Special Topics in Digital Systems Design

Papageorgiou Spyridon 2nd Laboratory Exercise

Table of Contents

Note

The necessary files for each question along with the logs are located in the ***./requested/question/*** folder.

Special Topics in Digital Systems Design

Papageorgiou Spyridon 2nd Laboratory Exercise

Question 1

1.

By performing synthesis and fault simulation, we observe that by using the test vectors stored in the c1355_test_set.v file we achieve a coverage of 99.33%. The results are not what we expect since we detect only the faults for which their corresponding test vectors were found. As shown in the c1355_test_set.v file, vectors were generated for 1614 faults while when we add faults -all for the simulation we see that there are 1650 faults. So, instead of having 3 undetected faults, we have 11.

2.

With these commands:

- write faults uncolapsed_detected_faults.dat -class DT -uncollapsed -replace
- write faults colapsed_detected_faults.dat -class DT -collapsed -replace

We save the detected faults by applying fault collapsing and non-fault collapsing. The uncollapsed faults are 1639 while the collapsed ones are 1054, i.e. we have a reduction of about 35%. The list of undetected faults can be found in the not_detected_faults.dat file.

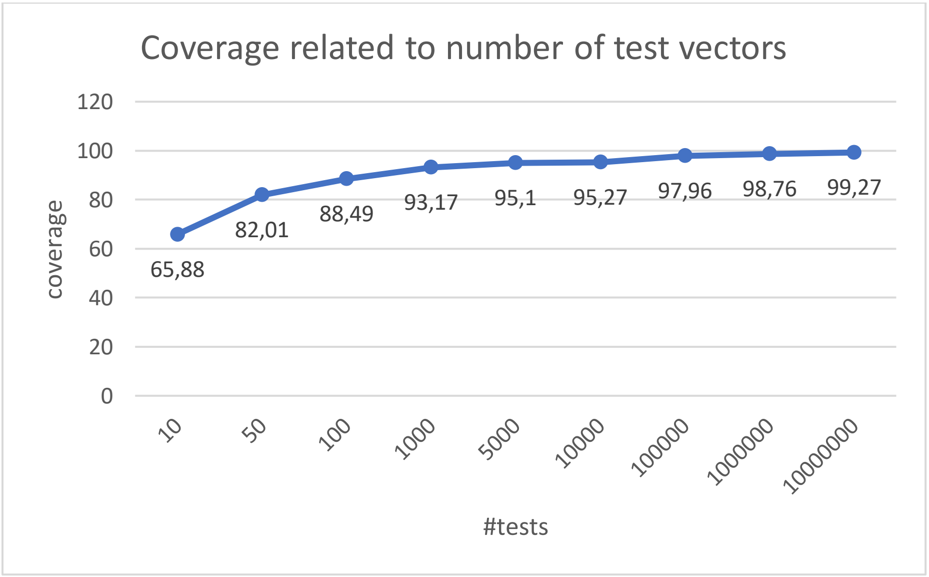
3.

By removing the faults resulting from the modules AOI32D1 and simulating for the same vectors, we achieve a coverage of 99.22%. Since the faults we removed do not belong to the undetected, the function that gives us the coverage is: $coverage = \frac{total - 11}{total}$. This function is incremental and we decreased $total$, so the coverage decreased as well.

4.

By applying simulation for the faults found in the c1355_faults.dat file, we achieve a coverage of 99.72%. This is because we now only have 3 undetected faults.

5.



As can be seen from the graph, with a few initial vectors the coverage increases considerably, while as we grow their number this increase becomes slower and does not reach 100%. This is because some faults are detected by specific vectors that have a low probability of occurring. Because of this, it is difficult to find them in a pseudo-random way even if we apply 10,000,000 vectors. As mentioned in the question, the fact that the file indicates 10,000,000 vectors does not mean that they are different. This can be seen for example in the c7552_test_set_5000.v file which has 181 different vectors, as well as in the c7552_test_set_10000.v file which we would expect to have twice as many but only has 175.

