

Division 6 - Lincoln Laboratory  
Massachusetts Institute of Technology  
Lexington 73, Massachusetts

SUBJECT: ALGEBRAIC SYNTHESIS OF LOGICAL FEEDBACK NETS

To: John F. Jacobs

From: Richard C. Jeffrey

Date: October 6, 1954

Approved: *John F. Jacobs*  
John F. Jacobs

Abstract: Boolean algebra is extended by adding to the Boolean or "instantaneous" transformations (not, and, or, etc.) a delay transformation symbolized by " $\Delta$ ." The behavior of digital devices which contain feedback or delay (e.g., flip-flops) is described in the extended symbolism. Logical networks containing feedback and delay are correlated with sets of equations. An operator notation is introduced and used to simplify network equations. The technique is illustrated in the design of an accumulative binary adder which makes essential use of feedback and delay.

### 1. Introduction

We shall assume that the reader is familiar with the elements<sup>1</sup> of Boolean algebra and with its application to the design of logical nets<sup>1</sup>, and shall go on to describe a simple extension of that algebra which considerably widens its range of application. To clarify what we are doing we shall first survey the kinds of logical nets to see which of them are describable by Boolean algebra and which are not.

It will be simplest if at first we confine ourselves to clocked nets, that is, to nets in which time may be regarded as a sequence of distinct "moments",  $t = \dots, -1, 0, 1, 2, \dots$ . The history of any junction (input, output or intermediate point) in such a net for the period  $0 \leq t \leq n$  would be written by stating the value, 0 or 1, of the signal at that junction at  $t = 0, 1, 2, \dots, n$ . We shall call such histories "messages" and write them as strings of zeroes and ones between brackets. For example, (01101) will be the message:

- 
1. For introduction to Boolean algebra, see Serrell, "Elements of Boolean Algebra...", Proceedings of the I.R.E., Vol. 41, No. 10, pp. 1366-1380. The term "logical net" is borrowed from Burks and Wright, "Theory of Logical Nets", Proceedings of the I.R.E., Vol. 41, No. 10, pp. 1357-1365. The term applies to digital computers, to sections of digital computers, to relay nets, etc. Block diagrams (such as the ones in Fig. 2) and sets of equations are both representations of logical nets.

t	0	1	2	3	4
value	0	1	1	0	1

We shall use greek letters as variables for 0 and 1; latin letters will be variables for messages. Boolean functions will be represented by -("not"), &("and"), v("and/or") and ⊕(exclusive "or"). In Fig. 1 these functions are defined for 0 and 1 as arguments. Thus,  $\bar{0}$  is defined to be 1 and  $0 \& 1$  is defined to be 0; but expressions like " $\bar{(0110)}$ " and " $(0110) \& (1100)$ " are left undefined. We shall now remedy this situation.

Definition 1.  $\bar{x}$  or  $\bar{X}$  is the result of applying - to each component of x.  $x \& y$  is the result of applying & to the first component of x and the first component of y, then to the second components, and so on. Similarly for v and ⊕.

$$\text{Then } \bar{(0110)} = (\bar{0} \bar{1} \bar{1} \bar{0}) = (1001),$$

$$(0110) \& (1100) = (0 \& 1 \ 1 \& 1 \ 1 \& 0 \ 0 \& 0) = (0100),$$

$$(0110) v (1100) = (0 v 1 \ 1 v 1 \ 1 v 0 \ 0 v 0) = (1110),$$

and so on.

The Boolean transformations operate only on contemporary digits of messages. For example, v operates on the t = 0 digits of x and y to produce the t = 0 digit of xvy; the t = 1 digit of xvy comes from the t = 1 digit of x and the t = 1 digit of y, and so on. There is no transformation made up solely of Boolean functions which operates, say, on the t = i digit of one message and the t = i+1 digit of another to produce the t = i+2 digit of the result. Let us call a transformation (logically) "instantaneous" if the i<sup>th</sup> digit of the output message is always a function of the i<sup>th</sup> digit(s) of the input message(s). A temporal transformation will be defined as one in which the i<sup>th</sup> digit of the output message depends on an earlier digit of at least one of the input messages. It is the temporal transformations which involve "memory" or "storage".

Apparently the Boolean transformations are all instantaneous. This means that the applicability of Boolean algebra to the design of logical nets is restricted to instantaneous nets, that is, to nets which effect instantaneous transformations. (A net will be called "instantaneous" in the logical sense when its actual delay is small compared to the period of the clock, i.e., the interval between t = i and t = i+1.)

In order to extend the applicability of Boolean algebra to temporal nets, we shall add to it a delay transformation, Δ. Δx is the result of delaying each digit of x by one clock-period. For example:

t	...	-1	0	1	2	3	...
x	...	1	0	1	1	...	...
Δx	...	...	1	0	1	1	...

The rule is: to obtain  $\Delta x$ , shift  $x$  one place to the right.

