

Digital Computer Laboratory  
 Massachusetts Institute of Technology  
 Cambridge, Massachusetts

SUBJECT: DESIGN OF A DIGITAL COMPUTER BY BOOLEAN ALGEBRA

To: N. H. Taylor

From: R. C. Jeffrey, I. S. Reed

Date: May 20, 1952

Abstract: The techniques described in E-458-1 are illustrated in a real-life situation: the design of a 4-order machine employing an unusual method of central control. For definiteness the memory size is taken to be 8 words, but no logical complexity is added when this is increased to a realistic figure.

1.0 SPECIFICATIONS FOR THE MACHINE: NOMENCLATURE

1.1 Words: 5 bits, interpreted as in WWI, with negative numbers represented in nine complement form.

Interpreted as a number:  $\overbrace{Sg(X)}$

Interpreted as an instruction:  $\overbrace{X_0 X_1 X_2 X_3 \boxed{X_4}}^{Op(X) \quad Ad(X)}$

1.2 Memory: 8 words, stored in flip-flops. An actual memory would use some other device, such as iron cores; but the analysis for the flip-flop case can easily be applied to whatever device is actually used.

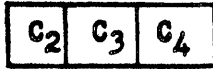
1.3 Registers

Arithmetic Element

<u>A-Register</u>	<u>B-Register</u>																				
Receives number from storage	Accumulator (A subtracter)																				
<table border="1" style="width: 100%; text-align: center;"> <tr> <td>A<sub>0</sub></td> <td>A<sub>1</sub></td> <td>A<sub>2</sub></td> <td>A<sub>3</sub></td> <td>A<sub>4</sub></td> </tr> <tr> <td colspan="2"><math>\overbrace{Op(A)}</math></td> <td colspan="3"><math>\overbrace{Ad(A)}</math></td> </tr> </table>	A <sub>0</sub>	A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>	A <sub>4</sub>	$\overbrace{Op(A)}$		$\overbrace{Ad(A)}$			<table border="1" style="width: 100%; text-align: center;"> <tr> <td>B<sub>0</sub></td> <td>B<sub>1</sub></td> <td>B<sub>2</sub></td> <td>B<sub>3</sub></td> <td>B<sub>4</sub></td> </tr> <tr> <td colspan="5"><math>\overbrace{Sg(B)}</math></td> </tr> </table>	B <sub>0</sub>	B <sub>1</sub>	B <sub>2</sub>	B <sub>3</sub>	B <sub>4</sub>	$\overbrace{Sg(B)}$				
A <sub>0</sub>	A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>	A <sub>4</sub>																	
$\overbrace{Op(A)}$		$\overbrace{Ad(A)}$																			
B <sub>0</sub>	B <sub>1</sub>	B <sub>2</sub>	B <sub>3</sub>	B <sub>4</sub>																	
$\overbrace{Sg(B)}$																					

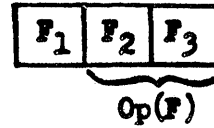
C-Register

Holds address of next memory register to be used.



F-Register

Operation counter; central control.

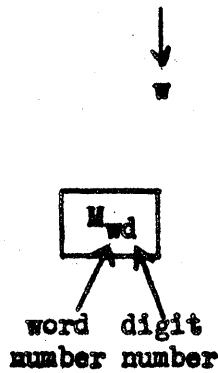


Start Flip-Flop



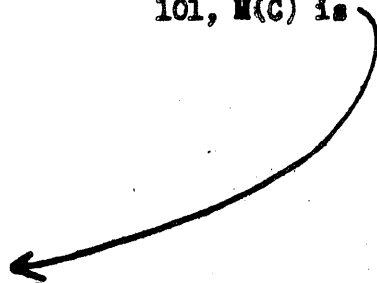
Memory

→ d



M <sub>00</sub>	M <sub>01</sub>	M <sub>02</sub>	M <sub>03</sub>	M <sub>04</sub>
M <sub>10</sub>	M <sub>11</sub>	M <sub>12</sub>	M <sub>13</sub>	M <sub>14</sub>
M <sub>20</sub>	M <sub>21</sub>	M <sub>22</sub>	M <sub>23</sub>	M <sub>24</sub>
M <sub>30</sub>	M <sub>31</sub>	M <sub>32</sub>	M <sub>33</sub>	M <sub>34</sub>
M <sub>40</sub>	M <sub>41</sub>	M <sub>42</sub>	M <sub>43</sub>	M <sub>44</sub>
M <sub>50</sub>	M <sub>51</sub>	M <sub>52</sub>	M <sub>53</sub>	M <sub>54</sub>
M <sub>60</sub>	M <sub>61</sub>	M <sub>62</sub>	M <sub>63</sub>	M <sub>64</sub>
M <sub>70</sub>	M <sub>71</sub>	M <sub>72</sub>	M <sub>73</sub>	M <sub>74</sub>

"M(C)" denotes the address in memory corresponding to the number in the C-register. Thus if the C-register holds 101, M(C) is



1.4 Operations

The decisions as to word length, number of registers, etc., summarized above come under the heading of machine planning, and were arrived at by cut-and-try. The last and most delicate part of the planning concerns the operation of central control.

<u>Code No.</u>	<u>Name</u>	<u>Description</u>
00xyz	Halt	The machine stops, i.e., all flip-flops remain in the states they were in when the operation was executed, until the start button is depressed, at which time program timing begins.
01xyz	Conditional Subprogram	If the sign digit of the B-register is 1, take next instruction from memory location xyz. If the sign of the B-register is 0, take next instruction in sequence.
10xyz	Subtract	Take number from memory location xyz and put it in the A-register; subtract it from the contents of the B-register and leave the result in B.
11xyz	Shift right	Shift the contents of the B-register right, depositing the original contents of B in M(C).

## 2.0 TIMING DIAGRAM

### 2.1 Notation

The above four operations are to be performed as sequences of commands to perform the elementary independent functions of which the machine is capable. In order to describe these functions we adopt a compact terminology:

$M(C)$ : The memory location whose number is stored in C.

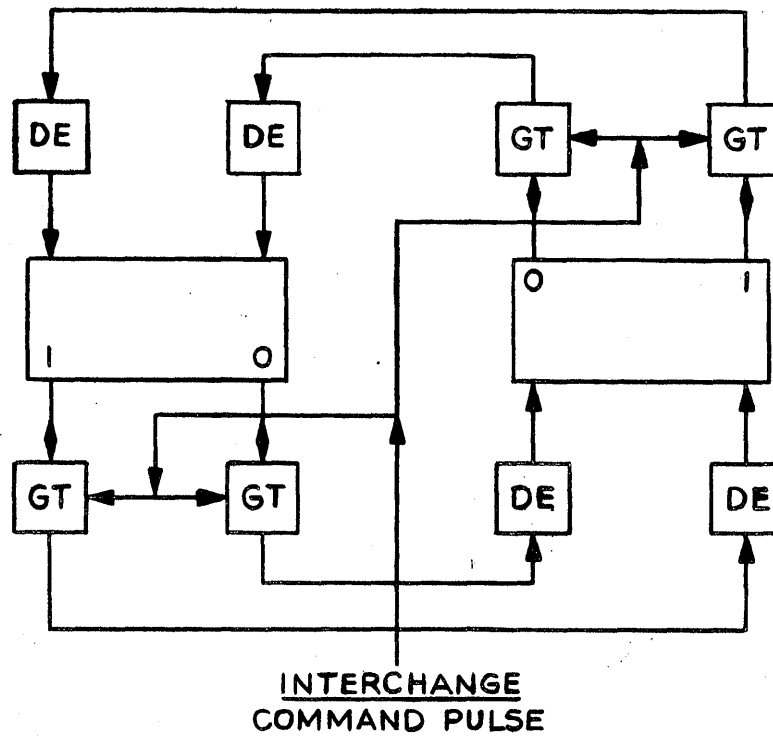
X: The X register.

(X): The contents of the X register.

$(X) \Rightarrow Y$ : The contents of register X at time t appear in register Y at time  $t + \epsilon$ .

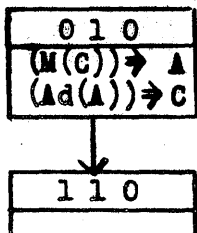
Then it is possible to have "simultaneous" interchanges:

" $(X) \Rightarrow Y$  and  $(Y) \Rightarrow X$ " means that the contents of X at t appear in Y at  $t + \epsilon$ , and that the contents of Y at t appear in X at  $t + \epsilon$ . This can be implemented by delay elements (which we shall henceforth assume to be built into the FF's):



We now draw a flow diagram for central control (next page). The numbers at the tops of the boxes will represent states of the F-counter, and the notations inside the boxes represent the commands which are performed by the first time pulse during which the F-counter is in the indicated state.

