USER'S MANUAL

FOR

FPGA XC3S500E DEVELOPMENT BOARD

Manufactured By

LOGSUN SYSTEMS

(ISO 9001:2008 Certified)

Mobile: 9850588864

Ph.- 02024356456/24351400

Email: info@logsun.com / info@logsunonline.com

Visit us at: www.logsun.com /www.logsunonline.com

Table of contents

Chapter 1: Introduction

1. a. What is programmable logic?

In the world of digital electronic systems, there are three basic kinds of devices: memory, microprocessors, and logic. Memory devices store random information such as the contents of a spreadsheet or database. Microprocessors execute software instructions to perform a wide variety of tasks such as running a word processing program or video game. Logic devices provide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform.

Fixed Logic versus Programmable Logic

Logic devices can be classified into two broad categories - fixed and programmable. As the name suggests, the circuits in a fixed logic device are permanent, they perform one function or set of functions - once manufactured, they cannot be changed. On the other hand, programmable logic devices (PLDs) are standard, off-the-shelf parts that offer customers a wide range of logic capacity, features, speed, and voltage characteristics - and these devices can be changed at any time to perform any number of functions.

With fixed logic devices, the time required to go from design, to prototypes, to a final manufacturing run can take from several months to more than a year, depending on the complexity of the device. And, if the device does not work properly, or if the requirements change, a new design must be developed.

With programmable logic devices, designers use inexpensive software tools to quickly develop, simulate, and test their designs. Then, a design can be quickly programmed into a device, and immediately tested in a live circuit.

FPGAs

The major type of programmable logic devices is field programmable gate arrays (FPGAs).FPGAs offer the highest amount of logic density, the most features, and the highest performance. The largest FPGA now shipping, part of the Xilinx Virtex™ line of devices, provides eight million "system gates" (the relative density of logic). These advanced devices also offer features such as built-in hardwired processors (such as the IBM Power PC), substantial amounts of memory, clock management systems, and support for many of the latest, very fast device-to-device signaling technologies. FPGAs are used in a wide variety of applications ranging from data processing and storage, to instrumentation, telecommunications, and digital signal processing.

Advantages of PLD

Fixed logic devices and PLDs both have their advantages. Fixed logic devices, for example, are often more appropriate for large volume applications because they can be mass-produced more economically. For certain applications where the very highest performance is required, fixed logic devices may also be the best choice.

However, programmable logic devices offer a number of important advantages over fixed logic devices, including:

- PLDs offer customers much more flexibility during the design cycle because design iterations are simply a matter of changing the programming file, and the results of design changes can be seen immediately in working parts.
- PLDs do not require long lead times for prototypes or production parts - the PLDs are already on a distributor's shelf and ready for shipment.
- PLDs do not require customers to pay for large NRE costs and purchase expensive mask sets - PLD suppliers incur those costs when they design their programmable devices and are able to amortize those costs over the multi-year lifespan of a given line of PLDs.
- PLDs allow customers to order just the number of parts they need, when they need them, allowing them to control inventory. Customers who use fixed logic devices often end up with excess inventory which must be scrapped, or if demand for their product surges, they may be caught short of parts and face production delays.

- PLDs can be reprogrammed even after a piece of equipment is shipped to a customer. In fact, thanks to programmable logic devices, a number of equipment manufacturers now tout the ability to add new features or upgrade products that already are in the field. To do this, they simply upload a new programming file to the PLD, via the Internet, creating new hardware logic in the system.

1.b. Product Information

The Spartan®-3E family of Field-Programmable Gate Arrays (FPGAs) is specifically designed to meet the needs of high volume, cost-sensitive consumer electronic applications. The Spartan-3E family builds on the success of the earlier Spartan-3 family by increasing the amount of logic per I/O, significantly reducing the cost per logic cell. New features improve system performance and reduce the cost of configuration. These Spartan-3E FPGA enhancements, combined with advanced 90 nm process technology, deliver more functionality and bandwidth per dollar than was previously possible, setting new standards in the programmable logic industry. The Spartan-3E family is a superior alternative to mask programmed ASICs. FPGAs avoid the high initial cost, the lengthy development cycles, and the inherent inflexibility of conventional ASICs. Also, FPGA programmability permits design upgrades in the field with no hardware replacement necessary, an impossibility with ASICs.

The FPGA board is a low cost universal platform for testing and verifying designs based on Xilinx PLD's. The purpose of FPGA project board is to teach the basic concepts of VLSI designing along with various electronics circuits .using this project-board user can verify his PLD design with complete application and also it gives a complete set of modules for project development for students. FPGA development board comes along with onboard various interfacing sections.

In FPGA Development Kit we use SPARTAN XC3S500E. The XC3S500E having 500K system gates,10476 equivalent logic cells,73k distributed RAM bit's,360K block RAM bit's, 232 maximum user I/O and 92 maximum differential I/O pairs.

1. **Keyboard section**
2. **LCD section**

3. **ADC/DAC section**
4. **DC motor section**
5. **Stepper motor section**
6. **Seven segment section**
7. **Seven segment + Keyboard section**
8. **Relay Buzzer section**

- **HDL codes to realize all the logic gates**
  1. **Design and Simulate of adder, Carry Look Ahead Adder.**
  2. **Design of 2-to-4 decoder**
  3. **Design of 8-to-3 encoder(without and with parity)**
  4. **Design of 8-to-1 multiplexer**
  5. **Design of 4 bit binary to gray converter**
  6. **Design of multiplexer/ Demultiplexer ,comparator**
  7. **Design of full adder using 1 modeling styles**
  8. **Design of flip flop: SR,JK,T,D**
  9. **Design of 4-bit binary ,BCD counter or any sequence counter**
  10. **Design of N-bit Register of Serial-in Serial-out, Serial in Parallel out, parallel in Serial out and Parallel in Parallel out**
  11. **Design of ALU to perform- ADD,SUB,AND-OR**

Note : Implementing the above designs on Xilinx.

Above sections are on the FPGA DEVELOPMENT KIT .

1 .c. Peripherals:

1.1Liquid crystal display:

In LGS-FPGA/CPLD 16x2 LCD is given in the form of plug and play. LCD can be connected to the FPGA through the port. LCD is connected in the 8 bit mode. And the standard subroutine is given so that the application can be easily demonstrated and also for further implementation the subroutine can be easily embedded for which one has to do very few changes.

1.2 DC motor:

Electrical motors are everywhere around us. Almost all the electro-mechanical movements we see around us are caused either by an A.C. or a DC motor**.** Here we will be exploring this kind of motors. This is a device that converts DC electrical energy to a mechanical energy. This DC or direct current motor works on the principal, when a current carrying conductor is placed in a magnetic_field, it experiences a torque and has a tendency to move. This is known as motoring action. If the direction of current in the wire is reversed, the direction of rotation also reverses.

1.3 Stepper motor:

A stepper motor step motor is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any feedback sensor, as long as the motor is carefully sized to the application.

1.4 Seven Segment:

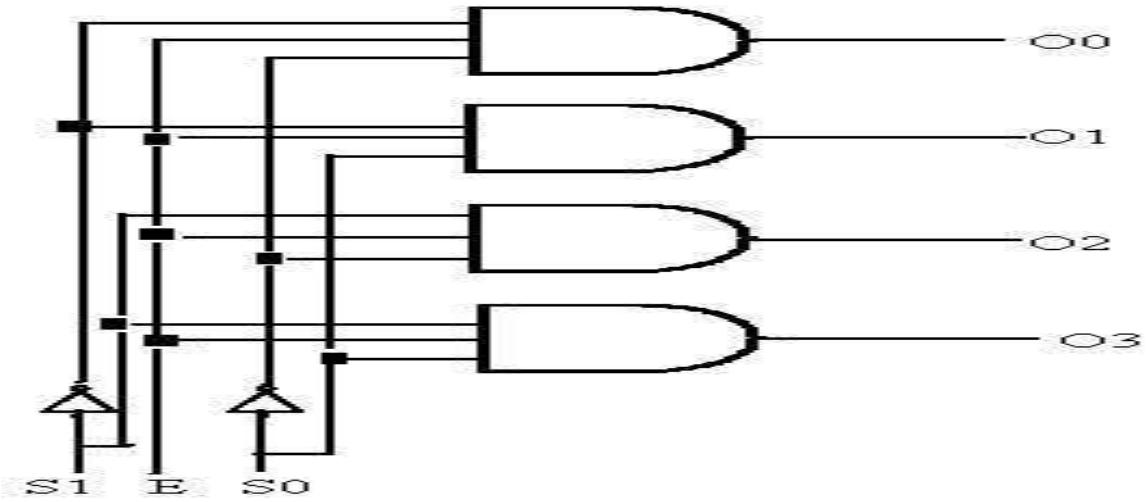
A seven-segment display, or seven-segment indicator, is a form of electronics display___device for displaying decimal numerals that is an alternative to the more complex dot_matrix displays. Seven-segment displays are widely used in digital_clocks, electronic meters, basic calculators, and other electronic devices that display numerical information

1.5 Decoder

In digital electronics, a decoder can take the form of a multiple-input, multiple-output logic circuit that converts coded inputs into coded outputs, where the input and output codes are different e.g. n-to-2n , binary-coded decimal decoders. Decoding is necessary in applications such as data multiplexing, 7 segment display and memory address decoding.

S_1	S_0	E	O_0	O_1	O_2	O_3
×	×	0	0	0	0	0
0	0	1	1	0	0	0
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	1

Truth table of 2-to-4 decoder



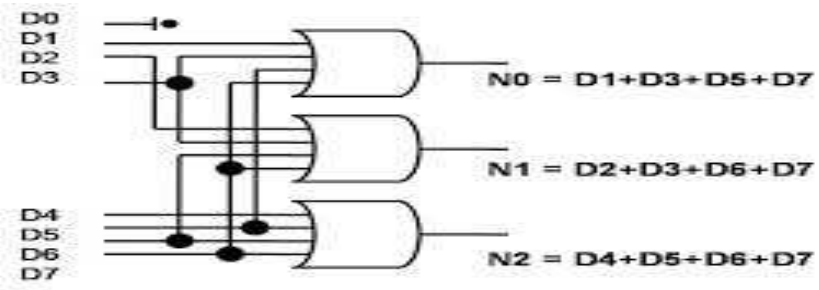
Logic Diagram of 2-to-4 decoder

1.8 Encoder

Unlike a multiplexer that selects one individual data input line and then sends that data to a single output line or switch, a **Digital Encoder** more commonly called a **Binary Encoder** takes *ALL* its data inputs one at a time and then converts them into a single encoded output. So we can say that a binary encoder is a multi-input combinational logic circuit that converts the logic level "1" data at its inputs into an equivalent binary code at its output.

I0	I1	I2	I3	I4	I5	I6	I7	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Truth table of 8-to-3 encoder



8-to-3 line Encoder

1.9 Multiplexer and Demultiplexer

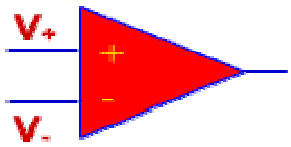
Multiplexer means many into one. A multiplexer is a circuit used to select and route any one of the several input signals to a signal output. A simple example of a non electronic circuit of a multiplexer is a single pole multiposition switch.

Demultiplexer means one to many. A demultiplexer is a circuit with one input and much output. By applying control signal, we can steer any input to the output. Few types of demultiplexer are 1-to 2, 1-to-4, 1-to-8 and 1- to 16 demultiplexer.

1.10 Comparator

A comparator is the simplest circuit that moves signals between the analog and digital worlds. What does a comparator do?

- Simply put, a comparator compares two analog signals and produces a one bit digital signal. The symbol for a comparator is shown below.



- The comparator output satisfies the following rules:
  - When V_+ is larger than V_- the output bit is 1.
  - When V_+ is smaller than V_- the output bit is 0

1.11 Flip Flop

In the electronics world, a **flip-flop** is a type of circuit that contains two states and is often used to store state information. By sending a signal to the flip-flop, the state can be changed. In sequential logic, it is the basic element of storage. Flip-flops are used in a number of electronics, including computers and communications equipment.

Inputs			Outputs		
S	R	C	Q	Q'	Comments
0	0	↑	Q	Q'	No change
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	?	?	Invalid

S-R Flip flop

Inputs			Outputs		
J	K	C	Q	Q'	Comments
0	0	↑	Q	Q'	No change
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	Q'	Q	Toggle

J-K Flip-flop

Inputs		Outputs		
D	C	Q	Q'	Comments
0	↑	0	1	RESET
1	↑	1	0	SET

D Flip-flop

1.12 Sequence Detector

The objective of this tutorial is to introduce the use of sequential logic. The sequence is a sequential circuit. In sequential logic the output depends on the current input values and also the previous inputs. When describing the behavior of a sequential logic circuit we talk about the state of the circuit. The state of a sequential circuit is a result of all previous inputs and determines the circuit's output and future behavior. This is why sequential circuits are often referred to as state machines. Most sequential circuits (including our sequence detector) use a clock signal to control when the circuit changes states. The inputs of the circuit along with the circuit's current state provide the information to determine the circuit's next state. The clock signal then controls the passing of this information to the state memory. The output depends only on the circuit's state; this is known as a Moore Machine.

