A New Two Task Algorithm for Clock Mode Fault Simulation in Sequential Circuits

Fredrick J. Hill, Eltayeb Abuelyamen, Huang Wei-Kang and Shen Guo-Qiang

Department of Electrical and Computer Engineering
The University of Arizona

ABSTRACT: A new approach to clock mode simulation of sequential circuits is introduced. Surrogate fault propagation is used for processing stored faults and extracting new faults from combinational logic. Problem fault types are analyzed and treated as exceptions.

I. INTRODUCTION

In the recent past interest in simulation of test sequences has been limited to simulating test vectors for combinational logic networks [1,2,4,7]. Justification for this approach has been primarily the availability of level sensitive scan design techniques (LSSD) which provides direct access to the inputs and outputs of all combinational logic networks on a VLSI chip. While the LSSD approach eases the task of test generation it does not promise minimal length test sequences. Efforts to minimize both testing time and logic overhead require test generation and fault simulation of sequential circuits.

This paper develops a completely new approach to clock mode fault simulation for sequential circuits. Those problems which make fault simulation of sequential circuits more complicated than repetitive simulation of combinational logic are identified and treated as special cases. For this reason we identify our approach by the acronym SFSSE (Synchronous Fault Simulation by Surrogates with Exceptions). We shall demonstrate that SFSSE will enjoy a speed advantage over parallel simulation similar to that enjoyed by Hong's method [1,4] over parallel simulation of combinational logic. It will in fact implement a variation of this method for simulation of faults not already stored in memory elements together with separate techniques responsive to stored faults. When fully implemented, SFSSE will be integrated into our existing sequential circuit test generation program SCIRTSS [8]. In the interim the accurate SCIRTSS parallel simulation will serve as a check and speed benchmark for the new approach. Like the SCIRTSS parallel version, SFSSE will enjoy an obvious speed advantage over existing gate level simulators.

II Options

The candidate techniques for a clock mode fault simulator include all of the simulators currently used for simulating combinational logic, a subset of which have already been used in various forms of sequential circuit fault simulation. These are listed as follows: (1) Parallel fault [5], n-1 faulty networks are represented by bits of the host computer word and simulated simultaneously; (2) Deductive [9], the list of faulty networks differing from the good network is determined as each gate is simulated; (3) SFP (single fault propagation) [3], each faulty network is simulated independently, terminating if the effect of that fault disappears or reaches a primary output; (4) Concurrent [7], where gates representing faulty networks are simulated if inputs differ from the good network values; (5) Inference from surrogate fault simulation (to be described) [1, 4]; (6) Critical Path Tracing [2], an approximate method similar to but faster than surrogate fault simulation for combinational logic; (7) Parallel pattern [6], similar to single fault propagation except that the n bits of the computer word are used to represent n test patterns processed simultaneously.

Sequential circuit fault simulation must be completely accurate. Useable as a tool for verifying correctness of test sequences determined by other methods. This condition disqualifies critical path tracing which is approximate even for combinational logic. Method 7 is not applicable to sequential circuits where input patterns must be applied in sequence. Deductive and concurrent simulation and SFP enjoy an advantage over parallel simulation in that the former require progressively less execution time per clock period as the simulation progresses and a larger portion of the faults are detected. Ozguner [3], for example, demonstrated deductive

CH2540-3/88/0000/0583$01.00 © 1988 IEEE

and SFP to be substantially faster than parallel for simulation of complete sets of test vectors on a set of combinational logic networks including between 200 and

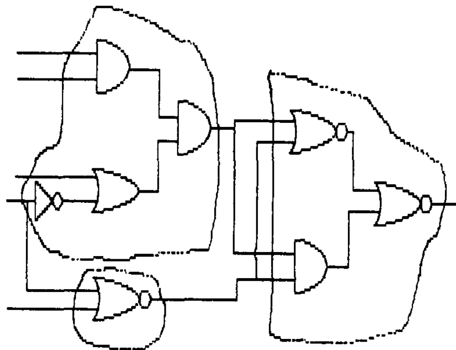

Figure 1 Three FFR'S of a Subnetwork

2000 gates. Surrogate fault propagation has recently been demonstrated by Nishida et. al. [4] to be faster than concurrent simulation of combinational logic.

Making a distinction between primary and secondary outputs and allowing faults to be stored expands the footprint of single faults in the SFP method and lengthens the list of faults which must be processed at each node by a deductive simulation. Both of these methods will be slowed when applied to sequential circuits. This paper will extend Surrogate fault propagation (Hong's method) to sequential circuits.

After each clock period of simulation of a sequential circuit by SFSSE, a list of all faults stored will be associated with each memory element. A fault is said to be stored in a memory element, if the value of that memory element for the corresponding single fault network differs form the good network value. A fault whose effect appears at a primary or secondary output of the logic network is sensitized. A fault having once been driven to a primary output will be called detected.

Two major tasks must be accomplished as a new test vector is applied for the next clock period of execution.

Task 1: Determine the sensitivity of each FOS (fanout-stem to be defined below).

