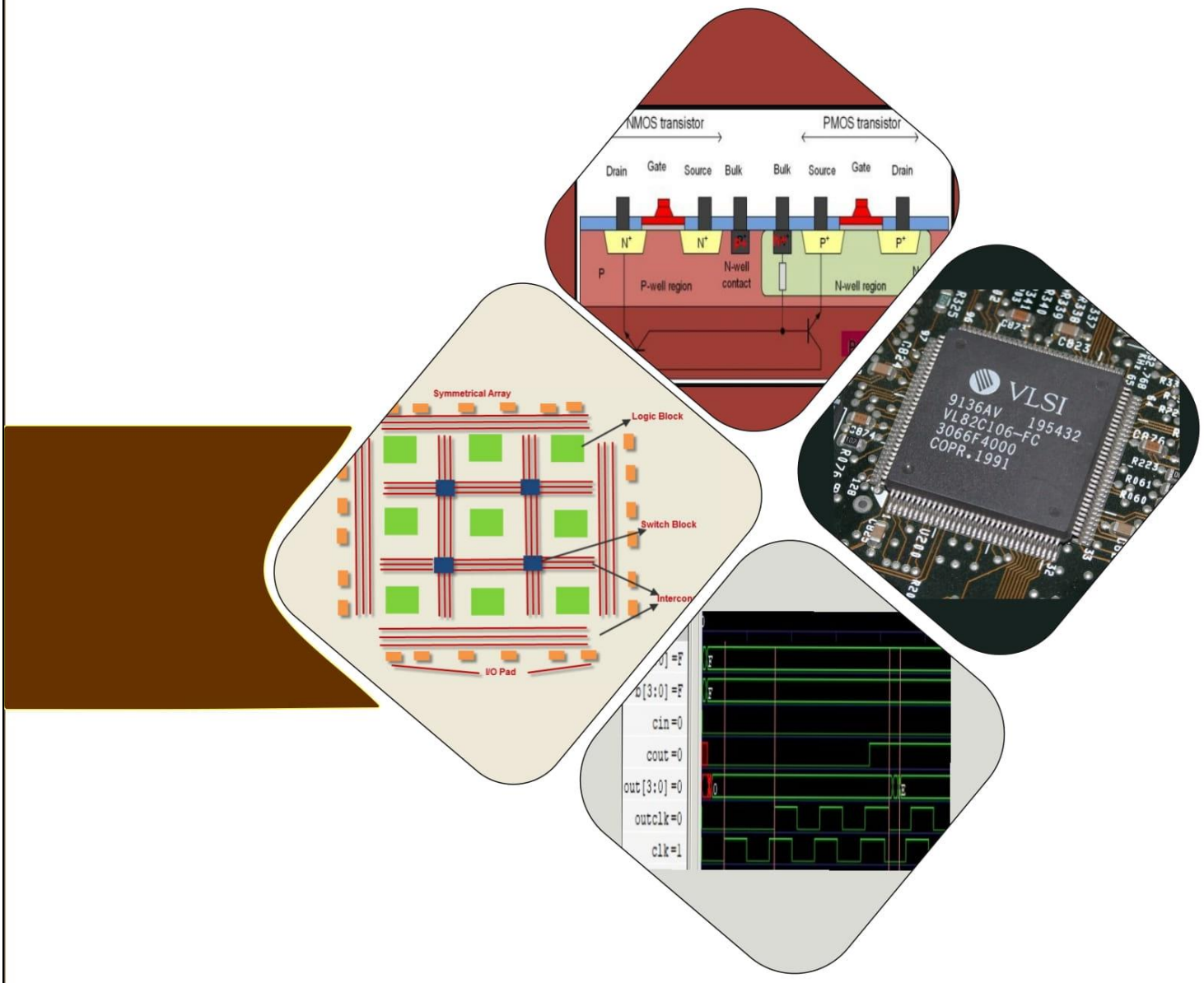
SCHEME :K

Name :_____
Roll No.:_____ Year : 20 ___ 20 __
Exam Seat No. :_____

LABORATORY MANUAL FOR

VLSI APPLICATIONS (316340)



ELECTRONICS ENGINEERING GROUP

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI
(Autonomous)(ISO21001:2018)(ISO/IEC27001:2013)

VISION

To ensure that the Diploma Level Technical Education constantly matches the latest requirements of technology and industry and includes the all-round personal development of students including social concerns and to become globally competitive, technology led organization.

MISSION

To provide high quality technical and managerial manpower, information and consultancy services to the industry and community to enable the industry and community to face the changing technological and environmental challenges.

QUALITY POLICY

We, at MSBTE, are committed to offer the best-in-class academic services to the students and institutes to enhance the delight of industry and society. This will be achieved through continual improvement in management practices adopted in the process of curriculum design, development, implementation evaluation and monitoring system along with adequate faculty development Programs.

CORE VALUES

MSBTE believes in the followings:

- Skill development in line with industry requirements.
- Industry readiness and improved employability of Diploma holders.
- Synergistic relationship with industry.
- Collective and cooperative development of all stake holders.
- Technological interventions in societal development.
- Access to uniform quality technical education.

A Laboratory manual

for

VLSI Applications

(316340)

Semester – VI

(AO/DE/EJ/ET/EX/IE)

Maharashtra State
Board of Technical Education, Mumbai

(Autonomous) (ISO 21001:2018) (ISO/IEC 27001:2013)

Maharashtra State
Board of Technical Education, Mumbai
(Autonomous) (ISO 21001:2018) (ISO/IEC 27001:2013)

4th Floor, Government Polytechnic Building, 49, Kherwadi,
Bandra (East), Mumbai – 400051.

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION

Certificate

This is to certify that Mr./Ms. ……. Roll No. ……. Of Sixth Semester of Diploma in ...…… of Institute ……... (Code: ……) has attained pre-defined practical outcomes (PROs) satisfactorily in course **VLSI Applications (316340)** for the academic year 20…… to 20…… as prescribed in the curriculum.

Place: …… Enrollment No.: ……

Date: …… Exam Seat No.: ……

Course Teacher **Head of Department** **Principal**

Seal Of Institute

Preface

The primary focus of any engineering laboratory/field work in the technical education system is to develop the much-needed industry relevant competencies and skills. With this in view, MSBTE embarked on this innovative 'K' Scheme curricula for engineering diploma programs with outcome- based education as the focus and accordingly, a relatively large amount of time is allotted for the practical work. This displays the great importance of laboratory work, making each teacher, instructor and student realize that every minute of the laboratory time needs to be effectively utilized to develop these outcomes, rather than doing other mundane activities. Therefore, for the successful implementation of this outcome-based curriculum, every practical has been designed to serve as a ***'vehicle'*** to develop this industry identified competency in every student. The practical skills are difficult to develop through 'chalk and duster' activity in the classroom situation. Accordingly, the 'K' scheme laboratory manual development team designed the practicals to ***focus*** on the ***outcomes***, rather than the traditional age-old practice of conducting practical's to 'verify the theory' (which may become a byproduct along the way).

This laboratory manual is designed to help all stakeholders, especially the students, teachers and instructors to develop in the student the predetermined outcomes. It is expected from each student that at least a day in advance, they have to thoroughly read through the concerned practical procedure that they will do the next day and understand the minimum theoretical background associated with the practical. Every practical in this manual begins by identifying the competency, industry relevant skills, course outcomes and practical outcomes which serve as a key focal point for doing the practical. The students will then become aware about the skills they will achieve through the procedure shown there and necessary precautions to be taken, which will help them to apply in solving real-world problems in their professional life.

This manual also provides guidelines to teachers and instructors to effectively facilitate student- centered lab activities through each practical exercise by arranging and managing necessary resources in order that the students follow the procedures and precautions systematically ensuring the achievement of outcomes in the students.

VLSI (Very Large Scale Integration) technology plays a pivotal role in the design and development of modern electronic systems, enabling the integration of millions of transistors onto a single chip. It forms the foundation of advanced processors, memory devices, digital circuits, and system-on-chip (SoC) architectures used in computing, communication, automation, and consumer electronics. Understanding the principles of MOS devices, CMOS technology, digital logic design, and chip-level implementation is essential for aspiring engineers in the fields of electronics, communication, and embedded systems.

Although best possible care has been taken to check for errors (if any) in this laboratory manual, perfection may elude us as this is the first edition of this manual. Any errors and suggestions for improvement are solicited and highly welcome.

Program Outcomes (POs)

Following program outcomes are expected to be achieved through the practical of the course.

PO1: Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the broad-based Electronics Engineering group program problems.

PO2: Problem analysis: Identify and analyze well-defined Electronics Engineering group program problems using codified standard methods.

PO3: Design/ development of solutions: Design solutions for broadly-defined technical problems and assist with the design of Electronics Engineering group program systems components or processes to meet specified needs.

PO4: Engineering Tools, Experimentation and Testing: Apply modern Electronic Engineering group program tools and appropriate technique to conduct standard tests and measurements.

PO5: Engineering practices for society, sustainability and environment: Apply appropriate Electronics Engineering group program technology in context of society, sustainability, environment and ethical practices.

PO6: Project Management: Use Electronics Engineering group program management principles individually, as a team member or a leader to manage projects and effectively communicate about well- defined engineering activities.

PO7: Life-long learning: Ability to analyze individual needs and engage in updating in the context of Electronics Engineering group program technological changes.

List of Industry Relevance Skills

The aim of this course is to attend following industry/employer excepted outcome through various teaching learning experiences:

- Develop VLSI-based electronic circuit/component using VHDL
- Ability to write, simulate, and debug hardware description language programs-essential for FPGA/CPLD design and ASIC development.
- Using tools like Xilinx ISE, Vivado, ModelSim to verify design correctness before hardware implementation – a Key step in professional VLSI workflows.
- Hands-on experience programming and testing designs on FPGA/CPLD trainer kits bridges the gap between simulation and real hardware.

Practical - Course Outcome matrix

Course Level Learning Outcomes (COs)

CO1 - Interpret CMOS technology circuits and its applications.

CO2 - Develop digital circuits on CPLD and FPGA devices

CO3 - Use VHDL to develop and test digital circuits.

CO4 - Develop VHDL program for given application.

CO5 - Interpret VHDL simulation and synthesis

Sr. No.	Title of the Practical	CO 1	CO 2	CO 3	CO 4	CO 5
1.	*Identification of internal block and pin configuration of FPGA and CPLD	-	✓	-	-	-
2.	*Installation of EDA tool and the relevant libraries for VLSI code development	-	-	✓	-	-
3.	*Develop VHDL code for basic and universal gate for data flow model	-	-	✓	-	-
4.	Develop VHDL code for basic and universal gate for behavioral model	-	-	✓	-	-
5.	*Realize the half and full Adder on FPGA board	-	-	✓	-	-
6.	*Realize the Multiplexer on FPGA board	-	-	✓	-	-
7.	Realize the De-multiplexer on FPGA board	-	-	✓	-	-
8.	Design 4:2 encoder on FPGA board	-	-	✓	-	-
9.	Design 3:8 decoder on FPGA board	-	-	✓	-	-
10.	*Realize the D and T flipflop on FPGA board	-	-	✓	-	-
11.	Design Comparator on FPGA board	-	-	✓	-	-
12.	Design Up Counter on FPGA board	-	-	✓	-	-
13.	Design Synchronous counter on FPGA board	-	-	✓	-	-
14.	Design binary to gray code converter circuit using FPGA board	-	-	✓	-	-
15.	*Design digital to analog converter (DAC) using FPGA board	-	-	-	✓	-
16.	*Design stepper motor Controller using FPGA board	-	-	-	-	✓
17.	Design of 4-bit ALU/ sequence detector using FPGA board	-	-	-	-	✓

Guidelines to Teachers

1. Teacher should provide the guideline with demonstration of practical to the students with all features.
2. Teacher shall explain prior concepts to the students before starting of each practical.
3. Involve students in the performance of each experiment.
4. Teacher should ensure that the respective skills and competencies are developed in the students after the completion of the practical exercise.
5. Teachers should give opportunities to students for hands-on experience after the demonstration.
6. Teacher is expected to share the skills and competencies to be developed in the students.
7. Teacher may provide additional knowledge and skills to the students even though not covered in the manual but are expected of the students by the industry.
8. Finally give practical assignments and assess the performance of students based on tasks assigned to check whether it is as per the instructions.
9. Teacher is expected to refer complete curriculum document and follow guidelines for implementation
10. At the beginning of the practical which is based on the simulation, teacher should make the students acquainted with required simulation software environment.
11. Teachers should use the projector/smart board effectively by preparing materials in advance, ensuring clear visibility, engaging students interactively, and maintaining proper digital discipline and equipment safety.

Instructions for Students

1. Listen carefully to the lecture given by the teacher about course, curriculum, learning structure, skills to be developed.
2. Organize the work in the group and make a record of all observations.
3. Do the calculations and plot the graph wherever it is required in the practical
4. Students shall develop maintenance and safety skills as expected by industries.
5. Student shall attempt to develop related hand-on skills and gain confidence.
6. Student shall develop the habits of evolving more ideas, innovations, skills etc. those included in scope of manual
7. Student should develop the habit to submit the practical on date and time.
8. Student should prepare well while submitting a write-up of exercise.

Content Page

List of Practical's and Progressive Assessment Sheet

Note: Out of above suggestive LLOs –

- '*' Marked Practical's (LLOs) Are mandatory.
- Minimum 80% of above list of lab experiment are to be performed.
- Judicial mix of LLOs are to be performed to achieve desired outcomes.

Practical No.1: *Identification of internal block and pin configuration of FPGA and CPLD

I Practical Significance

A Field Programmable Gate Array (FPGA**)** is a reconfigurable digital device consisting of an array of logic cells and programmable interconnects, all of which are fully controlled by the user. This flexibility allows designers to implement, modify, and optimize digital circuits according to specific application requirements. Similarly, a Complex Programmable Logic Device (CPLD) is an integrated circuit used to design and implement digital hardware systems such as those found in mobile devices and embedded applications. Through this practical, students will gain hands-on experience in identifying various functional blocks, architecture, and pin configurations of FPGA and CPLD, thereby understanding their roles in digital system design.

II Industry/Employer Expected Outcome

The aim of this course is to attend following industry/employer excepted outcome through various teaching learning experiences:
"Develop VLSI-based electronic circuit/component using VHDL."

III Course Level Learning Outcome

Develop digital circuits on CPLD and FPGA devices.

IV Laboratory Learning Outcomes

- LLO 1.1 Identify various blocks of FPGA and CPLD.
- LLO 1.2 Test the functionality of various pins of FPGA and CPLD.

V Relevant Affective Domain related outcomes

- Demonstrate in exploring and implementing digital circuits using CPLD and FPGA platforms.
- Practice good housekeeping.
- Exhibit responsibility while handling programmable logic devices and related laboratory equipment.

VI Relevant Theoretical Background

Field-programmable gate array (FPGA): A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing, hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare).
FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together" – somewhat like many logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complex blocks of memory

Complex Programmable Logic Device (CPLD): A Complex Programmable Logic Device (CPLD) is a combination of a fully programmable AND/OR array and a bank of microcells. The AND/OR array is reprogrammable and can perform a multitude of logic functions. Microcells are functional blocks that perform combinatorial or sequential logic, and also have the added flexibility for true or complement, along with varied feedback paths

VII a. Suggested block diagram:

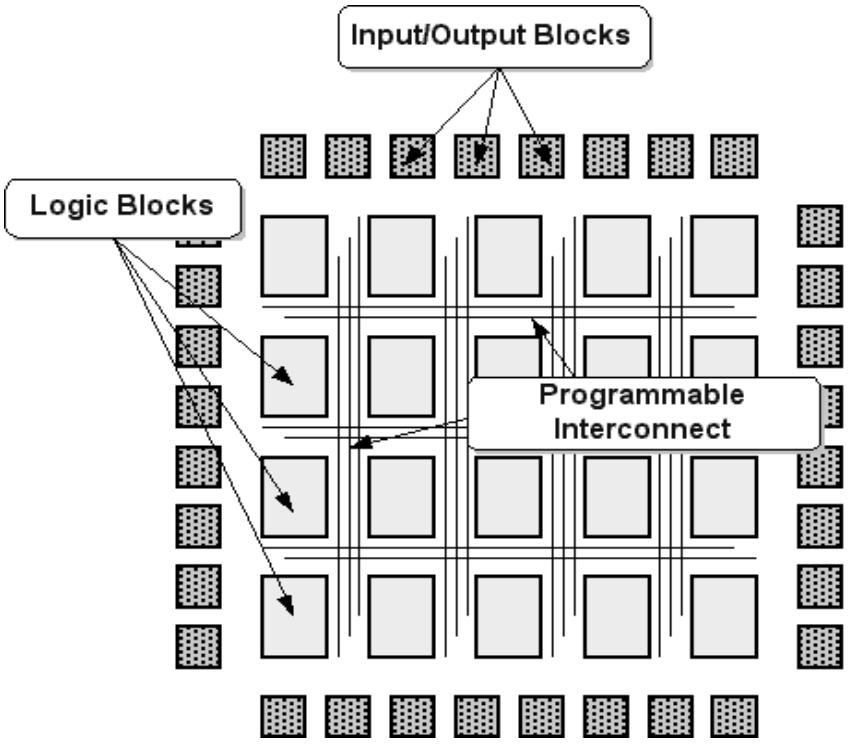


Fig. no. 1.1 Block diagram of FPGA Development board

.

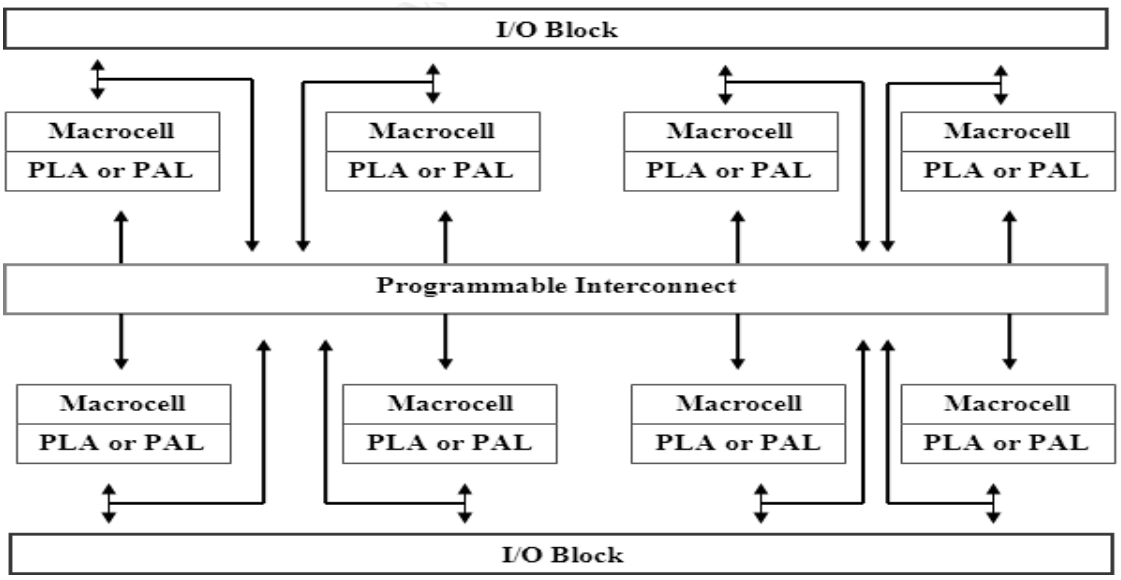


Fig. no. 1.2 Block Diagram of CPLD

b. Actual block diagram used in laboratory with related equipment rating.

- **CPLD Development board:**

- **FPGA Development board:**

VIII Required Resources/apparatus/equipment with specifications

Table 1.1 Required Resources

S.N.	Instrument / Components	Specification	Quantity
1	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208),On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit., On board, 2 Crystal 8MHz and 25MHz. .JTAG Interface (Boundary Scan),PROM Interface (XCF02S),40 pin, 4 header connector for external I/O's	1 No.
2	Desktop PC	Personal computer with latest configuration. Loaded with open-source IDE, simulation and program downloading software, UPS and Antivirus.	1 No.
3	CPLD	CPLD Development Board (e.g., Xilinx XC9500, Altera MAX7000, , Power Supply (5V / 3.3V), Connecting Wires / Jumpers	

IX Precautions to be followed

1. Do not power up the development board when identifying the components on the board.
2. Refer data sheets for the given development board.

X Procedure

- Place the kit on the ESD mat and connect nothing to it. Inspect the board visually.
- Locate and note labels printed on PCB: device part number, JTAG header, VCC/VCCIO pins, GND, CLK, LED groups, switches. Photocopy or take a photo of the board labels for the record.
- Find the board's pinout silkscreen and compare with the device datasheet and board user manual. Note any discrepancies.
- With power off, use the multimeter in continuity mode to confirm ground plane: probe board GND silkscreen and other GND points (e.g., connector shell). Record locations.
- Identify VCC, VCCIO, VCCINT pins from datasheet / silkscreen. With power off, do not apply any voltage checks beyond continuity. Document the supply voltages required (e.g., 3.3V, 2.5V).

XI Resources

Table 1.2 Resources

Sr. No.	Name of Resource	Specifications	Quantity
1			
2			
3			

XII Actual Procedure

……

……

……

……

……

……

XIII Observation Table

Observe the pin out diagram for FPGA and give the functions of following pins:

Table 1.3 Functions of FPGA Trainer Kit pins

Sr No	Pins	Functions
1.	JTAG	
2.	DCI	
3.	DUAL	
4.	VCCO	
5.	GCLK	

Table 1:4 Functions of CPLD Trainer Kit pins

Sr No.	Pins	Functions
1	VCC	
2	GND	
3	I/O Pins	
4	CLK	
5	RESET	
6	JTAG	
7	OE (Output Enable)	
8	TDI / TDO / TMS / TCK	
9	VCCINT / VCCIO	
10	GCLK	

XIV Result :

……

……

XV Interpretation of results:

……

……

XVI Conclusion and recommendation:

……

……

XVII Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Explain the process of assigning input and output pins in CPLD/FPGA devices.
2. Describe the procedure used to test the functionality of various pins of FPGA/CPLD.
3. Give one real-life example where FPGA/CPLD is used.

[Space for Answers] (If required attach separate page)

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

……
……
……
……
……

XVIII References/Suggestions for further reading

Table 1:5 References

Sr . No	Link / Portal	Description
1	https://www.youtube.com/watch?v=gCAYY0fHPq4andt=329s	NPTEL IIT Kharagpur Lecture 60: PAL, PLA, CPLD, FPGA
2	CPLD Datasheet Xilinx DS054 XC9500XL High-Performance CPLD Family Data Sheet, Data Sheet	XC9500XL High-Performance CPLD Family Data Sheet
3	FPGA Datasheet Agilex™ 3 FPGAs and SoCs C-Series Product Table	Agilex™ 3 FPGAs and SoCs C-Series Product Table

XIX Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		**60 %**
1	Handling of the components	10%
2	Identification of components	20%
3	Connections between the components	20%
4	working in teams	10%
Product Related: 10 Marks		**40%**
5	Interpretation of result	10%
6	Conclusion	10%
7	Practical related questions	15%
8	Submitting the journal in time	5%
Total (25 Marks)		**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.2: *Installation of EDA tool and the relevant libraries for VLSI code development

I Practical Significance

The Electronic Design Automation (EDA) industry develops software to support engineers in the creation of new integrated-circuit (IC) designs. Due to the high complexity of modern designs, EDA facilitates almost every aspect of the IC design flow, from high-level system design to fabrication. EDA addresses designers' needs at multiple levels of electronic system hierarchy, including integrated circuits (ICs), multi-chip modules (MCMs), and printed circuit boards (PCBs). This practical develop skill for EDA software tools installation

II Industry/Employer Expected Outcome

The aim of this course is to attend following industry/employer excepted outcome through various teaching learning experiences:
"Develop VLSI-based electronic circuit/component using VHDL."

III Course Level Learning Outcome

Develop digital circuits on CPLD and FPGA devices.

IV Laboratory Learning Outcomes

- LLO 2.1 Install relevant EDA (such as Xilinx software) tool for VHDL.
- LLO 2.2 Check the VHDL libraries installed in VHDL environment.

V Relevant Affective Domain related outcomes

- Maintain accuracy and attention to detail while checking VHDL libraries.
- Exhibit willingness and curiosity to explore new EDA tools and VHDL environments.
- Follows good lab practices, respects licensing terms, and promotes responsible software handling.

VI Relevant Theoretical Background

Xilinx ISE (Integrated Synthesis Environment) is a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer.
Xilinx ISE is a design environment for FPGA products from Xilinx, and is tightly-coupled to the architecture of such chips, and cannot be used with FPGA products from other vendors. The Xilinx ISE is primarily used for circuit synthesis and design, while ISIM or the Model Sim logic simulator is used for system-level testing. Other components shipped with the Xilinx ISE include the Embedded Development Kit (EDK), a Software Development Kit (SDK) and Chip Scope Pro.

VII Suggested block diagram: ------- NA -------

VIII Required Resources/apparatus/equipment with specifications

Table 2.1 Required Resources

No	Instrument / Components	Specification	Quantity
1	Desktop PC	Personal computer with latest configuration. Loaded with open-source IDE, simulation and program downloading software, UPS and Antivirus.	01

IX Precautions to be followed

Follow steps in sequence to execute system properly**.**

X Procedure

Steps for installing Xilinx Software (EDA Tool)

1. Click on the setup file to install Xilinx GUI icon

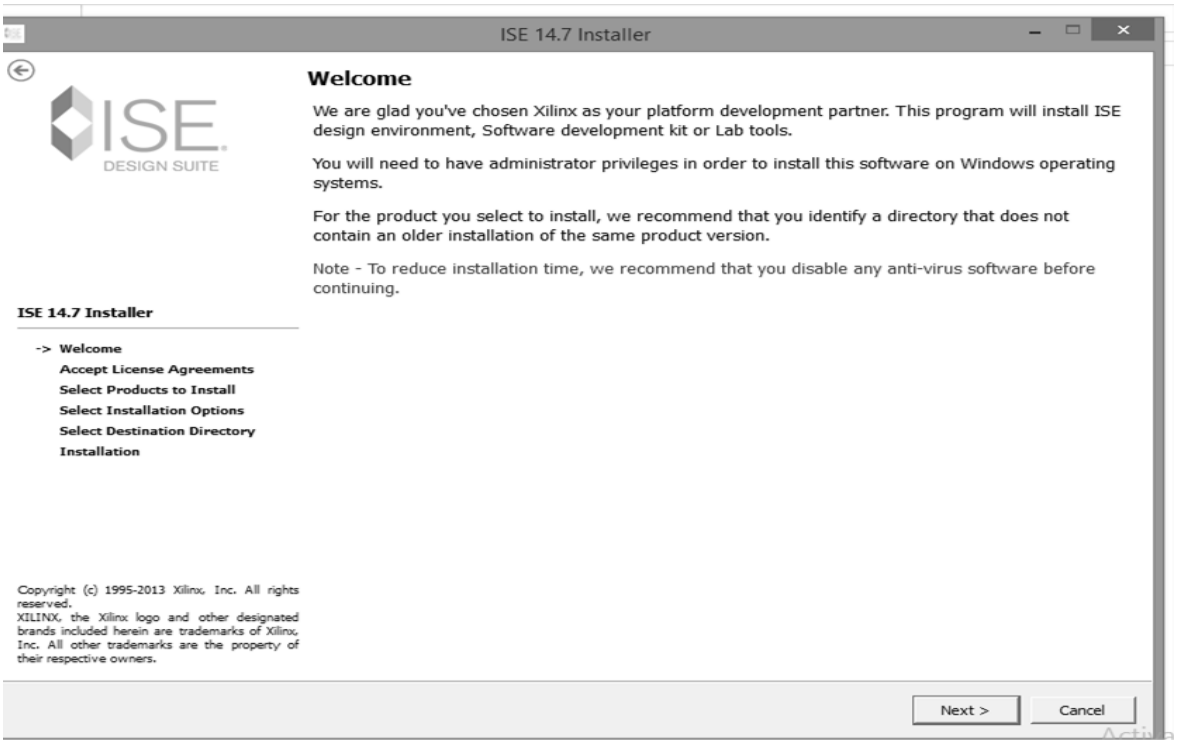


Fig. no. 2.1: Xilinx Installer Welcome Screen.

2. Click next on the Xilinx Agreement window.

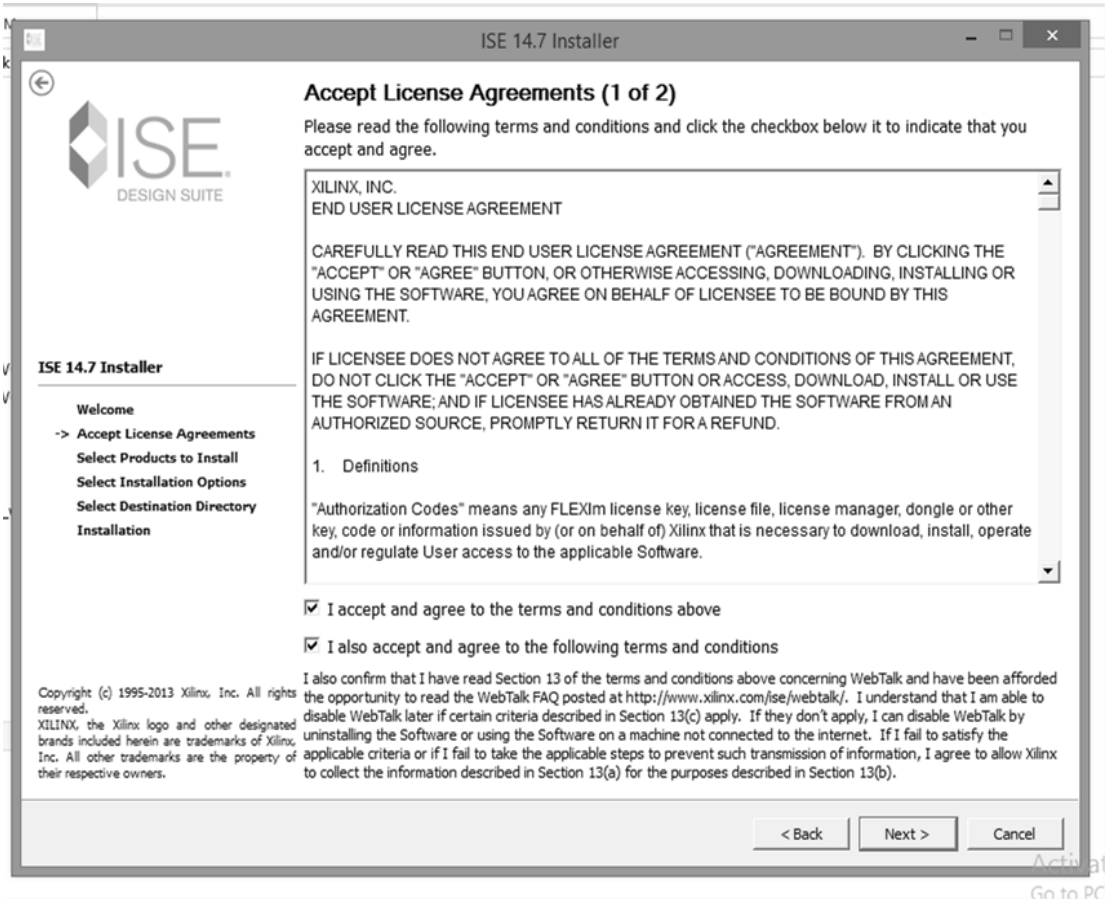


Fig. no. 2.2 a: Xilinx License Agreement window

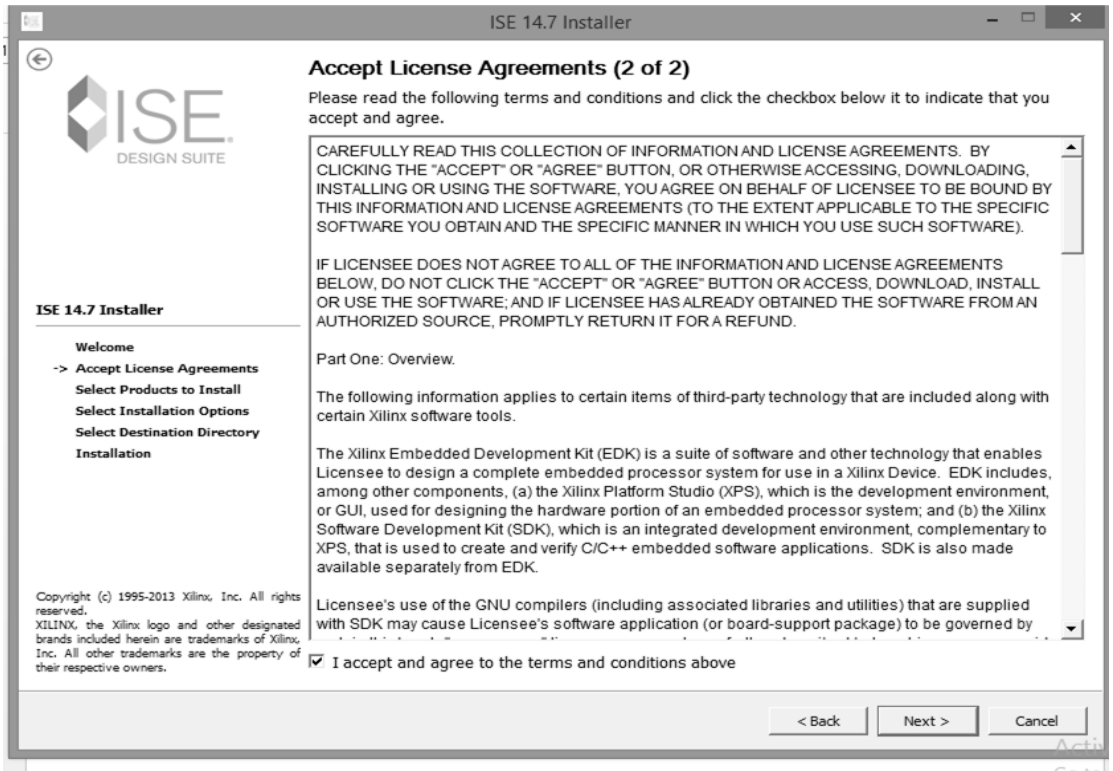


Fig. no. 2.2b: Xilinx License Agreement window contd…

3. Select the products to install.

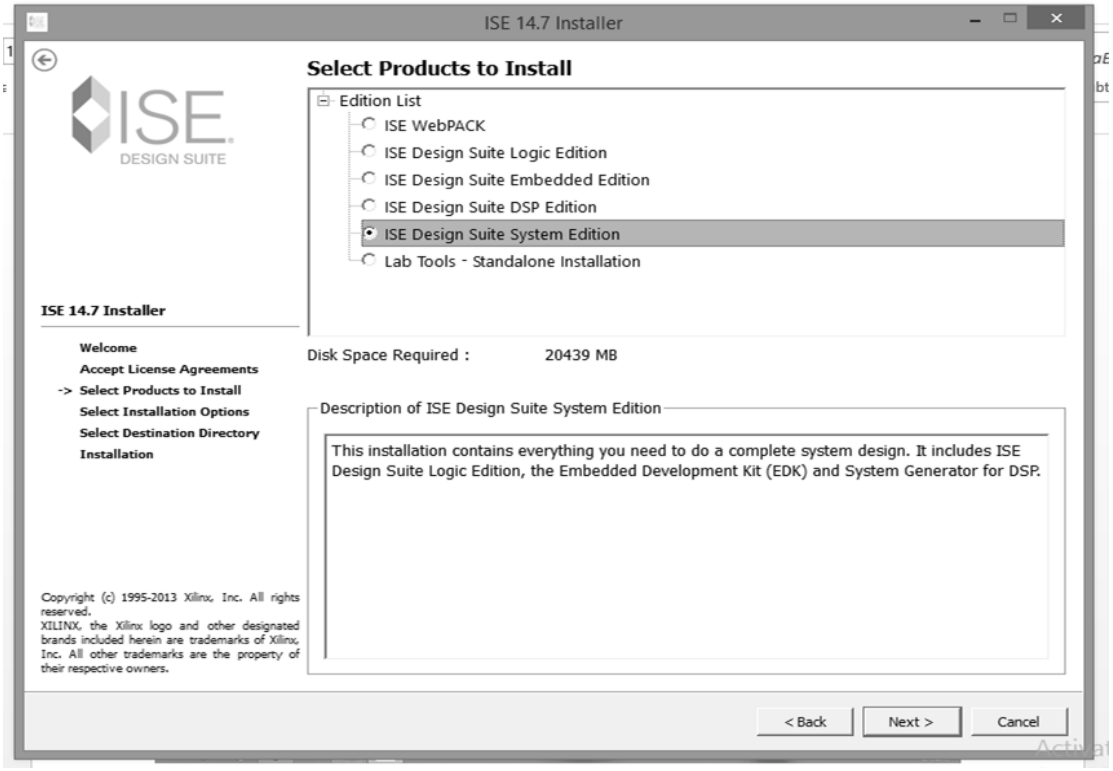


Fig. no. 2.3: Select products to install.

4. Select the Installations options.

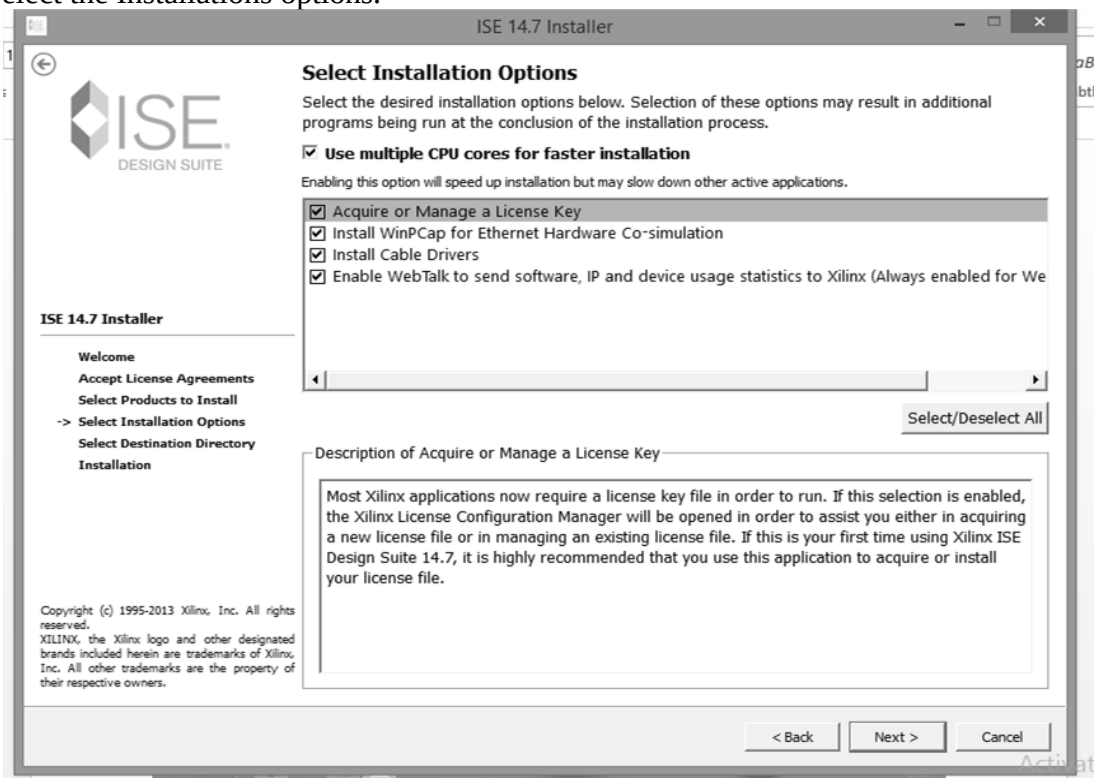


Fig. no. 2.4: Installations options

5. Select the destination path.

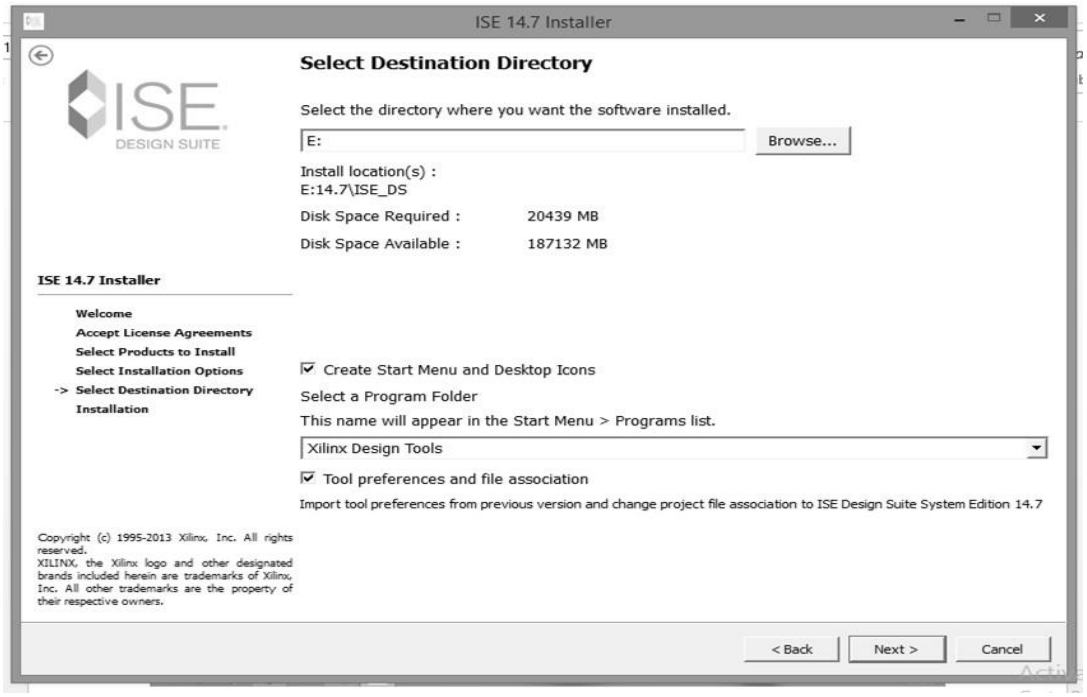


Fig. no. 2.5: Select Destination path

6. Click Install on the Installation Summary window to start the installation.

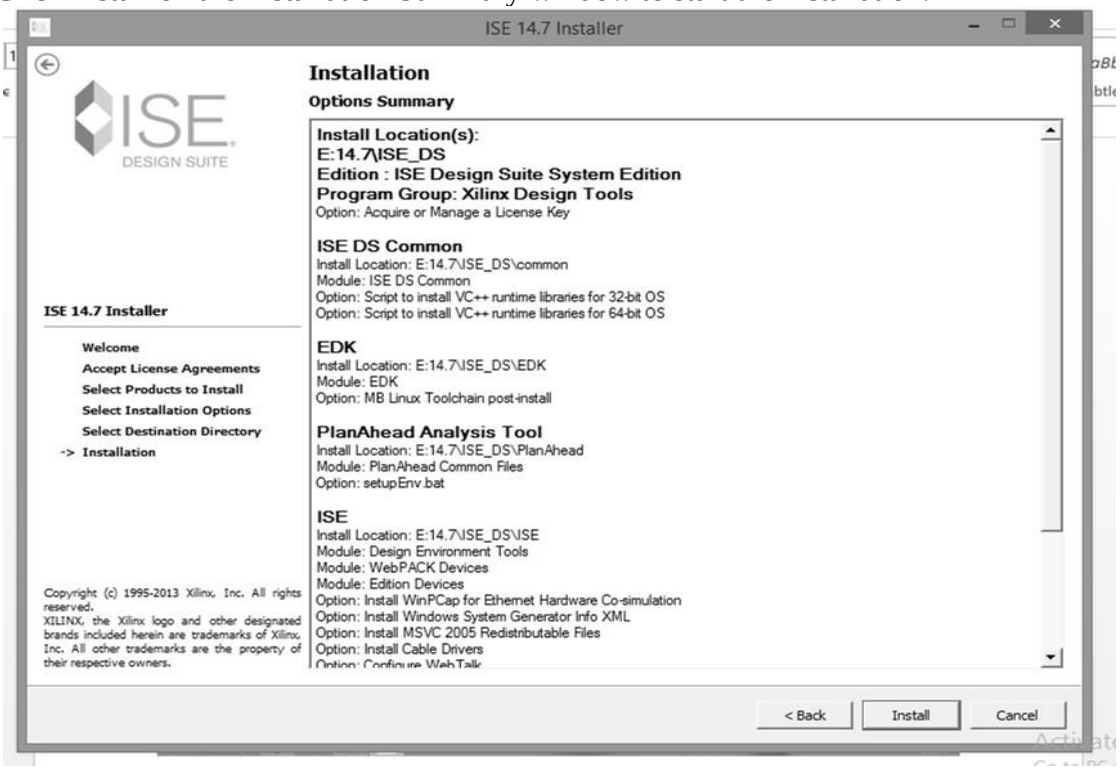


Fig. no. 2.6: Installation Summary options.

