# Tutorial #1

The goal of this lab is to get familiar with the mechanics of designing digital systems using VHDL and ALTERA's FPGAs.

The development board used in this class is ALTERA's DE2-115. The board provides the following hardware:

- Altera Cyclone IV EP4CE115F29C7 FPGA device
- Altera Serial Configuration device EPCS64
- USB Blaster (on board) for programming; both JTAG and Active Serial (AS) programming modes are supported
- 2MB SRAM
- Two 64MB SDRAM
- 8MB Flash memory
- SD Card socket
- 4 Push-buttons
- 18 Slide switches
- 18 Red user LEDs
- 9 Green user LEDs
- 50MHz oscillator for clock sources
- 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (8-bit high-speed triple DACs) with VGA-out connector
- TV Decoder (NTSC/PAL/SECAM) and TV-in connector
- 2 Gigabit Ethernet PHY with RJ45 connectors
- USB Host/Slave Controller with USB type A and type B connectors
- RS-232 transceiver and 9-pin connector
- PS/2 mouse/keyboard connector
- IR Receiver
- 2 SMA connectors for external clock input/output
- One 40-pin Expansion Header with diode protection
- One High Speed Mezzanine Card (HSMC) connector
- 16x2 LCD module

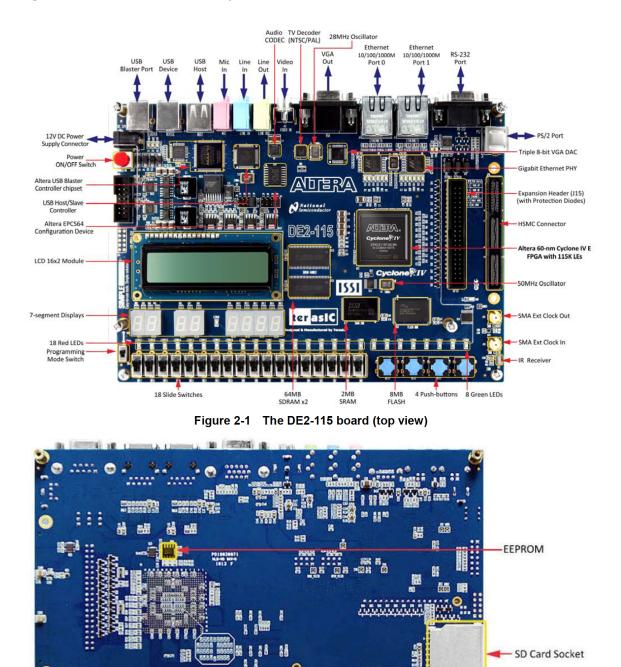


Figure 2.1 and 2.2 shows the layout of the DE2-115 board.



ä

ĕ

5

0

.

.

. . The objectives of the lab are:

- 1. Complete (and read thoroughly) the attached tutorial (skip pp. 36-39). Before you can use Quartus II software you will have to setup the license
- 2. Make sure to skim through the DE2-115 User Manual to learn more about the wide range of features offered by the board.
- 3. (Optional: ... but recommended) Install Altera Web Edition and Altera Modelsim Starter Edition on your PC/Laptop and go through the first five interactive tutorials (Quartus II introduction, Create a Design, Compile a Design, Run Timing Analysis, Configure a Device)

# NOTE:

The only class of constrains set in this tutorial is the mapping between the I/O ports of the design and the FPGA PINs.

In general, there is another class of constrains that must to be set: the timing specifications. Timing are critically important for a successful design. Timing constrains are set creating a Synopsys Design Constraints File (.sdc) that the Quartus II TimeQuest Timing Analyzer uses during design compilation.

# ALTERA's "Development" Methodology

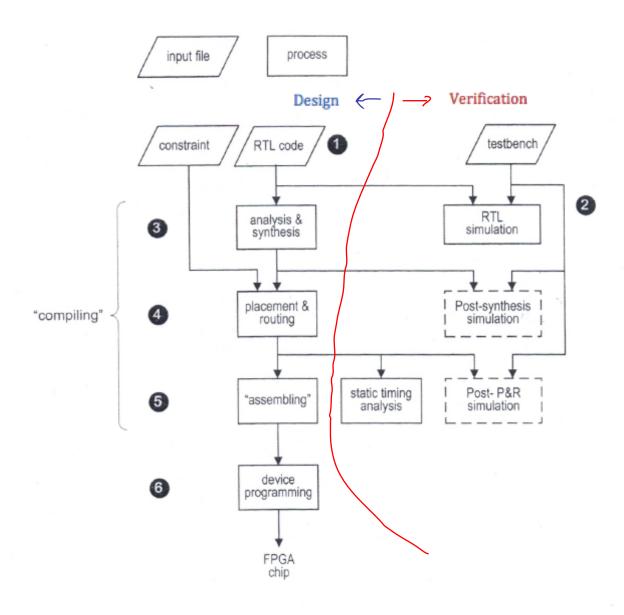


Figure 3.7 Development flow.

### RTL = Register Transfer Level

Analysis and Synthesis = check HDL code and construct gate level netlist

P&R = derive physical layout inside the FPGA chip (a.k.a. *fitting*)

Assembling = generate the configuration file (a.k.a. bit file)

Device programming: = downloading the configuration file into the target device

# Overview of Altera's Quartus II software tools

ISE (integrated Software Environment) Window Structure:

- 1. Project Navigator
- 2. Tasks Window (a.k.a. process window)
- 3. Messages Window (ak.a. log window)
- 4. Workplace Window

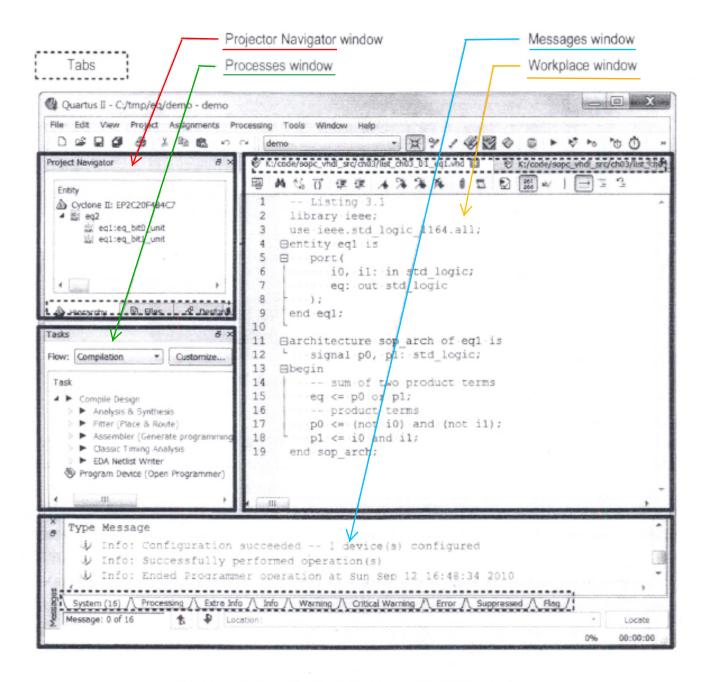


Figure 3.8 Typical Quartus II GUI window.

# Checklist of Basic Development Steps (FPGA based design)

- 1. Create the Logic Design
  - a. Create a project (workspace selection, device selection, EDA tools selection  $\rightarrow$  set preferences)
  - b. Create or add HDL files
  - c. Check HDL syntax (analysis)
- 2. Create an HDL Test bench and perform RTL simulation
- 3. Synthesis and Implementation
  - a. Create constrains (Pin Assignment, Timing, etc)
  - b. Run the Synthesis tool and the fitter tool (P&R)
  - c. Create and check design reports
- 4. Assemble and Program
  - a. Connect the download cable (USB blaster)
  - b. Run the assembler tool to generate the configuration file (a.k.a. bit file)
  - c. Download the configuration file

# **Quartus II Introduction Using VHDL Design**

This tutorial presents an introduction to the Quartus II 9.1 CAD system. It gives a general overview of a typical CAD flow for designing circuits that are implemented by using FPGA devices, and shows how this flow is realized in the Quartus II software. The design process is illustrated by giving step-by-step instructions for using the Quartus II software to implement a very simple circuit in an Altera FPGA device.

The Quartus II system includes full support for all of the popular methods of entering a description of the desired circuit into a CAD system. This tutorial makes use of the VHDL design entry method, in which the user specifies the desired circuit in the VHDL hardware description language. Two other versions of this tutorial are also available; one uses the Verilog hardware description language and the other is based on defining the desired circuit in the form of a schematic diagram.

