# Verification of the Logic Structure of an Experimental Switching System on a Digital Computer

By DOLORES C. LEAGUS, C. Y. LEE and GEORGE H. MEALY

(Manuscript received September 9, 1958)

*The verification problem is concerned with the construction on a computer of a logical program which satisfies all the design specifications prescribed for an experimental switching system and with the process of putting calls through the computer simulation to evaluate the system's logical structure.*

## I. INTRODUCTION

The experimental switching system,[1] unlike its predecessors the electromechanical switching systems, is a universal machine in the sense that its actions are dictated by an internally stored program. In order to verify whether the system behaves according to the designers' intent, it is therefore necessary to check both the program and the circuitry. Since by far the larger portion of the logical complexity of the switching system resides in the program rather than in the hardware, it might be supposed that a corresponding proportion of effort would be required to check the program in the system laboratory and, further, that the program could not be checked until the system circuitry was functioning properly.

A strikingly similar problem arises in computer development: when a new computer is in final development, programs must be written and verified by the time the first model of the computer is ready for use. The solution to this problem is, when feasible, to use an existing computer together with a so-called interpretative program which simulates the actions of the new computer. In this type of simulation, internal processing can almost always be simulated exactly, but at a sacrifice of processing speed; terminal equipment of the computer usually cannot be simulated exactly.

From our present point of view, the experimental switching system is a computer, with its central control corresponding to the control and

arithmetic sections, the flying spot store and barrier grid store corresponding to internal storage and the flip-flop groups and miscellaneous flip-flops in central control corresponding to the various registers and triggers in the arithmetic and control sections of a more conventional computer.

The IBM 704 was used for verifying programs for the experimental switching system, and the 704 program which was constructed for this purpose will be described in the following section. In this work, 19 different call patterns, involving intraoffice calls, outgoing calls, partial dials, calls to busy lines, wrong numbers and various other conditions were processed. A complete record of the progress of the calls was kept on each of the call patterns, showing the exact conditions of pertinent registers and conditions of flip-flops in the experimental switching system at the completion of every network operation. The 704 program can be used to record the conditions of every bit in the system's storage at any time.

## II. THE COMPUTER PROGRAM

The functions of the computer program are to simulate the operations of the central control, the scanner and the network control of the experimental switching system; to carry out the switching system program orders stored in the flying spot store; and to give output indications of whether calls submitted to the system are successfully completed.

In the switching system, actions are completed in units of 5 milliseconds. Each 5-millisecond period is called a cycle. A cycle is also the time that the switching system takes to go once through its main stored-program loop. In the simulation program on the computer, two counters are provided: a cycle counter and an order counter. The cycle counter keeps a count of the switching system program cycles, thus serving as the clock of the computer program. The order counter advances once each time a system order is carried out, but is cleared in the beginning of each cycle. It therefore keeps a count of the number of system orders executed within a cycle.

In the switching system customer actions and actions of the central control, scanner and network all occur in parallel, whereas on the computer these actions must be serialized. In the computer program, the serial positions in time of these actions are all in reference to the contents of the cycle counter.

In order to simulate the actions of the experimental switching system, the computer program contains complete images of the flying spot store and the barrier grid store as well as images of associated flip-flop registers.

In addition to these images, the computer program also includes an input program, an order control program, a network control program, an output program, a frequency-count program for order usage and other auxiliary programs.

### 2.1 *The Input Program*

The inputs received by the computer program consist of simulated dial pulses representing calls from customers. The form in which the dial pulses are received is illustrated in Fig. 1. Let us suppose that the telephone number dialed is WH 2-1111 and that the first off-hook pulse occurs at time $t_0$. Then the sequence of times $t_1, t_2, \cdots, t_{38}$ represents the number dialed and $t_t$ represents the time at which the call is terminated. If this call happens to be the only one in the system from time $t_0$ to $t_t$, the computer then receives as its input the sequence $t_0, t_1, \cdots, t_t$ in that order. If there should be several calls present in the system simultaneously, the sequence of times of each call is then interlaced with the other sequences and ordered according to the actual time of occurrence, and the complete list is stored in the computer in that order.

