HAZARD DETECTION BY A QUINARY SIMULATION OF LOGIC DEVICES WITH BOUNDED PROPAGATION DELAYS

Daniel W. Lewis
General Electric Company

ABSTRACT

Effective logic simulation programs must consider device propagation delays to be bounded values. This requires that the logic devices be simulated by models which use a multi-valued logical algebra.

A quinary algebra is developed and employed in special algorithms which not only accurately predict the behavior of a logic circuit for all values of delay, but also detect the possibility of latent hazards and race conditions.

A sample problem is simulated, and conclusions drawn.

INTRODUCTION

Several logic simulation programs have been developed to assist in the design and analysis of switching circuits, but these programs are not enjoying wide usage by logic designers. Basically, the problem lies in the fact that the simulation models of the logic devices have not been complete enough to reliably predict the behavior of a circuit for all combinations of device delays.

Users of logic simulation programs typically select the maximum propagation delay specification for each device, assuming that this is the "worst-case" situation. However, it can easily be shown that other combinations of delays may actually be the "worst-case". For example, consider the circuit of Figure 1. Assume that it is part of a larger circuit, and that for the time period of consideration:

$$a = 0 \quad \text{and} \quad b = 0.$$

Furthermore, assume that all devices are selected from a batch for which the propagation delay lies in the range:

$$5 \leq \delta \leq 15 \text{ nanoseconds.}$$

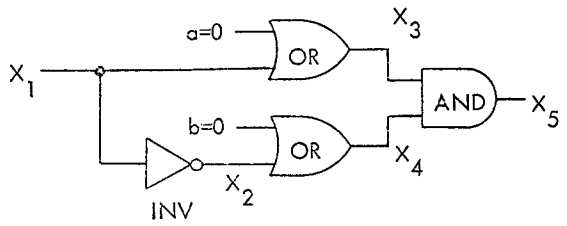


Figure 1. Hazard Example

If the circuit is simulated using the maximum delay for all devices, then there is no output pulse for the given input transition as shown in Figure 2. However, Figure 3 illustrates an assortment of delays which yields an output pulse from the circuit. This situation is known as a "hazard", and is obviously an important consideration in the

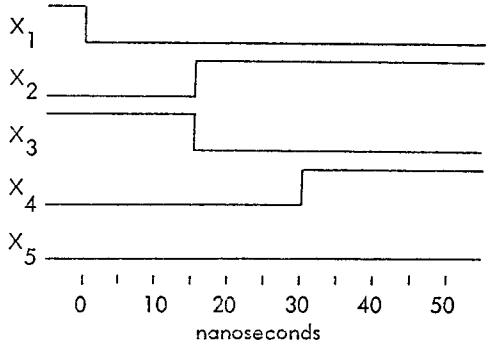


Figure 2. Timing Diagram for Maximum Delays

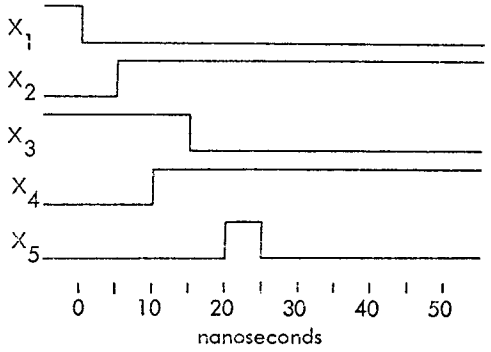


Figure 3. Timing Diagram for Random Delays

design of logic networks since hazards can cause a circuit to malfunction. Several simulation runs could be made with random choices of delays to try to find such hazards, but it is doubtful that all such delay-dependent situations would be uncovered if given a large logic network. Therefore, it is obvious that any useful and realistic simulation must be based on device models which simultaneously consider all possible values of propagation delay. In other words, given a device with an input transition, the model must predict the earliest and the latest possible time that the device output could change. The models developed in this report (and the corresponding simulation program) reliably locate all hazards by treating propagation delays as bounded values.

