May 10, 1957

SIGMA FILE MEMO NO. 1

SUBJECT: Results of a Preliminary Investigation of the Control of D.C. Asynchronous Logic.

Purpose:

The purpose of this memo is to discuss some of the methods of control used in d.c. asynchronous logic.

Basic Considerations:

All serial machine logic has what might be called a chain effect. Figure 1 shows a series of registers A, B, C, D, E, etc. Information is moved from A to B, and C to D concurrently (phase 1) and from B to C and D to E etc. concurrently (phase 2). The rate at which information emerges from E is the total time of phase 1 and phase 2. The time for a particular piece of information to proceed through the chain is the sum of all the phases required to move it. Considering just the delay due to the logical operations between triggers it is clear that splitting the operations in half with additional registers will double the rate of information transfer since the transfer between any two triggers is one half as long. Also it is seen that one operation can not be completed in less than 2 phases. For instance, if an adder was placed between A and B, the addition would not be complete until the sum was transferred to C.

The control of the information flow can require as much or more time than the logical operation itself. An optimum is accomplished by performing the control functions concurrently with the logical operations and making the time required by the control equal that required by the logical operation.

Control:

The flow of information through d.c. asynchronous logic is controlled by the concurrence of selected conditions instead of a pulse as used in synchronous logic. Reliable, fast, and economical methods of producing and detecting these conditions is the major control problem. Completion of a logical operation can be detected by:

1. Some invariable characteristic of the logic.

2. Inverting the operation to reproduce the original data.

3. Performing the operation in parallel.

The first method is the most attractive but the hardest to obtain, since every point in the logic is dependent on the input for its status and not on the completion of the operation. The third method is the next best and the most practical since we know it can be done and it requires less time than the method of inverting the operation. When using the third method, the completion of an operation is detected by a comparison of the results of the two parallel operations. The comparison is taken between the contents of the register following the logical operation and the results of the parallel checking logic. This insures that not only the results are correct but that they are properly set in the register, therefore the compare condition equal allows the gates leading to the register to be closed.

A sample system is shown in figure 2. This is a two register system including two logical operations and the necessary controls. The more general controls of the computer work through the local controls. The control portion, which is basically a logical ring using triggers B and C, receives signals from compares 1, 2, 3, and 4 and responds by opening or closing one of the two gates. At the time information arrives from a previous source gate 1 is closed. Gate 1 is opened by the ring condition B C which is set upon the concurrence of the following conditions:

1. The ring formed by triggers B and C is in condition $B \bar{C}$
2. Compare 1 is not equal $(\bar{E}_1)$
3. Compare 3 is equal (E_3)
4. The previous system is properly set (E_x).

These conditions assure us of the following respectively:

1. Previous control steps have been completed.
2. When gate 1 is opened, there will not be a false indication from compare 1 that the logical operation is complete.
3. Gate 2 is closed.
4. Source of information is properly set.

The opening of gate 1 sets Register 2, commences a comparison with its parallel logic on compare 1, and allows the information modified by logical operation 1 to proceed through logical operation 2 to gate 2 which at that time is closed. Upon concurrence of the following conditions the ring to set to condition $\bar{B} C$ which closes gate 1.

1. Compare 1 is equal (E_1), which indicates that logical operation 1 is complete.
2. Compare 4 is not equal $(\bar{E}_4)$. This is required to prevent getting a false equal when we close gate 1.
3. The control ring is in condition B C.

Gate 2 is then opened upon concurrence of:

1. E_4
2. $\overline{E}_2$
3. Ring is in condition $\overline{B}$ C.

It is opened by setting the ring to condition $\overline{B}$ $\overline{C}$.

The sequence of conditions table in Figure 2 shows the conditions at each step, and under the heading Necessary Conditions For Change In Ring lists those conditions which are a minimum for uniquely describing the system condition. These conditions are the ones used for stepping the ring.

