Functional Test Generation for Synchronous Sequential Circuits

M. K. Srinivas, James Jacob, and Vishwani D. Agrawal

Abstract-We present a novel, highly efficient functional test generation methodology for synchronous sequential circuits. We generate test vectors for the growth (G) and disappearance (D) faults using a cube description of the finite state machine (FSM). Theoretical results establish that these tests guarantee a complete coverage of stuck faults in combinational and sequential circuits, synthesized through algebraic transformations. The truth table of the combinational logic of the circuit is modeled in the form known as personality matrix (PM) and vectors are obtained using highly efficient cube-based test generation method of programmable logic arrays (PLA). Sequential circuits are modeled as arrays of time-frames and new algorithms for state justification and fault propagation through faulty PLA's are derived. We also give a fault simulation procedure for G and D faults. Experiments show that test generation can be orders of magnitude faster and achieves a coverage of gate-level stuck faults that is higher than a gate-level sequential-circuit test generator. Results on a broad class of small to large synthesis benchmark FSM's from MCNC support our claim that functional test generation based on G and D faults is a viable and economical alternative to gate level ATPG, especially in a logic synthesis environment. The generated test sequences are implementation-independent and can be obtained even when details of specific implementation are unavailable. For the ISCAS'89 benchmarks, available only in multilevel netlist form, we extract the PM and generate functional tests. Experimental results show that a proper resynthesis improves the stuck fault coverage of these tests.

I. INTRODUCTION

The *growth* (*G*) and *disappearance* (D) faults in the combinational function of a circuit are a subset of the faults normally modeled in the programmable logic array (PLA) implementation [1]. It is known that the tests for G and D faults cover all stuck faults in any two level implementation of the combinational logic [2]. For certain synthesis styles [3], [4], these tests will also cover *all* single **stuck** faults in the multilevel combinational circuit.

The main contribution of this paper is a sequential circuit test generation algorithm based on the G and D fault model and its implementation. Many sequential circuit test generators use the timeframe expansion method where the circuit is represented as an iterative array of its combinational logic [5]. At the core of such a method, there usually is a combinational test generation algorithm. In order to find a test sequence, the test generator repeatedly uses the combinational algorithm. Thus, the overall efficiency depends upon how well this algorithm performs. We model the combinational logic at the functional level by its personality matrix (PM) and develop an efficient cube-based test-generation algorithm to obtain test sequences for G and D faults in the finite state machine (FSM). Our recent research [2], [6] has shown the feasibility of this approach. In this paper, we give the theoretical validation of the fault model along with the algorithms and experimental results on a broad range of synthesized sequential circuits.

Manuscript received March 25, 1994; revised April 21, 1995 and March 27, 1996. This paper was recommended by Associate Editor W. K. Fuchs.

M. K. Srinivas was with the Indian Institute of Science, Bangalore 560 012, India. He is now with the CAIP Center, Rutgers University, Piscataway, NJ 08855 USA.

J. Jacob is with the Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560 012, India.

V. D. Agrawal is with Bell Laboratories, Murray Hill, NJ 07974 USA. Publisher Item Identifier S 0278-0070(96)05039-7.

Earlier approaches to functional test generation for a combinational circuit include the derivation of a universal test set (UTS) assuming specific implementation of the function that is either a unate gate network [7] or some other restricted gate network [7], [8]. A UTS is derived from an extended truth table of minterms and is unique for a given function. In our approach, the tests derived for G and D faults from the minimal sum of products (SOP) form of a function are not unique for the function. For most of the practical circuits that are binate, the size of the UTS is 2" [8], where n is the number of inputs to the function, whereas the size of the tests for G and D faults is much smaller.

Recent approaches [9], [10] to functional test generation for sequential circuits rely on the transition fault model. Although a test for a transition fault is found very quickly, the number of single transition faults can he very large even for relatively small machines. Furthermore, in some cases, it becomes necessary to consider multiple transition faults to achieve adequate single stuck-fault coverage. In our approach, the number of functional (G and D) faults is quite reasonable and is generally of the same order as the number of single stuck faults in gate-level implementations of the sequential function. Still, like other functional approaches, our method also generates implementation independent test sequences. These tests have been shown to achieve high fault coverage of stuck faults in specific multilevel implementations and the test generation can be performed much faster compared to the conventional gate-level methods.

Ghosh et. al. [11] use a cube based technique for justification and propagation on the fault free FSM, employing the ON and OFF sets of the PO's and next state outputs. In our approach, we use the faulty FSM for state justification and fault propagation to generate valid test sequences. We only require the ON sets of the PO and next state functions. Even though we target G and D faults in the extracted PLA, the advantage of this fault model is that the test sequences can be applied to any multilevel implementation of the sequential function synthesized using algebraic factorization. The method of Ghosh et. al. [11] specifically targets the stuck faults of the given implementation. If their method is used to target faults in the two-level logic equivalent circuit, then the tests will he suitable for other implementations. However, the use of our functional technique will still save time in comparison to the gate level technique, as shown in Section IV. Another difference in their approach is that justification sequences always start from a reset state, whereas we continue from the final state of the previous test sequence. Other approaches [12]--[14] use manipulation of the state transition graph (STG) for obtaining justification and propagation sequences. Among these, [12] achieves the best performance through the use of binary decision diagrams.

Our functional test generation method is particularly suited to an automatic synthesis environment. In the synthesis of FSM's, after the state assignment is done, **the** circuit is described as a combinational function. The description at this point is often in the form of Boolean *cubes* and resembles the functional specification of a PLA in the PM form. A nomedundant form of the PM is easily obtained using the available tools [15], and is the input to our test generator. For the tests to retain their fault coverage, it is preferable to use only the testability preserving transformations [3], [4] in the synthesis of multilevel logic from the two-level single-output minimized form.

For circuits that are available only in the multilevel netlist form, we extract the PM. As shown by Kohavi and Kohavi [16], if the PM is irredundant then the tests derived for a subset of the G and D faults will cover all stuck faults in the original circuit. These target faults are obtained from a fanout free transformation of the circuit, allowing fanouts only on PI's. Thus, the tests are specific to the given circuit. To cover stuck faults in all multilevel circuits synthesized through algebraic factoring, one must consider all G and D faults.

Even though most of the multilevel logic is synthesized from a single two-level minimized PM, there are cases like the arithmetic or parity functions where the number of cubes in the two-level SOP form is exponential in the number of PI's. Our method presently cannot handle these cases efficiently. However, the technique can he extended to large gate-level combinational and sequential circuits if we partition them into interconnections of moderately sized functional blocks. Each functional block can then be represented as a PM and justification and propagation algorithms can be extended to handle such an interconnection of PM's. The complexity of test generation in an interconnection of PM's will range between the test generation complexity for a single PM and that for an interconnection of primitive gates like AND, OR, NAND, NOR, and NOT. The partitioning approach needs further investigation.

In Section II, we present the basic technique of test generation for PLA's. The validation of the G and D fault model with theoretical results is presented in Section III. Functional test generation for the synthesized combinational and sequential circuits is described in detail with experimental results in Sections IV and V, respectively. In Section VI, we present the results of functional test generation for general combinational and sequential circuits that are already available in the multilevel form. We also give results after resynthesizing these circuits by algebraic factorization. The results of a prototype implementation of our functional test generator are compared to those of a gate-level commercial ATPG tool, Gentest [17], to demonstrate the efficiency of our approach.

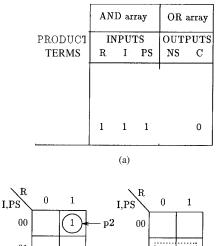
II. BACKGROUND

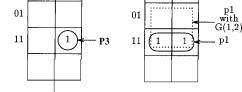
In this section, we review the technique of cube based test generation for combinational PLA's. Fig. 1(a) shows the PM description of the combinational portion of an example FSM having two primary inputs (PI's) R and I. one flip-flop (FF), and one primary output (PO) C. The FF output that feeds hack into the combinational logic is the present state (PS) input and the FF input is the next state output (NS) of the combinational logic. Fig. 1(b) shows the Karnaugh maps of the PO and next state functions. Fig. 1(c) shows a two-level AND-OR implementation of the FSM.

