2014 Spring CSE140L

Digital Systems Laboratory

by

Dr. Choon Kim

CSE Department, UCSD
chk034@eng.ucsd.edu
Mealy vs. Moore FSM

Mealy Machine: \( y(t) = f(x(t), s(t)) \)
Moore Machine: \( y(t) = f(s(t)) \)

\[ s(t+1) = g(x(t), s(t)) \]

Mealy Machine

Moore Machine
FSM design example – Moore vs. Mealy

A circuit which removes one 1(i.e., the first 1) from every string of 1s on the input stream:

<table>
<thead>
<tr>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>..... 00000...... =&gt; ..... 00000......</td>
<td></td>
</tr>
<tr>
<td>..... 01010...... =&gt; ..... 00000......</td>
<td></td>
</tr>
<tr>
<td>..... 00110...... =&gt; ..... 00100......</td>
<td></td>
</tr>
<tr>
<td>..... 01110...... =&gt; ..... 01100......</td>
<td></td>
</tr>
<tr>
<td>..... 10111...... =&gt; ..... 10010......</td>
<td></td>
</tr>
</tbody>
</table>
FSM design example – Moore vs. Mealy

- Remove one 1 from every string of 1s on the input.
Moore FSM Verilog model: (Not necessary working solution)

module reduce (input clk, in,
    output reg out);

parameter zero = 2'b00, one1 = 2'b01, two1s = 2'b10;
reg[2:1] state, next_state;

always @(posedge clk) begin
    state = next_state;
end

always @(in, state) begin
    case (state)
        zero: begin
            out = 0;
            if (in) next_state= one1;
            else next_state= zero;
        end
        one1: begin
            out = 0;
            if (in) next_state= two1s;
            else next_state= zero;
        end
        two1s: begin
            out = 1;
            if (in) next_state= two1s;
            else next_state= zero;
        end
    endcase
end

end module
Mealy FSM Verilog model: (Not necessary working solution)

module reduce (input      clk,  in,  
                output     reg   out);

parameter      zero = 1'b0,   one = 1'b1;
reg   state,  next_state;

always @(posedge  clk)  begin
  state = next_state;
end

always  @(in,   state)  begin
  case (state)
    zero:    begin
      out = 0;
      if (in)     next_state= one;
      else       next_state= zero;
    end
    one:    begin
      out  = 0;
      if (in)    begin
        out = 1;
        next_state= one;
      end
      else      begin
        out = 0;
        next_state= zero;
      end
    end
  endcase
end
end module
LAB#3: Vending Machine Controller Design

• It has many states – Power_on_init, 0 cents, 5 cents, 10 cents, ....
• How to move from one state to other state?
• How to detect & handle dispensing?
• How to detect & handle credit-card and coin input?
• How to detect & handle error cases?
• ....
Some LAB3 works to be implemented
(Suggestions only...not necessary to follow)

- **Key point of design is...** Develop an accurate **state diagram** first!
  Then convert it into Verilog codes using the style shown in the class.

- **Reading inputs**(sw0,1,2,...,9..., etc.):
  - always(*).... or assign....

- **What states are needed?**
  - An initial state(for when power is turned on)
  - Many other states(for VM operation)

- **VM operation:**
  - two always blocks (one for state transition, the other for output & next state determination)
  - various cases handling(dispensing, credit-card, one-dollar bill input, etc.)
  - Report mode is when sw[9] = 1 **with all other sw are down!**