ECE 281 | Lesson 20 | Jeff Falkinburg

Overview:

- VHDL
  - Sequential Logic
- HW#3 Due Lesson 22
- Do midterm feedback

Entity - Box w/ Ports I/O

Architecture - Describes what is inside the box
- Two types of ~~Arch~~ Arch description
  - Behavioral
  - Structural - Described

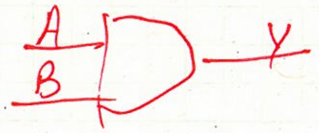
Process - "box" that listens for signals (wires)
- when signal changes, process runs
- Can be scary!
  - adds memory
  - harder to visualize circuit
- If you want a Flip Flop you need a process.

generic:

```
process (sensitivity list)
    statements
end process;
```

specific:

```
process (A, B)
  Y <= A and B;
end process;
```

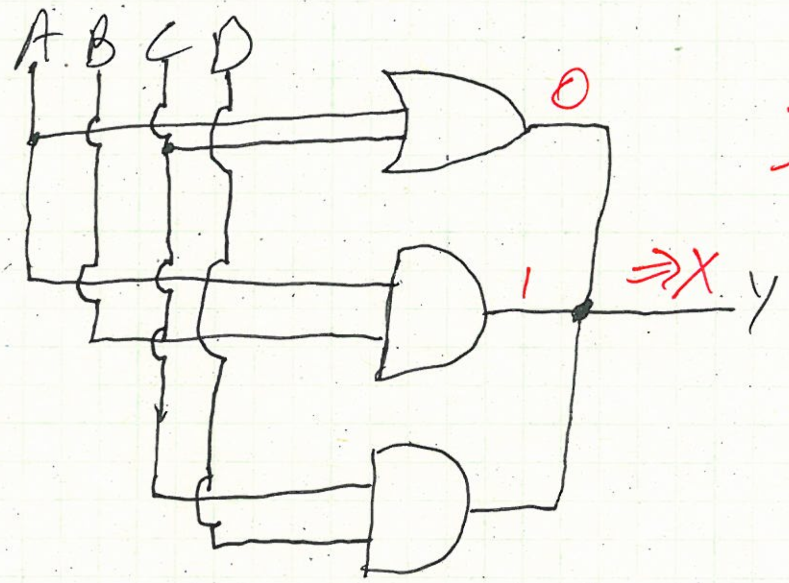


not in Process:

```
y <= A or C;
y <= A and B;
y <= C and D;
```

If statements are not in a process; which gets assigned to y?

-- All of them

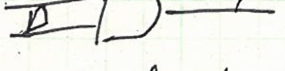


Is this Bad?

-- yes

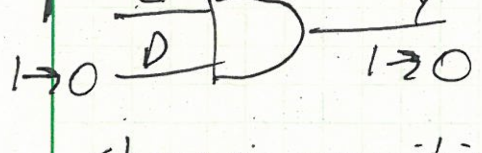
-- gives us potential for 'X' illegal values

```
process (A, B, C, D)
  y <= A or C;
  y <= A and B;
  y <= C and D;
end process;    -- what outputs?
```

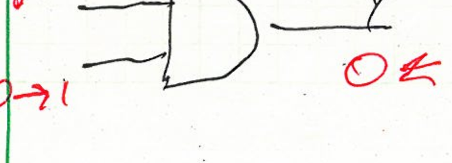


schedule is over written!!

Change sensitivity list to (C, D)



Change sensitivity list to (C)



Implicitly creates memory due to sensitivity list since it remembers what it was

Create a D-F/F in VHDL:

```
entity Flop is
   port (clk, D : in  std_logic;
              Q : out std_logic);
end;
architecture synth of flop is
  begin
     process (clk)
        if rising_edge (clk) then
           Q <= D;
        end if;
     end process;
  end synth;
```

could be named Behavioral, structural, struct, or synth

Build stuff (otherwise, behavior)

we only want it evaluated on rising edge of clk.

only runs when clk changes

only when rising edge 0 → 1

How do we change process to implement a D-Latch?

```
if clk = '1'
   Q <= D;
end if;
```

DO NOT USE:

clk'event and clk = '1'

Do use instead:

rising_edge (clk)

why? - we don't want to add logic gates to a clock Line, but if it is a control line then it is ok.

Variables: - only exist in process.