MODELING DEVICE CHARACTERISTICS

Actual logic devices have two main characteristics: their specific logic function, and their inherent propagation delay. Thus any modeling can be separated into two parts as shown in Figure 4. Such a model assumes that all delays are associated with device outputs, and are therefore lumped at those points. This infers that delays due to wire length are negligible compared to delays due to logic devices.

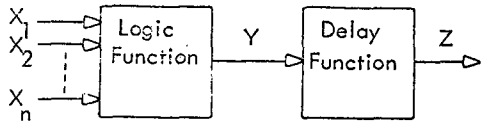


Figure 4. Basic Device Model

Part 1: Logic Function

Assume that the input logic signals to a particular device have been sampled at some instant of time so that their values are stable. These inputs are then applied to the first part of the modeling process which is a time-independent evaluation of the logic function of the device.

For example, if the device is an OR gate, the logic function is given by

$$Y = X_1 \text{ OR } X_2 \text{ OR } \ldots\ldots X_n$$

where the sampled input signals are denoted by $X_1, X_2, \ldots\ldots X_n$.

If the device is a T flip-flop, then the logic function is given by

$$Y^n = T \text{ EXOR } Y^{n-1}$$

where the superscript n denotes the n^{th} calculation of Y, and T = 1 means that the flip-flop has been triggered by a clock pulse.

Although these functions are normally evaluated using a Boolean algebra of binary variables[1], other algebras of multi-valued variables are sometimes more useful.[2, 3, 4, 5c, 6, 7, 8]

Part 2: Delay Function

The second part of the modeling process uses the propagation delay parameter(s) of the particular device and the results accumulated from Part 1 to compute the behavior of the device at its output terminal.

Fixed Propagation Delay

Every physical logic device will exhibit a non-zero amount of propagation delay. If the delay of a single device is known, its delay characteristic can be described by

$$Z(t) = Y(t-\delta)$$

where

$$Y(t) = \{F\ X_1(t), X_2(t), \ldots\ldots X_n(t)\}$$

and δ is the propagation delay of the device.

In a simulation program, time is a variable which changes in discrete steps so that

$$t_i = t_{i-1} + \Delta t.$$

Therefore, if we compute

$$n = \left[\frac{\delta}{\Delta t}\right]$$

to the nearest whole integer, then the delay characteristics can be expressed incrementally by

$$Z_i = Y_{i-n}.$$

This relationship is given in block diagram form by Figure 5.

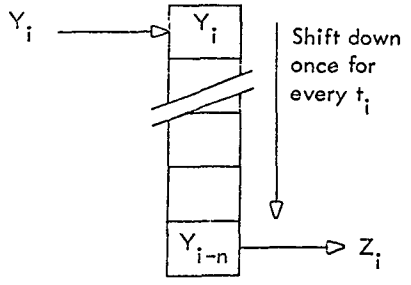


Figure 5. Fixed Delay Function

The delay function for fixed propagation delay can be thought of as a storage vector for the values calculated by the logic function. For each t_i the values in the vector are shifted down, and the new

value of Y_i is placed at the top. The value at the bottom of the storage vector is Y_{i-n}, or the present state of the output logic signal, Z_i.

Bounded Propagation Delay [2][5a]

The consideration of a propagation delay as a bounded value requires that the design and analysis allow for a period of time between the earliest and the latest transition of the output during which the output of the device is indeterminate. For example, consider the example of Figure 6.

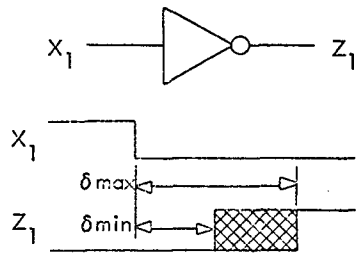


Figure 6. Bounded Delay Example

The output signal of the inverter is shown relative to the given input signal. The period of time when the output is indeterminate is shown as the cross-hatched region defined by the difference between δ_{min} and δ_{max}, the minimum and maximum propagation delay specifications. Note, however, that the signal transition has a direction (obviously there are two possible directions) impled by the signal values at either end of the transition.

