minimum crosstalk, which is the best TDT waveform among the four types of layouts. Pattern IV in Fig. 9 shows similar waves as the Dz line. A higher peak in the TDT waveform than the one from the Dz line is observed, although TDR waveforms are very similar to each other.

The measured peak values are listed in Table III for comparison with the simulations.

VI. Conclusion

The modified version of GA for the design of a delay line is introduced, and the optimal designs are verified by measuring crosstalk with manufactured PCB boards. Because GA requires a large number of function evaluations, a fast and efficient way of crosstalk calculation is proposed by discretizing the line into segments and superposing the crosstalk from each line segment. The methodology of making offsprings and mutation for the discretized line layout chromosome is suggested. The proposed GA has shown the excellence; we obtained the well-known layouts: the spiral and the meander layouts. In addition, a new layout named Dz layout was found for the minimum disturbance area of the unadulterated waveform. The optimum layouts are manufactured, and the measured waveforms were compared with the simulation results.

In calculating crosstalk, we have made simplifying assumptions, although a more elaborate model may be adopted: 1) The crosstalk is generated only at the parallel adjacent line, and 2) the rising time of the step function (T_r) is consistent until it travels to the ending point. The current version of the modified GA has an exponentially increasing time complexity as the PCB board size (C_{width} and C_{height}) increases. To alleviate this problem, a better scheme for the definition of chromosome (currently composed of 3 bits), as well as a fast-converging algorithm, is necessary.

References

[1] R. B. Wu and F. L. Chao, "Flat spiral delay line design with minimum crosstalk penalty," *IEEE Trans. Compon. Packag. Technol.*, vol. 19, no. 2, pp. 397–402, May 1996.

[2] R. B. Wu and F. L. Chao, "Laddering wave in serpentine delay line," *IEEE Trans. Compon. Packag. Technol.*, vol. 18, no. 4, pp. 644–650, Nov. 1995.

[3] B. J. Rubin and B. Singh, "Study of meander line delay in circuit boards," *IEEE Trans. Microw. Theory Tech.*, vol. 48, no. 9, pp. 1452–1460, Sep. 2000.

[4] I. Ndip, W. John, and H. Reichl, "Minimizing reflections and cross-talk in chip packages," in *Proc. 7th Electron. Packag. Technol. Conf.*, 2005, pp. 44–48.

[5] K. S. Jhang, "Layer assignment for crosstalk minimization in a three layers gridded channel routing," in *Proc. 5th Int. Conf. VLSI CAD*, 1997, pp. 403–405.

[6] S. Y. Kulkarni and K. V. V. Murthy, "Multichip module structures for minimizing crosstalk effects in high-speed applications," in *Proc. Int. Conf. Electromagn. Interference Compat.*, 1995, pp. 34–39.

[7] M. A. Elgamel and M. A. Bayoumi, "An efficient minimum area spacing algorithm for noise reduction," in *Proc. Electron., Circuits Syst.*, 2003, pp. 862–865.

[8] T. Y. Ho, Y. W. Chang, S. J. Chen, and D. T. Lee, "Crosstalk- and performance-driven multilevel full-chip routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 6, pp. 869–878, Jun. 2005.

[9] J. Lienig, "A parallel genetic algorithm for performance-driven VLSI routing," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 29–39, Apr. 1997.

[10] W. D. Guo, G. H. Shiue, C. M. Lin, and R. B. Wu, "Comparisons between serpentine and flat spiral delay lines on transient reflection/transmission waveforms and eye diagrams," *IEEE Trans. Microw. Theory Tech.*, vol. 54, no. 4, pp. 1379–1387, Jun. 2006.

[11] O. M. Ramahi, "Analysis of conventional and novel delay lines: A numerical study," *Appl. Comput. Electromagn. Soc. J.*, vol. 18, no. 3, pp. 181–190, Nov. 2003.

[12] K. C. Gupta, R. Garg, I. Bahl, and P. Bhartia, *Microstrip Lines and Slotlines*, 2nd ed. Hoboken, NJ: Wiley, 2000, ch. 3.

[13] S. H. Hall, G. W. Hall, and J. A. McCall, *High-Speed Digital System Design*. Norwood, MA: Artech House, 1996, ch. 8.

