Chibudem [Christian] Offodile

December 8th, 2023

PODEM

The PODEM function is a fault-oriented test vector generator based on the PODEM (Path Oriented Decision Making) test generation algorithm for digital logic circuits. The inputs to the program are a logic circuit and a target fault within the circuit. The logic circuit will be input in a text file and the text file will be processed by a separate function and passed into the PODEM function as a linked list of gates. The target fault will be input from a separate text file. The output of the PODEM function is a pointer to a test vector that detects the target fault. The test vector pointer is interrogated in the main function and printed to an output file (outputfile.txt).

Simulation Results

Circuit	Fault	Test Vector
s27.txt	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
s298f_2.txt	Net 70 s-a-1	01X1XXXXXXXXXX0XX
	Net 73 s-a-0	111XXXXXXXXXXX0XX
	Net 26 s-a-1	XX1X1XXX0XXXXXXXX
	Net 92 s-a-0	X10101XXXXXX0X0XX
	Net 38 s-a-0	01X0XXXXXXXXXX0XX
	Net 46 s-a-1	X1010XXXXXXX0X0XX
	Net 3 s-a-1	11001XXX0X0XXX0XX

	Net 68 s-a-0	X1XX1XXXXXXXX00XX
s344f_2.txt	Net 166 s-a-0	01X00XXXXX011XX0XXXXXXXX
	Net 71 s-a-1	10XXXXXXXXXXXXXXXXXXXXXX
	Net 16 s-a-0	10XXXXXXXXXXXXX1XXXXXXXX
	Net 91 s-a-1	111XXXXXXXXXXXXXXXXXXXXX
	Net 38 s-a-0	X1XXXXXXXXXXX1XXXXXXXXXX
	Net 5 s-a-1	XXXX0XXXXXXXXXXXXXXXXXXX
	Net 138 s-a-0	01XX00XXXX0X11X0XXXXXXXX
	Net 91 s-a-0	10XXXXXXXXXXXXXXXXXXXXXX
s349f_2.txt	Net 25 s-a-1	XXXXXXXXXXXXXXX1XXXXXXXX
	Net 51 s-a-0	00XXXXXXXXXXXXX0XXXXXXXX
	Net 105 s-a-1	01X11XXXXX010XX0XXXXXXXX
	Net 105 s-a-0	01X1XXXXXX1XXXX0XXXXXXXX
	Net 83 s-a-1	01XX000XXX0X0X10XXXXXXXX
	Net 92 s-a-0	01XX001XXX0X01X0XXXXXXXX
	Net 7 s-a-0	XXXXXX1XXXXXXXXXXXXXXXXX
	Net 179 s-a-0	101XXXXXXXXXXXX0XXXXXXXX

User Manual: Fault Simulator and PODEM

Note:
The deductive fault simulator and the PODEM program are implemented in a single program, however they will be treated as separate for the purposes of this document. Two separate project folders (with identical code) will also be submitted for the deductive fault simulator and PODEM. The different functions can be selected by user commands and the contents of the input files.

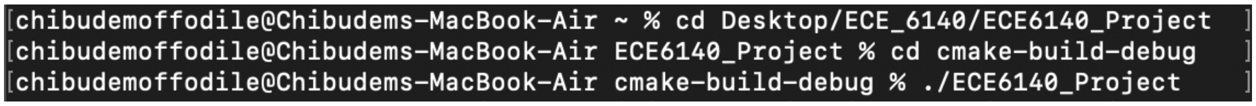
User Manual: Deductive Fault Simulator

Simulation Environment:
The program was developed in the CLion IDE using C++ (2017 version). The project files can be imported into an IDE and run using C++17. Alternatively, the program executable can be run in a command line interface such as the terminal app on MAC or PowerShell/Command Prompt in windows.

