I. Starting ISE Software in Linux

Before the first start of ISE in Linux, you should set your profile to modify the environment variables such that ISE can be run after every next log in to your account. To do that, please follow the steps below:

1. Open terminal
2. Use command **gedit ~/.profile** to open profile**.**
3. Add **. /vol/share/software/Xilinx/14.7-profile** at the end of file. Please note that there is a space between "." and "/".
4. Close the file, logout and login again.

Then to start ISE, run the following command in terminal:

- run_ise

Notes:

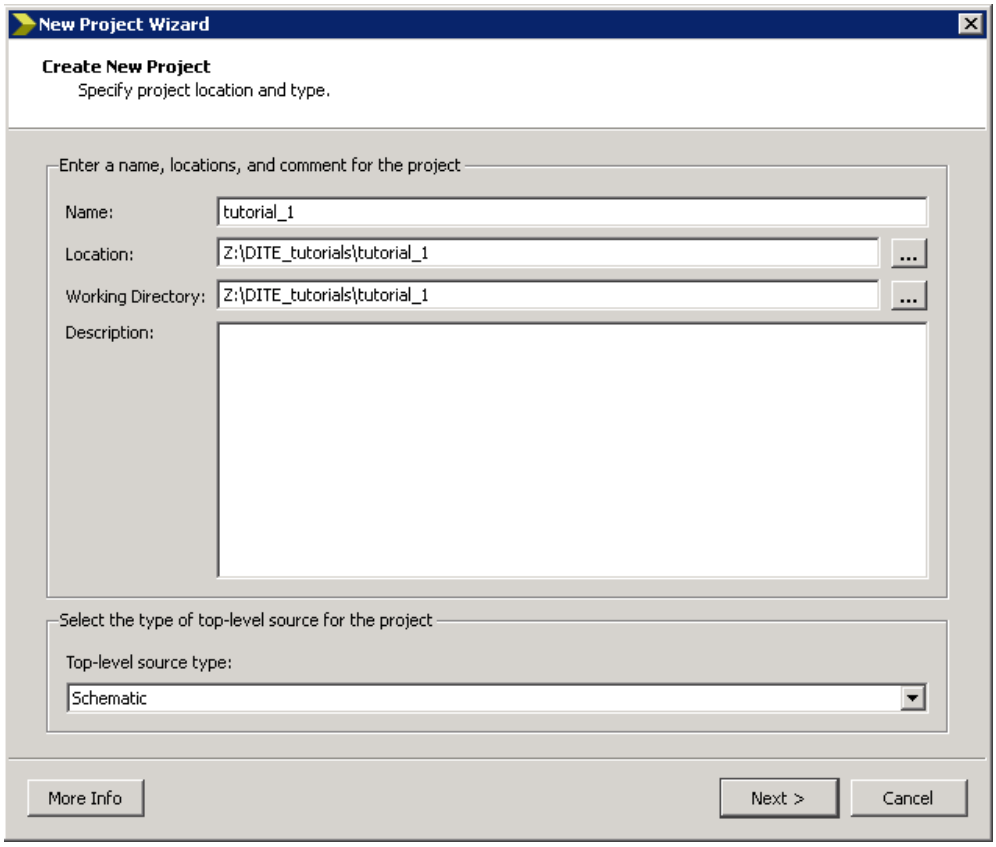
- If Xilinx License Error Message Box appears, **click > OK**.
- If Xilinx License Configuration Manager appears **click > Close**.

II. Create a New Project

To create a new ISE project select **File > New Project**. The page *Create New Project* appears.

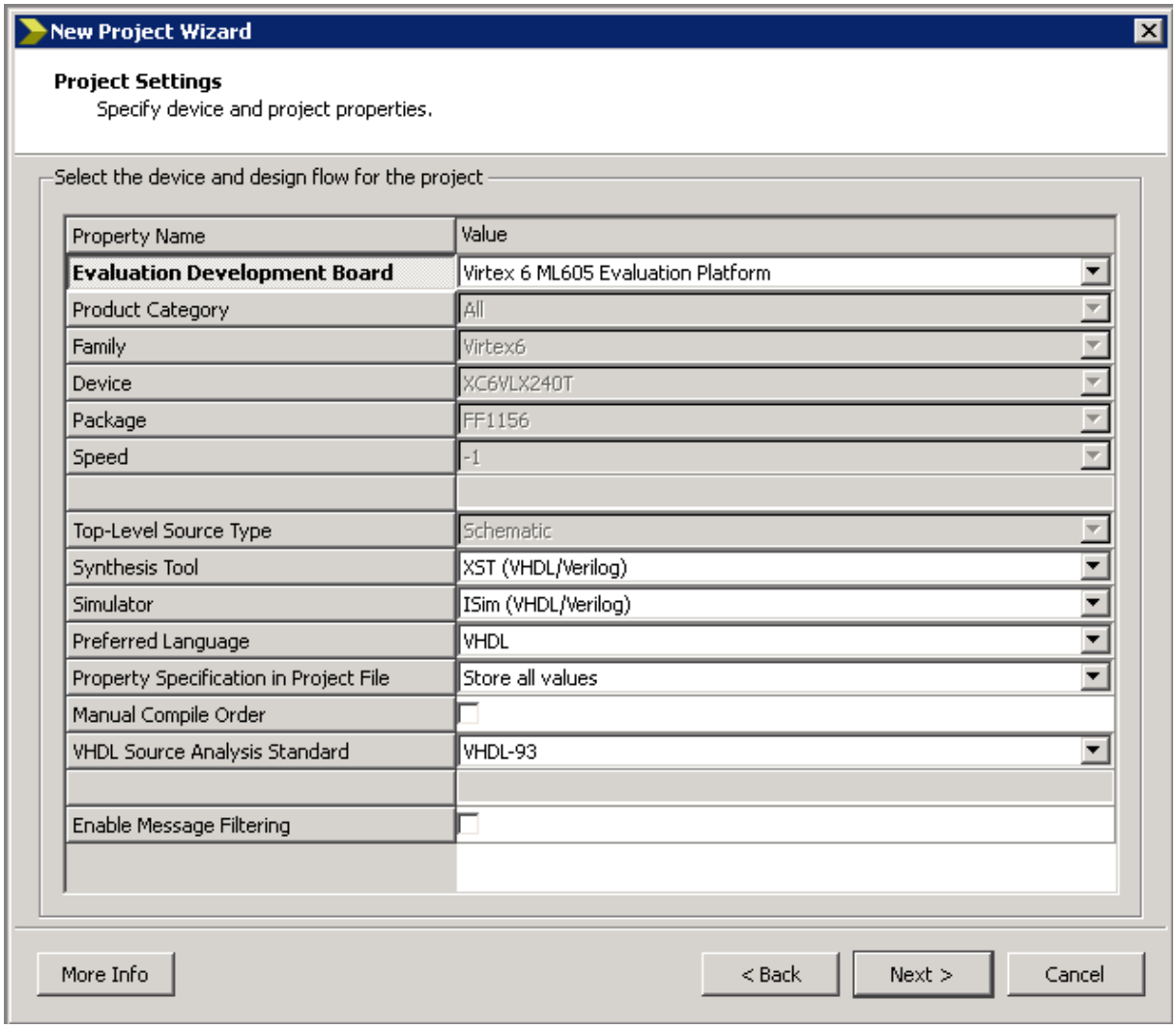
A Create New Project page

1. In the field Project ***Name***, type **tutorial_1**.
   - You can choose another name that does not contain any white spaces.
2. In the field Project ***Location***, browse to a location (a directory under your **home** directory) for the new project.
   - Note that: A **tutorial_1** subdirectory is created automatically.
3. In the field ***Top-level source type*****, select > *Schematic***.
4. **Click > *Next*** to move to the page ***Project Settings***.



