USER'S MANUAL

FOR

FPGA SPARTEN-6
DEVELOPEMENT BOARD

Manufactured By

LOGSUN SYSTEMS

B-2, RAUNAK, 134/4+3B, NEAR MAYUR COLONY,

KOTHRUD, PUNE-411038

Ph.-020 24356456/24351400
Mob no. 9850588864

Email: - support@logsun.com / info@logsun.com

Table of contents:

Chapter 1: Introduction

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.

B. Product Information

The Spartan®-6 family provides leading system integration capabilities with the lowest total cost for high-volume applications. The thirteen-member family delivers expanded densities ranging from 3,840 to 147,443 logic cells, with half the power consumption of previous Spartan families, and faster, more comprehensive connectivity. Built on a mature 45 nm low-power copper process technology that delivers the optimal balance of cost, power, and performance, the Spartan-6 family offers a new, more efficient, dual-register 6-input lookup table (LUT) logic and a rich selection of built-in system-level blocks. These include 18 Kb (2 x 9 Kb) block RAMs, second generation DSP48A1 slices, SDRAM memory controllers, enhanced mixed-mode clock management blocks, SelectIO™ technology, poweroptimized high-speed serial transceiver blocks, PCI Express® compatible Endpoint blocks, advanced system-level power management modes, auto-detect configuration options, and enhanced IP security with AES and Device DNA protection. These features provide a lowcost programmable alternative to custom ASIC products with unprecedented ease of use. Spartan-6 FPGAs offer the best solution for high-volume logic designs, consumer-oriented DSP designs, and cost-sensitive embedded applications.

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 XC 6SLX9. The XC 6SLX9 having 9152 equivalent logic cells, 90k distributed RAM bit's,maximum 576K block RAM bit's, 102 maximum user I/O and 2 memory controller block.

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**
9. **Serial transmitter / Receiver**
10. **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.

C. Peripherals:

1.1 Liquid 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:

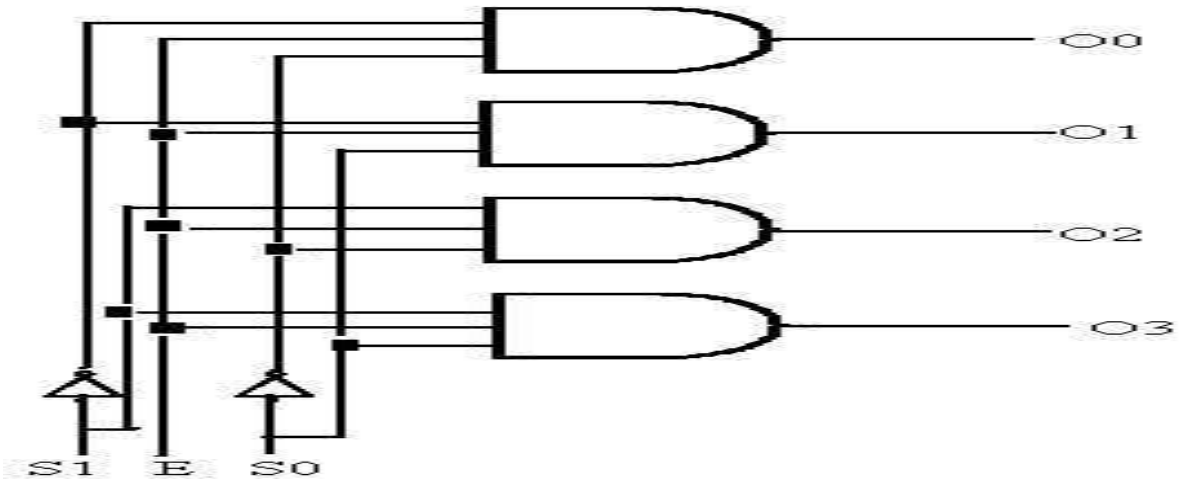
A seven-segment display, or seven-segment indicator, is a form of electronics display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays. Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic 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
x	x	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



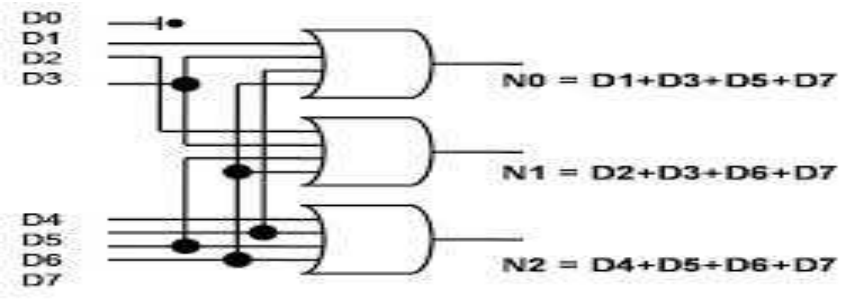
Logic Diagram of 2-to-4 decoder

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 De-multiplexer

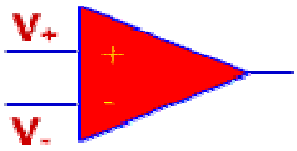
Multiplexer means many into one. A multiplexer is a circuit used to select and route any one of the several input signals to a signal output. A simple example of a non electronic circuit of a multiplexer is a single pole multiposition switch.

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

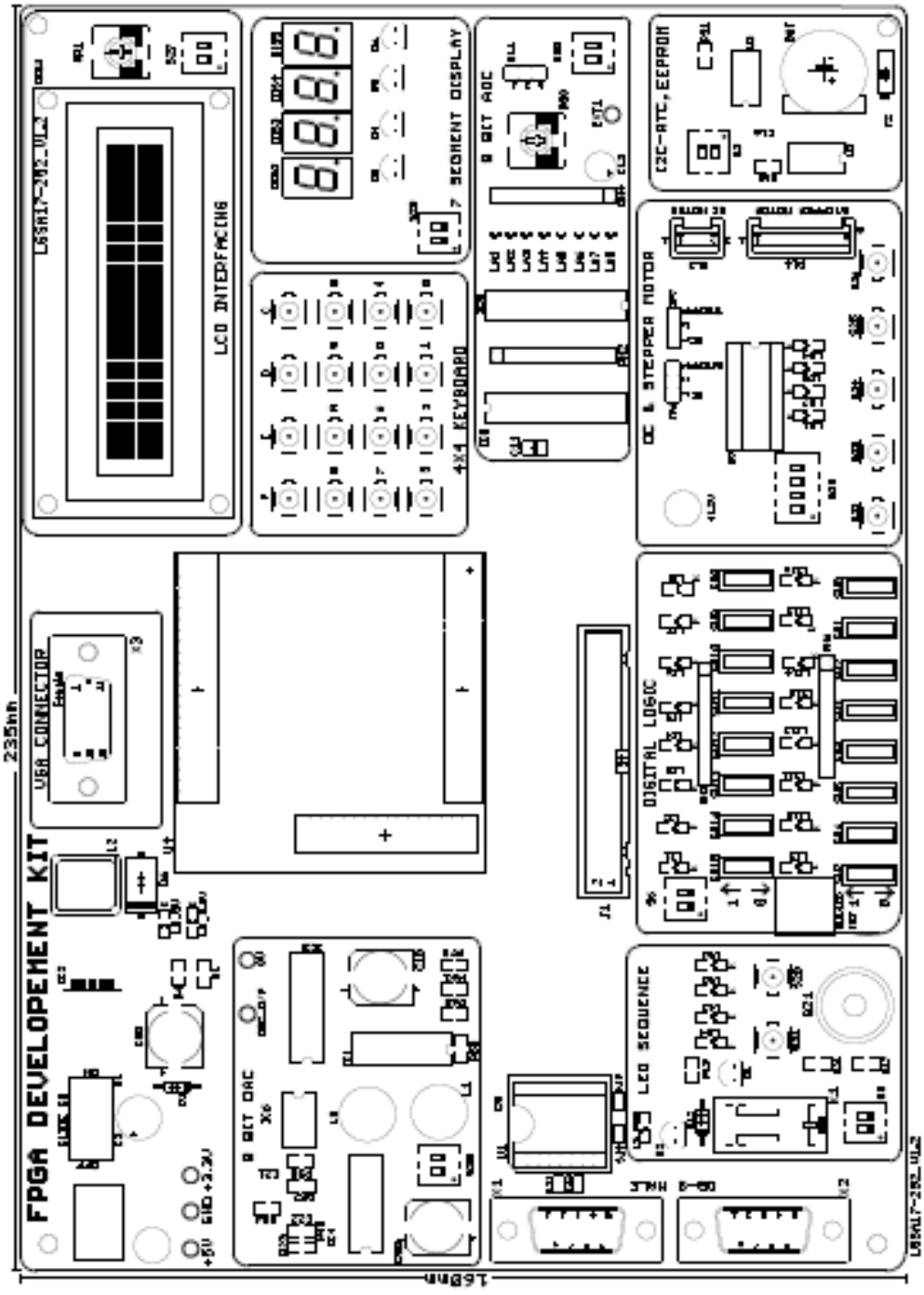
The objective of this tutorial is to introduce the use of sequential logic. The sequence is a sequential circuit. In sequential logic the output depends on the current input values and also the previous inputs. When describing the behavior of a sequential logic circuit we talk about the state of the 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

A Features and Specification of **FPGA DEVELOPMENT KIT**

- **Xilinx FPGA module**
  - XC 6SLX9 FPGA from Xilinx
- Supports for Xilinx PLD's
- Voltage supported 12V
- All FPGA I/Os accessible through headers
- User selectable configuration modes using,JTAG
- LED indicated output
- 7-Segment display
- Liquid crystal display (LCD) section
- 4*4 keyboard matrix
- 16*2 character LCD display
- Relay buzzer section.
- Stepper and DC motor section
- Digital Logic
- 8 bit DAC section
- 8 bit ADC section
- VGA connector section
- Serial communiction section

Chaptor 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.
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.

