User Manual

Fault Simulator and Test Generator

TABLE OF CONTENTS

(a) User Manual: Deductive Fault Simulator

Simulation Environment:

The simulation environment needed to run the program is Python 3. To be specific, the version that I used to verify that the program worked is Python 3.12.8.

README:

1. Setup tool to test the program:

Please ensure that the Python environment has support for the following modules: enum, argparse, random. Python 3.12.8 version mentioned above should have support for these modules by default.

2. Run the program, take netlist and input, write output:

From the main directory (Final_Project), please change working directory to the source code's directory by using the following command:

```
cd Deductive_FS/src
```

To run the program, please use the command format below:

```
python3 demo_sim.py --circuit <netlist name> --inputs <input vector or list of input vectors> --output <output filepath> [--faults <faults filepath>] [--cumulative]
```

The square brackets mean that the corresponding argument is optional.

For clarity, I provide explanation for the additional arguments as below:
`--circuit` (required): user defines the name of netlist (circuit) in Final_Project/Deductive_FS/files/ directory (if you want to test with new netlist, please put the new netlist file in the Final_Project/Deductive_FS/files/ directory).
`--inputs` (required): user defines input vector or comma-separated list of input vectors that the fault simulator needs to handle.
`--output` (required): user defines the path of file to write the output.
`--faults` (optional): user defines the path of file that stores the stuck-at faults to simulate for the fault simulator; if it is not defined, all faults in all nets are simulated.
`--cumulative` (optional): if it is not present, program performs separate fault simulation with each input vector in the list defined with `--inputs`; if it is present, program performs cumulative fault simulations and adds faults detected by each input vector to the list of faults detected so far in that run.

For each input vector, the output file has information of the netlist (circuit), input vector, faults detected (if `--cumulative` is not included in command) or all faults detected so far (if `--cumulative` is included in command) along with number of faults detected.

To clarify how to use the above commands, I provide some examples with screenshots of steps and corresponding output files generated. Because the commands are a bit long, please zoom in the screenshots if it is a bit hard to see initially.

Example 1: As illustrated in Figure 1, I change the working directory from the main directory to the source code's directory and run fault simulator with netlist s27.txt from Final_Project/Deductive_FS/files/ directory, input vectors 1110101 and 1101101, output file as Final_Project/Deductive_FS/test_files/fault_sim_demo1.txt file. The program simulates all faults in all nets of netlist and performs separate fault simulation with each input vector. The content of this example's output file is shown in Figure 2.

