among internal signals on the activity of an asynchronous sequential circuit.

III. Example

The origin and purpose of (1)–(5) will be illustrated in Table I, which should be considered as a portion of a larger flow table.

The transition from state 1 to state 2 involves the race between \( Y_1 \) and \( Y_2 \), which may be critical (see Table I). Let us assume that the change \( Y_1 = 0 \rightarrow 1 \) will follow the change \( x = 0 \rightarrow 1 \) with minimum possible delay \( \Delta_m = d_m + D_m + D_m^2 \). In this (worst) case the total state \((x, y_1, y_2) = (1, 0, 0)\) will not change during the time interval \( \Delta_m \). This time interval is sufficient to secure the change \( y_2 = 0 \rightarrow 1 \) if

\[
d_M + D_M < \Delta_m + d_m
\]

and the change \( z = 0 \rightarrow 1 \) if

\[
d_M + D_M^2 < \Delta_m + d_m
\]

Consider now that the change \( y_2 = 0 \rightarrow 1 \) will follow the input change with maximum possible delay \( \Delta_M = d_M + D_M^2 + D_M^3 \). In this (worst) case the values of signals \( Y_1, Y_2, \) and \( Z \) may be influenced by the race between \( y_1 \) and \( y_2 \) during the time interval \( \Delta_M + d_M - \Delta_m - d_m \). Consequently, the error pulses produced by this race, as well as error pulses produced by zero-order hazards, will not be transmitted through the inertial delay elements \( d_1, d_2, \) and \( d_3 \) if

\[
\Delta_M + d_M - \Delta_m - d_m < D_M^2
\]

\[
\Delta_M + d_M - \Delta_m - d_m < D_M^3
\]

IV. Conclusions

The influence of \( m \)-order hazards and races on the activity of the asynchronous sequential circuit can be eliminated by using inertial delay elements in case: 1) the circuit operates in fundamental mode; 2) the circuit is arranged in accordance with Fig. 1; and 3) the inertial delay elements that satisfy the inequality (6) are physically realizable.

In comparison with conventional methods of hazard correction the following features of the use of inertial delay elements may be stressed:

1) simplification of the circuit synthesis and saving of gates (neither hazard-free combinational logic, nor critical race-free state assignment are indispensable); 2) increase of the circuit resistance to interfering signals; and 3) drop in the circuit speed (inertial delay elements increase the response time to any change of input state).

REFERENCES


A Note on a Modified Ternary Simulator Capable of Initializing All Fault Machine Memory Elements

MICHAEL E. GRAVITZ

Abstract—Generation of test patterns and diagnostics for complex digital modules often involves a ternary \((0, 1, u)\) simulation program. A non-Boolean technique, presented herein, initializes \( u \) (unknown) values in such a way that diagnostic resolution and accuracy is enhanced. A pure ternary simulator lacks this initialization capability.

Index Terms—Fault detection, fault machine, fault signature, good machine, initialization, input test vector, ternary simulator.

I. INTRODUCTION

Ternary logic simulator programs are finding widespread application as tools aiding in the production of test vectors for digital circuit diagnosis [1]–[3]. Resultant test vectors are used in a computer-driven digital tester that verifies or fault isolates modules consisting of from 30–120 IC packages. All diagnostics are performed by the tester and its output is a tabulation of those IC’s that may be faulty. When ternary simulator-generated test vectors are used in this fashion, it is imperative that simulator accuracy be maximized. This correspondence discusses a method of eliminating fault detections that are based on \( u \)-logic values ("possible" faults). This is accomplished by utilizing a set of rules that assign 1-0 values to the \( u \)’s when certain information about the circuit becomes determinate. Practical experience has shown that a simulator equipped with this feature can eliminate repeated "possible" detections of the same fault, thereby reducing storage requirements in the digital tester and improving overall diagnostic resolution.

II. EXAMPLES OF INITIALIZE \( u \)’S

When power is initially applied to a circuit, memory elements assume random unknown states. The simulator assigns to these outputs the "unknown" logic value \( u \). Since a ternary simulator uses the logic value \( u \) to represent unknown values (although still \( 0 \) or \( 1 \)), \( u \)’s appearing at the output of memory elements propagate from there to other circuit nodes. The \( u \)'s will be removed from the outputs only if the memory elements are initialized by appropriate manipulation of their inputs. As a specific example, consider the \( D \)-type edge-triggered flip-flop of Fig. 1.

Prior to application of the first test vector, the simulator will assign both the \( Q \) and \( \bar{Q} \) outputs a logic value \( u \). In order to eliminate the \( u \)'s at the outputs of the flip-flop, a vector must be applied to the inputs of the driving logic circuitry such that a known value is applied to \( D \) and an appropriate clock pulse to the clock line. When this can be done, firm 1-0 logic values can be assigned to the flip-flop outputs.

Manuscript received August 14, 1972; revised January 10, 1973.