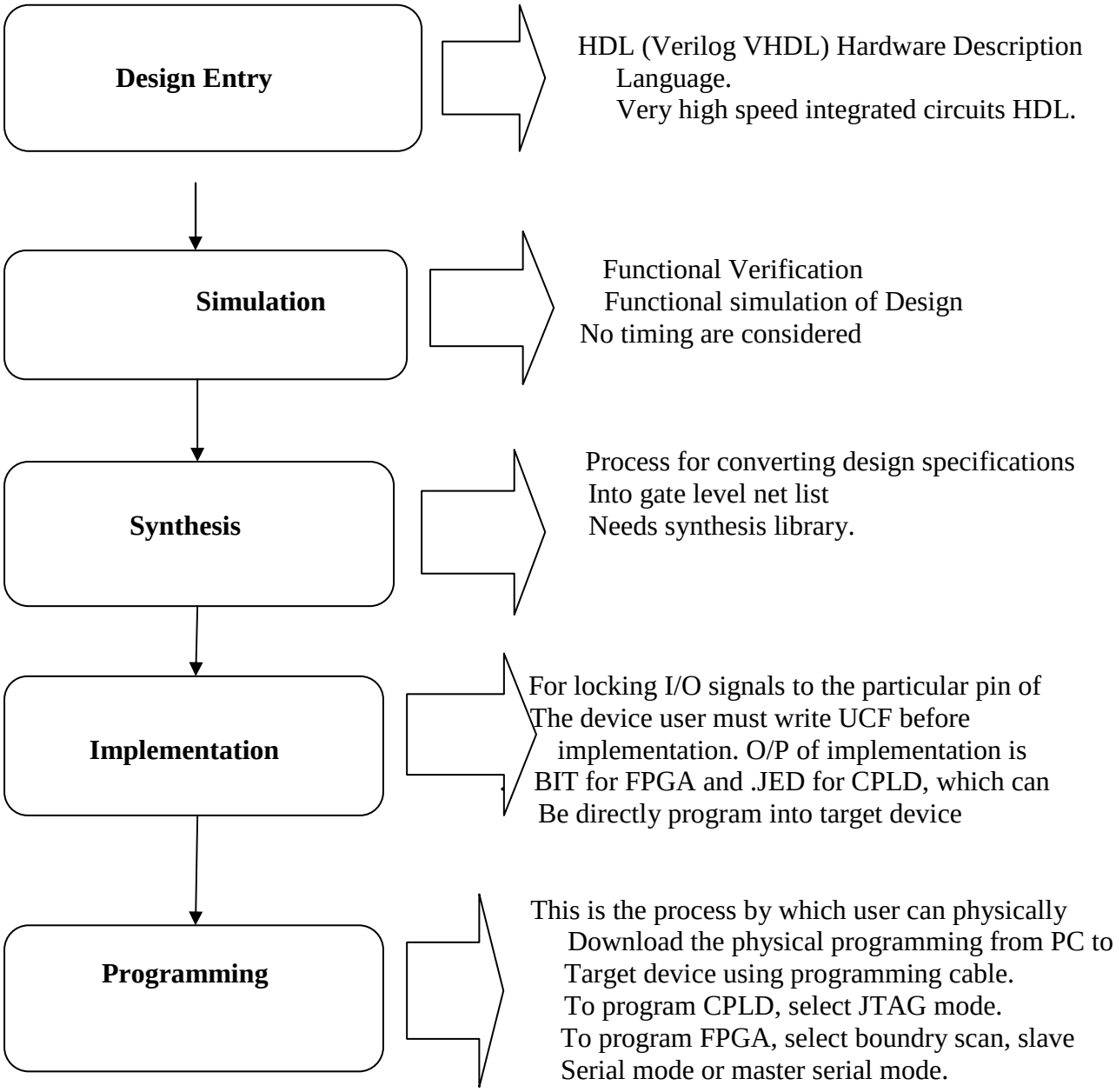
9) PLD devices are sensitive to surge current and voltages.

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

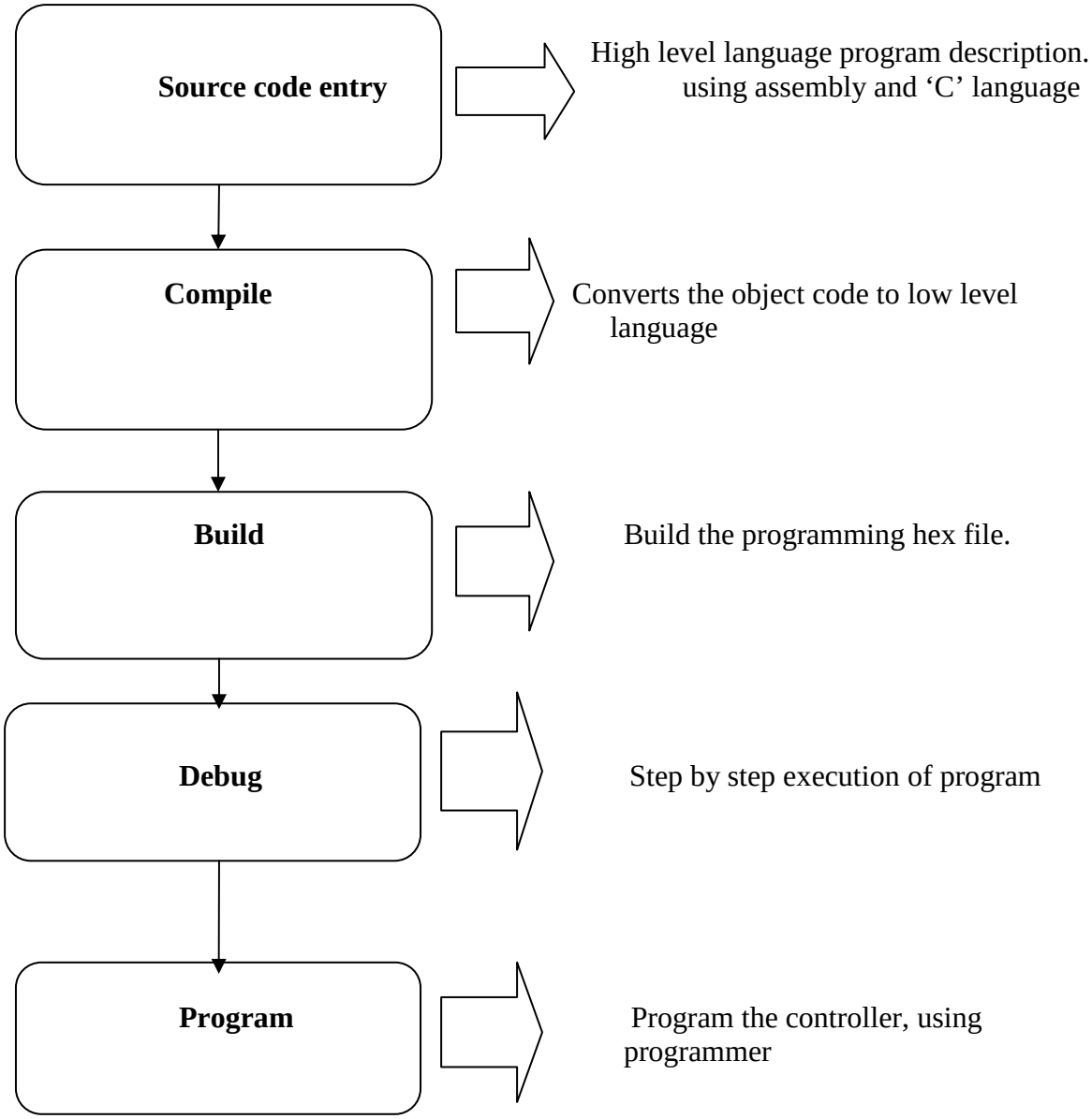
11) Proper selection of switch is must. When any one of section is in use then ON the switch only for that section. & make sure that switches on other sections are OFF.

Chapter 5: Design Flow

A. PLD design flow



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

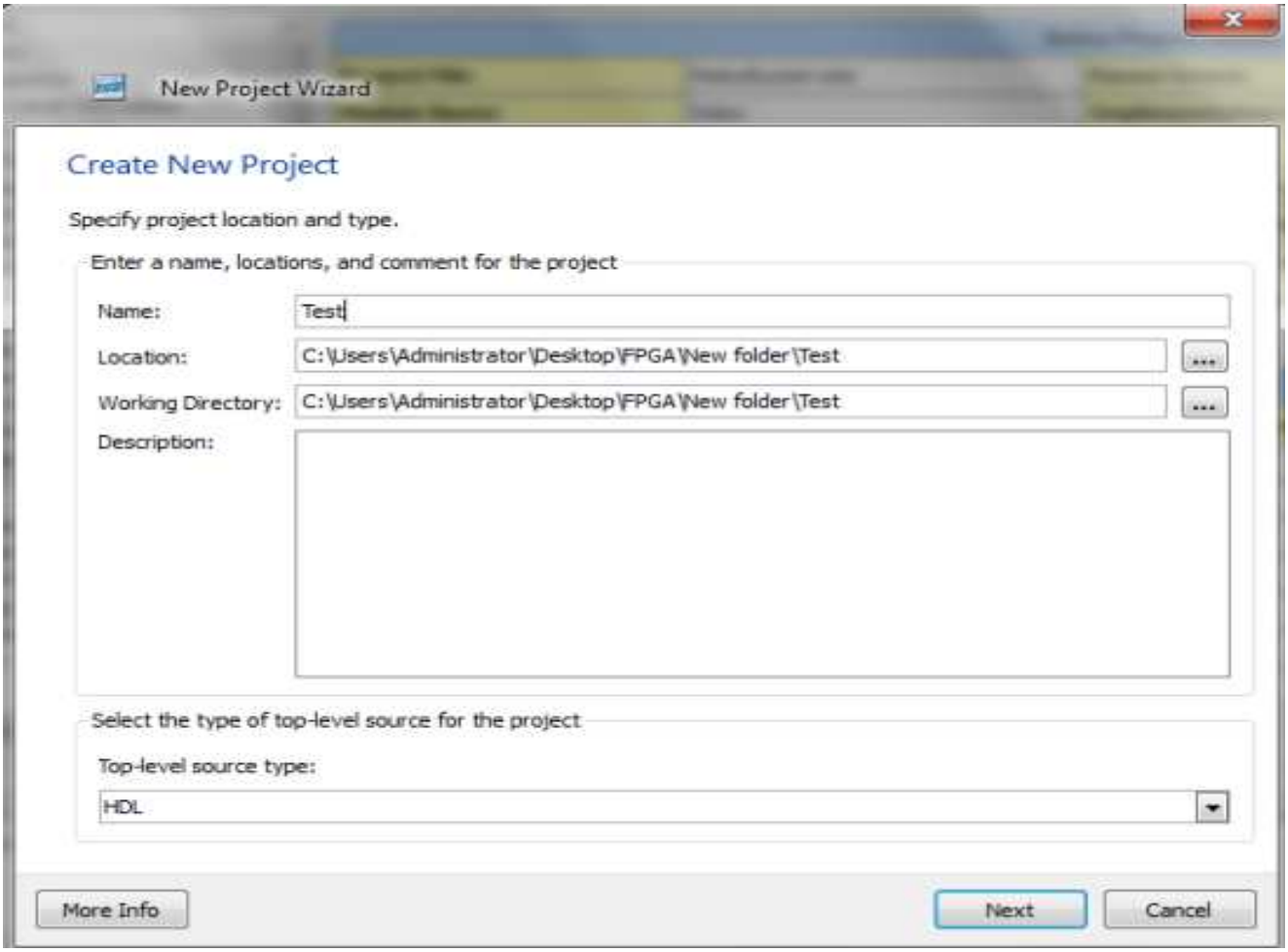
Step 1: open Xilinx ISE design suite 12.2

