

Combinational Test Generation for Acyclic Sequential Circuits using a Balanced ATPG Model*

Yong Chang Kim[†]

University of Wisconsin-Madison
Madison, WI 53706 USA
kimy@ece.wisc.edu

Vishwani D. Agrawal

Bell Labs, Lucent Tech.
Murray Hill, NJ 07974 USA
va@research.bell-labs.com

Kewal K. Saluja

University of Wisconsin-Madison
Madison, WI 53706 USA
saluja@engr.wisc.edu

Abstract

To create a combinational ATPG model for an acyclic sequential circuit, all unbalanced fanouts, i.e., fanouts reconverging with different sequential depths, are moved toward primary inputs using a retiming-like transformation. All flip-flops are then shorted and unbalanced primary input fanouts are split as additional primary inputs. A combinational test vector for a fault in this model is converted into a vector sequence that detects the corresponding fault in the original circuit. An analysis classifies the undetected faults in this model as either untestable or multiply-testable. The latter, typically less than 5% of all faults, are modeled as special single faults in the combinational model. This procedure correctly treats various types of faults, namely, (a) faults detectable by repeating a pattern, (b) faults only detectable by non-repeated patterns, (c) faults only testable as multiple faults in the combinational model, and (d) sequentially untestable faults. IS-CAS '89 benchmark results verify that the given procedure achieves identical fault coverage and efficiency as a sequential ATPG and uses less CPU time.

1 Introduction

Two commonly used design for testability (DFT) methods are full scan [11] and partial scan [2]. In full scan, we scan all flip-flops (FFs) and use combinational automatic test pattern generation (ATPG). In partial scan, a subset of FFs is scanned. Thus, hardware overhead and testing time are reduced over full scan. However, one must use a sequential ATPG program whose complexity is significantly greater than that of combinational ATPG used in the full scan design.

Cheng and Agrawal [3, 8] and Kunzmann and Wunderlich [17] have proposed partial scan methods that break cycles to make the sequential structure acyclic in the test mode. An acyclic circuit has a finite *maximum sequential depth* (d), defined as the largest number of FFs on any path between

primary inputs and outputs. Thus, using the time-frame expansion procedure [1], any fault can be tested by combinational ATPG applied to a circuit obtained by cascading the combinational logic d times. Miczo [19] proposes the conversion of an acyclic circuit to a combinational model by copying the entire fanin cone of each FF. Min and Rogers [20] use a combinational ATPG model in which all FFs are shorted. They repeat the same vector ($d + 1$) times, but any undetected faults require sequential ATPG to achieve 100% fault efficiency. Kunzmann and Wunderlich [17, 22] propose a combinational ATPG model capable of generating both repeated and non-repeated vectors.

Besides being acyclic, a sequential circuit can have other structural properties that facilitate ATPG. A circuit is called a *balanced* circuit [15] if it is acyclic and all paths between any pair of nodes have the same number of FFs, where a node can be a primary input (PI), gate, FF or primary output (PO). The number of FFs on a path is referred to as the *sequential depth* of that path. Gupta et al. [15] propose a partial scan technique in which scan and other hardware are added to make the circuit balanced and testable via combinational ATPG.

Balakrishnan and Chakradhar [6] define *strongly balanced* circuits, a subclass of balanced circuits, in which all paths between PI and PO have the same sequential depth. They use combinational ATPG to derive tests that are significantly more compact than the partial scan sequences of the previous method [15]. Balakrishnan and Chakradhar [5] also propose a software transformation procedure to reduce (but not completely eliminate) the number of FFs in the ATPG model. That method, however, requires sequential ATPG since the ATPG model may contain FFs.

According to Fujiwara et al. [12, 13, 14, 16, 21] a circuit that appears balanced when we ignore the PI nodes is called an *internally balanced* circuit. They use combinational ATPG by temporarily splitting the PI fanouts causing unbalance into separate PIs. Test sequences are then constructed for the internally balanced circuit, which requires less DFT overhead than a balanced circuit.

The present work is closely related to the proposal of Kunzmann and Wunderlich [17, 22]. We give algorithms to produce the smallest combinational ATPG model for any general

*This work was supported in part by the National Science Foundation grant MIP9714034.