If we define a phase to be the time from setting one register to entering the next, we can see that a phase requires the time to set a register, take two comparisons, open a gate, close a gate, and advance through two ring positions. Times now available indicate that the speeds expected of the basic circuits are approximately 20×10^{-9} for AND, OR and TRIGGER circuits. Using these figures the time required to complete two phases as illustrated in figure 2 is given by

$$T = 2T_R + 2T_G + 4T_C + 4T_T + 6T_C$$

$2T_R$	-	Time to set 2 registers	40 musec
$2T_G$	-	Time to open and close 2 gates (2 levels each)	160 musec
$4T_C$	-	Time to make 4 compares (3 levels each)	240 musec
$4T_N$	-	4 control trigger operation	80 musec
$8T_L$	-	6 control AND circuits	120 musec
		T =	640 musec

The time for the logic operations is not included since it is done concurrently with some of the control operations. The speed of the basic two register systems can be greatly improved by establishing tolerances on the relative speeds of the basic circuits. In other words the system of Figure 2 is intended to provide reliable operation despite an infinite relative variation in the speed of the basic circuits. Tolerances of $\pm$ 30% about some average value would allow a much simpler and much faster control system. For instance, if the average speed were 15 millimicroseconds, with such tolerances a speed of

300 millimicroseconds per local operation would not be too difficult to obtain. In addition all of the equipment in the dotted lines marked CONTROL would be eliminated along with compare 3 and compare 4.

Speed can also be improved by making the control system operate more gates concurrently. A system which uses this idea is shown in Figure 4. It is faster since the adder is used each phase and the accumulation is shifted between two registers. The added equipment is another register and more gates. The control of the gates would be done in a manner similar to figure 2. The time sequencing of the gates is shown in the sequence chart. The second adder is necessary to provide the compare which indicates the completion of the addition process. Opening the A gates allows the contents of Register 2 to be added to ACC 2 and the sum is developed in ACC 1. The compare equal of the sums in ACC 1 and the check adder indicate the completion of the addition. The A gates are closed, the B gates are opened, and Register 1 is added to ACC 1 and the sum is developed in ACC 2.

Thus by shifting our accumulator there is no need for a second phase in which to transfer its contents.

<u>Conclusions:</u>

At the present state of development the most severe limitation on the speed of d.c. asynchronous logic is the control functions. Reliable systems provide information rates in the order of 600 millimicroseconds per unit of information. Large speed improvements can be made by:

1. Adapting logical systems similar to that shown in figure 3.

2. Establishing tolerances on the speed of the basic circuits.

3. Design the logic such that a distinctive condition is provided by the completion of a phase of logic. (This would also eliminate parallel equipment and compare circuits)

4. Use semi-analog methods such as slow transistors in the longest path through the logic. (This is really a combination of 2 and 3 above.)

5. Either select components as to speed and use faster ones in the control circuits or use a faster type of transistor in the control circuits.

E. I. Jordan

E. I. Jordan
Engineering Planning

EIJ:jlc

Distribution list:

Dr. G. A. Blaauw ✓
Mr. W. Brooks
Dr. W. Buchholz
Mr. R. A. Gregory
Mr. J. E. Griffith
Mr. K. W. Kaeli
Mr. S. Pitkowsky
Mr. R. Preiss
Mr. D. Sweeney
Mr. W. Winger
Mr. W. Wolensky

