Evan Apinis
ECE 5724 Homework 4
3/15/24

1. Introduction

For this assignment, we were tasked with performing fault simulations for a circuit containing a mixture of combinational and sequential logic. This was done by separating the combinational and sequential components before using a virtual testing Verilog testbench to test the SSC circuit on a list of faults. The assignment provided the netlist and bench files for the SSC design, but they would require modifications before completing the test bench. The following steps were assigned to us for this homework.

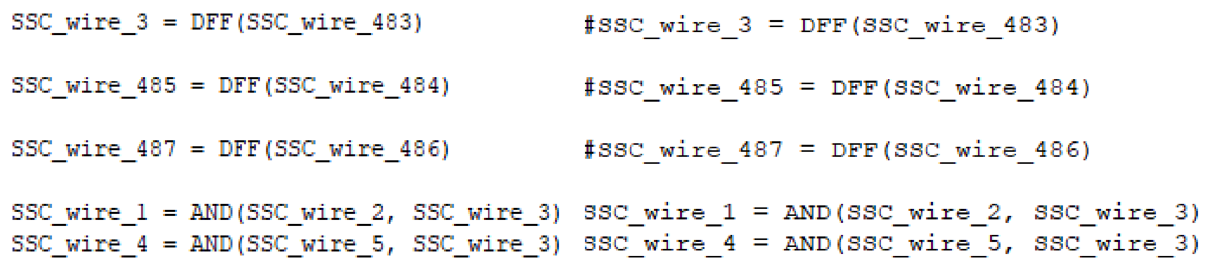
- Unfold "SSC.bench" file and separate the combinational part as discussed in the course lectures.
- Apply Atalanta to the unfolded file to generate a good test for the combinational part of the circuit.
- Insert full-scan into the SSC_net module in the "netlist_SSC_V1.v" file.
- Develop a virtual tester to verify the generated test. Write down the reported number of collapsed faults, number of detected faults, and fault coverage.

2. Test Generation Processes

Before being able to run the testbench for the provided SSC circuit several steps were completed along the way. The first step was to prepare the provided bench file to be able to generate appropriate test vectors. This involved unfolding the bench file and then using Atalanta to generate test vectors and the CUT responses for the bench circuit. The final step was to prepare the netlist for simulation by implementing full-scan insertion on the flip-flops.

2.1. Unfolding Bench File

To generate test vectors for the SSC module effectively and achieve a high fault coverage the design should have its sequential and combinational components separated. This unfolding of the circuit was done in the SSC.bench file provided to us. To unfold the circuit the sequential elements, being the flip flops, were removed from the bench file. A before and after screenshot of this process is shown in Figure 1 with the unfolded file shown on the right.



```
SSC_wire_3 = DFF(SSC_wire_483)             #SSC_wire_3 = DFF(SSC_wire_483)

SSC_wire_485 = DFF(SSC_wire_484)           #SSC_wire_485 = DFF(SSC_wire_484)

SSC_wire_487 = DFF(SSC_wire_486)           #SSC_wire_487 = DFF(SSC_wire_486)

SSC_wire_1 = AND(SSC_wire_2, SSC_wire_3)   SSC_wire_1 = AND(SSC_wire_2, SSC_wire_3)
SSC_wire_4 = AND(SSC_wire_5, SSC_wire_3)   SSC_wire_4 = AND(SSC_wire_5, SSC_wire_3)
```

Figure 1 – Unfolding of SSC.bench With Removal of DFF

The next step in this process was to make pseudo inputs and outputs for the module. With the removal of the flip-flops, the nets connected to the DFF need to be dealt with to avoid having floating nets. The proper way to do this would be to make them visible as inputs and outputs to allow control of them for simulating the combinational logic they are connected to. To do this the outputs of the DFF are turned into pseudo primary inputs (PPI) which allows for a testbench or simulation tool to set an input value and allow the gates connected to this net to be tested. The DFF inputs are turned into pseudo primary outputs (PPO) to allow for the gates preceding this net to be tested by simulation tools. Figure 2 demonstrates some of these PPIs and PPOs added to the bench file.

```
INPUT (SSC_wire_3)     OUTPUT (SSC_wire_483)
INPUT (SSC_wire_485)   OUTPUT (SSC_wire_484)
INPUT (SSC_wire_487)   OUTPUT (SSC_wire_486)
```

Figure 2 – Unfolded SSC.bench PPI and PPO

2.2. Atalanta Test Generation

With the unfolded bench file, Atalanta was applied to generate a set of test vectors. In previous assignments where Atalanta was used the command (atalanta-M -t c432.pat -W 1 c432.bench) was used to generate a test pattern for the given bench file. In this case, however, we are looking to stimulate the PPI and PPO of the bench file. To do this I used the following Atalanta command, (atalanta-M -t c432.pat -W 2 c432.bench). This command will generate the CUT response for the test vector being applied. This means that the SSC.tst file used in the testbench will contain the primary inputs (PI) and PPI as part of its test vector. It will also contain the primary outputs (PO) and PPO responses which can be used to validate the fault list generated by Atalanta. Figure 3 demonstrates how the SSC.tst stimulus file is formatted to be applied to the virtual test. The st_size label refers to the number of flip-flops because these are the PPI and PPO generated from each flip-flop to be chained together later.

