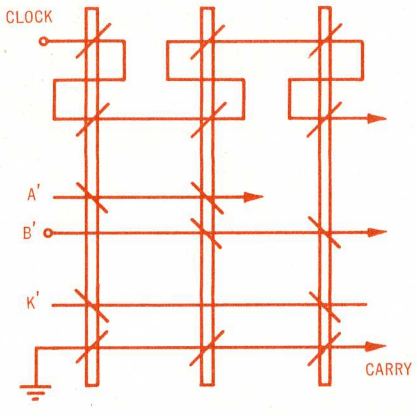
ELECTRONICS DIVISION

The National Cash Register Company

INHIBIT-CORE LOGIC



A Technique for Using Memory Cores as Logical Elements

L. J. ANDREWS

OVER THE YEARS there have been many outstanding papers which can be assembled under the collective title "Component Failure Analysis." Included in this group are the tube-life prediction tables, the derating charts for condensors, the maximum current versus useful life curves for diodes, etc.; and each company that has a customer service organization has in its files records loosely titled "Plug-in Failures, Their Cause and Cure." An examination of the records will show that in the majority of cases the "active" elements are at fault. Active elements are defined here as those elements which amplify a changed state of their inputs. It would seem, then, that the path to reliability is to remove as many active elements as possible from the system; that is, given some specific design problem, to time-share the active elements as much as possible in keeping with the flexibility required of the overall design. But this philosophy is not without its attendant apparent disadvantage. To use a minimum of active elements, a maximum of switching elements is required. To utilize the minimum active elements concept, a switching element approaching the ideal is required.

The ideal switching element should have the following characteristics: 1. there should be a minimum number of passive elements per switch, 2. the switch should not load the logical propositions, 3. the switch should be lossless, 4. the switch should have high discrimination, 5. the switch should be compatible with other system components, 6. the switch should be easily fabricated, and 7. it should have all the miscellaneous properties such as high speed, high output, and small physical size; and should be inexpensive, shock resistant, temperature insensitive, etc.

An available item that to a reasonable degree fulfills the miscellaneous properties is the small ferrite core usually used in memory applications. How these cores can approach the ideal switch and perform other useful functions as a result of a unique system concept is the subject of this paper.

The Inhibit-Wound Core

The inhibit-wound core is a deceptively simple component. In its rudimentary form it consists of a small ferrite core with one or more driving sources, an inhibiting proposition, and a sense winding. Each "winding" consists of a single wire through the core; the clocks and propositions each carry half-select current. Using the mirror symbols for core-winding senses[1,2] it is apparent that the core in Fig. 1 will change state from a *one* to a *zero* and back again as the C_R terminal alternates between positive and negative potential. A *one* is defined here as the up direction when current flowing into a slant bar is seemingly reflected up. A *zero* is thus also defined as current into a slant bar reflected down the core. The polarity of the sense signal need not be specified at this time.

In Fig. 1, then, the clock C_R has the ability to write a *one* when positive, and can read a *one* or *zero* when negative. The previous *one* or *zero* state is detected during read by the presence or absence of a large flux change and corresponding voltage induced in the sense winding. The propositions X_i have the restrictions that they may have current only during the write phases of the clock, their currents may only be of half-select or zero amplitude, and their currents will be only positive, i.e., away from the terminus of Fig. 1, and then only if the proposition is true. The relations between the propositions and the clock phases is best illustrated by referring to Fig. 2. During time T_0, proposition X_i is true, current is then flowing in the wire X_i in the direction previously designated as positive. The proposition X_i' (if it exists for use elsewhere) does not have current during the T_0 write time. Conversely, during T_1 and T_2 the wire designated X_i carries no current while wire X_i' has half-select current during write periods. It is a natural consequence, of course, that the core of Fig 1, wired with proposition X_i and subjected to the wave forms of Fig. 2, will have the states 0, 1, 1, and 0 at the end of timing periods T_0, T_1, T_2, and T_3 respectively. The core then can be said to have taken the complement of the information held in sequence by the proposition X_i within the framework of the system defined. Of more immediate interest is the case where not X_i but X_i' is the proposition present as an inhibitory signal. Here the core will assume information as repre-

L. J. ANDREWS is with the Electronics Division of the National Cash Register Company, Hawthorne, Calif.

The author expresses his appreciation to Walt Edwards and James Hudson, who supplied the initial impetus for this work.

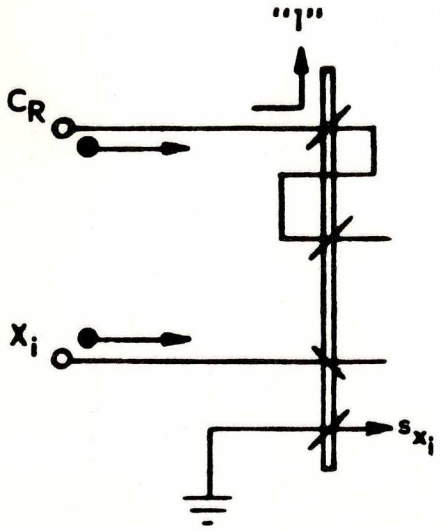


Fig. 1. Basic inhibit-core logical element

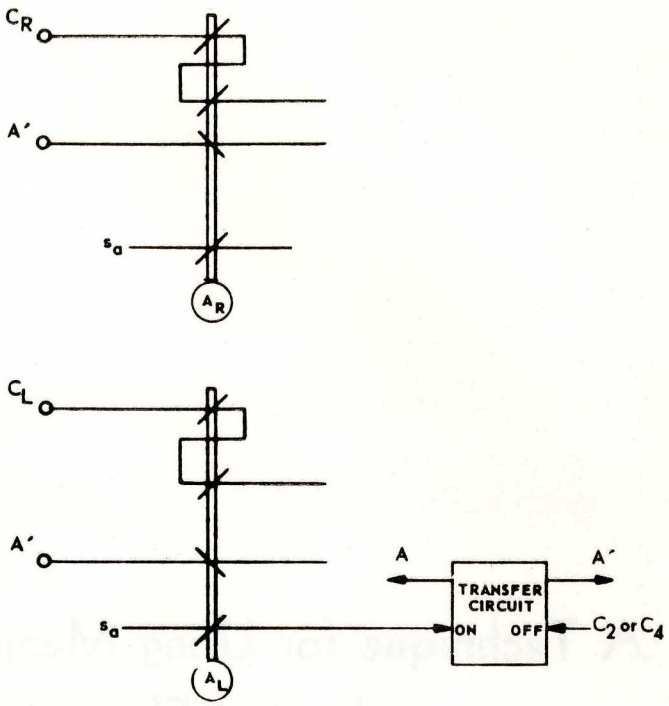


Fig. 3. Inhibit-core logic recirculation

sented by the true term, sequentially 1, 0, 0, 1. A magnetic core with a read-write clocking term, the prime of the proposition to be present in the core as an inhibiting winding, and provision for sensing signal comprise the fundamental unit for Inhibit-Core Logic.

Inhibit-Core Logic

Of prime importance in a computing system is the ability to read information out of storage. If the reading process is destructive, means must be provided to recirculate the just read information in case it is needed at some later time.

Consider the schematic diagram of Fig. 3 in which the two clocks C_R and C_L have the phase relationship illustrated in Fig 4. Note particularly that core A_R is similar to that of Fig. 1 and core A_L is also the same, with the exception that the cycle write-read is introduced in the interval between the read-write of core A_R. All similarly labeled terms are assumed to be in series. The box labeled "transfer circuit" is in essence merely a half-cycle delay amplifier pair, but for this example may be considered to be a d-c flip-flop with the following characteristics. The flip-flop will respond to the voltage on the sense wire corresponding to a changed core only during the read intervals (C_1 or C_3 of Fig. 4). This voltage may only turn the flip-flop on. The flip-flop will always go off at the end of a write interval (C_2 or C_4 of Fig. 4). The outputs A and A' are considered to be gated by the two write intervals. Therefore, a pulse of half-select current is present on either the A or A' during, and only during, each of the write intervals.