Step 2 : Create new project

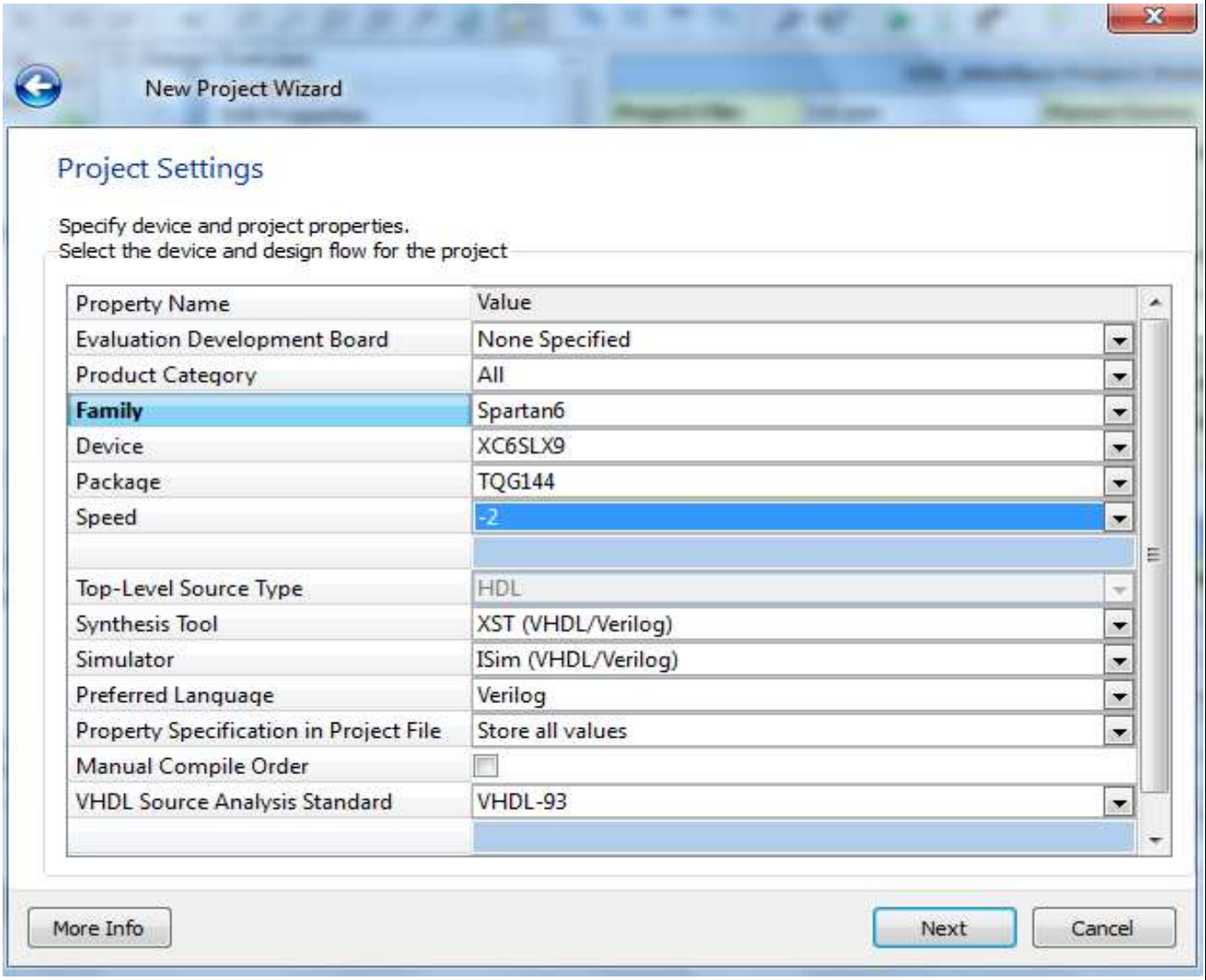
1. To create a new project, open Project Navigator either from the Desktop shortcut icon or by selecting **Start > Programs > Xilinx ISE Design Suite 12.2 > ISE > Project Navigator.**

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:

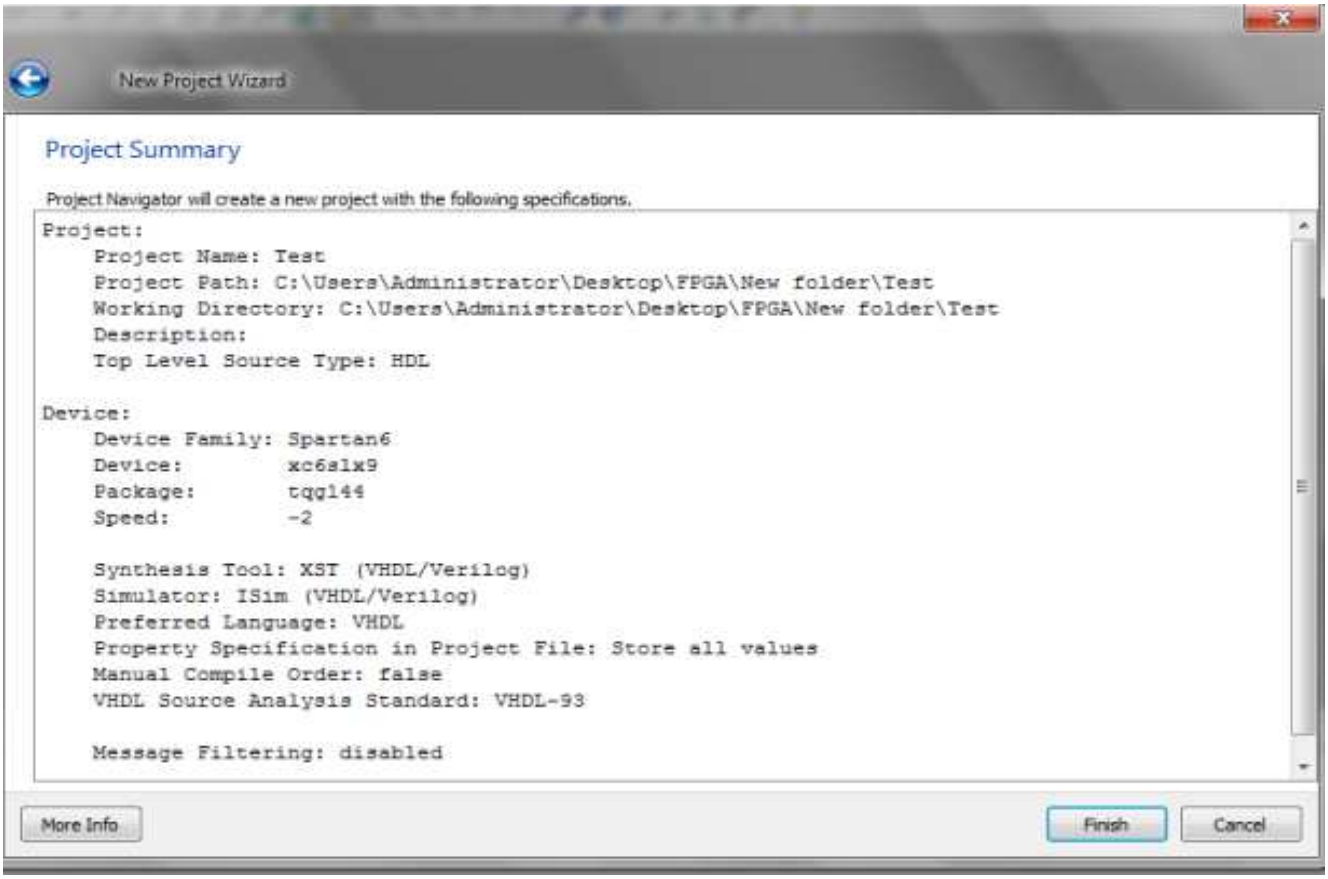


For FPGA, select

1) **Family : Spartan 6s**
2) **Device : XC6SLX9**
3) **Package : TQG144**
4) **Speed : -2**
5) **Preferred Language : VHDL**

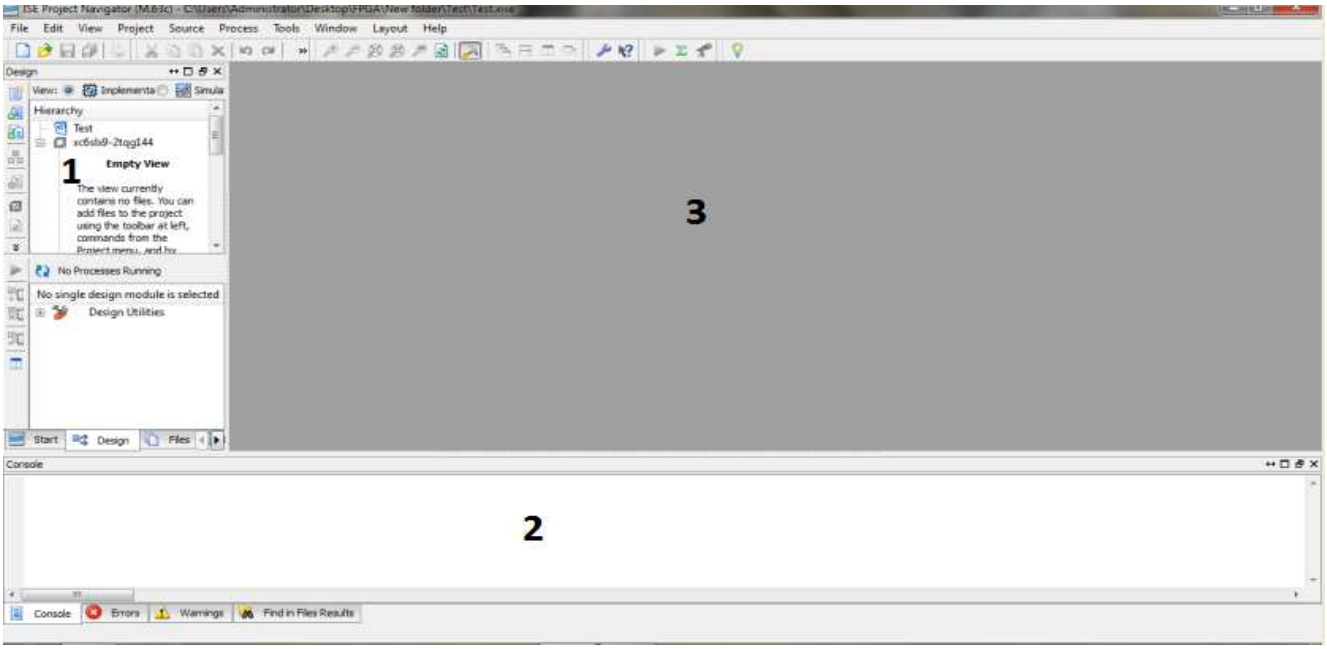
5. The next two dialog boxes give you the option of adding new or existing source files to your project.Since we will fulfill these steps later, click next without adding any source files.

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.



