A DIGITAL LOGIC SIMULATOR PROGRAM

W. JOS. HEWITSON

A DISSERTATION

in the

Faculty of Engineering

Presented in partial fulfilment of the requirements for
the Degree of Master of Engineering at
Sir George Williams University
Montreal, Canada

April, 1974

ABSTRACT

A DIGITAL LOGIC SIMULATOR PROGRAM

W. JOS. HEWITSON

A digital logic simulator program is proposed for logic design verification, and as an aid to students in learning digital logic design. Logic to be simulated is specified by entering device types and nodal connections. A model is proposed with sufficient timing detail to allow the simulator system to detect timing errors which are currently found by prototype debugging.

The simulation algorithm is efficient in that combinational levels are re-evaluated only if their values are needed because of a change in one of their inputs. The simulation system will simulate both synchronous and asynchronous circuits.

The simulation system is written in FORTRAN and operates on-line in a conversational mode on a time-shared computer. It is currently implemented on the CDC 6400 at Sir George Williams University.

ACKNOWLEDGMENTS

The author wishes to express his thanks to Dr. G. Martin for his help and guidance in initially selecting this topic, and to Dr. T. Fancott for his constructive criticism, suggestions and guidance in the preparation of this dissertation.

Thanks are also due to Mrs. P. Graham who did such an excellent job on typing the manuscript.

TABLE OF CONTENTS

			Page
ABSTRACT			iii
ACKNOWLEDGEMENTS			iv
ILLUSTRATIONS			viii
INTRODUCTION			1
CHAPTER 1		REVIEW OF LITERATURE	4
	1.0	LOGIC SIMULATION PROCESS	4
	1.1	SIMULATION TECHNIQUES	6
	1.1.1	Compiler Simulation	6
	1.1.2	Table Driven Simulation	8
	1.1.3	Combined Compiler and Table Driven Simulation	9
	1.2	TYPES OF DIGITAL CIRCUITS	9
	1.3	SPECIAL SIMULATOR PACKAGES	10
	1.3.1	General Purpose System Simulators	11
	1.3.2	Pert	11
CHAPTER 2		SYSTEM SIMULATOR CHARACTERISTICS	14
	2.0	SIMULATOR SYSTEM OVERVIEW	14
	2.1	SYSTEM INPUTS	17
	2.2	DEVICE MODEL	19
	2.3	TABLE STRUCTURE	21
	2.3.1	Element Description Table	21
	2.3.2	Node Name Table	22
	2.3.3	Current and Old Value Tables	22

			Page
	2.3.4	Function Description Table	22
	2.3.5	Time Queue Table	23
	2.3.6	External Input Signal Table	24
	2.3.7	Device Fanout Table	24
	2.3.8	Table Structure for New Devices	27
	2.4	SIMULATION ALGORITHM	29
	2.4.1	Detailed Description of Simulation Algorithm	30
CHAPTER 3		THE SIMULATION PROGRAM	37
	3.0	OUTLINE OF PROGRAM	37
	3.0.1	General	37
	3.0.2	Main Section	37
	3.0.3	Subroutine CIRBLD	38
	3.0.4	Subroutine EXTRTN	38
	3.0.5	Subroutine PARRTN	39
	3.0.6	Subroutine PRTRTN	39
	3.0.7	Subroutine NEWCIR	40
	3.0.8	Subroutine SIMRUN	40
	3.0.9	Subroutine EDITRN	41
	3.0.10	Subroutine CORRTN	42
	3.0.11	Subroutine NDEVRTN	42
	3.1	STRUCTURE OF THE PROGRAM	42
	3.2	CAPABILITIES OF THE PROGRAM	76
	3.3	TRUTH TABLES FOR DEVICES	77
	3.4	OPERATING PROCEDURES	82
	3.4.1	System Commands	82
	3.4.2	Device Types and Names	83

Page

3.4.3 Entering the Circuit Description 83

3.4.4 Entering the External Signal Input Values 85

3.4.5 Obtaining An Edited Printout of the Circuit to be Simulated 87

3.4.6 Specifying Output Print Options 87

3.4.7 Changing System Parameters 88

3.4.8 To Enter A New Circuit Into The System 89

3.4.9. Defining A New Device To the System 89

3.4.10 Performing The Simulation 90

3.4.11 Making Changes In The Circuit Description 92

3.5 INPUT FORMATS 92

CONCLUSIONS 99

APPENDIX A 101

BIBLIOGRAPHY 111

LIST OF ILLUSTRATIONS

		Page
FIGURE 1.0	THE LOGIC SIMULATION PROCESS	5
FIGURE 2.0	SAMPLE SYSTEM INPUTS	18
FIGURE 2.1	COMPARISON OF ACTUAL CIRCUIT AND DEVICE MODEL	20
FIGURE 2.2	TABLE STRUCTURE	25
FIGURE 2.3	TABLE STRUCTURE MODIFICATIONS FOR USER DEFINED DEVICES	28
FIGURE 2.4	DETAILED FLOW CHART OF SIMULATION ALGORITHM	33
FIGURE 3.0	STRUCTURE OF PROGRAM	43
FIGURE 3.1	FLOW CHART- MAIN SECTION	45
FIGURE 3.2	FLOW CHART - SUBROUTINE CIRBLD	46
FIGURE 3.3	FLOW CHART - SUBROUTINE EXTRTN	48
FIGURE 3.4	FLOW CHART - SUBROUTINE PARRTN	50
FIGURE 3.5	FLOW CHART - SUBROUTINE PRTRTN	53
FIGURE 3.6	FLOW CHART - SUBROUTINE SIMRUN	56
FIGURE 3.7	FLOW CHART - SUBROUTINE EDITRN	66
FIGURE 3.8	FLOW CHART - SUBROUTINE CORRTN	68
FIGURE 3.9	FLOW CHART - SUBROUTINE PRTLIN	70
FIGURE 3.10	FLOW CHART - SUBROUTINE NDVRTN	72
FIGURE 3.11	DEVICE TRUTH TABLES	78
FIGURE 3.12	INPUT FORMATS	93
FIGURE A.1	SAMPLE CIRCUIT FOR SIMULATION TEST	109

INTRODUCTION

Digital Logic Simulation is the process whereby the functioning of a digital logic circuit, based on some model, is tested by the varying of both circuit parameters and simulation conditions.

Computer simulation has come into widespread use to study the behaviour of systems too large or complicated to deal with analytically. Alternatives to simulation are mathematical analysis or experimentation with the actual system or a prototype of the system. If a system is large, mathematical analysis may become very difficult or even impossible. Also, experimentation with actual or prototype systems can be costly and time-consuming. Moreover, the ever-increasing reliability requirements being placed on digital circuitry has made some sort of computerized simulation and fault diagnosis systems a must. A computer can uncover many timing problems in digital design that might have previously been found only by breadboarding. The simulation could also be used to predict the output of the circuit due to specified faults as well as predict the output of the fault free circuit.

Logic simulation complements rather than replaces actual hardware debugging. It can be used to minimize the debugging effort when hardware becomes available by having already checked out the logic so that the engineer can concentrate on other fault areas. Therefore, the simulator can assist digital system design in the following areas: by checking logic design alternatives, verifying fault test procedures, and obtaining detailed circuit operational reference data.

Some sort of computerized logic simulator would also be beneficial in the teaching of digital logic design. In this way, inexperienced students could be provided with a reasonably realistic and instructive imitation of digital hardware through which they could familiarize themselves with machine behaviour and design without actual hardware experience.

The selection of a simulation model, or models, is the most important problem in any simulation system. The model should be simple enough to use and real enough to give a reasonable approximation of the performance of the system.

In this dissertation, a system is proposed for digital design logic verification. Models for the logic circuitry have been selected, data structures for representing these models are given, and the simulation algorithm is outlined. No race or hazard analysis has been provided in this simulator. However, timing problems in the circuit can be detected as the system will print out nodal values at each unit delay time, if so instructed. The system output would then give a visual representation of the hazards present. One of the features of the simulator is that combinational levels are evaluated only when their values are needed, and only if they have changed since last evaluated.

Chapter I of this dissertation will be concerned with definitions, the various methods used in logic simulators, and explanations of some existing simulators. Chapter 2 will explain the table structure and simulation algorithm used in this simulator system, while Chapter 3 will deal with the actual program itself - operating instructions, structure, subroutine descriptions, etc.

Appendix A will contain sample terminal simulation runs for the circuits shown in Figure A-1.

CHAPTER I

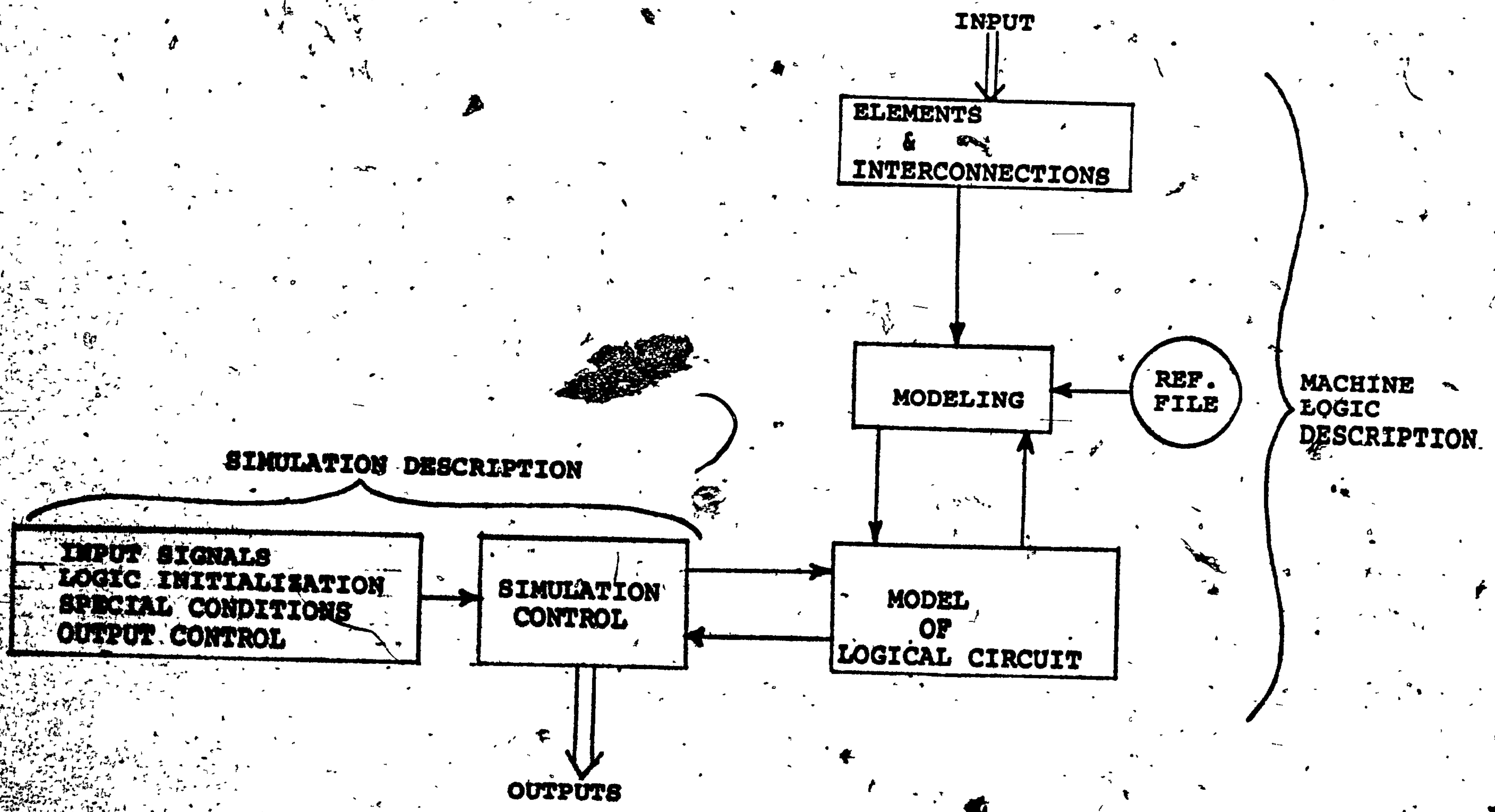
REVIEW OF LITERATURE

1.0 - LOGIC SIMULATION PROCESS

The simulation process combines the inputs describing the network with additional information in the simulator, converts the simulation commands describing the conditions of the simulation and dynamically executes the model by tracing logic values through the network.

The Logic Simulation Process, shown in Figure 1.0, consists of two phases: the machine logic description phase and the simulation description or processing phase. In the machine description phase, the input data statements describing the digital circuit are converted and combined with stored information to form the model of the logic network. The simulation commands direct the simulation processing by specifying inputs. Initial logic states and other special conditions, to obtain a realistic output, are also specified as inputs to the simulation process.

The logic circuit description and the simulation conditions should be able to be changed independently of each other. By changing the system description, different designs can be tested with the same simulation conditions, and similarly different simulation conditions could be tested with the same network without restating the circuit description.



LOGIC-SIMULATION PROCESS

FIGURE 1.0

1.1 - SIMULATION TECHNIQUES

The following techniques have been used for digital logic simulations. The relative advantage or disadvantage for each type would depend on the simulation model being used. The model is formed from the source input statements either by compiling the input statements into a set of subprograms or routines, or by translating the input statements into a table structure representing the network. Inputs to these simulators are generally Boolean expressions representing the network.

Compiled simulation uses the host computer to replace each nodal description with a sequence of code capable of simulating the node. The simulation is then performed by executing the compiled code.

Table driven simulators store network data in tables to be used in the simulation; Linkages between the logic elements are stored as pointers in the tables. The simulation is performed by manipulation of the data in these tables.

Another possibility is the use of a mixture of the two techniques described above.

1.1.1. - Compiler Simulation

The compiled simulator models the entire circuit as one unit and therefore one macro-model. The compiled simulator levels and transforms the circuit into a form that can be dealt with collectively. As most logic level simulators use Boolean equations to represent the elements of the network and their interconnections, the subprograms produced by the compiler represent a Boolean type operation. The indicated operations in each routine or subprogram is performed directly on Boolean variables. The resulting binary output of this evaluation is then passed on to the next level of the network.

The machine code executed must be in the same order in which the logic elements are activated by the signal paths. The order in which the elements in the Boolean expressions are evaluated is determined in a pre-compilation phase prior to the generation of the object code. After logic levels have been assigned to the gates, the gates are sorted by logic levels.

The level assigned to a signal is determined by the number of levels between an element and the system inputs. System inputs are assigned level zero. All other inputs have a level one greater than the level of the inputs of the gate of which it is an output. In this way, sequences of object code, ordered by element level, are then produced. When the simulation is executed, the elements are evaluated in order of increasing level with the elements of the lowest level processed first. Hence leveling of the circuit ensures that the output of any gate is not calculated until all its inputs are known.

As a result of levelizing, any updating of the network usually requires a reassembly of the code, and there is no easy provision for selectively testing a portion of a network.

1.1.2. - Table Driven Simulation

In a table driven simulator, the input source statements are translated into a data structure representing the network. In contrast to the compiler simulator, the table driven simulator models the circuit by breaking it into smaller blocks which can be individually modeled according to their type. In this way, the table driven simulator deals directly with elements. This type of simulator would determine, during simulation, what elements are to be evaluated next and then use a generalized routine to evaluate all elements of one type.

The simulation would be performed by a program which would follow the linkage pointers from one element to the next. The program would make use of the various entries in the tables to perform the required operation. The evaluation routine to be used would be determined by an entry in an "ELEMENT TYPE" column with values of inputs identified by pointers in a "FAN-IN" entry.

Unlike the compiler version, this type of simulator would only use one set of object code to simulate each logical operation. The required routine would just be executed each time that type of operation or element was to be evaluated.

Basic entries required to describe an element would be: 1) name given to the element; 2) element type; 3) pointers to the input and output nodes of the element. Linkages between elements would be determined by the program by means of the pointers to the inputs and outputs of individual elements.

Unlike the compiled code model, modifications can be easily made to the network by changing pointers in the tables. The table driven version is more general than the compiler version and can handle a large majority of circuit types, but it does have the disadvantage of being slower than the compiled version.

1.1.3 - Combined Compiler and Table Driven Simulation

The compiler simulator and table driven simulator could be combined by replacing the element type pointer in the tables with an address that points to object code to simulate that particular logic operation. This would, in essence, be a table driven simulation with that particular object code being executed each time that particular logic element was to be evaluated.

1.2.0 - TYPES OF DIGITAL CIRCUITS

Types of circuits that can be simulated are referred to as either asynchronous, synchronous, or a combination of both. The terms asynchronous and synchronous refer to the manner in which timing considerations are handled in the model.

Asynchronous operation means that the output of a gate at any time is determined solely by its inputs. The input values are used to calculate the device output as soon as the input values change.

Synchronous operation means that the output of a device is not solely dependent on its inputs. A synchronous device would only change state under control of clock pulses which occur at regular intervals. For a flip flop type device, the gated inputs would only be used to determine the output when the clock pulse occurred.

Purely synchronous circuits could be simulated very effectively by a compiled code simulator because the logic is only allowed to change at specified intervals (clock times) and device delays need not be taken into account.

However, the ability to simulate asynchronous circuits relies on the ability to accurately represent the time associated with the evaluation of signal values. In a real circuit, cumulative delays of logic elements may cause circuit behaviour to be asynchronous with respect to the clock. In this case, a table-driven simulation may give more realistic results.

1.3.0 - SPECIAL SIMULATOR PACKAGES

As well as specific simulation systems written for a specific task, there exists general purpose simulation programs which could be used in logic simulation. These programming systems consist of the general purpose system simulators, GPSS and SIMSCRIPT, and PERT[1].

[1] - PROGRAM EVALUATION AND REVIEW TECHNIQUE

1.3.1 - General Purpose System Simulators

Logic behaviour can be simulated by any general purpose simulator that models a discrete environment. The structure of the network, its behaviour and the simulation conditions can be expressed using the simulation language.

GPSS defines as its basic entities:

1) transactions which move through the system;
2) facilities which are operated upon by these transactions;
3) blocks which provide the rules by which transactions flow.

For logic simulation, the above concepts would represent: 1) signal pules, 2) digital elements, 3) the signal paths of the network. The logic simulation rules are defined using the general purpose simulator language structures. A signal must be acted upon by a gate before continuing to move through the system. When their inputs change, gates are put in an event queue and are replaced when their outputs change. Time advances according to the time at which the next event is to occur. The output and report generation features of the simulators would describe the system flow and state changes and also provide summary statistics.

1.3.2 - Pert

Pert was originally developed to identify critical paths and provide statistical estimates of end dates in large projects. A modification of this technique has been used to obtain worst-case delay information. Information such as most probable time delay, critical paths, and the probability of any given delay are yielded by using PERT.

When used as a scheduling aid, PERT requires as input three estimates concerning the length of time it will take to do a given job. These estimates are: 1) the most likely time, 2) the shortest time, and 3) the longest time. As output PERT would provide the probability of finishing the job at any given time.

In the application of PERT to logic systems, three estimates of circuit delay as indicated above, would have to be specified. Then by using the PERT procedure, the probability of any given delay can be computed. Timing slack and critical paths can also be determined.

In normal PERT usage, the network is a flow graph between completed events represented by nodes. No events take place until all input events connected to it have been completed. The branches between the nodes represent the time required to achieve the next event given that all input events have occurred. In the application of PERT to logic design, the nodes are replaced by logic blocks. Time delays are thus associated with the blocks rather than the interconnecting lines. The lines or branches represent signal flow and have no significance with respect to delay. Two sets of delays are normally used, one indicating the time for the element to turn ON and the other the time for the element to turn OFF. A logical AND is represented by using the longest time delay of an input branch and an OR is represented by using the earliest time of any input for critical path computation.

To use the PERT technique, the basic data required would be: 1) three delay estimates for each type of logic block; shortest, most likely, and longest delay; 2) a normal distribution curve in tabular form. With this data, the application of the PERT procedure would yield quantitative estimates for:

1) Probability that an output will occur by a given time.
2) Critical paths.
3) Timing slack allowable between various inputs.

CHAPTER 2

SYSTEM SIMULATOR CHARACTERISTICS

2.0 - SIMULATOR SYSTEM OVERVIEW

The simulator described in this report is a table-driven simulator. The simulator works at the gate level rather than the subsystem level where registers, adders, etc., are modelled.

The simulation algorithm uses a technique called Selective System Trace [3] [9] for combinational circuits. This is the evaluating of gate outputs only when there has been a change in one of the inputs. In an actual circuit, the number of elements actually changing state at any one time is very small compared to the total number of elements (figures of 1 to 2.5% have been stated in the literature [9] [20]). Any simulation technique which follows the signal propagations through the circuit and only evaluates the gates required virtually imitates a given network and represents a very efficient method of simulating the network.

Earlier simulators had cycled through the network evaluating all elements in the network each clock time whether required or not.

This simulator is capable of simulating both synchronous, asynchronous or a combination of both types of circuits. All synchronous devices, i.e. flip flops, are evaluated at the beginning of each clock cycle.

Subroutines are used to simulate the normally very simple behaviour of the element types. Element outputs are not calculated by checking all input values in a truth table. Unique values are checked first and if they exist, the remaining inputs are not checked. For example, for a 4 input NAND gate, if the first input had a value of 0, then the output of the gate would be 1 no matter what the other inputs are. The other inputs are disregarded thus cutting down simulation time. This is accomplished by using conditional branches and not with AND or OR instructions.

This simulator uses three simulation values, the normal values of 1 and 0, and X to represent the unknown or "don't-care" condition. The X value is used during initialization to represent the complete unknown state of the network at the beginning of simulation. While the simulation could be started from some specified initial state other than the unknown (all nodes having value X), this would assume that in the actual circuit it is possible to set this state, and disregards the possibility of inherent design faults preventing this state from being set. The only way to guarantee an accurate representation is to start with a completely unknown circuit state and specify system inputs to initialize the network. This simulator will initialize the circuit by setting all node values to X and all external input values to 0. The user may override the initial zero setting of the external inputs and set any external input to the value desired to obtain the required initial state from which to test the network.

Delays are specified as a built in characteristic of each element grouping. These delays values are not fixed and can be overridden by a user specified value.

The usefulness of any simulator is really determined by its output. A very efficient and realistic simulator would be of no benefit if the outputs derived were of no use to anyone. The minimum a user should be able to specify is: 1) a list of the signals to be monitored, 2) output formats, and 3) the time at which output is to be produced. This simulator allows the user to specify which nodes to monitor, whether output is to be printed every clock time or every unit delay time, and also to specify conditions under which output is to be produced, i.e. when a specified node has a certain value. The simulator output is a chart of signal value VS time with each unit delay time (if requested) in each clock cycle indicated on the printed output as well as the clock time.

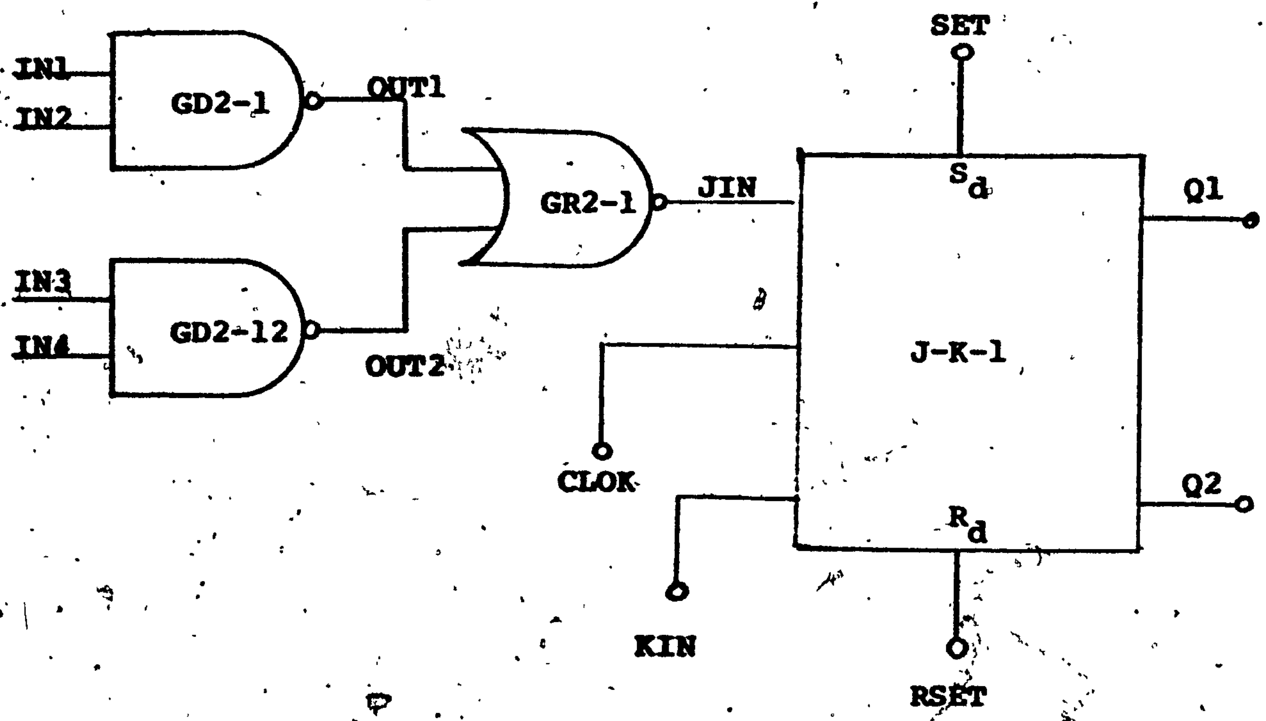
As stated in the Introduction, there is no hazard analysis performed by this simulator. However, possible hazard conditions can be detected by specifying the output to be printed every unit delay time. In this way, possible hazard situations, such as all inputs to a gate changing simultaneously, can be detected visually.

If a network being simulated should begin to oscillate, the simulator will abort the simulation after a specified amount of time and indicate this condition with an appropriate message.

2.1 - SYSTEM INPUTS

The majority of simulators studied by the author either used a special design language to describe the network or specified the network in terms of Boolean equations. Both of these methods were rejected for this simulator. The large number of Boolean equations required to specify a digital network makes it difficult for anyone who is not acquainted with the design to deduce its behaviour from a listing of the equations. Likewise, it could be very tedious and time consuming to formulate the design in such detail. As for a special design language to specify the network, it was felt that requiring a casual user to learn such a language detracted from the actual use of the simulator.

The inputting of network information is very oriented towards the user. To code the network, the user needs only a well-labelled circuit diagram. All nodes (inputs and outputs of all devices) are labelled and all devices assigned a name which indicates the circuit type - see Chapter 3 for the names of the devices simulated by this program. Hence, the description of any one element consists of - element type, name and input and output designations or labels. The user then is required only to describe each gate and its corresponding inputs and outputs independently of all other elements in the



CIRCUIT TO BE SIMULATED

```
GD2-1 IN1 IN2 OUT1
GD2-12IN3 IN4 OUT2
GR2-1 OUT1OUT2JIN
J-K-1 JIN KIN CLOKSET RSETQ1  Q2
```

Element type, Number, Input-output labels

SIMULATOR INPUT REQUIRED TO DESCRIBE THE CIRCUIT

FIGURE 2.0

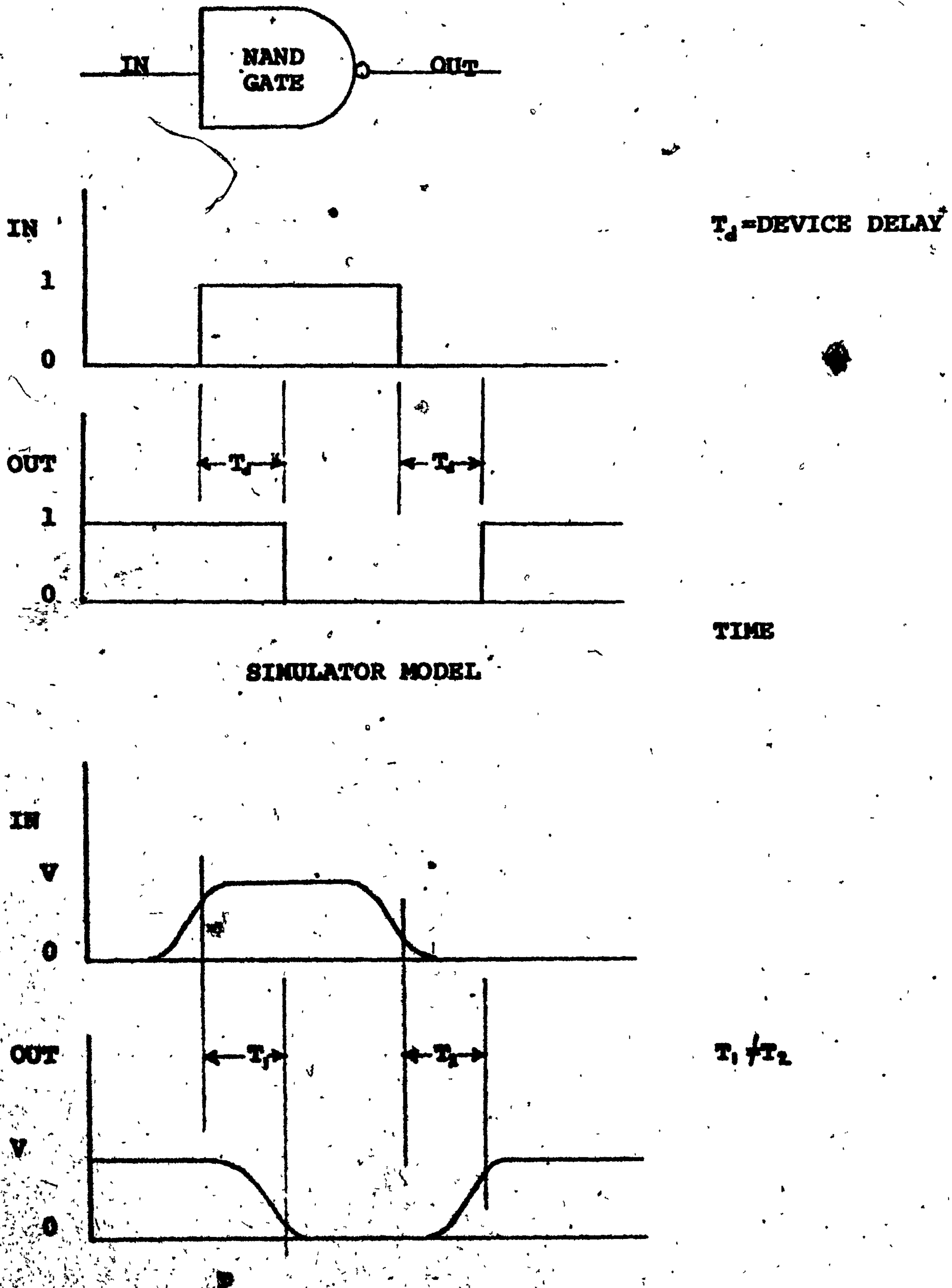
network. The interconnection of the elements is performed by the program. The formats for these element descriptions, as well as simulation command formats, are specified in Chapter 3. A sample circuit and its required system input description is shown in Fig. 2.0.

2.2 - DEVICE MODEL

The simulator uses an idealized gate model in which signal transitions are assumed to occur instantaneously rather than with finite rise and fall times. See Fig. 2.1.

Delays are specified as a built-in characteristic of element groupings. The logic simulation is time quantized and signals change value only at discrete time units. The value assigned to these units is relative since gate propagation delays are specified in terms of an integral number of these units.