The author was with the McDonnell Aircraft Company, St. Louis, Mo. He is now with the Department of Electrical Engineering, University of Missouri, Columbia, Mo. 65201.
Values of $u$ at the outputs of all memory elements can be eliminated unless the only means of initialization (of a particular fault machine) involves a path through the faulted node. Now consider Fig. 2, a D-type flip-flop with its clock line stuck-at-zero (s-a-0).

For the example of Fig. 2 there exist no test vectors that can be applied to initialize the flip-flop. Its outputs will remain $u$'s. During test-vector generation, vectors will be applied that will exercise the clock line of the s-a-0 fault machine. If, for a particular test vector, the good machine output at $Q$ were to be a 1 and the fault machine were to pass this test, it would follow that the $u$ on the fault machine at $Q$ must also be a 1 and should be so assigned. Pure ternary simulators are unable to make this assignment. On a later test-vector application to the same fault machine, if the test-pass response at $Q$ were 0, clearly the fault would be detected since the flip-flop output would remain a 1 (assuming that flip-flop outputs propagate to primary outputs).

Because the simulator could not make the correct 1-0 assignment to $u$, it would be unable to firmly detect the fault. Firm-fault detection is dependent upon 1-0 discrepancies. As a result the fault will remain in the list of fault machines to be simulated and will be called out as a "possible" detection (detection dependent on the correct value of $u$'s in the fault machine) for the remainder of input test vectors that may "possibly" detect it. "Possible" fault detection increases simulator run time, decreases diagnostic resolution, and increases the size of the fault dictionary.

III. SIMULATOR CONSTRAINTS AND MODIFICATIONS TO ELIMINATE $u$'S

Item 1: The good machine must be completely initialized before application of the technique.

Item 2: Every memory element must have a subscripted $u_j$ associated with its output until input vectors or initialization information drive it to a known state.

Item 3: A circuit-dependent array must be developed that associates primary outputs with those $u_j$ that propagate to them.

Item 4: For each input test vector, each fault machine output must be compared with the good machine output. A "possible" diagnostic would be generated when the fault machine output signature has at least one $u_j$ that is not present in the good machine output. Set a flag to indicate such a condition.

Item 5: When a $u_j$ discrepancy between the signatures of the fault machine and the good machine is indicated (by Item 4), all combinations of the discrepant $u_j$ must be simulated and each 1-0 combination assigned to the $u_j$ is treated as a separate fault machine.

Item 1 is required to ensure that the technique does not make a 1-0 assignment to discrepant $u_j$ that exist in fault machines but are not attributable to the modeled fault. Item 2 is a bookkeeping operation, and provides a means of determining if the fault machine and the no fault machine have different $u$ outputs. Prior to any fault simulation, the Item 3 array must be generated. Note that the array does not require logical $u$-manipulation (i.e., use of $u$ and $\bar{u}$). The array could be obtained by a trace program working from the wire list describing the circuit to be simulated. As an example, consider Fig. 3, which shows the $u_j$ associated with memory elements and the output pins to which they Propagate.

When a pure ternary simulator encounters a fault machine having a fault signature with $u$'s at output pins other than those that occur in the good machine, depending upon the values of those $u$'s, it is possible to detect a circuit fault. Fault detection is dependent on the $u$'s taking logic values which drive the fault machine outputs to values other than those produced at outputs of the good machine. This dependency is what the technique seeks to eliminate. If a $u_j$ discrepancy should occur, as defined by Item 4, a flag is set so that the subroutine defined by Item 5 is executed.

Item 5 is the interface between the technique and ternary simulator. $u_j$ discrepancy results in the simulation, for all possible 1-0 combinations, of all $u$'s that are different from those at the good machine's output. For each 1-0 combination a separate fault machine is initiated that possesses the internal variable values dictated by the 1-0 combination associated with it. These fault machines now replace the fault machine which originally produced the discrepant $u_j$. The number of additional fault machines is $2^N$ where $N$ is the number of discrepant $u_j$. Note that all of the newly initiated fault machines possess the same faulted node as the fault machine they replace. However, each new fault machine differs by internal variable assignment as dictated by the 1-0 values assigned to the $u_j$. When the newly initiated fault machines have been established, the $u_j$ that produced their internal variable assignments are deleted from the array of Item 3. By utilizing this technique, repeated "possible" diagnostics can be completely eliminated while still maintaining complete fault dictionary coverage.

A test vector affecting one of the fault machines which replaced the fault machine that produced the discrepant $u_j$ will be firmly detected, as 1-0 discrepancies will exist between the affected fault machine and the fault-free machine.

IV. EXAMPLES

Several examples will now be given to illustrate operation of the modified simulator. Consider first a flip-flop clock-line fault of s-a-0 (Fig. 2). Assume that the driven logic circuitry propagates all changes in the output of the flip-flops.