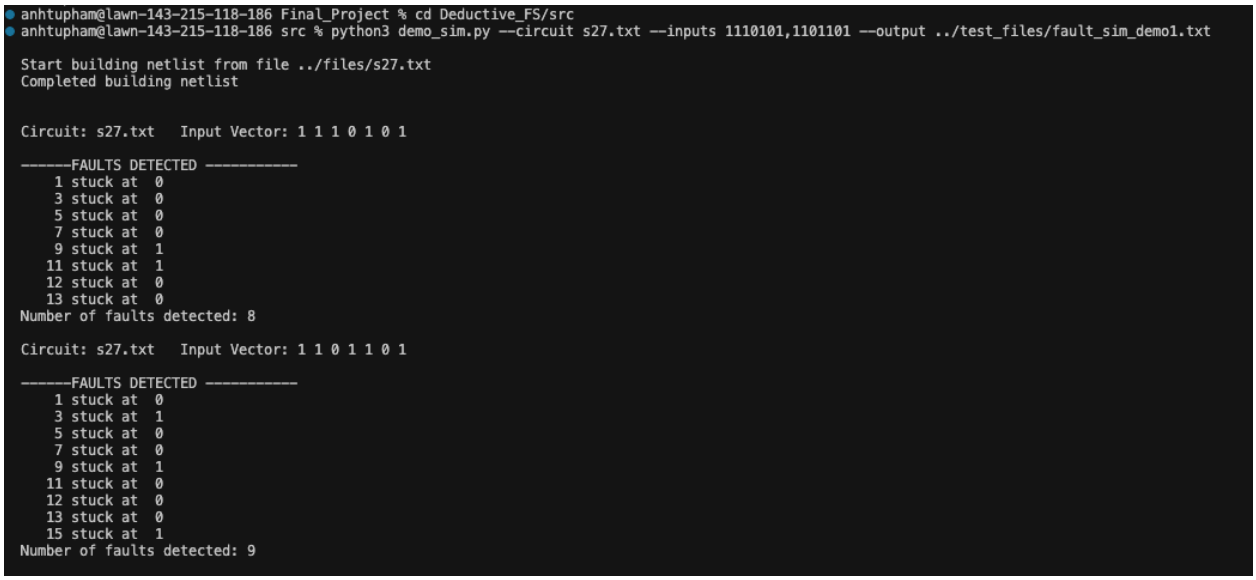

Figure 1: Fault simulator's Example 1

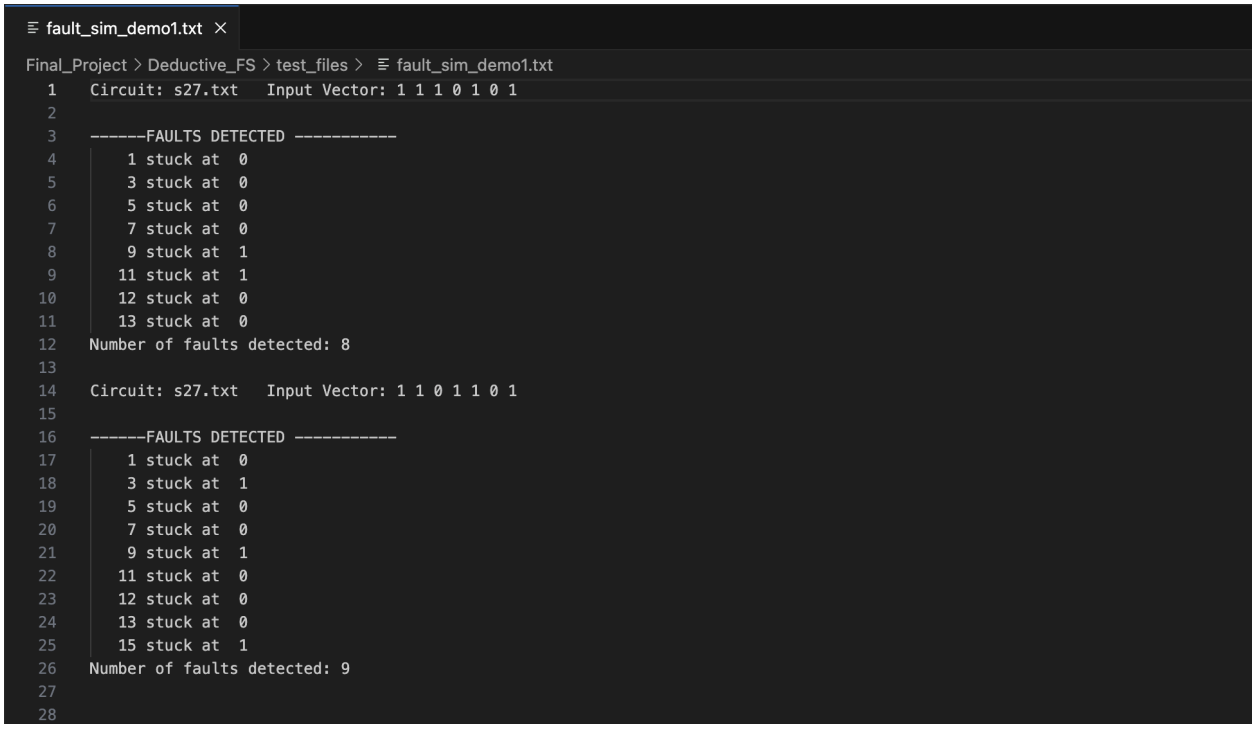

Figure 2: Output file for Fault simulator's Example 1

To prepare for the next 2 examples, I introduce a new faults file named s27_faults.txt in Final_Project/Deductive_FS/test_files/ directory as shown in Figure 3.

Figure 3: The new faults file

Example 2: As demonstrated in Figure 4, I run the fault simulator with netlist s27.txt from Final_Project/Deductive_FS/files/ directory, input vector 1110101, output file as Final_Project/Deductive_FS/test_files/fault_sim_demo2.txt file, and faults to simulate from Final_Project/Deductive_FS/test_files/s27_faults.txt file. The content of this example's output file is shown in Figure 5.