Chapter 2: Features and Specification

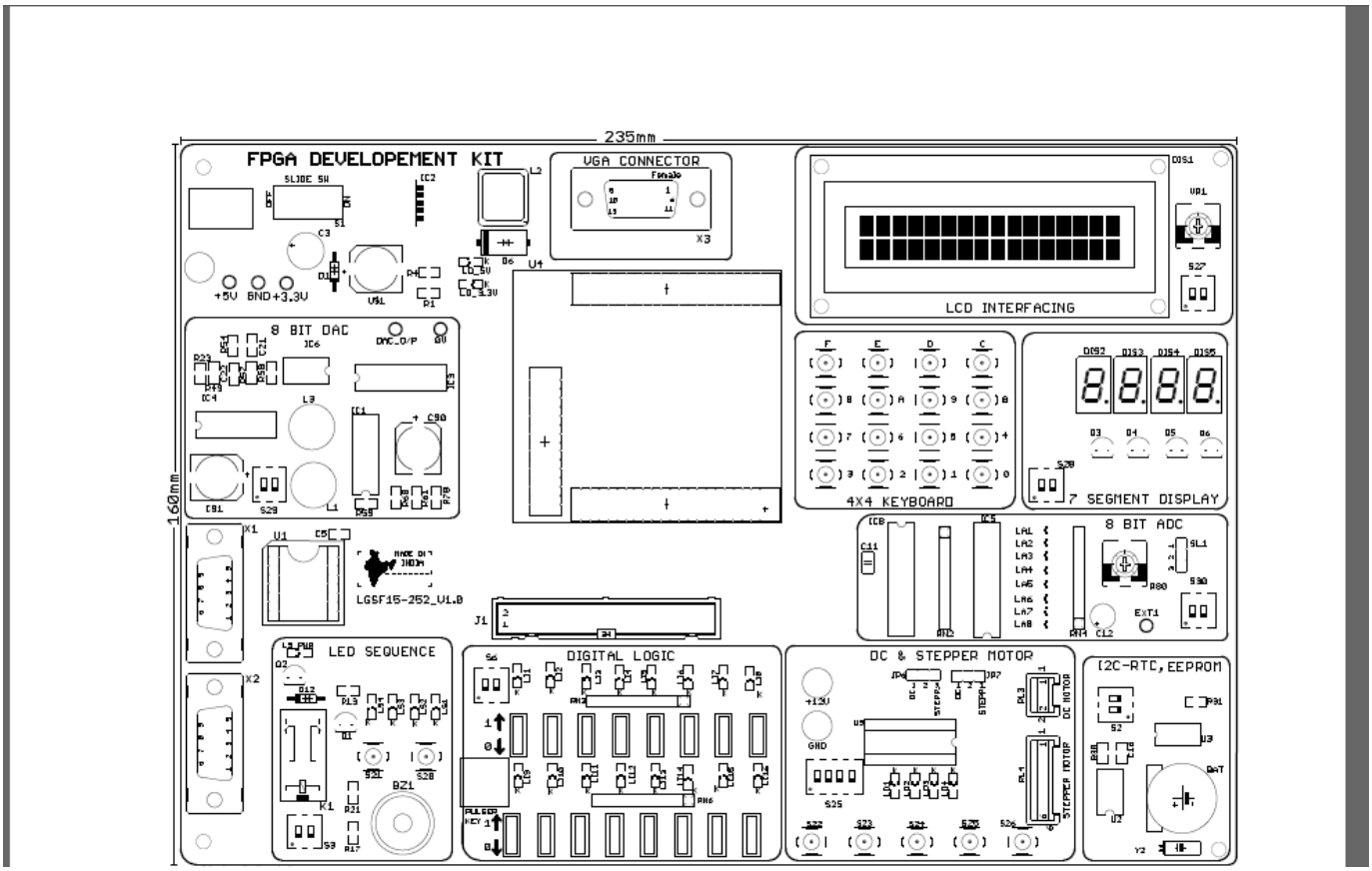
2. A Features and Specification of FPGA DEVELOPMENT KIT

- Supports for Xilinx PLD's
- Voltage supported 12V
- All FPGA I/Os accessible through headers
- 7-Segment display
- User selectable configuration modes using, Flash PROM,JTAG
- LED indicated output
- 4*4 keyboard matrix
- 16*2 character LCD display
- Short circuit protection

2. B Individual module specification

- **Keyboard section**
  - 4*4 keypad
- **Liquid crystal display (LCD) section**
  - 16 * 2 character LCD display
- **Relay buzzer**
  - Relay ON indication
- **Xilinx FPGA module**
  - XC3S500E FPGA from Xilinx

Chapter 3: Diagram



Chapter 4: Precautions

1) Verify the power on LED status after applying power to the trainer.

2) Connect the cable to the trainer only after conferming the above.

3) During downloading make sure that the jumper selections are proper.

4) Select the proper configuration mode during programming, else programming can fail.

5) Take care for adapter position before plugging on the board; this may cause damage to PLD device on power ON if plugged incorrectly.

6) Do not touch the FPGA, as your body static charge may damage FPGA.

7) Before implementation, it is necessary to lock the pins in user constraint file (UCF) as per the design and I/O is used.

8) For downloading the bit stream, the downloading circuit requires stable supply; hence it is recommended to use power supply given along with the trainer board.

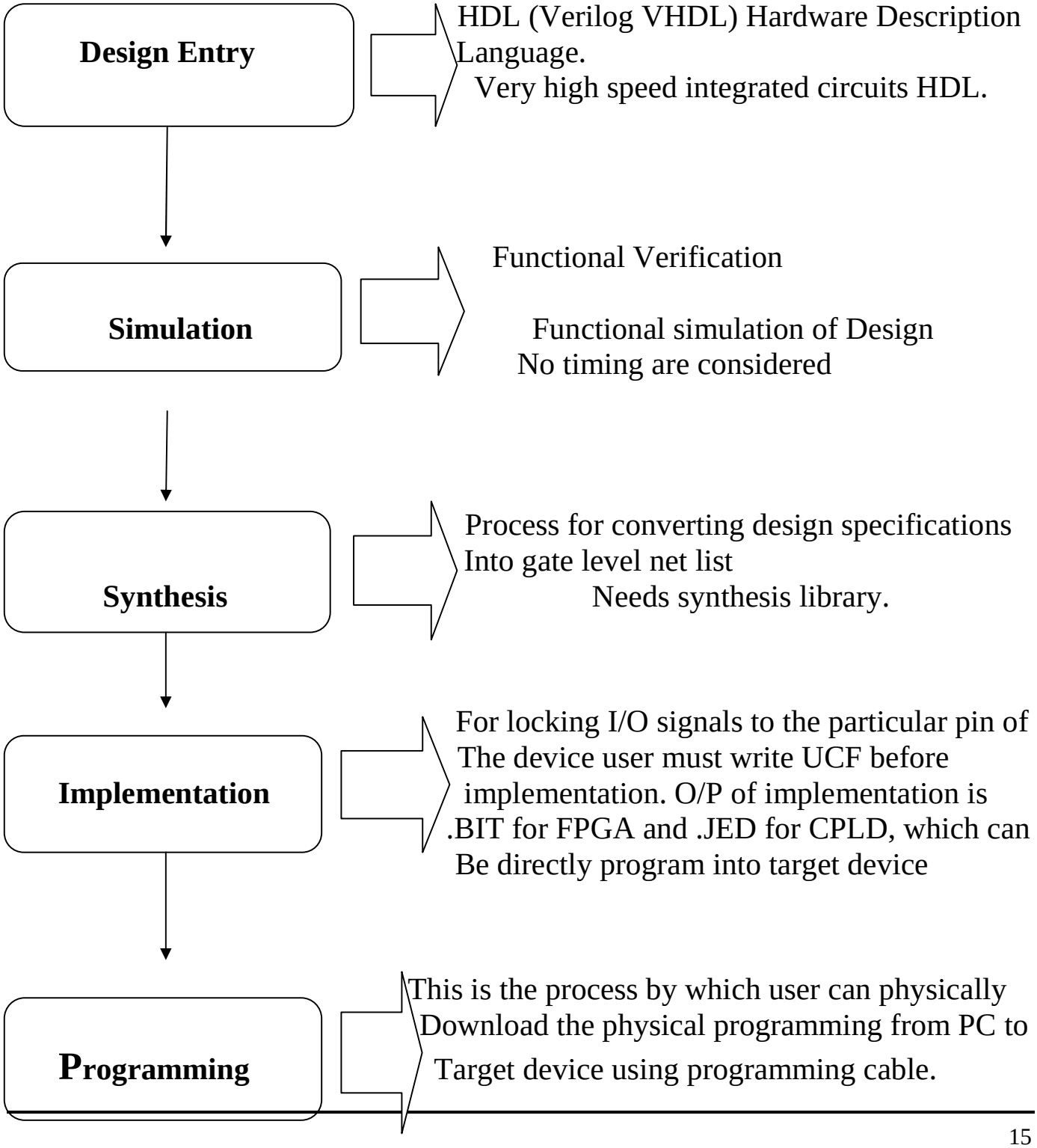
9) PLD devices are sensitive to surge current and voltages.

10) Kindly remove cable from holding its headers only. Removing the cables from holding its wires may cause damage to cable joints.

11) Proper selection of switch is must. When any one of section is in use then ON the switch only for that section. & make sure that switches on other sections are OFF.

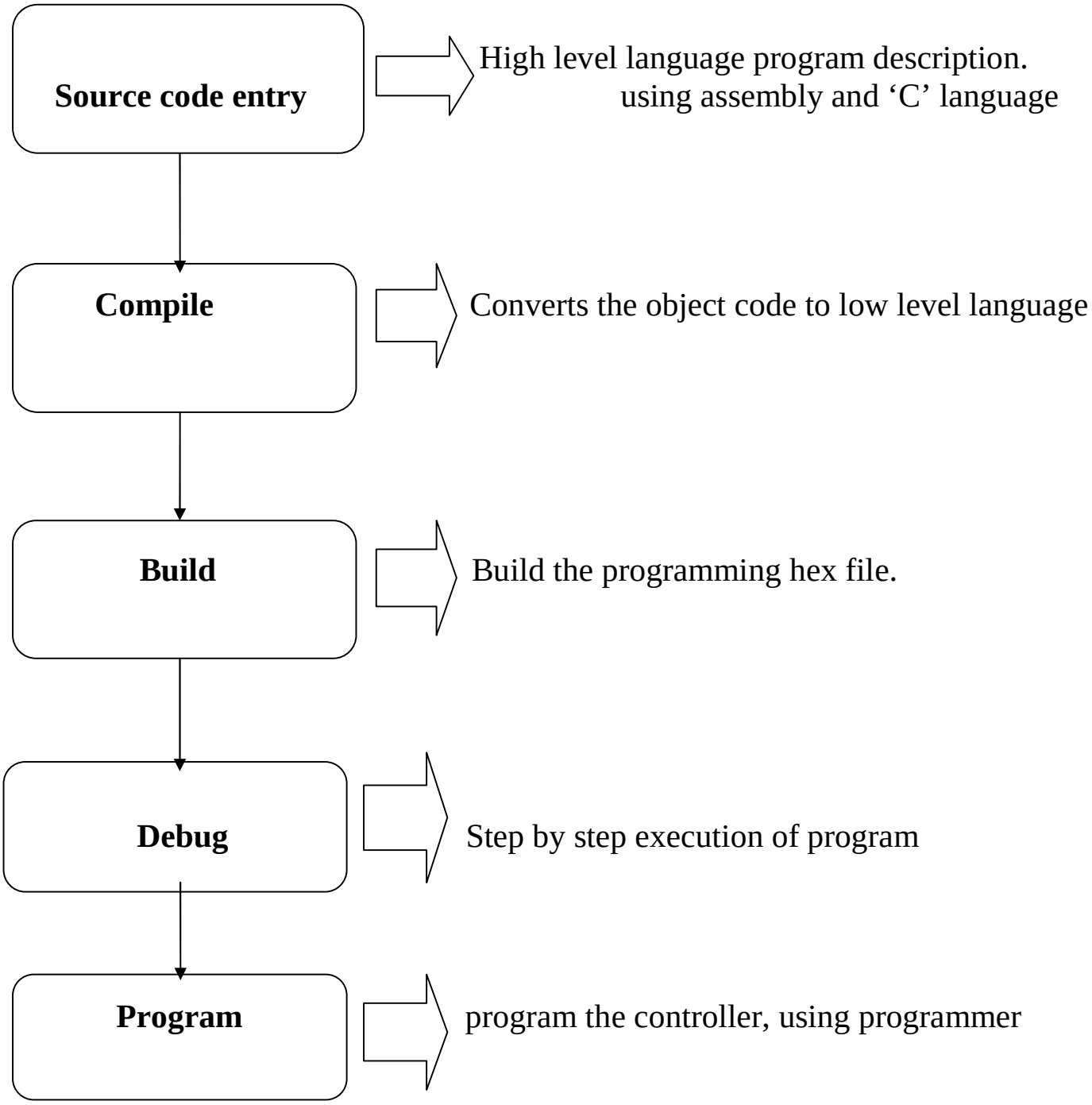
Chapter 5: Design Flow

5. A PLD design flow



To program CPLD, select JTAG mode.
To program FPGA, select boundry scan, slave Serial mode or master serial mode.

5. b Microcontroller design flow



5. C using EDA tools

In this chapter we will see how a project can be created in Xilinx EDA tool and how we can proceed to use FPGA project board to perform our experiment.
We take the example of AND gate and implement on devices.

