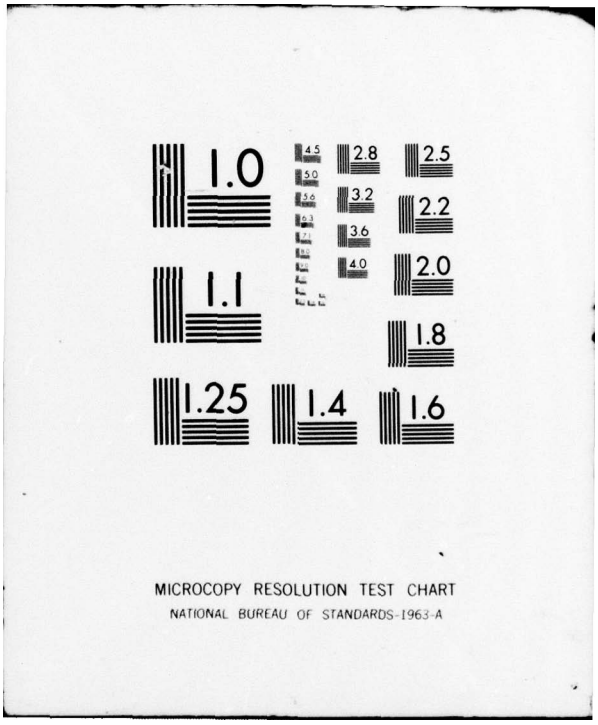
AD-A052 616 TEXAS UNIV AT AUSTIN ELECTRONICS RESEARCH CENTER F/G 9/2
DETECTION OF STATIC AND DYNAMIC HAZARDS IN LOGIC NETS,(U)
JUN 77 A K BOSE, S A SZYGENDA F44620-76-C-0089
UNCLASSIFIED AFOSR-TR-78-0689 NL

1 OF 1
AD A052816

END
DATE FILMED
5-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFOSR-TR-78-0689	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): Detection of Static and Dynamic Hazards in Logic Nets	5. TYPE OF REPORT & PERIOD COVERED: INTERIM
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Ajoy K. Bose and S. A. Szygenda	8. CONTRACT OR GRANT NUMBER(s): F44620-76-C-0089
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Electronics Research Center, The University of Texas at Austin, Austin, Texas 78712	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F, 2305/A9
11. CONTROLLING OFFICE NAME AND ADDRESS: AF Office of Scientific Research (NE), Building #410, Bolling AFB, D.C. 20332	12. REPORT DATE: June 1977
	13. NUMBER OF PAGES: 5 pp.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

multi-level simulation, static hazards, dynamic hazards

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper discusses an algorithmic procedure for the detection of static and dynamic hazards in a digital simulation environment. The procedure uses a signal representation scheme that decomposes multi-level signals to a binary form which are used along with a binary description of the network to realize hazard detection. Moreover this procedure can also be used to realize multi-level simulation andprovides the user with various options about the accuracy that can be obtained at varying costs.

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE

403 789

AFOSR-TR- 78-0689

DETECTION OF STATIC AND DYNAMIC HAZARDS IN LOGIC NETS

Ajoy K. Bose and S. A. Szygenda
Electrical Engineering Department
The University of Texas-Austin
Austin, Texas 78712

Summary

This paper discusses an algorithmic procedure for the detection of static and dynamic hazards in a digital simulation environment. The procedure uses a signal representation scheme that decomposes multi-level signals to a binary form which are used along with a binary description of the network to realize hazard detection. Moreover this procedure can also be used to realize multi level simulation and provides the user with various options about the accuracy that can be obtained at varying costs.

1. Introduction

The basic techniques discussed in this paper provide a methodology for multi-level simulation of logic nets. Based on a two valued (binary) description of the net, a simulation procedure is proposed which achieves 5 level[1] and 9 level[2,3] simulation of the net. The detection of static and dynamic hazards provides the logical extension from 5 level to 9 level simulation.

Though 5 level simulation is fairly commonplace, it is generally restricted to simple gates and flip-flops. The immediate problem faced, in the handling of complex devices, is to incorporate in the simulator a complete description of the behavior of the device for all possible conditions that could exist at its inputs. In a 5 level simulator, a device with 5 independent inputs could have 3125 (5^5) different input conditions. The specification of an output for each and every input condition is not only a lengthy process, it is also extremely error prone. Over and above, the use of boolean algebra as a tool is denied because the unknown signal condition does not comply with its postulates.[4] These problems worsen substantially in the case of nine level simulation.