The PM consists of two arrays: the AND array and the OR array. The cubes or product terms in the AND array are denoted as p1, p2, and p3. A cube is a conjunction of literals where the literals are the input variables appearing in their uncomplemented or complemented forms. The inputs are the PI and PS signals. In this case, $p1 = I.PS, p2 = R.\overline{I}.\overline{PS}$, and p3 = R.I.PS. The output functions realized are given by

$$NS = p2 + p3 = R.I.PS + R.I.PS$$
$$C = p1 = I.PS.$$

A missing literal in a product term causes a G fault. If the literal corresponding to the *j*th input x_j is missing from product term pi, the corresponding G fault is denoted as G(i,j). For example, if the literal *I* is missing from product term p1, then this product term will grow as shown by the dotted lines in the Karnaugh map in Fig. 1(b). This is the fault G(1,2). A missing product term from an output function in the OR array causes a D fault. If the product term pi is missing from the *k*th output function, the corresponding D fault is denoted as D(i, k). For example, the fault D(2, 1) will cause the product term p2 to vanish from the Karnaugh map of the first output function NS in Fig. 1(b).





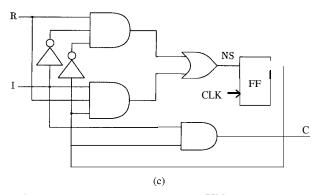


Fig. 1. Functional description of an example FSM. (a) Personality matrix (PM). (b) Karnaugh maps of combinational functions. (c) A two-level implementation of the example FSM.

For the G fault G(i, j), we define the *candidate test cube* (CTC) as the cube pi with the *j*th input complemented. For the D fault D(i,k), the CTC is defined as the cube pi itself. For a PM, we define PTLIST(i,k) as the set of product terms connected to the kth output function excluding pi. For the **PM** in Fig. 1, PTLIST $(2, 1) = p3 = \{111\}$.

We assume that the reader is familiar with the basic cube operations [18] such as union (U), intersection (\Box) , and set difference or sharp (#). Now the tests for the fault G(i, j) and D(i, k) detected on the kth output (provided pi is connected to this output) can be given as CTC # PTLIST(i, k), where the appropriate definition of CTC is used. A G fault may be detectable on any output fed by the affected product term, but a D fault can only be detected at the output whose function is affected by the fault.

Consider the fault G(2,3) in the PM of Fig. 1. CTC = $\{101\}$; PTLIST(2,1) = $p3 = \{111\}$. Hence test for G(2.3) = CTC # PTLIST(2.1) = $\{101\}$. The computation of tests for G and D faults using the above method is straightforward and allows

very efficient implementation [1], [19]. The generation of a complete test sequence for FSM's is discussed in detail in Section V.

III. FAULT MODEL

The functional faults we consider for test generation are a subset of the *crosspoint faults*, commonly used to model defects in PLA's [1]. Of the four types of crosspoint faults, namely, growth (G), shrinkage (S), appearance (A), and disappearance (D) faults, we have chosen only the G and D faults as the target faults for combinational and sequential logic circuits. We will refer to them as *functional faults* since they have a direct representation in the Boolean function of the circuit. The primary usefulness of G and D faults stems from their ability to model stuck faults in irredundant two-level circuits and a certain class of multilevel combinational circuits as shown by the following results available in the literature.

- i) All single stuck faults in an irredundant two-level single or multiple output circuit are detected by the tests for G and D faults of the equivalent PLA, provided the tests set each PO to zero at least once [20].
- ii) In an irredundant two-level circuit in which all single stuck faults are detectable, the test vectors for all single stuck faults will also detect all multiple stuck faults, provided we can find an ordering $z_1 \dots z_q$ among the q output functions such that all stuck faults in the subcircuit feeding output z_j are detected via one or more outputs $z_1 \dots z_i$ ($1 \le i \le j \le q$) [20]. This result, together with result i) implies that the test vectors that detect all single G and D faults in the equivalent PLA, will detect all multiple stuck faults in the irredundant two-level multiple output circuit provided the vectors conform to an output ordering constraint. Such a constraint can be easily satisfied by any test generator.
- iii) If we only use algebraic factorization of the minimized irredundant two-level single output Boolean function to realize a multilevel circuit, then all multiple faults in the multilevel circuit will be testable [3]. Also, the test vectors that detect all single stuck faults in the irredundant two-level single output circuit will cover all single and multiple stuck faults in the synthesized multilevel circuit.
- iv) Testability preserving transformations consisting of algebraic factorization, applied to any prime and irredundant single output minimized two-level combinational circuit, preserve single fault testability [4]. This means that the tests for all single stuck faults in the original two-level circuit will cover all single stuck faults in the synthesized (transformed) multilevel circuit. In all of our experiments, it was observed that the functional vectors gave 100% fault coverage for single stuck faults in the synthesized multilevel circuits.

For our theorems and experiments we use testability preserving synthesis that consists of the following.

- 1) Single output minimization of the PM that guarantees the function to be prime and irredundant with respect to every output and, hence, completely multifault testable;
- 2) Synthesis of the above PM using only algebraic factorization.

According to our experience, single-output minimization may not lead to any significant increase in area compared to multiple-output minimization. We conducted an experiment on 26 of the MCNC91 synthesis benchmark FSM's and 18 of the ISCAS89 benchmark circuits. For each circuit we obtained both single-output and multipleoutput minimized forms using ESPRESSO. We then synthesized these functions using SIS 1.1 program obtained from University of California, Berkeley, which implements single cube, multiple cube, and complement extraction. The program provides the size of multilevel implementations in terms of the literal count. For the 44 circuits we synthesized, the multiple-output minimization procedure required -9.3% to +69.0% more literals as compared to the single-output minimization procedure. Rajski and Vasudevamurthy [4] use a synthesis technique consisting of single cube, double cube, complement and dual expression extraction that results in a large area overhead for single-output minimized circuits. In their results, the area overhead for multiple-output minimization and synthesis compared to single-output minimization and synthesis was -49.1%to +7.1%. The smaller overhead for single-output minimized circuits as we obtained by SIS is probably due to multiple-cube extraction in synthesis. It is worth mentioning that the circuit speed can be improved by retiming transformations that are known to preserve the fault coverage of tests [21]. Preceding results lead to the following theorem.

Theorem1: The test sequences for the single G and D faults of a single output minimized PM will cover all single (and multiple) stuck faults in the multilevel combinational circuit that is synthesized using testability preserving transformations.

Proof: A single output minimized PM is prime and irredundant with respect to every output and there is no product term sharing for any output. Let N1 be a single output minimized PM, N2 the equivalent two-level AND-OR circuit, and let N3 be synthesized from N1 using only algebraic factorization. Algebraic factorization ensures that N3 is also prime and irredundant with respect to every output [4]. The test set for single G and D faults in N1 will cover all multiple G and D faults in N1 and all single and multiple stuck faults in N2 as the output ordering criteria according to result ii) is automatically satisfied. Now every single and multiple stuck fault in N3 has an equivalent single or multiple stuck fault in N2 according to results iii) and iv). Therefore, the test vectors derived for single G and D faults of N1 will cover all single and multiple stuck faults in the multilevel combinational circuit N3 that is synthesized using testability preserving transformations.

Theorem 1 is not directly applicable to sequential circuits or FSM as the results i)-iv) are valid only for combinational circuits. It should be noted that the tests for G and D faults of the PM of an FSM do not guarantee a complete coverage of all single stuck faults on those PI's and PS stem lines that have fanouts leading only to the next state (NS) lines (and not to any PO), and the multiple stuck faults (whose fault effects lead only to NS lines) in the equivalent two-level (AND-OR) FSM (and the multilevel synthesized circuit). This is due to the fact that these PI and PS stem faults and the multiple stuck faults are equivalent to multiple G and D faults, and can be masked in the reconvergence structure across the time frames in an Iterative Logic Array (ILA) model of the FSM, if they are not explicitly targeted for test generation. If, however, the G and D fault tests of a single output minimized PM of an FSM cover all single stuck faults of an equivalent two-level (AND-OR) FSM, then these tests will also cover all single stuck faults of the multilevel FSM whose combinational portion is synthesized using testability preserving transformations. For complete single stuck fault coverage of the synthesized FSM, in addition to the single G and D faults, a few of the multiple G and D faults that are equivalent to the PUPS stem faults may have to be considered for test generation (only if they are not already covered by the single G and D tests). Test generation for multiple G and D faults in our cube based algorithm is as simple as test generation for single G and D faults. Introduction of multiple G and D faults involves changing more than one bit in the PM for the faulty FSM. This multiple fault is introduced only during justification and propagation, whereas for the activation vector any of the tests for the constituent single G or D fault will be sufficient to activate the multiple fault.

