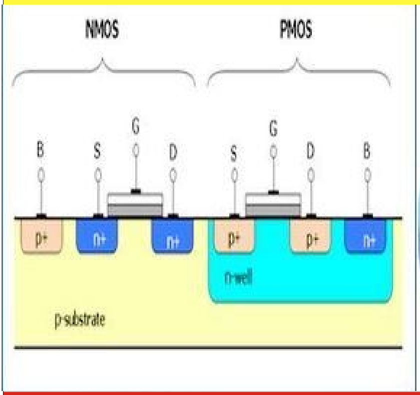
I

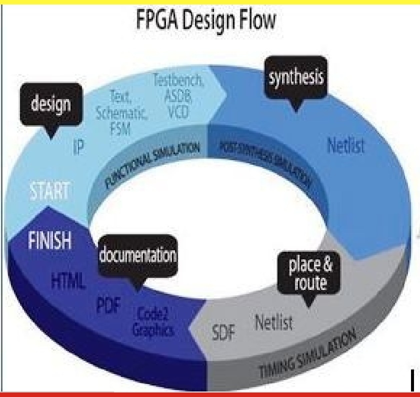
Name			
Roll No.		Year 20	20
Exam Seat No.			

ELECTRONICS GROUP | SEMESTER - VI | DIPLOMA IN ENGINEERING AND TECHNOLOGY

A LABORATORY MANUAL FOR Very Large Scale Integration with VHDL (22062)

(EJ)





MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI

(Autonomous) (ISO 9001 : 2015) (ISO / IEC 27001 : 2013)

A Laboratory Manual for

Very Large Scale Integration with VHDL

(22062)

Semester-VI

(EJ)

Maharashtra State

Board of Technical Education, Mumbai

(Autonomous) (ISO-9001-2015) (ISO/IEC 27001:2013)

Maharashtra State
Board of Technical Education, Mumbai

(Autonomous) (ISO-9001-2015) (ISO/IEC 27001:2013)

4th Floor, Government Polytechnic Building, 49, Kherwadi,

Bandra (East), Mumbai – 400051.

(Printed on November 2019)

Maharashtra State
Board of Technical Education

Certificate

This is to certify that Mr. / Ms. ……………………………………
Roll No……………………….of ……….. Semester of Diploma in……..…………………..………………………….of Institute ……………………………………….……(Code………………..) has attained pre-defined practical outcomes(PROs) satisfactorily in course **VLSI with VHDL (22062)** for the academic year 20…….to 20…..... as prescribed in the curriculum.

Place ………………. Enrollment No……………………

Date:…..................... Exam Seat No. …………………....

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

Seal of the 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 ‗I' Scheme curricula for engineering diploma programme with outcome-base education as the focus and accordingly, 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 to realize that every minute of the laboratory time need 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 ‗I' scheme laboratory manual development team designed the practical's 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 pre-determined 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 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.

In the present scenario of electronics technology, CMOS is a vital important and basic need in the design/development of almost all products in the range from consumer to industrial and telecommunication engineering area. Functional capabilities of this technology leads to advanced Very Large Scale Integration, large density of components, high speed of operation, less area with less power dissipation. Therefore imparting knowledge of VLSI and its tools is need of today. After completion of this course, students will be able to develop applications in the area of digital electronics using VLSI design tools.

Although all care has been taken to check for mistakes in this laboratory manual, yet it is impossible to claim perfection especially as this is the first edition. Any such errors and suggestions for improvement can be brought to our notice and are highly welcome

Programme Outcomes (POs) to be achieved through Practical of this Course

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

PO1. **Basic knowledge:** Apply knowledge of basic mathematics, sciences and basic engineering to solve the broad-based Electronics and Telecommunication engineering problems.

PO2. **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems

PO3. **Experiments and practice:** Experiments and practice: Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems

PO4. **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations

PO5. **The engineer and society:** Assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to practice in field of Electronics and Telecommunication engineering

PO6. **Environment and sustainability:** Apply Electronics and Telecommunication engineering solutions also for sustainable development practices in societal and environmental contexts

PO7. **Ethics:** Apply ethical principles for commitment to professional ethics, responsibilities and norms of the practice also in the field of Electronics and Telecommunication engineering.

PO8. **Individual and team work:** Function effectively as a leader and team member in diverse/ multidisciplinary teams

PO9. **Communication:** Communicate effectively in oral and written form

PO10. **Life-long learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry.

Program Specific Outcomes (PSO) (What s/he will be able to do in the Electronics and Telecommunication engineering specific industry soon after the diploma programme)

PSO1. Electronics and Telecommunication Systems: Maintain various types of Electronics and Telecommunication systems.

PSO2. EDA Tools Usage: Use EDA tools to develop simple Electronics and Telecommunication engineering related circuits.

Practical- Course Outcome matrix

Course Outcomes (COs)

a. Develop design flow for the given application using VLSI tools.
b. Interpret CMOS technology circuits with their specifications.
c. Use relevant VHDL model for given application.
d. Debug VHDL programme for the given application.
e. Maintain FPGA based circuits

Pro. No.	**Practical Outcomes (PrO)**	**CO a.**	**CO b.**	**CO c.**	**CO d.**	**CO e.**
1.*	Identify internal block and pin configuration of FPGA and CPLD using datasheet.	✓	-	-	-	-
2.*	Develop flow chart of CMOS IC fabrication using relevant website.	-	✓	-	-	-
3.*	Install EDA tool (VHDL) for VLSI application.	-	-	✓	-	-
4.*	Implement any two gates using Data flow and Behavioral model.	-	-	-	✓	-
5.	Implement Half /full adder / subtractor using FPGA.	-	-	-	✓	-
6.	Implement 8:1 multiplexer using FPGA.	-	-	-	✓	-
7.	Implement 1:8 Demultiplexer using FPGA.	-	-	-	✓	-
8.	Implement T and D-flip-flop using FPGA.	-	-	-	✓	-
9.	Implement 2:4 Decoder using FPGA.	-	-	-	✓	-
10.	Implement 8:3 Encoder using FPGA.	-	-	-	✓	-
11.	Implement up-counter using FPGA.	-	-	-	✓	-
12.	Implement synchronous counter using FPGA.	-	-	-	✓	-
13.	Implement binary to gray code converter using FPGA.	-	-	-	✓	-
14.*	Build /Test DAC using FPGA	-	-	-	-	✓
15.	Implement Stepper motor controller using FPGA.	-	-	-	-	✓
16.*	Implement four Bit ALU or sequence generator using FPGA.	-	-	-	-	✓

List of Industry Relevant Skills

The following industry relevant skills of the competency '**Maintain VLSI based electronic circuits and equipments**' are expected to be developed in students by undertaking the practical's of this laboratory manual.

1. Develop the flow chart for CMOS fabrication.
2. Identify the various blocks of FPGA and CPLD.
3. Develop 'VHDL' code using IDE tools.
4. Use Input/output port pins of FPGA.
5. Interface FPGA KIT and desktop PC.

Guidelines to Teachers

1. Teacher is expected to refer complete curriculum document and follow guidelines for implementation
2. Teacher should provide the guideline with demonstration of practical to the students with all features.
3. Teacher shall explain prior concepts to the students before starting of each practical
4. Involve students in performance of each practical.
5. Teacher should ensure that the respective skills and competencies are developed in the students after the completion of the practical exercise.
6. Teachers should give opportunity to students for hands on experience after the demonstration.
7. Teacher is expected to share the skills and competencies to be developed in the students.
8. Teacher may provide additional knowledge and skills to the students even though not covered in the manual but are expected the students by the industry.
9. Give practical assignment and assess the performance of students based on task assigned to check whether it is as per the instructions.
10. Assess the skill achievement of the students and COs of each unit.
11. At the beginning Teacher should make the students acquainted with any of the simulation software environment as few experiments are based on simulation.
12. It is desirable to paste the photo of actual practical setup or draw block diagram of practical setup.
13. Practical No.1 , 2 and 3 should not be consider for Practical (ESE-End Semester Exam).

Instructions for Students

1. Listen carefully the lecture given by teacher about course, curriculum, learning structure, skills to be developed.
2. Before performing the practical student shall read lab manual of related practical to be conducted.
3. For incidental writing on the day of each practical session every student should maintain a ***dated log book*** for the whole semester, apart from this laboratory manual which s/he has to ***submit for assessment to the teacher***.
4. Organize the work in the group and make record of all observations.
5. Students shall develop maintenance skill as expected by industries.
6. Student shall attempt to develop related hand-on skills and gain confidence.
7. Student shall develop the habits of evolving more ideas, innovations, skills etc. those included in scope of manual
8. Student shall refer technical magazines, IS codes and data books.
9. Student should develop habit to submit the practical on date and time.
10. Student should well prepare while submitting write-up of exercise.

Content

List of Practical's and Progressive Assessment Sheet

Sr. No.	Title of the practical	Page No.	Date of performance	Date of submission	Assessment marks(25)	Dated sign. of teacher	Remarks (if any)
1.*	Identify internal block and pin configuration of FPGA and CPLD using data sheet.	1					
2.*	Develop flow chart of CMOS IC fabrication using relevant website.	12					
3.*	Install EDA tool (VHDL) for VLSI application.	21					
4.*	Implement any two gates using Data flow and Behavioral model.	31					
5.	Implement Half /full adder / subtractor using FPGA.	49					
6.	Implement 8:1 multiplexer using FPGA.	62					
7.	Implement 1:8 Demultiplexer using FPGA.	74					
8.	Implement T and D-flip-flop using FPGA.	86					
9.	Implement 2:4 Decoder using FPGA.	98					
10.	Implement 8:3 Encoder using FPGA.	111					
11.	Implement up-counter using FPGA.	123					
12.	Implement synchronous counter using FPGA.	137					
13.	Implement binary to gray code converter using FPGA.	151					
14.*	Build /Test DAC using FPGA	164					
15.	Implement Stepper motor controller using FPGA.	176					
16.*	Implement four Bit ALU or sequence generator using FPGA.	190					
	TOTAL						

- *The practical marked as '*' are compulsory,*
- Column 6th marks to be transferred to Performa of CIAAN-2017.

Practical No. 01: Identify internal block and pin configuration of FPGA and CPLD using data sheet.

I Practical Significance

A field-programmable gate array (FPGA) is a regular structure of logic cells (or modules) and interconnect, which is under user's complete control. This allows the user to design, program and make changes to their circuit as per the application. CPLD is "Complex programmable logic devices", is an integrated circuit that application designers design to implement digital hardware like mobile phones. This practical will help the student to identify various blocks and pin configuration of FPGA and CPLD.

II Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronic circuits and equipments."**

- Identify the various blocks of FPGA and CPLD.
- Understand the architecture of FPGA and CPLD.
- Understand the significance of pins and their use.

IV Relevant Course Outcome

Develop design flow for the given application using VLSI tools.

V Practical Outcome

Identify internal block and pin configuration of FPGA and CPLD using data sheet..

VI Relevant Affective domain related Outcome(s)

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

VII Minimum Theoretical Background

a) 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 conFig.d 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 complete blocks of memory, ref fig.. no. 1.1.

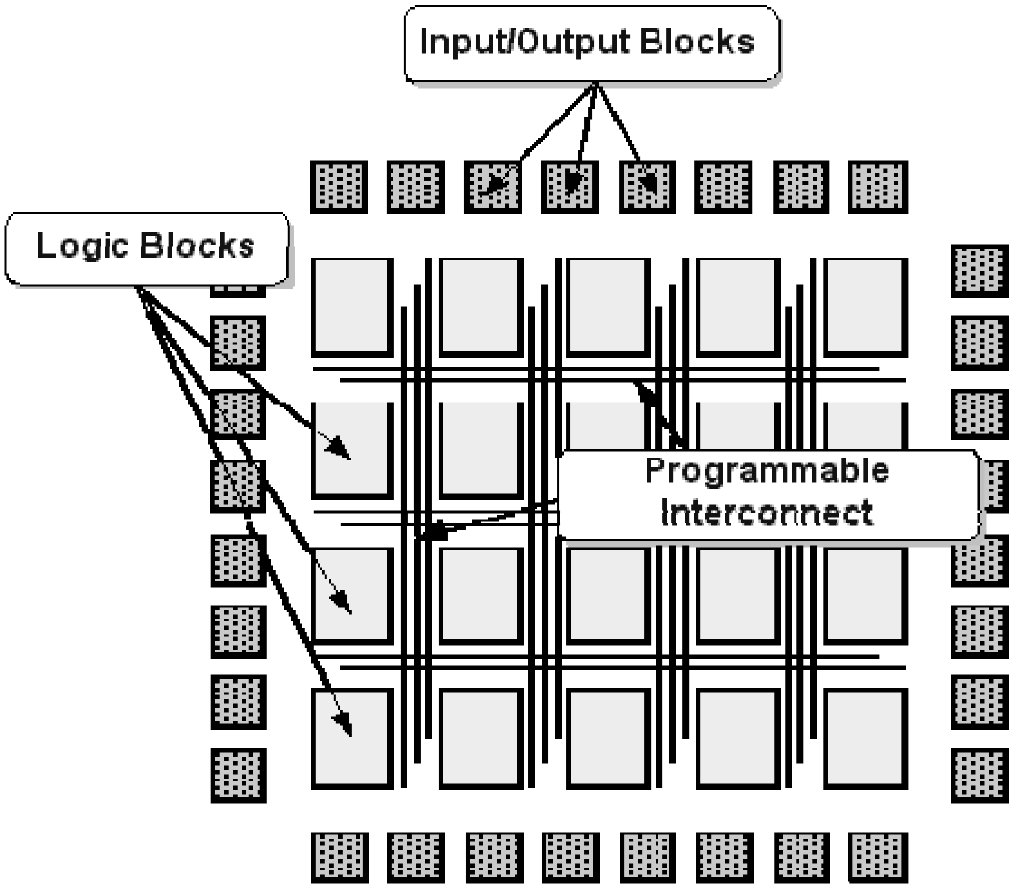


Fig 1.1 Block diagram of FPGA

Pin Configuration of FPGA [XC3S400]:

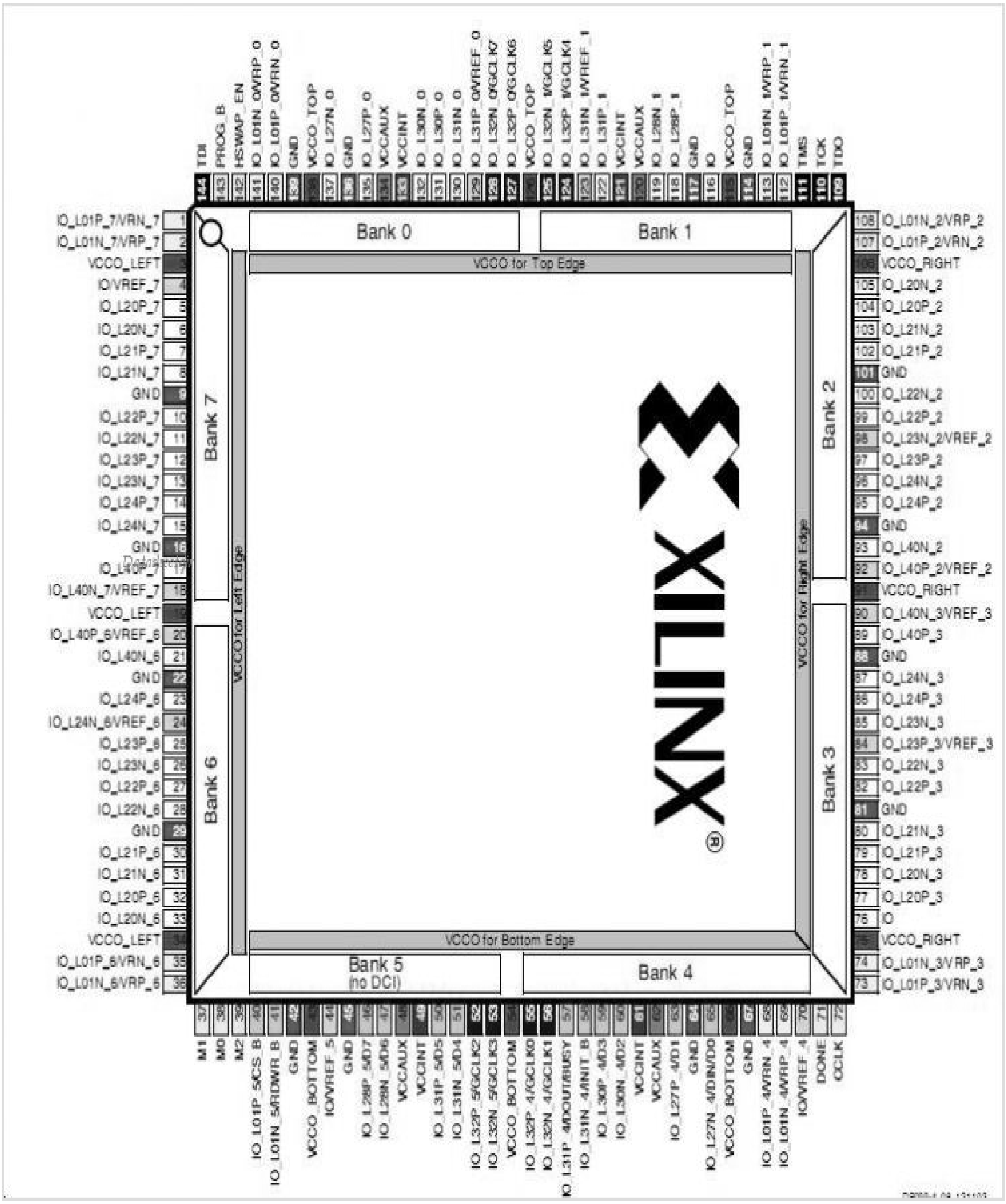


Fig 1.2 Pin Diagram of XC3S400

Table 1.1: Pin functions of XC3S400

Type	Description	Pin Name(s) in Type
I/O	Unrestricted, general-purpose user-I/O pin. Most pins can be paired together to form differential I/Os.	IO, IO_Lxxy_#
DUAL	Dual-purpose pin used in some configuration modes during the configuration process and then usually available as a user I/O after configuration. If the pin is not used during configuration, this pin behaves as an I/O-type pin. There are 12 dual-purpose configuration pins on every package.	IO_Lxxy_#/DIN/D0, IO_Lxxy_#/D1, IO_Lxxy_#/D2, IO_Lxxy_#/D3, IO_Lxxy_#/D4, IO_Lxxy_#/D5, IO_Lxxy_#/D6, IO_Lxxy_#/D7, IO_Lxxy_#/CS_B, IO_Lxxy_#/RDWR_B, IO_Lxxy_#/BUSY/DOUT, IO_Lxxy_#/INIT_B
CONFIG	Dedicated configuration pin. Not available as a user-I/O pin. Every package has seven dedicated configuration pins. These pins are powered by VCCAUX.	CCLK, DONE, M2, M1, M0, PROG_B, HSWAP_EN
JTAG	Dedicated JTAG pin. Not available as a user-I/O pin. Every package has four dedicated JTAG pins. These pins are powered by VCCAUX.	TDI, TMS, TCK, TDO
DCI	Dual-purpose pin that is either a user-I/O pin or used to calibrate output buffer impedance for a specific bank using Digital Controlled Impedance (DCI). There are two DCI pins per I/O bank.	IO/VRN_# IO_L xxy_ #/VR N_# IO/V RP_# IO_Lxxy_#/VRP_#
VREF	Dual-purpose pin that is either a user-I/O pin or, along with all other VREF pins in the same bank, provides a reference voltage input for certain I/O standards. If used for a reference voltage within a bank, all VREF pins within the bank must be connected.	IO/VREF_# IO_Lxxy_#/VREF_#
GND	Dedicated ground pin. The number of GND pins depends on the package used. All must be connected.	GND
VCCAUX	Dedicated auxiliary power supply pin. The number of VCCAUX pins depends on the package used. All must be connected to +2.5V.	VCCAUX
VCCINT	Dedicated internal core logic power supply	VCCINT

Type	Description	Pin Name(s) in Type
	pin. The number of VCCINT pins depends on the package used. All must be connected to +1.2V.	
VCCO	Dedicated I/O bank, output buffer power supply pin. Along with other VCCO pins in the same bank, this pin supplies power to the output buffers within the I/O bank and sets the input threshold voltage for some I/O standards.	VCCO_# TQ144 Package Only: VCCO_LEFT, VCCO_TOP, VCCO_RIGHT, VCCO_BOTTOM
GCLK	Dual-purpose pin that is either a user-I/O pin or an input to a specific global buffer input. Every package has eight dedicated GCLK pins.	IO_Lxxy_#/GCLK0, IO_Lxxy_#/GCLK1, IO_Lxxy_#/GCLK2, IO_Lxxy_#/GCLK3, IO_Lxxy_#/GCLK4, IO_Lxxy_#/GCLK5, IO_Lxxy_#/GCLK6, IO_Lxxy_#/GCLK7
N.C.	This package pin is not connected in this specific device/package combination but may be connected in larger devices in the same package.	N.C.

FPGA Development board:

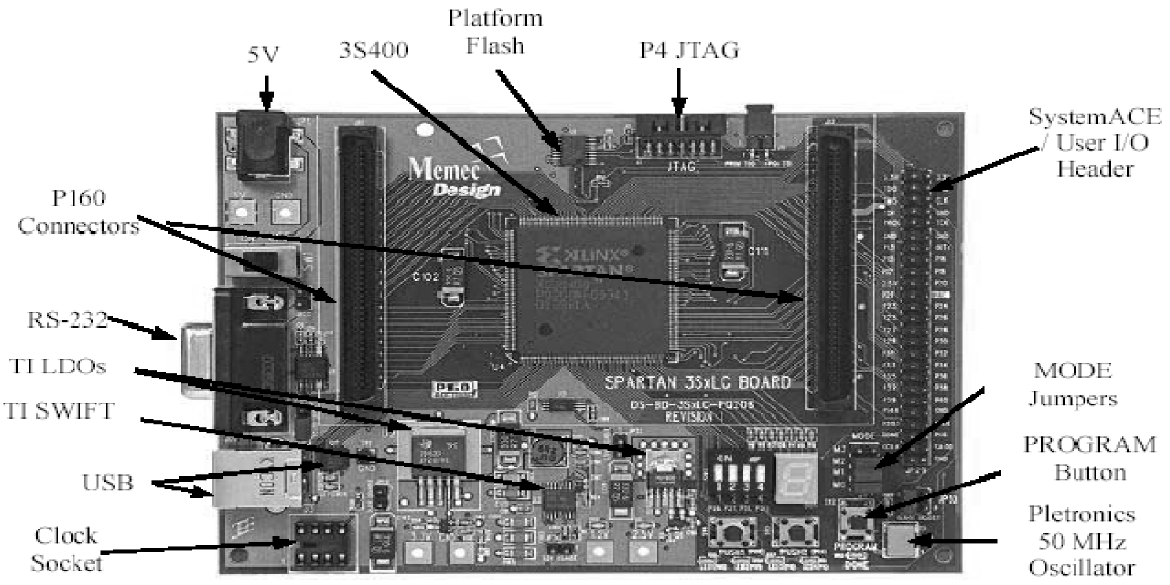


Fig 1.3 FPGA Development Board

b) 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 macrocells. The AND/OR array is reprogrammable and can perform a multitude of logic functions. Macrocells are functional blocks that perform combinatorial or sequential logic, and also have the added flexibility for true or complement, along with varied feedback paths. Ref fig.. no. 1.4.

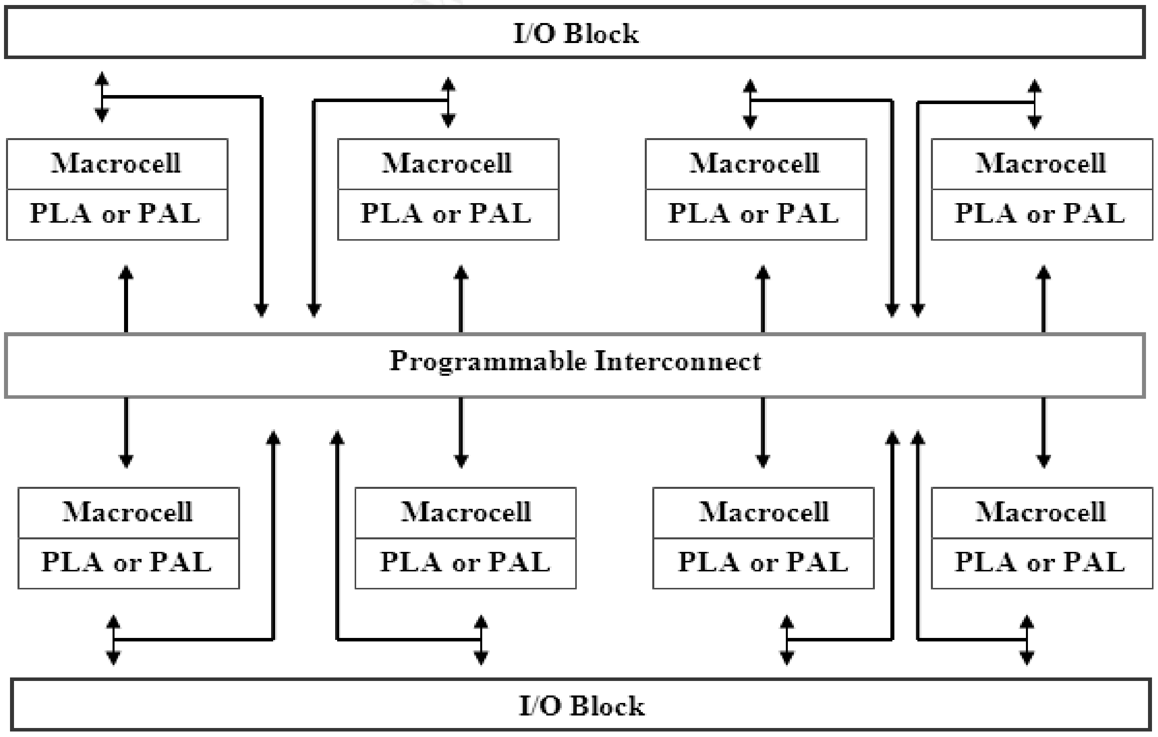


Fig 1.4 Block Diagram of CPLD

Pin configuration of CPLD :

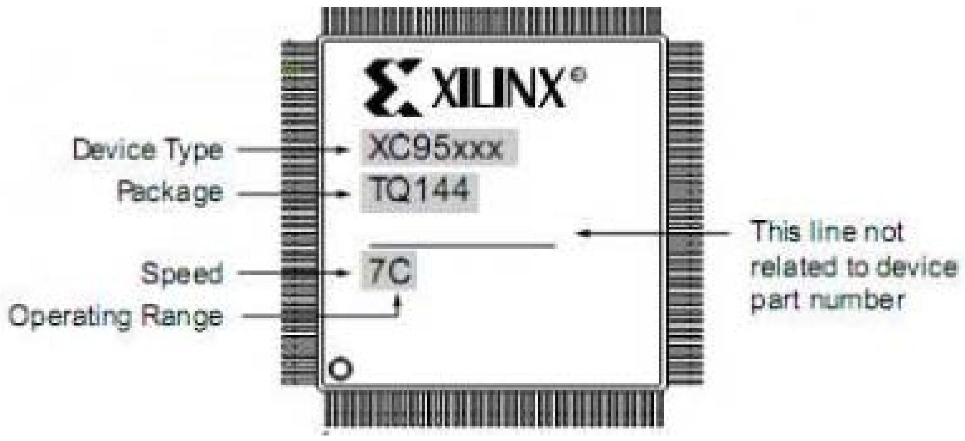


Fig 1.5 Package Pinout Diagram

Table no 1.2: General Pin Functions of CPLD IC

Sr. No	Pin name	Functions
1.	IO pins	Each external I/O pin can be used as an input, an output, or a bidirectional pin according to device programming. The I/O pins at the bottom are also used for special purposes.
2.	GCK	"Global Clocks" (GCK).Each macro cell can be programmed to

		use a selected clock input.
3.	GSR	"Global Set/Reset" (GSR).Each macro cell can use this signal as an asynchronous Preset or Clear.
4.	GTS	"Global Three State Controls" (GTS).One of the signals can be selected in each macro cell to output enable the corresponding output driver when the macro cell's output is hooked to an external I/O pin.

CPLD Development Board:

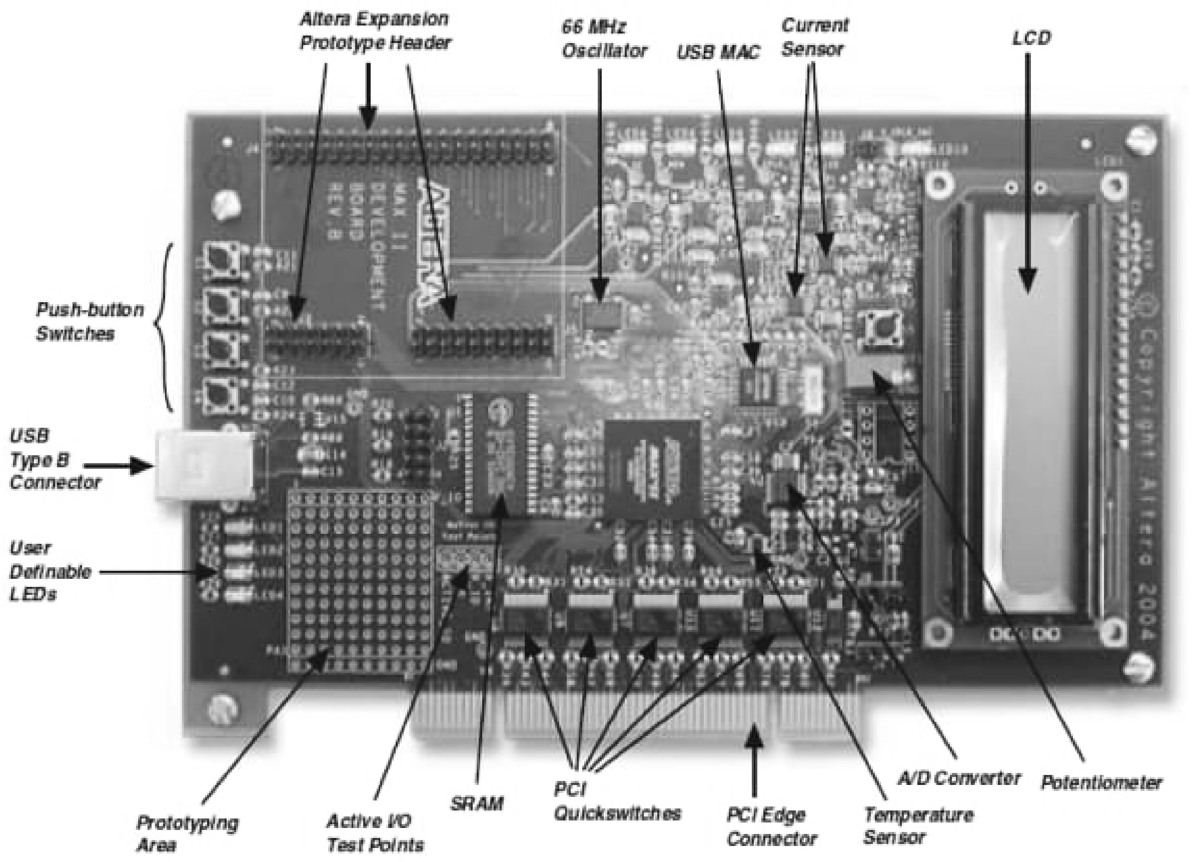


Fig1.6 CPLD Development Board

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.

Sr. No	Instrument / Components	Specification	Quantity
2.	Desktop PC	Loaded with open source IDE, simulation and program downloading software.	1 No.
3.	Data Sheets of Xilinx CPLD and FPGA		

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 Resources Used

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

XI Precautions Followed (use blank sheet provided if space not sufficient)

....

....

....

....

....

....

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

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

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

XIII Conclusion and Recommendation

....

....

....

....

....

XIV 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. List the features of FPGA.
2. List the features of CPLD.
3. Compare FPGA and CPLD on the basis of following points:
   a. Power consumption.
   b. Number of input and output pins.
   c. Complexity in board design and layout.
   d. Prediction of speed performance of design.

[Space for Answers]

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

XV References / Suggestions for further reading

1 http://www.ti.com/lit/ds/symlink/ads1258-ep.pdf
2 http://www.datasheetdir.com/ADS1258+24bit-Analog-Digital-Converter
3 http://www.datasheetdir.com/CPLD
4 https://www.dataman.com/media/datasheet/Atmel/ATF1502ASV_doc1615.pdf

XVI Assessment Scheme

	Performance indicators	**Weightage**
	Process related (15 Marks)	**60%**
1	Identifying the development board	30%
2	Identifying components on developer kit	20%
3	Follow ethical practices.	10%
	Product related (10 Marks)	**40%**
4	Correct pin functions	20%
5	Answer to sample questions.	15%
6	Timely Submission of report Answer to sample questions.	05%
	TOTAL (25 Marks)	**100%**

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 02: Develop flowchart of CMOS IC fabrication using relevant website.

I Practical Significance

Complementary metal–oxide–semiconductor (CMOS) is a type of MOSFET (metal—semiconductor field-effect transistor) semiconductor device fabrication process used for constructing integrated circuits (ICs). CMOS technology is used in microprocessors, microcontrollers, memory chips, and other digital logic circuits. CMOS technology is also used for several analog circuits such as image sensors (CMOS sensor), data converters, and highly integrated transceivers for many types of communication. This practical will help the student to understand the CMOS IC fabrication process.

II Relevant Program Outcomes (POs)

- **Discipline knowledge: Apply** Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: ‘**Maintain VLSI based electronic circuits and equipments.”**

- Understand the CMOS IC fabrication.
- To develop the flowchart for CMOS fabrication.

IV Relevant Course Outcome

Interpret CMOS technology circuits with their specifications.

V Practical Outcome

Develop flowchart of CMOS IC fabrication using relevant website.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum Theoretical Background

For less power dissipation requirement CMOS technology is used for implementing transistors. If we require a faster circuit then transistors are implemented over IC using BJT. Fabrication of CMOS transistors as IC’s can be done in three different methods.

- The N-well / P-well technology, where n-type diffusion is done over a p-type substrate or p-type diffusion is done over n-type substrate respectively.

- The Twin well technology, where NMOS and PMOS transistor are developed over the wafer by simultaneous diffusion over an epitaxial growth base, rather than a substrate.

  The silicon On Insulator process, where rather than using silicon as the substrate an insulator material is used to improve speed and latch-up susceptibility.

MOS Fabrication Steps:

The CMOS fabrication process flow is conducted using following basic fabrication steps while manufactured using N- well/P-well technology.

Making of CMOS using N well

Step 1: First we choose a substrate as a base for fabrication. For N- well, a P-type silicon substrate is selected.

P- Substrate

Fig 2.1: Selection of Substrate

Step 2 – Oxidation: The selective diffusion of n-type impurities is accomplished using SiO2 as a barrier which protects portions of the wafer against contamination of the substrate. SiO_2 is laid out by oxidation process done exposing the substrate to high-quality oxygen and hydrogen in an oxidation chamber at approximately 1000^0c

oxidation
P- Substrate

Fig 2.2: Oxidation

Step 3 – Growing of Photoresist: At this stage to permit the selective etching, the SiO2 layer is subjected to the photolithography process. In this process, the wafer is coated with a uniform film of a photosensitive emulsion.

Photoresist
oxidation
P- Substrate

Fig 2.3: Growing of Photoresist

Step 4 – Masking: This step is the continuation of the photolithography process. In this step, a desired pattern of openness is made using a stencil. This stencil is used as a mask over the photoresist. The substrate is now exposed to **UV rays** the photoresist present under the exposed regions of mask gets polymerized.

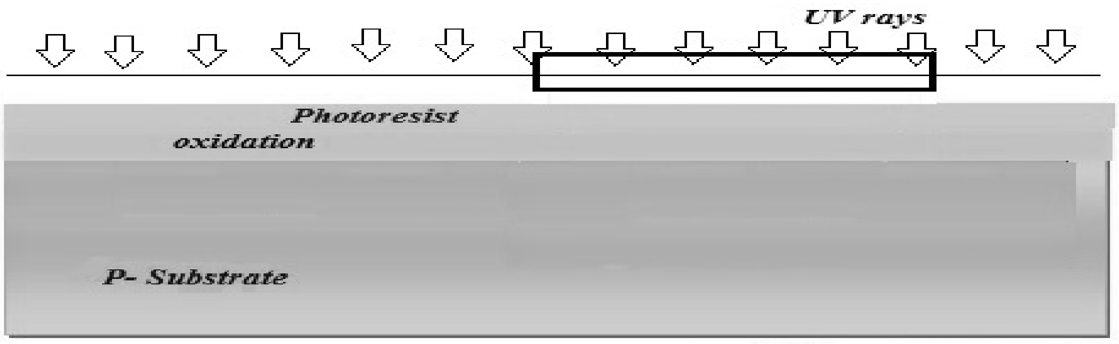


Fig 2.4: Masking

Step 5 – Removal of Unexposed Photoresist: The mask is removed and the unexposed region of photoresist is dissolved by developing wafer using a chemical such as Trichloroethylene.

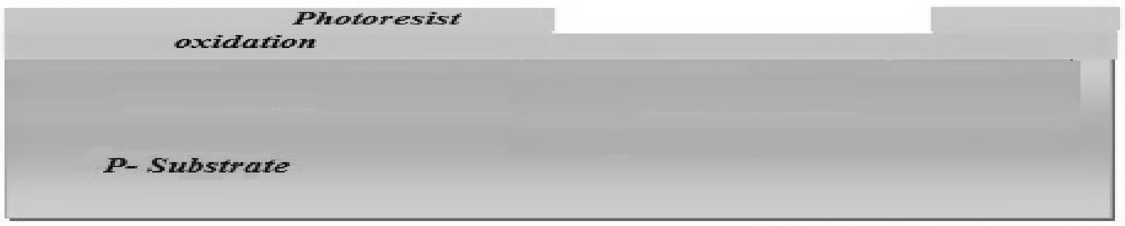


Fig 2.5: Removal of Unexposed Photoresist

Step 6 – Etching: The wafer is immersed in an etching solution of hydrofluoric acid, which removes the oxide from the areas through which dopants are to be diffused.

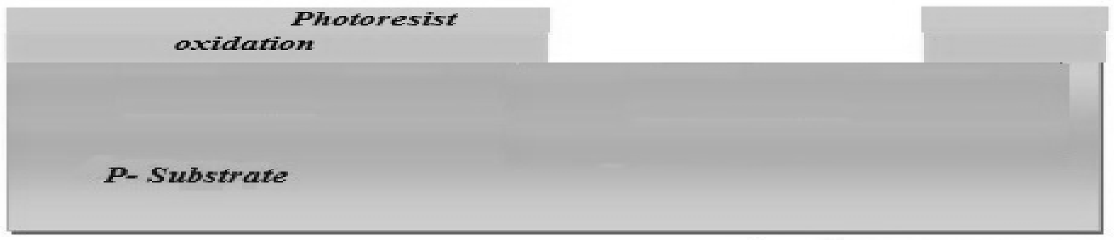


Fig 2.6: Etching

Step 7 – Removal of Whole Photoresist Layer: During the **etching process**, those portions of SiO2 which are protected by the photoresist layer are not affected. The photoresist mask is now stripped off with a chemical solvent (hot H2SO4).