For example, in the upper left corner on page 5

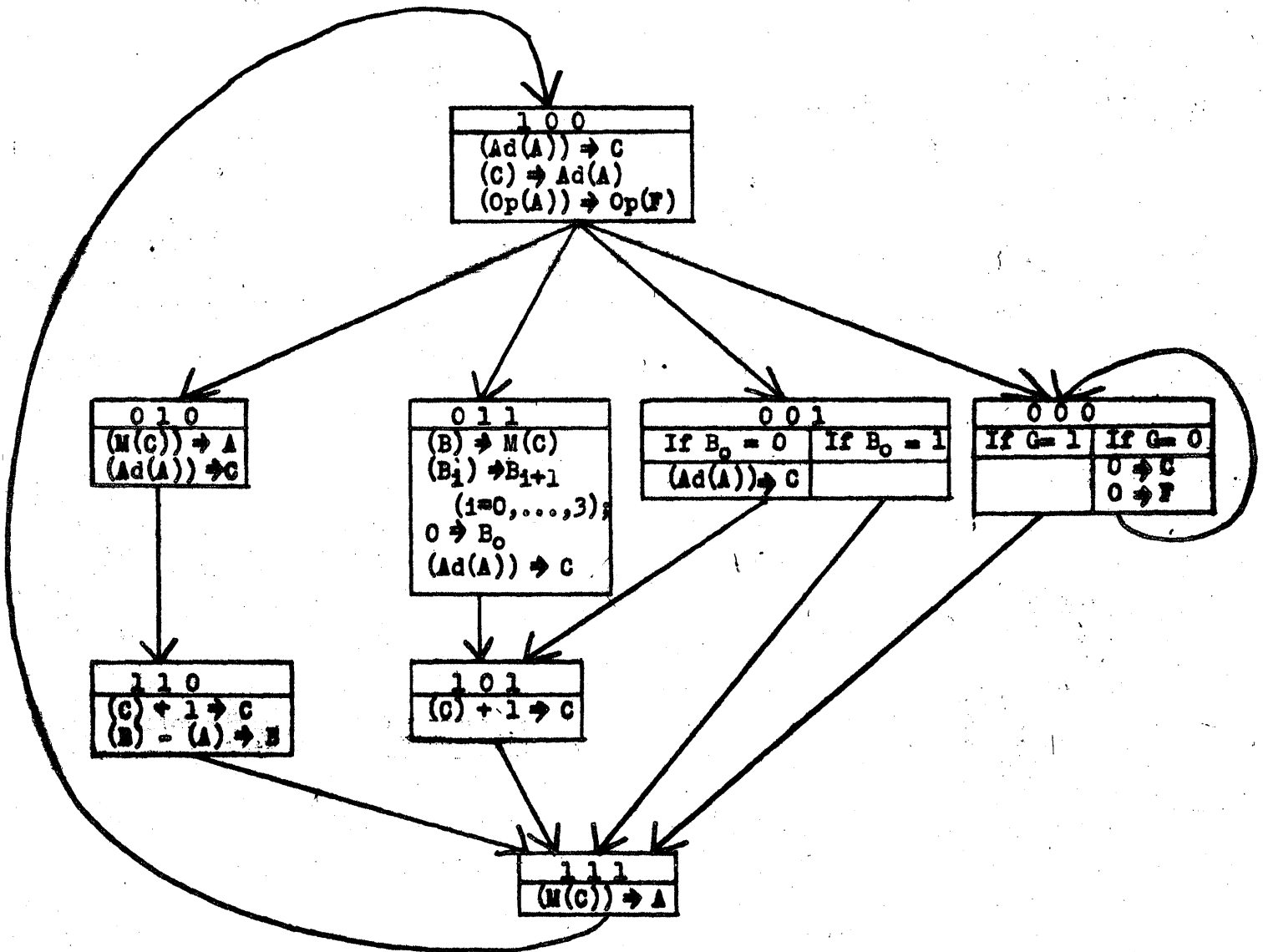


means that when the F-register is in the configuration  $F_1 = 0, F_2 = 1, F_3 = 0$  the commands read out of memory location (C) into the A-register and read the last three (address) digits of the A-register into the C-register are performed and the F-counter changes to the configuration 110 (i.e.,  $F_1$  is complemented). This change in (F) is strictly speaking a command, but is indicated by the arrow between the boxes instead of by a third notation inside the upper box.

For further explanation of the flow diagram, see page 6.

2.2 Flow Diagram

(The numbering of the boxes should be thought of as arbitrary, for the present. See Section 4.0 below.)



Note: If G = 0, then regardless of the state of F, both the F and C registers are cleared:

If G = 0
0 → C
0 → F

Program timing occurs at 111 and 100. Note that the C-register is used both as a storage selection register and a program counter: in 110 and 101 we add 1 to (C). Each of the four paths after 100 corresponds to an operation:

- 010 begins the operation timing for subtract (10xyz)
- 011 begins the operation timing for shift (11xyz)
- 001 begins the operation timing for cp (01xyz)
- 000 begins the operation timing for halt (00xyz)

The four way decision is determined by the result of the command  $(Op(A)) \Rightarrow Op(F)$ , which reads  $(A_0)$  into  $F_2$  and  $(A_1)$  into  $F_3$ . At the beginning of operation timing for operation number  $v$   $w$  the F-register will be in configuration  $0vw$ , since previously we had  $(A_0) = v$  and  $(A_1) = w$  for  $Op(A)$ . It is also necessary to arrange that  $F_1$  be complemented when the F-counter leaves configuration 100.

The reader should now verify that the sequences of commands specified in the flow diagram really do add up to the four operations listed above. The logical details are discussed in section 3.

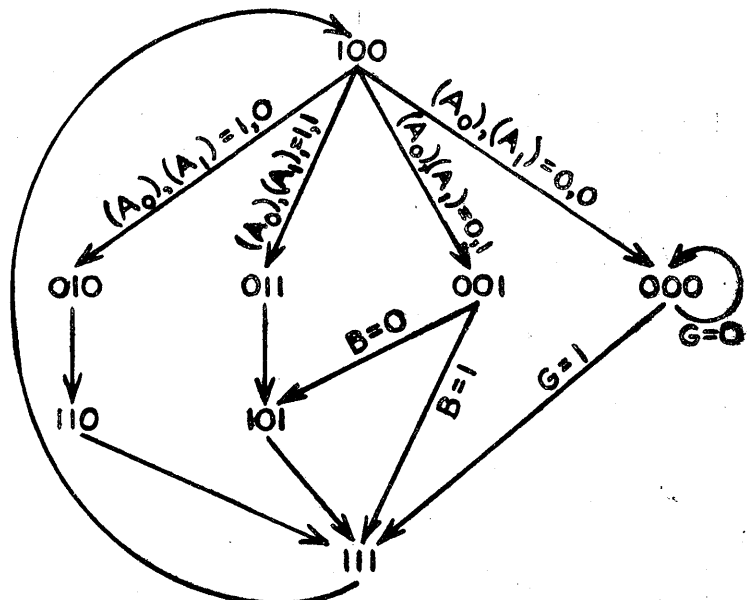
### 3.0 DESIGN OF THE MACHINE: INPUT EQUATIONS

#### 3.1 Design of the F-Register

Apparently the F-register is to be a 3 stage binary counter with the following cycle:

We name the configurations of the F-register:

- $P_0 = F_1 F_2 F_3 \text{ --- } 000$
- $P_1 = F_1 F_2 F_3 \text{ --- } 001$
- $P_2 = F_1 F_2 F_3 \text{ --- } 010$
- $P_3 = F_1 F_2 F_3 \text{ --- } 011$
- $P_4 = F_1 F_2 F_3 \text{ --- } 100$
- $P_5 = F_1 F_2 F_3 \text{ --- } 101$
- $P_6 = F_1 F_2 F_3 \text{ --- } 110$
- $P_7 = F_1 F_2 F_3 \text{ --- } 111$

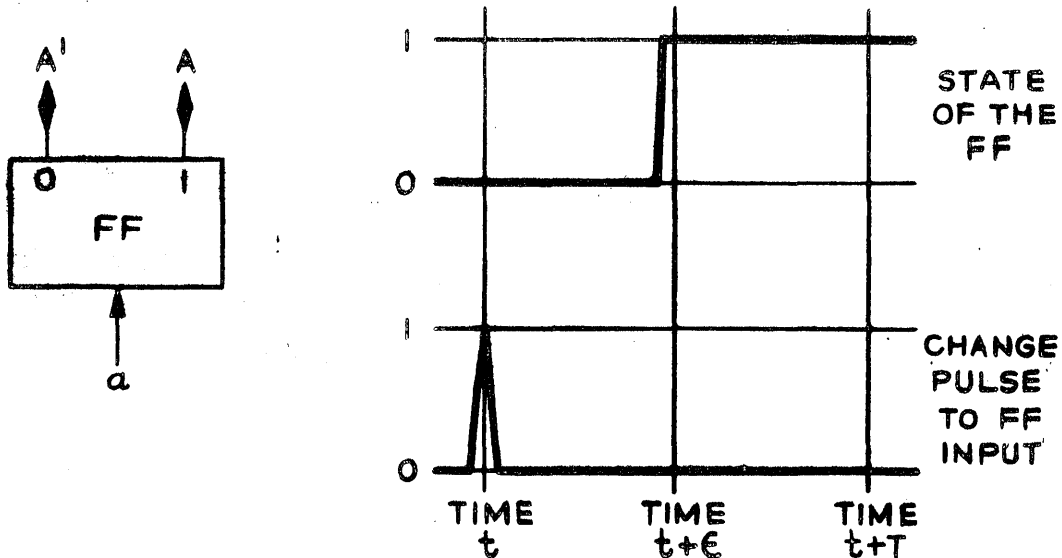


The block diagram on page 9 shows how voltages corresponding to the  $P_i$  are generated.

