THE BELL SYSTEM TECHNICAL JOURNAL
SAFEGUARD SUPPLEMENT
Printed in U.S.A.

SAFEGUARD Data-Processing System:

Architecture of the Central Logic and Control

By J. W. OLSON

(Manuscript received January 3, 1975)

The Central Logic and Control (CLC) unit is the digital computer that controls SAFEGUARD. Its development represents the first reduction to practice of large-scale multiprocessing in a computer system. This paper describes the CLC and explains some of the decisions behind its design.

I. INTRODUCTION

The Central Logic and Control (CLC) represents the first practical application of the multiprocessing concept to a large-scale computing system. A modular design is employed in which as many as ten processors and two Input/Output Controllers (IOCs) share as many as 32 memory racks. The units are interconnected by a flexible switching network that allows the system to be partitioned into two independent computers. Partitioning can be controlled by software, and complete reconfiguration may be accomplished in less than one second.

This paper focuses on the architecture of the CLC, and on how system requirements influenced the decisions behind its design.

II. DESIGN PHILOSOPHY

2.1 System requirements

Availability, reliability, and performance requirements are placed on the CLC because of its importance to SAFEGUARD. The data-processing system is required to be fault-tolerant. This means that the system must be able to perform its workload in the presence of any single malfunction. In addition, the CLC is allowed only a limited amount of down time. High-reliability specifications are placed on each of the components from which the CLC is fabricated to increase the mean-time-to-failure. High CLC performance requirements are dictated by the nature of its primary job, controlling a radar tracking system in real time and the launch/guidance of missile interceptors. Sufficient

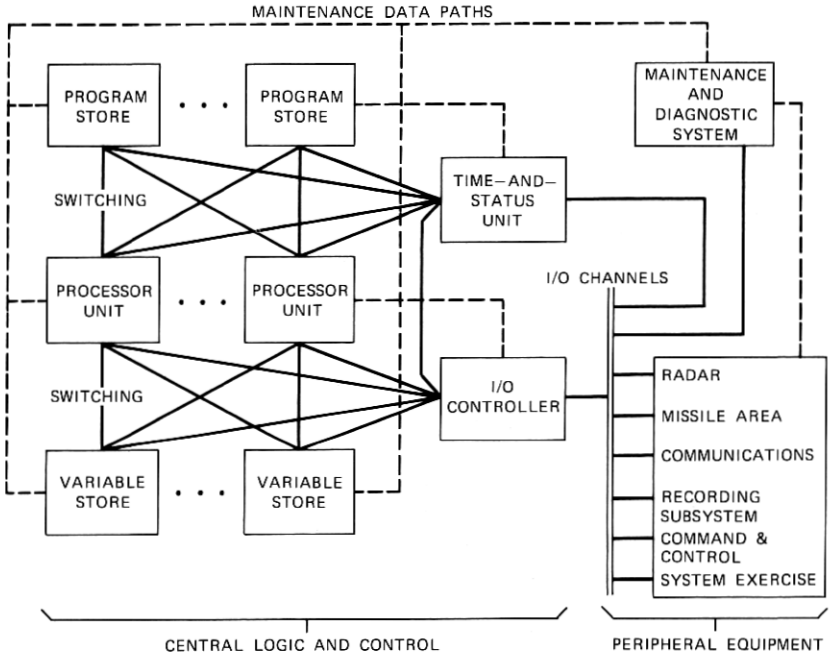

Fig. 1—Central Logic and Control unit.

reserve power must be available to handle peak loads. A block diagram of the CLC is shown in Fig. 1.

2.2 Resulting architecture decisions

2.2.1 Modularity

The CLC is composed of five types of elements: up to ten processors, sixteen racks of program store, sixteen racks of variable store, two IOCs, and two time-and-status units. This system is capable of operation with only one element of each type and may grow in a modular fashion. The IOC provides peripheral-world access to the computer while the time-and-status unit provides a number of special functions which include real-time clocking, monitoring system status, and controlling the configuration of the hardware resources in the system. The multiple elements communicate via well-defined interfaces and are interconnected by a flexible switching network.

The method of interconnecting elements within the multiunit computing system must permit ease of growth and be consistent with the availability and reliability requirements. The switching method chosen for the CLC is based upon a distributed implementation of the switching network such that a portion of the switch is included with each unique

system element. Both economic and availability considerations favor a distributed switch in which each added processor and storage element comes with its own portion of the switching system to allow smooth system growth. System availability is enhanced because a failure of a portion of the distributed switching system affects only the unique element to which it is attached.

All communication among elements of these five types is handled asynchronously on a request-and-acknowledge signaling basis. The collection of processors is capable of asynchronously accessing any of the collection of memory elements. The switching network is such that if each processor makes an access to a different memory element, then all may receive service simultaneously. Priority circuits at each memory element resolve conflicting requests sequentially.

2.2.2 Multiple processors

Although it would have been possible to design a single processor system with sufficient performance, the CLC is a multiprocessor machine for three reasons. First, a single processor sufficiently powerful would have been a complex machine, difficult to design and difficult to get working. Second, a single-processor system would not have been expandable; if a more powerful machine were later found necessary and none were available, major software changes would have been required. Also, multiple processors satisfy a wide range of processing requirements including smaller applications. Finally, the multiprocessor design increases availability because processing can continue even if some processors have failed.