This list of times is examined once every cycle, when the contents of the cycle counter is compared with the next time entry appearing in the list. If this entry is not the same as the contents of the counter, no change in line condition has occurred in that cycle. If the two numbers are the same, then a change in line condition has occurred. This change is recorded by the computer program and the computer program enters the network control program.

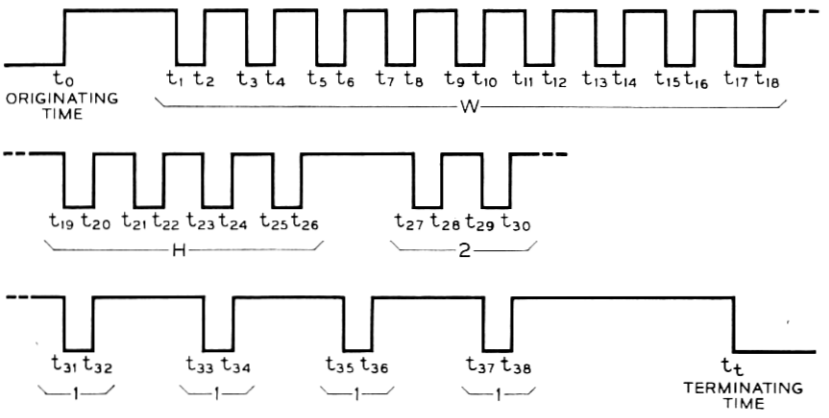


FIG. 1 — Form in which dial pulses are received..

### 2.2 *The Order Control Program*

The function of the order control program is to interpret and execute each of the 52 types of system program orders. At the heart of the order control program are some nine subroutines. Each system order is composed of five parts (A, B, C, D and E) and, according to the code for each part, a different set of subroutines is selected for the execution of the order by the computer program.

For example, the order 0,0,11,47,0 says that a customer line condition is to be matched against some spot in the barrier grid store. The address of the line is stored in the scanner register and the X part of the barrier grid store address is stored in the barrier grid register. The Y part of the barrier grid store address is the D code, or 47. If a mismatch occurs, the address of the next order to be executed is to be found in transfer register 1. If a match occurs, no transfer action is to be taken.

This order is carried out by the order control program in several steps. First, the order code is split into five subcodes, A, B, C, D and E. The subcodes A, B, C and E cause the order control program first to enter a subprogram which finds and stores the condition of the customer line in question, and then to enter another subprogram which finds and stores the condition of the spot in the barrier grid store in question. The two bits are then compared and no action is taken if a match occurs. If a mismatch occurs, a third subprogram is entered. The functions of this subprogram are, among other things, to set the return address, to set certain 704 addresses in preparation for the system program transfer, to check and see whether the transfer is to a new program order or for translation and, finally, to check the 704 console switches to see if transfer mode operation (Section 2.4) is desired.

### 2.3 *Network Control Program*

The function of the network control program is to simulate all the interactions between the network and central control circuitry. In the switching system, there are nine possible commands the common control circuitry may transmit to the network control, and the network can answer in one of two ways: "a command has been carried out partially," or "a command has been carried out completely and successfully." The network control program also sets and resets appropriate flip-flops, and checks and records conditions of all lines and trunks.

As an example, the command 0,20 says to release a distribution network connection from the A side of the distribution network; the A side distribution network trunk address is stored in the A side trunk selector

register. This command is carried out by the network control program in several steps. First, contents of the A side trunk selector register are gated to the A side trunk identifier register. The B side trunks are then scanned to see which one is connected to the A side trunk in question. The fact that both of these trunks are now made idle by the release command is then recorded. Finally, the network reports back to the central control that the command has been carried out completely and successfully.

### 2.4 *The Output Program*

There are three modes of operation in the computer program, giving three types of outputs. The desired mode of operation can be selected manually on the 704 console. In the first or normal mode of operation, the machine gives an output only under one of the following two situations:

i. The network had been requested to perform some action and has completed this action, or

ii. Dialed digits are being outpulsed to a distant office.