Design flow for Xilinx ISE series software

Step 1: open Xilinx ISE design suite 12.2

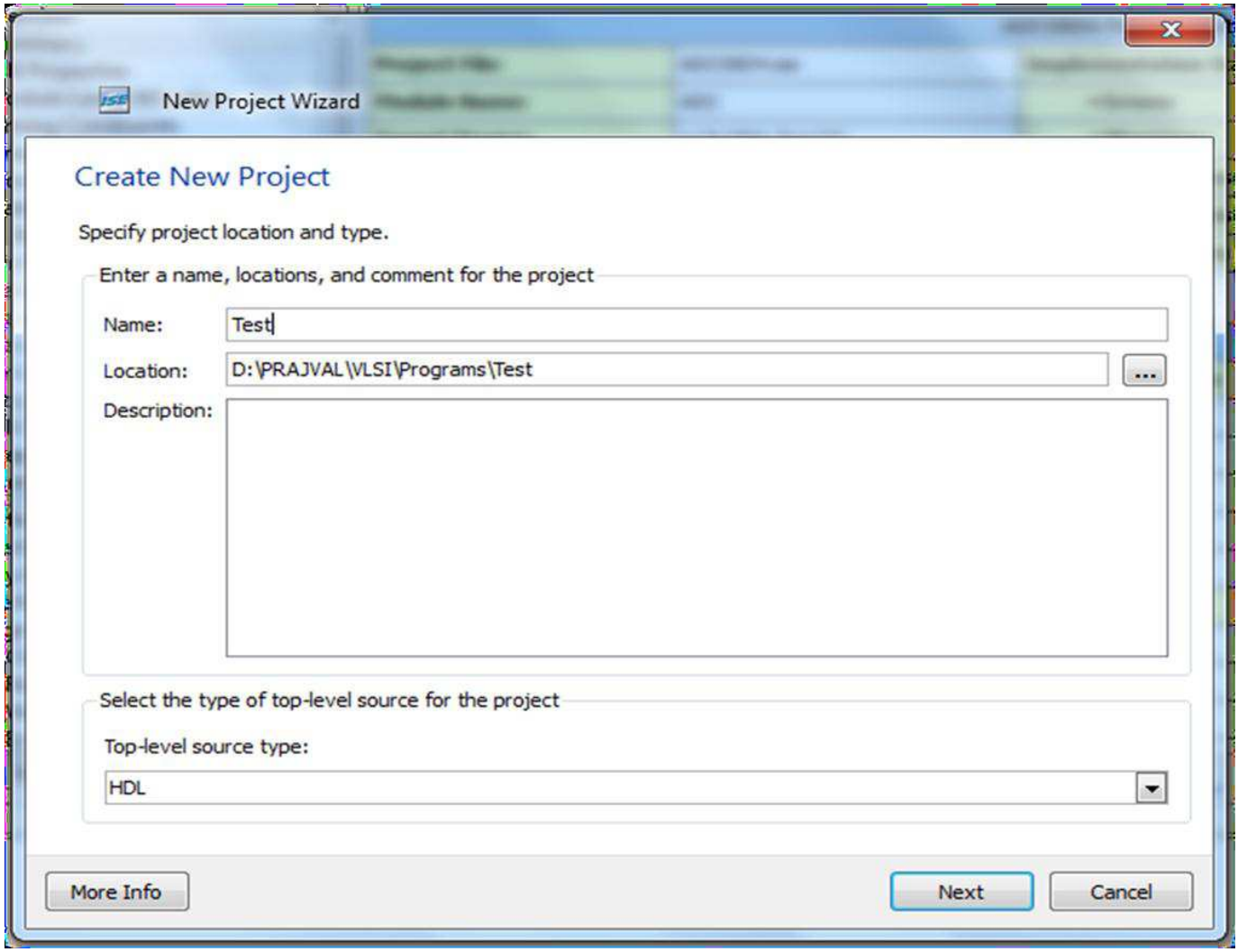
Step 2 : Create new project

---

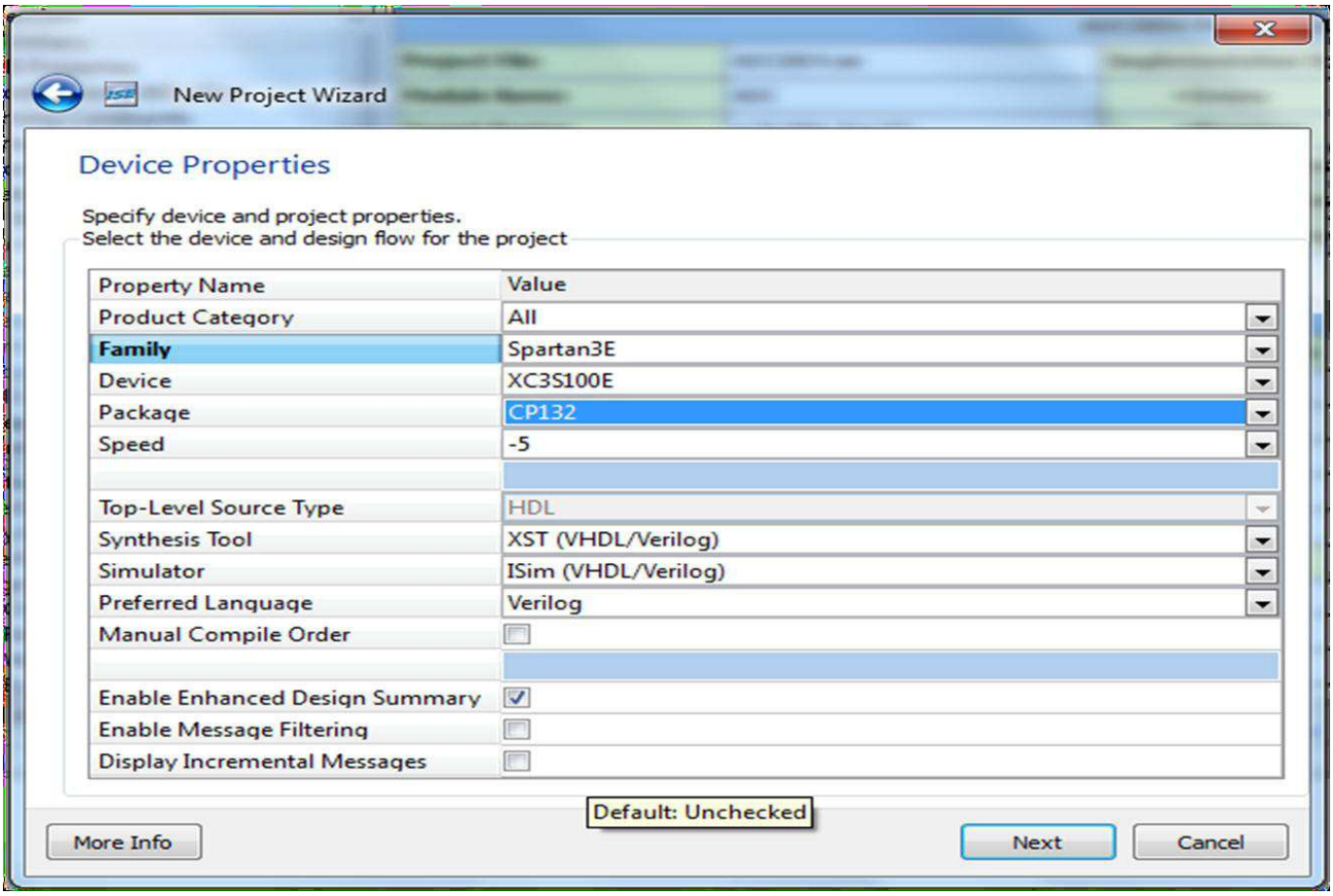
1. To create a new project, open Project Navigator either from the Desktop shortcut icon or by selecting **Start > Programs > Xilinx ISE Design Suite 12.2 > ISE > Project Navigator**.

2. In Project Navigator, select the New Project option from the Getting Started menu (or by selecting Select File > New Project).

3. This brings up a Dialog box where you can enter the desired project name and project location. You should choose a meaningful name for easy reference. In this tutorial, we call this project “Test” and save it in a local directory. You can place comments for your project in the Description text box. We use HDL for our top-level source type in this tutorial.



4. The next step is to select the proper Family, Device, and Package for your project. This depends on the chip you are targeting for this project. The appropriate settings for a project suited for the BASYS 2 board are as follows:

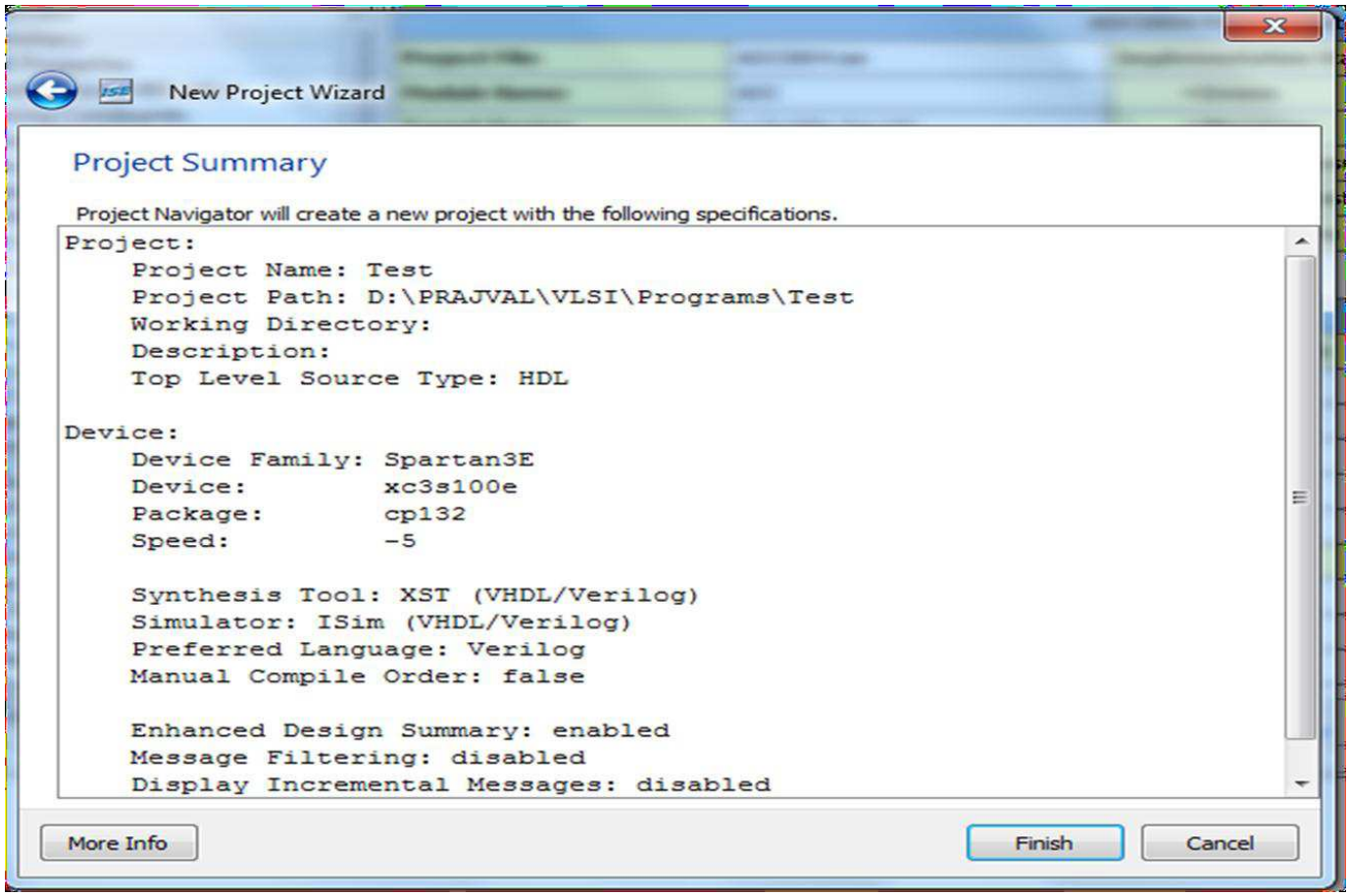


Once the appropriate settings have been entered, click next. Above device properties are for CPLD.
For FPGA, select

1) **Family : Spartan 3E**
2) **Device : XC3S500E**
3) **Package : PQ208**
4) **Speed : -4**
5) **Preferred Language : VHDL**

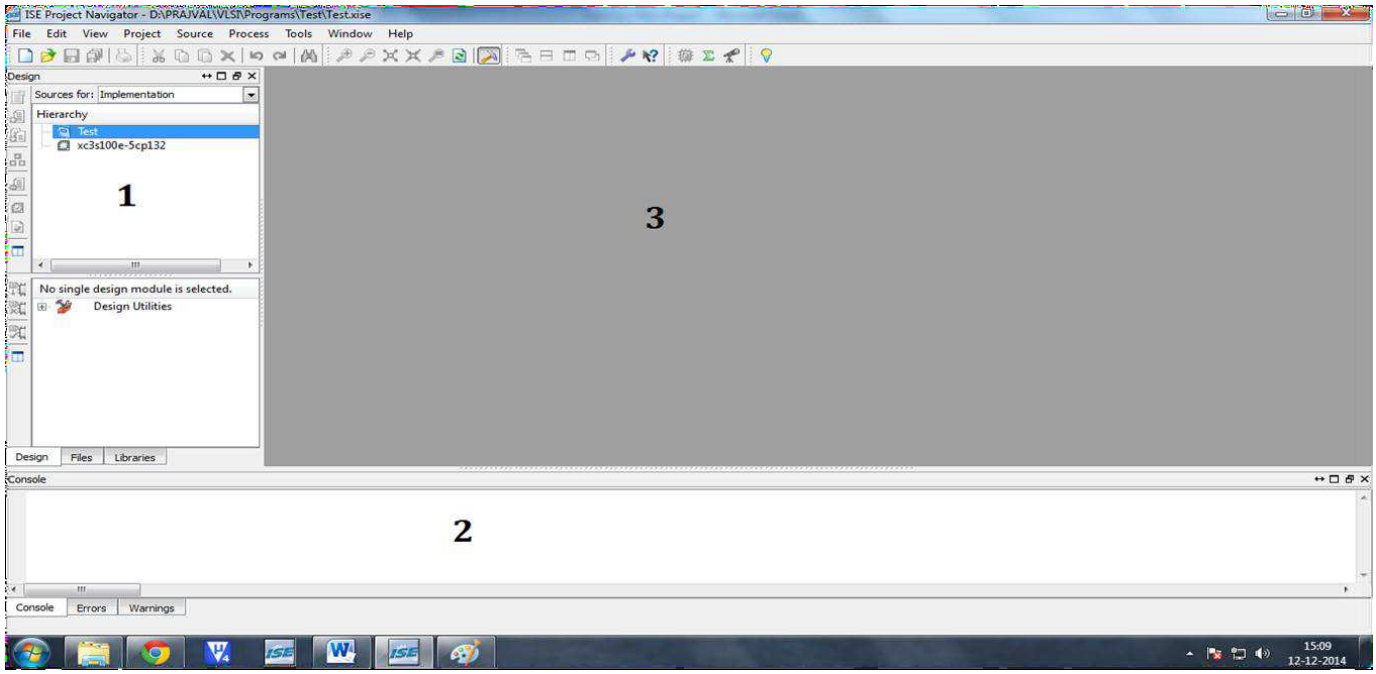
5. The next two dialog boxes give you the option of adding new or existing source files to your project. Since we will fulfill these steps later, click next without adding any source files.

