Electronic Systems

Assignment Practical Tips

Ing. Luca Zulberti – luca.zulberti@phd.unipi.it
Prof. Massimiliano Donati – massimiliano.donati@unipi.it
Prof. Luca Fanucci – luca.fanucci@unipi.it

Template for Process

Combinational

```
-- Comb Process
comb_process_label: process(a, b, [...])
-- Sensitivity list shall include all the signals
-- that are read by the process (the inputs of the
-- comb network).
begin
  c <= '0';
  d <= (others => '0');
  -- It is considered good practice to assign a
  -- default value to comb outputs. This prevents
  -- the inferring of latches!

  -- Comb function
  if ([...]) then
    c <= [...];
    d <= [...];
  end if;
end process;
```

Template for Process

Sequential

```
-- Seq Process, async reset, active low
seq_process_label: process(clk, a_rst_n)
begin
  if (a_rst_n = '0') then
    -- reset values for all signals
    -- assigned within the process
    a <= '0';
    b <= (others => '0');
  elsif (rising_edge(clk)) then
    -- Process Body
    a <= [...];
    b <= [...];
  end if;
end process;
```

```
-- Seq Process, sync reset, active high
seq_process_label: process(clk)
begin
  if (rising_edge(clk)) then
    if (rst = '1') then
      -- reset values for all signals
      -- assigned within the process
      a <= '0';
      b <= (others => '0');
    else
      -- Process Body
      a <= [...];
      b <= [...];
    end if;
  end if;
end process;
```

Instance of components

- Component declaration:
  - Component declaration is basically the copy of the original entity (both generics and ports)
- Component instance:
  - When instancing a component, remember to specify:
    - «generic map» (if generics are used)
    - «port map»
  - Generics and ports names on the **left** side of assignments must be the same of the original entity
  - Generics, ports and signals on the **right** side of assignment can have the same name of the ones on the left (e.g: clk)
  - Multiple instances of the same component are allowed, provided that they have different labels

Use of Variables

- Know the difference between variables and signals (See 1st set of slides).
- Variables may be necessary when working with loops and indexing.
- Variables can be used to store temporary values and improve reading of process VHDL code.
- If you use them, you will be asked to motivate why you used a variable instead of a signal.

Use of Constants

- Beware, do not use initialised signals as a constant. Vivado may prune them if they only keep their default value.
- Tricky, it will work without any problem or warning in simulation, but may give issues during Synthesis
- Use Constants!

```
-- Wrong use
signal CONSTANT_VALUE : integer := 10;
signal CONSTANT_VEC   : std_logic_vector(N-1 downto 0)
                        := (N-3 => '1', others => '0');
-- Correct use
constant CONSTANT_VALUE : integer := 10;
constant CONSTANT_VEC   : std_logic_vector(N-1 downto 0)
                        := (N-3 => '1', others => '0');
```