The simulator does not take into consideration such factors as a change in device delay with loading, delays associated with lead lengths, variations of parameters from logic block to logic block, stray capacitance and other signal circuit properties. Also device turn-on and turn-off delays are usually different. This simulator uses one delay value for both turn-on and turn-off delays. The delays are normally set up as a constant value, usually either the nominal value furnished by the manufacturer or the worst-case delay. Although the actual value in a real gate may vary from this



SIMULATOR MODEL

ACTUAL CIRCUIT

COMPARISON OF ACTUAL CIRCUIT AND SIMULATOR MODEL

FIGURE 2.1

value, it is generally sufficient to measure the maximum time for the signal to propagate through the network, for most simulation requirements.

The program initializes all element delays to a value of one. The user can override this value and specify for his simulation run, the relative delays required for each element grouping in terms of delay time units.

2.3 - TABLE STRUCTURE

The Basic simulator consists of the following tables: the Element Description table, the Node Name table, the Time Queue table, the Function Description table, tables which contain the current value and old value of each node (or signal), an External Input Signal Value table, and a Device Fanout table.

2.3.1 - Element Description Table

The Element Description table contains the information describing the network to be simulated. The size of the table is 100 rows by 8 columns. Each row contains the descriptive information for one element. The first entry in the table is a pointer to an entry in the Function Description table which determines the type of element and contains parameters for that element. The remaining 7 entries in each row are pointers to entries in the Node Name table and represent the inputs and output nodes of the element. Seven was chosen because it takes seven entries to fully describe a flip-flop device - 2 gated inputs, a clock input, a direct set input, a direct reset input and 2 outputs.

2.3.2 - Node Name Table

This table, as the name implies, stores the names of all the nodes in the network. The table has room to store 500 names. On input, the node names are stored in this table and the position of the name in the table serves as the number of the node, which is subsequently used in other tables, for example the Element Description table.

2.3.3 - Current and Old Value Tables

These tables are parallel to the Name Table and each contains space for 500 entries. In these tables are stored the current signal value and previous signal value of all nodes in the network. The Old Value Table is required as some devices, i.e. flip-flops, respond to a transition from one logic level to another. Thus the evaluation routine for this type of element checks this old value before calculating the new output. The Current Value table entry for a node is updated at the point in time when that node is scheduled to change value. Hence, the value of any node is known at any unit delay time. The contents of the Current Value table are updated to the Old Value table after each unit delay time and after each clock cycle.

2.3.4 - Function Description Table

The Function Description table contains parameters for the device types simulated by the program. This table has provision for 20 entries with each entry consisting of 3 parameters. The parameters are: 1) a pointer to an evaluation routine for the element, 2) number of inputs, and 3) the element delay. Used with this table and parallel with it is a table containing the coded name of the devices that the system is able to simulate. The position of the coded name in the table is the pointer that is stored in column 1 of the Element Description Tables.

Setting up the device delay and number of inputs as table values minimizes the number of evaluation routines needed for simulation. By using this table, the same routine would be used for a 4 input NAND gate with a delay of 5, as would be used by a 2 input NAND gate with a delay of 7.

The table is functionally divided into two parts. All gate descriptions are found in the first 14 entries with entries 15 to 20 used for edge sensitive devices (flip-flops). This division is used when evaluating synchronous devices at the beginning of each new clock cycle. The element table is scanned for entries with device pointers of 15 or greater.

2.3.5 - Time Queue Table

The Time Queue table contains the events that occur at some time t, where t is the index of the time queue table. This table is parallel to the Name table, has 500 entries with 2 columns per entry. This table is used to schedule future events. There is an entry in this table for every node in the system. This table stores the future value of the node and the delay time at which this new value will occur. When the simulator run-time (unit delay time) matches that of the time stored in this table, the value specified is updated to the Current Value table for that node.

2.3.6 - External Input Signal Table

This table is used to store the pointers to the Name table for the nodes designated as inputs and the clock values assigned to these nodes. This table has provision for twenty nodes and values for fifty clock times to be stored. At the beginning of each clock cycle, the values for the external inputs at that clock time are updated to the Current Value Table.

2.3.7 - Device Fanout Table

The Fanout table is parallel to the Name table and contains ten entries per row. It is, in fact, a fanout table for every node in the system. The entries in this table are pointers to an entry in the Element table, each entry in the Element table representing a logic device. Any device for which a particular node is an input will be pointed to by an entry in the Fanout table for that node. A maximum of ten fanout devices is allowed per node. This table is used during the simulation run to determine which devices are affected by a change in value of a node so that only these devices affected are evaluated and the rest ignored.

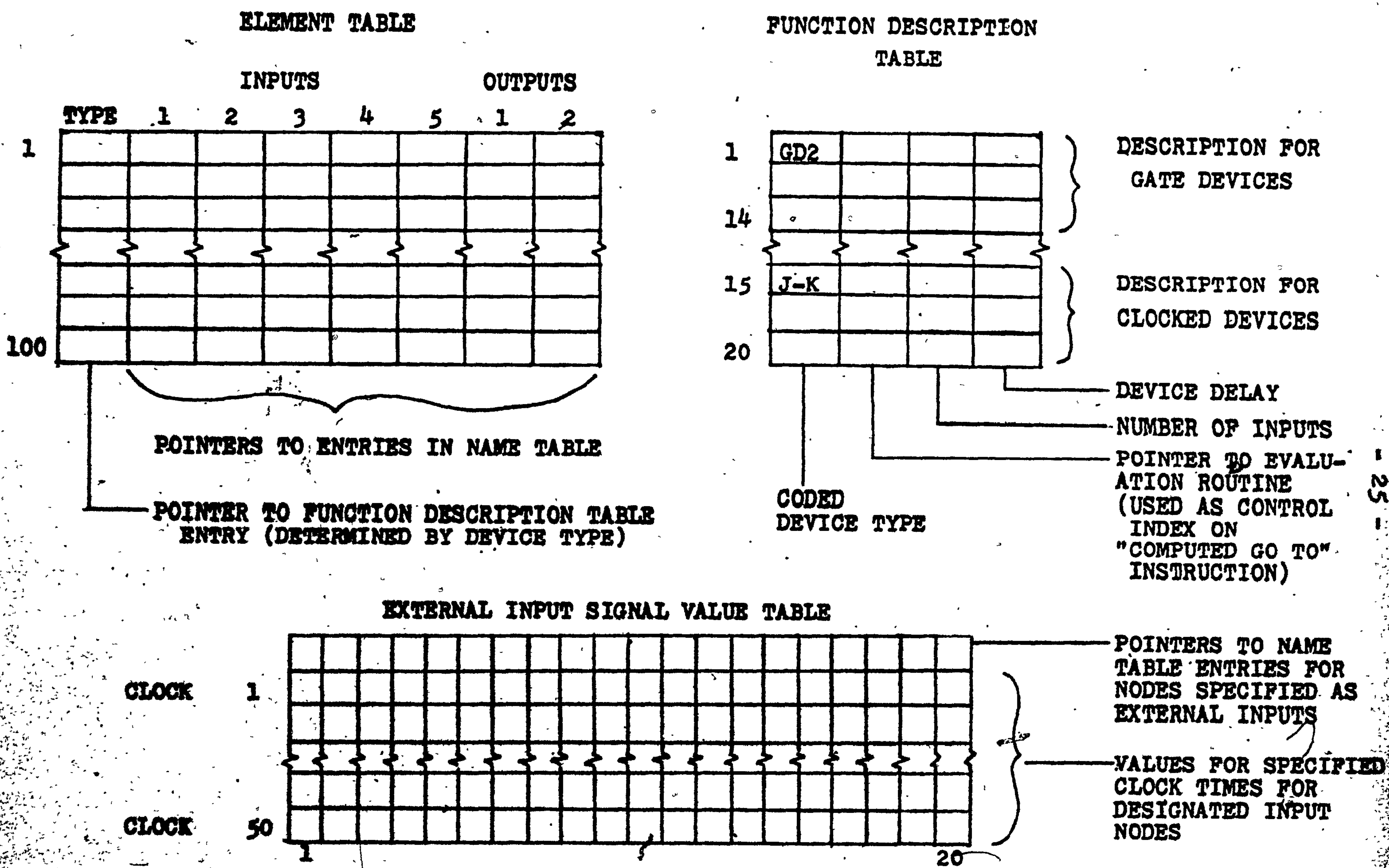
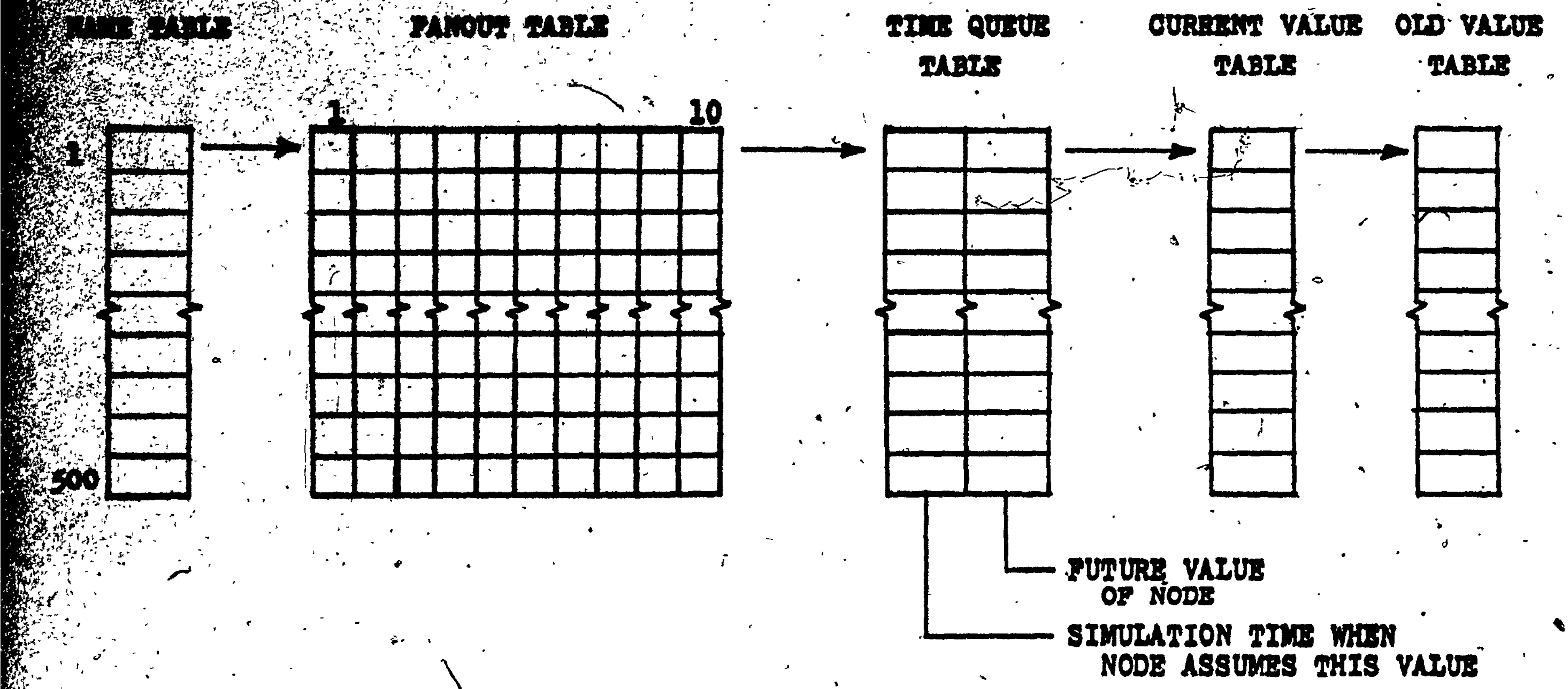

ELEMENT TABLE
INPUTS
OUTPUTS
TYPE
1
2
3
4
5
1
2
1
100
POINTERS TO ENTRIES IN NAME TABLE
POINTER TO FUNCTION DESCRIPTION TABLE ENTRY (DETERMINED BY DEVICE TYPE)
FUNCTION DESCRIPTION TABLE
1
GD2
14
15
J-K
20
DESCRIPTION FOR GATE DEVICES
DESCRIPTION FOR CLOCKED DEVICES
DEVICE DELAY
NUMBER OF INPUTS
POINTER TO EVALUATION ROUTINE (USED AS CONTROL INDEX ON "COMPUTED GO TO" INSTRUCTION)
CODED DEVICE TYPE
EXTERNAL INPUT SIGNAL VALUE TABLE
CLOCK 1
CLOCK 50
1
20
POINTERS TO NAME TABLE ENTRIES FOR NODES SPECIFIED AS EXTERNAL INPUTS
VALUES FOR SPECIFIED CLOCK TIMES FOR DESIGNATED INPUT NODES

TABLE STRUCTURE

FIGURE 2.2



NOTE: TABLES ARE IN PARALLEL IN THAT ALL VALUES IN THE TABLES REFER TO THE NODE NAME OCCUPYING THE SAME LOCATION IN THE NAME TABLE.

TABLE STRUCTURE (CONT.)
FIGURE 2.2

2.3.8 - Table Structure For New Devices

In order to provide the capability of defining new devices to the simulator for any simulation run, it was necessary to modify the basic table structure. The modifications to the structure involve adding two tables, an Auxiliary Description table and a Truth table. At the present time, the system allows only two new devices to be defined during a simulation, hence there are only two entries in the Auxiliary Description table and only two Truth tables set up.

The Auxiliary Description table has four parameters for each entry. These parameters are: 1) the number of input nodes, 2) the number of output nodes, 3) if the device is an edge sensitive device - the value of the signal before the triggering edge, 4) a pointer to the Truth table used for this device. An entry in the Function Description table is also used for the new device. The Function Description table has six empty positions, three for gate devices and three for edge-sensitive or clocked devices. To enter a new device, the Function Description table is scanned and the first available position for that device type, i.e. gate or clocked, is used. The entries in that position are as follows: device name, pointer to a standard evaluation routine, a pointer to the entry in the Auxiliary Description table, and the device delay.

The size of the Truth tables set up for each new device is 25 rows by 7 columns. This allows a device with five inputs and two outputs to be described. The system requires all input and output combinations to be specified i.e. for a 5 input device, all 25 input combinations and their respective outputs must be specified.

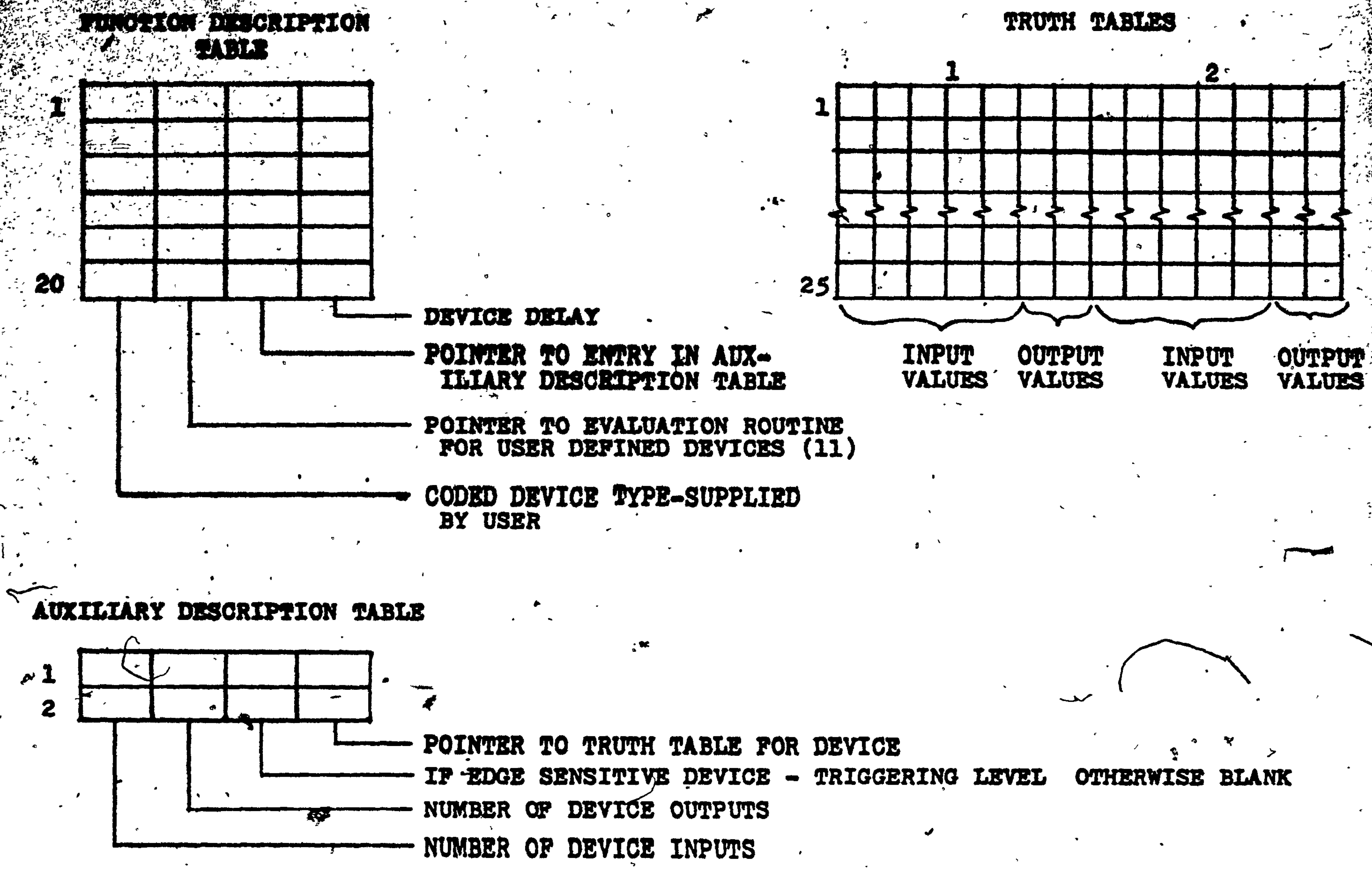

FUNCTION DESCRIPTION TABLE
1
20
DEVICE DELAY
POINTER TO ENTRY IN AUXILIARY DESCRIPTION TABLE
POINTER TO EVALUATION ROUTINE FOR USER DEFINED DEVICES (11)
CODED DEVICE TYPE-SUPPLIED BY USER
TRUTH TABLES
1
2
1
25
INPUT VALUES
OUTPUT VALUES
INPUT VALUES
OUTPUT VALUES
AUXILIARY DESCRIPTION TABLE
1
2
POINTER TO TRUTH TABLE FOR DEVICE
IF EDGE SENSITIVE DEVICE - TRIGGERING LEVEL OTHERWISE BLANK
NUMBER OF DEVICE OUTPUTS
NUMBER OF DEVICE INPUTS

TABLE STRUCTURE MODIFICATION FOR USER DEFINED DEVICES

FIGURE 2.3

See Fig. 2.3 for the table structure for User defined devices.

2.4 - SIMULATION ALGORITHM

The model used to simulate the action of the interconnection of several gates is just as important as the gate model. The accuracy of the simulation depends on a good circuit model and the objective is to imitate the action of the actual circuit as closely as possible. A problem encountered with simulation is the time synchronization of the model elements. A real network will perform certain events simultaneously while the simulation must perform these events in a sequential manner. Since a computer performs only a sequence of operations, digital simulation must inherently be discrete. Thus the sequence of device evaluation must be time synchronized in order to realistically model the system behaviour. Using the tables described in the Table Structure section, the simulation is performed as follows:

1) All values that exist in the Time Queue table at the current simulation time are updated to the Current Value table.
2) Using the Fanout table, all devices that are affected by this change are re-evaluated using the new value of the node.
3) The result of this re-evaluation, if different from the current value of the output node, is stored in the Time Queue table with a schedule time equal to the current time plus the propagation delay of the signal.

4) The current time is incremented by 1 unit delay time and the Time Queue checked for entries with a schedule time equal to the current time. When found the process is repeated again.

This is basically the simulation technique employed - only those devices whose inputs have changed are evaluated and all evaluations at a given time are done before tracing the signal path further.

A more detailed explanation of the simulation program is given in the next paragraph along with a Flow Chart (Fig. 2.4).

2.4.1 - Detailed Description of Simulation Algorithm

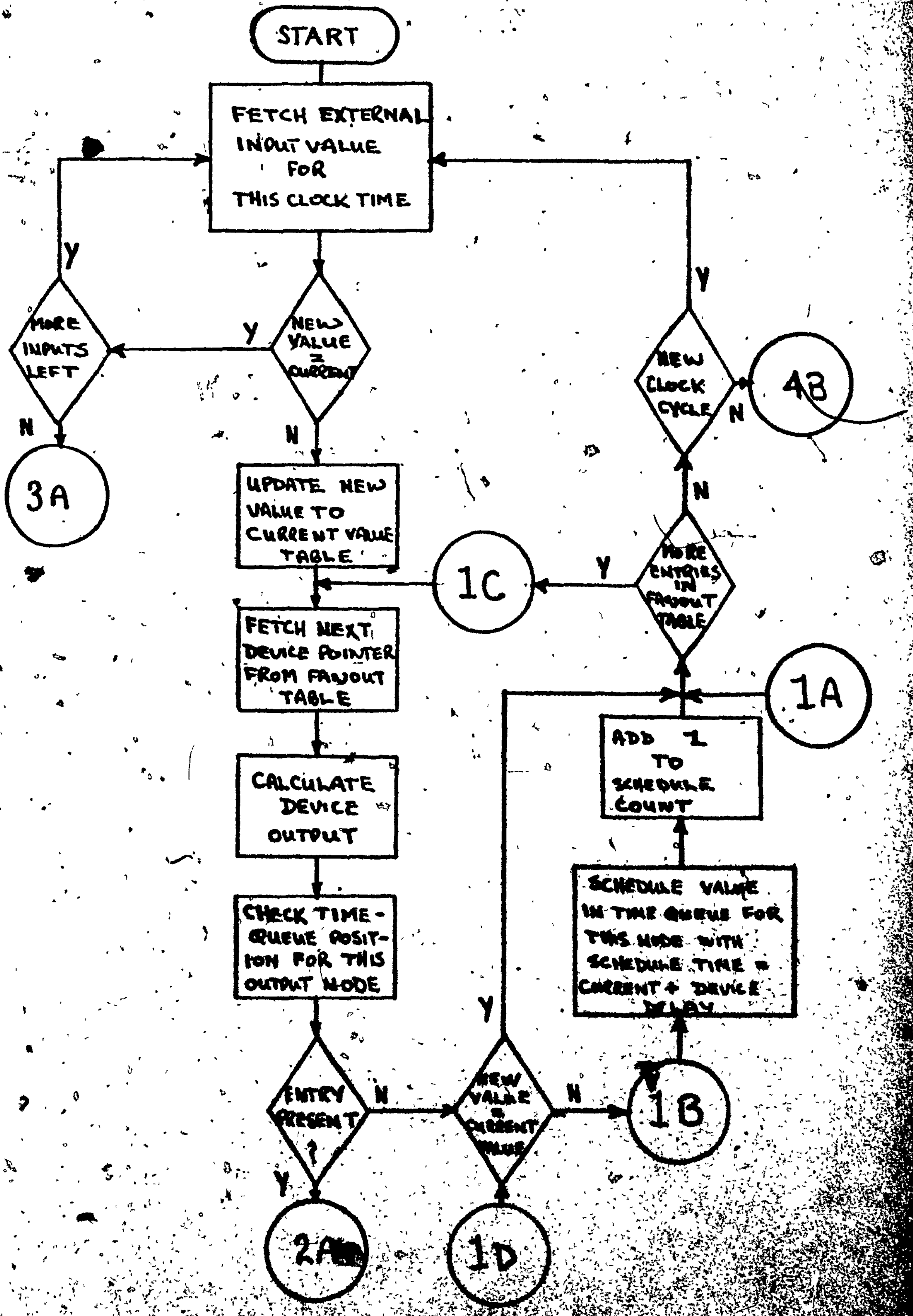
Fig. 2.4 is a detailed Flow Chart of the simulation technique used in this program.

To begin the simulation, the external input values are read from the External Input table. The new value is compared to the current value in the Current Value table. If equal, no further action is taken for this external node and the value for the next external input is retrieved. If the two values are not equal, the new value is updated to the Current Value table. By means of the Fanout table, all devices which are affected by this change have their outputs calculated. If the calculated output of the device is not equal to the current value of the output node, the new value is stored in the Time Queue entry for this node. The schedule time stored is equal to the current time plus the specified device delay. A count is kept of all nodes scheduled. After all external inputs have been updated to the Current Value table, and all outputs calculated and scheduled for the devices affected, the synchronous devices are evaluated.

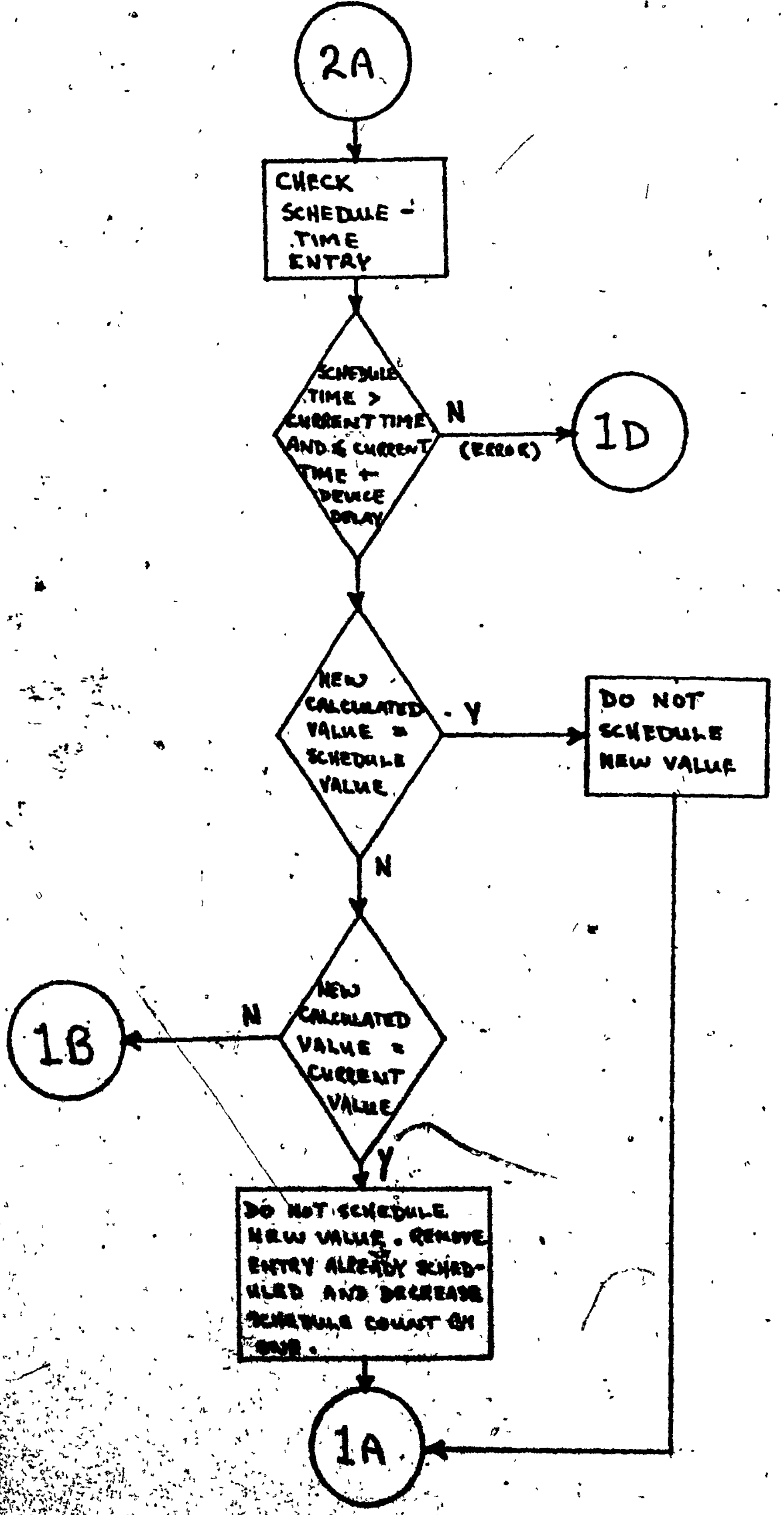
The Element table is scanned for synchronous devices and their outputs calculated. Any outputs that differ from the current value of the nodes is scheduled as outlined above and the schedule count incremented. After all synchronous devices have been evaluated, the combinational network is simulated by tracing the signal through the network.

The simulation time is initially set to 1 and the Time Queue is scanned for any entries with a schedule time equal to the simulation time. If an entry is found, the schedule value is updated to the Current Value table and the entry is deleted. The schedule count is then decremented by 1. The devices specified in the Fanout table for this node are evaluated one at a time. The Time Queue position for the output node just calculated is checked to see if there is an entry there already. If no entry, the calculated value is compared to the current value of the output node. If equal, no further action is taken. If not equal, the node is scheduled as explained above and the schedule count incremented by one. If an entry is there, the schedule time of the entry is then checked. This is done to suppress any spikes that may occur. A spike is defined as an attempt to change the output of a logic device faster than its inherent propagation delay. If the schedule time does not fall within the current simulation time and the simulation time plus the device delay, this represents an error and the newly calculated value is compared to the current value. If not equal, the node is scheduled as above and the schedule count incremented by one. If the schedule time falls in this range, the schedule value is compared to the newly calculated value. If equal, the new value is not scheduled. If not equal, the new value is compared to the current value. As this simulator uses 3 simulation values, the new value may or may not be equal to the current value. If not equal to the current value, the new value is scheduled without altering the schedule count. If the values are equal, this condition then represents a spike as defined above. The new value is not scheduled and the entry in the Time Queue is deleted and the schedule count decremented by one.

This process is repeated by incrementing the simulation time until the schedule count is zero. At this point the circuit has been simulated for one clock time. This process is repeated until the desired number of clock times have been simulated.

