A Fixed-Program Data Processor for Banking Operations

J. GOLDBERG

ERMA is a large, special-purpose data-processing system designed by Stanford Research Institute for the Bank of America. The system includes both paper-handling and electronic data-processing functions, and its purpose is the keeping of records for commercial checking accounts. The first model will serve 32,000 active accounts and later will be extended to serve 50,000 active accounts.

In addition to keeping records and handling the paper checks and deposit slips, the system must reject overdrafts and stop-payment items, must provide the branches with a wide variety of random and scheduled information, and must proof-check the flow of data into the system. In order to meet the stringent time schedule of the system, the ERMA computer must do its work on-line. The computer was designed to fit the requirements of the system and, as a result, it is a highly specialized machine.

In its physical construction, the computer resembles modern digital machines: It stores data on magnetic drums and tapes, on keyboards, on punched paper tape, in relays, and in vacuum-tube flip-flops; its data are alphanumeric, are represented by binary-coded decimal digits, and are subject to arithmetic operations in an electronic adder.

In its logical organization, it bears little resemblance to digital computers of general purpose, either in addressing, in the structure of its central control, or in the role assigned to the arithmetic unit. The machine is permeated so deeply by the consequences of its special external requirements that it is of interest to trace the evolution of its logical design from these primary conditions.

ERMA actually contains four separate subcomputers, which operate simultaneously in the areas of magnetic drums, magnetic tapes, paper tape and a line printer. Each subcomputer has its special circumstances, and although all have their programming wired in, they differ substantially in their realization. The examples given here will be drawn only from the magnetic drum posting subcomputer, and will be used to illustrate the evolution of the following features:

1. Input items are processed using several large files stored on magnetic drums, and only a very small amount of fast-access storage is used.

2. Instead of having a single control unit performing operations serially, the work in the drum area is done by simple, special circuits, working simultaneously on different parts of the drum.

3. The programming for each mode of operation is wired into the central control circuits, and the wiring is switched as each mode is called up.

Description of Basic Operations

The primary task of the machine is to keep bank records for commercial checking accounts. The raw material for the data-processing function is a huge unsorted mass of paper checks and deposits, each addressed to a particular account. The end products are the familiar monthly statements for each account, complete with service charges, and a host of special lists and control totals for bank use.

Fig. 1 shows the major components and operations. Fig. 2 shows the drum posting area in greater detail. Five operators receive batches of customers' checks and deposit slips, and enter the items into the machine from their keyboards. At each board, the account number is read automatically from the paper to the keys and the operator depresses the dollar amount keys. The operator then presses a key instructing the machine as to the operation to be performed on the particular item on the board; for example, debit entry, stop payment posting, balance print-out, etc. ERMA automatically sweeps through the boards, processing the data on one board at a time. In the usual operation of debit entry, an incoming document, addressed to a given account, is subject to an acceptance test. If it is not rejected, it is posted; that is, it is subtracted from the "balance" file maintained for all accounts on the drum, and recorded separately on a special drum file. Later, it is sorted out for storage in the magnetic tape file kept for its account. At suitable intervals, service charges are calculated and posted, and statements are printed. Each day, special lists and balances are also issued for use. Also, the machine may be quizzed at random intervals concerning any customer's balance.

One of the basic requirements of on-line operation in ERMA is that the operators must have quick access to a large amount of information on four separate files. The "hold" and "stop-payment" files are relatively small, containing less than 1,000 17-digit words, but the file of all customers' balances contains over 50,000 10-digit words, making a total of over 500,000 decimal digits. Having these files available makes it possible for the operators to check each item immediately for overdrafts and stop-payments, and to provide instant information about the status of each account. Furthermore, having a separate record of each balance provides an excellent crosscheck on the transfer of items to permanent storage on the magnetic tape files. In order to provide a true check on overdrafts, the balance file must be continually updated by those debits and credits which are accepted for posting. Further, it is desirable to be able to post or remove hold items and stop-payment orders at any time.