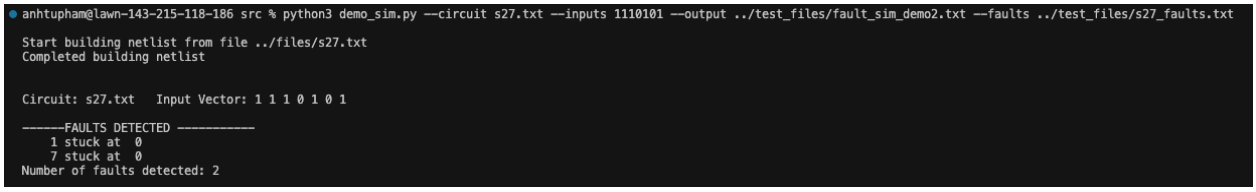

Figure 4: Fault simulator's Example 2

Figure 5: Output file for Fault simulator's Example 2

Example 3: As shown in Figure 6, I run the fault simulator with netlist s27.txt from Final_Project/Deductive_FS/files/ directory, input vector 1110101, output file as Final_Project/Deductive_FS/test_files/fault_sim_demo2.txt file, faults to simulate from Final_Project/Deductive_FS/test_files/s27_faults.txt file, and ask the program to perform cumulative fault simulations (with each input vector, the program shows all faults that have been detected so far in this run). The content of this example's output file is shown in Figure 7.

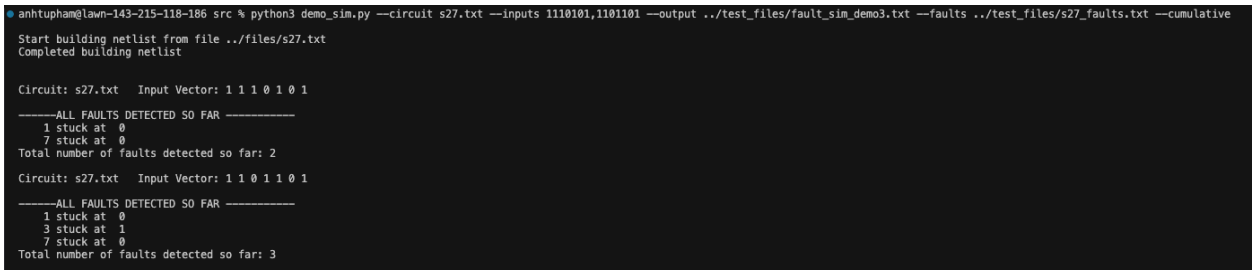

Figure 6: Fault simulator's Example 3

Figure 7: Output file for Fault simulator's Example 3

(b) User Manual: PODEM (Test Generator)

Simulation Environment:

The necessary simulation environment to run the program is Python 3. For detail, I used Python 3.12.8 version to verify that the program worked.

README:

1. Setup tool to test the program:

Please ensure that the Python environment has support for the following modules: enum, argparse, random. The above Python 3.12.8 version should support these modules by default.

2. Run the program, take netlist and input, write output:

From main directory (Final_Project), please change the working directory into the source code's directory by using the following command:

```
cd PODEM/src
```

The program can then be run by using the following command format:

```
python3 demo_gen.py --circuit <netlist name> --faults <stuck-at fault or list of stuck-at faults> --output <output filepath> [--enable_sim]
```

The square brackets mean that the corresponding argument is optional.

To clarify the command format above, I provide the explanation for the additional arguments as shown below:

`--circuit` (required): user defines the name of netlist (circuit) in Final_Project/PODEM/files/ directory (if you want to test with new netlist, please put the new netlist file in the Final_Project/PODEM/files/ directory).

`--faults` (required): user defines stuck-at fault or space-separated list of stuck-at faults; each stuck-at fault is in the form of <faulty net>,<stuck-at value>.

`--output` (required): user defines the path of file to write the output.

`--enable_sim` (optional): if it is not present, the program only runs test vector generation(s); if it is present, the program also runs fault simulation with each generated test vector if the corresponding input fault is detectable. This option supports cross-verification of test generator and fault simulator.

For each input fault, the output file contains the information of the netlist (circuit), the input fault, and the corresponding test vector generated or the message "The fault is undetectable" if there is no satisfying test vector. If `--enable_sim` is included in command and the fault is detectable, the output file also contains the information from fault simulation with the generated test vector.

To further clarify how the above commands can be used, I provide some examples with screenshots of the steps and the corresponding output files generated.