With the wide variety of devices available to today's logic designer, the task of keeping him provided with accurate multi-level simulators has become increasingly difficult. Over and above the problems discussed in the preceding paragraph, there are problems associated with providing the user with a suitable means for describing his devices to the simulator and many others.

Hazard detection, which is discussed in this paper, forms a part of an overall effort to investigate some of these problems. In order to provide a better understanding of the overall objectives, simulation techniques have also been discussed.

The paper initially assumes the working environment of a 5 level simulator. The isolation of the hazard detection procedure to form an independent process is straightforward.

2. Signal Representation

The circuit to be tested can be described using any conventional binary technique. The simulation and hazard detection procedures require only a binary description and thus a binary truth table is sufficient. Any other form of binary description will work just as well.

Input signals are represented in terms of a current value and a future value. Signal values used in a 5 valued simulator along with their current value-future value representation are shown in Fig. 1. This representation is similar to the data structure used in many 5 valued simulators[5] and is easily adaptable to parallel simulation.

Symbol	Signal Description	Current Value	Future Value
1	Logical 1	1	1
0	Logical 0	0	0
U	Upward Transition	0	1
D	Downward Transition	1	0
E	Unknown Signal	{ 0 1 }	{ 0 1 }

Fig. 1 Signal Representation

The representations for the first four values are obvious. The unknown signal is provided with a dual representation where it could have a current and future value of either 0 or 1. A static condition for the unknown signal has been assumed which is true in most circumstances. More elaborate representation of unknown signals is also possible and will be discussed in a later section.

3. Formulation of Input Vectors

For a logic circuit with n-inputs, the input condition at any instant of time can be represented by the vector

$$\vec{x} = \langle x_1, x_2, \cdots x_n \rangle$$

where the input signals $x_i \in \{0,1,U,D,E\}$. For an input vector, where the inputs $x_i \in \{0,1\}$, the output can be determined from the binary truth table or from any binary convention used to represent the circuit. If this output is y, then $y \in \{0,1\}$.

Using the current value-future value representation for the input signals, the following pseudo input vectors can be generated:

1) The current input vector:

$$\vec{x}_c = \langle x_{1c}, x_{2c}, \cdots x_{nc} \rangle$$

where x_{ic} is the current value of the input signal x_i.

2) The future input vector:

$$\vec{x}_F = \langle x_{1F}, x_{2F}, \cdots x_{nF} \rangle$$

where x_{iF} is the future value of the input signal x_i.

3) Intermediate input vectors:

$$\vec{x}_{Ij} = \{\langle x_{1k}, x_{2k}, \cdots x_{nk} \rangle | x_{ik} \in \{x_{ic}, x_{iF}\}\}$$

where the vectors are generated by taking all possible different combinations of current values and future values of the inputs. Thus a possible intermediate input vector could be

$$\vec{x}_{I(Ex.)} = \langle x_{1c}, x_{2F}, x_{3c}, \cdots x_{nF} \rangle$$

Two intermediate vectors generated in this manner would be the current and future input vectors. These are ignored to avoid redundancy.

When $x_{ic} = x_{iF}$, which is the case when there is no transition at the corresponding input, the two sets of intermediate vectors generated by x_{ic} and x_{iF} are identical and hence only one is retained. The number of unique intermediate vectors depends on the number of changing inputs and if m inputs change, the number of unique intermediate vectors equal (2^m-2).

If any input is an unknown signal, the process is performed twice and two sets of pseudo input vectors are obtained. The first set is obtained by using a current and future value of 0 for the corresponding input and the second set using a current and future value of 1.

The process of pseudo vector generation has been isolated in the beginning to facilitate the following discussion. In the implementation of the hazard detection procedure, the intermediate input vectors are generated as a part of the hazard detection algorithm. Since the algorithm terminates following the detection of a hazard, this assures that no intermediate vector is generated that is not needed.

4. Element Evaluation Procedure

The current, future and intermediate input vectors generated in this manner comprise only of binary values for the inputs and can be used with the binary description of the circuit to obtain a pseudo output which is also binary. This operation is denoted using the following notation.

$$\text{Output} = f\ (\text{Input})$$

where f denotes the binary output evaluation procedure.