A *zero* in core A_R of Fig. 3 at the beginning of time T_1 is subjected to the wave form of Fig. 4. At read interval C_1 clock C_R is negative, current is then flowing in the direction to set the core to the *zero* state. Core A_L is dormant so there is ideally no signal present on the sense

Fig. 2 (below). Time relationships between propositions and clocking phases

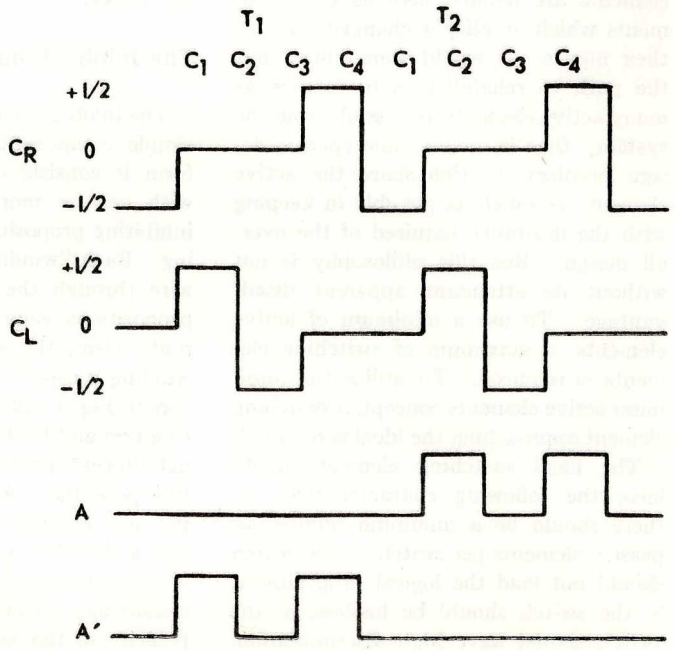
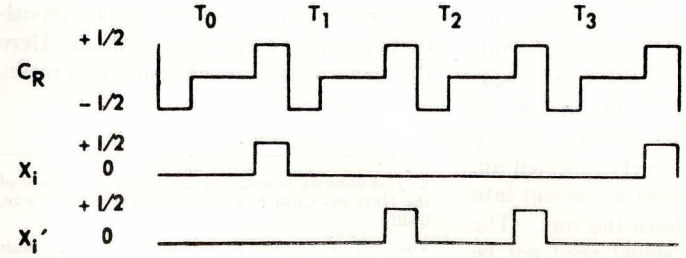


Fig. 4 (right). Phase relationships between register clock, logic clock, and wave forms obtained during recirculation



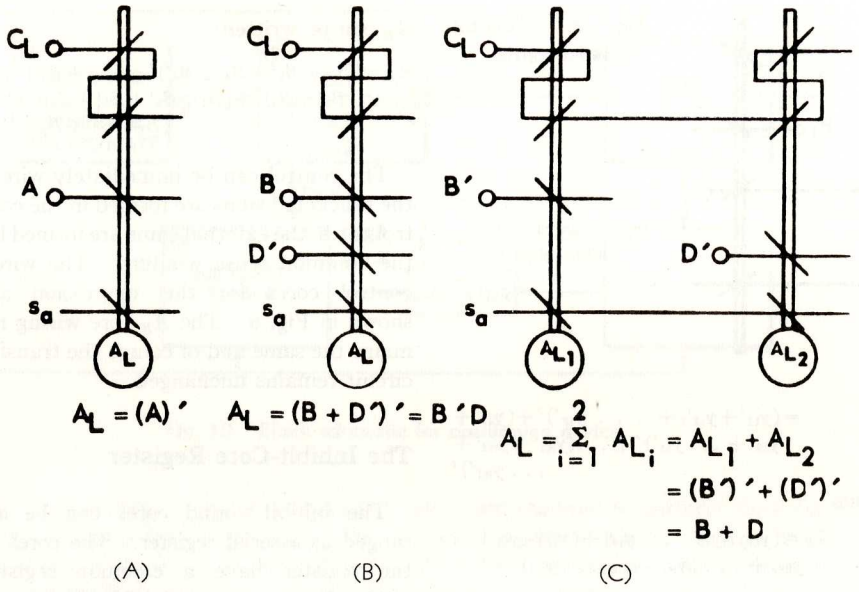


Fig. 5. Logical negation, product and sum

winding. The transfer circuit will then have an output during the write cycle C_2 on output line A'. Core A_L, if it is originally in the *zero* state, would ordinarily be set to the up or *one* state by clock C_L. However, current flowing in line A' inhibits the writing and A_L remains a *zero*. A' has no effect upon core A_R. Clock C_L now reverses sense, but the core, having remained in the *zero* state, has no flux change. A', then, again is true during write phase C_4 and inhibits the writing of a *one* in core A_R. At the end of time T_1 the two cores are in their original states. Conversely if there is a *one* in core A_R at the beginning of time T_2, C_R will now change the *one* to a *zero* during C_1. This change is sensed and the transfer circuit provides current on A during C_2 but no current on A'. Clock C_L is then able to write a *one* in A_L. The *one* is immediately read at C_3, the transfer circuit again has no output on A' at C_4 and the *one* is rewritten in core A_R. Again the original conditions are obtained. The significant point is that the inhibit-wound core A_L was used to control the inhibit-wound core A_R during recirculation of that core. All this is accomplished with a single time-shared transfer circuit. This basic philosophy which will be shown to be simple, inexpensive, and extremely versatile should be carried in mind throughout.

In the following, A_R will be the controlled core and will be wired as shown in Fig. 3. The wiring of the control cores will then establish a selected function in A_R. The previous example of recirculation shows that a signal on the sense winding during C_3 will allow C_R to write a *one* in A_R at C_4.

Negation can be accomplished by the circuit of Fig. 5(A). A *one* read from A_R will inhibit this logic (controlling) core during C_2. There is no output at C_3; A' is then true at C_4 and inhibits the writing of a *one* back in A_R leaving A_R a *zero*. A *zero* read during C_1 will not cause an inhibition in the logic core. The logic core therefore has an output at C_3 and C_R writes a *one*. *One's* and *zero's* are then interchanged in this operation.

Logical AND can be accomplished in a single core as shown in Fig. 5(B). The logical equations pertaining to these circuits will be discussed later but the physical argument is as follows: It is known that if a *one* is allowed to be written in the logic core a *one* will then be written in the controlled core. It is desired that the result of the logical product $B'D$ be in A_R; in Fig. 5(B) if proposition B is true it is not this case, so let B inhibit A_L. If D' is true it is not this case, so let D' also inhibit A_L. Thus the only time A_L is not inhibited is when the proposition $B'D$ is true. $B'D$ then appears in A_R.

Logical OR requires an inhibit-wound core for each proposition of the sum. The summing actually takes place on a common sense winding. Again reasoning physically in Fig. 5(C): the logical sum $B+D$ is wanted in the controlled core; core A_{L_1} will have a *one* if B is true (if not inhibited by B'), core A_{L_2} will have a *one* if D is true. If either or both of these cores has an output during C_3 the transfer circuit will respond. The sum $B+D$ will then appear in A_R.

These three basic functions are of interest, but the equations of a respectable computing system are almost never composed of single logical terms of a few elements. To point out the versatility of inhibit-core logic the basic circuits must be re-examined in the light of their logical equations.

