The Spartan-3E Tutorial 1: Introduction to FPGA Programming

Version 2.0

Author: Jasmine Banks

Acknowledgements

Parts of this tutorial are based on an earlier version written for Project Navigator version 9.2, written by Michael Bakker, Matthew Grace and Warwick Kilroy, as part of ENB345 – Advanced Design in 2008.

Glossary

IOB	Input/Output Block
LUT	Look Up Table
UUT	Unit Under Test
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

Table of Contents

page

Acknowledgements 3
Glossary 5
1.0 Introduction 15
1.1 Design Functionality 15
1.2 Relevant Documentation 15
1.3 Scope 15
1.4 Changes in this Version 15
2.0 Equipment 19
3.0 Procedure 19
3.1 Startup 19
3.2 Additional Help 20
3.3 Creating a New Project 21
3.4 Adding a New VHDL Source 24
3.5 Editing the VHDL Source 29
3.6 Syntax Checking 31
3.7 Pin Assignment 33
3.8 Synthesize, Translate, Map and Place & Route 38
3.9 Design Summary 41
3.10 Download Design to Board 44
3.11 Running the Program on the Spartan-3E Board 57
4.0 Using a TestBench and Simulating the Design 59
4.1 Adding a TestBench 59
4.2 top_level.vhd – TestBench Code 63
4.3 Editing the TestBench 66
4.4 Running the Simulator 69
5.0 Further Information 75
6.0 References 77
Appendix A – top_level.vhd 79
Appendix B – top_level_tb.vhd 81

List of Figures

Page

Figure 2.1: View of the Project Navigator software while running ………….......................... 17

Figure 2.2: Spartan-3E Development Board …………………………….......................... 18

Figure 3.1: Project Navigator Software Startup Window ……………………………………. 19

Figure 3.2: Help Menu ……………..……………………………………………….………… 20

Figure 3.3: New Project Wizard, Create New Project Page ……………………..…………… 21

Figure 3.4: New Project Wizard, Project Settings Page ………….…………………...……… 22

Figure 3.5: New Project Wizard, Project Summary Page ….……..………………………….. 23

Figure 3.6: Adding a new source file to the project …….……….....……...…………………… 24

Figure 3.7: New Source Wizard, Select Source Type ……………………………...………..… 24

Figure 3.8: New Source Wizard, Define Module …….……………………………...……….. 26

Figure 3.9: New Source Wizard, Summary ………………...………………………………… 27

Figure 3.10: New source file top_level.vhd is now displayed in a tab ………………….….… 28

Figure 3.11: top_level.vhd, as displayed in Project Navigator, before editing ………….….... 29

Figure 3.12: top_level.vhd, as displayed in Project Navigator, after editing …….………... 30

Figure 3.13: Portion of Project Navigator screen with Synthesize – XST expanded ….....….. 31

Figure 3.14: A green tick next to Check Syntax shows that no errors were found ……….…. 32

Figure 3.15: A red cross next to Check Syntax indicates that an error was found ………….... 33

Figure 3.16: Portion of Project Navigator screen, with User Constraints expanded ….….…… 35

Figure 3.17: Dialog Box asking if you wish to create an Implementation Constraint File …... 35

Figure 3.18: Initial appearance of PlanAhead window ……………………………….…… 36

Figure 3.19: I/O Ports displayed in a separate window ………….…………………………… 37

Figure 3.20: I/O Ports window with individual ports expanded ……………………..……..… 37

Figure 3.21: I/O Ports window with values filled in 38

Figure 3.22: Portion of Project Navigator screen, with Implement Design expanded ……..…. 39

Figure 3.23: Portion of Project Navigator screen, with Implement Design expanded, after Translate, Map and Place & Route have successfully been run …….......….. 40

Figure 3.24: Viewing the Design Summary tab ………………………………………………. 41

Figure 3.25: Design Summary tab …………………………………………………………. 42

Figure 3.26: Design Summary tab with scroll bar pulled down …………………..……….. 43

Figure 3.27: Portion of Project Navigator screen, with Implement Design expanded ......…… 44

Figure 3.28: Portion of Project Navigator screen, after Generate Programming File has successfully been run …………………………………………….….....……. 45

Figure 3.29: The initial iMPACT window ……………………….…………………….……… 46

Figure 3.30: iMPACT window, after double-clicking on Boundary Scan …….….…….……. 47

Figure 3.31: iMPACT window, showing Initialize Chain selected …..………...……………. 48

Figure 3.32: iMPACT window, assign configuration files ………..………..…..………. 49

Figure 3.33: iMPACT window, assigning the configuration file for the xc3e500e …......…… 50

Figure 3.34: iMPACT window, dialog box asking if we wish to attach an SPI or BPI PROM. 51

Figure 3.35: iMPACT window, bypassing the xcf04s ……………….………………...……. 52

Figure 3.36: iMPACT window, bypassing the xc2c64a ………..………………...……….. 53

Figure 3.37: iMPACT window, Device Programming Properties dialog box ……....……….. 54

Figure 3.38: iMPACT window, showing the device chain ……………..…….……..……. 55

Figure 3.39: iMPACT window, options which appear when right clicking on the xc3s500e .. 55

Figure 3.40: iMPACT window, after the program has been successfully downloaded to the Spartan-3E board ……………………………….……………...…….. 56

Figure 3.41: The Spartan-3E board with the program running …..………………......……… 57

Figure 4.1: The New Source Wizard with values filled in for setting up a test bench ……..… 59

Figure 4.2: New Source Wizard, Associate Source window…………………..…………… 60

Figure 4.3: New Source Wizard, Summary window …………………………….……… 61

Figure 4.4: Simulation view …………………………………………………………….. 62

Figure 4.5: top_level_tb.vhd – entity …………………………………………..…..……… 63

Figure 4.6: Declaration of the top_level component in the top_level_tb architecture .………. 63

Figure 4.7: Signal declarations in the top_level_tb architecture ..……………………………. 64

Figure 4.8: Instantiation of the Unit Under Test (top_level) in the top_level_tb architecture... 64

Figure 4.9: <clock>_process in the top_level_tb architecture ....…… 64

Figure 4.10: Declaration of top_level component ……………………………………….……… 65

Figure 4.11: Commenting out several lines of code at once ……………………….…..…….. 66

Figure 4.12: Expected input and output waveforms …………………….…….….……..……. 68

Figure 4.13: Portion of Project Navigator screen, with ISim Simulator expanded.…..………. 69

Figure 4.14: Portion of Project Navigator screen, with Behavioral Check Syntax selected …. 70

Figure 4.15: Portion of Project Navigator screen, with Simulate Behavioral Model, and
Process Properties selected .…………………………………….………………. 70

Figure 4.16: Changing Simulation Run Time ………………………………….………….. 71

Figure 4.17: ISim Window .………………….……………………………………..…....……. 72

Figure 4.18: Simulation waveform display …………..……………….....................……….. 72

List of Tables

Page

Table 1.1: Functionality implemented by this tutorial .…………………………….……… 15

Table 3.1: Input/output ports of the top_level entity …………...…………………………….. 34

Table 3.2: Values to enter in the I/O Ports window……………………………………....... 37

Table 3.3: Design Summary tab ………………………………………………………… 42

1.0 Introduction

This tutorial is designed to help new users become familiar with using the Spartan-3E board. The tutorial steps through the following:

- Writing a small program in VHDL which carries out simple combinational logic.
- Connecting the program inputs and outputs to the switches, buttons and LEDs on the Spartan-3E board.
- Downloading the program to the Spartan-3E board using the Project Navigator software.
- Simulating the program using the iSim Simulator.

1.1 Design Functionality

The functionality that is implemented by this tutorial is summarised in Table 1. Essentially, four of the LEDs on the board are lit depending on various combinations of slider switches and push buttons that are on.

LED0 ON when	SW0 **OR** SW1
LED1 ON when	SW2 **OR** SW3
LED2 ON when	(SW0 **OR** SW1) **AND** (SW2 **OR** SW3)
LED3 ON when	Push Button 1 **OR** Push Button 2

Table 1.1: Functionality implemented by this tutorial.

1.2 Relevant Documentation

Before commencing the tutorial, it would be helpful to download the **Spartan-3E FPGA Starter Kit Board User Guide** [1]. This describes the pins of the FPGA chip and the settings required to connect them to the various input/output devices on the Spartan-3E board.

1.3 Scope

This tutorial is designed to help the user who is just starting to “get into” the Spartan-3E, by stepping through the process of creating a simple design and getting it running on the board. It is not designed to be a tutorial on VHDL – for help with VHDL syntax, the user can consult with a number of textbooks on the subject, such as [2,3], or find help online.

1.4 Changes in this version

The following have been added or modified in Version 2.0 of this tutorial:

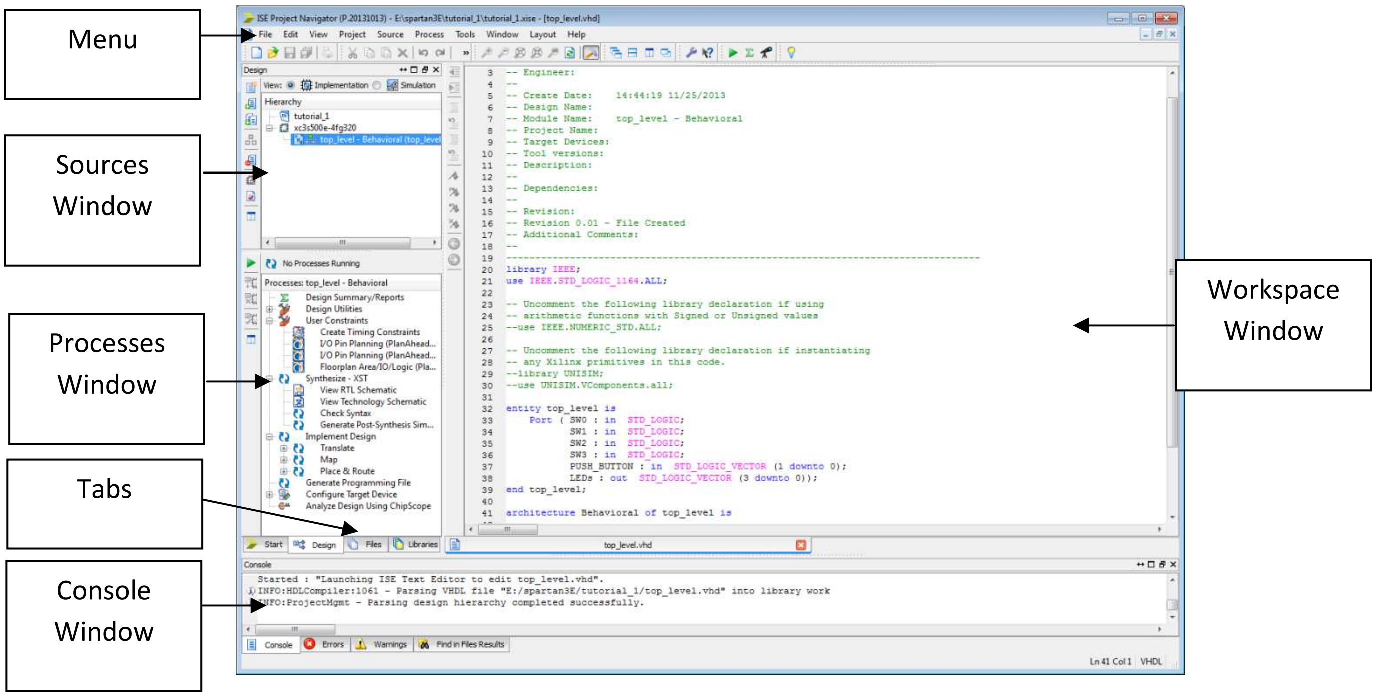
- Re-implemented using Version 14.7 of the ISE Project Navigator Software.
- Added a section on simulation using iSIM.

2.0 Equipment

The following are required to for working through this tutorial:

- The Xilinx ISE Project Navigator software. Version 14.7 was used in this tutorial, but older versions of the software can be used. The software can be downloaded with a free WebPack license from the Xilinx website, http://www.xilinx.com/. The user will need to register and log in.

  Figure 2.1 shows a view of the Project Navigator software while running, and labels some of the components and sub-windows that are typically displayed.


Menu
Sources Window
Processes Window
Tabs
Console Window
Workspace Window
ISE Project Navigator (P.20131013) - E:\spartan3E\tutorial_1\tutorial_1.xise - [top_level.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
tutorial_1
xc3s500e-4fg320
top_level - Behavioral (top_level
No Processes Running
Processes: top_level - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Create Timing Constraints
I/O Pin Planning (PlanAhead...
I/O Pin Planning (PlanAhead...
Floorplan Area/IO/Logic (Pla...
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Sim...
Implement Design
Translate
Map
Place & Route
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start
Design
Files
Libraries
top_level.vhd
Console
Errors
Warnings
Find in Files Results
Ln 41 Col 1
VHDL

```
-- Engineer:
--
-- Create Date:    14:44:19 11/25/2013
-- Design Name:
-- Module Name:    top_level - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity top_level is
    Port ( SW0 : in  STD_LOGIC;
           SW1 : in  STD_LOGIC;
           SW2 : in  STD_LOGIC;
           SW3 : in  STD_LOGIC;
           PUSH_BUTTON : in  STD_LOGIC_VECTOR (1 downto 0);
           LEDs : out  STD_LOGIC_VECTOR (3 downto 0));
end top_level;

architecture Behavioral of top_level is
```

```
 Started : "Launching ISE Text Editor to edit top_level.vhd".
INFO:HDLCompiler:1061 - Parsing VHDL file "E:/spartan3E/tutorial_1/top_level.vhd" into library work
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
```

Figure 2.1: View of the Project Navigator software while running.

- The Spartan 3 Start-up Kit, including the Spartan-3E Development Board, Power Cable, JTAG cable and USB cable for PC connection. The Spartan-3E Development board is shown in Figure 2.2.

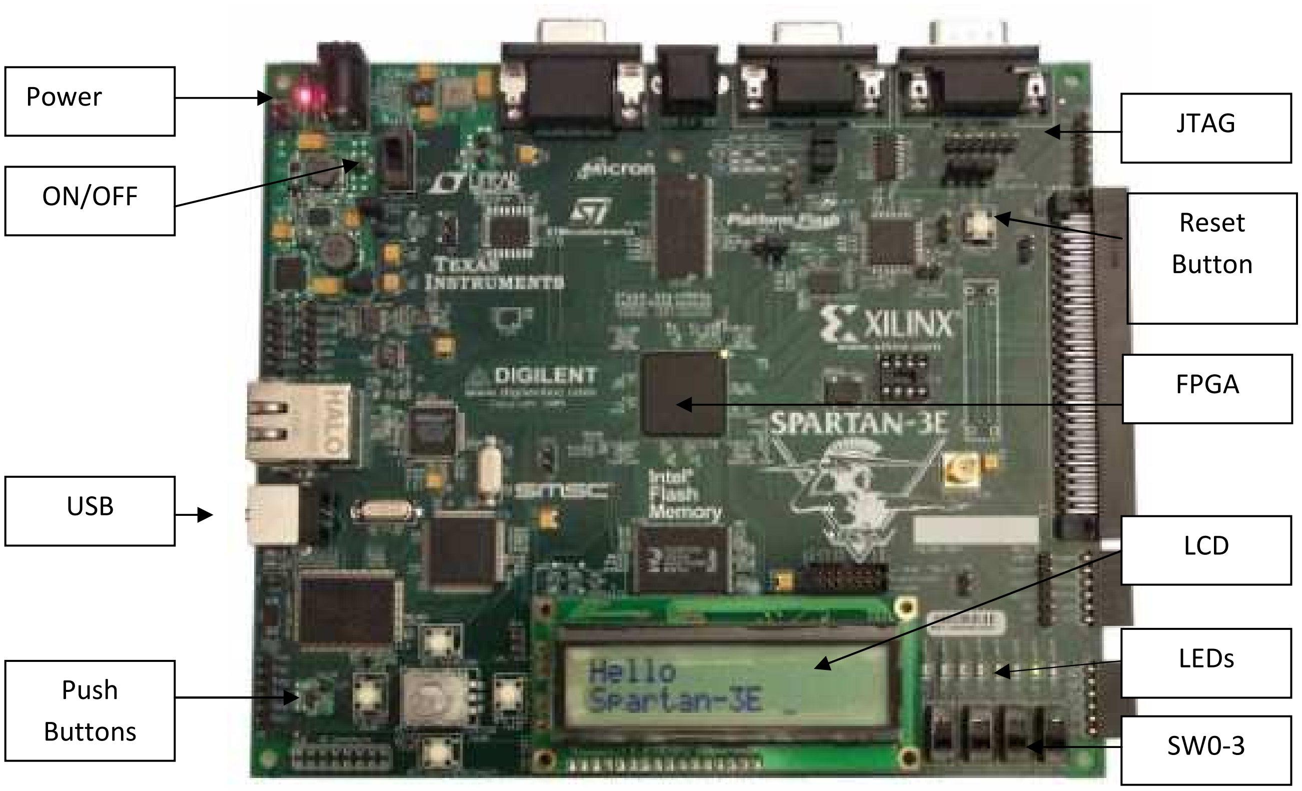


Figure 2.2: Spartan-3E Development Board.

3.0 Procedure

3.1 Startup

Start the Project Navigator software by selecting:

Start→All Programs→XILINX Design Tools→Xilinx ISE Design Suite 14.7→ISE Design Tools→32 bit Project Navigator

or

Start→All Programs→XILINX Design Tools→Xilinx ISE Design Suite 14.7→ISE Design Tools→64 bit Project Navigator

depending on your system. The Xilinx Project Navigator software should start. The initial window which appears on startup should appear as shown in Figure 3.1.

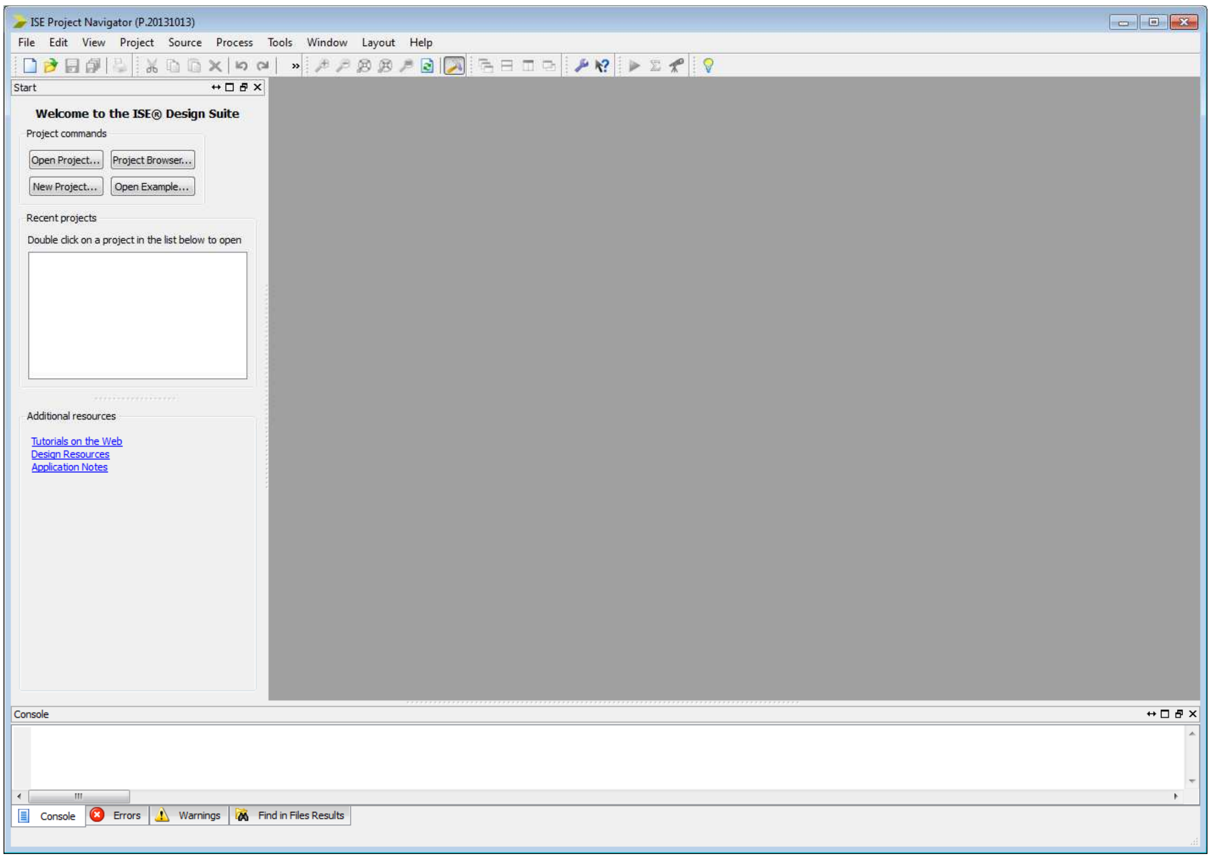


Figure 3.1: Project Navigator Software Startup Window.

3.2 Additional Help

As shown in Figure 3.2, help can be accessed through the **Help** menu in the Project Navigator software.

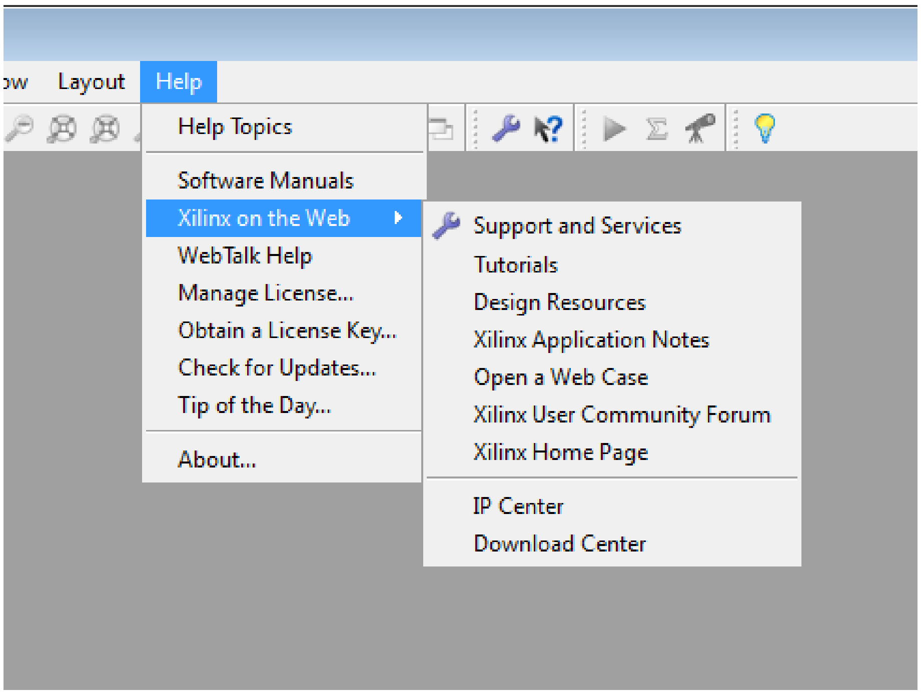


Figure 3.2: Help Menu.

Documentation, software downloads and forums can be accessed at http://www.xilinx.com/support/.

Further example designs using the Spartan-3E Starter Kit can be downloaded from http://www.xilinx.com/products/boards/s3estarter/reference_designs.htm [4].

3.3 Creating a New Project

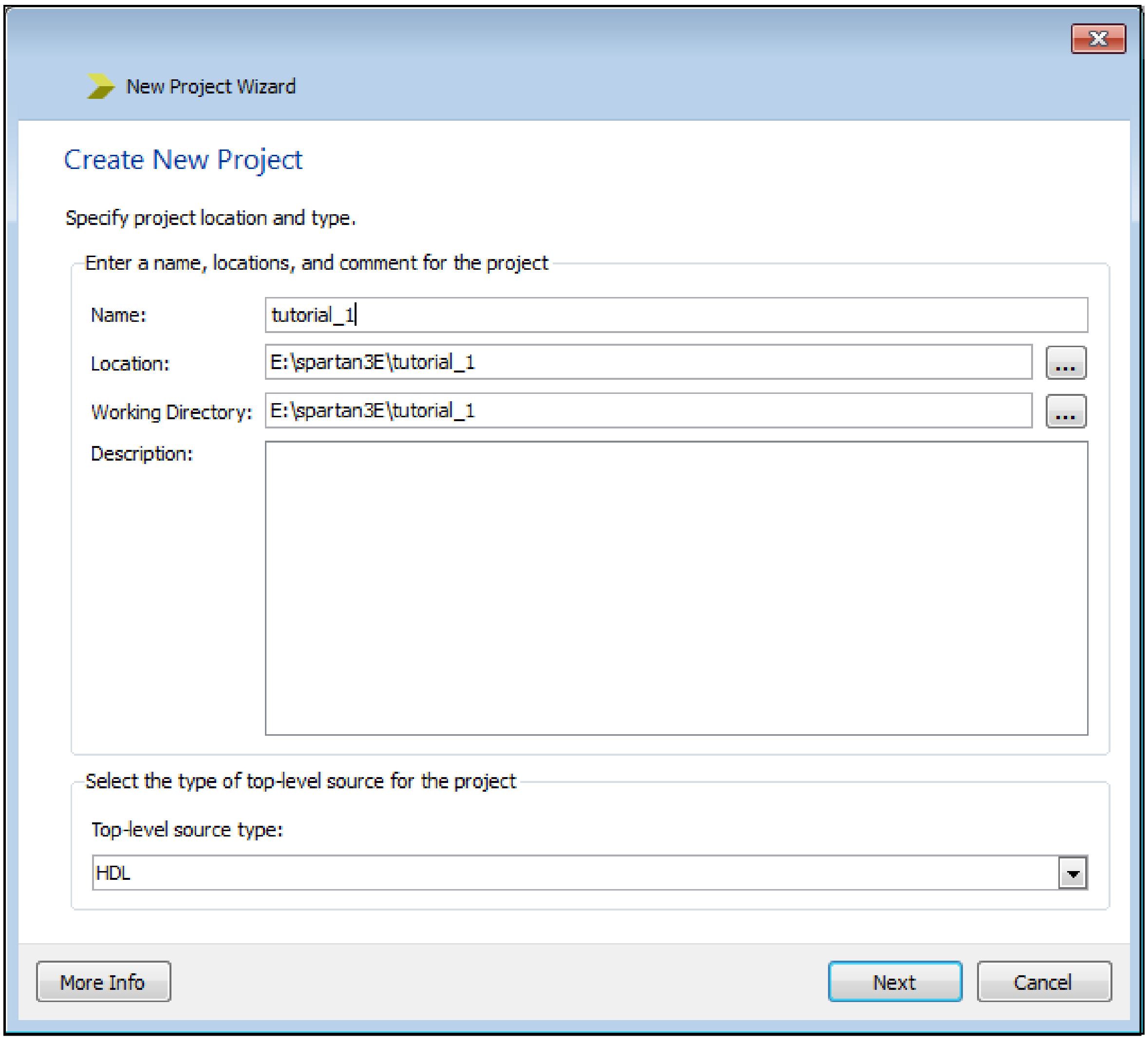
1. Select **File→New Project**. The **New Project Wizard** will appear.

2. Type **tutorial_1** in the **Name:** field.

3. Choose an appropriate **Location:** and **Working Directory:** for your project.

4. Verify that **Top-level source type:** is selected as **HDL**.

5. The properties should now be set as shown in Figure 3.3. Click **Next** to move to the **Project Settings** page.


New Project Wizard
Create New Project
Specify project location and type.
Enter a name, locations, and comment for the project
Description:
Select the type of top-level source for the project
More Info
Next
Cancel

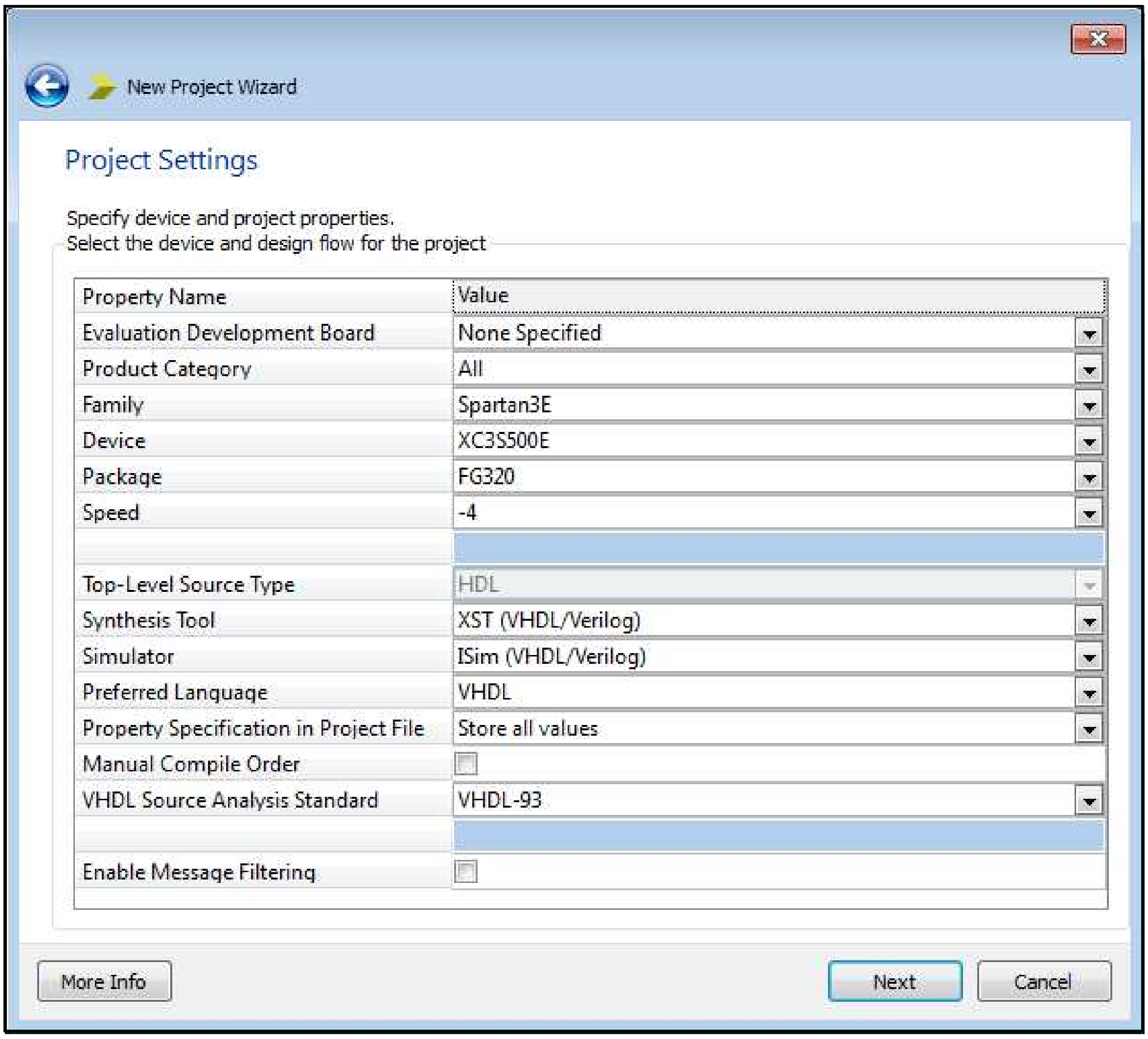
Name: tutorial_1
Location: E:\spartan3E\tutorial_1
Working Directory: E:\spartan3E\tutorial_1
Top-level source type: HDL

Figure 3.3: New Project Wizard, Create New Project Page.