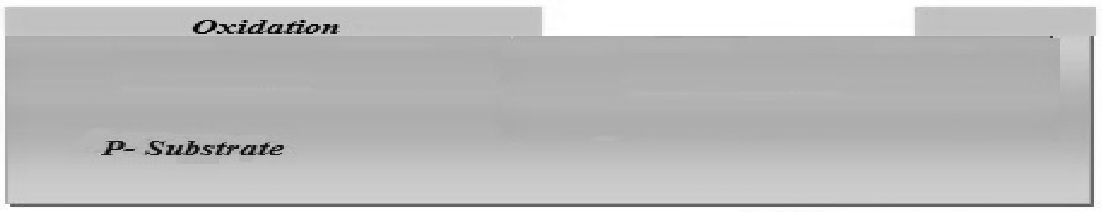


Fig 2.7: Removal of **Photoresist Layer**

Step 8 – Formation of N-well: The n-type impurities are diffused into the p-type substrate through the exposed region thus forming an N- well.

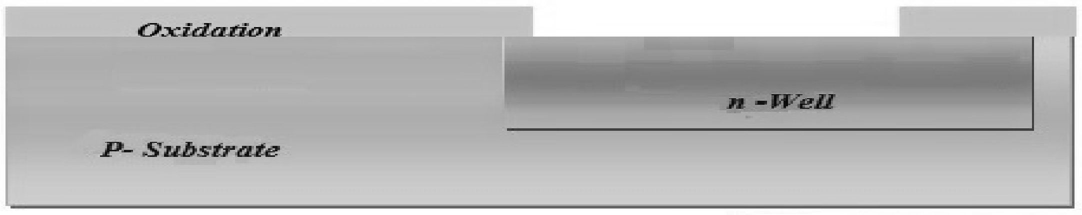


Fig 2.8: Formation of N-Well

Step 9 – Removal of SiO2: The layer of SiO2 is now removed by using hydrofluoric acid.

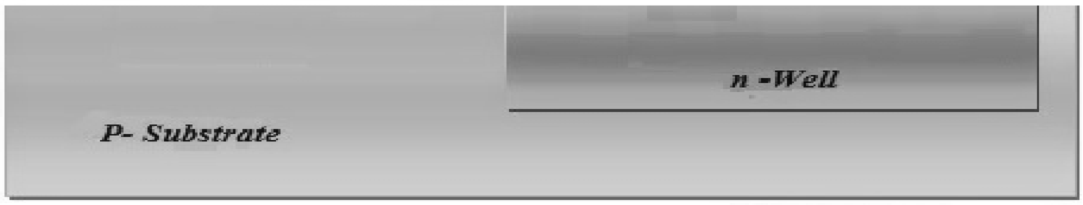


Fig 2.9: SiO2 Removal of SiO2

Step 10 – Deposition of Polysilicon: The misalignment of the gate of a **CMOS transistor** would lead to the unwanted capacitance which could harm circuit. So to prevent this “Self-aligned gate process” is preferred where gate regions are formed before the formation of source and drain using ion implantation.

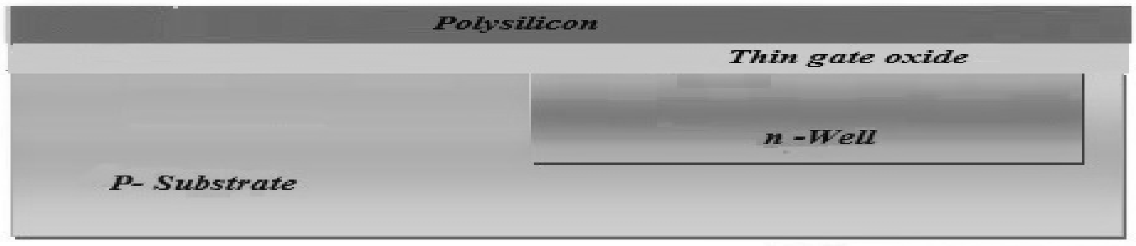


Fig 2.10: Polysilicon Deposition

Polysilicon is used for formation of the gate because it can withstand the high temperature greater than 8000^0c when a wafer is subjected to annealing methods for formation of source and drain. Polysilicon is deposited by using **Chemical Deposition Process** over a thin layer of gate oxide. This thin gate oxide under the Polysilicon layer prevents further doping under the gate region.

Step 11 – Formation of Gate Region: Except the two regions required for formation of the gate for **NMOS and PMOS transistors** the remaining portion of Polysilicon is stripped off.

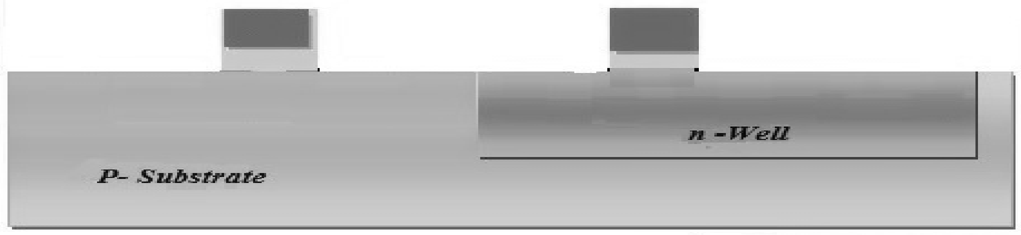


Fig 2.11: Formation of Gate Region

Step 12 – Oxidation Process: An oxidation layer is deposited over the wafer which acts as a shield for further **diffusion and metallization processes**.

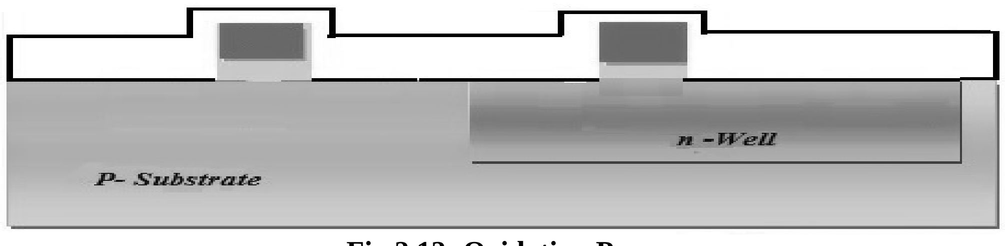


Fig 2.12: Oxidation Process

Step 13 – Masking and Diffusion: For making regions for diffusion of n-type impurities using masking process small gaps are made.

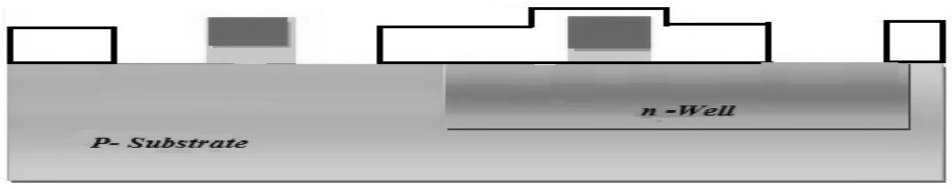


Fig 2.13.a: Masking and Diffusion

Using diffusion process three n+ regions are developed for the formation of terminals of NMOS.

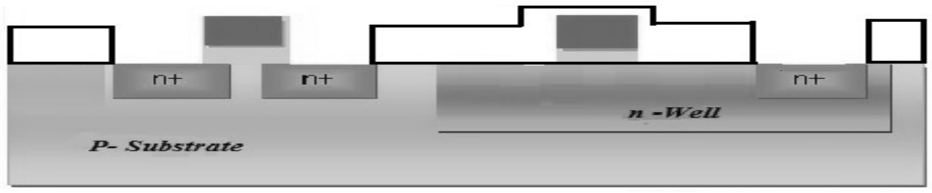


Fig 2.13.b: Masking and Diffusion

Step 14 – Removal of Oxide: The oxide layer is stripped off.

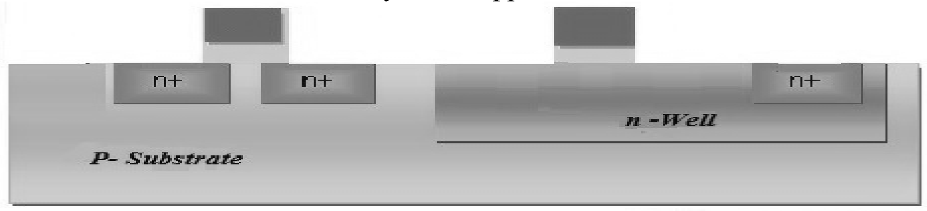


Fig 2.14: Removal of Oxide

Step 15 – P-type Diffusion: Similar to the n-type diffusion for forming the terminals of PMOS p-type diffusion are carried out.

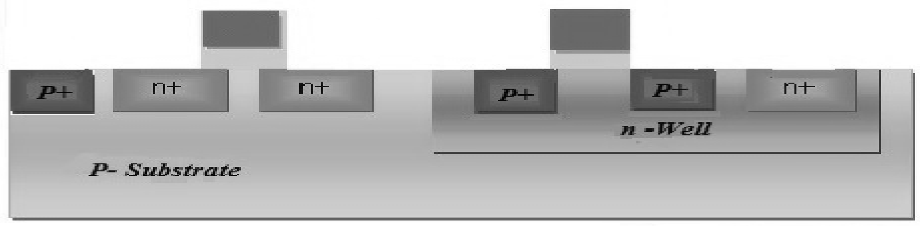


Fig 2.15: P-Type Diffusion

Step 16 – Laying of Thick Field oxide: Before forming the metal terminals a thick field oxide is laid out to form a protective layer for the regions of the wafer where no terminals are required.

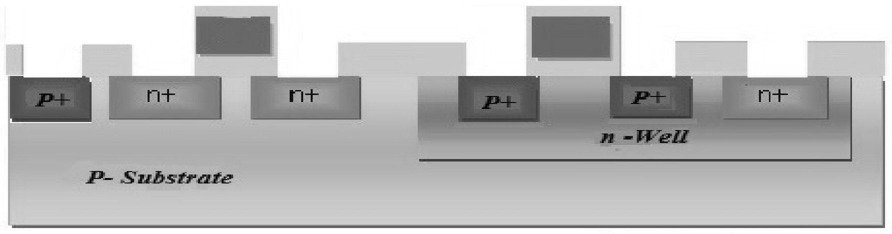


Fig 2.16: Laying of Thick Oxide

Step 17 – Metallization: This step is used for the formation of metal terminals which can provide interconnections. Aluminum is spread on the whole wafer.

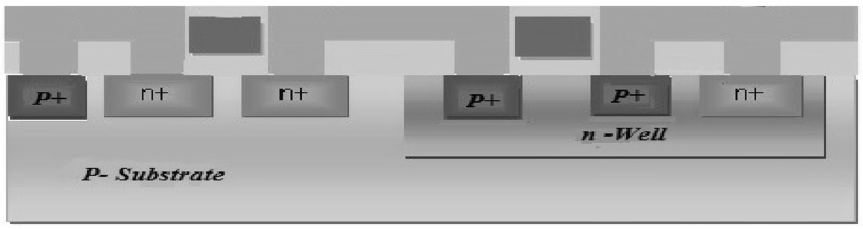


Fig 2.17: Metallization

Step 18 – Removal of Excess Metal: The excess metal is removed from the wafer.

Step 19 – Formation of Terminals: In the gaps formed after removal of excess metal terminals are formed for the interconnections.

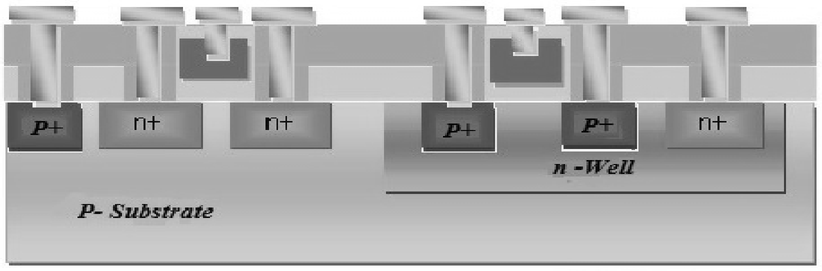


Fig 2.18: Terminal Formation

Step 20 – Assigning the Terminal Names: Names are assigned to the terminals of **NMOS and PMOS transistors**.

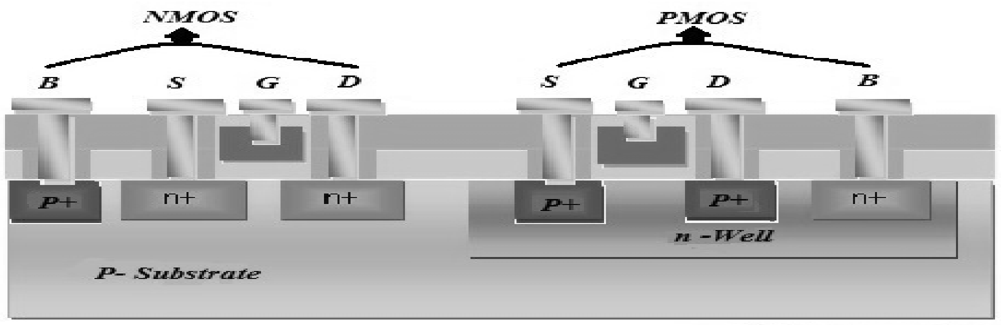


Fig 2.19: Terminal Naming

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

Draw the flowchart for the CMOS IC fabrication using N well Process.

IX Conclusion and Recommendation

....
....
....
....
....

X 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. List the CMOS applications.
2. Draw the process diagram of CMOS Fabrication using P well process.
3. Draw the flowchart for the CMOS fabrication using Twin tub process.

[Space for Answers]

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

XI References / Suggestions for further reading

1 https://www.elprocus.com/the-fabrication-process-of-cmos-transistor/
2 https://www.slideshare.net/cmos-fabrication
3 https://www.electronics-tutorial.net/CMOS-Processing-Technology/Twin-tub-Process/

XII Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**60%**
1	Identifying the fabrication process	30%
2	Listing the steps in fabrication process.	20%
3	Follow ethical practices.	10%
Product related (10 Marks)		**40%**
4	Correct material selection	20%
5	Answer to sample questions.	15%
6	Timely Submission of report Answer to sample questions.	05%
	Total (25 Marks)	**100%**

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 03: Install EDA tool for VLSI application.

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 touches 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).

II Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronic circuits and equipments.”**

- Install the EDA tool for VLSI applications.
- To use the EDA tool in implementing of various applications.

IV Relevant Course Outcome

Use relevant VHDL model for given application.

V Practical Outcome

Install EDA tool for VLSI application.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum 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 config. 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.

VIII Resources Required:

Sr. No.	Instrument / Components EDA tool	Specification	Quantity
1.	Desktop PC	Preferably OS: Windows 7	01
2.	EDA tool	Xilinx ver. 14.7	01

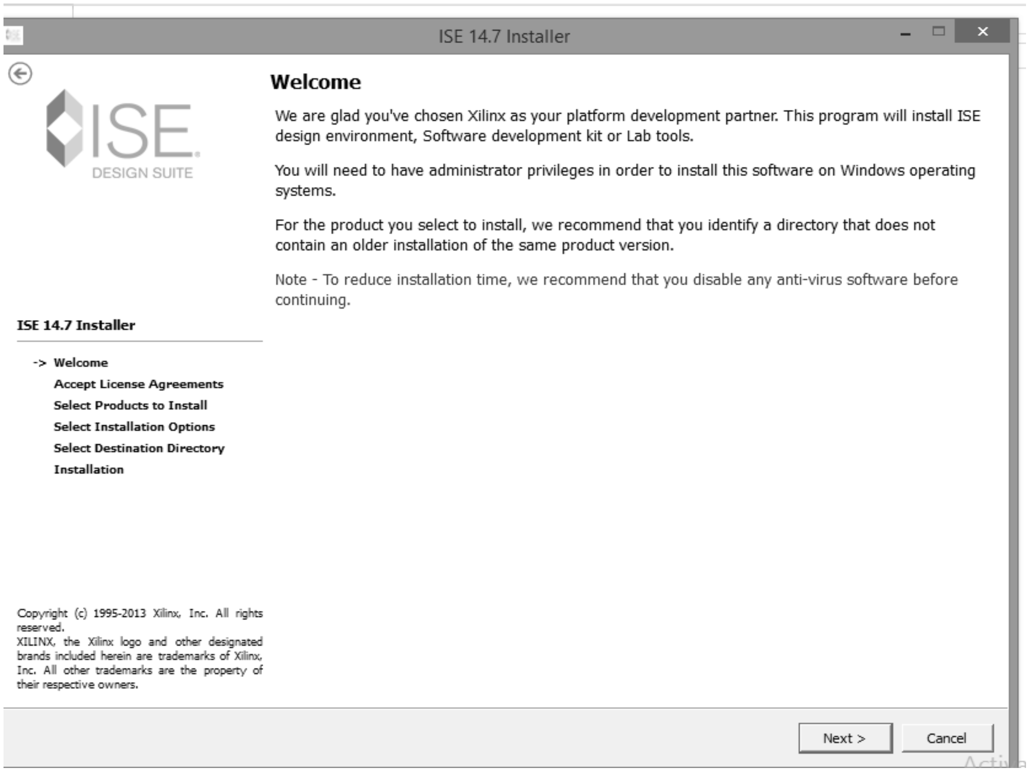
IX Precautions to be followed:

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

X Procedure:

Steps for installing Xilinx Software (EDA Tool)

1. Click on the setup file to install Xilinx software.



ISE 14.7 Installer

Welcome

We are glad you've chosen Xilinx as your platform development partner. This program will install ISE design environment, Software development kit or Lab tools.

You will need to have administrator privileges in order to install this software on Windows operating systems.

For the product you select to install, we recommend that you identify a directory that does not contain an older installation of the same product version.

Note - To reduce installation time, we recommend that you disable any anti-virus software before continuing.

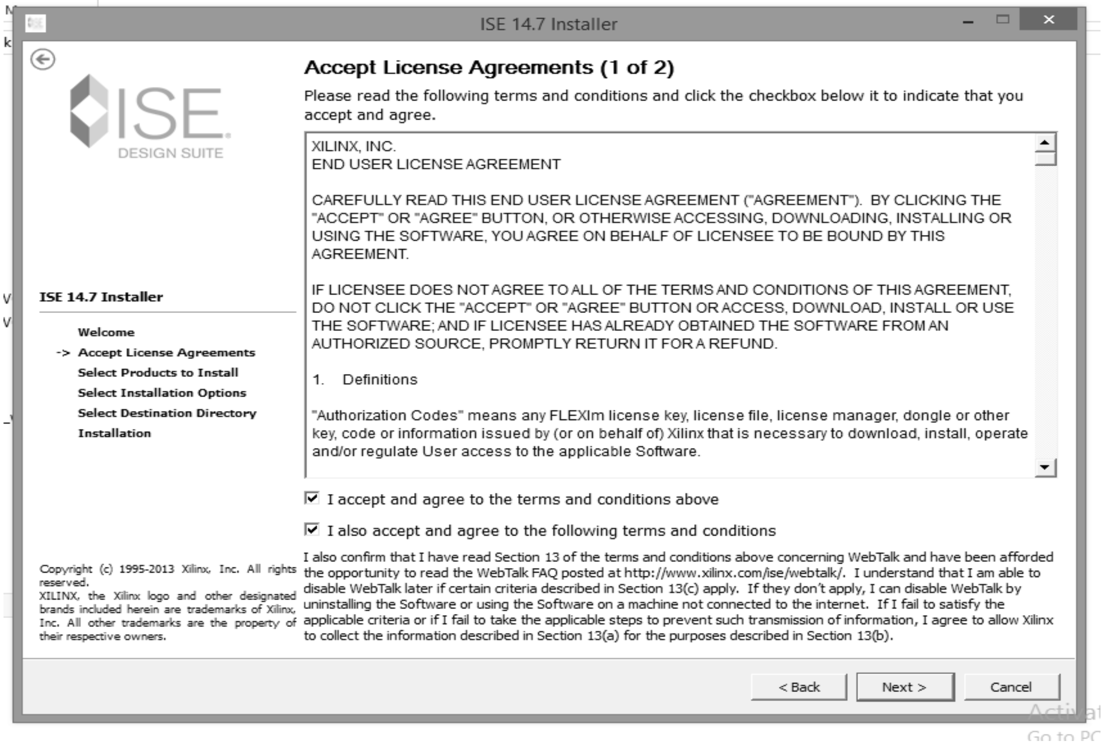
ISE 14.7 Installer

-> Welcome
Accept License Agreements
Select Products to Install
Select Installation Options
Select Destination Directory
Installation

Copyright (c) 1995-2013 Xilinx, Inc. All rights reserved.
XILINX, the Xilinx logo and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

Fig 3.1: Xilinx Installer Welcome Screen.

2. Click next on the Xilinx Agreement window.



ISE 14.7 Installer

Accept License Agreements (1 of 2)

Please read the following terms and conditions and click the checkbox below it to indicate that you accept and agree.

XILINX, INC.
END USER LICENSE AGREEMENT

CAREFULLY READ THIS END USER LICENSE AGREEMENT ("AGREEMENT"). BY CLICKING THE "ACCEPT" OR "AGREE" BUTTON, OR OTHERWISE ACCESSING, DOWNLOADING, INSTALLING OR USING THE SOFTWARE, YOU AGREE ON BEHALF OF LICENSEE TO BE BOUND BY THIS AGREEMENT.

IF LICENSEE DOES NOT AGREE TO ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT, DO NOT CLICK THE "ACCEPT" OR "AGREE" BUTTON OR ACCESS, DOWNLOAD, INSTALL OR USE THE SOFTWARE; AND IF LICENSEE HAS ALREADY OBTAINED THE SOFTWARE FROM AN AUTHORIZED SOURCE, PROMPTLY RETURN IT FOR A REFUND.

1. Definitions

"Authorization Codes" means any FLEXlm license key, license file, license manager, dongle or other key, code or information issued by (or on behalf of) Xilinx that is necessary to download, install, operate and/or regulate User access to the applicable Software.

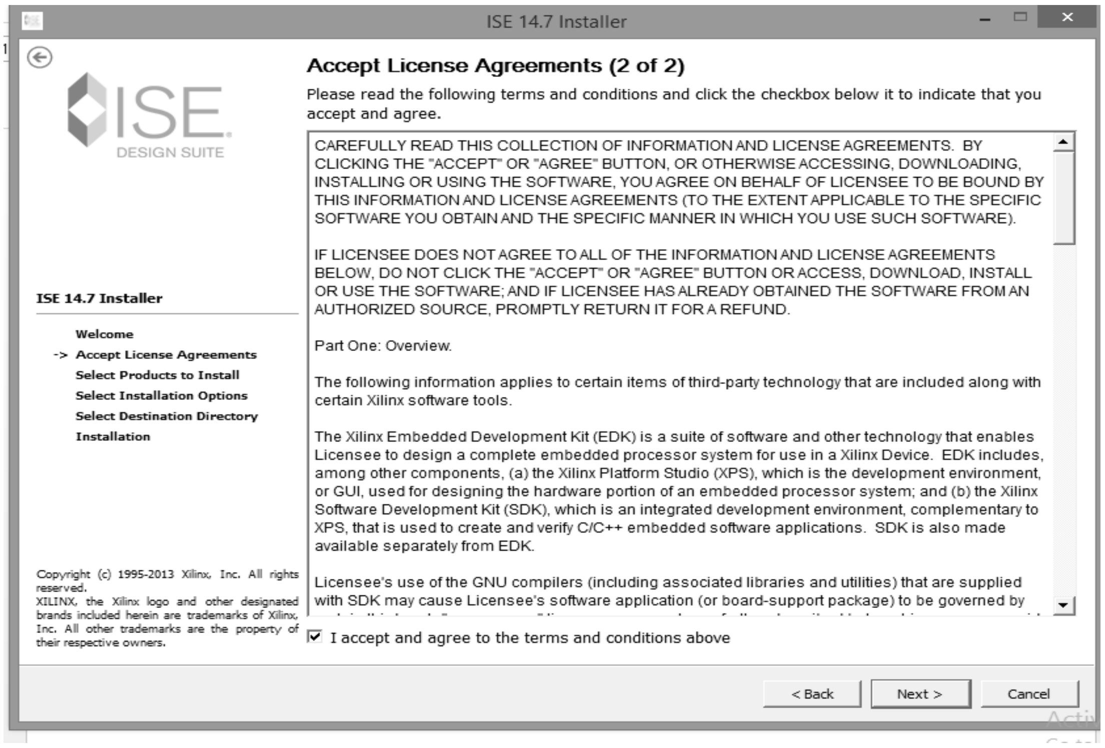
☑ I accept and agree to the terms and conditions above

☑ I also accept and agree to the following terms and conditions

I also confirm that I have read Section 13 of the terms and conditions above concerning WebTalk and have been afforded the opportunity to read the WebTalk FAQ posted at http://www.xilinx.com/ise/webtalk/. I understand that I am able to disable WebTalk later if certain criteria described in Section 13(c) apply. If they don't apply, I can disable WebTalk by uninstalling the Software or using the Software on a machine not connected to the internet. If I fail to satisfy the applicable criteria or if I fail to take the applicable steps to prevent such transmission of information, I agree to allow Xilinx to collect the information described in Section 13(a) for the purposes described in Section 13(b).

ISE DESIGN SUITE
ISE 14.7 Installer
Welcome
-> Accept License Agreements
Select Products to Install
Select Installation Options
Select Destination Directory
Installation
Copyright (c) 1995-2013 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.
< Back
Next >
Cancel

Fig 3.2 a: Xilinx License Agreement window



ISE 14.7 Installer

Accept License Agreements (2 of 2)

Please read the following terms and conditions and click the checkbox below it to indicate that you accept and agree.

CAREFULLY READ THIS COLLECTION OF INFORMATION AND LICENSE AGREEMENTS. BY CLICKING THE "ACCEPT" OR "AGREE" BUTTON, OR OTHERWISE ACCESSING, DOWNLOADING, INSTALLING OR USING THE SOFTWARE, YOU AGREE ON BEHALF OF LICENSEE TO BE BOUND BY THIS INFORMATION AND LICENSE AGREEMENTS (TO THE EXTENT APPLICABLE TO THE SPECIFIC SOFTWARE YOU OBTAIN AND THE SPECIFIC MANNER IN WHICH YOU USE SUCH SOFTWARE).

IF LICENSEE DOES NOT AGREE TO ALL OF THE INFORMATION AND LICENSE AGREEMENTS BELOW, DO NOT CLICK THE "ACCEPT" OR "AGREE" BUTTON OR ACCESS, DOWNLOAD, INSTALL OR USE THE SOFTWARE; AND IF LICENSEE HAS ALREADY OBTAINED THE SOFTWARE FROM AN AUTHORIZED SOURCE, PROMPTLY RETURN IT FOR A REFUND.

Part One: Overview.

The following information applies to certain items of third-party technology that are included along with certain Xilinx software tools.

The Xilinx Embedded Development Kit (EDK) is a suite of software and other technology that enables Licensee to design a complete embedded processor system for use in a Xilinx Device. EDK includes, among other components, (a) the Xilinx Platform Studio (XPS), which is the development environment, or GUI, used for designing the hardware portion of an embedded processor system; and (b) the Xilinx Software Development Kit (SDK), which is an integrated development environment, complementary to XPS, that is used to create and verify C/C++ embedded software applications. SDK is also made available separately from EDK.

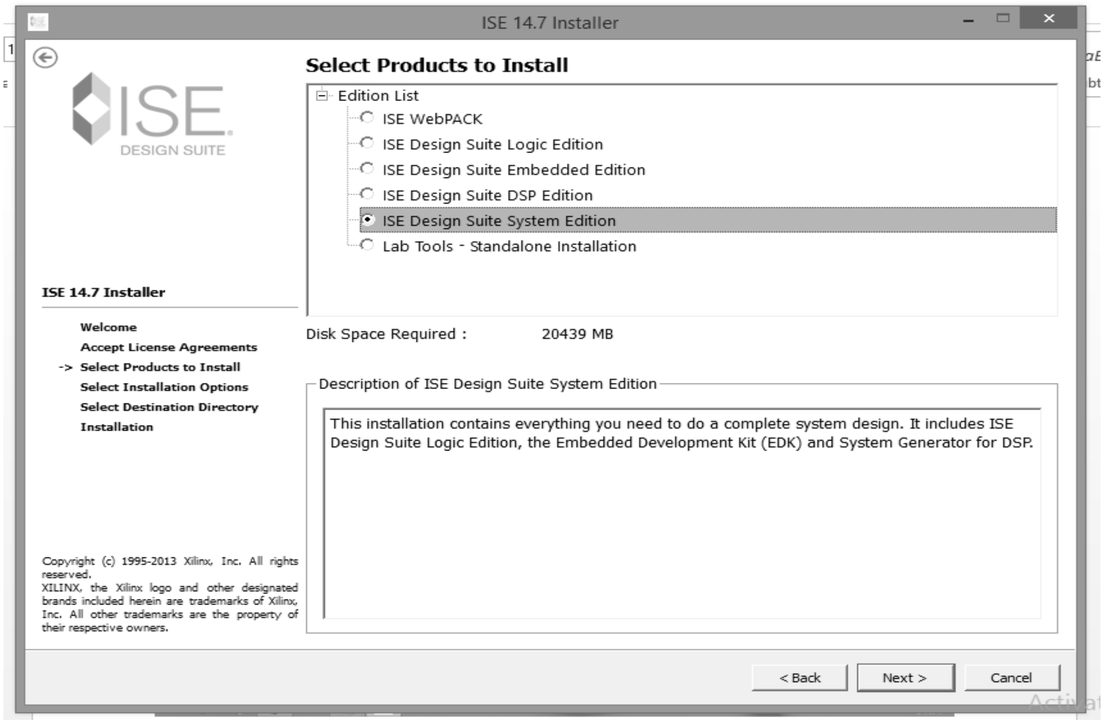
Licensee's use of the GNU compilers (including associated libraries and utilities) that are supplied with SDK may cause Licensee's software application (or board-support package) to be governed by

☑ I accept and agree to the terms and conditions above

ISE DESIGN SUITE
ISE 14.7 Installer
Welcome
-> Accept License Agreements
Select Products to Install
Select Installation Options
Select Destination Directory
Installation
Copyright (c) 1995-2013 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.
< Back
Next >
Cancel

Fig 3.2b: Xilinx License Agreement window

3. Select the products to install.



Select Products to Install

Edition List
- ISE WebPACK
- ISE Design Suite Logic Edition
- ISE Design Suite Embedded Edition
- ISE Design Suite DSP Edition
- [✓ ISE Design Suite System Edition]
- Lab Tools - Standalone Installation

Disk Space Required : 20439 MB

Description of ISE Design Suite System Edition

This installation contains everything you need to do a complete system design. It includes ISE Design Suite Logic Edition, the Embedded Development Kit (EDK) and System Generator for DSP.

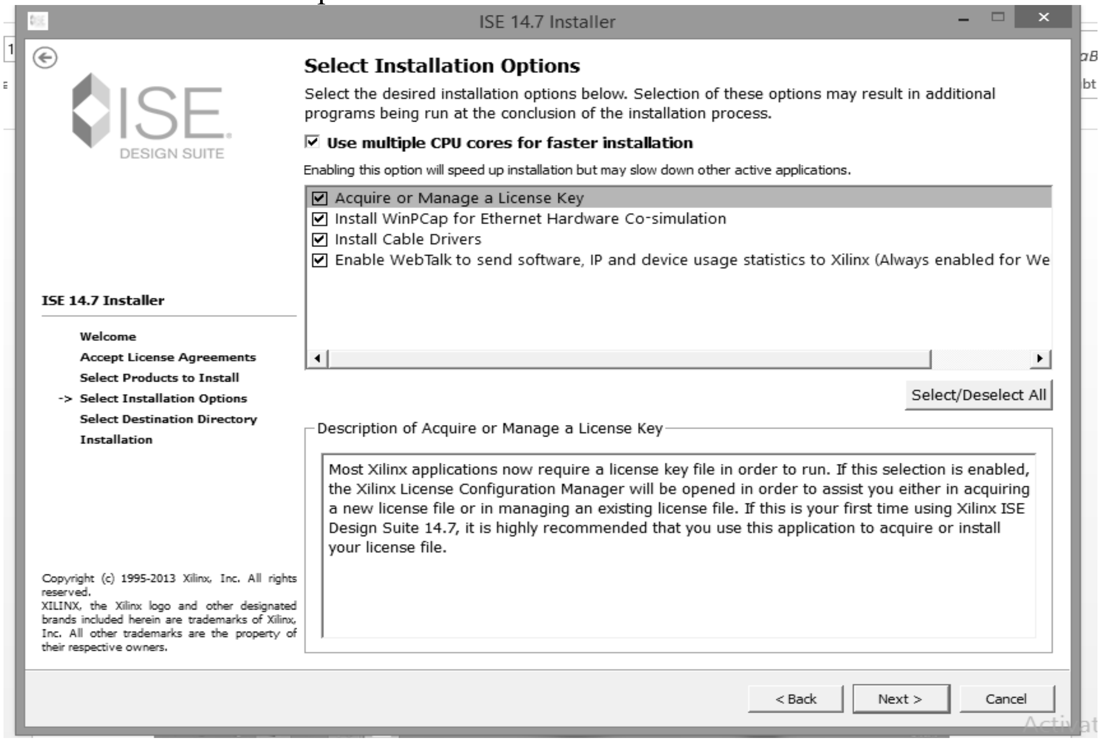
ISE 14.7 Installer

Welcome
Accept License Agreements
-> Select Products to Install
Select Installation Options
Select Destination Directory
Installation

Copyright (c) 1995-2013 Xilinx, Inc. All rights reserved.
XILINX, the Xilinx logo and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

Fig 3.3: Select products to install.

4. Select the Installations options.



Select Installation Options

Select the desired installation options below. Selection of these options may result in additional programs being run at the conclusion of the installation process.

☑ **Use multiple CPU cores for faster installation**

Enabling this option will speed up installation but may slow down other active applications.

☑ Acquire or Manage a License Key
☑ Install WinPCap for Ethernet Hardware Co-simulation
☑ Install Cable Drivers
☑ Enable WebTalk to send software, IP and device usage statistics to Xilinx (Always enabled for We

Description of Acquire or Manage a License Key

Most Xilinx applications now require a license key file in order to run. If this selection is enabled, the Xilinx License Configuration Manager will be opened in order to assist you either in acquiring a new license file or in managing an existing license file. If this is your first time using Xilinx ISE Design Suite 14.7, it is highly recommended that you use this application to acquire or install your license file.

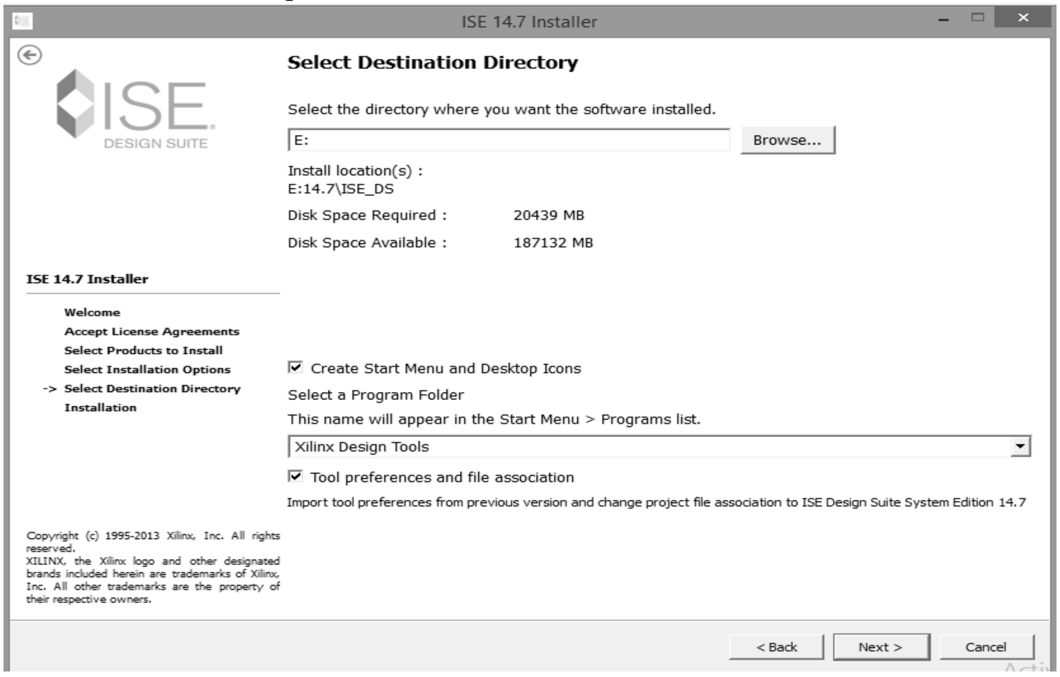
ISE 14.7 Installer

Welcome
Accept License Agreements
Select Products to Install
-> Select Installation Options
Select Destination Directory
Installation

Copyright (c) 1995-2013 Xilinx, Inc. All rights reserved.
XILINX, the Xilinx logo and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

Fig 3.4: Installations options

5. Select the destination path.


ISE 14.7 Installer

Select Destination Directory

Select the directory where you want the software installed.

E:

Install location(s) :
E:14.7\ISE_DS

Disk Space Required : 20439 MB

Disk Space Available : 187132 MB

ISE 14.7 Installer

- Welcome
- Accept License Agreements
- Select Products to Install
- Select Installation Options
- -> Select Destination Directory
- Installation

☑ Create Start Menu and Desktop Icons

Select a Program Folder

This name will appear in the Start Menu > Programs list.