Theorem 2: The test sequences for the single G and D faults in a single output minimized PM of an FSM will cover all single stuck

faults, except the faults on the PI and PS stem lines (that have fanouts leading only to NS lines and not to any PO) in the equivalent twolevel AND-OR FSM and the multilevel FSM whose combinational portion is synthesized using testability preserving transformations.

Proof: By results i) and ii), the single G and D faults of a single output minimized PM will cover all single and multiple faults in the equivalent two-level AND-OR combinational circuit. This is because the output ordering requirement is implicitly satisfied by a single output minimized PM, as there are no shared product terms.

Let N1 be a single output minimized PM and N2 the equivalent two-level AND-OR circuit. The multilevel circuit N3 is synthesized from N1 using only algebraic transformations. According to Theorem 1, the vector set T1 that detects all single G and D faults in N1, will also detect all multiple G and D faults in N1, and all single and multiple stuck faults in N2 and N3, as every fault in N3 has an equivalent fault in N2 [3], [4].

Now FF's are added to N1, N2, and N3 to form sequential circuits. Some of the PI's which are transformed to PS inputs become uncontrollable and some of the PO's which are transformed into NS outputs become unobservable in a single time frame. Now the single G and D fault test set for FSM N1 does not guarantee to cover all multiple G and D faults as these faults may get masked in the fanout reconvergence structure across time frames in the ILA model of the FSM.

The vector set T1 (derived for N1) consists of activation vectors for all faults (single and multiple) in FSM's N2 and N3. The STG's for FSM's N1, N2, and N3 are identical for the good circuits. For every single fault f 3 in FSM N3 there is an equivalent single fault f_2 in FSM N2 [4], and an equivalent single G or D fault f_1 (or a multiple G and D fault for the PUPS stems that have fanouts) in FSM N1. If t_1 is the activation vector in T1 for f_1, t_1 is also the activation vector for f_2 and f_3 . The STG with f_3 for FSM N3 is exactly the same as the STG with f2 for FSM N2, and the STG with f1 for FSM N1. Therefore, if PS of t1 can be justified in FSM N1, it can also be justified in FSM's N2 and N3, and the justification sequence is the same for all three FSM's, N1, N2, and N3. Similarly, if the good and faulty NS of tl can be differentiated in FSM N1, they can be differentiated in FSM's N2 and N3, and the propagation sequence will be same in all FSM's. The presence of a justification and propagation sequence for a fault entirely depends on the STG and is independent of the structure of any particular implementation. As the PIPS stem (that have fanouts leading only to NS lines and not to any PO) faults in FSM's N2 and N3 are equivalent to a multiple G and D fault in FSM N1, these faults are not guaranteed to be detected by the test set for single G and D faults of FSM N1. The single G and D faults that are equivalent to all other single stuck faults (except PI/PS stem faults having no branches leading to a PO) of the FSM's N2 and N3, are explicitly considered for test generation in the FSM N1. The stuck faults on the stem of a PI/PS line which have at least one fanout branch leading to a PO, will be detected by the single G or D fault on this fanout branch leading to the PO, as such faults will be detected in a single time frame. Hence, if all single G and D faults are detected in the FSM N1 by the test sequence S1, the same sequence S1 will cover all single stuck faults except the PI and PS stem line (that do not have a fanout branch leading to a PO) faults and the multiple stuck faults in the equivalent two-level FSM N2 and the synthesized FSM N3.

IV. FUNCTIONAL TEST GENERATION FOR SYNTHESIZED COMBINATIONAL CIRCUITS

We implemented a new cube based test generation and fault simulation program, GDCOMB, in C language. Tests can be generated using any PLA test generation algorithm [19], [22]. However, instead

N FOR CO	MBINATIONA	l Part c	of FSM's	(SUN SPA	RCSTATION	2)	
Perso	nality N	rix	N	Aulti-leve	l Implem	entatio	n
	(COM				entes	
G-D	No, of	CPU	Stuck	Faults	No. of	cov	CPU
Faults	Vect.	Sec.	Total	cov %	Vect.	%	Sec.
48	12	0.01	69	100	13	100	0.12
48	16	0.01	76	100	15	100	0.18
30	10	0.01	40	100	8	100	0.06
28	10	0.01	41	100	11	100	0.06
53	21	0.01	77	100	17	100	0.12
45	18	0.01	62	100	10	100	0.12
42	13	0.01	61	100	11	100	0.06
881	155	0.17	567	100	67	100	26.25
314	77	0.01	318	100	52	100	1.13
140	36	0.01	152	100	27	100	0.36
166	32	0.01	196	100	24	100	0.36
120	28	0.01	140	100	27	100	0.36
360	55	0.02	299	100	38	100	0.89
149	32	0.01	166	100	30	100	0.48
136	30	0.01	153	100	23	100	0.30
165	46	0.01	191	100	27	100	0.48

306

331

428

759

132

244

526

1066

1087

1127

100

100

100

100

100

100

100

100

100

100

50

51

74

112

23

30

73

108

139

149

100

100

100

100

100

100

100

100

100

100

1.07

1.25

2 27

5.07

0.36

0.84

3.94

11.81

13.96

16.28

 TABLE I

 Test Generation for Combinational Part of FSM's (SUN SPARCSTATION 2)

95

103

151

275

44

44

106

204

346

369

0.06

0.06

0.12

0.41

0.01

0.01

0.06

0.29

0.53

0.53

352

383

590

1316

126

239

625

1341

1547

1678

of considering all crosspoint faults, we only generate tests for G and D faults, and perform fault simulation after every vector is generated. Fault simulation involves finding the Hamming distance between the test vector and the product terms. A G fault (D fault) on **a** product term is detected if the distance between the product term cube and the test vector is one (zero) and no other product term cube with **a** zero distance from the test vector is connected to the same output.

Circuit

Name

train4

train11

bbtas

dk27

ex1

ex2 ex3

ex4 ex5

ex6

ex7

dk17

opus

bbsse

sse

cse

 $\mathbf{s1}$

bbara

dk14

dk16

sand

styr

planet

lion

mc lion9 PI, PO,

Prod. Terms

5, 7, 14

6, 5, 16

4, 3, 10

4, 3, 9

6, 5, 17

5, 5, 13

4, 5, 12 14, 24, 145

7, 7, 62

6, 6, 28 10, 13, 39

6, 6, 33

8, 11, 75

6, 6, 40

5.5.31

9, 10, 40

11, 11, 61

11, 11, 62

11, 11, 95

13, 11, 188

7, 5, 26

6, 8, 51

7, 8, 104

13, 25, 235

16, 14, 228

14, 15, 228

We employed GDCOMB to derive tests from the PM description of the combinational portion of 26 of the MCNC synthesis benchmark FSM's. These results are given in Table I. GDCOMB derived vectors to cover all G and D faults. The multilevel combinational circuits were synthesized from the single-output minimized twolevel description, employing *algebraic* factorization and a simple technology mapping scheme that uses only primitive gates of up to four inputs and inverters. The synthesis system MIS [23] was used in our experiments. Single-output minimization does not lead to any large increase in area (10 out of 44 circuits had up to 10% increase in area) compared to multiple-output minimization. Any degradation in performance can be improved by retiming transformations, which preserve the fault coverage of tests [21]. Functional vectors (derived by GDCOMB) were then used to simulate all collapsed single stuck faults in synthesized multilevel implementations of these circuits. The coverage, as shown in Table I, was 100% for all circuits. Since only testability preserving transformations [3], [4] were employed, the 100% fault coverage was expected. The two-level and multilevel combinational circuits of the FSM's were irredundant with respect to single stuck faults.

Table I also gives the results of test generation for stuck faults in multilevel circuits by a gate-level test generator, Gentest [17]. While

both test generators could cover all faults, the run times of GDCOMB are significantly better. Vector sets of Gentest are, however, smaller. This is because the vector sets of GDCOMB are independent of the implementation. Such implementation-independent tests can also be derived from Gentest if vectors are generated for all single stuck faults in two-level AND-OR circuits. Actual experiments showed that the fault set size and vector set size were comparable to those of the G and D faults and the GDCOMB tests, but the run times of Gentest were even higher than those given in Table I. The use of test vectors generated from two-level AND-OR description was reported by Dave and Patel [24].