7. Installation of the software continues.

Fig. no. 2.7: Installation Window

8. Click Finish on the Install Complete window.

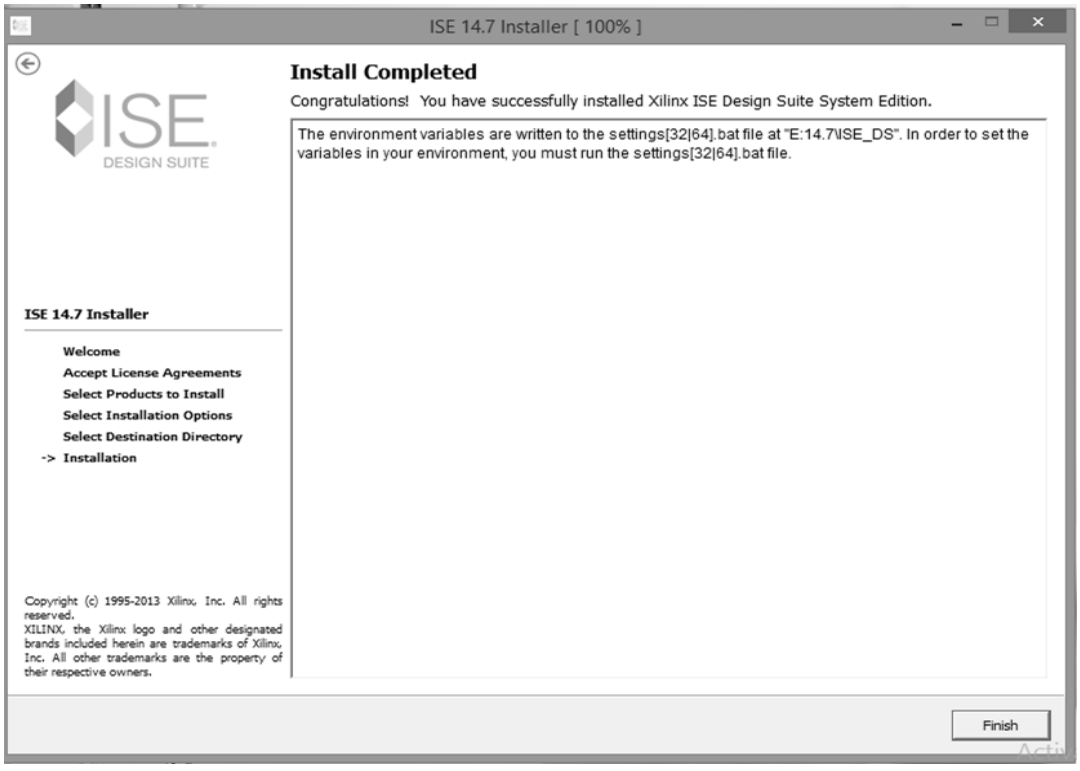


Fig. no. 2.8: Install complete window

9. Click Next on the License Configuration window.

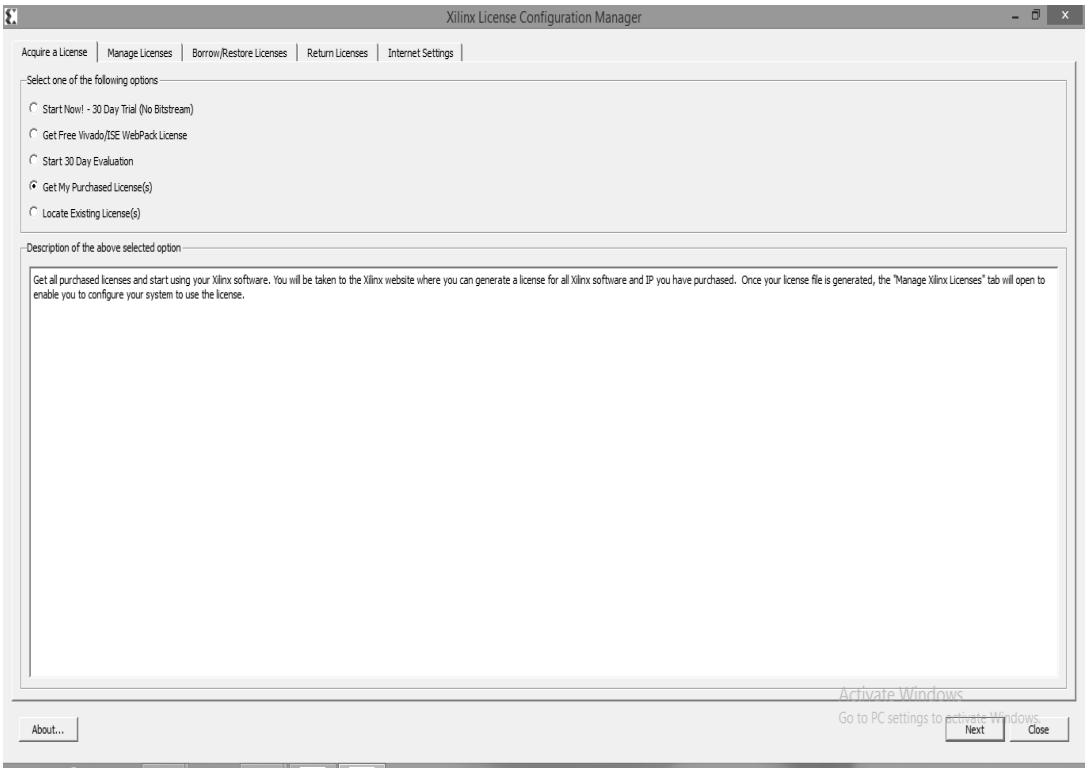


Fig. no. 2.9: License Configuration Window

10. After the license is accepted the Xilinx Project navigator icon is created on Desktop. Click on the icon to run the Xilinx EDA Tool.

Fig. no. 2.10: Xilinx Initialize Window

XI Resources

Table 2.2 Resources

Sr. No.	Name of Resource	Specifications	Quantity
1			

XII Actual Procedure

……

……

……

……

……

……

XIII Observation Table : -------NA-------

XIV Result: -------NA-------

XV Interpretation of results: -------NA-------

XVI Conclusion and recommendation

……

……

XVII Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Is it important to install compatible libraries with EDA tools Explain shortly.
2. List the main stages of VLSI design supported by EDA tools.
3. List common installation errors occur, with its solution.

[Space for Answers] (If required attach separate page)

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XVIII References/Suggestions for further reading

Table 2.3 References

Sr . No	Link / Portal	Description
1	https://en.wikipedia.org/wiki/Xilinx_ISE	Xilinx ISE information
2	http://allpcworld.com/xilinx-ise-design-suite-14-7-free-download/	Xilinx ISE Design Suite 14.7 Free Download
3	https://docs.amd.com/r/en-US/ug908-vivado-programming-debugging/Launching-the-Vivado-Lab-Edition-from-the-Command-Line-on-Windows-or-Linux	Launching the Vivado Lab Edition from the Command Line on Windows or Linux

XIX Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		**60 %**
1	Handling of the components	10%
2	Identifying the features of IDE	20%
3	Follow ethical practices.	20%
4	working in teams	10%
Product Related: 10 Marks		**40%**
5	Correct procedure followed	10%
6	Conclusion	10%
7	Installation related questions	15%
8	Submitting the journal in time	5%
Total (25 Marks)		**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.3: *Develop VHDL code for basic and universal gate for data flow model

I Practical Significance

Teaches the fundamentals of digital logic implementation in VHDL using **dataflow modeling** (concurrent signal assignments).Reinforces mapping between Boolean algebra and hardware (how gates become synthesized logic). Builds skills required for FPGA/CPLD design flow: write → simulate → synthesize → program → test.Forms the basis for designing more complex combinational and sequential circuits.

II Industry/Employer Expected Outcome

The aim of this course is to attend following industry/employer excepted outcome through various teaching learning experiences:
"Develop VLSI-based electronic circuit/component using VHDL."

III Course Level Learning Outcome

Use VHDL to develop and test digital circuits.

IV Laboratory Learning Outcomes

- LLO 3.1 Test the functionality of basic logic gates using VHDL Data flow model.
- LLO 3.2 Test the functionality of universal logic gates using VHDL Data flow model..

V Relevant Affective Domain related outcomes

- Follow good coding and documentation practices for reproducible lab results.
- Explore variations with different inputs and gate chaining.
- Debug simulation/synthesis issues methodically.
- Work in pairs to cross-check test vectors and board results.

VI Relevant Theoretical Background

VHDL modeling styles: structural (instantiating components), behavioral (processes / variables), and **dataflow** (concurrent signal assignments using Boolean operators).
Dataflow model: uses concurrent assignments like `Y <= A and B;` to express combinational logic directly, each assignment represents a continuous mapping from signals to outputs.

Synthesis mapping: synthesizer turns Boolean expressions into LUTs/gates on an FPGA or into logic on CPLD; NOT, AND, OR, NAND, NOR, XOR are basic primitives.
Timing and hazards: purely combinational dataflow has no clock; consider propagation delay and possible static/dynamic hazards when combining gates.

VII Circuit Diagram

a) Suggested Circuit diagram:

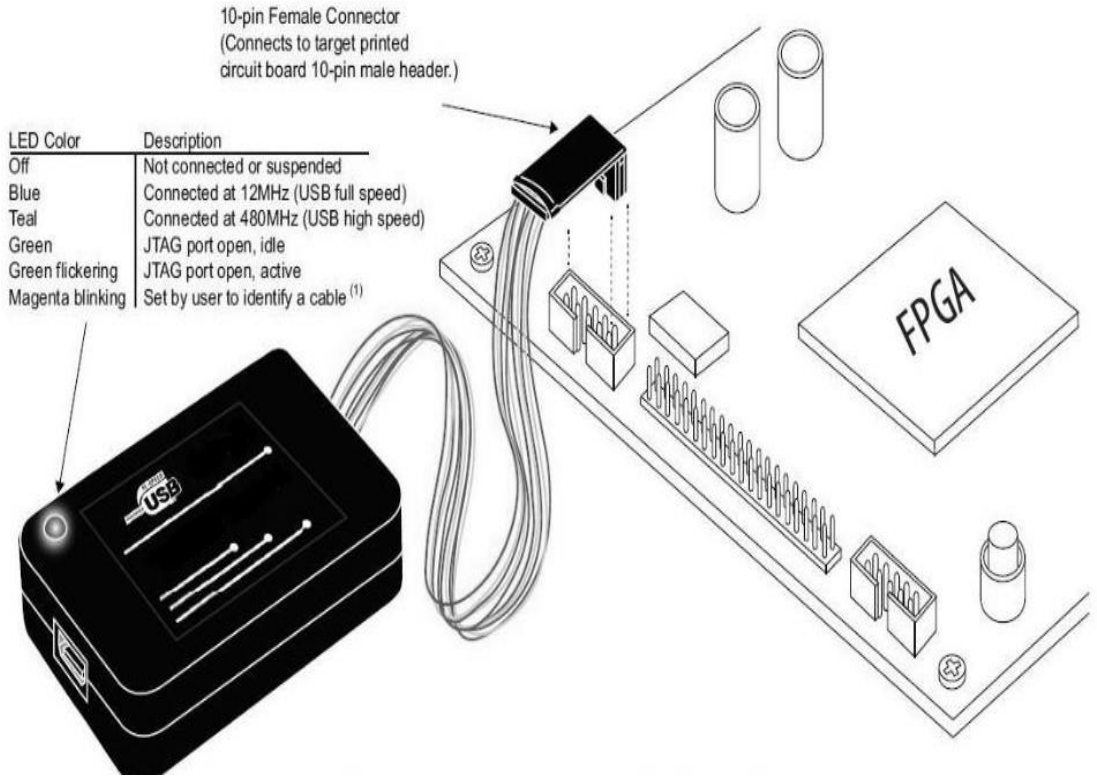


Fig. no. 3.1 Practical Setup with JTAG CABLE

b) Actual block diagram used in laboratory with related equipment rating. CPLD Development board:

VIII Required Resources/apparatus/equipment with specifications

Table 3.1 Required Resources

No	Instrument / Components	Specification	Quantity
1.	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208),On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit., On board, 2 Crystal 8MHz and 25MHz. .JTAG Interface (Boundary Scan),PROM Interface (XCF02S),40 pin, 4 header connector for external I/O's Or any other kit with equivalent specifications	1 No.
2.	Desktop PC	Loaded with open source IDE, simulation and program downloading software, UPS and Antivirus.	1 No.

IX Precautions to be followed

1. Check the syntax / rules of VHDL Programming.
2. Do not power up the board before completing connections.

X Procedure

1. Create the Xilinx ISE project for your top-level FPGA design, by doing the following in ISE:
2. In the ISE software, select File > New Project. In the Project name and Project location fields, enter the project name and location, respectively.
3. Select HDL or Schematic as the Top-level source type, and click Next

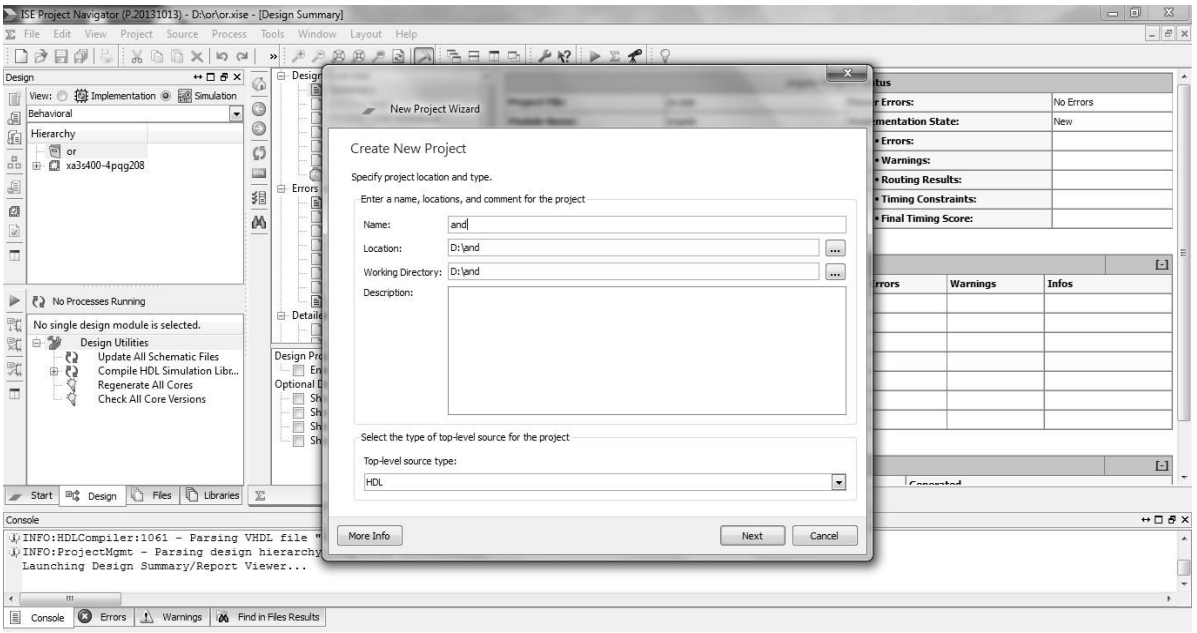


Fig. no. 3.2: Create new project

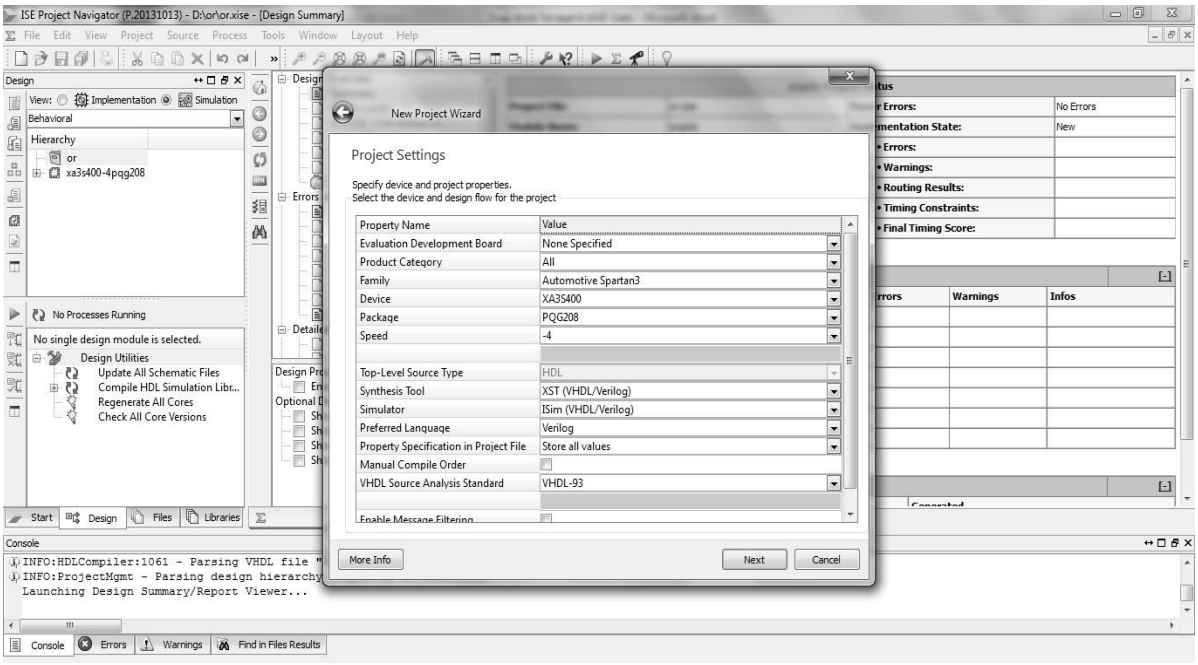


Fig. no. 3.3: Project File Settings

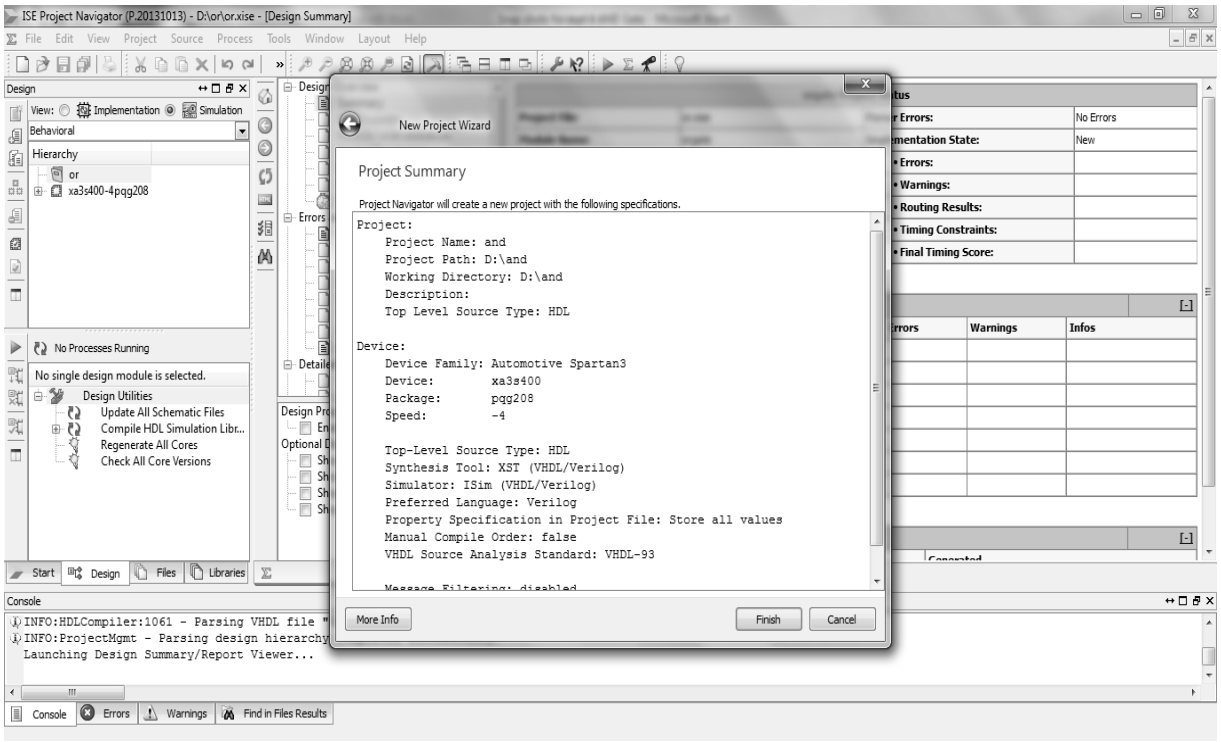


Fig. no. 3.4: Project Summary

4. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".

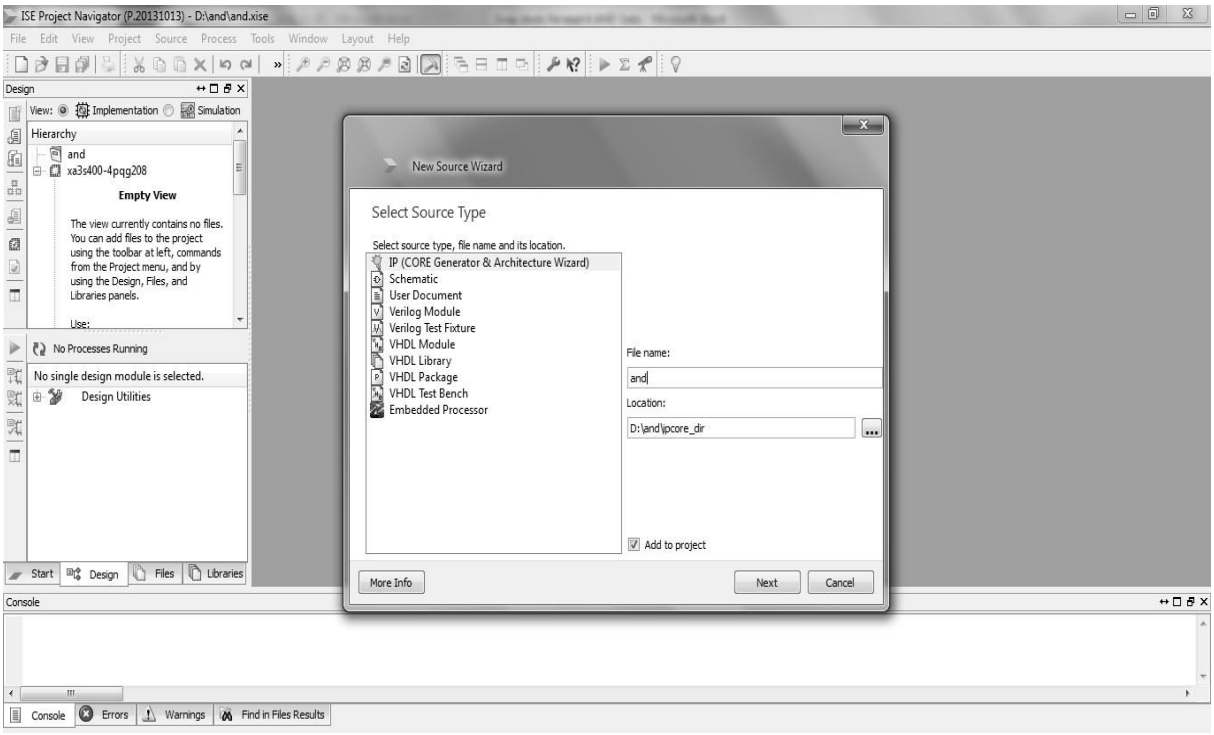


Fig. no. 3.5: Selecting Source Type

5. Define Module". Enter the entity used in design and then click "Next".

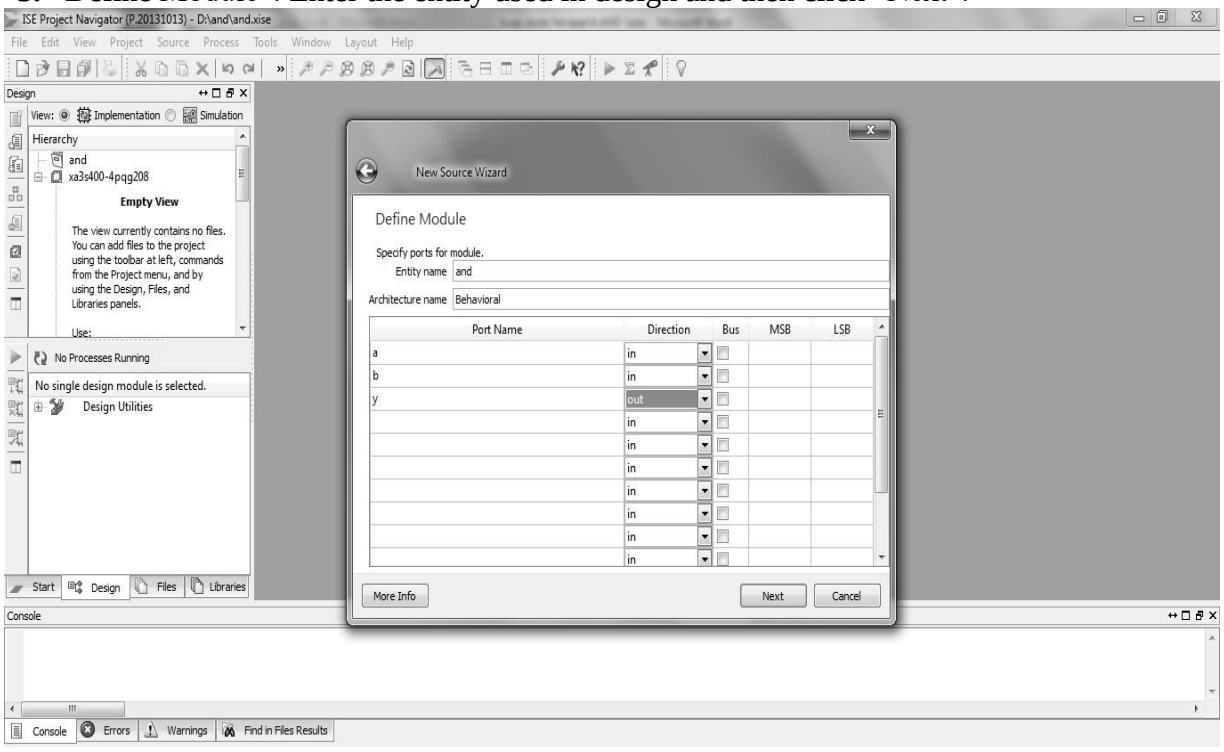


Fig. no. 3.6: Define Entity

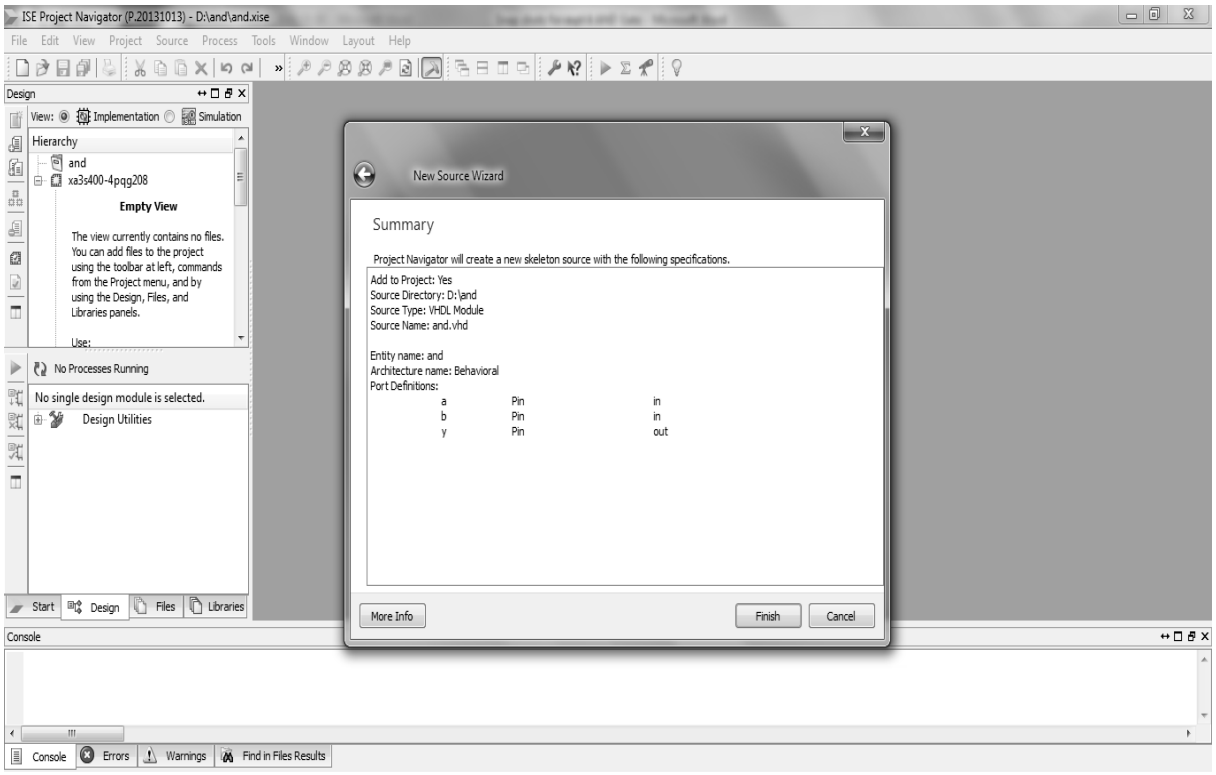


Fig. no.3.7 Entity Summary

6. Develop VHDL code for given problem and synthesize the .vhd file and view RTL schematic.

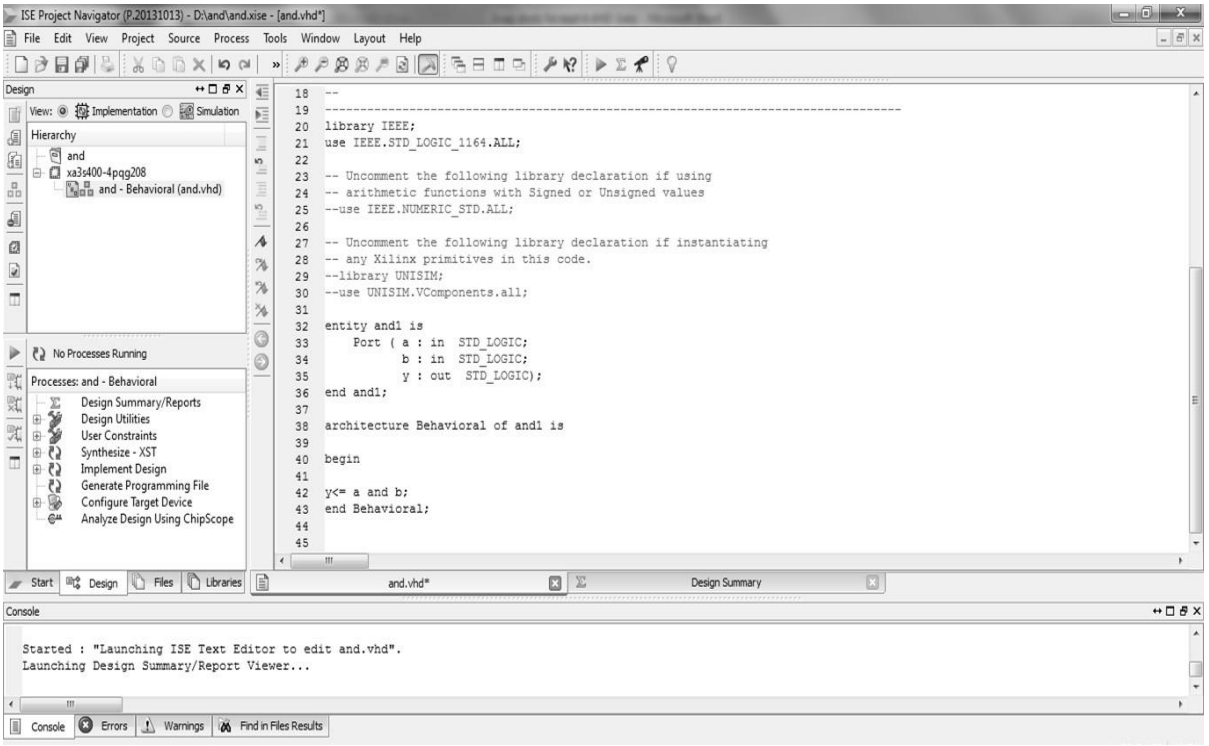


Fig. no. 3.8: VHDL Coding

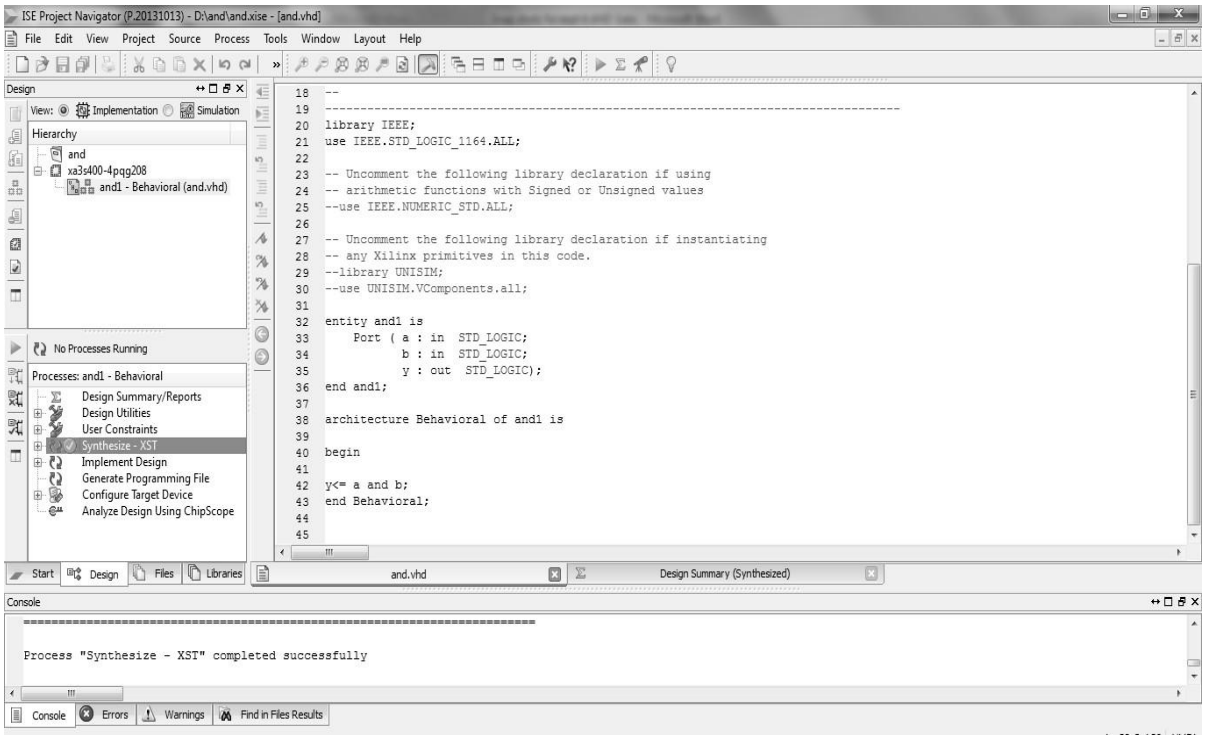


Fig. no. 3.9: Synthesize the .vhd file

7. Create Test Bench file- A **test bench** is **HDL** code that allows you to provide a documented, repeatable set of stimuli that is portable across different simulators. A **test bench** can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional **testing.**
8. Go to implementation to simulation tab, right click on main source file and create Test Bench file for simulation.

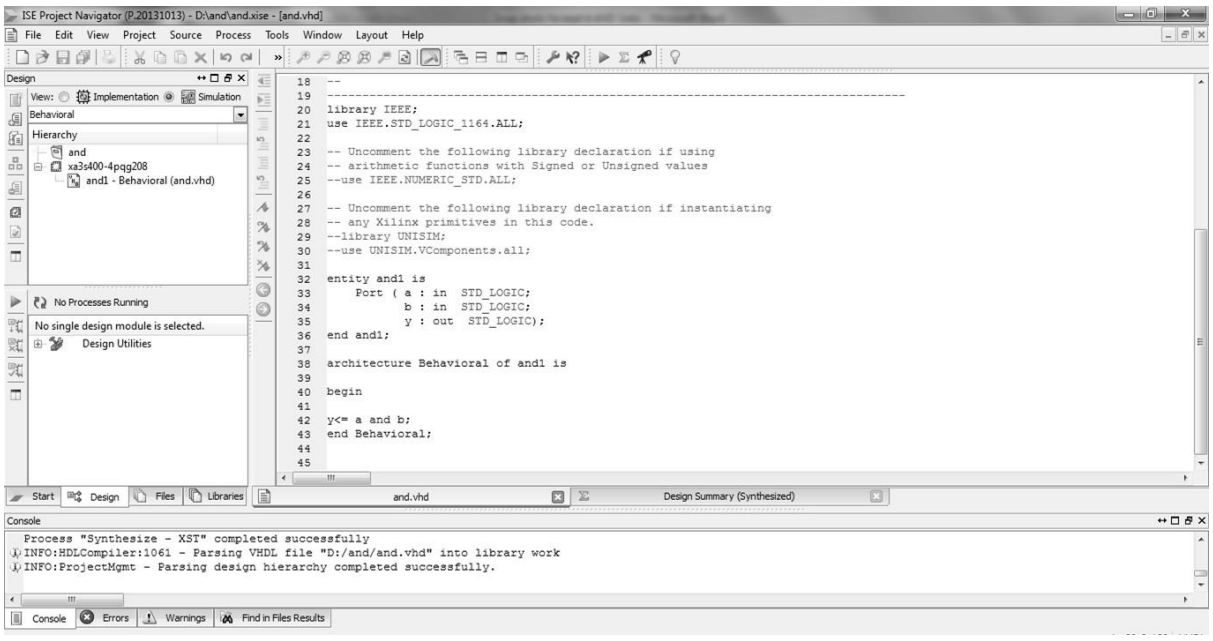
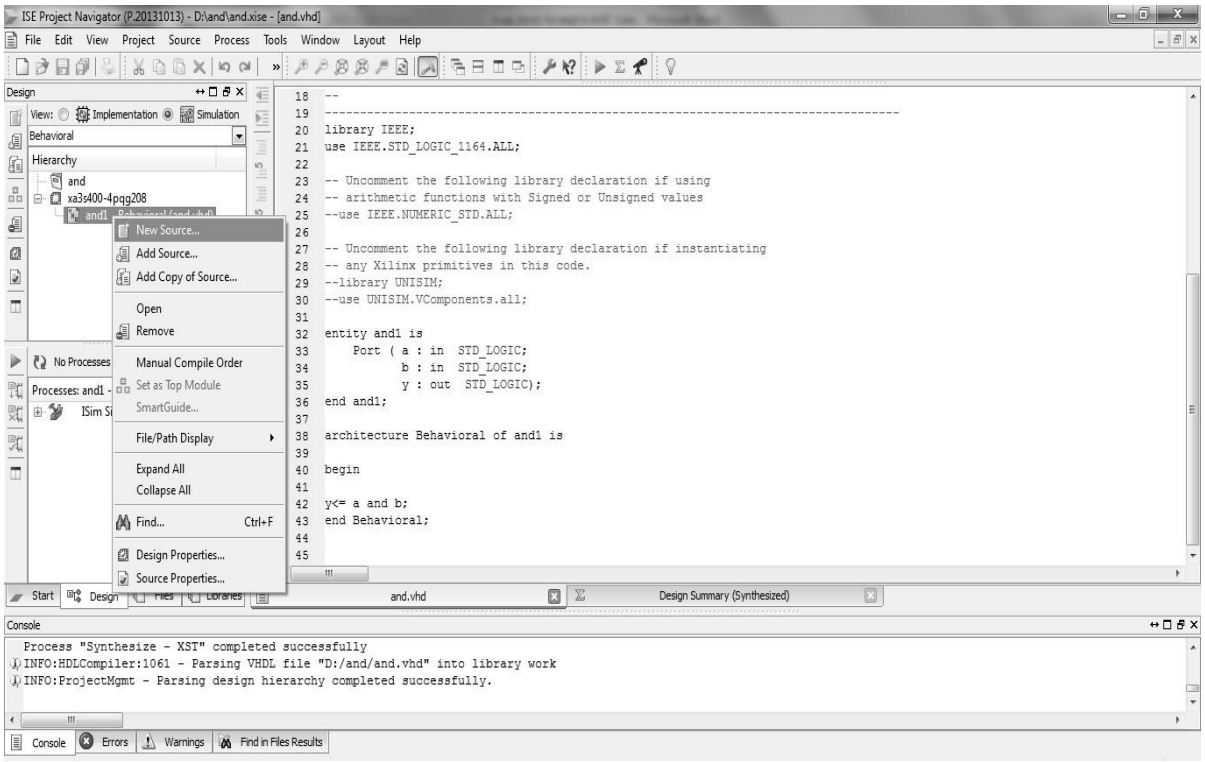


Fig. no. 3.10 Select Simulation tab to create test bench file




```
18  --
19  ----------------------------------------------------------------------------------
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 instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity and1 is
33      Port ( a : in  STD_LOGIC;
34             b : in  STD_LOGIC;
35             y : out  STD_LOGIC);
36  end and1;
37
38  architecture Behavioral of and1 is
39
40  begin
41
42  y<= a and b;
43  end Behavioral;
44
45
```

Fig. no. 3.11 (a): Creating test bench file

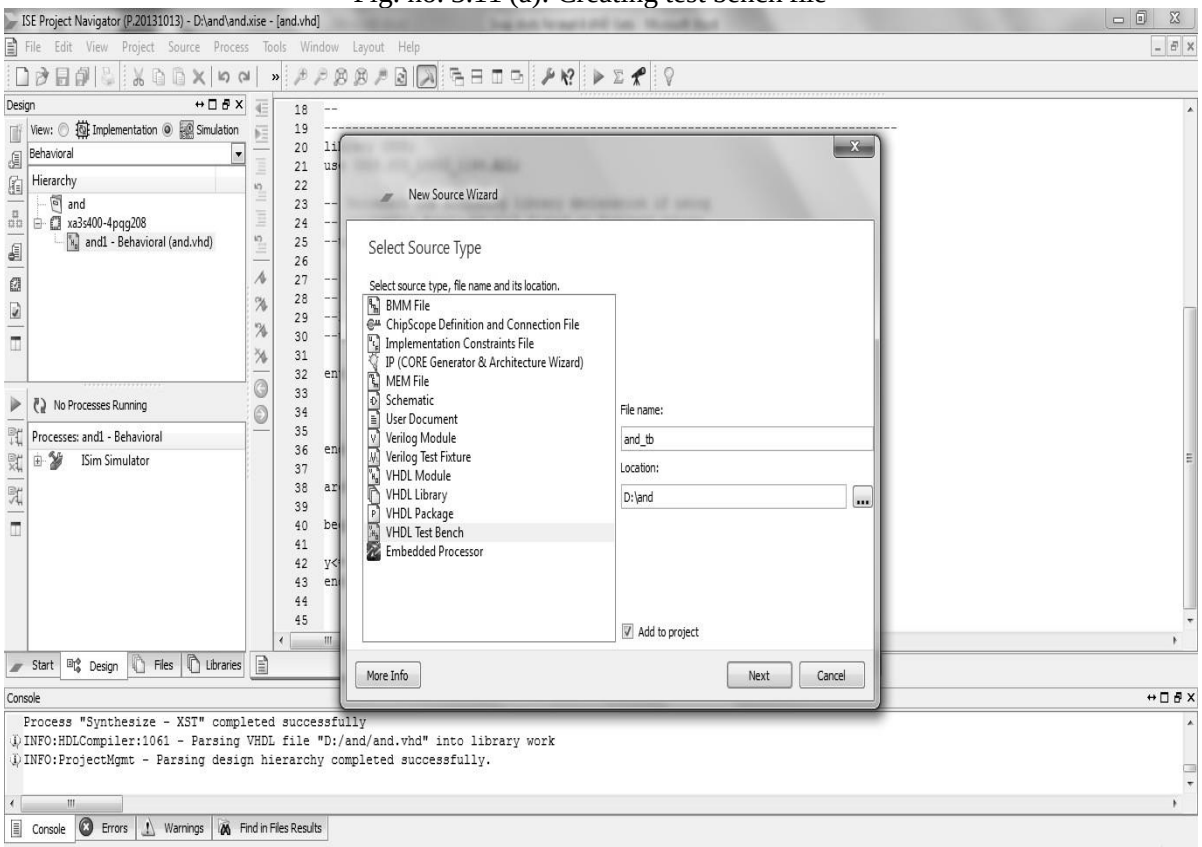


Fig. no. 3.11(b): Creating test bench contd..

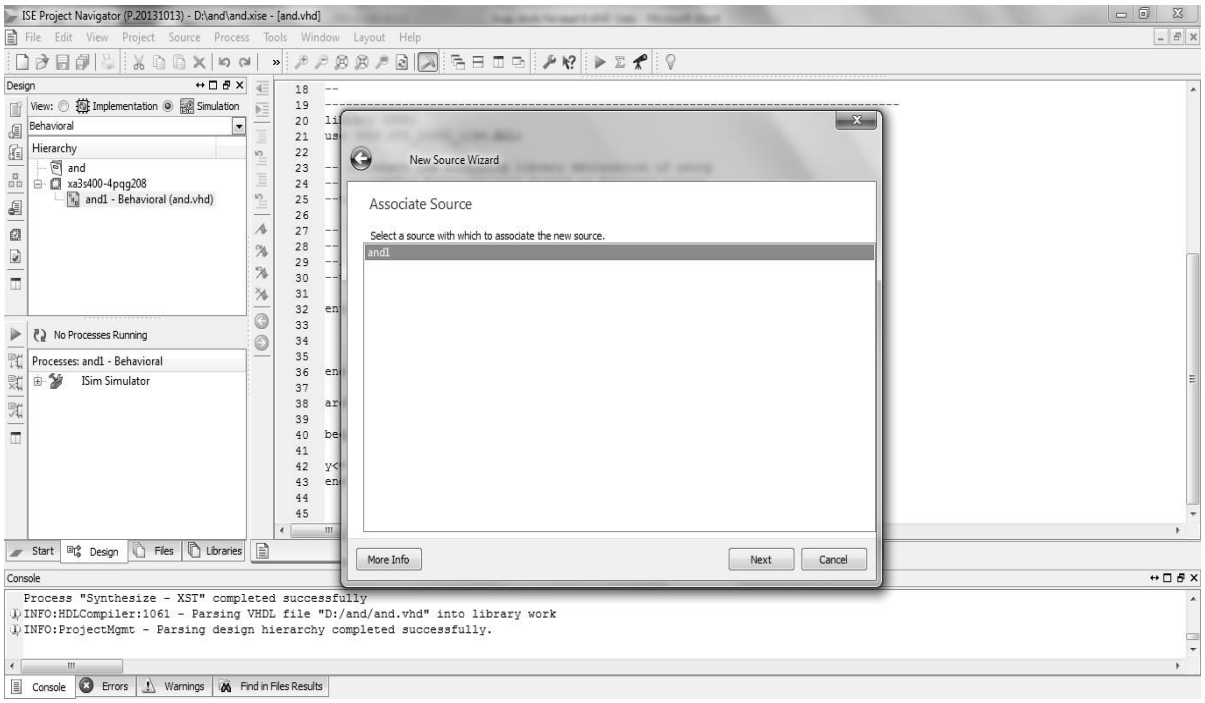


Fig. no. 3.11(c): Creating test bench contd..

