COMPANY CONFIDENTIAL

[illegible]

November 12, 1956

PROJECT STRETCH

FILE MEMO NO. 48

SUBJECT: A Procedure for the Stretch Automation of Design

BY: L. S. Snyder

DATE: October 15, 1956

In the six months of this group's existence, two simultaneous experiments have been tried. One was the use of a table to represent the 705 Card Reader control circuits and the conversion of this table into Boolean Algebra expressions. The other was to take these expressions and manipulate them logically so as to simplify them and try to improve or test the actual machine logic. A more detailed explanation is given in Stretch Memo No. 39F. There are several lessons that were learned from these experiments. The first experiment proved that Boolean expressions can be formed from a tabular representation of circuits. However, it also showed the inadequacy of the table used and indicated how it should be expanded. It also showed the advantages of using a card format with which Keypunch operators were familiar. The second experiment ran into much difficulty. We have not discovered any successful solution to the problem of finding a minimum factored equation of a Boolean expression. Also, we have not been able to discover a good way of correlating two different expressions which have many of their inputs in common, so as to indicate a common use of circuits. A small group will concentrate on Logic in case there are further developments in this area. On the basis of this experiment the main group will concentrate on a 704 program to do the routine engineering and bookkeeping associated with a machine design effort.

The main burden on a designer using any type of program aid is the translation of his data into a format acceptable to the machine. In order to minimize this burden the designers who are specifying the logic should have to do as little semantic conversion as possible. A program format which used names of lines as the data for logical codes and which specified a name of the resulting line has been considered. It should be easy to build up a computer from this program using circuit criteria to change the logic to actual circuits. A set of codes would be set up to include all basic logical connectives such as And, Or, Exclusive Or, etc. But beyond this, more sophisticated operation codes such as Full Adder, Half Adder, Repeat, etc. could be added. For an example, see Appendix. If a card format similar to the 704 UASAP Assembly Program format is used, we would be able to use much of this already debugged program for our input program. This type of input can be easily explained to a logical designer and should require little translation from rough diagrams or charts.

The next requirement for an overall system is an adequate method of representing the STRETCH circuits within the 704. We are going to sue a tabular representation with a one-to-one correspondence between entries in the table and circuits of the machine. Each entry will be the equivalent of a logical block in a systems diagram. The proposed entry will consist of eight 704 words. A half word is being allocated for circuit specifications, a half word is for timing considerations and four and one half words will be used for input and/or output addresses. A half word is to be used for extending the table entry in case of need. Two words are being reserved for the physical location of the circuit and its output lines. See Figure I. This table should include enough information to be used as data for further programs, such as pluggable unit layout and back panel wiring. The extension address will permit extending table entries in case of need. An educated guess has been made as to the number of circuits which will be required. Seventeen bits are being used as addresses for the tabulated information. Three bits will specify a tape unit. Five bits will specify the number of the record within the unit. Eight bits will specify the circuit within a record. The remaining bit will be used to indicate whether this is the 0 or 1 output line of the specified circuit. The present proposal presumes that records will be of drum length, i.e., 2048 words, and all manipulation will be done between main memory and a drum so as to increase the speed of operation. This tabular system is a flexible one which will permit an extension in case of need and allows for the addition of data for future programs.

The input data in program form must be interpreted by the computer so that the alphanumeric information can be manipulated by the computer as meaningful information. The logical chain which is specified by the program will be identified by the mneumonic code specified on the program sheet. This code will be associated with the address of the last table entry in its logic formation. An index will be made so that any time the name, i.e., mne/monic code, of any particular circuit is referred to, the 704 can look up the pertinent table entry and get whatever information. is required. This will permit the designers to use easily remembered code names and at the same time the computer can arrange its data in some coherent system.

The logical circuitry which will be specified by the table entries is deduced from the operation code in the program format. The computer will create table entries corresponding to the circuits needed to perform these operations. The program will then search the existing tables for redundant circuits. It may also factor multiway And and Or circuits in order to perform simple circuit minimization. If any circuit is to drive more than the maximum allowed by circuit criteria, the program can assign table entries to the required powering circuits. It may be able to determine whether fast transistors are necessary and use these only when needed. Any other formulated criteria should also be applied by the program and not the designer. Thus a designer may specify logic without bothering to translate this into circuitry with the assurance that some minimal correctly designed machine will result.

Another major but routine procedure in the design procedure is processing changes. With an EDPM system, the process of changing the recorded logic can consist of a simple add and subtract process in which individual circuits or lines are specified by the designer and the program will alter the table. On the other hand, it would be possible to change greater blocks of logic by instructing the program that the present logic behind a line is to be changed and then specify the new logic. A program would then have to create the new circuits and remove all those circuits used for the old logic which are not used by other lines. A tape record would be kept of all changes made. If a design or transcription error was made this tape could be used to restore the machine record to its original status. Either of these systems would make machine circuit changes easier and more accurate than in the past.

A further refinement which can be added to the automation sequence is the various classes of timing tests. To test for circuit delays, either a separate program or part of the assembly program can add the delays given for each circuit and so compute the over-all delay. If the over-all delays permitted is noted in the original input, decisions based upon the allowable delay can be programmed. The program can select slower transistors if timing permits or parallel logic if the time delay of the original circuits was too long. A program would test for logical noise by simulating machine conditions for each pulse time. It could then step through the logical chain of affected circuits accumulating delay in some time unit less than a pulse time. If the program computed partial rises and falls, all significant logical noise could be noted and indicated to a designer. These types of programs would lift the burden of detailed timing investigations from the designers shoulders.

