



COMPUTER ARCHITECTURE
LAB1
SYNTHESIZED VHDL CODING

FCIS Ainshams University
Spring2021

AGENDA

- Need to reprogrammable hardware (FPGA)
- Introduction to VHDL
- Code Structure
- Hands-on 1: AND gate + Simulator Environment
- Data types
- Generic building blocks of MIPS:
 - Hands-on 2: sl2
 - Hands-on 3: generic 2×1 MUX

NEED TO PROGRAMMABLE HARDWARE

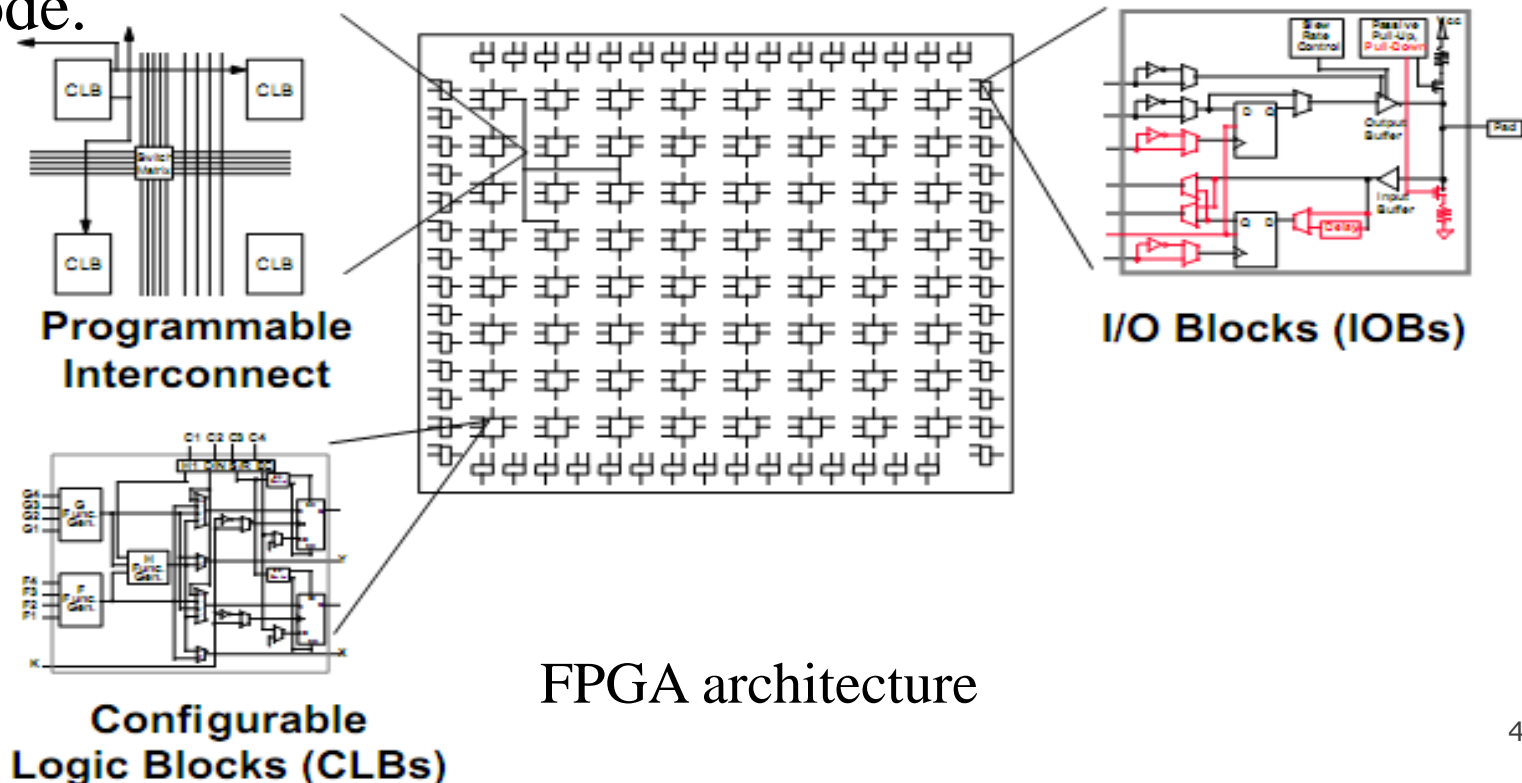
To Design a hardware solution

- Define the problem
- Design the Logic circuit
- Implement the design
- Evaluate and test the hardware circuit
- What you will do to change this hardware solution
- This is the ASIC (application-specific integrated circuit)

Ex: after we launch a satellite to its orbit, we need to change some of its logic ???!

FPGA

Field Programmable Gate Arrays (FPGA) is an **integrated circuit** that is capable of being **reprogrammed** after its manufacture using **HDL** code.



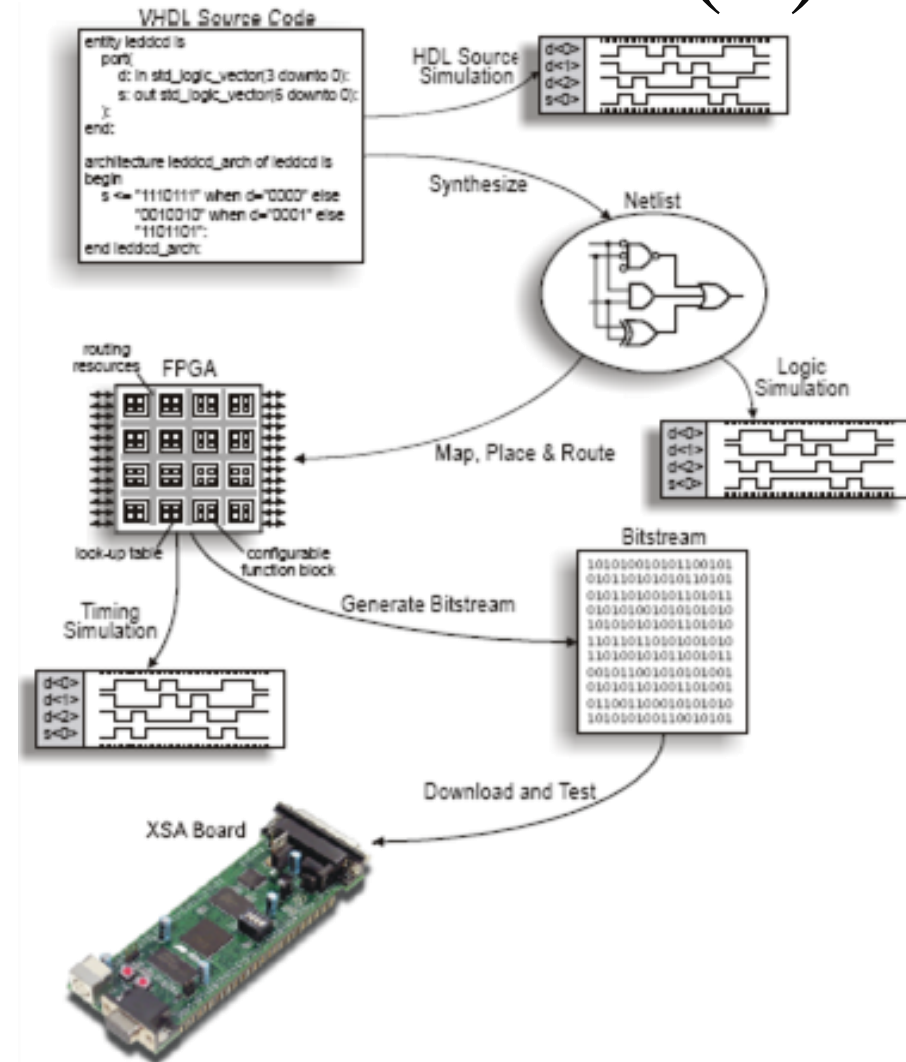
INTRODUCTION TO VHDL (1)

- VHDL stands for VHSIC HDL.
 - VHSIC: Very High-Speed Integrated Circuits
 - HDL: Hardware Description Language
- It describes the behavior of an electronic circuit or system.
 - It can be translated into a hardware circuit
 - It can be tested on software simulation before hardware implementation
- VHDL is a standard, technology/vendor independent language, and is therefore **portable** and **reusable**.

