3/4 November 2021

TESTING AND FAULT TOLERANCE

Laboratory Session 5: "TDF & Test Compression"

Student Name	Student Surname	Student ID

Riccardo Cantoro (riccardo.cantoro@polito.it)		Nick Deligiannis (nikolaos.deligiannis@polito.it)

	La Greca	S281589

SAVE THIS FILE IN PDF AND SUBMIT IT

TDF & Test Compression

[A] Transition-Delay Fault Model

The transition-delay fault model is like the stuck-at fault model, except that it attempts to detect **slow-to-rise** and **slow-to-fall** nets instead of stuck-at-0 and stuck-at-1. A slow-to-rise fault as a defect translates to the transition from 0 to 1 for that specific net not being in-line with the functional/operating speed of the circuit but it is rather delayed. Similarly, a slow-to-fall fault as defect translates to the transition from 1 to 0 for that specific net not being in-line with the speed of the circuit.

We will see now how to adapt the basic ATPG flow from LAB2 and LAB4 to the transition-delay fault model.

In TestMAX the set_delay command is used to specify options for the transition delay fault model and the respective test generation and fault simulation. TestMAX supports several ATPG modes for applying transition-delay tests. You select the required mode with the following command while in DRC mode:

```
DRC-T> set_delay -launch_cycle OPTION
```

The following options are supported:

Launch-On Shift (LoS): Specified by the `last_shift` option, TetraMAX ATPG launches a test pattern **in the last scan load cycle when the scan enable is active** (when in scan-shift mode). It exercises target transition faults and then **captures the**

new logic values in a system clock cycle when the scan enable is inactive (when in capture mode).

```
DRC-T> set_delay -launch_cycle last_shift
```

Launch-On Capture (LoC) / System Clock: Specified by the `system_clock` option (which is also the default ATPG mode for transition-delay faults), TestMAX ATPG launches a test pattern using a normal system clock. It exercises the target transition faults and then captures the new logic values with a **subsequent** system clock. If you use the `system_clock` parameter, you must enable Fast-Sequential patterns with the `-capture_cycle` option of the `set_atpg` command.

```
DRC-T> set_delay -launch_cycle system_clock
```

The following Figures showcase how the clock and the scan enable signals behave during the aforementioned configurations.

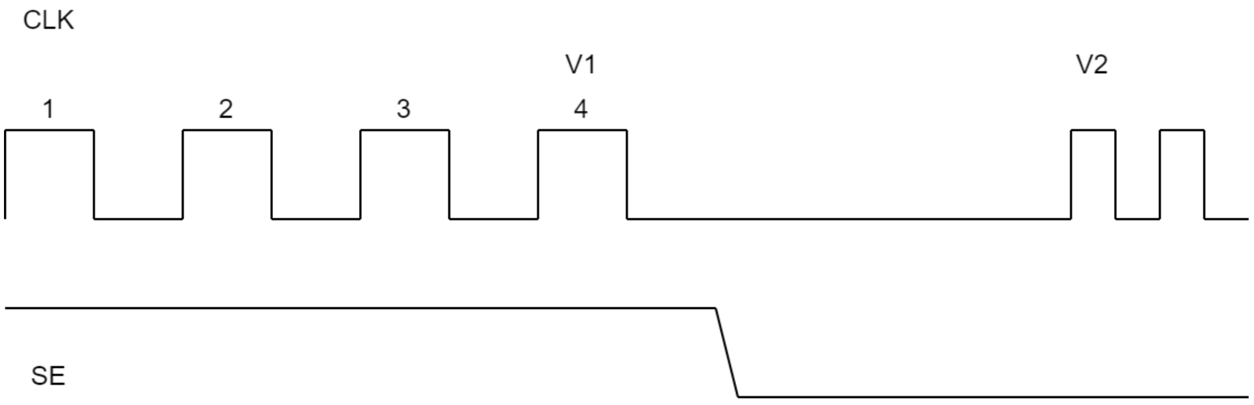


Figure 1: Launch-On Capture

In LOC the last shift of the scan chain also initializes the inputs of the respective combinational block and the first functional clock is launching a transition within that block (V1 vector). Note that the scan enable signal is de-activated at this point. Then, the second functional clock captures the propagated transition at the outputs. Thus, assuming a chain of length N, the V1 vector is loaded by N slow clock cycles. Then, two fast clock cycles are launching and capturing the transition within and from the combinational block.