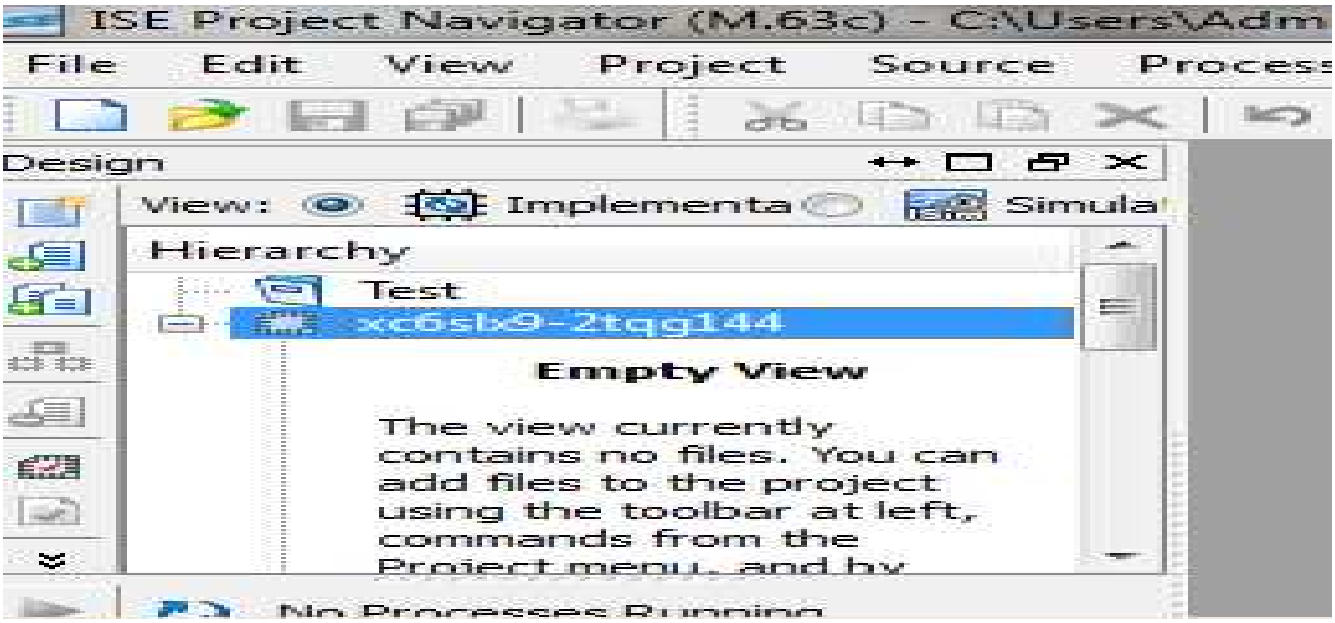
The Design panel (1) contains two windows: a Sources window that displays all source files associated with the current design and a Process window that displays all available processes that can be run on a selected source file.

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.

---

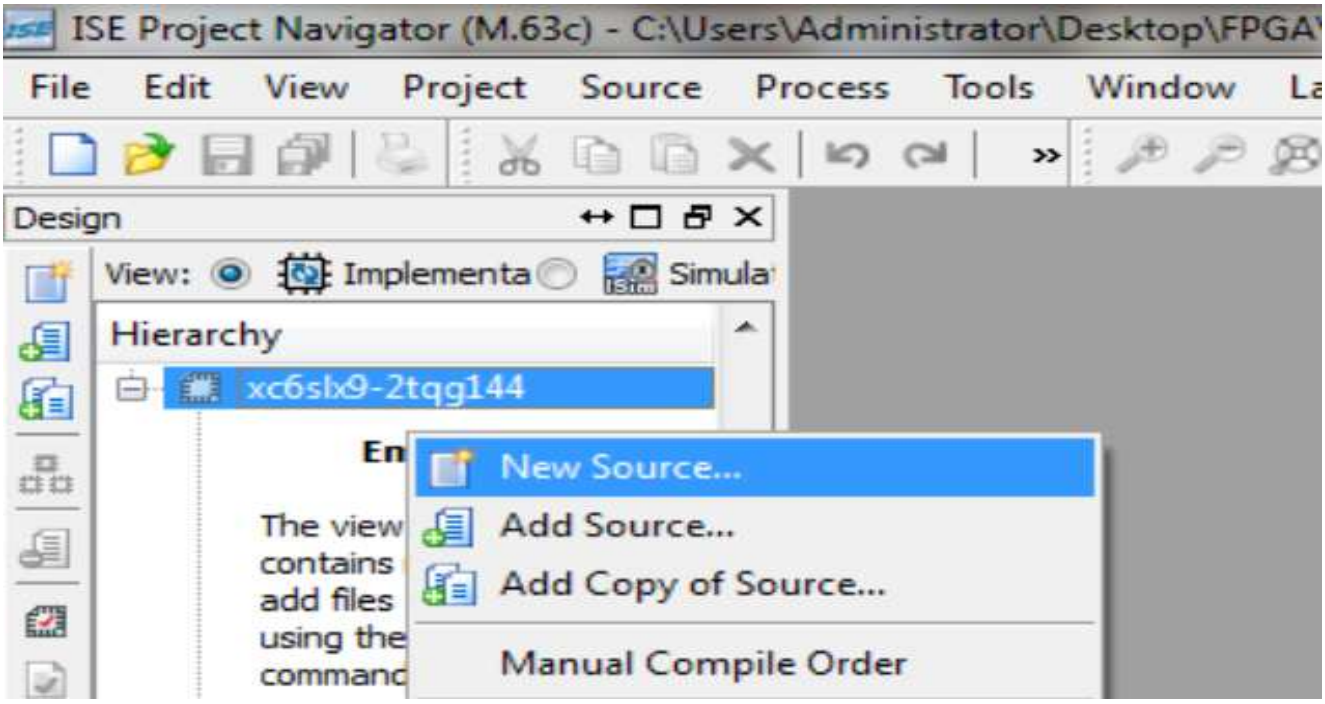
c) Adding New Source Files

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.

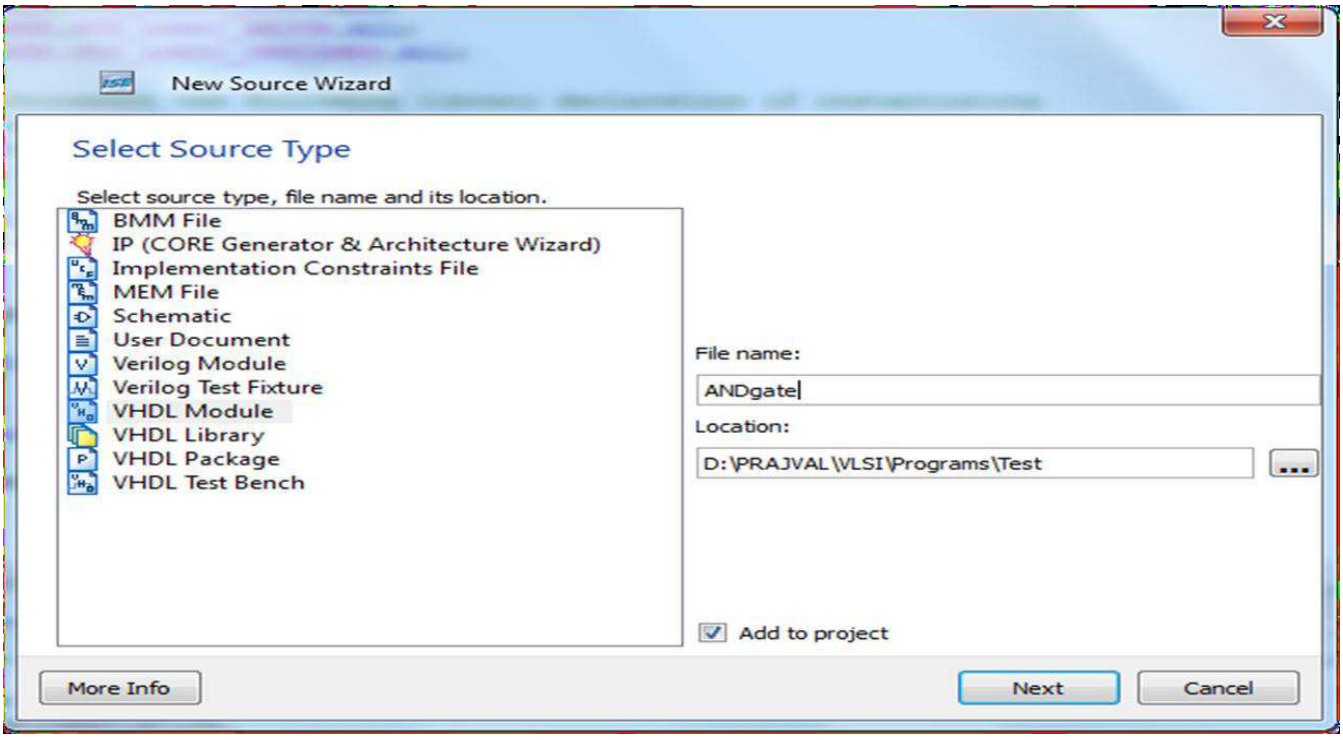


---

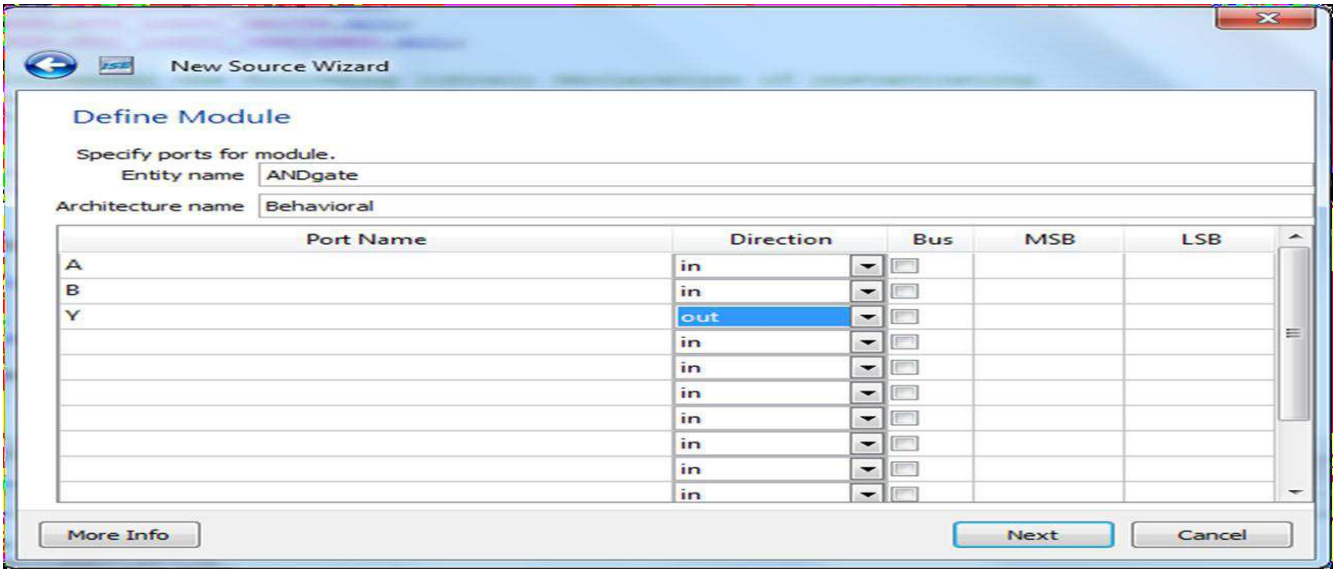
2. You can add a new or existing source file to the project. To do this, right-click on the target device and select one of the three options for adding source files.



