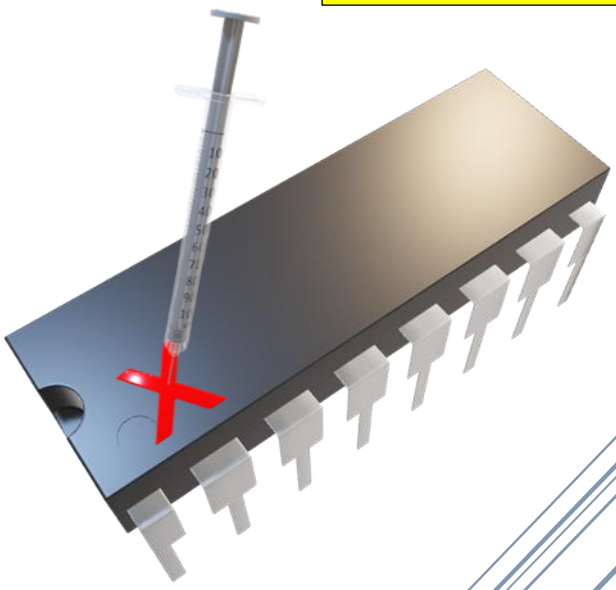
10 October 2024

TESTING AND FAULT TOLERANCE

Laboratory Session 1: "Fault Injection Concepts"

Student Name	Student Surname	Student ID
Manuel	Riso	S329514

SAVE THIS FILE IN PDF AND SUBMIT IT ON "EXERCISE"



Riccardo Cantoro
(riccardo.cantoro@polito.it)

Michelangelo Bartolomucci
(michelangelo.bartolomucci@polito.it)

Files and Scripts of LAB1

All files listed here are included in your remote /home directory under lab1 folder.

Filename	Description
ex1_dataflow.vhd	ex1 circuit (dataflow variant)
ex1_structural.vhd	ex1 circuit (structural variant)
ex1_structural_tmax.vhd	ex1 circuit (structural variant, TestMAX compliant)
ex1_testbench.vhd	Testbench for ex1 circuit
pdt2002.v	Technology library models (for simulation of structural/synthesized circuits)
pdt2002.db	Technology library models (for synthesis)
convert_faults.sh	Bash script that converts TestMAX faults to QuestaSim force commands
synthesis.sh	Bash script that invokes DesignCompiler
synthesis_script.tcl	DesignCompiler script that performs the logic synthesis of ex1 circuit
simulation_script.tcl	QuestaSim script with commands for the logic simulation run
tmax.sh	Bash script that invokes TestMAX
tmax_script.sh	TestMAX script that produces the fault lists of ex1 circuit

[A] Circuit Fault Lists

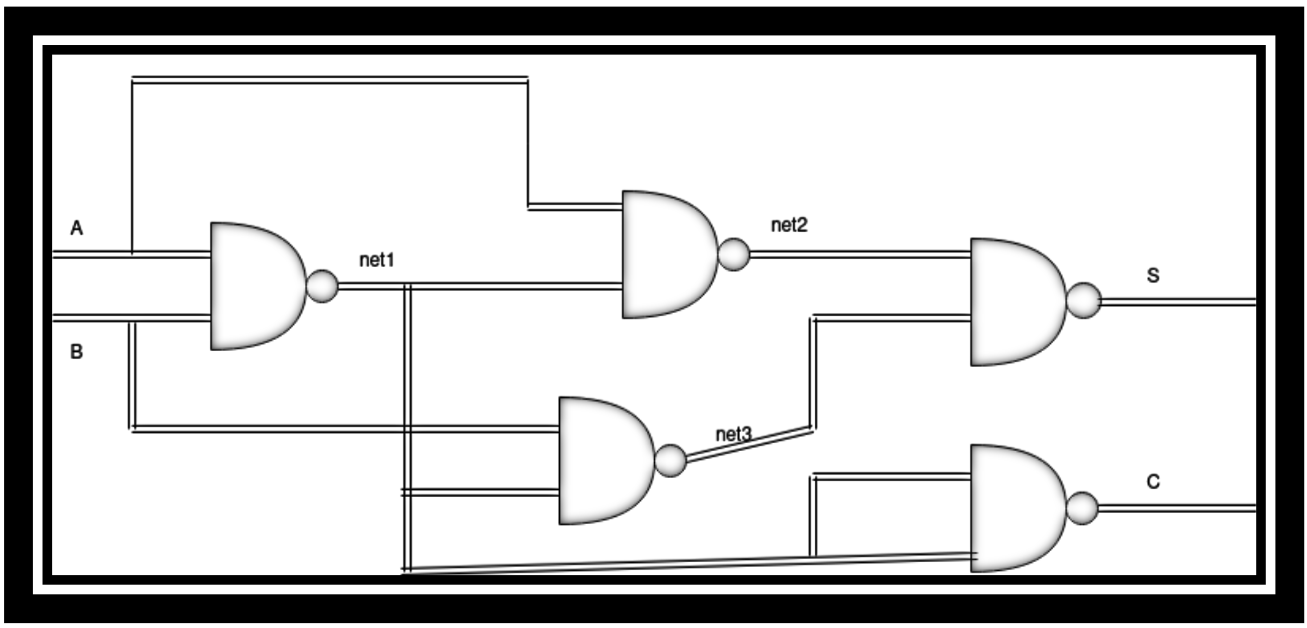
The combinational circuit ex1 is provided in both *dataflow* and *structural* formats.