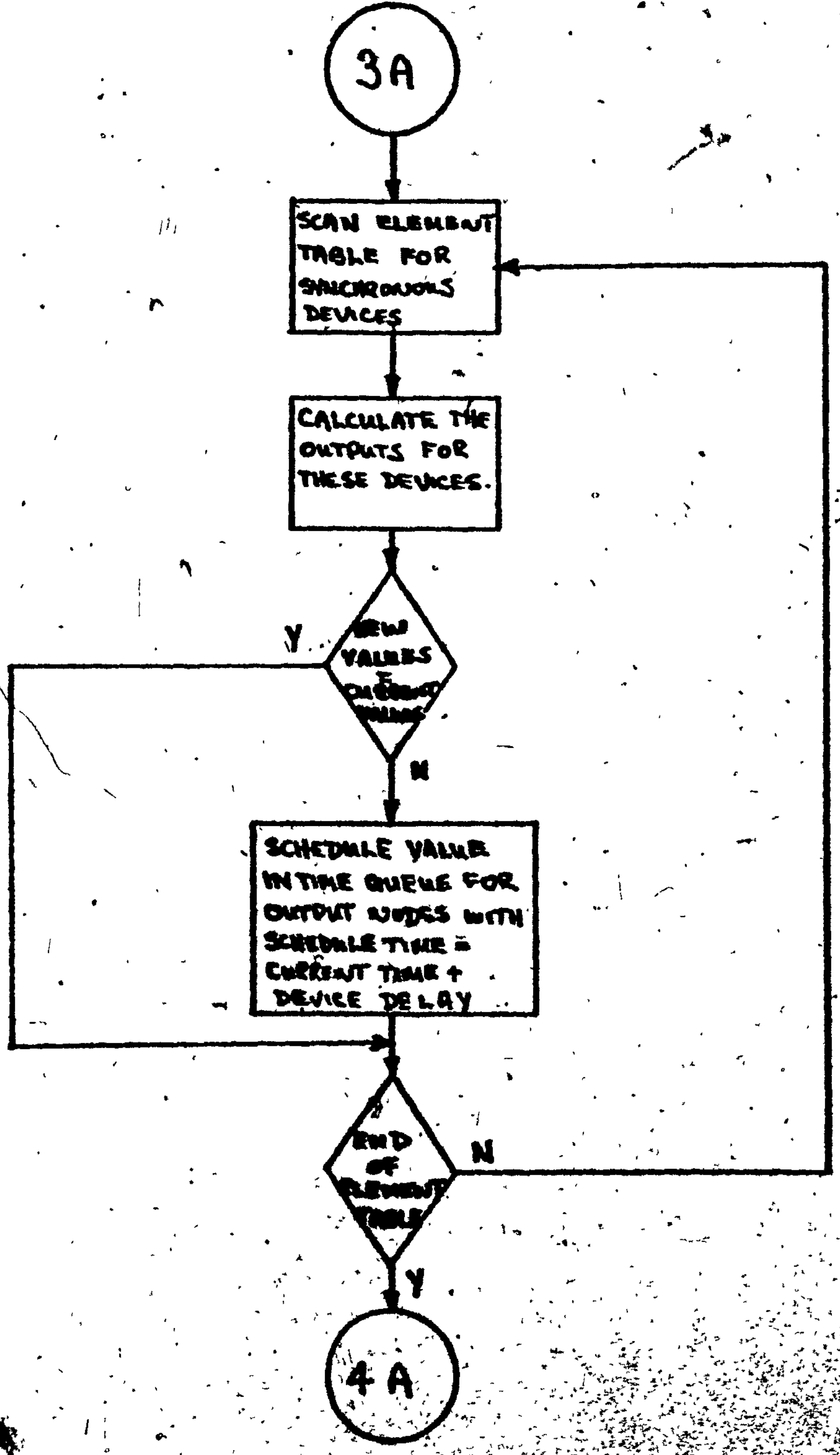

START
FETCH EXTERNAL INPUT VALUE FOR THIS CLOCK TIME
Y
MORE INPUTS LEFT
N
3A
Y
NEW VALUE = CURRENT
N
UPDATE NEW VALUE TO CURRENT VALUE TABLE
FETCH NEXT DEVICE POINTER FROM FANOUT TABLE
CALCULATE DEVICE OUTPUT
CHECK TIME-QUEUE POSITION FOR THIS OUTPUT NODE
ENTRY PRESENT ?
Y
2A
N
NEW VALUE = CURRENT VALUE
Y
N
1D
1B
SCHEDULE VALUE IN TIME QUEUE FOR THIS NODE WITH SCHEDULE TIME = CURRENT + DEVICE DELAY
ADD 1 TO SCHEDULE COUNT
1A
MORE ENTRIES IN FANOUT TABLE
Y
1C
N
NEW CLOCK CYCLE
Y
N
4B

DETAILED FLOW CHART OF SIMULATION ALGORITHM
FIGURE 2.4 PAGE 1 of 4


2A
CHECK SCHEDULE - TIME ENTRY
SCHEDULE TIME > CURRENT TIME AND ≤ CURRENT TIME + DEVICE DELAY
N
(ERROR)
1D
NEW CALCULATED VALUE = SCHEDULE VALUE
Y
DO NOT SCHEDULE NEW VALUE
N
NEW CALCULATED VALUE = CURRENT VALUE
N
1B
Y
DO NOT SCHEDULE NEW VALUE. REMOVE ENTRY ALREADY SCHED-ULED AND DECREASE SCHEDULE COUNT BY ONE.
1A

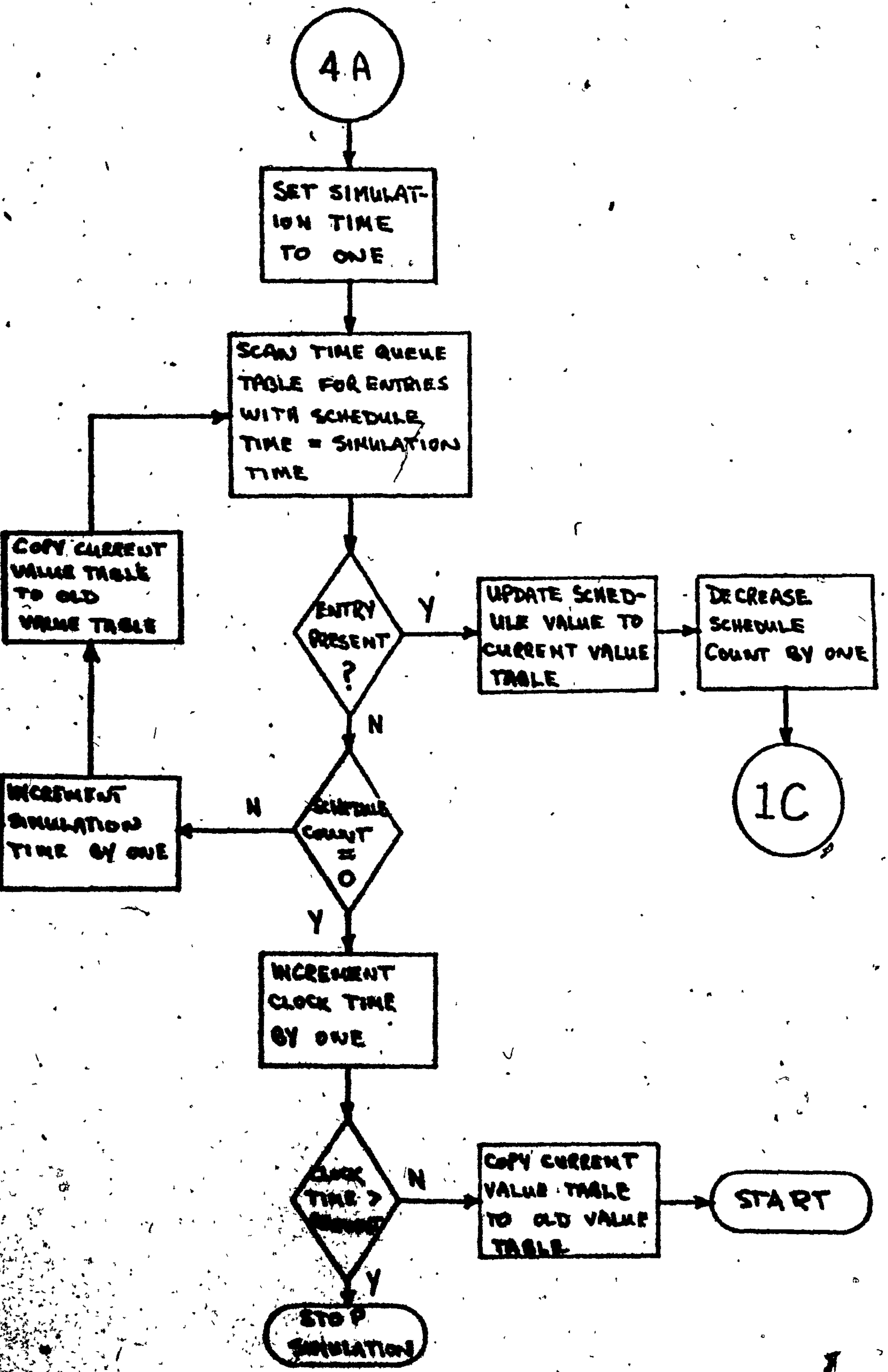
DETAILED FLOW CHART OF SIMULATION ALGORITHM

FIGURE 2.4 PAGE 2 of 4


3A
SCAN ELEMENT TABLE FOR SYNCHRONOUS DEVICES
CALCULATE THE OUTPUTS FOR THESE DEVICES.
NEW VALUES = CURRENT VALUES
Y
N
SCHEDULE VALUE IN TIME QUEUE FOR OUTPUT NODES WITH SCHEDULE TIME = CURRENT TIME + DEVICE DELAY
END OF ELEMENT TABLE
N
Y
4 A

DETAILED FLOW CHART OF SIMULATION [illegible]

FIGURE 2.4 [illegible]


4A
SET SIMULATION TIME TO ONE
SCAN TIME QUEUE TABLE FOR ENTRIES WITH SCHEDULE TIME = SIMULATION TIME
ENTRY PRESENT ?
Y
UPDATE SCHEDULE VALUE TO CURRENT VALUE TABLE
DECREASE SCHEDULE COUNT BY ONE
1C
N
SCHEDULE COUNT = 0
N
INCREMENT SIMULATION TIME BY ONE
COPY CURRENT VALUE TABLE TO OLD VALUE TABLE
Y
INCREMENT CLOCK TIME BY ONE
CLOCK TIME > [illegible]
N
COPY CURRENT VALUE TABLE TO OLD VALUE TABLE
START
Y
STOP SIMULATION

DETAILED FLOW CHART OF SIMULATION ALGORITHM
FIGURE 2.4 PAGE 4 of 4

CHAPTER 3

THE SIMULATION PROGRAM

3.0 - OUTLINE OF PROGRAM

3.0.1 - General

The program consists of a main section and seventeen subroutines. The seventeen subroutines consist of eight system subroutines and nine device evaluation subroutines.

3.0.2 - Main Section

This routine is the command level portion of the program. In this routine, the initialization subroutine (INITRN) is called to initialize all tables and variables. This routine will call the subroutines listed below when the command shown is entered.

Subroutine	Command
CIRBLD	CIR
EXTRTN	EXT
PARRTN	PAR
PRTRTN	POP
NEWCIR	NCT
SIMRUN	SIM (SNI)
NDVRTN	NDEV
EDITRN	EDIT
CORRTN	CORR

The command END will cause the program to terminate.

3.0.3 - Subroutine CIRBLD

This routine is called by the main section when the command "CIR" is entered. This routine is responsible for accepting the input circuit description, validating the device type, and building up the ELEMENT (circuit description) table, the device FANOUT table and the NAME table. The names of the nodes are stored in the NAME table and pointers are placed in the Element table to point to the respective node names. A pointer to the device evaluation routine is also inserted in the Element table. The Fanout table is parallel to the Name table and contains pointers to the devices in the Element table which are connected to this node.

Two subcommands, "EDIT" and "CORR," call in subroutines EDITRN and CORRTN respectively.

The command "*END" can be entered at any point during the circuit build to exit from the routine back to command level.

3.0.4 - Subroutine EXTRTN

This subroutine builds up the external input table. This routine asks for the names of all external inputs to the circuit and their values for each clock time. The routine then builds up a table for 50 clock times maximum. This routine is called by the command "EXT". Return is made automatically to the command level after all input nodal values have been specified. The routine can also be exited by entering "*END" during the entering of input values for the nodes specified. In this case, all pointers in the EXTERNAL VALUE table are reset and return made to the system command level.

3.0.5 - Subroutine PARRTN

The command "PAR" will call subroutine PARRTN. This subroutine allows the delay time and number of inputs for the devices simulated by the program to be changed. The maximum number of inputs allowed is five and the largest time delay allowed is 99.

3.0.6 - Subroutine PRTRTN

The print options subroutine (PRTRTN) is called when the command "POP" is entered. This routine is used to set up the user's print-out requirements. In response to directives from the program, the user specifies whether he wishes the output to be printed every clock time or every unit delay time. The user can also specify that printed output is to occur only when certain nodes have a specified value. The user can specify up to ten nodes with their respective values so that output will only be printed when one of these nodes has the value indicated. The user can also specify which nodes are to be monitored and printed. Up to eighteen nodes can be specified. The default values are the first eighteen nodes listed in the NAME table. In response to the directive to enter the names of the nodes to be printed, the user can either enter up to 18 node names, *END or ALL. The latter two responses result in the default conditions being set up. If no errors have occurred on inputting the node names (e.g. specifying a node that does not exist), return is made automatically to the command level. If an input error has occurred, all the node names must be entered again.

3.0.7 - Subroutine NEWCIR

This is not a subroutine like the others. When the command "NCT" is entered, the initialization subroutine (INITRN) is called to reset all variables and initialize all tables and arrays. A transfer is then made to the circuit build subroutine (CIRBLD) to allow the new circuit to be entered.

3.0.8 - Subroutine SIMRUN

This routine is called when the command "SIM" or "SNI" is entered. This is the routine which simulates the circuit for the number of clock times specified. If no external inputs or input values have been specified, the subroutine displays the appropriate message and control is returned to the command routine. The circuit is initialized by setting all node values to X and all external nodes to 0 and then looping through the circuit until no further elements change state. The user may specify initial settings of external nodes before the circuit is initialized. Use of the command "SNI" will allow the user to re-enter the simulation subroutine without the circuit being initialized. This command would be used when the user wishes to enter in a new sequence of clock pulses and continue to simulate the circuit without resetting the current circuit state. However, the circuit must be initialized the first time it is simulated. An attempt to use the command "SNI" on a new circuit which had not previously been initialized will result in the circuit being initialized anyway.

This routine calls in subroutines: NANDRTN, NORRTN, DRVRTN, AORRTN, XORRTN, JKRTN, DFFRTN, RSRTN, and USERTN to simulate NAND/AND gates, NOR/OR gates, drivers, and-or-inverter gates, exclusive-or gates, J-K flip flop, D flip flops, R-S flip flops and user-defined-devices respectively.

After the required number of clock pulses has been simulated, two commands, "CONT" and "REP" can be entered. CONT will allow the simulation to continue for a specified number of clock pulses until 50 is reached. REP will repeat the simulation with the previous input values but without initializing the circuit beforehand.

*END will cause the subroutine to be terminated and control returned to command level.

Another subroutine PRTLIN is called as needed to print the values of the requested nodes.

3.0.9 - Subroutine EDITRN

This routine can either be called at system command level or in the circuit build mode by entering the command "EDIT".

At system command level, the subroutine will ask what function is to be performed; i.e. list the Fanout for each device or print the circuit description from the Element table. Two responses, "CIR" and "FAN", will call in the appropriate function.

When called from circuit build mode, this subroutine will list out the contents of the Element table only.

3.0.10 - Subroutine CORRTN

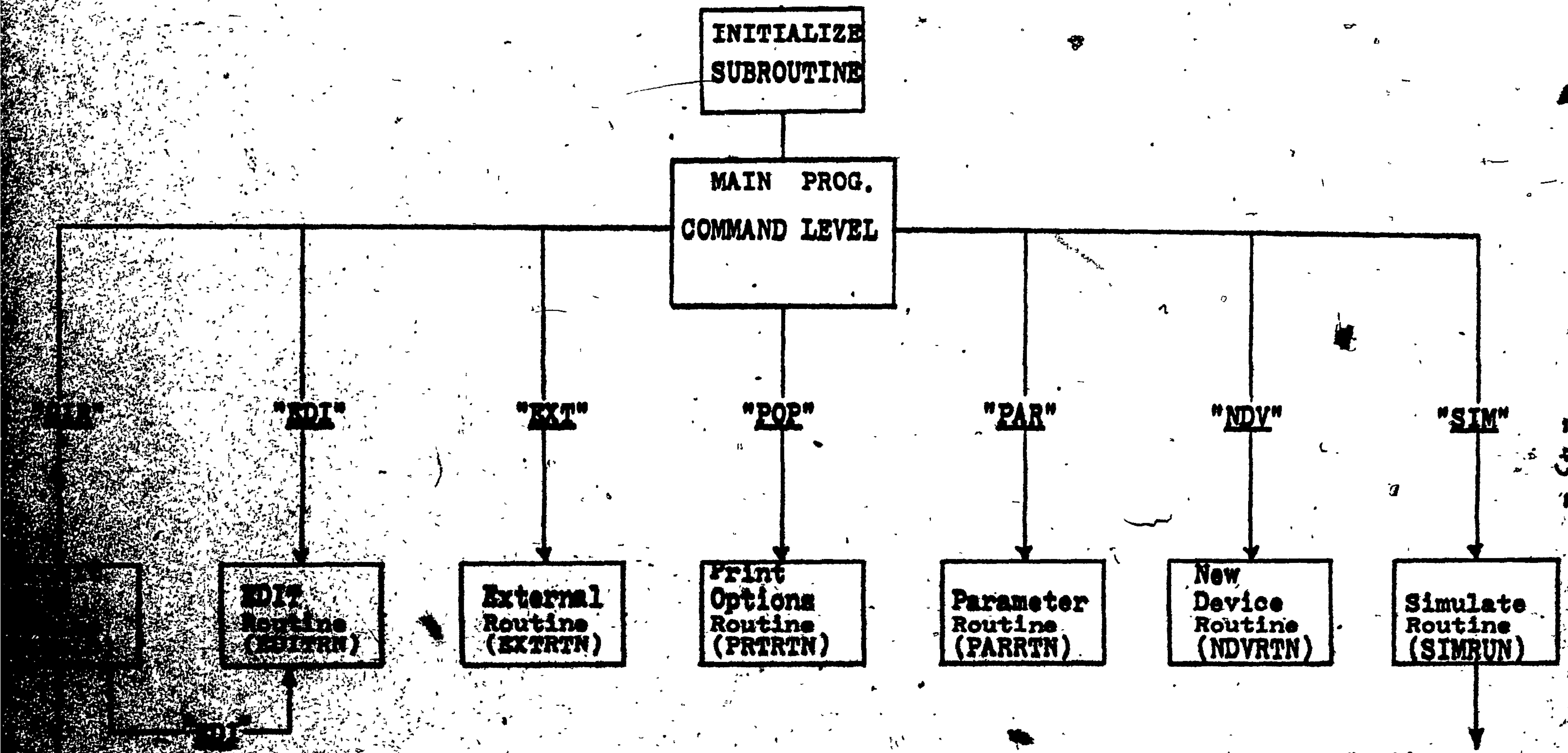
This subroutine is called only from the circuit build subroutine (CIRBLD). This routine allows corrections to be made to the circuit description table. No deletions of device entries in the table are allowed, only corrections or replacements to existing entries. The FANOUT table is also updated to reflect the changes in the circuit description. No changes are made to the NAME table.

3.0.11 - Subroutine NDEVRTN

This routine is called when the command "NDEV" is entered. This subroutine allows a user-defined device to be entered into the device table. The number of inputs (maximum 5) and outputs (maximum 2) are entered along with the device name. The device delay and truth table are also entered, along with information as to whether it is an edge-sensitive device (i.e. a flip flop) or a gate. This device remains in the device table for the duration of the program. The description of any device can be replaced or changed as well in this routine.

3.1 - STRUCTURE OF THE PROGRAM

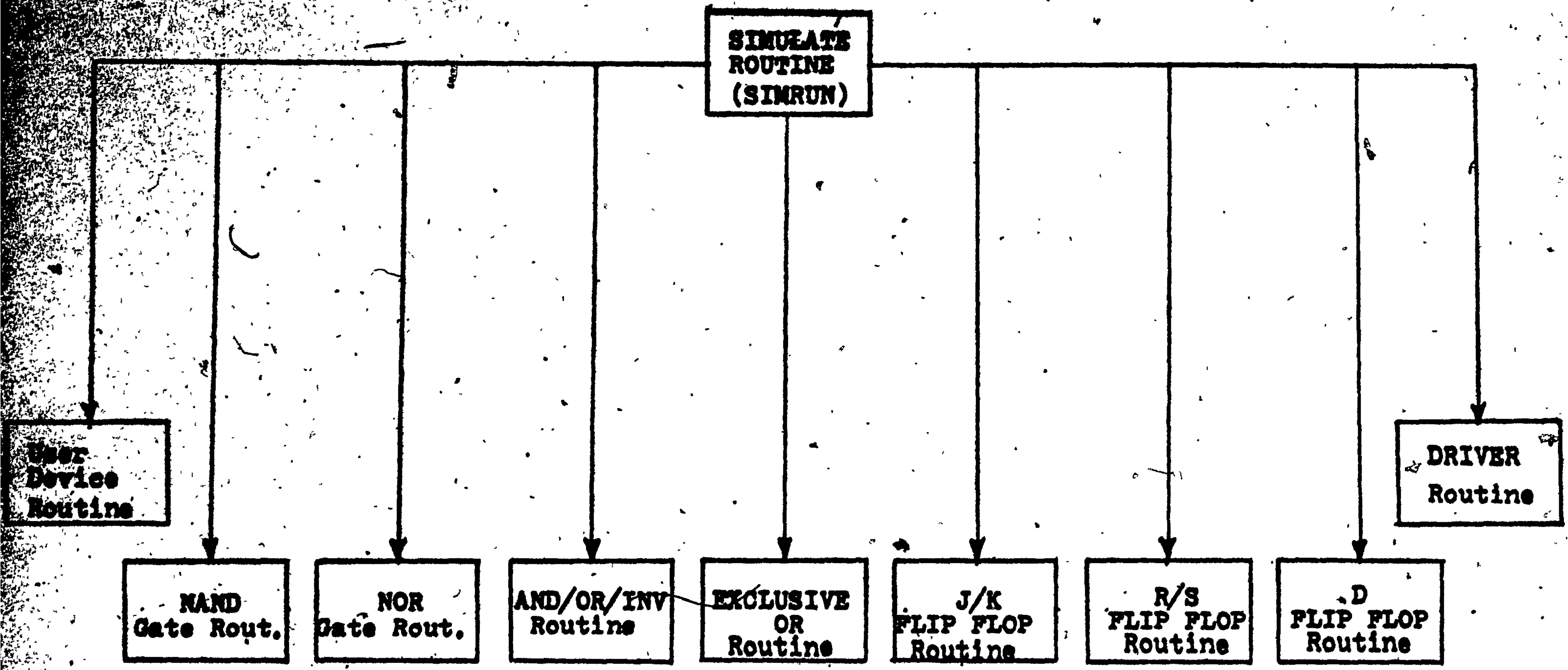
The subroutine structure of the program is shown in Figure 3.0. Flow Charts for all the routines described in Paragraph 3.0 are shown in Figures 3.1 to 3.10.


INITIALIZE SUBROUTINE
MAIN PROG. COMMAND LEVEL
"EDI"
"EXT"
"POP"
"PAR"
"NDV"
"SIM"
EDIT Routine (EDITRTN)
External Routine (EXTRTN)
Print Options Routine (PRTRTN)
Parameter Routine (PARRTN)
New Device Routine (NDVRTN)
Simulate Routine (SIMRUN)
"EDI"

Continued on following page

STRUCTURE OF PROGRAM
FIG. 3.0

PAGE 1 of 2.



STRUCTURE OF PROGRAM

FIG. 3.0 (Cont.)

Page 2 of 2.

MAIN
"COMMAND LEVEL"

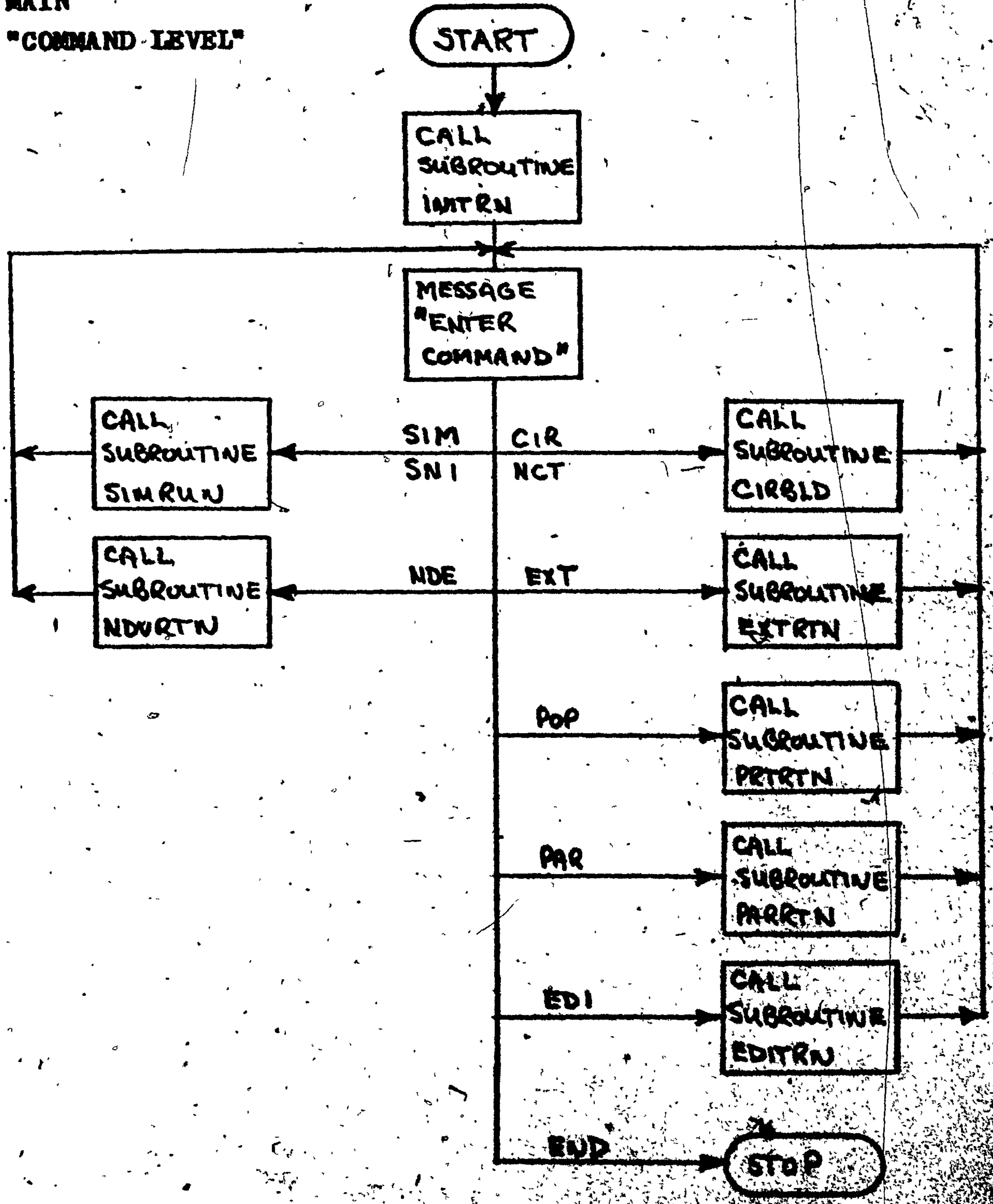


FIGURE 3.1 Page 1 of 1

CIRCUIT BUILD
SUBROUTINE
"CIRBLD"

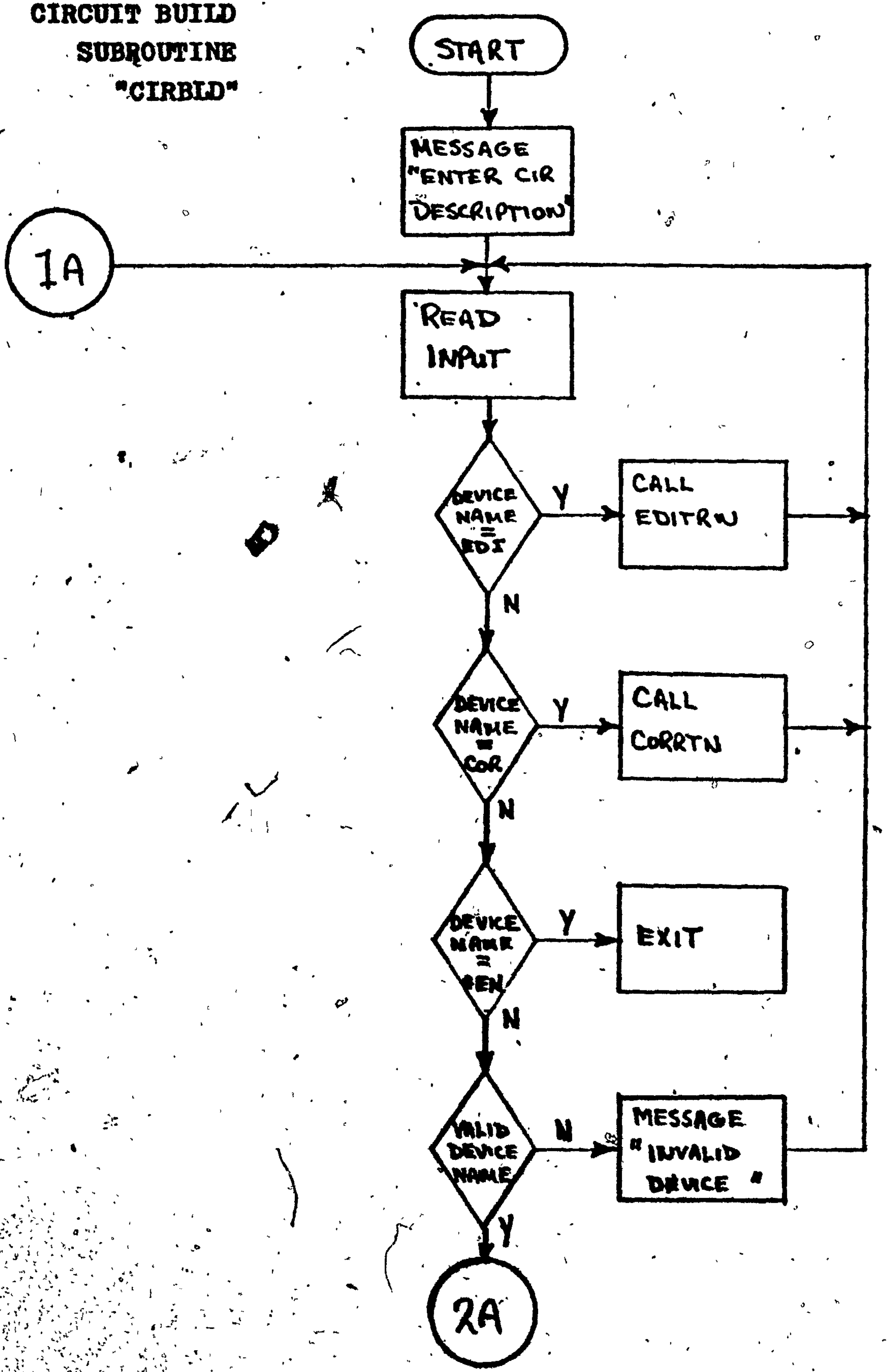


FIGURE 3.2

Page 1 of 2

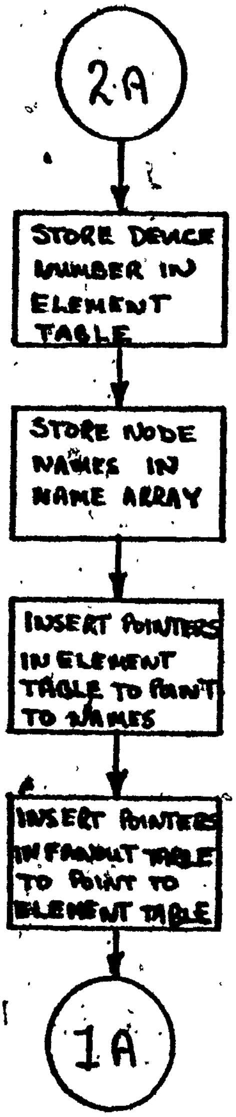

2A
STORE DEVICE NUMBER IN ELEMENT TABLE
STORE NODE NAMES IN NAME ARRAY
INSERT POINTERS IN ELEMENT TABLE TO POINT TO NAMES
INSERT POINTERS IN FANOUT TABLE TO POINT TO ELEMENT TABLE
1A

FIGURE 3.2 (Cont.) Page 2 of 2

EXTERNAL SIGNAL
ROUTINE
"EXTRTN"