[†]Summer Intern (1998 and 1999) at Bell Labs, Lucent Technologies, Murray Hill, NJ 07974.

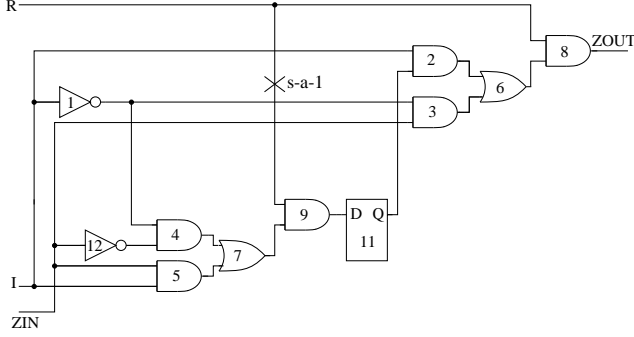


Figure 1. A single output circuit.

acyclic circuit (which does not have to be balanced), requiring no hardware modification in the original circuit. Because of splitting of signals, certain faults of the original circuit map onto multiple faults. Only in cases, where such multiple faults mask each other, we generate multiple fault tests. Our test generation system correctly deals with such situations and obtains a 100% fault efficiency. We assume that the given sequential circuit is synchronous and acyclic. The technique is also applicable to other circuits via partial scan [8] or resynthesis [9].

2 Test Generation Approach

Our test generation method has three steps, combinational ATPG model generation, combinational test generation and test transformation.

2.1 Combinational ATPG model generation

Gupta et al. [15] have defined a *balanced* sequential structure.

Definition 1 A synchronous sequential circuit is said to be balanced if it is acyclic and for any pair of signal nodes (PI, FF, gates and PO) v_1 and v_2 in the circuit all directed paths (if any) from v_1 to v_2 are of equal sequential weight (depth). The weight or depth of a path is the number of FFs on it.

Consider the circuit in Figure 1. It is acyclic but not balanced. Notice that two paths between nodes R and $ZOUT$ (AND gate 8) have sequential depths of 0 and 1, respectively. Thus, PI R can supply different values to the inputs of gate 8 via the two paths. In order to allow combinational ATPG to assign multiple values to a signal like R , we generate a *balanced combinational circuit* (BCC) model.

Algorithm 1 BCC Generation

- **Levelization:** In a single pass, starting from PIs (PIs have 0 weight), assign weights to all PO nodes, where a PO weight is the maximum number of FFs on any path between the PO and all reachable PIs. The weight of a node g is written as $w(g)$. Initialize all non-PO nodes to unassigned weights.

- **Balancing:** A node is balanced if all of its fanout nodes with assigned weights have the same weight. Recursively, starting at POs, determine weights for all reachable node until every nodes in the circuit is balanced and there is no node with unassigned weight. Two cases occur as non-PO nodes g_i are repeatedly processed:

Case 1 Node g_i either has a single fanout node g_j with assigned weight $w(g_j)$, or has multiple fanout nodes among which all nodes with assigned weight have the same value $w(g_j)$. Then, $w(g_i) = w(g_j)$, if g_i is a combinational gate or PI, or $w(g_i) = w(g_j) - 1$, if g_i is a FF.

Case 2 Node g_i has fanouts to multiple nodes of which n , g_{jq} , $1 \leq q \leq n$, have assigned weights, $w(g_{jq})$. Suppose there are m , $m \leq n$, distinctly different values among these n weights. Then, node g_i is duplicated as m nodes, g_{i1} through g_{im} , each of the same type (i.e., PI, gate or FF) as g_i . If g_i is a PI, then duplication creates $m - 1$ new PIs. Otherwise, inputs to the duplicated nodes are supplied by adding $m - 1$ new fanouts to each fanin node of g_i . A duplicated node g_{ip} , $1 \leq p \leq m$, fans out to all those nodes among g_{jq} , $1 \leq q \leq n$, that have the same weight. This transformation is known as *duplicate and split* (DAS). Next, the weights of the duplicated nodes are determined by Case 1.