README
There are 3 implementation files and 2 header files associated with the deductive fault simulator. The implementation files are main.c, podem.c, and Classes.c. The header files are podem.h and Classes.h. The deductive fault simulator includes four main modes. In this section we will only discuss the three modes that pertain to the deductive fault simulator. The fourth mode is PODEM and will be discussed in the section titled *User Manual: PODEM*. There are also three mandatory text files included in the project folder: *infault.txt, userVector.txt,* and *outputfile.txt*. These text files allow the user to interact with the program and must not be deleted. A new text file with the same name can be added to the *cmake-build-debug* folder of the project if any text file is deleted.

The Deductive Fault Simulator can be run in five steps:

1. Import the program into an IDE or a command line interface (Terminal or Command Prompt) and run the program using C++17. The process for running the program in a command line interface will be focused on because the IDE process is simpler and varies between different IDEs. In a command line interface, the program can be run by moving to the directory that contains the program executable and running the executable as shown below. The program executable is called *ECE6140_Project* and it is contained in the *cmake-build-debug* folder within the main *ECE6140_Project* folder. Per the project directives, the main *ECE6140_Project* folder will be renamed *Deductive_FS*.

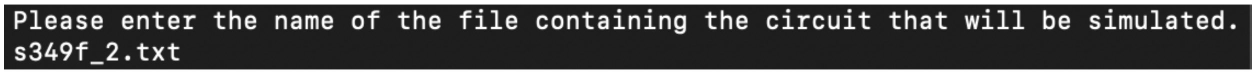


```
[chibudemoffodile@Chibudems-MacBook-Air ~ % cd Desktop/ECE_6140/ECE6140_Project   ]
[chibudemoffodile@Chibudems-MacBook-Air ECE6140_Project % cd cmake-build-debug    ]
[chibudemoffodile@Chibudems-MacBook-Air cmake-build-debug % ./ECE6140_Project     ]
```

Figure 1. Running the program in Terminal (MACOS)

2. Once the program is running, the user will be prompted to input a file name. As shown below, the user should enter the name of the file that contains the circuit that the

deductive fault simulation will be run on. This file must be placed in the *cmake-build-debug* folder. Some default circuit files are already contained in the folder.



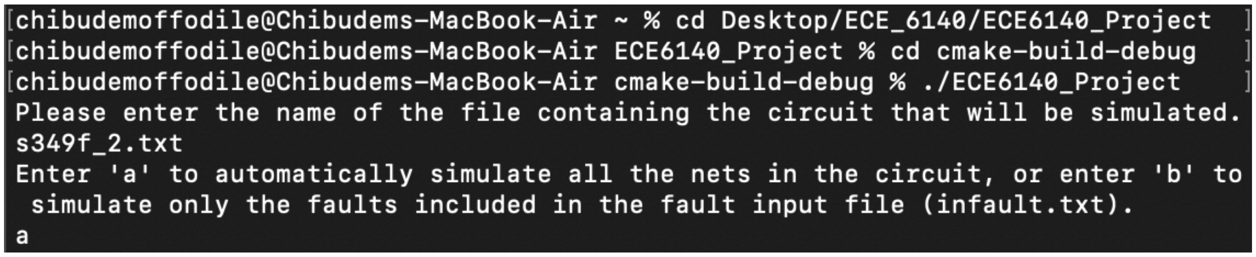
```
Please enter the name of the file containing the circuit that will be simulated.
s349f_2.txt
```

Figure 2. Prompt 1

3. Once the file name has been entered, the program asks the user to choose whether to simulate all the faults in the circuit (enter *a*) or just the faults included in the input fault file (enter *b*). To test the deductive fault simulator, the user can always choose *a*. However, option *b* can be used for running the deductive fault simulator with specific target faults. The target faults must be entered into the *infault.txt* file prior to choosing option *b*. The target faults must be entered in the following format:

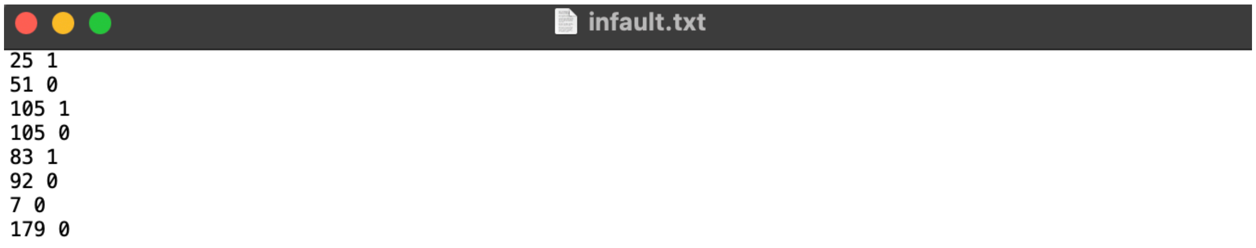
   12 0

   Where 12 is the ID/number of the wire that the fault is on, and 0 refers to the fault value (either stuck at 0 or stuck at 1). As shown in *Figure 4* below, each fault should exist on a separate line and there should be no empty lines above any faults.



```
[chibudemoffodile@Chibudems-MacBook-Air ~ % cd Desktop/ECE_6140/ECE6140_Project  ]
[chibudemoffodile@Chibudems-MacBook-Air ECE6140_Project % cd cmake-build-debug   ]
[chibudemoffodile@Chibudems-MacBook-Air cmake-build-debug % ./ECE6140_Project    ]
Please enter the name of the file containing the circuit that will be simulated.
s349f_2.txt
Enter 'a' to automatically simulate all the nets in the circuit, or enter 'b' to
 simulate only the faults included in the fault input file (infault.txt).
a
```

Figure 3. Prompt 2



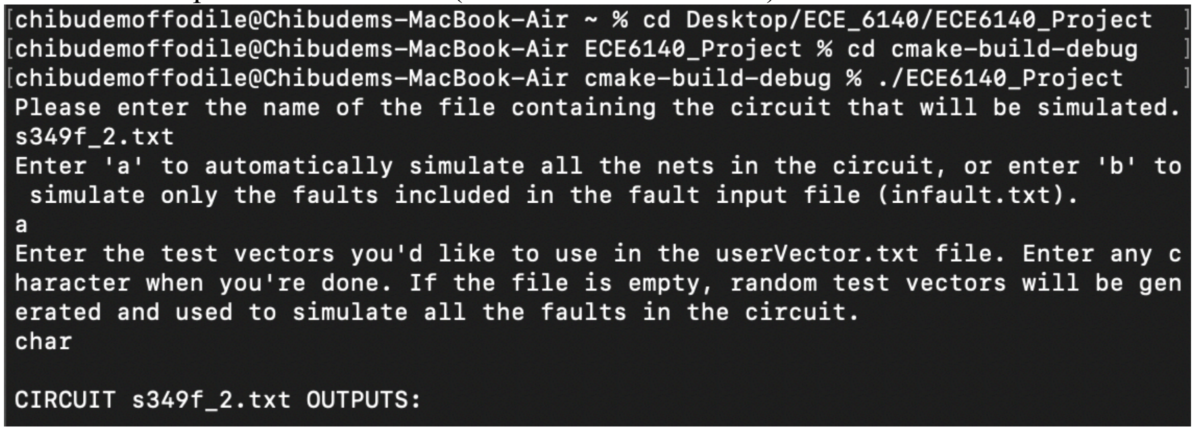
infault.txt

```
25 1
51 0
105 1
105 0
83 1
92 0
7 0
179 0
```

Figure 4. Example of infault.txt content

4. The program asks the user to enter the test vectors that will be used in the deductive fault simulation into the *userVector.txt* file. The test vector bits should be entered in the *userVector.txt* file in the order that the input wire numbers are defined in the circuit input file. There should be no spaces between the bits of the test vectors, each test vector should be on its own line, and there should be no empty lines above any test vectors. The input test vectors should be entered in binary (no X input, only 0 and 1). The program