A basic classification of logical nets is illustrated in Fig. 2. Note that there are no instantaneous, feedback nets. The reason for this restriction can be seen by considering the effect of shorting out the delay element in the feedback net at the lower right of Fig. 2. Suppose that the input is 1. What is the output? If the output is 0 then so is the feedback input; but  $1 \oplus 0 = 1$ , which contradicts the assumption that the output is 0. Then the output must be 1. But then the feedback input is 1, and  $1 \oplus 1 = 0$ , which contradicts the assumption that the output is 1. Thus if instantaneous feedback nets are admitted without restriction, contradictions arise.<sup>2</sup>

Both practically and theoretically, feedback nets are of great importance; since all feedback nets are temporal, they cannot be described by ordinary Boolean algebra. The rest of this paper will be devoted to methods for synthesizing feedback nets, using the algebra described below.

## 2. Binary Network Algebra

Let us adopt the convention that if  $x$  is a message and  $i$  is an integer then  $x(i)$  is the  $t = i$  digit of  $x$ . For example, if  $x$  is the message defined in the table above,  $x(0) = 0$  and  $\Delta x(0) = 1$ . Now we can define  $\Delta$  as follows.

Definition 2.  $(\Delta x)(t) = x(t - 1)$ .

Strictly speaking, definition 2 implies that we are thinking of the messages as extending indefinitely to the left and right. If this were not true and  $x$  were defined, say, only for non-negative values of  $t$ , then definition 2, with  $t = 0$ , would say " $(\Delta x)(0) = x(-1)$ ". This would leave  $\Delta x$  undefined for  $t = 0$ ;  $\Delta^2 x$  would be undefined for both  $t = 0$  and  $t = 1$ , and so on. In practice we shall solve this problem by specifying a few values of  $x$  to the left of  $t = 0$  when we expect to be dealing with  $\Delta x$ ,  $\Delta^2 x$ , etc.

The set of all binary messages is a Boolean algebra when  $\bar{x}$ ,  $x \& y$ , etc. are understood as in definition 1 above. The result of adding the operator  $\Delta$  to the system will be called "binary network algebra". In the binary network algebra,  $\Delta$  may be used with the Boolean operations to produce such messages as  $(\Delta x + \bar{y}) \& x$ . Further,  $\Delta$  may be applied repeatedly to produce  $\Delta^2 x (= \Delta(\Delta x))$ ,  $\Delta^3 x (= \Delta(\Delta^2 x))$ , etc. Transformations compounded out of  $\Delta$  and the Boolean transformations in any way will be called "network transformations" since they comprise exactly the transformations which can be implemented by logical nets.

$\Delta$  never transforms different messages into the same message;

<sup>2</sup> Instantaneous feedback nets are discussed in greater detail in Burks & Wright, op. cit.

if  $x \neq y$  then  $\Delta x \neq \Delta y$ . In other words,  $\Delta$  is a one-to-one transformation. This permits us to cancel " $\Delta$ " from both sides of " $\Delta x = \Delta y$ " to get a statement, " $x = y$ ", which is true if and only if the original statement is true. But the situation is different for network transformations generally. Transformations compounded out of  $\Delta$  and Boolean transformations are not in general one-to-one, and although we can always deduce " $f(x) = f(y)$ " from " $x = y$ ", we cannot go in the other direction. For example, let  $f(x) = \Delta x + x$ ,  $x = \dots 00000 \dots$  ( $x(t) = 0$  for all  $t$ ) and  $y = \dots 11111 \dots$  ( $y(t) = 1$  for all  $t$ ).  $x$  and  $y$  as defined above are not changed by being shifted:  $\Delta x = x$  and  $\Delta y = y$ . Then  $f(x) = \Delta x \oplus x = x \oplus x = \dots 00000 \dots$  and  $f(y) = \Delta y \oplus y = y \oplus y = \dots 00000 \dots$ . Here  $f(x) = f(y)$  but  $x \neq y$ . Summary: if  $f$  is any network transformation, " $x = y$ " implies " $f(x) = f(y)$ ", but the converse is not generally true. Network transformations are not in general one-to-one.

It should be noted that  $\Delta$  is an isomorphism of the binary network algebra with itself. This means that if  $f(x, y, \dots, z)$  is an expression built up by applying network operators to " $x$ ", " $y$ ",  $\dots$ , " $z$ ", then  $f(\Delta x, \Delta y, \dots, \Delta z) = \Delta f(x, y, \dots, z)$ :  $\Delta$  can be "factored out". For example,  $\Delta(\Delta x \vee \Delta^2 y) \& \Delta^2 z = \Delta(\Delta(\Delta x \vee y) \& \Delta z) = \Delta^2((\Delta x \vee y) \& z)$ .

A very important device for the application of the binary network algebra is the operator notation. For example, we shall write " $(\Delta^2 \oplus \Delta)x$ " for " $\Delta^2 x \oplus \Delta x$ ". In general, whenever a compound expression is built up by applying  $\Delta$  and Boolean functions to a single variable, the expression may be re-written in the form " $(---)x$ ", where  $---$  is an operator. This sometimes requires the use of the two new operators defined below.

