IRE Transactions

on ELECTRONIC COMPUTERS

Volume EC-6 **Number 1**

CONTRIBUTIONS

The Logic of Bidirectional Binary Counters . . . . . *Marcel J. E. Golay* 1

The Logical Design of a Simple General Purpose Computer . . . . . *Stanley P. Frankel* 5

A Transistor-Driven Magnetic-Core Memory . . . . . *E. LeRoy Younker* 14

Current Steering in Magnetic Circuits . . . . . *J. A. Rajchman and H. D. Crane* 21

An Electronic Analog Multiplier Using Carriers . . . . . *Erich S. Weibel* 30

CORRESPONDENCE

Picard's Method and Analog Computation . . . . . *Rajko Tomovich* 34

Extending the Operational Time of Analog Integrators . . . . . *George H. Myers* 34

An Integral-Error-Squared Method for Evaluating Analog Computer Components . . . . . *Robert A. Bruns* 35

A Note on the Accuracy of Differential Analyzers . . . . . *A. Fuchs* 36

REVIEWS

Ways of Developing Soviet Computer Production 37

PGEC Membership Survey . . . . . *W. L. Martin and S. R. Olson* 49

Review of Electronic Computer Progress During 1956 55

Contributors 60

PGEC News 61

Reviews of Current Literature . . . . . *Harry D. Huskey* 63

PUBLISHED BY THE

Professional Group on ELECTRONIC COMPUTERS

The Logic of Bidirectional Binary Counters*

MARCEL J. E. GOLAY†

Summary—**The counters without short-time internal memory, conceived by Bigelow, disclosed by Ware, and extended by Brown, are discussed from the standpoint of their respective logic. It is shown that the (self-instructed) bidirectional[1] counter of Brown has a more rigorous logic than the unidirectional counter of Ware; the operation of Brown's bidirectional counter being subjected to the only restriction that its speed be compatible with the operating speed of its individual toggles, whereas the operation of Ware's counter is predicated upon the existence of unspecified buffering states in the input of each stage, to prevent run-away conditions. These buffering states, which occur naturally in Brown's bidirectional counter, can be provided explicitly in unidirectional counters by replacing the two transfer circuits of Ware's counter stage, which are controlled only by one toggle of the preceding stage, by two of the four transfer circuits of Brown's bidirectional counter stage, all four of which are controlled by both toggles of the preceding stage.**

This paper introduces the viewpoint that a bidirectional counter of Brown's type is a counter in which the state of one toggle of each stage determines which toggle of the next stage is master, while the state of the other toggle of each stage determines whether the slave of the next stage shall be like or unlike the master. This viewpoint permits a succinct discussion of the several possible interstage connections, and of the several counting codes obtained for each connection. In particular, it can be shown with a minimum of steps that the code exhibited by the "true" toggle is always binary, or a simple modification thereof, while the code shown by the "false" toggle is always the Gray code, or a simple modification thereof.

Introduction

In a first article,[2] Ware discussed the concept, based on a principle originally suggested by J. H. Bigelow, of unidirectional counters without short-time internal memory. This new type of counter employs a pair of toggles in each stage and transfer circuits, which alternately cause toggle F_i to be like toggle T_i, and toggle T_i to be unlike toggle F_i. This arrangement differs basically from the usual type of Eccles-Jordan bistable toggle stage, for the latter obeys the single command: "Change," which, if analyzed in terms of the operations required, means, "Remember for a short time what you are now, and become different from that." The requirement of a short-time memory in the single toggle counter stage derives from the first half of the order written above, whereas in Ware's counter stage the previous state of the toggle which has just changed is stored in the other toggle until the next command. Ware noted that two additional transfer circuits with separate instructions could permit a counter to operate bidirectionally. But in a subsequent article,[3] Brown noted that four internally instructed transfer circuits would permit a counter to reverse its counting direction, without separate instructions.

The purpose of this discussion is to examine the basic difference between the logic of Ware's unidirectional and Brown's bidirectional counters, and to introduce a point of view which permits a succinct representation and classification and a direct examination of the several types of internal logic which are possible with these bidirectional counters.

Discussion

The counter stage of Ware, which consists of two toggles and two causal connections (gates, transfer circuits) between these, is characterized by a curious difference between its input and its output. Its input is essentially ternary, for it may be either a "1" from the preceding T_{i-1} toggle, which commands toggle F_i to be like toggle T_i, or a "0" from the T_{i-1} toggle, which commands toggle T_i to be unlike F_i, or a "neither," which implies that "0" and "1" must not coexist during changes.[4] This third buffering condition, not explicitly provided in Ware's counter stages, is required to prevent the first two conditions from overlapping, for such an overlap would cause a "runaway" condition. Conversely, the stage output is double binary, for it consists of the two states of each toggle. Only half of this output, *i.e.*, the state of the T toggle, is utilized as input to the following stage, without explicit provision for the third buffering state, the "neither" condition mentioned.

In contrast, Brown's bidirectional counter stage is characterized by an input and an output which are both double binary; this provides automatically the buffering transition states just mentioned, as seen below.

Fig. 1 illustrates one of several possible cause and effect connections between an $(i-1)$st and an ith stage in a bidirectional counter of the type described by Brown. These connections are represented by four compartments in a plane, whose coordinates are the voltages in the X_{i-1} and Y_{i-1} toggles, which are conveniently represented by the same symbols.[5]

* Manuscript received by the PGEC, May 25, 1956; revised manuscript received October 10, 1956.

† Consultant to the Philco Corporation, Philadelphia, Pa.

[1] In this discussion, the expression "bidirectional counters" will refer exclusively to selfinstructed bidirectional counters, the expression "reversible counters" being reserved for counters requiring separate instructions for a reversal of the direction of counting.

[2] W. H. Ware, "The logical principle of a new kind of binary counter," Proc. IRE, vol. 41, pp. 1429–1437; October, 1953.

[3] R. M. Brown, "Some notes on logical binary counters," IRE Trans., vol. EC-4, pp. 67–69; June, 1955.

[4] The physical interpretation of this requirement, in the case of an electronic counter, is that the two conductors carrying the voltages of the two plates of the $i-1$ toggle should never both carry, during a change of the $i-1$ toggle, a voltage high enough to operate simultaneously the two respective transfer circuits they control.

[5] The X and Y symbols are used in preference to Ware's and Brown's T and F in order to maintain symmetry in the presentation, and to avoid a premature conclusion as to the respective toggle functions. These symbols are variously used to designate a toggle, or the quantized, "0" or "1," state of a toggle, or again some "analog" parameter of a toggle such as voltage. Simplicity without ambiguity has resulted from this multiple use of the same symbols.

The four compartments shown in Fig. 1 are nearly quadrant shaped, as "and" gates[6] would make them, and within each, one of the following four conditions exists: if $X_{i-1}=1$ and $Y_{i-1}=1$, then $Y_i \to X_i$ (X_i is made like Y_i); if $X_{i-1}=0$ and $Y_{i-1}=1$, then $\bar{Y}_i \to X_i$ (X_i is made unlike Y_i), etc. Instead of inactive buffer zones, an overlap of active zones has been illustrated, in order to emphasize the immunity of this system to zone overlap as long as the state of the input is represented by a point which travels from quadrant to adjacent quadrant by crossing two lines only in the $X_{i-1}Y_{i-1}$ plane. For instance, the overlap of the $Y_i \to X_i$ and $\bar{Y}_i \to X_i$ zones means that during a transition, the slave X_i toggle is temporarily subjected to conflicting requirements, but, since the master Y_i toggle remains unaffected, the situation is resolved unambiguously at the end of the transition. Likewise, the overlap of the $X_i \to Y_i$ and $Y_i \to X_i$ zones means that during a transition, both toggles are master and slave simultaneously, but to the same end, namely likeness of their states, so that no action can take place during this transition.

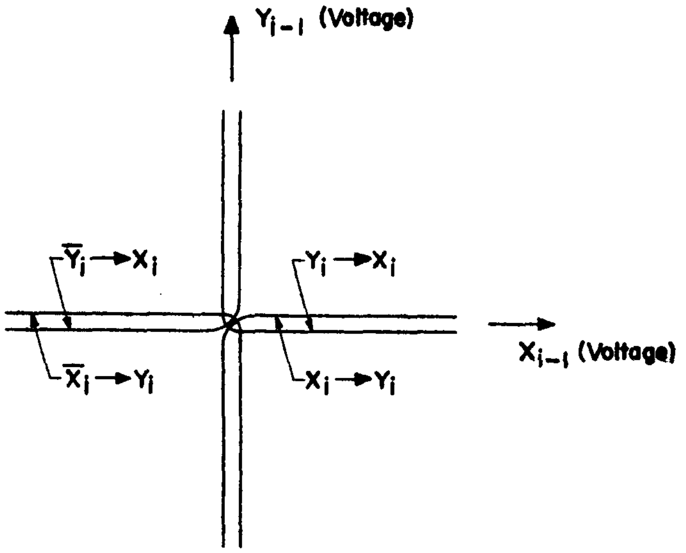

Fig. 1—Cause and effect from $(i-1)$st to ith stage by means of "and" gates.

The immunity of the system to zone overlap permits replacing the "and" gates by simple gates, each controlled by the addition of two voltages. For instance, the "and" gates which perform the logical function.

$$\text{when } X_{i-1} = 1 \text{ and } Y_{i-1} = 1$$
$$\text{then } Y_i \to X_i$$

can be replaced by a simple gate which performs the logical functions

$$\text{when } X_{i-1} + Y_{i-1} > A$$
$$\text{then } Y_i \to X_i$$

where X_{i-1} and Y_{i-1} in the inequality designate the voltages of the $(i-1)$st stage toggles, and where A is so chosen that the gate is opened when these voltages are both high, and only then.

The straight line $X_{i-1}+Y_{i-1}=A$ determines the half plane to the right of and above it, in which the transfer $Y_i \to X_i$ is activated, and has been illustrated in Fig. 2, together with the other three lines along which the three voltages $X_{i-1}-Y_{i-1}$, $-X_{i-1}-Y_{i-1}$, and $-X_{i-1}+Y_{i-1}$ determine similar boundary conditions for the respective operation of the other three gates: $\bar{Y}_i \to X_i$, $\bar{X}_i \to Y_i$, and $X_i \to Y_i$.

