Making Combinatorial Logic - if Statement

- if is used to create combinatorial logic and mixed combinatorial sequential logic (i.e., counters).
- if is used within always_comb, always_ff, or initial statements.
- Use blocking assignment with always_comb, non-blocking assignment with always_ff
- If more than one statement is needed for if or else, you must bracket them with begin and end.
- Expression true when non-zero, false when X, Z or zero.

```verilog
module mux2_1 (  
    input    sel,  
    input    din_0, din_1,  
    output   reg d_out  
);  
    always_comb  
        if (sel)  
            d_out = din_1;  
        else  
            d_out = din_0;  
    endmodule
```
Poorly nested IF statements can produce slow logic
Some paths are far slower than others.

module serial_mux (  
    input    a,b,c,d,x,y,z,  
    output logic d_out  
);  

always_comb  
    if (z)  
        d_out = a;  
    else if (y)  
        d_out = b;  
    else if (x)  
        d_out = c;  
    else  
        d_out = d;  
endmodule
if without else gives you a latch!

module mux3_1_latchy (  
    input [1:0] sel,  
    input [3:0] din_a, din_b, din_c,  
    output logic [3:0] d_out  
);  
always_comb  
  if (sel==2’b00)  
    d_out = din_a; else  
  if (sel==2’b01)  
    d_out = din_b; else  
  if (sel==2’b10)  
    d_out = din_c;  
endmodule

Synthesis assumption: hold last value if other conditions fail
Making Combinatorial Logic - if Statement

IF without else... compiles without error!

Model Technology ModelSim SE-64 vlog 10.5 Compiler 2016.02 Feb 12 2016
vlog mux3_1_latchy.sv
-- Compiling module mux3_1_latchy
Errors: 0, Warnings: 0

But in synthesis, we get an error

Inferred memory devices: mux3_1_latchy line 7 in file mux3_1_latchy.sv.
===========================================================================
| Register Name | Type  | Width | Bus | MB | AR | AS | SR | SS | ST |
---------------------------------------------------------------------------
| d_out_reg     | Latch | 4    | Y   | N  | N  | N  | -  | -  | -  |
===========================================================================
Warning: mux3_1_latchy.sv:7: Netlist for always_comb block contains a latch. (ELAB-974)
Making Combinatorial Logic - if Statement

System Verilog if...else Modifiers

- SystemVerilog introduced two if...else modifiers
  - priority
  - unique
- Both reduce ambiguities and help catch design errors.
- Both are assertions (simulation error reporting mechanisms)
- Both require that at least one if branch condition be taken.
- Supported in Synopsys Design Compiler and ModelSim.
- Not supported in Altera Quartus.
Making Combinatorial Logic - if Statement

- unique...
  - signals designer intent that order of conditions is not important
  - informs synthesis that the conditions should be mutually exclusive. Synthesis should check to see if they are mutually exclusive, and if so, optimize out any inferred priority.
  - informs simulation to do run-time checks and issue a warning if more than one condition is true.
  - informs simulation to issue a warning if no branch condition is true.
Making Combinatorial Logic - if Statement

- priority...
  - signals designer intent that order of conditions must be maintained.
  - informs simulation it is OK if more than one condition is true.
  - informs simulation to issue a warning if no branch condition is true.
We usually see priority and unique used with if statements that have a missing final else statement that would cover all possible statements.

Placing a concluding else statement would remove the checking for a match since else covers everything else.

The use of priority and unique will signal designer intent and also prevent unwanted latches.

module if_w_latch(
input [2:0] sel,
input a,b,c,
output logic mux_out
);

always_comb
    if (sel == 3’b001) mux_out = a;
    else if (sel == 3’b010) mux_out = b;
    else if (sel == 3’b100) mux_out = c;
endmodule
Making Combinatorial Logic - if Statement

- Using either priority or unique as the situation dictates, removes the latch.

module if_wo_latch(  
    input [2:0] sel,  
    input a,b,c,  
    output logic mux_out  
);

    always_comb  
        unique if (sel == 3’b001) mux_out = a;  
        else if (sel == 3’b010) mux_out = b;  
        else if (sel == 3’b100) mux_out = c;

endmodule