Combination Logic

January 5, 2018 12:21

Lab 1 Briefing

Purpose: re-acquainting with Quartus II design tools
Goal: Baccarat Game

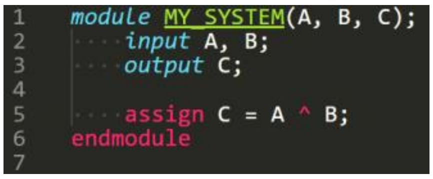
Combination Blocks

- Based on combination logic (output is only dependant on the current inputs)
- Example such as adding
- No history
- Consists of Boolean gates but **not flip-flops** (which is used for memory)

The combination blocks that will be used are for:

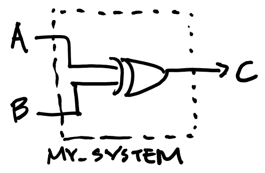
1. 7 segment LED display
2. Computation of score of a hand

Example of a basic combinational gate in Verilog:

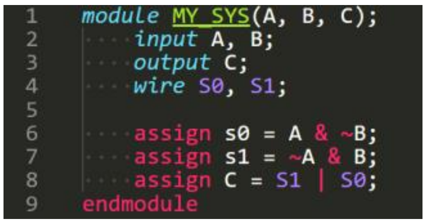


```
module MY_SYSTEM(A, B, C);
    input A, B;
    output C;

    assign C = A ^ B;
endmodule
```



Use **wire** to connect signals, consider this example:



```
module MY_SYS(A, B, C);
    input A, B;
    output C;
    wire S0, S1;

    assign s0 = A & ~B;
    assign s1 = ~A & B;
    assign C = S1 | S0;
endmodule
```

Recipe to create combinational components

1. Determine Boolean equation for each output
2. Write boolean equation as concurrent signal/wire assignments

Output change based on changes in the inputs

```
always @(<sensitivity list>)
begin
    <sequential statements>
end
```

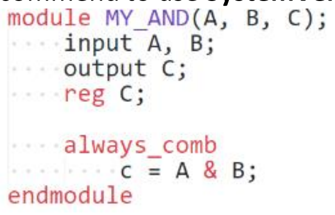
A process is created this way and assignments of signals can be performed. Here's an example:

```
module MY_AND(A, B, C);
    input A, B;
    output C;
    reg C;

    always @(A or B)
        c = A & B;
endmodule
```

In this case, events of A and B should cause C to be re-evaluated. Since we are essentially "reading" A and B, they are in the *sensitivity list*

It is recommend to use **SystemVerilog**, so we use always_comb instead:



```
module MY_AND(A, B, C);
    input A, B;
    output C;
    reg C;

    always_comb
        c = A & B;
endmodule
```