Lab # 8

Finite state machine
FSM

December, 2013
Introduction

A state machine represents a system as a set of states, the transitions between them, along with the associated inputs and outputs. So, a state machine is a particular conceptualization of a particular sequential circuit. State machines can be used for many other things beyond logic design and computer architecture.

Why We Need FSM?

Problem
We’re hired to design a digital elevator controller for a four-floor building

- 1st Solution:
  Design a counter that counts up and down 00, 01, 10, 11, 10, 01, 00, ...
  Problem: Never stops!

- 2nd Solution:
  Add “Stop” button that disables counter.
  Problem: Have to press button when elevator happens by.

We need a way to have user inputs into a complex system

Counters - Next state based on current state

- If counter is in state ‘101’, next state is ‘110’
- No inputs (other than reset, enable)

Finite State Machines - Next state is a function of the current state and the inputs

- If the elevator is on floor 00 and the UP button is pressed on floor 10, then move to floor 01
- If current state is 00 if UP2, Next state is 01

Finite State Machines (FSM)

- Any Circuit with Memory is a Finite State Machine, Even computers can be viewed as huge FSMs.
- Design of FSMs Involves
  - Defining states
  - Defining transitions between states
  - Optimization / minimization
- Above Approach is Practical for Small FSMs Only.
- A finite state machine is a sequential logic circuit which moves between a finite set of states, dependent upon the values of the inputs and the previous state. The state transitions are synchronized on a clock.
- There are many ways to describe a finite state machine in VHDL. The most convenient is with a process statement. The state of the machine can be stored in a variable or signal, and the possible states conveniently represented with an enumeration type.
**Moore FSM**

Output is a Function of Present State Only.
Outputs are independent of the inputs, i.e. outputs are effectively produced from within the state of the state machine.
**Mealy FSM**

Output is a Function of a Present State and Inputs. Outputs can be determined by the present state alone, or by the present state and the present inputs, i.e. outputs are produced as the machine makes a transition from one state to another.
Moore vs. Mealy FSM:

- Moore and Mealy FSMs Can Be Functionally Equivalent, So Equivalent Mealy FSM can be derived from Moore FSM and vice versa.
- Mealy FSM Has Richer Description and Usually Requires Smaller Number of States.
- Mealy FSM Computes Outputs as soon as Inputs Change, So Mealy FSM responds one clock cycle sooner than equivalent Moore FSM.
- Moore FSM Has No Combinational Path between Inputs and Outputs, Moore FSM is more likely to have a shorter critical path.

Moore FSM - Example:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity Moore is
  port(input, clk, rst : in std_logic;
       output : out std_logic);
end Moore;
```
architecture Moore of Moore is
  type state is (S0, S1, S2);
  signal moore_state : state;
begin
  moore_process : process (clk, rst)
    begin
      if (rst = '1') then
        moore_state <= S0;
      elsif (clk = '1' and clk'event) then
        case moore_state is
          when S0 =>
            if (input = '1') then
              moore_state <= S1;
            end if;
          when S1 =>
            if (input = '0') then
              moore_state <= S2;
            end if;
          when S2 =>
            if (input = '0') then
              moore_state <= S0;
            else
              moore_state <= S1;
            end if;
          end case;
      end if;
    end process;
  output <= '1' when moore_state = S2 else '0';
end Moore;

> Mealy FSM - Example:
Lab Exercise

Implement the following FSMs using VHDL.
Homework Exercises

Implement the following FSMs using VHDL.

1.

\( Y \leftarrow '0'; \)
\( Z \leftarrow A \text{ and } B; \)

2.

\( Y \leftarrow A \text{ nor } B; \)
\( Z \leftarrow '1'; \)