The following set of output values are generated using the pseudo input vectors:

1) The current output: $y_c = f(\vec{x}_c)$

2) The future output: $y_f = f(\vec{x}_F)$

When none of the signals are unknown and when no hazard analysis is required, the current and future values of the output y can be interpreted according to Fig. 1 such that $y \in \{0,1,U,D\}$. If however one of the inputs is unknown, 2 different sets of pseudo input vectors are generated. The input vectors are denoted by $\vec{x}_0$ and $\vec{x}_1$ corresponding to the unknown signal being treated as a 0 or 1. For both of these input vectors, the two sets of pseudo input vectors generated yield two different values for the output, y_0 and y_1 corresponding to $\vec{x}_0$ and $\vec{x}_1$, where $y_0, y_1 \in \{0,1,U,D\}$. The resultant value of the output y, is now obtained using the following procedure.

Step 1: If $y_0 = y_1$, then $y = y_0 = y_1$

Step 2: If $y_0 = \bar{y}_1$, then $y = E$

Step 3: If $y_0 \neq y_1$ and $y_0 \neq \bar{y}_1$, then $y = G(y_0,y_1)$

The function G denotes the greater of the two signal values according to the following fule.

$$U = D > 1 = 0$$

If the steps are executed sequentially, y_0 and y_1 cannot satisfy the equality sign in the third step because that condition would have been handled by steps 1 and 2. Thus in the third step, one of the signals is in transition while the other is a stable signal and the resultant output is the transition signal.

Steps 1 and 3 can be replaced by a one step look up procedure by using the table shown in Fig. 2. The values y_0 and y_1 form the two parameters that are used to access the table and the table content specifies the output. The values of y_0 and y_1 in the table include the unknown signal E. There are required for situations where more than one input is unknown.

y_1 \ y_0	0	1	U	D	E
0	0	E	U	D	E
1	E	1	U	D	E
U	U	U	U	E	E
D	D	D	E	D	E
E	E	E	E	E	E

Fig. 2 Recombination of Output Signals

When more than one input signal is unknown, the process of evaluating the output is divided into several identical output evaluation procedures. Outputs are obtained for all conditions of the unknown signals (current and future values of 0 and 1). These outputs are combined using the three rules specified earlier and the following one:

Step 4: If y_0 or y_1 equal E, then $y = E$.

This step should precede all the others since once an unknown output signal is obtained, the process of output recombination becomes irrelevant and the resultant output will be E. In general, the output recombination can be done using the table in Fig. 2 to yield the final output.

Fig. 3 shows an example of this process when two inputs are unknown. A logic circuit with n-inputs is assumed where the inputs x_1 and x_2 are unknown. Each of the outputs y_{00}, y_{10}, y_{01}, y_{11} are evaluated using the procedure discussed earlier where none of the inputs were unknown. These are combined using Fig. 2. If at any stage, a signal value of E is obtained, the remaining steps can be skipped and the resultant value of y can be set equal to E. As shown in Fig. 3, the procedure for output evaluation, where one or more input signals are unknown, is recursive and can be easily implemented in simulation. The number of output evaluations required for each unknown input grows exponentially as a power of 2. If there are m unknown inputs, then the output has to be evaluated 2^m times.

The rapid growth of the number of output evaluations with unknown inputs makes the entire procedure inefficient. However, in most applications, the situation where several inputs are unknown is an unrealistic situation created due to limitations in the simulation procedure. The user is generally not interested in knowing the circuit behavior when several of its inputs are unknown and these situations can be detected and side-stepped to prevent the efficiency from deteriorating.

Furthermore, the scope of the unknown signal varies with the number of simulation levels used. For 3 valued simulation, all signal conditions other than a stable 0 or 1 is classified as unknown. This puts all transitions

