***N*-detect TDF ATPG and Compression**

**Introduction:**

*N*-detect test patterns have been shown to be an effective way to improve test quality [Benware 03]. 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 [Hamzaoglu 98] [Xiang 14], to make your ATPG *better than the other competitors*.

**Required Commands:**

In this project, we need you to create some commands, so we can choose different modes. First, for the *TDF ATPG* mode, you should build the flag “-tdfatpg”, then we can simply type the following command to operate our TDF ATPG.

./atpg –tdfatpg ../sample\_circuits/c17.ckt > ../tdf\_patterns/c17.pat

Second, we need a flag “-compression” to indicate that we will do the compression; otherwise, we will not.

./atpg –tdfatpg –compression ../sample\_circuits/c17.ckt > ../tdf\_patterns/c17.pat

Third, we need a augment “-ndet number” for the number *N*. The flag (*e.g. -ndet*) is followed by the number of detection (*e.g. number=N*). For example, if we want to specify 8 detection, we can simply type the following command.

./atpg –tdfatpg –ndet 8 ../sample\_circuits/c17.ckt > ../tdf\_patterns/c17.pat

Of course, the above two commands can be used together like this:

./atpg –tdfatpg –ndet 8 –compression ../sample\_circuits/c17.ckt > ../tdf\_patterns/c17.pat

We still use **the same circuits as our PA#3** with the one single scan chain that includes all PI and PO. So you can use the golden fault simulation of PA#3 to grade your fault coverage.

**Assignments:**

You can add your flags in file *tpgmain.cpp*, and write your ATPG code in file *tdfatpg.cpp*, which should be created by yourself. It’s free for you to modify other files, but you should clearly write down which part you modified in your report. You can find some references about test compression at the end of this document. Notice that you can NOT use the complete dictionary to do the test compaction due to customer’s memory limitation.

1) (Mandatory) Please fill in the following table with *N*=1 and *N*=8.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| circuit number | Test length w/o compression | fault coverage | run time | Test length w/ compression | fault coverage | run time | Test length reduction |
| C432 |  |  |  |  |  |  |  |
| C499 |  |  |  |  |  |  |  |
| C880 |  |  |  |  |  |  |  |
| C1355 |  |  |  |  |  |  |  |
| C2670 |  |  |  |  |  |  |  |
| C3540 |  |  |  |  |  |  |  |
| C6288 |  |  |  |  |  |  |  |
| C7552 |  |  |  |  |  |  |  |

Test length reduction is (*TLw/ocompress* – *TLw/compress*)/ *TLw/ocompress* x *100%*

2) (optional) You can analyze data in any other ways that can show your advantage. For example,

A. You can draw a figure to show test length growth from *N*=1, 2, 3,…, 8 (before and after compression).

B. You can also compare the difference in test length reduction among: DTC\_only, STC\_only, and both\_DTC&STC.

3) (Mandatory) Please explain your innovations and novel algorithm clearly in your report.

**Grading:**

85% ATPG results (by your ranking in the class)

15% Presentation and Report

**More about grading:**

1. The report is expected to be **in similar form as a paper** which should have the following sections: **problem description, past research, proposed technique, experimental results, discussion,** and **reference**s. Finally, you should add a **contribution** section which describes the **concrete contribution** of each individual team member. Every team member is graded independently according to his/her contribution.

2. Different from previous PA, this project is graded **by the ranking** of your work in the whole class. So please write very clearly in your report what are your innovation and novel algorithm. How much improvement can you achieve by adding your ideas.

3. ATPG results are ranked in the order of three factors: **fault coverage, test length, and run time**.ATPG results will be first sorted by fault coverage (for *N*=8). If the fault coverage are the same, then the results will be sorted by test length. If your fault coverage is less than **1%** lower than but test length is **10%** better than your competitor, then your rank can be promoted. If test length and fault coverage are approximately the same, then run time will be considered. All cases should be finished in **10 minutes**. For example,

|  |  |  |  |
| --- | --- | --- | --- |
| Rank | Fault coverage | Test length | Run time |
| 1 | 99% | 80 | 1:00 |
| **2** | **97%** | **100** | **6:00** |
| 3 | 98% | 120 | 6:00 |
| **4** | **96%** | **151** | **3:00** |
| 5 | 96% | 150 | 6:00 |
| 6 | 95.9% | 149 | 10:00 |

**Submission deadlines:**

1) You can **post your results to the class on 6/2** so that you can know your relative ranking among the class.

2) You are required to **submit your code to COOL on 6/22 midnight (NO DELAY)**. Please make a directory ***<team\_number>\_project*** and copy 3 items /*src*, *report.pdf*, *readme* into directory. You can use the following command to compress a whole directory:

tar -zcvf <filename>.tgz <dir> Then submit a single \*.*tgz* file to COOL system. Include everything so that your code can be easily compiled using ‘make’.

3) Please submit a **hardcopy of your report in the class of 6/9**. Please write your report in the format of a paper.

4) Please demo your program to professor **on 6/23, 9AM-12 in EE2-339**. You are allowed to submit a revised report in the demo.

**References:**

[Benware 03] B. Benware , C. Schuermyer , S. Ranganathan , R. Madge , P. Krishnamurthy,” Impact of multipledetect test patterns on product quality, “IEEE *Int’l Test Conference*, 2003.

[Hamzaoglu 98] I.Hamzaoglu, J.Patel, “Test set compaction algorithms for combinational circuits”, ICCAD 1998.

[Xiang 14] 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)

[Remersaro 09] Remersaro, Santiago, et al. "A scalable method for the generation of small test sets." 2009 Design, Automation & Test in Europe Conference & Exhibition. IEEE, 2009.

[Kumar 13] Kumar, Amit, et al. "On the generation of compact test sets." 2013 IEEE International Test Conference (ITC). IEEE, 2013.

[Lin 01] 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.

---

**Copying source code results in zero grade for both students!**

---

COPYRIGHT ANNOUNCEMENT

The copyright of this PODEM program belongs to the original authors. This program is only for our education purpose.