels of control. Interfaces in programming can be developed among the various large programming components, such as: input, output, analysis, information retrieval, and so on. This is of great advantage in large programming jobs, especially where pieces of the programming might be subcontracted. If there is independent programming in independent computers with a simple communication interface among them, programming of the computers can be relatively independent. With executive programs, for example, the separation from the executive control program of the input/output control program greatly simplifies synchronization and memory allocation problems.

7.2.8.5 Summary of Advantages

In summary, a list of possible advantages of the multi-computer system design follows:

1) The equipment complement can be tailored to a specific task by selecting the most useful set of modules for that task.

2) The system can be expanded incrementally as the task changes in nature and in size, thereby keeping an efficient system-to-task match.

3) Parallel processing is achieved, i.e., a number of tasks can be processed at the same time without having to time share equipment.

4) Automatic scheduling and sharing of peripheral units is facilitated.

5) System reliability is increased by module redundancy instead of system redundancy.

6) Module reliability is increased because large capacities are achieved by multiple modules.

7) Maintenance is made simpler since some modules may be used to detect and locate equipment malfunction in the other modules.

8) Continuous real-time system operation is practical since individual modules can be removed for preventive or emergency maintenance without making the overall system inoperative.

9) Tasks that involve widely varying equipment configurations can be handled with high efficiency.

10) A large number of simultaneous requests by men and machines can be rapidly processed by a modular multi-computer system.

11) Improved utilization of equipment may be achieved.

12) Reduction of program checkout time and production run turn-around time is facilitated.

13) Multi-processing and multi-programming facilitate on-line debugging, man-machine interaction, and remote operations and communication.

14) It may be argued that minimization of overall system cost is achieved in terms of cost performance economics.

15) Compatibility with tactical deployment procedures is enhanced.

16) A modular multi-computer system should obsolesce at a slower rate than a single computer system.

7.2.8.6 Disadvantages

Some of the advantages cited above are advanced tentatively since several of these points are open to question and may even belong on the other side of the ledger (e.g., 9), 12), 14)). Others are highly dependent on the successful solution of substantial problems of scheduling and communication (1), 4), 11), 12)).

A few disadvantages are listed below. In general, they are those of program and system complexity, potential conflicts of hardware usage, and coordination overhead.

1) The aggregate of coordination overhead functions may constitute an appreciable proportion of the required computers' activities. Included in such overhead functions are information exchange requirements, extend executive overhead, load balancing, and task monitoring.

2) Conflicts of priority and equipment usage are possible. For example the same device may be needed by more than one processor, requiring that one wait. Memory accessing of the same memory by more than one processor may require queuing procedures.

3) Although improved hardware utilization was cited as an advantage, it should be noted that two computers are usually less than twice as powerful as one. Hardware utilization efficiency results only if, by scheduling, a higher utilization of all system equipment may be maintained.

4) The programming of multi-computers is considerably more complex unless methods can be developed to make the several processors appear to the programmer as one. To obtain this capability may itself be a complex undertaking.

7.2.9 Conclusions

Any future plans for the acquisition of computer systems by the Navy for ACDS should be heavily influenced by multi-computing considerations. The most important aspect of multi-computers is the reliability which can be achieved through module redundancy rather than total system redundancy. Improved performance is obtained by the automatic scheduling and sharing of peripheral equipment among the processing units.

Command and control systems are especially amenable to approaches by multi-computer systems. These real-time systems are large scale, involving many different types of tasks proceeding simultaneously (communications handling, display data handling, internal processing, information retrieval, etc.). The various tasks can be compartmentalized and this compartmentalization can be related to the system components. Also, the changing patterns of the data processing load related to the threat or the military situation is especially suitable for multi-computer approaches.

Computer systems for ACDS should employ multi-computing and multi-programming. In some cases it will make sense to have the computing elements time shared by many processing functions. In other words, if multi-programming and multi-computing are considered as two extremes, future systems should not look like either extreme since there are advantages of both, and there are reinforcing advantages to a combination of both. It can be argued that multi-programming supplies a motivation for multi-computing.

In general, because switching and control techniques are involved with multi-computer systems, the equipment cost is probably somewhat higher. However, these equipment costs are offset by savings in allowing incremental additions of equipment and by the increased service, responsiveness, and reliability of the total system.

Since the switching problems involved with the multi-computer systems for tactical command and control might be quite different than those necessary in commercial applications, studies of the switching problem should be undertaken. These studies should cover the requirements as well as the various possibilities for hardware implementation. For example, simulation studies could be undertaken to better understand the type and frequency of switching within a system since these answers will greatly affect equipment development and equipment manufacturing costs. A specific important question, for example, is whether solid state switching techniques are necessary or whether electromechanical devices such as dry reed switching devices can be employed.

Studies should also be undertaken on the various topological approaches to multi-computer systems such as the number and type of modules required. Similarly, studies should continue with the various approaches to executive control and programming; various executive control programs should be developed and tried.

Multi-computer systems are probably not significantly more difficult to program than other systems. There is, however, a need for highly developed system support in the form of carefully designed executive programs. In general, the user programmer should not have to be unduly aware of the multi-computer aspect of the system and should be able to approach the system as if he were programming a single computer.

Executive systems for achieving program control should be given much consideration in the design and early application of the multi-computer system. The program control system requires as much design consideration as the hardware itself. It is noted that preparation of the executive programs, however, is a one-time expense and pays dividends in the operations programs support afforded.

Programming costs for multi-computer systems are not significantly higher than for single computer system. Executive control for any system deals mainly with items which are independent of whether it is a one-computer system or a distributed multi-computer system. Executive control in the multi-computer system involved different kinds of complexities (not necessarily more) than that control with a single processing unit system. There are many philosophies of executive control through programming. For example, there is the hierarchy approach which was considered in the RW-400 work, as compared with the distributed or shared control in the D-825. In the hierarchy approach to executive control, the control can be vested in one processing unit or can be decentralized to many.

Several general design criteria are advanced for the development of multi-computer systems for ACDS:

1) They should be composed of modular elements capable of communicating freely. The restriction that the processing units, I/O control, and memory units should be housed in the same package should be removed.

2) A major portion of the high-speed memory should be shared by or accessible to the processing elements. The use of common storage eliminates the necessity of large amounts of inter-processor communication.

3) All peripheral equipment and auxiliary memory should be switchable to any one of the main frames or processing units which could conceivably be used to support it or transmit data to it. All displays should be accessible to all computing equipment of data files which could conceivably supply information to them.

4) Arithmetic/control units should be small enough so that two to four units can be used to supply the needed processing. The degree of granularity allows the redundancy necessary for reliability and provides growth in small enough increments to allow cost savings.

5) Different types of main processing units (central processor, communications handling, display data handling, etc.) are acceptable. However, the employment of modules should be consistent with the principle espoused in 4) above.

6) The system should have a limited self-organizing capability such that it can schedule tasks, allocate modules, and reconfigure the system according to simple procedural directions (both programmed and operator controlled).

7) The system should be self-sufficient requiring little operator intervention in the usual situation, yet be instantly responsive to developing situations.

8) The equipment itself must be capable of accepting information from many sources asynchronously and without data loss. This implies an extensive interrupt capability and the establishment of fine gradations of priority.

7.3 ASSOCIATIVE MEMORIES

7.3.1 Introduction

To keep pace with computer problems of ever increasing complexity, the usual solution has been to demand greater speeds of computing components. Further speed advantages may be expected with the application of micro-miniaturization techniques. Devices currently under development have switching speeds measured in fractions of nano-seconds. However, theoretic bounds of maximum speed appear to be implicit in certain physical parameters which at this time seem to be immutable. It also is evident that a number of important problems are still not computable at current or anticipated speeds by conventionally organized computers.

As it becomes more difficult if not impossible to achieve order of magnitude speed advantages, increasing attention is given to the possibilities of utilizing parallel processing techniques to achieve additional computing power. The development of associative memories constitues one approach in this direction.

The use of associative memories in future command systems appears promising. Several areas for which associative memories are candidate are:

Radar data - track data correlation.

Scan-to-scan radar data correlation.

Track data retrieval.

Weapons assignment.

Threat evaluation.

Sorting.

File search for communication and display.

Message format conversion.

ECCM.

Decoy discrimination.

The development of associative memories is very broad and it is difficult and perhaps inaccurate to generalize on all such implementation with a single evaluation. Currently, there is a wide assortment of designs in various stages of development using many different hardware approaches. Therefore, we indicate the kinds of capabilities that are provided in current designs, discuss programming implications and the kinds of applications for which associative memories seem best fitted, and consider potential extensions of the capabilities in the direction of parallel processing. We then place associative memories in the context of concomitant developments, both competing and reinforcing, and finally, discuss their pertinence to the development of future tactical command and control systems.

7.3.2 Implications to Advanced Tactical Data Systems

The programmer tends to regard the advent of associative memories as a blessing since it provides him a new capability. However, since the development has not won widespread acceptance, and saleable commercial versions appear to be some way off, it must be examined closely from the standpoint of capability per unit cost. In particular, the following questions must be considered:

1) Do associative memories provide a capability for a number of applications of interest to future tactical command and control systems?

2) Are other developments (e.g., hardware advances and new software techniques) likely to achieve the same capabilities at less cost? Will applications be found which justify development to the point that commercial versions will be in production and readily available by 1970?

3) Do associative memories appear to be more useful or less useful for tactical command and control problems than for general computing problems as a whole?

4) Do associative memories provide computing power at times of system stress? (i.e., at times when normal system excess capacity is unavailable)?

5) Are there uses to which associative memories may be put which will achieve results unobtainable by other means?

The first questions are typical of any decision involving differential costs. The last question, however, alludes to one of the distinguishing characteristics of military applications and to command and control problems in particular.

These characteristics are those of urgency and immediacy. Military applications sometimes demand capabilities which can be supplied only by utilizing devices on the frontier of technological development. Where matters of national interest and perhaps survival are at stake, absolute performance is somethimes demanded by means which would not seem justified according to usual cost/performance formulae.

In summary, the inclusion of associative memories in a future tactical system must be justified either on the basis of intrinsic economic advantage (more capacity per unit cost) or on the basis of unique performance capabilities in critical situations.

7.3.3 Definitions

The term associative memory (AM) as used in this report encompasses a number of current and postulated developments, several of which are perhaps more descriptively called by other names. Some of these designations are:

1) Content addressable memories.
2) Search memories.
3) Parallel search memories.
4) Recognition memories.
5) Data addressed memories.
6) Tag memories.
7) Catalog memories.

The term content-addressable-memory (CAM) is perhaps the most descriptive term since the capability to address by contents is the basic characteristic common to most proposed designs. The term search memory is indicative of the type of application anticipated and, in conjunction with the word parallel, suggests the basic advantage over conventional computer organization, i.e., parallel logical operation. The associative aspect of these memories derives from the ability to obtain quickly associated information by specifying a set of key information referred to as associative descriptors.

Proliferation of terms is due not only to the usual chronic difficulty of achieving standard terminology, but also to the fact that researchers are anticipating usages from differing viewpoints. Therefore, emphasis is placed on different aspects of the same basic design depending on the applications envisioned.

7.3.3.1 Content Addressable Characteristics

The addressing technique employed for conventional memories requires that the location of the information to be retrieved be known. This is sometimes referred to as coordinate addressing or addressing by location. An alternative addressing technique is the retrieval of stored information on the basis of content. Word cells are accessed by the characteristics of the stored data rather than by the physical location of the cell. The physical location is immaterial in this case. Memories with this capability are said to be content-addressable. The term content-addressable-memory is somewhat broader than the associative memory designation since several other computer memory organizations not generally recognized as associative memories depend on the content-addressing capability.

The development of new addressing techniques is not without precedent. It is sometimes suggested that the development of content addressing is analogous to that of indirect or other recognized addressing techniques (e.g., relative, indexed, implicit, immediate, truncated, etc.) To a similar (or greater) extent, content addressing provides generality to the common task of "operand fetching."

The problems of addressing are taken for granted to such an extent by the programmer that he is perhaps not aware of the somewhat superficial relationship of an address with a quantity of information. Unfortunately, the programmer, particularly, the machine language programmer, must often deal largely with addresses rather than information This preoccupation with location imposes a large housekeeping and data organization requirement which is expensive in terms of machine execution time, and storage and programming effort. Although it is not unreasonable to expect the programmer to keep track of and in some cases plan where information is to be kept, it is interesting to conjecture concerning the advantages to be obtained by a programming technique substantially free from the addressing requirement. This is feasible with several designs of content addressable memories. Information is simply stored in the first available cell and may be retrieved later in one cycle (or two) without knowledge of the physical whereabouts of the information.

7.3.3.2 Parallel Search Characteristics

Some writers prefer the designation "Parallel Search Memory" (PSM) which conveys what is felt to be the basic advantage of the design, i.e., the capability for simultaneous search of multiple registers for stored data identified by its characteristics rather than by its location. This term is descriptive of the type of application for which associative memories seem best fitted, e.g., those tasks for which a conventional computer would perform a sequential search through data cells. The speed advantage derived from parallel logical operation could amount to an order of magnitude performance improvement. The parallel processing aspect of associative memories is the most exciting characteristic. Extensions of this capability include the concept of simultaneous computation in a distributed logic machine.

7.3.3.3 Associative Characteristics

The associative memory designation derives from the facility to obtain quickly associative information. This may be accomplished in several ways. For some designs the physical address is obtained when a successful search is terminated. This address, which is usually a function of the address of the desired information, may then be used to retrieve the associative data. In other designs, only a part of the contents of each cell is used for the purpose of the search, and the remainder of the cell (which may or may not have content-addressable properties) consists of associative information. Alternately, the associative information may be obtained by means of an address contained in the accessed word which, in turn, is used to access (conventionally or not) another area of memory. Thus, it is possible that a word may be compared with several thousand quantities in one access time; when a match with a selected subset occurs, the associative data may, in turn, be used to call other subsets in an associative chain.

Some writers reserve the term associative memory for those designs which are used in conjunction with a conventional memory, assuming that the associative information is stored there, and that the key (or index) to it is obtained by associative memory interrogations.

7.3.3.4 Other Terminology

Other designations which refer to the same basic capabilities are found in the literature on the subject. Examples of these are the recognition memory and the data-addressed memory (DAM). The term tag memory is sometimes used to describe a memory which limits the search capability to certain bits (called the tag field) of the cells to be searched. The catalog memory is used in the same connection to refer to a memory in which the tag field encompasses the entire memory cell.

Finally, it should be noted that the term associative memory has only recently come to refer to a particular hardware design. Formerly, the term was used to refer to the use of conventional memory for purposes of associative logical processes. This usage of the term persists and is of continuing validity.

7.3.4 Associative Memory Capabilities

Associative memories currently under consideration present a broad spectrum of capability -- from the simplest search based on the criterion of equality, to searches using selected bit patterns (masks) and based on any of a dozen search criteria, and extending to proposals which include computation on matched items. The feature, in common with most designs, is the capacity to interrogate simultaneously a region of the associative memory based on comparison with an external key. The characteristic property is economical search speed. In the more advanced designs, individual memory cells are capable of elementary logical decisions in relating the stored data to the search criteria. By a modest extension of the logical capability, parallel operation of simple conventional commands is obtained.

7.3.4.1 Kinds of Capabilities

Associative memory functions can be classified as: (1) search functions, (2) read function, and (3) write functions. The range of each type is quite broad. Accordingly, the combinative possibilities for the design of an associative memory are extremely varied. Thus, many tradeoffs exist which may be manipulated for optimization. An important outgrowth of this fact will be the eventual ability to specify a capability which is very closely oriented to specific application needs.

Search functions vary according to the types and combinations of search criteria and the various ways of specifying the search field to be interrogated. A typical search function will involve the specification of a number of search descriptors (e.g., search mode indicators, search criteria, masks, comparands, etc.).

The reading functions relate to the type of reaction desired as a result of the search. For example, data contained in the addressed cell (as in conventional location addressing) and/or the physical location could be desired as an output to the search. Multiple match resolution may be a readout feature. Other read function by-products may include a tally of the number of matches or simply an indication of whether one match or no match was found.

Write functions may include conventional coordinate addressing, but may also allow other write modes. One method of writing would load the first empty (zero) location with the word to be entered. This feature has a data packing effect and tends to preserve contiguous blocks in the associative memory. The ability to write selective bits (or fields) in a particular word is usually provided. This is achieved by forming the logical product of the operand and a mask register. A parallel write operation is provided in some designs. This allows the entire memory, or a selected subset of memory, to be changed in parallel.

7.3.4.2 Search Criteria

Search criteria are presented simultaneously to all or to selected subsets of the associative memory. Criteria other than identity with the externally presented key are possible. For example, the associative memory may be designed to search on the basis of: greater than, less than, between limits, minimum, maximum, next higher, next lower, etc. Mixed modes may also be specified involving combinations of independent searches performed in sequence. This is also referred to as a successive search criterion. In addition, a simultaneous search criterion may be provided which accomplishes a search on multiple criteria in one search cycle. Composite searches in which different search criteria are used for each of several fields in the search word are also proposed in some designs. The "m of n" logical relationship is also possible; thus, a successful search is defined as one which satisfies m out of n criteria (e.g., any three out of four specified characteristics).

The satisfaction of the specified search criterion is said to result in a "match." A search may result in a match, no match, or in some cases multiple matches. Some designs include the option of specifying that the search be terminated after the first match or after n matches occur, where n is a preset threshold parameter. A more usual method is to provide a multi-response resolver which provides unique to non-unique processing to allow a serial treatment of matched items. Typically, the matched cells are identified, i.e., tagged or marked, in order that the contents may subsequently be retrieved. Typically, one or more tag bits are provided with each cell for this purpose.

A select pattern or mask is sometimes used in conjunction with the match word to specify that only certain bits of each cell be examined. This permits the use of a completely variable field structure to fit any format requirement and allows any or all bits to participate in the search. Memory partitioning may also be provided. This refers to the ability to search on segments of memory selectively. This capability is especially powerful if the capability to reconfigure search memory segmentation is also provided.

7.3.4.3 Associative Descriptors

As indicated in the foregoing discussion, input parameters to the search process are made up of several elements. The input arguments, when considered separately from the data, are sometimes called the associative descriptors. A list of these parameters in the approximate order of logical complexity is given in Table 7-1.

As parallel operations become more intricate, the required (distributed) logic is extended. The additional capabilities are, in some cases, prohibitively expensive. This factor motivates simplicity of design and, where possible, the use of programming to extend the logical properties of the hardware. For example, it may be found more economical to provide the sequential retrieval of multiple matches rather than the more complex capability to distinguish a unique match from a non-unique match and an automatic tally of the number of matches.

In this connection, a renewal of motivation for the use of micro-programming may be justified. Many of the more complex logic (i.e., instructions) may be derived by programming. In general, the parallel execution of commands of modest complexity will depend on the effective combination of logical subcommands of an even lower order.

Table 7-1

Associative Descriptors

Input Parameter	Explanation
Comparand	The quantity to be compared with search memory entries.
Secondary comparand	For between limits searches, the upper and lower limit is specified.
Search criteria	This parameter specifies the search criteria upon which the search is based (e.g., greater than).
Mask	This parameter indicates the subset of bits of each memory cell to be used for comparison with the comparand.
Threshold limit	It may be desired that the first match or that n matches result in termination of the search. The threshold parameter specifies this limit.
Operator	In more complex versions, parallel logical operation upon a subset of the associative memory is specified. The operator indicates the logical operations (e.g., inclusive or exclusive or, delete, insert, etc.). Arithmetic operations are also feasible but require a higher order of circuit complexity.
Search mode	This parameter may specify the type of answer desired by the search mode, e.g., retrieval of matched entry, address of matched entry, operation upon the matched entry, number of matches, etc.
Memory area delimiter	This parameter limits the area of search within a specific region of the associative memory.

7.3.4.4 Organization of Associative Memories

Several methods may be used to obtain the associative information after a search interrogation. In some designs, only part of the cell is used for the search (is content addressable) and the remainder of the cell constitutes the associative information. This arrangement is shown in Figure 7-16 (A). Another method of deriving the associative memory is by using the physical address of the matched cell to indicate the location of the associative information in conventional memory (Figure 7-16 (B). These two arrangement are logically identical if the somewhat arbitrary division of boundaries of word length are ignored and it is assumed that both sets of associative data reside in conventional memory. A third possibility is that the address of the associative information is contained in the accessed word (Figure 7-16 (C).

Although an ordering of addresses is implied in the diagrams, any number of methods may be used to transform the locations in Figure 7-16 (B). Similarly, A, B, C, and D as shown in Figure 7-16 are simply address tags indicating locations in conventional memory and need not appear in any particular order.

7.3.5 Programming Considerations

New developments in computer hardware invariably suggest changes in programming techniques and software methods. For associative memory programming, the conventional instruction repertoire must be modified and extended in order to provide those functions required to manipulate the associative memory search descriptors.

7.3.5.1 Associative Memory Instructions

A number of command sets have been postulated - some of them highly structured.[16,17,18,19,20] Some designs assume that the associative memory is used in conjunction with a conventional memory which contains the instruction strings (programs). Others assume that the associative memory contains the instructions. In this case there are programming considerations regarding differentiating instructions from data. Contents addressing may be used for this purpose and for controlling the program sequence (i.e., the next instruction to be operated is determined by a search of indicator bits.) Instruction decoding done within the associative structure often uses a stored logic design.

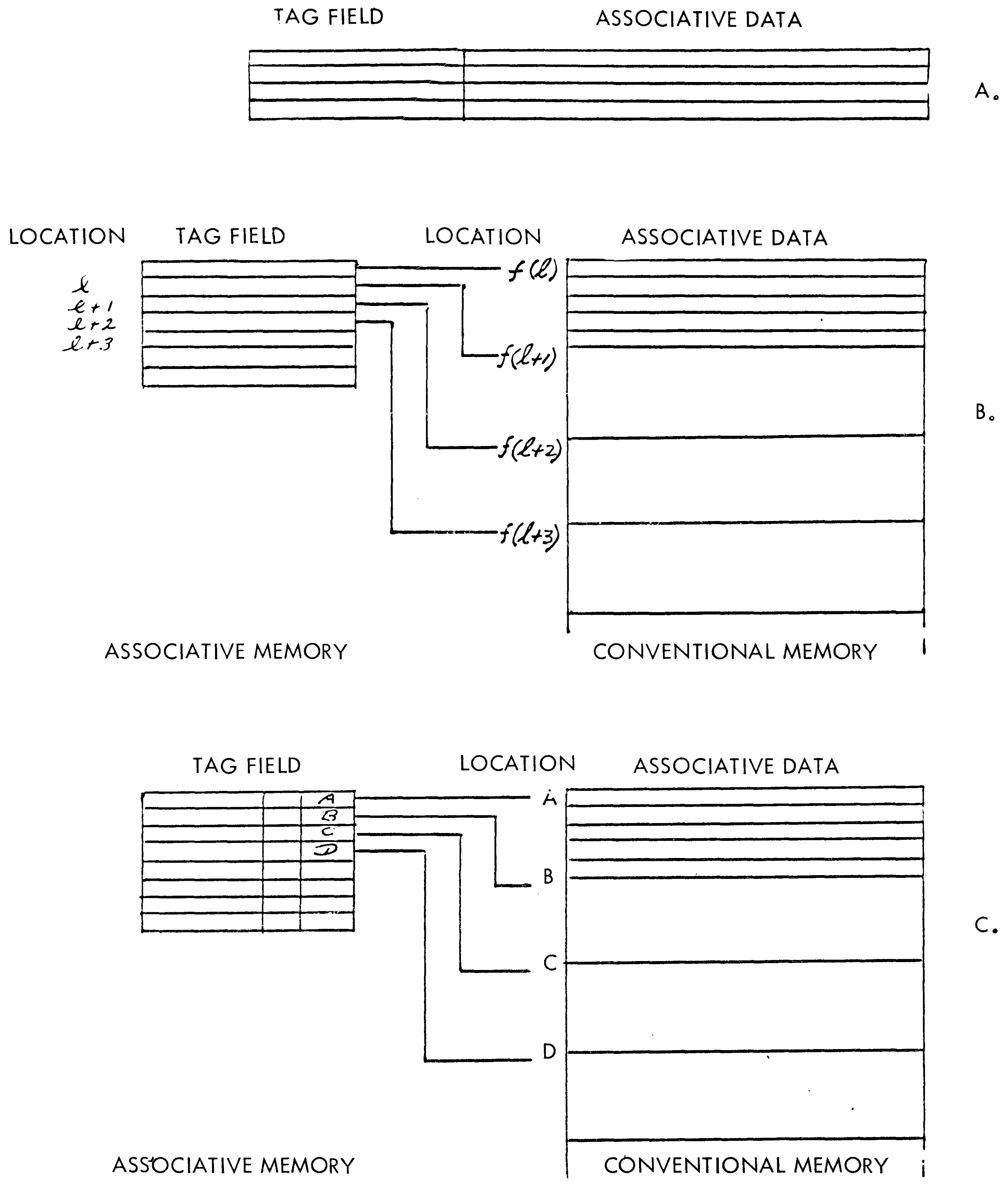


Figure 7-16. Associative Memory Organization

The basic commands used for programming associative memories will tend to be elemental for some time to minimize the considerable cost in providing distributed logic capabilities. It is important to consider the differential cost of hardware vs. software implementation (i.e., cost vs. processing time) for each new feature which is contemplated as a potentially useful addition to associative memory instruction repertoires.

Instructions used for associative memories can be classified according to whether they call for the performance of central logical operation (as with a conventional command) or for the exercise of the logical circuitry; associated with each cell executed over either the entire contents or a selected subset of the associative memory. These categories might be termed central logic and distributed logic, respectively. The latter term is more widely applied to computer organizations. Examples of the first type are: the loading of central control registers (e.g., data register, mask register, comparand register, etc.) preparatory to operation upon previously retrieved values. The latter type are search commands which are selectively operated as specified by the setting of search designators. These include the ability to store data in cells by parallel write (store) commands. It is also possible to mechanize one or more simple operators at each cell, providing a parallel computation capability.

A paper by Estrin and Fuller describes algorithms for arithemetic and logical commands which are executable in parallel for all words in memory or for some input set determined by a previous search.[19] The commands are classified as basic memory commands, basic logical commands and compound commands. An important feature of this instruction set is a replacement addressing mode which allows writing into many word cells simultaneously.

Several methods of writing into an associative memory are possible. If the intention of the write operation is to store a quantity in the first available cell, the write command is preceeded by a search for a zero word. Alternately, a conventional (location addressing) mode may be used if the location is known, as may be the case when one value is to be replaced with another. Another technique for storage of data involves the setting of location tag variables in a field of the associative memory. In effect, this method provides the individual cells with names; unique names, if desired.

7.3.5.2 Programming Ease

It is of particular concern for implementation of military applications that programming methods be tractable in terms of reasonable programmer effort. It is likely that devices requiring complex and intricate programming methods will be adjudged as unattractive selections for advanced tactical data systems unless adequate interpretive or compiler tools are available to make the programming task manageable.

At this time, the implications of associative memories on this question are not very clear. Obvious usages to obtain efficient retrieval do not appear difficult to apply. The programming for a search process is simply to specify the various search descriptors and retrieve the matched items. However, if the more complex parallel computing capabilities are assumed, questions of usage arise for which answers are not obvious. A question such as "How do we utilize the capability to do 1000 multiply operations at the same time?" resolves into the more general question of how problems may be formulated to be solved in parallel. In short, the ability to program parallel processing routines will prove to be much more difficult than that of programming a search. For the most part, these instructions will consist of loading the search memory control registers, specifying the search mode, executing the search command, and retrieving the matching quantities. For example, the instruction sequence might be:

1) Load Test Word (Comparand)
2) Load Mask Word
3) Specify Search Mode
4) Execute Search (tag matched cells)
5) Retrieve tagged quantity(s)

On a higher level of code, a macro-calling-sequence might specify all the foregoing functions. For example,

LOAD A, M

could be used to retrieve the (first) cell which contains A in the field(s) specified by the Mask M. Similarly,

STO X

could call for the storage of X in the first available cell and could represent the following constituent commands:

Load Search Word (with zeros)

Execute Search (Tag first zero cell found)

Retrieve address of Tagged Cell

Write X into Tagged Cell

Macros of the simple sort indicated above would help simplify the programming task and would not pose an unusual task to existing assembly and compiler programs.

7.3.5.3 System Uses

The natures of the order codes differ widely, largely as a function of the varying design approaches to associative memories. The basic differences stem from whether the associative memory is visualized as a peripheral element for use with a conventional computer, as special circuitry in a system organization composed of several (possibly diverse) memory components, or as the central memory element in what might be called an associative computer.

7.3.5.4 Associative Computer

The use of an associative memory as the main memory in a computer was anticipated by several writers.[16,21,22] Davies describes a design in which an extension of the principles inherent in the associative memory point to the possibility of complex simultaneous computations.[16] He provides a command list composed of logically simple subcommands and demonstrates the feasibility of using microprogramming techniques to extend the repertoire. This is illustrated by a multiply routine code built from the simpler commands in the command list. By extension, general computation can be performed simultaneously throughout the memory upon selected data (i.e., on data satisfying selected search criteria which are then tagged and thereby made candidate for further computation). Davies concludes that the organization of such a computer is particularly suited to the type of computation in which the same program must be performed on a large number of data sets. This is characteristic of many data processing problems and of a number of scientific calculations as well. In fact, it raises the question of how often problems originally formulated in some other way can be recast into this form.

The last comment is pertinent not only to this particular design but to the subject of associative memories in general and to other novel developments in the direction of paralleled processing designs.

Rosin has described the "Organization of an Associative Cryogenic Computer" wherein instructions must be in sequence but not necessarily contiguous and thus data words may be intermixed.[22] He utilizes control bits for interrupt control to identify instructions, specify the addressing mode,and for sequencing control.

The instruction word format consists of twenty-five control bits, a 48-bit tag field and a 64-bit data field. An interesting feature of the design is that the contents-addressing feature provides the instruction sequencing function. This is accomplished by a masked search (based on a comparison with two control bits) for the next instruction to be operated.

7.3.5.5 Associative Memory Used as a Subsystem

The associative memory is more often suggested as a device to be used in conjunction with a conventional computer organization. As such, it consitutes a special purpose element in a system providing an augmentation of computing power in those areas in which associative memories show promise.

Such subsystems have been proposed by several writers and implied by others. The associative memory is also suggested as an appropriate component in the "variable structure" of the fixed-plus-variable computer proposed by Estrin.[23]

Another novel usage is the Associative Logic Parallel System (ALPS) described in a paper by Seeber and Lindquist.[24] This system features parallel associative memory modules used in conjunction with a set of computer modules. It uses the associative memory to store instructions as well as data. The program structure is in the form of a binary tree. Associative logic is used to control the parallel computers.

A "Search Memory Subsystem for a General Purpose Computer" is described by Kaplan.[25] The unit operates similarly to a conventional arithmetic unit, receiving data directly from main memory and placing the results of the search in an accumulator. This design contains a search memory of 4096 36-bit words and 4096 48-bit words of data memory. A number of working registers are provided: a search register containing the comparand, a mask register, a mode of search register, and a parameter register. The parameter register provides for variable length field designation. The first bit of each field is indicated by setting a "1" in the first bit of the field of the parameter register. All other bits are set to "0."

7.3.6 Applications

The applicability of associative memories to future tactical data systems depends largely on whether significant portions of the processing problems are of a type for which the content-addressing capability is demonstrated to possess decided advantages over conventional (location) addressing. It is, therefore, important to determine those problem characteristics which tend to guarantee that problem solution time will be decreased (or effective work increased) with the use of this capability; and to define classes of data processing operations which make effective use of the inherent parallelism afforded by associative memories.

Content-addressing is efficient relative to location addressing for those applications in which:

1) Files of data are manipulated and the search for individual items in the data store is a significant part of the data processing task.
2) Fast or immediate access to data is needed in order to avoid table search functions.
3) Data is retrieved on the basis of a multiplicity of reference properties and cross referencing between files is a frequent program activity.
4) List organized data storage is required.
5) Sorting is performed.
6) Encyclopedic data is stored from which retrieval of individual items is called for.
7) Cataloging and cross indexing is required.
8) Comparisons with data arguments are needed quickly and test answers concerning item characteristics required.
9) It is necessary to order data quickly for output or in the case where data tends to become disordered dynamically during processing, or must be reordered according to varying criteria.
10) Problem solution is obtained by solving large sparce matrices.

These characteristics are to a large extent concentrated on data-oriented problems. In particular, data with the following attributes:

1) Data with multiple arguments.

2) Data for which unpredictable growth of tables , arrays, or other structures appears likely.

3) Data with a large proportion of void or zero elements.

Retrieval execution time is largely independent of table length. The advantage of an associative memory is, therefore, more pronounced as file size increases.

Several applications which are anticipated for future tactical data systems are of the type indicated in the foregoing list of characteristics. These are discussed briefly in subsections to follow. Beyond the special applications for which advantages are easily seen lie new usages derived from new software techniques. These further advantages will accrue from the reformulation of problems from the standpoint of explicitly exploiting inherent parallelism, and from the unknown but predictable advances to be derived from experimentation in these techniques when associative memories are a practical hardware reality.

7.3.6.1 Application Studies

Relatively little has been done in the area of detailed associative memory application studies. Instead, the emphasis of effort has been on the engineering aspects of the development. The current challenge appears to be to develop a workable version with sufficiently strong logical properties at a cost which is likely to result in (commercial) acceptance of the design. Thus, significant work on the applications of associative memories is much harder to find in the literature than descriptions of the development of (yet another) "economical associative cell," "logical design for associative memory" or "organization of an associative computer." Another factor, however, is that many striking advantages appear even from a superficial investigation, so that further motivation is not needed to spur designers to continued efforts. The utility is recognized; an economic mechanization is the next step.

There are several reports, however, of substantial scope which give quantitative evaluations of associative memory performance contrasted with standard computing methods.

A paper by Gall contrasts the performance of a sample tracking problem using the USQ-20 Computer with the performance of a search memory used in two alternate configurations of the USQ-20 on the same problem.[26] This application shows the advantages of an associative memory to an impressive degree. The results of this study are discussed briefly under breakdown 7.3.6.3 of this section.

A comprehensive study of content-addressable memory systems is contained in a report prepared by R. H. Fuller at the University of California under the sponsorship of the Office of Naval Research.[20] The problem areas investigated are function optimization, visual pattern recognition, solution of elliptic difference equations, real-time vehicle surveillance and determination of a critical path through a graph. Performance figures are obtained for content-addressable memories of several types. Characteristics based on current implementation design efforts were assumed. The implementation media considered were: ferrite core, cryogenic tunnel diode, and permalloy film. Solution times are compared with performance using the IBM 7090, which is held to be representative of current large-scale, general-purpose computers. The 7090 is used to compare test problems for which software techniques are highly developed and associative memories are also considered candidate. A 7090-CAM configuration was also studied.

Of particular interest are yet other solution time comparisons with highly-parallel computer organizations: the Spatial Computer (SPAC), the Solomon Computer developed at Westinghouse, and a computer specially designed for list processing, specifically, a 7090 modification.

For these comparisons a CAM Model is developed and a highly structured instruction repertoire defined. A relatively small set of operations is proposed from which algorithms for the more complex commands are derived. The properties of the content addressable memory are extended somewhat to include certain properties usually associated with highly parallel machines (e.g., the capability to communicate with neighboring word cells). The CAM is assumed to be a subsystem used in conjunction with a conventional computer. A simulation package was developed which derived execution times for the various alternatives investigated. Various conclusions were formed, several of which follow:

1) The CAM configuration outperformed the conventional computer with speed advantages largely dependent on the application considered. The reported gains range from two (function optimization) to several thousand (solution of partial differential equations, air traffic control and visual pattern recognition).

2) The extensions proposed for implementation of arithmetic and logical commands similar to those proposed for distributed logic (highly parallel, cellular) computers provides a comparable capability with a markedly less complex (and thus cheaper) design.

3) The CAM is suggested as an appropriate element in the variable part of the fixed-plus-variable computing structure.

4) The proposed organization provides dynamic storage features and capabilities for list searches that are more efficient than the list organization methods used in conventional (location addressed) machines.

Another report prepared for the Advanced Planning Division of the Naval Analysis Group of ONR investigates the use of associative memories for two applications: "A Pattern Recognition Process for Bubble Chamber Pictures," and "An Automated Sea Surveillance System." Both these applications require extensive searching of computer memory during solution. Comparison solution times are given for two organizations: a Hardware Associative Memory (HAM) and a Programmed (software) Associative Memory (PAM). A software simulation of associative memory properties was developed for the PAM configuration which utilized an IBM 7094; the HAM design was patterned on the "Fixed Plus Variable Computer" design of G. Estrin.[19]

The evaluation of these designs was based on assumptions of feasibility as reported by current researchers. The reported results are somewhat more realistic than other studies since they include the time required for the processing of an entire problem, rather than only those portions which favor associative memories. The study concluded that:

1) "A 256 to 512 word associative memory may speed up the Data Association Process, which is part of the Pattern Recognition of Bubble Chamber Pictures, and the Merging of Records of Unidentified Ships, which is part of the Automated Sea Surveillance program, by approximately 30%.

   These two processes potentially represent a large part of the workload in the respective systems. The cost of up to a 512 word associative memory is well justified by a 30% increase in speed on a computer of the size of the IBM 7094.

2) An associative memory of 4096 word capacity will speed up the Sailing Plan Generation and Modification Process in the Automated Sea Surveillance System by 50%. However, these processes represent only a small fraction of the total workload of the system."

These conclusions indicate the nature of the types of differential cost decisions which the system planner must make regarding appropriate usage and optimum size of associative memories.

7.3.6.2 Data Correlation

A general classification of computer problem is data correlation. An example of this type of problem is the requirement in many high-track capacity surveillance radar systems to correlate radar (sonar) data with established tracks. It is customary in such systems to keep account of the currently known trackable entities in the computer store. These entities, once recognized, become known as "tracks". The information of interest concerning each track is usually kept as a body of information (as a block of track data) and dynamically updated by current inputs. Examples of track characteristics are track position, track velocity, track identification, threat priority, predicted position, etc.

The association of a set of incoming radar returns with existing tracks is called radar correlation. This function is performed by determining, according to tracking correlation logic, if a given return lies within a volume of space which is centered about a predicted track position. If it does, it is said to correlate.

With a conventional computer the correlation process is quite time consuming since each return must be compared with all track positions. Therefore, the processing time increases approximately as the square of the number of current tracks. However, with a parallel search memory the position of each return can be compared with all track positions in one search time. The track correlation problem solution time thus increases only linearly with the number of tracks. The process is particularly efficient in an associative memory possessing between-limits search capabilities.

This application is one for which a dramatic time advantage can be demonstrated. The processing time ratio is illustrated in Figure 7-17. In the conventional computer, a searched for value can be found, on the average, by a search of $^n/2$ items, where n is the number of tracks. Total searches for correlation would then be approximately $\frac{kn}{2}$ where k is the number of returns processed. This compares with k searches needed by the associative memory.

Another aspect of radar correlation is scan-to-scan correlation, i.e., the correlation of radar returns received on successive scans. Ambiguous and spurious returns are eliminated as a result of this process. Another associated function is the recognition of new tracks and track initiation.

Although no single measure of effective improvement can be claimed for associative memories in this application, such measures can usually be computed as a function of the number of tracks in the system and the relative magnitude of this application to that of the system computing tasks as a whole. It should be noted that the computing power of the usage tends to be applied at times of system stress, i.e., during high traffic situations.

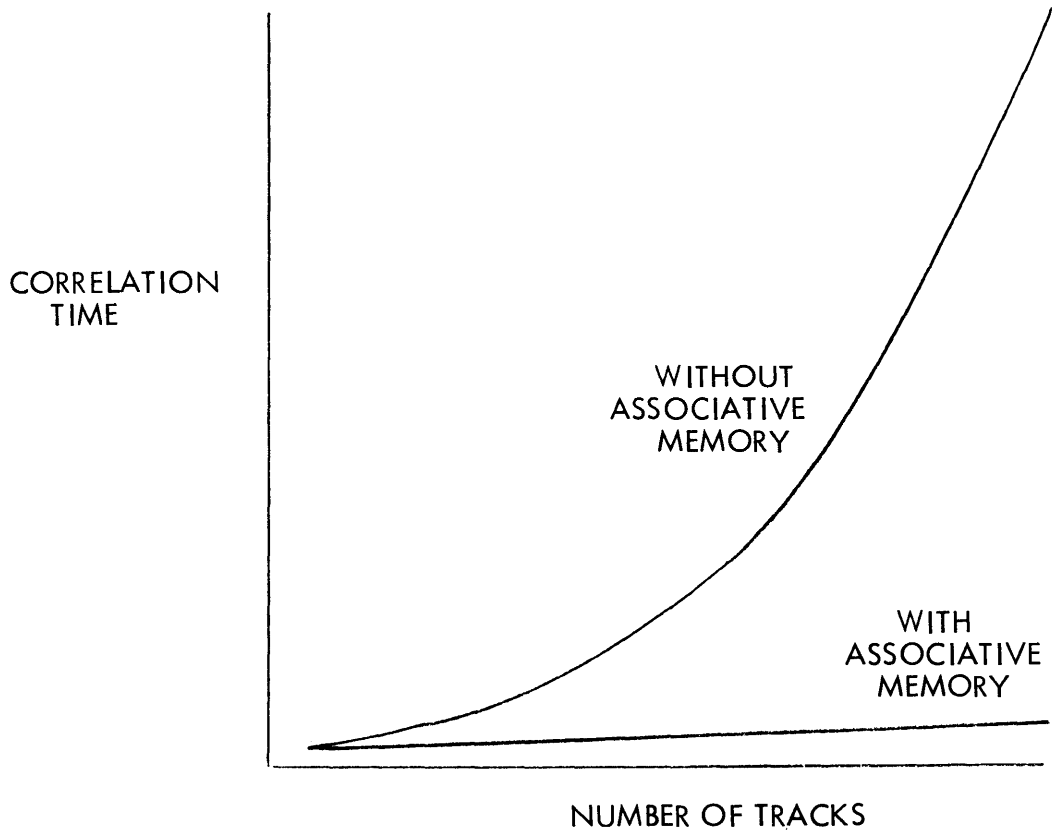


Figure 7-17. Comparison of Track Correlation Speed With and Without Associative Memory

7.3.6.3 Track Data Updating and Retrieval

In the general application areas of surveillance and tracking, other associative memory usages are apparent. In such applications, it is often required that track data be updated frequently from data received externally; it is frequently found necessary to retrieve track information by specifying one or more of the track characteristics. In a conventional memory, the ordering of track data is according to one or more of these characteristics (e.g., track number, track channel number). Sometimes data concerning the same track is kept in different physical locations and ordered differently as may be required. This will ordinarily necessitate that special cross referencing items be maintained. If the ordering criterion is not known (as in the radar data correlation problem), a search is required entailing observation of each item. With the use of an associative memory, track data could be retrieved by specifying any of its known attributes.

A sample tracking problem of this type was used to evaluate the performance of three alternate systems in the report prepared by Gall.[26] The systems evaluated were:

1) Only the USQ-20,
2) USQ-20 with peripheral search memory and
3) USQ-20 with integrated search memory.

The problem was a four-dimensional search for a "track" which satisfied between-limits search criteria for X and Y coordinates, greater than or equal to the Z coordinate and classified by a status item as hostile. Execution time analysis resulted in curves for the sample problem solution as a function of the number of items searched and the frequency of search memory updating. The advantages for the associative memory were substantial although somewhat difficult to quantitatively assess because of other problem assumption details; and because the purpose of the report was to evaluate alternate associative memory systems rather than demonstrate associative memory effectiveness. The derived advantages were also largely sensitive to the number of tracks to be processed and ranged up to 100 to 1 for high-track loads.

7.3.6.4 Sorting

Sorting involves the sequencing or resequencing of data such that the newly formed sequence satisfies some specified ordering relation. Sorting has become a major computer application and one which is quite (computer) time consuming. The importance of the problem is evident from the emphasis placed on research into efficient sorting methods. Sort programs of considerable length and complexity are written, and improvement of method is continuously sought.

Several aspects of the sorting problem are related to the development of associative memories. The most obvious advantage for such a usage is that in many cases the need for sorting is eliminated. Since the data may be addressed by its known characteristics, the need for ordering it is often obviated.

However, the need for sorting would still be required for certain tasks. For example, when sorted data must be transmitted to some other memory device, for preparation of sorted output lists, etc. If sorting is necessary, the associative memory may efficiently be put to the task.

To effect a sort, if the maximum (minimum) search criterion is available, a search is made for the maximum (minimum) value and it is retrieved. This data entry is then effectively removed from successive searches and the process repeated.

A paper by Seeber and Lindquist proposes an "Associative Memory with Ordered Retrieval" and presents a simple left to right binary search algorithm similar to the method used by the human brain to approach the problem.[27] This memory uses a parallel-by-bit and parallel-by-word approach. In an earlier paper by Seeber, a self sorting memory is described.[28] Incoming words are entered in random order and are placed in memory in proper relationship with other quantities. As each word is entered, it is compared with all words already entered. However, this organization requires the use of a dummy register between each associative memory cell to facilitate the insertion process.

7.3.6.5 Information Retrieval

Information retrieval is characterized as a request/response process. The requirement of this type of problem is to obtain information in a timely fashion based on a partial description of the desired data. Another characteristic frequently associated with this problem is the dynamic nature of the data. Data ordering, file updating or sorting may be required frequently. A requirement for command and control system information retrieval is that the response patterns may change dynamically in order that the most relevant information be the most readily accessible. For some systems, the response is required almost immediately in real time. There is some doubt that the typical computer organization and instruction repertoire is well designed for this type of problem.

There are several approaches to the problem. One approach involves the arrangement of data into a stratified structure which facilitates retrieval. In particular, list structures have been found effective. Further benefits can be derived by the development of special instruction sets to provide the manipulative functions needed.

Another approach is to develop special hardware to provide additional capability. Since a characteristic of associative memories is the capability to bypass scanning and sorting in many instances, it is often considered for this application.

However, another characteristic of the information retrieval problem is the generally voluminous size of the data files. Since the development of extremely large associative memories in the future appears remote for economic reasons, the use of associative memories to contain only a select amount of key data is indicated for indexing into lower (or higher) levels of the file structure.

For future tactical data systems, commodious storage capacity will be needed and retrieval of selected data will be required on a real time basis. The advantage of the associative memory for this application would be two-fold: speed of retrieval and a lessening of the requirement for frequent reordering of the data files. The latter consideration is particularly important for problems where there does not appear to be any one best ordering scheme.

However, it must be acknowledged that the approaches to information retrieval by programming methods is highly developed. It is in this area that sophisticated programming tools are said to offer an economical alternative to hardware associative memories. In particular, the use of list structure storage and list processing methods have proved extremely useful. Historically, the same motivations which have given impetus to associative memory development have resulted in the development and refining of list processing techniques.

7.3.6.6 List Processing

The idea of an associative memory was borne of a need to provide lists of information as described in some experiments on heuristic programming by Newell, Shaw, and Simon. In this application it was impossible to predict the ultimate lengths of the various lists so a technique was invented to make the location of the various items independent of any physical location or specific address. The address of the next item on a list was simply stored with each list item, so that sequential list items were scattered randomly in the memory. Unused cells -- ones which were available for assignment -- were "chained" in this manner also. There is no doubt that motivation for the development of contents addressable and associative memories was provided by this application.

As a result of these efforts, special list processing languages have been developed. These languages (e.g., IPL-V) are used to solve many problems in the areas of information processing and artificial intelligence. The primary difference between list processing and conventional programs lies in the different data structures used. In conventional programs, data is ordinarily stored according to some preset ordering scheme. In list processing programs, the data is list oriented such that each data item contains not only the list entry information, but the location of the succeeding (and sometimes, preceding) entry. Another characteristic is that the list structure may evolve during processing; the lists may be dynamically changed in size, structure, or content. The list structures may be highly complex since data may be filed according to different ordering requirements. Although list processing techniques are highly developed, retrieval is time consuming and somewhat inefficient as compared to the direct retrieval obtained by associative memory usage.

A promising approach to list processing problems would be the use of an associative memory as an adjunct to the various list processing techniques. Several advantages can be gained by retaining a list structure in an associative memory environment. The associate memory could increase the capability of present techniques by directly retrieving list elements.

Another possibility is to design a machine especially for list processing. Such a machine organization is described by Wigington.[30] The machine postulated incorporates a search memory to be used as an automatic symbol table. The organization also includes a high-speed scratch pad memory used in conjunction with special working registers and a main memory of 32000 60-bit words.

7.3.6.7 System Programming

The production of system support programs such as assemblers, compilers, translators, etc. is an appropriate application area for associative memories. In such programs, the functions of scanning, table building, table lookup and table search are very common and, in fact, constitute a large part of the computer activity. In a machine featuring an associative memory, these functions are greatly facilitated. Tables used during the assembly process could be stored in the associative memory to improve search speeds.

Another aspect of assembly and compiler programs is relocation of programs and data and the assignment of absolute locations. Contents addressing eliminates the need for some of this activity since data can be entered without undue concern about storage allocation. For example, it is not necessary to determine in advance if contiguous areas of core storage of the appropriate size are available nor even keep track of data locations if retrieval keys are stored with the data. The need for translation from symbolic to absolute storage location may be to some extent eliminated. In a sense, the cells could contain their own names (simply another attribute -- an array of bits which could be used as a basis for search).

The associative memory would enhance the efficiency of current assembly and compiler programs. However, the design of new systems programs oriented specifically around the associative memory could result not only in increased processing speed, but the addition of new capabilities. The development of powerful information retrieval languages is a particular possibility. Addition could be made to the language to incorporate search features. These would be in the form of additional statements and extensions of present ones.

An important trend in computer usage is the development of computer programs to translate present written programs from the machine language of one computer to another. The conservation of programming inventories is exceedingly vital and such programs allow widespread use of common software libraries. The associative memory would find a natural application in the writing of such translator programs.

Interpretive programming would be given new life by associative memory usage. Multi-programming and time sharing techniques would also be enhanced. Another usage may be anticipated in multi-computer systems with the associative memory utilized as a shared memory element by several processors.

Other programs which could make important use of an associative memory are loaders, monitors, debugging programs, overlay processors, and multi-program monitors.

7.3.6.8 Other Applications

The associative memory would be useful for file searching of data for display, communication and data updating. Other applicable areas of interest are ECCM, decoy discrimination, threat evaluation message decoding and general areas such as:

Air traffic control.

Automated intelligence.

Automatic abstracting.

Matrix arithmetic.

Document retrieval.

Language translation.

Compiler writing.

Data retrieval (Medical, Legal).

Payroll.

Problem solving.

Perpetual inventory.

Linear programming.

Automated teaching.

Communications switching.

Airline reservation.

Process control.

Vehicle registration

Mathematical applications.

Many of these have no forseeable usage in connection with tactical systems. However, the fact that there is such a wide potential usage does relate to the question of whether such devices will be available in economically commercial versions. If the range of applicability is broad enough, the motivation for commercial development will be provided so that successful implementation will not depend entirely on subsidized development.

7.3.7 Conclusions

Mechanization studies to date have not justified large scale implementation efforts nor resulted in decisive acceptance of the idea of associative memories. However, many designs have been considered and logical organizations described. Associative memories with contents-addressable, parallel-search capacities unquestionably provide a potential capability for a number of applications of interest to future tactical system planners. The absolute advantages in terms of differential cost are less clear.

Since these constitute a sizable part of the operating system, it may be concluded that associative memories are more useful for these applications than for general computing problems as a whole. The number of applications in other areas is sufficient to suggest that motivation exists for continued interest, development, and implementation of associative memories largely independent of military or governmental subsidization.

The advantages of associative memory usages are those of speed and flexibility. Since it is possible to retrieve desired information at a rate which is independent of the amount of memory searched, associative memories may save substantial time in some applications and provide a good tradeoff of storage saving in others. The associative memory is especially attractive from the logician's point of view. A case can also be made for increased ease of programming since the requirement for certain housekeeping functions concerned with problems of addressing are simplified. However, this is offset by the necessity for using techniques which are currently unfamiliar to the programming community. Thus, if it can be anticipated that a new type of programming problem will be created calling for a new class of computer instructions, the resulting programming efforts will also yield benefits in yet unforeseen efficiencies.

Frequent updating of data bases constitute a large part of the processing task required of command system computers. This task would be facilitated by an associative memory since search may be made on the basis of varying search characteristics or upon status change indicators. The associative memory would also prove efficient for retrieving data for the output of summary and detail information.

The drawbacks are largely those related to implementation difficulties which are currently evident and the attendant costs associated with development and production. It is obvious that the initial cost for workable associative memories will be much higher than for conventional memories of equivalent size. Even after successful installation of production models in working systems, this ratio may not be expected to change appreciably. However, the important cost consideration is that of total system cost vs. overall system capability. It is in this context that the utility and economy of associative memories will ultimately be weighed.

The question resolves to one of availability at reasonable cost and of appropriate usage. The associative memory should be employed in areas of application for which its properties exhibit unusual capability. Its use must be justified in the command/control environment either by usual cost/capability ratio criteria or by providing a unique computing capability at critical times. In this connection, it is important to note that the power of the associative memory is brought to bear in high traffic situations and situations of system stress, rather than simply providing a largely unneeded addition to excess margin capacity. Associative memories possess certain properties that make them more powerful in several applications. Should they prove as useful as expected, they will significantly influence system design.

For a number of applications for which associative memories are considered, software techniques have been devised to a high degree of sophistication to provide a similar capability and in some cases, to simulate the operation of an associative memory. However, either speed or storage must be sacrificed to obtain equivalent capability. In most cases the software implementation depends on a careful ordering of data to facilitate retrieval involving periodic reordering and file updating. The possibility of combining these highly developed software techniques with a hardware associative memory is a further possibility not to be overlooked.

From a system standpoint associative memories may provide a unique capability in special situations for which physical or environmental constraints determine unusual processor requirements. They may be used peripheral to the main computer or integrated into a special computer organization including a hierarchy of memory elements. In time, the associative memory may be provided as an optional component by computer manufacturers.

Several design characteristics appear to be advantageous for associative memory application. The tendency toward long word length was noted. It is also desirable to build associative memories in uniform logical modules in order to reduce fabrication costs and to facilitate maintenance. Extensions of the properties of associative memories provide additional capabilities other than searching only (e.g., self sorting, parallel arithmetic, etc.). The economic advisability of these extensions is not clear at present.

The approach taken to discover desired characteristics should include the development of measures of relevance for individual applications. Simulation and analysis techniques should be used to investigate various logical organizations and to consider the effects of these variations on the individual application.

Literature is currently divided among the topics of associative memory mechanization, design of associative memory cell circuitry and associative processor organization. Relatively little is available in the form of application studies. Such studies could be undertaken profitably to develop tests of merit for various designs and to compare associative memory performance with conventional memories in various areas of interest.

We cannot anticipate clearly which designs will prove most economically feasible in the next decade but would suspect that the simpler ones will become available first. The cost of design is determined largely by the complexity of the logic associated with each memory cell rather than the total number of elements involved. This argues for elemental type logic augmented by micro-programmed combinations. Cost considerations may dictate the use of relatively slow elements. It is quite possible that a very fast (and efficient) system operation could result from the use of moderately fast elements at a reasonable cost. It seems unlikely that very large associative memories can be used to economical advantage as primary memory elements in the time period 1970 to 1980. However, an associative memory of modest size would provide excellent complementary characteristics for use in a system composed of other memory and computing elements. It is suggested that such a memory be regarded as a likely candidate for inclusion in future tactical data system designs.

7.4 STORED LOGIC AND MICROPROGRAMMED COMPUTERS

7.4.1 Introduction

For a number of years the term stored logic has been equated through usage with microprogramming. Although the literal definitions of these terms, if they could be agreed upon, might indicate that a distinction should be made between them, it would be a minor one; perhaps, simply, a matter of the point of view.

Historically, the term microprogramming is attributed to Wilkes. The greatest area of agreement concerning the definition among writers on the subject is that it is difficult.[31,32,33] For the most part, the question is side stepped and earlier definitions are cited.

The difficulty is that the concept was initially seen as a radical departure from conventional design, was implemented somewhat incompletely (in terms of the original concept) and has since come to be thought of in terms of these implementations. It, therefore, takes on varying meanings and associations in different applications of the design.

The term microprogramming has become ambiguous primarily because the techniques are commonly thought of from two different viewpoints--the engineers' and the programmers'. The engineering viewpoint considers microprogramming as a technique for specifying computer processes in terms of aggregate subprocesses. The programmer defines microprogramming as the capability to control directly lower level logical operation.

An ambitious definition is given by Glantz:[34]

"Microprogram (noun)--a program of analytic instructions which the programmer intends to construct from the basic subcommands of a digital computer; a sequence of pseudo-commands which will be translated by hardware into machine sub-commands; a means of building various analytic instructions as needed from the subcommand structure of a computer; a plan for obtaining maximum use of the abilities of a digital computer by efficient use of the subcommands of the machine."

"Microprogram (verb)--to plan an analytical process in a pseudo-code which is to be reduced to the subcommands of a digital computer; to plan an analytical operation in terms of the subcommand structure of a digital computer; to plan an analysis which will use the subcommands of a computer in an optimum fashion."

A conclusion to be derived by the various definitions is the relative nature of the term. If "command" is read for "sub-command" and "code" for "pseudo-code," the definitions given by Glantz could easily apply to the word program. The difference turns out to be one of degree.

The definition hinges on the words "subcommand" and "subcommand structure" by which is meant simply the manipulation of smaller elements of logic than is usual. The term pseudo-command, although a hackneyed term, may mean almost anything, and is used here to indicate that the code is different or at least unconventional.

Initially, Wilkes envisioned a design somewhat more specific. He conceived the possibility of dynamically alterable instruction sets incorporating the use of two control matrices, a "connection" matrix and a "sequencing" matrix. One matrix would determine a number of control states and the other would select the specific micro-operations for a particular state. The micro-operations performed would vary, depending on which set of control logic was in effect.

From this idea of variable logical machines which depend on the state of a control matrix, grew the notion that the programmer could determine a unique order code by combining basic building blocks of logic (variously called, micro-operations, micro-commands or sub-commands).

The concept is derived from the fact that the typical machine instruction consists of a sequence of basic elementary operations which are, however, fixed (or wired in), i. e. implemented by hardware. These sequences are often complicated and intricate. It was felt that a basic defect of the conventional machine was the probability of the superfluous performance of certain of these subcommands serving no useful purpose in the computation involved.

It was, therefore proposed that the basic machine operations be made available to the programmer. It was recognized that the selection of these basic elements would be of paramount importance in order that the combinative properties of those chosen would allow the programmer to develop a powerful logical machine. In effect, the logical design of the instruction set would be done by the programmer.

Mercer defines microprogramming as "the technique of designing the control circuits of an electronic digital computer formally to interpret and to execute a given set of machine operations, as an equivalent set of sequences of micro-operations; elementary operations that can be executed in one pulse time."[35]

This would tend to place the responsibility in the hands of the logic designer and there is, perhaps, a continuing validity in this viewpoint. However, the original fascination of the concept lay in the possibility that the order code could ultimately be chosen at will by the programmer.

The "one-pulse" criterion, however, is considerably diluted in later developments, although, one of the characteristics of stored logic is the relatively small number of clock pulses per computer instruction. Furthermore, the concept of programming using individual microcommands is not strictly realized (in the sense of a one-for-one sequencial specification by the programmer). Rather, the programmer ordinarily specifies that a particular set of microcommands occur (perhaps a dozen or so). He may specify explicitly that a particular one will operate, but usually in combination with others. He may modify in a sensitive manner his choice of microcommands, and may combine them in many ways. In this respect, however, the stored logic computer may not be so different from the conventional computer which also may have a sensitive control of operation (with various modifiers in its instruction word). Indeed, an occasional debunker's pastime is the "explaining away" of the difference attributed to stored logic computers in conventional terms.

Nevertheless, we shall attempt to characterize the development by describing the successful commercial adaptations of the principle and to indicate certain directions that the development of this concept may take. For although there does not appear to be precise agreement as to what constitutes microprogramming or stored logic, and further, whether intrinsically it is a good design, the effects of the development to date are undeniable and the future implications are far-reaching.

It will be seen that certain of the early motivations for this type of design are no longer so compelling due to other developments (mostly hardware), and that certain other trends have perhaps reinforced the reason for its continued use.

7.4.2 Descriptions of Current Stored Logic Computers

Rather than attempt a rigorous or composite definition of stored logic, it is perhaps more instructive to consider the common characteristics of the various stored logic implementations, and to indicate those attributes that, it is generally agreed, characterize its development. Parenthetically, it might be noted that the earlier thinking in some respects is the more sophisticated and is perhaps deserving of attention as a sub-topic in the somewhat neglected field of basic computer organization.

The subject of stored logic was presented in a series of articles in the February, 1964 issue of Datamation, and the material contained there was drawn on in preparing this report.[36,37,38,39,40] The approach taken in these articles was that of describing the commercial machines which were currently marketed as stored logic machines; and the concept is described in terms of these machines. These computers are the TRW-130/133/530 computers, the PB-440 and the C-8401.

However, these computers in some respects are as different from each other as they are from the conventional computer (with which stored logic computers are invariably contrasted). And, perhaps, even more, they depart from the original concepts of stored logic and micro-programming as described by Wilkes. We will, however, examine the characteristics of these machines briefly, noting the common attributes and the distinctive features of each.

1) <u>TRW-130 (AN/UYK-1), TRW-530, TRW-133</u>
   The TRW-130 is the forerunner of this family of computers. It was initially designed under a Navy contract to serve as a militarized multi-general purpose computer to be used primarily for shipboard use. The considerable success of this computer was probably due more to other design characteristics (small size, militarized construction, ruggedness, ability to operate with high reliability under adverse environmental conditions),--than to its stored logic design. It is claimed, however, that the stored logic method permitted a simplicity of hardware

which would have been impossible to implement economically if a more standard design had been adopted.