[14] A. Feller, H. R. Haupp, and J. J. Digiacomo, "Crosstalk and reflections in high-speed digital systems," in *Proc. AFIPS Fall Jt. Comput. Conf.*, 1965, vol. 27, pp. 511–525.

[15] G. Kim, D. G. Kam, and J. Kim, "TDR/TDT analysis by crosstalk in single and differential meander delay lines for high speed PCB applications," in *Proc. Conf. Rec. IEEE Int. Symp. Electromagn. Compat.*, vol. 3, pp. 657–662.

[16] Z. Michalewicz, *Genetic Algorithms Plus Data Structures Equals Evolution Programs*. New York: Springer-Verlag, 1996, ch. 1, 3. 3rd rev.

On Complete Functional Broadside Tests for Transition Faults

Hangkyu Lee, Irith Pomeranz, and Sudhakar M. Reddy

***Abstract*—It was shown before that tests applied under nonfunctional operation conditions, which are made possible by scanning in an unreachable state, may lead to unnecessary yield loss. To address this issue, functional broadside tests were defined as broadside tests that use only reachable states of the circuit as scan-in states. Earlier procedures for generating functional broadside tests were not complete, i.e., they did not always detect all the detectable faults or prove that all the undetectable faults are undetectable. In this paper, we address the completeness of the functional broadside tests for transition faults. We describe the implementation of a test-generation procedure that can, for every transition fault, either find a functional broadside test or prove that the fault is undetectable under the functional broadside tests. We present experimental results where complete results are achieved for almost all the benchmark circuits considered.**

***Index Terms*—Functional broadside tests, overtesting, reachable state, scan, test generation.**

I. Introduction

The rapid increase in clock frequency of integrated circuits necessitates the detection of defects that affect the timing behavior of the circuit. Timing defects are typically modeled by delay faults. Transition faults are widely used as a delay-fault model. Test application methodologies for the delay faults in scan designs can be categorized into enhanced scan testing [1], skewed-load testing [2], and broadside testing [3]. All the techniques may suffer from unnecessary yield loss due to overtesting. Even broadside tests, which apply the second vector of every test in functional mode, may result in overtesting [4]. Overtesting for delay faults can be viewed in two ways. Under the first view, overtesting is due to the fact that redundant faults in the original circuit before scan insertion become detectable after scan insertion. It has been observed in [4] that detecting the redundant faults may lead to unnecessary yield loss, i.e., good chips may be discarded as faulty or placed in a lower performance bin due to the detection of a redundant fault. Under the second view, overtesting is caused by

Manuscript received May 3, 2007; revised July 21, 2007. This paper was recommended by Associate Editor N. K. Jha.

H. Lee is with Intel Corporation, Austin, TX 78746 USA (e-mail: hangkyu.lee@intel.com).

I. Pomeranz is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA.

S. M. Reddy is with the Department of Electrical and Computer Engineering, University of Iowa, Iowa City, IA 52242 USA.

Digital Object Identifier 10.1109/TCAD.2008.915531

nonfunctional operation conditions during test. Due to the insertion of scan, a synchronous sequential circuit may enter functionally unreachable states when it is in the test mode. It has been observed in [5] that the nonfunctional operation conditions during test can increase switching activity significantly. The excessive switching activity may lead to supply voltage droops and, as a result, cause defect-free chips to fail delay testing. It has also been observed in [4] that functionally unreachable states, which were loaded into a synchronous sequential circuit by scan, may activate functionally unsensitizable paths and, hence, cause functionally untestable faults to be detected.

We consider synchronizable circuits. As in [6], we assume that functional operation of the circuit starts after applying a synchronizing sequence, which takes the circuit to a fully specified state regardless of the initial state. The synchronization state is a reachable state. Every state that the circuit visits after it is synchronized is also a reachable state.

We consider synchronous sequential circuits with full scan for simplicity. A broadside test for a full-scan circuit consists of two patterns. We denote a two-pattern test by $\langle u_1 \cdot s_1, u_2 \cdot s_2 \rangle$. The test application scheme is as follows. We first load s_1 into the circuit by scan. We next apply u_1 to the primary inputs. The next state s_2 for u_1 and s_1 is latched in the flip-flops. We next apply the primary input vector u_2. Finally, the next state s_3 for u_2 and s_2 is latched in the flip-flops and scanned out.