- The dataflow format is a unique module which makes use of VHDL primitives (and, or, nand, xor etc.) and wires.
- The structural format instantiates sub-module (so called cells described in the technology library (pdt2002).

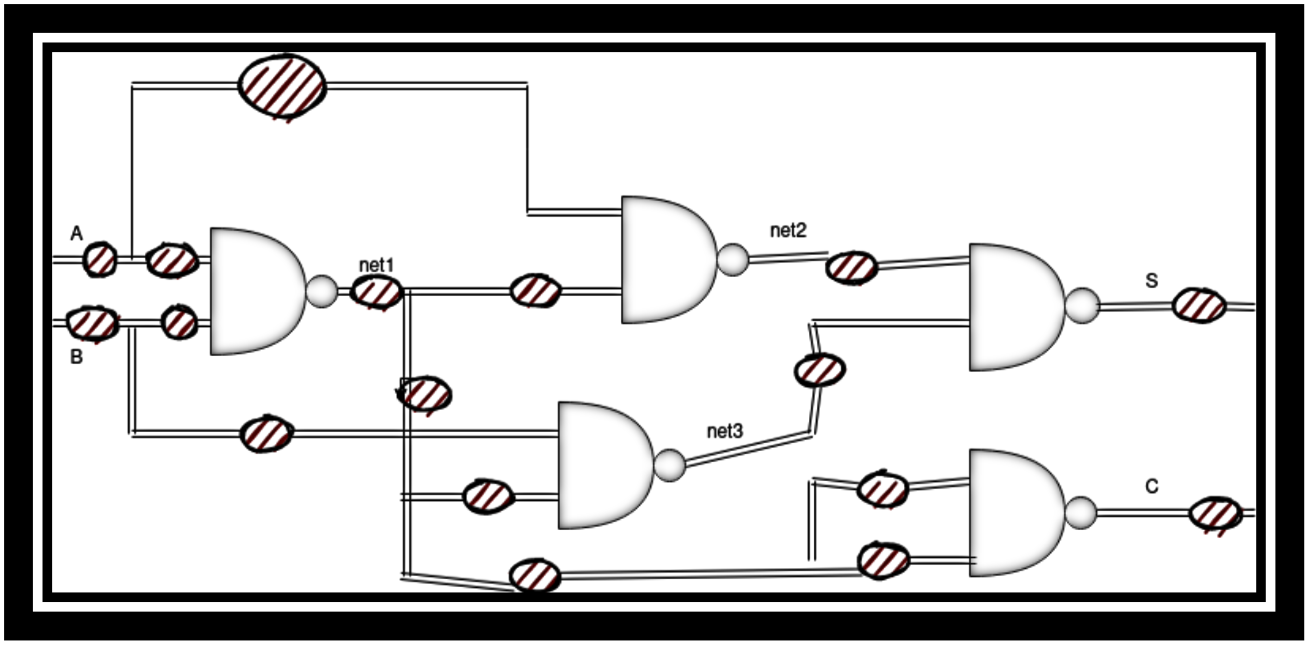
A structural version of the circuit can be obtained from a high-level description of the circuit (behavioral, dataflow) by means of logic synthesis.

[A.1] Tasks:

1. Draw the gate-level schematic of the ex1 circuit.
2. Compute the size of the full stuck-at fault list.



The total amount of stuck-at fault is 34.



