Process Construct

Last updated 2/1/24

Process Construct

- The process construct allows portions of the VHDL code to be executed only under certain conditions
  - The code is only executed when a signal in its sensitivity list has changed
  - See example
- The process construct allows the more flexible if/else and case statements to be used
- The process construct ONLY updates sequential signals at the end of the process
  - See example

Process Construct

- Structure

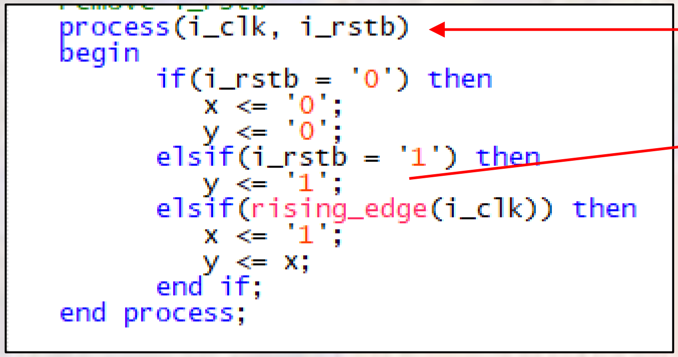
Optional label
Cannot be a duplicate of any other process label or signal name

```
label process (sensitivity list)
  begin
    hdl code
  end process;
```

Sensitivity list
The process block is not evaluated unless a signal in the sensitivity list has changed

Process Construct

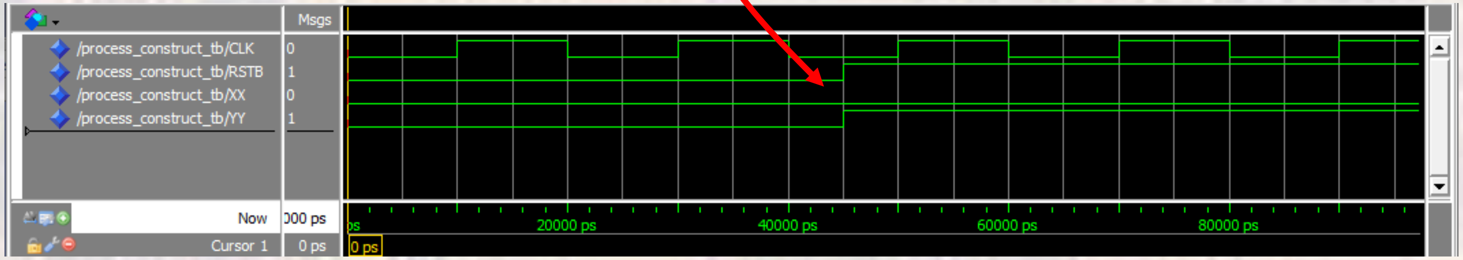
- The code is only executed when a signal in its sensitivity list has changed – correct version



```
process(i_clk, i_rstb)
begin
      if(i_rstb = '0') then
         x <= '0';
         y <= '0';
      elsif(i_rstb = '1') then
         y <= '1';
      elsif(rising_edge(i_clk)) then
         x <= '1';
         y <= x;
      end if;
end process;
```

Note: i_rstb IS in sensitivity list

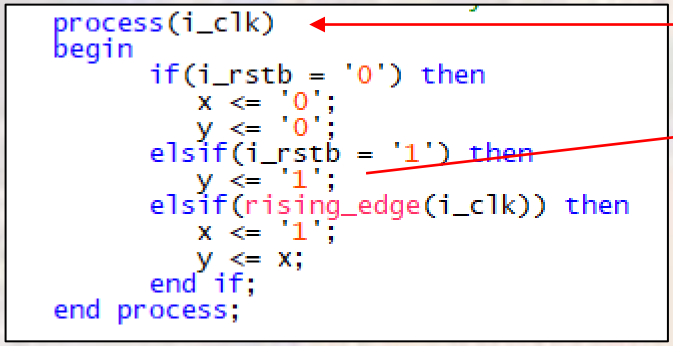
Expect Y → 1 when i_rstb → 1



Msgs
/process_construct_tb/CLK 0
/process_construct_tb/RSTB 1
/process_construct_tb/XX 0
/process_construct_tb/YY 1
Now 000 ps
Cursor 1 0 ps
ps
20000 ps
40000 ps
60000 ps
80000 ps
0 ps