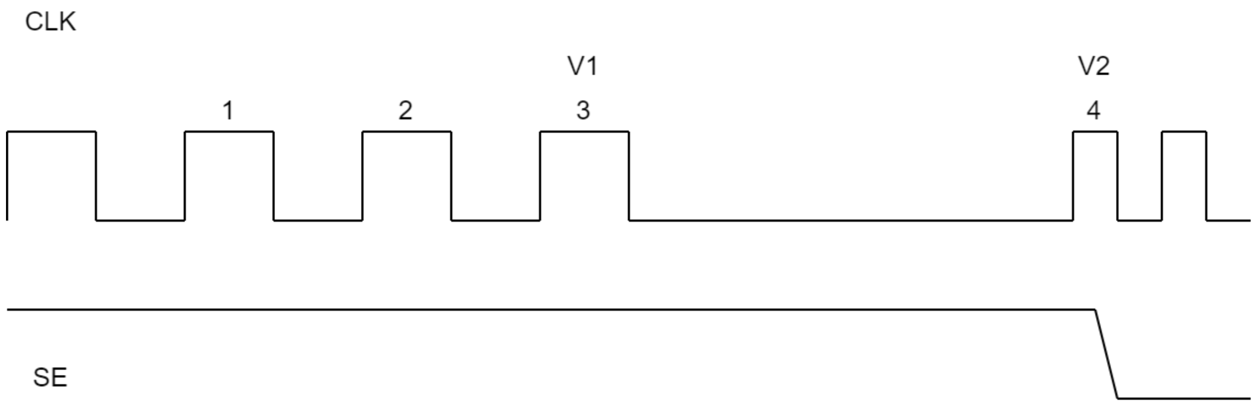


Figure 2: Launch-On Shift

In LOS the last shift clock is used to launch a transition in the combinational block. During launching, the scan enable signal remains active. So, assuming a scan chain of length N, the first N-1 bits of the V1 vector are shifted-in by a slow clock and initialize the logic values in the inputs of the combinational block. Then, the N-th shift with the fast clock launches the transition while the second fast clock captures it. The scan enable signal is de-activated after the **first** functional clock pulse.

The `-pi_changes` or `-nopi_changes` options of the `set_delay` command specify whether patterns are allowed to change values on primary inputs (PIs) between the launch clock and the capture clock. This will have no effect on LOS transition fault patterns. Example:

```
DRC-T> set_delay -launch_cycle system_clock -nopi_changes
```

The `set_atpg` command sets the parameters that control the ATPG process.

Launch-On Shift (LOS): It is based strictly on the **Basic-Scan ATPG** engine. Therefore, when you use `run_atpg` in the `last_shift` mode, `TestMAX` uses Basic-Scan ATPG only and generates Basic-Scan Patterns.

Launch-On Capture (LOC)/System Clock: It is based on **Fast-Sequential ATPG** engine. If Fast-Sequential ATPG is not enabled when you use `run_atpg` in the `system_clock` mode, `TestMAX` reports and error message (M236).

You can enable Fast-Sequential `ATPG` by using the following command:

```
TEST-T> set_atpg -capture_cycles d
```

You must set the effort level `d` to at least 2 for the `system_clock` mode. If you try to set it to 1, the `set_atpg` command returns an invalid argument error. For full-scan designs, you can set the effort level d to 2. For partial-scan designs, a number greater than 2 might be necessary to obtain satisfactory transition-delay fault coverage.

