VLSI Final Project Report

TESTING & FAULT TOLERANCE IN DIGITAL SYSTEMS

John Adams && Mary Mouro | VLSI Testing | April 30, 2019

COMPILE INSTRUCTIONS

1. Download and unzip our project named “vlsi_Final_Project”from canvas
2. Run command $ cd vlsi_Final_Project
3. Run command $ pip3 install sympy
4. Run command $ sudo apt-get install libz-dev
5. Run command $ cd vender/minisat
6. Run command $ make config prefix=$PREFIX
7. Run command $ sudo make install
8. Run command $ mv build/dynamic/bin/minisat $PATH TO vlsi_Final_Project/bin
9. move benchmark (.ckt files) to the project vlsi_Fault_TG/benchmarks
10. cd to project directory vlsi_Final_Project/src
11. run command "python3 main.py"

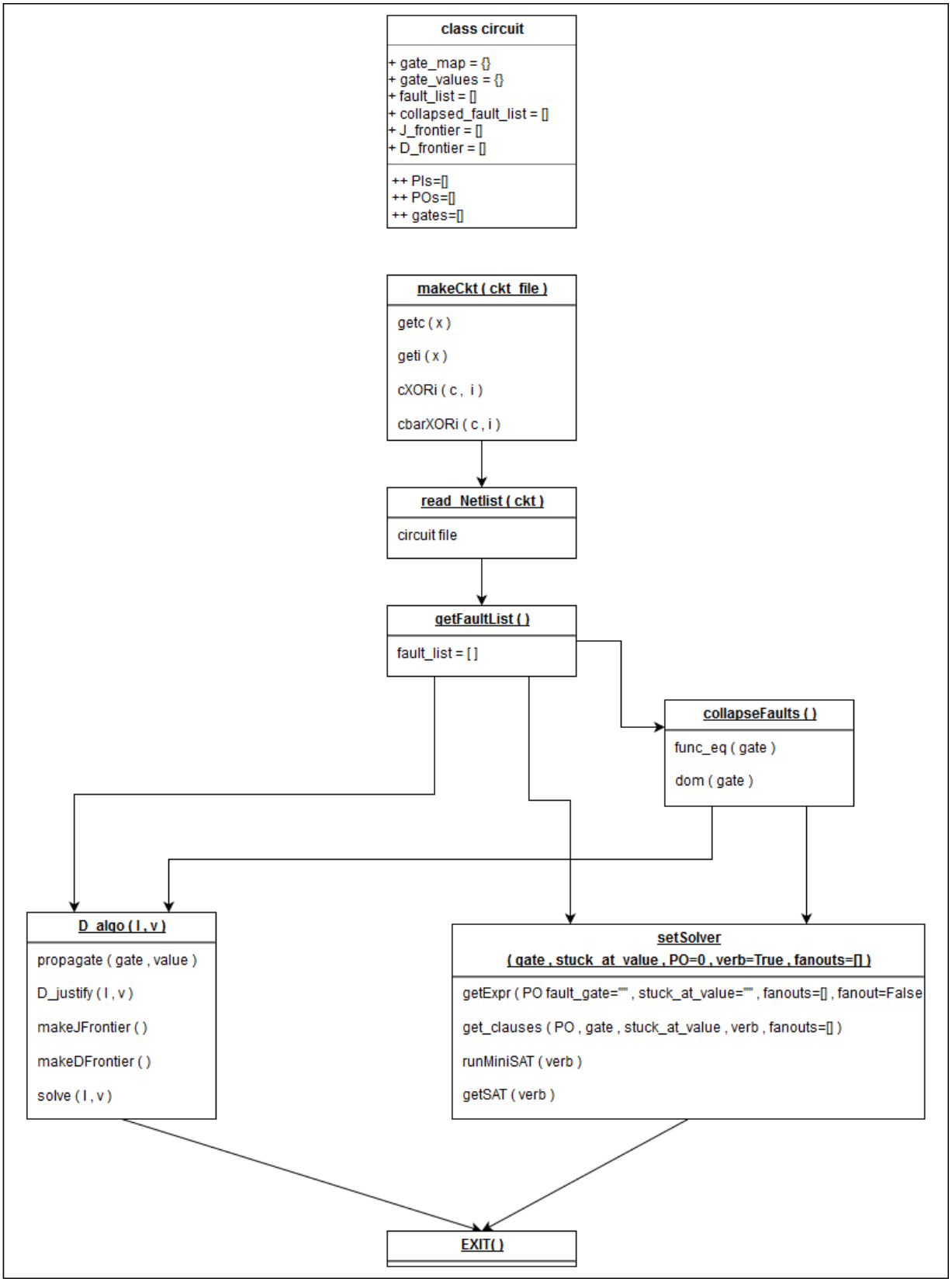
NOTES:

Tested on linux Ubuntu 18.04 LTS
Minisat set solver has some dependencies that need to be installed, other project dependencies are listed below

DEPENDENCIES:

sympy
libz-dev
python3
g++

CODE STRUCTURE



RESULTS

```
jadams @ beast ~/dev/repos/vlsi Fault TG (master *%=)
└─ $ ▶ /usr/bin/python3 /home/jadams/dev/repos/vlsi Fault TG/src/main.py


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


Enter a number from 0 to 5:
```