Xilinx Design Tools

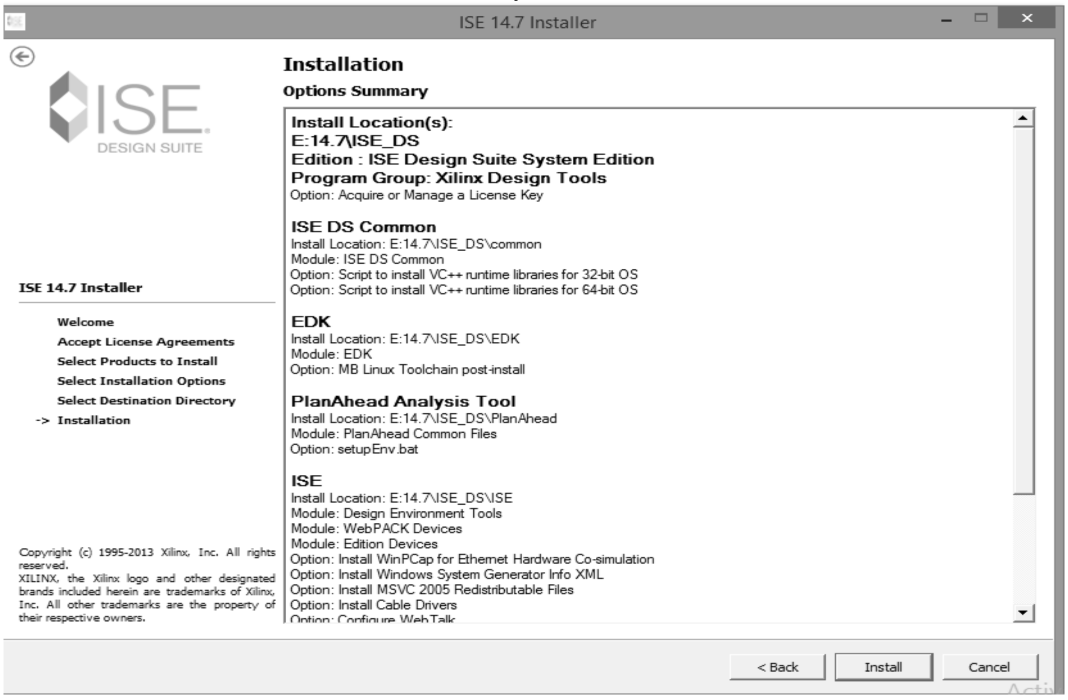
☑ Tool preferences and file association

Import tool preferences from previous version and change project file association to ISE Design Suite System Edition 14.7

Copyright (c) 1995-2013 Xilinx, Inc. All rights reserved.
XILINX, the Xilinx logo and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

Fig 3.5: Select Destination path

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


ISE 14.7 Installer

Installation

Options Summary

Install Location(s):
E:14.7\ISE_DS
Edition : ISE Design Suite System Edition
Program Group: Xilinx Design Tools
Option: Acquire or Manage a License Key

ISE DS Common
Install Location: E:14.7\ISE_DS\common
Module: ISE DS Common
Option: Script to install VC++ runtime libraries for 32-bit OS
Option: Script to install VC++ runtime libraries for 64-bit OS

EDK
Install Location: E:14.7\ISE_DS\EDK
Module: EDK
Option: MB Linux Toolchain post-install

PlanAhead Analysis Tool
Install Location: E:14.7\ISE_DS\PlanAhead
Module: PlanAhead Common Files
Option: setupEnv.bat

ISE
Install Location: E:14.7\ISE_DS\ISE
Module: Design Environment Tools
Module: WebPACK Devices
Module: Edition Devices
Option: Install WinPCap for Ethernet Hardware Co-simulation
Option: Install Windows System Generator Info XML
Option: Install MSVC 2005 Redistributable Files
Option: Install Cable Drivers
Option: Configure WebTalk

ISE 14.7 Installer

- Welcome
- Accept License Agreements
- Select Products to Install
- Select Installation Options
- Select Destination Directory
- -> Installation

Copyright (c) 1995-2013 Xilinx, Inc. All rights reserved.
XILINX, the Xilinx logo and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

Fig 3.6: Installation Summary options.

7. Installation of the software continues.

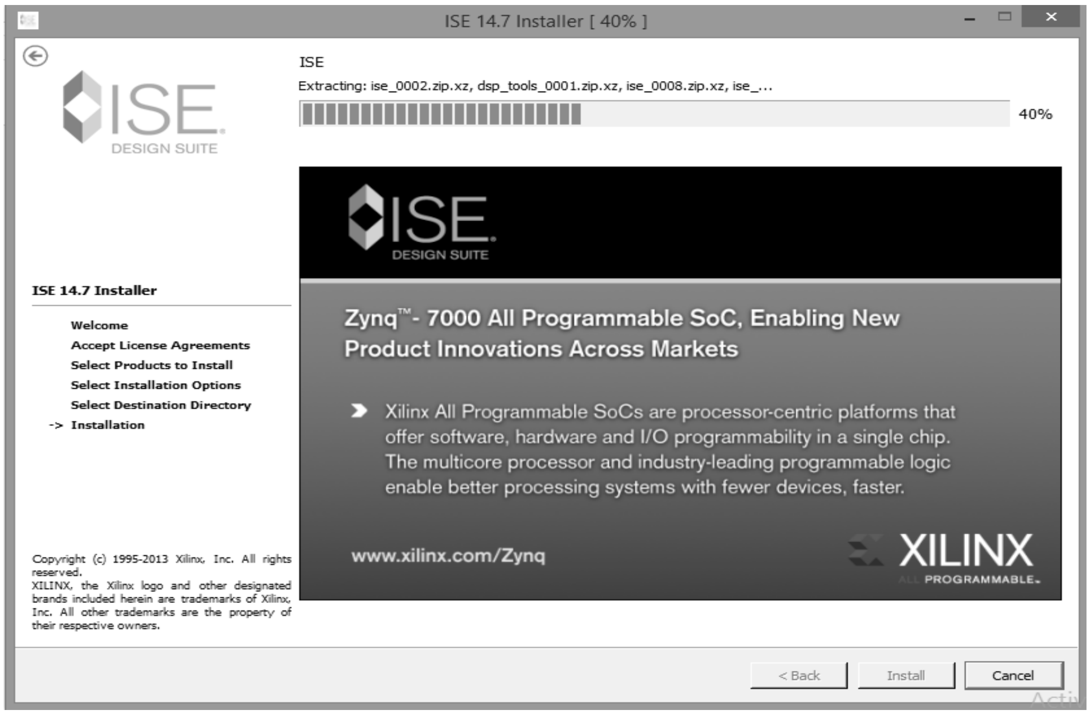
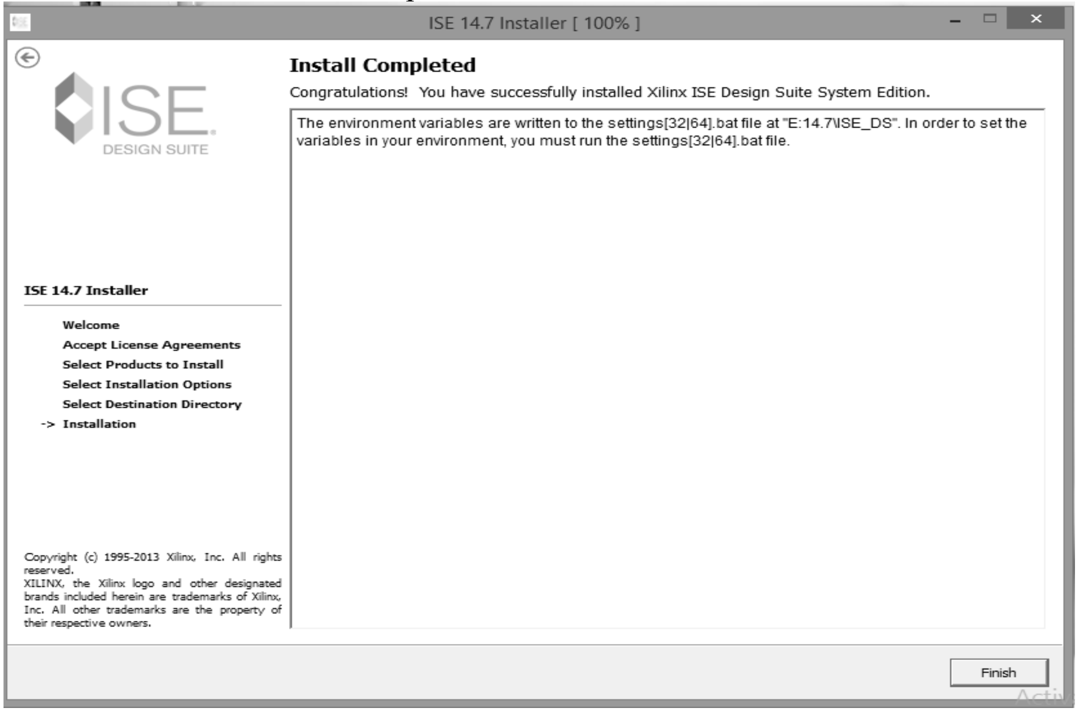

ISE 14.7 Installer [40%]
ISE DESIGN SUITE
ISE
Extracting: ise_0002.zip.xz, dsp_tools_0001.zip.xz, ise_0008.zip.xz, ise_...
40%
ISE 14.7 Installer
Welcome
Accept License Agreements
Select Products to Install
Select Installation Options
Select Destination Directory
-> Installation
Zynq™- 7000 All Programmable SoC, Enabling New Product Innovations Across Markets
Xilinx All Programmable SoCs are processor-centric platforms that offer software, hardware and I/O programmability in a single chip. The multicore processor and industry-leading programmable logic enable better processing systems with fewer devices, faster.
www.xilinx.com/Zynq
XILINX ALL PROGRAMMABLE.
Copyright (c) 1995-2013 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.
< Back
Install
Cancel

Fig 3.7: Installation Window

8. Click Finish on the Install Complete window.


ISE 14.7 Installer [100%]
ISE DESIGN SUITE
ISE 14.7 Installer
Welcome
Accept License Agreements
Select Products to Install
Select Installation Options
Select Destination Directory
-> Installation
Copyright (c) 1995-2013 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.
Finish

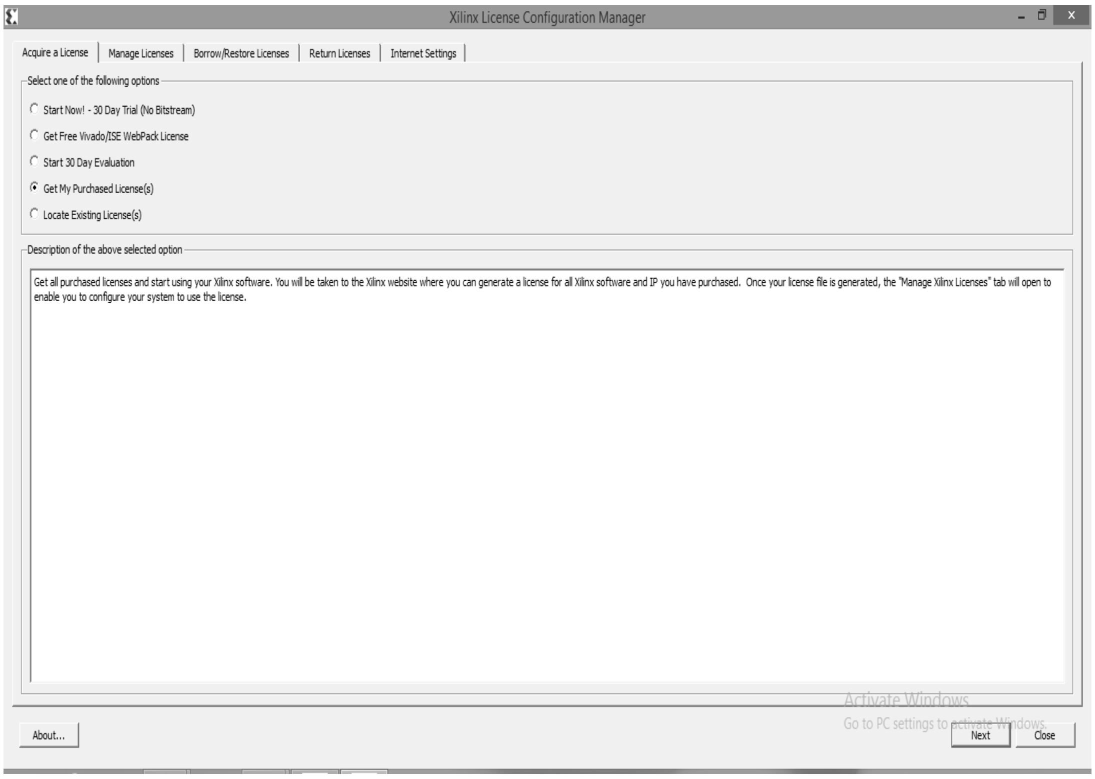
Install Completed

Congratulations! You have successfully installed Xilinx ISE Design Suite System Edition.

The environment variables are written to the settings[32|64].bat file at "E:14.7\ISE_DS". In order to set the variables in your environment, you must run the settings[32|64].bat file.

Fig 3.8: Install complete window

9. Click Next on the License Configuration window.



Select one of the following options

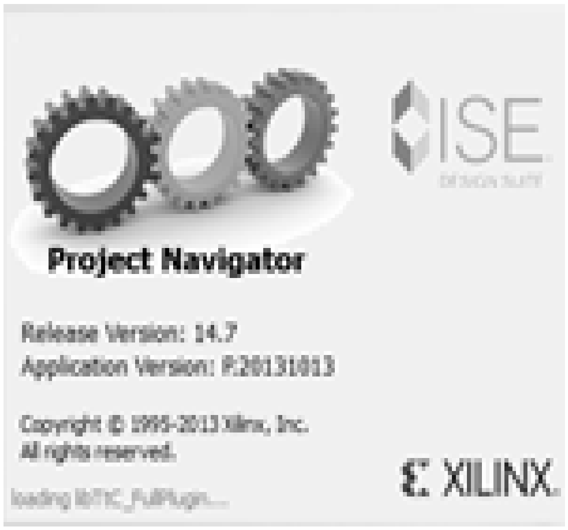
- ○ Start Now! - 30 Day Trial (No Bitstream)
- ○ Get Free Vivado/ISE WebPack License
- ○ Start 30 Day Evaluation
- ● Get My Purchased License(s)
- ○ Locate Existing License(s)

Description of the above selected option

Get all purchased licenses and start using your Xilinx software. You will be taken to the Xilinx website where you can generate a license for all Xilinx software and IP you have purchased. Once your license file is generated, the "Manage Xilinx Licenses" tab will open to enable you to configure your system to use the license.

Fig 3.9: 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.



Release Version: 14.7
Application Version: P.20131013

Copyright © 1995-2013 Xilinx, Inc.
All rights reserved.

loading libTIC_FullPlugin....

Fig 3.10: Xilinx Initialize Window

XI Resources Used:

Sr. No.	Instrument /Components/ EDA tool	Specification	Quantity
1			

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

....
....
....
....
....

XIII Actual Procedure followed:

....
....
....
....
....

XIV Conclusion and Recommendation

....
....
....
....
....

XV 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. List features of Xilinx software.
2. Define the terms simulation, synthesis..

[Space for Answers]

....
....
....
....
....
....
....

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

XVI References / Suggestions for further reading

1 https://en.wikipedia.org/wiki/Xilinx_ISE
2 https://www.xilinx.com/support/documentation/sw_manuals/xilinx10/isehelp/ise_c_overview.htm
3 https://www.xilinx.com/html_docs/xilinx2019_1/sdsoc_doc/introduction-mwq1504034373484.html
4 http://allpcworld.com/xilinx-ise-design-suite-14-7-free-download/

XVII Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**60%**
1	Identifying the given IDE	30%
2	Identifying the features of IDE	20%
3	Follow ethical practices.	10%
Product related (10 Marks)		**40%**
4	Correct procedure followed	20%
5	Answer to sample questions.	15%
6	Timely Submission of report Answer to sample questions.	05%
	Total (25 Marks)	**100%**

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 04: Implement any two logic gates using Data flow and Behavior Model.

I Practical Significance

VHDL stands for very high-speed integrated circuit hardware description language which is one of the programming language used to model/ describe the hardware of a digital system by dataflow, behavioral and structural style of modeling. This practical will help the students to model the logic gates using different modelling techniques.

II Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronic circuits and equipments."**

- Use relevant VHDL model for the given applications.
- Debug VHDL programme for the given application.

IV Relevant Course Outcome

Debug VHDL programme for the given application..

V Practical Outcome

Implement any two logic gates using Data flow and Behavior Model.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum Theoretical Background

VHDL is used for describing hardware. There are four styles of modelling/ description using VHDL.

- Dataflow Description: A data flow description describes the transfer of data from input to output and between signals. The view of data as flowing through a design, from input to output. An operation is defined in terms of collection of data transformations, expressed as concurrent statements.
- Behavioral Description: A Behavioral modelling describes the design in terms of circuit or system behavior using algorithms. This high level language uses language constructs that resemble a high level software programming language.

- Structural Description: A structural description describes the circuit structure in terms of the logic gates used and the interconnect wiring between the logic gates to form a circuit netlist. This view is expressed by component instantiations.
- Mixed Style of Modelling: Mixed style of modelling is combination of different modelling styles to describe an entire hardware.

VHDL Flow Elements: The VHDL provides five different types of primary constructs called design units:

- Entity
- Architecture
- Configuration
- Package
- Package body

Entity declaration: It defines the names, input output signals and modes of a hardware module.

Syntax:

entity entity_name **is**
    Port declaration;
  end entity_name;

An entity declaration should starts with 'entity' and ends with 'end' keywords.
Ports are interfaces through which an entity can communicate with its environment. Each port must have a name, direction and a type. An entity may have no port declaration also. The direction will be input, output or inout.

In	Port can be read
Out	Port can be written
Inout	Port can be read and written
Buffer	Port can be read and written, it can have only one source.

Architecture: It describes the internal description of design or it tells what is there inside design. Each entity has at least one architecture and an entity can have many architecture. Architecture can be described using structural, dataflow, behavioral or mixed style. Architecture can be used to describe a design at different levels of abstraction like gate level, register transfer level (RTL) or behavior level.

Syntax:

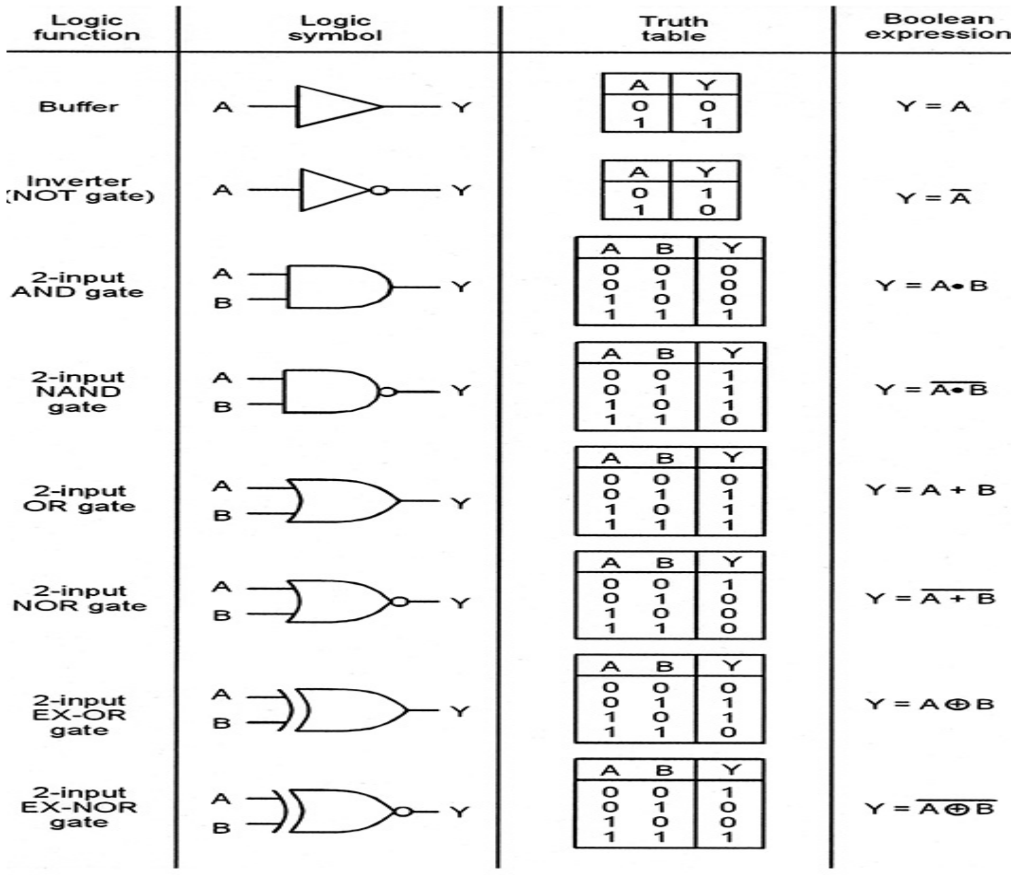
architecture architecture_name **of** entity_name
architecture_declarative_part;
begin
  Statements;
end architecture_name;

Here we should specify the entity name for which we are writing the architecture body.

The architecture statements should be inside the begin and end keyword.

Architecture declarative part may contain variables, constants, or component declaration.

Logic Gate symbols and truth table:



Logic function	Logic symbol	Truth table	Boolean expression
Buffer	A Y	A Y 0 0 1 1	$Y = A$
Inverter (NOT gate)	A Y	A Y 0 1 1 0	$Y = \overline{A}$
2-input AND gate	A B Y	A B Y 0 0 0 0 1 0 1 0 0 1 1 1	$Y = A \bullet B$
2-input NAND gate	A B Y	A B Y 0 0 1 0 1 1 1 0 1 1 1 0	$Y = \overline{A \bullet B}$
2-input OR gate	A B Y	A B Y 0 0 0 0 1 1 1 0 1 1 1 1	$Y = A + B$
2-input NOR gate	A B Y	A B Y 0 0 1 0 1 0 1 0 0 1 1 0	$Y = \overline{A + B}$
2-input EX-OR gate	A B Y	A B Y 0 0 0 0 1 1 1 0 1 1 1 0	$Y = A \oplus B$
2-input EX-NOR gate	A B Y	A B Y 0 0 1 0 1 0 1 0 0 1 1 1	$Y = \overline{A \oplus B}$

Fig 4.1: Logic Gates Symbols, Truth Table, Boolean Equation

VIII Resources Required:

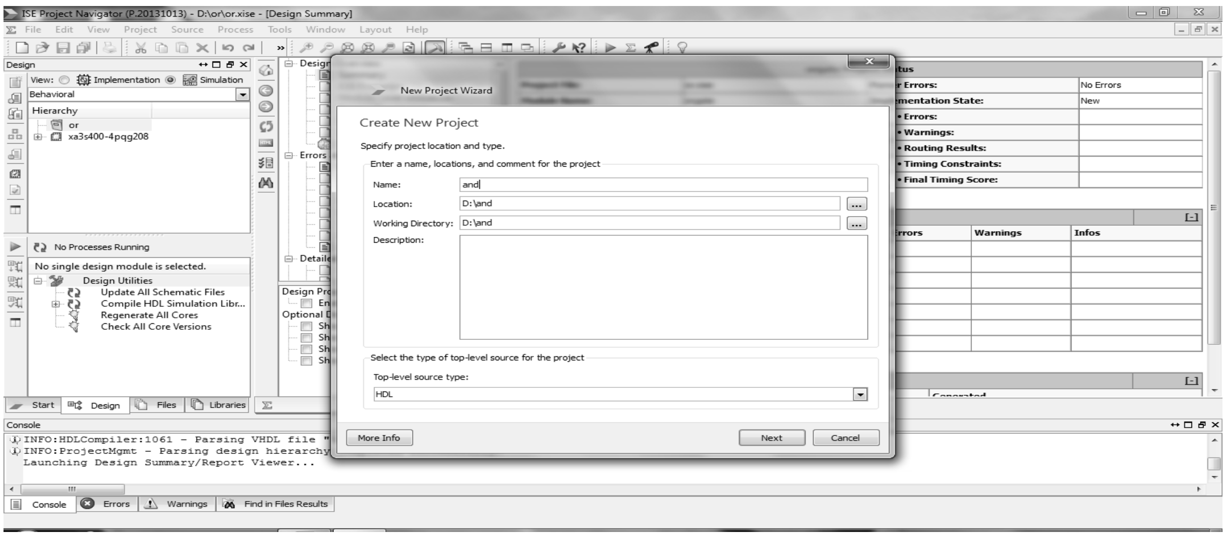
Sr. No.	Instrument / Components	Specification	Quantity
1	Desktop PC	Loaded with open source EDA tool (VHDL)	1 No.

IX Precautions to be followed:

Check the syntax / rules of VHDL Programming.

X Procedure:

1. Create the Xilinx ISE project for your top-level FPGA design, by doing the following in ISE:
   In the ISE software, select File > New Project.(refer fig. 4.2)
   In the Project name and Project location fields, enter the project name and location, respectively. (refer fig. 4.2)
   Select HDL or Schematic as the Top-level source type, and click Next. (fig. 4.5)


ISE Project Navigator (P.20131013) - D:\or\or.xise - [Design Summary]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
or
xa3s400-4pqg208
No Processes Running
No single design module is selected.
Design Utilities
Update All Schematic Files
Compile HDL Simulation Libr...
Regenerate All Cores
Check All Core Versions
Start Design Files Libraries
New Project Wizard
Console Errors Warnings Find in Files Results

Create New Project

Specify project location and type.

Enter a name, locations, and comment for the project

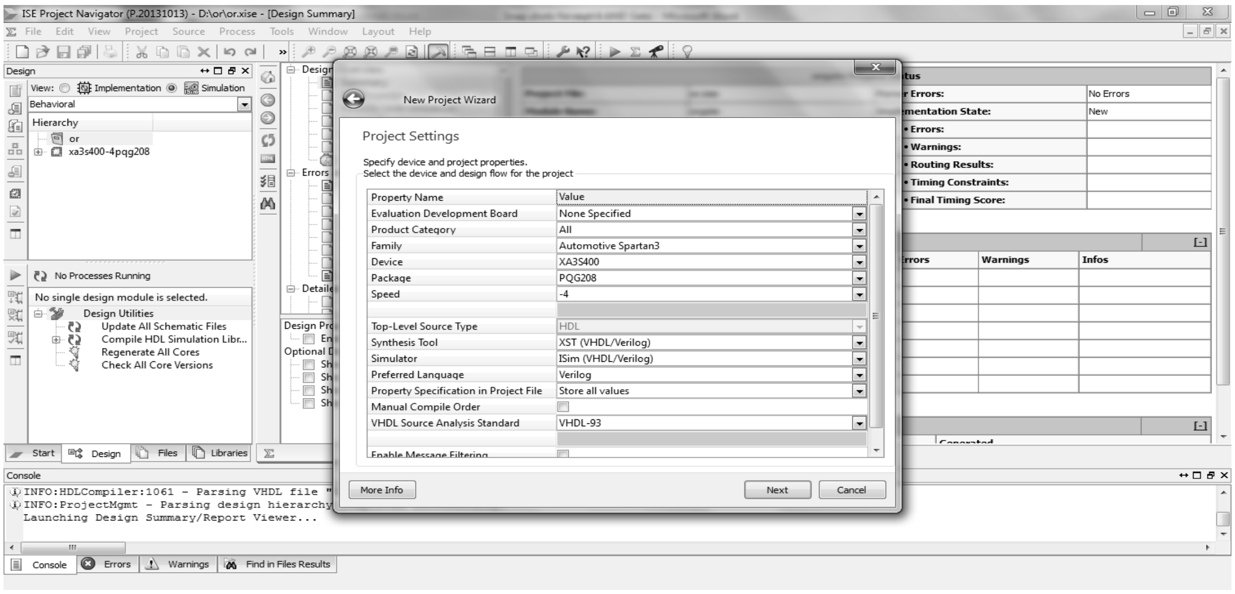
Name:	and
Location:	D:\and
Working Directory:	D:\and
Description:	

Select the type of top-level source for the project

Top-level source type: HDL

```
INFO:HDLCompiler:1061 - Parsing VHDL file "
INFO:ProjectMgmt - Parsing design hierarchy
Launching Design Summary/Report Viewer...
```

Fig 4.2: Create new project


ISE Project Navigator (P.20131013) - D:\or\or.xise - [Design Summary]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
or
xa3s400-4pqg208
No Processes Running
No single design module is selected.
Design Utilities
Update All Schematic Files
Compile HDL Simulation Libr...
Regenerate All Cores
Check All Core Versions
Start Design Files Libraries
New Project Wizard
More Info
Next
Cancel
Console Errors Warnings Find in Files Results

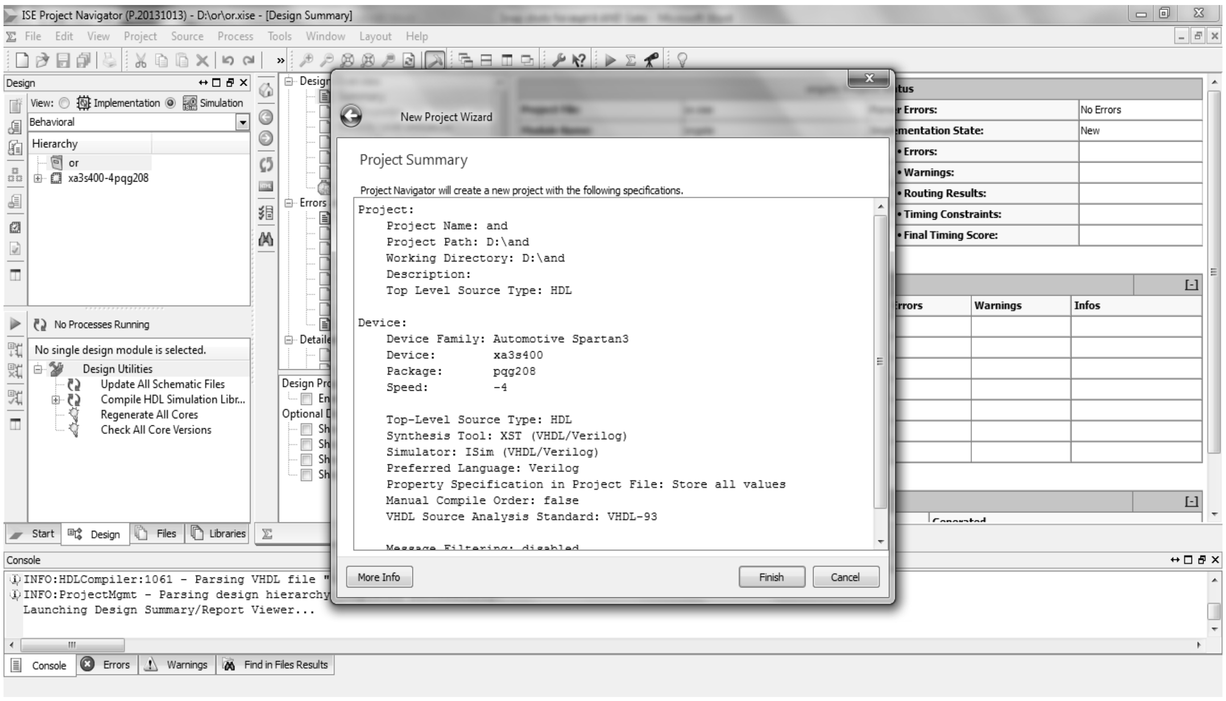
Project Settings

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

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Automotive Spartan3
Device	XA3S400
Package	PQG208
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

```
INFO:HDLCompiler:1061 - Parsing VHDL file "
INFO:ProjectMgmt - Parsing design hierarchy
Launching Design Summary/Report Viewer...
```

Fig 4.3: Project File Settings



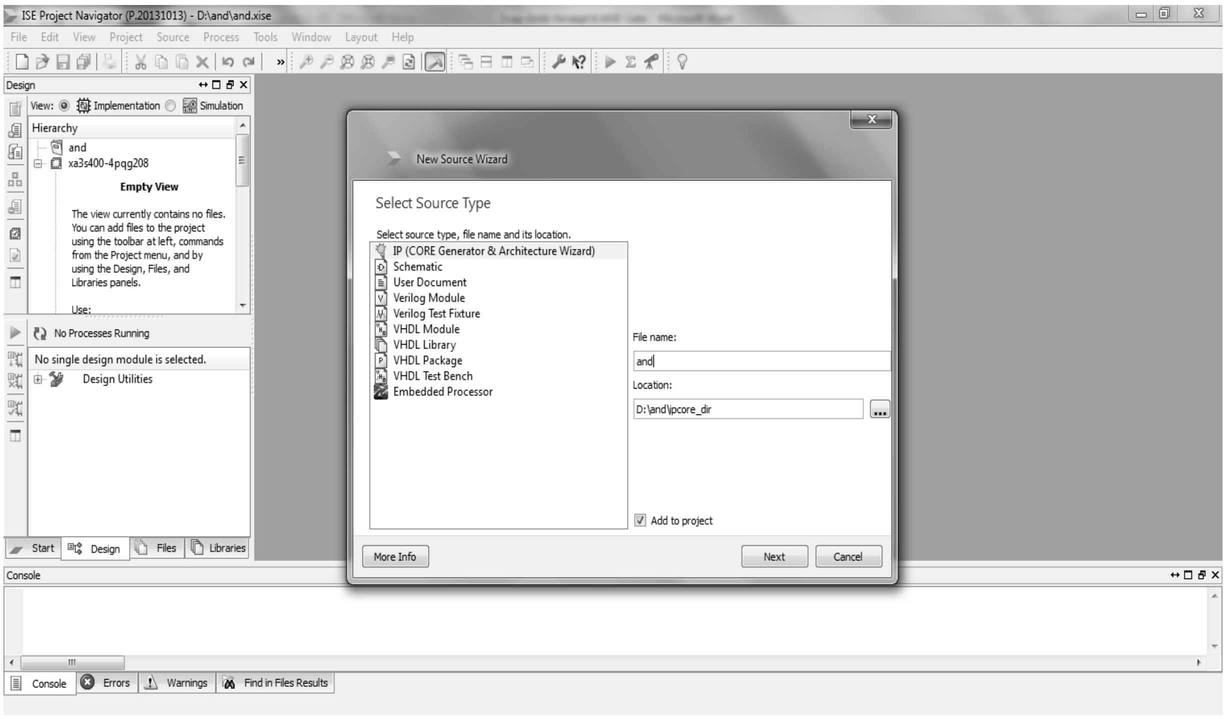
Project Navigator will create a new project with the following specifications.

```
Project:
    Project Name: and
    Project Path: D:\and
    Working Directory: D:\and
    Description:
    Top Level Source Type: HDL

Device:
    Device Family: Automotive Spartan3
    Device:        xa3s400
    Package:       pqg208
    Speed:         -4

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

    Message Filtering: disabled
```

```
INFO:HDLCompiler:1061 - Parsing VHDL file "
INFO:ProjectMgmt - Parsing design hierarchy
Launching Design Summary/Report Viewer...
```

Fig 4.4: Project Summary

2. Create New Source file. Select Source Type" select the Source type and give the name to the source then click "Next". .(refer fig.. 4.5)



Select source type, file name and its location.

- IP (CORE Generator & Architecture Wizard)
- Schematic
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench
- Embedded Processor

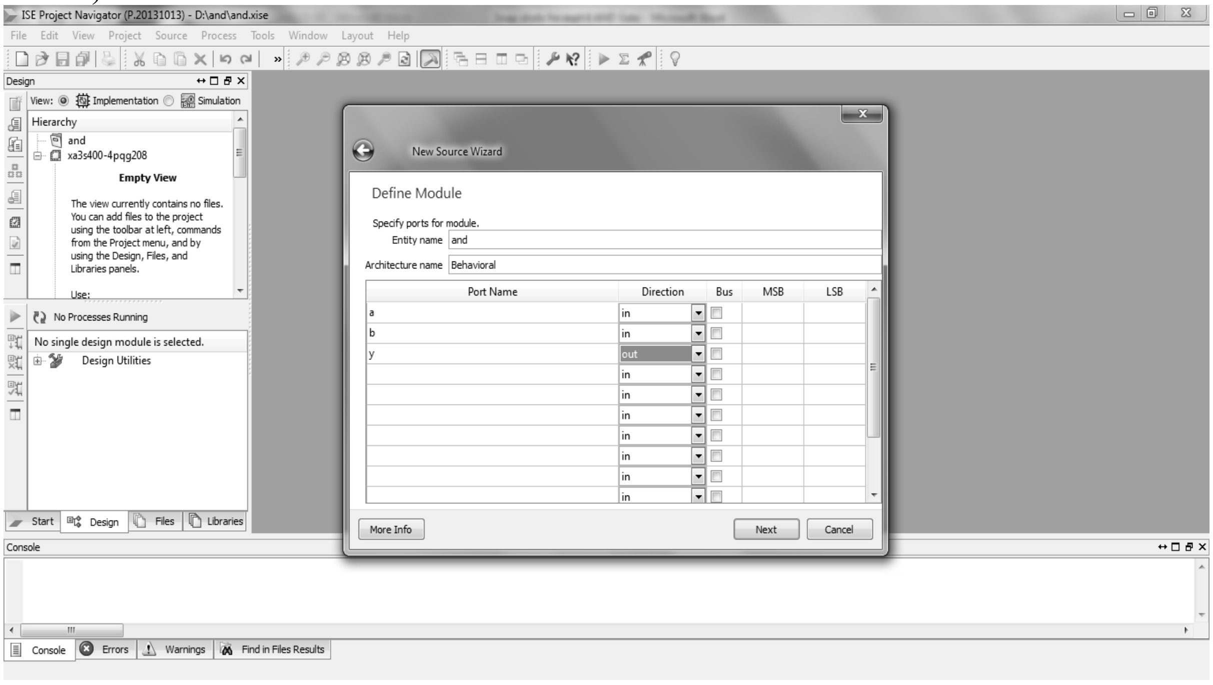
File name: and

Location: D:\and\ipcore_dir

☑ Add to project

Fig 4.5: Selecting Source Type

3. Define Module: Enter the entity used in design and then click “Next”. (refer fig.. 4.6 and 4.7)


ISE Project Navigator (P.20131013) - D:\and\and.xise
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
and
xa3s400-4pqg208
Empty View
The view currently contains no files. You can add files to the project using the toolbar at left, commands from the Project menu, and by using the Design, Files, and Libraries panels.
Use:
No Processes Running
No single design module is selected.
Design Utilities
Start Design Files Libraries
Console
Console Errors Warnings Find in Files Results
New Source Wizard
More Info
Next
Cancel

Define Module

Specify ports for module.

