Report: PODEM

Submitted by: Mohammad Adnaan

GT ID: 903519892

PODEM algorithm:

1. First the logic simulator built earlier is modified to incorporate the x (unknown), D (1/0) and $\overline{D}$ (0/1) values.

2. The given circuit file is taken as input and all nodes are initialized as x. Input fault list is also read from the given fault list file.

3. Few utility functions are built. They are output_check() , build_D_frontier(), test_not_possible(), Implication(), Objective() and Backtrace().

4. output_check() function checks whether the value of any output node of the circuit is D or $\overline{D}$. If any output node is D or $\overline{D}$ then PODEM has successfully generated a test.

5. build_D_frontier() checks which gates have any input D or $\overline{D}$ with output x and include them in D-frontier.

6. test_not_possible() function checks whether D frontier is empty and any node except output has value D or $\overline{D}$. It indicates D or $\overline{D}$ cannot be propagated to output. Besides it also checks whether the given fault node has any other value except D or $\overline{D}$ and no input of the gate connected to this node has value x. This situation indicates fault cannot be excited. In these above two cases test_not_possible() returns test is not possible.

7. Implication() function recalculates the value of nodes after an input value assignment.

8. Objective() function returns a node number and value which is to be generated in order to propagate the value D or $\overline{D}$ to output.

9. Backtrace() function using the objective outcome assigns logic value to output.

10. Finally, podem() function is built using the following algorithm

```
for every fault in fault list begin:
   if (output_check() is successful )
      return Success;

   if (test_not_possible)
      return Failure;

   node,value=Objective();
   input node, value =Backtrace (node, value);
   implications (input node, value);
   build_D_frontier();

   if (PODEM() is successful)
      return Success;

   implications (input node, value̅ );
   build_D_frontier();

   if ( PODEM() is successful)
      return Success;

   implications (input node, value=x)
   return Failure;
end
```

11. All generated test vectors are then verified using deductive fault simulator that the faults are detected.

PODEM results

Circuit File: s27	
Fault location	**Test vector**
Net 16 s-a-0	x0x10x0
Net 10 s-a-1	x00xxx0
Net 12 s-a-0	1xxx1xx
Net 18 s-a-1	11x101x
Net 17 s-a-1	10x00x0
Net 13 s-a-0	1xxx1xx
Net 6 s-a-1	x0x10x0
Net 11 s-a-0	x10xxxx

Circuit File: s298f_2	
Fault location	**Test vector**
Net 70 s-a-1	01x1xxxxxxxxxx0xx
Net 73 s-a-0	111xxxxxxxxxxx0xx
Net 26 s-a-1	xx1x1xxx0xxxxxxx
Net 92 s-a-0	x10101xxxxxx0x0xx
Net 38 s-a-0	01x0xxxxxxxxxx0xx
Net 46 s-a-1	x1010xxxxxxx0x0xx
Net 3 s-a-1	1101xxxxxxxxxx0xx
Net 68 s-a-0	x1xx1xxxxxxxx00xx

Circuit File: s344f_2	
Fault location	**Test vector**
Net 166 s-a-0	01x00xxxx011xx0xxxxxxx
Net 71 s-a-1	10xxxxxxxxxxxxxxxxxxxx
Net 16 s-a-0	10xxxxxxxxxxxx1xxxxxxx
Net 91 s-a-1	111xxxxxxxxxxxxxxxxxxx
Net 38 s-a-0	x1xxxxxxxxxx1xxxxxxxxx
Net 5 s-a-1	xxxx0xxxxxxxxxxxxxxxxx
Net 138 s-a-0	01xx00xxxx0x11x0xxxxxxx
Net 91 s-a-0	10xxxxxxxxxxxxxxxxxxxx

Circuit File: s349f_2	
Fault location	**Test vector**
Net 25 s-a-1	xxxxxxxxxxxxxxx1xxxxxxxx
Net 51 s-a-0	00xxxxxxxxxxxxx0xxxxxxxx
Net 105 s-a-1	01x1000xxx01xx10xxxxxxxx
Net 105 s-a-0	01x1xxxxxx1xxxx0xxxxxxxx
Net 83 s-a-1	01xx000xxx0x0x10xxxxxxxx
Net 92 s-a-0	01x0001xxx0001x0xxxxxxxx
Net 7 s-a-0	xxxxxx1xxxxxxxxxxxxxxxxx
Net 179 s-a-0	101xxxxxxxxxxxx0xxxxxxxx

User Manual: Fault Simulator and PODEM

User Manual: Deductive fault simulator:

The program has been built using C++. The program can be run from Linux terminal.

