**N-detect TDF ATPG and Compression**

**(Due: 1/2 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 test compression* (STC) and another is *dynamic test compression* (DTC).

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

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

**Assignments:**

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*. 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 file. Notice that you cannot use the complete dictionary to do the test compaction.

**I/O Files:**

We explain the I/O information using the following circuit, *c17*. We chain the PIs and POs up, the head of the chain is the first PI (*e.g.* *PI1*), and the tail of the chain is the last PO (*e.g.* *PO2*).



Figure 1. *c17* circuit

The following file is our circuit file (*c17.ckt*). At the beginning, the circuit name is shown after the keyword “name”. Second, circuit primary inputs (PIs) go after the keyword “i”. Third, the circuit primary outputs (POs) go after the keyword “o”. Last, the rest of gates go after the keyword “g” and its gate index. Take “g1 nand 6GAT(3) 3GAT(2) ; 11GAT(5)” for example, it means g1 is a NAND gate, and inputs of this NAND gate are signal 6 and signal 3, and output of this NAND gate is signal 11. The numbers in the parentheses are total gates index, which can be ignored here. Notice that we chain the PIs and POs according to the order they appear in the file. In this case, we chain the PIs and POs from *1GAT(0)* to *23GAT(9)*.

|  |
| --- |
| name C17.iscas  i 1GAT(0)  i 2GAT(1)  i 3GAT(2)  i 6GAT(3)  i 7GAT(4)  o 22GAT(10)  o 23GAT(9)  g1 nand 6GAT(3) 3GAT(2) ; 11GAT(5)  g2 nand 3GAT(2) 1GAT(0) ; 10GAT(6)  g3 nand 7GAT(4) 11GAT(5) ; 19GAT(7)  g4 nand 11GAT(5) 2GAT(1) ; 16GAT(8)  g5 nand 19GAT(7) 16GAT(8) ; 23GAT(9)  g6 nand 16GAT(8) 10GAT(6) ; 22GAT(10) |

Figure 2. *c17* circuit file

The output pattern file should follow the format below. The circuit data is shown at the beginning. Then, all test vectors are shown after the keyword “T”, which composed of two parts. The two parts are separated by a blank, first part is for our primary inputs, and the second part is for our shifting bit in launch on shift technique.

|  |
| --- |
| #Circuit Summary:  #---------------  #number of inputs = 5  #number of outputs = 2  #number of gates = 6  #number of wires = 11  T'00110 1'  T'10111 0'  T'10001 1'  T'01000 0'  T'11011 1'  T'01100 0'  T'10000 1'  T'01111 0' |

Figure 3. *c17* pattern file

Please make sure your pattern can be run using golden\_tdfsim

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 a hardcopy report and show critical parts of your code and explain them in your report.

**Grading:**

85% ATPG results

15% Report

ATPG results are ranked by 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 your fault coverage is less than **1%** lower than others but test length is more than **10%** better, then your rank can be promoted. If test length and fault coverage are approximately the same, then run time will be considered. For example,

|  |  |  |  |
| --- | --- | --- | --- |
| Rank | Test length | Fault coverage | Run time |
| 1 | 80 | 99% | 1:00 |
| 2 | 100 | 97% | 1:00 |
| 3 | 130 | 98% | 2:00 |
| 4 | 150 | 96% | 4:00 |
| 5 | 149 | 95.9% | 10:00 |

**Testing:**

We will provide alpha test on 12/26 and beta test on 1/2. If you submit the code on time, TA will do a grading report for you.

**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 in the class of 1/2**. 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.