**N-detect TDF ATPG and Compression**

**(Due: 1/3 9AM)**

**Introduction:**

We provide the C source codes of an ATPG (automatic test pattern generation) program for single stuck-at faults. This ATPG system has two major functions: test pattern generation and fault simulation. However, in order to make our tool stronger, we want to add a new function: N-detect transition delay fault ATPG. On the other hand, 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 compression and another is dynamic compression.

Your job is to build the whole TDF ATPG with N detection, and also do the test patterns compression. We have two kinds of compression, one is dynamic compression, and another one is static compression.

**Required Commands:**

In this project, we need you to create some commands, so we can choose different mode. 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 decide if we do the test compression, just as follow. If we add this flag, we will do the compression, otherwise, we will not.

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

Third, we need a flag “-ndet number” for the number of N detection. Notice that the flag (*e.g. -ndet*) is followed by the number of detection (*e.g. number*). For example, if we want to operate our tool with 8 detection, we can simply type the following command.

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

**Assignment:**

Before you start, you should read the *readme* file that explains important data structure of this PODEM code. You can add your flags in file *tpgmain.c*, and write your ATPG code in file *tdfatpg.c*, which is added by yourself. Also, you need to modify *makefile* because you add a new file. It’s free for you to modify other files, but you should clearly write down which part you modify in your report. You could find out some references about test compression at the end of this file. Notice that you cannot use the complete dictionary to do the test compaction.

1) Please fill in the following table with 8-detection.

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

2) You should draw a figure which tell us the relation between number of detection and the average of the improved ratio after doing the compression of all cases (test patterns).

3) Please print out the critical parts of your code and explain it.

**Grading:**

We will take care of three parts: fault coverage, test length and run time.The score of these parts are 75% of the total score.In this project, we will operate your tool with 8 detection. Scores are given by relative ranking of the three parts as mentioned above, the most important is test length, then is run time, the last is fault coverage. Also, your final report is 25% of your total score.

**Submission:**

Make a directory *<team\_number>\_project*

Please copy 3 items /*podem*, *report*, *readme* into directory. Then submit a single \*.*tgz* file to CEIBA system. Please submit your code on *ceiba*. Include everything so that your code can be easily compiled using ‘make’. Also submit a hardcopy of your report on class or to BL-427 by the deadline. You can use the following command to compress a whole directory:

tar -zcvf <filename>.tgz <dir>

**Reference:**

[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)

---

**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.