Example 1: As illustrated in Figure 8, I change the working directory from the main directory to the source code's directory, then I run test generator with netlist s27.txt from Final_Project/PODEM/files/ directory, input faults as net 9 s-a-1 and net 12 s-a-0, and output file as Final_Project/PODEM/test_files/gen_demo1.txt file. The program simply runs the test vector generations. The content of this example's output file is illustrated in Figure 9.

```
anhtupham@lawn-143-215-118-186 Final_Project % cd PODEM/src
anhtupham@lawn-143-215-118-186 src % python3 demo_gen.py --circuit s27.txt --faults 9,1 12,0 --output ../test_files/gen_demo1.txt

Start building netlist from file ../files/s27.txt
Completed building netlist


Circuit: s27.txt   Input fault: Net 9 s-a-1

Test vector generated: 0000100

Circuit: s27.txt   Input fault: Net 12 s-a-0

Test vector generated: 1000100
```

Figure 8: Test generator's Example 1

gen_demo1.txt

Final_Project > PODEM > test_files > gen_demo1.txt

```
1   Circuit: s27.txt   Input fault: Net 9 s-a-1
2
3   Test vector generated: 0000100
4
5   Circuit: s27.txt   Input fault: Net 12 s-a-0
6
7   Test vector generated: 1000100
8
9
```

Figure 9: Output file for Test generator's Example 1

Example 2: With this example, I observe how the test generator behaves with undetectable fault. I introduce a new netlist file named new_netlist.txt in Final_Project/PODEM/files/ directory as shown in Figure 10.
With this netlist, (net 6) = (net 1) · (net 2) + (net 1) · (net 2)' = (net 1) · ((net 2) + (net 2)') = (net 1) · 1 = (net 1), so stuck-at fault with net 2 should be undetectable.

new_netlist.txt

Final_Project > PODEM > files > new_netlist.txt

```
1   AND 1 2 4
2   INV 2 3
3   AND 1 3 5
4   OR 4 5 6
5   INPUT 1 2 -1
6   OUTPUT 6 -1
```

Figure 10: The new netlist (circuit) file

As shown in Figure 11, I run test generator with netlist new_netlist.txt from Final_Project/PODEM/files/ directory, input faults as net 2 s-a-1, and output file as

Final_Project/PODEM/test_files/gen_demo2.txt file. The message “The fault is undetectable” is shown. This example’s output file’s content is illustrated in Figure 12.

```
anhtupham@lawn-143-215-118-186 src % python3 demo_gen.py --circuit new_netlist.txt --faults 2,1 --output ../test_files/gen_demo2.txt

Start building netlist from file ../files/new_netlist.txt
Completed building netlist


Circuit: new_netlist.txt   Input fault: Net 2 s-a-1

The fault is undetectable
```

Figure 11: Test generator’s Example 2

gen_demo2.txt

Final_Project > PODEM > test_files > gen_demo2.txt

```
1   Circuit: new_netlist.txt   Input fault: Net 2 s-a-1
2
3   The fault is undetectable
4
5
```

Figure 12: Output file for Test generator’s Example 2

Example 3: As shown in Figure 13, I run test generator with netlist s27.txt from Final_Project/PODEM/files/ directory, input faults as net 5 s-a-0 and net 14 s-a-0, output file as Final_Project/PODEM/test_files/gen_demo3.txt file, and enable fault simulation with each test vector generated. This example’s output file’s content is shown in Figure 14.

```
anhtupham@lawn-143-215-118-186 src % python3 demo_gen.py --circuit s27.txt --faults 5,0 14,0 --output ../test_files/gen_demo3.txt --enable_sim

Start building netlist from file ../files/s27.txt
Completed building netlist



Start building netlist from file ../files/s27.txt
Completed building netlist


Circuit: s27.txt   Input fault: Net 5 s-a-0

Test vector generated: 0000100

Circuit: s27.txt   Input Vector: 0 0 0 0 1 0 0

------FAULTS DETECTED ----------
    1 stuck at  1
    2 stuck at  1
    5 stuck at  0
    7 stuck at  1
    9 stuck at  1
   10 stuck at  1
   11 stuck at  1
   12 stuck at  1
   15 stuck at  0
Number of faults detected: 9

Circuit: s27.txt   Input fault: Net 14 s-a-0

Test vector generated: 1100000

Circuit: s27.txt   Input Vector: 1 1 0 0 0 0 0

------FAULTS DETECTED ----------
    1 stuck at  0
    2 stuck at  0
    3 stuck at  1
    5 stuck at  0
    7 stuck at  0
    9 stuck at  1
   11 stuck at  0
   12 stuck at  0
   13 stuck at  0
   14 stuck at  0
   15 stuck at  1
   20 stuck at  1
Number of faults detected: 12
```