The TRW-130 contains 8192 words of 15 bit storage with a six microsecond read-write cycle. The TRW-530 is very similarly organized but has an 18 bit word and certain additional logical options due to the longer instruction word. The TRW-133 incorporates the same design as the TRW-130 but is three times as fast, with a two microsecond cycle.

Operation may be thought of as occurring on three levels in the TRW machines; the microcommand level, the machine instruction level and the interpretive level. Microcommands are not accessible individually to the programmer although he specifies them in combinations (explicitly and implicitly) at the machine code level. The machine code command is given the name Logand (L<u>og</u>ical Comm<u>and</u>) and occupies one word of computer memory. A string of logands may be combined to form a routine called a Logram (L<u>og</u>ical Pro<u>gram</u>). These routines which are written in a closed subroutine form operate in a sequential fashion and are called into operation by the programmer specifying a list of routines to be operated (a Logram Calling Sequence). When the computer is used in this manner it is said to be operating in the interpretive mode.

The logical organization features accessible working registers which are available to the programmer for the various machine functions to be performed, sometimes interchangeably. It is the individual transfers between these registers which are recognized as the microcommands which are defined for this machine. These registers are used for arithmetic, memory addressing, logical, and control transfer purposes, and for input/output and temporary storage. For example, the

P register is used as an addressing register, as an extension of the arithmetic register, as a shifting register, contains the quotient in division, the least significant part of the product in multiplication, and also acts as an instruction counter for the interpretive level of programming. Incrementing logic is available and, therefore, indexing and program counter sequencing may be assumed by these working registers.

The programmer of the lower (machine code) level of coding is given the name logrammer, presumably because he is composing lograms. He programs (or lograms) using logands. The term logander, however, is not valid. The coder who uses logram calling sequences is called a programmer.

The instruction word format (logand format) features an address option field that provides unusual addressing flexibility. The address for the current operand is ordinarily found in one of four of the working registers mentioned earlier and is specified by the address option field. Indirect addressing is also available and the combination properties of this addressing scheme are designed to minimize addressing overhead.

The instruction word has various formats and may contain two functional commands (operations codes) per word. These will explicitly call for the execution of particular microcommands. It also contains a control field which has to do with memory accessing (allowing or inhibiting) and address incrementation.

It is sometimes maintained that the combinative properties of this word allows a vast number of unique instructions variously estimated at 8 to 12 thousand. Only a relatively small fraction of these are meaningful, however, and fewer yet are useful. Such sales arguments miss the point since the real strength of the machine involves the way combinations of logands (the more common ones, usually) may be put together, rather than the ability to call on an unusual or esoteric instruction from the large number available.

The higher order interpretive language consists of a string of logram calling sequences. The symbolic names of the lograms are arbitrary in the sense that the programmer can name and design his own. The assembly program will assign the starting addresses for the corresponding logand strings.

The logram calling sequence is specified to the computer by placing in sequencial cells the starting addresses of the corresponding machine code subroutines. Interspersed among these addresses are the addresses of any operands needed. Thus, the interpretive mode code (the logram calling sequence) consists simply of a string of addresses of subroutines and operands. It is said that these lograms correspond to the wired-in instructions of other computers, but a closer look would suggest they correspond more closely simply to closed subroutines, which, in fact, is what they are. However, a unique method of subroutine linkage is used which obviates the necessity for an interpretive routine to sequence the routines. Each subroutine provides its own linkage to the succeeding routine by accessing the address supplied in the calling sequence and placing it in the machine program counter. To facilitate this method, the computer (program) maintains essentially two separate program counter registers which indicate the current position in the calling sequence and the program counter location within the current subroutine.

The interpretive level instruction repertoire is called the Basic Logram Package. The instructions defined by this set resemble those of a one address computer, including single and double precision commands. In addition, special logram packages are available, e.g. floating-point package, matrix arithmetic, etc. These routines require memory space and, in general, only those routines needed in the application should be loaded. Although, initially, wide varieties of instruction repertoires were anticipated, including those which could simulate those of other machines, in practice the Basic Logram Set is most commonly used. In some ways, the interpretive level is the more cumbersome. The most attractive alternative to those familiar enough with the machine operation is to descend to the machine code level using the more efficient methods available there.

The interpretive mode overhead tends to be constant (approximately two logands per logram) which constitutes a rather high cost for the simpler lograms. For example, the logram add command costs 18 microseconds (on the TRW-133), the logand add, only 4 microseconds. Therefore, a combination of the two codes is sometimes preferred, using logands for the simpler functions (add, shift and those commands that can utilize the efficient memory addressing available on level, i.e. load, store, and indexing) and using lograms for the more complex such as sine, cosine, BCD to binary conversion, etc. The computer, when programmed in this way, approaches very closely the typical usage one would expect on our conventional computer using machine code and closed subroutines.

2) PB-440

The motivation for the development of the PB-440 was similar to that held by the AN/UYK-1 designers - the desire to develop the capability to tailor-make instruction sets specially suited to the application. An important feature was added and the claim was made that the first dual memory stored logic computer had been developed. A honogenous memory design, it was felt, would just barely hold its own compared with conventional designs (presumably because of the interpretive mode overhead) and therefore, it would be advantageous to place the strings of microsteps in a special module of fast memory.

The PB-440, then, has two classes of memory; a main memory, which operates at a five microsecond cycle time, and logic memory (or "control" memory) which is a non-destructive biax memory with a one microsecond read time. The minimum configuration of the computer has 4096 and 256 24-bit words of these types of memory, respectively.

The relatively small amount of fast memory was sufficient to define certain basic instruction sets which could be modified or replaced by reading in new ones, and it was expandable in 256 word modules, if desired.

The format of the PB-440 allows two micro-orders, sometimes called micro-steps, per instruction word. Any of 64 separate micro-order codes may be specified. In addition, the format contains two modifier fields per micro-order which will ordinarily indicate one of seven working registers as operand source and/or destination. Here, as in the TRW machines, the working registers are available to program manipulation (i.e. accessible to the programmer) to an unusually large degree.

The routines stored in fast memory are called microutines. They are called into play by the "control sequence" which uses special instructions designed for the purpose. It was noted that the higher language level operation code for the TRW machines was, in fact, simply an address; the address in core memory of the start of the logram. In the PB-440 the operation code will ordinarily refer to a mic-routine by number (i.e. 1 of 64), and utilize a jump table to facilitate rapid micro-instruction interpretation.

From a hardware standpoint, two-level programming is a fiction, since the computer will always remain on one level (i.e. the lower). The interpretive level (which is sometimes referred to as pseudo-code) consists of a macro-instruction control sequence which simply specifies which subroutines are to operate. This is true of all the computers discussed in this section.

Special instruction sets include a systems-oriented command list, a scientific/engineering problem-oriented command list, and a FORTRAN set. These are interchangeable by computer memory loading. The instruction sets are normally stored in fast memory but are also executable from main memory. The micro orders are tailored to recognize various data formats such as floating point, or sign-magnitude numbers, and alphanumeric characters.

Program optimization involves the use of the time between main memory accesses, referred to as "shadow time," during which useful computing may be accomplished (as long as it does not involve further main memory access).

3) <u>C-8401</u>

The dual memory concept is also implemented in the Collins Computer, the C-8401 Data Processor. The microprograms are stored in the fast memory, called the instruction memory. This memory is composed of 1024 36-bit words with a read time of one microsecond. The formats of the instruction word allows either two or three transfers to occur. The transfers relate to the exchange of information between exchange registers, certain of which are associated with logical or arithmetic functions. Each of the basic operations is associated with a particular register. A large number of working registers are thus made available to the programmer.

The main memory is composed of 4096 (expandable in 4K modules) 16 bit words with a five microsecond cycle. Thus, up to 15 micro-instructions can be performed during each main memory cycle time. Notice that this assumes a micro-instruction to be a part (or field) of a computer instruction word.

Macro-instructions are stored in main memory and constitute a higher level problem-oriented language. The interpretive mode linkage is effected by an interpretive routine called RNI (Read Next Instruction). This routine maintains an address counter which is stored in one of the exchange registers. It also provides branching in the instruction memory to the subsequent micro-programming to be performed.

The C-8401 was designed primarily as a communication network processor. One of the distinctive features of the machine is the ability to control I/O operations from many sources at the same time. Although this is not unique in modern computers, it is facilitated to an unusual degree by the computer design which incorporates the external exchange registers for this purpose. Input and output is specified by separate microprograms selected by the RNI microprogram in the same manner that other microprograms are activated. The machine was designed with a single application in mind but is suitable for other applications by software modification. This conforms to the basic rationale of the stored logic principle.

7.4.3 Characteristics of Stored Logic

To the general observation that programming may be undertaken on a lower level of abstraction on stored logic computers, using generally smaller logical elements, we shall add certain other characteristics of the stored logic implementations to date.

1) Multi-level programming - interpretive operation is featured on each of the computers discussed. Although complete programs may be prepared on the machine code level, the machines are specifically designed to facilitate subroutinization.

2) Multiple Instruction Counters - In accordance with 1), above.

3) Relatively few clock pulses per computer instruction.

4) Accessible Working Registers - The internal registers of the machine are available for minute manipulation involving transfers, temporary storage, addressing, as well as arithmetic computation and control.

5) Adaptive instruction sets - All claim the feasibility of custom made instruction sets to suit individual applications.

It is noted that there tends to be fewer clock pulses per machine instruction. Usually, however, the clock pulse contains several micro-operations and the compounding of useful functions is considered a design advantage. This doubling up of logical operations is seen in the fact that all three computers allow at least two command functions per computer word.

The PB-440 and the C-8401 have in common the use of fast control memory modules for the storage of the stored logic routines. It should be noted that the development of fast (control) memory modules for special purposes is not a unique stored logic feature. It is a common characteristic in recent entries in the computer field.

And, finally, these machines have the common characteristic of admitting to being stored logic computers. The implementation of micro-programming techniques is more widespread than the number of machines which admit to being stored logic would indicate. Stored logic is now claimed only by those manufacturers who are committed to it. No longer can much benefit be derived from claiming it as an innovation. If the term stored logic does not survive, it will probably be because of semantic difficulties and the current uncertainty among computer manufacturers as to whether the designation has positive or negative sales value. The term microprogramming is somewhat more acceptable currently and probably more descriptive.

Micro-programming is a prominent design feature of the IBM System/360. The emphasis is not on custom made instruction sets but rather on instruction repertoire compatibility and standardization. Several other manufacturers have indicated willingness to conform to this new instruction repertoire and, in at least one case, the translation will be achieved with micro-programming techniques.

7.4.4 Other Current Usages

Although the number of computers which may be described as stored logic machines is small, the number of computers which use micro-programming in some sense is considerable. The use of micro-programming is particularly evident in connection with the development of read-only fixed stores. Discussion of read-only memories is found in other sections (7.5.2). Motivations are also seen for the use of micro-programming in connection with multi-programming and with the development of associative memories. It could also be descriptively applied to certain programming aspects of highly parallel computers.

7.4.4.1 Micro-programming and Associative Memories

Micro-programming may receive renewed interest in conjunction with other developments. For example, with the impending advent of practical models of associative memories, interest in micro-programming may be renewed. Micro-programming and associative memories may turn out to complement and reinforce each other for the following reasons:

A basic rationale for the advantages of the stored logic design assumes that the hardware commands are to be kept simple to decrease cost, and that the software development, i.e. combinations of microcommands, makes up the difference in capability. However, with the associative memory design calling for the computer logic to be to some extent distributed, we are confronted again with subcommands of an extremely elemental order. The hardware/software cost ratio is altered by the fact that the logic is distributed (i.e. repeated) throughout memory. Further, if attempts are made to increase the distributed logic complexity, the resulting cost is multiplicative as a function of the size of the memory. This suggests that the logic remain simple and that where possible commands of greater logical power should be derived by combing the smaller subcommands (micro-programming). Examples of this type of basic command combinations are developed by Davies [16] and Fuller and Estrin.[19]

7.4.42 Multi-programming with Micro-program Control

The system design of CIRRUS, a multi-program computer with micro-program control is described by Allen and others.[41] This organization uses a prewired read-only memory to contain the micro-programs, a main memory, a set of general-purpose working registers and a micro-program unit. These elements may be connected in various ways providing unusual flexibility. The fixed store contains 4096 words of 36 bits with a read time of 0.3 μsec. (This is considerably more extensive than the fast memories of other dual memory designs.) A main memory of 8192 words is also provided.

The distinctive feature of the CIRRUS organization is that it is designed for efficient concurrent execution of several programs. It operates on a priority basis with each program having the ability to preempt programs of lower priority. The time-sharing decisions are determined by the central processor, and switching is achieved by microcoded interprogram routines. Allowance is made for the operations of up to seven programs.

Development of this design was sponsored by the University of Adelaide, South Australia.

7.4.4.3 Micro-programming and List Processing

The organization of a computer especially for list processing applications is suggested by Wigington,[30], and by Muth and Skidmore.[42]

The first paper proposes the use of a conventional random access memory and a set of special registers. The constituent microsteps for several macro-instructions (INSERT, DELETE, SEARCH and EXECUTE) are given. In the second paper, a more complex organization is proposed including a search memory, a fast scratchpad, and a large conventional memory. Here again, microcommand sequences for basic list processing command functions are described. Specification of basic processes for mainpulating list structures by combining elemental commands is a characteristic of the current list processing languages IPLV, LISP, COMIT.

7.4.4.4 Micro-programmed Control for Computing Systems

This topic is the title of a paper prepared by Gerace reporting on the design of the CEP computer built at Pisa, Italy.[43] This represents one of the few efforts to design a large-scale micro-programmed system. It is described as an extension of Wilkes design. The cycle time varies according to the micro-operation to be executed. The variable cycle timing system permits a speed advantage of this synchronized computer very nearly as great as that of an asynchronous one. This computer organization is also considered efficient for multiprogramming operation.

The control of computers by micro-programming methods is of evident interest elsewhere in Europe. At a Symposium on Advanced Computer Organization held during the IFIPS Computer Conference held in Munich in 1962, the use of micro-program control of computers was advanced by Wheeler (UK).[44] Wheeler suggests the use of read-only stores to control register transfers and sequencing functions. It is judged that error detection is facilitated in this approach to control. At the same symposium, Yaroslavsky and Timofeev (USSR) proposed the use of a similar fixed memory for storing micro-programs as a control signal source for the operating circuits. An interesting extension is the use of two or more of these computers operating in tandem with micro information exchange between them. This provides speed advantages and contributes to reliability by facilitating search for failing elements.

7.4.4.5 The SD-2Computer

This computer was designed to fulfill a particular environment, space and capability requirement with emphasis on ease of maintenance.[45] The computer consists of a 4096 word memory and a diode storage micro-program unit. The word length is 19 bits with a five bit operation code field. The emphasis of this machine organization is on simplicity of design in order that the computer be readily understood (to be easily maintained). This computer is intended for use as an on-line controller.

7.4.4.6 The KT Pilot Computer

An experimental computer was developed jointly by Kyoto University and Tokyo Shibaura Electric Co., Ltd., which is patterned after the Librascope SD-2 Computer.[46] A unique feature of this computer is the mechanization of the fixed memory. This memory consists of 256 x 80 phototransistors packed into a small rectangle used in conjunction with a punched card. By replacing the punched card with one of a different pattern the computer operator can effectively change the micro-order repertoire. Other types of fixed memory are also contemplated to provide both fixed micro-programs and alterable (micro-programmable) logic.

7.4.4.7 IBM/360 System

One of the most extensive current uses of the micro-programming technique is seen in the design of the IBM/360 system.

Read-Only Storage, (ROS), is used for all models up to but not including the Model 70. These devices contain micro-programmed routines which provide the advantages of instruction compatibility throughout the product line, and flexibility in terms of compatibility with other IBM systems.

This usage of micro-programming qualifies as a stored logi- machine from the engineers (or logic designers) standpoint but probably not from the standpoint of the programmer, since the stored routines are not program alterable.

This usage is described in greater detail in connection wi th read-only memories (Section 7.5.2.3).

7.4.5 Evaluation

The advantages and disadvantages of stored logic as a design principle are difficult to weigh. It could be argued that the stored logic design has not been the most compelling reason for success or failure of those computers which have used it; nor even the most important feature. In any case, commercial success is not a valid indicator of design excellence since the two seem to correlate only casually.

The advantages are perhaps most often summed up in the word "flexibility"; flexibility in the sense that varieties of instructions may be produced; that there is a selection of programming methods; that the instruction repertoire may be changed by reading in a new set of microsteps. Whole "logical" computers may be designed to suit particular problem requirements; other computer repertoires may be simulated to retain software investments; and special instructions can be designed as needed and added to the growing library of routines. The user has the freedom to select an instruction set best suited to his application.

Stored logic appears to offer certain cost savings to the manufacturer. The less complicated control logic, the lower number and types of components, (common control circuits replace individual circuit elements), together with the opportunities for standardization of component modules make it intrinsically an attractive design.

Another advantage which was cited by early writers is that the order code may be changed late in the development of a new machine. And, of course, the interpretive language can be modified even after it is built. This reflects the early concern regarding rapidly changing instruction repertoires. Thus, stored logic was seen as a way of delaying obsolescence. Currently, however, there is a tendency to perpetuate code structures, at least among families of computers, in order to maintain compatibility.

Most applications of the stored logic design have been closely associated with interpretive operation; and so the interpretive penalty is often cited as a primary drawback to the method. It is claimed, however, that the stored logic design is specifically aimed at reducing this overhead and it is probably true that interpretive operation is facilitated by this design.

Although the interpretive operation is considered the primary programming method, the lower level machine code is sometimes preferred. This is occasionally necessary to extract maximum efficiency for critical computation.

The interpretive overhead cost must also be weighed in terms of storage as well as execution time. For, in the case of stored logic computers, the (interpretive) instruction repertoire must be stored. In the case of the dual memory machines this storage is quite expensive. This argument may be turned around, however. The instruction repertoire saves storage in the sense that subroutines save storage. And the dual memory machines assuredly have a compelling reason for storing the instruction set in fast memory.

The primary objection to stored logic computers is that it is difficult to program them. It is felt that the logical complications that must be dealt with are enormous; that the programmer should not only have a thorough understanding of the subcommands, but should have a knowledge of the internal logic of the computer and even the circuitry involved. It is said that micro-programming is not intended for the casual user.

It is likely that the actual difficulty of programming stored logic computers is exaggerated. Although it takes a little longer to develop facility at the lower language level, programmers experienced in micro-programming are usually enthusiastic. They consider it challenging; and sometimes it takes on the characteristic of an intellectual recreation. However, what is often overlooked among those who (modestly) insist its a "snap," is that, while it may not seem more difficult to them, it will very likely take considerably longer to write a string of code by micro-programming than it would with conventional code. The apparently greater latitude to compose elegant code even when machine efficiency results, can sometimes turn out to be a false economy in terms of work accomplished per unit cost.

However, it is argued that once the software is developed to provide the desired interpretive instruction repertoire, programming is as easy as for any other computer. The theory is that a small team of micro-programmers (perhaps one) may serve to prepare all special instructions that may be required, and will generate new repertoires as the need is seen. This is a perfectly valid point of view, and the capability to tailor-make an instruction set is certainly one of the most powerful arguments for micro-programming. Unfortunately, however, this kind of software has turned out to be extremely expensive. This is due not only to the cost of the programming effort but also to the concomitant costs of library maintenance, documentation, and attendant activities associated with users' groups and software development generally.

It is usually found more expedient to use combinations of instructions already available than to develop new ones that are more efficient. Private instructions make the rounds unofficially (usually to avoid the bother of getting them accepted as a part of the library), and standard usages become difficult to maintain. The concept of private order codes for variable machines does present some procedural problems. Too much flexibility may be a disadvantage.

The consensus among those who are familiar with the cost trade-offs involved tends to suggest (reluctantly) that stored logic as practiced by the programmer is not paying its way. A partial solution is to not allow the applications programmer the discretion of instruction repertoire alteration. He is presented with a "logical" machine that is unalterable. This unfortunately tends to negate the basic advantage of micro-programming, i.e., flexibility. Another approach is to fix the stored logic (a contradiction in terms?) in the machine. At least one manufacturer is using this approach, and has, in effect, a plugboard type of stored logic modularity.

The stored logic design should be evaluated in the context of various other developments affecting its utility. One important consideration has been the changing size both logical and physical, of hardware components. Initially stored logic received considerable impetus from the fact that hardware replacment of large component modules was expensive and sometimes difficult. Use of smaller logical elements was found to be more economical. Lower component count, standardization of pluggable replacements, and savings in control logic; these factors were all felt to be especially compatible with the stored logic design.

A counter trend seems now in effect which suggests that larger and more complex logical components may be produced now at a fraction of earlier costs. This, together with the tendency towards miniaturization, may limit the degree of divisibility that is economical. As the cost of hardware components decreases, the motivation for small micro-logics may be expected to diminish. In this connection, it is noted that the ratio between hardware and software costs is changing. There is little evidence that the latter may be reduced in the same dramatic fashion as the former.

With the increasing cost of software, use of existing software inventories becomes very important. One approach to this problem is the development of translators to allow programs written in the language of one computer to be executable on another. Stored logic machines are very amenable to this type of implementation and, as the "host" computer, will not pay so severe an execution time penalty ordinarily as would one with a conventional design.

Perhaps in the larger view, it is not too significant if the stored logic concept is maintained as an entity (although the term micro-programming is almost certain to continue to be applied to whatever seems convenient and appropriate). The advances associated with this development; language flexibility by sensitive manipulation of small logical elements, standardization of hardware components, dual or multiple memories (hierarchies) to suit varying computational demands, and translations of language repertoires to use software inventories, are likely to be of an enduring nature in computer technology.

The concept has diverged in development and has been diluted in implementation. It has turned out to be a variation rather than a radical departure from conventional design. Investigation in this direction is incomplete, however, and the techniques involved are certainly worthy of continuing study in consideration of the larger topic of computer organization. Perhaps the development will go full circle with a new look at Wilkes' control matrices.

7.4.6 Conclusions

The terms stored logic and micro-programming have become somewhat ambiguous. This primarily is due to the fact that they are commonly thought of from two different points of view; the engineers' and the programmers'. The engineering viewpoint considers micro-programming as a technique for specifying computer processes in terms of aggregate subprocesses. The programmer defines micro-programming as the capability to directly control lower level logical operation.

Stored logic has been a considerable influence on the design of computers and will continue to influence computer design. Flexibility is the principal claim of stored logic proponents. They claim that the computer can be tailored to the problem. By tailoring an instruction repertoire to fit an application, execution speed and programming efficiency may be increased. The price, however, is paid in terms of increased software systems cost and the costs which are incurred due to the lack of a "standard machine." Applications programming costs may be lower because of "tailored machine"; however, the software investment is higher. The programmer has a greater flexibility with micro-programmed computers since he can increase the speed/cost ratio by an expenditure of programming time.

The theoretical advantage of providing several alternate machine order codes is seldom exploited. Tailor-made instruction sets are usually not forthcoming from the manufacturers, and the user cannot usually afford this type of effort. For most applications, optimizing the instruction set from problem to problem is difficult to justify. The advantage is seen only when frequency of use warrants these techniques and when the changing of repertoires is reasonably convenient (e.g., pluggable instruction repertoires).

Micro-programming currently is more often used for purposes of standardization of repertoires rather than for diversification of repertoires. This is seen in the use of micro-programmed routines for simulation of one computer by another, and, as in the case of the IBM/360 system, to provide product line instruction compatibility.

Stored logic has not gained public computer acceptance. Almost universally and uniformly, sales successes have been due to factors other than the stored logic idea itself. For instance, in the case of the AN/UYK-1 a more important sales fact has been that the computer is Mil Spec, NTDS compatible, and officially Navy-designated. TRW, Packard-Bell, and Collins, with modest computer images and modest public relations, have not been able to convince the technical community. This fact clouds the issues and presents many psychological blocks to stored logic acceptance. Currently, the degree to which the technique has been developed is very low. If one of the larger companies were to actively pursue the technique, the complexion of the technology would change considerably and so would its public acceptance.

Stored logic should not be the foremost criterion in the selection of a machine for future Navy use. Further requests for quotation on future computers should neither demand it nor exclude it. Rather, they should encourage flexibility and adaptability from the logic and programming points of view and let the designer-manufacturer develop an approach best suited.

It should be noted that there is renewed and continuing interest in micro-programming techniques due to the advent of techniques in the use of associative memories, distributed computers, and read-only and fast-read memory devices. The door for stored logic and micro-programming should certainly be left open for ACDS applications. Studies which would develop the best approaches and more quantitatively study pros and cons should be sponsored by the Navy.

7.5 MEMORY HIERARCHIES

7.5.1 General

There is an increasing use of memory hierarchies in computers and computer system design. This is evident not only in the proliferation of the numbers of memory units but also in the varying capabilities and capacities of the individual units. The usual method of categorization is according to speed, but memories may also be classified according to size (capacity), and capability (e.g., random access, read-only, content addressable, etc.). The ideal hierarchy is usually described as one with a fine gradation of speed and capacity characteristics, from small amounts of very high speed storage (registers, scratchpad storage), through high speed memory main storage, through successively larger amounts of lower speed storage and finally, large amounts of bulk storage such as that provided by discs or drums. The steps from one type of storage to the next should be somewhat equal in magnitude. Although it is sometimes suggested that there are gaps in this continuum, memory devices to suit the speed/size/cost requirements of the user are fairly complete.

Although memory hierarchies are characterized mostly by speed and size, other aspects are also noteworthy. Specialization of memory functions, topological relationship of memories with processors, and control hierarchy are also important considerations.

There is a trend in computer design to divide the traditional elements of input, storage, arithmetic (computing), control, and output into separate modular units. This provides possibilities for the manipulation of these units in the system design which provides flexibility and enhanced performance capability, (usually in terms of parallel processing). Another trend is the functional exchange of some of these elements. Thus, it is seen that memory cells are sometimes endowed with logical capability (associative memories, distributed logic machines) and processors are extended to include associated memory elements (shift registers, scratchpads, etc.).

Automatic data transfers among the elements of a memory hierarchy is an interesting possibility. Ideally, the current operands of interest should reside in the fastest, most accessible, memory elements. Transfers between main memory and small high speed storage (scratchpad) units are accomplished which provide faster access to the working registers. On a slower time scale, transfers from main storage to buffer storage or to bulk memory are effected as the main memory becomes saturated. Conversely, there is the possibility of automatically refurbishing high speed storage from bulk storage. A notable example of this is the one-level store concept used in the design of the Atlas computer which allows the user to take advantage of the main core, drums, and an extensive inventory of memory types, as though they were all immediate access.

Transfers may, of course, be accomplished as a function of the computer program but this is not without expense in terms of programmer effort and computer time. If such transfers could be accomplished automatically on the basis of pre-selected criteria (e.g., frequency of use, recency of use, preset index of importance, etc.) the effective computer speed would be increased. Several algorithms are possible to evaluate the chosen criteria; however, only the most simple methods would be amenable to hardware mechanization at a cost commensurate with the resulting time saving.

One of the most interesting challenges involves methods for the automatic utilization of hierarchies of memories so that the programmer need not know the whereabouts of the data to be accessed. The possible freeing of the programmer from the addressing requirement was suggested in connection with associative memories.

7.5.2 Read Only Memories

An important type of memory which is finding increasing application in computer systems organization is the read-only memory (ROM). The cost of read-only memory is considerably less than for read-write memory of equivalent speed. This type of memory is referred to as permanent since the machine using it is ordinarily not capable of changing its contents. The designation semi-permanent is also used to refer to memories which are physically removable, (e.g., card-changing memories) but not electrically alterable. As with associative memories, the basic characteristic is the non-destructive read-out (NDRO) capability. It is possible to combine these capabilities to produce a read-only associative memory.

There is a tendency to equate the terms, read-only memory and NDRO memory. This is not quite accurate since read/write memories may also be NDRO (as is the case with some content-addressable memories). The term read-only is not entirely precise, in any case, since the memory must be written into initially. The term is often used to refer to fast read, slow write memories. One criteria for classification of read-only memories is whether the contents are fixed in construction or are capable of being written into electrically (i.e., mechanically or electrically alterable).

A number of important usages are envisioned for read-only memories, among these are:

1) Storage of micro-programs (stored logic).
2) Storage of arithmetic tables.
3) Multiprogram control.
4) Function generators.
5) Storage of executive control programs.
6) Code conversion (e.g., radix conversion).
7) Storage of tables of branches.
8) Interrupt processing routine storage.
9) Storage of test routines.

Read-only memories have not achieved extensive application because the areas of economic use are somewhat limited. The basic advantage is greater access speed at lower cost. The relative savings obtained from eliminating the additional circuitry to obtain full read/write capability is not considerable enough in the general case to warrant the special system design to use the advantage. In several cases, however, read-only memories have been effectively used, and the design is postulated in a number of proposed systems designs.

7.5.2.1 Kinds of Read-Only Memories

Read-only capability may be implemented by a number of different methods (See Section 6). Basic methods for detecting the stored information are capacitance coupling, eddy currents, electromagnetic coupling (mutual inductance), fiber and twistor sensing. There are various mechanization techniques for these memories, the most common is the card-changeable memory, but other types such as solenoid arrays, thin film, and rope memories are also receiving attention. It is also possible to use conventional storage devices (such as drum storage) with the write circuitry locked out to achieve read-only characteristics.

7.5.2.2 Uses of the Read-Only Memory

One of the important motivations for development of read-only memories is for storage of microprogram routines (stored logic). The development of dual memory stored logic machines is noted in Section 7.4.2. The use of fast storage for operation of pseudo-instructions (macros) constitutes a distinct enhancement of the stored logic design. For several of the dual or multi-memory computer organizations, the fast memory has non-destructive read-out capability and fast read, slow write characteristics. Use of read-only memories for micro-program storage is seen in the IBM-360, Ferranti Atlas and CIRRUS system designs . The read-mostly memory, where the read time is considerably shorter than the write time, is used as the high speed memory element of the PB-440.

Another use of read-only memory is for storage of control programs (e.g., executive or monitor programs) and for input/output processors and interrupt processor routines. They may be used to substitute for a part of main memory or for faster access for large capacity storage at low cost. The primary characteristic of programs which can effectively exploit the read-only memory speed, are those which are relatively immune to change and whose frequency of usage is sufficient to result in significant time saving. Several computers use a form of read-only memory as a control memory, for special input/output parameters to indicate buffer storage regions or for interrupt processor starting locations. It is also an appropriate usage for interpretive programming for storage of tables of branch locations to enable quick switching from routine to routine.

The read-only memory is also considered for storage of arithmetic or function tables. Used in this fashion, the device takes on the appearance of a function generator where the input is one input variable (presented in the form of an address) and the output is a particular function of the input variable . If the read-only memory is also content-addressable, the input variable may also be stored.

Examples of functions which could be stored are sine, cosine, square root, arc tangent, etc. Another possibility is the use of the device for purposes of data linearization in converting engineering units. Effectively, the read-only storage can substitute for computation and allows manipulation of the time/storage trade-off equation. This is particularly evident if the table look-up method is combined with interpolation.

Other applications include code conversion, radix conversion, and multiprogram control. Read-only memories have also been suggested for storage of the dictionary in automatic language translation and for the storage of the program for an electronic telephone switching system. They can be used to effect information retrieval by providing the cross-referencing associations needed. In addition, large read-only memories could be used to contain encyclopedic data which is not subject to change.

In some spaceborne systems, a large read-only memory is used for storing the program in conjunction with a small read/write scratchpad memory. This insures that the program will not be destroyed by malfunctions in other parts of the system and it reduces the weight and power consumption required for the total memory capacity.

In some ground-based computers, a smaller read-only memory has been used in conjunction with a large read/write main internal memory. In this latter application, the read-only memory is used only to store constants, subroutines, or micro-instructions that are changed relatively infrequently. This permits faster access to this information than to information stored in the main memory.

7.5.2.3 Current Read-Only Usages

An example of the use of a read-only memory for several purposes is seen in the fixed store of the Ferranti Atlas.[47] This memory consists of a woven screen wire mesh into which ferrite plugs are inserted which determine the numerical values (i.e., bit pattern) of the store. This store is 8192 words in length and is not electrically alterable. The Atlas fixed store is used for peripheral equipment control and for containing subroutines to perform several of the more complicated macro functions, thus extending the hardware order code. These routines, called extracodes, appear to the programmer the same as basic instructions. Control commands for storage allocation, monitoring, and interrupt processing are also contained in this store. The organization of the Atlas computer is described in Section 7.5.4.

Read-only storage (ROS), as implemented in the IBM/360 system, provides considerable flexibility and code compatibility over the entire range of the System/360 as reported in a paper on the IBM System/360 Engineering. "There were two major reasons for the general adoption of read-only storages in System/360.[48]

1) Assist downward compatibility due to the cost advantages. Read-only storage (ROS) showed an advantage in cost over the circuitry which it replaced. ROS is used primarily in the control section of the system and its advantages becomes more pronounced when more functions are to be performed. Therefore, ROS units showed a method of maintaining full compatibility by allowing complex function even in the smallest systems.

2) Flexibility, such as the implementation of compatibility features with other IBM systems. A unique flexibility is achieved by being able to add to the control section of the computer, when implemented by a ROS, without significantly affecting the remainder of the system. One result is to allow certain System/360 models to operate as another computer, such as the 1401, by adding to but without redesigning the system. This ROS concept can be combined with software to offer almost any performance from pure simulation (no ROS) to maximum performance (no software)."

This use of read-only storage to contain micro-programs is reminiscent of stored logic computers but probably qualifies as a stored logic from the logic designers' point of view rather than the programmers' since the stored routines are not program alterable. Several types of read-only memory are used, and determination of the particular method is not deemed critical in the system design. ROS is implemented for all models up to but not including the Model 70.

Read only memories are seen as memory elements in a number of other computer organizations for containing micro-programmed routines. Several of these are described in Section 7.5.4 including the CIRRUS organization, the SD-2 Computer, and the KT Pilot Computer.

7.5.2.4 Evaluation

The advantages of read-only memory rest in the prospects for development of:

1) Very fast memories of modest size for specialized uses.
2) Very large memories for large quantities of data which require updating infrequently.

From a commercial standpoint, manufacturers have often found it advisable to offer a conventional capability which is attractive to a larger number of potential buyers, rather than a specialized device for which development amortization is spread over a narrower base. This suggests that in some instances, memories with full read/write capability may be available as cheaply as their read-only counterparts. Thus, potential cost reduction must be weighed against the mass production economic factor. This type of engineering economics also applies to associative memories and, in general, to the question of special purpose vs. general purpose costs trade-off. Except for certain instances where memory protection might be desirable, there is little intrinsic value in the "no-write" capability.

The utility of read-only memories, therefore, resolves largely to questions of application fit and costs savings. If such cost savings can be demonstrated, the range of usefulness for read-only memories is sufficiently broad to suggest favorable consideration as components in the memory hierarchy.

7.5.3 Scratchpads

The use of small high speed memories is often recommended to bridge the considerable gap between logic speed and main memory speed. The interposing of such a memory in this way has the advantage of increasing effective speed to an impressive degree if the programs are data stored. Such memories are sometimes referred to as scratch-pads. The term scratchpad is not a very precise one currently since it may be applied to a number of different types of memories. It may refer to a particular segment of main memory which is simply a reserved area, distinguished only by usage, or it may be endowed with special hardware characteristics (e.g., content addressable, read only), or it may be a separate memory unit with special speed characteristic. It is also possible that it is simply an aggregate of registers used for control purposes (sometimes called control memory). An early usage of the term was seen in the design of the TRW-130 (AN/UYK-1) which contained a 64 word scratchpad area in its main memory. This memory is distinguished from the rest of memory in that it is directly addressable by means of a 6-bit field in the instruction word. These cells were used for special parameters, control words, and temporary storage.

Another usage of very high speed scratchpad memories is seen in connection with multi-processing. The scratchpad could be shared among the several processors thus compounding the speed advantage. An alternate method would use local scratchpad memories for each processor. The scratchpad may be thought of as an extension of machine registers and the manipulation power on a micro-logic level may be counted as one of its primary advantages. The use of a scratchpad for this purpose in a diverse memory hierarchy is illustrated in the machine organization of Wigington[30]. This machine is designed primarily as a list processor and includes a search memory and a main memory in addition to the scratchpad storage. The scratchpad is used for manipulating address fields and the search memory is used as an automatic symbol table.

The use of small high speed memories to increase effective speed (whether called a scratchpad or not) is seeing increasing usage and to good effect. The importance of this type of memory usage depends to a large extent on the degree of frequency of use of the high priority data or instructions contained. This usage is particularly powerful if methods for periodically refurbishing the high speed memory with the most critical data are provided. This possibility of automatically mechanized transfers of this sort was suggested in a previous section.

7.5.4 The Ferranti Atlas Computer

The use of a hierarchy of memories (stores) in a single computer organization is illustrated in the design of the Ferranti Atlas Computer.[47] The basic motivation of this design is to allow operation of several programs simultaneously and at the same time allow each user to write his program as if he were the only user of the machine. The machine is said to utilize "one store" concept from the user's standpoint, while from a hardware standpoint a great variety of storage levels exist.

The main storage consists of a large drum. Blocks of 512 words each, sometimes referred to as "pages," are brought into core memory when needed. These transfers are made simultaneously with computing functions.

The core store features four separate access systems. The core is divided into stacks which are interleaved in pairs to allow overlapping of commands. Instructions are brought from memory in pairs, and accesses to the various elements of the arithmetic unit, as well as the memory stacks, are accomplished simultaneously. The parallelism of this operation results in an effective reduction of the nominal cycle rate, which is 2 microseconds.

The fastest store is a fixed (read-only) store with an access time of 0.3 microseconds. An executive routine resides in this store which oversees drum, and other I/O transfers, and performs a highly sophisticated scheduling operation intended to optimize utilization of the overall system.

The fixed store also contains subroutines, called extracode routines. These are used to provide the more complicated functions of the instruction repertoire. This fixed store uses a part of the core store as a private working space for the fixed store routines.

In addition to the memories listed above, there is what is called the V-store which is composed of a set of address registers for control of system transfers. The V-store is also used only by the fixed store routines.

A diagram of the Atlas memory elements is shown in Figure 7-18.

In summary, it is seen that the power of the Atlas derives primarily from a highly developed organization of its memory elements. The system features both multiprogramming and microprogramming concepts. Although the system makes use of many different types of memory, the user may regard these elements substantially as a single level, very large store.[49]

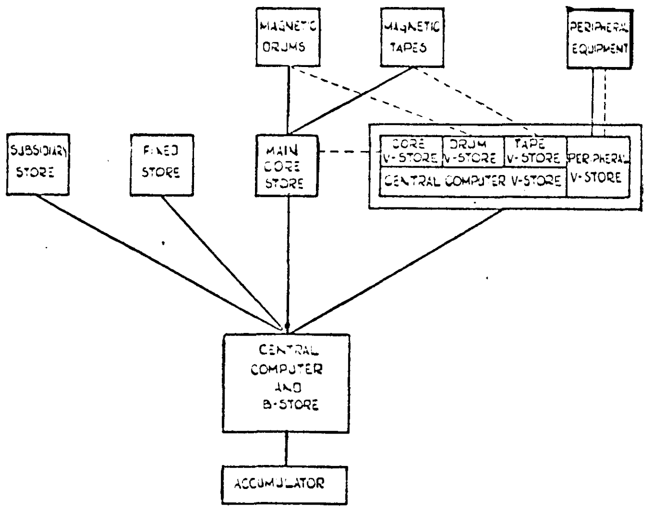


Figure 7-18. Block Diagram of the ATLAS Computer

7.5.5 The Fixed Plus Variable Computer Organization

An experimental computer organization which incorporates the strong features of both general purpose and special purpose computing components is being developed at UCLA. This computer organization called the Fixed Plus Variable Computer organization, first formulated in 1959, is being developed under the sponsorship of the Office of Naval Research and the Atomic Energy Commission.

The Fixed Plus Variable Structure consists of a conventional general purpose computer (fixed components), and a number of special purpose modules (variable components) which may be reconfigured according to the requirements of the particular computing problem. The stated goal of the Fixed Plus Variable Structure Computer is:

> "To permit computations which are beyond the capabilities of present systems by providing an inventory of high speed substructures and rules for interconnecting them such that the entire system may be temporarily distorted into a problem oriented special purpose computer." 50

The variable part of the organization is expected to change from problem to problem and from day to day. However, the same basic hardware substructures may be used in a variety of special purpose structures. This organization is seen as an evolving machine which may be reconfigured both electronically and mechanically. The basic advantage is the efficiency that may be achieved by reconnecting the components either physically or by program into an organization peculiarly effective for the problem at hand. The flexibility of the special purpose devices raises the question of whether the structure is meaningful without the fixed component. The rationale for using a commercial general purpose computer within the structure is to take advantage of the developmental investment available in such computers as measured in terms of libraries of subroutines, procedures for communication with the computer via standardized peripheral equipment, and availability of higher order languages. The IBM 7090 was selected as the initial "fixed" part of the system.

A supervisory control unit coordinates the activities of the various processors and controls data transfers. A conceptual diagram of the fixed-plus variable structure computer system is shown in Figure 7-19.

Programming of the variable part is conventional in that the operations and locations of operands are specified sequentially. However, in addition, attention must be given to assignment of the appropriate device or processor to the individual tasks. The procedures required to select the optimum set of substructures are highly complex; therefore, consideration was given to the development of automatic assignment in the variable structure system.[51] It has been suggested that the programming task should be undertaken by a team composed of a numerical analyst, programmer and design engineer.[52]

The economic feasibility of this design for general use is somewhat uncertain since the costs of tailoring a machine organization for individual problems are likely to be substantial in terms of human effort, physical change cost, and actual execution time. In particular, restructuring requires considerable redesign and construction effort. The utility depends on whether there is a significant number of problems for which physical restructuring results in computing advantage sufficient to offset the cost of such restructuring. There also tends to be some loss of performance due to the requirement of reorganizability. Another problem, in addition to the one of

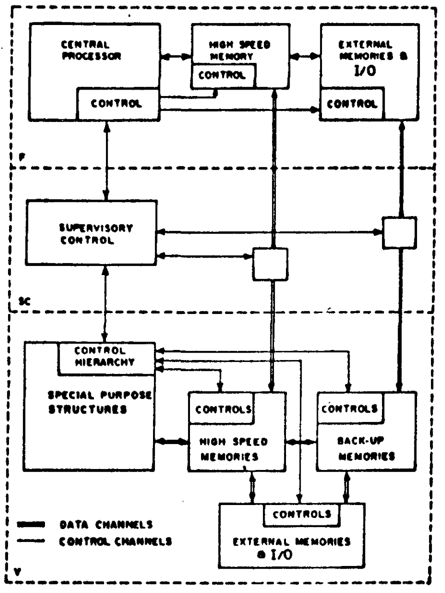


Figure 7-19. Block Diagram of the Variable Structure Computer System

assigning, sequencing, and allocation of tasks, is the question of when to reconfigure. The ultimate programmability of such an evolving organization depends on the definition of procedures which will resolve these questions in a reasonably satisfactory fashion.

Speed advantages of from 2.5 to 1000 have been reported, depending on the problem characteristics, as compared with general purpose computer solutions. Although these speed advantages are substantial, they must be weighed against the costs indicated in the foregoing discussion.

The organization of tactical computer systems in the 1970 period will probably include a wide array of diverse elements in some respects similar to the fixed plus variable organization. However, it is doubtful that the same degree of structural fluidity will pertain. It is inadvisable to reconfigure hardware elements, and, in effect, redesign the system arrangement from problem to problem, except as this can be managed by normal switching and modularization techniques. The dynamics of the type of system change needed for tactical systems is on a much slower time scale than that anticipated by the fixed plus variable proponents.

The fixed plus variable structure is particularly useful as a research tool for experiments in computer organization and for evaluation of new circuits and devices. The same basic inventory of elements can be arranged to constitute a variety of special purpose systems, and as weaknesses of the system become apparent, indicated improvements are incorporated. It could also be used for study relative to establishing better design criteria for building future general purpose computers.

It is axiomatic that usage influences (or should influence) design; but it is also noted that design sometimes influences usage. The latter is the more distorting effect. If design is kept in a fluid state, conceivably, this distortion could be removed.

7.6 HIGHLY PARALLEL COMPUTERS

7.6.1 Introduction

It has been concluded in earlier sections that an important segment of the current efforts toward improved computer organization is in the direction of providing parallel processing capability. The subject to be considered now may be thought of as an extension of multi-computer systems. It is suggested by Comfort that the term, highly parallel machine, is a subclass of the term, multiprocessor, and that a highly parallel machine may be classified as one having more than an arbitrary number of subprocessors, which he selects to be 1000.[2]

By this definition, there are a number of computer organizations which are currently under investigation. To a large extent, they share the same basic potential advantages and the same basic defects. Their utility depends largely on the degree of inherent parallelism in the problems for which they are proposed; and to the extent that the problems can be redefined to exploit this parallelism, the design may be deemed efficient.

7.6.2 Types of Highly Parallel Computers

As was the case with associative memories, highly parallel machine organizations have come to be described according to various designations, each more or less descriptive of the individual development under consideration. Some of these are:

1) Distributed logic computer
2) Cellular computer
3) Iterative circuit computer
4) Pattern recognition computer
5) Spacial computer
6) Orthogonal computer
7) Multi-layer computer

These terms are not synonymous, however. They have distinct, although overlapping meanings.

The distributed logic designation is somewhat the more encompassing and, in fact, transcends the subject of highly parallel computers. The feasibility of a high degree of parallelism is obtained by dispersing the central control logic to the individual memory cells and thus the logic is said to be distributed. The degree of control logic retained centrally is a criterion for classification for the highly parallel designs currently under investigation. If the logic is entirely distributed, it is said to be locally controlled. If primary control resides in a central control unit, the computer is sometimes referred to as a global computer since central control is retained and is broadcast throughout memory.

Associative memories with parallel search capabilities are distributive logic machines since they contain a multiplicity of independent registers each capable of independent logical decision.

The term cellular computer is sometimes applied to refer to the distribution of computing power over a large network of (usually) identical modules (cells). The interest in identical simple units is heightened by the prospects of the development of economical batch processing techniques.

The Iterative Circuit Computer (ICC) was proposed by Holland.[53,54] His multi-layer design is more generally referred to as a Holland machine, currently. The principal characteristic is the large number of homogeneous modules capable of containing many independent programs which may all be operated simultaneously. Other iterative circuit computers have also been proposed which are for the most part modifications of the Holland machine.[55,56] However, these have somewhat more central control than is true of the original Holland computer.

Several pattern recognition computers have been proposed which incorporate a cellular distributed logic organization.[57,58] An early one was described by Unger and is referred to as a spatial computer (SPAC).[57] The orthogonal computer is another specific design proposed by Shooman.[59]

7.6.3 The SOLOMON Computer

The SOLOMON computer was designed at Westinghouse to solve problems which involve a large number of data sets for which the same program is applicable.[60, 61, 62] The designation Simultaneous Operation Linked Ordinal MOdular Network, was applied subsequent to the selection of SOLOMON as the name for this design. It is composed of a large number of identical processing elements (PE's) each of which contains the 64-word storage banks and a serial address. The original design calls for 1024 processing elements arranged in a 32 x 32 rectangular array. Each PE is able to communicate with each of four neighboring PE's. The entire organization is under control of a central processor. At any point in time the same instruction may be exerted on as many as 1024 different data items stored in corresponding registers of the respective storage areas. The modules which respond to any particular command are determined by the state of a mode address field. Thus, the program is executed with varying numbers of PE's participating. Transfer from one program to another is affected by changing the mode registers. The mode registers may also be altered locally, i.e., by intra-module action.

The instruction repertoire includes addition, subtraction, multiplication and division commands. Commands are executed serially in each PE. Communication between neighboring PE's is effected via "edge" registers. Operands may also be broadcast from the central processor to any or all PE's . In this way, storage may be saved in the PE storage banks since quantities common to all PE's can be stored in the central memory.

Potential uses for the SOLOMON computer are reported in a paper by Ball, et al. Satellite surveillance is suggested as an appropriate application area. Speed advantages in the range of from 60 to 200 over current large scale computers are advantageous in numerical weather forecasting, air traffic control, nuclear reactor calculations, photo-reconnaissance, and communication and transportation problems. It's use has also been suggested for problems involving solution of partial differential equations and matrix problems and for function optimization.

In summary, it is seen that the SOLOMON computer is appropriate for problems with the property of having parallel data sets which can be manipulated by the same program. The primary disadvantage is the extremely large number of components required for a workable version. The expense would be a formidable objection and machine utilization is relatively low. The design is supported by significant application studies, however, for several important problems.

An extension of the SOLOMON organization would involve a planar configuration allowing intercommunication between any processing unit and its six nearest neighbors in a three-dimensional organization. This configuration would facilitate the path-building process.

7.6.4 Holland Machine

A highly speculative design for a computer capable of simultaneous operation of an arbitrarily large number of programs has been suggested by Holland. He termed his computer design on iterative circuit computer, but such designs have since been termed Holland machines. The development was motivated to provide a vehicle suitable for establishing a theory of adaptive systems. This work was conducted at the University of Michigan.

The Holland machine differs from other highly parallel organizations significantly in the area of control. The schemes presented in the foregoing sections retain the principle of central control. The Holland machine assumes an entirely local control. Another unique feature is a relative addressing technique which involves obtaining the operand by construction of paths to the desired locations. This is described by some as addressing by pointing.

7.6.4.1 Organization and Operation

The Holland machine is a iterative homogeneous structure composed of a set of interconnected modules arranged in a rectangular array. Each module consists of a multipurpose storage register which may contain an instruction or piece of data; and may be called upon to act as a storage element, an accumulator, or to serve a communication function with its neighbors.

The operation is synchronously controlled and all instructions which are defined as currently active within the independent subprograms are executed simultaneously in a three-phase sequence. The first phase allows for modification of the storage register and for changes in state of the control registers. During the second phase, the operands are obtained. These are linked to the active modules by a sequence of gates. Essentially, a path is opened which is terminated at the module containing the operand. (Other programs can use the same operand modules.) The operands are routed along the opened paths to arithmetic modules, where the operations are to be performed.

During the third phase, the instructions contained in active modules are executed. The operation may involve both the operand derived in Phase 2 and quantities to be found in the associated arithmetic units (A-modules).

The programs can be written so that under local control they can connect with other program segments. They can also be controlled directly from the input source.

Advantages and Disadvantages

The Holland machine is regarded as potentially powerful for most of the applications for which parallel processors are considered. It is a good research vehicle because the parallelism of other designs are also achievable in the Holland machine. As with parallel processors of other types, a speed advantage may be derived for select problems which can be formulated for parallel operation. It has the advantage of operating parallel program sequences which are not necessarily identical, rather than operating the same program on parallel data sets as with the SOLOMON computer.

However, the list of disadvantages at this point in time appear formidable. The foremost difficulty is that the machine as initially postulated may be practically unprogrammable. Unfortunately, the programming art as currently developed finds little applicability for a geometrical organization wherein programs are spatially-organized. Problems of unpredictable interaction of the subprograms, path interference and possible inadvertant "dead ends" appear to be unavoidable. The question of programmability depends largely on whether a compiler can be implemented.

Another apparent disadvantage is the large amount of hardware implied in the machine design. An inefficient utilization of the hardware is also evident since at any one time only a fraction of the componentry is used. The generality obtained by allowing all modules to have identical and multiple capabilities (e.g., to act as storage arithmetic and shifting registers, for instruction interpretation, etc.) is inordinately expensive.

7.6.4.2 Modified Holland Machines

Although the Holland machine was designed primarily as a theoretical tool for the study of adaptive systems, it is considered by several as a prototype for a practical computer. Several efforts are underway to improve the practicability of the concept by selected modifications of the design. Comfort lists the characteristics of the Holland machine which limit its effectiveness and proposes that the following modifications be considered.[2]

1) Extended operation codes
2) Different types of modules
3) Generalized path-building
4) Compiler (development)

By segregating the arithmetic modules and limiting the decoding and arithmetic capabilities of the other modules, he claims that machine size may be reduced by a factor of five and that hardware utilization is improved by a factor of three.

Garner and Squire consider the extension of the 2-dimensional iterative circuit computer to an N-cube arrangement for which each module has N neighbors.[63] Two problems are evaluated--a matrix inversion problem and a path-building problem. The performance of the N-cube ICC is contrasted with the 2-dimensional ICC and with a conventional computer. The N-cube ICC is shown to be markedly superior primarily because it exhibits fewer path interference difficulties.

Pattern Recognition Computers

The highly parallel computer organization finds an important application in those problems for which spatial relationships must be represented. Conversely, this type of problem is one for which conventional computer organizations seem relatively inept. The early motivations for highly parallel machines stem from this realization and the desire to design a computer whose organization is better suited to this type problem.

7.6.4.3 The Spatial Computer

An early pattern recognition computer is described by Unger.[57] This computer is called the Spatial Computer (SPAC) and represents one of the earliest cellular designs.

The organization consists of a rectangular array of modules called principal registers (PR's),and a master control unit which broadcasts the commands to the modules. Each module consists of a small amount of memory and a one-bit accumulator. There is also provision for a limited storage in memory elements located between neighbor cells (including diagonally located neighbors). The modules have the ability to communicate to nearest neighbors (to left, right, above and below) by means of shift instructions. Other instructions provided are add, multiply, invert, and transfer and invert. Also included is a link command which records in the memory element between (PR's) whether or not ones are located in both accumulators.

Illinois Pattern Recognition Computer - ILLIAC III

This computer was specifically designed for the processing of visual information.[58] The computer organization consists of a Pattern Articulation Unit (PAU), a Taxicrinic Unit (TU), and an Arithmetic Unit (AU). The PAU consists of a 32 x 32 array of identical processing modules called stalactites. It also contains a core memory called a transfer memory. This memory is capable of operating as an associative memory. Inter-communication between neighboring cells is similar to the SPAC computer except that the PR's may communicate with eight nearest neighbors.

This design was developed for the particular application of bubble-chamber data processing.

7.6.4.4 Multilayer Iterative Circuit Computer

A multilayer arrangement is described by Gonzalez.[55] The organization consists of three layers each composed of an iterative rectangular network of modules. The functions of each plane are distinct and are associated with a three-phase cycle. The first plane which contains the program to be operated is called the program plane. The middle layer is called the control plane; it interprets the instructions to be operated in the third plane, called the computing plane. The corresponding modules of the three planes are connected by a communication line. Each module contains an accumulator, a register, a decoder and several switching matrixes.

A three phase operation simultaneously on all levels provides a staggered or cascading effect. This provides a look ahead feature which compensates for the delay introduced by the three-phase cycle. In this organization is seen specialization of functions according to layer. Geometrical operations are provided in addition to arithmetic and logical ones. Path building is established by row and column communication lines, which determine current operator and operand locations.

This computer differs from the Holland ICC by having a central control unit and functionally specialized modules. The organization is proposed for use in problems involving spatial relationships.

7.6.4.5 Orthogonal Computer

An orthogonal computer has been described by Shooman.[59] The technique used is called vertical data processing (VDP). Processing is parallel by word and serial by bit. The memory is visualized as composed of columns which may be separately addressed. Multiple data are processed simultaneously over the entire memory on those words for which mask bits are appropriately set. The mask bits also provide the decision functions likened to branching in a conventional computer.

This computer also provides for horizontal (conventional) memory access.

This organization was analyzed for three applications:

1) An FICA payroll computation

2) Number ordering

3) Mnemonic code translation as required in assembly programs.

The speed advantage reported ranged from 32 to 660 as compared with a conventional computer.

7.6.4.6 Distributed Logic Computers

A content addressable distributed logic computer organization designed for manipulating lists is described by Lee.[64] A process of retrieval by cell association is proposed. The design is similar to other contents-addressable designs with the added feature of inter-communicating cells. Data is retrieved by sequential interrogations of the memory cells. The first interrogation locates the first element of the string. The cell containing this element will alert a neighboring cell making it elibible for the next search. The process is continued until the entire string is retrieved. It is noted that the contents of the initial cells may be used as tags or names to identify the information to follow. This constitutes a slightly different form of addressing; names of data are stored along with the data itself. Data may be referred to by name without knowledge of its whereabouts. However, this method does require contiguous storage of list items. This implies problems of utilization of spare storage between lists, and the necessity of periodic restructuring if the list elements must be added or deleted frequently. This addressing method is implicit in other associative memory designs as well.

Lee, in his concluding remarks, presents the rationale that a distributed logic organization may provide a framework for more powerful programming methods and can ultimately ease (rather than complicate) the programmer's task.

"The most obvious asset of such an organization is the tremendous speed it offers for retrieval. Suitable programs can also be developed to make the organization extremely flexible. In addition, we believe what is called the macroscopic concept of logical design, away from addressing, from scanning, from searching, and from counting, is equally important. We must, at all cost, free ourselves from the burdens of detailed local problems which only befit a machine low on the evolutionary scale of machines. Furthermore, because the cells are all identical, mass production techniques should be developed in which a whole block of circuitry can be formed at once."

All the highly parallel organizations described in this section qualify as distribute logic machines,as do the more complex associative memory designs. However, other descriptive designations have been preferred for the other designs.

7.6.4.7 Summary and Conclusions

The general class of parallelism involved in the designs discussed in the foregoing subsections is fascinating to contemplate. These organizations stimulate the imagination with their strangeness and give promise (however veiled) of self adaptive programming, self organizing systems and, therefore, learning (possibly intelligent) machines. For our purpose, however, the justification for consideration must evolve from a determination that a highly parallel computer can be developed to a point of utility to make it an economical system component or that it can execute needed tasks beyond the power of other organizations. More particularly, the potential usefulness depends on whether the defects apparent in current designs can be overcome.

These problems appear to be extremely formidable; particularly if considered in the context of future tactical system usage. These disadvantages were discussed in connection with the individual designs. The difficulties inherent to this type of organization may be summarized as those of cost and programmability. The cost factor is somewhat augmented by the reluctance on the part of the designers to limit either the capability of the modules, or the number of the modules. To maintain generality, designers are reluctant to limit the theoretical capabilities and thus curtail their study efforts; nor does design of a machine with fewer modules result in an effective compromise, it is felt. The hardware utilization, therefore, to date is extremely low. An attendant difficulty associated with the tendency to insist on a large number of modules, is that the resultant machine is potentially massive and perhaps difficult to incorporate into the system.

The programming problem is perhaps even more severe. The experience built up over the years with serial machines is only partially applicable. The problems of spatial constraints, the importance of relative positioning of data and programs creates formidable problems of allocation. Path interference and restricted intercommunication must also be coped with along with the necessity for careful scheduling and solution of precedence problems. A large measure of success will depend on whether compilers can be written to make communication with this organization tractable in terms of workable programs obtained from reasonable human effort. There is reason for certain optimism in this area since dramatic changes in the language required for communication with machines give promise for greater leverage in the use of programming effort.

The element of time may also be considered a drawback. Although investigators are uniformly enthusiastic, the concensus suggests that practical implementation is some way off. Emphasis has not been on development of a practical version, but on the development of theories of organization.

A number of steps must precede a practical implementation. Analytic and simulation studies of considerable scope must be undertaken to point to possible organizational improvements. This should be coupled with extensive investigation concerning the characteristics of problems whose solutions justify the considerable expense involved. Many limited versions may then be built for experiment and demonstration. Finally, large prototypes will be built, probably costing several millions of dollars each. The designers and builders of such machines are understandably reluctant to take this last step until the underlying theories are highly developed. There is little question about the need for parallelism but great uncertainty about the most efficient means of implementation.

The potential importance of these developments should not be minimized since impressive speed advantages have already been demonstrated. The organization is also particularly efficient for many of the applications of interest to tactical system planners. In general, they may be characterized as the same type of usage for which associative memories were found efficient. In addition, highly parallel organizations are considered promising for problems for which problem solution is an evolving process, and for which program modification is undertaken as a gradual and self-adaptive procedure. The hopes for development of adaptive self programming and self organizing systems are valid but likely to prove elusive.

The question is not whether the highly parallel organization is efficient for the problems of interest, but rather, whether it is as efficient as other alternative means which will be available. In particular, the associative memory with certain extensions may be regarded as providing a large degree of parallelism at considerably less expense. The continuum of capability may be described with the simplest search memories at one end and the Holland machine at the other. Starting with the basic parallel search capability, the ability to change memory cells simultaneously is added (with a parallel write operation); it is then possible to combine elemental logic by microprogramming to provide parallel computation; the next capability is the capacity for the memory cells to communicate or affect each other. (This is roughly at the arbitrary boundary point we have chosen for highly parallel machines). At this point we have the capability for parallel operation by a program upon many data sets (SOLOMON, SPAC). Finally, the operation of independent autonomous programs operating in parallel is proposed. (Holland machine). It seems likely that the practical utilization of parallel processing for advanced tactical systems of 1970 will stop short of the highly parallel machines.

7.7 COMPUTER SYSTEM ORGANIZATION

7.7.1 Introduction

This section identifies important computer design parameters for Advanced Command Data Systems applications, attempts to place bounds on these parameters by extrapolating present trends, and indicates areas where further study is required to permit design decisions. The analyses of Advanced Command Data Systems requirements and the studies of technological trends presented in earlier sections provide a basis for many of the conclusions reached in this section. A consideration of Advanced Command Data Systems general computer requirements is followed by a review of present NTDS compatible computers. In subsequent sections an examination of trends in computer technology leads to an investigation of the many factors to be considered in a detailed design of a basic computer. The basic computer is then considered, as a member of a family of computers, in a time sharing environment, and in a multi-computer configuration.

7.7.2 Computer Considerations for ACDS

General requirements for ACDS computers may be derived from an analysis of data processing functions necessary to support a representative command post, a study of appropriate computer implementations, and a consideration of the hardware constraints imposed by the ACDS environment. The following observations are based on a previous analysis of the command posts for an Attack Carrier Strike Force with particular emphasis on the CTF function.