In order to simulate bounded propagation delays, one value of Z_i must be calculated for every possible delay. For example, let

$$n_1 = \left\lceil \frac{\delta\,min}{\Delta t} \right\rceil \text{ and } n_2 = \left\lceil \frac{\delta\,max}{\Delta t} \right\rceil .$$

Thus there will be n_2-n_1+1 possible solutions for Z_i. These solutions are then used to determine a value which is representative of all of the possible solutions.

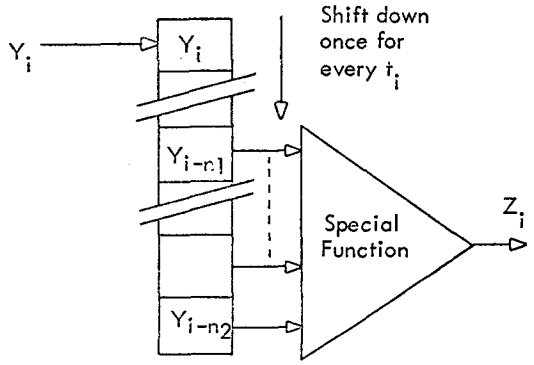


Figure 7. Bounded Delay Function

Just as for the fixed delay function, the bounded delay function of Figure 7 uses a storage vector for the values calculated by Part 1 of the modeling process. Similarly, for each t_i the values in the vector are shifted down, and the new value of Y_i is placed at the top. However, now all of the values from Y_{i-n_1} to Y_{i-n_2} are inputs to a special function which determines a value for Z_i which is representative of all possibilities.

The set of normal binary values (0=FALSE; 1=TRUE) is no longer adequate to represent the value of Z_i. Now higher-order number systems with new value assignments must be used when Z_i is not known to be exactly "TRUE" or exactly "FALSE"*.

QUINARY LOGICAL ALGEBRA

The logical algebra normally used for logic design is known as Boolean algebra[1]. It involves variables (X, Y, etc.) which may have values from the set:

$$X \in \{0, 1\}$$
where 0 = FALSE
and 1 = TRUE.

This representation implies that logic signals may only be classified as either exactly TRUE or exactly FALSE. In order to study the behavior of logic circuits during periods when their values are not exactly defined, other mathematical representations which allow non-binary values are required. The quinary logical algebra defined in this report meets this requirement by using five symbols to represent various logic conditions as shown in Figure 8.

Quinary symbol	Interpretation
0	Equivalent to state 0 of the two-valued switching algebra
1	Equivalent to state 1 of the two-valued switching algebra
0/1	The representation of a logic signal during a period of time allowed for a transition from state 0 to state 1
1/0	The representation of a logic signal during a period of time allowed for a transition from state 1 to state 0
½	The representation of a logic signal not covered by any of the above (I.e., hazards and race conditions)

Figure 8. Quinary Symbol Interpretations

*Note that this requires the logic function to handle non-binary input values.

The quinary logical operators are based on the following set of axiomatic rules, similar to those of binary Boolean algebra:

$$0\ \text{AND}_5\ X = 0 \qquad 0\ \text{OR}_5\ X = X$$
$$1\ \text{AND}_5\ X = X \qquad 1\ \text{OR}_5\ X = 1$$
$$X\ \text{AND}_5\ X = X \qquad X\ \text{OR}_5\ X = X$$

where $X \in \{0,\ 1,\ 0/1,\ 1/0,\ \text{and}\ \frac{1}{2}\}$.

This set defines the quinary AND and the quinary OR operations for nineteen of the twenty-five ordered pairs of quinary values. The remaining six pairs are those among the quinary values 0/1, $\frac{1}{2}$, and 1/0. The AND and OR operations for these pairs are indeterminate and are thus assigned values of $\frac{1}{2}$. The complete AND and OR operators of quinary logical algebra are summarized in the following truth tables:

X \ Y	0	0/1	½	1/0	1
0	0	0	0	0	0
0/1	0	0/1	½	½	0/1
½	0	½	½	½	½
1/0	0	½	½	1/0	1/0
1	0	0/1	½	1/0	1

$X\ \text{AND}_5\ Y$

X \ Y	0	0/1	½	1/0	1
0	0	0/1	½	1/0	1
0/1	0/1	½	½	½	1
½	½	½	½	½	1
1/0	1/0	½	½	1/0	1
1	1	1	1	1	1

$X\ \text{OR}_5\ Y$

Figure 9. Quinary AND and OR Operators

It can be shown by an exhaustive test of all cases that these relationships obey the associative and commutative laws.