Task 2. Identify newly detected and stored faults.

a.) Using the FOS sensitivity from Task 1, determine those faults driven by the test vector to primary or secondary outputs from their points of origin in the logic network.

b.) Identify those stored faults (memory element outputs are also FOS's) propagated through the network to primary or secondary outputs.

c.) More careful simulation of exceptional faults

III Propagation and Interpretation of Surrogate Faults

Elimination of all fanout in a combinational logic network except reconvergent fanout requires partitioning of the network into overlapping subnetworks, one for each primary output. Gates are duplicated as necessary to form an independent realization of each output. Once partitioning is accomplished, each reconvergent fanout node is called a fanout stem (FOS).

Behind each FOS y is a fanout free region (FFR) identified by the following algorithm.

```
Include gate y in set S;
WHILE S is not empty
   Move gate j from S to FFR;
   FOR each input i of gate j
      IF the source gate of i is not a PI
         or FOS put source gate in S
      ENDIF
   ENDFOR;
ENDWHILE.
```

Application of this algorithm to a network with three FOS's is shown in Fig. 1.

Definition 1: For a given logic network N and a given input test vector T, if the value of the network output, f, is a function of the value of node y, then y is network sensitive in N.

Network sensitivity relates directly to the notion of a Boolean difference. That is, df/dy evaluated for test vector T is 1 if and only if y is network sensitive in N. The following definition for a fanout free region is consistent.

Definition 2: A node x in an FFR with FOS y is FFR sensitive if and only if

$$dy/dx = 1$$

Fig. 2 depicts one particular FFR behind FOS g of a subnetwork with output f. Because all paths from the arbitrary node h in the FFR pass through the FOS g, the following theorem is an immediate application of the chain rule of Boolean differences.

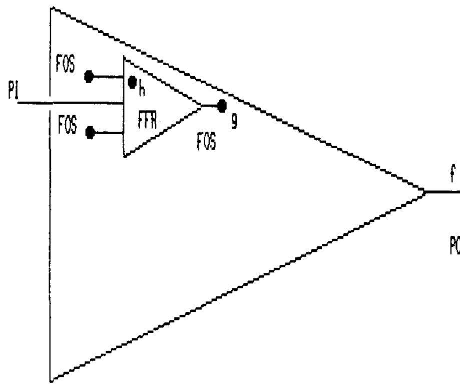

Figure 2 Sensitivity of a Node in an FFR

Theorem 1: If node g is subnetwork sensitive in a subnetwork with output f and node h is FFR sensitive in an FFR with FOS g, then h is subnetwork sensitive.

A test vector which causes a node to be subnetwork sensitive and forces the good network value of that node to be 0 (1) will cause the SA1 (SA0) fault for that node to be detectable at the subnetwork output. This together with Theorem 1 allows definition of SFSSE in terms of the two tasks given in the previous section. S. J. Hong [1] first described the technique for accomplishing task 2a by tracing back from each sensitive FOS through all possible sensitive paths to the boundaries of the FFR. The procedure is straight forward in a fanout free network. Our initial implementation of SFSSE will use parallel simulation to accomplish task 1 for each subnetwork. The step will be omitted, if good network values in a sub network do not change, or if all faults within that subnetwork are already detected.

The subnetwork sensitivity of a particular FOS y may be determined by simply imposing the variable y as the value of that node and computing values of successive gates using y and the test vector input values. The output of any gate may now assume any one of four values 0, 1, y, and $\bar{y}$. These four values may be represented by two bits of a binary vector as given by the table in Fig. 3. It may be seen that the first bit of the vector will always be the good network value. The remaining bit which indicates whether the node is a function of the FOS node will be called an FOS bit. A vector representing the values of all FOS's at a network node will consist of one bit for the good network and one bit for each FOS. Because task 1 does not require imposition of faults at each gate input it is much less time consuming than parallel fault simulation.

As simulation begins all bits in vectors representing gate inputs are assigned good network values. Each gate is simulated in turn by vector execution of the gate operation on the input vectors.

if good network value at the FOS = 0		if good network value at the FOS = 1	
good bit, FOS bit	node value	good bit, FOS bit	node value
[0,0]	0	[0,0]	0
[0,1]	y	[0,1]	$\bar{y}$
[1,0]	$\bar{y}$	[1,0]	y
[1,1]	1	[1,1]	1

Figure 3 Coding of Node Values

After the vector at FOS i is computed, the value of the good network at that node is checked; and the opposite value is assigned to bit i of the vector just computed for the node. Execution of this process for a network with two fanout stems is shown in Fig. 4.

IV. Exceptional Faults

A convenient state of affairs would have the effects of a single fault stored in two memory elements be independent of each other and of any faults in subnetworks to which they were connected. Task 2c would then be unnecesssary. Faults propagated to the output of a subnetwork would be the union of fault lists stored at all memory elements sensitive within that subnetwork. The final fault lists for the clock period would then be then the union of these lists and the lists determined by task 2a. This process would result in an accurate simulation for the large majority of faults but not all faults.

Definition 3: A compound fault is a fault which is stored in a memory element whose output is connected to the subnetwork containing the point of origin of that fault.