Under the first situation the following information is printed out on the 704 printer: the time (in terms of 5-millisecond intervals) at which the network action is completed; the number of stored program orders that were executed in the particular 5-millisecond interval involved; the network action itself; the contents of various flip-flops and the contents of the originating registers, ringing register, network register and disconnecting registers.

In the second mode of operation (transfer mode I), in addition to the print-out under normal mode, a transfer record for the stored program is written on tape, which can then be put on a peripheral tape-to-print converter to give a printed transfer record. A transfer record consists of a count of the number of actual transfers made in the stored program, the transfer order executed, the return address for the transfer order, the address to which the transfer is made and the 5-millisecond interval in which the transfer has occurred. The return address in this case means the address to which the common control would have gone if the transfer had not been made.

A sample transfer record for one of the call patterns is shown in Table I. The first transfer order executed is 0,0,1,48,0, which says to read the bit at address $X,Y$ of the barrier grid store, where $X = 48$ and $Y$ is to be found in the barrier grid register. The order itself is stored in address $X = 2$, $Y = 15$ of the flying spot store, and the address transferred to is

TABLE I

| Transfer Count | Return Address | | Order Executed | | | | | Transfer to Address | | 5-Millisecond Interval Count |
|---|---|---|---|---|---|---|---|---|---|---|
| | X | Y | A | B | C | D | E | X | Y | |
| 00001 | 03 | 15 | 0 | 0 | 001 | 048 | 0 | 18 | 15 | |
| 00002 | 21 | 15 | 0 | 0 | 001 | 050 | 0 | 26 | 49 | |
| 00003 | 31 | 49 | 2 | 0 | 010 | 028 | 0 | 10 | 28 | |
| 00004 | 14 | 28 | 0 | 2 | 000 | 026 | 0 | 08 | 30 | |
| 00005 | 15 | 30 | 0 | 0 | 004 | 004 | 0 | 22 | 40 | 00001 |
| 00006 | 03 | 15 | 0 | 0 | 001 | 048 | 0 | 18 | 15 | |
| 00007 | 21 | 15 | 0 | 0 | 001 | 050 | 0 | 26 | 49 | |
| 00008 | 31 | 49 | 2 | 0 | 010 | 028 | 0 | 10 | 28 | |
| 00009 | 14 | 28 | 0 | 2 | 000 | 026 | 0 | 08 | 30 | |
| 00010 | 16 | 30 | 0 | 0 | 004 | 006 | 0 | 22 | 40 | 00002 |
| 00011 | 03 | 15 | 0 | 0 | 001 | 048 | 0 | 18 | 15 | |
| 00012 | 21 | 15 | 0 | 0 | 001 | 050 | 0 | 26 | 49 | |
| 00013 | 31 | 49 | 2 | 0 | 010 | 028 | 0 | 10 | 28 | |
| 00014 | 14 | 28 | 0 | 2 | 000 | 026 | 0 | 08 | 30 | |
| 00015 | 15 | 30 | 0 | 0 | 004 | 004 | 0 | 22 | 40 | 00003 |
| 00016 | 03 | 15 | 0 | 0 | 001 | 048 | 0 | 18 | 15 | |
| 00017 | 21 | 15 | 0 | 0 | 001 | 050 | 0 | 26 | 49 | |
| 00018 | 31 | 49 | 2 | 0 | 010 | 028 | 0 | 10 | 28 | |
| 00019 | 14 | 28 | 0 | 2 | 000 | 026 | 0 | 08 | 30 | |
| 00020 | 18 | 30 | 0 | 0 | 004 | 008 | 0 | 22 | 40 | 00004 |
| 00021 | 03 | 15 | 0 | 0 | 001 | 048 | 0 | 18 | 15 | |
| 00022 | 21 | 15 | 0 | 0 | 001 | 050 | 0 | 26 | 49 | |
| 00023 | 31 | 49 | 2 | 0 | 010 | 028 | 0 | 10 | 28 | |
| 00024 | 14 | 28 | 0 | 2 | 000 | 026 | 0 | 08 | 30 | |
| 00025 | 13 | 30 | 1 | 0 | 002 | 002 | 0 | 28 | 05 | |
| 00026 | 30 | 05 | 1 | 0 | 000 | 008 | 0 | 07 | 03 | |
| 00027 | 10 | 03 | 2 | 0 | 020 | 054 | 0 | 20 | 54 | |
| 00028 | 25 | 54 | 0 | 0 | 001 | 048 | 0 | 26 | 54 | |
| 00029 | 29 | 54 | 0 | 0 | 001 | 050 | 0 | 22 | 40 | 00005 |
| 00030 | 03 | 15 | 0 | 0 | 001 | 048 | 0 | 18 | 15 | |
| 00031 | 21 | 15 | 0 | 0 | 001 | 050 | 0 | 26 | 49 | |
| 00032 | 31 | 49 | 2 | 0 | 010 | 028 | 0 | 10 | 28 | |
| 00033 | 14 | 28 | 0 | 2 | 000 | 026 | 0 | 08 | 30 | |