Definition 3. The identity operator,  $I$ :  $Ix = x$ .

Definition 4. The complementation operator,  $N$ :  $Nx = \bar{x}$ .

For example:  $\Delta(\Delta \oplus I)x = (\Delta^2 \oplus \Delta)x = \Delta^2 x \oplus \Delta x$ ;  
 $((\Delta^2 \oplus N) \& I)x = (\Delta^2 \oplus N)x \& x = (\Delta^2 x \oplus \bar{x}) \& x$ .

It will sometimes be convenient to have names for special messages. In particular, the constants  $\underline{0} = \dots 000 \dots$  and  $\underline{1} = \dots 111 \dots$  will be useful.

Definition 5.  $\underline{0}$  is the message of which every component is 0: for all  $t$ ,  $\underline{0}(t) = 0$ .

Definition 6.  $\underline{1}$  is the message of which every component is 1: for all  $t$ ,  $\underline{1}(t) = 1$ .

The "vectors" " $\underline{0}$ " and " $\underline{1}$ " combine with latin letters in the same way that the "scalars" " $\underline{0}$ " and " $\underline{1}$ " combine with greek letters. Note that " $x(t)$ ", " $(\Delta x)(0)$ ", etc. count as scalars.

<u>Scalars</u>	<u>Vectors</u>
$0 \& a = 0$	$\underline{0} \& x = \underline{0}$
$0 \vee a = a$	$\underline{0} \vee x = x$
$0 \oplus a = a$	$\underline{0} \oplus x = x$
$\bar{0} = 1$	$\bar{\underline{0}} = \underline{1}$
$1 \& a = a$	$\underline{1} \& x = x$
etc	etc.

As will be seen in the next section, the operator notation is useful in suggesting simplifications of temporal transformations. In particular, the following theorem will be useful.

Theorem 1.  $\Delta^2 \circ I = (\Delta \circ I)^2$ .

Proof  $(\Delta^2 \circ I)x = \Delta^2 x \circ x$ .  $(\Delta \circ I)^2 x =$   
 $(\Delta \circ I) (\Delta \circ I)x = (\Delta \circ I) (\Delta x \circ x) =$   
 $\Delta(\Delta x \circ x) + I(\Delta x \circ x) = \Delta^2 x \circ \Delta x \circ \Delta x \circ x =$   
 $\Delta^2 x \circ \underline{0} \circ x = \Delta^2 x \circ x$ .

The following generalization of theorem 1 can be proved by induction.

Theorem 2. If n is a power of 2,  $\Delta^n + I = (\Delta + I)^n$ .

Note that the operation of "application", symbolized by juxtaposition of operators, obeys the associative and distributive laws, but not the commutative law. If  $O_1$  is an operator, " $O_1 O_2 x$ " is to be interpreted:  $O_1(O_2 x)$ . " $O_1 O_2 O_3 x$ " is  $O_1(O_2(O_3 x))$ , etc. But grouping does not matter:  $O_1(O_2 O_3) = (O_1 O_2) O_3 = O_1 O_2 O_3$ . Further, application is distributive on the right and left over  $\&$ ,  $\vee$ ,  $\circ$  and any other Boolean function of two variables. For example:  $O_1(O_2 \& O_3) = O_1 O_2 \& O_1 O_3$  and  $(O_1 \& O_2) O_3 = O_1 O_3 \& O_2 O_3$ . But application is not commutative:  $O_1 O_2$  need not be the same as  $O_2 O_1$ . For example, let  $O_1 = N$ ,  $O_2 = Iv\Delta$  and  $x = \dots 001001001 \dots$ . Then  $O_1 O_2 x = \dots 010010010 \dots$  while  $O_2 O_1 x = \dots 111111111 \dots$ .

### 3. Synthesis of a Feedback Net

Of all the temporal networks, feedback nets are the least accessible to ordinary Boolean techniques. As an example of the application of the binary network algebra to such nets we shall now synthesize a binary adder which makes use of feedback to minimize equipment. It will be seen that the operator notation is particularly helpful in suggesting simplifications.

Fig. 3 is a first statement of the problem. The equations shown there for  $K$  and  $\sigma$  might be realized directly by an instantaneous net such as the one shown in Fig. 4. It should be noted that we have no guarantee that physical devices corresponding to all the blocks in Fig. 4 are available. For example, it may be necessary to synthesize the  $\circ$  blocks out of  $\&$ ,  $\vee$  and  $\neg$  according to the identity,  $\alpha \circ \beta = (\alpha \& \beta) \vee (\alpha \& \bar{\beta})$ . The statement of a synthesis problem must include a list of available "black boxes" or physical operators.

