Lecture 16

Alternative 1

From the example from the previous lecture, we can also choose state codes for the different states using
A = 0001, B = 0010, C = 0100, D = 1000
Then

Initial State	w = 0	w = 1	z
0001	0001	0010	0
0010	0100	0010	0
0100	0001	1000	0
1000	0100	0010	1

By inspection, the only way to get to state C is from B or D, i.e.

$$Y_3 = y_2\overline{w} + y_4\overline{w} = \overline{w}(y_2 + y_4)$$

Similarly,

$$Y_4 = y_3 w$$

$$Y_2 = w(y_1 + y_2 + y_4)$$

$$Y_1 = \overline{w}(y_1 + y_3)$$

$$z = y_4$$

Alternative 2

- Make a state diagram in which each state represents the last three values of w.

Previous State	w = 0	w = 1	z
000	000	100	0
001	000	100	0
010	001	101	0
011	001	101	0
100	010	110	0
101	010	110	1
110	011	111	0
111	011	111	0

Correspondingly, we define y_i and Y_i, so that

$$Y_3 = w$$

$$Y_2 = y_3$$

$$Y_1 = y_2$$

$$z = y_3\overline{y_2}y_1$$

This can be constructed as a series of flip flops with $D_1 = w, Q_1 = D_2 = y_3, Q_2 = D_3 = y_2, Q_3 = y_1$, where the output z is constructed in an AND gate as defined above.

Verilog

- 3 sections: flip flops, state tables, output

```
module FSM (input logic w, clock, resetn, output logic z);
    typedef enum logic [1:0] (A, B, C, D) statetype;
    statetype ps, ns;

    always_ff @(posedge clock, negedge resetn)
        if(!resetn)
            ps <= A;
        else
            ps <= ns;

    always_comb
        case(ps)
            A: if(w) ns = B;
                else ns = A;
            B: if (w) ns = B;
                else ns = C;
            C: if (w) ns = D;
                else ns = A;
            D: if (w) ns = B;
                else ns = C;
        endcase

    assign z = (ps==D);
endmodule
```