README:

1. Unzip the file ***DST_project_Mohammad_Adnaan***

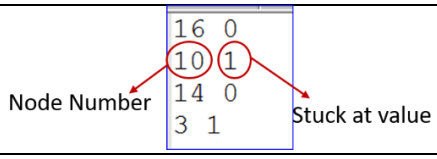
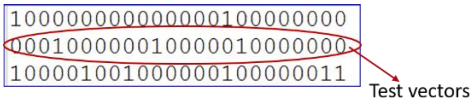
2. The deductive fault simulator can be run in four modes dictated by the two parameters.

mode_all_fault=1 test_vector_all=1	All stuck at 0 and stuck at 1 faults are considered and 100 random test vectors are taken as input. The detected faults for every test vector is calculated and saved in the file **`DST_project_Mohammad_Adnaan/Deductive_FS/files/detected_faults.txt/`** The fault coverage information is saved in **`DST_project_Mohammad_Adnaan/Deductive_FS/files/fault_coverage.txt/`**
mode_all_fault=0 test_vector_all=1	Only the given faults in the file **`DST_project_Mohammad_Adnaan/Deductive_FS/files/input_fault_list.txt/`** are considered and 100 random test vectors are taken as input. The detected faults for every test vector is calculated and saved in the file **`DST_project_Mohammad_Adnaan/Deductive_FS/files/detected_faults.txt/`**
mode_all_fault=1 test_vector_all=0	All stuck at 0 and stuck at 1 faults are considered and test vectors given in the file **`DST_project_Mohammad_Adnaan/Deductive_FS/files/input_test_vector_file.txt/`** are taken as input. The detected faults for every test vector is calculated and saved in the file **`DST_project_Mohammad_Adnaan/Deductive_FS/files/detected_faults.txt/`** The fault coverage information is saved in **`DST_project_Mohammad_Adnaan/Deductive_FS/files/fault_coverage.txt/`**
mode_all_fault=0 test_vector_all=0	Only the given faults in the file **`DST_project_Mohammad_Adnaan/Deductive_FS/files/input_fault_list.txt/`** are considered and test vectors given in the file **`DST_project_Mohammad_Adnaan/Deductive_FS/files/input_test_vector_file.txt/`** are taken as input. The detected faults for every test vector is calculated and saved in the file **`DST_project_Mohammad_Adnaan/Deductive_FS/files/detected_faults.txt/`**

3. The circuit files are kept in the following folder

```
DST_project_Mohammad_Adnaan/Deductive_FS/files
```

4. Edit the files `input_fault_list.txt` and `input_test_vector_file.txt` kept in this folder if required.

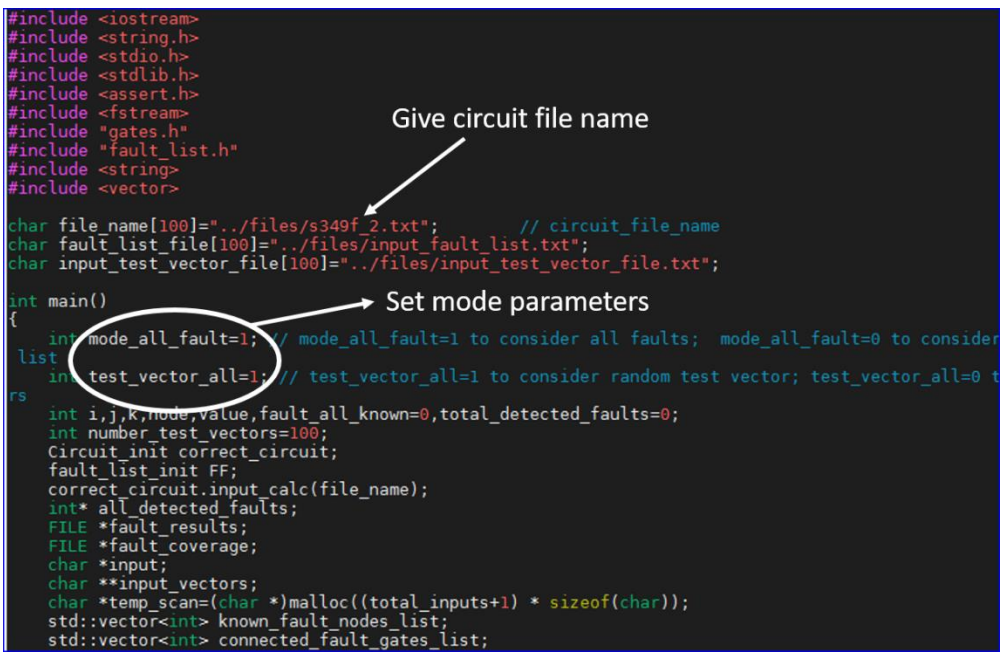
	
Input fault list file format	**Input test vector file format**

4. Navigate to the **src** folder of **Deductive_FS** . All source code files are kept here.

```
DST_project_Mohammad_Adnaan/Deductive_FS/src/
```

5. Run command `vi deductive.cpp` and press I in the keyboard to edit the main file in order to select the circuit file and mode parameters.

