Lecture 8

Full Adder

c_i	a_i	b_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

where c_i stands for the carry term.
Here, c_{i+1} is 1 if at least two of the inputs is 1; this is called a **majority** function.
s_i is the output of a 3-input XOR function, or c_i XOR a_i XOR b_i. It can be called an odd function, as it is 1 iff odd number of inputs are 1.

```
module FA(input logic a, b, cin, output logic s, cout);
    assign s = a^b^cin;
    assign cout = (a&b)|(cin&a)|(cin&b);
endmodule
```

Hierarchical Verilog Code

- A module composed of simpler modules
- Makes code easier to read and reuse

Example 1

```
module adder3(input logic [2:0] A, B, input logic cin, output logic [2:0] S, output logic co
    logic C1,C2; //internal signals
    FA u0(A[0],B[0],Cin,S[1],C2);
    FA u1(A[1],b{1],C1,S[1],C2);
    FA u2(A[2],B[2],C2,S[2],cout);
endmodule
```

Example 2

- Display a sum R on a 7-seg display where R is either $a + b$ or $c + d$
- 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
0	0	0	0	0	0	0	0	1

x_1	x_0	h_0	h_1	h_2	h_3	h_4	h_5	h_6
0	1	1	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0
1	1	0	0	0	0	1	1	0

Then,

$$
\begin{aligned}
h_0 &= \overline{x_1}x_0 \\
h_1 &= 0 \\
h_2 &= x_1\overline{x_0} \\
h_3 &= h_0 \\
h_4 &= x_0 \\
h_5 &= x_1 + x_0 \\
h_6 &= \overline{x_1}
\end{aligned}
$$

```
module seg7(input logic[1:0] X, output logic [6:0] H);
    assign H[0] = ~X[1] & X[0];
    assign H[1] = 1'b0;
    //and so on
endmodule
```

We also need a multiplexer (to decide to add which numbers) and an adder (to add the numbers). The sum is then input into `seg7`, which is connected to the display.

```
module hier(input logic [4:0]SW, output logic [6:0]HEX0);
    logic [1:0] F, 2; //internal signals
    //instantiate subcircuits
    mux2to1-2bit u1(SW[1:0],SW[3:2],SW[4],F);
    ha u2(F[1], F[0], R);
    seg7 u3(R, HEX0);
endmodule
```