B Project Settings page

1. In the field ***Evaluation Development Board,*** **select >** ***Virtex 6 ML605 Evaluation Platform***.
2. In the field ***Simulator*****, select >** ***ISim(VHDL/Verilog)***.
3. In the field ***Preferred Language*****, select >** ***VHDL***.
4. **Click >** ***Next*** to move to the page ***Project Summary***.
5. **Click >** ***Finish*** in the page ***Project Summary***.

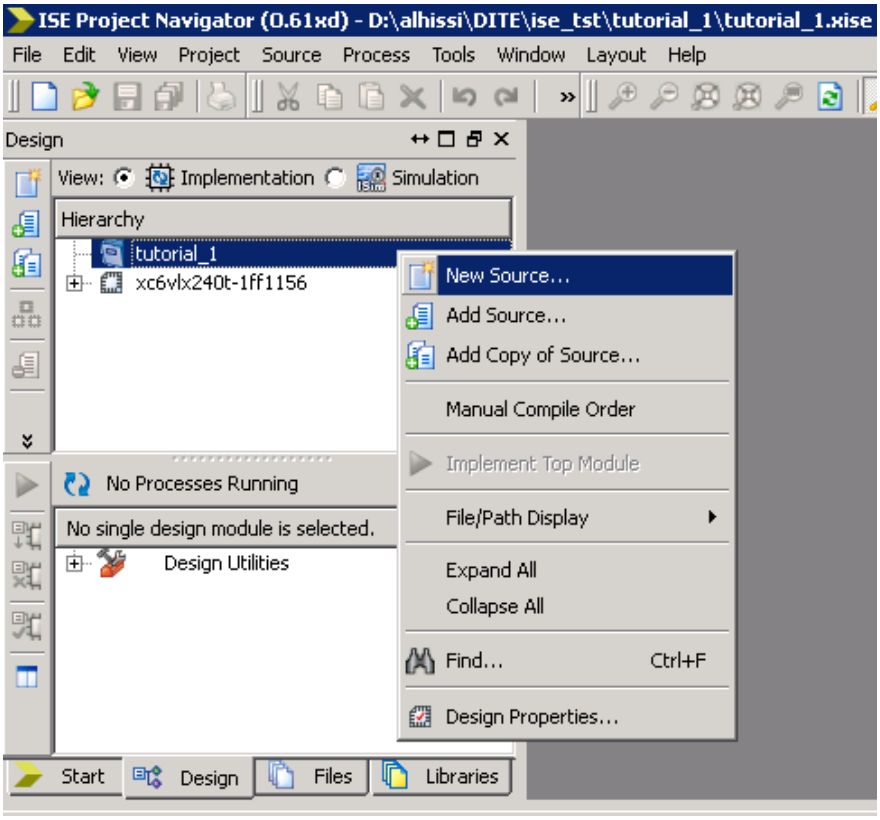


III. Create a New Design

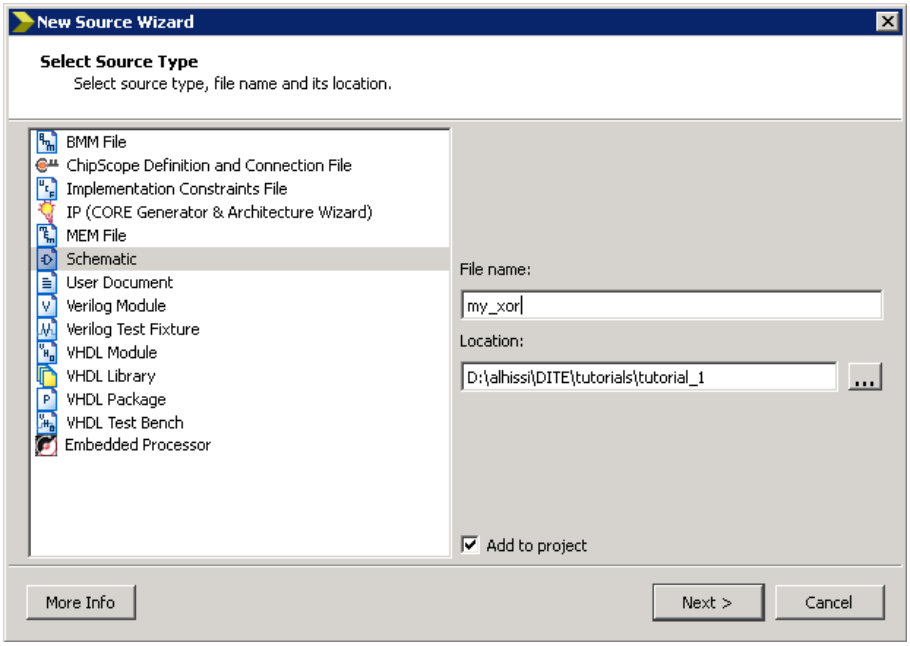
To study how to create a new design, we will design in this section a 2-Input X-OR Gate. The X-OR Function is defined as: **Y = A1 xor B1 = A1'B1 + A1B1'**.

A Create a Schematic Source

1. In ***ISE Design Suite*** that appears on the left side of ISE, click on the ***Design*** tab to go to the ***Design Panel***.
2. In the ***Design Panel***, **right-click** on the icon ***tutorial_1*** and **select > *New Source*** to move to the page ***Select Source Type***.