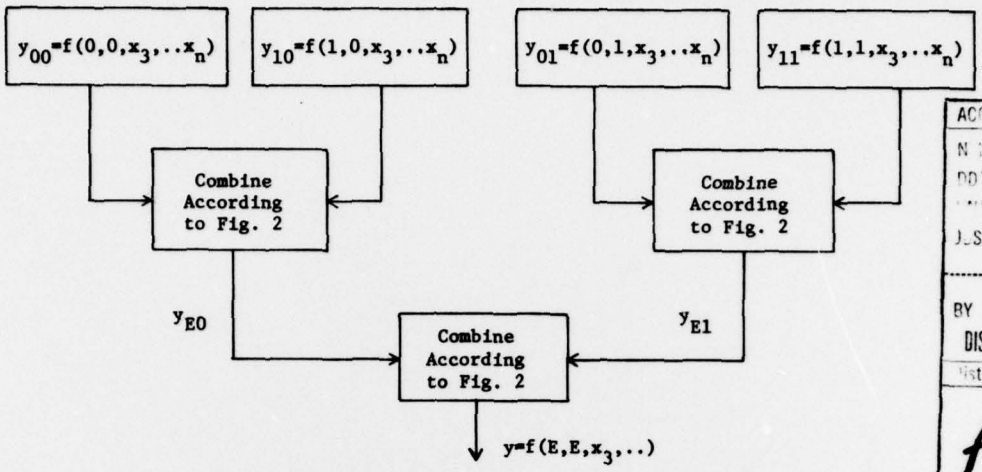


Fig. 3 Output Evaluation for 2 Unknown Inputs

and hazards also in the unknown category. For 5 valued simulation, transitions are known conditions so the scope of unknown signals is diminished. For 9 valued simulation, hazards also become known conditions and unknown signals are further diminished. Since one of the basic results of this approach is higher level simulation (5 level simulation has been discussed and 9 level simulation is to be discussed in a later section), the scope of the unknown signal in simulation, where this approach is used, will be very nominal.

Many simulators suffer from the disadvantages that arise from interpreting the complement of an unknown signal as an unknown signal. In the following logic circuit, if the output:

$$y = c \cdot x_1 + \bar{c} \cdot x_2,$$

and if both x_1 and x_2 are 1 and c = E, the simulator would predict

$$y = E \cdot 1 + \bar{E} \cdot 1 = E + E = E.$$

In reality however, if E is either a 0 or a 1, the output is a 1. By the procedure discussed in this section, y is evaluated once for c = 0 and again for c = 1 and then combined according to the table in Fig. 2. Carrying out the procedure, the following results are obtained

$$y_0 = 0 \cdot 1 + 1 \cdot 1 = 1$$

$$y_1 = 1 \cdot 1 + 0 \cdot 1 = 1$$

Combining using Fig. 2, y = 1.

The more realistic handling of unknown signals justifies the use of this procedure in situations where one or two input signals are unknown. The situation where many input signals are unknown can still be dealt with at the user's option but the situation would probably be too artificial to warrant the expense.

5. Hazard Detection Procedure

Hazard detection is carried out by using the intermediate input vectors. The current and the future input vectors yield the current and future values for the output y_c and y_F. If $y_c = y_F$, the possibility of static hazards have to be investigated whereas if $y_c \neq y_F$, the output is in transition and possibilities of dynamic hazards exist. The hazard detection procedure is carried out by the following steps.

Step 1: Using the intermediate input vectors, generate intermediate outputs.

$$y_{Ii} = f(\vec{x}_{Ii})$$

Step 2: If $Y_c = y_F$ and $Y_{Ii} \neq y_c = y_F$, then a static hazard exists.

Step 3: If $y_c \neq y_F$, then if all y_{Ii} are identical, no dynamic hazard exists.
If $y_{Ij} \neq y_{Ik}$ for any j and k, then a dynamic hazard is present.

The intermediate input vectors are generated one at a time and one of the two tests is performed depending on the values of y_c and y_F. The procedure can be terminated as soon as a hazard is detected but has to exhaust all intermediate vectors to establish the absence of a hazard.

The number of intermediate vectors that can be generated when m inputs change is (2^m-2). Thus where m=1, the number of intermediate vectors is zero and no hazards are predicted.

An algorithm which performs the hazard detection is shown in Fig. 4.

If one of the inputs is an unknown signal, the hazard detection process has to be repeated for the two possible values of that signal.

The hazard detection procedure exhaustively tests the logic circuit for all possible combination of input conditions that can exist due to transitions at the input. It is intuitively obvious that if the current and future values of the output are the same while there is the possibility of a situation at the input, during the transitions, that makes the output different from this steady value, then the possibility of a static hazard exists. Similarly if the output has different current and future values, it is in transition. If all the intermediate values that can arise during the transition are the same as the current value or the future value, then the transition is hazard free. If however, during the transition, two different intermediate output

values are different, the output may go through both of them. Each intermediate output is an output condition that can occur due to the ambiguity in the transitions at the inputs. Thus if the output goes through both of the different intermediate outputs, a dynamic hazard exists.