In the table above, the binary representation of the subscript of P indicates, by the distribution of zeros, the location of the primes in the product of the F's.

Now the successive states of the counter are determined by information external to the counter, as well as from the configuration of the counter itself. For example, from P<sub>3</sub> the counter must go to P<sub>5</sub>, but from P<sub>1</sub> it may go to P<sub>5</sub> or to P<sub>7</sub>. It is necessary to indicate the basis on which this selection is made; this is done in the cases where a choice is necessary.

Assume that the FF's used have one input, such that if that input is pulsed at time t, the FF is complemented at time t+ε, ε being a delay less than T, the period of the clock.



$$A(t + T) = A(t) \oplus a(t) \quad (\text{See E-458-1, p. 13})$$

Now to design the F-counter it is sufficient to write three equations, one for each stage, specifying when the change inputs are to be pulsed. We shall work through the lattice on page 6 (or page 5) level by level for each input. The equations we are about to build up are illustrated by a block diagram on page 9.

For the first digit (F<sub>1</sub>) we see that in going from P<sub>4</sub> (=100) on the first level to the second, F<sub>1</sub> is to be complemented regardless of which of the four configurations on the second level is selected:

$$f_1 = (P_4 + \dots\dots\dots)E$$

Here E denotes the clock pulse: the changes are to occur at each clock pulse, so that an operation will require at most 4 clock pulses for its completion.

To get the second term: if  $P_2 (=010)$  is the second level configuration selected, the first digit must change again to get the configuration 110:

$$f_1 = (P_4 + P_2 + \dots)E$$

Ditto in case 011 was the second level configuration selected: to get from 011 to 101,  $F_1$  must be complemented:

$$f_1 = (P_4 + P_2 + P_3 + \dots)E$$

Similarly, whichever value  $B_0$  may have, the F-counter will go from the configuration 001 to a configuration (101 or 111) in which the first digit is changed:

$$f_1 = (P_4 + P_2 + P_3 + P_1 + \dots)E$$

Finally, the counter goes from 000 to 111 in case  $G=1$ . Thus  $F_1$  will change in case  $P_0G$ . (" $P_0G$ " means  $P_0 = 1$  and  $G = 1$ .)

$$f_1 = (P_4 + P_2 + P_3 + P_1 + P_0G)E$$

Since  $F_1$  does not change in going from 111 to 100, the above formula represents all the conditions under which  $F_1$  must be complemented.

Similarly, for  $F_2$  we have

$$f_2 = (A_0P_4 + P_3 + B_0P_1 + GP_0 + P_5 + P_7)E$$

where the first term derives from the fact that in the two cases where  $A_0 = 1$  [ $(O_p(A)) = 10$  and  $(O_p(A)) = 11$ ] the second digit of "100" must be changed, but in the other two cases (when "100" goes to "001" or to "000") this is not necessary. Finally

$$f_3 = (A_1P_4 + GP_0 + P_6 + P_7)E$$

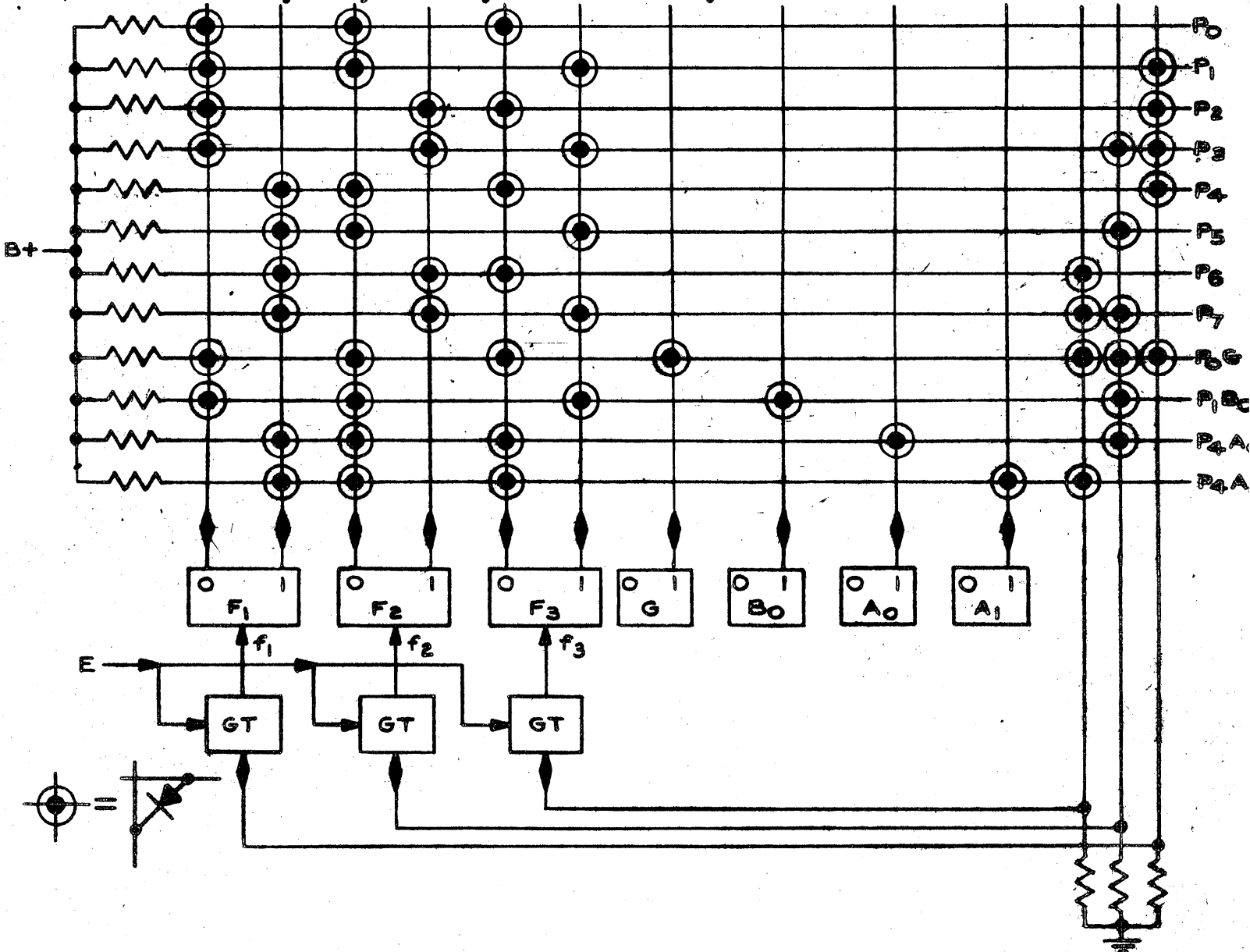
It remains to implement the condition that if  $G=0$ ,  $O=F$ . Note that if a halt is programmed, the F-counter steps from 100 to 000 and in so doing clears  $G$ . But as will be seen in Section 3.35 below,  $G$  can also be cleared by a pushbutton. We wish to provide that if  $G=0$  from any cause, the F-counter will assume the configuration 000 on the next time pulse.

This is most easily accomplished by writing three equations for the clear inputs to the F flip-flops:

$$f_{0i} = G^i E \quad i = 1, 2, 3$$



Now these equations may be realized immediately by a two-level diode matrix which, although it does not represent a minimization of number of crystals, still may be electronically desirable.

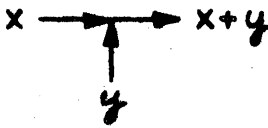
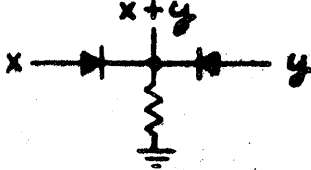
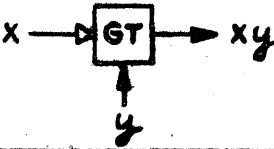
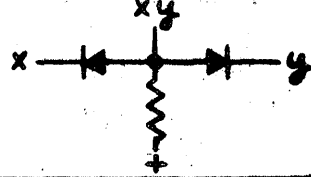
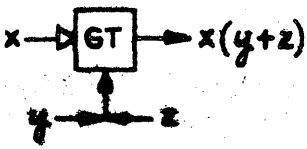
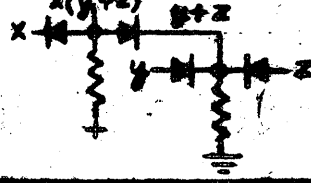
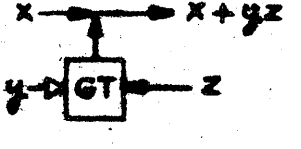
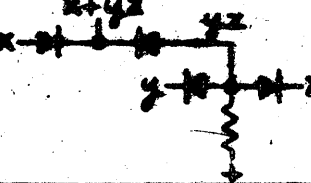
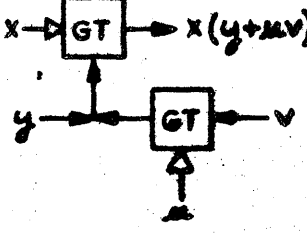
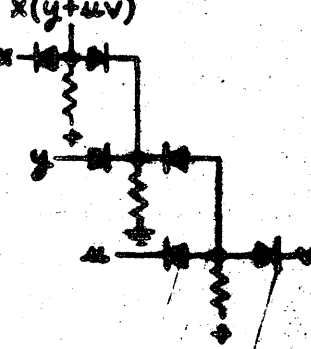


Here we have indicated only those 3 outputs of the diode matrix which feed back into the F-register. Actually we require other outputs; command pulses to the rest of the machine. They will be considered below in connection with the FF registers which they control.