The last step in the design process involves configuring the designed circuit in an actual FPGA device. To show how this is done, it is assumed that the user has access to the Altera DE2-115 Development and Education board connected to a computer that has Quartus II software installed. A reader who does not have access to the DE2-115 board will still find the tutorial useful to learn how the FPGA programming and configuration task is performed.

The screen captures in the tutorial were obtained using the Quartus II version 5.0; if other versions of the software are used, some of the images may be slightly different.

#### **Contents:**

Typical CAD flow Getting started Starting a New Project VHDL Design Entry Compiling the Design Pin Assignment Simulating the Designed Circuit Programming and Configuring the FPGA Device Testing the Designed Circuit Computer Aided Design (CAD) software makes it easy to implement a desired logic circuit by using a programmable logic device, such as a field-programmable gate array (FPGA) chip. A typical FPGA CAD flow is illustrated in Figure 1.

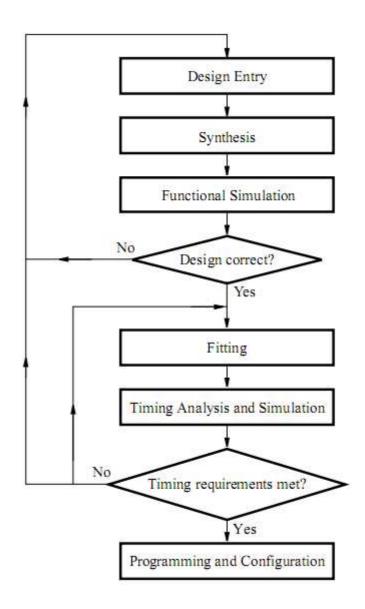


Figure 1. Typical CAD flow.

The CAD flow involves the following steps:

- **Design Entry** the desired circuit is specified either by means of a schematic diagram, or by using a hardware description language, such as VHDL or Verilog
- **Synthesis** the entered design is synthesized into a circuit that consists of the logic elements (LEs) provided in the FPGA chip
- **Functional Simulation** the synthesized circuit is tested to verify its functional correctness; this simulation does not take into account any timing issues

- Fitting the CAD Fitter tool determines the placement of the LEs defined in the netlist into the LEs in an actual FPGA chip; it also chooses routing wires in the chip to make the required connections between specific LEs
- Timing Analysis propagation delays along the various paths in the fitted circuit are analyzed to provide an indication of the expected performance of the circuit
- **Timing Simulation** the fitted circuit is tested to verify both its functional correctness and timing
- Programming and Configuration the designed circuit is implemented in a physical FPGA chip by programming the configuration switches that configure the LEs and establish the required wiring connections

This tutorial introduces the basic features of the Quartus II software. It shows how the software can be used to design and implement a circuit specified by using the VHDL hardware description language. It makes use of the graphical user interface to invoke the Quartus II commands. Doing this tutorial, the reader will learn about:

- Creating a project
- Design entry using VHDL code
- Synthesizing a circuit specified in VHDL code
- Fitting a synthesized circuit into an Altera FPGA
- Assigning the circuit inputs and outputs to specific pins on the FPGA
- Simulating the designed circuit
- Programming and configuring the FPGA chip on Altera's DE2-115 board

### 1 Getting Started

Each logic circuit, or subcircuit, being designed with Quartus II software is called a project. The software works on one project at a time and keeps all information for that project in a single directory (folder) in the file system. To begin a new logic circuit design, the first step is to create a directory to hold its files. To hold the design files for this tutorial, we will use a directory introtutorial. The running example for this tutorial is a simple circuit for two-way light control.

Start the Quartus II software. You should see a display similar to the one in Figure 2. This display consists of several windows that provide access to all the features of Quartus II software, which the user selects with the computer mouse. Most of the commands provided by Quartus II software can be accessed by using a set of menus that are located below the title bar. For example, in Figure 2 clicking the left mouse button on the menu named File opens the menu shown in Figure 3. Clicking the left mouse button on the entry Exit exits from Quartus II 9.1 software. In general, whenever the mouse is used to select something, the left button is used. Hence we will not normally specify which button to press. In the few cases when it is necessary to use the right mouse button, it will be specified explicitly.

🍕 Quartus II	
<u>File Edit View Project Assignments Processing Tools Mindow Help</u>	
Project Navigator	
Hierarchy E Files	
Tasks( • ×	
Flow: Compilation	
	ARTUS® II Version 9.1
Image: Constraint of the second se	Documentation
Y Type Message       System ( Processing ) Extra Info ) Info ) Warning ) Critical Warning ) Error ) Signature       Message:       Image:       Image: <t< th=""><th>uppressed À Flag / ✓ Locate</th></t<>	uppressed À Flag / ✓ Locate
tor nergy press re	

Figure 2. The main Quartus II display.

🗞 Quartus II	
<u>File E</u> dit <u>V</u> iew <u>P</u> roj	ect <u>A</u> ssignments
D gew	Ctrl+N
🚰 Open	Ctrl+0
<u>C</u> lose	Ctrl+F4
🔀 New Project <u>W</u> izard.	
🚰 Open P <u>r</u> oject	Ctrl+J
Convert MAX+PLUS II	[ Project
Save Projec <u>t</u>	
Clos <u>e</u> Project	
<b>_</b> <u>S</u> ave	Ctrl+S
Save <u>A</u> s	
Save Current Report	t Section As
<u>F</u> ile Properties	
Create 🦯 Update	
Export	
Convert Programmin;	g Files
🚺 Page Set <u>u</u> p	
🐧 Print Pre <u>v</u> iew	
🖨 Print	Ctrl+P
Recent F <u>i</u> les	
Recent Projects	
E <u>x</u> it	Alt+F4

Figure 3.An example of the File menu.

For some commands it is necessary to access two or more menus in sequence. We use the convention **Menu1 > Menu2 > Item** to indicate that to select the desired command the user should first click the left mouse button on **Menu1**, then within this menu click on **Menu2**, and then within **Menu2** click on **Item**. For example, **File > Exit** uses the mouse to exit from the system. Many commands can be invoked by clicking on an icon displayed in one of the toolbars. To see the command associated with an icon, position the mouse over the icon and a tooltip will appear that displays the command name.

#### 1.1 Quartus II Online Help

Quartus II software provides comprehensive online documentation that answers many of the questions that may arise when using the software. The documentation is accessed from the menu in the Help window. To get some idea of the extent of documentation provided, it is worthwhile for the reader to browse through the Help menu. For instance, selecting **Help > How to Use Help** gives an indication of what type of help is provided. The user can quickly search through the Help topics by selecting **Help > Search**, which opens a dialog box into which key words can be entered. Another method, context-sensitive help, is provided for quickly finding documentation for specific topics. While using most applications, pressing the F1 function key on the keyboard opens a Help display that shows the commands available for the application.

#### 2 Starting a New Project

To start working on a new design we first have to define a new design project. Quartus II software makes the designer's task easy by providing support in the form of a wizard. Create a new project as follows:

1. Select **File > New Project Wizard** to reach the window in Figure 4, which indicates the capability of this wizard. You can skip this window in subsequent projects by checking the box "**Don't show me this introduction again**". Press **Next** to get the window shown in Figure 5.

# New Project Wizard: Introduction

The New Project Wizard helps you create a new project and preliminary project settings, including the following:

×

Figure 4.Tasks performed by the wizard.

What is the working direct D:/introtutorial	ory for this project?	
in a si ana		<u></u>
What is the name of this p light	roject?	
ngen		
	p-level design entity for this project? The entity name in the design file.	his name is case sensitive
light		
	¥	
Use Existing Project Set	tings	

Figure 5.Creation of a new project.

2. Set the working directory to be introtutorial; of course, you can use some other directory name of your choice if you prefer. The project must have a name, which is usually the same as the top-level design entity that will be included in the project. Choose light as the name for both the project and the top-level entity, as shown in Figure 5. Press **Next**. Since we have not yet created the directory introtutorial, Quartus II software displays the pop-up box in Figure 6 asking if it should create the desired directory. Click **Yes**, which leads to the window in Figure 7.

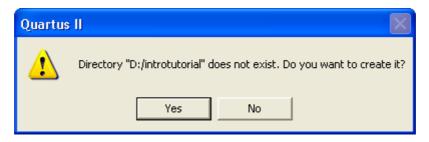


Figure 6.Quartus II software can create a new directory for the project.

