Functional Scan Chain Testing

Douglas Chang
CS Department
University of California
Santa Barbara, CA 93106

Mike Tien-Chien Lee
Avant! Corp.
46871 Bayside Parkway
Fremont, CA 94538

Kwang-Ting Cheng
ECE Department
University of California
Santa Barbara, CA 93106

Malgorzata Marek-Sadowska
ECE Department
University of California
Santa Barbara, CA 93106

Abstract

Functional scan chains are scan chains that have scan paths through a circuit's functional logic and flip-flops. Establishing functional scan paths by test point insertion (TPI) has been shown to be an effective technique to reduce the scan overhead. However once the scan chain is allowed to go through functional logic, the traditional alternating test sequence is no longer enough to ensure the correctness of the scan chain. We identify the faults that affect the functional scan chain, and show a methodology to find tests for these faults. Our results have the number of undetected faults at only 0.006% of the total number of faults, or 0.022% of the faults affecting the scan chain.

1 Introduction

Automatic test pattern generation (ATPG) for sequential circuits is in general a difficult problem. Design-for-test (DFT) techniques, such as full scan [11] and partial scan [1, 3, 5], have been proposed to reduce the search space to help ATPG software perform more efficiently. These techniques facilitate the testing of a sequential circuit by interconnecting selected flip-flops into a shift register during test mode. However the area and delay overhead from conventional scan can be significant due to the extra scan multiplexers in the scan flip-flops, and the extra routing for the scan chain.

Functional scan paths can be used to reduce the scan overhead by using mission functional logic and mission flip-flops to establish a scan path. Establishing scan paths through functional logic has been used to reduce scan overhead for control paths [6, 9], for data paths [8], and also for built-in self-test [2]. Note that functional scan paths not only reduce the scan overhead in terms of area, but by using functional scan on the critical path, it can also eliminate or reduce the scan performance overhead [7].

Functional scan paths can be established by test point insertion (TPI) [7]. TPI establishes a functional scan path between two flip-flops by finding a combinational path between them and forcing the side-inputs to this path to non-controlling values during scan mode. This is done by inserting test points and using the primary input assignments to force nets to 0 or 1 during scan mode. A single test point may help establish several scan paths, and therefore the TPI method can be more area efficient than the conventional MUXed-scan flip-flop substitution. Furthermore, the dedicated scan wiring is saved by using sensitized functional scan paths. TPI has been shown to be an effective technique to establish functional scan paths thereby reducing scan overhead [7].

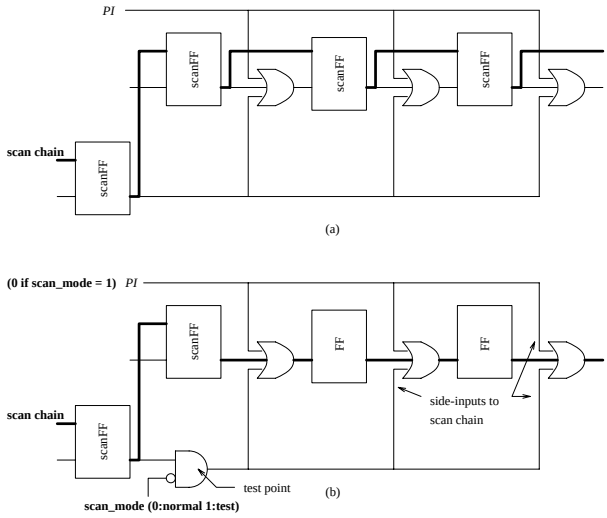

Figure 1: (a) conventional full scan; (b) test point insertion to reduce scan overhead.

Using TPI to establish functional scan paths is illustrated by an example in Figure 1. Figure 1(a) gives a portion of a sequential circuit using the conventional full scan methodology, which needs four scan flip-flops and extra scan chain wiring (thick lines). By inserting a test point and applying 0 at the primary input PI during scan mode (scan_mode = 1), as shown in Figure 1(b), a functional scan path is established and the scan overhead is reduced.