The remaining logical operation of complementation is defined by the following truth table.

X	½
0	1
0/1	1/0
½	½
1/0	0/1
1	0

$\text{NOT}_5\ X$

Figure 10. Quinary NOT Operator

No attempt has been made to define these functions in terms of Post algebra[9]. Rather, the algebra is meant to be a "table-look-up" process based on rather obvious and intuitive relationships, and it is designed to yield a value of $\frac{1}{2}$ whenever there is the possibility of a hazard.

SIMULATION ALGORITHMS

A logic simulation program is a repetitive process involving certain algorithms which are the implementations of the models used to represent the logic devices. Three algorithms have been developed here to implement the models of combinational logic, flip-flops, and bounded delays.

Combinational Logic Algorithm

Since quinary logical algebra is both associative and commutative, then the specific logic function of any combinational device may be separated into inverters and two-input AND and OR gates. Then the truth tables of the algebra may be used to evaluate the logic function, and determine the input to the bounded delay function. An example of such a decomposition is illustrated in Figure 11.

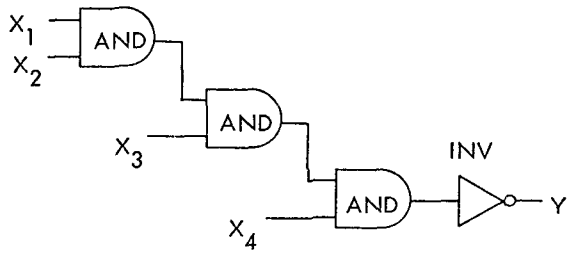


Figure 11. Decomposition of 4-Input NAND Gate

Flip-Flop Algorithm

There are many types of flip-flops currently available to the logic designer, such as the J-K, R-S, T, and D types. Each of these can be thought of as a basic memory cell and some associated combinational logic. Consider a basic memory cell which has a clock input that causes the cell to change state from its present state to the complement of its present state every time the clock input is pulsed. Furthermore, let the associated combinational logic define the clock for the cell as a function of the cell's present state and any inputs. Then, for example, a D type flip-flop can be realized by a logic circuit such as the one shown in Figure 12. A quick examination of the circuit will show that the flip-flop realization behaves as a D type, since the output Q takes the value of the input D after every clock pulse.

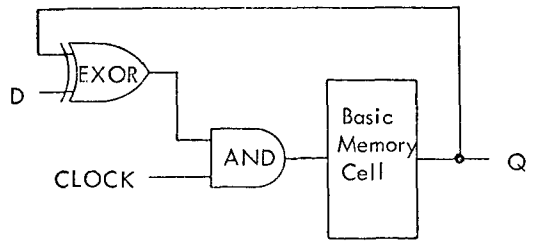


Figure 12. Realization of D Type Flip-Flop

Some flip-flops also have a pair of direct inputs which override all other inputs. These should be added to the basic memory cell as shown in Figure 13.

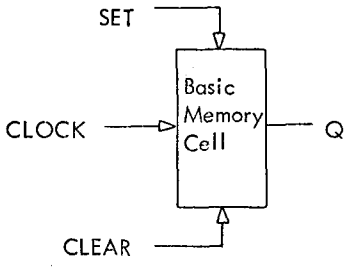


Figure 13. Improved Memory Cell

This basic memory cell with some associated combinational logic can now be used to realize any type of flip-flop similar to the use of the combinational logic algorithm to realize any type of combinational logic device.

The study of the simulation algorithm of this model of a basic memory cell can be simplified if the input states are separated into two cases:

Case 1: SET $\neq$ 0 and/or CLEAR $\neq$ 0.

In this case the direct inputs (SET and CLEAR) override any clock input pulses. Then the output (to be used as an input to the bounded delay algorithm) behaves according to the flow chart of Figure 14.