7.7.2.1 ACDS Data Processing Requirements

The Attack Carrier Strike Force command posts requiring data processing and computational support are:

1) Commander Task Force/Officer in Tactical Command
   a) Anti-Aircraft Warfare Commander
   b) Air Operations Coordinator
   c) Fleet Electronic Warfare Commander
   d) Screen Coordinator.

2) Screen Commander

3) Contact Area Commander

4) Air Operations

5) Task Element

The CTF/OTC command post and its four components are expected to be located on an attack carrier. In addition, each attack carrier will contain an Air Operations Command post. Screen Commanders will most probably be located on a lead destroyer (DLG) while Contact Area Commanders will typically operate from DD's. Task Element command posts apply to all NTDS equipped ships and aircraft.

7.7.2.2 Computer Implementation of ACDS Functions

Data processing tasks common to the ACDS command functions are:

1) Console and group display handling

2) Communications handling

3) Data base updating

4) Data retrieval and formatting for output

5) Computations

6) Executive control functions

Typical computational tasks accomplish tracking, CAP vectoring, mission success modeling, grid lock calculations and threat evaluation and weapons assignment model solutions. The computer implications of these tasks vary from one command post to another. A detailed analysis of the CTF function, however, has provided information pertinent to the implementation of that processing task and from which conclusions may be drawn about the equipment required for other command posts.

Three important computer system design parameters are:

1) Typical data rates

2) Total data base size

3) Program memory requirements

An analysis of message traffic at the command post input interface has indicated that the computer assigned this task must process 600-700 characters/second. Similar estimates of the size of data files comprising the data base indicate that approximately one million 30 bit words of bulk memory will be required for this purpose. A summary of program storage requirements, presented in Table 7-2, indicates that approximately 120,000 words of random access memory will be required for all CTF programs.

The analyses of section four suggest that the data processing workloads for the command posts under consideration are in the following approximate ratios (normalized for the CAC post):

1)	CTF	5
2)	Screen Commander	2
3)	CAC	1
4)	Air Operations	2
5)	Task Element	3

It has also been estimated that the CAC requirement can be met with a computational capability roughly equivalent to the AN/USQ-20A computer.

7.7.2.3 Hardware Constraints

The operating environment for Navy tactical data systems varies from shipborne NTDS use, airborne Airborne Tactical Data System use, to land Marine Tactical Data System use. MTDS has a further requirement to be helicopter transportable. In all three of these environments, the space occupied by computing equipment is an important consideration. For MTDS and ATDS weight is also of importance. Since equipment designed for airborne use would meet the requirements for land and sea use, it might be concluded that any computing equipment should be designed for all three. However, airborne equipment has tended to be more expensive than ruggedized surface equipment and might thus negate the advantage of standardization. Also the form of airborne equipment is sometimes specialized to the location in which it will be used, particularly peripheral equipment such as display consoles. For these reasons, it seems likely that future tactical data systems will be based on two series of modular equipment -- airborne and surface equipment.

Table 7-2. Summary of Program Memory Requirements For The ACDS Attack Carrier Task Force Command Post

Task	Instructions	Working	Tables	Buffers	Total
1. Display	7,900	----	2,400	19,000	29,300
2. Communications	5,650	300	1,050	4,150	11,150
3. Data Base Maintenance	18,000	2,000	1,600	1,500	23,100
4. Data Formatting	1,500	----	----	----	1,500
5. Computations	12,100	4,600	7,550	----	24,250
6. Executive Control	16,000	1,500	4,500	6,000	28,000
Total	61,150	8,400	17,100	30,650	117,300

It is desirable that each of these have standard system interfaces and, where possible, have identical performance characteristics. This is of particular importance for the processor, memory, and standard I/O modules, where certain programming aids are of use in both series of equipment.

Reliability of individual modules must increase as systems become more complex. MTBFs of 1000 hours and more for present Navy computers should be improved in next generation equipment.

7.7.3 Present Computers

The role of Naval forces in meeting the challenges of modern warfare has led to the development of computers especially designed for military environments. These computers function as the central processing elements in systems designed to provide rapid response to both defense and attack situations. They must be capable of operation in severe physical environments, demonstrate high reliability and have relatively low maintenance requirements.

A number of successful Navy computers have been designed and produced by UNIVAC. The Naval computers USQ-17, CP-642A/USQ-20, CP-642B/USQ-20, and the CP-667 are direct descendents of the earlier UNIVAC 1103A and 1105. In addition, two smaller computers have been built with NDTS interfaces: the TRW AN/UYK-1 and UNIVAC's Model 1218. Among these, only the CP-642A/USQ-20 has been officially designated a NTDS computer. It is in current and widespread use by the Navy as the unit computer of NTDS.

The UNIVAC designation for the Q-20A is Model 1206. It is identical to the earlier Q-17 but uses an improved form of logic circuits. The Q-20B (UNIVAC Model 1212) is a later version of the Q-20A with a compatible instruction set and improved execution times. The CP-667 is a newer (1964) computer equivalent to the Q-20B in one mode. It has a longer word length and more powerful instructions in another mode. Also, UNIVAC's Model 1830 computer, to be deliverable in 1965, will be a microminiaturized version of the Q-20B using integrated circuit components.

The hardware and programming characteristics of the Q-20A, Q-20B, CP-667 and AN/UYK-1 are summarized in Tables 7-3 through 7-6. This information provides a basis for comparing basic computer capabilities and for observing evolutionary trends. The AN/UYK-1 will not be compared with the other computers because of its shorter word length and fundamentally different logical structure.

Table 7-3. CP-642A/USQ-20 (V)

Arithmetic mode	parallel, binary, ones complement, full and half word operations
Word length	30 bits
Memory size, speed	main memory: 32K, 8 μsec. cycle plugboard bootstrap: 16 words
Instruction format	single address
index registers	7
indirect addressing	no
instruction types	16 arithmetic 23 logical 7 memory 9 branching 7 I/O
Typical execution times	add 16 μsec. multiply 35-112 μsec.
I/O channels	12 input 12 output
buffering	memory cycle stealing
transfer rate	60kc word rate (max.)
inter-computer	2 special I/O channels
Real time features	internal real time clock
Power requirements	2.5KW
Reliability (est.)	1500 hrs. MTBF
Special features	

Table 7-4. CP-642B/USQ-20 (V)

Arithmetic mode	parallel, binary, ones complement full and half word operations
Word length	30 bits
Memory size, speed	main memory: 32K, 4 μsec. cycle control memory: 64 words 67 μsec. cycle NDRO bootstrap: 64 words, 67 μsec. cycle
Instruction format	single address
index registers	7
indirect addressing	no
instruction types	16 arithmetic 23 logical 7 memory 9 branching 7 I/O
Typical execution times	add: 8 μsec. multiply: 32-48 μsec. square root: 48 μsec.
I/O channels	16 input 16 output
buffering	memory cycle stealing
transfer rate	250kc word rate (max.)
inter-computer	16 channels (max.)
Real time features	internal real time clock
Power requirements	2KW
Reliability (est.)	1500 hrs. MTBF
Special features	special mode of divide provides square root capability compatible with CP-642A

Table 7-5. CP-667

Arithmetic mode	parallel, binary, ones complement half, full, or double length operations
Word length	30/36 bits (selectable at operator's console)
Memory size, speed	main memory: 2 banks 65K, 2 μsec. cycle (1 μsec. eff.) control memory: 256 words, 400 nsec. cycle NDRO bootstrap: 64 words, 400 nsec. cycle
Instruction format	single address
index registers	120
indirect addressing	yes
instruction types	30 arithmetic 24 logical 12 memory 10 branching 8 I/O
Typical execution times	add: 2 μsec. multiply: 18 μsec. floating add: 6 μsec. floating multiply: 18 μsec.
I/O channels	16 input channels 16 output channels
buffering	memory buffered and/or program controlled
transfer rate	250kc word rate/channel 500kc word rate (max.)
inter-computer	16 channels (max.)
Real time features	internal real time clock interrupt features
Power requirements	4.2KW
Reliability (est.)	greater than 1000 hrs. MTBF
Special features	compatible with Q-20A, Q-20B in 30-bit mode 8 sets arithmetic and index registers memory lock out scatter read, gather write hybrid, micro-electronic circuits

Table 7-6. AN/UYK-1

Arithmetic mode	parallel, binary, 2's complement
Word length	15 bits
Memory size, speed	8192 words, 6 μsec. cycle
Instruction format	single or multiple address
index registers	as determined by stored logic
indirect addressing	yes
instruction types	
Typical execution times	add: 12 μsec. multiply: 57μsec.
I/O channels	two 15-bit channels one 30-bit channels
buffering	by processor
transfer rate	30kc word rate
inter-computer	uses basic channels
Real time features	multi-level automatic interrupts
Power requirements	600 watts
Reliability (est.)	1000 hrs. MTBF
Special features	order structure variable by stored logic 82 microcommands

The 1830 computer was identified earlier as an integrated circuit version of the Q-20B and will exhibit essentially the same performance characteristics. It will occupy less than five cubic feet and will weigh approximately 50 pounds or about 1/15 the volume and weight of the equivalent Q-20B. MTBF for the 1830 is estimated at 8,000 to 15,000 hours, or approximately 10 times the reliability experienced on the Q-20A.

Table 7-7 presents a speed/cost comparison for the three computers under discussion. The adjusted cost for the CP-667 has been obtained by estimating the differential cost of its larger memory due both to greater word capacity and its longer word. No actual memory cost data are available; it is assumed that a computer of this word length with a 32K memory would have approximately equal cost in memory and non-memory portions as is true for many commercial computers. The normalized speed/cost figures correspond to a speed/cost improvement of about 15% per year. The CP-667 performance figures do not reflect its greater capabilities in the 36-bit mode.

Table 7-7. Speed/Cost Comparison

	Q-17, Q-20A	Q-20B	CP-667
Memory Cycle	8 usec.	4 usec.	2 usec.
Add Time	16 usec.	8 usec.	4 usec. (no overlap) 2 usec. (overlap)
Multiply Time	35-112 usec.	32-48 usec.	18 usec.
Memory Size	32K x 30 bits	32K x 30 bits	131K x 36 bits
Approx. Cost	$185,000	$245,000	$750,000
Adjusted Cost	$185,000	$245,000	$260,000
Speed/Cost (Normalized)	1	1.5	2.6 (no overlap) 2.5-5 (overlap)
First Delivery	1958	1961	1964

7.7.3 Trends in Computer Systems

Computer system organization is in a state of flux. Factors are at work which will have great impact on the computers of the 1970-80 era. These factors are many, including new applications, higher levels of system complexity, extended supporting software, advanced peripheral equipment, new concepts in computer organization, and revolutionary changes in electronic components.

7.7.3.1 Component Technology

Electronic components will undoubtedly be the most significant single factor influencing computer systems in this era. Changes in computer organization, software developments and system complexity will in many ways merely reflect the "era of microelectronics". The reader is reminded of these implications of the use of microelectronics in computer systems as compared to discrete components:

1) Cost for equivalent circuit capability will be sharply reduced in most cases.
2) Functional circuits (counters, shift registers, etc.) will be stressed rather than basic circuits (individual flip-flops and standard logic)
3) Design tradeoffs will be much different, favoring symmetry and reduced terminal count.
4) Reliability will be greatly improved (although the spares problem will be aggravated).
5) Weight and volume requirements for electronics will be greatly reduced (reduced wire length will permit greater circuit speed).

Other components will be improved during this era. For example, the present trend of more and faster memory per dollar will continue, further aided by microelectronics. It is also likely that memories with new characteristics such as content addressing may find system use in the 1970s. In the area of peripheral equipment, improvements which make peripheral devices more usable to the Navy can be expected, such as a larger selection of rugged equipment and improvements in user characteristics. One emphasis partly due to microelectronics, will be to do more of the peripheral function electronically, and less of it mechanically. In addition to physical improvements, a comparable virtual improvement in peripheral equipment can be expected from better design in computer programs.

7.7.3.2 Computer Organization

The principal features of today's general purpose computers are remarkably similar to the Von Neumann computer of the early 1950s. The change from vacuum tube to transistor technology in retrospect has had a relatively modest effect on computer organization. Integrated circuits, however, are much more likely to affect basic computer structure.

For computers requiring only modest performance, the benefits of microelectronics might principally be applied in the form of reduced cost, size, and weight, rather than a more complex organization. For computer systems which are performance limited, the trend will be to a greater parallelism of structure. One trend which has already become apparent is the use of a much larger array of registers in the computer, both in the processor and in I/O channels. However, it is too early to predict the direction which the industry will take in more general forms of parallel structure, or indeed, if it will take only one direction. Three possible directions parallel computer organization might take are suggested below:

1) Multiple special purpose processors which together constitute a general purpose processor.
2) Multiple identical processors which share certain facilities such as memory access interrupt structure, etc.
3) Arrays of identical processors which communicate principally with their neighbors and with their local memory.

Parallel organization (1) is found in existing and planned computers which are very large, and which derive their computing speed from overlapped processing. Examples of special purpose processors are add/subtract units, multiply units, divide units, and variable length computation units. This type of organization is usually accompanied by substantial memory look-ahead features.

Organization (2) is a variation from conventional multiple computer configurations in that several processors reside in one unit and share certain facilities. This type of structure is now found in I/O, with multiple autonomous channel control units sharing memory and supervisory control.

Organization (3) is typical of computers such as the Holland and SOLOMON machines described earlier. The applicability of this type of computing array to problems which are spatially structured has been demonstrated. However, applicability to generalized command and control is questionable at this time.

7.7.3.3 Communications and Displays

Communication and display subsystems have increasing importance in future Navy command and control systems. In communications, higher transmission rates and more communication sources will combine to produce a much greater information bandwidth. Certain ACDS functions, such as rapid transferral of a command post from one ship to another, require careful examination of communication capacity. In such a massive transfer there must be a much greater emphasis on absolute fidelity of transferred information.

In another area, increased requirements for store-and-forward message communication can also be expected. With greater dependence of non-Navy forces on computer-based communications systems, a potential situation for interchange is developing. Complete automaticity of this function, including substantial amounts of memory, is desirable. The handling of Fieldata and ASCII codes will be greatly increased in this environment. For the 1970's, there may be renewed emphasis on the adoption of a single code for data interchange.

In displays, CRT consoles supporting the individual user seem desirable. This type of display which connects to the computer system in an all-digital manner has been used for a number of years, and it can now be considered a mature peripheral device. Refreshing the information displayed on CRT consoles should be automatic -- computer handling should be confined to changing the data being displayed. This implies display buffering either at each console, or dedicated memory for supporting a number of displays.

The ACDS tactical environment makes certain background information desirable. The detail in such background cannot be properly handled by generation of symbols and line segments, requiring other methods of presenting background information such as projection or video mixing.

Group displays which are capable of rapid updating have not achieved a level of performance which is satisfactory. While extensive use of this type of display could not now be recommended, further developments of basic group display techniques should be encouraged and monitored.

7.7.3.4 Software

Trends in software development of particular significance to ACDS have two principal directions. First, a trend is taking place towards greater simplicity and effectiveness in use of systems. One form this takes is that of languages. Since command and control involves such diverse activities as air operations, threat evaluation, communications handling, and information retrieval, its language requirements are truly complex. One example of a language area which needs development is that related to display of retrieved information. Present display methods employ display generators which are quite inflexible, in turn requiring that the system depend on highly formatted displays and display change data. Display flexibility would simplify and extend the use of displays, and might also make displays more effective in an overload environment.

The second trend in software development is to produce greater computer system efficiency and dependability. This involves concepts of time sharing, multiprocessing and degraded operation which are contained principally in the executive control program. Successful preparation of an executive program requires combined knowledge of the system functions, the system hardware, and the working programs.

7.7.4 <u>Basic Computer Organization</u>

The internal organization or logical structure of a computer is properly based upon:

1) a consideration of the uses which are anticipated for the computer,

2) the characteristics and capabilities of existing equipment, and

3) the current states of the art and discernible trends in system organization and component technology.

An understanding and appreciation of the problem is obviously important. A knowledge of existing equipment is also essential, particularly if compatibilities are to be achieved. Finally, it is necessary to establish cost/value trade-offs in the light of technological advances and changes in user emphasis based on experience.

The subsequent sections, 7.7.5, 7.7.6 and 7.7.7, consider the topics of families of computers, time sharing, and multiple computer systems in greater detail.

7.7.4.1 Control Philosophy

Control philosophy is here defined to include both a basic design approach and the essential features of a particular plan. Several basic approaches have already been identified. Previous discussions of multi-computers dealt with unique configurations of "conventional" stored program computers and did not imply an otherwise unique form of control at the basic computer level. The highly parallel computers were found to be of questionable value in the Navy tactical environment and were observed to have inherent disadvantages of relatively high cost and difficulty of programming.

It may be concluded that the conventional Von Neumann computer, in which a stored program of instruction is retrieved from memory and sequentially executed, embodies the most generally useful control philosophy yet proposed. The set of instructions employed, however, may depend upon the specific method of control. For example, a stored logic approach might lead to a set of simple instructions which in combination make up more conventional instructions. However, this need not be the case, and the trend in stored logic/microprogramming is to use this technique to mechanize conventional instructions. As such, it may be looked upon merely as an alternate way of implementing control which is chosen or rejected on the basis of cost and performance.

Improvements to this basic approach will most likely take the form of sophistications in implementation. Many serious "sophistications" seek to improve processing speeds by introducing some form of parallelism into the control process. A major observation here is the relative independence between the operations of instruction processing, operand access and instruction execution. This creates the opportunity to look ahead in the program and perform these phases of successive instructions in a cycle overlap fashion.

The "look ahead" technique is an effective way of improving performance when complex algorithms have been employed and circuit and memory speeds have already been improved as far as practicable. That is, further improvements must reduce waiting times by operating the memory, control unit, and central processing unit in parallel. Approaches in which several processing units are operated in parallel from either private or shared memory are considered to be forms of multi-computing or, in the extreme case with many simple processors, examples of highly parallel structures.

To satisfy multi-processing requirements, an adequate control philosophy must provide for orderly interrupts according to a program determined priority. The storing and reconstituting of working registers and program status information during interrupt processing has been facilitated in some recent computers by the provision of several sets of registers. The registers may be implemented in flip flop form or as words in a fast control memory.

7.7.4.2 Instruction Format and Repertoire

There is current widespread use of the single address instruction format which usually includes designators for indexing and indirect addressing forms of address modification. These formats imply fixed register assignments within the processing unit and have been generally accepted as leading to the most useful and flexible instruction repertoires.

Although the single address format has remained popular, the number of instructions expressed in that format and hence the number of instruction code bits required has tended to increase in each new generation of computers. The requirement for index registers seems to have been a matter of preference and is relatively stable at 7 or 15 (3 or 4 bit designator) for larger computers. The use of the address field to contain a literal or short operand for immediate execution is a useful variation of the single address format.

Data word length of present Navy computers has been quite satisfactory for tactical data use, with occassional requirements for double precision computation. The use of variable length operands and character addressing is advantageous in digit arithmetic and in message handling as it occurs in NTDS, for example, in display preparation.

The adoption of another distinct data format for representing numbers in exponential or floating point form led first to standard routines for performing arithmetic or floating point data and later to the inclusion of floating point instructions in basic repertoires. Because it accommodates wider ranges in data magnitude and simplifies scaling problems, floating point representation has become a practical requirement for all computers applied to even modest computations.

Several machines of recent design have gained both speed and flexibility from the use of a set of generalized registers within the processing unit. A corresponding instruction format permits reference to pairs of registers whose contents serve as operands for the common arithmetic and logical operations. In effect these registers provide a local, fast access storage which permits operations involving several operands to proceed without memory access delays. An additional format which contains a memory address field and register designators permits data transfer between registers and memory, and allows the generalized registers to be used either as index registers, base address registers, or registers for computational results.

7.7.4.3 Internal Memory

The internal memory of a typical stored program computer performs in the following roles:

1) Program storage
2) Data buffer and storage
3) Processing and control registers
4) Calculational scratch pad
5) Bootstrap program

Within the same computer the memory may have different implementations appropriate to its function. Conventional core memory serves both as program and data storage. The registers and scratch pad take the form of separate, small high speed memories in some very capable computers. The bootstrap may be in an NDRO memory unit or, equivalently, wired as a permanent part of main memory.

Main memory capacity is related to the size of all programs which must be executed cyclicly or upon demand, and to the amount of data which must be held in random access, ready reference form. Studies of one ACDS computer task have indicated that on the order of 100,000 words of random access memory with one million words of bulk memory backup may be required.

Another important aspect of internal memory is its modularity, that is, the ease of adding increments of memory. Modules with capacities as low as 4,000 words permit ease of tailoring memory to application requirements. Modularity also implies bus communication with the memory and the possibility of operating the independent modules in an overlapped fashion. The study of multi-computer systems indicated the potential need to share modules of memory among several computers and their memory buffered I/O devices.

Also identified was the multiprocessing need for memory protection features. Memory protection may be implemented on a word by word basis, by memory module, or by the use of registers to specify protected regions of the memory system.

7.7.4.4 I/O Implementation

In early computers the control I/O devices and programmed data transmission often represented timing complexities and programming difficulties. Present medium and large scale computers overcome these problems by providing data paths between I/O devices and internal memory. The processor prescribes a memory area containing a data block, initiates the data transmission and proceeds to other processing until the completion of transmission is signalled by a program interruption.

A valuable extension of the memory buffered I/O philosophy is the recent concept of a data channel unit with a greater capability for autonomous operation. The data channel is activated by an I/O instruction, receives and decodes I/O commands from memory, and directly controls I/O devices. Thus, it performs complex operational sequences without processor intervention. Further, the ability to reinstruct devices more quickly for successive operations permits the realization of higher data rates.

More sophisticated data channels may be constructed to provide for economical multiplexed operation of many devices with slower data rates. The multiplexed channels are capable of executing I/O command sequences for each device and of controlling information flow between it and prescribed areas of memory.

In the multi-computer systems analysis, a need for inter-computer communication was identified. Early attempts at providing this capability often resulted in an implied master-slave relationship which did not exist in the desired multi-computer configuration. A solution to this problem, which permits interconnection of data channels, is required for ACDS applications.

An alternate approach to the problem of computer to computer communication in a multi-computer system may be to use shared memory to provide common access to the required data. A detailed study of this problem and its systems programming implications is required to determine applicability to ACDS tasks.

7.7.5 Family of Computers

Computer manufacturers and users frequently refer to a particular group of computers as a "family". It is the purpose of the following discussion to define the family concept, to examine the benefits accruing from family relationships, and to consider the relevance of these benefits to the ACDS problem.

7.7.5.1 Family Concept

The family designation is usually reserved for the case where two or more computers of different size or capability are offered by a manufacturer or applied by the same user. Although successive generations of one computer may be so described, more typically the members of the family co-exist and may be chosen on the basis of their suitability for selected tasks. It is the exhibition of similar characteristics by the different sized members which identifies the family and permits efficiencies in their application.

7.7.5.1.1 Program Compatability

A valuable family characteristic is that of program compatibility; that is, the ability to run programs written for one member on another member without modification. Program compatibility is important because it:

1) Simplifies the training of programmers,

2) Reduces program development costs, and

3) Permits system growth without the necessity to replace correctly operating programs.

If both upwards and downwards compatibility are not achieved in the design of the family, the third factor requires only upwards compatibility while the second requires both. Although one could conceivably write all programs for the smallest member of the family and take advantage of upwards compatibility, such an approach would not realize the full capability of the larger members. Effective program compatibility can be achieved with varying degrees of efficiency and desirability at the following levels:

1) Symbolic language (compiler level)

2) Machine language (instruction level)

3) By microprogrammed sequences

Symbolic Language

In the first level, compatibility results from the specification of a common symbolic language for use in all program preparation, regardless of which computer may actually run the program. The individual computers of the family need not be identical at the machine language level but each must be equipped with a compiler program to generate its own machine language version of the symbolically stated program. Although compilers traditionally produce less efficient object programs than manual coding, the advantages of a common, application oriented, symbolic language may dictate its use in any case. Programming compatibility at a symbolic level would provide the three benefits noted earlier, requiring that only systems programmers and maintenance personnel be familiar with the various machine language instruction repertoires.

Machine Language

Compatibility at the machine language or instruction level implies a strong logical similarity between members of the family. Logical functions may, however, be implemented with different electronic "building blocks" and instruction execution times may vary from member to member. If upwards and downwards compatibility is required, there must be a one-to-one correspondence between instruction set features. Great care must be exercised during machine design to ensure that differences in implementation do not introduce subtle (but disastrous) differences in instruction effect. Where upwards compatibility accomplishes the desired purposes, the instruction repertoires of the smaller members may be subsets of the progressively larger members.

Complete instruction set compatibility provides the three benefits stated above, permits the development of universal system software, and simplifies training of maintenance personnel. Adherence to a doctrine of machine language compatibility, however, does have the adverse effect of inhibiting the introduction of new techniques and technologies during the subsequent design of replacement machines or additional members. That is, second generation machines may be less efficient because of the necessity to accommodate a previously established word length and instruction repertoire (with all of the implied logic structure) while at the same time introducing newer, more desirable features which cannot take advantage of that hardware required for compatibility.

Microprogrammed Sequences

The stored logic or microprogrammed technique of computer design, previously discussed in section 7.5, offers a third approach to achieving program compatibility in a family of computers. It was noted in this discussion that one of the principal characteristics of microprogrammed computers is the basic nature of the micro-operations and, in some cases, the freedom to assign processing functions to a number of general registers. These attributes facilitate "interpretive mode" operation, permitting microprogrammed computers to simulate (or emulate) a given instruction repertoire with considerably more efficiency than would be the case with a "conventional" computer. The benefits of program compatibility at an instruction set level can thus be achieved by equipping a microprogrammed computer (or a family of microprogrammed computers) with appropriate sequences of micro-orders to perform equivalent operations for each instruction in the set.

7.7.5.1.2 Input/Output Compatibility

Like programming compatibility, the ability to control and communicate with a common group of input/output devices is an important characteristic of a computer family. It is not essential that the individual computers have identical I/O instructions but the circuit properites and logic of the interface must agree if all members are to use the common devices.

I/O compatibility is a necessary condition to the use of several different members in a multi-computer configuration with variable or shared peripheral responsibilities and with inter-computer communication requirements. A considerable emphasis on I/O compatibility stems from the relatively high cost of designing and installing special purpose adapters and equipment interface. Also, it is generally true that system growth involves the substitution of a faster, more capable processor along with the addition of more I/O interconnections. That is, there is usually the requirement for the new processor to continue to function with the existing I/O channels. One can envision an evolutionary type of growth, however, in which a next generation processor is equipped with a number of improved, higher speed I/O channels in addition to the normal complement of "standard" channels.

In modern computer systems with memory buffered I/O features, I/O compatibility among family members also implies similarities in methods of memory communication. In particular, if two or more processors are to share common external memory modules with assigned I/O channels, standardized memory communication is required.

An additional memory related aspect of I/O compatibility is the recognition that in order to make efficient use of memory for data storage, memory word length should be equal to the I/O channel width or a simple multiple thereof. This integer relationship eliminates program steps and avoids compromising I/O transfer rates because of the need for programmed formatting.

7.7.5.1.3 Parts Interchangeability

The ability to reassign system functions within a multi-computer network may be thought of as a form of parts interchangeability. The previous discussions of I/O and memory compatibilities disclosed additional examples of parts interchangeability at the unit and module levels. Similarity or compatibility also typically exists in a computer family at a replacement part or subassembly level. Although the concept of a computer family does not demand this type of compatibility, it has not been uncommon to make use of a family of plug-in circuits and logic elements whenever possible in the design of a computer series.

7.7.5.2 Relevance to ACDS

The present NTDS is built around the concept of a unit computer (AN/USQ-20) and the use of multiple computers when a requirement exceeds the capabilities of a single unit computer. The range of applications evident in ACDS suggests that computers both smaller and larger than the Q-20B will be needed to fulfill present recognized needs. Moreover, a small family of computers with varying size, capability and environmental characteristics will apparently be required to accommodate problem growth and meet long range needs.

In considering such a family, the characteristics discussed in the previous section are all relevant to ACDS planning. To simplify programmer training, minimize program development costs, and facilitate system growth, some form of program compatibility should be achieved. While it is fairly certain that a common symbolic language should be adopted, the necessity of instruction level compatibility is not so obvious. Further, the selection of a hardware or a micro-programmed approach to implementing a common instruction set would have to be made on the basis of a more detailed, design oriented study.

The practical requirement for evolutionary change in any fleet-wide installation of equipment places relatively more importance on I/O compatibility. The expectation of multi-computer configurations further underlines this emphasis. Also, the anticipated pattern of system growth, in which a new more powerful processor must work with previously installed I/O equipments, requires careful attention to this family characteristic.

The use of common plug-in circuits and logic elements is important in the ACDS environment because it permits a reduction in the number of spares and specialized test equipment which must be carried for a given complement of equipment and eases associated logistics problems. Additionally, it simplifies the training of maintenance personnel and the actual performance of the maintenance function.

7.7.6 Computer Time Sharing

Multiprogramming, or time sharing, has been previously discussed in terms of its combined programming and equipment considerations. This section attempts to relate computer time sharing to ACDS requirements, and to identify trends in time sharing mechanization.

7.7.6.1 Time Sharing Concept

An area of computer study which has had increasing emphasis in recent years is the "simultaneous" use of one or more computers for several different tasks. At a gross level of details, time sharing is not new. For many years computer programs have been designed to use a processor for one task while I/O is engaged in other tasks (true simultaneity for buffered I/O). Also, real-time systems have been designed to utilize a program which is capable of commutating rapidly through many sub-programs. The newer aspects of time sharing are the accomplishment of apparent simultaneity of computer programs by much tighter interleaving of individual programs, and with appropriate protection features to prevent one program from interfering with another.

The need for this intimate time sharing of computers has been heightened by the increase in use of on-line manual inputs such as the consoles used in command and control systems. At MIT, SDC and RAND, time-sharing system studies are based on manual job entries from consoles. A number of operational systems are successfully in use with large numbers of data entry devices such as airline reservation and stock market quotation systems. However, there is a symmetry of purpose that makes these much less complex systems than their counterparts in command and control.

Apart from man-machine considerations, there are similar situations which stand to benefit from time sharing. Wherever appreciable excess computing capacity exists, other programs can, in principle, use the remaining capacity. It is this improvement in efficiency which is the principal benefit to be derived from time sharing. This benefit is similar to the benefit obtained in communications by the use of multiplexing techniques to better utilize the available bandwidth.

The more significant obstacles which stand in the way of realization of effective time sharing are:

1) Those factors which contribute to wasted time in making transitions from one program to another, thereby eroding efficiency gains

2) Those factors which permit program interaction, thereby violating the integrity of individual programs.

As an example of the desired characteristics of time sharing, consider three computer programs each capable of execution on three identical computers. Suppose, as an alternative, a computer four times faster than each of the three computers is available at a moderately higher cost. In this example the objective of time sharing is to interleave the three programs in the single larger computer to reduce system cost. Of course, this must be done in a manner which keeps the input and output for each individual program valid; also, the control program which performs the time sharing must not be larger than one of the job programs.

It is obvious in this example that the two previously mentioned obstacles -- wasted time in program switchover and loss of program integrity -- require careful consideration. To be optimized, both require special hardware and software features. These features can be subdivided into two categories, processor time sharing features and memory time sharing features. The processor time sharing features determine to a great extent the program switchover efficiency, which in turn is directly related to the frequency with which switchovers are made. Memory time sharing features are predominant in determining the level of program protection as well as program and data relocation.

7.7.6.2 ACDS Requirements

Computers in all command posts, at every level of command, have a requirement for program concurrency. In the limit this may merely consist of a self-test program interleaved with an operating program. Typically, however, it is desirable to perform a number of concurrent operational tasks. Some of these tasks which have a counterpart in virtually all ACDS command posts are as follows:

1) Position-keeping (e.g., grid lock)
2) Communications data handling
3) Manual inputs (commands and inquiries)
4) Data base maintenance
5) Display updating
6) Resource allocation
7) Threat evaluation
8) Self-test

Under stress conditions the sum of these tasks will exceed the capacity of one computer. On the other hand, a task such as position-keeping requires only a fraction of a computer at any time in spite of its frequent updating requirements. Tasks such as threat evaluation have greatly varying processing requirements and are further complicated by a fast response requirement, particularly with regard to low-flying aircraft. Communications, data base maintenance and display updating have much in common since activity in one area will frequently imply activity in other areas.

In specific terms, ACDS requires time sharing in some form. It may be cyclic processing, that is, a fixed sequencing through task assignments, or it may be demand processing. Demand processing is best suited to tasks which are generated in a sporadic manner such as inquiries from consoles or received messages over communication paths. Time sharing should be introduced in ACDS in a manner which involves little system risk. Some guidelines are as follows:

1) The frequency of program switchover should be modest, such that the dwell time of each program is in the tens of milliseconds. This prevents undue complications due to unanticipated inefficiencies in switchover.

2) Memory time sharing should be minimized in the sense that program and data relocation is avoided where possible. This simplifies both the hardware features and the executive program.

3) Input/output considerations should be such that complex timing rules need not be observed. Also, an integral part of program integrity is the maintenance of "need to have" restrictions for all users under all stress conditions.

7.7.6.3 Time Sharing Features

Time sharing of general purpose computers relies on both hardware and software for its implementation. Early systems had a deficiency of hardware features, placing an added burden on computer programming. This has contributed to a somewhat distorted picture of the complexity of time sharing. On the other hand, it would be unrealistic to assume that added hardware features will eliminate the need for time sharing software.

The hardware features which facilitate computer time sharing can be essentially divided into those related to the processor and those related to the memory. Processor time sharing is aided by features related to scheduling and to the mechanics of program switchover. Scheduling features are those which aid determination that one program should supersede another, either on the basis of completion of a phase of the program in execution, by interruption for a new task, or by routine cycling of programs. Features which facilitate the actual switchover to a different program are those which can efficiently store the state of a program. (In this case it is assumed that the program being supplanted was not necessarily at a convenient pausing point.) A summary of these processor time sharing features is given below.

1) Program Scheduling Features:

   a) real time clock
   b) elapsed time interrupts
   c) I/O completion interrupts
   d) user interrupts
   e) communication interrupts
   f) priority masking of interrupts
   g) interrupt enabling

2) Program Switchover Features:

   a) return jump operation (stores program counter)
   b) word-organized storable program indicators (overflow, divide check, etc.)
   c) rapid storage of all register contents in memory
   d) rapid loading of program counter, indicators and registers from memory
   e) alternatively, multiple sets of program counters, indicators and registers

Several of the features mentioned are basic to real time computer systems even if time sharing is not employed. The last item referred to -- multiple sets of program counters, indicators and registers -- approaches the characteristics of a multiple computer configuration. However, high speed scratchpad memory can be used in this manner as in the CP-667 and thus lead to extremely efficient program switchover.

Hardware features for memory time sharing relate principally to memory protection and memory allocation. Memory protection may consist of features which restrict the region of memory which an individual program may access. Features such as program boundary registers do this by defining a block of contiguous words in memory and restricting program access to this block. A higher level of protection is provided by inhibiting writing into particular word positions. For information which does not change, a type of memory which is unalterable by any program can be used. These features are summarized below.

3) Memory Protection Features:

a) program boundary registers
b) verification of stored identifying program numbers
c) write inhibit flag bit
d) write inhibit by memory zone
e) program unalterable memory

Memory allocation features are of importance if the time-shared programs and data do not have a fixed memory assignment. The problem which then demands solution is that addresses appearing in instructions must be adjusted to reflect the actual location of programs and data in memory. This is referred to as program and data relocation, and can be accomplished by making address adjustments when the program is loaded, or at the time it is executed. If done at load time, it can be accomplished entirely by software -- at execution time, hardware is required. This typically takes the form of a base register which has its contents added to all addresses. The problem of memory allocation is greatly complicated if an individual program is assigned to several different non-contiguous blocks in memory. In this case, memory protection is similarly complicated. The general problem of this type is referred to as _dynamic memory allocation_. Much work remains to be done in this area. Associative memories of high speed have been suggested for dynamic memory allocation.

4) Memory Allocation Features:

a) address modification by program loader (software).

b) address modification by program loader using address relocation flag (hardware aid)

c) base register

d) multiple base registers

e) associative access to base registers

Programming requirements for time sharing have been implicit in much of the preceding discussion. Characteristics of time sharing must be based on system analysis, at least to the extent that worst case requirements are examined. The most significant part of the executive control program for time sharing is that of scheduling. Here the decisions of cyclic versus demand processing are implemented, together with considerations of priorities and peak requirements. Most of the hardware features just cited require program setup: base registers, program boundary registers, interrupt masks, interrupt enable, etc. It should be emphasized that while ACDS requires time sharing, it is desirable to do this in its simpler forms. In particular, complex memory allocation should be avoided. On the other hand, memory protection features are extremely desirable and should be incorporated in ACDS in its earliest stages.

7.7.7 Multi-computer Systems

The topic of multi-computer systems and multi-processing has been previously discussed. This section summarizes features which are desirable in the implementation of multi-computer systems as related to ACDS requirements.

7.7.7.1 Typical Configurations

Several configurations could be adjusted to meet the needs of ACDS. The principal requirements are modularity, efficiency of use, and adequate protection to the system in the event of module failure. Implicit in modularity is reasonable ability for modules to detect errors in transmitted data and to have rapid system substitution in the event of failure.

Despite the potential adequacy of a number of configurations, certain unifying trends are apparent in multiple computer organizations. These are cited with respect to processor/memory relationships, processor hierarchies, and peripheral equipment.

Processor/memory relationship has tended towards the time multiplexed bus rather than the selection of one bank of memory by one processor for a span of time. Access by two or more computers to one bank of memory is more efficient than queuing for its use. It should be recalled that memory within a bank can still be protected piecewise by program boundary registers or similar techniques.

The use of processors of more than one size (processor hierarchies) in the same system may have merit, particularly if a program compatible family of computers is used. This is particularly relevant if a processor is sized for a given major task -- differences in processing requirements leading to different sizes. In a family of computers, the notion of identical multi-computer features is significant.

Peripheral equipment has proved to be a complex problem in multi-computer systems. On the one hand, equipment such as a printer has essentially one purpose for lengthy spans of time. On the other hand, the printer should not have a fixed allocation to one computer, since its failure should not imply printer failure. A generalization of the buffered I/O channel appears to be the solution to this problem. In effect, there need be no direct (non-switchable) communication between processor and peripheral equipment. Rather, the I/O connection is to memory and the processor connection is to memory, resulting in ready substitution of either module.

7.7.7.2 Multi-processing Features

As for time-sharing, multi-processing is achieved by a combination of hardware and software. In fact, all of the time sharing features discussed in the preceding section are germane to multi-processing. Some additional hardware features are itemized as they apply to all modules, to processor modules, to memory modules, and to I/O control.

1) Multi-processing Features, All Modules;
    a) parity bit (or equivalent) attached to all transmitted data
    b) error checking on all received data
    c) self-contained power regulation or isolation
    d) circuits minimizing propagation of faults to busses
    e) capability for rapid removal of module from system

2) Multi-processing Feature, Processor Modules;
    a) time-sharing features (see preceding section)
    b) intercomputer communication capability
    c) interruptable idle state
    d) system reset

3) Multi-processing Features, Memory Modules;
   a) memory protection features in conjunction with processor (see preceding section)
   b) memory allocation features in conjunction with processor (see preceding section)
   c) time multiplexed word and address busses
   d) overlapped cycles (if more than one memory bank per module)

4) Multi-processing Features, I/O Control;
   a) standard interface
   b) capability for multiplexed channels
   c) symbolic addressing for all I/O modules

Accompanying these hardware features must be a carefully designed executive program. Significant advances have been made in this kind of control programming in recent years -- the question is no longer one of feasibility, but one of quality and effectiveness. As discussed in the next section, one aspect of multi-computer control which requires much care and planning is that of recovery from failure.

7.7.7.3 Degraded Operation

A most important consideration for tactical systems is the ability to retain partial effectiveness in the face of one or more equipment failures. If it is assumed that redundant modules will be available on standby, the problem of failure recovery is principally involved in failure detection and equipment substitution. In an operational sense this produces a transient from which full recovery can be made unless essential data is lost. If, on the other hand, redundant modules will not be available when a failure occurs, system analysis of the alternatives must be performed to determine if a change in procedure might have to be employed.

A fundamental factor in automatic compensation for failure (and one occasionally neglected) is adequate failure detection. It is extremely important that most module failures be automatically detectable in a timely fashion, either by self-indication or by detection by another module of an incorrect response or hangup.

After detection of failure, it must usually be assumed that the integrity of certain preceding processing was compromised, and that rollback or data rejection should take place. This is a complex consideration and must be carefully examined at the system level and reflected into automatic recovery procedures. For ACDS, specific objectives should be defined with regard to which module failures will be compensated, the rollback mechanism, and any alterations to operating procedures which might result.

7.7.7.4 ACDS Implications

Multi-processing has been an essential ingredient in NTDS and has established a framework for evolutionary improvement. While it can be anticipated that these improvements will add complexity to control programs, these can be contained by clear specification of objectives. From an equipment point of view, it would appear that present organizational features will continue to endure for some time. However, it would also seem that there will appear some time prior to 1980 rather significant organizational features which must be considered revolutionary. These changes will be traceable mainly to new design tradeoffs associated with microelectronics. It is too early to ascertain in detail what these changes will be, but a parallelism of structure -- and hence a new form of multiprocessing -- is strongly indicated. The resulting new levels of performance and reliability would in turn lead to vastly improved tactical command systems.

Section 8
PROGRAMMING

8.1 INTRODUCTION

One problem in the implementation and operation of real-time computer based systems, as typified by the NTDS and ACDS, is the timely and economical acquisition of computer programs which increase rather than limit the system capabilities. Programming is an art, and the success of programming projects is a function of the talent, intelligence, ingenuity and interest of programmers. All too often, system designers depend on programming talent to correct hardware design errors. Under these circumstances, program development is difficult to plan and schedules are not met. The end product is significantly different from that envisioned, and the wasted effort contributes an excessive amount of extra costs.

This section is a description and evaluation of techniques and concepts which may be employed to make programming projects more tractable. Parts of the section are devoted to a definition of terms, a description of the state-of-the-art, a prognostication of the state-of-the-art, and to specific recommendations.

8.2 COMPUTER PROGRAMS AND COMPUTER PROGRAMMERS

Computer programs are the means by which a computer connected to peripheral devices can efficiently perform required functions. The primary task of the computer programmer is the generation of computer programs from well defined specifications. In the past, the programmer has performed other tasks needed to reduce a problem, however vaguely stated, to a sequence of operations performed by a system incorporating a computer. The programmer's function has been the analysis of the problem and the synthesis of computer hardware and software to produce an automatic procedure.

This concept was workable when the hardware system was limited to a computer with an input mechanism, an output mechanism and some auxiliary storage. It is not workable when the programmer is presented with a large real-time problem and a large

number of arbitrarily selected hardware components. In such cases, the generation of the computer programs themselves may be only one element of importance as far as the development of the overall system is concerned. The following items are at least as important as the programs themselves:

1) The design of operational procedures and displays; the analysis of the human factors;
2) The scheduling and management of the computer programming effort;
3) The analysis of the problem and the specification of the manner in which the system is to be synthesized;
4) The design and implementation of the programs which permit the generation of computer programs in an efficient and economical way.

There is no question that experienced programmers are needed to participate in these tasks including participation in the specification of system hardware. But, in the implementation of ACDS it is suggested that a comprehensive and integrated effort be made to specify procedures for hardware and software simultaneously.

8.3 PROGRAMMING TOOLS

8.3.1 Summary

The following is a discussion of the programming tools which are important in the context of ANTACCS. The tools are defined, described, and evaluated. Development trends are presented. Particular attention has been given to the definition of terms. Because the field of computer programming is dynamic, there is a wide divergence of opinion, even among specialists, concerning the meaning of many of the programming terms in common use. The definitions are not intended to be authoritative and are included only to achieve consistency and clarity.

8.3.2 General Discussion

The programming tools described are in common use today. They are most appropriate for the programming of independent computers constructed on the Von Neuman model. At the present stage of programming technology these programming tools may be used to produce modular sub-programs. The sub-programs must later be integrated into the system according to specifications developed during a system analysis and synthesis period.

Programming tools serve several purposes. They assist the programmer in the construction of a program, in debugging or eliminating mistakes, and eliminating redundant effort. More important, when the programming effort is large, they assist in coordinating and standardizing individual efforts. They may be generally classified as languages, input/output control systems, checkout tools, monitors and executives.

In the programming context, a language is a defined set of symbols and the rules governing the manner and sequence in which the symbols may be combined. Programming languages are not the same as natural languages. Confusion is possible because those who use and are most familiar with them have found the nomenclature and tools of analysis of natural languages to be very helpful and appropriate. Hence there is a tendency to overemphasize similarities and correspondences. Rather, these "languages" are members of a class of formal systems of expression similar to the equations of mathematics. Although the power of programming languages is limited, their utility must not be underestimated. By providing a powerful notation, they allow a programmer to concentrate on a method of problem solution, rather than on the problems associated with organization of machine instructions.

In the period immediately following the introduction of general purpose computers, it was necessary for programs to be written in machine language. In a machine language, machine instructions are written in a form the machine can use directly. The overwhelming majority of large machines which have been built use binary codes internally. For these machines the direct external representation of internal codes must be written in the form of octal or hexadecimal numbers. Because programmers are inherently more capable of manipulating alphabetic symbols and decimal numbers, the process of programming in machine language is slow, laborious and error prone.

A result has been the development and the wide use of programming languages to permit the use of symbols and decimal numbers together with language processors. These, in turn, translate the symbolic statements into a machine language.

The use of programming languages is generally less efficient than when programmers produce machine language programs directly. Machine language programs generally occupy less memory and require less computer time for their execution. As a rule, the more general purpose the programming language, the more inefficient it becomes both from the standpoint of the computer time (required by the language processor) and the efficiency of the machine language produced. On the other hand, programming languages are extremely valuable because of the ease with which programs can be constructed, checked out, and maintained. An approach becoming more and more popular is to retain ease of programming and to increase the efficiency of the language processor through the development of programming languages designed especially for the application.

The individual machine instructions for input and output are usually not very powerful, but are very numerous and are complicated to use. They are designed to accommodate, at minimum hardware cost, a large variety of requirements which are not always encountered at each computer installation. This problem has been solved by the development of input/output control programs. These programs are designed to coordinate input and output operations and to alleviate program coding and checkout problems without sacrificing processing efficiency.

The many attempts made to accelerate the program checkout process have resulted in the development of standard techniques and general purpose programs developed especially to both assist the programmer and to discipline his program checkout efforts. The development of coordinated sets of such programs has progressed to the point that use of the term "checkout languages" is entirely appropriate.

Most computer installations pass through a phase during which the availability of computer time is no problem. Later on, however, when scheduling problems develop, it becomes apparent that the usual program checkout run requires a minimum of ten to fifteen minutes of which less than a minute is devoted to computer operation. The earliest solution to this disparity was the use of the technique of batch processing under control of a monitor routine. In batch processing the inputs to the computer consist of programs and data on cards or magnetic tape. The card decks are stacked in the order in which the runs are to be made and, normally, the information is transferred to magnetic tape. Under control of the monitor routine the computer then executes the entire sequence of programs and places the outputs on magnetic tape. This is later translated to cards, printed or plotted. This technique not only improves the efficiency with which the computer is used, but improves the efficiency of the programmer by forcing him to plan in advance for the occurrence of all manner of errors in programming and machine malfunctions.

When a digital computer is connected to devices not entirely under its direct control, an interface or boundary between the devices exists. In this section on programming, interface will be taken to mean not only the definition of the data and control information that passes over the boundary, but also the rules and conventions required to integrate the devices or connect them into a useful system. Usually each device connected into a computer, either because of the way it is used or because of the way the hardware is constructed, has a unique interface. Executives provide the means by which the interface problem is solved for all programs.

Executive programs relieve the programmer of the intricate problems associated with the interface between the computer and the devices to which it is connected. They provide overall system control. They minimize the changes which must be made in the computer programs where the system is used in a different way or hardware components are added.

Other functions commonly performed by executive programs are the scheduling of tasks initiated by the external devices and the allocation of hardware facilities for their execution.

8.3.3 Languages and Language Processors

Language processors are computer programs which translate programming languages into machine language. Following a short general discussion of programming languages are evaluations of several existing languages and their processors together with an evaluation of CS-1, the language processor and language currently used in programming for NTDS.

Programming languages are classified four ways: machine-oriented, procedural, problem oriented, and non-procedural. The classifications are not exclusive. For instance, a language may be both procedural and problem oriented.

Machine-oriented languages are designed for a particular machine. All assembly languages are in this classification because of the basically one-to-one correspondence between machine instructions and statements. The early computer languages that differed from machine language (e.g. FORTRAN 1 and its predecessors, PACT 1, A-2, etc.) were christened "Problem-Oriented Languages" (or POL's) by the original UNCOL study group. This was an unfortunate choice of terms, since they could not describe a <u>problem</u> to be solved, but only the <u>procedures</u> to solve the problem. Later, this distinction was made, and now purists distinguish between "Procedure-Oriented Languages" and "Problem-Oriented Languages." An early example of the latter was the 9 PAC Report Generator language. Here the <u>problem</u> is to produce a report. The language provides its processor with a pictorial description of the report and the names of the data to be contained herein, but <u>says nothing about the procedure</u> to be used in producing the report.

Hereafter we call such languages non-procedural. Note that current non-procedural languages are limited to the class of problem for which the computational procedures are predetermined. The concept, nevertheless, is important for ACDS, since there is a high probability that the only effective programming language for use with systems which are organized in novel ways will be non-procedural. However, they are not treated in detail, because none currently exist which is sufficiently related to a future ACDS problem.

External computer languages, or source languages, have been previously discussed in a general way. It is important to maintain the distinction between the language and its processor. The processor is the routine which translates it into basic internal machine language, or object language. This distinction must be kept in mind although the Language/Processor pair must frequently be discussed at the same time. Before proceeding to a discussion of existing Language/Processor pairs, language processors in general will be briefly considered.

One type of language processor is called "interpreter." An interpreter accepts expressions of a language, usually one statement at a time. Interpreters translate each statement into internal machine language and immediately execute it, however, before translating the next statement. Thus they are simple to implement. However, the translation is not preserved, and must be done again each time the program is run on the computer.

Another type of language processors is called "compilers." (A subset, which process symbolic assembly languages, is usually called "assemblers.") These accept an entire program and translate it completely into internal machine language. Sometimes intermediate languages are used with two or more translation steps. The translation is preserved for execution at any subsequent time. Compilers are more difficult to implement. Currently they are usually considered more efficient than interpreters, since the translator process is not repeated needlessly.

In discussing language processors, another distinction must be kept in mind. There may be many different processors for each language, for different machines, or even for the same machine. Some processors may emphasize speed of translation. Others may emphasize conservation of memory space during translation. Others may aim at translations into object code which will economize storage space or which will run as fast as possible. Rarely can all these objectives be achieved by a single processor.

8.3.3.1 Syntax and Semantics

In the following discussions two terms appear, syntax and semantics, which have a special meaning when used in connection with programming languages.

"Syntax" refers to the rules for the formation of permissable expressions without regard for the meaning of the expressions. All expressions have a structure. That is, they are composed of elements (individual symbols). The syntax dictates the way in which these elements may be combined. Many programming languages permit the construction of arithmetic formulae. A part of the syntax of the programming language are the rules for the construction of permissable arithmetic formulae.

"Semantics" refers to the correspondence of expressions and their intended meanings or interpretation. Another definition is the meaning jointly assigned to the constructions of a language because of context or the relationship of statements. As an example, in the case of the languages which permit the construction of arithmetic formulae, the semantics will confirm the fact that the variables are numbers. However, if the language also permits the construction of Boolean statements, the semantics will determine the operation which takes place when a statement is encountered containing variables connected by a Boolean operator.

Semantics is concerned with meaning and significance. A program designed to solve a problem is syntactically correct if written in accordance with the rules of a language; it has semantic content to the extent that it embodies a method of forming the required solutions.

8.3.3.2 Symbolic Assembly Languages

Symbolic Assembly Languages (SAL's) – sometimes called Machine-Oriented Languages (MOL's) – are the most universally used computer languages. The capability they afford is frequently indispensible. They will probably provide the most convenient way of programming some problems.

In general, a SAL provides a means by which the instructions for a particular machine may be written symbolically. That is, locations in memory may be assigned alphanumeric names, and the machine instructions may be referred to by mnemonic symbols. Every useful machine instruction is assigned its mnemonic symbol. All numbers may be written in decimal notation. The advantage of writing programs in an assembly language is that the programmer can maintain almost complete control over every detail of the operation. He can use almost every capability of the machine without sacrificing either the ability to work with symbols or the relocatability of programs. A program is relocatable when moved from one portion of the internal memory to another and the reference to memory cells are automatically adjusted so that the routine in its new location will be executed properly.

The disadvantage of using an assembly language is that every detail of the operation must be specified explicitly or implicitly. As a result, the process of programming in an assembly language is time consuming, and the programs produced are difficult to read or interpret because the statements can seldom be grouped into sequences which are very meaningful to human beings.

A modern technique is the "Meta-Assembler" approach. In this case the characteristics of the object machine (the machine on which the object language is to run) are considered as input data to the processor. Thus the processor can produce, from the same source language, object code to run on any one of several machines.

8.3.3.3 FORTRAN

A version or dialect of FORTRAN was one of the first languages for which a widely used operational compiler was completed. It is important for several reasons. It is well suited to a broad range of scientific applications. FORTRAN processors have been written for a large number of different types of computers. The concepts FORTRAN embodies have instigated much research in computer languages and many attempts to develop computer languages for non-scientific applications. The problems encountered in writing FORTRAN processors have prompted the development of most of today's compiler technology.

FORTRAN programs are composed of statements: arithmetic, control, input/output, call, and specification. Arithmetic statements are used to perform numerical calculations. Control statements permit branching of control following decisions. Input/output statements specify format statements which, in turn, contain the items necessary for defining input and output data formats and the mode of translation to representations used internally. Call statements are used to link the main routine with subroutines. Specification statements are used to allocate data and temporary storage areas.

Some FORTRAN dialects or variations of FORTRAN permit arithmetic operations to be performed on integer, real, and double precision numbers. The analogous operations for complex numbers and logical constants may also be permitted. The capabilities of a particular FORTRAN processor are a function both of the machine for which it was written and the capability of those who wrote it. Processors for smaller computers have fewer facilities. The newer processors contain fewer program steps, work faster, and produce more efficient machine language programs. The extra facilities which are provided in some cases, or the particular facilities omitted in the processors for the smaller machines, depend on the taste and background of the writers.

A distinguishing characteristic of FORTRAN dialects is the manner in which real numbers are represented and manipulated. The scientific notation is assumed throughout. That is, every quantity (except for zero) is represented by two numbers: a mantissa, which is less than one but which has a non zero high order digit, and an exponent, which defines the number of integral digits or number of zeros preceding the first fractional digit. Input and output are decimal, but, if the machine is binary, the internal representations are binary. When machines have memories which are addressed in words, single precision quantities are stored in one word and double precision quantities are stored in two words.

The basic language is most powerful for applications requiring matrix manipulations. It is a very useful tool for most scientific applications. For the normal type of commercial problem it is rather inefficient, and caution should be exercised in using it on applications which are largely logical. But it would not be difficult to enrich a FORTRAN processor with functions and subroutines to the point that it would be useful for applications involving logic and be, in some cases, quite satisfactory for commercial data processing.

The principal drawback to FORTRAN is its vague and less than optimum syntax, and its tendancy to contain a degree of dependence. Because of this, a FORTRAN program which operates properly after being translated by a given FORTRAN processor may not work properly after being translated by another FORTRAN processor which was written from the same specifications. Because the syntax is less than optimum, the processors are more complicated than necessary.

8.3.3.4 COBOL

The COBOL language was developed by representatives of computer users, Government installations, and computer manufacturers at the instigation of the Department of Defense. The intent of the committee was to provide a problem-oriented programming language for computer users with business applications. The language was to be machine independent, amenable to further development, and easy to learn. The programs produced were to be self-documenting. Ideally, the problems of converting programs for use on different types or later models of computers would be eliminated and programs could be produced by personnel relatively inexperienced in the workings or language of computers.

The committee did a very good job. But, because no technical breakthroughs were made, the announced goals were achieved at the expense of other considerations which have proved to be equally as important. A business-oriented language was achieved, but the language is not attractive for use in any other area which has come to light so far. The language is machine independent, but the programs it produces are machine dependent to a significant extent. Programs which run efficiently on one computer do not run efficiently on another. Some manufacturers have special added features to the language which encourage the programmer to write programs which are more efficient for their equipment, but cannot be run on competitive equipment. The language is officially amenable to development, but it is evident now that, although it is amenable to change, the basic concepts preclude the integration of new techniques in a reasonable manner. Ease of learning and self-documentation were achieved at the expense of enforced verbosity.

It can be predicted that the business data processing world will move rapidly away from symbolic assembly languages. Many will adopt COBOL, but many others may delay the move until they see if NPL will become the de facto standard.

8.3.3.5 ALGOL

The word ALGOL is a contraction of international algorithmic language. As its name might imply, it is the product of an international committee of computer language experts. Quoting from the report which defines and describes the language, "This is a language suitable for expressing a large class of numerical processes in a form sufficiently concise for direct automatic translation into the language of programmed automatic computers". It has the following important characteristics:

1) It is a procedural language. The non-procedural aspects are minimal.

2) The syntax is both concise and precise. The form in which it has been presented will probably become the canonical form for any new languages that are developed. It has been given the special name "Backus (or Backus-Naur) Normal Form" after these members of the development committee who invented it.

3) In the United States, acceptance and implementation have been slow.

4) It is recognized as the publication language for programs, algorithms, and techniques of general interest. But no input/output equipment can handle its character set.

5) To date, it has no official input/output facilities.

Several factors have precluded rapid acceptance and widespread implementation. The language requires the use of many uncommon symbols with special meanings which are useful only in the writing of ALGOL programs. Manufacturers have been reluctant to provide the hardware required to read and print the symbols directly. The language is sufficiently unique that programmers have been reluctant to invest the effort necessary to master it. Some of the features are sufficiently difficult to implement and would be used so rarely that only subsets of the language have been incorporated into processors. In short, it has been a language for the expert.

On the other hand, ALGOL, as an influence on future languages will assume increasing importance with the passage of time. Its generality and capabilities are as great or greater than the class of computers for which it is intended. Peripheral equipment which will accommodate larger character sets, at such a price that it can be dedicated to the programmer, will be made available. Above all, the facility it affords for the precise, concise, and unambiguous expression of procedures in a machine independent manner will make it the prototype of languages for the computer expert of the future.

8.3.3.6 NELIAC

NELIAC (from Naval Electronics Laboratory International ALGOL Compilers) is, as one might expect, a dialect of ALGOL. It was developed concurrently with ALGOL58, a predecessor of the current publication ALGOL.

The importance of the language is not the facility it provides, but in the way it has been implemented and used.

Of greatest importance is the fact that the processor is written in its own language. There are two major effects. It is possible to expand the compiler to accommodate any special features which may be desirable: it is self compiling. And, a compiler for one computer may be written and compiled on a second computer: it is machine independent.

Next, a decompiler for NELIAC has been written. A decompiler is capable of producing symbolic programs from machine language programs. Through the use of a decompiler on one machine and a compiler on another, direct translation of programs may be achieved.

8.3.3.7 JOVIAL

JOVIAL is corporate procedure oriented language produced by the System Development Corporation. It incorporates many of the features of FORTRAN and ALGOL together with special features gained from the System Development Corporation's experience with the development of the SAGE air defense system.

Although the language is patterned after ALGOL, use is made of self-explanatory English words and ordinary algebraic notations in a manner similar to FORTRAN or COBOL. Unlike most FORTRAN and COBOL processors, JOVIAL processors permit the incorporation of assembly language instructions.

One of the most important features of the language is the COMPOOL concept which permits a programmer to refer to and manipulate COMPOOL data. The COMPOOL is a library for a large programming system which supplies the JOVIAL compiler with data description parameters. Thus, it is unnecessary for the individual programmer to be concerned about or, in most cases, even be aware of data formats. By changing the data descriptions in the COMPOOL, it is also possible to change the manner in which the data is manipulated in all the programs making up the system.

Machine independence is achieved by introducing an intermediate language. The JOVIAL processors incorporate a generator which translates the JOVIAL programs to the intermediate language and a translator which converts the intermediate language to the machine language of a particular computer.

Of all the languages discussed, JOVIAL seems to be a good model for a future ACDS. The most serious objection which can be raised is that the language is quite old and that no special provisions have been incorporated for programs which must be executed in a time shared environment.

8.3.3.8 NPL

NPL is an acronym for New Programming Language, the programming language currently being implemented for IBM's System/360. Although it is not radically new, it must be included because of the importance it is expected to have in the near future.

It is anticipated that the language will be extensively used. IBM intends to market System/360 to all its customers as a single product line and NPL will be the primary and best-supported programming language offered. Other manufacturers intend to market "compatible equipment" with compatible software. Probably, some of them already are implementing NPL for their existing equipment. Thus, NPL may become the main programming language in general use and the single language into which the computing industry places a substantial developmental effort.

In general, the language is based on FORTRAN, is influenced by ALGOL, incorporates the data handling capabilities of COBOL, and recognizes problems associated with multi-programming and multi-processing. NPL can be thought of as FORTRAN with added capabilities. Some of the more important capabilities that have been added include:

8.3.3.8.1 File Handling

Files may be described, created, and edited. The structure (item size, identification and format) may be defined so that the item attributes need not be considered in later data translation and numeric computation operations. Items may be stored and retrieved with commands such as GET, PUT, MOVE and SEARCH.

8.3.3.8.2 Time Sharing

Programs written for NPL are assumed to be executed under the control of an executive program. The syntax of the language is such that execution of a program in a time shared environment is possible. Conversely, to date, the executive has not been defined and the language contains no instructions to the executive.