Several studies to address the problem of overtesting have been reported [4]–[11]. In [10], a procedure in generating scan-based tests with reachable states for stuck-at faults is described. The procedure in [10] is extended to transition faults in [6]. The extended procedure produces a broadside test set with reachable states. Such tests are referred to as functional broadside tests in [6]. Functional broadside tests are scan based, which is similar to the broadside tests. They are referred to as "functional" since they only use reachable states which are also possible during the functional operation.

The procedures proposed in [6] accept a test set $S_{\rm C}$ or a test sequence T and produce a new test set $S'_{\rm C}$ that consists of functional broadside tests by using a simulation-based process. These procedures do not guarantee that all the faults detectable by the functional broadside tests will be detected. A procedure for identifying the undetectable faults is also used in [6] to provide an upper bound on the number of detectable faults. In some cases, the number of detected faults reaches the upper bound, indicating that the results are complete. However, for many circuits, there is a gap between the number of detected faults and the number of faults that were not identified as undetectable.

Our goal in this paper is to eliminate or at least reduce the gap between the number of faults that are known to be detectable by the functional broadside tests and the number of faults that are not known to be undetectable. To define the gap, we denote by D the number of faults that are known to be detectable by the functional broadside tests, by U the number of faults that are known to be undetectable, and by F the total number of faults. For a given test-generation procedure, we have that $D \leq F - U$. The gap is defined as $F-U-D$. Our goal is to increase D and U so as to reduce this gap to zero. For this purpose, we implement a test-generation procedure that has two phases.

The first phase uses simulation-based techniques and constrained test generation for the full-scan circuit. The use of the simulation-based techniques in the first phase is motivated by their ability to detect large numbers of faults, as shown in [6].

In the second phase, we apply a restricted version of a sequential test generator for the faults which are not detected in the first phase. This procedure can detect all the detectable faults and identify all the undetectable faults given enough time.

Due to the use of the sequential test generation, the computational complexity of the proposed procedure is high. Its benefit is that it will allow us to evaluate other efficient procedures with respect to their ability to achieve the maximum possible fault coverage. This can be done by comparing their results with those of the complete procedure for circuits where the complete procedure is applicable.

The rest of this paper is organized as follows. Section II gives a brief description of the first phase which uses simulation-based and constrained full-scan test generation. In Section III, we describe the second phase based on a sequential test generator. Experimental results are presented in Section IV.

II. Phase 1: Simulation-Based Techniques and Constrained Test Generation

In the first phase, we initially generate an unrestricted broadside test $t_i = \langle u_1 \cdot s_1, u_2 \cdot s_2 \rangle$ for a target fault. We then try to define a broadside test that uses only reachable states based on the initial test. We use simulation-based techniques and constrained test generation. We apply simpler techniques first. A fault may be targeted several times, using new reachable states identified during the test-generation process, to eventually obtain a functional broadside test for the fault.

The set of reachable states $S_{\rm reach}$ available during the test generation initially includes only a synchronization state. Whenever new reachable states are identified, $S_{\rm reach}$ is updated. New reachable states are obtained by using the state s_2 of a new test $\langle u_1 \cdot s_1, u_2 \cdot s_2 \rangle$ and are also obtained during the dynamic state search process described later.

We find a synchronizing sequence, and a synchronization state, by a simulation-based process that uses a three-value simulation. We initially set the present state of the circuit to the all-unspecified state. We construct the sequence by considering consecutive time units one after the other. For every time unit, we generate a primary input vector ν by specifying the inputs one at a time and selecting a value with a larger number of specified next-state variables for every input. We apply ν starting from the present state and check if the number of specified next-state variables is larger than the number of specified present-state variables. If not, we generate additional 32 primary input vectors by mutating ν. We then select a primary input vector that leads to the largest number of specified next-state variables. The next state becomes the present state for the following time unit. We repeat this process up to a predefined number of time units or until a fully specified state is reached.