3.2 Digression on "Levels"

Note that the three equations for the F-register can be factored, resulting in a reduction in the number of gates and mixers required for the realization of central control. But this numerical reduction is at the expense of an increase in the number of levels of gating and mixing. We illustrate the meaning of "levels" by examples.

EXAMPLE

No. of Levels	Equations	Block Diagrams	One Electronic Realization (Diode)
1	$x + y$		
	$xy$		
2	$x(y + z)$		
	$x + yz$		
3	$x[y + uv]$		

Apparently in terms of the equations, the number of levels is the same as the functional complexity of the expression for the output. That is,  $x + y$  and  $xy$  are functions whose arguments are variables;  $x(y + z)$  and  $x + (yz)$  are functions (the first a product and the second a sum) in which one of the arguments is itself a function of variables.

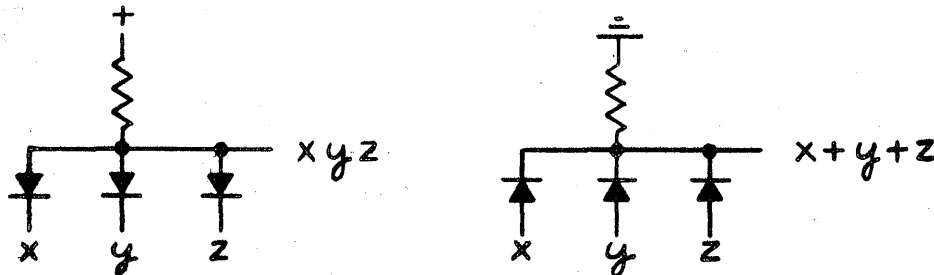
Finally,  $x[y + uv]$  is a product one of whose factors is a sum one of whose terms is a product: " $\pi\{x, \sigma\{y, \pi\{u, v\}\}\}$ " in functional notation where  $\pi(a, b) = ab$  and  $\sigma(a, b) = a + b$ . Briefly, the number of levels in a Boolean function is identical with the number of nestings of parentheses within each other.

In terms of block diagrams, the number of levels in a device is determined by drawing lines from each input to the output; the largest number of gates and mixers through which such a line passes in the level of the system. Finally, the reason we are interested in levels is that if a function of level  $n$  is realized with diodes,  $n$  is the largest number of diodes through which any input current must pass in order to reach the output. In a high-level diode network considerable attention must be paid to the values of voltage and resistance associated with the diodes. Other difficulties arise with high-level vacuum tube and transistor networks. In all cases the difficulties simply require engineering attention: high-level networks are realizable.

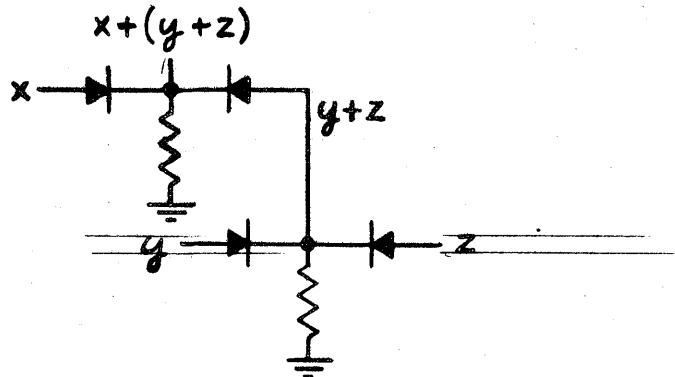
One last comment: in such expressions as " $xyz$ " and " $x + y + z$ ", the product and sum are regarded as functions of three variables:

$$\pi_3(x, y, z) = xyz \text{ and } \sigma_3(x, y, z) = x + y + z$$

Electronically this corresponds to the use of three-input diode devices for mixing and gating:



and therefore three-input gates and mixers are first level networks. However, expressions like " $x + (y + z)$ ", which might be written in functional notation as " $\sigma_2(x, \sigma_2(y, z))$ " have as their direct realizations two-level mixers like:



in which the longest current path passes through two diodes. The two sorts of mixers are logically equivalent, and the one-level type is used where possible.

To illustrate these points, let us factor the equations of the F-register:

$$\begin{aligned}
 f_1 &= P_1 + P_2 + P_3 + P_4 + P_0G \quad (\text{Ignore the "E" for the present.}) \\
 &= \underline{F_1 F_2 F_3} + \underline{F_1 F_2 F_3^0} + \underline{F_1 F_2 F_3} + F_1 F_2 F_3^0 + \underline{F_1 F_2 F_3^0} G \\
 &= F_1 (F_2 F_3 + F_2 F_3^0 + F_2 F_3 + F_2 F_3^0 G) + F_1 F_2 F_3^0 \\
 &= F_1 \left[ F_3 (F_2 + F_2^0) + F_3^0 (F_2 + F_2^0 G) \right] + F_1 F_2 F_3^0 \\
 &= F_1 \left[ F_3 + (F_2 + G) \right] + F_1 F_2 F_3^0 \quad (\text{by 2 applications of the rule: } x + x'y = x + y)
 \end{aligned}$$

$$f_1 = \left[ F_1 (F_2 + F_3 + G) + F_1 F_2 F_3^0 \right] E$$

$$\begin{aligned}
 f_2 &= A_0 P_4 + P_3 + B_0 P_1 + G P_0 + P_5 + P_7 \\
 &= A_0 \underline{F_1 F_2 F_3^0} + F_1 F_2 F_3 + B_0 F_1 F_2 F_3 + G F_1 F_2 F_3^0 + \underline{F_1 F_2 F_3} + \underline{F_1 F_2 F_3} \\
 &= F_1 (A_0 F_2 F_3^0 + F_2 F_3 + F_2 F_3) + F_1 (F_2 F_3 + B_0 F_2 F_3 + G F_2 F_3^0) \\
 &= F_1 (A_0 F_2 F_3^0 + F_3) + F_1 \left[ F_3 (F_2 + B_0 F_2) + G F_2 F_3^0 \right] \\
 &= F_1 (F_3 + A_0 F_2^0) + F_1 \left[ F_3 (F_2 + B_0) + G F_2 F_3^0 \right]
 \end{aligned}$$

$$f_2 = \left\{ F_1 (F_3 + A_0 F_2^0) + F_1 \left[ F_3 (F_2 + B_0) + G F_2 F_3^0 \right] \right\} E$$

$$\begin{aligned}
 f_3 &= A_1 P_4 + G P_0 + P_6 + P_7 \\
 &= A_1 \underline{F_1 F_2 F_3^0} + G F_1 F_2 F_3^0 + \underline{F_1 F_2 F_3^0} + \underline{F_1 F_2 F_3} \\
 &= F_1 (A_1 F_2 F_3^0 + F_2 F_3^0 + F_2 F_3) + G F_1 F_2 F_3^0 \\
 &= F_1 (A_1 F_2 F_3^0 + F_2) + G F_1 F_2 F_3^0
 \end{aligned}$$

$$f_3 = \left[ F_1 (F_2 + A_1 F_3^0) + G F_1 F_2 F_3^0 \right] E$$

We have now reduced the total number of diodes (assuming a realization as a diode net) from 55 in the two-level matrix to 41 [10 for  $f_1$ , 19 for  $f_2$ , 12 for  $f_3$ ; this can be determined directly from the equation by counting  $n$  diodes for each  $n$  input gate or mixer. E.g., for  $f_3$ ..... ignoring E..... we have a 2 input mixer (the plus in front of "G") of which the inputs are a 4 input gate and a 2 input gate of which the inputs are  $F_1$  and a 2 input mixer of which the inputs are  $F_2$  and a 2 input gate. Adding up the numbers of inputs underlined above gives the 12 diodes required for  $f_3$ .]

We shall not draw the block diagrams for the factored equations. It would be very messy, and an engineer who understands the algebraic notation could design the network from the equations as easily as he could from the block diagrams (the equations have the added advantage of compactness: three rows of symbols, instead of a page crammed with boxes, lines and arrows).

### 3.3 The Arithmetic Element

#### 3.31 The Subtractor (See diagram on p. 26)

The most complicated single command the machine must execute is subtract  $[(B) - (A) \Rightarrow B]$ : from the contents of the B-register, subtract the contents of the A-register and deposit the result back in  $B_d$

Now representing the binary number stored in the A-register by 'A', and similarly for B, and denoting the nines complement (result of complementing each digit) of A by ' $\bar{A}$ ', we have

$$B = B_0 2^0 + B_1 2^{-1} + B_2 2^{-2} + B_3 2^{-3} + B_4 2^{-4}$$

$$A = A_0 2^0 + \dots + A_4 2^{-4}$$

$$\bar{A} = (1 - A_0) 2^0 + \dots + (1 - A_4) 2^{-4}$$

$$A + \bar{A} = 2^0 + \dots + 2^{-4} = 2 - 2^{-4}$$

$$-A = \bar{A} + 2^{-4} \rightarrow 2$$

$$\boxed{B - A = B + \bar{A} + 2^{-4} - 2}$$

We can now construct a table which describes the operation of the  $i$ th stage of our subtractor; it is the ordinary table for binary addition where the second term is  $A_i$  instead of its complement,  $\bar{A}_i$ . Note that we require an "instantaneous" carry output,  $K_{i-1}$ .