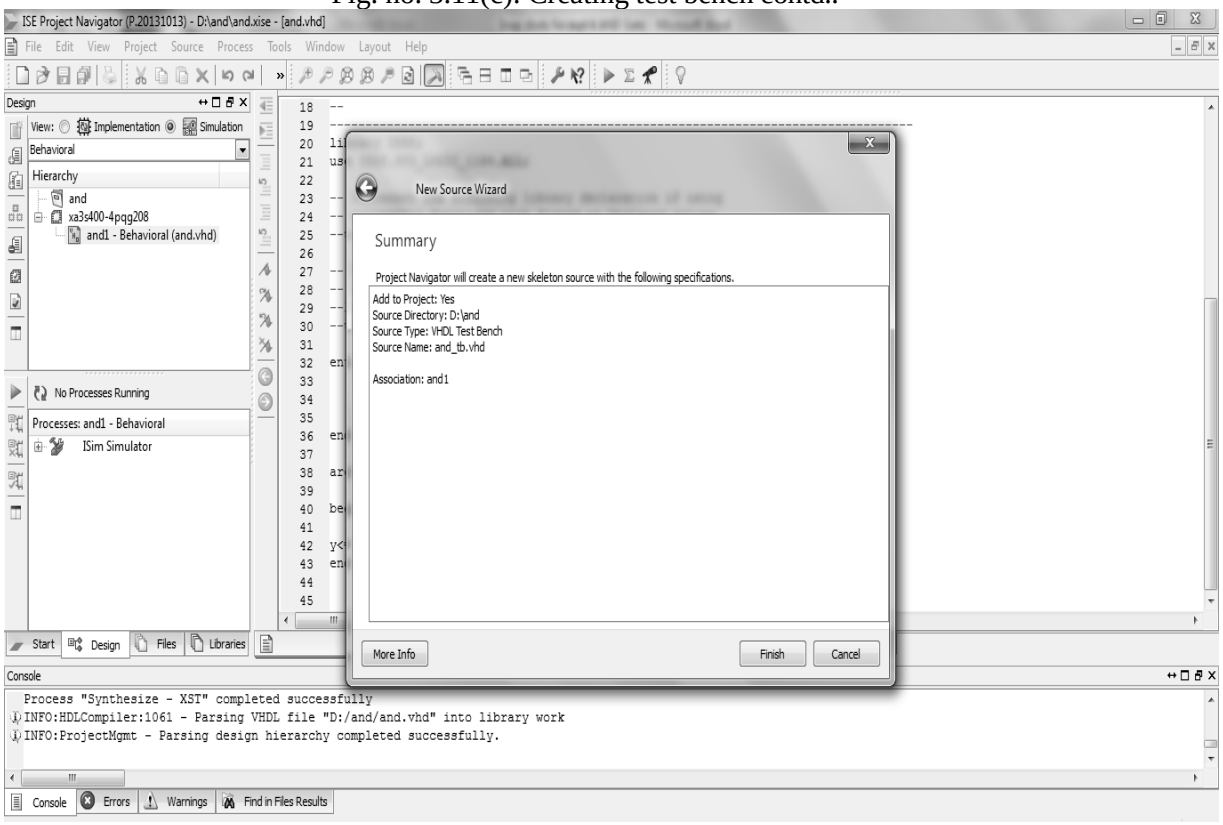


Fig. no. 3.12 Created Test Bench

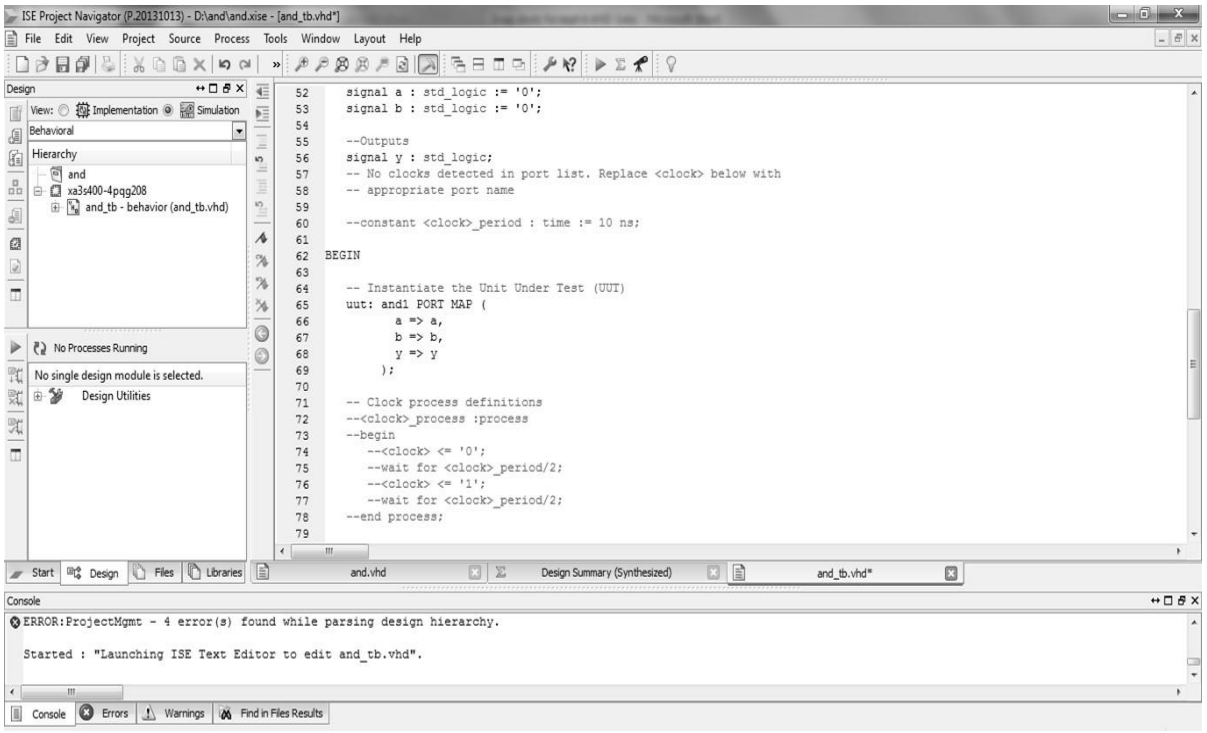


Fig. no. 3.13: Disable clock parameters [as per .vhd file]

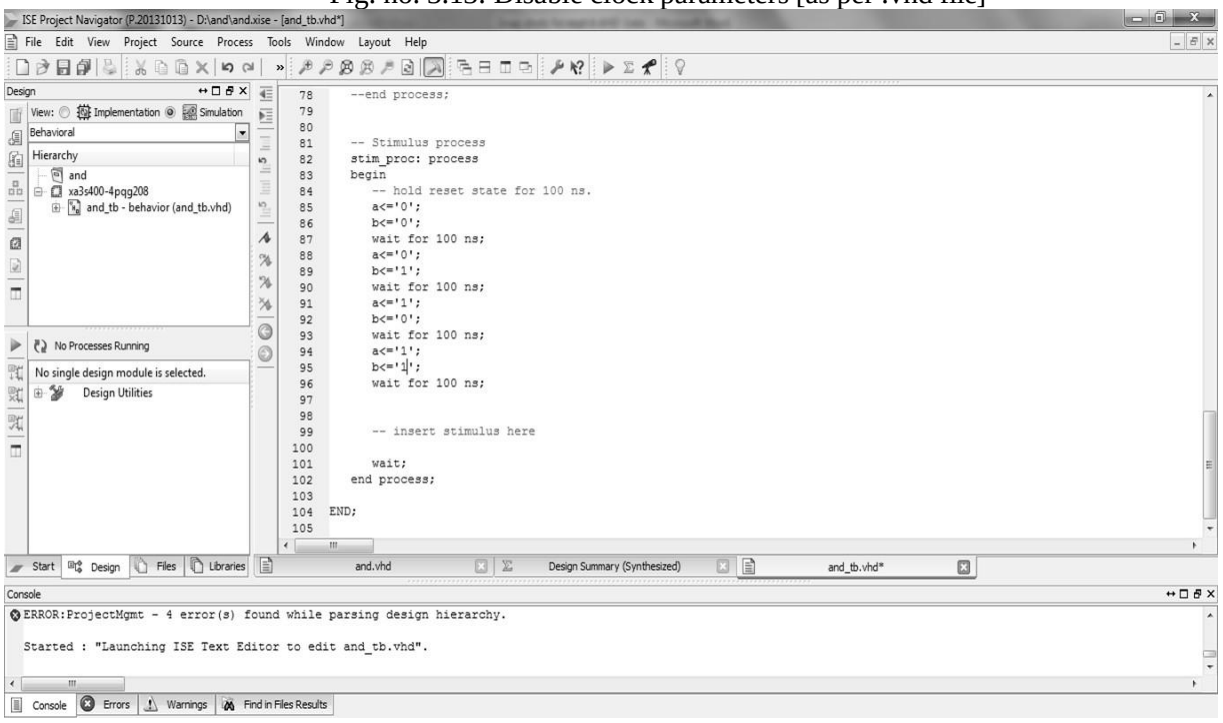


Fig. no. 3.14: Initialize Input values

9. In order to view test files, select the box of “Simulation” in the “View Panel” of the “Design” panel. In the “Process Panel,” double click on the “Behavioral Check Syntax” to make sure that you didn’t make any syntax errors while making changes.

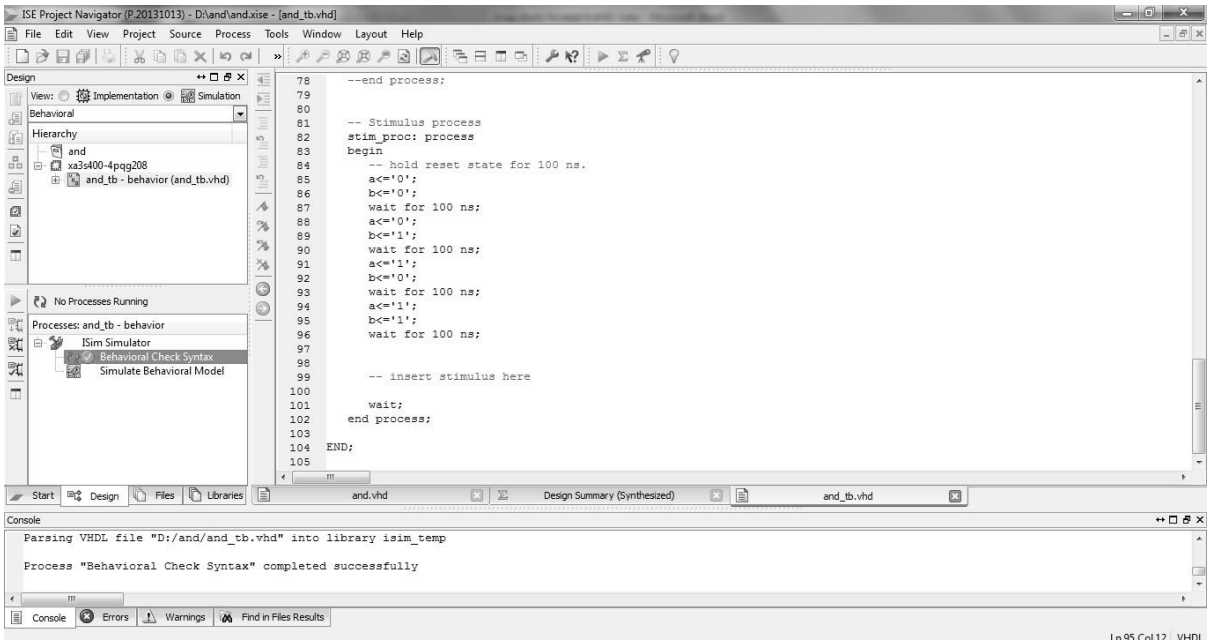


Fig. no. 3.15: Behavioural Check Syntax

10. Double click on “Simulate Behavioral Model” in the “Process Pane”, which will open the ISim software with your test bench loaded. (refer Fig. no. 4.16)

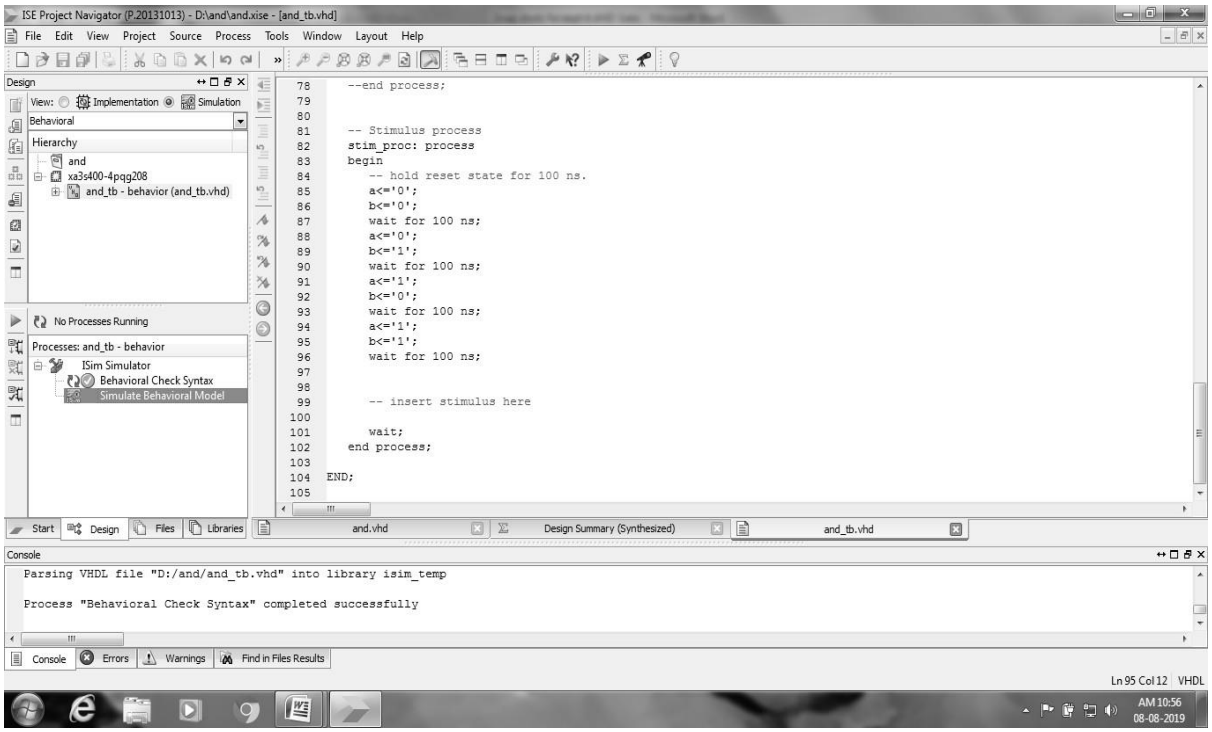


Fig. no. 3.16: Simulation Check window

11. ISim simulator window will open with your simulation executed, as shown in Fig.. where one can simulate designs and check errors if any. (Fig. no. 4.17)

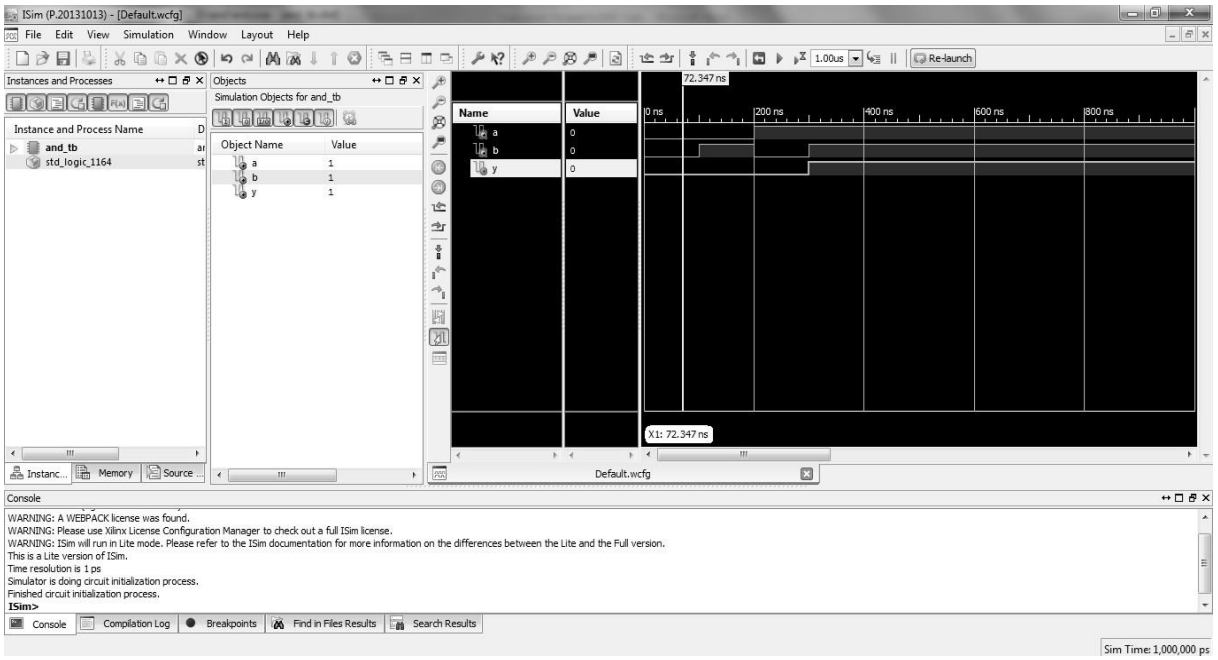


Fig. no. 3.17: Simulation Output

Sample VHDL code for basic and universal gate for data flow model:

1) AND gate

```
library ieee;
use ieee.std_logic_1164.all;

entity and_gate is
port (
   A : in  std_logic;
   B : in  std_logic;
   Y : out std_logic  );
end and_gate;

architecture dataflow of and_gate is
begin
  -- concurrent dataflow assignment
Y <= A and B;
end dataflow;
```

2) NAND gate (universal)

```
library ieee;
use ieee.std_logic_1164.all;

entity nand_gate is
port (
   A : in  std_logic;
   B : in  std_logic;
   Y : out std_logic  );
end nand_gate;
```

```
architecture dataflow of nand_gate is
begin
  Y <= not (A and B);
end dataflow;
```

3) **Program Statement for Student:** Implement NOT, OR, NOR, XOR

XI Resources

Table 3.2 Resources

Sr. No.	Name of Resource	Specifications	Quantity
1		--	
2			
3			

XII Actual Procedure

……

……

……

……

……

……

……

……

……

……

……

……

XIII Observation Table (use blank sheet provided if space not sufficient)

Observations for NOT, AND, OR, NAND, NOR, XOR or Problem Statement given to Student.

Table 3.3 Observation table

Input		Output					
A	**B**	**NOT**	**AND**	**OR**	**NAND**	**NOR**	**XOR**
0	**0**						
0	**1**						
1	**0**						
1	**1**						

XIV Result

……

……

……

XV Interpretation of result

……

.....……

XVI Conclusion and recommendation

……

……

XVII Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Give the syntax of Entity and Architecture with example?
2. Explain the Process statement with an example
3. Explain any hazard observed in gate combinations and propose mitigation (e.g., adding small inertial delays in simulation for demonstration).

[Space for Answers] (If required attach separate page)

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XVIII References/Suggestions for further reading

Table 3.4 Reference

Sr . No	Link / Portal	Description
1	https://startingelectronics.org/software/VHDL-CPLD-course/tut2-AND-and-OR-gates/	AND Gates, OR Gates and Signals in VHDL
2	https://www.tutorialspoint.com/vlsi_design/vlsi_design_vhdl_introduction.htm	VLSI Design - VHDL Introduction, VLSI Design Useful Resources
3	https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/	VHDL Modelling Styles: Behavioral, Dataflow, Structural
4	https://de-iitr.vlabs.ac.in/exp/realization-of-logic-functions/theory.html	Realization of logic functions with the help of universal gates NAND and NOR Gate

XIX Assessment Scheme

	Performance Indicators	**Weightage**
	Process Related : 15 Marks	**60 %**
1	Coding ability	30%
2	Debugging ability	10%
3	Follow ethical practices.	20%
	Product Related: 10 Marks	**40%**
5	Correct connections to FPGA Board	20%
6	Answer to sample questions.	15%
7	Timely Submission of report Answer to sample questions.	05%
	Total (25 Marks)	**100 %**

Marks Obtained			**Dated signature of Teacher**
Process Related(15)	**Product Related(10)**	**Total (25)**	

Practical No.4: Develop VHDL code for basic and universal gate for behavioral model

I Practical Significance

Behavioral modelling is used to describe functional behaviour of circuits without specifying gate-level implementation, good for early design, verification, and testbench creation. Helps to quickly describe logic functionality and test it via simulation before committing to structural/synthesizable implementations. Reinforces concepts of concurrency vs sequential processes, signal vs variable assignments, and sensitivity lists.

II Industry/Employer Expected Outcome

The aim of this course is to attend following industry/employer excepted outcome through various teaching learning experiences:
"Develop VLSI-based electronic circuit/component using VHDL."

III Course Level Learning Outcome

Use VHDL to develop and test digital circuits.

IV Laboratory Learning Outcomes

- LLO 4.1 Test the functionality of basic logic gates using VHDL behavioral model.
- LLO 4.2 Test the functionality of universal logic gate using VHDL behavioral model.

V Relevant Affective Domain related outcomes

- Follow safe practices.
- Maintain tools and equipment.
- Follow ethical practices.

VI Relevant Theoretical Background

- **VHDL models:** Dataflow (concurrent signal assignment), Behavioral (process-based descriptions, sequential statements), and Structural (component instantiation and interconnection).
- **Behavioral model:** Uses process blocks and sequential statements (if, case, variable assignments, wait) to describe the behavior. Processes may be sensitive to signals and execute when those signals change.
- **Signals vs Variables:** Signals represent wires/registers visible outside the process. Variables are local to a process and update immediately within the process (but their values only appear outside when assigned to a signal).
- **Synthesis considerations:** Not all behavioral constructs are synthesizable. Keep processes combinational (if with complete sensitivity list or process(all)) and avoid infinite wait loops or non-synthesizable file I/O for synthesizable code.

VII Circuit diagram

a) Suggested Practical Setup:

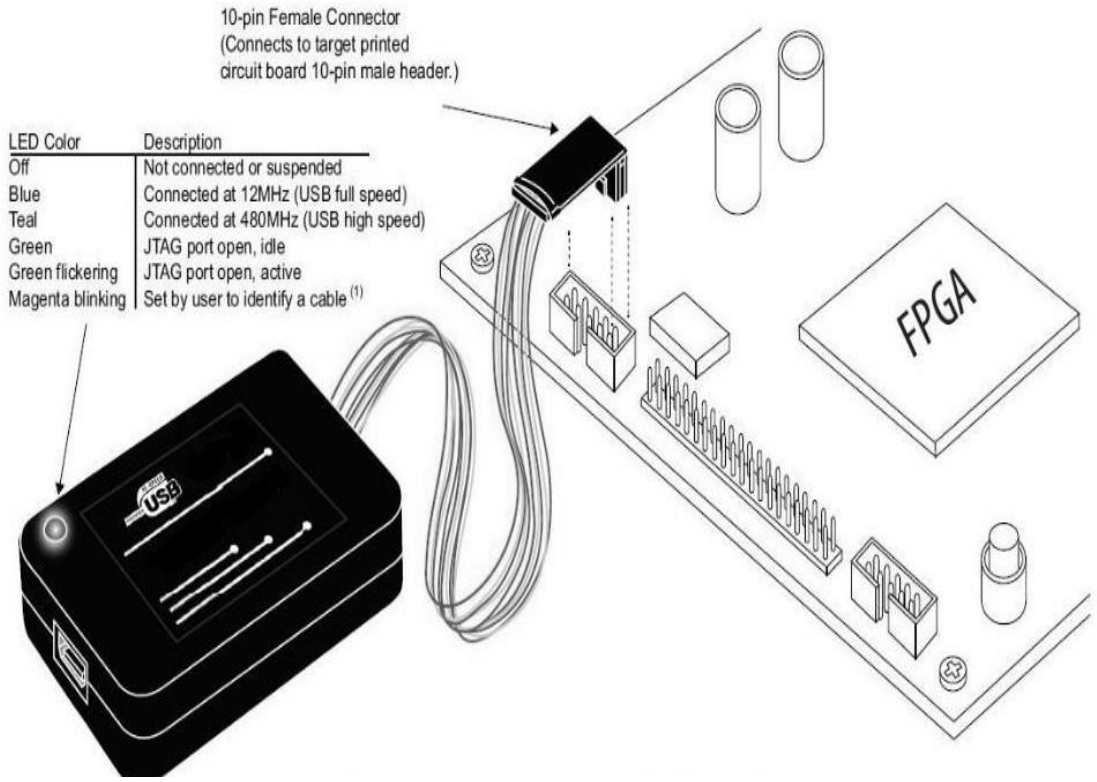


Fig. no. 4.1 Practical Setup with JTAG CABLE

b) Actual Practical Setup used in laboratory with related equipment rating:

VIII Required Resources/apparatus/equipment with specifications

Table 4.1 Required Resources

Sr. No	Instrument / Components	Specification	Quantity
1	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208),On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit., On board, 2 Crystal 8MHz and 25MHz. .JTAG Interface (Boundary Scan),PROM Interface (XCF02S),40 pin, 4 header connector for external I/O's	1 No.
2	Desktop PC	Personal computer with latest configuration. Loaded with open-source IDE, simulation and program downloading software, UPS and Antivirus.	1 No.

IX Precautions to be followed

1. Check the syntax / rules of VHDL Programming.
2. Do not power up the board before completing connections.

X Procedure

1. Create the Xilinx ISE project for your top-level FPGA design, by doing the following in ISE:
2. In the ISE software, select File > New Project. In the Project name and Project location fields, enter the project name and location, respectively.
3. Select HDL or Schematic as the Top-level source type, and click Next
4. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".
5. Define Module". Enter the entity used in design and then click "Next".
6. Develop VHDL code for given problemand Synthesize the .vhd file and view RTL schematic.
7. Create Test Bench file- A test bench is HDL code that allows you to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing.
8. Go to implementation to simulation tab, right click on main source file and create Test Bench file for simulation.
9. In order to view test files, select the box of "Simulation" in the "View Panel" of the "Design" panel. In the "Process Panel," double click on the "Behavioral Check Syntax" to make sure that you didn't make any syntax errors while making changes.Double click on "Simulate Behavioral Model" in the "Process Pane", which will open the ISim software with your test bench loaded.
10. ISim simulator window will open with your simulation executed, where one can simulate designs and check errors if any.

- **Sample VHDL code for basic and universal gate for data flow model:**

1) AND gate

```
library ieee;
use ieee.std_logic_1164.all;

entity and2_beh is
port (
a : in std_logic;
b : in std_logic;
y : out std_logic );
end entity;

architecture behavioral of and2_beh is
begin

-- combinational process (behavioral)
comb_proc: process(a,b)

begin

if (a = '1' and b = '1') then
y <= '1';
else
y <= '0';

end if;
end process comb_proc;
end architecture behavioral;
```

2) NAND gate (universal gate)

```
library ieee;
use ieee.std_logic_1164.all;

entity nand_beh is
port (a,b: in std_logic; y: out std_logic);
end nand_beh;

architecture behavioral of nand_beh is
begin
proc: process(a,b)
begin
if (a = '1' and b = '1') then
y <= '0';
else
y <= '1';
end if;
end process;
end behavioral;
```

3) NOR gate (universal)

```
library ieee;
use ieee.std_logic_1164.all;

entity nor_beh is
port (a,b: in std_logic; y: out std_logic);
end nor_beh;

architecture behavioral of nor_beh is
begin
proc: process(a,b)
begin
if (a = '0' and b = '0') then
y <= '1';
else
y <= '0';
end if;
end process;
end behavioral;
```

4) **Program Statement for Student:** Implement NOT, OR, XOR using dataflow model

XI Resources

Table 4.2 Resources

Sr. No.	Name of Resource	Specifications	Quantity
1			
2			
3			

XII Actual Procedure

……
……
……
……
……
……
……
……
……
……
……
……
……

XIII Observation Table (use blank sheet provided if space not sufficient)

Observations for NOT, AND, OR, NAND, NOR, XOR gates

Table 4.3 References

Input		Output					
A	B	NOT	AND	OR	NAND	NOR	XOR
0	0						
0	1						
1	0						
1	1						

XIV Result

……

……

..……

XV Interpretation of result

……

.....……

XVI Conclusion and recommendation

……

……

XVII Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. State the purpose of a sensitivity list in a process block
2. Modify the AND gate to create a 3-input AND using behavioral modeling.
3. List errors might occur during compilation of VHDL code. Give examples.

[Space for Answers] (If required attach separate page)

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XVIII References/Suggestions for further reading

Table 4.4 References

Sr . No	Link / Portal	Description
1	https://startingelectronics.org/software/VHDL-CPLD-course/tut2-AND-and-OR-gates/	AND Gates, OR Gates and Signals in VHDL
2	https://www.tutorialspoint.com/vlsi_design/vlsi_design_vhdl_introduction.htm	VLSI Design - VHDL Introduction, VLSI Design Useful Resources
3	https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/	VHDL Modelling Styles: Behavioral, Dataflow, Structural
4	https://de-iitr.vlabs.ac.in/exp/realization-of-logic-functions/theory.html	Realization of logic functions with the help of universal gates NAND and NOR Gate

XIX Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		**60 %**
1	Coding ability	30%
2	Debugging ability	10%
3	Follow ethical practices.	20%
Product Related: 10 Marks		**40%**
4	Correct connections to FPGA Board	20%
5	Answer to sample questions.	15%
6	Timely Submission , Answer to sample questions.	05%
Total (25 Marks)		**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.5: *Realize the half and full Adder on FPGA board

I Practical Significance

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This practical will help the students to implement the various combinational circuits like Half or Full adder/ subs tractor using FPGA.

II Industry/Employer Expected Outcome

The aim of this course is to attend following industry/employer excepted outcome through various teaching learning experiences:
"Develop VLSI-based electronic circuit/component using VHDL."

III Course Level Learning Outcome

Use VHDL to develop and test digital circuits.

IV Laboratory Learning Outcomes

- LLO 5.1 Test the functionality of half and full adder using VHDL code.
- LLO 5.2 Test the simulated Test bench waveform.

V Relevant Affective Domain related outcomes

- Follow safe practices.
- Maintain tools and equipment.
- Follow ethical practices.

VI Relevant Theoretical Background

Half Adder: A Logic circuit used for the addition of two one bit numbers is known as Half adders. It has two inputs A and B and two outputs Sum (S) and Carry (C).

Table 5.1: Truth Table for Half adder

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

From the truth table the output equations are: $S = \overline{A}B + A\overline{B}$

Carry (C) = A.B

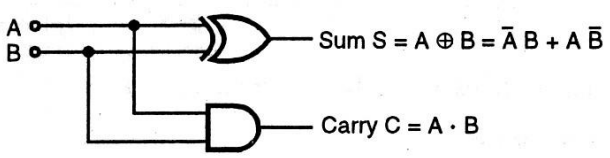


Fig. no. 5.1: Half Adder

Full Adder: In Half adder there is no provision to add the carry generated by lower bits while adding present inputs that is when multibit addition is performed. Hence a third input is added and this circuit is used to add A, Band Cin where A , B are present state inputs and

Cin is the last state output that is previous carry. This circuit is known as Full Adder.

Table No 5.2: Truth table for Full Adder

A	B	Cin	Sum (S)	Carry (C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

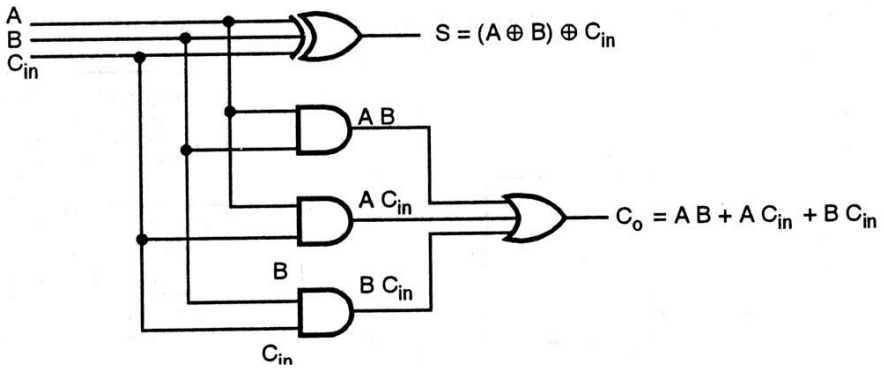


Fig. no. 5.2: Full Adder

VII a) Suggested Practical Sample:

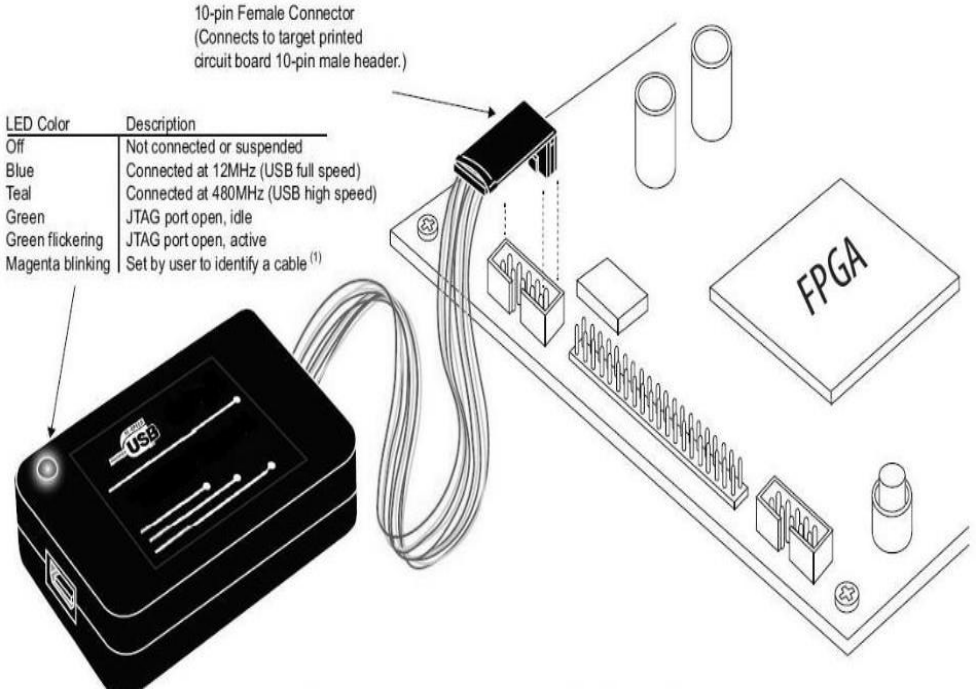


Fig. no. 5.3 Practical Setup with JTAG CABLE

b) Actual Practical Setup used in laboratory with related equipment rating:

VIII Required Resources/apparatus/equipment with specifications

Table no.5.3 Required Resources

No	Instrument / Components	Specification	Quantity
1.	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208),On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit., On board, 2 Crystal 8MHz and 25MHz. .JTAG Interface (Boundary Scan),PROM Interface (XCF02S),40 pin, 4 header connector for external I/O's	1 No.
2.	Desktop PC	Loaded with open source IDE, simulation and program downloading software, UPS and Antivirus.	1 No.

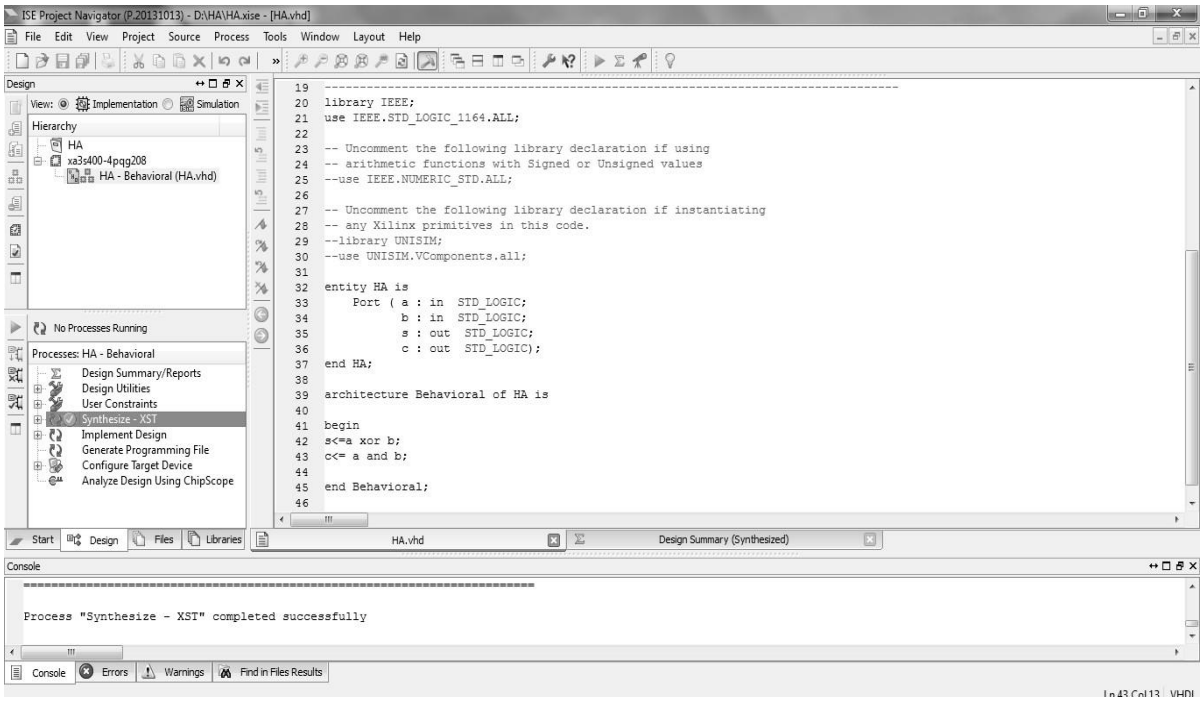
IX Precautions to be followed

1. Check the syntax / rules of VHDL Programming.
2. Do not power up the board before completing connections.

X Procedure

1. Create the Xilinx ISE project for your top-level FPGA design, by doing the following in ISE:
2. In the ISE software, select File > New Project. In the Project name and Project location fields, enter the project name and location, respectively.
3. Select HDL or Schematic as the Top-level source type, and click Next
4. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".
5. Define Module". Enter the entity used in design and then click "Next".
6. Develop VHDL code for given problem and synthesize the .vhd file and view RTL schematic.
7. Create Test Bench file- A test bench is HDL code that allows you to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing.
8. Go to implementation to simulation tab, right click on main source file and create Test Bench file for simulation.
9. In order to view test files, select the box of "Simulation" in the "View Panel" of the "Design" panel. In the "Process Panel," double click on the "Behavioral Check Syntax" to make sure that you didn't make any syntax errors while making changes.Double click on "Simulate Behavioral Model" in the "Process Pane", which will open the ISim software with your test bench loaded.
10. ISim simulator window will open with your simulation executed, where one can simulate designs and check errors if any.

a) Create HA .vhd file:




```
19  ----------------------------------------------------------------------------------
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 instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity HA is
33      Port ( a : in  STD_LOGIC;
34             b : in  STD_LOGIC;
35             s : out  STD_LOGIC;
36             c : out  STD_LOGIC);
37  end HA;
38
39  architecture Behavioral of HA is
40
41  begin
42  s<=a xor b;
43  c<= a and b;
44
45  end Behavioral;
46
```

Fig. no. 5.4: HA .vhd file

b) Create HA test bench file:

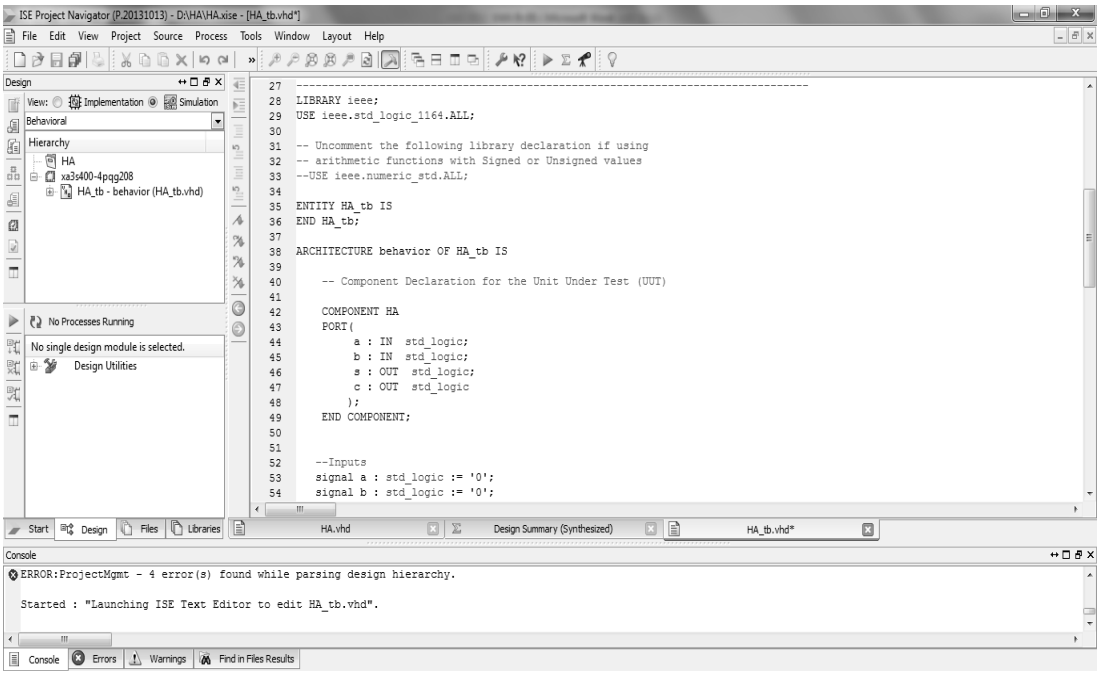


Fig. no. 5.5: HA Test Bench file

c) HA test bench Simulation waveforms:

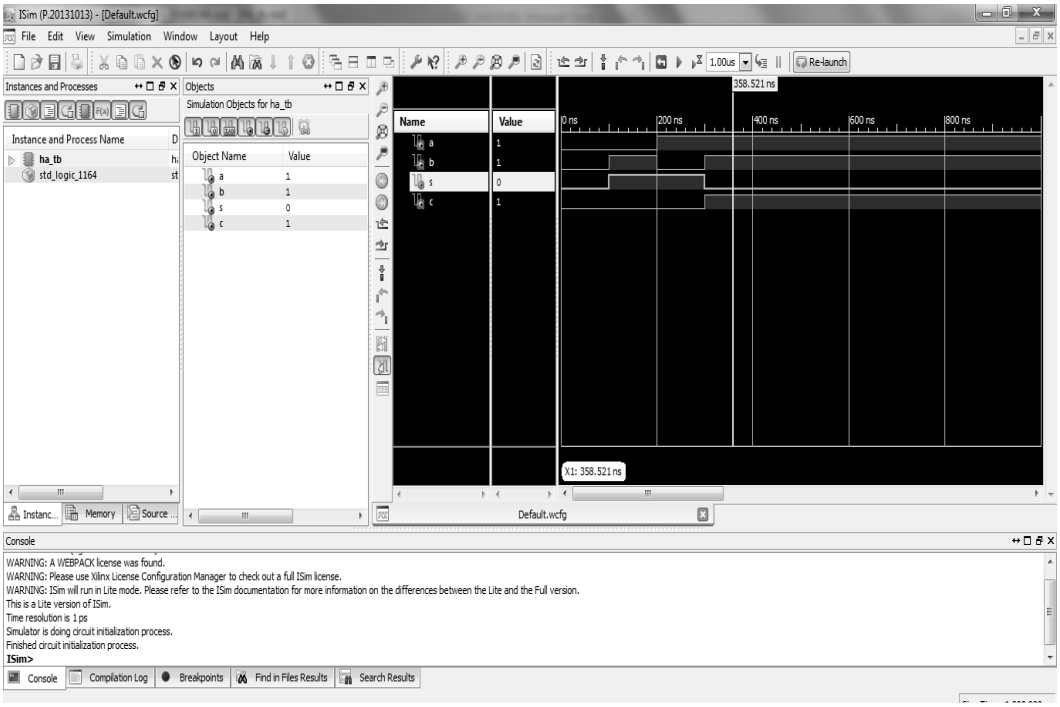


Fig. no. 5.6 HA Simulation output

- **Sample VHDL code to implement Half Adder and Full Adder:**

1) Half Adder:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOG IC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity HA is
Port ( A, B : in std_logic;
```

```
S, C : out std_logic);
end HA;
architecture dataflow of HA is
begin
S<= A xor B;
C<= A and B;
end dataflow;
```

2) Full Adder:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOG IC_ARITH.ALL;
useIEEE.STD_LOGIC_UNSIGNED.ALL;
entity full_adder is
port(A, B, Cin :in bit;
S, C:out bit);
end full_adder;
architecture data of full_adder is
begin
S<= A xor B xor Cin;
C <= ((A and B) or (B and Cin) or (A and Cin));
end data;
```

3) **Program Statement for Student:** Implement Half subtractor and Full subctractor using behavioral model

XI Resources

Table 5.4 Required Resources

Sr. No.	Name of Resource	Specifications	Quantity
1		--	
2			
3			

XII Actual Procedure

……
……
……
……
……
……
……
……
……
……
……
……
……
……

XIII Observation Table (use blank sheet provided if space not sufficient)

a) Observations for Half Subtractor for Problem Statement given to Student.

Table 5.5 Observation table

Input		**Output**	
A	**B**	**Difference**	**borrow**
0	**0**		
0	**1**		
1	**0**		
1	**1**		

b) Observations for Full Subtractor for Problem Statement given to Student.

Table 5.6 Observation table

Input			**Output**	
A	**B**	**Bin**	**Difference**	**borrow**
0	**0**	**0**		
0	**0**	**1**		
0	**1**	**0**		
0	**1**	**1**		
1	**0**	**0**		
1	**0**	**1**		
1	**1**	**0**		
1	**1**	**1**		

XIV Result

……

……

..……

XV Interpretation of result

……

.....……

XVI Conclusion and recommendation

……

……

XVII Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Write the VHDL code for half adder using Behavior Modelling.
2. Write the VHDL code for full subtractor using if-then-else statement**.**
3. Write VHDL code for full adder using mixed style of modelling.

[Space for Answers] (If required attach separate page)

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XVIII References/Suggestions for further reading

Table 5.7 References

Sr . No	Link / Portal	Description
1	https://de-iitr.vlabs.ac.in/exp/half-full-subtractor/	To Study and Verify Half and Full Subtractor through Virtual lab
2	https://startingelectronics.org/software/VHDL-CPLD-course/tut2-AND-and-OR-gates/	AND Gates, OR Gates and Signals in VHDL
3	https://www.tutorialspoint.com/vlsi_design/vlsi_design_vhdl_introduction.htm	VLSI Design - VHDL Introduction, VLSI Design Useful Resources

XIX Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		**60 %**
1	Coding ability	30%
2	Debugging ability	20%
3	Follow ethical practices.	10%
Product Related: 10 Marks		**40%**
4	Correct connections to FPGA Board	20%
5	Answer to sample questions.	15%
6	Timely Submission , Answer to sample questions.	05%
Total (25 Marks)		**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Process Related (10)**	**Total (25)**	

Practical No.6: *Realize the Multiplexer on FPGA board

I Practical Significance

Multiplexers are fundamental combinational building blocks used in communication, processor data paths, ALUs, memory selection, and digital control circuits. Implementing MUX on FPGA shown mapping behavioral logic to physical hardware. Enhances understanding of FPGA pin mapping, constraints file usage, HDL coding styles, and verification methods. Helps students correlate simulation results with actual hardware behaviour.

II Industry/Employer Expected Outcome

The aim of this course is to attend following industry/employer excepted outcome through various teaching learning experiences:
"Develop VLSI-based electronic circuit/component using VHDL."

III Course Level Learning Outcome

Use VHDL to develop and test digital circuits.

IV Laboratory Learning Outcome

LLO 6.1 Test the functionality of 4:1 multiplexer using VHDL code

V Relevant Affective Domain related outcomes

- Follow safe practices.
- Maintain tools and equipment.
- Follow ethical practices.