A. State Replacement Process

In the state replacement process, we replace the state s_1 of the initial broadside test $\langle u_1 \cdot s_1, u_2 \cdot s_2 \rangle$ for a fault f by a reachable state from $S_{\rm reach}$. We do not change input vectors u_1 and u_2. s_2 is determined by the response of the circuit to the first pattern.

We first sort reachable states included in $S_{\rm reach}$ in increasing order of Hamming distances from s_1. If there is a reachable state $s_1^{\rm c}$ covered by s_1, we define a functional broadside test $\langle u_1 \cdot s_1^{\rm c}, u_2 \cdot s_2^{\rm c} \rangle$ for f by replacing s_1 with $s_1^{\rm c}$ (c stands for covered). We then perform fault simulation using the new functional broadside test and consider a new target fault. Even if there is no reachable state covered by s_1, we define a new test $\langle u_1 \cdot s_1^{\rm u}, u_2 \cdot s_2^{\rm u} \rangle$ by replacing s_1 with a reachable state $s_1^{\rm u}$ for every $s_1^{\rm u}$ in $S_{\rm reach}$ (u stands for uncovered). We perform fault simulation using the test. Since $s_1^{\rm u}$ is not covered by s_1, it is not guaranteed that the new test detects f. We store the test only if it detects at least one fault which is not detected yet. If f is detected, we consider a new fault. We repeat this process up to MAXREP times or until all the reachable states in $S_{\rm reach}$ have been considered. MAXREP is a user-defined parameter.

B. Constrained Broadside-Test-Generation Process

If the target fault f was not detected by the state replacement process, we go to the constrained broadside-test-generation process. In this process, we generate a broadside test for f under the constraint of setting the initial state to a reachable state.

We first sort reachable states in $S_{\rm reach}$ in increasing order of Hamming distances from s_1. We then select a reachable state $s_1^{\rm g}$ in the sorted order (g stands for generation). We try to generate a broadside test for f under constraints that force the initial state of the circuit to be $s_1^{\rm g}$. If it is successful, we have a test $\langle u_1^{\rm g} \cdot s_1^{\rm g}, s_2^{\rm g} \cdot s_2^{\rm g} \rangle$ that is a functional broadside test for f. We perform fault simulation of the test and select a new target fault. If it is not successful, we select another reachable state in the sorted order and repeat the test-generation process up to MAXGEN times. MAXGEN is a user-defined parameter. Since a reachable state is a fully specified state, the constraint to set the initial state to a reachable state is often not compatible with the original objectives of the test-generation process to activate a fault and propagate its effect to an output. However, such incompatibility is easily identified by simple implications. Therefore, we can consider a significant number of reachable states in the constrained broadside-test-generation process.

C. Dynamic State Search Process

The state replacement process and the constrained broadside-test-generation process try to define the functional broadside tests based on reachable states that exist in $S_{\rm reach}$. In this section, we present a simulation-based method to search for a reachable state required to define a functional broadside test for a target fault if it cannot be found in $S_{\rm reach}$. The method is applied to find a reachable state covered by the state cube s_1 of the initial broadside test.

A simulation-based procedure to compute the reachable states starting from the all-unspecified state is described in [6] and [10]. Starting from the all-unspecified state, this procedure explores new states by constructing a tree. The root of the tree contains the all-unspecified state. The next level of the tree contains next states that are reached by applying a limited number of input vectors starting from the all-unspecified state. For every level of the tree, a single state is selected, and its next states form the next level of the tree. Every fully specified state s obtained during this process is a reachable state since the tree contains an input sequence into s that starts from the all-unspecified state. In our approach, we start with a known reachable state at the root of the tree, and we allow more than one state at each level to be selected for producing new reachable states. The number of input vectors applied to compute the next states for every selected state is a constant denoted by NUMRAND. The number of states selected for the root of the tree is a constant MAXSEARCH.

D. Broadside-Test-Regeneration Process

In the techniques described in the previous sections, we try to generate a functional broadside test based on an initial unconstrained broadside test $t_i = \langle u_1 \cdot s_1, u_2 \cdot s_2 \rangle$ for a target fault. If the techniques based on t_i fail to generate a functional broadside test, the same techniques based on a different test $t_i^{\rm d}$ may be successful (d stands for different). Therefore, if t_i does not lead to defining a functional broadside test for a target fault, we generate a different broadside test $t_i^{\rm d}$ and repeat all the techniques described in the previous sections being guided by $t_i^{\rm d}$. We generate a new test $t_i^{\rm d}$ a constant number of times denoted by MAXMOD.

