Toronto Metropolitan University

Department of Electrical, Computer and Biomedical Engineering

COE 328 – Digital Systems

Lab 5 - VHDL for Sequential Circuits: Implementing a Customized State Machine (2 weeks)

---

1 Objectives:

- To simulate and verify the operation of a sequential circuit.
- To design a finite state machine (FSM) that cycles through the individual digits of your student ID using the assigned state diagrams.
- To explore the FSMs with different state assignments.

2 Pre-Lab Preparation

1. You will be assigned one of the state machines described by the state diagrams shown in Figure 1.
2. Your implementation will a Moore state machine as assigned by your lab instructor. Produce a state table and state-assigned table for your customized state machine.
3. A Moor FSM is presented in Figure 2 in a general form. C_1, C_2 and C_3 are input and output combinational circuits. Design the logic equations for each of the Flip-Flop inputs shown in this figure.
4. Draw the logic diagram for your circuit.
5. Create a file *lab5.vhd* to program the Cyclone-II EP2C35F672C6 FPGA
   (Hint: Use any of the methods represented in Figures 8.29, 8.33, 8.35 -see the text book, 3rd edition).

3 Laboratory Work

1. Create the subdirectory *lab5* in your work directory, and copy the all the file *lab5.vhd* to this subdirectory.
2. The state machine must transition through all of the states starting from the state s_0 (see Figure 1). The output signals in each state are defined by the digits of a student ID: {d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9}. The first digit (d_1) is ignored. For the remaining digits, the signal assignment is as follows. If the FSM is in state s_0, then the output signal must be d_2, or s_0/d_2. Similarly, for the other states and output signals, the assignment is: s_1/d_3, s_2/d_4, s_3/d_5, s_4/d_6, s_5/d_7, s_6/d_8, s_7/d_9, as shown in Figure 1. When the FSM cycles through the states s_0 … s_7, the output signals should go through the corresponding student ID digits. The sample code given in Figure 3 represents a student ID equal to **500435429** (you will need to complete this code in compliance with your ID number).
3. Compile your designs.
4. Assign all Input/Output signals to any dedicated Input/Output pins of the Cyclone-II FPGA on the prototype board (see Pin Assignment Tables in Lab3). Re-compile your design.
   NOTE:
   - All the output LEDs are active **HIGH**. (i.e., a logic '1' will turn the LED's on).
   - All the 7-segment displays are active **LOW** (i.e., a logic '0' will turn the respective segment on).
   - The **reset** signal must be assigned to the pushbutton[0] switch on the prototype board (PIN_G26).
   - The **clk** signal must be assigned to the to the pushbutton[3] switch on the prototype board (PIN_W26).
5. Implement/program all your designs into the Cyclone II FPGA.

6. Every output signal (digit) of the Student ID (signal student_id) should be displayed on the right 7-segment display, while the present state of the FSM (signal yfsm) is displayed on the left 7-segment display.
7. Design your circuits as outlined and demonstrate results to the instructor by displaying both the states and student ID digits utilizing the 7-segment displays on the prototype board.
   NOTE: Re-use the 7-segment module from Lab 3 to display states and student ID digits.

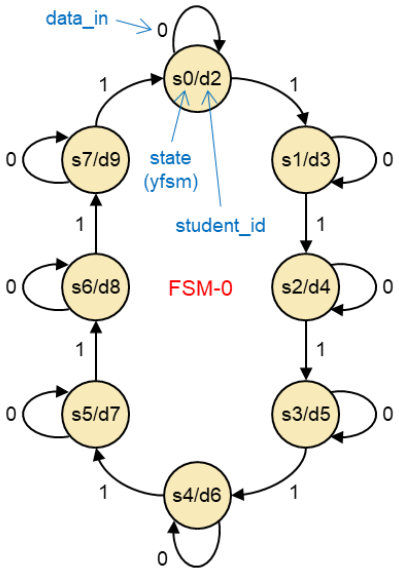

Figure 1a: State Diagram Assignment – FSM-0 (a sample)