6. Fill in the properties as follows:

- Product Category: **All**
- Family: **Spartan3E**
- Device: **XC3S500E**
- Package: **FG320**
- Speed Grade: **-4**
- Top-Level Source Type: **HDL**
- Synthesis Tool: **XST (VHDL/Verilog)**
- Simulator: **ISim (VHDL/Verilog)**
- Preferred Language: **VHDL**
- Property Specification in Project File: **Store All Values**
- Manual Compile Order: **unchecked**
- VHDL Source Analysis Standard: **VHDL-93**
- Enable Message Filtering: **unchecked**

The properties should now be filled in as shown in Figure 3.4.



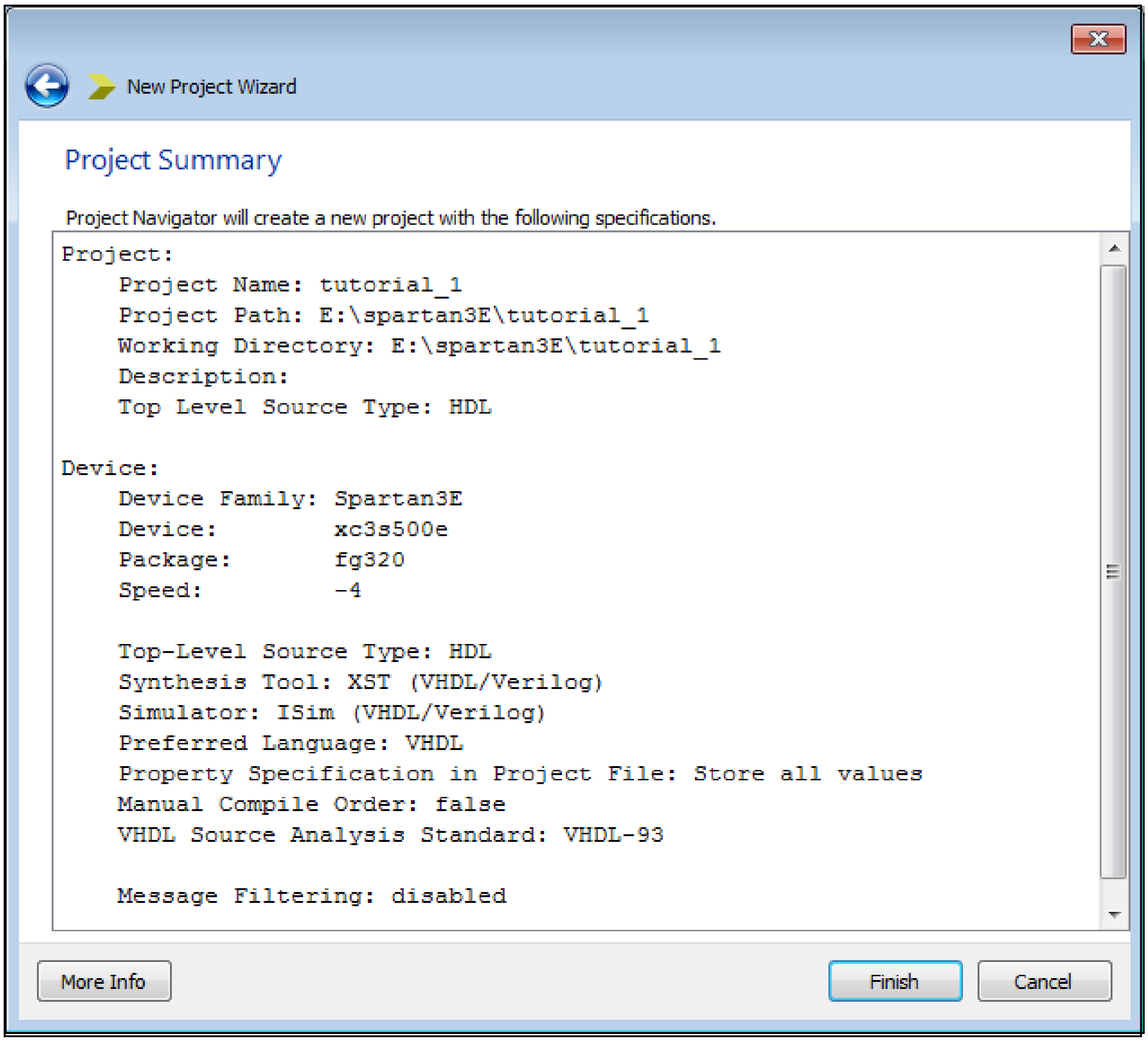
Project Settings

Specify device and project properties.
Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3E
Device	XC3S500E
Package	FG320
Speed	-4
Top-Level Source Type	HDL
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	☐

Figure 3.4: New Project Wizard, Project Settings Page.

7. Click **Next** to move to the **Project Summary** page, which will appear as shown in Figure 3.5.



Project Navigator will create a new project with the following specifications.

```
Project:
    Project Name: tutorial_1
    Project Path: E:\spartan3E\tutorial_1
    Working Directory: E:\spartan3E\tutorial_1
    Description:
    Top Level Source Type: HDL

Device:
    Device Family: Spartan3E
    Device:        xc3s500e
    Package:       fg320
    Speed:         -4

    Top-Level Source Type: HDL
    Synthesis Tool: XST (VHDL/Verilog)
    Simulator: ISim (VHDL/Verilog)
    Preferred Language: VHDL
    Property Specification in Project File: Store all values
    Manual Compile Order: false
    VHDL Source Analysis Standard: VHDL-93

    Message Filtering: disabled
```

Figure 3.5: New Project Wizard, Project Summary Page.

8. Click **Finish** to exit the New Project Wizard.

3.4 Adding a New VHDL Source

1. Select **Project→New Source** as shown in Figure 3.6. The **New Source Wizard** will appear.

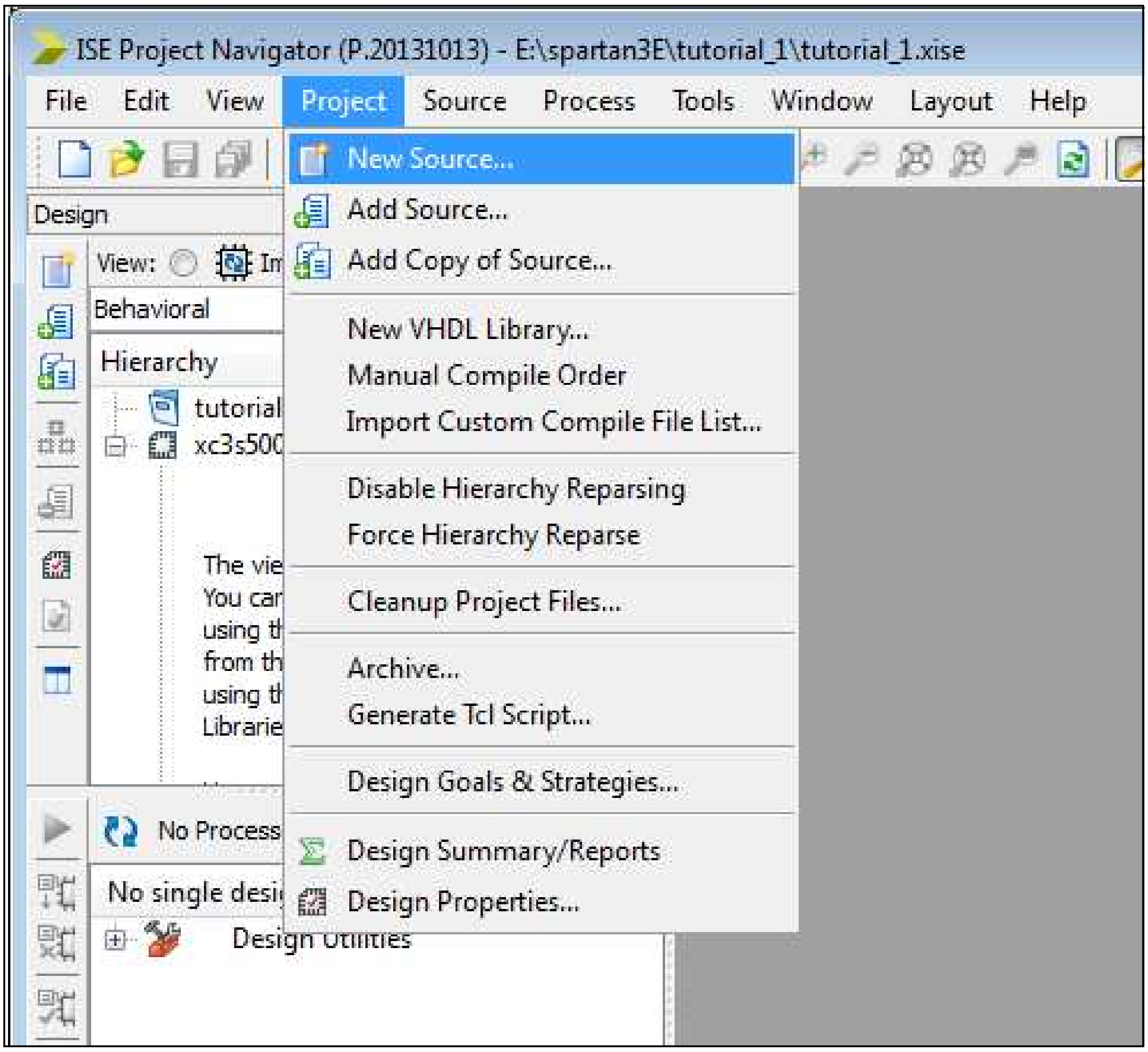


Figure 3.6: Adding a new source file to the project.

2. Select Source Type as **VHDL Module**.

3. Enter the file name as **top_level**, and enter the location of the file (same as the project location entered earlier.

4. Verify that the **Add to project** box is checked. The New Source Wizard should now appear as shown in Figure 3.7.

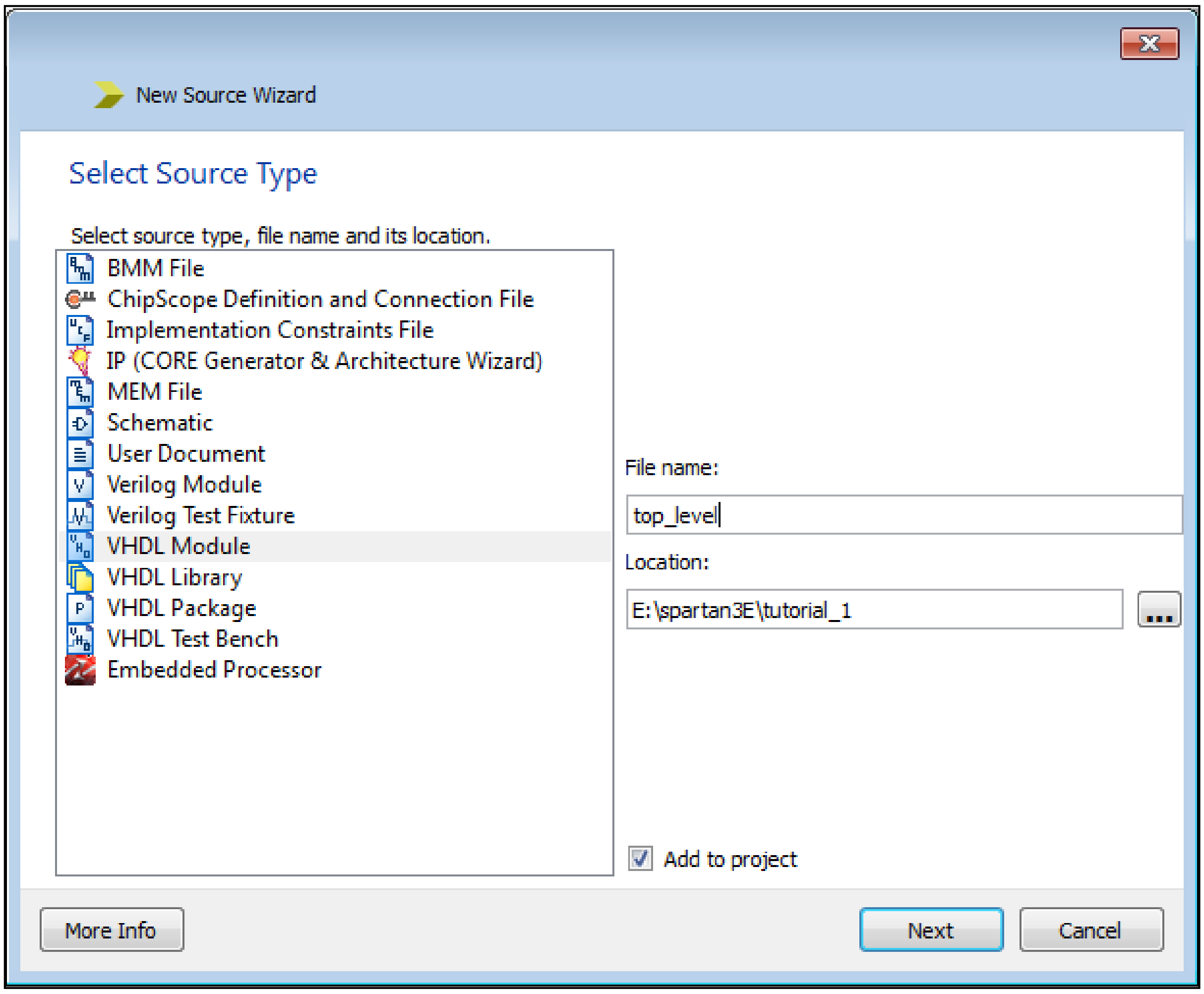


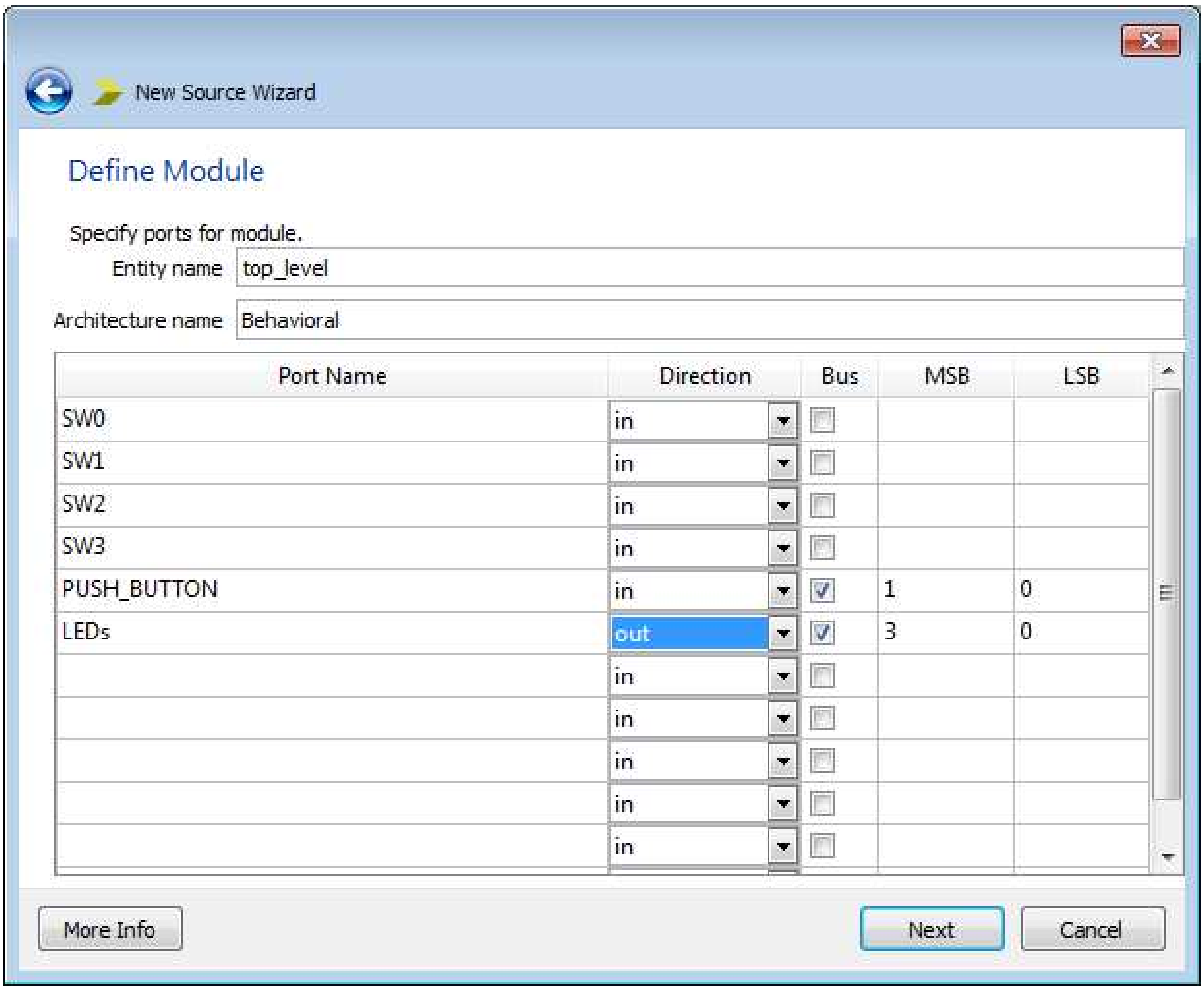
Figure 3.7: New Source Wizard, Select Source Type.

5. Click **Next** to go to the **Define Module** window.

6. Define the ports (inputs and outputs of the design) by entering the information as shown in Figure 3.8. These ports are described as follows:

- SW0-3 will be single bit inputs, and will be connected with the slide switches on the Spartan-3E.
- PUSH_BUTTON will be an input consisting of two bits, and be connected with two of the push button switches on the Spartan 3E. Since this is a multiple bit input, make sure that the **Bus** check box is checked, **MSB** (most significant bit) is set to 1, and **LSB** (least significant bit) is set to 0.
- LEDs will be an output consisting of four bits, and will be connected with four of the LEDs on the Spartan-3E. Since this is a multiple bit output, make sure that the **Bus** check box is checked, **MSB** is set to 3, and **LSB** is set to 0.

7. The **Define Module** window will now appear as shown in Figure 3.8.



Define Module

Specify ports for module.

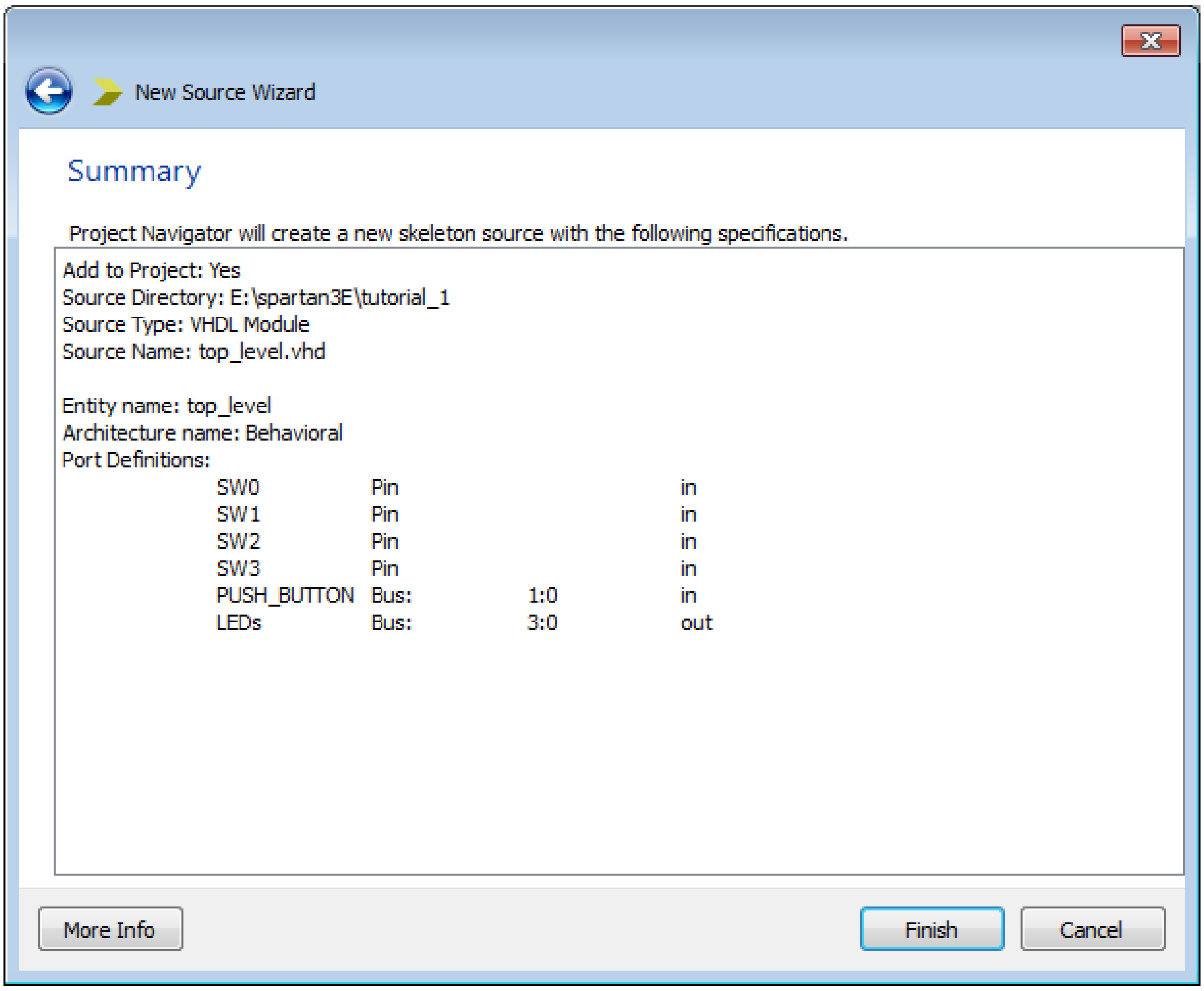
Entity name: top_level

Architecture name: Behavioral

Port Name	Direction	Bus	MSB	LSB
SW0	in	☐		
SW1	in	☐		
SW2	in	☐		
SW3	in	☐		
PUSH_BUTTON	in	✓	1	0
LEDs	out	✓	3	0
	in	☐		
	in	☐		
	in	☐		
	in	☐		
	in	☐		

Figure 3.8: New Source Wizard, Define Module.

8. Click **Next** to move to the **Summary** page, as shown in Figure 3.9.



Summary

Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: E:\spartan3E\tutorial_1
Source Type: VHDL Module
Source Name: top_level.vhd

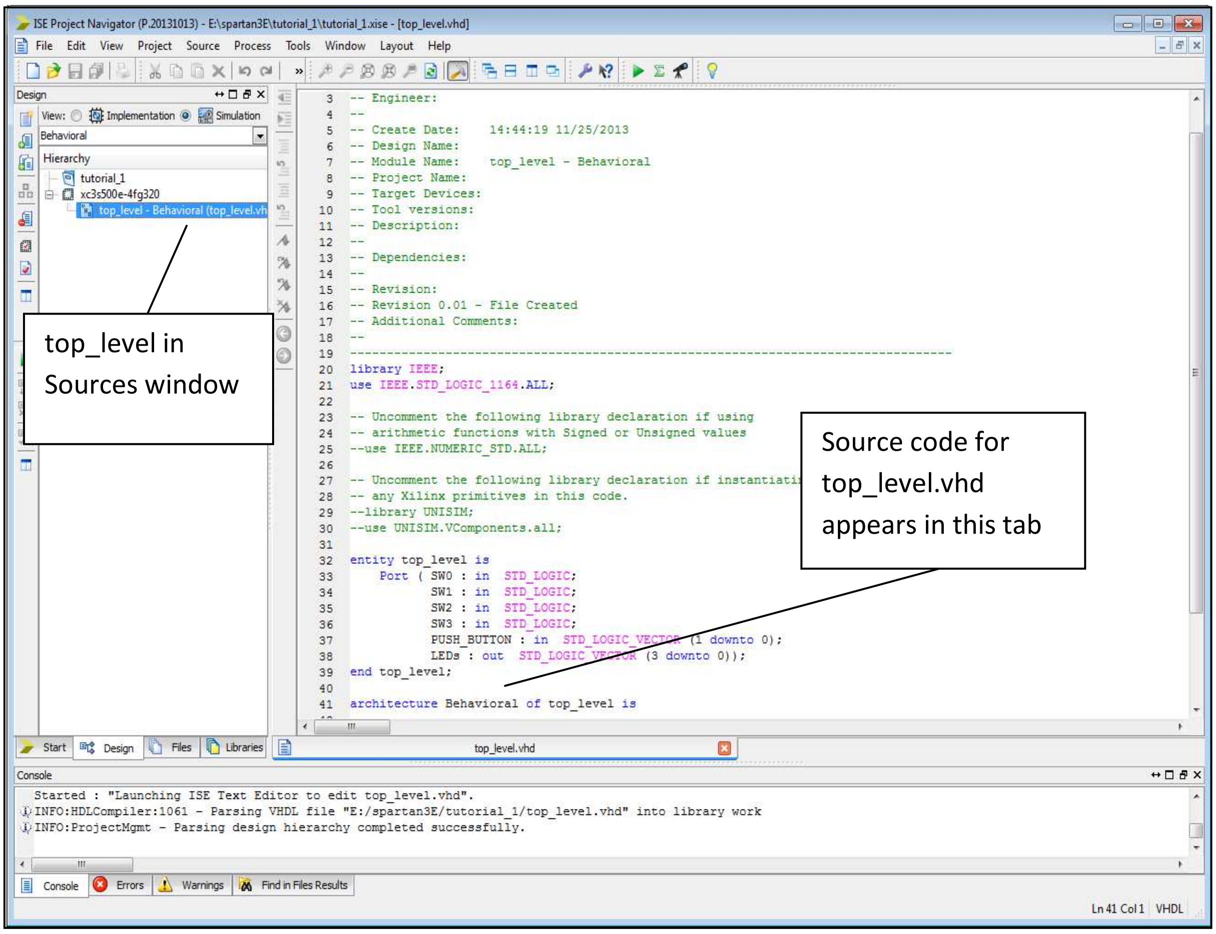
Entity name: top_level
Architecture name: Behavioral
Port Definitions:

SW0	Pin		in
SW1	Pin		in
SW2	Pin		in
SW3	Pin		in
PUSH_BUTTON	Bus:	1:0	in
LEDs	Bus:	3:0	out

Figure 3.9: New Source Wizard, Summary.

9. Click **Finish** to exit the New Source Wizard.

As shown in Figure 3.10, **top_level** will now appear in the Sources window. Double-clicking on **top_level** in the Sources window will display the file, **top_level.vhd** in a tab.


ISE Project Navigator (P.20131013) - E:\spartan3E\tutorial_1\tutorial_1.xise - [top_level.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
tutorial_1
xc3s500e-4fg320
top_level - Behavioral (top_level.vh
top_level in Sources window
Source code for top_level.vhd appears in this tab
Start Design Files Libraries
top_level.vhd
Console
Console Errors Warnings Find in Files Results
Ln 41 Col 1 VHDL

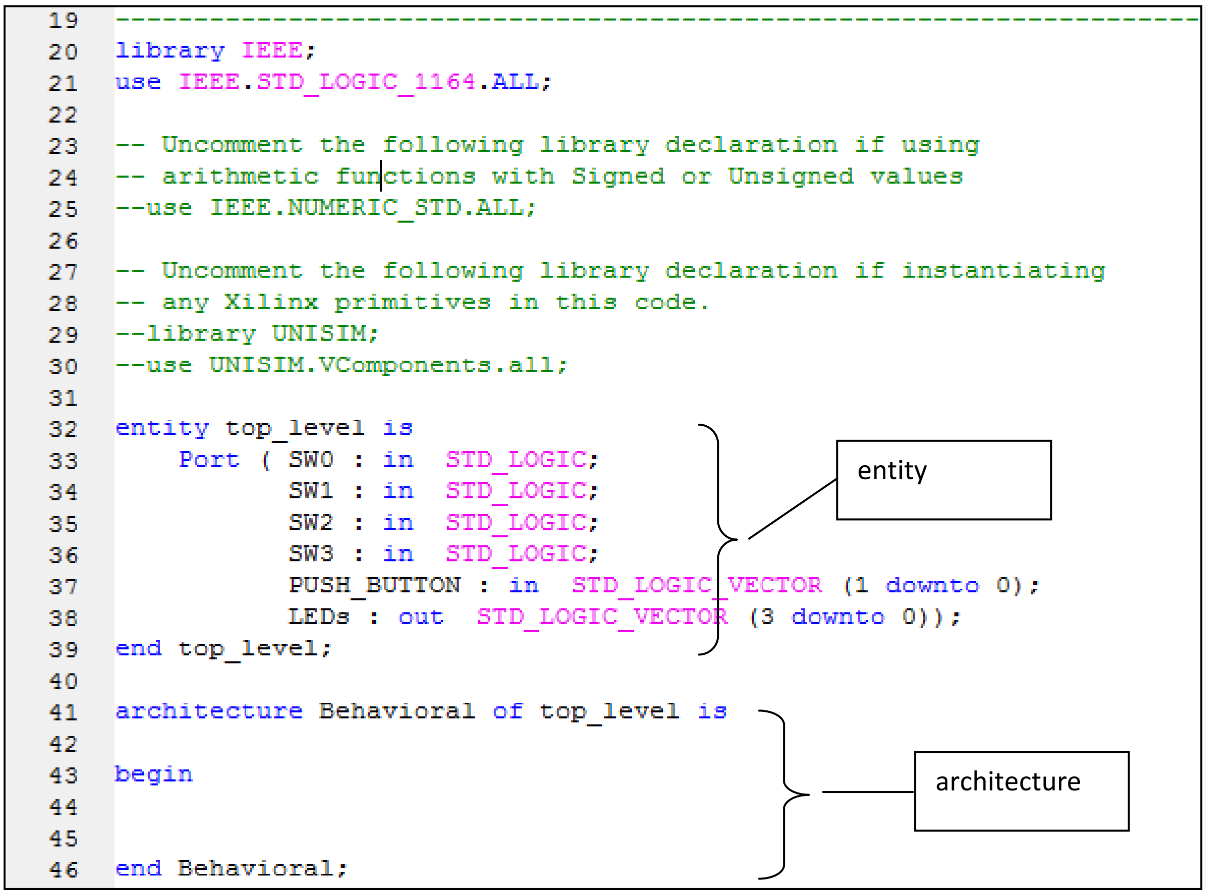
```
-- Engineer:
--
-- Create Date:    14:44:19 11/25/2013
-- Design Name:
-- Module Name:    top_level - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiati
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity top_level is
    Port ( SW0 : in  STD_LOGIC;
           SW1 : in  STD_LOGIC;
           SW2 : in  STD_LOGIC;
           SW3 : in  STD_LOGIC;
           PUSH_BUTTON : in  STD_LOGIC_VECTOR (1 downto 0);
           LEDs : out  STD_LOGIC_VECTOR (3 downto 0));
end top_level;

architecture Behavioral of top_level is
```

```
Started : "Launching ISE Text Editor to edit top_level.vhd".
INFO:HDLCompiler:1061 - Parsing VHDL file "E:/spartan3E/tutorial_1/top_level.vhd" into library work
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
```

Figure 3.10: New source file top_level.vhd is now displayed in a tab.