$B_1(t)$	$A_1^0(t)$	$K_1(t)$	$K_{i-1}(t)$	$B_1(t + T)$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

all possible inputs
 $B + \bar{A}$

We want in the B-register at time  $t + T$  not  $B + \bar{A}$ , but rather  $B + \bar{A} + 2^{-4} - 2$ , which we have shown above to be equal to  $B - A$ . (cf. end-around carry)

To get from  $B + \bar{A}$  to  $B + \bar{A} + 2^{-4} - 2$ : we add the  $2^{-4}$  by requiring that  $K_4$ , the carry input to the first stage, be 1.

We subtract the 2 by failing to provide a  $B_{-1}$  flip-flop.

Then a realization of the table which has the two additional properties just listed will subtract the contents of A from the contents of B and leave the result in the B-register T seconds after the subtract command pulse,  $P_6E$  (note that according to the table on p. 5 the  $(B) - (A) \Rightarrow B$  command goes out when the F-counter is in the configuration 110 ( $P_6$ ) and a clock pulse (E) occurs).

The input to  $B_1$  is to be pulsed whenever we wish  $B_1(t + T)$  to be the complement of  $B_1(t)$  and a subtract command pulse occurs. Comparing the first and last columns of the table we get

$$b = (B_1^0 A_1 K_1 + B_1^0 \bar{A}_1 \bar{K}_1 + B_1^1 A_1 K_1 + B_1^1 \bar{A}_1 \bar{K}_1) P_6 E$$

If this seems wrong, notice that the table lists values of  $A^0$ , not values of A, and that  $A_1^0 = 0$  means:  $A_1 = 1$ .

This expression can be greatly simplified by factoring  $A_1 K_1$  from the first and third terms, and  $\bar{A}_1 \bar{K}_1$  from the other two:

$$b = (A_1 K_1 + \bar{A}_1 \bar{K}_1) P_6 E = (A_1 \oplus K_1) P_6 E \quad (i = 0, \dots, 4)$$

which fact might have been seen directly from the table.

The carry output from this stage is

$$K_{i-1} = (B_i^0 A_i^0 K_i + B_i^1 A_i^1 K_i + B_i^2 A_i^2 K_i + B_i^3 A_i^3 K_i) P_6 E$$

$$K_{i-1} = [A_i^i K_i + B_i (A_i^i \oplus K)] P_6 E$$

$$= A_i^i K_i P_6 E + B_i b \quad i = 1, 2, 3, 4; K_4 = 1$$

### 3.32 Shifting

The B-register is used in shifting as well as subtracting. Then before we adopt the above equation for the input to the B-register we must add terms which take account of the additional inputs required for shifting.

Referring to the diagram on p. 5 we see that the function (B) - (A)  $\Rightarrow$  B, which occurs in configuration 110 of the F-counter, is not the only one that involves B.

In configuration 011 ( $P_3$ ) of the F-counter two other functions are performed: (B)  $\Rightarrow$  M[C] and  $(B_i) \Rightarrow B_{i-1}$ , it being understood that the number which appears in M[C] is (B) before the shift. (See remarks on "simultaneous" interchanges between registers in Section 2.1.) In the shift a zero is introduced in the left-most digit position,  $B_0$ .

The transfer to memory, (B)  $\Rightarrow$  M[C], does not concern the input to the B-register: it will be taken account of in the input equations to memory. Then the input equations for b will be complete when we add expressions to realize  $(B_i) \Rightarrow B_{i+1}$  ( $i = 0, 1, 2, 3$ );  $0 \Rightarrow B_0$ . This is most easily done using the clear and set inputs to the B flip-flops, although it might have been accomplished with only the complement input, with suitable gating. Then we consider that we have the WWI type of FF with three inputs, labeled as shown; the complete equations for the B-register are then

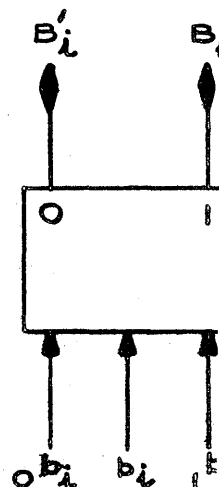
$$b_i = (A_i^i \oplus K_i) P_6 E \quad (i = 0, 1, 2, 3, 4) \quad (\text{difference digit})$$

$$K_{i-1} = A_i^i K_i P_6 E + B_i b \quad (i = 1, 2, 3, 4) \quad (\text{carry})$$

$$K_4 = 1 \quad (\text{"end-around carry"})$$

$$\left. \begin{aligned} 1^i b_i &= B_{i-1}^i P_3 E \\ 0^i b_i &= B_{i-1}^i P_3 E \end{aligned} \right\} \quad (i = 1, 2, 3, 4) \quad (\text{shift right})$$

$$0^0 b_0 = P_3 E \quad (\text{destructive shift})$$



That is, when the F-counter is in configuration  $P_3$ , the clock pulse E acts as a command for the contents of register no.  $i-1$  to appear in register no.  $i$ , and for a zero to appear in register  $B_0$ . It is assumed, once again, that there is a delay inherent in the inputs to our FF's such that this shifting will occur after the transfer to memory has taken place.

3.33 The A-Register (See diagram, p. 25)

To complete the discussion of the arithmetic element we write the equations of the inputs to the A-register. Transfers into A occur on configurations 100, 010, and 111 ( $P_4$ ,  $P_2$  and  $P_7$ ) of the operation counter (see p. 5). Again it will be simplest to use the set and clear inputs rather than the complement input:

$$\left. \begin{aligned}
 1^a_i &= \left[ C_1^i P_4 + M_{hi} (P_2 + P_7) \right] E \\
 0^a_i &= \left[ C_1^i P_4 + M_{hi} (P_2 + P_7) \right] E \\
 1^a_i &= M_{hi} (P_2 + P_7) E \\
 0^a_i &= M_{hi} (P_2 + P_7) E
 \end{aligned} \right\} \begin{array}{l} i = 2, 3, 4 \\ \\ i = 0, 1 \end{array} \quad h = (C)$$

It is necessary to treat  $i = 0, 1$  separately since there are no  $C_0$  or  $C_1$  flip-flops. Now we must explicitly indicate how  $h = (C)$  is to be implemented.

Let us assign names to the configurations of the C-register in analogy to the  $P_i$  as names for the configurations of the F-register:

$$\begin{aligned}
 \Gamma_0 &= C_2^1 C_3^1 C_4^1 \\
 \Gamma_1 &= C_2^1 C_3^1 C_4^0 \\
 \Gamma_2 &= C_2^1 C_3^0 C_4^1 \\
 \Gamma_3 &= C_2^1 C_3^0 C_4^0 \\
 \Gamma_4 &= C_2^0 C_3^1 C_4^1 \\
 \Gamma_5 &= C_2^0 C_3^1 C_4^0 \\
 \Gamma_6 &= C_2^0 C_3^0 C_4^1 \\
 \Gamma_7 &= C_2^0 C_3^0 C_4^0
 \end{aligned}$$

Now the C-register will be in one and only one of these configurations at any time and hence for any fixed  $i$

$$\sum_{h=0}^7 M_{hi} \Gamma_h = M_{0i} \Gamma_0 + \dots + M_{7i} \Gamma_7$$

will be 0 or 1 accordingly as the  $i^{\text{th}}$  digit of the word in memory which corresponds to (C) is 0 or 1.

This implements the condition  $h = (C)$  which we added verbally to the equation above.



Then the final equations for the A-register are

$$\left. \begin{aligned}
 1^a_1 &= C_1 P_4 + (P_2 + P_7) \left( \sum_{h=0}^7 M_{hi} \left[ \begin{array}{c} \square \\ h \end{array} \right] \right) E \\
 0^a_1 &= G_1 P_4 + (P_2 + P_7) \left( \sum_{h=0}^7 M_{hi} \left[ \begin{array}{c} \square \\ h \end{array} \right] \right) E \\
 1^a_1 &= (P_2 + P_7) \left( \sum_{h=0}^7 M_{hi} \left[ \begin{array}{c} \square \\ h \end{array} \right] \right) E \\
 0^a_1 &= (P_2 + P_7) \left( \sum_{h=0}^7 M_{hi} \left[ \begin{array}{c} \square \\ h \end{array} \right] \right) E
 \end{aligned} \right\} \begin{array}{l} i = 2, 3, 4 \\ i = 0, 1 \end{array} \quad h = 0, \dots, 7$$

3.34 The C-Register

The inputs to the C-register are affected in configurations 100, 010, 011, 001 (in case  $B_0 = 0$ ) and 101 of the operation counter, and in case  $G = 0$ , regardless of the state of  $F$  (see p. 5). Only three different things go on:

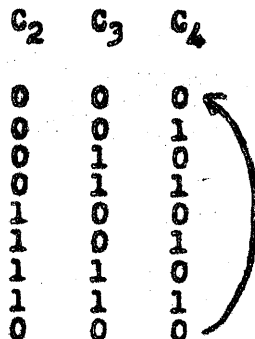
- (1)  $(Ad A) = C$  (on  $P_4, P_2, P_3$  and  $P_1 B_0$ )
- (2)  $0 = C$  (on  $G^1$ )
- (3)  $(C) + 1 = C$  (on  $P_5$ )

The first two functions are easily handled as above, using the set and clear inputs:

$$\left. \begin{aligned}
 1^c_1 &= (P_1 B_0 + P_2 + P_3 + P_4) A_1 E \\
 0^c_1 &= [(P_1 B_0 + P_2 + P_3 + P_4) A_1^c + G^1] E
 \end{aligned} \right\} i = 2, 3, 4$$

The " $G^1$ " term in the equation for  $0^c_1$  provides that when the computer is started up, the first memory address to be referred to will be 000.