All of these operations must be performed with perfect accuracy. In practice, this means that most transfers between storage media must be double-checked by repeated access. In order to keep up with the heavy flow of input data, the computer is allowed only 0.2 second to process each incoming item. This requirement of repeated access to such a large file in such a short time clearly leads to the use of magnetic drum storage. In ERMA, two large drums are used, to which the access time is relatively high (i.e., 33 microseconds maximum).

It is at this point that the ERMA computer breaks with general computer practice in that the logical design of the machine is tied very closely to a relatively slow-access memory. To the designer of a general-purpose computer, this dependence on a slow-access memory may be surprising for it is usually the character of the memory that has the greatest effect on machine structure. In a general-purpose machine, a slow drum is used only as a large back-up store, which occasionally delivers a block of data to a rapid-access working store. In this way, the slow-access memory need exercise only a secondary effect on machine design. Also, the usual general-purpose practice is to use a single arithmetic unit with the rapid access store, to perform many operations in sequence.

In ERMA, however, there is not enough time permitted to perform the repeated operations required on the "current-

J. GOLDBERG is with the Stanford Research Institute, Menlo Park, Calif.

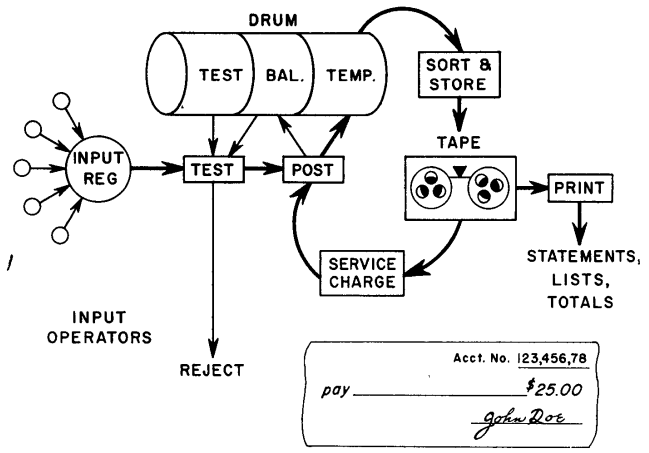

Fig. 1. Magnetic drum posting sub-computer

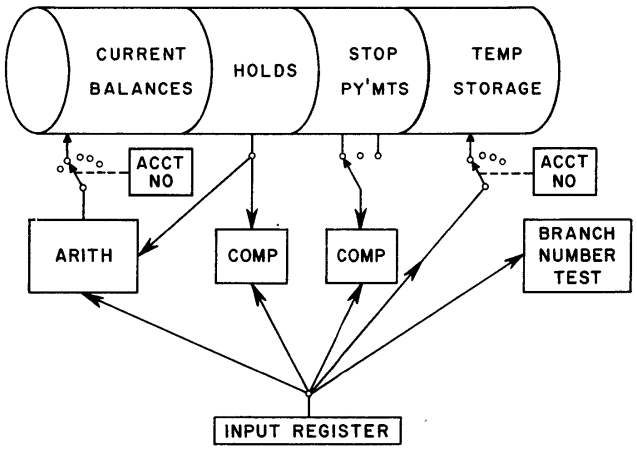

Fig. 2. Drum posting area

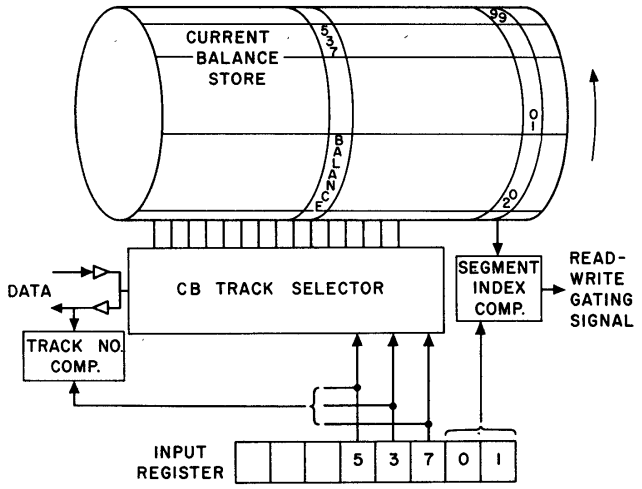

Fig. 3. Addressing method for "current balance" section of drum

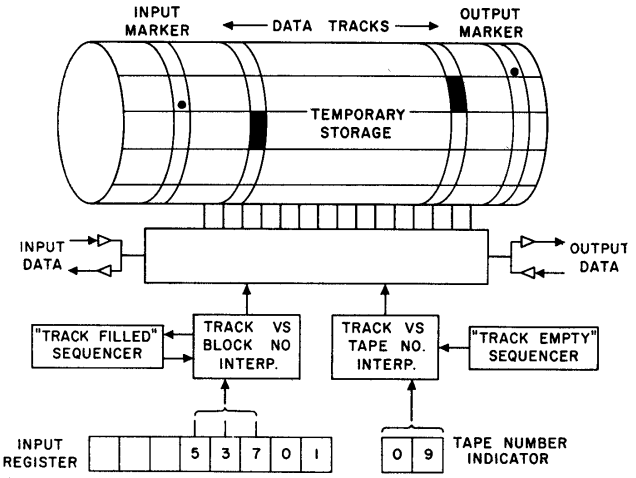

Fig. 4. Addressing method for "temporary storage" section of drum

"balance," "hold," "stop-payment," and "temporary-storage" files, all in sequence. This does not mean, however, that separate arithmetic units are needed for each file. The operations on the current-balance and temporary-storage files are quite simple, and may be accomplished by very simple circuits. On the other hand, because of the randomness of the data in the hold and stop-payment files, the operations of searching and selective extraction of items belonging to particular accounts are performed much more efficiently by a specially-wired search unit than by a programmed arithmetic unit.