Real Provide P	Project	Vizard:	Add Fi	les [pag	e 2 of 5]			×
Sele	ect the desig	n files vou wa	nt to inclu	de in the proie	et. Click Add All I	o add all desig	n files in the	
					add design files t			
<u>F</u> ile	name:						Add	
Fi	le name			Type	Lib	Design enti	Add All	
							<u>R</u> emove	
							<u>P</u> roperties	]
							Up	
							Down	
<						>		
Spe	ecify the path	names of an	y non-defa	ult libraries.	U <u>s</u> er Libra	ries		
				< Back	Next >	Finish	即消	
			_	V Dack	Mext /	rinisn	40,113	

Figure 7.The wizard can include user-specified design files.

3. The wizard makes it easy to specify which existing files (if any) should be included in the project. Assuming that we do not have any existing files, click **Next**, which leads to the window in Figure 8.

New Project Wizard:	Family	& Devid	e Sett	ings [p:	age 3 o	f 5]	×
Colorithe (configurated device of							
Select the family and device y	iou want to i	arget for co	mpliation.				_
Device family				-Show in 'Av	/ailable dev	ice' list	
Eamily: Cyclone IV E			-	Pac <u>k</u> age:	Any	-	
Devices: All				Pin <u>c</u> ount:	Any	•	I
Target device				Sp <u>e</u> ed grad	le: Any	-	
Auto device selected by	y the Fitter			Show a	dvanced de	evices	
<u>Specific device selecte</u>	d in 'Availab	le devices'	list	🗖 HardCo	py compatit	ole only	
Available devices:							_
Name	Core v	LEs	User I/	Memor	Embed	PLL	~
EP4CE115F23C8L	1.0V	114480	281	3981312	532	4	_
EP4CE115F23C9L	1.0V	114480	281	3981312	532	4	
EP4CE115F23I7	1.2V	114480	281	3981312	532	4	
EP4CE115F23I8L	1.0V	114480	281	3981312	532	4	
EP4CE115F29C7	1.2V	114480	529	3981312	532	4	
EP4CE115F29C8 EP4CE115F29C8L	1.2V 1.0V	114480 114480	529 529	3981312	532 532	4	
EP4CE115F29C9L	1.0V 1.0V	114480	529 529	3981312 3981312	532 532	4	<b>v</b>
<						>	
Companion device							_
HardCopy:						-	I
Limit DSP & RAM to Har	rdCopy devi	ce resource	:				
		< Back	Next	> v:	nish	取消	
	_	/ Dack	Mext	/ F1	nisn		

Figure 8.Choose the device family and a specific device.

4. We have to specify the type of device in which the designed circuit will be implemented. Choose Cyclone IV as the target device family. We can let Quartus II software select a specific device in the family, or we can choose the device explicitly. We will take the latter approach. From the list of available devices, choose the device called EP4CE115F29C7 which is the FPGA used on Altera's DE2-115 board. Press "**Next**", which opens the window in Figure 9.

New Project Wizard: EDA Tool Settings [page 4 of 5]
Specify the other EDA tools in addition to the Quartus II software used with the project.
Design Entry/Synthesis
Tool name: KNone>
Format:
Run this tool automatically to synthesize the current design
Simulation
Tool name: <none></none>
Format:
Timing Analy\$
Tool name: None>
Format:
Run this tool automatically after compilation
< Back Next > Finish 取消

Figure 9.Other EDA tools can be specified.

5. The user can specify any third-party tools that should be used. A commonly used term for CAD software for electronic circuits is EDA tools, where the acronym stands for Electronic Design Automation. This term is used in Quartus II messages that refer to third-party tools, which are the tools developed and marketed by companies other than Altera. Since we will rely solely on Quartus II tools, we will not choose any other tools. Press Next.

6. A summary of the chosen settings appears in the screen shown in Figure 10. Press Finish, which returns to the main Quartus II window, but with light specified as the new project, in the display title bar, as indicated in Figure 11.

New Project Wizard: Su	mmary [page 5 of 5]	$\mathbf{X}$
When you click Finish the project	t will be created with the following settings:	
when you click rimsh, the project	t will be cledted with the following settings.	
Project directory:		
E:/test_tutorial/light/		
Project name:	light	
Top-level design entity:	light	
Number of files added:	0	
Number of user libraries added:	4	
Device assignments:		
Family name:	Cyclone IV E	
Device:	EP4CE115F29C7	
EDA tools:		
Design entry/synthesis:	<none></none>	
Simulation:	<none></none>	
Timing analysis:	<none></none>	
Operating conditions:		
VCCINT voltage:	1.0V	
Junction temperature range:	0-85 <del>癖</del>	
		_
	< Back Next > [Finish] 取消	

Figure 10.Summary of the project settings.

🍕 Quartus II - D:/introtutorial/light - light	
<u>File Edit View Project Assignments Processing Tools Mindow Melp</u>	
🗅 📽 🖬 🗿 💩 🕺 🕸 🙉 🕫 🗢 🛛 🖬 🔂 🖉	
X / Ø Ø @ > \$ % 00 \$ 9 @ 0	
Project Navigator X	
Entity	
▲     Cyclone IV E: EP4CE11       ▲     →       1ight 者圖	
	л⊻≜∖.
	TTO TT
Flow: Compilation	Version 9.1
Task	
E Compile Design	View Quartus II Information
Edit Se	
	<b>Occumentation</b>
× Type Message	
g System Λ Processing λ Extra Info λ Info λ Warning λ Critical Warning λ Error λ Su	uppressed ) Flag /
System & Processing & Extra Info & Info & Warning & Critical Warning & Error & Su         Message:         Image:	Locate
For Help, press F1	Idle IVM

Figure 11.The Quartus II display for the created project.

### 3 Design Entry Using VHDL Code

As a design example, we will use the two-way light controller circuit shown in Figure 12. The circuit can be used to control a single light from either of the two switches, x1and x2, where a closed switch corresponds to the logic value 1. The truth table for the circuit is also given in the figure. Note that this is just the Exclusive-OR function of the inputs x1 and x2, but we will specify it using the gates shown.

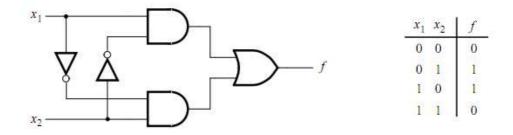


Figure 12. The light controller circuit.

The required circuit is described by the VHDL code in Figure 13. Note that the VHDL entity is called light to match the name given in Figure 5, which was specified when the project was created. This code can be typed into a file by using any text editor that stores ASCII files, or by using the Quartus II text editing facilities. While the file can be given any name, it is a common designers' practice to use the same name as the name of the top-level VHDL entity. The file name must include the extension vhd, which indicates a VHDL file. So, we will use the

name light.vhd.

LIBRARY ieee ; USE ieee.std\_logic\_1164.all ; ENTITY light IS

```
PORT ( x1, x2 : IN STD_LOGIC ;
f: OUT STD_LOGIC ) ;
```

END light ;

ARCHITECTURE LogicFunction OF light IS BEGIN f <= (x1 AND NOT x2) OR (NOT x1 AND x2); END LogicFunction ;

Figure 13.VHDL code for the circuit in Figure 12.

#### 3.1 Using the Quartus II Text Editor

This section shows how to use the Quartus II Text Editor. You can skip this section if you prefer to use some other text editor to create the VHDL source code file, which we will name light.vhd. Select **File > New** to get the window in Figure 14, choose VHDL File, and click **OK**. This opens the Text Editor window. The first step is to specify a name for the file that will be created. Select **File > Save As** to open the pop-up box depicted in Figure 15. In the box labeled Save as type choose VHDL File. In the box labeled File name type light. Put a checkmark in the box **Add file to current project**. Click **Save**, which puts the file into the directory introtutorial and leads to the Text Editor window shown in Figure 16. Maximize the Text Editor window and enter the VHDL code in Figure 13 into it. Save the file by typing **File > Save**, or by typing the shortcut **Ctrl-s**.

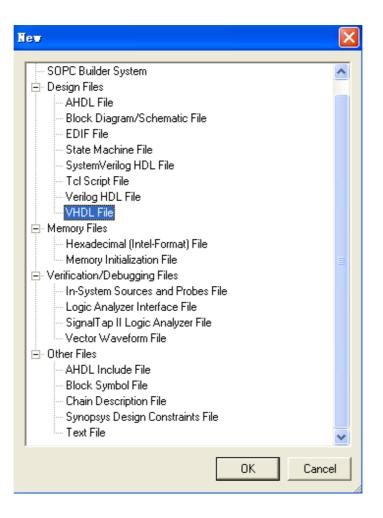


Figure 14.Choose to prepare a VHDL file.

