Faculty of Electrical & Electronics Engineering
BEE3233 Electronics System Design

Laboratory 3: Finite State Machine (FSM)

Mapping CO, PO, Domain, KI : CO2,PO3,P5,CTPS5
CO2: Construct logic circuit using HDL.[PO3, P5, CTPS3] CO4: Design finite state machines based on electrical & electronics engineering problem.[PO2, C5] PO3: Identify, formulate and provide effective solution to engineering problem P5: Complex Overt Response CTPS3: Ability to get ideas and find alternative solutions

Learning Outcomes:

a) The concept of finite state machine
b) Finite state machines programming in Verilog
c) Create text fixture for the system
d) Implement the proposed design on FPGA

General Xilinx Tips

1. SAVE EARLY AND OFTEN (in your own memory device!!)

   Xilinx is notorious for crashing at the most inopportune times. Do yourself a favor and save.

2. At the end of a lab session (or any work session), archive your project using the Xilinx utility (this will ensure you save everything), and save this zip archive on your ENIAC drive or on a flash drive. Do NOT assume files will remain on the lab computers or that "your" computer will be available at a later time.

3. Make sure all components are connected. Loose wires are a frequent cause of problems.

4. Try your hand at debugging first before calling me ☺. You will learn a lot by struggling through problems that seem hard at first.

5. Read all instructions carefully before starting the lab. Often, there will be a little detail that ends up being very important.

Introduction to Finite State Machines

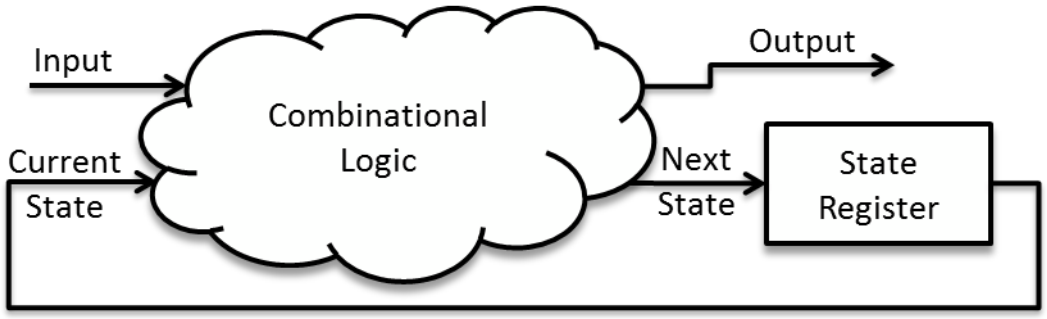


Figure 1. Schematic of a finite state machine

Finite state machines, or FSMs, are a broad class of sequential circuits. A FSM is a system which transitions from one discrete state to another, and has a finite number of states. The state is stored in a state register, typically implemented as a set of D-flip flops. The next state of the FSM is determined from its inputs and its current state. The output of the FSM may be determined from the input and present state (known as a Mealy machine) or the present state only (a Moore machine).

State diagrams are used to define how a state machine transitions from one state to the next. The first step in implementing a state machine is to draw its state diagram. A very simple state diagram is shown in Figure 2.

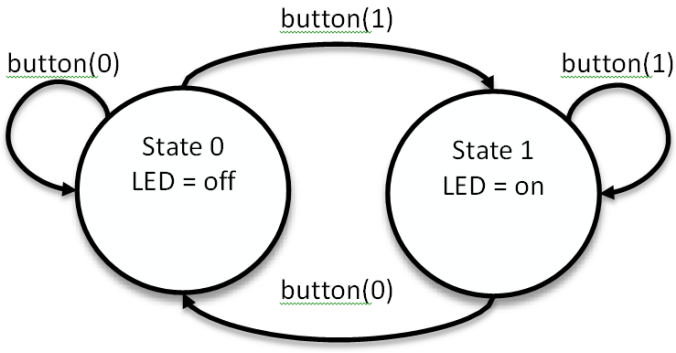


Figure 2. An example state diagram, for a simple state machine

This "bubble" state diagram has two states, State 0 and State 1. The name of the state and any outputs that are set in that state (LED) are shown inside the bubble. The state transitions, which depend on the inputs (button(0) and button(1)) are shown using arrows. This state diagram shows that if the machine is in State 0, and we press button(1), we'll transition to State 1 and the LED will turn on. Pressing button(1) again will do nothing. Pressing button(0), however, will transition back to State 0 and turn the LED off. If we were to implement this (or any other) state machine in a hardware description language such as VHDL, we would structure the code as shown in Figure 3. Using this kind of structure allows the synthesis tool to detect and optimize the state machine.

