Programming Assignment #1

Simulator

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. For test pattern generation, it implements the PODEM algorithm [Goel 1981]. For fault simulation, it implements the parallel fault simulation algorithm. In this PA, we first show you how to run *fault simulation* mode. Then, you are asked to fill in the holes in the fault simulator and logic simulation code.

Tutorial:

First, please enter the bin directory, and run the golden binary to see results of fault simulation mode. Notice that you should run these commands on Linux platforms. Otherwise, it may not work. Simply type the following commands.

```
cd bin
./golden_atpg -fsim  ../reports/golden_c17.report
../sample_circuits/c17.ckt
```

Then you will see results generated by fault simulation.
Second, leave the bin directory, enter the *src* directory, and compile the source code by typing.

```
cd ..
cd src
make
```

This code should be compiled correctly on Linux machines. In case you see any compilation error, please fix it before you proceed. If you succeed, an executable file '`atpg`' is generated. Then, in the same directory, run this software in fault simulation mode by typing the following command.

```
./atpg -fsim ../reports/ golden_c17.report ../sample_circuits/c17.ckt
```

The number of total faults, detected faults, and fault coverage should be shown. Notice that the fault coverage is not correct now because there are some holes in the source code. Your job is to fill in the holes so that your outputs are the same as those generated by the golden binary.

I/O Files:

We explain the results using the following circuit, *c17*.

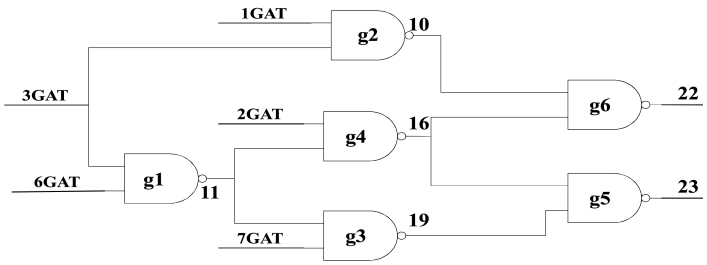

Figure 1. *c17* circuit

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

Programming Assignment #1

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.

```
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 following file is our test pattern (golden_*c17.report*), which is generated by our ATPG tool. The circuit data is shown at the beginning. Then, all test vectors are shown after the keyword *"T"*. The other ATPG results are shown after test vectors: fault coverage, number of backtracks, etc.

```
#Circuit Summary:
#---------------
#number of inputs = 5
#number of outputs = 2
#number of gates = 6
#number of wires = 11
#atpg: cputime for reading in circuit ../sample_circuits/c17.sim: 0.0s 0.0s
#atpg: cputime for levelling circuit ../sample_circuits/c17.sim: 0.0s 0.0s
#atpg: cputime for rearranging gate inputs ../sample_circuits/c17.sim: 0.0s 0.0s
#atpg: cputime for creating dummy nodes ../sample_circuits/c17.sim: 0.0s 0.0s
#number of equivalent faults = 28
#atpg: cputime for generating fault list ../sample_circuits/c17.sim: 0.0s 0.0s
T'00110'
T'10111'
T'10001'
T'01000'
T'11011'
T'01100'
T'10000'
T'01111'
#number of aborted faults = 0
#number of redundant faults = 0
#number of calling podem1 = 8
#total number of backtracks = 0
#FAULT COVERAGE RESULTS :
#number of test vectors = 8
#total number of gate faults = 34
#total number of detected faults = 34
#total gate fault coverage = 100.00%
#number of equivalent gate faults = 28
#number of equivalent detected faults = 28
#equivalent gate fault coverage = 100.00%
#atpg: cputime for test pattern generation ../sample_circuits/c17.sim: 0.0s 0.0s
```

Figure 3. *c17* input file (generated by ATPG)

Programming Assignment #1

Guide for tracing code:

1. Readme provided a short introduction about the data structure for this code. Please read it first.

2. Please read *sim_.cpp* and *faultsim.cpp* carefully. In both files, you can search "TODO" to find the part you need to fill.

3. In *sim.cpp*, *sim()* is a function performing good simulation. Besides, the simulation process is event driven.

3. In *faultsim.cpp*, *fault_simulates_vectors* is a function that will simulate all the patterns provided in pattern file. fault_sim_a_vector is a function that simulated each pattern. *inject_fault_value* will be called in *fault_sim_a_vector*.

Assignments:

Before you start, you should read the *readme* file that explains important data structure of this PODEM code. Please fill in the holes in the fault simulation code. Enter *src* directory and find the file *sim.cpp* and *faultsim.cpp*, there are some holes in these files. You can simply type the keyword "TODO" to find out the holes. After you fill in these holes, your results should be same as our golden results.

1) Please fill in the following table in your report.

circuit number	number of test vector	number of gates	number of total faults	number of detected faults	number of undetected faults	fault coverage
C499						
C1355						
C6288						
C7552						

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

3) (10% bonus) In our parallel fault simulation algorithm, faults will be dropped once they have been detected. Now, we would like to support *N*-detect in our fault simulation, and that mean every fault should be detected *N* times before dropping. You should support the command below,

```
./atpg -fsim <pattern file> -ndet <N>  <circuit file>
```

N is a number from 1 to 8. For example,

```
./atpg -fsim ../sample_circuits/c17.input -ndet 3 ../sample_circuits/c17.ckt
```

When you print *vector[i] detects m faults* on the screen. Please make sure that these *m* faults have been detected for *N* times.

Grading:

80% correctness
20% report
10% bonus

Programming Assignment #1

Submission:
Make a directory <*student_id*>*_pa1*
Please copy 4 items /*src*, *report*, *makefile*, *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'. You can use the following command to compress a whole directory: `tar -zcvf <filename>.tgz <dir>`

Reference:
[Goel 1981] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," IEEE Trans. on Computers, Vol. 30, No.3, pp215-222, 1981.

Copying source code results in zero grade for both students!
