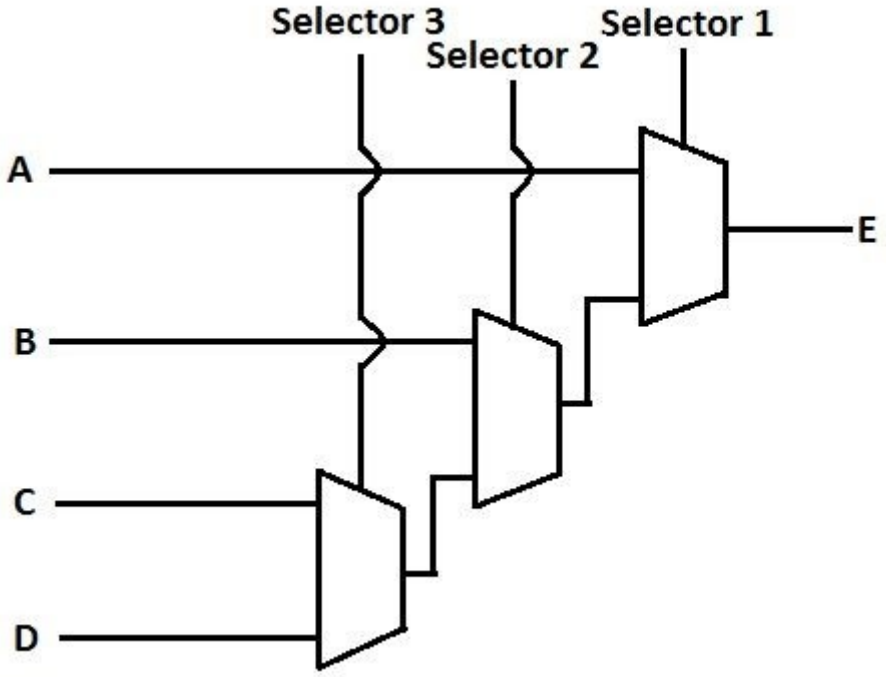
Lesson 11

Combinational Logic (part 2)

Mohd Saufy Rohmad
Technical Trainer and Consultant

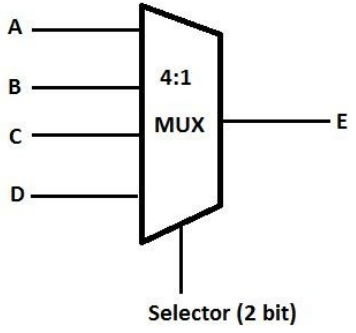
if-else vs case

- In the last hour we demonstrated how to design a 2:1 multiplexer with if-else and case.
- So one might think that logic constructed with if-else and case will be same always.
- But its not.
- Whenever we are going for more than 2 inputs in a multiplexer, then if-else and case creates different type of logic.
- Take a look at the following code,



if-else vs case

- To create a 4:1 mux, we need to use the following VHDL with a case statement,




```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity mux_case is
port (
 A,B,C,D: in std_logic;
 Selector: in std_logic_vector(1 downto 0);
 E : out std_logic );
end mux_case;

architecture mux_case_arch of mux_case is

begin

process(A,B,C,D,Selector)
begin
case Selector is
when "00" => E <= A;
when "01" => E <= B;
when "10" => E <= C;
when others => E <= D;
end case;
end process;
end mux_case_arch;
```

Latch Inference

- We have to be very careful while writing the if-else and case statements.
- If they are not written properly, a latch can be created at the output.
- This latches can mess up the whole timing result.
- Therefore, the circuit might give wrong results at the output.

Sensitivity List

- The sensitivity list includes all the signals that works as an input in a combinational process.
- The signals of the sensitivity list are included in the parentheses next to the process keyword in VHDL.

  process(sensitivity list)
- In the example given above, A,B,C,D and Selector are all input signals of the 4:1 multiplexer. Thats why all of them should be in the sensitivity list.

  process(A,B,C,D,Selector)

- Include all the input signals in the process sensitivity list for combinational logic

- Failure to include all the input signals would result in a unwanted behavior in the simulation. This is one of the most common mistakes of VHDL.

If-Else

- This if-else is wrongly written,

```
process(state)

begin
 if state = "001" then
        cs <= "01";
  elsif state = "010" then
        cs <= "10";
   elsif state = "100" then
        cs <= "11";
 end if;
end process;
```

If / else

- Because, it did not specify each possible state. This will create output latches. The correctly written case statement would be,

```
process(state)
begin
 if state = "001" then
        cs <= "01";
  elsif state = "010" then
        cs <= "10";
   elsif state = "100" then
        cs <= "11";
else
       cs <= "00";
 end if;
end process;
```

Case

- This case statement is wrongly written,

```
process(state)
begin
 case(state) is
when "001" =>
        cs(0) <= '1';
when "010" =>
        cs(1) <= '1';
when "100" =>
        cs(1 downto 0) <= "11";
when others =>
        cs(1 downto 0) <= "00";
 end case;
end process;
```

Case

- Because, it did not specify all possible outputs for each state.
- This will create output latches. The correctly written case statement would be:

```
process(state)
begin
 cs <= "00";
 case(state) is
when "001" =>
...
```

Thank You