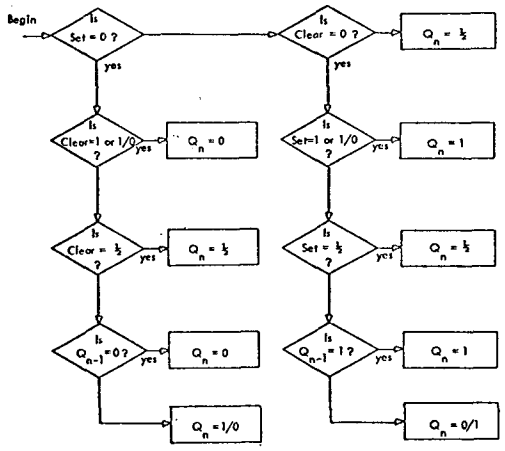


Figure 14. Flip-Flop Algorithm (SET $\neq$ 0 and/or CLEAR $\neq$ 0)

As shown by the flow chart, if either the SET input or the CLEAR input is 0, then the other must not be 0, and the value of the output is determined by the value of the latter input. However, if neither of the inputs is 0, then a "race" condition exists since the flip-flop is simultaneously being set and cleared. In such a situation, the output is indeterminate ($\frac{1}{2}$).

The derivation of this part of the algorithm is based on rather obvious and intuitive relationships which characterize the behavior of SET and CLEAR inputs in general.

Case 2: SET = 0 and CLEAR = 0.

In this case the direct inputs are disabled, and the behavior of the basic memory cell is a function of its previous state (Q_{n-1}) and the CLOCK input. Therefore, the previous value ($CLOCK_{n-1}$) and the present value ($CLOCK_n$) of the clock must be compared to determine if a transition* has toggled the flip-flop. Figure 15 is a flow chart of this portion of the algorithm:

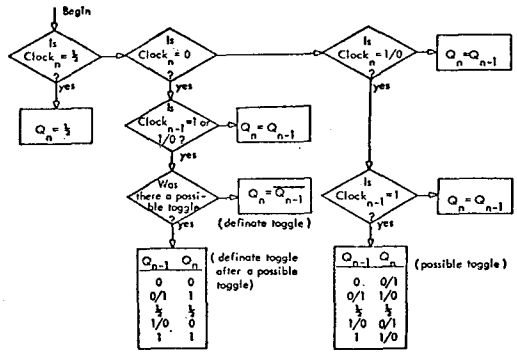


Figure 15. Flip-Flop Algorithm (SET and CLEAR = 0)

*It is assumed here that only negative transitions will toggle the flip-flop.

Considering Figure 15, it should be obvious that an indeterminate ("$\frac{1}{2}$") CLOCK must cause the output to be indeterminate. Furthermore, the state of the flip-flop will continue to be indeterminate until it is redefined by either the SET or CLEAR input. If the flip-flop is toggled by a CLOCK transition, the output is determined according to whether the toggle was a definite toggle or only a possible toggle as shown in the figure. A possible toggle imples that there is a possibility that the flip-flop output may have changed, but such a change is not guaranteed.

Bounded Delay Algorithm

As discussed previously, the propagation delay of a logic device should be simulated as a bounded value. This may be accomplished by an algorithm based on Figure 7. In such an algorithm there are two main parts. First there is a storage vector for the values calculated by Part 1 of the modeling process (Y_i through Y_{i-n_2}), and secondly, the values of Y_{i-n_1} through Y_{i-n_2} are inputs to a special function which selects a value for Z_i which is representative of all of the n_2-n_1+1 possible solutions.

The selection of a value for Z_i depends on the order of the values of Y_{i-n_1} through Y_{i-n_2} as well as their actual values. There are four types of sequences of the values of Y that can occur, and these determine the value of Z_i in the following manner:

TYPE 1 (Identical inputs): If all of the values of Y_{i-n_1} through Y_{i-n_2} are the same, then Z_i must take that value.

TYPE 2 (Indeterminate inputs): If any of the values of Y_{i-n_1} through Y_{i-n_2} is "$\frac{1}{2}$", then the value of Z_i is also "$\frac{1}{2}$".

