VLSI Testing Final Project

N-detect TDF ATPG and Compression

Kai-Hsiang Hu
B09901153

Chia-Cheng Wu
B09901098

Bo-Ru Shi
B09901081

***Abstract*—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. We are required to add a new function: N-detect transition delay fault ATPG (LOS mode only). 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).**

***Index Terms*—ATPG, trasition dalay fault, n-detect, test compression**

I. Problem Description

A set of circuits in the form of gate level Verilog files and a sample code for generating patterns for the detection of Single Stuck at Faults (SAF) was given and that the completion of functions which would allow for N-detect Automatic Test Pattern Generation (ATPG) of Transition Delay Faults (TDFs) was the main objective with 3 benchmarks for performance ranking: Fault Coverage (FC), Test Length (TL), and Run Time. The completed program should allow for user to input a command containing the circuit file with which TDF test patterns are generated and exported to a file as per user's input request. For example, the input command should resemble the following: $./atpg – tdfatpg - ndet8 - compression../sample_circuits/c17.ckt > ../tdf_patterns/c17.pat$ The token "-tdfatpg" is used to inform the program that TDF ATPG is desired. Tokens "-ndet 8" and "-compression" are optional which commands the program generate test pattern with an arbitrary (less than 8) N-detect amount and test length compression respectively.

II. Past Research

A. Fault Model: Transition Delay Fault

Transition Delay Fault (TDF) models are critical for detecting defects that affect the timing of integrated circuits. These faults occur when a circuit fails to transition from one logic state to another within the required time frame, leading to timing violations. The TDF model is essential for identifying slow-to-rise or slow-to-fall transitions, which can severely impact circuit performance. By focusing on these delay faults, we can ensure more robust and reliable integrated circuit designs.

B. Automatic Test Pattern Generation

Automatic Test Pattern Generation (ATPG) is a key process in testing integrated circuits, designed to generate test vectors that can effectively detect faults. ATPG tools automate the creation of these test patterns, reducing the time and effort required compared to manual methods. The goal is to achieve high fault coverage with minimal test patterns. Effective ATPG strategies, such as PODEM [2] and FAN-ATPG [3], are crucial for improving the quality and efficiency of circuit testing.

C. Test Pattern Compression

Dynamic Test Compression (DTC) involves the use of algorithms like PODEM-X [4] to reduce the number of test patterns while maintaining or improving fault coverage. This technique dynamically analyzes the circuit and test patterns to identify and eliminate redundancies, which can be further refined by static techniques.

Static Test Compression (STC), on the other hand, focuses on optimizing the test patterns before they are applied to the circuit. It complements dynamic compression methods by providing an final reduction in test pattern size.

III. Proposed Technique

A. TDF ATPG Algorithm

The Transition Delay Fault ATPG algorithm is designed to generate test patterns specifically for detecting TDFs. This algorithm first applies the Single Stuck-at Fault (SSF) model to an initial vector (V2) by PODEM algorithm [2] and then verifies a preceding vector (V1). In Launch-On-Shift (LOS) mode, direct control of V1 is not possible. Therefore, the algorithm shifts back an unknown (x) value and determines if there is a possible primary input (PI) assignment for V1. If no valid assignment is found, the algorithm backtracks to explore alternative solutions. This process ensures comprehensive testing for TDFs and enhances the overall fault coverage.

B. N-Detect

After generating a test pattern with our TDF ATPG algorithm, the N-Detect method records primary inputs (PIs) with unknown (X) values. These X values are then filled randomly, and the resulting patterns are used to simulate and record outputs for fault dropping. This approach allows for multiple detections of each fault, reducing the likelihood of undetected faults and increasing the robustness of the testing process.

C. Dynamic Test Compression

By applying BFS in PODEM-X, we can trace back from the outputs to identify secondary faults, allowing for more effective compression of test data. This approach helps in reducing the overall test time and cost, making it a valuable technique in modern integrated circuit testing.