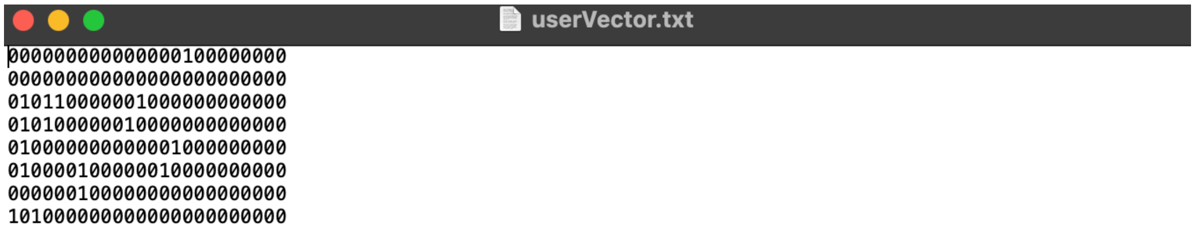
waits for the user to enter any character in the console before it continues. The wait allows the user to make changes to the *userVector.txt* file before continuing. If the *userVector.txt* file is empty and the user entered *a* for the second prompt, random test vectors will be generated and used to run the deductive fault simulator until 95% fault coverage is reached or 10000 test vectors have been used. If the user entered *b*, the user should provide test vectors (in the *userVector.txt* file) to run the deductive fault simulator.



```
[chibudemoffodile@Chibudems-MacBook-Air ~ % cd Desktop/ECE_6140/ECE6140_Project  ]
[chibudemoffodile@Chibudems-MacBook-Air ECE6140_Project % cd cmake-build-debug   ]
[chibudemoffodile@Chibudems-MacBook-Air cmake-build-debug % ./ECE6140_Project    ]
 Please enter the name of the file containing the circuit that will be simulated.
 s349f_2.txt
 Enter 'a' to automatically simulate all the nets in the circuit, or enter 'b' to
  simulate only the faults included in the fault input file (infault.txt).
 a
 Enter the test vectors you'd like to use in the userVector.txt file. Enter any c
 haracter when you're done. If the file is empty, random test vectors will be gen
 erated and used to simulate all the faults in the circuit.
 char

 CIRCUIT s349f_2.txt OUTPUTS:
```

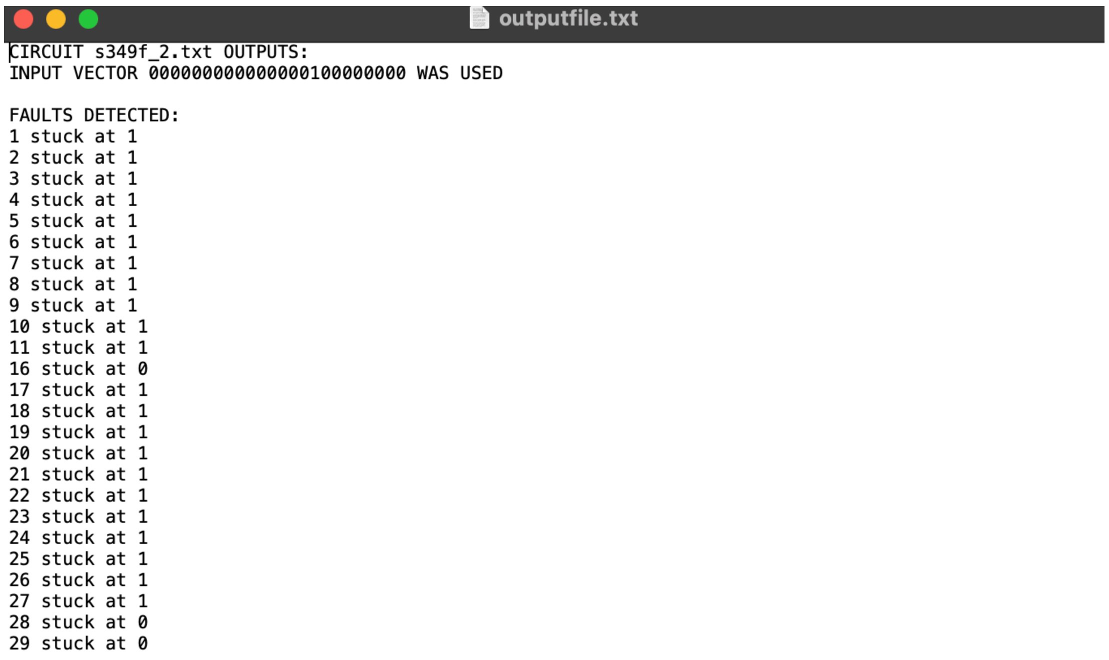
Figure 5. Prompt 3



```
000000000000000100000000
000000000000000000000000
010110000001000000000000
010100000010000000000000
010000000000001000000000
010000100000010000000000
000000100000000000000000
101000000000000000000000
```