The net result is that ERMA's arithmetic unit is used only to add and subtract dollar values, and the remainder of the work of the computer is performed by a number of simple, efficient circuits, operating simultaneously on the asynchronous data of the various files. Of course, each task could be done by a separate arithmetic unit and fast memory, but at prohibitive cost.

The following examples will describe some of the individual operations.

The Hold File

A word in the hold file consists of an account number, sign, symbol, and dollar amount, and every word in the file must be compared with the item in the input register. If the two words are identical, the entire process terminates. If the account number is the same, the dollar amount must be added to a running total of hold items.

These tests could be accomplished by transferring the file to a fast-access memory and applying the tests repeatedly using a programmed arithmetic unit. Even when the one revolution required to read the file from a drum into a fast-access memory is ignored, the application of a typical program of standard instructions would, by conservative estimates, require a clock rate roughly ten times that of the ERMA computer (152 kc).

In ERMA, the hold file operation is performed simultaneously on four independent drum tracks by a special comparator unit together with an arithmetic unit for the summation of "hold" values; the operations on each word are performed directly on the word as it is read on the drum, and the entire search takes one drum revolution. The comparator itself contains two flip-flops per track, with a moderate number of gate and buffer elements.

Current Balance and Temporary Storage Files

Posting a current balance consists of extracting one word from a specified address on the drum, performing three additions involving two words already in the arithmetic unit and the input register respectively, and returning the sum to the same drum address. Temporary storage