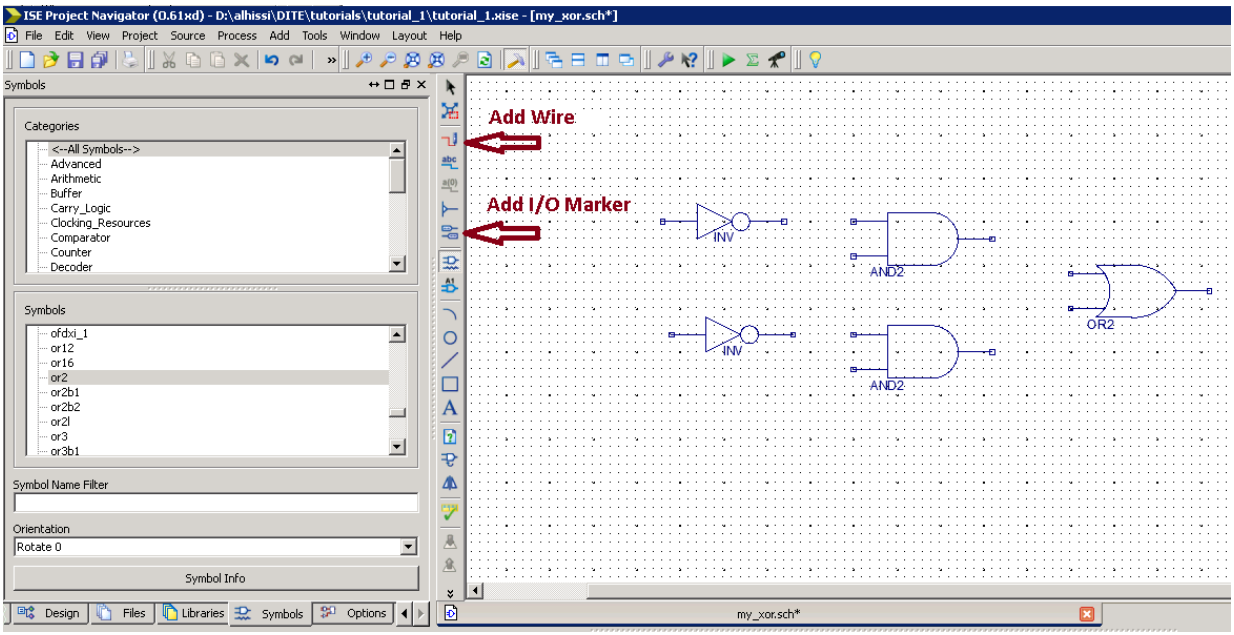
3. The page ***Select Source Type***
   i) In the field ***File Name***, type **my_xor**.
      - You can choose another name that does not contain any white spaces.
   ii) From the Column at the left-side, select ***Schematic*** as a Source Type.
   iii) Tick the option > ***Add To Project***.
   iv) **Click > *Next*** to move to the page ***Project Summary***.



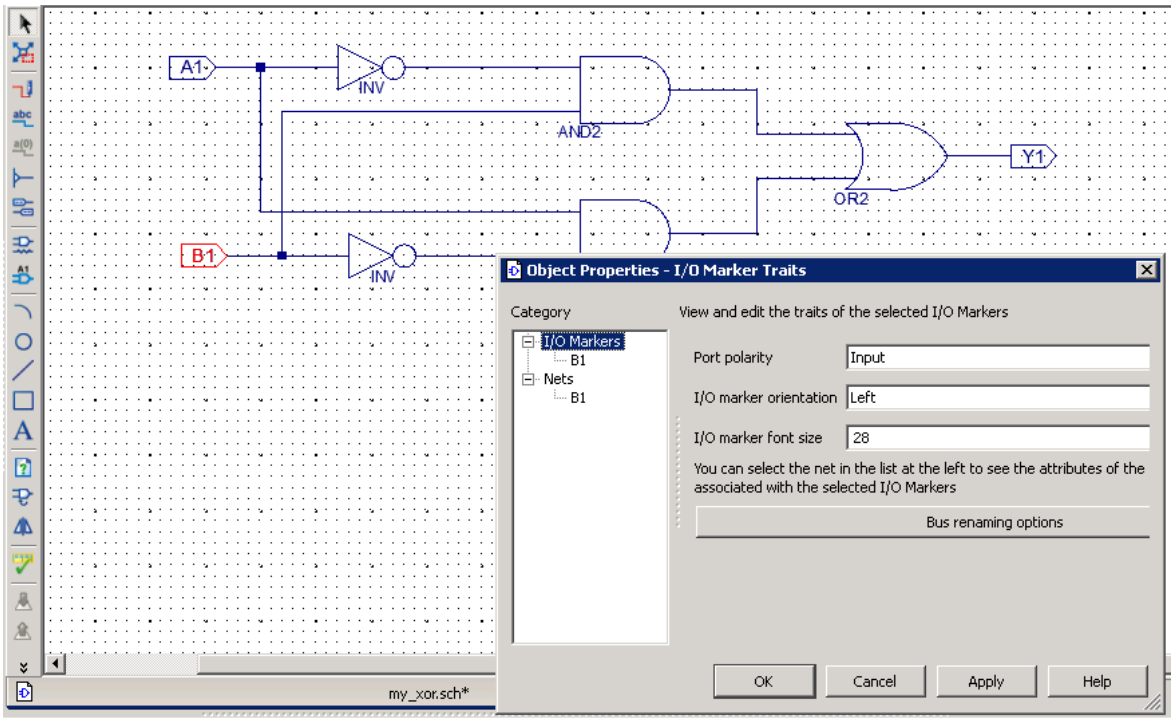
4. In the page **Project Summary** , ***click*** **> Finish**.
   - A Schematic source file "*my_xor.sch*" is added to the project.

B Edit the Schematic File

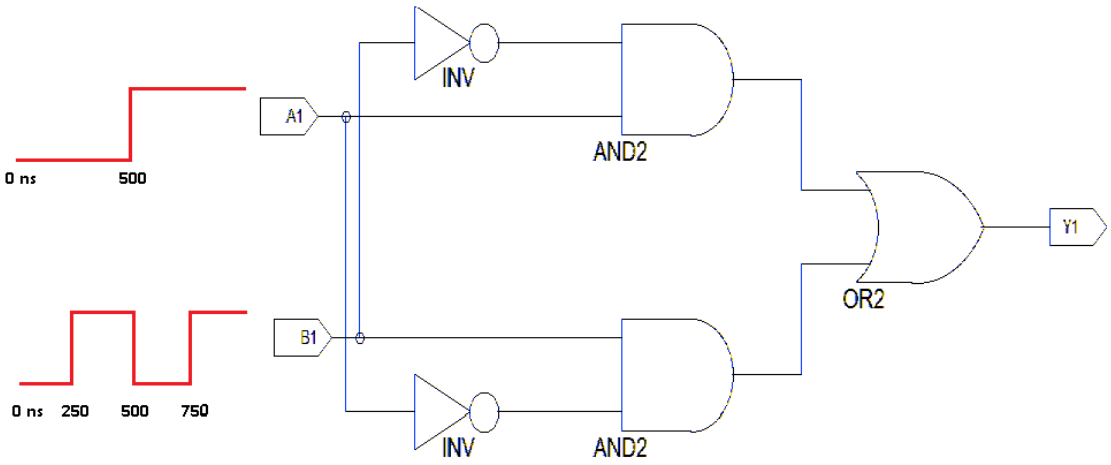
1. In ***ISE Design Suite,*** go to the ***Design Panel*** and open the source file ***my_xor.sch*** by double clicking it.
2. In ***ISE Design Suite***, go to the ***Symbols Panel***.
3. From the alphabetically ordered symbols appear in the ***Symbols panel*** select the required symbols for our design, and add them to the schematic file.
   - The required Symbols are (**Two 2-Input And gates, Two 1-Input Inverters, and One 2-Input Or gates**).
4. To connect the gates in Schematic file, **Select > *Add* > *Wire*** and use the wires to draw the connections.
   - You might need to **Zoom-In** the Schematic file to be able to connect the gates.
5. To connect the Input / Output Ports to our design, **select *Add* > *I/O Marker*** and connect two ports to the input and one to the output.
   - Both Add > Wire and Add I/O Marker can be found in the panel of the icons appears at the left of the schematic.
6. Rename the ***Port*** by **double clicking** it, selecting ***Nets*** and typing the required name, (**A1, B1, or Y1**).