To examine the importance of testability preserving transformations in synthesis, we experimented with the circuit styr. A multilevel implementation of this circuit was synthesized from a multiple-output minimized PM description of the function, and a technology mapping that used a standard-cell library. Multiple-output minimization does not guarantee the function to be prime and irredundant with respect to every output. The multiple output minimized PM of styr had only 118 cubes as compared to 228 in the single-output minimized version, and the number of gates in the synthesized multilevel circuit were 277 and 379, respectively. GDCOMB obtained a test set of 391 vectors for the 1239 G and D faults in the PM description in 0.7 s. However, the vectors covered only 99.45% (i.e., 1085 of 1091) of detectable stuckfaults in this multilevel implementation. Multiple-output minimized circuits need not be prime and irredundant with respect to each output, which is a necessary condition for preservation for multifault testability. Multiple-output minimized circuits may contain redundant multiple faults which would transform into a redundant single fault after algebraic factorization as shown by the counter example in [3].

Testgen(M) FSM M; Goodstate(M) = Reset;For every undetected fault F do { If Faultystate(F) \neq Goodstate(M) then { Pseq = Propagate(F); /* Step 3 */If propagated to PO then Faultsim(Pseq); } else { $Cvecs = Generate_activation_vector(F); /* Step 1 */$ If Cvecs generated then { Jvecs = Justify(Cvecs); /* Step 2 */ If justified then Pseq = Propagate(F); /* Step 3 */ If justified and propagated to PO then Faultsim(Jseq+Pseq); } } }

Fig. 2. The main test generation algorithm.

Results in [4] show that multiple-output minimization leads to a minor loss in single stuck fault coverage and most of those undetected faults are redundant.

V. FUNCTIONAL TEST GENERATION FOR SYNTHESIZED FSM'S

We use an extension of the PLA test generation method described in Section II to derive tests for the G and D faults in an FSM. The algorithm consists of the following three steps:

> Step 1: Combinational Test generation. Step 2: State Justification.

Step 3: Fault Propagation.

We sandwich the combinational test vector from Step 1 (derived using the cube based algorithm of Section 11) between the state justification and fault propagation sequences to obtain a complete test sequence for the G or D fault under consideration. A functional fault simulator for G and D faults was implemented and was used to reduce the fault list after the test sequence for a fault is generated. If the target fault is already activated (i.e., its effect has reached an NS line) by previously generated vectors, only fault propagation (Step 3) to a PO is performed. The pseudocode for the main test generation algorithm is shown in Fig. 2.

5.1 State Justification

The combinational test generation procedure, as implemented in GDCOMB produces the list of all test vectors C_t for a fault f with possible *don't care* entries for some PS bits. Any one of these vectors is sufficient to activate the fault in the time frame t. The state S_t (**PS** part of C_t) corresponding to any chosen vector must be justified to the state S, in which the fault-free FSM was left after the application of the last vector in the previous test sequence (generated for the preceding fault). The excitation state for the very first fault considered is justified to the reset state, assuming that the machine can be reset at power up even under faulty conditions. If an initialization sequence or a hardware reset is available, this could be any other minterm state instead of the reset (all 0) state. This assumption, used for simplicity in the present implementation, is not a basic limitation of our technique and can be removed in future implementations. Since all cube operations given in Section II are defined for the three-state logic (0, 1, X), a justification sequence starting from a completely

unknown state can be derived. Excitation states for all other faults are justified to the state in which the FSM was left after the application of the test sequence for the preceding fault.

Justification involves finding a sequence of vectors that will bring the FSM from the state S_t to the state S_t . In our implementation, reverse time processing is used to generate a justification sequence, starting from current state S_t , back to S_t . If any of the vectors generated in time frame t has the PS S_t which covers the state S_p , we do not have to generate a justification sequence for the fault under consideration. If no such cube exists, first we check if a single vector justification sequence exists. This can be done by finding all input cubes that have the fanin states of S_t (PS part) in the previous time frame t + 1. These cubes can be found by simple cube intersection (\square) and sharp (#) operations on the ON set cubes of the next state functions (with the fault) corresponding to the state S_t .

As an example, suppose we want to find all input cubes that will set the three next state signals in t + 1 as $NS_1 = 1, NS_2 = 0$, and $NS_3 = 1.$ i.e., $S_t = \{101\}$. The input cubes that have the famin states of S_t are given by $NS_1 \sqcap \overline{NS_2} \sqcap NS_3 = (NS_1 \# NS_2) \sqcap NS_3$, where NS_1, NS_2 , and NS_3 are the faulty ON sets of the corresponding next state lines. Once we find the famin cubes C_{t+1} of S_t , we search C_{t+1} for a cube whose PS portion covers S,. If such a cube exists in C_{t+1} , we have found a single vector justification sequence. Otherwise, we heuristically select a cube in C_{t+1} whose PS part S_{t+1} does not subsume S_t (to avoid self loops), find the fanin cubes C_{t+2} of S_{t+1} and search for a cube in C_{t+2} whose PS portion covers S_p . This process is continued until either we reach a predefined limit on the length of the justification sequence or a user defined time limit or when the set of fanin cubes for a given S_{t+x} becomes empty, at which point backtracking is started. Backtracking is done by advancing the time frame forward (by decrementing t) and choosing the next available cube among the fanin cubes to be justified. If no more cubes are available, advancing the time frames continues until we exhaust all cubes in C_t of the time frame t = 0.

One significant advantage of our G and D fault model is that the effect of a fault can be represented by a single bit change in one of the cubes constituting the ON set of the affected function. For example, the effect of a G fault G(i, j) is to change the jth variable in cube pi to X from 0 or 1. Similarly, the effect of a D fault D(i, k) is to change the kth entry in the output part of cube pi from 1 to 0. A G fault causes the expansion of the affected cube in the functions fed by it, as seen on the Karnaugh map [see Fig. 1(b)]. A D fault will cause a cube to disappear from the affected function, and thus cause a contraction of the function as seen on Karnaugh map.

The justification procedure used here differs from the one used in [11] in the following aspects:

We use only the ON sets of the PO and PS functions. Thus we save on derivation and storage of OFF sets.

We use the FSM with the fault introduced during justification. They use fault-free justification repeatedly until a test is found. We do not reuse the justification sequences already generated, as we justify to the last state from the preceding test sequence in the faulty FSM, unlike [11] which always seeks justification to a reset state in the fault free FSM.

5.2 Fault Propagation

This step is required if the initial combinational test vector generated for the target fault does not propagate the effect of the fault to any PO but only to one or more next state lines. We will illustrate the algorithm using the example FSM of Fig. 1. Consider the G fault G(2,3) in the FSM. Step 1 yields the combinational test vector 101 as we derived in Section II. This implies that 10 is the test vector and the machine should be in state 1 for this test to be effective. Let

836

	No. of	No. of	No. of	Personal	y Matrix	Synthesi:	d Circuit
FSM	Inputs	outputs	Flip-	Product	G-D	Number	Stuck
		-	Flops	Terms	Flts.	of Gates	Flts.
mc	3	5	2	14	48	21	69
lion9	2	1	4	16	48	26	76
train4	2	1	2	10	30	13	40
lion	2	1	2	9	28	13	41
train11	2	1	4	17	53	25	77
bbtas	2	2	3	13	45	21	62
dk27	1	2	3	12	42	19	61
ex1	9	19	5	145	881	198	567
ex2	2	2	5	62	314	113	318
ex3	2	2	4	28	140	50	152
ex4	6	9	4	39	166	66	196
ex5	2	2	4	33	120	47	140
ex6	5	8	3	75	360	106	299
ex7	2	2	4	40	149	58	166
dk17	2	3	3	31	136	54	153
opus	5	6	4	40	165	65	191
bbsse	7	7	4	61	352	105	306
sse	7	7	4	62	383	112	331
cse	7	7	4	95	590	157	428
s 1	8	6	5	188	1316	279	759
bbara	4	2	3	26	126	42	132
dk14	3	5	3	51	239	85	244
dk16	2	3	5	104	625	182	526
planet	7	19	6	235	1341	360	1066
sand	11	9	5	228	1547	354	1087
styr	9	10	5	228	1678	379	1127

 TABLE II

 CHARACTERISTICS OF BENCHMARK FSM's

us assume that we generated a justification sequence successfully. In step 3, we require a sequence that will propagate the fault effect to a PO. Since the fault effect has reached the next state output NS we look for a product term of a PO function which uses the PS input corresponding to this output NS and derive the input conditions to observe this product term signal at the PO. This can be achieved by simple cube operations. Let pi be the chosen product term which uses PS_{i} , connected to PO z_{k} and does not have conflict in other PS positions with the output on the corresponding next state lines. We derive PTLIST(i, k) for pi with respect to output z_k . In order to avoid simultaneous propagation of D and \overline{D} values to the PO, we change the bit corresponding to input PS in the cubes in PTLIST(i,k) to x if it differs from the value of the bit position for PS in pi. PTLIST also consists of cubes that do not have any conflict with p_i and are connected to z_k . Now, $p_i \# PTLIST(i, k)$ will give the required sensitizing condition. In our example (Section 11), p1 is the only product term for the output C which is the only PO. PTLIST(1,2) is empty. Hence the propagation sequence is X1.