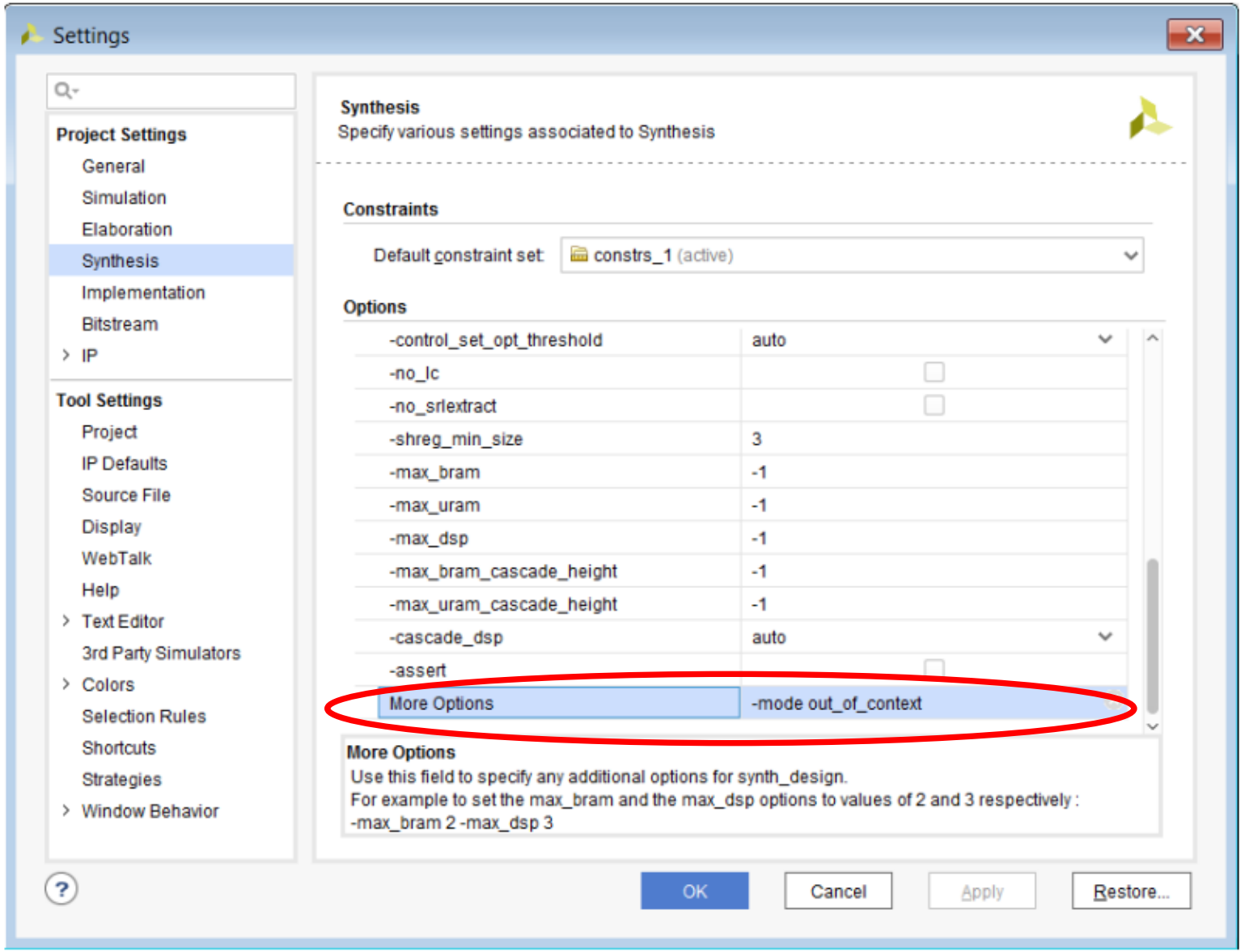
Signal Initialisation

- You MUST initialise all the signals used as input in Testbenches.
- You MUST NOT initialise any other signals.

- Tricky! You may apparently fix bugs by initialising signals in your HDL sources, but you are just hiding the problem, which will be back in FPGA operation.

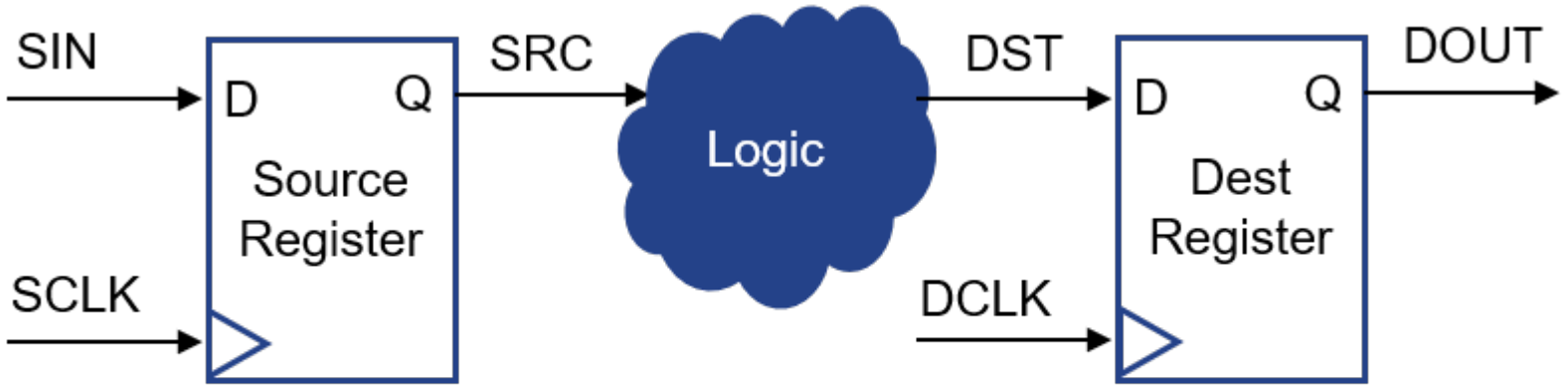
Out-of-context Synthesis mode

- If you want to perform also the Implementation of your Circuit, but you do not want to map the I/O of your circuit to the FPGA physical pins, you can tell Vivado to skip that part and perform the circuit implementation within the inner part of the FPGA.
- Just add "-mode out_of_context" within the synthesis options.



I/O Registers

- Vivado evaluates timing only for “Register - Logic - Register” paths.
- In order for Vivado to correctly evaluate the timing of your circuit, you must put a register barrier at all inputs and outputs. It is good practice to put these barriers in a separate entity (a wrapper entity).



Multiple Clock Domains

- All the Assigned projects are solvable with a **single clock domain**.
- Handling multiple clock domain within the same circuit is an advanced skill:
  - Clock Domain Crossing (CDC)
- If you choose to exploit multiple clocks within your design, take care of:
  - Physical interfacing of clocks.
  - Proper timing constraints.
- If you decide to use multiple clocks, you are strongly encouraged to discuss your project with us before the final submission of the assignment.

End, Questions?