To generate a new test $t_i^{\rm d}$ for a target fault, we generate $t_i^{\rm d}$ under the constraint that the initial state of $t_i^{\rm d}$ would be different than s_1. For this purpose, we select a specified bit of s_1. Suppose that bit i is selected and that its value under s_1 is 0(1). In the initial state of $t_i^{\rm d}$, we set bit i to 1(0).

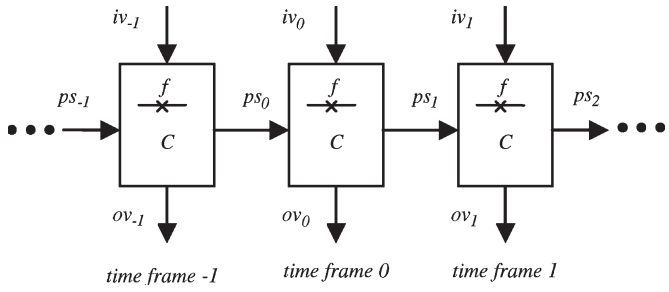

Fig. 1. ILA model for the sequential-test-generation procedure.

III. Phase 2: Sequential-Test-Generation-Based Techniques

In the first phase, $S_{\rm reach}$ may not be the complete set of reachable states. In addition, we may not try to use every reachable state in $S_{\rm reach}$ in generating a functional broadside test. Therefore, the first phase is not complete. In the second phase, we use a restricted sequential test generator, targeting only the faults which are not detected in the first phase, in order to achieve completeness.

A. Sequential-Test-Generation Procedure

In this section, we briefly describe a sequential-test-generation procedure for stuck-at faults, which is adjusted for generating the functional broadside tests targeting transition faults later. Sequential test generation is typically based on an iterative logic array (ILA) which models the operation of a synchronous sequential circuit by an array of combinational logic blocks (time frames), as shown in Fig. 1. iv_i, ps_i, and ov_i represent the input vector, the present state, and the output vector of time frame i, respectively. Each block C in the ILA represents the combinational logic of the synchronous sequential circuit (C stands for combinational). The input vectors ordered in time $\langle iv_{-k}, iv_{-k+1}, \ldots, iv_{m-1}, iv_m \rangle$ form an input sequence for the sequential circuit. A single stuck-at fault in the sequential circuit is equivalent to multiple stuck-at faults in the ILA. In other words, every block C in the ILA contains the stuck-at fault, as shown in Fig. 1.

A test-generation procedure for a sequential circuit based on the ILA typically consists of two steps: fault activation and propagation, and state justification. In the fault activation and propagation step, the fault is activated at a certain time frame by creating opposite values at the fault site between the fault-free and the faulty circuits. The fault effect is then propagated to an output in the same time frame or in a later one. In the state-justification step, we try to find an input sequence which takes the sequential circuit from the all-unspecified state to the present state of the time frame in which the fault is activated.

B. Functional Broadside Test Generation Based on a Sequential Test Generator

In [6], a complete method for generating the functional broadside tests using a sequential test generator was outlined. We implement this method in the second phase of the proposed test-generation procedure. We next describe how we adjust a sequential-test-generation procedure for stuck-at faults into a complete procedure for generating the functional broadside tests for transition faults.

Suppose that our goal is to generate a functional broadside test for a slow-to-rise (slow-to-fall) transition fault on a line l in a synchronous

sequential circuit. We assume that the next-state lines as well as the primary outputs of the last time frame (time frame 1) are observable. We then need to generate a test sequence to assign 0 (1) to line l in time frame 0 and detect the stuck-at 0 (1) fault in time frame 1 starting from the all-unspecified state. This can be modeled by an ILA which has the stuck-at 0 (1) fault only in time frame 1 in addition to the initialization requirement in time frame 0. We thus use only one time frame (the last time frame in the ILA) in propagating the fault effect to an observable point. The state-justification process is not modified.