One important issue that has not been fully addressed by the previous research work is the testing of the functional scan chain itself. This is critical because the subsequent testing procedures assume that the scan chain is fault-free. The traditional method of testing a dedicated scan chain is to apply an alternating sequence of 0s and 1s (e.g., 00110011...). This is sufficient to detect faults when using a traditional scan chain with dedicated scan wiring since a stuck-at 1 (s-a-1) or a stuck-at-0 (s-a-0) fault on the dedicated scan wiring will cause the scan output to have a tail of 1s (0s). Subsequent testing of the circuit logic is then performed with the assumption that the scan chain is fault-free. However the alternating sequence may not be enough to test a functional scan chain. For example,

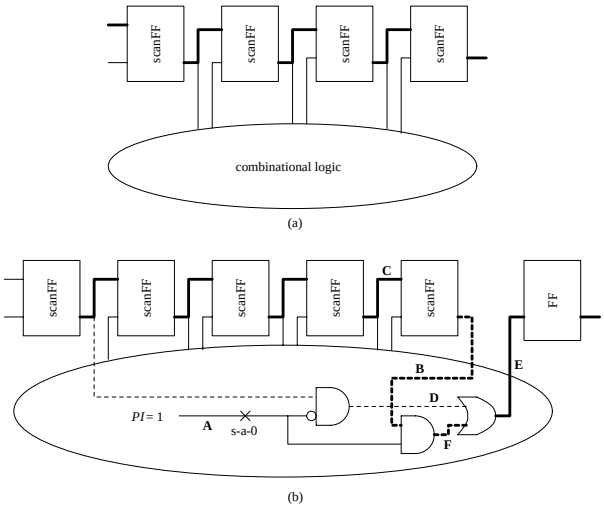

Figure 2: (a) Alternating sequence detect faults when using dedicated scan chain, but (b) will not detect fault A s-a-0 when using functional scan chain

Figure 2a shows a sequential circuit using the traditional scan design, and Figure 2b shows a sequential circuit using a functional scan path (shown by thick dashed lines). In Figure 2a any s-a-1 (0) fault on the scan chain (shown by thick lines) can be detected by the alternating sequence, and any fault in the functional logic will not affect the scan chain. However in Figure 2b a stuck-at-0 fault on net A in the functional logic will affect the functional scan chain. Under the presence of this fault, the functional scan path changes from the thick dashed line to the thin dashed line. The length of the scan chain is reduced by four. However since the alternating sequence (00110011...) repeats every four clock cycles, the alternating sequence will not detect this fault. Thus in addition to the alternating sequence we propose to perform additional testing to ensure the correctness of the scan chain.

In this paper we study the problem of functional scan chain testing. This problem is to find test vectors to detect faults in the combinational logic that affect the functional scan chain. These test vectors are applied during the scan chain testing phase (i.e., the circuit is strictly kept in scan mode). Since the testing of the functional scan chains detects some faults in the combinational logic, these detected faults can be dropped from the fault list for the subsequent phase of testing the combinational logic through the use of the fault-free scan chain.

In section 2 we give a brief overview of our approach. In section 3 we discuss how to find the faults that affect the scan chain. In section 4, we give a procedure for finding test vectors for the faults that affect the scan chain. In section 5, we give another procedure to find tests for the faults not yet detected. In section 6 we give the experimental results. Finally, in section 7 we give the conclusion.

2 Overview of our approach

Our approach is to start with the circuit model in scan mode. This is the circuit with test points inserted, and with primary inputs assigned to values according to TPI (including setting scan_mode = 1). Also during TPI, the nets in the scan chain are identified. Then the faults f_{sc} that may affect the scan chain are identified. A straight-forward method to find test vectors to detect f_{sc} is to use sequential ATPG on the scan mode circuit model. However, even though the scan mode circuit model is much simpler than the original circuit, in large circuits, this could still involve a significant amount of CPU time. We thus attempt to reduce the number of faults that need to be detected using sequential ATPG by using a three step screening process. Each step will detect some faults, and leave the remaining undetected faults for the next step. Sequential ATPG will only be used in the final step.

1. The first step is to identify the faults f_{easy} that cause a net on the scan chain to be s-a-0 or s-a-1. These faults will be detected by the alternating sequence, which will be the first test sequence to be applied to the circuit. The remaining faults, $f_{hard} = f_{sc} - f_{easy}$ are input to step two.