For purposes of this example we shall confine ourselves to the physical operators shown in Fig. 5. These constitute a set adequate for constructing a large class of logical nets (most of the Whirlwind computer at M.I.T. uses just these packages). In Fig. 5, capital letters and solid

arrowheads represent "dc levels," while small letters and open arrowheads represent 0.4 $\mu$ sec. pulses. Thus Fig. 5 states some physical restriction on nets compounded out of these physical operators. For example, it rules out nets in which the (dc) output of a flip-flop is connected to the (pulse) input of a delay line. In the case of the gate tube where the inputs are electronically different from each other the dc input is distinguished by a blacked-in corner. The puzzling statement " $x\bar{y}y$ " in connection with the pulse mixer is a consequence of the requirement that both inputs may not be pulsed at the same time (to avoid jitter). In the case of pulses, we interpret "1" as the presence of a pulse and "0" as the absence of a pulse. Then the requirement that the two inputs may not be pulsed simultaneously is equivalent to eliminating the case in which both inputs are 1. But as Fig. 1 shows,  $\alpha = \beta = 1$  is the only case in which  $\alpha \odot \beta$  differs from  $\alpha \vee \beta$ . Therefore, in networks which obey our restriction, the action of the pulse mixer is equally well described by  $\odot$  or  $\vee$ . Once again, the restriction forbids us to construct networks in which both inputs to the pulse mixer are 1 at the same time.

Note that two sorts of equations are given for each physical operator, one of them in the notation of the binary network algebra and the other in functional notation e.g.,  $z = g(x, Y) = x\bar{y}$  for the gate-tube. The functional symbols refer specifically to the physical operators in Fig. 5. Thus, " $g(x, Y)$ " refers specifically to a pulse, the output of a gate tube one of whose inputs is a pulse and the other a dc level. " $g(x, y)$ " is not a meaningful expression, since the "g" requires a capital letter (dc level) as its second argument; but " $x\bar{y}$ " is an expression of the binary network algebra and is not tied down to any specific physical operator.  $x\bar{y}$  cannot be formed by a gate tube, but nevertheless it is a permissible expression.

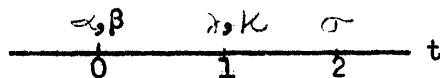
The flip-flop equation may be new to the reader. The physical operator involved is the Eccles-Jordan multivibrator with the two inputs tied together so that a pulse at the input, after a delay, complements the output. In fact, another output is available from the plate of the other tube, but this is not needed for our present purposes. In cases where it is used, it would be given a separate designation, say "Z", and would have the equation:

$$Z = F'(x, Y) = \bar{Y}$$

The correctness of the flip-flop equation can be seen by considering two cases: (1) If x is not pulsed at time t,  $x(t) = 0$  and " $Y = \Delta(x\bar{y})$ " or " $Y(t+1) = x(t) \odot Y(t)$ " becomes " $Y(t+1) = 0 \odot Y(t) = \bar{Y}(t)$ "; if x is pulsed at time t, the equation becomes " $Y(t+1) = 1\bar{y}(t) = \bar{Y}(t)$ ". Thus, if x is not pulsed, Y remains unchanged, and if x is pulsed, Y is complemented.

We can now state the synthesis problem more fully: using the physical operators of Fig. 5, we wish to design a logical net which satisfies the equations of Fig. 3. Note that the equations in Fig. 3 do not specify the relative times of occurrence of  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\sigma$  and  $\kappa$ . These may be logically simultaneous, as in the instantaneous adder (Fig. 4) or, as in the adder which will now be synthesized,  $\sigma$  and  $\kappa$  may occur some number of clock periods after  $\alpha$ ,  $\beta$ , or  $\gamma$ .

Now we shall investigate case illustrated below



where  $\alpha$  and  $\beta$  appear at  $t = 0$ , the carry input ( $\gamma$ ) and carry output ( $\kappa$ ) occur at  $t = 1$ , and  $\sigma$ , the sum-digit, does not appear until  $t = 2$ . Many other arrangements are possible; e.g., we might investigate the case where  $\alpha, \beta, \gamma$  and  $\kappa$  all appear at  $t = 0$  and  $\sigma$  occurs at  $t = 1$ . The reader should imagine that the synthesis we are about to undertake, using the timing illustrated above, is part of a survey of solutions to the problem under various assumptions as to the relative timing of signals. We shall not attempt to justify here the selection of the specific timing chosen; for present purposes, it is part of the data of the problem.

Now our adder, in context, is part of a digital computer and must satisfy some electronic requirements on the inputs and outputs. Suppose the adder's environment requires:  $\beta, \gamma$  and  $\kappa$  are to be 0.1 $\mu$ sec. pulses and  $\alpha$  and  $\sigma$  are to be d.c. levels; further,  $\alpha$  and  $\sigma$  are to time-share a single output terminal. These requirements are summed up in Fig. 6, where pulse terminals have open arrowheads and lower case roman letters as labels, and solid arrowheads and roman capitals indicate dc levels. Note that we require that terminals c, b and k be quiescent (no pulse = 0) except when  $\gamma, \beta$  and  $\kappa$  occur.

