

A Manual to FPGA and PCB Designing

Report on MINI PROJECT

Sayak Haldar (111205038)
Shuvam Bosana (111205042)
Subham Banerjee (111205014)

Bengal Engineering and Science University, Shibpur,
Howrah-03

October 20, 2013

Project Outline

- Introduction
- Our Work with FPGA
- Familiarisation with VHDL
- Working with PCB
- Future Scope
- References

Introduction

- Preamble
- Familiarisation with FPGA
- Familiarisation with VHDL
- Exploring latex
- Exploring Beamer

Preamble

This is our 3rd semester "Mini Project" work in which we decided to fulfill the following broad objectives:

- 1 To familiarise with FPGA(Field Programmable Gate Array) and implement basic and complex combinational Logic circuits using Quartus-A hardware accelerating software.
- 2 To be familiar with VHDL(Very High Speed Integrated Circuit Hardware Description Language).
- 3 To be familiar with the idea of documentation using Latex.

In this mini project we have especially stressed on the practical implementation of our design in the Quartus software. We have drawn the schematic block diagram using this software and simulated it, generating a vector waveform file and finally we viewed the output. Further we assigned pins to our file corresponding to those FPGA interface board. We mounted the files into the FPGA and used an Oscilloscope to observe the signal output which generated.

Familiarisation with FPGA:

As an assignment of the mini project we took up the task of synthesizing a few combiantional and sequential circuit designs with FPGA. It helped us again a basic insight to its advantages and helped us understanding the nature of signals obtained in an oscilloscope when it is connected with FPGA board.

Familiarisation with VHDL:

We studied the hardware implementations using VHDL (Very High speed integrated circuit Hardware Description Language) and understood its advantages and disadvantages over the Schematic Block Diagram which we used in Quartus software to synthesize the logic circuits.

Exploring Latex

We have learnt Latex which is a strong document creating software. In this semester, we have explored the process of creating a bibliography in latex and gained knowledge about the advantage of using latex over all other document creating software in case of creating bibliography of a document. Some other advantages of using latex are:-

- ① LaTeX provides very high quality and is extremely customizable.
- ② It's extremely stable, no matter how complex the documents are.
- ③ It's free and Open Source, we can study and improve everything as we do on this site.

- ④ LaTeX is portable concerning its implementation, your document source and its output – all is cross-platform.
- ⑤ It provides a logical approach to create documents instead of a physical, enhancing consistency.
- ⑥ Your document is safe because the file format is open and there's no virus threat.

Exploring Beamer

we have learnt beamer, which is the presentation package of Latex in this semester. We have explored the process of creating a presentation using the beamer presentation package and gained knowledge about the advantages of using it over other presentation creating software like ms powerpoint. One of its major advantages over powerpoint is its output format, which is pdf and is platform independent. Another advantage is that, if we have to use lots of mathematical notation in our presentation ,its better to use beamer then.

- A few Words on FPGA
- Quartus II
- Schematic Design Entry Method with Quartus Graphical Simulation
- Launch Quartus II software
- Design the circuit
- Compile the Half Adder
- Pictorial representation

Our work with FPGA:

1.A Few words on FPGA

- FPGA are fine grain devices containing hundreds (up to 100000) of tiny blocks of logic with flip-flops , combinational logic and memories.
- The FPGA are RAM based. They need to be configured at each power-up and it needs external configuration memory.
- They have special resources to implement binary counters , arithmetic functions like adders, comparators and RAM.
- The idle power consumption is low.
- FPGA is suited for timing circuit because they have more registers.

2.Quartus II:

In this mini project we have used Quartus II : 9th edition to synthesize combinational circuits in an FPGA. The FPGA we used is of the Cyclone family (EP1C12Q240CB) . Quartus II is a tool which provides us with not only Schematic Block Diagram method of designing a circuit but also provides us with a platform for designing the same using a hardware accelerating language such as VHDL and VERILOG. This hardware accelerating software, Quartus is of great use as it helps generate functions, compile our designs, add clock pulses with various duty cycle , count and frequency.Further it allows us to have an idea of the shape of the curve generated. If correct pins of a FPGA interface board is assigned in this software by the user, we can get the required function generated in the LCD display of the interface board or even in an oscillator.