8.3.3.8.3 Data Base Definition

Quantities used in computation (variables) may be defined to be global, local or own. Global quantities have the same value and are located in the same place for all subprograms which use them. Local quantities may be referenced only by the subprograms for which they are defined and during the execution of that subprogram. Own quantities are used to communicate information to subroutines of the subprograms. Global quantities, then, may be thought of as items of a COMPOOL for the program in question.

NPL is an advanced language for the general user because it allows him to write programs to be operated in a time shared environment without being required to be aware of memory or storage allocation procedures. For ACDS, however, the language itself is not superior because the syntax is not radically improved, explicit provision is not made for the assimilation of problem-oriented statements, and a solution to a central problem for ACDS, that of facilities allocation and scheduling, has not been defined. However, there are significant advantages of NPL for ACDS:

1) Character manipulation.
2) Data directed I/O.
3) Real time programming features such as provision of automatic handling of reentrant procedures.
4) Operations on bit-strings.

8.3.3.9 CS-1

CS-1 is the language currently being used by the Navy in the preparation of programs for NTDS. The following comments relate it to the languages which have been described.

CS-1 provides the facilities normally found in a machine oriented language. It is of necessity machine dependent. The compiler is written for the CP-642 computer.

The language provides some, but not all the facilities found in FORTRAN. Provision is made for control and arithmetic statements but not for input/output statements. The level of sophistication of control and arithmetic statements is not as high as in FORTRAN. Since it does not include input and output statements, the file definition facilities of COBOL are not present. The syntax is very simple and uncomplicated when compared to ALGOL. No facilities are provided for self-compiling.

Only those functions necessary to the NTDS application have been provided. Facilities that were not provided are floating point arithmetic, arithmetic function subroutine, extended precision arithmetic, text formatting routines, etc. These latter generally are found in compilers which must deal with a wide range of applications.

Special facilities are memory allocation, interface definition, and debugging statements.

The tailoring of capability language has not been accompanied by an increase in compilation speed. At last report, some of the statements had not been implemented.

8.3.4 Input/Output Control Systems

The computer programs required to read, write and coordinate the operation of input/output equipment are some of the most difficult to design, program, and checkout. Hence, it is standard practice to concentrate this type of program into a single package presenting a standard interface to other computer programs. Such package, concerned with the interface between the computer and any other devices completely under its control, will be referred to as an input/output control system.

Input/output control systems represent a careful balance between the constraints imposed by the hardware and the constraints imposed by the programs which the computer executes. They are both machine and application dependent, but are designed to permit the programmer to exercise a maximum of control with a minimum of effort.

A general input/output control system will read data, write data, check for erroneous data, check for malfunctions of the peripheral devices, check for computer malfunction, maintain a record of the status of peripheral equipment and execute standard malfunction and error procedures. Erroneous data checks will include checks for error codes (parity, redundancy, error correction, etc.) and format (too much data, too little data, numeric, alphabetic, binary, etc.). Peripheral equipment status classifications are those for connected, disconnected, operating, malfunctioning, etc. Standard malfunction and error procedures may be automatic (reread or rewrite following the detection of erroneous data) or involve the operator (restart after peripheral equipment malfunction).

In existing systems, the execution of input/output control system programs represents an excessive and unavoidable overhead. These programs require both computer memory for storage and computer time for execution. By applying the modularity and economic special purpose components afforded by advances in hardware technology together with an adequate overall system design, it will be possible to so standardize the input and output procedures that such programs can almost be eliminated. The amount of programming and computer time under actual operation devoted to input/output control provides a criteria for effective system design.

8.3.4.1 Monitors

Monitors were first conceived for use with off-line computer systems; that is, with computer systems for which inputs and outputs were entirely under the control of the central processor. In most computer installations it is not unusual for the computer time required for the execution of a single task or run to average less than ten minutes. The majority of these runs are used for program checkout and require less than a minute. The time between tasks, if each task is set up after the last is removed from the equipment, will average at least five minutes. Monitors provide the means by which a group or batch of tasks can be executed following a single setup. For this reason, the computers in these installations are called batch processors, and the operation under a monitor system is called batch processing.

Monitor systems usually incorporate an input/output control system, an assembler, one or more compilers, a program checkout package, and a program library from which frequently executed programs can be retrieved and run automatically. In the past, inputs were stacked and converted to magnetic tape, and outputs were stacked on magnetic tape for later conversion to cards or for printing.

Even with a monitor system the central processor of a large system will be idle much of the time awaiting the completion of input and output operations. A recent innovation involves a smaller satellite computer connected to the larger computer. The monitor resides in the smaller computer which assembles inputs for and distributes outputs from the larger computer. The work is processed continuously — so that it is no longer a batch processing system but has evolved into a "continuous-flow" system. The earliest such system in widespread use is the IBM 7044/7094 Direct Coupled System.

The value of a monitor system lies not only in the computer operating economies which are possible, but also in the reduction of manhours required to produce checked out programs. To use a monitor system, the programmer is required to think through and document each step the monitor is to perform and to examine the consequences of each mistake. In reality, this is a small task and inhibits an irresistible tendency in programmers to "try it and see if it works".

8.3.4.2 Program Checkout and System Test Tools

Good debugging and program testing tools, when they are properly used, reduce immensely the computer and programmer time required to check out programs. The availability of adequate tools for system test is imperative for the implementation of large systems. Nevertheless, because their development is uninteresting, and requires patience and an incredible attention to detail, debugging and testing tools are usually developed subsequent to the development of all other programs.

The functions of computer program checkout tools are to provide programmers with the ability to examine the results of the execution of his program in the minutest detail. The consensus is that the basic program checkout tool provides the ability to obtain, at specified points during the program's execution, the status of specified portions of main memory.

It is advantageous if the presentation of the memory status takes a form which is as close to the source language of the program as possible. Numbers should be presented in decimal, and instructions presented symbolically. To achieve this capability, there must be a close relationship between the language processor and the checkout tools. In one instance, this close relationship was achieved by embedding the language processor within the checkout package.

Program checkout tools are useful only for the check out of individual programs. In order to test time-shared or multiprogrammed systems, special system test tools are required. Even though two or more programs may operate properly when executed independently, the timing relationships and the facility allocations may be such that neither operates in parallel.

A good set of system test tools has three components. The first is capable of generating test data with specified characteristics. The second is capable of driving the system with these inputs and recording the system outputs. The third component is capable of analyzing the outputs for anomalies. It must be possible for all three components to be executed concurrently.

8.3.4.3 Executives

A program which is the master controller of all other programs which run on the computer is referred to as an executive. An executive performs the functions of an input/output control system and, depending on the application, for any of the following functions:

1) Scheduling - The scheduling and/or sequencing of the tasks the system is to perform on the basis of manual inputs, external interrupts, predetermined internal sequences, or priorities.
2) Facilities Allocation - The allocation of the system hardware to the tasks which are to be performed.
3) Real-Time Control - The coordination of system activities and real-time data handling requirements. Real-time data has the distinguishing characteristic that the computer inputs and outputs are determined by other system components. Real-time control involves the acceptance, analysis, control, coordination, and response to real-time data.
4) Data Buffering - The control, temporary storage, queuing, and movement of data through the system.
5) Diagnostics - The automatic detection of errors and malfunctions and the automatic execution of malfunction procedures.
6) Restart - The automatic storage and updating of the files which must be saved in order to restart the system after the malfunction of the system together with the programs required to restart the computer following its repair.
7) System Readiness - The automatic, periodic initiation of those procedures which are required to determine the operability of various system components.
8) System Performance - The automatic gathering, recording, and analysis of parameters needed to evaluate system performance.

For the NTDS systems, the amount of executive programming has been minimized. Sophisticated schemes for real-time control have been avoided, a minimum of queuing of real-time inputs is permitted; the diagnostics are limited to the detection of malfunctions; restart is almost identical to start; system readiness is determined by system use; and system performance parameters are not gathered automatically. The entire program for a particular system is in memory at all times. Input and output are accomplished in much the same way as they would be by an input/output control system. The computers are so constructed that the information in core memory is retained after system malfunction. The diagnostics are completely independent programs. Although the concepts which have been implemented by the NTDS programs are simple and effective, the system is inflexible and the equipment is not used as efficiently as it might be.

In minimizing the executive functions for NTDS, the problems which have proved difficult in the implementation of other more sophisticated systems have been avoided. However, if the power of future computers is to be exploited for ACDS it will be mandatory to make efficient use of modern techniques, which are currently becoming commonplace, such as interrupt-oriented systems, multiprogramming, and multiprocessing. Thus, ACDS will require an extensive executive which incorporates all the functions which have been named. By performing an adequate system analysis, it is now possible to implement a sophisticated executive without undue difficulty. Dynamic memory allocation implies provisions for scheduling, facilities allocation and data buffering. Stochastic inputs and demands for output require the capabilities of real time asynchronous control. Due to the large number of special operations and many procedures which must be performed following malfunction of a system in which memory is dynamically allocated, restart after system malfunction must be judiciously automated. Diagnostics, system readiness, and system performance should be included to maximize system operating time and guarantee system effectiveness. Therefore, these capabilities will be incorporated.

These functions pertain only to the operation of the system itself. They are independent of the functions of the programs which relate directly to an ACDS application. Further, their detailed specifications would be a product not of an analysis of the ACDS problem, but of an analysis of the operation of the system. Hence, it is logical to group them into a single program package called an executive control program.

8.4 COMPUTER PROGRAM PRODUCTION

8.4.1 Introduction

Navy personnel are intimately familiar with the large majority of problems which will be encountered in the production of the computer programs for an ACDS through their experience in the implementation of the current Naval Tactical Data Systems. Much of this experience is directly reflected in the procedures and techniques currently being used at the Fleet Computer Programming Centers such as at FCPCPAC which was visited in the course of the study. This section begins, with a short summary of these techniques and procedures, followed by a discussion of some areas requiring particular emphasis and is concluded with a description of a program production process for the future which will provide more effective control of the process.

8.4.2 Computer Program Production at FCPCPAC.

Although fairly standard equipment modules are employed, each NTDS is tailored to fit the ship on which it is installed. The requirements for computer programs are received from BuShips by the staff responsible for their development in the form of specifications for weapon systems, weapon system configuration specifications, or descriptions of new functions to be performed. Because standard computer hardware modules are used and because the computer program model is fixed, requirements may exceed system capability. As a result, alternate sets of requirements are negotiated.

Operational specifications defining those requirements which can be accommodated are produced by FCPCPAC and approved by BuShips. These become the basis for functional specifications. Functional specifications describe the computer program modules which satisfy the requirements. They serve four functions. They are used for program design, coding, and debugging, for the development of test procedures, for the development of operator manuals, and for the development of the final description of the system.

Following the debugging of the individual computer program modules, they are integrated at the FCPCPAC into a single computer program which must perform according to the test procedures. The tests are repeated following shipboard installation.

The FCPCPAC also performs computer program maintenance and error correction. Program modifications are based on information collected during the actual operation of the system.

Following the debugging of the individual computer program modules, they are integrated at the FCPCPAC into a single computer program which must perform according to the test procedures. The tests are repeated following shipboard installation.

The FCPCPAC also performs computer program maintenance and error correction. Program modifications are based on information collected during the actual operation of the system.

8.4.2.1 Program Production Management

One of the great strengths of the Navy's NTDS computer program production facilities is its management systems. Program development flows from general requirements to detailed specifications, computer programs, and program documentation in an orderly manner. Tasks are defined, estimates are made of manpower requirements, and progress is monitored through the use of PERT techniques.

This emphasis and reliance on management techniques is unusual. In the implementation of some other existing real-time systems, emphasis has been placed on sophisticated programming techniques, often at the expense of good management. Frequently, equally good results have been achieved by each. One may speculate on the degree to which higher efficiencies may be achieved by introducing more sophisticated programming teachniques into the FCPCPAC environment.

8.4.2.2 Documentation

At the FCPCPAC, the need for adequate documentation is well understood. Documents containing the right kind of information are generated at the proper time by the right people. Adequate provision is made for modification. For ACDS, the process would be enhanced if one change were made. It is recommended that increased emphasis be placed on the detailed documentation of coding.

Documentation is expensive and it retains its value for limited periods. Specifications by their very nature are of little use subsequent to the production of the item specified. Because it is never possible to be certain that a program has been completely checked out, even the final documentation is constantly in need of revision.

In recognition of this situation, many attempts have been made to achieve self-documentation. In general, none of these has been wholly successful. An attempt was made to make COBOL programs self-documenting by requiring the programmer to use words of the English language in a restricted way. The result was more annoying than helpful. Various other attempts have been made to invent conventions which would make assembly language or FORTRAN programs self-documenting, but none have gained wide acceptance. Another approach is the decompiler of NELIAC. Under this concept a program is used to translate machine language to the programming language. In a somewhat similar vein, IBM developed a routine for the IBM 709 computer which drew flow charts from machine language. In both these cases however, the semantic values are lost and the results are usually unintelligible to anyone except the author.

In view of the foregoing, it is improbable, that for ACDS, suitable substitutes for the existing kinds of documentation will have to be found.

As was stated previously, the detailed program documentation for NTDS will need to be augmented to accommodate ACDS requirements. The NTDS requirement for detailed program documentation is limited because few people need to understand it. The size of the programs is limited by the size of core memory, which is small. Programs consist of smaller modules which can be understood by a single person. Program maintenance is performed by the author of the program or his successor. For ACDS, a precise means must be developed for communicating the detailed contents of programs because many people will need to understand them. The size and the complexity of the system will probably be such that the system designer, the programmer, the persons involved in system test, and the people who perform system maintenance will not be the same. Each will need to understand the system in order to perform his function properly.

8.4.3 Software Organization

The speed with which programs can be produced and the ease with which programming systems may be tested, maintained, and modified is a function of the software. That is, a function of the manner in which program modules are defined and implemented.

The need for a modular approach for large programs has given rise to the concepts of macro-instructions, closed subroutines and program segments. Macro-instructions are usually a short sequence of machine instructions used with high frequency. Closed subroutines are longer sequences of instructions which are included in the program only once and are linked to the main sequence whenever needed. If the program is so long that it will not fit conveniently into the available main or fast memory of the computer, it must be broken into two or more independent parts. Sophisticated language processors are capable of inserting standard sequences of instructions into programs as subroutines or macro-instructions in the most efficient manner. However, techniques for automatically segmenting programs are in their infancy.

The first NTDS was constructed on a trial and error approach. With a working system available, it has been possible to construct subsequent systems on the same model. The computer programs for a given system are logically divided into program segments or "modules" which are associated with specific functions. This technique permits a new system to be formed from an old one by discarding unwanted modules and adding new ones. Within modules, memory is allocated by methods which are basically manual.

It will not be possible to continue using this simple approach in a time shared system or one employing extensive auxiliary memory devices. For these systems, no general way, both efficient and simple, has been found to relate system functions and program modules. Further, unless every effort is made to minimize the duplication of instruction sequences, and the useless repetition of their execution, the performance of the system may be seriously degraded.

In this respect, advantage should be taken of some of the knowledge gained in such systems as the SABER real time airlines' reservation systems and NASA's real-time control center for Gemini and Apollo (RTCC).

The SABER systems consist of two computers which share access to disk files and drum auxiliary memories and are connected by means of telephone lines to air-line terminals throughout the world. The systems are used by airlines' personnel to keep track of reservations, passengers and planes. Agents are supposed to receive replies from the computer three seconds after entering or requesting information. After a cursory analysis it was decided that the flow of data to and from the auxiliary storage system would place an upper bound on system performance. After installation, it was discovered that the computers could not keep up with the data. It was necessary to double the size of core memory and reorganize the programs to accommodate the workload.

When the RTCC was implemented, a sophisticated executive was designed and written. Subsequently, the system was analyzed using a general purpose system simulator. The analysis showed that the executive which was designed specifically to eliminate bottlenecks in the flow of data had introduced other bottlenecks. As in the case of SABER, a reorganization of the program was needed so that the system workload could be accommodated. In both cases, the lesson learned was that superficial analysis is dangerous economy. Detail study must precede freezing of specifications.

8.4.4 Hardware

This section discusses the relationship of computer hardware characteristics to computer program production from a programmer's standpoint.

In every case, the computer programming techniques previously discussed are designed to allow the programmer to concentrate on user problems rather than on hardware problems. In view of the problems associated with computer program production, testing, and maintenance, it would be foolish to incorporate into the system hardware which complicates the programming problem unless a compensating programming technique is provided at the same time. Alternative hardware organizations which simplify the programming problem or enhance system capabilities without complicating the programming problem must necessarily be evaluated on the basis of cost and efficiency.

Based on the foregoing criteria, the following are evaluations of the software implications for some hardware implementations and techniques.

8.4.4.1 Multiprogramming

Multiprogrammed systems must be accompanied by a sophisticated executive that permits the programmer to concentrate on the application rather than program scheduling problems, input/output problems, storage problems, restart problems and overall system performance. They must also be accompanied by the hardware necessary to protect the programs and their data from destruction by other programs which have been executed in an erroneous manner.

8.4.4.2 Multi-processors

Systems which incorporate multi-computers must be accompanied by executives which permit the programmer to concentrate on the application rather than scheduling problems, storage problems, and coordination and timing problems.

8.4.4.3 Associative Memories

The memories must be accompanied by an input/output package. The package must be sufficiently sophisticated that the programmer need not be concerned with exceptional conditions such as errors or malfunctions, with data formatting and retrieval problems, or with timing problems.

8.4.4.4 Micro-programming

Micro-programming must be accompanied by an interpretive program which permits the programmer to write and check out programs to be written and checked out in a machine-independent language. The programmer should not be required to be aware of the inner working of the micro-programming capability.

8.4.4.5 Highly Parallel Computers

These computers must be accompanied by a scheme which permits the programmer to be unaware that a problem is executed on a computer organized in this novel way, but does not preclude him from exploiting the knowledge whenever this is desirable.

8.4.4.6 On-Line Displays

Must be accompanied by an input/output package which permits the programmer to remain unconcerned about data formats, errors, malfunctions, and timing problems.

8.4.4.7 New Input/Output and Storage Equipment

Must be accompanied by input/output packages which permit the programmer to remain unconcerned about data formats, errors, malfunctions, and performance and timing problems.

8.4.5 A Model for Computer Program Production

The following is a suggested model process for the production of computer programs. Careful implementation will solve the majority of known problems associated with the production of computer programs for systems typified by ACDS. Only available techniques are specified, but the structure is such that future developments will serve to further enhance its attractiveness.

Implicit in the structure are provisions for the orderly growth of the system. Attention is given to:

1) Specification of requirements.
2) System analysis.
3) Software development.
4) System installation.
5) System operation.
6) Program maintenance.
7) System reliability.
8) Hardware modification and expansion.
9) Optimization of system performance.
10) Expansion of system capability.

Last, but not least, automation of the above factors is considered.

The model may not be ideal in at least three respects. Implementation will require the liberal application of both specialized knowledge and talents. Considerable manpower must be expended in coordinating individual activities and in preparing intermediate and final documentation. Lastly, the process may need to be repeated each time extensive modifications are made in the overall organization of the system hardware.

The system, which does not exist, is described in the present tense.

8.4.5.1 Assumptions

Prior to the implementation of the computer program production, it is assumed that adequate programming tools have been provided, an analysis of the system has been performed, the system has been synthesized and the results of the synthesis are well documented. Further, as the system develops, it must be possible to repeat the analysis of the system whenever necessary.

8.4.5.1.1. Programming Tools

The programming tools used fall into three categories: monitor, language, and checkout. An on-line programming system is provided or a monitor is used to efficiently schedule and maximize the time available on the computer for program checkout, and a language and language processor are used to reduce programming time. Checkout tools are used to minimize the time required for the checkout of individual programs.

The monitor has no capabilities beyond those which would normally be expected. The on-line programming system is simply a computerized mode of programming. The system consists of a computer system and a number of on-line consoles for the programmers. The consoles can be operated simultaneously. Programs are written using a typewriter. As each statement is written, it is executed, thus permitting coding and checkout to take place simultaneously.

The language is somewhat different from those in use today in the sense that a careful distinction is made between the processor and the language. In general, the language is tailored to the command and control application and can be extended and altered. For example, it incorporates a well-developed capability, similar to the COMPOOL concept of JOVIAL, which simplifies the programmers reference to standard items of data, data formats, and processes. The processor on the other hand, is written i n a machine-independent language. It is self-compiling and incorporates the features of an algorithmic language. Programmers are able to express problems in the most convenient way and the investment in programs is no longer a factor in the selection of newer computers.

The checkout tools are also somewhat better than those in use today. A checkout language is employed. The checkout statements are used by the compiler to appropriately modify machine language programs. When executed, the modified programs produce data for a post processor. The post processor is capable of decompilation, of providing the information required for analyzing the status of the programs at any point, and of recording the flow of data through a series of programs. When the program has been checkout out, a working version of the machine language programs is produced by recompiling without the check-out statements.

8.4.5.1.2 System Synthesis

The following is a general definition of the requirements to be satisfied by a system analysis. These requirements can be detailed to any level, hence the entire process can and should be implemented using a computer.

The initial analysis of the system requirements and the subsequent synthesis of a system are not part of the computer program production process because of the heavy emphasis which must be placed on the organization and selection of equipment. Completion of the initial synthesis is considered to be equivalent to the preparation of system functional specifications of the system. The functional specifications define the requirements for computer program production and contain the information necessary for the coordination of all further developmental effort.

During the analysis, the system workload is defined, data flow and storage are planned, the hardware facilities and their characteristics are specified, and many system parameters are determined.

The definition of the workload is essentially a compilation of the manual operations and their quantitative effect on the system. Estimates must be made of their frequency, and their interrelationships must be defined. Some operations will be cyclic, some will be initiated frequently, but on a random basis, and some will be part of sequences which are required to accomplish a larger task which is not entirely automated.

The operations result in the flow of data throughout the system. This flow is carefully planned, otherwise the system can neither accommodate peak loads nor be used in a reasonably efficient manner. In planning this flow it is necessary to define the data processing requirements, the requirements for temporary storage, permanent storage requirements, and the stepwise sequence of data handling. If priorities exist, these also are specified.

Neither the operations nor the data flow are meaningful unless they are consistent with available hardware. As with data flow, the hardware resources are planned to preclude the saturation of the system and to insure a reasonably high use of hardware components. To allocate facilities is to specify the correspondence of hardware devices with fixed requirements for storage and processing and to specify the sequence of shared facilities use.

Initially, this type of information is used to measure the quantity of hardware needed, its effectiveness, the impact of alternative modes of operation, and the impact of alternate hardware and software organizations. Subsequently, the analysis is repeated to predict overload conditions, and to measure excess system capacity. In addition, it is used to evaluate the effect of random inputs, to prove that system jamming cannot occur, and to measure the efficiency of the use of hardware components. Because it is a tedious task which must be repeated frequently, a computer program is provided.

The synthesis is well documented. The definition of the operations forms the basis for detailed operation specifications and the preparation of operator's manuals. Definition of the data flow forms the basis for the preparation of detailed specifications for data formats. By-products of the synthesis are computer program specifications for the executive programs and functional specifications for the operational programs.

8.4.5.2 The Production Process

All computer programs proceed through the same basic sequences: requirements, functional specifications, program specifications, coding, and checkout. The requirements are a description of the process to be programmed and the constraints to be observed. There is a one-to-one relationship between functional specifications required and job program modules. They contain a description of what the module is to accomplish, the input, the output, and, if necessary, a description of the algorithms or required mathematical formulas. Program specifications describe in detail the manner in which the functional specifications are to be implemented. Programs are coded from the program specifications. Following coding, the programs are checked out using the process described in connection with testing.

At the time the process is initiated, a management system is established, and a test system is designed. Coding of the executive routines is started and functional specifications for the operational programs are prepared.

The data processing requirements available as a result of the system synthesis are used to develop manpower schedules, assignments, and costs. Management techniques such as PERT/COST are used to monitor progress.

The functional specifications for the executive program are developed as a part of the system analysis. Hence, work is started immediately on program specifications, coding, and checkout. Thus, the executive is written and completely tested before the testing of operational programs is initiated.

8.4.5.3 The Executive

The executive has capabilities which are developed in a minimal way. In general, these are capabilities associated with the automation of routine system maintenance, operation, and management tasks. For example, equipment diagnostic routines are called into execution automatically when a failure occurs. Not only is a remedial procedure carried out, but the point of failure is precisely located for ease of correction. As another example, periodic confidence tests are conducted on pieces of equipment which are controlled by the computer. Finally, the executive is capable of logging deviations from expected performance and of providing information for future modifications.

8.4.5.4 Testing

Program and system testing is a task of the same magnitude as system design and computer program production. Hence, it is given equal attention.

Program test begins with off-line checkout. In off-line checkout the program being tested does not share the computer with other programs and is executed without interruption. Off-line checkout is accomplished in two phases; unit and string test. Unit test is the testing of individual sub-programs. String test is the testing of sequences of sub-programs.

To complete system test, it is necessary to resort to simulation. As far as is known, there is no other practical way to test a time shared system consisting of a limited number of hardware components. Simulation is used, first, to verify overall performance as predicted by the system analysis and, second, to verify the proper operation of programs when they are executed in parallel.

Performance is verified first by passing synthetic data through synthetic operational programs. Synthetic data is data which is identified, occupies specified amounts of storage and uses input/output equipment for specified lengths of time. Synthetic programs are programs which are identified, occupy specified quantities of storage, need specified amounts of computer time for their execution, use other synthetic programs as subroutines and branch to other synthetic programs. The parameters of the synthetic data and programs are obtained from statistics of the system analysis. This approach also verifies the proper operation of the executive program and exercises the system hardware.

Subsequently, the system is tested in the presence of simulated data and real programs. Simulated data differs from real data only in that its contents and time of entry are predetermined.

The testing process is so designed that the system operates properly with mixtures of synthetic, simulated, and real data. Thus, preliminary tests of the system are accomplishes in a full-load environment.

Implementation of a process test of this type requires the development of two types of test programs. One type of test program is needed for the rapid generation of synthetic data and programs or simulated data. The other type of program is needed to drive the system in a way that the timing relationships of the inputs and outputs may be measured.

Because exercise of the system in the manner described requires test equipment some of the planning for system test must be a part of system design. The complement of special equipment needed for system test is identical to that required for system maintenance and to insure system readiness.

8.5 SPECIAL TOPICS

8.5.1 Problem-Oriented Languages

A problem-oriented language has been defined as a programming language which is designed for a particular application. This is not a universally accepted definition. In the opinion of many, the terms problem-oriented and non-procedural are synonomous. It is agreed that the degree to which a language is problem-oriented can be measured by evaluating the degree to which it is non-procedural, but the definition given is preferred because for most applications, including that of ACDS, it is probable that no completely non-procedural language can be found.

It is clear that none of the programming languages previously discussed are truly problem-oriented as far as ACDS is concerned. They are procedure-oriented. They can only be used by trained programmers. None of them have either the semantics or the syntax required to express explicitly and take advantage of a significant portion of the concepts implied by tactical command and control, multi-computers, multi-programming, and time sharing.

In the past, the implementation of a new language for each special purpose was impractical, except for the minor modification of an existing language. However, with the language processor techniques now available, this is no longer true. Hence, the question for ACDS is not whether a problem-oriented language is desirable or feasible, but to what extent one can be implemented and how far it will be possible to extend the capabilities it affords from the trained programmer to the uninitiated user.

Since it is not possible to specify a problem-oriented language for ACDS at this time, nor even its fundamental characteristics, a discussion of some languages is appropriate. These suggest approaches which might be taken alone or in combination.

The most primitive approach to a problem-oriented language is the generator type program. Early programs of this type were report generators and sort generators. These programs used the problem description to connect subroutines into the proper order to form a program. This concept has been expended into very powerful business data processing systems, for example, BEST, developed by National Cash Register Company.

Another approach which shows the most promise, is the program package which solves a specific problem in a general way. These include packages for solving linear programming problems and special purpose languages such as:

1) STRESS, a tool for performing stress analysis;
2) APT, for programming machine tools;
3) SKETCHPAD, for automatic drafting;

and other packages for information processing and list processing.

A third approach is a special purpose procedural language such as COBOL. In view of the large amount of effort which has gone into the development of procedural languages, this approach appears the least attractive. To be effective the extent of the vocabulary and the breadth of processing will have to be tailored to the skill levels of the user.

8.5.2 On-Line Programming

The concept of on-line programming is very important to ACDS. If it is implemented in concert with a computerized documentation process, much of the programming effort can be self-documenting. An on-line programming system can also be used as a laboratory tool in the development of a problem-oriented language.

At the present time, there exist several on-line programming systems. These are, however, largely experimental, laboratory training, or research tools. Most of these operate in a time-shared environment. Unfortunately, none of these are directed toward the solution of the production of computer programs for an ACDS type system.

JOSS, developed by the RAND Corporation, represents an attempt to use a computer as a very powerful hand calculator. MAC, developed by Massachusetts Institute of Technology, represents an attempt to provide each user, in a time-shared environment, with the equivalent of a private computer, complete with all the facilities of a large computer laboratory. The System Development Corporation has developed a system equivalent to MAC.

Though basically an information retrieval system with limited on-line capability, ADAM, developed by MITRE, incorporates a syntax-driven compiler which permits each user to construct his own language.

8.5.3 Implicitly Programmed Systems

In his keynote address to the 1963 Fall Joint Computer Conference, Major General C. H. Terhune, Jr., Commanding Office of the Electronic Systems Division, Air Force Systems Command, introduced to the information processing community a new concept which he called "implicitly programmed systems". A discussion of the concept is appropriate both because of the interest, excitement, and confusion which his remarks have generated and because the techniques which are used in achieving an "implicitly programmed system" could easily be applied to ACDS.

General Terhune introduced the concept with these remarks:

"We concluded that nothing short of absolute re-thinking of computer design, programming and operation could give the Air Force the kind of computer capability it needs. We call on the computer industry to support such complete reorientation by focusing on a new class of data processors which will emphasize user controlled evolution and flexibility. In supplying this emphasis it is reasonable to assume that new systems would have certain crucial characteristics.

"Primarily, they should be programmed to provide the user complete freedom to structure his system to fit his needs. He should be able to accept new types of messages simply by describing their formats to the computer. It should be possible for the user to create new files, and have them automatically updated, without programming newer routines or tables. Displays and reports should not have to be preprogrammed by professional designers. If the need for special displays were generated spontaneously, the user should be able to describe them to the computer in much the same way that he now queries the data base. It shouldn't be necessary to rewrite special purpose housekeeping functions for each new job. But the functions should be part of a larger class of data manipulation programs, thus giving the user complete freedom to work in the problem setting unconstrained by a fixed set of subroutines."

General Terhune added:

"Technically, the freedom is not unreasonable."

Specialists in information processing agree that implementation of this concept or something like it has been their goal for many years. They agree that the goal has not been achieved. Further, they agree that achievement, in a general sense, will take some time. The possibility that the goal will be realized for the ACDS time period is reasonably good.

From General Terhune's remarks it is clear that today's situation is intolerable. He points out that in order to take advantage of a digital computer for the solution of a problem, today's user must develop a clear and precise picture of his requirements, must communicate the requirements to a programmer, and then must wait for, first, the program production and, then, the computer solution. An implicitly programmed system would permit the user to solve his own problem immediately. In order to extend his current capability he would employ the identical language and methodology that he learned for routine use of his current capability.

Note that the essence of implicit programming is:

1) A special purpose application-oriented language/methodology for routine use.

2) An open-ended characteristic of this language/methodology so that it can be used to extend its current capability.

Note further that it is immaterial whether the language/methodology is procedural or non-procedural. This choice is determined by the requirements of each application.

Today, one of the most promising examples of an implicitly programmed system is the display-oriented console system. With systems of this type, the user can request or store both alphanumeric and graphical displays. He is thus able to receive from, and provide to, the computer information concerning relationships. In addition, because of the intimate man-machine interaction, the computer can be used to prompt the user with requests for information and to inform him immediately of inconsistencies and deficiencies in the information he provides. Current projects, now underway, give promise of yielding a complete implicitly programmed system as previously described.

In the ACDS context there appears to be a clear requirement for a number of such application-oriented language/methodology techniques, tailored to each class of users. There is no technical reason why all should not have an implicit programming capability.

8.6 CONCLUSIONS AND RECOMMENDATIONS

8.6.1 Conclusions

Innovations in field of programming are being made rapidly, though not as rapidly as innovations in hardware technology. These occur more rapidly than the innovations are being accepted. The following are some expected milestones and their implications.

1) By 1968 - The techniques of system analysis and synthesis will be sufficiently well developed that multi-programmed multi-processor systems can be designed with precision and greater ease than at present. Executive programs and operating systems will permit the programmer to be concerned exclusively with the application. These programs will already have been written by specialists. They will normally be an item the user purchases rather than develops.

2) By 1970 - On-line programming systems will probably be the dominant form of programming. Problem-oriented languages will have proliferated to the point that the lack of such language for command and control will be considered a great deficiency. Since the language processor will be produced by a specialist who writes to specifications, it too will be an item purchased especially for the system. At this point, the Navy will find it desirable to use especially trained naval personnel to generate programs in these on-line command and control languages.

3) By 1975 - The implicitly programmed system will be a reality. The need for the application-oriented programmer with special training will have disapproved.

Subsequent to 1968, the computer specialist will probably write all his programs, executives, diagnostics, language processors, checkout tools and the like, in a highly algorithmic machine independent language. Thus, the effort required to produce these programs, and consequently the cost of procuring them, will gradually diminish. It is expected that innovations in hardware logic and facilities will be closely tied to the requirements for processing this type of program.

With these conclusions in mind, it is recommended that the Navy consider improving its present program production facilities to accommodate the requirements of a future ACDS in two phases; near and long-term. In the near-term it is recommended that the Navy step up its use of state-of-the-art techniques so that changes in the organization of both the equipment and the software can be easily accommodated. In the long-term, it is recommended that the functions of personnel associated with the implementation and use of the command and control systems accommodate advances in command and control system technology.

8.6.2 Near-term Recommendations

It is recommended that the Navy make certain improvements to its current program production facilities which will have an immediate impact and provide the Navy with the capability to assimilate further improvements.

First, it is recommended that a monitor system be installed to improve the program checkout process. The first effect of the installation of such a system will be the reduction in computer time required for checkout by a factor of at least five. A second, and corollary effect will be a reduction in manhours required for checkout by a factor of two. Second, it is recommended that the CS-1 Compiler be replaced by a more powerful tool. In practice its use is limited to program assembly. JOVIAL provides the most suitable current model for the command and control application. Particular emphasis should be placed on the intermediate language and its processor so that the investment in the programs written in the primary language need not be lost. However, the possible rapid development of NPL or other new language has the potential to replace JOVIAL as the model language in the future.

Third, a more highly automated program and system test facility should be provided. The present procedure is to construct a computer program system and perform a lengthy sequence of manual operations. Since the sequence of manual operations is well documented, there appears no reason they could not be performed automatically. Thus gross tests could be made rapidly, and final tests could be made more repeatable.

Fourth, it is recommended that a means be provided to enhance the communication between the user of the system, the programmers, and the system designers. The user may not be qualified to design the system, but an interested user is surely the best source of information concerning mistakes which ought not be repeated. Similarly, although programmers are not qualified to design hardware, interested programmers are the best critics of proposed equipment.

Fifth, it is recommended that the computer be used in the maintenance of documentation. The major effect of this approach is the impersonal imposition of standards. Higher quality and intelligibility with less effort are corollary effects.

8.6.3 Long-term Recommendations

Future tactical command and control systems will undoubtedly increasingly incorporate time-sharing, multi-programming, and multi-processing concepts. It has been pointed out previously that a thorough system analysis is required to guarantee the performance of such systems. Also, a by-product of the analysis will be a well-defined software organization. With this information available, computer programming production ceases to be a problem. Hence, the major problems of system implementation will shift from programming to system design and system use.

For the long-term period, it will be necessary for the Navy to make those organizational and operational changes which will make effective system design possible and encourage the development and implementation of more effective operating procedures and techniques.

Effective system design is an organizational and technical problem, but the development of operational techniques will require the participation of Naval personnel who are intimately familiar with tactical problems. The programming concepts discussed which promise to be of most value in this contract are those of application-oriented languages, with implicit programming capabilities.

Section 9
ADVANCED USAGE TECHNIQUES

9.1 DEFINITION OF ARTIFICIAL INTELLIGENCE

Artificial intelligence efforts are typified by indefiniteness of the final system configuration or program. The engineer or the programmer presents a system whose final configuration is determined by its reaction to a learning or teaching environment. The term "artificial intelligence" is applied to an area which has been defined by many other titles. A favorite heading for this whole area is Bionics[1]. Other portions of the literature refer to:

1) Self-organizing systems
2) Adaptive or self-adaptive systems
3) Heuristics

Much of pattern recognition is usually included under one of the above headings.

The difficulty of programming computers became evident as soon as the early computers were built. One of the reactions to this barrier to the use of computers was the creation of programming languages; another reaction was investigations into the area of artificial intelligence. The computer by modifying its own structure (self-adaptive systems) or by modifying its program (heuristics programs) would take on part of the task of the problem solver. This task would be in addition to its already assumed task of computation.

Artificial intelligence, in this context, endowed machines with a new set of attributes. These had previously been considered as belonging only to living organisms and never to mechanical devices. These attributes included:

1) Recognition of the goal
2) Exploratory actions
3) Evaluation of the effectiveness of the exploratory action and achieving the goal
4) Modification of the actions (or the cause of the action).

The studies in the area of artificial intelligence have been exciting and stimulating. As yet, however, they have produced little which allows its techniques efficiently to supplant more conventional procedures.

Heuristics generally applies to the modification of a computer program. Pattern recognition is usually restricted to that portion of artificial intelligence analogous to visual identification. Bionics originally was restricted to electronic simulations of biological mechanisms. The term has become more embracing and may well replace artificial intelligence as the preferred expression.

9.1.2 Motivation Behind Present Research

The motivation behind much bionics research is based upon a need to extract the desirable features of biological mechanisms. Among these desirable features is the reliability of animal systems[2].

The search for patterns of reliability in animal systems represents a reversal of the original philosophy behind computers. Computers were supposed to be far more reliable than people. Indeed, this is still true. The reliability of computers, however, depends upon the system remaining undamaged. With computers, one is lucky to get even a wrong answer if any part of the system undergoes failure or injury. Animal systems, on the other hand, are not only capable of repairing damage, they are also usually capable of performing adequately despite the damage. As systems get more complex, the need for the animal type of reliability becomes more urgent.

Another reason behind the investigation of bionics systems is the tremendous capability of such systems for parallel behavior[1]. A certain amount of this behavior has always existed in analog computations. Digital computers are more typically serial in operation. The RW 400 polymorphic system was an early attempt to achieve parallel computation, as were other efforts at simultaneous processing.

The need to achieve either speed or efficiency by imitating some of the parallel capabilities of biological systems forms a powerful motivation in today's research. The most impressive of the parallel capabilities of the human system is that associated with recall from memory[3].

Animal systems are capable of learning through experience. The motivation behind research in heuristics has been to give computer programs a similar capability[4].

As computers have progressed from arithmetical devices to machines used in more complex situations, there has been a desire to develop some scheme that would allow the handling of concepts[5,6]. Any person ever faced with attempting information retrieval from a running text has felt the need to deal with concepts rather than with words.

Of course, some of the reasons behind the present research has been motivated by research for exploration of fundamental phenomena. There is no doubt that the problems are challenging and glamorous. Such motivaticn is honorable and has led to important developments throughout the history of our culture.

Another motivation equally ancient in our culture, though not always considered honorable, has been the desire to build a monster or synthetic animal system. The imitation of animal-like processes merely because they are animal-like pervades the literature. It is one of the ways of submitting many theories to experimental test.

9.1.3 Early Efforts

The first important mention of artificial intelligence appeared in Turing's classic article on thinking published in 1950[7]. Turing did not outline a functional machine; he described instead the possibility of creating a computer whose responses to questions would leave one unable to decide whether the questions had been answered by a machine or by a person. Turing went to great effort to attack objections to the concept of a machine actually thinking. Though these objections are still prevalent, a tremendous amount of research has since been done in the area of artificial intelligence.

In 1955 papers on artificial intelligence were presented at the Western Joint Computer Conference. Other early conferences included the Conference on the Mechanization of Thought Processes held at the National Physical Laboratory at Teddington, England in 1958,and ONR sponsored conferences in May 1959 and 1960[8,9].

The first contracts in self-organizing systems let by the Office of Naval Research were those to Rosenblatt and Widrow[10]. This work resulted in the Perceptron and the Adaline. Other early work in pattern recognition was due to Bledsoe and Browning[11].

9.1.4 Problem Description

9.1.4.1 Adaptive Behavior

Sklansky gives the following definition of adaption: "We define adaption as a manifestation in a machine of one or more of the following three possibilities:

1) Stability of success in a changing environment;

2) Growth of success in a fixed environment;

3) Stability of success in the face of failure of machine parts."[12]

Change is the highlight in Sklansky's definition; i.e., that one builds adaptive systems because the future is not properly predictable. Lack of predictability can also center around the individual's inability to define a problem. Thus, much of the efforts in the area of pattern recognition have not been due to the expectation that the researcher would be surprised, but rather centered around an attempt to have the machine discover important recognition characteristics for its own purpose.

9.1.4.2 Learning Systems

Rosenblatt used three types of elements in his learning systems[13]. These corresponded to sensing devices, associative elements and responding devices (see Figure 9-1).

Sensory elements are connected to associative elements and associative elements to response units. The connections between sensory and associative elements are made at random. Since there is no overwhelming argument as yet for any particular order of connections, random connections have generally been chosen as the most defensible technique.

Two types of connections are made: an excitatory connection and an inhibitory connection. Excitatory and inhibitory connections are also made in a random manner from the associative to the responding cells. Excitatory connections, when activated, cause the next receiving cell to respond. This response could consist of the cell sending a signal along its connectors. The response can cause the cell to integrate the signal with previously received signals. The resultant sum would then determine whether a signal is to be transmitted. Inhibitory connections either paralyze the receiving cell or reduce the resultant sum.

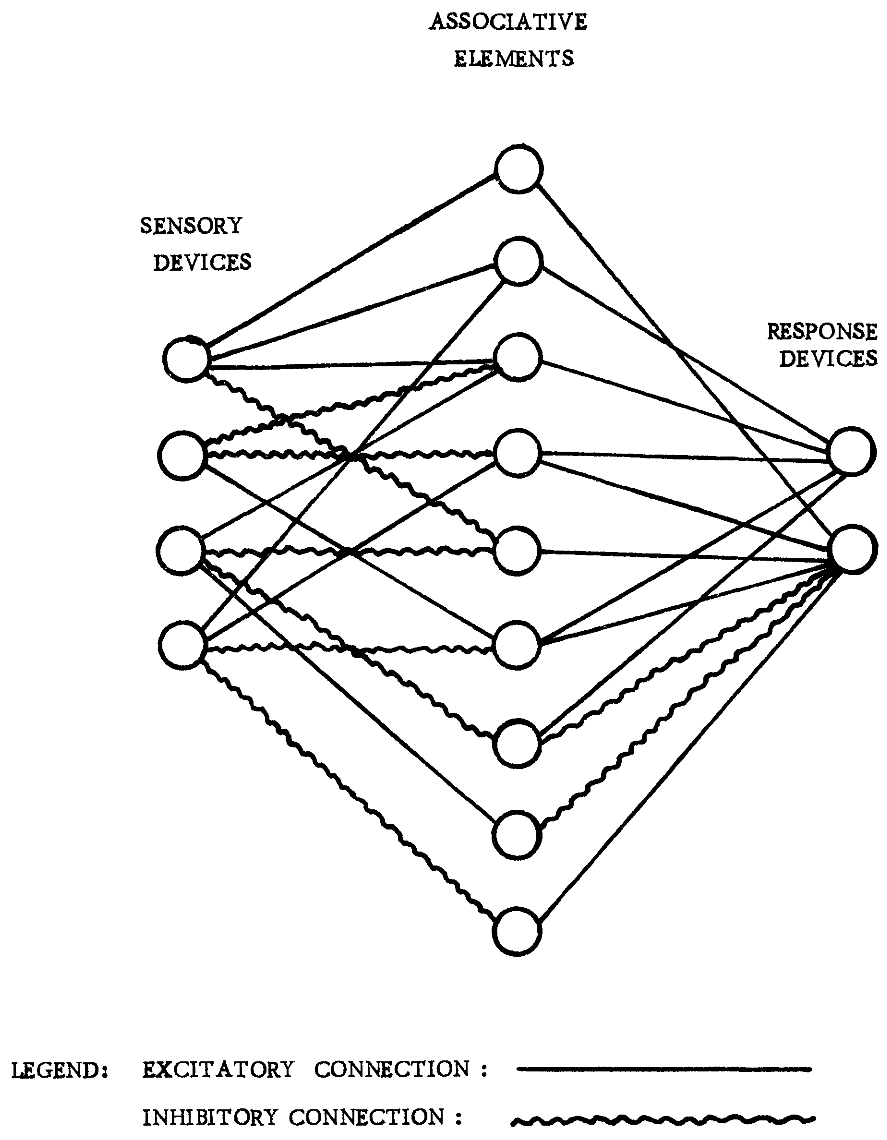


Figure 9-1. Perceptron-Type Learning Network

In some systems the amount of time for the signal to travel along the connections is a variable. Under other systems the receiving element, whether it be an associative cell or a responding cell, has a variable threshold. This variation might be due either to training or to a decay that is analogous to forgetting.

Some adaptive systems have used the technique of reward and punishment, i.e., if the system gives the correct response, the threshold of all cells participating in this response is lowered; if a system gives an incorrect response, the thresholds are increased.

Other systems merely punish. They use the technique of letting well enough alone when the correct answer has been given and of only tampering with the thresholds or the amount of time for a signal to travel along the connections when an incorrect answer has been received.

An alternate technique centers around the permissive attitude of not caring whether the system gives the correct or incorrect answer, but only that the answers are consistent; i.e., if a binary choice is provided, the system can define which response corresponds to "yes" or "no", to "left" or "right", to "1" or "0", etc. Under such a technique, the system uses an internal type of reinforcement. It modifies itself every time an input is received. The modification tends to make the present response to that input more likely.

Under the above systems, redundancy is a major factor since one cannot predict ahead of time which are the important associative elements and which are the important connections. A large number of elements tends to allow a satisfactory final state to be achieved.

In any of the systems the number of associative elements is very much greater than the number of sensory elements. In turn, the number of sensory elements is much greater than that of responding elements.

These elements are attempts to imitate neurons. Part of the stimulus towards redundancy comes from the tremendous amount of neurons existing in animal systems. Electronic neural networks have been created and their adaptive properties shown[14]. Other authors such as Griffith describe more complete neurons[15]. The design features of his more complete neuron centers around a more thorough imitation of the neurons existing in nature.

9.1.5 Techniques for Creating and Testing Systems

The systems that have been described lack capability. For example, the Mark I Perceptron could recognize only 8 letters anywhere in a 20 x 20 field or 20 letters placed in the center of the field[13]. It appears that in order to create a machine capable of making a thoroughly significant contribution toward the solution of previously unsolved problems, a large number of elements are needed.

When something is defined as machine learning, the natural tendency is to take advantage of digital computers to achieve computer simulation and test the proposed system. This has been the history of many efforts.

Arguments for the creation of special purpose hardware center around avoiding the slowness of digital computers. It is the parallel behavior of learning systems which gives hope for superior accomplishment; it is the same parallel structure which has made simulation on digital computers so slow. Rosenblatt points out that using special purpose equipment is much faster than computer calculation of whether the learning process will succeed[13].

A question of computer simulation of proposed learning systems versus creation of special purpose hardware is more than just a process of measuring the relative economic benefits. A danger exists that simulation may distort the problem.

Computer simulation often tends to make the researcher forget that the eventual fabrication must use many simple (and, therefore, hopefully inexpensive) elements placed together. Too often, what is simulated can only be fabricated when one has a powerful central processor as an aid.

An alternative to either computer simulation or the creation of special hardware would be simple algorithms testing whether the system would converge to a useful configuration. Mays states: "Algorithms for nontrivial networks of adaptive elements are not very well understood....I know of proposed algorithms for such networks, but I have no satisfactory proof that algorithms will cause convergence to a solution in a finite amount of time. However, experimental results indicate that these algorithms cause convergence most of the time."[16]

Fitzhugh is working on an approach analogous to statistical mechanics[17]. His consideration of polystable systems may yield some analytic clues as to whether it is worthwhile to assemble large systems without having to go to the expense of computer simulation.

The large number of elements needed for even simple useful systems is highlighted in Herscher and Kelley's paper[18]. They describe the functional model of the frog's retina. The model incorporates the four basic functions of the frog's sight and requires 32,000 individual circuit components.

Another system referred to was Large Artificial Nerve Net (LANNET) which uses a 1024 decision element network[19]. Expense can be inferred by the fact that the system is large; its modest capability is indicated by preliminary tests reported on for the running of a simple maze.

9.1.6 Pattern Recognition

The program of the 1955 Western Joint Computer Conference included three pioneering papers in pattern recognition[20,21,22]. In this early work, Dinneen was concerned with some experiments with block capital As, and Os, some squares and some triangles. Even with this comparatively simple universe to deal with, the paper is more concerned with internal transformations of the input so that recognition can be attempted than with the results of actual recognition.

Successful recognition was achieved by Rosenblatt with the Perceptron. As noted earlier, the Mark I Perceptron could recognize eight letters anywhere in a 20 x 20 field or 20 letters placed in the center of the field[13].

The Adaline of Widrow also performed pattern recognition. Rather than being concerned with character recognition, early Adaline or Madaline experiements were concerned with observing an unstable pendulum (balanced from the bottom) so that the rest of the learning system could balance the pendulum[23].

The close alliance of pattern recognition and learning machines is emphasized by the fact that Rosenblatt's work and Widrow's work are considered pioneering efforts in machine learning and are referred to by ONR, the sponsoring agency, as their two earliest contracts in pattern recognition[10].

Warren McCulloch and associates performed pattern recognition experiments that centered around the simulation of the frog's eye. Early experiments with the retina of a frog's eye showed that a frog's vision consisted of four different operations:

1) Sustained contrast detection
2) Net convexity detection
3) Moving edge detection
4) Net dimming detection[25]

Herscher and Kelley are able to report that they have a functional model of the frog's eye which incorporates the four basic feature abstraction functions[18]. As was previously stated, this required 32000 individual circuit components. The frog's eye, however is very simple as compared to human eyes. (The frog does not seem to see stationary objects. He will starve to death surrounded by food if it is not moving[25].) The solution of the problem for the frog's eye does give high promise toward the capability of producing general visilogs.

9.1.7 Character Recognition

Much of the early pattern recognition work found its first practical outlets in the area of character recognition. The important advances that have been made in this area are illustrated by the two facts observed below. They indicate that character recognition equipment should be commercially available as off-the-shelf items in the 1970 era.

The present state-of-the-art of character recognition is such that the Department of Revenue of the State of Colorado has contracted for equipment that will recognize typewriter input at the rate of 2000 characters per second. Though similar purchases have been made by associated research branches of the Department of Defense, there was usually the recognition that these agencies have the mission to extend the state-of-the-art. State revenue bureaus have no such motivation.

Bryan, in his work in character recognition, had to reject typewriter input and constrained handprinting in his experiments in adaptive pattern recognition because they caused few errors[24]. His equipment rejected one element of randomness. Every sensory element was connected to every associate element. There were as many excitatory as inhibitory elements. Randomness was applied to the selection of which lines were to be excitatory.

9.1.8 Learning

9.1.8.1 Examples

Widrow's broom balancing experiment is an example of a type of learning that present systems are capable of achieving. Learning in Widrow's case was to consist of the system recognizing the position of the broom (pendulum) and moving a small truck in its one dimensional degree of freedom in order to keep the broom from falling. Other types of learning are those associated with the pattern recognition referred to previously. Experiments have also been conducted in games such as the Tower of Hanoi and with more serious problems such as weather prediction. The complexity attainable is still not impressive. Earlier mention was made of the fact that maze running was considered an adequate preliminary test to something labeled as a large neural net. Examples of learning which are impressive because something useful and previously unattainable has been achieved still await us. The work already done is highly impressive because mechanical devices have exhibited behavior analogous to similar animal learning.

9.1.8.2 All or Nothing Accomplishment

Rock and Estes have performed experiments showing that learning is accomplished on an all or nothing basis[26]. These experiments show that human subjects do not converge in a learning situation. Things snap into place without partial learning having occurred. Learning machines, on the other hand, tend to converge to a solution. The disparity towards the two types of behavior is indicative of the extreme remoteness of animal learning to supposedly imitative machine learning.

9.1.8.3 Feasibility of Useful Learning Machines

At the 1964 Fall Joint Computer Conference, Mays indicated that the pre-processing of weather information was of the highest importance in achieving successful convergence in experiments performed at Stanford University[16]. This pre-processing reflected an already understood method of solution on the part of the researcher. In general, the work that has been done has centered around problems that we know how to solve. The question arises as to whether there are significant problems capable of being handled by learning machines. Certainly the most impressive learning machine available is a human being. With today's fear of the population explosion, we seem to be in no danger of a shortage of available people far more intelligent than any machine that has been proposed. With estimates running from 10^{10} neurons to 10^{15} neurons for the human nervous system, it would appear that there is plenty of spare capacity to handle any problem that could be presented to a bionic system.

Justification for using learning machines depends on environments to which humans cannot adjust.

9.1.9 Heuristics

Two types of heuristics have dominated. One type is heuristic programs; the other permits the problem solver to use heuristic techniques.

Heuristic programs lead to methods of solution. Minsky presented a list of methods used in heuristic programs[4]. They are:

1) Methods which set up searches.
2) Methods which set up new problems as sub-goals.
3) Methods which set up new problems as models.
4) Methods of characterization or description.
5) Administrative methods which include:
    a) difficulty estimates;
    b) self-problem utilities; and
    c) methods for selecting methods.
6) Methods of self-improvement.

Impressive results have been achieved using heuristic methods in playing games on computers. Papers describing chess playing, checker playing and theorem solving programs have been published [27,28,29,30]. Though the results have been impressive, as yet nothing practical has been achieved. Difficulties center around the relative slowness of digital computers in doing "thinking." A reasonably trained human can outrace the computer and is, of course, far less expensive.

The use of heuristic techniques by the problem solver has been more effective than heuristic programs. Rather than attempt to foresee all the possible program difficulties, the problem solver is allowed to use his superior intelligence and adaptability.

A man-machine partnership is created. This is in contrast to an attempt to endow the machine with a man's heuristic capabilities. Thus, through the use of consoles, man provides the essential ingredients of goal recognition, exploratory action, evaluation, and, if necessary, modification of procedure.

Major emphasis is now concentrated on easing man's ability to communicate with the machines through consoles. In contrast, heuristic programs receive only minor attention.

9.1.10 Competition with Human Processes

9.1.10.1 Associative Recall

One of the most impressive characteristics of the human memory is its ability to accomplish associative recall. Until recently, computer memories required serially-performed searches if the exact address was unknown. Hardware developments leading towards associative memories where the total memory is searched in parallel have attempted to remedy this deficiency.

In the 1962 Spring Joint AFIPS Conference, Richard F. Reiss presented a paper describing a nine-postulate machine[3]. The purpose of this machine was to simulate previously held psychological interpretations of the way the human memory operated. The operation of these postulates on a computer would have been a slow and expensive procedure. Reiss' paper and the non-artificial intelligence hardware researcher in the area of parallel search memories highlight the need for solving the problem of associative recall if any of the artificial intelligence devices are to successfully simulate simple human intelligence.

9.1.10.2 Humanoids

Much of the work has centered around attempts to build humanoids. The computer art has given another outlet to this drive. Many of the computer systems described in the literature are anthropomorphic.

Bionic systems which imitate animal systems for the purpose of learning by modeling are understandable. Most of these systems, however, do not reflect modeling of biological aspects so much as they reflect an attempt to create something which would imitate a biological mechanism.

9.1.10.3 Adaptive Systems

There are many adaptive systems that have been placed in computer programs that are not identified as such in the literature. Certainly, every program has built-in modification features. Most take different paths depending upon the input data. The general difference between those adaptive programs running on computers today and those reported in the literature as adaptive programs is that in the former case the problem solver has determined to use his intelligence to dictate the procedures that will be followed. The latter programs find researchers using devices such as randomly selected connections to avoid imposing control. Yet, as was previously observed, problems solved this way are not significant problems. Behind much of this work resides the hope that possibly a machine so constructed can learn to solve problems that are beyond us.

9.1.11 The Future

9.1.11.1 Projected Program

An interdisciplinary creation of scientists whose basic training is in biology but with the ability to take advantage of mathematical and computer tools is called for in order to give impetus to the present work[31]. One has the feeling that once those more trained in the sciences enter the field, major advances will take place. The direction that these advances might take are given by Gwinn and Butsch as follows:[32]

1) Primary Elements and Techniques in Engineering Bionics:
    a) Biological laws and effects (in progress)
    b) Visilog (in progress)
    c) Analog of the ear (in progress)
    d) Living cellular analogs (planned)
    e) Biological materials and physical analogs (planned)

2) Man-Machine Interface:

   a) Electromyograph (in progress)

   b) Muscle substitute (in progress)

   c) Tactile perception (in progress)

   d) Intercommunication concepts and compatibility (planned)

   e) Analysis of intercommunication mechanisms (planned)

   f) Psychophysical effects (planned)

   g) Biological instrumentation (planned)

3) Bionic Subsystem Techniques:

   a) Probability state variable (PSV) devices (in progress)

   b) Crystal color centers (in progress)

   c) Bionic logic theory (planned)

   d) Structure of the central nervous system and physical analogs (planned)

4) Bionic Systems Work Elements:

   a) Peripheral access lattice structures (in progress)

   b) Neurotron networks (in progress)

   c) Epistemology (fundamentals) (in progress)

   d) Adaptive "Sandwich" (in progress)

   e) AF system decomposition into operating parameters (planned)

   f) AF system goal structure and analysis (planned)

   g) Statistical analysis of performance (planned)

   h) Reliability of system through bionics (planned)

5) Experimental Synthesis of Bionic Subsystems and Systems:

   a) Theoretical studies of network behavior (planned)

   b) Empirical studies of network behavior (planned)

   c) Multidirectional information transfer techniques (planned)

   d) Associative memories (in progress 1/4)(planned 3/4)

6) Experimental Adaptive and Autonomous System

   a) Artron (artificial neuron) networks (planned).

   b) Optical neuromimes (planned)

   c) Network access research (planned)

   d) Taxonomy of adaptive systems (in progress)

   e) Information coding of living systems and physical analogs (in progress 1/3) (planned 2/3)

   f) Sequential network logic (in progress 1/4) (planned 3/4)

   g) Error detecting and correcting codes (in progress 1/4) (planned 3/4)

7) Structure-Functional Relationships Work Elements:

   a) Relationships between information measures and matter energy (planned)

   b) Bionic evolution (planned)

9.1.11.2 1970 to 1980

Pattern recognition of alphanumeric characters has already been shown feasible. Pattern recognition of more arbitrary features will probably be capable of implementation. Much of the ability to recognize patterns will depend upon the capability to indicate the desired patterns. Present accomplishments in character recognition are due to the ability to specify the recognition set.

Machines exhibiting artificial intelligence in a self-organizing sense are unlikely to find practical implementation during the 1970-1980 period. The large number of components required and the need to find an inexpensive way to assemble them makes it unlikely that such learning devices will be able to compete with humans.

Heuristics was a sought after goal of the late 1950s. Its attraction was based upon having a digital computer show a semblance of doing what had previously been considered uniquely human tasks. The early 1960s have shown the tendency to depart from the spectacular imitation of animal-like processes and substitute in its stead man-machine cooperation. On-line man and machine shared processing, supported by capable bilateral communications between each, is exhibiting a reasonable approach towards combining the best capabilities of man and machine. The success of such schemes will depend on the efficient presentation of information to man and clear, unambiguous, man-made machine instructions. It seems unlikely that a machine will have the full responsibility for heuristic procedure. It is far more likely that the machine will present a set of already stored possibilities to the operator upon request. Such implementation does not fall into the area of heuristics usually referred to in literature.

Bionics will probably be utilized for obtaining a better understanding of how biological processes work. It is unlikely that major bionic advances will result in practical implementation for the 1970 to 1980 ACDS. As indicated above, that decade will probably be dominated by man-machine relationships rather than acute dependence upon man as in earlier decades or a devotion to the automatic processes which highlighted the 1950s. The field of artificial intelligence consists of a series of brilliant first experiments whose major results in support of computation must be attained.

9.2 SYSTEM SELF-DIAGNOSIS

Insofar as command and control systems are completely digital in the makeup of their machinery, they are also subject to complete and automatic self-diagnosis procedures. This is so because the descriptive rules for the correct operation of any digital system form a precise and finite set.

When the machinery of a command and control system includes analog devices, the problem of self-diagnosis is more complicated if only because the rules of what constitutes "correct" operation of an analog device cannot be stated in terms of bi-valued (Boolean) logic. Hence, such rules are not readily used by a digital system. Recent attempts to make systems completely self-diagnosing have been rare and incomplete.

9.2.1 Graceful Degradation

Graceful degradation, or the ability to lose parts of the system and continue to function, is an important requirement in many systems. Frequently it is permissible to suspend some system functions while continuing priority functions at full rate. In other instances, it is more suitable to continue all functions, but increase time delays, or lower sequencing or repetition rates. Some allowances for degraded performance are not related to system health, but rather to an abnormal traffic state. For instance, in the QUOTRON system queries for extended lists of stocks and their performance, where the lists are selected by category, (i.e., industrials) are deferred during periods of high traffic in single, individual queries. The lists are compiled and sent when the traffic load returns to normal, and even then they are handled on a low-priority basis.