So far we have been stating the problem. We have now reached the point where the binary message algebra can be applied to the solution of the problem, and it will be found that, as usual, the problem statement has been more time consuming than its solution,

First, to find the transformation for the sum we must determine A as a binary message-function of A, b and c such that  $\sigma = \alpha \oplus \beta \oplus \gamma$  is satisfied. Such a transformation can easily be found by translating (delaying) A, b and c so as to align  $\alpha, \beta$  and  $\gamma$  with  $\sigma$ , as follows.

t =	0	1	2	3	4
A =	$\alpha$	?	$\sigma$		
$\Delta^2 A =$			$\alpha$	?	$\sigma$
$\Delta^2 b =$			$\beta$	0	0
$\Delta c =$		0	$\gamma$	0	

Now, as the  $t = 2$  column above testifies, the relation

$$A = \Delta^2 A \oplus \Delta^2 b \oplus \Delta c \tag{1}$$

implies  $\sigma = \alpha \oplus \beta \oplus \gamma$ , so that (1) is suitable as a transformation for the sum. But perhaps this transformation can be simplified. Let us try.

The condition can be written

$$\sigma = \alpha \oplus \beta \oplus \gamma \oplus \text{any number of zeroes,}$$

since in general,  $v \oplus 0 = v$ . Then it cannot hurt if we add  $\Delta b$  and  $\Delta^2 c$  to the right-hand side of (1), since both of these are 0 at  $t = 2$ .

t =	0	1	2	3	4
$\Delta b =$		$\beta$	0	0	
$\Delta^2 c =$			0	0	0

We now have:

$$A = \Delta^2 A \oplus \Delta^2 b \oplus \Delta c \oplus \Delta b \oplus \Delta^2 c$$

which may be rearranged using the laws " $x \oplus y = y \oplus x$ " and " $x = y \oplus x$  implies  $x \oplus y = z$ " to read:

$$\Delta^2 A \oplus A = \Delta^2 b \oplus \Delta b \oplus \Delta^2 c \oplus \Delta c \quad (2)$$

Now (2) is somewhat more regular-looking than (1) and may for that reason reduce to something simpler. This was the motivation for adding in  $\Delta b$  and  $\Delta^2 c$ . Using the operator notation, (2) may be rewritten:

$$\begin{aligned} (\Delta^2 \oplus I)A &= \Delta((\Delta \oplus I)b \oplus (\Delta \oplus I)c) \\ &= \Delta(\Delta \oplus I)(b \oplus c) \end{aligned}$$

or, since  $\Delta^2 \oplus I = (\Delta \oplus I)^2$ ,

$$(\Delta \oplus I)^2 A = \Delta(\Delta \oplus I)(b \oplus c) \quad (3)$$

So far, (3) says the same thing as (2), but with this difference: it suggests simplification by "cancelling"  $\Delta \oplus I$  from both sides to yield

$$(\Delta \oplus I)A = \Delta(b \oplus c) \quad (4)$$

or

$$\Delta A \oplus A = \Delta b \oplus \Delta c$$

or

$$A = \Delta(A \oplus b \oplus c) \quad (5)$$

which is much simpler than the transformation (1) with which we started.

But is the process which led from (1) to (5) legitimate? More specifically is the cancellation of " $\Delta \oplus I$ " from both sides of (3) allowable? Since  $\Delta \oplus I$  is not a one-to-one transformation (See Section 2) we may deduce  $(\Delta \oplus I)x = (\Delta \oplus I)y$  from  $x = y$  but not vice-versa. Apparently the cancellation of  $\Delta \oplus I$ , which led from (3) to (5) says not that (3) implies (5), but rather that (5) implies (3). But this is just what we want. If (5) implies (3), then any net which implements (5) must also implement (3).



More exactly, we already know that (3) together with the boundary conditions  $A(0) = \alpha$ ,  $b(0) = \beta$ , etc. is satisfied only by sets of messages  $A_i, b_i, c_i$  which have the property  $A_i(2) = \sigma = \alpha \oplus \beta \oplus \gamma$ . Since (5) implies (3), then all the sets of messages which satisfy (5) must satisfy (3) as well, and therefore have the property that  $A_i(2) = \sigma$ . Now the boundary conditions are:

t	0	1	2
A	$\alpha$		
b	$\beta$	0	0
c	0	$\gamma$	0

where  $A(1)$  and  $A(2)$  are left open and all values of A, b and c are left open for  $t < 0$  and  $t > 2$ . (5) can be stated:

$$A(t+1) = A(t) \oplus b(t) \oplus c(t) \tag{6}$$

Then

$$\begin{aligned} A(1) &= A(0) \oplus b(0) \oplus c(0) \\ &= \alpha \oplus \beta \oplus 0 = \alpha \oplus \beta \end{aligned} \tag{7}$$

$$\begin{aligned} A(2) &= A(1) \oplus b(1) \oplus c(1) \\ &= \alpha \oplus \beta \oplus 0 \oplus \gamma = \alpha \oplus \beta \oplus \gamma \end{aligned} \tag{8}$$

Then any net which effects transformation (5) will give the proper transformation for the sum. The process of finding such a net, using only the operators of Fig. 5 is shown in Fig. 7. We may represent  $b \oplus c$  by  $m(b,c)$  because  $b (= \dots\beta 0 0 \dots)$  and  $c (= \dots 0 \gamma 0 \dots)$  are never both 1 at the same time. The pulse output of the mixer is then applied to the input of the flip-flop.