TYPE 3 (Ordered inputs): If the values of Y, taken in order from Y_{i-n_2} to Y_{i-n_1}, correspond to an increasing (decreasing) sequence such as "0"-"0/1"-1" ("1"-"1/0"-"0") with any one of these values either repeated or missing, then the value of Z_i is "0/1" ("1/0").

TYPE 4 (Unordered inputs): If the values of Y_{i-n_1} through Y_{i-n_2} do not correspond to an increasing or decreasing sequence, then the value of Z_i is "$\frac{1}{2}$".

These types yield a value for Z_i which is worst-case representation of the values of Y_{i-n_1} through Y_{i-n_2}, each of which is a possible value for Z_i, corresponding to each of the possible values of propagation delay of the device. This is based on rather obvious and intuitive relationships, similar to the definition of the quinary logical algebra.

Consider the examples of Figure 16.

Y_{i-n_1}	0	0/1	.	.	1	1	0	0
.	0	0/1	.	1/0	1	1	0	0
.	0	0/1	.	.	0/1	1	0	1
.	0	0/1	½	.	0/1	1	1/0	1
.	0	0/1	.	0/1	0/1	0	1	0
Y_{i-n_2}	0	0/1	.	.	0	0	1	1
Z_i	0	0/1	½	½	0/1	0/1	1/0	½

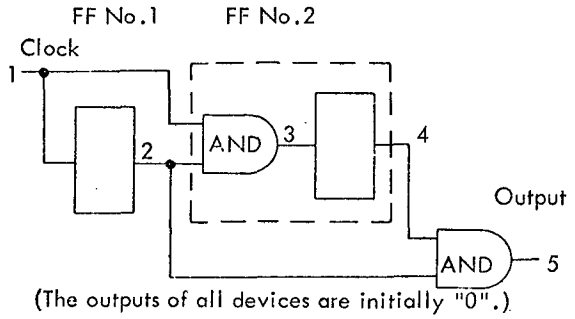
Figure 16. Bounded Delay Algorithm Examples

TYPICAL SIMULATION PROBLEM

Suppose it is necessary to design a logic circuit which will provide a hazard-free output signal after three clock pulses have occurred, assuming that the circuit is initially reset.

Synchronous Binary Counter

Consider the circuit illustrated by Figure 17. This is a synchronous binary counter with the output provided by an AND gate which decodes the "11" count of the flip-flops. The first flip-flop



(The outputs of all devices are initially "0".)

Figure 17. Synchronous Binary Counter

toggles once for every clock pulse, but the second flip-flop has an enable input which allows it to toggle only for those clock pulses which occur while the first flip-flop is set. The delay characteristic of every device can be described by

minimum propagation delay = 0 nsec,

maximum propagation delay = 10 nsec.

In order to simulate this circuit, however, gate number three must have a zero maximum delay as well. It is part of the realization of the second flip-flop, and the propagation delay of that device is specified by the delay characteristic of the basic memory cell.

The output of the computer simulation program is given in Figure 18. It reveals that there is a hazard at the output of the decoding AND gate*. Close examination of the simulation will show that this is due to simultaneous opposite transitions at the two inputs of the gate. Obviously, another design is required.

*It has been shown [10] that a value of 1/2 may occur in the simulation if and only if a hazard or race condition exists in the logic circuit.

