Exp#7

Finite State Machine Design in Verilog

COE203
Digital Logic Laboratory

Dr. Ahmad Almulhem
KFUPM
Spring 2009
Sequential Circuits

- Sequential circuits consist of both combinational logic and storage elements.
- Sequential circuits can be:
  - **Mealy-type**: outputs are a combinatorial function of both Current State signals and primary inputs.
  - **Moore-type**: outputs are a combinatorial function of Current State signals.
Generalized Verilog Mealy Model

```verilog
reg Y, D, Z;

// register (current state)
always @(posedge CLK or posedge Reset)
begin
    if (Reset) Y = 0 else Y = D;
end;

// next state logic
always @(X or Y)
begin
    D = F1(X, Y);
end;

// output logic
always @(X or Y)
begin
    Z = F2(X, Y);
end;
```

```diagram
---
```

X --- F2 --- Z
  |     |
  |     |
Y --- F1 --- D
  |     |
  |     |
CLK  --- Reset
```

Register
Generalized Verilog Moore Model

```verilog
reg Y, D, Z;

// register (current state)
always @ (posedge CLK or posedge Reset)
begin
  if (Reset) Y = 0 else Y = D;
end;

// next state logic
always @ (X or Y)
begin
  D = F1(X, Y);
end;

// output logic
always @ (Y)
begin
  Z = F2(Y);
end;
```
# 2-Way Traffic Controller

![Diagram of a 2-way traffic controller with directions and signal states.]

<table>
<thead>
<tr>
<th>Traffic</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EW only</td>
<td>EW Signal green</td>
</tr>
<tr>
<td></td>
<td>NS Signal red</td>
</tr>
<tr>
<td>NS only</td>
<td>NS Signal green</td>
</tr>
<tr>
<td></td>
<td>EW Signal red</td>
</tr>
<tr>
<td>EW &amp; NS</td>
<td>Alternate</td>
</tr>
<tr>
<td>No traffic</td>
<td>Previous state</td>
</tr>
</tbody>
</table>
Design

INPUTS

- Sensors X[1:0]
  X[0]: car coming on NS
  X[1]: car coming on EW

- Clock and Reset.

OUTPUTS

- Signals S[1:0]:
  S[0]: NS is green
  S[1]: EW is green
module Traffic_Controller (Clock, Reset, X, S);
  input[1:0] X;
  input Clock, Reset;
  output[1:0] S;
  reg currentState, nextState;
  reg[1:0] S;

  // register (Current State)
  always @( ... )
    begin
      ... 
    end

  // next state logic
  always @( ... )
    begin
      ... 
    end

  // output logic
  always @( ... )
    begin
      ... 
    end
endmodule

You need to debounce the clock!
Hint: Check Lab4