6. Before the new project is created, the New Project Wizard gives you a project summary consisting of the selected specifications you have chosen for the project. Make sure all settings are correct before clicking Finish to end the New Project Wizard.



b) Project Navigator Overview

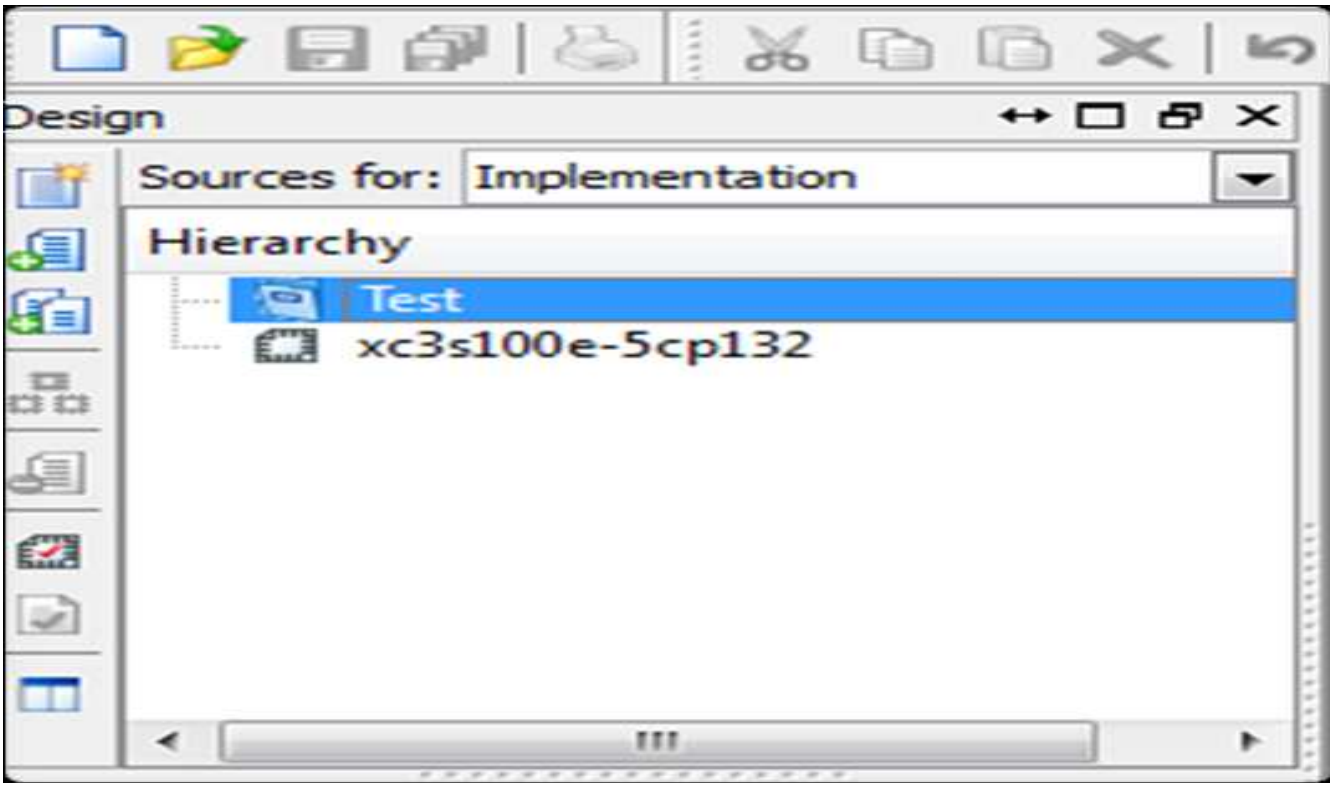
Once the new project has been created, ISE opens the project in Project Navigator. Click the Design tab to show the Design panel and click the Console tab to show the Console panel.



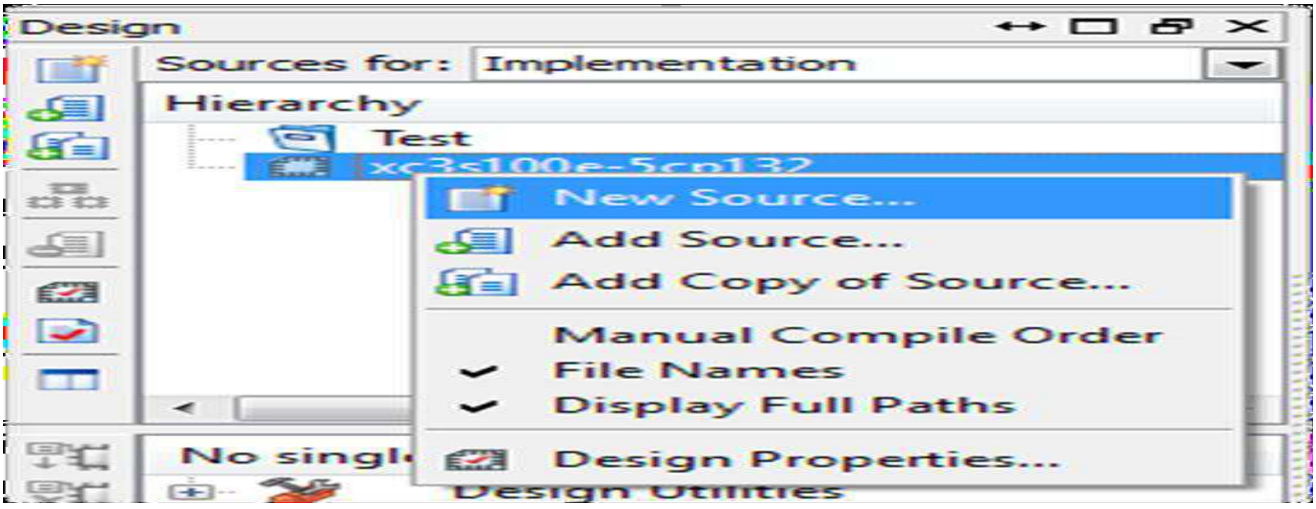
The Design panel (1) contains two windows: a Sources window that displays all source files associated with the current design and a Process window that displays all available processes that can be run on a selected source file.
The Console panel (2) displays status messages including error and warning messages.
The HDL editor window (3) displays source code from files selected in the Design panel.

c) Adding New Source Files

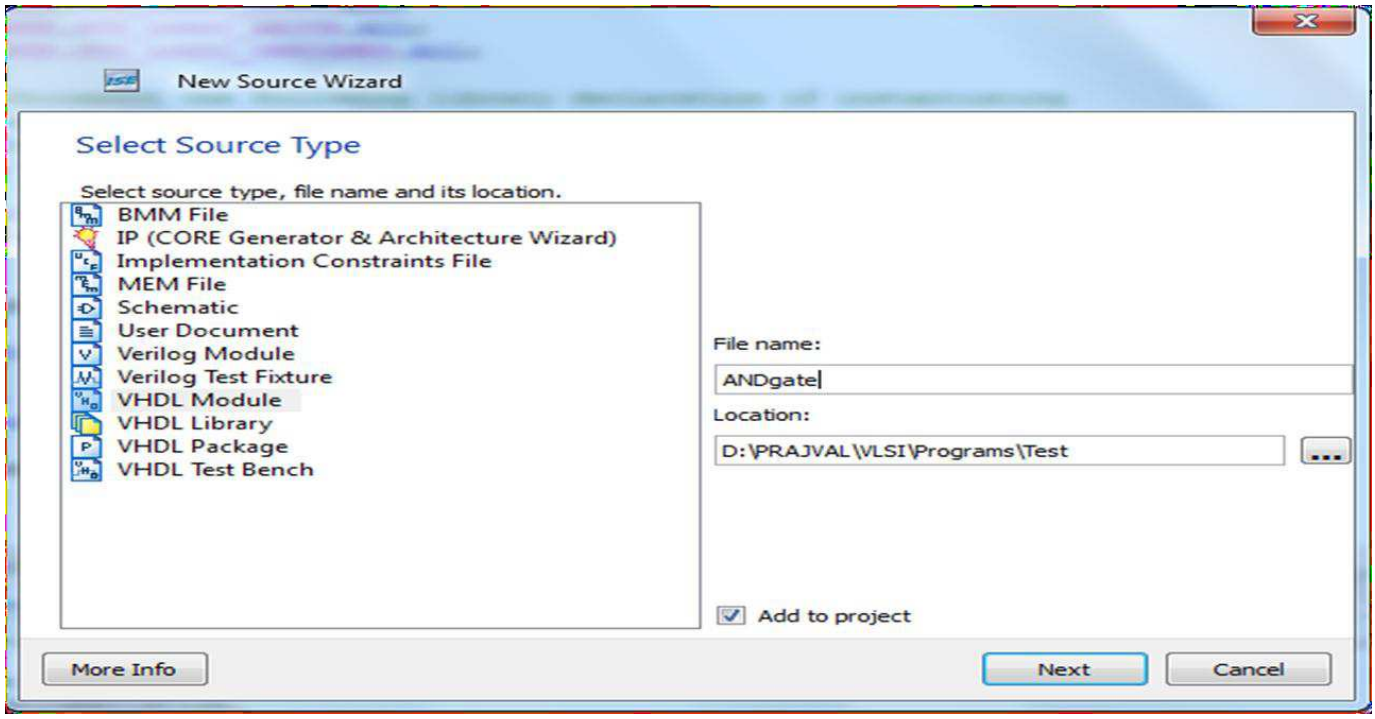
1. Once the new project is created, two sources are listed under sources in the Design panel: the Project file name and the Device targeted for design.



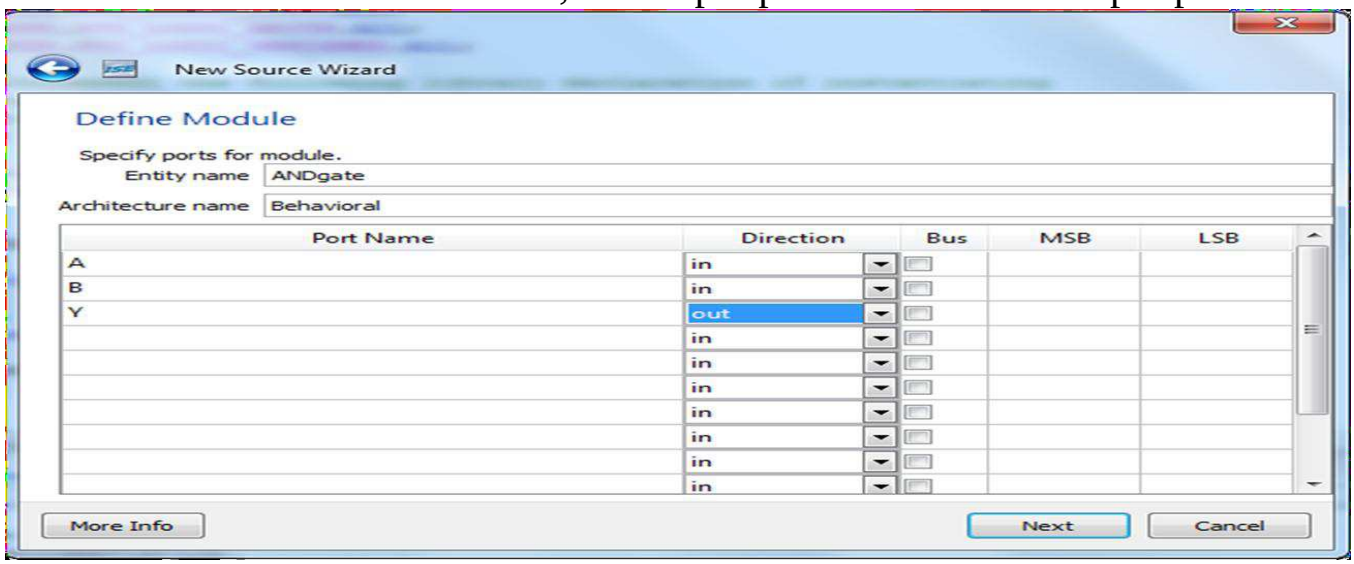
2. You can add a new or existing source file to the project. To do this, right-click on the target device and select one of the three options for adding source files.