7. To check the correctness of the Schematic, **select > *Tools* > *Check Schematic***.
   - This check figures out the mistakes such as floating pins or unconnected wires. However, It cannot figure out a faulty design.
8. **Save** the final schematic file “my_xor.sch” which contains the final design.



IV. Create a New Test Bench



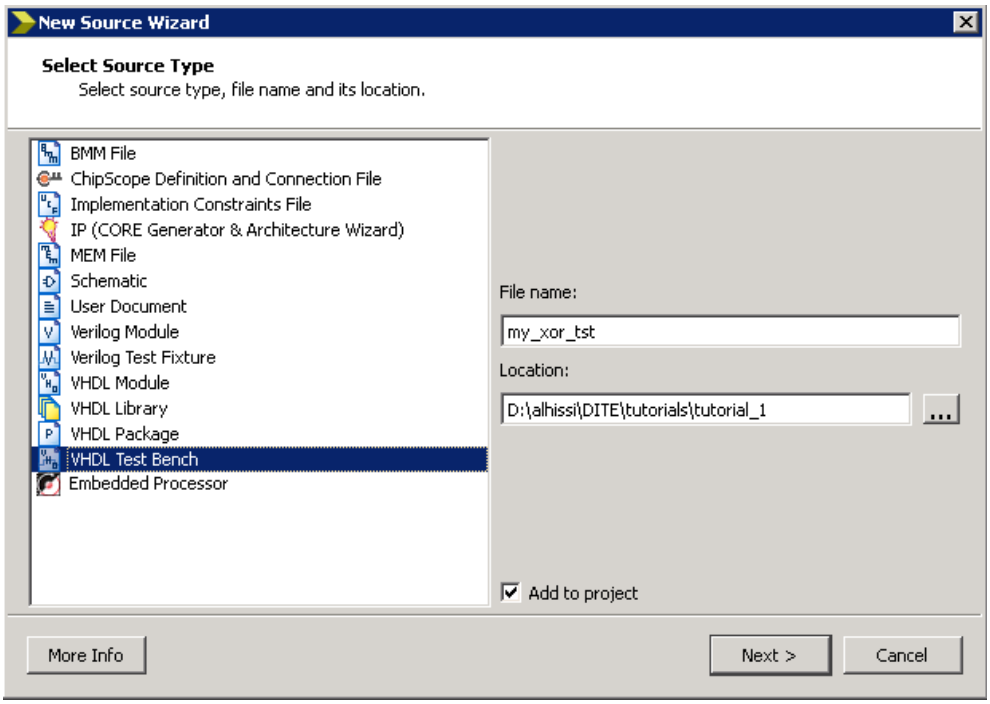
To verify the correctness behavior of our design, we need to simulate it. However, before the simulation, we have to create a test bench that

stimulates the input ports of our design with different input values. We provide the set of stimuli to our design using a VHDL source file.

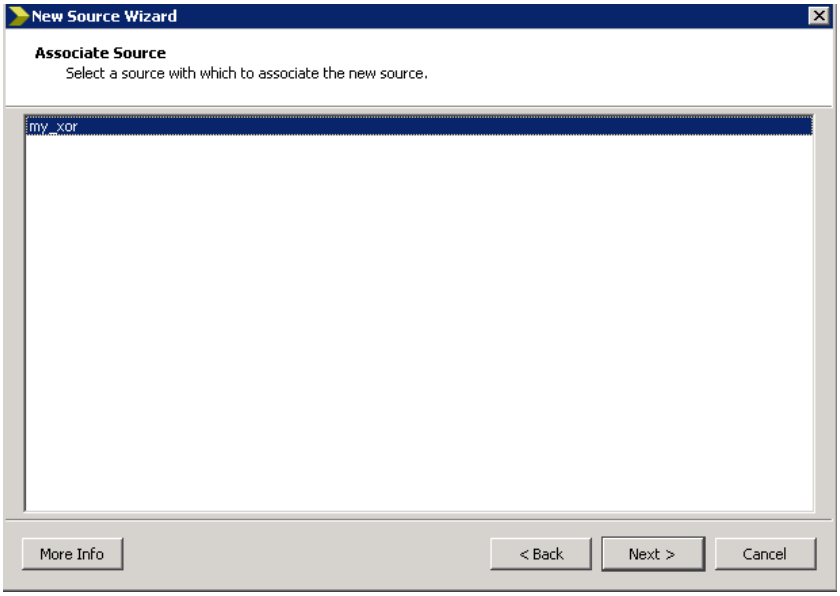
A Create a VHDL Source

1. In ***ISE Design Suite***, appears on the left side, go to the ***Design Panel***.
2. In the ***Design Panel***, **right-click** on the icon ***tutorial_1*** and **select > *New Source*** to move to the page ***Select Source Type***.
3. The page ***Select Source Type Page***
   i) In the field ***File Name***, type **my_xor_tst**.
   - You can choose another name that does not contain any white spaces.

   ii) From the Column at the left-side, select ***VHDL Test Bench*** as a Source Type.
   iii) Tick the option **> *Add To Project***.
   iv) **Click > *Next*** to move to the page ***Associate Source***.

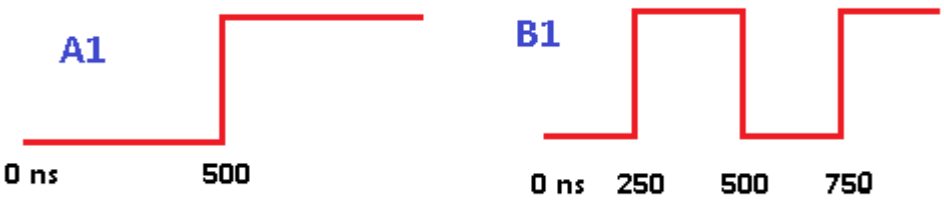


4. The page ***Associate Source***
   i) **Select > *my_xor*** as the source with which we want to associate our test bench.
   ii) **Click > *Next*** to move to the page ***Project Summary***.
5. In the page ***Project Summary* , click > *Finish***.
   - A VHDL source file ***my_xor_tst*** is added to the project.



