VHDL Assignment #6: Finite State Machines in VHDL

In this coding assignment you will learn how to describe finite state machines (FSMs) in VHDL. Specifically, you will design a sequence detector circuit.

1 Instructions

- Due date: Friday, December 4, 2020 by 5:00pm.
- Submission is in teams using myCourses (only one team member submits). In the report, provide the names and McGill IDs of the team members.
- Late submissions will incur penalties as described in the course syllabus.

2 Learning Outcomes

After completing this lab you should know how to:

- Design FSMs in VHDL
- Design sequence detector circuits

3 Sequence detector

In this assignment, you will first implement a sequence detector circuit that takes a sequence of bits as its input and detects two different bit patterns in the input sequence. Specifically, you are asked to design a circuit based on a Moore-type FSM(s) with an asynchronous reset and an enable signal, that takes a sequence of bits as its input "*seq*" and has two outputs "*out_1*" and "*out_2*". The outputs should be "$out\_1 = 1$" and "$out\_2 = 1$" at the clock cycle following the input bit patterns 1011 and 0010, respectively; they should be 0, otherwise. In other words, "*out_1*" detects the sequence 1011, and "*out_2*" detects the sequence 0010. Note that state transitions only occur when the enable signal is high. Otherwise, the FSM stays at its current state. An example of the desired behavior is

seq	0011011011001001011011
out_1	0000000100100000000100
out_2	0000000000000010010000

Note: It is best to use two FSMs, each detecting one of the two bit patterns. Your VHDL code will be based on the state diagram(s) for your FSMs. So, it is important that you first come up with a simple state diagram. In fact, you should not need more than five states for each of the two FSMs.

Use the following entity declaration for your VHDL description of the sequence detector:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity sequence_detector is
Port (seq        : in  std_logic;
      enable     : in  std_logic;
      reset      : in  std_logic;
      clk        : in  std_logic;
      out_1      : out std_logic; -- generates 1 when "1011" is detected; otherwise 0.
      out_2      : out std_logic); -- generates 1 when "0010" is detected; otherwise 0.
end sequence_detector;
```

Once you have your circuit described in VHDL, you should simulate it. Write a testbench and perform a functional simulation for your VHDL description of the FSM-based circuit.

4 Sequence Counter

In this part, you are required to implement a sequence counter circuit that counts how many times each of the two patterns occurs in the input bit stream. The sequence counter circuit can be realized using the sequence detector circuit, which detects the patterns, followed by two 3-bit up-counters (one for each output of the FSM-based circuit) that keep track of the number of detected patterns. More specifically, each counter is incremented whenever its corresponding pattern has been detected. Use the sequence detector and the 3-bit up-counter (from VHDL5) to implement the sequence detector circuit with an asynchronous reset and an enable signal.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity sequence_counter is
Port (seq        : in  std_logic;
      enable     : in  std_logic;
      reset      : in  std_logic;
      clk        : in  std_logic;
      cnt_1      : out std_logic_vector(2 downto 0); -- counts the occurrence of "1101".
      cnt_2      : out std_logic_vector(2 downto 0)); -- counts the occurrence of "0010".
end sequence_counter;
```

An example of the desired behavior is

seq	0011011011001001011011
cnt_1	0000000111222222222333
cnt_2	0000000000000011122222

Note that when the enable signal of the sequence detector circuit is low, its counter and FSM-based instances hold their previous value and state, respectively.

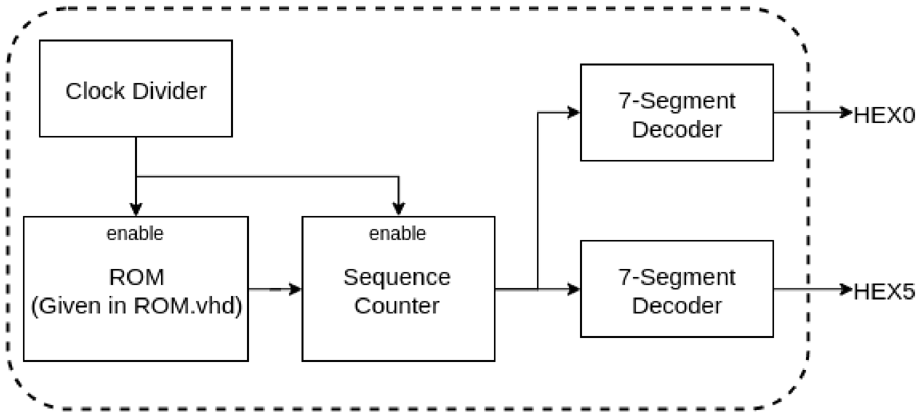
5 Reading the input sequence from ROM

So far, you have designed a circuit that counts the number of occurrences of two different patterns within the input sequence of bits. To demonstrate the functionality of your circuit, you will test it using a sequence of bits, and show the number of occurrences of each pattern.The test sequence is stored in a read-only-memory (ROM) and the bits are provided to the circuit one after the other every 1 second. The ROM circuit inputs are clock, asynchronous reset, and enable signals, and the output is a single bit of the under-test sequence. The ROM produces each bit at the positive edge of the clock signal when the enable signal is high. The VHDL code for the ROM is given in `ROM.vhd` (attached to this assignment). To read each bit of the under-test sequence every 1 second, you will need to create an instance of the clock divider circuit that you designed in VHDL5 assignment to enable both the ROM and sequence counter circuits at the appropriate rate (once per second).

The 7-segment decoder that you designed in VHDL5 assignment, will be used to display the occurrence number of each pattern. Similar to VHDL5, output in std logic vector will be sufficient.

To avoid long simulation delays, we will use 10 Hz as the clock frequency for of all the components (*i.e.*, clock divider, ROM, and sequence counter).

The high-level architecture of the circuit is shown in the following figure.



Use the following entity declaration to wrap up the complete circuit:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity wrapper is
Port (reset    : in  std_logic;
      clk      : in  std_logic;
      HEX0     : out std_logic_vector (6 downto 0),
      HEX5     : out std_logic_vector (6 downto 0));
end wrapper;
```

6 Deliverables

Once completed, you are required to submit a report explaining your work in detail, including answers to the questions related to this assignment. You will find the questions in the next section. You must also submit all of your VHDL, run.do, and .sdc files along with Log content of the synthesized circuits and timing analysis (remember to use the "create_clock -period 20 [get_port clk]" command for timing analysis in EDAPlayground) You must also submit all of your testbench files. If you fail to submit any part of the required deliverables, you will incur grade penalties as described in the syllabus.

7 Questions

Please note that even if some of the waveforms may look the same, you still need to include them separately in the report.

1. Why is it better to use two FSMs, rather than one, in the implementation of the sequence detector from Section 3?
2. Briefly explain your VHDL code implementation of all circuits.
3. Provide waveforms for each of the individual circuits (each section) and for the wrapper.
4. Perform timing analysis of the wrapper and find the critical path(s) of the circuit. What is the delay of the critical path(s)?
5. Report the number of pins and logic modules used to fit your 3-bit up counter design on the FPGA board.