If during step 3, we cannot find a product term of **a** PO that uses the required **PS** variable, we propagate the fault effect to some other next state output before we finally reach **a** PO, taking care not to allow propagation to the same next state line as in **a** previous time frame. To check for fault masking due to multiple path propagation, **a** fault simulation just for the fault under consideration is done for every propagation vector (in every time frame) as soon as it is generated. If there is fault masking, then **a** different product term is chosen for propagation.

For fault propagation, we always use the ON set of the faulty function, if the function is affected by the fault. Note that we can obtain the ON set of a faulty function by \mathbf{a} simple bit change in one of its cubes. This propagation algorithm is quite different from

the one discussed in [11], **as** we use only the ON sets, and always generate valid propagation sequences with the fault introduced.

5.3 Fault Simulation

A simple fault simulator based on the single-fault propagation technique is implemented. When **a** test sequence is found for a fault, the vectors in the sequence are run through the fault simulator. For each vector first a fault free logic simulation is carried out with the PI portion concatenated with the fault free state bits for the PS portion. The next state bits (fault free state) is saved for consideration with the next vector. Next, for each fault, modifications are introduced in the PM and the simulation is repeated with the PI portion of the vector concatenated with the PS portion in the faulty machine for the fault under consideration. The output responses are compared in the PO portion to check whether the fault is detected. If the fault is not detected, the next state bits (faulty state) are saved for the particular fault for consideration with the next vector.

Since we use the cube description of the logic function, simulation requires finding the Hamming distance of the vector to product terms. The outputs of all those product terms that have a distance of zero with the vector will be set to logic 1, and all other outputs will be set to logic 0.

5.4 Experimental Results

We developed a *C* program, GDSEQ, to generate test sequences for **PLA** based FSM's and general sequential circuits whose combinational function can be obtained in PM form. We experimented on 26 of the synthesis benchmark FSM's. The characteristics of these circuits are shown in Table II. These circuits were available as symbolic state tables. The PM of the combinational portions of

	Pers	onality M	atrix		Mu	lti-level l	tplemen	tation	
FSM			GD	SQ				Gentest	
	No. of	G-D	TGen.	FSim.	Useful	Stuck	No. of	Stuck	CPU
	Vect.	Flt.	CPU	CPU	Vect.	Fault	Vect.	Fault	Sec.
		cov %	Sec.	Sec.		Cov %		cov %	
mc	31	100.0	0.01	0.06	31	100.0	25	100.0	0.2
lion9	42	97.91	0.06	0.06	42	98.7	60	98.7	3.0
train4	46	100.0	0.01	0.06	26	100.0	14	97.5	0.1
lion	16	100.0	0.01	0.01	14	100.0	19	100.0	1.0
train11	41	86.79	0.06	0.12	41	90.9	60	97.4	3.0
bbtas	59	93.33	0.01	0.01	59	100.0	44	100.0	2.0
dk27	19	97.61	0.01	0.01	19	98.4	27	98.4	0.7
ex1	690	99.54	9.54	3.28	690	99.8	251	98.4	4693.0
ex2	283	100.0	1.19	1.19	283	100.0	247	98.7	2022.0
ex3	102	100.0	0.06	0.24	102	100.0	110	100.0	18.0
ex4	107	95.18	0.12	0.24	107	97.9	98	32.6	106.0
ex5	68	100.0	0.12	0.18	61	100.0	103	100.0	41.0
exв	125	96.66	0.36	0.59	97	97.6	103	97.6	35.0
ex7	58	61.74	0.76	0.21	58	80.7	83	80.1	1118.0
dk17	69	100.0	0.59	0.12	69	100.0	66	100.0	48.0
opus	107	100.0	0.30	0.24	107	100.0	105	100.0	28.0
bbsse	493	100.0	1.61	1.31	486	100.0	149	99.7	125.0
sse	439	93.47	1.49	1.37	439	99.1	233	98.8	395.0
cse	676	100.0	4.53	2.56	676	100.0	340	98.6	470.0
s1	1141	85.86	44.14	6.56	1141	98.8	304	93.5	3925.0
bbara	151	100.0	0.17	0.37	150	100.0	120	100.0	10.0
dk14	89	100.0	0.12	0.39	79	100.0	64	100.0	10.0
dk16	191	100.0	1.56	1.44	179	100.0	346	100.0	1403.0
planet	644	99.9	50.80	5.39	624	100.0	509	100.0	8446.0
sand	1536	100.0	56.66	12.52	1519	100.0	580	100.0	22425.0
styr	1237	100.0	90.66	12.52	1208	100.0	754	100.0	35868.0

 TABLE III

 FUNCTIONAL TEST GENERATION (GDSEQ) VERSUS GENTEST FOR SYNTHESIZED FSM's (SUN SPARCSTATION 2)

 TABLE IV

 FUNCTIONAL TEST GENERATION FOR LARGE FSM'S (RS 6000/580)

FSM	PI	РО	FF	Prod.	GD	Stuck	Gates	CPU	Vecs	GD Flt	Stuck Flt
				Terms	Flts	Flts		sec		$\operatorname{Cov}(\%)$	$\operatorname{Cov}(\%)$
scf	27	56	7	385	2989	1665	583	28	1228	78.7	93.5
sbc	40	56	28	555	3194	1726	564	1006	899	54.7	74.2
dsip	228	197	224	4022	25128	6491	2370	10541	4572	61.4	92.7
\mathbf{key}	258	193	228	5926	39576	8466	3089	10044	2612	88.3	96.4
bigkey	262	197	224	5926	39576	9051	3563	7625	2146	85.3	95.0

these FSM's were obtained by state assignment using MUSTANG [25] and single output logic minimization using ESPRESSO [15]. The multilevel FSM's were obtained from the PM by synthesis using MIS [23] performing only algebraic factorization. The last six circuits in Table II were state minimized using STAMINA [26] before state assignment and synthesis.

The results obtained from GDSEQ are given in Table III. GDSEQ generated test sequences for all G and D faults in 15 of these circuits. The coverage of G and D faults in other circuits was lower due to reasons like sequential redundancy, time frame limit, and the backtrack time limit used in justification stage of the program. **As** stated earlier, a power-up reset was assumed only at the beginning of the test sequence.

Next, the GDSEQ vectors were used to simulate all collapsed single stuck faults in the multilevel gate implementations of FSM's using a differential fault simulator [27]. **As** shown in Table III, these

vectors gave 100% fault coverage for 18 of the synthesis benchmarks. The lower stuck fault coverage for the remaining circuits is due to reasons **like** GDSEQ aborting on some G and D faults, sequential redundancies in the FSM, and the presence of PI/PS stem faults not guaranteed to be covered by tests for single G and D faults (see Theorem 2, Section 111).

It should be noticed that the stuck fault coverage is always higher than that of the G and D fault coverage. The *useful vectors* given in Table III were obtained when the vector set was truncated after detection of the last fault.