3.5 Editing the VHDL Source

In Section 3.4, a new VHDL source file, **top_level.vhd** was created. This is actually an ASCII text file that could be edited in any text file editor. However, it is convenient to edit the file using the Project Navigator software. The initial **top_level.vhd**, as displayed in Project Navigator, is shown in Figure 3.11.



```
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity top_level is
    Port ( SW0 : in  STD_LOGIC;
           SW1 : in  STD_LOGIC;
           SW2 : in  STD_LOGIC;
           SW3 : in  STD_LOGIC;
           PUSH_BUTTON : in  STD_LOGIC_VECTOR (1 downto 0);
           LEDs : out  STD_LOGIC_VECTOR (3 downto 0));
end top_level;

architecture Behavioral of top_level is

begin


end Behavioral;
```

Figure 3.11: top_level.vhd, as displayed in Project Navigator, before editing.

In Figure 3.11, it can be seen that Project Navigator colour codes the text of VDHL files, to make them easier to read. Comment lines, which start with "- -" are displayed in green. Reserved words of the VHDL language are displayed in blue, while VHDL types are displayed in red. Everything else is left as black. Note that when code excerpts are discussed in this tutorial, the same colour coding as Project Navigator is used, to assist with readability.

The code in Figure 3.11 contains an **entity** and an **architecture** section. The **entity** section defines the inputs and outputs of this hardware block. In this case these using the have been automatically added using the New Source Wizard. The inputs SW0, SW1, SW2 and SW3 are one bit inputs of type **STD_LOGIC**. The input PUSH_BUTTON is two bits in length, and is of type **STD_LOGIC_VECTOR**. The output LEDs is four bits in length, and is also of type **STD_LOGIC_VECTOR**.

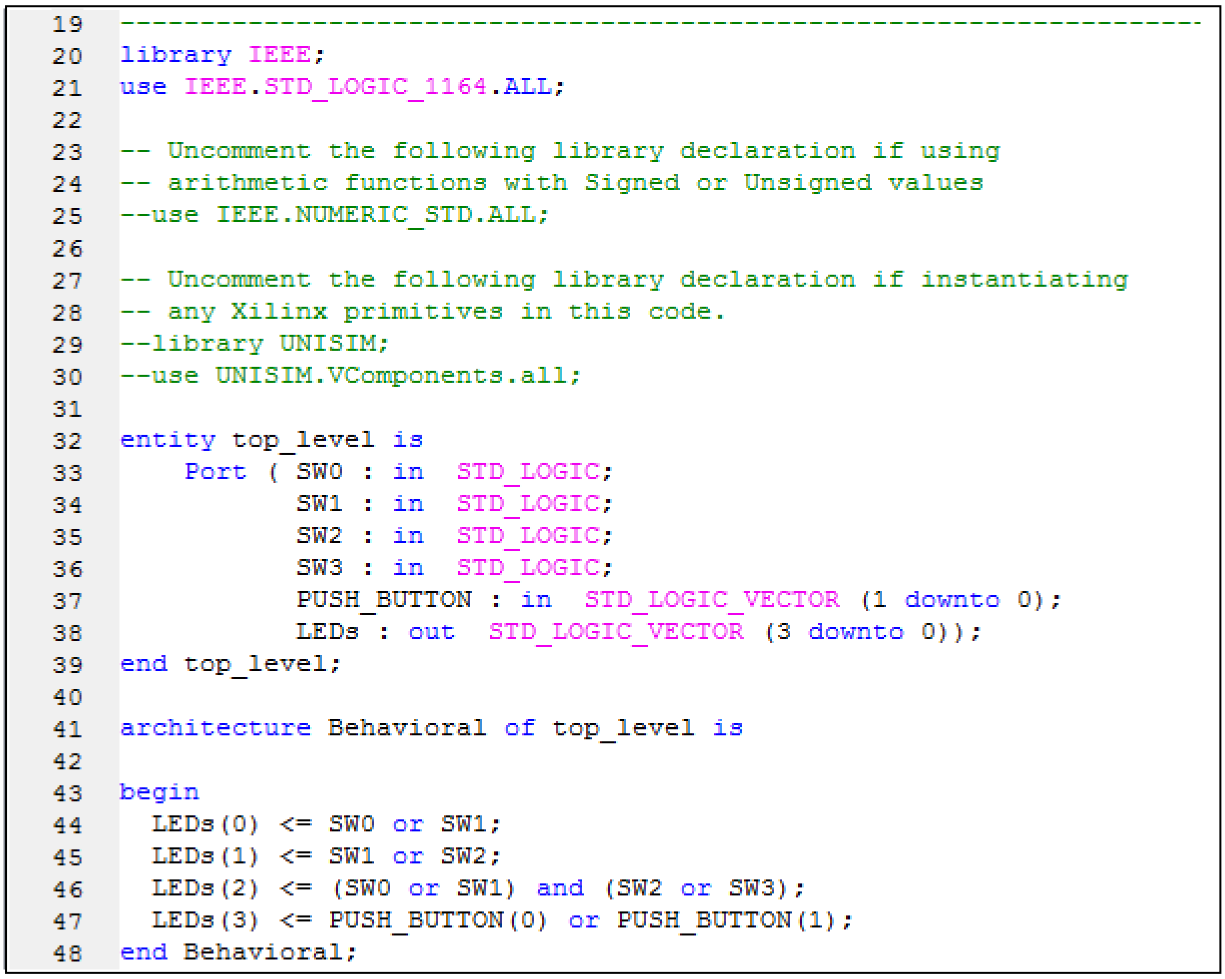
The **architecture** section contains code which implements this hardware. It can be seen that this is initially empty. If this code were downloaded onto the FPGA, it would do nothing. We therefore need to add our own code between the **begin** and **end** statements in the **architecture** block.

Code changes are to be made as follows:

1. Insert the code below between the **begin** and **end** statements in the **architecture** block. This will implement the functionality given in Table 1.1.

```
LEDs(0) <= SW0 or SW1;
LEDs(1) <= SW2 or SW3;
LEDs(2) <= (SW0 or SW1) and (SW2 or SW3);
LEDs(3) <= PUSH_BUTTON(0) or PUSH_BUTTON(1);
```

2. Save the file by selecting **File** → **Save** from the main menu. After editing the top_level.vhd source file will appear as shown in Figure 3.12.



```
--------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity top_level is
    Port ( SW0 : in  STD_LOGIC;
           SW1 : in  STD_LOGIC;
           SW2 : in  STD_LOGIC;
           SW3 : in  STD_LOGIC;
           PUSH_BUTTON : in  STD_LOGIC_VECTOR (1 downto 0);
           LEDs : out  STD_LOGIC_VECTOR (3 downto 0));
end top_level;

architecture Behavioral of top_level is

begin
  LEDs(0) <= SW0 or SW1;
  LEDs(1) <= SW1 or SW2;
  LEDs(2) <= (SW0 or SW1) and (SW2 or SW3);
  LEDs(3) <= PUSH_BUTTON(0) or PUSH_BUTTON(1);
end Behavioral;
```

Figure 3.12: top_level.vhd, as displayed in Project Navigator, after editing.

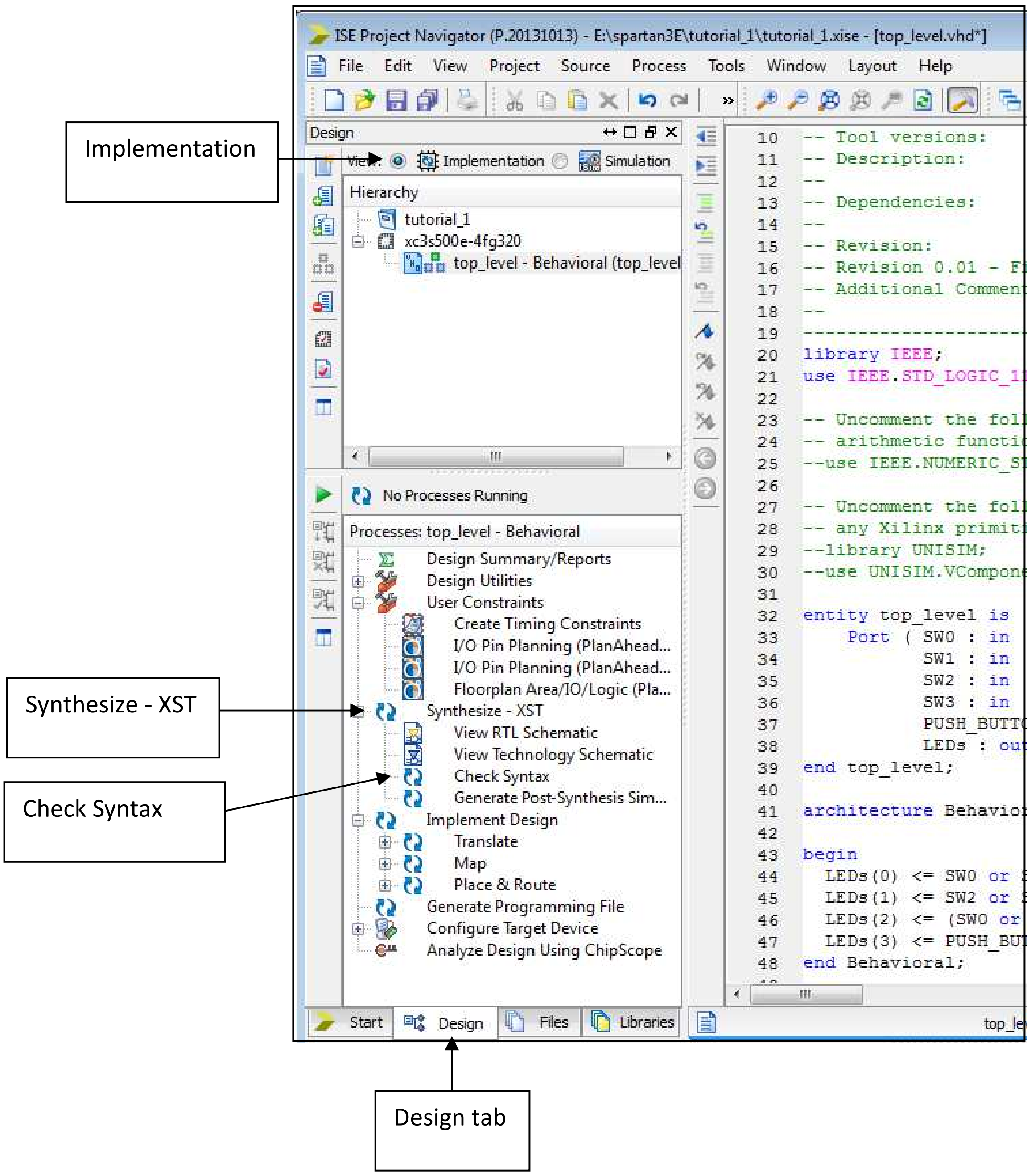
3.6 Syntax Checking

The next step is to do syntax checking, to check that the VHDL code has been entered correctly. The following steps refer to the Project Navigator screen of Figure 3.13.

1. Verify that the **Implementation** check box toward the top left of the screen has been selected.

2. Verify that the **Design** tab has been selected.

3. If it is not already expanded, click on the '+' next to **Synthesize – XST**. This will expand out to show various items, including **Check Syntax**.


ISE Project Navigator (P.20131013) - E:\spartan3E\tutorial_1\tutorial_1.xise - [top_level.vhd*]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
tutorial_1
xc3s500e-4fg320
top_level - Behavioral (top_level
No Processes Running
Processes: top_level - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Create Timing Constraints
I/O Pin Planning (PlanAhead...
I/O Pin Planning (PlanAhead...
Floorplan Area/IO/Logic (Pla...
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Sim...
Implement Design
Translate
Map
Place & Route
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
top_le
Implementation
Synthesize - XST
Check Syntax
Design tab

```
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - Fi
-- Additional Comment
--
--------------------
library IEEE;
use IEEE.STD_LOGIC_1

-- Uncomment the foll
-- arithmetic functi
--use IEEE.NUMERIC_ST

-- Uncomment the foll
-- any Xilinx primiti
--library UNISIM;
--use UNISIM.VCompone

entity top_level is
    Port ( SW0 : in
           SW1 : in
           SW2 : in
           SW3 : in
           PUSH_BUTT
           LEDs : ou
end top_level;

architecture Behavio

begin
  LEDs(0) <= SW0 or
  LEDs(1) <= SW2 or
  LEDs(2) <= (SW0 or
  LEDs(3) <= PUSH_BU
end Behavioral;
```

Figure 3.13: Portion of Project Navigator screen with Synthesize – XST expanded.

4. Double-click on **Check Syntax**. After some time, a green tick should appear beside **Check Syntax**, as shown in Figure 3.14. If instead, a red cross appears, this means a syntax error has been found. Any errors should be fixed before proceeding. For example, in Figure 3.15, a syntax error has been purposely made, in that a space has been left between “SW” and “0” in “SW0”.If a syntax error has been found, an error message should appear in the **Console Window**, which should assist with diagnosing and fixing the problem (see Figure 3.15).

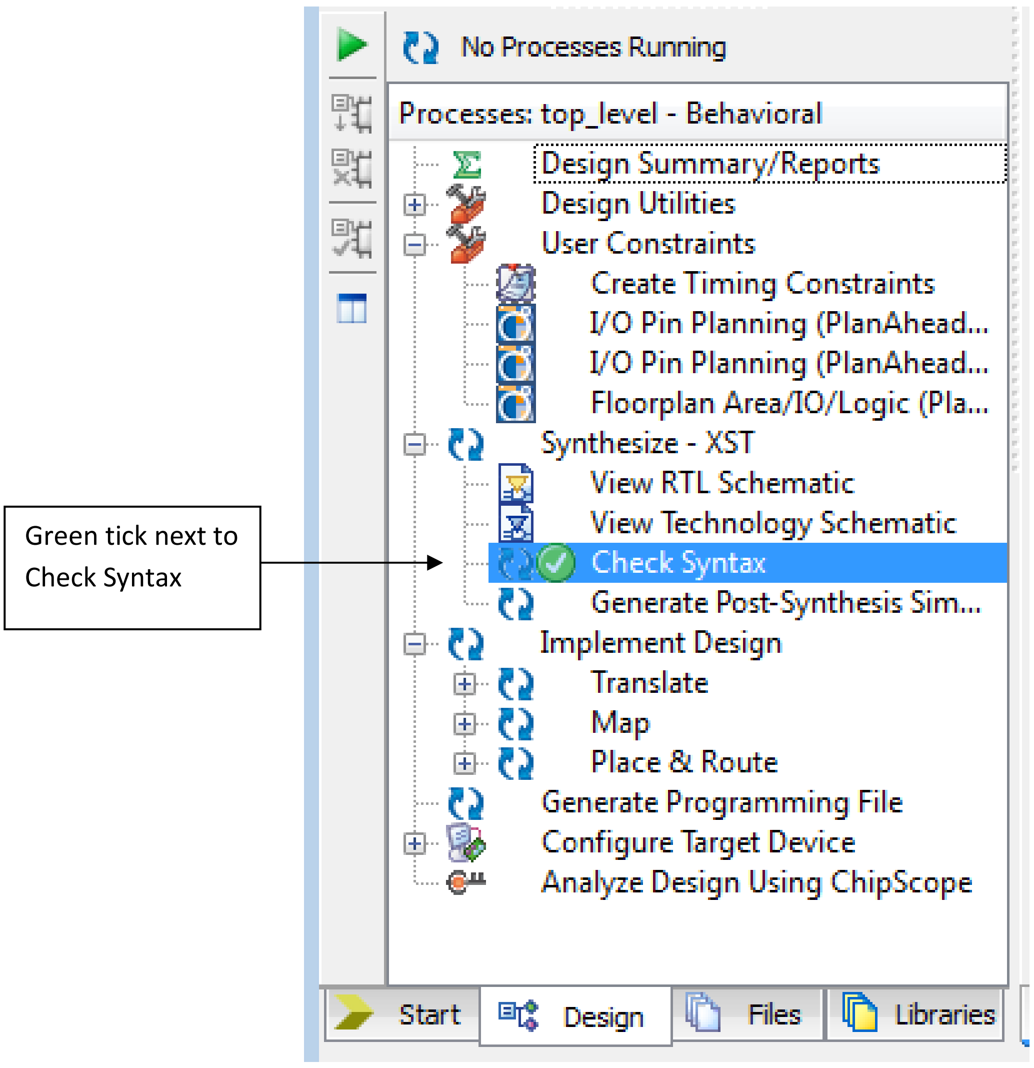
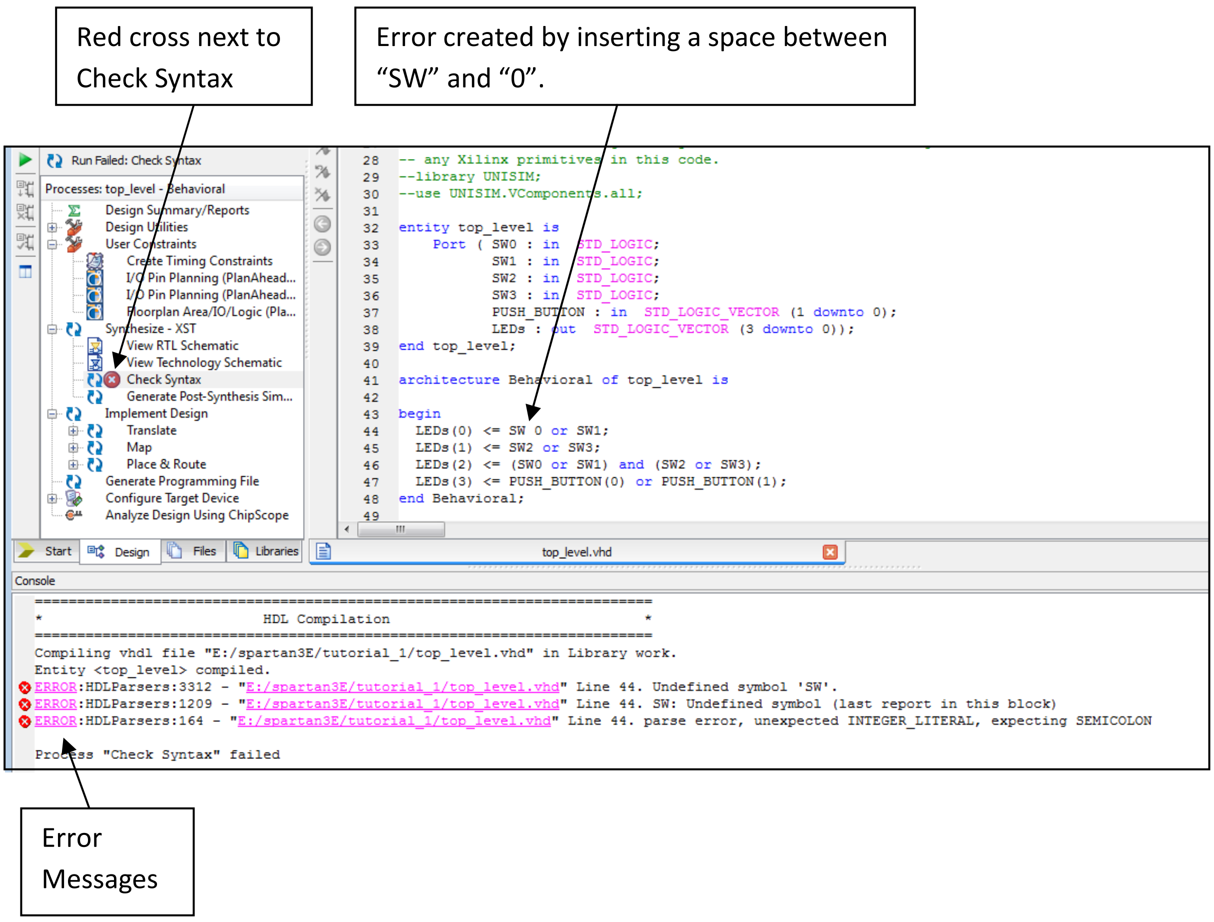

No Processes Running
Processes: top_level - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Create Timing Constraints
I/O Pin Planning (PlanAhead...
I/O Pin Planning (PlanAhead...
Floorplan Area/IO/Logic (Pla...
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Sim...
Implement Design
Translate
Map
Place & Route
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start
Design
Files
Libraries
Green tick next to Check Syntax

Figure 3.14: A green tick next to Check Syntax shows that no errors were found.


Red cross next to Check Syntax
Error created by inserting a space between “SW” and “0”.
Error Messages
Run Failed: Check Syntax
Processes: top_level - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Create Timing Constraints
I/O Pin Planning (PlanAhead...
I/O Pin Planning (PlanAhead...
Floorplan Area/IO/Logic (Pla...
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Sim...
Implement Design
Translate
Map
Place & Route
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start
Design
Files
Libraries
top_level.vhd
Console

```
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity top_level is
33      Port ( SW0 : in  STD_LOGIC;
34             SW1 : in  STD_LOGIC;
35             SW2 : in  STD_LOGIC;
36             SW3 : in  STD_LOGIC;
37             PUSH_BUTTON : in  STD_LOGIC_VECTOR (1 downto 0);
38             LEDs : out  STD_LOGIC_VECTOR (3 downto 0));
39  end top_level;
40
41  architecture Behavioral of top_level is
42
43  begin
44    LEDs(0) <= SW 0 or SW1;
45    LEDs(1) <= SW2 or SW3;
46    LEDs(2) <= (SW0 or SW1) and (SW2 or SW3);
47    LEDs(3) <= PUSH_BUTTON(0) or PUSH_BUTTON(1);
48  end Behavioral;
49
```

```
==========================================================================
*                          HDL Compilation                               *
==========================================================================
Compiling vhdl file "E:/spartan3E/tutorial_1/top_level.vhd" in Library work.
Entity <top_level> compiled.
ERROR:HDLParsers:3312 - "E:/spartan3E/tutorial_1/top_level.vhd" Line 44. Undefined symbol 'SW'.
ERROR:HDLParsers:1209 - "E:/spartan3E/tutorial_1/top_level.vhd" Line 44. SW: Undefined symbol (last report in this block)
ERROR:HDLParsers:164 - "E:/spartan3E/tutorial_1/top_level.vhd" Line 44. parse error, unexpected INTEGER_LITERAL, expecting SEMICOLON

Process "Check Syntax" failed
```

Figure 3.15: Example where an error was purposely introduced. A red cross next to Check Syntax indicates that an error was found.

3.7 Pin Assignment

Recall that the VHDL code for the **top_level** entity is:

```
entity top_level is
    Port ( SW0 : in STD_LOGIC;
           SW1 : in STD_LOGIC;
           SW2 : in STD_LOGIC;
           SW3 : in STD_LOGIC;
           PUSH_BUTTON : in STD_LOGIC_VECTOR (1 downto 0);
           LEDs : out STD_LOGIC_VECTOR (3 downto 0));
end top_level;
```

We wish to connect the inputs and outputs of the top_level entity to the switches, buttons and LEDs on the Spartan-3E board. For example, we wish to read inputs SW0, SW1, SW2 and SW3 from the four slider switches. These are connected to pins L13, L14, H18 and N17 of the FPGA chip.

Similarly, we wish to read inputs PUSH_BUTTON(0) and PUSH_BUTTON(1) from two of the push buttons on the board. In this case, we will use the North and East buttons, which are connected to pins V4 and H13 of the FPGA. Finally, we wish to connect the outputs LEDs(0), LEDs(1), LEDs(2) and LEDs(3) to four of the LEDs on the board. In this case we will use the four right-most LEDs, corresponding to pins F12, E12, E11 and F11 respectively.

For each input and output port of the top_level entity, Table 3.1 lists the name of the device on the Spartan-3E board that we wish to connect the port to, a description of what it physically corresponds to (slider switch, push button or LED), and the FPGA pin number that it is connected to. This information comes from the Spartan-3E FPGA Starter Kit Board User Guide [1].

Port name	Spartan-3E board device name	Description	FPGA pin
SW0	SW0	Slider switch	L13
SW1	SW1	Slider switch	L14
SW2	SW2	Slider switch	H18
SW3	SW3	Slider switch	N17
PUSH_BUTTON(0)	BTN_North	Push button	V4
PUSH_BUTTON(1)	BTN_East	Push button	H13
LEDs(0)	LD0	LED	F12
LEDs(1)	LD1	LED	E12
LEDs(2)	LD2	LED	E11
LEDs(3)	LD3	LED	F11

Table 3.1: Input/output ports of the top_level entity; and the name, description and FPGA pin no. of the devices on the Spartan-3E board that the ports will be connected to.

The following steps are used to connect the inputs and outputs to the switches, buttons and LEDs on the Spartan-3E board:

1. As shown in Figure 3.16, if it has not already expanded, click on the '+' next to **User Constraints**. This will expand out to show various items, including **I/O Pin Planning (PlanAhead) – Pre-Synthesis**.

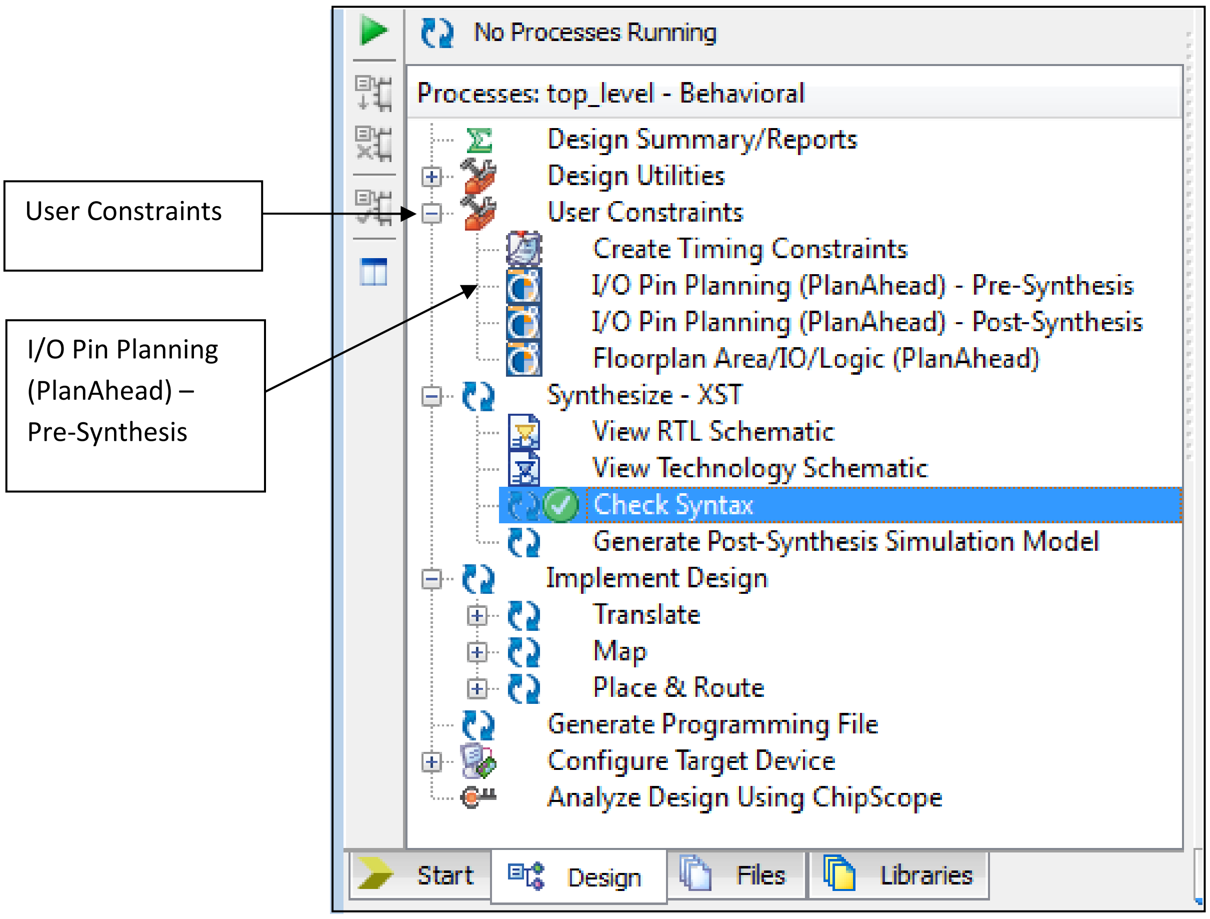
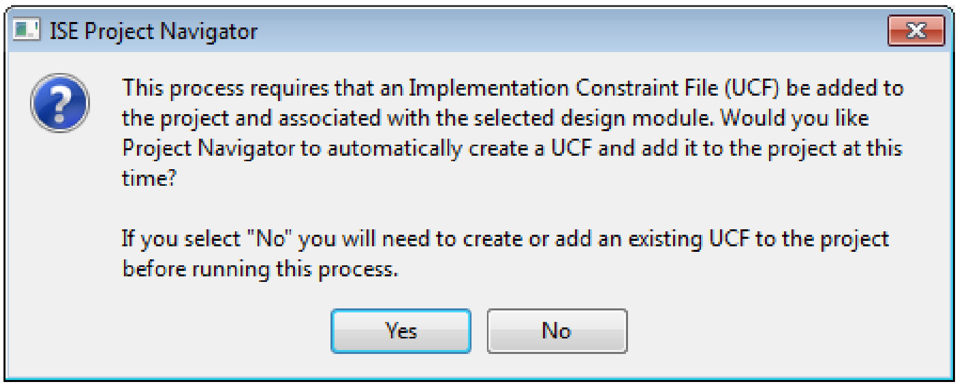

User Constraints
I/O Pin Planning (PlanAhead) – Pre-Synthesis
No Processes Running
Processes: top_level - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Create Timing Constraints
I/O Pin Planning (PlanAhead) - Pre-Synthesis
I/O Pin Planning (PlanAhead) - Post-Synthesis
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Translate
Map
Place & Route
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start
Design
Files
Libraries

Figure 3.16: Portion of Project Navigator screen, with User Constraints expanded.

2. Double-click on I/**O Pin Planning (PlanAhead) – Pre-Synthesis**. The first time this is done, the window of Figure 3.17 will appear, asking whether it is OK to create a UCF file. Click **Yes**.