3. In this tutorial, we will create a new source file, so select New Source from the list. This starts the New Source Wizard, which prompts you for the Source type and file name. Select VHDL module and give it a meaningful name (we name it ANDgate).



4. When you click next you have the option of defining top-level ports for the new VHDL module. We chose A, B as input ports and Y as an output port.



Click next and then finish completing the VHDL source file creation.

---

d) HDL Editor Window

Once you have created the new VHDL file, the HDL Editor Window displays the ANDgate.vhd source code.

```
18  --
19  ----------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.STD_LOGIC_ARITH.ALL;
23  use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25  ---- Uncomment the following library declaration
26  ---- any Xilinx primitives in this code.
27  --library UNISIM;
28  --use UNISIM.VComponents.all;
29
30  entity ANDgate is
31      Port ( A : in  STD_LOGIC;
32             B : in  STD_LOGIC;
33             Y : out  STD_LOGIC);
34  end ANDgate;
35
36  architecture Behavioral of ANDgate is
37
38  begin
39
40
41  end Behavioral;
```

ISE automatically generate lines of code in the file to get you started with circuit development. This generated code includes:

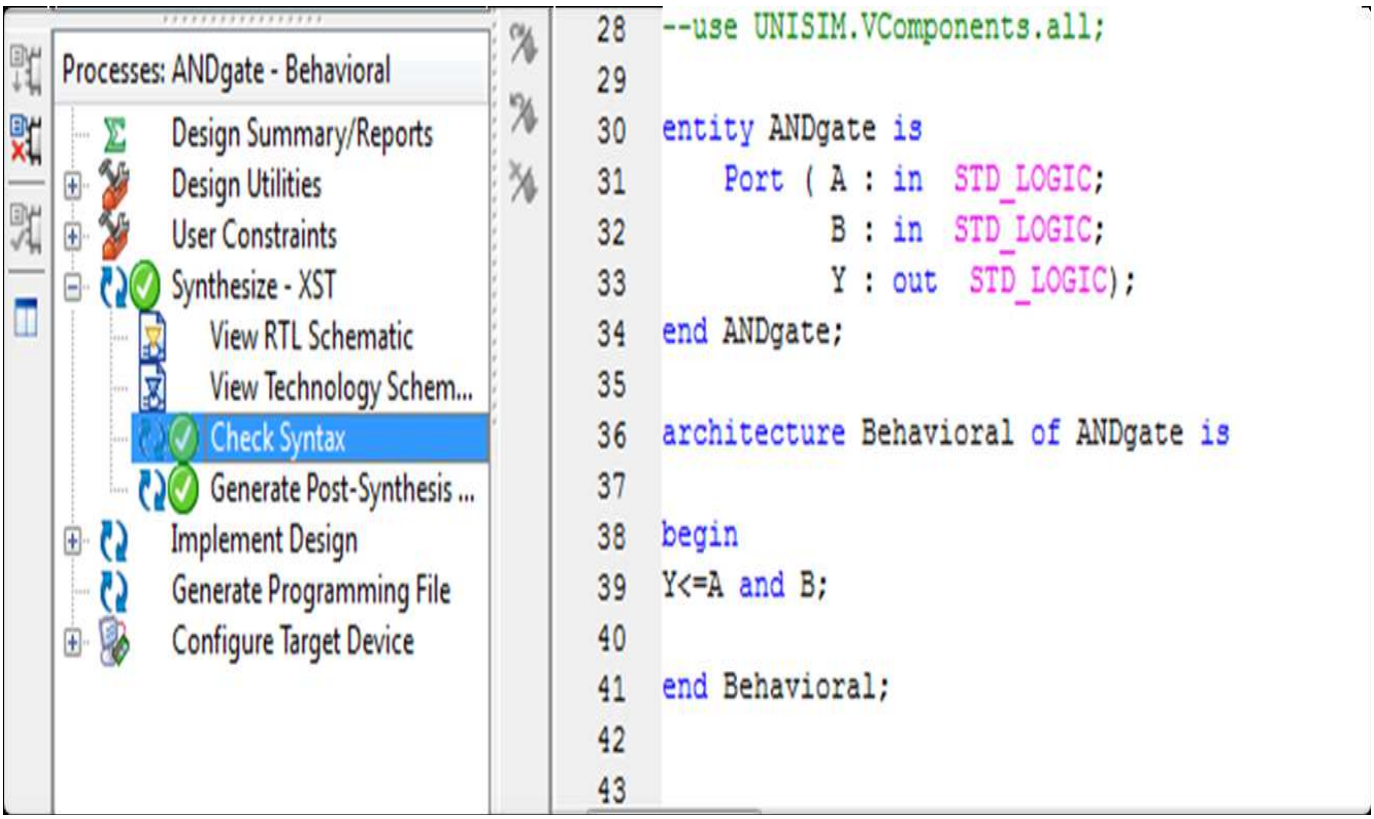
• library definitions

• An entity statement

• An architecture statement with begin and end statements included

• A comment block template for documentation.

The actual behavioral or structural description of the given circuit is to be placed between the “begin” and “end Behavioral” statements in the file. Between these statements, you can define any VHDL circuit you wish. In this tutorial we use a simple combinational logic example, and then show how it can be used as a structural component in another VHDL module. We start with the basic logic equation: Y<= (A·B).
The following figure shows the implementation & synthesis:

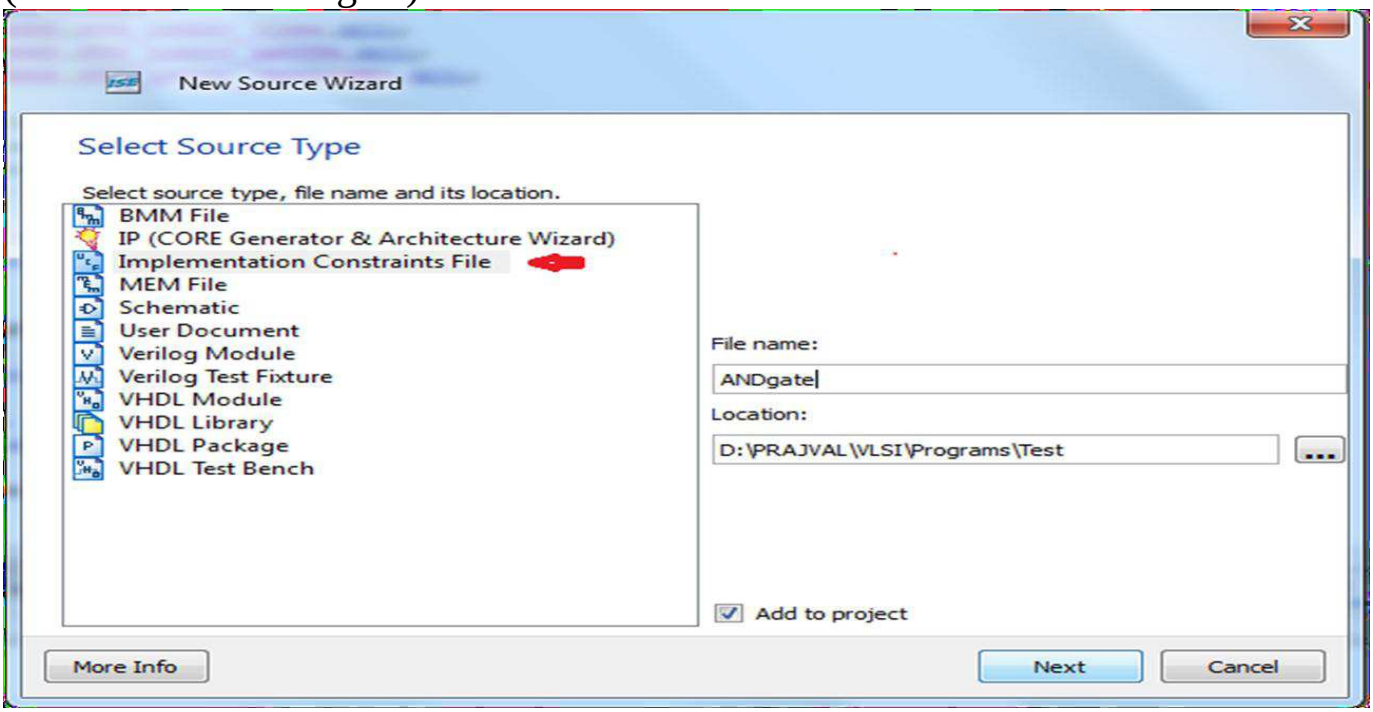


If there are any syntax errors in the source file, ISE can help you find them. For example, the parentheses around the A and B inputs are crucial to this implementation; without them, ISE would return an error during synthesis.

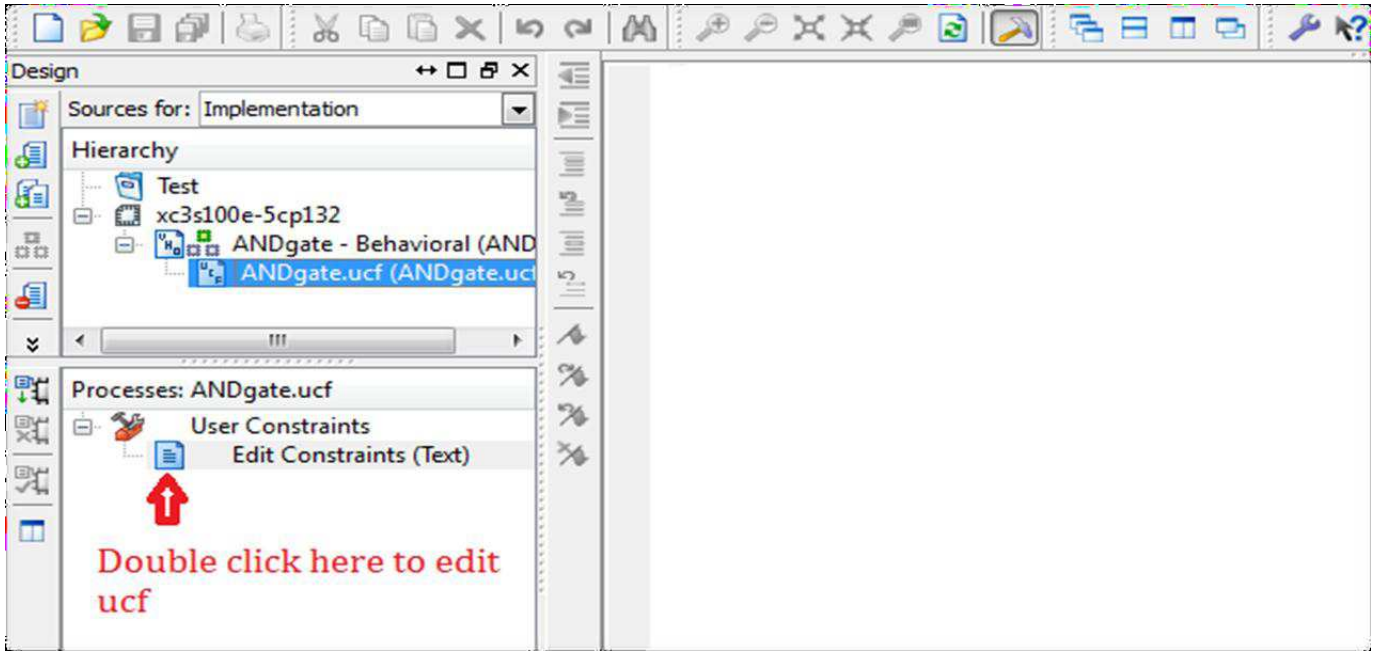
e) UCF File Creation

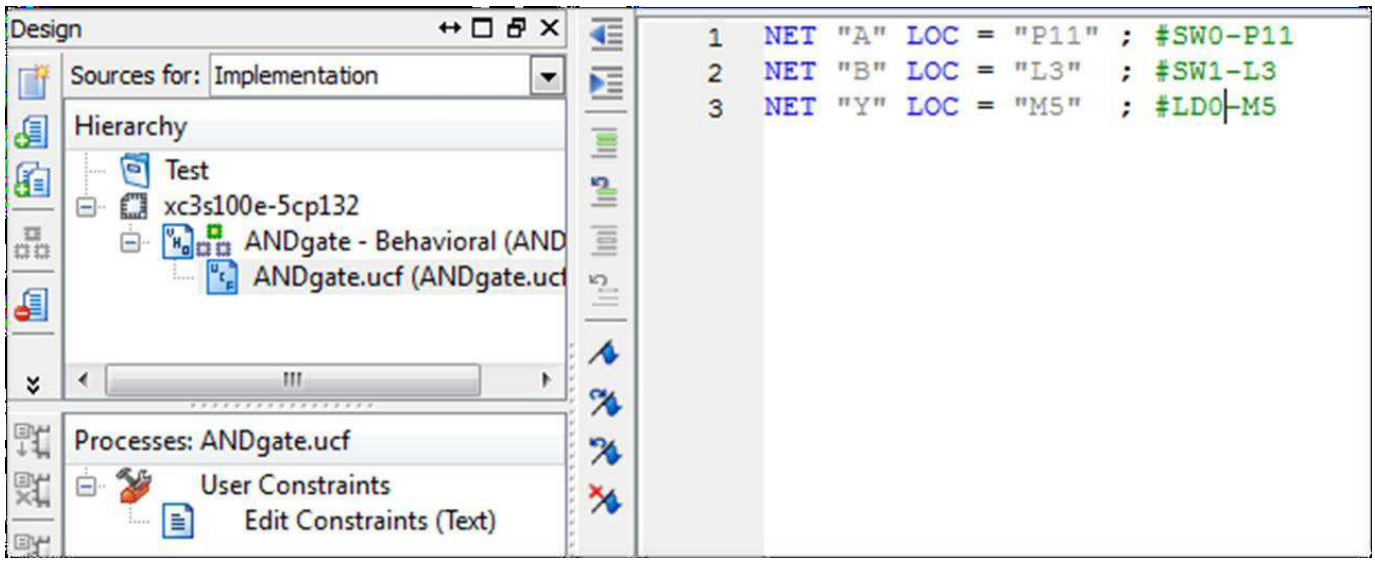
The Xilinx tools use a User Constraints File (.ucf file) to define user constraints like physical pin to circuit net mappings. This is sometimes referred to as an Implementation Constraints File. The .ucf file can be modified inside ISE using a text editor.