Lastly, you have to select the transition-delay fault model as:

```
TEST-T> set_faults -model transition
```

#LOS summary	**#LOC summary**
… set_delay -launch_cycle last_shift … set_faults -model transition …	… set_delay -launch_cycle system_clock set_delay -nopi_changes set_atpg -capture_cycles 2 #(or higher) … set_faults -model transition …

<u>[A.1] Tasks:</u>

1. Complete the following table after running the transition-delay ATPGs for b06.

Delay mode	Capture cycles	PI changes	Test coverage	Patterns	TAT [ccs]
LOS	-	-	92.93%	23	287
LOC	2	no	66.03%	13	179
LOC	2	yes	93.37%	28	347

[B] Test Compression

We have seen that scan-chains can be proven really beneficial for achieving significantly high test coverage percentages, especially when the CUT is a sequential circuit. However, as the chips get bigger and bigger, the ratio of logic to be tested per pin of the integrated circuit increases dramatically. The large volume of scan test data can cause a significant increase in the test time and the required tester memory.

The test compression technique was developed to alleviate this issue. When the ATPG engine generates a test for a single fault or a set of faults, only a small amount of the scan cells must take specific values to exercise this specific fault (or group of faults respectively). The rest of the scan cells are filled with don’t care values (typically random). Thus, loading and unloading these vectors is not very efficient in terms of tester time and the tester time is something very important and limited as a resource. The test compression takes advantage of this small number of significant values per targeted fault to reduce the test data and the test time.

The general idea is to appropriately modify the design to increase the total number of internal scan chains, with each of them being shorter in length than in the original scan circuit. These internal chains are now driven by a **decompressor** circuit that is inserted to the original design, which allows for continuous flow to feed the internal scan chains of the circuit.

With the now increased number of test chains, not all the outputs can be sent to the output pins of the circuit. Thus, a **compressor** is also added to the design between the internal scan chain outputs and the “tester” scan chain outputs and is synchronized with the decompressor circuit.

In LAB3 we have seen how we can insert scan chains in a design by using the design compiler. Specifically, we used the command:

```
set_scan_configuration -chain_count X
```

To create X scan chains within the targeted circuit. To further create more internal (compressed) scan chains for the purposes of the test compaction, we will use the following command:

```
set_scan_compression_configuration -chain_count Y
```

Where Y is the number of internal scan chains we wish to insert to the circuit. Figures 3 and 4 depict an example of such configuration.

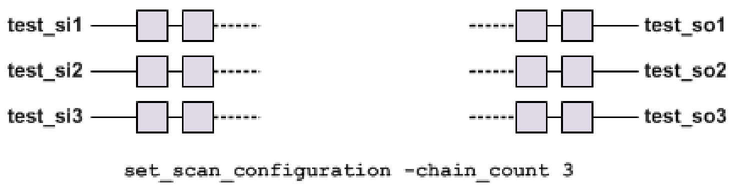


Figure 3: Standard scan mode

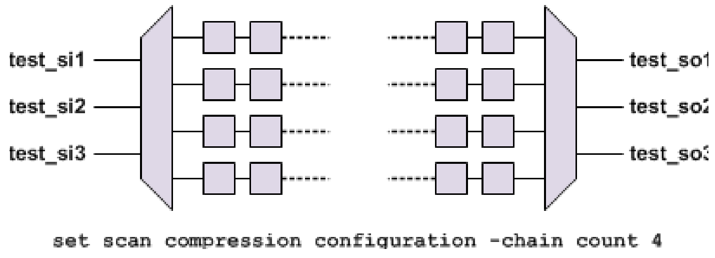


Figure 4: Compressed scan mode

You can use the provided .tcl script to modify the b12 circuit and equip it with scan chains as:

```
dc_shell-xg-t -f b12_compressor.tcl
```

Furthermore, you can modify the number of scan chains and internal scan chains added to the circuit by modifying the values at lines 13 and 14 respectively of the .tcl script.

After the successful completion of the scan chains insertion you can invoke the GUI with:

```
dc_shell> start_gui
```

And you can navigate the enhanced design and inspect the decompressor/compressor modules schematics like in the image below. Then starting from the decompressor, you can navigate towards the POs of the circuit by following (double-clicking) at the outputs of the decompressor. You will eventually reach the scan cells of the design. If you start from the compressor module, then you can navigate towards the PIs of the design by following (double-clicking) at the inputs of the compressor.