6. Set the desired mode parameters and circuit file.




```
#include <iostream>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <fstream>
#include "gates.h"
#include "fault_list.h"
#include <string>
#include <vector>

char file_name[100]="../files/s349f_2.txt";        // circuit_file_name
char fault_list_file[100]="../files/input_fault_list.txt";
char input_test_vector_file[100]="../files/input_test_vector_file.txt";

int main()
{
    int mode_all_fault=1; // mode_all_fault=1 to consider all faults;  mode_all_fault=0 to consider
 list
    int test_vector_all=1;// test_vector_all=1 to consider random test vector; test_vector_all=0 t
rs
    int i,j,k,node,value,fault_all_known=0,total_detected_faults=0;
    int number_test_vectors=100;
    Circuit_init correct_circuit;
    fault_list_init FF;
    correct_circuit.input_calc(file_name);
    int* all_detected_faults;
    FILE *fault_results;
    FILE *fault_coverage;
    char *input;
    char **input_vectors;
    char *temp_scan=(char *)malloc((total_inputs+1) * sizeof(char));
    std::vector<int> known_fault_nodes_list;
    std::vector<int> connected_fault_gates_list;
```

7. After setting the parameters press escape in keyboard, type the command `:wq` and press enter to save the changes and exit the file.

8. In the terminal run the command

```
make all
```

9. Run the following command to set execution permission for the program

```
chmod +x deductive
```

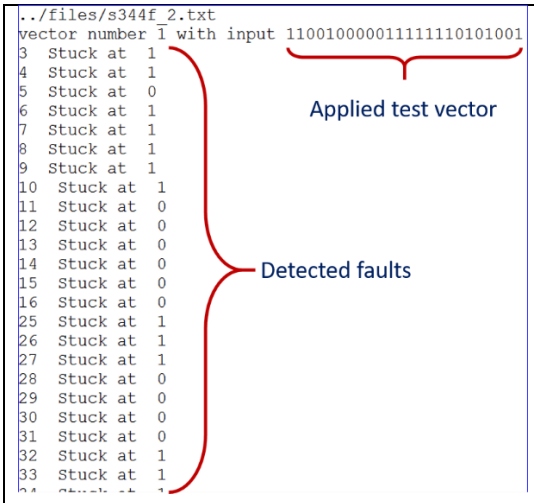
10. Finally run the following command to run the deductive simulator

```
./deductive
```

11. Output results are saved in the folder

DST_project_Mohammad_Adnaan/Deductive_FS/files/

12. The output results are saved in the files **detected_faults.txt** and **fault_coverage.txt**

	**Detected tests output file format**
../files/s349f_2.txt number of vector 1 total faults 378 detected faults 74 fault coverage 0.196 number of vector 2 total faults 378 detected faults 161 fault coverage 0.426 number of vector 3 total faults 378 detected faults 220 fault coverage 0.582 number of vector 4 total faults 378 detected faults 233 fault coverage 0.616 number of vector 5 total faults 378 detected faults 235 fault coverage 0.622 number of vector 6 total faults 378 detected faults 240 fault coverage 0.635 number of vector 7 total faults 378 detected faults 246 fault coverage 0.651 number of vector 8 total faults 378 detected faults 251 fault coverage 0.664 number of vector 9 total faults 378 detected faults 258 fault coverage 0.683 number of vector 10 total faults 378 detected faults 258 fault coverage 0.683 number of vector 11 total faults 378 detected faults 278 fault coverage 0.735 number of vector 12 total faults 378 detected faults 280 fault coverage 0.741 number of vector 13 total faults 378 detected faults 289 fault coverage 0.765 number of vector 14 total faults 378 detected faults 307 fault coverage 0.812 number of vector 15 total faults 378 detected faults 309 fault coverage 0.817 number of vector 16 total faults 378 detected faults 315 fault coverage 0.833 number of vector 17 total faults 378 detected faults 315 fault coverage 0.833 number of vector 18 total faults 378 detected faults 315 fault coverage 0.833 number of vector 19 total faults 378 detected faults 316 fault coverage 0.836 number of vector 20 total faults 378 detected faults 316 fault coverage 0.836 number of vector 21 total faults 378 detected faults 317 fault coverage 0.839 number of vector 22 total faults 378 detected faults 317 fault coverage 0.839 number of vector 23 total faults 378 detected faults 317 fault coverage 0.839	**Fault coverage output file format**

User Manual: PODEM:

The program has been built using C++. The program can be run from Linux terminal.