3.Schematic Design Entry Method with Quartus Graphical Simulation

Here we describe the basic steps in detail in order to design a combinational logic circuit in the Quartus 2 software. It will be easier to demonstrate as well as to understand these steps by taking an example. Note that these steps are totally basic and generalised. Any circuit can be designed following them. Just to make the steps more understandable we present these steps for developing and simulating a Half Adder using Schematic Design Entry Method.

Launch Quartus II Software

- ❶ Launch the Quartus II software.
- ❷ Go to File - Project Wizard (on the File menu, click New Project Wizard)
- ❸ New Project Wizard: Introduction windows appears. It shows you the five steps you may need to go through in setting up a new project. Click Next.
- ❹ New Project Wizard: Directory, Name, Top-Level Entity window appears.
 - a. Enter the working project directory
 - b. Enter project name: HalfAdder. As you type HalfAdder name, Top-Level Entity name will be automatically filled with HalfAdder.

- c. Click next. When asked to create the directory, click Yes.
- 5 New Project Wizard: Add Files window appears. You have no files to add now. Hence, just click Next.
- 6 New Project Wizard: Family and device settings window appears. Scroll up/down the family choices and choose Cyclone I family. Scroll up/down the available devices and choose the FPGA number you are working with. Leave other options as they are. Click Finish.

Design the circuit

- 1 File New. New file option window appears.
- 2 Select Block Diagram/ Schematic File. Click OK. BDF Editor window appears.
- 3 Double click anywhere on the window. Select Symbol window appears.
- 4 Type xor in the Name box. Click Ok. Drag the XOR symbol and put it in the middle of the window.
- 5 Repeat steps 3 and 4 but with and2 symbol name to put a 2-input AND symbol at the bottom of the XOR symbol.
- 6 Input ports and output ports are symbols with names "input" and "output", respectively. Thus, repeat steps 3 and 4 with symbol name "input" and "output" to put two input ports and two output ports on the window. Click and drag the symbols.

- 7 Click the orthogonal node tool in the list of tool on the left of the BDF window .To connect point A to point B, position the cursor on point A, click and hold the left button of the mouse, drag the line and drop it at point B.

Example: To connect input symbol to the XOR symbol input, position the cursor on the right end point of the input symbol, click and hold the left mouse button, drag the line and drop it on the input of the XOR symbol. Follow the steps above connect the input symbols and the output symbols to the XOR and AND symbols.

- 8 Click the cursor in the list of tool on the left of the BDF window . Double click the input symbol and change the input name to X. Double click the other input symbol and change the input name to Y. Repeat the steps to name the output names to SUM and CARRY, respectively.


- 9 Double click anywhere on the window. Select Symbol window appears.
- 10 Type title in the Name box. Click Ok. Drag the title symbol and put it the lower right- hand part of the window
- 11 Double click the title symbol, the title block properties window appears. Click any field that you want to modify and change the value.
- 12 Save the bdf file as HalfAdder.bdf, the same name you used as the project name with bdf extension.

Compile the half adder

Go to Processing Start Compilation. If there is errors in the message window, fix the errors and compile the design again. Ignore the warnings.

Simulate the half adder (Timing Simulation)

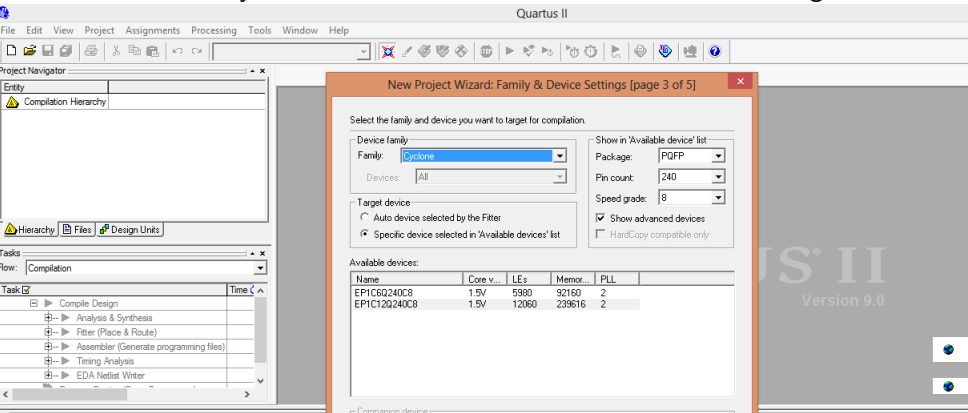
- 1 File - New.New file option window appears.
- 2 Click Other Files tab and select Waveform Vector File. Click Ok. Waveform Vector File (wvf) windows appears.
- 3 Left click anywhere in the Name pane (left of wvf window) and click Insert Insert Node or Bus. Insert Node or Bus window appears.