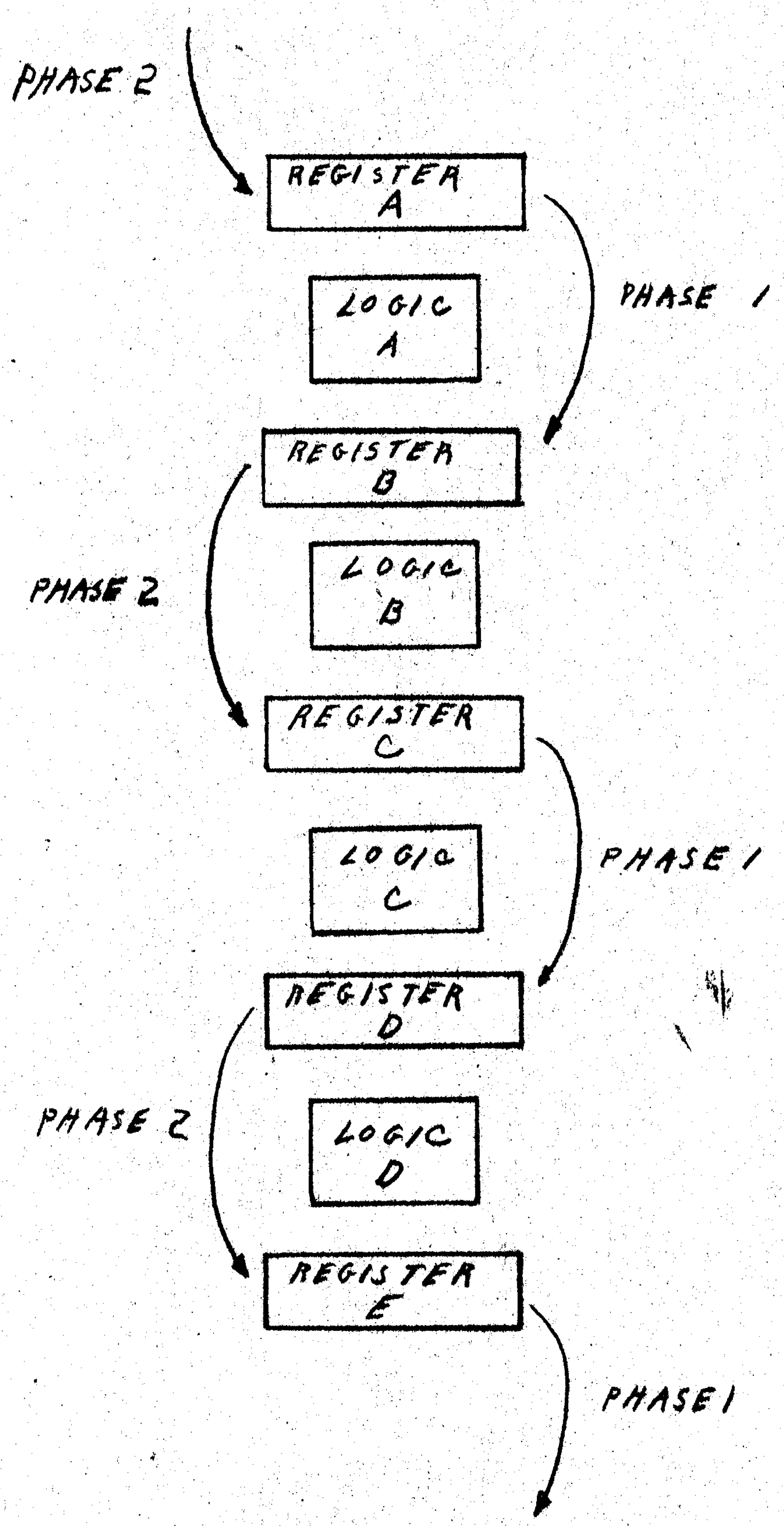