Now we shall repeat this synthesis process on a compressed scale to find a net which generates the carry.

$$k = (\gamma \& (\alpha \oplus \beta)) \vee (\alpha \& \beta) \tag{9}$$

The inputs and outputs are shown below:

$$\begin{aligned} A &= (\alpha \quad \alpha \oplus \beta \quad \sigma) \\ b &= (\beta \quad 0 \quad 0) \\ c &= (0 \quad \gamma \quad 0) \\ \hline k &= (0 \quad k \quad 0) \end{aligned}$$

To force k to be  $k$  (as defined in (9) above) at  $t = 1$  we can use the transformations:

$$k = (\alpha \& A) \vee \Delta(A \& b) \tag{10}$$

This transformation can be implemented as follows:

$$k = m(g(c,A), d(g(b,A))) \tag{11}$$

Note that in each case, one input to the gate tube is a level and the other is a pulse, as required. The input to the delay line is the (pulse) output of a gate tube. We must verify that the two (pulse) inputs to the mixer are never both 1. The inputs are:

t =	0	1	2	3
$g(c,A) = \alpha A$	0	$(\alpha \oplus \beta) \& \gamma$	0	
$d(g(b,A)) = \Delta(A \& b)$		$\alpha \& \beta$	0	0

Now except at  $t = 1$ , there is apparently always one zero in the two rows, within the range ( $0 \leq t \leq 2$ ) of our interest. At  $t = 1$ , this seems not to be the case, but a closer examination shows that  $(\alpha \oplus \beta) \& \gamma$  and  $\alpha \& \beta$  cannot both be 1 (since  $\alpha \& \beta = 1$  only when  $\alpha = \beta = 1$ ; but with  $\alpha = \beta = 1$ ,  $\alpha \oplus \beta = 0$  and therefore  $(\alpha \oplus \beta) \& \gamma = 0$ ).

The net of Fig. 8 combines the nets for sum and carry.

4. Conclusion

The binary network algebra bears the same relation to logical nets generally that Boolean algebra has to instantaneous nets. In the synthesis of instantaneous nets by Boolean algebra, cost is minimized by minimizing the number of occurrences of Boolean operators in the corresponding equations. In the synthesis of temporal nets by the binary network algebra, occurrences of " $\Delta$ " must be minimized as well.

The derivation of equation (5) from equation (1) in section 3 above is typical of one of the most important cases of minimization. Consider the series of equations

$$A = \Delta a + \Delta A \tag{12.1}$$

$$A = \Delta a + \Delta^2 a + \Delta^2 A \tag{12.2}$$

$$A = \Delta a + \Delta^2 a + \Delta^3 a + \Delta^3 A \tag{12.3}$$

.....

$$A = \Delta a + \dots + \Delta^n a + \Delta^n A \tag{12.t}$$

where each equation is gotten from its predecessor by substituting " $\Delta a + \Delta A$ " for " $A$ ", as permitted by equation (12.1). Let us call the highest exponent of  $\Delta$  which occurs in an equation the "degree" of that equation. The successive equations (12) have degrees 1,2,3,...,t and in the sequence each equation implies all equations of higher degree. The converse is not true; for example,

$$A = \overline{\Delta a \oplus \Delta A} \tag{13}$$

implies (12.2) but is not equivalent to (12.1). Now by lowering the degree of an equation we simplify the corresponding net, and in a synthesis problem we may replace an equation by any other equation which implies it.

This last statement requires some explanation. A network equation  $E$  determines the output of the corresponding net as one of a certain set  $S$  of messages. If  $e$  implies  $E$ ,  $e$  determines the output as one of a set  $s$  of messages, where  $s$  is included in  $S$ . It may be that  $e'$  also implies  $E$ , where  $e'$  determines the output as a set  $s'$  different from  $s$  but also included in  $S$ . This is the case when, for example, both (12.1) and (13) imply (12.2). If  $a(0) = a(1) = A(1) = 0$ , (12.1) determines  $A(2)$  as 1 while (13) determines  $A(2)$  as 0. But (12.2) is neutral on the subject: it determines  $A(2)$  as equal to  $A(0)$ , which may be 0 or 1 depending on the value of  $A$  at  $t = 0$ . Thus, (12.2) is weaker or vaguer than either (12.1) or (13). Confining our attention to  $t = 0, 1, 2$  we find that (12.2) is satisfied by messages 000 and 101 equally. Replacing (12.2) by (12.1) selects 000 as the actual message; replacing it by (13) selects 101 as the actual message. But if (12.2) is the sole problem-statement for the sub-net under consideration we have been given freedom to choose either 000 or 101 as the actual message.