Possibly the simplest kind of graceful degradation is found in design of the AN/UYK-1 computer. This computer has the capability of storing all contents of memory, plus the contents of necessary registers, if power fails. A special interrupt condition is generated at restoration of power, and a permanently-stored power-failure interrupt routine permits putting the computer back on the air.

Diagnostic and checkout in the RTDHS of Pacific Missile Range contains many of the elements of graceful degradation, and elements of preventive diagnosis as well. The checking out of system components and diagnosing fault conditions have been included under control functions of the executive programs, since such activities can be thought of as special cases of exercising the system programs.

The flow of system checkout from one type of test to another is essentially a control function performed by the master control program. Use is made of the regular system programs, since these are as much a part of the system as the hardware.

The system checkout program provides for cycling through a predetermined set of tests, and for repetition of a test or tests at discretion of the console operator or from an external request. The objective of the system checkout routine is to assure that all system components are operating properly, and that the system operates as a whole when the individual components are made to function together. A secondary task is to display the system condition to the system controller or the console operators. The system checkout program may be required to demonstrate to the missile flight safety officer that the system is functioning properly.

The ability to recycle arbitrarily through any set of checkout routines furnishes a valuable diagnostic aid for use throughout the whole system. For instance, communications may be repeated or radars may be driven in selectable fashion. This permits maintenance personnel to use the computer to pinpoint trouble spots.

One of the earliest uses of redundancy to check performance was found in the UNIVAC I, the first large-scale, high-speed computer offered for commercial sale. UNIVAC I made use of dual arithmetic and control units which worked in parallel. Self checking circuits looked for comparison between the units at each clock time and set a fault and halt condition at disagreement.

Graceful degradation is found in the navigation system of Polaris submarines. There the AN/UYK-1 computer and its associated data processor (a timing, control, and sequencing device) control the high-accuracy (satellite doppler) navigation set. Programs permit complete diagnosis of this set, and are set up to derive useful information from imcomplete messages, and to remember previous information and select the best of it. But this navigation set is only an adjunct to the NAVDAC computer which makes use of such inputs as celestial and radio fixes, ship's log, gyrocompass, etc., as well as servicing the inertial navigation system, and doing continuous dead reckoning. Complete failure of all the automatic navigation systems still leaves the possibility of dead reckoning and celestial navigation by hand, using the time-honored methods. Still there are about three levels of redundancy in the built-in systems, each yielding a slightly less accurate navigation basis.

9.2.2 Multi-Computer Health Check

The problems of program recovery and fault discovery in multi-computer systems are closely related. Some of the computing errors may be prevented by periodic health checks of the system. The amount of checking which can be done is actually a function of the time that is available between the operational processes that the computers must perform. However, some time must be allocated for checking. For checking purposes we can design a background diagnostic which will check the system when it would otherwise be idle.

It may, however, be necessary to give the diagnostic program a definite priority so that we can be assured that a certain minimum amount of checking will be done.

The scope of the diagnostics can vary greatly. Generally the diagnostics will check communication links and the status of peripheral devices. They will also include cross checking between the computers of the multi-computer system. The transmitting computer during each communication might, for example, transmit a dummy computation result so that a carefully selected set of instructions are performed, all the circuits are exercised, and a coded result is sent with the communication.

The receiving computer performs the same exercise and compares its results with that of the other computer. A difference would initiate further checking of the communication link and of the individual computers.

The chief aim of the diagnostic health check is to discover failures as quickly as possible after they occur. If this is done we then have a chance of taking remedial action before important data can become lost.

Some failures affect the data but not the programs. These can be discovered by programming methods. Failures which cause the equipment to stall cannot be discovered by programming methods but require hardware features for their detection.

9.2.3 Hardware Requirements

The achievement of capability for graceful degradation in computer systems places some requirements on hardware design. While the programming of graceful degradation is primarily a software function, certain hardware characteristics are essential to permit programming of a control program that can function and determine the system configuration.

Possibly the most straightforward example of the needed hardware characteristics is found in the hardware (and software) of the Burroughs DUAL B-5000. The needed characteristics are the following:

1) Processor interrupt logic. It must be possible for any processor to interrupt any other in the system, and to interrupt itself. The logic design of interrupts takes various forms, and is often related to a wired-in priority scheme.

2) There must be a switching scheme, controllable from the master processor (and more than one processor must, of course, be able to control, or access and execute the master control program) permitting refiguration of the system from the hardware and interconnection standpoint.

3) Extensive ability for control and data communication, with provision for alternate paths, must be built into the system.

4) The ability to add modules, either in real or extended time, permits the system to access or release equipment (or to grow with requirements of environment which may change with time). In some systems this ability in real time permits functioning as a single large system or a number of smaller ones.

9.2.4 Fault Isolation

Maintenance and reliability are two concerns facing the command and control system planner. A necessary maintenance technique is that of locating and correcting a fault when one occurs.

The usual approach to solving this problem is the writing of diagnostic programs for the system computer or computers. Such programs detect and isolate computer failures by exercising the command structure of the machine. Often this approach does not work because it is predicated on an operating computer. The computer may be malfunctioning so that the diagnostic program cannot even be read into memory.

Manual diagnostic procedures can be developed which guarantee to detect and isolate faults when the computer is incapable of reading a diagnostic program. To accomplish this, it is necessary to relate each operation code to the logic structure of the machine and its reaction to the code. Using a computer to break down the logic equations of each system device into a set of elementary functions, manual procedures can be defined which will provide fault detection and isolation.

It is always possible to establish a basic level of operating logic functions, even at the lowest stage. Using this level as a base, and executing the manual procedures, it is possible to extend this level upward until eventually the entire system is functioning properly.

It is also always possible to take the logic equations for every digital device in a system and break them down into simple logic functions.

The logic equations for each digital device form a system master logic file. By using information retrieval and sorting techniques on this file it is possible to create tables showing the interrelationship of every logical element in the system. These tables can then be used to develop both manual procedures and diagnostic computer programs, by combining elemental functions from the various tables.

The procedures can be structured to isolate faults by a process of elimination; i.e., if a procedure does not work correctly, another is tried that uses almost the same logic. This process of elimination is carried out until the fault is isolated.

First the necessary logic equations must be derived. The next step is to take these equations and examine them to determine how each operation used by the system relates to the logics. Once this has been done, it is possible to design the diagnostics needed.

A diagnostic is usually a straightforward series of operation codes which attempts to diagnose the system by testing the operations in a logical sequence. But diagnostics can also use this approach of testing the machine logic structure, rather than operation code structure.

By selectively introducing malfunctions into a system Boolean model, it is possible to arrive at a diagnostic which is capable of detecting all possible faults.

Once the method of designing and implementing a fault-isolating program has been derived, it is a short step to apply it to any digital system.

Section 10
SOURCE MATERIALS AND REFERENCES

10.1 INTRODUCTION

This section contains source material and reference material from Sections 2 through 9, arranged in numerical order.

10.2 SOURCE MATERIAL

The various organizations listed in the following sub-sections have been contacted by personnel assigned to this study. Discussions with these organizations provide a basis for much of the information presented in this report. The discussions covered techniques and approaches that have not been adequately described in published literature and the opinions of experts in specific areas in these organizations were solicited concerning the advantages, limitations and future prospects for the various techniques and procedures described in this report.

10.3 REFERENCES

An extensive list of references pertinent to the study of the various topics described in this report are listed by section. A study of these references has contributed to the material presented in this report. Some of the more pertinent and important of these have been referenced in the text and listed in the section as specific numbered references. These numbers correspond to numbered citations in the test. Direct quotations have also been used where noted.

10.4 SECTION 2 INPUT/OUTPUT SOURCE MATERIAL AND REFERENCES

10.4.1 Organizations Contacted

Input/output technology and equipment have been discussed with representatives from a number of government agencies and manufacturing companies in the course of this study. The following are a list of major sources of information and the techniques which have been discussed with each.

Army Electronics Research and Development Group Fort Monmouth, New Jersey	Keyboards Printers Incremental magnetic tape Punched paper tape
Bridge Division of CDC Philadelphia, Pennsylvania	Punched card equipment Printers
Control Data Corporation	Punched cards Paper tape Magnetic tape
Data Equipment Company Santa Ana, California	Graphical input devices
Data Products Corporation	Militarized printers
Digital Data Corp.	Incremental magnetic tape
Fairchild Semiconductors	
General Dynamics Electronics San Diego, California	Printers
Honeywell Boston, Massachussetts	Mildata storage
National Cash Register Dayton, Ohio	Mildata storage
National Cash Register Hawthorne, California	Character recognition
Philco Computer Division	Punched card Printers

Philco Research Lab	Character recognition equipment
Potter Instrument	Printers Incremental magnetic tape
Remington Rand Univac New York, New York	Punched card Magnetic tape Printers Character recognition
Remington Rand Univac Philadelphia, Pennsylvania	Punched card Magnetic tape Printers
Remington Rand Univac St. Paul, Minnesota	Magnetic tape
Rome Air Development Center Rome, New York	Keyboards Printers

10.4.2 References

1. "A Method of Voice Communication with a Digital Computer," S. R. Petrick and H. M. Willett, Proceedings Joint Computer Conference, New York, N.Y., Vol. 18, pp. 11-24, Dec. 13-15, 1960.

2. "System Reads Three Type Fonts," J. M. Carroll, Electronics, pp. 49, Dec. 20, 1963, McGraw Hill Publication.

3. "Programming Pattern Recognition," G. P. Dinneed, Proceedings Western Joint Computer Conference, Los Angeles, California, March 1-3, 1955.

4. "Digital Data Communication Techniques," J. M. Wier, Proceedings of the IRE, Vol. 49, No. 1, pp. 196-209, Jan. 1961.

5. "Printing Equipment for Medium, Intermediate, and Large Size Computers," Staff of Cresap, McCormick and Paget, Control Engineering, Jan. 1962, pp. 91-95.

6. "Digital Printers, Editorial Survey," Instruments and Control Systems, Vol. 32, May 1959, pp. 700-707.

7. "Tape Printer Applications," W. R. Beall, Instruments and Control Systems, Vol. 32, May 1959, pp. 708-709.

8. "Analog Input and Output System for a Real-Time Process Control Computer System," C. A. Walton, Joint Automatic Control Conference--Proceedings 13, 4.1-4.6, June 1962.

9. "Punched Card as Information Carrier and as Technical Problem," K. Lindner, Feinwerktechnik, 67(2) :55-61 1963.

10. "Digital Circuit Techniques for Speed Analysis," G. L. Clapper, IEEE-- Transactions on Communications and Electronics 66:296-304, May 1963.

11. "Minutes of ASA Committee X3.1"(Optical Character Recognition) and its Subcommittees, American Standards Association, Sectional Committee X3 on Computers and Data Processing.

12. "Machine recognition of human language," Nilo Lindgren IEEE Spectrum Mar. 1965 pp. 114-136.

13. "The Rand Tablet: A Man-Machine Communication Device," Mr. M. R. Davis, T. O. Ellis, p. 325.

10.5 SECTION 3 DISPLAY SYSTEMS REFERENCES

1) "The Information Display Field as it Exists Today," R. M. Davis, Proceedings of the Third National Symposium of the Society for Information Display, February 1964.

2) "DODDAC-An Integrated System for Data Processing, Interrogation and Display," W. F. Bauer and W. L. Frank, Proceedings of the Eastern Joint Computer Conference, December 1961.

3) "Utilization of Lenticular Film for Full Color Display Systems," D. J. Morrison and H. N. Oppenheimer, Proceedings of the Fourth National Symposium of the Society for Information Display, Washington, D.C., October 1964.

4) "Information Handling in the Defense Communications Complex," T. J. Heckelman, and R. H. Lazinski, Proceedings of the Eastern Joint Computer Conference, 1961.

5) "The ICONORAM System," G. W. N. Schmidt, Datamation, January 1965.

6) "COGS, A Generalized Display Language and System," J. H. Dinwiddie, Paper presented at symposium sponsored by Chespeake Bay Chapter of Association for Computing Machinery, Baltimore, Maryland, June 1964.

7) "Input/Output Software Capability for a Man/Machine Communication Image," T. R. Allen and J. E. Foote, Proceedings of the Fall Joint Computer Conference, October 1964.

8) "A Computer Technique for Producing Animated Movies," K. C. Knowlton, Proceedings of the Spring Joint Computer Conference, April 1964.

9) "Programmer's Handbook, IOC DPC Utility and Test Subsystem Task Test Tool," J. Adler, C. Brody, and G. Farrance, TM-LO 741/020/00, SDC, April 1963.

10.6 SECTION 4 DISPLAY TECHNOLOGY SOURCE MATERIAL AND REFERENCES

10.6.1 Organizations Contacted

Display techniques and systems have been discussed with personnel of a number of different companies and governmental agencies in the course of this study. The following list indicates the companies and governmental agencies with whom display technology has been discussed and the type of displays discussed with each:

Bunker-Ramo Corporation Canoga Park, California	Light-valve displays Continuous-strip photographic projection displays CRT displays
General Dynamics/Electronics San Diego, California	Charactron CRT displays Light-valve displays
General Telephone Laboratories Bayside, Long Island, N. Y.	Continuous-sheet electroluminescent displays with XY matrix addressing Acoustic/electroluminescent displays
RCA Laboratories Princeton, New Jersey	Ferroelectric displays
Laboratory for Electronics Boston, Massachusetts	Magnetic thin-film displays
NCR National Cash Register El Segundo, California	Electromechanical photochromic displays CRT-Photochromic projection displays
Sylvania Waltham, Massachusetts	Discrete alphanumeric character displays Continuous-sheet electroluminescent displays with XY matrix selection
Stanford Research Institute Menlo Park, California	Magnetic thin-film displays Modulated crystal filter displays
Rome Air Development Center Rome, New York	Light-valve displays,Modulated crystal interference-filter displays Thermo-plastic displays Laser displays

USAER&DL Fort Monmouth, New Jersey	Photochromic displays Laser-luminescent displays Electroluminescent displays Fiber-optic CRT displays Lenticular film Photographic projection displays Display memories
U.S. Navy Bureau of Ships Washington, D. C.	Display equipment Display systems Photographic projection displays Lenticular film Light valves Electroluminescent displays CRT displays Large screen displays
Hughes Ground Systems Fullerton, California	Large screen displays Console type displays
U. S. Marine Corps, Bureau of Ships, Washington, D. C.	Tactical displays Large screen displays
Librascope Glendale, California	Libracote display panel

10.6.2 References

1. "The Information Display Field As It Exists Today," Davis, R. M., Proceedings of the Society For Information Display, Third National Symposium on Information Display, pp viii-xv, San Diego California, February 1964.

2. "Displays, Papers, and Lighting," Stocker, A. C., Information Display, Vol. 1, No. 1, pp 16-26, September/October, 1964.

3. "A Summary of the State-Of-The-Art of Large Scale Military Displays," LaSalle, R. H., RADC Technical Report, Rome Air Development Center, December 1964.

4. "Display System Design," Loewe, R. T. Electronic Information Display Systems, Spartan Books, Washington, D. C., 1963, pp. 15-37.

5. "Basic Concepts in the Logic and Organization of Displays for the Norad Command and Control Systems," Duffy, M. L., and Smith, C. D., Proceedings of MIL-E-CON 7, Washington, D. C., Sept. 9-11, 1963, pp. 251-253.

6. "Physical Principles of Displays - Classification," Talmadge, H. G.,Jr., Electronic Information Display Systems, Spartan Books, Washington, D.C., 1963, pp. 69-86.

7. "Cathode-Ray-Tubes," Darne, F. R., Electronic Information Display Systems, Spartan Books, Washington, D. C., 1963, pp. 87-109.

8. "Criteria for Group Display Chains for the 1962-65 Time Period," Technical Documentary Report No. RADC-TDR-62-315, Rome Air Development Center, pp. 1-2, July 1962.

9. "Digital-To-Visible Character Generators," Boyd, S. H., Electro-Technology, Vol. 75, No. 1, pp. 77-88, January 1965.

10. "Some Pragmatic Considerations Influencing the Selection of Air Force Display Techniques," Kennedy, E. J., Proceedings of the Society For Information Display, Third National Symposium on Information Display, pp. 1-17, San Diego, California, February 1964.

11. "Which Read-Out?," Part I, II, III, IV, EEE- The Magazine of Circuit Design Engineering, August, September, October, November, 1964.

12. "Advanced Display Techniques Through the Charactron Shaped Beam Tube," Redman, J. H., Society for Information Display Symposium, March, 1963.

13. "Display Requirements of the Integrated Management Information Systems - 1968-1970," James, P., and Dittberner, D., Information Display, Vol. 1, No. 2, pp. 26-31, November/December 1964.

14. "A Synopsis of the State of the Art of Dynamic Plotting Projection Displays," Anderson, R., Second National Symposium of the Society for Information Display, New York, October 1963.

15. "Colordata Display," Hughes Aircraft Co. Brochure, Fullerton, California, 1963.

16. "Artoc Displays," Loewe, R. T., Electronic Information Display Systems, Spartan Books, Washington, D. C., 1963, pp. 231-246.

17. "DODAC - An Integrated System for Data Processing Interrogation and Display," Bauer, W. F., and Frank, W. L., Proceedings EJCC, Washington, D. C. December 1961.

18. Unpublished Manuscript, Luxenberg, H. R., 1964.

19. "Kalvar," Nieset, R. T., Data Systems Design, Vol. 1, No. 9, pp. 43-44, September 1964.

20. "New Approaches in Photography," Robillard, J. J., Photographic Science and Engineering, Vol. 8, No. 1, pp. 18-34, January, February, 1964.

21. "Photochromic Dynamic Display," Haley, E. J., Electronic Information Display System, Spartan Books, Washington, D. C., 1963. pp. 110-120.

22. "Epic Display," Bjelland, H. L., Proceedings 3rd National Symposium on Information Display, San Diego, California, pp. 286-299, February 1964.

23. "Real Time CRT Photochromic Projection Display," Stetten, K. J., Society for Information Display Symposium, Washington, D. C., October 1 - 2, 1964.

24. "Solid Crystal Modulates Light Beams," Lindberg, E., Electronics, Vol. 36, No. 51, pp. 58-61, December 20, 1963.

25. "Electroluminescence Principles and Applications," Strock, L. W., IEEE Spectrum, pp. 68-74, November, 1964.

26. "Display Applications of Electroluminescence," Wasserman, M. S., Electronic Information Display Systems, Spartan Books, Washington, D. C., 1963, pp. 121-125.

27. "Electroluminescent Display Devices," Greenberg, Irving, IEEE Spectrum, pp. 75-83, November 1964.

28. "Non Linear Resistors Enhance Display Panel Contract," Blank, H. G., O'Connell, J. A., and Wasserman, M. S., Electronics, August 3, 1963.

29. "Solid State Display Device," Yardo, Stephen, Proceedings of the IRE, December 1962.

30. "The Application of Electroflors to Display Instrumentation," Alburger, J. A., Photo-Optics Workshop, Sponsored by Society of Photographic Instrumentation Engineers, Los Angeles, California, June 1, 1964.

31. "Laser Display Techniques," Rugori, A. D., Proceedings Second Society for Information Display National Symposium, 1963.

32. "Laser Displays," Fowler, V. J., Data Systems Design, Vol. 1, No. 9, pp. 50-51, September 1964.

33. Lasers, Lengyel, B. A., John Wiley & Sons, Inc., New York, New York, 1962.

34. "A Fast, Digital-Indexed Light Deflector," Kulcke, W., Harris, T., Kosanke, K., and Max, E., IBM Journal of R & D, Vol. 8, No. 1, pp. 64-67, January 1964.

35. "Special Report: Advanced Displays," LaFond, C. D., and Pay, R., Missiles and Rockets, Vol. 14, No. 14, pp. 31-37, October 5, 1964.

36. " A True 3-D Display," Ruderfer, M., Data Systems Design, Vol. 1, No. 9, pp. 11-14, October 1964.

37. "The Little Train That Wasn't," Novotny, G. V., Electronics, Vol. 37, No. 30, pp. 86-89, November 30, 1964.

38. "A Review of Panel-Type Display Devices," Josephs, J. J., Proceedings of the IRE, August 1960.

39. "Optoelectronics - Key to Isolation," Wunderman, I., and Loebner, E. E., Electronic Design, Vol. 13, No. 1, pp. 30-35, January 4, 1965.

10.7 SECTION 5 COMPONENT AND PACKAGING TECHNOLOGY SOURCE MATERIAL AND REFERENCES

10.7.1 Organizations Contacted

Components and packaging techniques have been discussed with personnel of a number of different companies and governmental agencies in the course of this study. The following list indicates the companies and governmental agencies with whom components and packaging techniques have been discussed, and the topics discussed with each:

Motorola Semiconductor Div. Phoenix, Arizona	Integrated circuit sense amplifiers Integrated circuit storage registers Monolithic integrated circuits Hybrid integrated circuits
Remington Rand UNIVAC St. Paul, Minnesota	Hybrid integrated circuits Integrated circuit reliability & failure analysis Packaging techniques
Control Data Corp.	Integrated circuit applications Packaging techniques
Autonetics Anaheim, California	Monolithic integrated circuits Integrated field-effect-transistor circuits Integrated circuit packaging techniques
Bunker-Ramo Corp. Canoga Park, California	Monolithic integrated circuits Integrated circuit sense amplifiers
RCA Laboratories Princeton, New Jersey	Integrated field-effect transistors Metal-oxide-semiconductor integrated circuits Active thin-film integrated circuits
Hughes Semiconductor Div. Newport Beach, California	Integrated circuit packaging techniques Monolithic integrated circuits Hybrid integrated circuits Active thin-film integrated circuit
Sylvania Waltham, Massachusetts	Tunnel diode circuits

ONR Washington, D. C.	Optical components
RADC Rome, New York	Optical components
SRI Menlo Park, California	All-magnetic logic Cellular logic for integrated circuits Electron beam fabrication Fluid logic
Fairchild Semiconductor Mountainview, California	Monolithic integrated circuits Hybrid integrated circuits Interconnection and packaging techniques
Optics Technology Belmont, California	Fiber optics Laser techniques
NASA Washington, D. C.	Active thin-film integrated circuits Monolithic integrated circuits Packaging techniques
National Security Agency Fort Mead, Virginia	Optical techniques Integrated circuits
Hughes Ground Systems Fullerton, California	Environmental stabilization of electronic equipment
Raytheon SC Division Mountainview, California	Monolithic integrated circuits Packaging techniques MOS circuits
Signetics Corp. Mountainview, California	Monolithic integrated circuits Field effect devices Packaging techniques
Honeywell Electronics DP Div. Wellesley Hills, Mass.	Logic circuits
Burroughs Paoli, California	Thin-film circuits
Navy BuShips	Packaging and maintainability techniques Integrated circuits
USAER&DL Fort Monmouth, New Jersey	Lasers High-speed serial circuits
Texas Instruments Dallas, Texas	Multi-circuit chip fabrication Monolithic integrated circuits

Remington Rand UNIVAC Blue Bell, Pennsylvania	Fluid logic (pneumatic) Interconnection techniques

Discussions with personnel of these organizations provided a basis for much of the information presented in this report. In addition to discussing techniques and approaches that have not been adequately described in published literature, the opinions of experts in specific areas in these organizations were solicited concerning the advantages, disadvantages, limitations and future prospects for different component and packaging techniques.

10.7.2 Literature

An extensive list of references pertinent to the study of components and packaging techniques is given in the Bibliography. A study of many of these references has contributed to the material presented in this report. Some of the more pertinent and important of these referred to in the text have been extracted from the bibliography and listed in this section as specific numbered references. These numbers correspond to numbered citations in the text. Direct quotations have been used where noted.

COMPONENT AND PACKAGING TECHNOLOGY REFERENCES:

1. "Microelectronics - Which Approach and When", Webster, W. M., 1964 WESCON, Los Angeles, California, August 25, 1964.

2. "Choosing Logic for Microelectronics", Skoures, A. E., Electronics, Vol. 36, No. 47, pp 23-26, October 4, 1963.

3. "Trends in Logic Circuit Design", Lambert, A., Electronics, pp 38-45, December 6, 1963.

4. "Choice of Logic Forms for Integrated Circuits", Phelps, M., Jr., Electrical Design News, Cahners Publishing Co., January 1964.

5. Mildata Study, Quarterly Progress Report #1, August 12, 1963 to November 8, 1963, DA-63, 039-AMC-03275 (E), Honeywell Electronic Data Processing, 3 December 1962.

6. "The Case for Cryogenics?", Ittner, W. V., Proceedings 1962 FJCC, pp 229-231, Philadelphia, Pa., December 1962.

7. "Pneumatic Log" I-IV, Holbrook, E. L., Control Engineering, July, August, November 1961, and February 1962.

8. "Fiber Optics and the Laser", Kapany, N. S., paper presented at the New York Academy of Sciences Conference on the Laser, New York, New York, May 4-5, 1964.

9. "The Status of Optical Logic Elements for Nanosecond Computer Systems", Tippett, J. T., 1963 Pacific Computer Conference, IEEE, Pasadena, Calif., pp 47-53, March 15-16, 1963.

10. "Possible Uses of Lasers in Optical Logic Functions", Koster, C., 1963 Pacific Computer Conference, IEEE, Pasadena, Calif., pp 54-62, March 15-16, 1963.

11. "A Survey of Tunnel-Diode Digital Techniques", Sims, R. C., Beck, E. R., Jr., and Kamm, V. C., Proceedings of the IRE, Vol. 49, No. 1, pp 136-146, January, 1961.

12. "300 mcs Tunnel Diode Logic Circuits", Cooperman, M., 1963 Pacific Computer Conference, IEEE, Pasadena, Calif., pp 166-186, March 15-16, 1963.

13. "Design of an All Magnetic Computing System", Crane, H. D. and Van DeRiet, E. K., IRE Transactions on Electronic Computers, Vol. EC-10, No. 2, pp 207-232, June 1961.

14. "The Case for Magnetic Logic", Rogers, J., and Kings, J., Electronics, Vol. 37, No. 17, pp 40-47, June 1, 1964.

15. "All Magnetic Digital Circuit Fundamentals", Newhall, E. E., Digest of 1964 International Solid State Circuits Conference, pp 16-17, Philadelphia, Pa., February 1964.

16. "All Magnetic Digital Circuits and Application Problems", Baker, T., and Dillon, C., Digest of 1964 International Solid State Circuits Conference, pp 18-19, Philadelphia Pa., February 1964.

17. "Study of Applications of All-Magnetic Logic to Naval Systems", Adams, M. B., and Van DeRiet, I. E., Final Report on ONR Contract No. N-onr-2332(oo), Stanford Research Institute, November, 1963.

18. "Microelectronics - Military Participation and Objectives", Alberts, R., 1964 WESCON, Los Angeles, Calif., August 25, 1964.

19. "Big Computer Goes in Small Package", Electronics, pp 28-29, March 14, 1964.

20. "Solid Logic Technology: Versatile, High-Performance Microelectronics", Davis, E. M., Harding, W. E., Swartz, R. S., Korning, J. J., IBM Journal of Research & Development, Vol. 8, No. 2 pp 102-114.

21. "Digital Computer Aspects of Integrated Circuit Applications", Platzek, R. C., and Goodman, H. C., Proceedings Nat'l. Winter Convention on Military Electronics, Los Angeles, Vol. III, pp 2-34 - 2-53, February 5-7, 1964.

22. "Microelectronics - Where, Shy, and When", O'Connell, E. P. and Brauer, J. S., Proceedings Nat'l. Winter Convention on Military Electronics, Los Angeles, Calif., Vol. III, pp 2-1, Feb. 5-7, 1964.

23. "1964: The Year Micro Circuits Grew Up", Electronics, pp 10-11, March 14, 1964.

24. "The Economic Impact of Integrated Circuitry", Haggarty, P. E., IEEE Spectrum, Vol. 1, No. 6, pp 80-82, June 1964.

25. "Film Technology", McLean, David, Verbal Presentation at the IEEE Integrated Circuit Workshop, Monterey, Calif., September 17-18, 1964.

26. "Monolithic Integrated Circuits, Philips, A. B., IEEE Spectrum, Vol. 1, No. 6, pp A-3 - 101, June 1964.

27. "The Minimization of Parasitics in Integrated Circuits by Dielectric Isolation", Maxwell, D. A., Beeson, R. H., and Allison, D. F., 1964 WESCON, Los Angeles, Calif., August 25, 1964.

28. "Integrated Linear Circuits", Bailey, D., Electronic Products, pp 50, June, 1964.

29. "Integrated High-Frequency D.C. Amplifiers", Breuer, D. R., TRN Space Technology Laboratories Report No. 9374.1 - 0001, Redondo Beach, Calif., June 10, 1964.

30. "Types of Integrated Circuits", Hogan, C. L., IEEE Spectrum, Vol. 1, No. 6, pp 63-71, June 1964.

31. "Utilization of New Techniques and Devices in Integrated Circuits", Second Quarterly Report, AF Contract No. AF 33(657) - 11185 (Pacific Semiconductor Inc.) 1 August 1963 - 31 August 1963.

32. "The Future of Thin-Flim Active Devices", Feldman, Charles, Electronics, Vol. 37, No. 4, pp 23-26, January 24, 1964.

33. "Thin-Film Circuit Technology: Part III-Active Thin-Film Devices", Fowler, A. B., IEEE Spectrum, Vol. 1, No. 6, pp 102-111, June 1964.

34. "Integrated Thin-Film Networks", Western Electronic News, pp 22-24, November 1964.

35. "Metal-Oxide-Semiconductor Field-Effect Transistors", Heiman, F. P., and Hofstein, S. R., Electronics, Vol. 37, No. 30, pp 50-61, November 30, 1964.

36. "MOS and Thin-Film Transistor Circuits", Hertzog, G., Verbal Presentation at the IEEE Integrated Circuit Workshop, Monterey, Calif., September 17-18, 1964.

37. "Integrated-MOST Memory", Schmidt, J. D., Verbal Presentation at the IEEE Computer Memory Workshop, Lake Arrowhead, Calif., September 10-11, 1964.

38. "Integrated Circuit Limitations", Moore, G., Verbal Presentation at the IEEE Integrated Circuit Workshop, Monterey, Calif., September 17-18, 1964.

39. "Integrated Circuits", Noyce, R. N., IEEE Student Journal, Vol. 2, No. 9, pp 31-34, September 1964.

40. "DONUT: A Threshold Gate Computer", Coates, C. L., and Lewis, P. M., IEEE Transactions on Electronic Computers, Vol. EC-13, No. 3, pp 240-247, June 1964.

41. "How to Achieve MAJORITY and THRESHOLD LOGIC with Semiconductors", Sauer, W. A., Electronics, Vol. 36, No. 48, pp 23-25, November 29, 1963.

42. "Improvement of Electronic-Computer Reliability Through Use of Redundancy", Brown,W. G., Tierney, J., and Wasserman, R., IRE Transactions on Electronic Computers, Vol. EC-10, pp 407-415, September 1961.

43. "Majority Voting Protects Aircraft and Pilot", Moreines, H., Worthington, R., and Thomas, F., Electronics, Vol. 37, No. 16, pp 85-91, May 18, 1964.

44. "Redundant Circuit Design: Payoffs and Penalties", Suran, J. J., EEE, Vol. 12, No. 9, pp 51-55, September 1964.

45. "Information Redundancy and Adaptive Structures", Angell, J. B., Digest of Technical Papers, pp 84-85, 1964 Solid State Circuits Conference, Philadelphia, Pa., February 19, 20, 21, 1964.

46. "The Use of Passive Redundancy in Electronic Systems", Suran, J. J., IRE Transactions on Military Electronics, Vol. MIL-5, pp 202-208, July 1961.

47. "Redundancy Techniques for Computing Systems", Wilcox, R. H., and Mann, W. C., Spartan Books, Washington, D. C., 1962.

48. "Use of Circuit Redundancy to Increase System Reliability", Suran, J. J., Digest of Technical Papers, pp 82-83, 1964 Solid State Circuits Conference, Philadelphia, Pa., February 19, 20, 21, 1964.

49. "Digital-Circuit Reliability through REDUNDANCY", Sorensen, A. A., Electro-Technology, pp 118-125, July 1961.

50. "Realization of the Reliability Potential for Microelectronics", Gould, E. B., III, and Wiley, C. A., Proceedings National Winter Convention on Military Electronics, pp 2-61-67, February 5-7, 1964, Los Angeles, California.

51. "Reliability", Bird, Carl, Informal Presentation at IEEE Integrated Circuit Workshop, Monterey, California, September 17, 18, 1964.

52. "Go-it-alone Circuits Advocated for Apollo Spaceship's Computer," Electronics, Vol. 37, No. 31, pp 92-97, Dec. 14, 1964.

53. "Failure Modes in Integrated and Partially Integrated Microelectronic Circuits," Anderson, G. P. and Erickson, R. A., Proceedings of the Second Annual Symposium on the Physics of Failure in Electronics, Sept. 25-26, 1963.

54. "Failure Modes in Thin Film Circuits," Smith, P., Genser, M., and Serrette, R., 1964 Components Conference, Washington, D. C., May 5, 1964.

55. "Physics of Failure in Electronics," Electromechanical Design, pp 28-33, November 1964.

56. "Progress Report on Attainable Reliability of Integrated Circuits for Systems Application," Partridge, J., Hanley, D., and Hall, E., ONR Symposium on Microelectronics and Large Systems, Washington, D. C., November 17-18, 1964.

57. "Microavionics: Important Considerations for Aerospace System Design," Keonjian, E., and Danner, F., Proceedings of the 1964 National Convention on Military Electronics, pp 945-949, September 1964.

58. "Integrated Circuit Technology," Narud, J., verbal presentation at the IEEE Integrated Circuit Workshop, Monterey, California, Sept. 17-18, 1964.

59. "Digital Computer Aspects of Integrated Circuit Applications," Platzek, R. C., and Goodman, H. C., Proceedings of the 1964 Winter Convention on Military Electronics, pp 2.34-2.35, Los Angeles, Calif., Feb. 5-7, 1964.

60. Interim Technical Documentary Progress Report No. ASD-TDR-7-998-3, Arinc Research Corp., Jan. 1 - April 1, 1963, prepared under Contract AF 33(657)-8785, Supplement 1.

61. "Government Needs and Policies in the Age of Microelectronics," J. M. Bridges, The Impact of Microelectronics, pp. 31-40, McGraw Hill Publishing Co., New York, New York, 1963.

62. "Integrated Circuits in Military Equipment," R. N. Noyce, IEEE Spectrum, Vol. 1, No. 6, pp 71-72, June 1964.

63. "Promised IC Benefits HERE," Mathews, W., Electronic News, Nov. 30, 1964.

64. "Lilliputian Circuits," Gall, P., Wall Street Journal, April 1, 1964.

65. "A Breakthrough in Cost Effectiveness Through Micro-electronics," Lowell, A.C., National Electronics Conference, Chicago, Ill., Oct. 20, 1964.

66. "Silicon FEB Techniques," Kilby, J. S., Solid State Design, Vol. 5, No. 7, pp 32-37, July 1964.

67. "Space Radiation Affects MOS FET's," Hughes, H. L., and Giroux, R. R., Electronics, Vol. 37, No. 32, pp 58-60, Dec. 28, 1964.

68. "System Speed with Integrated Circuits," Powers, G., Electronic Design News, pp 20-27, May 1964.

69. "Sophisticated Thin-Film Computer Holds Aerospace System Potential," LaFond, Charles D., Missiles and Rockets, Vol. 15, No. 16, pp 33-35, October 19, 1964.

70. "Flip Chips Easier to Connect," Carr, E. G., Electronics, pp 82-84, October 18, 1963.

71. "Production-Line Packaging of Solid-State Circuits," Garibotti, D. J., and Ullery, Lee R., Jr., Electronics, Vol. 37, No. 24, pp 73-79, Sept. 7, 1964.

72. "Interconnection and Organization of Functional Electronic Blocks," Winsker, H., and MacIntyre, R., Paper presented at the 16th Annual National Aerospace Electronics Conference, May, 1964.

73. "Interconnection of Integrated Circuit Flat Packs in Autonetics Improved Minuteman Program," Harmon, Elise F., North American Aviation Autonetics, Technical Publication T4-358/33.

74. "Advances in Electronic Circuit Packaging," Vol. 3, Proceedings of the Third International Electronic Circuit Packaging Symposium, University of Colorado, Plenum Press, New York, 1963.

75. "Interconnection and Organization of Functional Electronic Blocks," United Aircraft Corporation, Norden Division, Norwalk, Connecticut, Contract No. AF 33(657)-8720, Technical Documentary Report No. RTD-TDR-63-4042, October, 1963.

76. "UXL -- New Technique for Interconnections," Western Electronic News, pp 24-28, December, 1964.

77. "Modern Electronics Packaging," McKenzie, Alexander, A., Electronics, pp 34-48, February 7, 1964.

78. Micro Electronics, Keonjian, E., McGraw Hill Publishing Co., New York, New York, 1963.

79. "Laser Welding for Microelectronic Interconnections," Rischall, H., and Shackleton, J., 1964 Electronic Components Conference, Washington, D. C.

80. "Cellular Linear - Input Logic," Minnick, R. C., and Short, R. A., Final Report on AF19(628)-448, Project 4641, Tank 4641C1, Stanford Research Institute, February 1964.

81. "Evolution of the Concept of a Computer on a Slice," Sack, E. A., Lyman, R. C., and Chang, G. Y., Proceedings of the IEEE, Vol. 52, No. 12, pp 1713-1720, December 1964.

10.8 SECTION 6 MEMORY TECHNOLOGY SOURCE MATERIAL

Memories have been discussed with personnel of a number of different companies and governmental agencies in the course of this study. The following list indicates the companies and governmental agencies with whom memories have been discussed and the type of memories discussed with each:

Rome Air Development Center Rome, New York	Cryogenic Memories, Random Access Memories, Non-Rotating Mass Memories, Cryogenic Associative Memories, Magnetic Associative Memories Plated Wire Memories Permolloy Sheet Memories Associative Memories Associative Processors
Control Data Corporation St. Paul, Minnesota	Magnetic Thin Film Memories
Remington Rand, Univac	Coincident Current Magnetic Core Matrix Memories, Word Organized Magnetic Core Matrix Memories, Magnetic Thin Film Memories, Associative Memories
Bunker-Ramo Corporation Canoga Park, California	Woven Screen Memories, Biax Associative Memories, Thin Film Plated Wire Assoc. Mems. Transfluxor Associative Mems.
Fabritek Emery, Wisconsin	Coincident-Current Mag. Core Matrix Memories, Magnetic Thin-Film Memories
N C R Hawthorne, California	Plated Magnetic Rod Memories Photochromic Memories
Sylvania Waltham, Massachusetts	Permanent "unit-record" magnetic card memories
Philco Research Lab. Newport Beach, California	Magnetic Thin Film Memories, Biax read-mostly memory, Biax associative memory

RCA Laboratories Princeton, New Jersey	Cryogenic Random-Access Mems. Cryogenic Associative Mems. Laminated Ferrite Memories
Laboratory for Electronics	Permalloy Sheet Random-Access Mems., Bernoulli-Disc Rotating Memories, Bernoulli-Disc Mass Memories
Autonetics, Anaheim, California	Memory Systems and Hierarchies
U.S. Army Engineering Research and Development Lab., Fort Monmouth, N. J.	Block Oriented Random-Access Mems., (static ferro-acoustic storage), Memory Systems and Hierarchies, Magneto-optical and Electro-optical Memories for Displays, Glass delay-line Memories Plated Wire Memories Ferro-Acoustic Memories
Office of Naval Research Washington, D. C.	Magnetic Core Memories, Read only Memories, Cryogenic Memories, Associative Memories, Optical Memories
National Security Agency Fort Mead, Maryland	Thin-Film Memories Cryogenic Memories
U.S. Navy Bureau of Ships Washington, D. C.	Associative Memories, Magnetic Disc Files, Magnetic Card Memories, Mass Memories, Militarized Paper Tape Equipment
Hughes Ground Systems Div. Fullerton, California	Associative Memories, Magnetic Domain Wall Shift Registers
IBM Federal Systems Div. Rockville, Maryland	IBM Divisions Militarized Disc File
MIT Lincoln Labs. Lexington, Mass.	High-density, large-capacity Magnetic Thin-Film Memories
Burroughs Paoli, Pennsylvania	Thin-Film Memory Planes
Librascope Glendale, California	TOKO Plated Wire Memory

Texas Instruments Dallas, Texas	Thin-Film Memories
Remington Rand Univac Blue Bell, Pennsylvania	Plated Wire Memories, Planar Thin-Film Memory Electromechanical Mass Memories
Stanford Research Institute Menlo Park, California	Thin-Film Techniques
Fairchild Semiconductor Palo Alto, California	Integrated Circuit Memories

10.9 SECTION 7 MACHINE SYSTEMS ORGANIZATION REFERENCES

1. "Why Multicomputers?," W. F. Bauer, Datamation, September 1962.

2. "Highly Parallel Machines," W. B. Comfort, Datamation, September 1962.

3. "The RW-400 - A New Polymorphic Data System," R. E. Porter, Datamation, January/February 1960.

4. "The Burroughs D825," J. P. Anderson, Datamation, April 1964.

5. "The Design of the B5000 System," W. Longergan and P. King, Datamation, May 1964.

6. "Planning the 3600," C. T. Casale, Proceedings FJCC, 1962.

7. "PMR Real-Time Data Handling System, System Description and Programming Concept," Informatics, Inc., Unpublished Report to Headquarters, PMR, 1963.

8. "A Multiple Satellite Real-Time Control Network," V. White, IEEE Transactions on Military Electronics, October 1963.

9. "Generalized Multiprocessing and Multiprogramming," A. J. Critchlow, Proceedings, FJCC, 1963.

10. "Multi-Level Programming for a Real-Time System," A.B. Shafritz, A. E. Miller, and K. Rose, Proceedings EJCC, 1966.

11. "Computer Design from the Programmers' Viewpoint," W. F. Bauer, Proceedings, EJCC, 1958.

12. "An Automatic Sequencing Procedure with Applications to Parallel Programming," E. S. Swartz, Journal of ACM, Vol. 8, p. 513.

13. "New Concepts in Computing System Design," G. M. Amdahl, Proceedings of the IRE, May 1962.

14. "Analysis of Computing Load Assignment in a Multi-Processor Computer," M. Aoki, G. Estrin and R. Mandell, Proceedings, FJCC, 1963.

15. "Multi-computer Programming for a Large-Scale Real-Time Data Processing System," G. E. Pickering, E. G. Mutschler and G. A. Erickson, Proceedings, SFCC, 1964.

16. "Design for an Associative Computer," P. M. Davies, Proc. Pacific Computer Conference, Pasadena, California, March 1963.

17. "A Cryogenic Data Addressed Memory," V. L. Newhouse and R. E. Fruin, Proc. Spring Joint Computer Conf., May 1962, p. 89.

18. "A Magnetic Associative Memory," J. R. Kiseda, H. E. Seelbach, W. C. and M. Teig, IBM J. Res. and Dev., April 1961, pp. 106-121.

19. "Algorithms for Content Addressable Memories," G. Estrin and R. H. Fuller, 1963 Pacific Computer Conference, IEEE.

20. "Content - Addressable Memory System Report" No. 63-25, Nonr 233(52), R. H. Fuller, Department of Engineering, University of California, Los Angeles, California.

21. "An Associative Processor," R. E. Ewing and P. M. Davies, Proc., FJCC, November 1964, pp. 147-159.

22. "An Organization of an Associative Cryogenic Computer," R. F. Rosin, Proc. AFIPS SJCC, May 1962, pp. 203-212.

23. "Organization of Computer Systems - The Fixed Plus Variable Structure Computer," G. Estrin, Proc. WJCC, San Francisco, California, March 1960, pp. 33-37.

24. "Associative Logic for Highly Parallel Systems," R. R. Seeber and A. B. Lindquist, Proc. FJCC, November 1963, pp. 489-493.

25. "A Search Memory Subsystem for a General Purpose Computer," A. Kaplan, Proc. FJCC, November 1963, pp. 193-200.

26. "A Hardware-Integrated GPC/Search Memory," R. G. Gall, Proc. FJCC, November 1964, pp. 159-173.

27. "Associative Memory with Ordered Retrieval," R. R. Seeber and A. B. Lindquist, IBM J. Res. and Dev., January 1962.

28. "Associative Self-Sorting Memory," R. R. Seeber, Jr., Proc. EJCC, December 1960, pp. 179-188.

29. "The Logic Theory Machine," A. Newall and H. A. Simon, IRE Trans. on Information Theory, Vol. I-T-2, 1956, No. 3, pp. 61-79.

30. "A Machine for a General Purpose List Processor," R. L. Wigington, Trans. IEEE on Electronic Computers, Vol. EC-12, Dec. 1963, p. 707-714.

31. "The Best Way to Design an Automatic Calculating Machine," N. V. Wilkes, Manchester University Computer Inaugural Conference, Proceeding, July 1951.

32. "Micro-programming and the Design of the Control Circuits in an Electronic Digital Computer," M. V. Wilkes and J. B. Stringer, Proceedings of the Cambridge Philosophical Society, April 1953.

33. "Microprogramming, M. V. Wilkes, Proc. EJC, Dec. 3-5, 1958, pp. 18-20.

34. "A Note on Microprogramming", H. T. Giantz, Journal of the Association for Computing Machinery, April 1956.

35. "Micro-Programming", Robert J. Mercer, Journal of the Association for Computing Machinery, April 1957.

36. "Microprogramming and Stored Logic," Lowell Amdahl, Datamation, Feb. 1964, pp. 24-26.

37. "The TRW-133 Computer", W. C. McGee, Datamation, Feb. 1964, pp. 27-29.

38. "The PB-400 Computer," E. O. Boutwell, Datamation, Feb. 1964, pp. 30-32.

39. "The C-8401," L. Beck and F. Keeler, Datamation, Feb. 1964, pp. 33-35.

40. "Stored Logic Programming and Applications," Richard H. Hill, Datamation, Feb. 1964, pp. 36-39.

41. "CIRRUS, An Economical Multiprogram Computer with Microprogram Control," M. W. Allen, T. Pearcey, J. P. Penny, G. A. Rose, and T. G. Sanderson, IEEE Transactions, Dec. 1963.

42. "A Memory Organization for an Elementary List-Processing Computer," V. O. Muth and A. K. Scidmore, IEEE Transactions on Electronic Computers, June 1963.

43. "Microprogrammed Control for Computing Systems," G. B. Gerace, IEEE Transactions on Electronic Computers, Dec. 1963.

44. Symposium on Advanced Computer Organization, AFIPS Computer Conference, Munich, 1962.

45. "The Design of a General-Purpose Microprogram-Controlled Computer with Elementary Structure," Thomas W. Kampe, IRE Transactions on Electronic Computers, June, 1960.

46. "The KT Pilot Computer - A Microprogrammed Computer with a Phototransistor Fixed Memory," H. Hagiwara, K. Amo, S. Matsushita and H. Yomanchi.

47. "The Atlas Computer," C. H. Devonald, and J. A. Fotheringham, Datamation, May 1961.

48. "IBM System/360 Engineering," P. Fagg, J. L. Brown, J. A. Hipp, D. T. Doody, J. W. Fairclough, and J. Greene, AFIPS Proceedings FJCC, 1964.

49. "The Central Control Unit of the Atlas Computer," F. H. Sumner, G. Haley, and E. C. V. Chen, IFIP Congress, Munich, 1962, p. 657.

50. "Organization of computer systems; the fixed plus variable structure computer," G. Estrin, Proc. Western Joint Computer Conf., 1960, pp. 33-40.

51. "Automatic Assignment of Computations in a Variable Structure Computer," G. Estrin and R. Turn, IEEE Transactions Vol. EC-12, No. 6, Dec. 1963.

52. "Parallel Processing in a Restructural Computer System," G. Estrin, B. Bussel, R. Turn, J. Bibb, IEEE Transaction, Vol. EC-12, Dec. 1963.

53. "Iterative Circuit Computers," J. R. Halland, Proceedings of the Western Joint Computer Conference, San Francisco, Calif. May 3-5, 1960.

54. "A Universal Computer Capable of Executing an Arbitrary Number of Sub-Programs Simultaneously," J. R. Holland, Proceedings of the Eastern Joint Computer Conference, Boston, Massachusetts, Dec. 1-3, 1960.

55. "A Multilayer Iterative Circuit Computer," Rodolfo Gonzales, IEEE Transactions on Electronic Computers, Dec. 1963.

56. "A Parallel Computer Organization and Mechanizations," J. K. Hawkins, C. J. Munsey, IEEE Transactions, June 1963.

57. "Pattern Recognition and Detection, S. H. Unger, Proceedings of the IRE, Vol. 47, October 1959, pp. 1737-1752.

58. "Design of a Pattern Recognition Digital Computer - Part 1: General Introduction," B. H. McCormick, Report No. 125, Digital Computer Laboratory, University of Illinois, Champaign Urbana, Ill.

59. "Parallel Computing with Vertical Data," W. Shooman, Proceedings of the Eastern Joint Computer Conference, New York, Dec. 13-15, 1960.

60. "The Solomon Computer," D. L. Slotnick, W. C. Borck and R. C. McReynolds, Proceedings of the Fall Joint Computer Conference, Philadelphia, December 1962.

61. "On the Use of the Solomon Parallel Processing Computer," J. R. Ball, R. C. Bollinger, T. A. Jeeves, R. C. McReynolds, D. H. Shaffer and D. I. Caplan, Proceedings of the Fall Joint Computer Conference, Philadelphia, December 1962.

62. "The Solomon Computer," J. Gregory and R. McReynolds, IEEE Transaction, Dec. 1963.

63. "Iterative Circuit Computers," Harvey L. Garner, J. S. Squire, Proceedings of the 1962 Workshop on Computer Organization.

64. "Intercommunicating Cells, Basis for a Distributed Logic Computer," C. Y. Lee, Proceedings of the Fall Joint Computer Conference, Philadelphia, December 1962.

10.10 SECTION 9 ADVANCED USAGE TECHNIQUES REFERENCES

1. "Systems and Information," W. Ross Ashby, IEEE Transactions, Volume MIL-7, April-July 1963.

2. "Reliability of Biological Systems," Warren S. McCullock.

3. "An Abstract Machine Based on Classical Association Psychology," Richard F. Reiss, Spring Joint Computer Conference 1962.

4. "Some Methods of Artificial Intelligence and Heuristic Programs," Marvin Minsky, Proceedings of the Symposium on Mechanization of Thought Process, National Physical Laboratory, Teddington, England, November 1958.

5. "Adaptive Mechanisms in Digital 'Concept' Processing," Manfred Kochen, Joint Automatic Control Conference of 1962.

6. "A Theory of Simple Concepts with Applications, " Janerji and Windeknecht.

7. "Can a Machine Think?," A.M. Turing, Mind,1950.

8. "Self-Organizing System," Marshall Yovits and Scott Cameron, Proceedings of an Interdisciplinary Conference, May 1959.

9. "Principles of Self-Organization," Von Foerster and Zopf, 1960 Conference on Self-Organizing Systems

10. "Unconventional Computer of the Future," Richard Wilcox, Naval Research Reviews, January 1964.

11. "Pattern Recognition and Reading by Machine," Bledsoe and Browning, Proceedings of the Eastern Joint Computer Conference, 1959.

12. "Adaption and Feedback," J. Sklansky, Joint Automatic Control Conference of 1962.

13. "Analytic Techniques for the Study of Neural Nets," F. Rosenblatt, Joint Automatic Control Conference of 1962.

14. "A Flexible Neural Logic Network," G. J. Dusheck, T. G. Hilinski, F. L. Putzrath, IEEE Transaction, Volume MIL-7, April-July 1963.

15. "A Model of the Plastic Neuron," V. V. Griffith, IEEE Transactions, Volume MIL-7, April-July 1963.

16. "Comments on Learning and Adaptive Machines for Pattern Recognition," C. Hugh Mays, 1964 Fall Joint Computer Conference

17. "Some Considerations of Polystable Systems," Howard S. Fitzhugh II, IEEE Transactions, Volume MIL-7, April-July 1963.

18. "Functional Electronic Model of the Frog Retina," M. B. Herscher and T. P. Kelley, IEEE Transactions, Volume MIL-7, April-July 1963.

19. "Large Artificial Nerve Net (Lannet)", David F. Guinn, IEEE Transactions, Volume MIL-7, April-July 1963.

20. "Programming Pattern Recognition," G. P. Dinneen, Proceedings of the Western Joint Computer Conference, Los Angeles, California, 1955.

21. "Generalization of Pattern Recognition in a Self-Organizing System," Proceedings of the 1955 Western Joint Computer

Appendix A
STORAGE ELEMENTS

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Appendix A
STORAGE ELEMENTS

1.0 INTEGRATED CIRCUIT MEMORIES

Integrated semiconductor circuits are discussed in detail in Components and Packaging (Appendix B), but their application as storage elements deserves additional consideration. The most obvious storage application of integrated circuit techniques is in registers where each bit position of the register is mechanized with a single integrated circuit flip-flop. Registers mechanized with individual integrated circuits will be available within the next year operating at cycle times in the order of 50 nanoseconds, i.e., the rate at which the flip-flop may be repetitively set, sensed, and reset. New developments in integrated circuits are leading to propogation delays in the order of 2 nanoseconds per gate, but the complete cycle time for a flip-flop is several times the propogation delay of an individual gate. By 1970, integrated flip-flops may be available with cycle times less than 20 nanoseconds. A small scratch-pad memory mechanized with elements of this type is somewhat slower overall, of course, since some time is required for addressing and selection.

Integrated circuit flip-flops are expected to be the primary method of mechanizing individual registers in new machines within the next few years. Prior to 1970, advances in the state of the art in integrated circuits will permit the fabrication of several stages of a register (perhaps the entire register) on a single silicon monolithic integrated circuit chip. Those applications in which transistorized flip-flops are now used will quickly give way to individual flip-flop integrated circuits, and to multi-stage integrated circuits before 1970.

In a somewhat different approach, a prototype memory has been developed in which a special integrated circuit storage element is used with XY matrix addressing and selection techniques. This element, selected by a coincidence of X and Y signals, is simpler than a conventional flip-flop (e.g., only a single output is required). The element was designed to operate at very low power levels (approximately 30 milliwatts per element) and for a cost per bit that is relatively low for high-speed scratch-pad type memories. The designers state, "Although the cost in $/bit/nanosecond for integrated circuit memory is higher when compared with magnetic core memory, it is lower than magnetic thin film memories having a cycle time of 300 nanoseconds. Also, a 100 to 200 nanosecond memory system having a size of say 256 words by 24 bits, can be readily produced in quantity in 1964."[1] At present, individual chips and packages are used for each bit of storage.

The next step beyond the use of multi-stage integrated circuit chips is the fabrication of several hundred bits, or more, of semiconductor storage in a single silicon chip using XY matrix addressing and selection techniques. This type of active circuit storage, using bipolar transistor techniques, is anticipated for small high-speed control memory applications by 1970. At present, however, the most advanced work in this direction involves the use of metal-oxide semiconductor field-effect type elements. Several companies are actively working on this type of storage array for control and scratch-pad memories.

One semiconductor manufacturer has reported work on a small low-cost random-access memory array mechanized with MOS elements which is significant in that a single chip contains an order of magnitude more components than other integrated circuits now being manufactured. The efforts to date have produced a complete memory array of 16 four-bit words on a single silicon chip 115 mils by 145 mils in size (approximately ten times the area of a conventional integrated circuit). About 30 chips can be fabricated on one silicon wafer with present yields of 4 to 5 good chips per wafer. The total read/write cycle time for this device is 4 microseconds -- a 2 microsecond read time and a 2 microsecond write time. The relatively slow speed of 4 microseconds results from the fact that the MOS is a high impedance device. The rise time is approximately 75 nanoseconds and the fall time approximately 575 nanoseconds. Although this type device is slow compared to integrated circuit flip-flops using bipolar transistor techniques, it is of great interest because the MOS devices are much easier to fabricate than epitaxial integrated circuits. Since significantly fewer processing steps are involved, the fabrication of large arrays of elements with reasonable yields is more feasible. A speed improvement of an order of magnitude is probably the best that can be expected in this type storage.

As integrated circuit technology and yields improve, workers in the field anticipate the fabrication of somewhat similar arrays of storage elements permitting significantly higher speeds through the use of planar epitaxial transistor elements rather than MOS elements. Mr. Rex Rice, of Fairchild Semiconductor, has predicted that large integrated circuit arrays using bipolar transistor techniques will eventually replace magnetic core memories in applications where relatively high speed (e.g., 0.5 to 1 microsecond) is required and where a volatile type memory is acceptable.[3] He also believes that volatile storage is acceptable for more internal memory applications than is generally recognized. This prediction implies that high-speed integrated circuit storage arrays of this type can be built at a cost that will be more than competitive with magnetic core memories, and that power requirements per element can be reduced significantly. The power required per stage is a serious consideration for any active storage array since holding power is required for each bit position in the array.

2.0 TUNNEL DIODE MEMORIES

Storage elements using the negative resistance characteristic of tunnel diodes have been investigated extensively for high-speed scratch-pad type memories with cycle times of 10 to 100 nanoseconds and capacities of 128 to 512 words. A few years ago, it appeared that this might be the only feasible method of achieving these characteristics; however, other techniques such as epitaxial integrated circuit arrays and small magnetic thin-film memories offer promise of providing these characteristics (except possibly for the extreme fast end of the speed range) at a significantly lower cost. Since no requirement is anticipated for such extreme speeds in a 1970 ACDS, tunnel diode storage is not a major competitor for use in such a system. Hence, these techniques have not been investigated in great detail since other storage techniques can meet the future requirements for a shipboard data system at significantly lower costs. Even optimistic projections imply future costs for high-speed tunnel diode memories of no less than 50 cents per bit.

3.0 PLANAR MAGNETIC THIN FILM MEMORIES

Planar magnetic thin film memories represent one of the more promising approaches to batch fabrication of storage arrays. At present, they are used primarily in small high-speed control and scratch-pad memories. Future applications as very large low-cost memories are anticipated by 1970.

Most previous thin-film developments used an array of discrete elements fabricated by vacuum deposition processes on a substrate such as glass or aluminum.[4,5] Each element is about 1,000 angstroms thick and is approximately 25 x 50 mils in area. Some more recent developments use either a continuous sheet[6] (see Figure 1), or narrow strips[7,8] of thin magnetic film vacuum deposited on the substrate. The X and Y drive lines are either vacuum deposited on the same substrate with appropriate insulation between them or, in some cases, the drive lines and sense lines are fabricated on a separate substrate which is then mechanically superimposed over the one containing the magnetic elements. Registration problems are less severe for the strip and the continuous sheet arrays.

Anisotropic material is used so that there is a preferred direction (or easy axis) of magnetization. In a conventional planar thin-film memory, the word line runs parallel to the easy axis of magnetization (creates a flux pattern in the hard direction) and the digit and sense wires run parallel to the hard axis of magnetization (creates a flux pattern in the easy direction). A word wire then rotates all bit positions of a word in the hard direction, but they tend to flip back when released. A relatively small digit signal pushes them into the one state or back to the zero state as desired. Since the flux path is not closed, the sense signal is small due to the flux being only rotated rather than switched. This results in a smaller sense signal and a tendency to creep or demagnetize along the outer edges of the bit spot. Due to the latter effect, a large number of disturb signals may tend to destroy the information stored in the bit position. This creep problem is overcome in most cylindrical film (plated wire) memories since the magnetic material closes on itself, hence there is no tendency to creep at the edges. To alleviate the creep problem, the film is made thinner. The thicker the film the greater the tendency to demagnetization and creep. On the other hand, the thinner the film the less the sense signal, hence a compromise must be made between the desired sense signal and minimum creep in choosing the thickness of the film.

