Lecture 14

Asynchronous Reset

- Case 1: if `clk` $= 0$, `resetn` $= 0$, gives $\overline{Q} = 1$, $Q = 0$
- Case 2: if `clk` $= 1$, `resetn` $= 0$, gives $\overline{Q} = 1$, $Q = 0$
- Verilog Code

```
module D_FF(input logic D, clock, resetn, output logic Q);
    always_ff @(posedge clock, negedge resetn)
    //resetn appears in sensitivity list because it can directly affect Q
    if (resetn == 0)
        Q <= 1'b0;
    else
        Q <= D;
endmodule
```

- Multi-bit register with reset

```
module reg8(input logic [7:0] D, resetn, clock, output logic [7:0] Q);
    always_ff @(posedge clock)
    begin
        if(!resetn)
            Q <= 8'b0;
        else
            Q <= D;
    end
endmodule
```

- Enable input: Determines whether data is loaded on clock edge, passes new value D to flip flop if true, recycles old state if false

Counters

- Circuits that count up or down

Clock Cycle	$Q_2Q_1Q_0$
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111
8	000

- Q_0 toggles every cycle

$$Q_0(t+1) = Q_0(t) \text{ XOR } 1$$

- Q_1 toggles when $Q_0 = 1$

$$Q_1(t+1) = Q_1(t) \text{ XOR } Q_0(t)$$

- Q_2 toggles when Q_0 and Q_1 are 1

$$Q_2(t+1) = Q_2(t) \text{ XOR } (Q_0(t)Q_1(t))$$

- To pause the counter, one can use an Enable input, so if E = 0, the flip flops cannot change

```
module upcount(input logic [3:0] R, resetn, clock, E, L, output logic [3:0] Q);
    always_ff @(posedge clock, negedge resetn)
    begin
        if (!resetn)
            Q <= 4'b0;
        elseif (L)
            Q <= R
        elseif (E)
            Q <= Q + 1;
    end
endmodule
```