When Figs. 1 and 2 are compared, it is noted that the nearly quadrant-shaped zones of Fig. 1 are replaced by half-planes in Fig. 2, in which there are now four quadrant zones of double causation, four strip zones of single causation, and an internal rectangular zone of no causation. Clockwise or counter-clockwise input changes can take place along the dotted rectangle shown, which may either cross or not cross the quadrants.[7]

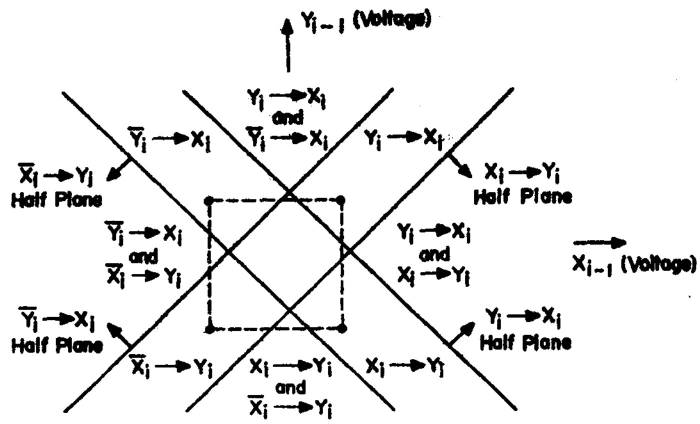

Fig. 2—Cause and effect from $(i-1)$st to ith stage by means of simple gates.

It follows from the foregoing that a unidirectional unambiguous counter can be realized by providing only the two gates corresponding to opposite strips, which are required for a given direction of rotation of the $X_{i-1}Y_{i-1}$ point (*e.g.*, the $\bar{X}_i \to Y_i$ and $Y_i \to X_i$ gates in the case of a clockwise rotation of the $X_{i-1}Y_{i-1}$ point), but that the full unambiguity of such a counter is predicated upon each operating gate being controlled by both toggles of the preceding stage, either on an "and," or on a voltage addition basis.

It can be concluded from the above discussion of the counter logic illustrated by both Figs. 1 and 2 that the causality between the $(i-1)$st and ith stages is exactly as if the Y_{i-1} state determined whether X_i or Y_i is

[6] The pair of gates required for a single transfer circuit between two toggles will be referred to simply as a gate.

[7] If the $X_{i-1}Y_{i-1}$ point travels clockwise starting at the lower right-hand corner of the rectangular path illustrated, where $X_{i-1}=1$, $Y_{i-1}=0$, and if $X_i=Y_i=0$ at the start, the X_iY_i points resulting from two complete travels of $X_{i-1}Y_{i-1}$ on its rectangular path will be: 00, 01, 01, 11, 11, 10, 10, 00, 00.

master, while the X_{i-1} state determines whether the slave shall be like or unlike the master in the ith stage. Thus, the cause and effect relationship between one stage and the next described by Figs. 1 and 2 may be illustrated succinctly by Fig. 3, which shows which toggle of any stage is master, depending upon the state of the preceding Y toggle, and whether the slave shall be like ("0") or unlike ("1") the master, depending on the state of the preceding X toggle.[8] It is immediately apparent that there are eight ways of establishing the stage-to-stage causality. The next phase of this discussion will determine the various counting codes represented by the sequence of X and Y states during each of the eight possible counting operations.

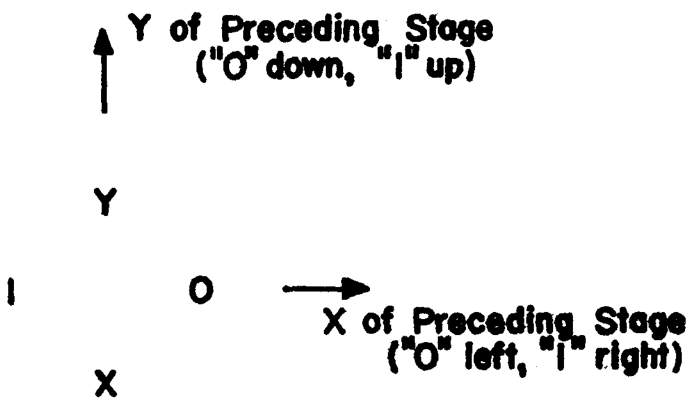

Fig. 3—Simple diagram of cause and effect between stages.

Inspection of Fig. 3 indicates that a counterclockwise succession of states in the $X_{i-1}Y_{i-1}$ plane is characterized by a $\frac{1}{4}$ cycle lead[9] of the X_{i-1} toggle with respect to the Y_{i-1} toggle, and that the leading toggle in the ith stage will be the one which demands likeness when it is master; *i.e.*, the one followed by a zero, when Fig. 3 is traversed counterclockwise. It follows that the cyclical order of the states will be the same in successive stages when the order X–0–Y–1 is counterclockwise, whereas they will alternate when the order X–0–Y–1 is clockwise.

It will be noted that one of the toggles will change whenever there is a change in a toggle of the preceding stage which causes a 0–1 or a 1–0 transition in Fig. 3 (the X toggle for the case shown). This is a property of the binary code, and the code exhibited by the 0–1 axis toggles will be a binary code, whenever it is the same toggle transition in any one stage which causes the homologous toggle in the next stage to change; *i.e.*, whenever the transitions, which cause transitions in the homologous toggle in the next stage, have like directions in all stages, corresponding, as was seen above, to a counterclockwise X–0–Y–1 order. Otherwise, the code formed will be the binary code in which the alternate binary digits are added 1 (mod. 2), and will be termed "altered binary." The binary code will read up for clockwise state transitions whenever it is the 1–0 transition which causes the next 0–1 axis toggle to change, and will be reversed and read down otherwise.

Following Ware's nomenclature, and designating by T (for true) the toggles in the 0–1 axis and by F (for false) the others, the state of the toggles in the $(i-1)$st and the ith stage are bound by the condition

$$F_i + T_i + T_{i-1} \equiv C \text{ (mod. 2)} \qquad (1)$$

where C designates which of 0 or 1 is at the $T_{i-1}=0$ coordinate. (In Fig. 3 $T=X$, $F=Y$, $C=1$.)

Relation (1) indicates that, when $C=0$, and when T forms the binary code, the $T_n, F_n, F_{n-1}, \cdots, F_1$ ensemble forms an $n+1$ digit Gray code, when there are n stages (F_1 is not determined by the nonexistent T_0, but, together with T_1, determines what T_0 would have been had there been a preceding 0th stage). When $C=1$, and when T forms the binary code, the code obtained for F will be the Gray code, to all binary digits of which 1 has been added (mod. 2) and will be termed the reentered Gray code.

Likewise, when $C=0$, and when T forms the altered binary code, the reentered Gray code will also be obtained for F, whereas the Gray code is obtained for F from the altered binary code of T when $C=1$.

Except for the inclusion of T_n, the Gray and reentered Gray codes thus obtained correspond to the False Rank numbers of Brown, which, in turn, correspond to those of Ware, with the further exception of the binary digit of lowest rank, because Ware does not tabulate what Brown terms $\frac{1}{2}$ counts.

The several codes which can be obtained with the eight possible state-to-state connections can be deduced from the preceding discussion and have been tabulated in Table I. As an example, consider the fifth counter connection:

$$\begin{array}{ccc} & 0 & \\ X & & Y, \\ & 1 & \end{array}$$

the detailed causal connections of which are: if $X_{i-1}=0$ and $Y_{i-1}=0$, then $\overline{X}_i \rightarrow Y_i$; if $X_{i-1}=1$ and $Y_{i-1}=0$, then $\overline{Y}_i \rightarrow X_i$; if $X_{i-1}=1$ and $Y_{i-1}=1$, then $Y_i \rightarrow X_i$; if $X_{i-1}=0$ and $Y_{i-1}=1$, then $X_i \rightarrow Y_i$. The clockwise $X0Y1$ order indicates that the XY point rotation will alternate from stage to stage, and that, together with the circumstance that the 01 axis is the Y axis, determines that the code formed by the Y axis will be the altered binary code.

[8] For instance, when $Y_{i-1}=1$, Y_i is master, regardless of the value of X_{i-1}; and, when $X_{i-1}=0$, the slave of the ith stage is unlike the master, regardless of the value of Y_{i-1}.

[9] In the successive states:

$$\left.\begin{array}{l} X = 0\ 1\ 1\ 0\ 0\ 1 \\ Y = 0\ 0\ 1\ 1\ 0\ 0 \end{array}\right\} \cdots X$$

will be said to lead.

TABLE I

SYNOPSIS OF BIDIRECTIONAL COUNTER CONNECTIONS AND CODES

Causality	X code	Y code	Alternating Rotations
Y 1 0 X	Binary	Reentered Gray	No
X 1 0 Y	Altered Binary	Gray	Yes
Y 0 1 X	Reversed Altered Binary	Reentered Gray	Yes
X 0 1 Y	Reversed Binary	Gray	No
0 X Y 1	Gray	Reversed Altered Binary	Yes
1 X Y 0	Gray	Reversed Binary	No
0 Y X 1	Reentered Gray	Binary	No
1 Y X 0	Reentered Gray	Altered Binary	Yes

Furthermore, we have $C=1$, because there is a 1 at the $Y=T=0$ coordinate, and, since the Y code is reversed binary, the X code will be the Gray code.

If the counter has four stages, their successive states will be as follows for an assumed input, which causes a counterclockwise rotation of the first stage, and for the initial conditions: $X_1=Y_1=X_2=X_3=X_4=0$.

$X_1=F_1$	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0 ←
$Y_1=T_1$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0 ←
$X_2=F_2$	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0 ←
$Y_2=T_2$	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1 ←
$X_3=F_3$	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0 ←
$Y_3=T_3$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0 ←
$X_4=F_4$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0 ←
$Y_4=T_4$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1 ←

→ Gray Code
→ Altered Binary Code

Or, with separate grouping of the altered binary and Gray code digits:

Altered Binary Code

Y_4	Y_3	Y_2	Y_1
1	0	1	0
1	0	1	0
1	0	1	1
1	0	1	1
1	0	0	0
1	0	0	0

Gray Code

Y_4	X_4	X_3	X_2	X_1
1	0	0	0	0
1	0	0	0	1
1	0	0	1	1
1	0	0	1	0
1	0	1	1	0
1	0	1	1	1 etc.