In the circuit of Fig. 1 it may be verified, from a truth table, for instance, that the equation Core$=(X_i)'$ represents the state of the core at the end of its clock cycle. The proposition X_i may be any combination of AND's and OR's. If, as it is used, the proposition X_i is the inhibiting term the equation becomes Core$=(X_i')'=X_i$. Now if it is agreed for the present that the inhibiting term for the controlled core shall always be the prime output of the transfer circuit whose input

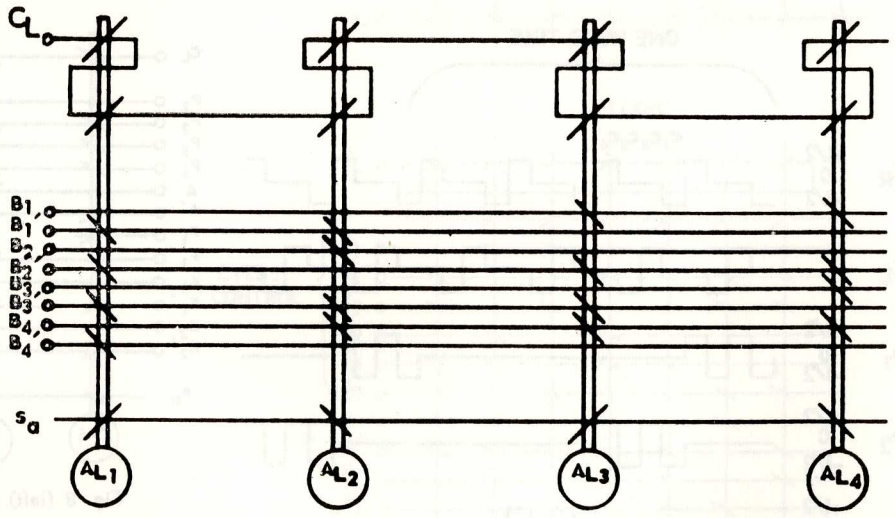


Fig. 6. A more general example of the mechanization of a Boolean function

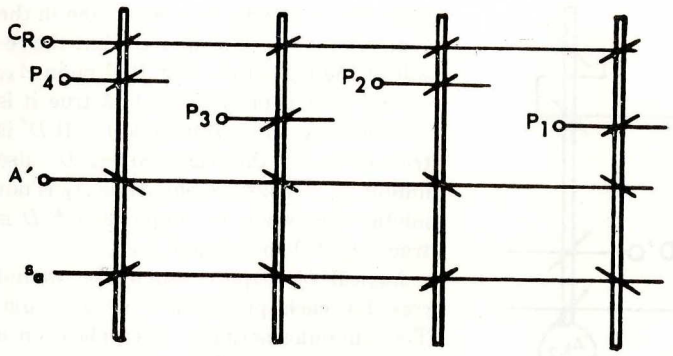


Fig. 7. Four-bit serial register

is the sense winding of the controlled core, the controlled core such as A_R of Fig. 3 will have the equation $A_R=(A')'=A$. It has been demonstrated that if the sense winding for the transfer circuit A_R has a voltage representative of a large change in flux in any of the A_L control cores during the C_3 read time, A_R will be true. The equation for A_R can then be written $A_R=A_L=A_{L_1}+A_{L_2}+A_{L_3}+\ \ldots\ +A_{L_n}$, where the plus sign indicates the logical sum. The A_{Li} terms are each inhibit-wound cores and so have the expression $A_{Li}=(Y')'$ where Y again is some arbitrary desired Boolean function. Since a magnetic core can be inhibited by one or a number of inhibiting terms, all having essentially equivalent effect, the controlling core provides a logical sum of its input windings. Thus the equation $Y'=y_1'+y_2'+y_3'+\ \ldots\ +y_j'+\ \ldots\ +y_n'$ can be written, where the y_j' are the single term propositions each threading a A_{Li} control core. The logical equation can be expressed as follows:

$$A_R=A_L=\sum_{i=1}^{n}A_{Li}=A_{L_1}+A_{L_2}+\ \ldots\ +A_{L_n}$$
$$=(Y_1')'+(Y_2')'\ \ldots\ (Y_n')'$$
$$=(y_{11}'+y_{12}'{}_r+\ \ldots\ y_{1r}')'+(y_{21}'+y_{22}'+\ldots y_{2s}')'+\ldots(y_{n1}'+y_{n2}'+\ldots y_{nt}')'$$

Thus

$$A_R=(y_{11}y_{12}y_{12}\ \ldots\ y_{1r})+(y_{21}y_{22}y_{23}\ \ldots\ y_{2s})+\ \ldots\ (y_{n1}y_{n2}y_{n3}\ \ldots\ y_{nt})$$

where the meaning of the subscripts is obvious.

The logical function to appear in the controlled core becomes the result of the individual inhibiting terms threaded through the controlling cores. It should be noted that this expression is the general case of a Boolean function expressed as a sum of products. There exists[3] a rigorous mathematical proof for the identity, (see last step of foregoing logical equation), but the derivation in this manner has additional merit. Working backwards gives a prescription for mechanizing any Boolean function. Perhaps an example here will be of value.

Suppose a function is to be mechanized in A_R:

$$A_R=B_1B_2B_3B_4+B_1B_2'B_3B_4'+B_1'B_2B_3B_4'+B_1'B_2B_3'B_4'$$

Going back one step in the derivation, A_R can be written:

$$A_R=(B_1'+B_2'+B_3'+B_4')'+(B_1'+B_2+B_3'+B_4)'+(B_1+B_2'+B_3'+B_4)'+(B_1+B_2'+B_3+B_4)'$$

The control can be immediately wired; the bracketed sums are formed in the control cores, the external sums are formed by the common sense winding. The wired control cores for this expression are shown in Fig. 6. The A_R core wiring remains the same and of course the transfer circuit remains unchanged.

The Inhibit-Core Register

The inhibit-wound cores can be arranged as a serial register. The cores of the register have a common register clock, C_R; a common inhibiting term, A'; and a common sense-winding leading to a single transfer circuit. If now these cores are to perform as part of a serial register they must be scanned sequentially by some noninterfering wave form. Further, in a practical register provision must be made for treating the digits in the register both individually or collectively. For instance, if a register is merely read, the digits are each treated alike, but if an arithmetic process is being carred out on the information the logic for the sign digit must be different than for the rest. The schematic of a 4-bit serial register is illustrated in Fig. 7, and wave forms which allow performance as required are shown in Fig. 8. It requires a coincidence between one of the clocks and a P timing signal each of half-select amplitude to have sufficient drive to switch a core. These coincidences have special significance; the $P-C_R$ coincidence during C_1 is

