N-Detect TDF ATPG and Compression

Chia-Jen Nieh[1], Zong-Han Wu[1], and Chia-Chun Chien[1]
[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106319, Taiwan
r12943013@ntu.edu.tw, zhwu@eda.ee.ntu.edu.tw, r12943110@ntu.edu.tw

***Abstract*—This project demonstrates the N-detect automatic test pattern generation (ATPG) for the transition delay fault (TDF) model with dynamic and static test compression (DTC and STC).**

***Index Terms*—N-detect, ATPG, TDF, DTC, STC**

I. Problem Description

The **N-detect TDF ATPG and Compression problem** is formulated as follows:

Given a circuit with primary inputs (PIs), primary outputs (POs), logic gates, connection between them, and a scan chain connecting the PIs and POs, our goal is to generate a list of test patterns supporting the launch-on-shift (LOS)-mode testing for the TDF model within a reasonable runtime limit. The objective is to maximize the fault coverage and to minimize the test length (the number of test patterns).

II. Past Research

Since N-detect ATPG and test compression are challenging but necessary for high test quality and memory-efficient testing, many previous works were related to N-detect ATPG and test compression. Benware *et al.* [2] emphasized how N-detect ATPG can improve product quality. Hamzaoglu, Patel [3], Xiang *et al.* [4], Remersaro *et al.* [5], and Kumar *et al.* [6] highlighted the advantages of performing tests with smaller, compact test sets. On the other hand, scan designs greatly improve the testability of chips. Lin *et al.* [7] studied on test compression for chips with scan designs.

III. Proposed Techniques

A. N-detect TDF ATPG

While the algorithm of 1-det TDF ATPG is taught in class, N-det TDF ATPG is not. We need to extend the algorithm for 1-det to N-det.

At first, we thought an unfilled pattern should be counted as one pattern. For example, 'xx010' is counted only once. The reason behind is the fault effect would be triggered and propagated from the last three PI assignments, i.e., '010' and the first two PIs will not contribute to the propagation and thus have no meaning to be fully filled by 0 and 1 for four times. However, if we counted following this rule, the fault coverage of 2-det TDF ATPG for c17 would have only 45%, which was obviously wrong. So, if we do not turn on compression, this simple example pattern will count 4 times and ATPG will continuously find unfilled patterns if N is not achieved. On the other hand, if compression is turned on, ATPG should output the unfilled pattern for DTC to have more compact and meaningful patterns. It's worth noting that the number of unfilled patterns of ATPG generated for DTC should be set empirically. Besides, consider *only one* unfilled pattern 'xx010' is found, and N = 3, then the benefit of doing DTC on this unfilled pattern is to save the only one needless pattern. In such case, we do not need to do DTC on this unfilled pattern and just output the fully filled 4 patterns to STC.

It's interesting to note that PODEM is not optimal for N-det scheme, and the reason behind this is the backtrack mechanism. Consider an experiment, we set N = 5 and test ATPG on c17. The fault coverage evaluated by ATPG is 61.76%. The patterns are then collected to do TDF N-det simulation. Surprisingly, the fault coverage evaluated by simulation is 94.12%. We highlight a conclusion that PODEM is not optimal for N-det scheme here because if one of the PIs in the decision tree is backtracked to 0 or 1 and an unfilled pattern is found afterwards, this PI might have no effect on fault propagation and can be set to don't-care. In other words, the pattern could be potentially counted by 2 times, 4 times, etc, Thus PODEM is not optimal for N-det scheme and the fault coverage is underestimated in the ATPG stage. This issue induces a change that we need to output the patterns even if ATPG thinks a fault can't be detect by N times. Because in the STC stage, the fault might be accidentally detected by patterns generated from other faults.

The mechanism for filled patterns generating when compression is not turned on is described below. We have two attempts for filled patterns generating. In the first attempt, we try to fully fill the unfilled patterns until N-det, and the last unfilled pattern will produce more accumulated patterns than N. In this attempt, if the number of don't-care PIs is large, the memory will be run out of. For example, if the unfilled pattern is 'xxxxxxxxxx1' and we pursue 5-det, in this attempt, $2^{10} = 1024$ filled patterns will be generated. It's straightforward when the number of PIs is quite large, the memory will be run out of very easily. So, we modify this attempt to the second attempt. The second attempt follow a rule, if the filled PIs have made the number of patterns enough, the rest PIs will be filled randomly. For the same example, since we need 5-det, 'xxxxxxxxxx1' is filled to '0xxxxxxxxx1' and '1xxxxxxxxx1'. Then '0xxxxxxxxx1' and '1xxxxxxxxx1' are filled to '00xxxxxxxx1', '10xxxxxxxx1', '01xxxxxxxx1' and '11xxxxxxxx1'. Since 5-det is not achieved, '00xxxxxxxx1' is filled to '000xxxxxxx1' and '001xxxxxxx1'. Until now, 5-det is achieved, and then