We used the latest version of the sequential test pattern generator Gentest [17] to verify the efficiency of GDSEQ. Gentest is a gatelevel test generator and uses the time frame expansion method. It has a differential fault simulator [27] to remove detected faults from the fault list after a test sequence is generated for a target fault. Besides being available to us, Gentest is quite comparable

	No. of	No. of	No. of	Extract	ed PLA	Multile	vel Circuit	SUN Spare	2 CPU s
FSM	In-	Out-	Flip-	Prod.	G-D	No. of	Stuck	Cube	Mini-
	puts	puts	Flops	Terms	Flts.	Gates	Flts.	Enumeration	mization
s27	4	1	3	15	47	10	32	0.12	0.14
s208	11	2	8	30	202	96	215	0.29	0.49
s298	3	6	14	68	309	119	308	0.96	0.96
s344	9	11	15	249	1495	160	342	2.39	2.37
s349	9	11	15	249	1495	161	350	2.42	2.37
s382	3	6	21	167	1080	158	399	2.32	1.62
s386	7	7	6	51	407	159	384	1.10	0.61
s400	3	6	21	167	1080	162	424	2.63	1.71
s444	3	6	21	167	1080	181	474	2.89	1.58
s510	19	7	6	109	580	211	564	1.58	0.75
s526	3	6	21	142	740	193	555	1.93	1.40
s526n	3	6	21	142	740	194	553	1.84	1.40
s820	18	19	5	126	969	289	850	0.92	2.98
s832	18	19	5	126	969	287	870	0.92	2.98
s1196	14	14	18	1050	10077	529	1242	54.31	41.26
s1238	14	14	18	1050	10073	508	1355		39.57
s1488	8	19	6	277	1876	653	1486	1.05	3.46
s1494	8	19	6	277	1876	647		1.09	3.28

TABLE V CHARACTERISTICS OF ISCAS'89 BENCHMARK CIRCUITS

 TABLE VI

 Test Generation for COMBINATIONAL PART of ISCAS'89 Circuits (SUN SPARCStation 2)

		Perso	onalitv M	atrix	Multi-level Implementation					
Circuit								Gentest		
Name	Prod. Terms	G-D	No. of		Stuck	Faults	No. of	cov	CPU	
		Faults	Vect.	Sec.	Total	Cov %	Vect	%	Sec.	
s27	7, 4, 15	47	13	0.01	32	100.0	11	100.0	0.06	
s208	19, 10, 30	202	95	0.06	215	100.0	43	100.0	0.72	
s298	17, 20, 68	309	92	0.06	308	94.8	51	100.0	1.01	
s344	24, 26, 249	1495	339	1.01	342	98.8	35	100.0	0.89	
s349	24, 26, 249	1495	339	1.01	350	98.3	32	99.4	0.89	
s382	24, 27, 167	1080	214	0.47	399	98.5	58	100.0	1.31	
s386	13, 13, 51	407	143	0.06	384	100.0	83	100.0	2.21	
s400	24, 27, 167	1080	214	0.41	418	85.9	52	86.1	1.67	
s444	24, 27, 167	1080	219	0.47	474	96.6	54	97.0	1.55	
s510	25, 13, 109	580	120	0.17	564	100.0	78	100.0	2.68	
s526	24, 27, 142	740	216	0.35	555	96.9	92	99.8	2.45	
s526n	24, 27, 142	740	216	0.35	553	97.1	92	100.0	2.39	
s820	23, 24, 126	969	301	0.47	850	99.1	178	100.0	8.29	
s832	23, 24, 126	969	308	0.47	870	97.2	169	98.4	8.58	
s1196	32, 32, 1050	10077	2148	23.26	1242	99.6	211	100.0	14.02	
s1238	32, 32, 1050	10073	2170	22.84	1355	94.3	222	94.9	19.39	
s1488	14, 25, 277	1876	306	0.59	1486	99.5	178	100.0	16.05	
s1494	14, 25, 277	1876	306	0.59	1506	98.8	178	99.2	16.99	

in performance to other state-of-the-art test generation programs [28]. We generated test sequences for stuck faults using Gentest in the multilevel implementations of FSM's. The vector set size, fault coverage (detected faults/total faults), and CPU times for Gentest on SUN SPARCstation 2 are given in Table III. Here also, power up reset is assumed. It may be seen that the stuck fault coverage of functional vectors is always equal or higher than the stuck fault coverage obtained by Gentest vectors, except in the case of *train11*. In *train11* the fanout branches of both the PI's and one PS lead only to NS lines resulting in some of the stem faults not being detected by the single G and D fault test sets, and there were also a few propagation failures leading to a lower G and D fault coverage. Further, the CPU

time for GDSEQ (including time to simulate the functional vectors) is far less than the test generation time taken by Gentest. GDSEQ turns out to be up to 1152 times faster on these circuits.

To examine the advantage of using testability preserving transformations in the synthesis of sequential circuits, we experimented with the circuit *styr*. The multilevel combinational portion of *styr* synthesized from a multiple-output minimized PM (as described in Section IV) was used to construct the multilevel FSM. GDSEQ generated a test sequence of 1354 vectors for the 1239 G and D faults in multiple-output minimized PLA based FSM. These vectors gave only 98.9% coverage (i.e., 1080 of 1091) of detectable stuck faults in the multilevel implementation, as opposed to the 100% coverage

	Pers	onalitv M	atrix	Multi-level Implementation							
FSM			GDS	SEO				Gentesi			
	No. of	G-D	TGen.	FSim.	Useful	Stuck	No. of	Stuck	CPU		
	Vect.	Flt.	CPU	CPU	Vect.	Fault	Vect.	Fault	Sec.		
		cov %	Sec.	Sec.		cov %		cov %			
s27*	23	89.4	0.02	0.03	14	100.0	17	100.0	0.1		
s208	260	44.6	0.56	2.19	232	69.3	134	69.8	6.6		
s298	187	72.8	2.84	0.83	182	86.0	184	88.3	530.0		
s344	129	26.1	700.50	1.16	129	90.6	138	97.7	972.0		
s349*	122	26.9	690.90	0.61	122	91.7	57	86.0	658.0		
s382*	907	39.8	448.60	4.80	677	87.4	789	76.2	2129.0		
s386*	317	64.1	1.66	1.75	296	81.7	220	81.3	2666.0		
s400*	572	35.2	343.77	4.21	543	81.6	3149	76.4	4213.0		
s444	387	30.6	81.14	4.38	387	75.5	2073	77.6	1882.0		
s510*	550	99.1	35.74	2.14	550	100.0	53	89.2	1622.0		
s526	330	44.9	306.44	4.95	330	67.6	2256	67.9	4106.0		
s526n	312	41.5	312.50	6.18	257	65.6	2250	69.6	7574.0		
s820*	1345	88.6	145.80	14.47	1345	94.0	379	87.7	20624.0		
s832*	1527	88.4	96.05	17.98	1500	92.1	446	88.2	21235.0		
s1196	4512	99.6	416.35	45.92	4061	98.4	344	99.8	27.0		
s1238*	4722	99.6	451.96	83.51	4231	93.1	185	82.4	42.0		
s1488*	1502	83.5	116.49	43.85	1502	94.2	528	89.5	35419.0		
s1494*	1529	85.1	115.04	43.85	1529	94.5	489	91.0	47413.0		

 TABLE VII

 FUNCTIONAL TESTING (GDSEQ) VERSUS GENTEST FOR ISCAS'89 CIRCUITS (SUN SPARCSTATION 2)

obtained for the testable implementation reported in Table III. The lowering of fault coverage due to multiple-output minimization, though not large, is noticeable.

In Table IV, we give results for some of the larger FSM's that were available in the Berkeley Logic Interchange Format (BLIF). In BLIF, the FSM is described as a network of interconnected functional blocks where each functional block is a single output SOP. We used MIS to convert from BLIF to a single PM by the functions read-blif and write-pla. The circuit parameters after synthesis using algebraic transformations are indicated as number of gates and stuck faults in Table IV. As these circuits require large amounts of CPU time and memory, we ran GDSEQ on an IBM RS 6000/580 workstation for these circuits. The time taken for GDSEQ and the number of vectors generated are shown in Table IV. Table IV also shows the stuck fault coverage with functional vectors on the synthesized circuit, which again is higher than the coverage of functional faults. Our results demonstrate the capability of GDSEQ to obtain high stuck fault coverages even for very large FSM's. For circuits that are larger than the ones shown in Table IV, in terms of number of product terms, it is better to handle them as an interconnection of two-level PM's, rather than a single PM.

VI. FUNCTIONAL TEST GENERATION FOR GENERAL COMBINATIONAL AND SEQUENTIAL CIRCUITS

To study the efficiency of the Functional vectors on general circuits that are not synthesized using testability preserving transformations we experimented with the ISCAS'89 benchmark circuits that were already available in the multilevel form. The characteristics of these circuits are shown in Table V. As the two-level functional description for these circuits is not available we extracted the ON set cubes of all PO and next state functions. The Sun SPARCstation 2 run times for cube extraction using a Podem-based technique [29] and single output minimization using Espresso [15] are given in the last two columns of Table V.