💡 You can use draw.io to draw the circuit.

Run the provided TestMAX script and check the results. The provided script is responsible for the following actions:

1. Read the structural circuit files and the technology library.
2. Build the top-level module.
3. Perform the design rule checking (drc).
4. Set the fault list type (full or collapsed).
5. Add all faults.
6. Write the faults to a file.

After the successful execution of the script you will find two new text files in the directory lab1/run. By checking the content of the full fault list, you can understand how faults are grouped in **equivalence classes**. An equivalence class can be easily identified in the full fault list, where a fault is (eventually) followed by one or more lines that contain the symbols "- -". All such faults, including the one without the mark, are equivalent and can be collapsed. The collapsed fault list contains the so-called **prime** faults.

[A.2] Tasks:

1. Report the total number of equivalence classes for the circuit.
2. Explain how the prime faults are selected from the full fault list.

The total number of equivalent classes is 20.

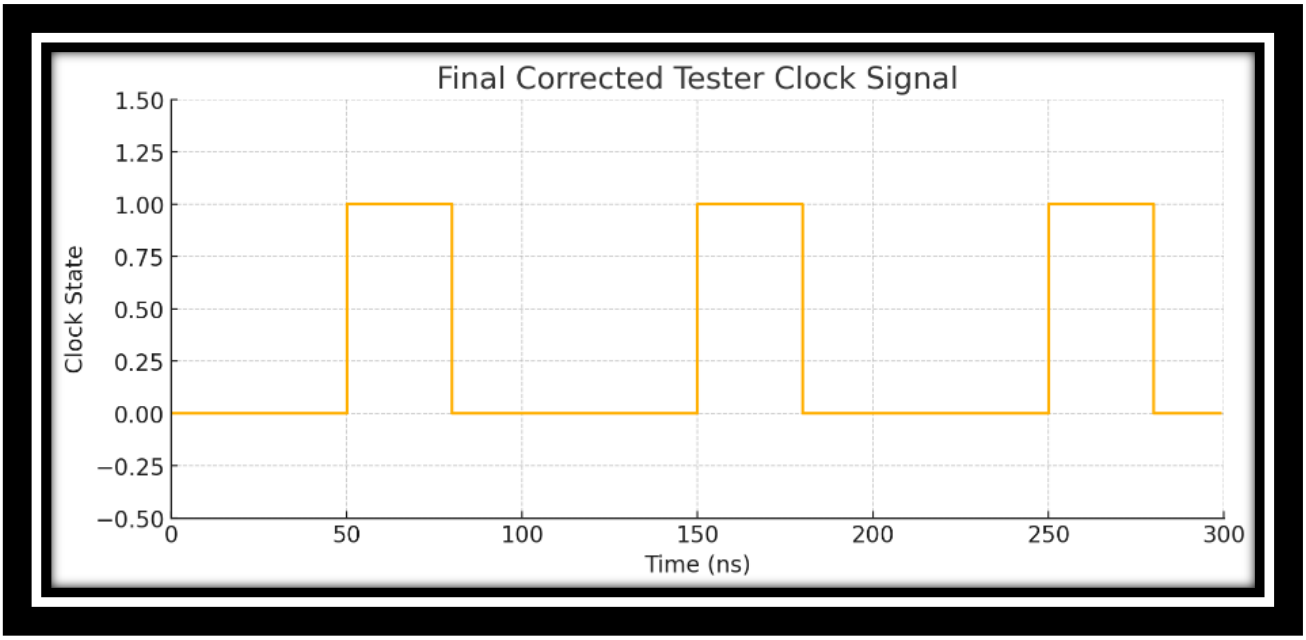
For every equivalent class, is selected only one fault from the full fault list, so the number of fault is collapsed, so it's reduced.

[B] Testbench (TB)

A VHDL testbench for the circuit ex1 is provided. It can be used to apply stimuli to the circuit and to monitor the circuit responses to that stimulus. Look inside the source code of the testbench.

[B.1] Tasks:

1. Draw the waveform of the clock signal.
2. Annotate on the previously drawn waveform:
    a. The exact times in which stimuli is applied.
    b. The exact times in which the circuit outputs are observed.



The circuit outputs are observed after 0ns.

