Finite State Machines

Laboratory Assignment 11
ECE 201: Digital Circuits and Systems

Learning Objectives

After completing this lab, you will be able to

- design a finite state machine
- implement a finite state machine using a HDL
- understand the tradeoffs that come with the way in which a FSM is realized on a FPGA

Equipment and Materials

- DE10-Lite Board
- Intel Quartus Prime Design Software v15.1

Pre-Lab Assignment

Prior to attending your assigned laboratory section, complete the following tasks

- Watch this video that provides a practical example of how a FSM can be used to carry out computation: Designing a Vending Machine Using Finite State Automata
- Complete the design the first FSM by determining the logic that will feed the flip-flops in part 1.

Part I: Sequence Detector

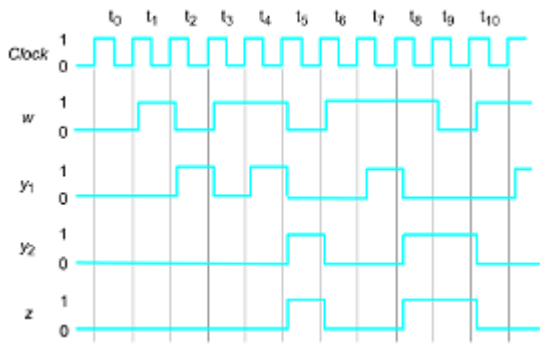

Figure 1: Required timing for the output z.

Your first task is to design and implement a simple Finite State Machine that detects a two-bit sequence (two consecutive ones coming in on a serial input, w). Please refer to the design explained in the notes. Since the design is provided, spend time studying the simulation waveform so that you can understand how the circuit functions and how it maps back to the original FSM specification. Design and implement your circuit on your DE-series board as follows:

1. Create a new Quartus project for the FSM circuit.
2. Write a VHDL file that instantiates D flip-flops in the circuit and which specifies the logic expressions that drive the flip-flop input ports. Use only simple assignment statements in your VHDL code to specify the logic feeding the flip-flops.

   Use the toggle switch SW_0 as an active-low synchronous reset input for the FSM, use SW_1 as the w input, and the pushbutton KEY_0 as the clock input which is applied manually. Use the red light $LEDR_9$ as the output z, and assign the state flip-flop outputs to the red lights $LEDR_8$ to $LEDR_0$.
3. Include the VHDL file in your project, and assign the pins on the FPGA to connect to the switches and the LEDs.
4. Simulate the behavior of your circuit.
5. Once you are confident that the circuit works properly as a result of your simulation, download the circuit into the FPGA chip. Test the functionality of your design by applying the input sequences and observing the output LEDs. Make sure that the FSM properly transitions between states as displayed on the red LEDs, and that it produces the correct output values on $LEDR_9$.

Part II: State Assignment Modification

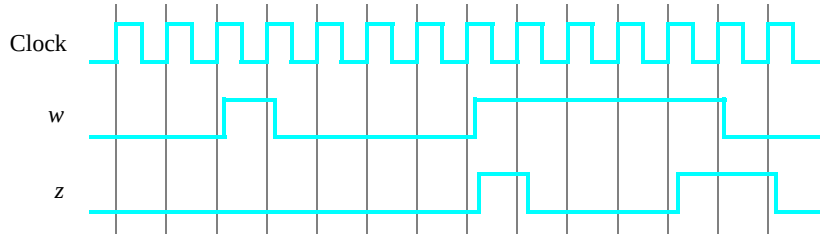

Figure 2: Required timing for the output z.

We wish to expand the finite state machine from Part I so that it now recognizes sequences of four consecutive 1s or four consecutive 0s. There is an input w and an output z. Whenever $w = 1$ or $w = 0$ for four consecutive clock pulses the value of z has to be 1; otherwise, $z = 0$. Overlapping sequences are allowed, so that if $w = 1$ for five consecutive clock pulses the output z will be equal to 1 after the fourth and fifth pulses. Figure 2 illustrates the required relationship between w and z.