The faults that are not detected are 50 and belong to the category "Hard to Detect Faults" or "Random Pattern Resistant".

Question 2

1.

Performing the test vector extraction process, we see that in the DRC stage we have 74 violations "unstable nonscan DFF when clocks off", i.e. as many as the flip-flops of the circuit. This means that the tool sees that our circuit is sequential but cannot apply the optimal algorithms for sequential circuit ATPGs since it does not have complete control over flip-flops at all times. These violations predispose us to the fact that the results of ATPG will not be very good.

This is subsequently confirmed, as the ATPG process finally achieved a coverage of 0.29%. We also see that most of the faults that were not detected are "ATPG untestable", i.e. the algorithm failed to extract a test vector for them. This is because there is no control over the sequential elements, to the extent that the algorithm can use them to check or propagate an error.

2.

The results are shown in the table below:

	method	#tests	coverage(%)	#not_detected	type
	deterministic	176	100	1	Undetectable
Pseudo-random	10.000	224	99,95	2	Not detected
	100.000	202	99,98	1	Not detected
	1.000.000	209	99,98	1	Not detected

As can be seen, the deterministic method has the best results in terms of test set size, coverage and categorization of faults. The methods of lines 3 and 4 achieved essentially the same results as deterministic, simply because they do not have the knowledge that the one error they did not find is undetectable, they categorize it as not detected and thus reduce coverage. The line 2 method happened not to detect another error.

In this circuit there do not seem to be any "Random Pattern Resistant" faults since by increasing the length of the sequence we see that everything is detected except for one error which is undetectable.

3.

By limiting the desired test coverage to 97.5%, we observe that the ATPG process stops as soon as it exceeds it (it reached up to 98.77%) and does not continue for those faults that have not been found. In fact, the tool issues a warning, notifying that the process has stopped after the percentage we set has been reached.

By limiting the maximum number of test vectors to 200, we observe that the ATPG process stops as soon as it reaches them and does not continue for those faults that have not been found. With this limitation we achieve a coverage rate of 99.53%.

By limiting the maximum number of backtracks of the ATPG algorithm to 2, we observe that the process stops having achieved a 100% coverage rate. In other words, this restriction, for this circuit, did not affect the results. The only difference with the default options (10 backtracks) is that with them the test set consists of 207 vectors, while with the constraint it consists of 218. This is expected since a higher abort limit implies a greater effort in the extraction of vectors and since we have a greater effort, these vectors have better compatibility with the existing ones so they can be more compressed.

4.

By performing the procedure described in the question, we see that with the given test set we achieve a coverage of 93.17% in contrast to the one (93.64%) shown in its file. Also, during fault simulation we have 6852 faults, while the test set file has 6888. Bypassing these incompatibilities, we have 468 not_detected faults from fault simulation which we write to a file and then, for them alone, we perform a deterministic ATPG process achieving 100% coverage. Finally, we write in a file the resulting test set.

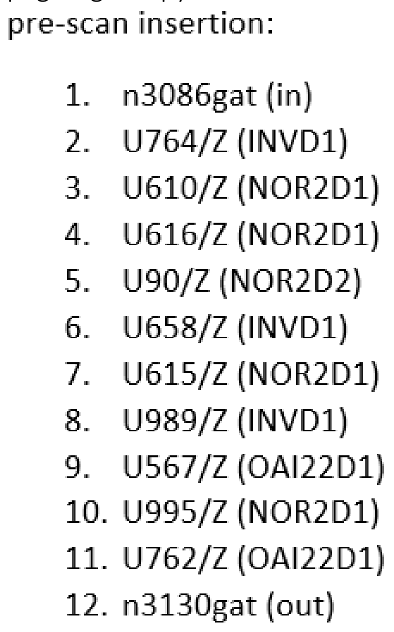
Question 3

1.

The results before and after scan insertion are shown in the following table.