Run the testbench using the provided bash script, which invokes the QuestaSim shell and executes the simulation `tcl` script. The testbench includes a "monitor" process, which reports, for each pattern, the values applied to the primary inputs (PIs) and the values observed on the primary outputs (POs). After the logic simulation, the monitor produces a new text file in in the directory lab1/run.

The objective of this exercise is to inject faults in the circuit and then to compare the results against the fault-free simulation: in case the values of a primary output changes for at least a pattern, then the fault is detected. The procedure to follow in the fault injection campaign is the following:

1. Run the fault-free simulation and rename the monitor text file (e.g., monitor_gold.txt)
2. Inject a fault and then re-run the simulation, which produces the new `monitor.txt`
3. Compare monitor_golden.txt and monitor.txt

For comparing, you can use the cmp or diff binaries/commands available in Linux.

Faults can be injected by modifying the simulation tcl script. Inside the tcl script you can find some examples. In order to check the behavior of the internal signals when a fault is injected, you can run the simulation using the QuestaSim GUI. In order to run the GUI, you need to slightly modify the bash script simulation.sh by commenting out the command that invokes the vsim shell and uncommenting the one that invokes the GUI.

Inside the GUI window, there is also a terminal that can be used to execute commands. For example, you can copy and paste the commands of the tcl script. Furthermore, you can add the waveforms of the internal signals **before** executing the simulation with the run command. Also, you can inspect waveforms produced in previous simulations.

Waveforms are written into wlf files. The name of the output file can be specified by specifying it with the -wlf parameter of the vsim command (like in simulation.sh). You can view a wlf file by executing the following command:

```
vsim -view FILENAME.wlf
```

[C] Pin Faults

QuestaSim can inject faults in the wires (i.e., VHDL signals) that connect the various gates of the circuit, as you have seen in the previous exercise (see [B] Testbench (TB)). However, if you look at the fault list produced by TestMAX in the first exercise (see [A] Circuit Fault Lists), you will notice that such faults are related to input and output pins of the gates. You can observe via logic simulation that, if you force a gate pin to 0 or to 1, then all the other gates that are connected to that pin by a single wire are also forced to the same value (as well as the wire itself). This behavior is fine if the wire interconnects only two gates.

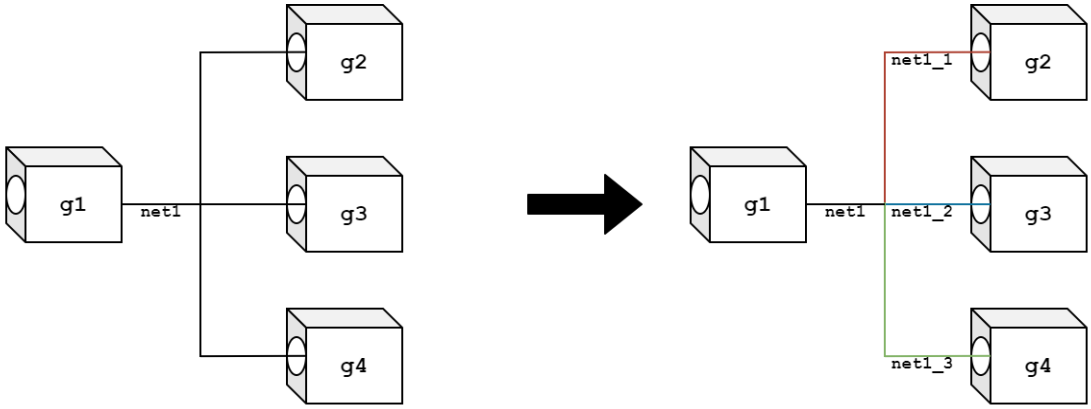
[C.1] Tasks:

1. **Identify** and **report** the faults in the fault list produced by TestMAX for the ex1 circuit that cannot be properly injected using the current version of the circuit.

In the case of the fault list produced by TestMAX for the ex1, examinating the full fault list generated, I can notice some faults that cannot be injected properly:

- U1/I1
- U2/I2
- U1/I2
- U3/I2
- U3/I1
- U2/I1
- U5/I1

To overcome this problem, pin faults can be emulated by adding additional wires to the circuit, following the example in the figure below.