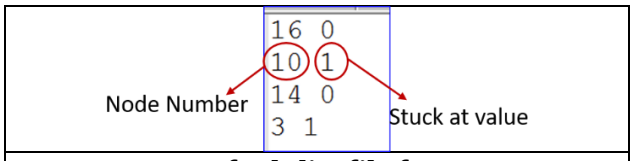
README:

1. Unzip the file ***DST_project_Mohammad_Adnaan***

2. The circuit files are kept in the following folder

```
DST_project_Mohammad_Adnaan/PODEM/files
```

3. Edit the file `input_fault_list_podem.txt` kept in this folder to give input fault list for which test vector will be generated.



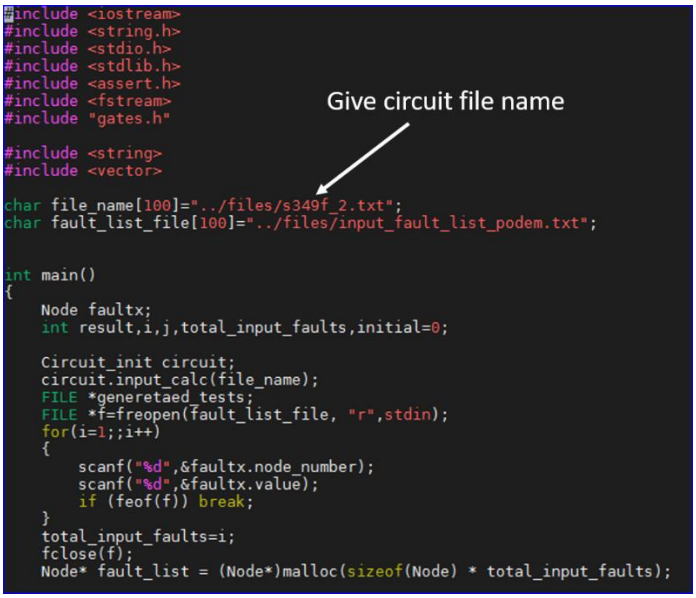
Input fault list file format

4. Navigate to the **src** folder of **PODEM** . All source code files are kept here.

```
DST_project_Mohammad_Adnaan/PODEM/src
```

5. Run command `vi podem.cpp` and press I in the keyboard to edit the main file in order to select the circuit file and mode parameters.

6. Set the desired circuit file.




```
#include <iostream>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <fstream>
#include "gates.h"

#include <string>
#include <vector>

char file_name[100]="../files/s349f_2.txt";
char fault_list_file[100]="../files/input_fault_list_podem.txt";


int main()
{
    Node faultx;
    int result,i,j,total_input_faults,initial=0;

    Circuit_init circuit;
    circuit.input_calc(file_name);
    FILE *generetaed_tests;
    FILE *f=freopen(fault_list_file, "r",stdin);
    for(i=1;;i++)
    {
        scanf("%d",&faultx.node_number);
        scanf("%d",&faultx.value);
        if (feof(f)) break;
    }
    total_input_faults=i;
    fclose(f);
    Node* fault_list = (Node*)malloc(sizeof(Node) * total_input_faults);
```

7. After setting the circuit file name press escape in keyboard, type the command `:wq` and press enter to save the changes and exit the file.

8. In the terminal run the command

```
make all
```

9. Run the following command to set execution permission for the program

```
chmod +x podem
```

10. Finally run the following command to run the deductive simulator

```
./podem
```

11. Output results are saved in the file

```
DST_project_Mohammad_Adnaan/PODEM/files/generated_tests.txt
```

12. The generated test vectors file format is as follows:

```
Test for Node 25 stuck at 1:   xxxxxxxxxxxxxxx1xxxxxxxx
Test for Node 51 stuck at 0:   00xxxxxxxxxxxxx0xxxxxxxx
Test for Node 105 stuck at 1:   101xxxx0xxxxxxxxxxxxxxxx
Test for Node 105 stuck at 0:   101xxxx1xxxxxxxxxxx0xxxx
Test for Node 83 stuck at 1:   01xx001xxx0x01x0xxxxxxxx
Test for Node 92 stuck at 0:   01xxxxx0xxxxxxxxxx0xxxxx
Test for Node 198 stuck at 0:   FAILURE
Test for Node 7 stuck at 0:   xxxxxx1xxxxxxxxxxxxxxxxx
```