INTRODUCTION TO VHDL (2)

Circuit Design Flow:

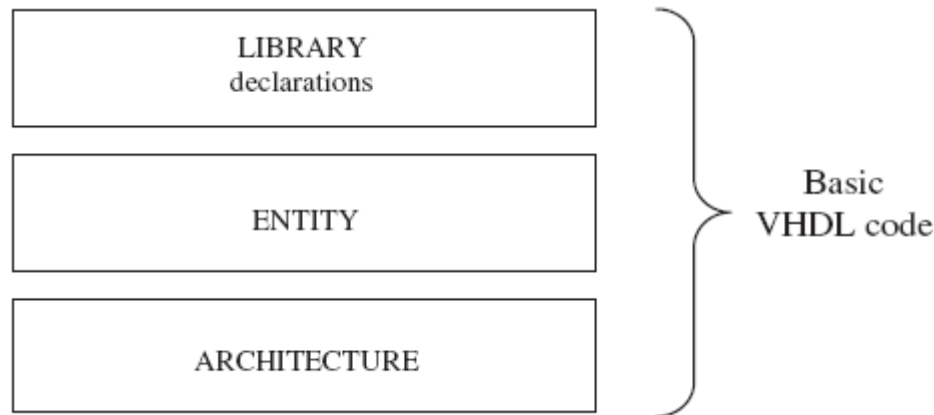
- VHDL Code & simulation
- Synthesized to Netlist file
- Map, Place and route
- Generated Bit-stream
- Download and test



CODE STRUCTURE

VHDL Code is divided into 3 parts:

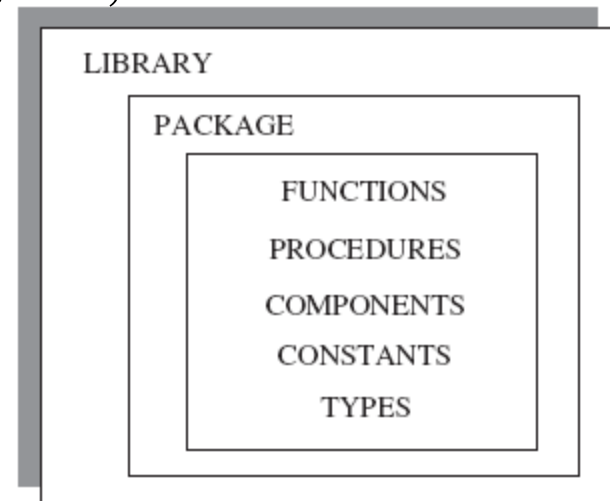
- Library declaration: like using statement in C#
- Entity: specifies I/O pins of the circuit
- Architecture: describes the behavior or function of the circuit



CODE STRUCTURE: LIBRARY(1)

LIBRARY declarations:

- Contains a list of all libraries to be used in the design.
- Most common libraries are ieee, std, work
- A library contains packages,
- and a package contains parts
(data types & subprograms)



CODE STRUCTURE: LIBRARY (2)

To use a library,

```
LIBRARY library_name;
```

```
USE library_name.package_name.  
package_parts;
```

For example:

```
LIBRARY ieee;           -- A semi-colon (;) indicates  
USE ieee.std_logic_1164.all; -- the end of a statement or  
LIBRARY std;           -- declaration, while a double  
USE std.standard.all;  -- dash (--) indicates a comment.  
LIBRARY work;  
USE work.all;
```

CODE STRUCTURE: ENTITY (1)

Entity is the most basic building block in a design. It is a list (with specifications) of all input and output pins (ports) of the circuit.

```
entity <entity_name> is
port (
    <port_name> : <mode> <type>;
    <other ports>...);
end <entity_name>;
```

- port_name: any name, except VHDL reserved words
- signal_mode: IN, OUT, or INOUT.
- signal_type: BIT, STD_LOGIC, INTEGER,...

CODE STRUCTURE: ENTITY (2)

For example, let us consider the AND gate entity, its entity can be described as:

```
entity and_gate IS
```

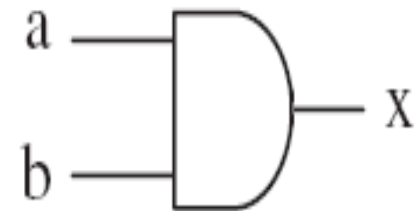
```
port ( a : in STD_LOGIC;
```

```
       b : in STD_LOGIC;
```

```
       x : out STD_LOGIC);
```

```
    x <= a AND b;
```

```
end and_gate;
```



CODE STRUCTURE: ARCHITECTURE

Architecture contains the VHDL code, which describes the behavior of the entity.

```
ARCHITECTURE archi_name OF entity_name IS
```

```
    [declarations]
```

```
BEGIN
```

```
    (code)
```

```
END archi_name;
```

CODE STRUCTURE: ARCHITECTURE

For example, the architecture of AND gate should be:

```
ARCHITECTURE myarch OF and_gate IS
```

```
BEGIN
```

```
    x <= a AND b;
```

```
END myarch;
```

CODE NOTES

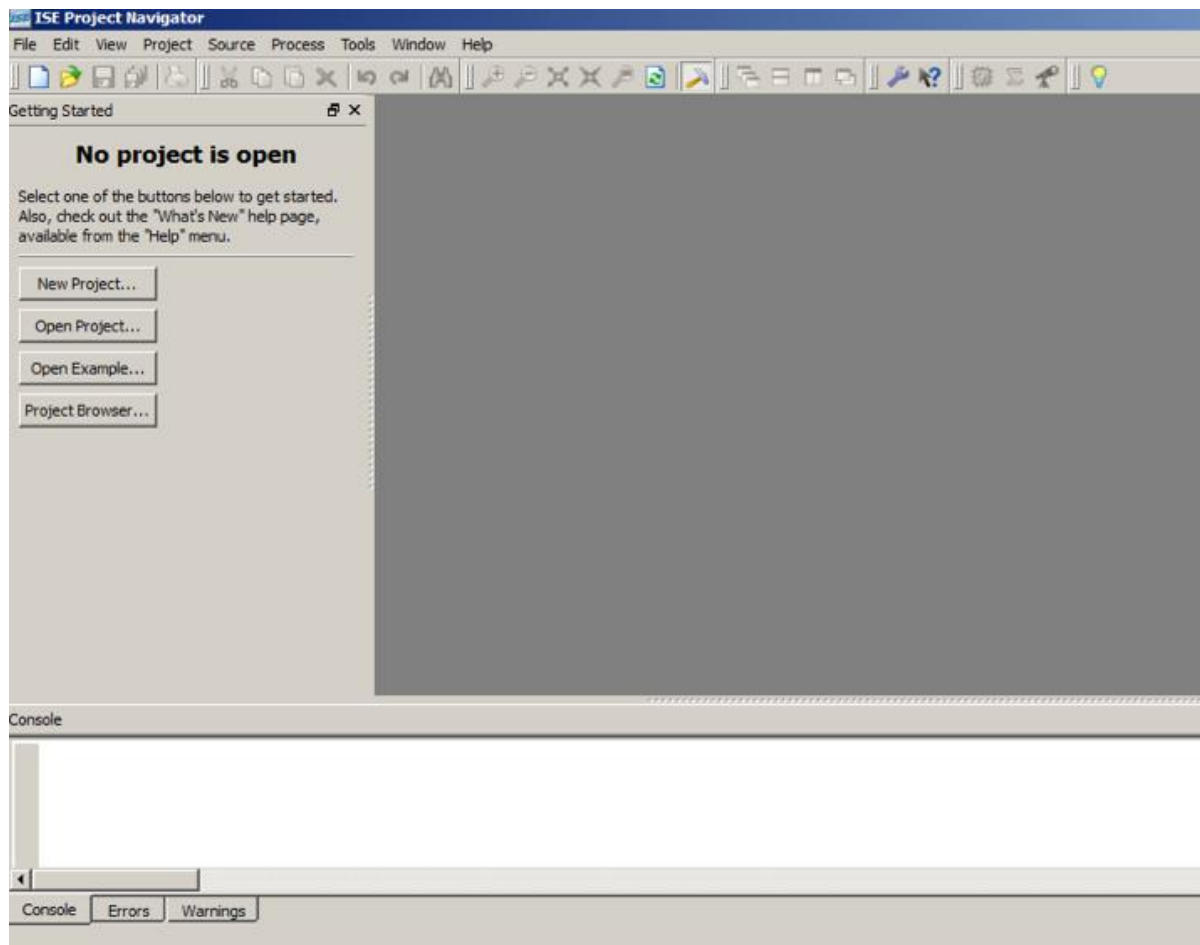
- VHDL is case insensitive.
- Its statements are inherently concurrent (parallel).
- Only statements placed inside a PROCESS, FUNCTION, or PROCEDURE are executed sequentially.
- VHDL is a hardware description language, so our main goal is the RTL not reducing number of code lines.

HANDS-ON1: AND GATE CIRCUIT

Implement and test the AND gate circuit.