VI Relevant Theoretical Background

A multiplexer (MUX) is a combinational circuit that selects one input out of multiple data inputs and forwards it to the output based on select-line values.
4:1 Multiplexer Structure:

- 4 data inputs: D0, D1, D2, D3
- 2 select inputs: S1, S0
- 1 output: Y

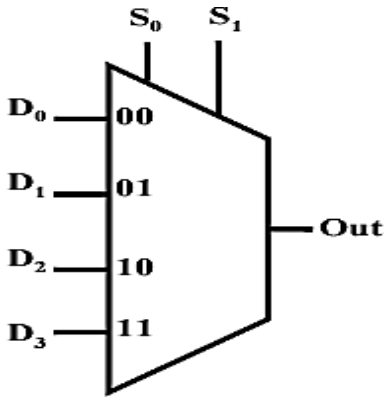


Fig. no. 6.1 Block Diagram of 4:1 Multiplexer

- **Truth Table of 4:1 MUX**

Table 6.1: Truth Table of 1:4 Multiplexer

S1	S0	Output Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

VII a) Suggested Practical Sample:

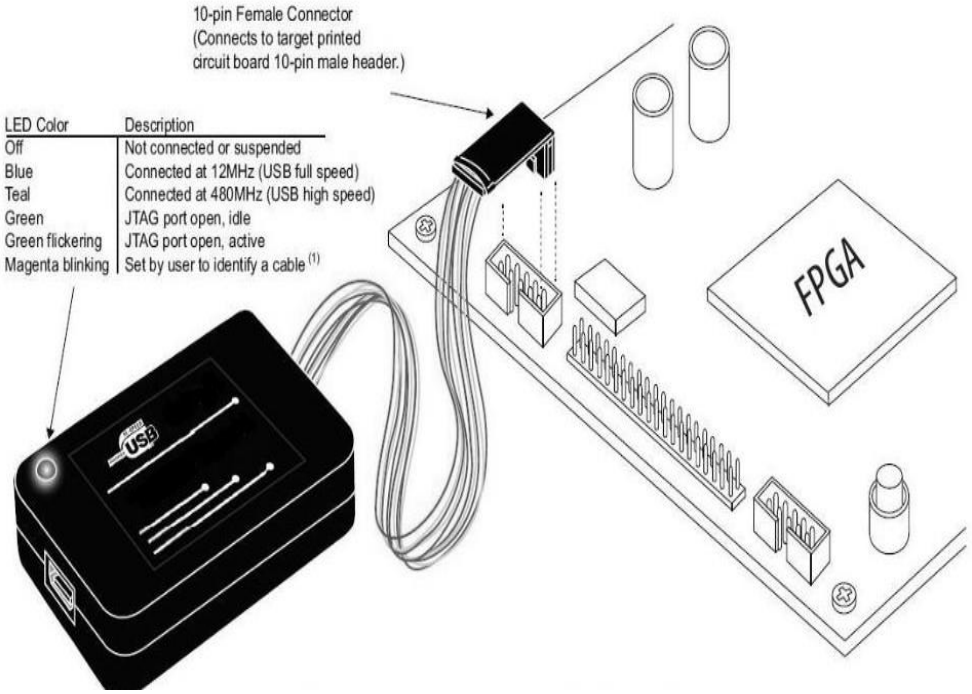


Fig. no. 6.2 Practical Setup with JTAG CABLE

b) Actual Practical Setup used in laboratory with related equipment rating:

VIII Required Resources/apparatus/equipment with specifications

Table 6.2 Required Resources

Sr.No	Instrument / Components	Specification	Quantity
1.	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208),On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit., On board, 2 Crystal 8MHz and 25MHz. .JTAG Interface (Boundary Scan),PROM Interface (XCF02S),40 pin, 4 header connector for external I/O's	1 No.
2.	Desktop PC	Loaded with open source IDE, simulation and program downloading software, UPS and Antivirus.	1 No.

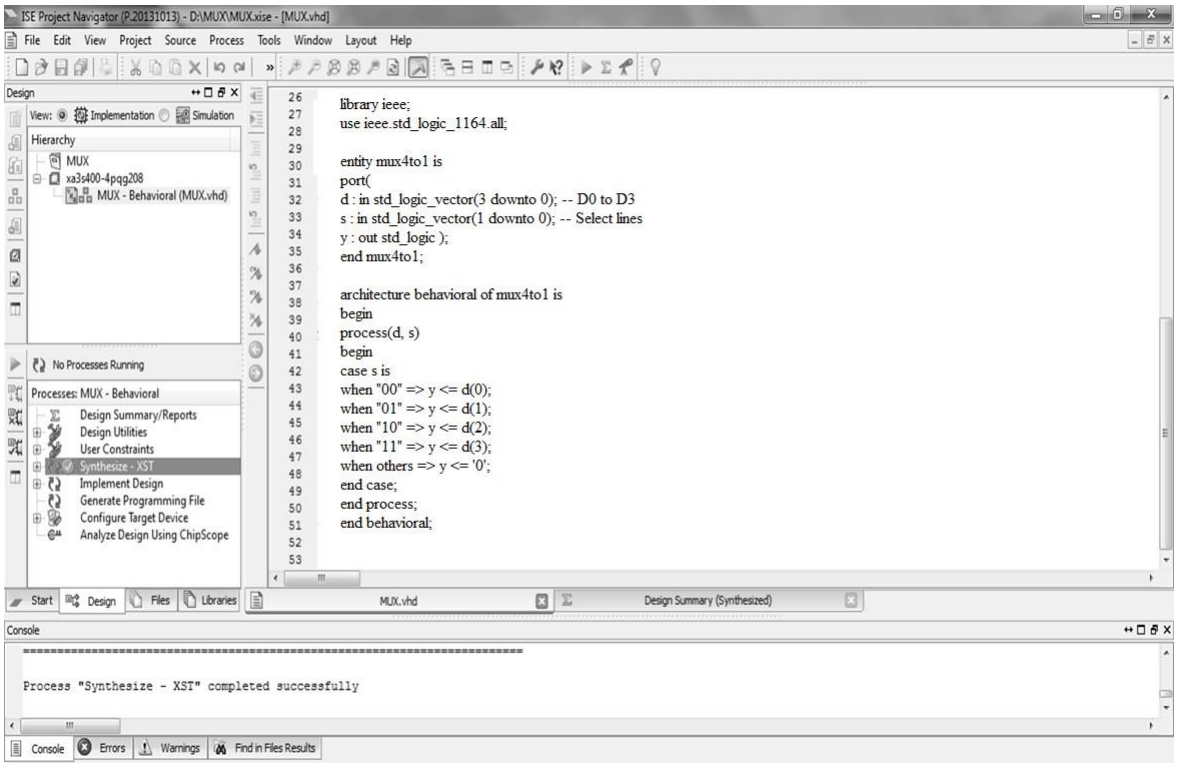
IX Precautions to be followed

1. Check the syntax / rules of VHDL Programming.
2. Do not power up the board before completing connections.

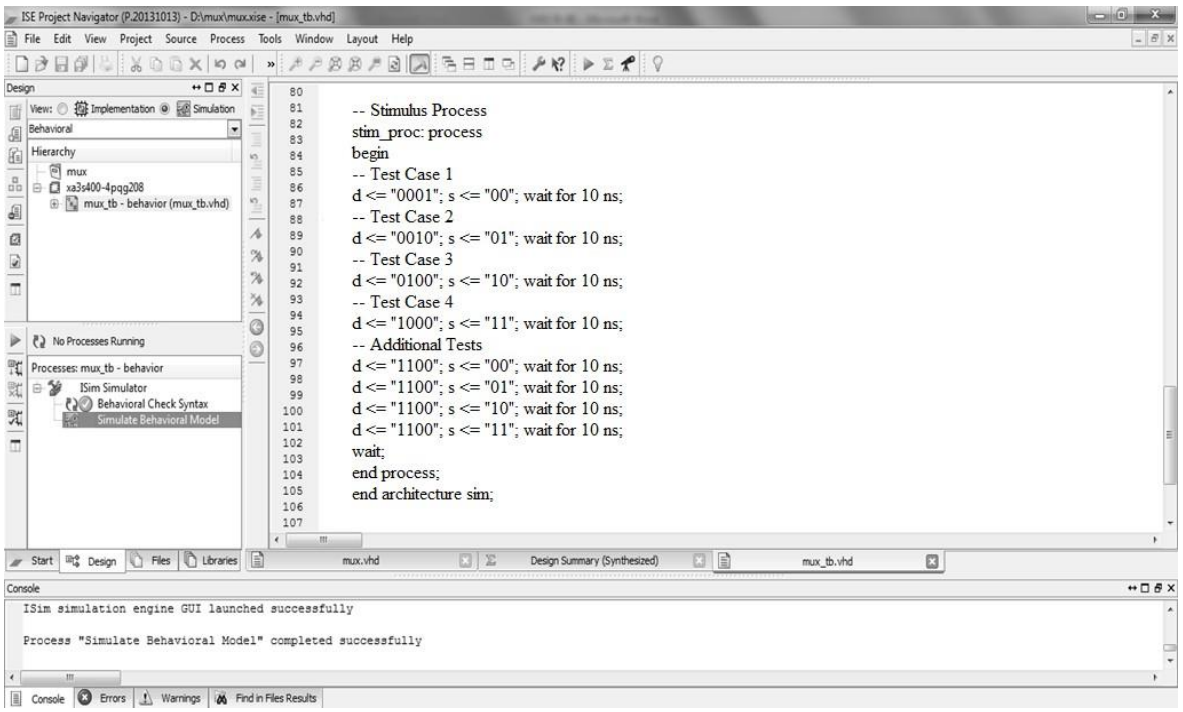
X Procedure

1. Create the Xilinx ISE project for your top-level FPGA design, by doing the following in ISE:
2. In the ISE software, select File > New Project. In the Project name and Project location fields, enter the project name and location, respectively.
3. Select HDL or Schematic as the Top-level source type, and click Next
4. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".
5. Define Module". Enter the entity used in design and then click "Next".
6. Develop VHDL code for given problem and synthesize the .vhd file and view RTL schematic.
7. Create Test Bench file- A test bench is HDL code that allows you to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing.
8. Go to implementation to simulation tab, right click on main source file and create Test Bench file for simulation.
9. In order to view test files, select the box of "Simulation" in the "View Panel" of the "Design" panel. In the "Process Panel," double click on the "Behavioral Check Syntax" to make sure that you didn't make any syntax errors while making changes.Double click on "Simulate Behavioral Model" in the "Process Pane", which will open the ISim software with your test bench loaded.
10. ISim simulator window will open with your simulation executed, where one can simulate designs and check errors if any.

a) Create 4:1 multiplexer .vhd file:



b) Create 4:1 multiplexer test bench file:



- **Sample VHDL Code to Implement 4:1 MUX**

1. Behavioral Model:

```
library ieee;
use ieee.std_logic_1164.all;

entity mux4to1 is
port(
d : in std_logic_vector(3 downto 0); -- D0 to D3
s : in std_logic_vector(1 downto 0); -- Select lines
y : out std_logic );
end mux4to1;

architecture behavioral of mux4to1 is
begin
process(d, s)
begin
case s is
when "00" => y <= d(0);
when "01" => y <= d(1);
when "10" => y <= d(2);
when "11" => y <= d(3);
when others => y <= '0';
end case;
end       process;
end behavioral;
```

2. Testbench for 4:1 Multiplexer

```
library ieee;
use ieee.std_logic_1164.all;

entity tb_mux4to1 is
end entity;

architecture sim of tb_mux4to1 is
signal d : std_logic_vector(3 downto 0) := (others => '0');
signal s : std_logic_vector(1 downto 0) := (others => '0');
signal y : std_logic;

-- Component Declaration
component mux4to1
port(
d : in std_logic_vector(3 downto 0);
s : in std_logic_vector(1 downto 0);
y : out std_logic
);
end component;

begin

-- Instantiate the Unit Under Test (UUT)
```

```
UUT: mux4to1
port map(
d => d,
s => s,
y => y
);

-- Stimulus Process
stim_proc:  process
begin

-- Test Case 1
d <= "0001"; s <= "00"; wait for 10 ns;
-- Test Case 2
d <= "0010"; s <= "01"; wait for 10 ns;
-- Test Case 3
d <= "0100"; s <= "10"; wait for 10 ns;
-- Test Case 4
d <= "1000"; s <= "11"; wait for 10 ns;

-- Additional Tests
d <= "1100"; s <= "00"; wait for 10 ns;
d <= "1100"; s <= "01"; wait for 10 ns;
d <= "1100"; s <= "10"; wait for 10 ns;
d <= "1100"; s <= "11"; wait for 10 ns;

wait;
end process;
end architecture sim;
```

2. **Program Statement for Student:** Implement 4:1 MUX using Dataflow Model

XI. Resources

Table 6.3 Resources

Sr. No.	Name of Resource	Specifications	Quantity
1			
2			
3			

XII. Actual Procedure

……

……

……

……

……

……

……

……

……

……

……

……

XIII. **Observation Table (**use blank sheet provided if space not sufficient)

Observations for 4:1 MUX for Problem Statement given to Student.

Table 6.4 Observation table

S0	S1	Y
0	0	
0	1	
1	0	
1	1	

XIV. Result

……

……

..……

XV. Interpretation of result

……

.....……

XVI. Conclusion and recommendation

……

……

……

XVII. Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Write VHDL behavioral code for a 8:1 MUX..
2. State the role of synthesis in FPGA programming
3. Write VHDL code 4:1 MUX using 2:1 MUX components.

[Space for Answers] (If required attach separate page)

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……
……
……
……
……
……
……
……
……
……
……

XVIII References/Suggestions for further reading

Table no.6.5 References

Sr. No.	Links/ Portal	Description
1	https://www.tutorialspoint.com/vlsi_design/vhdl_programming_for_combinational_circuits.htm	VHDL Programming Combinational Circuits
2	https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/	VHDL Modelling Styles: Behavioral, Dataflow, Structural
3	https://de-iitr.vlabs.ac.in/exp/multiplexer-demultiplexer/theory.html	Implementation of 4x1 multiplexer and 1x4 demultiplexer using logic gates.

XIX Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		**60 %**
1	Coding ability	30%
2	Debugging ability	10%
3	Follow ethical practices.	20%
Product Related: 10 Marks		**40%**
5	Correct connections to FPGA Board	20%
6	Answer to sample questions.	15%
7	Timely Submission , Answer to sample questions.	05%
Total (25 Marks)		**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Process Related (10)**	**Total (25)**	

Practical No.7: Realize the De-multiplexer on FPGA board I

Practical Significance

De-multiplexers (DEMUX) are essential components in digital systems where a single data input must be routed to multiple destinations based on select-line control. A 1:8 De-Multiplexer (DEMUX) is widely used in digital systems to route a single data input to one of eight output lines. By implementing the De-MUX in VHDL and downloading it onto an FPGA, students gain exposure to synthesizable HDL coding**,** pin assignment**,** and timing verification**.**

II Industry/Employer Expected Outcome

The aim of this course is to attend following industry/employer excepted outcome through various teaching learning experiences:
"Develop VLSI-based electronic circuit/component using VHDL."

III Course Level Learning Outcome

Use VHDL to develop and test digital circuits.

IV Laboratory Learning Outcome

LLO 7.1 Test the functionality of 1:8 De multiplexer using VHDL code.

V Relevant Affective Domain related outcomes

- Follow good coding and documentation practices for reproducible lab results.
- Maintain tools and equipment.
- Follow ethical practices.
- Work in pairs to cross-check test vectors and board results.

VI Relevant Theoretical Background

A DEMUX performs the reverse operation of a multiplexer. It takes one input (D) and sends it to one active output among eight outputs, depending on the three select lines (S2, S1, S0).
1:8 De multiplexer

- Input (Din): Single data input
- Select Lines (S2, S1, S0): Choose which output receives the data
- Outputs (Y0 to Y8): One active at a time

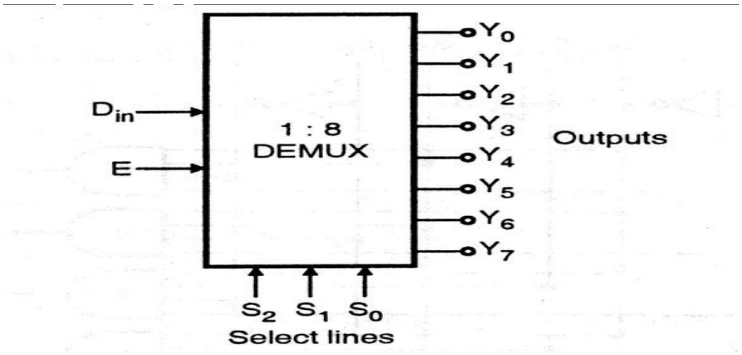


Fig. no. 7.1: Block Diagram of 1:8 Demultiplexer

- Truth Table of 1:8 DEMUX

Table 7.1: Truth Table of 1:8 Demultiplexer

S2	S1	S0	Activated Output	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	Y0 = D	0	0	0	0	0	0	0	D
0	0	1	Y1 = D	0	0	0	0	0	0	D	0
0	1	0	Y2 = D	0	0	0	0	0	D	0	0
0	1	1	Y3 = D	0	0	0	0	D	0	0	0
1	0	0	Y4 = D	0	0	0	D	0	0	0	0
1	0	1	Y5 = D	0	0	D	0	0	0	0	0
1	1	0	Y6 = D	0	D	0	0	0	0	0	0
1	1	1	Y7 = D	D	0	0	0	0	0	0	0

VII Circuit diagram

a) Suggested Practical Setup:

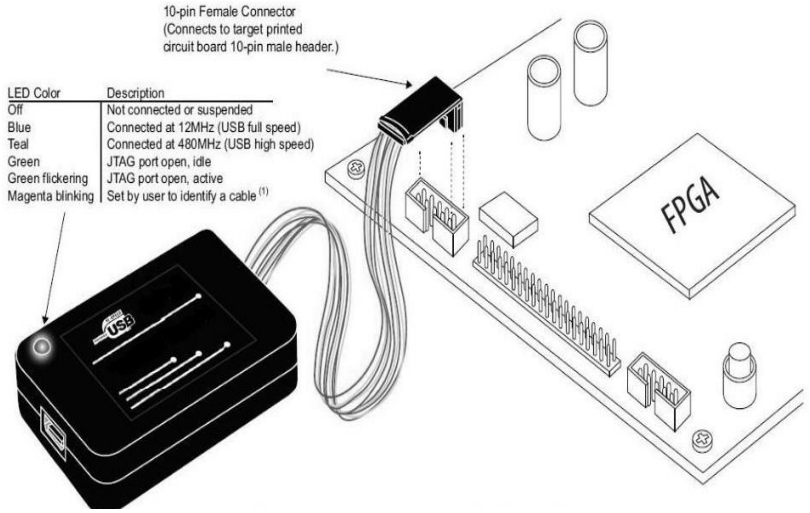


Fig. no. 7.2 Practical Setup with JTAG CABLE

b) Actual Practical Setup used in laboratory with related equipment rating:

VIII Required Resources/apparatus/equipment with specifications

Table no.7.2 Required Resources

Sr. No	Instrument / Components	Specification	Quantity
1	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208),On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit., On board, 2 Crystal 8MHz and 25MHz. .JTAG Interface (Boundary Scan),PROM Interface (XCF02S),40 pin, 4 header connector for external I/O's	1 No.
2	Desktop PC	Personal computer with latest configuration. Loaded with open-source IDE, simulation and program downloading software, UPS and Antivirus.	1 No.

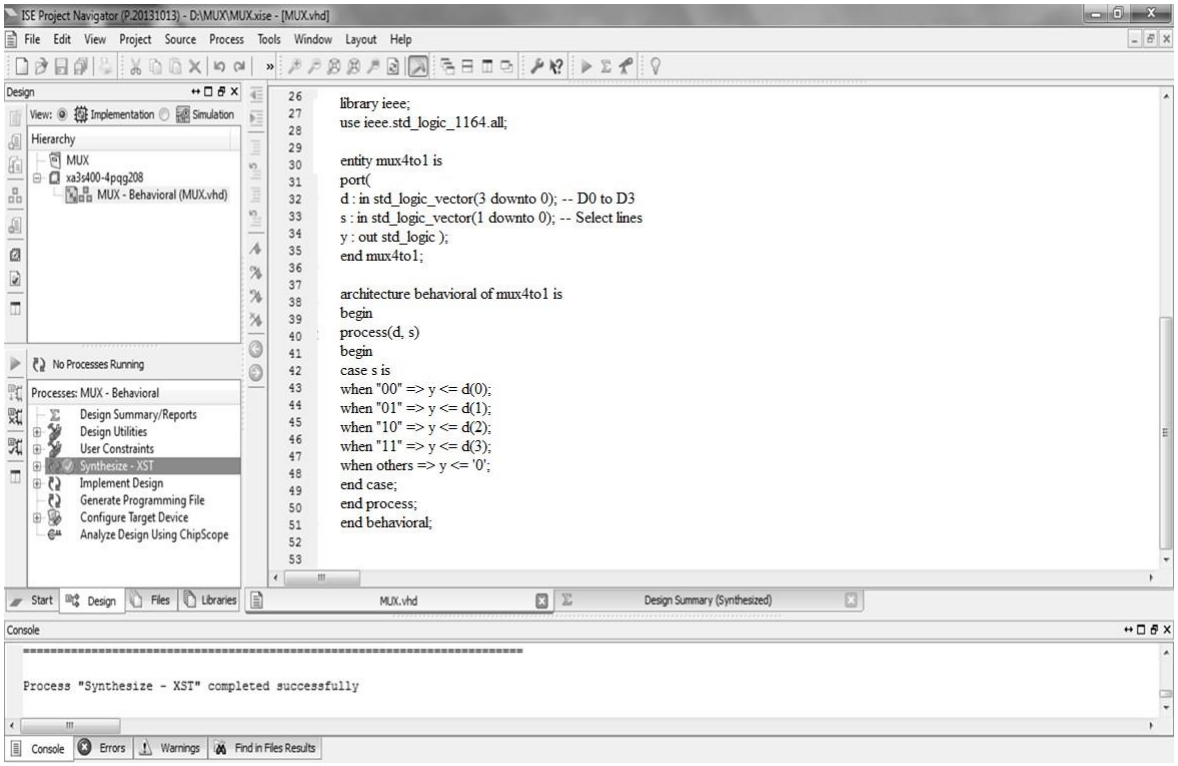
IX Precautions to be followed

1. Check the syntax / rules of VHDL Programming.
2. Do not power up the board before completing connections.

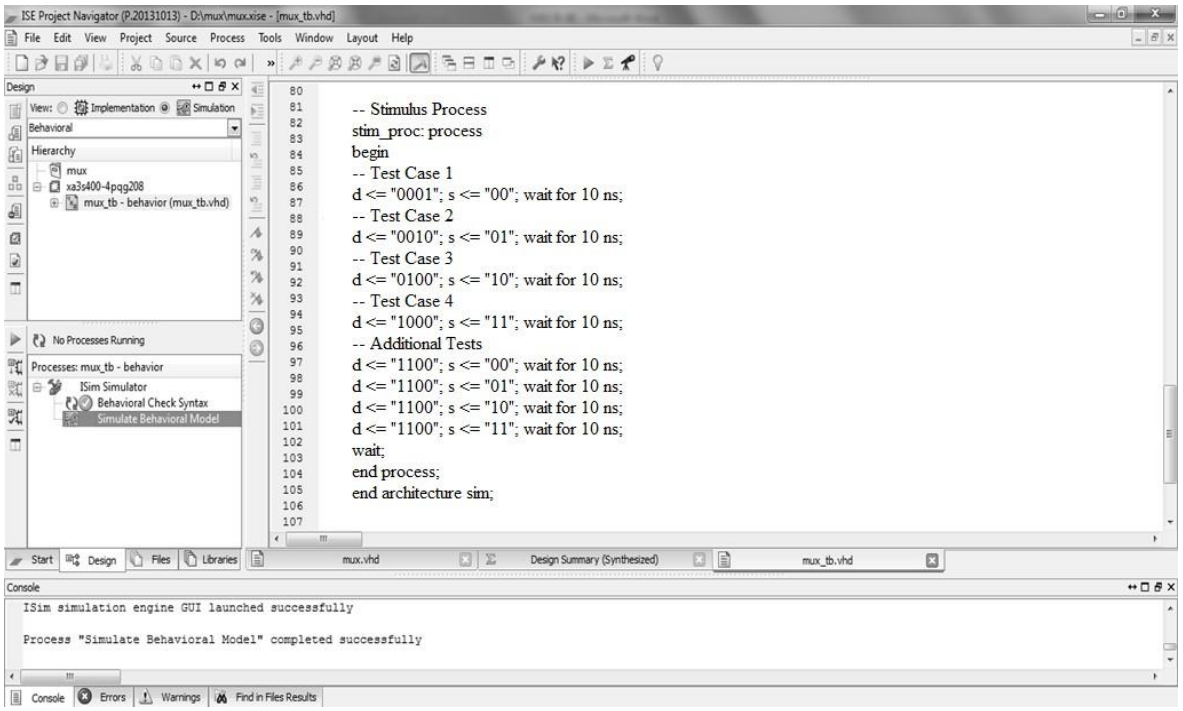
X Procedure

1. Create the Xilinx ISE project for your top-level FPGA design, by doing the following in ISE:
2. In the ISE software, select File > New Project. In the Project name and Project location fields, enter the project name and location, respectively.
3. Select HDL or Schematic as the Top-level source type, and click Next
4. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".
5. Define Module". Enter the entity used in design and then click "Next".
6. Develop VHDL code for given problem and synthesize the .vhd file and view RTL schematic.
7. Create Test Bench file- A test bench is HDL code that allows you to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing.
8. Go to implementation to simulation tab, right click on main source file and create Test Bench file for simulation.
9. In order to view test files, select the box of "Simulation" in the "View Panel" of the "Design" panel. In the "Process Panel," double click on the "Behavioral Check Syntax" to make sure that you didn't make any syntax errors while making changes. Double click on "Simulate Behavioral Model" in the "Process Pane", which will open the ISim software with your test bench loaded.
10. ISim simulator window will open with your simulation executed, where one can simulate designs and check errors if any.

a) Create 1:8 Demultiplexer .vhd file:



b) Create 1:8 Demultiplexer test bench file:



- **Sample VHDL Code to Implement 1:8 DEMUX**

1. **Behavioral Model:**

```
library ieee;
use ieee.std_logic_1164.all;

entity demux1to8 is
port(
d : in std_logic;
s : in std_logic_vector(2 downto 0);
y : out std_logic_vector(7 downto 0) );
end demux1to8;

architecture behavioral of demux1to8 is
begin
process(d, s)
begin
y <= (others => '0'); -- default all zero
case s is
when "000" => y(0) <= d;
when "001" => y(1) <= d;
when "010" => y(2) <= d;
when "011" => y(3) <= d;
when "100" => y(4) <= d;
when "101" => y(5) <= d;
when "110" => y(6) <= d;
when others => y(7) <= d;
end case;
end     process;
end behavioral;
```

2. **Testbench for 1:8 Demultiplexer**

```
library ieee;
use ieee.std_logic_1164.all;

entity tb_demux1to8 is
end tb_demux1to8;

architecture sim of tb_demux1to8 is
signal d : std_logic := '0';
signal s : std_logic_vector(2 downto 0) := (others => '0');
signal y : std_logic_vector(7 downto 0);

component demux1to8
port(
d : in std_logic;
s : in std_logic_vector(2 downto 0);
y : out std_logic_vector(7 downto 0)
);
end component;
```

```
begin
UUT: demux1to8
port map(
d => d,
s => s,
y => y );

stim_proc: process
begin
d <= '1';

s <= "000"; wait for 10 ns;
s <= "001"; wait for 10 ns;
s <= "010"; wait for 10 ns;
s <= "011"; wait for 10 ns;
s <= "100"; wait for 10 ns;
s <= "101"; wait for 10 ns;
s <= "110"; wait for 10 ns;
s <= "111"; wait for 10 ns;

d <= '0';
s <= "010"; wait for 10 ns;
s <= "111"; wait for 10 ns;

wait;
end process;
end sim;
```

2. **Program Statement for Student:** Implement 1:8 Demultiplexer using Dataflow Model

XI. Resources

Sr. No.	Name of Resource	Specifications	Quantity
1			
2			
3			

XII. Actual Procedure

……

……

……

……

……

……

……

……

……

……

XIII. Observation Table (use blank sheet provided if space not sufficient)

Observations for 1:8 Demultiplexer for Problem Statement given to Student.

Table 7.3 Observation table

Input			Data	Output							
S2	S1	S0	D	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	1								
0	0	1	1								
0	1	0	1								
0	1	1	1								
1	0	0	1								
1	0	1	1								
1	1	0	1								
1	1	1	1								

XIV. Result

……

……

……

XV. Interpretation of result

……

……

XVI. Conclusion and recommendation

……

……

XVII. Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Modify the VHDL code to implement an active-high enable input
2. Explain the behavior of the circuit if the select lines take undefined values (X or Z).
3. Write the VHDL code for 1:4 demultiplexer using the ‘if else” statement.

[Space for Answers] (If required attach separate page)

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XVIII References/Suggestions for further reading

Table 7.4 References

Sr . No	Link	Description
1	https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/	Dataflow Style of Modelling, Behavioral Style of Modelling, Structural Style of Modelling, RTL Design:
2	https://www.tutorialspoint.com/vlsi_design/vhdl_programming_for_combinational_circuits.htm	VHDL Programming Combinational Circuits
3	https://www.rfwireless-world.com/source-code/VHDL/1X8-DEMUX-vhdl-code.html	1x8 Demultiplexer VHDL Source Code
4	http://vhdlbynaresh.blogspot.com/2013/07/design-of-1-8-demultiplexer-using-when.html	Design of 1 : 8 Demultiplexer Using When-Else (VHDL Code).

XIX Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		**60 %**
1	Coding ability	30%
2	Debugging ability	10%
3	Follow ethical practices.	20%
Product Related: 10 Marks		**40%**
5	Correct connections to FPGA Board	20%
6	Answer to sample questions.	15%
7	Timely Submission , Answer to sample questions.	05%
Total (25 Marks)		**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Process Related (10)**	**Total (25)**	

Practical No.8: Design 4:2 encoder on FPGA board

I Practical Significance

A 4:2 encoder is a key combinational circuit widely used in digital systems to convert one of several active inputs into a smaller set of binary outputs. They are mainly used in logical circuits as well as data transfer circuits. This practical will help the students to gain experience with FPGA synthesis, pin assignment, and testing, Develop skills for implementing digital components used in processors, interrupt controllers, and communication systems.

II Industry/Employer Expected Outcome

The aim of this course is to attend following industry/employer excepted outcome through various teaching learning experiences:
"Develop VLSI-based electronic circuit/component using VHDL."

III Course Level Learning Outcome

Use VHDL to develop and test digital circuits.

IV Laboratory Learning Outcome

LLO 8.1 Interpret the output of 4:2 encoder using VHDL code.

V Relevant Affective Domain related outcomes

- Follow good coding and documentation practices for reproducible lab results.
- Maintain tools and equipment.
- Follow ethical practices.
- Work in pairs to cross-check test vectors and board results.

VI Relevant Theoretical Background

Encoders are crucial components in various digital electronics applications such as data transmission, controlling and automation, communication, signal processing, etc.An encoder consists of a certain number of input and output lines. Where, an encoder can have maximum of "2^n" input lines whereas "n" output lines. Hence, an encoder encodes information represented by "2^n" input lines with "n" bits.
A 4 to 2 Encoder is a type of encoder which has 4 (22) input lines and 2 output lines. It produces an output code (i.e., convert input information in a 2-bit format) depending on the combination of input lines. The block diagram of a 4 to 2 Encoder is shown in the following figure.

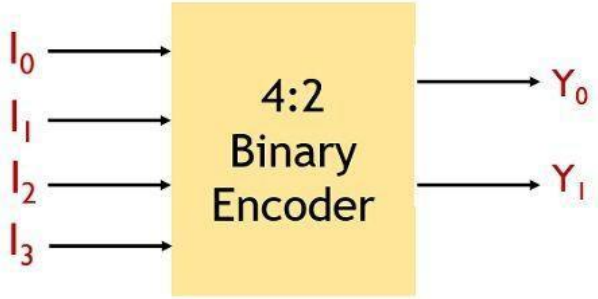


Fig 8.1: Block diagram of 4:2 encoder

- **Truth Table for 4:2 Encoder**

Table 8.1 Truth Table for 4:2 encoder

Input				Output	
I3	**I2**	**I1**	**I0**	**Y1**	**Y0**
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

VII Circuit diagram

a) Suggested Practical Sample:

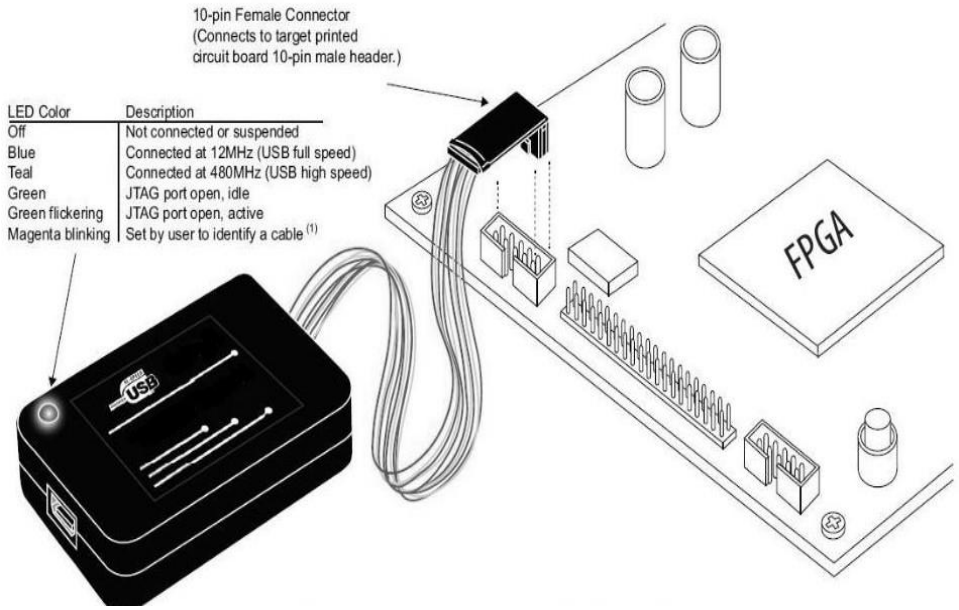


Fig. no. 8.2 Practical Setup with JTAG CABLE

b) Actual Practical Setup used in laboratory with related equipment rating:

VIII Required Resources/apparatus/equipment with specifications

Table no.8.2 Required Resources

Sr. No.	Instrument / Components	Specification	Quantity
1.	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208),On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit., On board, 2 Crystal 8MHz and 25MHz. .JTAG Interface (Boundary Scan),PROM Interface (XCF02S),40 pin, 4 header connector for external I/O's	1 No.
2.	Desktop PC	Personal computer with latest configuration. Loaded with open-source IDE, simulation and program downloading software, UPS and Antivirus..	1 No.

IX Precautions to be followed

1. Check the syntax / rules of VHDL Programming.
2. Do not power up the board before completing connections.

X Procedure

1. Create the Xilinx ISE project for your top-level FPGA design, by doing the following in ISE:
2. In the ISE software, select File > New Project. In the Project name and Project location fields, enter the project name and location, respectively.
3. Select HDL or Schematic as the Top-level source type, and click Next
4. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".
5. Define Module". Enter the entity used in design and then click "Next".
6. Develop VHDL code for given problem and synthesize the .vhd file and view RTL schematic.
7. Create Test Bench file- A test bench is HDL code that allows you to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing.
8. Go to implementation to simulation tab, right click on main source file and create Test Bench file for simulation.
9. In order to view test files, select the box of "Simulation" in the "View Panel" of the "Design" panel. In the "Process Panel," double click on the "Behavioral Check Syntax" to make sure that you didn't make any syntax errors while making changes. Double click on "Simulate Behavioral Model" in the "Process Pane", which will open the ISim software with your test bench loaded.
10. ISim simulator window will open with your simulation executed, where one can simulate designs and check errors if any.

a) 4:2 encoder .vhd file diagram :-

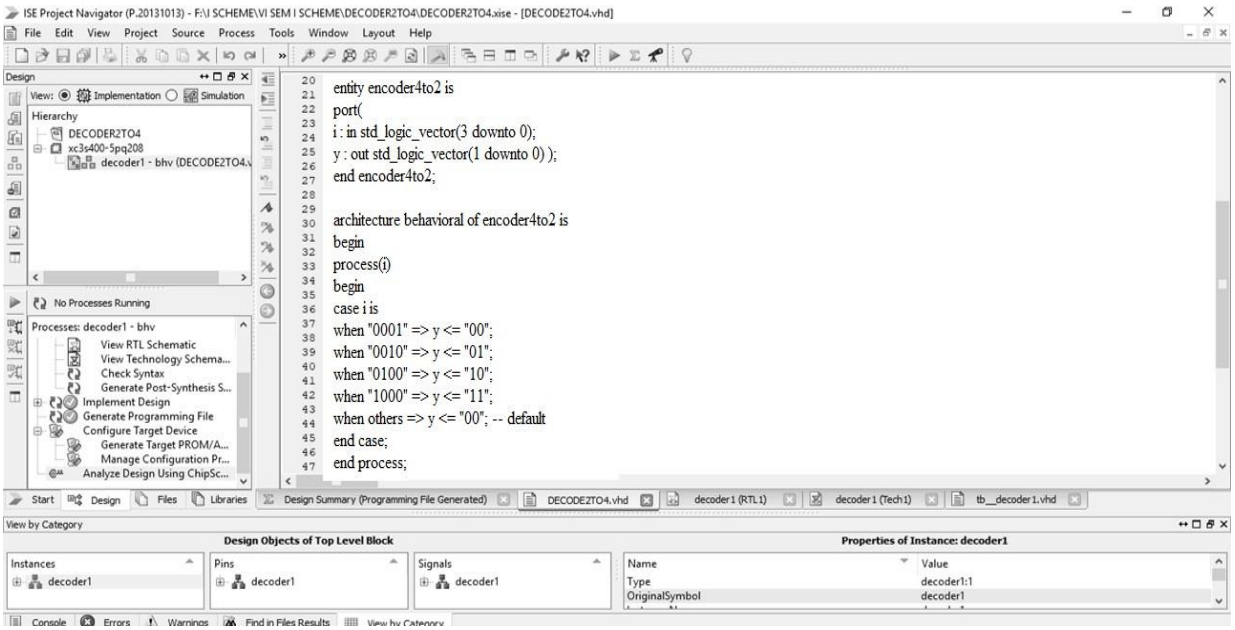


Fig. no. 8.3. 4:2 encoder.vhd file

b) 4:2 encoder test bench file

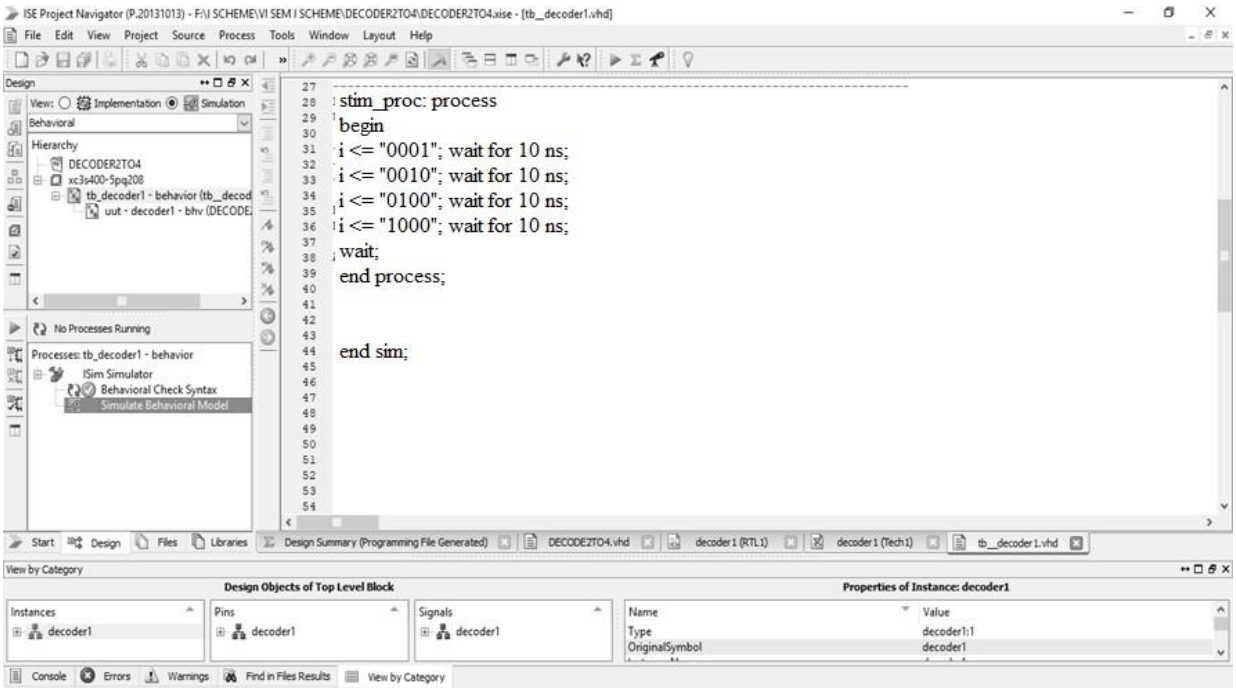


Fig. no. 8.4. 4:2 encoder Test bench file

c) 4:2 encoder test bench Simulation waveforms

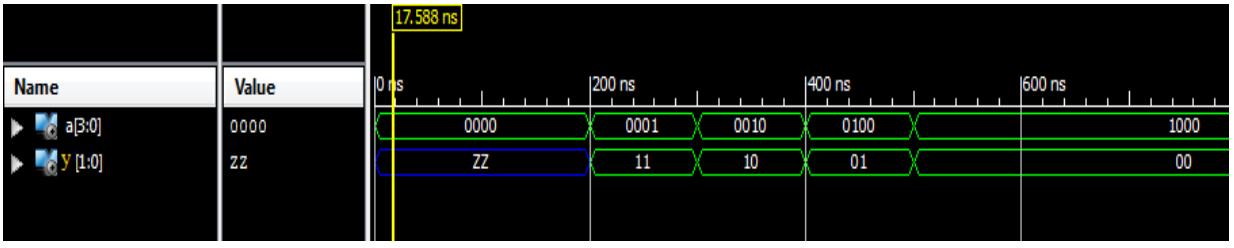


Fig. no. 8.5. 4:2 encoder Simulation waveforms

- **Sample VHDL Code to Implement 4 to 2 encoder**

1. **VHDL Code using if else statement (Behavioral Model).:**

```
library ieee;
use ieee.std_logic_1164.all;

entity encoder4to2 is
port(
i : in std_logic_vector(3 downto 0);
y : out std_logic_vector(1 downto 0)
);
end encoder4to2;

architecture behavioral of encoder4to2 is
begin
process(i)
begin
case i is
when "0001" => y <= "00";
when "0010" => y <= "01";
when "0100" => y <= "10";
when "1000" => y <= "11";
when others => y <= "00"; -- default
end case;
end process;
end behavioral;
```

2. **VHDL Testbench for 2 to 4 decoder:**

```
library ieee;
use ieee.std_logic_1164.all;

entity tb_encoder4to2 is
end tb_encoder4to2;

architecture sim of tb_encoder4to2 is
signal i : std_logic_vector(3 downto 0) := (others => '0');
signal y : std_logic_vector(1 downto 0);

component encoder4to2
port(
i : in std_logic_vector(3 downto 0);
y : out std_logic_vector(1 downto 0)
);
end component;
begin

UUT : encoder4to2
port map(
i => i,
y => y );
```

```
stim_proc: process
begin
i <= "0001"; wait for 10 ns;
i <= "0010"; wait for 10 ns;
i <= "0100"; wait for 10 ns;
i <= "1000"; wait for 10 ns;
wait;
end process;
end sim;
```

3. **Program Statement for Student:** Design VHDL Code for 8 to 3 encoder using if else statement

XI. Resources

Table no.8.3 Resources

Sr. No.	Name of Resource	Specifications	Quantity
1			
2			
3			

XII. Actual Procedure

……

……

……

……

……

……

……

……

……

……

XIII. Observation Table (use blank sheet provided if space not sufficient)

Observations for 8 to 3 encoder for Problem Statement given to Student.

Table 8.4 Observation table

INPUT		OUTPUT			
A1	A2	B3	B2	B1	B0
0	0				
0	1				
1	0				
1	1				

XIV. Result

……

……

XV. Interpretation of result

……

.....……

XVI. Conclusion and recommendation

……

……

XVII. Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Design VHDL Code using FPGA board for 8 to 3 encoder using case statement.
2. Design VHDL Code using FPGA board for 16 to 4 encoder using if else statement.
3. Design VHDL Code using FPGA board for 16 to 4 encoder using case statement

[Space for Answers] (If required attach separate page)

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XVIII References/Suggestions for further reading

Table no.8.3 References

Sr . No	Link / Portal	Description
1	https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/	Dataflow Style of Modelling, Behavioral Style of Modelling, Structural Style of Modelling, RTL Design:
2	https://www.tutorialspoint.com/vlsi_design/vhdl_programming_for_combinational_circuits.htm	VHDL Programming Combinational Circuits
3	https://www.tutorialspoint.com/vlsi_design/vlsi_design_vhdl_introduction.htm	VLSI Design - VHDL Introduction, VLSI Design Useful Resources
4	https://www.allaboutcircuits.com/technical-articles/sequential-vhdl-if-and-case-statements/	Sequential VHDL: If and Case Statements

XIX Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		**60 %**
1	Coding ability	30%
2	Debugging ability	10%
3	Follow ethical practices.	20%
Product Related: 10 Marks		**40%**
5	Correct connections to FPGA Board	20%
6	Answer to sample questions.	15%
7	Timely Submission , Answer to sample questions.	05%
Total (25 Marks)		**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Process Related (10)**	**Total (25)**	

Practical No. 09: Implement 3:8 Decoders on FPGA Board.

I Practical Significance

In electronics, a decoder is a digital circuit that detects a specified combination of input bits (code) generally binary inputs and converts that code in a specified output levels. A decoder has n input lines and from one to 2^n output lines to indicate the presence of one or more n bit combinations. This practical will help the students to implement the 3:8 decoder using FPGA.

II Industry/Employer Expected Outcome

The aim of this course is to attend following industry/employer expected outcome through various teaching learning experiences-

"Develop VLSI-based electronic circuit/component using VHDL".

III Course Level Learning outcome

CO4- Develop VHDL program for given application.

IV Laboratory Learning Outcome

LLO 9.1 Interpret the output of 3:8 decoder using VHDL code.

V Relevant Affective domain related Outcome(s)

- Follow safety practices.
- Maintain tools and equipment.
- Follow ethical practices.

VI Relevant Theoretical Background

A Decoder is combinational logic circuit that converts n bit binary code into 2^n output channels in such way that only one output channel is activated for each one of the possible combinations of inputs.

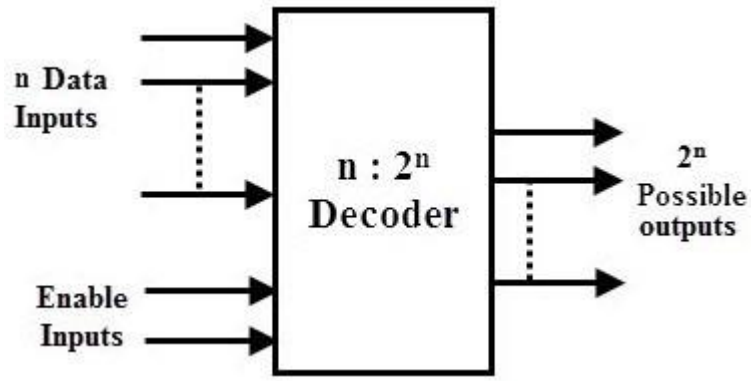


Fig. no. 9.1: Symbol for N:2^N Decoder

Here, the decoder has N input lines and M (2^N) output lines. In a decoder, each of the N input lines can be a 0 or a 1, hence the number of possible input combinations or codes be equal to (2^N).For each of these input combinations, only one of the M output lines will be active, and all other output lines will remain inactive.

