# 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.

▶New Project Wizard		×
<b>Create New Project</b> Specify project loc	: cation and type.	
Enter a name, locati	ions, and comment for the project	
Name:	tutorial_1	
Location:	Z:\DITE_tutorials\tutorial_1	
Working Directory:	Z:\DITE_tutorials\tutorial_1	
Description:		
Select the type of to Top-level source typ Schematic	pp-level source for the project	
More Info	Next >	Cancel

### **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*.

alect the device and design flow for the p	project
Property Name	Value
Evaluation Development Board	Virtex 6 ML605 Evaluation Platform
Product Category	IA
Family	Virtex6
Device	XC6VLX240T
Package	FF1156
Speed	-1
Top-Level Source Type	Schematic
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	

### 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*.
- In the *Design Panel*, right-click on the icon *tutorial\_1* and select > *New Source* to move to the page *Select Source Type*.

<b>∑</b> I	5E Pro	oject N	avigato	r (0.61x	d) - D:\	alhis	ssi\D	ITE\is	se_t	st\tut	orial	_1\t	utorial	_1.xise
File	Edit	View	Project	Source	Proces	s T	ools	Winde	ow	Layout	He	lp		
	) 🦻	ÐÉ	16	] X [	ì È	×	ß	(24	»		P	ø	Ø /	2
Desig	IN					++	8	×						
ľ	View:	۰ 🔯	] Impleme	entation (	o 🔝 :	5imula	ation							
6	Hiera	irchy												
8		📑 tuto	orial_1	6644 <b>5</b> 7		-*	New	Source	a	_	_		1	
	<b>⊡</b> …	хсь	VIXZ4UC-1	m1156			Add	Source						
						i i i	Add	Convic	of So	urce				
							Manu	ual Con	npile	Order				
*	5	No Pro	cesses Ru	unning			Imple	ement	Top	Module				
- 96	No s	ingle de	sign mod	lule is sele	cted.		File/F	Path Di	ispla	У		•		
	÷۴	¥	Design Ut	ilities			Ехра	and All						
							Colla	pse All	I					
						A	Find.				Ctrl+	-F		
							Desi	gn Prop	perti	es				
>	) Start		Design	Fi Fi	les [	Li	brarie	s						

#### 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.

>New Source Wizard	×
Select Source Type Select source type, file name and its location.	
BMM File         ChipScope Definition and Connection File         Implementation Constraints File         IP (CORE Generator & Architecture Wizard)         MEM File         Schematic         User Document         Verilog Module         VHDL Module         VHDL Nodule         VHDL Package         Who Test Bench         Embedded Processor	File name: my_xor Location: D:\alhissi\DITE\tutorials\tutorial_1 
More Info	Next > Cancel

- 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

- 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.
- 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).
- 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.
- 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).

ISE Project Navigator (0.61xd) - D:\alhissi\DITE\tuto	orials\tutorial_1	\tuto	rial_1.xise - [my_xor.sch*]
File Edit View Project Source Process Add Tools	Window Layout	Help	
×   🗠 🍽 🗶 🛛 🖧 📗 🍪 🖬 🖌	» 🛛 🏓 🔎 🎉	<b>B</b> /	* 🖻 🔁 j 🖶 🗉 🖻 j 🖉 😥 🗶 j 🕨 🛛 🗶 j 🖓
Symbols	⇔⊡₽×	k	
Categories		X	Add Wire
< All Sumbole >		٦.	
Advanced	-	abc	
Arithmetic		=(0)	
Buffer		<u>a(0)</u>	
Carry_Logic		►	Add I/O Marker
Clocking_Resources		'	
Comparator		-00	
Counter			
Decoder	<u> </u>	<del>**</del>	AND2
		÷ 🐴	)
Carlada			
Symbols		$\sim$	0000
ofdxi_1	<b></b>	0	
or12			
or16		1/	/
or2		i 🗖	4102
or2b1			
or2b2		βA	
or2			
or3	-	1	
]		-₽	
Symbol Name Filter		2412	
		113	
O Martin Para		×	
Orientation		8	
Kotate U	<b>•</b>	~	1
	1	æ	
Symbol Info			
		÷	
🛛 🕫 Design 👘 Files 🚺 Libraries 🔜 Symbols 🌮	Options		my_xor.sch*

- 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*.
- 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.

> New Source Wizard	×
Select Source Type Select source type, file name and its location.	
Image: Second Structure S	File name:          my_xor_tst         Location:         D:\alhissi\DITE\tutorials\tutorial_1
More Info	Next > Cancel

- 4. The page *Associate Source* 
  - 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:



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.
- 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.