(a) A state diagram for this new FSM is shown in Figure 3. For this part you are to manually derive an FSM circuit that implements this state diagram, including the logic expressions that feed each of the state flip-flops. Note that the one-hot code enables you to derive these expressions by inspection. To implement the FSM use nine state flip-flops called $y_8, \ldots, y_0$ and the one-hot state assignment given in Table 1.

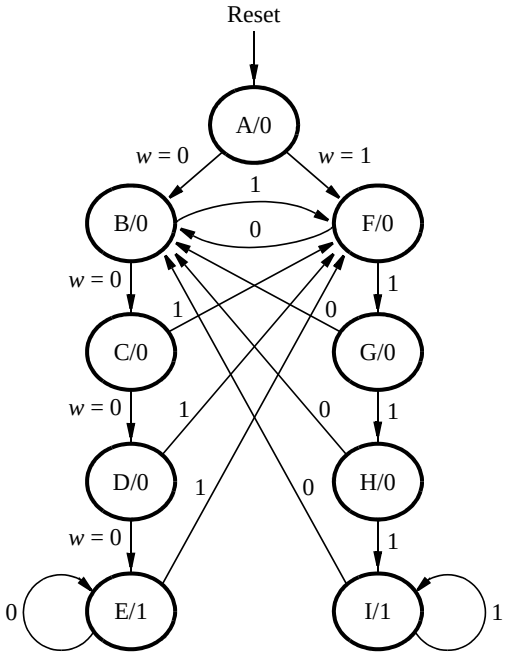

Figure 3: A state diagram for the FSM.

Name	State Code $y_8y_7y_6y_5y_4y_3y_2y_1y_0$
A	000000001
B	000000010
C	000000100
D	000001000
E	000010000
F	000100000
G	001000000
H	010000000
I	100000000

Table 1: One-hot codes for the FSM.

(b) Compile your new circuit and test it.

Part III: Shift Register Implementation

The sequence detector can be implemented in a straightforward manner using shift registers, instead of using the more formal approach described above. Create VHDL code that instantiates two 4-bit shift registers; one is for recognizing a sequence of four 0s, and the other for four 1s. Include the appropriate logic expressions in your design to produce the output z. Make a Quartus project for your design and implement the circuit on your DE-series board. Use the switches and LEDs on the board in a similar way as you did for Parts I and II and observe the behavior of your shift registers and the output z. Answer the following question: could you use just one 4-bit shift register, rather than two? Explain your answer.

Part IV: Coke Machine Controller

For this part, you are to write VHDL for a FSM using an alternative coding- style. In this version of the code you should **not** manually derive the logic expressions needed for each state flip-flop. Instead, describe the state table for the FSM by using a VHDL CASE statement in a PROCESS block, and use another PROCESS block to specify the output. For this assignment you will design a FSM controller for a "soda pop" machine. Design and implement your circuit on your DE-series board as follows:

(a) Complete problem 4 from Homework 13

(b) Complete problem 5 from Homework 13

(c) Create a new Quartus project for the FSM circuit.

(d) Write a new VHDL file that models the Coke machine FSM controller described in the homework. Use the toggle switch SW_0 as an active-low synchronous reset input for the FSM, use additional switches to represent the input. Use the pushbutton KEY_0 as the clock input which is applied manually. Use the red light $LEDR_9$ as the output. Create a signal that specifies the current state and is monitored via the red LED or 7-segment display.

(e) Include the VHDL file in your project, and assign the pins on the FPGA to connect to the switches and the LEDs.

(f) Simulate the behavior of your circuit.

(g) Once you are confident that the circuit works properly as a result of your simulation, download the circuit into the FPGA chip. Test the functionality of your design by applying the input sequences and observing the output LEDs. Make sure that the FSM properly transitions between states as displayed on the red LEDs, and that it produces the correct output values .