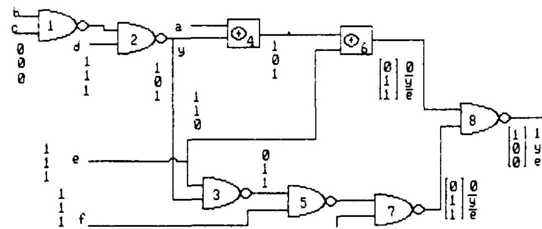

Figure 4 Computation of FOS Sensitivity

Definition 4: A multiply stored fault is a fault stored in two or more memory elements whose outputs are connected to the same subnetwork.

Multiply stored faults can be viewed as extensions of reconvergent fanout paths in the iterative network analogy of the sequential circuit. Equally well both a multiply stored fault and a compound fault resemble coexisting multiple faults in the combinational logic network. Using the notation g/f, where g is the good network value and f the faulty network value, Fig. 5a illustrates a case where a fault multiply stored in two flip-flops will be detected where a fault stored in only one of the two memory elements would not be detected. Fig. 5b illustrates a case in which a fault effect stored only in memory element 5 would be detected while the same fault stored in both memory elements 5 and 10 (values indicated in parentheses) would not be detected.

The situation is similar for a compound fault. Fig. 6a illustrates a case where the storage of the effect of G5I1 SA1 in memory element 10 prevents the detection of the fault at its point of origin. The values for no stored fault (in parentheses) show the original fault to be detected. Fig. 6b shows a case where a compound fault will be detected while the original fault would not be detected in the absence of the stored fault. Because multiply stored and compound faults are uncommon, treating them separately is easily justified in light of the

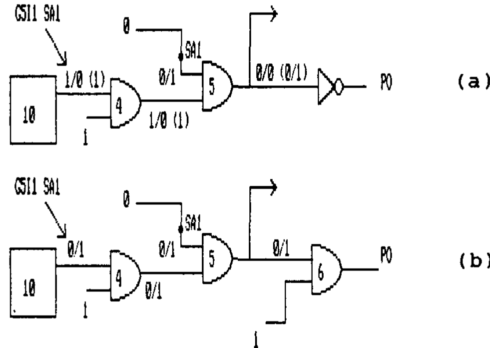

Figure 5 Multiply Stored Faults

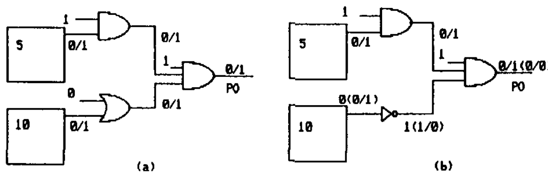

Figure 6 Compound Faults

simplicity and potential efficiency of tasks 1 and 2a and b.

VI. Projections and Conclusions

This paper documents a new approach to fault simulation in sequential circuits. The method is based on propagation of surrogate faults and is independent of any particular implementation. Running time projections have been made and these appear to be justified by preliminary results. Space restrictions preclude detailed justification but the run time projection takes the following form where g is the number of gates in the original network and Rf is the redundancy factor resulting from the partition of the network into overlapping subnetworks.

$$\text{Time} = K1\ (Rf/32)\ g + K2\ g + K3\ g^2 \quad (1)$$

The quadratic term results from merging and searching of fault lists which takes place only once each clock period. K3 is sufficiently small that this term will not be bothersome for networks of many thousands of gates. Usually some such comparison of fault lists is buried in overhead so that results can be printed out each clock period. In contrast the fundamental computation of the conventional parallel simulation varies as the second power of g.

REFERENCES

1. Hong, S.J., "Fault Simulation Strategy for combinational Logic Networks," Proc. 8th Int. Conf. on Fault Tolerant Computing, 1978. pp. 96-99.
2. M. Abramovici, P.R. Menon, and D.T. Miller, "Critical Path Tracing - An Alternative to Simulation," 20th Design Automation Conf. 1983, pp 214-220.
3. Ozgunèr, F., et al., "On Fault Simulation Techniques," J. Des. Auto. and Fault Tolerance Computing, Vol.3, No.2, pp. 83-92.
4. Nishida, T., et al., "RFSIM: Reduced Fault Simulation," IEEE Trans. on Computer Aided Design," May 1987, pp. 392-403.
5. Seshu, S., "On an Improved Diagnosis Program," IEEE Trans. on Electronic Computers," pp. 76-81, Feb. 1965.
6. Waicukauski, J.A. et. al., "Fault Simulation for Structured VLSI," VLSI System Design, Dec. 1985, pp. 20-32.
7. Ulrich, E., "High Speed Concurrent Fault Simulation with Vectors and Scalars." Proc. 17th Design Automation Conf., 1980, pp. 374-380.
8. Mohsenni Behbahani, A., F. J. Hill, and M. Y. Patel, Intelligence Driven Test Sequence Generator for VLSI Design, Microprocessing and Microprogramming, North-Holland, 18 (1986) pp.355-362.
9. Armstrong, D. B., "A Deductive Method for Simulation Faults in Logic Circuits," IEEE Trans. on Computers, Vol. C-21, 1972