Suppose that the sequential test generator creates a test sequence $\langle iv_{-k}, iv_{-k+1}, \ldots, iv_0, iv_1 \rangle$, which implies that if we apply the test sequence, the circuit goes from the all-unspecified state ps_{-k} at time frame -k to ps_0 at time frame 0. If ps_0 is fully specified, ps_0 is a reachable state by definition 1. By setting $s_1 = ps_0$, $u_1 = iv_0$, $s_2 = ps_1$, and $u_2 = iv_1$, we obtain a functional broadside test $\langle u_1 \cdot s_1, u_2 \cdot s_2 \rangle$ for the target fault. If ps_0 is partially specified, ps_0 represents a set of states which includes at least one reachable state. We can find a reachable state ps'_0 included in ps_0 by simulating the test sequence $\langle iv_k, iv_{k-1}, \ldots, iv_{-2}, iv_{-1} \rangle$ from the synchronization state or the reset state. If we set $s_1 = ps'_0$, $u_1 = iv_0$, $s_2 = ps'_1$, and $u_2 = iv_1$, we obtain a functional broadside test for the target fault. Here, ps'_1 is the next state obtained under $s_1 = ps'_0$ and $u_1 = iv_0$.

The generation of the functional broadside tests using a sequential test generator is a complete method in the sense that we can always generate a functional broadside test for a given fault or prove that the fault is undetectable provided with sufficient time and memory space.

The second phase is implemented based on the sequential-test-generation procedure MIX [12]. We briefly review the techniques used in MIX that are relevant to our procedure. As a preprocessing step, MIX attempts to identify values that cannot be assigned on a single flip-flop and on pairs of flip-flops. These values are used to avoid unreachable states in the fault activation and propagation step and the state-justification step. State justification is performed backward in time by considering several time frames simultaneously. The number of time frames required for state justification is estimated by using controllability and observability measures. In order to determine dependences among flip-flop values, dynamic learning at the present-state lines of a time frame under consideration is performed. To guide the decision process during the state-justification step, MIX uses a partial state transition graph in which information on all the reachable states visited during the test-generation procedure is recorded. To eliminate the search in the irrelevant space during the fault activation and propagation step, MIX identifies and removes J-frontier elements which do not contribute to the fault activation and propagation.

IV. Experimental Results

The procedure in generating the functional broadside tests for transition faults is applied to ISCAS-89 and ITC-99 benchmark circuits. We only consider circuits for which we can compute synchronizing sequences from the all-unspecified state using a three-value logic. This precludes nonsynchronizable circuits such as s9234, s13207, and others. These circuits are also untestable as sequential circuits, which is also a problem for the sequential-test-generation process embedded in our test-generation procedure.

We use the following parameters. We use an identical value for MAXREP and MAXGEN. We increase these parameters in increments of 100 up to 800 if this increases the number of detected faults. We use NUMRAND $= 100$, MAXSEARCH $= 100$, and MAXMOD $= 10$. In addition, we set a limit of TL $= 12$ h on the runtime of the first phase.

In Table I, we show the results of the test-generation procedure. Under column Fault, we show the total number of faults. The selected values of MAXREP $=$ MAXGEN are given under column Limit. Under column Phase 1, the number of faults detected (Detect) and the number of undetectable faults (Undet) identified in the first phase are given. Faults are proven to be undetectable by the broadside-test-generation procedure in the first phase. A fault undetectable under broadside testing is undetectable under functional broadside testing since the functional broadside tests are a subset of the broadside tests. Under column Phase 2, the number of faults detected and the number of faults shown to be undetectable in the second phase are listed. Under column Abort, we show the number of faults aborted after the second phase. The number of tests generated and the runtimes in seconds on a Sun U80 Workstation are shown under columns Test and Time, respectively.