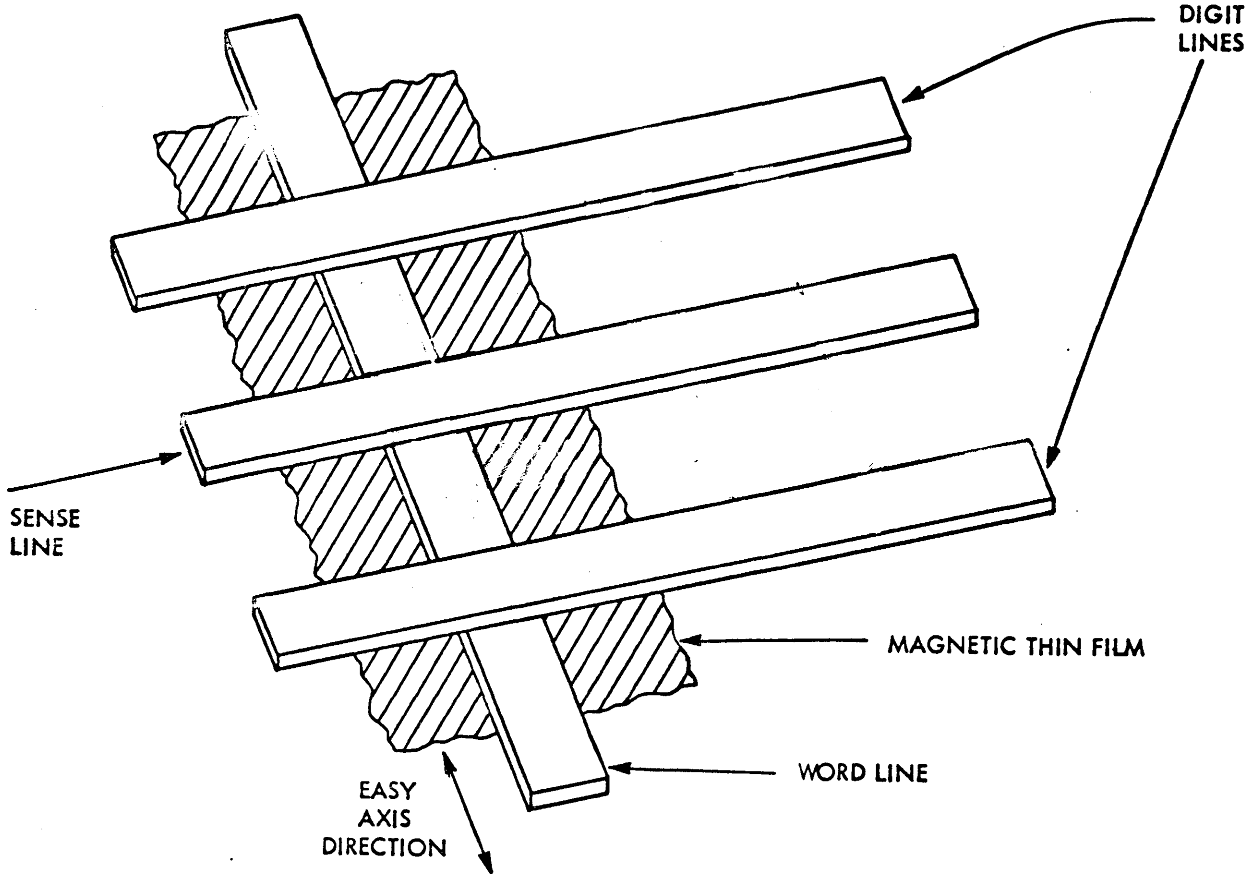


Figure 1(a). Typical Word, Sense, and Digit Line Geometry for Thin-Film Memory

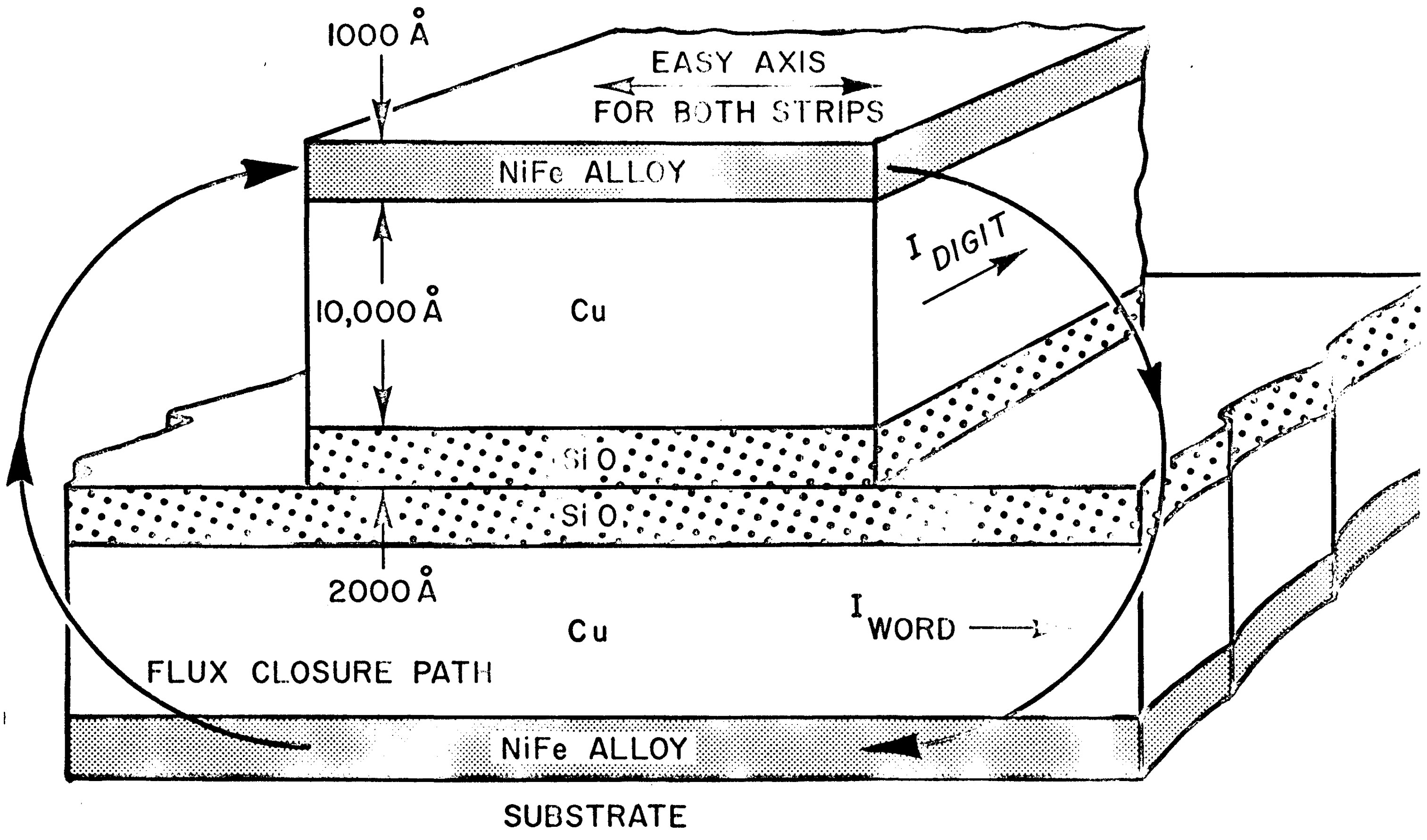


Figure 1(b). Cross-section of High Density Thin-Film Storage Element

A number of problems have plagued thin-film memories including:

1) Dispersion - The direction of easy axis is not uniform over the entire film (should be less than 5°).

2) Skew - The direction of the easy axis is not exactly as intended (should be less than 5°).

3) Low word sense signal - Since the flux path is not closed and the flux is not switched but only rotated, word sense signals are 0.1 to 0.2 millivolt.

4) High drive signal followed by low sense signal - It is necessary to balance out the high drive signal on the sense wire. The drive signal is approximately 10 volts compared to a 0.1 to 0.2 millivolt sense signal.

5) Magnetostriction - A material of approximately 19.6% iron and 80.4% nickel is required for zero magnetostriction. If the iron content is slightly too high, a negative stress will deteriorate the rectangularity of the hysteresis loop; if the nickel content is too high, a positive stress signal will decrease the rectangularity of the hysteresis loop.

6) Yield - Perhaps 30% to 50% yield can be obtained in small planes (e.g. 128 x 24 bits) but it is not economically sound to build a large memory out of a multitude of small planes because of the interconnection and mounting problem. To make an economical large array, it is necessary to fabricate a large plane with a large number of bit positions, but the yield problem is much worse in this case. A 50% yield on small planes might imply an almost negligible yield on very large planes if 100% good spots are required. Therefore, for large planes, it may be necessary to develop some redundancy techniques so that the system can operate with a limited number of bad spots. Because of these problems, this type of memory is usually operated in a word-oriented fashion rather than a coincident current mode. This increases the cost of the associated electronics. On the other hand, the use of domain

wall rotation and the absence of a closed flux path permit higher speed operation in this type of memory than in a magnetic core matrix memory.

Use of glass substrates for planar thin-film memories requires higher drive power, but problems of flux trapping in the substrate are avoided. Use of metal substrates causes current spreading and hence provides poorer operating margins, but more uniform films can be obtained and closer packing densities and lower impedances are permitted.

Many companies are working on magnetic thin-film memories. It has been predicted that within the next few years they will replace magnetic core matrix memories for high-speed applications (less than 1 microsecond cycle time) and for very large capacity applications, but it is questionable whether they will replace coincident current magnetic core matrix memories for slower speed applications (cycle times of 2 microseconds or greater) requiring less than 10^6 bits.

The major application area in which planar thin-film memories are currently being used is that of small high-speed control memories. The next most likely application area is that of very large-capacity low-cost memories, but a memory of this type is approximately five years away. The use of thin-films in the intermediate area between a few hundred words and perhaps 10^6 bits is more questionable. In the small memory the array cost is negligible, the electronics predominate, and the main advantage is the speed that can be achieved (in the order of 100 ns or so) that cannot be matched by cores. In the large memory of over 10^6 bits, the advantage is in the low array cost when the array cost begins to exceed the electronics cost. However, in the intermediate area, the speed of core memories will drop to 1/2 microsecond, and they may be cheaper than equivalent thin-film memories for a long time. For larger capacity memories of less than 10^6 bits where the electronics are still a significant portion of the cost, the electronics of core memories are cheaper than those of thin-film memories. Magnetic core memories will continue to dominate the range from a few hundred words of microsecond memory to approximately 10^6 bit memories (the main range of internal memories) for some time, but the batch fabrication aspects of thin-film memories may alter this balance by 1970.

Thin-film elements can now be rotated in approximately one nanosecond, but the associated electronics cause selection to take at least 35 nanoseconds. Thin-film memories can provide non-destructive read out, require less power than core memories, and are batch fabricated. However, the low sense signal level requires a good differential amplifier; fast rise time circuits are required, and it is difficult to adequately control the magnetic variables to produce uniform elements. Memories of 4000 words capacity with 100 nanosecond cycle times, and 64,000 word memories with 500 nanosecond cycle times appear reasonable for the future.

A 400 nanosecond cycle time thin-film memory being fabricated for the Johnsville Naval Air Station is typical of current work in thin-film memories. This memory is word-organized, contains 1024 words of 50 bits each, and operates in a 400 nanosecond cycle time. The array is composed of film deposits approximately 900 angstroms thick that are rectangular in shape, measuring 25 x 50 mil. A single film is used for each bit. Manufacturing costs of current film stacks are between 20 cents and 50 cents a bit,and development of a film plane within the next year costing approximately 5 cents a bit is expected. These costs are for the fabricated planes only and do not include electronics.

Both X and Y axis lines are superimposed on the same printed circuit card with a plastic deposit separating them. The digit lines are approximately 32 mils wide, each consisting of two 10-mil lines separated by a 12 mil gap. The two 10-mil lines are joined at both ends of the board and are terminated in a plated-through hole for interconnection. The drive line is approximately 23 mils wide, each consists of two 9-mil lines divided by a 5-mil gap. The drive lines are also reconnected at both ends in plated holes, which are fed through the opposite side of the adjoining board, and fed back parallel to the first line to provide a proper termination, and to provide noise cancelling. Each memory plane consists of a total of 112 substrates which are placed on a single printed circuit board. This is done to reduce the number of interconnections involved on the board and to decrease the total number of film spots that are deposited on a single substrate, thus increasing the yield. The reverse printed circuit board is placed over the film and the total assembly is bolted together between two holding plates. The drive and sense lines are then terminated.

Higher performance thin-film memory work is represented by the memory being developed for the REDMAN program in which the memory planes are fabricated by depositing a continuous film of nickel cobalt on a polished aluminum substrate.[6] This memory has a capacity of 4096 words of 50 bits per word organized as 8192 25-bit words.

The density is 625 bits per square inch with a 80 mil spacing on the digit lines and 60 mil spacing on the word lines providing 3200 bits on a 3 by 3 inch substrate. The cycle time is 140 nanoseconds and the access time 100 nanoseconds. The first version of this memory using discrete semiconductor components for the electronics is scheduled for completion in June 1965. The next step will use all integrated circuits for the logic circuits and sense amplifiers. Integrated circuit word drivers (500 milliamps) and bit drivers ($\pm$ 150 milliamps) are approximately 18 months away.

This program is expected to lead to array costs of 1 cent per bit and complete memory costs of 10 cents to 15 cents per bit by 1967 for an 8,192 to 16,384 word memory operating at a cycle time of 100 nanoseconds. The cycle time for this capacity memory may be reduced to 50 ns by 1968 or 1969.

A semi-automated thin-film fabrication system has been developed for the REDMAN memory in which 50 substrates can be processed in one evacuation. The 50 substrates are initially stacked on one side of a specially designed vacuum chamber. The substrates are then passed one at a time successively through a substrate heating area, a magnetic film deposition area, a silicon oxide deposition area, and into an output stack on the other side of the vacuum chamber.

Another development program is pointed toward the batch fabrication of a very large (10^6 words) low-cost memory using magnetic thin-film techniques.[7,8] Those working on this development believe that the real future of magnetic thin-film memories is in the large-capacity low-cost area and that there is more to be gained in this area than in the small-capacity high-speed area. Their approach is to fabricate high-density planes by first depositing a thin continuous permalloy film on a thick glass substrate. Next, a continuous copper film is deposited on top of the permalloy film. Conventional photo-resist, masking, and etching techniques are used to etch through both the copper and permalloy to the glass substrate leaving thin strips of permalloy with copper on top of the permalloy. This assures exact registration between the copper that will become one of the drive wires and the permalloy film that is the actual storage media. A similar glass substrate with permalloy and copper strips is placed at right angles over the first one (with suitable insulation) providing a storage matrix where each bit of storage is defined by the cross-over of two copper/permalloy strips. This permits a two wire addressing, driving and sense system in which the upper and lower permalloy elements at each cross-over form a semi-closed-loop magnetic structure.

The glass substrate is 8 inches long by 1.6 inches wide and approximately 1/4" to 1/2" thick. After etching, the lines are 2 mils wide on 8 mil centers. The permalloy film is 300 to 1,000 angstroms in thickness and the copper conductors are 20,000 to 40,000 angstroms in thickness. The resistance of each copper lead is approximately 25 to 55 ohms. The creep problem (interference between adjacent bits in high-density planes) is the major difficulty to be overcome. The strip technique, with the copper conductors deposited and etched on top of the permalloy, overcomes the registration problem for high-density planes.

Yields of 50% have been experienced in low-density planes (50% of the planes had no bad elements), but the yield to be expected with the high-density planes is unknown. A combination of fabrication techniques and the provision of some redundancy to permit eliminating a few bad lines is being depended upon to provide a reasonable yield. The following advantages are cited for the use of high-density planes in this program:

1) Cost (more bits per fabrication process);
2) Fewer interconnections;
3) Short electrical lead lengths in very large memories.

The goal of the present development project is a 10^6 word memory with 36 bits per word. Within the next year, a 10^6 bit prototype should be in operation. The total 10^6 word memory will require 30 memory stacks, 30 word drivers, 40 digit drivers, and 2,000 sense amplifiers. The memory will be organized as 18,000 words (linear select) of 2,000 bits each. The designers estimate that sense amplifiers can be built for $20 each and the entire memory (36 x 10^6 bits) for $100,000,or 0.3 cent per bit.

The current state of the art in planar thin-film memories is illustrated by the following listing which shows the status of four memories intended for specific military computers by one manufacturer:

Words	Bits/Word	Speed	Status
128	16	3 megacycle	In production
128	50	4 megacycle	Operational
8000	50	2 megacycle	1/2 operational
1024	24	20 megacycle (NDRO) 10 megacycle (R/W)	Goal of early 1965

Small capacity magnetic thin-film memories have already been used in several computers to provide a few hundred words capacity for multiple high-speed registers requiring read/write cycle times of fractions of a microsecond. In one machine, a small magnetic thin-film memory is used to mechanize multiple arithmetic control registers. In another, a magnetic thin-film memory is used to provide a small high-speed multiplexed input/output buffer that also serves as an internal scratch-pad memory. At least one special purpose computer is currently being designed with an 8,196 word thin-film main memory.

Improvements in the cost of magnetic thin-film memories are being made rapidly, so that their cost may be competitive with linear-select or word-oriented core memories in the near future. The choice between linear-select magnetic core memories and magnetic thin-film memories will be made primarily on a basis of speed vs. cost. Because of the inherent magnetic and electronic techniques involved, there is little cost saving in slowing down a magnetic thin-film memory to operate at speeds slower than 0.5 microsecond cycle time. On the other hand, there are significant cost penalties in trying to operate magnetic core memories at less than one microsecond. These considerations indicate that magnetic thin-film memories will be in widespread use during the 1970's.

4.0 CYLINDRICAL MAGNETIC THIN-FILM (PLATED WIRE) MEMORIES

Another type of magnetic thin-film memory is fabricated by plating a magnetic film on a wire substrate.[9] The wire substrate then serves as one of the electrical conductors in the system. A closed flux path is obtained by the magnetic film surrounding the wire in a small region. A cylindrical thin film of this type offers the advantages that the closed flux path requires smaller digit currents, produces a larger sense signal, and is not as susceptible to creep. The fact that the wire substrate is used as either the digit or word conductor reduces the mechanical registration problems in the fabrication of the memory. The major problem with this type of memory has been the difficulty of producing satisfactory cylindrical films. Recent developments in these fabrication techniques have made this type of memory appear very promising.

In a recent paper, "Plated Wire Magnetic Thin Film Memories,"[10] presented at the 1964 Intermag Conference, Danylchuk and Perneski presented the following comparison of plated wire and planar film memories.

"1) Production method and control

Comparison: Plated wire is produced and tested in a continuous, serial process. Flat films are batch produced.

Conclusion: For plated wire, small numbers of bits may be rejected if bad. For flat films entire arrays may have to be rejected due to failure of a single bit.

Comparison: The plated wires can be expected to be subjected to strain during fabrication of a memory plane. Flat films, which are plated on rigid substrates, are relatively free of strains due to handling.

Conclusion: During production a careful control of the magnetostriction of the plated wire must be maintained in order to prevent adverse strain effects.

"2) Output signal level

Comparison: The plated wire circumferential mode can use film thicknesses of 1 μ (10,000Å) and more due to the closed flux path at remanence. Flat film thicknesses, which are limited by demagnetizing fields, are normally on the order of 1000Å.

Conclusion: Output signals are considerably higher for circumferential mode plated wire memories, while bit packing densities are comparable to those achieved with flat films. The axial mode plated wire has no advantage over flat films in this category since its film thickness is also limited by demagnetizing fields.

"3) Current levels

Comparison: Optimum coupling exists between fields produced by currents in the plated wire and the magnetic film deposited on the wire. For flat films, both digit and word solenoids must be added to the film array, and the opening of these solenoids are normally from 0.003" to 0.005". Also, slotted drive lines are normally necessary to permit penetration of the field generated by the uppermost conductor to the magnetic film in the planar construction.

Conclusion: For memory organization using plated wire as the digit line (circumferential mode), small (15 ma) digit currents are required. Alternatively, an axial mode memory which uses the plated wire as a word line, requires relatively small (200 ma) word currents. On the other hand, since the plated wire diameter (0.005") sets a lower limit on the area of the solenoid enclosing the wire, word currents are 0.7 to 1 amp for the circumferential mode,and digit currents are approximately 0.2 amp for the axial mode. For flat films digit currents are approximately 0.2 amp, while word currents are approximately 0.3 amp. It must also be pointed out that using the plated wire as a digit or word line leads to a much higher characteristic impedance than for a corresponding flat film memory line."

In one type of cylindrical thin-film memory used in a commercial computer the direction of magnetization of the cylindrical elements is parallel to the center conductor rather than circumferential to it.[11]

In this rod memory, a cylindrical thin film of magnetic material is deposited over a conductive substrate, but the axial switching mode produces an open flux path element. A multiple turn winding is placed over the plated rod for each bit position. The plating material is essentially isotropic and proper operation of the memory does not depend upon anisotropy in the material.

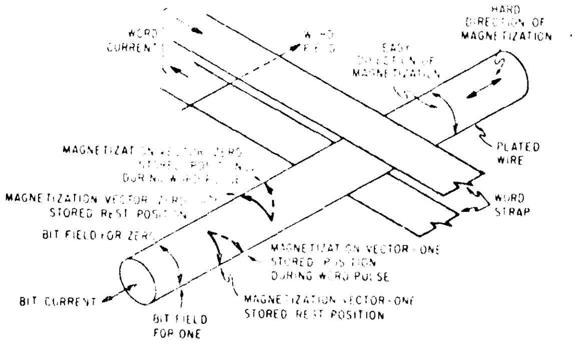
The manufacturer claims that this magnetic rod memory is more advantageous than either magnetic core matrix memories or planar thin-film memories in the range of 0.5 to 1.5 microseconds and hopes to lower this range to approximately 0.3 microsecond. Magnetic core matrix memories are more advantageous at slower speeds in the range of 2 to 6 microseconds; magnetic thin-film memories are more advantageous at very fast speeds of about 0.1 to 0.25 microseconds. They estimate that in the range of 0.3 to 1.5 microseconds, the magnetic rod memory will have approximately the same price per bit as a two microsecond coincident current magnetic core matrix memory; approximately 25 cents per bit including electronics. The electronics are compatible with and similar to those of magnetic core memories with a sense output of 80 millivolts being provided. Two major factors are cited as making the rod memory economically feasible.

1) Rod manufacture is a continuous process including the automatic wiring of a pair of spiral wires around the rod.

2) Tooling has been developed to permit automatic fabrication of the memory stack (including multiple turn windings into which the rods are inserted).

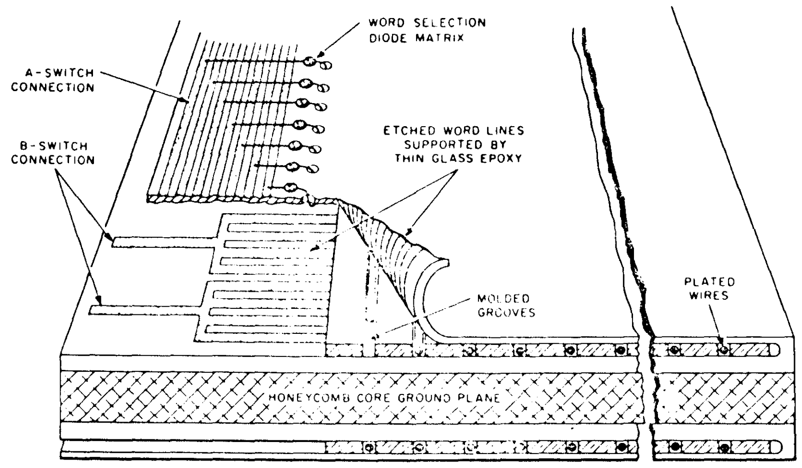
The lack of a closed flux path is recognized as a disadvantage, but the use of a high coercivity magnetic material, permitted by the tight coupling of the multiple turn winding, overcomes this disadvantage. As a result of the high coercivity of the material, the tight coupling, and multiple turn windings, fast switching and large output signals can be obtained.

In another type of cylindrical thin film (plated wire) memory, the direction of magnetization of the cylindrical elements if circumferential to the plated wire providing a closed flux path (see Figures 2 and 3). A 100,000 bit plated wire buffer memory of this type has been delivered to NASA for use in a spacecraft.[12] It operates at a sequential bit rate of 100 kc with 500 milliwatts power consumption.[13] Similar techniques are presently being used on an RADC contract for a six-month feasibility study of a 100×10^6 bit plated wire memory.



WORD CURRENT
WORD FIELD
EASY DIRECTION OF MAGNETIZATION
HARD DIRECTION OF MAGNETIZATION
PLATED WIRE
WORD STRAP
MAGNETIZATION VECTOR ZERO STORED POSITION DURING WORD PULSE
MAGNETIZATION VECTOR ZERO STORED REST POSITION
BIT FIELD FOR ZERO
MAGNETIZATION VECTOR-ONE STORED POSITION DURING WORD PULSE
MAGNETIZATION VECTOR-ONE STORED REST POSITION
BIT CURRENT
BIT FIELD FOR ONE

Figure 2. Information Storage on Plated Wire



WORD SELECTION DIODE MATRIX
A-SWITCH CONNECTION
B-SWITCH CONNECTION
ETCHED WORD LINES SUPPORTED BY THIN GLASS EPOXY
MOLDED GROOVES
PLATED WIRES
HONEYCOMB CORE GROUND PLANE

Figure 3. Sketch of Plated-Wire Memory Plane

A 10,000 angstrom permalloy film is plated on a 5 mil beryllium copper wire. This wire acts as both a substrate for the plating and as the digit drive and sense wire. During the plating process a polarizing current is sent down the wire to give a magnetic easy axis circumferential to the wire. The word drive conductors consist of flat metal straps placed over the parallel digit wires.

The word drive straps are either returned under the digit wires or terminated in a ground plane beneath the digit wires. The digit wires are placed on 15 mil centers and the word drive straps on 45-70 mil centers. The bit density is approximately 1,500 bits per square inch.

The plated wire costs approximately 0.1 cent per bit. One million bit memories have been quoted at approximately 8 cents to 9 cents per bit for one of a kind memories, including electronics. The target cost for the 10^8 bit RADC memory is 1 cent per bit, but the actual cost will probably be lower than this.

The film switching time is approximately 40 to 80 ns. The following cycle times are anticipated:

1) 100 ns read/write cycle for small control memories;

2) 150 ns read cycle for 4000 word NDRO memories;

3) 250 ns write cycle for 4000 word read/write memories.

This memory uses low-level sense signal switching and selection prior to the sense amplifier. The same two-transistor switching matrix is used for the sense signal and the digit drive signal. Hence, one end of the matrix is connected directly to the beryllium copper wire substrate that serves as digit drive line and sense line.

The sense signal from the storage array is 10 to 15 millivolts. A unipolar word drive current of 500 to 1000 milliamps, and a bipolar digit current of approximately ± 35 milliamps are used. Due to the low inductance resulting from the geometry of the memory array plane, the back voltage on the word drive line is quite low (about 3 volts across 200 wires). For 10^8 bit memories fabricated in 10^5 bit banks, this plated wire memory achieves an average of approximately 525 bits per wired interconnection compared to 150 for thin-film memories by the same manufacturer.

The 10^8 bit memory for RADC is designed in 10 banks of 10^7 bits each. The digit lines will be 8 feet long and the word lines 6 feet long. The memory is to be packaged in a 4 x 5 x 1.5 foot package including the power supplies and electronics. A portion of this memory has been fabricated to prove feasibility. The breadboard consists of a 4 x 5 foot ground plane with a limited number of full length bit lines intersected by a limited number of full length word lines. In the final memory, each module will consist of a 3.5 x 4.5 foot array of four planes (essentially a 7 x 9 array folded on itself in each direction). The 100 x 10^6 bit memory will be assembled from 9.4 x 10^6 bit modules. Each word line will control 64 words of 72 bits each (4608 bits per word line) with the 72 digit drivers and 72 sense amplifiers switched to one of the 64 words on the selected word line. A 10 microsecond read/write cycle is planned, but a 1 microsecond cycle can be achieved for a slightly higher cost -- possibly 20%.

A number of plated wires with a reasonable yield are being fabricated currently, but with the present technique, it would take approximately one year (assuming a 50% yield) to make the necessary number of wires for a 100 x 10^6 bit memory. As a result, one of the goals of a manufacturing study to be conducted next year will be to cut this time to three months by developing methods of plating at a rate of 15 inches per minute and of plating three wires in parallel through the same set of baths.

The following advantages are cited for this plated wire memory:

1) Cheap;

2) High speed for this size memory;

3) Simple fabrication;

4) Closed flux path structure;

5) Non-destructive read out (NDRO) on both read and write, i.e., write drive current on the word line does not destroy the stored information in those bit lines which do not have a bit drive signal. This permits a very large word, only a portion of which is read or written at a given time. Hence, the selection of a word drive line selects a large number of bit positions comprising the memory word, but only a limited number of these comprising the machine word are actually read or written as determined by the bit drive line switching matrix.

The disadvantages are:

1) Relatively high power (200 watts);
2) Size (20 to 30 cu. feet);
3) Weight (400 to 500 lbs.).

These are disadvantages relative to the memory being developed under the competitive study contract, but not relative to disc files.

Except for very fast control or scratch-pad memories, plated wire memories using techniques similar to those in this RADC memory are among the more promising competitors for the major types of storage required in a 1970 ACDS.

This plated wire memory study for RADC is in competition with a parallel study of a permalloy sheet toroid memory described in the next subsection. The initial goals for these studies are:

1) Cost to military user (less than 1 cent per bit)
2) Cycle times (under 100 microseconds)
3) Capacities (100×10^6 bits)
4) Volume (less than 30 cu. ft.)
5) Power (less than 500 watts)
6) Weight (less than 200 lbs.)
7) MTBF (5000 hrs.)
8) NDRO and random access to the word
9) Military specifications for shock, vibration, temperature, etc.

The studies to date indicate that all of these specifications will be met or exceeded. It is anticipated that the costs can be roughly comparable to the 0.2 cents to 0.25 cents per bit for militarized disc files. If the preliminary results of the two six-month study contracts under way at present prove out in final design and production, the random access mass memories will equal or exceed the characteristics of the militarized disc file currently being procured by the Navy in all respects except cost. If the cost can be made competitive, these memories will represnt a significant advance in large capacity storage for tactical military applications; but there is, of course, an appreciable difference in the availability of these memories (approximately mid-1967) and the militarized disc file (early 1965). Also, it is quite possible that by the time one of these random access memories is available, the capacity of the militarized disc file will have been increased and the cost per bit decreased. Although a cost differential may exist for random access magnetic memories for a long time, they will certainly offer advantages in speed, reliability, size, weight, and power.

Two competitive six month study contracts were awarded by the Air Force in May 1964 - one for the plated wire memory discussed above, and one for the permalloy sheet toroid memory discussed in the next subsection. Both contracts will likely be continued on a competitive basis through the technical feasibility phase and the first part of the manufacturing feasibility phase (approximately December 1965). At that time, the best technique will be selected and the work will be continued with one of the contractors. The approximate schedule for the development of 100 x 10^6 bit random access memories is:

Tasks	Duration	Time Schedule	Comments
Study	6 months	May 1964 - Nov. 1964	Essentially completed
Technical feasibility	9 months	Feb. 1965 - Nov. 1965	Includes design of integrated circuit bit drivers and sense amplifiers and completion of skeleton memory with 1 x 10^6 bit capacity by both contractors. Probably only 10^5 bits actual working storage - amplifiers, drivers, etc.
Manufacturing techniques feasibility	Phase I 10 months	Feb. 1965 - Dec. 1965	Study of manufacturing and fabrication techniques for both approaches.
	Phase II 5 months	Jan. 1966 - May 1966	Expansion of the 1 x 10^6 bit memory from the technical feasibility study to a 10 x 10^6 bit prototype using the manufacturing techniques developed during Phase I, but by only one contractor.
First-full scale model	16 months (approx.)	Feb. 1966 - May 1967	Full 100 x 10^6 bit memory using manufacturing techniques of the one selected approach and contractor.

5.0 PERMALLOY SHEET TOROID MEMORY

The permalloy sheet toroid memory is another approach to the batch fabrication of a low-cost large-capacity memory. In this technique, a permalloy sheet is bonded to a substrate. A pattern is printed on photo resist covering the sheet, and the permalloy is etched away to leave a matrix of toroidal permalloy storage elements. Several successive printing, etching, and plating processes are involved in making the connections from one side of the array to the other through the toroids and the connections on both sides of the board for the drive and sense wires in the array (see Figure 4).

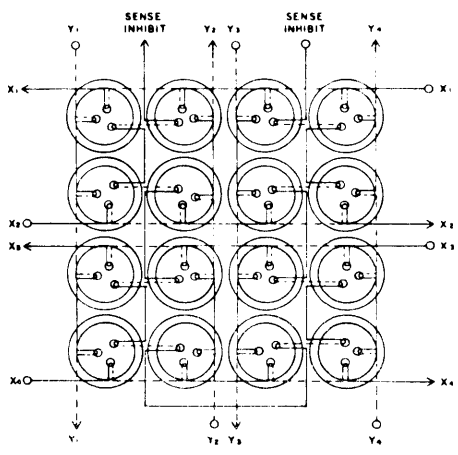


Figure 4. Three-Conductor Wiring Pattern for a Permalloy Sheet Toroid Memory Plane

This design effectively combines a number of known techniques into an interesting batch fabrication technique for a large memory array. This memory is described in greater detail in a paper presented by Fuller at the 1964 Intermag Conference.[14]

This development program is "aimed at improving processing speeds by providing data processing systems with random-access memories approaching the speed of ferrite-core matrix-memories, but at a cost that economically permits mass memory capacities."

The goal in this development is a large capacity memory in which coincident current selection is used to reduce the electronic costs, and in which a very large memory plane (256 x 256 bits) is used to reduce storage wiring fabrication costs as well as drive, selection, and sense electronics costs. In a 6.5 million bit memory module, it is planned that the smallest unit that is individually handled in processing is a matrix plane of 65,000 bits. The cost objectives for this development are 0.3 cents per bit or less.

This is the second approach to 10^8 bit memory supported by the two RADC study contracts discussed in the preceding subsection. During the study contract, 16 x 16 planes have been fabricated. In a few of these, all 256 bit positions are good. The contractor is now working on 64 x 64 planes with 256 x 256 considered the next step. The ultimate memory will be made of 6.5×10^6 bit modules each containing 100 of the 256 x 256 planes. The memory has a read/write cycle time of 35 microseconds and an access time of 15 microseconds. It is doubtful that these speeds can be improved significantly.

Coincident current write and an unusual type of coincident select read are used in this memory. Coincidence of two frequencies (rf) is used in read selection with the sum or difference frequencies (resulting from the non-linear characteristics of the magnetic toroid) being detected for sensing. This technique is NDRO and provides a signal to noise ratio of 10 or 20 to 1.

Problems with magnetostrictive effects due to the stress in the permalloy before it is bonded to the substrate have been experienced, but this problem has apparently been overcome by heat treating the permalloy after bonding to relieve the stress. Getting the process under control and providing sufficiently clean rooms for the fabrication are major problems.

For the 10^8 bit memory, the permalloy sheet toroid approach requires less power (approximately 80 watts), smaller size (12 cubic feet), and less weight (100 lbs.), but it will always be significantly slower than the plated wire approach. The fabrication process is also more complex.

This type memory is more promising for large auxiliary on-line storage applications than for internal storage because of its slow speed. The eventual cost will be a major factor in determining its usefulness as auxiliary storage for a 1970 system.

6.0 LAMINATED FERRITE MEMORY

The laminated ferrite memory is another promising approach to batch fabricated memories. Different versions of this type of memory are proposed for both small-capacity, high-speed, control memory applications, and for large-capacity, medium-speed, main internal memory applications. This memory and the flute memory discussed later differ from other batch-fabricated memory arrays in that ferrites rather than thin metallic films are used for the magnetic media.

Basically, the memory consists of a matrix of X and Y wires imbedded in a solid sheet of ferrite (see Figure 5). In fabricating the memory plane, a pattern of parallel conductors

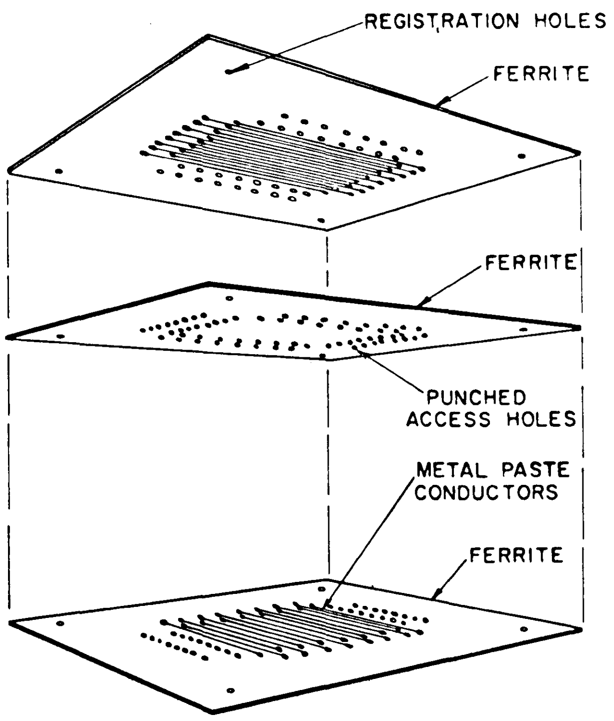


Figure 5. Laminate Details for Laminated Ferrite Memory

is printed on a glass substrate by a "silk-screening type process." A film of ferrite is spread over the conductor pattern on the substrate by a process called "doctor-blading." After the ferrite binder dries, the ferrite is peeled off the substrate with the conductors imbedded in the ferrite sheet. This sheet is approximately 0.0025 inches thick. A second ferrite sheet is made with the conductor pattern running at right angles to that in the first sheet. A third ferrite sheet, without imbedded conductors and only 0.0005 inches thick, is inserted between the two ferrite plates with the orthogonal conductors. This three-sheet sandwich is laminated by pressing the sheets together at moderate pressures and temperatures. Sintering the laminated sheets in a controlled temperature furnace provides a uniform isotropic material.

The matrix of conductors provides the necessary wires for a two wire memory system in which the wires in the word oriented direction are used as read and write drive lines,and the wires in the perpendicular direction are used as sense lines and digit drive lines. The write current generates a closed flux path in a plane perpendicular to the plane of the read/write drive wire. The current through the digit wire rotates the magnetic vector slightly. The methods of reading and writing in this type of array have been described in detail in a paper by Shahbender, Wentworth, Hotchkiss, and Rajchman.[15]

The laminated ferrite memory is proposed for two different applications. The first is as a high-speed control memory with approximately a 100 nanoseconds read/write cycle. In this application, a 256 word memory with 64 bits per word will require approximately 350 milliamps of drive current and will give a sense signal of $\pm$ 10 millivolts. The second application is as a large-capacity medium-speed internal memory with a read/write cycle time of 1 to 3 microseconds. In this application, a drive current of approximately 50 milliamps is required to produce a sense signal of 1.2 millivolts.

A 256 x 256 array is currently being developed in the laboratory and 64 x 64 bit arrays are said to be in production with a sufficient yield in which all elements in a plane are good. The following characteristics have been achieved in production memories with 64 words of 32 bits each:

1) Sense signal input is 20-40 millivolts;
2) Read current is 400 milliamps;
3) Two cross-overs per bit;
4) Conductors on 15 mil centers;
5) Conductor size is 5 mils x 1 mil.

It is necessary to operate this memory in a word organized manner, but the manufacturer does not consider this a limitation. Their argument in support of this is: "Although coincident current memories require less electronics, the electronics are more expensive because of problems with noise in the sense line, tolerances, and back voltage on the drive wires. On the other hand, word organized memories require more electronics, but they are less expensive since less critical tolerances are placed on the material, there is less noise on the sense line, and less critical tolerances are placed on the drive wire signal. Therefore, the amount of electronics is greater for the word organized memory but the total cost of the electronics may not be significantly greater because of the lesser requirements placed on them."

Significant success in reducing the cost of a memory depends upon the use of relatively cheap integrated circuits to permit the electronics associated with the memory to be made by batch-fabrication processes. In a typical memory, the wired and tested array represents only approximately halfof the total cost of the memory, with the other half going for the electronics. As a result, array fabrication techniques such as the laminated ferrite can reduce the total memory cost by only 50%, even if they are assumed to lower the array cost to essentially zero. Concentrated efforts on reducing the cost of integrated circuit electronics for such memories are required, if significant improvements in memory costs are to be achieved. Hence, great emphasis is being placed on developing integrated diode arrays for word selection. Operating large memories with currents under 50 milliamps would permit the use of integrated circuits and keep the capacity low, but this has not been achieved yet. Strips of 10 diodes and 8 x 8 matrix arrays of diodes that have a back voltage in excess of 60 volts and a drop of 1.5 volts at 400 milliamps have been fabricated for use in laminated ferrite memories.

Costs for the laminated ferrite memory are estimated to be in the order of 1 cent per bit for memories with capacities of 10^7 to 10^8 bits, speeds of 1 - 3 microseconds, and word sizes of 200-400 bits per word. The manufacturer expects to be making laminated ferrite memories on a commercial basis with 10^6 bits capacity in two years, 10^7 bits capacity in four years, and 10^8 bits capacity in five years.

7.0 FLUTE MEMORY

Another promising approach to a batch fabricated memory array using ferrite material is the flute memory which is very similar conceptually to the laminated ferrite memory. However, the fabrication techniques are different. The characteristics and fabrication techniques of this memory were described in a paper by Bartkus and several other authors.[16]

The planes in the flute memory are fabricated by sandwiching a pre-prepared grid of wires between two dies, each of which has matching grooves filled with a mixture of ferrite thermosetting resin binder. The grooves containing ferrite are parallel to the word lines of the wire grid so that when the two halves of the mold are placed together, the word line is completely imbedded in the ferrite tube formed by the ferrite in the corresponding grooves of the upper and lower halves of the mold. The bit lines of the wire grid are orthogonal to the word lines and intersect each of the parallel ferrite strips encasing the word lines (see Figure 6).

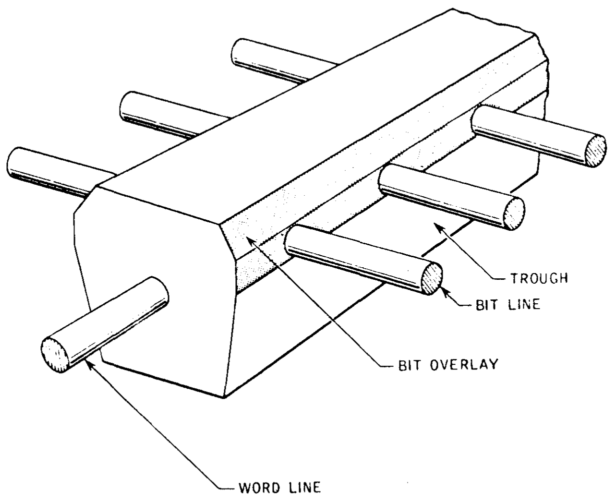


Figure 6. Flute Memory Elements with Word and Bit Lines Imbedded in Ferrite Bar

A typical memory plane consists of 50 ferrite tubes intersected by 100 bit lines to give a capacity of 5000 bits per plane. The ferrite area surrounding the intersection of a bit line and a word line defines an individual bit position. Yields of 36% have been realized in the batch-fabrication process. Cycle times of 250 nanoseconds are considered possible by the manufacturer.

The geometry used in the version described in the paper referenced above has been deemphasized because of the magnetic structure at each element (similar tothat of the laminated ferrite memory). A new geometry, designed to provide a magnetic structure very similar to that of a magnetic core to permit coincident current operation in large memories, has been described by Elfant.[17] In the new structure, the wire in the ferrite bar is not completely encased by ferrite but is embedded in the upper portion of the ferrite bar with part of the surface of the wire exposed. These ferrite bars with the partially embedded wires are placed parallel to one another. Bare (but insulated) wires are placed on top of these in a perpendicular direction to form a matrix. Additional ferrite strips are then placed at 45° angles to these wires so that the ferrite strips cross the intersection of the embedded and unembedded wires. The entire array is then sintered to give the ferrite the proper magnetic properties. The 45° ferrite strips and the ferrite in which the first wire is partially embedded give a magnetic structure at each wire intersection that is essentially similar to a toroidal magnetic core. Elfant states that this structure is almost as easy to fabricate as the original flute geometry and that it has the advantage of permitting coincident current operation due to the improved magnetic structure similar to that of a magnetic core. This is very important for a large capacity memory.

The new planes being fabricated contain a matrix of 100 x 150 bits (15,000 bits per plane). Densities of 10^4 bits per square inch are estimated for the future. Ten percent redundant leads are provided to permit the elimination of a limited number of bad elements. A special ferrite is used to permit a low firing temperature (945 to 950°C) while providing a relatively square hysteresis loop.

There are no indications as to whether this memory is to go into production, nor are any projected cost estimates available. However, it is a promising approach that will be in competition with the laminated ferrite, woven screen, plated wire, and other batch fabricated memories during the design of a 1970 era system.

8.0 MAGNETIC CORE MEMORIES

Magnetic core memories consisting of individual discrete magnetic cores threaded by selection, write, and sense wires to form a matrix or storage array have been the best established and most widely used memory technology for several years. This type of memory is so well established and its basic principles so well understood that it is not necessary to present a detailed description of core memories in this report.[18]

There are two major types of core memories. In the first, the selection is accomplished by coincidence of a current in one wire threaded through the core in the X direction and a current in another wire threaded through the same core in the Y direction, each conducting one half the required selection current. In the second, the selection is by a semiconductor or magnetic decoding tree which provides excitation current to all of the bits of the selected word with digit wires carrying inhibit currents through those bit positions of the word in which zeros are to be stored. The former are called coincident-current core memories and the latter linear-select or word-oriented core memories. In general, coincident-current core memories are cheaper because of the XY matrix addressing and selection (i.e., less selection and addressing electronics), but slower because of the tighter tolerances required on drive signals and the limitation on the total drive current that can be utilized (i.e., less than two times the minimum switching current). Word-oriented core memories, on the other hand, tend to be more expensive because of the linear selection circuitry required, but faster because of the less critical tolerances and the ability to switch the core faster by providing significantly more than the minimum switching current.

The replacement of magnetic cores by other memory technology has been predicted frequently. However, the magnetic core technology is so widespread and well established that it has presented a difficult and rapidly moving target for other technologies. For example, in approximately 1960, the advanced state of the art in magnetic core memories was represented by 4 microsecond cycle time and a cost of 25 cents to 50 cents per bit. By 1962, the speed had increased to 2 microseconds cycle time for roughly the same cost per bit. In late 1964, the advanced state of the art is represented by 1 microsecond cycle time memories for 10 cents to 15 cents per bit. Two or three years ago, few would have predicted the 500 nanosecond cycle time core memories now being designed for delivery in commercial computers in two years. This increase in speed and decrease in cost has resulted from two major factors - reduction in core size from 80 mils OD (50 mils ID) to 12 mils OD (7 mils ID) and significant advances in semi-conductor technology. The latter has permitted faster and higher power drive and selection circuitry and lower cost high-gain sense amplifiers.

In the past, anticipated advantages of other memory technologies, such as magnetic thin-films or cryogenics, were based on their enhanced characteristics over those anticipated for core memories. Unfortunately for other technologies, core memory technology has advanced so rapidly that by the time the proposed characteristics of the other storage devices were realized, those characteristics were no longer superior to characteristics available from magnetic core memories. In view of this past history, it is dangerous to predict that magnetic core memories will be replaced by other technologies by a given date, such as 1970. On the other hand, it is difficult to believe that these advances in magnetic core technology can be continued indefinitely. Additional advances in magnetic materials and semiconductors can be anticipated, but it is unlikely that the size of discrete cores can be reduced significantly below 12 mils OD and 7 mils ID and still permit the threading of addressing and sensing wires. As a result, magnetic core memories will eventually give way to either integrated circuit arrays or magnetic thin-films for small very high-speed memories and to some batch fabrication techniques, such as plated wires or integrated ferrite structures, for very large-capacity, low-cost memories. Magnetic core memories will probably remain the major competitor for main internal memories requiring cycle times of one to two microseconds and capacities of a few hundred thousand words or less for several years. Even this position will be threatened by 1970 - particularly if costs of some of the batch fabricated approaches live up to expectations.

The current advanced state of the art in magnetic core memories is represented by the following characteristics for main high-speed internal memories offered by three different manufacturers for delivery in 1965:

High Speed Internal Core Memories

R/W Cycle Time	Capacity in Words	Bits Per Word	OEM Price Per Bit
1 microsecond	8,000 words	56	12 cents
1 microsecond	16,000 words	40	13 cents
0.9 microsecond	16,000 words	12	14 cents

and the following characteristics for very large capacity auxiliary memories offered by three different manufacturers for delivery in 1965:

Large Capacity Core Memories

R/W Cycle Time	Capacity in Bits	OEM Price Per Bit
3 microseconds	1,000,000 bits	3 cents
3 microseconds	100,000,000 bits	1.5 cents
8 microseconds	20,000,000 bits	3.5 cents
12 microseconds	5,000,000 bits	1.5 cents

The figures quoted above represent significant improvements in core memory technology. One manufacturer privately announced a 0.5 microsecond magnetic core memory using 12 mil OD and 7 mil ID cores for a commercial computer to be delivered within the next two years. A large capacity coincident current core memory has been announced as part of a commercial computer to be delivered in 1965. This memory is available in banks of 262,144 words of 72 bits per word giving a total capacity per bank of 20,000,000 bits. A 20-30 mil core is used. The quoted sales price of 2-1/2 to 3 cents per bit implies a manufacturing cost under 1 cent per bit. The memory is fabricated in two groups of 32 planes each containing a matrix of 256 x 1152 cores. The memory has an 8 microsecond cycle time and 3 microsecond access time.

For any new technology to replace an established technology with 15 years of cumulative development and operational experience, the new technology must inherently offer significant advantages over the established technology. This significant advantage over the established magnetic core technology is most likely to result from the batch fabrication aspects of other memory technologies.

9.0 GLASS DELAY LINE MEMORIES

With higher speed logical components available, it may be desirable in some future computers to operate the machine in a bit serial manner with a very high bit rate. In this case, a high-speed bit-serial memory would be compatible with the other parts of the machine. If a high-speed bit-serial memory is desired, glass delay lines represent one of the more promising means of mechanizing such a memory. Glass delay lines are similar conceptually to the ultrasonic delay lines used in early computers such as Univac but provide higher frequencies, larger capacities, and better time and temperature stability.[19]

Typical characteristics for glass memories are:

1) Average access times - 5 to 500 microseconds
2) Bit rates - 5×10^6 to 20×10^6 bits per second
3) Capacities (for 100 delay lines) - 2×10^6 to 5×10^6 bits
4) Costs - 2 cents to 50 cents per bit

These characteristics represent a cost performance tradeoff between magnetic cores and magnetic drums, but the bit rate of 20,000,000 bits per second is higher than can be readily achieved with other types of memories.

Glass delay line memories usually use piezo-electric input/output transducers and either a rod of glass (for average access times below 50 microseconds) or a flat plate of glass (for average access times above 50 microseconds). In the latter type, carefully machined surfaces on the glass are used as reflecting media to permit a multi-path pattern within the glass plate. This increases the effective length of the delay line since this is a function of the total path length.

It is doubtful that glass delay line memories will be suitable for the central processor in a 1970 ACDS, but memories of this type may be useful in special purpose or peripheral equipment.

10.0 WOVEN SCREEN MEMORY

The woven aperture screen memory described in a paper by Davis and Wells represents a completely different approach to batch-fabrication of memory planes.[20] In this memory, the individual planes or storage arrays are fabricated by a weaving process on looms similar to those used for the commercial weaving of textiles. The woven cell is formed by weaving the proper combination and geometry of insulated wires and bare metallic wires to form an orthogonal matrix of drive and sense wires, threading magnetic cells. The magnetic cells are formed by plating the bare metallic wires (see Figure 7).

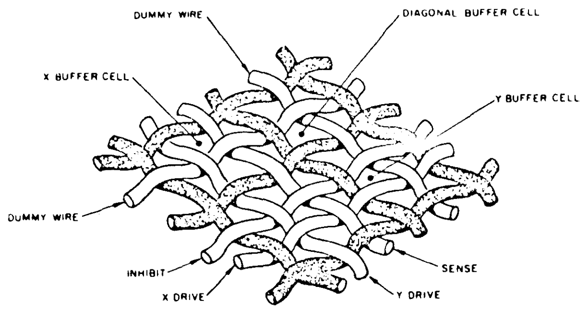


Figure 7. Typical 4-Wire Single-Turn Cell in Woven Screen Memory

An electrical deposition process is used to plate a permanent magnetic material on the square aperture cells formed by the bare metallic wires. This process forms a closed flux path resulting from the plating of a pair of bare metallic wires in the X direction and an intersecting pair in the Y direction. The square magnetic cell formed by these four wires had been threaded previously in the weaving process by the insulated wires representing the sense line, the inhibit line, and the X and Y drive lines.

Since the plating process affects only the uninsulated wires, it is possible to weave the entire plane prior to the plating process. When the memory cell arrays are formed by the plating process, all of the sense and drive wires are already threaded through the entire plane. The details of the fabrication and characteristics of this type of memory were described in greater detail in the paper by Davis and Wells.

A 10^8 bit memory assembled from 128 x 256 bit matrix planes would provide a 10 microsecond cycle time for 36 bit words. A memory of this capacity would be broken into 22 modules of storage planes. There is some uncertainty as to whether a sense preamplifier will be required for each plane. This is a question of major significance since 6336 such preamplifiers would be needed. If the need for these preamplifiers can be avoided, the cost of a large capacity memory of this type will be significantly reduced. This is related to the problem discussed in connection with the laminated ferrite memory - a true low-cost large-capacity memory requires batch fabrication and significant cost reductions in the memory electronics as well as in the storage planes themselves.

The designers expect the cost of coincident current woven planes in assembled stack modules to be about 0.1 to 1 cent per bit in production quantities. However, this does not include the cost of the electronics which could be considerable if sense preamplifiers are required.

The manufacturer is also working on a woven screen memory for internal applications - a two microsecond 8,000 word, 30 bit memory. The manufacturer expects this memory to be more rugged, to have higher temperature tolerances, and to offer a lower cost than equivalent magnetic core matrix memories. The anticipated operating temperature range, about 105°C should be of particular interest for military applications.

The development of integrated circuit sense amplifiers and drivers for memories will have a significant effect on the cost of large batch fabricated memory systems. An integrated circuit sense amplifier developed under Air Force contract AF33(657)-11185 has been considered for the woven screen memory.[21] This sense amplifier is capable of sensing a 400 microvolt input signal to provide a one volt output with a maximum delay of 30 nanoseconds and a cycle rate of 20 megacycles. The write drive amplifier represents a more difficult problem that has not yet been adequately solved in integrated circuit form. This is expected by 1966.

A different approach to weaving a memory array is represented by recent work in Japan. This has been reported in papers by Maeda and Matsushita[22] and by Oshima, Futami, and Kamibayashi.[23] The magnetic structure and geometry of this type memory is quite different from that described above. Although the array is woven, plated wires are used as the storage elements similar to those discussed previously under cylindrical thin-film memories. The plated wire acts as the storage medium, and also as the digit drive lines. In this woven array, insulated wires woven at right angles to the plated wires act as the word drive lines. This type of woven memory differs from the previous one in that the closed flux path magnetic element is the plating around a single wire rather than that formed by the rectangular intersection of four plated wires. This type of woven memory essentially represents a different fabrication technique for plated wire or cylindrical thin-film memories discussed in a preceding subsection rather than a different type memory.

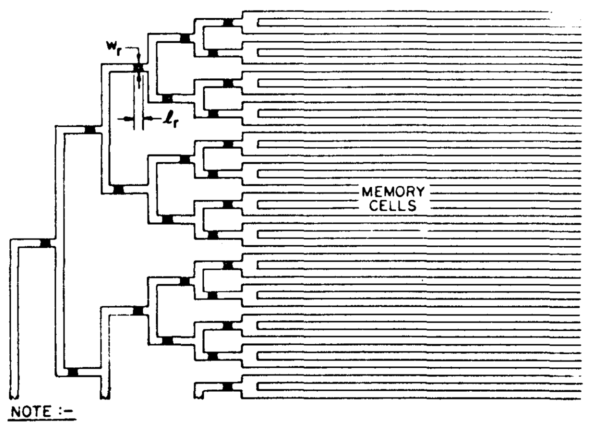
11.0 CONTINUOUS-SHEET CRYOGENIC RANDOM-ACCESS MEMORIES

Cryogenic memories are based on the principle that at temperatures near absolute zero degrees, the resistance of certain materials (super-conductors) may be either zero or some finite non-zero resistance depending upon the magnitude of the magnetic field surrounding the superconductor. In one type of continuous-sheet cryogenic memory under development, the storage medium is a superconducting tin film. This tin film, a lead sense line, and lead drive strips are fabricated by vacuum deposition techniques on a two-inch square substrate.[24] These metallic films are all insulated from one another by vacuum deposited insulating films (silicon monoxide). The sense line is beneath the storage plane and is oriented diagonally to and directly under the intersections of the two sets of drive strips which are orthogonal to one another.

Coincidence of the X and Y drive currents at an intersection of the drive strips produces a magnetic field sufficiently strong to switch the region of the storage plane beneath the intersection out of the superconducting state, thus permitting magnetic flux to link through the plane in that region. When the currents are removed, the flux linking the small region of the continuous sheet is trapped, and persistent currents are established in the storage plane to support this trapped flux. The stored information can be read by subsequently applying a coincidence of drive currents of proper polarity, and then sensing the voltage change on the sense line.

In a later version, a cavity sensing technique is used rather than a zigzag sense line.[24] A second continuous tin film is located a slight distance beneath the storage plane and connected to it electrically along one edge. When the proper polarity drive currents create a magnetic field sufficient to destroy the superconducting state in the region of the intersection, a change of flux is created beneath the intersection within the cavity between the storage plane and the sense plane. This causes a sense pulse at output tabs connected to the sense plane. Cavity sensing avoids the necessity for accurate registration of the sense line beneath the plane with the intersections of the X and Y drive lines on top of the plane. However, in early work with cavity sensing, the output signal has not been uniform over the entire plane. Recent results indicate that this problem may have been overcome, but it is too early to say which sensing technique will eventually prove superior.

This memory is made by a batch-fabrication process in which a continuous sheet superconducting storage film and sense film and the X and Y drive strips are vacuum deposited for an entire plane. In addition, the X and Y selection matrix is mechanized by means of cryotrons that are vacuum deposited on the same plane and connected to the drive strips by the deposition process. The ability to fabricate the cryotron selection matrix at the same time the storage plane is fabricated is one of the major advantages of cryogenic technology. A cryotron selection tree and associated drive strips are illustrated in Figure 8. A complete selection matrix would include a similar set of cryotrons and drive strips superimposed at right angles to the ones shown.



NOTE :-
FOR SIMPLICITY ALL CONTROL (ADDRESS) LINES ARE NOT SHOWN.
CRYOTRON GATES ARE SHOWN BY DARKENED AREAS.

Figure 8. Cryotron Selection Tree and Associated Drive Strip

A 16,384 bit memory plane (128 x 128), including 508 cryotrons for XY selection, has been fabricated on a 2 by 2 inch substrate. Twenty-six vacuum deposition steps using 16 different masks are required. The planar density is approximately 10,000 bits per square inch. Two such planes have been operated satisfactorily. Typical anticipated characteristics for cryogenic memories are:

1) 3 to 5 microsecond cycle time
2) 10^6 bits in a 10-inch square plane
3) 10^9 bits in a complete memory
4) 300 to 400 ma selection current
5) 1 to 2 mv sense signal
6) 10^9 bit memory will cost approximately $1,000,000 (or 0.1 cents per bit)
7) 3.5° Kelvin operating temperature

Two good arrays (128 x 128) have been fabricated, but both of these have been damaged by repeatedly taking them in and out of the liquid helium. A 262,124 bit plane (512 x 512) with 0.0005 inch spacing between wires (2 mil line width) on a four-inch square plane is under development in the laboratory. Planes of 1024 x 1024 on a 6 by 6 inch substrate are an eventual goal. By 1965, this manufacturer anticipates having a 10^7 bit prototype memory.

The costs for a 50 plane stack of 262,000 bit planes (13×10^6 bits) have been estimated by Burns to be:[25]

1)	Stack	5,000
2)	Stack assembly & wiring	5,000
3)	32 address drivers	4,000
4)	50 sense amplifiers	20,000
5)	100 drivers	10,000
6)	20 timing boards	2,000
7)	4 power supplies	8,000
8)	Hardware & controls	20,000
9)	Refrigerator	25,000
	TOTAL	$ 99,000

He also estimated that 262,000 word cryogenic memories can be built at a cost of approximately 1 cent per bit,and that 1,000,000 word memories would be a fraction of a cent per bit. On this basis, 262,000 words is not a competitive size for cryogenic memories and 1,000,000 words (50 x 10^6 bits) is probably the minimum size for which cryogenic technology should be considered.

The problems with cryogenic memories are both fundamental and economic. The major technical problem is attaining uniformity.

Major cost factors in the manufacture of a cryogenic memory are:

1) The cost of a refrigerator
2) The cost of sense amplifiers
3) The cost of planes, which is highly dependent upon yield rate
4) The cost of interconnecting planes.

Because of the basic overhead cost in the refrigerant, the cryogenic memory is not considered a good approach to a small memory, but becomes more advantageous as the size increases. The cryogenic approach may provide a more likely way to reach capacities of 10^9 bits, but other batch fabrication techniques will be better for capacities of 10^7 bits. The crossover between the two may occur somewhere around 10^8 bits.

RADC is sponsoring a cryogenic memory project that is due to be completed in September 1965. At that time, a 3 x 10^6 bit feasibility model will be delivered. It is anticipated that a 27 month manufacturing techniques contract will be initiated in January 1965 in conjunction with WADC. This contract will call for the development of manufacturing techniques and for a non-militarized memory with 1000 planes 1024 x 1024 each. The goal is a 10^9 to 10^{12} bit memory with a cycle time less than one microsecond and with costs in the order of 0.01 cents per bit.

The future of cryogenic memories is questionable. To date, design and fabrication problems, and rapid advances in other technologies have prevented the cryogenic technology from establishing a foothold. Some believe that cryogenics will represent the best approach to very large random access memories with capacities of 10^9 bits and above.

12.0 FERROACOUSTIC MEMORY

The term ferroacoustic memory is used to refer to a memory approach involving magnetic storage with acoustic access.[26,27] Two developments of this type have been sponsored by the U. S. Army Engineering Research and Development Laboratory at Fort Monmouth, New Jersey.

Although this is a batch-fabricated memory approach, it differs from those described previously in that it is not a random access memory. It is block oriented with random access to the beginning of a block but serial access to information within the block. A batch-fabricated solid state memory of this type could be a replacement for large magnetic drums, magnetic disc files, and possibly magnetic tape.

Previous types of delay line memories were volatile and suffered from the large amount of electronics necessitated by the requirement for regenerating and recirculating the information in each individual delay line. The necessity for recirculating the information results in the loss of stored information if power is interrupted. The requirement for electronic circuitry to be in continuous use for each individual line implies a severe cost penalty in very large capacity memories. The ferroacoustic approach permits static storage of information without continuous recirculation which, in turn, permits the electronic read and write circuitry to be switched from one line to another as part of the addressing and selection process.

In this type of memory, data is stored on a thin magnetic film plated on an acoustic tube through which a center conductor is inserted (see Figure 9). The actual storage is in a closed flux path around the acoustic tube similar to the storage in cylindrical thin-film memories discussed previously. However, the access is not made by coincidence of two electrical signals, as in the memories discussed previously, but rather by the coincidence of an electrical signal and an acoustic signal.

