Evan Apinis
ECE 5724 Homework 2
2/18/24

1. Introduction

For this assignment, we were tasked with performing fault collapsing on the c5315 benchmark circuit. This circuit is one of the ISCAS-85 benchmarks, a benchmark containing combinational logic. In this assignment, we were provided with several starting files, for the c5315 benchmark we had the c5315 Verilog netlist made from primitive components such as an and gate or inverter. Additionally, we were provided with a netlist equivalent c5315 bench format file. Two important pieces of software for this assignment were Atalanta and HOPE. Atalanta is an automatic test pattern generator that can also complete fault simulations for stuck-at faults in combinational circuits. HOPE is a pure fault simulation software that can be used for combinational and synchronous sequential circuits in a parallel fashion. Using the provided pieces of software and c5315 benchmarking files the following tasks were assigned.

- Apply Atalanta to c5315.bench file to generate a test set for the circuit. Store the resulting test vectors in a file named “c5315.pat”. Write down the reported number of collapsed faults, number of detected faults, and fault coverage for future comparisons.
- Apply HOPE to the c5315.bench file and use “c5315.pat” as the input test set. Write down the reported number of collapsed faults, number of detected faults, and fault coverage for future comparisons.
- Write a Verilog testbench that performs serial fault simulation for the module c5315_net using c5315.pat as the circuit test set. Your testbench must at least report the number of collapsed faults, number of detected faults, and fault coverage.
- Write a Verilog code generating fault dictionary for c5315_net, use c5315.pat as the test set.

2. Atalanta Simulation

The Atalanta simulation tool was used to generate a test pattern from the c5315 netlist equivalent bench file. This test pattern would be used in the future to perform serial fault simulations both by the simulation software, and the Verilog testbench. The Atalanta simulation tool can also perform fault simulations after the test pattern has been generated. Both the test pattern and fault lists can be generated in the same run using the following command.

atalanta-M -t c5315.pat -W 1 -F c5315.flt c5315.bench

The result was a test pattern file containing 124 test patterns with each pattern containing 178 bits, matching the input bit width of the c5315 benchmarks netlist. It also generated a fault list file after completing the test pattern generation. This fault list contained 5350 total faults and included the design nets where these faults occur. Using all of this Atalanta can generate a reports file to give numerical detail to the number of faults detected by adding the “-P c5315.rep” argument to the command line. A screenshot of this report shown in Figure 1 shows that there were 5350 collapsed faults. It detected 5291 faults for a coverage of 98.897%. The report also

notes that there were 59 redundant faults identified. One thing observed in the report is that although the test patterns file produced by Atalanta had 124 patterns the report indicates that the final number of test patterns used for fault simulations was 115. The documentation, however, does note that this number is generated after the compaction of test patterns, therefore reducing the number of patterns applied.

```
gates: 2307
iv: 178
ov: 123
i_patterns: 218
patterns: 115
faults: 5350
d_faults: 5291
r_faults: 59
time: 0.016
```

Figure 1 – Atalanta Fault List Report

3. Hope Simulation

The HOPE simulation tool is a parallel fault simulator that uses the test patterns generated by Atalanta to complete its analysis. One thing noted in the assignment was that the input format used by HOPE varies slightly from that generated by Atalanta. The test pattern file c5315.pat is generated by Atalanta in the following format.

110110110110001011001000010111000110101001110010111011101000101111001111…
100011010101000000111010101111100001010100100111111010110100010110110011…

These test patterns contain just the input bits to be applied to the FUT or GUT design being tested. The HOPE tool requires that there is a starting indicator for each line that dictates the line number. Therefore, a second test patterns file was created that was identical to the first but with the following changes.

1: 110110110110001011001000010111000110101001110010111011101000101111001111…
2: 100011010101000000111010101111100001010100100111111010110100010110110011…

