EEE 5001 VLSI TESTING FINAL PROJECT REPORT : N-DETECT TDF ATPG AND COMPRESSION

Jian-Heng Liu, Zhong-Yu Sui, Hung-Ling Liu

National Taiwan University Taipei 10617, Taiwan

ABSTRACT

This project tackles the challenge of generating efficient N-detect transition delay fault (TDF) automatic test pattern generation (ATPG) while minimizing test costs and memory usage. Our approach employs PODEM, random simulation, and reachability analysis to identify effective test patterns for each fault. These fault-pattern pairs are ranked and optimized using PODEM-X to enhance fault coverage (FC) and pattern quality, thus reducing test length. Finally, static test compression (STC) is applied to further minimize memory usage. Our method achieves a superior balance of high FC and reduced test length, making it highly effective for modern VLSI testing.

***Index Terms*—** Testing, PODEM, transition delay fault, fault ranking, SCOAP, dynamic data compression, static data compression.

1. PROBLEM DESCRIPTION

N-detect test patterns have been shown to be an effective way to improve test quality [1]. However, N-detect test patterns are very long so the test cost is high. As the chief engineering of NTU ATPG systems, you are required to add a new function: N-detect transition delay fault ATPG (LOS mode only). You 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). This is a very competitive project so you are free to apply any innovative ideas, like [2] [3], to make your ATPG better than the other competitors.

2. PAST RESEARCH

In this session, we will introduce the algorithms that have been developed in previous outstanding research and are suitable for integration into our workflow.

2.1. Reachability and Detectability

Reachability [3] is how to control a wire. Different from SCOAP, it is represented by a set of value-PI pairs, thus it might tell us the PI values we need to specify to control a wire.

There are 0- and 1- control reachability, which tell us the PI assignment needed to make a wire 0 and 1. The computation is just like SCOAP, which unions the control reachability of needed FI, and picks the minimum sized one if multiple assignments on FI are available. For example, if W = AND(A,B), then

$$RC_1(w) = RC_1(A) \ \cup \ RC_1(B) \tag{1}$$

$$RC_0(w) = \begin{cases} RC_0(A) & \text{if} |RC_0(A)| < |RC_0(B)| \\ RC_0(B) & \text{else} \end{cases} \tag{2}$$

There is also observation reachability, which indicates the PI needed to make the signal of a wire propagate to some output. Similarly, it is the control reachability of other FI unions with observation reachability of FO. For example, if W = AND(A,B), then

$$RO(A) = RC_1(B) \bigcup RO(W) \tag{3}$$

If there is a fanout stem, just pick the one with the smallest size. To know the PI that can detect a transition fault with LOC, i.e. detectability, we thus have

$$det(w/STR) = RC_0(w^1) \cup RC_1(w^2) \cup RO(w^2) \tag{4}$$

$$det(w/STF) = RC_1(w^1) \cup RC_0(w^2) \cup RO(w^2) \tag{5}$$

where w^1, w^2 is the wire w in frame 1 and frame 2. Notice that LOS can also use the two-frame model, thus we can apply it to LOS, which is detailed in part 3.2.

2.2. PODEM

The PODEM algorithm [4] is a highly efficient test generation method. One of the key advantages of PODEM is its simplicity compared to the D-algorithm, while still being a complete algorithm capable of generating a test if one exists. PODEM uses heuristics to efficiently search through all possible primary input patterns, ensuring that a test is found if it exists, or confirming that no test is possible once the search space is exhausted.

2.3. PODEM-X

The PODEM-X algorithm [4] is an extension of the PODEM algorithm. It begins with the standard PODEM process, where a primary fault is selected and a test cube is generated. For each selected secondary fault, a new test cube is generated based on the initial test cube. If this generation fails, the algorithm moves on to the next secondary fault while keeping the previous faults in the list for future attempts.

By integrating this iterative process, PODEM-X significantly enhancing overall fault coverage and maintaining test efficiency.

2.4. Reverse Order Fault Simulation

[5] introduces Reverse Order Fault Simulation(ROFS), which allows for simulation of ATPG-generated patterns in reverse order to remove redundant patterns (those that cannot detect faults). This process reduces the number of patterns, achieving test set compression (TSC). Through ROFS, patterns can be compressed very quickly without the need for a dictionary.