- 4 Click Node Finder. Node Finder window appears.
- 5 Select Pins: all in the Filter selection box and click List. Move all signals from the left pane (Nodes Found) to the right pane (Nodes Selected) by clicking . Click OK. On the Insert Node or Bus window click OK again. The vwf window appears.
- 6 Arrange the signals in the Name pane in logical order. This arrangement allows you to read the output easier.
- 7 Right click on Y and under Value Count Value. Count Value window appears. Click Timing tab and change Count Every value to 20 ns. Click OK.
- 8 Right click on X and under Value Count Value. Count Value window appears. Click Timing tab and change the Multiplied by value to 2. Click OK. If you have another input to set, say in a design that has four inputs, the multiplied by values must be set 2 times higher than the lesser significant input.

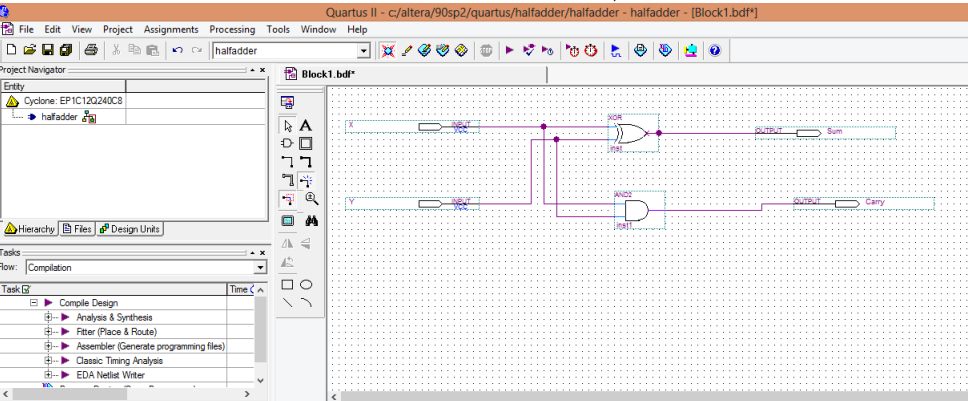
- 9 View Zoom Out as many times as necessary so that you could see all possible combinations of X and Y up to 80 ns.
- 10 Save your HalfAdder.vwf file.
- 11 First, you will do the timing simulation where you can see the signal delay from inputs to outputs (Simulation mode: Timing). Go to Processing Simulator Tool. Simulator Tool window appears. Check the Overwrite simulation input file with simulation results box. Click Start. If you do not see the Start button, close the message window at the bottom.
- 12 If the simulation is successful, click Open to open the vwf file and see the result of simulation.

Now, we will give a pictorial representation of some important steps of this process.

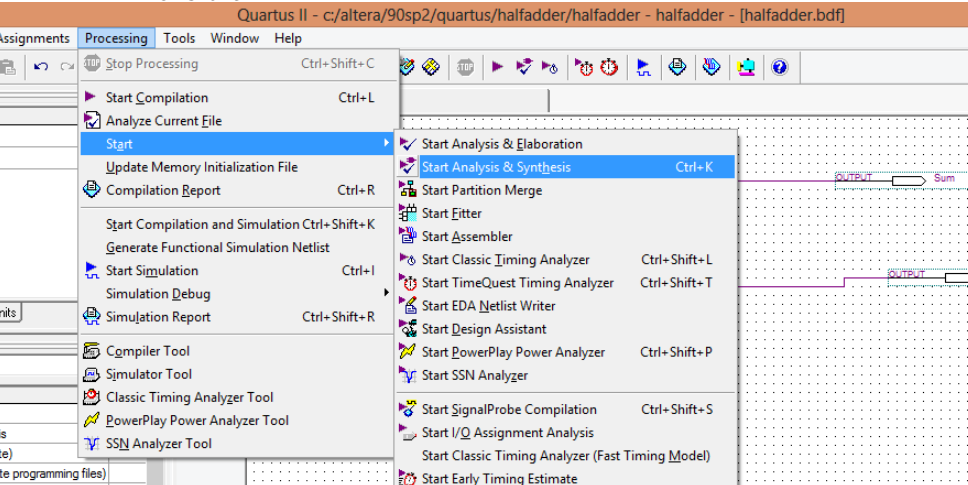
1. This is how we choose the family and the model no. of the FPGA we are working with.