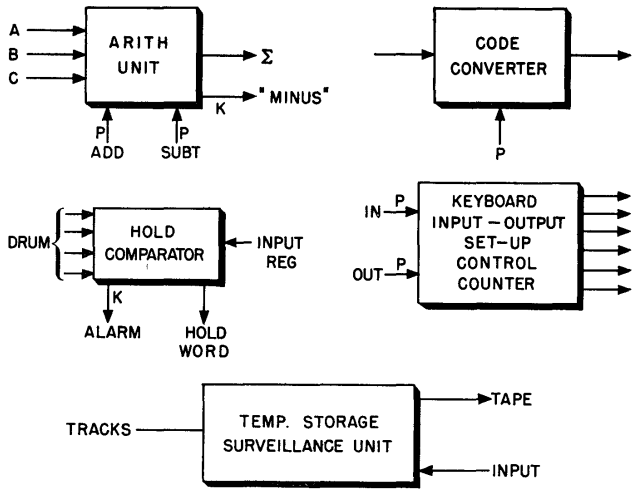

Fig. 5. Complex service units

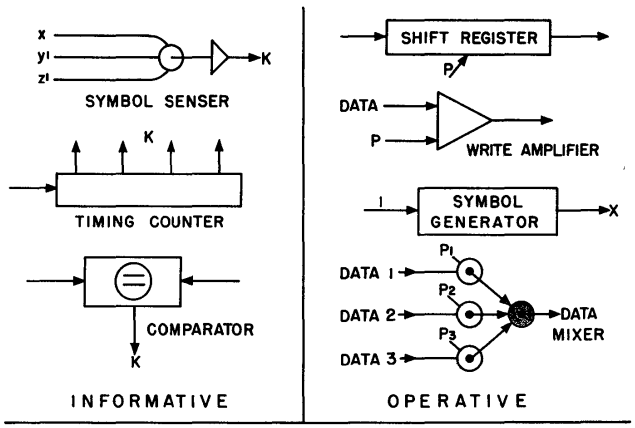

Fig. 6. Simple service units

P = control input
K = information output

posting consists of writing the input word in a specified track on the first empty space following a fixed-reference point. Both of these postings are followed by inverse operations for checking purposes.

It is true that a sequence of coded instructions could program these operations; however, for these functions the fast-access memory associated with it would be used primarily for operations on the program itself, rather than for the storage of data, and would not speed up the operation because the process is still limited by the need for random access to the files on the large drum store. Also, since the balance and temporary-storage files are asynchronous, two programmed units would be needed.

Fig. 3 illustrates the rather conventional addressing method used for access to the current balance (CB) section of the drum. The first three account number digits in the input register index a crossbar switch to connect the single "read-write" amplifier to the proper track (all drum data are serial-serial). The last two digits are compared with digits recorded on a coordinate track to produce a "read-write" gating signal for the desired balance word. The index number of each track are recorded on the track itself and is read as a check on the crossbar. Since only one track of the CB section is used for any one input process, the crossbar is set up only once in a routine.

Fig. 4 describes the addressing for the temporary storage (TS) section of the drum. The only thing of special interest temporary storage operations is addressing, since the incoming item is simply written on it its entirety, then read back for verification. In this case, the operation on the data is insignificant compared to the addressing problem. This store is the first stopping point for an item's record on the way to its particular magnetic tape file. It also serves as a medium for a sorting process.

Once the proper track is selected, the input word must be written in the first empty space found after the "band-start" point. This point is indicated by a movable pulse recorded on a special marker track. The actual track is selected by a special switching unit which simultaneously recognizes the status and demands of the input register, the drum, and the tapes. This unit assigns drum tracks to tapes on the basis of traffic load; it delivers filled tracks to permanent tape storage; and it channels incoming items to the proper drum tracks. Its internal structure is not important to an understanding of the drum-posting area.

Summary of Drum Addressing

To summarize the addressing situation in the drum area of the ERMA computer:

1. Fundamental requirements lead to the use of a large medium-access-time store.

2. Many processes must occur simultaneously. Because data appear asynchronously from different files, the access time is long, and there is not enough time allowed for sequential operation.

3. These processes may be accomplished efficiently by individual, special-purpose circuits which obviate the need for multiple arithmetic units and fast-access memories.

System Organization

Two basic facts have influenced the internal organization of the computer; first, many operations must occur in parallel, and, second, the machine must be able to switch instantly among many modes of processing data. Since the functions of control, arithmetic, and storage are included in almost all of the parallel working units, it is not too profitable to analyze the computer into these three areas. It is more instructive to divide the machine between "service" units and "program-control" units.