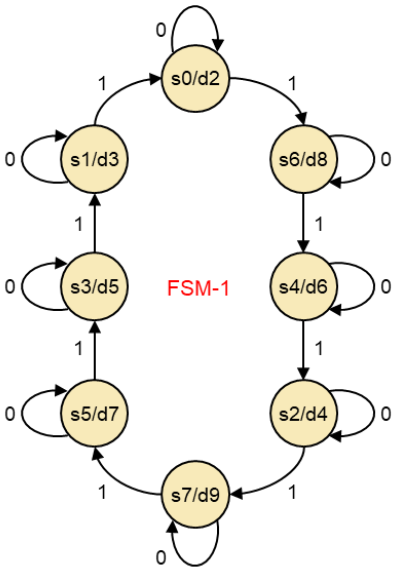

Figure 1b: FSM-1

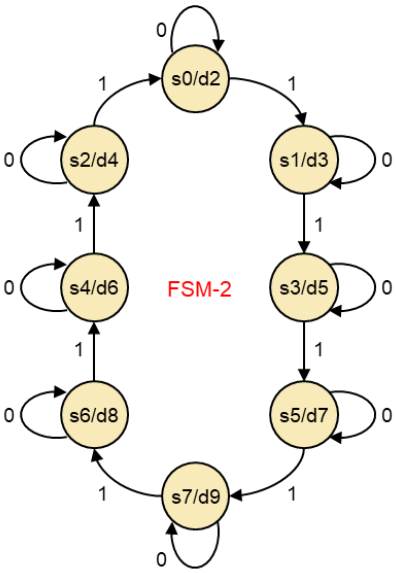

Figure 1c: FSM-2

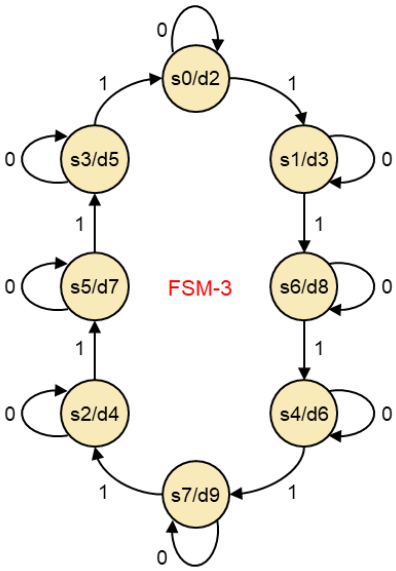

Figure 1d: FSM-3

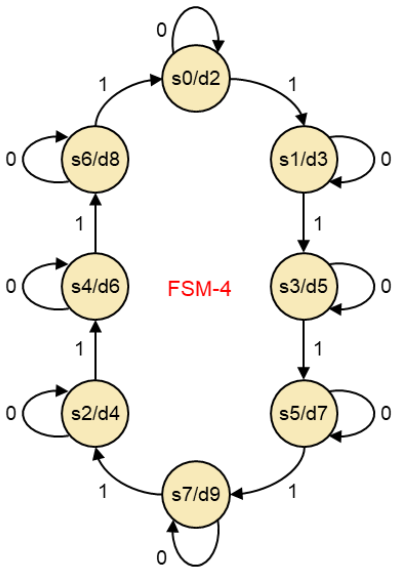

Figure 1e: FSM-4

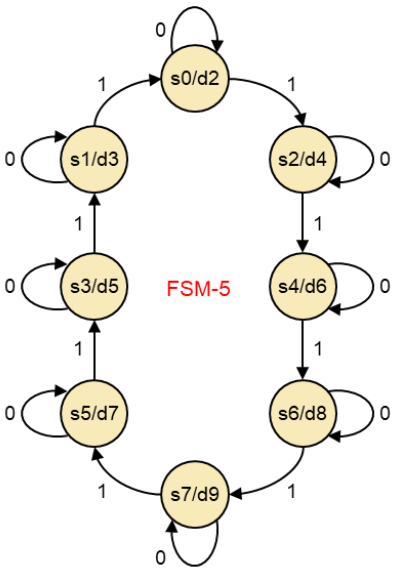

Figure 1f: FSM-5

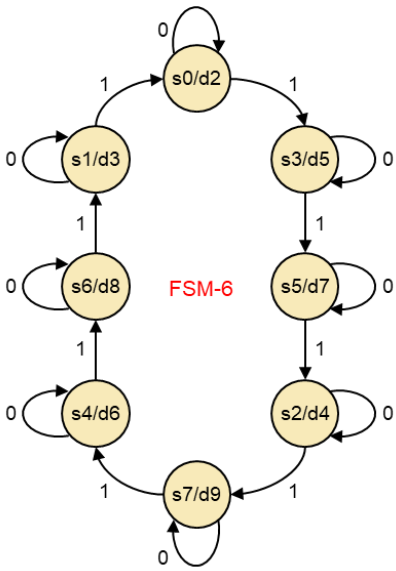

Figure 1g: FSM-6

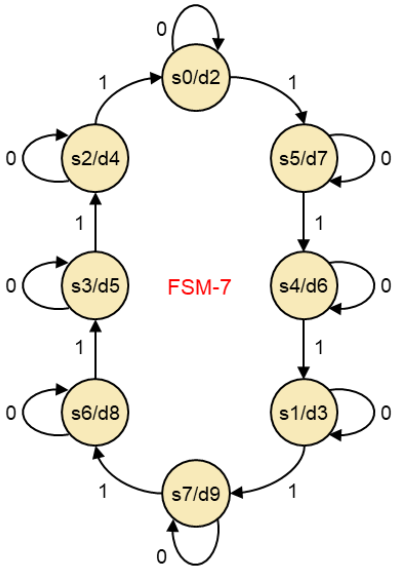

Figure 1h: FSM-7

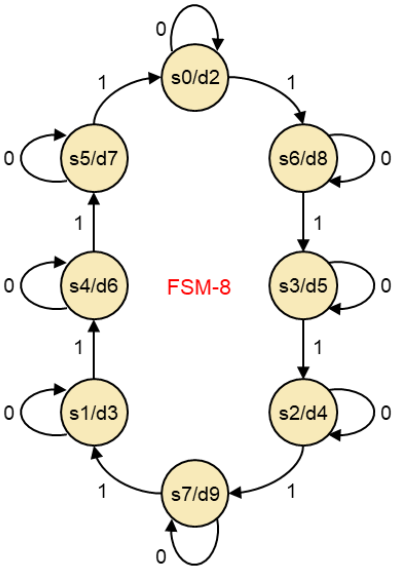

Figure 1i: FSM-8

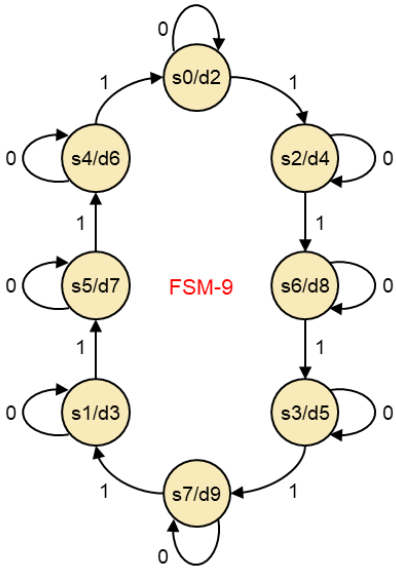

Figure 1j: FSM-9

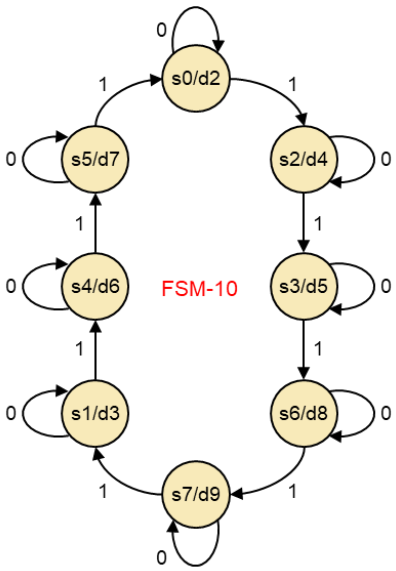

Figure 1k: FSM-10

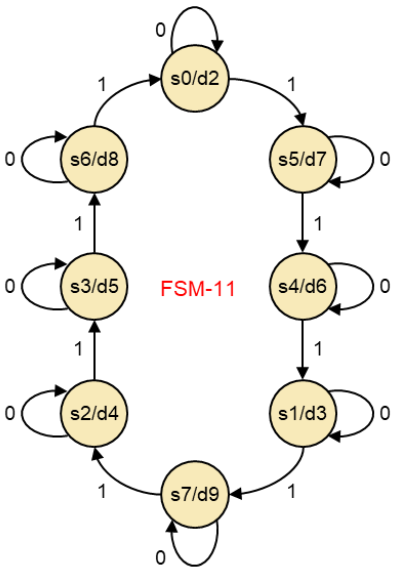

Figure 1l: FSM-11

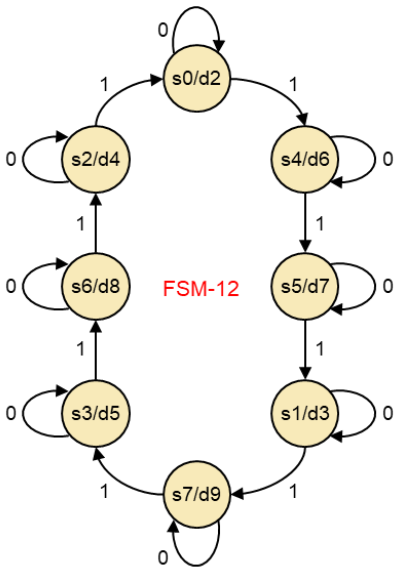

Figure 1m: FSM-12

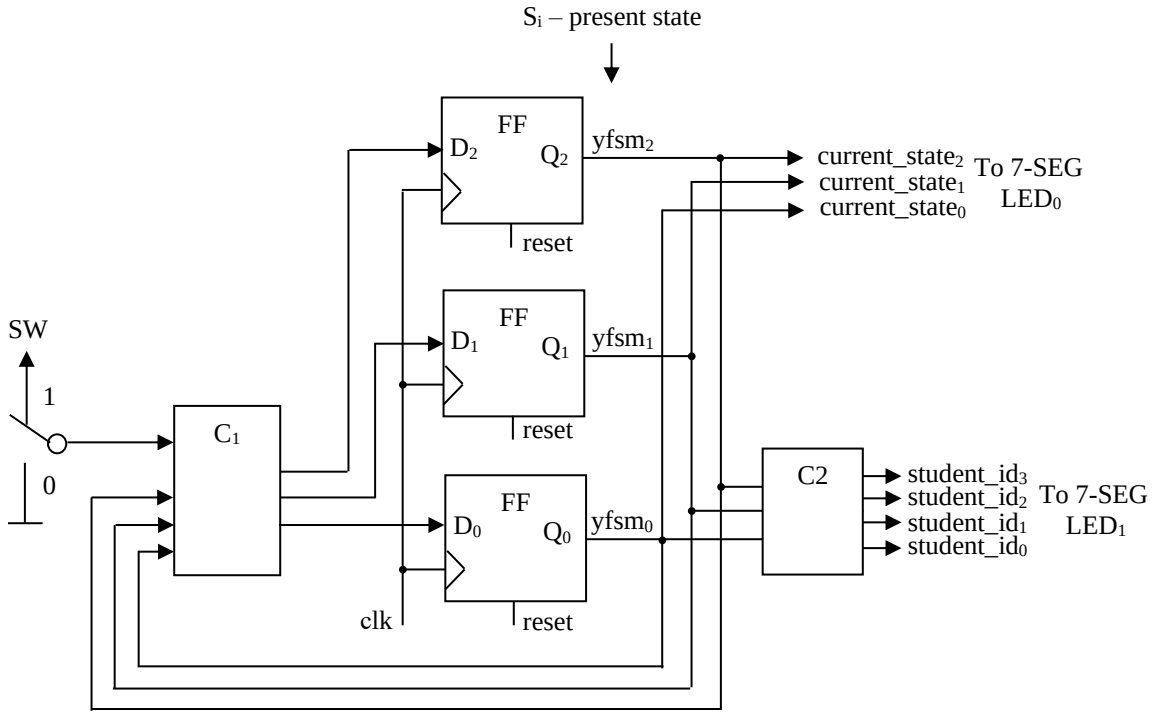