2.2.3 Two memory types

A multiprocessor design hinges around its storage design. A number of possible strategies are available to handle the necessary references of the multiple processors to main storage. The first strategy used in the design is the splitting of main storage into two independent portions called program (or instruction) store and variable (or operand) store. This organization doubles the data flow rate to each processor at the expense of independent instruction and operand fetch circuitry within each processor. One of the reasons for this architecture is to physically separate programs and data sets for reliability purposes. Thus, program store is a read-only memory, while variable store is a read-write memory which holds real-time I/O data and provides storage for the results of calculations.* To optimize memory utilization of the CLC

* Program store is read-only in the sense that processors have no instructions that write data into it. Software can alter program store via the store transfer unit which is described in a later section.

during the software development phase, additional switching paths are provided from variable store to each processor to allow instructions as well as operands to be stored in variable store.

2.2.4 "n + 1" redundancy

To achieve nearly continuous operation as economically as possible, the CLC employs $n + 1$ redundancy. Each of the five types of elements has at least a single replacement that is not required for running the application software and is therefore redundant. For example, if the application software requires 15 racks of program store for execution, then at least 16 are provided. The $n + 1$ element may be switched in to replace a failed element.

2.2.5 Partitioning

The CLC can be partitioned into two independently operating computers, each capable of executing its own job stream. By convention, these two partitions have been differentiated by the terms green and amber, with green usually the larger of the two fractions. However, since the computer is composed of a number of modular elements, the boundary defining which are green and which are amber is almost completely flexible, as illustrated in Fig. 2. In fact, all elements may be brought into the green partition to operate as a very large multiprocessor computer with as many as ten processors sharing the job load. As a further degree of flexibility, some elements, such as memory elements, may be placed into a shared green/amber state where they are available to both partitions simultaneously. Finally, an element may be defined

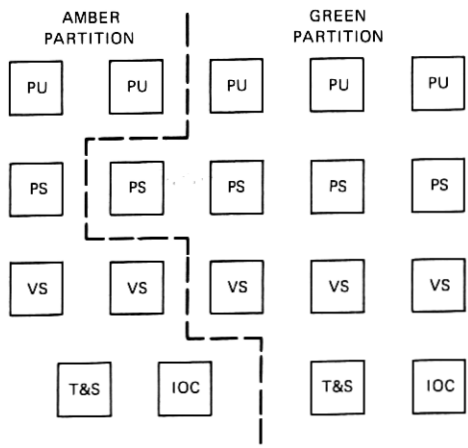

Fig. 2—Element partitioning within the CLC.

to be neither green nor amber and is said to be isolated. This state is necessary to remove malfunctioning elements without shutting down the entire system.

It is even more significant that partitioning is under program control. Further, the control logic for effecting partitioning is redundant. There is a fundamental asymmetry to the control of partitioning which allows the green partition to have priority over the amber partition. The partitioning logic may be placed into a state whereby a master/slave relationship exists between the green and amber partitions. Control software residing on the green partition may alter the partitioning of the system at any time. The amber or slave partition can in no way alter the partition boundaries. This will be described in more detail in Section 3.4.1.

III. DETAILED DESCRIPTION

3.1 The processors

The processor is the most important element in establishing the real-time computing capacity of the CLC, so the design of a high-speed processor has been a primary goal. Each processor contains three control units that operate asynchronously with respect to each other. Timing within each control unit is overlapped to some degree so that more than one instruction may be in execution. High-speed arithmetic algorithms and associated logical implementations have been exploited advantageously to increase the flow of operands through the arithmetic sections. The resulting processor design can execute successive fixed-point add operations on full-word 32-bit operands at an average rate of 4.15 million per second.

The processor organization, as shown in Fig. 3, is best explained by considering a typical arithmetic operation. Three functions must be performed: instruction fetch, operand fetch, and arithmetic execution. Three control units allow these functions to be overlapped, thus avoiding simple concatenation of the functions for successive instructions.

The Program Control Unit (PCU) prefetches instruction words from program store into an instruction word buffer. The PCU then extracts instructions from the buffer and determines which of the control units will participate in executing the instructions. For those instructions involving operand access, the operand control unit will address variable store to fetch or store all operands to be used internal to the processor. For those instructions involving arithmetic operations, the arithmetic control unit will perform all fixed-point and floating-point arithmetic.

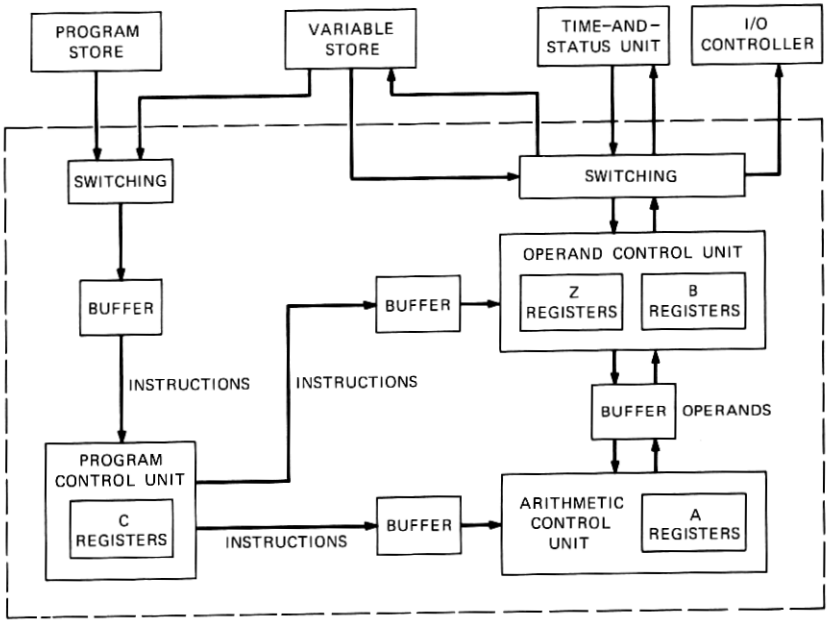

Fig. 3—Processor unit.