CONCLUSION

Brown has noted that a bidirectional counter of the type discussed here may be used to determine unambiguously[10] the position of a shaft. The logic of this counter justifies the view that true electrical gearing is realized with it and is analogous to the mechanical gearing of a binary counter with Geneva gears. Its operating speed is limited by the properties of the toggles, just as the speed of the mechanical counter is limited by the mechanical properties of its parts. The essential difference is that a mechanical counter has the memory of a miscount in the form of a broken gear. The absence of a similar memory in the electrical counter, for which an interruption of power spells the same disaster as a broken gear in a mechanical counter, is the basis for the engineer's distrust of electrical gearing in operational equipment of a highly responsible character such as, for example, a gun director. On the other hand, the enormous savings which could be effected by electrical gearing should make it attractive in a host of applications. Thus, simulating equipment for storing operational problems, in which several shaft positions are recorded with a full binary number, could be considerably reduced in bulk, if only the changes were noted, in the form of the X_1Y_1 input of a reversible counter.

Other applications suggest themselves, such as the reckoning of the traversing motion of a machine tool, the count of the positive or negative frequency difference of two oscillators, one of which may be slaved to the other much more gently than if capture had to occur within a quarter cycle, the tracking of Doppler counts to measure displacements or distances, etc.

The more complete logic of the bidirectional counter leads one to speculate as to any basic difference between the kind of problems for which unidirectional or separately instructed reversible counting only is required, and those for which bidirectional counting fulfills better the needs of the situation. A rough rationalization may lead to the view that the first are mathematical problems, and the second are physical ones.

[10] Unlike separately instructed reversible counters, selfinstructed bidirectional counters do not require that the "ripple" of a count in any one direction have terminated, before the counter may be instructed to count in the opposite direction, after which the counting operation in that new direction may proceed.

The Logical Design of a Simple General Purpose Computer*

STANLEY P. FRANKEL†

Summary—**The logical design described here is used in MINAC, partially constructed at the California Institute of Technology, and LGP-30, manufactured by Librascope Inc. These serial binary digital computers make use of magnetic drum bulk storage and use three circulating registers and fifteen flip-flops. The procedures used in performing the sixteen elementary operations are described. These descriptions indicate the circumstances in which each flip-flop or circulating register input is activated. The Boolean algebraic equations summarizing these circumstances constitute the logical design.**

Introduction

THE LOGICAL design described here was largely composed in the course of work of the Digital Computing Group of the California Institute of Technology. A breadboard model of a computer based on this logical design, called MINAC, was completed at C.I.T. in 1954 and served to check much of the design. A production version of this machine, called the LGP-30, has been completed by Librascope Inc. of Glendale, Calif. Although MINAC and LGP-30 differ in a few details of logical design the present description is substantially correct for either.

The LGP-30 has been discussed in two previous publications. One[1] describes its elementary operations and the ways in which these are used to perform complex calculations. The other[2] discussed the useful range of applications of magnetic drum computers in general, with particular reference to LGP-30. It is the purpose of the present paper to present in almost complete detail the logical design structure held in common by MINAC and LGP-30. Their constructional techniques and methods of arithmetic manipulation are described only to the extent necessary to this purpose.

Constructional Techniques

The primary memory device of MINAC is a magnetic drum. Information is held on the drum in three forms. The bulk memory is held in 64 tracks, each served by one "head" which records data in it and can subsequently read the recorded data. The information recorded in each track consists of 64 *words*, each of 32 bits (binary digits). A second type of memory of shorter access time is provided by three *circulating registers*, each consisting of a recording head and one or more reading heads following it (in the sense of drum rotation) in the same track. The time during which the 32 bits of a word are presented by a head of the bulk memory is called a *word period*. The time elapsing between the recording of a digit in a circulating register and its presentation by a reading head is about 32 digit periods so as to permit recirculation of information in one word period. The third form of information storage on the drum is represented by the *timing tracks*. Each of these is served by a single reading head which reads permanently recorded information determined only by the angular position of the drum. The digits presented by three timing tracks are combined to form various timing signals, denoted t, u, v, x, y, z. These are shown in Fig. 1.

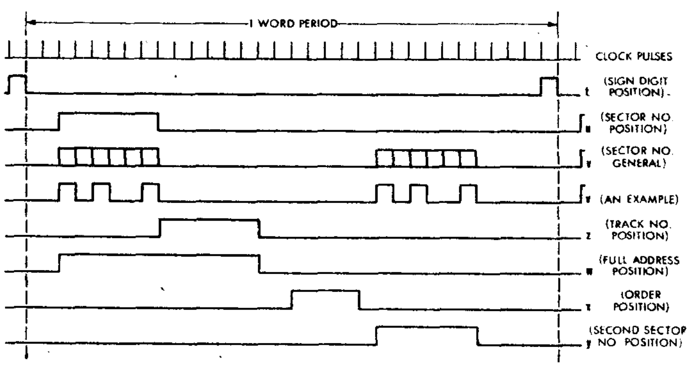

Fig. 1—Signals derived from the timing tracks.

Information which must be presented continuously over many digit periods is held in *toggles* (flip-flop circuits). Each of these can be set to one or the other of two stable states, designated 1 and 0, at the end of each digit period. The logical design consists primarily of the specification of the circumstances under which each toggle is set to 1 or to 0. If neither input is activated the toggle retains its prior setting during the next digit period. The logical design also specifies the digit recorded in each circulating register in each digit period and whether a digit is to be recorded in the bulk memory and, if so, what digit and in which track.

The input and output of data are mediated by a Flexowriter, a punched paper tape controlled typewriter. In the input process each character read from the tape sets some of the MINAC toggles. In output the state of the toggles controls the firing of a set of thyratrons which effect the closure of the relays by which the Flexowriter is operated.

* Manuscript received by the PGEC, June 1, 1956; revised manuscript received October 16, 1956.
† 1764 Redondo Ave., Long Beach, Calif.

[1] S. Frankel and J. Cass, "The Librascope general purpose computer, LGP-30," *Instruments and Automation*, vol. 29, pp. 264–270; February, 1956.

[2] S. Frankel, "Useful applications of a magnetic drum computer," *Elec. Eng.*, vol. 75, pp. 634–639; July, 1956.

The logical design is realized by a *logical network*, composed primarily of resistors and crystal diodes. Its inputs are the settings of the toggles expressing the present internal state of the computer proper (as distinct from the memory held on the drum) as well as the digits being presented by the timing tracks, circulating registers, and bulk memory of the drum. Its outputs go to the recording heads of the circulating registers, to the selected head of the bulk memory (if a recording operation is in progress), to the inputs which set the toggles for the next digit period, and to the Flexowriter thyratrons. Each input or output is, for each digit period, a binary (Boolean) variable. Each input variable is presented in duplicate, *e.g.*, by wires from the two sides of the toggle. This facilitates the expression of the logical design primarily by the monotonic Boolean functions *And* and (nonexclusive) *Or* that are easily realized by diode networks.

Functional Design

The action of MINAC consists of a series of elementary operations, each performed upon one word of its memory and determined by one instruction word. An instruction word holds an *order* indicating the operation to be performed and one *address* identifying the word of the memory involved in the operation. Instructions held in successively numbered memory locations are normally obeyed in succession. Exceptions to this rule occur for "control transfer" instructions which specify the address at which the next instruction is to be sought. One circulating register, the *counter*, primarily recirculates the address in which the next instruction is to be found. As an instruction is read from the memory it is recorded in the *instruction register* where it is retained to direct the execution of that instruction.

The third circulating register, the *accumulator*, holds a number—usually the result of the last arithmetic operation. In an arithmetic operation the number held in the accumulator is combined with a number drawn from the bulk memory and the result retained in the accumulator. The accumulator normally recirculates with a period of one word period, like the counter and instruction register. During the performance of a multiplication or division the capacity of the accumulator is increased to two word periods by the use of a second reading head.

Sixteen elementary operations are provided, as shown in Table I. The arithmetic operations act upon signed (algebraic) numbers, represented in binary expansion as described below.

TABLE I

Instruction Order List Showing Code for Each Instruction

	Code	Instruction	Effect
Arithmetic	0001	B m*	Bring. Clear the accumulator, and add the contents of location m to it.
	1110	A m	Add contents of m to the contents of the accumulator, and retain the result in the accumulator.
	1111	S m	Subtract the contents of m from the contents of the accumulator, and retain the result in the accumulator.
	0111	M m	Multiply the number in the accumulator by the number in memory location m, terminating the result at 30 binary places.
	0110	N m	Multiply the number in the accumulator by the number in m, retaining the least significant half of the product.
	0101	D m	Divide the number in the accumulator by the number in memory location m, retaining the rounded quotient in the accumulator.
	1001	E m	Extract, or logical product order, *i.e.*, clear the contents of the accumulator to zero in those bit positions occupied by zeros in m.
Transfer Control	1010	U m	Transfer control to m unconditionally, *i.e.*, get the next instruction from m.
	1011	T m	Test, or conditional transfer. Transfer control to m only if the number in the accumulator is negative.
Record	1100	H m	Hold. Store contents of the accumulator in m, retaining the number in the accumulator.
	1101	C m	Clear. Store contents of the accumulator in m and clear the accumulator.
	0010	Y m	Store only the address part of the word in the accumulator in memory location m, leaving the rest of the word undisturbed in memory.
	0011	R m	Return address. Add "one" to the address held in the counter register (C) and record in the address portion of the instruction in memory location m. The counter register normally holds the address of the next instruction to be executed.
Misc.	0100	I	Input. Fill the accumulator from the Flexowriter.
	1000	P †	Print a Flexowriter symbol. The symbol is denoted by the track number part of the address (x).
	0000	Z †	Stop. Contingent on five switch ($T_1 \cdots T_5$) settings on the control panel.

* The address part of the instruction is denoted by m when it refers to a memory location, by † when only the track number is significant. For example, m might be 4732, meaning Sector 32 of track 47.

Requirements for Continuous Memory

Several phases of the operation of the computer, each terminated at the end of a word period, are to be distinguished: in phase 1 (abbreviated $\phi 1$) the instruction next to be obeyed is sought. This requires selecting the appropriate track and waiting for the desired word in that track to appear. In $\phi 2$ which occupies just one word period the instruction word is set into the instruction register. In $\phi 3$ the *operand word* (*i.e.*, the word in the bulk memory designated by the address then held in the instruction register) is sought by a process similar to that of $\phi 1$. In $\phi 4$, which lasts for one word period, the operation is executed, except for the prolonged operations, multiplication and division. For these the execution extends into phases 5, 6, 7, and 8. These extend the time for execution somewhat beyond the time for one drum revolution.

Except in prolonged operations the phases are distinguished by two toggles, named *F* and *G*. Phases 5 to 8 are distinguished from phases 1 to 4 by the use of a third toggle, *H*.

