N-Detect TDF ATPG and Compression

Nick, Utkarsh, Benjamin
Group -10
VLSI Testing Final Project Report

I. Problem Description

In this final project we are required to perform the N-detect TDF (Transition Delay Fault) ATPG for Launch-on-Shift Mode. ATPG results are ranked in the order of three factors: fault coverage (for N=8) will be the priority followed by test length and run time. All the cases should be finished in 10 minutes. We also need to compress our test patterns because the memory limitation of the Automatic Test Equipment (ATE). We have two kinds of compression, one is Static Test Compression (STC) and another is Dynamic Test Compression (DTC). Also we cannot use the complete dictionary to do the test compaction due to the customer's memory limitation.

II. Past Research

Test generation for stuck-at faults in combinational circuits tried to generate a minimal test set to cover all testable faults in as less as possible CPU time. In the earlier PODEM algorithms, the recursive learning test generation scheme solved CPU time and complete coverage problems very well[1].

But Delay testing can detect performance-related defects that might not be discovered by single stuck-at fault tests. Detecting a path delay fault or a transition fault requires the application of a vector pair to the combinational portion of the circuit under test. In section 3, we have proposed the effective algorithm for synchronous sequential circuits to generate delay tests and fault simulate them[1].

In this project the Transition fault model for combinational circuits has been discussed. There are two transition faults associated with each signal: a slow-to-rise fault and a slow-to-fall fault. A slow-to-rise (slow-to-fall) transition fault is logical model for a defect that delays a rising (falling) transition[2].

The test data generated for detecting delay faults are usually much greater than that of stuck-at faults. Test compaction attempts to reduce the number of test vectors and the size of the test data, which is one of the limiting factors for delay fault testing. Test compaction techniques can be classified into two general categories: 1) static test compaction and 2) dynamic test compaction[1].

III. Proposed Technique

A. Number of Iterations for N-Detect

We performed N-detect 4 times to improve fault coverage.

B. Launch-on-shift

Given a fault f,

1) Apply ATPG program on f to generate test pattern V2.
2) Shift V2 in 1 bit, and try to fill in 0 or 1 to activate fault.
3) If any case success, then output(V1,V2) in LOS form. Else output failed to detect this fault.

C. N-Detect ATPG

The algorithm used for n-detect ATPG[3] is as follows:

1) Perform single-detect fault simulation with single-detect pattern set T_1 for all faults.
2) Save all faults detected by single-detect fault simulation with pattern set T_1(TFMD).
3) Set the number of detections N.
4) For K=1 to N-1
   - Perform multiple-detect fault simulation with pattern sets T_1 to T_K for TFMD faults.
   - Save faults detected K times (Fk).
   - Target faults Fk and perform single-detect ATPG to increase the number of detections by one.
   - Save the patterns to T_{k+1}.
5) Perform multiple-detect fault simulation with pattern sets T_1 to T_N for all faults to obtain multiple-detect fault coverage profile.

D. Static Test Compression

1) Reverse the test patterns order.
2) Do fault simulation check whether the pattern still useful or not. If no, then remove the pattern.
3) Some redundant patterns have already been deleted.
4) Sort the test patterns depending the number of faults they can detect.
5) Repeat step 2, the pattern which detects less fault maybe is the necessary pattern.
6) Reverse the test patterns order.
7) Repeat step 2, the pattern which detects more fault maybe is the dominating pattern.
8) Arrange the patterns in random order and repeat step 2. Until the random order have remove enough redundant patterns.
9) Print the pattern and show the result.

IV. Dynamic Test Compression

Following is our idea for DTC:
Another podem function(called second_podem) is written in which the unknown value of test pattern is filled with random 0 or 1.

1) Check all the primary output wires.
2) If unknown, then check fault on the same wire.
3) If fault detected then perform second_podem(). Else check next output wire.
4) After checking all output wire for unknown, do random fill for the test pattern.