3.1.1 Program control unit

The PCU supplies instructions to the operand and arithmetic control units. Reference to program store is by absolute address from a location specified by a program address counter. A change from sequential operation can be effected either by interrupt or by executing a jump. Instruction sequencing is optimized by use of four double-word buffer registers that form an instruction stack. Whenever branches in the instruction sequence are encountered, alternate path fetching is employed to fetch both the normal path word and the jump path word. Both of these words are placed in the instruction stack to await a jump decision. Since many jumps are conditional to an arithmetic test within the processor, having both paths available will in general reduce the time needed to proceed regardless of which jump decision is made. In addition to the above optimizing, short instruction loops may be entirely contained within the instruction stack and executed without further access to program store. To smooth and optimize instruction flow to the other control units, instruction list buffers exist at the interface between the program control unit and the other control units.

3.1.2 Arithmetic control unit

The arithmetic control unit contains fifteen addressable A-registers for temporary storage of operands. All arithmetic operations are performed on operands from the A-registers with results returned to these registers. The registers make data currently in use immediately available to the processor. Within the arithmetic control unit, A_0 is defined to be a fixed accumulator for all arithmetic operations. The A_0-register functions alone as a single-length accumulator or in conjunction with an extension register to form a double-length accumulator. The double-length accumulator will handle the double-length results obtained for multiply operations and will hold the quotient and remainder for divide operations. The two-address arithmetic instructions will always place the result in A_0 and have the option to overwrite the second named register. This method allows some of the generality of a three-address format without the need for a third address.

3.1.3 Operand control unit

The operand control unit fetches operands from variable store; it performs any required operand fetching address arithmetic itself. Fifteen addressable B-registers provide temporary storage of addresses or index values. The operand control unit can perform shifts and edits on data contained in the B-registers. (Edits are instructions that access only a selected portion of a register.) Data can be exchanged between B-registers and A-registers.

The operand control unit provides a set of 15 addressable Z-registers which are used to control the operation of the entire processor. Interrupt jump and return addresses are found in the Z-registers. Memory protection is controlled by these registers; the appropriate bit in a Z-register must be set to allow the processor write access to a particular segment of variable store. One of the Z-registers is a delta clock which acts as an alarm clock. The delta clock will generate an interrupt if it is not reset before a selected primary countdown interval is exceeded.

3.2 The memories

To further increase the data-flow rate between processors and main storage, program and variable store are further subdivided into modular groupings, as shown in Fig. 4. Variable store is organized as 16 independent racks, with an independent data path from each rack to each of the processors. Since queuing is heavier at program store than at variable store, program store is organized as 32 independent modules with an independent data path from each module to each of

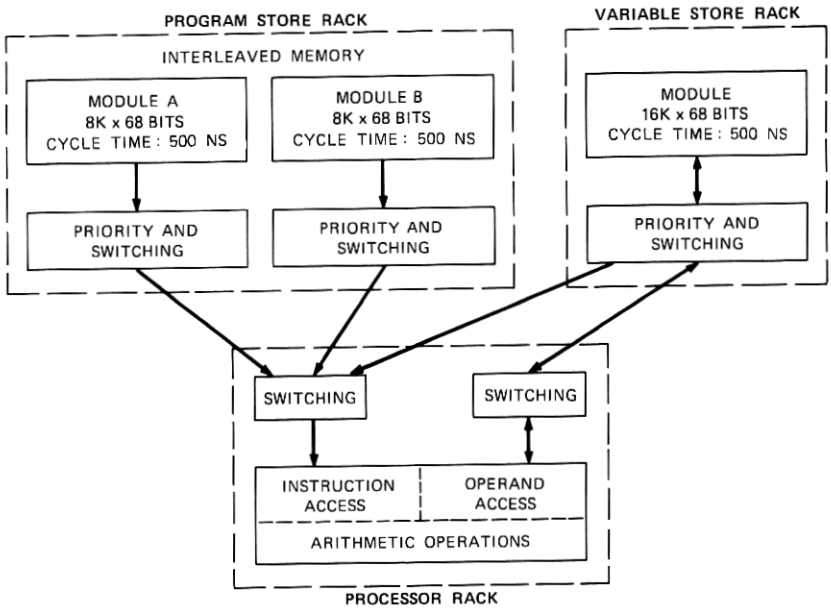

Fig. 4—Processor main-storage organization.

the processors. Processor addressing is interleaved between two modules; that is, the address structure is arranged so that adjacent program store words reside in two separate modules.

The memory module cycle time of 500 ns and the double-word size of 64 bits are selected to provide a memory bandwidth in excess of that required for maximum performance of a single processor. Each program-store and variable-store rack holds 16,384 64-bit words. There are four parity bits associated with each memory word.

In a multiprocessor system, the need frequently arises to prevent one processor from modifying data that another processor is accessing. A lock mechanism is also needed to avoid IOC and processor interference at variable store. To allow resolution of these problems, a special memory instruction called biased fetch is included. A biased fetch reads a word from variable store and, in one memory cycle, restores the word with the upper two bits set to binary ones. (Two bits are chosen because the parity of the memory word is not regenerated during the read/modify/write cycle.) The original word, *before* modification, is returned to the processor or IOC. The processor or IOC can test the upper two bits of this word to determine whether access to the data has been granted. If these bits are zeroes, the data are available; if they are ones, the data are not available.