Figure 13: Test generator’s Example 3

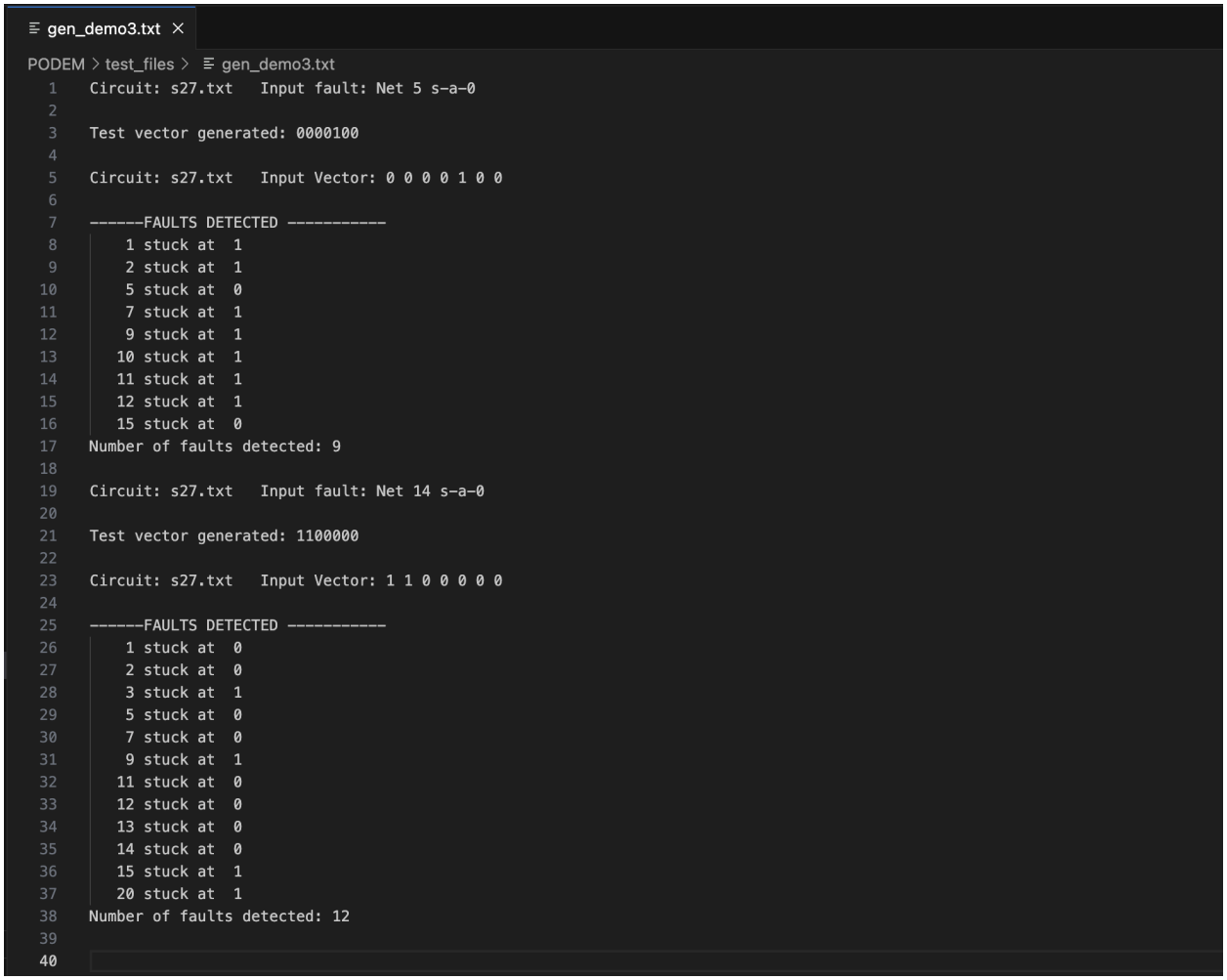

Figure 14: Test generator's Example 3

As shown in Figure 13 and Figure 14, the test vector generated by the test generator for an input fault is able to detect a list of faults including that input fault.