- **Combinationalization:** Short all FFs or replace FFs with buffers to get the BCC.

Algorithm 1 assigns a weight to every node reachable from one or more POs. The recursion over POs leaves each node with one unique weight. Case 1 determines the weight of a balanced node. Case 2 deals with unbalanced nodes. The *duplicate and split* (DAS) procedure basically moves unbalanced fanouts one level backward (toward PIs). It leaves the function of the circuit unchanged if a PI and its split copies assume the same signal value. Successive application of DAS eventually moves all unbalances to PIs, whose splitting creates a perfectly balanced structure.

The circuit in Figure 2 (a) illustrates the DAS transformation. The figure shows a portion of a circuit that is to be balanced. The weight of node g_k is $w(g_k)$. Using Case 1 of Algorithm 1, the weight of node g_f is found to be $w(g_k) - 1$. The node g_j has two fanouts that reconverge at g_k . These fanouts are unbalanced since the fanout nodes, g_k and g_f , have different weights, $w(g_k)$ and $w(g_k) - 1$, respectively. Applying Case 2 of Algorithm 1, we duplicate g_j as g_{j1} and g_{j2} , and split the fanouts so g_{j1} and g_{j2} have one fanout each to g_k and g_f , respectively. We now apply Case 1 of Algorithm 1, to assign weights, $w(g_k)$ to g_{j2} and $w(g_k) - 1$ to g_{j1} . Notice, nodes g_h and g_i are now unbalanced and we will need

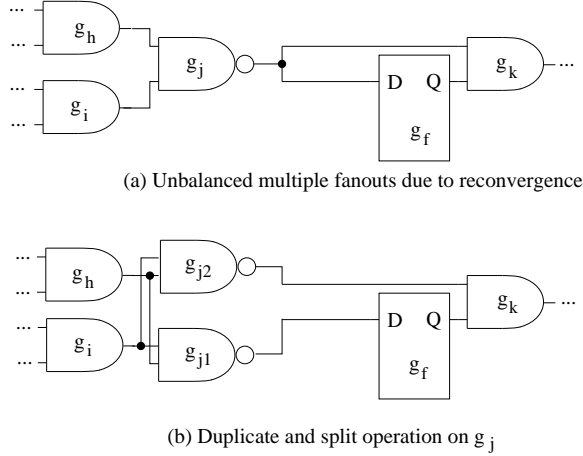


Figure 2. A duplicate and split (DAS) example.

to apply DAS to them. Figure 2 (b) shows the DAS transformation on g_j , which moved unbalanced fanouts from g_j to the fanin nodes, g_h and g_i . The use of DAS provides separate signal paths in the combinational model so that the copies of a signal may assume different values in different time frames. The recursive DAS transformation may create a circuit with new PIs having specific weights. When DAS is applied to all gates recursively, all unbalancedness is moved to PIs.

No duplication or splitting occurs in a single-output balanced circuit. In a multiple-output balanced circuit, some PIs having different weights with respect to different POs do split. Although no PI supplies more than one value to detect a fault, the splitting provides the proper sequencing of the values of various PIs in the sequential test (Subsection 2.3).

This circuit-transformation is similar to retiming [18]. However, retiming proposed by Leiserson and Saxe [18] cannot always be performed directly to the multiple fanout point. Using DAS, we can apply the retiming to move all unbalanced fanouts to PIs.

As we see from Figure 2 (b), the DAS operation on non-PI vertices will not alter the functional behavior of the circuit. Thus, the resultant circuit is functionally equivalent to the original one after recursive DAS transformations are performed only on internal fanouts and PI vertices are not modified. However, a single fault before the modification may map to multiple locations if it was duplicated. In a sequential circuit, an unbalanced fanout node may supply different values to branches during different time frames. To model such unbalanced fanout nodes, we modify the circuit by adding PIs that represent input values for different time frames. The effect of a single fault being repeated in multiple time frames is modeled as multiple faults in the duplicated logic of the combinational model.