The add one function can be implemented by a simple counter using the complement inputs. The cycle for the counter is:



Apparently the stages should be complemented as follows:

$$c_2 = (C_2^1 C_3 C_4 + C_2 C_3 C_4) P_5 E$$

$$= C_3 C_4 P_5 E$$

$$c_3 = C_4 P_5 E$$

$$c_4 = P_5 E$$

$$c_2 = C_3 c_3$$

$$c_3 = C_4 c_4$$

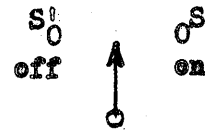
$$c_4 = P_5 E$$

### 3.35 The Start Flip-Flop, G

On the halt command, on the time pulse before the operation (F) counter reaches configuration 000 the G flip-flop is cleared and therefore the F-counter sticks on 000 (see p. 5). The machine is to be started again by use of a switch, S, which sets G and allows the next clock pulse to step the operation counter into configuration 111. The machine can be stopped not only on the halt command, but at any time by setting S to off.

$$0G = (P_4 A_0 A_1 + S^0) E$$

$$1G = SE$$



Shown in rest position.  
Can be locked in off.  
Contact in on state is momentary.

### 3.36 The Memory

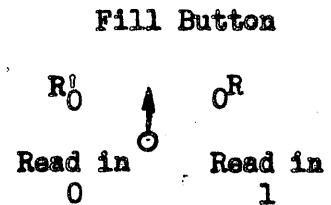
Read-in to the memory is to be accomplished in one of two ways

(1) Program: in the 011 configuration of the operation counter (B)  $\Rightarrow$  N[C].

(2) Fill Buttons: the individual flip-flops can be filled "by hand"; this method is used for reading in programs.

The "fill buttons" are switches which in their rest positions excite neither  $0^m$  nor  $1^m$ .

$$\left. \begin{aligned} 1^m_{wd} &= (R_{wd} + B_d \prod_w P_3)E & w &= 0, \dots, 7 \\ 0^m_{wd} &= (R^0_{wd} + B^0_d \prod_w P_3)E & d &= 0, \dots, 5 \end{aligned} \right\}$$



Shown in rest position.  
Contact is momentary  
in both positions.

We now have a set of equations which completely describe the logic of our machine. They are summarized on p. 21 and p. 22.

#### 4.0 GENERAL REMARKS

There are three general stages in the development of the logic of any particular computing machine. In order of decreasing generality of the decisions involved, they are:

- (1) Planning. (Given the purpose of the machine, what operations shall be included? What word-length? What speed per operation?.....)
- (2) Combinatorial Decisions. (What general arrangement of devices will best implement the plan: what shall be the cycle of the operation counter? What order code? What basic functions (commands)?.....)
- (3) Design. (What configuration of memory-elements, gates, mixers, etc. best realizes the results of the combinatorial decisions?)

In the case of the present "sample" computer these steps went somewhat as follows:

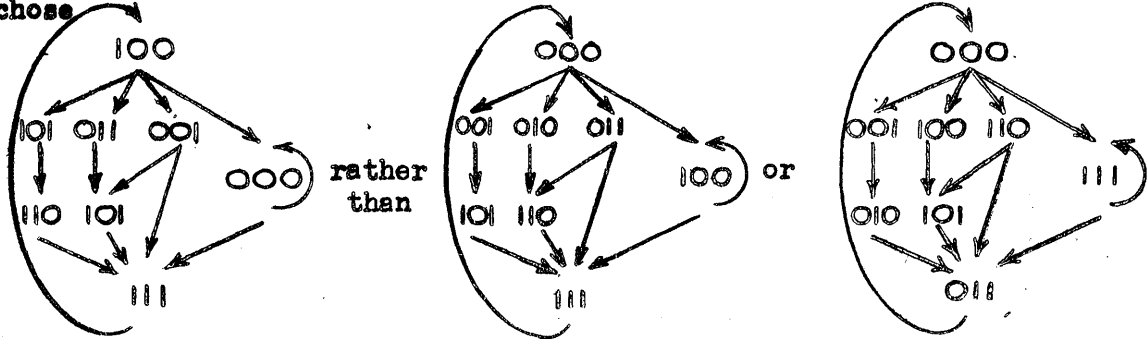
(1) Planning: The purpose of the machine was to serve as an example of a method of design. Therefore it should have a small memory, short word length and few operations; but none of these should be so short, small or few as to make the design problem trivial. The results of the planning were presented, without discussion, on the first three pages.

(2) Combinatorial Decisions: These are summarized by the table on p. 5. There the cycle of the operation-counter is shown and the groups of basic commands indicated. Since the operations are few and simple, no thought at all was devoted to the coding of them (00 =halt, 01 = cp, etc.)

(3) Design: Throughout the design we referred to the table on p. 5 which summarized the combinatorial decisions. The design is almost mechanical, once the table has been drawn.

The sort of problem involved in the combinatorial decisions is illustrated by the binary coding of the 8 boxes in the lattice on p. 5.

We chose



because we thought that of all the  $8!$  ways of assigning numbers from 0 through 7 to the 8 points in the lattice, the one selected resulted in the simplest design. Of course we did not draw up all  $8!$  lattices and from them write  $8!$  sets of equations for the machine, examine them and choose the one that contained the fewest components. Such a program might be carried out by a machine like WWI, but not by human beings. Rather we considered e.g. that it would be desirable to have as little difference as possible between the numbers assigned to the points in the second level of the diagram, the outcomes of the four-way choice after 100. Accordingly, in a simple-minded way, we decided to use the four numbers beginning with "0" for those points. Similarly in the binary decision after 001 we chose "101" and "111" as the numbers of the next configurations because they differ from each other only in a single digit.

To systematize such combinatorial decisions, three developments would be helpful:

(1) A mathematical theory of the desirability of such choices as the two mentioned above (which were made on an intuitive basis).

(2) Assignment of numerical values to different components and their configurations (the unit might be dollars, or speed, reliability or some combination of those). Any such assignment should be in general terms so that the parameters might be changed as the characteristics of available components change.

(3) Programs which would allow a high-speed computer to survey large numbers of possibilities such as the  $8!$  assignments of numbers to the points in the lattice above. It is possible to write down a fully mechanical procedure for simplifying equations and even for going from a diagram like the one on p. 5 to a set of equations, so that this development is possible - and perhaps practical.

5.0 SUMMARY

5.1 Complete Set of Equations for the Machine

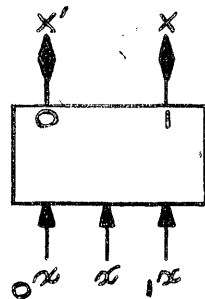
5.1.1 Abbreviations

$P_i = F_1^{*} F_2^{*} F_3^{*}$  where the  $n^{th}$  "\*" is either a prime or a blank, depending on whether the  $n^{th}$  digit in the binary representation of  $i$  is a "0" or a "1". e.g.,  $P_5 = P_{101} = F_1 F_2' F_3$ .

$$\square_i = C^* C^* C^*. \text{ E.g., } \square_1 = \square_{001} = C_1' C_2' C_3$$

$$\sum_{i=1}^n X_i = X_1 + X_2 + \dots + X_n$$

5.1.2 Notation for Flip-Flops



5.1.3 The Operation Counter (F-Register)

$$f_1 = (P_0 G + P_1 + P_2 + P_3 + P_4) E$$

$$f_2 = (P_0 G + P_1 B_0 + P_3 + P_4 A_0 + P_5 + P_7) E$$

$$f_3 = (P_0 G + P_4 A_1 + P_6 + P_7) E$$

$$0 f_i = G' E \quad (i = 1, 2, 3)$$

5.1.4 The Accumulator (B-Register)

$$b_i = (A_i' \oplus K_i) P_6 E \quad (i = 0, 1, 2, 3, 4)$$

$$\left. \begin{aligned} 1^b_i &= B_{i-1} P_3 E \\ 0^b_i &= B_{i-1}' P_3 E \end{aligned} \right\} (i = 1, 2, 3, 4)$$

$$0^b_0 = P_3 E$$

$$K_{i-1} = A_i' K_i P_6 E + B_i b \quad (i = 1, 2, 3, 4); \quad K_4 = 1$$

5.15 The A-Register

$$\left. \begin{aligned} 1a_i &= \left[ C_i P_4 + (P_2 + P_7) \left( \sum_{h=0}^7 M_{hi} \begin{bmatrix} \cdot \\ h \end{bmatrix} \right) E \right] \\ 0a_i &= \left[ C_i P_4 + (P_2 + P_7) \left( \sum_{h=0}^7 M_{hi} \begin{bmatrix} \cdot \\ h \end{bmatrix} \right) E \right] \end{aligned} \right\} \begin{array}{l} i = 2, 3, 4; \\ h = 0, \dots, 7 \end{array}$$

Same for  $i = 0, 1$  except that 1st term is missing.