We employed GDCOMB to derive tests from the PM description of the combinational portion of 18 of the ISCAS'89 benchmarks. These

TABLE VIII CHARACTERISTICS OF RESYNTHESIZED ISCAS'89 CIRCUITS

	No. of	No. of	No. of	Extract	ed PLA	Multile	vel Circuit
FSM	In-	Out-	Flip-	Prod.	G-D	No. of	Stuck
	puts	puts	Flops	Terms	Flts.	Gates	Flts.
s27	4	1	3	15	47	13	37
s208	11	2	8	30	202	69	204
s298	3	6	14	68	309	87	281
s344	9	11	15	249	1495	131	402
s349	9	11	15	249	1495	131	402
s382	3	6	21	167	1080	135	411
s386	7	7	6	51	407	91	291
s400	3	6	21	167	1080	135	411
s444	3	6	21	167	1080	136	406
s510	19	7	6	109	580	190	494
s526	3	6	21	142	740	142	448
s526n	3	6	21	142	740	142	448
s820	18	19	5	126	969	202	597
s832	18	19	5	126	969	202	597
s1196	14	14	18	1050	10077	637	1758
s1238	14	14	18	1050	10073	648	1784
s1488	8	19	6	277	1876	444	1210
s1494	8	19	6	277	1876	444	1210

results are given in Table VI. GDCOMB derived vectors to cover all G and D faults. These vectors were then used to simulate all *collapsed* single stuck faults in the available multilevel form of the circuit. The coverage of single stuck faults by the Functional vectors is less than 100% in many cases as expected, as these multilevel forms were not synthesized using testability preserving transformations. Table VI also gives the results of test generation for stuck faults in the multilevel circuits by a gate-level test generator, Gentest [17]. The run times of GDCOMB are better for all circuits except s344, s349, s1196, and s1238. The number of G and D faults for these circuits is four to nine times larger than that of the stuck faults, but in all cases the average time per fault taken by GDCOMB is much smaller than that of Gentest.

		Perso	onality M	atrix	- N	Multi-leve	l Implem	entatio	n		
Circuit	PI, PO,	GDCOMB						entest			
Name	Prod. Terms	G-D	No. of	CPU	Stuck	Faults	No. of	cov	CPU		
		Faults	Vect.	Sec.	Total	cov %	Vect.	%	Sec.		
s27	7, 4, 15	47	13	0.01	37	100	8	100	0.06		
s208	19, 10, 30	202	95	0.06	204	100	47	100	0.72		
s298	17, 20, 68	309	92	0.06	281	100	40	100	0.72		
s344	24, 26, 249	1495	339	1.01	402	100	43	100	1.07		
s349	24, 26, 249	1495	339	1.01	402	100	43	100	1.07		
s382	24, 27, 167	1080	214	0.47	411	100	56	100	1.25		
s386	13, 13, 51	407	143	0.06	291	100	44	100	0.95		
s400	24, 27, 167	1080	214	0.41	411	100	56	100	1.25		
s444	24, 27, 167	1080	219	0.47	406	100	54	100	1.25		
s510	25, 13, 109	580	120	0.17	494	100	81	100	2.56		
s526	24, 27, 142	740	216	0.35	448	100	50	100	1.25		
s526n	24, 27, 142	740	216	0.35	448	100	50	100	1.49		
s820	23, 24, 126	969	301	0.47	597	100	99	100	3.28		
8832	23, 24, 126	969	308	0.47	597	100	97	100	3.22		
s1196	32, 32, 1050	10077	2148	23.26	1758	100	325	100	26.90		
s1238	32, 32, 1050	10073	2170	22.84	1784	100	358	100	29.35		
s1488	14, 25, 277	1876	306	0.59	1210	100	125	100	9.13		
s1494	14, 25, 277	1876	306	0.59	1210	100	123	100	9.00		

TABLE IX TEST GENERATION FOR COMB. PART OF RESYNTHESIZED ISCAS'89 CIRCUITS (SUN SPARCSTATION 2)

TABLE X FUNCTIONAL TEST GENERATION ON RESYNTHESIZED FSM'S (SUN SPARCSTATION 2)

FSM			GD	EQ				Gentest	
	No. of	G-D	TGen.	FSim.	Useful	Stuck	No. of	Stuck	CPU
	Vect.	Flt.	CPU	CPU	Vect.	Fault	Vect.	Fault	Sec.
		Cov %	Sec.	Sec.		cov %		Cov %	
s27*	23	89.4	0.02	0.03	23	100.0	21	100.0	0.1
s208	260	44.6	0.56	1.37	232	67.1	129	67.7	6.4
s298	187	72.8	2.84	0.59	159	93.6	185	94.3	316.0
s344	129	26.1	700.50	0.89	129	84.6	199	95.3	3979.0
s349	122	26.9	690.90	0.77	122	86.3	117	91.3	1021.0
s382	907	39.8	448.60	5.75	678	88.8	3566	94.2	3890.0
s386*	317	64.1	1.66	0.89	288	93.8	168	93.8	1318.0
s400	572	35.2	343.77	4.47	543	85.1	3415	94.2	2759.0
s444	387	30.6	81.14	4.05	301	78.1	2516	92.4	4763.0
s510*	550	99.1	35.74	2.03	498	99.8	52	88.5	2596.0
s526*	330	44.9	306.44	3.70	330	83.0	3	11.6	955.0
s526n*	312	41.5	312.50	3.63	174	80.6	3	11.6	955.0
s820*	1345	88.6	145.80	5.84	1245	97.5	321	94.6	7213.0
s832*	1527	88.4	96.05	6.26	1500	97.5	320	94.6	7135.0
s1196*	4512	99.6	416.35	60.00	4419	99.4	232	84.0	65.0
s1238*	4722	99.6	451.96	95.00	4630	99.5	254	82.9	66.0
s1488*	1502	83.5	116.49	18.40	1407	97.9	459	95.0	6821.0
s1494*	1529	85.1	115.04	18.30	1529	98.3	565	94.9	11784.0

We employed GDSEQ to derive tests for the two-level FSM's constructed out of the extracted PM description of the combinational portion of these circuits. The results obtained from GDSEQ are given in Table VII. The table also shows the time taken to simulate faults with functional vectors applied to the original multilevel gate implementations of the ISCAS'89 circuits and the stuck fault coverage obtained. The last three columns show the number of vectors, time for test generation, and fault coverage with the gate-level test generator, Gentest. In all cases, the stuck fault coverage is higher than that of the G and D fault coverage, indicating the superior

quality of functional vectors. Comparing functional vector coverages with Gentest coverages of detected stuck faults, 11 out of 18 circuits have a better or equal coverage with functional vectors. This is due to the fact that some faults aborted by Gentest wcrc detected by GDSEQ vectors. Of these 11 circuits, 9 also took less CPU time than Gentest. These 11 circuits are shown with an asterisk in Table VII. The CPU times for the functional approach and that of Gentest cannot be compared meaningfully for all ISCAS'89 circuits since the coverages are different. However, considering the 11 circuits where the GDSEQ vectors gave equal or greater coverage than Gentest, the functional approach (test generation + simulation) is up to 781.8 (for s386) times faster compared to Gentest, with an average speedup of 51.07. The CPU times for s1238 is higher than Gentest, as the number of G and D faults is more than seven times the number of stuck faults in the original circuit. The functional vector coverage of stuck faults is marginally lower (at most 2.7%) than Gentest coverages for five circuits, s208, s298, s444, s526, and s1196. The remaining two circuits (s344 and s526n) have lower fault coverage for GDSEQ than that obtained by Gentest. For these circuits, GDSEQ can be run with different values of time limit and frame limit per fault to generate more functional vectors for an improvement in multilevel stuck fault coverage.

6.1 Experimental Results on Resynthesized Circuits

To study the effect of resynthesis and testability on general circuits we resynthesized the ISCAS'89 circuits using testability preserving transformations. These circuits were synthesized from the extracted PM's as discussed above. The speed of these circuits can be improved by retiming transformations [21], that also preserve testability. The characteristics of the resynthesized ISCAS'89 circuits are shown in Table VIII. Most of these resynthesized circuits are smaller in terms of the number of gates and the number of stuck faults.

We simulated the functional vectors generated earlier by GD-COMB, on synthesized combinational portion for stuck faults. Table IX gives the results of simulation and test generation with Gentest for the synthesized combinational portions. As expected, the functional vectors gave 100% stuck fault coverage on the synthesized circuit, and the efficiency of GDCOMB is better than Gentest in terms of CPU time and time taken per fault, also.