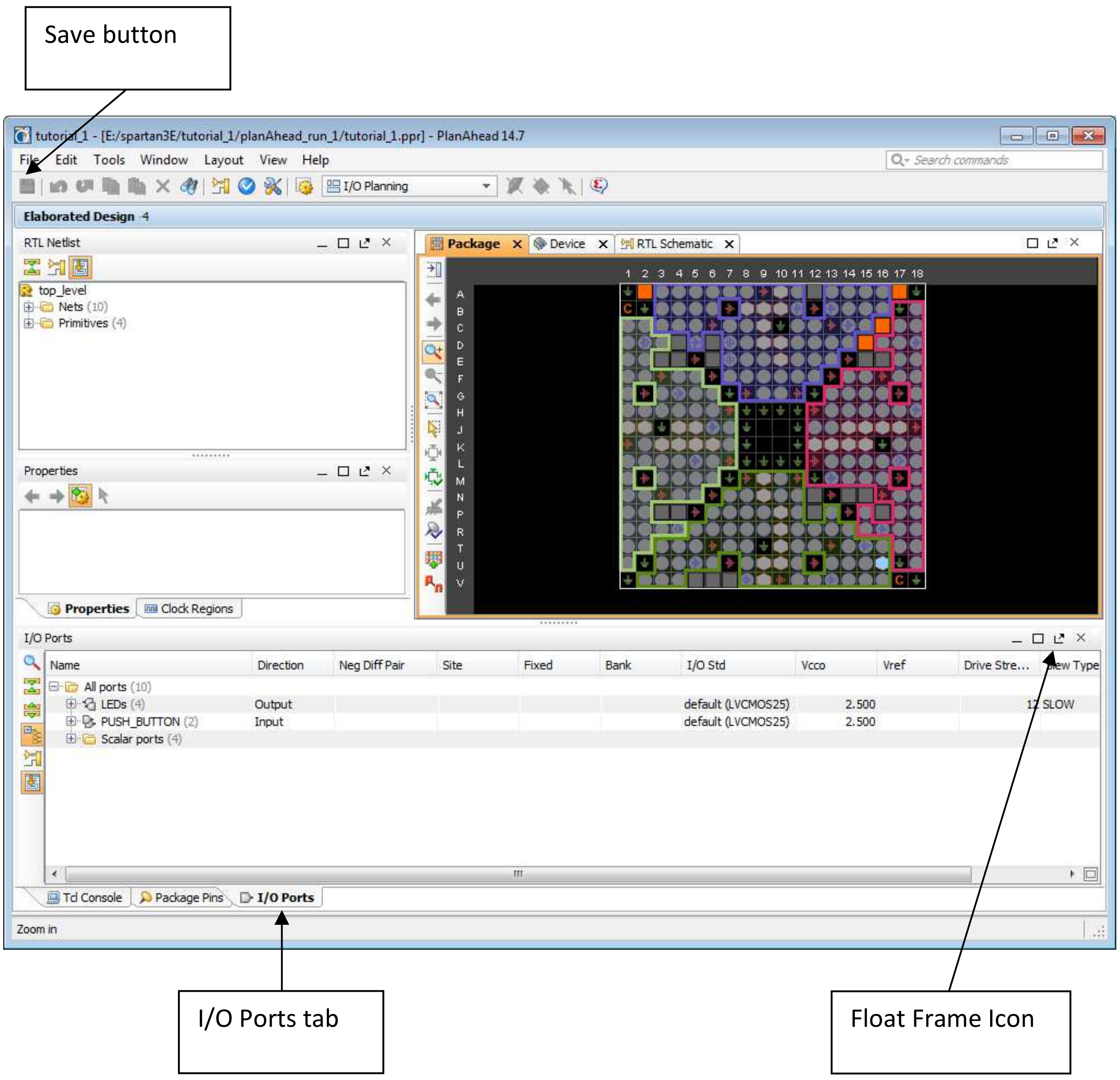

ISE Project Navigator

This process requires that an Implementation Constraint File (UCF) be added to the project and associated with the selected design module. Would you like Project Navigator to automatically create a UCF and add it to the project at this time?

If you select "No" you will need to create or add an existing UCF to the project before running this process.

Figure 3.17: Dialog Box asking if you wish to create an Implementation Constraint File.

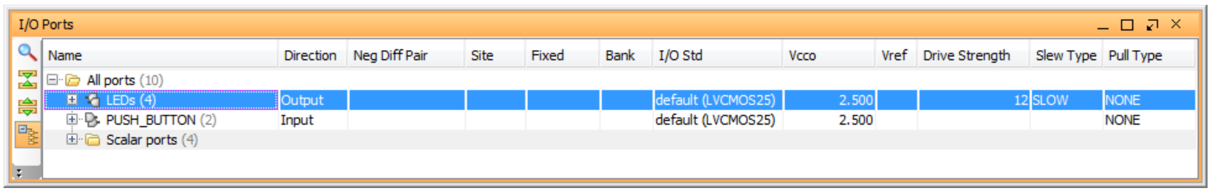
3. After clicking **Yes** in Figure 3.17, the **PlanAhead** window of Figure 3.18 will appear. Click on the **I/O Ports** tab, and then the **float frame** icon. This display the I/O Ports in a separate window, as shown in Figure 3.19.



Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Stre...	Slew Type
All ports (10)										
LEDs (4)	Output					default (LVCMOS25)	2.500		12	SLOW
PUSH_BUTTON (2)	Input					default (LVCMOS25)	2.500			
Scalar ports (4)										

Figure 3.18: Initial appearance of PlanAhead window.

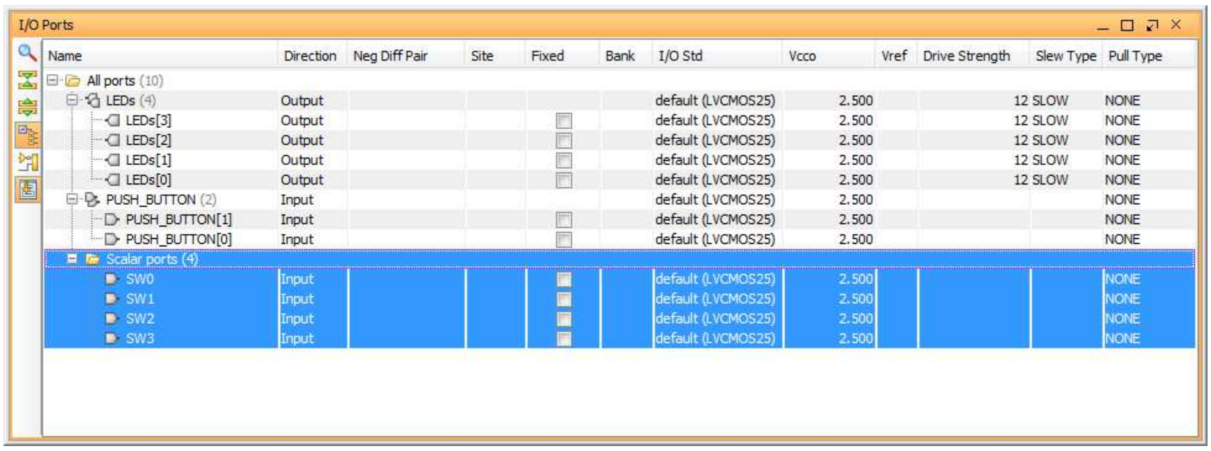
4. Click on the '+' next to **LEDs(4)**, **PUSH_BUTTON(2)** and **Scalar ports** in Figure 3.19. This will display all individual input/outputs as shown in Figure 3.20.



I/O Ports

Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type
All ports (10)											
LEDs (4)	Output					default (LVCMOS25)	2.500		12	SLOW	NONE
PUSH_BUTTON (2)	Input					default (LVCMOS25)	2.500				NONE
Scalar ports (4)											

Figure 3.19: I/O Ports displayed in a separate window.



I/O Ports

Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type
All ports (10)											
LEDs (4)	Output					default (LVCMOS25)	2.500		12	SLOW	NONE
LEDs[3]	Output			☐		default (LVCMOS25)	2.500		12	SLOW	NONE
LEDs[2]	Output			☐		default (LVCMOS25)	2.500		12	SLOW	NONE
LEDs[1]	Output			☐		default (LVCMOS25)	2.500		12	SLOW	NONE
LEDs[0]	Output			☐		default (LVCMOS25)	2.500		12	SLOW	NONE
PUSH_BUTTON (2)	Input					default (LVCMOS25)	2.500				NONE
PUSH_BUTTON[1]	Input			☐		default (LVCMOS25)	2.500				NONE
PUSH_BUTTON[0]	Input			☐		default (LVCMOS25)	2.500				NONE
Scalar ports (4)											
SW0	Input			☐		default (LVCMOS25)	2.500				NONE
SW1	Input			☐		default (LVCMOS25)	2.500				NONE
SW2	Input			☐		default (LVCMOS25)	2.500				NONE
SW3	Input			☐		default (LVCMOS25)	2.500				NONE

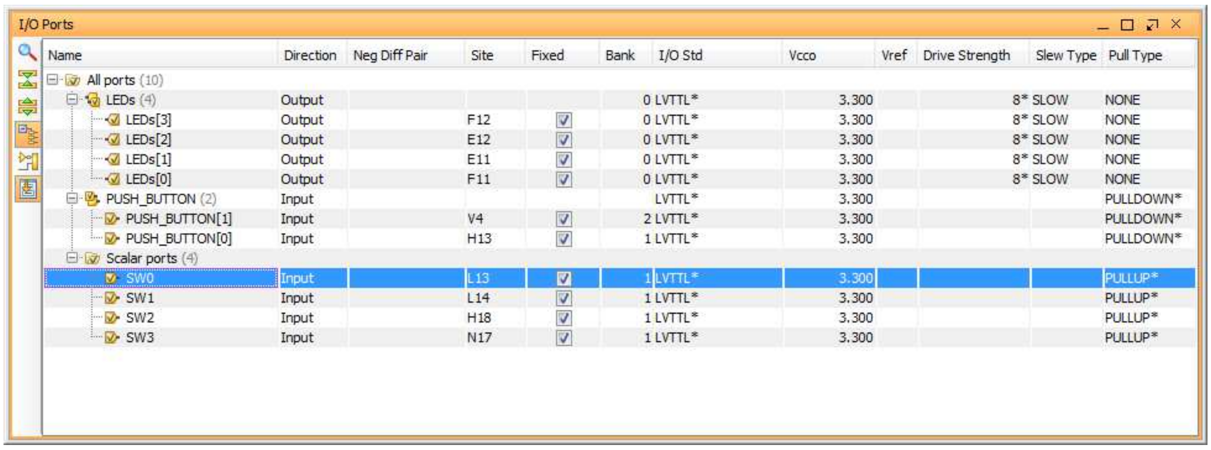
Figure 3.20: I/O Ports window with individual ports expanded.

5. Enter the **Site**, **I/O Std**, **Drive Strength**, **Slew Type** and **Pull Type** columns, with the values given in Table 3.2. The **Bank** and **Vcco** columns will be filled in automatically when the **Site** and **I/O Std** columns respectively are filled in.

Port	Site	I/O Std	Drive Strength	Slew Type	Pull Type
LEDs[0]	F12	LVTTL	8	SLOW	
LEDs[1]	E12	LVTTL	8	SLOW	
LEDs[2]	E11	LVTTL	8	SLOW	
LEDs[3]	F11	LVTTL	8	SLOW	
PUSH_BUTTON[0]	V4	LVTTL			PULLDOWN
PUSH_BUTTON[1]	H13	LVTTL			PULLDOWN
SW0	L13	LVTTL			PULLUP
SW1	L14	LVTTL			PULLUP
SW2	H18	LVTTL			PULLUP
SW3	N17	LVTTL			PULLUP

Table 3.2: Values to enter in the I/O Ports window.

The I/O Ports window should now appear as shown in Figure 3.21.



I/O Ports

Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type
All ports (10)											
LEDs (4)	Output				0	LVTTL*	3.300		8*	SLOW	NONE
LEDs[3]	Output		F12	✓	0	LVTTL*	3.300		8*	SLOW	NONE
LEDs[2]	Output		E12	✓	0	LVTTL*	3.300		8*	SLOW	NONE
LEDs[1]	Output		E11	✓	0	LVTTL*	3.300		8*	SLOW	NONE
LEDs[0]	Output		F11	✓	0	LVTTL*	3.300		8*	SLOW	NONE
PUSH_BUTTON (2)	Input					LVTTL*	3.300				PULLDOWN*
PUSH_BUTTON[1]	Input		V4	✓	2	LVTTL*	3.300				PULLDOWN*
PUSH_BUTTON[0]	Input		H13	✓	1	LVTTL*	3.300				PULLDOWN*
Scalar ports (4)											
SW0	Input		L13	✓	1	LVTTL*	3.300				PULLUP*
SW1	Input		L14	✓	1	LVTTL*	3.300				PULLUP*
SW2	Input		H18	✓	1	LVTTL*	3.300				PULLUP*
SW3	Input		N17	✓	1	LVTTL*	3.300				PULLUP*

Figure 3.21: I/O Ports window with values filled in.

6. Click the **Save** button in the PlanAhead window (location of Save button is shown in Figure 3.18), to save the entered pins. The PlanAhead window can be closed at this stage.

3.8 Synthesize, Translate, Map, and Place & Route

The next stage involves going through the Synthesize, Translate, Map and Place and Route Steps. These steps are carried out by the Project Navigator software, and are briefly described as follows:

- Synthesize: generates netlists for each source file.
- Translate: merges multiple files into a single netlist.
- Map: the design is mapped to slices and I/O blocks.
- Place and Route: works out how the design is to be placed on the chip and components connected.

1. As shown in Figure 3.22, if it is not already expanded, click on the '+' next to **Implement Design**. This will expand out to show the **Translate**, **Map** and **Place & Route** stages.

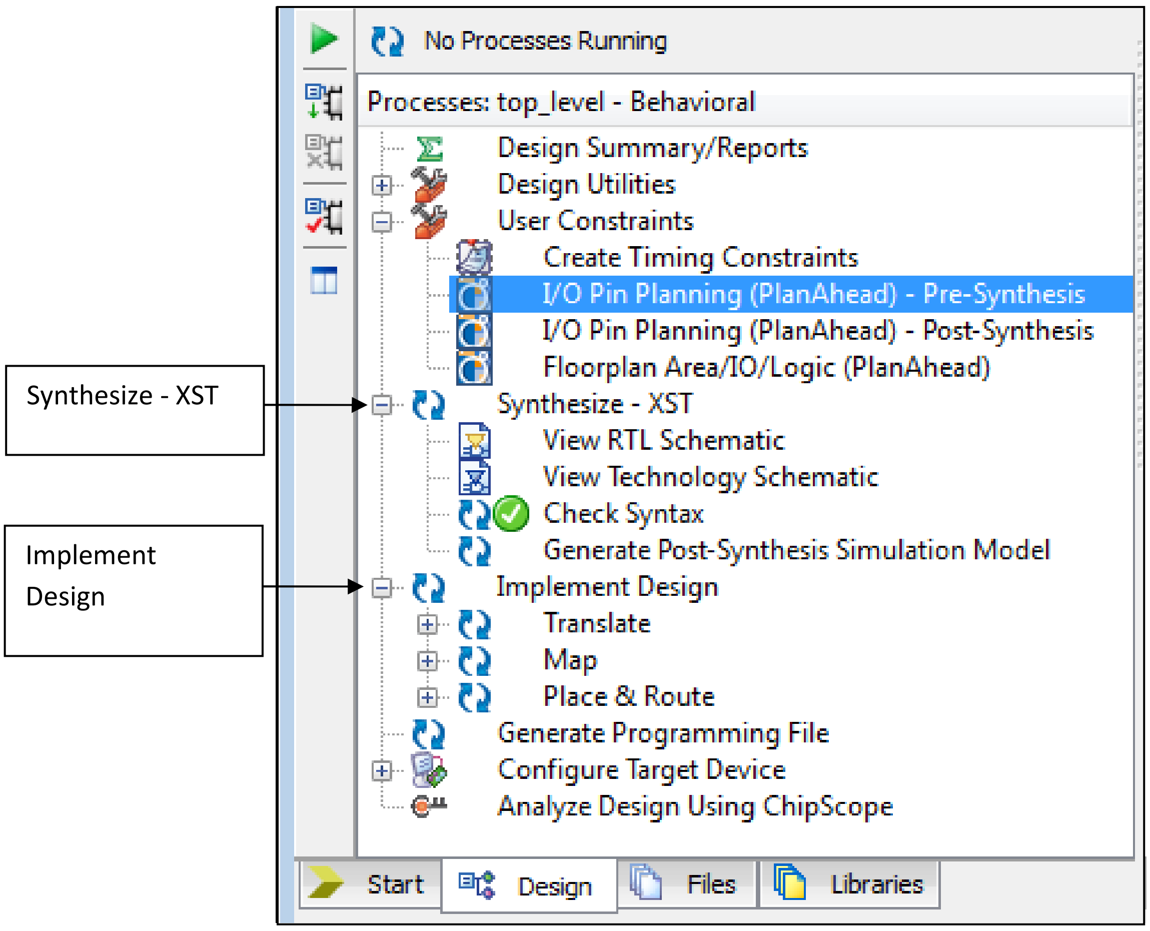

Synthesize - XST
Implement Design
No Processes Running
Processes: top_level - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Create Timing Constraints
I/O Pin Planning (PlanAhead) - Pre-Synthesis
I/O Pin Planning (PlanAhead) - Post-Synthesis
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Translate
Map
Place & Route
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start
Design
Files
Libraries

Figure 3.22: Portion of Project Navigator screen, with Implement Design expanded.

2. Double-click on Implement Design. This will first cause Synthesize – XST to run. Next, Translate, Map and Place & Route will run in turn. As each stage is completed, a green tick will appear next to it. After all three stages are complete, the Project Navigator screen will appear as shown in Figure 3.23.

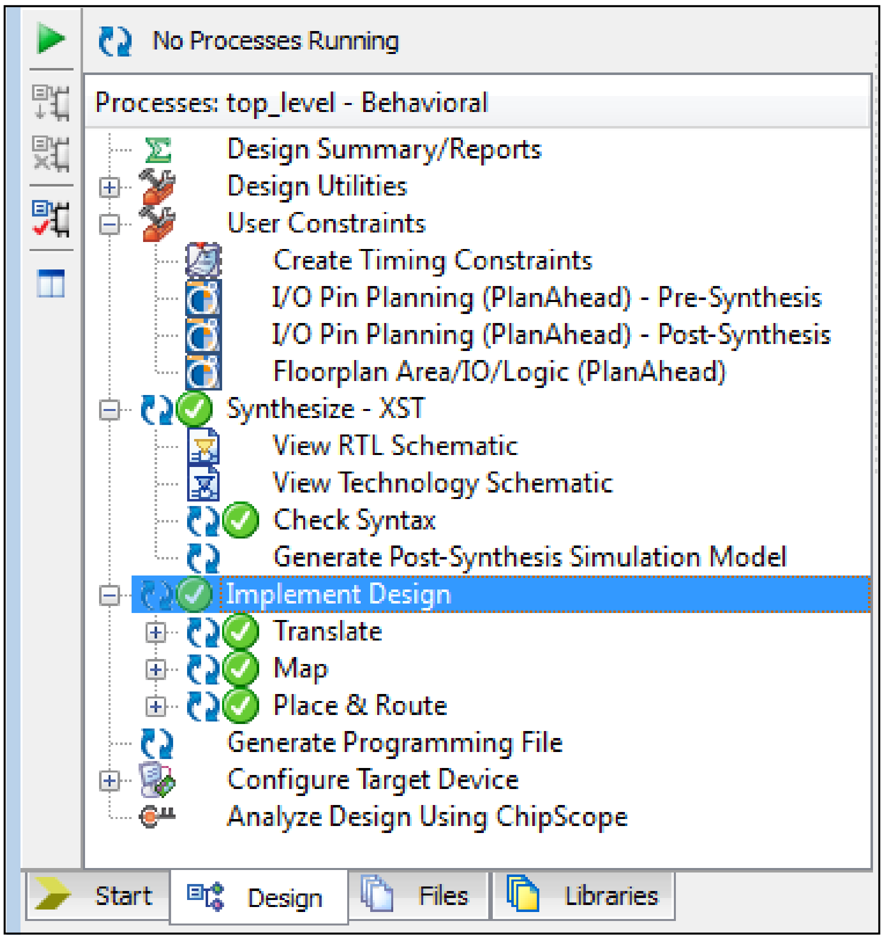
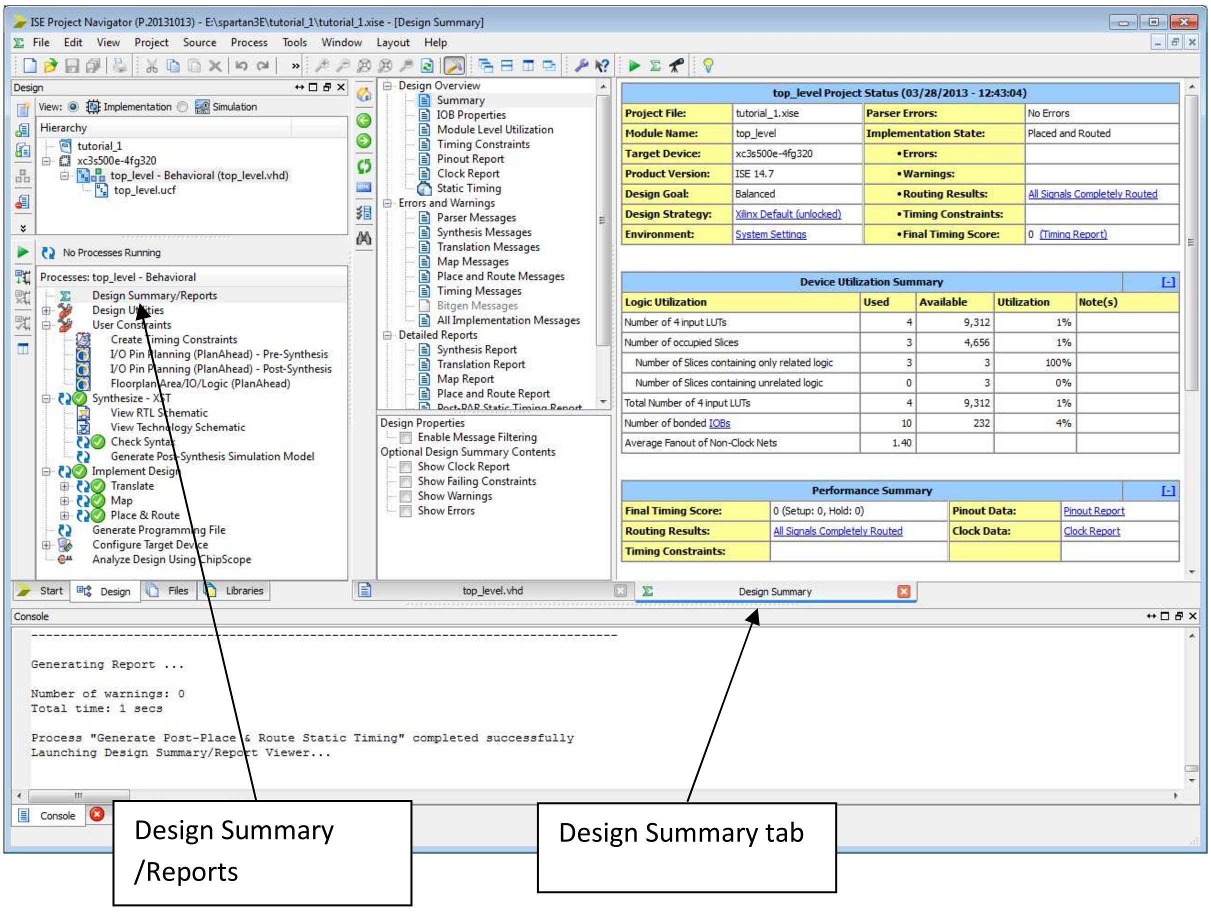

No Processes Running
Processes: top_level - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Create Timing Constraints
I/O Pin Planning (PlanAhead) - Pre-Synthesis
I/O Pin Planning (PlanAhead) - Post-Synthesis
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Translate
Map
Place & Route
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start
Design
Files
Libraries

Figure 3.23: Portion of Project Navigator screen, with Implement Design expanded, after Translate, Map and Place & Route have successfully been run.

3.9 Design Summary

At this stage it may be interesting to view the Design Summary. The **Design Summary** tab can be displayed by double-clicking on **Design Summary/Reports** in the Processes window.



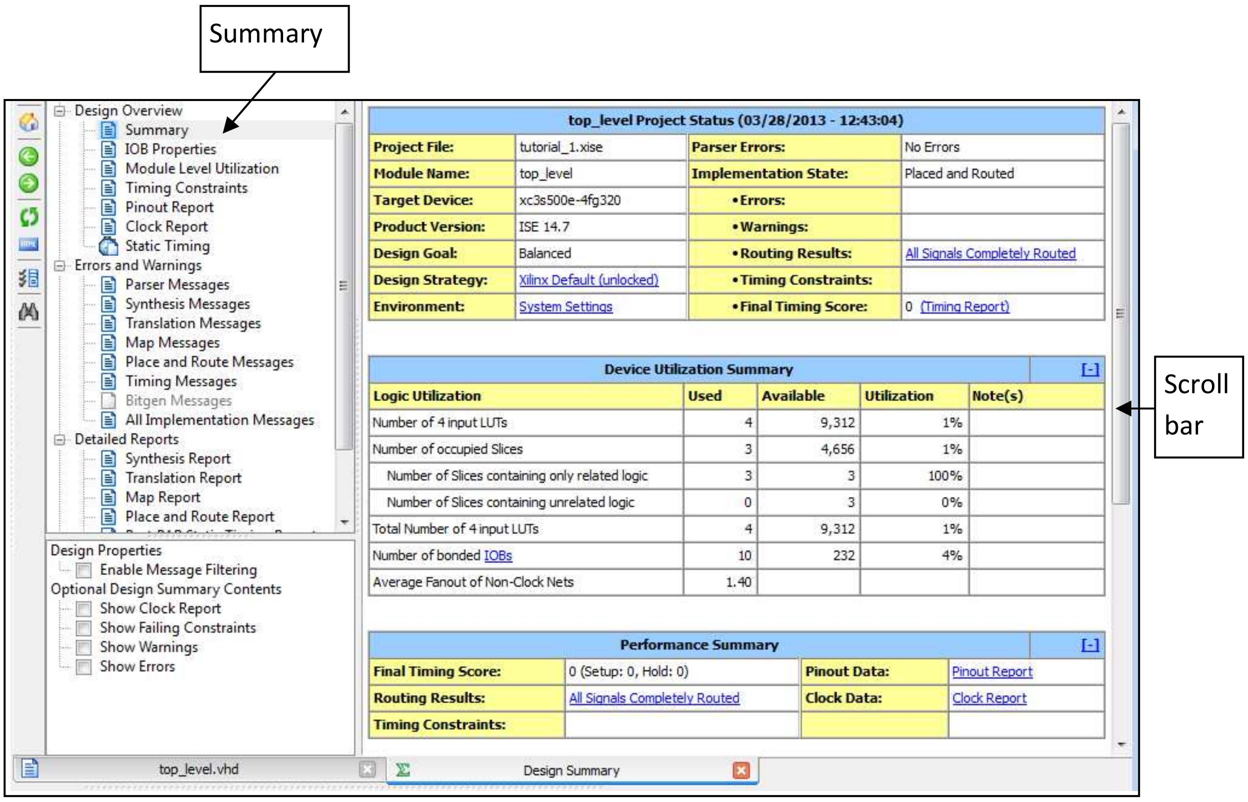
top_level Project Status (03/28/2013 - 12:43:04)			
Project File:	tutorial_1.xise	Parser Errors:	No Errors
Module Name:	top_level	Implementation State:	Placed and Routed
Target Device:	xc3s500e-4fg320	•Errors:	
Product Version:	ISE 14.7	•Warnings:	
Design Goal:	Balanced	•Routing Results:	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	•Timing Constraints:	
Environment:	System Settings	•Final Timing Score:	0 (Timing Report)

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	4	9,312	1%	
Number of occupied Slices	3	4,656	1%	
Number of Slices containing only related logic	3	3	100%	
Number of Slices containing unrelated logic	0	3	0%	
Total Number of 4 input LUTs	4	9,312	1%	
Number of bonded IOBs	10	232	4%	
Average Fanout of Non-Clock Nets	1.40			

Performance Summary			
Final Timing Score:	0 (Setup: 0, Hold: 0)	Pinout Data:	Pinout Report
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report
Timing Constraints:			

```
--------------------------------------------------------------------------------

Generating Report ...

Number of warnings: 0
Total time: 1 secs

Process "Generate Post-Place & Route Static Timing" completed successfully
Launching Design Summary/Report Viewer...
```

Figure 3.24: Viewing the Design Summary tab.



- Design Overview
  - Summary
  - IOB Properties
  - Module Level Utilization
  - Timing Constraints
  - Pinout Report
  - Clock Report
  - Static Timing
- Errors and Warnings
  - Parser Messages
  - Synthesis Messages
  - Translation Messages
  - Map Messages
  - Place and Route Messages
  - Timing Messages
  - Bitgen Messages
  - All Implementation Messages
- Detailed Reports
  - Synthesis Report
  - Translation Report
  - Map Report
  - Place and Route Report

Design Properties
- [ ] Enable Message Filtering

Optional Design Summary Contents
- [ ] Show Clock Report
- [ ] Show Failing Constraints
- [ ] Show Warnings
- [ ] Show Errors

top_level Project Status (03/28/2013 - 12:43:04)

Project File:	tutorial_1.xise	**Parser Errors:**	No Errors
Module Name:	top_level	**Implementation State:**	Placed and Routed
Target Device:	xc3s500e-4fg320	**•Errors:**	
Product Version:	ISE 14.7	**•Warnings:**	
Design Goal:	Balanced	**•Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	
Environment:	System Settings	**•Final Timing Score:**	0 (Timing Report)

Device Utilization Summary [-]

Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	4	9,312	1%	
Number of occupied Slices	3	4,656	1%	
Number of Slices containing only related logic	3	3	100%	
Number of Slices containing unrelated logic	0	3	0%	
Total Number of 4 input LUTs	4	9,312	1%	
Number of bonded IOBs	10	232	4%	
Average Fanout of Non-Clock Nets	1.40			

Performance Summary [-]

Final Timing Score:	0 (Setup: 0, Hold: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:			

Figure 3.25: Design Summary tab.