Figure 6. Example of userVector.txt content

5. After a character is entered in step 4, the program will run. Some peripheral information will be output on the console, but the main program outputs will be in the *outputfile.txt* file. An example of the program outputs is shown below. The program typically takes one second to run.



```
CIRCUIT s349f_2.txt OUTPUTS:
INPUT VECTOR 00000000000000100000000 WAS USED

FAULTS DETECTED:
1 stuck at 1
2 stuck at 1
3 stuck at 1
4 stuck at 1
5 stuck at 1
6 stuck at 1
7 stuck at 1
8 stuck at 1
9 stuck at 1
10 stuck at 1
11 stuck at 1
16 stuck at 0
17 stuck at 1
18 stuck at 1
19 stuck at 1
20 stuck at 1
21 stuck at 1
22 stuck at 1
23 stuck at 1
24 stuck at 1
25 stuck at 1
26 stuck at 1
27 stuck at 1
28 stuck at 0
29 stuck at 0
```

Figure 7. Example of deductive fault simulator outputs

User Manual: PODEM

Simulation Environment:

The program was developed in the CLion IDE using C++ (2017 version). The project files can be imported into an IDE and run using C++17. Alternatively, the program executable can be run in a command line interface such as the terminal app on MAC or PowerShell/Command Prompt in windows.

README

There are 3 implementation files and 2 header files associated with the PODEM program. The implementation files are main.c, podem.c, and Classes.c. The header files are podem.h and Classes.h. There are also three mandatory text files included in the project folder: *infault.txt, userVector.txt,* and *outputfile.txt*. These text files allow the user to interact with the program and must not be deleted.

The PODEM program can be run in five steps:

1. Import the program into an IDE or a command line interface (Terminal or Command Prompt) and run the program using C++17. In a command line interface, the program can be run by moving to the directory that contains the program executable and running it as shown in *Figure 1*. The program executable is called *ECE6140_Project* and it is contained in the *cmake-build-debug* folder within the main *ECE6140_Project* folder. Per the project directives, the main *ECE6140_Project* folder will be renamed *PODEM*.

2. Once the program is running, the user will be prompted to input a file name. As shown in *Figure 2*, the user should enter the name of the file that contains the circuit that the PODEM program will be run on. This file must be placed in the *cmake-build-debug* folder.