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.

```
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if insta
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity Test is
33  end Test;
34
35  architecture Behavioral of Test is
36
37  begin
38
39
40  end Behavioral;
```

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

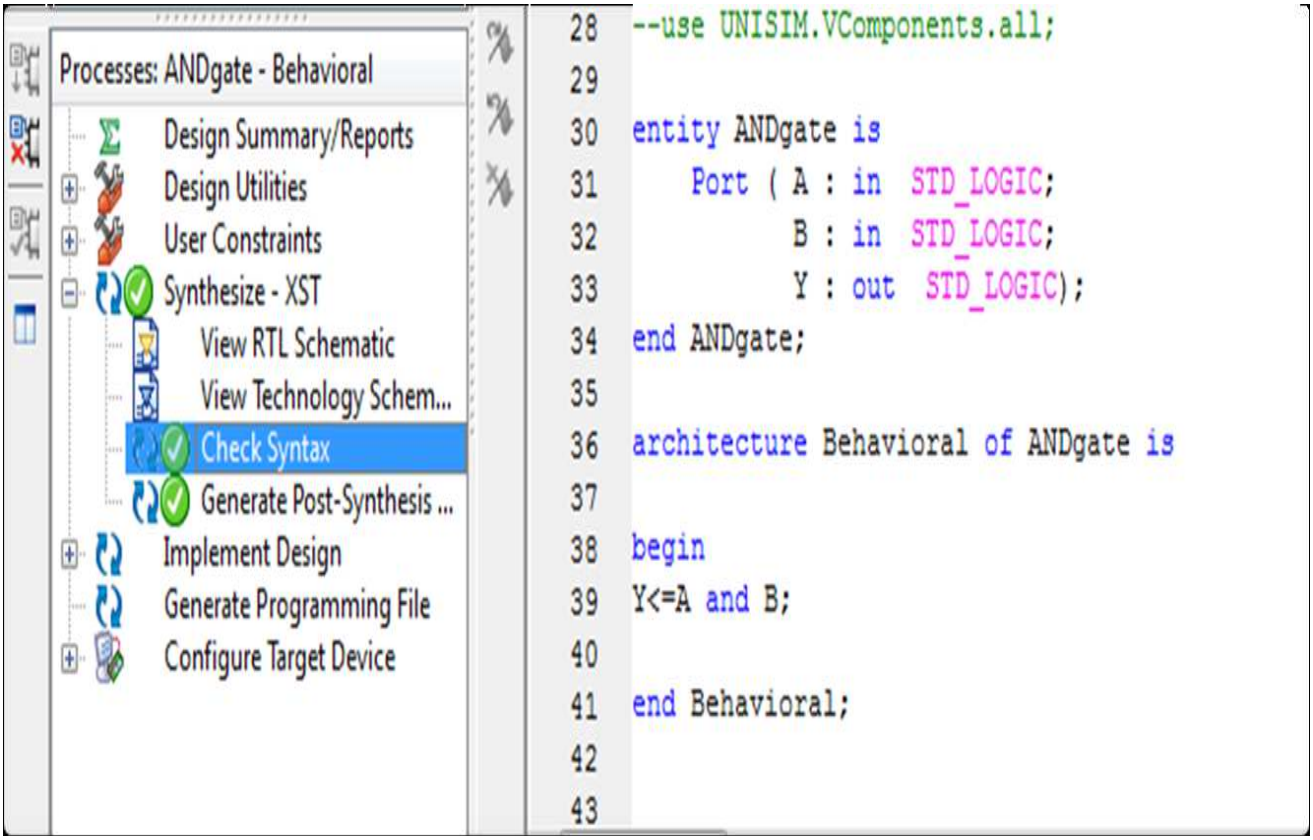
• library definitions

• An entity statement

• An architecture statement with begin and end statements included

• A comment block template for documentation.

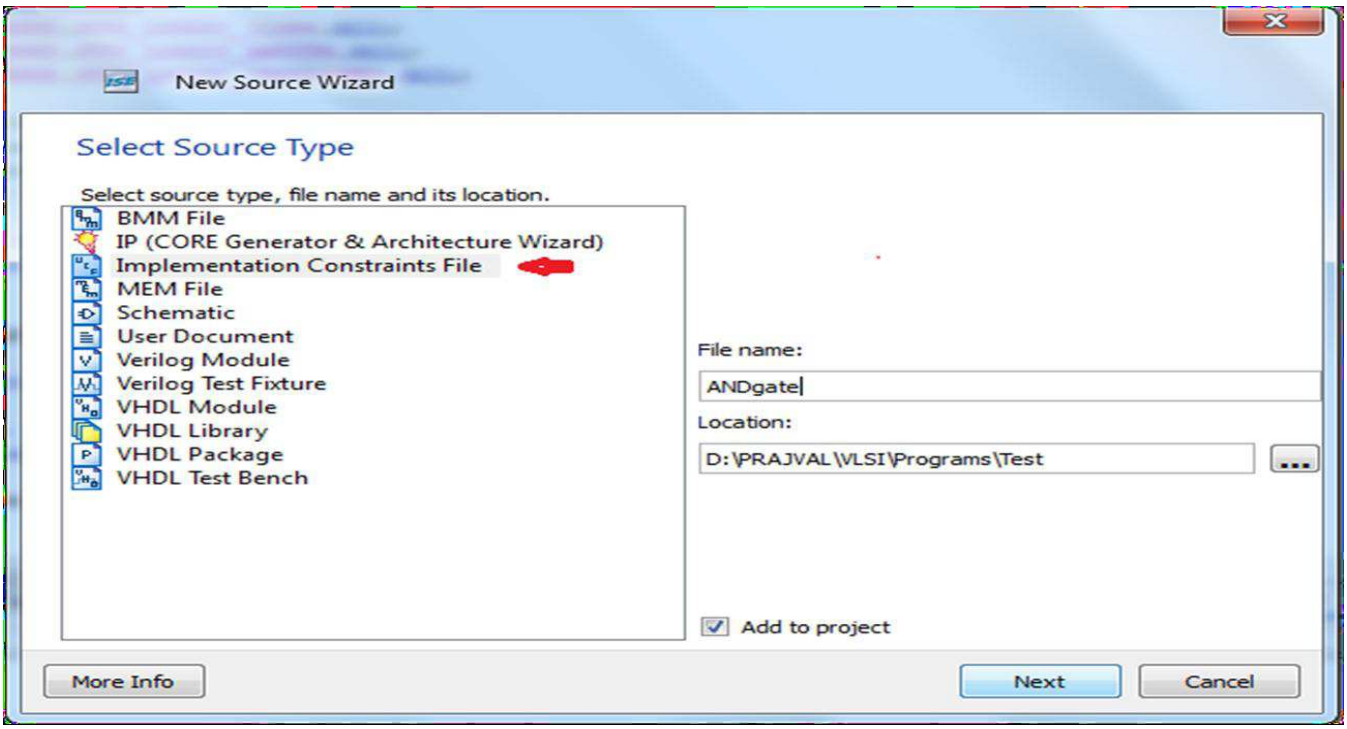
The actual behavioral or structural description of the given circuit is to be placed between the “begin” and “end Behavioral” statements in the file. Between these statements, you can define any VHDL circuit you wish. In this tutorial we use a simple combinational logic example, and then show how it 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:



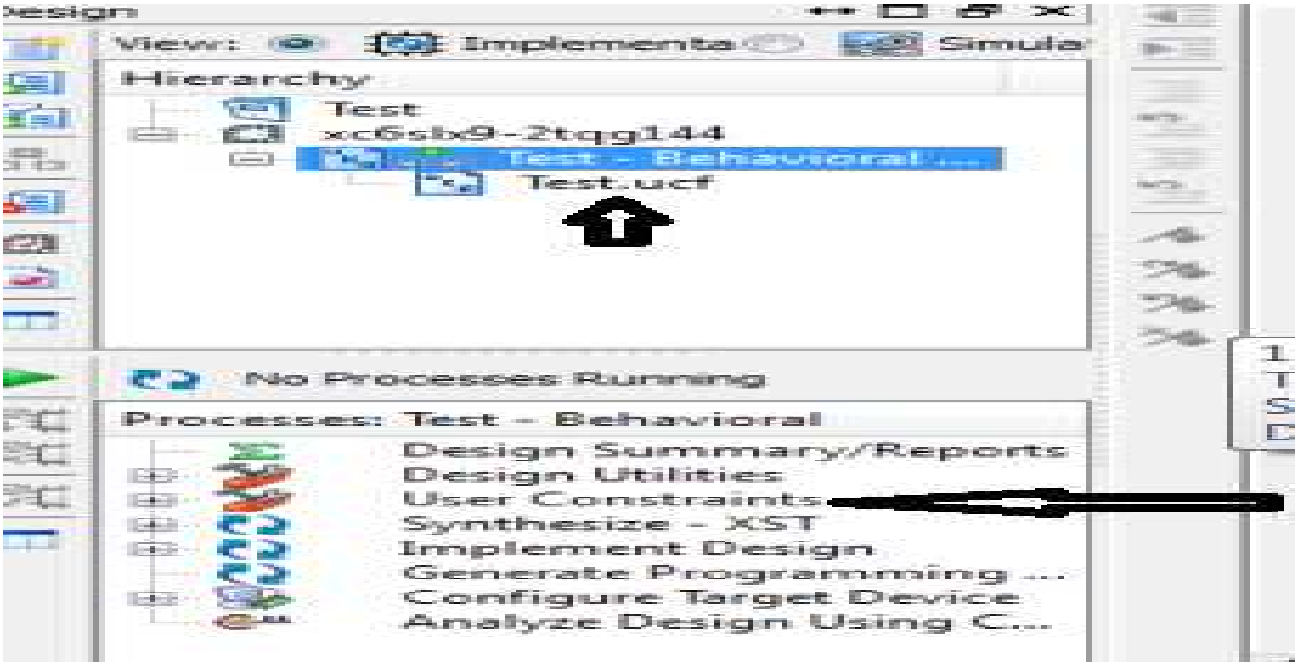
If there are any syntax errors in the source file, ISE can help you find them. For example, the parentheses around the A and B inputs are crucial to this implementation; without them, ISE would return an error during synthesis