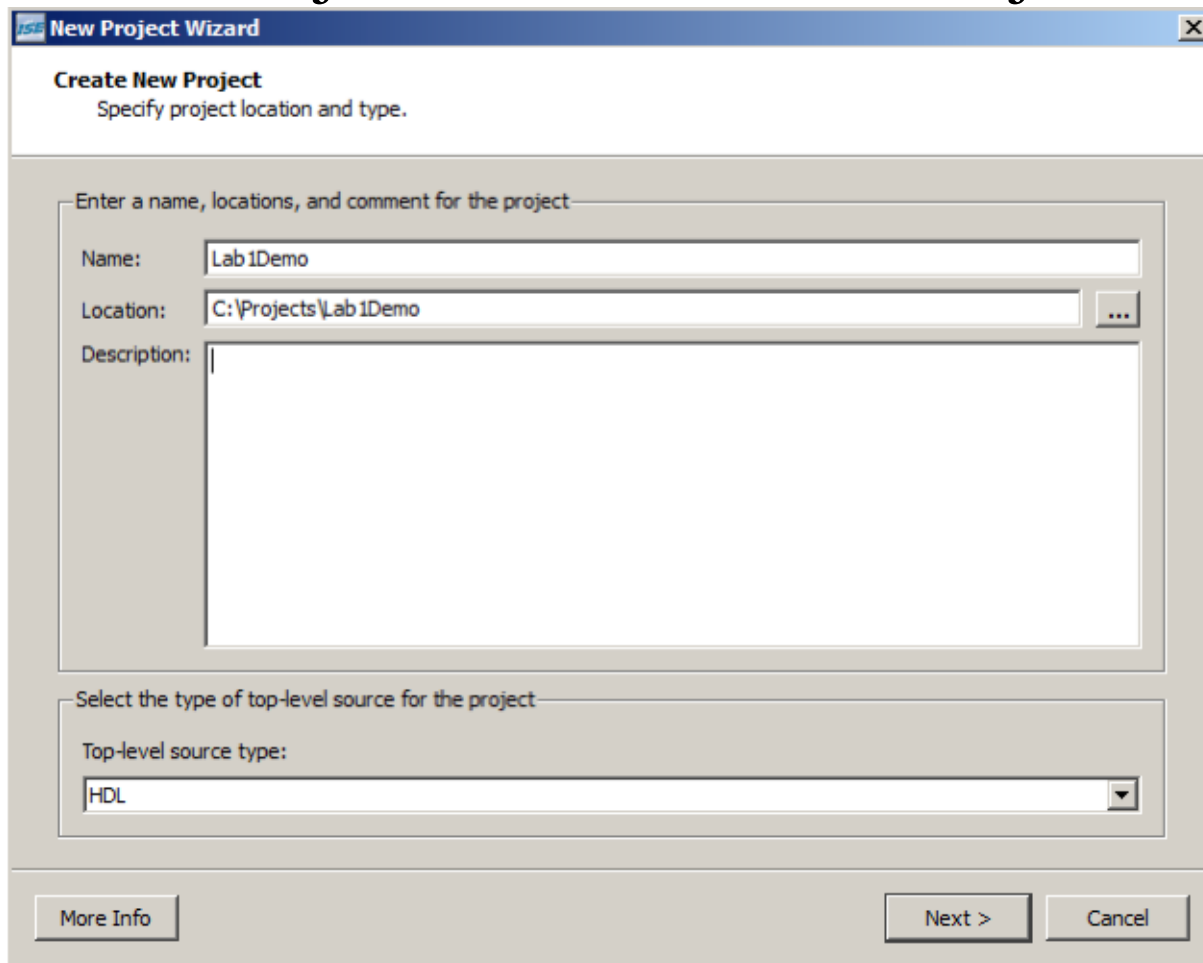
XILINX GETTING STARTED (1)

We will use Xilinx ISE 12 to write and simulate VHDL code



XILINX GETTING STARTED (2)

Create a New Project from File > New Project



The image shows a screenshot of the 'New Project Wizard' dialog box in Xilinx. The window title is 'New Project Wizard'. The main heading is 'Create New Project' with the instruction 'Specify project location and type.' Below this, there is a section titled 'Enter a name, locations, and comment for the project'. This section contains three input fields: 'Name' with the value 'Lab1Demo', 'Location' with the value 'C:\Projects\Lab1Demo' and a browse button (...), and a 'Description' text area. Below this section is another section titled 'Select the type of top-level source for the project'. It contains a 'Top-level source type:' dropdown menu with 'HDL' selected. At the bottom of the dialog, there are three buttons: 'More Info', 'Next >', and 'Cancel'.

XILINX GETTING STARTED (3)

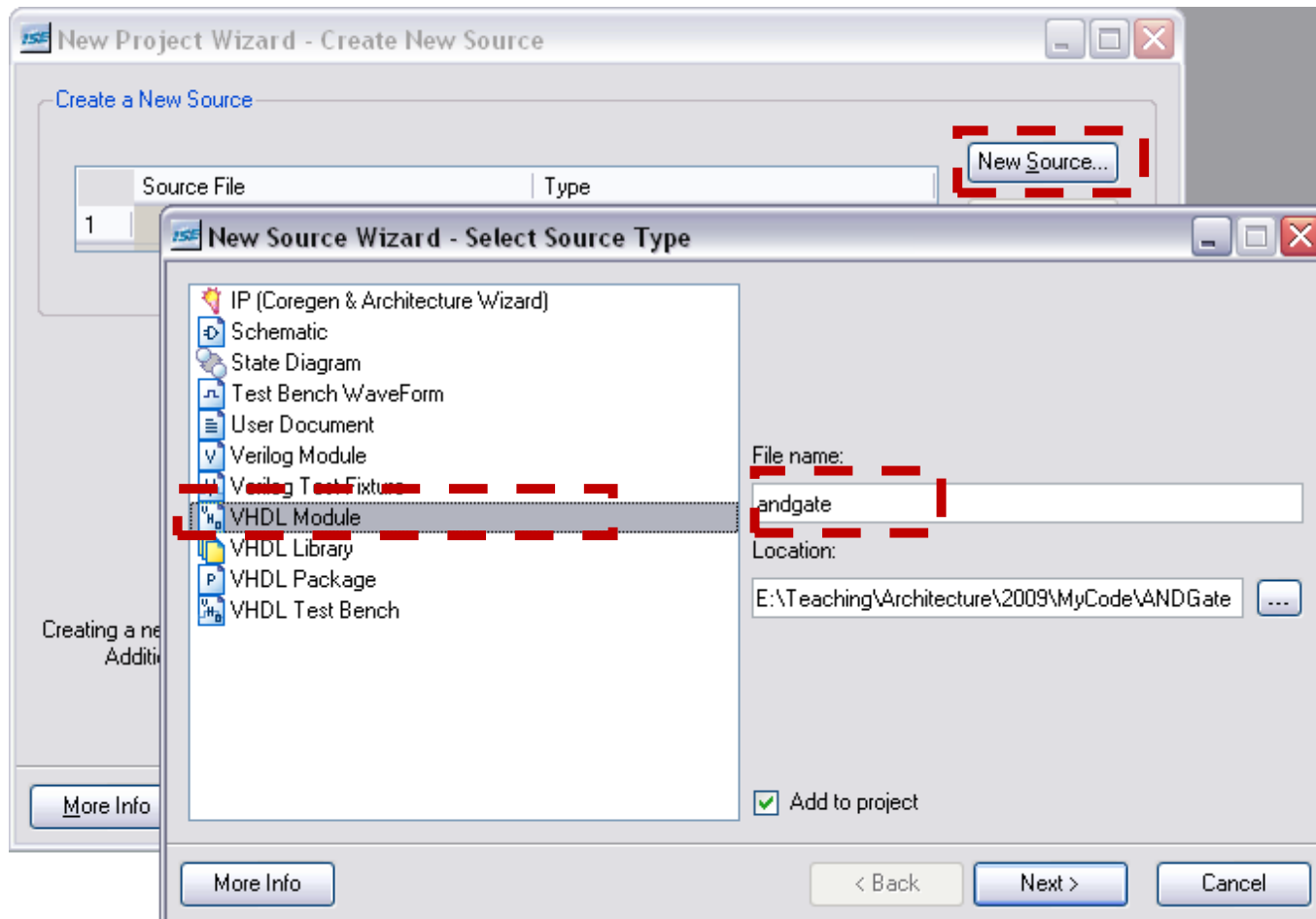
Specify the Language to be “VHDL”

The screenshot shows the 'New Project Wizard - Device Properties' dialog box. The title bar includes the ISE logo and the text 'New Project Wizard - Device Properties'. Below the title bar, there is a subtitle 'Select the Device and Design Flow for the Project'. The main area contains a table with two columns: 'Property Name' and 'Value'. The 'Preferred Language' property is highlighted with a red dashed box and has 'VHDL' selected in its dropdown menu. Other properties include Product Category (All), Family (Virtex2P), Device (XC2VP20), Package (FF896), Speed (-7), Top-Level Source Type (HDL), Synthesis Tool (XST (VHDL/Verilog)), Simulator (ISE Simulator (VHDL/Verilog)), Enable Enhanced Design Summary (checked), Enable Message Filtering (unchecked), and Display Incremental Messages (unchecked). At the bottom of the dialog, there are three buttons: 'More Info', '< Back', and 'Next >', and a 'Cancel' button.

Property Name	Value
Product Category	All
Family	Virtex2P
Device	XC2VP20
Package	FF896
Speed	-7
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISE Simulator (VHDL/Verilog)
Preferred Language	VHDL
Enable Enhanced Design Summary	<input checked="" type="checkbox"/>
Enable Message Filtering	<input type="checkbox"/>
Display Incremental Messages	<input type="checkbox"/>