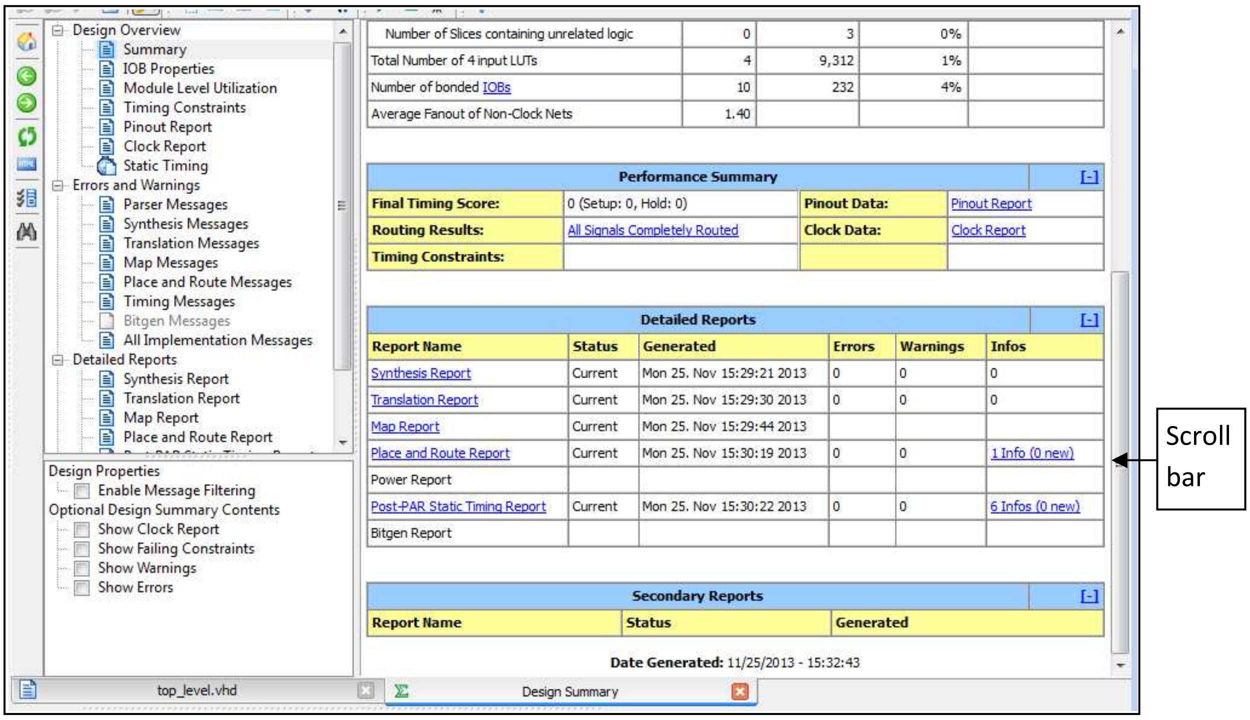
Initially, the **Summary** report is displayed as shown in Figure 3.25. The **Project Status** table at the top right of Figure 3.25 displays some general information about the project, including the **Implementation State** which is **Placed and Routed** at this stage.

The next table down is the **Device Utilization Summary**, which shows how much of the resources on the FPGA are used by the project. The number of 4-input lookup tables, occupied slices and Input/Output Blocks used by the project are listed in Table 3.3. It can be seen that this project uses only a small amount of the Spartan3E’s resources.

	Used	Available
4-input Look Up Tables (LUTs)	4	9132
Occupied Slices	3	4656
Input/Output Blocks (IOBs)	10	232

Table 3.3: Design Summary tab.

Pulling down the scroll bar in Figure 3.25 will reveal the **Performance Summary, Detailed Reports** and **Secondary Reports** tables, as shown in Figure 3.26. Clicking on links to the detailed reports such as **Synthesis Report**, **Translation Report**, **Map Report** and **Place and Route Report** will result in that report being displayed. To return to the **Summary** report, double-click on **Summary**, which appears under **Design Overview**, as shown in Figure 3.25.



- Design Overview
  - Summary
  - IOB Properties
  - Module Level Utilization
  - Timing Constraints
  - Pinout Report
  - Clock Report
  - Static Timing
- Errors and Warnings
  - Parser Messages
  - Synthesis Messages
  - Translation Messages
  - Map Messages
  - Place and Route Messages
  - Timing Messages
  - Bitgen Messages
  - All Implementation Messages
- Detailed Reports
  - Synthesis Report
  - Translation Report
  - Map Report
  - Place and Route Report

Design Properties
- Enable Message Filtering

Optional Design Summary Contents
- Show Clock Report
- Show Failing Constraints
- Show Warnings
- Show Errors

Number of Slices containing unrelated logic	0	3	0%	
Total Number of 4 input LUTs	4	9,312	1%	
Number of bonded IOBs	10	232	4%	
Average Fanout of Non-Clock Nets	1.40			

Performance Summary [-]

Final Timing Score:	0 (Setup: 0, Hold: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:			

Detailed Reports [-]

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Mon 25. Nov 15:29:21 2013	0	0	0
Translation Report	Current	Mon 25. Nov 15:29:30 2013	0	0	0
Map Report	Current	Mon 25. Nov 15:29:44 2013			
Place and Route Report	Current	Mon 25. Nov 15:30:19 2013	0	0	1 Info (0 new)
Power Report					
Post-PAR Static Timing Report	Current	Mon 25. Nov 15:30:22 2013	0	0	6 Infos (0 new)
Bitgen Report					

Secondary Reports [-]

Report Name	Status	Generated

Date Generated: 11/25/2013 - 15:32:43

Figure 3.26: Design Summary tab with scroll bar pulled down.

3.10 Download Design to Board

The next steps involve generating the program file, and downloading it to the Spartan-3E board using iMPACT.

1. As shown in Figure 3.27, click on the '+' next to **Configure Target Device**. This will expand out to show the **Manage Configuration Project (iMPACT)** option.

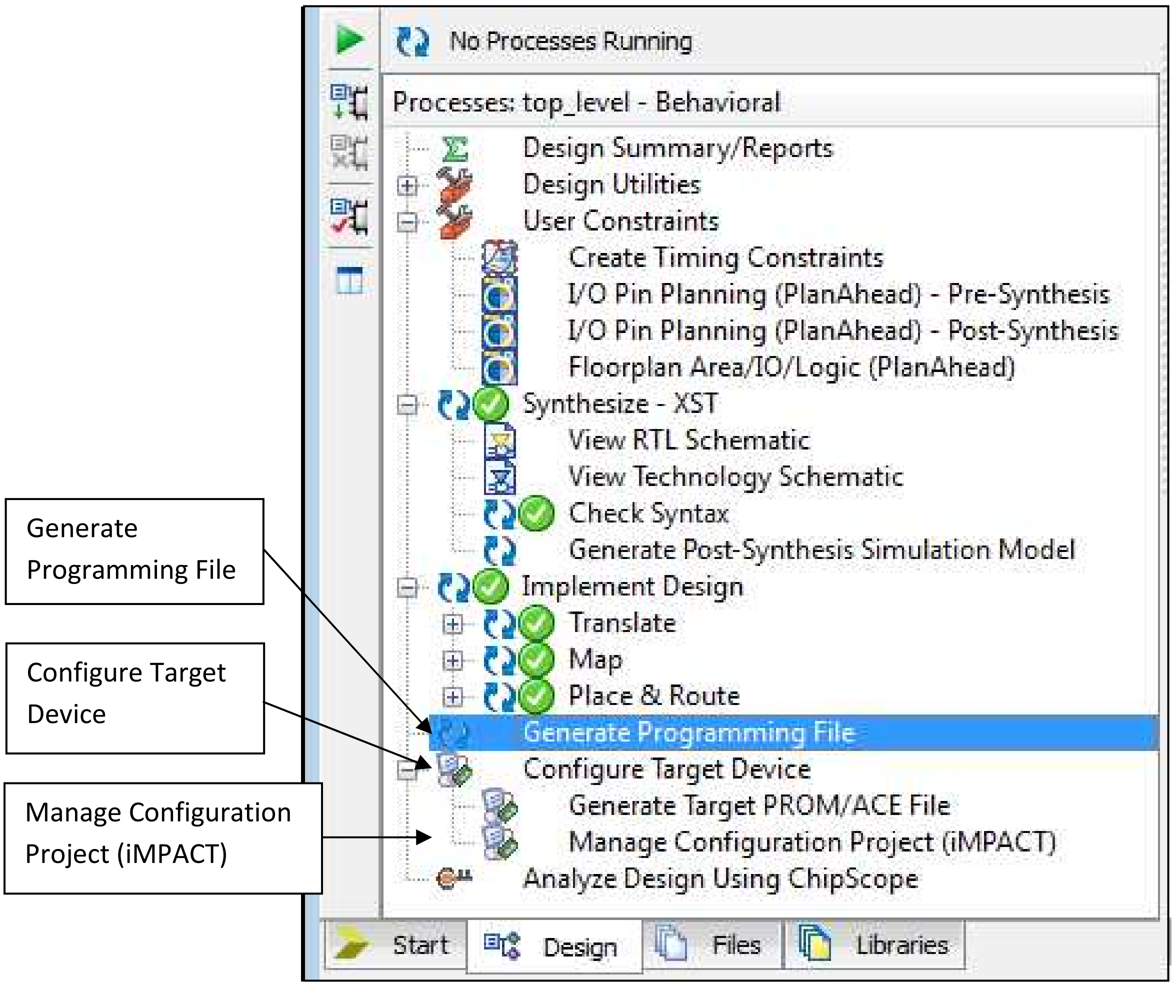
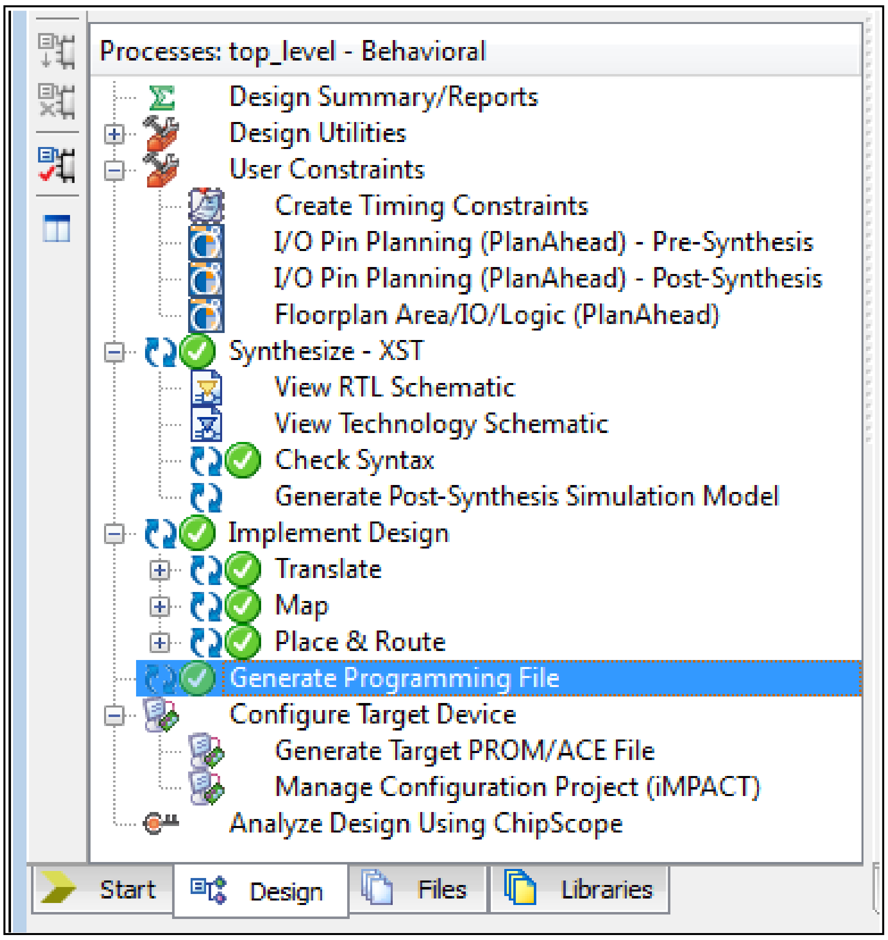

No Processes Running
Processes: top_level - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Create Timing Constraints
I/O Pin Planning (PlanAhead) - Pre-Synthesis
I/O Pin Planning (PlanAhead) - Post-Synthesis
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Translate
Map
Place & Route
Generate Programming File
Configure Target Device
Generate Target PROM/ACE File
Manage Configuration Project (iMPACT)
Analyze Design Using ChipScope
Start
Design
Files
Libraries
Generate Programming File
Configure Target Device
Manage Configuration Project (iMPACT)

Figure 3.27: Portion of Project Navigator screen, with Implement Design expanded.

2. Double-click on **Generate Programming File**. When this has successfully run, a green tick will appear next to **Generate Programming File** as shown in Figure 3.28.

3. Connect the power cable and the USB cable to the Spartan-3E board (refer to Figure 2.2 for locations of the power and USB plugs). Plug the USB cable from the Spartan-3E into the PC, and make sure the Sparan-3E board is switched on.



Processes: top_level - Behavioral

- Design Summary/Reports
- Design Utilities
- User Constraints
  - Create Timing Constraints
  - I/O Pin Planning (PlanAhead) - Pre-Synthesis
  - I/O Pin Planning (PlanAhead) - Post-Synthesis
  - Floorplan Area/IO/Logic (PlanAhead)
- ✓ Synthesize - XST
  - View RTL Schematic
  - View Technology Schematic
  - ✓ Check Syntax
  - Generate Post-Synthesis Simulation Model
- ✓ Implement Design
  - ✓ Translate
  - ✓ Map
  - ✓ Place & Route
- ✓ Generate Programming File
- Configure Target Device
  - Generate Target PROM/ACE File
  - Manage Configuration Project (iMPACT)
- Analyze Design Using ChipScope

Figure 3.28: Portion of Project Navigator screen, after Generate Programming File has successfully been run.

4. Double-click on **Manage Configuration Project (iMPACT).** The iMPACT window should appear as shown in Figure 3.29.

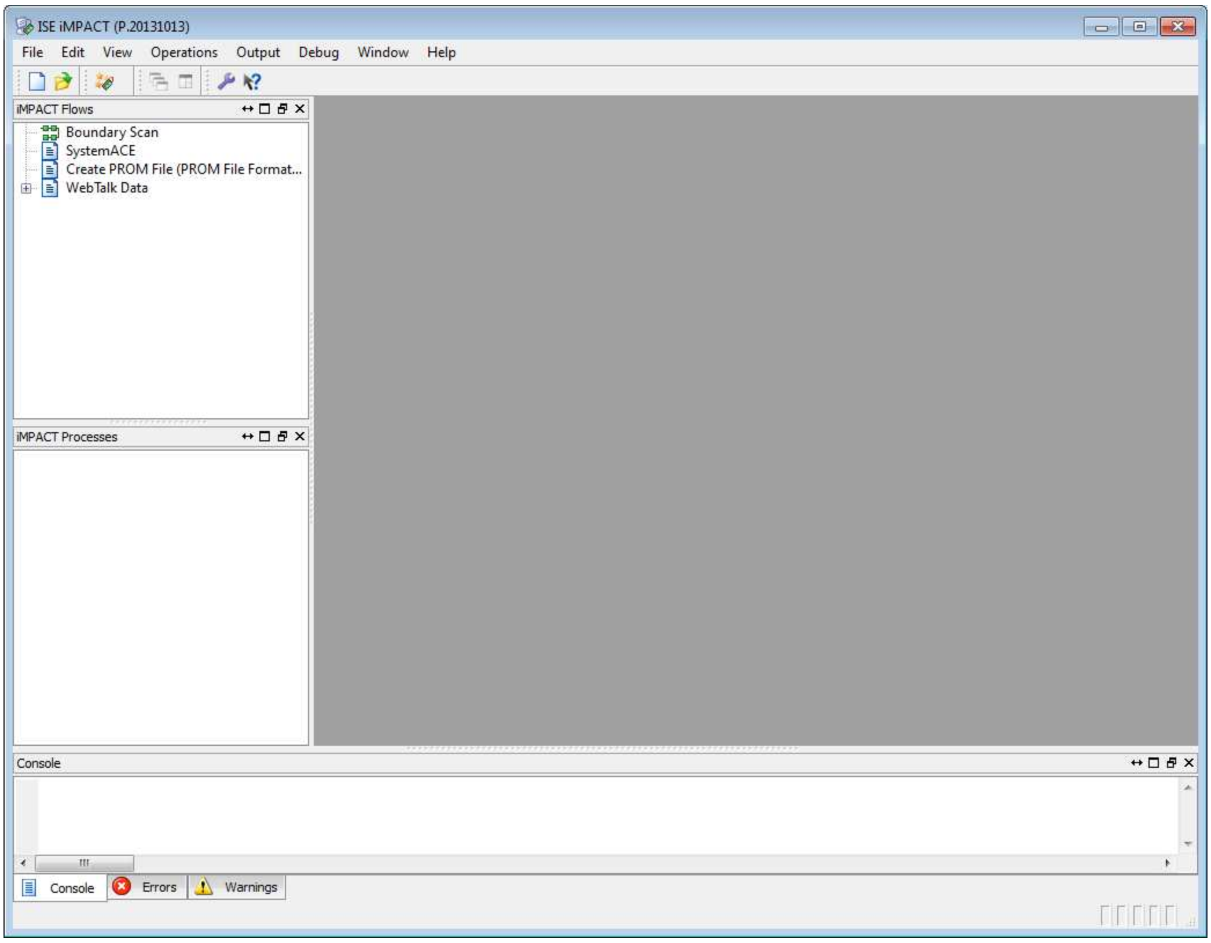
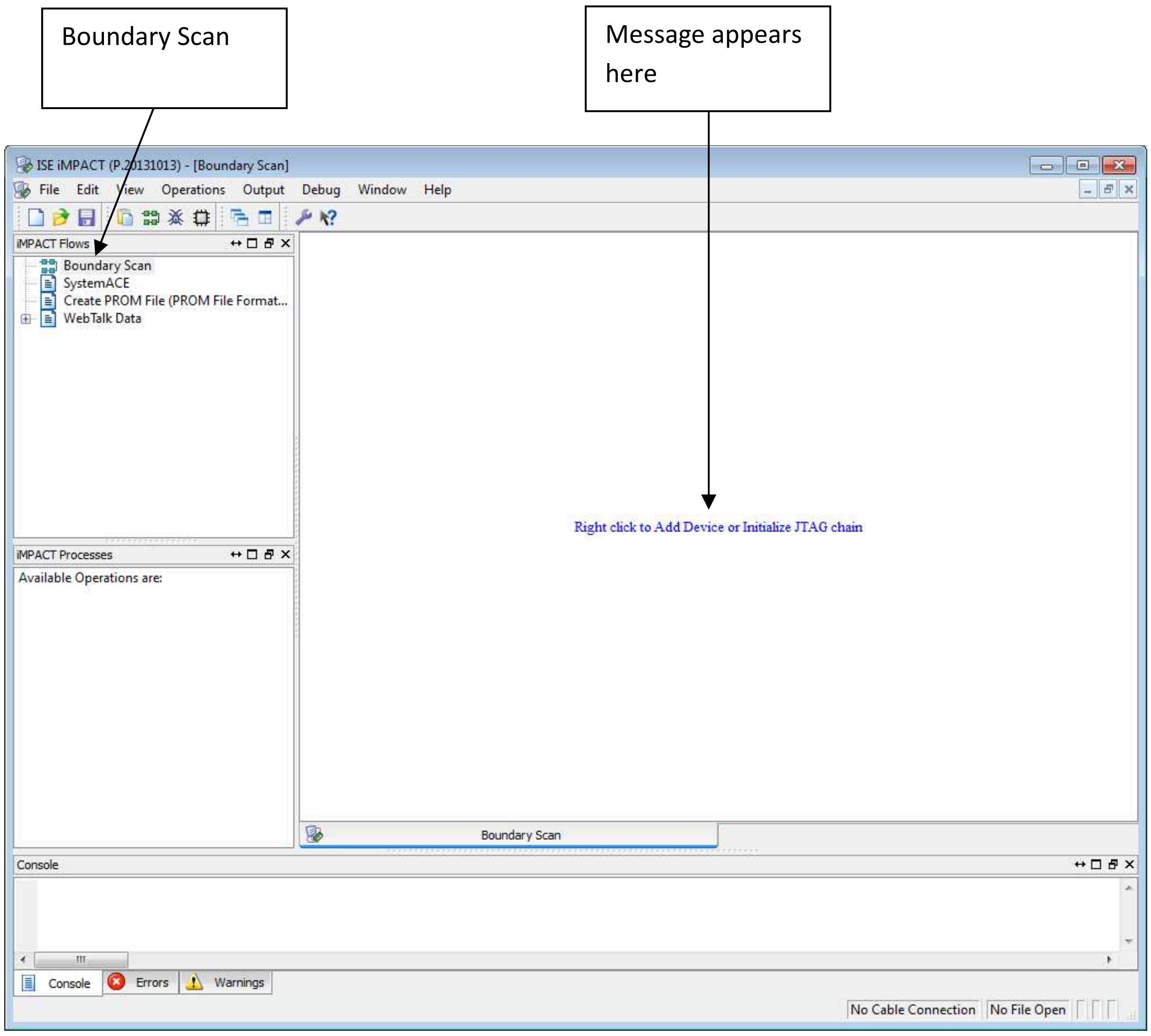


Figure 3.29: The initial iMPACT window.

5. Double-click on Boundary Scan as shown in Figure 3.30. The message "Right click to Add Device or Initialize JTAG Chain" should appear to the right.



Right click to Add Device or Initialize JTAG chain

Figure 3.30: iMPACT window, after double-clicking on Boundary Scan.

6. Right click on the text “Right click to Add Device or Initialize JTAG Chain”, and select **Initialise Chain**, as shown in Figure 3.31.

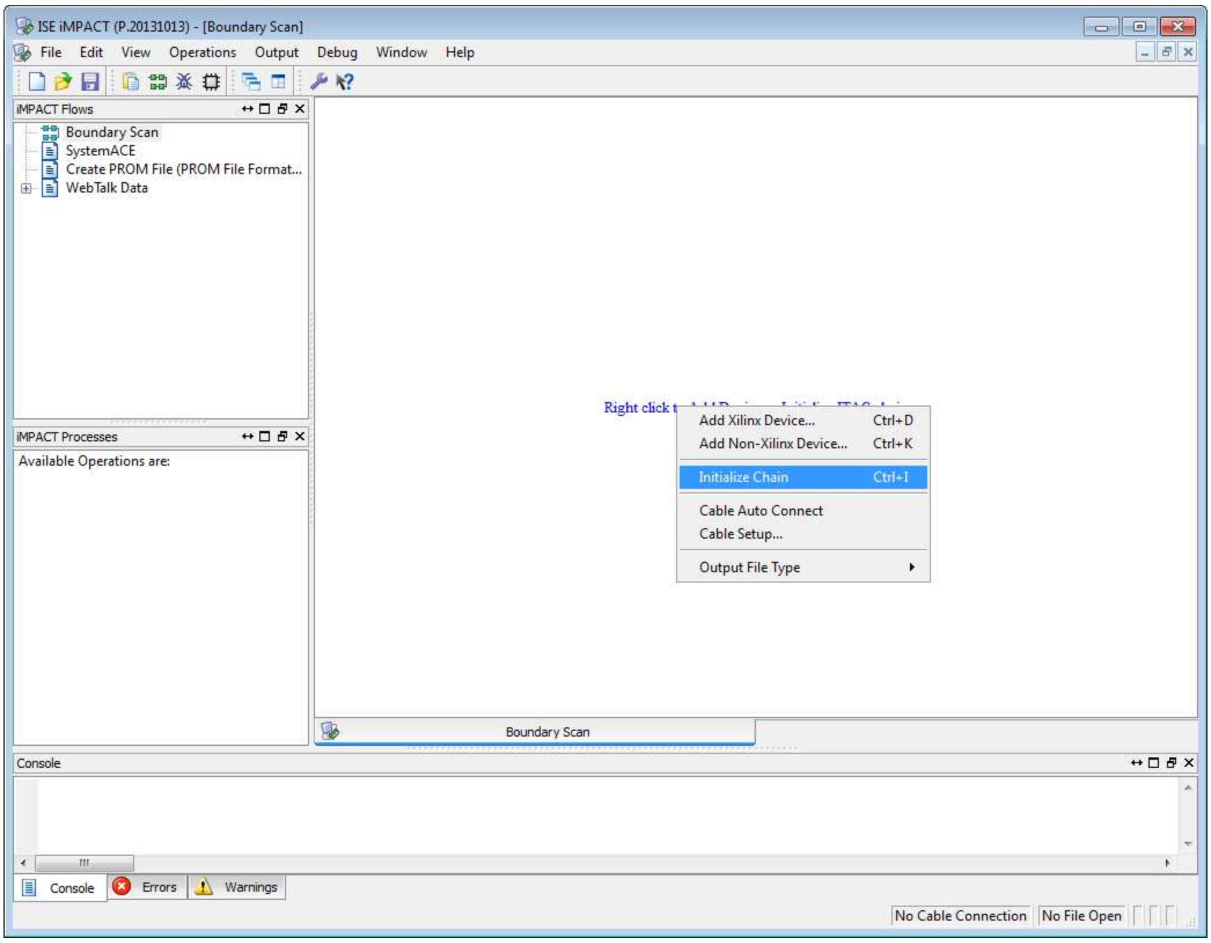
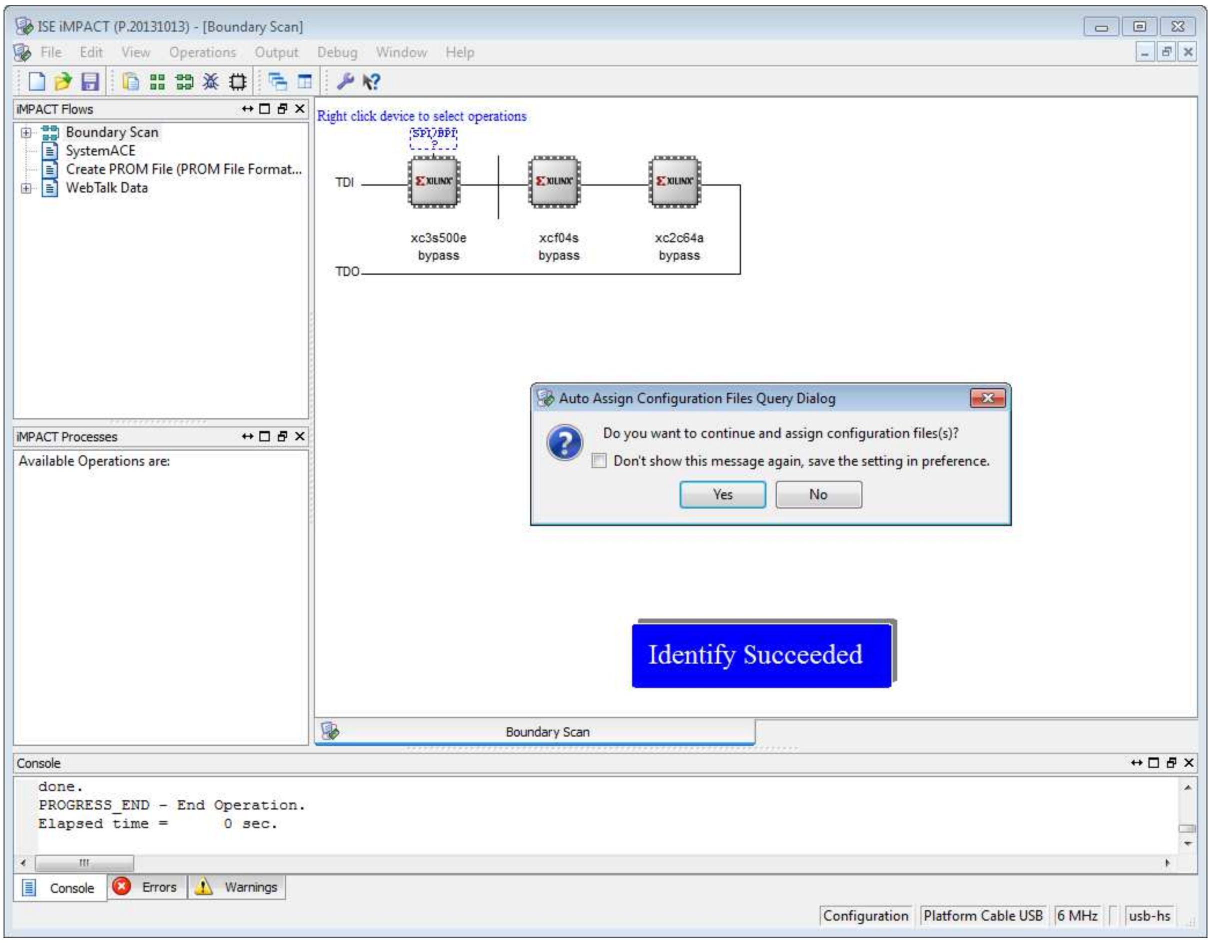


Figure 3.31: iMPACT window, showing Initialize Chain selected.

7. After a while, a picture of a “chain” should appear, along with the message **Identify Succeeded** in a blue box (Figure 3.32). The first chip, the **xc3s500e**, is the FPGA chip that we wish to program. The other two, **xcf04s** and **xc2c64a**, are other chips on the board that will be bypassed.

   A dialog box, asking “Do you wish to continue and select configuration file(s)?” will appear, as shown in Figure 3.32. Click **Yes**.



Do you want to continue and assign configuration files(s)?

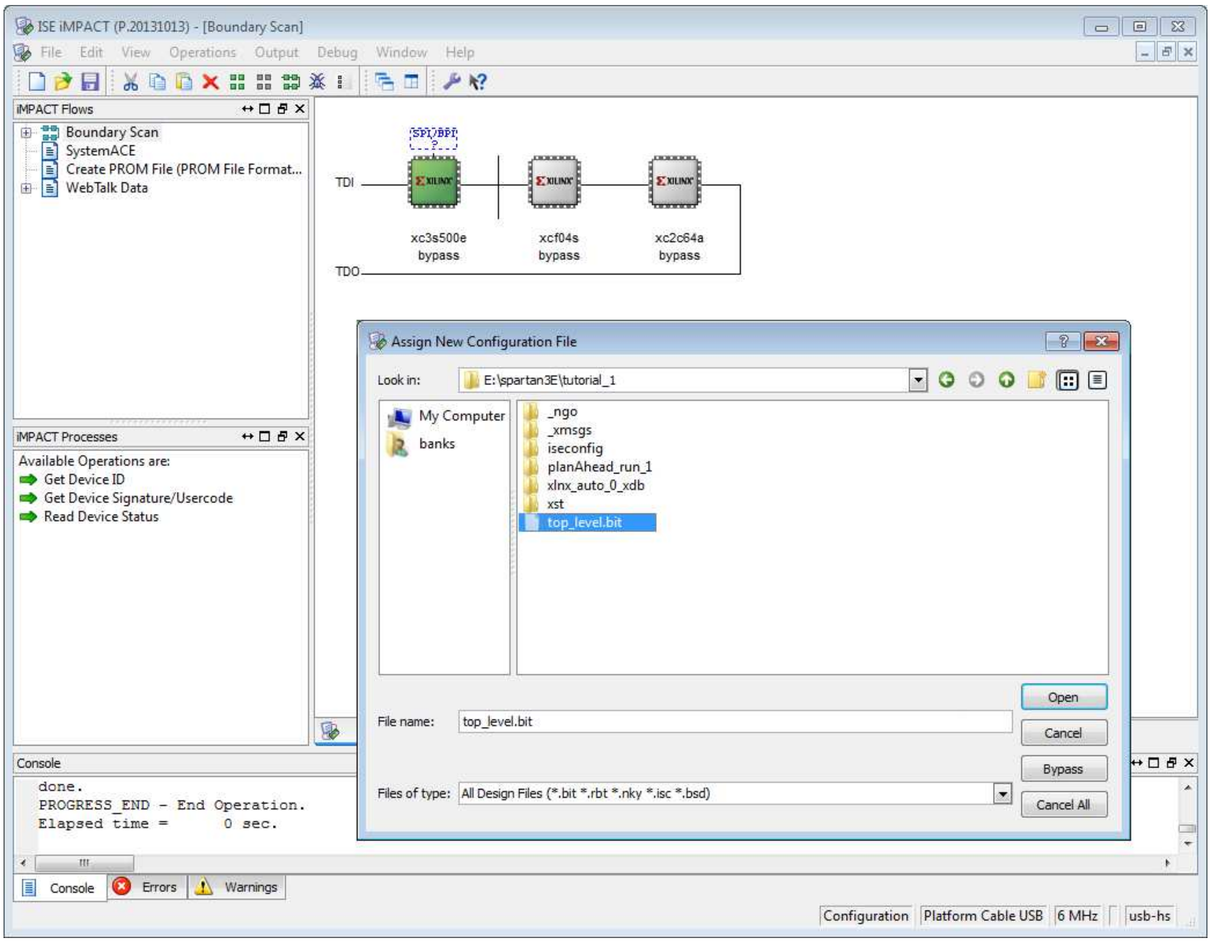
Don't show this message again, save the setting in preference.

