DAT093
Lab Feedback and Guidance 1

Mehrzad Nejat

Department of Computer Science & Engineering
Chalmers University of Technology

Sep. 2017

Organization is very important !

- Spacing and indentation

  Vertical and Horizontal

- Comments

  Use it to clarify the purpose of each section (even for yourself!)

- Naming

  Use meaningful names

  Use ‘_’

  Use different naming format for different types e.g. variables

```
If C1 then
Do something
else if C2
If C3 then
For i in range I
If C4
Do something
Else Do something else
End if
End loop
Else Do something else
End if
End if
End if
```

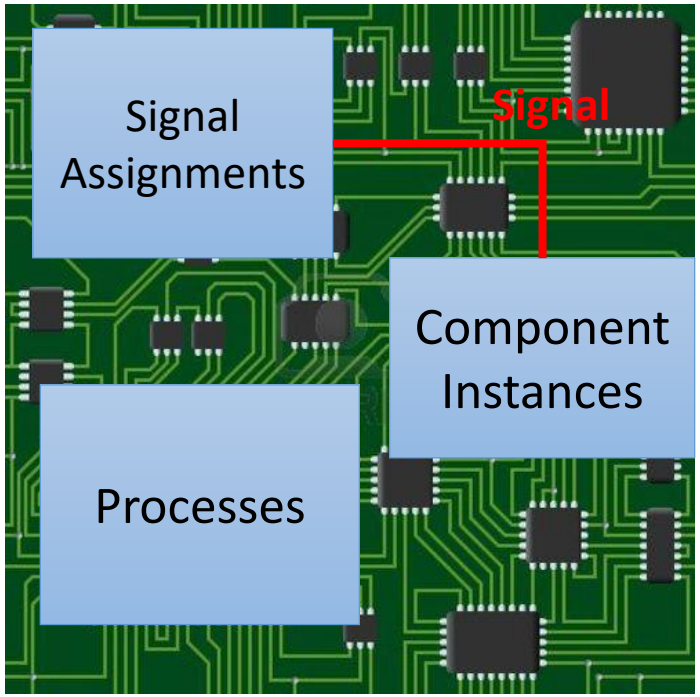

```
If Condition1 then  -- When Condition 1
   Do something
else if Condition2    -- When Condition 2
   If Condition3 then
                -- Loop over the range …
      For i in range loop
         If Condtioin4
            Do something
         Else
            Do something else
         End if
      End loop

   Else
      Do something else
   End if
End if
End if
```

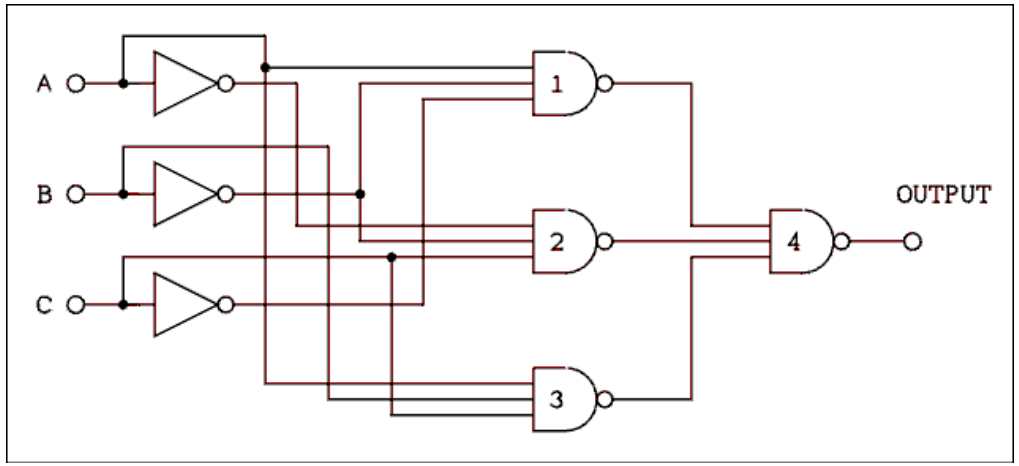
It is very difficult to review and debug a disorganized code