5.16 The Memory-Selection Register (C)

$$\left. \begin{aligned} 1c_i &= XA_i E \\ 0c_i &= (XA_i + G_i) E \end{aligned} \right\} i = 2, 3, 4$$

where  $X = P_1 B_0 + P_2 + P_3 + P_4$ .

$$c_i = C_{i+1} \quad (i = 2, 3)$$

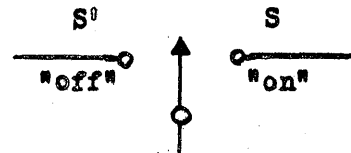
$$c_4 = P_5 E$$

5.17 The Start-Stop Flip-Flop (G)

$$0g = (P_4 A_0 A_1 + S^i) E$$

$$1g = SE$$

S is a double-throw switch which can be locked in the off position, makes momentary contact in the on position, and has a rest position in which neither contact is energized.

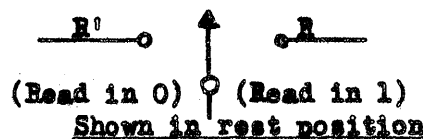


Shown in "rest" position.

5.18 The Memory Registers (M)

$$\left. \begin{aligned} 1^m_{vd} &= (R_{vd} + B_d \begin{bmatrix} \cdot \\ v \end{bmatrix} P_3) E \\ 0^m_{vd} &= (R^0_{vd} + B^0_d \begin{bmatrix} \cdot \\ v \end{bmatrix} P_3) E \end{aligned} \right\} \begin{array}{l} v = 0, \dots, 7; \\ d = 0, \dots, 5 \end{array}$$

The  $R^i$ 's are double-throw switches ("fill buttons") which make momentary contact on either side and have a rest position in which both sides are open.



Shown in rest position

## 5.2 The Flip-Flops

These are assumed to have an inherent delay between receipt of an input pulse and the resulting change in the output. Speaking loosely, this delay permits reading new information into a register "at the same time" that the old information is being read out, and permits "simultaneous" transfers of information between two registers. (See Section 2.1.)

## 5.3 Operating Instructions

1. Turn on power and allow tubes to warm up. During this time the machine will be cycling through some meaningless program if the G flip-flop happens to start out holding a "1".
2. Lock switch S in the off position. This sets G to 0, sets the operation counter to 000, and reads 000 into C. Read program and data into memory with "push-buttons" (R).
3. Release S (nothing happens while S is in neutral position), push it to the on position and release. This sets G to 1 and the operation counter steps into 111, the beginning of program timing. Since C holds 000, the first word to be taken out of memory and transferred into the A-register will come from the first memory register (at location 000). The program should be stored with this fact in mind.
4. The program should end with a halt command, 00xyz, where x y z may be any number. This freezes the contents of all FF's except for G and the F and C registers, all of which are cleared. A new program may now be read into storage via the R switches and the machine may be re-started by pushing the S switch over to on.
5. Read-out is via neon bulbs attached to the memory FF's (not shown above).

## 5.4 Block Diagrams

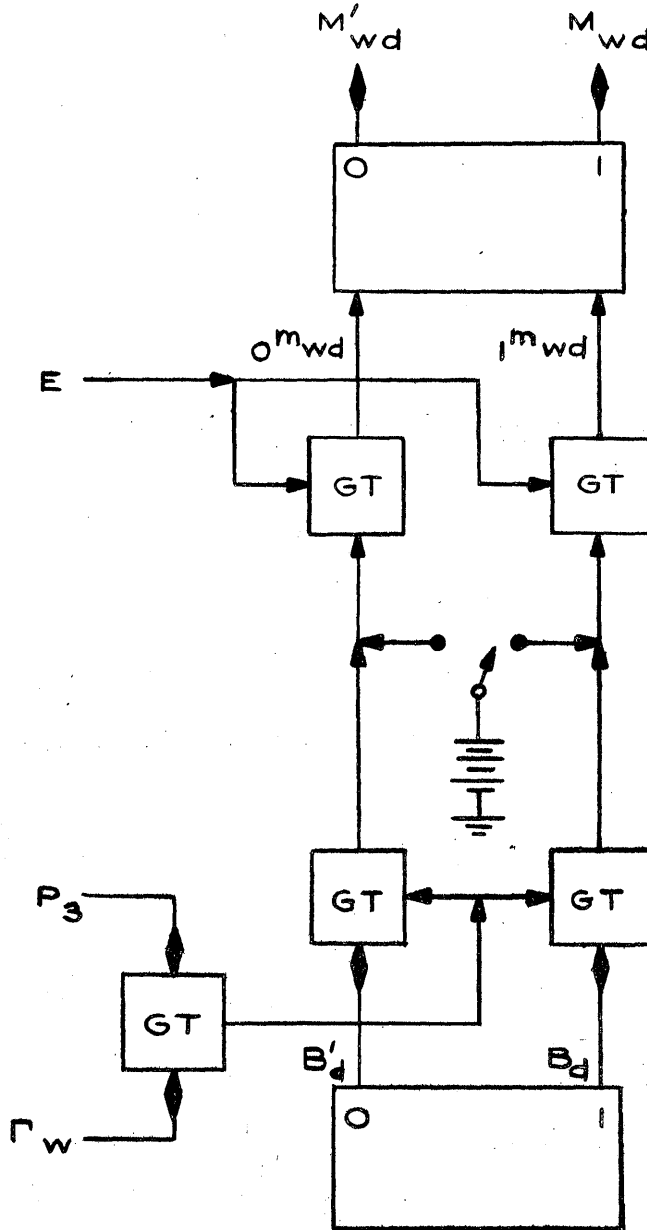
People who are unfamiliar with the algebraic notation may find it helpful to translate some of the equations into block diagrams. A little bit of this will go a long way toward promoting familiarity with the algebraic symbolism; with a little practice it will be found that as much information can be read directly from a few lines of equations as from a dense page of block diagrams.

Block diagrams for a few of the registers follow.

Note that the equations for a register show the inputs to that register. To find outputs of register X, look through all the equations for occurrences of the letter "X". (In the case of the F and C registers, look for "P" and "I" as well, since these abbreviate products of "F" and "C"s.)

**Memory**

$$\left. \begin{aligned} I_{wd}^n &= (R_{wd}' + B_d \Gamma_w P_3) E & w = 0, \dots, 7 \\ O_{wd}^m &= (R_{wd}^0 + B_d^0 \Gamma_w P_3) E & d = 0, \dots, 5 \end{aligned} \right\}$$



Identical picture  
for each memory cell.

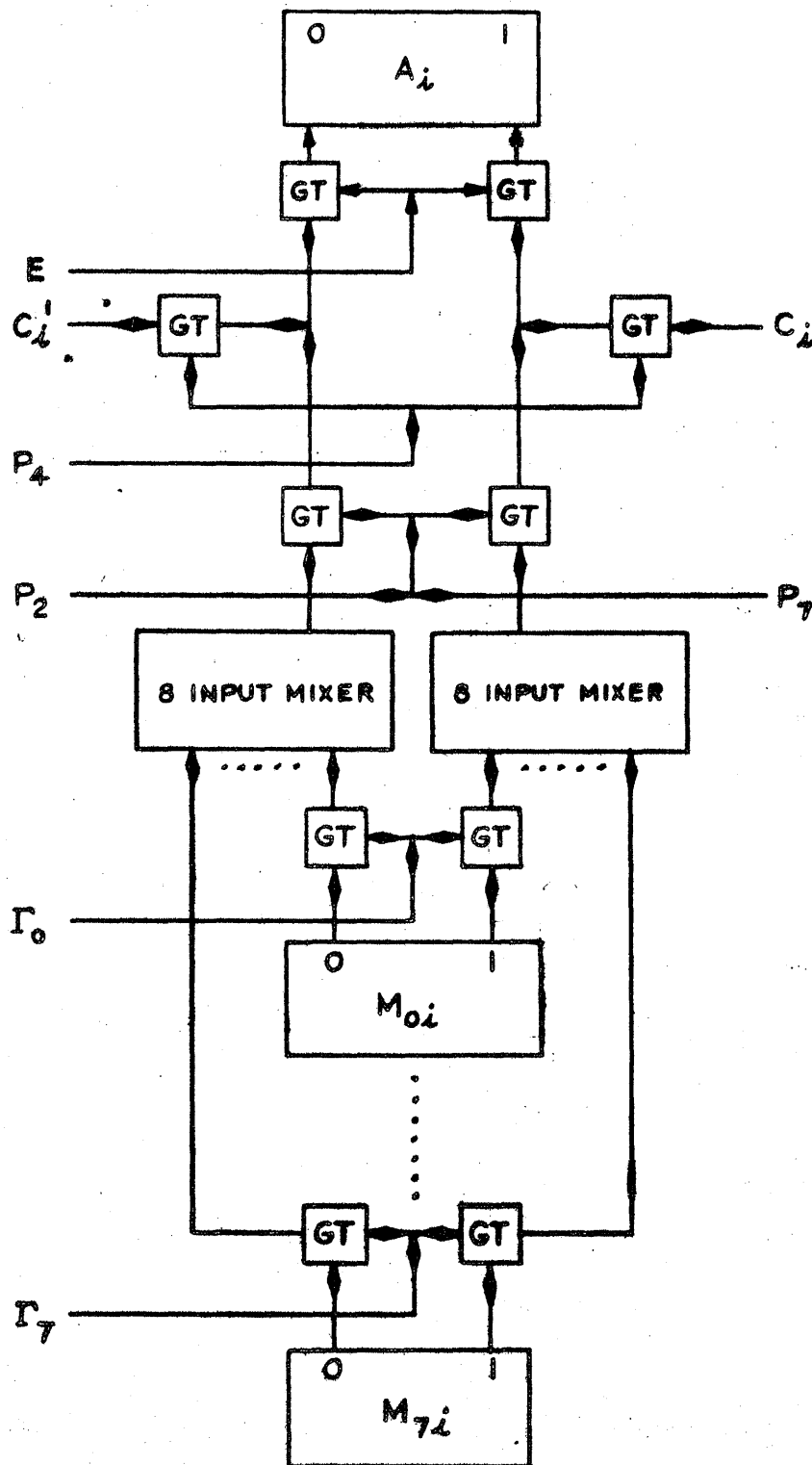
In case a magnetic core memory is used with an I/2 selection system, the read-in gates are simply points where pairs of insulated wires cross inside the cores.