The service units perform the detailed tasks of transfer and transformation of data. They are specialized circuits, and quite varied, but their essential feature is that they act the same way in all routines. The function of the program-control units is to order the various service-unit operations in a manner characteristic of each specified routine. These routines contain the familiar minor cycles, the jump points, and the convergence points found in computer programs, but the program is actually wired into the control units. Since many operations are simultaneous, there are many data paths, so that a unique path must be specified by the control unit for each transfer. Also, drum data may appear with and without an account number, so that word operations may have different timing modules.

The drum-posting area has about 25 ways of processing the data in its files and input register. Examples are: debit posting, "stop-payment cancellation," "current-balance print-out," and "hold entry."

The basic operations in the typical routine are (1) transferring words among the various input and output registers, the internal electronic registers, and the various portions of the magnetic drum, (2) subjecting the data in the files and registers to tests of identity, relative size, and orders of position in files, and, (3) performing transformations on the data, such as arithmetic operations and code conversions on the whole, and on portions of a data word.

As described previously, some of these jobs are done entirely by specially wired

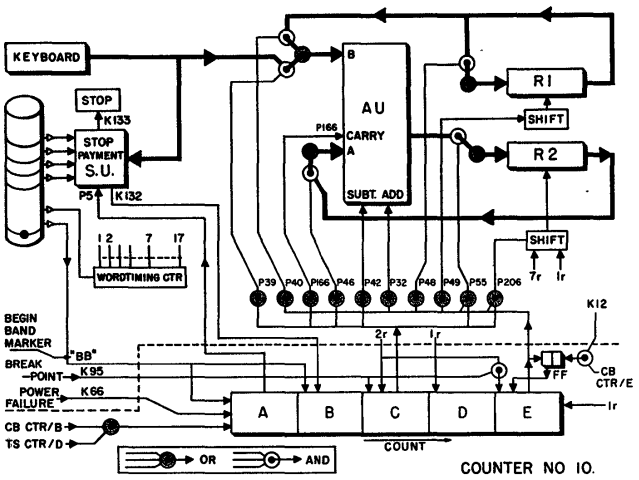

Fig. 7. Segment of drum-posting control set up for "debit entry" routine

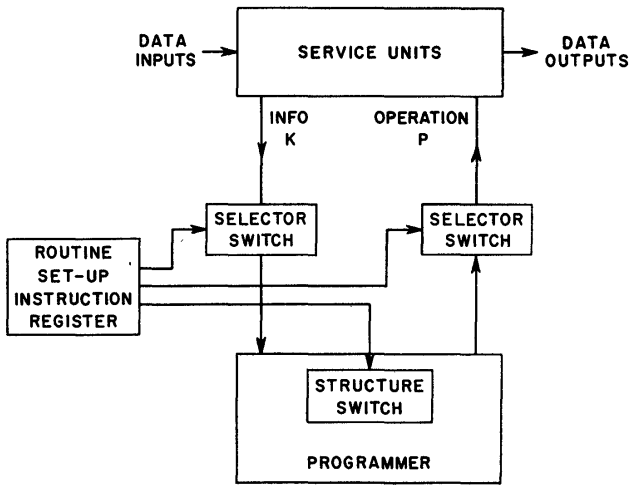

Fig. 8

subcontrol units. Examples shown in Fig. 5 are: the "hold" comparator and extractor, an arithmetic unit, a code converter, a keyboard input-output relay-control unit, and the temporary-storage surveillance unit.

Some simpler service units are shown in Fig. 6. Some, such as symbol sensers, comparators, and counters provide special information concerning the nature of signals observed; some, such as writing amplifiers, shift registers, and symbol generators, perform a special operation on data; some do both. It has been convenient to call channels carrying informative signals from service units "*K* leads," and those carrying instruction signals to the service units "*P* leads."