2. Combinational ATPG with fault list f_{hard} is applied to the scan-mode circuit model to get a set of test vectors. However, even if combinational ATPG finds a test vector to detect a fault, this vector may not really detect that fault. This is because to apply a test vector from combinational ATPG, the scan chain should be fault-free which may not be true. Thus we must follow combinational ATPG by sequential fault simulation to determine which faults are really detected. Note that the circuit is fixed in the scan mode during the entire sequence (including the combinational test). Since we are using combinational ATPG to generate a test set, this approach is applicable to a full scan environment. However, in a partial scan environment, we can use a test set of random vectors. The remaining undetected faults, $f_{remaining}$ are input to step 3.

3. The faults in $f_{remaining}$ are examined to determine where they affect the scan chain. If a fault could reach the scan chain in locations $l_i, l_{i+1}, \ldots, l_j$, where $l_i < l_{i+1} < \cdots < l_j$, then the scan chain before l_i is fault-free and controllable, while the scan chain after l_j is fault-free and observable (assuming single stuck-at fault model). This increased controllability and observability is used along with sequential ATPG to find tests for the faults in $f_{remaining}$.

In the following sections we discuss each of the above steps in greater detail.

3 Find Faults Affecting the Scan Chain

Faults that affect the scan chain are found by the following procedure. First find the forward implication cone of each fault. Under the presence of the fault, nets in the forward implication cone can change from any of the values {0, 1, X} to any of {0, 1, X}. For example in

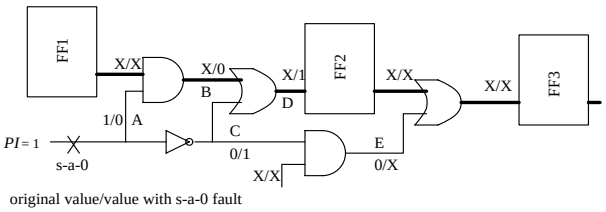

Figure 3: Finding the forward implication of a fault

Figure 3 under the s-a-0 fault at PI shown, the value at net A is changed from 1 to 0, net B from X to 0, net C from 0 to 1, net D from X to 1, and net E from 0 to X. Note that the forward implication cone indicates not only whether a fault affects the scan chain, but also where (i.e., between which pair of flip-flops) it affects the scan chain. Furthermore, a fault may affect the scan chain in more than one location. After computing the new net assignments due to the forward implications of the fault, each fault is categorized into one of three types depending on the last location it affects the scan chain. The last location is used because a fault captured in that flip-flop is guaranteed observable at the scan-out. The three categories are:

1. Under the presence of the fault, a net on the scan chain is assigned to a 1 or a 0 (i.e., net is s-a-1 or s-a-0).
2. Under the presence of the fault, a side-input to the scan chain is assigned an unknown value (X).
3. neither 1 nor 2.

Some examples of faults in category 1 for the circuit shown in Figure 2 are net B s-a-1, net C s-a-0, or net D s-a-1 (causes E to be s-a-1). An example of category 2 is Figure 2, net A s-a-0, which causes net D to have an unknown value. The fault net A s-a-0 could also be placed in category 1 because net F, which is on the scan chain is s-a-0 under the presence of this fault. However we give category 2 priority over category 1. This means that any fault that is in both category 1 and 2 is placed in category 2. This is because faults in category 1 can be detected by the alternating sequence, and thus are easy to detect, while faults in category 2 may not be detected by the alternating sequence. An example of a fault in category 3 is Figure 2, net A s-a-1.

An example of a fault affecting multiple locations is the s-a-0 fault at PI shown in Figure 3. The fault affects the scan chain between flip-flops FF1 and FF2 because it affects net B and D which are on the scan chain (category 1). It also affects the scan chain between flip-flops FF2 and FF3 because it affects net E, which is a side-input to the scan chain (category 2). This fault is placed into category 2 because the last place it affects the scan chain is between FF2 and FF3.

Any fault that falls into category 3 will not affect the scan chain. This is because the scan chain can only be affected by faults on nets in the scan chain or by faults that affect the chain's side-input nets. The former faults are covered by category 1. For the latter faults, there are three possibilities. A side-input gets a controlling value, which causes a net in the scan chain to be s-a-1 or s-a-0 (category 1), or a side-input gets an unknown value (category 2), or a side-input becomes a non-controlling value, which means the scan chain is not affected by the fault (category 3). The faults in category 1 can be detected by the alternating sequence and the faults in category 3 do not affect the scan chain. Thus we now concentrate on the faults in category 2 (f_{hard}).