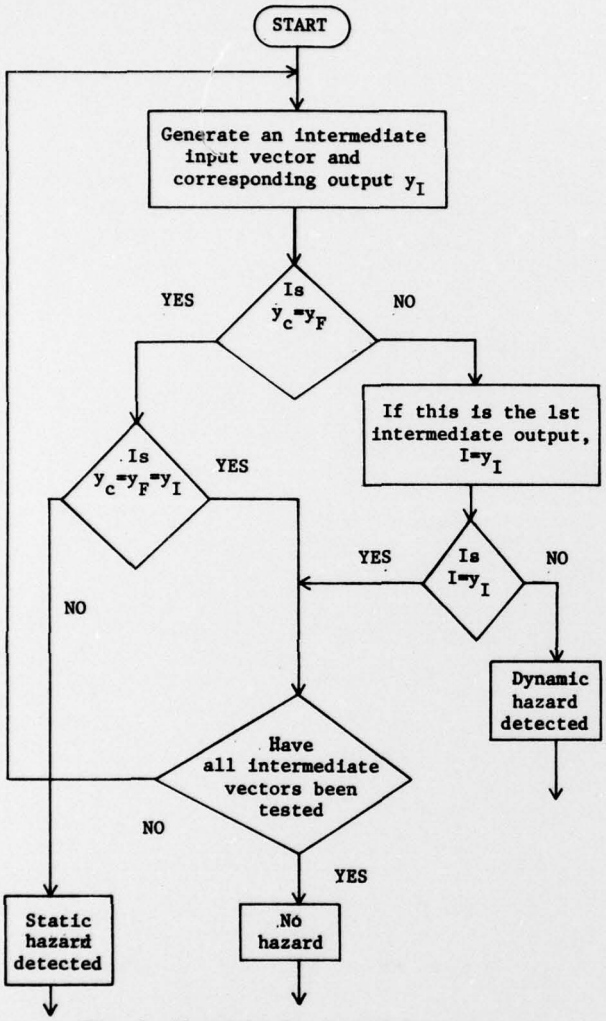


Fig. 4 Algorithm for hazard detection

The following example illustrates the hazard detection procedure.

Example 1

Consider a 3 input logic circuit described by the truth table of Fig. 5. The input vector $\vec{x} = \langle x_1, x_2, x_3 \rangle$

Case 1 $\vec{x} = \langle U,U,1 \rangle$

The current input vector is

$$\vec{x}_c = \langle 0,0,1 \rangle$$

and the future input vector is

$$\vec{x}_F = \langle 1,1,1 \rangle$$

The corresponding outputs are

$y_c = 0$ and $y_F = 0$, thus the output is 0.

inputs			output
x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Fig. 5 Truth Table for 3 input logic circuit

The intermediate input vectors are

$$\vec{x}_{I1} = \langle 0,1,1 \rangle$$

and

$$\vec{x}_{I2} = \langle 1,0,1 \rangle$$

The corresponding outputs are

$$y_{I1} = 1 \text{ and } y_{I2} = 1$$

Since $y_c = y_F \neq y_{I1}$ or y_{I2}, a static hazard exists. y_{I2} need not have been evaluated since y_{I1} alone indicates the static hazard.

Case 2 $\vec{x} = \langle U,U,0 \rangle$

thus $\vec{x}_c = \langle 0,0,0 \rangle$

$\vec{x}_F = \langle 1,1,0 \rangle$

and $y_c = 0$ and $y_F = 1$. The output is 'U'.

The intermediate vectors are

$$\vec{x}_{I1} = \langle 0,1,0 \rangle$$

$$\vec{x}_{I2} = \langle 1,0,0 \rangle$$

The corresponding outputs are

$$y_{I1} = 0 \text{ and } y_{I2} = 0$$

Since $y_{I1} = y_{I2}$, the upward transition is hazard free.

Case 3 $\vec{x} = \langle U,D,U \rangle$

thus $\vec{x}_c = \langle 0,1,0 \rangle$

$\vec{x}_F = \langle 1,0,1 \rangle$

and $y_c = 0$ and $y_F = 1$. The output is 'U'.

The intermediate vectors and the corresponding outputs are:

$$\vec{x}_{I1} = \langle 0,0,1 \rangle \qquad y_{I1} = 0$$

$$\vec{x}_{I2} = \langle 1,1,0 \rangle \qquad y_{I2} = 1$$

Since $y_{I1} \neq y_{I2}$, a dynamic hazard exists. Though more intermediate vectors exist, they need not be generated.

The way these detected hazards are to be handled depends on the environment in which this procedure is used. They may either be ignored or propagated as an unknown signal or propagated as a hazard itself. The last option leads to nine valued simulation.

6. Nine-valued Simulation

The commonly used signal levels in nine valued simulation are shown in Fig. 6.