To add a .ucf file to your design, go to the Sources window and right-click the source file that requires user constraints. Select the Add New Source option in the drop-down menu. The New Source Wizard prompts you for the Source type and file name. Select Implementation Constraints File and give it a meaningful name (we'll name it ANDgate).



To edit the .ucf file, select it in the sources window, expand the User Constraints option in the Processes window below, and double-click the Edit Constraints (Text) option. A blank text editor appears.



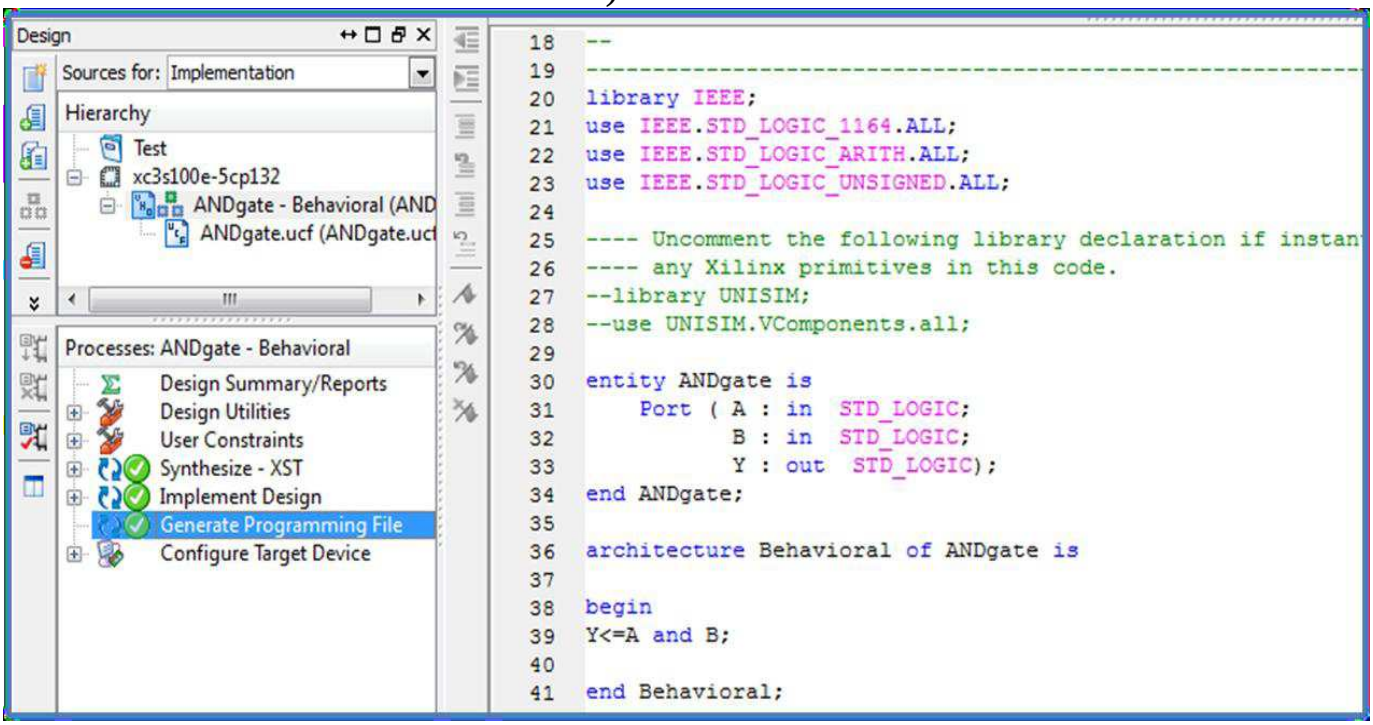


To associate a physical pin with a given net name, type: NET “net name” LOC =”XXX”; on a line in the .ucf file. In the statement, “net name” (quotes included) is the name of the net to attach to pin number XXX (quotes included).

For our example project, the two inputs are assigned to switches 0 through 1 and the output is assigned to LD0 on the BASYS2 board. The finished .ucf file is as follows:

f) Programming File Generation

Now we are ready to create a programming file (.bit) for the BASYS2 FPGA.
Go to the Sources window and select the top-level module (indicated by the three blocks shown with the source name.)



Now go to the Processes window where there are three particular processes in a row:

1. Synthesize – XST

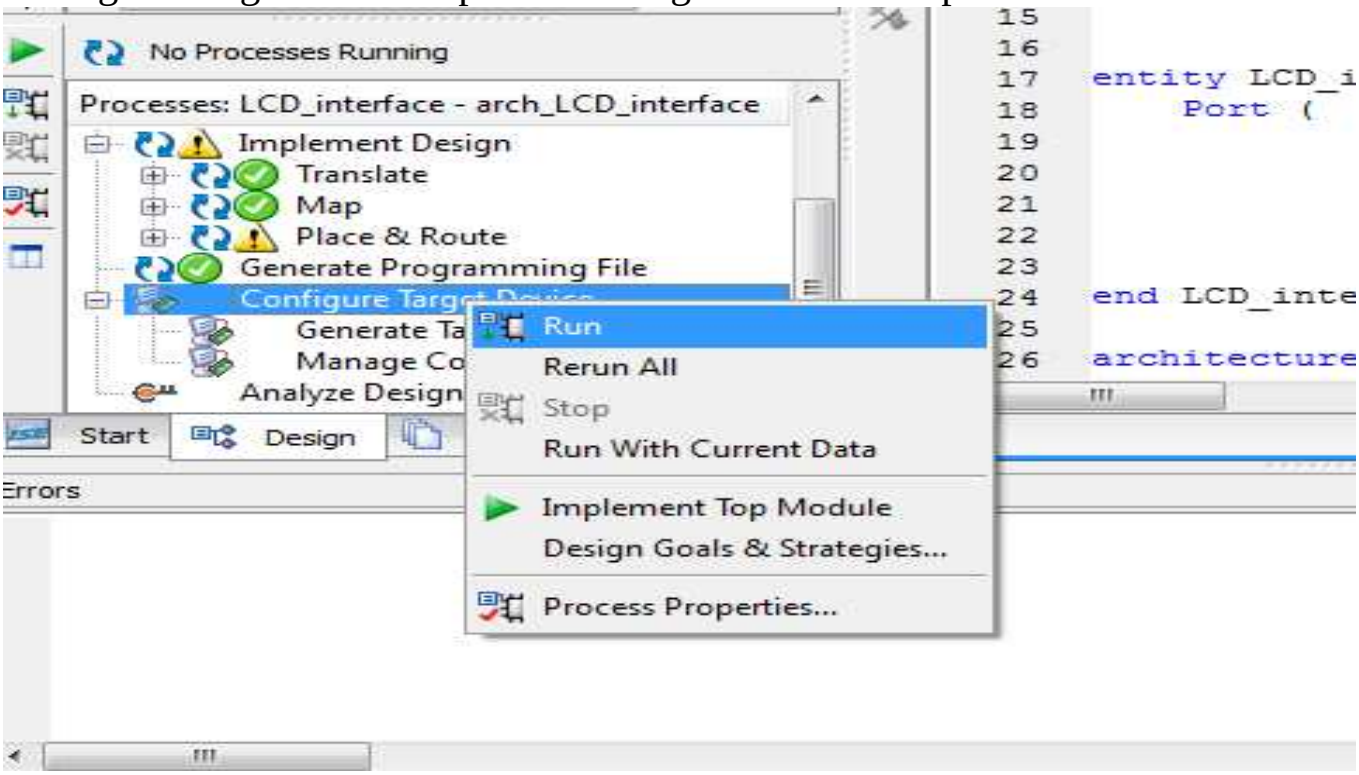
2. Implement Design

3. Generate Programming file

Run the synthesis process by both double-clicking on Synthesize or left clicking and selecting the run option. This process analyzes the circuit you have created, checking for valid connections, syntax, and structure, to verify that the circuit is valid and synthesizable.
If the Synthesize process does not return any errors, you can move on and run the Implement Design process. This process uses various algorithms to map out the digital circuit and then creates place and route information so that it can be placed on the physical FPGA.

g) Programming Board

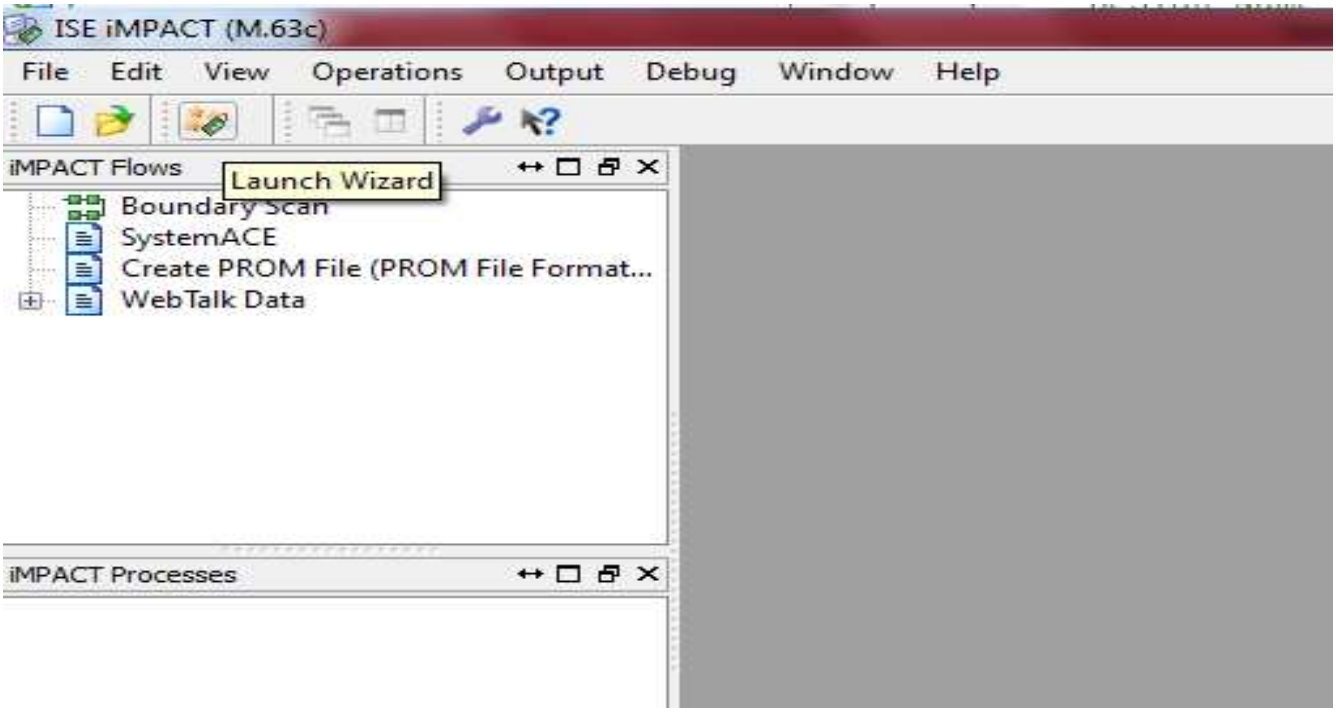
1. After generating programming file double click or Right click and Run on Configure Target Device Option warning window will open click on OK.