4 Combinational ATPG & Sequential Fault Simulation

For each fault $f \in f_{hard}$, a sequence of test vectors to detect f must be generated, or it must be shown that f is sequentially undetectable (in scan mode), in which case f does not affect the scan chain. This is under the constraint that the circuit is strictly kept in scan mode. For functional scan, some primary inputs may be set to constant values in scan mode to establish the scan chain. However, the primary inputs that are not used to establish the functional scan chain can be used to detect the fault. Also, in addition to the scan-out, all the primary outputs can be used to observe the fault effect.

To find test vectors to detect the faults in f_{hard} we could apply random test vectors or use sequential ATPG. However random test vectors will be unable to identify sequentially undetectable faults. Sequential ATPG, on the other hand, is able to identify undetectable faults, but is very time-consuming. We thus choose to apply combinational ATPG targeting the faults in f_{hard} on the combinational portion of the *scan-mode circuit*. The test vectors returned by combinational ATPG are then modified to be applied to the sequential circuit. In this modification scan-in and scan-out sequences are added before and after the combinational vector similar to traditional scan sequence conversion. However, it should be stressed again that the combinational vector is also applied in scan mode.

The test sequences created above may fail to detect faults because the target fault may damage the scan chain, and mask the fault effect during the scan out phase. Thus we need to apply sequential fault simulation using the test sequence created above to find which faults in f_{hard} are really detected. After fault simulation the remain undetected faults, $f_{remaining}$, are targeted.

Note that this step can possibly identify some undetectable faults since any fault found to be combinationally undetectable (by the combinational ATPG), must also be sequentially undetectable.

5 Targeting the Remaining Faults

After the above step, the remaining faults $f_{remaining}$ are either sequentially undetectable or just hard to detect. To find test vectors for these faults we must use the fault location information. This will allow us to use the fault-free sections of the scan chain to get increased controllability and/or observability for test generation. Given a fault we can determine the location(s) where it affects the scan chain as described in section 3 above. Then, assuming the single stuck-at fault model, the scan chain ahead of the first fault location is guaranteed fault-free and therefore

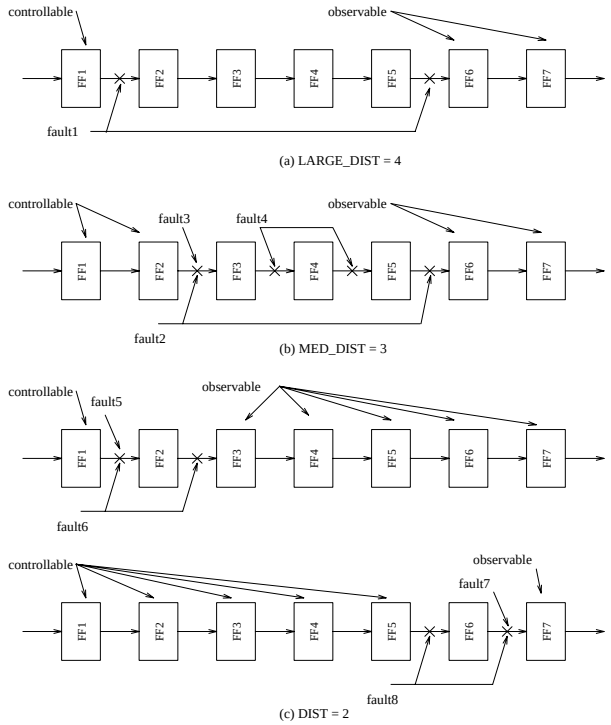

Figure 4: Even if a fault affects the scan chain, part of the chain can still be used.