3:8 Decoders: As shown in Fig. no..9.2, 3: 8 De-coder having three input lines and eight output lines. It has no select lines like demultiplexer which are used as the control signals.
When this decoder is enabled with the help of enable input E, then it's one of the eight outputs will be active for each combination of inputs. The operation of this 3-line to 8-line decoder can be analyzed with the help of its truth table which is given below in truth table 9.1 of 3:8 decoder.

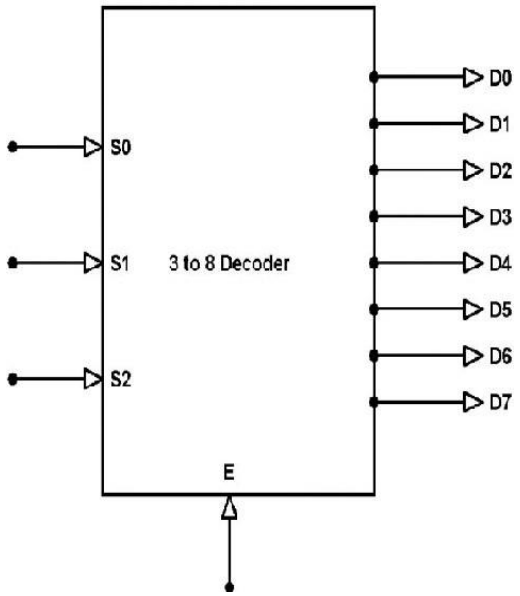


Fig. no. 9.2: 3:8 Decoder

Table 9.1: Truth Table of 3:8 Decoder

S0	S1	S2	E	D0	D1	D2	D3	D4	D5	D6	D7
x	x	x	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0
1	0	1	1	0	0	1	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

VII Circuit Diagram used in Laboratory with related equipment rating:

a) Suggested Practical JTAG cable setup:-

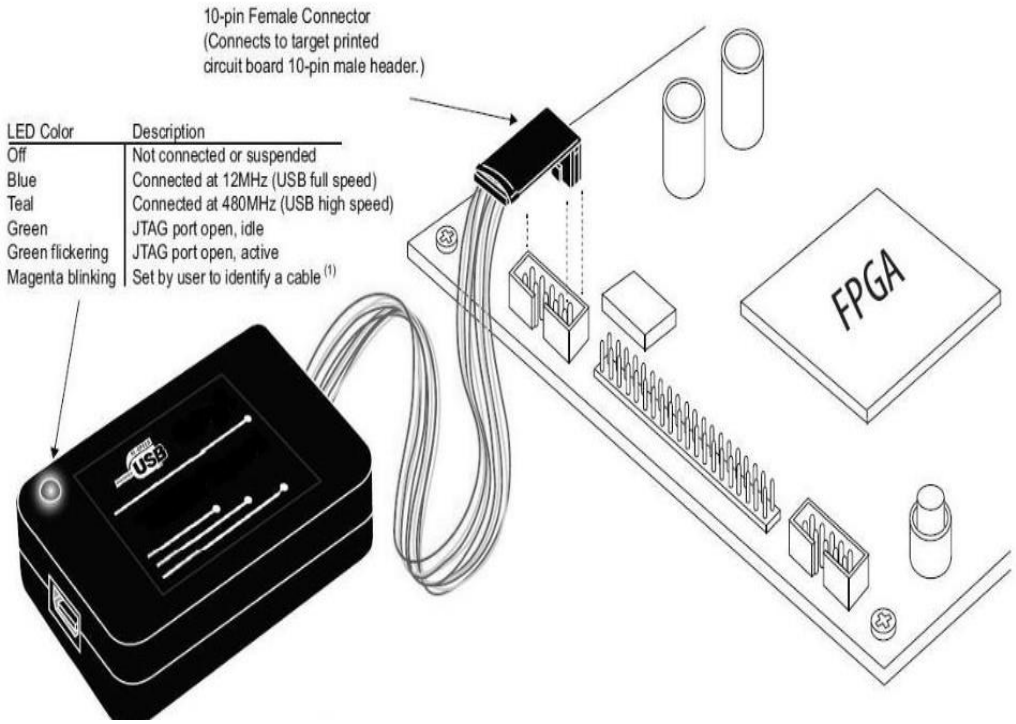


Fig. no. 9.3 a: Practical Setup with JTAG CABLE

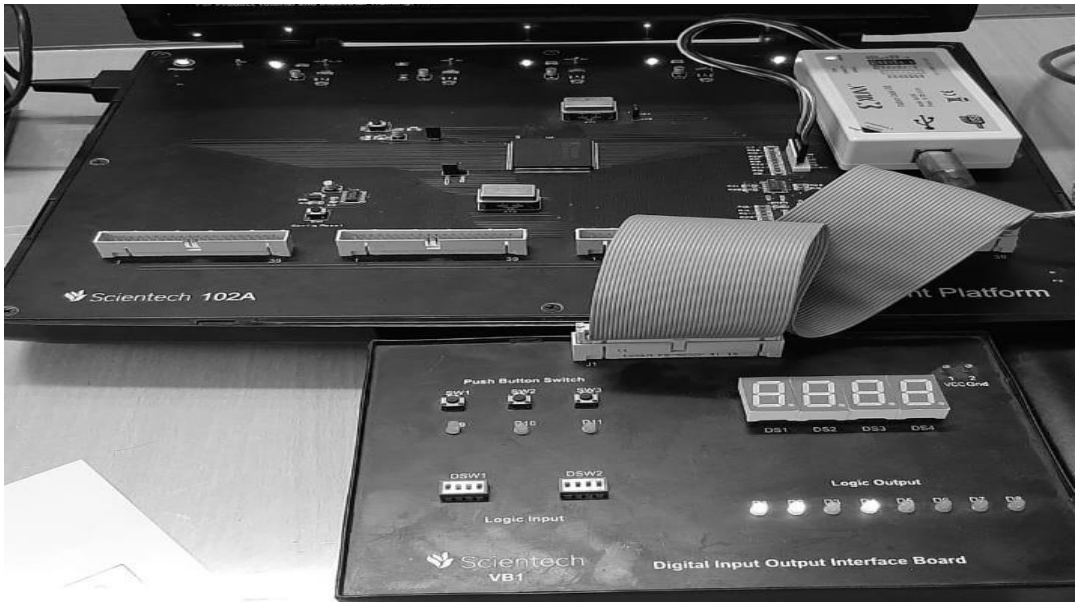


Fig. no.. 9.3.b: Lab Practical Setup

b) Create Decoder .vhd file:

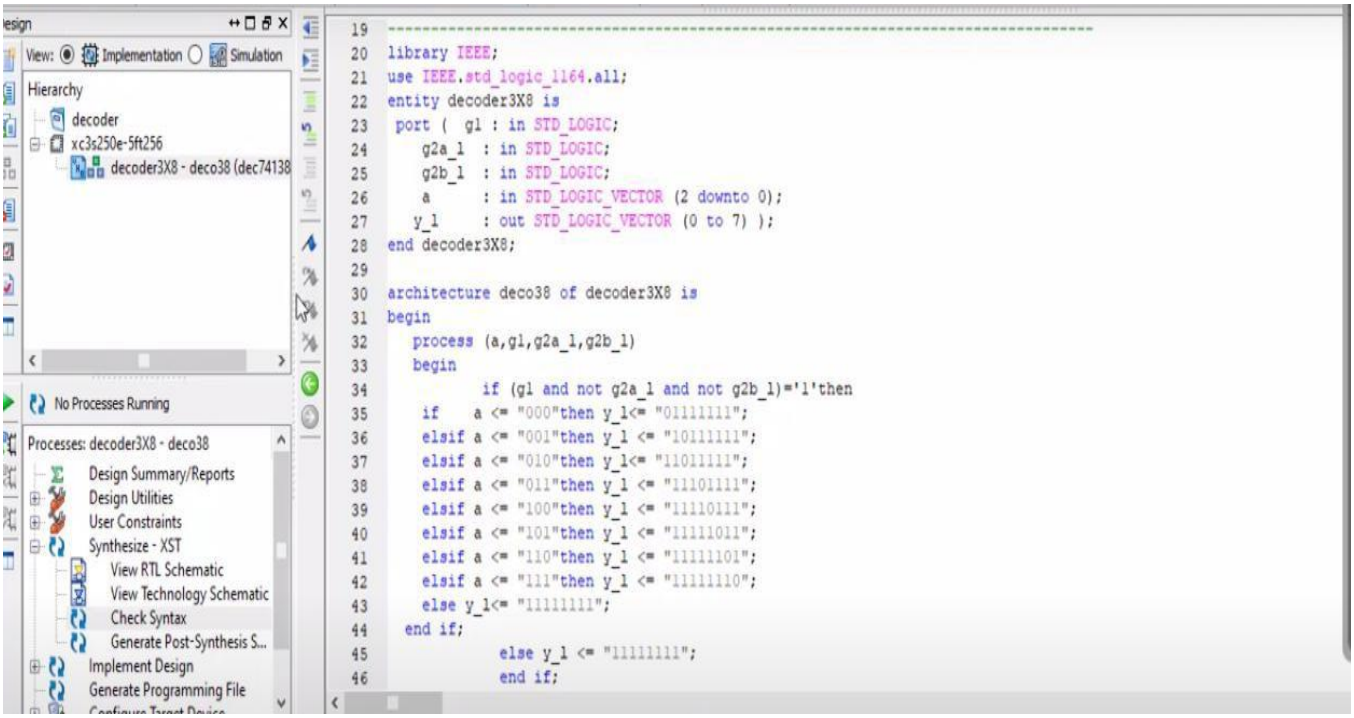


Fig. no. 9.4: Decoder .vhd file

c) Create Decoder test bench file:

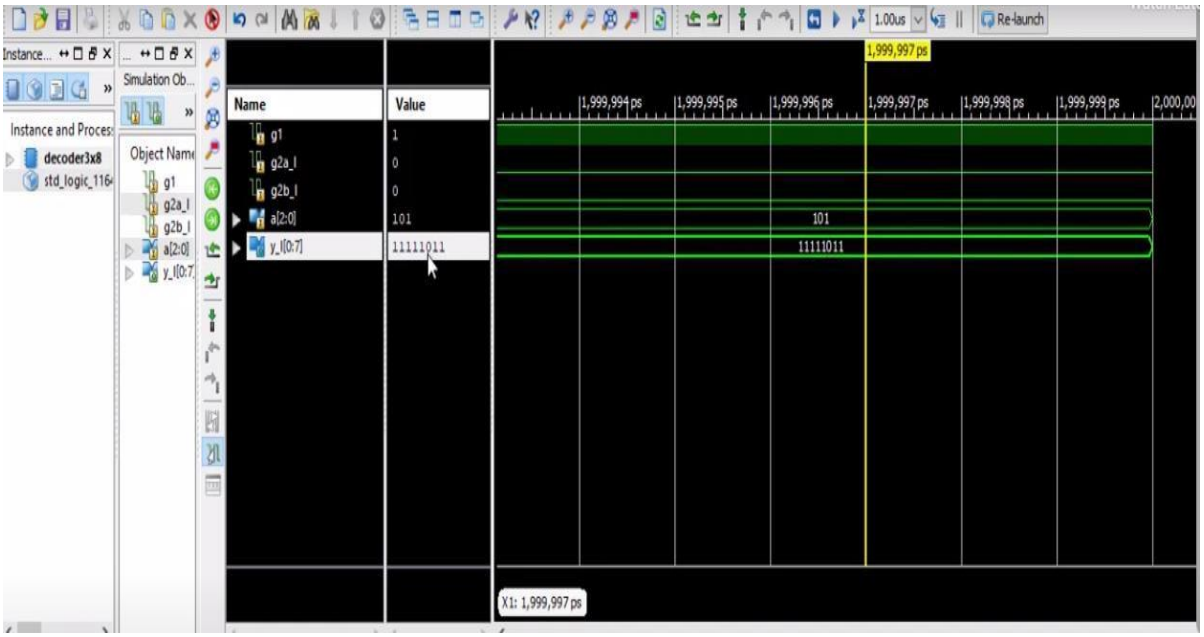


Fig. no. 9.5: Demux Test Bench file

d) Decoder test bench Simulation waveforms:

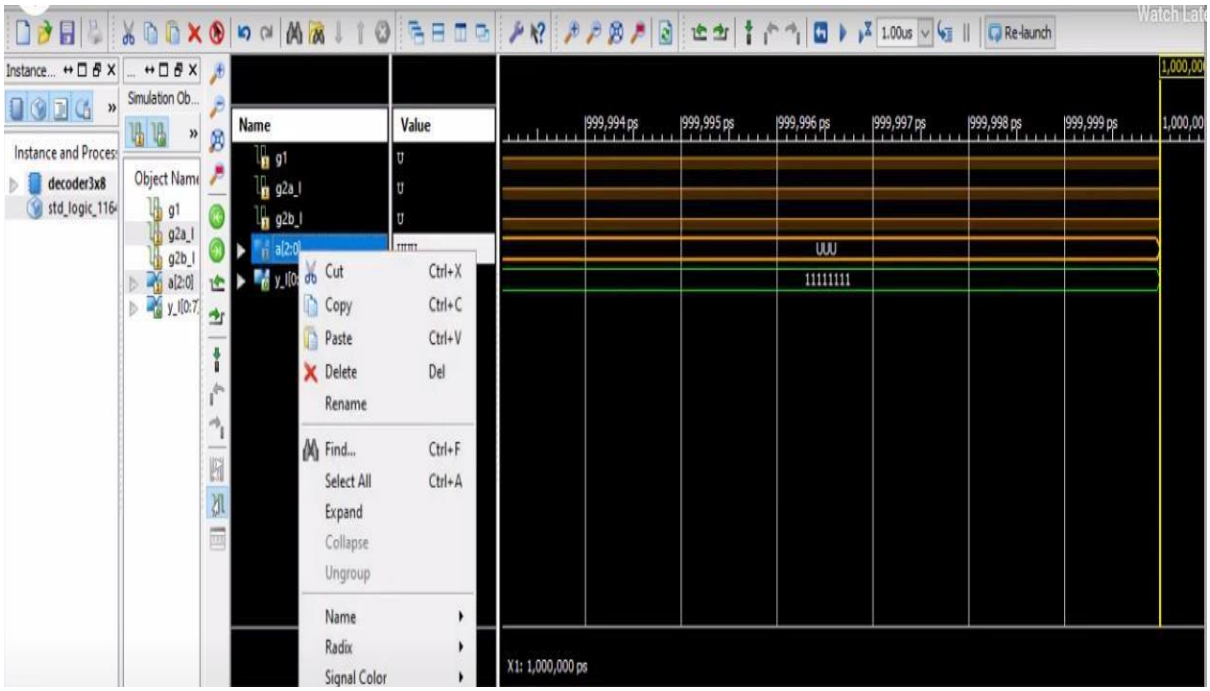


Fig. no. 9.6 Decoder Simulation output

e) Actual practical set up used in laboratory:

VIII Resources Required/Apparatus/Equipment with specifications:

Sr. No	Instrument / Components	Specification	Quantity
1.	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208),On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit., On board, 2 Crystal 8MHz and 25MHz.JTAG Interface (Boundary Scan),PROM Interface (XCF02S),40 pin, 4 header connector for external I/O's	1 No.
2.	Desktop PC	Personal computer with latest configuration. Loaded with open-source IDE, simulation and program downloading software, UPS and Antivirus.	1 No.

IX Precautions to be followed:

1. Check the syntax / rules of VHDL Programming.
2. Do not power up the board before completing connections.

X Procedure:

1. Create the Xilinx ISE project for your expected FPGA design, by doing the following in ISE:
    a. In the ISE software, select File > New Project.
    b. In the new project wizard, enter the project name and location, respectively.
    c. Select HDL or Schematic as the Top-level source type, and click Next.
2. Create New Source file. Select Source Type" selects the Source type and give the name to the
   source then click "Next".
3. Define Module". Enter the entity used in design and then click "Next".
4. Develop VHDL code for given problemand Synthesize the .vhd file and view RTL schematic.
5. Create Test Bench file- A **test bench** is **HDL** code that allows you to provide a documented, repeatable set of stimuli that is portable across different simulators. A **test bench** can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional **testing.**
6. Go to implementation to simulation tab , right click on main source file and create Test Bench file for simulation.
7. In order to view test files, select the box of "Simulation" in the "View Panel" of the "Design" panel. In the "Process Panel," double click on the "Behavioral Check Syntax" to make sure that you didn't make any syntax errors while making changes.
8. Double click on "Simulate Behavioral Model" in the "Process Pane", which will open the
   ISim software with your test bench loaded.
9. ISim simulator window will open with your simulation executed, as shown in Figure 9.6. where you are able to simulate your designs and check for errors.
10. After simulation implement 3 as 8 decoder Using FPGA development board.
11. Go to User Constraints – select Floorplan Area/IO/Logic (PlanAhead) – Windows – Properties
    – now assign pin configuration and save.
12. Go to Implement Design – Run all.

13. Go to Generate Programming Fileand Run
14. Go to Configure Target Deviceand Run – Impact wizard opened – select Device 2 (as per practical Kit) -- open Bit file -- and initialize Chain – Right click on Xilinx IC And select Program.

Sample Program: Implement 3:8 Decoder using dataflow modeling.

3:8 Decoder:

VHDL Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use
IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Decoder3to8 is
   Port ( S : in STD_LOGIC_VECTOR (2 downto
        0); Q : out STD_LOGIC_VECTOR (7
        downto 0));
end Decoder3to8;

architecture dataflow of Decoder3to8

is begin
  Q <= "00000001" when S="000" else
         "00000010" when S="001"
         else "00000100" when
         S="010" else "00001000"
         when S="011" else
         "00010000" when S="100"
         else "00100000" when
         S="101" else "01000000"
         when S="110" else
         "10000000";
end dataflow;
```

Program Statement for Student: Implement 2:4 Decoder.

2:4 Decoder

VHDL Code

XI Resources Require:

Sr. no.	Instrument /Components	Specification	Quantity
1.			
2.			

XII Actual Procedure (use blank sheet provided if space not sufficient)

……

……

……

……

……

……

……

……

……

XIII Observation Table (use blank sheet provided if space not sufficient)

Observations for Problem Statement given to Student.

Table 9.2: Truth Table of 3:8 Decoder

S0	S1	S2	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0								
0	0	1								
0	1	0								
0	1	1								
1	0	0								
1	0	1								
1	1	0								
1	1	1								

XIV Results

……..…….

……..…….

XV Interpretations of result

……..…….

……..…….

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results)

……..……

……..……

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. Give the syntax of With –select statement.
2. Write the VHDL code for 2:4 decoder using the ‘if else” statement.
3. Write the test bench code for3:8 decoder .

[Space for Answers] (If required attach separate page)

……..……

……..……

……..……

XVIII Suggested referencesfor further reading

Sr no.	Link/Portal	Description
1	https://www.tutorialspoint.com/digital-electronics/digital-electronics-decoders.htm	Decoder tutorial
2	https://www.engineersgarage.com/vhdl-tutorial-13-design-3x8-decoder-and-8x3-encoder-using-vhdl/	VHDL programming
3	https://www.elprocus.com/designing-3-line-to-8-line-decoder-demultiplexer/	ISE Quick start tutorial
4	http://ece2day.blogspot.com/2012/10/vhdl-tutorial-introducction-part-9.html	VHDL tutorial
5	https://www.allaboutcircuits.com/technical-articles/sequential-vhdl-if-and-case-statements/	VHDL programming
6	http://vhdlbynaresh.blogspot.com/2013/07/design-of-3-8-decoder-using-when.html	ISE Quick start tutorial

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related(15 Marks)		**60% (15)**
1	Coding ability	30%
2	Debugging ability	20%
3	Working with FPGA	10%
Product related (10 Marks)		**40%(10)**
4	Correct connections to FPGA Board	20%
5	Answer to sample questions.	15%
6	Timely Submission , Answer to sample questions.	05%
	TOTAL	**100% (25)**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No. 10: Realize T and D flip flop on FPGA Board.

I Practical Significance

A flip flop is an electronic circuit with two stable states that can be used to store binary data. The stored data can be changed by applying varying inputs. Flip-flops and latches are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems. This practical will help the students to implement the T and D flip flop using FPGA.

II Industry/Employer Expected Outcome

The aim of this course is to attend following industry/employer expected outcome through various teaching learning experiences.
Develop VLSI-based electronic circuit/component using VHDL.

III Course Level Learning Outcome

CO4 - Develop VHDL program for given application.

IV Laboratory Learning Outcomes

LLO 10.1 Test the functionality of D flipflop using VHDL code.
LLO 10.2 Test the functionality of T flipflop using VHDL code.

V Relevant Affective domain related Outcomes

a. Follow safety practices.
b. Maintain tools and equipment.
c. Follow ethical practices.

VI Relevant Theoretical Background

Flip-flops and latches are used as data storage elements. It is the basic storage element in sequential logic.

D type FF: D type FF has only one input referred to as D input or Data input. The input is applied to S/J and it is inverted before applied to either R/K input. the input applied at the D terminal will appear at the output after one clock pulse hence it is referred to as Delay or D FF. for this complete clock cycle as the input is hold this circuit is known as LATCH.

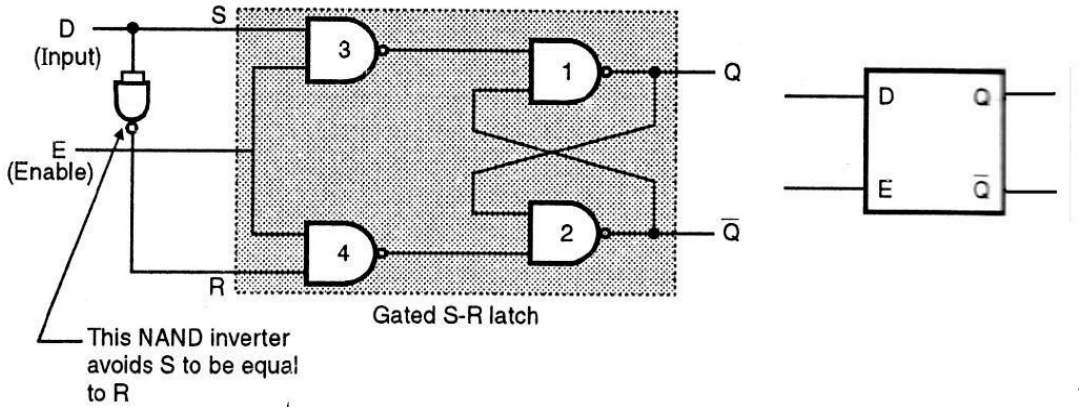


Fig. no. 10.1: D Flip Flop

Table 10.1: Truth Table for D Flip flop

Input D	Output Q
0	0
1	1

T Type FF: In a JK FF then the resulting FF is referred as T Type FF. it has only one input refereed as t input. If T=1 then it acts as toggle switch i.e. the output changes

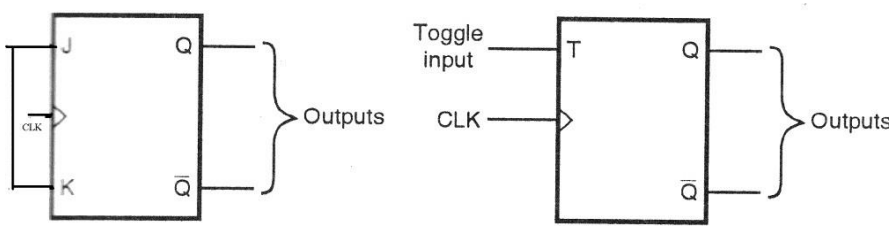


Fig. no. 10.2: T Flip Flop

Table 10.2: Truth Table for T Flip Flop

Input T	Output Q
0	Qn
1	$\overline{Qn}$

VII Circuit diagram used in Laboratory with related equipment rating.

a) **Suggested Practical Circuit Diagram:**

Practical JTAG cable setup:-

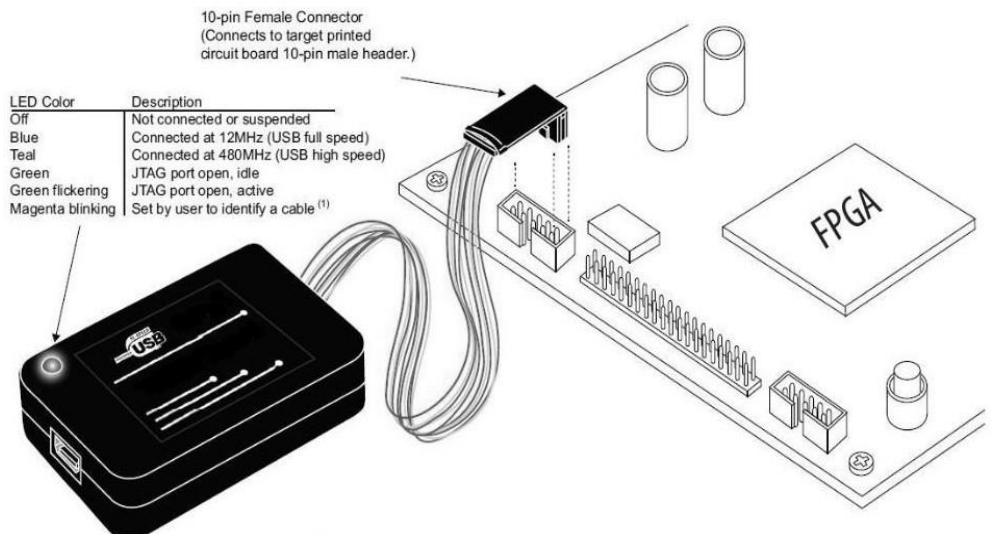


Fig. no. 10.3 a: Practical Setup with JTAG CABLE

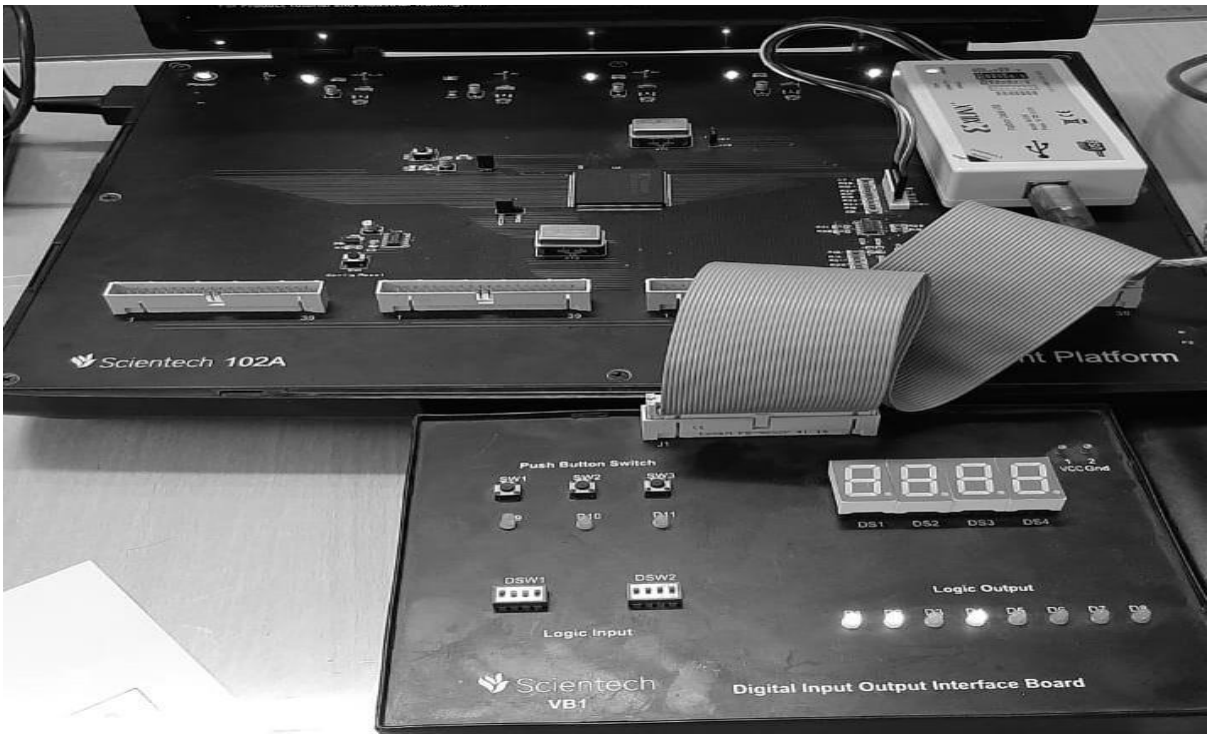


Fig. no.. No.10.3.b: Lab Practical Setup

b) Create DFF .vhd file:

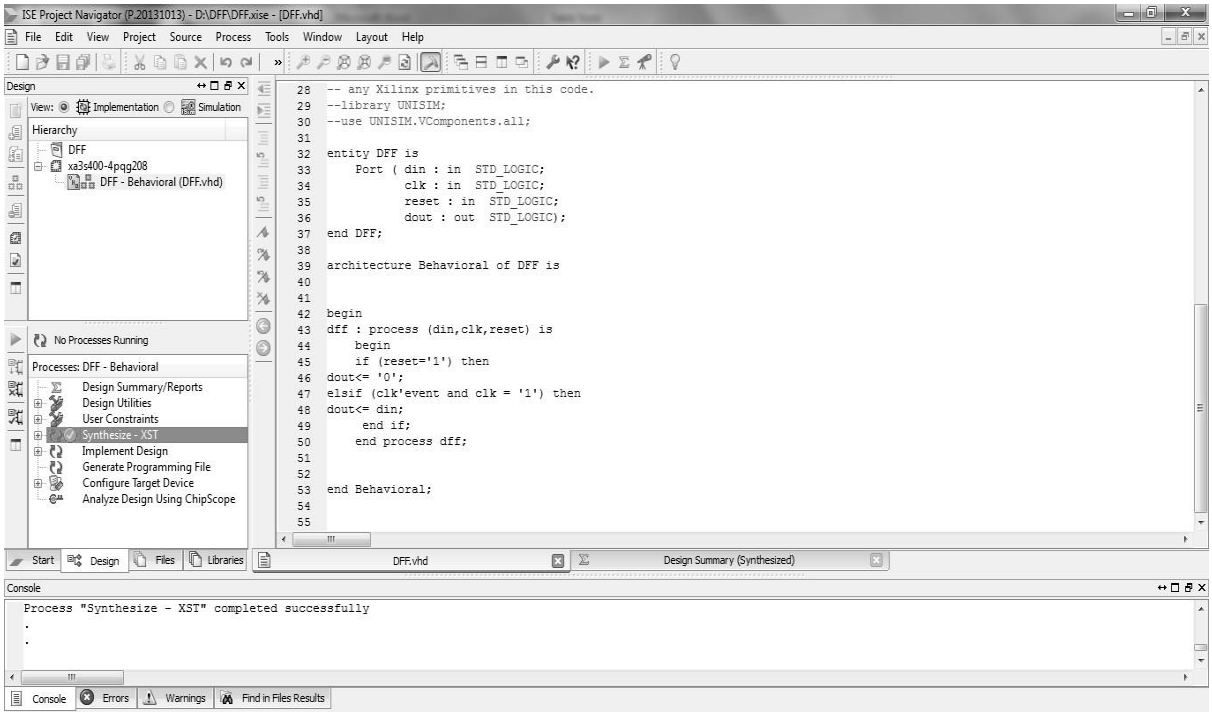


Fig. no. 10.4: DFF .vhd file

c) Create DFF test bench file:

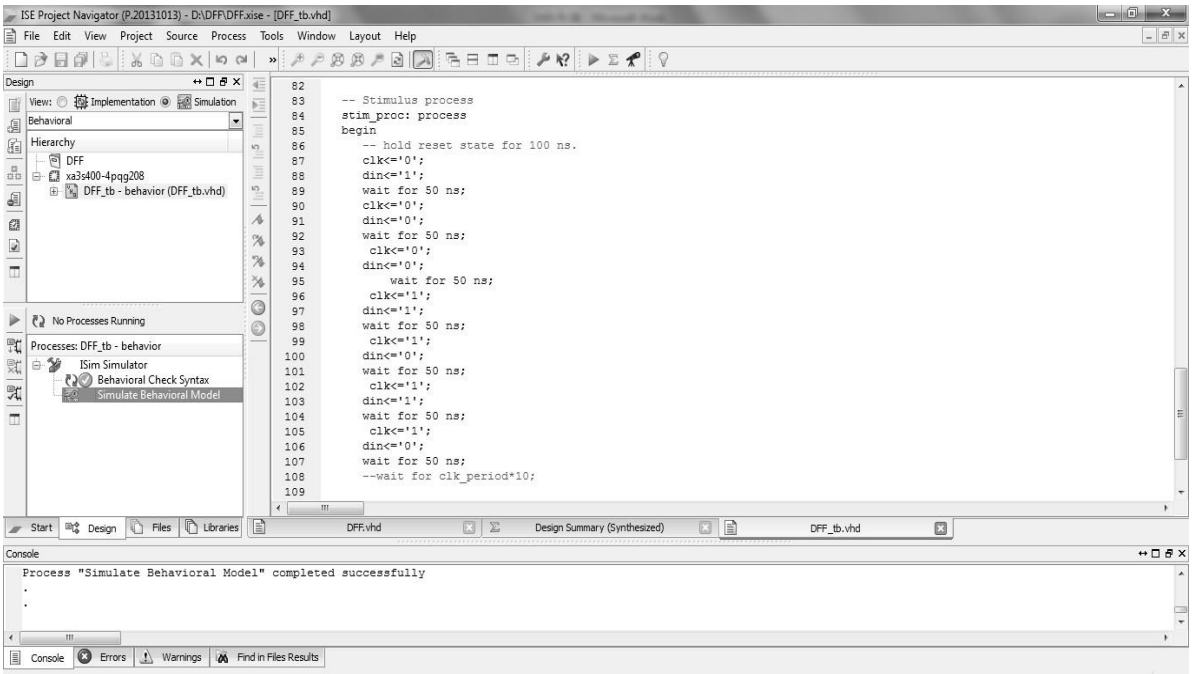


Fig. no. 10.5: DFF Test Bench file

d) DFF test bench Simulation waveforms:

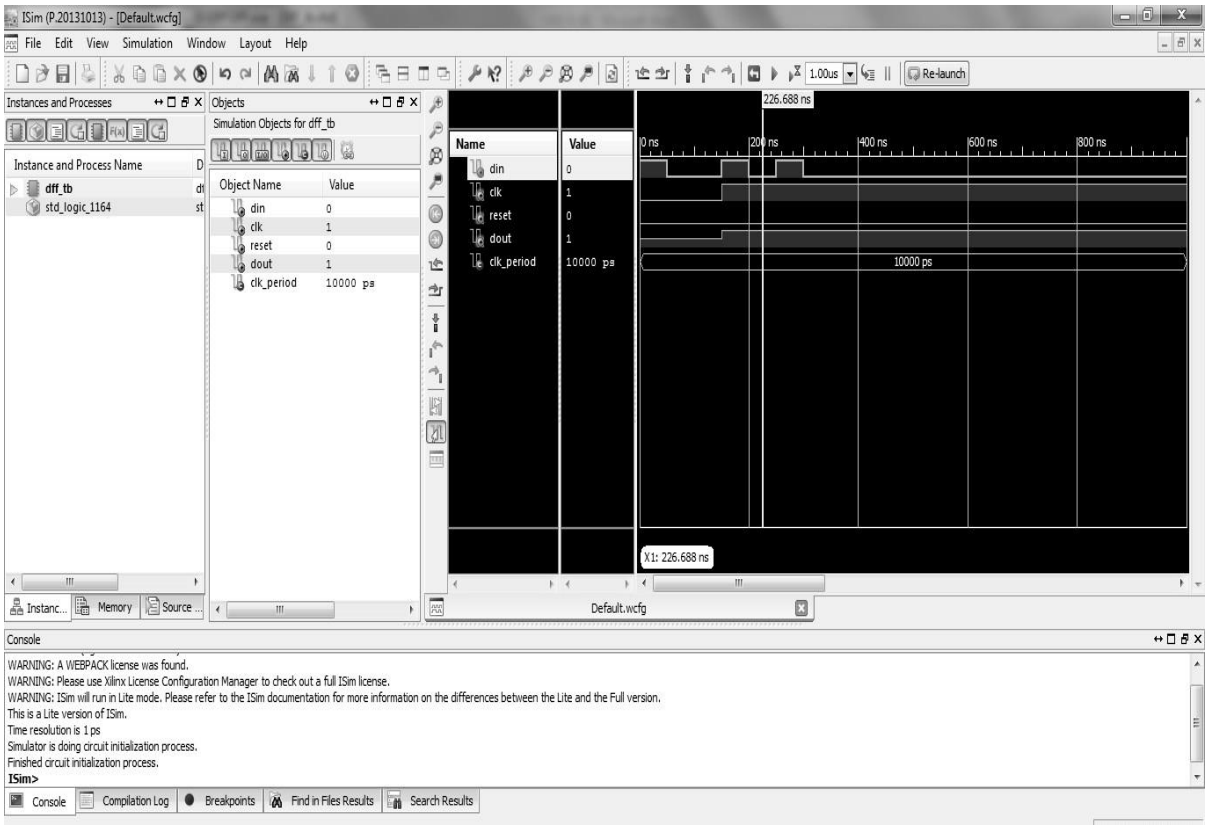


Fig. no. 10.6 DFF Simulation output

e) Actual Practical set up used in laboratory:

VIII Resources Required:

Sr. No.	Instrument / Components	Specification	Quantity
1	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208), On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit. On board, 2 Crystal 8MHz and 25MHz. JTAG Interface (Boundary Scan), PROM Interface (XCF02S), 40 pin, 4 header connector for external I/O's	1 No.
2	Desktop PC	Personal computer with latest configuration. Loaded with open-source IDE, simulation and program downloading software, UPS and Antivirus.	1 No.

IX Precautions to be followed:

a. Check the syntax / rules of VHDL Programming.
b. Do not power up the board before completing connections.

X Procedure:

1. Create the Xilinx ISE project for your expected FPGA design, by doing the following in ISE:
2. In the ISE software, select File > New Project.
3. In new project wizard, enter the project name and location, respectively.
4. Select HDL or Schematic as the Top-level source type, and click Next.
5. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".
6. Define Module". Enter the entity used in design and then click "Next".
7. Develop VHDL code for given problemand Synthesize the .vhd file and view RTL schematic.
8. Create Test Bench file- A test bench is HDL code that allows you to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing.
9. Go to implementation to simulation tab , right click on main source file and create Test Bench file for simulation.
10. In order to view test files, select the box of "Simulation" in the "View Panel" of the "Design" panel. In the "Process Panel," double click on the "Behavioral Check Syntax" to make sure that you didn't make any syntax errors while making changes.
11. Double click on "Simulate Behavioral Model" in the "Process Pane", which will open the ISim software with your test bench loaded.
12. ISim simulator window will open with your simulation executed, as shown in Fig. where you are able to simulate your designs and check for errors.
13. After simulation implement T and D Flipflop Using FPGA development board.
14. Go to User Constraints – select Floorplan Area/IO/Logic (Plan Ahead) – Windows – Properties
    – now assign pin configuration and save.
15. Go to Implement Design – Run all.

16. Go to Generate Programming Fileand Run
17. Go to Configure Target Deviceand Run – Impact wizard opened – select Device 2 (as per practical Kit) -- open Bit file -- and initialize Chain – Right click on Xilinx IC And select Program.

Sample Program: Implement T and D flip flop using FPGA

T Flip Flop :

VHDL Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity Toggle_flip_flop is
  port(
    t : in STD_LOGIC;
    clk : in STD_LOGIC;
    reset : in STD_LOGIC;
dout : out STD_LOGIC
  );
end Toggle_flip_flop;

architecture T_FFof Toggle_flip_flop is
begin
tff : process (t,clk,reset) is
variable m : std_logic : = '0';
begin
if (reset = '1') then
 m : = '0';
elsif (rising_edge (clk)) then
if (t = '1') then
m : = not m;
 end if;
end if;
dout< = m;
end process tff;
end T_FF;
```

D Flip Flop:

VHDL Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity d_flip_flop is
    port(
       din : in STD_LOGIC;
       clk : in STD_LOGIC;
       reset : in STD_LOGIC;
       dout : out STD_LOGIC);
```

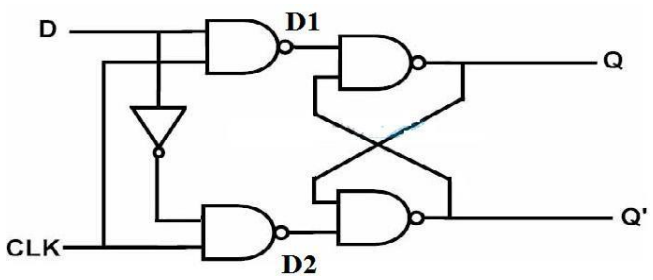
```
end d_flip_flop;

architecture D_FF of d_flip_flop is
begin
dff : process (din,clk,reset) is
   begin
   if (reset='1') then
dout<= '0';
elsif (clk'event and clk='1') then
dout<= din;
    end if;
   end process dff;
end D_FF;
```

Problem Statement for Student:

1. Write the VHDL code for D flip flop with respect to the structure shown:

VHDL Code

2. Using dataflow method i.e. finds characteristic equation T=? D=?



XI Resources:

Sr. no.	Instrument /Components	Specification	Quantity
1.			
2.			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

……...……..……

……...……...

……...……..…

……...……..…

……...……..…

……...…..…

……..

……....……

……...……..……

……..…

……..…

……...

……...

……..……..……..…

……...……..……

XIII Observations (use blank sheet provided if space not sufficient)

Observations for Problem Statement given to Student.

Table 10.3: Truth Table for D Flip Flop

CLK	D	Y
0	**0**	
1	**0**	
1	**1**	

XIV Results (output of Program)

……

……

……

……

XV Interpretations of result

……

……

XVI Conclusions and Recommendation

……

……

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. Write the VHDL code for T flip flop using structural modeling and Data flow method.
2. Write the VHDL test bench code for T Flip Flop.
3. Compare data flow and Behavioral method

[Space for Answers] (If required attach separate page)

……

……

……

……

……

……

……

……

……

……

……

……

XVIII Suggested referencesfor further reading

Sr no.	Link/Portal	Description
1	https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/	VHDL programming
2	https://www.tutorialspoint.com/vlsi_design/vlsi_design_vhdl_introduction.htm	VHDL programming
3	https://www.tutorialspoint.com/vlsi_design/vhdl_programming_for_sequential_circuits.htm	ISE Quick start tutorial
4	http://ece2day.blogspot.com/2012/10/vhdl-tutorial-introducction-part-7.html	VHDL tutorial

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related(15 Marks)		**60% (15)**
1	Coding ability	30%
2	Debugging ability	20%
3	Working with FPGA.	10%
Product related (10 Marks)		**40%(10)**
4	Correct connections to FPGA Board	20%
5	Answer to sample questions.	15%
6	Timely Submission , Answer to sample questions.	05%
	TOTAL	**100% (25)**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No. 11: Design comparator on FPGA board.

I Practical Significance

Comparator is a combinational logic circuit that is used to compare the magnitudes of two binary numbers. Comparators are used in several different electronic circuits like analog to digital converters, voltage level detectors, zero-crossing detectors, etc.. This practical will help the students to develop programming skills i.e. Implement 2 bit comparator on FPGA board using Xilinx ISE tools.

II Industry/Employer Expected Outcome

The aim of this course is to attend following industry/employer expected outcome through various teaching learning experiences

Develop VLSI-based electronic circuit/component using VHDL.

III Course Level Learning outcome

CO4- Develop VHDL program for given application.

IV Laboratory Learning Outcome

LLO 11.1 Test the functionality of 2-bit comparator using VHDL code.

V Relevant Affective domain related Outcome(s)

- Follow safety practices.
- Maintain tools and equipment.
- Follow ethical practices.

VI Relevant Theoretical Background

Depending on the number of bits, the following are some main types of comparators used in digital circuits

- 1 bit comparator
- 2 bit comparator
- 4 bit comparator

Consider the 2-bit magnitude comparator compares the values represented by two 2-bit binary numbers and then generates an output that indicates whether one number is equal to or greater than or less than the other.

Fig. no..: 11.1 Block diagram of 2-Bit Comparator

Table 11.1: Truth table of 2 Bit Comparator

Inputs				Outputs		
A_1	A_0	B_1	B_0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

The Boolean expression for the outputs after solving k maps for following cases

Case-1 (A>B) output= $\overline{A1}\ B1 + (A1 \odot B1)\ \overline{A0}\ B0$

Case-2 (A=B) output= $(A0 \odot B0)\ (A1 \odot B1)$

Case-3 (A<B) output= $A1\ \overline{B1} + (A1 \odot B1)\ A0\ \overline{B0}$

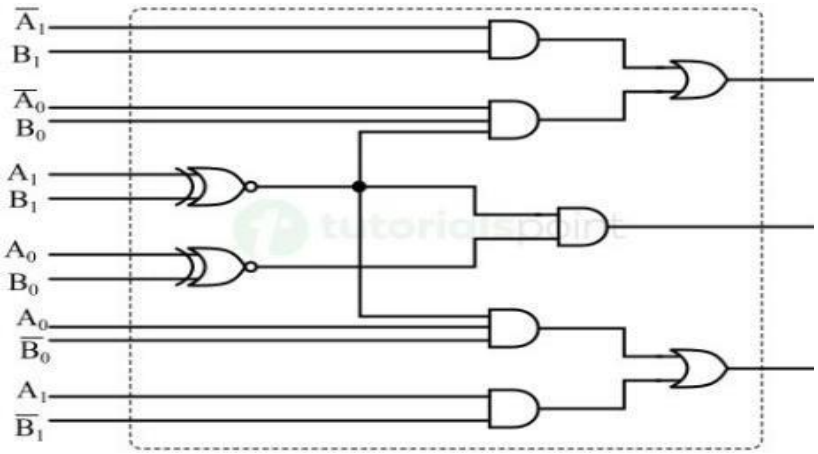


Fig. no.: 11.2 Logic diagram of 2 Bit Comparator

VII Circuit Diagram used in Laboratory with related equipment rating.

a) Suggested Practical Circuit Diagram:

Practical JTAG cable setup:-

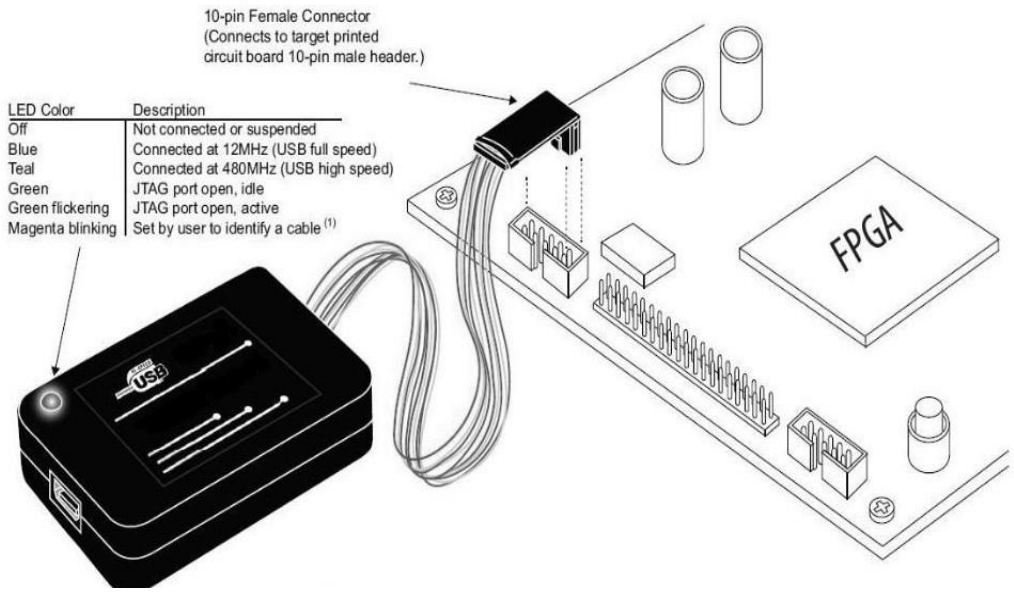


Fig. no. 11.3 a: Practical Setup with JTAG CABLE

Fig. no. 11.3.b: Lab Practical Setup

b) Actual Practical set up used in laboratory:

VIII Required Resources /Apparatus/Equipment with specifications

Table 11.2 Required Resources

Sr. No.	Instrument / Components	Specification	Quantity
1	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208), On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit. On board, 2 Crystal 8MHz and 25MHz. JTAG Interface (Boundary Scan), PROM Interface (XCF02S), 40 pin, 4 header connector for external I/O's	1 No.
2	Desktop PC	Personal computer with latest configuration. Loaded with open-source IDE, simulation and program downloading software, UPS and Antivirus.	1 No.

IX Precautions to be followed:

a. Check the syntax / rules of VHDL Programming.
b. Do not power up the board before completing connections.

X Procedure

1. Create the Xilinx ISE project for your expected FPGA design, by doing the following in ISE:
2. In the ISE software, select File > New Project.
3. In new project wizard, enter the project name and location, respectively.
4. Select HDL or Schematic as the Top-level source type, and click Next.
5. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".
6. Define Module". Enter the entity used in design and then click "Next".
7. Develop VHDL code for given problemand Synthesize the .vhd file and view RTL schematic.
8. Create Test Bench file- A test bench is HDL code that allows you to provide a documented,

repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing.

9. Go to implementation to simulation tab , right click on main source file and create Test Bench file for simulation.
10. In order to view test files, select the box of "Simulation" in the "View Panel" of the "Design" panel. In the "Process Panel," double click on the "Behavioral Check Syntax" to make sure that you didn't make any syntax errors while making changes.
11. Double click on "Simulate Behavioral Model" in the "Process Pane", which will open the ISim software with your test bench loaded.
12. ISim simulator window will open with your simulation executed, as shown in Fig. where you are able to simulate your designs and check for errors.
13. After simulation implement it using FPGA development board.
14. Go to User Constraints – select Floorplan Area/IO/Logic (Plan Ahead) – Windows – Properties
    – now assign pin configuration and save.
15. Go to Implement Design – Run all.
16. Go to Generate Programming Fileand Run

XI Actual Procedure (use blank sheet provided if space not sufficient)

……...……

……...……

……...……

……...……

……...……

……...……

……...……

……...……

……...……

……...……

XII Resources

Table 11.2 Resources

Sr. No.	Name of Resource/Instrument /Components	Suggested Broad Specification	Quantity
1.			
2.			

XIII Observation Table

Table 11.3: Output of 2 Bit Comparator

Sr. no	A1	A0	B1	B0	A>B	A<B	A=B
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							

XIV Results (output of Program)

……

……

……

……

XV Interpretations of result

……

……

XVI Conclusions and Recommendation

……

……

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO
more such questions so as to ensure the achievement of identified CO

1. Design VHDL Code using FPGA board for 1 bit counter using dataflow modeling.
2. Write VHDL Code for 4 comparator using structural modeling.

[Space for Answer] (If required attach separate page)

……

XVIII Suggested referencesfor further reading.

Sr no.	Link/Portal	Description
1	http://vhsichdl.blogspot.com/2013/04/vhdl-code-for-comparator-library-ieee.html	Decoder tutorial
2	https://www.fpga4student.com/2017/08/vhdl-code-for-comparator.html	VHDL programming
3	https://www.tutorialspoint.com/vlsi_design/vhdl_programming_for_combinational circuit.html	ISE Quick start tutorial
4	http://ece2day.blogspot.com/2012/10/vhdl-tutorial-introducction-part-7.html	VHDL tutorial
5	https://electronicstopper.blogspot.com/2017/07/2 bit compotator - in-vhdl-with-testbench.html	VHDL programming

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related(15 Marks)		**60% (15)**
1	Coding ability	30%
2	Debugging ability	20%
3	Working with FPGA	10%
Product related (10 Marks)		**40%(10)**
4	Correct connections to FPGA Board	20%
5	Answer to sample questions.	15%
6	Timely Submission , Answer to sample questions.	05%
	TOTAL	**100% (25)**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.12: Design up-counter on FPGA Board

I Practical Significance

A counter is a device which stores the number of times a particular event or process has occurred, often in relationship to a clock signal. There are two types of counters: a) up counters b) down counters.

This practical will help the students to develop programming skills i.e. up-counter application on FPGA board using Xilinx ISE tools.

II Industry/Employer Expected Outcome

The aim of this course is to attend following industry/employer expected outcome through various teaching learning experiences:

"Develop VLSI-based electronic circuit/component using HDL."

III Course Level Learning outcome

CO-4 Develop VHDL program for given application.

IV Laboratory Learning Outcome

Interpret the output of MOD-10 Up counter using VHDL code.

V Relevant Affective domain related Outcome(s)

- Follow safe practices.
- Maintain tools and equipment.
- Follow ethical practices.

VI Relevant Theoretical Background

Up counter:-

Each of the higher-order flip-flops are made ready to toggle (both J and K inputs "high") if the Q outputs of all previous flip-flops are "high." Otherwise, the J and K inputs for that flip-flop will both be "low," placing it into the "latch" mode where it will maintain its present output state at the next clock pulse. Since the first (LSB) flip-flop needs to toggle at every clock pulse, its J and K inputs are connected to Vcc or Vdd, where they will be "high" all the time.

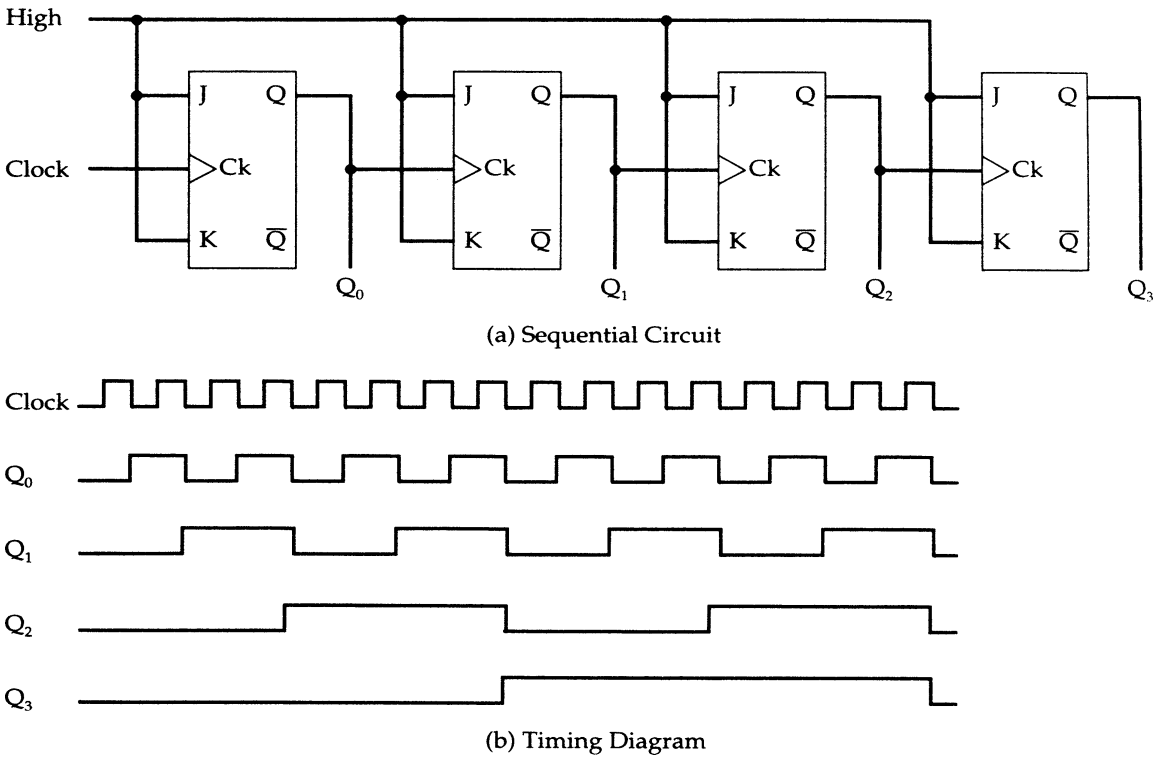


Fig. no. 12.1: Four bit up-counter and its timing diagram

VII Circuit diagram:

a) Practical JTAG cable setup:-

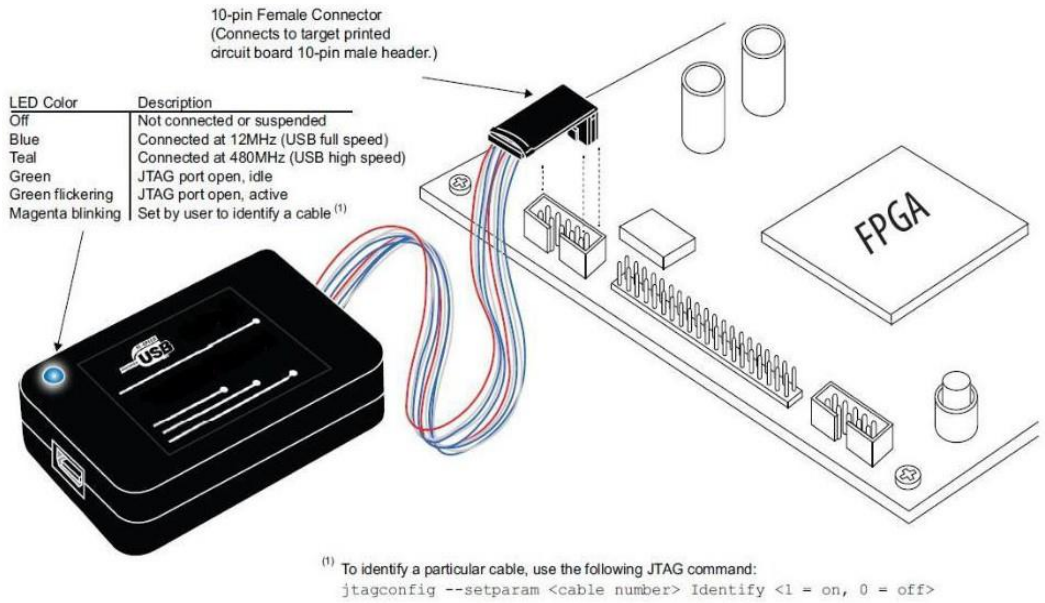


Fig. no.. 12.2.a: Practical Setup with JTAG CABLE

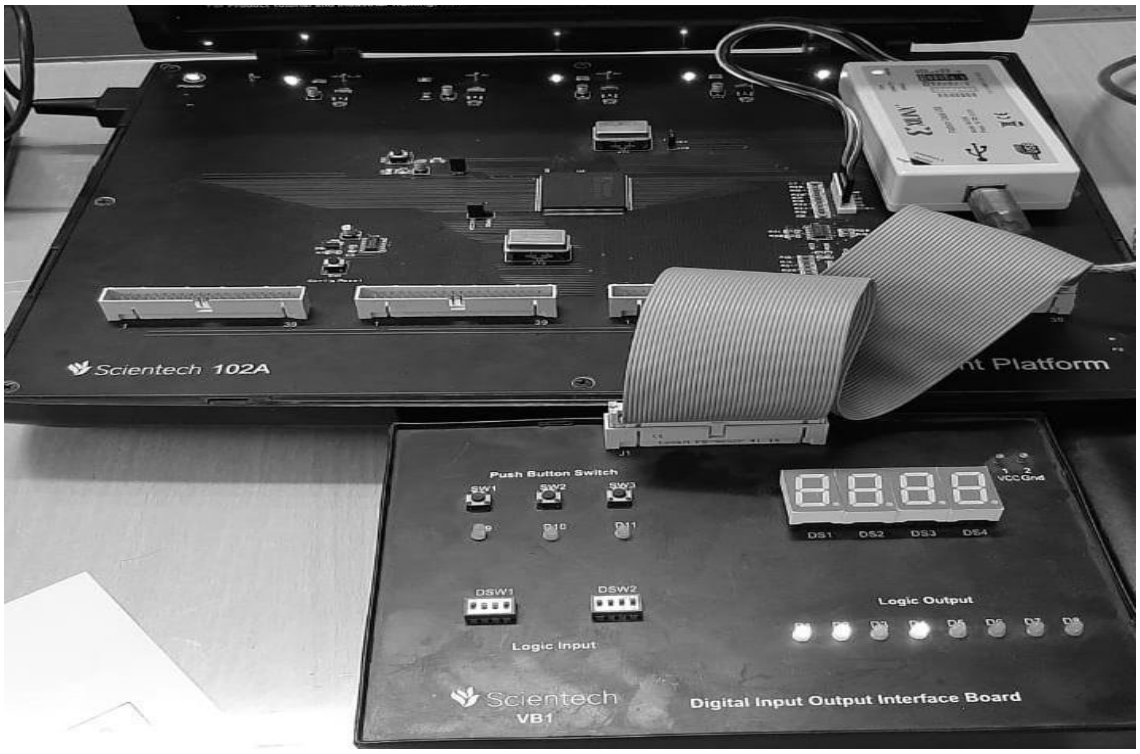


Fig. no. 12.2.b: Lab Practical Setup

b) Create binary to Up-counter .vhd file diagram :-

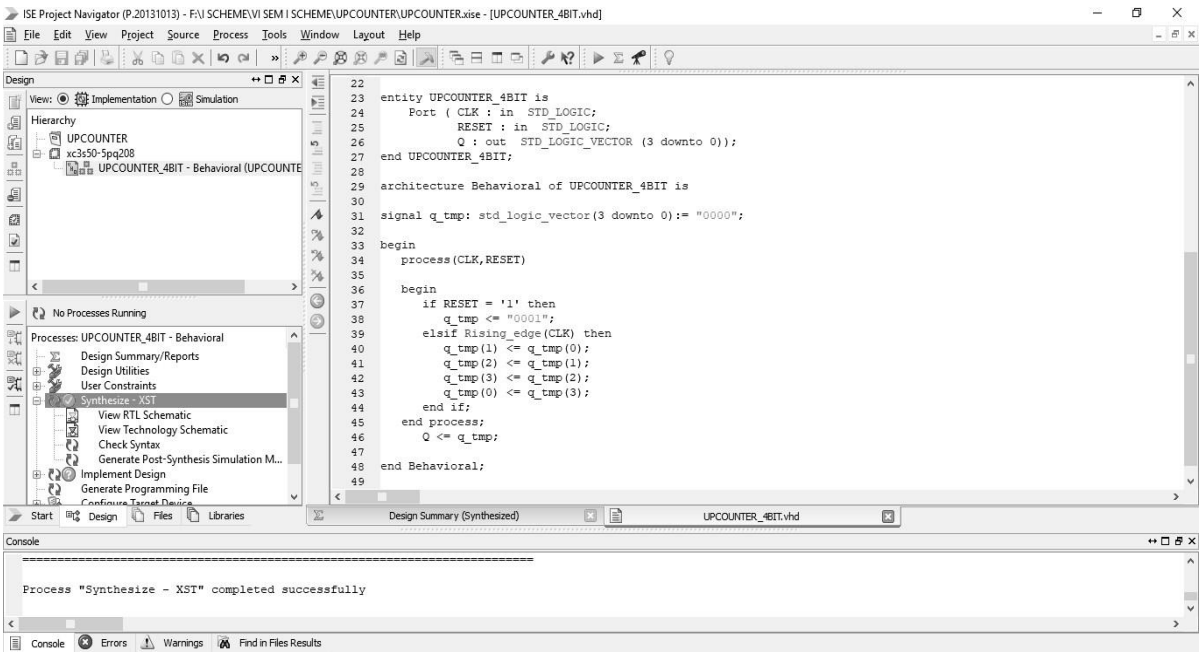


Fig. no.. 12.3: .vhd file diagram

c) Create binary to Up-counter test bench file-

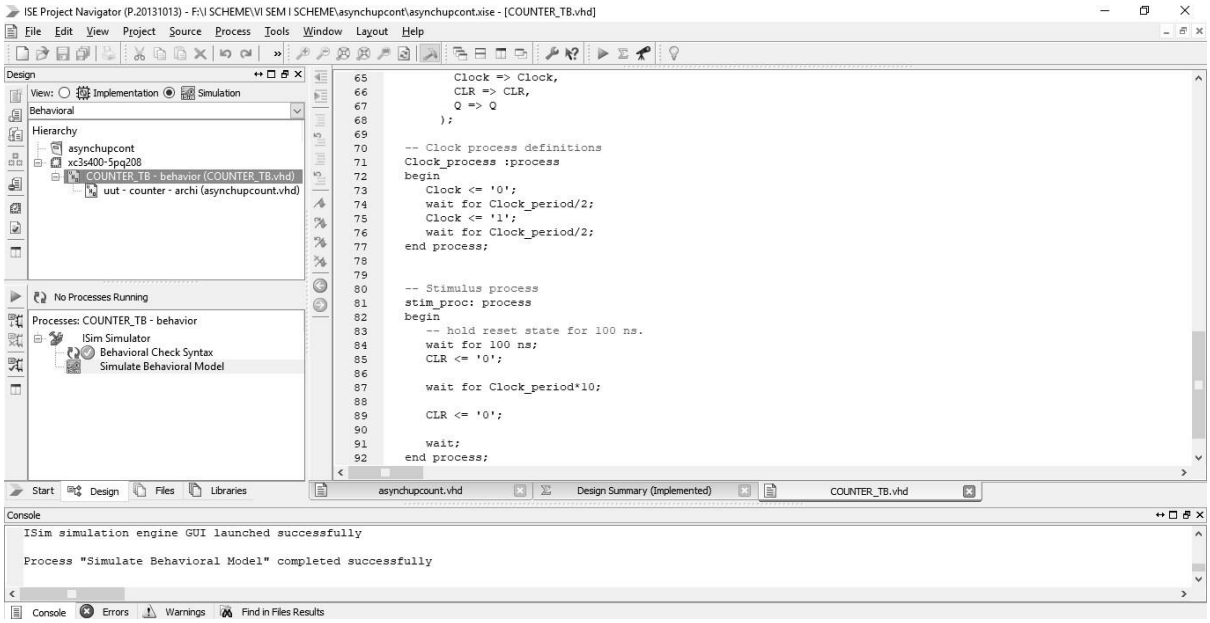


Fig. no. 12.4: test bench file diagram

d) binary to Up-counter test bench Simulation waveforms-

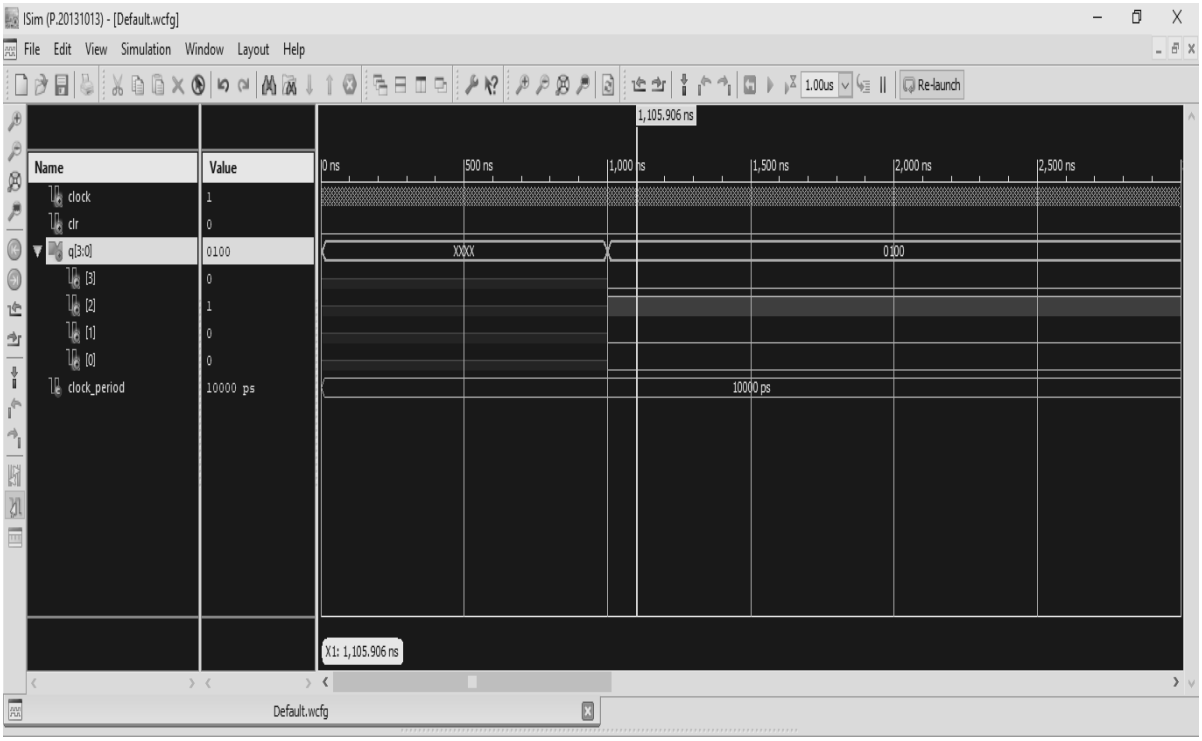


Fig. no.. 12.5: Simulation diagram

e) Actual practical set up used in laboratory

VIII Required Resources/apparatus/equipment with specifications

Table 12.1 Resources

Sr. No.	Instrument / Components	Specification	Quantity
1	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208),On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit.,On board, 2 Crystal 8MHz and 25MHz.JTAG Interface (Boundary Scan),PROM Interface (XCF02S),40 pin, 4 header connector for external I/O's	1 No.
2	Desktop PC	Personal computer with latest configuration. Loaded with open-source IDE, simulation and program downloading software, UPS and Antivirus.	1 No.

IX Precautions to be followed

1. Use proper Mains cord.
2. To avoid fire or shock hazards, observe all ratings and marks on the instrument.
3. Check syntax / rules for VHDL programming.

X Procedure

1. Create the Xilinx ISE project for expected FPGA design, by doing the following in ISE: In the ISE software, select File > New Project.
   In new project wizard, enter the project name and location, respectively. Select HDL or Schematic as the Top-level source type, and click Next.
2. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".
3. Define Module". Enter the entity used in design and then click "Next".
4. Develop VHDL code for given problem and Synthesize the .vhd file and view RTL schematic. Ref Fig no.11.3.
5. Create Test Bench file- A test bench is HDL code that allows to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file

input and output, and conditional testing.

6. Go to implementation to simulation tab , right click on main source file and create Test Bench file for simulation.
7. In order to view test files, select the box of "Simulation" in the "View Panel" of the "Design" panel. In the "Process Panel," double click on the "Behavioral Check Syntax" to make sure that didn't make any syntax errors while making changes.
8. Double click on "Simulate Behavioral Model" in the "Process Pane", which will open the ISim software with r test bench loaded.
9. ISim simulator window will open with simulation executed, as shown in Ref Fig where able to simulate designs and check for errors.
10. After simulation implement it using FPGA development board.
11. Go to User Constraints – select Floorplan Area/IO/Logic (PlanAhead) – Windows – Properties –now assign pin configuration and save.
12. Go to Implement Design – Run all.
13. Go to Generate Programming File and Run
14. Go to Configure Target Device and Run – Impact wizard opened – select Device 2 (as per practical Kit) -- open Bit file -- and initialize Chain – Right click on Xilinx IC And select Program.

SAMPLE PROGRAM 1: **Step 1.Develop VHDL code for Up-counter**

VHDL Code using When statement

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counter is
        port(Clock, CLR : in  std_logic;
        Q : out std_logic_vector(3 downto 0));
        end counter;

architecture archi of counter is
signal tmp: std_logic_vector(3 downto 0);

begin
        process (Clock, CLR)
        begin
        if (CLR='1') then
              tmp <= "0000";
                  elsif (Clock'event and Clock='1') then
                        tmp <= tmp + 1;
                  end if;
             end process;

            Q <= tmp;
  end archi;
```

Step 2: Develop Test Bench file for simulation.

VHDL Code

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY COUNTER_TB IS
END COUNTER_TB;

ARCHITECTURE behavior OF COUNTER_TB IS
   COMPONENT counter
   PORT(
        Clock : IN  std_logic;
        CLR : IN  std_logic;
        Q : OUT  std_logic_vector(3 downto 0)
       );
   END COMPONENT;
     signal Clock : std_logic := '0';
  signal CLR : std_logic := '0';

          --Outputs
  signal Q : std_logic_vector(3 downto 0);

  -- Clock period definitions
  constant Clock_period : time := 10 ns;

BEGIN

          -- Instantiate the Unit Under Test (UUT)
  uut: counter PORT MAP (
        Clock => Clock,
        CLR => CLR,
        Q => Q
      );

  -- Clock process definitions
  Clock_process :process
  begin
                    Clock <= '0';
                    wait for Clock_period/2;
                    Clock <= '1';
                    wait for Clock_period/2;
  end process;
   -- Stimulus process
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
```

```
                    CLR <= '0';
    wait for Clock_period*10;

                    CLR <= '0';
    wait;
  end process;
END;
```

Problem statement for student: Design VHDL Code for up counter using case statement.

Step 1.Develop VHDL code for Up-counter

VHDL Code

Step 2: Develop Test Bench file for simulation.

VHDL Code

XI Resources

Sr. No.	Instrument /Components	Specification	Quantity
1			
2			
3			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

……...

……...

……...

……...

……...

……...

……...

……...

……...

……..……..

……..……..

……..……..

……..……..

……..……..

……...

……...

XIII Observations (use blank sheet provided if space not sufficient)

Truth table of Binary up counter is _____(verified/ not verified) using FPGA development board.

XIV Result (Output of the Program)

……..……

……..……

XV Interpretation of Results (Give meaning of the above obtained results)

……..……

……..……

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the

interpretation of results)

……..……

……..……

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. Design VHDL Code using FPGA board for Binary Down-counte**r.**
2. Write VHDL code for 4 bit up-down counter using if else statement.

[Space for Answers] (If required attach separate page)

……..

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

……..……

XVIII Suggested referencesfor further reading

Sr no.	Link/Portal	Description
1	https://www.pantechsolutions.net/vhdl-code-to-simulate-4-bit-binary-counter-by-software-using-spartan-3-starter-kit	4 bit binary counter tutorial
2	https://www.xilinx.com/support/documentation/data_sheets/ds610.pdf	VHDL programming
3	https://forums.xilinx.com/	ISE Quick start tutorial
4	https://www.geeksforgeeks.org/differences-between-synchronous-and-asynchronous-counter/	VHDL tutorial

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related(15 Marks)		**60% (15)**
1.	Coding and Debugging ability	30%
2.	Making connections of hardware	20%
3.	Follow ethical practices.	10%
Product related (10 Marks)		**40%(10)**
4.	Correctness of algorithm/ Flow chart	20%
5.	Relevance of output of the problem definition.	15%
6.	Timely Submission , Answer to sample questions.	05%
	TOTAL	**100% (25)**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No. 13: Design synchronous counter on FPGA board.

I. Practical significance

In Synchronous Counter all the individual output bits changing state at exactly the same time in response to the common clock signal with no ripple effect and therefore, no propagation delay in the system. Synchronous Counter, the external clock signal is connected to the clock input of every individual flip-flop within the counter so that all of the flip-flops are clocked together simultaneously (in parallel) at the same time. In other words, changes in the output occur in "synchronization" with the clock signal. This practical will help the students to develop programming skills that is up/down synchronous counter application on FPGA board using Xilinx ISE tools.

II. Industry/Employer Expected outcome

The aim of this course is to attend following industry/employer expected outcome through various teaching learning experiences:

"Develop VLSI-based electronic circuit/component using HDL".

III. Course Level Learning outcome(s)

CO-4 Develop VHDL program for given application.

IV. Laboratory Learning Outcome(s)

Develop VHDL code for 4-bitUp/Down Synchronous counter and test the circuit on FPGA board

V. Relevant Affective domain related Outcome(s)

- Follow safe practices.
- Maintain tools and equipment.
- Follow ethical practices.

VI. Relevant Theoretical Background :

In Asynchronous binary counter the output of one counter stage is connected directly to the clock input of the next counter stage and so on. The result of this is that the Asynchronous counter suffers from what is known as "Propagation Delay" in which the timing signal is delayed a fraction through each flipflop. The solution over this problem is synchronous counter. In synchronous counters, the clock inputs of all the flip-flops are connected together

and are triggered by the input pulses. There are two types of counter :

1. Up counter
2. Down counter.

As well as counting "up" from zero and increasing or incrementing to some preset value, it is sometimes necessary to count "down" from a predetermined value to zero allowing us to produce an output that activates when the zero count or some other preset value is reached.

Table 13.1: Truth table of Up / down Synchronous counter.

CLK ' event	UP_DOWN	Q3	Q2	Q1	Q0	UP_DOWN	Q3	Q2	Q1	Q0
1	1	0	0	0	0	0	1	1	1	1
1	1	0	0	0	1	0	1	1	1	0
1	1	0	0	1	0	0	1	1	0	1
1	1	0	0	1	1	0	1	1	0	0
1	1	0	1	0	0	0	1	0	1	1
1	1	0	1	0	1	0	1	0	1	0
1	1	0	1	1	0	0	1	0	0	1
1	1	0	1	1	1	0	1	0	0	0
1	1	1	0	0	0	0	0	1	1	1
1	1	1	0	0	1	0	0	1	1	0
1	1	1	0	1	0	0	0	1	0	1
1	1	1	0	1	1	0	0	1	0	0
1	1	1	1	0	0	0	0	0	1	1
1	1	1	1	0	1	0	0	0	1	0
1	1	1	1	1	0	0	0	0	0	1
1	1	1	1	1	1	0	0	0	0	0

Note: UP_DOWN = 1 UP counter
UP_DOWN = 0 DOWN counter

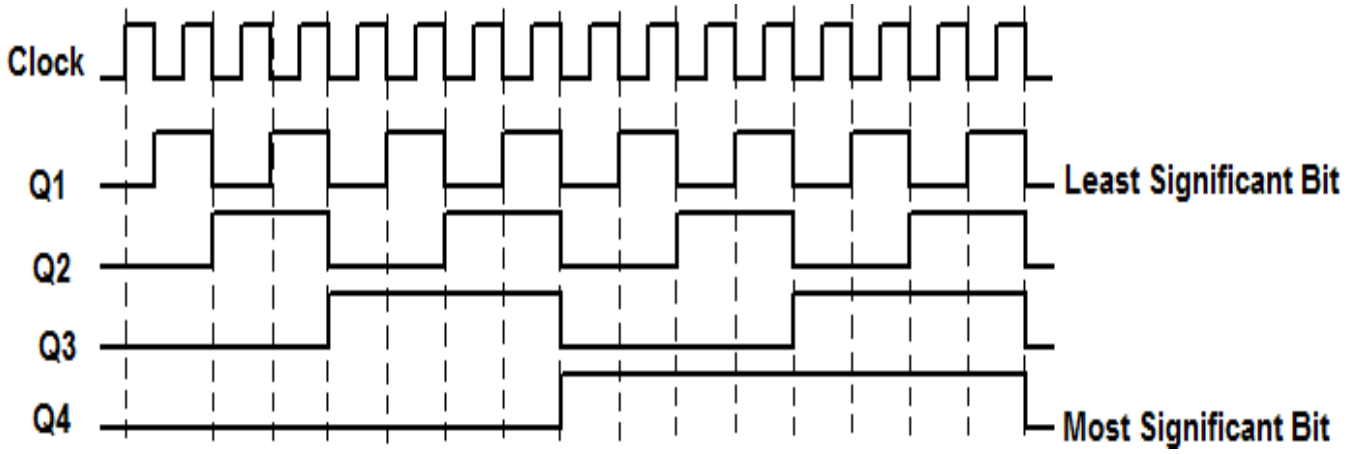


Fig. no.. 13.1: Four-bit Up counter timing waveform.

VII. Suggested circuit diagram used in Laboratory with related equipment rating.

a) Suggested Practical JTAG cable setup: -

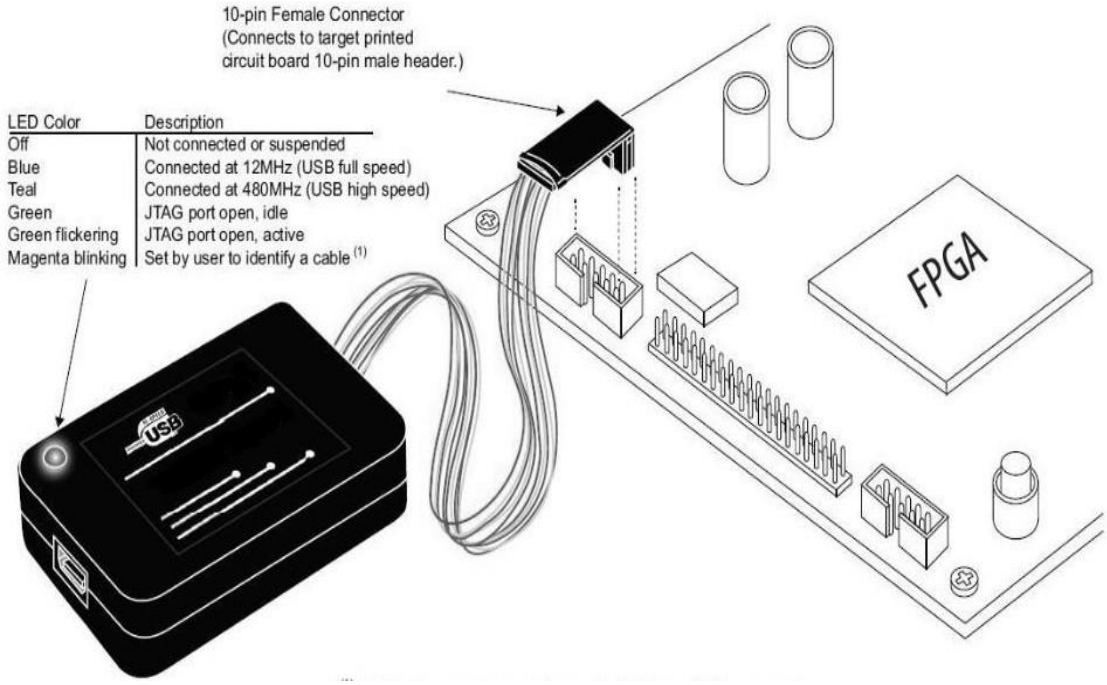


Fig. 13.2.a: Practical Setup with JTAG CABLE

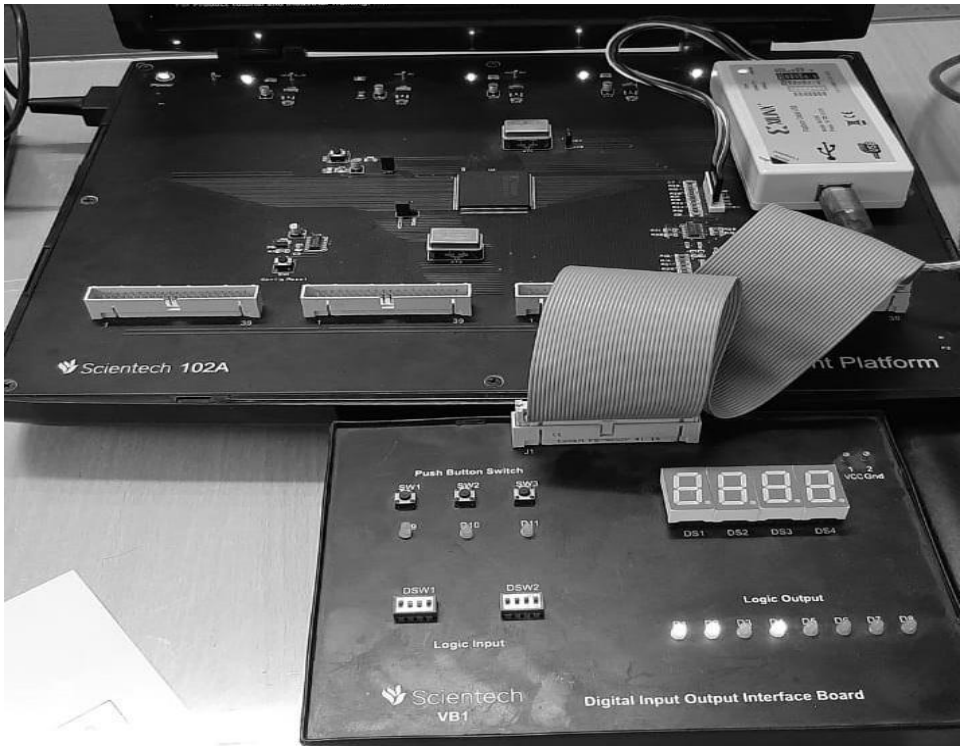


Fig. no.. 13.2.b: Lab Practical Setup

b) Create 4 bit Up/down counter .vhd file :-

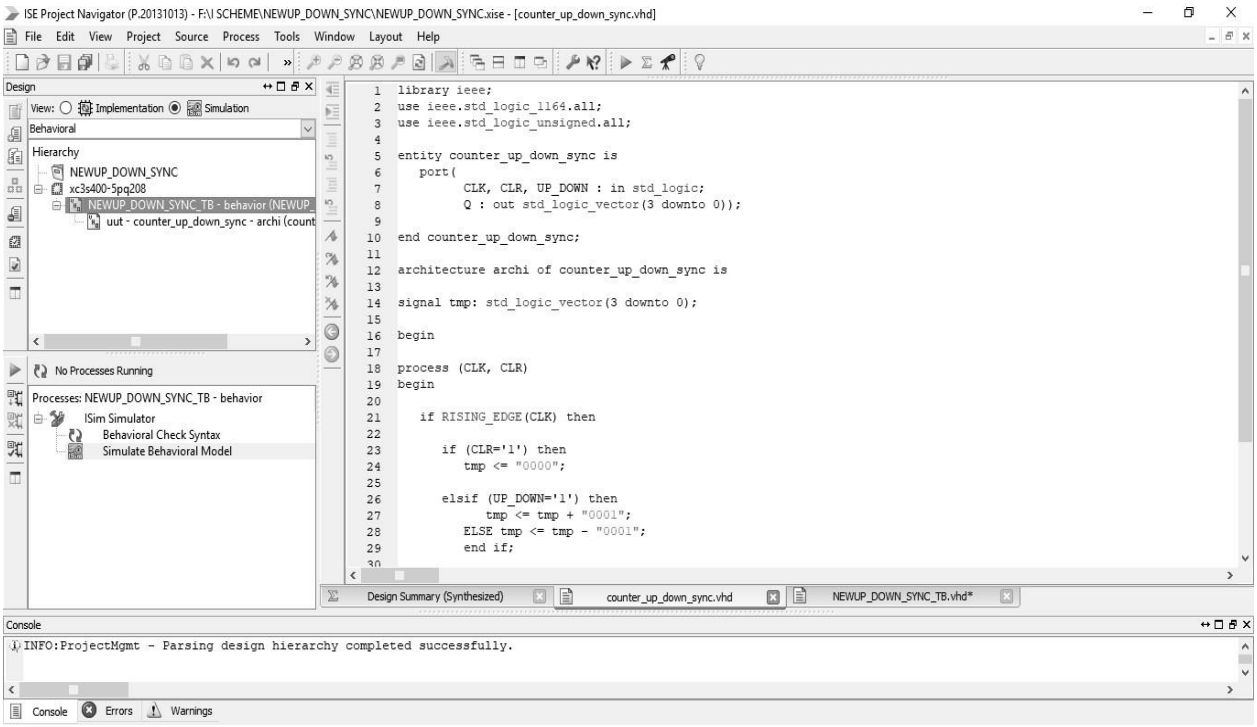


Fig. no.. 13.3: .vhd file

c) Create 4 bit Up/down counter test bench file-

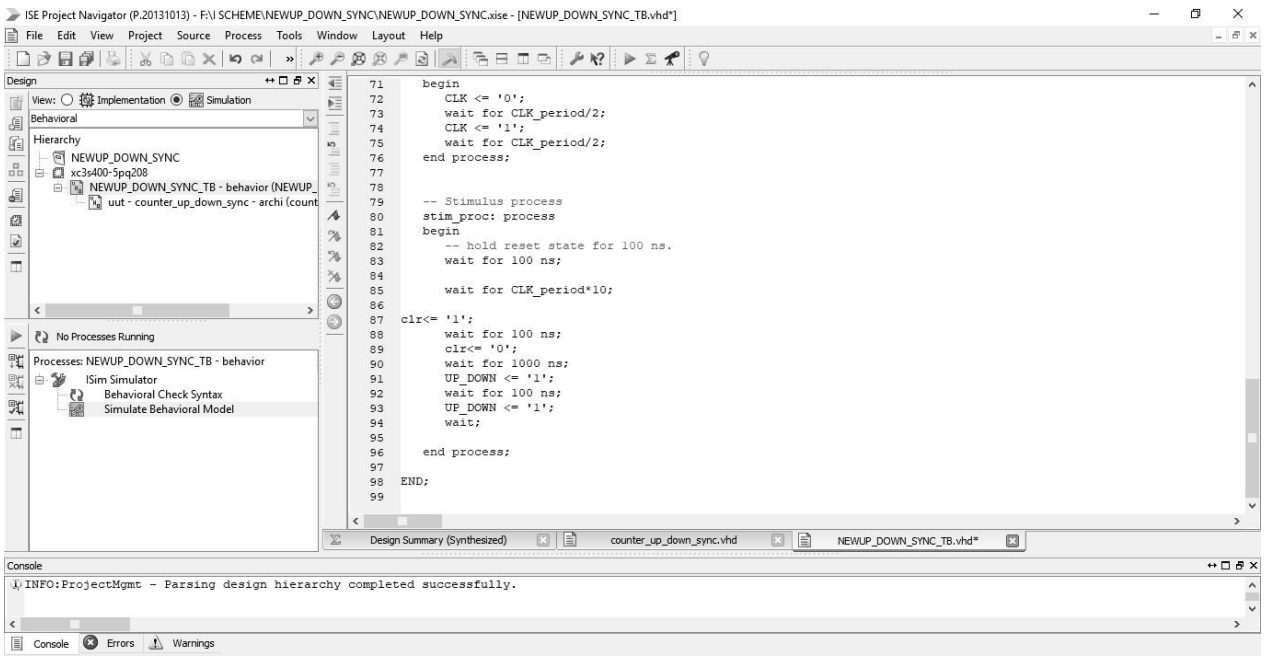


Fig. no.. 13.4: Test bench file

d) 4 bit Up/down synchronous counter test bench Simulation waveforms-

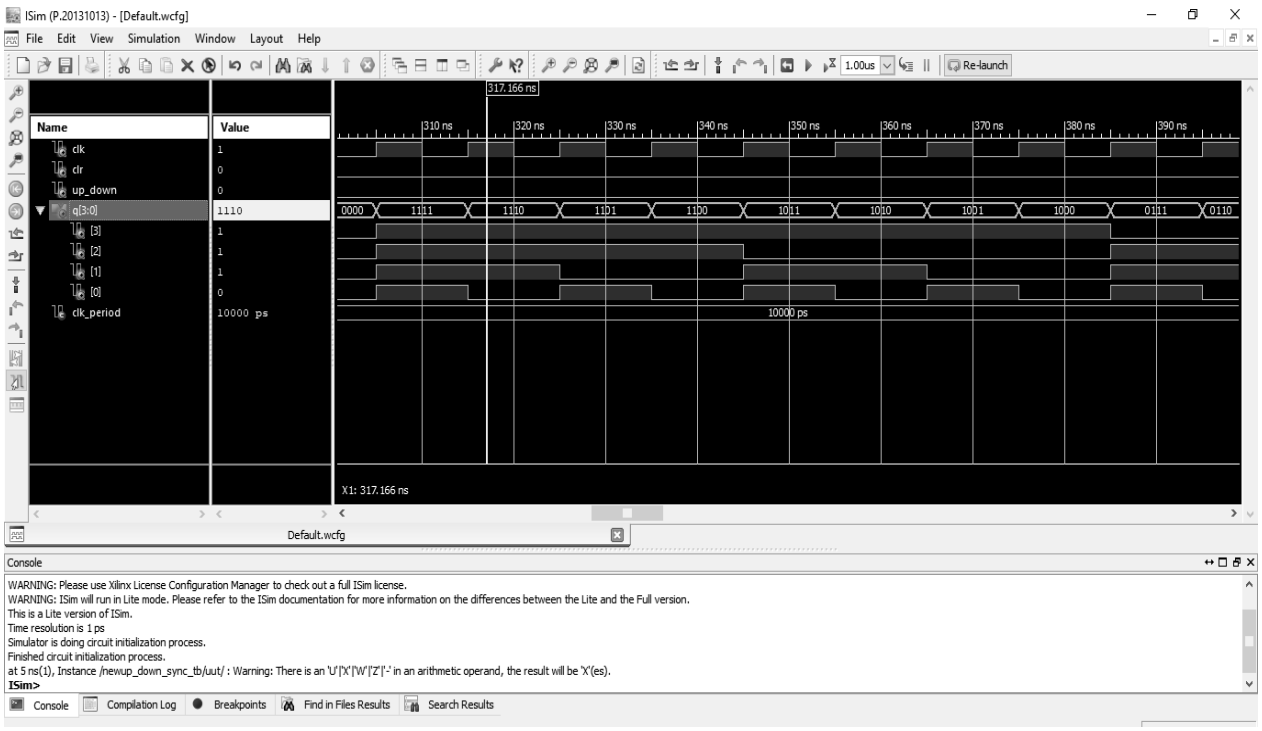


Fig. no..13.5: Simulation output

e) Actual practical set up used in laboratory

VIII. Required Resources/apparatus/equipment with specifications

Table 13.2 Required Resources

Sr. No	Instrument /Components	Specification	Quantity
1.	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208),On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit., On board, 2 Crystal 8MHz and 25MHz.JTAG Interface (Boundary Scan),PROM Interface (XCF02S),40 pin, 4 header connector for external I/O's	1 No.
2	Desktop PC	Personal computer with latest configuration. Loaded with open-source IDE, simulation and program downloading software, UPS and Antivirus..	2 No.

IX. Precautions to be followed

1. Use proper Mains cord.
2. To avoid fire or shock hazards, observe all ratings and marks on the instrument.
3. Check syntax / rules for VHDL programming.

X. Procedure

1. Create the Xilinx ISE project for top-level FPGA design, by doing the following in ISE:In the ISE software, select File > New Project.
2. In the Project name and Project location fields, enter the project name and location, respectively. Select HDL or Schematic as the Top-level source type, and click Next.

3. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".
4. Define Module". Enter the entity used in design and then click "Next".
5. Develop VHDL code for given problem and Synthesize the .vhd file and view RTL schematic. Ref Fig no.13.3.
6. Create Test Bench file- A test bench is HDL code that allows to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing. Ref Fig no.13.4.
7. Go to implementation to simulation tab , right click on main source file and create Test Bench file for simulation.
8. In order to view test files, select the box of "Simulation" in the "View Panel" of the "Design" panel. In the "Process Panel," double click on the "Behavioural Check Syntax" to make sure that didn't make any syntax errors while making changes.
9. Double click on "Simulate Behavioural Model" in the "Process Pane", which will open the ISim software with r test bench loaded.
10. ISim simulator window will open with simulation executed, as shown in Ref Fig no.13.5. where able to simulate designs and check for errors.
11. After simulation implement counter using FPGA development board.
12. Go to User Constraints – select Floorplan Area/IO/Logic (Plan Ahead) – Windows – Properties– now assign pin configuration and save.
13. Go to Implement Design – Run all.
14. Go to Generate Programming File and Run
15. Go to Configure Target Device and Run – Impact wizard opened – select Device 2 (as per practical Kit) -- open Bit file -- and initialize Chain – Right click on Xilinx IC And select Program.

SAMPLE PROGRAM 1: **Step 1. Develop VHDL code for 4 bit Up/down synchronous counter**

VHDL Code using When statement

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;


ENTITY counter_up_down_sync is

port (
CLK, CLR, UP_DOWN: in std_logic;
Q : out std_logic_vector(3 downto 0));
end counter_up_down_sync;

architecture archi of counter_up_down_sync is

signal tmp: std_logic_vector(3 downto 0);

begin

process (CLK, CLR)
```

```
begin

if RISING_EDGE(CLK) then
      if (CLR='1') then
            tmp <= "0000";
     elsif (UP_DOWN='1') then
          tmp <= tmp + "0001";


  else tmp <= tmp - "0001";
end if;
 end if;

end process;
Q <= tmp;

end archi;
```