TABLE I
Results of the Test-Generation Procedure

Circuit	Fault	Limit	Phase 1		Phase 2		Abort	Test	Time [s]
			Detect	Undet	Detect	Undet			
s298	508	100	355	105	0	48	0	62	966
s344	552	100	497	30	0	25	0	76	613
s349	566	100	505	36	0	25	0	71	635
s382	646	100	488	146	0	12	0	66	7014
s386	690	100	505	160	0	25	0	100	576
s444	764	100	554	196	0	14	0	76	4778
s526	948	300	571	358	0	17	2	120	4711
s641	734	100	575	35	0	124	0	149	3000
s713	918	100	648	141	0	129	0	171	11972
s820	1574	100	1281	291	2	0	0	249	92
s832	1614	100	1290	324	0	0	0	270	119
s1196	2110	100	2108	2	0	0	0	347	15
s1238	2316	100	2234	82	0	0	0	394	20
s1423	2512	500	2207	273	0	11	21	353	77209
s1488	2770	100	2529	241	0	0	0	304	237
s1494	2810	100	2548	262	0	0	0	303	118
s5378	7040	800	3433	11	1920	1663	13	648	162317
s35932	63502	100	54598	8903	1	0	0	376	9957
b02	112	100	98	14	0	0	0	27	1
b03	694	100	609	28	0	57	0	61	2582
b04	2710	100	2248	111	0	351	0	333	43237
b06	236	100	199	33	0	4	0	26	12
b07	1940	100	1419	203	0	310	8	81	202807
b08	732	500	575	150	1	6	0	80	267
b09	722	100	531	137	0	54	0	82	2422
b10	824	100	589	164	0	71	0	93	1908
b11	2518	200	1622	530	5	361	0	207	54461

For all the circuits except s526, s1423, s5378, and b07, we generated tests for all the detectable faults and identified all the undetectable faults under the functional broadside testing. This demonstrates the completeness of the test-generation procedure. In addition, for almost all circuits, the first phase generated the functional broadside tests for all the detectable faults. This confirms the effectiveness of the simulation-based techniques in generating the functional broadside tests [6]. A very small number of faults is detected by the second phase for s820, s35932, b08, and b11. For s5378, the first phase reaches the time limit TL and switches to the second phase before considering all the faults in the first phase.

Some faults in s526, s1423, s5378, and b07 are aborted since the state-justification process could not justify a target state even if we allowed a large backtrack limit.

To compare the results of the procedures from [6] with the results of our procedure, we define a gap for every procedure as the difference between the total number of faults and the number of faults resolved. A fault is resolved if it is detected or proved to be undetectable. For our procedure, the gap corresponds to the number of aborted faults in Table I. In [6], faults are proved to be undetectable by a procedure

TABLE II
RESULTS OF THE FAULT SIMULATION FOR REDUNDANT FAULTS

Circuit	Test	Fault	Redundant	Detected
s298	62	508	153	0
s344	76	552	55	0
s349	71	566	61	0
s382	66	646	146	0
s386	100	690	185	0
s444	76	764	196	0
s526	120	948	361	0
s641	149	734	159	0
s713	171	918	270	0
s820	249	1574	291	0
s832	270	1614	324	0
s1196	347	2110	5	3
s1238	394	2316	85	3
s1423	353	2512	273	0
s1488	304	2770	241	0
s1494	303	2810	262	0
s5378	648	7040	1789	142
s35932	376	63502	8903	0
b02	27	112	19	5
b03	61	694	75	17
b04	333	2710	458	0
b06	26	236	37	0
b07	81	1940	553	127
b08	80	732	150	0
b09	82	722	183	0
b10	93	824	210	0
b11	207	2518	778	0

that identifies only a subset of the undetectable faults. The gap can thus be due to the failure to identify faults as undetectable or due to the failure to generate a test for a detectable fault. Two notable examples are s5378, where the procedures from [6] have gaps of 826 and 91, whereas our procedure has a gap of 13, and b03 where the corresponding gaps are 55, 55, and 0. Overall, the results demonstrate that the proposed procedure is complete in cases where the earlier procedures were not complete. The opposite does not happen for any circuit.

