Digital Design

2018/19

prof. dr. Patricio Bulić

Rok Češnovar

Practical Exercise summary

- Digital design in VHDL
- Building a System-on-Chip
  - building a VGA controller
  - building a PS2 controller
  - connecting and using a PicoBlaze
  - …
- Requirements to pass the Practical Exercises
  - build and defend a seminar project

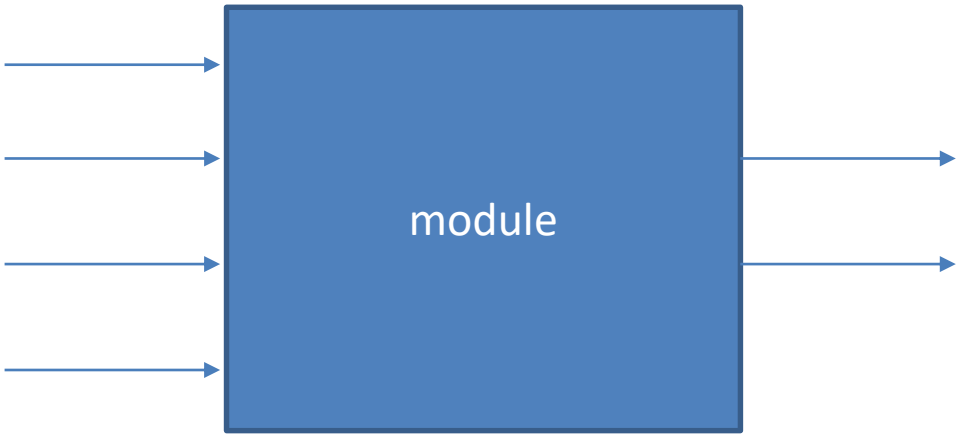
VHDL

- VHSIC Hardware Description Language
  - VHSIC = very-high-speed integrated circuits
- Language to describe digital systems/hardware and modeling/simulating circuits
- VHDL description is syntehsized and programmed to a field programmable gate array (FPGA)

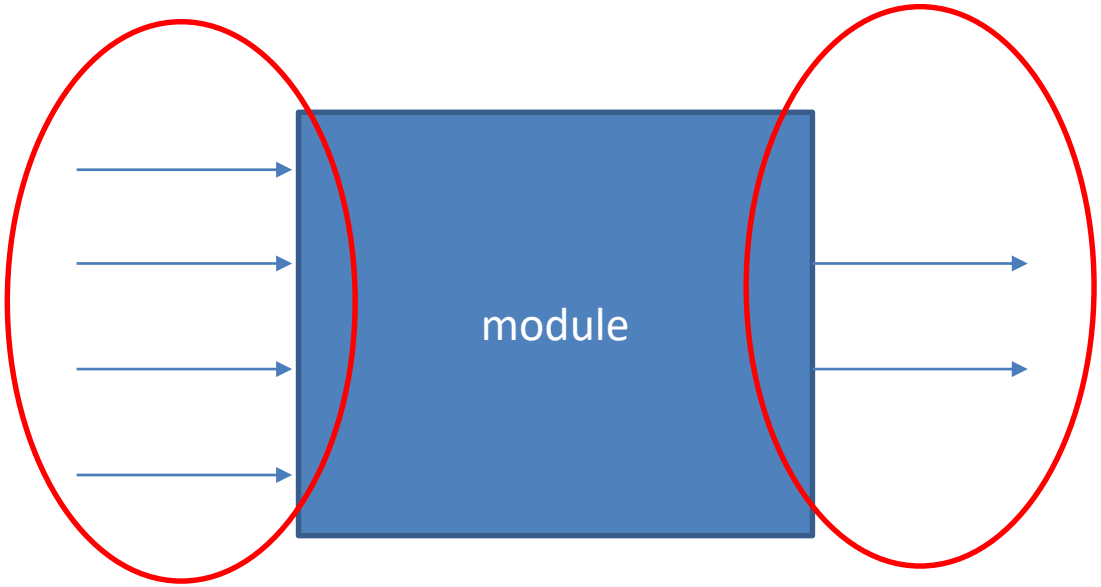
Exercise equipment/software

- Software
  - Xillinx ISE Webpack (link on e-učilnica/moodle)
- Hardware
  - Development board Digilent Nexys4
    - Xilinx Artix-7 with a 100 MHz clock
    - Input/Output: VGA, USB (PS2 emulator), LED, buttons, switches, 7 segment display, 3-color LED, accelerometer/gyroscope, …