Save As				
Save As Save in: My Recent Documents Desktop My Documents	introtutorial		<b>← € </b> Ċ*	
My Computer	File name: Save as type:	light.vhd         VHDL File (*.vhd)*.vhdl)         ✓         Add file to current project		Save Cancel

# Figure 15.Name the file.

🕸 ligh	t.vhd			
<b>.</b>	1			
<b>44</b> 0.5				
商编 7 7 7 7 7 7 7 7 7 7 7 7 7				
20				
$\mathbf{a}$				
267 268 ab/				
 ≦				
	<			>

# Figure 16.Text Editor window.

Most of the commands available in the Text Editor are self-explanatory. Text is entered at the insertion point, which is indicated by a thin vertical line. The insertion point can be moved either by using the keyboard arrow keys or by using the mouse. Two features of the Text Editor are especially convenient for typing VHDL code. First, the editor can display different types of VHDL statements in different colors, which is the default choice. Second, the editor can automatically indent the text on a new line so that it matches the previous line. Such options can be controlled by the settings in **Tools > Options > Text Editor**.

#### 3.1.1 Using VHDL Templates

The syntax of VHDL code is sometimes difficult for a designer to remember. To help with this issue, the Text Editor provides a collection of VHDL templates. The templates provide examples of various types of VHDL statements, such as an ENTITY declaration, a CASE statement, and assignment statements. It is worthwhile to browse through the templates by selecting **Edit > Insert Template > VHDL** to become familiar with this resource.

#### 3.2 Adding Design Files to a Project

As we indicated when discussing Figure 7, you can tell Quartus II software which design files it should use as part of the current project. To see the list of files already included in the light project, select Assignments > Settings, which leads to the window in Figure 17. As indicated on the left side of the figure, click on the item Files. An alternative way of making this selection is to choose Project > Add/Remove Files in Project.

General	Files				
Files					
Libraries	project the design fil		de in the project. Click Add /	All to add all desig	gn files in the
Device	project directory to t	ne project.			
Operating Settings and Conditions     Compilation Process Settings					
- EDA Tool Settings	<u>File name:</u>				Add
Design Entry/Synthesis	File name	Turne	Library Design entry/s		
- Simulation	light.vhd	Type VHDL File	Library Design entry/s <none></none>	y HDL vers	Add All
- Timing Analysis	ligne.vna	VIDENIE	(NORE/		Remove
- Formal Verification					110110110
Physical Synthesis					Up
Board-Level					
- Analysis & Synthesis Settings					<u>D</u> own
					Description
Verilog HDL Input					<u>P</u> roperties
Default Parameters					
Fitter Settings					
Timing Analysis Settings					
TimeQuest Timing Analyzer					
E Classic Timing Analyzer Settings					
Classic Timing Analyzer Report Assembler					
Assembler Design Assistant					
- SignalTap II Logic Analyzer					
- Logic Analyzer Interface					
- Simulator Settings					
- Simulation Verification					
Simulation Output Files					
- PowerPlay Power Analyzer Settings	<			>	
SSN Analyzer					

Figure 17.Settings window.

If you used the Quartus II Text Editor to create the file and checked the box labeled Add file to current project, as described in Section 3.1, then the light.vhd file is already a part of the project and will be listed in the window in Figure 17. Otherwise, the file must be added to the project. So, if you did not use the Quartus II Text Editor, then place a copy of the file light.vhd, which you created using some other text editor, into the directory introtutorial. To add this file to the project, click on the File name: button in Figure 17 to get the pop-up window in Figure 18. Select the light.vhd file and click Open. The selected file is now indicated in the Files window of Figure 17.Click OK to include the light.vhd file in the project. We should mention that in many cases the Quartus II software is able to automatically find the right files to use for each entity referenced in VHDL code, even if the file has not been explicitly added to the project. However, for complex projects that involve many files it is a good design practice to specifically add the needed files to the project, as described above.

Select File				
Look in:	introtutorial		+ 🗈 💣 📰•	
My Recent Documents Desktop My Documents My Computer	Dincremental_db			
	File name:	Vhdl1.vhd	•	Open
	Files of type:	Design Files (*.tdf;*.vhd;*.vhdl;*.v;	*.vlg;*.verilog 💌	Cancel



### 4 Compiling the Designed Circuit

The VHDL code in the file light.vhd is processed by several Quartus II tools that analyze the code, synthesize the circuit, and generate an implementation of it for the target chip. These tools are controlled by the application program called the Compiler.

Run the Compiler by selecting **Processing > Start Compilation**, or by clicking on the toolbar icon  $\blacktriangleright$  that looks like a purple triangle. As the compilation moves through various

stages, its progress is reported in a window on the left side of the Quartus II display. Successful (or unsuccessful) compilation is indicated in a pop-up box.

Acknowledge it by clicking **OK**, which leads to the Quartus II display in Figure 19. In the message window, at the bottom of the figure, various messages are displayed. In case of errors, there will be appropriate messages given.

When the compilation is finished, a compilation report is produced. A window showing this report is opened automatically, as seen in Figure 19. The window can be resized, maximized, or closed in the normal way, and it can be opened at any time either by selecting **Processing** 

**> Compilation Report** or by clicking on the icon 🖗 .The report includes a number of sections listed on the left side of its window. Figure 19 displays the Compiler Flow Summary section, which indicates that only one logic element and three pins are needed to implement this tiny circuit on the selected FPGA chip.

## 4.1 Errors

ow Summary	
Flow Status	Successful - Wed Jul 07 21:32:05 2010
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Full Version
Revision Name	light
Top-level Entity Name	light
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Preliminary
Met timing requirements	N/A
Total logic elements	1 / 114,480 ( < 1 % )
Total combinational functions	1 / 114,480 ( < 1 % )
Dedicated logic registers	0 / 114,480 (0 %)
Total registers	0
Total pins	3 / 529 ( < 1 % )
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0/4(0%)

Figure 19.Display after a successful compilation.

Quartus II software displays messages produced during compilation in the Messages window. If the VHDL design file is correct, one of the messages will state that the compilation was successful and that there are no errors. If the Compiler does not report zero errors, then there is at least one mistake in the VHDL code. In this case a message corresponding to each error found will be displayed in the Messages window. Double-clicking on an error message will highlight the offending statement in the VHDL code in the Text Editor window. Similarly, the Compiler may display some warning messages. Their details can be explored in the same way as in the case of error messages. The user can obtain more information about a specific error or warning message by selecting the message and pressing the **F1** function key.

To see the effect of an error, open the file light.vhd. Remove the semicolon in the statement that defines the function **f**, illustrating a typographical error that is easily made. Compile the erroneous design file by clicking on the icon. A pop-up box will ask if the changes made to the light.vhd file should be saved; click Yes. After trying to compile the circuit, Quartus II software will display a pop-up box indicating that the compilation was not successful. Acknowledge it by clicking OK. The compilation report summary, given in Figure 21, now confirms the failed result. Expand the **Analysis & Synthesis** part of the report and then select Messages to have the messages displayed as shown in Figure 22. Double-click on the first error message. Quartus II software responds by opening the light.vhd file and highlighting the statement which is affected by the error, as shown in Figure 23.

Flow Status	Flow Failed - Wed Jul 07 21:44:29 2010
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Full Version
Revision Name	light
Top-level Entity Name	light
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Preliminary
Met timing requirements	N/A
Total logic elements	N/A until Partition Merge
Total combinational functions	N/A until Partition Merge
Dedicated logic registers	N/A until Partition Merge
Total registers	N/A until Partition Merge
Total pins	N/A until Partition Merge
Total virtual pins	N/A until Partition Merge
Total memory bits	N/A until Partition Merge
Embedded Multiplier 9-bit elements	N/A until Partition Merge
Total PLLs	N/A until Partition Merge

Correct the error and recompile the design.

Figure 21.Compilation report for the failed design.

Type	Message
<u> </u>	Info: ************************************
🗉 🤨	Info: Running Quartus II Analysis & Synthesis
(i)	Info: Command: quartus_mapread_settings_files=onwrite_settings_files=off light -c light
- Q	Info: Parallel compilation is enabled and will use 2 of the 2 processors detected
8	Error (10500): VHDL syntax error at light.vhd(12) near text "END"; expecting ";"
	Info: Found 0 design units, including 0 entities, in source file light.vhd
🕀 😟	Error: Quartus II Analysis & Synthesis was unsuccessful. 1 error, 0 warnings
8	Error: Quartus II Full Compilation was unsuccessful. 3 errors, 0 warnings
	$\lambda$ Processing (8) $\langle$ Extra Info $\lambda$ Info (5) $\lambda$ Warning $\lambda$ Critical Warning $\lambda$ Error (3) $\lambda$ Suppressed $\lambda$ Flag /
Message: 0 o	f14 🏤 🦻 Location:

Figure 22.Error messages.