can be assumed to be fully controllable (i.e., treated as primary inputs during test generation). Similarly the scan chain after the last fault location is fully observable. The scan chain ahead of the fault can be used to scan-in values, and similarly the scan chain after the fault can be used to scan-out values. This is shown in Figure 4a, where a fault affects the scan chain between FF1 and FF2 and also between FF5 and FF6. However FF1 is still controllable, and FF6 and FF7 are still observable. We will call this circuit 1-1.C,6-7.O to indicate that flip-flip 1 to flip-flop 1 are controllable, and flip-flip 6 to flip-flip 7 are observable. This controllability and observability can be used to help sequential ATPG detect the fault. Although in general sequential ATPG is time-consuming, in this case we find that it is quite effective. This is because the scan-mode circuit is much simpler than the operation-mode circuit. The fault-free scan-mode circuit is simply a shift register. Also using the increased controllability and observability reduces the length of the scan chain and the size of the scan-mode circuit. Thus sequential ATPG can be efficiently applied in this instance. For each fault $f_r \in f_{remaining}$, a maximally controllable/observable circuit can be created as described above. Then sequential ATPG with fault list $\{f_r\}$ can be applied to the created circuit model to detect f_r. However if $|f_{remaining}|$ is large, this may be inefficient since sequential ATPG would have to be run in the worst case $|f_{remaining}|$ times. Instead we propose to divide up the faults into three groups. The basic idea with these groups is to let the sequential ATPG have enough controllability and observability to be able to detect the faults in its fault list, but to minimize the number of times that sequential ATPG has to be run.

Groups 1 and 2 only contain faults that affect the scan chain in multiple locations ($l_1, l_2, \ldots, l_n$). Additionally, the faults in group 1 have $\max(l_i) - \min(l_j) \geq LARGE\_DIST$, where $LARGE\_DIST$ is a user parameter. Similarly, the faults in group 2 have $LARGE\_DIST > \max(l_i) - \min(l_j) \geq MED\_DIST$, where MED_DIST is a user parameter. All other faults go into group 3. Group 1 faults have only a little additional controllability/observability. Group 2 faults have some more controllability/observability. Group 3 faults have considerable controllability/observability. For example, given the 8 faults that affect the scan chain as shown in Figure 4a-c, and given $LARGE\_DIST = 4$, $MED\_DIST = 3$, then fault1 is in group 1, fault2 is in group 2, and all other faults are in group 3.

All faults in group 1 are treated individually, i.e. a maximally controllable/observable circuit will be created for each fault and sequential ATPG will be applied to detect the fault. For example, for fault1 in Figure 4a, the circuit 1-1.C,6-7.O will be created. Sequential ATPG with the fault list {fault1} will be applied to the modified scan-mode circuit model to test for fault1.

A maximally controllable/observable circuit, n-m.C,o-p.O will be created for each fault f in group 2 as well. However, the fault list for sequential ATPG includes f and also any faults whose $\min(l_j) \geq m$ and $\max(l_i) < o$. These faults can also be detected in n-m.C,o-p.O. For example for fault2 in Figure 4b, the circuit 1-2.C,6-7.O will be created. Sequential ATPG with fault list {fault2, fault3, fault4} will be applied to this circuit to test for fault2, fault3, and fault4. Finally, the group 3 faults are divided into the minimal number of groups such that for each group, $\min(l_i) - \max(l_j) \leq DIST$, where $DIST$ is a user parameter. For example, in Figure 4c, if $DIST = 2$ then the circuit 1-1.C,3-7.O will be created to test for fault5 and fault6, and the circuit 1-5.C,7-7.O will be created to test for fault7 and fault8.

After the above procedure, if there are still undetected faults, we target each of these final faults (f_{final}) individually (as done to group 1). In this last step, we give the sequential ATPG program additional time to try to find a test for these faults. In the case where a circuit is divided into multiple scan chains, if a fault does not affect all of the scan chains, then the flip-flops in the unaffected scan chains are both observable and controllable. All faults that affect more than one scan chain are placed into group 1.

6 Experimental results

We implemented the above functional scan chain testing technique, and tested it on the 12 largest ISCAS '89 benchmarks. The circuits in our test suite were optimized by the SIS script, *script.algebraic* [10], and mapped for minimal area using the technology libraries *nand-nor.genlib* and *mcnc_latch.genlib*. Our technique uses combinational ATPG, sequential simulation, and sequential ATPG. We use the stg3 program [4] to do all of these. Our experiments were performed on a SPARCstation 4.