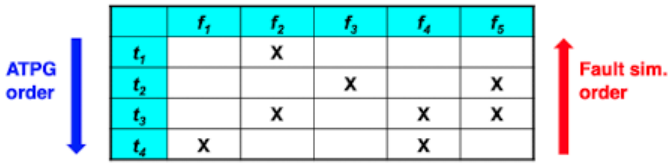

Fig. 1. Reverse Order Fault Simulation

3. PROPOSED TECHNIQUE

In this section, we introduce our novel approach that builds upon past research to further enhance the efficiency and effectiveness of N-detect TDF ATPG. Our proposed technique incorporates several innovative ideas and heuristics that set it apart from previous methods

3.1. Random Simulation

Before ATPG, we attempted to use random simulation to identify easily detected faults first. This allows subsequent PODEM to avoid generating patterns for each fault, thereby speeding up the process.

Using this method significantly reduces runtime but increases test length. Considering the evaluation criteria, we decided not to use random simulation. A detailed analysis is provided in the experimental results section.

3.2. SCOAP and Reachability

During PODEM, the order of choosing path to backtrace and propagate fault can affect efficiency a lot. Rather than the length of a path, we first choose to use controllability as the reference to make the choice, i.e. choose the path with a smaller value of controllability. We compute controllability simply by SCOAP, which is linear time to the size of the circuit.

We also try to use the value of reachability [3] instead of SCOAP. To compute the reachability of the LOS framework, we still use the two-frame model, with PI in frame 1 shifted and wired to frame 2, as shown in Fig. 2.

We only take the concept of reachability, the pattern compression part is still PODEMX, not the one mentioned in [3]. For pattern generation, we use the PI set of detectability as the pattern. If the set has conflict, i.e. indicate a PI signal to be both 0 and 1, we would go back and use PODEM to find the pattern.

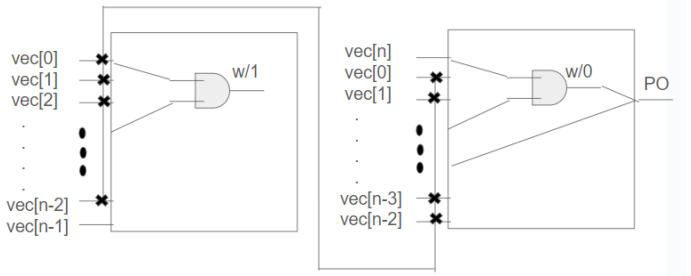

Fig. 2. Two-frame model of STF on wire w for LOS, we let ith PI in frame 1 wired to i+1th PI in frame 2, and thus we can compute just as in LOC

3.3. Ranking

After generating fault-pattern pairs, we calculate a rank for each pair to enhance the quality of our ATPG flow. This ranking process helps to sort the fault list, like Fig. 3, allowing us to focus on higher-quality patterns that improve fault coverage (FC) and reduce test length. We propose three ranking methods, each with distinct advantages and suitable for different cases:

3.3.1. Count of Don't Care Bits in the Pattern

Patterns with more don't care bits are more flexible, which can lead to better results during subsequent PODEM-X optimization. This flexibility allows for more effective pattern enhancement and fault coverage improvement.

3.3.2. Number of Faults Detected by the Pattern in TDF Simulation

Patterns that detect more faults can drop more faults from the list during the PODEM-X process, accelerating ATPG and reducing test length. This method prioritizes patterns that contribute to quicker and more comprehensive fault detection.

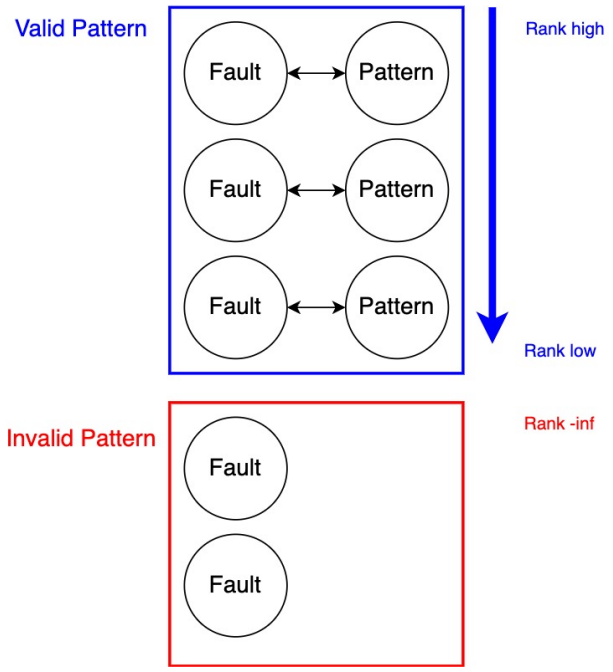

Fig. 3. Ranked fault list

3.3.3. Sum of SCOAP Values of Faults Detected by the Pattern

SCOAP values indicate the difficulty of detecting a fault. A higher sum of SCOAP values detected by a pattern suggests that the pattern is of higher quality. This heuristic prioritizes patterns that detect harder-to-find faults, potentially improving overall test effectiveness.