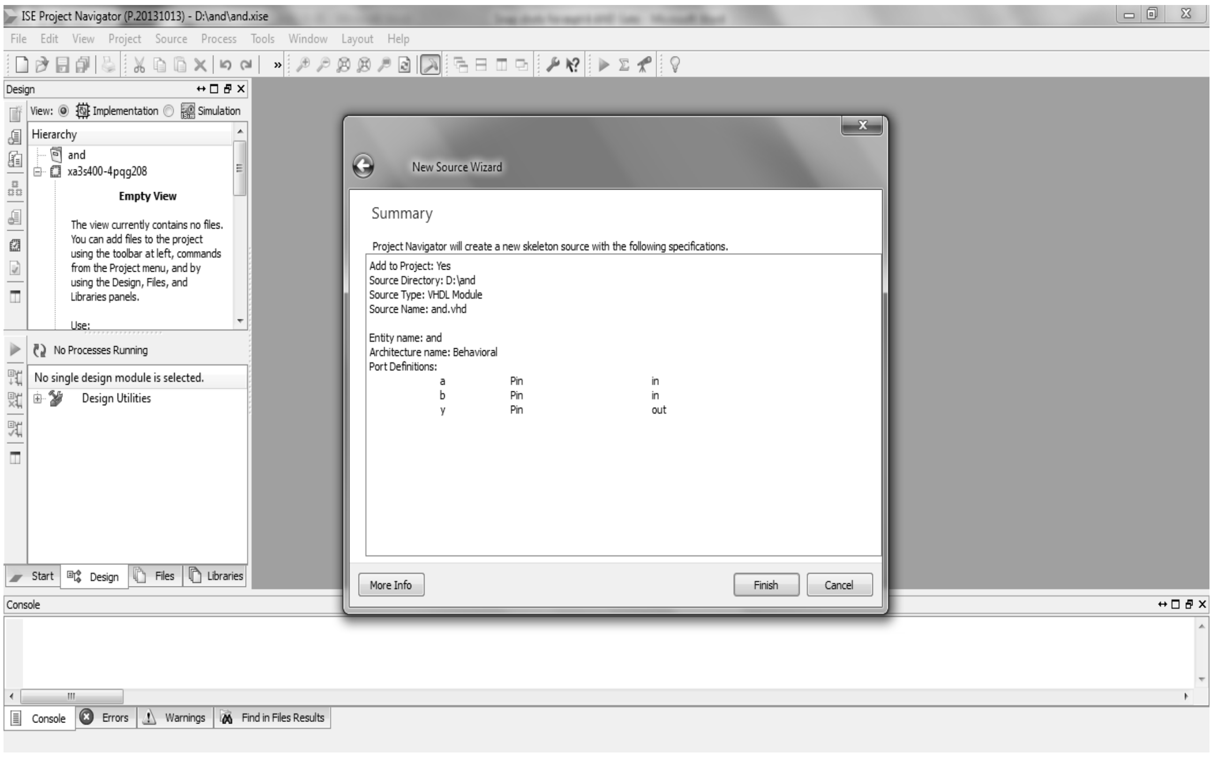
Entity name: and

Architecture name: Behavioral

Port Name	Direction	Bus	MSB	LSB
a	in			
b	in			
y	out			
	in			
	in			
	in			
	in			
	in			
	in			
	in			

,

Fig 4.6: Define Entity


ISE Project Navigator (P.20131013) - D:\and\and.xise
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
and
xa3s400-4pqg208
Empty View
The view currently contains no files. You can add files to the project using the toolbar at left, commands from the Project menu, and by using the Design, Files, and Libraries panels.
Use:
No Processes Running
No single design module is selected.
Design Utilities
Start Design Files Libraries
Console
Console Errors Warnings Find in Files Results
New Source Wizard
More Info
Finish
Cancel

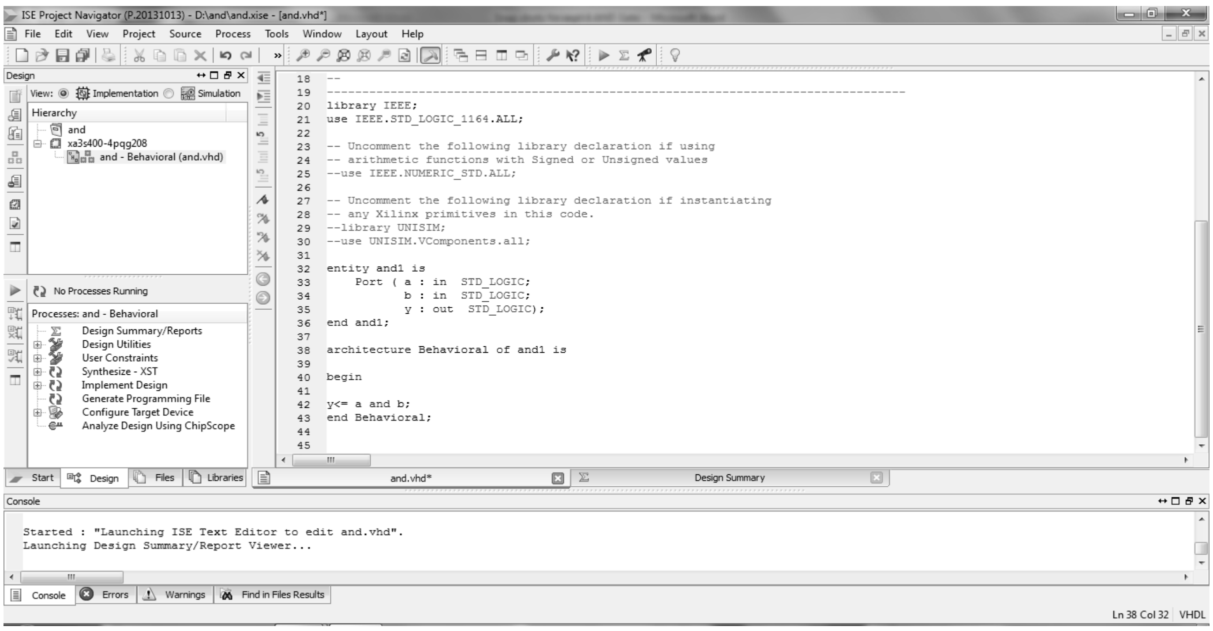
Summary

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

```
Add to Project: Yes
Source Directory: D:\and
Source Type: VHDL Module
Source Name: and.vhd

Entity name: and
Architecture name: Behavioral
Port Definitions:
        a        Pin        in
        b        Pin        in
        y        Pin        out
```

Fig4.7 Entity Summary

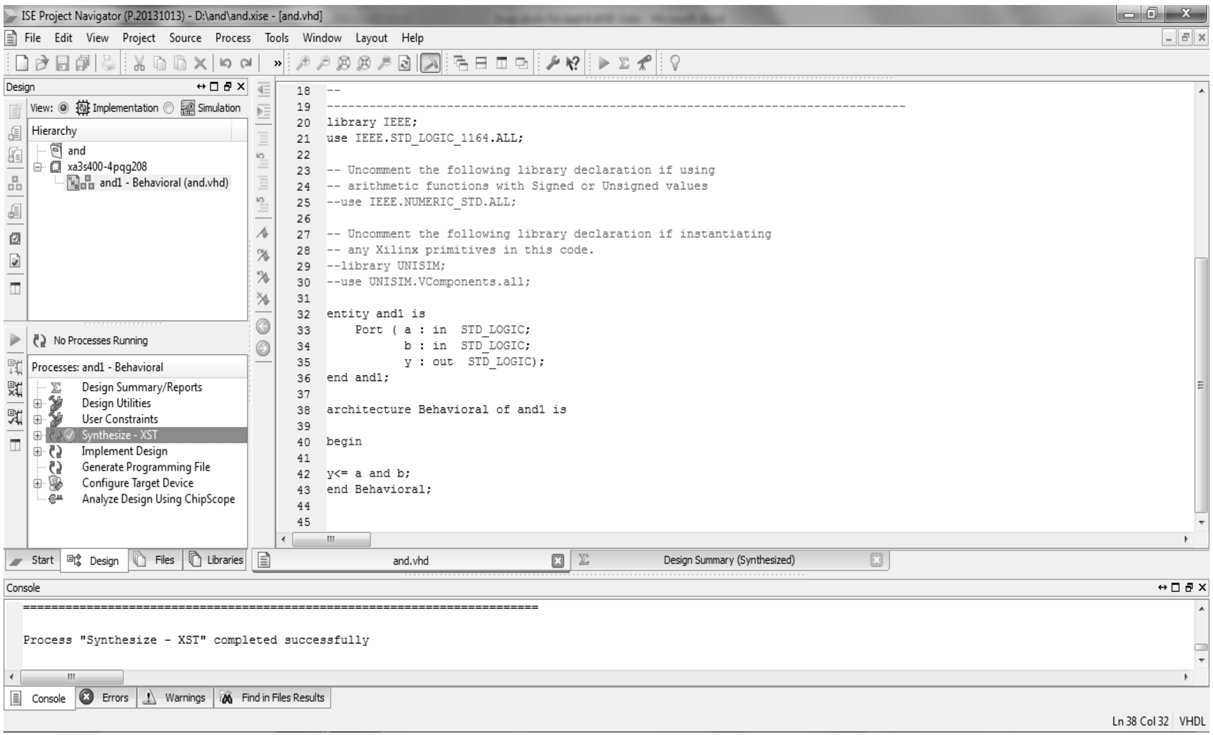
4. Develop VHDL code for given problem and Synthesize the .vhd file and view RTL schematic. (refer fig.. 4.8 and 4.9)


ISE Project Navigator (P.20131013) - D:\and\and.xise - [and.vhd*]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
and
xa3s400-4pqg208
and - Behavioral (and.vhd)
No Processes Running
Processes: and - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
and.vhd*
Design Summary
Console
Console Errors Warnings Find in Files Results
Ln 38 Col 32 VHDL

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

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

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

entity and1 is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           y : out  STD_LOGIC);
end and1;

architecture Behavioral of and1 is

begin

y<= a and b;
end Behavioral;
```

```
Started : "Launching ISE Text Editor to edit and.vhd".
Launching Design Summary/Report Viewer...
```

Fig 4.8: VHDL Coding

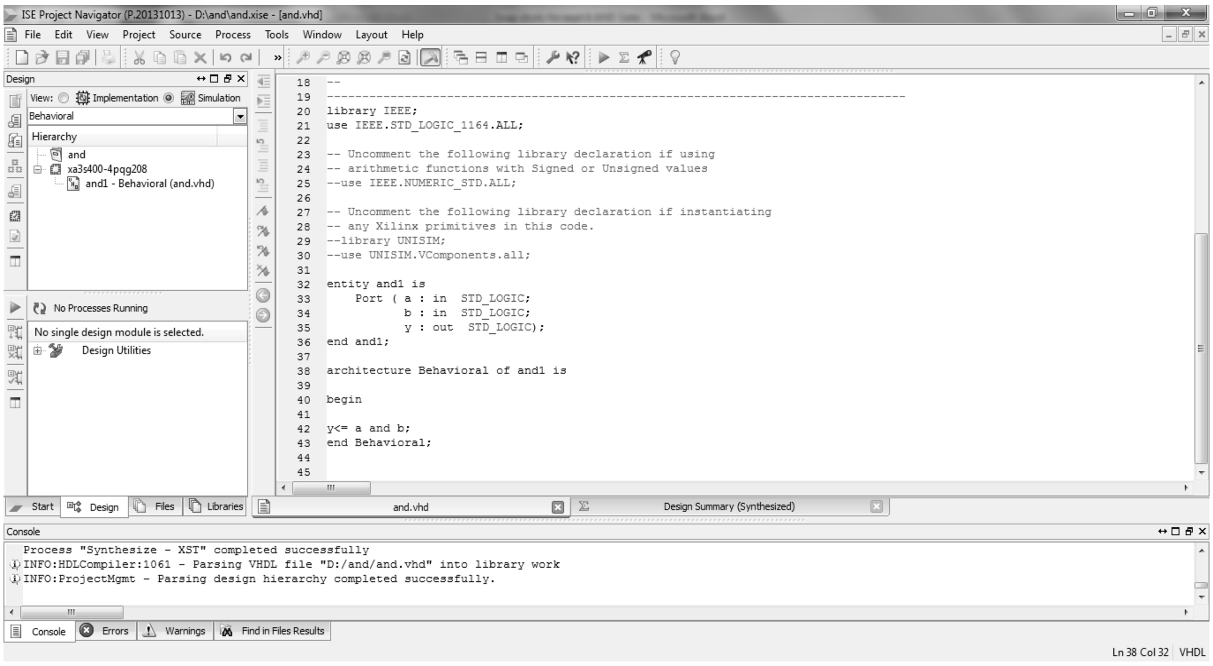

ISE Project Navigator (P.20131013) - D:\and\and.xise - [and.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
and
xa3s400-4pqg208
and1 - Behavioral (and.vhd)
No Processes Running
Processes: and1 - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
and.vhd
Design Summary (Synthesized)
Console
Console Errors Warnings Find in Files Results
Ln 38 Col 32 VHDL

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

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

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

entity and1 is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           y : out  STD_LOGIC);
end and1;

architecture Behavioral of and1 is

begin

y<= a and b;
end Behavioral;
```

```
=========================================================================

Process "Synthesize - XST" completed successfully
```

Fig 4.9: Synthesize the .vhd file

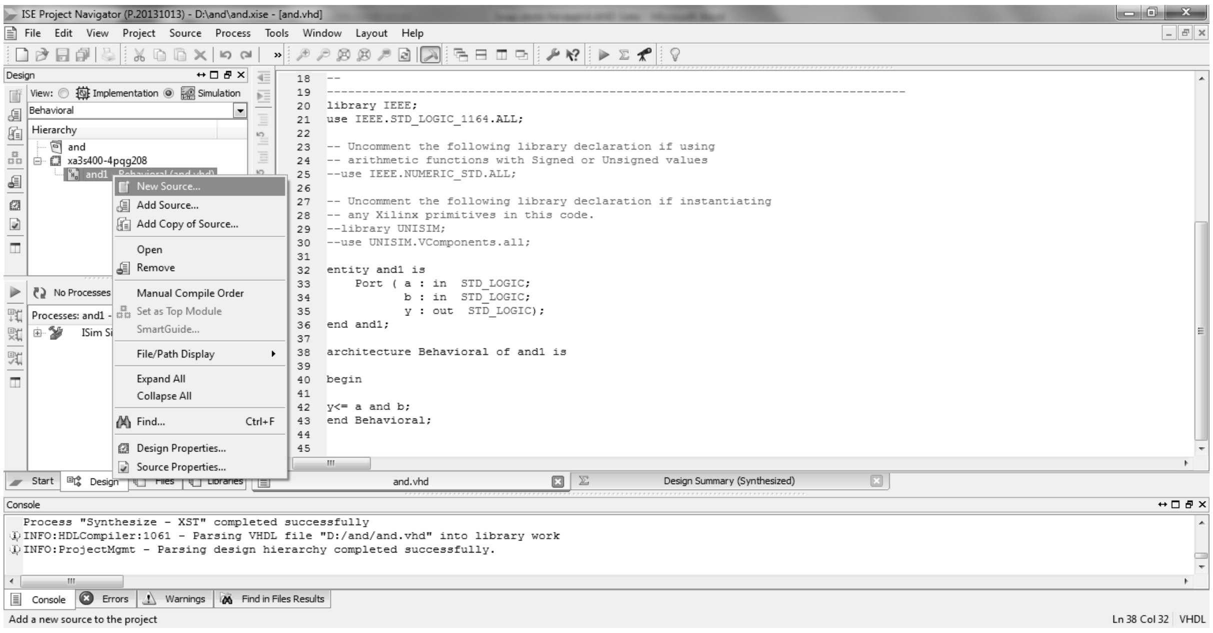
5. Create Test Bench file- A **test bench** is **HDL** code that allows you to provide a documented, repeatable set of stimuli that is portable across different simulators. A **test bench** can be as simple as a file with clock and input data or a more complicated file that includes error checking, file 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 extend with abc_tb. (refer fig..4.10-12)


ISE Project Navigator (P.20131013) - D:\and\and.xise - [and.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
and
xa3s400-4pqg208
and1 - Behavioral (and.vhd)
No Processes Running
No single design module is selected.
Design Utilities
Start Design Files Libraries
and.vhd
Design Summary (Synthesized)
Console Errors Warnings Find in Files Results
Ln 38 Col 32 VHDL

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

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

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

entity and1 is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           y : out  STD_LOGIC);
end and1;

architecture Behavioral of and1 is

begin

y<= a and b;
end Behavioral;
```

```
Process "Synthesize - XST" completed successfully
INFO:HDLCompiler:1061 - Parsing VHDL file "D:/and/and.vhd" into library work
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
```

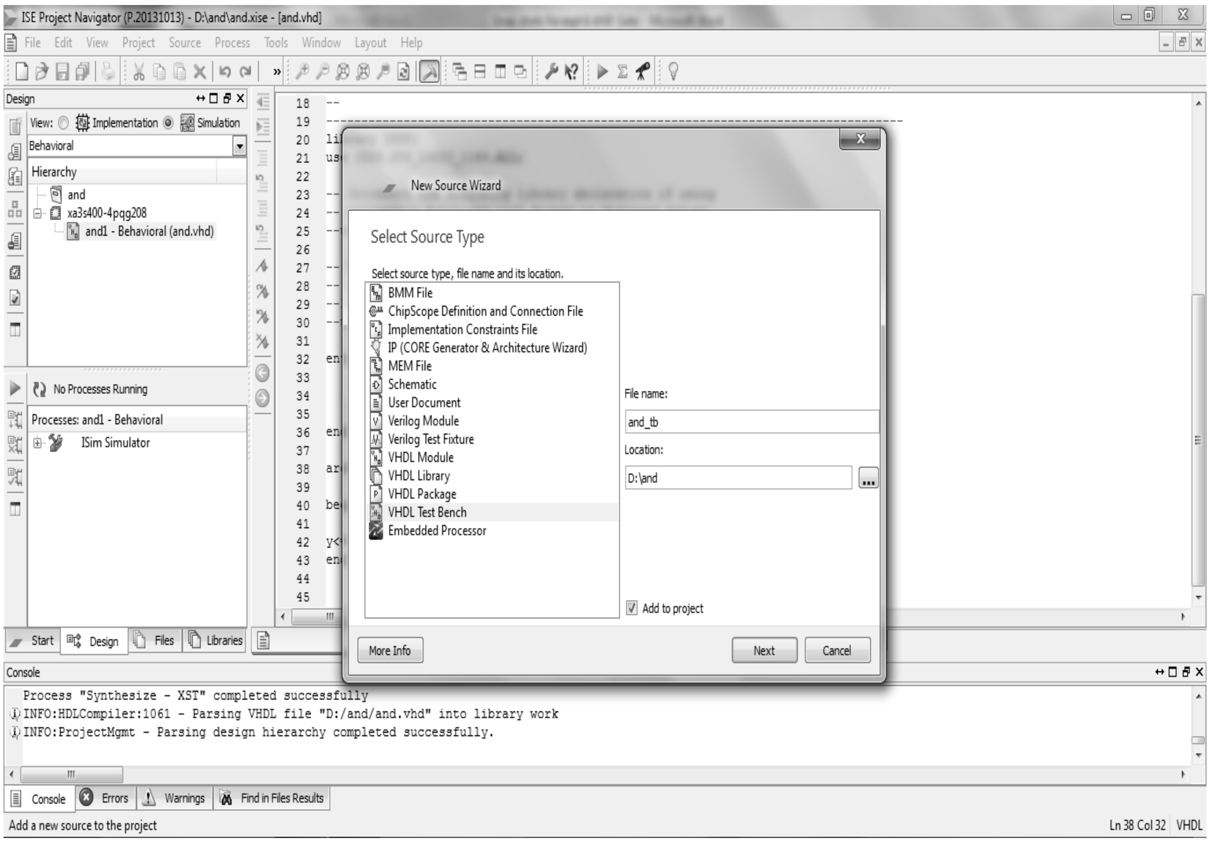
Fig 4.10 Select Simulation tab to create test bench file



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

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

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

entity and1 is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           y : out  STD_LOGIC);
end and1;

architecture Behavioral of and1 is

begin

y<= a and b;
end Behavioral;
```

```
Process "Synthesize - XST" completed successfully
INFO:HDLCompiler:1061 - Parsing VHDL file "D:/and/and.vhd" into library work
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
```

Fig 4.11 (a): Creating test bench file



Select source type, file name and its location.

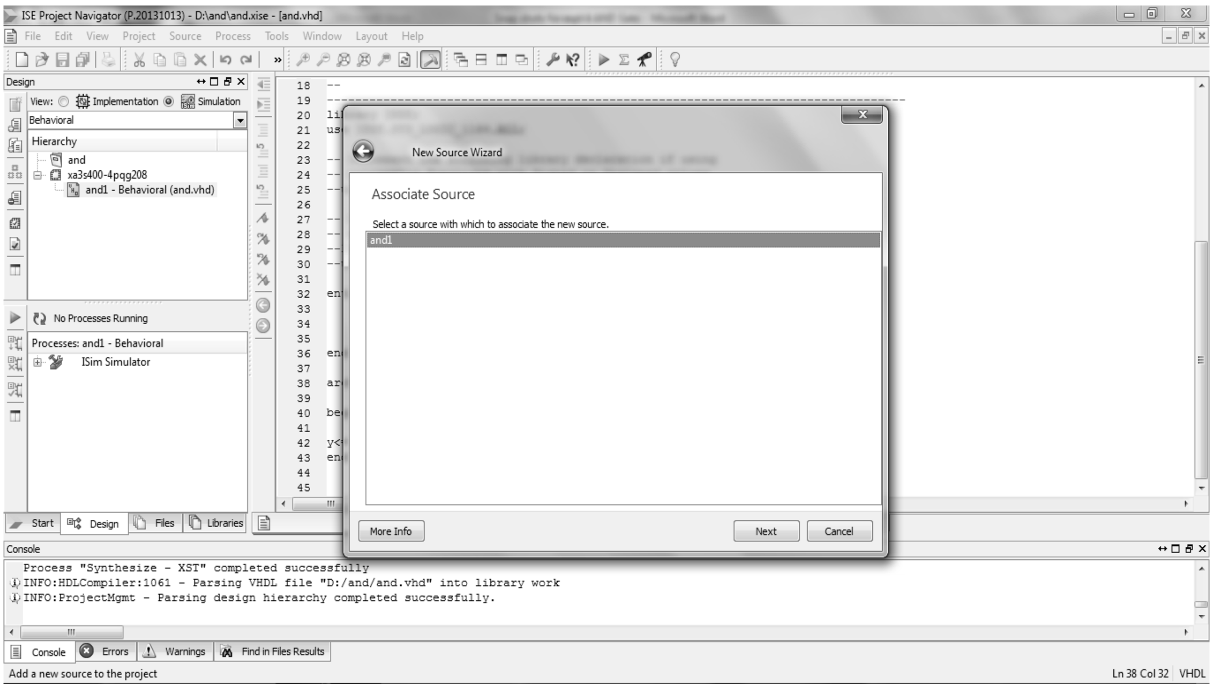
File name: and_tb

Location: D:\and

Add to project ✓

```
Process "Synthesize - XST" completed successfully
INFO:HDLCompiler:1061 - Parsing VHDL file "D:/and/and.vhd" into library work
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
```

Fig 4.11(b): Creating test bench

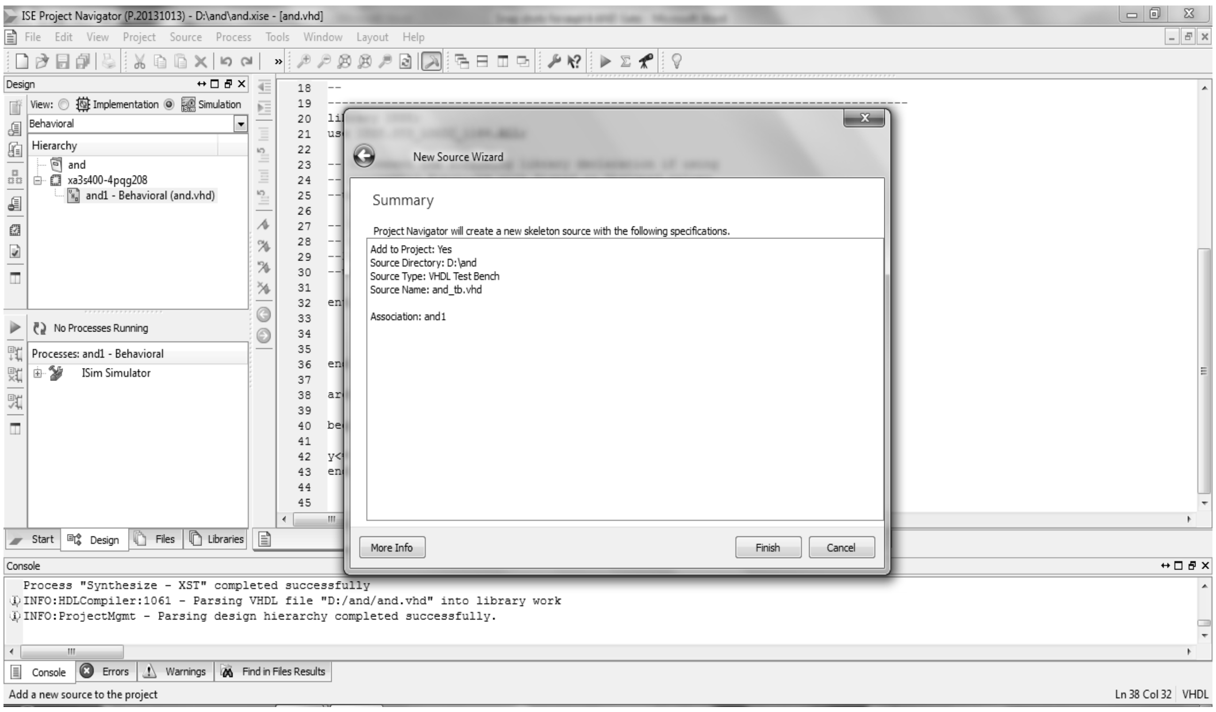


Select a source with which to associate the new source.

and1

```
Process "Synthesize - XST" completed successfully
INFO:HDLCompiler:1061 - Parsing VHDL file "D:/and/and.vhd" into library work
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
```

Fig 4.11(c): Creating test bench



ISE Project Navigator (P.20131013) - D:\and\and.xise - [and.vhd]
New Source Wizard
More Info
Finish
Cancel

Summary

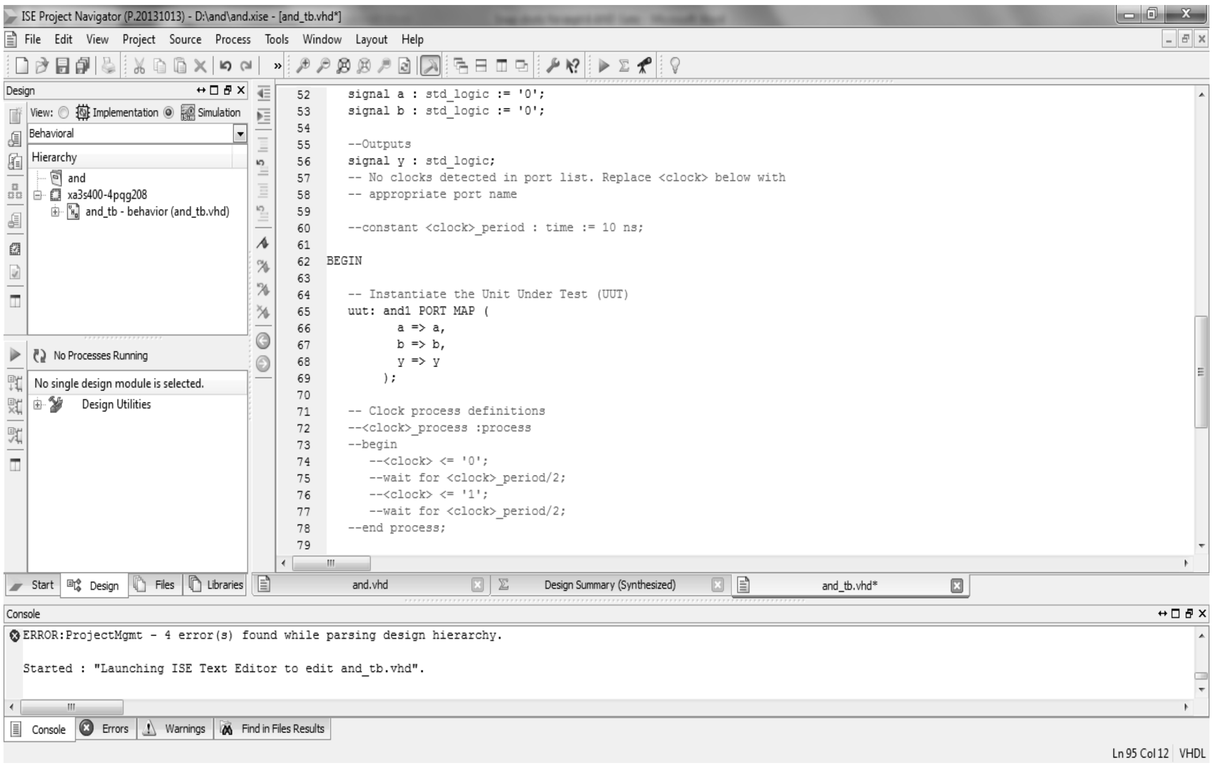
Project Navigator will create a new skeleton source with the following specifications.

```
Add to Project: Yes
Source Directory: D:\and
Source Type: VHDL Test Bench
Source Name: and_tb.vhd

Association: and1
```

```
Process "Synthesize - XST" completed successfully
INFO:HDLCompiler:1061 - Parsing VHDL file "D:/and/and.vhd" into library work
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
```

Add a new source to the project

Fig 4.12 Created Test Bench

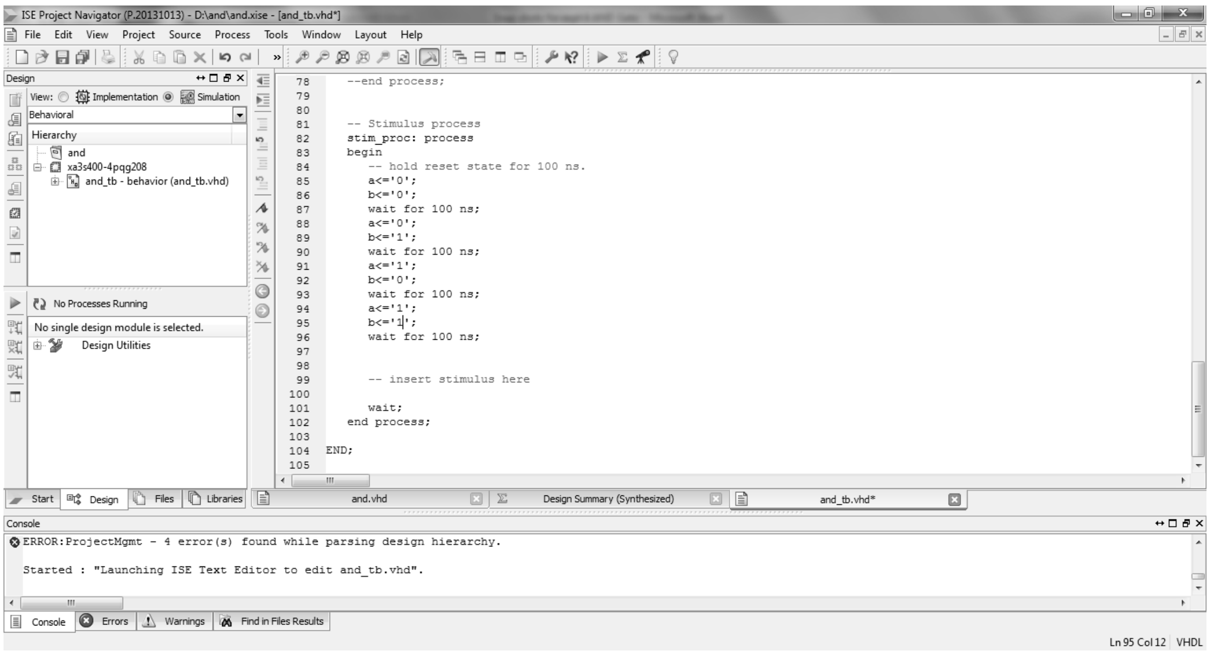


ISE Project Navigator (P.20131013) - D:\and\and.xise - [and_tb.vhd*]
and
xa3s400-4pqg208
and_tb - behavior (and_tb.vhd)
and.vhd
Design Summary (Synthesized)
and_tb.vhd*

```
   signal a : std_logic := '0';
   signal b : std_logic := '0';

   --Outputs
   signal y : std_logic;
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name

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

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: and1 PORT MAP (
          a => a,
          b => b,
          y => y
        );

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

```

```
ERROR:ProjectMgmt - 4 error(s) found while parsing design hierarchy.

Started : "Launching ISE Text Editor to edit and_tb.vhd".
```

Fig 4.13: Disable clock parameters [as per .vhd file]

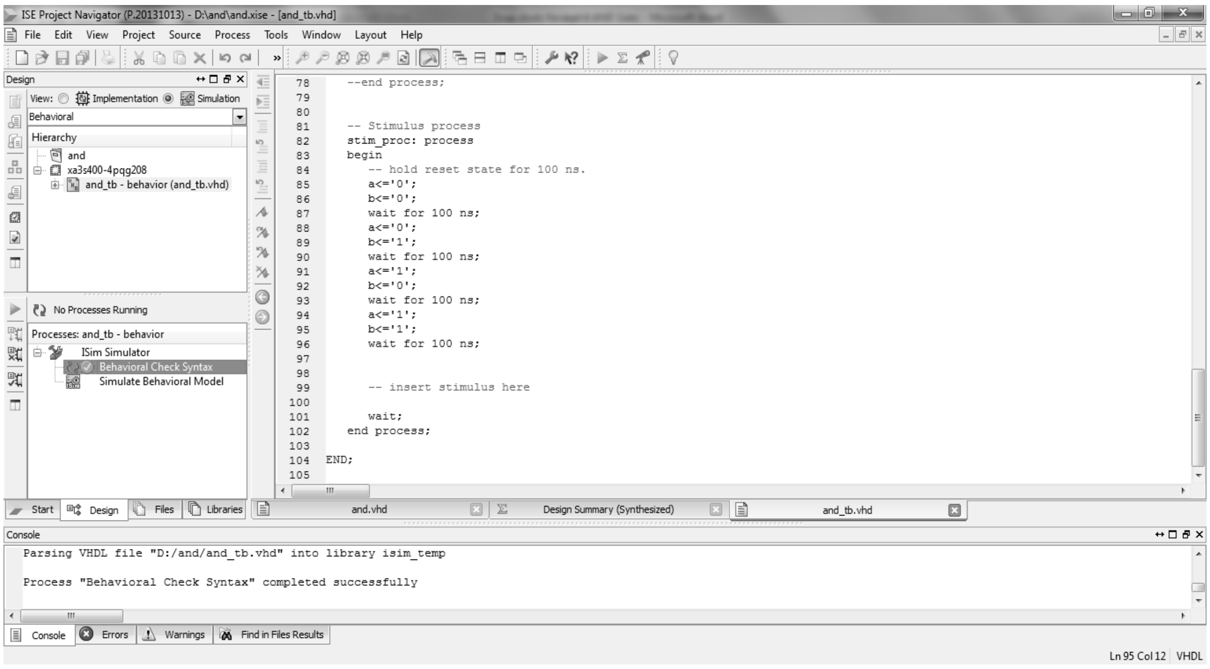

ISE Project Navigator (P.20131013) - D:\and\and.xise - [and_tb.vhd*]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
and
xa3s400-4pqg208
and_tb - behavior (and_tb.vhd)
No Processes Running
No single design module is selected.
Design Utilities
Start Design Files Libraries
and.vhd
Design Summary (Synthesized)
and_tb.vhd*
Console
Console Errors Warnings Find in Files Results
Ln 95 Col 12 VHDL

```
   --end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      a<='0';
      b<='0';
      wait for 100 ns;
      a<='0';
      b<='1';
      wait for 100 ns;
      a<='1';
      b<='0';
      wait for 100 ns;
      a<='1';
      b<='1';
      wait for 100 ns;


      -- insert stimulus here

      wait;
   end process;

END;
```

```
ERROR:ProjectMgmt - 4 error(s) found while parsing design hierarchy.

Started : "Launching ISE Text Editor to edit and_tb.vhd".
```

Fig 4.14: Initialize Input values

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.
   (refer fig.. 4.15)

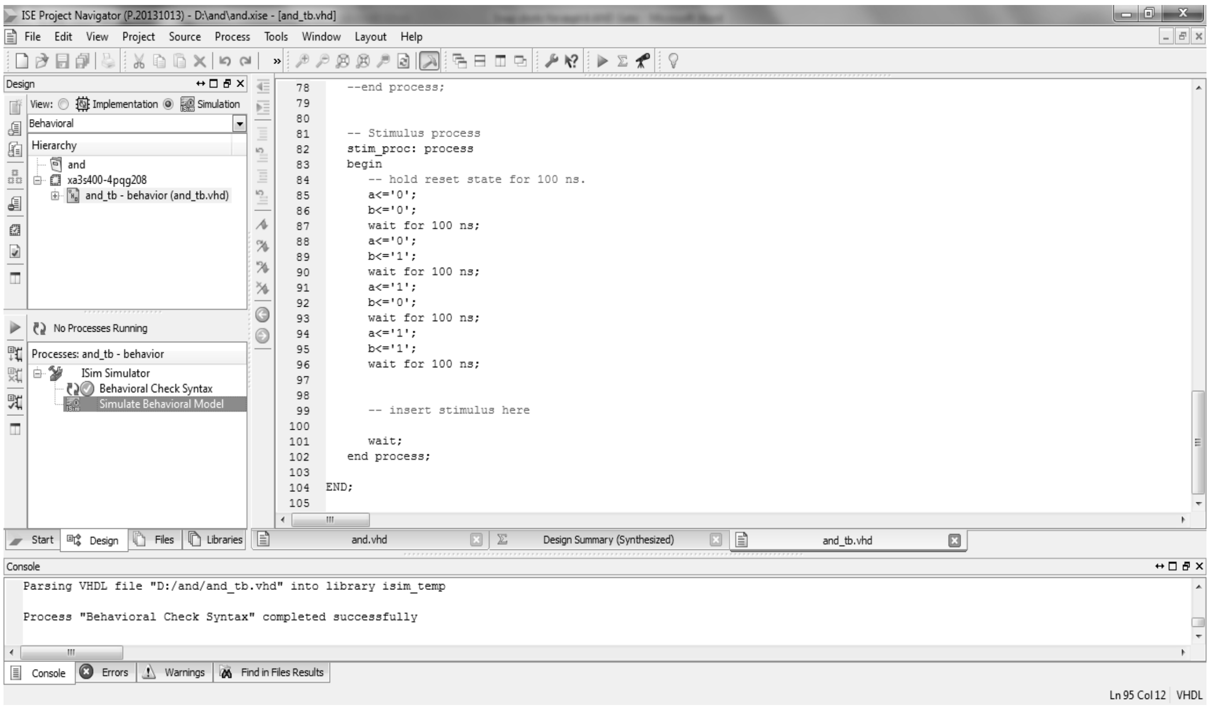

ISE Project Navigator (P.20131013) - D:\and\and.xise - [and_tb.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
and
xa3s400-4pqg208
and_tb - behavior (and_tb.vhd)
No Processes Running
Processes: and_tb - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start Design Files Libraries
and.vhd
Design Summary (Synthesized)
and_tb.vhd
Console
Console Errors Warnings Find in Files Results
Ln 95 Col 12 VHDL

```
   --end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      a<='0';
      b<='0';
      wait for 100 ns;
      a<='0';
      b<='1';
      wait for 100 ns;
      a<='1';
      b<='0';
      wait for 100 ns;
      a<='1';
      b<='1';
      wait for 100 ns;


      -- insert stimulus here

      wait;
   end process;

END;
```

```
Parsing VHDL file "D:/and/and_tb.vhd" into library isim_temp

Process "Behavioral Check Syntax" completed successfully
```

Fig 4.15: Behavioral Check Syntax

8. Double click on “Simulate Behavioral Model” in the “Process Pane”, which will open the ISim software with your test bench loaded. (refer fig.. 4.16)