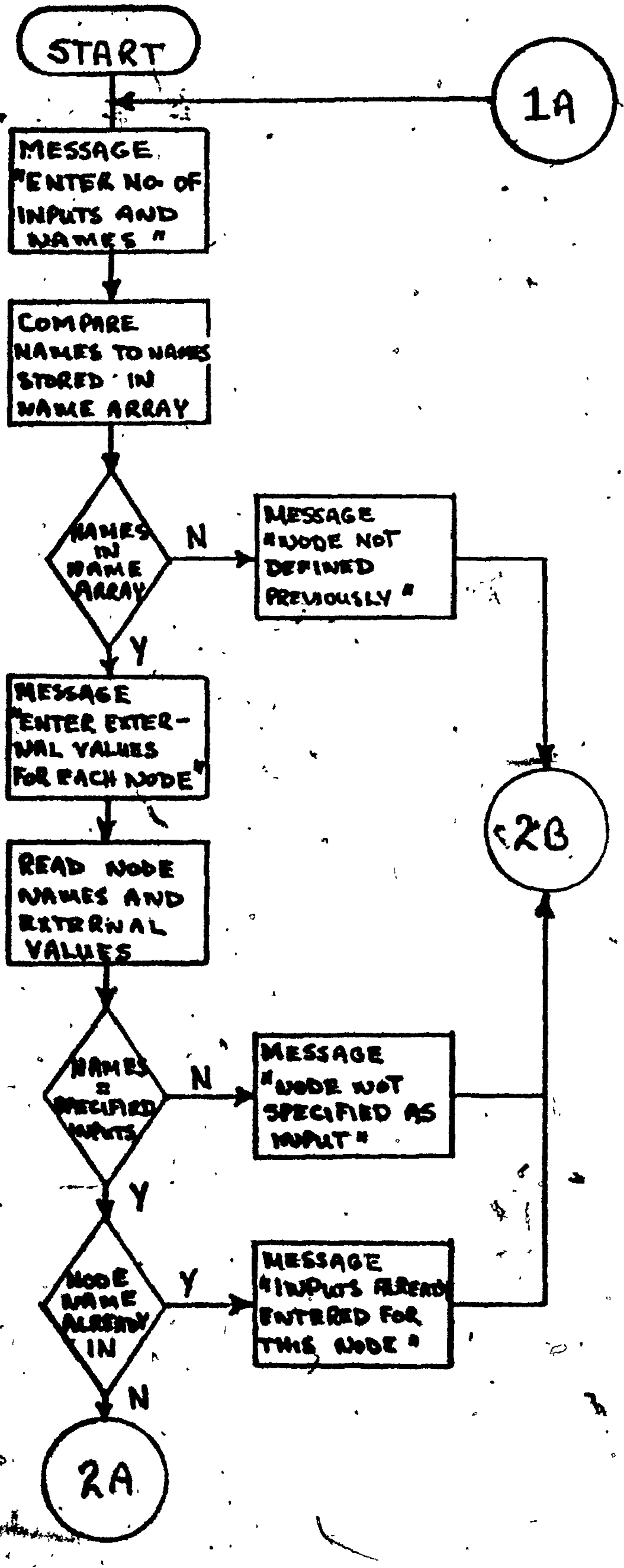

START
1A
MESSAGE "ENTER No. OF INPUTS AND NAMES"
COMPARE NAMES TO NAMES STORED IN NAME ARRAY
NAMES IN NAME ARRAY
N
MESSAGE "NODE NOT DEFINED PREVIOUSLY"
Y
MESSAGE "ENTER EXTERNAL VALUES FOR EACH NODE"
2B
READ NODE NAMES AND EXTERNAL VALUES
NAMES = SPECIFIED INPUTS
N
MESSAGE "NODE NOT SPECIFIED AS INPUT"
Y
NODE NAME ALREADY IN
Y
MESSAGE "INPUTS ALREADY ENTERED FOR THIS NODE"
N
2A

FIGURE 3.3

Page 1 of 2

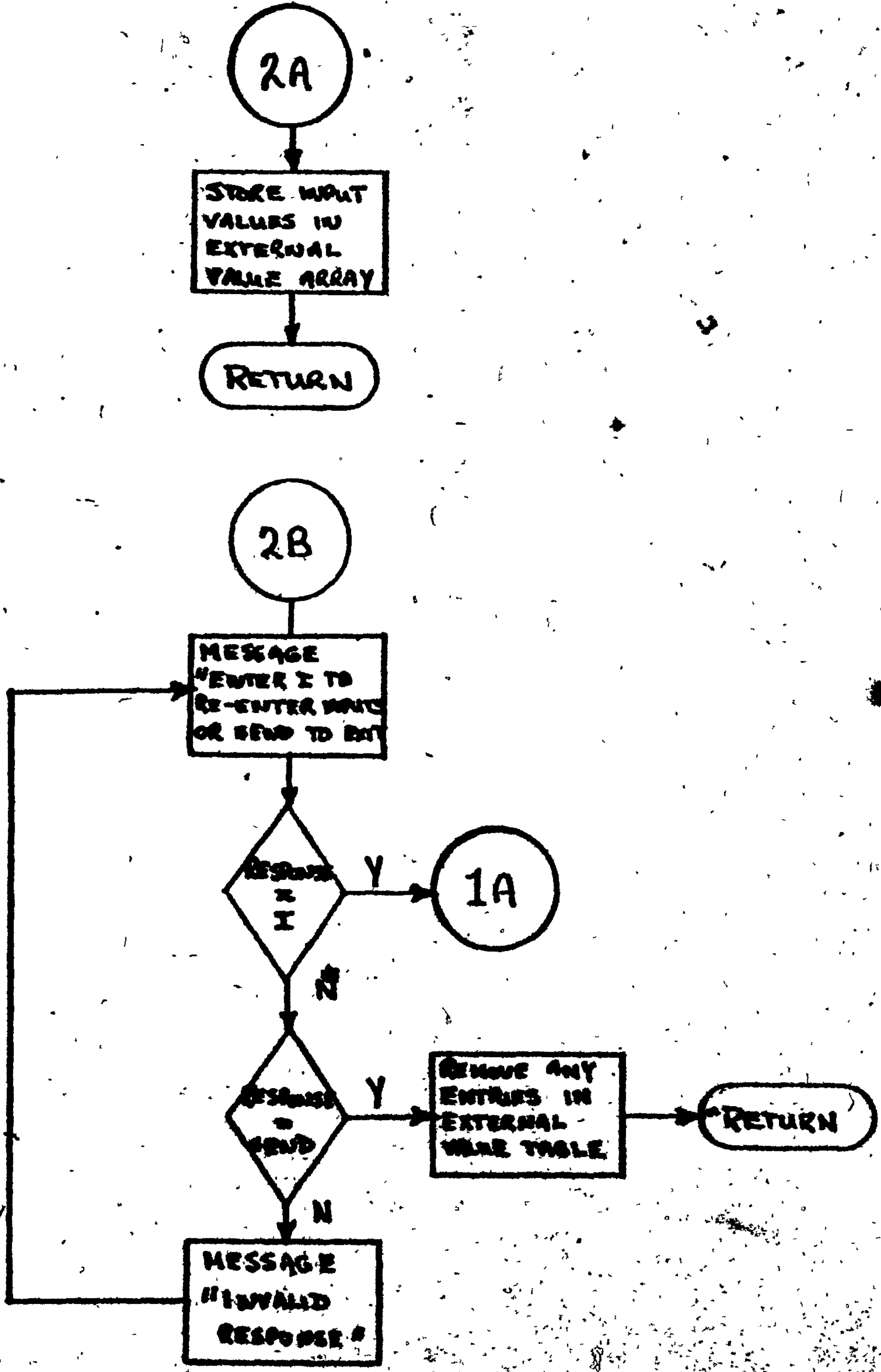

2A
STORE INPUT VALUES IN EXTERNAL VALUE ARRAY
RETURN
2B
MESSAGE "ENTER I TO RE-ENTER UNITS OR SEND TO EXIT
RESPONSE = I
Y
1A
N
RESPONSE = SEND
Y
REMOVE ANY ENTRIES IN EXTERNAL VALUE TABLE
RETURN
N
MESSAGE "INVALID RESPONSE"

FIGURE 3.3 (Cont.) Page 2 of 2

CHANGE PARAMETERS SUBROUTINE "PARRTN"

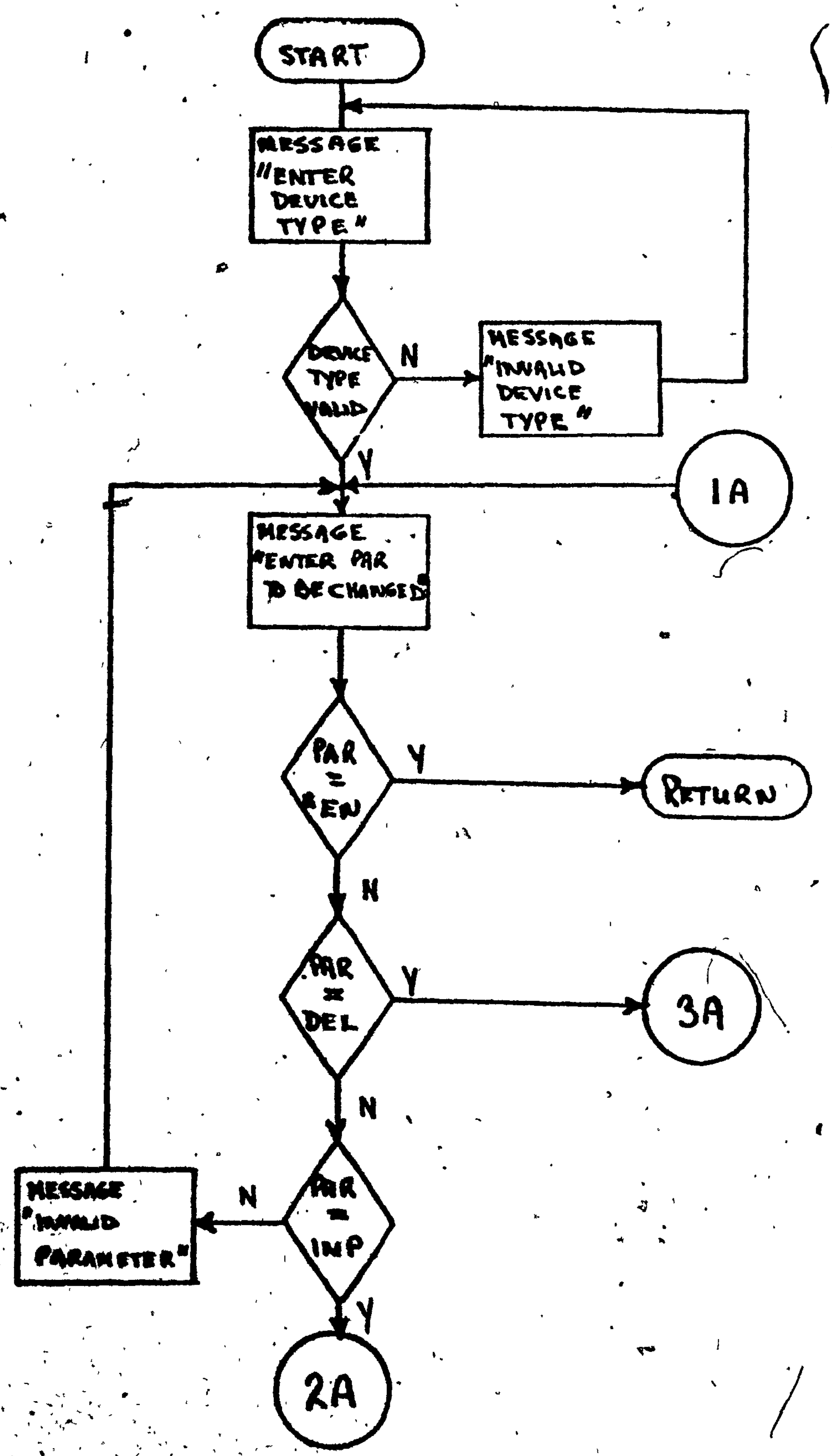

START
MESSAGE "ENTER DEVICE TYPE"
DEVICE TYPE VALID
N
MESSAGE "INVALID DEVICE TYPE"
Y
1A
MESSAGE "ENTER PAR TO BE CHANGED"
PAR = $EN
Y
RETURN
N
PAR = DEL
Y
3A
N
PAR = INP
N
MESSAGE "INVALID PARAMETER"
Y
2A

FIGURE 3.4

Page 1 of 3

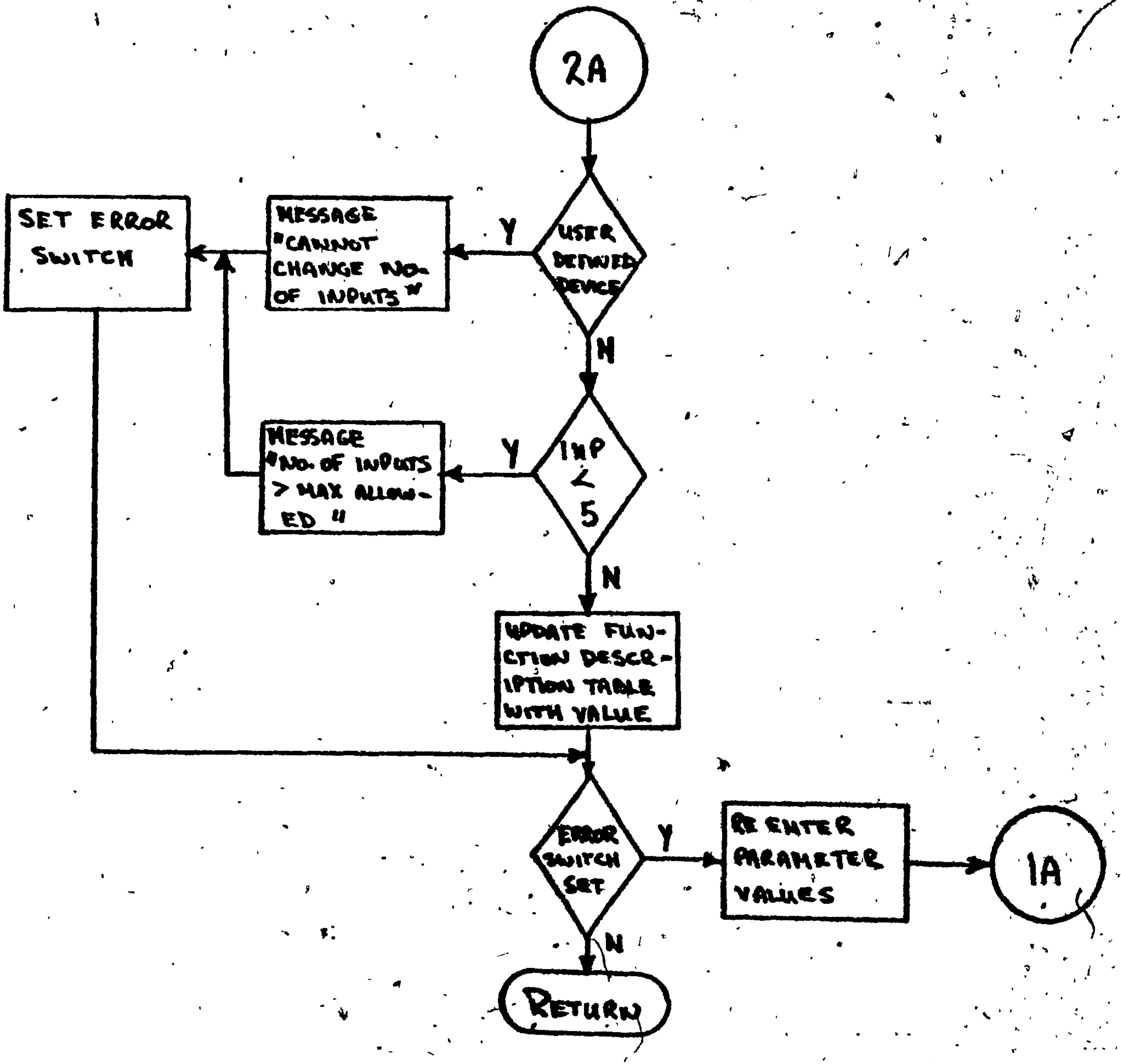

2A
USER DEFINED DEVICE
Y
N
MESSAGE "CANNOT CHANGE NO. OF INPUTS"
SET ERROR SWITCH
INP < 5
Y
N
MESSAGE "NO. OF INPUTS > MAX ALLOWED"
UPDATE FUNCTION DESCRIPTION TABLE WITH VALUE
ERROR SWITCH SET
Y
N
RE ENTER PARAMETER VALUES
1A
RETURN

FIGURE 3.4 (Cont.) Page 2 of 3

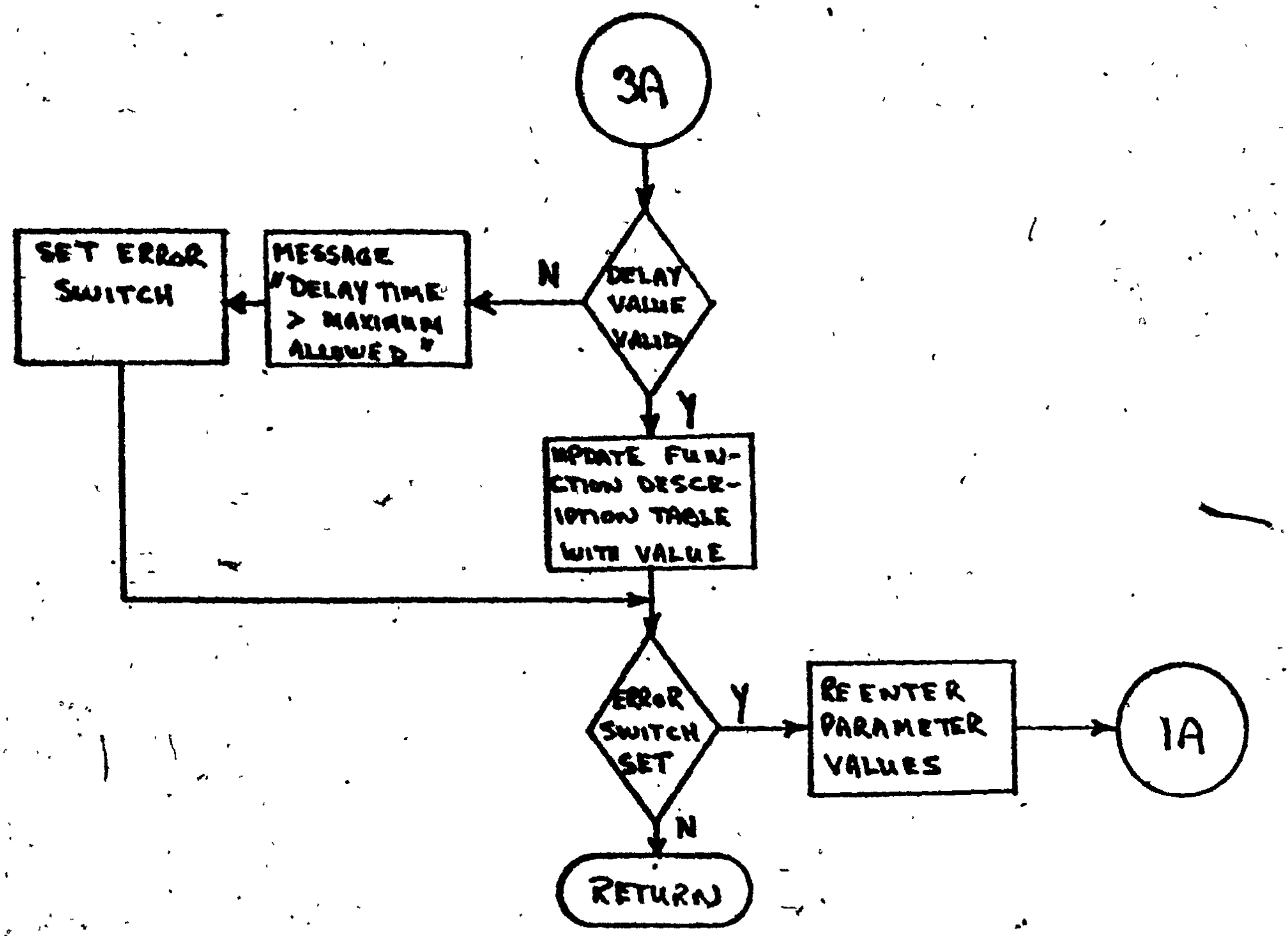


FIGURE 3.4 (Cont.)   Page 3 of 3

PRINT OPTIONS ROUTINE
"PRTRTN"

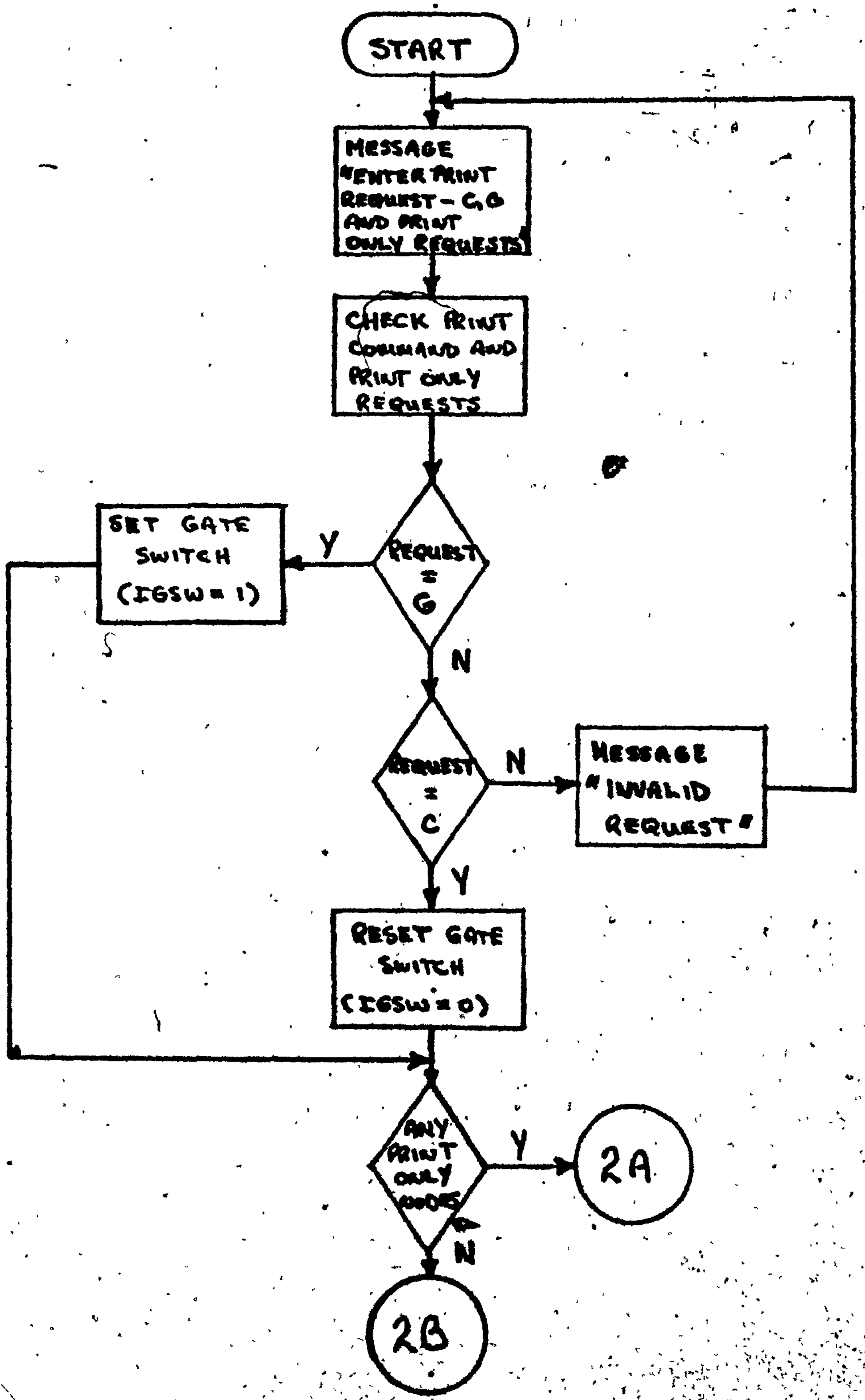

START
MESSAGE "ENTER PRINT REQUEST - C,G AND PRINT ONLY REQUESTS"
CHECK PRINT COMMAND AND PRINT ONLY REQUESTS
REQUEST = G
Y
SET GATE SWITCH (IGSW = 1)
N
REQUEST = C
N
MESSAGE "INVALID REQUEST"
Y
RESET GATE SWITCH (IGSW = 0)
ANY PRINT ONLY NODES
Y
2A
N
2B

FIGURE 3.5

Page 1 of 3

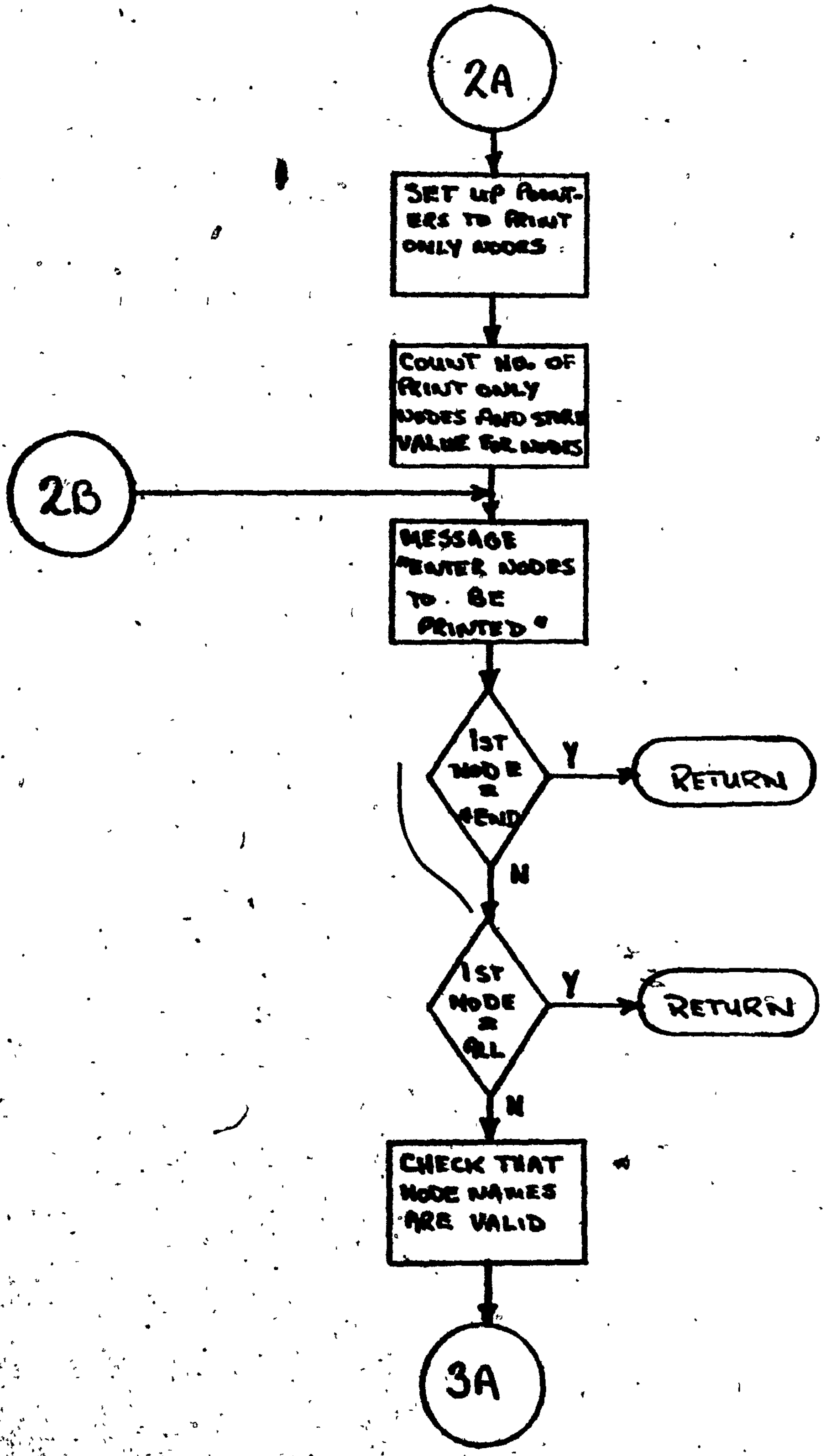

2A
SET UP POINTERS TO PRINT ONLY NODES
COUNT NO. OF PRINT ONLY NODES AND STORE VALUE FOR NODES
2B
MESSAGE "ENTER NODES TO BE PRINTED"
1ST NODE = #END
Y
RETURN
N
1ST NODE = ALL
Y
RETURN
N
CHECK THAT NODE NAMES ARE VALID
3A

FIGURE 3.5 (Cont.) Page 2 of 3

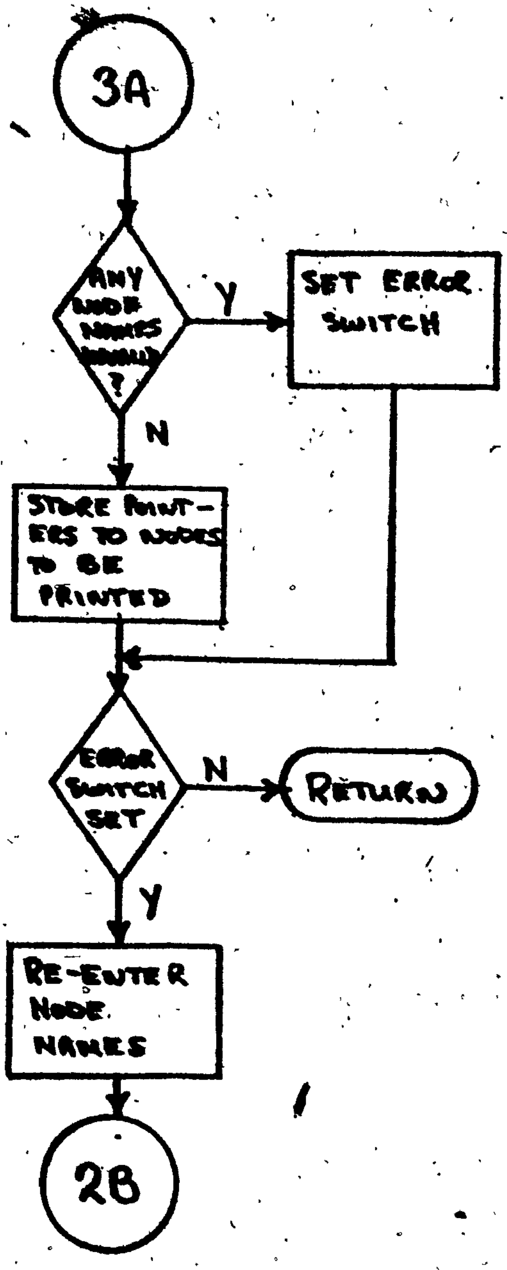


FIGURE 3.5 (Cont.) Page 3 of 3

SIMULATE SUBROUTINE, "SIMRUN"