B Edit the VHDL File

The VHDL File should be edited to stimulate the following inputs:



1. Open the file ***my_xor_tst.vhd*** and move to the following section at the end of the file:

```
-- *** Test Bench - User Defined Section ***
  tb : PROCESS
  BEGIN
    WAIT; -- will wait forever
  END PROCESS;
-- *** End Test Bench - User Defined Section ***
```

2. Before wait statement, add the following commands:

   A1 <= ‘0’, ‘1’ after 500 ns;

   B1 <= ‘0’, ‘1’ after 250 ns, ‘0’ after 500 ns, ‘1’ after 750 ns;

   ➢ So, this section should be modified to:

```
-- *** Test Bench - User Defined Section ***
  tb : PROCESS
  BEGIN
    A1 <= ‘0’, ‘1’ after 500 ns;
    B1 <= ‘0’, ‘1’ after 250 ns, ‘0’ after 500 ns, ‘1’ after 750 ns;
    WAIT; -- will wait forever
```

```
    END PROCESS;
  -- *** End Test Bench - User Defined Section ***
```

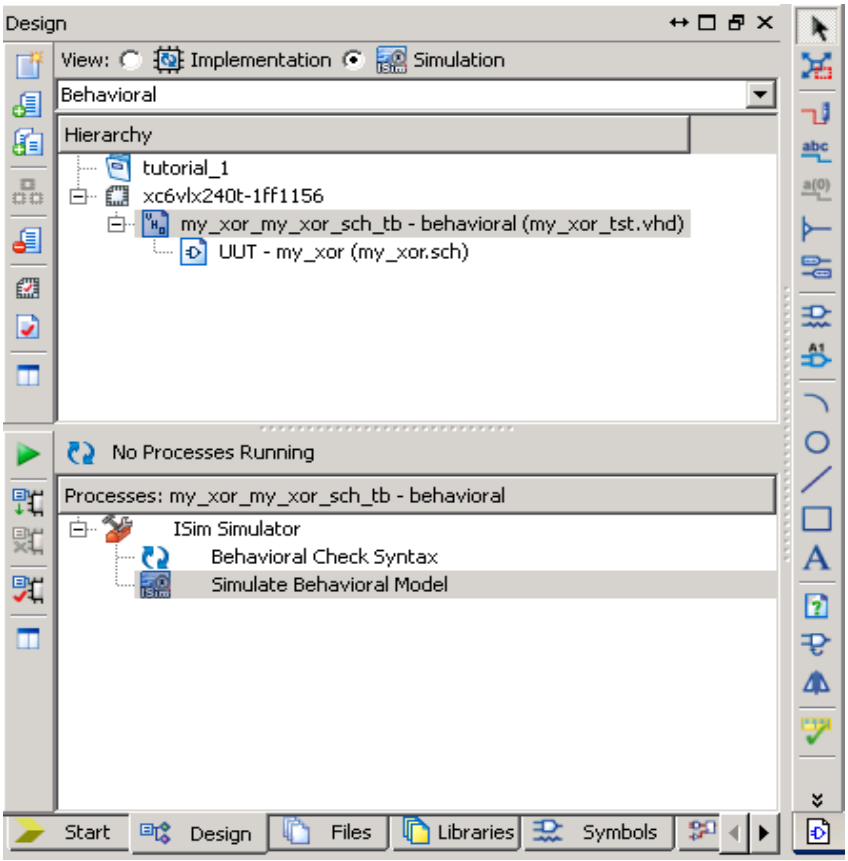
3. Save the Final VHDL file ***my_xor_tst.vhd*** which contains the final test bench.

V. Simulate our Design

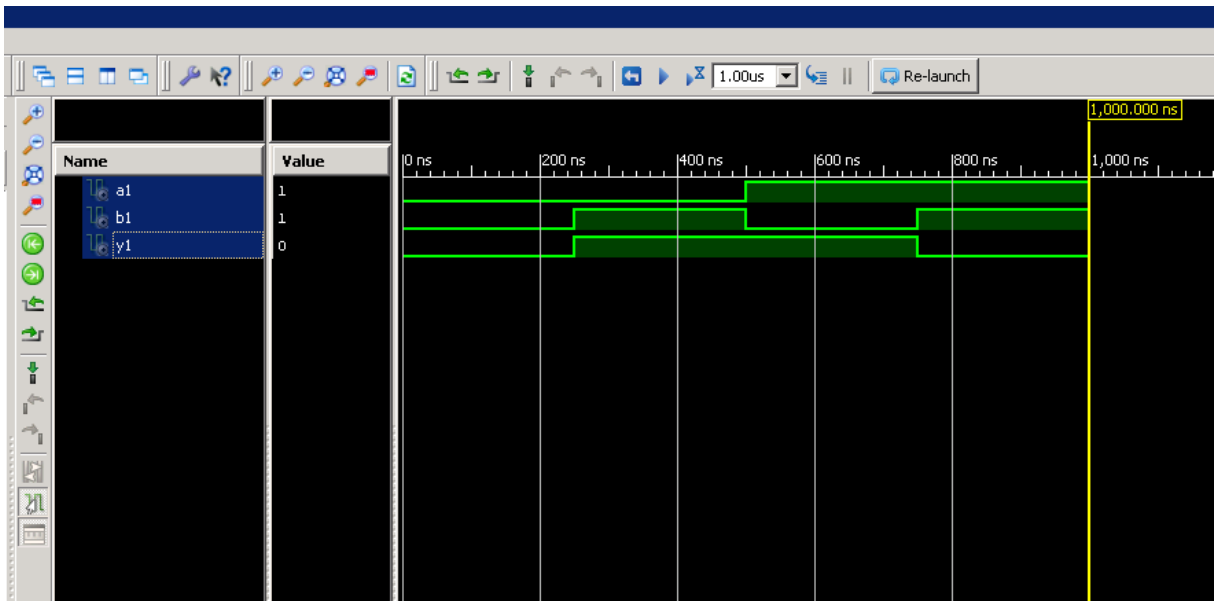
In this section, we will simulate our design to verify that it behaves as we expect. We will use the Integrated Simulator (**ISim**).

A ISIM

1. Open ***Design Panel***. In ***Design Panel View***, **select > *Simulation.***
2. In ***Design Panel > Hierarchy***, **select > *my_xor_tst.vhd***.
3. In ***Design Panel > Processes > ISim Simulator*, double click > *Simulate Behavioral Model*** to open the Integrated Simulator (**ISim**).



4. ISIM Window
    i) **Zoom-Out** to view the whole simulation time, the Default ***simulation time*** is **1000 ns**.
    ii) The Simulator shows the three ports **A1, B1 and Y1**.
    iii) Compare the value of Y1 with A1 and B1. **Y1** Should always equal to **A1 X-OR B1**.

