TÉCNICO LISBOA

DESIGN, TEST AND RELIABILITY OF ELECTRONIC SYSTEMS

2nd Project

2023/2024

1. Objective

The objective of this project is to add **BIST-per-scan** to the circuit assigned to your group.

2. General Specifications

2.1 The *Single Stuck-At* fault coverage for the whole design must be above **95%**. This fault coverage must be obtained in a **single BIST session**, i.e., activation of the **`bist_start`** signal to initiate the BIST session and then wait until the end of the BIST session (**`bist_end`** = 1). The inputs of the CUT (Circuit Under Test) **cannot change** while the BIST is running.

2.2 The maximum number of test vectors that can be applied is 1,024.

2.3 All Verilog modules (excluding the *testbench*) must be synthesized using the *RTL Compiler* (`genus`) from *Cadence.*

2.4 The `hal` tool must be used to check for good Verilog coding style. All **behavioral** Verilog files (excluding the *testbench*) must be clean of errors classified as `Severity: Fatal` and `Severity: Error`.

2.5 The faults must be inserted in all modules described in Verilog (excluding the *testbench*).

2.6 The inputs and outputs of the circuit assigned to you must be included in the port list of the top-level module (module that interconnects all other modules).

2.7 Must use maximum-length LFSRs (primitive polynomials).

2.8 The observable nodes for the fault simulator are **only** the outputs named **`bist_end`** and **`pass_nfail`**.

2.9 The controller interface described in the next section must be respected.

NOTE: To reach the value specified in 2.1, you may need to apply the BIST optimization techniques presented in the section "*BIST Optimization*" of chapter "*6. Built-In Self-Test*".

3. Specifications of the BIST Controller

NOTE: The signals **`clock`**, **`bist_start`**, **`reset`** and **`bist_end`** are identical to the 1st project.

Inputs of the BIST controller

`clock`: The controller must be sensitive to the rising edges of the `clock`.

`bist_start`: After a 0→1 transition in this signal, a new sequence must be initiated. The **`bist_start`** value is captured on the rising edge of the clock. While the full sequence is not completed, further transitions in the **`bist_start`** signal must be ignored.

reset: The **reset** must be synchronous and active at the logic level '1'. This signal is used to restart the state machine and the counters, preparing the controller to start a new sequence. When the **reset** signal goes LOW and the **bist_start** signal is HIGH, a new sequence **must not** be restarted. The controller must wait for a 0→1 transition in the **bist_start** signal to start a new sequence.

Outputs of the BIST controller

bist_end: The **bist_end** signal must go HIGH when the sequence is completed. The **bist_end** signal must go LOW when the **reset** signal is activated or the **bist_start** signal has a 0→1 transition.

pass_nfail: This signal contains the information *pass/fail* (1/0). The **pass_nfail** signal must hold its value until a **reset** or a **bist_start** condition is activated.

4. Recommendations

1. As a first step, synthesize the module with the scan chain and store the synthesis result in a Verilog file. In a second step, synthesize this module together with all remaining behavioral Verilog files. By default, the RTL Compiler automatically removes the hierarchy of the project, when the synthesis effort is set to high (`-effort high`). So, the final synthesis result is a single Verilog module.
2. To demonstrate that the insertion of the BIST did not change the operation of the original circuit, you should
   a. simulate the circuit with the list of vectors provided on the website before inserting the BIST controller, and later,
   b. simulate the whole circuit with the same vectors after insertion of the BIST controller.

   The behavior must be identical in both simulations.
3. Since the execution of the tools generate a lot of files, it is recommended that you save your own files in folders (e.g., sources, testbenches, scripts, etc.). When the commands in the `Makefile` are executed, they should get the input files from those folders.

   The files submitted in the delivery must respect this structure:

   `Makefile` (file)

   `Report` (file)

   `Hal report` (file)

   `scripts` (folder): contains all scripts used for synthesis and simulations.

   `sources` (folder): contains only the Verilog source files, **no testbenches.**

   `testbenches` (folder): contains all testbenches and vectors file (`.vec`).

5. Delivery

The delivery package must be submitted in the **Fenix** and must include, at the least, the following items (items 1. and 2. are mandatory to reproduce the results, but they do not have impact in the evaluation):

1. `Makefile` to generate the synthesis and simulations results required for this project. This `Makefile` can be the same file provided on the website, customized for your project. Delete the commands that cannot be used in your project.

   Before submission, check that the `Makefile` can generate all results (synthesis, simulations and fault simulation), by just typing `make` or `make <target>`.

2. Scripts used for synthesis, fault simulation, etc. Those scripts can be the same files provided on the website but customized for your project.
3. Verilog source files
4. One testbench file for each of the following test cases:
    a. **Normal mode**: Normal operation using the test vectors provided for your circuit.
    b. **BIST mode**: BIST-per-scan operation (used to obtain the fault coverage defined in the specifications).
    c. **Controller mode**: Validation of the BIST controller interface (similar to the 1st project: activation of the **`reset`** while the sequence is running, reactivation of the **`bist_start`** while the sequence is running, etc.). **NOTE**: The fault coverage defined in the specifications cannot be obtained in this test case.
5. `Hal` report file (obtained using the option `File` → `Generate Report` …)
6. Report including …
    a. Block diagram (Verilog modules and interconnections between those modules). The buses must indicate their number of bits.
    b. LFSRs: polynomials and corresponding Verilog code.
    c. Excerpt pf the synthesis report from `genus` highlighting the **gate count** and the **area of the circuit**[1]. These parameters must be presented and compared for the following conditions:
        - Original circuit (without the scan chain and without the BIST modules)
        - Full circuit (includes scan chain and all additional modules to implement the BIST)
    d. Report of the fault simulator including the fault coverage and the list of undetected faults.
    e. Analysis of the results and conclusions (can include suggestions to improve the project).

6. Evaluation Criteria

The evaluation is based on the following criteria:

3. Quality of the Verilog code: **20%** (synthesis errors and warnings have a negative impact in this criteria)
4. Simulation results: **30%**
5. HAL report: **3,75%**

6.a. Block diagram: **8,75%**

6.b. LFSR polynomials and corresponding Verilog code: **7,5%**

6.c. GENUS reports and analysis of results: **13,75%**

6.d. Report of the fault simulator: **11,25%**

6.e. Analysis of the results and conclusions: **5%**

[1] To get the gate count and the area of the circuit, you can use the command "`report_gates`" in `genus`.