TESTING AND FAULT TOLERANCE

Laboratory Session 4: “Full-Scan Designs”

|  |  |  |
| --- | --- | --- |
| **Student Name** | **Student Surname** | **Student ID** |
| Salvatore Gabriele | La Greca | S281589 |

**SAVE THIS FILE IN PDF AND SUBMIT IT ON “EXERCISE”**

A picture containing text

Description automatically generatedA picture containing blue, dark

Description automatically generatedA cartoon of a castle

Description automatically generated with medium confidence

**!!Prerequisites!!** TestMAX Flow (see [Appendix B](#_Appendix_B:_TestMAX))

# **[A] Scan-Based ATPG**

In this exercise, we present a basic scan based ATPG Design flow for Synopsys TestMAX. In **full-scan designs**, additional pins are used to put the circuit in test mode, where sequential ***cells*** are connected into scan-chains. For example, MUX-scan flip-flops contain 2 additional inputs (see Figure 1).

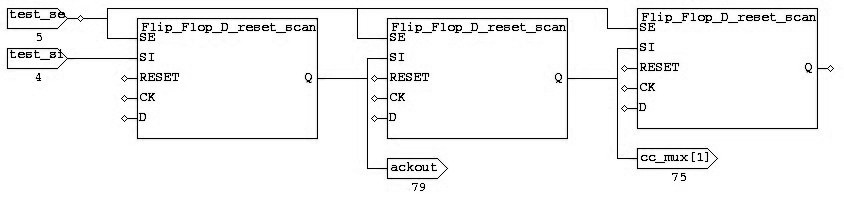


Figure 1 - Mux-Scan flip-flops connected into a scan chain

Scan cells are identified during the DRC process. Starting from the scan-chain output ports, a back-trace is performed through a sensitized path. Gates in the traced path are placed into scan cells. After completing the DRC process, you can report the list of scan-chains with the following command:

* TEST-T> report\_scan\_chains

The output should look like that:

chain group length input\_port output\_port

------- ----- ------ --------------- ----------------

c1 sg0 230 a\_scanin1 a\_scanout1

c2 sg0 225 a\_scanin2 a\_scanout2

c3 sg0 230 a\_scanin3 a\_scanout3

For each of the scan-chains, it is possible to report the list of all the scan cells of the path, like we do with the following command:

* TEST-T> report\_scan\_cells c1

Which yields an output that looks like this:

chain cell type inv gate# instance\_name (type)

------- ---- ------- --- ------ -------------------------------

c1 0 MASTER IN 147 reg4/r (N\_LATCH)

c1 1 MASTER IN 145 reg3/r (N\_LATCH)

DSLAVE IN 146 reg4/lat1 (P\_LATCH)

c1 2 MASTER IN 143 reg2/r (N\_LATCH)

SCANTLA IN 144 reg3/lat1 (P\_LATCH)

Position 0 indicates the scan cell closest to the scan-out pin, position 1 is adjacent to it, etc. Scan-chains are defined in the ScanStructures block of the STIL procedure file. You can verify how they are specified by looking at the file that we provide.

**[A.1] Tasks:**

1. Draw the scan-chain of the circuit b06 after performing the DRC (use the file b06\_scan.spf).
2. Report the name of each cell as well the scan-in and scan-out pins
3. How many non-scan D FFs are present?

1)

2) \uscite\_reg[2] (Flip\_Flop\_D\_reset\_scan)

\uscite\_reg[1] (Flip\_Flop\_D\_reset\_scan)

\state\_reg[2] (Flip\_Flop\_D\_reset\_scan)

\state\_reg[1] (Flip\_Flop\_D\_reset\_scan)

\state\_reg[0] (Flip\_Flop\_D\_reset\_scan)