Since the packaging of the STRETCH components has not yet been crystallized, no work has been done in this area. There should be enough information in the circuit table to use it as the input data to any program which will be written later on.

The above approach can benefit the Stretch design program even if only the basic parts are ready when the final design is started. Since the basic tie in between different programs is the table representing circuits, programs can be altered without effecting others and any further programs added when they become necessary. That will mean that as more parts of the design procedure become routinized, these can also be put on the computer. Thus the Logical Designers should be relieved of much routine engineering.

L. S. Snyder

LSS/aww

November 12, 1956

PROJECT STRETCH

AUTOMATION OF DESIGN MEMO #1 APPENDIX

SUBJECT: A Format for Logical Design Programming

BY: L. S. Snyder

Y AND TAPOP, RDCALL:

The lines designated Tape Operation and Read Call are to be AND'ed. The result will be temporarily referred to as Y.

X OR TESYNC, NOY:

The line designated as Test Synch is to be OR'ed with inverse output of the line temporarily referred to as Y and the result will be temporarily referred to as X.

REP REP 1 BIT, 7. 1. 1:

Repeat the next instruction seven (7) times, changing the designation of the output and input by 1 both as to numeric address and mneumonic designation, increasing the first character if it is numeric or the last character if the first character is alphabetic.

SUM1 ADR BIT1, BIT2, BIT3; CARY1:

The lines designated as BIT 1, BIT 2, and BIT 3, are the inputs of a full adder whose sum output is to be designated as SUM 1 and whose carry output is to be designated as CARY 1.

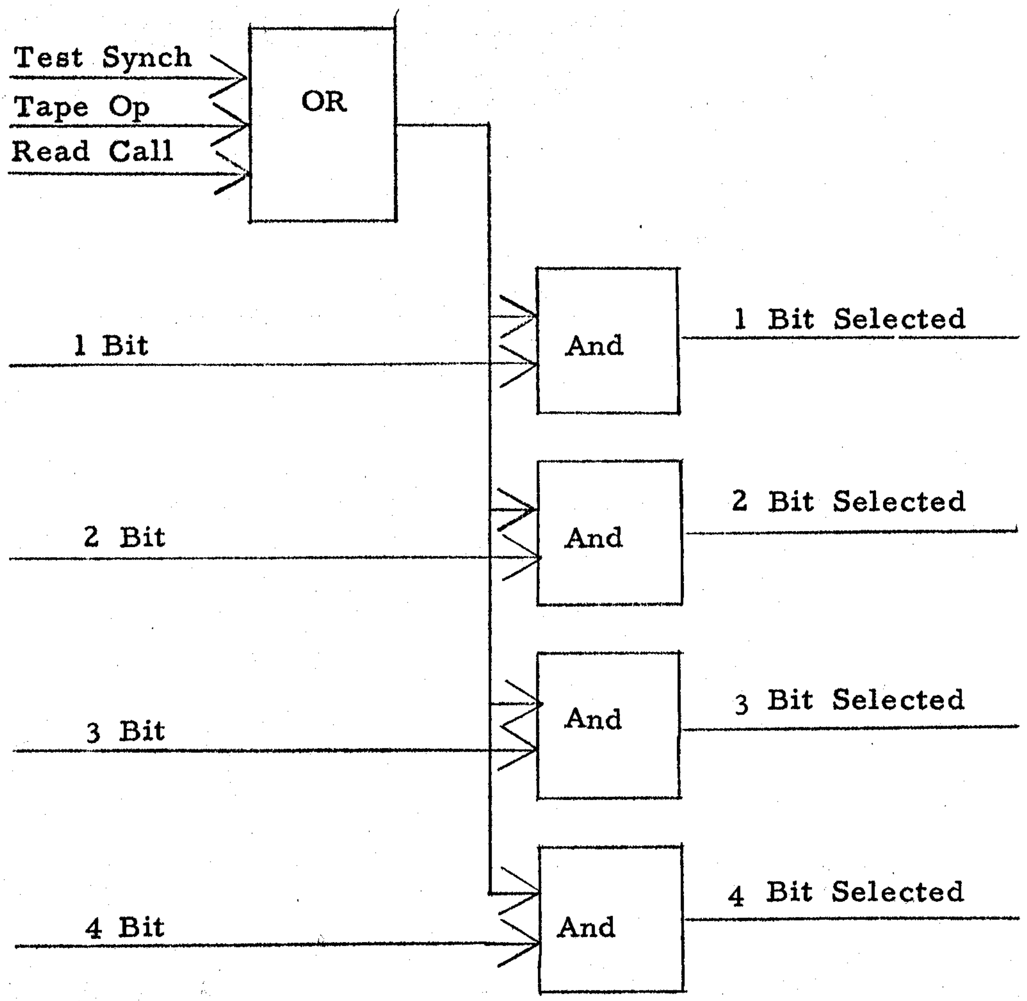
FIGURE I

Bit Position: Word	P	1 17	18 34	35
0	Circuit Description		Timing Data	
1	X	I/O Address	I/O Address	X
2	X	I/O Address	I/O Address	X
3	X	I/O Address	I/O Address	X
4	X	I/O Address	I/O Address	X
5	X	I/O Address	Extension location	Y
6	Physical Location Data			
7				

X = 0	Input Address
X = 1	Output Address
Y = 1	More I/O Addresses at extension location

3	5	8	1
Tape Unit	Record Number	Drum Location	Inverted Output

Address Bit Assignment



Systems Diagram

```
X         OR  TESYNC, TAPOP, RDCALL
REP       REP1BIT, 4. 1. 1
1BTSEL    AND1BIT, X
```

Equivalent Program