3.4. X-filled

During PODEM-X, patterns with unknown inputs are generated. We randomly assign 0s or 1s to the unknown inputs and use the same pattern to generate multiple patterns for simultaneous simulation to detect other faults. Using X-fill to drop faults can more efficiently utilize patterns generated by PODEM to achieve n-detect.

3.5. Static Data Compression

In the static data compression phase, we implemented based on [5] and also performed two heuristic preprocessing to speed up the process and improve the compression ratio.

3.5.1. Pattern Sorting

Assign a score to each pattern based on the number of faults it can detect. Sort the patterns in ascending order according to their scores.

3.5.2. Identify Important Faults

Since we already have the fault coverage (FC) generated by ATPG, we initially simulate to identify which faults are detected. We call these important faults. During data compression, we only need to focus on these important faults in the undetected fault list, which significantly speeds up the process.

After preprocessing, we not only used ROFS but also attempted to randomly shuffle the compressed patterns. We iteratively repeated the ROFS steps until there were no improvements for three consecutive iterations.

4. EXPERIMENTAL RESULTS

Table 1 is the best experimental result we obtained with ndet = 8.

We analyzed the results of random simulation and found that in some cases (e.g., c7552), random simulation can detect many faults in Fig. 4. However, further analysis revealed that the test length becomes very long in Fig. 5, and a significant amount of time will be spent on data compression in Fig. 6. We speculate that this is because the quality of patterns generated by random simulation is not good, requiring more patterns to achieve the same fault coverage. After weighing the pros and cons, we decided not to use random simulation.

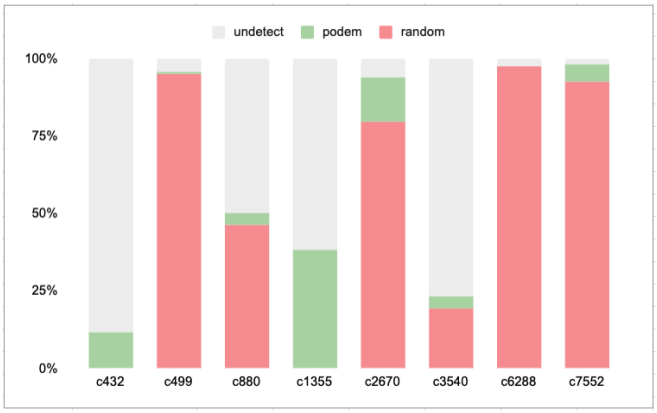

Fig. 4. Fault detected analysis

We also compare the result of using reachability or SCOAP value in PODEM. Both are 8-det ATPG with pattern sorting but no random simulation. The result is in the Fig 7. We can see that both have good fault coverage, but overall using reachability causes longer test length. Also, as reachability computation takes more time than SCOAP, we decide not to use reachability.

Table 1. Best experimental result

	c432	c499	c880	c1355	c2670	c3540	c6288	c7552	Average
FC	11.62	95.31	50.38	38.41	93.91	23.19	97.39	98.19	63.55
TL	161	603	408	450	838	556	515	1368	612
runtime	0.03	2.06	3.74	0.64	14.51	5.98	52.74	46.99	15.84