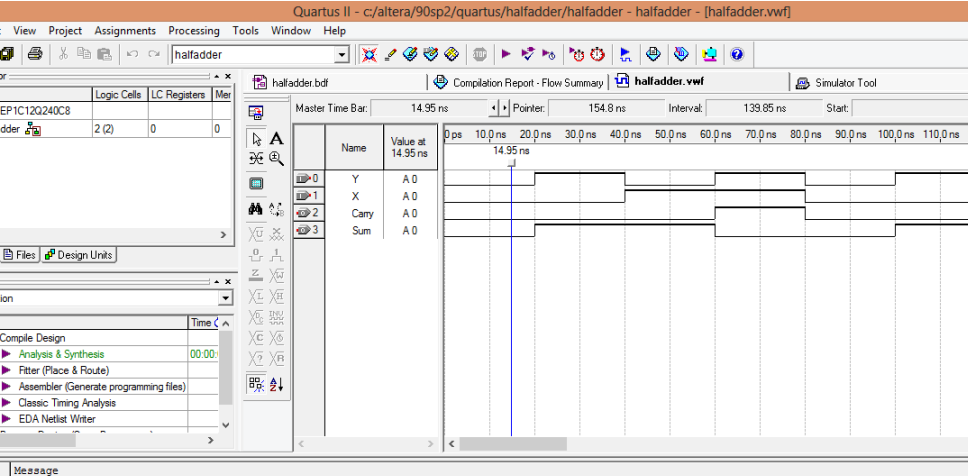
2. This is how we build the block diagram/schematic file.



3. this is how to process it after the block diagram/Schematic file is drawn.



4. after we have done our work with HalfAdder.vwf file ,its time for simulation



5. Compilation is successful

Quartus II - c:/altera/90sp2/quartus/halfadder/halfadder - halfadder

File Project Assignments Processing Tools Window Help

halfadder

Logic Cells LC Registers Mer

EP1C12Q240C8

2 (2) 0 0

Files Design Units

Compile Design

Time

Analysis & Synthesis 00:00

Fitter (Place & Route)

Assembler (Generate programming files)

Classic Timing Analysis

EDA Netlist Writer

Simulation period

☒ Run simulation until all vector stimuli are used

☐ End simulation at: 100 ns

Simulation options

☒ Automatically add pins to simulation output waveforms

☐ Check outputs Waveform Comparison Settings...

☐ Setup and hold time violation detection

☐ Glitch detection

☒ Overwrite simulation

☐ Generate Signal A

☐ Generate VCD File

Quartus II

Simulator was successful

OK

Start Stop Open Report

Message

Info: Option to preserve fewer signal transitions to reduce memory requirements is enabled

Info: Simulation partitioned into 1 sub-simulations

Info: Simulation coverage is 100.00 %

- Getting familiar with VHDL
- Basic structure of a VHDL file
- Entity Declaration
- Generic
- Architecture Body item Behaviour Model

Getting familiar with VHDL

VHDL is a language for describing electronic systems. It arose out of the United States Government's Very High Speed Integrated Circuits (VHSIC) program, initiated in 1980. In the course of this program, it became clear that there was a need for a standard language for describing the structure and function of integrated circuits (ICs). Hence the VHSIC Hardware Description Language (VHDL) was developed, and subsequently adopted as a standard by the Institute of Electrical and Electronic Engineers (IEEE) in the US. There are some important differences between VHDL and a conventional procedural language. VHDL is inherently parallel, i.e.; commands, which correspond to logic gates, are executed in parallel, as soon as a new input arrives.

Basic structure of a VHDL file

A digital system in VHDL consists of a design entity that can contain other entities that are then considered components of the top level entity. Each entity is modelled by an entity declaration and an architecture body. Each entity can be considered as the interface to the outer world that defines the input and output signals. The architecture body contains the description of the entity and is composed of interconnected entities, processes and components, all operating concurrently.