While an instruction or an operand word is being read (or written) the appropriate track of the bulk memory must be selected. Six toggles, named P_1, $P_2, \cdots, P_6$, perform this selection. They are set serially during phases 1 and 3, then remain unchanged during the word period in which the track selection must be exercised.

In a similar fashion, the order must be continuously in evidence during phases 4 to 8. Since $16=2^4$ orders are to be distinguished, four toggles suffice to mark them. They are denoted Q_1, Q_2, Q_3, Q_4.

Two more toggles are used, K and L. The primary duty of L is to hold carry digits in addition and subtraction processes. (Since these operations are performed serially, only one carry bit need be "remembered" in each digit period.) The search operations of phases 1 and 3 are performed with the help of toggle K, as described below.

During the execution of a multiplication or division (after $\phi 4$ in which the multiplicand or divisor is read into the instruction register) there are additional requirements for continuous memory. These requirements are met by the P toggles, which after $\phi 4$ are not needed in their track selecting capacity. The functions of the toggles are summarized in Table II.

TABLE II
TOGGLE FUNCTIONS

Name	Primary Function	Other Functions
F G H	Phase discrimination	
K	Sector search	K used for augmenting control address
L	Carry digit	
Q_1 Q_2 Q_3 Q_4	Hold order (Cf. Table I)	$Q_2=1$ in $\phi 1$ for blocked state
P_1 P_2 P_3 P_4	Track selection	P_1 marks odd word in periods $\phi 5$ to $\phi 8$ P_1 to P_4 hold character in input
P_5 P_6		P_5 holds sign of multiplier or divisor P_6 holds digit of multiplier or sign of remainder

THE DEVELOPMENT OF THE LOGICAL DESIGN

Except for the occurrence of prolonged operations, H stays in its 0 state, as indicated by the symbol $\mathbf{H}$. The first four phases are distinguished by the four states of F and G, as follows:

Phase	F State	G State	Symbol	Duration (word periods)
$\phi 1$	0	0	$\mathbf{F}\mathbf{G}\mathbf{H}$	1 or more
$\phi 2$	0	1	$\mathbf{F}G\mathbf{H}$	1
$\phi 3$	1	0	$F\mathbf{G}\mathbf{H}$	1 or more
$\phi 4$	1	1	$FG\mathbf{H}$	1

The "product" of two or more symbols indicates the simultaneous occurrence of the indicated settings. Thus the symbol for $\phi 2$, $\mathbf{F}G\mathbf{H}$, has the value 1 only if $F=0$ (hence $\mathbf{F}=1$) *and* $G=1$ *and* $H=0$.

The last digit period of a word period is marked by a signal denoted by t, derived from the timing tracks. Since each phase change occurs at the end of a word period, the symbol t is included as a factor in each of the F and G setting equations. The phases 1 to 4 occur cyclically in the order listed. Toggle F has the state 0 during phases 1 and 2. It is set to the 1 state at the end of $\phi 2$. It holds the state 1 during $\phi 3$ and $\phi 4$, then is set to 0 as $\phi 1$ is again entered. Since $\phi 2$ and $\phi 4$ consist of only one word period each the settings of F are easily described. F is set to 1 at time t of any $\phi 2$ period, to 0 at t of a $\phi 4$ word period. The circumstances calling for setting it to 1 are denoted F', those calling for setting it to 0 are denoted $\mathbf{F}'$. (The prime here indicates the new condition of the toggle. It should not be mistaken for negation, which is here indicated by boldface.) Thus we have the two following partial equations (the + following an expression indicates that other equations show contributions to that input):

$$F' = \mathbf{F}G\mathbf{H}t +, \tag{1}$$

$$\mathbf{F}' = FG\mathbf{H}t +. \tag{2}$$

G is set to 0 on either of these two occasions on which F is changed. This may be written as

$$\mathbf{G}' = \mathbf{F}G\mathbf{H}t + FG\mathbf{H}t +.$$

The symbol + may be read as *or*. If either or both terms joined by + have the value 1 so also does the sum. By an elementary operation of logical algebra this $\mathbf{G}'$ expression may be reduced to

$$\mathbf{G}' = G\mathbf{H}t +. \tag{3}$$

(In the following no explanation of algebraic manipulations will be given.)

G is set to 1 at the ends of phases 1 and 3. The system provided for determining the durations of these phases is described below. It makes use of toggle K which will be found in the state 1 at the time t only for the last word period of either of these phases. The end of a phase 1 or 3 may thus be recognized by the occurrence of $\mathbf{G}\mathbf{H}Kt$. Accordingly,

$$G' = \mathbf{G}\mathbf{H}Kt +. \tag{4.1}$$

(The presence of a decimal fraction in the expression number indicates that a revision of this term is introduced below.)

The Instruction Search

In $\phi 1$ a search is conducted for the instruction whose address is being recirculated in the counter. The digits presented by the counter are denoted C. The part of the address (six bits) which determines the track selection is set up on the P toggles by a process described below. The remaining six bits of the address determine which word of the selected track is wanted, hence the time at which $\phi 1$ should end. The six digit periods in which this *sector number* is presented by C are marked by a signal u derived from the timing tracks (cf. Fig. 1). Another timing track signal v presents a particular sequence of six digits for each of the 64 word periods of a drum revolution. In each word period it "announces" the sector number of the word period following immediately thereafter. To determine whether a word period of $\phi 1$ is to be the last word period of that phase the digits announced by v are compared with the digits presented by C during the six digit periods marked u. Agreement in all six digits calls for termination of $\phi 1$ at the end of that word period. To detect this agreement toggle K is set to 1 at the end of each word period;

$$K' = t; \tag{5}$$

thereafter disagreement sets it back to 0.

$$\mathbf{K}' = \mathbf{FGHu}(v\mathbf{C} + \mathbf{v}C) +. \tag{6.1}$$

Thus finding K in the state 1 at time t indicates that agreement has been found, as was assumed in the discussion leading to (4.1). [It is to be noted that the input described by (5) brings K to the 1 state only *after* the digit period in which it is examined in (4.1).]

The Operand Search

In $\phi 3$ a similar process is carried out, differing only in that the address of the word sought is carried in the instruction register, which presents the digits R, rather than in the counter. The two search processes are thus described by (5) and (6.2).

$$\mathbf{K}' = \mathbf{GHu}(v\mathbf{r} + \mathbf{v}r) +, \tag{6.2}$$

where r is C during $\phi 1$ and R in $\phi 3$.

$$r \equiv FC + FR. \tag{7.1}$$

Track Selection

Like the time of entry to phases 2 and 4, the track to be selected is indicated by the address circulating in C during $\phi 1$, or R during $\phi 4$. In either case it is indicated by the digits r defined above. The six digit periods of each word period during which the track number part of an address is presented by C or R are marked by the signal z. During z in phases 1 and 3 the track number r is set into the toggles $P_1, P_2, \cdots, P_6$. For this purpose these six toggles are connected as a *shifting register;* the digits r are inserted into P_1 and passed down the chain to P_2, etc. The digit periods in which this setting takes place are denoted p;

$$p \equiv GHz +. \tag{8.1}$$

During this time P_1 is set to the digit r,

$$P_1' = pr +; \qquad \mathbf{P}_1' = p\mathbf{r} +, \tag{9.1}$$

P_2 is set to the digit P_1, *etc.*

$$P_2' = pP_1 +; \qquad \mathbf{P}_2' = p\mathbf{P}_1 +, \text{ etc.} \tag{10}$$

At the end of z, P_6 holds the first (least significant) digit of the track number, P_5 the second, etc. These settings are retained for the remainder of that word period and, if that word period terminates phase 1 or 3, into the succeeding phase 2 or 4.

Order Setting

In $\phi 2$ the instruction is read from the main memory. The digits presented by the main memory are denoted V. The instruction is set into the register R during $\phi 2$ and recirculated there during the subsequent $\phi 3$. This is represented by the equation,

$$R'' = FGHV + FGHR +. \tag{11.1}$$

Here R'' denotes a digit being recorded in the instruction register. Similarly digits set into the accumulator and counter are denoted A'' and C''. (They are not toggle inputs like the singly primed symbols.) The instruction also includes a four digit order which is set into the Q toggles in $\phi 3$ in the same way as the track number is set into the P toggles. The four digit periods in which the order appears are marked by the signal x. Then

$$Q_1' = \mathbf{FGH}xR; \qquad \mathbf{Q}_1' = \mathbf{FGH}x\mathbf{R}; \tag{12}$$

$$Q_2' = \mathbf{FGH}xQ_1 +; \quad \mathbf{Q}_2' = \mathbf{FGH}x\mathbf{Q}_1 + ; \text{ etc.} \tag{13}$$

With a few exceptions, described below, these settings of the Q toggles are held without change until the next occurrence of a $\phi 3$.

Accumulator Input

The execution of orders in $\phi 4$ is chiefly concerned with the behavior of the accumulator. It recirculates its content without change in the first three phases; and also in $\phi 4$ on orders U, T, H, Y, R, P, and Z. As may be verified by use of Table I, these orders are collectively described by:

$Q_1\mathbf{Q}_3\mathbf{Q}_4 + \mathbf{Q}_2(\mathbf{Q}_3 + \mathbf{Q}_4)$ occurs for U, T, H, Y, R, P, Z.

