Project

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							

Project

C1355							
C2670							
C3540							
C6288							
C7552							
Average							

Test length reduction is $(TL_{w/ocompress} - TL_{w/compress})/ TL_{w/ocompress}$ x *100%*

2) (Mandatory) 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:

15% Presentation and Report (see the other word file)

85% ATPG results (see the following)

Scores of ATPG results:

Average ATPG results of eight circuits (N=8, with STC+DTC) are ranked by three factors: **fault coverage, test length, and run time**. The team(s) with the best fault coverage rank will get 100 points. The second rank will be deducted by four points and etc. Teams with very close fault coverage will be ranked the same (e.g. teams 4, 5, 6 in the following table.) Test length is ranked in a similar way but the full score is 50 with 2 points for each rank below. Runtime is also ranked in a similar way but the full score is 50 with 1 point for each rank below. The total score is the summation of three scores.

Team	Fault coverage	FC score (-4 points for each rank)	Test length	TL score (-2 points for each rank)	Run time	RT score (-1 points for each rank)	Total score
1	99%	100	80	50	1:00	50	200
2	97%	92	100	48	6:00	**48**	188
3	98%	96	120	46	5:59	**48**	160
4	96%	**88**	151	**44**	3:00	49	181
5	96%	**88**	150	**44**	6:01	**48**	180
6	95.95%	**88**	170	42	10:00	47	177

Submission deadlines: see the other word file

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.

Project

[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!