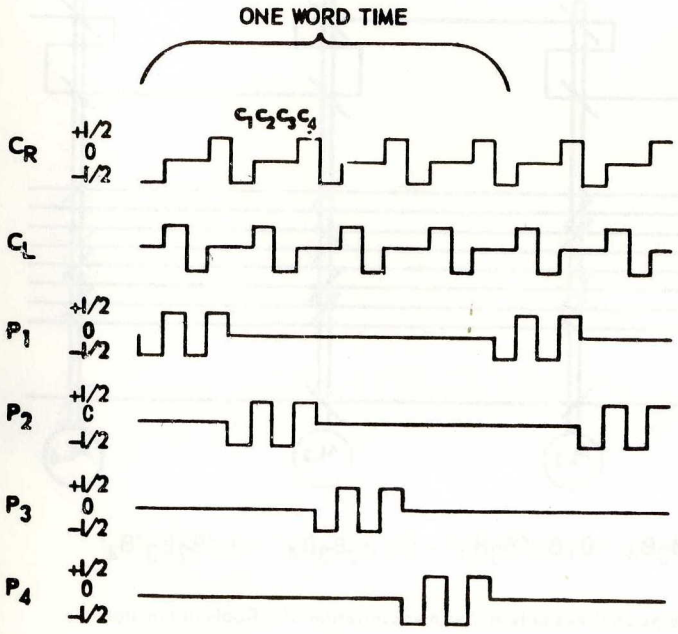


Fig. 8 (left). Wave forms for four-bit serial register

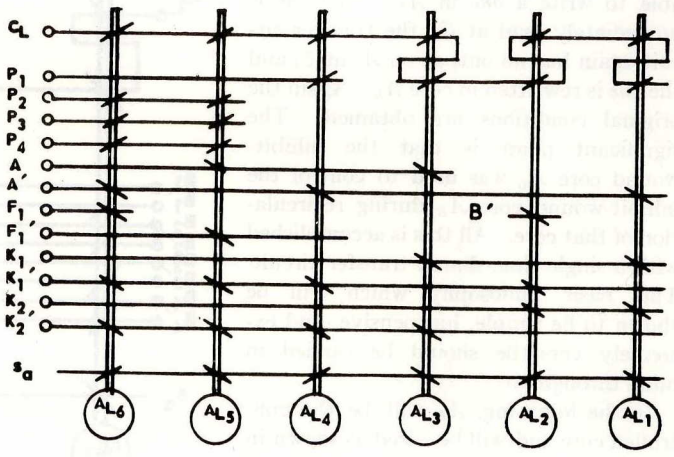


Fig. 9 (above). Control logic for four-bit register

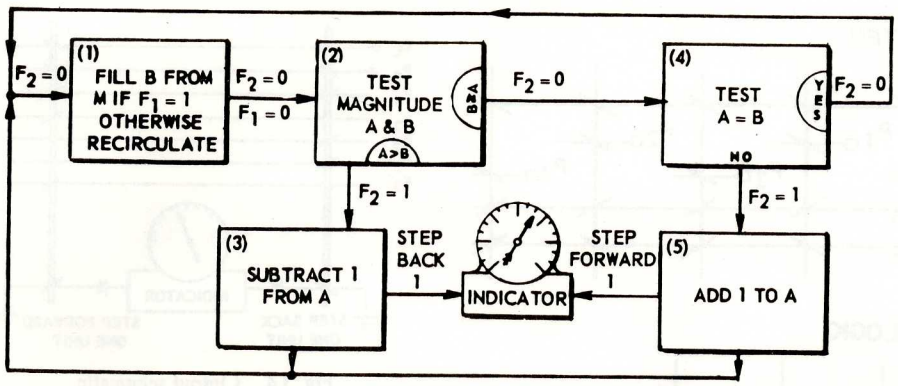


Fig. 10. Block schematic for positioning device

called "read register," the $P-C_L$ coincidence at C_2 is called "write logic," the $P-C_L$ coincidence at C_3 is called "read logic," and the last coincidence of $P-C_R$ is labeled "write register." At each P-time, then, the operation sequence is as follows: 1. read the information contained in the core of the register corresponding to the active P term and put the result in the transfer circuit; 2. using the information contained in this and any other transfer circuits, store the result of the prewired logical function to be performed on the register digit at this time in a logic core; 3. read the results of this logical manipulation into the original transfer circuit; 4. inhibit or not the writing of a *one* into the selected digit of the register. Fig. 9 shows the total number of cores and the wiring schematic to perform the logic required for recirculation, transferring, complementing, and counting in the 4-bit register. These operations should be taken as representative rather than exhaustive. The two propositions K_1 and K_2 define which of the four operations is to be done and correspond to program control. When $K_2'K_1'$ is true the programming inhibits are released from core A_{L_1} and the A' inhibit term then controls recirculation of the register at all P-times just as in Fig. 3. If $K_2'K_1$ is true it should be apparent that if B' is the output of another similar register, its formation will be transferred digit by digit to the A register by means of core A_{L_2}. Core A_{L_3} provides complementation of the information in register A including the sign position (assumed to be in the LSD digit, P_1). Of special interest are the three cores A_{L_4}, A_{L_5}, and A_{L_6}. If proposition F_1 is considered to be the output of some carry flip-flop, after the sign digit is recirculated unchanged by core A_{L_4}, cores A_{L_5} and A_{L_6} provide the addition of one (under command of K_2K_1) to the number held in the three other digits. The register will now count. It is to be noted that no additional logical cores are required to perform these same operations on a register of any length; these four operations, and in fact any number of logical manipulations, are possible on a register of any length with the one transfer circuit required of this simple case.

Flip-Flops

Flip-flops, such as the F_1 term referred to in the previous section, are a special case of the inhibit-core register concept. The register consists of only one core. Since the output of a flip-flop usually must be available at each P-time the single core is supplied with the continuous clock C_R and therefore is wired like the A_R core in Fig. 3. The logic (controlling propositions or grid equations) are then mechanized in a fashion similar to that for the actual registers. It is significant that no matter how complex the switching may have to be in order to time-share a particular flip-flop, only a single transfer circuit is required.

An Illustrative System

An example to demonstrate the technique is a position device. It could be used to set a pointer, shaft position, or graph plotter corresponding to some binary number. Only the pertinent detail necessary to illustrate the technique will be given.

A block schematic for the positioning device is shown in Fig. 10. The loop is considered to be continuously running initially with the pointer at zero, and the registers cleared. New information is taken from a memory register M into a storage register B. The third register A keeps account of the current position of the pointer. When the number in A reaches the number held in B the pointer will have been supplied with a sufficient number of pulses to indicate the number in B. New information can be inserted any time the cycle passes through block 1 becoming the new or present desired setting. Fig. 10 is the block schematic for the example. Figs. 11, 12, 13, and 14 show the core wiring for the complete switching operations for the A register,

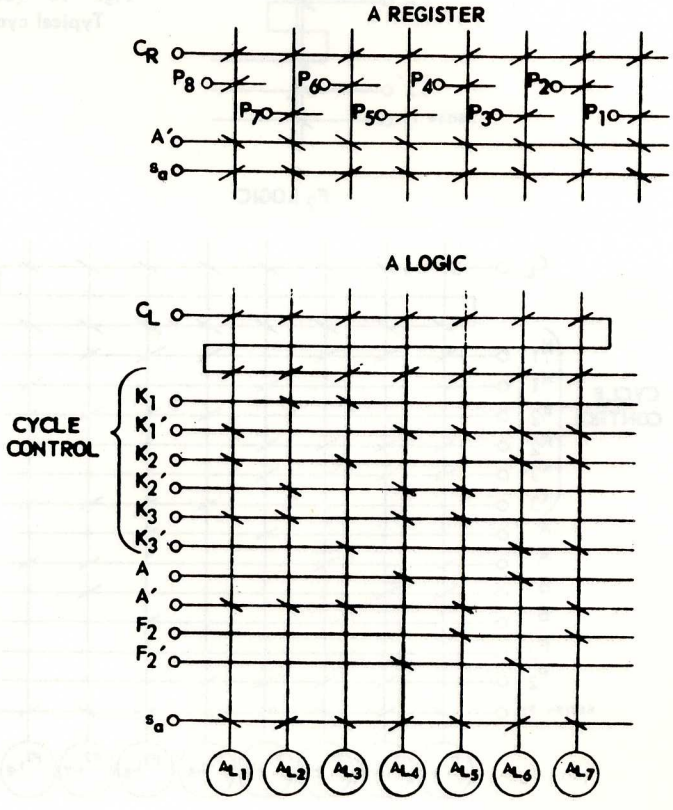
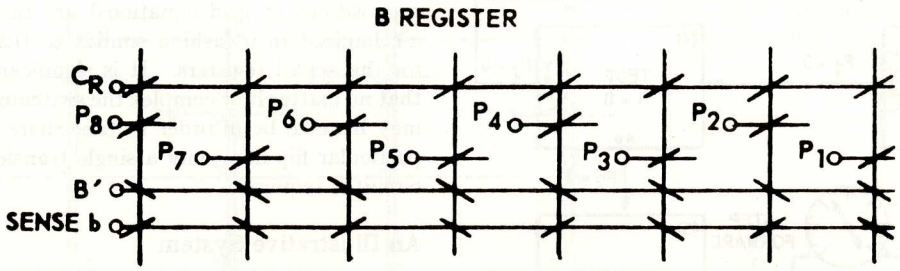