Thus the normal recirculation of A is described by,

$$A'' = AH[F + G + Q_1\mathbf{Q}_3\mathbf{Q}_4 + \mathbf{Q}_2(Q_3 + \mathbf{Q}_4)] + . \tag{14}$$

For the remaining orders the input to A in $\phi 4$ is as follows: on order B, A is set to V; on order E to the product AV. Together these may be described as $\mathbf{Q}_2\mathbf{Q}_3Q_4(\mathbf{Q}_1+A)V$. On orders A and S, described by $Q_1Q_2Q_3$, the output of the add-subtract mechanism, here abbreviated b, is set into the accumulator. On orders M, N, and D the accumulator content is recirculated unchanged in $\phi 4$ except, for reasons described below, for the omission of the sign digit (at time t). This input to the accumulator is described by $\mathbf{Q}_1Q_2(Q_3+Q_4)A\mathbf{t}$. In the Input process four digits have (prior to $\phi 4$) been read from the Flexowriter tape and set into the toggles, P_1, P_2, P_3, P_4. In $\phi 4$ the accumulator content is recirculated through these four toggles; P_1 following A, P_2 following P_1, *etc.*, and A'' following P_4. This contributes a term $\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3Q_4P_4$ to A''. To induce the motion of digits down the chain of P toggles, expression (8.1) is replaced by

$$p \equiv FGHz + FGHz(Q_1 + Q_2 + Q_3 + Q_4) + FGHQ_1Q_2Q_3Q_4. \quad (8)$$

For order I the last term in p produces the transfer of digits in $\phi 4$, while the parenthetical factor in the second term suppresses the usual transfer in $\phi 3$. Eq. (9.1) expressing the setting of P_1 must also be modified as

$$P_1' = pGr + pGA + ; \quad P_1' = pGr + pGA + . \quad (9)$$

Altogether the inputs to A in phases 1 to 4 are described by (14) and by

$$A'' = AHQ_1Q_2(Q_3 + Q_4)t + FGH[Q_2Q_3Q_4(Q_1 + A)V + Q_1Q_2Q_3b + Q_1Q_2Q_3Q_4P_1] + . \quad (15)$$

Instruction Register Input

After $\phi 3$ it is no longer necessary for the instruction register to retain the instruction. For the prolonged orders it is used to store the multiplicand or divisor (read from the bulk memory in $\phi 4$). Thus we change the R'' equation as follows:

$$R'' = FGHV + FGHV + (G + H)R = GHV + (G + H)R. \quad (11)$$

Counter Input

In the part of each word period marked $w \equiv u + z$ the counter holds the address of the instruction next to be obeyed. This information is read, and acted upon, in $\phi 1$. To prepare for the next use of this *control address* it is augmented by unity in $\phi 2$. This operation is performed with the help of toggle K, which, as described above, is set to 1 at the end of each word period. In $\phi 2$, K is set to zero whenever the digit 0 occurs in the control address; that is, on the occurrence of wC,

$$K' = FGHwC + , \quad (16.1)$$

while the counter content is complemented whenever $K = 1$.

$$C'' = FGHw(KC + KC) + . \quad (17)$$

The change in control address produced by the U order is effected by transferring the content of the instruction register to the control register in $\phi 4$ on this order.

$$C'' = FGHQ_1Q_2Q_3Q_4R + FGH(Q_1 + Q_2 + Q_3 + Q_4)C + . \quad (18)$$

Test Execution

The order T is to have the same effect as U provided the sign (t) digit position of the accumulator is occupied by a 1 or if the corresponding digit of the instruction is a 1 and the *external transfer switch*, z_0, is closed. These two circumstances are expressed by $t(A + Rz_0)$. To effect the transfer the Q_4' equation is given the term

$$Q_4' = Q_1Q_2Q_3(A + Rz_0)t + , \quad (19)$$

which transforms a T into a U-order in $\phi 3$. (These two orders are distinguished only by the setting of the Q_4 toggle.)

The Blocked State

To provide a way of stopping a computation a "blocked state" is introduced. This is done by making the advance from $\phi 1$ to $\phi 2$ contingent on the occurrence of $Q_2 = 1$. Then (4.1) is replaced by,

$$G' = GHKt(F + Q_2) + . \quad (4)$$

In any situation not requiring blockage Q_2 is set to 1 (or allowed to remain at 1) on entering $\phi 1$. When blockage is required Q_2 is set to 0, then $\phi 1$ lasts indefinitely. The *start button* effects a release from blockage by setting Q_2 to 1. A variety of circumstances produce blockage: the stop order, Z, induces blockage contingent on the settings of external switches and of the P toggles as set by the address accompanying the Z order. A *one-operation* switch causes blockage after each operation. An overflow in an addition or subtraction or an improper division causes blockage, so as to show that the correct result cannot be represented in the usual way. Provision is made to produce a blocked state on first turning on the computer, and as an aid in timing the Input process. All of these effects are peripheral to the operation of the computer and will not be further described here.

Addition and Subtraction

On orders A and S a sum or difference is set into the accumulator in $\phi 4$. The digits of the sum or difference will be denoted b. The Add-Subtract mechanism makes use of toggle L to hold carry digits. It has been set to 0 prior to its use in addition or subtraction. The two inputs are denoted i and j. [In subtraction the number (j), *i.e.*, the number formed by the digits j, is subtracted from (i).] In addition the carry digit, L, is set to one following the simultaneous occurrence of 1-digits in the two inputs. It is set to 0 if i and j are 0. If i and j differ the setting of L is left unchanged. Thus a carry is initiated by ij, is terminated by ij, and is propagated by ij or ij. However, L is always set to 0 at the end of the operation, after the t digit period. Thus the carry in addition is described by

$$L' = ijt, \quad (20.1)$$

$$L' = ij + t. \quad (21.1)$$

A digit of the sum, b, is the sum modulo 2 of the digits i, j, and L. Thus

$$b \equiv Lij + Lij + Lij + Lij. \quad (22)$$

In the execution of orders A and S, occurring in $\phi 4$, the two inputs are A and V respectively. Thus

$$i \equiv AH + , \quad (23)$$

$$j \equiv VH + . \quad (24)$$

These equations, together with (15), describe the performance of an addition. This process of addition is most easily understood if each number is regarded as expressed in a simple binary expansion with the least significant digit appearing first and the most significant digit appearing at time t. For example the digit at time t might be assigned the value unity, the preceding digit the value $\frac{1}{2}$, the one before that the value $\frac{1}{4}$, etc. Actually, a different system for the interpretation of numbers is normally used in this computer. It differs from this only in that the digit at time t is assigned the value -1, which permits representing signed (algebraic) numbers in the range -1 to (but not including) $+1$. Either interpretation is consistent with the above description of addition or with the process of subtraction described below. However the system for introducing blockage on an improper addition or subtraction (*i.e.*, one which produces a result beyond the capacity of the representation) is made to conform to the signed number interpretation. So also are the processes of multiplication and division.

With this signed number interpretation a number, say (x), formed of digits x is approximately the negative of the number formed from the complementary digits, $\mathbf{x}$, (where $\mathbf{x}=1-x$). More precisely, $(\mathbf{x})$ lacks one unit in its least significant digit to be $-(x)$. Thus if $(\mathbf{j})$ were added to (i) the result would be nearly the difference, $(i)-(j)$. By complementing i rather than j, a similarly deficient difference, $(j)-(i)$, is obtained. The correct desired difference, $(i)-(j)$, can now be obtained by complementing that sum. Accordingly the rule for subtraction is obtained from the equations above by replacing i by $\mathbf{i}$, and then complementing (22) for the sum digit. These two changes, however, bring (22) back to its original form. Thus (22) describes the result of subtraction as well as of addition, while the carry equations are replaced by

$$L' = (is + \mathbf{is})jt, \tag{20.2}$$

$$\mathbf{L}' = (\mathbf{is} + is)\mathbf{j} + t. \tag{21.2}$$

Here s indicates situations in which a subtraction is performed, $\mathbf{s}$ an addition. The codes for Add and Subtract differ only in the setting of toggle Q_4, hence

$$s \equiv \mathbf{H}Q_4 + . \tag{25}$$

(In phases 5 to 8 other conditions determine s as well as i and j.)

Multiplication and Division

At the end of $\phi 4$, F and G are set to 0 as described above [(2) and (3)]. This usually initiates a $\phi 1$. However, on orders M, N, and D toggle H is set to 1 at the same time, thus producing a $\phi 5$. This is described by

$$H' = \mathbf{FGH}t\mathbf{Q}_1Q_2(Q_3 + Q_4). \tag{26}$$

In phases 5 to 8 a succession of arithmetic processes is carried out during successive intervals of time each extending over two word periods. Toggle P_1 is used to mark off these pairs of word periods. It is set to 0 at the end of $\phi 4$, thereafter to 1 and 0 alternately. This alternation is expressed by the terms,

$$P_1' = \mathbf{H}\mathbf{P}_1t + ; \qquad \mathbf{P}_1' = GP_1t + \mathbf{H}P_1t + . \tag{27}$$

Phase Changes for Multiplication and Division

The marking and durations of phases 5 to 8 are as follows:

Phase	Marked	Duration
5	$\mathbf{FG}H$	2 word periods
6	$F\mathbf{G}H$	61 word periods
7	FGH	2 word periods (for M and D only)
8	$\mathbf{F}GH$	1 word period (for D only)

The return to $\phi 1$ occurs after $\phi 6$ on order N, after $\phi 7$ on order M, and after $\phi 8$ for D.

The beginning of $\phi 6$ occurs after the second word period of $\phi 5$, in which $P_1=1$. It is indicated by the term

$$F' = \mathbf{FG}HP_1t + . \tag{28}$$

To mark the end of $\phi 6$, use is made of a part of the content of the counter, marked by the timing signal y, which is not used by the control address. In each word period the timing signal v announces a sector number during the digit periods y as well as during u, as shown in Fig. 1. This second sector number announcement is copied into the counter during each word period (hence, in particular, the last) of $\phi 3$ and held there during the subsequent phases 4, 5, and 6. This is represented by

$$C'' = \mathbf{GH}vy + (G + H)yC + . \tag{29}$$

During $\phi 6$ toggle K is used to seek agreement between v and C just as it is in $\phi 1$, except that agreement is sought during time y rather than during u. For this purpose (6.2) is replaced by

$$\mathbf{K}' = (\mathbf{GH}u + Hy)(v\mathbf{r} + \mathbf{v}r) + , \tag{6}$$

and r must now be redefined as

$$r \equiv \mathbf{F}HR + (F + \mathbf{H})C. \tag{7}$$

Phases 4, 5, and 6 together occupy one full drum revolution. Thus the end of $\phi 6$ is marked by Kt which indicates that the sector number recorded in C during v of the last word period of $\phi 3$ has been recognized. If the order is M or D, which are distinguished from N by the presence of Q_4, $\phi 7$ is to be entered. Thus

$$G' = \mathbf{G}HKQ_4t + . \tag{30}$$

On order N the end of $\phi 6$ calls for return to $\phi 1$, produced by the terms

$$F' = F\mathbf{H}K\mathbf{Q}_4t + , \tag{31}$$

$$\mathbf{H}' = FHK\mathbf{Q}_4t + . \tag{32}$$

Phase 7 lasts for two word periods. Its end is recognized by the appearance of $FGHP_1t$, which is used to set toggle F to 0.

$$\mathbf{F}' = FGHP_1t + . \tag{33}$$

If the order is D this setting produces $\phi 8$. On order M, which is distinguished from D by the presence of Q_3, it is $\phi 1$ which is to be entered, hence toggles G and H must also be set to 0. Thus

$$G' = FGHP_1Q_3t + , \qquad (34)$$

$$H' = FGHP_1Q_3t + . \qquad (35)$$

After one word period $\phi 8$ ends and $\phi 1$ is begun, thus

$$G' = FGHt + , \qquad (36)$$

$$H' = FGHt + . \qquad (37)$$

Multiplication Procedure

In $\phi 4$ the operand number is picked up in the instruction register and is held there in the later phases. This number serves as the multiplicand. The previous accumulator content is kept recirculating in the accumulator and functions as the multiplier. In order to provide storage capacity for the successive partial products the accumulator is extended to slightly over twice its normal length by the use of a second reading head. The digits presented by this second head are denoted by A^*. A digit, A'', recorded in the accumulator is presented by A^* in the 65th following digit period, that is after a delay of one digit period more than two word periods. Thus information rerecorded from A^* appears every other word period but precessing by a one digit period delay per circulation. The enlarged storage capacity of the accumulator is shared by the multiplier and the growing partial product. The partial product is initially of one word length and progressively grows to about two word length. As each digit of the multiplier is used it is dropped from recirculation, hence the storage requirement of the multiplier concurrently drops from one word length to zero.