Figure 2: A general form of a Moor FSM

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

entity machine IS
        port (data_in, clk, reset      : in    std_logic ;
                              student_id : out  std_logic_vector(3 downto 0);
                           current_state : out  std_logic_vector(2 downto 0));
end simple ;

architecture fsm of machine is
     type state_type is (s0, s1, s2, s3, s4, s5, s6, s7);
     signal yfsm : state_type ;
 begin
     process ( clk, reset )
     begin
        if reset = '1' then
             yfsm <= s0 ;
        elsif (clk'EVENT AND clk = '1') then
             case yfsm is
                when s0 =>
                     if data_in = '1' then
                          yfsm <= s1 ;
                     end if ;
                when s1 =>
                     …
                when s7 =>
                     …
             end case ;
        end if ;
     end process ;
     process ( yfsm )
     begin
        case yfsm is
           when s0 => current_state <= "000" ;
                student_id <= "0000" ; -- d2
           when s1 => current_state <= "001" ;
                stedent_id <= "0000" ; -- d3
                …
           when s7 => current_state <= "111" ;
                stedent_id <= "1001" ; -- d9     St. ID
        end case ;      -- d1 d2 d3 d4 d5 d6 d7 d8 d9
     end process ;      --  5   0   0   4   3   5   4   2   9
end fsm ;           -- states:  s0  s1 s2 s3 s4 s5 s6 s7
```

Figure 3: Student ID number is 500435429