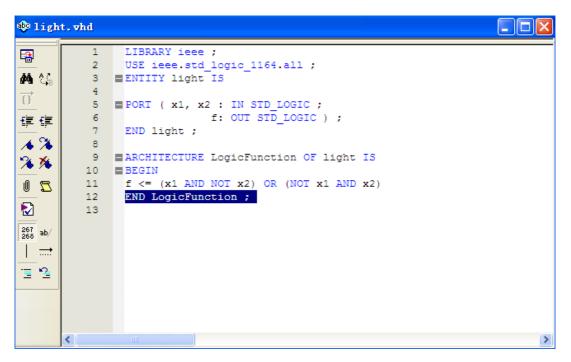


Figure 23.Identifying the location of the error.

### 5 Pin Assignment

During the compilation above, the Quartus II Compiler was free to choose any pins on the selected FPGA to serve as inputs and outputs. However, the DE2-115 board has hardwired connections between the FPGA pins and the other components on the board. We will use two toggle switches, labeled **SW0** and **SW1**, to provide the external inputs, **x1** and **x2**, to our example circuit. These switches are connected to the FPGA pins **AB28** and **AC28**, respectively. We will connect the output f to the green light-emitting diode labeled LEDG0, which is hardwired to the FPGA pin **E21**.

Pin assignments are made by using the Assignment Editor. Select **Assignments > Pins** to reach the window in Figure 24. Under Category select Pin. Double-click on the entry <<new>> which is highlighted in blue in the column labeled To. The drop-down menu in Figure 25 will appear. Click on **x1** as the first pin to be assigned; this will enter **x1** in the displayed table. Follow this by double-clicking on the box to the right of this new **x1** entry, in the column

labeled Location. Now, the drop-down menu in Figure 26 appears. Scroll down and select **PIN\_AB28**. Instead of scrolling down the menu to find the desired pin, you can just type the name of the pin (**AB28**) in the Location box. Use the same procedure to assign input x2 to pin **AC28** and output f to pin **E21**, which results in the image in Figure 27. To save the assignments made, choose **File > Save**. You can also simply close the Assignment Editor window, in which case a pop-up box will ask if you want to save the changes to assignments; click **Yes**. Recompile the circuit, so that it will be compiled with the correct pin assignments.

🦪 A.	ssign	ment Editor						
-	×	Category:					- 🖓 All 💍	Timing Disc Options
0	×	Show assignment	ts for specific nodes;					
	Node							Check All
	le Fil							Uncheck All
4	Filter:							Delete All
	Information: 지 :	Edit: X	<b>V</b> < <new>&gt;</new>					×
•		From	То	Assignment Name	Value	Enabled		
	1	< <new>&gt;</new>	< <new>&gt;</new>	< <new>&gt;</new>				
«»								

Figure 24. The Assignment Editor window.

То	22
1	
øf	1947 - 443 1947 - 443
₩×1	
ш~х2	

Figure 25.The drop-down menu displays the input and output names.

Named: 🛛 🗸 «» Edit: 🗶 🗸									
		Node Name	Direction	Location	I/O Ba	nk	VREF Group	I/O Standard	Reserved
1	0	f	Output					2.5 V (default)	
2		x1	Input	PIN_AB28				2.5 V (default)	
3		x2	Input	PIN AB25	IOBANK 5	Row I/O	DIFFIO R44	D	1
4		< <new node="">&gt;</new>		PIN_AB26	IOBANK_5	Row I/O	DIFFIO_R44		
				PIN_AB27	IOBANK_5	Row I/O	DIFFIO_R42	p	
				PIN_AB28	IOBANK_5	Row I/O	DIFFIO_R42	n	
				PIN_AC1	IOBANK_2	Row I/O	DIFFIO_L37	ı	
				PIN_AC2	IOBANK_2	Row I/O	DIFFIO_L37	5	
				PIN_AC3	IOBANK_2	Row I/O	DIFFIO_L38	5	
				PIN_AC4	IOBANK_2	Row I/O	DIFFIO_L49	1	

#### Figure 26.The available pins.

3	Quartus II - D:/introtutorial/light - light - [Pin Planner]								
Fi	ile <u>E</u> dit <u>V</u> iew P <u>r</u> ocessing <u>T</u> ools <u>W</u> indow								
<b>x</b> +	Named: K K K K								
		Node Name		Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
	1	0	f	Output	PIN_E21	7	B7_N0	2.5 V (default)	
	2		x1	Input	PIN_AB28	5	B5_N1	2.5 V (default)	
	3	•	x2	Input	PIN_AC28	5	B5_N2	2.5 V (default)	
	4		< <new node="">&gt;</new>						

Figure 27.The complete assignment.

The DE2-115 board has fixed pin assignments. Having finished one design, the user will want to use the same pin assignment for subsequent designs. Going through the procedure described above becomes tedious if there are many pins used in the design. A useful Quartus II feature allows the user to both export and import the pin assignments from a special file format, rather than creating them manually using the Assignment Editor. A simple file format that can be used for this purpose is the comma separated value (CSV) format, which is a common text file format that contains comma-delimited values. This file format is often used in conjunction with the Microsoft Excel spreadsheet program, but the file can also be created by hand using any plain ASCII text editor. The format for the file for our simple project is

To, Location x1, PIN\_AB28 x2, PIN\_AC28 f, PIN\_E21

By adding lines to the file, any number of pin assignments can be created. Such csv files can be imported into any design project. If you created a pin assignment for a particular project, you can export it for use in a different project. To see how this is done, open again the Assignment Editor to reach the window in Figure 27. Now, select **File > Export** which leads to the window in Figure 28. Here, the file light.csv is available for export. Click on **Export**. If you now look in the directory introtutorial, you will see that the file light.csv has been created.

Export				
Save in: My Recent Documents Desktop My Documents	introtutorial db incremental_dl	2	<b>← € ☆ </b>	
My Network Places	File name: Save as type:	light.csv Comma Separated Value File (*.csv	)	Export Cancel

Figure 28.Exporting the pin assignment.

You can import a pin assignment by choosing **Assignments > Import Assignments**. This opens the dialogue in Figure 29 to select the file to import. Type the name of the file, including the csv extension and the full path to the directory that holds the file, in the File Name box and press **OK**. Of course, you can also browse to find the desired file.

Import Ass	ignments	
Specify the sou	rce and categories of assignments to import.	
<u>F</u> ile name:		 Categories
🔽 Copy existin	g assignments into light.qsf.bak before importing	Advanced
	OK	Cancel

Figure 29.Importing the pin assignment.

For convenience when using large designs, all relevant pin assignments for the DE2-115 board are given in the file called DE2-15\_pin\_assignments.csv in the directory DE2-115\_tutorials\design\_files, which is included on the CD-ROM that accompanies the DE2-115 board and can also be found on Altera's DE2-115 web pages. This file uses the names found in the DE2-115 User Manual. If we wanted to make the pin assignments for our example circuit by importing this file, then we would have to use the same names in our VHDL

design file; namely, SW(0), SW(1) and LEDG(0) for x1, x2 and f, respectively. Since these signals are specified in the DE2-115\_pin\_assignments.csv file as elements of arrays SW and LEDG, we must refer to them in the same way in the VHDL design file. For example, in the DE2-115\_pin\_assignments.csv file the 18 toggle switches are called SW [17] to SW [0]; since VHDL uses parentheses rather than square brackets, these switches are referred to as SW (17) to SW (0). They can also be referred to as an array SW (17 downto 0).

## 6 Simulating the Designed Circuit

Before implementing the designed circuit in the FPGA chip on the DE2-115 board, it is prudent to simulate it to ascertain its correctness. Modelsim software can be used to simulate the behavior of a designed circuit. Next we will introduce how to use the Modelsim to simulate, as follows:

1. Run ModelSim.

Click **Start > Project > Altera > ModelSim-Altera 6.5b** or double-click the ModelSim icon on the desktop. Then the interface shown as Figure 30 will appear. If you have ever set up projects in ModelSim, it will automatically open the latest project.