Concurrency in VHDL

- Hardware Description Language (HDL) is basically different from a programming language
- Realize that you are describing a hardware in VHDL and all the components are working concurrently
- Signals are wires that connect different components and carry values

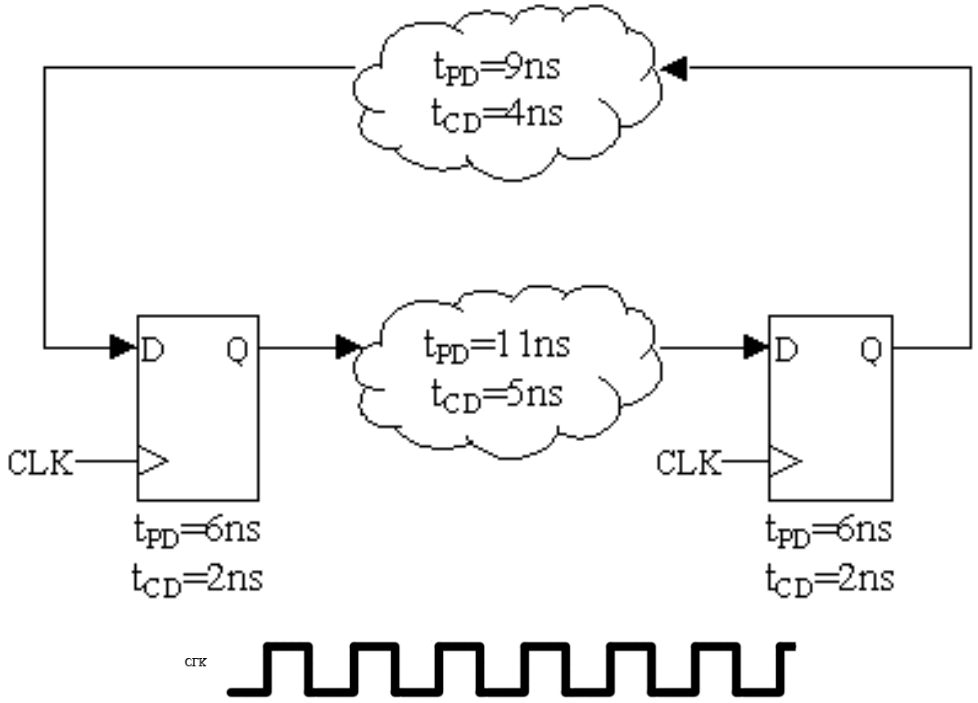


Combinational vs Sequential



Any Input set *almost* instantly result in an output

- There is a sequence and timing in the production of the outputs
- The synchronization is done with the CLK signal
- Flip-Flops are used for keeping the values



Combinational vs Sequential

- Step1: Decide whether you are describing a combinational or a sequential component.

  - Step2 – Combinational: Avoid using process statement if you can. Instead use conditional signal assignment (*s <= '0' when …. Else …*).