In each pair of word periods of $\phi 6$ the multiplicand, recirculating in the instruction register, is or is not added to the partial product held in the accumulator as a corresponding digit of the multiplier is 1 or 0. Most of the digits of the partial product are presented by A^* during the "odd" word periods, marked by P_1, some however have precessed into the succeeding "even" word period. For this reason the addition process is extended to two word periods. A precaution, described below, is taken to prevent falsification of the circulating digits of the multiplier. The multiplicand is presented by R only to one word period length. It is extended to two word period length by repetition of its sign (t) digit in all digit periods of the second (even) word period.

In the process of addition or subtraction in $\phi 4$ as described above an exception to the normal behavior of toggle L is made for the t digit period. In phases 5 to 8 for multiplication (distinguished from division by Q_3) that exception is restricted to the even word periods, thus extending the process to two word periods. Eqs. (20.2) and (21.2) are now replaced by the following:

$$L' = (is + is)j(t + HQ_3P_1), \qquad (20)$$

$$L' = (is + is)j + t(H + Q_3 + P_1). \qquad (21)$$

The addition of the (extended) multiplicand in each step of the multiplication during $\phi 6$ is controlled by a digit of the multiplier held in toggle P_6 during that pair of word periods. This digit was set into P_6 in the t digit period preceding these word periods, as described by

$$P_6' = HP_1tA^*Q_3 + ; \quad P_6' = HP_1tA^*Q_3 + . \qquad (38)$$

Similarly P_6 picks up the sign digit of the multiplier at the end of $\phi 4$ and holds it during $\phi 5$, as described by

$$P_6' = FGHtA + ; \quad P_6' = FGHtA + . \qquad (39)$$

Since the action in $\phi 5$ is controlled by the sign digit of the multiplier the multiplicand is (or is not) subtracted rather than added as in $\phi 6$. Since, moreover, this is the first step of the multiplication there is no previous partial product.

The multiplicand sign is needed throughout the even word periods. It is therefore picked up by toggle P_5 at the end of $\phi 4$ and held through phases 5 to 7.

$$P_5' = FGHtV + ; \quad P_5' = FGHtV + . \qquad (40)$$

In phases 5 to 7 the accumulator records the sum (or difference), b, except for the digits appearing at time t of even word periods. These are suppressed to prevent their precessing into the odd word periods. Thus

$$A'' = Hb(t + P_1) + . \qquad (41.1)$$

The inputs to the add-subtract unit are as follows: in $\phi 5$ a subtraction is performed, thereafter additions.

$$s = F \qquad \text{on } HQ_3.$$

In the routine part of the multiplication, performed in $\phi 6$, the inputs to the adder are A^* and R extended by repetition of its sign digit, P_5, during the even word period and contingent on the presence of a 1 as multiplier digit, P_6. This is expressed by,

$$i = A^*; \qquad = (P_1R + P_1P_5)P_6 \qquad \text{on } FGHQ_3.$$

In $\phi 5$, in which a subtraction is done, these inputs are slightly modified. The factor, P_6, in the second term of j is omitted. This has the effect of subtracting the repeated digit, P_5, in the even word period even if the multiplier is positive. That is equivalent to depositing the digit P_5 in the small gap separating the growing partial product from the multiplier digits. If the multiplicand is negative, $P_5 = 1$, the 1-digit so deposited serves to guard the multiplier digits from erosion in the later additive steps and does no harm to the growing product. Thus for $\phi 5$ the inputs are,

$$i = A^*P_1; \qquad j = P_1RP_6 + P_1P_5 \qquad \text{on } FGHQ_3.$$

On order N the completed less significant part of the product is recorded in the accumulator in the last word period of $\phi 6$, and the execution of the operation is then terminated. On order M a completed more significant half is recorded in the first (odd) word period of $\phi 7$.

However, to present this result in the normal form it must be delayed by one digit period. This is accomplished by adding A to itself in the even word period of $\phi 7$, after which the operation is terminated. The odd word period of $\phi 7$ is like those of $\phi 6$. Thus $\phi 7$ for multiplication is described by

$$i = A^*; \quad j = P_5P_6 \quad \text{on } FGHP_1Q_3,$$
$$i = A; \quad j = A \quad \text{on } FGHP_1Q_3.$$

This description of the multiplication process may now be summarized as follows:

$$s = FHQ_3 + , \tag{42}$$
$$i = HQ_3[A^*(FG + P_1) + AGP_1] + , \tag{43}$$
$$j = HQ_3[RP_6P_1G + P_1P_5(P_6 + F) + AGP_1] + . \tag{44}$$

Division Procedure

The procedure for division is similar to one which has been described by Burks, Goldstine, and von Neumann.[8] It is a nonrestoring system, in which each step brings the remainder toward zero by subtracting or adding the divisor as its sign agrees or disagrees with that of the remainder. It makes use of the expanded accumulator, like multiplication, to provide space for the storage of the growing set of quotient digits and to provide, by its precession, for the doubling of the previous remainder at each step. As in multiplication each step requires two word periods.

The divisor is picked up in $\phi 4$ and held thereafter in the instruction register, as described above. Its sign is held by P_5 as described by (40). The sign of the dividend is held through $\phi 5$ by P_6 as described by (39). Subsequently the sign of each remainder is set into P_6 and held for two word periods. A new remainder is formed and recorded in each odd word period. It is, however, convenient to pick up its sign in P_6 at the end of each even word period, at which time it is presented by A. Thus,

$$P_6' = HP_1tAQ_3 + ; \quad P_6' = HP_1tAQ_3 + . \tag{45}$$

In each odd word period of phases 5 to 7 ($\phi 8$ has only an even word period) the doubled prior remainder (or for $\phi 5$ the dividend) is corrected by the subtraction or addition of the divisor (R), as the two signs held in P_5 and P_6 are alike or differ.

In $\phi 5$ the first input, i, is to be the dividend which is presented by A except for its sign digit. The sign digit is held in P_6, hence the dividend can be reconstituted as $A + P_6t$. Thus,

$$i = A + P_6t; \quad j = R, \quad s = P_5P_6 + P_5P_6 \quad \text{on } HFP_1Q_3.$$

In the odd word periods of phases 6 and 7 the doubled remainder is presented by A^*,

[8] A. Burks, H. Goldstine, and J. von Neumann, "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," Institute for Advanced Study, Princeton, pt. 1, 2nd ed. vol. 1, pp. 23–29; September 2, 1947.

$$i = A^*; \quad j = R; \quad s = P_5P_6 + P_5P_6 \quad \text{on } HFP_1Q_3.$$

In even word periods of phases 5, 6, and 7 the extended accumulator recirculates without change.

$$i = A^*; \quad j = 0 \quad \text{on } H(F + G)P_1Q_3.$$

The even word period part of A^* is gradually filled by the sign digits of the remainders recorded in the odd word period. In the even (and only) word period of $\phi 8$ the digits presented by A^* consists entirely of these remainder sign digits, each of which, together with P_5, determined the direction of one of the corrections used in the progressive reduction of the remainder. The first correction was determined by the sign of the dividend, which by $\phi 8$ has precessed out of the accumulator. However, in a "proper" division the magnitude of the divisor exceeds that of the dividend, hence the sign of the first remainder must be opposite to that of the dividend. These two signs were used at an earlier stage to induce blockage on improper division by a process mentioned above (but not described in detail).

Each digit presented by A^* in $\phi 8$ is combined with P_5 to form a digit, q_p, defined by

$$q_p = A^*P_5 + A^*P_5$$

where q_{31} corresponds to A^* in the first digit period, q_{30} in the second, etc. to q_0 corresponding to A^* in the t digit period. Each $q_p = 1$ indicates a subtraction of the divisor, each $q_p = 0$ indicates an addition of the divisor in the progressive reduction of remainders, except that q_0 corresponds to two successive opposite corrections applied to the dividend and first remainder which, since $\phi 8$ has been reached without interrupting blockage, may be presumed to have been of opposite sign. Taking account of the doubling of the remainder at each step it can be seen that the dividend has been brought approximately to zero by the subtraction of the divisor multiplied by the following number:

$$q = -(2q_0 - 1) + \tfrac{1}{2}(2q_0 - 1) + \tfrac{1}{4}(2q_1 - 1) + \cdots + 2^{-32}(2q_{31} - 1)$$
$$= -q_0 + \tfrac{1}{2}q_1 + \tfrac{1}{4}q_2 + \cdots + 2^{-31}q_{31} - 2^{-32}.$$

Thus q_0, q_1, etc. are the sign digit and progressively less significant digits of an indefinitely continued true quotient in accordance with the system of number representation described above. To produce a rounded quotient of sign and 30 significant digits, the digit, q_{31}, is added to the least significant position of the number (q_p) in $\phi 8$. This digit is available in the form

$$q_{31} = P_5P_6 + P_5P_6$$

during $\phi 8$. Its addition to the least significant digit position is more conveniently accomplished by subtracting it from all digit positions. Thus the action in $\phi 8$ is represented by

$$i = A^*P_5 + A^*P_5; \quad j = P_5P_6 + P_5P_6; \quad s = 1 \quad \text{on } FGH.$$