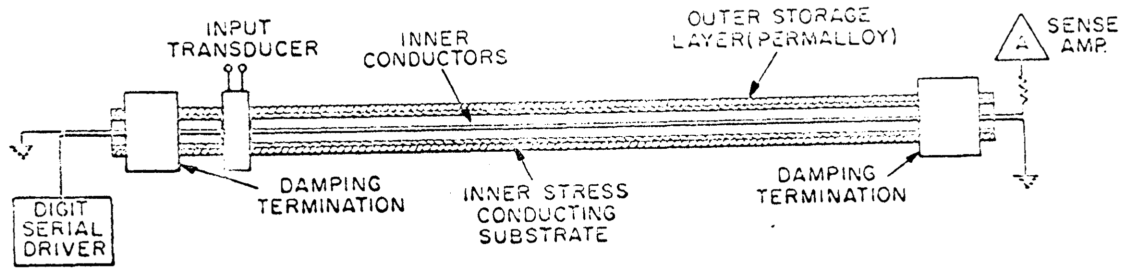


Figure 9. Typical Ferroacoustic Storage Line

The concept for this type of memory is based on the change of coercivity of magnetic materials, such as permalloy plating, when the material is placed under mechanical stress. An alternate approach is based on the fact that the anisotropic characteristics of thin permalloy films are changed by stress.

In a memory based on these principles, a mechanical shock wave is initiated in the acoustic tube by a suitable transducer (e.g. magnetostrictive, semiconductor, or piezo-electric). The mechanical shock wave travels down the tube at a speed determined by the propagation constant of the material. The coercivity of the magnetic material plated on the tube changes as the shock wave passes under it and returns to its normal condition after the shock wave has passed. As a result, a temporary change in the coercivity of the magnetic media is propagated down the line.

To write or store information, an electrical signal, corresponding to the serial bit pattern of 1s and 0s to be stored, is placed on the center conductor while the acoustic signal is traveling down the line. This electrical signal causes the magnetic cylinder to be linked by a varying flux pattern depending upon the bit pattern represented by the electrical signal. The flux pattern generated by the electrical signal corresponding to a "1" is sufficient to force the state of magnetization of the magnetic material into one direction in an area where the coercivity has been altered by the mechanical stress wave. The magnetic flux generated by a "0" signal will force the state of magnetization into the other direction in the area where the coercivity has been changed. As a result, a bit position is determined by the time coincidence of the electrical signal and the acoustic signal. As the acoustic signal travels down the line, a "1" or "0" corresponding to the electrical signal will be written in the position defined by the front of the shock wave.

Read-back is achieved by transmitting an acoustic signal down the line and sensing the changes in the electrical signal generated on the center conductor by the magnetostrictive effect of the shock wave traveling down the line. This signal varies, depending upon the stored pattern on the line.

The time required to write or read a complete block corresponds to the time the shock wave requires to travel from one end of the line to the other. The rate at which the stored information is read or written corresponds to the frequency of the electrical signal representing the bit pattern. This frequency is limited by the physical size of the area of magnetic material whose coercivity is altered by the shock wave at a given instant. This in turn determines the bit density and hence the capacity of a line of a given length.

The information is stored statically in the magnetic media and is not carried by the acoustic wave. This is in contrast to a normal acoustic delay line in which the information is actually carried by the acoustic wave. In the ferroacoustic type memory described here, the acoustic signal acts only as an access mechanism. A single acoustic signal is required to read a complete line or write a complete line.

The goals of the Army development program are to provide an all-electronic block-oriented random-access memory with the following characteristics:

1) Removable medium permitting non-volatile shelf storage
2) Small medium size (200 cubic inches)
3) 4,000,000 characters (28 x 10^6 bits) per module (approximately equal to one reel of magnetic tape)
4) 4096 characters per block
5) 1024 blocks per module, approximately 0.002 cents per bit ($500 per module) off-line storage cost not including read/write electronics
6) 1 microsecond random access to a block
7) Inherent sequential transfer within the block (ready-strobe to 1.5 - 3 million characters per second)
8) Read-write unit - 1/10 the power of tape unit; 1/10 the weight of tape unit; 1/2 the size of tape unit

If this development program is successful, it will be of great significance to future computer systems. At present, there is no economic all-electronic/magnetic replacement for large capacity electromechanical storage devices such as magnetic disc files, magnetic-card memories, and magnetic tape units. Such a replacement will be essential to future systems if the speed, cost, reliability, and size limitations of electromechanical input/output and auxiliary storage equipment are to be avoided for very large capacity storage functions (now handled by devices such as disc files and magnetic tapes).

If this ferroacoustic approach proves feasible, it offers the following advantages:

1) Static storage
2) Semi-serial access not requiring a physical coincidence of selection wires for each bit position
3) Ability to switch read and write mechanisms from one line to another
4) Large capacity semi-random access bulk storage with no mechanically moving parts
5) Possibility of off-line storage by plugging alternate blocks of delay lines into a read/write device
6) Low cost per bit of storage

A future all-electronic magnetic tape replacement will almost certainly be a block-oriented rather than a random-access-type storage device. Although a large-capacity, random-access, mass memory offers certain unique advantages, it is very unlikely that such a device can compete on a cost per bit basis with semi-serial electromechanical devices. The requirement for the physical intersection of electrical signal lines for each bit position,and the access electronics,will not permit on-line storage costs of a few millicents per bit by 1970. A semi-serial or block-oriented device providing random access to a block of information, but serial access within the block, will be necessary to permit the read/write electronics to be time shared by a serial bit train.

13.0 MAGNETIC SURFACE STORAGE

Magnetic surface recording is usually used in on-line auxiliary storage and off-line auxiliary storage. This technique has also been used for internal storage in the form of drums or discs, but magnetic core storage has essentially replaced these electromechanical devices in shipboard applications where maintenance and environment are critical factors. However, to date there has been no good replacement for on-line and off-line auxiliary storage. It is believed that some of the technologies discussed previously will replace these devices,but the timing of this is uncertain. Certainly, for the next few years, electromechanical devices using magnetic surface recording will be dominant. Off-line storage (e.g. magnetic tape units) has been discussed in the Input/Output section, but on-line auxiliary storage is discussed here. On-line devices of this type, frequently referred to as mass memories, are used to provide a large-capacity, fast, semi-random-access storage.[28,29] They should be under direct on-line control of the computer, addressable by the computer, eraseable, and reuseable. All devices of this type that are currently available are electromechanical. All-electronic/magnetic on-line auxiliary storages for military applications will be available by 1970 but all of these (with the possible exception of the ferroacoustic storage discussed previously), will be significantly more expensive in terms of cost-per-bit for very large capacity storage. This results largely from the fact that the all-electronic/magnetic approaches (e.g. laminated ferrite memory, woven screen memory, flute memory, etc.) require addressing each individual word. The electromechanical devices are block oriented in the sense that access is made to a particular track on a disc, drum or card, and then all information stored in that track (or block) is read or written serially by the same electronic circuitry.

Of the all-electronic/magnetic approaches to large capacity memory that have been discussed in previous parts of this report, only the ferroacoustic approach is block oriented and hence offers some promise of competing with the electromechanical devices by 1970 on a purely economic basis.

The characteristics of the major types of electromechanical mass memories available today are summarized in the comparison shown in Table 6-5. (Section 6; part I of Volume V). The values shown were chosen as typical of each type of unit but frequently they represent a wide range of possible values. In some cases, certain characteristics of an individual device may vary significantly from the values shown. The characteristics of primary interest include capacity, cost, average access time, and data transfer rate. Some of these are difficult to compare because of the different physical characteristics of the devices. For example, a large magnetic drum with a head for every track will have a continuous data transfer rate equal to the instantaneous transfer rate if the heads are switched electronically. However, the continuous data transfer rate for a disc file with moving heads will be significantly slower than the instantaneous transfer rate due to the necessity for interrupting data transfer while moving the head from one position to the next. Similar differences on a more detailed level exist between different devices of the same type.

As a result of problems of this type, comparison tables such as the one shown in Table 6-5 present at best a gross comparison for electromechanical devices. In selecting a device for any specific application, it is necessary to go into a more detailed comparison of the specific pecularities and quirks of each of the leading contenders as they relate to that application, if a proper decision is to be made. These comparisons in Table 6-5 cover commercial devices that are not designed to military specifications - otherwise, the costs would be much higher.

The Navy has two types of mass memories under development for shipboard use. The first of these is a militarized version of a disc file. This is due for delivery in the first quarter of 1965, with the following typical characteristics:

1) 100×10^6 bits capacity (in 2 modules)
2) 0.2 to 0.25 cents per bit
3) 200 millisecond average access time
4) 3,000 watts power
5) 1,800 lbs. weight
6) 63 cubic feet volume

The second unit, also due for delivery in the first quarter of 1965, is a militarized magnetic card unit. This unit will have the following characteristics:

1) 185 to 225 millisecond access time
2) 32 containers possible per system
3) 32 magnetic cards per container
4) 32 blocks per card
5) 875 words per block
6) 36 bits per word
7) 1×10^6 bits per card
8) Cost for first 50,000,000 bits - approximately 0.3 to 0.4 cents per bit
9) Next 50,000,000 bits - approximately 0.1 cent per bit (50,000,000 bits additional for $50,000)

The major advantages and disadvantages of the different types of mass memories are summarized in Table 1.

Future improvements in these devices can be anticipated. In 1962, A. S. Hoagland pointed out that the storage density of one manufacturer's commercial disc files increased from 2000 bits per square inch in 1956 to 25,000 bits per square inch in 1961.[30] He then predicted that storage densities of "one million bits per square inch (e.g. approximately 5000 bpi, 200 tpi) will become the state-of-the-art" within the next few years. A few months earlier, M. Jacoby predicted densities of 3000 bpi and 500 tpi (1.5 million bits per square inch) would "become commonplace in a few years".[31] He then indicated that these densities could provide storage capacities of 10 to 100 billion bits if a possible increase in the physical size is also considered. Thus, increases in capacity of tens to hundreds of times over the largest present mass memories can be anticipated.

The cost per unit can be expected to decrease even with the larger capacities as the technology is improved and more manufacturing experience is obtained. Hence, the cost per bit of storage can also be expected to decrease by one to two orders of magnitude -- possibly to 0.0001 cents per bit for the mass memory itself (not including control and buffering electronics). However, militarized versions of this type of device may cost at least an order of magnitude more.

Although the picture for the future of capacity and cost appear bright, there is little hope for significant improvements in average access times for moving-media mass memories. Due to the inherent mechanical motions involved, we cannot expect improvements of as much as an order of magnitude over available devices. For significant improvements in access time, we must turn to the non-moving-media type devices.

14.0 ASSOCIATIVE MEMORIES

A completely different memory concept has received considerable attention recently in which stored data is located by association rather than by physical location (numeric address). This type of memory is usually called an associative memory, a content addressed memory, or a search memory.[32] The former term is more widely accepted and will be used in this dicussion.

Table 1. Advantages and Disadvantages of Different Electromechanical Mass Memories

Type of Mass Memory	Advantages	Disadvantages
Fixed-Head Magnetic Drums	Fast access, no mechanical head motion, high continuous data transfer rate.	Low capacity, high cost per bit, poorer volumetric efficiency, large number of heads.
Moving-Head Magnetic Drums	Large capacity, low cost per bit, possibility of parallel reading or writing from multiple heads to greatly increase instantaneous data transfer rate.	Poorer volumetric efficiency, relatively large number of heads for medium speed access or slower access if fewer heads.
Fixed-Head Magnetic Discs	Fast access, medium capacity, no mechanical head motion, high continuous data transfer rate.	High cost per bit of storage, large electronic switching matrix, large number of heads.
One-Dimension Moving Head Magnetic Discs	Large capacity, possibility of multiple simultaneous accesses if heads are positioned independently, low cost per bit compared to fixed head units, possibility of parallel reading or writing from multiple heads to greatly increase instantaneous data transfer rate.	Relatively large number of heads, somewhat higher cost per bit compared to two-dimension disc unit, medium speed access.
Removable-Stack Discs	Large off-line capacity, low cost per bit of off-line storage, combines on-line random-access capability with large off-line capacity.	Limited on-line capacity, higher cost per bit of on-line storage.
Magnetic Card Memory	Large off-line capacity, low cost per bit of off-line storage, combines on-line randon-access capability with large off-line capacity, individual cards can be copied, replaced or inserted.	Slower access, card wear and replacement necessitates eventual rewriting of entire card.

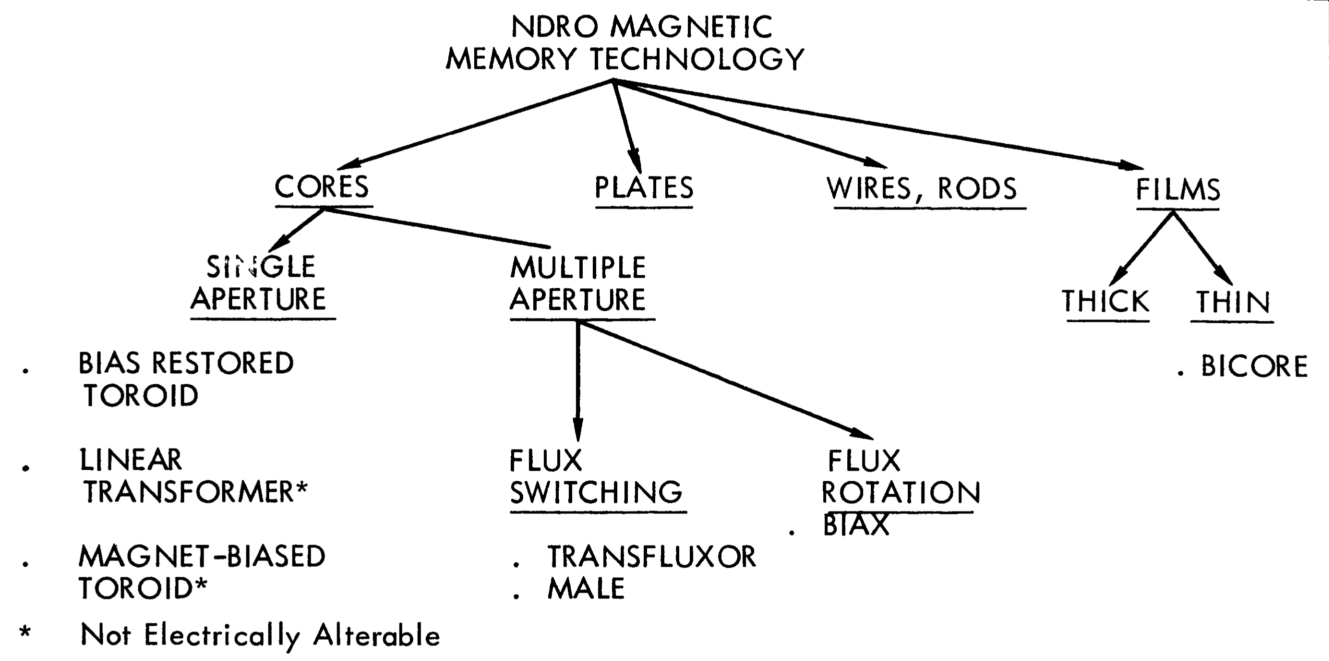
Associative memories are addressed by content rather than by a unique numeric address. An associative memory involves sufficient logical capability to permit all memory locations to be searched essentially simultaneously -- i.e. within some specified memory cycle time. The search may be made on the basis of the entire contents of each location or upon the basis of selected bit positions of each location. Thus, it is possible to search for all words meeting certain criteria or for all words in which a certain tag portion of the word meets the criteria. Searches may be made on the basis of equality, greater-than-or-equal-to, less-than-or-equal-to, and between limits.

Associative memories are also referred to as content addressed memories and search memories. In a sense, all of these terms are misleading in that the unit is not an associative "memory" but rather an associative "processor" since the memory function involves a minor portion of the total hardware and costs. The major portion of the hardware and costs is that involved in the logical operations necessary to accomplish the parallel search of memory locations. For example, a magnetic associative memory frequently requires a sense amplifier for each word location, and logical capability for each word (in some cases for each bit portion) to permit the parallel search capability. These hardware elements are not required for the memory function itself. The memory function alone might account for only 1/4 or 1/5 of the total costs based on comparing the cost of the associative memory with that of a random-access memory with equivalent capacity and access time.

Magnetic, cryogenic and semiconductor techniques have been used in mechanizing associative memories. Magnetic associative memory techniques usually use a non-destructive-read-out magnetic element such as a multiaperture core, a plated wire, or a thin film. The different approaches to magnetic associative memories and a comparison of four specific types have been presented in a paper by Shohara and are shown in Table 2.[33]

The mechanization of most search algorithms requires an argument register, a mask register, auxiliary registers, input and output registers, a shift circuit, and logic circuits. A typical search or associative memory block diagram is illustrated in Figure 10. Different methods of mechanization of searches include parallel by bit, serial by bit, parallel by word, and serial by word.

Table 2. Magnetic Associative Memory Techniques



COMPARISON OF MAGNETIC ASSOCIATIVE MEMORIES

	TRANSFLUXOR	MALE	BIAX	BICORE
SOURCE DATE	2/63	1/64	12/63	11/63
CAPACITY	1024/12	8/10	16/8	128/24
INTERROGATION				
MODE:	PARALLEL	PARALLEL	SERIAL	PARALLEL
RATE:	6 μsec	5 μsec	0.1 μsec	300 nsec
MATCH SENSE	DOUBLE BINARY	LOGICAL	TERNARY	DOUBLE BINARY

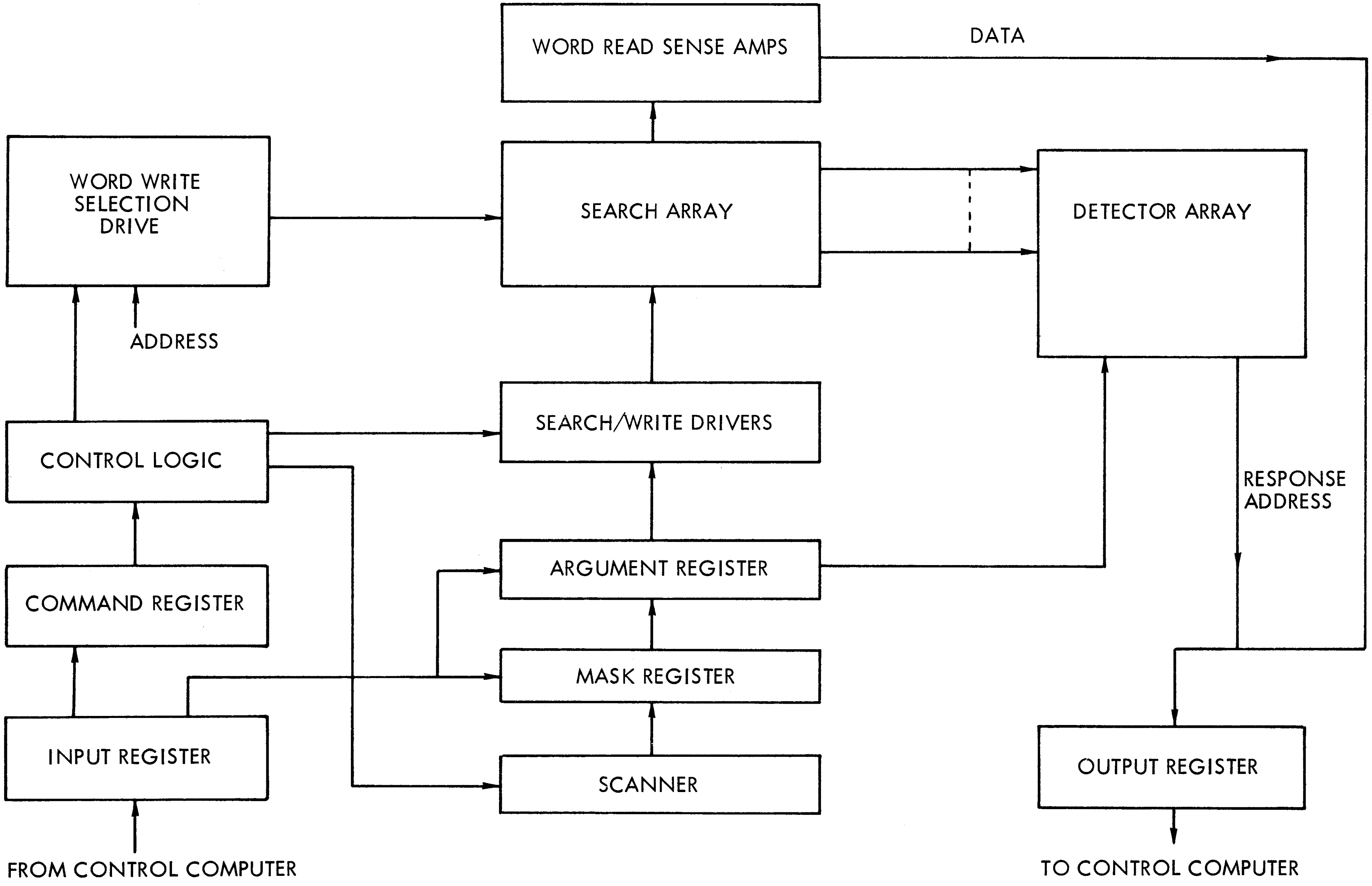


Figure 10. Typical Search Memory Block Diagram

In a paper by McAteer, Capobianco, and Koppel, some of the characteristics for different mechanizations of associative memories using the biax magnetic element are compared.[34] Since this table illustrates the trade-offs and characteristics to be considered in designing and selecting an associative memory, it is reproduced as Table 3.

15.0 CRYOGENIC ASSOCIATIVE MEMORIES

Work on cryogenic associative memories has also been underway at several companies.[35,36] The cryogenic approach is probably worthwhile only for very large capacity associative memories because of the overhead cost in the refrigerator. The cryogenic associative memory under development at one company is essentially a two "core" per bit type using two parallel continuous sheet planes. The associative feature depends upon whether a corresponding spot in both planes is storing the same information. If they are not, the flux between the two planes cancels out. This continuous sheet cryogenic memory uses lead for the super-conductive wiring, tin for the continuous-sheet superconductive storage planes and silicon monoxide for the insulators. These are all deposited in approximately 20 different deposition steps with different masks for each. Fabrication is currently mechanized, with a circular jig holding the 20 masks in the vacuum so that each mask can be rotated into place at the proper time.

In another cryogenic associative memory being developed, cryotrons are used for the association logic. So far, a 10 x 12 bit associative array using 2350 cryotrons per array has been fabricated. The present goal is a 5000 word associative memory with 48 bits per word. This is an 18 month program. A 10% yield on planes is expected. If the plane yield is sufficiently high, a 10,000 word memory will be fabricated rather than a 5,000 word memory. Both of these cryogenic associative memory developments are being sponsored by Rome Air Development Center.

Table 3

Associative Memory System Characteristics

Scheme	Mechanization	Relative Exact Match Search Time	Relative Limit Search Time	Data Writing	Data Reading (Additional Array Requirements)	Restriction On Fields for Limit Search	Write "Don't Care" Bit Available
1	Two-BIAX-per-bit Signal-no signal Binary sense output	$\frac{M}{k}$	$\frac{3F}{2} + \frac{F^2}{2k}$ (ave) $3F + \frac{F^2}{k}$ (max)	Whole word only	Additional array windings required	Fixed fields only	Yes
2	Two-BIAX-per-bit Signal-no signal Ternary sense output	M	M	Whole word only	Additional array windings required	No restrictions (any combination of bits selected by mask permissable)	Yes
3	Two-BIAX-per-bit Signal-signal Binary sense output	M	$\frac{3F}{2} + \frac{F^2}{2}$ (ave) $3F + F^2$ (max)	Unrestricted as to location and number of bits	No new windings required	Fixed fields only	Yes
4	One-BIAX-per-bit Signal-signal Binary sense output	M	$3F + \frac{F^2}{2}$ (ave) $6F + F^2$ (max)	Unrestricted as to location and number of bits	No new windings required	Fixed fields only	No

LEGEND: M = Number of bits per word
k = Number of bits per word interrogated simultaneously
F = Field length used in limit search

Prepared by Mcateer, Copabianca & Koppel

16.0 MAGNETIC ASSOCIATIVE MEMORIES

Magnetic associative memories differ from most magnetic random-access memories from a hardware standpoint in the following ways:

1) The sense lines are word oriented;
2) The drive lines are bit oriented;
3) There is a large number of sense amplifiers as a result of the word orientation of the sense line;
4) The sense amplifiers have a high duty cycle which tends to make them more expensive;
5) A large amount of logical circuitry is required.

The costs of the sense amplifiers and logical circuitry are the main factors contributing to excessive costs of associative memories compared to those of random-access memories. An associative memory also requires a non-destructive-read-out storage element since it is not feasible to rewrite every word location for each search. The trade-off between speed and cost is affected by factors such as:

1) Bipolar vs monopolar sense signals;
2) Poor signal-to-noise ratio resulting from interrogating a larger number of bits;
3) The number of bit positions interrogated in parallel;
4) Fixed vs unrestricted fields;
5) Parallel by word vs serial by word.

Although some magnetic associative memories use a single aperture magnetic core, it is convenient to use a multi-aperture device such as the Transfluxor or the Biax element. The Transfluxor approach requires two elements per bit and requires a greater drive signal since it is necessary to actually change the direction of saturation of the material around the interrogate hole. The Biax element requires a relatively expensive sense amplifier, but it is attractive for associative memories for three reasons:

1) A fast read capability;

2) A non-destructive read-out feature;

3) A bipolar sense signal.

Cylindrical thin film plated wires are also being investigated for associative memory applications. The cylindrical thin film plated wire appears attractive in that it offers the possibility of a large sense signal resulting from the closed magnetic path, and requires a relatively small read/write drive current.

One Bureau of Ships contract for an associative memory using the Biax approach was recently terminated, largely because projected cost goals could not be met. This associative memory was to store 2048 words of 36 bits each with the ability to search the entire memory for an exact match, for greater than, for less than, for between limits, and for simultaneous or successive combinations of these.[37] It also was to provide for arbitrary masking to control the bit positions on which the search is based. The results of the search can indicate match or no match, a count of the number of locations upon which a match was obtained, the addresses of all positions in which a match was obtained, or the data contained in each location in which a match was obtained. Optionally, all matched words can be modified automatically after a search. The basic search rate was to be 100 nanoseconds per bit position for an exact match. Therefore, a search of all 26 bit positions of the entire memory would require 3.6 microseconds. For all other search modes, 300 nanoseconds per bit position would be required--10.8 microseconds to search on all 36 bits. As a conventional read/write memory, the cycle time was 6 microseconds.

In this approach, one hole of the Biax memory is wired as a conventional coincident current read/write memory, using the Biax element in the same way that a simple magnetic core would be used. The orthogonal hole is then used for the associative feature. In the associative mode, given bit positions of all words are interrogated simultaneously, and successive bit positions of the memory are interrogated sequentially. All bits of a given word are sensed by the same sense wire, so that a sequential signal corresponding to the sequencing through the bit positions is obtained from an individual sense amplifier for each word position. The interrogation is conducted so that a negative signal from the bipolar sense signal during the active part of the cycle indicates the absence of a match. Therefore, the indication of any negative signals from a particular sense amplifier during this active period indicates a match for equality in that word position. The determination of greater than or less than is accomplished by noting the bit position from which the first negative signal occurs for a particular word. This signal, indicating a difference in that bit position, combined with the indication of whether the control word has a one or a zero in that bit position will then indicate which is greater.

A contract for a different associative memory using Biax elements has been issued by Wright Air Development Center for delivery in mid-1965. This 2,048 word memory is to achieve cost savings by using integrated circuit sense amplifiers.

Another type of magnetic associative memory using the MALE element has recently been delivered to the Navy. This memory has a capacity of 256 words of 36 bits each. Exact-match search time is 5 microseconds. This associative memory is to be tested with the CP667 computer.

RADC has issued an RFP for a 32,000 word associative memory, is currently procuring a 2,000 word room temperature associative memory (Bilock), and has an associative element (ferro-electric) study under contract. An RFP for another associative element study is to be released soon. Associative processor studies are underway with four contractors. Some of these call for a hybrid approach using an associative memory in conjunction with a random access memory.

17.0 INTEGRATED CIRCUIT ASSOCIATIVE MEMORIES

Several studies of the use of semiconductor circuits to mechanize associative memories have been described.[38] Using discrete semi-conductor components makes the cost of associative memory too expensive to permit more than a few words to be mechanized in this way. However, the rapid advances in integrated circuit techniques may change this significantly so that it may be feasible to mechanize several hundred words of associative memories with integrated circuit techniques. One such approach using a hybrid circuit technology (SLT modules) has been described by Lindquist.[39]

A 256 word associative memory of 72 bits per word has been mechanized with the new SLT modules. This associative memory provides all of the search features normally considered for associative memories (exact match, between limits, etc.). It requires 26,000 SLT modules and costs approximately $2. - $3. per bit. The interrogate time is 200 nanoseconds.

This associative memory was designed for use in controlling the addressing of a scratch-pad memory that is used in conjunction with a large random access memory. The scratch-pad memory would store the more recent data and instruction words read from main memory. The associative memory would store the addresses of these words as an identifier. The usefulness of this approach is based on the assumption that more recently read words are the most likely to be used again in the near future. As a result, having these words available much more rapidly than other words from main memory can significantly speed up the operation of the machine. As a word is read from main memory, it is also stored in the next available location in the scratch-pad memory and its address in main memory is stored in an associative memory location related to the scratch pad location in which the word is placed. As operand or instruction addresses are given, they are searched against the contents of the associative memory while the access is being made to the main memory. Since the associative memory search time and the scratch pad memory access time are much faster than the main memory access time, recently used words in the scratch-pad memory can be located and read out rapidly without waiting for the completion of the access to the main memory.

18.0 ASSOCIATIVE MEMORY USES AND ORGANIZATION

Associative storage techniques can be considered in three major categories from the use or organization standpoint:

1) Associative main internal memory.
2) Hybrid internal memory using small associative memory in conjunction with large random-access main memory.
3) Associative processor in which logical processing elements as well as storage are distributed in associative cells.

By even an optimistic estimate, associative memories will be at least twice the cost of random-access memories of equivalent capacity and circuit (or switching) speed, unless some unforeseen technological breakthrough occurs. This factor is closer to four at present. Consequently, only a few very special applications will be able to justify the cost of an associative main memory. This cost must be justified in terms of the processing speed gained versus that gained by using the same money to buy a faster memory, a larger memory, greater parallelism, or a more sophisticated and capable computer. This cost certainly cannot be justified in an ACDS application.

The use of several hundred words of associative memory in conjunction with a large random-access main memory is much easier to justify. A limited amount of associative memory can greatly facilitate processing in a number of applications, while not increasing the cost of the major part of the storage. In a sense, this hybrid use permits paying a price for the logical capability of the associative memory without increasing the cost of the storage function per se for most of the memory. This use of a small associative memory may well be justified for some major ACDS operations such as target track correlation.

Many advocates of associative memories are now thinking primarily in terms of associative processors. The reasons for this are at least twofold:

1) The cost of logical functions in each cell or each word must be made very cheap if large associative memories are to be feasible; hence, additional logic can be added economically. (This is a particularly tempting argument with respect to cryogenic associative memories where the "overhead" costs are high.)

2) A large associative memory cannot be used efficiently unless the processing functions are distributed with the storage cells.

Associative processors may be an important factor in the computer field at some time in the future. However, the problems to be solved are such that this will not occur prior to 1970. This is particularly true of the economic problems - a fact frequently overlooked in discussions of the advantages of associative processors. This was illustrated by a recent presentation in which a proponent of a particular associative processor approach claimed that a machine mechanized with 1,000,000 groups of associative processing cells would do a 1,000 x 1,000 matrix operation several orders of magnitude faster than an IBM 7090. The comparison did not point out that such a system would require 4×10^9 cryotrons, which is also several orders of magnitude more components than an IBM 7090.

In summary, from a technical standpoint, the hybrid approach of using several hundred words of associative memory in conjunction with a large random-access memory offers some promise for a 1970 era ACDS. However, it is unlikely that the state of the art of technology and cost will permit the use of either large associative main memories or associative processors in naval tactical systems by 1970.

19.0 READ-ONLY MEMORIES

If truly read-only memories are used in a future ACDS, such use will probably be restricted to three functions:

1) Storage of constants and fixed subroutines;

2) Storage of large reference tables;

3) Storage of critical data or programs (e.g. memory protection).

For these types of functions, a true read-only memory that can be changed only manually or mechanically (e.g., by inserting a punched sheet) will be satisfactory, but other functions may require read-mostly memories. These can be considered a special case of read-only memories which are normally only read, but which are capable of being altered electronically. The question of how frequently they may be altered and the difficulty of this alteration is a relative matter that determines whether they should be considered read-only or read-mostly storage. If read-mostly storage devices are used, it will probably be for one of the following functions:

1) Storage of constants and fixed subroutines;

2) Storage of subroutines of micro-operations, if a micro-programmed type of machine organization is selected.

Read-only and read-mostly memories are considered for applications such as those listed above only if they are either significantly cheaper or significantly faster than a more conventional read/write memory. The low cost is a significant factor for the storage of relatively fixed data,while the high speed is important for storage of micro-operations.

Several types of read-only memories have been developed including the card-changeable twistor, capacity-sensing read-only memories, and photographic storage. Available read-mostly memories include the Biax, the Transfluxor, the Piggy-Back twistor, and non-destructive read-out versions of plated wire and planar thin-film memories.

Some of the memory technologies discussed previously, such as the plated wire memory, have a non-destructive-read-out (NORO) characteristic that permits them to be used as read-mostly memories also.

Read-only and read-mostly memories are not considered in greater detail in this report since their use in a future ACDS is not certain and since there are no new developments currently underway that offer significantly higher speeds or lower costs in a 1970 time period than those that will be offered by some of the read/write memories discussed in this report.

20.0 OTHER MEMORY TECHNIQUES

This report obviously has not discussed all types of memory devices or developments. Questions considered in selecting memory technologies for investigation and discussion include whether it is:

1) Batch fabricated;
2) Expected to be a major competitor in 1970;
3) All electronic or magnetic;
4) Suitable for use in a Navy shipboard environment;
5) A type possessing characteristics likely to be required in a 1970 ACDS;
6) Representative of a particular class of memories;
7) Either an established technology or a significant advance, if a new technology.

These considerations lead to the omission of a number of types of memories, such as electromechanical memories where all-electronic or magnetic alternatives now exist - e.g. magnetic-drum or disc internal memories.

APPENDIX A

References

1. "An Integrated Circuit Memory," Cole, R.H. and Smitha, P., IEEE Computer Group Local Symposium on Computer Circuit Technology, Los Angeles, California, December 1 1964.

2. "Integrated-MOST Memory," Schmidt, J.D., verbal presentation at the IEEE Computer Memory Workshop, Lake Arrowhead, California, September 10-11, 1964.

3. "System Design Trends," Rice, R., verbal presentation at the IEEE Integrated Circuit Workshop, Monterey, California, September 17-18, 1964.

4. "Magnetic Films - Revolution in Computer Memories," Chang, C. and Fedde, G., Proceedings 1962 FJCC pp. 213-224, May 1962.

5. "The Future of Thin Magnetic Films," Bittman, E.E., Large Capacity Memory Techniques for Computing Systems, pp 411-420, Macmillan Publishing, New York, 1962.

6. "REDMAN Film Memory," Toombs, D., verbal presentation at the IEE Computer Memory Workshop, Lake Arrowhead, California, September 10-11, 1964.

7. "High Density Magnetic Film Memory Techniques," Crowther, T.S., 1964 Proceedings Intermag Conference, pp 5-7-1 - 5-7-6, April 1964.

8. "Future Developments in Large Magnetic Film Memories," Raffel, R.I., Ninth Annual Conference on Magnetism and Magnetic Materials, Atlantic City, N.J. November 1963.

9. Journal of Applied Physics <u>30</u>, Long, T.R., pp 1235, 1960.

10. "Plated Wire Magnetic Film Memories," Danylchuk, I. and Perneski, A.J., 1964 Proceedings Intermag Conference, 5-4-1 - 5-4-6, April 1964.

11. "The Magnetic Rod - A Cylindrical Thin Film Element," Meier, D.A. and Kolk, A.J., Large Capacity Techniques for Computing Systems, pp. 195-212, Macmillan Publishing, New York, 1962.

12. "A Reliable Very Low Power Plated Wire Spacecraft Memory," Fedde, G.A. and Guttroff, G.H., Proceedings National Electronics Conference, Chicago, Ill., October 1964.

13. "Design of Low Power Circuits for Aerospace Memories," Chong, C.F., Schwartz, E.N., Nelson, C.A., and Guttroff, G.H., Conference Proceedings of 8th International Convention on Military Electronics, Washington, D.C., September 1964.

14. "System and Fabrication Techniques for a Solid State Random Access Mass Memory," Fuller, H. McCormack, T., and Battarel, C., 1964 Proceedings Intermag Conference, pp 5-5-1 - 5-5-4, April 1964.

15. "Laminated Ferrite Memory," Shahbender, R., Wentworth, C., Hotchkiss, K., Li, K., and Rajchman, J.A., Proceedings FJCC, Las Vegas, Nevada, Vol. 24, pp 77-90, November 12-14, 1963.

16. "An Approach Towards Batch-Fabricated Ferrite Memory Planes," Bartkus, E., Brownlow, J., Crape, W., Elfant, R., Grebe, K., and Gutwin, O., IBM Journal of Research and Development, pp 17-176, Vol. 8, No. 2, April 1964.

17. "New Ferrite Structure," Elfant, R., verbal presentation at the IEEE Computer Memory Workshop, Lake Arrowhead, California, September 10-11, 1964.

18. "Computer Memories," Rajchman, J.A., Proceedings IRE, 49, 1, January 1961, pp 115-116.

19. "Glass Memories," Corning Electronics, Bradford, Pa., Technical Brochure, RPP 8/63 5M.

20. "Investigation of a Woven-Screen Memory System," Davis, J.S., and Wells, P.E., Proceedings FJCC, pp. 311-326, Vol. 24, Las Vegas, Nevada, November 12-14, 1963.

21. "Integrated High-Frequency D.C. Amplifier," Breuer, D.R., TRW Space Technology Laboratories Report #9374.1-0001, Redondo Beach, California, July 10, 1964.

22. "Woven Thin-Film Memories," Maeda, H. and Matsushita, A., 1964 Intermag Conference Proceedings, pp 8-1-1 - 8-1-6, April 1964.

23. "The Plated-Wire Memory Matrix," Oshima, S., Futami, K., and Kamibatashi, T., 1964 Proceedings Intermag Conference, pp 5-1-1 - 5-1-6, April 1964.

24. "A Large Capacity Cryoelectric Memory with Cavity Sensing," Burn, L.L., Christiansen, D.A., and Gange, R.A., Proceedings 1963 FJCC, pp 91-99, November 12-14, 1963.

25. "Cryogenic Memories," Burns, L.L. verbal presentation at the IEEE Computer Memory Workshop, Lake Arrowhead, California, September 10-11, 1964.

26. "Ultrasonic Approach to Data Storage," Gratian, J.W., and Freytag, R.W., Electronics, Vol. 37, No. 15, pp 67-72, May 4, 1964.

27. "Digital Computer Peripheral Memory," First, Second, and Third Quarterly Reports (July 1 - September 30, 1963, October 1 - December 31, 1963, and January 1 - March 3, 1964) U.S.A.E.R.&D.L. Contract No. DA 36-039-AMC-03248 (E) prepared by RCA Laboratories.

28. "Review and Survey of Mass Memories," L.C. Hobbs, Proceedings FJCC, pp 295-310, Vol. 24, November 12-14, 1963.

29. "Comparison: Major Types of Mass Memories," L.C. Hobbs, Data Systems Design, Vol. 1, No. 1, pp 18-21, January 1964.

30. "Mass Storage," Hoagland, A.S., Proceedings IRE Vol. 50, pp 1087-1092, May 1962.

31. "A Critical Study of Mass Storage Devices and Techniques with Emphasis on Design Criteria," M. Jacoby, IRE PG MIL, National Winter Convention on Military Electronics, 1962.

32. "Search Memories," IEEE Computer Group Local Symposium, Los Angeles, California, May 26, 1964.

33. "Survey of Present and Potential Search Memory Implementation Techniques," Shohara, S., IEEE Computer Group Local Symposium on Search Memories, Los Angeles, California, May 26, 1964.

34. "Associative Memory System Implementation and Characteristics," McAteer, J.E., and Capobiano, J.A., and Koppel, R.L., Proceedings FJCC, pp 81-92, November 1964, San Francisco, California.

35. "A Cryogenic Data Addressed Memory," Newhouse, V.I., and Fruin, R.E., Proceedings Spring Joint Computer Conference, Vol. 21, pp 89-100, May 1-3, 1962.

36. "Design of a Fully Associative Cryogenic Data Processor," Pritchard, J.P., Jr., and Wald, L.D., 1964 Proceedings of Intermag Conference, pp 2-5-1 - 2-5-4, April 1964.

37. "Theory, Organization, and Performance of a Search Memory," Koerner, R.J., and Scarbrough, A.D., Local Symposium on Search Memory, Los Angeles District of IEEE, May 26, 1964.

38. "Solid State Associative Cells," Lee, E.S., Proceedings of the 1963 Pacific Computer Conference, pp 96-108, March 16, 1963.

39. "Integrated Circuit Associative Memories," Lindquist, H., verbal presentation at the IEEE Computer Memory Workshop, Lake Arrowhead, California, September 10-11, 1964.

Appendix B
COMPONENTS AND PACKAGING

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Appendix B
COMPONENTS AND PACKAGING

1.0 CRYOGENIC LOGIC

Cryogenic logic and switching devices, such as the cryotron have been proposed for computer use for approximately ten years (see Figure 1). During this time, they have not proved superior to semiconductor techniques. Although there is some controversy concerning this, most workers in the field concede that logical components are the least likely application for cryogenic techniques in a computer.[1] Most of the cryogenic research and development work remaining at this time is concentrated on associative and large-capacity memories. Cryogenic logic and switching is considered primarily as an adjunct to a large-capacity or an associative cryogenic memory. The use of cryotrons for the selection tree in a large-capacity cryogenic memory is an example. It is very unlikely that cryogenic logic techniques will be competitive with semiconductor integrated circuit logic for general use in 1970 era naval tactical systems.

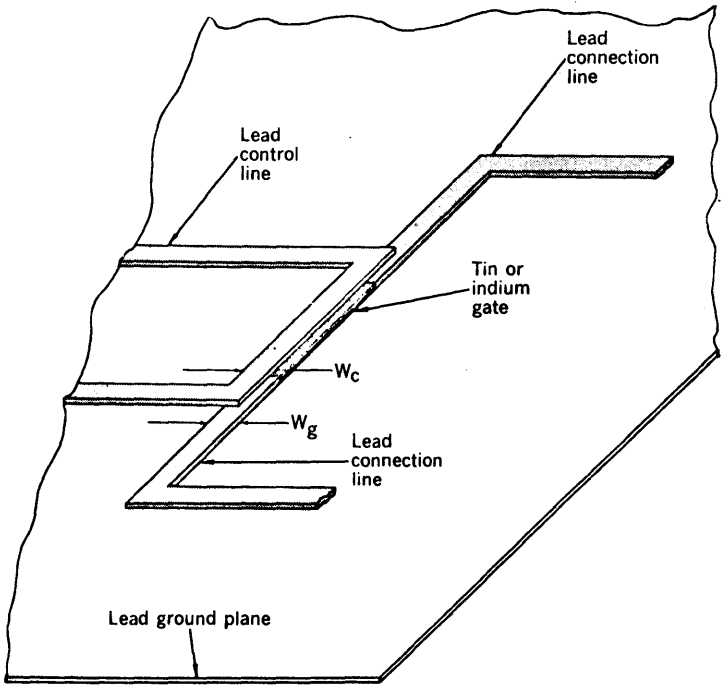


Figure 1. Diagram of An In-Line Cryotron

2.0 FLUID LOGIC

Fluid logic is usually mechanized by hydraulic or pneumatic systems in which small mechanical movements are used to switch the path or flow of a hydraulic or pneumatic medium; e.g., oil or air.[2] The major disadvantages of fluid logic are the relatively slow response time and the size and weight compared to equivalent integrated semi-conductor circuits. The signal is limited to the velocity of sound and the switching time is in the order of 1 millisecond. Fluid logic offers more environmental immunity which is advantageous in adverse environments such as high temperature, electromagnetic fields, and nuclear radiation. It emits no RF radiation and can be highly reliable. Fluid logic may also offer advantages (e.g., lower cost) in applications where the initial input information is in a mechanical form and a mechanical output is required. An example of this is a desk calculator where the initial inputs are key depressions and the outputs are mechanical printing operations.

Two or three companies have built small demonstration calculators using fluid logic. Fluid logic is amenable to batch-fabrication in the form of injection molding techniques. Fluid logic may be applicable to some shipboard functions, such as weapon direction and ship's control. However, fluid logic will not be competitive with integrated circuits for logical functions in the central processor and auxiliary equipment for the information processing portion of a 1970 ACDS.

3.0 OPTICAL LOGIC

There has been considerable interest in the possibility of using optical logic devices in computers because of the inherent speed theoretically possible when working with light. Recent developments in fiber optics and lasers have accelerated this interest.[3, 4, 5] Some of the characteristics of lasers that make them attractive for computer use are:

1) The output is coherent and monochromatic
2) Very-high frequencies are possible
3) The beam is highly collimated
4) High-power intensity
5) Capable of either continuous or pulse operation.

Fiber optics have the capability of conducting light around curved paths, and hence they offer interconnection possibilities similar to the use of wires in carrying electrons. However, lasers and fiber optics will see use in memories and display areas before they will be successfully used as logical components. Optical logic techniques offer promise for the long-range future, but they will not be feasible for use in a 1970 system.

4.0 SPECIAL SEMICONDUCTOR ELEMENTS

A number of unique or special purpose semiconductor devices have been proposed for use in computers. Of these, the most serious consideration has been given to the tunnel diode.[6,7] Tunnel diodes have been proposed for high-speed, small-capacity memories as well as high-speed logical components. Of the various approaches to "kilomegacycle circuits" tunnel diodes are considered the most practical, although the rates of approximately 200 - 500 megacycles at which they have been used do not quite fall in the kilomegacycle range. Soon after they were introduced four or five years ago, tunnel diodes were considered by many people to be an exciting solution to the high-speed computer circuit problem. However, difficulties in working with a two terminal device such as the tunnel diode have seriously dampened this enthusiasm. The problems of interconnection techniques for tunnel diode circuits operating at a frequency of several hundred megacycles have proved to be difficult. Systems become very expensive as a result of the discrete mechanical configurations required for interconnections and shielding. Although tunnel diode logic circuits are feasible, they will not be competitive with integrated circuit techniques for use in an ACDS in 1970 since such high speeds are not required.

5.0 ALL-MAGNETIC LOGIC

Magnetic elements can be used in a digital system for logic as well as for storage functions. However, magnetic elements have not enjoyed the widespread use as logical elements that they have as memory elements. This is, of course, due to the difference in the nature and requirements of memory components and logic components. A single word location in a memory is addressed at one time, and a large driving current, low sense signal, and destructive readout are acceptable. On the other hand, for a logic element it is necessary to provide some form of gain, and to sense the state of the device without changing it.

A number of applications for all-magnetic logic and a number of types of logical configurations and elements have been described in the literature.[8, 9, 10, 11] (See Figure 2) All-magnetic logic offers several distinct advantages, including:

1) High reliability
2) Radiation resistance
3) High temperature operation
4) Low standby power
5) Non-volatile
6) Low power required at low frequencies
7) Cost (in some cases).

The major disadvantages in the use of all-magnetic logic have been the inherent slow speed and the lack of a steady state output indication. Most work on all-magnetic logic has been in the 5 to 10 kc range. Shift cycle rates of 150 to 200 kc may be possible by 1970, but speed is probably limited to a few hundred kilocycles at most. The speed is limited by switching time, heating, and current requirements. For most applications all-magnetic logic circuits have not proved themselves sufficiently advantageous with respect to either cost or performance to encourage their use in place of the simpler and more common semiconductor circuits.

The characteristics of all-magnetic logic are ideal for certain applications such as an onboard computer in a deep space probe. In this application, very low speeds are acceptable and radiation resistance, low standby power (since the spacecraft is inoperative for the long cruise periods of the mission), and high reliability are important. For some shipboard functions such as peripheral equipment, weapon direction systems, and ship's control systems where high speeds are not required, magnetic logic may prove a good choice. Applications of all-magnetic logic in shipboard systems were analyzed in a recent study.[12]

However, for tactical data system applications, all-magnetic logic will not be competitive with semiconductor integrated circuits. The radiation resistance characteristic and the high reliability would be important if future developments prove that integrated circuits are not as reliable as magnetic logic. However, it

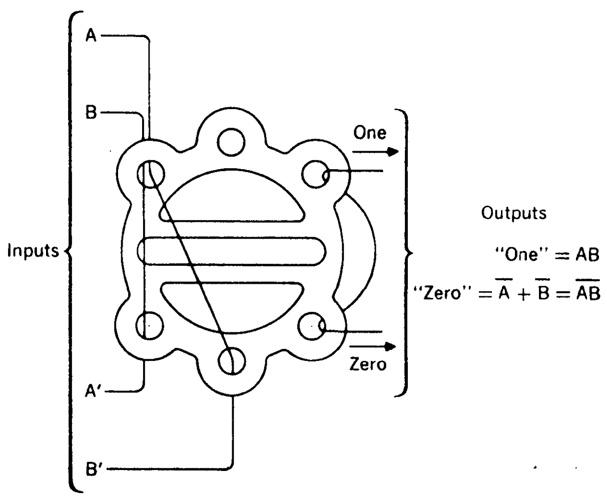


Figure 2. Multiaperture Ferrite Core Connected As An AND Gate for Magnetic Logic

appears that semiconductor integrated circuits will equal or approach the reliability of all-magnetic logic, and that the cost of these elements will be less than that for the discrete elements used in magnetic logic. The speeds of all-magnetic logic are not sufficient for the central processor since the multi-apertured devices found most suitable for magnetic logic have been limited to rates of a few hundred kilocycles per second. It is possible that new developments in thin-film integrated magnetic circuits in the next few years may enhance the performance and significantly reduce the cost of all-magnetic logic but it is unlikely that this will occur by 1970.

6.0 SEMICONDUCTOR INTEGRATED CIRCUITS

Integrated semiconductor circuits are by far the outstanding candidate for the logical mechanization of a 1970 naval tactical data system. Integrated circuits have been proved feasible and successful, and are currently being used in several military computers. The first integrated circuit demonstration computer in 1961 showed a 10 to 1 improvement in reliability over a comparable discrete component computer.[13] Hybrid integrated circuits are used in the Remington Rand CP667 computer for NTDS and in the new IBM System/360 commercial computer.[14, 15] Monolithic integrated circuits have been used in the Autonetics Monica Computer, by MIT in the Apollo computer, by AC Spark Plug in the MAGIC computer, and are being used by Litton in the computer for the F-111 (TFX).[16] Estimates of the military use of integrated circuits range from 40 to 50% of all military electronics in 1970 to approximately 75% in 1973.[17, 18, 19] Since digital circuitry is more adaptable to integrated circuit techniques, the estimates are even higher for computer and data processing equipment - approximately 70% by 1970. Actually, this figure will probably be closer to 90% (not including the memory) for new digital equipment designed to become operational in 1970.

Integrated circuits are not basically new components in the sense that lasers are, but rather they represent radically new methods of fabricating and packaging semiconductor circuitry. The reduction in the number of discrete components, resulting from fabricating complete circuits as a single component, offers significant advantages in terms of reliability, cost, and size. Batch-fabrication of volume quantities of integrated circuits will result in significantly lower costs than is achieved by the present printed circuit and hand wiring of basic components and circuit modules.

Based on considerations of reliability, cost, size, weight, and environmental conditions, it is reasonable to expect that integrated circuits will account for almost all of the logical components in a 1970 shipboard or ground-based military system.

There are five basic types of integrated circuits although these are sometimes called by different names and in some cases grouped differently. The term hybrid is particularly confusing since it is applied to thin-film passive components with discrete active components and to thin-film passive components with monolithic active components. These five types are:

1) Hybrid discrete active and thin-film (or thick-film) passive
2) Monolithic
3) Hybrid monolithic/thin-film
4) Active thin-film
5) Metal oxide semiconductor (MOS)

There are good reasons for the development and use of each of these. These reasons are discussed in greater detail in this Appendix.

6.1 HYBRID DISCRETE/THIN-FILM (or THICK-FILM) CIRCUITS

In this type of circuit, passive elements, such as resistors and capacitors, are printed on a ceramic or glass substrate by either vacuum deposition of thin-film elements, or by printing of thick-film elements in a process similar to silk screening.[15] Discrete (but unpackaged) active components are connected to printed or deposited interconnections on the same substrate (see Figure 3). The combination is then packaged as a single unit. This is an interim type circuit that was developed before monolithic and hybrid monolithic circuits were technically feasible for large scale production. Their main area of application is where a high ratio of passive to active components is required.

This type of circuit offers the advantage that the passive components can be made cheaply with tightly controlled tolerances. Relatively large values of capacitance can be fabricated and resistance values can be maintained within about one per cent.

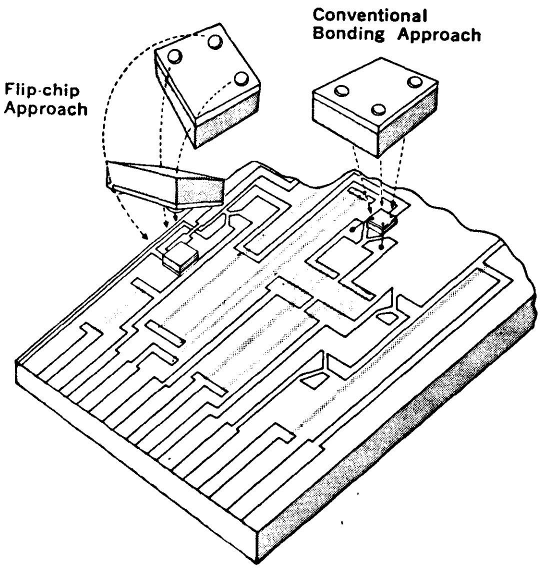


Figure 3. Discrete Devices Being Placed Into Position On A Substrate By Conventional and Flip-Chip Methods

The values and tolerances available through the use of thin-film passive components is illustrated by a table of characteristics presented in a recent talk by McLean.[20]

Table 1

CAPACITORS AND RESISTORS

CAPACITORS			
Dielectric Material	Dielectric Constant	Capacitance (uf/cm^2)	Precision
$Ta_2 0_5$	22	0.10	$\pm$ 3 %
Si0	6.8	0.02	+ 3 %

RESISTORS		
Material	Ohms/Square	Adjustability
Nichrome	10 - 300	1 %
$Sn0_2$	25 - 400	1 %
CrSi0	30 - 1,000	1 %
PdAg Glaze	500 - 10,000	1 %
Ta(N)	5 - 200	0.1 %
Td(O)	50 - 1,000	0.1 %

As a result, this approach is more adaptable at present to linear circuits, such as differential amplifiers and analog circuitry, than is the monolithic integrated circuit. This type of circuit also has the advantage of inherent isolation - no interactions and parasitic capacitances between the different elements as is the case for the monolithic integrated circuit. It has the disadvantage that the active elements must be handled as discrete elements. The reliability is probably not as high, due to the handling of the active elements and the soldering of these elements to the printed interconnections on the substrate. The cost will be higher and large arrays of logical circuits cannot be batch-fabricated.

One commercial computer manufacturer uses five different families of hybrid logic circuits with propagation delays of 200, 20, 9, 6, and 2 nanoseconds.

With the possible exceptions of high-power or high-voltage circuits, it is believed that this type of circuit will phase out before 1970 with preference being given to the second and third type of integrated circuits discussed next.

6.2 MONOLITHIC INTEGRATED CIRCUITS

This type of circuit is completely integrated.[21] Active elements (e.g., resistors and capacitors) necessary to perform a specific circuit function or set of circuit functions, are fabricated by a series of diffusion processes in a single silicon chip. Interconnections are fabricated by vacuum deposition processes on top of the diffused components (see Figures 4 and 5). This circuit has the advantage that all components in the circuit are made during the same series of operations, and that multiple circuits of this type can be batch-fabricated in a single set of operations.

This type of circuit should ultimately be cheaper to fabricate and more reliable due to the ability to make all interconnections by vacuum deposition processes. It is more adaptable to the batch-fabrication of large interconnected arrays such as a major segment of an arithmetic unit. There have been three major disadvantages with respect to monolithic integrated circuits to date:

1) The interaction between semiconductor elements diffused in the same silicon chip and the resulting parasitic capacitances.

2) Difficulty in maintaining resistor tolerances better than approximately 20%.

3) Difficulty in fabricating capacitancies of more than a few micro-microfarads.

There is no easy method of fabricating inductors, but fortunately this presents no problem for digital circuits. The yield of this type circuit has not been as satisfactory since any individual bad element makes the entire circuit bad. It is difficult to get accurate information on the yield experienced by manufacturers, but estimates range from approximately 1% to 20% for present high-grade military type circuits with yields

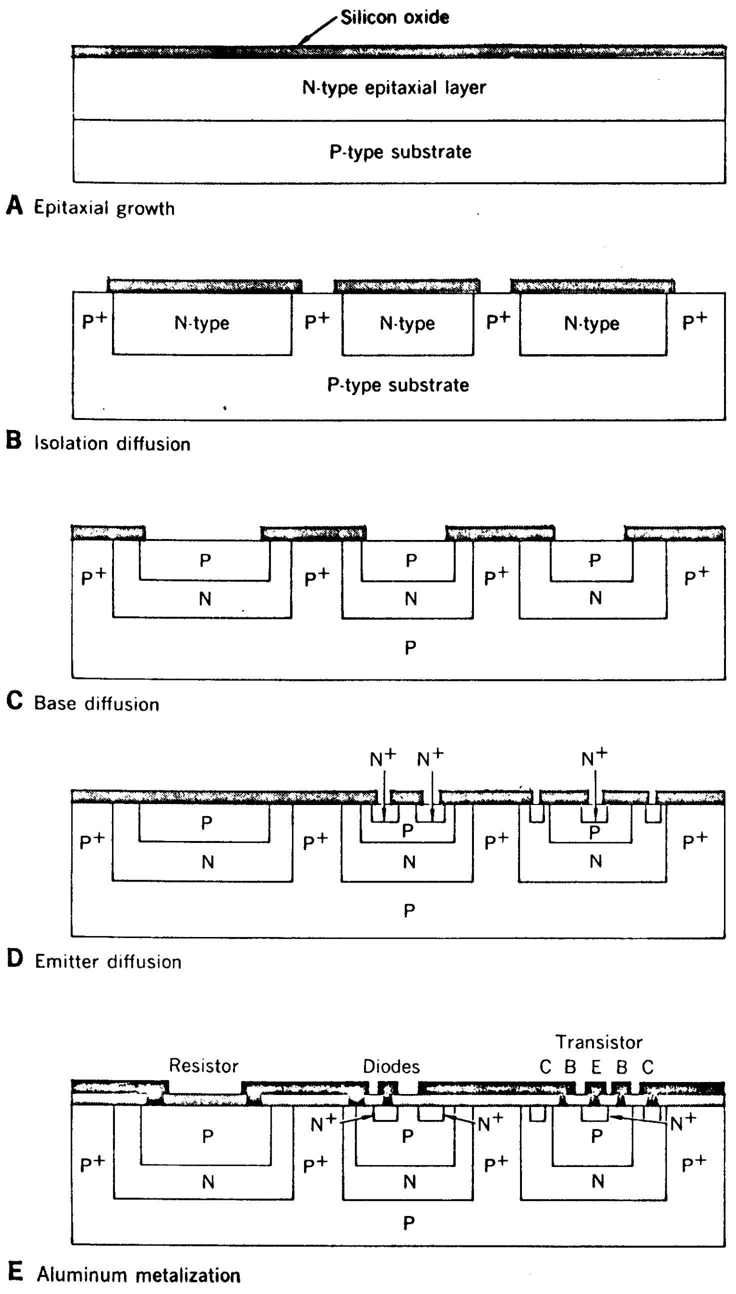


Figure 4. Sequence of Photoresist and Diffusion Processes

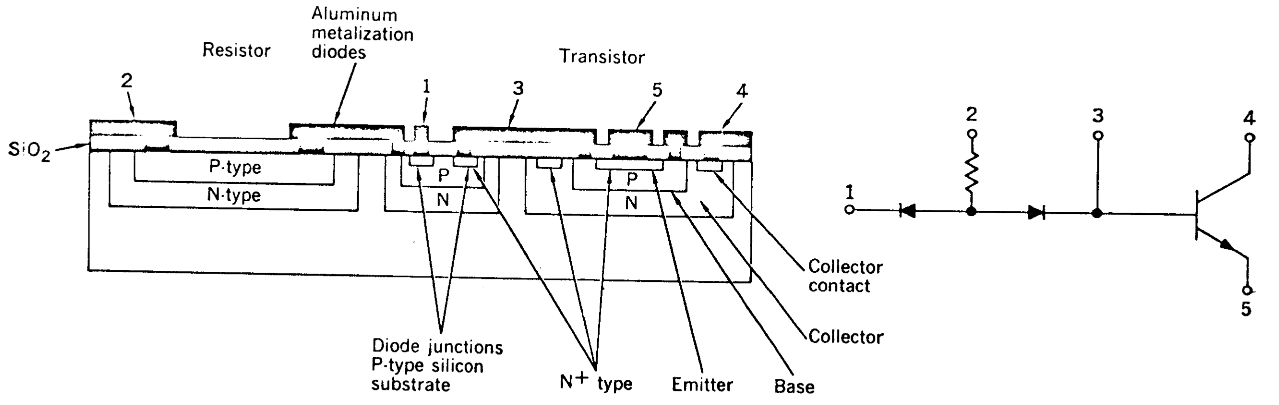


Figure 5. Cross-Sectional View of a Typical Integrated Circuit

of 50 to 90% predicted for the future. Propagation delays at 2 to 5 nanoseconds are available today. Present state-of-the-art permits approximately 30 transistors and 60 to 80 total components on a single chip. These figures should double within a year and increase to several hundred by 1970.

