UtahStateUniversity

Department of Electrical and Computer Engineering
Utah State University

ECE 5480/6480: VLSI Testing and Verification
Instructor: Dr. Roy

Lab 04
Assigned Date: 02/15/2018
Due Date: 02/22/2018

In this lab, the goal is applying some set of test vectors and then getting the information of faults and test coverage for each of them.

Before starting the lab, you need to be familiarized with some related concepts and commands for this lab.

TetraMAX has three modes for generating patterns that are according to the following:

Basic Scan Only

1) Very fast

2) Excellent coverage for Full-Scan search

3) Combinational-Only

Important: Basic-Scan can be similar to D-Algorithm.

Command:

```
run_atpg basic_scan_only
```

Fast Sequential

1) Higher fault coverage (Especially for near-full scan designs).

2) Good for shadow logic around memories.

3) Limited non-scan (flip-flops, latches, bus keeper, RAMs)

4) Clocks controlled by PIs

5) Work on Primary Inputs (PIs)

Important: Fast Sequential can be similar to Sequential PODEM (S - PODEM) Algorithm.

Sequential PODEM Algorithm = It is the developed version of PODEM Algorithm that can be applied to both combinational and sequential circuits for detecting their faults. Each node can take on one of 11 different values in this algorithm. Sometimes called the "11 Values". They are as follows:

$$D,\ \overline{D},\ 0,\ 1,\ X,\ P,\ P0,\ P1,\ 1P,\ 0P, \text{ and } PP$$

Note: Fast Sequential can help to detect AU (ATPG Untestable) remaining after Basic-Scan ATPG.

Command:

```
run_atpg fast_sequential_only
```

Full Sequential

1) Similar to Fast-Sequential

2) Powerful engine supporting more complex designs

Command:

```
run_atpg full_sequential_only
```

Important: You can combine these modes to get new mode (algorithm) for generating patterns.

Comparison of D-Algorithm and PODEM Algorithm

D-Algorithm

1) Emphasis on D-propagation.

2) Full-Scan search

3) May be inefficient due to constant backtracking.

Reason: The D algorithm includes a condition which often results in undoing a previous decision and making a different choice. In the worst case, all possible choices may have to be tried.

4) Some faults require multiple path sensitizations.

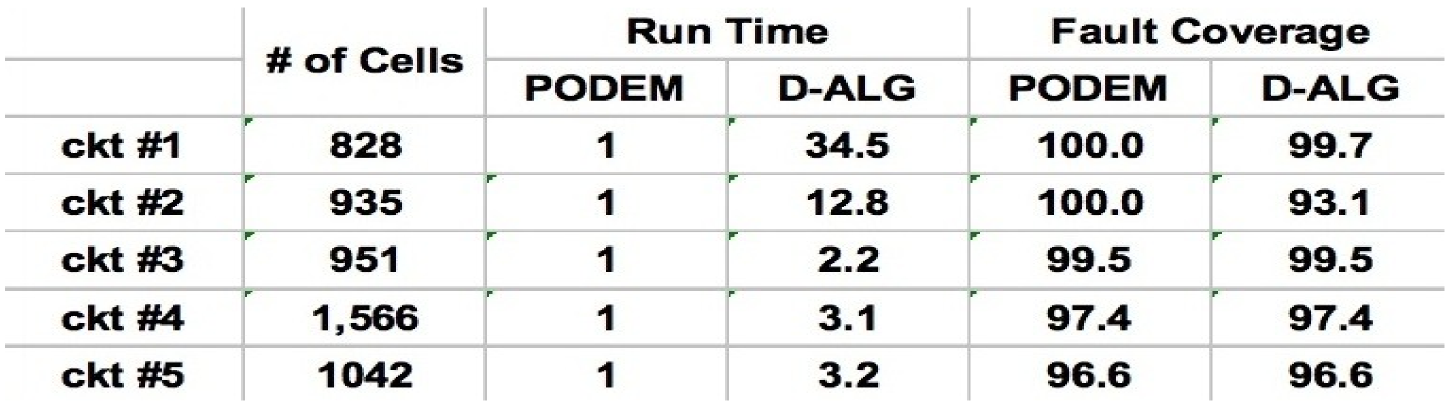
<u>PODEM Algorithm</u>

1) Work on Primary Inputs (PIs).

2) Search space is smaller than that of D-Algorithm.

3) Exponential complexity.