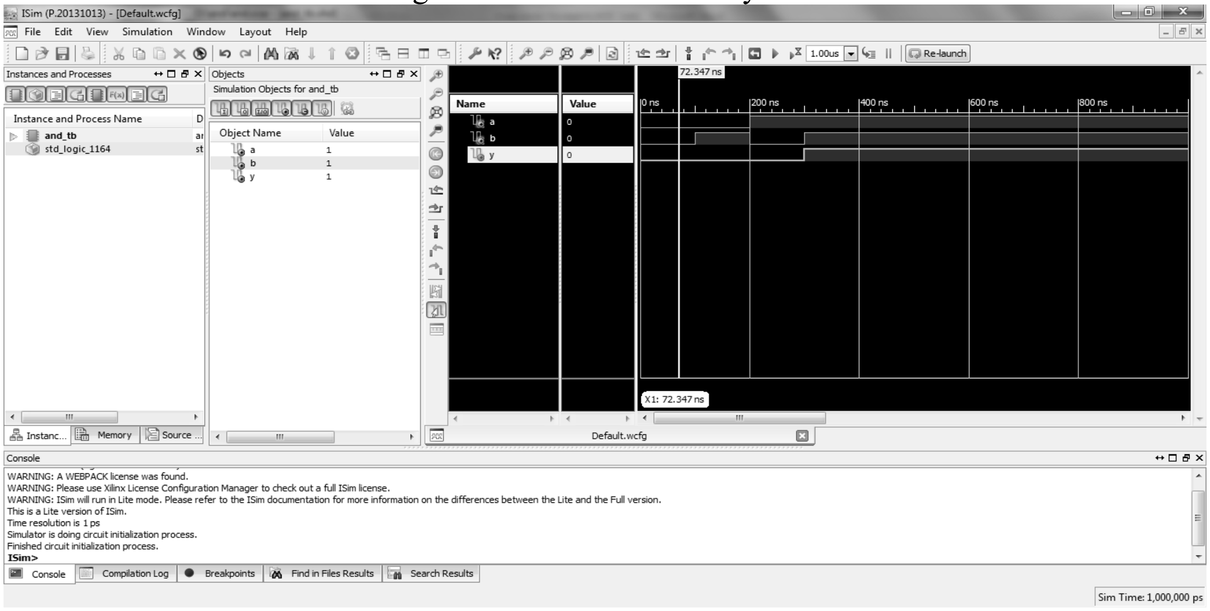
```
    --end process;


    -- Stimulus process
    stim_proc: process
    begin
       -- hold reset state for 100 ns.
       a<='0';
       b<='0';
       wait for 100 ns;
       a<='0';
       b<='1';
       wait for 100 ns;
       a<='1';
       b<='0';
       wait for 100 ns;
       a<='1';
       b<='1';
       wait for 100 ns;


       -- insert stimulus here

       wait;
    end process;

 END;
```

```
Parsing VHDL file "D:/and/and_tb.vhd" into library isim_temp

Process "Behavioral Check Syntax" completed successfully
```

ISE Project Navigator (P.20131013) - D:\and\and.xise - [and_tb.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
and
xa3s400-4pqg208
and_tb - behavior (and_tb.vhd)
No Processes Running
Processes: and_tb - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start Design Files Libraries
and.vhd
Design Summary (Synthesized)
and_tb.vhd
Console
Console Errors Warnings Find in Files Results
Ln 95 Col 12 VHDL

Fig 4.16: Simulation Check window

9. ISim simulator window will open with your simulation executed, as shown in Fig. 4.17 where one can simulate designs and check errors if any.



```
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Object Name	Value
a	1
b	1
y	1

Name	Value
a	0
b	0
y	0

ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
1.00us
Re-launch
Instances and Processes
Instance and Process Name
and_tb
std_logic_1164
Objects
Simulation Objects for and_tb
72.347 ns
0 ns
200 ns
400 ns
600 ns
800 ns
X1: 72.347 ns
Instanc... Memory Source ...
Default.wcfg
Console
Console Compilation Log Breakpoints Find in Files Results Search Results
Sim Time: 1,000,000 ps

Fig 4.17: Simulation Output

Sample Program: Implement AND gate using Data flow and Behavior Model.

Data Flow Model:

VHDL Code

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

Entity AND-GATE is
Port ( A,B : in  STD_LOGIC;
        P : out  STD_LOGIC);
end AND-GATE;

Architecture dataflow of AND_GATE is
begin
P < = A and B;
end dataflow;
```

Behavior Model:

VHDL Code

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

Entity AND is
port(
   A,B: in std_logic;
   C: out std_logic
);
end AND;

Architecture arch of AND is
   begin
   process(a, b)
   begin
   if a='1' and b='1' then
      c <= '1';
   else
        c <= '0';
   end if;
   end process;
end arch;
```

Program Statement for Student: Implement OR Gate using Data flow and Behavior Model.

Data Flow Model:

VHDL Code

Behavior Model:

VHDL Code

XI Resources Used:

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

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

....
....
....
....
....

XIII Precautions followed:

....
....
....
....
....

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

Observations for Problem Statement given to Student.

A	B	Y= A + B
0	0	
0	1	
1	0	
1	1	

XV Results (output of Program)

....
....
....
....
....

XVI Interpretation of result

....
....
....
....
....

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

....
....
....
....
....

XVIII 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 term model. List different types of Model.
2. Compare different types of model.
3. Give the syntax of Entity and Architecture with example.
4. Implement NAND gate using Behavioral Model.
5. Explain the Process statement with an example.

[Space for Answers]

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

XIX References / Suggestions for further reading

1 https://startingelectronics.org/software/VHDL-CPLD-course/tut2-AND-and-OR-gates/
2 https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/
3 https://www.tutorialspoint.com/vlsi_design/vlsi_design_vhdl_introduction.htm

XX 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 of report Answer to sample questions.	05%
	Total (25 Marks)	**100%**

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 05: Implement Half/Full adder/ subtractor using FPGA.

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 Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronic circuits and equipments."**
Debug VHDL programme for the given application.

IV Relevant Course Outcome

Debug VHDL programme for the given application.

V Practical Outcome

Implement Half/Full adder/ subtractor using FPGA.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum Theoretical Background

a) 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

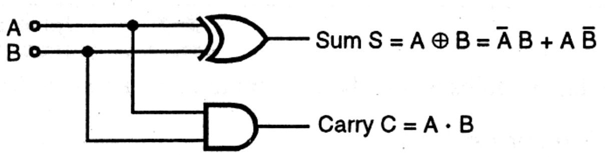


Fig 5.1: Half Adder

b) 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

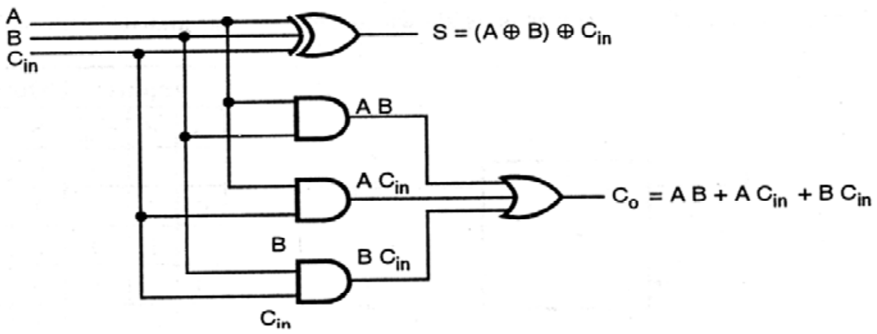


Fig 5.2: Full Adder

c) Half Subtractor: A logic circuit used for subtraction of B (Subtrahend) from A (minuend) where A and B are 1-bit numbers is referred to as Half Subtractor.

Table 5.3: Truth table for Half Subtractor

A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

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

$B = \overline{A}B$

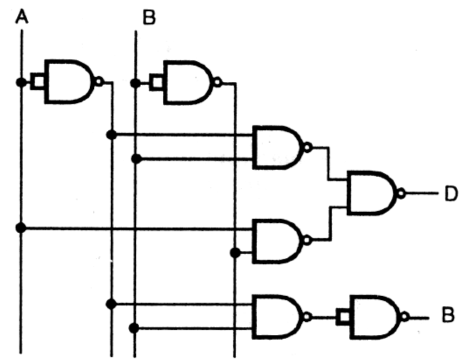


Fig 5.3: Half Subtractor

d) Full Subtractor: A full subtractor is used for performing multibit subtraction where the borrow from the previous bit position is available. This circuit has three inputs A (minuend), B (subtrahend) and Bin (borrow from previous stage) and two outputs Difference (D) and Borrow (B0).

Table 5.4: Truth table for Full Subtractor

A	B	Bin	Difference(D)	Borrow (B0)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

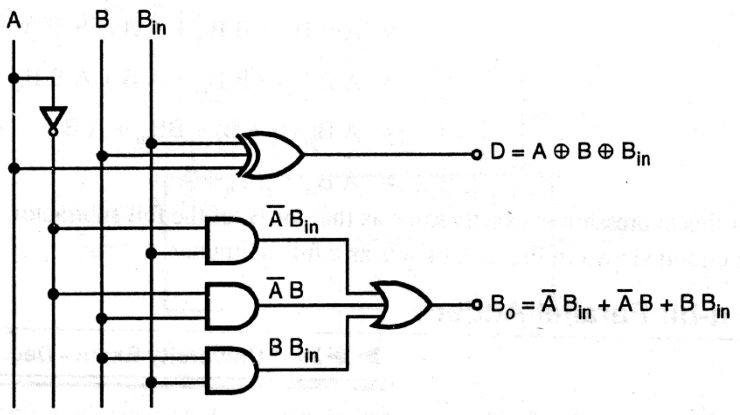
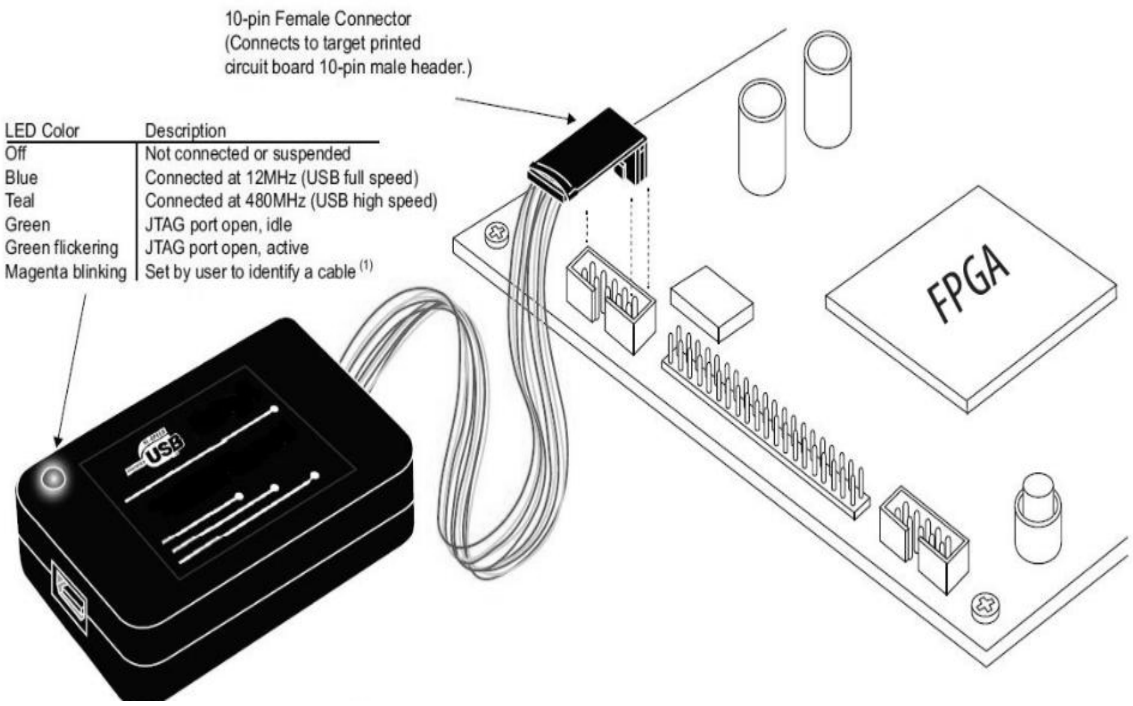

A
B
B_{in}
$D = A \oplus B \oplus B_{in}$
$\bar{A} B_{in}$
$\bar{A} B$
$B B_{in}$
$B_o = \bar{A} B_{in} + \bar{A} B + B B_{in}$

Fig 5.4: Full Subtractor

VIII Practical Circuit Diagram:

a) Practical JTAG cable setup:-


10-pin Female Connector
(Connects to target printed
circuit board 10-pin male header.)
USB
FPGA

LED Color	Description
Off	Not connected or suspended
Blue	Connected at 12MHz (USB full speed)
Teal	Connected at 480MHz (USB high speed)
Green	JTAG port open, idle
Green flickering	JTAG port open, active
Magenta blinking	Set by user to identify a cable (1)

Fig. 5.5a: Practical Setup with JTAG CABLE

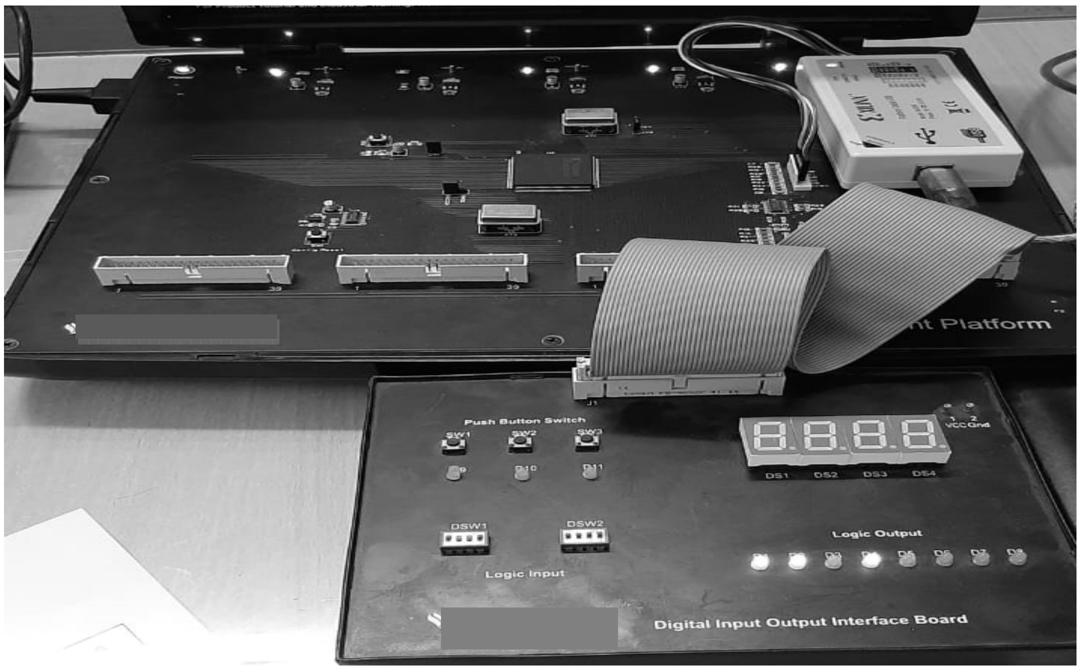
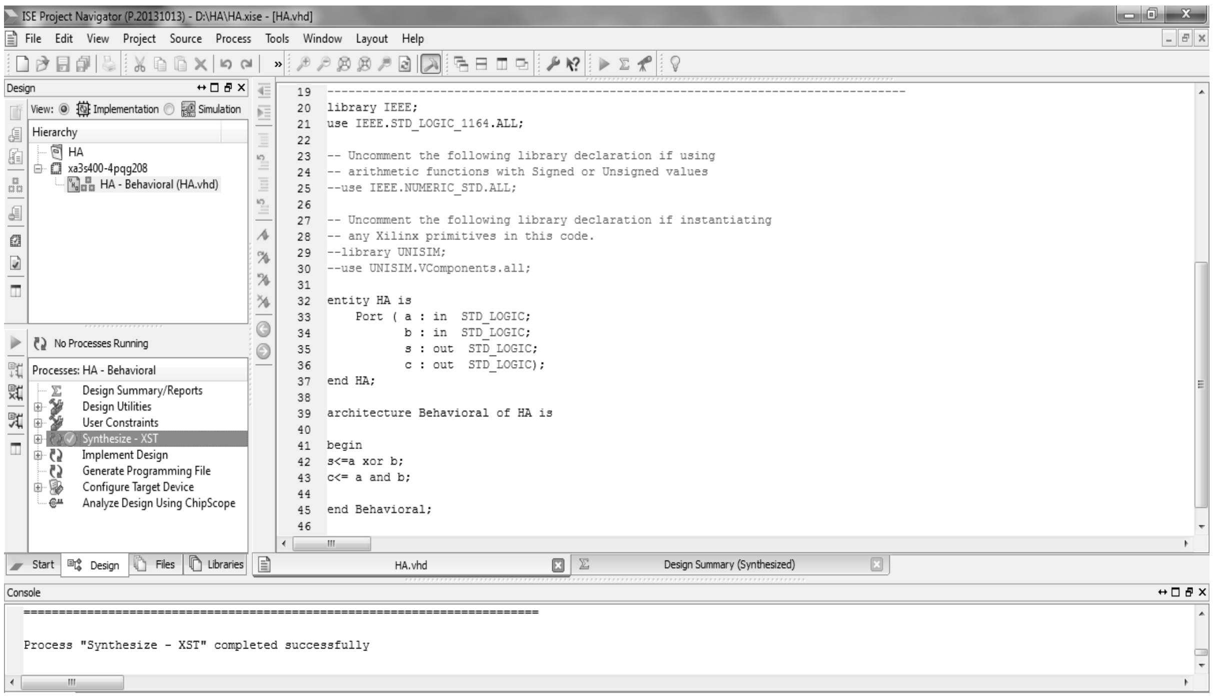


Fig. 5.5.b: Lab Practical Setup

b) Create Encoder HA .vhd file:

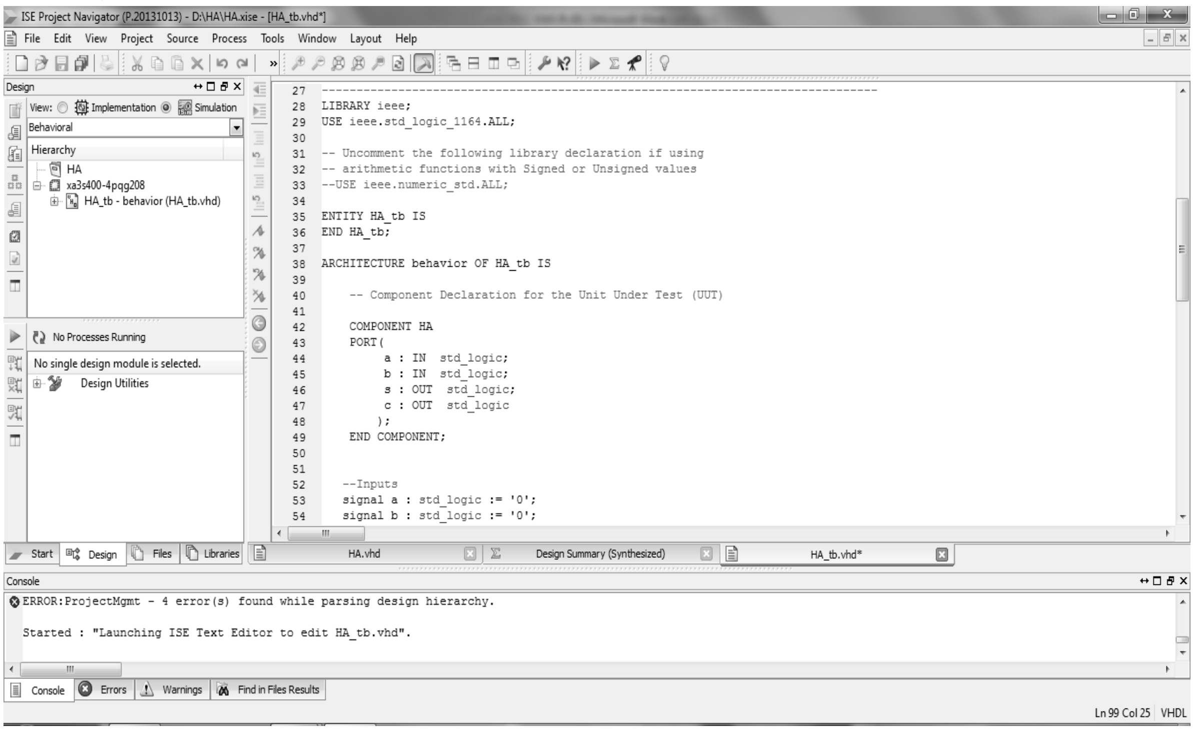


```
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

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

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

entity HA is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           s : out  STD_LOGIC;
           c : out  STD_LOGIC);
end HA;

architecture Behavioral of HA is

begin
s<=a xor b;
c<= a and b;

end Behavioral;
```

```
===========================================================================

Process "Synthesize - XST" completed successfully
```

Fig 5.6: HA .vhd file

c) Create Encoder HA test bench file:

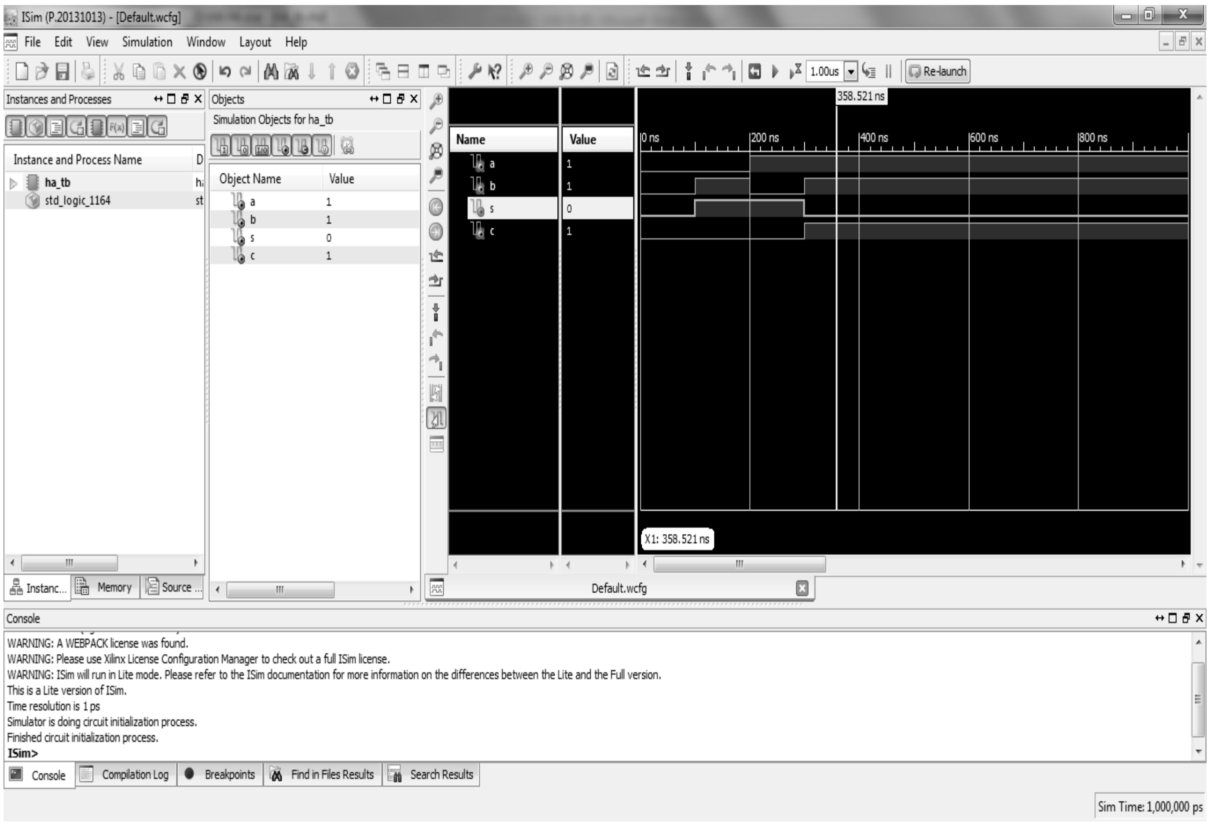


```
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

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

ENTITY HA_tb IS
END HA_tb;

ARCHITECTURE behavior OF HA_tb IS

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

    COMPONENT HA
    PORT(
         a : IN  std_logic;
         b : IN  std_logic;
         s : OUT  std_logic;
         c : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal a : std_logic := '0';
   signal b : std_logic := '0';
```

```
ERROR:ProjectMgmt - 4 error(s) found while parsing design hierarchy.

Started : "Launching ISE Text Editor to edit HA_tb.vhd".
```

Fig 5.7: HA Test Bench file

HA test bench Simulation waveforms:



Object Name	Value
a	1
b	1
s	0
c	1

```
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Sim Time: 1,000,000 ps

Fig 5.8 HA Simulation output

d) Actual Experimental set up used in laboratory:

IX 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	Loaded with open source IDE, simulation and program downloading software.	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 top-level FPGA design, by doing the following in ISE:
   In the ISE software, select File > New Project.
   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.
5. Create Test Bench file- A **test bench** is **HDL** code that allows documentation, 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 there are no 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 Fig. 5.8 where one can simulate designs and check for errors.
10. After simulation implement 2 as 4 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 File and Run

14. Go to Config. 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: Implement Half Adder and Full Adder using data flow model.

Half Adder:

VHDL Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOG IC_ARITH.ALL;
useIEEE.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;
```

Full Adder:

VHDL Code

```
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;
```

Program Statement for Student: Implement Half and Full Subtractor using data flow model.

<u>Half Subtractor:</u>

VHDL Code

<u>Full Subtractor :</u>

VHDL Code

XI Resources Used:

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

XII Actual Procedure Followed (use additional sheets if required)

....
....
....
....
....

XIII Precautions followed:

....
....
....
....
....

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

Observations for Half Subtractor for Problem Statement given to Student.

A	B	D	B
0	0		
0	1		
1	0		
1	1		

XV Results (output of Program)

....
....
....
....
....

XVI Interpretation of result

....
....
....
....
....

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

....
....
....
....
....

XVIII 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 Behavioral Model.
2. Write the VHDL code for full subtractor using if-then-else statement**.**
3. Write VHDL code for full adder using mixed style of model.

[Space for Answers]

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

XIX References / Suggestions for further reading

1 https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/
2 https://www.tutorialspoint.com/vlsi_design/vlsi_design_vhdl_introduction.htm
3 https://www.tutorialspoint.com/vlsi_design/vhdl_programming_for_combinational_circuits.htm
4 http://ece2day.blogspot.com/2012/10/vhdl-tutorial-introducction-part-7.html

XX 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 of report Answer to sample questions.	05%
	Total (25 Marks)	**100%**

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 06: Implement 8:1 Multiplexer using FPGA.

I Practical Significance

In electronics, a multiplexer (or mux) is a device that selects between several analog or digital input signals and forwards it to a single output line. Multiplexers are used to implement Boolean functions of multiple variables. This practical will help the students to implement the 8:1 using FPGA.

II Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronic circuits and equipments."**

- Use relevant VHDL model for the given applications.
- Debug VHDL programme for the given application.

IV Relevant Course Outcome

Debug VHDL programme for the given application.

V Practical Outcome

Implement 8:1 Multiplexer using FPGA.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum Theoretical Background

A multiplexer (Mux) is a device that allows digital information from several sources to be routed on to a single line for transmission over that line to a common destination. The basic multiplexer has several data input lines (D_0, $D_{1,}$….D_n-1) and single output line(Y). It has also data select inputs (S_0, $S_{1,}$……S_{m-1}) which permits digital data on any one input to be switched to the output line. Multiplexers are also known as Data Selectors. (refer fig. 6.1)

The select lines *m* and the input lines *n* are related by the formula: 2^m= n.

Depending upon the digital code applied at the select inputs one out of 'n' data sources is selected and transmitted to the single output Y.

Strobe [E or G] signal is used for cascading of inputs. It is an active low signal means it will perform the intended function only when it is active low i.e. at logic 0.

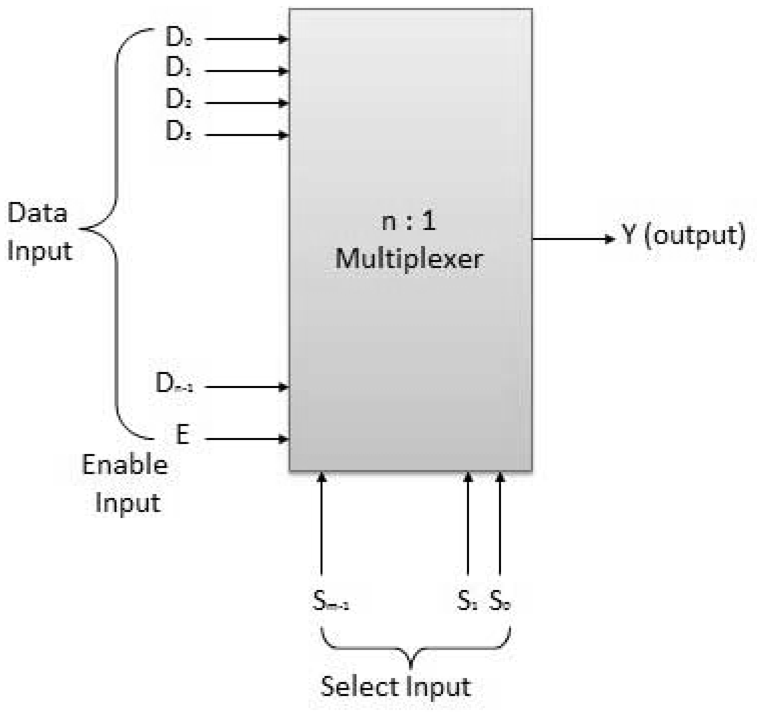
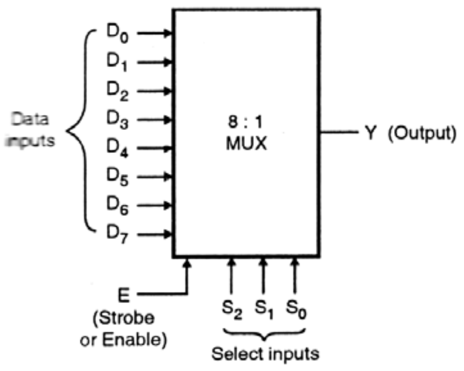


Fig 6.1: Basic Symbol of Multiplexer

8: 1 Multiplexer:

An 8-to-1 multiplexer consists of eight data inputs D0 through D7, three input select lines S2 through S0 and a single output line Y. Depending on the select lines combinations, multiplexer decodes the inputs.

Fig. 6.2 shows the block diagram of an 8:1 multiplexer with Enable input pin which enable or disable the multiplexer. Since the number data bits given to the MUX are eight then 3 bits ($2^3=8$) are needed to select one of the eight data bits



:

Fig 6.2: Block Diagram of 8:1 Multiplexer

The truth table for an 8-to1 multiplexer is given below with eight combinations of inputs so as to generate each output corresponds to input.

Table 6.1: Truth Table of 8:1 Multiplexer

Enable	Select inputs			Output
E	**S_2**	**S_1**	**S_0**	**Y**
0	X	X	X	0
1	0	0	0	D_0
1	0	0	1	D_1
1	0	1	0	D_2
1	0	1	1	D_3
1	1	0	0	D_4
1	1	0	1	D_5
1	1	1	0	D_6
1	1	1	1	D_7

For example:

- If S2= 0, S1=1 and S0=0 then the data output Y is equal to D2.

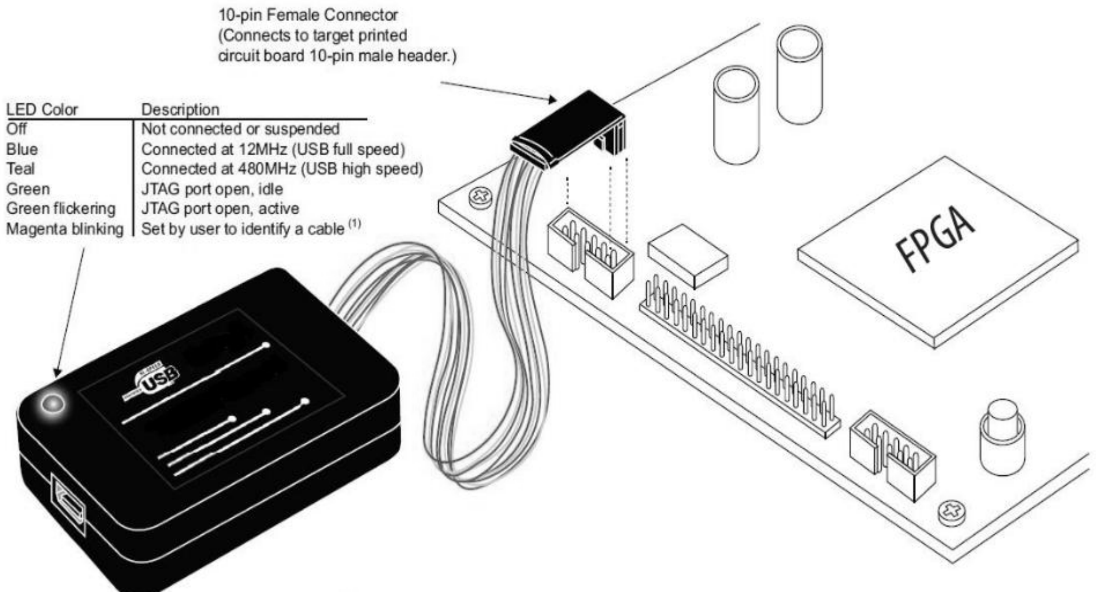
Similarly the data outputs D0 to D7 will be selected through the combinations of S2, S1 and S0.

From the above truth table, the Boolean equation for the output is given as:

$$Y = \overline{S}_2\overline{S}_1\overline{S}_0D_0 + \overline{S}_2\overline{S}_1S_0D_1 + \overline{S}_2S_1\overline{S}_0D_2 + \overline{S}_2S_1S_0D_3 + S_2\overline{S}_1\overline{S}_0D_4 + S_2\overline{S}_1S_0D_5 + S_2S_1\overline{S}_0D_6 + S_2S_1S_0D_7$$

VIII Practical Circuit Diagram:

a) Practical JTAG cable setup:-


10-pin Female Connector
(Connects to target printed
circuit board 10-pin male header.)
FPGA
USB

LED Color	Description
Off	Not connected or suspended
Blue	Connected at 12MHz (USB full speed)
Teal	Connected at 480MHz (USB high speed)
Green	JTAG port open, idle
Green flickering	JTAG port open, active
Magenta blinking	Set by user to identify a cable (1)

Fig. 6.3 a: Practical Setup with JTAG CABLE

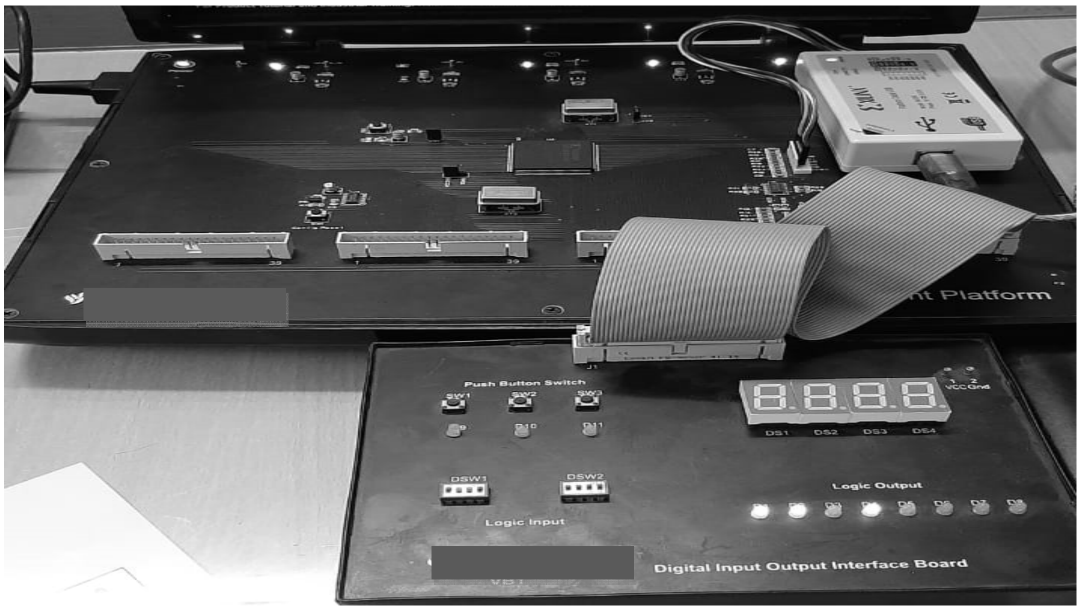
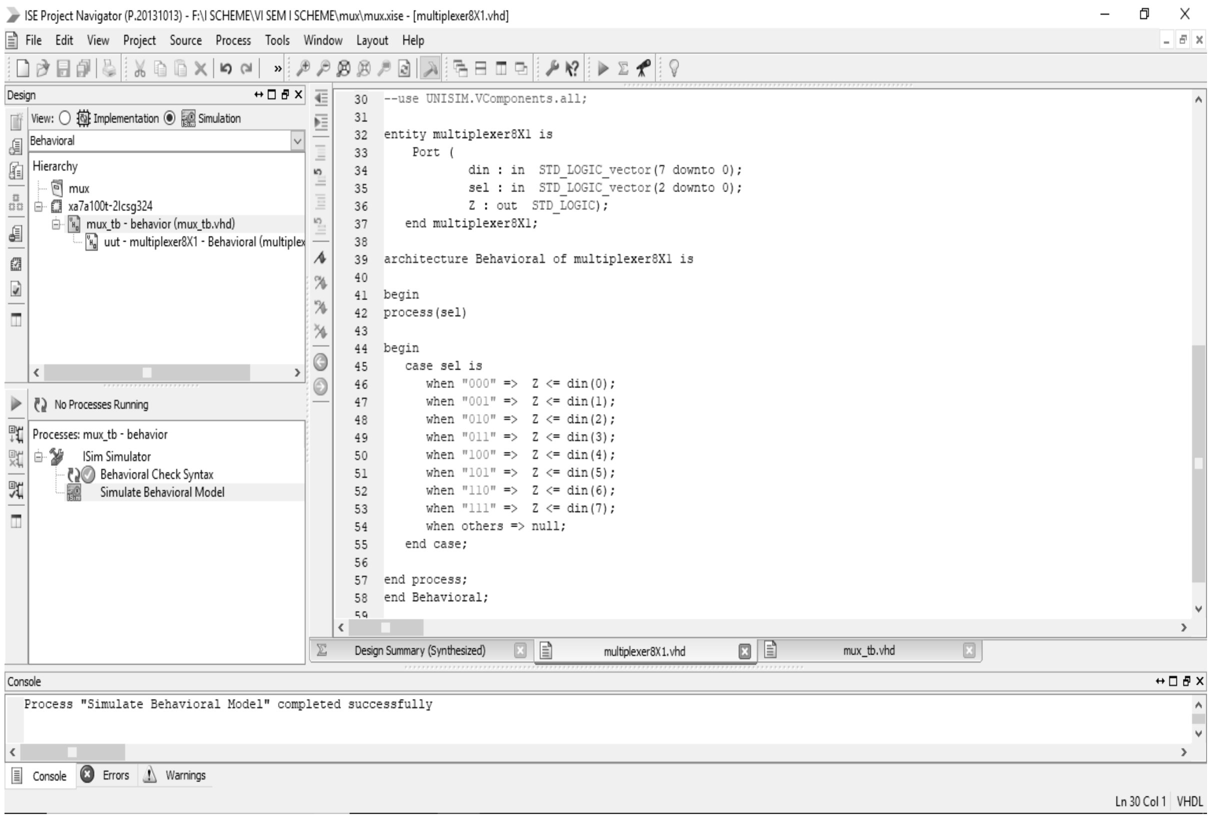


Fig. 6.3.b: Lab Practical Setup

b) Create MUX .vhd file:

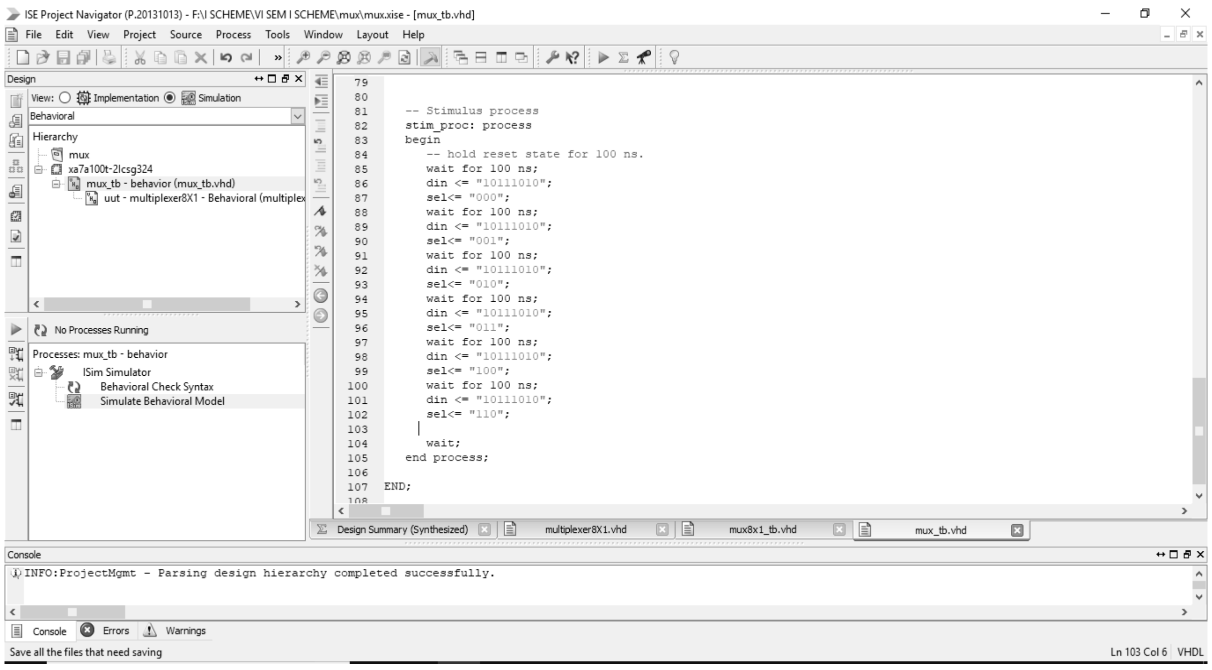

ISE Project Navigator (P.20131013) - F:\I SCHEME\VI SEM I SCHEME\mux\mux.xise - [multiplexer8X1.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
mux
xa7a100t-2lcsg324
mux_tb - behavior (mux_tb.vhd)
uut - multiplexer8X1 - Behavioral (multiplex
No Processes Running
Processes: mux_tb - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Design Summary (Synthesized)
multiplexer8X1.vhd
mux_tb.vhd
Console
Errors
Warnings
Ln 30 Col 1 VHDL