Describing a basic module

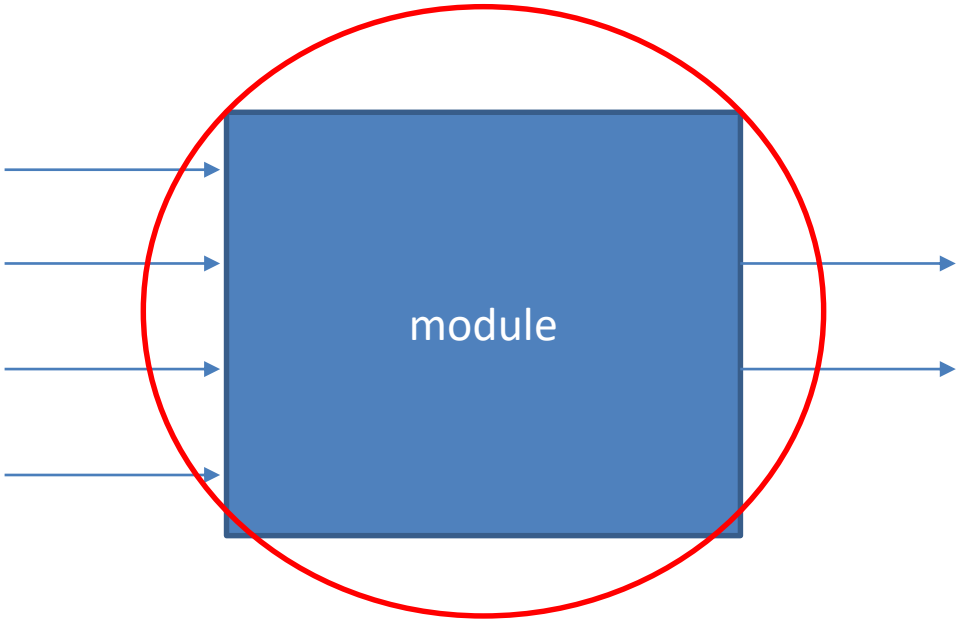


Describing a basic module



Define the input/output

Describing a basic module



define the modules logic (what to do with the input/output)

Describing a basic module in VHDL

```
entity module_name is
        //define the input/output
        port( … );
end module_name;

architecture circuit_description of module_name is
//define inner signals
begin
        //describe the circuit
end circuit_description;
```

Declare the input/output

```
entity module_name is
port (
        signal_name : direction signal_type;
         signal_name _2 : direction signal_type;
        …
         signal_name _n: direction signal_type
        );
end module_name;
```

- Direction: in, out, inout, buffer
- Signal type: std_logic, std_logic_vector()

Examples

```
entity module_name is
port (
        a: in std_logic;
        b: out std_logic
        );
end module_name;

entity module_name is
port (
        a: in std_logic_vector(7 downto 0);
        b: out std_logic_vector(0 to 4)
        );
end module_name;
```

Define inner signals

```
architecture Behavioral of module_name is
        signal signal_name: signal_type;
begin …
```

Assignment

Syntax:

signal <= expression;

Examples:

a <= '0'; //assign a constant

b <= "01001"; //assign a vector

c(3 downto 0) <= "0111"; //assign a part of a vector

Example – logic gate

```
entity and_xor_gate is
        port(
                a: in std_logic;
                b: in std_logic;
                c: out std_logic;
                d: out std_logic
                );
end and_xor_gate;
```

Describe the circuit

```
architecture Behavioral of and_xor_gate is
begin
        //assignment statements
        c <= a and b;
        d <= a xor b;
end Behavioral ;
```

Module input/output and FPGA

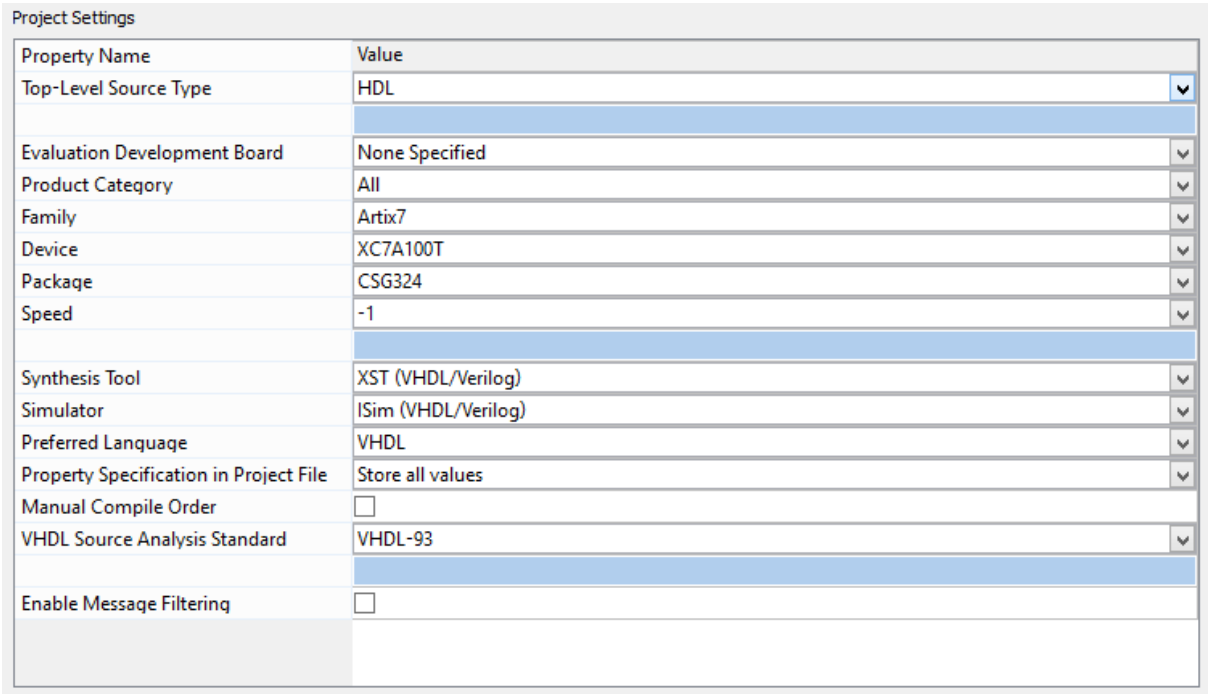
- Before synthesis we need to define the where the signals of the out-most module go on the FPGA
  - example: a and b are switches, c & d are LEDs
- We do this in a UCF (user constraints file)
- Syntax

```
NET "a" LOC = "J14";
NET "b<1>" LOC = "H18";
NET "b<0>" LOC = "G18";
```