Further development of the logical network algebra should provide stronger techniques for lowering the degree of equations such as (12.2) and (12.3). In addition it is desirable to know when we have reached an irreducible equation, that is, an equation whose degree cannot be lowered. (12.1) and (13) are examples of irreducible equations: the occurrences of  $\Delta$  in them cannot be eliminated (this would give an instantaneous net). In addition there are equations such as  $A = a + \Delta^2 A$  of degree higher than 1 which are irreducible. A technique for recognizing irreducibility would save effort wasted on attempts to lower the degree of such equations by trial and error.

  
Richard C. Jeffrey

RCJ/djb

Distribution List

P.R. Bagley	F.E. Heart	B.E. Morriss
R.D. Buzzard	W.A. Hosier	J.A. O'Brien
W.A. Clark	J.F. Jacobs	I.S. Reed
G.P. Dinneen	J. Ishihara	H.K. Rising
M.A. Epstein	L.R. Jeffery	C.J. Schultz
B.G. Farley	T.A. Kalin	N.H. Taylor
M.D. Feldstein	W.K. Linvill	F.A. Webster
M.J. Frazier	R.P. Mayer	C.A. Zrackett

Drawings:

A-60420,	A-60422	A-59225	A-59224
A-60421	A-60423	A-59228	A-59226

$a$	$\neg a$ (or $\bar{a}$ )
0	1
1	0

$a$	$\beta$	$a \& \beta$	$a \vee \beta$	$a \oplus \beta$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

FIG. 1

## BOOLEAN FUNCTIONS

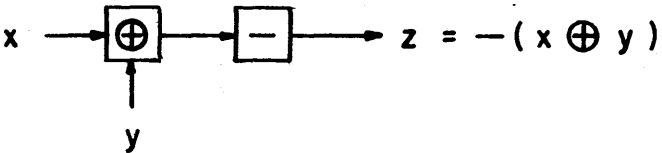
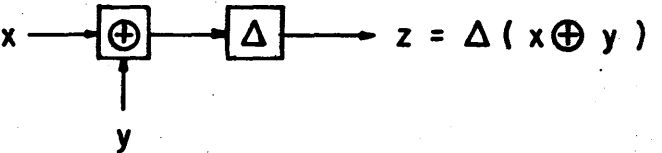
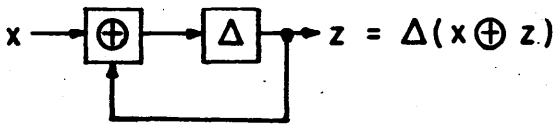
EXAMPLES	CASCADE	FEEDBACK
INSTANTANEOUS		NONE
TEMPORAL		

FIG. 2

CLASSIFICATION OF LOGICAL NETS

$a$	$\beta$	$\gamma$	$\kappa$	$\sigma$
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

$a$  = ADDEND  
 $\beta$  = AUGEND  
 $\gamma$  = CARRY IN  
 $\kappa$  = CARRY OUT  
 $\sigma$  = SUM

---


$$\kappa = [\gamma \& (a \oplus \beta)] \vee (a \& \beta)$$

$$\sigma = a \oplus \beta \oplus \gamma$$

FIG. 3

BOOLEAN EQUATIONS FOR BINARY ADDER

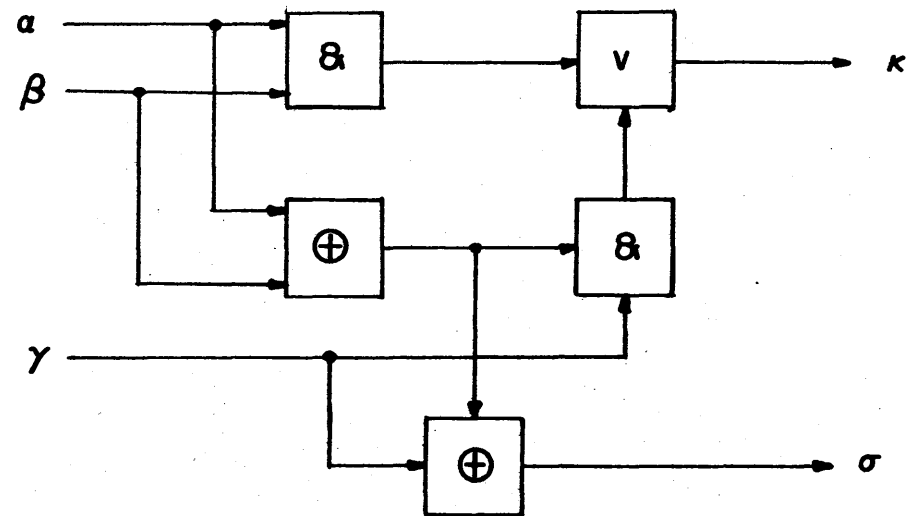


FIG. 4

INSTANTANEOUS BINARY ADDER

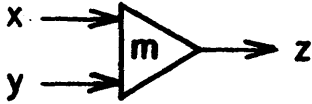
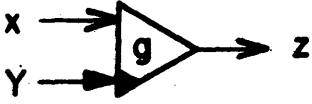
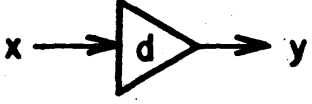