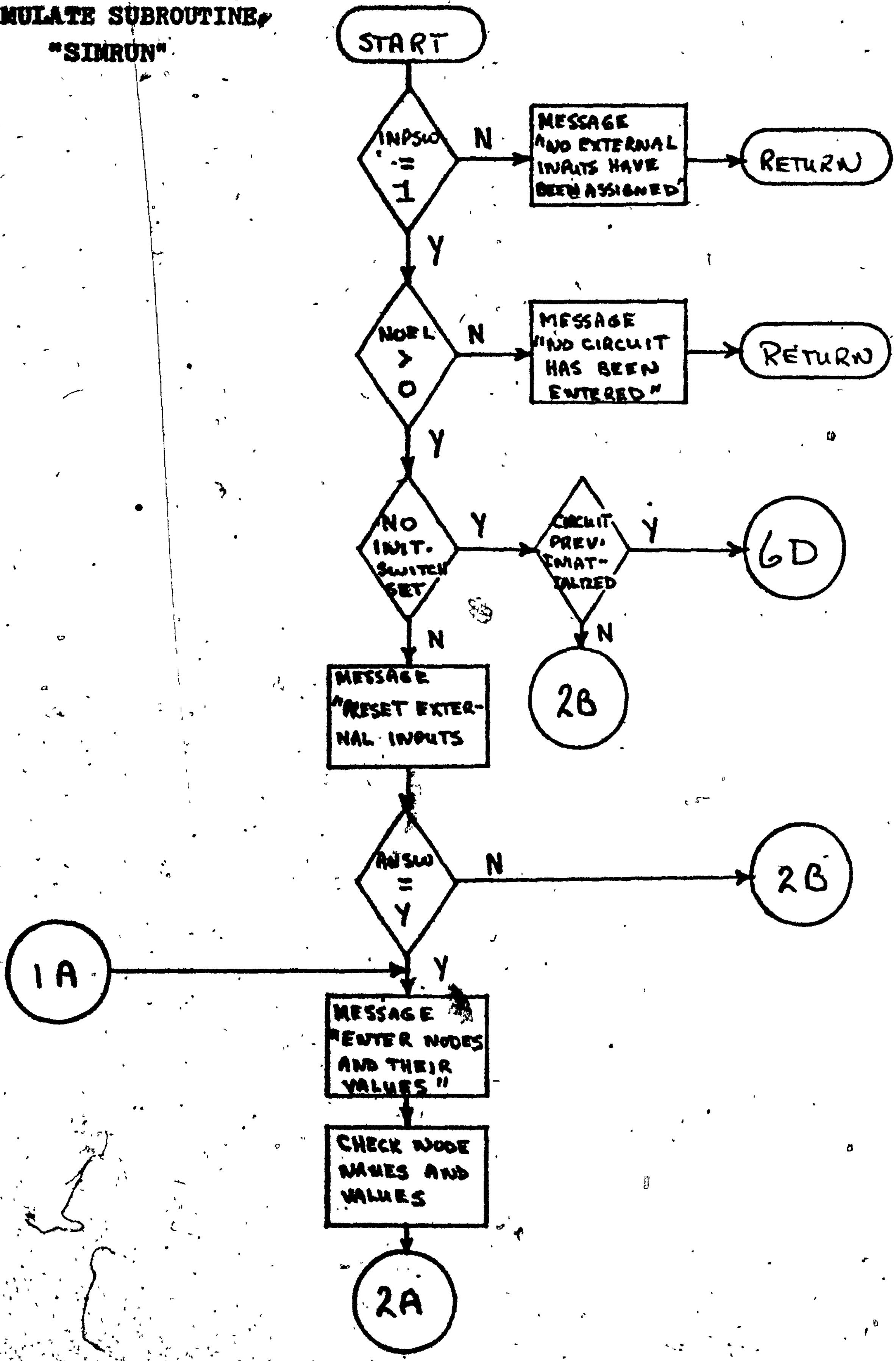

START
INPSW = 1
N
MESSAGE "NO EXTERNAL INPUTS HAVE BEEN ASSIGNED"
RETURN
Y
NOEL > 0
N
MESSAGE "NO CIRCUIT HAS BEEN ENTERED"
RETURN
Y
NO INIT. SWITCH SET
Y
CIRCUIT PREVI INITIAL-IALIZED
Y
6D
N
2B
N
MESSAGE "RESET EXTERNAL INPUTS
ANSW = Y
N
2B
1A
Y
MESSAGE "ENTER NODES AND THEIR VALUES"
CHECK NODE NAMES AND VALUES
2A

FIGURE 3.6

Page 1 of 10

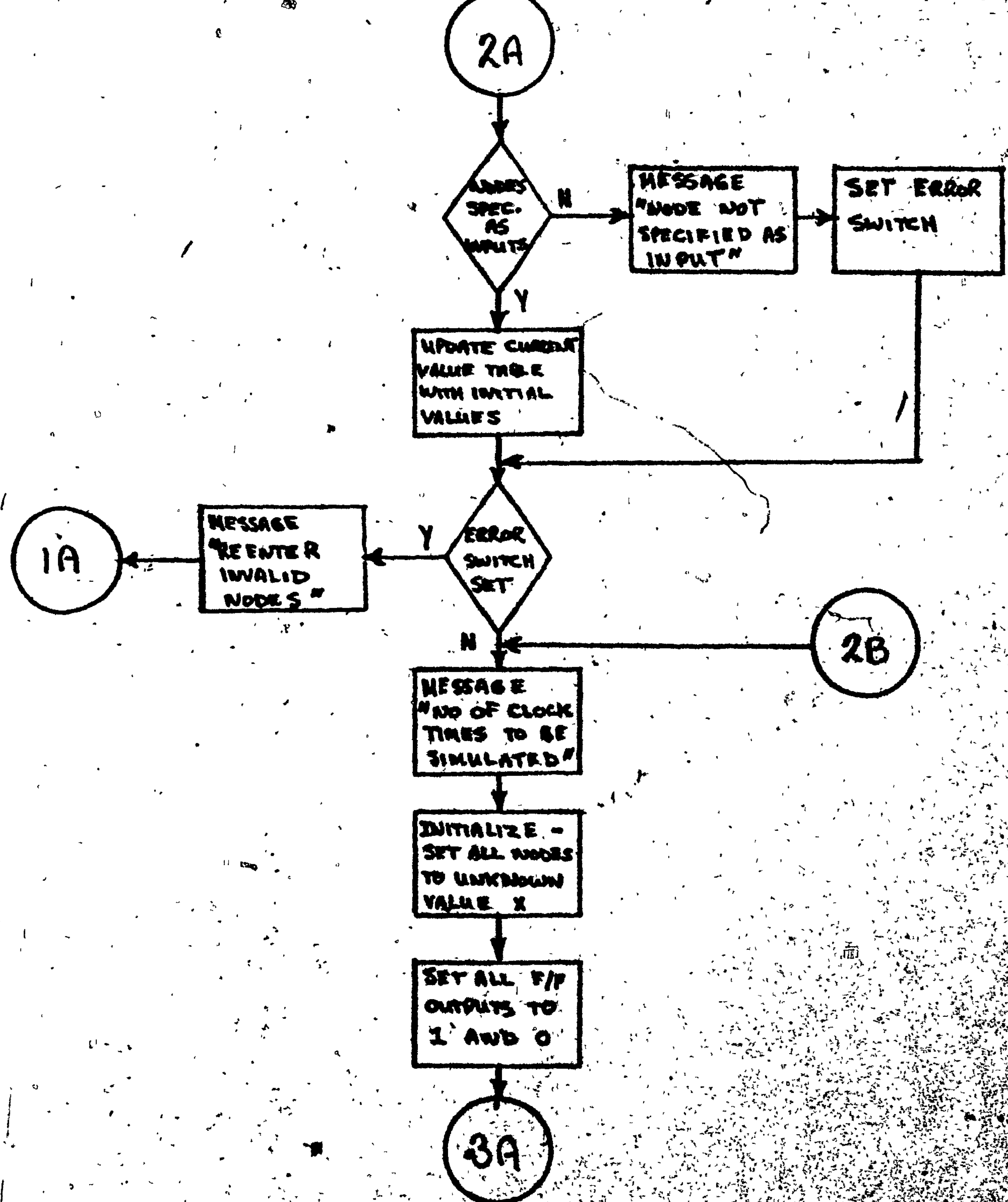

2A
NODES SPEC. AS INPUTS
N
MESSAGE "NODE NOT SPECIFIED AS INPUT"
SET ERROR SWITCH
Y
UPDATE CURRENT VALUE TABLE WITH INITIAL VALUES
ERROR SWITCH SET
Y
MESSAGE "REENTER INVALID NODES"
1A
N
2B
MESSAGE "NO OF CLOCK TIMES TO BE SIMULATED"
INITIALIZE - SET ALL NODES TO UNKNOWN VALUE X
SET ALL F/F OUTPUTS TO 1 AND 0
3A

FIGURE 3.6 (Cont.)

Page 2 of 10

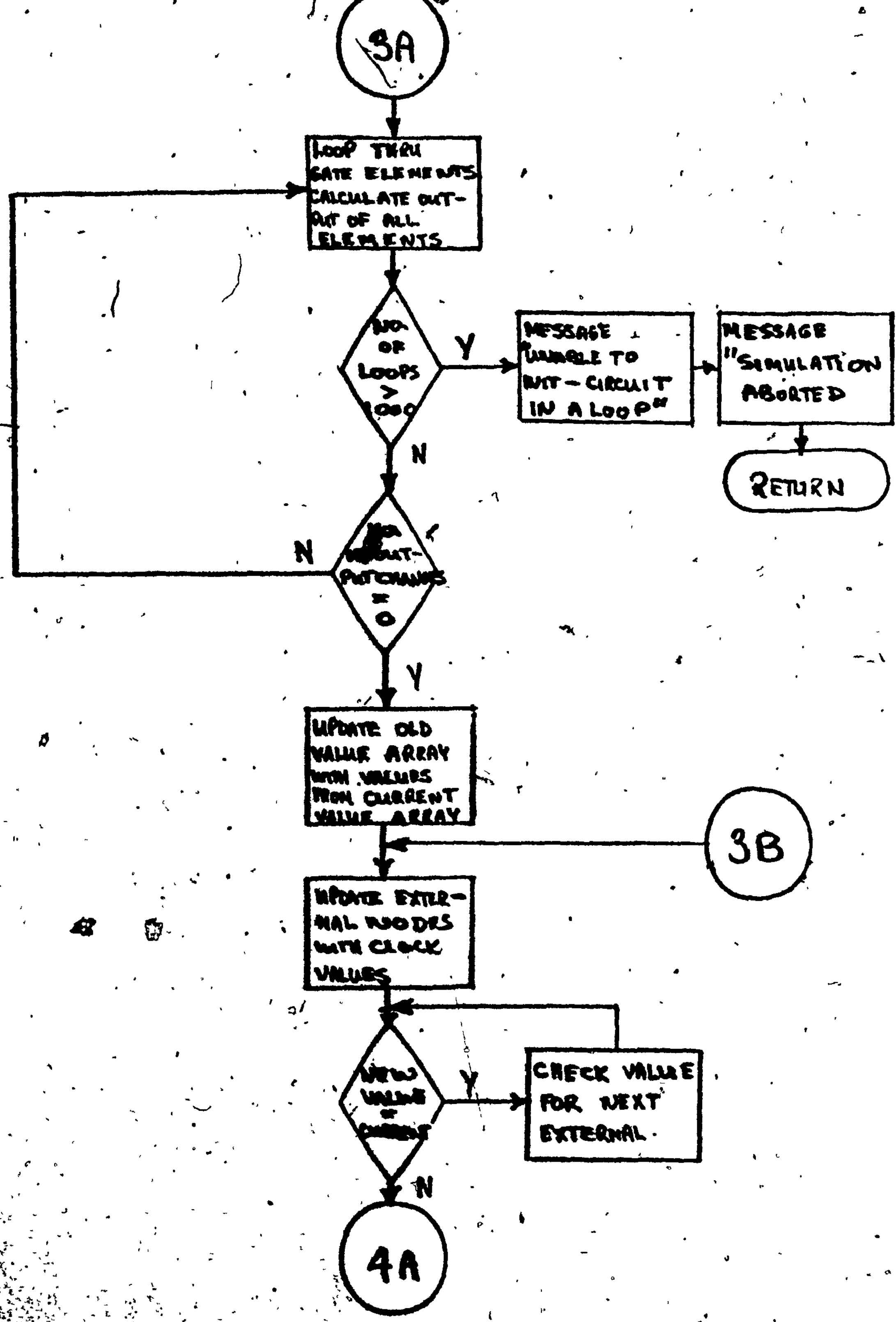

3A
LOOP THRU GATE ELEMENTS CALCULATE OUT-PUT OF ALL ELEMENTS
NO. OF LOOPS > 1000
Y
MESSAGE "UNABLE TO INIT-CIRCUIT IN A LOOP"
MESSAGE "SIMULATION ABORTED
RETURN
N
NO. OF OUT-PUT CHANGES = 0
N
Y
UPDATE OLD VALUE ARRAY WITH VALUES FROM CURRENT VALUE ARRAY
3B
UPDATE EXTER-NAL NODES WITH CLOCK VALUES
NEW VALUE = CURRENT
Y
CHECK VALUE FOR NEXT EXTERNAL
N
4A

FIGURE 3.6 (Cont.)   Page 3 of 10

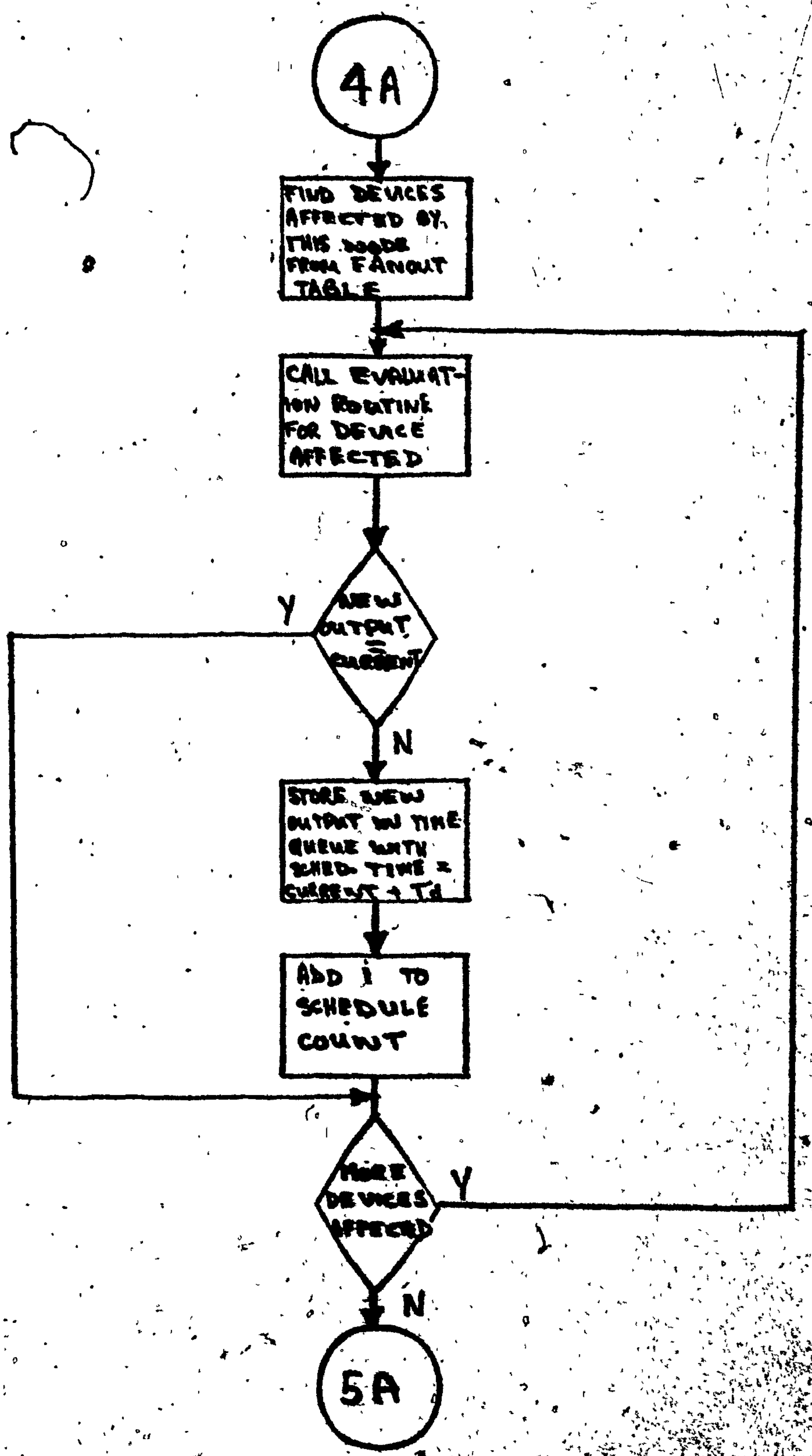

4A
FIND DEVICES AFFECTED BY THIS NODE FROM FANOUT TABLE
CALL EVALUATION ROUTINE FOR DEVICE AFFECTED
NEW OUTPUT = CURRENT
Y
N
STORE NEW OUTPUT IN TIME QUEUE WITH SCHED. TIME = CURRENT + Td
ADD 1 TO SCHEDULE COUNT
MORE DEVICES AFFECTED
Y
N
5A

FIGURE 3.6 (Cont.) Page [illegible] of 10

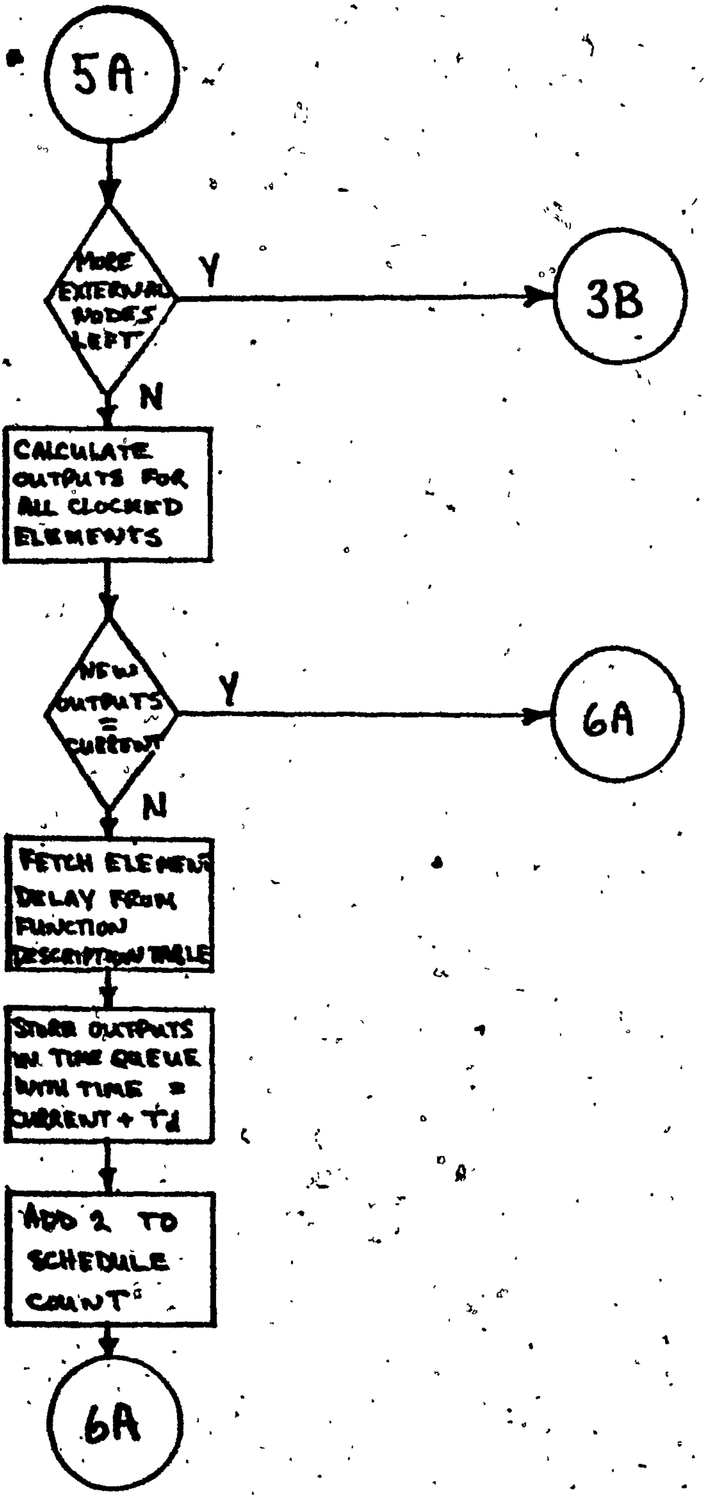

5A
MORE EXTERNAL NODES LEFT
Y
3B
N
CALCULATE OUTPUTS FOR ALL CLOCKED ELEMENTS
NEW OUTPUTS = CURRENT
Y
6A
N
FETCH ELEMENT DELAY FROM FUNCTION DESCRIPTION TABLE
STORE OUTPUTS IN TIME QUEUE WITH TIME = CURRENT + Td
ADD 2 TO SCHEDULE COUNT
6A

FIGURE 3.6 (Cont.) Page 5 of 10

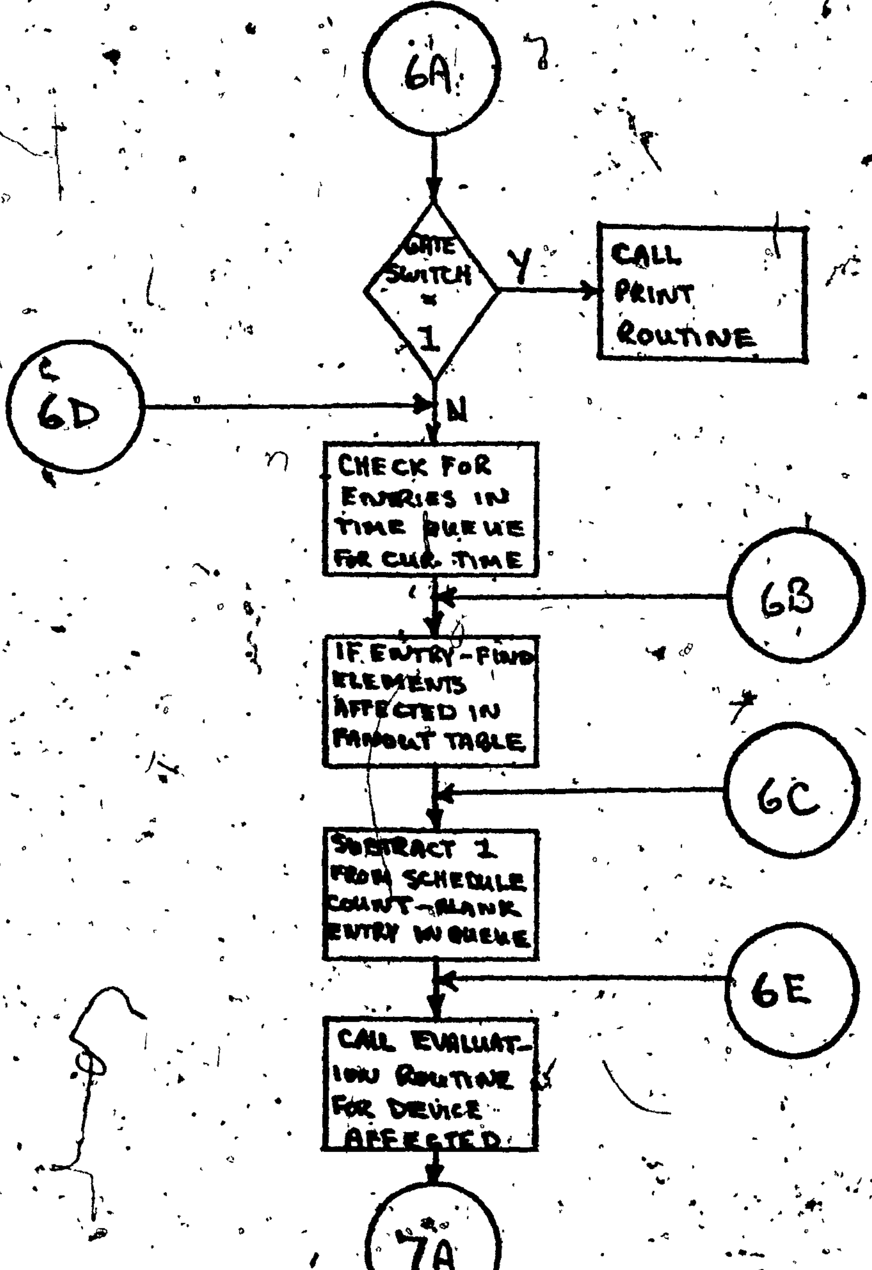

6A
GATE SWITCH = 1
Y
CALL PRINT ROUTINE
6D
N
CHECK FOR ENTRIES IN TIME QUEUE FOR CUR. TIME
6B
IF ENTRY-FIND ELEMENTS AFFECTED IN FANOUT TABLE
6C
SUBTRACT 1 FROM SCHEDULE COUNT-BLANK ENTRY IN QUEUE
6E
CALL EVALUAT-ION ROUTINE FOR DEVICE AFFECTED
7A

FIGURE 3.6 (Cont.) Page 6 of 10

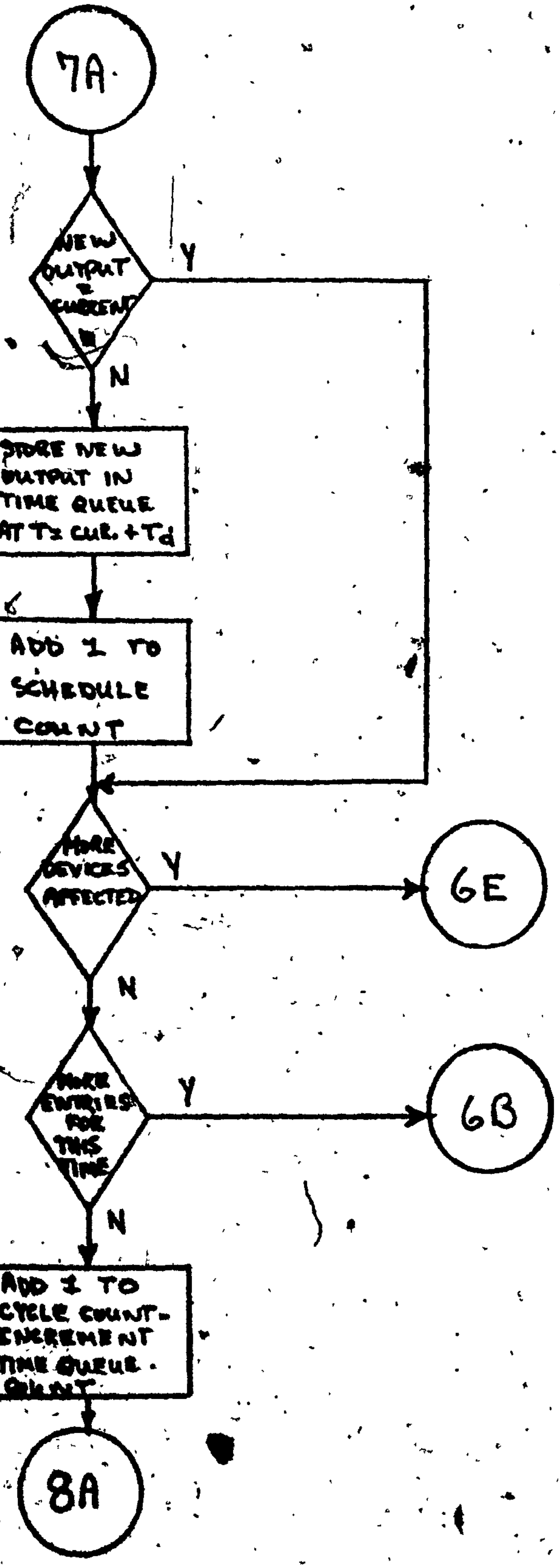

7A
NEW OUTPUT = CURRENT
Y
N
STORE NEW OUTPUT IN TIME QUEUE AT T= CUR. + Td
ADD 1 TO SCHEDULE COUNT
MORE DEVICES AFFECTED
Y
6E
N
MORE ENTRIES FOR THIS TIME
Y
6B
N
ADD 1 TO CYCLE COUNT- INCREMENT TIME QUEUE COUNT
8A

FIGURE 3.6 (Cont.) Page 7 of 10

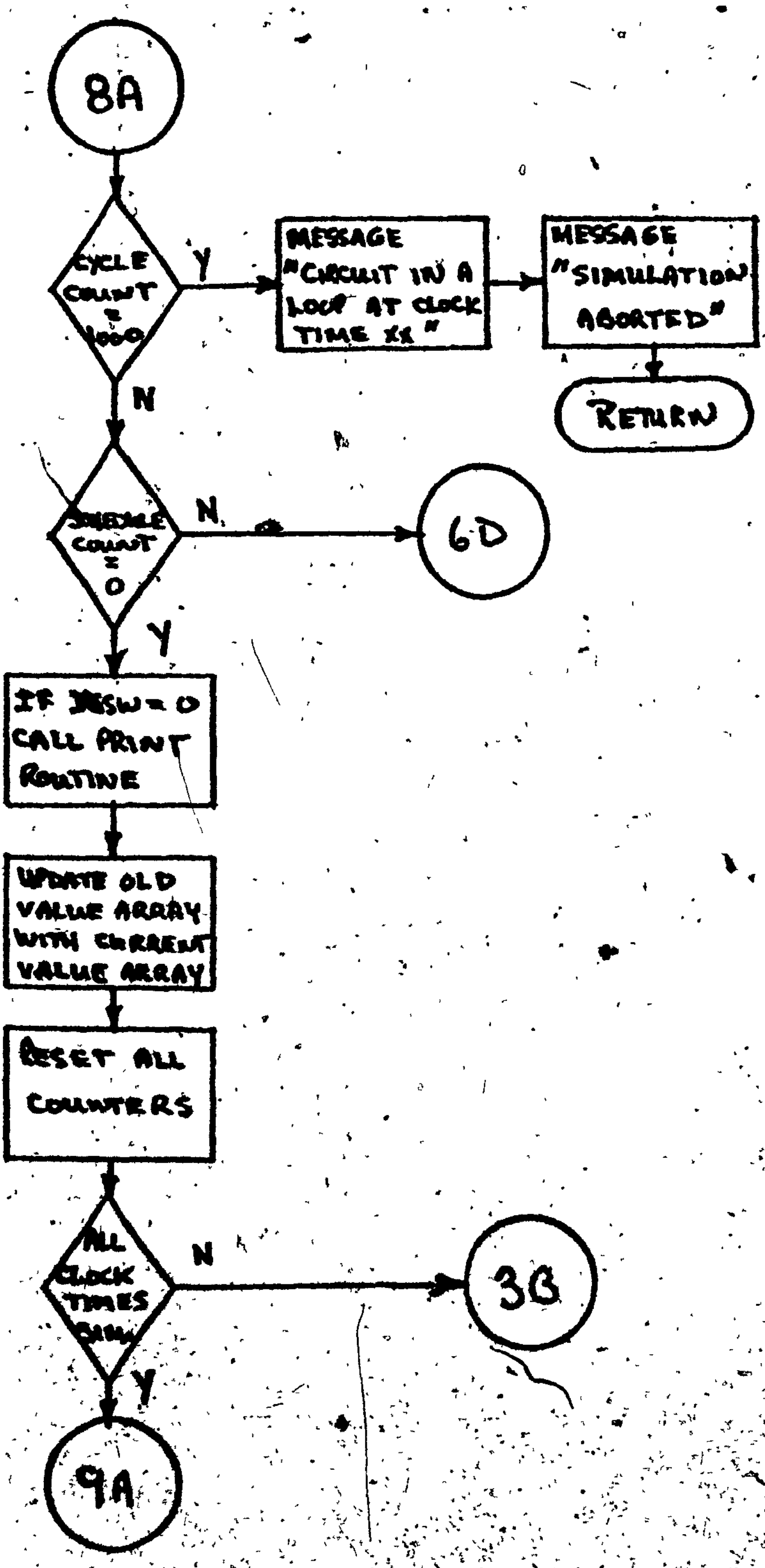

8A
CYCLE COUNT = 1000
Y
MESSAGE "CIRCUIT IN A LOOP AT CLOCK TIME XX"
MESSAGE "SIMULATION ABORTED"
RETURN
N
SCHEDULE COUNT = 0
N
6D
Y
IF DSW = 0 CALL PRINT ROUTINE
UPDATE OLD VALUE ARRAY WITH CURRENT VALUE ARRAY
RESET ALL COUNTERS
ALL CLOCK TIMES SIM.
N
3B
Y
9A