M HodelSim ALTER	A STARTER	EDITION 6.5b - Custom Altera Version	
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>C</u> omy	pile <u>S</u> imulat	e A <u>d</u> d L <u>i</u> brary T <u>o</u> ols Layo <u>u</u> t <u>W</u> indow <u>H</u> elp	
<u>N</u> ew ▶	<u>F</u> older	2 🕰   🛤 🗄 🖬 📕 🤹 🕮 🚑 述 🛛 🖄 🖄	
<u>O</u> pen	Source		
Load	Project		
<u>C</u> lose	Library		
I <u>m</u> port			
Export •	Туре	Path	
Save	Library	D:/my_proj/modelsim/light/work	
S <u>a</u> ve As	Library	\$MODEL_TECH//altera/vhdl/220model	
Report	Library	\$MODEL_TECH//altera/verilog/220m	
Change Directory	Library	\$MODEL_TECH//altera/vhdl/altera	
Use Source	Library	\$MODEL_TECH//altera/vhdl/altera_mf	
Source Directory	Library	\$MODEL_TECH//altera/verilog/altera	
Datasets	Library	\$MODEL_TECH//altera/verilog/altera	
<u></u>	Library	\$MODEL_TECH//altera/vhdl/altgxb	
En <u>v</u> ironment 🕨	Library	\$MODEL_TECH//altera/vhdl/altgxb	
Page Setup	Library	\$MODEL_TECH//altera/verilog/altgxb	
Print	Library	\$MODEL_TECH//altera/vhdl/arriagx	
Print Postscript	Library	\$MODEL_TECH//altera/vhdl/arriagx	
	Library	\$MODEL_TECH//altera/verilog/arriag	-
Recent Directories 🕨			a x
Recent Projects			
Close <u>W</u> indow	light_simu	Alaim aastrin20alaam/ /wadalaim ini	
<u>Q</u> uit	l/arabs/mod	delsim_ase\win32aloem//modelsim.ini	
ModelSim>			
			-
		<no design="" loaded=""></no>	

Figure 30.ModelSim Start Interface

## 2. Create a new project.

- a. Select File > New > Project (Main window) from the menu bar to get the Create Project window where you can input a Project Name, Project location (directory) and Default Library Name (Figure 31). You can also refer to the library settings from a selected .ini file or add them directly into the project. The default library is located in the path of the project compilation files, and here we usually choose work (default).
- b. Input light\_simu in the Project Name field.
- c. Input **D:/my\_proj/modelsim/light** in Project Location where the project file will be stored. Here we need to know that ModelSim can not automatically build a directory for a new project, so we have to input the path instead of browsing one.
- d. Choose work (default) as the Default Library Name.
- e. Choose ...modelsim\_ase/modelsim.ini as the Copy Setting From.
- f. Click OK.

M Create Project	×
Project Name	
light_simu	
Project Location	
D:/my_proj/modelsim/light	Browse
Default Library Name work	
Copy Settings From	
2/modelsim_ase/modelsim.ini Bro	owse
Copy Library Mappings C Reference Library	ry Mappings
ОК	Cancel

Figure 31.Create Project Window

3. Then the dialog window shown as Figure 32 will appear, which indicates that the project directory does not exist. Click on the **OK** button to create the directory for a new project.



Figure 32.Confirm the establishment of a new directory

4. Once you click **OK** to accept the new project settings, a blank Project window and the "**Add items to the Project**" window will appear (Figure 33), in which you can create a new design file, add an existing file, create a simulation configuration or add a folder for organization purposes.

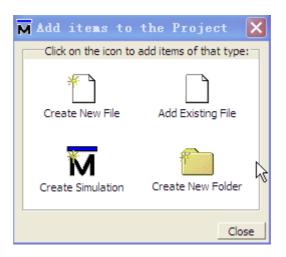


Figure 33.Add items to a Project

- 5. Add existing file.
- a. Click on the Add Existing File icon.

This leads to the **Add file to Project** window displayed in Figure 34. This window allows you to browse the existing files, specify the file type, specify a folder to which the file will be added, and identify whether to leave the file in its current location or copy it to the project directory.

M Add file to Project	$\mathbf{X}$
File Name	
D:/introtutorial/light.vhd	Browse
Add file as type	Folder Top Level
Reference from current location	C Copy to project directory
	OK Cancel

Figure 34.Add existing file to Project

b. Click on the **Browse** button for the File Name field. This leads to the "**Select files to** add to project" window and displays the contents of the current directory. Select light.vhd and click **Open**.

c. Click **OK** to add the files to the project.

d. Click **Close** to dismiss the Add items to the Project window.

## 6. Create new file.

Either Clicking on the **Create New File** icon of Figure 33 or selecting **File >New > Project** from the menu bar can create a new file, here we choose the first method: the **Create Project File** window shown in Figure 35 will appear. Specify the file type, specify a folder to which. Here we input **ligthVHD** as File Name, choose **VHDL** for Add file as type and Top Level for Folder (Folder means the path to which the file will be added and the Top Level means the project path we just set), then click OK. Click **Close** to close the **Add items to the Project** window.

M Create Project File	
File Name ligthVHD Add file as type VHDL	Folder Top Level
	OK Cancel

Figure 35.Add new files to Project

7. Then, the Project option will appear in the Workspace window, which includes lightVHD.vhd. There is also a question mark in Status bar which indicates the document has not been compiled. Double click this file to get the edit window (Figure 36 in which we input our design files as follows:

🕅 ModelSim ALTERA STARTER EDITION 6.5b - Custom Altera V	ersion
<u>F</u> ile <u>E</u> dit <u>V</u> iew Compile <u>S</u> imulate A <u>d</u> d Source T <u>o</u> ols Layout <u>W</u> indow <u>H</u>	elp
X * X D E X □ - 2 = 4 . Layout NoDesign ▼	
	proj/modelsim/light/ligthVHD.vhd 👘 🗰 🗙
▼Name ∇ Status Type Order Modified Ln∓	<u> </u>
With Bigth VHIDL vhid         VHIDL 1         06/13/10         04:40:09         PM         1           With Bigth VHIDL vhid         VHIDL 0         06/13/10         04:34:33         PM         2	
Library Project	<u>}</u>
A Transcript	۲
<pre>file mkdir D:/my_proj/modelsim/light # reading C:\altera\91\modelsim_ase\win32aloem//modelsim.ini # Loading project light # reading C:\altera\91\modelsim_ase\win32aloem//modelsim.ini # Loading project light_simu ModelSim&gt;</pre>	
Ln: 1 Col: 0 Project : light_simu	<no design="" loaded=""></no>

### Figure 36.Edit new Files

In the lightVHD.vhd file window, please input the testbench codes, Click the Save icon.

library ieee; use ieee.std\_logic\_1164.all; entity lightVHD is end lightVHD; ---architecture behav of lightVHD is component light port( x1 :in std\_logic; x2 :in std\_logic; f :out std\_logic); end component;

-----

```
signal ain :std_logic :='0';
signal bin :std logic :='0';
signal cout :std_logic;
begin
----instantiate
U1 :light port map(x1 => ain,x2 => bin,f => cout);
  ----ain stimulus
   Process
 begin
     ain<='0';
    wait for 20 ns;
     ain<='1';
   wait for 20 ns;
       end process;
-----bin stimulus
process
begin
     bin<='0';
    wait for 40 ns;
        bin<='1';
             wait for 20 ns;
     end process;
----
end behav;
```

#### 8. Compile the files.

a. Right-click either ligth.vhd or lightVHD.vhd in the Project window and select **Compile > Compile All** from the pop-up menu. ModelSim not only compiles files but changes the symbol in the Status column to a green check mark as well. A green check mark means the compile succeeded, meantime, there "**Compile of lightVHD.vhd was successful**" in green will appear in the transcript window(Figure 37). If the compilation fails, the symbol will be a red "**X**", and you will see an error message in the Transcript window.

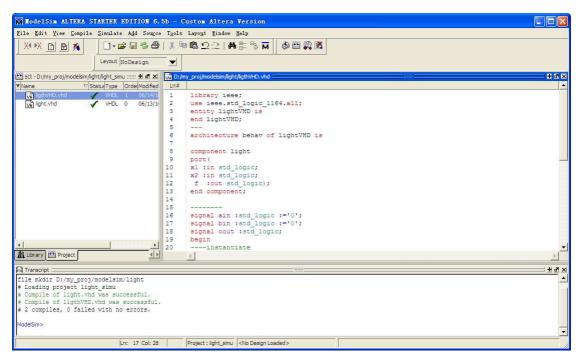


Figure 37.Compile the files

9. Load the Design and Click Menu **Simulate > Start Simulate**, then the interface shown in Figure 38 will appear.

a. Click the Design tab.

b. Click on the + icon to expand the work library (Figure 38).

Here we select **lightvhd** as our simulation object, and then work **lightvhd** module will appear in Design unit. Resolution is the time accuracy for simulation, which is set to default, finally click **ok**.