- The input/output labels are found in the reference manual of the development board or on the development boards

Xillinx ISE Design Tools Project for the Nexys4

- Open Project Navigator
- File -> New Project
  - Choose a name and folder
- Choose the following settings -> Next -> Finish



Add a new module

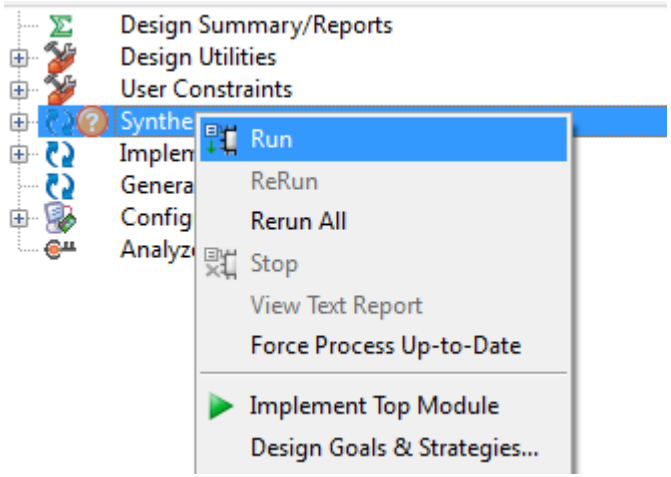
- Project -> New Source
- Choose VHDL Module
- Choose a name -> Next -> Next -> Finish

Adding the UCF

- Project -> New Source
- Choose Implementation Constraints File
- Choose a name -> Next -> Finish

Syntax check and synthesis

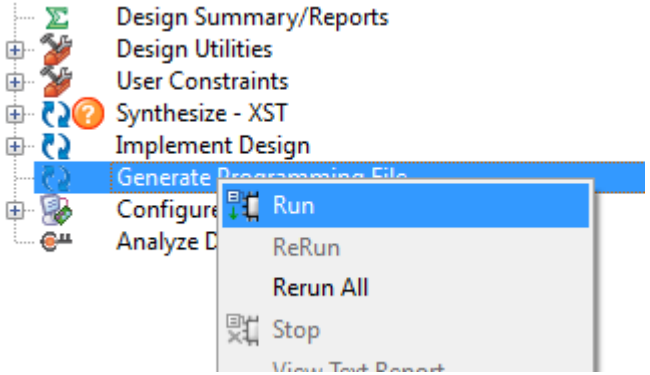
- Right click on Synthesize -> Run



- Check Syntax for only a syntax check

FPGA programming

- Right click on Generate Programming File



- When a green tick appears we can program the FPGA chip

Program the FPGA

- Double click on Configure Target Device
- ISE iMPACT
  - Boundary Scan
  - Right click in the right blank area -> Initialize Chain
  - Yes
  - Find the created .bit file
  - No
  - Choose Bypass and OK
  - Right click on XC7A100T and Program

Condition assignment statements

```
signal <= expression1 when condition1 else
expression2;

signal <= expression1 when condition1 else
          expression2 when condition2 else
          expression3;
```

Condition operator:

- equal, not equal =, /=
- greater, less than,… >,<,>=,<=

Exercise

- Create a comparator of two 4-bit numbers
  - output = 2, when the first number is greater
  - output = 1, when the second number is greater
  - output = 0, when the numbers are equal