Initially, these outputs will be assigned a value of $u$. Assume, that for the good machine, the $Q$ output is a 1 after some test vector. In the faulted machine, the $Q$ output remains at $u$. From Item 4, a discrepancy between fault and good machine is seen to exist. Item 5 specifies that the $u$ must now be made a 0 and a 1 in two separate fault machines. If a test vector is now applied, setting the good machine to a 0 (at $Q$), the faulted machine would remain at a 1 thereby detecting the
faulted clock line s-a-0. Of course, the fault machine with Q assigned 0 will be detected at the vector where the u discrepancy became apparent.

Fig. 4 shows a three-input NOR gate with a s-a-0 on one of its inputs and u's on the other two provided by two uninitialized flip-flops. For this circuit consider the good machine to have a 1 at the node where the s-a-0 fault resides. The good machine output will be 0, while the fault machine output will be u1, u2. This meets the discrepancy requirements of Item 4 but not the requirements of Item 1. The good machine is not initialized. For this circuit, at its present state, the technique will take no action. This example shows the need for complete good machine initialization.

As a final example, consider Fig. 5. Many faults exist in this circuit but only a few shall be considered, those marked (1), (2), (3), (4), and (5). Observe that the u's have already been driven to the outputs and that no logical operations were required for this process. It is reemphasized that this is a non-Boolean procedure. Let the first input test vector applied be

\[ T = J_1 = J_2 = J_3 = J_4 = 1 \]
\[ K_1 = K_2 = K_3 = K_4 = 0 \]

with the clock = 010.

The response of the good machine is Q1 = Q2 = Q3 = Q4 = 1, Z1 = Z2 = Z3 = 0, and Z4 = 1. For the first fault machine, the output of flip-flop 2 will remain at u2; hence the output of the fault machine will be Z1 = Z2 = u2, Z3 = 0, and Z4 = 1. Comparing the fault machine and good machine output signatures, it is evident that a u discrepancy, as defined in Item 4, exists. Item 5 requires assignment of values to u2 such that two fault machines are generated, one with u2 = 1 and the other with u2 = 0. The u2 = 0 fault machine is detected immediately while the u2 = 1 fault machine requires further vector application. For this fault machine, if another input test vector (and clock) is applied with J3 = 0, T = K2 = 1, J1 = K1 = J3 = K3 = J4 = K4 = 0, it would be detected.

Fault machine (2), after application of the first input test vector, will have an output signature of Z1 = 0, Z2 = 0, Z3 = 0, Z4 = 1. This signature is identical to the good machine signature. Since the u's were initialized by the input pattern, no action needs to be taken. Note that the discrepancy criterion of Item 4 is not met.

Fault machine (3), after application of the first input test vector, will have an output signature of Z1 = 0, Z2 = 1, Z3 = 1, and Z4 = 1. Comparing this signature with the good machine output, it is seen that there is a "firm" detection attributable to 1-0 differences in the outputs Z2 and Z3. Here again, the technique is not applicable as the discrepancy criterion of Item 4 is not met.

Fault machine (4), after application of the first input test vector will have an output signature of Z1 = u1, u2; Z2 = u3, u4; Z3 = u3, u4, and Z4 = u1, u4. A u discrepancy, as defined by Item 4, exists. As there are four different u1 at the outputs, there are 16 combinations of 1-0 assignments that can be made to the u1. It turns out, for this circuit that only the u1 combination corresponding to the good machine memory state output will provide good machine output. Hence 15 of the 16 u1 combinations are detected with no further test-vector application. The fault machine with the good machine memory state of u1 at Q1 = u2 = u3 = u4 = 1 can be detected with the application of a second test vector (and clock) with J1 = J3 = J4 = 0 and K1 = K2 = K3 = K4 = T = 1. This is an unusual situation in that only one of many u1 combinations produced good machine output. In general, this will not be the case. Usually several 1-0 combinations assigned to multiple u1 will produce good machine output. This is significant because simulator run time can be increased if it is necessary to replace single fault machines with many as will be the case when there are multiple discrepant u1. Control of this activity is sometimes necessary to ensure cost effective simulator operation.

For the final fault, (5), consider a different first input test vector. Let the first input be J1 = J2 = J3 = J4 = K1 = K2 = K3 = K4 = 0 and T = 1. In the good machine the output signature would be Z1 = u1, u2, Z2 = u2, u3, Z3 = u3, u4, and Z4 = u1, u4. In the fault machine the output would be Z1 = u1 u2, Z2 = u3, Z3 = u3 u4, and Z4 = 1. In this case, it is the good machine that has u's where the fault machine has a firm value. The technique is again not applicable as the criterion of Item 1 and Item 4 is not met. Although this fault could easily be detected by application of the appropriate input test vector, for this fault and two of the previous faults, the only instance where the u-initialization technique is applicable is when a very specific form of discrepancy occurs—when a "possible" diagnostic can take place.

V. Conclusions

A procedure has been presented which, when applied to a ternary simulator, eliminates "possible" diagnostic callouts for a fault machine while still allowing the simulator to perform correct diagnosis for the same fault machine on a subsequent input test vector. Means other than Boolean manipulation of the u value were utilized in implementing the technique. This circumvents the very real simulator array limitations described by Breuer [4] while still allowing efficient simulator operation.

References