TIME		1	2	3	4	5
0.		0	0	0	0	0
0.1000E-07		0	0	0	0	0
0.2000E-07		0	0	0	0	0
0.3000E-07		0	0	0	0	0
0.4000E-07		0	0	0	0	0
0.5000E-07		1	0	0	0	0
0.6000E-07		1	0	0	0	0
0.7000E-07		1	0	0	0	0
0.8000E-07		1	0	0	0	0
0.9000E-07		1	0	0	0	0
0.1000E-06		0	0/1	0	0	0
0.1100E-06		0	1	0	0	0
0.1200E-06		0	1	0	0	0
0.1300E-06		0	1	0	0	0
0.1400E-06		0	1	0	0	0
0.1500E-06		1	1	1	0	0
0.1600E-06		1	1	1	0	0
0.1700E-06		1	1	1	0	0
0.1800E-06		1	1	1	0	0
0.1900E-06		1	1	1	0	0
0.2000E-06	*	0	1/0	0	0/1	0.5
0.2100E-06	*	0	0	0	1	0.5
0.2200E-06		0	0	0	1	0
0.2300E-06		0	0	0	1	0
0.2400E-06		0	0	0	1	0
0.2500E-06		1	0	0	1	0
0.2600E-06		1	0	0	1	0
0.2700E-06		1	0	0	1	0
0.2800E-06		1	0	0	1	0
0.2900E-06		1	0	0	1	0
0.3000E-06		0	0/1	0	1	0/1
0.3100E-06		0	1	0	1	0/1
0.3200E-06		0	1	0	1	1
0.3300E-06		0	1	0	1	1
0.3400E-06		0	1	0	1	1
0.3500E-06		1	1	1	1	1
0.3600E-06		1	1	1	1	1
0.3700E-06		1	1	1	1	1
0.3800E-06		1	1	1	1	1
0.3900E-06		1	1	1	1	1
0.4000E-06		0	1/0	0	1/0	1/0
0.4100E-06		0	0	0	0	1/0
0.4200E-06		0	0	0	0	0
0.4300E-06		0	0	0	0	0
0.4400E-06		0	0	0	0	0
0.4500E-06		0	0	0	0	0

Figure 18. Simulation of Synchronous Binary Counter

CONCLUSIONS

Hazards and race conditions can occur in switching networks whenever the propagation delay characteristics of the logic devices are ignored. Furthermore, it is not sufficient to merely consider the maximum possible delay of each device; rather, the entire distribution of delays from the minimum to the maximum must be considered. The bounded delay algorithm described in this report meets this requirement.

The practicality of the quinary logical algebra has been demonstrated by transforming the simulation algorithms into an actual Fortran program, and it has been shown that this program can actually be used as an efficient aid to detect hazards, while verifying the design of a switching network.

BIBLIOGRAPHY

(1) Boole, G., "An Investigation of the Laws of Thought", London, 1854 (reprinted by Dover Publications, New York)

(2) Eichelberger, E.B., "Hazard Detection in Combinational and Sequential Switching Circuits", IBM System Journal, Vol. 9, No. 2, pp. 90-99, March 1965

(3) Yoeli, M. and Rinon, S., "Application of Ternary Algebra to the Study of Static Hazards", JACM, Vol. 11, No. 1, January 1964, pp. 84-87

(4) Langdon, G.G., "Analysis of Asynchronous Circuits Under Different Delay Assumptions", IEEE Transactions on Computers, Vol. C-17, No. 12, December 1968, pp. 1131-1143

(5) Unger, S.H., "Asynchronous Sequential Switching Circuits", Wiley-Interscience, 1969

(5a) pp. 118 (concept of bounded delays)

(5b) pp. 138-139 (consideration and relative importance of rise times)

(5c) pp. 179-183 (ternary logic test for hazards)

(6) Metze, G.A., "An Application of Multivalued Logic Systems to Circuits", Proceedings of Symposium on Circuit Analysis, pp. 11-1 through 11-14, University of Illinois, Urbana, 1955

(7) Muller, D.E., "Treatment of Transition Signals in Electronic Switching Circuits by Algebraic Methods", IRE Transactions on Electronic Computers, Vol. EC-8, No. 3, pp. 401, September 1959

(8) Jephson, J.S., McQuarrie, R.P., and Vogelsberg, R.E., "A Three-value Computer Design Verification System", IBM System Journal, Vol. 8, No. 3, pp. 178-188, 1969

(9) Post, E.L., "Introduction to a General Theory of Elementary Propositions", American Journal of Mathematics, Vol. 43, pp. 163-185, 1921

(10) Lewis, D.W., "Hazard Detection by a Quinary Simulation of Logic Devices with Bounded Propagation Delays", Master's Thesis, Syracuse University, January 1972

(11) Unger, S.H., "Hazards and Delays in Asynchronous Sequential Switching Circuits", IRE Transactions on Circuit Theory, Vol. CT-6, pp. 12-25, March 1959

(12) Fantauzzi, Guiseppe, "An Algebraic Model for the Analysis of Logical Circuits", Societa Italiana Telecomunicazioni, Siemans, S.p.A., an unpublished report