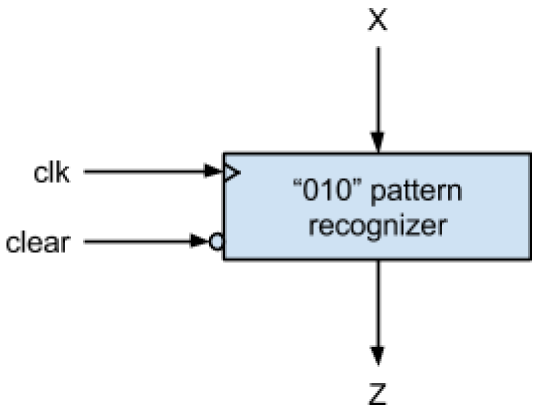
Lab 3: Design and implementation of a pattern recognizer

1 Overview

The goal of this lab is to design and build a sequential “010” pattern recognizer. The system behavior will be defined with a Moore machine.
The circuit will have the set of ports depicted in the Figure below:

- A binary input, **X**, through which bits are received
- A binary output, **Z**, which is set to “1” when the last three bits received forms the sequence “010”, and “0” in any other case.
- A **clk** input, the clock.
- A **clear** input, for asynchronous initialization (in Low, when clear=0).



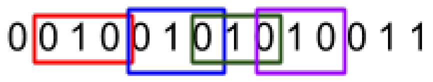
2 Development

2.1 Design phase

- The circuit design must be completed at home. It must be summarized in the corresponding lab notebook.
- The behavior of the circuit must be designed as a Moore machine using D flip-flops, NAND gates and inverters.

2.2 Building and debugging phase

- Use the following ICs to mount the circuit: **7474** (2 D flip-flops, triggered by rising edge and asynchronous initialization signals - to 0 and 1 - active Low), **7400** (4 2-input NAND), **7410** (3 3-input NAND gates), and **7404** (6 inverters).
- Connect **X** and **clear** inputs to switches.
- Connect **clk** to a pulse switch
- Connect **Z** to a led
- Check that you circuit recognizes the pattern “010” with overlapping. To this end, try some sequences that include several “010” patterns, like this one:



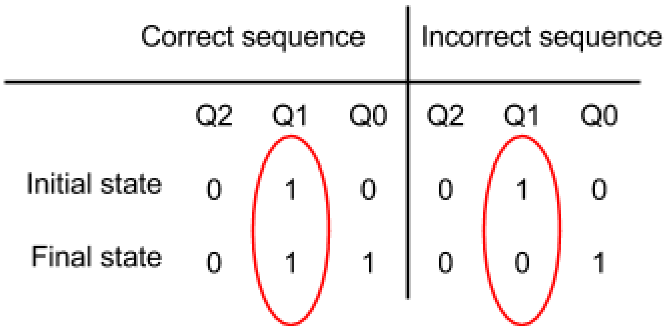
- If the circuit does not correctly work, then it must be debugged to find the errors and fix them. To debug a sequential system we must follow the algorithm below.
- If the circuit is correctly working, show the final design to you lab. professor.

3 Debugging sequential circuits

The following will explain a method for determining the errors that occur in the sequential behavior of a system that has been designed and implemented.

To facilitate this task, it is desirable to connect the values of the current state (even if they are not outputs of the specification) to the LEDs or the display to quickly check the state transition that is failing.
We must configure the wrong case, i.e., the binary output that is not the expected one for a given input. Then, we proceed as follows:

i. Review the design we are implementing looking for errors.
   a. If the circuit can be perfectly simulated then we may ensure that the design is correct.
ii. Once we are sure that the error is due to the physical (hardware) implementation, the sequence of states must be analyzed for the input that triggers the error.
   a. If the state transition is correct, then the error is the combinational network that computes the output (CN1). Apply the algorithm to debug combinational circuits to this module.
   b. If the state transition is not correct, then the error is located in the combinational network that computes the next state (CN2) or the set of flip-flops. Go to point iii.
iii. Configure the circuit in the state of the incorrect transition. Let's consider the following incorrect state transition:

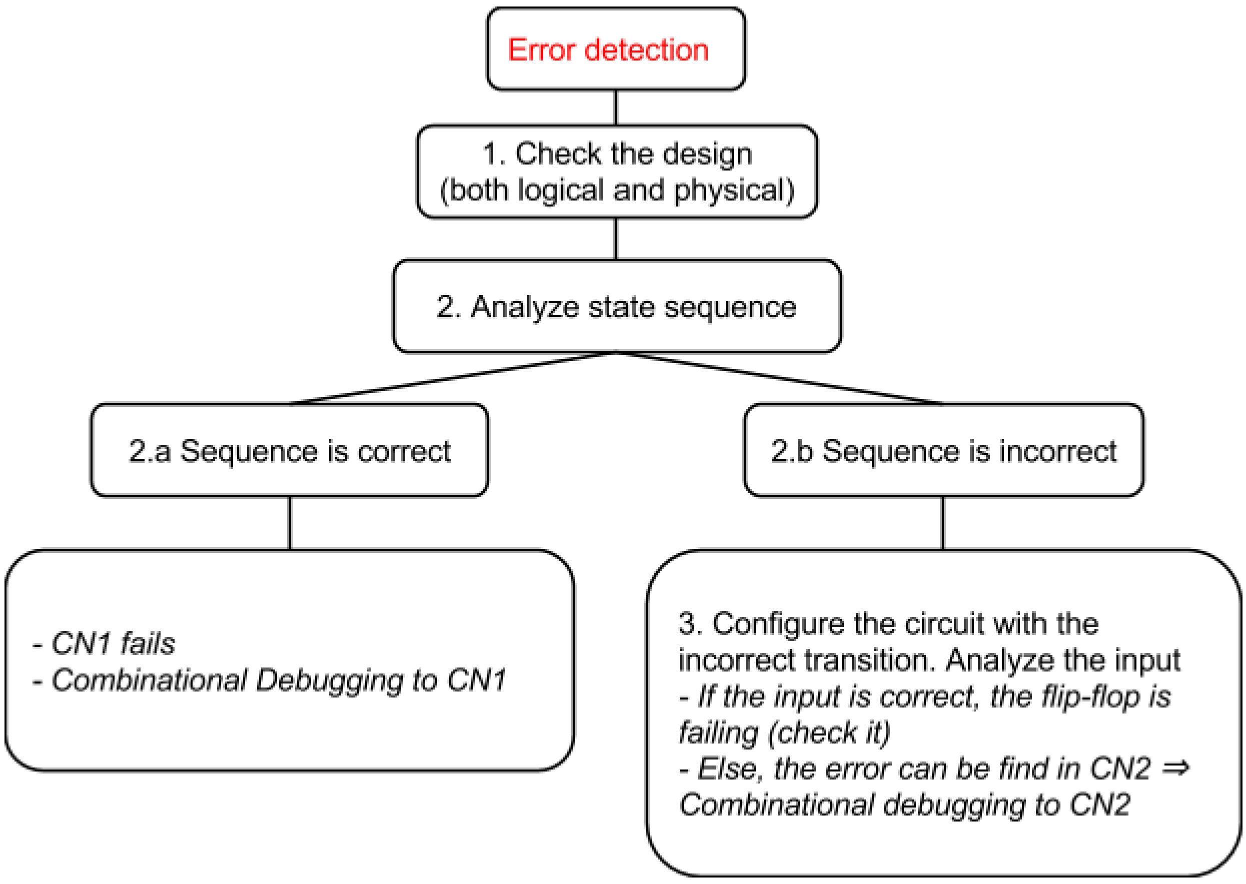


	Correct sequence			Incorrect sequence		
	Q2	Q1	Q0	Q2	Q1	Q0
Initial state	0	1	0	0	1	0
Final state	0	1	1	0	0	1

In this example we may observe that instead of the correct transition 010→011, the system performed the incorrect transition 010→001. The first step is to find the flip-flop that produced the incorrect output, in this example Q1 (it is the only flip-flop with an incorrect bit). Once we have identified the incorrect flip-flop:

- If we observe that with the current input and with the flip-flop truth table we are going to obtain the incorrect final state, then the flip-flop is perfectly working (in other words, although the next state was incorrect, the input was configured to trigger this incorrect transition). Thus, the error is in CN2, the combinational network that generates the next state. Apply the algorithm to debug combinational circuits to CN2.
- On the contrary, if we observe that the current input is correct, i.e., it will trigger a correct state transition, then the problem is with the flip-flop. There exist several reasons for this malfunctioning:
  - The clock signal is not properly configured
  - The asynchronous inputs have been activated
  - Flip-flop bad polarized
  - Broken chip

The following Figure summarize the process:


Error detection
1. Check the design (both logical and physical)
2. Analyze state sequence
2.a Sequence is correct
2.b Sequence is incorrect
- CN1 fails
- Combinational Debugging to CN1
3. Configure the circuit with the incorrect transition. Analyze the input
- If the input is correct, the flip-flop is failing (check it)
- Else, the error can be find in CN2 ⇒ Combinational debugging to CN2