Figure 3 shows the BCC of the circuit in Figure 1, where the FF is replaced with a buffer. As is evident from the figure, R , I , ZIN and gate 1 have unbalanced fanouts. These

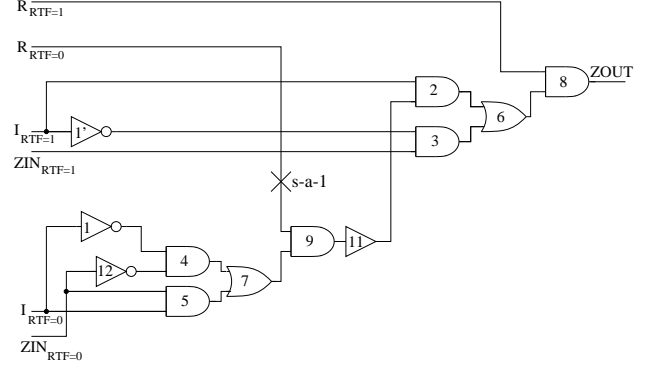


Figure 3. BCC for the circuit in Figure 1.

PIs and the gate with unbalanced fanouts must be balanced using Algorithm 1. The multiple fanouts of node 1 are unbalanced since they reconverge at node 6 with different sequential depths. Thus, 1 is duplicated and split by Algorithm 1.

For obtaining the results of Section 3, the algorithm of this subsection was implemented by representing the circuit as a *weighted* directed acyclic graph (DAG). The vertices of this DAG are PIs, gates and POs. An arc between a vertex-pair represents a signal flow path and the integer weight of the arc equals the number of FFs on the path.

2.2 Combinational test generation

A single fault in the original circuit maps onto a set of multiple faults or a single fault in the BCC depending upon whether or not the target fault site has been duplicated and split. The fault mapping algorithm is given below:

Algorithm 2 Fault Mapping

- **Single fault mapping:** A fault in the original circuit maps onto a single fault in BCC if there is no duplicated node at the corresponding fault site in the BCC.
- **Multiple fault mapping:** A single fault of the original circuit maps onto a set of multiple faults if DAS transformation has created more than one copy of the fault site in the BCC. All copied lines and the original line are then the sites of simultaneously occurring single faults in the BCC.

If there is a test for a *mapped* single fault in the BCC, then the fault will be detected in the sequential circuit by the sequentialized BCC test (see Subsection 2.3). Otherwise, the fault is untestable. Similarly, if there is a test for a *mapped* multiple fault in the BCC, then fault is detectable in the sequential circuit. Otherwise, the fault is untestable.

The multiple fault mapping is a sufficient condition to detect the “original” fault, but it is not a necessary condition. If we generate a test using BCC for single faults on each multiple fault site, then some of those tests may detect the fault in

the sequential circuit. However, such single fault assumption for a multiple fault does not guarantee the detection and the multiple fault model may be necessary for some faults. As we observe in Section 3, only a small fraction of faults needs to be modeled as multiple faults in the BCC. Thus, combinational ATPG can provide a 100% fault efficiency with proper fault modeling.

Sometimes the splitting of a signal in BCC is not due to unbalance but is there because the signal feeds multiple POs with different sequential depths (Subsection 2.1). A test for a single fault on any of these split signals will detect the corresponding fault in the sequential circuit and no multiple-fault mapping is required in such cases.

2.3 Test vector to sequence transformation

Let d_{max} denote the maximum sequential depth of the original sequential circuit. Weights assigned to PIs in the BCC are called *relative time frames* (RTFs). RTFs determine the time sequence of signal values at the PI. For example, *ZIN* of the circuit shown in Figure 1 has unbalanced fanout branches and the corresponding BCC in Figure 3 has *RTFs* of 0 and 1. Using the RTF information, we can determine how to convert a combinational test vector of BCC to a sequential test. Each combinational vector produces a sequence of vectors of length up to $d_{max} + 1$. In general, the vector sequence length can be less than or equal to $d_{max} + 1$, since we only need to propagate the effect of the fault to an observable PO. We transform the combinational test vector generated for BCC to a vector sequence for the original circuit as follows:

Algorithm 3 Test-Transformation: *Suppose that the BCC has $k + 1$ copies of a PI node x with RTFs of $\{t_0, t_1, \dots, t_k\}$, $k \leq d_{max}$, sorted in ascending order, and the values on these $k + 1$ copies in a combinational test derived for BCC are $\{v_0, v_1, \dots, v_k\}$. The value of the PI x in the t_i th vector of the sequence is v_i . If some value t_j , $0 \leq j \leq k$, does not occur among the RTFs of x , then a random value (0 or 1) is assigned to the PI x in the t_j th vector.*

Consider a stuck-at-1 fault on input R of gate 9 as shown in Figure 1. The corresponding fault in the BCC is shown in Figure 3. Using BCC, the test is found to be $\{R_{RTF=0}, I_{RTF=0}, ZIN_{RTF=0}\} = \{011\}$ and $\{R_{RTF=1}, I_{RTF=1}, ZIN_{RTF=1}\} = \{111\}$. Since the maximum depth of the circuit is 1, each combinational vector derived for BCC will be sequentialized to two vectors for time frames 0 and 1, respectively. For time frame 0, we take the test input of $\{R_{RTF=0}, I_{RTF=0}, ZIN_{RTF=0}\}$ and apply vector $\{011\}$ to $\{R, I, ZIN\}$. Then for the time frame 1, we take $\{R_{RTF=1}, I_{RTF=1}, ZIN_{RTF=1}\}$ and apply vector $\{111\}$ to $\{R, I, ZIN\}$. Since copies of $\{R, I, ZIN\}$ exist for both $RTF = 0$ and $RTF = 1$, sequential vector is fully specified and there is no need to fill random values. Thus, the

generated sequential test input for $\{R, I, ZIN\}$ is a two vector sequence, $\{011\}$ and $\{111\}$. Using the *GENTEST* ATPG system [10], a fault coverage of 100% was obtained for both original circuit and BCC shown in Figures 1 and 3.

3 Results

We have implemented the algorithm presented in this paper in a C language program called *BCC*. Tables 1 and 2 show the results of experiments on the ISCAS 89 benchmark circuits. For the cyclic ISCAS 89 benchmark circuits, we scanned minimal sets of FFs to make the kernel circuit acyclic. Using improved algorithms, in most cases we scanned fewer FFs than reported by Min and Rogers [20]. We only present the results of applicable circuits, i.e., acyclic circuits with at least one unscanned FF. For test generation, we used the *GENTEST* ATPG system [10] for both combinational and sequential circuits, running on a Sun Ultra II workstation with dual 200 MHz processors.

We first generate combinational tests for all single stuck-at faults of BCC. Then we use a sequential fault simulator to simulate the sequentialized vector set. Each fault, not detected by this vector set, is processed and classified as: (1) an untestable single-mapped fault, (2) an undetected multiple-mapped fault, or (3) a detected multiple-mapped fault. Only the second type of faults are modeled as multiple faults in the BCC for additional tests or untestability classification.

Table 1 gives a comparison between the combinational ATPG using the BCC and the conventional sequential ATPG. Fault Coverage (FC), fault efficiency (FE), test vector length (Vectors), total test generation time (TGT) and averaged test generation time per fault (TPF) for both combinational and sequential ATPG are included in the table, under columns *FC*, *FE*, *Vectors*, *TGT* and *TPF*, respectively. Last two columns show the speed-up ratios of combinational over sequential approaches for TGT and TPF. As expected, the simpler combinational model ATPG is significantly faster than the sequential ATPG. We observe as much as 5.9 times speed-up.

It is important to notice that the results are generated using a default time limit set by *GENTEST* ATPG. Same time limit was used for both combinational and sequential test generation. For circuits with aborted faults, it is possible to increase the time limit per fault to make the differences in the test generation times more dramatic. In *all* cases, the new method yielded equal or better fault coverages (FC) and fault efficiencies (FE) as shown in Table 1. The combinational approach gave a higher fault efficiency for s9234, as two aborted faults were detected. Higher fault coverage (one more fault detected) was obtained for s38584 in less time than the sequential ATPG. However, test vector lengths of BCC are much longer than sequential method.

Above results are reported without any vector compaction, but there are several ways to compact the combinational-derived test vectors. One can compact the BCC vectors with

Table 1. Test generation results (Sun Ultra II, 200MHz Dual CPUs): combinational vs. sequential ATPG.