V. Experimental Results

TABLE I
N-Detect=1 For Single Iteration

Circuit Number	Test Length w/o compression	Fault Coverage(%)	Run time
C17	11	88.23	0.0
C432	30	9.81	0.2
C499	151	89.12	1.2
C880	126	45.19	1.3
C1355	116	37.30	3.6
C2670	340	89.44	7.7
C3540	180	21.92	40.5
C6288	143	97.19	5.1
C7552	657	96.20	38.1

TABLE II
N-Detect=8 For Single Iteration

Circuit Number	Test Length w/o compression	Fault Coverage	Run time
C17	72	73.52	0.0
C432	106	7.92	0.3
C499	494	83.89	2.1
C880	379	42.53	1.8
C1355	441	36.97	4.8
C2670	1001	86.62	12.3
C3540	522	20.61	43.5
C6288	319	96.61	13.9
C7552	1654	94.44	67.6

TABLE III
N-Detect=1 After Four Iteration

circuit number	Test length w/o compression	fault coverage(%)	run time(s)	Test length w/ compression	fault coverage(%)	run time(s)	Test length reduction(%)
C17	11	88.23	0.0	5	88.23	0.0	54.54
C432	30	9.81	0.1	15	9.81	0.2	50
C499	151	89.12	1.1	68	89.12	1.4	54.96
C880	126	45.19	1.3	57	45.19	1.5	54.76
C1355	116	37.30	3.6	57	37.30	3.8	50.86
C2670	340	89.44	7.6	144	89.44	9.2	57.64
C3540	180	21.92	39.5	74	21.92	40.7	58.88
C6288	143	97.19	5.1	69	97.19	8.4	51.74
C7552	657	96.20	38.4	274	96.20	46.7	58.29

TABLE IV
N-Detect=8 After Four Iteration

circuit number	Test length w/o compression	fault coverage(%)	run time(s)	Test length w/ compression	fault coverage(%)	run time(s)	Test length reduction(%)
C17	48	82.35	0.0	45	82.35	0.0	6.25
C432	178	11.17	0.7	172	11.17	1.0	3.37
C499	527	85.14	5.1	488	85.14	6.6	7.4
C880	492	44.81	5.8	408	44.81	6.8	17
C1355	441	36.97	14.6	431	36.97	16.3	2.26
C2670	1207	89.17	32.5	1083	89.17	42.0	10.27
C3540	614	21.47	170.2	538	21.47	167.0	12.37
C6288	323	96.65	22.7	291	96.65	44.2	9.9
C7552	2252	96.28	171.4	1909	96.28	211.4	15.23

- $Test\ Length\ Reduction = (TL_{\text{w/ocompress}} - TL_{\text{w/compress}}) / TL_{\text{w/ocompress}} X 100\%$

VI. Obervation and Comparison

In the paragraph, we want to prove our decision and design method is correct and more powerful than original one. That is, we are going to use the comparison of experiment data to explain our ideas and to show the improvement.

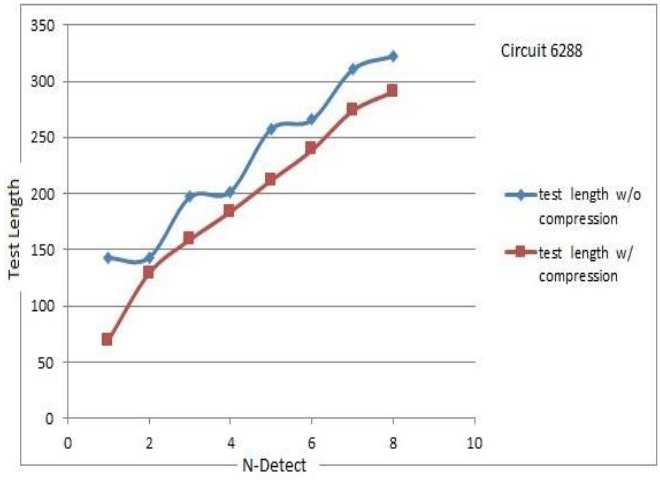

Fig. 1. Graph showing Test Length for circuit 6288

In Fig. 1, we compare the test length of C6288 circuit under compression or not. We can observe that through N-detected = $1 \sim 8$, the growth of test pattern is slower and more stable with compression. This shows the power of our static test compression.

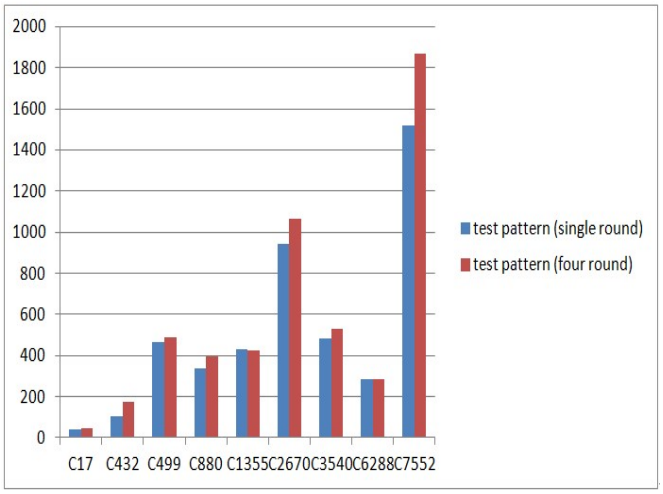

Fig. 2. Graph showing Test Length for each circuit in Single/Four Iteration(s) for 8-det after compression