Fig. 11. The A register schematic



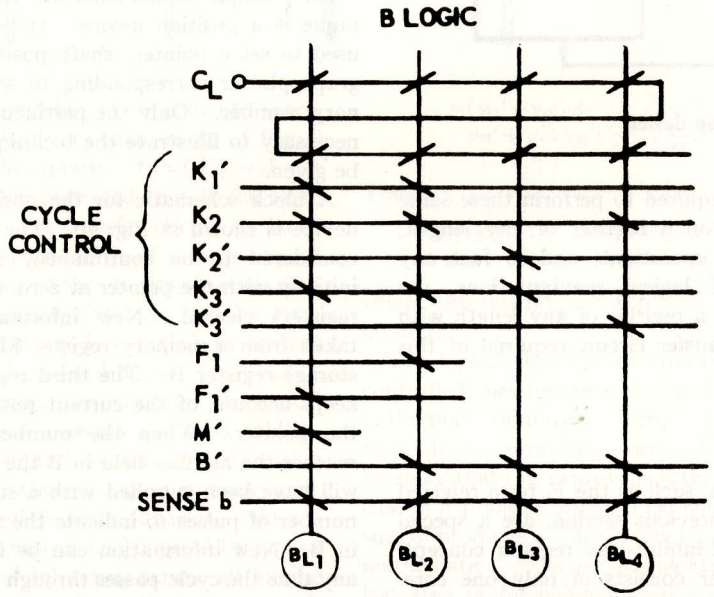


Fig. 12 (above). The B register schematic

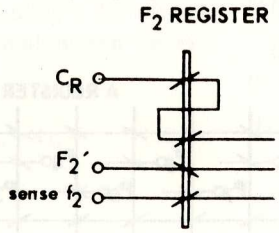
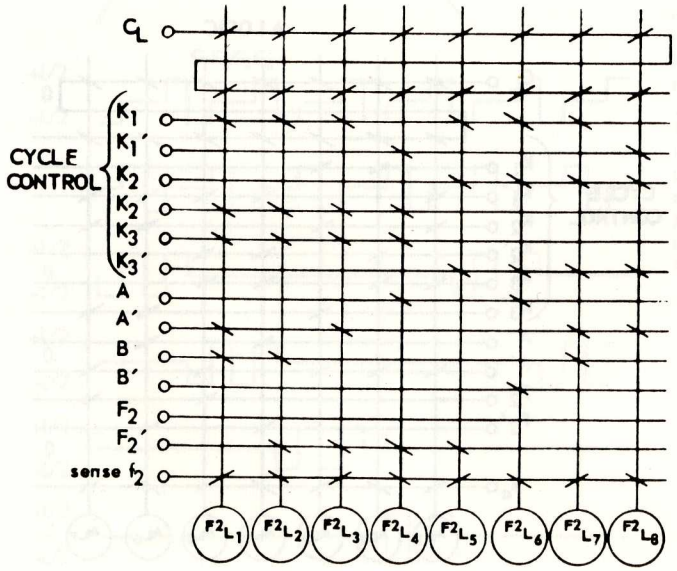


Fig. 13 (below left). The F_2 flip-flop



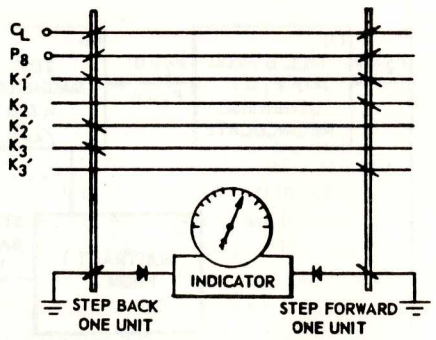


Fig. 14. Output schematic

B register, F_2 flip-flop, and output circuit respectively. The operation cycle is controlled by the three flip-flops K_1, K_2, K_3, and will be discussed later.

In the A logic, cores A_{L1}, A_{L2}, and A_{L3} provide recirculation in blocks 1, 2, and 4 respectively; cores A_{L4} and A_{L5} contain the logic for the subtraction of one in block 3; cores A_{L6} and A_{L7} contain the logic for the addition of one in block 5.

In the B register, core B_{L1} inserts information from the M register if flip-flop

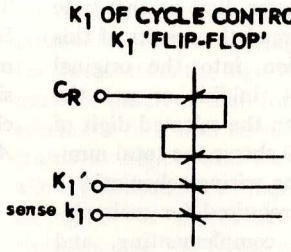
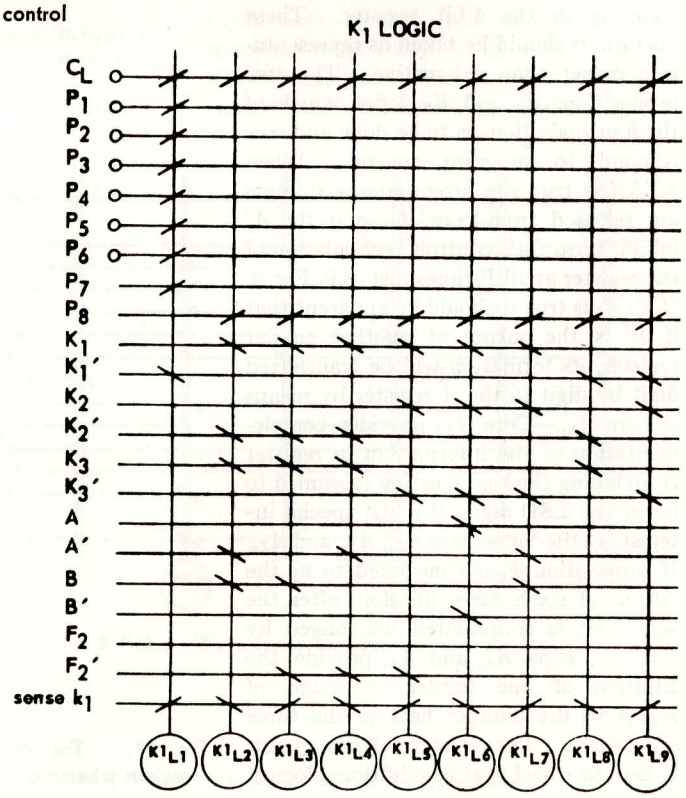


Fig. 15 (below right). Typical cycle control



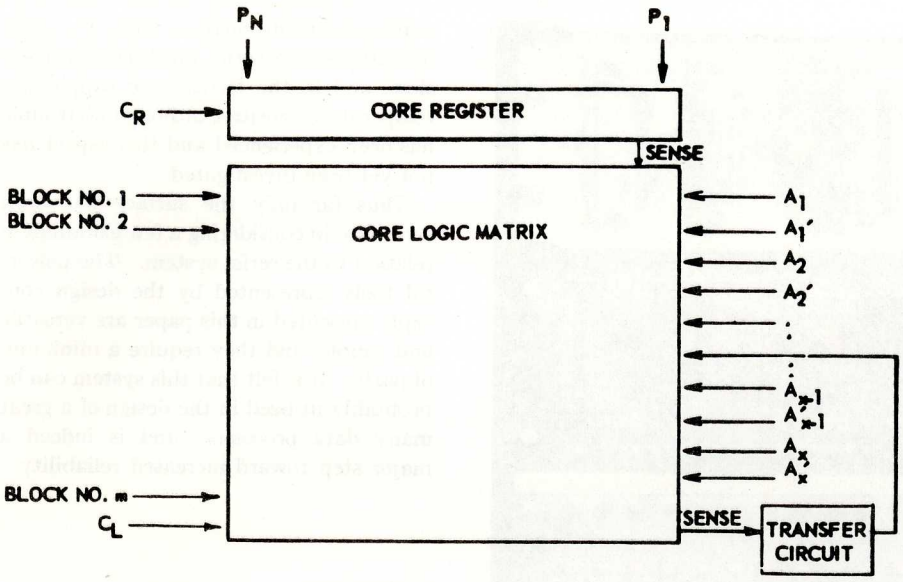


Fig. 16. Format of register

F_1 is true, core B_{L_2} recirculates the old information if F_1 is false, and cores B_{L_3} and B_{L_4} recirculate the B register during blocks 2, 3, 4 and 5. It is perfectly permissible to simplify logically the expression for the control before specifying the wiring of the cores.

The time-sharing aspect of the over-all concept is not very well brought out in examining the registers. More insight is obtained from the flip-flop logic. In Fig. 13 cores $F2_{L_1}$, $F2_{L_2}$, and $F2_{L_3}$ provide the logic required for F_2 to contain the results of the test magnitude specified in block 2; core $F2_{L_4}$ makes F_2 provide the borrow operation in block 3; cores $F2_{L_5}$, $F2_{L_6}$ and $F2_{L_7}$ provide the logic required for F_2 to contain the result of the Test Equality in block 4; finally, core $F2_{L_8}$ provides the carry for the addition operation of block 5. There is no control core corresponding to block 1. This implies that F_2 will be automatically set to zero as it should be.