4) Faster than D-Algorithm.



	# of Cells	Run Time		Fault Coverage	
		PODEM	D-ALG	PODEM	D-ALG
ckt #1	828	1	34.5	100.0	99.7
ckt #2	935	1	12.8	100.0	93.1
ckt #3	951	1	2.2	99.5	99.5
ckt #4	1,566	1	3.1	97.4	97.4
ckt #5	1042	1	3.2	96.6	96.6

Comparison of Run-Time and Fault Coverage between D-Algorithm and PODEM Algorithm

Working with Fault Lists

TetraMAX maintains a list of potential faults for a design, along with the categorization of each fault. A list of faults is stored in a fault list file (An ASCII file which can be read and written using the ***read_faults*** and ***write_faults*** commands).

A fault list file contains one fault entry per line. Each entry consists of three items separated by one or more spaces. The first item indicates the stuck-at value (***sa0*** or ***sa1***), the second item is the two-character fault class code, and the third item is the pin path name to the fault site. Any additional text on the line is treated as a comment.

If the fault list contains equivalent faults, then the equivalent faults must immediately follow the primary fault on subsequent lines. Instead of a class code, an equivalent fault is indicated by a fault class code of "`--`".

NOTE: TetraMAX ignores blank lines and lines that start with a double slash and a space (`//`).

NOTE: You can control whether the fault list contains equivalent faults or primary faults by using the ***-report*** option of the ***set_faults*** command or the ***-collapsed*** or ***-uncollapsed*** option of the ***write_faults*** command.

Important Commands:

Here are some commands that you will find useful while implementing this lab.

1) Writing all faults (overwrites file filename):

```
TEST-T> write_faults filename -all -replace
```

2) Using an existing fault list from another tool or from a previous run:

```
TEST-T> read_faults filename
```

3) Reporting the collapsed fault list:

```
TEST-T> set_faults -report collapsed

TEST-T> report_faults -summary
```

4) Reporting uncollapsed fault list:

```
TEST-T> set_faults -report uncollapsed

TEST-T> report_faults -summary
```

5) Reporting detailed information about fault list:

```
TEST-T> set_faults -summary verbose

TEST-T> report_faults -summary
```

6) Reporting fault coverage:

```
TEST-T> set_faults -fault_coverage

TEST-T> report_faults -summary
```

7) Limit Generating Patterns:

```
set_atpg -patterns N     // (default: 0 - to disable limit)
```

ATPG process stops when the N patterns are generated or fault is detected.

8) Generating random test vectors:

```
run_atpg -random
```

Important Equations:

Test coverage = Detected Faults / Detectable Faults

Fault coverage = Detected faults / All Faults

ATPG effectiveness = ATPG - Resolvable Faults / All Faults

Fault Class Hierarchy

1) DT - Detected

- DS = Detected by Simulation
- DI = Detected by Implication
- DR = Robustly Detected Delay Fault

2) PT - Possibly Detected

- AP = ATPG Untestable (Possibly Detected)
- NP = Not Analyzed (Possibly Detected)

3) UD – Undetectable

- UU = Undetectable Unused
- UT = Undetectable Tied
- UB = Undetectable Blocked
- UR = Undetectable Redundant

4) AU - ATPG Untestable

- AN = ATPG Untestable (Not Detected)

5) ND - Not Detected

- NC = Not Controlled
- NO = Not Observed

6) -- - Equivalent to fault listed above it.

LAB PROCEDURE:

1) Apply the given 10 test vectors (included in test_vectors.stil) to the given circuit (circuit-3.v). What is the test coverage and the fault coverage?

2) Use Fast Sequential Only mode to generate a set of test_vectors to detect all faults. How many tests are generated and how many faults are found to be "Untestable"?

3) Generate random 10 test vectors using Full Sequential Only mode and apply it to the circuit. What is the test coverage and fault coverage for these 10 random vectors?

4) Write a small C/C++ program that generates 10 random test vectors. Then, apply the test vectors to the circuit. What is the test coverage and fault coverage?

DELIVERABLES:

Please submit the following via canvas:

1) A table which contains the test coverage and the fault coverage for all four tasks.

2) A text file which includes the output from tetramax showing the test coverage and fault coverage for all four tasks.

3) 3 .stil files which include the patterns used for steps 2, 3 and 4 in the LAB PROCEDURE.

4) The C/C++ program which generates 10 random vectors.

Good Luck