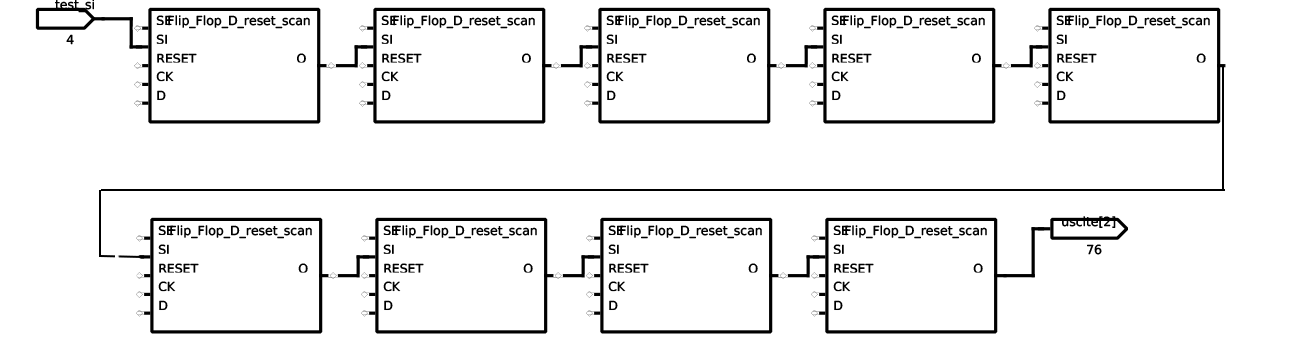
enable\_count\_reg (Flip\_Flop\_D\_reset\_scan)

\cc\_mux\_reg[2] (Flip\_Flop\_D\_reset\_scan)

\cc\_mux\_reg[1] (Flip\_Flop\_D\_reset\_scan)

ackout\_reg (Flip\_Flop\_D\_reset\_scan)

3) There are 0 non-scan DFFs



* Use the command report\_primitives –summary

In a full-scan design, TestMAX performs the ATPG process in Basic Scan Mode. The ATPG process of full-scan designs is dramatically faster compared to equivalent non-scan sequential circuits (as you will see when comparing with LAB2’s Full-Sequential ATPG attempts). In fact, TestMAX considers a full-scan design as a combinational circuit, where each scan cell is a secondary/pseudo-primary input/output, directly accessible.

**[A.2] Tasks:**

1. Run a stuck-at ATPG for the b06 circuit.

|  |  |
| --- | --- |
| **Total faults** | 432 |
| **Detected** | 432 |
| **Aborted** | 0 |
| **Untestable** | 0 |
| **Test coverage** | 100% |
| **Fault coverage** | 100% |
| **Patterns** | 17 |
| **Test application time (TAT) [clock cycles]** | 199 |
| **CPU Time** | 0.01 |

1. Complete the following table.

* Compute the TAT either manually or by inspecting the STIL file

The scan-patterns can be saved into a STIL file by using the following command:

* TEST-T> write\_patterns FILENAME.stil -internal -format stil

Alternatively, it is possible to print them in the TestMAX shell, like in the example below:

* TEST-T> report\_patterns -internal -chain 1 -all

Pattern 0 (basic\_scan-chain\_test)

Time 0: load 1 = 001100110

Time 1: force\_all\_pis = 100111

Time 2: measure\_all\_pos = 110000

Time 3: unload 1 = 001100110

Pattern 1 (basic\_scan)

Time 0: load 1 = 001101011

Time 1: force\_all\_pis = 100010

Time 2: measure\_all\_pos = 010011

Time 3: pulse clocks clock (1)

Time 4: unload 1 = 111101101

We have already seen how it is possible to mask some of the primary outputs with the add\_po\_masks command. In this way, the fault effect propagated to such, now masked, output is not considered as detected by the fault simulator. Furthermore, we have seen how the ATPG can be forced to produce specific values on some of the primary inputs via the add\_pi\_constraints.

The same masks and constraints can now be specified for pseudo-primary inputs (PPIs) and pseudo-primary outputs (PPOs) by using the add\_cell\_constraints command as:

add\_cell\_constraints

<0 | 1 | x | 0x | 1x | ox | xx>

<[ chain\_name | instance\_name>

[-position {< cell\_pos1 | sci> [ cell\_pos2 | sci]>} | -all>>]

[-index number]

* The choices 0, 1, or x indicate the constrained value that should be placed during the end of the scan chain loading. When a 0, 1 or x constraint is applied, the scan cell is always loaded with this value for every pattern. The PPIs are constrained using the aforementioned choices.
* A ox constraint indicates that the observed value is always masked or considered to be X. The PPO is masked using this choice.