The output circuit of Fig. 14 is self-explanatory. If the loop containing block 3 is traversed a positive pulse is delivered to the indicator to step back; if the loop contains block 5 a pulse to step forward is entered.

Of special interest is the cycle control since it directs the sequence of events based upon decisions obtained during the course of operation. An example is the LSD digit of the block counter $K1$ the logic for which is illustrated in Fig. 15. The cycle control must supply an inhibit-or-not pulse at C_2 for each of the eight bits of the P-times. These are automatically provided by arranging the cycle counter flip-flops to recirculate the information supplied them at the previous word time during the next word time. In Fig. 15 this function is provided by core $K1_{L_1}$. The absence of a controlling core during P_8 of block 1 will set zero to be recirculated in block 2. In block 2 a loop split is made based in a test magnitude operation. The result of the test will appear in flip-flop F_2, and in particular the result will be written in F_2 at P_8 time. If this decision is presented and written in K_1 at the same time, K_1 will provide the correct path during the next word time. Cores $K1_{L_2}$, $K1_{L_3}$, and $K1_{L_4}$, are wired to make this decision P_8 time. In the same manner the decision in block 4 is made by cores $K1_{L_5}$, $K1_{L_6}$, and $K1_{L_7}$. Cores $K1_{L_8}$ and $K1_{L_9}$ could have been wired as a recirculation since K_1 will not change going from either block 3 or 5 back to block 1, instead they are wired to set *one* unconditionally (a lack of inhibiting terms on the logic core automatically allows setting of the register core to *one*). The wiring for the other two flip-flops, K_2 and K_3, follows a similar pattern.

The storage, logical operations, output gating, and operational cycling have been done with a total of 6 required transfer circuits and about 70 small ferrite memory cores in addition to the clock and P sources. Of course in a larger system the ratio of cores to active elements is a great deal higher.

A Small Complete Machine

Looking back through Figs. 11, 12, 13, and 15 it can be seen that the wiring has a particular format, namely, that of Fig. 16. For each register there is associated a group of logic cores and a transfer circuit. These three items form a structure complete in itself which is operationally as reliable as the one transistor that may be required in the transfer circuit and as versatile as the designer may demand.

To establish the feasibility of this concept a small computer was constructed. It has a fixed program; an 8-bit word; and an 8-word memory, four words of which are available for storing selectable three-address commands. Information is inserted manually into the memory. The operations the machine will perform are addition, subtraction, test magnitude and multiplication. There are 22 block numbers. In operation the operator selects the control number and the machine will carry out the command named by that number. The registers, logic, and memory entail 193 size F-394 S-3 ferrite cores and eight transfer circuits complete. The design technique was essentially that described in this paper with the exception of the cycle control scheme; an 8-beat controllable shift register is used. The machine, since it uses S-3 ferrite cores and is serial in operation, has a bit operation time of 20 microseconds. Fig. 17 shows the machine; of course no attempt has been made to reduce the size.

Because of the small sample of the parts concerned, statistical data on operation would be of little value. However, the machine has been running continuously for some time and has yet to suffer an operational malfunction.

Conclusion

A computer design philosophy based on an inhibit-wound magnetic-core concept has been discussed. The major advantages are the following: 1. a minimum number of active elements are used for computation and control, 2. since all logical terms are inhibits they are never required to furnish power to their load, 3. all switching is done magnetically but never with a core driving core unit, 4. these elements are completely compatible with themselves and other system units, 5. since the propositions are entered in the logic core as single wires the number of terms of a product that can be formed in a core is limited only by the number of wires that the hole will accommodate, and 6. the system of logic inherently allows low power consumption. In fact, these advantages are very similar to the desired characteristics of the ideal switch discussed in the introduction.

The major disadvantages are the following: 1. the rather intricate wiring that is required is not conducive to automatic wiring techniques, 2. the 4-cycle clock and the inherent switching time of a ferrite core when changed with two half-

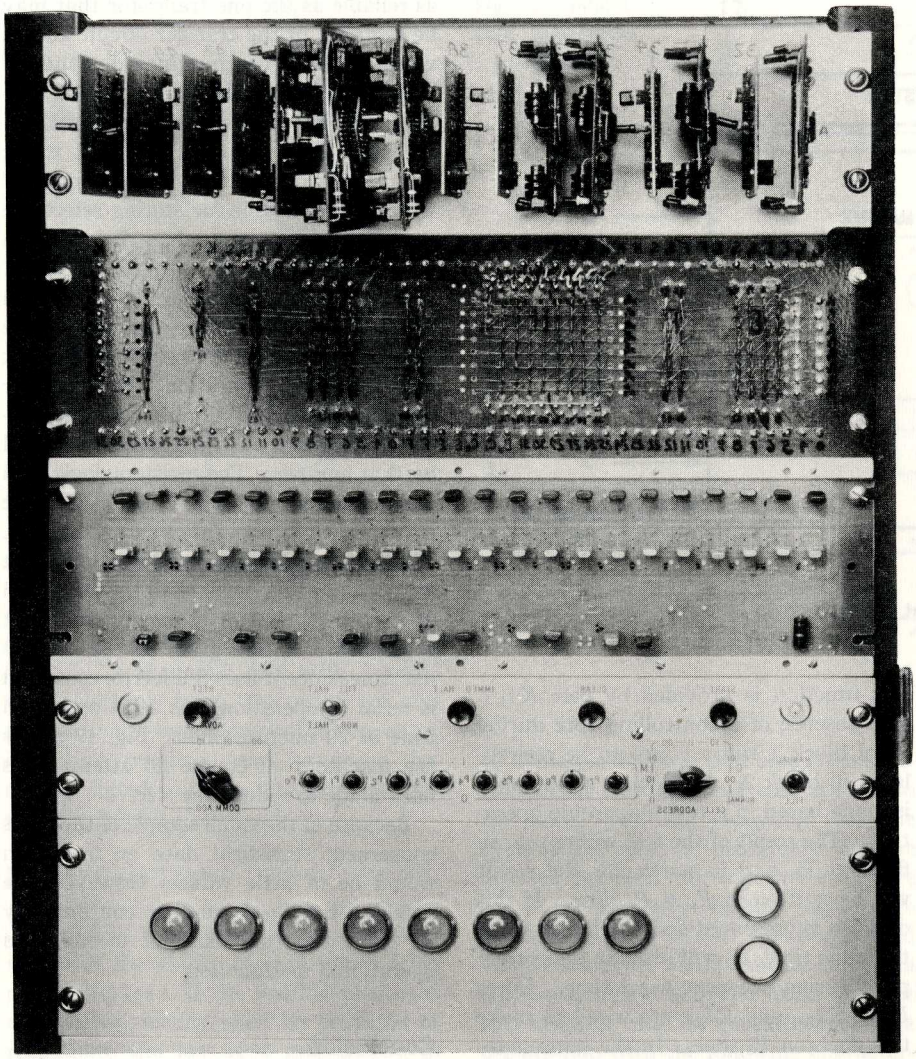


Fig. 17. Small core computer

select coincident currents make the basic operations in a serial machine relatively slow, and 3. the ferrites are supposedly temperature-sensitive although no trouble has been experienced and this aspect has not yet been investigated.

Thus far only the surface has been scratched in considering a few examples in relation to the serial system. The powerful tools represented by the design concepts presented in this paper are versatile and simple, and they require a minimum of parts. It is felt that this system can be profitably utilized in the design of a great many data processors and is indeed a major step toward increased reliability.

References

1. Pulse-Switching Circuits Using Magnetic Cores, **M. Karnough.** *Proceedings*, Institute of Radio Engineers, New York, N. Y., vol. 43, no. V, Aug. 1952, pp. 570–84.

2. A Proposed Symbol for Magnetic Circuits, **R. P. Mayer.** *Engineering Note E-472*, Digital Computer Laboratory, Massachusetts Institute of Technology, Cambridge, Mass.,

3. Inverted-Core Logic and the Synthesis of Boolean Functions Therefrom, **David Ellis, Lad Andrews.** *Research Report*, NCR-ED Bulletin no. 111, National Cash Register Company, Hawthorne, Calif., Oct. 1956.