One additional modification was needed to the bench file used in the Atalanta test pattern generation and simulation. The HOPE tool follows the ISCAS-89 netlist format except for the caveat that the first line of the bench file should be # followed by the name of the circuit. Without this modification, the HOPE software would not run. The results of the simulation, shown in Figure 2, are very similar to the Atalanta result of Figure 1. It was reported that there were 5350 collapsed faults with the number of detected faults being 5291. These are the exact results provided by Atalanta which led to a coverage of 98.897% for the c5315 benchmark. For the HOPE simulation the full 124 patterns were applied that had been generated by Atalanta, rather than the compressed 115 seen previously.

```
******   SUMMARY OF SIMULATION RESULTS   ******
1. Circuit structure
   Name of circuit                          : c5315
   Number of primary inputs                 : 178
   Number of primary outputs                : 123
   Number of flip-flops                     : 0
   Number of gates                          : 2307
   Level of the circuit                     : 49

2. Simulator input parameters
   Simulation mode                          : file (c5315.pat)

3. Simulation results
   Number of test patterns applied          : 124
   Fault coverage                           : 98.897 %
   Number of collapsed faults               : 5350
   Number of detected faults                : 5291
   Number of undetected faults              : 59

4. Memory used                              : 15482 Kbytes

5. CPU time
   Initialization                           : 8697491.667 secs
   Fault simulation                         : 0.250 secs
   Total                                    : 8697491.917 secs
```

Figure 2 – HOPE Fault Simulation Results

4. Serial Fault Simulation

The next portion of the assignment was to move away from using fault simulation software and implement a serial fault simulation technique in Verilog using a testbench. Simulations for the testbench were performed in ModelSim with the addition of two DLL files, FaultCollapsing.dll and FaultInjection.dll, whose functions will be described in further detail later. The beginning basis of this testbench was to instantiate two identical copies of the c5315 netlist Verilog module. One of these was instantiated as the GUT or “golden model” while the other was the FUT or module to be injected with faults. The purpose of this testbench design is to allow for the comparison of module outputs between the two whilst injecting faults and computing test pattern outputs. This methodology was described in the lecture with the following image seen in Figure 3.

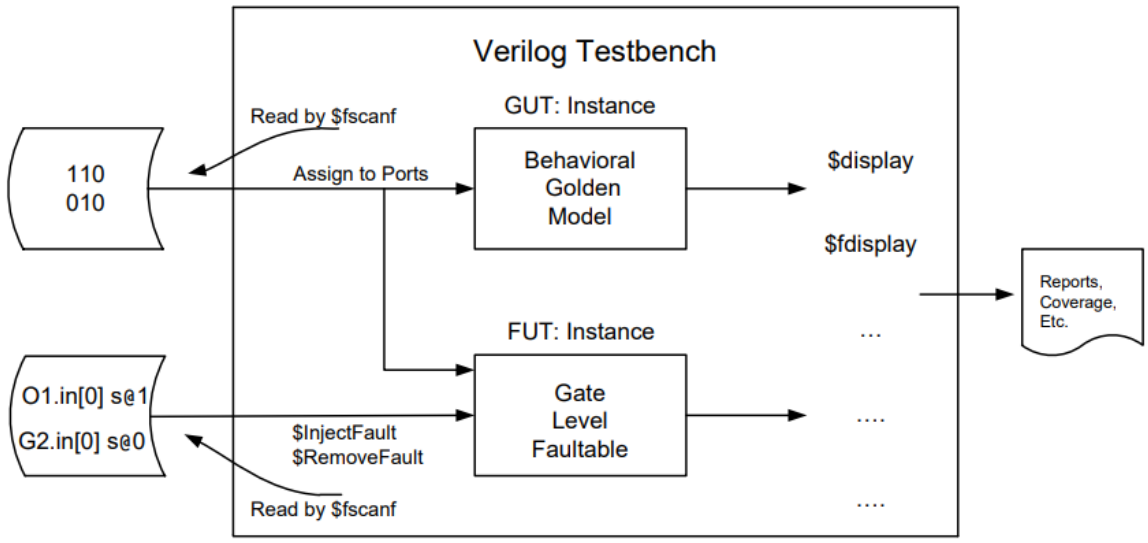

Figure 3 – Testbench Hierarchical Design for Serial Fault Simulation

To complete the fault simulation in the testbench the fault list generated by Atalanta needed to be converted to a format that is friendly to the $fscanf system task. This is done by using the $FaultCollapsing system task added by its associated DLL. This system task is used to read the fault list and convert it to a format that can be easily read by a Verilog simulation tool such as ModelSim. It does this by converting the original netlist name to the associated net name in the testbench hierarchy. At this point, the fault testing will begin by looping through the length of the fault list. During each loop, the testbench will inject the given fault into the FUT and then apply the set of test patterns to both the GUT and FUT to compare outputs. A code snippet is provided in Figure 4 to demonstrate the application of each fault.

```
status = $fscanf (faultFile, "%s s@%b\n", wireName, stuckAtVal);
$InjectFault ( wireName, stuckAtVal);
testFile = $fopen ("c5315.pat", "r");
```