Monolithic integrated circuits are well suited to digital applications where component values are not as critical, but they are not satisfactory for most types of linear circuits at present because of the interactions and the difficulty in controlling tight tolerances. Intensive research and development efforts are being expended on the problems of monolithic integrated circuits,and rapid progress is being made. Several companies have reported success in isolating the components in a monolithic integrated circuit to reduce the parasitic capacitance.[22, 23] Two different approaches to component isolation are illustrated in Figures 6 and 7. This should increase the speed of circuits to this type and permit their application in certain types of linear circuits. It is anticipated that this type of circuit will be the major integrated circuit technique used in digital applications, including shipboard data systems, within the next few years.

6.3 HYBRID MONOLITHIC/THIN-FILM CIRCUITS

In this type of circuit active elements, and possibly certain passive elements, are diffused into a single silicon chip as in the preceding case.[28] However, additional thin-film passive elements as well as interconnections are fabricated on top of the silicon chip by vacuum deposition processes (see Figures 8 and 9).

This technique combines many of the advantages of the first type of hybrid circuit discussed with the advantages of the completely monolithic integrated circuit. Tight tolerances on resistors and capacitors can be maintained and relatively large values of capacitance fabricated while not handling discrete components. Thin-film resistor and capacitor characteristics listed for the first type of hybrid circuit discussed previously are also applicable to this type.

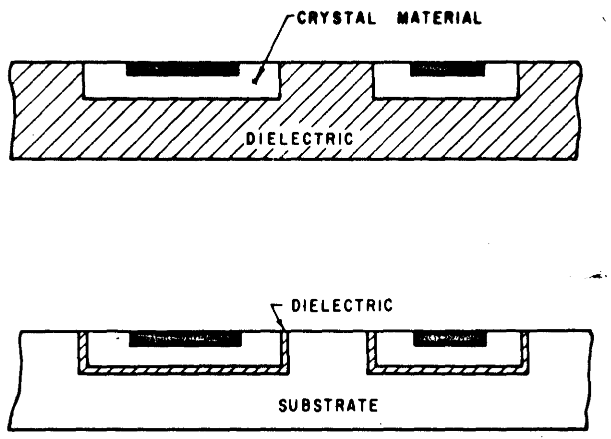


Figure 6. "Ideal" Isolation Technique and Practicable Dielectric Technique

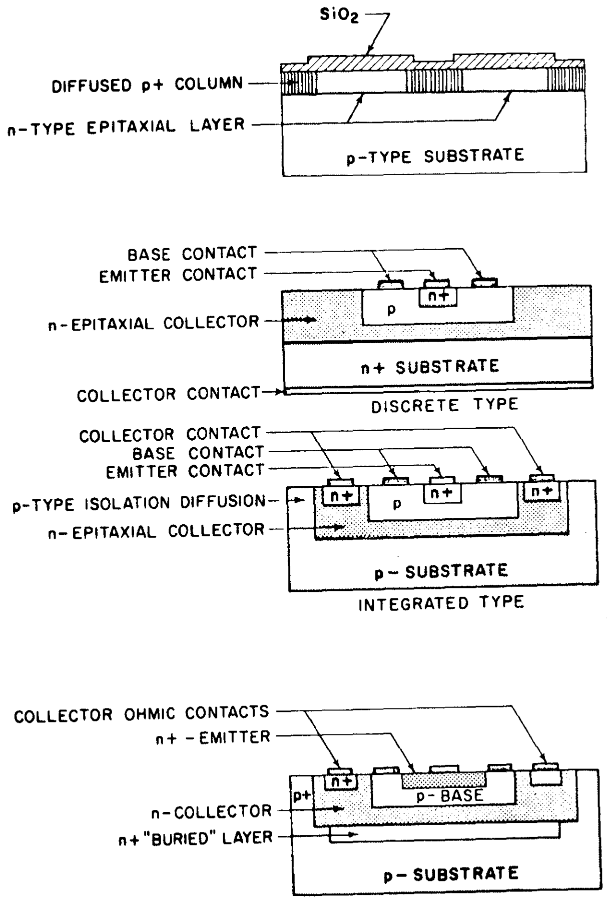


Figure 7. Channel Diffusion or Junction Isolation Techniques

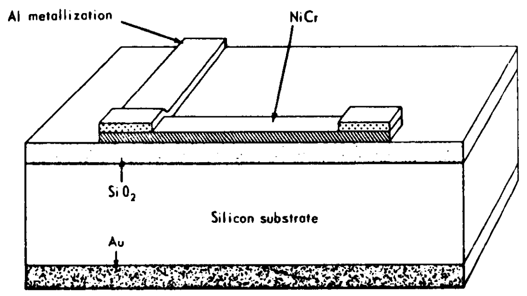


Figure 8. Thin-Film Resistor

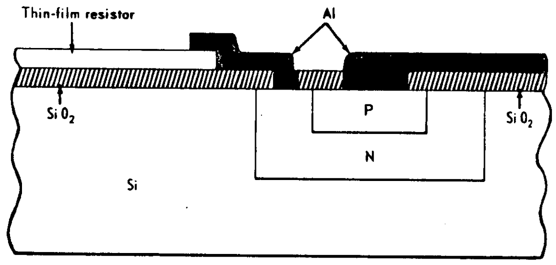


Figure 9. Hybrid Thin-Film Resistor and Transistor

Batch-fabrication of arrays of elements and circuits in a single set of processes, and higher reliability resulting from vacuum deposited interconnections are achieved. With this type circuit, it is possible to obtain many of the cost and reliability advantages of the completely monolithic integrated circuit while fabricating higher quality components. The fabrication of linear integrated circuits, such as differential amplifiers and other analog type circuits, is facilitated. This is illustrated by a "high frequency DC amplifier" described recently by Breuer.[24] Several hundred thousand ohms of resistance and several hundred micromicrofarads of capacitance can be obtained on an integrated circuit using this hybrid approach. Resistor tolerances of better than 10%, and capacity tolerances of two parts per million can be obtained relatively easily. Higher resistor tolerances can be achieved by "trimming" the resistors during the test operation. A comparison of resistor and capacitor characteristics available through diffusion in monolithic silicon circuits and through deposition of thin-films in hybrid circuits has been presented by Hogan and are reproduced in Tables 2 and 3.

Monolithic integrated circuit application to linear circuits has not progressed as far due to the problems with interaction between components, parasitic capacitance, and the difficulty of fabricating larger values of capacitance. As a result, most of the success in integrating linear circuits has been with hybrid integrated circuits. Differential amplifiers and other types of analog computer circuits have been difficult to mechanize with monolithic circuits for these reasons. A good deal of effort has been expended on certain types of linear circuits for computers - particularly sense amplifiers for memories.[24, 26]

Satisfactory memory sense amplifiers in integrated circuit form will be available within a year. This will have a significant effect on memory costs for large capacity memories as discussed in the memory section of this report. Magnetic memory drive circuits have been difficult to mechanize in integrated circuit form because of the power handling requirements. The solution to this problem is not as close as the solution to the sense amplifier problem. Hybrid circuits will be used along with the completely monolithic integrated circuit for the next five to eight years at least. Monolithic integrated circuits will be used wherever possible, with the hybrid monolithic/thin-film circuit being used to complement and supplement them where higher tolerance components or larger values of capacitance are required. In other parts of this report,

Table 2. Integrated Circuit Resistor Characteristics

Parameter	Monolithic Silicon Components		Thin Film Components	
	Diffused p	Diffused n	Nichrome	Cermet
Sheet resistance, ohms/square	100-300	2.5	40-400	100-1000
Resistance per substrate area*, ohms/mil^2	50-150	1.25	20-200	50-500
Temperature coefficient, ppm/^{o}C	+2800 to 1500	+100	$\pm$100 adjustable to within $\pm$10%	-55
Power dissipation per active resistor area**, mW/mil^2	3	3	2	
Maximum voltage, volts	20	6	-	-
Tolerance for high yield, per cent ***	$\pm$20	-	$\pm$8	$\pm$8
Distributed capacitance****, pF/mil^2	0.2	0.6	-	-

Prepared by C. Lester Hogan

* Assumes 1-mil-wide stripe, and 1-mil-wide spacing.
** Depends upon package and heat sinking.
*** Tighter tolerances available at higher cost.
**** Negligible using recent EPIC techniques.

Table [illegible]. Integrated Circuit Capacitor Characteristics

Parameter	Monolithic Silicon Components				Thin-Film Components	
	Single Diffused p-n Junction	Double Diffused p-n Junction	Thermally Grown SiO_2	SiO	Boroalumino-Silicate Glass	Tantalum Oxide
pF/mil^2	0.1 at $V_{bias} = 0$	1.0 at $V_{bias} = 0$	0.25	0.01	0.4	2.5
V_{max}, volts	30	6	50	50	50	20
Dissipation factor, %,						
1 kc/s	10	100		2.5	0.2	0.8
1 Mc/s			0.7	0.7	0.2	0.3
10 Mc/s			2.0			
Capacitor temperature coefficient, ppm/°C	Low	Low	Low	± 200 ± 50	+115	+400
Capacitor voltage sensitivity	$V^{-1/2}$	$V^{-1/2}$	0	0	0	0
Polar	Yes	Yes	No	No	No	No
Shunt capacitance, per cent component value	25	25	18	0	0	0
Leakage current at 5 volts, A/pF	10^{-9}	10^{-9}	10^{-9}			

Prepared by C. Lester Hogan

these two types are discussed interchangeably as "integrated circuits" with the assumption that monolithic techniques will be used where possible with thin-film passive components deposited on top of the chip where necessary to achieve required component values or tolerances. Unless the isolation problem in the monolithic integrated circuit is completely overcome, the hybrid monolithic thin-film approach may also be used to permit higher speeds (below 2 nanoseconds propagation delay). This will be the second major type component used in 1970 ACDS.

6.4 ACTIVE THIN-FILM ELEMENT CIRCUITS

In this type of circuit, both the active components and the passive components are fabricated by vacuum deposition of thin-film elements. (See Figure 10.)[27,28] Cadmium-sulphide thin-film circuits represent the most promising approach to thin-film active circuits at present. Predictions concerning the date at which active thin-film elements will become feasible vary widely -- from "almost immediately" to "not less than five years". The longer estimate is probably the more accurate one with the possible exception of a related but somewhat different approach that requires a single crystal passive substrate such as sapphire. An insulated-gate field-effect transistor has been fabricated in this way by depositing or growing silicon on a single crystal passive substrate (sapphire), depositing an oxide insulator such as silicon monoxide, and depositing aluminum plates for connections and distributed capacitance.[29] (See Figure 11.) Because of the majority carrier nature, field effect transistor devices have been expected to offer better radiation resistance characteristics. This device is attractive because of its simplicity and the fact that it is quite amenable to batch-fabrication of large interconnected arrays with minimum interaction.

It is doubtful that the use of thin-film active elements, such as thin-film transistors, in a system will be feasible by 1970.

6.5 METAL-OXIDE-SEMICONDUCTOR (MOS) CIRCUITS

In this type of device a single diffusion process is required in a semiconductor (usually silicon) chip.[30] This is a field-effect device in which metal electrodes are deposited on two slightly separated diffused areas (e.g. N type material for a P type chip), an insulator (e.g. silicon monoxide) is deposited on the chip between the two diffused areas, and the third metal electrode is deposited on the insulator (See Figures 12 and 13.)

Although MOS devices have advantages from the fabrication standpoint, they cannot match the 5 ns speeds quoted for bipolar transistors. Present work with field effect devices indicate potential speeds in the range of 50 - 100 ns. Most of the success with field-effect devices to date has been with the metal-oxide-semiconductor (MOS.) Since significantly fewer processing steps are involved,

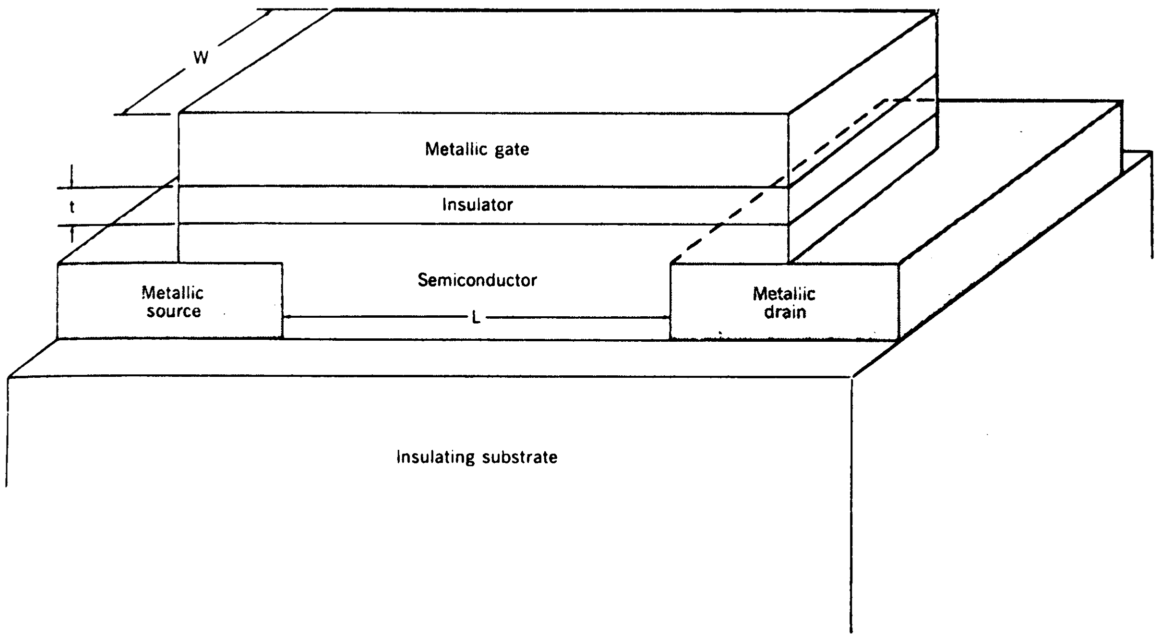


Figure 10. Schematic Diagram of a Thin-Film Transistor

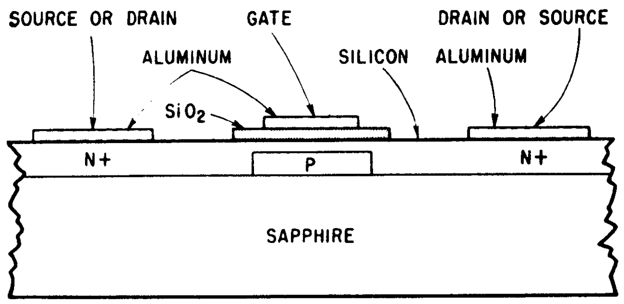


Figure 11. Silicon-on-Sapphire Field-Effect Transistor

the fabrication of large arrays of elements with reasonable yields is more feasible. In contrast to bipolar transistors, MOS elements are high impedance devices. The following advantages have been cited for MOS devices:

1) There is no problem with "current hogging" in DCTL type circuits.

2) High fan-out can be obtained.

3) Complementary symmetry circuits can be made on the same substrate with only one extra diffusion process.

4) They are simple to fabricate and large arrays of circuits can be fabricated.

Complementary symmetry permits one MOS transistor to be used as a load switch for the other.[31] This increases the reliability and radiation resistance since shifts in the characteristics of the devices have a lesser effect if the load line is the characteristic of another MOS rather than a straight line resistor type load. As the characteristics of one MOS change due to external conditions, the characteristics of the other change also, resulting in a lesser circuit effect from the net change. Complementary symmetry devices have been fabricated by diffusing areas of N type material into the P type substrate, then diffusing P type material into the N type areas.

One manufacturer has reported work on a small low-cost random-access memory array mechanized with MOS elements,which is significant in that a single chip contains an order of magnitude more components than other integrated circuits now being manufactured.[32] The efforts to date have produced a complete memory array of 16 four-bit words on a single silicon chip 115 mils by 145 mils in size (approximately ten times the area of conventional integrated circuit). About 30 chips can be fabricated on one silicon wafer with present yields of four to five good chips per wafer. The total read/write cycle time for this device is four microseconds -- a two microsecond read time and a two microsecond write time. The relatively slow speed of four microseconds results from the fact that the MOS is a high impedance device. They have estimated that a speed improvement of an order of magnitude is probably the best that can be expected with this approach. However, on a laboratory basis, 150 - 200 nanosecond MOS type storage elements similar to those have been fabricated by another company by taking advantage of complementary symmetry.[31]

Most companies have experienced a surface instability in metal oxide semi-conductors that is very temperature dependent. Two companies have reported that this surface instability, caused by charge motion at the metal-to-oxide interface (charge leakage around the gate), can be overcome by phosphorous treating of the surface.

MOS integrated circuits have recently received intensified interest. The number of companies offering discrete MOS elements has doubled during the past year. Although the technology is not as well established and the speed capability is more limited, MOS integrated circuits will probably be the major competitor for monolithic and hybrid monolithic/thin-film integrated circuits for a 1970 system - particularly where large arrays of interconnected circuits on a single chip are applicable.

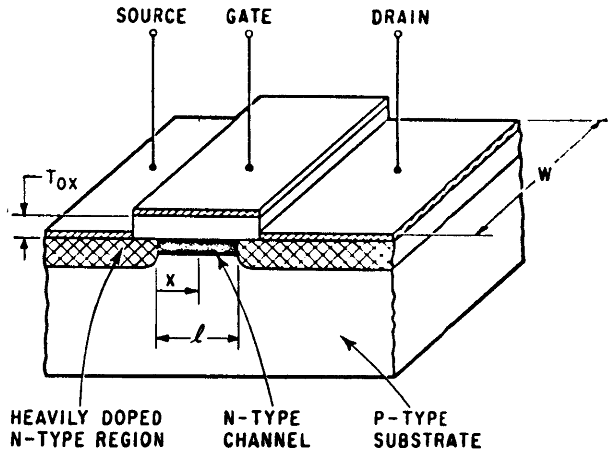


Figure 12. Cross-Sectional View of an Insulated-Gate MOS Field-Effect Transistor

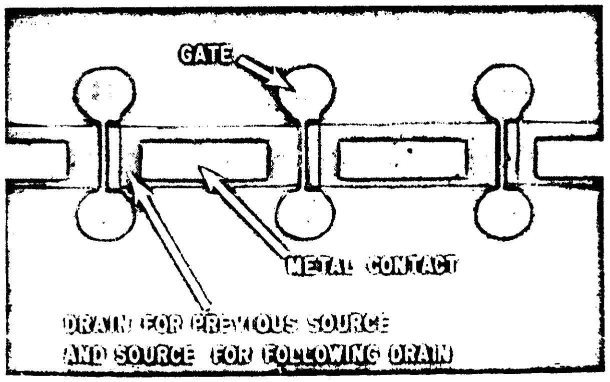


Figure 13. Ladder Array of Enhancement Field-Effect Transistors

7.0 ENVIRONMENTAL CONDITIONS

Current semiconductor technology is meeting the environmental requirements for shipboard applications. Semiconductor integrated circuits will be able to meet all of the necessary environmental conditions for shock, vibration, temperature, and humidity for use in a 1970 ACDS. Kilby has reported on tests of integrated circuits to military specifications saying, "Severity levels several times greater than those of MIL 19500, the most applicable specification, are required to cause failure."[33] He presented a summary of the results of an environmental test program that is reproduced as Table 4. These test results show the ability of integrated circuit technology to meet military specifications for shipboard equipment.

Temperature affects semiconductor characteristics and reliability, but these effects can be kept within required limits in the operating range of $-55^{\circ}C$ to $+125^{\circ}C$. An indication of the effect on speed is given by a reported increase in propagation delay for a particular type of monolithic integrated circuit from 12 nanoseconds at $25^{\circ}C$ to 20 nanoseconds at $125^{\circ}C$.[34] A plot of propagation delay versus temperature in the same report by Powers is reproduced as Figure 14. Such effects are accommodated by specifying the worst case ($125^{\circ}C$) delay for the circuit, and by designing equipment accordingly.

Semiconductor monolithic integrated circuits using planar transistors will have essentially the same characteristics with respect to generation of electromagnetic interference and susceptibility to electromagnetic interference and radiation as the transistorized discrete component circuits used to mechanize shipboard computers at present. There is a good possibility that integrated circuits using MOS or other field-effect type devices will be more resistant to nuclear radiation because of their majority carrier nature.[30, 31] However, this has not been fully proved by adequate amounts of testing, and some controversy has arisen as to the extent of this radiation resistance. Recently, Hughes and Giroux of the Naval Research Laboratory questioned this theory and ran radiation effect tests on MOS devices that indicate some radiation damage.[35] They state, "The results of exposing commercial MOS transistors to radiation indicate that caution must be exercised when utilizing this type of device for space applications." However, this is not a serious problem for a 1970 ACDS since the equipment will not be subjected to a more severe nuclear environment than will the personnel on the ship.

Prepared by
J.S. Kelly

Table 4. Semiconductor Network Environmental Evaluation

Test	Number Tested	Conditions	Fail
Thermal Shock	30 *	5 Cycles -0 C to 100 C	0
		45 Cycles	0
		135 Cycles	1
		270 Cycles -0	0
Temperature Cycling	73 *	5 Cycles -55 C to 125 C	0
	40	45 Cycles	0
	40	135 Cycles	0
	20	160 Cycles	0
Shock	73 *	1,500 G, 0.5 MS, Total 20 Blows,	0
	40	2,000 G 4 Planes	0
	50	3,000 G	0
	40	3.500 G, 0.2 MS	0
Vibration Variable Frequency	83 *	20 G, 100-2000 CPS, 3 Planes	0
	50	30 G	1
	40	40 G	0
	50	50 G	0
Vibration Fatigue	50 *	20 G, 60 CPS, 96 Hours	0
	50	30 G 9 Hours	0
	40	40 G 9 Hours	0
	40	50 G 9 Hours	0
Constant Acceleration	68 *	20,000 G	1
	29	25,000 G	1
	28	30,000 G	1
	32	35,000 G	1
	26	40,000 G	2
	24	45,000 G	1
	28	50,000 G	1
Moisture Resistance	10 *	10 Cycles	0
	10	20 Cycles	0
	10	30 Cycles	1
	9	40 Cycles	1
	8	50 Cycles	1
Salt Atmosphere	20 *	24 Hours, 5% solution	0

* Tests at MIL 19500 Level

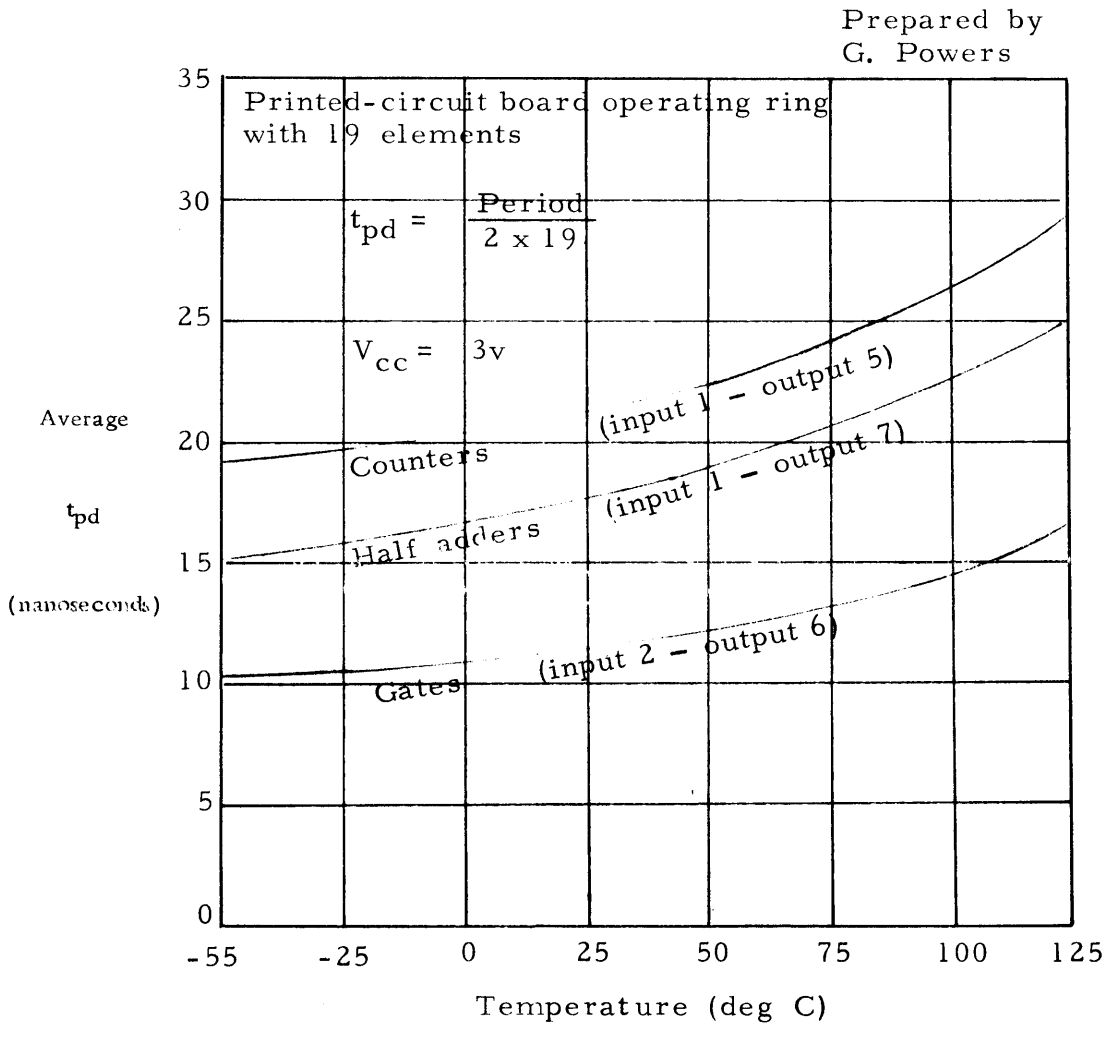


Figure 14. Propagation Delay Versus Temperature

8.0 BATCH-FABRICATION IMPLICATIONS AND CONSIDERATIONS

Throughout the discussions of components and packaging and in discussions elsewhere of memories for 1970 systems, the concept of batch-fabrication has been emphasized. The development of this technology is the key to cost, reliability, and maintainability improvements in a 1970 ACDS. The integrated circuit technologies discussed are the leading and most highly developed forms of batch-fabrication in the computer field.

Batch-fabrication techniques for computer circuits, memories, and display equipment have progressed much faster in the past two years than had been generally anticipated. Within the next five years, these techniques will have a dramatic impact on the computer field. It is likely that the computer field will also supply the major impetus for accelerated development of batch-fabrication techniques since digital equipment and systems are most amenable to the repetitive use of relatively standard circuits. Although many development projects are underway on individual batch-fabrication techniques, very little work on the effect of these techniques on machine organization and on the consolidation of different techniques into a batch-fabricated computer system design has been reported.

Batch-fabrication techniques are frequently included under the general term "micro-electronics", but this is misleading. "Micro-electronics" places emphasis on size reductions rather than on the fabrication of large numbers of circuit elements in a single set of processing operations. Batch-fabrication offers additional significant advantages - reduced cost, improved reliability and maintainability, and reduced power requirements.

Full advantage can be taken of advances in batch-fabrication techniques only when their effect on machine organization and systems design is better understood. Only a small part of the potential for batch-fabrication processes will be realized if individual techniques, such as integrated circuits, are used to directly replace components or subassemblies in existing types of equipment and system designs. It is necessary to re-orient the thinking of computer designers when working with batch-fabrication techniques.

Methods of machine organization to provide more highly repetitive logical organizations are needed. Certain portions of present computers, such as successive stages in the adder of a parallel machine, are repetitive; but other portions, such as the control unit, tend to have unique non-repetitive logical configurations.

If large arrays of logical circuits are to be fabricated in a single set of processing operations with acceptable yields, techniques will be required for easily eliminating bad elements or for permitting the array to work with a limited number of bad elements. This is done at present in the fabrication of some memory arrays where redundant rows and columns are provided to permit substitution for rows or columns that contain bad elements. However, there are no equivalent techniques for logical arrays in use in present systems.

Criteria used previously to select efficient designs (e.g. minimizing the number of flip-flops or the logical terms in a Boolean equation) will not be valid. Since adding an additional flip-flop to a single silicon chip containing an array of integrated circuits will have a minor effect on the cost of the array, flip-flop and gate counts will be much less important than the number of circuit packages required. Emphasis will be placed on reducing the number of packages - even at the expense of significantly increasing the complexity of each package.

Machine organization techniques using "inefficient logic" in order to facilitate the fabrication of larger arrays of logical elements in a single set of processing operations will be important in future systems.

Batch-fabrication of logical circuitry and memories will make it possible to fabricate sophisticated computers and central processors in very small packages. As a result, the imbalance between input/output and peripheral equipment on the one hand and computers and central processors on the other hand will be intensified. Since there is little hope at this time for equivalent improvements in the size and cost of peripheral equipment, the central processor in a computer system designed a few years in the future may be mounted in the bottom of a tape unit (or of one of a large number of other peripheral devices). Hence, for a shipboard data system, the overall effect of batch-fabrication techniques will be limited by the input/output and the complexity, and possibly the inefficiency, of the central processor and memory to minimize input/output and peripheral operations.

These are but a few of the many questions that need to receive more intensive investigation to assure the optimum utilization of batch-fabrication techniques in future digital equipment and systems.

9.0 MAJORITY LOGIC AND VOTING REDUNDANCY

9.1 MAJORITY LOGIC

In discussing majority logic, it is necessary to consider the context in which the term is used. Strictly speaking, majority logic is a special case of threshold logic in which a logical element generates one logical output if over 50% of the inputs are present and generates the opposite logical output if less than 50% of the inputs are present. There should be an odd number of inputs to avoid ambiguity. The threshold level is set at $\frac{n+1}{2}$ where n is the number of inputs.

This is in contrast to a conventional Boolean logical element in which the threshold is set at either 0% or 100%. In the zero threshold instance, one logical output of a conventional Boolean gate is generated if none of the inputs are present and the opposite logical output if any one or more of the inputs are present. In the 100% threshold case, one logical output is generated if all of the inputs are present and the opposite logical output if one or more of the inputs are missing. Any arbitrary function can be realized with majority logic elements and conversely any majority logic element can be simulated by a network of conventional Boolean logical elements. Hence, any Boolean logical design can be converted to majority logic by the proper manipulation of the Boolean expressions.

The use of majority logic has been proposed primarily for circuits, components, or physical realizations which are more amenable to majority logic than to the conventional Boolean functions of "AND" and "OR". The parametron and tunnel diode circuits are examples of devices of this type. Majority logic functions have frequently been discussed for equipment to be mechanized with tunnel diodes or parametrons (or phase lock oscillators). However, majority logic can also be mechanized with conventional components, such as resistors, diodes, and transistors, and used to mechanize a computer that might otherwise be mechanized with conventional Boolean logical elements.

One example of threshold elements and a computer mechanized with them has been described by Coates and Lewis.[36] However, these threshold elements were not restricted to the special "majority" case. This machine called "DONUT" was designed and built specifically to compare the use of threshold logic elements with conventional gates. In many cases, specific logical functions can be realized with a smaller number of threshold gates than would be required if conventional gates were to be used. However, since threshold gates are not inherently bistable or on-off devices, they are more sensitive to signal variations and component tolerances than conventional gates. Hence, theoretically possible component savings may not be actually realizable for a specified level of reliability.

The DONUT computer was built to answer questions concerning component savings and relative reliability in systems mechanized with threshold logic. For this specific machine, it was found that the number of threshold gates required was about one-quarter the number of "NOR" gates that would have been required with conventional Boolean functions. However, since a general threshold gate with different threshold levels was used, the reduction in the number of gates was significantly greater than if the design had been limited to gates using pure majority logic. Also, it does not follow that a reduction in the number of gates necessarily produces a reduction in the number of components. For example, if a majority gate were fabricated as a pure threshold device (with a consequent reduction in tolerance capability), the number of components would probably be reduced; but, if each majority logic were mechanized by conventional Boolean operators, such as "AND" and "OR", the total number of components might easily be greater rather than less. From the design of the DONUT computer, it was concluded that "threshold gate logic offers a means for trading tolerances (and to some extent speed) for numbers of components."[37] However, this may not be a desirable trade-off, and it is not clear that a saving would have been achieved if only majority elements were used. There is no indication that components used in a 1970 ACDS will be more amenable to majority logic. The semiconductor integrated circuits that are expected to dominate the logic mechanization of computers in that time frame are well suited to the mechanization of conventional Boolean functions. Hence, majority logic, in the sense discussed above, is not recommended for this system.

9.2 TRIPLE REDUNDANCY WITH A "VOTING" CIRCUIT

The second way in which the term "majority logic" is sometimes used is in connection with a redundancy system in which each logical function is triplicated and the "best-two-out-of-three" is taken as the correct result for the function.[38,39,40,41] This is also frequently referred to as "voting logic." It is interesting to note that in this type of mechanization, the logical function itself is usually mechanized with conventional Boolean elements such as "AND" and "OR" gates. The only majority logic element is the actual voting element that determines the "best-two-out-of-three" results of the triplicated logical function and passes this on as the result. Even this may be mechanized as a Boolean function rather than with a specific majority logic element. Hence, it is misleading to speak of this approach as "majority logic." To do so tends to confuse a redundance technique with a type of logical element.

There are many approaches to achieving higher reliability through the use of redundancy in a system. One of these is the triplication of portions of the system with a voting circuit or majority logic element used to select the proper output if one of the three functions differs from the other two. This can be done at any level; i.e., logical functions, equipment, subsystems, etc. Unfortunately, this approach in its simple form assumes a very high degree of reliability in the majority logic element itself. In fact, it is valid only if the majority logic element is more reliable than the logical elements used in mechanizing the triplicated functions. Since this may not be the case for a simple majority logic element, it may be necessary to triplicate the majority logic elements also. Triplicating these results in three outputs which must be accepted by the next level of logic. This requires an iterative type triplication which results in significant increase in the number of components.[41] The use of majority logic and redundancy of this type must be evaluated first against the need for redundancy in the system and then against other types of redundancy.

10.0 SOURCES OF INFORMATION AND REFERENCE BIBLIOGRAPHY

10.1 ORGANIZATIONS CONTACTED

Components and packaging techniques have been discussed with personnel of a number of different companies and governmental agencies in the course of this study. The following list indicates the companies and governmental agencies with whom components and packaging techniques have been discussed, and the topics discussed with each:

Motorola Semiconductor Div. Phoenix, Arizona	Integrated circuit sense amplifiers Integrated circuit storage registers Monolithic integrated circuits Hybrid integrated circuits
Remington Rand UNIVAC St. Paul, Minnesota	Hybrid integrated circuits Integrated circuit reliability and failure analysis Packaging techniques
Control Data Corp.	Integrated circuit applications Packaging techniques
Autonetics Anaheim, California	Monolithic integrated circuits Integrated field-effect-transistor circuits Integrated circuit packaging techniques
Bunker-Ramo Corp. Canoga Park, California	Monolithic integrated circuits Integrated circuit sense amplifiers
RCA Laboratories Princeton, New Jersey	Integrated field-effect- transistors Metal-oxide-semiconductor integrated circuits Active thin-film integrated circuits
Hughes Semiconductor Div. Newport Beach, California	Integrated circuit packaging techniques Monolithic integrated circuits Hybrid integrated circuits Active thin-film integrated circuit
Sylvania Waltham, Massachusetts	Tunnel diode circuits
ONR Washington, D. C.	Optical components

RADC Rome, New York	Optical components
SRI Menlo Park, California	All-magnetic logic Cellular logic for integrated circuits Electron beam fabrication Fluid logic
Fairchild Semiconductor Mountainview, California	Monolithic integrated circuits Hybrid integrated circuits Interconnection and packaging techniques
Optics Technology Belmont, California	Fiber optics Laser techniques
NASA Washington, D.C.	Active thin-film integrated circuits Monolithic integrated circuits Packaging techniques
National Security Agency Fort Mead, Virginia	Optical techniques Integrated circuits
Hughes Ground Systems Fullerton, California	Environmental stabilization of electronic equipment
Raytheon SC Division Mountainview, California	Monolithic integrated circuits Packaging techniques MOS circuits
Signetics Corp. Mountainview, California	Monolithic integrated circuits Field effect devices Packaging techniques
Honeywell Electronics DP Div. Wellesley Hills, Mass.	Logic circuits
Burroughs Paoli, California	Thin-film circuits
Navy BuShips	Packaging and maintainability techniques Integrated circuits
USAER&DL Fort Monmouth, New Jersey	Lasers High-speed serial circuits
Texas Instruments Dallas, Texas	Multi-circuit chip fabrication Monolithic integrated circuits
Remington Rand UNIVAC Blue Bell, Pennsylvania	Fluid logic (pneumatic) Interconnection techniques

Discussions with personnel of these organizations provided a basis for much of the information presented in this report. In addition to discussing techniques and approaches that have not been adequately described in published literature, the opinions of experts in specific areas in these organizations were solicited concerning the advantages, disadvantages, limitations and future prospects for different component and packaging techniques.

10.2 LITERATURE

An extensive list of references pertinent to the study of components and packaging techniques is given in the Bibliography. A study of many of these references has contributed to the material presented in this report. Some of the more pertinent and important of these referred to in the text have been extracted from the bibliography and listed in this section as specific numbered references. These numbers correspond to numbered citations in the test. Direct quotations have been used where noted.

11.0 COMPONENT AND PACKAGING TECHNOLOGY REFERENCES

1. "The Case for Cryogenics?", Ittner, W.V., Proceedings 1962 FJCC, pp. 229-231, Philadelphia, Pa., December 1962.

2. "Pneumatic Log" I-IV, Holbrook, E.L., Control Engineering, July, August, November 1961, and February 1962.

3. "Fiber Optics and the Laser", Kapany, N.S., paper presented at the New York Academy of Sciences Conference on the Laser, New York, New York, May 4-5, 1964.

4. "The Status of Optical Logic Elements for Nanosecond Computer Systems", Tippett, J.T., 1963 Pacific Computer Conference, IEEE, Pasadena, California, pp. 47-53, March 15-16, 1963.

5. "Possible Uses of Lasers in Optical Logic Functions", Koster, C., 1963 Pacific Computer Conference, IEEE, Pasadena, California, pp. 54-62, March 15-16, 1963.

6. "A Survey of Tunnel-Diode Digital Techniques", Sims, R.C., Beck, E.R., Jr., and Kamm, V.C., Proceedings of the IRE, Vol. 49, No. 1, pp. 136-146, January 1961.

7. "300 mcs Tunnel Diode Logic Circuits", Cooperman, M., 1963 Pacific Computer Conference IEEE, Pasadena, California, pp. 166-186, March 15-16, 1963.

8. "Design of an All Magnetic Computing System", Crane, H.D. and Van DeRiet, E.K., IRE Transactions on Electronic Computers, Vol. EC-10, No. 2, pp. 207-232, June 1961.

9. "The Case for Magnetic Logic", Rogers, J., and Kings, J., Electronics, Vol. 37, No. 17, pp. 40-47, June 1, 1964.

10. "All Magnetic Digital Circuit Fundamentals", Newhall, E.E., Digest of 1964 International Solid State Circuits Conference, pp. 16-17, Philadelphia, Pa., February 1964.

11. "All Magnetic Digital Circuits and Application Problems", Baker, T., and Dillon, C., Digest of 1964 International Solid State Circuits Conference, pp. 18-19, Philadelphia, Pa., February 1964.

12. "Study of Applications of All-Magnetic Logic to Naval Systems", Adams, M.B., and Van DeRiet, I.E., Final Report on ONR Contract No. N-onr-2332(oo), Stanford Research Institute, November 1963.

13. "Microelectronics — Military Participation and Objectives", Alberts, R., 1964 WESCON, Los Angeles, California, August 25, 1964.

14. "Big Computer Goes in Small Package", Electronics, pp. 28-29, March 14, 1964.

15. "Solid Logic Technology: Versatile, High-Performance Microelectronics", Davis, E.M., Harding, W.E., Swartz, R.S., Korning, J.J., IBM Journal of Research & Development, Vol. 8, No. 2, pp. 102-114.

16. "Digital Computer Aspects of Integrated Circuit Applications", Platzek, R.C., and Goodman, H.C., Proceedings Nat'l. Winter Convention on Military Electronics, Los Angeles, California, Vol. III, pp. 2-34 – 2-53, February 5-7, 1964.

17. "Microelectronics – Where, Why, and When", O'Connell, E.P. and Brauer, J.S., Proceedings Nat'l. Winter Convention on Military Electronics, Los Angeles, California, Vol. III, pp. 2-1, Feb. 5-7, 1964.

18. "1964: The Year Micro Circuits Grew Up", Electronics, pp. 10-11, March 14, 1964.

19. "The Economic Impact of Integrated Circuitry", Haggarty, P.E., IEEE Spectrum, Vol. 1, No. 6, pp. 80-82, June 1964.

20. "Film Technology", McLean, David, Verbal Presentation at the IEEE Integrated Circuit Workshop, Monterey, California, September 17-18, 1964.

21. "Monolithic Integrated Circuits, Philips, A.B., IEEE Spectrum, Vol. 1, No. 6, pp. A-3 – 101, June 1964.

22. "The Minimization of Parasitics in Integrated Circuits by Dielectric Isolation", Maxwell, D.A., Beeson, R.H., and Allison, D.F., 1964 WESCON, Los Angeles, California, August 25, 1964.

23. "Integrated Linear Circuits", Bailey, D., Electronic Products, pp. 50, June 1964.

24. "Integrated High-Frequency D.C. Amplifiers", Breuer, D.R., TRN Space Technology Laboratories Report No. 9374.1 - 0001, Redondo Beach, California, June 10, 1964.

25. "Types of Integrated Circuits", Hogan, C.L., IEEE Spectrum, Vol. 1, No. 6, pp. 63-71, June 1964.

26. "Utilization of New Techniques and Devices in Integrated Circuits", Second Quarterly Report, AF Contract No. AF 33(657)-11185 (Pacific Semiconductor Inc.) 1 August 1963 - 31 August 1963.

27. "The Future of Thin-Film Active Devices", Feldman, Charles, Electronics, Vol. 37, No. 4, pp. 23-26, January 24, 1964.

28. "Thin-Film Circuit Technology: Part III-Active Thin-Film Devices", Fowler, A.B., IEEE Spectrum, Vol. 1, No. 6, pp. 102-111, June 1964.

29. "Integrated Thin-Film Networks", Western Electronic News, pp. 22-24, November 1964.

30. "Metal-Oxide-Semiconductor Field-Effect Transistors", Heiman, F.P., and Hofstein, S.R., Electronics, Vol. 37, No. 30, pp. 50-61, November 30, 1964.

31. "MOS and Thin-Film Transistor Circuits", Hertzog, G., Verbal Presentation at the IEEE Integrated Circuit Workshop, Monterey, California, September 17-18, 1964.

32. "Integrated-MOST Memory", Schmidt, J.D., Verbal Presentation at the IEEE Computer Memory Workshop, Lake Arrowhead, California, September 10-11, 1964.

33. "Silicon FEB Techniques", Kilby, J.S., Solid State Design, Vol. 5, No. 7, pp. 32-37, July 1964.

34. "System Speed with Integrated Circuits", Powers, G., Electronic Design News, pp. 20-27, May 1964.

35. "Space Radiation Affects MOS FET's", Highes, H.L., and Giroux, R.R., Electronics, Vol. 37, No. 32, pp. 58-60, December 28, 1964.

36. "DONUT: A Threshold Gate Computer", Coates, C.L., and Lewis, P.M., IEEE Transactions on Electronic Computers, Vol. EC-13, No. 3, pp. 240-247, June 1964.

37. "How to Achieve MAJORITY and THRESHOLD LOGIC with Semiconductors", Sauer, W.A., Electronics, Vol. 36, No. 48, pp. 23-25, November 29, 1963.

38. "Improvement of Electronic-Computer Reliability Through Use of Redundancy", Brown, W.G., Tierney, J., and Wasserman, R., IRE Transactions on Electronic Computers, Vol. EC-10, pp. 407-415, September 1961.

39. "Majority Voting Protects Aircraft and Pilot", Moreines, J., Worthington, R., and Thomas, F., Electronics, Vol. 37, No. 16, pp. 85-91, May 18, 1964.

40. "Redundant Circuit Design: Payoffs and Penalties", Suran, J.J., EEE, Vol. 12, No. 9, pp. 51-55, September 1964.

41. "Information Redundancy and Adaptive Structures", Angell, J.B., Digest of Technical Papers, 1964 Solid State Circuits Conference, Philadelphia, Pa., pp.84-85, February 19, 20, 21, 1964.

Appendix C
DISPLAY CONSOLE SPECIFICATIONS

	American Bosch Arma FC-300	Bunker-Ramo BR-85	Bunker-Ramo DC 400C	Bunker-Ramo 211/212
Screen Size (Inches)	10 3/4 dia, 14 3/4 dia, 18 3/4 dia	12 x 16	6 x 6	3 1/2 x 4 1/2
Character Sizes	.37 x .28 max. 2 sizes	.19 x .12, .31 x .22 internal adj.	.19 x .12	.31 x .25 -25% int. adj.
Character Gen. Rate	500K	100K	50K	50K
Character Gen. Technique	Stroke	Dot	Stroke	5 x 7 Dot
Symbol & Char. Repertoire	64	64	64	
Accuracy of Positioning	.2%	.4%		
Max Vector Length (inches)	18 3/4	3		
Brightness (ft.-lamberts)	20	40	30	25
Max. Ambient Ill. (lum/ft^2)	25	10	5	30
Phosopher	P31 P28	P4	P7	P31
Spot Size, Inches	.010	.020	.020	.015
Stability	.1%	2%	2%	
Resolution Lines/Inches	100	32		30
Frame Rate	35	20	60	40
Storage Capacity (Words)	4096	4096	720	768
Word Size (Bits)	12	9	6	8
Type Storage	Core or delay line	Core	Core	Delay line
Max. Data Input Rate Words/Second	5K	100K	20K	240 char./sec.
Features	Rear projection line drawing, light pen, cursor, variable function keys, flat face CRT	Line drawing, light pen, status lights, cursor, fixed and var. function keys off-line message composition.	Status lights, fixed and variable function keys.	Light pen, editing marker, fixed function keys
Space Requirement (cu.ft.)	60	85	54	1.6
Weight (lbs)	800	1300	1000	45
Power Requirements (watts)	2.2K	3.6K	1K	70
Availability (months)	6	7-8	6-7	In. Prod.
Price		150K-170K	80K-100K	1050-1200
Auxilliary Equipments	Core Memory Char. Gen.	Digital Unit	Display Buffer	Control Unit, Char. Gen. Comp. Interface
Remarks	Brightness without proj. 2 lumens/ft^2 CRT image with white back-ground 40 lumens/ft^2	Price under re-evaluation	Buffer 56 cu. ft. 1000 lbs, 1800 watts, $60K-90K	Price does not include control unit

	Bunker-Ramo 203	Data Display DD10	Data Display DD40	Data Display DD80
Screen Size (Inches)	6 5/8 x 9 1/8	6 x 8	10 x 10, 12 x 12	10 x 10,.69 x.69* film plane
Character Sizes	.25 x .31- -25% int. adj.	.25 x .20	.12 x .06, .25x.12 .50 x .24	.12x.06,.18x.09, .25x.12, .37x.16
Character Gen. Rate	50K		125K	110K
Character Gen. Technique	5/7 Dot			Stroke
Symbol & Char. Repertoire		64	64	123
Accuracy of Positioning			.1%	.1%
Max Vector Length (Inches)			15.4	18.4
Brightness (ft.-lamberts)	25			40
Max. Ambient Ill. (lum/ft^2)	30			
Phosopher	P31	P4	P4 P31	P31 P11*
Spot Size, Inches	.015			.030 .001*
Stability				2%
Resolution Lines/Inches	30	4 char/in.		700*
Frame Rate	40	50	40	50 30*
Storage Capacity (words)	768	500/console	2048	2048
Word Size (Bits)	8	6	36	32
Type Storage	Delay line	Core	Core	Core
Max. Data Input Rate Word/Second	240 char./sec.	50K		90K
Features	Light pen, editing marker,fixed function keys	Off-line message composition Entry marker	Line drawing, blinking,light pen, cursor, variable function keys	Line drawing. Characters may be rotated 90°. Two intensities. Italic characters.
Space Requirement (cu.ft.)	2.4	2.2	80	64
Weight (lbs)	60		450	1615
Power Requirements (watts)	85			2K
Availability (months)	In Prod.	4-6	4-6	4-6
Price	1900	5200	50K	125K
Auxilliary Equipments	Char. gen. Control unit Comp. Interface	Control unit, Interface	Logical Unit	400 cycle generator-motor
Remarks	Char. gen. comm interface. Price does not include control unit.	Up to 64 displays/ control unit. Control unit-$25K	Buffer is optional. Logical unit-85cu ft, 1000 lbs. 3 additional monitors	*Parallel 5" display for film recording. Recorder requires 66 cu. ft. Char. size programmable.

	Digital Equipment Corp 340	Electrada 408-2	ITT Integrated Console	Information Products Corp 1570
Screen Size (Inches)	9 3/8 x 9 3/8	11 1/2 x 8	12 x 11 1/2*, 13x13	2 1/2 x 4 1/2
Character Sizes	.5 x .35 max 4 sizes	.2 x .14	.21 x .16	.16 x .1
Character Gen. Rate	30K			10K
Character Gen. Technique	Dot		Monoscope	5 x 7 Dot
Symbol & Char. Repertoire	128, 64	64	64	64
Accuracy of Positioning	.1%	1%	.1%	
Max Vector Length (inches)	9 3/8			
Brightness (ft.-lamberts)	Programmable	30	30	
Max. Ambient Ill. (lum/ft^2)			12	
Phosopher	P7	P4 P2		P20 P31
Spot Size, Inches	.030	.015	.004	
Stability	.5%	1%		
Resolution Lines/Inches	50	50	125	
Frame Rate		60	60	100
Storage Capacity (words)		11K char.		100
Word Size (Bits)	18	6	56	8
Type Storage		Drum		Delay line
Max. Data Input Rate Word/Second	333K	30K	10K	10K
Features	Line Drawing	Displays upper and lower page. Two underscore markers.	Additive color system, status lights, variable function keys.	
Space Requirement (cu.ft.)	80	72	75	1.5
Weight (lbs)	700			35
Power Requirements (watts)	1.5K	200		200
Availability (months)	2	6	9	2-3
Price	28.6K	48K	200K	4300
Auxiliary Equipments	Char. gen. Light pen, Interface			Control Unit
Remarks	Auxiliary buffer required		Color is by projection of 3-35 mm films *Alternate display for keyboard data	

	Information Products Corp. 1550	IBM 1015	IBM 2250	LFE SM2
Screen Size (Inches)	4 1/2 x 7 1/2	5 dia	12 x 12	21 dia
Character Sizes	.16 x .1	.1 x .07	.15 x .11, .10 x .07	.12 x .10
Character Gen. Rate	25K		67K*	13K
Character Gen. Technique	5 x 7 Dot	5x7 Dot	Stroke	5x7 Dot, 7x9 Dot
Symbol & Char. Repertoire	64	64	63	63
Accuracy of Positioning			.1%	
Max Vector Length (inches)			16.8	
Brightness (ft.-lamberts)				
Max. Ambient Ill.(lum/ft^2)				
Phosopher	P20 P31	P10	P7 P19	
Spot Size, Inches		.008	.025	
Stability				
Resolution Lines/Inches				
Frame Rate	50.		30	
Storage Capacity (words)	500	1200	4K, 8K, 16K	
Word Size (Bits)	8	8 bit	8	
Type Storage	Delay line	Storage Tube	Core	Disk
Max. Data Input Rate Word/Second	25K	650 char./sec.	240K	13K char./sec.
Features	Function keys	Line select switch, fixed function keys, status lights	Light pen, variable function keys, line drawing, cursor	Half-tone storage
Space Requirement (cu.ft.)	2	75	125	
Weight (lbs)	45	375	590	
Power Requirements (watts)	300	280	2.8K	
Availability (months)	2-3	24	24	
Price	5800	13K	34K - 75K	
Auxilliary Equipments	Control Unit	Control Unit $14K	Buffer,Char. Gen.	
Remarks	Keyboard has numerics, space & 6 letters or symbols.	10 Displays/Control Unit. Light background. Storage tube.	8-Model 2/Control Unit. CU has char. gen. & 8192 byte buffer. *10K using computer instead of c.g.	Storage Tube 20 min. max. display.

	LFE SM2A	LFE SM3	Marquardt DC 201	Philco READ
Screen Size (Inches)	14 x 19	2 1/2 x 4 1/2	7 3/4 x 7 3/4	16 & 19 dia, 14x19, 10&12 dia projection
Character Sizes	variable		1x1 max 3 sizes	7 sizes 8:1 range
Character Gen. Rate	500K	1000K	52K	222K
Character Gen. Technique	Stroke	5x5, 5x7, 7x9, 6x8 dot	Stroke	Stroke
Symbol & Char. Repertoire	64, 128 max		64	64 to 512
Accuracy of Positioning			.2%	.2%
Max Vector Length (inches)			7 3/4	1/8 screen size
Brightness (ft.-lamberts)			100	
Max. Ambient Ill. (lum/ft^2)	10			
Phosopher	P31	P4	P7	
Spot Size, Inches			.010	
Stability			1%	
Resolution Lines/Inches				
Frame Rate	50			30
Storage Capacity (words)	50K char.	50K char.	4096	4096
Word Size (Bits)	9	Dot Matrix Size	12	36
Type Storage	Disk	Disk	Core	Core
Max. Data Input Rate Word/Second			78K	950K char./sec
Features	Programmed intensity -4 levels, cursor	Status Lights	X-Y line drawing, Light Pen	Line drawing, light pen, cursor, off-line operation
Space Requirement (cu.ft.)		36	60	24
Weight (lbs)			800	325
Power Requirements (watts)				500
Availability (months)			6	
Price				
Auxilliary Equipments			CDC 160 computer	Control Unit
Remarks			Character size programmable	15 remote units/control unit. CU 60 cu. ft. 700 lbs 2000 watts

	Scientific Data Systems 9185-11	Stromberg-Carlson SC- 1060	Stromberg-Carlson SC-1090	Stromberg-Carlson SC-1090 AWRIS
Screen Size (Inches)	10 1/2 x 10 1/2	10 1/2 x 10 1/2	12 x 12	12 x 12
Character Sizes	.16 x .12, .32 x .24, .64 x .32	.1 x .07	.1 x .07 adjustable .2 x .14 to .07x.05	.1 x .07 adjustable .2 x .14 to .07x.05
Character Gen. Rate	25K	40K	30K	40K 80K max.
Character Gen. Technique	Stroke	Shaped Beam	Shaped Beam	Shaped Beam
Symbol & Char. Repertoire	64	64	64 96 max.	64
Accuracy of Positioning	.1%		.5%	.5%
Max Vector Length(inches)	14.2	15	2	17
Brightness (ft.-lamberts)		20	40	40
Max. Ambient Ill.(lum/ft^2)			10	10
Phospher	P7	P12	P14	P28
Spot Size, Inches		.010	.025	.025
Stability	.025%	.5%	1%	.5%
Resolution Lines/Inches	100	266	235	235
Frame Rate	char rate 30	25	30	30
Storage Capacity (words)			1024	1024
Word Size (Bits)	24	30	36	36 internal, 18 I/O
Type Storage	Core	Core	Core	Core
Max. Data Input Rate Word/Second	50K min.	40K	30K	60K, 18 bit
Features	Line drawing, Light Pen	4 Analog channels & time shared digital data	Line drawing, light pen, rear projection, expansion, off-centering, category selection	Cursor, line drawing hard copy printer. Data entry keys
Space Requirement (cu.ft.)	5.5	40	70	70
Weight (lbs)	70	350	1000	1000
Power Requirements (watts)	250	2K	2.5K	2.5K
Availability (months)	4	5	2	5
Price	7000	65K	31K	63K
Auxilliary Equipments	Control unit, Char gen., Vector gen.	Display Buffer	Display Buffer	Display Buffer
Remarks	Control unit $3500, Char. gen. $3000, Vector gen. $1000, Light pen $1400, Power supply $3000 SDS com. for mem.buf		Price for minimum configuration	

	Raytheon DIDS-1500	Raytheon DIDS-500	RCA IDC 6320	Tasker Instruments TC-100
Screen Size (Inches)	12 x 12	16 x 16	16 x 12	19 x 14
Character Sizes	.12x.08 , .19x.11, .28x.18 selectable	.25 x .15	variable	.125 x .1 computer controlled
Character Gen. Rate	300K	250K	4K	400K
Character Gen. Technique	Stroke	Stroke	mono scope	Stroke
Symbol & Char. Repertoire	91	122	128	32, 64, 128
Accuracy of Positioning	.2%	.2%	.1%	.2%
Max Vector Length (inches)	.75		8	23 1/2
Brightness, (ft.-lamberts)	30	50	20	
Max. Ambient Ill. (lum/ft^2)	30	30		
Phosopher	P31	P31	P4	P4
Spot Size Inches	.020	.020	.010	.020
Stability	.05%	.05%	.5%	.2%
Resolution Lines/Inches	50	50	100	50
Frame Rate	48	60	60	60
Storage Capacity (words)	4096	4096	36K	
Word Size (Bits)	18	9	10	
Type Storage	Core	Core	Core or Drum	Drum
Max. Data Input Rate Word/Second	35K	50K 18 bit	300K	
Features	Cursor, 20variable function keys, blinking.	Cursor, edit controls, hard copy printer, status lights	Line drawing with light gun status lights, security card	Line drawing
Space Requirement(cu.ft.)	21	24	30	8
Weight (lbs)	450	500	1.5K	220
Power Requirements (watts)	675	1.1	3K	3K
Availability (months)			6	
Price			125	
Auxiliary Equipments	Logic, character & vector gen.		Control Unit.	
Remarks	Logic Unit 54 cu. ft., 1050 lbs. 1025 watts	Mil - Spec.	Closed circuit TV on same screen.	

	Tasker Instruments 544	Tasker Instruments 916	Sanders Associates 820	Information Displays M11000
Screen Size (Inches)	16 x 2 4 screens	21 dia	13 1/2 x 13 1/2	12x16, 13x13, 15 dia
Character Sizes	.2 x .2		.1 x .075	.1x.075 to .5x.38
Character Gen. Rate	3.6K		33000	100K
Character Gen. Technique	Dot		shaped beam	stroke
Symbol & Char. Repertoire	64		64	64
Accuracy of Positioning		.1%	.2%	1%
Max Vector Length (inches)		21	2	4
Brightness (ft.-lamberts)		100	20	
Max. Ambient Ill. (lum/ft^2)	100		35	
Phosopher	P31	P20	P31	P31
Spot Size, Inches			.008	.010
Stability			.02%	.5%
Resolution Lines/Inches		30	100	100
Frame Rate	50	-	35 min.	30
Storage Capacity (words)	12		2300	
Word Size (Bits)	36		.12	20
Type Storage	Delay line	Dialectric Mesh	Core	
Max. Data Input Rate Word/Second	Keyboard entry		83K	150K
Features	Character marker, status lights, many keys, 8 large characters are comp.selectable	Adjustable persistence	Two vector widths light pen, status lights, rear projection TTY I/O mag. tape I/O	Line drawing, light pen, 4 sizes, 4 intensities keyboard, ci circle generator, texture control, jitter, all are options
Space Requirement (cu.ft.)	100	12	64	30
Weight (lbs)	500	275	500	580
Power Requirements (watts)	500	250	1K	1500
Availability (months)		3	3	
Price	50K	25K		22K-35K
Auxilliary Equipments		Char. Gen.	Buffer for computer interface	Char. Gen. Buffer
Remarks	4-4x2 CRT's form 72 char. line.	3 minute electrostatic storage. 2 input channels.	May be operated off-line	Price does not include auxilliary eqpt. or options

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* INFORMATICS INC. 15300 Ventura Blvd., Sherman Oaks, Calif.	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP -

3. REPORT TITLE
ADVANCED NAVAL TACTICAL COMMAND AND CONTROL STUDY, VOLUME V, TECHNOLOGY

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Summary of Work Concluded in Period January 1964 to January 1965

5. AUTHOR(S) *(Last name, first name, initial)*
Cohen, I.; Amdahl, L.; Bigelow, R., Craver, J.S.; Frank, W. L.; Granholm, J. W.; Hobbs, L.C., Mersel, J.; Morris, W. A.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
15 JANUARY 1965	763	295

8a. CONTRACT OR GRANT NO. Nonr-4388 (00)	9a. ORIGINATOR'S REPORT NUMBER(S) TR-65-58-2
b. PROJECT NO. - c. - d. -	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* None

10. AVAILABILITY/LIMITATION NOTICES
U.S. military agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through the Advanced Warfare Systems Division of the Office of Naval Research.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	ONR

13. ABSTRACT

This volume presents results of a study to identify, analyze, and evaluate information systems technology applicable to future tactical Navy Command and Control Systems. Included is a discussion of electronic data processing systems, memories, input/output devices, displays, computer programming, components and packaging, artificial intelligence, and self-diagnostic systems. These technical areas are analyzed on the basis of current technology and anticipated developments versus requirements of a 1970 era Naval tactical system.

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
INPUT/OUTPUT TECHNOLOGY						
DISPLAY SYSTEMS						
DISPLAY TECHNOLOGY						
COMPONENTS AND PACKAGING						
MEMORY TECHNOLOGY						
COMPUTER SYSTEMS ORGANIZATION						
PROGRAMMING						
ARTIFICIAL INTELLIGENCE						
SELF-DIAGNOSTIC SYSTEMS						

UNCLASSIFIED
Security Classification