FIGURE 3.6 (Cont.) Page 8 of 10

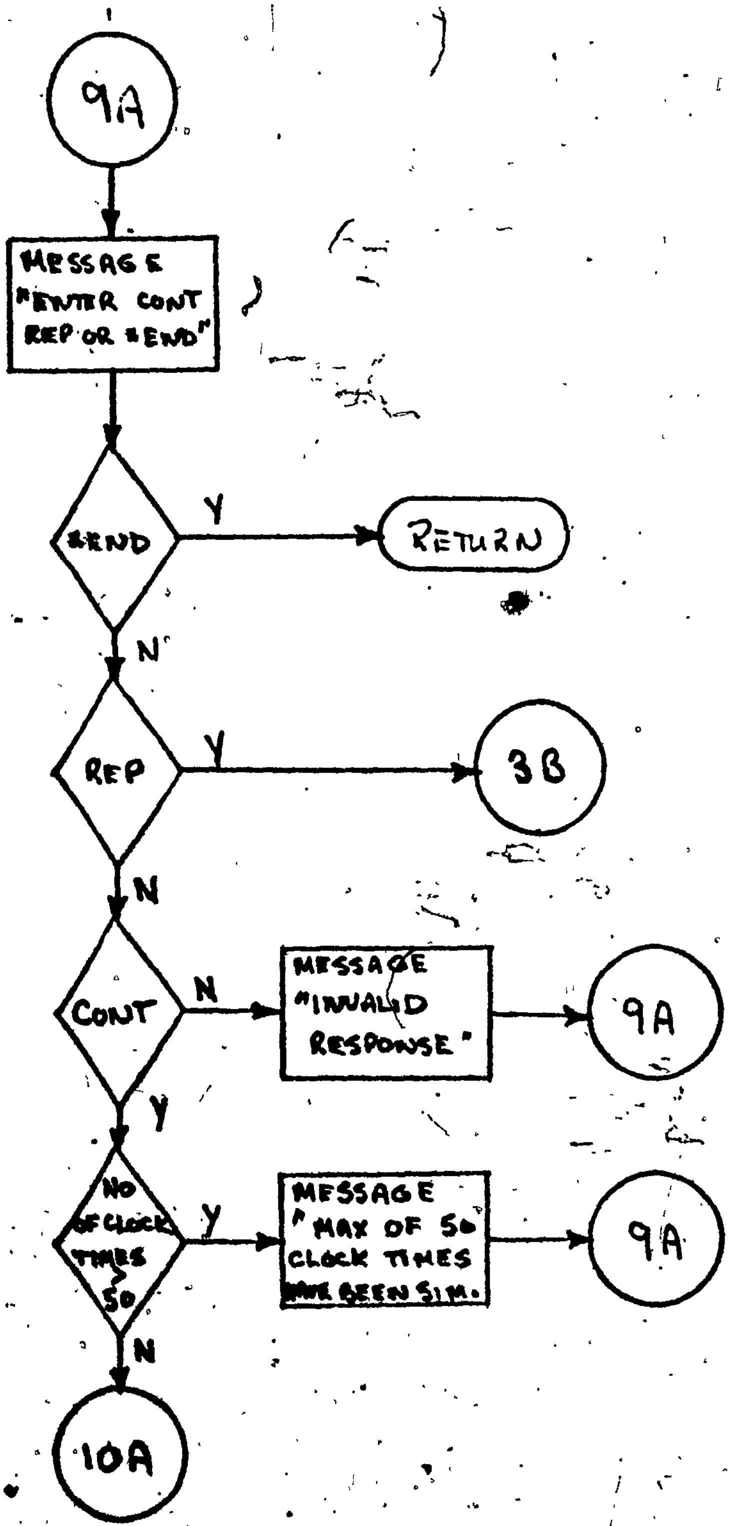

9A
MESSAGE "ENTER CONT REP OR #END"
#END
Y
RETURN
N
REP
Y
3B
N
CONT
N
MESSAGE "INVALID RESPONSE"
9A
Y
NO OF CLOCK TIMES > 50
Y
MESSAGE " MAX OF 50 CLOCK TIMES HAVE BEEN SIM.
9A
N
10A

FIGURE 3.6 (Cont.) Page 9 of 10

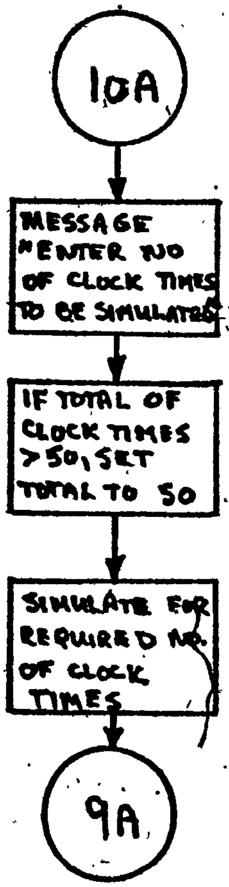

10A
MESSAGE "ENTER NO OF CLOCK TIMES TO BE SIMULATED
IF TOTAL OF CLOCK TIMES >50, SET TOTAL TO 50
SIMULATE FOR REQUIRED NO. OF CLOCK TIMES
9A

FIGURE 3.6 (Cont.) Page 10 of 10

EDIT SUBROUTINE
"EDITRN"

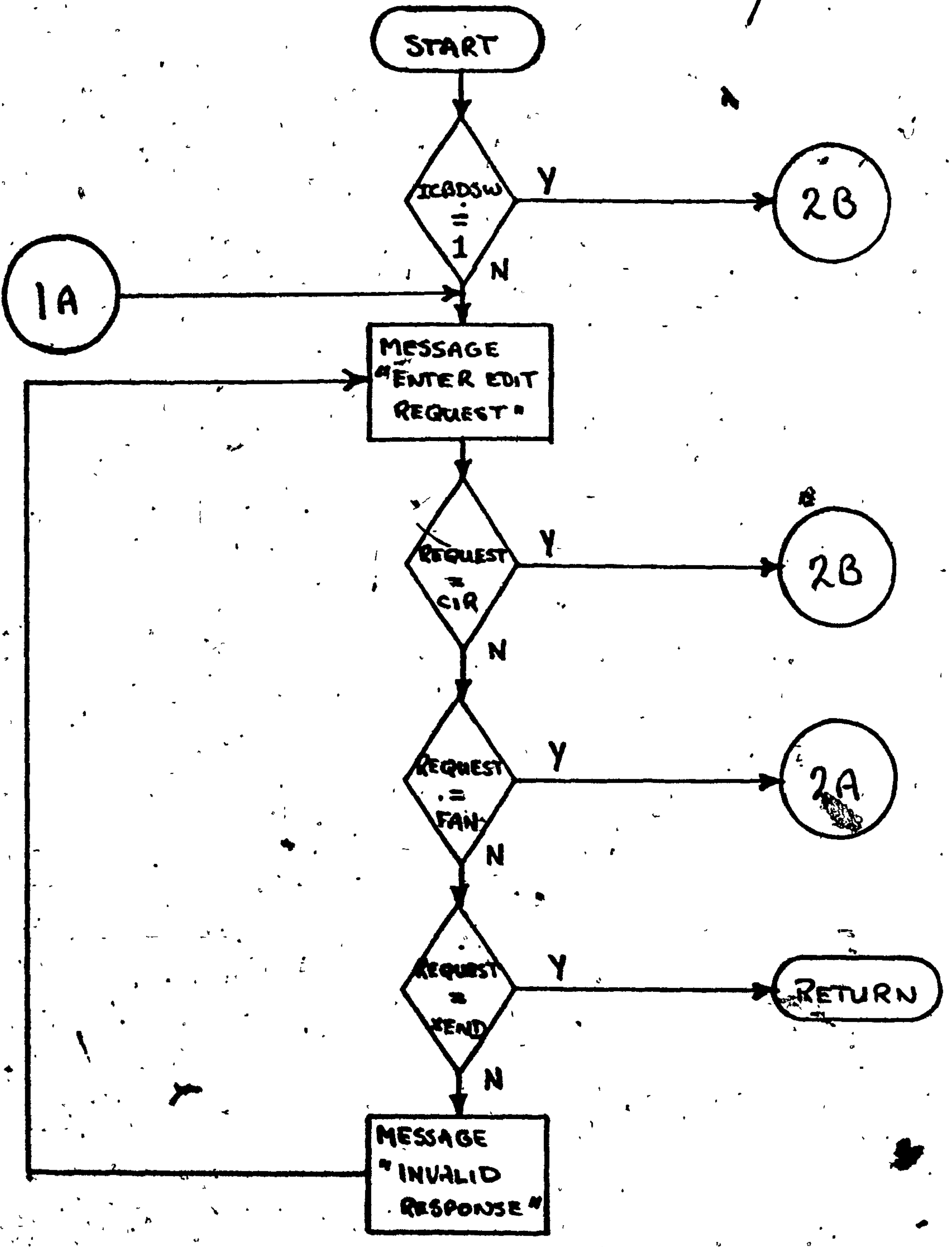


FIGURE 3.7

Page 1 of 2

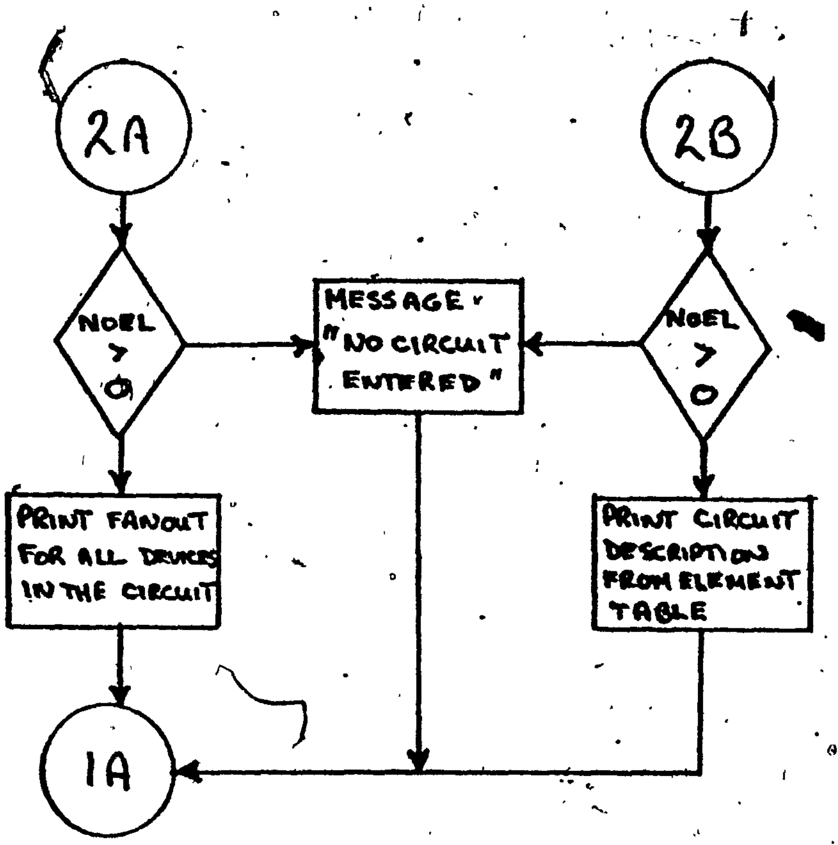

2A
2B
NOEL > 0
MESSAGE · "NO CIRCUIT ENTERED"
NOEL > 0
PRINT FANOUT FOR ALL DEVICES IN THE CIRCUIT
PRINT CIRCUIT DESCRIPTION FROM ELEMENT TABLE
1A

FIGURE 3.7 (Cont.)

Page 2 of 2

CORRECT ELEMENT
TABLE SUBROUTINE
"CORRTN"

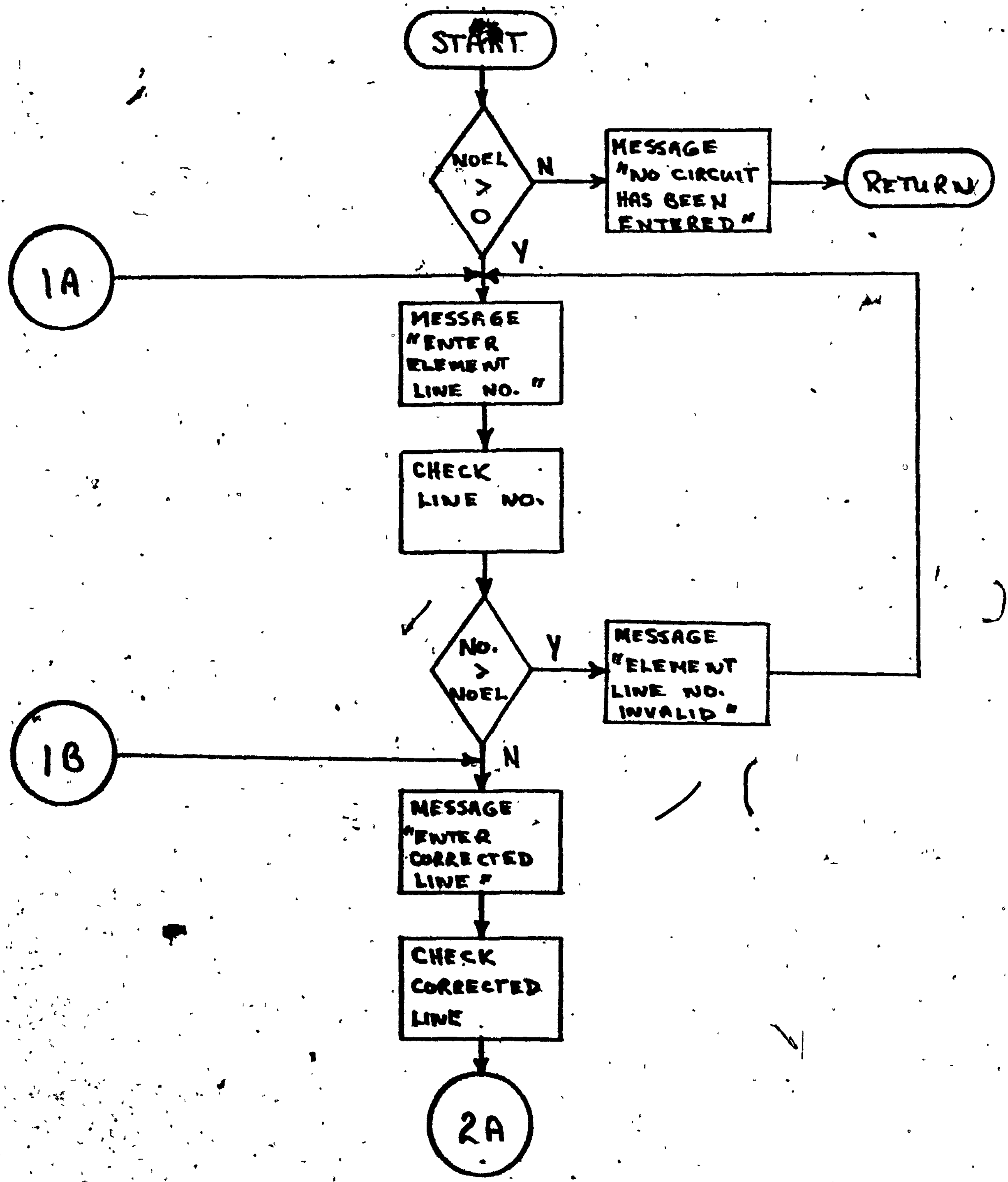

START
NOEL > 0
N
MESSAGE "NO CIRCUIT HAS BEEN ENTERED"
RETURN
Y
1A
MESSAGE "ENTER ELEMENT LINE NO."
CHECK LINE NO.
NO. > NOEL
Y
MESSAGE "ELEMENT LINE NO. INVALID"
1B
N
MESSAGE "ENTER CORRECTED LINE"
CHECK CORRECTED LINE
2A

FIGURE 3.8

Page 1 of 2

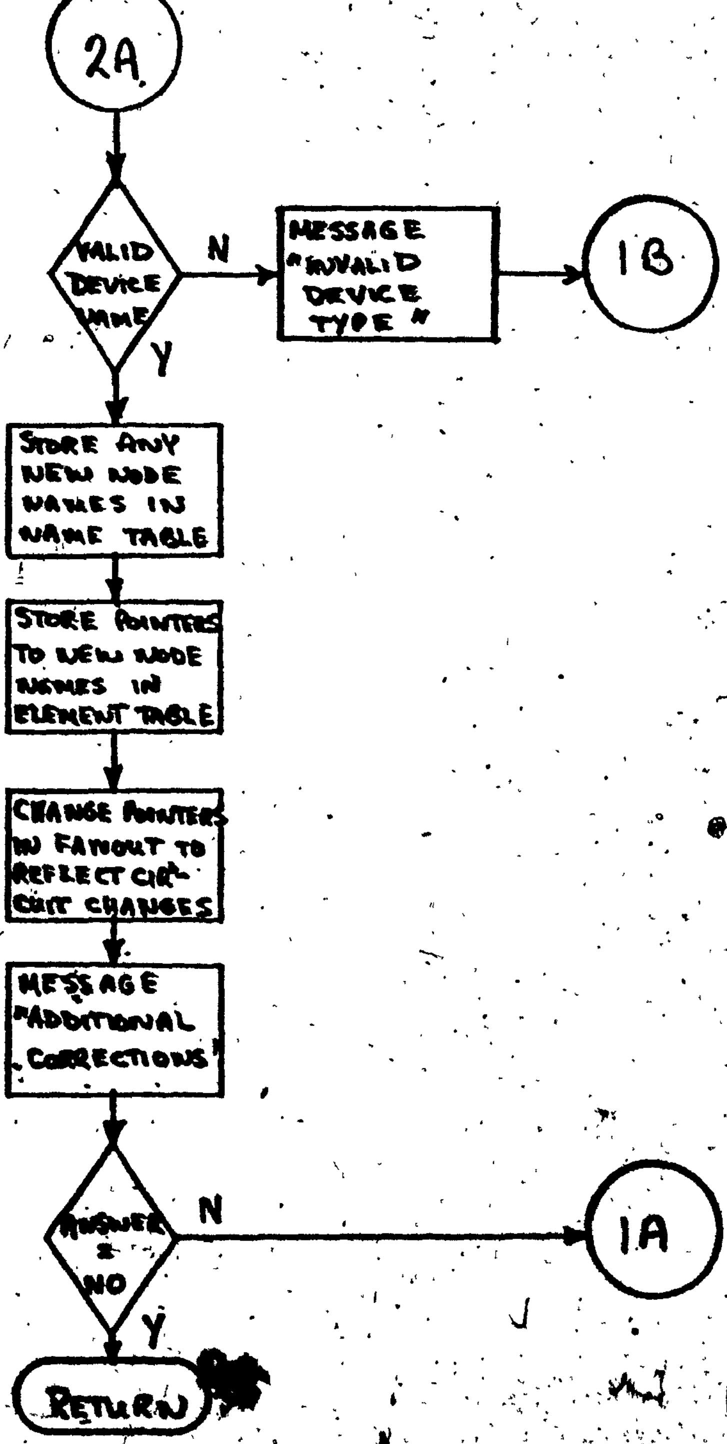

2A
VALID DEVICE NAME
N
Y
MESSAGE "INVALID DEVICE TYPE"
1B
STORE ANY NEW NODE NAMES IN NAME TABLE
STORE POINTERS TO NEW NODE NAMES IN ELEMENT TABLE
CHANGE POINTERS IN FANOUT TO REFLECT CIRCUIT CHANGES
MESSAGE "ADDITIONAL CORRECTIONS"
ANSWER = NO
N
Y
1A
RETURN

FIGURE 3.8 (Cont.)

Page 2 of 2

PRINT SUBROUTINE
"PRTLIN"

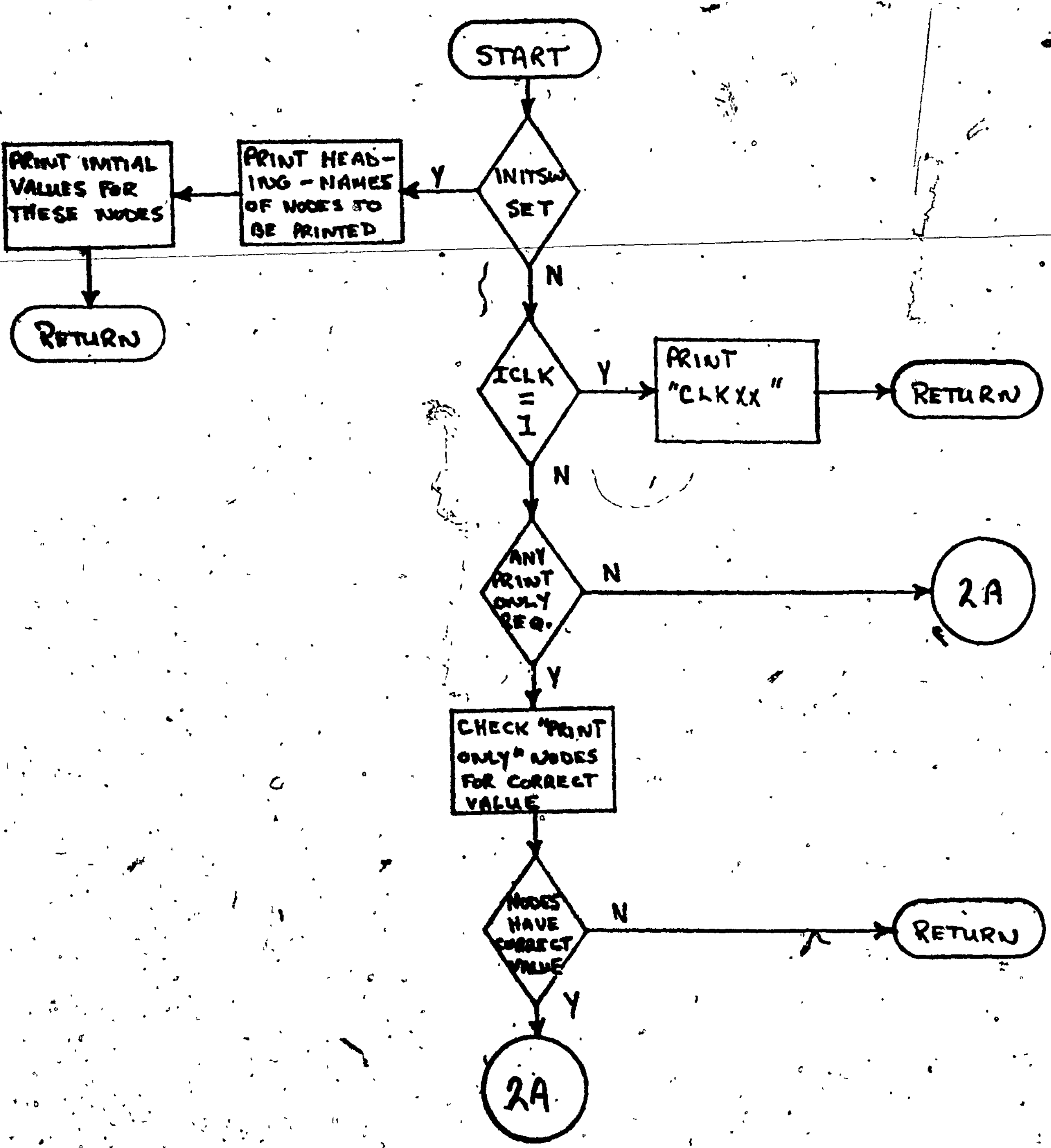

START
INITSW SET
Y
PRINT HEADING - NAMES OF NODES TO BE PRINTED
PRINT INITIAL VALUES FOR THESE NODES
RETURN
N
ICLK = 1
Y
PRINT "CLKXX"
RETURN
N
ANY PRINT ONLY REQ.
N
2A
Y
CHECK "PRINT ONLY" NODES FOR CORRECT VALUE
NODES HAVE CORRECT VALUE
N
RETURN
Y
2A

FIGURE 3.9

Page 1 of 2

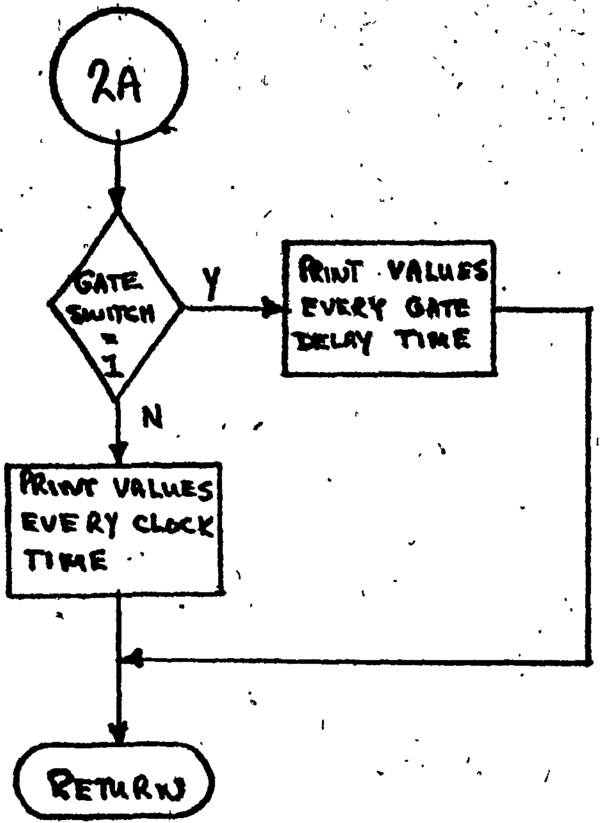

2A
GATE SWITCH = 1
Y
N
PRINT VALUES EVERY GATE DELAY TIME
PRINT VALUES EVERY CLOCK TIME
RETURN

FIGURE 3.9 (Cont.) Page 2 of 2

NEW DEVICE SUBROUTINE "NDVRTN"

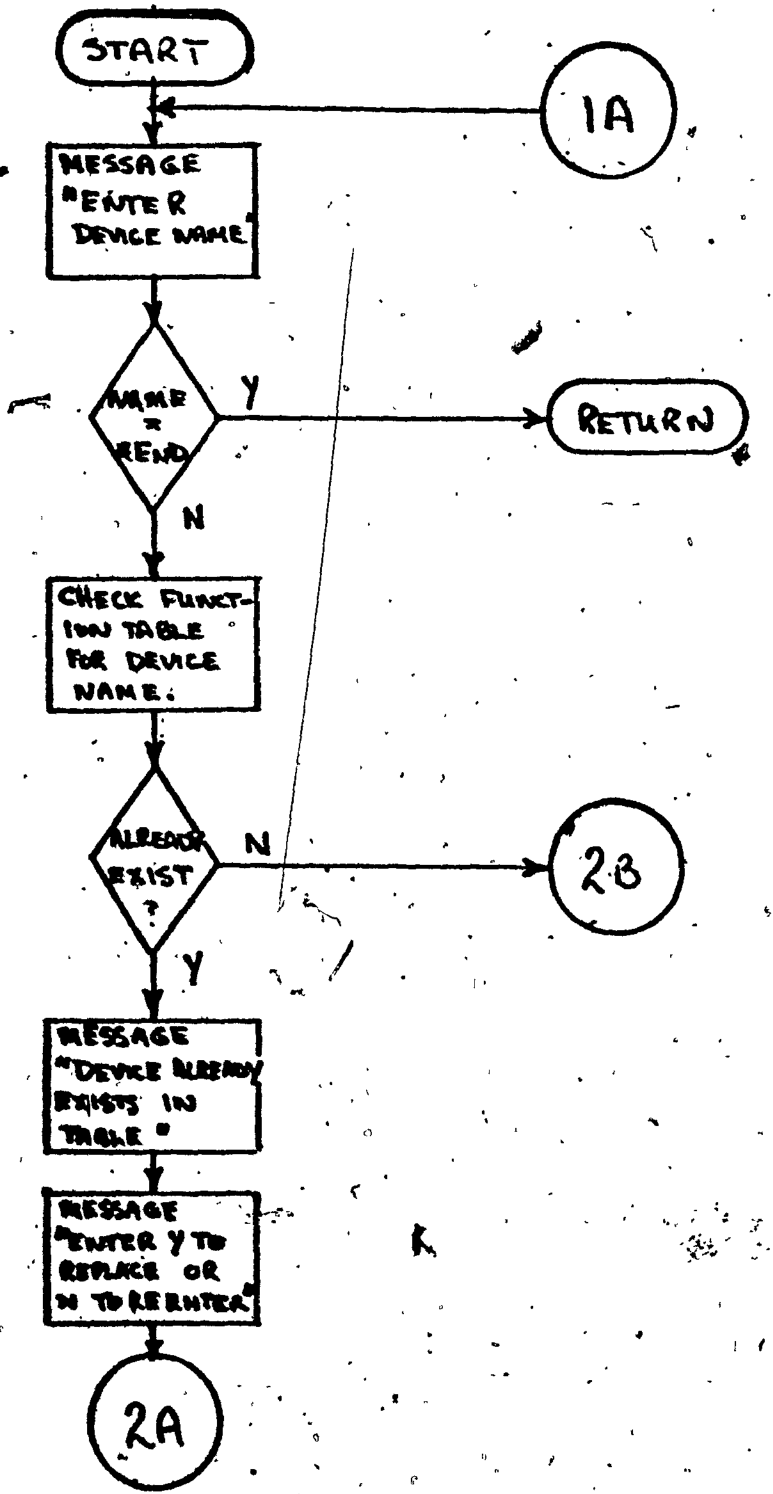

START
1A
MESSAGE "ENTER DEVICE NAME"
NAME = $END
Y
RETURN
N
CHECK FUNCT-ION TABLE FOR DEVICE NAME.
ALREADY EXIST ?
N
2B
Y
MESSAGE "DEVICE ALREADY EXISTS IN TABLE"
MESSAGE "ENTER Y TO REPLACE OR N TO REENTER"
2A