Circuit Name	Combinational ATPG					Sequential ATPG					ATPG speed-up	
	FC	FE	Vectors	TGT	TPF	FC	FE	Vectors	TGT	TPF	TGT	TPF
	%	%		s	ms	%	%		s	ms		
s382	100.00	100.00	116	0.10	1.72	100.00	100.00	81	0.18	3.08	2.0	1.9
s400	98.50	100.00	118	0.10	1.46	98.50	100.00	83	0.13	1.73	1.4	1.3
s444	97.52	100.00	112	0.14	1.63	97.52	100.00	77	0.15	1.79	1.1	1.2
s641	100.00	100.00	196	0.22	2.20	100.00	100.00	112	0.34	3.43	1.6	1.6
s713	94.88	100.00	216	0.94	5.17	94.88	100.00	118	1.02	5.76	1.2	1.2
s953	100.00	100.00	214	0.37	3.42	100.00	100.00	182	0.49	4.04	1.0	0.9
s1196	99.87	100.00	1456	1.10	2.69	99.87	100.00	304	1.33	6.47	1.2	2.4
s1238	96.64	100.00	1532	2.15	3.86	96.64	100.00	327	2.83	9.63	1.5	2.8
s1423	99.08	100.00	182	1.43	7.02	99.08	100.00	182	2.17	11.42	1.7	1.8
s5378	93.69	99.71	32000	293.22	108.88	93.69	99.71	1117	1268.00	1640.36	3.9	13.7
s9234	93.16	99.95	5460	410.21	216.01	93.16	99.94	1233	425.63	337.80	1.2	1.82
s13207	97.13	99.97	74773	417.08	85.94	97.13	99.97	2442	1008.04	739.57	2.6	9.2
s15850	96.65	99.97	183270	146.17	12.62	96.65	99.97	2507	853.49	535.78	5.9	42.6
s35932	89.80	100.00	258370	444.92	60.27	89.80	100.00	2377	569.07	122.01	1.5	2.3
s38417	99.25	99.54	116550	390.49	108.38	99.25	99.54	5360	860.87	358.84	2.2	3.3
s38584	95.66	99.96	405216	5105.90	236.01	95.65	99.95	5763	7293.11	1479.01	1.4	6.3

Table 2. Circuit statistics.

Circuit name	FFs			Max. depth	BCC size(+%)		MF %
	Total	Scan	Scan(%)		PI	gate	
s382	21	15	71.4	1	0.0	0.0	0.0
s400	21	15	71.4	1	0.0	0.0	0.0
s444	21	15	71.4	1	0.0	0.0	0.0
s641	19	15	78.9	1	0.6	2.0	0.0
s713	19	15	78.9	1	0.6	2.0	0.0
s953	29	23	79.3	1	0.0	0.0	0.0
s1196	18	0	0.0	3	52.6	264.3	0.0
s1238	18	0	0.0	3	54.2	264.3	0.7
s1423	74	71	95.9	1	1.7	6.8	0.0
s5378	179	30	16.8	19	226.1	778.5	2.5
s9234	228	152	66.7	4	39.9	114.4	4.2
s13207	669	310	46.3	22	110.7	240.8	1.6
s15850	597	441	73.9	29	243.4	627.5	1.4
s35932	1728	306	17.7	34	254.7	320.4	1.7
s38417	1638	1080	69.9	9	15.3	78.6	1.2
s38584	1425	1115	78.2	35	141.0	376.5	0.3

a simple reverse order simulations and drop the vectors that do not detect new faults. In general, we obtain about 50% reduction this way. After this set is sequentialized, the sequence is further compacted using sequential circuit fault simulation. The final number of compacted vectors can be similar to the sequentially-derived test length. For s5378, a 32000-vector BCC derived sequence reduced to about 16000 vectors when simple reverse-order combinational fault simulation was used. 32000 vectors were reduced to 5384 vectors with sequential fault simulation and to 4236 vectors when both methods were combined. In either case, added CPU time and processing time was less than 10% of sequential TGT.