TABLE I
RESULTS OF ATPG ($N = 1$)

Circuit	TL	FC (%)	RT (sec)	TL_c	FC_c (%)	RT_c (sec)	$(TL - TL_c)/TL \times 100\%$ (%)
c432	129	11.62	0.1	31	11.62	0.1	0.1
c499	1480	94.27	1.8	112	93.81	1.5	1.8
c880	923	50.00	0.9	93	50.24	0.7	0.9
c1355	1000	38.41	2.8	79	38.41	2.2	2.8
c2670	4641	92.84	8.0	219	92.81	6.3	8.0
c3540	1496	23.03	15.6	138	23.14	10.9	15.6
c6288	10072	97.66	255.5				
c7552	12676	98.07	79.2	396	98.03	67.8	79.2
Average							

TABLE II
RESULTS OF ATPG ($N = 8$)

Circuit	TL	FC (%)	RT (sec)	TL_c	FC_c (%)	RT_c (sec)	$(TL - TL_c)/TL \times 100\%$ (%)
c432	1032	11.62	0.1	79	11.44	0.1	0.1
c499	11832	94.73	2.5	271	93.89	1.6	2.5
c880	7384	50.38	1.1	1.1	1.1	1.1	1.1
c1355	6088	38.41	3.0	3.0	3.0	3.0	3.0
c2670	37128	93.11	10.6	10.6	10.6	10.6	10.6
c3540	11968	23.21	18.0	18.0	18.0	18.0	18.0
c6288	80184	97.66	300.0				
c7552	101408	98.14	239.0	239.0	239.0	239.0	239.0
Average							

we filled ’10xxxxxxxx1’, ’01xxxxxxxx1’, ’11xxxxxxxx1’, ’000xxxxxxx1’ and ’001xxxxxxx1’ in random to produce only 5 filled patterns to alleviate the memory usage and the run time.

B. *DTC: N-detect X-PODEM*

X-PODEM has been widely used in dynamic test compression due to its effectiveness and the compatibility to the original PODEM-based ATPG. Following the scheme in the lecture, we extend the idea to N-detect ATPG. In the N-detect scenario, a secondary fault can be found not only if an unknown PO can be backtraced, but also the compression itself cannot harm the N-detectability of primary fault, which we have to keep good track of.

Furthermore, in order to restrict run time to a an acceptable range, we also limit the iterations on finding patterns of the secondary fault. Specifically, we limit the 2 following things, (1)The continuous failure on trying different secondary faults cannot exceed a constant threshold(In our experiment we set 3). (2)The total number of tries cannot exceed a constant number(In our experiment we set to 10).

C. *N-detect First-Fit TC*

D. *N-detect X-Filled Reverse-Order STC*

The X-filled reverse-order STC is a popular method for reducing the test length due to its simplicity and effectiveness. In this project, we extend the X-filled reverse-order STC, making it support the N-detect version. After generating test patterns in III-A, III-B, we run TDF fault simulation for each pattern in the reverse order of the generation time. After simulating a test pattern, we drop the faults detected by the current and previous test patterns for N times before proceeding to the next test pattern. We eliminate a test pattern if it does not detect any fault in the undetected fault list.

Since the X-filled reverse-order STC is simple and well-studied during lectures, we do not detail its procedures in this final report. On the other hand, we highlight the extension to the N-detect version in the following paragraph. In the 1-detect version, a fault is dropped as long as it is detected by a pattern. However, in the N-detect ($N > 1$) version, we cannot drop the fault once detected by the current pattern. To ensure that each fault is detected for N times, we record the detected time for each fault in the undetected fault list during the reverse-order fault simulation. Once the detected time reaches N, the fault is dropped.

IV. EXPERIMENTAL RESULTS

We implemented our algorithms in the C++ programming language and conducted experiments on some circuits in the ISCAS’85 benchmark. All experiments were performed on a YYY-GHz XXXXXX(computer name) with ZZZGB memory. Since we introduce randomness in the pattern generation, it’s possible to get slightly different fault coveragetes as listed in the Table I and Table II.