$X = 18$, $Y = 15$ of the flying spot store. The next transfer order is encountered at address $X = 20$, $Y = 15$, the order executed is 0,0,1,50,0 and the address transferred to is $X = 26$, $Y = 49$, and so forth. The numbers at the right are the 5-millisecond intervals in which the transfers have occurred.

The third mode of operation (transfer mode II) is similar to the second except that, in addition to a transfer record on tape, the record is also printed immediately. The purpose of operating in this mode is to have immediate access to the transfer information.

### 2.5 *Frequency Count for Order Usage*

A program to record the frequency of usage of the stored program orders in any given period of time is available. Records were kept on two of the 19 call patterns and one of these is illustrated in Table II, in which three columns of data are shown. The first column is the stored-program order code, the second is an order usage count when the system is idle (i.e. there is no call in the system) and the third column shows an order usage count when there is a single call in the system. A comparison of columns 2 and 3 shows what orders are used and how often they are used because of the presence of the call in the system. From these records it appears that, of the 52 stored-program orders, about 60 per cent are used less than 1 per cent of the total time and about 20 per cent are used more than 88 per cent of the time.

### 2.6 *Other Auxiliary Programs*

The computer program includes about 75 error stops. These point out incorrect functioning of the system and possible computer error. The appearance of an illegal system program order or other malfunctioning of the system would lead the computer program to an error stop. The error stop code is readily identified and the error involved is found by consulting a list of possible errors prepared beforehand.

The computer program also includes several conversion programs, of which one converts the system orders from symbolic to binary and another converts the input dial pulses from decimal to binary.

## III. REMARKS AND CONCLUSIONS

The computer program attains a computer-to-real-time time-ratio of about 15 to 1 in the normal mode of operation and about 50 to 1 in transfer mode I. In other words, in the normal mode, the 704 takes about 75 milliseconds to go through the work involved in a 5-millisecond period in real time. To get a transfer record on tape, it would take the 704 about one quarter second to go through a 5-millisecond interval in real time.

This time ratio is achieved by taking advantage of the extra time normally inherent in the actual system. The experimental system as a whole is essentially a clocked system, in that the principal parts of the stored program are normally repeated once every 5 milliseconds, even though the work specified for each 5-millisecond interval is usually performed in less than 5 milliseconds. By making the computer program asynchronous — in the sense that each simulated 5-millisecond interval