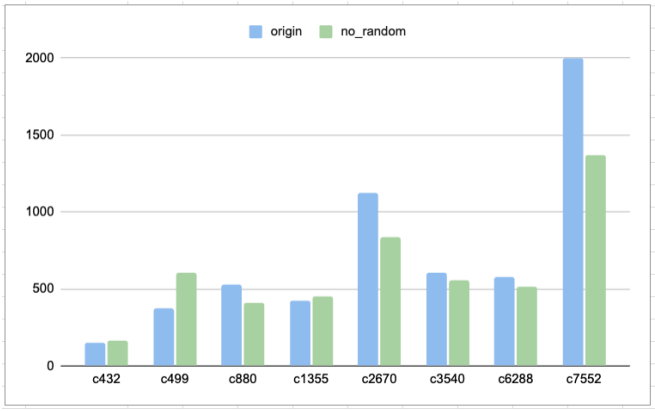

Fig. 5. Test length with/without random simulation

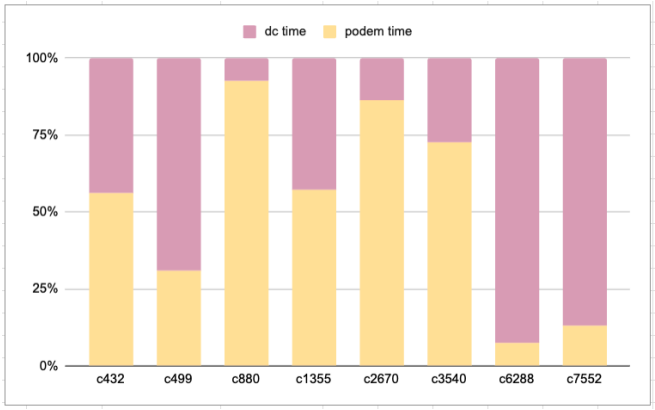

Fig. 6. Timing analysis with/without random simulation

Last but not least, we compared the test lengths generated for different cases with the number of detections (ndet) ranging from 1 to 8 in Table 2. We evaluated the test length for each case and calculated the average test length across all cases. This comparative analysis helps to understand the efficiency of our proposed technique in reducing test length as the number of detections increases.

5. DISCUSSION

5.1. Improved Backtrace Method

In our PODEX implementation, to detect the target fault, if the initial input assignment is incorrect, we often need to backtrack multiple times to find the correct pattern. This

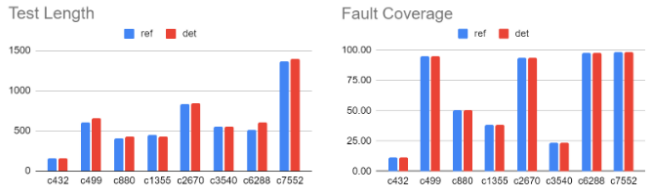

Fig. 7. The effect of Reachability

trade-off between fault coverage and runtime exists because the current SCOAP and logic level methods cannot quickly find the correct input assignment. A better backtrace method could potentially improve runtime and increase fault coverage.

5.2. Early Backtrack

Continuing from the first point, when we incorrectly assign a value to an input, detecting conflicts early and backtracking sooner can also shorten runtime.

5.3. Compression methods for n-detect

In this final project, we are implementing static data compression based solely on ROFS. Under the n-detect specification, there may be better compression methods to reduce more patterns and achieve a higher compression ratio.

6. KEY RESEARCH CONTRIBUTIONS

name	***works***	***percentage***
Jian-Heng	Reachability	33.3%
Zhong-Yu	SCOAP, PODEMX, Ranking	33.3%
Hung-Ling	Random simulation, Data compression	33.3%

Basically, everyone has participated in each work. Only the people who have a higher participation rate are listed.

7. CONTACT INFORMATION

- Jian-Heng: b08901032@ntu.edu.tw
- Zhong-Yu : johnsonsui3@gmail.com
- Hung-Ling: r12921045@ntu.edu.tw

Table 2. Test lengths comparison with different ndet

TL	c432	c499	c880	c1355	c2670	c3540	c6288	c7552	Average
ndet=1	20	84	57	54	120	75	84	181	84
ndet=2	42	169	100	109	232	141	147	345	161
ndet=3	60	237	159	165	335	210	238	50	182
ndet=4	80	306	222	220	433	299	318	708	323.25
ndet=5	120	438	328	329	633	444	449	1031	471.5
ndet=6	120	437	327	337	631	444	448	1033	472.125
ndet=7	138	485	386	382	742	488	514	1169	538
ndet=8	161	603	407	451	855	557	513	1371	614.75

8. REFERENCES

[1] B. Benware, C. Schuermyer, S. Ranganathan, R. Madge, P. Krishnamurthy, N. Tamarapalli, K.-H. Tsai, and J. Rajski, "Impact of multiple-detect test patterns on product quality," in *International Test Conference, 2003. Proceedings. ITC 2003.* IEEE Computer Society, 2003, pp. 1031–1031.

[2] I. Hamzaoglu and J. H. Patel, "Test set compaction algorithms for combinational circuits," in *Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, 1998, pp. 283–289.

[3] D. Xiang, W. Sui, B. Yin, and K.-T. Cheng, "Compact test generation with an influence input measure for launch-on-capture transition fault testing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 1968–1979, 2013.

[4] Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE transactions on Computers*, vol. 100, no. 3, pp. 215–222, 1981.

[5] M. Schulz, E. Trischler, and T. Sarfert, "Socrates: a highly efficient automatic test pattern generation system," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 126–137, 1988.