3.3 The input/output controllers

In any computing system, input/output is of paramount importance and frequently determines throughput. The I/O Controller (IOC), which is shown in Fig. 5, directs the flow of information between variable store and the peripheral devices. Processors are thus relieved from communicating directly with the peripherals. Processors and IOCs can operate simultaneously. The I/O subsystem, which consists of the IOC and its associated peripherals, is duplicated to achieve system availability requirements.

A basic feature of the IOC is its ability to simultaneously and continuously service several peripheral devices. The fastest way to service any individual peripheral device is to transfer its entire block of data by preempting all of the transfer facilities. Since this violates the rule of simultaneous service to several peripheral devices, it is necessary to time-share the IOC facilities among all devices.

Each IOC contains 16 channels; each channel contains independent input and output cables, thereby allowing full-duplex operation. Priority circuits are utilized to allow time-multiplexed operation of the channels.

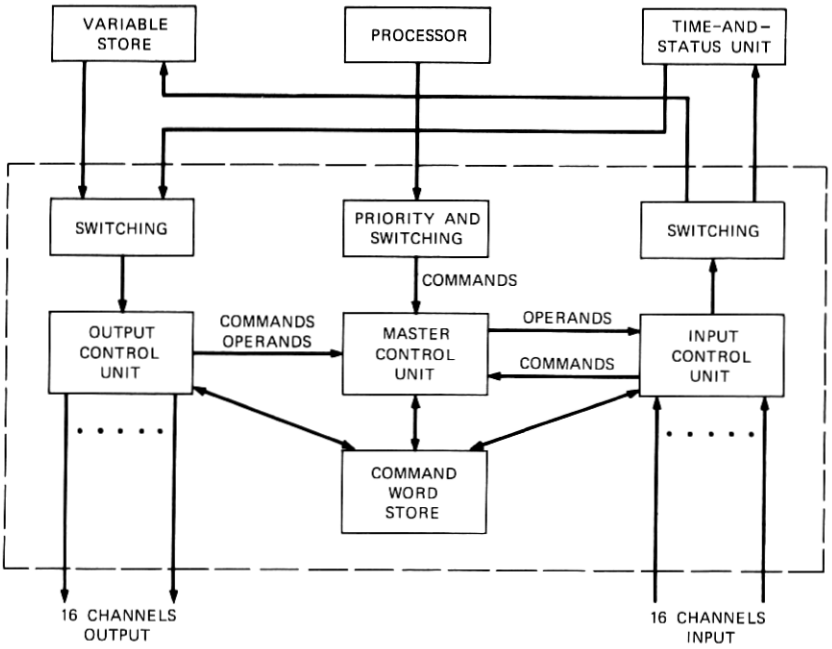

Fig. 5—Input/Output Controller.

Each peripheral is assigned a priority order which takes into consideration the allowable latency of a peripheral device requiring access to variable store. High-speed, synchronous devices usually are assigned higher priority channels than buffered, asynchronous devices.

The IOC is a programmable device. Its operations are controlled by commands it reads from variable store. The instruction repertoire includes jump commands and simple data operation commands. An IOC program can be initiated by a processor or by a peripheral device. The IOC accesses I/O programs by indirect addressing.

3.4 Associated equipment

Although the processors, the memories, and the IOCs are the principal components of the CLC, three other devices deserve mention: the status unit, the timing generator, and the store transfer unit. A block diagram of the time-and-status unit, which includes the above functions, is shown in Fig. 6.

3.4.1 Status unit

The status unit is essentially a register memory that may be read or written by all processors in a given partition. By reading from the status unit, processors obtain information about the condition of the data-processing system: parity errors, time-outs, power on-off, etc. By writing into the status unit, processors control the data-processing system.

Partitioning is controlled by signals from the status unit. Software can specify whether each component of the data-processing system is

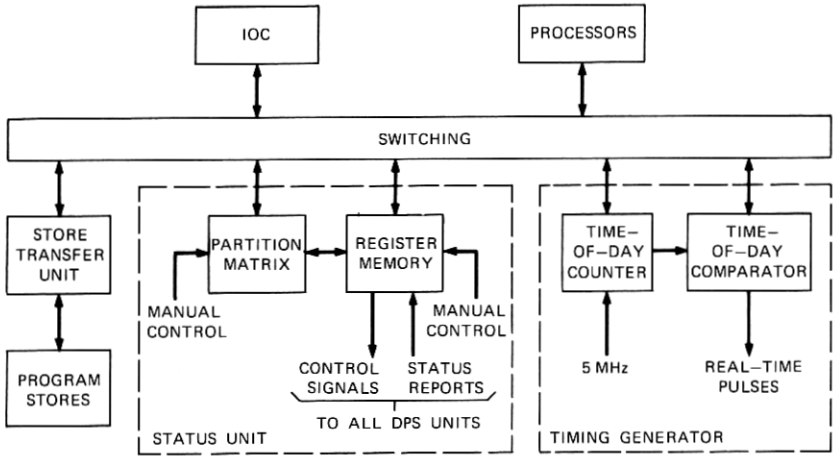

Fig. 6—Time-and-status unit.

to be partitioned green, partitioned amber, or is to be isolated. The status unit enables communication between elements in the same partition and disables communication between elements in different partitions or to elements which are isolated.

Since there are two status units, subtle logic-design problems exist. For example, status information from peripheral devices partitioned amber must affect only the amber status unit and not the green. One status unit must be designated the master and the other the slave in such a way that partitioning signals from the master status unit take precedence. Transients caused by powering up a status unit must not disturb this relationship.