Figure 4 – Testbench Code Snippet for Serial Fault Injecting

The testbench will continue to apply every test pattern to every fault in the fault list until it has finished every test. During this process, there will be two counters incremented. The first is the number of collapsed faults and the second is the number of detected faults. The collapsed fault counter increments after every fault have been removed and the detected faults increments whenever the output of the GUT and FUT are not equal, but only up to once per applied fault. These two metrics are used to generate the coverage percentage seen earlier from Atalanta and HOPE. The results of this testbench are shown in Figure 5. Here it can be seen that 5104 faults were injected to the FUT with 5045 faults detected. This provides a coverage of 98.84%, a close number to the 98.897% of HOPE. One aspect to notice is that the $FaultCollapse system task decreased the number of faults from 5350 to 5104.

```
# Number of Collapsed Faults:      5104 Number of Detected Faults:      5045 Fault Coverage: 0.988440
```

Figure 5 – Serial Fault Injection Testbench Results

5. Fault Dictionary

The final requirement of the assignment was to create a fault dictionary generated by the Verilog testbench for the c5315 netlist. This did not require too many additional modifications to the serial fault simulation testbench in the previous part. The main addition was to add additional logic for storing the result of every test pattern applied to each given fault. This result is not the FUT output itself but rather if the FUT and GUT had the same output for each test pattern at the given fault. A snippet of this testbench code will be provided in Figure 6 with the dictionary shown in Figure 7. In the code snippet, the wireName is the netlist name and forced fault value, with the syndrome being the compilation of fault comparison results described just before. The easiest way to see this is by viewing Figure 7. The net name is shown and then separated from the syndrome with a comma. It is also important to note that the screenshot has been cropped due to the length of the syndrome and the file is extended to the right further than shown. The fault dictionary also prints the coverage calculated in the serial fault simulation portion of the code.

```
$fwrite (dictionaryFile, "%s, %b \n", wireName, syndrome);
```

Figure 6 – Verilog Fault Dictionary Code Snippet

```
5087                c5315_tb.FUT.OR_211.in[1], 0000000000000100010001000000000000000000000000000000000000010000000000
5088                c5315_tb.FUT.OR_211.in[0], 0001000000000010000000000000000000000000000000000000000100000000000000
5089                c5315_tb.FUT.OR_212.in[3], 0010000000000000000000000000000000000000000000010000000000000000000000
5090                c5315_tb.FUT.OR_212.in[2], 0000000101000010000000010000010000000000000000000000000000000000000100
5091                c5315_tb.FUT.OR_212.in[1], 0000000000000000000000000000000000000000000000010000010000000000000000
5092                c5315_tb.FUT.OR_212.in[0], 0000001000000000100000000001000000000000010000010000000000000000000000
5093                c5315_tb.FUT.OR_213.in[3], 0000001000100000000010001010000010010000000000000000000000011001000000
5094                c5315_tb.FUT.OR_213.in[2], 0000000000000001000000110000001000000010000010000001000100000000100000
5095                c5315_tb.FUT.OR_213.in[1], 0001000100000000100000000100000000000000100000000000000100000000000110
5096                c5315_tb.FUT.OR_213.in[0], 1000000001001010000000000000000000001000011000000100000010000000000000
5097                c5315_tb.FUT.OR_214.in[3], 0010000000000010000000010000000000010000001010000010000000001000000000
5098                c5315_tb.FUT.OR_214.in[2], 0000000101100000000000000010011011000000000000100100000010000000000001
5099                c5315_tb.FUT.OR_214.in[1], 1001001000001000100000000100000000000001000000010000100000001010100000
5100                c5315_tb.FUT.OR_214.in[0], 0000000010000001010010101000000000000000100000010000010000100000010000
5101              c5315_tb.FUT.AND_717.in[1], 0010010001000000000100010000000000101100110010000000000010000000100100
5102              c5315_tb.FUT.AND_717.in[0], 0000100000100001000000001110000100010000001000001100000100001010100000
5103              c5315_tb.FUT.AND_718.in[1], 0001010000000100010111000000000000101110100000000001000010100000000101
5104              c5315_tb.FUT.AND_718.in[0], 1000100000000001000000101100001010010001001001000100001010000101110100010
5105 Coverage: 0.988440
```

Figure 7 – Fault Dictionary Result