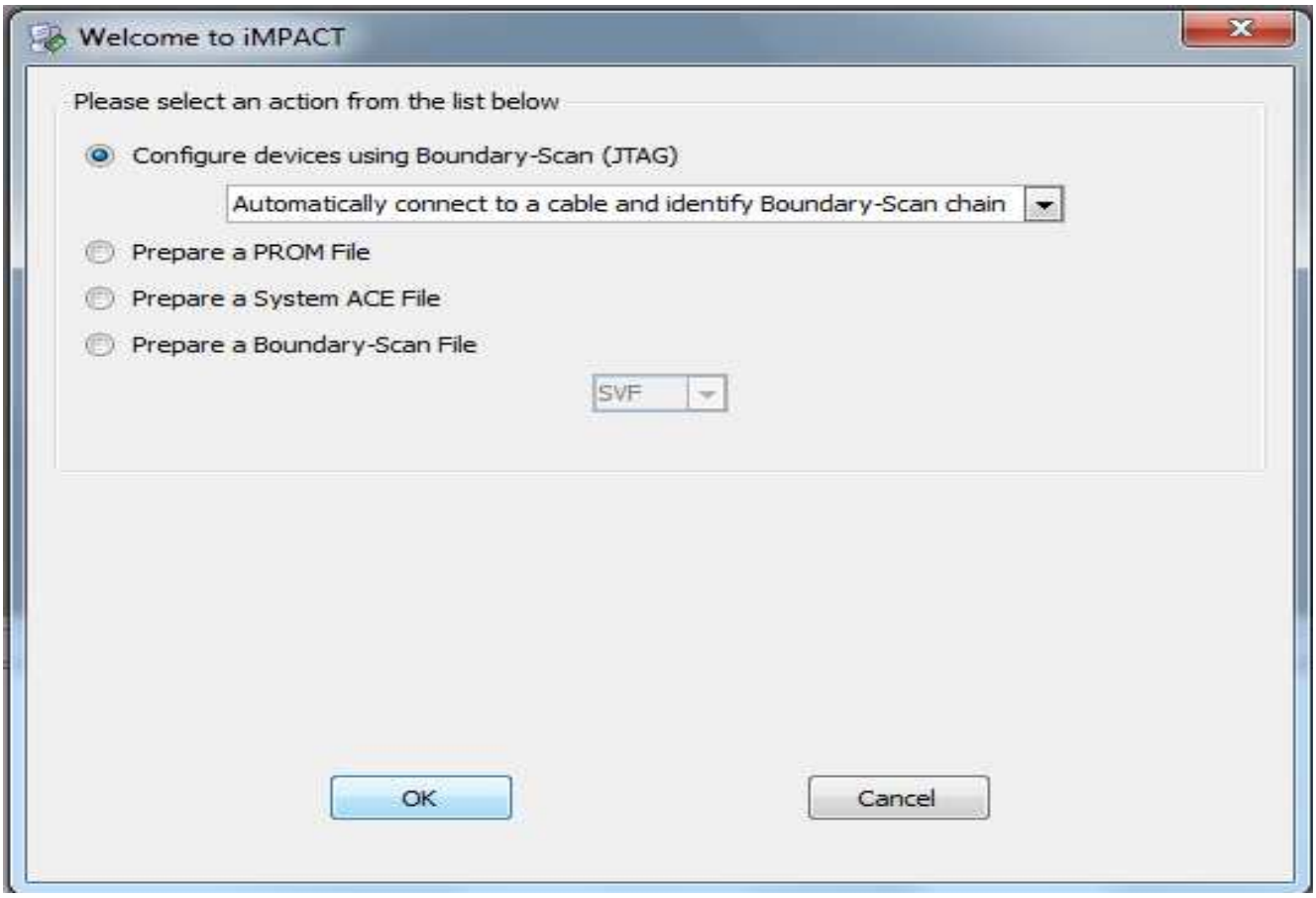
2. After this step ISE iMPAC (M.63c) window will open.

3. Now make connection between your PC and FPGA board via Xilinx Platform Cable USB by connecting 10 pin FRC cable to JTAG Connector of FPGA board.

4. Click on Launch Wizard option.



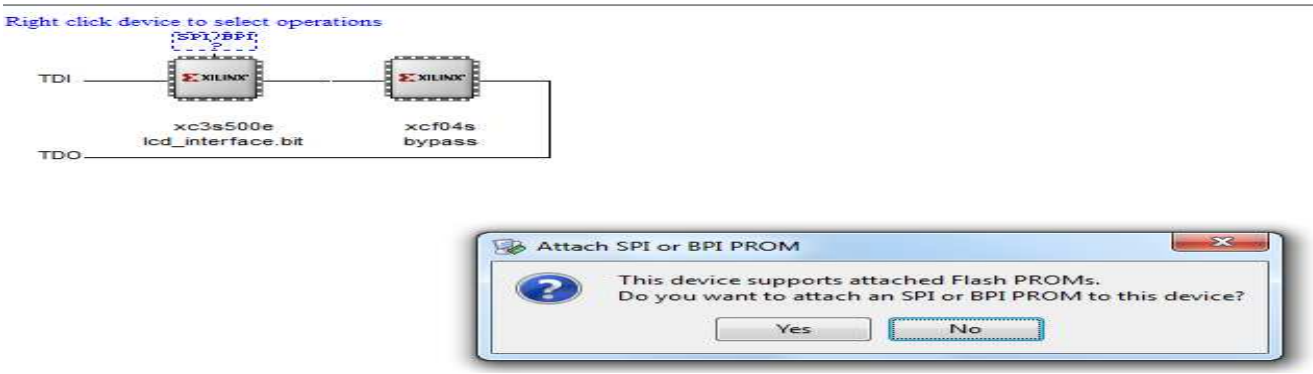
5. After click welcome window will open click OK.



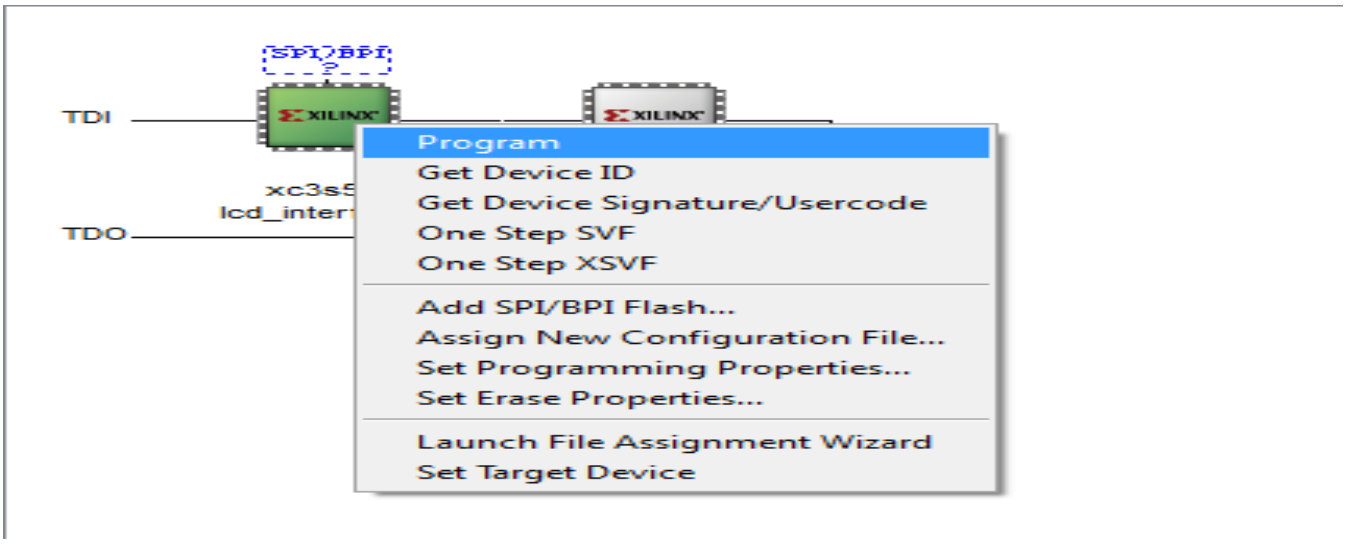
6. If everything ok ‘Identify Succeeded’ will display. At same time two “Assign New Configuration File” window will open one by one in this window click on Cancel after this “window Device Programming Properties” window will display click on cancel.

7. Now double click on device and Browse for .bit file which you have to program.

8. After opening your .bit file one warning window will open click on **NO.**



9. Now Right click on device and select program option after clicking new window will open click OK. If everything ok ‘Program Succeeded’ will display**.**



Chapter 6: Connector Configuration

6. A Keys used for digital logic section

Starting I/P & O/P keys from pulser key side
Upper side Keys used for select lines
Lower side keys used for I/P

INPUT		OUTPUT	
Switch No.	**FPGA Pin No.**	**LED No.**	**FPGA Pin No.**
SW0	**P31**	**LED0**	**P30**
SW1	**P25**	**LED1**	**P24**
SW2	**P19**	**LED2**	**P18**
SW3	**P12**	**LED3**	**P11**
SW4	**P5**	**LED4**	**P4**
SW5	**P205**	**LED5**	**P203**
SW6	**P199**	**LED6**	**P190**
SW7	**P192**	**LED7**	**P163**
SW8	**P29**	**LED8**	**P28**
SW9	**P23**	**LED9**	**P22**
SW10	**P16**	**LED10**	**P15**
SW11	**P9**	**LED11**	**P8**
SW12	**P3**	**LED12**	**P2**
SW13	**P202**	**LED13**	**P200**
SW14	**P196**	**LED14**	**P197**
SW15	**P189**	**LED15**	**P187**

Sr. no.	Experiment Name	Pins used as I/P	Pins used as O/P
1	**AND gate**	**A=SW0** **B=SW1**	**Y=LI16**
2	**Full Adder**	**A=SW0** **B=SW1** **C=SW2**	**SUM=LI16** **CARRY=LI15**
3	**Full Subtractor**	**A=SW0** **B=SW1** **C=SW2**	**SUB=LI16** **BOR=LI15**
4	**8-to-1 multiplexer**	**I(0)=SW0** **I(1)=SW1** **I(2)=SW2**	**Q=LI8** **Select lines** **SEL(0)=SW8**

		I(3)=SW3 **I(4)=SW4** **I(5)=SW5** **I(6)=SW6** **I(7)=SW7**	**SEL(1)=SW9** **SEL(2)=SW10**
5	**2-to-4 decoder**	**I(0)=SW0** **I(1)=SW1** **EN=SW15**	**D(0)=LI16** **D(1)=LI15** **D(2)=LI14** **D(3)=LI13**
6	**8-to-3 encoder**	**I(0)=SW0** **I(1)=SW1** **I(2)=SW2** **I(3)=SW3** **I(4)=SW4** **I(5)=SW5** **I(6)=SW6** **I(7)=SW7** **EN=SW15**	**Y(0)=LI16** **Y(1)=LI15** **Y(2)=LI14**
7	**Binary to gray converter**	**I(0)=SW0** **I(1)=SW1** **I(2)=SW2** **I(3)=SW3**	**G(0)=LI16** **G(1)=LI15** **G(2)=LI14** **G(3)=LI13**
8	**Sequence generator**	**Press pulser key OR RST** **CLR=SW4**	**Q=LI16**
9	**Synchronous counter**	**CLK=RESET** **RST=SW1**	**Y(0)=LI16** **Y(1)=LI15** **Y(2)=LI14** **Y(3)=LI13**
10	**Gray to binary converter**	**I(0)=SW0** **I(1)=SW1** **I(2)=SW2** **I(3)=SW3**	**B(0)=LI16** **B(1)=LI15** **B(2)=LI14** **B(3)=LI13**
11	**Carry-Loop-ahead adder**	**A(0)=SW0** **A(1)=SW1** **A(2)=SW2** **A(3)=SW3** **B(0)=SW4** **B(1)=SW5** **B(2)=SW6** **B(3)=SW7** **Cin=SW8**	**Cout=LI3** **Select lines** **S(0)=LI8** **S(1)=LI7** **S(2)=LI6** **S(3)=LI5**
12	**Comparator**	**NUM(0)=SW0** **NUM(1)=SW1** **NUM(2)=SW2** **NUM(3)=SW3** **NUM(0)=SW4** **NUM(1)=SW5**	**Less=LI16** **Equal=LI15** **Greater=LI14**

		NUM(2)=SW6 **NUM(3)=SW7**	
13	**Shift register** **1)PISO** **2)PIPO** **3)SIPO** **4)SISO**	**1)PISO** **PI(7)=SW0** **PI(6)=SW1** **PI(5)=SW2** **PI(4)=SW3** **PI(3)=SW4** **PI(2)=SW5** **PI(1)=SW6** **PI(0)=SW7** **2) PIPO** **PI(0)=SW0** **PI(1)=SW1** **PI(2)=SW2** **PI(3)=SW3** **PI(4)=SW4** **PI(5)=SW5** **PI(6)=SW6** **PI(7)=SW7** **3)SIPO** **Clk=RESET** **RST=SW8** **SI=SW0** **4)SISO** **SI=SW0**	**1)PISO** **Clk=RESET** **Reset=SW8** **Load=SW9** **SO=LI16** **2) PIPO** **PO(0)=LI16** **PO(1)=LI15** **PO(2)=LI14** **PO(3)=LI13** **PO(4)=LI12** **PO(5)=LI11** **PO(6)=LI10** **PO(7)=LI9** **3) SIPO** **PO(0)=LI16** **PO(1)=LI15** **PO(2)=LI14** **PO(3)=LI13** **PO(4)=LI12** **PO(5)=LI11** **PO(6)=LI10** **PO(7)=LI9** **4)SISO** **SO=LI16**
14	**Universal shift**	**Clr-SW0** **M0-SW1** **M1-SW2** **Dsr-SW3** **Dsl-SW4** **Din(0)=SW8** **Din(1)=SW9** **Din(2)=SW10** **Din(3)=SW11**	**Q(0)=LI8** **Q(1)=LI7** **Q(2)=LI6** **Q(3)=LI5**
15	**1-to-8 demultiplexer**	**Select line** **S(0)=SW8** **S(1)=SW9** **S(2)=SW10**	**D(0)=LI8** **D(1)=LI7** **D(2)=LI6** **D(3)=LI5** **D(4)=LI4** **D(5)=LI3** **D(6)=LI2** **D(7)=LI1**

16	**ALU_4 bit**	**A(0)=SW0** **A(1)=SW1** **A(2)=SW2** **A(3)=SW3** **B(0)=SW4** **B(1)=SW5** **B(2)=SW6** **B(3)=SW7** **Select lines** **S(0)=SW8** **S(1)=SW9** **S(2)=SW10**	**Y(0)=LI8** **Y(1)=LI7** **Y(2)=LI6** **Y(3)=LI5**
17	**Flip-Flop** **1)S-R** **2)J-K** **3)T** **4)D**	**CLK=P1O1(Reset)** **RST=SW0** **1)S-R FF** **S=SW8** **R=SW9** **2) J-K FF** **J=SW9** **K=SW8** **3)T FF** **T=SW8** **4)D FF** **Din=SW8**	**1)S-R FF** **Q=LI8** **Qbar=LI7** **P16** **2) J-K FF** **Q=LI8** **Qbar=LI7** **3)T FF** **Q=LI8** **4)D FF** **Q=LI8**

6. B Connector details and jumper settings of Spartan-3(FPGA) development board:-
For Xilinx Devices

1) DC motor:- Switch on S25and 1&2 short i.e JP6,1&2 short i.e JP7.