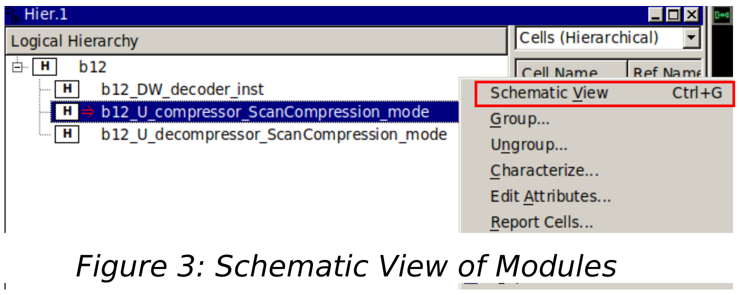


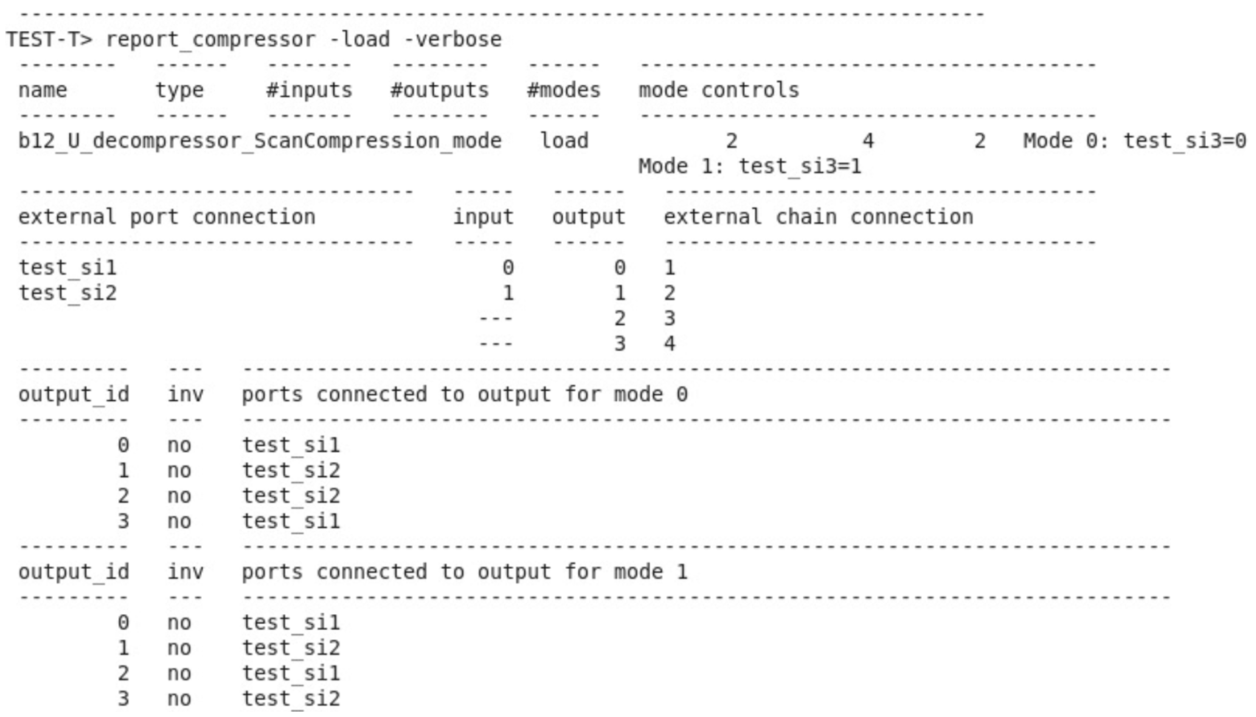
Figure 3: Schematic View of Modules

You can see that at the end of the .tcl script we use for the design compiler we write two different test protocols (.spf files). One to be used for normal scan, and the second to be used for test compression. You can now invoke TestMAX and read the b12 scan circuit. Then, while in DRC we can select the appropriate test protocol to either resort to basic scan or test compression.

