Lecture 8, Sep 26, 2022

Hierarchical Verilog

- Top-level module that is composed of simpler modules
- Makes code easier to read and reproduce
- Example: using full adder module to build adder for 2 3-bit inputs

```
module adder3(input logic [2:0] A, B,
              input logic cin,
              output logic [2:0] S,
              output logic cout);
    logic c1, c2;
    // Add first bits with carry in
    fa u0(A[0], B[0], cin, S[0], c1);
    // Add second bit and third bit
    fa u1(A[1], B[1], c1, S[1], c2);
    fa u2(A[2], B[2], c2, S[2], cout);
endmodule
```

- 3 instances of the full adder are needed, each needs a unique name

Example

- Example 2: a circuit that displays a sum R on a 7-segment display where R is either $a+b$ or $c+d$
  - Truth table for a 7-segment decoder (note a segment lights up when it is 0):

x_1 x_0	h_0 h_1 h_2 h_3 h_4 h_5 h_6 h_7
00	0000001
01	1001111
10	0010010
11	0000110

- Logic expressions for each column:
  - $h_0 = \bar{x}_1 x_0$
  - $h_1 = 0$
  - $h_2 = x_1 \bar{x}_0$
  - $h_3 = \bar{x}_1 x_0$
  - $h_4 = x_0$
  - $h_5 = x_1 + x_0$
  - $h_6 = \bar{x}_1$
- Verilog:

```
module seg7(input logic [1:0] x,
            output logic [6:0] h);
    assign h[0] = ~x[1] & x[0];
    // 1 represents number of bits, b indicates binary
    // d for decimal, h for hex
    assign h[1] = 1'b0;
    assign h[2] = x[1] & ~x[0];
    assign h[3] = ~x[1] & x[0];
    assign h[4] = x[0];
    assign h[5] = x[1] | x[0];
    assign h[6] = ~x[1];
endmodule
```

- Need a 2-bit 2-to-1 mux switching between a, b or c, d, with the output fed to a half adder, and then to a 7-segment decoder

```
module hier(input logic [4:0] SW,
            output logic [6:0] HEX0);
    logic [1:0] F, R;
    mux2to1_2bit(SW[1:0], SW[3:2], SW[4], F);
    ha u2(F[1], F[0], R);
    seg7 u3(R, HEX0);
endmodule
```