₹ Name	Type	Path
work	Library	D:/my_proj/modelsim/light/work
🔁 🔁 light	Entity	D:/introtutorial/light.vhd
😐 📔 lightvhd	Entity	D:/my_proj/modelsim/light/ligthVHD.vhd
⊕_ 220model	Library	\$MODEL_TECH//altera/vhdl/220model
	Library	\$MODEL_TECH//altera/verilog/220m
🕂 📕 altera	Library	\$MODEL_TECH//altera/vhdl/altera
→ altera_mf	Library	\$MODEL_TECH//altera/vhdl/altera_mf
	Library	\$MODEL_TECH//altera/verilog/altera
→ altera_ver	Library	\$MODEL_TECH//altera/verilog/altera
Design Unit(s)		Resolution
work.lightvhd		default 🗨
-		
Optimization		

Figure 38. Choose Simulation Object

## 10. Start simulation:

We're ready to start simulation. But before we do, we'll open the Wave window and add signals to it.

- 1. Open the Wave window.
  - a : Enter view wave at the VSIM> prompt in the Transcript window.

The Wave window opens in the right side of the Main window (Figure 39).

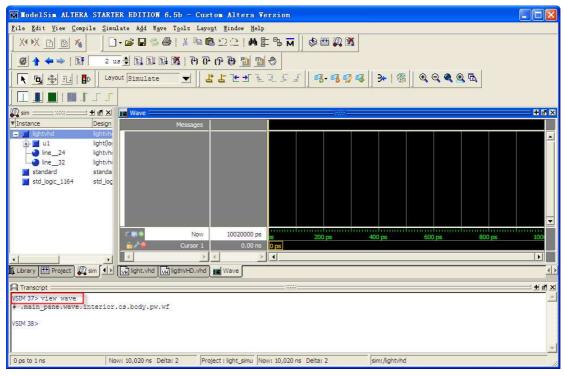


Figure 39. Using the command to open Wave Window

- 2. Add signals to the Wave window.
  - a: Enter **add wave sim:/lightvhd/**\* at the **VSIM>** prompt in the Transcript window. All signals in the design are added to the Wave window (Figure 40).

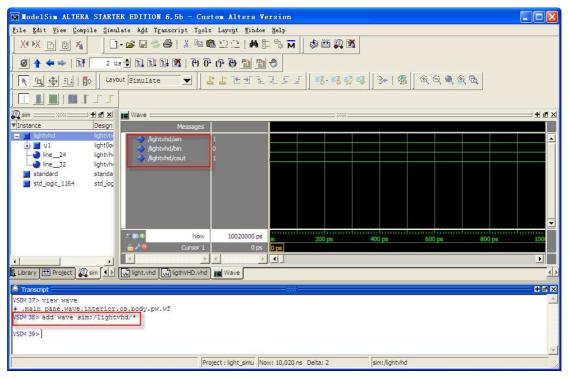


Figure 40.Using the command to add Signals to Wave Window

#### 3. Run the simulation

a. Enter **run 3us** at the **VSIM>** prompt in the Transcript window (Figure 41).

By observing, we can find that the simulation results are the same with our design ones.

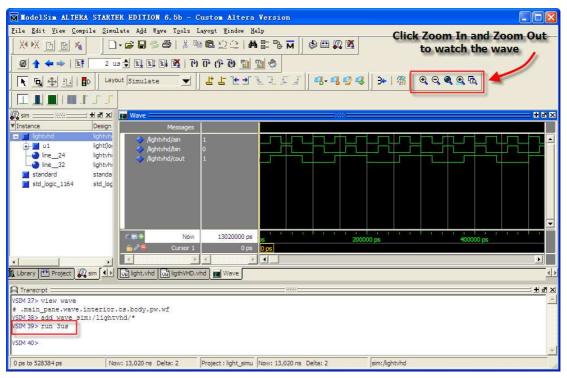


Figure 41. Waves Drawn in Wave Window

### 7 Programming and Configuring the FPGA Device

The FPGA device must be programmed and configured to implement the designed circuit. The required configuration file is generated by the Quartus II Compiler's Assembler module. Altera's DE2-115 board allows the configuration to be done in two different ways, known as JTAG and AS modes. The configuration data is transferred from the host computer (which runs the Quartus II software) to the board by means of a cable that connects a USB port on the host computer to the leftmost USB connector on the board. To use this connection, it is necessary to have the USB-Blaster driver installed. If this driver is not already installed, consult the tutorial Getting Started with Altera's DE2-115 Board for information about installing the driver. Before using the board, make sure that the USB cable is properly connected and turn on the power supply switch on the board.

In the JTAG mode, the configuration data is loaded directly into the FPGA device. The acronym JTAG stands for Joint Test Action Group. This group defined a simple way for testing digital circuits and loading data into them, which became an IEEE standard. If the FPGA is configured in this manner, it will retain its configuration as long as the power remains turned on. The configuration information is lost when the power is turned off. The second possibility is to use the Active Serial (AS) mode. In this case, a configuration device that includes some flash memory is used to store the configuration data.

Quartus II software places the configuration data into the configuration device on the DE2-115 board. Then, this data is loaded into the FPGA upon power-up or reconfiguration. Thus, the FPGA need not be configured by the Quartus II software if the power is turned off and on. The choice between the two modes is made by the **RUN/PROG** switch on the DE2-115 board. The **RUN** position selects the JTAG mode, while the **PROG** position selects the AS mode.

#### 7.1 JTAG Programming

The programming and configuration task is performed as follows. Flip the **RUN/PROG** switch into the **RUN** position. Select **Tools > Programmer** to reach the window in Figure 42. Here it is necessary to specify the programming hardware and the mode that should be used. If not already chosen by default, select JTAG in the Mode box. Also, if the USB-Blaster is not chosen by default, press the **Hardware Setup**... button and select the USB-Blaster in the window that pops up, as shown in Figure 43.

🗳 Quartus II	- D:/introt	utorial/light - lig	ht - [ligh	t.cdf*]							X
<u>F</u> ile <u>E</u> dit P <u>r</u> oce	essing <u>T</u> ools <u>W</u>	indow									
🔔 Hardware Setup	USB-Blaster [U	SB-0]		Mode:	JTAG		▼ Pi	rogress:	0	%	
Enable real-time ISP to allow background programming (for MAX II devices)											
🏓 Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	IS CL4
🖬 Stop	light.sof	EP4CE115F29	005634C9	FFFFFFF							<u> </u>
Auto Detect											
X Delete											
🗳 Add File											
🔛 Change File											
🖺 Save File											
Add Device											
T <sup>en</sup> Up	1										
_ 🤀 Down	<			1111							>
For Help, press Fi	1									NUM	11

Figure 42.The Programmer window.

Observe that the configuration file light.sof is listed in the window in Figure 42. If the file is not already listed, then click **Add File** and select it. This is a binary file produced by the Compiler's Assembler module, which contains the data needed to configure the FPGA device. The extension .sof stands for SRAM Object File. Note also that the device selected is EP4CE115F29C7, which is the FPGA device used on the DE2-115 board. Click on the **Program/Configure** check box, as shown in Figure 44.

Hardware Setup			X
Hardware Settings JTAG So Select a programming hardware hardware setup applies only to	e setup to use whe		es. This programming
Currently selected hardware: Available hardware items:	USB-Blaster [U No Hardware USB-Blaster [U	-	
Hardware USB-Blaster	Server Local	Port USB-0	Add Hardware Remove Hardware
			Close

Figure 43. The Hardware Setup window.

🖥 Quartus II	- D:/introtutoria	1/light – ligh	t – [light	t.cdf	•]							
<u>F</u> ile <u>E</u> dit P <u>r</u> oces	ssing <u>T</u> ools <u>M</u> indow											
🔔 Hardware Setup.	USB-Blaster [USB-0]				Mode: JTAG			▼ Pi	rogress:	0	0%	
Enable real-time ISP to allow background programming (for MAX II devices)												
🏓 Start	File	Device	Checksum	Userc	ode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	IS CLA
📲 Stop	light.sof	EP4CE115F29	005634C9	FFFFF	FFF							]
Auto Detect												
🗙 Delete												
🍰 Add File												
👺 Change File												
Save File												
🗳 Add Device												
1 Up												
	<											_ >
For Help, press F1											NUM	- //

Figure 44.The updated Programmer window.

Now, press Start in the window in Figure 44. An LED on the board will light up when the configuration data has been downloaded successfully. If you see an error reported by Quartus II software indicating that programming failed, then check to ensure that the board is properly powered on.

## 7.2 Active Serial Mode Programming

In this case, the configuration data has to be loaded into the configuration device on the DE2-115 board, which is identified by the name EPCS64. To specify the required configuration device select **Assignments > Device**, which leads to the window in Figure 45. Click on the **Device & Pin Options** button to reach the window in Figure 46. Now, click on the **Configuration tab** to obtain the window in Figure 47.