Available Operations are:

```
done.
PROGRESS_END - End Operation.
Elapsed time =      0 sec.
```

Figure 3.32: iMPACT window, assign configuration files.

8. The **Assign New Configuration File** window will appear (Figure 3.33). Select the file "top_level.bit", and click **Open**. This associates the file top_level.bit with the xc3s500e.



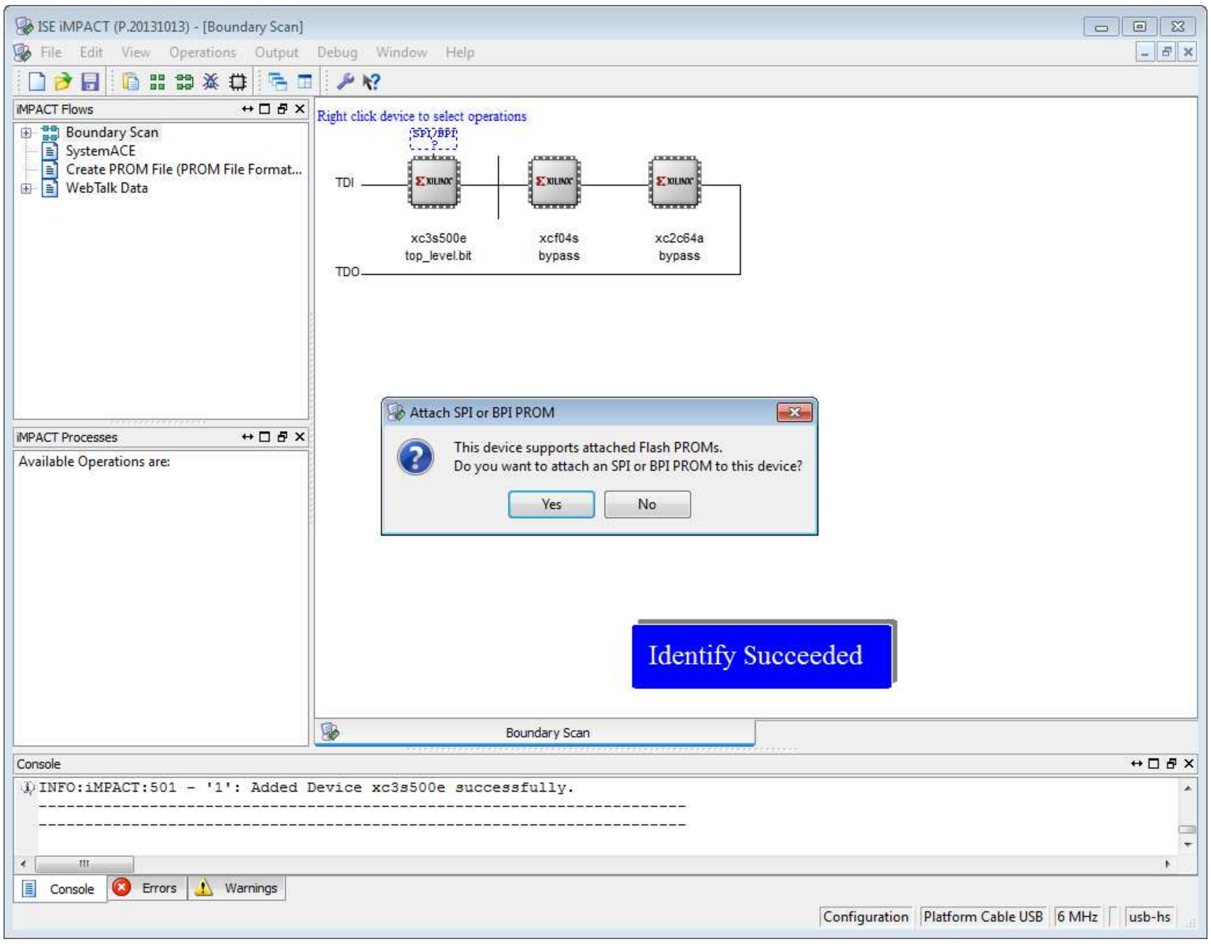
File name: top_level.bit

Files of type: All Design Files (*.bit *.rbt *.nky *.isc *.bsd)

```
done.
PROGRESS_END - End Operation.
Elapsed time =       0 sec.
```

Figure 3.33: iMPACT window, assigning the configuration file for the xc3e500e.

9. A message stating “This device supports attached flash PROMs. Do you want to attach an SPI or BPI PROM to this device?” will appear (Figure 3.34). This is not needed for this design. Click **No**.

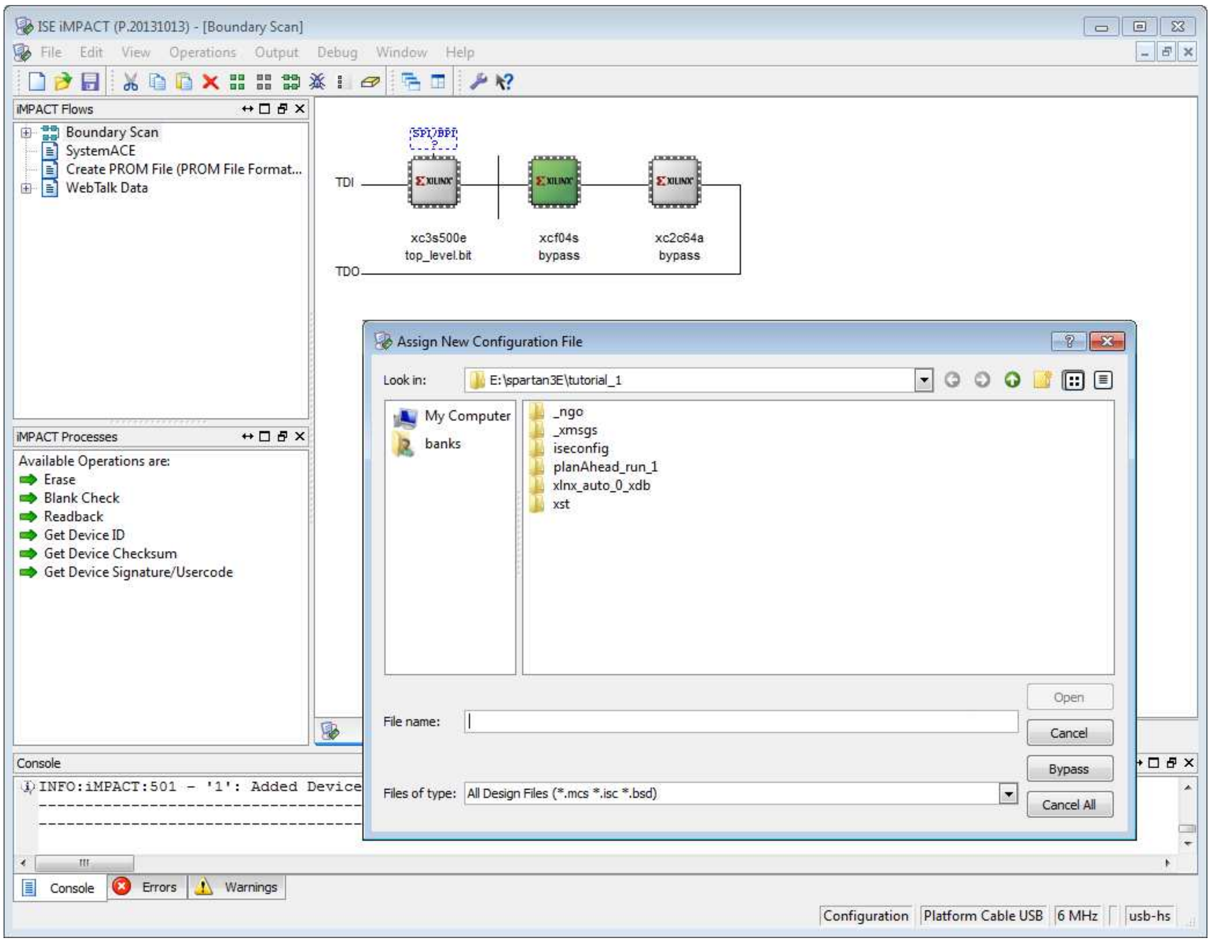


This device supports attached Flash PROMs.
Do you want to attach an SPI or BPI PROM to this device?

```
INFO:iMPACT:501 - '1': Added Device xc3s500e successfully.
----------------------------------------------------------------------
----------------------------------------------------------------------
```

Figure 3.34: iMPACT window, dialog box asking if we wish to attach an SPI or BPI PROM.

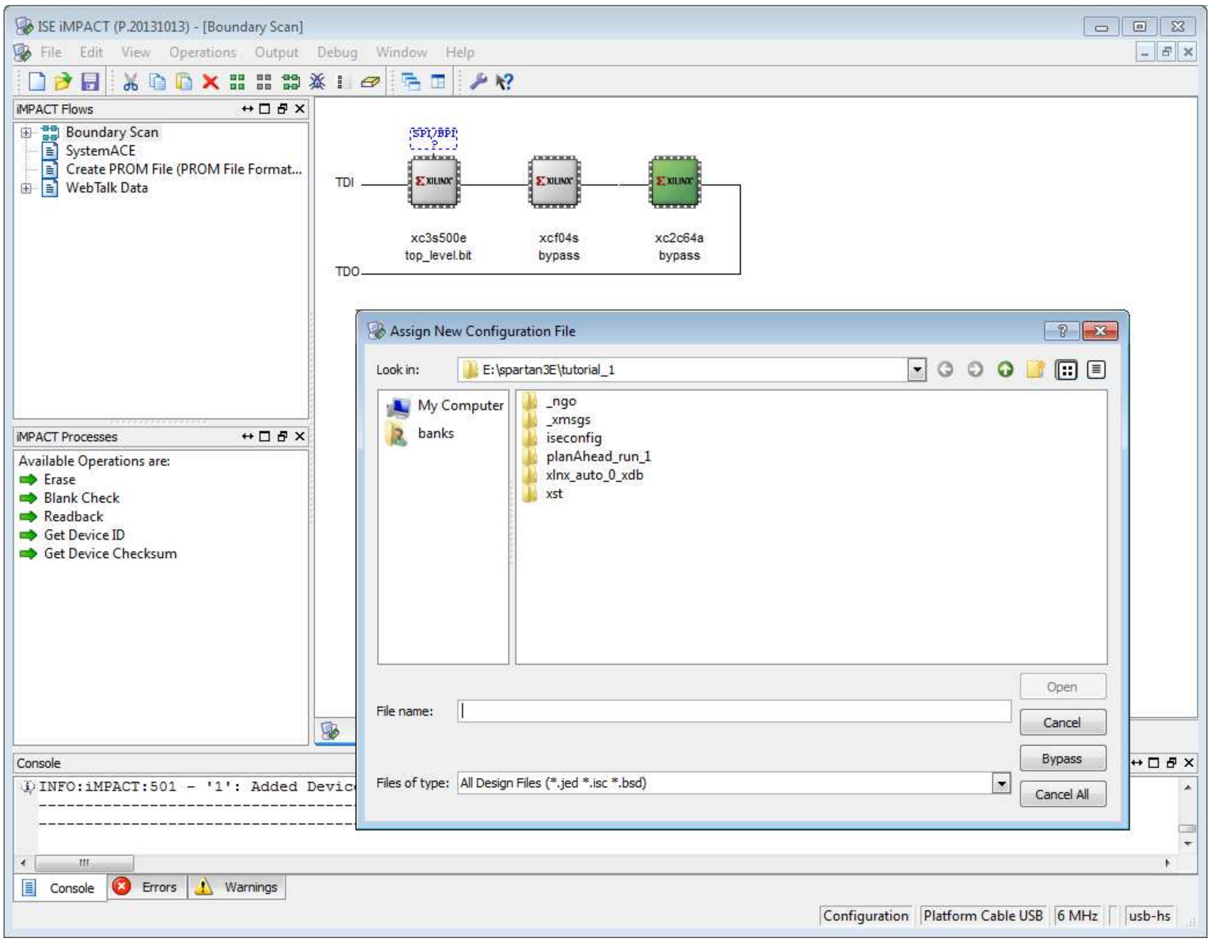
10. The **Assign New Configuration File** window will appear again (Figure 3.35). In this case click **Bypass**. This ensures that the xcf04s is bypassed.



```
INFO:iMPACT:501 - '1': Added Device
----------------------------------
----------------------------------
```

Figure 3.35: iMPACT window, bypassing the xcf04s.

11. The **Assign New Configuration File** window will appear yet again (Figure 3.36). Again click **Bypass**. This ensures that the xc2c64a is bypassed.



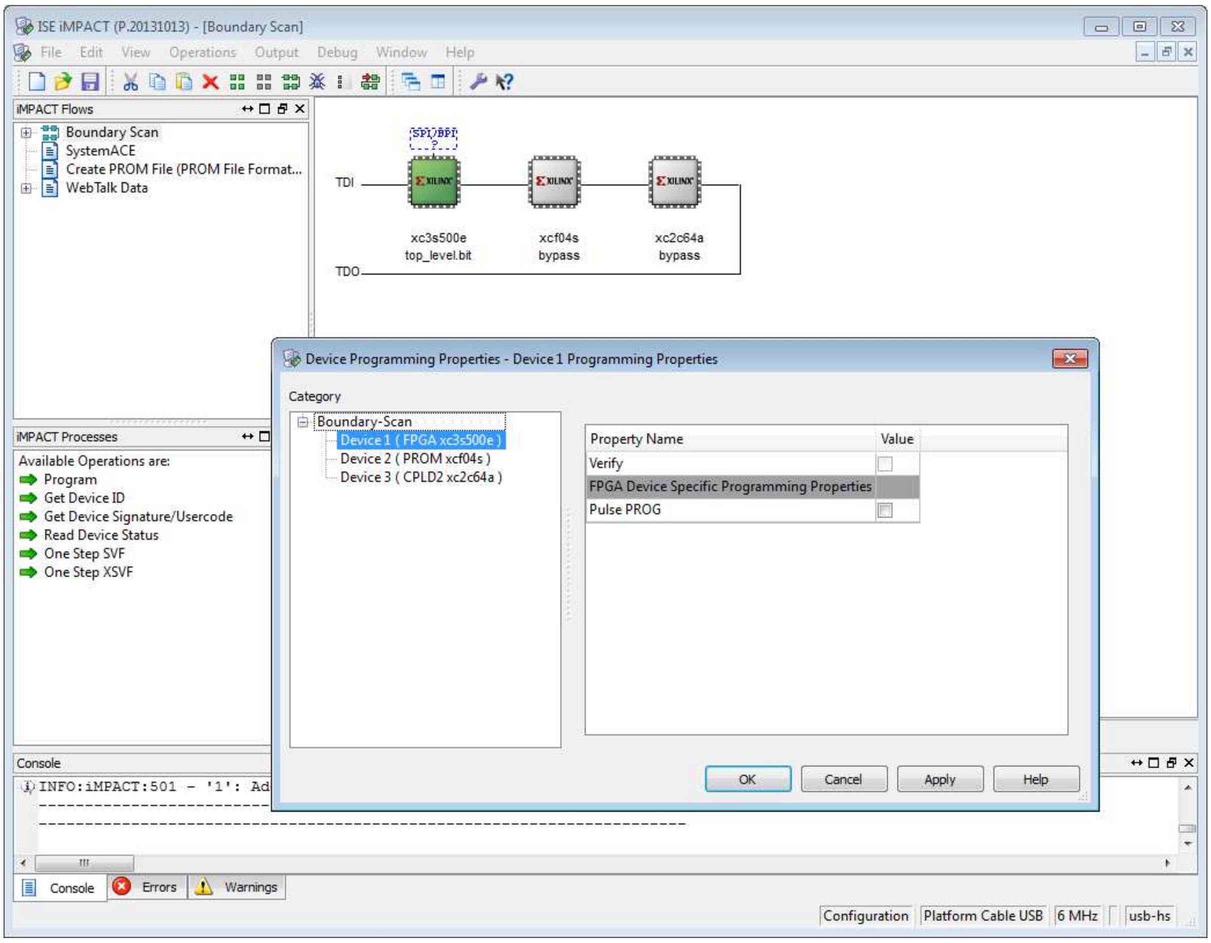
Look in: E:\spartan3E\tutorial_1

Files of type: All Design Files (*.jed *.isc *.bsd)

```
INFO:iMPACT:501 - '1': Added Devic
----------------------------------
----------------------------------
```

Figure 3.36: iMPACT window, bypassing the xc2c64a.

12. A window entitled **Device Programming Properties – Device 1 Programming Properties** will appear (Figure 3.37). Click **OK**.



Property Name	Value
Verify	☐
FPGA Device Specific Programming Properties	
Pulse PROG	☐

```
INFO:iMPACT:501 - '1': Ad
--------------------------
---------------------------------------------------------------------------
```

Figure 3.37: iMPACT window, Device Programming Properties dialog box.

13. The iMPACT window should now appear as shown in Figure 3.38. Right click on the xc3e500e chip (Figure 3.39) and select **Program**.

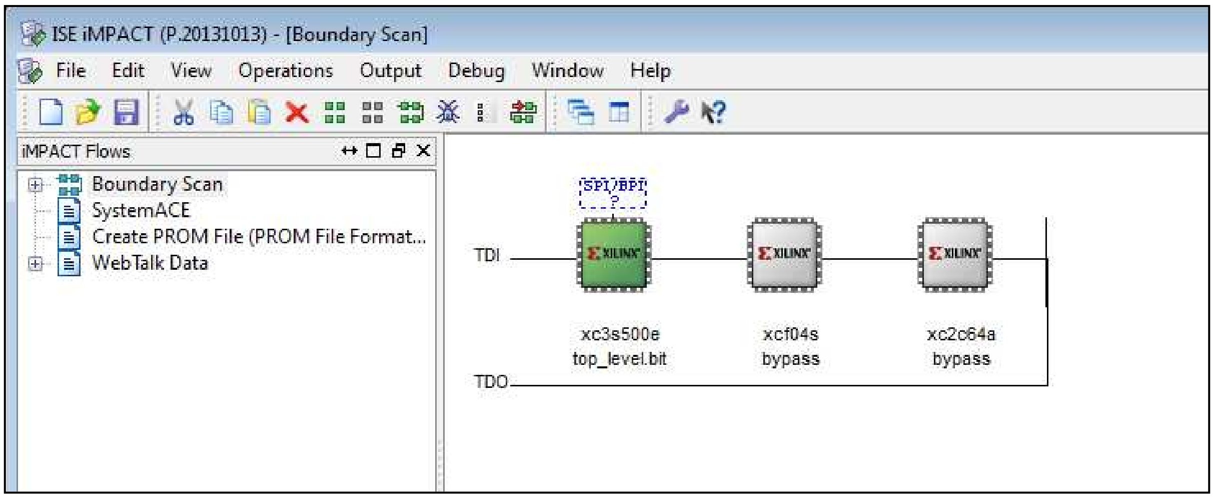

ISE iMPACT (P.20131013) - [Boundary Scan]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File Format...
WebTalk Data
SPI/BPI ?
TDI
TDO
xc3s500e top_level.bit
xcf04s bypass
xc2c64a bypass

Figure 3.38: iMPACT window, showing the device chain.

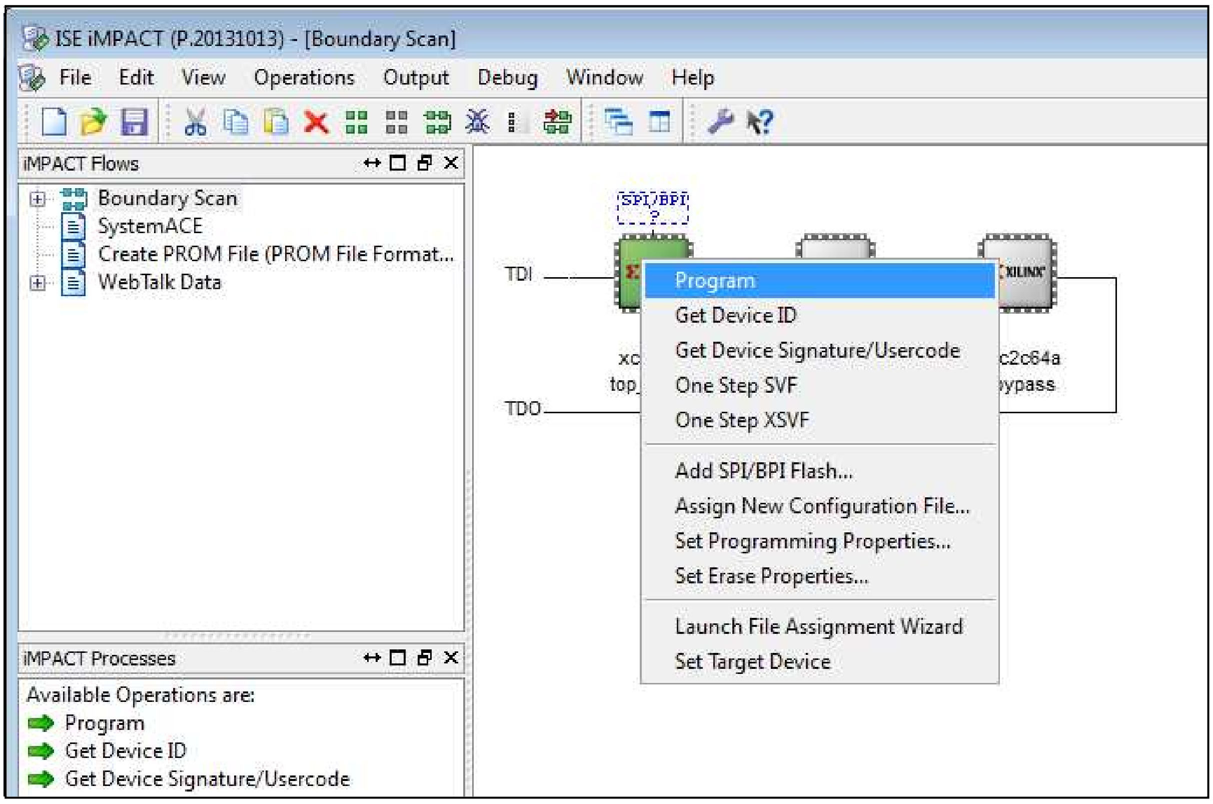
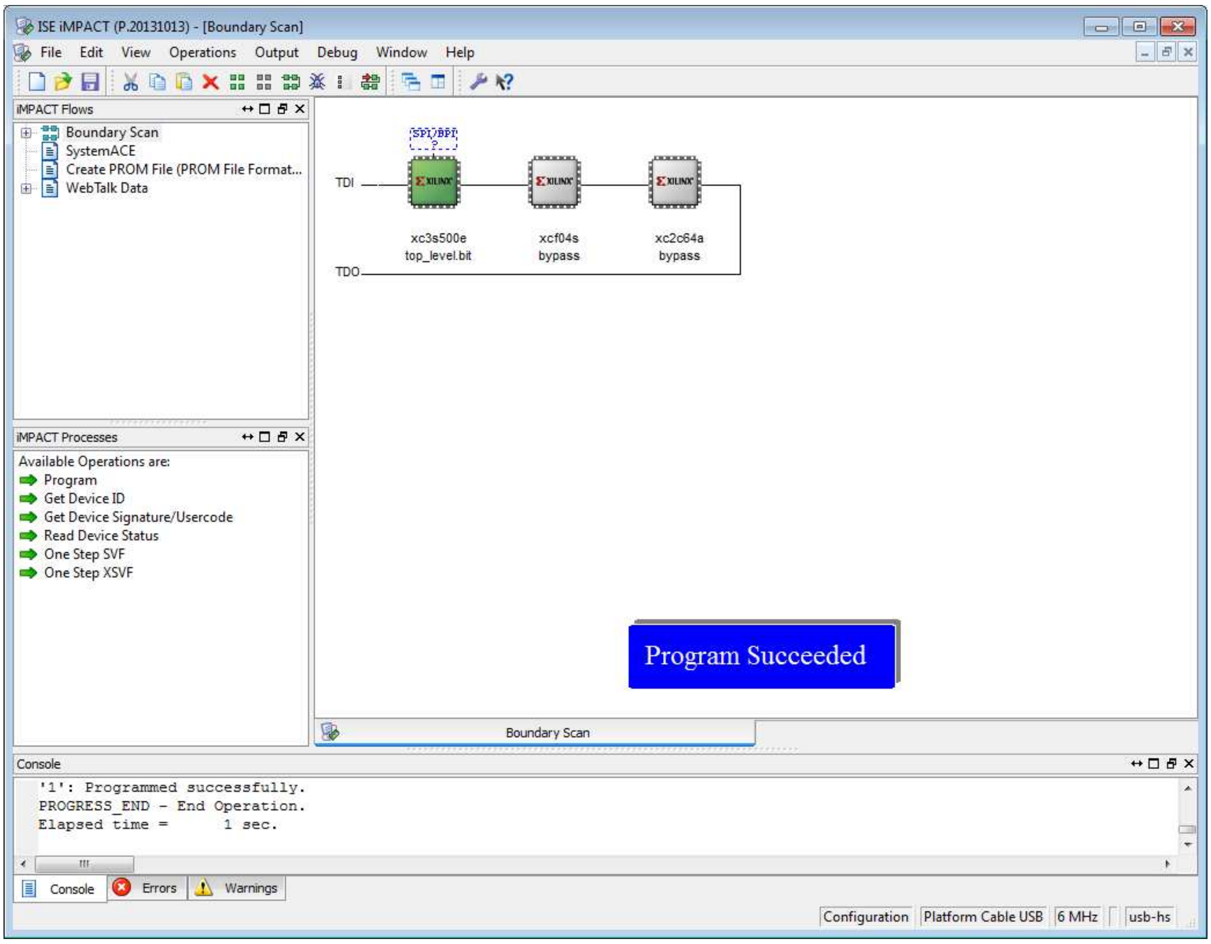

ISE iMPACT (P.20131013) - [Boundary Scan]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File Format...
WebTalk Data
iMPACT Processes
Available Operations are:
Program
Get Device ID
Get Device Signature/Usercode
SPI/BPI ?
TDI
TDO
Program
Get Device ID
Get Device Signature/Usercode
One Step SVF
One Step XSVF
Add SPI/BPI Flash...
Assign New Configuration File...
Set Programming Properties...
Set Erase Properties...
Launch File Assignment Wizard
Set Target Device

Figure 3.39: iMPACT window, options which appear when right clicking on the xc3s500e.

14. The program should now be downloaded to the Spartan-3E board. After the download is complete, the message "Program Succeeded" will appear in a blue box (Figure 3.40).



```
'1': Programmed successfully.
PROGRESS_END - End Operation.
Elapsed time =      1 sec.
```

Figure 3.40: iMPACT window, after the program has been successfully downloaded to the Spartan-3E board.

3.11 Running the Program on the Spartan-3E Board

The Spartan-3E board after downloading the program is shown in Figure 3.41. Note that the appearance of the LCD screen may differ from what is shown. The LCD screen will normally continue to display whatever was being displayed at the instant the new program was downloaded. In this case, the program that was running was the startup program that is shipped with the board, which displays a scrolling message on the LCD screen. The LCD screen is not used in this tutorial.

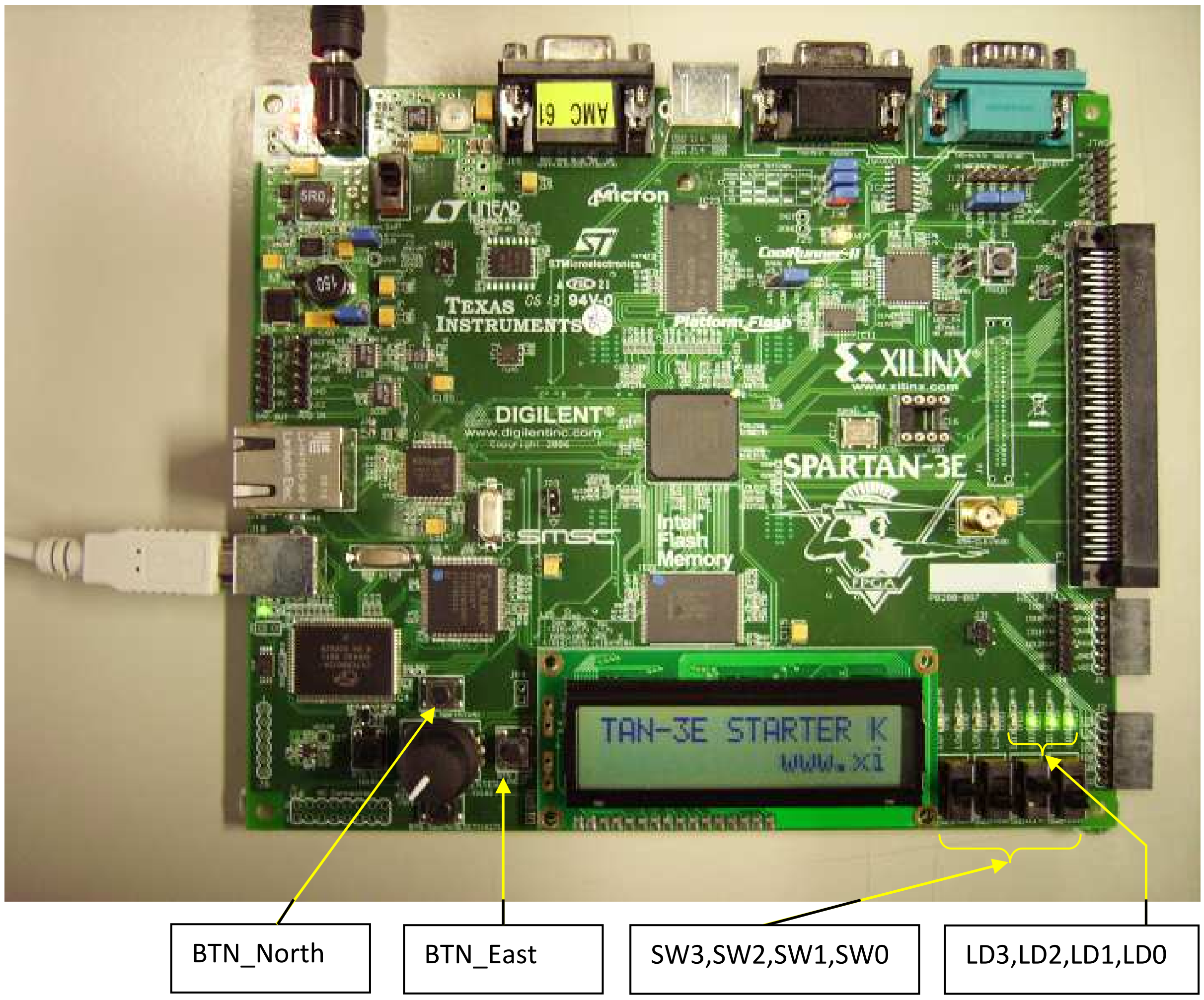


Figure 3.41: The Spartan-3E board with the program running.