D. Static Test Compression

Techniques such as reverse-order fault simulation are used to analyze and compress the test data statically. This method involves simulating faults in reverse order to identify unnecessary test patterns and eliminate them. Static test compression can significantly reduce the volume of test data, thus saving memory and speeding up the test process.

IV. Experimental Results

circuit number	Test length w/o compression	fault coverage	run time	Test length w/ compression	fault coverage	run time	Test length reduction
C432	36	11.62%	0.1	22	11.62%	0.2	38.89%
C499	124	89.25%	0.8	75	89.62%	1.3	39.52%
C880	173	50.00%	0.6	71	49.52%	0.9	58.96%
C1355	223	37.20%	3.5	54	37.27%	3.4	75.78%
C2670	250	91.89%	1.8	191	91.99%	2.3	23.60%
C3540	254	22.59%	17.5	83	22.45%	34.2	67.32%
C6288	78	97.31%	5.5	96	97.35%	15.1	-23.08%
C7552	508	97.67%	8.1	384	97.70%	13.8	24.41%
Average	205.75	62.19%	4.7375	122	62.19%	8.9	38.18%

Fig. 1. N-Detect 1 Fault Coverage, Test Length, and Run Time before and after compression

circuit number	Test length w/o compression	fault coverage	run time	Test length w/ compression	fault coverage	run time	Test length reduction
C432	272	11.62%	0	171	11.62%	0.1	37.13%
C499	1192	87.70%	0.4	624	88.95%	0.9	47.65%
C880	1464	49.71%	0.3	515	49.24%	0.7	64.82%
C1355	1936	36.79%	1.5	451	36.87%	1.9	76.70%
C2670	2096	91.78%	1.3	1328	91.66%	2.8	36.64%
C3540	2176	22.43%	7.4	629	22.35%	13.9	71.09%
C6288	696	97.47%	3.3	624	97.42%	8.6	10.34%
C7552	4168	97.72%	6.8	2533	97.68%	15.5	39.23%
Average	1750	61.90%	2.625	859.38	61.97%	5.55	50.89%

Fig. 2. N-Detect 8 Fault Coverage, Test Length, and Run Time before and after compression

As shown in Fig. 1. and Fig. 2., The effects of N detect on circuits have been minimal, obtaining similar fault coverage results for the same circuits, which in tern is at the expense of longer test length. However, it can be observed that compression has achieved a significant test length reduction of 38% and 50%. In the effort of compressing test length, the average run time has roughly doubled, indicating the trade off made between the two benchmarks. In terms of test time discrepancy between N-Detect 1 and N-Detect 8, it is the result of the programming running on different hardware; even so, it doesn't effect our comparison of run time.