→ assigned values based on Blocking statement

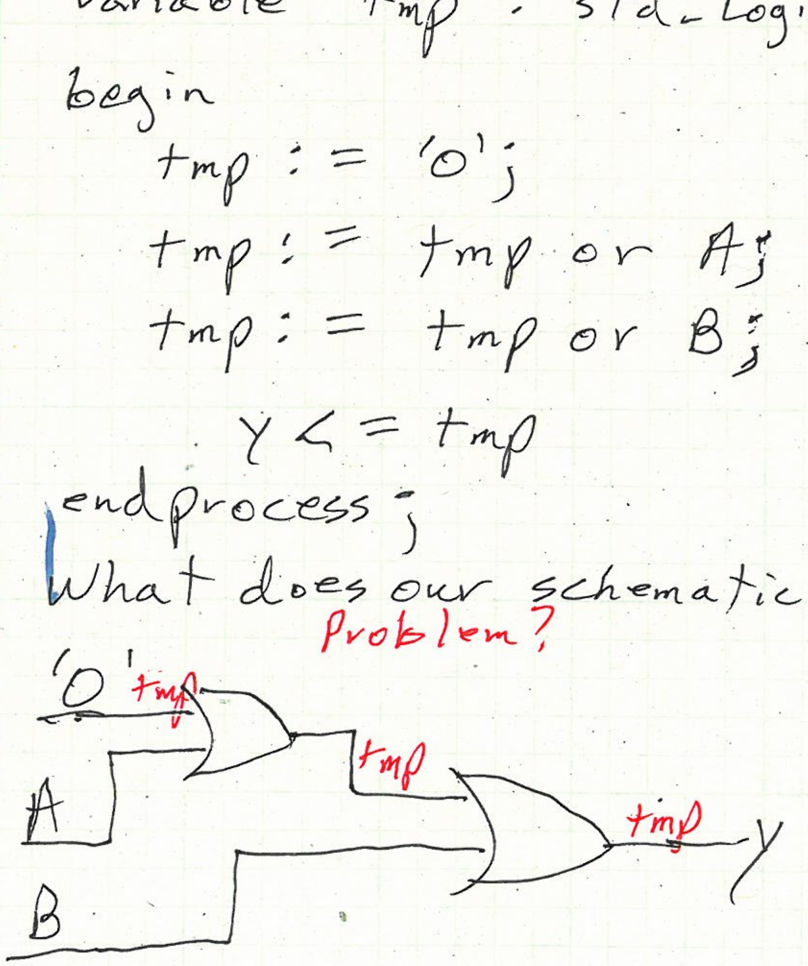
:= instead of <=

- values assigned immediately
  - No scheduling
- Blocking statement:
  - evaluated in order they appear
- non Blocking statement:
  - evaluated concurrently
  - Signals updated Later

```
process (A, B)
 variable tmp : std_Logic;
 begin
    tmp := '0';
    tmp := tmp or A;
    tmp := tmp or B;
      Y <= tmp
 endprocess;
```

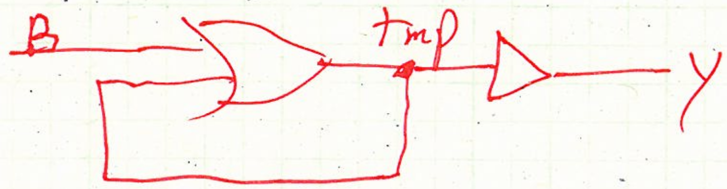
What does our schematic Look Like?

Problem?



```
Process (AB)          eliminate variable
 begin
   tmp <= '0';
   tmp <= tmp or A;
   tmp <= tmp or B;
   y <= tmp;
 end process;
```

Poor example of VHDL assigning a signal multiple times.



Rules for Process:

1) think before using variables
2) avoid "innovative" use of language constructs
* 3) avoid overwriting a signal
** 4) Only useprocesses for sequential circuits

* Only for ECE 281 outside of class use sparingly

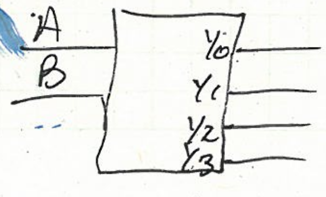
4.13 Write ~~V~~HDL module for 2:4 Decoder

4.15 Write ~~V~~HDL module to implement:

a) $Y = AC + \bar{A}\bar{B}C$

b) $Y = \bar{A}\bar{B} + \bar{A}B\bar{C} + \overline{(A+\bar{C})}$

Decoder 2:4


A
B
Y0
Y1
Y2
Y3