In the Configuration device box (which may be set to Auto) choose EPCS64 and click **OK**. Upon returning to the window in Figure 45, click **OK**. Recompile the designed circuit.

Libraries  Device Operating Settings and Conditions	Device										
<ul> <li>Operating Settings and Conditions</li> </ul>											
S	Select the family and device you want to target for compilation.										
Voltage Temperature			- Show in 'Available devices' list-								
Compilation Process Settings		Device family					rices' list				
Early Timing Estimate	Eamily: Cyclone IV E			-	Pac <u>k</u> age:	Any	-				
Incremental Compilation					Pin count:	Any	*				
Physical Synthesis Optimization	Devices: All			Ψ.	i in <u>c</u> ount.	Any					
EDA Tool Settings					Speed grade	: Any	•				
Design Entry/Synthesis	Target device				Show ad	vanced c	levices				
Simulation	C Auto device selecter	d by the Fitte	r.								
Timing Analysis		Specific device selected in 'Available devices' list					HardCopy compatible only				
- Formal Verification	C Other n/a										
Physical Synthesis	V Davier und	C Uther n/a					Device and Pin Options				
Board-Level	A 1919 1 1										
Analysis & Synthesis Settings	Available devices:										
WHDL Input	Name	Core v	and a start of the	User I/		Embed	PLL 🔥				
Verilog HDL Input Default Parameters	EP4CE115F23C7	1.2V	114480	281		532	4				
Fitter Settings	EP4CE115F23C8 EP4CE115F23C8L	1.2V 1.0V	114480 114480	281 281	- 2000 V3000 ND2 - 01	532 532	4				
- Timing Analysis Settings	EP4CE115F23C9L	1.0V	114480	281		532 532	4				
TimeQuest Timing Analyzer	EP4CE115F23I7	1.2V	114480	281		532	4				
E - Classic Timing Analyzer	EP4CE115F23I8L	1.0V	114480	281	3981312 5	532	4				
Classic Timing Analyzer Setting	EP4CE115F29C7	1.2V	114480	529	and the property of the proper	532	4				
- Assembler	EP4CE115F29C8	1.2V	114480	529	A STATISTICS AND A STATISTICS	532	4				
- Design Assistant	EP4CE115F29C8L	1.0V	114480	529		532 532	4				
- SignalTap II Logic Analyzer	<						>				
Logic Analyzer Interface	1 No. 1 1000		8 8								
Simulator Settings	Migration compatibility		- Companion	device							
Simulation Verification	Migration Devices		HardCopy:				v				
Simulation Output Files		-	TT LOOP THE	D C DANA	) HardCopy de						
PowerPlay Power Analyzer Setting	0 migration devices sele	cted	IN. Filling D.S.	IF & FIMIN (	ringracopy dev	Ace resou	lices				
SSN Analyzer											
×					ОК		Cancel				

Figure 45. The Device Settings window.

Device and Pin Options 🛛 🗙									
Dual-Purpose Pins         Voltage         Pin Placement           Error Detection CRC         Capacitive Loading         Board Trace Model         I/O Timing           General         Configuration         Programming Files         Unused Pins           Specify general device options. These options are not dependent on the configuration scheme.         Specify general device options.         Configuration									
Options:     Auto-restart configuration after error     Release clears before tri-states   Enable user-supplied start-up clock (CLKUSR)   Enable device-wide reset (DEV_CLRn)   Enable device-wide output enable (DEV_OE)   Enable INIT_DONE output   Enable OCT_DONE									
Auto usercode     JTAG user code (32-bit hexadecimal):     FFFFFFF									
In-system programming clamp state:       Delay entry to user mode:									
Description: Directs the device to restart the configuration process automatically if a data error is encountered. If this option is turned off, you must externally direct the device to restart the configuration process if an error occurs.									

Figure 46.The Options window.

Device and Pin Options 🛛 🗙										
Error Detection CRC       Capacitive Loading       Board Trace Model       I/O Timing         Dual-Purpose Fins       Voltage       Fin Flacement         General       Configuration       Programming Files       Unused Fins										
Specify the device configuration scheme and the configuration device. Note: For HardCopy designs, these settings apply to the FPGA prototype device.										
Configuration scheme: Active Serial (can use Configuration Device)										
Configuration mode: Standard										
Configuration device										
✓ Use configuration device: Auto										
Auto EPCS1 EPCS4 EPCS4 EPCS16 Force VCCID to be compatible EPCS64 EPCS128										
Generate compressed bitstreams										
Active serial clock source:										
Description:										
Specifies the configuration device that you want to use as the means of configuring										
<u></u>										
确定取消										

Figure 47.Specifying the configuration device.

The rest of the procedure is similar to the one described above for the JTAG mode. Select **Tools > Programmer** to reach the window in Figure 42. In the Mode box select **Active Serial Programming**. If you are changing the mode from the previously used JTAG mode, the pop-up box in Figure 48 will appear, asking if you want to clear all devices. Click **Yes**. Now, the Programmer window shown in Figure 49 will appear. Make sure that the Hardware Setup indicates the USB-Blaster. If the configuration file is not already listed in the window, press **Add File**. The pop-up box in Figure 50 will appear. Select the file light.pof in the directory introtutorial and click **Open**. As a result, the configuration file light.pof will be listed in the window. This is a binary file produced by the Compiler's Assembler module, which contains the data to be loaded into the EPCS64 configuration device. The extension .pof stands for Program/Configure check box, as shown in Figure 51.

Qı	uartus	: II 🦳 🔛 🔛 🔛 🔛 🔛 🔛
4	!	Some devices in current device list cannot be added to selected programming mode Active Serial Programming. Do you want to clear all devices in current device list and switch to selected mode?
		Yes No

Figure 48.Clear the previously selected devices.

🖺 Quartus II	- D:/intro	tutorial/light - 1:	ight - [ligh	t.cdf*]							
<u>F</u> ile <u>E</u> dit Proces	ssing <u>T</u> ools	<u>W</u> indow									
🌲 Hardware Setup.	USB-Blaster [	USB-0]		Mode:	Active Serial P	rogrammir	ng 🔻 P	rogress:	0	%	
Enable real-time IS	P to allow backg	round programming (for MAX II	devices)								
🏴 Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	CL
🖬 Stop											
Auto Detect											
X Delete											
灅 Add File											
💕 Change File											
🗳 Save File											
😂 Add Device											
🜓 Up											
🎩 Down 📘	<			1111					]		>
for Help, press F1										NUM	

Figure 49.The Programmer window with Active Serial Programming selected.

nming File				
introtutorial		•	+ 🗈 💣 🎫	
db incremental_dt modelsim light.pof	1			
File name:	light.pof			Open Cancel
	introtutorial	introtutorial db modelsim light.pof File name:	introtutorial         db         incremental_db         modelsim         light.pof	introtutorial       Image: Constrained and the second an

Figure 50.Choose the configuration file.

🖥 Quartus II	- D:/introtutori:	al/light – ligh	t - [light	.cdf*]								
<u>F</u> ile <u>E</u> dit P <u>r</u> oce	ssing <u>T</u> ools <u>W</u> indow											
🔔 Hardware Setup	Setup USB-Blaster [USB-0]				Mode: Active Serial Programming 💌 Progress:							
Enable real-time ISP to allow background programming (for MAX II devices)												
🏓 Start	File	Device	Checksum	Usercode	Program/ Configure	/erify Blank- Check	Examine	Security Bit	Erase	ISP CLAMP		
🖬 Stop	light.pof	EPCS64	712E5C94	00000000								
Auto Detect												
X Delete												
Add File												
👺 Change File												
🛱 Save File												
Add Device												
1 Up												
🔑 Down	]											
For Help, press Fi	1								NU	M //		

Figure 51.The updated Programmer window.

Flip the **RUN/PROG** switch on the DE2-115 board to the **PROG** position. Press **Start** in the window in Figure 50. An LED on the board will light up when the configuration data has been downloaded successfully.

### 8 Testing the Designed Circuit

Having downloaded the configuration data into the FPGA device, you can now test the implemented circuit. Flip the **RUN/PROG** switch to **RUN** position. Try all four valuations of the input variables **x1** and **x2**, by setting the corresponding states of the switches **SW1** and **SW0**. Verify that the circuit implements the truth table in Figure 12.

If you want to make changes in the designed circuit, first close the Programmer window. Then make the desired changes in the Block Diagram/Schematic file, compile the circuit, and program the board as explained above.

Copyright © 2010 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, mask work rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services. This document is being provided on an "as-is" basis and as an accommodation and therefore all warranties, representations or guarantees of any kind (whether express, implied or statutory) including, without limitation, warranties of merchantability, non-infringement, or fitness for a particular purpose, are specifically disclaimed.