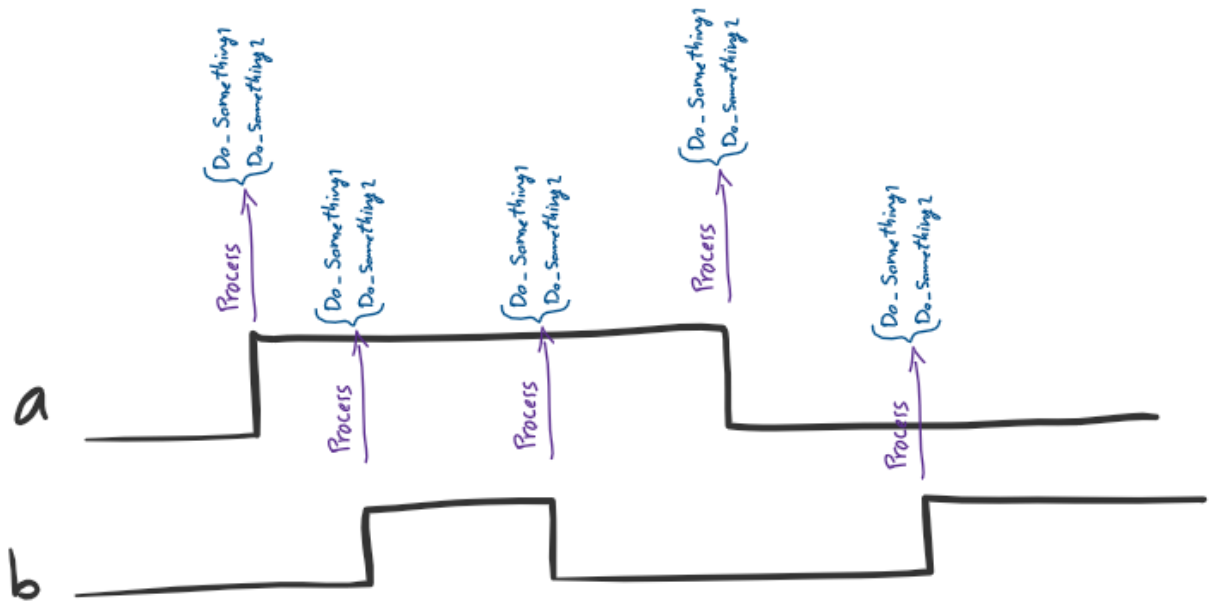
    Remember: always cover all the possible conditions (*have an 'else' at the end*).

  - Step2 – Sequential: decide if you have asynchronous or synchronous reset and whether it is active high or low. Use the corresponding process template.

Process

It is very important to understand the timing behavior of a process.

```
Process (a,b)
Begin
   Do_something1
   Do_something2
End
```



Process

A process is like a component with inputs and outputs

Inputs: What ever is read in the process

outputs: What ever is written to in the process

```
Process (a)
Begin

   if c = '1' then
         y <= x;
   else
         y <= z;
   end if;

End process;
```

Process

Combinational:

If you want to describe a combinational component with “process”, you must follow these rules:

1. Include all the inputs in the sensitivity list
   Any change in any input should wake the process

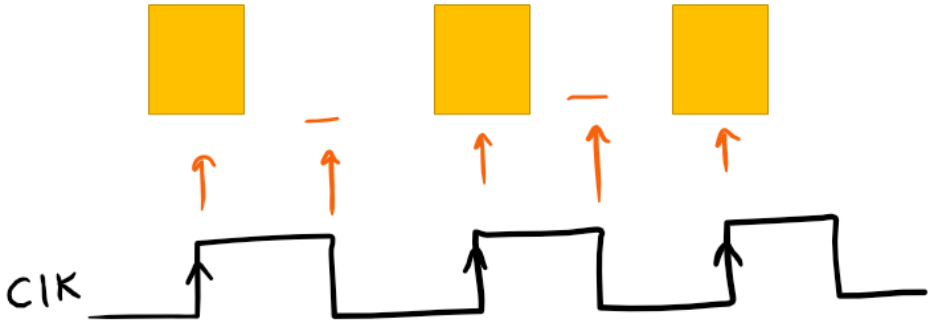
2. Make sure that you assign a value to each output in any possible condition

Process

Sequential:

In a clocked system. You must make sure that everything is synchronized to the rising edge of the clock (*except asynchronous reset if you have it*). Nothing should happen in any other time.

```
Process (clk)
Begin
   if rising_edge(clk) then
      if rst='1' then
         .....
      else
         .....
         .....
      end if;
   end if;
End Process
```




```
-- process with asynchronous reset
asynch_reset_proc: process (clk, rst_a)
begin
  if rst_a = '1' then
    ...
  elsif rising_edge(clk) then
    ...
  end if;
end process asynch_reset_proc;
```

Process

Sequential – Signal Delay:

```
Signal S0,S1,S2: std_logic;

Process (clk)
  variable V0,V1,V2: std_logic;
Begin
   if rising_edge(clk) then

       S1<=S0;
       S2<=S1;

       V1:=V0;
       V2:=V1;

   end if;
End Process
```