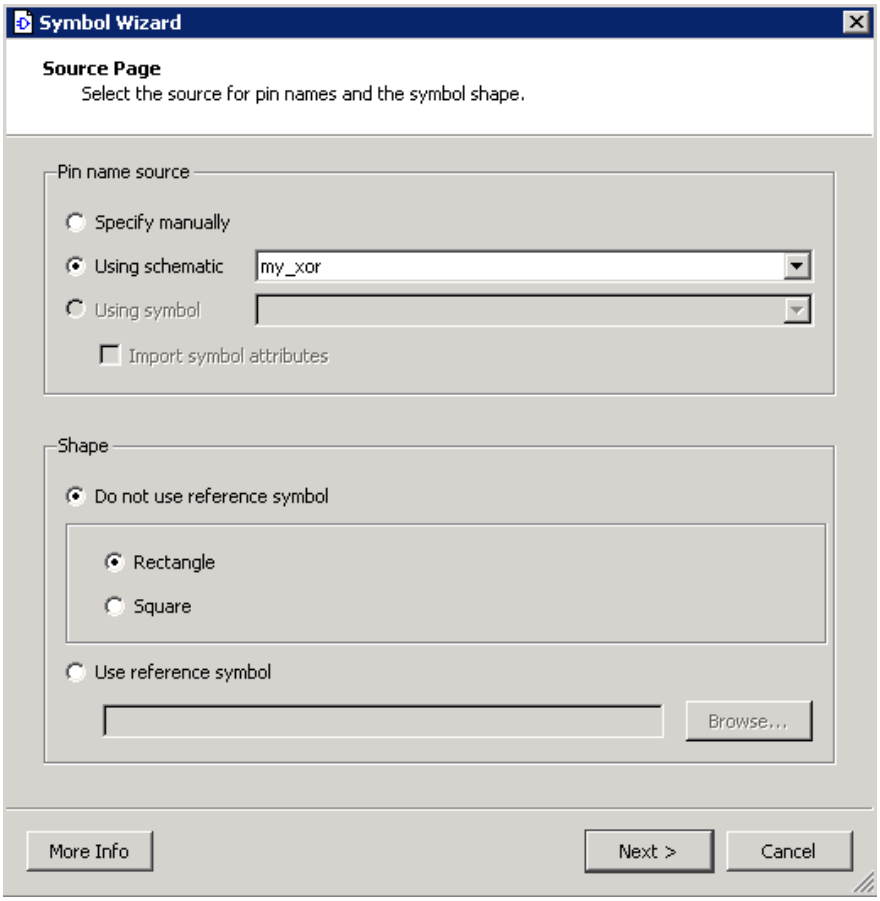


VI. Key Features

A Using Symbols

In order to build a modular well-structured and reusable design, it is good design-practice to use symbols. By using symbols, our small custom designs are stored as any standard symbol and can be reused to build larger designs.

1. Create our Own Symbol.
    i) Open the schematic file ***my_xor.sch***. **Select *Tools* > *Symbol Wizard***. The page ***Source Page*** appears.
    ii) In the page ***Source Page,*** in the field ***Pin name source***, **select > *Using Schematic*** and point to the schematic source file ***my_xor.sch***. **Select > *Next*** to move to the page ***Pin Page***.



iii) In the page ***Pin page***, keep the name of the symbol as the name of the schematic source file ***my_xor.sch***. **Select > *Next*** to move to the page ***layout Page***.

iv) In the page ***layout Page***, **Select > *Next*** again to move to the page ***Preview Page***.

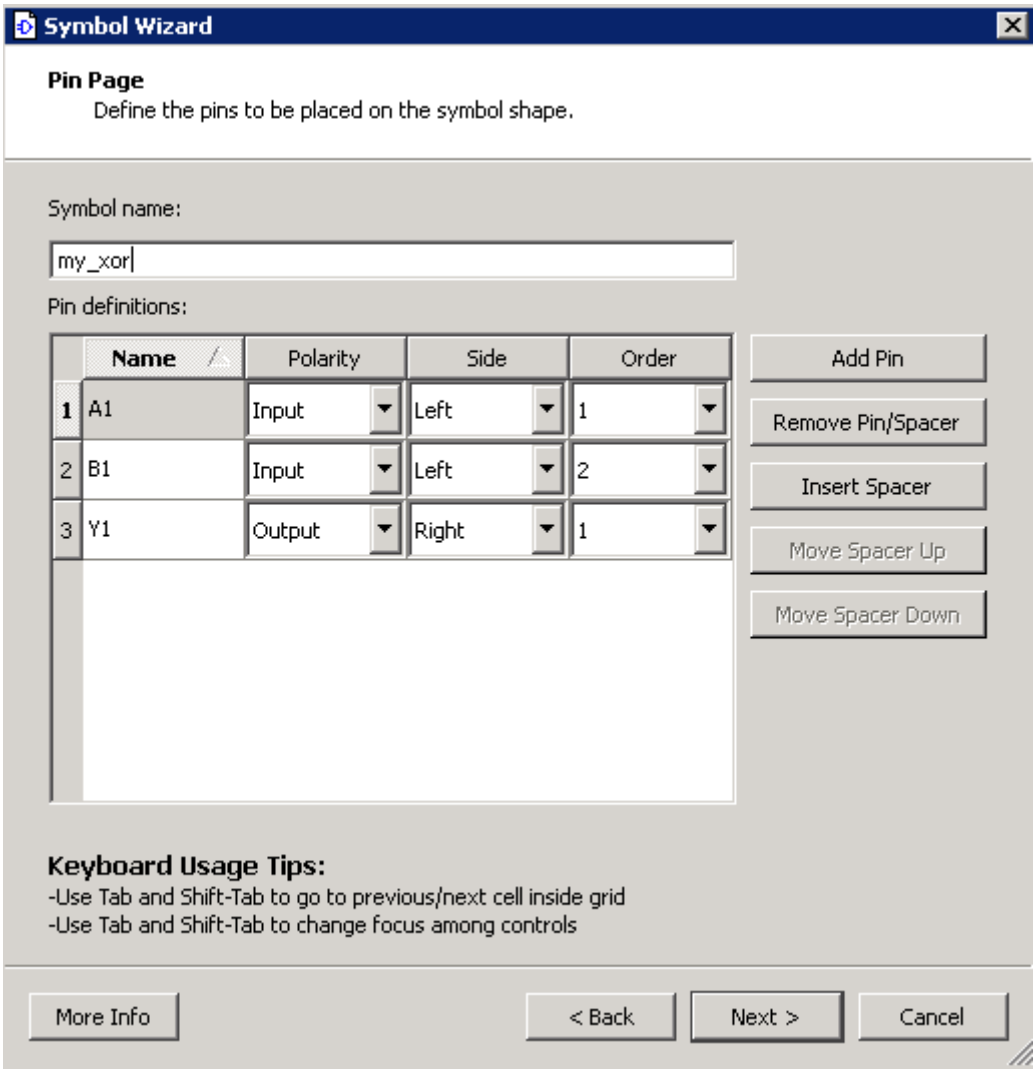
v) In the page ***Preview Page***, **select > *Finish***.