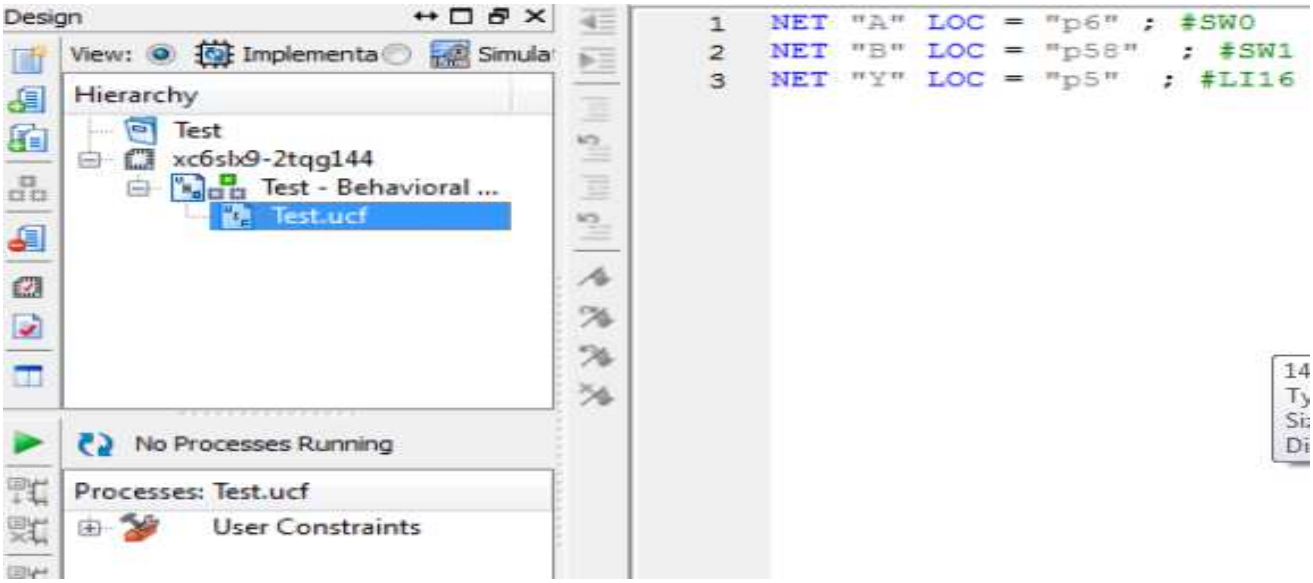
e) UCF File Creation

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.

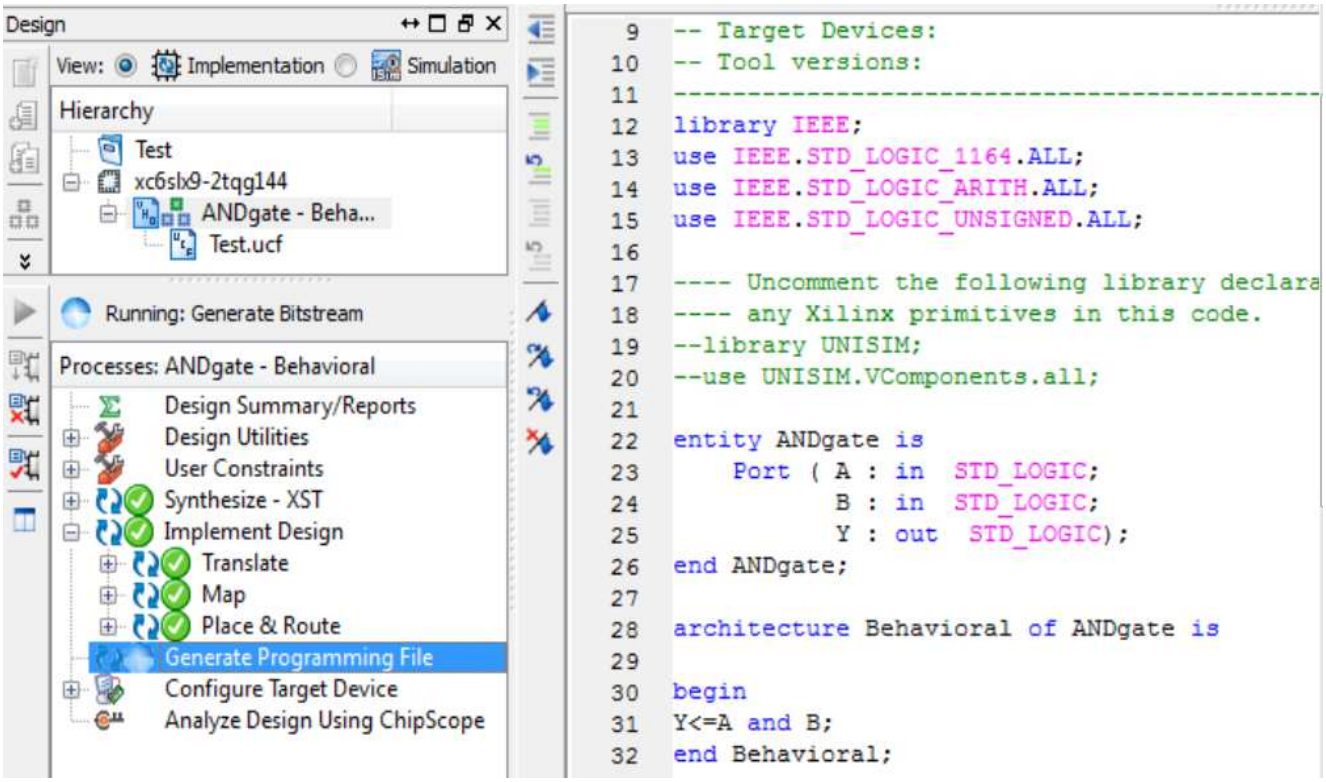




To associate a physical pin with a given net name, type: NET “net name” LOC =”XXX”; on a line in the .ucf file. In the statement, “net name” (quotes included) is the name of the net to attach to pin number XXX (quotes included).
For our example project, the two inputs are assigned to switches 0 through 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

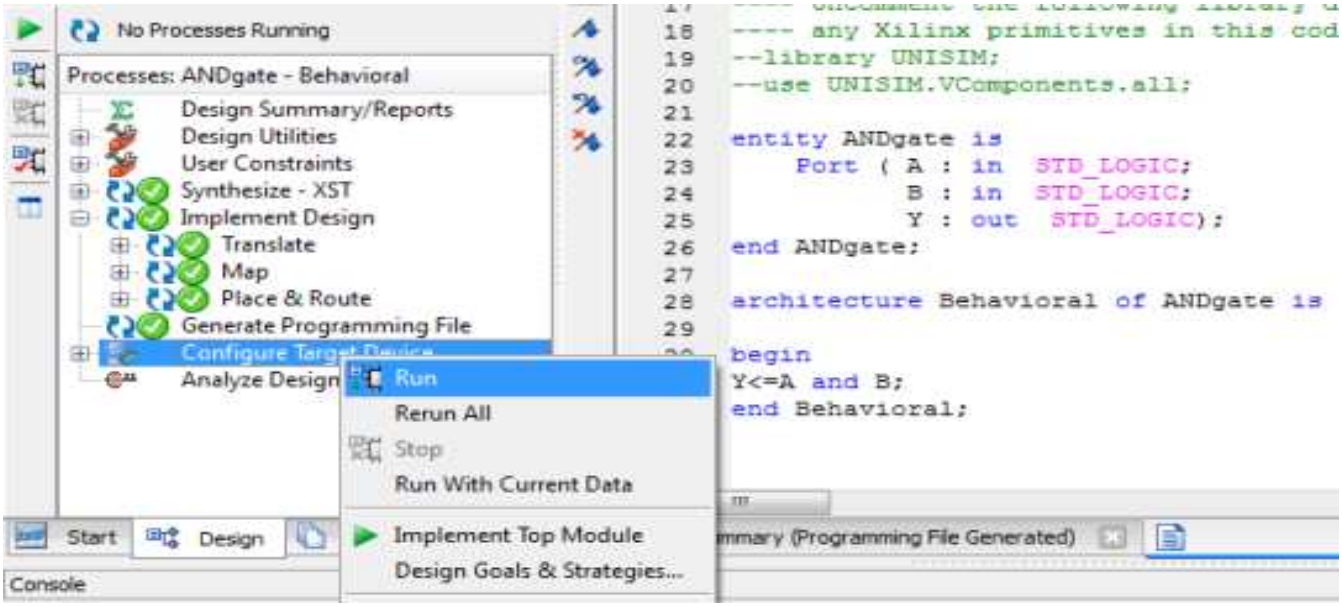
2. Implement Design

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

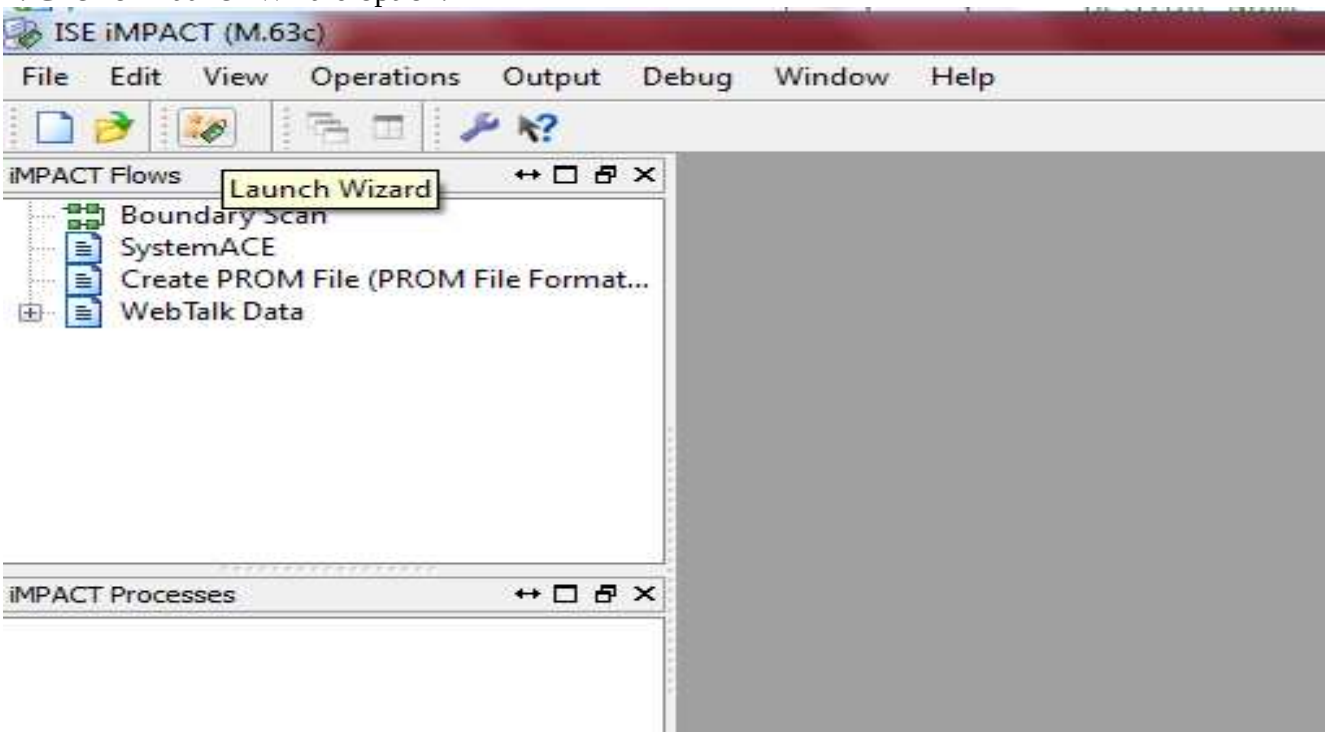
1. After generating programming file double click or Right click and Run on Configure Target Device Option warning window will open click on OK.