```
--use UNISIM.VComponents.all;

entity multiplexer8X1 is
    Port (
            din : in  STD_LOGIC_vector(7 downto 0);
            sel : in  STD_LOGIC_vector(2 downto 0);
            Z : out  STD_LOGIC);
   end multiplexer8X1;

architecture Behavioral of multiplexer8X1 is

begin
process(sel)

begin
   case sel is
      when "000" =>  Z <= din(0);
      when "001" =>  Z <= din(1);
      when "010" =>  Z <= din(2);
      when "011" =>  Z <= din(3);
      when "100" =>  Z <= din(4);
      when "101" =>  Z <= din(5);
      when "110" =>  Z <= din(6);
      when "111" =>  Z <= din(7);
      when others => null;
   end case;

end process;
end Behavioral;
```

```
Process "Simulate Behavioral Model" completed successfully
```

Fig 6.4: MUX .vhd file

c) Create MUX test bench file:

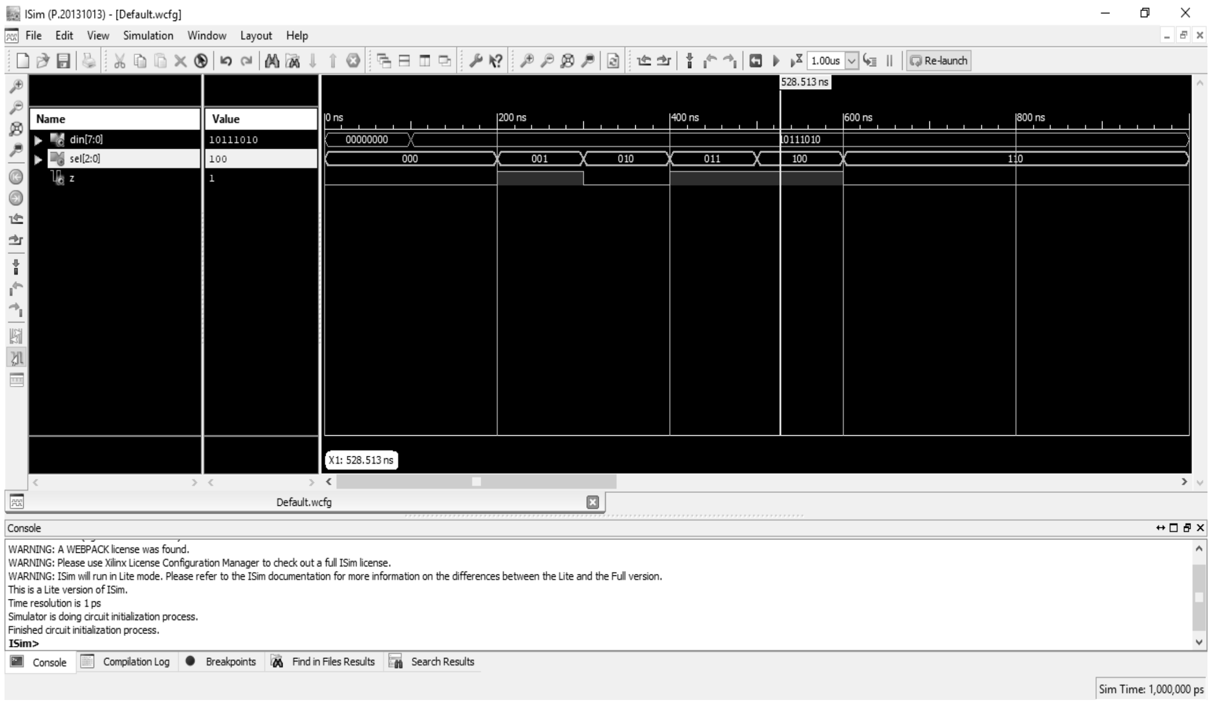


```
    -- Stimulus process
    stim_proc: process
    begin
       -- hold reset state for 100 ns.
       wait for 100 ns;
       din <= "10111010";
       sel<= "000";
       wait for 100 ns;
       din <= "10111010";
       sel<= "001";
       wait for 100 ns;
       din <= "10111010";
       sel<= "010";
       wait for 100 ns;
       din <= "10111010";
       sel<= "011";
       wait for 100 ns;
       din <= "10111010";
       sel<= "100";
       wait for 100 ns;
       din <= "10111010";
       sel<= "110";

       wait;
    end process;

END;
```

```
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
```

Fig 6.5: MUX Test Bench file

d) MUX test bench Simulation waveforms:



```
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Fig 6.6 MUX Simulation output

e) Actual Experimental set up used in laboratory:

IX 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	Loaded with open source IDE, simulation and program downloading software.	1 No.

X Precautions to be followed:

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

XI Procedure:

1. Create the Xilinx ISE project for your top-level FPGA design, by doing the following in ISE:

   In the ISE software, **select File > New Projec**t.

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.
5. Create Test Bench file- A **test bench** is **HDL** code that allows documentation, 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 there are on syntax errors.
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 Fig. 6.6. where can simulate designs and check for errors.
10. After simulation implement 8:1 multiplexer 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 Config. 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: Implement 8:1 Multiplexer.
8:1 Multiplexer:

VHDL Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOG IC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity mux8_1 is
port(

din : in STD_LOGIC_VECTOR(7 downto 0);
sel : in STD_LOGIC_VECTOR(2 downto 0)
Y : out STD_LOGIC
        );
end mux8_1;

architecture multiplexer8_1_arc of mux8_1 is
begin
      Y <= din(7) when (sel="000") else
          din(6) when (sel="001") else
          din(5) when (sel="010") else
          din(4) when (sel="011") else
          din(3) when (sel="100") else
          din(2) when (sel="101") else
          din(1) when (sel="110") else
          din(0);
end multiplexer8_1_arc
```

Program Statement for Student: Implement 4:1 Multiplexer using structural model.
4:1 Multiplexer

VHDL Code

XII Resources Used:

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

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

....
....
....
....
....

XIV Precautions followed:

....
....
....
....
....

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

Observations for Problem Statement given to Student.

S0	S1	Y
0	0	
0	1	
1	0	
1	1	

XVI Results (output of Program)

....
....
....
....
....

XVII Interpretation of result

....
....
....
....
....

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

....
....
....
....
....

XIX 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 if statement.
2. Write the VHDL code for 4:1 multiplexer using the "case" statement.

[Space for Answers]

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

XX References / Suggestions for further reading

1 https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/
2 https://www.tutorialspoint.com/vlsi_design/vlsi_design_vhdl_introduction.htm
3 https://www.tutorialspoint.com/vlsi_design/vhdl_programming_for_combinational_circuits.htm
4 https://www.electronicshub.org/multiplexerandmultiplexing/
5 http://ece2day.blogspot.com/2012/10/vhdl-tutorial-introducction-part-7.html
6 https://www.allaboutcircuits.com/technical-articles/sequential-vhdl-if-and-case-statements/

XXI 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 of report Answer to sample questions.	05%
	Total (25 Marks)	**100%)**

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 07: Implement 1:8 Demultiplexer using FPGA.

I Practical Significance

In electronics, a demultiplexer (or demux) is a device that takes a single input line and routes it to one of several digital output lines. Demultiplexer can be used to implement general purpose logic. By setting the input to true, the demux behaves as a decoder. This practical will help the students to implement the 8:1 using FPGA.

II Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronic circuits and equipments."**

- Use relevant VHDL model for the given applications.
- Debug VHDL programme for the given application.

IV Relevant Course Outcome

Debug VHDL programme for the given application.

V Practical Outcome

Implement 1: 8 Demultiplexer using FPGA.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum Theoretical Background

A Demultiplexer is a data distributor read as DEMUX. It is opposite to multiplexer or MUX. It is a process of taking information from one input and transmitting over one of many outputs.

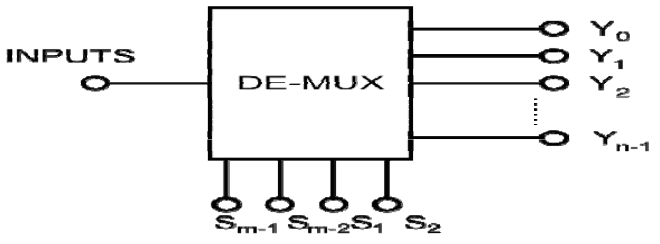


Fig 7.1: Symbol for Demultiplexer

Demultiplexing is the process of converting a signal containing multiple analog or digital signals backs into the original and separate signals. A demultiplexer of 2^n outputs has m select lines.
The enable input will enable the demultiplexer. The relation between the n output lines and m select lines as follows: $n = 2^m$

1:8 Demultiplexer: The block diagram of a 1: 8 De-MUX having only one data input D and eight outputs. It has three select lines S0, S1, S2 which are used as the control signals. There is a strobe input E which is active low. The data bit at input D is transferred to one of the eight outputs depending upon the states of the select lines.

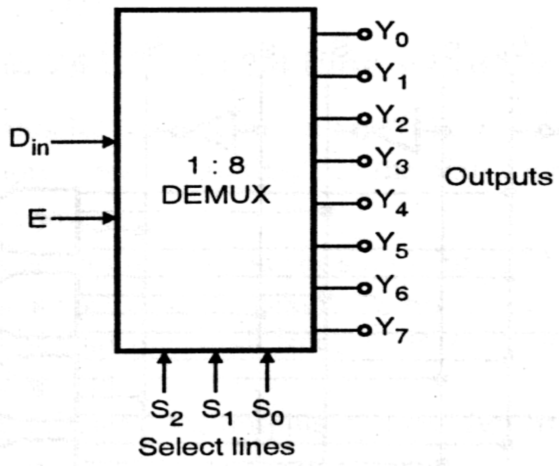


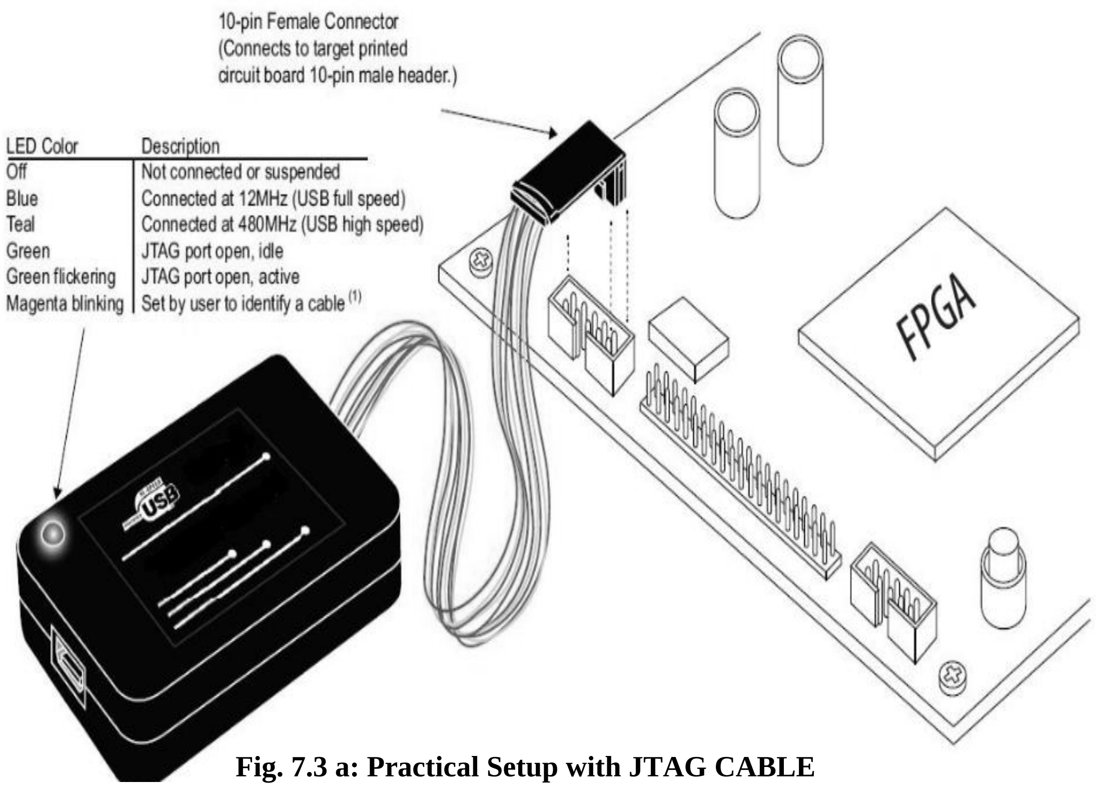
Fig 7.2: 1:8 Demultiplexer

Table 7.1: Truth Table of 1:8 Demux

Data Input	Select Inputs			Outputs							
D	S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	0	0	0	0	0	D
D	0	0	1	0	0	0	0	0	0	D	0
D	0	1	0	0	0	0	0	0	D	0	0
D	0	1	1	0	0	0	0	D	0	0	0
D	1	0	0	0	0	0	D	0	0	0	0
D	1	0	1	0	0	D	0	0	0	0	0
D	1	1	0	0	D	0	0	0	0	0	0
D	1	1	1	D	0	0	0	0	0	0	0

VIII Practical Circuit Diagram:

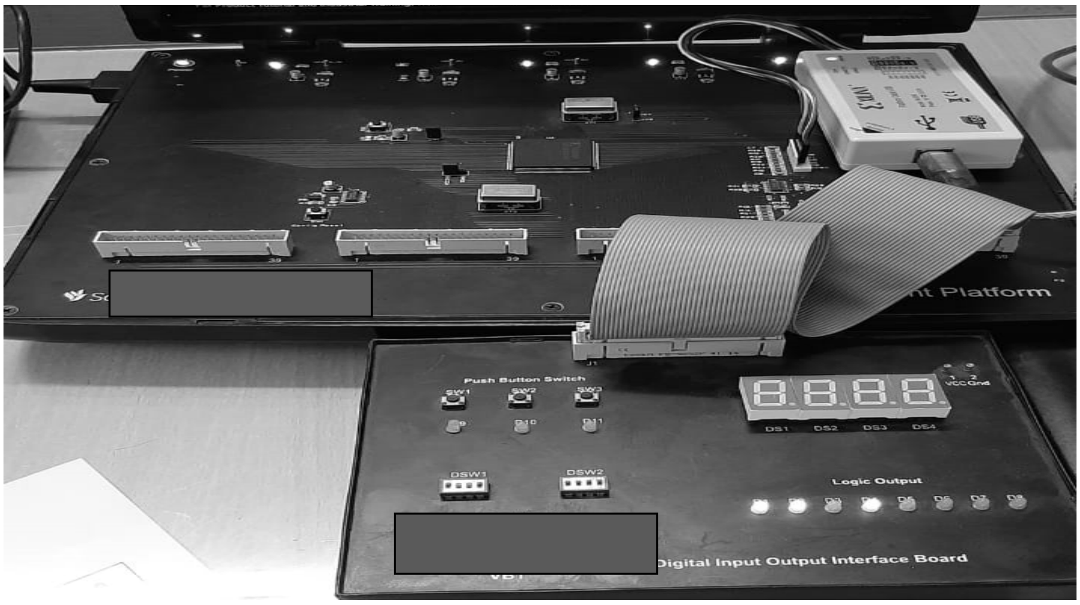
a) Practical JTAG cable setup:-



LED Color	Description
Off	Not connected or suspended
Blue	Connected at 12MHz (USB full speed)
Teal	Connected at 480MHz (USB high speed)
Green	JTAG port open, idle
Green flickering	JTAG port open, active
Magenta blinking	Set by user to identify a cable (1)

10-pin Female Connector
(Connects to target printed
circuit board 10-pin male header.)
USB
FPGA

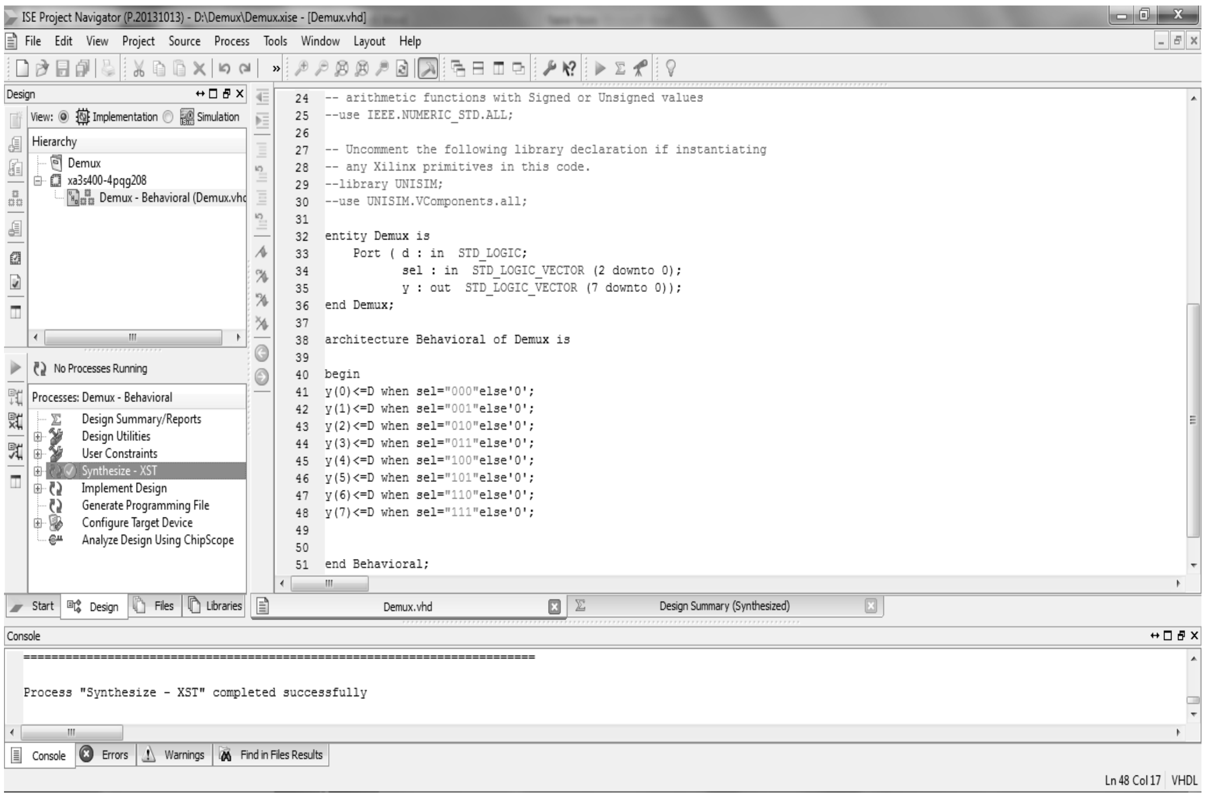
Fig. 7.3 a: Practical Setup with JTAG CABLE



[redacted]
[redacted]

Fig. 7.3.b: Lab Practical Setup

b) Create Demux .vhd file:

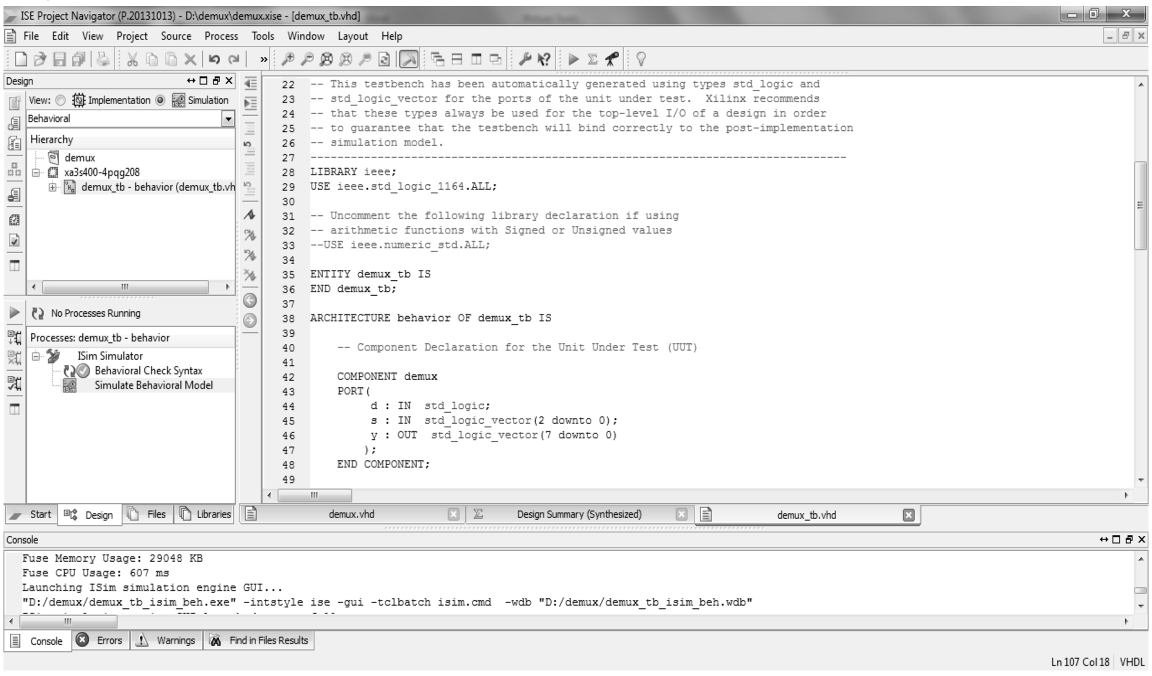


```
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

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

entity Demux is
    Port ( d : in  STD_LOGIC;
           sel : in  STD_LOGIC_VECTOR (2 downto 0);
           y : out  STD_LOGIC_VECTOR (7 downto 0));
end Demux;

architecture Behavioral of Demux is

begin
y(0)<=D when sel="000"else'0';
y(1)<=D when sel="001"else'0';
y(2)<=D when sel="010"else'0';
y(3)<=D when sel="011"else'0';
y(4)<=D when sel="100"else'0';
y(5)<=D when sel="101"else'0';
y(6)<=D when sel="110"else'0';
y(7)<=D when sel="111"else'0';


end Behavioral;
```

```
==========================================================================

Process "Synthesize - XST" completed successfully
```

Fig 7.4: Demux .vhd file

c) Create Demux test bench file:

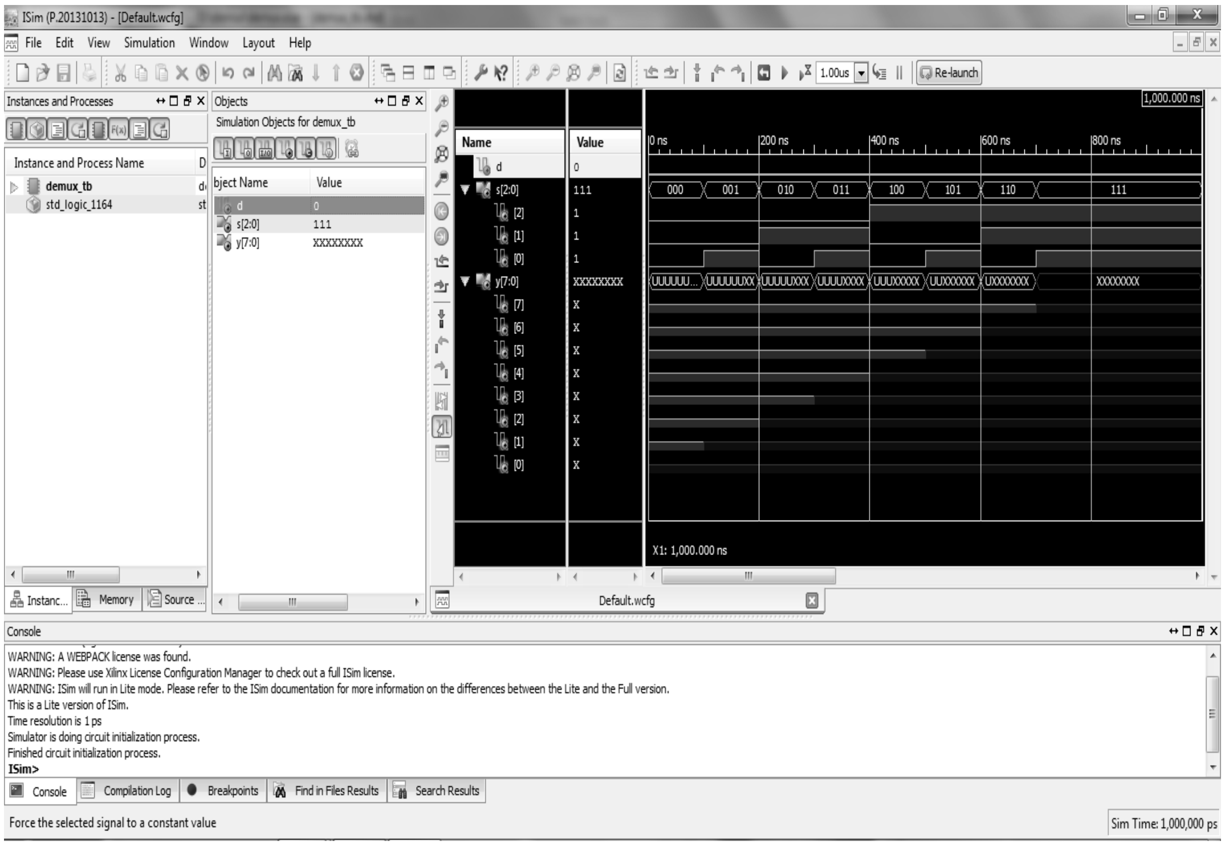


```
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

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

ENTITY demux_tb IS
END demux_tb;

ARCHITECTURE behavior OF demux_tb IS

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

    COMPONENT demux
    PORT(
         d : IN  std_logic;
         s : IN  std_logic_vector(2 downto 0);
         y : OUT  std_logic_vector(7 downto 0)
        );
    END COMPONENT;

```

```
Fuse Memory Usage: 29048 KB
Fuse CPU Usage: 607 ms
Launching ISim simulation engine GUI...
"D:/demux/demux_tb_isim_beh.exe" -intstyle ise -gui -tclbatch isim.cmd  -wdb "D:/demux/demux_tb_isim_beh.wdb"
```

Fig 7.5: Demux Test Bench file

d) Demux test bench Simulation waveforms:



```
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Force the selected signal to a constant value

Sim Time: 1,000,000 ps

Fig 7.6 Demux Simulation output

e) Actual Experimental set up used in laboratory:

IX 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	Loaded with open source IDE, simulation and program downloading software.	1 No.

X Precautions to be followed:

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

XI Procedure:

1. Create the Xilinx ISE project for your top-level FPGA design, by doing the following in ISE:
   In the ISE software, select File > New Project.
   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.
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 Fig. 7.6. where you are able to simulate your designs and check for errors.
10. After simulation implement 1:8 demux using FPGA development board.
11. 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 Config. 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: Implement 1:8 Demultiplexer using behavioral model.
1:8Demultiplexer:

VHDL Code

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

entity dmux1 is
port(D:in std_logic;
s:in std_logic_vector(2 downto 0);
y:out std_logic_vector(7 downto 0));
end demux1;

architectural behavioral of dmux1 is
begin
y(0)<=D when s="000"else'0';
y(1)<=D when s="001"else'0';
y(2)<=D when s="010"else'0';
y(3)<=D when s="011"else'0';
y(4)<=D when s="100"else'0';
y(5)<=D when s="101"else'0';
y(6)<=D when s="110"else'0';
y(7)<=D when s="111"else'0';
end behavioral;
```

Program Statement for Student: Implement 1:4 Demultiplexer using dataflow model.

<u>1:4 Demultiplexer</u>

VHDL Code

XII Resources Used:

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

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

....

....

....

....

....

XIV Precautions followed:

....

....

....

....

....

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

Observations for Problem Statement given to Student.

S0	S1	Y3	Y2	Y1	Y0
0	0				
0	1				
1	0				
1	1				

XVI Results (output of Program)

....

....

....

....

....

XVII Interpretation of result

....

....

....

....

....

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

....

....

....

....

....

XIX 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 when-else statement.
2. Write the VHDL code for 1:4 demultiplexer using the ‘if else” statement.

[Space for Answers]

....

....

....

....

....

....

XX References / Suggestions for further reading

1 https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/
2 https://www.tutorialspoint.com/vlsi_design/vlsi_design_vhdl_introduction.htm
3 https://www.tutorialspoint.com/vlsi_design/vhdl_programming_for_combinational_circuits.htm
4 https://www.rfwireless-world.com/source-code/VHDL/1X8-DEMUX-vhdl-code.html
5 http://ece2day.blogspot.com/2012/10/vhdl-tutorial-introducction-part-7.html
6 https://www.allaboutcircuits.com/technical-articles/sequential-vhdl-if-and-case-statements/
7 http://vhdlbynaresh.blogspot.com/2013/07/design-of-1-8-demultiplexer-using-when.html

XXI 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 of report Answer to sample questions.	05%
	Total (25 Marks)	**100%**

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 08: Implement T and D flip flop using FPGA.

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 Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronic circuits and equipments."**

- Use relevant VHDL model for the given applications.
- Debug VHDL programme for the given application.

IV Relevant Course Outcome

Debug VHDL programme for the given application.

V Practical Outcome

Implement T and D flip flop using FPGA.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum 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 holded this circuit is known as LATCH.

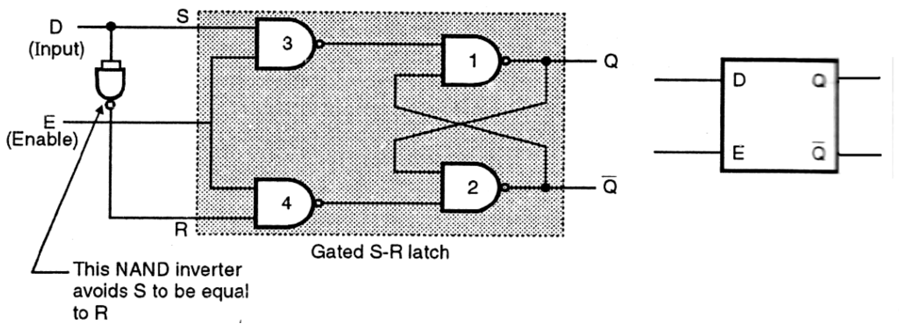


Fig 8.1: D Flip Flop

Table 8.1: Truth Table for D Flip flop

Input D	**Output Q**
0	0
1	1

<u>T Type FF</u>: 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

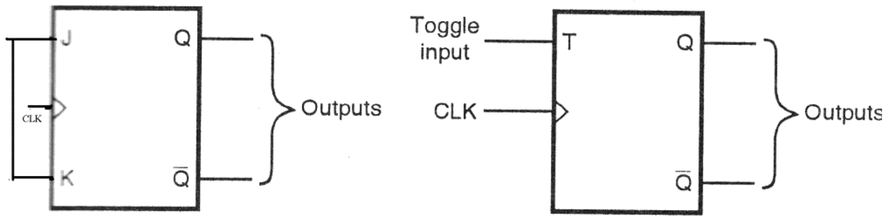


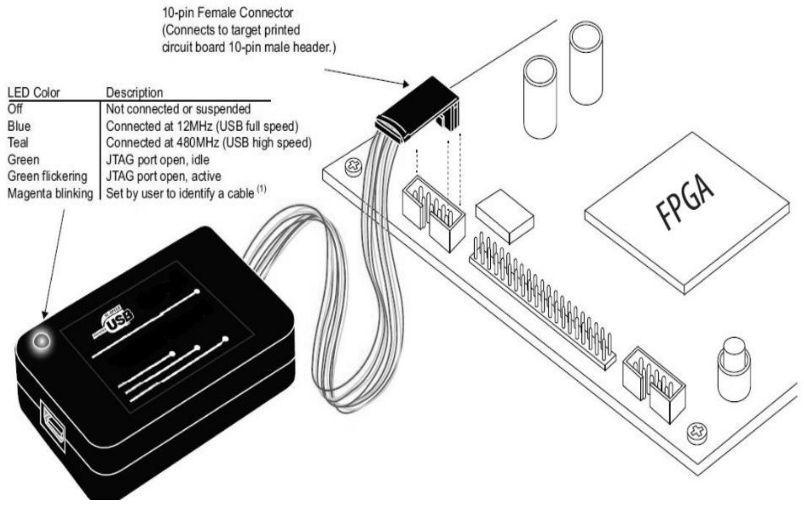
Fig 8.2: T Flip Flop

Table 8.2: Truth Table for T Flip Flop

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

VIII Practical Circuit Diagram:

a) Practical JTAG cable setup:-


10-pin Female Connector
(Connects to target printed circuit board 10-pin male header.)
USB
FPGA

LED Color	Description
Off	Not connected or suspended
Blue	Connected at 12MHz (USB full speed)
Teal	Connected at 480MHz (USB high speed)
Green	JTAG port open, idle
Green flickering	JTAG port open, active
Magenta blinking	Set by user to identify a cable (1)

Fig 8.3 a: Practical Setup with JTAG CABLE

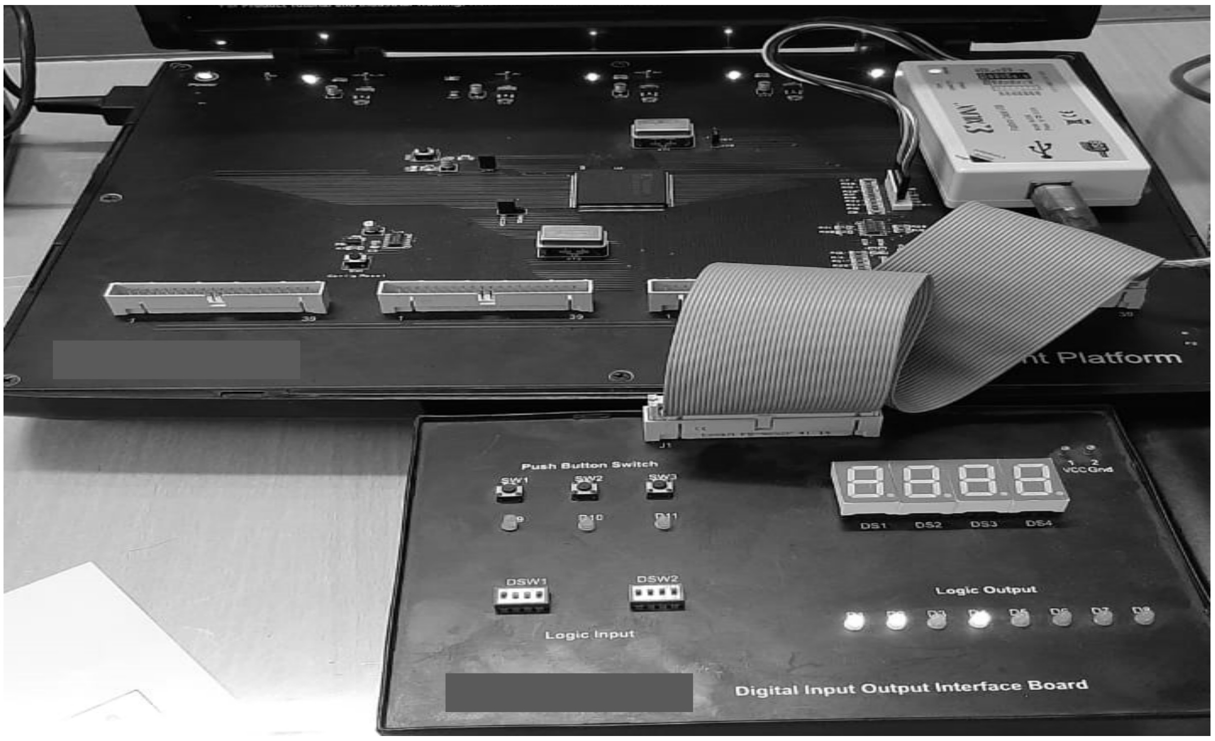
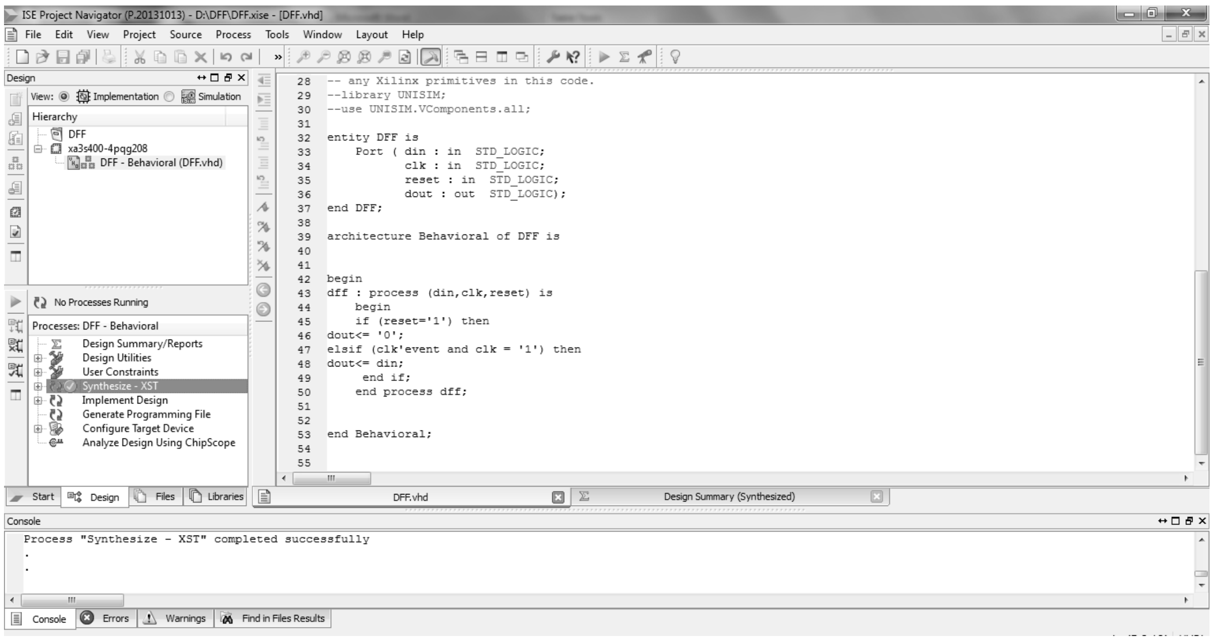


Fig 8.3.b: Lab Practical Setup

b) Create DFF .vhd file:

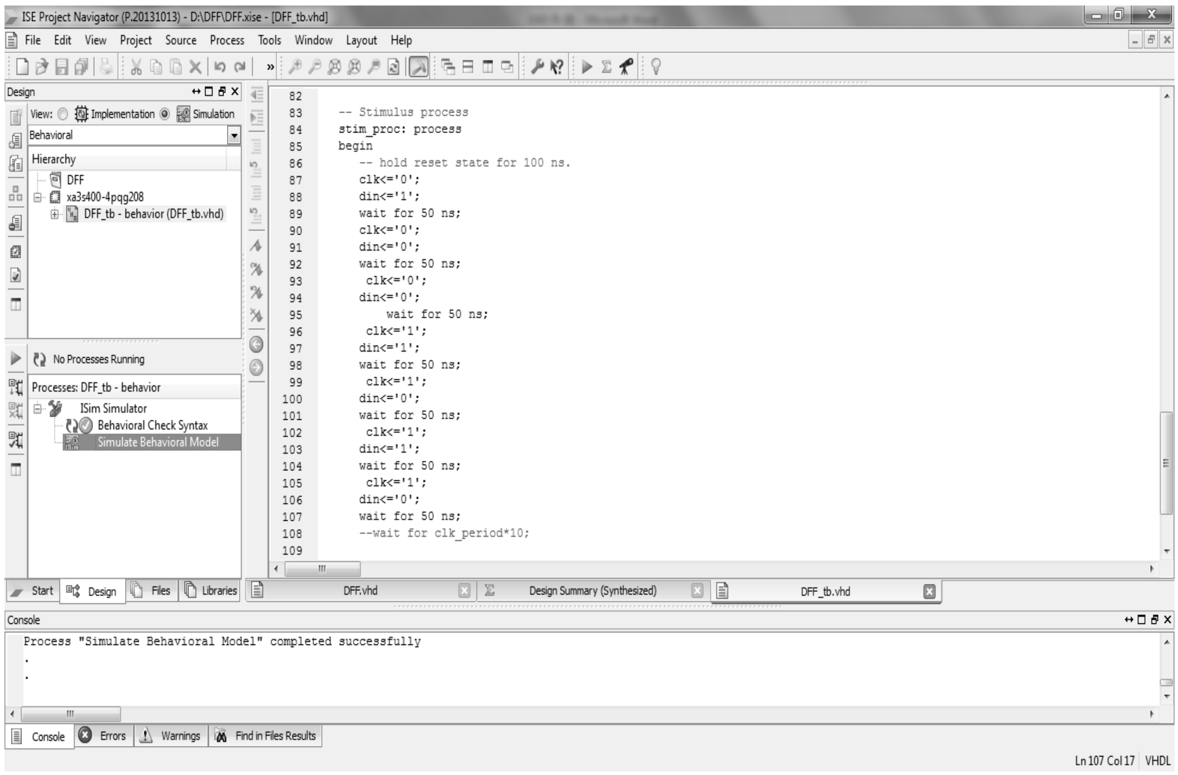

ISE Project Navigator (P.20131013) - D:\DFF\DFF.xise - [DFF.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
DFF
xa3s400-4pqg208
DFF - Behavioral (DFF.vhd)
No Processes Running
Processes: DFF - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start
Design
Files
Libraries
DFF.vhd
Design Summary (Synthesized)
Console
Errors
Warnings
Find in Files Results

