Review Problem 24

• Given the light display shown, build the FSM for a "move left" arrow traffic sign. It should animate an arrow moving left.

• Hint: Can any of the bulbs be connected to the same signal?

\[
\begin{align*}
55 &= \{A, O, P\} \\
51 &= \{C, N\} \\
54 &= \{C, H, U, Y\} \\
53 &= \{B, 55, X\} \\
52 &= \{A, D, Q, 4, 3, K\}
\end{align*}
\]
Vending Machine Example (cont.)

- State Table:

<table>
<thead>
<tr>
<th>DS</th>
<th>D</th>
<th>N</th>
<th>Open</th>
<th>PS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

\[ N_S = \overline{P_S} N + P_S \overline{N} \overline{D} \]
Vending Machine Example (cont.)

- **Implementation:**

\[
\text{open} = D + \overline{PS} \cdot N \quad \text{NS} = \overline{PS} \cdot N + PS \cdot \overline{D}
\]
FSMs in Verilog - Declarations

```
module simple (clk, reset, w, out);

input logic clk, reset, w;
output logic out;

enum { A, B, C} ps, ns; // Present state, next state
```
FSMs in Verilog - Combinational Logic

// Next State Logic
always_comb begin
    case (ps)
        A: if (w) ns = B;
        else ns = A;
        B: if (w) ns = C;
        else ns = A;
        C: if (w) ns = C;
        else ns = A;
    endcase
end

// Output Logic - could also be "always",
// or part of next-state logic.
assign out = (ps == C);
FSMs in Verilog – DFFs

// Sequential Logic (DFFs)
always_ff @(posedge clk) begin
    if (reset)
        ps <= A;
    else
        ps <= ns;
end

endmodule
Circuit Diagram of FSM
module simple_testbench();
    logic clk, reset, w, out;

    simple dut (.clk, .reset, .w, .out);

    // Set up the clock.
    parameter CLOCK_PERIOD=100;

    initial begin
        clk <= 0;
        forever #(CLOCK_PERIOD/2) clk <= ~clk;
    end
FSM Testbench (cont.)

// Design inputs. Each line is a clock cycle.

// ONLY USE THIS FORM for testbenches!!!

initial begin

@ (posedge clk); 1

reset <= 1; @ (posedge clk); 2
reset <= 0; w <= 0; @ (posedge clk); 3
@ (posedge clk); 4
@ (posedge clk); 5
@ (posedge clk); 6

w <= 1; @ (posedge clk); 7
w <= 0; @ (posedge clk); 8
w <= 1; @ (posedge clk); 9
@ (posedge clk); 10
@ (posedge clk); 11
@ (posedge clk); 12

w <= 0; @ (posedge clk); 13
@ (posedge clk); 14

$stop; // End the simulation.

end
endmodule