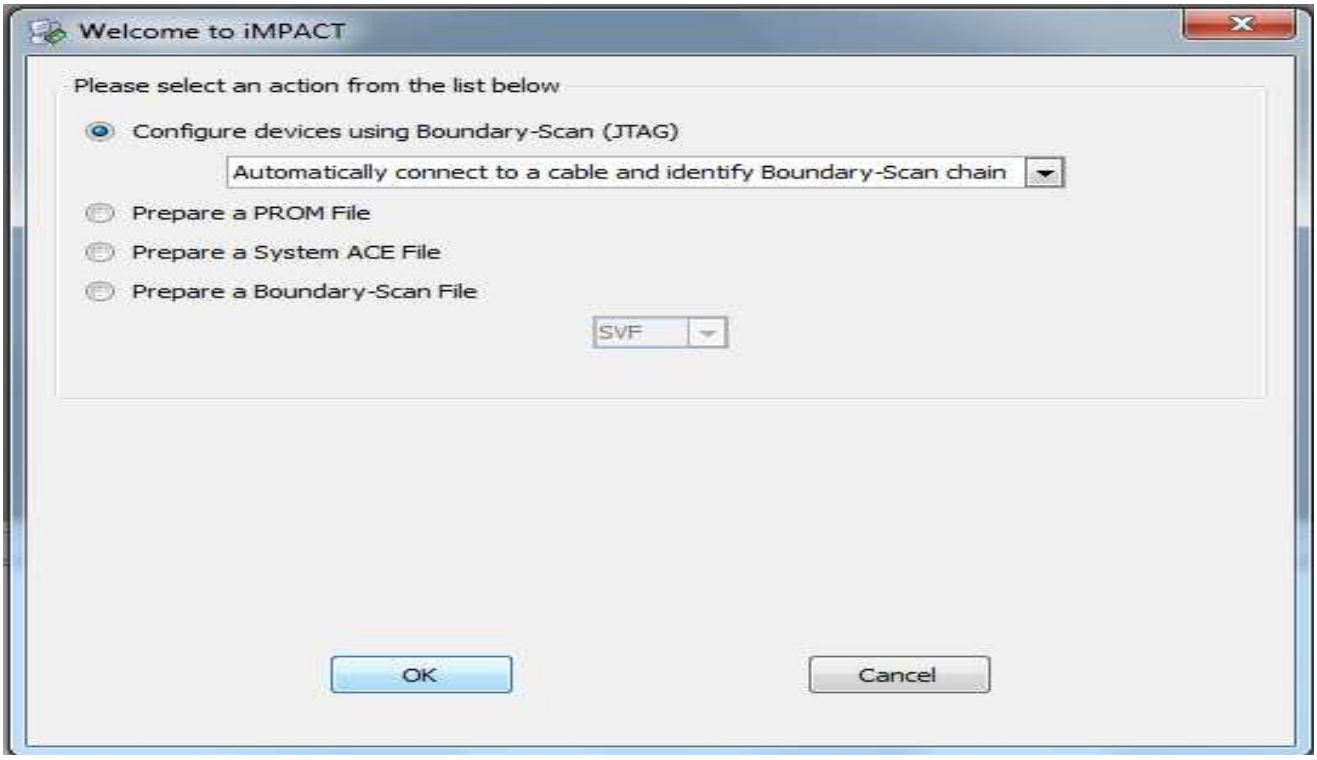
2. After this step ISE iMPAC (M.63c) window will open.

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.



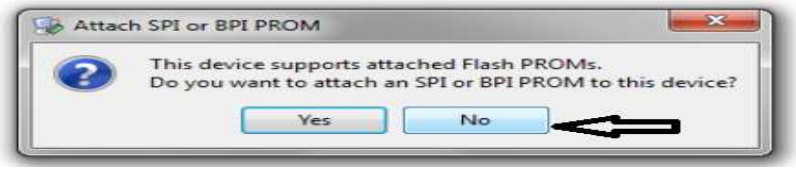
5. After click welcome window will open click OK.



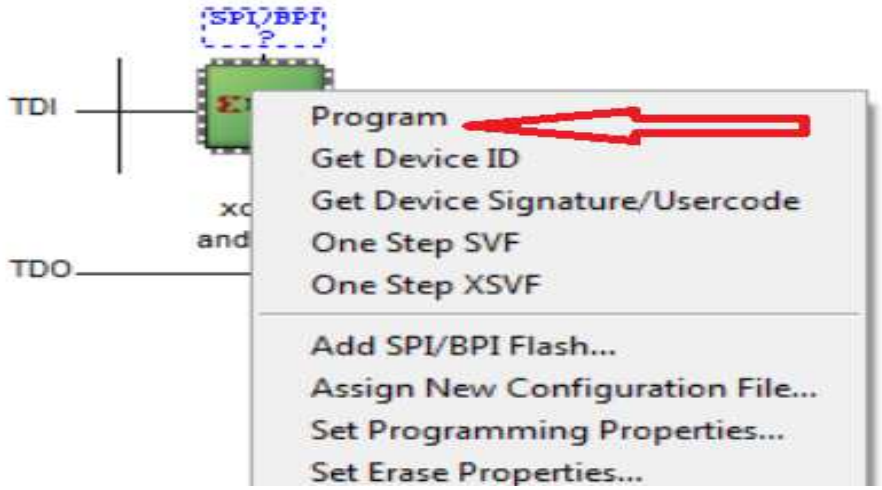
6. If everything ok 'Identify Succeeded' will display. At same time two "Assign New Configuration File" window will open one by one in this window click on Cancel after this "window Device Programming Properties" window will display click on cancel.

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**.**


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...

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	**P6**	**LI1**	**P124**
SW1	**P58**	**LI2**	**P134**
SW2	**P64**	**LI3**	**P138**
SW3	**P75**	**LI4**	**P142**
SW4	**P1**	**LI5**	**P2**
SW5	**P141**	**LI6**	**P16**
SW6	**P137**	**LI7**	**P62**
SW7	**P131**	**LI8**	**P7**
SW8	**P8**	**LI9**	**P111**
SW9	**P61**	**LI10**	**P127**
SW10	**P67**	**LI11**	**P140**
SW11	**P116**	**LI12**	**P144**
SW12	**P143**	**LI13**	**P115**
SW13	**P139**	**LI14**	**P114**
SW14	**P133**	**LI15**	**P10**
SW15	**P126**	**LI16**	**P5**

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** **I(3)=SW3** **I(4)=SW4** **I(5)=SW5** **I(6)=SW6** **I(7)=SW7**	**Q=LI8** **Select lines** **SEL(0)=SW8** **SEL(1)=SW9** **SEL(2)=SW10**
5	**2-to-4 decoder**	**I(0)=SW0** **I(1)=SW1**	**D(0)=LI16** **D(1)=LI15**

		EN=SW15	**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** **NUM(2)=SW6** **NUM(3)=SW7**	**Less=LI16** **Equal=LI15** **Greater=LI14**
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**	**1)PISO** **Clk=RESET** **Reset=SW8** **Load=SW9** **SO=LI16**

		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**	**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**	**Y(0)=LI8** **Y(1)=LI7** **Y(2)=LI6** **Y(3)=LI5**

		S(0)=SW8 **S(1)=SW9** **S(2)=SW10**	
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-6(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

NOTE: Keep the switch ON which experiment is running and Keep the other experiment switch OFF.

A. FPGA DVK

Experiment No. 1

Aim : - Interfacing of FPGA XC 6SLX9 to LCD.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, Platform cable

Procedure:-
1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to 7 Segment with 4*4 Keypad.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to LED sequence.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to DC motor.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to Stepper motor.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to ADC0804.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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.

Experiment No. 7

Aim : - Interfacing of FPGA XC 6SLX9 to DAC0808.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 Board.
5) Make the switch ON of 8 Bit DAC section (S29).
6)Connect the probe of CRO to the **Digital output** and **ground.**

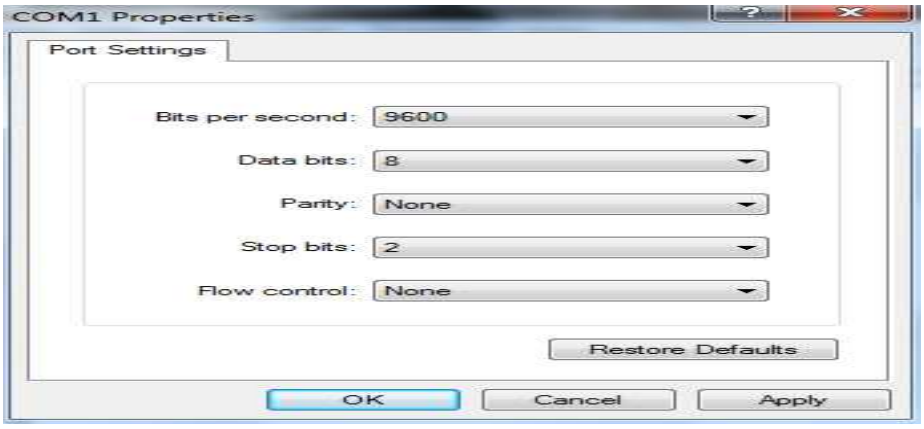
Experiment No. 6

Aim : - Interfacing of FPGA XC 6SLX9 to Serial transmitter / receiver.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, platform cable.

Procedure:-

1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 Board.
5) Connect the F-F serial cable from PC to FPGA DVK.
6)Open the hyperterminal and make setting



7**)** After settings press RESET ones will show the “LOGSUN SYSTEMS PUNE”

B. DIGITAL LOGICS

Experiment No. 1

Aim : - Interfacing of FPGA XC 6SLX9 to design and simulation of adder, Carry Look Ahead Adder.
Requirement: - FPGA XC 6SLX9 board, 12v adaptor, Platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA development board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to design 2-to-4 decoder.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, Platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA XC 6SLX9 and the board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to design 8-to-3 encoder (with and without parity).

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, Platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA XC 6SLX9 and the board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to design 8-to-1 multiplexer.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, Platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA XC 6SLX9 and the board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to design 4-bit Binary to Gray converter.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, Platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA XC 6SLX9 and the board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to design Multiplexer/Demultiplexer and comparator.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, Platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA XC 6SLX9 and the board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to design Full adder using 1 modeling style.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, Platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA XC 6SLX9 and the board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to design S-R, J-K, T, D flip-flop.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, Platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA XC 6SLX9 and the board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to design 4-bit Binary counter.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, Platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA XC 6SLX9 and the board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to design N-bit register of SISO,SIPO,PISO,PIPO.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, Platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA XC 6SLX9 and the board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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 6SLX9 to design Sequence generator.