The status unit also interacts with the IOC. If certain status unit bits change, the status unit presents a command request to the IOC. The IOC program thus initiated informs software of the event.

3.4.2 The timing generator

The timing generator performs two basic functions that are essential within a real-time system. The first is that of initiating activities at points in time that can be specified by program means. The second is that of providing an accurate time value which can be used in recording the time of occurrence of specific events during operation of the SAFEGUARD system. This is accomplished in the CLC by providing a time-of-day binary counter which is driven from a precise 5-MHz generator. As with other system components, for availability reasons the timing generator is duplicated. The timing generator is synchronized with a time-of-day standard. In addition, there is a procedure to synchronize the timing generator in the amber partition to the timing generator in the green partition. This is necessary whenever the amber timing generator is shut down for maintenance.

To fulfill the function of initiating activities at specified times, the timing generator performs time-notice comparisons of the time-of-day clock to a time-arranged list of orders stored within variable store. This activity is analogous to that of an alarm clock set to turn on various software processes. This function is handled via an I/O channel to relieve the processor from the housekeeping function of presenting time-notice orders to the clock. As long as the time-notice list has been prepared in advance, the IOC will methodically transfer a new order from the list maintained in variable store. In addition, the IOC will interact with the global data sets maintained in variable store to trigger various software events without necessarily providing a direct processor interrupt.

The second function of providing an accurate time value is accomplished by allowing all processors within the same partition to directly

access the clock and fetch time of day as a binary word. Access to the timing generator is designed so that, regardless of the number of processors in queue, each processor may obtain time of day in less than a microsecond. The time-of-day value can be used to attach a time tag to various recorded events or to determine whether certain system deadlines have been missed.

The timing-generator and status units may be thought of as hybrid devices within the CLC from the viewpoint that they may be accessed directly by a processor using the internal switching network within the computer or they may be accessed as a peripheral device using an I/O channel. As these devices either provide control information or report status, they are not accessed frequently during normal operations and so they share the same switching port and I/O channel. To take advantage of the economy of sharing interfaces, they are grouped together in the same equipment rack which is designated the time-and-status unit.

For partitioning purposes, the time-and-status units are paired with the IOCs to which they are attached. Thus, time-and-status unit number one is always configured in the same partition as I/O controller number one. The same philosophy holds true for many of the peripheral devices connected to the IOC in the SAFEGUARD system.

3.4.3 Store transfer unit

The time-and-status unit also includes a third function called the Store Transfer Unit (STU). The STU is the only device that can write into the program store elements. For reasons of economy, it shares the same direct switching interfaces with the timing-generator and status units. New program segments flow from either tape or disc through the IOC to the STU and into the appropriate rack of program store via the internal switching network within the computing system. During recovery of the CLC, the STU associated with the IOC that is on-line at the time handles the reloading of the tactical software process into program store.

3.5 Instruction repertoire

The instruction repertoire for the CLC processor has been specified to accommodate the addressing structure of the computer. The processor can address program and variable store. A 20-bit internal address is used which, when mapped into actual memory addresses, allows addressing the maximum of 256 K double-words for both program and variable store. In addition, the processor contains internal register areas for temporary storage of operands. All arithmetic operations are performed on operands from the registers. The use of this

type of memory hierarchy separates the two functions of operand fetch from main storage and arithmetic execution. The instruction repertoire takes this into account so that the access of operands from variable store is distinct from arithmetic operations.

The addressing structure of the CLC will accomodate dynamic relocation of data sets. This requires that the processor have the capability to store and modify addresses locally within its registers. A method of double indexing is employed, using the contents of as many as two B-registers and a 12-bit displacement value contained within the instruction itself, to form an address value.

There are two different instruction lengths, 16 bits or 32 bits. Most instructions work with operands contained in the fast internal registers. The method of addressing operands from these registers is characterized by the use of two-address instructions with register addresses in the range of 0 to 15. These instructions use the half-word (16-bit) length which contains an 8-bit operation code and two register addresses. Instructions which access variable store utilize the longer 32-bit instruction length. In addition to the operation code and address displacement value, the memory access instructions also specify two B-registers used in address generation and the source or destination register in either the A, B, or Z register areas. There is also an instruction which references variable-store operands in absolute fashion using a full 20-bit address field contained within the instruction. In addition, a subset of instructions, designated "true" instructions, permit constants to be stored within the instruction itself. These constants may be directly loaded into the internal registers of the processor.

The processor can handle both fixed-point and floating-point data represented in fractional two's complement notation. All arithmetic operations are normally performed on 32-bit operands for both fixed- and floating-point data. Exceptions include a half-multiply instruction, the ability to manipulate exponents, and the ability to perform address arithmetic on 20-bit values. Floating-point numbers are usually normalized. There is no hardware capability to perform double-precision arithmetic.