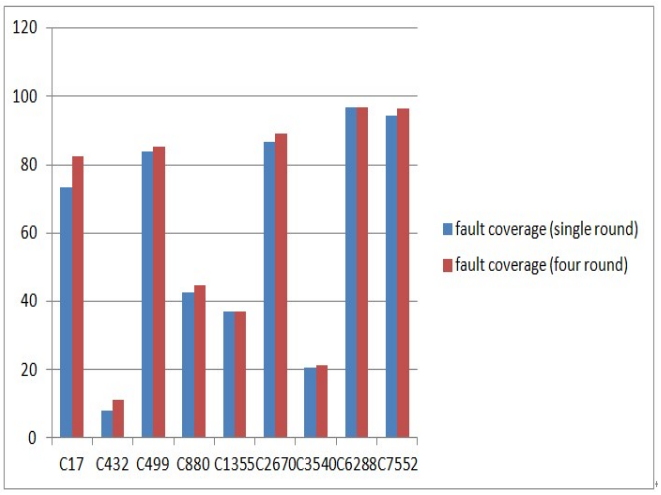

Fig. 3. Graph showing Fault Coverage for each circuit in Single/Four Iteration(s) for 8-det after compression

In Fig 2 and 3, we compare the test patterns/fault coverage of each circuit under single or four rounds. The growth of test patterns from single to four iterations is less than 10% basically. However, the growth of fault coverage from single to four iterations is more than 2% is most cases. This reveals why we decide to use four iterations n-det.

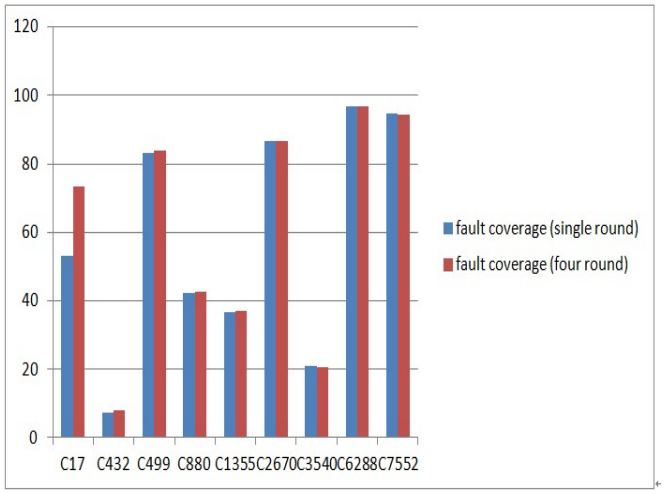

Fig. 4. Graph showing Fault Coverage for each circuit under reversing fault list or not

In Fig 4, we compare the fault coverage of each circuit under reversing fault list or not (single round). The experiment certain that reverse fault list or not would change the fault coverage, especially on the small circuits.

However, the influence of reversing fault list would decreasing on larger circuit, since most of PO have large overlapping input. That is, the order of fault is not so important in large circuits.

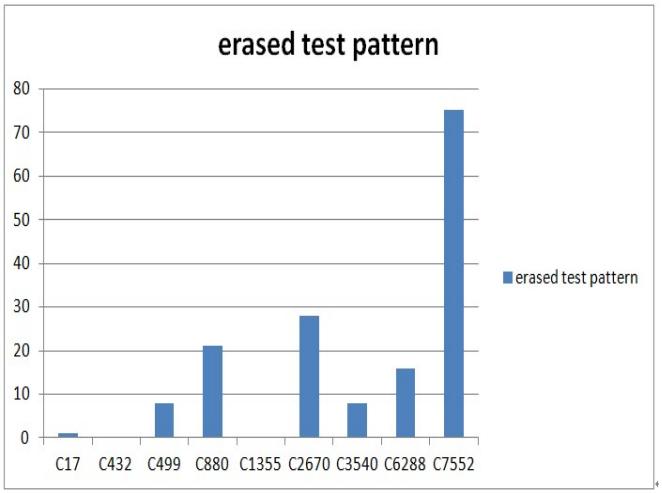

Fig. 5. Graph showing Erased Test Pattern for each circuit after reversing order

In Fig 5, we success to erase test patterns after reversing order. Originally, static test compression would stop right after reversing order with dropping test patterns. However, we decide to randomly choose a test pattern and try to erase it.