The entity declaration defines the NAME of the entity and lists the input and output ports. The general form is as follows:-

Example

```
entity NAME_OF_ENTITY is [ generic
generic_declarations);]
  port (signal_names : mode type;
        signal_names : mode type
        :
        signal_names: mode type)
  end[name_of_entity];
```

An entity always starts with the keyword `entity`, followed by its name and the keyword `is`. Next are the port declarations using the keyword `port`. An entity declaration always ends with the keyword `end`, optionally `[]` followed by the name of the entity.

- The `NAME_OF_ENTITY` is a user-selected identifier.
- `signal_names` consists of a comma separated list of one or more user-selected identifiers that specify external interface signals.
- `mode`: is one of the reserved words to indicate the signal direction :
 - a). `In`:-Indicates that the signal is an input
 - b). `out`:-indicates that the signal is an output of the entity whose value can only be read by other entities that use it.

- c). `buffer`: Indicates that the signal is an output of the entity whose value can be read inside the entity's architecture.
- d). `inout`: The signal can be an input or an output.
 - `type`:- a built-in or user-defined signal type. Examples of types are `bit`, `bit_vector`, `Boolean`, `character`, `std_logic`, and `std_ulogic`.
- a). `Bit`:- can have the value 0 and 1
- b). `Bit_vector`:- is a vector of bit values (e.g. `bit_vector (0 to 7)`)
- c). `std_logic`, `std_ulogic`, `std_logic_vector`, `std_ulogic_vector`: can have 9 values to indicate the value and strength of a signal. `Std_ulogic` and `std_logic` are preferred over the `bit` or `bit_vector` types
- d). `boolean`: can have the value `TRUE` and `FALSE`

- e). integer: can have a range of integer values
- f). real: can have a range of real values
- g). character: any printing character
- h). time: to indicate time

Generic

generic declarations are optional and determine the local constants used for timing and sizing (e.g. bus widths) the entity. A generic can have a default value. The syntax for a generic follows:-

```
generic (  
    constant_name:    type [:=value] ;  
    constant_name:    type [:=value] ;  
    :  
    constant_name:    type [:=value] );
```

Example

Four-to-one multiplexer of which each input is an 8-bit word.

entity mux4_to_1 is

```
    port (I0,I1,I2,I3:  in std_logic_vector(7 downto 0);
```

```
          SEL: in std_logic_vector (1 downto 0);
```

```
          OUT1: out std_logic_vector(7 downto 0));
```

```
    end mux4_to_1;
```

Example

An example of the entity declaration of a D flip-flop with set and reset inputs is:

```
entity dff_sr is
  port (D,CLK,S,R: in std_logic;
        Q,Qnot: out std_logic);
end dff_sr;
```

Architecture Body

The architecture body specifies how the circuit operates and how it is implemented. As discussed earlier, an entity or circuit can be specified in a variety of ways, such as behavioral, structural (interconnected components), or a combination of the above. The architecture body looks as follows:-

```
architecture architecture_name of NAME_OF ENTITY  
is
```

- declaration
 - components declarations
 - signal declarations
 - constant declarations
 - function declarations
 - procedure declarations

```
:  
  
Begin  
——statements
```

```
:  
  
end architecture_name;
```

Behaviour Model

The architecture body described at the behavioral level, is given below:-

architecture behavioral **of** BUZZER **is**
begin

WARNING <= (**not** DOOR **and** IGNITION) or (**not** SBELT **and**
IGNITION);
end behavioral;

Example

The behavioral description of a two-input AND gate:

entity AND2 **is**

port (in1, in2:**in** std_logic;

 out1: **out** std_logic);

end AND2;

architecture behavioral_2 **of** AND2 **is**

begin

 out1 <= in1 **and** in2;

end behavioral_2;

Circuits can also be described using a structural model that specifies what gates are used and how they are interconnected. The following example illustrates it.

architecture structural of BUZZER is

—— Declarations

component AND2

port in1, in2: **in** std_logic;
out1: **out** std_logic);

end component;

component OR2

port (in1, in2: **in** std_logic;
out1: **out** std_logic);

end component;

component NOT1

port (in1: **in** std_logic;

```
out1: out std_logic);  
end component;
```

——Declaration of signals used to interconnect gates

```
signal DOOR_NOT, SBELT_NOT, B1, B2:  std_logic;
```

```
begin
```