3.6 Hardware concept

The SAFEGUARD hardware concept permits fabrication of the data-processing system from a standard stock of racks, chassis, and integrated-circuit packages. The design is based upon integrated-circuit technology using a modified direct-coupled-transistor-logic circuit having circuit delays in the 5- to 6-ns range. The hardware provides a flexible system for interconnecting groups of integrated-circuit packages on chassis, and chassis into racks as shown in Fig. 7. To enhance

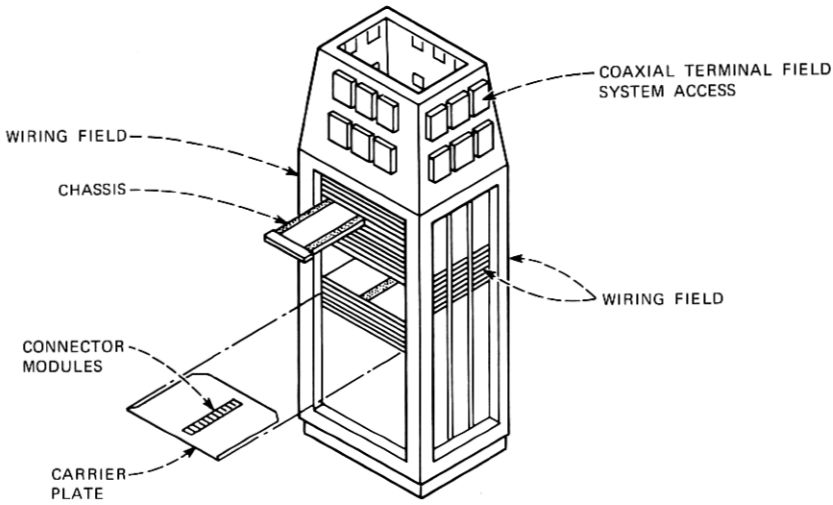

Fig. 7—SAFEGUARD rack.

reliability, the integrated-circuit packages are wire-wrapped to achieve connections on the chassis. Each chassis can accommodate 275 integrated-circuit packages and, therefore, more than 600 logic circuits. The chassis are housed in a water-cooled rack with two chassis mounted side by side on a chassis carrier plate which locates, supports, and cools the chassis. The chassis carrier plates are mounted on a 1-inch vertical pitch within the rack. There are a maximum of 59 levels in the rack housing 118 chassis.

It was recognized that a large multiprocessor would present a need for a large number of access connections. In fact, there is a need for more access connections to the chassis than could be provided with rear access only. Therefore, the chassis also uses both sides for additional access terminals. The rear contacts to the chassis are made in a conventional plug-in manner. The side contacts use a linear-actuated cam arrangement to engage the side contacts after the chassis has been situated properly in the rack. This arrangement results in wiring fields on three sides of the rack. In addition, internal connections are provided at the interface between the chassis, which are situated side by side on the carrier plate, to provide near-neighbor connections between groups of chassis. In total, the rack provides for more than 40,000 possible signal connections. It should be noted that having rack wiring on three sides has resulted in a diamond orientation of racks on a floor plan to allow physical access to all four sides of a rack. Rack-to-rack interconnections are provided by plug-in coaxial ter-

minal fields at the top of the rack which allow as many as 11,520 connections in this area.

To preserve the integrity of the high-speed pulse transmission between the various units that make up the multiprocessor, a characteristic impedance of 100 ohms is maintained for the transmission of all signals. Coaxial cables are used for all connections between racks and for all rack wiring runs in excess of five feet. Twisted pair is predominantly used to wire the rack. The chassis connector maintains a fixed impedance across the connection by providing both a signal and a ground path using a highly reliable double-contact arrangement to gain entry to a chassis.

The memory racks include a 16-K by 68-bit-per-word core memory unit and the associated interface logic switching circuits which provide interconnection to the multiple units in the system. The core memory units are air-cooled and operate at a cycle time of 500 ns and have an access time of 300 ns.

3.7 CLC performance

One of the primary reasons for the development of a parallel and modular computing system for SAFEGUARD is the potential for high performance. In addition to the properties this architecture possesses for high availability, a multiprocessor organization possesses a great deal of reserve power which, when applied to a problem with the appropriate degree of parallelism, can yield high performance. This is the type of problem which is associated with a radar tracking system and which must be solved in real time.

In a multiprocessor system, the processors gain access to main storage according to a priority rule. The rate at which each processor executes instructions depends, therefore, on the severity of this queuing at main storage. Throughput will be defined as the number of instructions of a particular instruction mix executed per second by n processors.

Adequate performance, or throughput, of a parallel processing system depends upon a number of hardware factors, which include the speed of the processor, the speed of program store including its priority circuit, the total number of processors relative to the total number of independently addressable program stores, and the number of instructions executed per memory word fetched. From a software viewpoint, the distribution of programs and data sets within the modular memory and the instruction mix of the particular programs in execution are also important factors which directly affect throughput.

Since variable store queuing will, in general, be less than that at program store, its effect has been eliminated in the throughput data

presented here. This has been done by dedicating a separate variable store rack to each processor for experimental studies.

Throughput data have been gathered using multiprocessor hardware with configurations containing as many as ten processors. Benchmark programs have been used which provide varying instruction mixes. Four instruction mixes were selected for testing. The NOP mix, consisting of no-operation instructions, defines an upper bound on throughput. The LOGICAL mix is a representative mix that is similar to CLC operating system code that might be executed during real-time operations. The MATH mix is also a representative mix, being a portion of the cosine subroutine from the CLC operating system. The JUMP mix consists exclusively of jumps and represents a kind of lower bound on throughput.

Figure 8 shows the effect of requiring all processors to execute out of one program store. The number of instructions executed per second increases with the number of processors until the program store is returning instructions as fast as it can. Throughput levels off when this point is reached, and a further increase in the number of processors does not increase throughput.