The Structure of the Program Control Units

The program-control unit is a sequential switching circuit consisting of flip-flops, counters, and logic elements prewired according to the prescribed program. It is driven by a set of input signals which are descriptive of the states of all of the service units and timing counters, and in response it produces a sequence of output signals which trigger one or more service units to perform their respective functions and specify data paths as needed. For operations performed by a complex service unit, the control unit acts merely to trigger or enable the operation. There are, however, various detailed operations which do not occur frequently enough to merit incorporation into a service unit. These are timed directly by the program control unit, with the result that the control has no steady rhythm.

Fig. 7 is a segment of the drum-posting control as it is set up for a debit entry routine. The following are significant:

1. The central control in this case (enclosed by dotted lines) is a ring-type counter together with a single flip-flop The arrows directed into a given counter stage are those conditions, taken in an AND sense, required to advance the counter into the indicated state. The counter is ring-like in that there is only one state on at once, and that the count proceeds in order from one state to its adjacent state, but the counter does not recycle. This counter is triggered to its first state by either of two other counter states elsewhere in the control unit. Thereafter it proceeds according to the *K* leads, timing leads, and interval control leads applied to its individual states. A state may last for from one clock cell interval (6.5 microseconds) to several drum revolutions. The timing pulses used in operating on the digits of a word are taken from a 17-state ring counter, synchronized with the drum.

2. The work of the states is done by energizing "*P*" buffers. Buffers are used to allow numerous counter states to energize the same service units. In this example, states *A*, *C*, and *E* are working states, while *B* and *D* are synchronizing states.

a. State *A* lasts for one drum revolution, by using the "begin-band" pulse as two successive advance conditions. It energizes *P5*, which gates on the "stop-payment-search" service unit to examine the SP file for the item set up in the input keyboard Search failure (*K* 132) permits the counter to advance.

b. State *B* serves only to delay the program until the word-timing counter cycles to the state suited for starting an arithmetic counter cycles to the state suited for starting an arithmetic operation.

c. State *C* energizes all "*P*" buffers required to subtract the keyboard item from the present contents of register 2 and store the difference back in register 2. Three of its loads establish the data paths used, and the remainder control the operations. The shifting of register 2 is precisely timed by its shift service unit over counts 8 to 1, inclusive, of the 17-state word-digit timing counter. The remaining control leads need merely overlap this interval roughly, so that state *C* may be triggered as early as the second count of the digit-timing counter.

d. State *D* delays the program until the digit-timing counter again reaches the second count, it provides a waiting state for the completion of some other concurrent operation as sensed by the *FF*, and permits break point operation if it is requested (*K* 95).

e. State *E* energizes all "*P*" buffers required to add the contents of register 1 and register 2, and store the sum in register 2. Note that some of the service units driven are the same used by state *C*.

3. The provision of individual advancing conditions for each state has several reasons and uses.

a. There is a need to intersperse long drum searches with single word operations of different modules in order to perform the required input item tests in the allotted time. This requires flexible timing choices.

b. Since several counters may be operating simultaneously, provision is needed for interlocking them at critical junctures.

c Break points may be assigned with great flexibility.

d. It is possible to apply feedback from the driven service unit to the driving states, so that the counter jams at the point of failure.

The program control unit has a different method of operation corresponding to each routine; each method has its own decision points, using the service units in some special order. As each new word enters the input register, the associated selection command causes a large switching to take place which essentially rewires large portions of the central control area, although many segments remain unchanged. The switching is done with relay contracts. The switching affects the control in three ways, as indicated by the three switches in Fig. 8:

1. It interconnects the components to

represent the logical structure of the operations to be performed (i.e., the decision points, the sequence of operations following each alternative, the convergence points, and the parallel operations).

2. It directs to each state the timing signals and service-unit status signals required for initiation and termination.

3. It connects each active state to drive one or more service units, with the following possibilities:

a. Several states may drive the same service unit through buffers.

b. A control state may either drive on a service unit directly, or may gate the output of one service unit to trigger another.

c. A control state may drive any number of service units; if a data transfer is required, the state may have to energize all gates required to complete the data paths when the service unit triggered is not restricted to a specific data path.

d. A control state also may do no external work, but act only as a synchronizing or delay state.