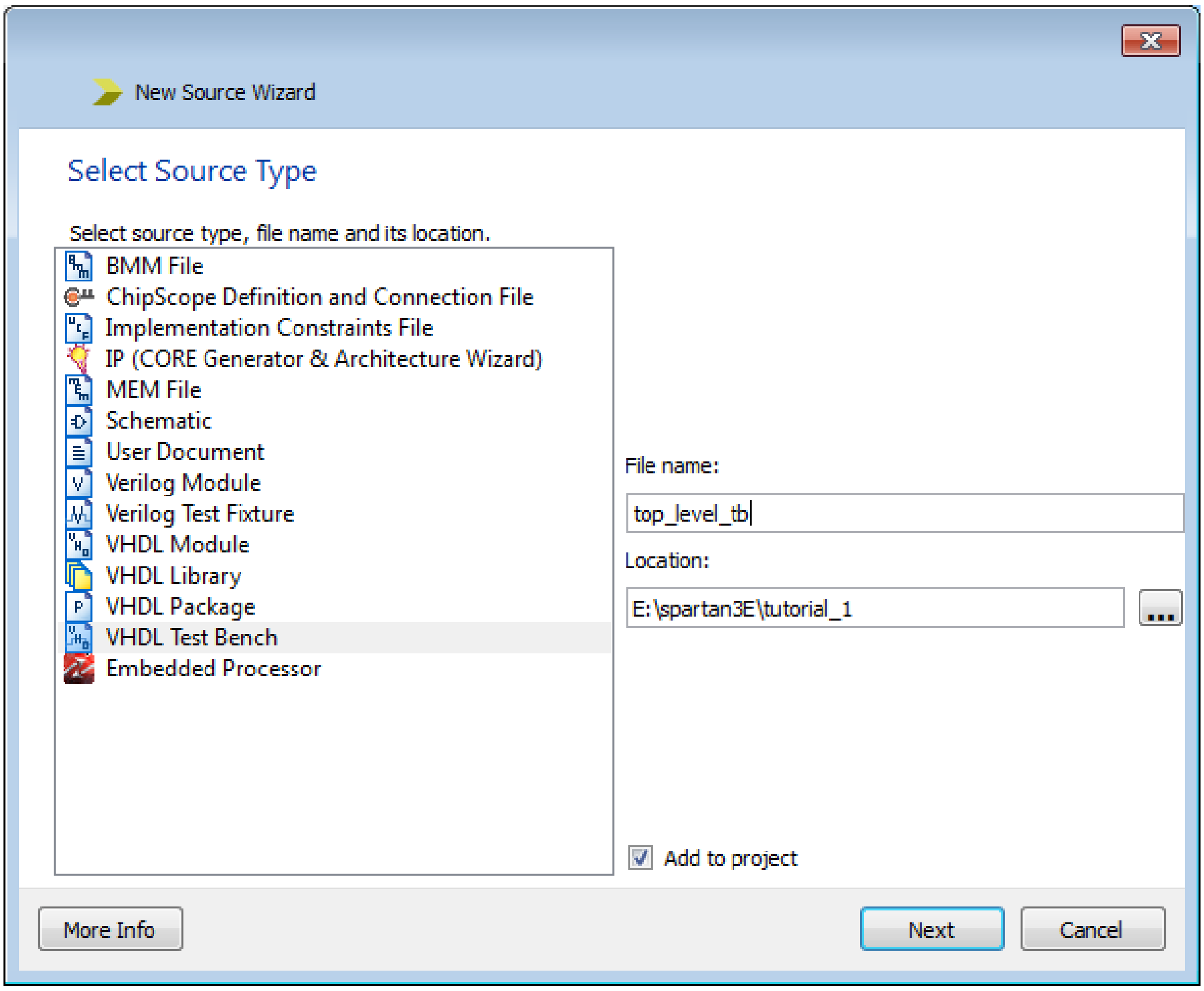
Now the program can be tested on the Spartan-3E. For the slider switches, SW0, SW1, SW2 and SW3, the switch in the UP position indicates that the switch is on. LD0 should switch on when either SW0 or SW1 are on. LD1 should switch on when either SW2 or SW3 are on. LD2 should switch on when ((SW0 or SW1) and (SW2 or SW3)) is true. Finally LD3 should switch on when either BTN_East or BTN_North are pressed.

4.0 Using a TestBench and Simulating the Design

A testbench is a VHDL module which controls the stimulus inputs to **top_level.vhd**. The outputs resulting from these inputs can then be viewed in the **ISim** simulator. It can then be verified that the **top_level** module behaves as expected.

4.1 Adding a TestBench

1. Select **Project→New Source** as shown previously in Figure 3.6. The New Source Wizard will appear. Select source type as **VHDL Test Bench**, and enter the file name **as top_level_tb**, as shown in Figure 4.1. Click **Next**.



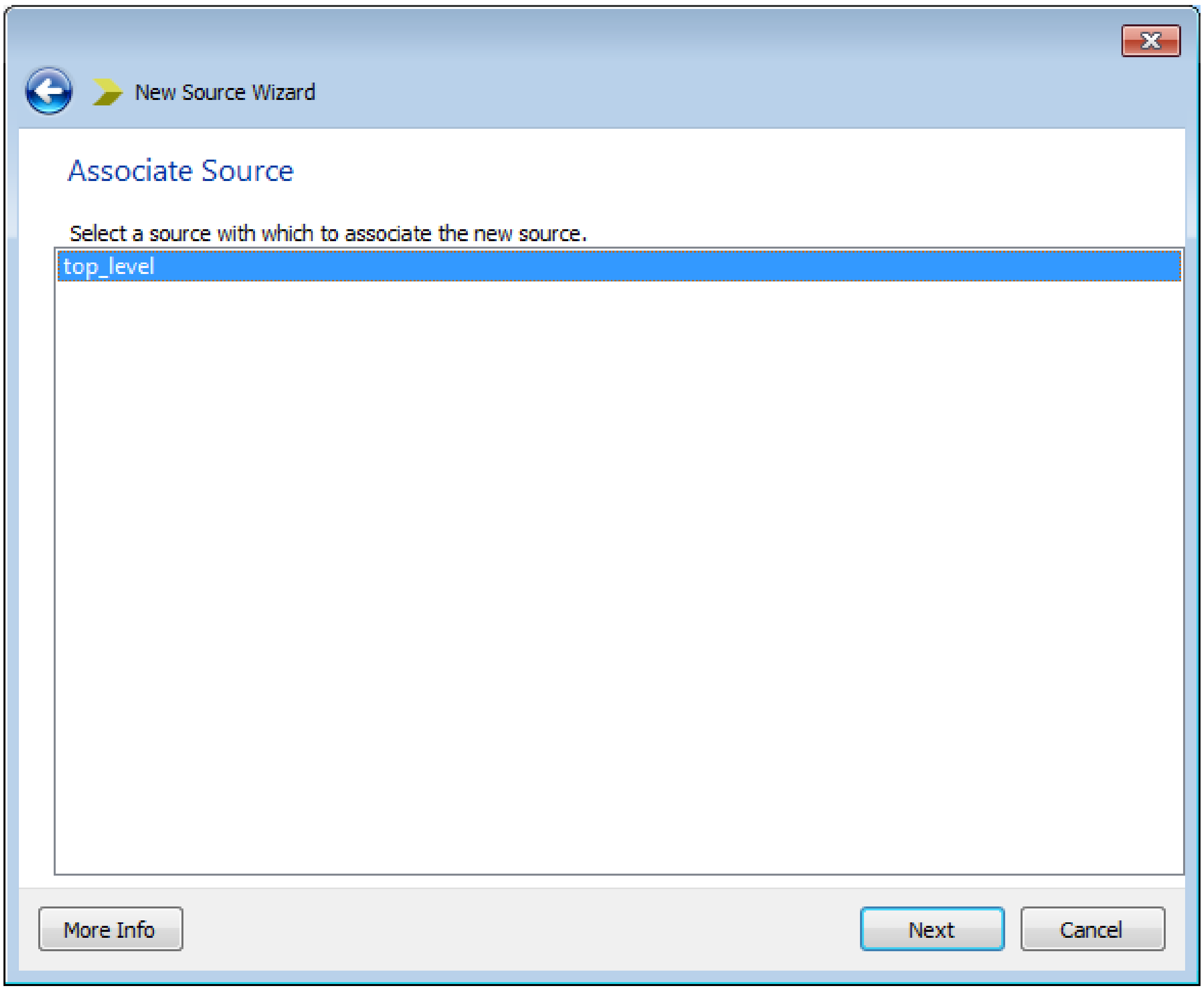
File name: top_level_tb

Location: E:\spartan3E\tutorial_1

[✓] Add to project

Figure 4.1: The New Source Wizard with values filled in for setting up a test bench.

2. The **Associate Source** window will appear as shown in Figure 4.2. top_level is highlighted as the source to be associated with the new test bench. Click **Next**.

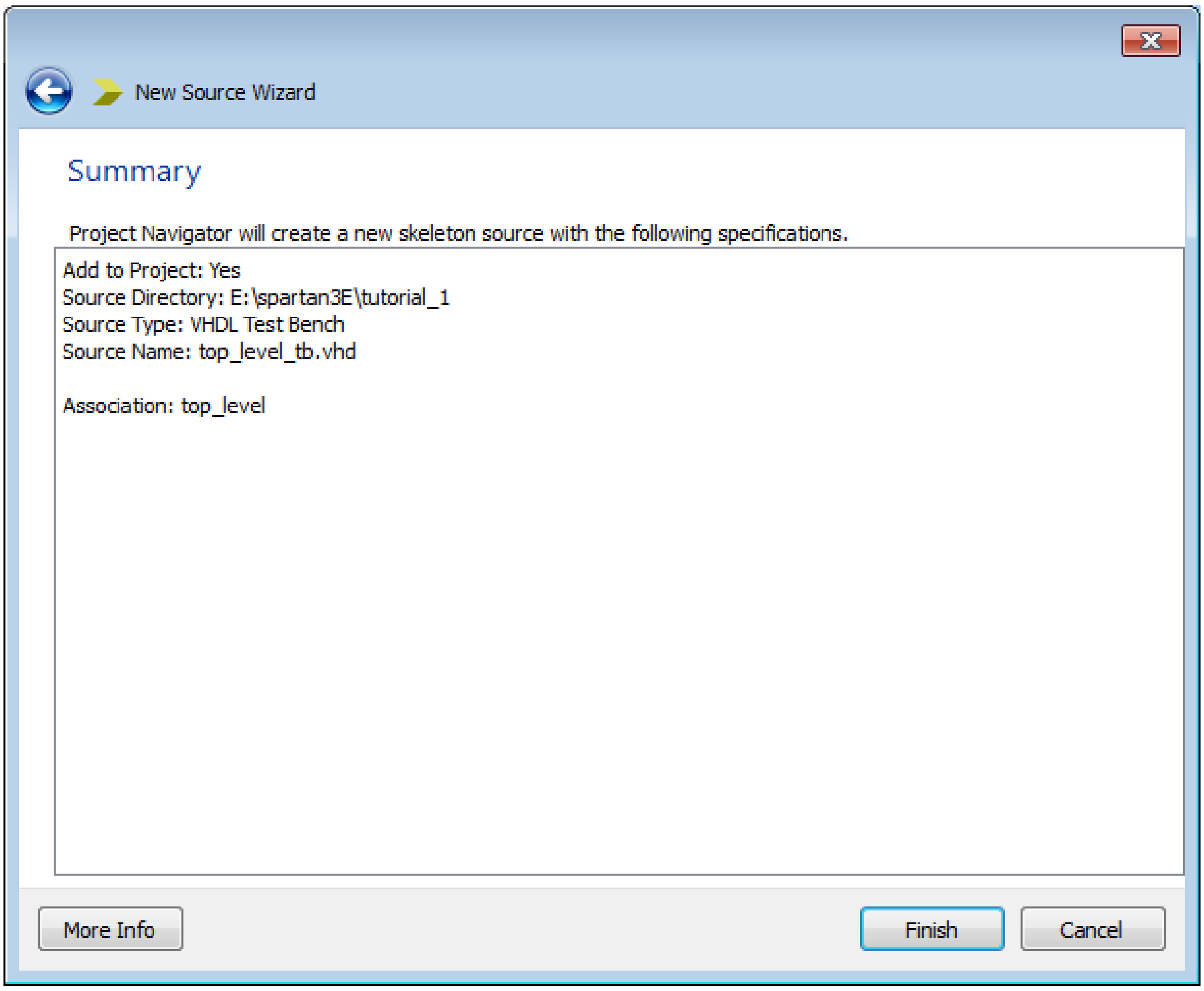


Select a source with which to associate the new source.

top_level

Figure 4.2: New Source Wizard, Associate Source window.

3. The **Summary** window shown in Figure 4.3 will appear. Click **Finish**.



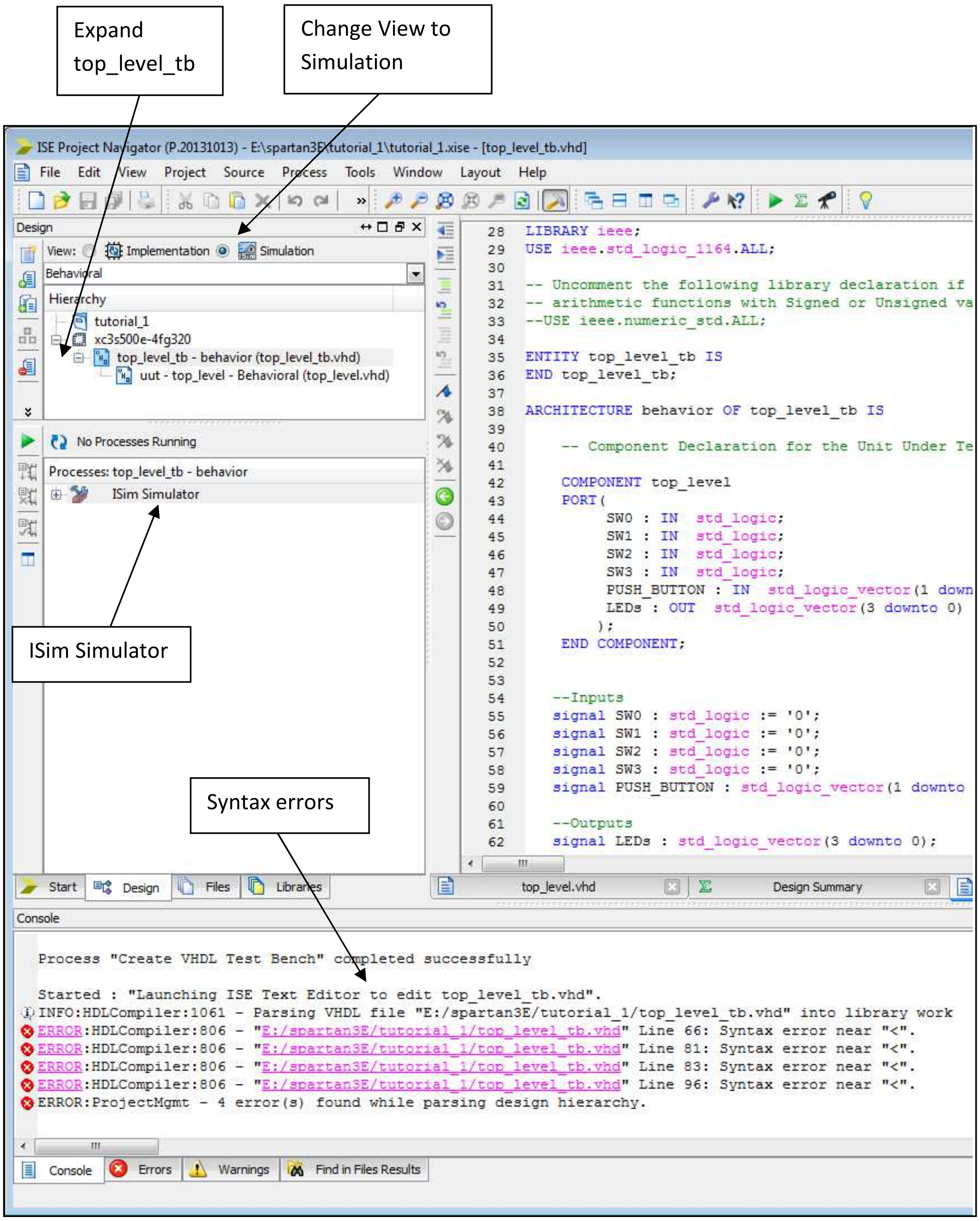
Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: E:\spartan3E\tutorial_1
Source Type: VHDL Test Bench
Source Name: top_level_tb.vhd

Association: top_level

Figure 4.3: New Source Wizard, Summary window.

3. Change the **View** from **Implementation** to **Simulation** by clicking the **Simulation** check box toward the top left of the screen, as shown in Figure 4.4. When this is done, **top_level_tb** will appear as the top level module. Click on the "+" next to **top_level_tb** to show the **uut - top_level** sub-module. Note that the **Processes** window will now contain an item called **ISim Simulator**.


Expand top_level_tb
Change View to Simulation
ISim Simulator
Syntax errors
ISE Project Navigator (P.20131013) - E:\spartan3E\tutorial_1\tutorial_1.xise - [top_level_tb.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
tutorial_1
xc3s500e-4fg320
top_level_tb - behavior (top_level_tb.vhd)
uut - top_level - Behavioral (top_level.vhd)
No Processes Running
Processes: top_level_tb - behavior
ISim Simulator
Start Design Files Libraries
top_level.vhd
Design Summary
Console
Console Errors Warnings Find in Files Results

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if
-- arithmetic functions with Signed or Unsigned va
--USE ieee.numeric_std.ALL;

ENTITY top_level_tb IS
END top_level_tb;

ARCHITECTURE behavior OF top_level_tb IS

    -- Component Declaration for the Unit Under Te

    COMPONENT top_level
    PORT(
         SW0 : IN  std_logic;
         SW1 : IN  std_logic;
         SW2 : IN  std_logic;
         SW3 : IN  std_logic;
         PUSH_BUTTON : IN  std_logic_vector(1 down
         LEDs : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal SW0 : std_logic := '0';
   signal SW1 : std_logic := '0';
   signal SW2 : std_logic := '0';
   signal SW3 : std_logic := '0';
   signal PUSH_BUTTON : std_logic_vector(1 downto

   --Outputs
   signal LEDs : std_logic_vector(3 downto 0);
```

```
Process "Create VHDL Test Bench" completed successfully

Started : "Launching ISE Text Editor to edit top_level_tb.vhd".
INFO:HDLCompiler:1061 - Parsing VHDL file "E:/spartan3E/tutorial_1/top_level_tb.vhd" into library work
ERROR:HDLCompiler:806 - "E:/spartan3E/tutorial_1/top_level_tb.vhd" Line 66: Syntax error near "<".
ERROR:HDLCompiler:806 - "E:/spartan3E/tutorial_1/top_level_tb.vhd" Line 81: Syntax error near "<".
ERROR:HDLCompiler:806 - "E:/spartan3E/tutorial_1/top_level_tb.vhd" Line 83: Syntax error near "<".
ERROR:HDLCompiler:806 - "E:/spartan3E/tutorial_1/top_level_tb.vhd" Line 96: Syntax error near "<".
ERROR:ProjectMgmt - 4 error(s) found while parsing design hierarchy.
```

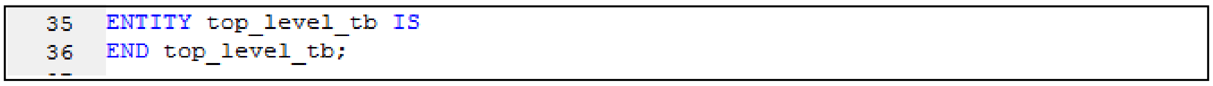
Figure 4.4: Simulation view.

5. Note that some syntax errors are reported at this stage, as shown in Figure 4.4. Ignore these for now.

4.2 top_level_tb.vhd – TestBench Code

This section briefly outlines different parts of the testbench code and their functions.

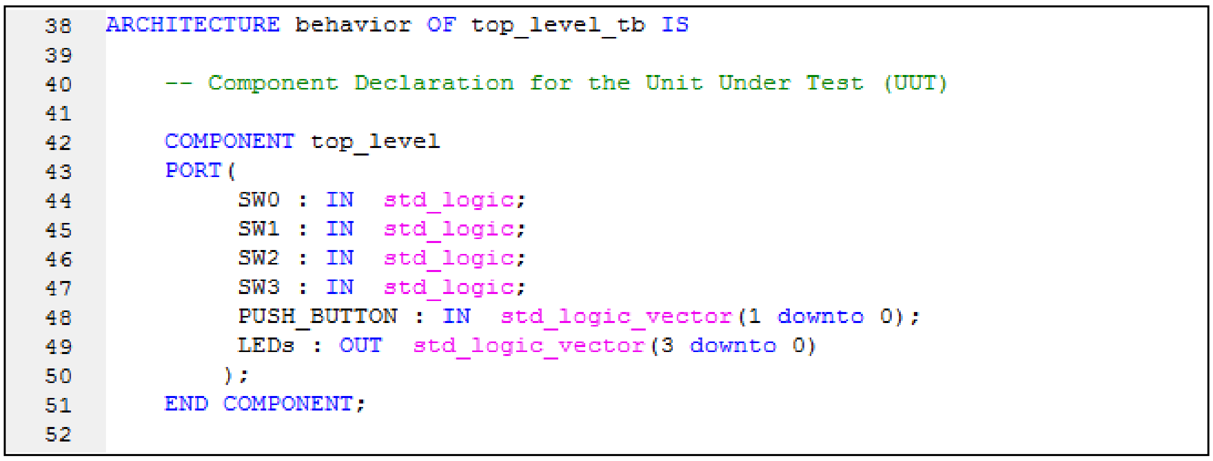
The entity for the **top_level_tb** testbench does not have any inputs or outputs, as shown in Figure 4.5.



```vhdl
ENTITY top_level_tb IS
END top_level_tb;
```

Figure 4.5: top_level_tb.vhd - entity.

The **top_level** component is declared in the architecture block of **top_level_tb**, as shown in Figure 4.6.

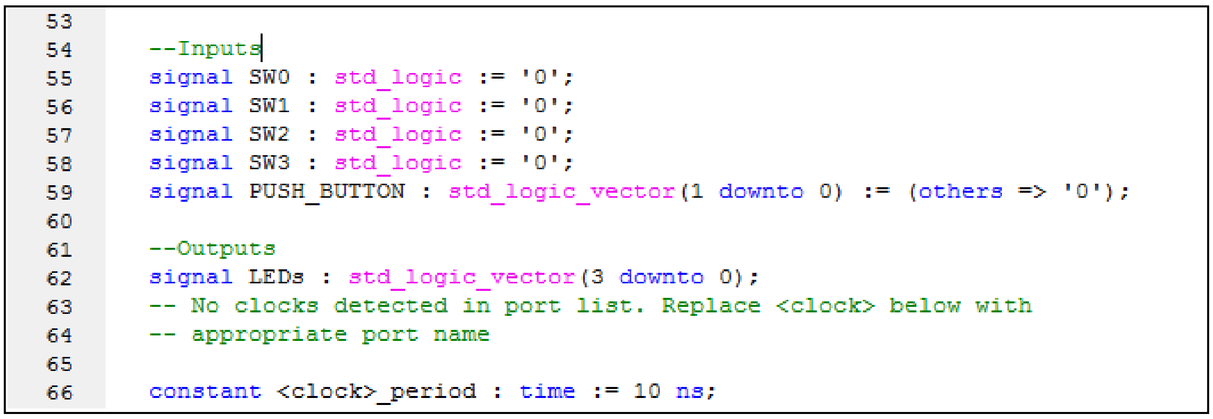


```vhdl
ARCHITECTURE behavior OF top_level_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT top_level
    PORT(
         SW0 : IN  std_logic;
         SW1 : IN  std_logic;
         SW2 : IN  std_logic;
         SW3 : IN  std_logic;
         PUSH_BUTTON : IN  std_logic_vector(1 downto 0);
         LEDs : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;

```

Figure 4.6: Declaration of the top_level component in the top_level_tb architecture.

Signals for the inputs and outputs of the module being tested, in this case **top_level**, are declared in Figure 4.7. The input signals, **SW0-4** and **PUSH_BUTTON**, are initialised to ‘0’. The output signal, **LEDs**, does not need to be inititalised.

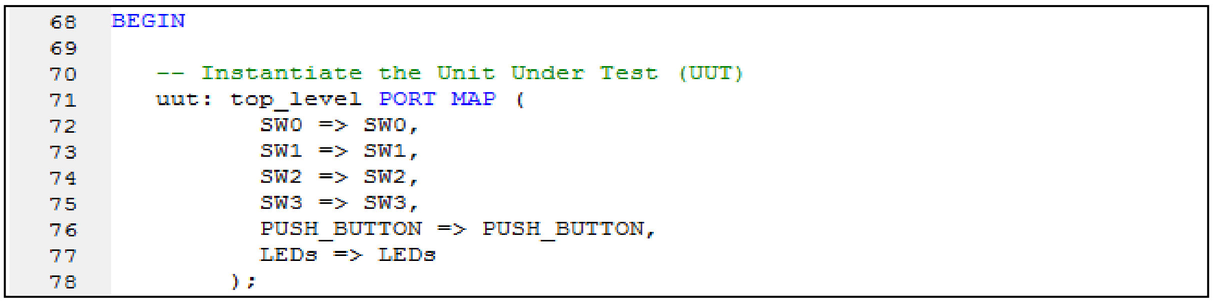
In addition, a constant for the clock period **<clock>_period**, has been automatically added, since it is often useful for synchronous designs that use a clock. It is expected that the user will replace **<clock>** with the name of the actual input signal that is used as the clock. The clock period, which is default to 10ns, can also be changed if needed. Since no clock is used in this design, this line of code will be later removed.



```
--Inputs
signal SW0 : std_logic := '0';
signal SW1 : std_logic := '0';
signal SW2 : std_logic := '0';
signal SW3 : std_logic := '0';
signal PUSH_BUTTON : std_logic_vector(1 downto 0) := (others => '0');

--Outputs
signal LEDs : std_logic_vector(3 downto 0);
-- No clocks detected in port list. Replace <clock> below with
-- appropriate port name

constant <clock>_period : time := 10 ns;
```

Figure 4.7: Signal declarations in the top_level_tb architecture.

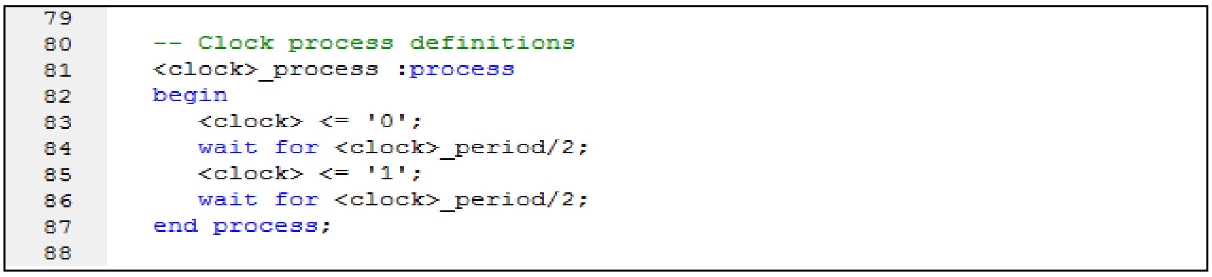
Figure 4.8 shows the first part of the **architecture** body, which instantiates the module that is being tested, in this case **top_level**. The module being tested is often called the **unit under test**, or **uut** in the code. The instantiation of this module maps the input and outport ports to signals.



```
BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: top_level PORT MAP (
          SW0 => SW0,
          SW1 => SW1,
          SW2 => SW2,
          SW3 => SW3,
          PUSH_BUTTON => PUSH_BUTTON,
          LEDs => LEDs
        );
```

Figure 4.8: Instantiation of the Unit Under Test (top_level) in the top_level_tb architecture.

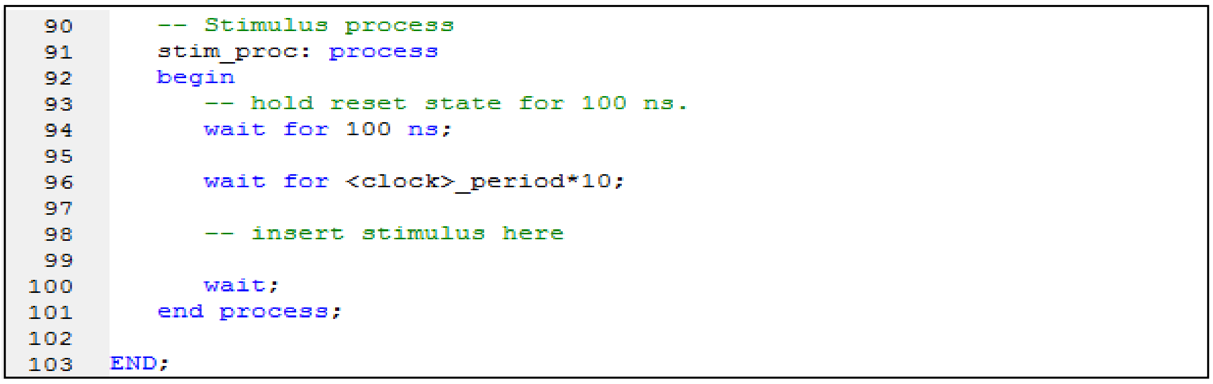
Figure 4.9 shows the next part of the **architecture** body, which defines a process which causes the clock signal to toggle every clock period / 2. Again, it is expected that the user will replace **<clock>** with the name of the actual input signal that is used as the clock. Since no clock is used in this design, these lines of code will be later removed.