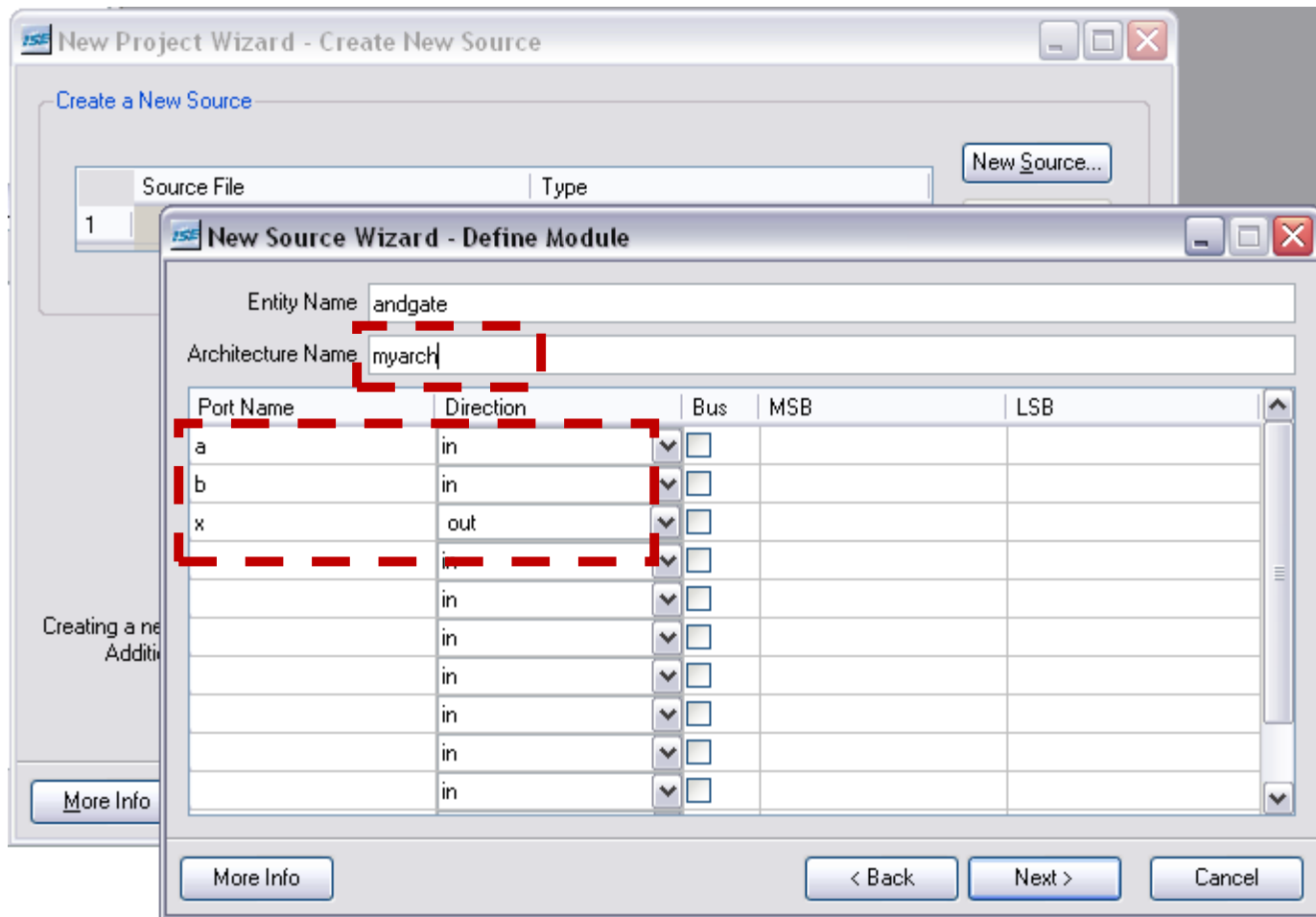
XILINX GETTING STARTED (4)

Choose New Source > VHDL Module and choose a name



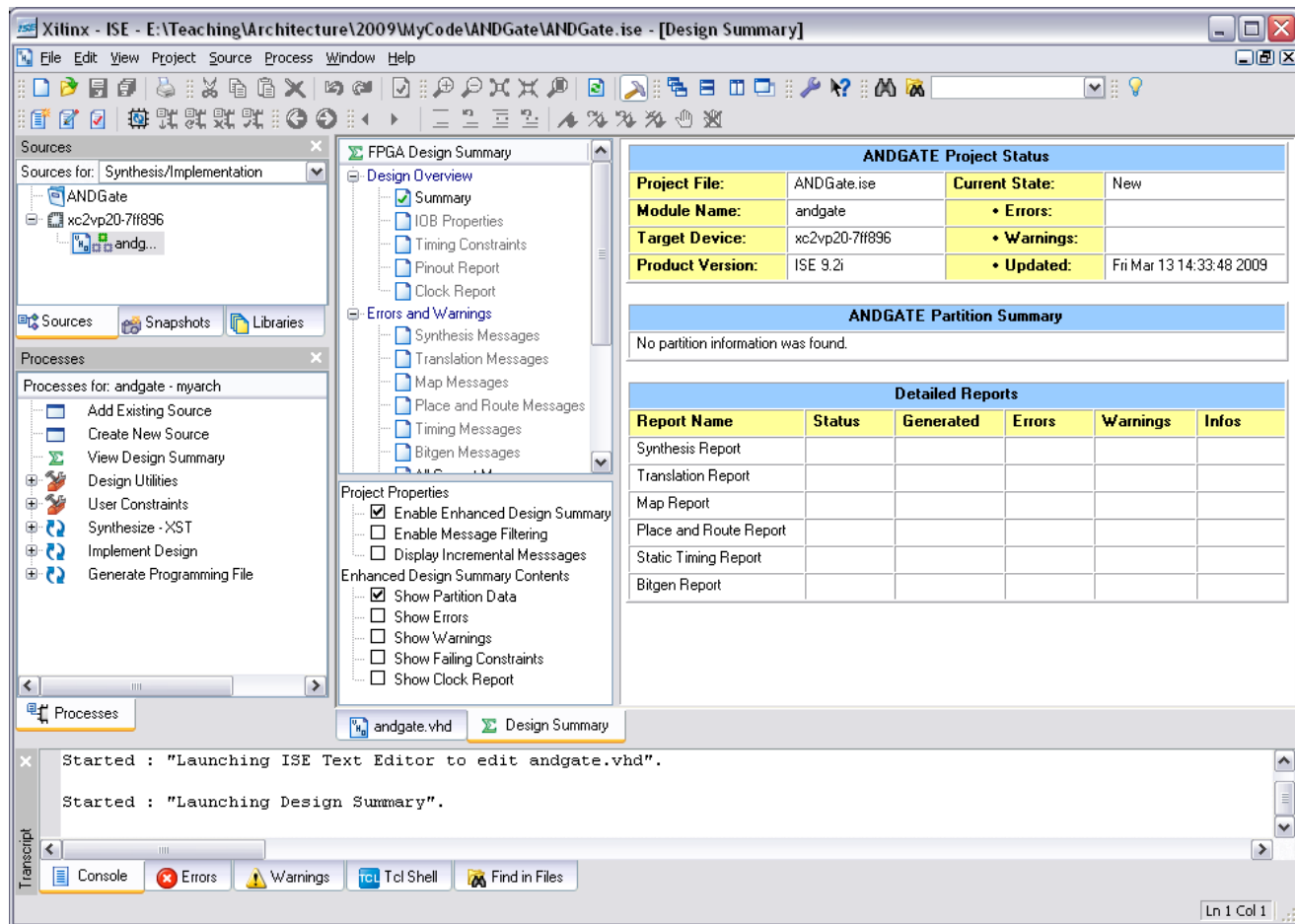
XILINX GETTING STARTED (5)

Use entity wizard to create your entity



XILINX GETTING STARTED (6)

Then click next buttons for subsequent dialogs till wizard is finished



The screenshot shows the Xilinx ISE Design Summary window for a project named 'ANDGATE'. The window is divided into several panes:

- Sources:** Shows the project sources for 'Synthesis/Implementation', including 'ANDGate' and 'xc2vp20-7ff896'.
- Processes:** Lists various processes for 'andgate - myarch', such as 'Add Existing Source', 'Create New Source', 'View Design Summary', 'Design Utilities', 'User Constraints', 'Synthesize -XST', 'Implement Design', and 'Generate Programming File'.
- FPGA Design Summary:** A tree view showing the design overview and errors/warnings sections.
- Project Properties:** A list of checkboxes for enabling enhanced design summary, message filtering, and displaying incremental messages.
- Enhanced Design Summary Contents:** A list of checkboxes for showing partition data, errors, warnings, failing constraints, and clock reports.
- ANDGATE Project Status:** A table summarizing project information.
- ANDGATE Partition Summary:** A section indicating that no partition information was found.
- Detailed Reports:** A table listing various reports and their status.
- Transcript:** A log of system messages, including 'Starting ISE Text Editor' and 'Starting Design Summary'.

Property	Value	Property	Value
Project File:	ANDGate.ise	Current State:	New
Module Name:	andgate	• Errors:	
Target Device:	xc2vp20-7ff896	• Warnings:	
Product Version:	ISE 9.2i	• Updated:	Fri Mar 13 14:33:48 2009

No partition information was found.