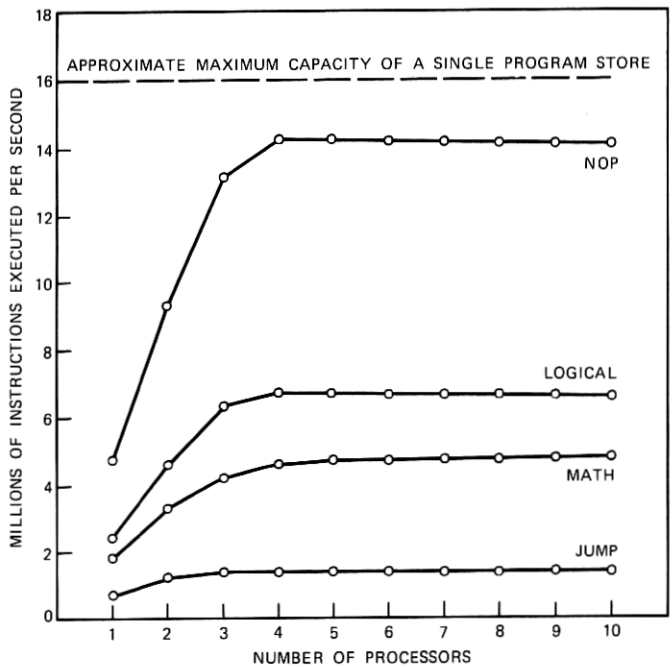

Fig. 8—N processors executing from one program store.

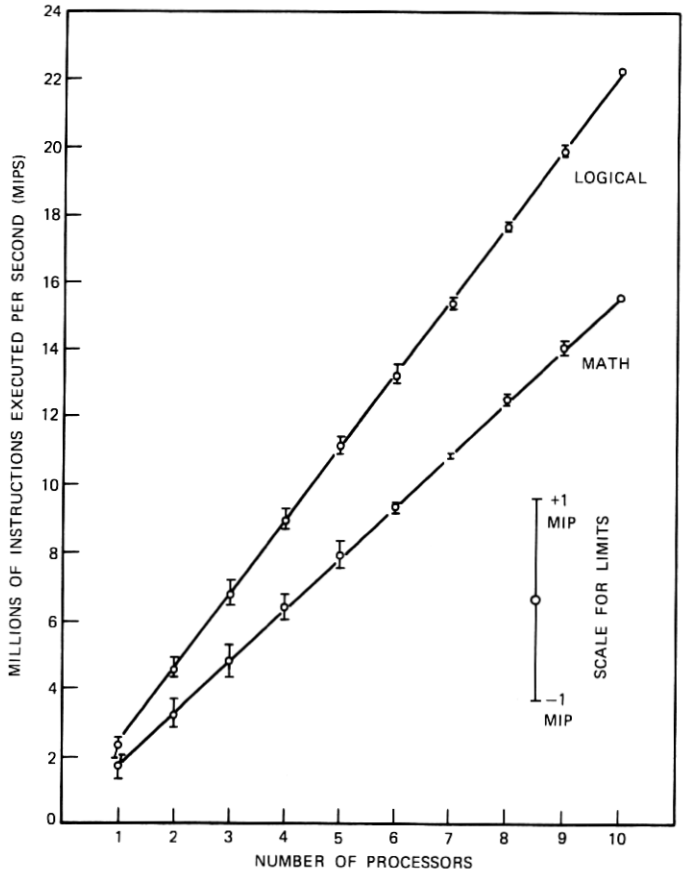

Fig. 9—N processors executing from N program stores.

Figure 9 shows the effect of providing an equal number of processors and program stores. For this case, the number of processors and program stores is incrementally increased from one to ten. The program stores are not dedicated to a processor on a one-for-one basis, but their access by the processors is randomized such that several processors may be attempting to read from the same program store at once. Hence, some reduction in throughput due to queuing is expected. The effect of queuing is small for one to ten processors. Figure 9 shows that throughput increases linearly with the number of processors. Data are shown for the LOGICAL and MATH mix only.

The data presented in Fig. 9 are for the case of an even distribution of memory access over all program stores. It is interesting to determine what happens to throughput for the case of an unequal work-load

distribution. A series of runs were made for both the LOGICAL and MATH mixes where the number of processors was kept equal to the number of program stores with one important difference. One of the program stores was selected as a "favored" program store and its fraction of total instructions executed was varied from 0 to 100 percent while the remaining program stores shared the remaining work load equally. Figures 10 and 11 show the results for the six to ten processor cases. The curves represent throughput as a function of the "favored" program store. Zero percent means ten processors are executing out of nine program stores. Note that throughput is a maximum when the "favored" program store shares equally in the work load.

The curves of Figs. 10 and 11 are useful in that they show the sensitivity of throughput to an unequal distribution of the work load in memory. For instance, if one considers a 10-percent reduction in throughput to be serious, the above curves show for the seven-processor case that a single program store can have almost 40 percent of the work load without a serious reduction in throughput. For the ten-processor case, the corresponding number is approximately 25 percent.

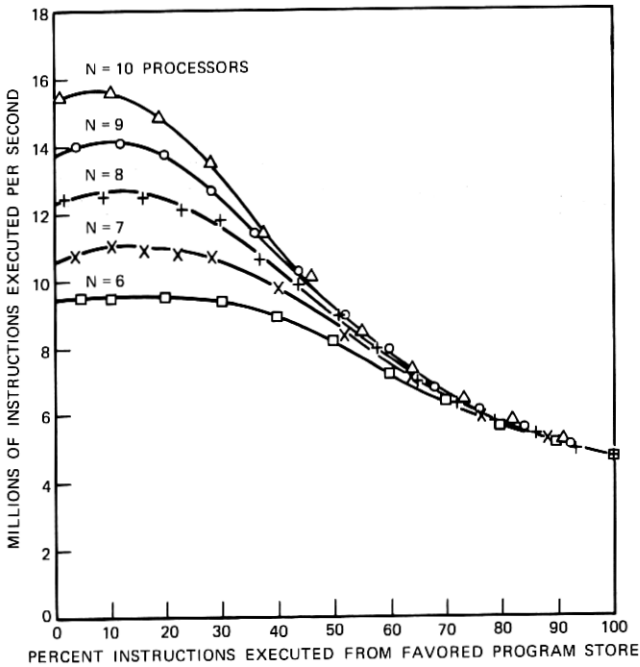

Fig. 10—Unequal program-store loading—MATH mix.