As mentioned in Section I, overtesting can be viewed as a consequence of detecting the redundant faults or of the nonfunctional operation conditions due to scan. The functional broadside tests address the second view by allowing only the reachable states as the scan-in states. We investigated the relation between the two views by simulating redundant transition faults under the functional broadside tests generated by the proposed procedure. We identified functionally redundant faults using the procedure from [13]. The results of fault simulation are given in Table II. We show the number of functionally redundant faults identified by [13] under column Redundant. The number of functionally redundant faults detected by the functional broadside tests is given under column Detected. For most of the circuits, the functional broadside tests do not detect any redundant faults. The functionally redundant faults may be detected by the functional broadside tests when the fault effect of a redundant fault can be propagated to a flip-flop but not to a primary output.

V. CONCLUSION

We described the implementation of a complete procedure in generating the functional broadside tests for transition faults. The functional broadside tests were proposed as a way to reduce unnecessary yield loss due to the nonfunctional operation conditions made possible by scanning in an unreachable state. The procedure consisted of two phases. The first phase was simulation based. In the second phase, we applied a restricted sequential test generator to the faults which were not detected in the first phase in order to detect the faults or prove that they are undetectable, thus achieving completeness. The effectiveness of the test-generation procedure was demonstrated by detecting all the detectable faults and identifying all the undetectable faults in almost all the benchmark circuits considered.

For completeness, the proposed procedure requires a sequential test generation. Thus, it is limited to circuits for which the sequential test generation is feasible. For such circuits, it provides results that can be used to estimate the completeness of other procedures which are less computationally intensive.

REFERENCES

[1] S. DasGupta, R. G. Walther, T. W. Williams, and E. B. Eichelberger, "An enhancement to LSSD and some applications of LSSD in reliability, availability and serviceability," in *Proc. Fault-Toler. Comput. Symp.*, 1981, pp. 880–885.

[2] S. Patil and J. Savir, "Skewed-load transition test: Part I, Calculus," in *Proc. Int. Test Conf.*, 1992, pp. 705–713.

[3] J. Savir and S. Patil, "On broad-side delay test," in *Proc. VLSI Test Symp.*, 1994, pp. 284–290.

[4] J. Rearick, "Too much delay fault coverage is a bad thing," in *Proc. Int. Test Conf.*, 2001, pp. 624–633.

[5] J. Saxena, K. M. Butler, V. B. Jayaram, S. Kundu, N. V. Arvind, P. Sreeprakash, and M. Hachinger, "A case study of IR-drop in structured at-speed testing," in *Proc. Int. Test Conf.*, 2003, pp. 1098–1104.

[6] I. Pomeranz and S. M. Reddy, "Generation of functional broadside tests for transition faults," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2207–2218, Oct. 2006.

[7] I. Pomeranz and S. M. Reddy, "On application of output masking to undetectable faults in synchronous sequential circuits with design-for-testability logic," in *Proc. Int. Conf. Comput.-Aided Des.*, 2003, pp. 867–872.

[8] H. Lee, I. Pomeranz, and S. M. Reddy, "A test generation procedure for avoiding the detection of functionally redundant transition faults," in *Proc. VLSI Test Symp.*, 2006, pp. 294–299.

[9] X. Liu and M. S. Hsiao, "Constrained ATPG for broadside transition testing," in *Proc. Int. Symp. Defect Fault Toler. VLSI Syst.*, 2003, pp. 175–182.

[10] I. Pomeranz, "On the generation of scan-based test sets with reachable states for testing under functional operation conditions," in *Proc. Des. Autom. Conf.*, 2004, pp. 928–933.

[11] Y. C. Lin, F. Lu, K. Yang, and K. T. Cheng, "Constraint extraction for pseudo-functional scan-based delay testing," in *Proc. Asia South Pac. Des. Autom. Conf.*, 2005, pp. 166–171.

[12] X. Lin, I. Pomeranz, and S. M. Reddy, "MIX: A test generation system for synchronous sequential circuits," in *Proc. Int. Conf. VLSI Des.*, 1998, pp. 456–463.

[13] G. Chen, S. M. Reddy, and I. Pomeranz, "Procedures for identifying untestable and redundant transition faults in synchronous sequential circuits," in *Proc. Int. Conf. Comput. Des.*, 2003, pp. 36–41.

[14] I. Pomeranz and S. M. Reddy, "On the use of fully specified initial states for testing of synchronous sequential circuits," *IEEE Trans. Comput.*, vol. 49, no. 2, pp. 175–182, Feb. 2000.