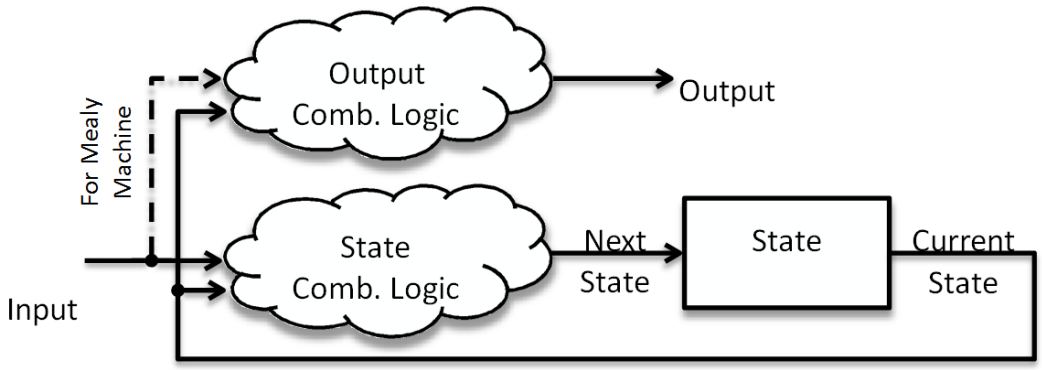


Figure 3. Typical FSM organization in HDL

Figure 3 shows the three parts of a VHDL implementation of a finite state machine:

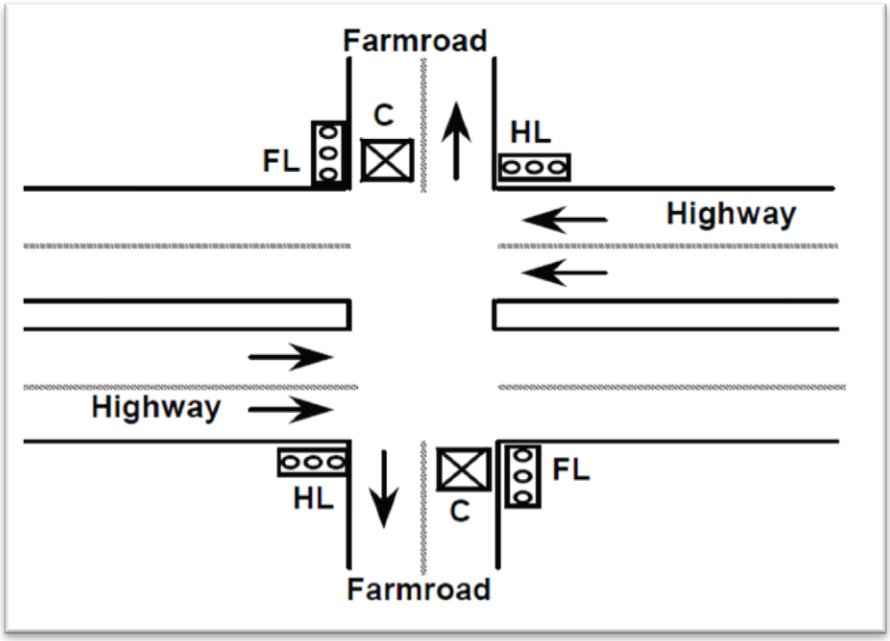
1. A combinational logic block that computes the output values given the current state, and optionally the input values.
2. A combinational logic block that computes the next state from the current state and the inputs.
3. A synchronous logic block that defines the state register's behavior.

This should become clearer as you progress on with this lab.

Mini Project 3 – Traffic Light Controller

Question:-

- A busy highway is intersected by a little used farmroad. Detectors C sense the presence of cars waiting on the farmroad. With no car on farmroad, light remain green in highway direction. If vehicle on farmroad, highway lights go from Green to Yellow to Red, allowing the farmroad lights to become green. These stay green only as long as a farmroad car is detected but never longer than a set interval.
- When these are met, farm lights transition from Green to Yellow to Red, allowing highway to return to green. Even if farmroad vehicles are waiting, highway gets at least a set interval as green.
- Assume you have an interval timer that generates a short time pulse (TS) and a long time pulse (TL) in response to a set (ST) signal. TS is to be used for timing yellow lights and TL for green lights.



Design a traffic light controller that satisfies the condition mentioned above.

1. Sketch the state diagram for the system using either Moore model or Mealy model.
2. Tabulate the state transition.
3. Verify your design in Xilinx ISE 10.1.
    a. Use Verilog to implement your system
    b. Create a testbench and simulate your system

NAME:-
ID:-

Evaluation Laboratory 3 (8%)

Print this and present it to me when you demonstrate your work.

Requirement	Points
Complete the lab on time (2 lab sessions)	/2
State diagram of the system	/2
Show the Verilog syntax code for the system	/2
Show simulations for the system • Accuracy • Comprehensive text fixture	/4
Implementation on FPGA Spartan 3E Board • Correct .ucf assignment • Display output on LCD	/4
Correct Answers • Q5, Q6, Q7	/6
Total	/20

1. State diagram of the system
2. State transition table
3. Write the Verilog syntax code for the system
4. Sketch / snapshot the simulation results
5. Record the FPGA resources consumed by your ALU design

FPGA Resources	Utilization	Total Available	Percentage Utilization
Slice LUTs (LookUp Tables)			
Bounded IOBs			

6. What is the maximum estimated frequency at which this design in your ALU can run?

   _____

   a. What is the corresponding minimum clock period?

   b. Maximum Frequency: _____ MHz Minimum Clock Period: _____ ns

7. Record the path reported under "Pad to Pad Report" – Place and Route Report.

Source Pad	Destination Pad	Delay

* Please hand in the design summary report of your ALU design along with this Lab 3 – Evaluation form.