Step 2: Develop Test Bench file for simulation

VHDL Code

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

ENTITY NEWUP_DOWN_SYNC_TB IS

END NEWUP_DOWN_SYNC_TB;

ARCHITECTURE behavior OF NEWUP_DOWN_SYNC_TB IS

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

COMPONENT counter_up_down_sync
```

```
PORT(
 CLK : IN
std_logic; CLR :
IN std_logic;
UP_DOWN : IN std_logic;
Q : OUT std_logic_vector(3 downto 0)
);
END COMPONENT;
--Inputs
signal CLK : std_logic
:= '0'; signal CLR :
std_logic := '0';
signal UP_DOWN : std_logic := '0';
--Outputs
signal Q : std_logic_vector(3 downto 0);
-- Clock period definitions
constant CLK_period : time := 10 ns;

BEGIN
-- Instantiate the Unit Under Test
(UUT) uut: counter_up_down_sync
PORT MAP ( CLK => CLK,
CLR => CLR,
UP_DOWN =>
UP_DOWN, Q => Q
);
-- Clock process
definitions
CLK_process
:process begin
CLK <= '0';
wait for CLK_period/2;
```

```
CLK <= '1';
wait for CLK_period/2;
end process;
-- Stimulus process
stim_proc:  process
begin
-- hold reset state for 100 ns.
wait for 100 ns;
wait for CLK_period*10;
clr<= '1';
wait for 100 ns;
clr<= '0';
wait for 1000 ns;
UP_DOWN <= '1';
wait   for   100   ns;
UP_DOWN <= '1';
wait;
end process;
END;
```

Problem statement for student: Design VHDL Code for 3 bit up/down synchronous counter using if else statement logic gates etc.

Step 1.Develop VHDL code for 3 bit up/down synchronous counter

VHDL Code

Step 2: Develop Test Bench file for simulation.

VHDL Code

XI. Resources

Table 13.3 Resources

Sr. No.	Instrument /Components	Specification	Quantity
1.			
2.			
3.			

XII. Actual Procedure Followed (use blank sheet provided if space not sufficient)

……..

……..

……..

……..

……..

……..……...……...

……..……...

XIII. Observations (use blank sheet provided if space not sufficient)

Truth table of up/down synchronous counter is _____(verified/ not verified) using FPGA development board.

XIV. Result (Output of the Program)

……..……...……...

……..……...

XV. Interpretation of Results (Give meaning of the above obtained results)

……..……...……...

……..……...

XVI. Conclusion and Recommendation (Actions/decisions to be taken based on the Interpretation of results)

.....

.....

XVII. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. Design VHDL Code using FPGA board for 3 Bit Binary Down-counter**.**

[Space for Answers] (If required attach separate page)

.....

.....

.....

.....

.....

.....

.....

.....

.....

XVIII. Suggested referencesfor further reading

Sr no.	Link/Portal	Description
1	https://www.pantechsolutions.net/vhdl-code-to-simulate-4-bit-binary-counter-bysoftware- using- spartan-3-starter-kit	Synchronous counter tutorial
2	https://circuitdigest.com/tutorial/synchronous-counter.	VHDL programming
3	https://forums.xilinx.com/	ISE Quick start tutorial
4	https://www.geeksforgeeks.org/differences-between-synchronous-andasynchronous-	VHDL tutorial

XIX. Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related(15 Marks)		**60% (15)**
4.	Coding and Debugging ability	30%
5.	Making connections of hardware	20%
6.	Working with FPGA.	10%
Product related (10 Marks)		**40%(10)**
7.	Correctness of algorithm/ Flow chart	20%
8.	Relevance of output of the problem definition.	15%
9.	Timely Submission of report, Answer to sample questions.	05%
	TOTAL	**100% (25)**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No. 14: Design Binary to Gray code converter using FPGA board

I Practical Significance

Binary Number is normally used number system in digital electronics. In many applications coding is essential and for the purpose variation in binary number system is required. For such applications excess-3, gray code and other coding methods are used. The Gray code was originally designed to prevent undesired transient states or outputs from electro- mechanical switches. Today, this code is used to facilitate error correction in digital communications and instrumentation This practical will help students to develop programming skills i.e. Binary to Gray code converter application on FPGA board using Xilinx ISE tools.

II Industry Relevant/Employer Expected Outcome

The aim of this course is to attend following industry/employer expected outcome through various teaching learning experiences:

"Develop VLSI-based electronic circuit/component using HDL".

III Course Level Learning Outcome

CO-4 - Develop VHDL program for given application.

IV Laboratory Learning Outcome

LLO 14.1 Test the functionality of 4-bit binary to gray code converter and Synthesize using FPGA.

V Relevant Affective domain related Outcome(s)

- Follow safe practices.
- Maintain tools and equipment.
- Follow ethical practices.

VI Relevant Theoretical Background

Gray Code system is a binary number system in which every successive pair of numbers differs in only one bit. It is used in applications in which the normal sequence of binary numbers generated by the hardware may produce an error or ambiguity during the transition from one number to the next.

For example, the states of a system may change from 3(011) to 4(100) as- 011 — 001 — 101 — 100. Therefore there is a high chance of a wrong state being read while the system changes from the initial state to the final state.

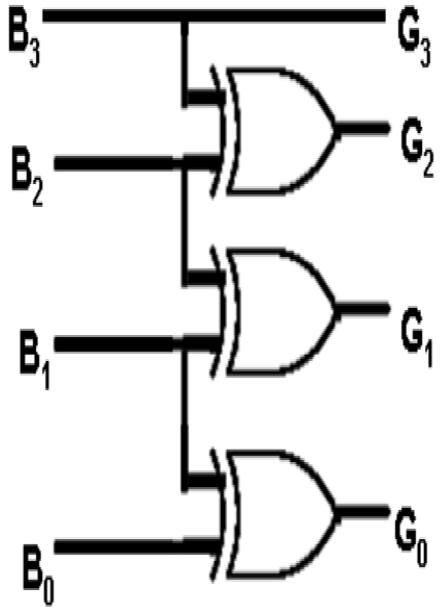
This could have serious consequences for the machine using the information. The Gray code eliminates this problem since only one bit changes its value during any transition between two numbers

Gray code equivalent of the given binary number is computed as follows:

g3 = b3
g2 = b3 XOR b2
g1 = b2 XOR b1
g0 = b1 XOR b0

A binary to gray code converter can be implemented using XOR gates. For n input, n-1 gates are required. As shown in the image below for 4 inputs, 3 XOR gates are used:

Table 14.1: Truth table of Binary to Gray code conversion.



Binary				Gray Code			
b_3	b_2	b_1	b_0	g_3	g_2	g_1	g_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Fig. no.. 14.1: Binary to Gray code conversion.

VII Circuit diagram used in laboratory with related equipment rating.

a) Suggested Practical JTAG cable setup: -

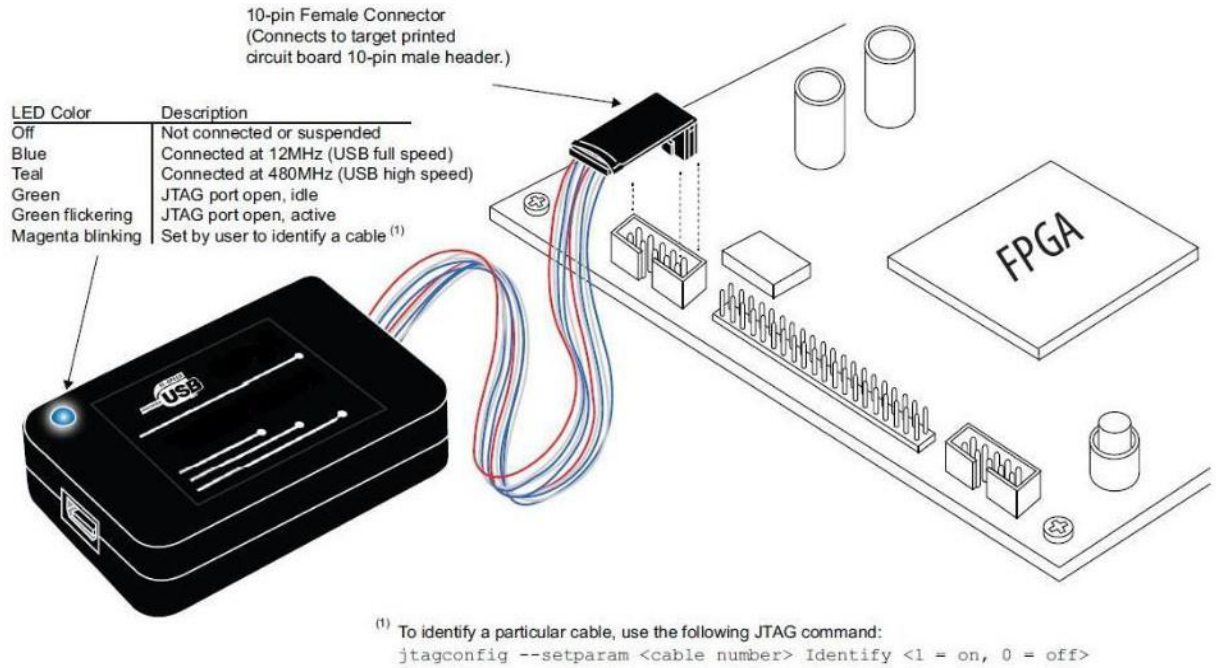


Fig. no.. 14.2.a: Practical Setup with JTAG CABLE

Fig. no. 14.2.b: Lab Practical Setup

b) Create binary to gray code converter .vhd file diagram :-

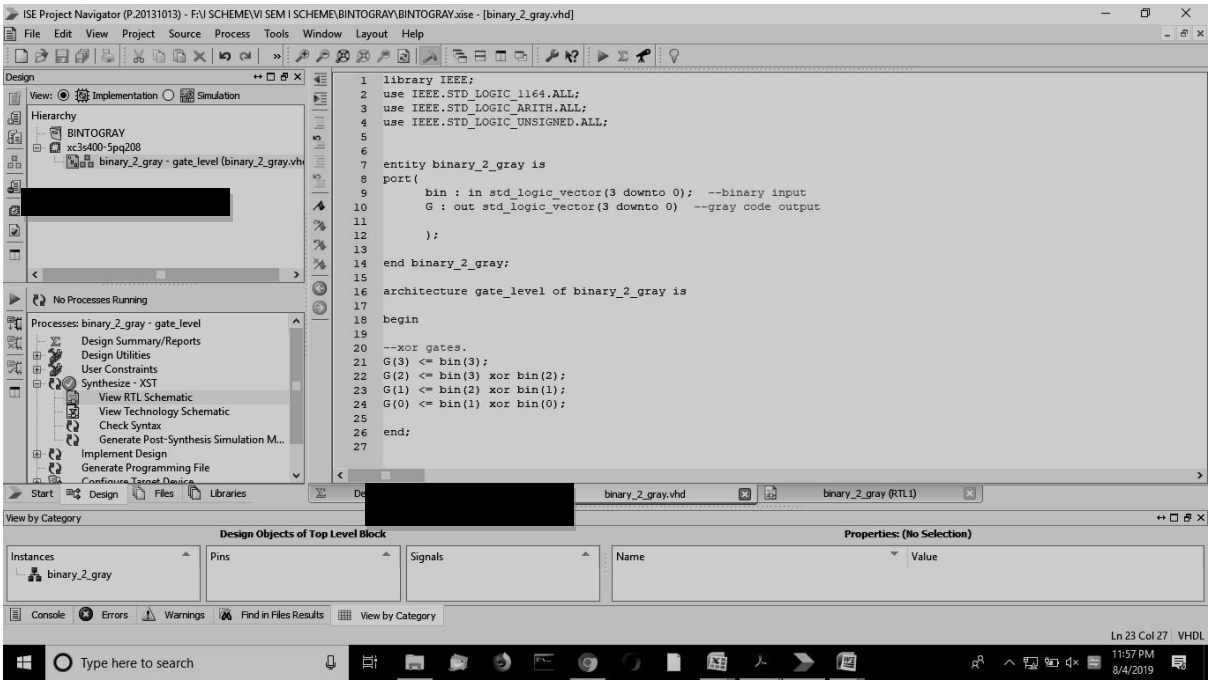


Fig. no. 14.3: .vhd file diagram

c) Create binary to gray code converter test bench file-

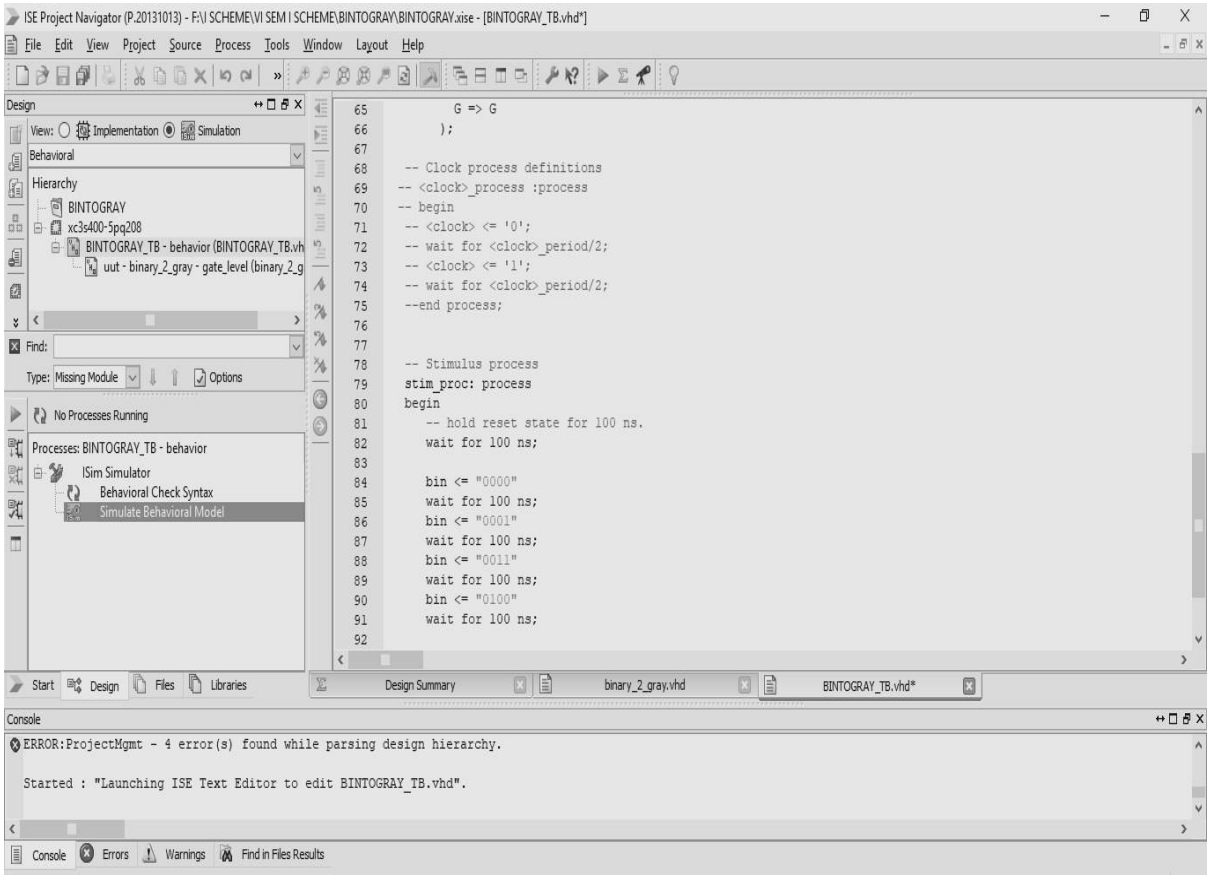


Fig. no. 14.4: test bench file diagram

d) binary to gray code converter test bench Simulation waveforms-

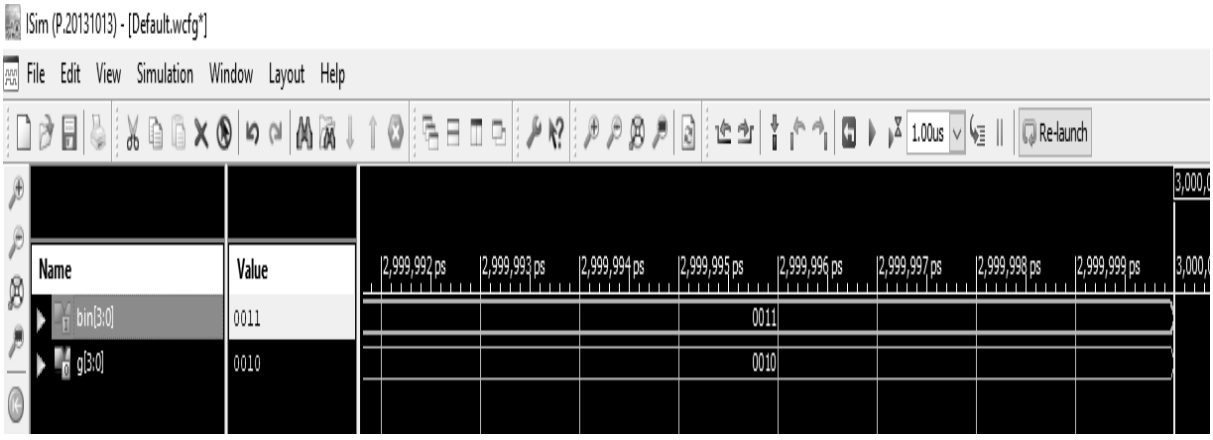


Fig. no. 14.5: Simulation diagram

e) Actual Experimental set up used in laboratory

VIII Resources Required

Table 14.1 Resources

Sr. No.	Instrument / Components	Specification	Quantity
1	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208),On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit.,On board, 2 Crystal 8MHz and 25MHz.JTAG Interface (Boundary Scan),PROM Interface (XCF02S),40 pin, 4 header connector for external I/O's	1 No.
2	Desktop PC	Loaded with open source IDE, simulation and program downloading software.	1 No.

IX Precautions to be followed

1. Use proper Mains cord.
2. To avoid fire or shock hazards, observe all ratings and marks on the instrument.
3. Check syntax / rules for VHDL programming.

X Procedure

1. Create the Xilinx ISE project for top-level FPGA design, by doing the following in ISE:
   a. In the ISE software, select File > New Project.
   b. In the Project name and Project location fields, enter the project name and location, respectively.
   c. Select HDL or Schematic as the Top-level source type, and click Next.
2. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".
3. Define Module". Enter the entity used in design and then click "Next".
4. Develop VHDL code for given problem and Synthesize the .vhd file and view RTL schematic. Ref Fig no.13.3.
5. Create Test Bench file- A test bench is HDL code that allows to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing. Ref Fig no.13.4.

6. Go to implementation to simulation tab , right click on main source file and create Test Bench file for simulation.
7. In order to view test files, select the box of “Simulation” in the “View Panel” of the “Design” panel. In the “Process Panel,” double click on the “Behavioral Check Syntax” to make sure that didn’t make any syntax errors while making changes.
8. Double click on “Simulate Behavioral Model” in the “Process Pane”, which will open the ISim software with test bench loaded.
9. ISim simulator window will open with simulation executed, as shown in Ref Fig no.13.5. where are able to simulate designs and check for errors.
10. After simulation implement it Using FPGA development board.
11. Go to User Constraints – select Floorplan Area/IO/Logic (PlanAhead) – Windows – Properties –
    now assign pin configuration and save.
12. Go to Implement Design – Run all.
13. Go to Generate Programming File and Run
14. Go to Configure Target Device and Run – Impact wizard opened – select Device 2 (as per practical Kit) -- open Bit file -- and initialize Chain – Right click on Xilinx IC And select Program.

SAMPLE PROGRAM :: **Step 1.Develop VHDL code for Binary to Gray conversion**

VHDL Code using When statement

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

 entity ENCODER8TO3 is
    Port ( a : in  STD_LOGIC_VECTOR (7 downto 0); b
         : out  STD_LOGIC_VECTOR (2 downto 0));
 end ENCODER8TO3;

 architecture Behavioral of ENCODER8TO3 is begin
 process(a)
 begin  case
 a is
  when "00000001" => b <= "000";
  when "00000010" => b <= "001";
  when "00000100" => b <= "010";
  when "00001000" => b <= "011";
  when "00010000" => b <= "100";
  when "00100000" => b <= "101";
  when "01000000" => b <= "110";
  when "10000000" => b <= "111";
  when others => b <= "ZZZ"; end
  case;
 end process;

 end Behavioral;
```

Step 2: Develop Test Bench file for simulation.

VHDL Code

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

ENTITY ENDCODER8TO3_TB IS
END ENDCODER8TO3_TB;

ARCHITECTURE behavior OF ENDCODER8TO3_TB IS

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

   COMPONENT ENCODER8TO3
   PORT(
       a : IN  std_logic_vector(7 downto 0);
       b : OUT  std_logic_vector(2 downto 0)
      );
   END COMPONENT;

  --Inputs
   signal a : std_logic_vector(7 downto 0) := (others => '0');

          --Outputs
   signal b : std_logic_vector(2 downto 0);
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name

   BEGIN
   uut: ENCODER8TO3 PORT MAP (
         a => a,
         b => b
       );

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

          a <= "00000001";
          wait for 100 ns;
          a <= "00000010";
          wait for 100 ns;
          a <= "00000100";
wait for 100 ns;
          a <= "00001000";
          wait for 100 ns;
```

```
        a <= "00010000";
        wait for 100 ns;
        a <= "00100000";
        wait for 100 ns;
        a <= "01000000";
        wait for 100 ns;
        a <= "10000000";
        wait;
        end process;
END;
```

Problem statement for student: Design VHDL Code for Gray to Binary encoder using if else statement logic gates etc.

Step 1: Develop VHDL code for Binary to Gray conversion.

VHDL Code using Dataflow method statement

Step 2: Develop Test Bench file for simulation.

VHDL Code

XI Resources

Sr. No.	Instrument /Components	Specification	Quantity
1			
2			
3			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

……

……

……

……

……..

……..

……..

……..

……..

……..

……..

XIII Observations (use blank sheet provided if space not sufficient)

Truth table of Binary to Gray is _____(verified/ not verified) using FPGA development board.

XIV Result (Output of the Program)

……

……

XV Interpretation of Results (Give meaning of the above obtained results)

……

……

XVI Conclusions and Recommendation

……

……

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. Write VHDL Code using FPGA board for Gray to Binary using When statement.
2. Write VHDL Code using FPGA board for Gray to Binary using Dataflow method statement.

[Space for Answers] (If required attach separate page)

……..……

……..……

……..……

……..……

……..……

……..……

XVIII Suggested references for further reading

Sr no.	Link/Portal	Description
1	https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_usb_blstr_ii_cable.pdf	VHDL programming
2	https://www.xilinx.com/support/documentation/data_sheets/ds610.pdf	EDA Tool
3	https://forums.xilinx.com/	ISE Quick start tutorial
4	https://www.geeksforgeeks.org/digital-logic-code-converters-binary-gray-code/	VHDL tutorial

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related(15 Marks)		**60% (15)**
1	Coding and Debugging ability	30%
2	Making connections of hardware	20%
3	Follow ethical practices.	10%
Product related (10 Marks)		**40%(10)**
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition.	15%
6	Timely Submission , Answer to sample questions.	05%
TOTAL		**100% (25)**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No. 15: Design Digital to Analog code converter (DAC)using FPGA board

I Practical Significance

In the real world, most of the data are available in analog form. There are two types of electronic converters namely Analog to Digital converter(ADC) and Digital to Analog converter (DAC). While manipulating the data, these two converting interfaces are essential. This practical will help the students to develop programming skills to Build / Test DAC application on FPGA board using Xilinx ISE tools.

II Industry/Employer Expected Outcome

The aim of this course is to attend following industry/employer expected outcome through various teaching learning experiences:

"Develop VLSI-based electronic circuit/component using HDL".

III Course Level Learning outcome

CO-4 Develop VHDL program for given application.

.

IV Laboratory Learning Outcome

LLO 15.1 Develop VHDL code for 8 -bit Digital to Analog converter and test the circuit using FPGA board.

V Relevant Affective domain related Outcomes

- Follow safe practices.
- Maintain tools and equipment.
- Follow ethical practices.

VI Relevant Theoretical Background

Digital to Analog Converter (DAC) :- It is a device that transforms digital data into an analog signal. A DAC can reconstruct sampled data into an analog signal with precision. The digital data may be produced from a microprocessor, Application Specific Integrated Circuit (ASIC) or Field Programmable Gate Array (FPGA), but ultimately the data requires the conversion to an analog signal in order to interact with the real world.

A D/A converter takes a precise number (most commonly a fixed-point binary number) and converts it into a physical quantity (example: voltage, pressure, temperature). D/A converters are often used to convert finite-precision time series data to a continually varying physical signal.

Digital Data:-
Evenly spaced discontinuous values, temporally discrete, quantitatively discrete. Analog Data (Natural Phenomena):-
Continuous range of values, temporally continuous, quantitatively continuous.

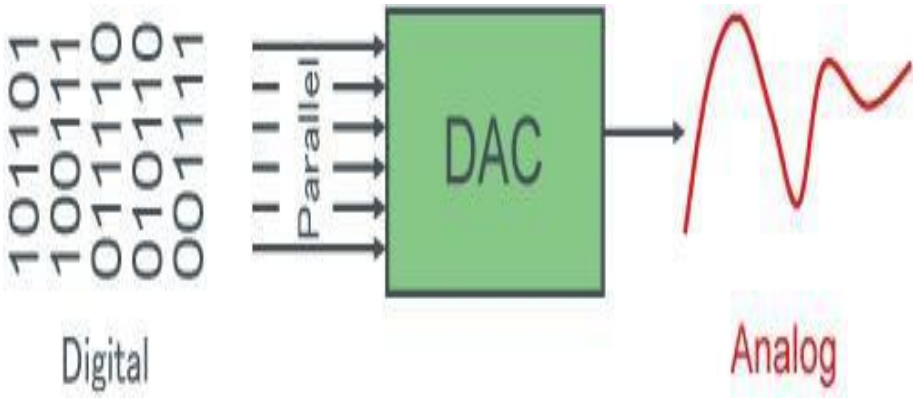


Fig. no. 15.1.a: Basic Digital to analog signal representation.

Fig. no. 15.1.b: Basic 8 Bit Digital to analog converter.

VII Suggested Circuit diagram used in Laboratory with related equipment rating:

a) Suggested Circuit diagram

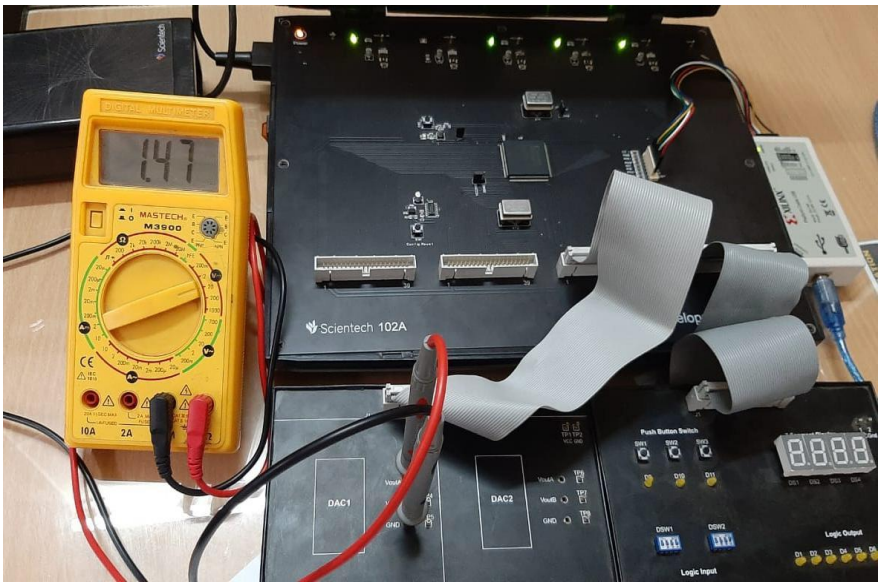


Fig. no. 15.2.a: Lab Practical Setup by set LSB data Bit high Analog output is low voltage

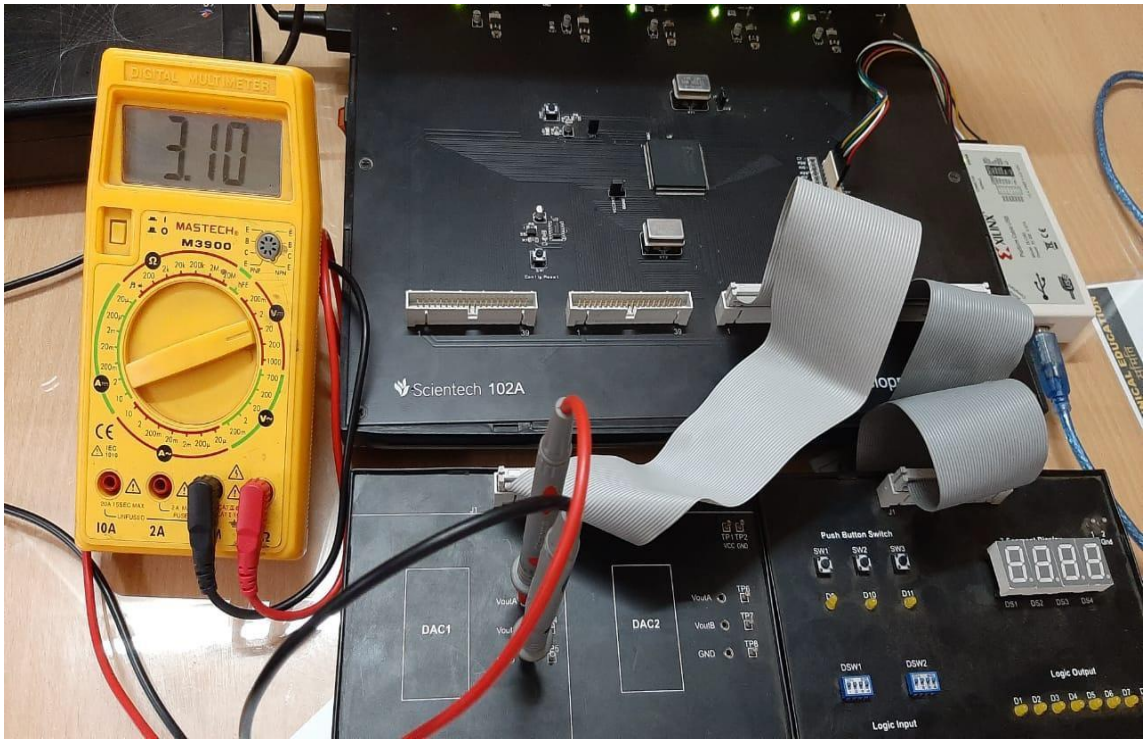


Fig. no. 15.2.b: Lab Practical Setup by set MSB data Bit high Analog output is high voltage

b) Create DAC .vhd file diagram :--

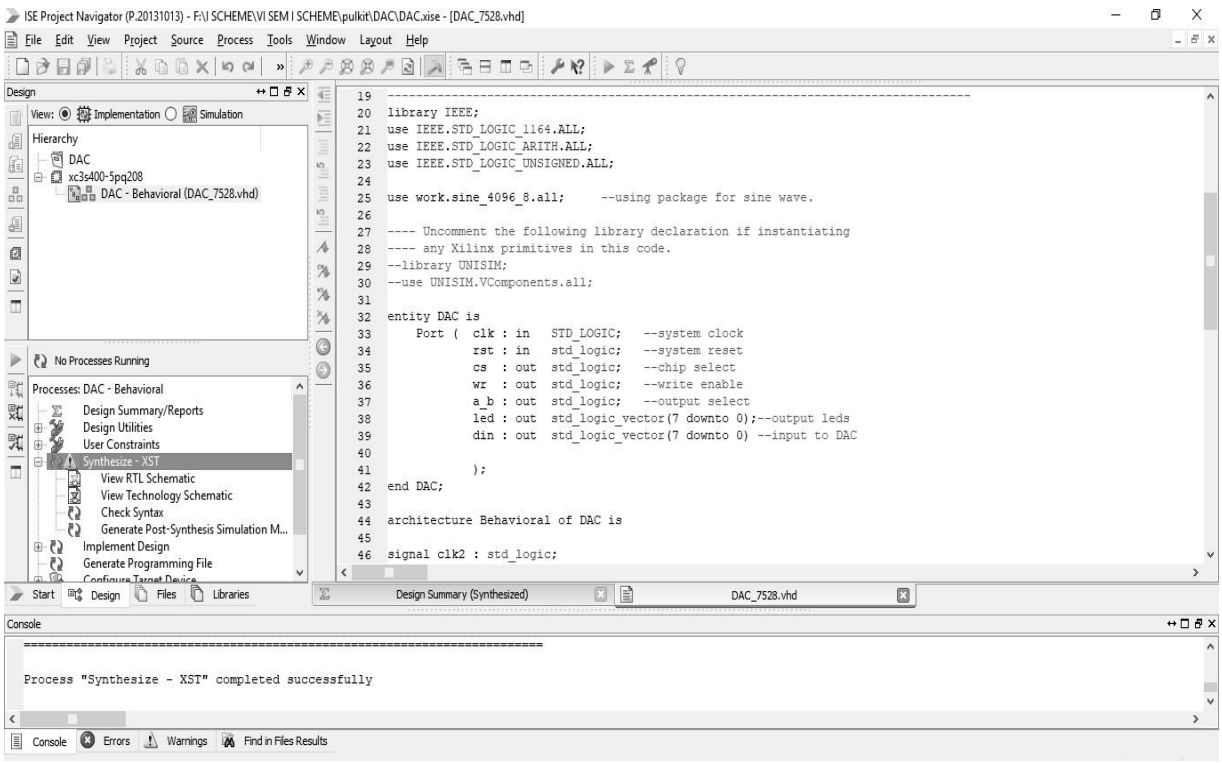


Fig. no. 15.3: .vhd file diagram

c) DAC Simulation waveforms by set input bit forcefully constant :-

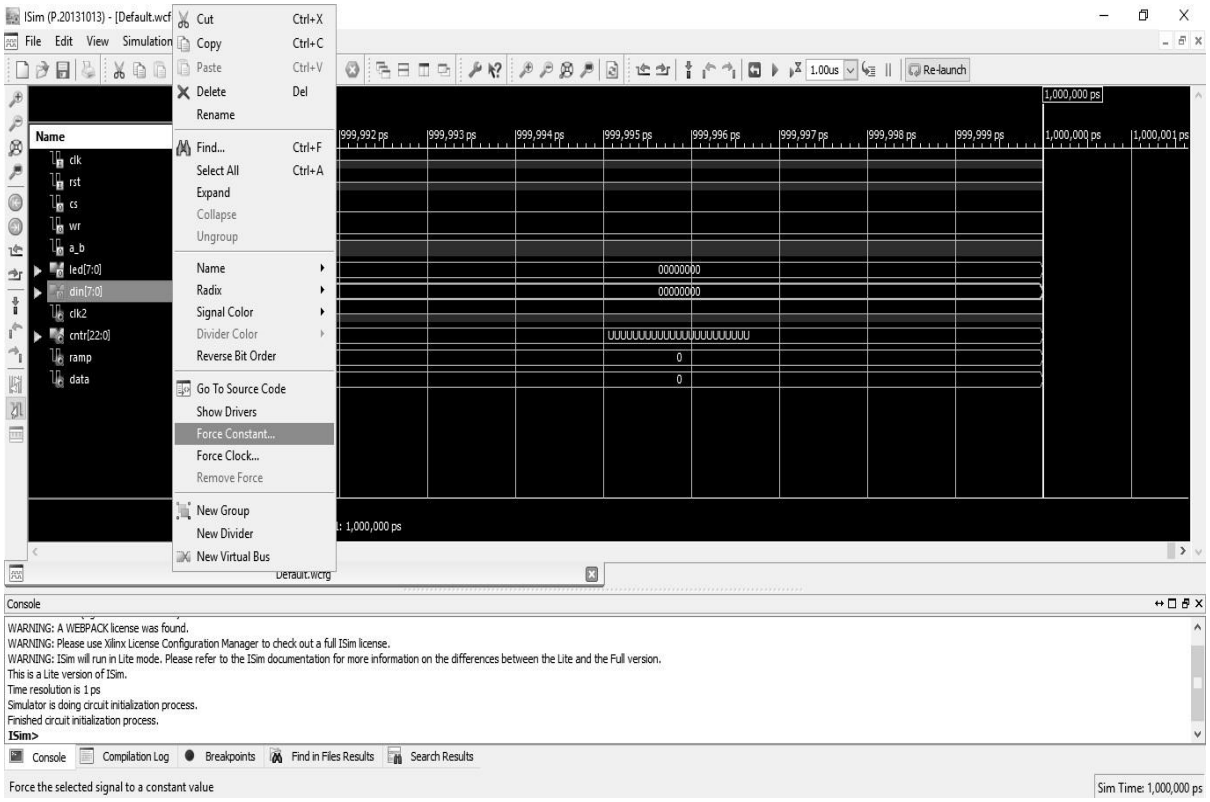


Fig. no. 15.4.a: Simulation diagram

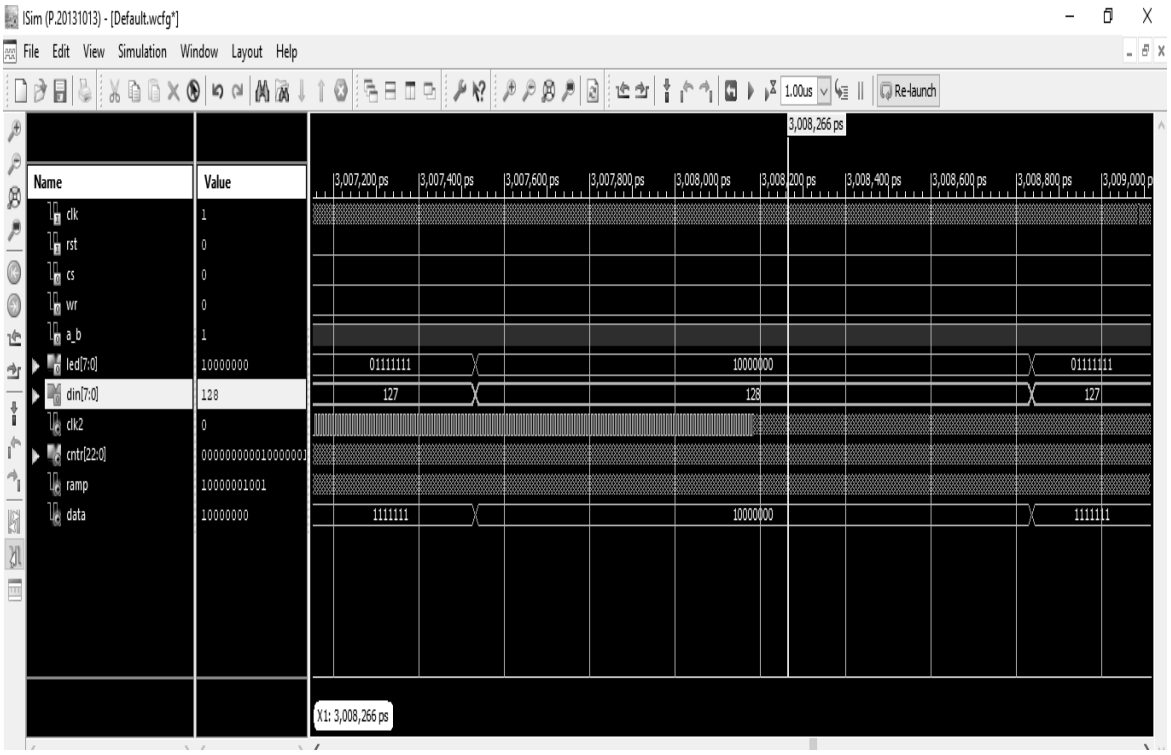


Fig. no. 15.4.b: Simulation diagram (arrow shows peak level of converted analog signal)

c. Actual practical setup diagram

VIII Resources Required

Table 15.1 Resources

Sr. No.	Instrument / Components	Specification	Quantity
1	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208), On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit., On board, 2 Crystal 8MHz and 25MHz.JTAG Interface (Boundary Scan), PROM Interface (XCF02S),40 pin, 4 header connector for external I/O's	1 No.
2	Desktop PC	Personal computer with latest configuration. Loaded with open-source IDE, simulation and program downloading software, UPS and Antivirus.	1 No.
3.	Digital Multi meter	Voltage: - DC: 600 volts; AC: 600 volts; DC Accuracy: ± 0.5% + 3 digit; AC Accuracy: ± 1% + 3-digit, Resistance: -40 M ohm; Resistance Accuracy: ± 1.5% + 3-digit, AC/DC voltage, resistance, capacitance, frequency measurement.	1 No.

IX Precautions to be followed

1. Use proper Mains cord.
2. To avoid fire or shock hazards, observe all ratings and marks on the instrument.
3. Check syntax / rules for VHDL programming.

X Procedure

1. Create the Xilinx ISE project for expected FPGA design, by doing the following in ISE:
   a. In the ISE software, select File > New Project.
   b. In the New project wizard enter the project name and location, respectively.
   c. Select HDL or Schematic as the Top-level source type, and click Next.
2. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".

3. Define Module". Enter the entity used in design and then click "Next".
4. Develop VHDL code for given problem and Synthesize the .vhd file and view RTL schematic. Ref Fig no.15.3.
5. Create Test Bench file- A test bench is HDL code that allows to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing.
6. Go to implementation to simulation tab , right click on main source file and create Test Bench file for simulation. Or
   a. Directly go to simulation tab – Process DAC behavioural – ISim simulator—select input entity bit – right click – select Force constant. Ref Fig no.15.4.a.
7. In order to view test files, select the box of "Simulation" in the "View Panel" of the "Design" panel. In the "Process Panel," double click on the "Behavioral Check Syntax" to make sure that didn't make any syntax errors while making changes.
8. Double click on "Simulate Behavioural Model" in the "Process Panel", which will open the ISim software with test bench loaded.
9. ISim simulator window will open with simulation executed, as shown in Ref Fig no.15.4.b where able to simulate designs and check for errors.
10. After simulation Build and Test DAC Using FPGA development board.
11. To create .ucf file -- Go to User Constraints – select Floorplan Area/IO/Logic (Plan Ahead) –
    Windows – Properties – now assign pin configuration and save.
12. Go to Implement Design – Run all.
13. Go to Generate Programming File and Run
14. Go to Configure Target Device and Run – Impact wizard opened – select Device 2 (as per practical Kit) -- open Bit file -- and initialize Chain – Right click on Xilinx IC and select Program.
15. Verify expected result on relevant FPGA lab kit.

SAMPLE PROGRAM 1: **Step 2. Develop VHDL code for DAC**

VHDL Code using if else statement

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use work.sine_4096_8.all;                --using package for sine wave.

entity DAC is
   Port ( clk : in  STD_LOGIC;                  --system clock
          rst : in  std_logic;                  --system reset
          cs  : out  std_logic;                 --chip select
          wr  : out  std_logic;                 --write enable
          a_b : out  std_logic;                 --output select
          led : out  std_logic_vector(7 downto 0);          --output leds
          din : out  std_logic_vector(7 downto 0)           --input to DAC
                                    );
```

```
end DAC;

architecture Behavioral of DAC is
signal clk2 : std_logic;
signal cntr : std_logic_vector(22 downto 0);
signal ramp : integer range 0 to 4100 ;
signal data : integer range 0 to 260 ;
begin

cs <= '0';
wr <= '0';
a_b<= '1';

clk2 <= cntr(1);
led <= conv_std_logic_vector(data,8);
din <= conv_std_logic_vector(data,8);

process(clk,rst)
begin
	if(rst = '1')then
		cntr <= (others => '0');
	elsif(clk'event and clk = '1')then
		cntr <= cntr + 1;
			if(cntr = "11111111111111111111")then
				cntr <= (others => '0');
			end if;
	end if;
end process;

process(rst,clk2)
begin
	if(rst = '1')then
		ramp <= 0;
	elsif(clk2'event and clk2 = '1')then
		ramp <= ramp + 1;
			if(ramp = 4095)then
				ramp <= 0;
			end if;
			data <= amp2(ramp);
	end if;
end process;
end Behavioral;
```

Problem statement for student: Design VHDL Code for DAC using if else statement.

Step 1: Develop VHDL code.

VHDL Code using if else statement performed in lab practical.

XI Resources

Table 15.2 Resources

Sr. no.	Instrument /Components	Specification	Quantity
1.			
2.			
3.			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

……..……...……...

……..……...……...

……..……...……...

……..……...……...

……...……...

……...……...

XIII Observation Table.