2. Reuse our created Symbol.

i) Create a new ***schematic source*** file as in Section II-A.

ii) Select the ***Symbols panel***. In the ***Symbols panel***, the new symbol ***my_X_OR*** appears and can be used as any other standard symbol.

- Never reuse the symbol ***my_X_OR*** in the original schematic, doing that creates a recursion that needs infinite hardware to be done. After creating the symbol ***my_X_OR***, it is a good design-practice not to change the original schematic ***my_X_OR.sch***.



B Using Buses

Buses are a convenient way to group related signals. This grouping produces a less cluttered, functionally clearer drawing. By grouping signals in a Bus, accessing the scalar signals that construct the bus is enabled by using Bus taps.

1. Creating a Bus

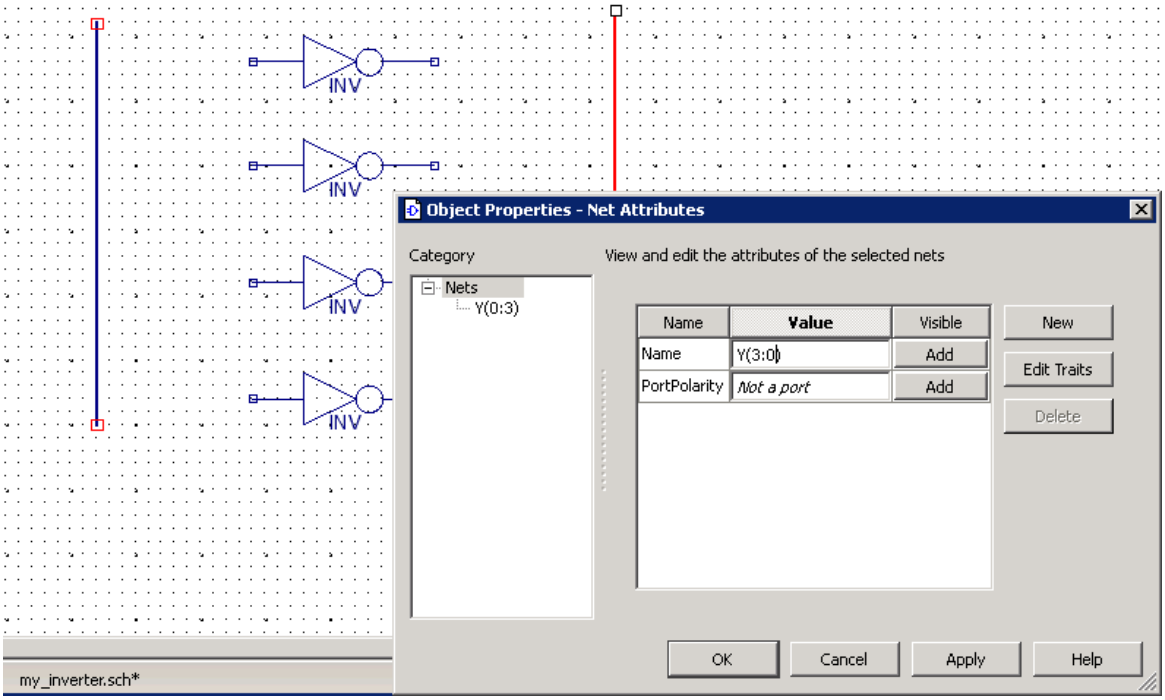
   We show in these steps how to create a **one 4-input inverter** using Four **1-Input Inverters**. We will connect the 4 input signals to one input Bus and the 4 output signals to one output bus.

   i) Create a new ***schematic source file*** as in Section II-A.
   ii) Using the ***Symbols panel***, add **Four 1-Input Inverters** to the new schematic.

iii) Use ***Add > Wire*** to add two floating wires to the schematic. These two wires will be the input and output buses.

iv) **Rename** the wires by **double clicking** it, **selecting > *Nets*** and changing the field ***Name***. To create a Bus, wire names must be in the format BusName(Starting Number:Ending Number) such as: **A(3:0)** and **Y(3:0)**.

- By changing a wire into a bus, it appears thicker than other wires.



v) Connect four Bus Tabs to each bus. ***Bus Tabs*** can be added by **selecting > *Add* > *Bus Tap***. These tabs give access to the four scalar signals in the bus.

- Bus tab could be rotated by changing the option at the Options panel.

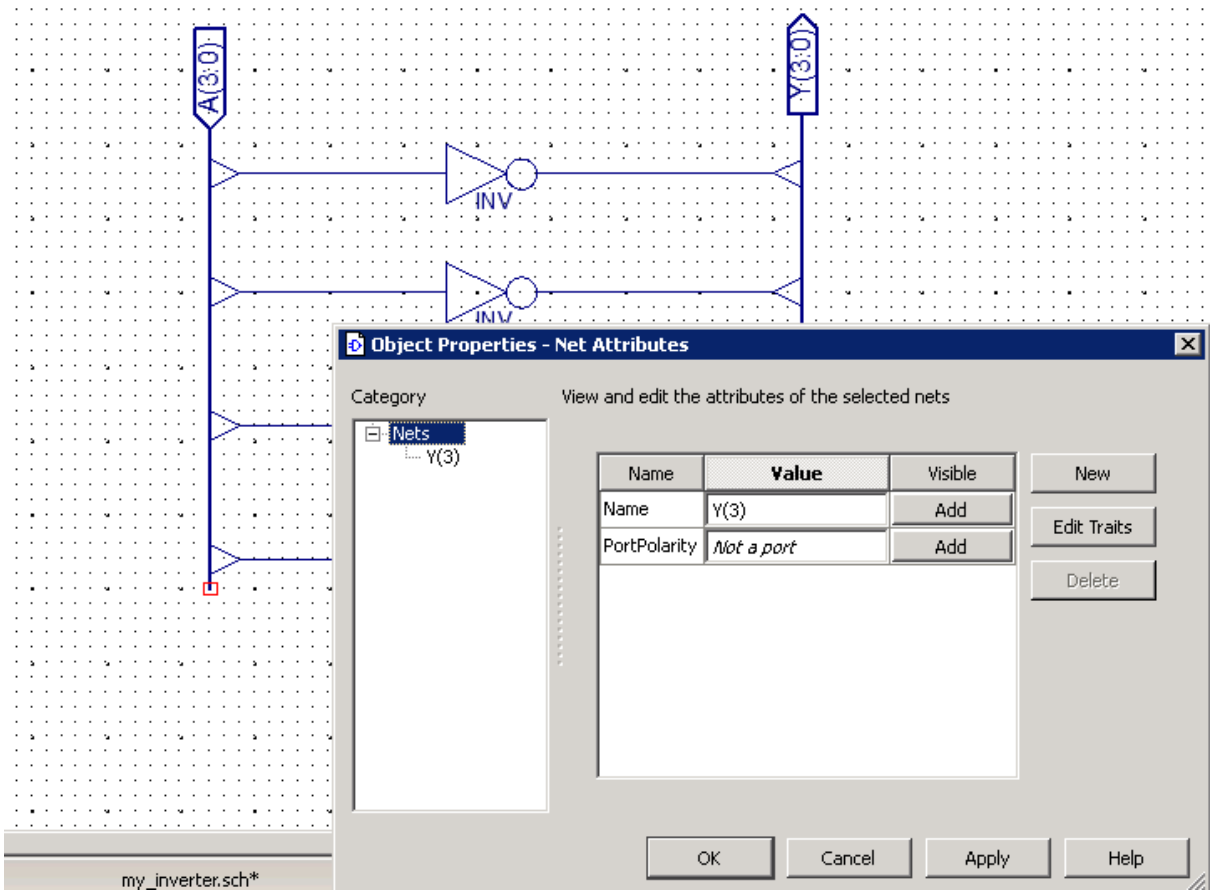
vi) **Select *Add* > *Wire*** and connect the input and output ports of the **Four 1-Input Inverters** to the bus tabs connected to the input and output buses.

