Lecture 17

Verilog for shift register FSM example

```
module FSM_shift(input logic w, clock, resetn, output logic z);
    logic [3:1] y;
    always_ff @(posedge clock, negedge resetn)
        begin
            if(!resetn)
                y <= 3'b000;
            else
                begin
                    // non-blocking assignments happen concurrently
                    y[3] <= w;
                    y[2] <= y[3];
                    y[1] <= y[2];
                end
        end
    assign z = y[3]&~y[2]&y[1];
endmodule
```

Traffic light controller

- On reset, light A is green, and light B is red
- Every clock cycle, if there is traffic on A, light A stays green, else it transitions to yellow then red. Light B stays red then transitions to green.
- Same for B
- State Table:

Present State	00	01	10	11	L_A	L_B
s_0	s_1	s_1	s_0	s_0	G	R
s_1	s_2	s_2	s_2	s_2	Y	R
s_2	s_3	s_2	s_3	s_2	R	G
s_3	s_0	s_0	s_0	s_0	R	Y

Using the conventional binary encodings,

y_2y_1	00	01	10	11
00	01	01	00	00
01	10	10	10	10
10	11	10	11	10
11	00	00	00	00

A K-map gives

$$Y_2 = y_2 \bigoplus y_1$$

$$Y_1 = \bar{y_2}\bar{y_1}\bar{T_A} + y_2\bar{y_1}\bar{T_B}$$

Output Encoding G: 00, Y: 01, R: 10

y_2y_1	L_A	L_B
00	00	10
01	01	10
10	10	00
11	10	01

$$L_A[1] = y_2$$
$$L_A[0] = \overline{y_2}y_1$$
$$L_B[1] = \overline{y_2}$$
$$L_B[0] = y_2y_1$$

Deriving a state diagram from a circuit

$$Y_1 = y_2w + \overline{y_1}w$$
$$Y_2 = y_2w + y_1w$$
$$z = y_2y_1$$

Then

y_2y_1	w = 0	w = 1	z
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

Labelling the states as A, B, C, D gives us our state diagram, showing that this FSM outputs z = 1 iff there are 3 consecutive w = 1 inputs.