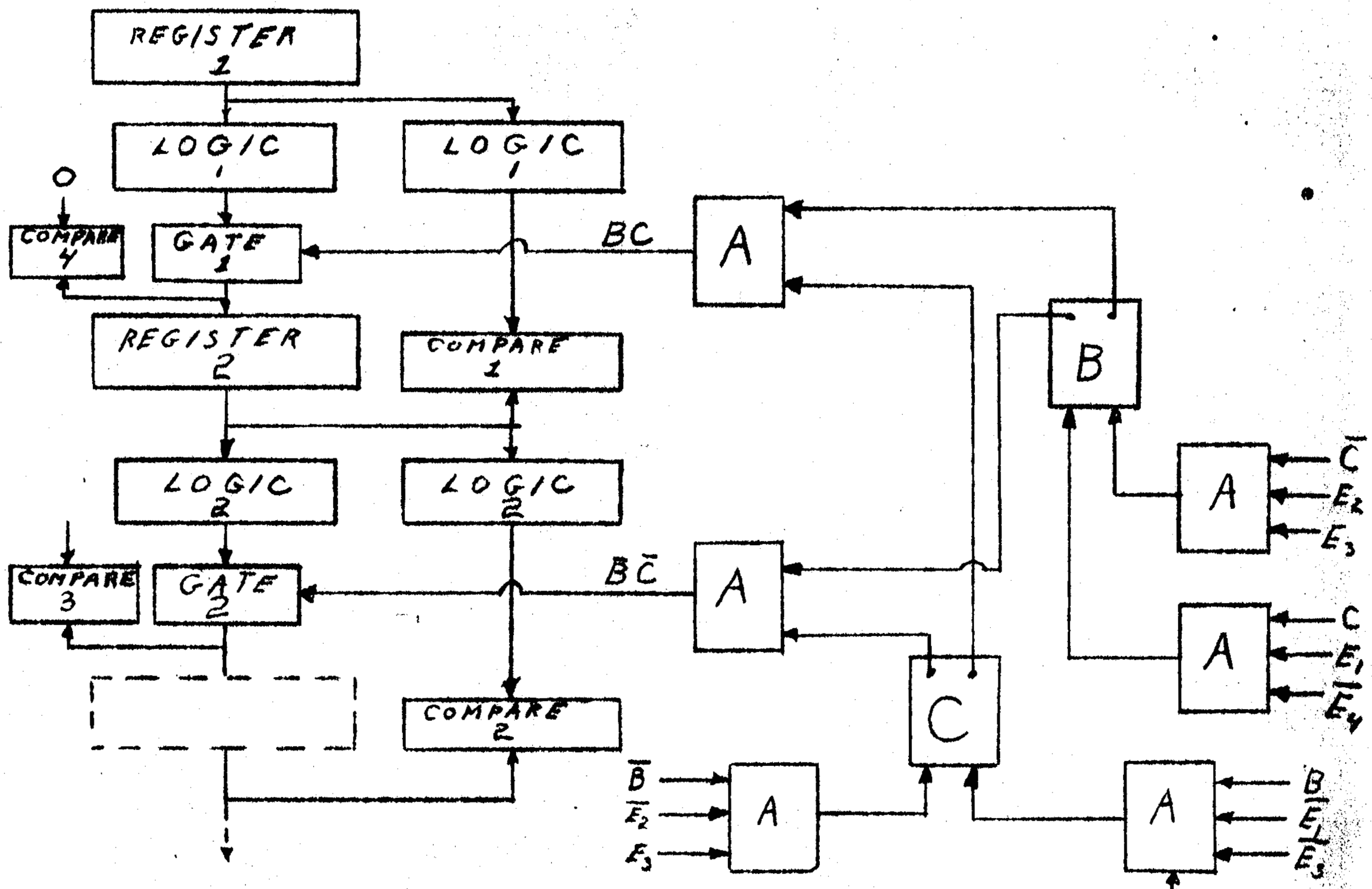