FIGURE 3.10

Page 1 of 4

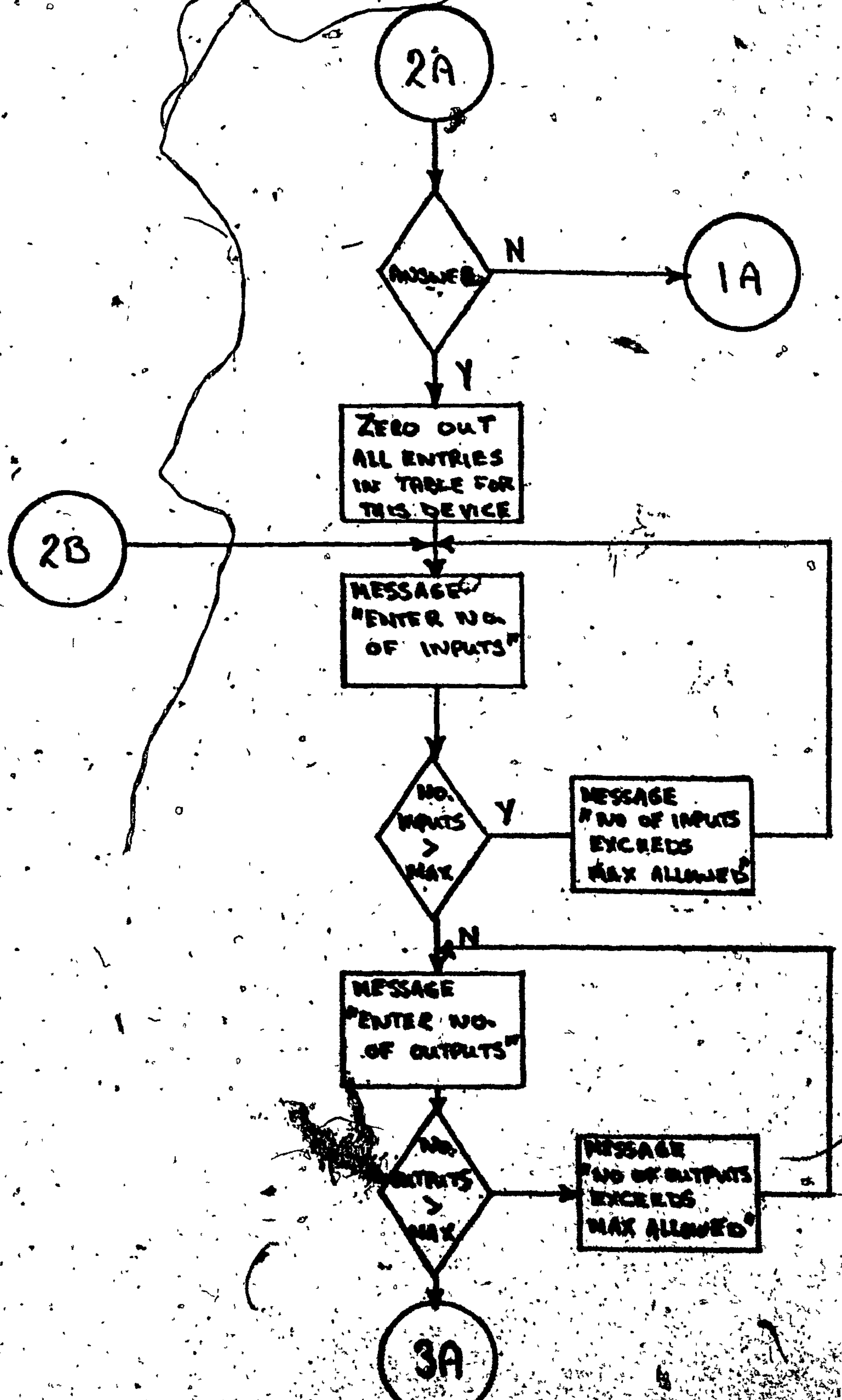

2A
ANSWER
N
1A
Y
ZERO OUT ALL ENTRIES IN TABLE FOR THIS DEVICE
2B
MESSAGE "ENTER NO. OF INPUTS"
NO. INPUTS > MAX
Y
MESSAGE "NO OF INPUTS EXCEEDS MAX ALLOWED"
N
MESSAGE "ENTER NO. OF OUTPUTS"
NO. OUTPUTS > MAX
MESSAGE "NO OF OUTPUTS EXCEEDS MAX ALLOWED"
3A

FIGURE 3.10 (Cont.) Page 2 of [illegible]

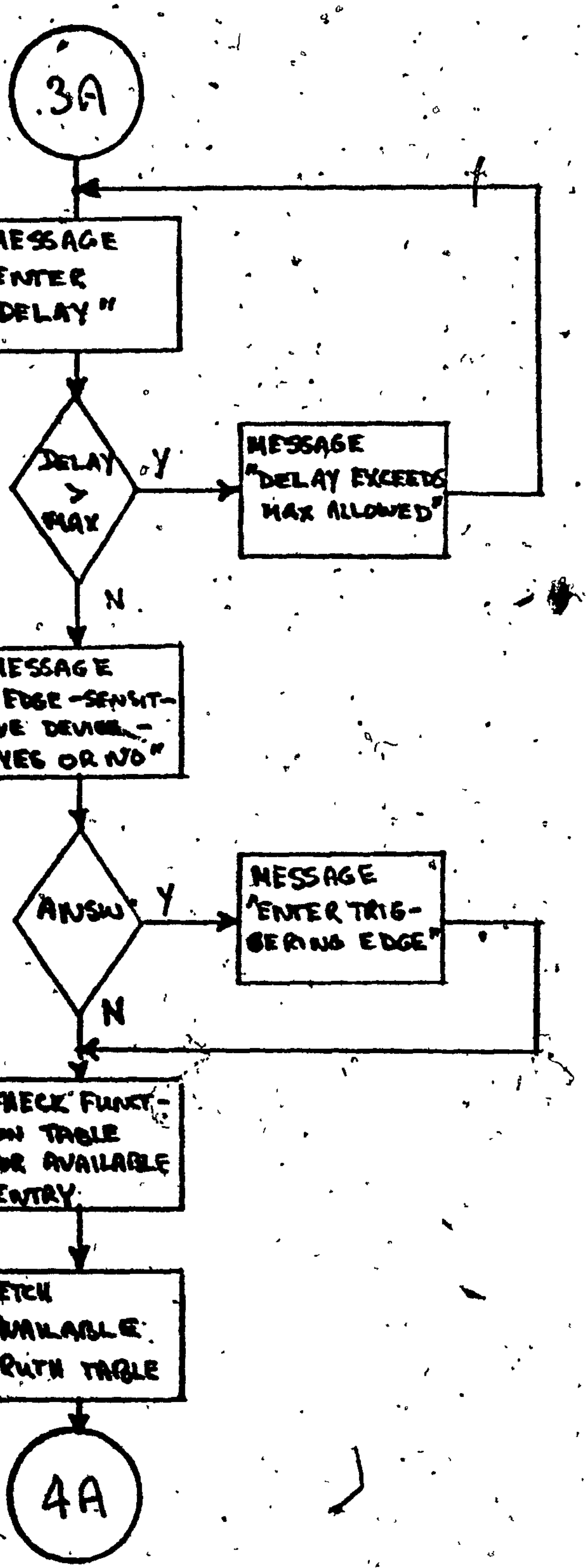

3A
MESSAGE "ENTER DELAY"
DELAY > MAX
Y
MESSAGE "DELAY EXCEEDS MAX ALLOWED"
N
MESSAGE "EDGE-SENSITIVE DEVICE - YES OR NO"
ANSW
Y
MESSAGE "ENTER TRIGGERING EDGE"
N
CHECK FUNCTION TABLE FOR AVAILABLE ENTRY
FETCH AVAILABLE TRUTH TABLE
4A

FIGURE 3.10 (Cont.) Page 3 of 4

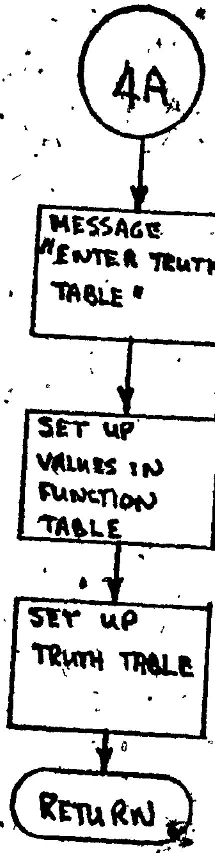


FIGURE 3.10 (Cont.)

Page 4 of 4

3.2 - CAPABILITIES OF THE PROGRAM

1) Devices - The program will simulate the following devices:

NAND gates (2 and 4 inputs)

AND gates (2 and 4 inputs)

NOR gates (2 and 4 inputs)

OR gates (2 and 4 inputs)

Xclusive OR gates (2 input)

AND-OR-INVERTER gates (4 inputs)

DRIVER (Inverter)

J/K FLIP FLOP

D FLIP FLOP

R/S FLIP FLOP

The maximum number of elements that can be simulated is 100. Up to 500 node names ean be handled.

2) External Inputs- up to 20 external inputs can be specified.

-input values for 50 clock times can be stored.

3) Printed Output- 18 nodes (restricted by page width) can be designated to be printed.

- up to 10 nodes can be specified so that output will only be printed when any 1 of the 10 has a specific value.

- printed output can be specified to be printed every clock time or at every gate delay time.

4) Parameters - the internal delay for each device type has initially been set to 1. This can be changed up to a value of 99.

- the number of inputs for gate devices can also be changed up to a maximum of 5.

5) User-Defined Devices - 2 user-defined devices can be described and used during any simulation run. These devices may either be gates or edge-sensitive devices such as flip flops.

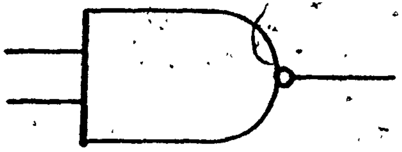
6) Simulation Values - the simulator uses 3 simulation values 0, 1, X. X is used for "don't care" or "don't know" conditions.

7) Device Fanout - a maximum of 10 output connections are allowed for each device.

3.3 - TRUTH TABLE FOR DEVICES

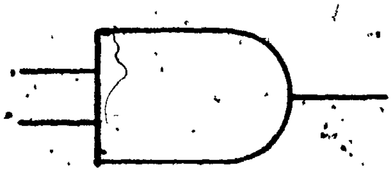
The truth tables for the devices simulated by the

NAND GATE



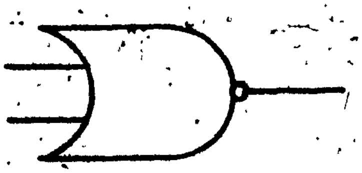
	0	1	X
0	1	1	1
1	1	0	X
X	1	X	X

AND GATE



	0	1	X
0	0	0	0
1	0	1	X
X	0	X	X

NOR GATE



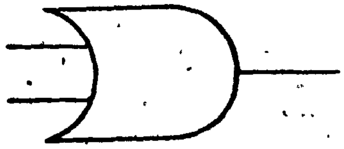
	0	1	X
0	1	0	X
1	0	0	0
X	X	0	X

DEVICE TRUTH TABLES

FIGURE 3.11

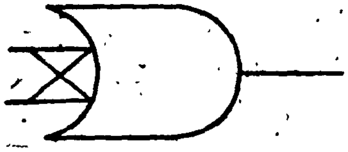
Page 1 of 4

OR GATE



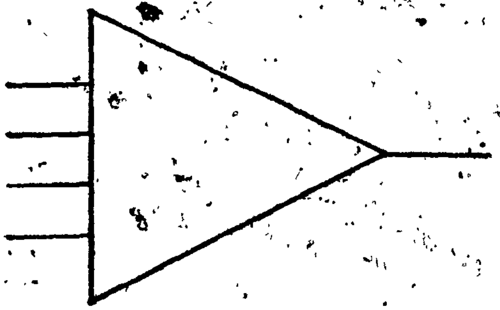
	0	1	X
0	0	1	X
1	1	1	1
X	X	1	X

EXCLUSIVE-OR



	0	1	X
0	0	1	X
1	1	0	X
X	X	X	X

AND-OR-INVERTER

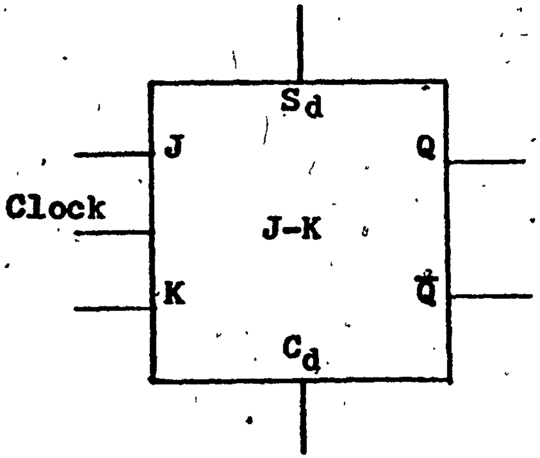


```
INPUT   0 0 0 0 1 1 1 1 0 0 0 X X X X 1 1 1 X X X X
        0 1 1 1 1 0 0 0 X X X X 0 0 0 X X X X 1 1 1
        0 0 1 1 1 1 0 0 0 X X X X 0 0 1 X X X X 1 1
        0 0 0 1 1 1 1 0 0 0 X X X X 0 1 1 1 X X X 1

OUTPUT  1 1 1 0 0 0 1 1 1 1 X X X 1 1 0 X X X X X 0
```

FIGURE 3.11 (Cont.)

J-K FLIP-FLOP



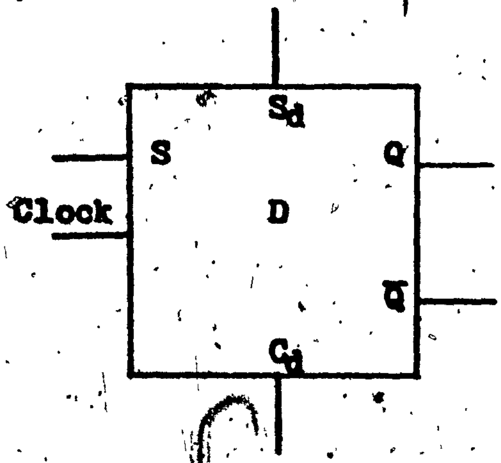
Trigger edge - 1⟶0 transition.

Asynchronous inputs, direct set (S_d) and direct clear (C_d), override the synchronous inputs. They are independent of all other inputs.

T_n		T_{n+1}
J	K	Q
0	0	Q_n
1	0	1
0	1	0
1	1	$\bar{Q}_n$

S_d	C_d	Q	$\bar{Q}$
1	1	NC	NC
0	1	1	0
1	0	0	1
0	0	-	-

D FLIP- FLOP



Trigger edge - 1⟶0 transition

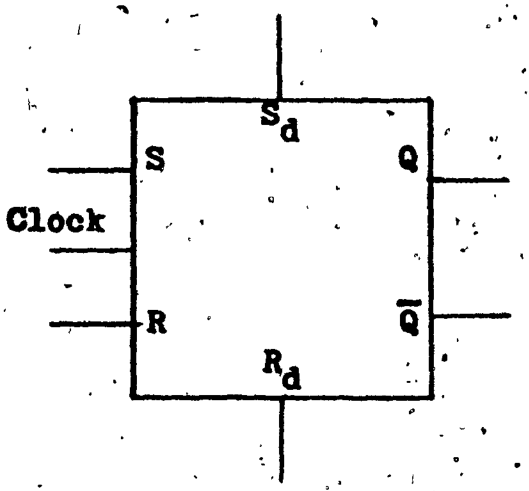
Direct inputs override clocked input operation.

DEVICE TRUTH TABLES
FIGURE 3.11 (Cont.)

T_n		T_{n+1}
S	Q	$\overline{Q}$
1	1	0
0	0	1

S_d	C_d	Q	$\overline{Q}$
0	0	—	—
1	0	1	0
0	1	0	1
1	1	Q	$\overline{Q}$

R-S FLIP FLOP



Trigger edge - 1 → 0 transition.

Direct inputs (S_d and R_d) override the clocked input operation.

T_n		T_{n+1}
S	R	Q
0	0	—
0	1	1
1	0	0
1	1	NC

S_d	R_d	Q	$\overline{Q}$
0	0	—	—
0	1	1	0
1	0	0	1
1	1	NC	NO

DEVICE TRUTH TABLES

FIGURE 3.11 (Cont.)

program are shown in Figure 3.11. The truth tables for all gate devices are three valued - 0, 1 and X. All flip flop devices trigger on a 1 to 0 transition, and all direct inputs (direct set and direct reset) override the clocked inputs.

3.4 - OPERATING PROCEDURES

3.4.1 - System Commands

The following commands are used at the SYSTEM COMMAND level of the program.

CIR - to call the "CIRCUIT BUILD" routine to enter the circuit description into the system.

EXT - to call the routine to build up the "EXTERNAL INPUT VALUE" table.

EDIT - to call in the "EDIT" routine which allows the user to print out either the circuit description or the fanout for the devices in the circuit.

PAR - to call in a routine which allows the user to change the delay time for all devices and the number of inputs for gate devices.

POP - to call in the "PRINT OPTIONS" routine.

NCT - this command is used when the user wishes to enter a new circuit description into the system.

NDEV - calls in the "NEW DEVICE" routine. This is used to enter a user-defined device into the system tables.

SIM (SNI)- calls in the "SIMULATION" routine.

SIM will initialize the circuit before beginning the simulation.

SNI will begin the simulation without initializing the circuit.

END - terminates the program.

3.4.2 - Device Types and Names

The following are the devices simulated by the program with their coded device names.

2 input NAND gate	GD2
4 input NAND gate	GD4
2 input AND gate	GA2
4 input AND gate	GA4
2 input NOR gate	GR2
4 input NOR gate	GR4
2 input OR gate	GO2
4 input OR gate	GO4
Exclusive -OR gate	XOR
AND/OR/INVERTER gate	AOR
Driver (Inverter)	DRV
J-K FLIP-FLOPS	J-K
D FLIP-FLOPS	DFF
R/S FLIP-FLOPS	R-S

3.4.3 - Entering the Circuit Description

Command used - CIR

The first time that the command "CIR" is entered, the message "ENTER CIRCUIT DESCRIPTION" will be printed. For any subsequent re-entry into the Circuit Build routine, the message will not be printed. The circuit description is entered, as per format shown in Format section, one device per line, up to a maximum of 100 devices. Node names are from one to four characters in length.

Subcommands - EDIT, CORR, *END

These commands can be entered at any time during circuit build.

EDIT will produce a formatted print-out of the circuit description with line numbers at the left-hand side. This print-out would be useful in detecting errors in the entering of the circuit description.

CORR allows corrections to be made to the circuit description table on a line basis. The line (device description) to be corrected is referenced by the numbers printed on the left in the EDIT print-out. The device description is corrected by re-entering the entire line.

Return from both of these routines is back to the circuit build routine at which point the user can continue to enter the remainder of the circuit description.

*END allows the user to exit from the circuit build routine back to the system command level.

NOTE - The circuit build routine can be re-entered at any time from the system command level by entering the command "CIR". The routine will then continue to build up the circuit description table from the last entry.

Any node which is permanently connected to a ground potential can be assigned the name "LO", and any node connected to a high or positive source can be assigned the name "HI". These are system reserved words and will assign the correct value to these nodes for the simulation.

3.4.4 - Entering the External Signal Input Values

Command used - EXT

A maximum of 20 nodes can be defined as external inputs to the simulators.

In response to the message, "ENTER NO. OF INPUTS AND NAMES", the number of inputs and their names are entered as shown in the input format section. The number of inputs must be a 2-digit number with leading zeroes if necessary.

Only ten node names may be entered on the first line. For a system with more than 10 inputs, the remainder, up to a maximum of 10, are entered on the next line.

The nodes assigned as inputs must have previously been entered into the system during the circuit build phase. Any node name that is not in the System Name Table will be rejected.

*END sub-command will allow the user to exit back to the system command level. In this case, all entries in the EXTERNAL INPUTS TABLE are deleted.

If all input nodes specified are valid, the system will respond with the message, "ENTER EXTERNAL VALUES FOR EACH NODE". The input values for each input node are entered either as a repetitive string or an absolute string of ones and zeroes.

For repetitive inputs the following example illustrates the method of entering the input values. The input sequence 1001100 is to be repeated for an input node ABCD -- it is entered in the following manner.

```
ABCDR071001100
 |  | |   |
 |  | |   └── input bit stream - clock time values.
 |  | └── number of digits in string to be repeated.
 |  └──R - indicates repeat the next 7 bits.
 └─── node name.
```

The system will build up the input vector for this node by repeating the 7 specified values until a maximum of 50 clock times have been established.

For non-repetitive inputs, an absolute string of ones and zeroes can be entered. An example of this method of entry is as follows:

A node BBBB has the following input values 111001011001 ------etc. It is specified as follows:

```
BBBBA00111001011001 ------
 |  | |      |
 |  | |      └──── input clock time values
 |  | └──────── must be zero - not used
 |  └── A - indicates - take values as they occur
 └─── node name
```

The input vector is built up by taking the values specified as the input clock time values.

After all inputs have been specified, return is made automatically to the system command level.

3.4.5 - Obtaining An Edited Printout Of The Circuit To Be Simulated

Command used - EDIT

In response to the command EDIT, the system will type out the message:

"ENTER EDIT REQUEST - CIR OR FAN".

"CIR" will cause an edited printout of the circuit description to be printed.

"FAN" will list all devices in the system by name, and indicate the devices to which they fan out to.

3.4.6 - Specifying Output Print Options

Command used - POP

Print options available are:-

1) Specifying up to 18 nodes to be printed. Default is first 18 nodes in the name table.
2) Specifying up to 10 nodes with specific values so that printed output will only occur when any one of the nodes has that value.
3) Specifying printed output to occur every clock time or every unit delay time.

In response to the message:

"ENTER PRINT REQUEST - C,G AND PRINT ONLY REQUEST",

the user can enter:

C - Print every clock time, or

G - Print every delay time or gate time,

plus up to 10 "print only" node names and their "print only" values.

In response to the message:

"ENTER NODES TO BE PRINTED",

the user can enter either: 1) up to 18 node names or 2) *END to exit back to the system command level. In case 2, default conditions will prevail and the first 18 node names in the name table will be monitored and printed.

3.4.7 - Changing System Parameters

Command used - PAR

This command allows the user to change the delay time for devices or the number of inputs for a gate device.

In response to the message:

"ENTER DEVICE TYPE",

the user enters one of the device names shown in Paragraph 3.4.2 of this section.

In response to the message:

"ENTER PAR TO BE CHANGED IN FORMAT - AAA 99AAA 99",

the user enters:

IMP + a 2 digit number with leading zeroes to indicate the new number of inputs, or

DEL + a 2 digit number with leading zeroes,
if necessary, to indicate the new delay
time, or
both of above.

3.4.8 - To Enter a New Circuit Into The System

Command used - NCT

This command will zero out the element table, the name table, external input table, and all pointers and counters. The system will respond with the message, "ENTER CIRCUIT DESCRIPTION". The user can then enter the new circuit.

NOTE: Any parameter values in the Function Description table that had been changed for the previous circuit, will not be reset. They can be changed back to their original values or new values by using the command PAR.

3.4.9 - Defining a New Device to the System

Command used - NDEV

The user may specify two new devices to be added to the system tables. They may be gate devices or edge-sensitive devices such as flip-flops.

The user will enter the new device name, number of inputs and outputs, the delay, whether the device is a gate or flip-flop type, and finally the truth table for the device in response to questions asked by the system. For flip-flop type devices the clock input is not counted as one of the

The size of the truth table expected by the system is:

number of columns = sum of number of inputs + number of outputs.

number of rows = (number of inputs.)^2

Return to system command level is done automatically after entering the truth table.

3.4.10 - Performing the Simulation

Commands used - SIM or SNI

The command "SIM" will call in the simulation routine and initialize all node values to X (unknown) and all input node values to zero. At this point, the system will ask if any input nodes are to be set to a value other than zero. If so, the name and value is entered in format shown in the format section. The simulator will then simulate the circuit with these values until a steady state is reached. This is the initialized state of the circuit.

The command "SNI" will call in the simulation routine without initializing the circuit. This can be used when it is desired to simulate more than the maximum 50 clock times. The first 50 clock times are simulated using the command "SIM". Return is then made to the system command level where the next 50 input clock values are entered using the command "EXT". The simulation can then continue by using the command "SNI" so that circuit node values are unchanged.

After initialization or entry to the simulation routine, the system will request the number of clock times to be simulated. The user responds with a 2 digit number indicating the number of clock times (up to a maximum of 50) to be simulated.

The simulation will then proceed with the specified node values being printed for each gate or clock time as desired.

After the specified number of clock times has been simulated, the system will request further simulation instructions by printing the message;

"ENTER CONT, REP, OR *END".

CONT - means to continue the simulation from this point. The system will again ask the number of clock times to be simulated. When the maximum 50 clock times have been reached, the simulation stops.

REP - means repeat the simulation using the previous specified number of clock times and input values without re-initializing the circuit. This will allow a circuit with input values of a repetitive nature to be continuously simulated.

*END - indicates the end of the simulation run and return is made to the system command level.

3.4.11 - Making Changes in the Circuit Description

Command used - CORR

This is a subcommand of the circuit build routine. To make corrections, the circuit build routine is called using the command "CIR", and the subcommand "CORR" is entered. Only corrections are allowed. No deletions of description entries are permitted. The system will ask which entry in the circuit description table is to be corrected. The correction is made by re-entering the entire line describing the device.

3.5 - INPUT FORMATS

Figure 3.12 shows the input data formats for the system. Also indicated is the routine and system messages to which the formats refer.

ROUTINE	MESSAGE
MAIN	"ENTER COMMAND"

1 3
AAA

FORMAT (A3)

ROUTINE	MESSAGE
CIRBLD	"ENTER CIRCUIT DESCRIPTION"

FORMAT (A3,A3,7A4)

	1 3	4 6	7 10	11 14	15 18	19 22	23 26	27 30	31 34
GATE DEVICES	DEVICE TYPE	DEVICE NO.	INPUT NODE #1	INPUT NODE #2	INPUT NODE #3	INPUT NODE #4	INPUT NODE #5	OUTPUT NODE #1	OUTPUT NODE #2
FLIP-FLOP DEVICES	DEVICE TYPE	DEVICE NO.	GATE INPUT #1	GATE INPUT #2	CLOCK INPUT	SET INPUT	RESET INPUT	OUTPUT Q (1)	OUTPUT $\overline{Q}$ (0)

INPUT FORMATS

FIGURE 3.12

Page 1 of 6

GATE INPUT #1	J	for J-K	Flip-Flop
	S	for D	Flip-Flop
	S	for R-S	Flip-Flop
GATE INPUT #2	K	for J-K	Flip-Flop
	Not used for	D	Flip-Flop (blank)
	R	for R-S	Flip-Flop

NOTE: For Flip-Flop devices, all unused nodes are entered as blanks. For gates: non-used input node fields (i.e. inputs 3,4,5 for a 2 input NAND gate) need not be entered as blanks. Only the number of fields necessary to describe the device is used - 3 for a 2 input NAND gate. The node names are entered as follows: INPUT NAME #1 INPUT NAME #2 OUTPUT. In this case, the output node name is entered in input-field No. 3.

INPUT FORMATS
FIGURE 3.12 (Cont.)

Page 2 of 6.

ROUTINE	MESSAGE
EDITRN	"ENTER EDIT REQUEST -CIR OR FAN"

1 3	
CIR FAN	

FORMAT (A3)

ROUTINE	MESSAGE
EXTRTN	"ENTER NO. OF INPUTS AND NAMES"

FORMAT (12, 10A4)

1 2	3 6	7 10	11 14	15 18	19 22	23 26	27 30	31 34	35 38	39 42
NUMBER OF INPUTS	NODE #1	NODE #2	NODE #3	NODE #4	NODE #5	NODE #6	NODE #7	NODE #8	NODE #9	NODE #10

IF THE NUMBER OF EXTERNAL INPUTS > 10, THE REMAINING ARE ENTERED ON A SECOND LINE.

FORMAT (10A4)

1 4	5 8	9 12	13 16	17 20	21 24	25 28	29 32	33 36	37 40
NODE #11	NODE #12	NODE #13	NODE #14	NODE #15	NODE #16	NODE #17	NODE #18	NODE #19	NODE #20

INPUT FORMATS

FIGURE 3.12 (Cont.)

Page 3 of 6

ROUTINE	MESSAGE
EXTRN	"ENTER EXTERNAL VALUES FOR EACH NODE"

ORMAT (A4,A1,I2,50A1)

1 4	5	6 7	8 87
NODE NAME	R or A	REP. COUNT	EXTERNAL INPUT VALUES - UP TO 50

PRTRTN "ENTER PRINT REQUEST - σ,G AND PRINT ONLY REQUESTS"

ORMAT (A1,10(A4,A1))

1	2 5	6	7 10	11	12 15	16	17 20	21	22 25	23	24 27	28	29 32	33	34 37	38	39 42	43	44 47	48
C or σ	NODE # 1	VALUE	NODE # 2	VALUE	NODE # 3	VALUE	NODE # 4	VALUE	NODE # 5	VALUE	NODE # 6	VALUE	NODE # 7	VALUE	NODE # 8	VALUE	NODE # 9	VALUE	NODE # 10	VALUE

"ENTER NODES TO BE PRINTED"

MAT (18 A4)

1 4	5 8	9 12	13 16	17 20	53 56	57 60	61 64	65 68	69 72
NODE # 1	NODE # 2	NODE # 3	NODE # 4	NODE # 5	NODE # 14	NODE # 15	NODE # 16	NODE # 17	NODE # 18

INPUT FORMATS
FIGURE 3.12 (Cont.)

Page 4 of 6

ROUTINE			MESSAGE
PARRTN	1 3 X X X	FORMAT (A3)	"ENTER DEVICE TYPE"
	1 4 6 INP ƀ or DEL ƀ | NEW VALUE	FORMAT (A4, 12)	"ENTER PAR TO BE CHANGED IN FORMAT AAA 99AAA 99"

INPUT FORMATS
FIGURE 3.12 (Cont.)

Page 5 of 6

ROUTINE	MESSAGE
SIMRUN	"PRESET EXTERNAL INPUT NODES" "ENTER NODES AND THEIR VALUES"

FORMAT (10(A4,A1))

1 4	5	6 9	10	11 14	15	16 19	20	21 24	25	26 29	30	31 34	35	36 39	40	41 44	45	46 49	50
NODE # 1	VALUE	NODE # 2	VALUE	NODE # 3	VALUE	NODE # 4	VALUE	NODE # 5	VALUE	NODE # 6	VALUE	NODE # 7	VALUE	NODE # 8	VALUE	NODE # 9	VALUE	NODE # 10	VALUE

INPUT FORMATS

FIGURE 3.12 (Cont.)

Page 6 of 6

CONCLUSION

A program has been written which makes possible the simulation of digital circuits as an available tool for design verification or teaching purposes. The program has been found to be very easy to use. New users have found the input format very easy to use as opposed to using a special design language to describe the circuit.

As for simulating real devices, the program does not use a fixed constant as the internal delay of an element grouping, but allows the user to designate his own delay to suit his simulation purposes. This allows for a more realistic simulation. A major feature of the program is the use of a selective trace procedure, i.e. that only devices whose inputs have changed are re-evaluated during each cycle rather than evaluating all devices each cycle.

Data input has been simplified. All that is required is a well labeled diagram. The description of the circuit is taken directly from the diagram. Each element is described with its input and output nodes as a single statement where the order of these statements is unimportant.

The program is structured such that the simulation conditions (inputs and outputs) can be changed independently of the circuit description and vice versa as outlined in Chapter 1 - Logic Simulation Process.

A sample terminal session is shown in Appendix A. The circuits simulated are shown in Figure A.1. The simulations show the program's capability to simulate both synchronous and asynchronous devices. Various features of the program are indicated on the simulation printout.

The simulation program has been programmed in FORTRAN and operates on-line in a conversation mode on a CDC 6400 computer.

