ECE 538: VLSI System Testing
Spring 2019

Krish Chakrabarty

Homework 4

Assigned: March 30, 2019

Due: April 19, 2019

Instructions:
You are required to work on the homework on your own. If you think a question has several interpretations, make reasonable assumptions and state them clearly. Please be neat and legible. Show all work in order to receive partial credit. Slide your homework under Prof. Chakrabarty's office door (128 Hudson Hall), leave it in his mailbox in Room 130 Hudson Hall, or submit by email.

Problem 1 (30 points) Path Delay and Small Delay Defect Testing using Synopsys TetraMax:
You have to use Synopsys TetraMax and generate test patterns for path delay and small delay defects for the scan inserted benchmark circuit *s5378_bench.*v.
Please go through the Synopsys TetraMax tutorial posted on the website. All required files can be downloaded from the directory mentioned in the link inside the tutorial:

a. (10 Points) Path Delay Faults: The Synopsys TetraMax tutorial illustrates test pattern generation for path delay faults for the 200 most-critical paths.

(i) Repeat the example using 50, 100, 150, 250, 300 most-critical paths. Present your results in the form Table 1 as shown in this document. List the fault coverage obtained, CPU time needed, and the number of patterns generated etc., as shown in Table 1.
(ii) The example script uses clock period of 0.15 ns. Please change the clock period to 0.10 ns and repeat Part (i). Did you observe any differences in the fault coverage obtained, CPU time needed, and the number of patterns generated? Explain your findings.

b. (15 Points) Small Delay Defects (SDD): The Synopsys TetraMax tutorial illustrates an example of test pattern generation for small-delay defects (SDDs) with a clock period of 1.1 ns and shows delay-effectiveness results when the slack is set to 10% of the clock period.

(i) Repeat the example using slack values of 15%, 20%, 25% and 30% of the clock period. List the delay effectiveness, SDQL value, CPU time needed, and the number of patterns generated etc., as shown in Table 2.
(ii) The example script for SDD detection is based on a clock period of 1.1 ns. Next change the clock period to 1.2 ns and repeat the Part (i) for slack values of 10%, 15%, 20%,

25% and 30% of the clock period. List the delay effectiveness, SDQL value, the CPU time needed, and the number of patterns generated, as shown in Table 2.
What differences did you observe compared to the results obtained in Part (i)?

(iii) Next change the clock period to 1 ns, repeat Part (i) for slack values of 10%, 15%, 20%, 25% and 30% of the clock period. What differences did you observe compared to the results obtained in Part (i) and Part (ii)?

Table 1. Sample results for path-delay faults
(Filled randomly to provide an example of how the results need to be reported in the homework)

Number of Critical Paths	1000	2000	3000	4000	5000
Total Faults	1000	2000	3000	3846	3846
Detected	750	1260	1725	1883	1883
Test Coverage	75.00%	63.00%	57.50%	48.96%	48.96%
Patterns	24	24	27	25	26
CPU Time	1.1 ns	1.4 ns	1.5 ns	1.6 ns	1.8 ns

Table 2. Sample results for small-delay defects (Clock period of 2.5 ns)
(Filled randomly to provide an example of how the results need to be reported in the homework)

Slack	10%	15%	20%	25%	30%
Total Faults	111802	111802	111802	111802	111802
Detected	111483	111483	111482	111483	111483
Delay Effectiveness	76.58%	84.50%	82.99%	83.24%	86.21%
SDQL	3791.45	3623.79	3554.68	3353.24	3442.69
Patterns	544	566	598	620	668
CPU Time	1.1 ns	1.4 ns	1.5 ns	1.6 ns	1.8 ns

Problem 2 (5 + 5 + 5 = 15 points) [Response compaction using LFSRs]: An LFSR with feedback polynomial $x^4 + x^3 + 1$ is used to compact the test response R = 1110101, i.e. $R(x) = x^6 + x^4 + x^2 + x + 1$.

(a) Draw the structure of the LFSR (you are free to choose either a Type-1 or a Type-2 LFSR).
(b) Use polynomial division to compute the value of the signature obtained.
(c) Verify your answer in (b) by simulating the LFSR.

Problem 3 [10 points] [SOC test infrastructure design]:
Consider an embedded core, referred to as C in a core-based SOC. C has 8 functional inputs a[0:7], 11 functional outputs z[0:10], 9 internal scan chains of lengths 12, 12, 8, 8, 8, 6, 6, 6, and 6 flip-flops, respectively, and a scan enable control sc. The test wrapper for C is to be connected to a 4-bit TAM. Design the wrapper for this core, and present your results in the form of the following table:

	Wrapper scan chain 1	Wrapper scan chain 2	Wrapper scan chain 3	Wrapper scan chain 4
Internal scan chains	Which scan chains are included? Provide number of scan elements.	Which scan chains are included? Provide number of scan elements.	Which scan chains are included? Provide number of scan elements.	Which scan chains are included? Provide number of scan elements.
Wrapper input cells	How many wrapper input cells are included?	How many wrapper input cells are included?	How many wrapper input cells are included?	How many wrapper input cells are included?
Wrapper output cells	How many wrapper output cells are included?	How many wrapper output cells are included?	How many wrapper output cells are included?	How many wrapper output cells are included?
Scan-in, scan-out length	Number of bits?	Number of bits?	Number of bits?	Number of bits?

Note: It is advisable to write a computer program (or a set of programs) to solve this problem.

Problem 4 (8+7=15 points) [SOC testing]

(a) Consider an embedded core in an SOC that must be optimally wrapped to minimize its testing time. The core has 4 internal scan chains (the scan chains are fixed, i.e., they cannot be redesigned) of length 12 bits, 12 bits, 8 bits, and 8 bits, respectively, 16 functional inputs and 8 functional outputs. Let the TAM width for this core be w. Design a wrapper for the core for $w = 2, 3, 4, 5, 6$.

(b) Next, consider an SOC with 8 identical copies of the core from (a). Design an efficient TAM for this SOC, assuming that the SOC-level TAM width is 36 bits.

Note: It is advisable to write a computer program (or a set of programs) to solve this problem.

Problem 5 (7+8=15 points) [Test Compression]:

Consider a core under test with 16 (balanced) scan chains, each scan chain of length 8 bits, and 100 test patterns. For the sake of this problem, create a set of pseudorandom patterns using a (software) random number generator, e.g., the rand() function in Unix. Use the following parameters:

- Probability of 1s: 0.02
- Probability of 0s: 0.03
- Probability of don't-cares: 0.95

(a) Design a fanout-based decompressor. You are free to use inverters if you see "inverse compatibility", i.e., one column is the complement of another.

(b) Design a decompressor based on reconfigurable broadcast scan. Use two configurations.

Note: It is advisable to write a computer program (or a set of programs) to solve this problem.

Problem 6 (20 points) [Modern Microprocessors: Testing and Design-for-Testability]:

For this problem, you are asked to write an essay on the testability features, test methods, and DFT strategies used for current-generation microprocessors. You need to do some research on your own, and collect information from the web, conference proceedings, and

journal articles. Your essay should be coherent, carefully crafted, and it should throw light on the comparative DFT features of these processors. You are limited to three pages of text, two tables, and two figures. Focus on microprocessors such as AMD Athlon, Sun's Niagara 2 CMP/CMT SPARC and Dual-Core UltraSPARC, IBM's Dual-Core Power5, and Intel's Itanium 2. In addition, feel free to write about any other modern microprocessor that excites you.