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report					
Translation Report					
Map Report					
Place and Route Report					
Static Timing Report					
Bitgen Report					

XILINX GETTING STARTED (7)

Double click on source file on left panel and complete the architecture code

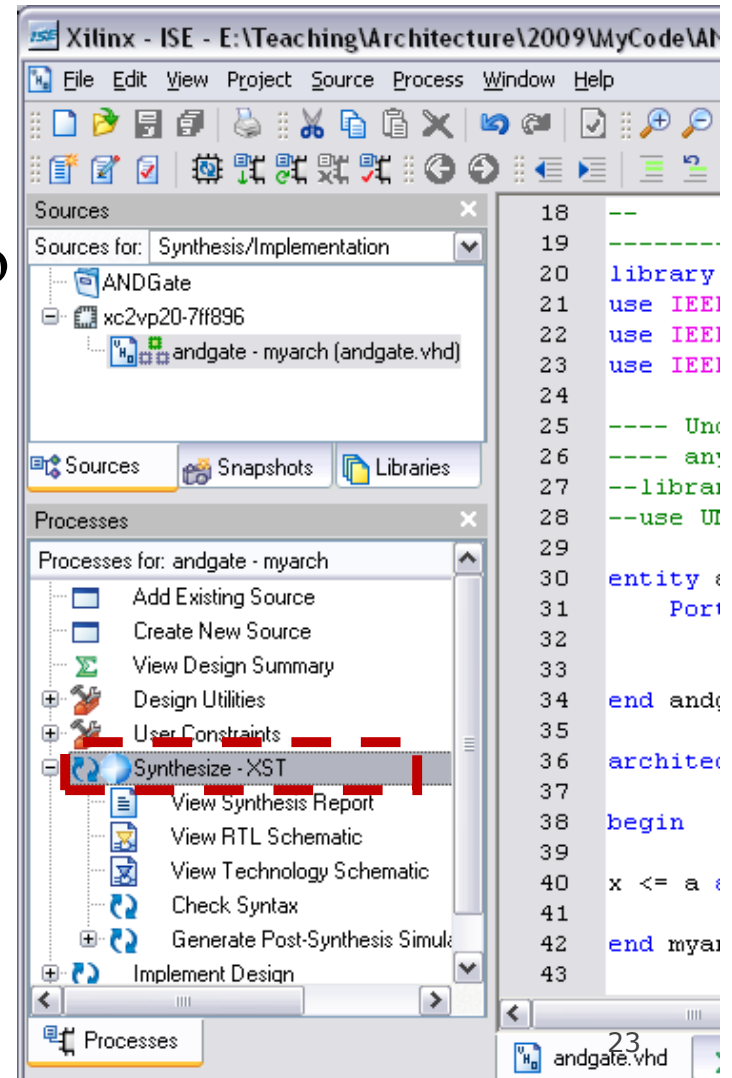
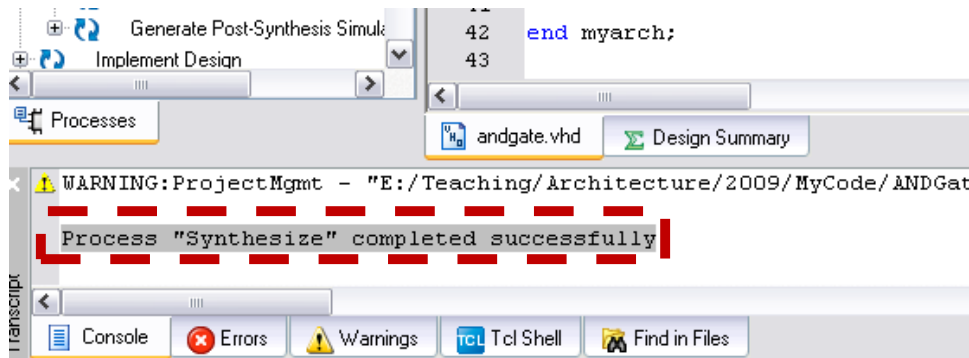
The screenshot shows the Xilinx ISE IDE with the following components:

- Sources Panel:** Shows the project structure for 'ANDGate'. The file 'andg...' is highlighted with a red dashed box.
- Processes Panel:** Shows the project processes for 'andgate - myarch', including 'Add Existing Source', 'Create New Source', 'View Design Summary', 'Design Utilities', 'User Constraints', 'Synthesize -XST', 'Implement Design', and 'Generate Programming File'.
- Code Editor:** Displays the VHDL code for 'andgate.vhd'. The code is annotated with yellow dashed boxes and labels:
 - Library Declarations:** Lines 20-23: `library IEEE;`, `use IEEE.STD_LOGIC_1164.ALL;`, `use IEEE.STD_LOGIC_ARITH.ALL;`, `use IEEE.STD_LOGIC_UNSIGNED.ALL;`
 - Entity Definition:** Lines 30-34: `entity andgate is`, `Port (a : in STD_LOGIC;`, `b : in STD_LOGIC;`, `x : out STD_LOGIC);`, `end andgate;`
 - Architecture:** Lines 36-43: `architecture myarch of andgate is`, `begin`, `<= a and b;`, `end myarch;`

XILINX GETTING STARTED (8)

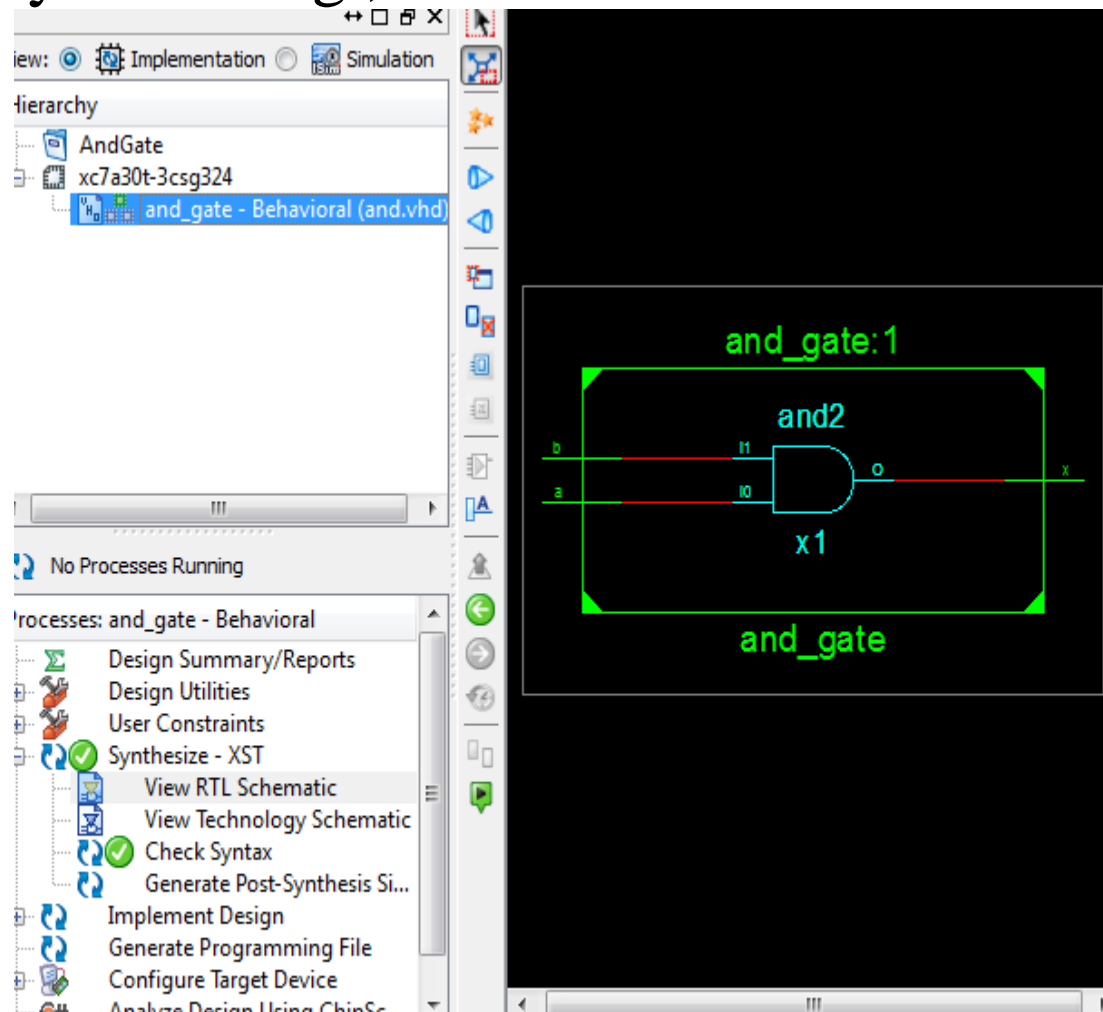
After editing, click **save** button

Then, in processes panel, **double click** “Synthesize – XST” item to synthesize your code



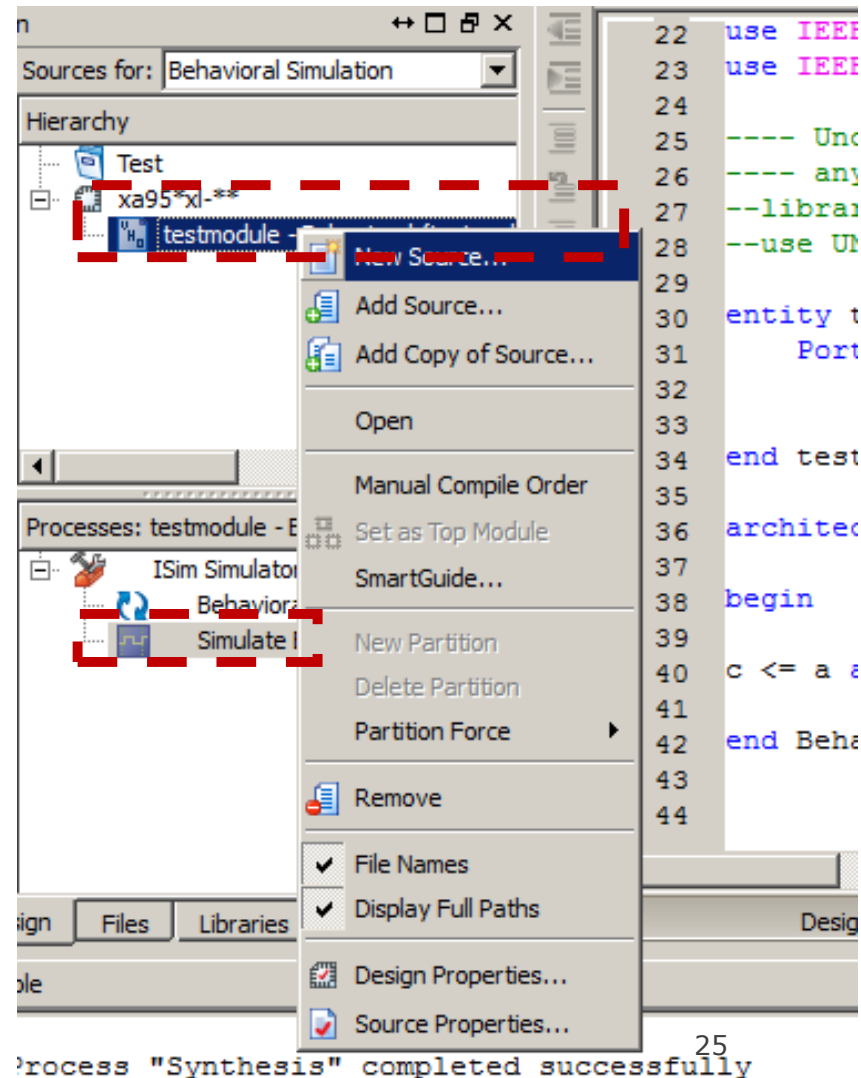
XILINX GETTING STARTED (9)