Process Construct

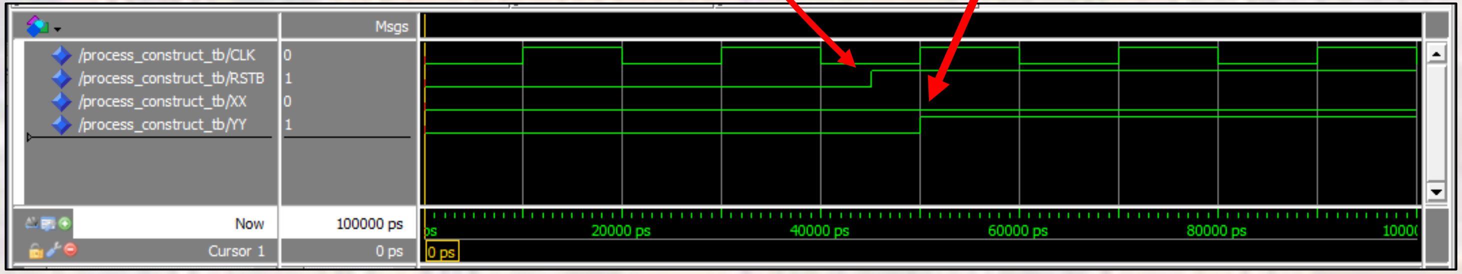
- The code is only executed when a signal in its sensitivity list has changed – incorrect version



```
process(i_clk)
begin
      if(i_rstb = '0') then
         x <= '0';
         y <= '0';
      elsif(i_rstb = '1') then
         y <= '1';
      elsif(rising_edge(i_clk)) then
         x <= '1';
         y <= x;
      end if;
end process;
```

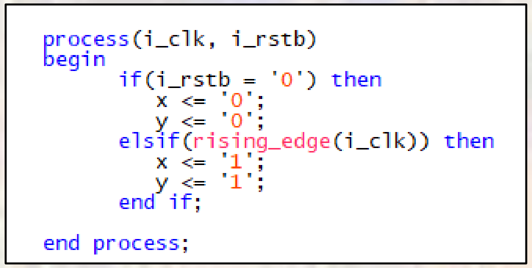
Note: i_rstb is NOT in sensitivity list

Expect Y → 1 when i_rstb → 1
but
The process is only called by i_clk
so
the change in i_rstb is not seen until the next clock edge


Msgs
/process_construct_tb/CLK 0
/process_construct_tb/RSTB 1
/process_construct_tb/XX 0
/process_construct_tb/YY 1
Now 100000 ps
Cursor 1 0 ps
ps
20000 ps
40000 ps
60000 ps
80000 ps
10000
0 ps

Process Construct

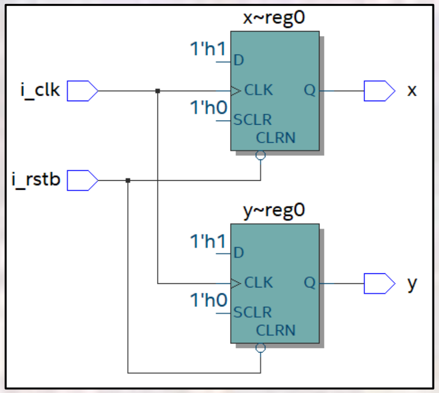
- Processes update sequential signals at the end of the process



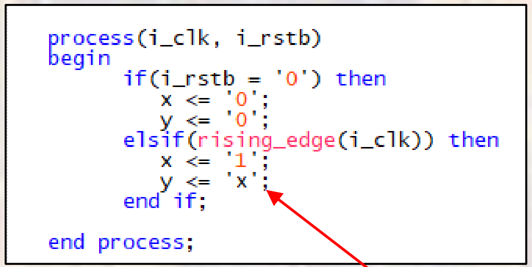
```
process(i_clk, i_rstb)
begin
     if(i_rstb = '0') then
        x <= '0';
        y <= '0';
     elsif(rising_edge(i_clk)) then
        x <= '1';
        y <= '1';
     end if;

end process;
```

Expected result:
x, y → 1 at the same time



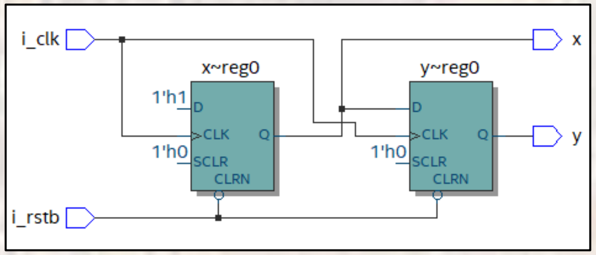
These appear to do the same thing:
Set x and y → 1 at the same time



```
process(i_clk, i_rstb)
begin
     if(i_rstb = '0') then
        x <= '0';
        y <= '0';
     elsif(rising_edge(i_clk)) then
        x <= '1';
        y <= 'x';
     end if;

end process;
```

Unexpected (but correct) result:
y is not updated to the value
of x until the next clk



X has not been changed to 1 at this point
It only becomes 1 at the end of the process

Process Construct

- Warning – Warning – Warning
  - If you do not complete an if-else with an else, a latch will be created
  - If you do not cover all cases in a case statement, a latch will be created
  - All paths/cases must be covered
  - The compiler will always warn you it created a latch

We do not want latches - EVER

I can see a latch in an RTL diagram from a mile away

The FF construct is one of very few exceptions