CONCEPTUAL ASPECTS OF INHIBIT CORE LOGIC

By K. O. King

Introduction

In the past decade the utilization of the magnetic core as the fundamental logical element has appeared increasingly attractive to the computer circuit designer. This attractiveness was largely due to the inherent higher order of stability and ruggedness found in a magnetic core. While transistors and diodes are significantly more reliable than vacuum tubes they still suffer from temperature variations and momentary overloads; and their characteristics change with age. Hence,their life is limited. The magnetic core element, consisting of several windings on a ferromagnetic toroid, has a stable indefinite life.

Recognizing these reliability features, several organizations have already developed computers using magnetic core logic mechanization. However, the general technique applied was to utilize the cores in conjunction with diodes. Basically, the core was used as a magnetic amplifier while the diodes performed most of the logic function. While this technique was successful, the logic structure reliability was not significantly more than one utilizing diodes exclusively. Even more important, no improvement in the maintainability of the system was effected.

In the inhibit core logic mechanization proposed, magnetic cores alone serve as the decision-making or logical elements. Conventional transistor flip-flops provide storage. Because all logic is accomplished by indefinite life elements, the reliability of the computer logic area is orders of magnitude greater than a comparable diode or transistor mechanization. Even more important, properties intrinsic to inhibit core logic mechanization make it possible to easily test both the logic, and the remaining circuitry, within the system. Thus, an entirely new and advanced concept of computer maintenance is effected.

The development of this computer maintenance concept is the principal purpose of this section.

Core Characteristics and Definitions

The magnetic core material used for this application is one-eighth mil 4-79 Molypermalloy. This material is characterized as having a nearly rectangular hysteresis loop and its saturation flux is sharply defined. Its characteristics are maintained over an extremely wide temperature range ($\pm$ 200°C).

Consider the basic core element of Figure C-1 and its associated hysteresis loop of Figure C-2. Assume that only the bias current, I_B, is flowing. The core material is then in a "zero" state at

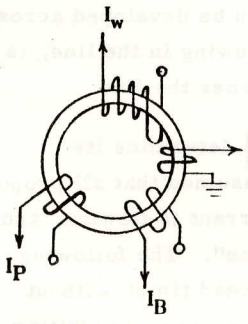


Figure C-1

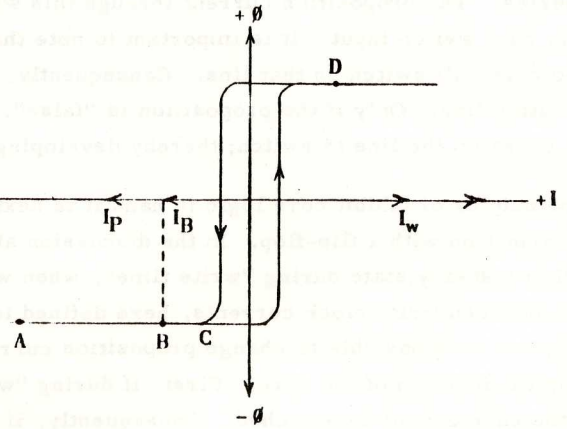


Figure C-2

point B. If now a "write clock" current pulse, I_W, of opposite polarity effect and of twice the coercive effect of the bias is applied, the values of magnetization of the core material will lie on a path on the hysteresis loop from B up to D, at which value the core will be in the "one" state; and, upon the fall of I_W, the values will lie on a path extending from D back to B. The change in flux, $\Delta\phi$, produced during this switching operation will develop a voltage across the sense output winding as shown in Figure C-3. Assume, next, that both I_B and the proposition current, I_p, are flowing and the "write-clock", I_W, is applied. The core material is initially at point A and lies on a path from A to C, then back to A. During this excursion the core does not "switch" but merely "shuttles". The resulting change in flux and sense output voltage is very small as indicated in Figure C-3.

It is important to note that a "switched core" output occurs only upon the application of a "write clock" in the absence of "inhibiting proposition" currents. One or more "inhibiting propositions" results in the core being "shuttled" but not "switched".

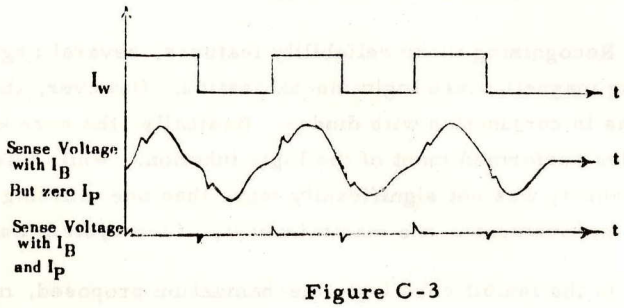


Figure C-3

Mechanization of Inhibit Core Logic

The basic core element of Figure C-1 can be more simply designated as shown in Figure C-4. In this model the propositions P_1, P_2.....P_N will be considered "true" when currents are flowing in their respective windings. If s, the sense winding output, is defined to be "true" for a switched core output the core element serves the following logical function:

$$s = C_W\left[P_1 + P_2 + \ldots\ldots P_N\right]' = C_W P_1' P_2' \ldots P_N'$$

Thus, the inhibit wound core serves as a "product" or "and" of N propositions. To create the "sum" or "or" of N "products" it is only necessary that the sense winding of the N product cores be connected in series. With this technique any logic whatsoever can be mechanized. To illustrate the mechanization in more detail, the following flip-flop input is shown in Figure C-5:

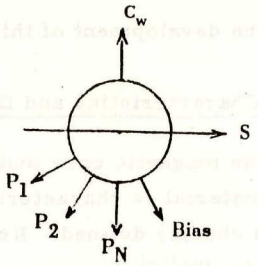


Figure C-4

$$f = C_W VXY' + C_W ZY' + C_w W$$

As indicated in Figure C-5 the same proposition windings, wound on many cores, are all connected in series. The proposition current through this winding may originate from a flip-flop, an input driver, or a switch input. It is important to note that if current is flowing in a proposition line no core will switch on that line. Consequently, no voltage can be developed across a "true" proposition line. Only if the proposition is "false", no current flowing in the line, is it possible for cores on the line to switch; thereby developing a voltage across the line.

With the principles of inhibit core logic in hand it is next necessary to determine its operation in conjunction with a flip-flop. In the discussion above it was assumed that all propositions remained in a steady state during "write time", when write clock current is flowing, and during the time between write clock currents, here defined to be "read time". The following considerations show it is possible to change proposition currents during "read time" without affecting the logical function of the core. First, if during "write time" there were "inhibiting propositions" the core cannot be switched. Consequently, if during "read time" the number of

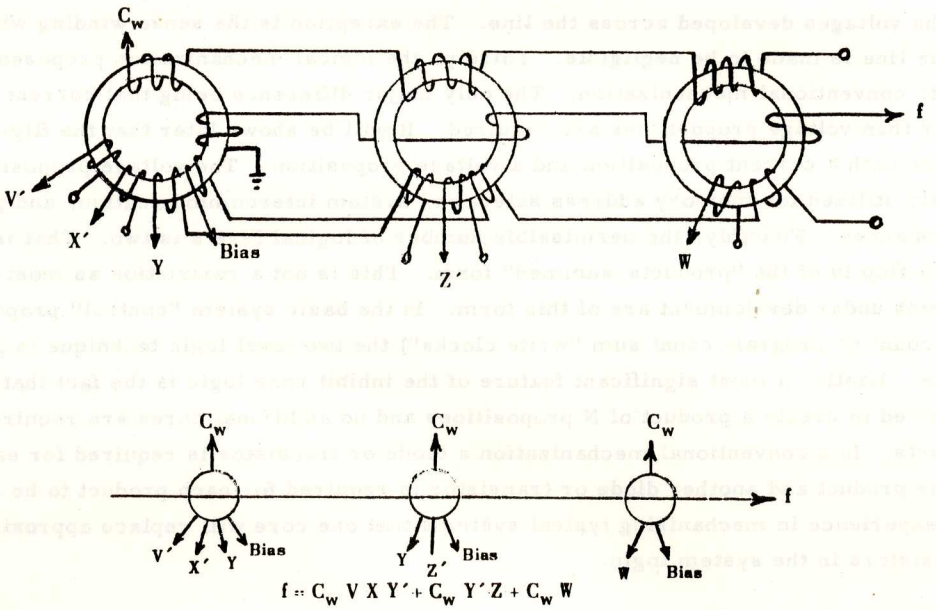


Figure C-5

"inhibiting propositions" change the core still can only "shuttle". Second, if there were no "inhibiting propositions" during "write time" the core will be "switched". Then, if during the following "read time" some "inhibiting propositions" become true, the only result is that the core "switches" faster during that "read time". The volt-second area of the sense output waveform remains the same.