APPENDIX A

```
****SAMPLE CIRCUIT - SYNCHRONOUS BINARY UP-DOWN COUNTER




ENTER COMMAND
? CIR  <-----------------------------------------   ENTER CIRCUIT BUILD MODE
ENTER CIRCUIT DESCRIPTION
? J-K-1 HI  HI   CLK          Q1  Q1-
? J-K-2 J2K2J2K2CLK           Q2  Q2-              ENTER CIRCUIT DESCRIPTION
? J-K-3 J3K3J3K3CLK           Q3  Q3-              DIRECTLY FROM DIAGRAM.
? GD2-1 UPENQ1   UPQ1
? GD2-2 DNENQ1- DNQ1
? GD2-3 UPQ1    DNQ1J2K2
? GD4-1 UPENQ1  Q2  HI  UPQ2
? GD4-2 DNENQ2- Q1- HI  DNQ2
? GD2-4 UPQ2DNQ2J3KE
? EDIT  <--------------------------------------     PRINT DESCRIPTION OF CIRCUIT
                  /--/    /--/    /--/    /--/    /--/    /--/    /--/
 1   J-K-1        HI      HI      CLK                     Q1      Q1-

 2   J-K-2        J2K2    J2K2    CLK                     Q2      Q2-

 3   J-K-3        J3K3    J3K3    CLK                     Q3      Q3-

 4   GD2-1        UPEN    Q1                              UPQ1

 5   GD2-2        DNEN    Q1-                             DNQ1

 6   GD2-3        UPQ1    DNQ1                            J2K2

 7   GD4-1        UPEN    Q1      Q2      HI              UPQ2

 8   GD4-2        DNEN    Q2-     Q1-     HI              DNQ2

 9   GD2-4        UPQ2    DNQ2                            J3KE
? COR  <-----------------------------------------   ENTER CORRECT MODE   MAKE CO
ENTER ELEMENT LINE NO.-2 DIGITS                        TO CIRCUIT DESCRIPTION.
? Ø9
ENTER CORRECT LINE
? GD2-4 UPQ2DNQ2J3K3
ADDITIONAL CORRECTIONS
? N
? *END  <------------------------------------------  EXIT CIRCUIT BUILD MODE.
ENTER COMMAND
? EXT  <-----------------------------------------> THIS COMMAND ALLOWS THE
ENTER NO OF INPUTS AND NAMES                        USER TO DEFINE THE EXT.
? Ø3CLK UPENDNEN                                    INPUTS.
ENTER EXTERNAL VALUES FOR EACH NODE
```

```
? CLK RØ2Ø1
? UPENAØØ111111111111111111ØØØØØØØØØØØØØØØØØØØ1111111111111111111
? DNENA:-ØØØØØØØØØØØØØØØØØØØ1111111111111111111ØØØØØØØØØØØØØØØØØØØ
ENTER COMMAND
? POP  <------------------------------------  PRINT OUTPUT COMMAND
ENTER PRINT REQUEST-G,C-AND PRINT ONLY REQUEST
? C  <--------------------------------------  PRINT OUTPUT EVERY CLOCK TIME.
ENTER NODES TO BE PRINTED
? UPENDNENQ1  Q2  Q3  <---------------------  PRINT VALUE OF THESE NODES ONLY.
ENTER COMMAND
? SIM  <------------------------------------  SIMULATE THE CIRCUIT
NO OF CLOCK TIMES TO BE SIMULATED
? 3Ø
PRESET EXTERNAL INPUT NODES-Y OR N
? N
       UPEN      Q1       Q3
            DNEN      Q2
INIT    Ø     1    1
CLK  1  1     Ø    1    1    1               UP ENABLE (UPEN=1)
CLK  2  1     Ø    1    1    1               THE COUNTER IS COUTING UP.
CLK  3  1     Ø    Ø    Ø    Ø ]
CLK  4  1     Ø    Ø    Ø    Ø ]
CLK  5  1     Ø    1    Ø    Ø ]
CLK  6  1     Ø    1    Ø    Ø ]
CLK  7  1     Ø    Ø    1    Ø ]
CLK  8  1     Ø    Ø    1    Ø ]
CLK  9  1     Ø    1    1    Ø ]
CLK1Ø   1     Ø    1    1    Ø ]
CLK11   1     Ø    Ø    Ø    1 ]
CLK12   1     Ø    Ø    Ø    1 ]
CLK13   1     Ø    1    Ø    1 ]
CLK14   1     Ø    1    Ø    1 ]
CLK15   1     Ø    Ø    1    1 ]
CLK16   1     Ø    Ø    1    1 ]
CLK17   1     Ø    1    1    1 ]
CLK18   1     Ø    1    [illegible]    1 ]
CLK19   Ø     1    Ø    [illegible]    Ø
CLK2Ø   Ø     1    Ø    Ø    Ø
CLK21   Ø     1    1    1    1 ]
CLK22   Ø     1    1    1    1 ]             DOWN ENABLE (DNEN=1)
CLK23   Ø     1    Ø    1    1 ]             THE COUNTER IS COUNTING DOWN.
CLK24   Ø     1    Ø    1    1 ]
CLK25   Ø     1    1    Ø    1 ]
CLK26   Ø     1    1    Ø    1 ]
CLK27   Ø     1    Ø    Ø    1 ]
CLK28   Ø     1    Ø    Ø    1 ]
CLK29   Ø     1    1    1    Ø ]
CLK3Ø   Ø     1    1    1    Ø ]
END OF SIMULATION RUN                        FINISHED [illegible] clock times
ENTER CONT,REP,OR *END
? CONT                                       CONTINUE THE SIMULATION
NO OF CLOCK TIMES TO BE SIMULATED.            FOR ANOTHER 15 CLOCK TIMES.
? 15
CLK31   Ø     1    Ø    1    Ø ]
CLK32   Ø     1    Ø    1    Ø ]
CLK33   Ø     1    1    Ø    Ø
```

```
CLK34  0   1   1   0   0
CLK35  0   1   0   0   0
CLK36  0   1   0   0   0
CLK37  1   0   1   1   1
CLK38  1   0   1   1   1
CLK39  1   0   0   0   0
CLK40  1   0   0   0   0
CLK41  1   0   1   0   0                 COUNTER IS COUNTING UP
CLK42  1   0   1   0   0                 AGAIN.
CLK43  1   0   0   1   0
CLK44  1   0   0   1   0
CLK45  1   0   1   1   0
ENTER CONT.REP.OR *END
? CONT
NO OF CLOCK TIMES TO BE SIMULATED
? 05
CLK46  1   0   1   1   0
CLK47  1   0   0   0   1
CLK48  1   0   0   0   1
CLK49  1   0   1   0   1
CLK50  1   0   1   0   1
ENTER CONT.REP.OR *END
? CONT
MAX OF 50 CLOCK TIMES HAS BEEN REACHED
ENTER CONT.REP.OR *END
? *END                                   EXIT SIMULATION MODE.
ENTER COMMAND
? EXT                                    EXTERNAL INPUT
ENTER NO OF INPUTS AND NAMES
? 03CLK UPENDNEN
ENTER EXTERNAL VALUES FOR EACH NODE
? CLK R0201                              SET NEW VALUES FOR INPUTS.
? UPENR010
? DNENR010
ENTER COMMAND
? SIM
NO OF CLOCK TIMES TO BE SIMULATED
? 08
PRESET EXTERNAL INPUT NODES-Y OR N
? N
      UPEN      Q1      Q3
           DNEN     Q2
INIT   0   1   1   0   1                 BOTH UPEN AND DNEN =0
CLK 1  0   0   1   1   1                 THE COUNTER IS NOT COUNTING.
CLK 2  0   0   1   1   1
CLK 3  0   0   0   1   1
CLK 4  0   0   0   1   1
CLK 5  0   0   1   1   1
CLK 6  0   0   1   1   1
CLK 7  0   0   0   1   1
CLK 8  0   0   0   1   1
END OF SIMULATION RUN
```

```
ENTER CONT.REP.OR *END
? *END
ENTER COMMAND
? EDIT  <---------------------------------     ENTER EDIT MODE.
ENTER EDIT REQUEST-CIR OR FAN
? CIR          <-------------------------      PRINT CIRCUIT DESCRIPTION.
        DEVICE  INPUT   INPUT   INPUT   INPUT   INPUT   OUTPUT  OUTPUT
                  1       2       3       4       5       1       2
                 (J)     (K)    (CLK)   (SET)   (RSET)

                 /--/    /--/    /--/    /--/    /--/    /--/    /--/
 1   J-K-1       HI      HI      CLK                     Q1      Q1-

 2   J-K-2       J2K2    J2K2    CLK                     Q2      Q2-

 3   J-K-3       J3K3    J3K3    CLK                     Q3      Q3-

 4   GD2-1       UPEN    Q1                              UPQ1

 5   GD2-2       DNEN    Q1-                             DNQ1

 6   GD2-3       UPQ1    DNQ1                            J2K2

 7   GD4-1       UPEN    Q1      Q2      HI              UPQ2

 8   GD4-2       DNEN    Q2-     Q1-     HI              DNQ2

 9   GD2-4       UPQ2    DNQ2                            J3K3
ENTER EDIT REQUEST-CIR OR FAN
? FAN     <-------------------------------     PRINT ALL DEVICES AND THE
             FANOUT                              DEVICES TO WHICH THEY FAN-
DEVICE        FANOUT DEVICES                    OUT TO.
J-K-1         GD2-1   GD4-1
              GD2-2   GD4-2
J-K-2         GD4-1
              GD4-2
J-K-3

GD2-1         GD2-3
GD2-2         GD2-3
GD2-3         J-K-2
GD4-1         GD2-4
GD4-2         GD2-4
GD2-4         J-K-3
ENTER EDIT REQUEST-CIR OR FAN
? *END    <-------------------------------      EXIT EDIT MODE.
ENTER COMMAND
?
INVALID COMMAND
```

****SAMPLE CIRCUIT - COMBINATIONAL NETWORK -- FULL ADDER

```
ENTER COMMAND
? NCT
ENTER CIRCUIT DESCRIPTION
? GO2-1 ABA ABB ABBA
? GO2-2 AB2 ABCICOUT
? GO2-3 CI1 CI2 SUM
? GD2-1 A   B   AB1
? GA2-1 A   AB1 ABA
? GA2-2 B   AB1 ABB
? GA2-3 A   B   AB2
? GA2-4 ABBACARIABCI
? GA2-5 ABBACIBACI1
? GA2-6 CARICIBACI2
? GD2-2 CARIABBACIBA
? EDIT
                      /--/  /--/  /--/  /--/  /--/  /--/  /--/
  1  GO2-1            ABA   ABB                     ABBA

  2  GO2-2            AB2   ABCI                    COUT

  3  GO2-3            CI1   CI2                     SUM

  4  GD2-1            A     B                       AB1

  5  GA2-1            A     AB1                     ABA

  6  GA2-2            B     AB1                     ABB

  7  GA2-3            A     B                       AB2

  8  GA2-4            ABBA  CARI                    ABCI

  9  GA2-5            ABBA  CIBA                    CI1

 10  GA2-6            CARI  CIBA                    CI2

 11  GD2-2            CARI  ABBA                    CIBA

? *END
ENTER COMMAND
? EXT
ENTER NO OF INPUTS AND NAMES
? 03A  B  CARI
```

NEW CIRCUIT COMMAND - REMOVES OLD CIRCUIT FROM TABLES AND ALLOWS NEW CIRCUIT TO BE ENTERED.

EXIT CIRCUIT BUILD MODE.

```
ENTER EXTERNAL VALUES FOR EACH NODE
? A    AØØØØØØ1111
? B    AØØØØ11ØØ11
? CARIAØØ1Ø1Ø1Ø1
ENTER COMMAND
? POP                                                         PRINT OUTPUT COMMAND.
ENTER PRINT REQUEST-G,C-AND PRINT ONLY REQUEST
? C                                                           PRINT EVERY CLOCK TIME.
ENTER NODES TO BE PRINTED
? A    B    SUM COUT                                          PRINT THESE NODES ONLY.
ENTER COMMAND
? SIM                                                         SIMULATE.
NO. OF CLOCK TIMES TO BE SIMULATED
? Ø6
PRESET EXTERNAL INPUT NODES-Y OR N
? N
       A         SUM
           B           COUT
INIT   Ø   Ø
CLK 1  Ø   Ø    Ø    Ø
CLK 2  Ø   Ø    1    Ø
CLK 3  Ø   1    1    Ø
CLK 4  Ø   1    Ø    1
CLK 5  1   Ø    1    Ø
CLK 6  1   Ø    Ø    1
CLK 7  1   1    Ø    1
CLK 8  1   1    1    1
END OF SIMULATION, RUN
ENTER CONT,REP,OR *END
? *END                                                        EXIT SIMULATION MODE.
ENTER COMMAND
? POP
ENTER PRINT REQUEST-G,C-AND PRINT ONLY REQUEST
? C                                                           PRINT EVERY CLOCK TIME
ENTER NODES TO BE PRINTED                                      AGAIN.
? A    B    CARISUM COUT
ENTER COMMAND
? SIM
NO OF CLOCK TIMES TO BE SIMULATED
? ØØ8
PRESET EXTERNAL INPUT NODES-Y, OR N
? N
       A          CARI     COUT
           B           SUM
INIT   Ø    Ø    Ø    1
CLK 1  Ø    Ø    Ø    Ø    Ø
CLK 2  Ø    Ø    1    1    Ø
CLK 3  Ø    1    Ø    1    Ø
CLK 4  Ø    1    1    Ø    1
CLK 5  1    Ø    Ø    1    Ø
CLK 6  1    Ø    1    Ø    1
CLK 7  1    1    Ø    Ø    1
CLK 8  1    1    1    1    1
END OF SIMULATION RUN
ENTER CONT,REP,OR *END
? *END
ENTER COMMAND
? POP
ENTER PRINT REQUEST-G,C-AND PRINT ONLY REQUEST
```

```
? 8                                         PRINT OUTPUT EVERY UNIT DELAY
ENTER NODES TO BE PRINTED                   TIME.
? *END
ENTER COMMAND                               DEFAULT VALUE FOR PRINT NODES
? SIM
NO OF CLOCK TIMES/TO BE SIMULATED           ALL NODES ARE TO BE PRINTED.
? 08
PRESET EXTERNAL INPUT NODES-Y OR N          ALL DEVICES HAVE DELAY OF 1.
? N
          ABA       ACBA      ACCI      CI1       SUM       B         CABI
               ABB       AB2       COUT      CI2       A         AB1       CIBA
INIT      0    0    0    0    0    0    0    0    0    0    0    1    0    1
CLK 1
          0    0    0    0    0    0    0    0    0    0    0    1    0    1
CLK 2
 DT 1     0    0    0    0    0    0    0    1    0    0    0    1    1    1
 DT 2     0    0    0    0    0    0    0    1    1    0    0    1    1    1
          0    0    0    0    0    0    0    1    1    0    0    1    1    1
CLK 3
 DT 1     0    1    0    0    0    0    0    0    1    0    1    1    0    1
 DT 2     0    1    1    0    0    0    0    0    0    0    1    1    0    1
 DT 3     0    1    1    0    0    0    1    0    0    0    1    1    0    1
 DT 4     0    1    1    0    0    0    1    0    1    0    1    1    0    1
          0    1    1    0    0    0    1    0    1    0    1    1    0    1
CLK 4
 DT 1     0    1    1    0    1    0    1    1    1    0    1    1    1    0
 DT 2     0    1    1    0    1    1    0    0    1    0    1    1    1    0
 DT 3     0    1    1    0    1    1    0    0    0    0    1    1    1    0
          0    1    1    0    1    1    0    0    0    0    1    1    1    0
CLK 5
 DT 1     1    0    1    0    0    1    0    0    0    1    0    1    0    1
 DT 2     1    0    1    0    0    0    1    0    0    1    0    1    0    1
 DT 3     1    0    1    0    0    0    1    0    1    1    0    1    0    1
          1    0    1    0    0    0    1    0    1    1    0    1    0    1
CLK 6
 DT 1     1    0    1    0    1    0    1    1    1    1    0    1    1    0
 DT 2     1    0    1    0    1    1    0    0    1    1    0    1    1    0
 DT 3     1    0    1    0    1    1    0    0    0    1    0    1    1    0
          1    0    1    0    1    1    0    0    0    1    0    1    1    0
CLK 7
 DT 1     1    1    1    1    0    1    0    0    0    1    1    0    0    1
 DT 2     0    0    1    1    0    1    1    0    0    1    1    0    0    1
 DT 3     0    0    0    1    0    1    1    0    1    1    1    0    0    1
 DT 4     0    0    0    1    0    1    0    0    1    1    1    0    0    1
 DT 5     0    0    0    1    0    1    0    0    0    1    1    0    0    1
          0    0    0    1    0    1    0    0    0    1    1    0    0    1
CLK 8
 DT 1     0    0    0    1    0    1    0    1    0    1    1    0    1    1
 DT 2     0    0    0    1    0    1    0    1    1    1    1    0    1    1
          0    0    0    1    0    1    0    1    1    1    1    0    1    1
END OF SIMULATION RUN
ENTER CONT.REP.OR *END
? *END
ENTER COMMAND
? EDIT                                      EDIT MODE.
ENTER EDIT REQUEST-CIR OR FAN
? FAN                                       PRINT FANOUT OF ALL DEVICES.
          FANOUT
DEVICE    FANOUT DEVICES
GD2-1     GA2-4   GA2-5   GD2-2
```

```
GA2-2      GO2-1
GA2-3      GO2-2
GA2-4      GO2-2
GA2-5      GO2-3
GA2-6      GO2-3
GO2-2      GA2-5  GA2-6
ENTER EDIT REQUEST-CIR OR FAN
? *END
```

EXIT EDIT MODE.

```
ENTER COMMAND
NDEV
```

NEW DEVICE COMMAND - DEFINE A NEW DEVICE TO THE SIMULATOR.

```
ENTER NEW DEVICE NAME - 3 CHARS
? UFF
ENTER NO OF INPUTS - 1 DIGIT
? 2
ENTER NO OF OUTPUTS - 1 DIGIT
? 2
```

ONLY 2 INPUTS DEFINED - FOR USER DEFINED DEVICES THE CLOCK INPUT IS NOT CONSIDERED AS AN INPUT. WHEN SIMULATING ONLY WHEN CLOCK EDGE IS CORRECT ARE INPUT VALUES CHECKED AGAINST TRUTH TABLE VALUES.

```
ENTER DELAY -2 DIGIT
? Ø1
EDGE SENSITIVE DEVICE-YES OR NO
? Y
ENTER TRIGGERING EDGE-EG. 1Ø OR Ø1
? 1Ø
ENTER TRUTH TABLE
? ØØXX
? 1ØØ1
? 11XX
? Ø11Ø
ENTER COMMAND
? NCT
```

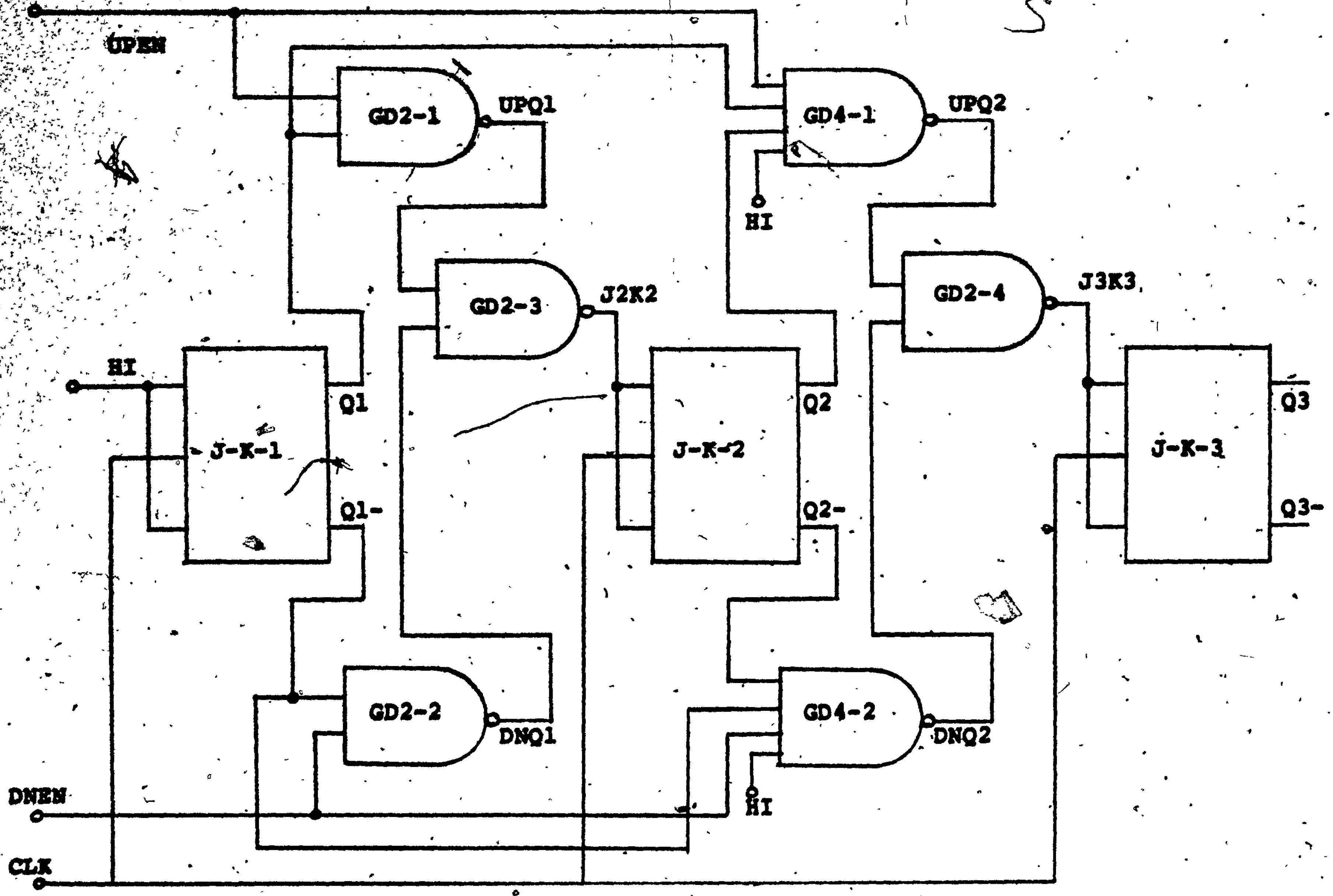
NEW CIRCUIT COMMAND - NOW ENTERING NEW CIRCUIT INTO SYSTEM.

```
ENTER CIRCUIT DESCRIPTION
? UFF-1 A1  A2  CLK             Q1  Q1-
? *END
ENTER COMMAND
? EXT
ENTER NO OF INPUTS AND NAMES
? Ø3A1  A2  CLK
ENTER EXTERNAL VALUES FOR EACH NODE
? A1  AØØØØØØ1111ØØØØØ1111
? A2  AØØØØØØØØØØ11111111
? CLK RØ2Ø1
```

2 METHODS OF ENTERING INPUT CLOCK VALUES - USING A AND R FORMAT.

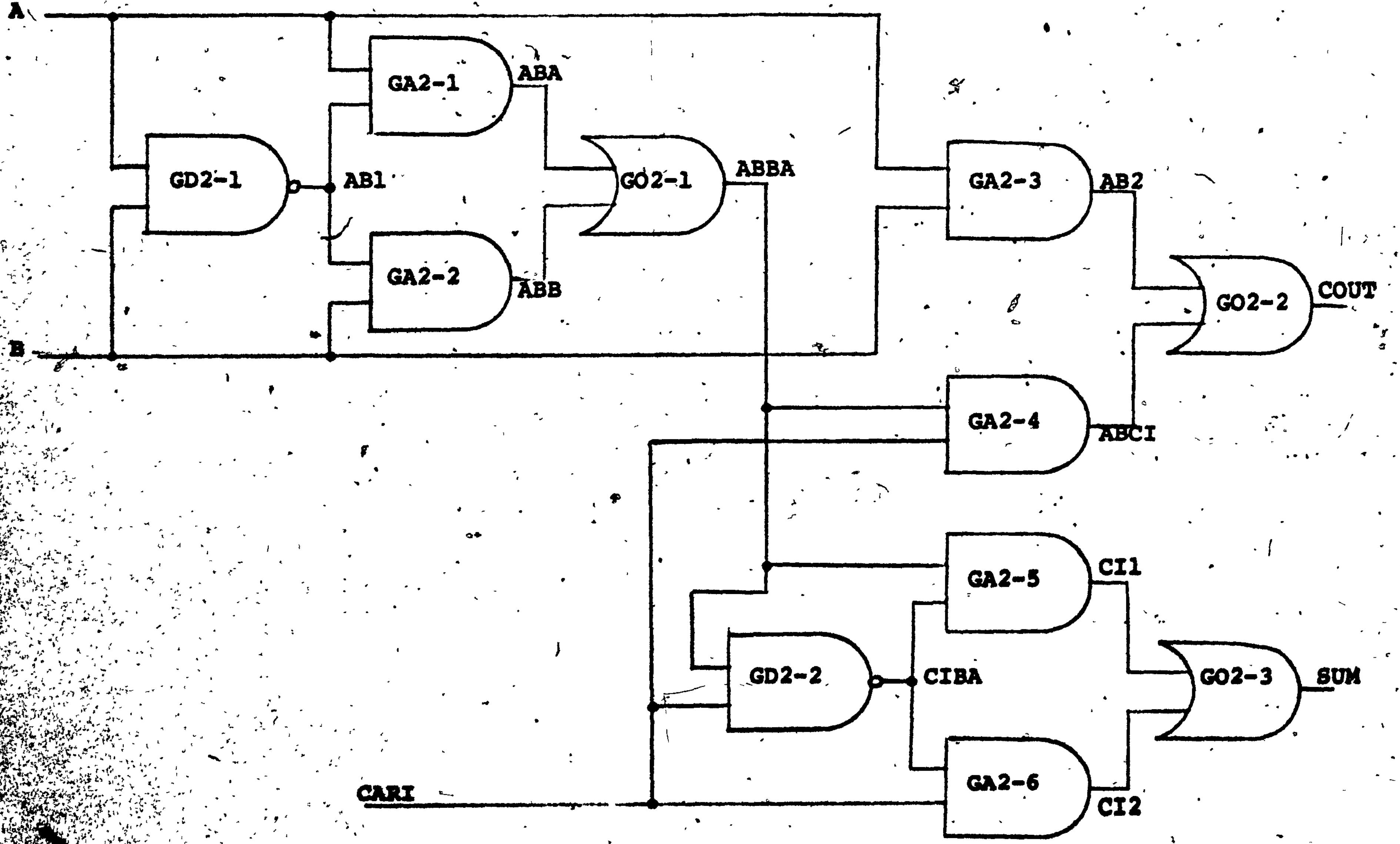
```
ENTER COMMAND
? SIM
NO OF CLOCK TIMES TO BE SIMULATED
? 1Ø
PRESET EXTERNAL INPUT NODES-Y OR N
? N
        A1          CLK         Q1-
              A2          Q1
INIT    Ø     Ø     Ø     1     Ø
CLK 1   Ø     Ø     Ø     1     Ø
CLK 2   Ø     Ø     1     1     Ø
CLK 3   Ø     Ø     Ø     X     X
CLK 4   Ø     Ø     1     X     X
CLK 5   1     Ø     Ø     X     X
CLK 6   1     Ø     1     X     X
CLK 7   1     Ø     Ø     Ø     1
CLK 8   1     Ø     1     Ø     1
CLK 9   Ø     1     Ø     Ø     1
CLK1Ø   Ø     1     1     Ø     1
END OF SIMULATION RUN
ENTER CONT,REP,OR *END
? *END
ENTER COMMAND
? END
```

SIMULATION OUTPUTS MATCH THOSE DEFINED IN TRUTH TABLE.



SYNCHRONOUS BINARY UP-DOWN COUNTER

FIGURE A.1



COMBINATIONAL NETWORK - FULL ADDER

FIGURE A.1 (CONT.)

BIBLIOGRAPHY

1. Modeling and Simulation of Digital Networks,
   Robert P. Larsen and M. Morris Mano,
   Communications of the ACM, Vol. 8, No. 5, May, 1965,
   Pages 308 - 312.

2. A Software System for Diagnostic Test Generation
   and Simulation of Large Digital Systems,
   Stephen A. Szygenda; University of Missouri
   National Electronics Annual Conference -1969, Page 657-662.

3. A Model And Implementation of a Universal Time
   Delay Simulator For Large Digital Nets,
   Stephen A. Szygenda, David M. Rousse, E.W. Thompson,
   University of Missouri,
   Spring Joint Computer Conference, 1970, Pages 207 - 216.

4. Models for Functional Simulation of Large Digital
   Computing Systems,
   D. M. Rousse and S.A. Szygenda, University of Missouri,
   Digest Record of Joint Conference on Mathematical
   and Computer-aided Design, 1969, Page 363.

5. Computers Come to the Aid of the Logic Designer,
   R.J. Diephuis,
   Electronics - March, 1971, Pages 50 - 54.

6. Computer-Aided Design: Simulation of Digital
   Design Logic,
   C. G. Hays,
   IEEE Transactions on Computers, Vol. C-18, No. 1,
   January, 1969, Pages 1 - 10.

7. Digital System Design Automation - A Method For Designing A Digital System As A Sequential Network System,
Giovanni B. Gerace,
IEEE Transactions on Computers, Vol. C-17, No. 11, November, 1968, Pages 1044 - 1061.

8. Simulation Of Large Asynchronous Logic Circuits Using An Ambiguous Gate Model,
S.G. Chappell and S.S. Yau,
Fall Joint Computer Conference, 1971, Pages 651 - 661.

9. Exclusive Simulation Of Activity In Digital Networks,
E. G. Ulrich,
Communications of the ACM, Vol. 12, No. 2, February 1969, Pages 102 - 110.

10. A Digital Logic Simulator For Teaching Purposes,
C.S. Wallace, Monash University, Victoria, Australia,
Proceedings of the Fourth Australian Computer Conference, 1969, Pages 215 - 219.

11. Optimizing Bit-Time Computer Simulation,
Esse H. Katz,
Communications of the ACM, Vol. 6, No. 11, November, 1963, Pages 679 - 684.

12. Compiler Level Simulation Of Edge Sensitive Flip Flops,
James T. Cain and Marlin H. Mickle,
Spring Joint Computer Conference, Vol. 30, 1967, Pages 757 - 759.

13. Digital Simulation Of Digital Systems,
Tobias H. Trygar and Marlin H. Mickle,
Simulation and Modeling Conference,
University of Pittsburg, April, 1969, Pages 70 - 76.

14. The CADSS Simulator,
Ernest A. Franke, Ph.D.
Simulation and Modeling Conference,
University of Pittsburg, April, 1969, Pages 79 - 83.

15. Bodis - A Block Oriented Digital Simulator,
Dr. Carl F. Evert, J. DeJean, M. Hartke,
Simulation and Modeling Conference,
University of Pittsburg, April, 1969, Pages 84 - 94.

16. An Asynchronous Look-Ahead Digital Systems Simulator,
Samuel B. Shipkgvitz,
Simulation and Modeling Conference,
University of Pittsburg, April, 1969, Pages 95 - 106.

17. Modelling And Simulation Of Digital Networks,
Robert P. Larsen and M. Morris Mano,
Communications of the ACM, Vol. 8, No. 5, May, 1965, Pages 308 - 312.

18. A Programming Language For Simulating Digital Systems,
Robert M. McClure,
Journal of the ACM, Vol. 12, No. 1, January, 1965, Pages 14 - 22.

19. Design Checkout Of Logic Networks,
Information Processing in Japan, 10, 1970,
Pages 90 - 96.

20. Design Automation Of Digital Systems,
Vol. 1, Theory and Techniques,
Melvin A. Breuer.

21. Pert as an Aid to Logic Design,
T. I. Kirkpatrick and N. R. Clark,
IBM Journal, March, 1966, Pages 135 - 141.