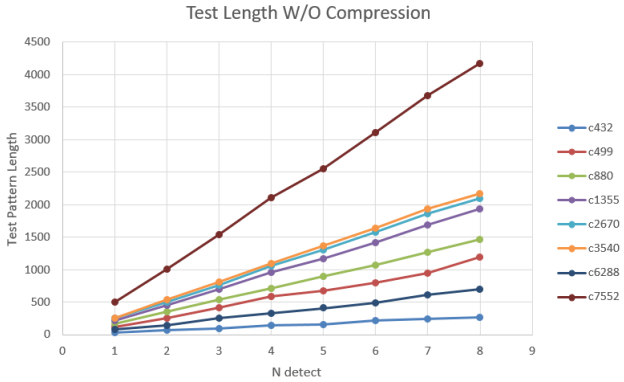

Fig. 3. Test Length relation with N-Detect without Compression

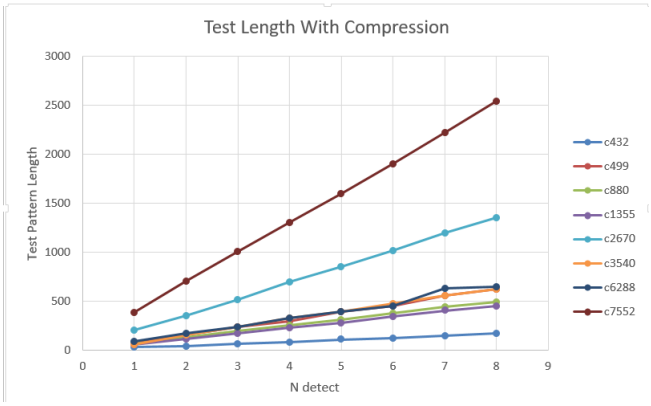

Fig. 4. Test Length relation with N-Detect with Compression

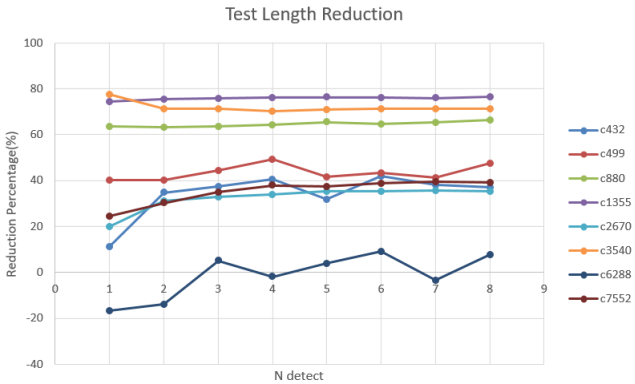

Fig. 5. Test Length Reduction correlation with N-Detect

From Fig. 3. and Fig. 4. it could be observed that test length grow linearly in relation to N-Detect. Moreover, the slope at which test length grows is dependent on the circuit under ATPG; that is, a circuit which requires more test patterns

by default would require even greater test length to achieve similar fault coverage results. Another phenomenon which could be identified is that compression would reduce the slope on nearly every N-Detect instance, along with Fig. 5. indicating that test length reduction performs similarly across all circuits, has shown the effectiveness of our compression techniques.

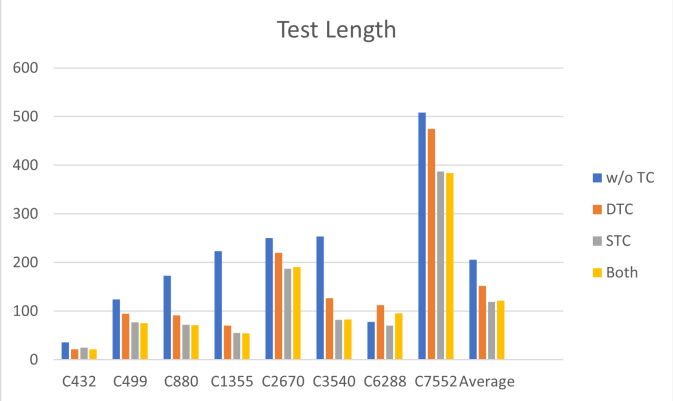

Fig. 6. Test Length relation with DTC and STC (N-Detect=1)

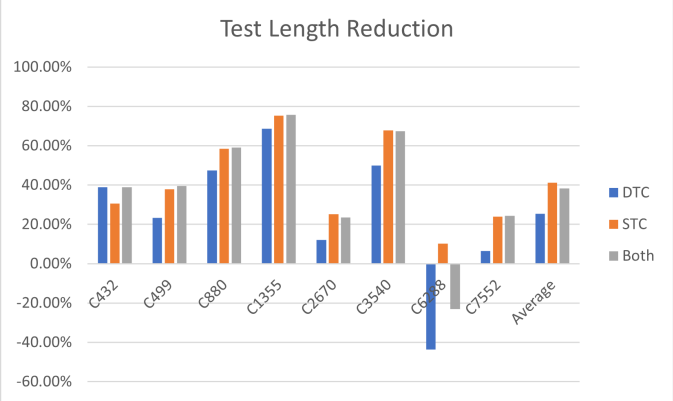

Fig. 7. Test Length reduction with DTC and STC (N-Detect=1)