Symbol	Signal Description	Current Value	Intermediate Values	Future Value
1	Logical 1	1	1 1	1
0	Logical 0	0	0 0	0
U	Hazard Free Upward Transition	0	0 OR 1 0 OR 1	1
D	Hazard Free Downward Transition	1	0 OR 1 0 OR 1	0
1*	Static 1 hazard	1	at least one 0	1
0*	Static 0 hazard	0	at least one 1	0
U*	Dynamic hazard	0	1 OR 0 0 OR 1	1
D*	Dynamic hazard	1	1 OR 0 0 OR 1	0
E	Unknown signal	{ 0 1	0 0 1 1	0 1 }

Fig. 6 Signal levels in nine-valued simulation

The simulation and hazard detection procedures discussed earlier can provide 9 valued outputs for 5 valued inputs into logic circuits. It remains to be shown that 9 valued inputs into the circuit can be handled and corresponding 9 valued outputs can be obtained by minor extension of this approach. This would indicate the applicability of this approach to 9 valued simulation.

To handle nine valued signals at the input, two intermediate input values are introduced in addition to the current and future input values. The representation of the nine valued signals with current, future and intermediate values is shown in Fig. 6. Where several choices have been indicated any one will do.

The simulation procedure used in the nine-valued case is similar to that used for the five valued case with the exception of the generation of intermediate input vectors. These vectors are now generated not only for all combinations of current and future values of the inputs but also for the combinations of intermediate values. Thus

$$\vec{x}_{Ik} = \langle x_{1j}, x_{2j}, \cdots x_{nj} \rangle$$

where $$x_{ij} \in \{x_{ic}, x_{iF}, x_{iI1}, x_{iI2}\}$$

If there are m inputs with hazards present, the maximum number of intermediate vectors would be 4^m-2. The number of intermediate input vectors for 1 input with a hazard present is 2. For 2 inputs with hazards, the number is 14. The number of intermediate input vectors grows very rapidly but these figures are only the worst case figures. The intermediate input vector generation process can be terminated following the detection of a hazard. Moreover, the situation where a circuit sees hazards at several inputs simultaneously is unrealistic. These situations can again be detected and side-stepped to keep the simulation efficient. In general however, the user has the option to realize nine level simulation to any degree he requires.

7. Conclusion

There are several attractive features of the approach discussed in this paper. Multi-level simulation of arbitrary devices are possible by requiring the user to provide only a two level description of his net. Neither the user nor the creator of the simulators are required to be concerned about the great many situations that could exist at the input of these nets.

Extensive flexibility can be obtained with very little effort. The unknown signal has been assumed to have a dual representation of either a 0 or a 1 in the preceding discussion. Dynamic conditions can be handled by assuming E to have a dual representation of a U and a D. Hazard conditions can be handled similarly and in general E could comprise any or all of these conditions. The outputs obtained in all these cases can be combined using the table in Fig. 2.

The hazard detection procedure involves a nominal amount of computation when the number of input signals in transition are few. When this number becomes large, the amount of computations grows very rapidly. An improvement to this situation is possible, in certain cases, especially when a binary truth table is used to specify the operation of the logic net. Instead of generating intermediate vectors and inspecting whether the corresponding outputs produce a hazard, the procedure could be reversed. The output which would indicate a hazard can be determined and the truth table look up operation reversed to obtain the corresponding inputs. These could be inspected to see if any of them correspond to a possible intermediate input vector. This process can be realized by splitting up the truth table into two parts, each corresponding to an output of 0 or 1, and would be more efficient if one part is smaller than the other.

The above technique and the others discussed earlier improve the efficiency of this approach. Worst case situations however could still exist which require extensive computation. In situations where this approach does not prove to be a viable technique, it can still be used as a tool to aid in the verification of the alternative approach that is being used.

References

1. Breuer, M.A. and A.D. Friedman, "Diagnosis and Reliable Design of Digital Systems," Computer Science Press, Inc., 1976.

2. Breuer, M.A. and L. Harrison, "Procedures for Eliminating Static and Dynamic Hazards in Test Generation," IEEE Trans. on Computers, Oct. 1974.

3. Fantauzzi, G., "An Algebraic Model for the Analysis of Logic Circuits," IEEE Trans. on Computers, June 1974.

4. Breuer, M.A., "A Note on Three Valued Logic Simulation," IEEE Trans. on Computers, April 1972.

5. Szygenda, S.A. and E.W. Thompson, "Digital Logic Simulation in a Time-Based Table-Driven Environment," Computer, March 1975.

This research was supported in part by the DoD Joint Services Electronics Program through the Air Force Office of Scientific Research (AFSC) Contract F44620-76-C-0089.