Table 2 shows statistics for ISCAS 89 benchmark circuits and their BCC. Three columns under *FFs*, namely *Total*, *Scan* and *Scan(%)*, give the total number of FFs, number of scan FFs used to make the circuit acyclic and % of scan FFs, respectively. The column *Max depth* shows the maximum sequential depth of *acyclic* circuits after removing the scan FFs.

The increase in numbers of PIs and gates of BCC over the original circuits are shown in two columns, *PI* and *gates* under the heading *BCC size (+%)*. The increase in PIs varies from 0 to 250% and increase in gates between 0 to 778%. In general, circuits with more FFs and larger maximum sequential depth have a larger increase in BCC size than the circuits with fewer FFs and smaller depth. *The reader should not mistake the increased BCC size for DFT hardware overhead. The BCC size is only an indicator of the size of the combinational ATPG model circuit.*

For some faults, the balanced combinational model has one-to-multiple mapping. These multiple faults correspond to faults at the same site in the original circuit at different time-frames. When the combinational ATPG program is not equipped with multiple fault detection capability, a single fault assumption still produces tests for most faults as we have found. However, we can easily model the multiple faults with a simple modification. A single stuck-at fault on an added signal feeding into two-input OR or AND gates inserted at fault sites represents the multiple stuck-at fault. The test generation time for such multiple-mapped fault is same as any other single fault. The last column in Table 2 shows the % of multiple-mapped faults (MF) that had to be modeled. We found that most (about 95%) of the undetected faults did not require multiple fault mapping.

4 Conclusion

Our target circuit for test generation is a circuit that is either originally acyclic or has been made acyclic by partial scan or any other DFT method. Our proposed test generation method for any general acyclic sequential circuit requires only a combinational ATPG. Our test generation method is based on transforming the unbalanced acyclic sequential circuit to a *fault equivalent* combinational circuit model by moving all unbalanced fanouts to PIs and then adding new PIs.

Added PIs allow combinational model to create non-repeated vectors to detect a fault in the original acyclic circuit. This is significantly different from a previous combinational model [20] where repeated test patterns were used. Our method detects all sequentially detectable faults, and also identifies all sequentially untestable faults to achieve maximum possible fault efficiency faster than the conventional sequential test generation approach. Because a combinational model is used to generate tests, the test generation time spent on detected as well as on untestable faults is significantly lower.

5 Future Studies

Since the realization that feedback in a sequential circuit adds to the test generation complexity, there have been numerous reports [3, 7, 8, 15, 17] on reducing the complexity of test generation by breaking cycles. Scan design is a widely used DFT method for breaking loops and cycles in a sequential circuit. Scan DFT can not only make the circuit loop/cycle-free, but can also make the circuit strongly balanced, weakly balanced or leave it unbalanced. However, there has been no research conducted on nor formal algorithms given for selecting scan FFs to make the circuit balanced beyond the work of Gupta et al. [15]. With a proper algorithm, we believe that more compact and efficient acyclic balanced sequential circuit can be obtained.

Similar to scan DFT, the multiple-clock (MC) can also be used to break loops in the circuits [4]. The use of MC DFT to break cycles may not be as straightforward as scan DFT, but it has its advantages. It can break (or block) loops without scanning in and out. We also believe that MC DFT can be used to assist in creating a balanced circuit. With a proper integration of the scan DFT and MC DFT, test generation of sequential circuits may be even more efficient and less costly.

Acknowledgment — Authors thank Arun Balakrishnan for supplying the MFVS for several ISCAS '89 circuits.