Table 1: Test suite.

name	*#gates*	*#FFs*	*#faults*	*#chains*
s1196	562	18	2110	1
s1238	589	18	2293	1
s1423	812	74	2854	1
s1488	588	6	2388	1
s1494	604	6	2434	1
s5378	1587	162	5722	1
s9234	1282	135	4598	1
s13207	2755	453	10172	2
s15850	4450	540	16236	2
s35932	14382	1728	48844	6
s38417	13757	1463	51968	5
s38584	12678	1294	47821	5
total	54046	5897	197440	27

Table 2: Finding easy and hard faults.

	faults affecting scan chain		
name	*#easy (%)*	*#hard (%)*	*CPU*
s1196	94 (5%)	21 (1%)	3s
s1238	117 (5%)	39 (2%)	3s
s1423	587 (21%)	138 (5%)	12s
s1488	102 (4%)	64 (3%)	4s
s1494	116 (5%)	42 (2%)	5s
s5378	698 (12%)	72 (1%)	27s
s9234	975 (21%)	257 (6%)	26s
s13207	2452 (24%)	514 (5%)	65s
s15850	3640 (22%)	599 (4%)	196s
s35932	16803 (34%)	879 (2%)	1913s
s38417	4978 (10%)	791 (2%)	448s
s38584	12034 (25%)	2875 (6%)	1074s
total	42596 (22%)	6291 (3%)	3776s

We first used TPI [7] to create scan paths through the functional logic. We then take advantage of the functional scan paths to create a functional scan chain. Except where functional scan paths are established, the ordering of the scan chain is arbitrary. This is important because different orderings will lead to faults affecting the scan chain in different locations, and thus potentially increasing or decreasing the fault coverage. We do not take advantage of this, but instead allow the designer to use this flexibility.

In Table 1 we show our test suite. The first three columns give the name of the circuit, the number of mapped gates in the circuit, and the number of flip-flips in the circuits. Column 4 contains the total number of considered faults in the circuit. Column 5 contains the number of scan chains in the circuit. We use multiple scan chains for the larger circuits to reduce the length of the scan chain to a reasonable size. In Table 2 we show the results of our initial step where we identify the faults affecting the scan chain and the faults that can be detected with the alternating sequence. Column 2(3) contains the number of faults that affect the scan chain and can (may not) be detected by the alternating sequence. Within the parenthesis is the percentage of those faults as compared to the total number of faults. As can be seen by the total figures, the $(42596 + 6291)/197440 = 24.8\%$ of the faults affect the scan chain. However, only $6291/197440 = 3.2\%$ of the faults may not be detected by the alternating sequence.

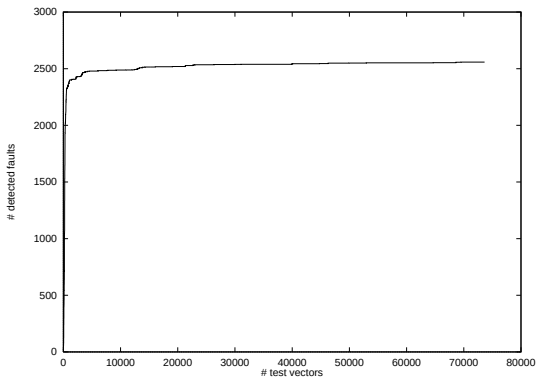

Figure 5: Number of simulated test vectors versus number of detected faults for s38584.

In Table 3, in columns 2-4 we show the number of detected faults, undetectable faults, and undetected faults after the combinational ATPG/sequential fault simulation step. The sum of these three equals $|f_{hard}|$ which is given in Table 2, column 3. Column 6 contains the number of increased controllability/observability circuits created for sequential ATPG. The first number is the initial number of circuits created from groups 1-3. The second number is the number of circuits created to target the final faults (f_{final}). In our experiment, we set the user parameters $LARGE\_DIST = \max(0.6 * maxsize, 50)$, $MED\_DIST = \max(0.25 * maxsize, 25)$, and $DIST = \max(0.15 * maxsize, 20)$, where $maxsize$ is the length of the longest scan chain in the circuit. In columns 7-9, we show the number of detected faults, undetectable faults, and undetected faults after sequential ATPG. The sum of these three equals column 4. As can be seen by the total figures, after the combinational ATPG/sequential fault simulation the number of undetected faults is reduced to $314/197440 = 0.159\%$ of the total number of faults and $314/(42596 + 6291) = 0.642\%$ of the number of faults affecting the scan chain. After sequential ATPG, the number of undetected faults is reduced to $11/197440 = 0.006\%$ of the total number of faults and $11/(42596 + 6291) = 0.022\%$ of the number of faults affecting the scan chain.