Table 15.3 Truth Table DAC

Sr. no.	Digital Data input	Analog output voltage in volts
1	0000 0000	
2	0000 0011	
3	0000 1111	
4	0011 1111	
5	1111 1111	

XIV Result (Output of the Program)

……...……

……..……...……..……

XV Interpretation of Results (Give meaning of the above obtained results)

……...……..……..…

……..…

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results)

……..…..…

……..……

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Write VHDL Code for ADC using FPGA board.

[Space for Answers] (If required attach separate page)

……...……...……

……..……

……..……

……..……

XVIII Suggested references for further reading

Sr no.	Link/Portal	Description
1	https://www.pantechsolutions.net/vhdl-code-to-simulate-4-bit-binary-counter-by-software-using-spartan-3-starter-kit	4 bit binary counter tutorial
2	https://www.xilinx.com/support/documentation/data_sheets/ds610.pdf	EDA tool
3	https://forums.xilinx.com/	ISE Quick start tutorial
4	https://www.rohm.com/electronics-basics/ad-da-converters/what-are-ad-da-converters	VHDL tutorial
5	https://www.elprocus.com/digital-to-analog-converter-dac-applications/	VHDL programming

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related(15 Marks)		**60% (15)**
1	Coding and Debugging ability	30%
2	Making connections of hardware	20%
3	Working with FPGA	10%
Product related (10 Marks)		**40%(10)**
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition.	15%
6	Timely Submission, Answer to sample questions.	05%
TOTAL		**100% (25)**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No. 16: Design Stepper motor controller using FPGA Board.

I. Practical Significance

FPGA provides a good combination to achieve both high speed and high precision motion on a compact hardware. It can easily drive full, half and micro-step mode applications due to the flexibility and the reduced computing time with the FPGA implementation. FPGA can drive several stepper motors simultaneously without increasing the processing time. This practical will help the students to develop skills to interface stepper motor to FPGA and rotate in clockwise and anticlockwise direction.

II. Industry/Employer Expected outcome(s)

The aim of this course is to attend following industry/employer expected outcome through various teaching learning experiences:

"Develop VLSI-based electronic circuit/component using HDL".

III. Course Level Learning outcome(s)

CO-4 Develop VHDL program for given application.

IV. Laboratory Learning Outcome(s)

Optimize the VHDL code to Rotate stepper motor in clockwise direction

V. Relevant Affective domain related Outcome(s)

- Follow safe practices.
- Maintain tools and equipment.
- Follow ethical practices.

VI. Relevant Theoretical Background

Stepper motor: -

The basic principle of stepper motor is electromagnetic induction which means while apply a current to the circuit, the EMF will be induced due to change of magnetic field of the circuit.

Working of stepper motor

The rotor shaft of the stepper motor is surrounded by the electromagnetic stator. The rotor and stator have poles which would be teethed or depends upon the motor type. When apply the electrical pulse to the stator which gets energized and produced EMF. The EMF cuts the conductor of the rotor. Now the torque will be introduced. This torque (force) is used to rotate the rotor shaft which means the rotor align itself along with stator.

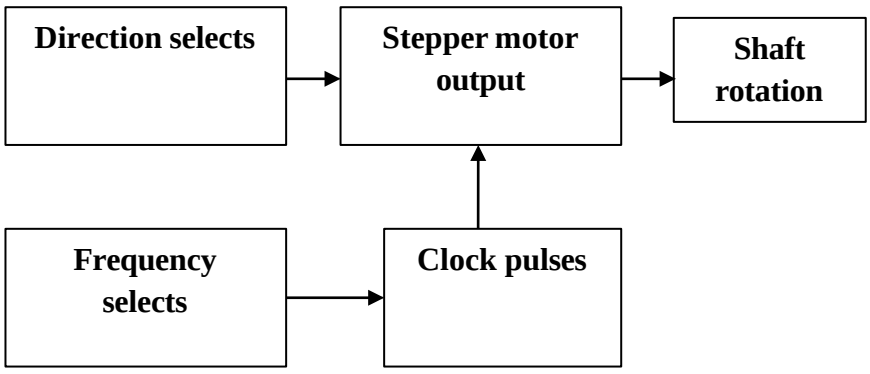


Fig. no.. 16.1: Basic Stepper motor system

The stepper motor operates at various modes such as full step, half step, Micro step Full step: -
The full step of stepper motor is 200 steps per revolution. So the angle of rotation is 1.8 degrees which is calculating from given formula.
Step angle = 360 degrees / no of steps. Step angle = 360 / 200 = 1.8 degrees.

Half step: -
The half step of stepper motor is 400 steps per revolution. So the angle of rotation is 0.9 degrees which is calculating from given formula.
Step angle = 360 degrees / no of steps. Step angle = 360 / 400 = 0.9 degrees.

Micro step:-
The advanced stepper motor has come with micro step system. Micro stepping of stepper motor means which subdivides the number of positions between poles for increasing the steps. Some micro stepping has a capable of divide a full step (1.8 degrees) into 256 micro steps. The result would be 51200 steps in one revolution (0.007 degrees/step).

Interfacing stepper motor with Spartan3 FPGA Development Kit:-
The motor is used to covert mechanical energy into electrical energy. Stepper motor is not like as any other motor which means is not rotating continuously. It is rotating step by step according to given electrical pulse. The FPGA cannot drive stepper motor directly. Because of the I/O capability of FPGA is 3.3v. So for the driver circuit are required able to drive the stepper motor. A full rotation divides into a number of equal steps. So this motor called as also stepper motor.

The four different color wires represent the four different windings. Red-Yellow and White-Blue are the four-color wires.

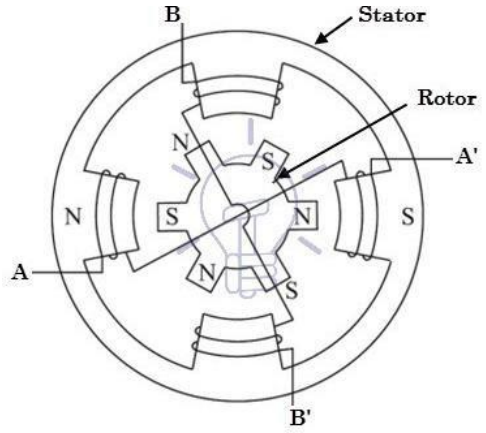


Fig. no.. 16.2: Stepper motor stator and rotator

L1-Red-1a
L2-Yellow-1b
L3-White-2a
L4-Blue-2b

The control sequence for rotating the motor is shown in the table 15.1 and 15.2.

The respective windings should be made high according to the sequence given in the table and the controller very well executes this function.

Table No. 16.1: Clockwise Rotation of stepper motor

Step	L1	L2	L3	L4
1	1	0	1	0
2	0	1	1	0
3	0	1	0	1
4	1	0	1	1

Table No. 16.2: Anti Clock Wise Rotation of stepper motor

Step	L1	L2	L3	L4
1	1	0	0	1
2	0	1	0	1
3	0	1	1	0
4	1	0	1	0

The above shown sequence is in full step mode. In a single step the motor rotates by 1.8 degree.

VII Circuit diagram used in Laboratory with related equipment rating.

a) Suggested Practical JTAG cable setup:

Fig. no..16.2: Lab Practical Setup

b) Create stepper motor .vhd file diagram :-

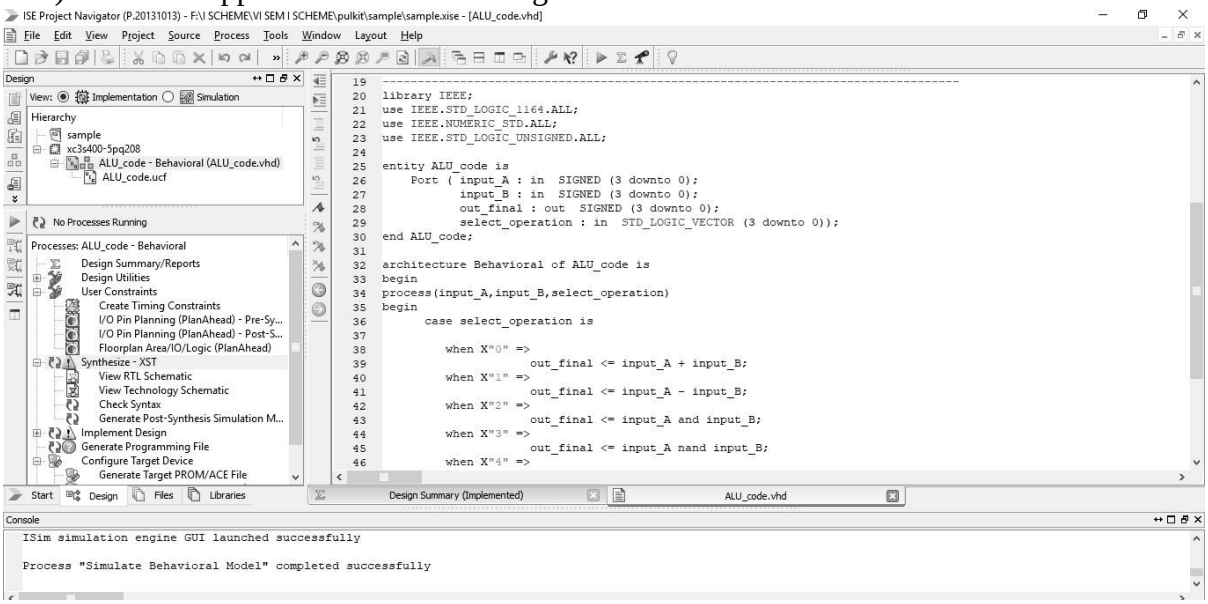


Fig. no.. 16.3: .vhd file diagram

c) Create View technology schematic diagram: -

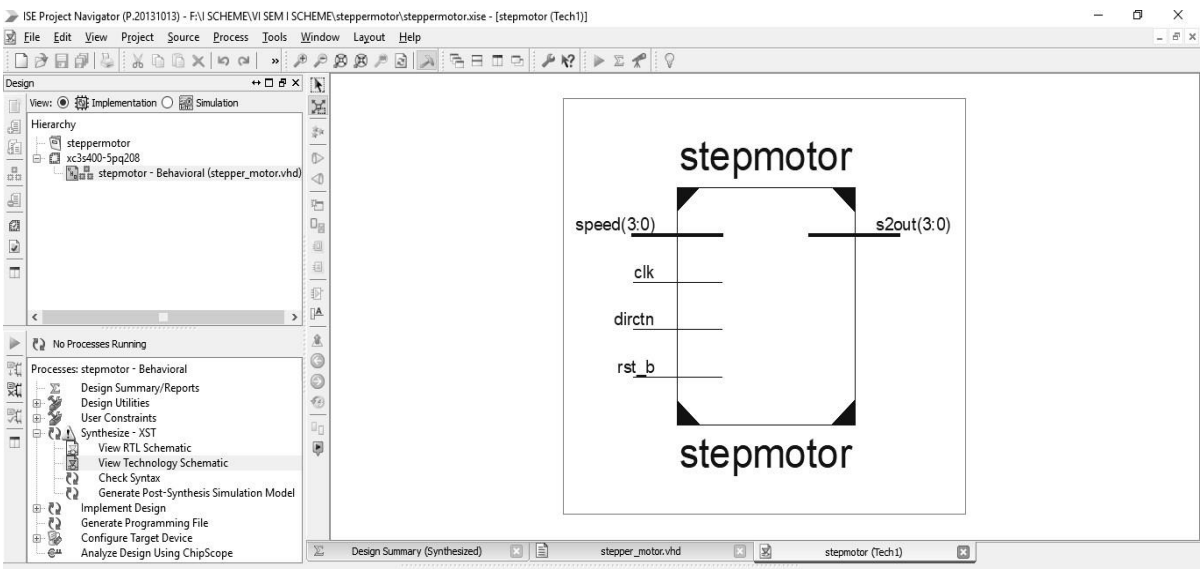


Fig. no.. 16.4: View technology schematic diagram

d) Stepper motor Simulation waveforms-

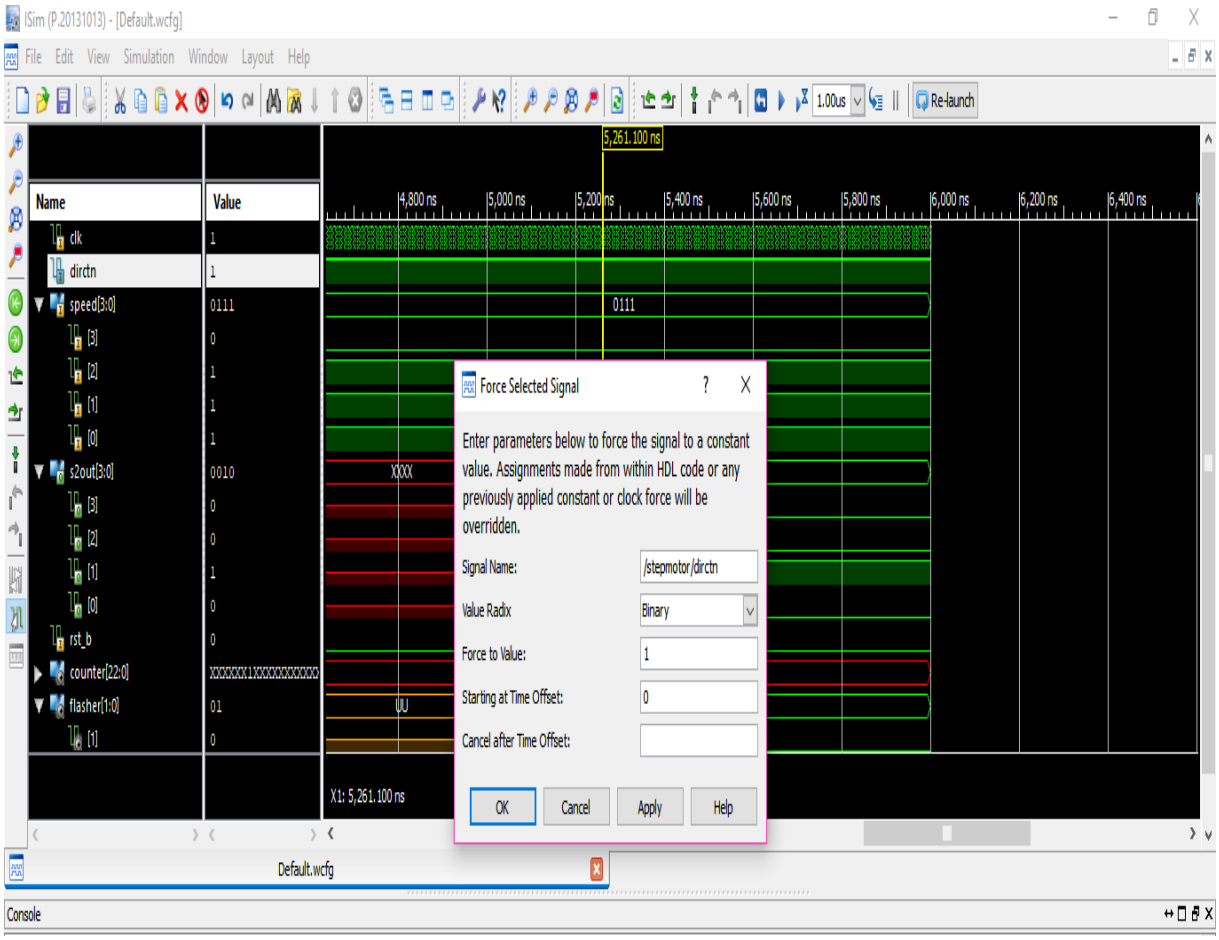


Fig. no.. 16.5: Force selected signal to generate simulation file diagram

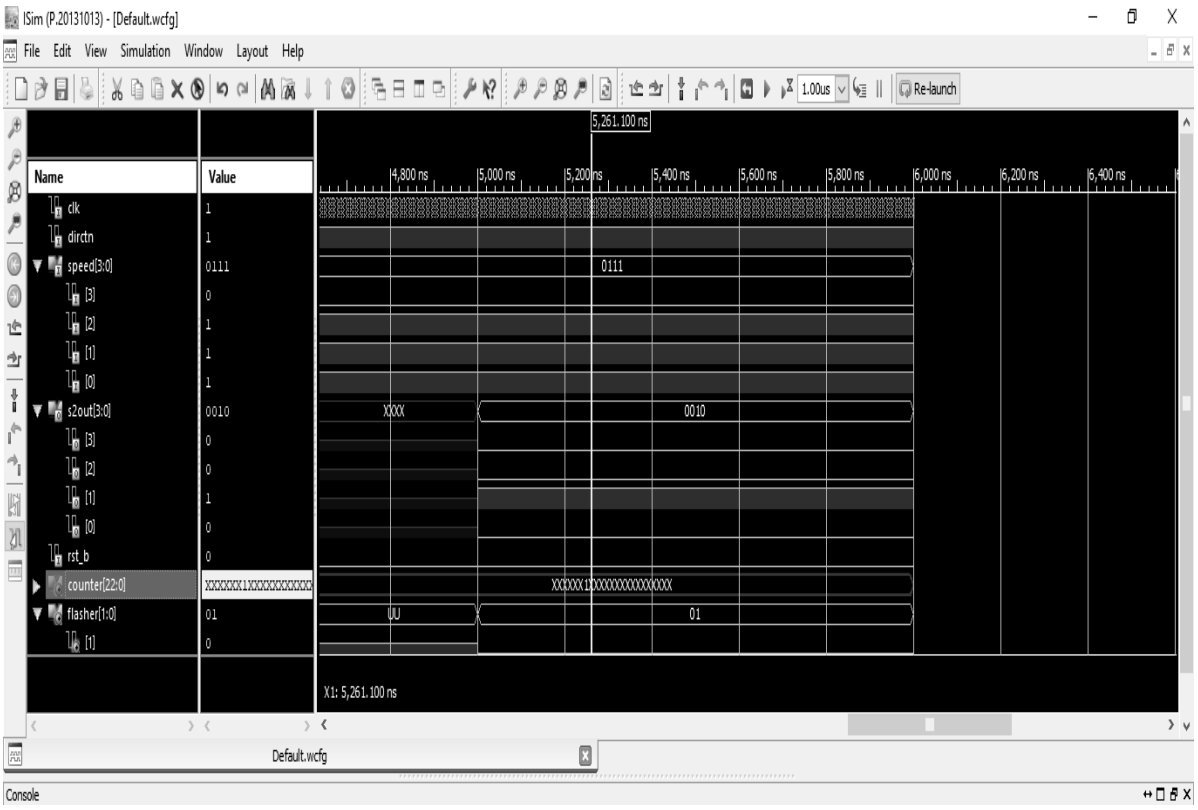


Fig. no.. 16.6: Simulation diagram

e) Actual Practical set up used in laboratory

VII. Required Resources/Apparatus/equipment with specifications

Table 16.3 Required Resources

Sr. No.	Instrument /Components	Specification	Quantity
1.	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208), On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit, On board, 2 Crystal 8MHz and 25MHz.JTAG Interface (Boundary Scan), PROM Interface (XCF02S), 40 pin, 4 header connector for external I/O's	1 No.
2.	Stepper Motor	With 5V / 1.2A rating, select frequency of rotation of motor through DIP switches, Select direction of rotation (clockwise or counter clockwise.) through toggle switch.	1 No.
3.	Desktop PC	Personal computer with latest configuration. Loaded with open-source IDE, simulation and program downloading software, UPS and Antivirus..	1 No.

IX Precautions to be followed

1.Use proper Mains cord.
2.To avoid fire or shock hazards, observe all ratings and marks on the instrument.
3.Check syntax / rules for VHDL programming.

X Procedure

1. Create the Xilinx ISE project for expected FPGA design, by doing the following in ISE:
    a. In the ISE software, select File > New Project.
    b. In new project Wizard, enter the project name and location, respectively.
    c. Select HDL or Schematic as the Top-level source type, and click Next.
2. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".
3. Define Module". Enter the entity used in design and then click "Next".
4. Develop VHDL code for given problem and synthesize the .vhd file and view RTL schematic. Ref Fig no.16.3 and 16.4.
5. Create Test Bench file- A test bench is HDL code that allows to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing.
6. Go to implementation to simulation tab, right click on main source file and create Test Bench file for simulation or directly simulate and apply entity value forcefully as shown in fig 15.5.
7. In order to view test files, select the box of "Simulation" in the "View Panel" of the "Design" panel. In the "Process Panel," double click on the "Behavioural Check Syntax" to make sure that didn't make any syntax errors while making changes.
8. Double click on "Simulate Behavioural Model" in the "Process Pane", which will open the ISim software with test bench loaded.
9. ISim simulator window will open with simulation executed, as shown in Ref Fig no.15.6. where are able to simulate designs and check for errors.
10. After simulation implement it Using FPGA development board.

11. Go to User Constraints – select Floorplan Area/IO/Logic (PlanAhead) – Windows – Properties –
    now assign pin configuration and save.
12. Go to Implement Design – Run all.
13. Go to Generate Programming File and Run
14. Go to Configure Target Device and Run – Impact wizard opened – select Device 2 (as per practical Kit set up) -- open Bit file -- and initialize Chain – Right click on Xilinx IC and select Program.

SAMPLE PROGRAM :- **Develop VHDL code for Stepper Motor.**

VHDL Code using if else statement

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use
IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--use UNISIM.VComponents.all; entity

stepmotor is
Port ( clk : in  STD_LOGIC;--input clock
dirctn : in  STD_LOGIC;--direction
speed : in  STD_LOGIC_VECTOR(3 downto 0);--frequency select
s2out: out STD_LOGIC_VECTOR(3 downto 0);--output to stepper 3
 flasher<=flasher+1; counter<="00000000000000000000000";

          end if; elsif(speed="0111")then
if(counter(17)='1')then
flasher<=flasher+1; counter<="00000000000000000000000";
end if;
elsif(speed="1111")then
if(counter(16)='1')then
flasher<=flasher+1;
counter<="00000000000000000000000";
end if;
end if;

if(dirctn='0')then--direction check
case flasher is
when "00"=>s2out<="1000”; --"0001”; --step 1 "0101”; --

when "01"=>s2out<="0100”; --"0010”; --step 2 "1001”; --

  when "10"=>s2out<="0010”; --"0100”; --step 3 "1010”; --
```

```
when "11"=>s2out<="0001”; --"1000”; --step 4 "1001”; --

      when others=>s2out<="XXXX"; end case;

 else case flasher is
      when "00"=>s2out<="0001”; --"1000”; --step 1"0110”; --

      when "01"=>s2out<="0010”; --"0100”; --step 2sout<="1010”; --

      when "10"=>s2out<="0100”; --"0010”; --step 3sout<="1001”; --

      when "11"=>s2out<="1000”; --"0001”; --step 4sout<="0101”; --

end if;
end if;
when others=>s2out<="XXXX"; end case;
end process; end Behavioral;
```

- **Problem statement for student:** Design VHDL Code for Stepper Motor using if else statement etc.

VHDL Code using if else statement

XI Resources

Table no. 16.4 Resources

Sr. No.	Instrument /Components	Specification	Quantity
1.			
2.			
3.			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

……

……

……

……

……

……

……

……

XIII Observations (use blank sheet provided if space not sufficient)

Stepper motor operations as per select direction input value is _____(verified/ not verified) using FPGA development board.

XIV Result (Output of the Program)

……

……

XV Interpretation of Results (Give meaning of the above obtained results)

……

……

XVI Conclusions and Recommendation

……

……

……

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. A stepper motor with a step angle of 7.5 degrees. Write requirement of steps per revolution.
2. Write 5 wires and 6 wires stepper motor technical specification.
3. Write FPGA limitations to drive stepper motor without driver circuit.
4. If a motor takes 180 steps per revolution then give the step angle for this motor.

[Space for Answers] (If required attach separate page)

XVIII Suggested references for further reading

Sr no.	Link/Portal	Description
1	https://www.xilinx.com/support/documentation/data_sheets/ds610.pdf	Decoder tutorial
2	https://forums.xilinx.com/	VHDL programming
3	https://www.researchgate.net/publication/258507854_FPGA_based_stepper_motor_controller	ISE Quick start tutorial
4	https://www.pantechsolutions.net/interfacing-stepper-motor-with-spartan3-fpga-development-kit	VHDL tutorial

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related(15 Marks)		**60% (15)**
	Coding and Debugging ability	30%
	Making connections of hardware	20%
	Working with FPGA	10%
Product related (10 Marks)		**40%(10)**
	Correctness of algorithm/ Flow chart	20%
	Relevance of output of the problem definition.	15%
	Timely Submission, Answer to sample questions.	05%
	TOTAL	**100% (25)**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No. 17: Implement four Bit ALU or sequence generator using FPGA

I Practical Significance

An arithmetic logic unit (ALU) represents the fundamental building block of the central processing unit of a computer. An ALU is a digital circuit used to perform arithmetic and logic operations.
A sequence detector is a sequential circuit that outputs high (1) when a particular pattern of bits sequentially arrives at its data input. This practical will help the students to develop programming skills i.e. four Bit ALU or sequence generator application on FPGA board using Xilinx ISE tools.

II Industry/Employer Expected outcome

The aim of this course is to attend following industry/employer expected outcome through various teaching learning experiences:
"Develop VLSI-based electronic circuit/component using HDL".

III. Course Level Learning outcome(s)

CO-5 Interpret VHDL simulation and synthesis

III Laboratory Learning Outcome(s)

Develop VHDL code for 4-bit ALU and simulate it using FPGA

IV Relevant Affective domain related Outcome(s)

- Follow safe practices.
- Maintain tools and equipment.
- Follow ethical practices.

V Relevant Theoretical Background

a) An arithmetic logic unit (ALU) is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer. Modern CPUs contain very powerful and complex ALUs. In addition to ALUs, modern CPUs contain a control unit (CU).

Most of the operations of a CPU are performed by one or more ALUs, which load data from input registers. A register is a small amount of storage available as part of a CPU. The control unit tells the ALU what operation to perform on that data, and the ALU stores the result in an output register. The control unit moves the data between these registers, the ALU, and memory.

All information in a computer is stored and manipulated in the form of binary numbers, i.e. 0 and 1. Transistor switches are used to manipulate binary numbers since there are only two possible states of a switch: open or closed. An open transistor, through which there is no current, represents a 0. A closed transistor, through which there is a current, represents a 1.

Following fig. shows how AND operation and OR operation is performed. Four bit two inputs and four-bit one output.

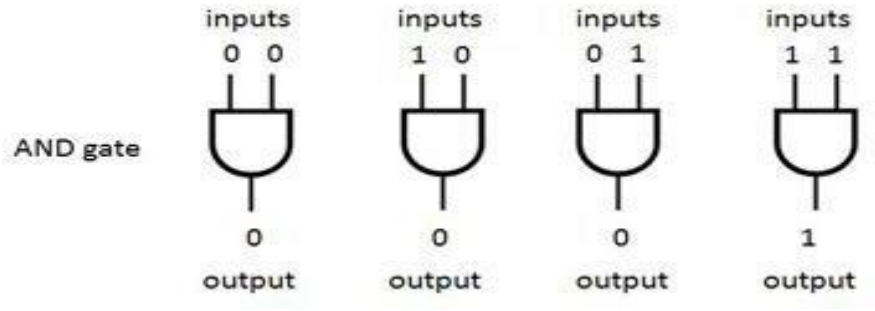


Fig. no. 17.1: AND operation**.**

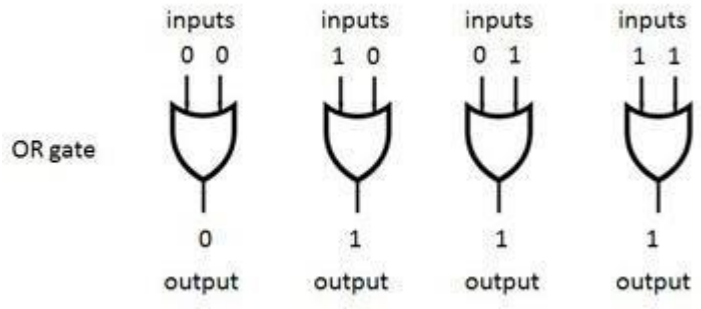


Fig. no. 17.2: OR operation**.**

Following table shows that ALU operation performed according to select the signal. With this logic we can develop the VHDL code.

Table 17.1: Signal and operation.

Select signal	Operation
0000	Addition
0001	Subtraction
0010	ANDing operation
0011	NANDing operation
0100	XORing operation
0101	XNORing operation
0110	ORing operation

b) A sequence detector is a sequential state machine. A sequence detector is a sequential state machine which takes an input string of bits and generates an output 1 whenever the target sequence has been detected.

There are two types
i.) Moore machine ii.) Mealy machine
i.) In a Moore machine, output depends only on the present state and not dependent on the input (x). Hence in the diagram, the output is written with the states. The state diagram of a Moore machine for a 101 detector is:

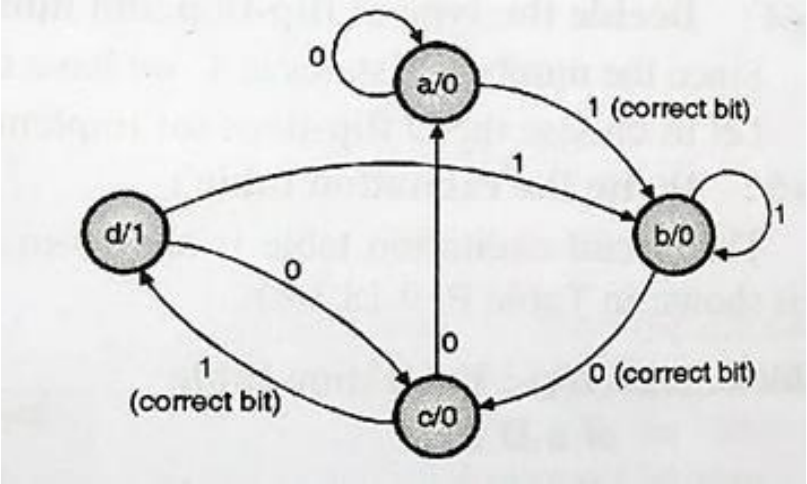


Fig. no. 17.3: State diagram of a Moore machine.

ii. In a Mealy machine, output depends on the present state and the external input (x).

VII Suggested Circuit diagram used in Laboratory with related equipment rating.

a. Suggested Practical JTAG cable setup: -

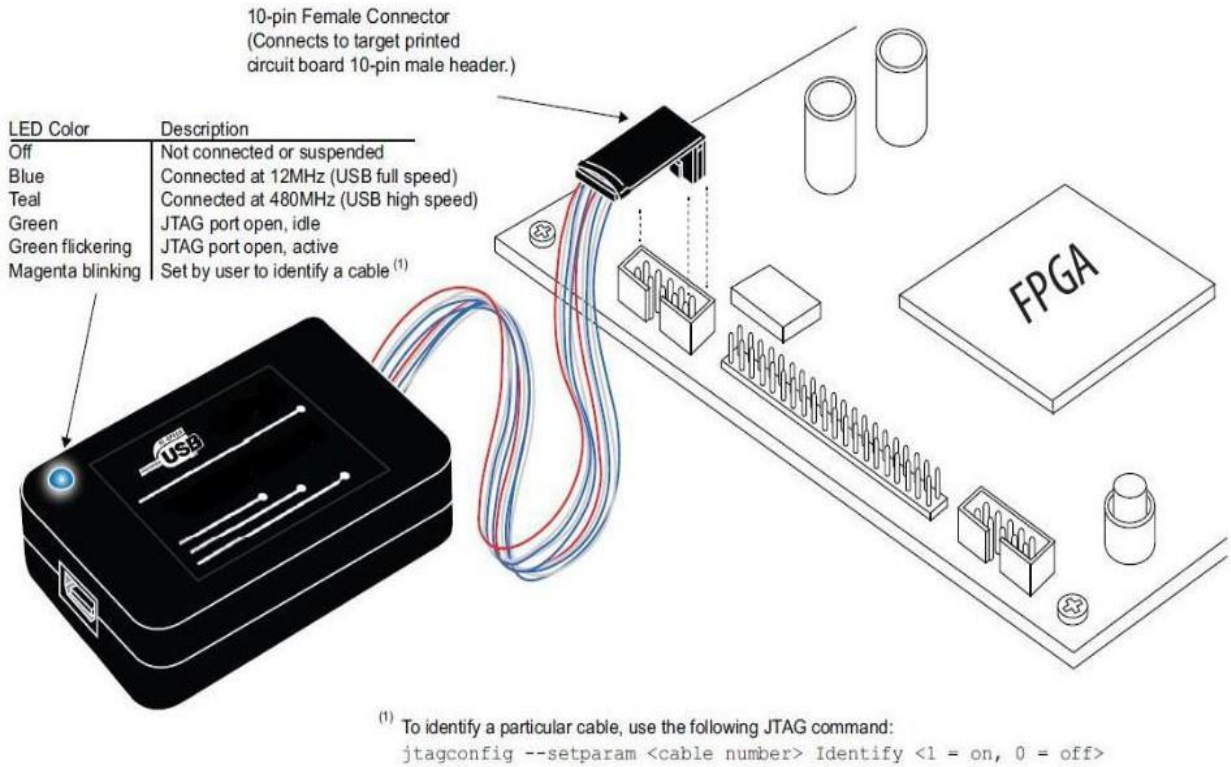


Fig. no. 17.4.a: Practical Setup with JTAG CABLE

Fig. no. 17.4.b: Lab Practical Setup

b. Create four Bit ALU .vhd file diagram :-

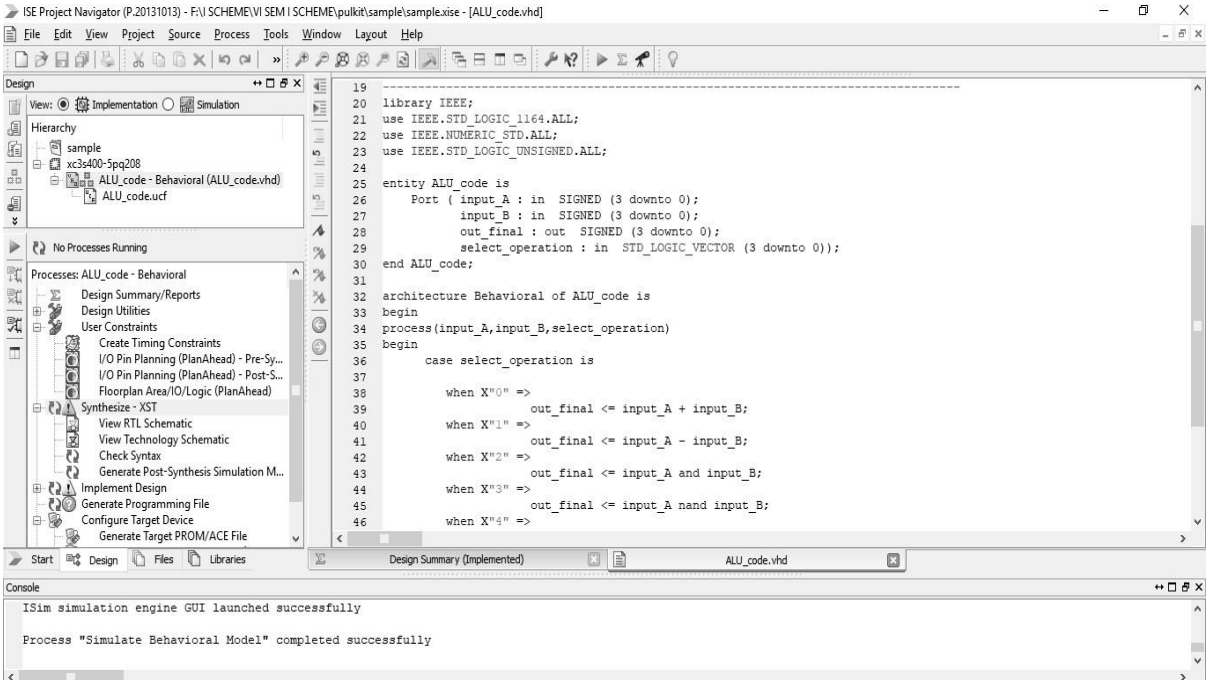


Fig. no. 17.5: .vhd file diagram

c. Apply entity values forcefully to generate simulation waveforms.

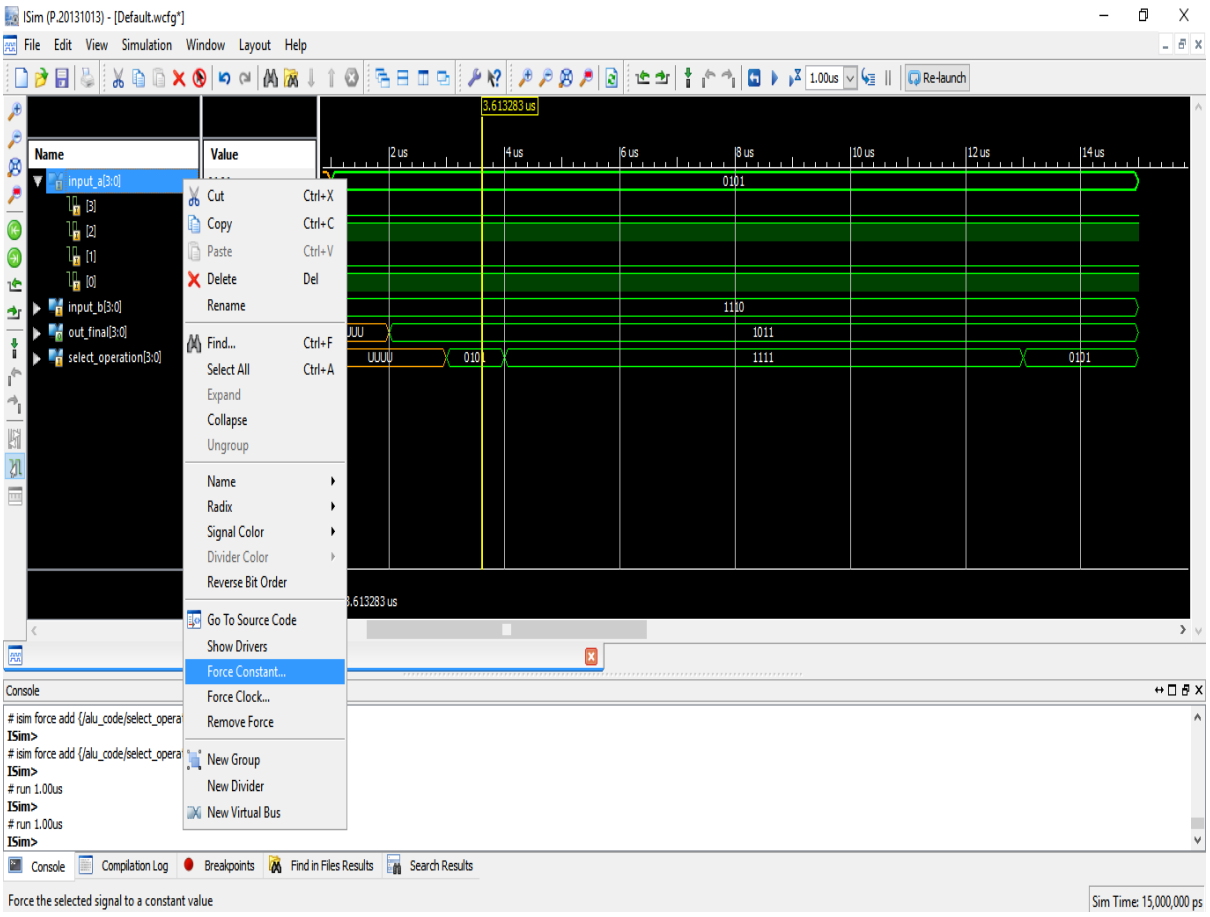


Fig. no. 17.6: test bench file diagram

d. Bit ALU Simulation waveforms-

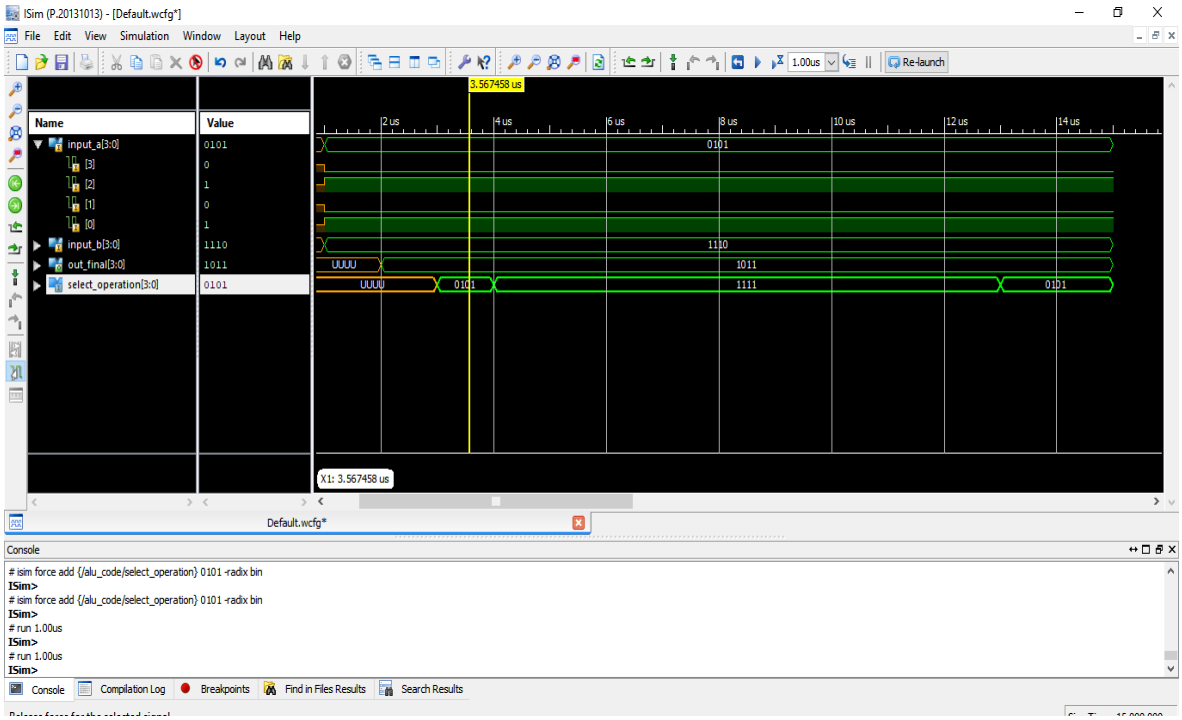


Fig. no. 17.7: Simulation diagram

e. Actual practical set up used in laboratory

VIII. Resources Required

Table 17.1 Resources Required

Sr. No.	Instrument / Components	Specification	Quantity
1	FPGA Development kit	Device: Xilinx FPGA (XC3S400 PQ208), On board +5V, +3.3V, +2.5V supply to FPGA and other hardware circuit., On board, 2 Crystal 8MHz and 25MHz.JTAG Interface (Boundary Scan), PROM Interface (XCF02S),40 pin, 4 header connector for external I/O's	1 No.
2	Desktop PC	Personal computer with latest configuration. Loaded with open-source IDE, simulation and program downloading software, UPS and Antivirus.	1 No.

IX Precautions to be followed

1. Use proper Mains cord.
2. To avoid fire or shock hazards, observe all ratings and marks on the instrument.
3. Check syntax / rules for VHDL programming.

X Procedure

1. Create the Xilinx ISE project for top-level FPGA design, by doing the following in ISE: In the ISE software, select File > New Project.

i. In the Project name and Project location fields, enter the project name and location, respectively. Select HDL or Schematic as the Top-level source type, and click Next.

2. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next".

3. Define Module". Enter the entity used in design and then click "Next".
4. Develop VHDL code for given problem and synthesize the .vhd file and view RTL schematic. Ref Fig no.17.7.
5. Create Test Bench file- A test bench is HDL code that allows to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing.
6. Go to implementation to simulation tab, right click on main source file and create Test Bench file for simulation or directly simulate and apply entity value forcefully as shown in fig 16.8.
7. In order to view test files, select the box of "Simulation" in the "View Panel" of the "Design" panel. In the "Process Panel," double click on the "Behavioural Check Syntax" to make sure that didn't make any syntax errors while making changes.
8. Double click on "Simulate Behavioural Model" in the "Process Pane", which will open the ISim

i. software with test bench loaded.

9. ISim simulator window will open with simulation executed, as shown in Ref Fig no.16.9. where are able to simulate designs and check for errors.
10. After simulation implement it using FPGA development board.
11. Go to User Constraints – select Floorplan Area/IO/Logic (Plan Ahead) – Windows – Properties –

i. now assign pin configuration and save.

12. Go to Implement Design – Run all.
13. Go to Generate Programming File and Run
14. Go to Configure Target Device and Run – Impact wizard opened – select Device 2 (as per practical Kit) -- open Bit file -- and initialize Chain – Right click on Xilinx IC and select Program.

SAMPLE PROGRAM: - Step 1. Develop VHDL code four Bit ALU.

VHDL Code using When statement

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ALU_code is
   Port (input_A: in SIGNED (3 downto 0);
         input_B: in SIGNED (3 downto 0);
         out_final: out SIGNED (3 downto 0);
        select_operation: in STD_LOGIC_VECTOR (3 downto 0));
end ALU_code;

architecture Behavioral of ALU_code is
begin
process(input_A, input_B, select_operation)
begin
                    case select_operation is
```

```
when X"0" => out_final <= input_A + input_B; when X"1" => out_final <= input_A -
input_B;
                        when X"2" =>  out_final <= input_A and input_B;
                        when X"3" => out_final <= input_A nand input_B;
                        when X"4" => out_final <= input_A xor input_B;
                        when X"5" => out_final <= input_A xnor input_B;
                        when X"6" => out_final <= input_A or input_B;
                        when others => null;
                end case;
end process;
end Behavioral;
```

Problem statement for student: Design VHDL Code for Gray to Binary encoder using if else statement logic gates etc.

Step 1: Develop VHDL code for four Bit ALU.

XI Resources

Table 17.2 Resources

Sr. No.	Instrument /Components	Specification	Quantity
1			
2			
3			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

……

……

……

……

……

……

……

……

……

……

XIII Observations (use blank sheet provided if space not sufficient)

ALU operations as per select input value is _____ (verified/ not verified) using FPGA development board.

XIV Result (Output of the Program)

……

……

XV Interpretation of Results

……

……

XVI Conclusions and Recommendation

……

……

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1.Write VHDL Code using FPGA board four Bit ALU using Case statement.

2. Write VHDL Code using FPGA board for sequence generator using if else statement.

[Space for Answers] (If required attach separate page)

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XVIII Suggested references for further reading

Sr no.	Link/Portal	Description
1	https://www.xilinx.com/support/documentation/data_sheets/ds610.pdf	VHDL programming
2	https://forums.xilinx.com/	ISE Quick start tutorial
3	https://en.wikipedia.org/wiki/Arithmetic_logic_unit	VHDL tutorial
4	https://study.com/academy/lesson/how-to-design-sequence-detectors-steps-example.html	Sequence Detector

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related(15 Marks)		**60% (15)**
	Coding and Debugging ability	30%
	Making connections of hardware	20%
	Working with FPGA	10%
Product related (10 Marks)		**40%(10)**
	Correctness of algorithm/ Flow chart	20%
	Relevance of output of the problem definition.	15%
	Timely Submission, Answer to sample questions.	05%
	TOTAL	**100% (25)**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	