PHASE 2
REGISTER A
LOGIC A
PHASE 1
REGISTER B
PHASE 2
LOGIC B
REGISTER C
LOGIC C
PHASE 1
REGISTER D
PHASE 2
LOGIC D
REGISTER E
PHASE 1

FIGURE 1

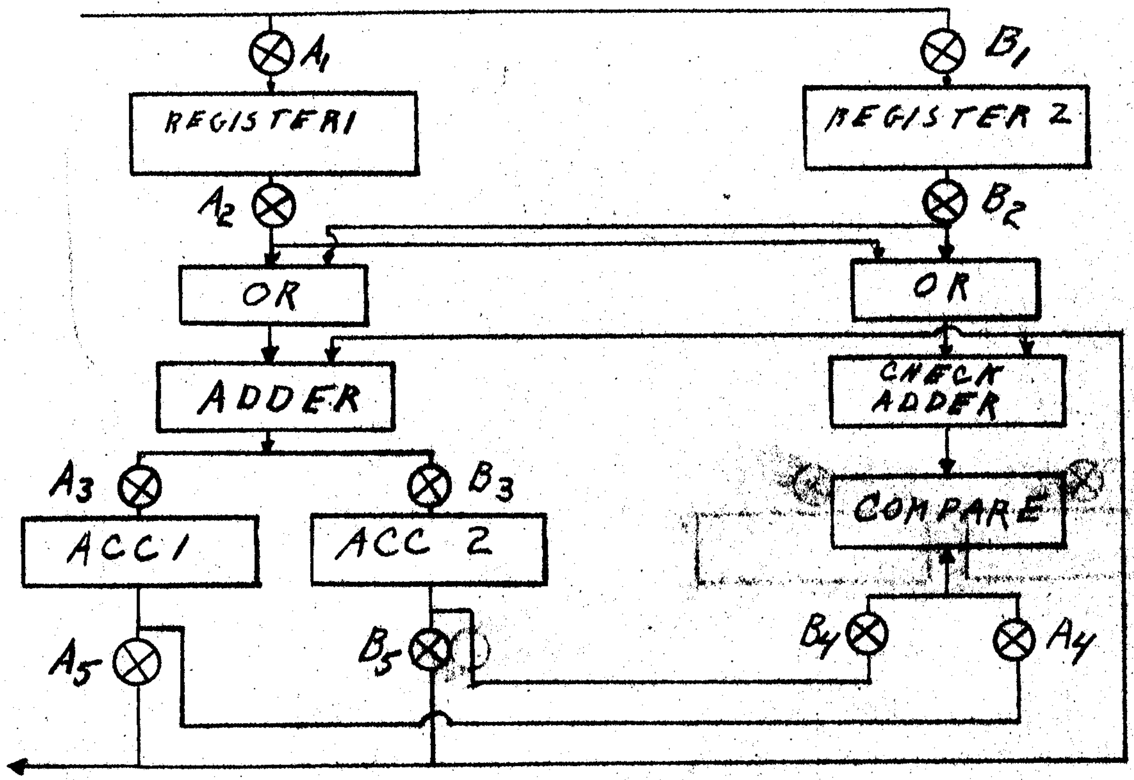


SEQUENCE OF CONDITIONS

	Condition	Necessary Conditions for change in ring	Ring changes to
Information arrives from previous operation. Gate 2 open Gate 1 closed	$\bar{C}\ \bar{B}\ \bar{E}_2\ E_1\ \bar{E}_3\ E_4$		
Compare 2 becomes equal.	$\bar{C}\ \bar{B}\ E_2\ E_1\ \bar{E}_3\ E_4$	$\bar{C}\ E_2\ E_3$	$B\ \bar{C}$
Gate 2 closes. Compare 3 becomes equal. Compare 1 must be unequal. E_x arrives indicating previous operation done	$\bar{C}\ B\ E_2\ \bar{E}_1\ E_3\ E_4$	$B\ \bar{E}_1\ \bar{E}_3\ E_x$	$B\ C$
Gate 1 opens. Compare 4 becomes not equal	$C\ B\ E_2\ \bar{E}_1\ E_3\ \bar{E}_4$		
Compare 1 becomes equal	$C\ B\ E_2\ E_1\ E_3\ \bar{E}_4$	$C\ E_1\ \bar{E}_4$	$\bar{B}\ C$
Gate 1 closes. Compare 4 becomes equal. Compare 2 must be unequal	$\bar{B}\ C\ \bar{E}_2\ E_1\ E_3\ E_4$	$\bar{B}\ \bar{E}_2\ E_3$	$\bar{B}\ \bar{C}$
Gate 2 opens. Compare 3 becomes not equal	$\bar{B}\ \bar{C}\ \bar{E}_2\ E_1\ \bar{E}_3\ E_4$		

Repeat for next operation.

FIGURE 2



SEQUENCE OF CONDITIONS.

Condition	Condition of Gates	
	Open	Closed
Addend arrives from memory to register 1	A_1 A_2 A_3 A_4 A_5	B_1 B_2 B_3 B_4 B_5
Addition of accumulator (Acc) 2 to register 1, sum developed in accumulator 1	A_1 A_2 A_3 A_4 A_5	B_1 B_2 B_3 B_4 B_5
* Compare indicates equal. A gates close, B gates open		
Addition of Accumulator 1 to Register 2, sum developed in Accumulators	B_1 B_2 B_3 B_4 B_5	A_1 A_2 A_3 A_4 A_5
* Compare indicates equal B bates close, A gates open REPEAT		

* Not all at control functions specified

FIGURE 3