**Constraint & Mask:**

* When a 0x constraint is specified, the scan cell is always loaded with a 0 and we will always observe X
* When a 1x constraint is specified, the scan cell is always loaded with a 1 and we will always observe X
* When a xx constraint is specified, the scan cell is always loaded with an X and we will always observe X.

Cell constraints are added during the DRC mode as well. For example, like in the following example:

* DRC-T> add\_cell\_constraints 1 MAIN/CPU/TP/FI/OFIFO/reg3
* DRC-T> add\_cell\_constraints ox MAIN/PER/PRT\_1/PORTIN/reg3
* DRC-T> add\_cell\_constraints xx c34 -position 6

It is possible to get the list of cell constraints with the following command (when in TEST mode)

* TEST-T> report\_cell\_constraints

**[A.3] Tasks:**

1. Perform an ATPG for the b06 circuit while applying each time the following PPI constraints and PPO masks:
   1. \state\_reg[0]
   2. \state\_reg[1]
   3. \state\_reg[2]
2. Complete the table below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PPI constraints** | **PPO Masks** | **Constraint value used** | **Test Patterns** | **Test coverage** | **ATPG Untestable faults** | **Aborted faults** |
| 0 | No | 0 | 5 | 50.47% | 213 | 0 |
| 1 | No | 1 | 5 | 53.19% | 198 | 0 |
| X | No | X | 4 | 25.69% | 321 | 0 |
| None | Yes | Ox | 16 | 82.41% | 76 | 0 |
| 0 | Yes | 0x | 5 | 44.21% | 241 | 0 |
| 1 | Yes | 1x | 5 | 43.75% | 243 | 0 |
| X | Yes | Xx | 4 | 25.69% | 321 | 0 |

* Use the appropriate value for the add\_cell\_constraints

The load/unload functions do shift operations on the scan-chains, while the force/measure functions act on the primary input/output ports of the CUT.

After reporting the faults detected by ATPG patterns, it is possible to analyze how the patterns are applied, both in text mode and by using the graphical interface (gui). You can select a fault that is marked as Detected by Simulation (DS) and analyze it with the analyze faults command, like in the example below:

* TEST-T> analyze\_faults -stuck 0 U50/I2 -verbose -display

------------------------------------------------------------------------------

Fault analysis performed for U50/I2 stuck at 0 (input 1 of OR gate 16).

Current fault classification = DS (detected\_by\_simulation).

------------------------------------------------------------------------------

Connection data: to=MASTER

Fault site control to 1 was successful (data placed in parallel pattern 0).

Observe\_pt=66(DFF) test generation was successful (data placed in parallel pattern 1).

Test pattern stimulus:

Load 1-2: 66=1

Load 1-3: 68=1

Load 1-4: 65=1

Force PI: 0(eql)=1

Force PI: 1(clock)=0

Force PI: 2(reset)=0

Force PI: 5(test\_se)=0

Pulse PI: 1(clock)=010

Test pattern detection path:

16(OR) 1/0

43(OR) 1/0

45(NAND) 0/1

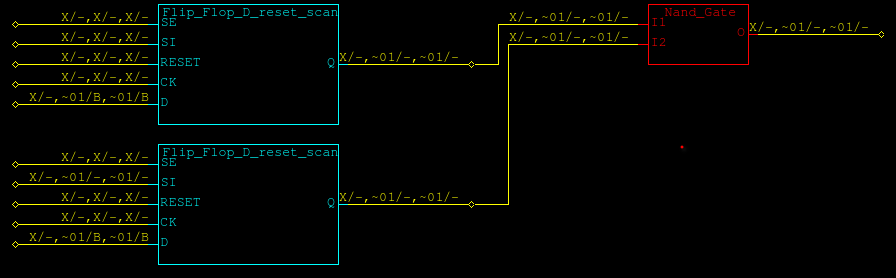
46(MUX) 0/1

66(DFF) 1/X

The gate\_report data is now set to "pattern:1".

In this example, TestMAX shows the values loaded in 3 scan cells i.e., cell#2 (id = 66), cell#3 (id = 68) and cell#4 (id = 65) of the scan chain as well as the values that were applied to the primary inputs of the circuit. You can notice that the Pulse PI procedure is applied to the clock signal in normal mode (since test\_se is previously set to 0), which propagates the effect of the fault up to an observable point (gate with id 66 i.e., scan cell #2). The scan cell captures the fault effect, which is the shifted-out and compared with the golden/expected value.

