CIS 4930 Testing & Fault Tolerance in Digital Systems Final Project

Problem Description

You need to code an automatic test generator that implements (a) fault collapsing; and (b) some test algorithms. We assume a single-stuck-fault (SSF) model. You may implement the generator in C/C++/JAVA. The following are the requirements for the program:

1. The program must run in an interactive mode.
2. The input to the program is a gate-level net-list.
3. The generator must parse the gate-level netlist and then report the following:
   (a) The final list of faults after performing fault collapsing
   (b) For each detectable fault, show the list of test vectors that will detect such a fault.
   (c) The list of undetectable faults (if any).

On program invocation, the following interactive menu should be presented to the user:

```
[0] Read the input net-list
[1] Perform fault collapsing
[2] List fault classes
[3] Generate tests (D-Algorithm or PODEM)
[4] Generate tests (Boolean satisfaibility)
[5] Exit
```

Figure 1 shows the general flow of the test pattern generator. First it reads the circuit (Option 0) and generates appropriate data structures. Then it performs fault collapsing (Option 1) and creates the fault classes. For Option 3, you can select one of the two test generation algorithms discussed in class to implement. For Option 4, you need to convert the test generation algorithm into a Boolean satisfiability problem, and then use an existing Boolean SAT solver to find the tests. Some of the modern efficient solvers can be found at `http://www.satlive.org/solvers/`. *Make sure you pick one of the CDCL solvers.*

Teaming

The project is performed by teams of **up to two** students. Both members of a team will receive the same score. Once formed, teams cannot be altered or switched, unless a member of a team withdraws from the course, or granted for extended absence from the course.

Team members should coordinate and work together closely, and make contributions equally. Note that your instructor does not have capacity to monitor the efforts and contributions from each member in a team, *you are responsible for picking your team member.*

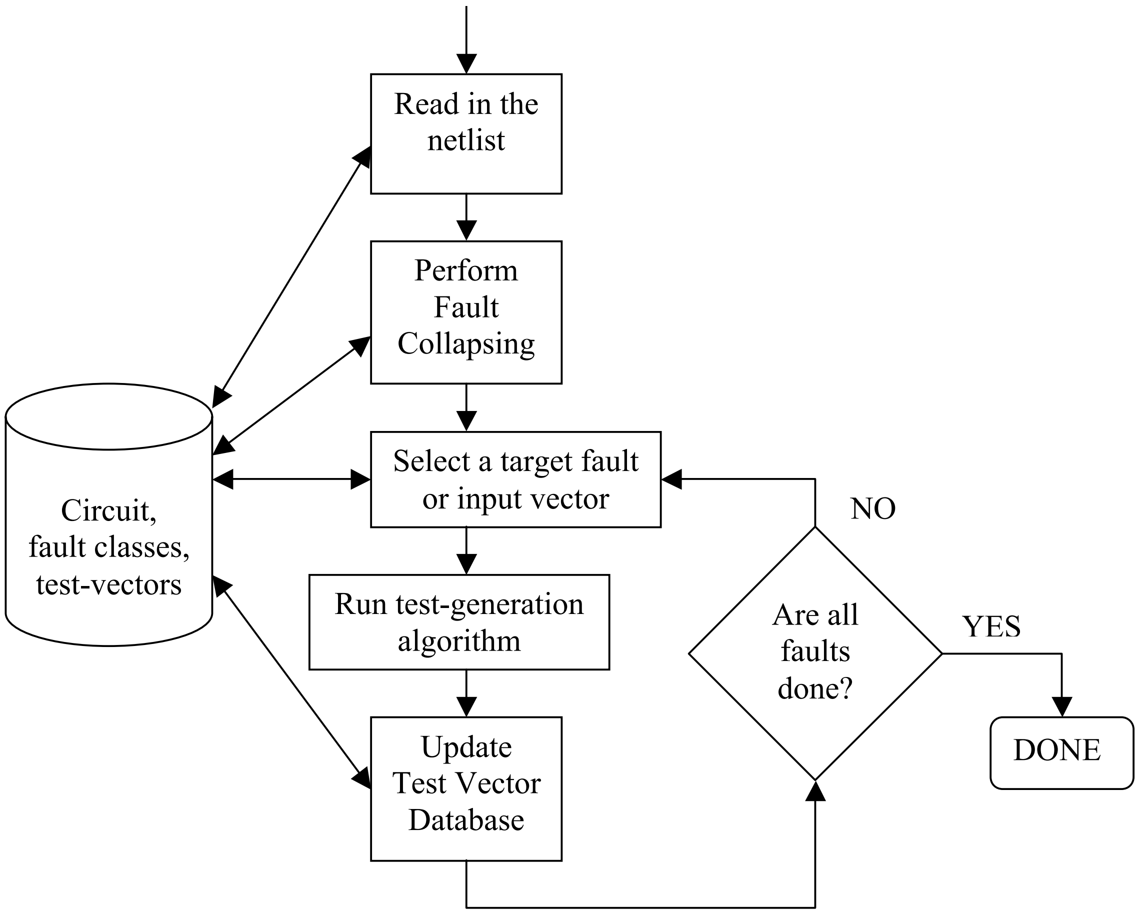
Read in the netlist
Perform Fault Collapsing
Select a target fault or input vector
Run test-generation algorithm
Update Test Vector Database
Circuit, fault classes, test-vectors
Are all faults done?
NO
YES
DONE

Figure 1: High level task flow for the ATPG.

Grade Distribution

This project carries 30% of the final grade, and it is evaluated as follows.

1. Coding style - *Readability* and *Modularity.*
2. Exits gracefully on wrong input entry (such as mistype, wrong option, non-existent file name, etc.)
3. A benchmark suite of 5 benchmarks will be provided to you. Your ATPG should run correctly for these five benchmarks.
4. Additional 5 benchmarks (not provided to you) will be run to check your code during your project demonstration.

Deliverables

1. Submit a final report summarizes your project, *i.e.* code structure, compile instructions, results from applying your generator to the five benchmark circuits.
2. Upload archived code in a single zipped file including a above final report, Makefile, and other necessary files.

Input Netlist format

The input netlist (self-explanatory) will be given in a format illustrated by the following example. Note that "$" denotes the start of a comment.

```
$c17 iscas example (to test conversion program only)
$---------------------------------------------------
$
$  total number of lines in the netlist ..............    17
$  simplistically reduced equivalent fault set size =     22
$        lines from primary input  gates .......     5
$        lines from primary output gates .......     2
$        lines from interior gate outputs ......     4
$        lines from **     3 ** fanout stems ...     6
$
$        avg_fanin  =  2.00,     max_fanin  =  2
$        avg_fanout =  2.00,     max_fanout =  2
$
1gat                                    $... primary input
2gat                                    $... primary input
3gat                                    $... primary input
6gat                                    $... primary input
7gat                                    $... primary input
                                        $... primary input
$
$

22gat                                   $... primary output
23gat                                   $... primary output
                                        $... primary output
$
$
$       Output  Type    Inputs...
$       ------  ----    ---------
        10gat   nand    1gat    3gat
        11gat   nand    3gat    6gat
        16gat   nand    2gat    11gat
        19gat   nand    11gat   7gat
        22gat   nand    10gat   16gat
        23gat   nand    16gat   19gat
```