The description of division may now be summarized as follows:

$$s = H\mathbf{Q}_3(P_5P_6 + P_5P_6 + FG) + , \quad (46)$$

$$i = H\mathbf{Q}_3[A^*(F + GP_1 + GP_5) + A^*P_5FG + FP_1(A + P_6t)] + , \quad (47)$$

$$j = H\mathbf{Q}_3RP_1 + FGH(P_5P_6 + P_5P_6) + . \quad (48)$$

The recording of a sign digit must not be suppressed in $\phi 8$, hence (52) is replaced by

$$A'' = Hb(t + P_1 + FG). \quad (41)$$

Record Orders

Information is recorded in the main memory during $\phi 4$ on the orders H, C, Y, and R. (Cf. Table I.) The time during which recording is performed is denoted f. On orders H and C it is all of $\phi 4$.

$$f \equiv FGQ_1Q_2\mathbf{Q}_3 + . \quad (49)$$

On orders Y and R recording is done only during the part of a word period, s, occupied by the address of an instruction. Thus

$$f \equiv FG\mathbf{Q}_1\mathbf{Q}_2Q_3s + . \quad (50)$$

The digits to be recorded will be denoted V''. On orders H, C, and Y they are the digits presented by the accumulator,

$$V'' = (Q_1 + \mathbf{Q}_4)A + . \quad (51)$$

On the order Return the address to be recorded is the second address following that of the memory location in which the R instruction was found. (The memory location immediately after that holding the R order is needed for a U order which takes control to the "subroutine" from which it is later returned as a result of the R order.) Since the counter content has already (in $\phi 2$) been advanced by one since finding the R instruction it must again be augmented by one to provide the digits V''. For this purpose toggle K is used in $\phi 4$ in the same way as in $\phi 2$. This is accomplished by omitting the factor F from (16.1),

$$K' = GHwC + . \quad (16)$$

The digits to be recorded on order R are then $(KC + \mathbf{KC})$,

$$V'' = \mathbf{Q}_1Q_4(KC + \mathbf{KC}) + . \quad (52)$$

Print Order

The execution of the Print instruction, marked by the signal e, occurs in $\phi 4$.

$$e \equiv FGQ_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_4. \quad (53)$$

What key of the Flexowriter is struck is determined by the state of the P toggles.

Input

The Input process takes place for the most part with the computer in its blocked state. The action of the Flexowriter on reading a tape symbol sets the Input code in the Q toggles and the digits to be inserted in the P toggles. The computer is then set into $\phi 3$, from which it proceeds to the execution of the Input "order" and then enters a blocked $\phi 1$ to await another tape symbol. A special tape symbol releases the computer from the blocked state to permit it to digest and dispose of the digits set into the accumulator under the control of an input routine.

The Complete Logical Equations

To complete the description of the logical design there remains only the assembly of the partial equations given above. The assembled equations are shown in Table III. Each equation is followed by a list of the

TABLE III

SUMMARY OF LOGICAL EQUATIONS

$F' = FGHt + FGHP_1t$	1, 28	$F' = FGHt + FHK\mathbf{Q}_4t + FGHP_1t$	2, 31, 33
$G' = GHKt(F + Q_2) + GHKQ_4t$	4, 30	$G' = GHt + FGHP_1Q_3t + FGHt$	3, 34, 36
$H' = FGHt\mathbf{Q}_1Q_2(Q_3 + Q_4)$	26	$H' = FHK\mathbf{Q}_4t + FGHP_1Q_3t + FGHt$	32, 35, 37
$K' = t$	5	$K' = (GHu + Hy)(vr + vr) + GHwC$	6, 16
$L' = (is + is)j(t + HQ_3P_1)$	20	$L' = (is + is)j + t(H + \mathbf{Q}_3 + P_1)$	21
$Q_1' = FGHxR$	12	$\mathbf{Q}_1' = FGHxR$	12
$Q_2' = FGHxQ_1$	13	$\mathbf{Q}_2' = FGHx\mathbf{Q}_1$	13
$Q_3' = FGHxQ_2$	13	$\mathbf{Q}_3' = FGHx\mathbf{Q}_2$	13
$Q_4' = FGHxQ_3$	13	$\mathbf{Q}_4' = FGHx\mathbf{Q}_3 + Q_1\mathbf{Q}_2Q_3(A + Rz_0)$	13, 19
$P_1' = pGr + pGA + HP_1t$	9, 27	$P_1' = pGr + pGA + GP_1t + HP_1t$	9, 27
$P_2' = pP_1$	10	$P_2' = pP_1$	10
$P_3' = pP_2$	10	$P_3' = pP_2$	10
$P_4' = pP_3$	10	$P_4' = pP_3$	10
$P_5' = pP_4 + FGHtV$	10, 40	$P_5' = pP_4 + FGHtV$	10, 40
$P_6' = pP_5 + HP_1tA^*Q_3 + FGHtA + HP_1tA\mathbf{Q}_3$			10, 38, 39, 45
$\mathbf{P}_6' = p\mathbf{P}_5 + HP_1tA^*Q_3 + FGHtA + HP_1tA\mathbf{Q}_3$			10, 38, 39, 45
$A'' = AH[F + G + Q_1\mathbf{Q}_3\mathbf{Q}_4 + \mathbf{Q}_2(Q_3 + \mathbf{Q}_4) + \mathbf{Q}_1Q_2(Q_3 + Q_4)t] + FGH[\mathbf{Q}_2\mathbf{Q}_3Q_4(\mathbf{Q}_1 + A)V + Q_1Q_2Q_3b + \mathbf{Q}_1Q_2\mathbf{Q}_3\mathbf{Q}_4P_4] + Hb(t + P_1 + FG)$			14, 15, 41
$C'' = FGHw(KC + KC) + FGHQ_1\mathbf{Q}_2Q_3\mathbf{Q}_4R + FGH(\mathbf{Q}_1 + Q_2 + \mathbf{Q}_3 + Q_4)C + GHvy + (G + H)yC$			17, 18, 29
$V'' = (Q_1 + \mathbf{Q}_4)A + \mathbf{Q}_1Q_4(KC + KC)$	51, 52	$R'' = GHV + (G + H)R$	11
$b = Lij + Lij + Lij + Lij$	22	$r = FHR + (F + H)C$	7
$f = FGQ_1Q_2\mathbf{Q}_3 + FG\mathbf{Q}_1\mathbf{Q}_2Q_3s$	49, 50	$e = GFQ_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_4$	53
$i = AH + HQ_3[A^*(FG + P_1) + AGP_1] + H\mathbf{Q}_3[A^*(F + GP_1 + GP_5) + A^*P_5FG + FP_1(A + P_6t)]$			23, 43, 47
$j = VH + HQ_3[RP_6P_1G + P_1P_5(P_6 + F) + AGP_1] + H\mathbf{Q}_3RP_1 + FGH(P_5P_6 + P_5P_6)$			24, 44, 48
$s = HQ_4 + FHQ_3 + H\mathbf{Q}_3(P_5P_6 + P_5P_6 + FG)$			25, 42, 46
$p = FGHz + FGHz(Q_1 + \mathbf{Q}_2 + Q_3 + Q_4) + FGH\mathbf{Q}_1Q_2\mathbf{Q}_3\mathbf{Q}_4$			8

partial equations drawn from the text above which compose it. A few equations have been simplified by elementary algebraic manipulation, but no attempt has been made to reduce the equations to a most compact form or to indicate the many constructional simplifications which can be found by algebraic manipulation of this description of the logical design.

The set of logical equations shown in Table III omits a number of features of the LGP-30 structure which can conveniently be described separately: the entire system for the induction of and relase from the blocked state has been omitted. So also has the setting of the P and Q toggles in the Input process. Various devices, not described here, permit the operator to check on the functioning of the computer or to control its action without reliance on instructions already stored in the memory. These devices are necessary, although auxiliary, since the above description provides no way of inserting the input routine into the memory. The formation of t, u, v, x, y, and z from the three timing tracks is not shown. On orders U and T, which make no use of an operand word, $\phi 3$ is limited to one word period by a means not shown. The recording of a 0 in the spacer bit is ensured in a way not shown.

ACKNOWLEDGMENT

The author is indebted to many friends for advice and helpful discussions in the development of this design. In particular, conversations with James Cass throughout the period of its development have been useful. A number of necessary corrections and improvements of the design have been made by Raymond Davis, William Reinholz, and James Cass of Librascope Inc. during the development of the LGP-30. The assistance of the Librascope staff in the preparation of this manuscript is gratefully acknowledged.

A Transistor-Driven Magnetic-Core Memory*

E. LEROY YOUNKER†

Summary—**A transistor-driven magnetic-core memory which has a capacity of 1024 18-bit words has been built and is being studied. Both the read and write operations employ the coincident-current technique. The memory-drive currents are developed by transistors and the desired memory location is selected by magnetic-core selection switches. Eighteen thousand, four hundred and thirty-two memory cores are used in the storage array, 48 switching cores are used in the selection switches, and 160 transistors are used in core-driving circuits and read-out amplifiers. A typical memory cycle, reading followed immediately by writing, requires 20 microseconds.**

INTRODUCTION

IN THE FEW years since the use of square-hysteresis-loop magnetic cores in memory devices was proposed[1] the magnetic-core memory has established itself as a very attractive memory device for digital computers. Among its virtues are excellent reliability, capability of high-speed operation, and possibility of large storage capacity in compact size. The circuits associated with most present-day core memories use vacuum tubes, which add substantially to the size and power consumption of the over-all memory system. The use of transistors with a magnetic-core memory makes possible the realization of an all-solid-state memory system in which the associated circuits are compatible with the core storage array in reliability, speed, compactness, and power consumption. This paper describes an 18,000-bit coincident-current magnetic core memory which is operated entirely by transistors. This memory has been built as part of a feasibility study of transistorized magnetic-core memories by the TRADIC[2] (Transistor Airborne Digital Computer) group at Bell Telephone Laboratories. The memory is described in enough detail to show where transistors are used and what requirements are imposed on them. Detailed transistor circuits are shown and experimental results are discussed.

DESCRIPTION OF MEMORY

Description of Digit Planes

In the construction of magnetic-core storage arrays, the individual cores are commonly mounted in square or rectangular mechanical assemblies. Since each core in such an assembly usually stores one binary digit of a number, the assembly is called a digit plane. The TRADIC memory is designed to store 18-bit numbers, so 18 digit planes are required. The memory can store 1024 numbers; consequently, each digit plane contains 1024 cores.

* Manuscript received by the PGEC, October 3, 1956.

† Bell Telephone Labs. Inc., Whippany, N. J.