**[A.4] Tasks:**

1. Run an ATPG for the [last row of the previous table](#LAST_ROW).
2. Provide a screenshot of the analysis of the fault U51/O stuck-at 0.
3. Which cell prevents the fault from being tested?

2)

3) U51 prevents the fault from being tested

# **[B] Scan-chain Insertion**

Scan-chains are inserted in the design after the synthesis step. You can run insert the scan-chains in the circuit b10 using the Design Compiler .tcl script, as follows (in the bash shell):

* dc\_shell-xg-t –f b10\_scan\_insertion.tcl

The script produces the full-scan netlist (b10\_scan.v) and the STIL procedure file (b10\_scan.spf).

**[B.1] Tasks:**

1. For circuits b10 and b12:
   1. Perform the scan-chain insertion
   2. Perform the stuck-at ATPG
2. Complete the following table, while assuming that capture cycles are 2 and that Primary Inputs may change.
3. Modify the given .tcl script to change the number of scan-chains inserted by modifying the chain\_count parameter of the set\_scan\_configuration command.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Circuit** | **B10** | | | **B12** | | |
| **Scan-chains** | 1 | 2 | 4 | 1 | 2 | 4 |
| **FFs in longest**  **scan-chain** | 17 | 9 | 5 | 121 | 61 | 31 |
| **Test coverage** | 100% | 100% | 100% | 100% | 100% | 100% |
| **Patterns** | 48 | 49 | 48 | 107 | 106 | 110 |
| **TAT** | 932 | 551 | 344 | 13285 | 6742 | 3664 |

## **Appendix A: Files of LAB4**

* **All files listed here are included in your remote /home directory under lab4 folder.**

|  |  |
| --- | --- |
| **Filename** | **Description** |
| b06\_scan.v | ITC’99 b06 netlist (full-scan) design |
| b06\_scan.spf | ITC’99 b06 STIL procedure file |
| b10.v | ITC’99 b10 netlist (non-scan design) |
| b10\_scan\_insertion.tcl | Design Compiler .tcl script for scan-chain insertion for the b10 circuit |
| b12.v | ITC’99 b12 netlist (non-scan design) |
| pdt2002.v | Technology library model |
| pdt2002.db | Technology library model database (for Design Compiler) |
| pdt2002.dc\_setup.tcl | Technology library setup script (for Design Compiler) |

# **Appendix B: TestMAX Flow**

**BUILD-T>**

***Read the VHDL/Verilog library models and netlist:***

* read\_netlist <HDL file name> –library [-insensitive]
* read\_netlist <HDL file name> –master [-insensitive]

***Elaborate the top-level***

* run\_build\_model <top-level module name>

build

*Fault simulation*

*ATPG*

***Set fault model***

* set\_faults -model stuck

***Create fault list or import it (one of the following)***

* add\_faults -all
* add\_faults <instance name> **or** add\_faults -module <module name>
* read\_faults <file name> -add [-force\_retain\_code] [-maintain\_detection]

***Read external patterns***

* set\_patterns -external <STIL file>

***Check external patterns***

* run\_simulation [-sequential]

***Run fault simulation***

* run\_fault\_sim [-sequential]

drc

**TEST-T>**

***Add clock/reset signals (not needed if you use an .spf file)***

* add\_clock <off-state value (01)> <signal name>

***Add Primary input constraints and output masks if needed***

* add\_pi\_constraints <01X value> <input port name>
* add\_po\_masks <output port name>

***Run default DRC or use .spf file***

* run\_drc [<SPF file name>]

**DRC-T>**

***Select internal patterns***

* set\_patterns -internal

***Set ATPG options (check the manual)***

* set\_atpg -help **or** man set\_atpg

***For sequential circuits***

* set\_atpg -full\_seq\_atpg

***Run ATPG***

* run\_atpg -auto\_compression

Run ATPG

run\_atpg[-sequential]

***Report summaries***

* set\_faults -summary verbose -fault\_coverage
* report\_summaries

***Write fault list***

* write\_faults <file name> -all -uncollapsed -replace