TABLE II

| Order Code | | | Usage, No Call | Usage, One Call |
|---|---|---|---|---|
| *A* | *B* | *C* | | |
| 0,1 | 0,1,2 | 0 | 872 | 944 |
| | | 1 | 1156 | 1161 |
| | | 2 | 835 | 860 |
| | | 3 | 0 | 0 |
| | | 4 | 867 | 998 |
| | | 5 | 0 | 0 |
| | | 6 | 54 | 87 |
| | | 7 | 0 | 2 |
| | | 8 | 0 | 19 |
| | | 9 | 0 | 5 |
| | | 10 | 432 | 432 |
| | | 11 | 0 | 344 |
| | | 12 | 0 | 0 |
| | | 13 | 1100 | 1160 |
| | | 14 | 0 | 1 |
| | | 15 | 0 | 0 |
| | | 16 | 0 | 1 |
| | | 17 | 0 | 1 |
| | | 18 | 0 | 1 |
| 2 | — | — | 1031 | 1115 |
| 3 | 0 | — | 0 | 0 |
| 3 | 1 | — | 0 | 64 |
| 3 | 3 | — | 803 | 1423 |
| 4 | — | — | 3879 | 4357 |
| 5 | — | — | 632 | 647 |
| 6 | — | — | 90 | 115 |
| 7 | 0 | — | 550 | 5126 |
| 7 | 1 | — | 0 | 29 |
| 7 | 2 | — | 0 | 355 |
| 7 | 3 | 0 | 0 | 101 |
| | | 1 | 393 | 404 |
| | | 2 | 0 | 0 |
| | | 3 | 0 | 0 |
| | | 4 | 0 | 3 |
| | | 5 | 0 | 11 |
| | | 6 | 0 | 2 |
| | | 7 | 27 | 46 |
| | | 8 | 0 | 0 |
| | | 9 | 0 | 0 |
| | | 10 | 1156 | 1162 |
| | | 11 | 1807 | 1858 |
| | | 12 | 0 | 81 |
| | | 13 | 0 | 4 |
| | | 14 | 0 | 4 |
| | | 15 | 0 | 0 |
| | | 16 | 0 | 0 |
| | | 17 | 0 | 0 |
| | | 18 | 0 | 0 |
| | | 19 | 0 | 2 |
| | | 20 | 0 | 1 |
| | | 21 | 0 | 2 |
| | | 22 | 0 | 0 |

is only as long as the work requires it to be — a sharp reduction in time ratio is attained.

To save time in putting calls through the computer program, customer dialing time is speeded up by a factor of about six (i.e. actual dialing time is six times longer). Under this condition, the longest call pattern involves some 4,000 5-millisecond intervals and takes approximately 5 minutes of computer time to go through.

In writing the computer program, we are confronted with three basic problems: the computer-to-real-time time-ratio, the process of understanding the system requirements and disciplines and the process of finding those output parameters which will give a good reflection of the correctness of the performance of the actual system.

The computer-to-real-time time-ratio determines whether a simulation problem is practically and economically possible to run. The simulation of any complex structure such as a telephone system is hampered at the start by the lack of commercially available computers which are capable of carrying out several logical programs simultaneously. It means that the work performed by a telephone system must be serialized before it can be programmed on a computer and, as a result, the computer program will be run at a much slower pace.

We have purposely omitted memory requirements in this list of problems. With the kinds of tape and optical storage now available, the amount of memory needed ceases to be a true obstacle in practice; rather, a basic consideration is the time requirement. In other words, we may say (with some exaggeration) that, in practice, as much memory as any person should need can be obtained, but that there may not be sufficient time in his life for him to make use of this memory.

The problem of understanding the system requirements and disciplines is a basic one in systems design. At the root is the question of how to find a simple and flexible language in which the requirements and disciplines can be spelled out concisely, correctly and easily. The logical design or analysis problem is then one of mapping (other names: compiling, automatic coding, automatic programming) the specifications in the basic language into the language of logical circuitry or the language of a computer.

As a result of this work, the logical validity of the program for the experimental system was verified in advance of the time when the program could be written into the flying spot store and verified in its ultimate environment. Until the program was used in the system, it could not be said to have been completely verified, since certain timing relations peculiar to the circuitry in the system could interact with the pro-

gram in an unfavorable manner. On the other hand, had the system itself been used for the entire program verification, the amount of time required for system testing would probably have been greatly increased. In the case of a suspected error, it might be quite difficult to determine whether the error was due to difficulty with the circuitry or to some peculiarity of the program. The use of a computer for program verification therefore has two great advantages: over-all developmental time for the system is reduced and logical difficulties can be made almost independent of electrical difficulties.

REFERENCE

1. Joel, A. E., Jr., An Experimental Switching System Using New Electronic Techniques, B.S.T.J., **37**, September 1958, p. 1091.