2) Stepper motor:- On switch S25and 2&3 short i.e JP6,2&3 short i.e JP7.

3) ADC:- 1&2 short i.e SL1 and switch on S30.

4) DAC:- Switch on S29 and connect CRO/DSO Probe to DAC_O/P and 0V.

Chapter 7: Experiments

7. A for FPGA

Experiment No. 1

Aim : - Interfacing of FPGA XC 3S500E to LCD.

Requirement: - FPGA XC 3S500E board, 12v adaptor, Platform cable

Procedure:-
1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA Xc 3S500E module of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of LCD section **(S27)** and see the O/P on display.

Experiment No. 2

Aim : - Interfacing of FPGA XC 3S500E to 7 Segment with 4*4 Keypad.

Requirement: - FPGA XC 3S500E board, 12v adaptor, platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA Xc 3S500E module of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of 7 segment section **(S28)** and see the O/P on display.

Experiment No. 3

Aim : - Interfacing of FPGA XC 3S500E to LED sequence.

Requirement: - FPGA XC 3S500E board, 12v adaptor, platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA Xc 3S500E module of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of LED sequence section (**S3**).
6) Press the switch S20 and S21 and see the corresponding LED sequence on LED's.

Experiment No. 4

Aim : - Interfacing of FPGA XC 3S500E to DC motor.

Requirement: - FPGA XC 3S500E board, 12v adaptor, platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA Xc 3S500E module of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5)Connect DC Motor at RL3.
6) Make the switch ON of DC motor section (**S25**) .

Experiment No. 5

Aim : - Interfacing of FPGA XC 3S500E to Stepper motor.

Requirement: - FPGA XC 3S500E board, 12v adaptor, platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA Xc 3S500E module of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5)Connect Stepper Motor at RL4.
6) Make the switch ON of Stepper motor section (**S25**) .

Experiment No. 6

Aim : - Interfacing of FPGA XC 3S500E to ADC0804.

Requirement: - FPGA XC 3S500E board, 12v adaptor, platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA Xc 3S500E module of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of 8 Bit ADC section (**S30**).
6)Reset the chip and vary the Pot R80 and see the output on LED's.

7. B HDL code to realize all the logic gates

Experiment No. 1

Aim : - Interfacing of FPGA XC 3S500E to design and simulation of adder, Carry Look Ahead Adder.
Requirement: - FPGA XC 3S500E board, 12v adaptor, Platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA XC 3S500E board of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of Digital logic section (**S6**) and use I/P switch as mentioned above, now see the O/P.

Experiment no. 2

Aim : - Interfacing of FPGA XC 3S500E to design 2-to-4 decoder.

Requirement: - FPGA XC 3S500E board, 12v adaptor, Platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA XC 3S500E and the board.
2) Connect platform cable to FPGA XC 3S500E board of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of Digital logic section (**S6**) and use I/P switch as mentioned above, now see the O/P.

Experiment No. 3

Aim : - Interfacing of FPGA XC 3S500E to design 8-to-3 encoder (with and without parity).

Requirement: - FPGA XC 3S500E board, 12v adaptor, Platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA XC 3S500E and the board.
2) Connect platform cable to FPGA XC 3S500E board of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of Digital logic section (**S6**) and use I/P switch as mentioned above, now see the O/P.

Experiment No. 4

Aim : - Interfacing of FPGA XC 3S500E to design 8-to-1 multiplexer.

Requirement: - FPGA XC 3S500E board, 12v adaptor, Platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA XC 3S500E and the board.
2) Connect platform cable to FPGA XC 3S500E board of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of Digital logic section (**S6**) and use I/P switch as mentioned above, now see the O/P.

Experiment No. 5

Aim : - Interfacing of FPGA XC 3S500E to design 4-bit Binary to Gray converter.

Requirement: - FPGA XC 3S500E board, 12v adaptor, Platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA XC 3S500E and the board.
2) Connect platform cable to FPGA XC 3S500E board of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of Digital logic section (**S6**) and use I/P switch as mentioned above, now see the O/P.

Experiment No. 6

Aim : - Interfacing of FPGA XC 3S500E to design Multiplexer/Demultiplexer and comparator.

Requirement: - FPGA XC 3S500E board, 12v adaptor, Platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA XC 3S500E and the board.
2) Connect platform cable to FPGA XC 3S500E board of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of Digital logic section (**S6**) and use I/P switch as mentioned above, now see the O/P.

Experiment No. 7

Aim : - Interfacing of FPGA XC 3S500E to design Full adder using 1 modeling style.

Requirement: - FPGA XC 3S500E board, 12v adaptor, Platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA XC 3S500E and the board.
2) Connect platform cable to FPGA XC 3S500E board of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of Digital logic section (**S6**) and use I/P switch as mentioned above, now see the O/P.

Experiment No. 8

Aim : - Interfacing of FPGA XC 3S500E to design S-R, J-K, T, D flip-flop.

Requirement: - FPGA XC 3S500E board, 12v adaptor, Platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA XC 3S500E and the board.
2) Connect platform cable to FPGA XC 3S500E board of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of Digital logic section (**S6**) and use I/P switch as mentioned above, now see the O/P.

Experiment No. 9

Aim : - Interfacing of FPGA XC 3S500E to design 4-bit Binary counter.

Requirement: - FPGA XC 3S500E board, 12v adaptor, Platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA XC 3S500E and the board.
2) Connect platform cable to FPGA XC 3S500E board of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of Digital logic section (**S6**) and use I/P switch as mentioned above, now see the O/P.

Experiment No. 10

Aim : - Interfacing of FPGA XC 3S500E to design N-bit register of SISO,SIPO,PISO,PIPO.

Requirement: - FPGA XC 3S500E board, 12v adaptor, Platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA XC 3S500E and the board.
2) Connect platform cable to FPGA XC 3S500E board of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of Digital logic section (**S6**) and use I/P switch as mentioned above, now see the O/P.

Experiment No. 11

Aim : - Interfacing of FPGA XC 3S500E to design Sequence generator.

Requirement: - FPGA XC 3S500E board, 12v adaptor, Platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA XC 3S500E and the board.
2) Connect platform cable to FPGA XC 3S500E board of which JTAG ident is written.
3) Open the Xilinx software and make setting as per given.
4) Download the .bit file for FPGA XC3S500E Board.
5) Make the switch ON of Digital logic section (**S6**) and use I/P switch as mentioned above, now see the O/P.

Chapter 8: Pin configuration

1. DIGITAL LOGIC:

PIN NUMBER	STANDARD NAME	FUNCTION NAME
P193	IO_L12N_0	PULSER KEY
P163	IO_L02N_0	LI9
P190	IO_L11N_0	LI10
P203	IO_L15N_0	LI11
P4	O_L02P_3	LI12
P11	IO_L04P_3	LI13
P18	IO_L06P_3	LI14
P24	IO_L08P_3	LI15
P30	IO_L10P_3	LI16
P187	IO	LI1
P197	IO_L13N_0	LI2
P200	IO_L14N_0	LI3
P2	IO_L01P_3	LI4
P8	IO_L03P_3	LI5
P15	IO_L05P_3	LI6
P22	IO_L07P_3	LI7
P28	IO_L09P_3	LI8
P192	IO_L12P_0	SW7
P199	IO_L14P_0	SW6
P205	IO_L16P_0	SW5
P5	IO_L02N_3	SW4
P12	IO_L04N_3	SW3
P19	IO_L06N_3	SW2
P31	IO_L10N_3	SW0
P189	IO_L11P_0	SW15
P196	IO_L13P_0	SW14
P202	IO_L15P_0	SW13
P3	IO_L01N_3	SW12
P9	IO_L03N_3	SW11
P16	IO_L05N_3	SW10
P23	IO_L07N_3	SW9
P29	IO_L09N_3	SW8

P25	IO_L08N_3	SW1

2. LED SEQUENCE

PIN NUMBER	PIN NAME	FUNCTION NAME
P185	IO_L10P_0	RELAY
P186	IO_L10N_0	BUZZER
P177	IO_L07P_0	S20
P172	IO_L05N_0	S21
P178	IO_L07N_0	LS1
P179	IO/VREF_0	LS2
P181	IO_L08N_0	LS3
P180	IO_L08P_0	LS4

3. DC& STEPPER MOTOR

PIN NUMBER	PIN NAME	SIGNAL NAME	FUNCTION
P39	IO_L13P_3	EN1	FOR **I/P(1,2)**
P42	IO_L14N_3	I/P1	
P41	IO_L14P_3	I/P2	
P47	IO_L15P_3	I/P4	
P45	IO/VREF_3	I/P3	
P49	IO_L16P_3	EN2	FOR **I/P(1,2)**
P33	IO_L11P_3	S22	START
P34	IO_L11N_3	S23	REV
P35	IO_L12P_3	S24	INC
P36	IO_L12N_3	S25	DEC
P40	IO_L13N_3	S26	STOP

4. 8 BIT DAC

PIN NUMBER	PIN NAME	SIGNAL NAME	FUNCTION
P160	IO_L01P_0	A1	DIGITAL INPUT
P153	IO_L16N_1/ LDC2	A2	DIGITAL INPUT
P152	IO_L16P_1/ LDC1	A3	DIGITAL INPUT
P151	IO_L15N_1/ LDC0	A4	DIGITAL INPUT
P150	IO_L15P_1/ HDC	A5	DIGITAL INPUT
P147	IO_L14N_1	A6	DIGITAL INPUT
P146	IO_L14P_1	A7	DIGITAL INPUT
P145	IO_L13N_1	A8	DIGITAL INPUT

5. VGA CONNECTOR

PIN NUMBER	PIN NAME	SIGNAL NAME	FUNCTION NAME
P144	IO_L13P_1	PIN 1	RED VIDEO
P140	IO_L12N_1/A0	PIN 2	GREEN VIDEO
P139	IO_L12P_1	PIN 3	BLUE VIDEO
P138	IO_L11N_1/A1	PIN 13	HORIZONTAL SYNC
P137	IO_L11P_1/A2	PIN 14	VERTICAL SYNC

6. I2C-RTC, EEPROM

PIN NUMBER	PIN NAME	
P55	IO_L01P_2	

P48	IO_L15N_3	

7. 8 BIT ADC

PIN NUMBER	PIN NAME	SIGNAL NAME	FUNCTION
P74	IO_L08P_2/D7	CS	CHIP SELECT
P75	IO_L08N_2/D6	RD	READ
P76	IO/D5	WR	WRITE
P77	IO_L09P_2	INTR	INTER REQUEST
P69	IO_L06N_2	A8	DIGITAL INPUT
P65	IO_L05N_2	A7	DIGITAL INPUT
P64	IO_L05P_2	A6	DIGITAL INPUT
P63	IO_L04N_2	A5	DIGITAL INPUT
P62	IO_L04P_2	A4	DIGITAL INPUT
P61	IO_L03N_2	A3	DIGITAL INPUT
P60	IO_L03P_2	A2	DIGITAL INPUT
P50	IO_L16N_3	A1	DIGITAL INPUT

8. 4x4 KEYPAD

PIN NUMBER	PIN NAME	FUNCTION NAME
P108	IO_L02P_1/A14	COLUMN 1
P109	IO_L02N_1/A13	COLUMN 2
P112	IO_L03P_1	COLUMN 3
P113	IO_L03N_1	COLUMN 4
P115	IO_L04P_1	ROW 1
P116	IO_L04N_1	ROW 2
P119	IO_L05P_1/A12	ROW 3
P120	IO_L05N_1/A11	ROW 4

9. LCD

PIN NUMBER	PIN NAME	FUNCTION NAME
P122	IO_L06P_1	D7
P123	IO_L06N_1	D6
P126	IO_L07P_1/A10	D5
P127	IO_L07N_1/A9	D4
P128	IO_L08P_1/A8	D3
P129	IO_L08N_1/A7	D2
P132	IO_L09P_1/A6	D1
P133	IO_L09N_1/A5	D0
P134	IO_L10P_1/A4	EN
P135	IO_L10N_1/A3	RS

10. SEVEN SEGMENT DISPLAY

PIN NUMBER	PIN NAME	FUNCTION NAME
P94	IO_L14N_2	DISPLAY 1
P93	IO_L14P_2	DISPLAY 2
P90	IO_L13N_2	DISPLAY 3
P82	IP_L11P_2	DISPLAY 4
P97	IO_L15N_2	A
P96	IO_L15P_2	B
P99	IO_L16P_2	C
P98	IO	D
P102	IO_L17P_2	E
P100	IO_L16N_2	F
P107	IO_L01N_1	G
P106	IO_L01P_1	DP

Chapter 9 : Sample Code

User can use sample code to work on FPGA development board, provided along with the kit in CD.
The list of sample codes provided is mentioned above.

NOTE: For queries please contact us at *support@logsun.com*