After synthesizing , click **view RTL Schematic** button



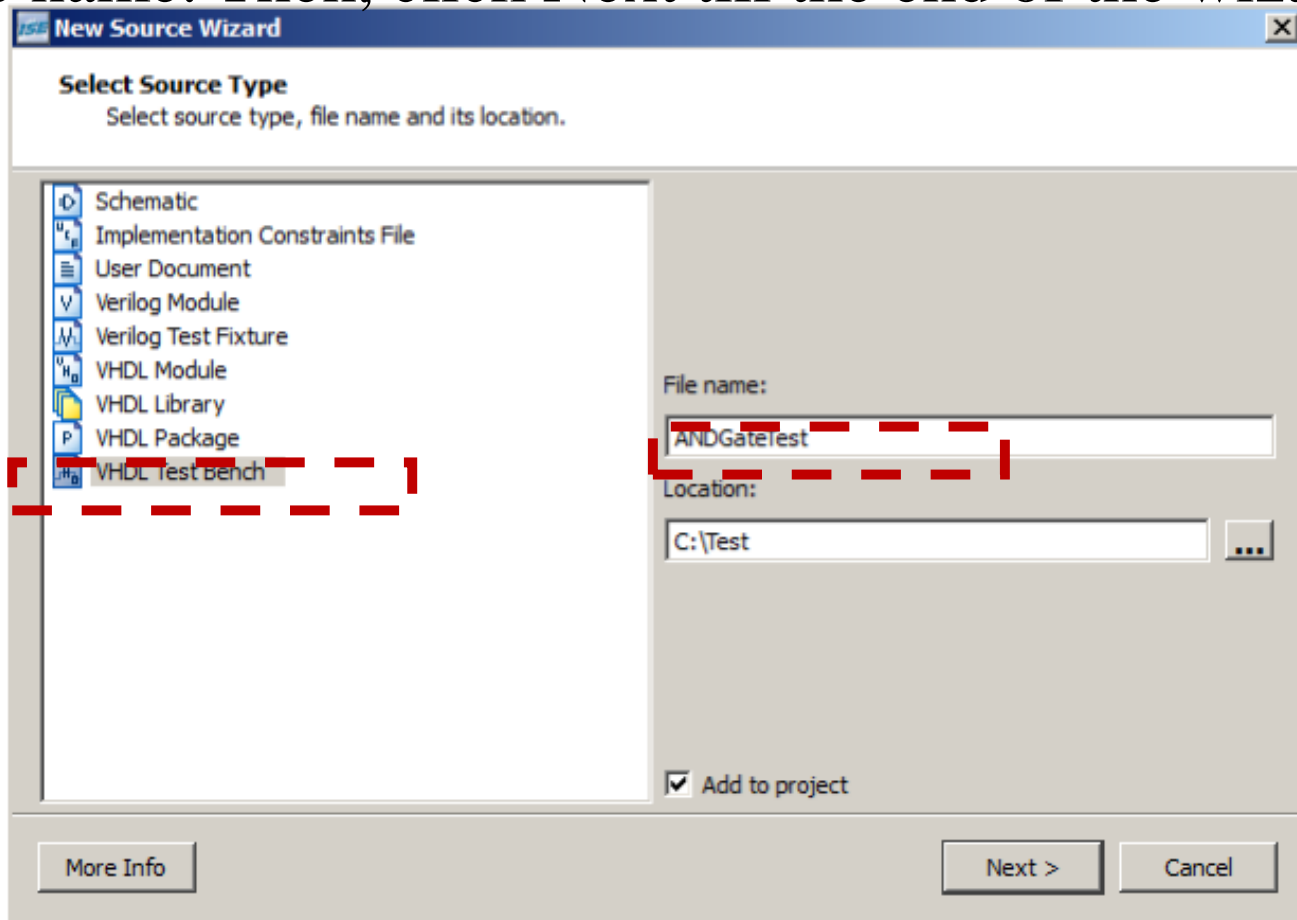
SIMULATION GETTING STARTED(1)

- To simulate your VHDL code, you need to add a simulation module
- To do so, in Sources panel, select “Behavioral Simulation”
- Right click the module you want to test and click on New Source



SIMULATION GETTING STARTED(2)

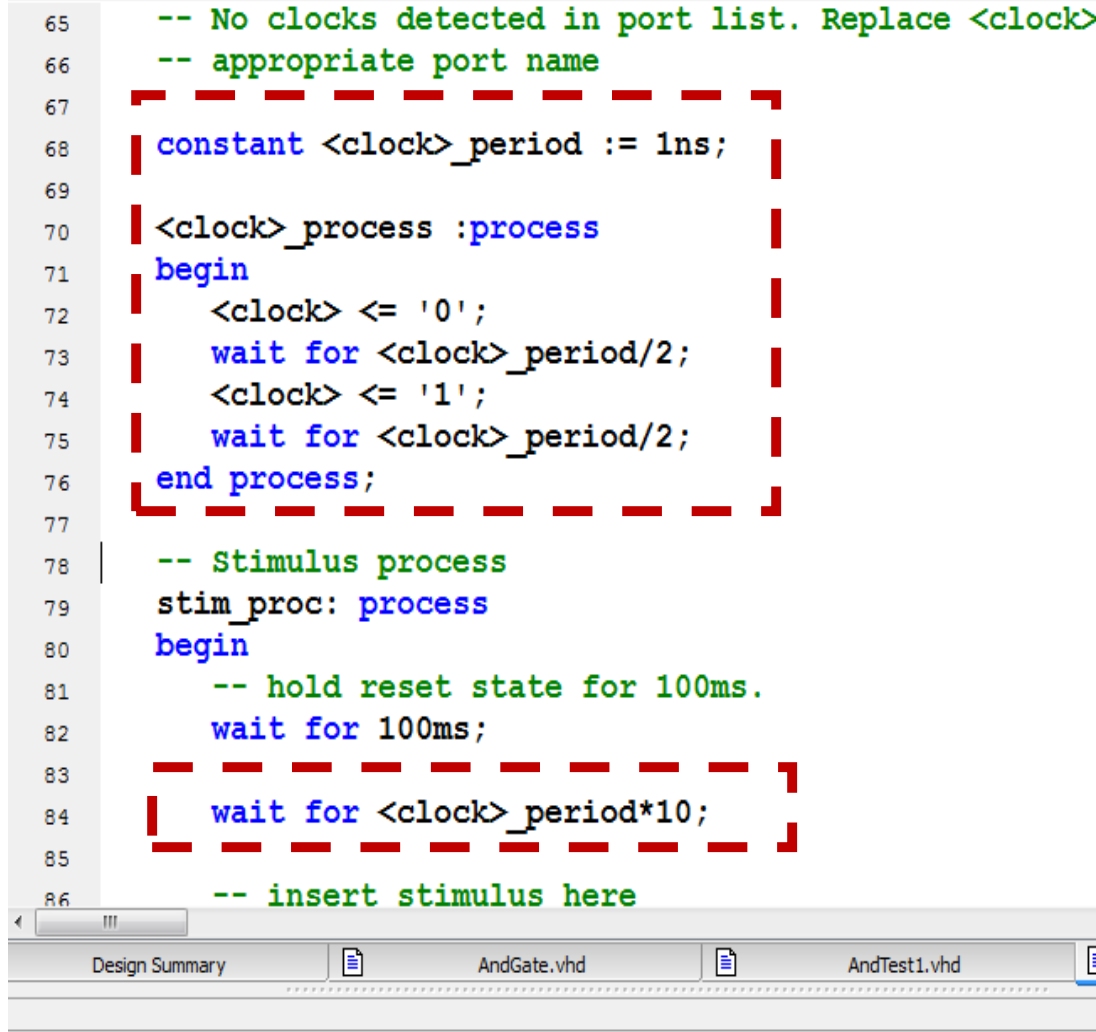
A wizard will pop-up, select “VHDL Test Bench” and write a file name. Then, click Next till the end of the wizard



SIMULATION GETTING STARTED(3)