In the system proposed the propositions change only when the flip-flops are triggered during "read time". The "switched core" output resulting from propositions going true during read time is shown in Figure C-6.

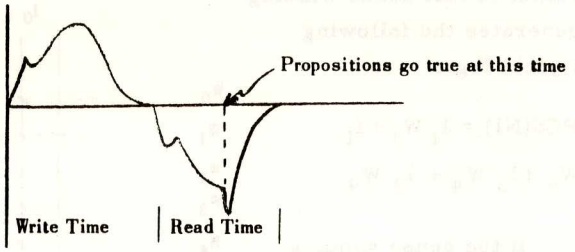


Figure C-6.

The general operation of inhibit core logic can be described in conjunction with the flip-flop model of Figure C-7. The flip-flop provides current outputs. If the flip-flop is "true", current flows in F but not in F′. If the flip-flop is "false", current flows in F′ but not in F. These flip-flop outputs are, then, the proposition currents to the logical cores. The state of the flip-flop is changed by a negative voltage pulse applied to either f or $_{o}$f. This triggering voltage is obtained directly from a sense winding in series with the required number of cores to accomplish the input logic for the flip-flop. It is a property of the flip-flop that its triggering discrimination is largely on a $\Delta\phi$ or volt-second basis. In this manner it discriminates very well against shuttle inputs, which are characterized by rather high voltage amplitude but small volt-second area; and it is not affected by the core switching faster during read time. The basic system timing is of course synchronous with the write clock determining the speed of the sequential logical operations.

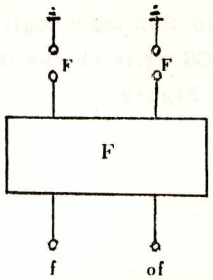


Figure C-7

It is now possible to designate several properties of the system proposed. Firstly, the flip-flop outputs need provide essentially no power. When current is flowing in a proposition line no voltage is developed across the line and when voltage is developed across the line no current is flowing through it. All power to trigger the flip-flop is derived from the write clock current and bias current, through transformer action to the sense output. Secondly, there is no interaction or loading of one core on another. This is true because line currents are constant, that is, are designed to be

independent of the voltages developed across the line. The exception is the sense winding where the current in the line is made to be negligible. Thirdly, the logical mechanization proposed is very analogous to conventional mechanization. The only major difference being that current propositions, rather than voltage propositions are required. It will be shown later that the flip-flop proposed provides both a current proposition and a voltage proposition. The voltage proposition is more conveniently utilized for memory address selection, system intercommunication, and possibly in a few other instances. Fourthly, the permissible number of logical levels is two. That is, all logic into the flip-flop is of the "products-summed" form. This is not a restriction as most contemporary systems under development are of this form. In the basic system "control" proposed (using program count or program count sum "write clocks") the two level logic technique is particularly adaptable. Lastly, a most significant feature of the inhibit core logic is the fact that but one core is required to create a product of N propositions and no additional cores are required to sum these products. In a conventional mechanization a diode or transistor is required for each proposition of the product and another diode or transistor is required for each product to be summed. It has been our experience in mechanizing typical systems that one core will replace approximately 4 diodes or transistors in the system logic.

Mechanization of Magnetic Core Control Logic

In any computer or processor there must be some basic control mechanization. A very formalized and versatile one is the program counter approach used in the NCR systems and in the system proposed. This basic control, in conjunction with a flow diagram approach to the logical design, yields a system that is very straight forward to understand and extremely easy to maintain. The application of this control approach was made practical with the conception of magnetic core program count sum arrays. This technique is already in use in the NCR 304 system.

An example of a program count sum array is shown in Figure C-8. It will be noted that access to any core is gained through addressing by an instruction register and a word counter. For a particular instruction, during a particular word time, one core can be made to switch back and forth at a repetition rate desired for write clocks. By winding sense windings through specific cores, the output of that sense winding generates the following typical logic.

$$PCS(N1) = I_1 W_1 + I_1 W_9 + I_3 W_4 + I_9 W_9$$

If the sense winding output is used to generate a write clock, this write clock can now be used directly in the flip-flop logical cores. A typical flip-flop input logic, using PCS write clocks, is shown in Figure C-9.

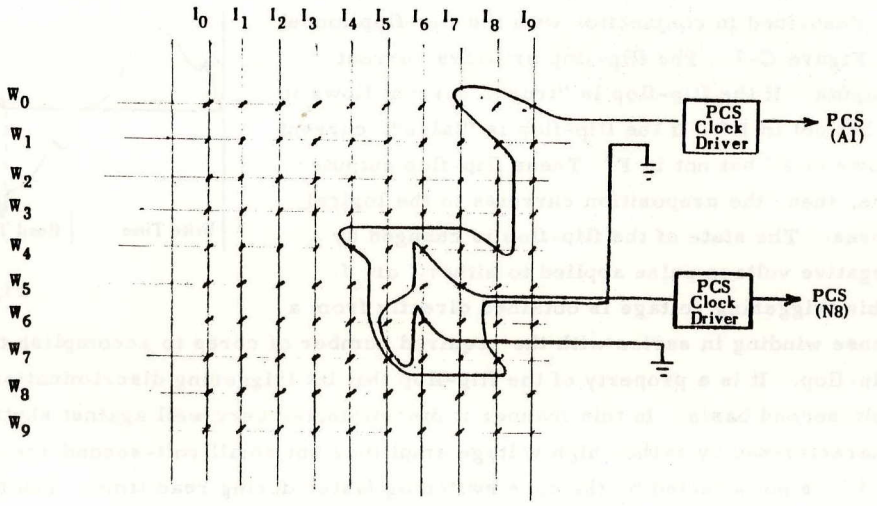


Figure C-8.

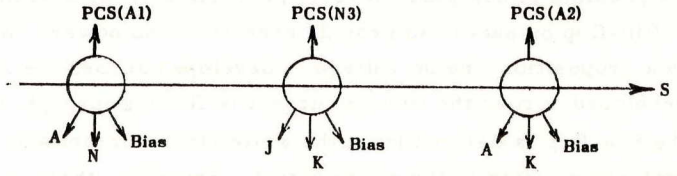
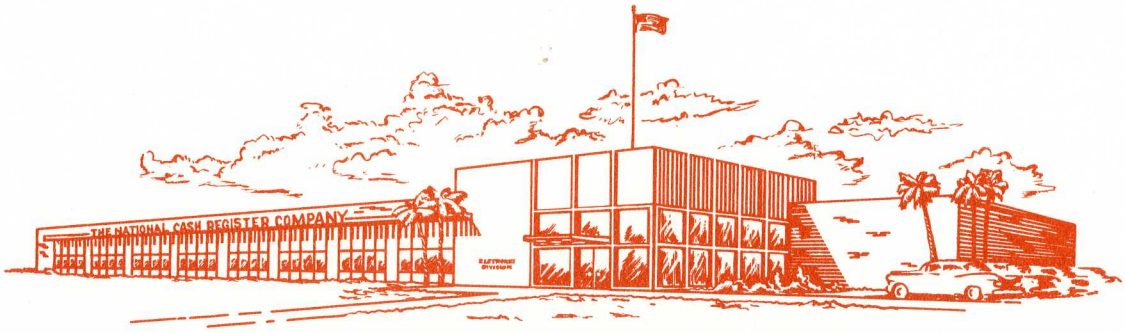


Figure C-9



National

Reg. U. S. Pat. Off.

THE NATIONAL CASH REGISTER COMPANY

1401 EAST EL SEGUNDO BOULEVARD

HAWTHORNE, CALIFORNIA