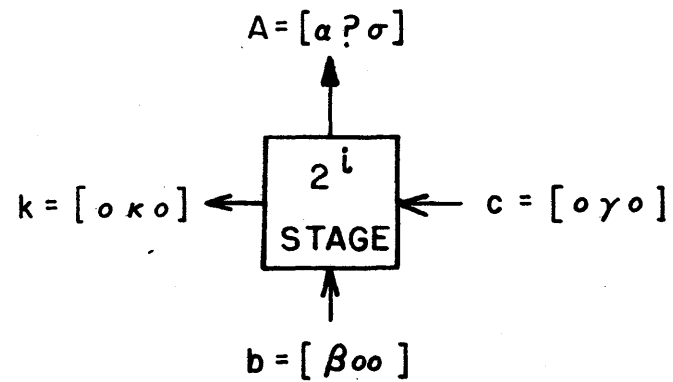
NAME	BLOCK DIAGRAM	EQUATIONS
PULSE MIXER		$z = m(x, y)$ $= x \vee y$ $= x \oplus y$
GATE TUBE		$z = g(x, Y)$ $= x \& Y$
DELAY LINE		$y = d(x)$ $= \Delta x$
FLIP-FLOP		$Y = F(x, Y)$ $= \Delta(x \oplus Y)$

FIG. 5

## PHYSICAL OPERATORS



A-59228



$$\text{SUM: } \sigma = a \oplus \beta \oplus \gamma$$

$$\text{CARRY: } \kappa = [\gamma \& (a \oplus \beta)] \vee (a \& \beta)$$

FIG. 6

PROBLEM STATEMENT

A-59224

$$A = \Delta ( A \oplus \underbrace{b \oplus c}_{m(b,c)} )$$
$$F ( A , m(b,c) )$$

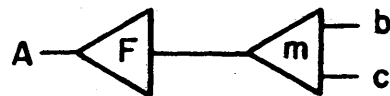


FIG. 7

FINDING A NET FOR THE SUM

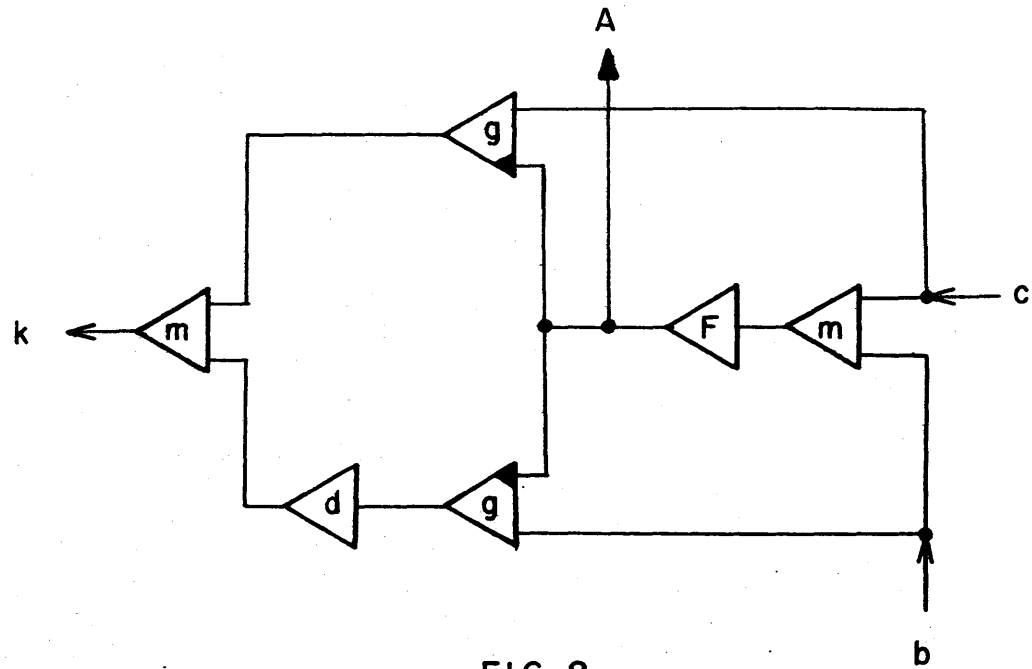


FIG. 8

SINGLE STAGE OF  
ADDER - ACCUMULATOR