References

- [1] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*. New York: IEEE Press, 1990.
- [2] V. D. Agrawal, editor, *Special Issue on Partial Scan Methods*, volume 7 of *J. Electronic Testing: Theory and Applic.* Boston: Kluwer Academic Publishers, Aug.-Oct. 1995. no. 1/2.
- [3] V. D. Agrawal, K.-T. Cheng, D. D. Johnson, and T. Lin, "Designing Circuits with Partial Scan," *IEEE Design & Test of Comput.*, vol. 6, no. 2, pp. 8–15, Apr. 1988.
- [4] V. D. Agrawal, S. C. Seth, and J. S. Deogun, "Design for Testability and Test Generation with two clocks," in *Proc. 4th Int. Symp. on VLSI Design*, Jan. 1991, pp. 44–51.
- [5] A. Balakrishnan and S. T. Chakradhar, "Software Transformations for Sequential Test Generation," in *The Fourth Asian Test Symp.*, Nov. 1995, pp. 266–272.
- [6] A. Balakrishnan and S. T. Chakradhar, "Sequential Circuits with Combinational Test Generation Complexity," in *Proc. 9th International Conference on VLSI Design*, Jan. 1996, pp. 111–117.
- [7] S. T. Chakradhar, A. Balakrishnan, and V. D. Agrawal, "An Exact Algorithm for Selecting Scan Flip-Flops," *J. Electronic Testing: Theory and Applic.*, vol. 7, no. 2, pp. 71–82, Oct. 1995.
- [8] K.-T. Cheng and V. D. Agrawal, "A Partial Scan Method for Sequential Circuits with Feedback," *IEEE Trans. Computers*, vol. 39, no. 4, pp. 544–548, Apr. 1990.
- [9] K.-T. Cheng and V. D. Agrawal, "State Assignment for Testable Design," *Int. Journal of Comp. Aided VLSI Design*, vol. 3, no. 3, pp. 291–308, 1991.
- [10] W. T. Cheng and T. J. Chakraborty, "GENTEST: An Automatic Test Generation System for Sequential Circuits," *Computer*, vol. 22, no. 4, pp. 43–49, Apr. 1989.
- [11] E. B. Eichelberger, E. Lindbloom, J. A. Waicukauski, and T. W. Williams, *Structured Logic Testing*. Englewood Cliffs, New Jersey: Prentice-Hall, 1991.
- [12] H. Fujiwara, "A New Class of Sequential Circuits with Combinational Test Generation Complexity," *IEEE Trans. on Computers*, vol. 49, no. 9, pp. 895–905, Sep. 2000.
- [13] H. Fujiwara, "A New Definition and A New Class of Sequential Circuits with Combinational Test Generation Complexity," in *Proc. Int. Conf. on VLSI Design*, Jan. 2000, pp. 288–293.
- [14] H. Fujiwara, S. Ohtake, and T. Takasaki, "Sequential Circuit Structure with Sombinational Test Generation Complexity and its Application," *Trans. IEICE (DI) (in Japanese)*, vol. J80-D-1, no. 2, pp. 155–163, Feb. 1997.
- [15] R. Gupta, R. Gupta, and M. A. Breuer, "The BALLAST Methodology for Structured Partial Scan Design," *IEEE Trans. Computers*, vol. 39, no. 4, pp. 538–548, Apr. 1990.
- [16] M. Inoue, E. Gizdarski, and H. Fujiwara, "Theorems for Separable Primary Input Faults in Internally Balanced Structures," *Information Science Technical Report*, vol. ISSN 0919-9527, no. NAIST-IS-TR2000004, pp. 1–5, Mar. 2000.
- [17] A. Kunzmann and H. J. Wunderlich, "An Analytical Approach to the Partial Scan Problem," *J. Electronic Testing: Theory and Applic.*, vol. 1, no. 2, pp. 163–174, Apr. 1990.
- [18] C. E. Leiserson and J. B. Saxe, "Retiming Synchronous Circuitry," *Algorithmica*, vol. 6, no. 1, pp. 5–35, 1991.
- [19] A. Miczo, *Digital Logic Testing and Simulation*. New York: Harper & Row, 1990.
- [20] H. B. Min and W. A. Rogers, "A Test Methodology for Finite State Machines using Partial Scan Design," *J. Electronic Testing: Theory and Applic.*, vol. 3, no. 2, pp. 127–138, May 1992.
- [21] T. Takasaki, T. Inoue, and H. Fujiwara, "Paritial Scan Design Methods Based on Internally Balanced Structure," in *Proc. Asia and South Pacific Design Automation Conf.*, Feb. 1998, pp. 211–216.
- [22] H. J. Wunderlich, "The Design of Random-Testable Sequential Circuits," in *Proc. Fault-Tolerant Comp. Symp.*, 1989, pp. 110–117.