[1] J. W. Forrester, "Digital information storage in three dimensions using magnetic cores," *J. Appl. Phys.*, vol. 22, pp. 44–48; January, 1951.

J. A. Rajchman, "Static magnetic matrix memory and switching circuits," *RCA Rev.*, vol. 8, pp. 183–201; June, 1952.

[2] Work supported by Contract AF33(600)-21536, U. S. Air Force, Air Materiel Command.

A digit plane of the TRADIC memory is shown in Fig. 1. The 1024 cores, each 0.080 inch OD, 0.050 inch ID, and 0.025 inch thick, are arranged in a rectangular pattern, 64 inches long by 16 inches wide. The photograph shows only one-half of the cores, because the assemblage of cores is folded over a center supporting plate. Notice the enlarged portion of the array which has been inserted into the upper part of Fig. 1.

Close examination would show that 6 wires pass through each core—2 of them run along a row and 4 run along a column. One pair of wires (one in a row and one in a column) is used to select a core in reading. A second pair (also one row and one column) is used to select the core in writing. One of the remaining wires in the column is the output winding and the other is the digit-inhibit winding. The purpose of the digit-inhibit winding is to control the binary value of the digit inserted into the plane during the writing operation.

The digit planes were built to Bell Laboratories specifications by the International Telemeter Corporation of Los Angeles. The eighteen digit planes were assembled at Bell Laboratories to form the three-dimensional storage array shown in Fig. 2. Strapping between digit planes connects together corresponding wires in adjacent planes that are used for selecting the desired memory location. For example, for selecting the proper column in reading, there are 16 wires which start at one end of the memory, pass through all 18 digit planes, and terminate at the other end of the memory. The output and digit-inhibit wires are individual to each digit plane—their terminations are indicated in the photograph.

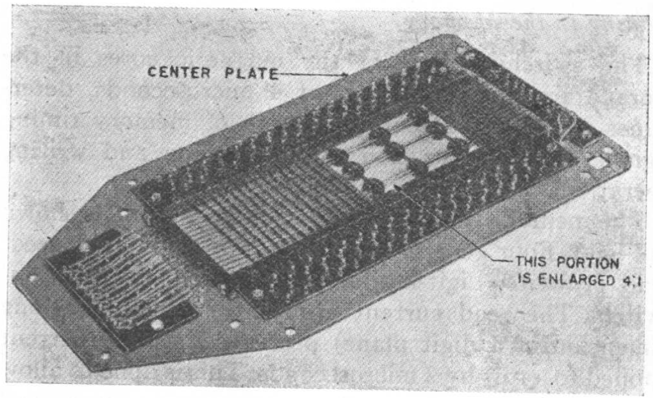

Fig. 1—Digit plane for TRADIC memory.

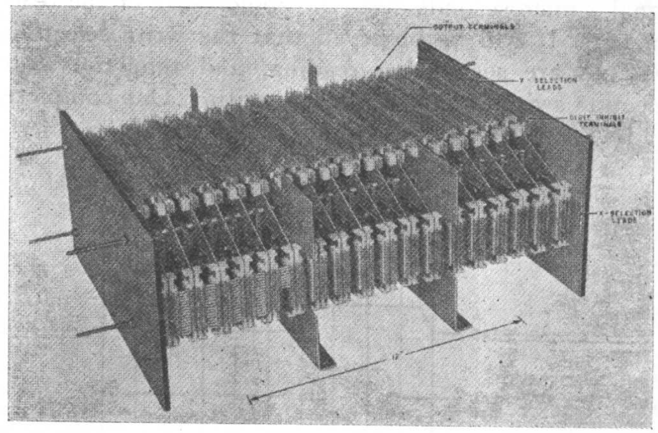

Fig. 2—Storage array for TRADIC memory.

Reading and Writing

Now let us consider the operations of reading and writing. To read a number from the memory, a current whose amplitude is one half the nominal switching current is applied to the proper row and a similar current is applied to the proper column. The polarity of the read currents is such that in each digit plane where the selected core is storing a binary ONE the core will switch and produce a voltage in the output winding. In the digit planes where the selected core is storing a ZERO, the core will not switch and essentially no voltage will appear in the output winding. Thus, the stored number appears in parallel form on the output windings of 18 digit planes.

To write a number into the memory, the desired location is selected by a coincidence of half currents on the proper row and column, as in reading. The polarity of the write currents is such that the selected core would be switched to the state of magnetization which represents a ONE.

Whether or not the core actually is switched is determined individually for each digit plane by the current in the digit-inhibit winding. The digit-inhibit winding is individual to a digit plane and passes through every core in that plane. In each digit plane where a ONE is to be written, no current flows in the digit-inhibit winding and the selected core is switched. In each plane where a ZERO is to be written, a half current whose polarity is opposite that of the write current flows in the digit-inhibit winding. One of the write currents is cancelled; the selected core does not get enough drive to switch and so stays in the state of magnetization which represents ZERO.

Selection of Memory Location

Memory locations can be selected in any order. During each read and write operation, the desired row and column in the storage array are selected by magnetic-core selection switches.[3] These switches operate in a two-phase manner. First, a switch is set to the desired position as determined by the number in the address register.

Second, the read (or write) current pulse is applied to the switch. This current flows through the switch into the selected row or column and at the same time resets the switch to a reference position. Separate switches are used to select the row and column in the reading and writing operations.

[3] M. Karnaugh, "Pulse-switching circuits using magnetic cores," Proc. IRE, vol. 43, pp. 570–584; May, 1955.

Timing in the Memory

The switching time of the magnetic cores in the storage array, which is about 4 microseconds, determines to a large extent the length of memory timing signals and the duration of the reading and writing operations.

The memory timing for one read-write cycle is shown in Fig. 3. Four general types of timing signals are used: read or write, digit-inhibit, strobe, and set-selection-switch. The read current applied to a column (long dimension of a digit plane) precedes the read current applied to a row by 2 microseconds. This is done to allow time for the disturbance caused by the column read current to die out before the row read current is applied.[4] The output of the memory is strobed by a 2-microsecond signal 2 microseconds after the coincidence of the read currents. It will be observed that the write selection switches are set during reading, and that the read selection switches are set during writing. One complete read-write cycle requires twenty microseconds.

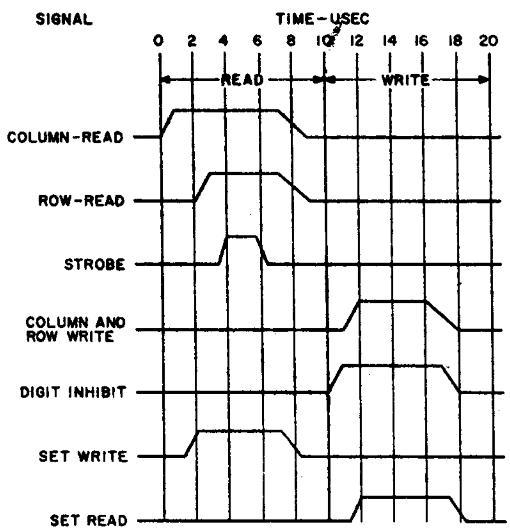

Fig. 3—Memory timing cycle.

Block Diagram of Memory

Fig. 4 shows a block diagram of the TRADIC memory. The storage array receives inputs from the selection switches and the digit-inhibit drive amplifiers, and supplies outputs to the read amplifiers. The inhibit drive and read amplifiers are provided on a per-plane basis. Only one each of the selection switches—column-read, column-write, row-read, and row-write—are provided for the entire storage array. Also only one transistor amplifier each is used to furnish the column-read current, the column-write current, the row-read current, and the row-write current for the entire array.

[4] R. Stewart-Williams, M. Rosenberg, and M. A. Alexander, "Recent advances in coincident current magnetic memory technique," unpublished paper presented at meeting of the Association for Computing Machinery at Ann Arbor, Mich.; June, 1954.

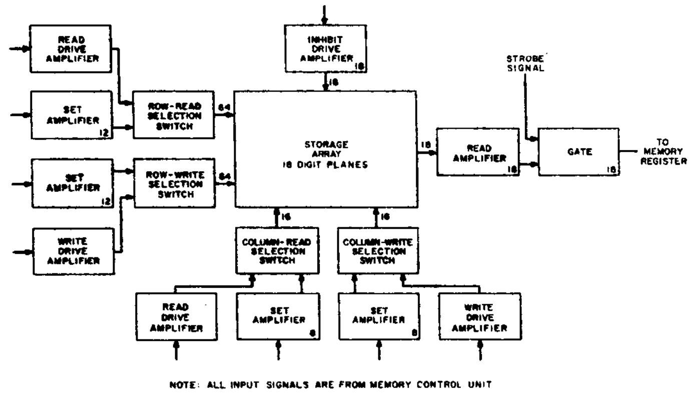

Fig. 4—Block diagram of memory.

The numbers on the lines which connect the storage array block and the adjacent blocks indicate the number of leads between units. The number inside a block indicates the required number of that unit. For example, 8 set-selection-switch amplifiers are required to set the column-write selection switch.

Detailed Transistor Circuits

In the magnetic-core memory, transistors are used to develop read and write currents, digit-inhibit currents, currents to set the magnetic-core selection switches, and to amplify the output of the digit planes. It is seen that the circuits most closely associated with a magnetic-core memory are of two types—first, circuits to provide the currents which switch magnetic cores, and, second, circuits to amplify the signal obtained from a switched memory core to a level which can drive circuits associated with the memory.

Drive Amplifiers

Amplifiers that supply current pulses to magnetic-core circuits in the selection switches or in the storage array are referred to as drive-current amplifiers. The drive amplifiers receive properly timed inputs of four milliamperes from the memory control circuits and are required to develop current pulses ranging from 70 to 200 milliamperes.

The digit-inhibit driver must supply a current pulse of about 160 milliamperes into the inhibit winding of a digit plane. The inhibit winding threads 1024 memory cores. While the inhibit-current is applied, no memory core switches, so except for small flux changes due to the nonsquareness of the memory-core hysteresis loop, the impedance of the winding is its dc resistance.

Fig. 5 shows the schematic of the digit-inhibit drive amplifier. The output transistor, Q_2, of the amplifier is essentially a switch. Normally, there is a high impedance from collector to ground and no current flows from the battery. When Q_2 is made to conduct, the collector voltage goes nearly to ground, and current flows through the digit-inhibit winding.

Now let us consider the amplifier in more detail. It uses two germanium-alloy transistors. Normally the