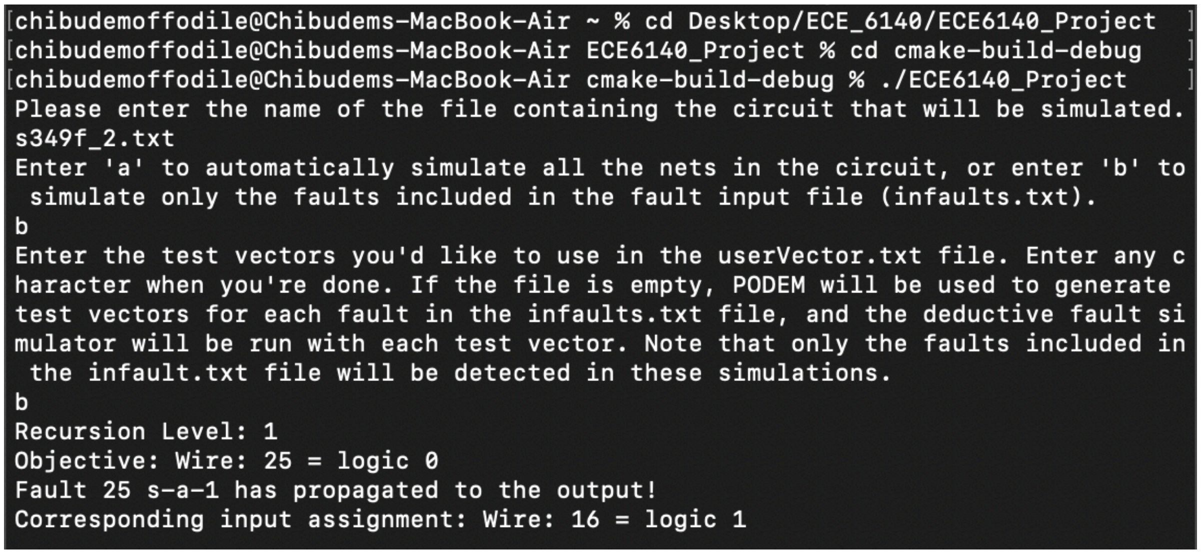
3. Once the file name has been entered, the program asks the user to choose whether to simulate all the faults in the circuit (enter *a*) or just the faults included in the input fault file (enter *b*). To test PODEM, the user **must** choose *b*. The PODEM program will generate test vectors for all the faults in *infault.txt*. The target faults must be entered into the *infault.txt* file prior to choosing option *b*. The target faults must be entered in the following format:

   12 0

   Where 12 is the ID/number of the wire that the fault is on, and 0 refers to the fault value (either stuck at 0 or stuck at 1). As shown in *Figure 4*, each fault should exist on a separate line and there should be no empty lines above any faults.

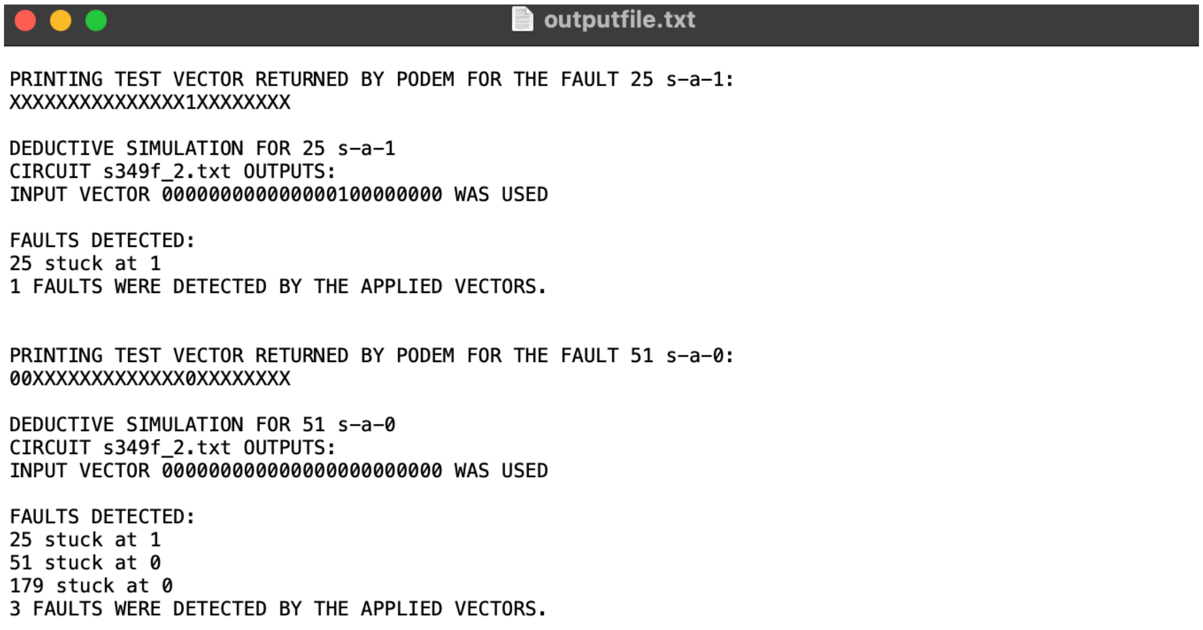
4. The program asks the user to enter the test vectors that will be used in the simulation into the *userVector.txt* file. To run PODEM, the *userVector.txt* file **must** be empty. The program waits for the user to enter any character in the console before it continues.