The A-Register (last three digits)

$$1^{a_i} = \left[ C_i P_4 + (P_2 + P_7) \left( \sum_{h=0}^7 M_{hi} \Gamma_h \right) \right] B \quad i = 2, 3, 4;$$

$$0^{a_i} = \left[ C_i' P_4 + (P_2 + P_7) \left( \sum_{h=0}^7 M_{hi}' \Gamma_h \right) \right] B \quad h = 0, \dots, 7$$



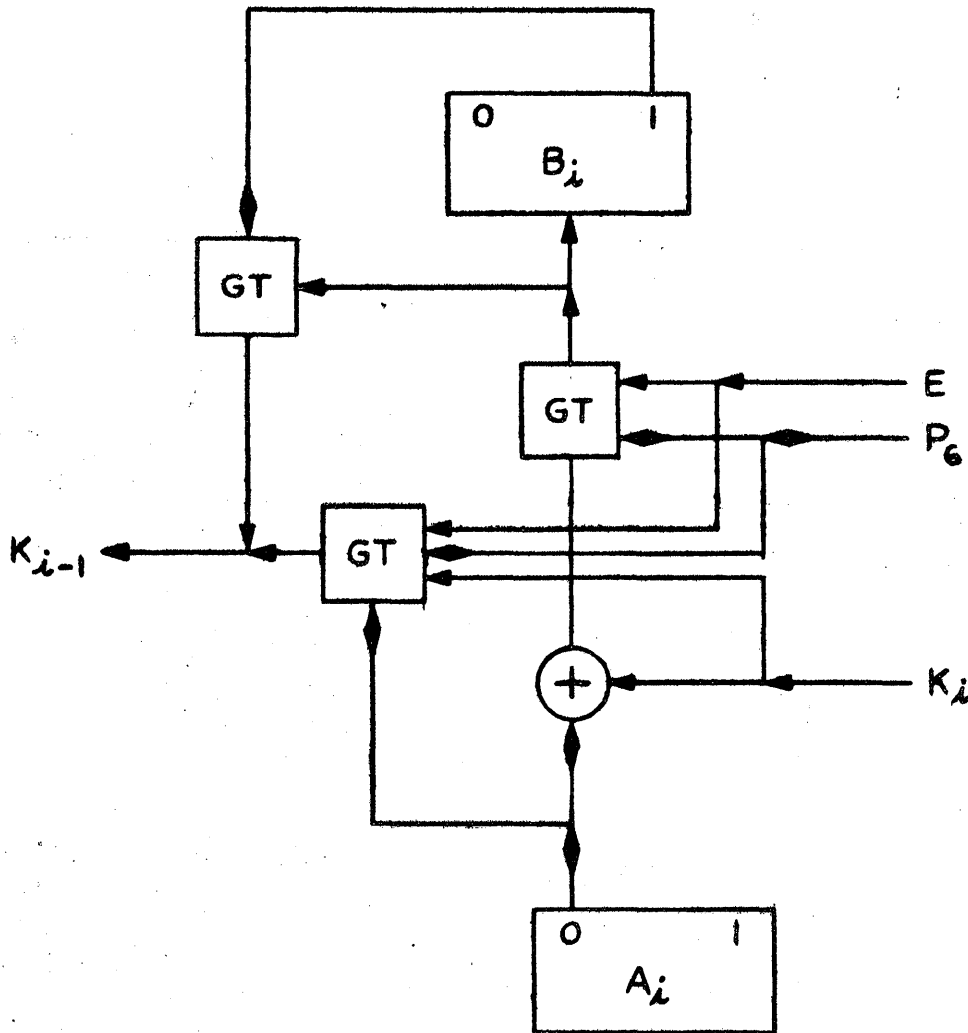
This memory system is not intended as a practical proposal. The following two modifications are desirable: (1) In reading out of memory, clear A on the preceding clock pulse and then read into only the set sides of the A FF's. This eliminates the left hand half of the memory read-out gates and the left hand 8 input mixer. (2) Use magnetic cores as memory cells. Then the remaining read-out gates are inherent in the cores, and the 8 input mixer is the sensing winding.

The Accumulator (complement inputs)

The set and clear inputs are used only for shifting. The reader can complete this diagram by adding them.

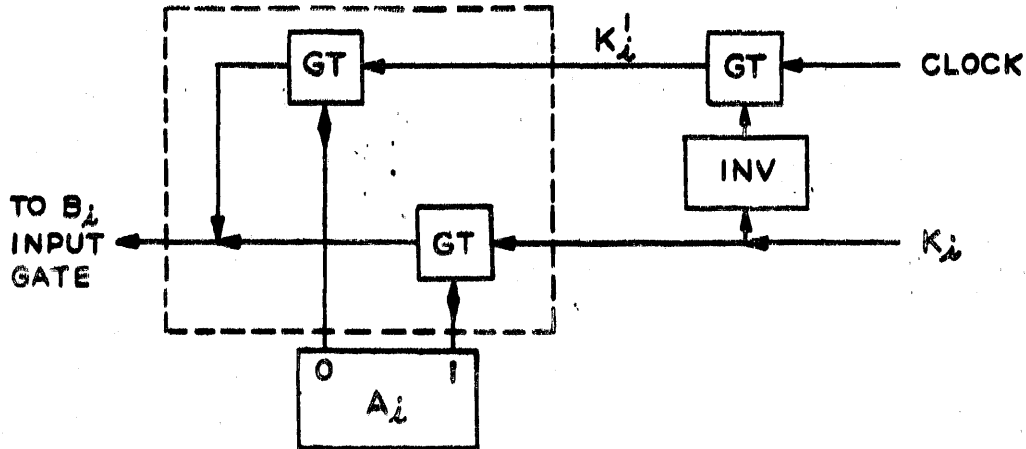
$$b_i = (A_i \oplus K_i) P_6 E \quad (i = 0, \dots, 4)$$

$$K_{i-1} = A_i K_i P_6 E + B_i b$$



represents a black box which gives a pulse output whenever one but not both of the inputs are present. See next page.

The black box for partial sum ( $\oplus$ ) is an oversimplification. Probably the best way to implement it with customary electronic components is to use not only  $A_i$  and  $K_i$  but also their complements:



This requires either providing an inverter to get  $K_i^1$  out of  $K_i$ , or building up  $K_{i-1}^1$  as an output from stage #i independent of  $K_i$ . (A carry zero line as well as a carry one line.)

SIGNED Richard C. Jeffrey  
Richard C. Jeffrey

Irving S. Reed  
Irving S. Reed

APPROVED [Signature]  
N. H. Taylor

RCJ/ISR/ep

- cc: G. R. Briggs
- D. R. Brown
- D. A. Buck
- H.R.J. Grosch
- W. A. Hosier
- J. Jacobs
- W. Linvill
- R. P. Mayer
- J. A. O'Brien
- W. Ogden
- K. H. Olsen
- W. N. Papian

BIBLIOGRAPHY

1. Jeffrey, R. C., Reed, I. S., "The Use of Boolean Algebra in Logical Design", Engineering Note E-458-1, Digital Computer Laboratory, M.I.T., (April, 1952).
2. Reed, I. S., Division II Status Report, Project Lincoln, (April, 1952).
3. Reed, I. S., "Some Mathematical Remarks on the Boolean Machine", Project Lincoln Technical Report No. 2, (December 19, 1951).