Finally, we success to erase more test patterns after reversing order without decreasing fault coverage. So, our

static test compression is more hardworking and more useful.

VII. List of our new ideas

- First New Idea in Ndet-ATPG. For the LOS-ATPG, our way is using PODEM to afford test pattern V_2, and then shift 1 bit with 0 or 1, tending to active transition delay fault. If we success, then this new test pattern is V_1.
- In N-detected ATPG, we copy the information of faults right after tdf-simulation with TFMD. And, return the information to faults right after finishing ATPG with F_k. In this way, we only need to do tdf-simulation on T_k, but not T_1 to T_k in each iteration.
- For the N-detected ATPG, if the run-time is short enought, we can consider to run double rounds, or even four rounds. We only have to use double iterator on for loop to implement this idea. Furthermore, it power is shown in Fig 2 and 3.
- For Static test compression part, we sort the test patterns by the number of faults they can detect. So we can find the redundant patterns more efficiently.
- At the last step of static test compression part, we randomly order the the patterns and do fault simulation. So, we can find some redundant patterns that we haven't found previously. Most important point is that we use the number of pattern removing to decided end for this step or not.

VIII. Discussion

- TableI shows Fault Coverage, Test Length, and Run Time for N-Detect=1 for the given circuits for single iteration.
- TableII shows Fault Coverage, Test Length, and Run Time for N-Detect=8 for the given circuits for single iteration.
- From TableI and III we can see that there is not much change in the fault coverage for N-Detect= 1 between single and four iterations without compression.
- From TableII and IV we can see that there is small increase in fault coverage and decrease in test length for N-Detect= 8 between single iteration and four iterations without compression.
- TableIII shows Fault Coverage, Test Length, and Run Time for N-Detect=1 for the given circuits before and after the compression for four iterations.
- TableIV shows Fault Coverage, Test Length, and Run Time for N-Detect=8 for the given circuits before and after the compression for four iterations.
- Figure1 shows the Test Length on y-axis and N-Detect on x-axis for circuit 6288 with and without compression.
- From the graph we infer two conclusions:
  1) When N-Detect=1, without compression Test Length= 143 and Fault Coverage= 97.19%. But with compression, Test Length= 69 and Fault Coverage= 97.19%.
  2) When N-Detect=1, without compression Test Length= 323 and Fault Coverage= 96.65%. But with compression, Test Length= 291 and Fault Coverage= 96.65%.
- Our proposed technique has successfully reduced the test length.
- We tried DTC but couldn't improve much results.

IX. References

[1] D.Xiang, W. Sui, B. Yin and K. Cheng, "Com pact Test Generation With an Influence Input Measure for Launch-On-Capture Transition Fault Testing," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 22, no. 9, pp. 1968-1979, Sept. 2014, doi: 10.1109/TVLSI.2013.2280170.

[2] Kwang-Ting Cheng, "Transition fault testing for sequential circuits," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 12, no. 12, pp. 1971-1983, Dec. 1993, doi: 10.1109/43.251160.

[3] B. Benware et al., "Impact of multiple-detect test patterns on product quality," International Test Conference, 2003. Proceedings. ITC 2003., Charlotte, NC, USA, 2003, pp. 1031-1040, doi: 10.1109/TEST.2003.1271091.

X. Contribution

1) Nick: N-Detect ATPG
2) Utkarsh: DTC idea and Report
3) Benjamin: STC

XI. Result Table

circuit number	Test length w/o compression	fault coverage	run time	Test length w/ compression	fault coverage	run time	Test length reduction
C432	178	11.17	0.7	172	11.17	1.0	3.48%
C499	527	85.14	5.1	488	85.14	6.6	7.99%
C880	492	44.81	5.8	408	44.81	6.8	20.52%
C1355	441	36.97	14.6	431	36.97	16.3	2.32%
C2670	1207	89.17	32.5	1083	89.17	42.0	11.44%
C3540	614	21.47	170.2	538	21.47	167.0	14.12%
C6288	323	96.65	22.7	291	96.65	44.2	10.99%
C7552	2252	96.28	171.4	1909	96.28	211.4	17.96%

Fig. 6. Resulting Table