5. Once a character is entered in step 4, the program will run. After a test vector is generated for a given fault using PODEM, the deductive fault simulator is run using the test vector to demonstrate that the test actually detects the fault. All the X values in the PODEM test vector are replaced with 0s for the deductive fault simulation. Note that since the user entered b, the deductive fault simulation done with the PODEM generated test vectors will only detect faults included in the *infault.txt* file. Some peripheral information will be output on the console, but the main program outputs will be in the *outputfile.txt* file. An example of the commands needed to run PODEM are shown in *Figure 8*, and an example of the program output is shown in *Figure 9*. The program typically takes one second to run.



```
[chibudemoffodile@Chibudems-MacBook-Air ~ % cd Desktop/ECE_6140/ECE6140_Project  ]
[chibudemoffodile@Chibudems-MacBook-Air ECE6140_Project % cd cmake-build-debug   ]
[chibudemoffodile@Chibudems-MacBook-Air cmake-build-debug % ./ECE6140_Project    ]
Please enter the name of the file containing the circuit that will be simulated.
s349f_2.txt
Enter 'a' to automatically simulate all the nets in the circuit, or enter 'b' to
 simulate only the faults included in the fault input file (infaults.txt).
b
Enter the test vectors you'd like to use in the userVector.txt file. Enter any c
haracter when you're done. If the file is empty, PODEM will be used to generate
test vectors for each fault in the infaults.txt file, and the deductive fault si
mulator will be run with each test vector. Note that only the faults included in
 the infault.txt file will be detected in these simulations.
b
Recursion Level: 1
Objective: Wire: 25 = logic 0
Fault 25 s-a-1 has propagated to the output!
Corresponding input assignment: Wire: 16 = logic 1
```

Figure 8. Commands and user inputs for running PODEM



outputfile.txt

```
PRINTING TEST VECTOR RETURNED BY PODEM FOR THE FAULT 25 s-a-1:
XXXXXXXXXXXXXXX1XXXXXXXX

DEDUCTIVE SIMULATION FOR 25 s-a-1
CIRCUIT s349f_2.txt OUTPUTS:
INPUT VECTOR 000000000000000100000000 WAS USED

FAULTS DETECTED:
25 stuck at 1
1 FAULTS WERE DETECTED BY THE APPLIED VECTORS.


PRINTING TEST VECTOR RETURNED BY PODEM FOR THE FAULT 51 s-a-0:
00XXXXXXXXXXXXX0XXXXXXXX

DEDUCTIVE SIMULATION FOR 51 s-a-0
CIRCUIT s349f_2.txt OUTPUTS:
INPUT VECTOR 000000000000000000000000 WAS USED

FAULTS DETECTED:
25 stuck at 1
51 stuck at 0
179 stuck at 0
3 FAULTS WERE DETECTED BY THE APPLIED VECTORS.
```

Figure 9. Example of PODEM program outputs