(a) (b)

```
g1: ... port map ( ... O  => net1 ...);
g2: ... port map ( ... I1 => net1 ...);
g3: ... port map ( ... I1 => net1 ...);
g4: ... port map ( ... I2 => net1 ...);
```

```
...
g1: ... port map ( ... O  => net1   ...);
g2: ... port map ( ... I1 => net1_1 ...);
g3: ... port map ( ... I1 => net1_2 ...);
g4: ... port map ( ... I2 => net1_3 ...);
...
net1_1 <= net1;
net1_2 <= net1;
net1_3 <= net1;
...
```

In the left circuit (a), when you inject a stuck-at fault in g4/I2, all the other pins connected to net1 are stuck at the same value (wrong implementation of the single stuck-at fault model). In the right circuit (b), g4/I2 is connected to net1_3, which is the only wire in the circuit that would be stuck at the same value of the pin, in case you do a force, while the other 3 nets would keep their functional values (correct implementation).

Apply the proper modifications to the circuit ex1 by modifying the structural VHDL file. Then, perform a fault injection campaign on the circuit using the fault list produced on the first exercise (see [A] Circuit Fault Lists).

You can use the provided script to convert a line in the fault list file to a force command for the simulation, as in the example below (the third parameter is the line number):

```
./convert_faults.sh run/fault_list_stuckat_full.txt 14 > run/inject_fault.tcl
```

In the example, the force command is written to a tcl file, which can be included in the simulation script (uncomment the line "source inject_fault.tcl"). In this way, you can simply choose another line in the fault list and re-run the simulation with another fault injected.

[C.2] Tasks:

1. Complete the table below.
2. **Report** the faults in the circuit that are untestable.
3. **Report** any redundant test pattern.

PIs	POs	Excited /not observed faults	Detected faults
00	00	C/s0, U5/I2/s0, U3/I2/s1	A/s0, A/s1, B/s0, B/s1, U1/O/s0, U1/I1/s0, U1/I2/s0, U1/O/s1, S/s0, S/s1, U3/I1/s1, U3/O/s1, U2/I2/s1, U2/I1/s1, U2/O/s1, U5/I1/s1, C/s1, U5/I2/s1
01	10	U1/I2/s0, U1/O/s1, S/s0, S/s1, U3/I2/s1, U3/I1/s1, U3/O/s1, U2/I2/s1, U2/I1/s1, A/s0, U2/O/s1, U5/I1/s1,A/s1, B/s0, U1/I1/s1, U5/I2/s1	B/s1, U1/O/s0, U1/I1/s0, C/s0, U5/I2/s0, U1/I2/s1, C/s1,
10	01	U5/I2/s0	A/s0, A/s1, B/s0, B/s1, U1/O/s0, U1/I1/s0, U1/I2/s0, U1/O/s1, S/s0, S/s1, U3/I2/s1, U3/I1/s1, U3/O/s1, U2/I2/s1, U2/I1/s1, U2/O/s1, U5/I1/s1, C/s1, U1/I1,s1, C/s0,
11	10	U1/I1/s0, U1/O/s1, S/s0, U3/I2/s1, U2/I1/s1, A/s0, B/s0, B/s1, U1/I2,s1, U5/I2/s1	A/s1, U1/O/s0, U1/I2/s0, C/s0, U5/I2/s0, U1/I1/s1, S/s1, U3/I1/s1, U3/O/s1, U2/I2/s1, U2/O/s1, U5/I2/s1, U5/I1/s1, C/s1

Every fault inside the collapsed faults list can be tested, there's at least one case in which the outputs differs from the golden ones.

For the redundant test pattern, there are some that repeats the same outputs, but in general different inputs generate, for at least one fault, different outputs, so we don't have redundant test patterns.

However, in a lot of situations, we have different single fault inside the circuit that generate the same outputs for every input patterns.

[D] Synthesized Circuit

The dataflow version of circuit ex1 can be synthesized with DesignCompiler using the provided script. After the successful execution of the script, you can find the synthesized circuit in both VHDL and Verilog formats. You can use the Verilog file for the TestMAX work (VHDL files need to be adjusted), while the VHDL is fine for the logic simulation. In order to use the synthesized file, modify the path in the simulation.sh script accordingly.

[D.1] Tasks:

1. Perform the fault injection campaign on the new, synthesized circuit.
2. Report any redundant fault(s).

I've performed the fault injection campaign on the collapsed fault list, there're some faults that generate the same output patterns, for every single input pattern.

For example, A/s0 generates the same outputs of U5/I1/s0, or B/s1 with U4/I2/s1, or A/s1 with U4/I1/s1.