	1a 🗆 🖬 🗣 🛛 🎤 😽 🗍	p p 🔉 🔊	🗟 🛛 🗠 🛨 🕴	_i <sup>(</sup> ⇒ <sub>1</sub> 🖸 🕨	▶ <sup>X</sup> 1.00us ▼ (	🥶 🛛 🗔 Re-lau	nch	
. 🎤	9							1,000.000 ns
	Name	Value	10 ns	200 ns	400 ns	600 ns	1800 ns	1,000 ns
	la ai	1						
	b1	1						
	Lig y1	••••••••••••••••••••••••••••••••••••••						
	-							
1								
I.								
12								
3	L							

### 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.

🖞 Symbol Wizard 🛛 🔀
<b>Source Page</b> Select the source for pin names and the symbol shape.
Pin name source         Specify manually         Using schematic         my_xor         Using symbol         Import symbol attributes
Shape © Do not use reference symbol © Rectangle © Square
C Use reference symbol Browse,
More Info Cancel

- 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.
  - 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.

#### 👩 Symbol Wizard

Pin Page

Define the pins to be placed on the symbol shape.

	/_xor				
in	definitions:				
	Name /	Polarity	Side	Order	Add Pin
1	A1	Input 🗾	Left 💌	1 •	Remove Pin/Spacer
2	В1	Input 💌	Left 💌	2 🔹	Insert Spacer
3	Y1	Output 💽	Right 💌	1 •	Move Spacer Up
					Move Spacer Down

#### **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).

	Object Properties - N Category     Output     Nets     Units     V(0:3)	et Attributes View and edit the Name Name	attributes of the select Value (3:0)	ed nets Visible Add	New Edit Traits
		PortPolarity	Not a port	Add	Delete
my_inverter.sch*		,	Cancel	Apply	Help

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).

· · · · · · · · · · · · · · · · · · ·				🏠			
				· · · · · <b>O</b> ] · · · ·			
•••••••••••••••••••••••••••••••••••••••						· · · · · · · · · · · ·	
🏹 .				🦻 🗖			
				· · · · · <b>Equi</b> · · · ·			
· · · · · · · · · · · · · · · · · · ·							
		National					
		1. Alex					
		INV					
				· · · · · · · · · · · · ·			
	<u> </u>			<			
		1 San Contra					
		MINW					
	D Object	Properties - Ne	et Attributes				X
							_
	· · · · · · ·						
	· · · · · · Category	Vi	iew and edit the	attributes of the select	ed nets		
	Nets						
		(0)					
	· · · · · · · · · · · · · · · · · · ·	(3)	Name	Value	Vicible	Nou	
· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	(3)	Name	Yalue	Visible	New	
	· · · · · · · · · · · · · · · · · · ·	(3)	Name	Yalue	Visible	New	
	γ	(3)	Name Name	Yalue Y(3)	Visible Add	New Edit Typita	
	·····γ	(3)	Name Name	Value           Y(3)	Visible Add	New Edit Traits	
	·····γ	3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits	
· · · · · · · · · · · · · · · · · · ·	·····γ	(3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits	
		(3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits Delete	
		(3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits Delete	
	ү	(3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits Delete	
		3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits Delete	
	·····γ	3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits Delete	
	·····γ	3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits Delete	
	·····γ	3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits Delete	
	·····γ	3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits Delete	
	·····γ	3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits Delete	
	·····γ	3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits Delete	
	·····γ	3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits Delete	
	μγ	3)	Name Name PortPolarity	V(3)         Not a port	Visible Add Add	New Edit Traits Delete	
	μΥ	3)	Name Name PortPolarity	V(3)         Not a port	Visible Add Add	New Edit Traits Delete	
	μΥ	3)	Name Name PortPolarity	V(3)         Not a port	Visible Add Add	New Edit Traits Delete	
	μΥ	3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits Delete	
	·····γ	3)	Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add	New Edit Traits Delete Help	1
my inverter.sch	ж. Y	3)	Name Name PortPolarity	Value       Y(3)       Not a port	Visible Add Add Add	New Edit Traits Delete Help	

- 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';.</li>

17	5 🗄 🗖 🖻 🗍 🌽 🍋 🖉	= 🖉 🥕 🍕 =	🛨 🕴 🗠 🐴	🔄 🕨 🔀 🗔 1.000	ıs 💌 ⁄ 📰 📗 🗖	Re-launch		
Æ								1,000
P	Name	¥alue	10 ns	200 ns	1400 ns	1600 ns	1800 ns	1,000
2	▶ 🏹 y[0:3]	0011	0110	X		0011		
~	▶ <b>『</b> ॑ a[0:3]	1100	1001	X		1100		
œ								
$\odot$								
1								
<b>2</b> 1								
1								
1								
÷ 🐴								
21								

#### 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 designpractice 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 ***
```