The counter shown in the foregoing example will be used in other routines. Some uses may be identical, but the counter may be pieced together with other counters in different arrangements. Some uses may be similar; for example, the work of one state may be inhibited by transferring its output bus to the zero level with a relay contact. In other uses, the majority of the input and output leads may be switched. An initial survey of the number of operations performed in consecutive groupings led to the switching of counters in blocks of four states each. The drum posting program unit handles its 26 routines with 19 counter blocks and 24 flip-flops. The longest routine uses eight counter blocks and 20 flip-flops.

Error-Checking Features

The fact that ERMA is a bank accounting machine and operates on-line means that errors are not merely inconvenient. Because all error cannot be prevented, and because ERMA must have all accounts in perfect balance by the end of the working day, errors must be found and corrected as quickly as possible. Since the best time to correct an error is when the paper document is in the hands of the operator, error checking should not be postponed. Because of the on-line nature of its operation, downtime must be minimized.

The following are some general techniques utilized in error-checking the drum-posting area:

1. Parity-check monitors are located at the output of all registers and drum-read amplifiers and may be switched onto commonly used busses. Each binary-coded decimal digit carries an even-parity redundancy bit.

2. Every addition or subtraction is followed by the opposite operation, using the sum read back from its permanent store.

3. All non-arithmetic postings, such as "temporary storage" write-on, are followed by reading back from the drum and comparing with the original items. If the source is a keyboard, the comparison is made using duplicate contacts on the key stems.

4. For certain lists, the keyboard is used as an output printer. Here, each key setup is verified by reading the key contacts back for comparison with the original source.

Summary

The ERMA computer is neither a stored program computer nor a plugboard computer, but it does contain some features of each. It is like a plugboard machine in that its program steps are wired in, and it is like a stored programmed machine in that each mode is called up by a coded instruction, which, however, is not subject to modification by the program.

It is a special purpose machine in that its detailed logical design directly reflects the external operational requirements. It needs a very large drum store to perform its operations on-line; the volume of data handled is large; many steps are required for processing each item and for accuracy checks; the data on the various files is essentially random, and the time allowed is short. Taken together, these conditions require performing numerous operations simultaneously rather than serially; each operation, however, can be performed by a simple unit operating at the moderate drum clock rate. Establishing service units which are unchanged in the various machine modes and switching the interconnections of the central control circuits to change programs provides a relatively inexpensive way of conducting the various modes of the data handling processes.

The Logical Design of a 1-Microsecond Parallel Adder Using 1-Megacycle Circuitry

A. WEINBERGER J. L. SMITH

Synopsis: The logical design of a parallel adder is developed which is capable of adding two 53-bit numbers in 1 microsecond. The design makes use of basically the same 1-megacycle circuitry which has been used successfully in the National Bureau of Standards' SEAC and DYSEAC computers. An analysis of the functional relationships of the carry digits to the augend and addend digits shows that it is feasible to form many carries simultaneously at the expense of relatively few components. The Boolean expressions for many successive carry digits can be expanded as explicit functions of some one lower-order carry, and of the relevant augend and addend digits. These somewhat complicated expressions are simplified by making substitutions for the common terms and factors they contain. These common terms and factors, called auxiliary carry functions, are implemented separately. All functional forms fit within the wide limits of gating complexity allowed by the type of circuitry to be used.

A. Weinberger and J. L. Smith are with the National Bureau of Standards, Washington, D. C.

THE development at the National Bureau of Standards of the diode capacitor memory,[1,2] which is capable of being read or written into at the rate of one word per microscecond, has made it worth while to build devices capable of processing information at comparable rates. Since the basic micro-operation common to most arithmetic processes is the adding together of two numbers, it seemed reasonable to design an adder having a cycle time no greater than 1 microsecond.

The major timing bind in an adder is in the production of carries, and in this paper the problem is attacked from the standpoint of logical organization. Although