We also simulated the functional vectors generated earlier by GDSEQ with functional vectors on the FSM's constructed out of the resynthesized combinational portions. Table X gives results of simulation and test generation with Gentest for the synthesized FSM's. Comparing functional vector coverages with Gentest coverages of the stuck faults, 11 out of 18 circuits have a better or equal coverage with functional vectors. Out of these 11 circuits, 9 are better in CPU times than Gentest for test generation. These 11 circuits are marked with an asterisk in Table X. The CPU times for s1196 and s1238 are higher as the number of G and D faults is more than five times larger than stuck faults. Considering the 11 circuits where the GDSEQ vectors gave equal or greater coverage than Gentest, the functional approach (test generation + simulation) is up to 516.86 (for s386) times faster compared to Gentest. The functional vector coverage of stuck faults is marginally (less than 0.7%) lower than gentest coverages for s208 and s298. Comparing the fault coverages in Table VII and X, resynthesis has improved the stuck fault coverage of functional vectors for 13 circuits and the Gentest coverages has also improved for 11 circuits. After resynthesis the functional vector coverage of stuck faults has reduced for four circuits, and Gentest coverages has reduced for six circuits.

VII. CONCLUSION

The model of growth and disappearance faults in the logic function of a FSM allows efficient test generation. We found that the functional test sequence derived by a prototype implementation of our test generation algorithm could achieve a very high coverage of single stuck faults in actual multilevel FSM implementations. We must, however, emphasize the usefulness of testability preserving transformations in synthesis as evidenced by our experiment with the circuit *styr*. The functional fault model also allows us to generate tests that are independent of the specific logic implementation. **A** major advantage of this approach is that functional test generation combined with fault simulation is considerably faster than gate-level algorithms that target stuck faults in a specific implementation. For the relatively few stuck faults that may not be detected by the functional test sequence, it is possible to generate additional tests using any gate-level sequential circuit test generator.

The proposed method is ideally suited for an automatic synthesis environment where the functional description of the combinational logic is available. Functional tests can be easily generated during synthesis, as the circuit description for test generation is the same as that for synthesis by MIS or SIS. Our results also show limitations of the method in dealing with very large circuits like some ISCAS'89 benchmarks where only gate-level description is given. The complexity of a single PM for the entire circuit can be prohibitive. A better approach would be to partition the circuit and solve the test generation problem for an interconnection of functional blocks, each described as a PM. We are currently exploring this approach. Another limitation arises when the given multilevel circuit is not prime and irredundant with respect to each output. The results, however, show that the loss of coverage is small. Test generation for those few undetected faults can be obtained by a gate-level test generator.

REFERENCES

- J. E. Smith, 'Detection of Faults in Programmable Logic Arrays," ZEEE Trans. Comput. vol. C-28, pp. 845–853, Nov. 1979. Also see "Author's reply," ZEEE Trans. Comput., vol. C-35, p. 931, Oct. 1986.
- [2] J. Jacob and V. D. Agrawal, "Functional test generation for sequential circuits," in *Proc. 5th Int. Conf. VLSZ Design*, Bangalore, Jan. 1992, pp 17-24.
- [3] G. D. Hachtel, R. Jacoby, K. Keutaer, and C. Morrison, "On properties of algebraic transformations and the multifault testability of multilevel logic," in *Proc. Int. Conf. Computer-Aided Design*, 1989, pp. 422–425.
- [4] J. Rajski and J. Vasudevamurthy, "Testability preserving transformations in multi-level logic synthesis," in *Proc. Int. Test Conf.*, 1990, pp. 265–273.
- [5] V. D. Agrawal and S. C. Seth, *Test Generationfor VLSI Chips*. Los Alamitos, CA: IEEE Computer Society, 1988.
- [6] M. K. Srinivas, J. Jacob, and V. D. Agrawal, "Finite state machine testing based on growth and disappearance faults," in *Proc. 22nd Int. Symp. Fault Tolerant Comput.*, July 1992, pp. 238–245.
- [7] S. M. Reddy, "Complete test sets for logic functions," *IEEE Trans. Comput.*, vol. C-22, pp. 1016–1020, Nov. 1973.
- [8] S. B. Akers, "Universal test sets for logic networks," *IEEE Trans. Comput.*, vol. C-22, pp. 835–839, Sept. 1973.
- [9] K-T. Cheng and J. Y. Jou, "Functional test generation for finite state machines," in *Proc. Int. Test Conf.*, 1990, pp 162–168.
- [10] I. Pomeranz and S. M. Reddy, "On achieving a complete fault coverage for sequential machines using the transition fault model," in *Proc. Design Automation Conf.*, 1991, pp 341–346.
- [11] A. Ghosh, S. Devadas, and A. R. Newton, "Test generation and verification for highly sequential circuits," *IEEE Trans. Computer-Aided Design.* vol. 10, pp. 652–667, May 1991.
- [12] H. Cho, G. D. Hachtel, and F. Somenzi, "Redundancy identification/removal and test generation for sequential circuits using implicit state enumeration," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 935–945. July 1993.
- [13] F. J. Hill and B. Huey, "SCIRTSS: A search system for sequential circuit test sequences," *IEEE Trans. Comput.*, vol. C-26, pp. 490–502, May 1977.
- [14] H. K. T. Ma, S. Devadas, A. R. Newton, and A. Sangiovanni Vincentelli, "Test generation for sequential circuits," *ZEEE Trans. Computer-Aided Design*, vol. 7, pp. 1081–1093, Oct. 1988.
- [15] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithmsfor VLSI synthesis*. Boston, MA: Kluwer Academic, 1984.
- [16] I. Kohavi and Z. Kohavi, "Detection of multiple faults in combinational logic networks," ZEEE Trans. Comput., vol. C-21, pp. 556–568, June 1972.
- [17] R. Bencivenga, T. J. Chakraborty, and S. Davidson, "GENTEST: The architecture of sequential circuit test generator," in *Proc. Custom Integrated Circuits Conf.*, 1991, pp. 17.1.1–17.1.14.

[18] D. L. Dietmeyer, Logic Design of Digital Systems. Boston, MA: Allyn and Bacon, 1971. using two-level representations," in *Proc. 26th Design Automation Conf.*, 1989, pp. 722–725.

- [19] J. Jacob and N. N. Biswas, "PLATES: An efficient PLA test pattern generator," in *Proc. Third Int. Workshop VLSI Design*, Bangalore, Jan. 1990, pp. 147–154.
- [20] J. Jacob and V. D. Agrawal, "Multiple Fault Detection in Two-Level Multi-output Circuits," J. Electron. Test.: Theory Appl., vol. 3, pp. 171–173, May 1992.
- [21] A. El-Maleh, T. Marchok, J. Rajski, and W. Maly, "On test set preservation of retimed circuits," in *Proc. 32nd Design Automation Con*\$, June 1995, pp. 176–182.
- [22] M. Robinson and J. Rajski, "An algorithmic branch and bound method for PLA test pattern generation," in *Proc. Int. Test Con*\$, Sept. 1988, pp. 784–795.
- [23] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang, "MIS: A multiple level logic optimization system," *ZEEE Trans. Computer-Aided Design*, vol. 6, pp. 1062–1081, Nov. 1987.
- [24] U. Dave and J. H. Patel, "A functional-level test generation methodology

- [25] S. Devadas, H. K. Ma, and A. R. Newton, "MUSTANG: State assignment of finite state machines targeting multilevel logic implementations," *IEEE Trans. Computer-Aided Design*, vol. 7, pp 1290–1300, Dec. 1988.
- [26] G. D. Hachtel, J. K. Rho, F. Somenzi, and R. Jacoby, "Exact and heuristic algorithms for the minimization of incompletely specified state machines," in *Proc. Euro. Conf. Design Automation*, Amsterdam, The Netherlands, Feb. 1991, pp. 184–191.
- [27] W. T. Cheng and M. L. Yu, "Differential Fault Simulation for Sequential Circuits," J. Electron. Test.: Theory Appl., vol. 1, pp. 7–13, Feb. 1990.
- [28] X. Chen and M. L. Bushnell, Efficient Branch and Bound Search with Application to Computer Aided Design. Boston, MA: Kluwer Academic, 1996.
- [29] H. K. T. Ma and S. Devadas, "Logic verification algorithms and their parallel implementation," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 181–188, Feb. 1989.