Delete the parts of the code that have “**clock**” because our circuit is a simple combinational circuit.

```
65 -- No clocks detected in port list. Replace <clock>
66 -- appropriate port name
67
68 | constant <clock>_period := 1ns;
69
70 | <clock>_process :process
71 | begin
72 |     <clock> <= '0';
73 |     wait for <clock>_period/2;
74 |     <clock> <= '1';
75 |     wait for <clock>_period/2;
76 | end process;
77
78 -- Stimulus process
79 stim_proc: process
80 begin
81     -- hold reset state for 100ms.
82     wait for 100ms;
83
84 | wait for <clock>_period*10;
85
86 -- insert stimulus here
```



SIMULATION GETTING STARTED(4)

Insert your Test code in the highlighted part

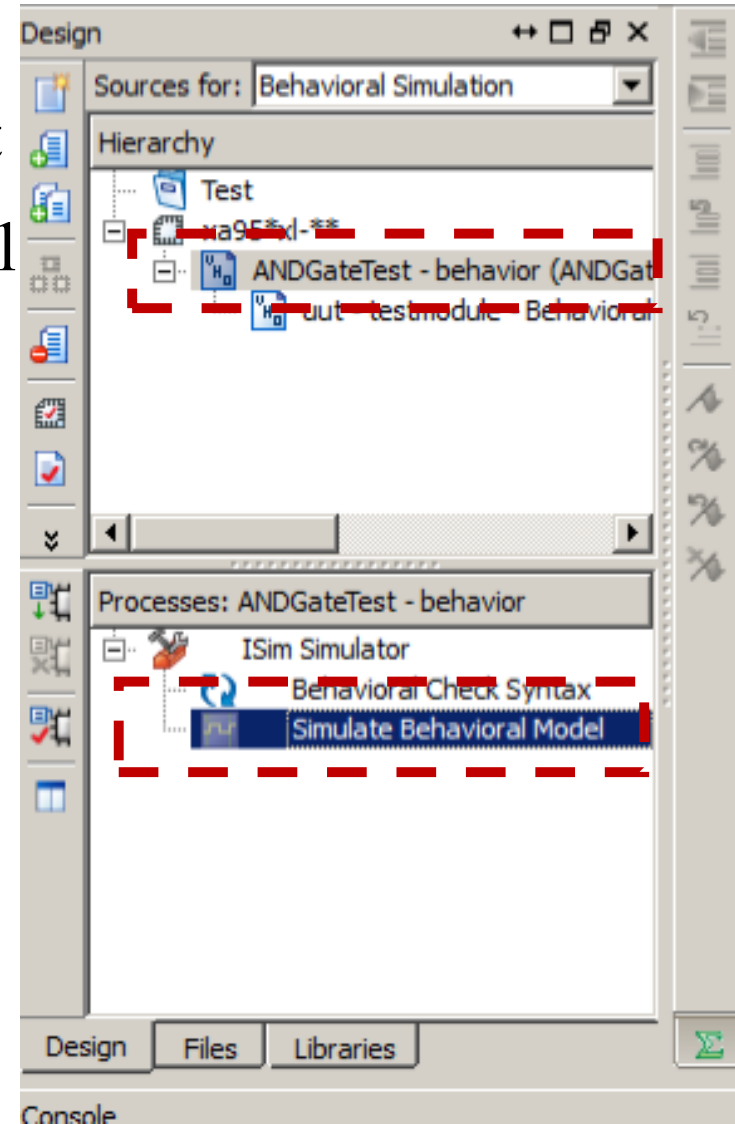
```
54     signal c : std_logic;
55
56 BEGIN
57
58     -- Instantiate the Unit Under Test (UUT)
59     uut: testmodule PORT MAP (
60         a => a,
61         b => b,
62         c => c
63     );
64
65     -- Stimulus process
66     stim_proc: process
67     begin
68         -- hold reset state for 100ms.
69         wait for 100ms;
70         -- insert stimulus here
71
72         wait;
73     end process;
74
75 END;
76
```

TEST CODE

```
stim_proc: process begin
    -- hold reset state for 100ms.
    wait for 0ns;
    a <= '1';
    b <= '0';
    wait for 100ns;
    a <= '1';
    b <= '1';
    wait for 100ns;
    wait;
end process;
```

RUN SIMULATION

1. Highlight the “behavior” link first
2. double click “Simulate Behavioral Model”



SIMULATION

Click on "Fit to Screen" button

The screenshot shows a simulation software interface. On the left, there is a table titled "Simulation Objects for andgatetest" with the following data:

Object Name	Value
a	1
b	1
c	1

In the center, there is a table with the following data:

Name	Value
a	1
b	1
c	1

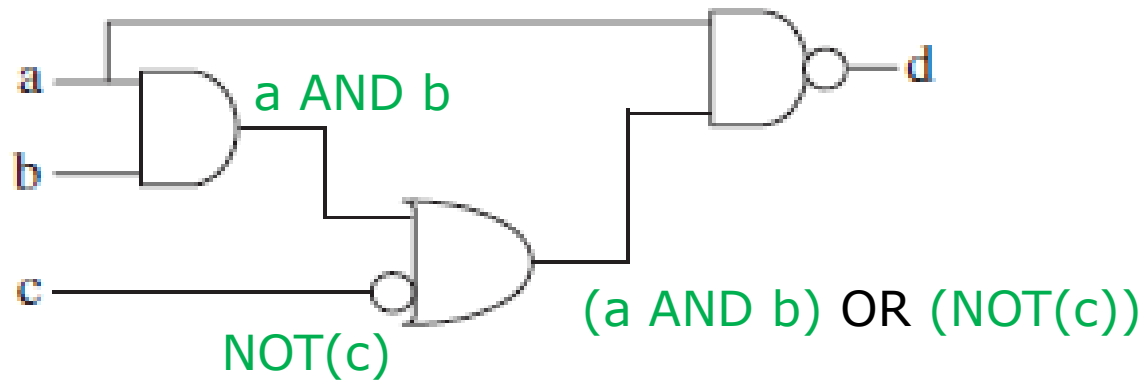
On the right, there is a timing diagram showing three green signals. The x-axis is labeled with "0 ns" and "500 ns". A scale bar at the top right indicates "1 000 ns". A tooltip at the bottom of the diagram shows "X1: 1 000 ns".

A red box highlights the "Fit to Screen" button in the toolbar, which is represented by a blue icon of a screen with four arrows pointing outwards. An arrow points from the text "Click on 'Fit to Screen' button" to this button.

EXERCISE: COMBINATIONAL CIRCUIT

Modify the AND gate to be the following logical expression

$$a \text{ AND } ((a \text{ AND } b) \text{ OR } (\text{NOT}(c)))$$



Solution:

$$d \leq a \text{ AND } ((a \text{ AND } b) \text{ OR } (\text{NOT}(c)))$$

DATA TYPES

We can define **SIGNAL** within the architecture to store intermediate values

Examples for data types:

- **BIT** (and **BIT_VECTOR**): 2-level logic ('0', '1').
- **STD_LOGIC** (**STD_LOGIC_VECTOR**): 8-valued logic system
- **BOOLEAN**: True, False.
- **NATURAL**: Non-negative integers

You can also use these data types to define **PORT** in **ENTITY** definition

DATA TYPES

BIT (and BIT_VECTOR): 2-level logic ('0', '1').

Examples:

SIGNAL x: BIT;

SIGNAL y: BIT_VECTOR (3 DOWNTO 0);

--y is a 4-bit vector, leftmost bit being the MSB.



SIGNAL w: BIT_VECTOR (0 TO 3);

--w is a 4-bit vector, rightmost bit being the MSB.



DATA TYPES

STD_LOGIC (STD_LOGIC_VECTOR): 8-valued

'X'	Forcing Unknown	(synthesizable unknown)
'0'	Forcing Low	(synthesizable logic '1')
'1'	Forcing High	(synthesizable logic '0')
'Z'	High impedance	(synthesizable tri-state buffer)
'W'	Weak unknown	
'L'	Weak low	
'H'	Weak high	
'_'	Don't care	

HANDS-ON2: SHIFTER CIRCUIT

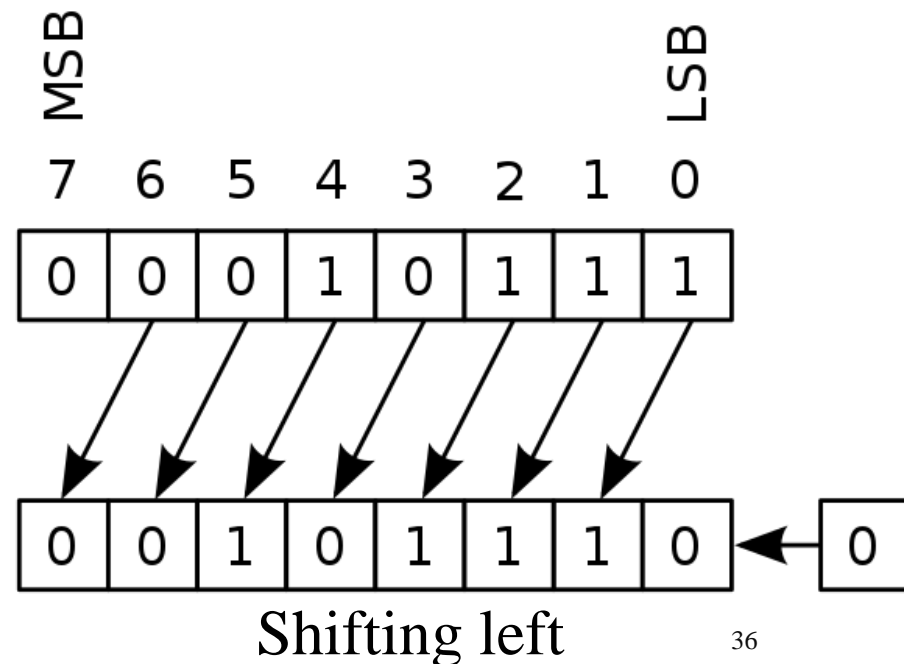
A shifter is a combinational circuit with one or more inputs and an equal number of outputs. The outputs are shifted with respect to the inputs.