V. DISCUSSION

A. *With/Without Compression*

- Test length comparison

B. *XXX*

$$a + b = \gamma \tag{1}$$

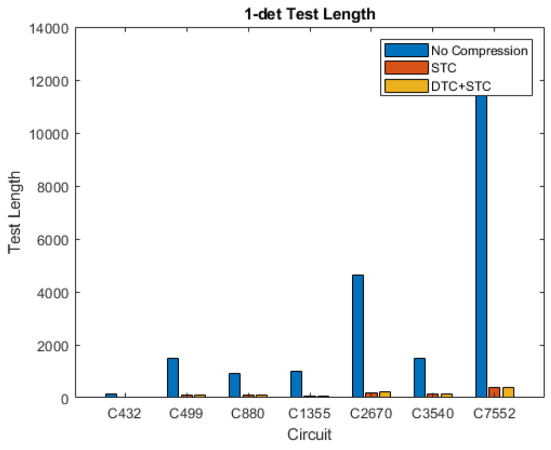

Fig. 1. The test length of 1-detect on each circuit benchmark.

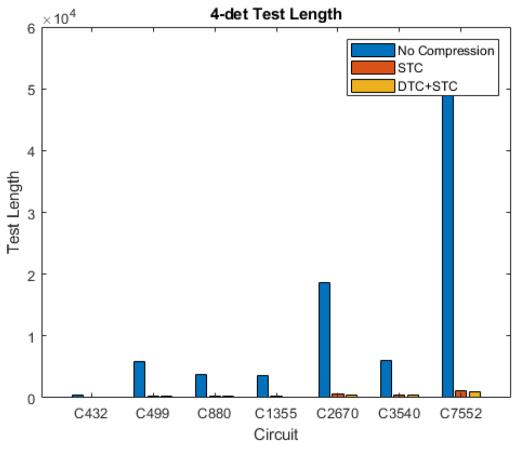

Fig. 4. The test length of 4-detect on each circuit benchmark.

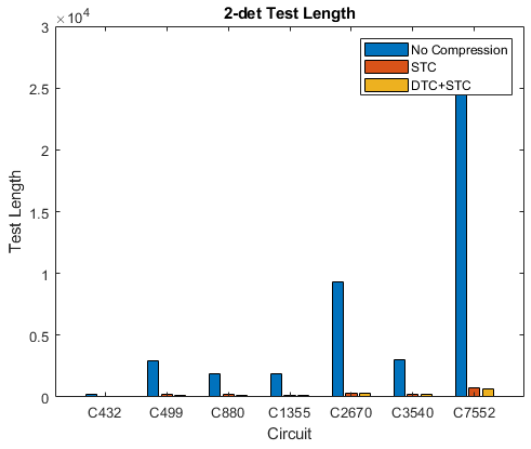

Fig. 2. The test length of 2-detect on each circuit benchmark.

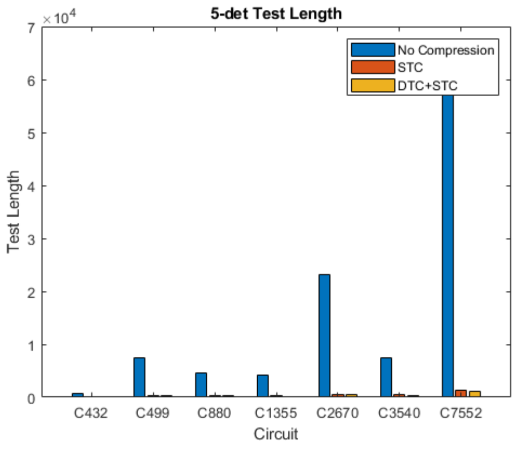

Fig. 5. The test length of 5-detect on each circuit benchmark.

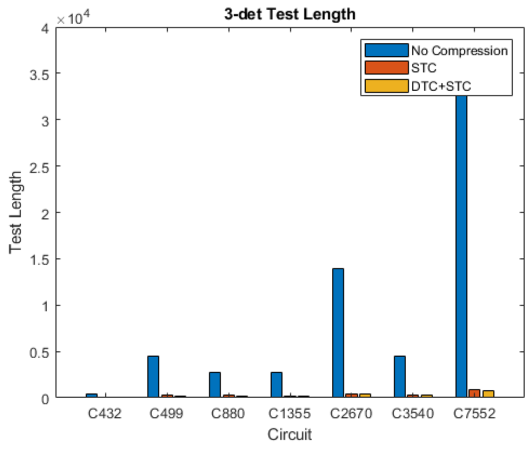

Fig. 3. The test length of 3-detect on each circuit benchmark.