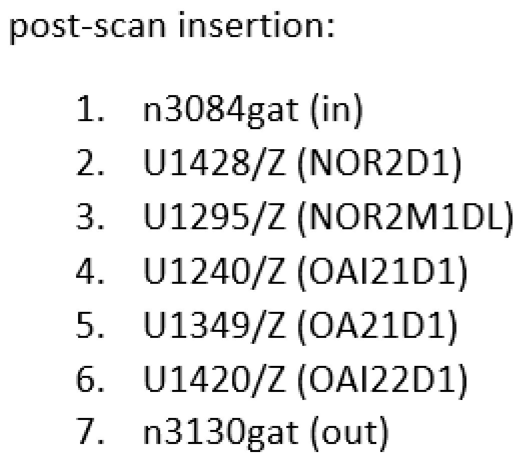
	pre-scan insertion	post-scan insertion
area	23868	27320
delay	6	6

The critical paths for each case are shown below.



pre-scan insertion:

1. n3086gat (in)
2. U764/Z (INVD1)
3. U610/Z (NOR2D1)
4. U616/Z (NOR2D1)
5. U90/Z (NOR2D2)
6. U658/Z (INVD1)
7. U615/Z (NOR2D1)
8. U989/Z (INVD1)
9. U567/Z (OAI22D1)
10. U995/Z (NOR2D1)
11. U762/Z (OAI22D1)
12. n3130gat (out)



post-scan insertion:

1. n3084gat (in)
2. U1428/Z (NOR2D1)
3. U1295/Z (NOR2M1DL)
4. U1240/Z (OAI21D1)
5. U1349/Z (OA21D1)
6. U1420/Z (OAI22D1)
7. n3130gat (out)

We observe that after applying the process, the area has increased by about 14%. There doesn't seem to be a delay overhead here, which doesn't make sense. The overhead we observe is due to the insertion of multiplexes at the input of each flip-flop of the circuit consisting of additional material, while at the same time it takes time away from the combination circuit.

2.

The testbench applies control to the test set that we export through the ATPG process of TetraMax. The results are stored in a stil file and then, with a python script(stil_preprocess.py) we read it and bring it to the format: <capture_pulse> <test_si> <test_se> <pi> <scan_in> <expected_po> < expected_so>. The formatted file is my_s5378.vec.

The testbench(s5378_tb) follows the steps:

1. We define and open the input(my_s5378.vec) and output(so.log, po.log) files.
2. We read the test set from the input file and for each pattern we do the following:
    a. We set the primary inputs of the circuit.
    b. We activate the SE signal and shift-in the chain values and at the same time shift-out the values of the previous pattern.
    c. After the primary outputs have stabilized after a period of time, we record them.
    d. We apply values to the SI and SE marks based on the stil file.
    e. Based on the CK signal of the stil file, we apply a clock pulse or not.
3. Once we have applied all the patterns and recorded all the outputs, we write them to the po.log and so.log files.
4. For each pattern we print the recorded and expected outputs and display an informative message for their correctness. We count the incorrect results and display the success rate of the test set.
5. We close the files and terminate.

Question 4

1.

For the synthesis, the run.tcl script of "example" was executed with the difference that the clock period is 5.5 instead of 4, so that there is no timing violation.

2.

By applying the ATPG process to the circuit, the advantages of scan insertion are obvious. The coverage we achieve is 100% as opposed to 0.27% in exercise 2. This is because we can now check the internal state of the circuit before applying any vector to its primary inputs. This feature allows the tool to bring the circuit to a known state without having to apply vectors to the inputs in any specific order.

3.

By creating more than one chain of sequential elements, we achieve a shorter time to enter a state before applying a test vector. This is because the length of the chains decreases and the parallelism increases, so the new state shift cycles in the sequential elements are reduced. This is confirmed by the s1423_1chain_myvectors.v and s1423_2chains_myvectors.v files. In the first case we need 9957 test cycles while in the second 5032.