```
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity DFF is
    Port ( din : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           dout : out  STD_LOGIC);
end DFF;

architecture Behavioral of DFF 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 Behavioral;
```

```
Process "Synthesize - XST" completed successfully
.
.
```

Fig 8.4: DFF .vhd file

c) Create DFF test bench file:

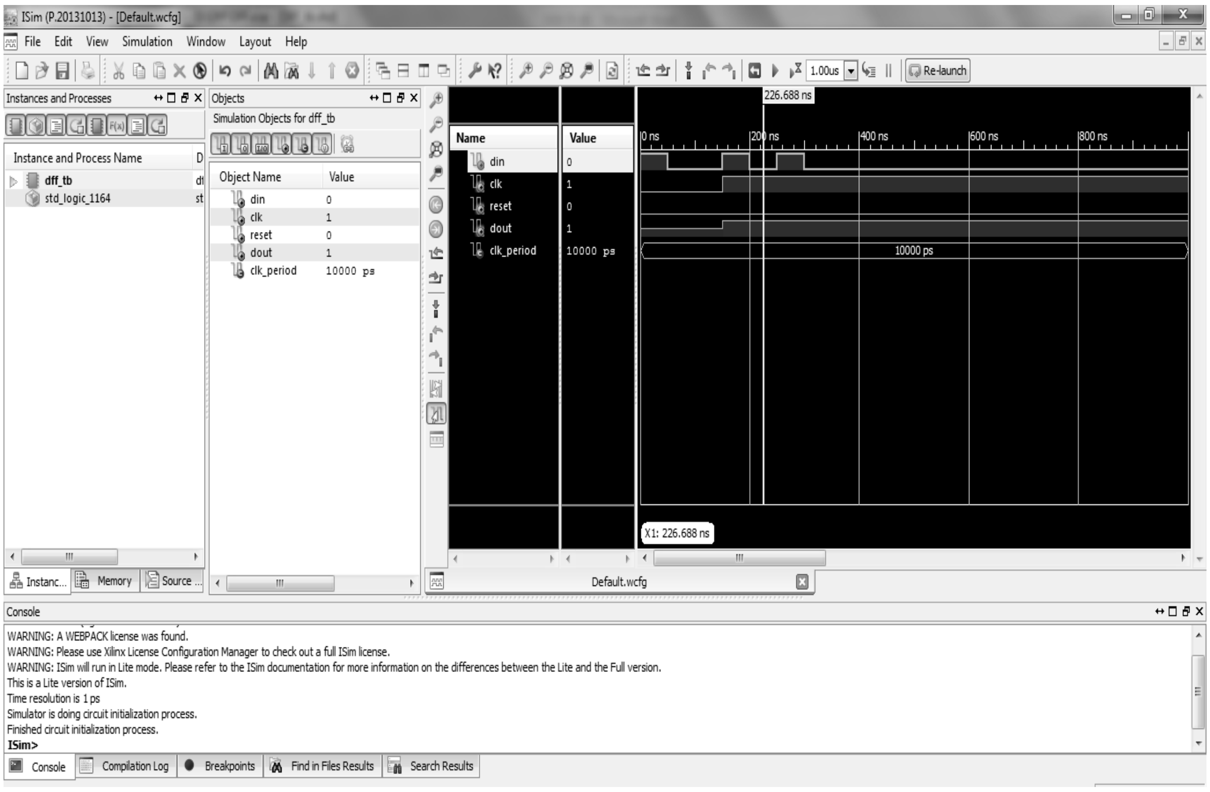

ISE Project Navigator (P.20131013) - D:\DFF\DFF.xise - [DFF_tb.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
DFF
xa3s400-4pqg208
DFF_tb - behavior (DFF_tb.vhd)
No Processes Running
Processes: DFF_tb - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start
Design
Files
Libraries
DFF.vhd
Design Summary (Synthesized)
DFF_tb.vhd
Console
Errors
Warnings
Find in Files Results
Ln 107 Col 17
VHDL

```

   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      clk<='0';
      din<='1';
      wait for 50 ns;
      clk<='0';
      din<='0';
      wait for 50 ns;
       clk<='0';
      din<='0';
          wait for 50 ns;
       clk<='1';
      din<='1';
      wait for 50 ns;
       clk<='1';
      din<='0';
      wait for 50 ns;
       clk<='1';
      din<='1';
      wait for 50 ns;
       clk<='1';
      din<='0';
      wait for 50 ns;
      --wait for clk_period*10;
```

```
Process "Simulate Behavioral Model" completed successfully
.
.
```

Fig 8.5: DFF Test Bench file

d) DFF test bench Simulation waveforms:



Instance and Process Name
dff_tb
std_logic_1164

Object Name	Value
din	0
clk	1
reset	0
dout	1
clk_period	10000 ps

```
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Fig 8.6 DFF Simulation output

e) Actual Experimental set up used in laboratory:

IX 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	Loaded with open source IDE, simulation and program downloading software.	1 No.

X Precautions to be followed:

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

XI Procedure:

1. Create the Xilinx ISE project for your top-level FPGA design, by doing the following in ISE:
   In the ISE software, select File > New Project.
   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.
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 Fig. 8.6. where you are able to simulate your designs and check for errors.
10. After simulation implement D flip flop using FPGA development board.
11. 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 Config 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: Implement T and D flip flop using FPGA using behavioral modelling.
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;
```

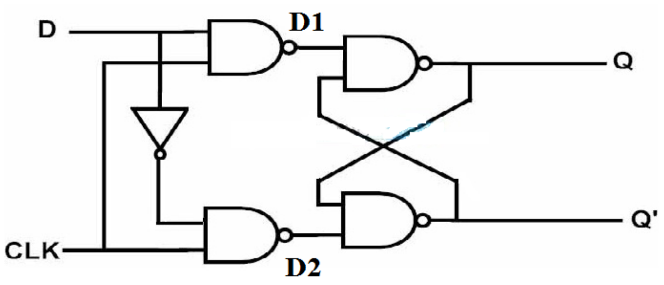
D Flip Flop:

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



VHDL Code

XII Resources Used:

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

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

....
....
....
....
....

XIV Precautions followed:

....
....
....
....
....

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

Observations for Problem Statement given to Student.

Clk	D	Y
0	0	
1	0	
1	1	

XVI Results (output of Program)

....
....
....
....
....
....
....

XVII Interpretation of result

....
....
....
....
....
....
....

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

....
....
....
....
....

XIX 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 modelling.
2. Write the VHDL test bench code for T Flip Flop using characteristic equation.

[Space for Answers]

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

XX References / Suggestions for further reading

1. https://buzztech.in/vhdl-modelling-styles-behavioral-dataflow-structural/
2. https://www.tutorialspoint.com/vlsi_design/vlsi_design_vhdl_introduction.htm
3. https://www.tutorialspoint.com/vlsi_design/vhdl_programming_for_sequential_circuits.htm
4. http://ece2day.blogspot.com/2012/10/vhdl-tutorial-introducction-part-7.html
5. https://electronicstopper.blogspot.com/2017/07/t-flip-flop-in-vhdl-with-testbench.html

XXI 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 of report Answer to sample questions.	05%
	Total (25 Marks)	**100%**

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 9: Implement 2:4 Decoder using FPGA.

I Practical Significance

The electronic Decoders are used with analog to digital converter. They are used to convert higher level instructions into CPU control signals. They are mainly used in logical circuits as well as data transfer circuits. This practical will help the students to develop programming skills that is. Implement a decoder application on FPGA board using Xilinx ISE tools.

II Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronic circuits and equipments.'**

- Develop 'VHDL' code using IDE tools.
- Use Input/output port pins of FPGA.
- Interface FPGA KIT and desktop PC.

IV Relevant Course Outcome(s)

Debug VHDL programme for the given application.

V Practical Outcome

Implement 2:4 Decoder using FPGA.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum Theoretical Background

- **Binary decoder :-** Binary decoder has n-bit input lines and 2^n output lines. It can be 2:4, 3:8 and 4:16 line configurations. Binary decoder can be easily constructed using basic logic gates. VHDL Code of 2 to 4 decoder can be implemented with structural and behavioral modeling.

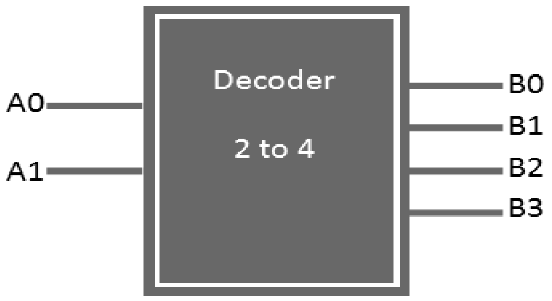
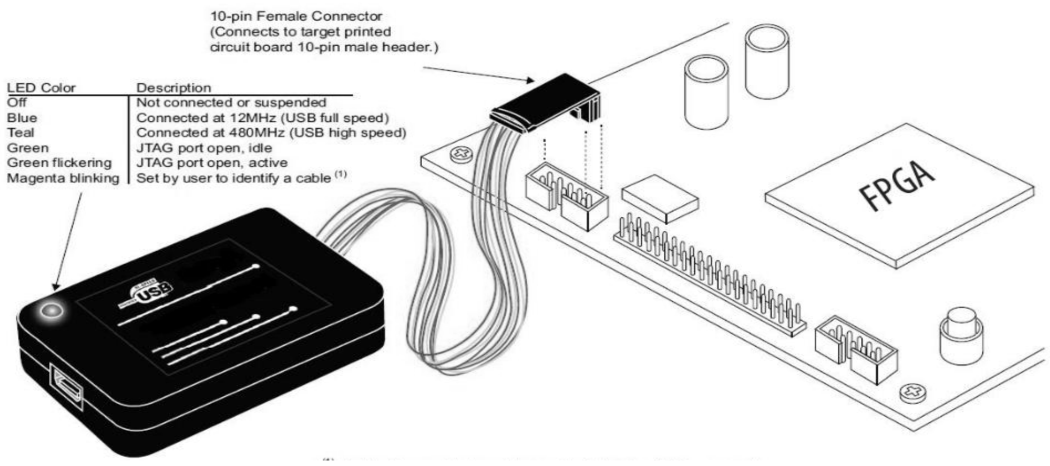


Fig. 9.1: 2 : 4 Decoder

Table No. 9.1: Truth Table for 2 : 4 Decoder

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

VIII Practical Circuit diagram:


10-pin Female Connector
(Connects to target printed
circuit board 10-pin male header.)
USB
FPGA

LED Color	Description
Off	Not connected or suspended
Blue	Connected at 12MHz (USB full speed)
Teal	Connected at 480MHz (USB high speed)
Green	JTAG port open, idle
Green flickering	JTAG port open, active
Magenta blinking	Set by user to identify a cable (1)

Fig. 9.2.a: Practical Setup with JTAG CABLE

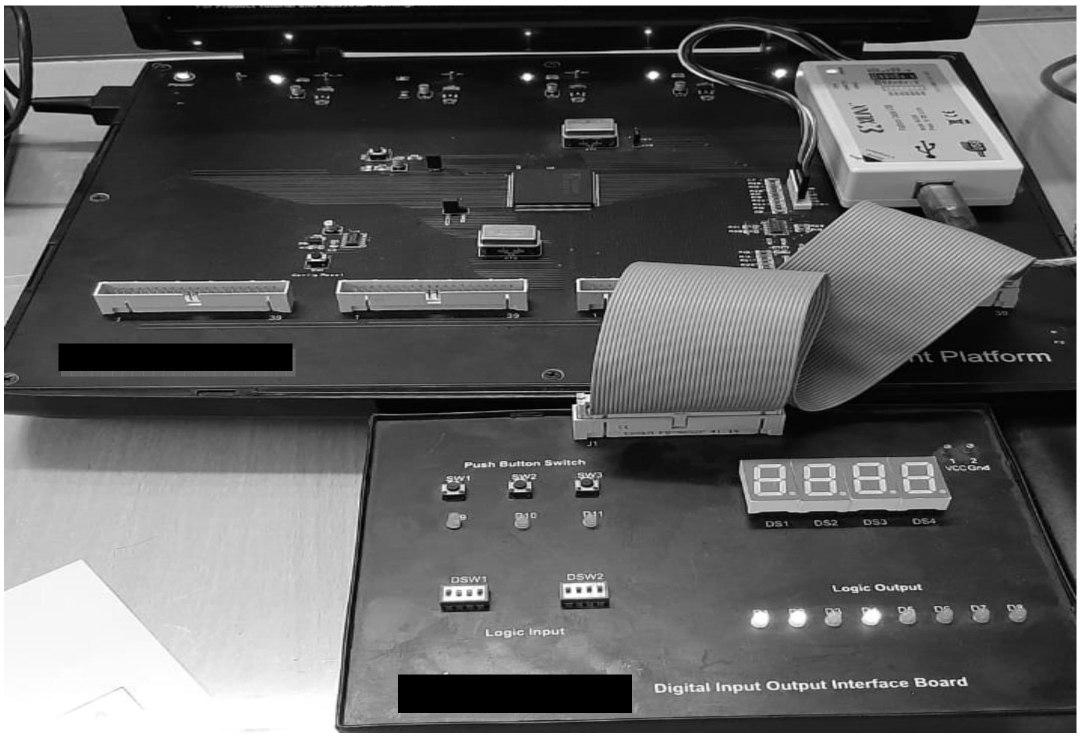
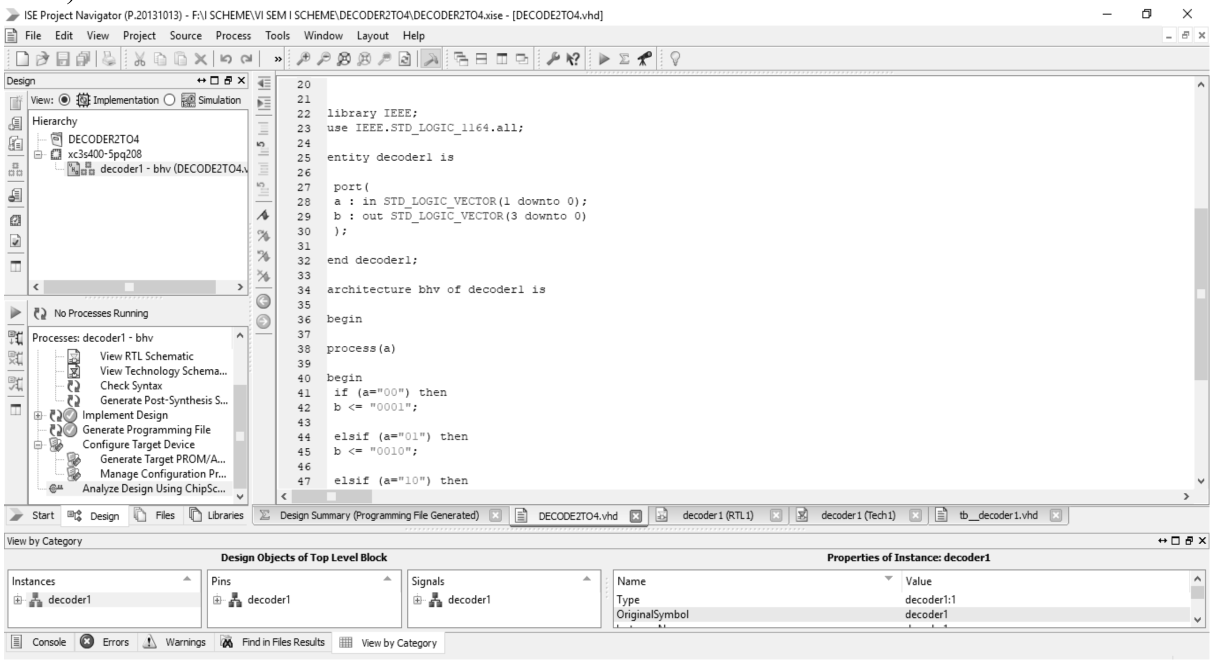

[redacted]
[redacted]

Fig. 9.2.b: Lab Practical Setup

a) Decoder 2 : 4 .vhd file :-


ISE Project Navigator (P.20131013) - F:\I SCHEME\VI SEM I SCHEME\DECODER2TO4\DECODER2TO4.xise - [DECODE2TO4.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
DECODER2TO4
xc3s400-5pq208
decoder1 - bhv (DECODE2TO4.v
No Processes Running
Processes: decoder1 - bhv
View RTL Schematic
View Technology Schema...
Check Syntax
Generate Post-Synthesis S...
Implement Design
Generate Programming File
Configure Target Device
Generate Target PROM/A...
Manage Configuration Pr...
Analyze Design Using ChipSc...
Start
Design
Files
Libraries
Design Summary (Programming File Generated)
DECODE2TO4.vhd
decoder1 (RTL1)
decoder1 (Tech1)
tb__decoder1.vhd
View by Category
Console
Errors
Warnings
Find in Files Results
View by Category

```


library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity decoder1 is

 port(
 a : in STD_LOGIC_VECTOR(1 downto 0);
 b : out STD_LOGIC_VECTOR(3 downto 0)
 );

end decoder1;

architecture bhv of decoder1 is

begin

process(a)

begin
 if (a="00") then
 b <= "0001";

 elsif (a="01") then
 b <= "0010";

 elsif (a="10") then
```

Design Objects of Top Level Block

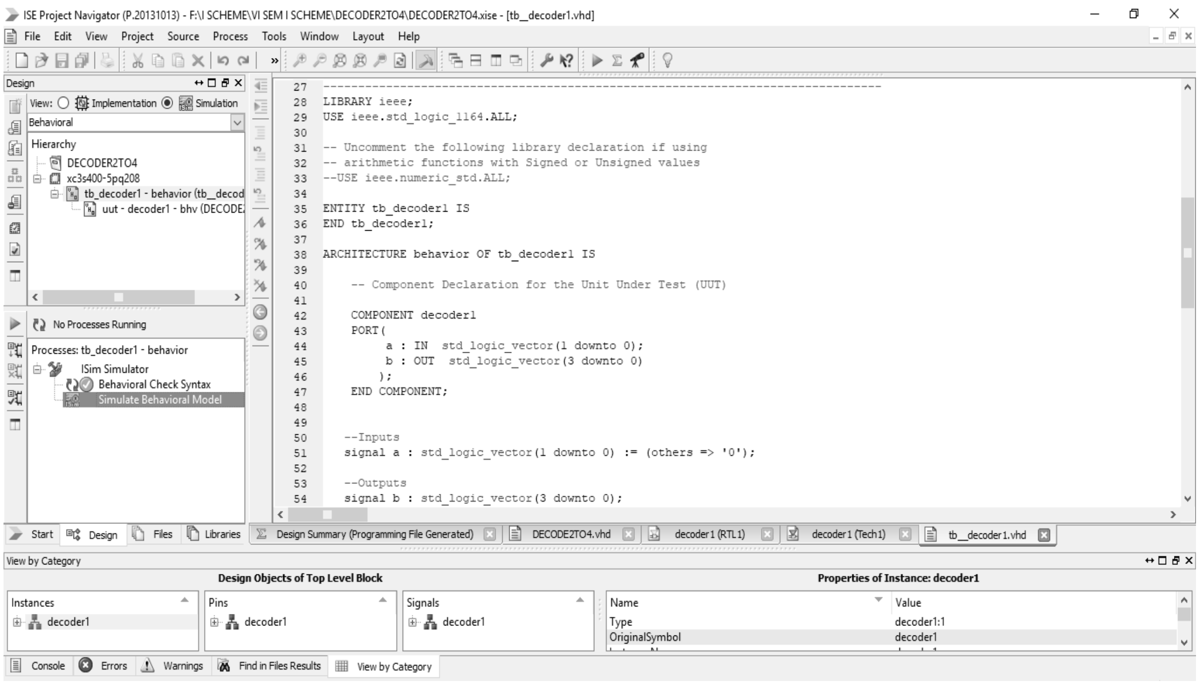
Instances	Pins	Signals
decoder1	decoder1	decoder1

Properties of Instance: decoder1

Name	Value
Type	decoder1:1
OriginalSymbol	decoder1

Fig. 9.3: .vhd file

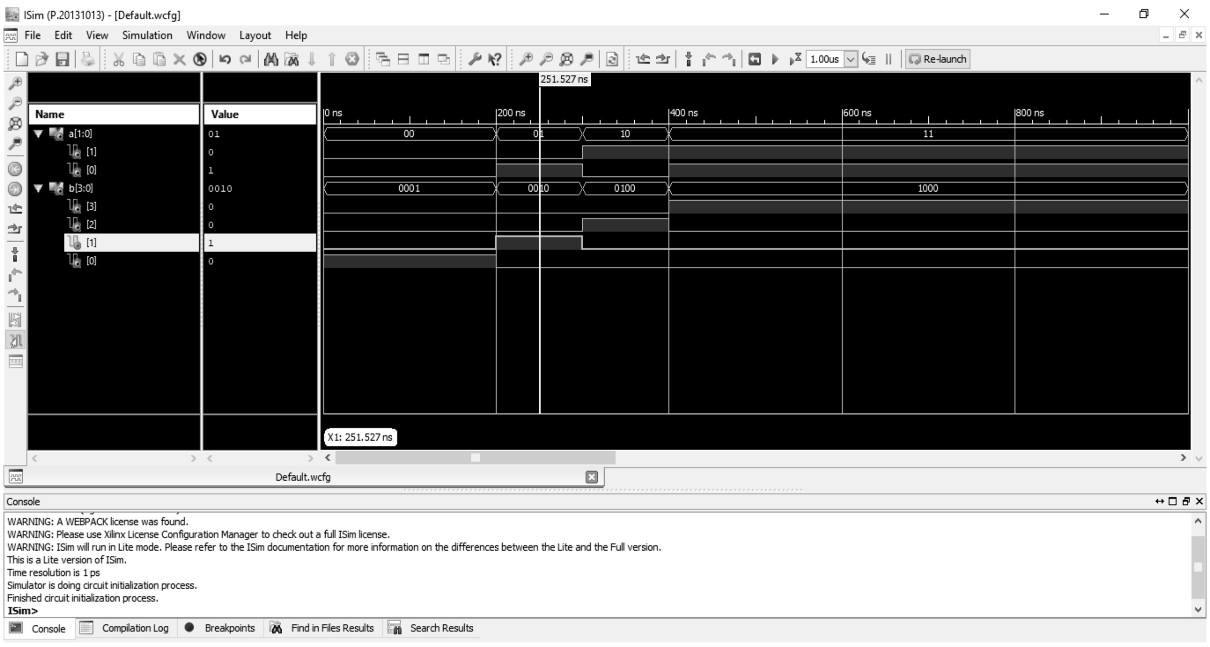
b) Decoder 2 : 4 test bench file-



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

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

ENTITY tb_decoder1 IS
END tb_decoder1;

ARCHITECTURE behavior OF tb_decoder1 IS

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

    COMPONENT decoder1
    PORT(
         a : IN  std_logic_vector(1 downto 0);
         b : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal a : std_logic_vector(1 downto 0) := (others => '0');

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

Fig. 9.4: Test bench file

c) Decoder 2 to 4 test bench Simulation waveforms-



```
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Fig. 9.5: Simulation output

d) Actual Experimental set up used in laboratory

IX 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	Loaded with open source IDE, simulation and program downloading software.	1 No.

X Precautions to be followed

1. Use proper Mains cord.
2. To avoid fire or shock hazards, observe all ratings and marks on the instrument.

XI Procedure

1. Create the Xilinx ISE project for your top-level FPGA design, by doing the following in ISE:
   In the ISE software, select File > New Project.
   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.9.3.
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. Ref Fig no. 9.4.
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 "Behavioural Check Syntax" to make sure that you 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 your test bench loaded.
9. ISim simulator window will open with your simulation executed, as shown in Ref Fig no. 9.5. where you are able to simulate your designs and check for errors.
10. After simulation implement 2 : 4 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 File and Run
14. Go to Config. 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 2:4 Decoder .

VHDL Code using if else statement.

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

entity decoder1 is
 port(
 a : in STD_LOGIC_VECTOR(1 downto 0);
 b : out STD_LOGIC_VECTOR(3 downto 0)
 );
end decoder1;

architecture bhv of decoder1 is
begin
        process(a)

        begin
        if (a="00") then
        b <= "0001";
        elsif (a="01") then
        b <= "0010";
        elsif (a="10") then
        b <= "0100";
        else
        b <= "1000";
        end if;
        end process;
end bhv;
```

Step 2: Develop Test Bench file for simulation.

VHDL Code

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

ENTITY tb_decoder1 IS
END tb_decoder1;

ARCHITECTURE behavior OF tb_decoder1 IS

   -- Component Declaration for the Unit Under Test (UUT)
   COMPONENT decoder1
   PORT(
        a : IN  std_logic_vector(1 downto 0);
        b : OUT  std_logic_vector(3 downto 0)
       );
   END COMPONENT;
```

```
           --Inputs
   signal a : std_logic_vector(1 downto 0) := (others => '0');
         --Outputs
   signal b : std_logic_vector(3 downto 0);

BEGIN

         -- Instantiate the Unit Under Test (UUT)
   uut: decoder1 PORT MAP (
        a => a,
        b => b
       );

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

         a <= "00";
         wait for 100 ns;
         a <= "01";
         wait for 100 ns;
         a <= "10";
         wait for 100 ns;
         a <= "11";
         wait;
         end process;
END;
```

Problem statement for student: Design VHDL Code for 3 to 8 decoder using if else statement.

Step 1: Develop VHDL code.

VHDL Code using case statement.

Step 2: Develop Test Bench file for simulation.

VHDL Code

XII Resources Used

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

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

....

....

....

....

....

XIV Precautions Followed (use blank sheet provided if space not sufficient)

....

....

....

....

....

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

Truth table of 2 : 4 decoder is _______(verified/ not verified) using FPGA development board.

XVI Result (Output of the Program)

....

....

....

....

....

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

....

....

....

....

....

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

....

....

....

....

....

XIX 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 : 8 decoder using case statement.
2. Design VHDL Code using FPGA board for 4 : 16 using if else statement.
3. Design VHDL Code using FPGA board for 4 : 16 decoder using case statement

[Space for Answers]

....

....

....

....

....

....

XX References / Suggestions for further reading

1. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_usb_blstr_ii_cable.pdf
2. https://www.xilinx.com/support/documentation/data_sheets/ds610.pdf
3. https://www.elprocus.com/designing-3-line-to-8-line-decoder-demultiplexer/
4. https://www.rfwireless-world.com/source-code/VHDL/3-8-decoder-vhdl-code.html

XXI Assessment Scheme

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

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 10: Implement 8:3 Encoder using FPGA.

I Practical Significance

Binary encoder has 2^n input lines and n-bit output lines. It can be 4 : 2, 8 : 3 and 16 : 4 line configurations. VHDL Code for encoder can be designed both in structural and behavioral modeling. This practical will help the students to develop programming skills that is Implement a encoder application on FPGA board using Xilinx ISE tools.

II Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain FPGA based circuits'.**

- Develop 'VHDL' code using IDE tools.
- Use Input/output port pins of FPGA.
- Interface FPGA KIT and desktop PC.

IV Relevant Course Outcome(s)

Develop basic applications using VHDL.

V Practical Outcome

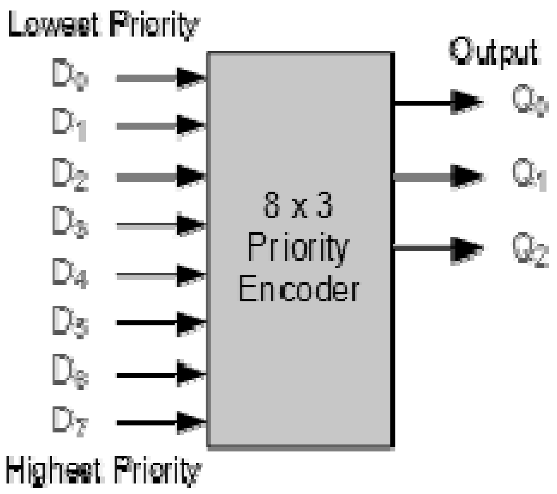
Debug VHDL programme for the given application.

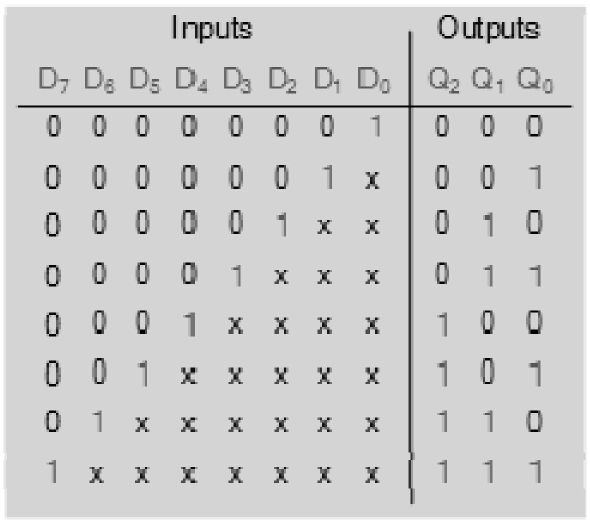
VI Relevant Affective domain related Outcome(s)

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

VII Minimum Theoretical Background

Binary encoder :- Priority Encoder generally, digital encoders produce outputs of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines. An "n-bit" binary encoder has 2^n input lines and n-bit output lines with common types that include 4:2, 8:3 and 16:4 line configurations. VHDL Code of 8:3 encoder can be easily implemented with structural and behavioral modeling.