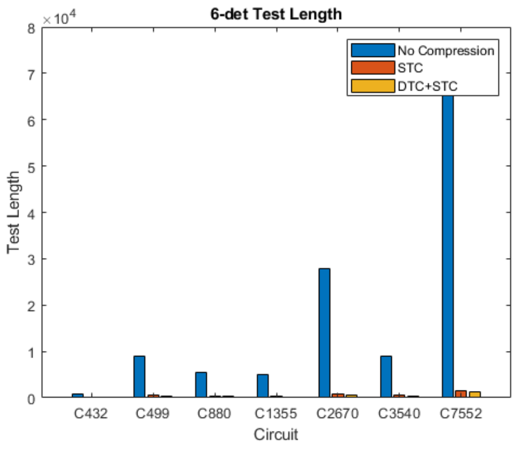

Fig. 6. The test length of 6-detect on each circuit benchmark.

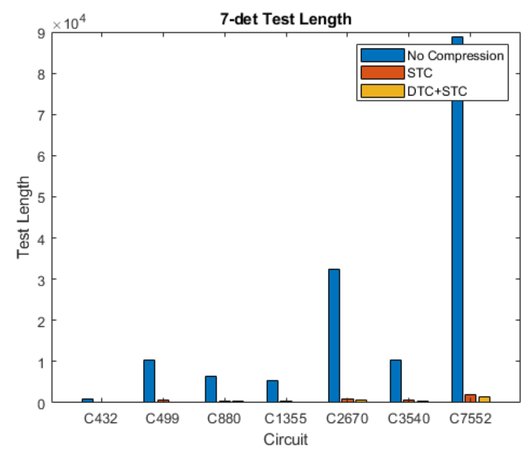

Fig. 7. The test length of 7-detect on each circuit benchmark.

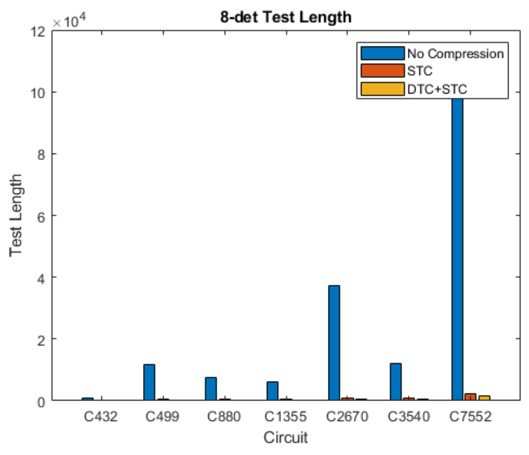

Fig. 8. The test length of 8-detect on each circuit benchmark.

C. XXX

VI. Contribution of Individual Team Members

A. Chia-Jen Nieh

N-det TDF ATPG, N-det TDF Simulation, Script (See the contribution of *joshuanieh* from the commits on Github [1])

B. Zong-Han Wu

Static test compression (mainly STC.cpp)

C. Chia-Chun Chien

Dynamic test compression (By modifying the function tdf2xpodem in tdfatpg.cpp)

References

[1] https://github.com/joshuanieh/Integrated_Circuit_Testing_Final_Project

[2] B. Benware , C. Schuermyer , S. Ranganathan , R. Madge , P. Krishnamurthy,” Impact of multipledetect test patterns on product quality, “IEEE Int’l Test Conference, 2003.

[3] I.Hamzaoglu, J.Patel, “Test set compaction algorithms for combinational circuits”, ICCAD 1998.

[4] Xiang, Dong, et al. ”Compact test generation with an Influence input measure for launch-on-capture transition fault testing, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 22.9 (2014).

[5] Remersaro, Santiago, et al. ”A scalable method for the generation of small test sets.” 2009 Design, Automation & Test in Europe Conference & Exhibition. IEEE, 2009.

[6] Kumar, Amit, et al. ”On the generation of compact test sets.” 2013 IEEE International Test Conference (ITC). IEEE, 2013.

[7] Lin, Xijiang, et al. ”On static test compaction and test pattern ordering for scan designs.” Proceedings International Test Conference 2001 (Cat. No. 01CH37260). IEEE, 2001.