——Component instantiations statements

```
signal DOOR_NOT, SBELT_NOT, B1, B2:  std_logic;  
begin  
    —— Component instantiations statements  
    U0: NOT1 port map (DOOR, DOOR_NOT);  
    U1: NOT1 port map (SBELT, SBELT_NOT);  
    U2: AND2 port map (IGNITION, DOOR_NOT, B1);  
    U3: AND2 port map (IGNITION, SBELT_NOT, B2);  
    U4: OR2 port map (B1, B2, WARNING);  
end structural;
```

Working with PCB

- What is A PCB?
- What is a PCB board Design?
- Our work with PCB designing softwares
- Working on some basic circuits and Decoder Matrix ROM

What is a PCB?

- Printed circuit boards are an insulating material used as a base, into which conductive strips are printed.
- The base material is generally fiberglass, and the conductive connections are generally copper and are made through an etching process.
- The main PCB board is called the motherboard, the smaller attachment PCB boards are called daughter boards or daughter cards.

What is PCB Board Design?

- PCB board design defines the electrical pathways between components.
- It is derived from a schematic representation of the circuit.
- When it is derived, or imported from a schematic design, it translates the schematic symbols and libraries into physical components and connections.

Our Work with PCB designing softwares

We have just started our work with PCB designing softwares. We hope that in our next semester, we will continue our work with it. We have used two softwares for our elementary work with PCBs.

- **Liveware:-** A sophisticated software package for designing and simulating electronic circuits. Switches, transistors, diodes, integrated circuits and hundreds of other components can all be connected together to investigate the behaviour of a circuit. There are no limits to what can be designed and no loose connections or faulty components to worry about. However, if the maximum ratings for any components are exceeded, they will explode on screen!
- **PCB Wizard:-** a powerful package for designing single-sided and double-sided printed circuit boards (PCBs).

It provides a comprehensive range of tools covering all the traditional steps in PCB production, including schematic drawing, schematic capture, component placement, automatic routing, Bill of Materials reporting and file generation for manufacturing. In addition, PCB Wizard offers a wealth of clever new features that do away with the steep learning curve normally associated with PCB packages.

What we do that we design our circuit in liveware. Then we simulate the circuit. Then by using PCB wizard, we convert the designed circuit to a PCB layout.

Working on some basic circuits and Decoder Matrix ROM

We have just started our work with PCB designing softwares. So far we have designed some basic circuits using 555 timer ic and a decoder matrix ROM.

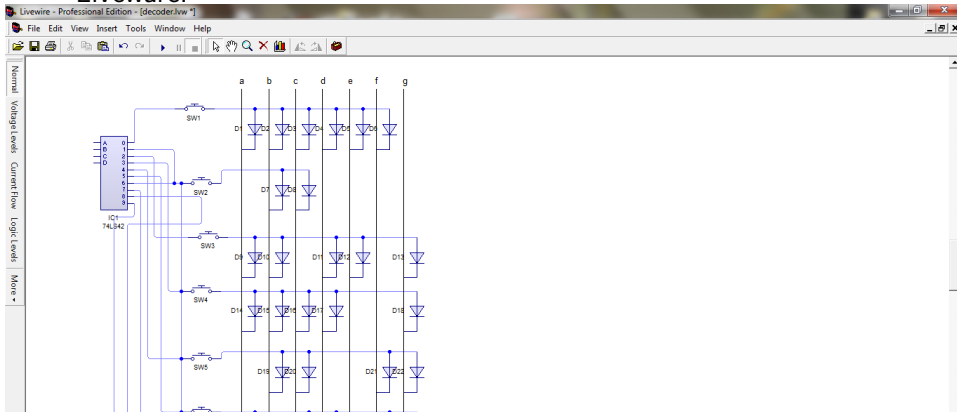
First, we will provide the basic steps of designing a Decoder matrix Rom. Then we will give a pictorial representation of the process.

- 1 Open the liveware software. The components to draw the desired circuit is available in the liveware. Draw the circuit diagram by choosing the right ic , resistor properties , diode properties etc.
- 2 After you have drawn the circuit, simulate it using the run button or by pressing F9.