If we look at the CPU times given in Table 2 column 4 and Table 3 columns 5 and 10, we can see that the majority of CPU time is spent in the combinational ATPG/sequential fault simulation step. In fact, it is the sequential fault simulation which is using most of the CPU time. This however can easily be reduced by reducing the size of the test vector set. In our experiments, it was observed that the large majority of detected faults are detected by the beginning part of the test sequence, thus the test set can be reduced with only a small increase in the number of

Table 3: Detecting the faults in f_{hard}.

	Combinational ATPG / Seq Fault Sim				*Sequential ATPG*				
name	*#det*	*#undetectable*	*#undetected*	*CPU*	*#circ*	*#det*	*#undetectable*	*#undetected*	*CPU*
s1196	15	6	0	1s	N/A	N/A	N/A	N/A	N/A
s1238	38	1	0	1s	N/A	N/A	N/A	N/A	N/A
s1423	96	29	13	15s	3,0	0	13	0	7s
s1488	61	2	1	1s	1,0	0	1	0	1s
s1494	29	4	9	1s	1,0	9	0	0	1s
s5378	57	5	10	31s	6,3	2	7	1	1245s
s9234	227	3	27	81s	8,2	2	21	4	3724s
s13207	473	15	26	467s	8,0	4	22	0	72s
s15850	531	12	56	2612s	10,1	0	56	0	380s
s35932	799	0	80	26174s	13,0	80	0	0	5553s
s38417	691	46	54	10275s	15,2	1	47	6	3516s
s38584	2558	279	38	51491s	13,2	5	33	0	1054s
total	5575	402	314	91150s	78,10	103	200	11	15553s

undetected faults. For example, the test set created for the circuit s38584 has 73641 vectors. However as shown in Figure 5, most of the detect faults are detected by the first 30000 vectors. Thus it is possible to reduce the size of the test set with only a small increase in the number of undetected faults. These undetected faults then become input to the sequential ATPG step.

7 Conclusion

We have presented an effective method of testing scan paths through functional logic. This is critical because the subsequent testing procedures assume that the scan chain is fault-free. Our method requires that the circuit be placed in scan mode, and the nets in the scan chain are identified. Then a three step screening process is used to detect the faults that affect the scan chain. We implemented our functional scan chain testing method to test functional scan paths created by TPI. In our experimental results, the number of undetected faults remaining is only 0.006% of the total number of faults, or 0.022% of the faults affecting the scan chain.

8 Acknowledgments

This work was partially supported by the National Science Foundation under grant MIP 9419119, and partially by Fujitsu through the California MICRO program.

References

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design*. CS Press, New York, 1990.

[2] L. Avra. Orthogonal built-in self-test. In *COMPCON Digest of Papers*, pages 452-457, Spr 1992.

[3] K.-T. Cheng and V. D. Agrawal. A partial scan method for sequential circuits with feedback. In *IEEE Trans. on Computers*, volume 39, pages 544-548, Apr. 1990.

[4] W.-T. Cheng. The BACK algorithm for sequential test generation. In *Int'l Conf. on Computer Design*, pages 66-69, 1988.

[5] D. H. Lee and S. M. Reddy. On determining scan flip-flops in partial-scan designs. In *IEEE Int'l Conf. on CAD*, pages 322-325, 1990.

[6] C.-C. Lin, M. T.-C. Lee, M. Marek-Sadowska, and K.-C. Chen. Cost-free scan: a low-overhead scan path design methodology. In *IEEE Int'l Conf. on CAD*, pages 528-533, 1995.

[7] C.-C. Lin, M. Marek-Sadowska, K.-T. Cheng, and M. T.-C. Lee. Test point insertion: scan paths through combinational logic. In *Design Automation Conf.*, pages 268-273, 1996.

[8] R. B. Norwood and E. J. McCluskey. Orthogonal scan: Low overhead scan for data paths. In *Int'l Test Conf.*, pages 659-668, 1996.

[9] R. B. Norwood and E. J. McCluskey. Synthesis-for-scan and scan chain ordering. In *IEEE VLSI Test Symposium*, pages 87-92, 1996.

[10] E. M. Sentovich, K. J. Singh, et al. SIS: A system for sequential circuit synthesis. Report M92/41, University of California, Berkeley, 1992.

[11] M. J. Y. Williams and J. B. Angell. Enhancing testability of large scale integrated circuits via test points and additional logic. In *IEEE Trans. on Computers*, volume C-22, pages 46-60, Jan. 1973.