Ex.: Left shifter circuit:

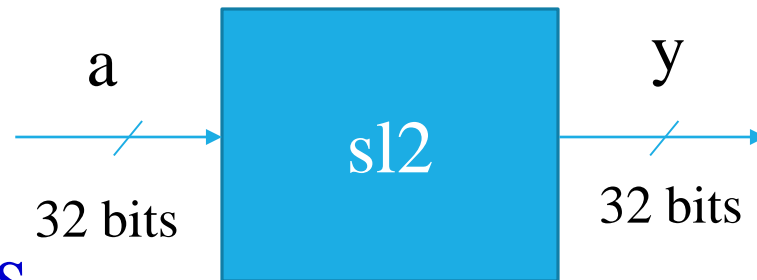
To multiply by powers of 2

In (8)	Out (8)
00000001	00000010
00000010	00000100
01010101	10101010

Out = IN(6-0) & “0”



HANDS-ON2: SHIFTER CIRCUIT



Shift left by 2
(multiply by 4)

```
ENTITY s12 IS
```

```
PORT(a: IN STD_LOGIC_VECTOR(31 downto 0);
```

```
  y: OUT STD_LOGIC_VECTOR(31 downto 0);
```

```
END s12;
```

```
ARCHITECTURE myarch OF s12 IS
```

```
BEGIN
```

```
y <= a(29 downto 0) & "00";
```

```
END myarch;
```

WHEN STATEMENT

WHEN/ELSE statement allows selecting the signal value according to specific conditions

For example,

```
ARCHITECTURE myarch OF myentity IS
```

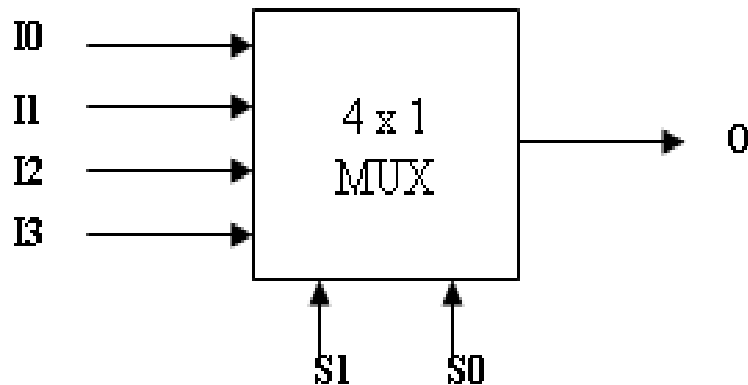
```
BEGIN
```

```
  y <=    a WHEN sel="00" ELSE  
          b WHEN sel="01" ELSE  
          c WHEN sel="10" ELSE  
          d;
```

```
END myarch;
```

MULTIPLEXER ($2^n \times 1$)

- Select one of the input 2^n to be produced at the output line
- Selection is done according to the selection lines (n lines)
- Each input line might be (single line) or (multiple lines = bus)



S1	S0	O
0	0	I0
0	1	I1
1	0	I2
1	1	I3

MULTIPLEXER (4 X 1) (1)

Multiplexer Implementation

- Structural:

$$O = I_0S_1'S_0' + I_1S_1'S_0 + I_2S_1S_0' + I_3S_1S_0$$

- Behavioral:

$O \leq I_0$ when $S_1S_0 = \text{“00”}$ else
 I_1 when $S_1S_0 = \text{“01”}$ else
 I_2 when $S_1S_0 = \text{“10”}$ else
 I_3 when $S_1S_0 = \text{“11”}$ else
'Z';

MULTIPLEXER (4 X 1) (4)

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY Mux IS
PORT(I3:      IN STD_LOGIC;
      I2:      IN STD_LOGIC;
      I1:      IN STD_LOGIC;
      I0:      IN STD_LOGIC;
      S:      IN STD_LOGIC_VECTOR ( 1 downto 0);
      O:      OUT STD_LOGIC);
END Mux;
```

```
ARCHITECTURE MuxArch OF Mux IS
BEGIN
  O <= I0 WHEN (S="00") ELSE
      I1 WHEN (S="01") ELSE
      I2 WHEN (S="10") ELSE
      I3 WHEN (S="11") ELSE
      'Z';
END MuxArch;
```

;

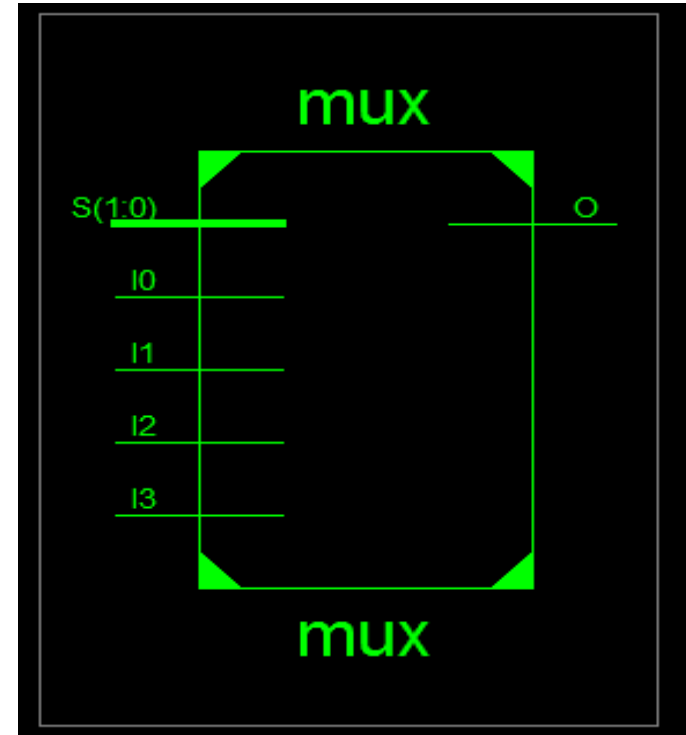
MULTIPLEXER (4 X 1) (4)

Let's see circuit RTL & the consumed instances.

This circuit consumes

- 1 Multiplexers 4*1

The same functionality, but with appropriate instances



MULTIPLIER (3BIT 4×1) USING BUS

ENTITY Mux IS

```
PORT(  I3:    IN STD_LOGIC_VECTOR(2 DOWNTO 0);
        I2:    IN STD_LOGIC_VECTOR(2 DOWNTO 0);
        I1:    IN STD_LOGIC_VECTOR(2 DOWNTO 0);
        I0:    IN STD_LOGIC_VECTOR(2 DOWNTO 0);
        S:     IN STD_LOGIC_VECTOR(1 DOWNTO 0);
        O:     OUT STD_LOGIC_VECTOR(2 DOWNTO 0) );
```

END Mux;

ARCHITECTURE MuxArch OF Mux IS

BEGIN

```
    O <=    I0 WHEN S="00" ELSE
            I1 WHEN S="01" ELSE
            I2 WHEN S="10" ELSE
            I3 WHEN S="11" ELSE
            "ZZZ";
```

END MuxArch;

MULTIPLEXER (n BIT 4×1)_{USING BUS}

ENTITY Mux IS

Generic (n: integer := 3);

```
PORT(  I3:      IN STD_LOGIC_VECTOR (n-1 DOWNT0 0);
        I2:      IN STD_LOGIC_VECTOR (n-1 DOWNT0 0);
        I1:      IN STD_LOGIC_VECTOR (n-1 DOWNT0 0);
        I0:      IN STD_LOGIC_VECTOR (n-1 DOWNT0 0);
        S:       IN STD_LOGIC_VECTOR (1 DOWNT0 0);
        O:       OUT STD_LOGIC_VECTOR (n-1 DOWNT0 0) );
```

END Mux;

ARCHITECTURE MuxArch OF Mux IS

BEGIN

```
    O <=      I0 WHEN S="00" ELSE
              I1 WHEN S="01" ELSE
              I2 WHEN S="10" ELSE
              I3 WHEN S="11" ELSE
              "ZZZ";
```

END MuxArch;

HANDS-ON3: MULTIPLEXER (n BIT 2×1)

Default value of n is 8 bits

```
ENTITY Mux2 IS
```

```
Generic (n: integer := 8);
```

```
PORT(I0,I1: IN STD_LOGIC_VECTOR(n-1 DOWNTO 0);  
      S:     IN STD_LOGIC ;  
      y:     OUT STD_LOGIC_VECTOR(n-1 DOWNTO 0));  
END Mux;
```

```
ARCHITECTURE MuxArch OF Mux2 IS
```

```
BEGIN
```

```
    y <=      I1 WHEN S ELSE I0;
```

```
END MuxArch;
```



Thanks