If you specify the compressor test protocol during DRC within TestMAX, you can use the `report_compressors` command as:

```
report_compressors [-all|-load|-unload] -verbose
  -load: reports information for the decompressor
  -unload: reports information about the compressor
  -all: reports all information
```

For example, for the `-load -verbose`, we have the following output:



```
 --------------------------------------------------------------------------
TEST-T> report_compressor -load -verbose
 --------   ------   -------   --------   ------   -----------------------------------
 name       type     #inputs   #outputs   #modes   mode controls
 --------   ------   -------   --------   ------   -----------------------------------
 b12_U_decompressor_ScanCompression_mode   load          2         4       2   Mode 0: test_si3=0
                                                   Mode 1: test_si3=1
 ------------------------------   -----   ------   ---------------------------------
 external port connection          input   output   external chain connection
 ------------------------------   -----   ------   ---------------------------------
 test_si1                              0        0   1
 test_si2                              1        1   2
                                     ---        2   3
                                     ---        3   4
 ---------   ---   ---------------------------------------------------------------------
 output_id   inv   ports connected to output for mode 0
 ---------   ---   ---------------------------------------------------------------------
         0   no    test_si1
         1   no    test_si2
         2   no    test_si2
         3   no    test_si1
 ---------   ---   ---------------------------------------------------------------------
 output_id   inv   ports connected to output for mode 1
 ---------   ---   ---------------------------------------------------------------------
         0   no    test_si1
         1   no    test_si2
         2   no    test_si1
         3   no    test_si2
```

The circuit has 3 basic scan chains and 4 internal scan chains. Here we can see that the two of the external scan chains drive the 4 internal scan chains according with two possible configurations according to the value of `test_si3` (mode 0 and mode 1).

The next step is to compare with the previous findings of our ATPG attempts with basic scan. Specifically we are going to determine which is the difference between the test compression approach and the basic scan, in terms of achieved fault coverage.

<u>Tasks [B.1]:</u>

1. Prepare two ATPG .tcl scripts for the transition-delay fault model for the b12 benchmark and report your ATPG setup:
    a. One script for using the compressor
    b. One script for bypassing the compressor

Setup: LOC with pi changes & cc = 2

Outer scan chains	Inner scan chains	Longest s.c. #FFs	Faults	Use test compr.		Bypass test compr.	
				Test patterns	Test coverage	Test patterns	Test coverage
1	2	61	5990	245	98.05 %	231	97.76 %
	5	25	6090	270	92.54 %	232	96.40 %
	30	5	6934	99	63.19 %	237	87.61 %
4	10	13	6560	248	96.58 %	239	91.05 %
	30	5	7142	282	88.94 %	236	85.94 %

2. Complete the table below after re-synthesizing the b12 benchmark each time with the requested number of scan chains.
3. Use the same setup and run the ATPG on the b12 circuit generated in Task B.2 of Lab4 (with 1 and 4 scan chains). Compare the results with the ones on the previous table. Which are the main differences? Try to motivate the answer.

We can notice that the number of patterns used is higher with a lower test coverage. This is because of the additional compressor logic that is added but not tested, introducing more faults.

Appendix A: Files of LAB5

- **All files listed here are included in your remote /home directory under lab5 folder.**

Filename	**Description**
`b06_scan.v`	ITC’99 b06 netlist (full-scan) design
`b06_scan.spf`	ITC’99 b06 STIL procedure file
`b12.v`	ITC’99 b12 netlist (non-scan design)
`b12_compressor.tcl`	`Design Compiler` `.tcl` script for scan-chain and compressor insertion for the b12 circuit
`b12_scancompress.spf`	Scan Compress test protocol for `TestMAX`
`b12_scan.spf`	Scan test protocol for `TestMAX`
`pdt2002.db`	Technology library model database (for `Design Compiler`)
`pdt2002.dc_setup.tcl`	Technology library setup script (for `Design Compiler`)
`pdt2002.v`	Technology library model (for `TestMAX`)