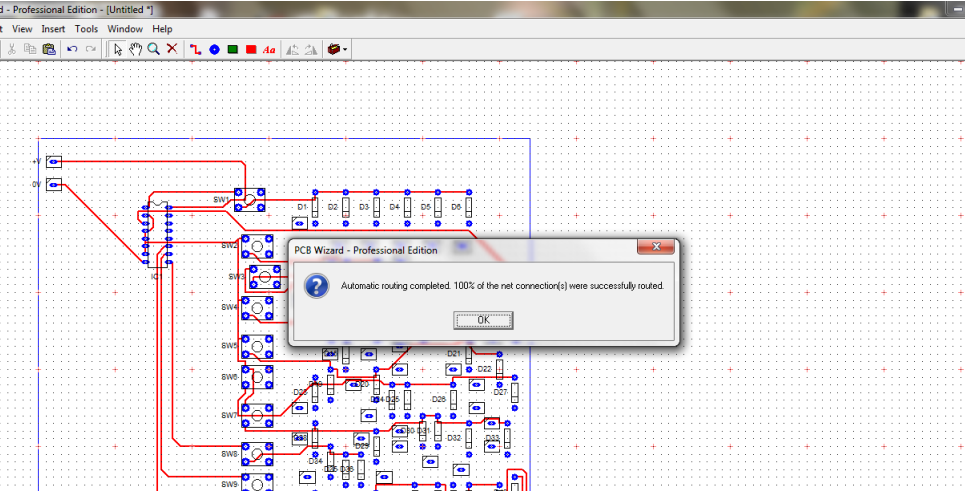
- ③ Before converting it into PCB, make sure that the PCB wizard window is open.
- ④ Then choose tool - Convert - Design to Printed Circuit Board from Liveware.
- ⑤ After the Convert to Printed Circuit Board is open, choose the option No, let liveware specify those options for me.
- ⑥ Then you will have to design the respective PCB in the PCB wizard window.

Pictorial Representation Of the Process

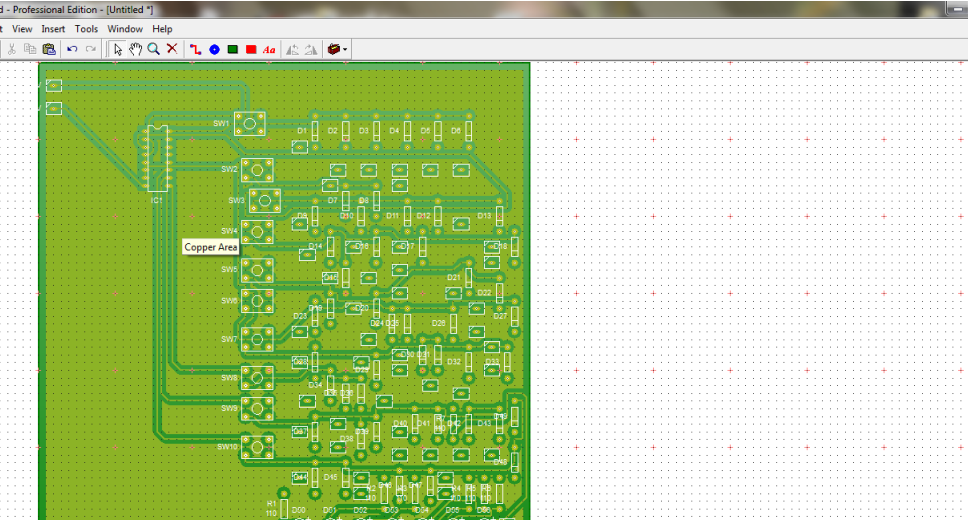
The circuit diagram of the decoder matrix ROM drawn with Liveware.



Creating the PCB with respect to the circuit diagram you have created in Liveware.



The final design of the respective PCB



Future scope of work

After studying the Quartus 2 software and working with FPGA , we are confident of being able to design more complex circuits and other required applications. We are now capable of studying these devices in further depths and actually using or implementing these using practical applications. The future scope of work is immense. Our goal is to study the methods of designing a PCB using a PCB drawing software and build a board similar to that of an FPGA containing interface that we used in this mini project.

- ➊ VHDL primer by Jan Van der Spiegel. University of Pennsylvania
- ➋ VHDL cookbook by Peter J Ashenden
- ➌ Latex- A document preparation system, User's guide and reference manual-Leslie Lamport
- ➍ www.google.co.in
- ➎ www.wikipedia.org
- ➏ www.wikibooks.org