vii) You must **rename** the connection wires to connect the ports of each **1-Input inverter** to one of the scalar signals inside the bus. To do that, **double-click** the connecting ***wire***, **select > *Nets*** and change the field ***Name*** to select one of the signals inside the bus, **A(3), A(2), A(1), A(0), Y(3), Y(2), Y(1) and Y(0)**.

viii) Connect ***I/O Markers*** to the two buses in our design.

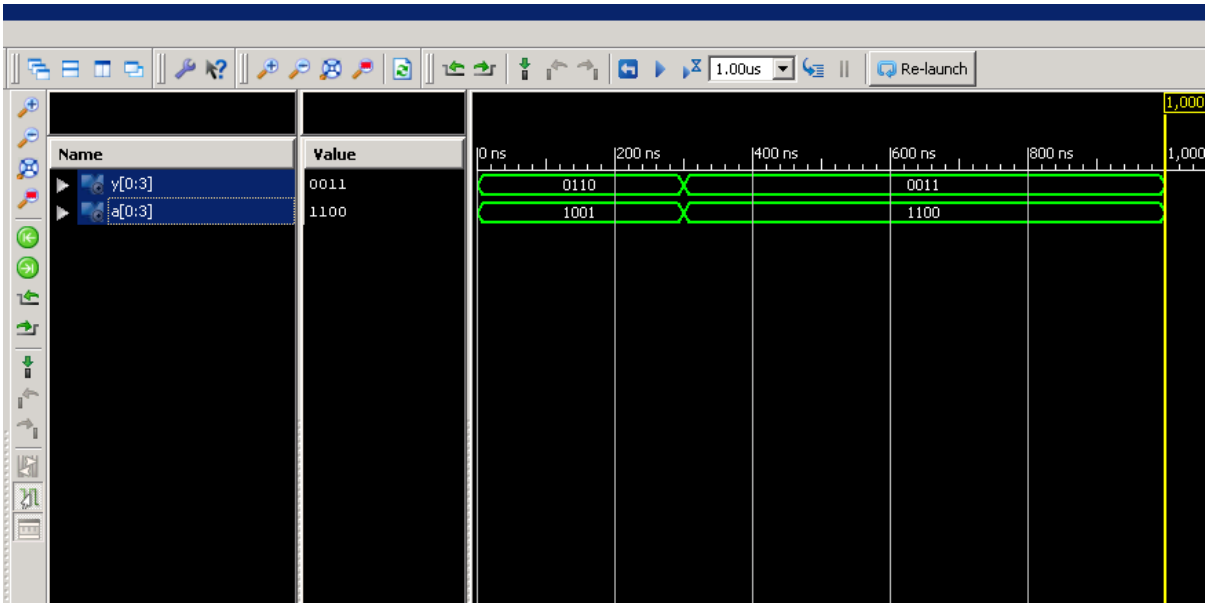
- The names of the markers appear in the same format as the bus PortNAME(3:0).

- Double click every ***I/O Marker*** and change (if needed) the field ***Port Polarity***. It should be ***Input*** for the input port **A(3:0)** and ***Output*** for the output port **Y(3:0)**.



ix) **Save** the design and create a test bench to test it.

x) In the VHDL test bench file, to assign a values to a bus you could use the assignment statement in the syntax **A<="0111"**; Instead of the four statements **A(3)<='0'**; **A(2)<='1'**; **A(1) <= '1'**; and **A(0) <= '1'**;.



C Generating Clock Signal

Clock signals are used as an input for sequential circuits. Thus, in order to simulate these circuits, VHDL test bench should generate this type of signals. In order to generate a clock signal in VHDL, a good design-practice is to dedicate a separate Process for the clock signal while keeping the stimuli for other signals in a different Process. Below, two ways are shown to generate a clock signal in VHDL.

1. Using **Sequential Process**

   The PROCESS ***clk_gen*** (Shown Below) generates 40 repetitions of a clock signal with a period 20 ns and duty cycle 50%. The other PROCESS ***tb*** is used to stimulate the other signals. Note that this way could be used to generate any customized repetition signal.

```
-- *** Test Bench - User Defined Section ***
clk_gen : PROCESS
BEGIN
        for i in 1 to 40 loop
                clk <= '1';
                wait for 10 ns;
                clk <= '0';
                wait for 10 ns;
        end loop;
        wait;
```

```
END PROCESS;

tb : PROCESS
BEGIN
        A1 <= '1', '0' after 300 ns;
        B1 <= '0', '1' after 600 ns;
        WAIT; -- will wait forever
END PROCESS;
-- *** End Test Bench - User Defined Section ***
```

2. Using **Single statement Process**

   The single statement, ***clk <= NOT clk after 10 ns;*** (Shown Below), can be used to generate infinite repetitions of a clock signal with a period of 20 ns and duty cycle of 50%. However using this statement requires initializing of the clock signal. The initialization has to be done during the signal instantiation as shown below. A separate PROCESS ***tb*** is used to stimulate the other signals.

```
SIGNAL clk   :        STD_LOGIC    := '1';  -- initialization
-- *** Test Bench - User Defined Section ***
        clk <= NOT clk after 10 ns;    -- single statement Process

tb : PROCESS
BEGIN
        A1 <= '1', '0' after 300 ns;
        B1 <= '0', '1' after 600 ns;
        WAIT;       -- will wait forever
END PROCESS;
-- *** End Test Bench - User Defined Section ***
```