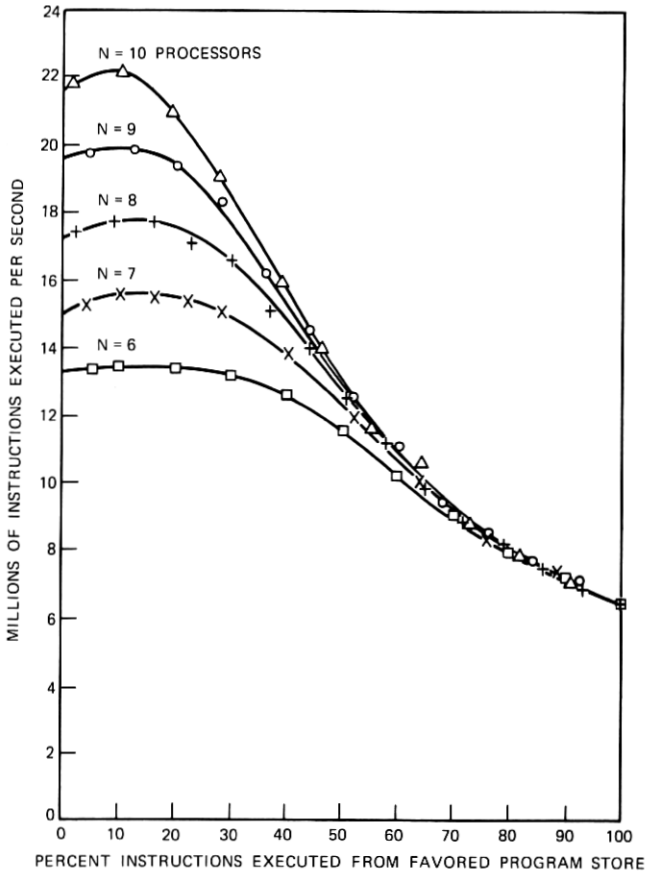

Fig. 11—Unequal program-store loading—LOGICAL mix.

Therefore, as long as the work load is not too unequally distributed, the dependence of throughput on work load distribution should not be critical. Throughput dependence on more than one program store having more than an equal share of the work load has not been investigated.

IV. CONCLUSIONS

4.1 Success of the modular design

The use of the well-defined interfaces and modular hardware building blocks capable of communication within the framework of a distributed switching system provides the basis for a dynamic computing complex—a structure that is capable of incorporating new functional units

offering unique economic or performance advantages.* This structure has been very useful in satisfying the wide range of computing applications within the SAFEGUARD system. These range from a single processor, nonredundant installation to a ten-processor, maximum-sized system. Not only does this structure handle the wide variations in system sizing, but it can easily accommodate changes that may result from new or revised system requirements.

4.2 Reduced cost for "n + 1" philosophy

Historically, early fault-tolerant systems, such as ESS-1, employed 100-percent redundancy through use of a complete standby system.[1] That is, the system required to support the full work load is duplicated, with data processing proceeding in parallel on each system. This organization is conceptually simple and upon detection of a failure in either system, the other system can carry on the data-processing work load.

The multiunit system approach to gaining high performance can provide high system availability without the need for costly, complete duplication. The $n + 1$ redundancy approach has reduced the amount of equipment added for redundancy and for system exercise to a fraction of that required for a complete standby system.

4.3 Instruction repertoire

The CLC instruction repertoire was designed long before CLC software was written. As a result, programmers seldom use certain instructions and often wish for others. For example, character manipulation instructions are lacking, as is one instruction that will store all processor registers.

4.4 Status-unit performance

The status unit, as implemented in the CLC design, represents a comprehensive method of gathering system status and providing configuration control information to the various parts of the data processing system. The use of the status unit to control the configuration of a partitionable machine is unique and has been proven successful during the SAFEGUARD project.

---

* This structure, for example, will very easily accomodate the addition of an array processor, such as the Parallel Element Processor Ensemble (PEPE), or it will easily allow direct connection of a high data rate peripheral subsystem to the modular variable stores. Although not a part of the present SAFEGUARD system, extensions to the multiunit architecture, as described above, have been seriously considered and are entirely feasible.

4.5 CLC performance

The performance of a multiprocessor system depends upon a number of factors including the speed of the processor, the speed of the memory element and the speed of its priority circuit, the total number of processors relative to the total number of independently addressable memory elements, and the number of instructions executed per memory word fetched. The distribution of programs and data memory and the instruction mix of the particular program being executed are also important. CLC performance as a function of the number of processors and the number of independent program-store data paths has been measured by D. B. Knudsen, and the information presented in Section 3.7 is a result of that effort.

V. SUMMARY

The requirement that a computer function properly even though some of its components fail has been a primary goal in the development of the SAFEGUARD computer. The multiprocessor approach was chosen to achieve high performance and availability. The multiunit architecture has provided a system which satisfies a wide range of computing requirements on the project through the use of a single design.

REFERENCE

1. R. W. Downing et al., "No. 1 ESS Maintenance Plan," B.S.T.J., *43*, No. 5, Part 1 (September 1964), pp. 1961–2020.