```

   -- Clock process definitions
   <clock>_process :process
   begin
      <clock> <= '0';
      wait for <clock>_period/2;
      <clock> <= '1';
      wait for <clock>_period/2;
   end process;

```

Figure 4.9: <clock>_process in the top_level_tb architecture..

Figure 4.10 shows the final part of the **architecture** body, which defines the stimulus process. This process controls the input signals which will be fed into the module that is being tested. In the next sections, the "--insert stimulus here" comment in the code will be replaced by code written by the user to change the values of the input signals. The simulation will then be run, and the resulting output signals viewed in the simulator.



```
   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

      wait for <clock>_period*10;

      -- insert stimulus here

      wait;
   end process;

END;
```

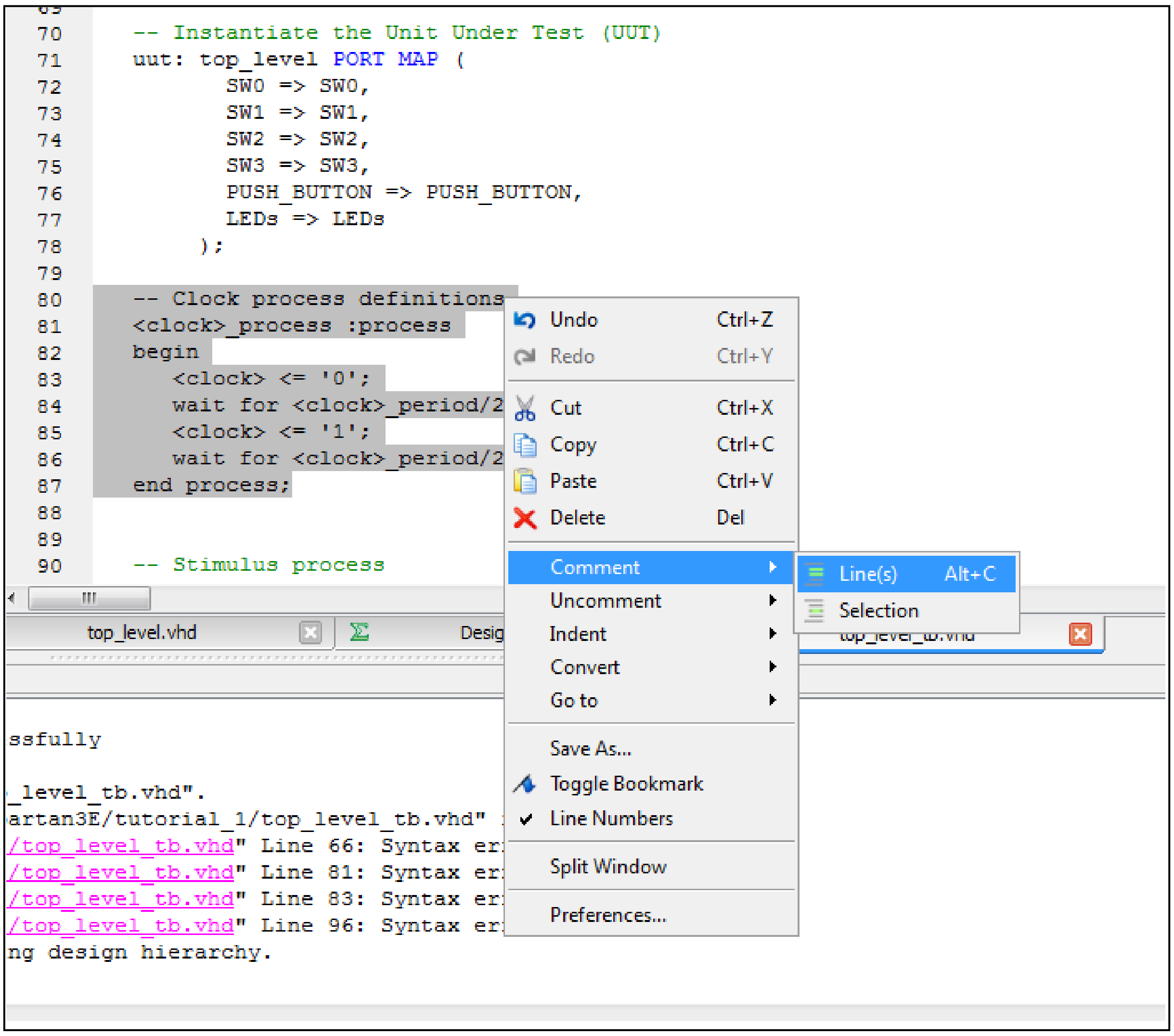
Figure 4.10: Declaration of top_level component.

4.3 Editing the TestBench

Code changes are to be made to **top_level_tb.vhd** as follows:

1. Comment out the **<clock>_period** constant (line 66 in Figure 4.7), the **clock process definition** (lines 80-87 in Figure 4.9), and **wait for <clock>_period*10** (line 96 in Figure 4.10). A clock is not used in this design and these lines are not needed. These lines are in fact the cause of the syntax errors in Figure 4.4.

   A line of code can be commented out in VHDL by placing "--" at the start of the line. Alternatively, a number of lines can be commented out at once, by selecting the code to be commented, right clicking on it, and selecting **Comment** → **Lines**, as shown in Figure 4.11.



```
70      -- Instantiate the Unit Under Test (UUT)
71      uut: top_level PORT MAP (
72              SW0 => SW0,
73              SW1 => SW1,
74              SW2 => SW2,
75              SW3 => SW3,
76              PUSH_BUTTON => PUSH_BUTTON,
77              LEDs => LEDs
78          );
79
80      -- Clock process definitions
81      <clock>_process :process
82      begin
83         <clock> <= '0';
84         wait for <clock>_period/2
85         <clock> <= '1';
86         wait for <clock>_period/2
87      end process;
88
89
90      -- Stimulus process
```

```
ssfully

_level_tb.vhd".
artan3E/tutorial_1/top_level_tb.vhd"
/top_level_tb.vhd" Line 66: Syntax er
/top_level_tb.vhd" Line 81: Syntax er
/top_level_tb.vhd" Line 83: Syntax er
/top_level_tb.vhd" Line 96: Syntax er
ng design hierarchy.
```

Figure 4.11: Commenting out several lines of code at once.

2. Insert the code below after the **"-- insert stimulus here"** comment (line 98 in Figure 4.10) and before the final **wait** (line 100 in Figure 4.10).

```
SW0 <= '1';
PUSH_BUTTON(0) <= '1';
wait for 10 ns;

SW1 <= '1';
PUSH_BUTTON(1) <= '1';
wait for 10 ns;

SW0 <= '0';
SW2 <= '1';
PUSH_BUTTON(0) <= '0';
wait for 10 ns;

SW1 <= '0';
SW3 <= '1';
PUSH_BUTTON(1) <= '0';
wait for 10 ns;

SW2 <= '0';
wait for 10 ns;

SW3 <= '0';
```

According to the functionality given in Table 1.1, this should result in the input and output waveforms as shown in Figure 4.12.

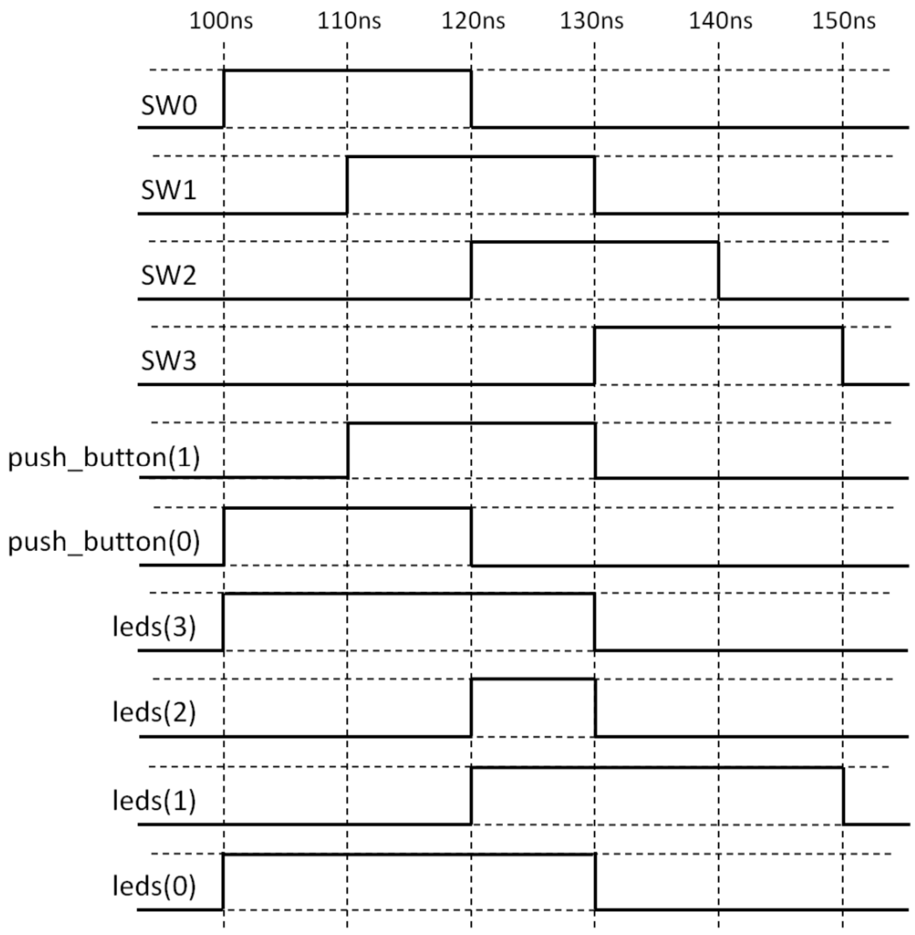


Figure 4.12: Expected input and output waveforms.

3. Save the file by selecting **File → Save** from the main menu. The syntax error messages should now disappear. A full listing of **top_level_tb.vhd** is provided in Appendix B.

4.4 Running the Simulator

1. As shown in Figure 4.13, if it is not already expanded, click on the '+' next to the **ISim Simulator**. This will expand out to show **Behavioral Check Syntax** and **Simulate Behavioral Model**.

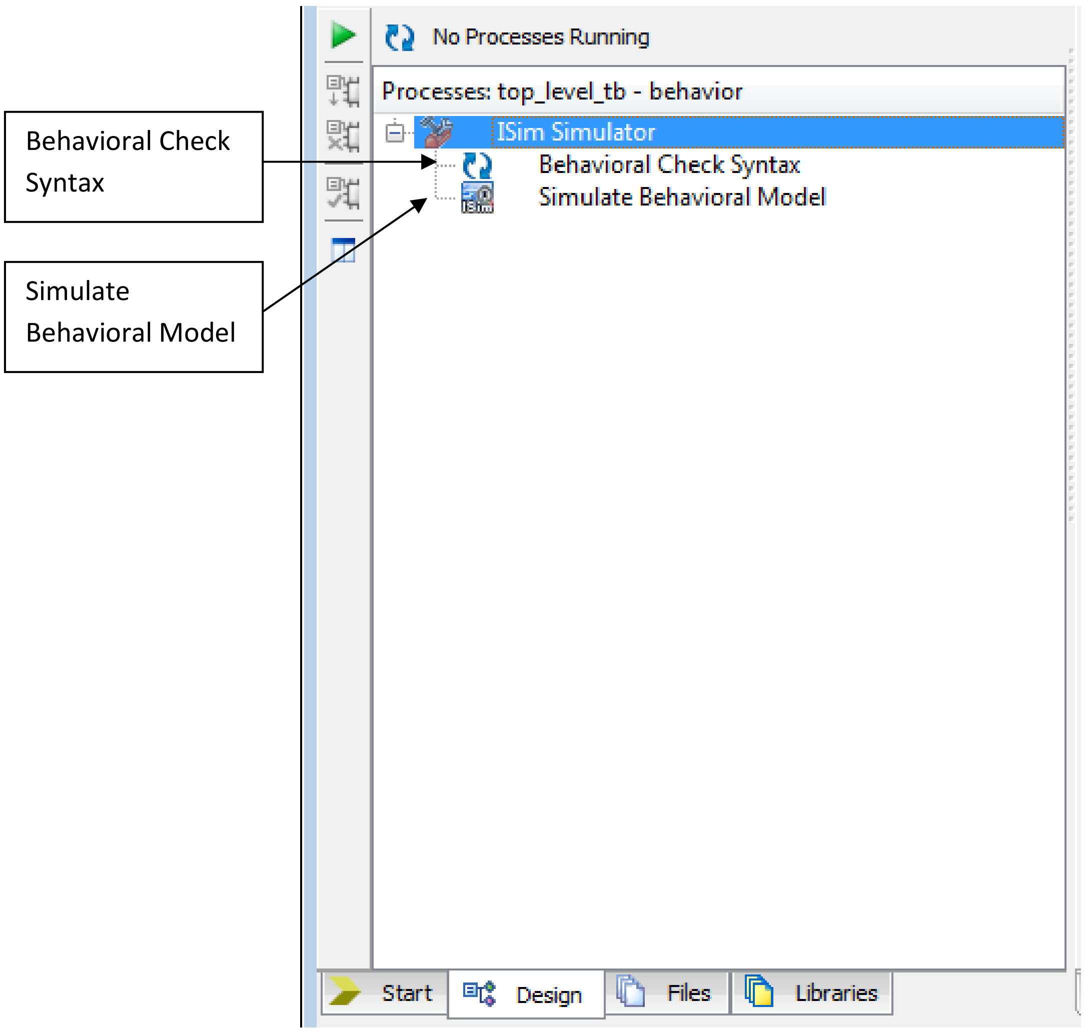


Figure 4.13: Portion of Project Navigator screen, with ISim Simulator expanded.

2. Double click on **Behavioral Check Syntax**.

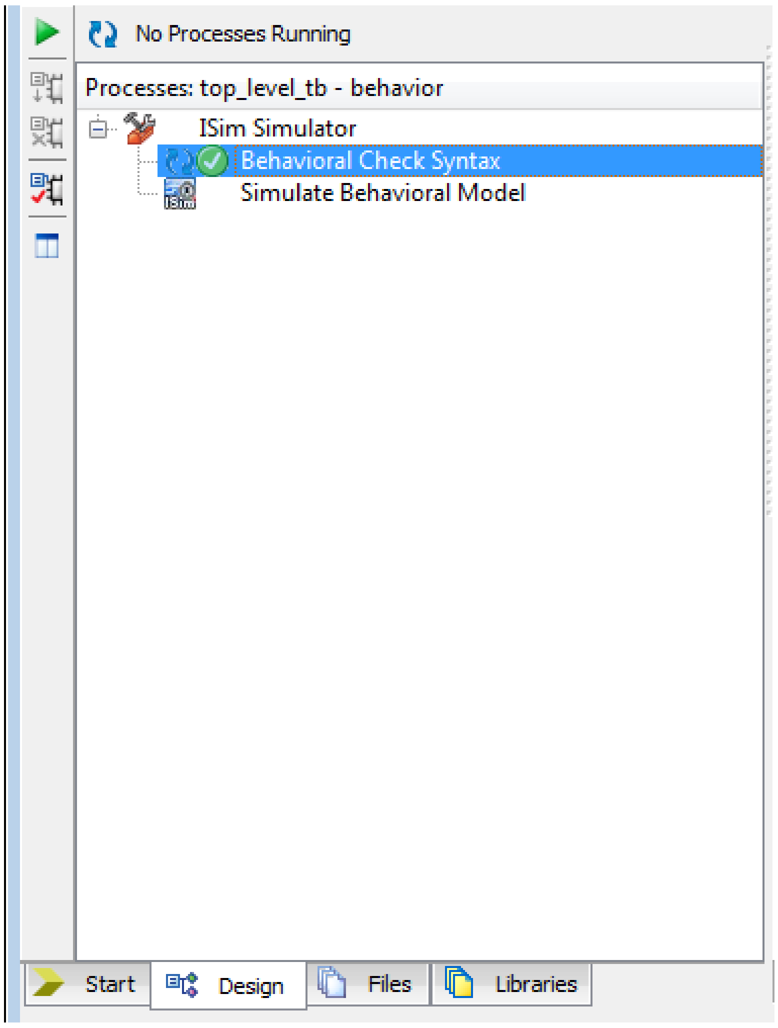


Figure 4.14: Portion of Project Navigator screen, with Behavioral Check Syntax selected.

3. Right click on **Simulate Behavioral Model**, and select **Process Properties**, as shown in Figure 4.15.

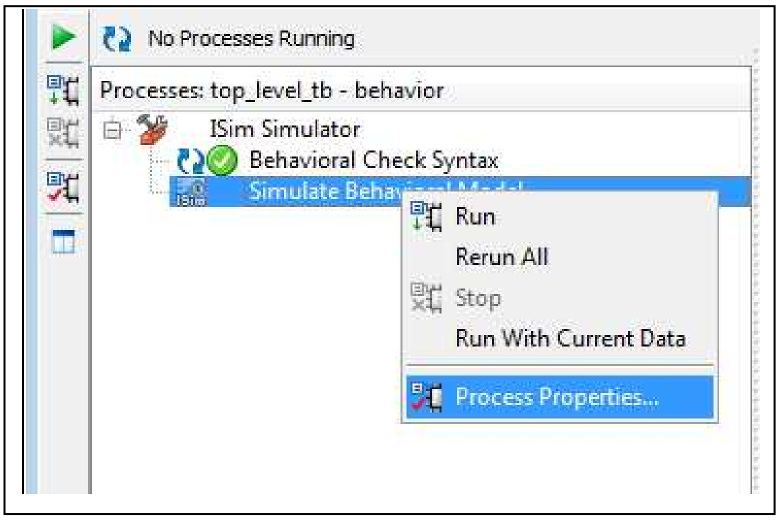
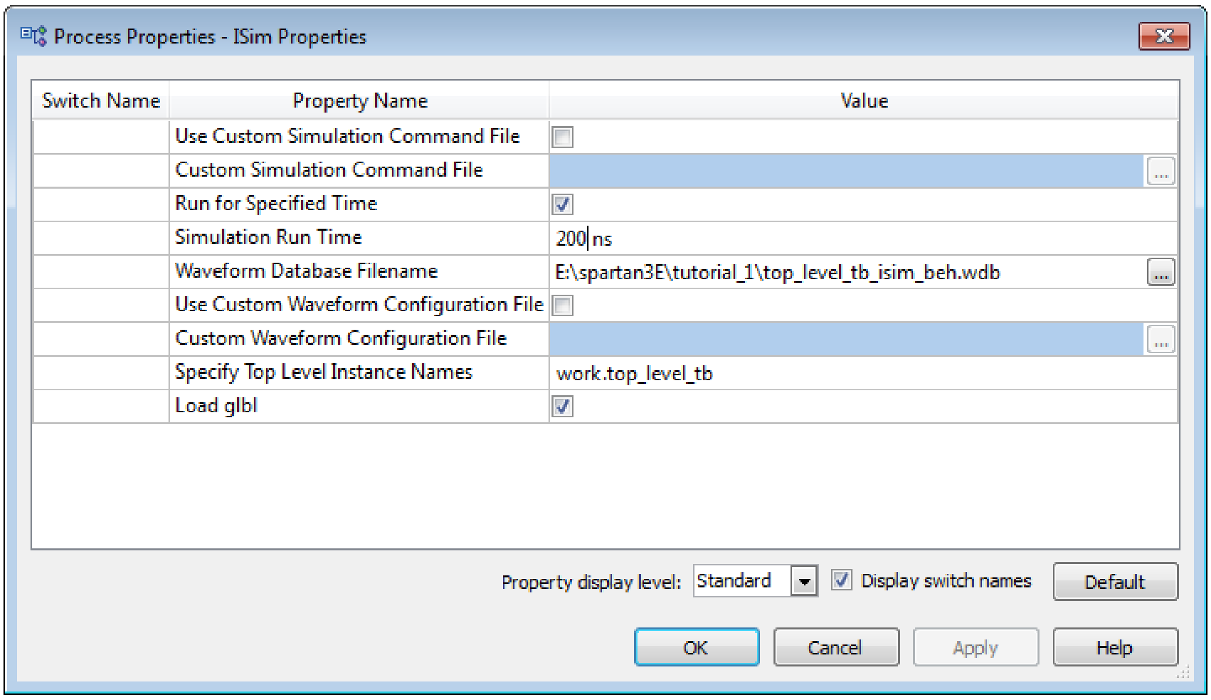


Figure 4.15: Portion of Project Navigator screen, with Simulate Behavioral Model, and Process Properties selected.

4. When the **Process Properties – ISim Properties** window appears, change Simulation Run Time to 200 ns, as shown in Figure 4.16.

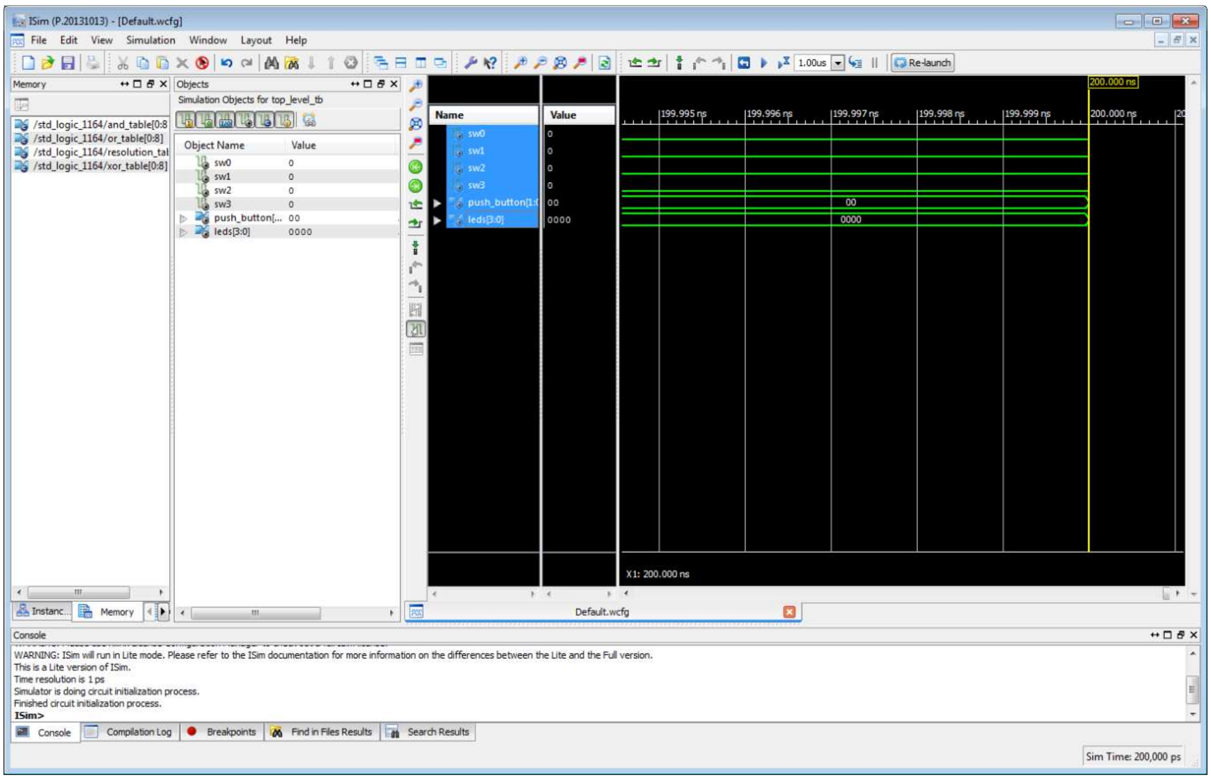


Switch Name	Property Name	Value
	Use Custom Simulation Command File	☐
	Custom Simulation Command File	
	Run for Specified Time	✓
	Simulation Run Time	200 ns
	Waveform Database Filename	E:\spartan3E\tutorial_1\top_level_tb_isim_beh.wdb
	Use Custom Waveform Configuration File	☐
	Custom Waveform Configuration File	
	Specify Top Level Instance Names	work.top_level_tb
	Load glbl	✓

Property display level: Standard ✓ Display switch names

Figure 4.16: Changing Simulation Run Time.

5. Double click on **Simulate Behavioral Model**. The ISim window will appear, as shown in Figure 4.17.



```
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Figure 4.17: ISim Window.

6. Choose **View→Zoom→To Full View** or press F6. This will result in the waveforms for the duration of the simulation being displayed, in this case from 0s to 200 ns. Click on the arrows to the left of the **push_button[1:0]** and **leds[3:0]** waveforms. This expands out the view of the individual signals. Figure 4.18 shows the display of waveforms for the input and output signals.

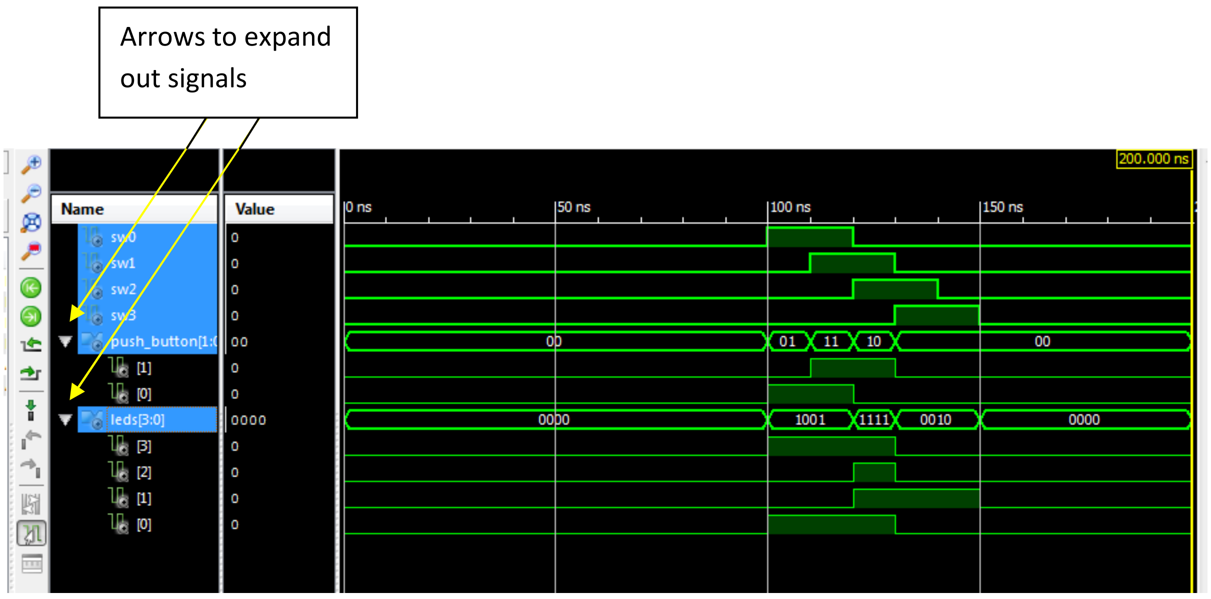


Figure 4.18: Simulation waveform display.

Input signals sw0-3, and push_button[1:0] in Figure 4.18, are determined by the stim_proc code of top_level_tb.vhd, discussed in Section 4.3. The leds[3:0] output signals are determined as a result of running the simulation. It can be seen that the leds[3:0] outputs are the same as the expected outputs in Figure 4.12.

5.0 Further Information

For further information about this tutorial, please contact:

Dr. Jasmine Banks
School of Electrical Engineering and Computer Science
Queensland University of Technology
GPO Box 2434, Brisbane 4001,
AUSTRALIA

Email: j.banks@qut.edu.au or jbanks@ieee.org.

5.0 References

[1] Spartan-3E FPGA Starter Kit Board User Guide, Online:
http://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf, accessed 25 Jan 2011.

[2] Roth, C. H., Digital Systems Design Using VHDL, PWS Publishing Company, 1998.

[3] Roth, C. H. And Kinney, L. L., Fundamentals of Logic Design, 6th edition, CENGAGE Learning, 2010.

[4] Spartan-3E FPGA Starter Kit Board Design Examples, Online:
http://www.xilinx.com/products/boards/s3estarter/reference_designs.htm, accessed 25 Jan 2011.

Appendix A – top_level.vhd

```
--------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    14:44:19 11/25/2013
-- Design Name:
-- Module Name:    top_level - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity top_level is
    Port ( SW0 : in  STD_LOGIC;
           SW1 : in  STD_LOGIC;
           SW2 : in  STD_LOGIC;
           SW3 : in  STD_LOGIC;
           PUSH_BUTTON : in  STD_LOGIC_VECTOR (1 downto 0);
           LEDs : out  STD_LOGIC_VECTOR (3 downto 0));
end top_level;

architecture Behavioral of top_level is

begin
  LEDs(0) <= SW0 or SW1;
  LEDs(1) <= SW2 or SW3;
  LEDs(2) <= (SW0 or SW1) and (SW2 or SW3);
  LEDs(3) <= PUSH_BUTTON(0) or PUSH_BUTTON(1);
end Behavioral;
```

Appendix B – top_level_tb.vhd

```
--------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:   16:23:07 11/25/2013
-- Design Name:
-- Module Name:   E:/spartan3E/tutorial_1/top_level_tb.vhd
-- Project Name:  tutorial_1
-- Target Device:
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: top_level
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic
-- and std_logic_vector for the ports of the unit under test.  Xilinx
-- recommends that these types always be used for the top-level I/O of a
-- design in order to guarantee that the testbench will bind correctly to
-- the post-implementation simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY top_level_tb IS
END top_level_tb;

ARCHITECTURE behavior OF top_level_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT top_level
    PORT(
         SW0 : IN  std_logic;
         SW1 : IN  std_logic;
         SW2 : IN  std_logic;
         SW3 : IN  std_logic;
         PUSH_BUTTON : IN  std_logic_vector(1 downto 0);
         LEDs : OUT  std_logic_vector(3 downto 0)
```

```
        );
    END COMPONENT;


   --Inputs
   signal SW0 : std_logic := '0';
   signal SW1 : std_logic := '0';
   signal SW2 : std_logic := '0';
   signal SW3 : std_logic := '0';
   signal PUSH_BUTTON : std_logic_vector(1 downto 0) := (others => '0');

   --Outputs
   signal LEDs : std_logic_vector(3 downto 0);
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name

--   constant <clock>_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: top_level PORT MAP (
          SW0 => SW0,
          SW1 => SW1,
          SW2 => SW2,
          SW3 => SW3,
          PUSH_BUTTON => PUSH_BUTTON,
          LEDs => LEDs
        );

--   -- Clock process definitions
--   <clock>_process :process
--   begin
--         <clock> <= '0';
--         wait for <clock>_period/2;
--         <clock> <= '1';
--         wait for <clock>_period/2;
--   end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

--      wait for <clock>_period*10;

      -- insert stimulus here
      SW0 <= '1';
      PUSH_BUTTON(0) <= '1';
      wait for 10 ns;
      SW1 <= '1';
      PUSH_BUTTON(1) <= '1';
      wait for 10 ns;

      SW0 <= '0';
      SW2 <= '1';
      PUSH_BUTTON(0) <= '0';
      wait for 10 ns;

      SW1 <= '0';
      SW3 <= '1';
      PUSH_BUTTON(1) <= '0';
      wait for 10 ns;

      SW2 <= '0';
      wait for 10 ns;

      SW3 <= '0';
      wait;
   end process;

END;
```