The program starts by prompting the user with the required interactive menu

Option [0] Read Netlist

we read in the netlist and show info about the circuit such as primary inputs and outputs and gate info.

```
Enter a number from 0 to 5: 0
[0] t5 26a v1.ckt
[1] t5 10.ckt
[2] t4 21.ckt
[3] t4 3.ckt
4 circuits detected! Please select the corresponding number to read the netlist
2
Circuit: t4 21.ckt

Primary Inputs:
['1gat', '2gat', '3gat', '4gat', '5gat']

Primary Ouputs:
['9gat']

Gates:
['10gat', 'and', '1gat', '2gat']
['6gat', 'nor', '3gat', '10gat']
['7gat', 'and', '3gat', '4gat']
['8gat', 'or', '5gat', '7gat']
['9gat', 'or', '6gat', '8gat']
```

It prompts you again to select which circuit and after a selection is made it prints the circuit information

Option [1] Fault Collapsing

It will show the relationship for dominance and equivalences and show the collapsed fault list

```
Enter a number from 0 to 5: 1
functionally equivalent faults:  {10gat-0,1gat-0,2gat-0}
1gat-0,2gat-0 will be removed from fault list
functionally equivalent faults:  {6gat-0,3gat-1,10gat-1}
3gat-1,10gat-1 will be removed from fault list
functionally equivalent faults:  {7gat-0,3gat-0,4gat-0}
3gat-0,4gat-0 will be removed from fault list
functionally equivalent faults:  {8gat-1,5gat-1,7gat-1}
5gat-1,7gat-1 will be removed from fault list
functionally equivalent faults:  {9gat-1,6gat-1,8gat-1}
6gat-1,8gat-1 will be removed from fault list

dominating relationship such that f dominates g (f,g)
f will be removed from fault list
(10gat-1,1gat-1)
(10gat-1,2gat-1)
(6gat-1,3gat-0)
(6gat-1,10gat-0)
(7gat-1,3gat-1)
(7gat-1,4gat-1)
(8gat-0,5gat-0)
(8gat-0,7gat-0)
(9gat-0,6gat-0)
(9gat-0,8gat-0)

Fault list after collapsing:
['10gat-0', '1gat-1', '2gat-1', '3gat-1', '3gat-0', '4gat-1', '5gat-0', '6gat-0', '7gat-0', '8gat-0', '8gat-1', '9gat-1']
```

Option [2] List fault classes

The 2nd option will a fault list of all SSFs if option [1] is not selected first, otherwise it will show the fault list similar to above. Here's an example of listing fault classes with fault collapsing.

```
Enter a number from 0 to 5: 2
All SSFs before collapsing:
1gat-0
1gat-1
2gat-0
2gat-1
3gat-0
3gat-1
4gat-0
4gat-1
5gat-0
5gat-1
6gat-0
6gat-1
7gat-0
7gat-1
8gat-0
8gat-1
9gat-0
9gat-1
```

Option [3] D-Algorithm

this option will show the assignments made to propagate the error and the test vector that detects a fault if it is detectable.

```
Enter a number from 0 to 5: 3
initalizing all values to x
5gat {'2gat': 'x', '3gat': 'x', 'output': 'x'}
6gat {'1gat': 'x', 'output': 'x'}
7gat {'1gat': 'x', '5gat': 'x', 'output': 'x'}
8gat {'6gat': 'x', '4gat': 'x', 'output': 'x'}
9gat {'7gat': 'x', '8gat': 'x', 'output': 'x'}
SUCCESSFUL PROPAGATIONS!
Assignments from propagation:
5gat {'2gat': 'x', '3gat': 'x', 'output': 'x'}
6gat {'1gat': 'D', 'output': 'D'}
7gat {'1gat': 'D', '5gat': 'x', 'output': 0}
8gat {'6gat': 'D', '4gat': 1, 'output': 'D'}
9gat {'7gat': 0, '8gat': 'D', 'output': 'D'}
D has propagated to PO and all lines justified!
Test vector:  [1, 1, 0, 0]  will detect  1gat  stuck at  0
```

Option [4] Boolean Satisfiability

This option uses a modern SAT solver to generate the test vectors to detect single stuck at faults.

```
Enter a number from 0 to 5: 4
1gat stuck at  0
faulty expression is:  4gat & ~1gat
fault free expression is:  (4gat & ~1gat) | (1gat & (2gat | 3gat))
faulty XOR fault free =  1gat & (2gat | 3gat)
WARNING: for repeatability, setting FPU to use double precision
============================[ Problem Statistics ]=============================
|                                                                             |
|  Number of variables:            3                                          |
|  Number of clauses:              1                                          |
|  Parse time:                  0.00 s                                        |
|  Eliminated clauses:          0.00 Mb                                       |
|  Simplification time:         0.00 s                                        |
|                                                                             |
============================[ Search Statistics ]==============================
| Conflicts |          ORIGINAL         |          LEARNT          | Progress |
|           |    Vars  Clauses Literals |    Limit  Clauses Lit/Cl |          |
===============================================================================
===============================================================================
restarts              : 1
conflicts             : 0             (0 /sec)
decisions             : 1             (0.00 % random) (738 /sec)
propagations          : 1             (738 /sec)
conflict literals     : 0             (-nan % deleted)
Memory used           : 22.29 MB
CPU time              : 0.001355 s

SATISFIABLE
input vector:  [1, 0, 1, 0]  will detect fault!
```

SUMMARY

The project uses the circuit class functions shown in the UML diagram in the code structure section and performs the reading of a netlist file, fault collapsing using fault equivalence and fault dominance relationships, and finally test generation using the D-Algorithm and the miniSAT solver. The D-Algorithm first attempts to propagate the error to the output and then goes back and justifies all the lines and returns the test vector that can detect the faults in the fault list. The setSolver function calculates the circuit's Boolean expression for the fault free circuit and the expression for the faulty circuit, XORS them, converts the XOR expression to CNF and then calls the miniSAT set solver.