Requirement: - FPGA XC 6SLX9 board, 12v adaptor, Platform cable.

Procedure:-
1) Connect the 12V adaptor to FPGA XC 6SLX9 and the board.
2) Connect platform cable to FPGA XC 6SLX9 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 XC 6SLX9 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
	LED	
P124	IO_L37P_GCLK13_0	DIGITAL LOGIC-LI1
P134	IO_L34P_GCLK19_0	DIGITAL LOGIC-LI2
P138	IO_L4P_0	DIGITAL LOGIC-LI3
P142	IO_L2P_0	DIGITAL LOGIC-LI4
P2	IO_L83P_3	DIGITAL LOGIC-LI5
P16	IO_L43N_GCLK22_IRDY2_3	DIGITAL LOGIC-LI6
P62	IO_L12P_D1_MISO2_2	DIGITAL LOGIC-LI7
P7	IO_L51N_3	DIGITAL LOGIC-LI8
P111	IO_L66N_SCP0_0	DIGITAL LOGIC-LI9
P127	IO_L36P_GCLK15_0	DIGITAL LOGIC-LI10
P140	IO_L3P_0	DIGITAL LOGIC-LI11
P144	IO_L1P_HSWAPEN_0	DIGITAL LOGIC-LI12
P115	IO_L65P_SCP3_0	DIGITAL LOGIC-LI13
P114	P114	DIGITAL LOGIC-LI14
P10	IO_L50P_3	DIGITAL LOGIC-LI15
P5	IO_L52N_3	DIGITAL LOGIC-LI16
	SWITCHES	
P6	IO_L52P_3	DIGITAL LOGIC-SW0
P58	IO_L14P_D11_2	DIGITAL LOGIC-SW1
P64	IO_L3N_MOSI_CSI_B_MISO0_2	DIGITAL LOGIC-SW2
P75	IO_L74P_AWAKE_1	DIGITAL LOGIC-SW3
P1	IO_L83N_VREF_3	DIGITAL LOGIC-SW4
P141	IO_L2N_0	DIGITAL LOGIC-SW5
P137	IO_L4N_0	DIGITAL LOGIC-SW6
P131	IO_L35N_GCLK16_0	DIGITAL LOGIC-SW7
P8	IO_L51P_3	DIGITAL LOGIC-SW8
P61	IO_L12N_D2_MISO3_2	DIGITAL LOGIC-SW9
P67	IO_L2P_CMPCLK_2	DIGITAL LOGIC-SW10
P116	IO_L64N_SCP4_0	DIGITAL LOGIC-SW11
P143	IO_L1N_VREF_0	DIGITAL LOGIC-SW12
P139	IO_L3N_0	DIGITAL LOGIC-SW13
P133	IO_L34N_GCLK18_0	DIGITAL LOGIC-SW14
P126	IO_L36N_GCLK14_0	DIGITAL LOGIC-SW15
P132	IO_L35P_GCLK17_0	PULSER KEY

2. LED SEQUENCE

PIN NUMBER	PIN NAME	FUNCTION NAME
P102	IO_L65P_SCP3_0	LED SEQUENCE- SW21
P105	IO_L64N_SCP4_0	LED SEQUENCE-SW20
P117	IO_L64P_SCP5_0	LED SEQUENCE-LS1
P118	IO_L63N_SCP6_0	LED SEQUENCE-LS2
P119	IO_L63P_SCP7_0	LED SEQUENCE-LS3
P120	IO_L62N_VREF_0	LED SEQUENCE-LS4
P121	IO_L62P_0	LED SEQUENCE-RELAY (Q2)
P123	IO_L37N_GCLK12_0	LED SEQUENCE-BUZZER(Q1)

3. DC& STEPPER MOTOR

PIN NUMBER	PIN NAME	FUNCTION
P5	IO_L52N_3	START- S22
P34	IO_L52P_3	INC- S23
P7	IO_L51N_3	REV-S24
P35	IO_L51P_3	DEC- S25
P10	IO_L50P_3	STOP-S26
P9	IO_L50N_3	L293D EN1
P17	IO_L43P_GCLK23_3	L293D EN2
P12	IO_L49P_3	L293D I/P1
P11	IO_L49N_3	L293D I/P2
P14	IO_L44N_GCLK20_3	L293D I/P3
P15	IO_L44P_GCLK21_3	L293D I/P4

4. 8 BIT DAC

PIN NUMBER	PIN NAME	FUNCTION
P98	IO_L34P_1	IC 0808-12 PIN(A8)
P97	IO_L34N_1	IC 0808-11 PIN(A7)
P100	IO_L33P_1	IC 0808-10 PIN(A6)
P99	IO_L33N_1	IC 0808-9 PIN(A5)
P102	IO_L32P_1	IC 0808-8PIN(A4)
P101	IO_L32N_1	IC 0808-7PIN(A3)
P105	IO_L1P_1	IC 0808-6PIN(A2)
P104	IO_L1N_VREF_1	IC 0808-5PIN(A1)

5. VGA CONNECTOR

PIN NUMBER	PIN NAME	FUNCTION NAME
P87	IO_L42N_GCLK6_TRDY1_1	RED VIDEO
P93	IO_L41P_GCLK9_IRDY1_1	GREEN VIDEO
P92	IO_L41N_GCLK8_1	BLUE VIDEO
P95	IO_L40P_GCLK11_1	HORIZONTAL SYNC
P94	IO_L40N_GCLK10_1	VERTICAL SYNC

6. SERIAL COMMUNICTION

PIN NUMBER	PIN NAME	FUNCTION
P112	IO_L66P_SCP1_0	RS-232 10 PIN
P114	IO_L15N_3	RS-232 9 PIN

7. 8 BIT ADC

PIN NUMBER	PIN NAME	FUNCTION
P21	IO_L42N_GCLK24_3	DB7
P23	IO_L41N_GCLK26_3	DB6
P24	IO_L41N_GCLK27_3	DB5
P26	IO_L37N_3	DB4
P27	IO_L37P_3	DB3
P29	IO_L36N_3	DB2
P30	IO_L36P_3	DB1
P32	IO_L2N_3	DB0
P33	IO_L2P_3	CS
P34	IO_L1N_VREF_3	RD
P35	IO_L1P_3	WR
P38	IO_L65N_CSO_B_2	INTR

8. 4x4 KEYPAD

PIN NUMBER	PIN NAME	FUNCTION NAME
P66	IO_L2N_CMPMOSI_2	COLUMN 3
P61	IO_L12N_D2_MISO3_2	COLUMN 2
P62	IO_L12P_D1_MISO2_2	COLUMN 1
P58	IO_L14P_D11_2	COLUMN 0
P75	IO_L74P_AWAKE_1	ROW 3
P126	IO_L2P_CMPCLK_2	ROW 2
P16	IO_L43N_GCLK22_IRDY2_3	ROW 1
P64	IO_L3N_MOSI_CSI_B_MISO0_2	ROW 0

9. LCD

PIN NUMBER	PIN NAME	FUNCTION NAME
P74	IO_L74N_DOUT_BUSY_1	D7
P79	IO_L47P_1	D6
P78	IO_L47N_1	D5
P81	IO_L46P_1	D4
P80	IO_L46N_1	D3
P83	IO_L45P_1	D2
P82	IO_L45N_1	D1
P85	IO_L43P_GCLK5_1	D0
P84	IO_L43N_GCLK4_1	EN
P88	IO_L42P_GCLK7_1	RS
GND	R/W PIN OF LCD	RW

10 . SEVEN SEGMENT DISPLAY

PIN NUMBER	PIN NAME	FUNCTION NAME
P40	IO_L64N_D9_2	DISPLAY 5(Q6)
P44	IO_L62P_D5_2	DISPLAY 4(Q5)
P43	IO_L62N_D6_2	DISPLAY 3(Q4)
P46	IO_L49P_D3_2	DISPLAY 2(Q3)
P48	IO_L48P_D7_2	SEGMENT A
P45	IO_L49N_D4_2	SEGMENT B
P51	IO_L31P_GCLK31_D14_2	SEGMENT C
P47	IO_L48N_RDWR_B_VREF_2	SEGMENT D
P57	IO_L14N_D12_2	SEGMENT E
P50	IO_L31N_GCLK30_D15_2	SEGMENT F
P59	IO_L13N_D10_2	SEGMENT G
P55	IO_L30N_GCLK0_USERCCLK_2	SEGMENT DP

11. 34 PIN CONNECTOR DETAILS

PIN NUMBER	STANDARD NAME	FUNCTION NAME
P126	IO_L36N_GCLK14_0	34 PIN FRC-PIN NO 3
P124	IO_L37P_GCLK13_0	34 PIN FRC-PIN NO 4
P131	IO_L35N_GCLK16_0	34 PIN FRC-PIN NO 5
P127	IO_L36P_GCLK15_0	34 PIN FRC-PIN NO 6
P133	IO_L34N_GCLK18_0	34 PIN FRC-PIN NO 7
P132	IO_L35P_GCLK17_0	34 PIN FRC-PIN NO 8
P16	IO_L4N_0	34 PIN FRC-PIN NO 9

P134	IO_L34P_GCLK19_0	34 PIN FRC-PIN NO 10
P139	IO_L3N_0	34 PIN FRC-PIN NO 11
P138	IO_L4P_0	34 PIN FRC-PIN NO 12
P141	IO_L2N_0	34 PIN FRC-PIN NO 13
P140	IO_L3P_0	34 PIN FRC-PIN NO 14
P143	IO_L1N_VREF_0	34 PIN FRC-PIN NO 15
P142	IO_L2P_0	34 PIN FRC-PIN NO 16
P1	IO_L83N_VREF_3	34 PIN FRC-PIN NO 17
P144	IO_L1P_HSWAPEN_0	34 PIN FRC-PIN NO 18
P116	IO_L64N_SCP4_0	34 PIN FRC-PIN NO 19
P2	IO_L83P_3	34 PIN FRC-PIN NO 20
P75	IO_L74P_AWAKE_1	34 PIN FRC-PIN NO 21
P115	IO_L65P_SCP3_0	34 PIN FRC-PIN NO 22
P67	IO_L2P_CMPCLK_2	34 PIN FRC-PIN NO 23
P16	IO_L43N_GCLK22_IRDY2_3	34 PIN FRC-PIN NO 24
P64	IO_L3N_MOSI_CSI_B_MISO0_2	34 PIN FRC-PIN NO 25
P66	P66	34 PIN FRC-PIN NO 26
P61	IO_L12N_D2_MISO3_2	34 PIN FRC-PIN NO 27
P62	IO_L12P_D1_MISO2_2	34 PIN FRC-PIN NO 28
P58	IO_L14P_D11_2	34 PIN FRC-PIN NO 29
P10	IO_L50P_3	34 PIN FRC-PIN NO 30
P8	IO_L51P_3	34 PIN FRC-PIN NO 31
P7	IO_L51N_3	34 PIN FRC-PIN NO 32
P6	IO_L52P_3	34 PIN FRC-PIN NO 33
P5	IO_L52N_3	34 PIN FRC-PIN NO 34

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*