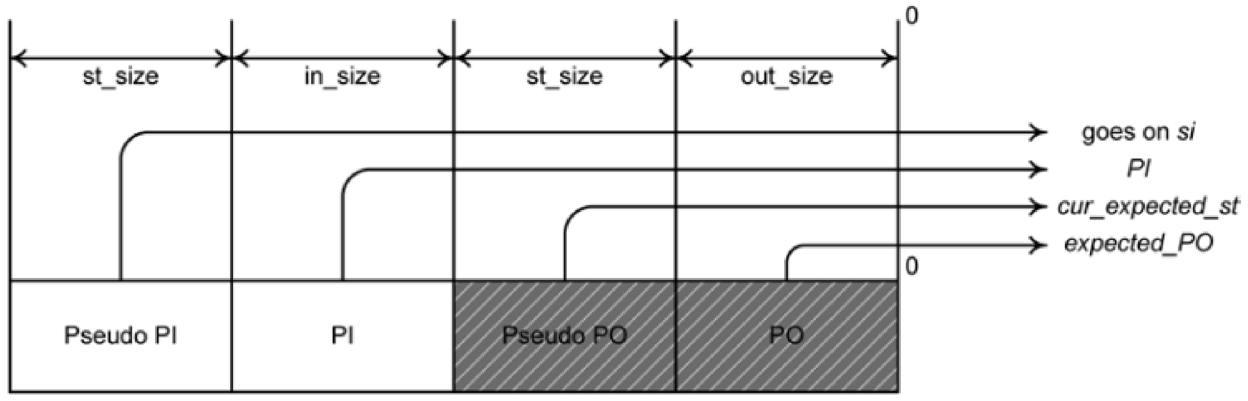


Figure 3 – Test Vector Format of SSC.tst

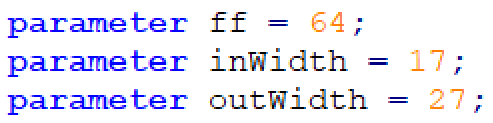
In total 73 test vectors were created in the process. The second responsibility of Atalanta was to generate the fault list. The list created contained a total of 1041 collapsed faults.

2.3. Full-Scan Insertion

To complete the virtual tester the netlist was first modified to enable the ability to test the sequential elements with scan chains. To do this the flip-flops used should be replaced with specialized flip-flops that allow for scanning and additional control. In this assignment, the flip-flops used in the netlist were from the primitives Verilog file provided to us which already contained the scan flip-flops. What had not been done however was the implementation of the scan chain. To do this the flip-flops were fed into one another forming a chain where the first flip-flop uses the Si input, and the output of the last flip-flop is assigned to So. This allows to serial shift in the inputs from the test vector generation and serially reading the output for verification and comparison.

3. Virtual Tester

The virtual tester testbench was designed to simulate the Verilog module containing flip-flops using the full scan insertion and unfolded test generation methods. The testbench begins by declaring several parameters that are used for indexing the test vectors created by Atalanta. The first is ff which stands for the number of flip-flops in the scan chain. In our instance, there are 64 flip-flops in the scan chain from the previous step. The next parameters are the input and output width and these stand for the number of primary inputs and primary outputs of the module, it does not include the clock and reset because these are supplied by the testbench. It also does not include the PPI and PPO from the unfolding because these are not present in the netlist and were only used to generate appropriate combinational tests. Figure 4 shows these parameters in the testbench.



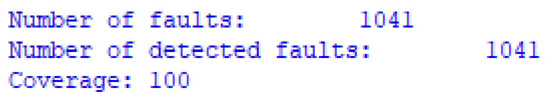
```
parameter ff = 64;
parameter inWidth = 17;
parameter outWidth = 27;
```

Figure 4 – Virtual Tester Parameters

The next portion of the testbench establishes several registers and wires to be used later on in the testbench. The most notable ones are the registers and wires that are used to store the different sections of the Atalanta test generation. These store the PO, PI scan chain input, and expected output and are sized using the appropriate parameters.

In the actual testing portion of the virtual tester, the testbench begins with collapsing the fault list created by Atalanta. This process resulted in a fault list containing 1041 collapses. The tester then continues into a set of two while loops. The outer loop will apply and remove faults while counting the number of detected faults based on results from the inner loop where the tests are applied and the results are captured. In the fault loop, the fault is injected from the collapsed list and many of the registers containing values from the test vectors are reset. In the testing loop, the tester will read a line from the test vector file and separate its values into several registers for PI, PO, and scan chain inputs based on parameter indexing. Then it will enter a repeat statement that will shift the inputs for the scan chain based on the number of flip-flops. Once this is done the tester will compare the output of the FUT and compare it to the expected output of the test vector. Based upon the results of this evaluation a flag will be set for further processing that will increase the counter for detected faults. Following all of this the fault is removed and the total number of faults tested is increased. Then the loop will continue until all cases have been exhausted.

After running the simulation on Modelsim the testbench reports the number of faults and the number of detected faults in addition to the coverage. It was mentioned previously that the fault list contained 1041 collapsed faults. The testbench reported 1041 faults indicating that it ran through all of the tests. In addition, by comparing the outputs from the test vectors and the faulty output, the tester found 1041 faults meaning that the coverage achieved was 100%.



```
Number of faults:        1041
Number of detected faults:        1041
Coverage: 100
```

Figure 5 – Virtual Tester Results