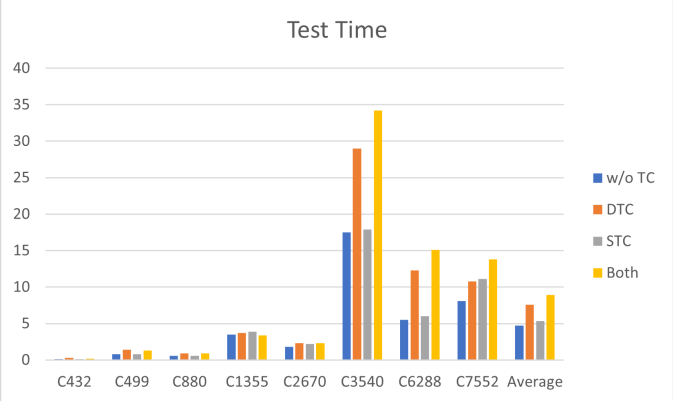

Fig. 8. Test Time with DTC and STC (N-Detect=1)

Fig. 5, 6, 7. compares the effectiveness of DTC and STC. It could be extrapolated from these graphs that both DTC and STC has a noticeable effect over test length for the majority of the circuits; and more specifically, STC offers the majority of the reduction. DTC, meanwhile, varies in performance and in c6288's case increases the test length; this should come as no surprise as DTC's test length reduction method highly depends on circuit structure and heuristics. As for the run time affected by STC and DTC, circuit structure appears to be the main factor affecting performance as c3540 takes the longest time. In addition to that, the run time for STC is also appears to be similar to the run time without compression for the majority of cases, indicating the longer run time for both STC and DTC applied should be attributed to the effects of DTC.

V. Discussion

To sum up, the program has managed to generate test patterns for the majority of easy-to-obtain TDFs and compresses test length in a tolerable amount of time; moreover, high N-Detect setting has little discrepancy in fault coverage compared to those generated without N-Detect. Certain ATPG and compression heuristics brought forth by other groups during the presentation day have been considered but ultimately weren't included in the final version due to high run time for less than a few percentage of fault coverage.

After applying Dynamic Test Compression (DTC), we observed a slight difference in fault coverage. In particular, one of the circuits (c6288) showed an increase in the number of test patterns required after DTC. Possible reasons for this anomaly could include the presence of complex fault conditions that are not easily compressible, leading to the necessity for additional patterns to achieve the same fault coverage. Additionally, certain faults might only be detectable through specific patterns that do not lend themselves well to compression, thereby increasing the overall pattern count post-compression. This underscores the need for balancing compression efficiency with fault coverage completeness.

VI. Contributions

B09901153 Kai-Hsiang Hu

- Basic TDF ATPG Function Coding
- N-Detect Function and Compression Function integration
- PowerPoint Presentation Making
- DTC STC Performance Data Organization and Charting
- Report Writing

B09901098 Chia-Cheng Wu

- N-Detect Function Coding
- N-Detect Function and Compression Function integration debugging
- PowerPoint Presentation Making
- N-Detect Performance Data Organization and Charting
- Report Writing

B09901081 Bo-Ru Shi

- DTC and STC function Coding
- Integration of N-Detect Function and Compression Function

- PowerPoint Presentation Making
- DTC STC Performance Data Organization and Charting
- Report Writing

REFERENCES

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

[2] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," IEEE Transactions on Computers, vol. 30, no. 03, pp.

[3] Fujiwara and Shimono, "On the Acceleration of Test Generation Algorithms," in IEEE Transactions on Computers, vol. C-32, no. 12, pp. 1137-1144, Dec. 1983.

[4] P. Goel and B. Rosales, "Podem-x: An automatic test generation system for vlsi logic structures," in 18th Design Automation Conference, 1981, pp. 260–268. 215-222, 1981