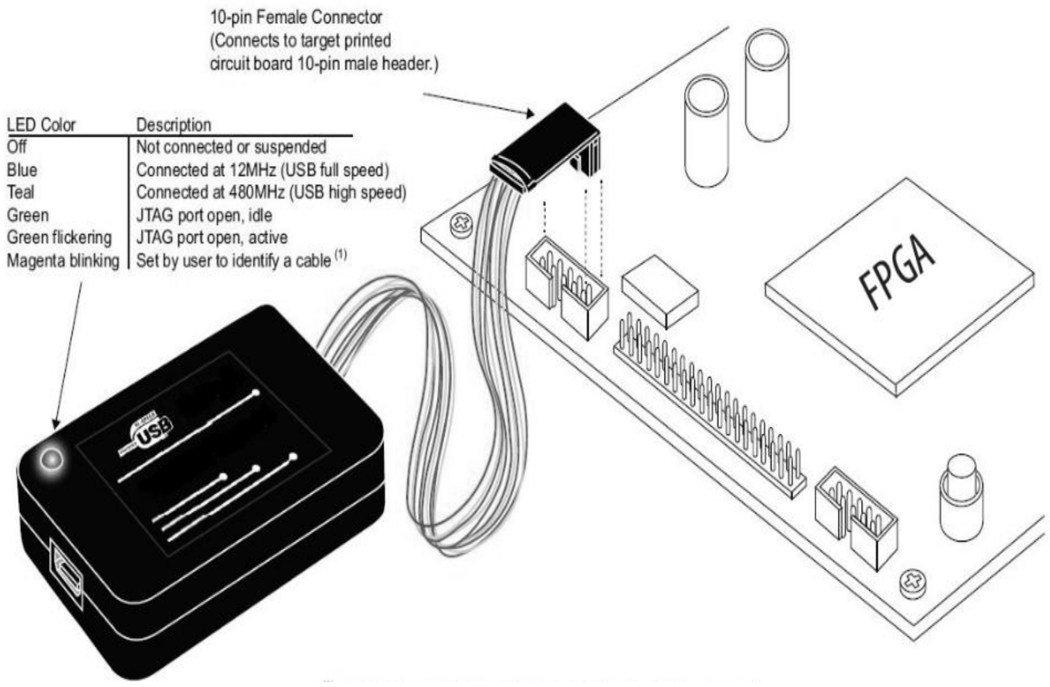
Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

X = dont care

Fig 10.1: 8:3 Priority Encoder and its Truth Table

VIII Practical Circuit diagram:

a) Practical JTAG cable setup:-


10-pin Female Connector
(Connects to target printed circuit board 10-pin male header.)
USB
FPGA

LED Color	Description
Off	Not connected or suspended
Blue	Connected at 12MHz (USB full speed)
Teal	Connected at 480MHz (USB high speed)
Green	JTAG port open, idle
Green flickering	JTAG port open, active
Magenta blinking	Set by user to identify a cable [1]

Fig 10.2.a: Practical Setup with JTAG CABLE

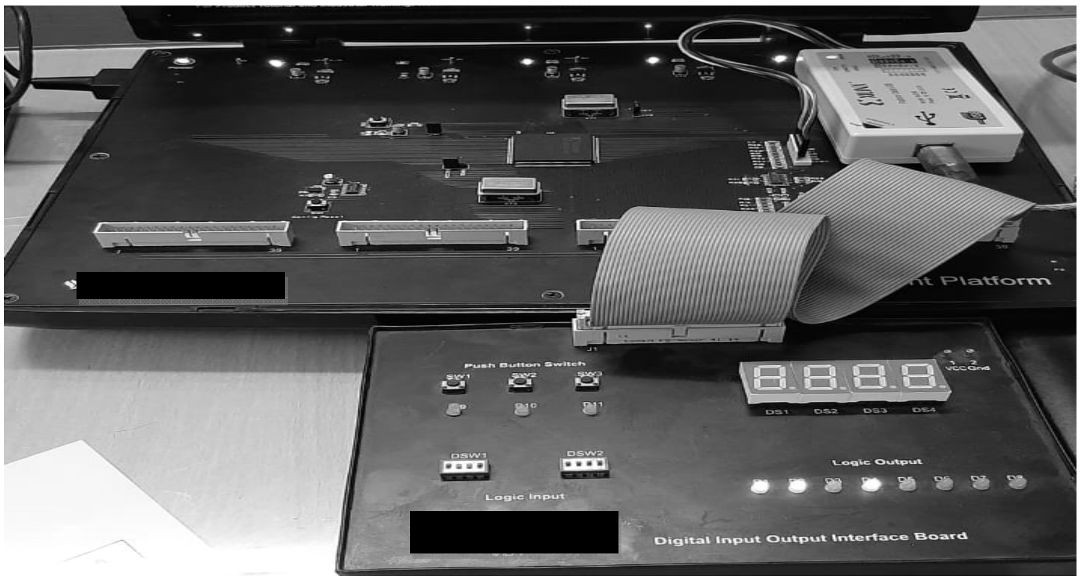
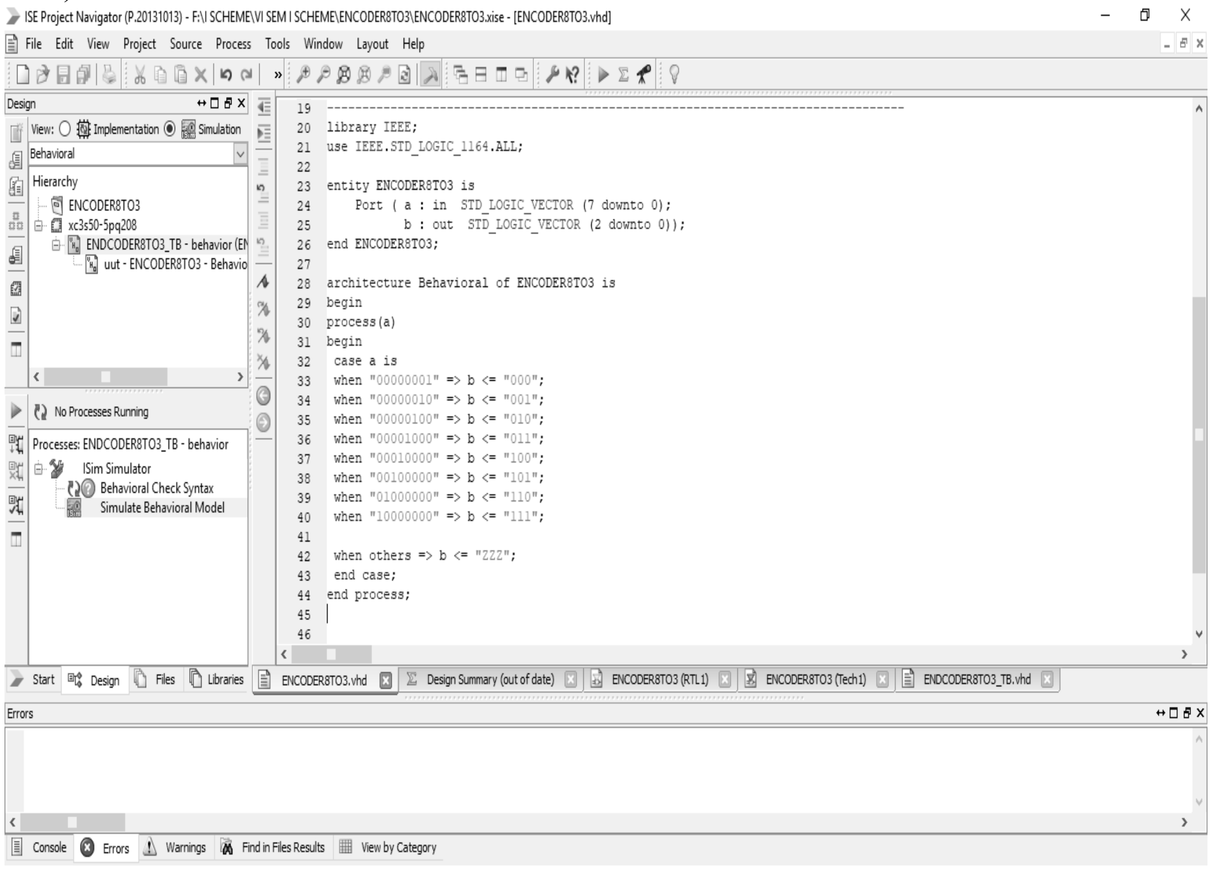


Fig 10.2.b: Lab Practical Setup

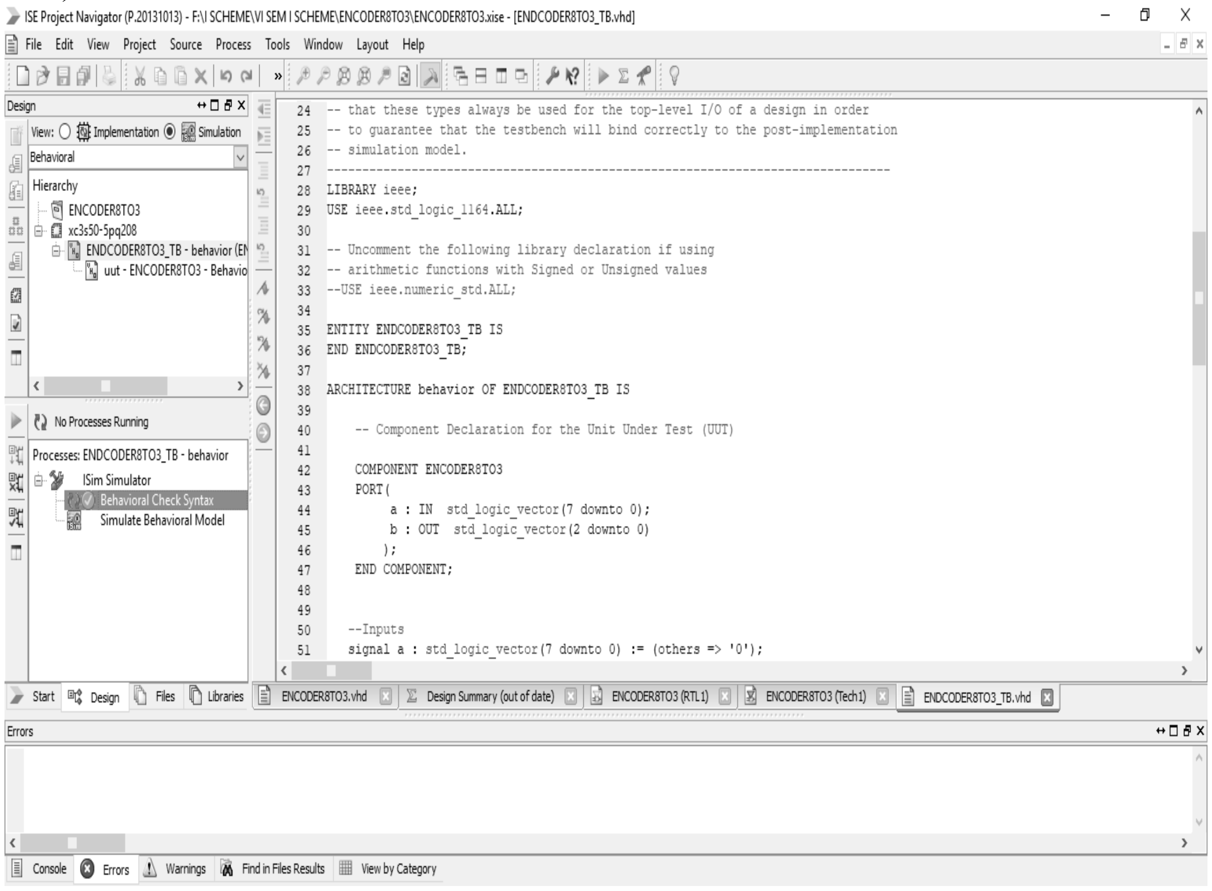
b) Create Encoder 8:3 .vhd file :-


ISE Project Navigator (P.20131013) - F:\I SCHEME\VI SEM I SCHEME\ENCODER8TO3\ENCODER8TO3.xise - [ENCODER8TO3.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
ENCODER8TO3
xc3s50-5pq208
ENDCODER8TO3_TB - behavior
uut - ENCODER8TO3 - Behavioral
No Processes Running
Processes: ENDCODER8TO3_TB - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start
Design
Files
Libraries
ENCODER8TO3.vhd
Design Summary (out of date)
ENCODER8TO3 (RTL1)
ENCODER8TO3 (Tech1)
ENDCODER8TO3_TB.vhd
Errors
Console
Errors
Warnings
Find in Files Results
View by Category

```
--------------------------------------------------------------------------------
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;


```

Fig 10.3: .vhd file

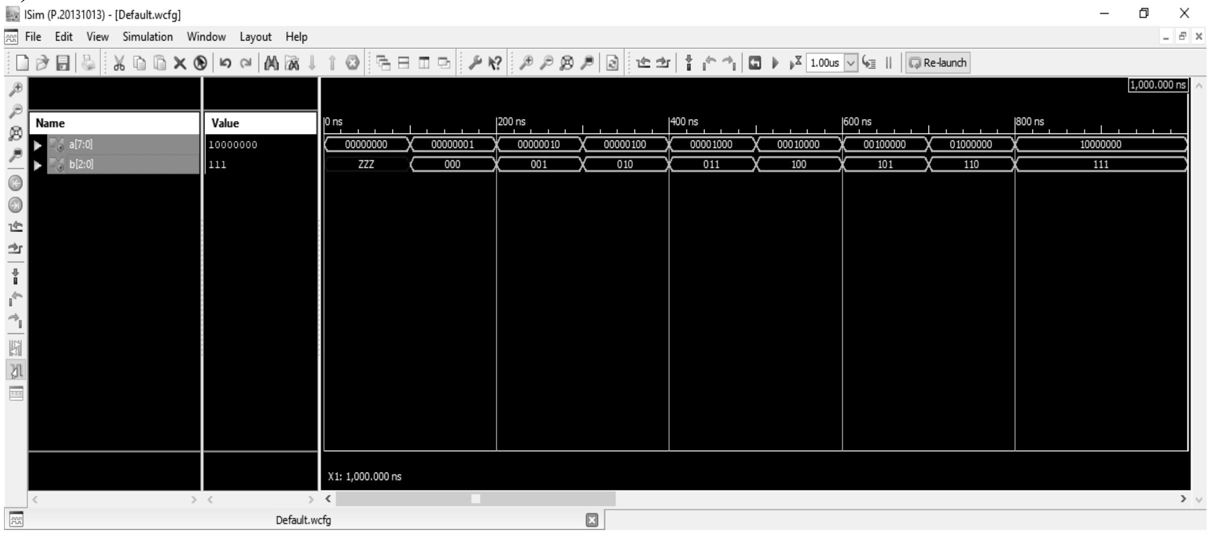
c) Create Encoder 8:3 test bench file-


ISE Project Navigator (P.20131013) - F:\I SCHEME\VI SEM I SCHEME\ENCODER8TO3\ENCODER8TO3.xise - [ENDCODER8TO3_TB.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
ENCODER8TO3
xc3s50-5pq208
ENDCODER8TO3_TB - behavior (EN
uut - ENCODER8TO3 - Behavio
No Processes Running
Processes: ENDCODER8TO3_TB - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start Design Files Libraries
ENCODER8TO3.vhd
Design Summary (out of date)
ENCODER8TO3 (RTL1)
ENCODER8TO3 (Tech1)
ENDCODER8TO3_TB.vhd
Errors
Console Errors Warnings Find in Files Results View by Category

```
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

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

ENTITY 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');
```

Fig 10.4: Test bench file

d) Encoder 8:3 test bench Simulation waveforms-



```
Name      Value
a[7:0]    10000000   00000000 | 00000001 | 00000010 | 00000100 | 00001000 | 00010000 | 00100000 | 01000000 | 10000000
b[2:0]    111        ZZZ      | 000      | 001      | 010      | 011      | 100      | 101      | 110      | 111
```

Fig 10.5: Simulation output.

e) Actual Experimental set up used in laboratory

IX 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	Loaded with open source IDE, simulation and program downloading software.	1 No.

X 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.

XI Procedure

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

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 10.3.
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. Ref Fig no.10.4.
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 "Behavioural Check Syntax" to make sure that you 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 your test bench loaded.
9. ISim simulator window will open with your simulation executed, as shown in Ref Fig no.10.5. where you are able to simulate your designs and check for errors.
10. After simulation implement 8:3 encoder 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 Config.. 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 8:3 Encoder

VHDL Code using Case 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 8 to 3 encoder using if else statement.

Step 1: Develop VHDL code.

VHDL Code using if else statement

Step 2: Develop Test Bench file for simulation.

VHDL Code

XII Resources Used

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

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

....
....
....
....
....

XIV Precautions Followed (use blank sheet provided if space not sufficient)

....
....
....
....

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

Truth table of 8:3 encoder is _______(verified/ not verified) using FPGA development board.

XVI Result (Output of the Program)

....
....
....
....
....

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

....
....
....
....
....

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

....
....
....
....
....

XIX 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 8:3 priority encoder using when statement.
2. Design VHDL Code using FPGA board for 16:4 priority encoder using if else statement.

[Space for Answers]

....
....
....
....
....
....

XX References / Suggestions for further reading

1. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_usb_blstr_ii_cable.pdf
2. https://forums.xilinx.com/
3. https://www.electronics-tutorials.ws/combination/comb_4.html
4. https://circuitdigest.com/tutorial/encoder-circuit-basics-truth-table-and-explanations
5. https://www.technobyte.org/vhdl-code-for-an-encoder-dataflow/

XXI Assessment Scheme

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

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 11: Implement up-counter using FPGA.

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 (MOD-N) application on FPGA board using Xilinx ISE tools.

II Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronics circuits and equipments.'**

- Develop 'VHDL' code using EDA tools.
- Use Input/output port pins of FPGA.
- Interface FPGA KIT and desktop PC.

IV Relevant Course Outcome(s)

Debug VHDL programme for the given application.

V Practical Outcome

Implement up-counter using FPGA.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum Theoretical Background

An Asynchronous counter can count using Asynchronous clock input. Counters can be easily made using flip-flops. As the count depends on the clock signal, in case of an Asynchronous counter, changing state bits are provided as the clock signal to the subsequent flip-flops. Those Flip-flops are serially connected together, and the clock pulse ripples through the counter. Due to the ripple clock pulse, it's often called a ripple counter. An Asynchronous counter can count $2^n - 1$ possible counting states.

Modulo or MOD counters are one of those types of counters. The configuration made in such a way that the counter will reset itself to zero at a pre-configured value and has truncated sequences.

So, if a counter with the specific number of resolutions (n-bit Resolution) count up to is called as full sequence counter and on the other hand, if it is count less than the maximum number, is called as a truncated counter.

To get the advantage of the asynchronous inputs in the flipflop, Asynchronous Truncated counter can be used with combinational logic.

Modulo 16 asynchronous counter can be modified using additional logic gates and can be used in a way that the output will give a decade (divided by 10) counter output, which is useful in counting standard decimal numbers or in arithmetic circuits. This type of counters called as Decade Counters.

Decade Counters requires resetting to zero when the output reaches a decimal value of 10.Following table shows 0-9 (10 steps) the binary number.

Table: Truth table of MOD 10 counter

CLR	CLK ' event	CLK NO.	Q3	Q2	Q1	Q0
1	1	0	0	0	0	0
0	1	1	0	0	0	0
0	1	2	0	0	0	1
0	1	3	0	0	1	0
0	1	4	0	0	1	1
0	1	5	0	1	0	0
0	1	6	0	1	0	1
0	1	7	0	1	1	0
0	1	8	0	1	1	1
0	1	9	1	0	0	0
0	1	10	1	0	0	1

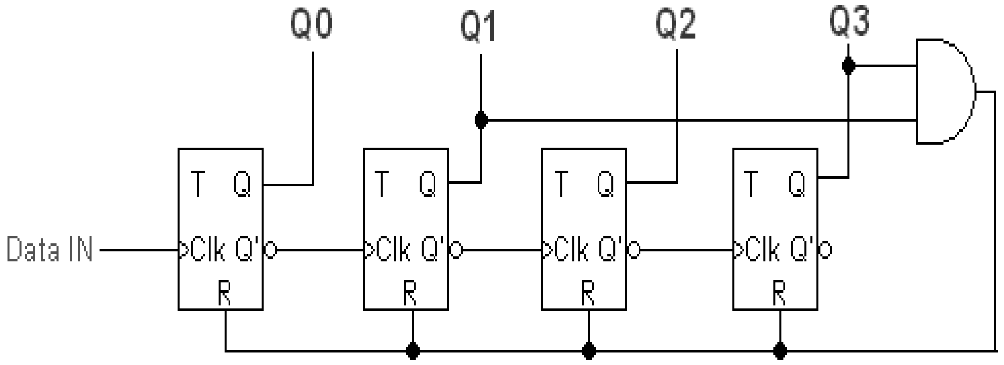


Fig. 11.1.a: Four bit up-counter (MOD-10/ Decade counter)

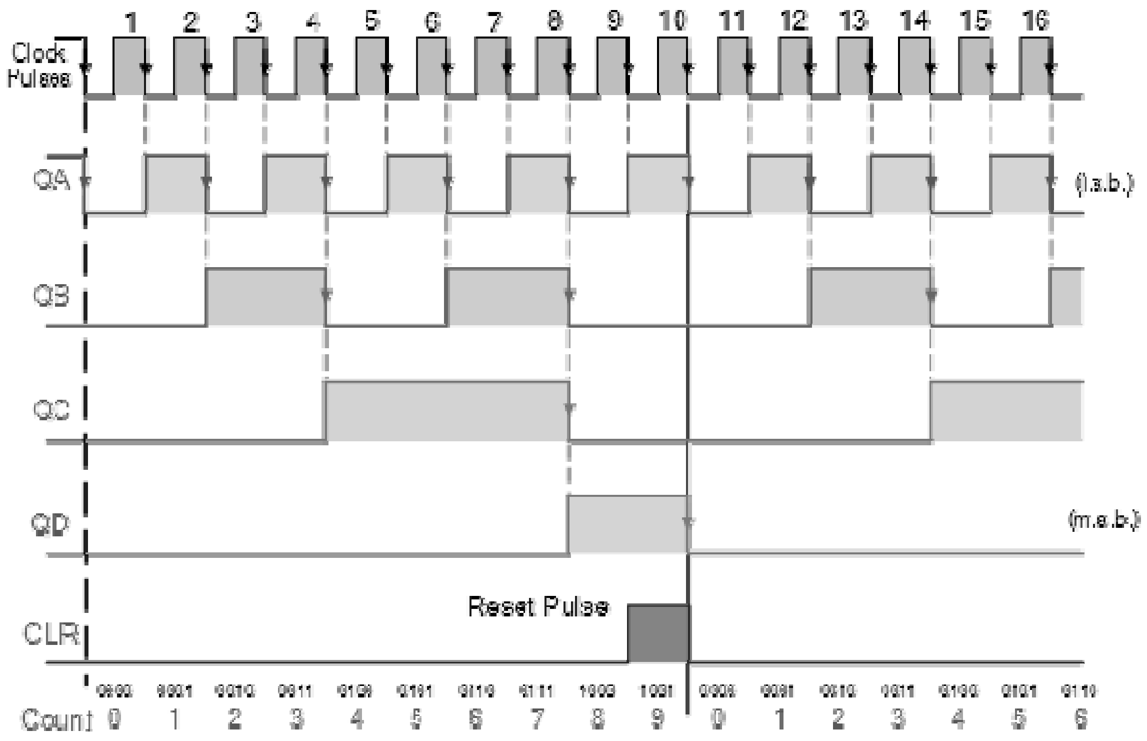
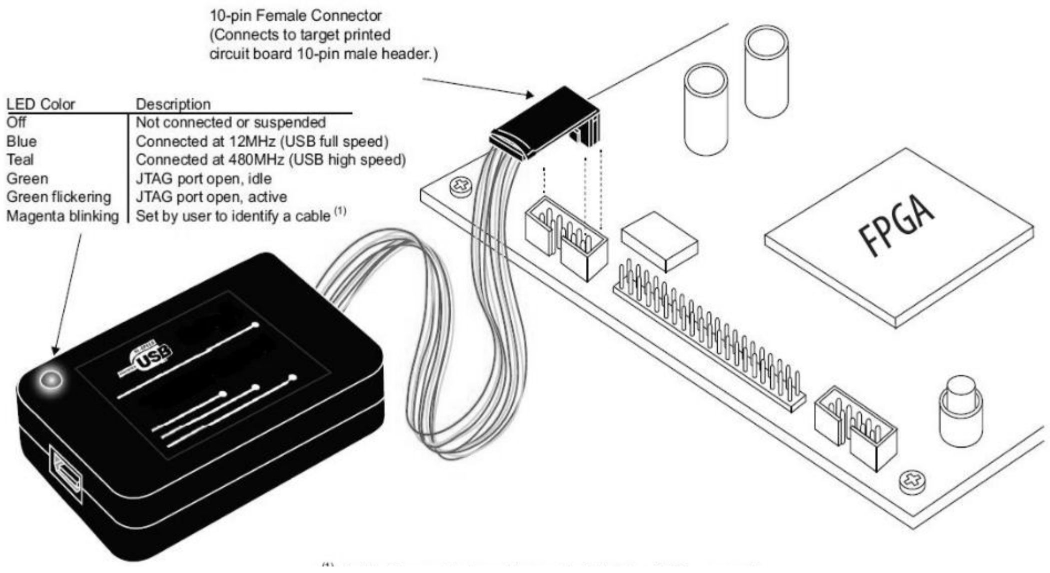


Fig. 11.1.b: Four bit up-counter (MOD-10/ Decade counter) timing waveforms.

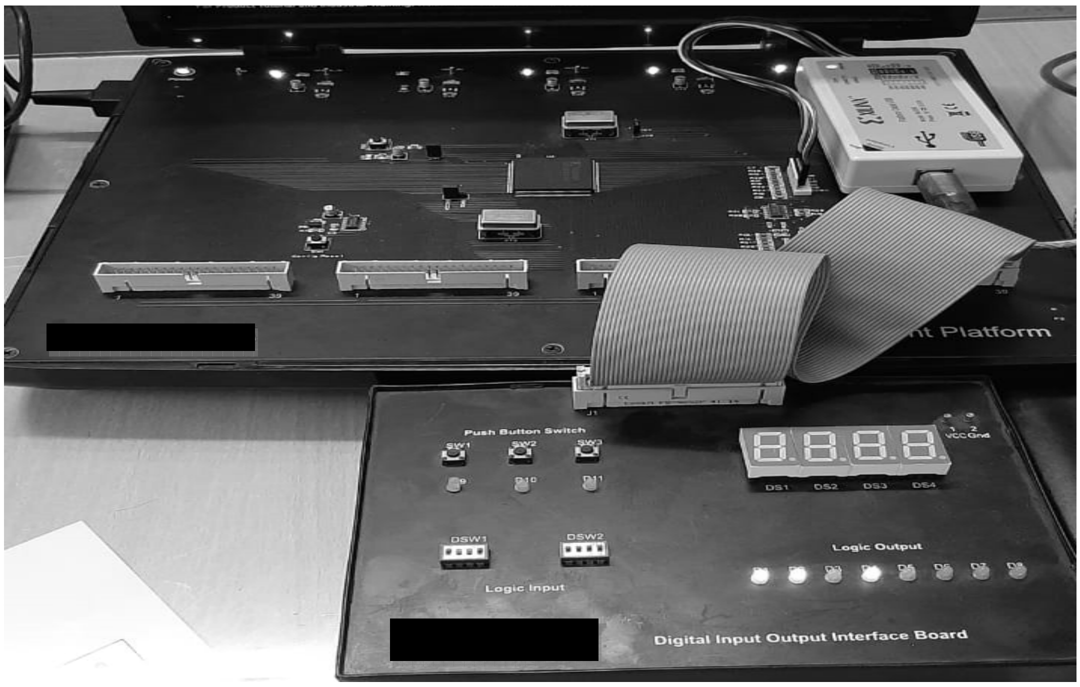
VIII Practical Circuit diagram:

a) Practical JTAG cable setup:-


10-pin Female Connector
(Connects to target printed circuit board 10-pin male header.)
FPGA
USB

LED Color	Description
Off	Not connected or suspended
Blue	Connected at 12MHz (USB full speed)
Teal	Connected at 480MHz (USB high speed)
Green	JTAG port open, idle
Green flickering	JTAG port open, active
Magenta blinking	Set by user to identify a cable [1]

Fig. 11.2.a: Practical Setup with JTAG CABLE

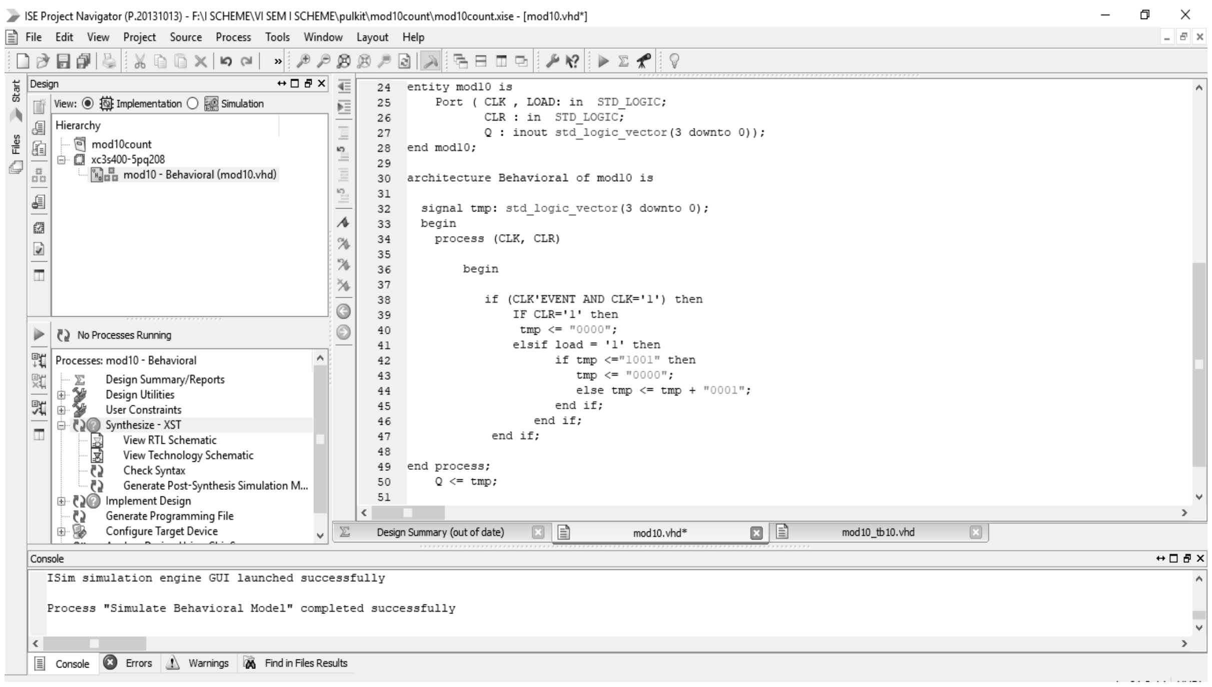


[redacted]

[redacted]

Fig. 11.2.b: Lab Practical Setup

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

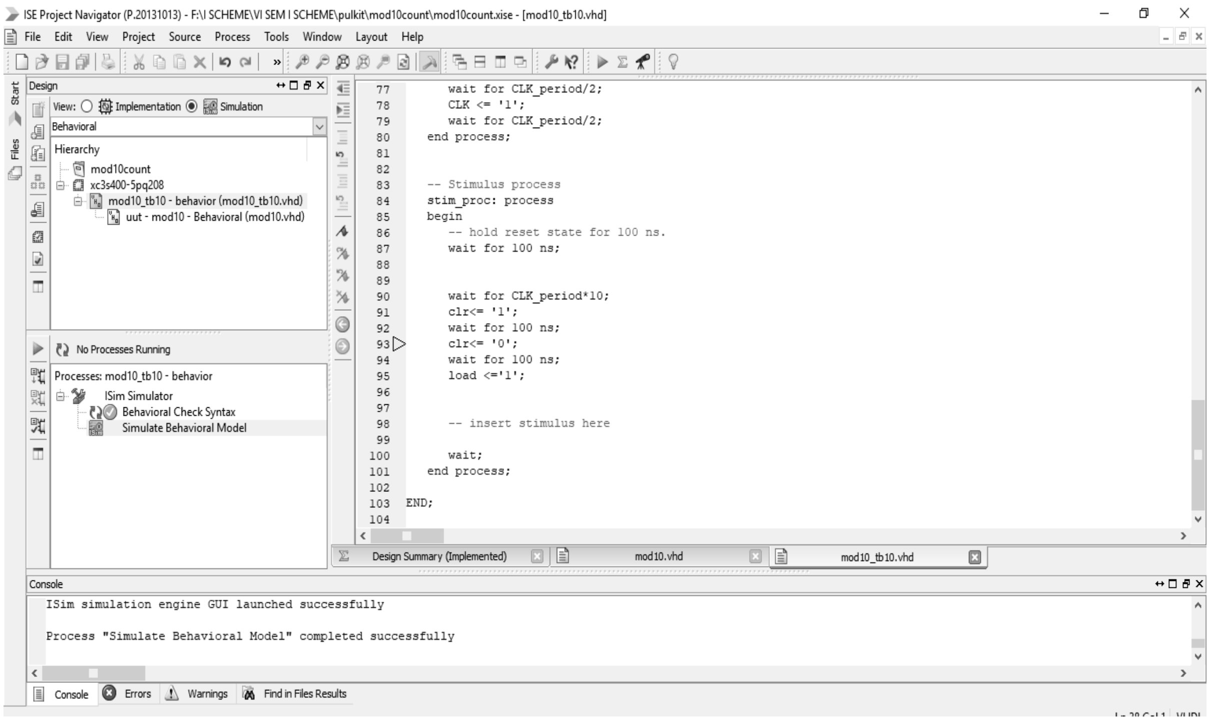

ISE Project Navigator (P.20131013) - F:\I SCHEME\VI SEM I SCHEME\pulkit\mod10count\mod10count.xise - [mod10.vhd*]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
mod10count
xc3s400-5pq208
mod10 - Behavioral (mod10.vhd)
No Processes Running
Processes: mod10 - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation M...
Implement Design
Generate Programming File
Configure Target Device
Design Summary (out of date)
mod10.vhd*
mod10_tb10.vhd
Console
Console
Errors
Warnings
Find in Files Results

```
entity mod10 is
    Port ( CLK , LOAD: in  STD_LOGIC;
           CLR : in  STD_LOGIC;
           Q : inout std_logic_vector(3 downto 0));
end mod10;

architecture Behavioral of mod10 is

  signal tmp: std_logic_vector(3 downto 0);
  begin
    process (CLK, CLR)

        begin

           if (CLK'EVENT AND CLK='1') then
               IF CLR='1' then
                tmp <= "0000";
               elsif load = '1' then
                     if tmp <="1001" then
                        tmp <= "0000";
                        else tmp <= tmp + "0001";
                     end if;
                  end if;
            end if;

end process;
    Q <= tmp;
```

```
ISim simulation engine GUI launched successfully

Process "Simulate Behavioral Model" completed successfully
```

Fig. 11.3: .vhd file diagram

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


ISE Project Navigator (P.20131013) - F:\I SCHEME\VI SEM I SCHEME\pulkit\mod10count\mod10count.xise - [mod10_tb10.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
mod10count
xc3s400-5pq208
mod10_tb10 - behavior (mod10_tb10.vhd)
uut - mod10 - Behavioral (mod10.vhd)
No Processes Running
Processes: mod10_tb10 - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Design Summary (Implemented)
mod10.vhd
mod10_tb10.vhd
Console
Console Errors Warnings Find in Files Results

```
      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 100 ns;
      load <='1';


      -- insert stimulus here

      wait;
   end process;

END;
```

```
ISim simulation engine GUI launched successfully

Process "Simulate Behavioral Model" completed successfully
```

Fig. 11.4: test bench file diagram

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

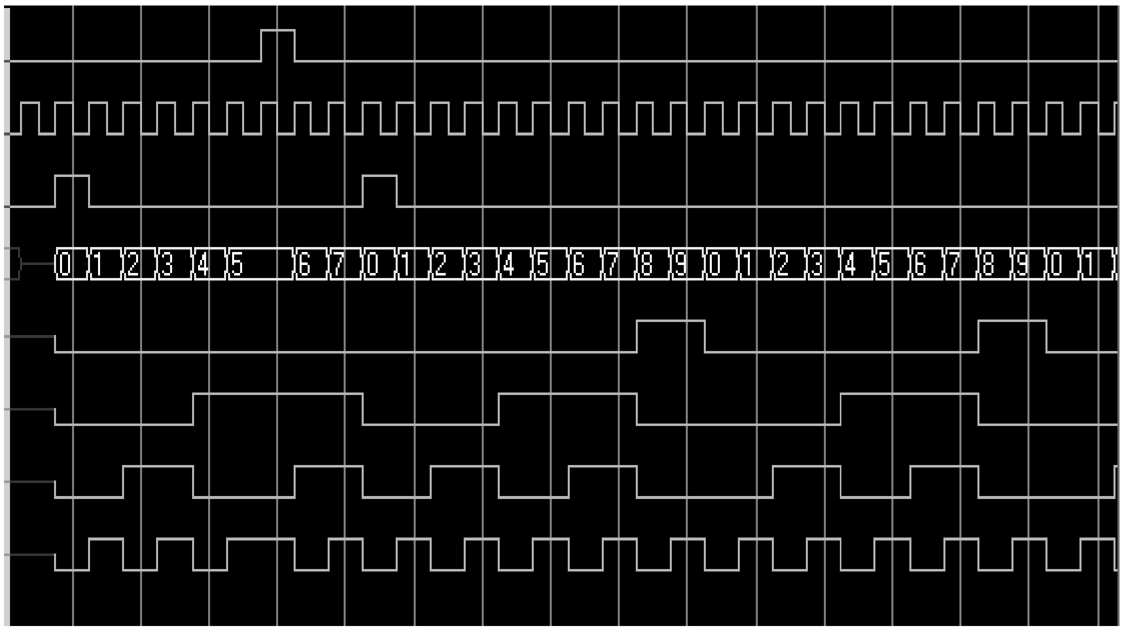


Fig. 11.5: Simulation diagram

e) Actual Experimental set up used in laboratory

IX 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	Loaded with open source IDE, simulation and program downloading software.	1 No.

X 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.

XI 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.
   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.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. Ref Fig no.11.4.
6. Go to implementation to simulation tab , right click on main source file and create Test Bench file for simulation. Or
   Directly go to simulation tab – Process DAC behavioural – ISim simulator—select input entity bit – right click – select Force constant.
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 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.14.3.b where able to simulate designs and check for errors.
10. After simulation Implement up-counter using FPGA.
11. To create .ucf file -- 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.
15. Verify expected result on relevant FPGA lab kit.

SAMPLE PROGRAM 1: **Step 1.Develop VHDL code for Mod-10 Up-counter**

VHDL Code using If else statement

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_signed.all;

entity mod10 is
   Port ( CLK , LOAD: in  STD_LOGIC;
         CLR : in  STD_LOGIC;
         Q : inout std_logic_vector(3 downto 0));
end mod10;

architecture Behavioral of mod10 is

  signal tmp: std_logic_vector(3 downto 0);
  begin
    process (CLK, CLR)
    begin

      if (CLK'EVENT AND CLK='1') then
                              IF CLR='1' then
                               tmp <= "0000";
                                      elsif load = '1' then
                                           if tmp <="1001" then
                                               tmp <= "0000";
                                                        else tmp <= tmp + "0001";
                                                        end if;
                                             end if;
                              end if;

end process;
   Q <= tmp;

end Behavioral;
```

Step 2: Develop Test Bench file for simulation.

VHDL Code

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
 	
ENTITY mod10_tb10 IS
END mod10_tb10;
 	
ARCHITECTURE behavior OF mod10_tb10 IS
 	
    -- Component Declaration for the Unit Under Test (UUT)
     COMPONENT mod10
    PORT(
        CLK : IN std_logic;
        LOAD : IN std_logic;
        CLR : IN std_logic;
        Q : INOUT std_logic_vector(3 downto 0)
       );
    END COMPONENT;
      --Inputs
   signal CLK : std_logic := '0';
   signal LOAD : std_logic := '0';
   signal CLR : std_logic := '0';
 	
          --BiDirs
   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: mod10 PORT MAP (
       CLK => CLK,
       LOAD => LOAD,
       CLR => CLR,
       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 100 ns;
                load <='1';
           wait;
  end process;

END;
```

Problem statement for student: Design VHDL Code for Mod-6 counter using if else statement.

Step 1.Develop VHDL code for Mod-6 Counter

VHDL Code

Step 2: Develop Test Bench file for simulation.

VHDL Code

XII Resources Used

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

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

....
....
....
....
....

XIV Precautions Followed (use blank sheet provided if space not sufficient)

....
....
....
....
....

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

Truth table of Binary Mod-6 up-counter is _______ (verified/ not verified) using FPGA development board.

XVI Result (Output of the Program)

....
....
....
....
....

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

....
....
....
....
....

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

....
....
....
....
....

XIX 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.**

[Space for Answers]

....
....
....
....
....
....
....
....
....
....

XX References / Suggestions for further reading

1. https://www.pantechsolutions.net/vhdl-code-to-simulate-4-bit-binary-counter-by-software-using-spartan-3-starter-kit
2. https://www.xilinx.com/support/documentation/data_sheets/ds610.pdf
3. https://forums.xilinx.com/
4. https://www.youtube.com/watch?v=VeKDFGoqwgU
5. https://www.geeksforgeeks.org/differences-between-synchronous-and-asynchronous-counter/

XXI Assessment Scheme

Performance indicators		Weightage
Process related (15 Marks)		**60%**
1	Coding and Debugging ability	30%
2	Making connections of hardware	20%
3	Follow ethical practices.	10%
Product related (10 Marks)		**40%**
4	Correct connection s to FPGA board.	20%
5	Relevance of output of the problem definition.	15%
6	Timely Submission of report, Answer to sample questions.	05%
	Total (25 Marks)	**100%**

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 12: Implement synchronous counter using FPGA.

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 Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronics circuits and equipment.'**

- Develop 'VHDL' code using EDA tools.
- Use Input/output port pins of FPGA.
- Interface FPGA KIT and desktop PC.

IV Relevant Course Outcome(s)

Debug VHDL programme for the given application.

V Practical Outcome

Implement up-counter using FPGA.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum 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 flip-flop. 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:

a) Up counter
b) 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 pre-set value is reached.

Table 12.1: 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

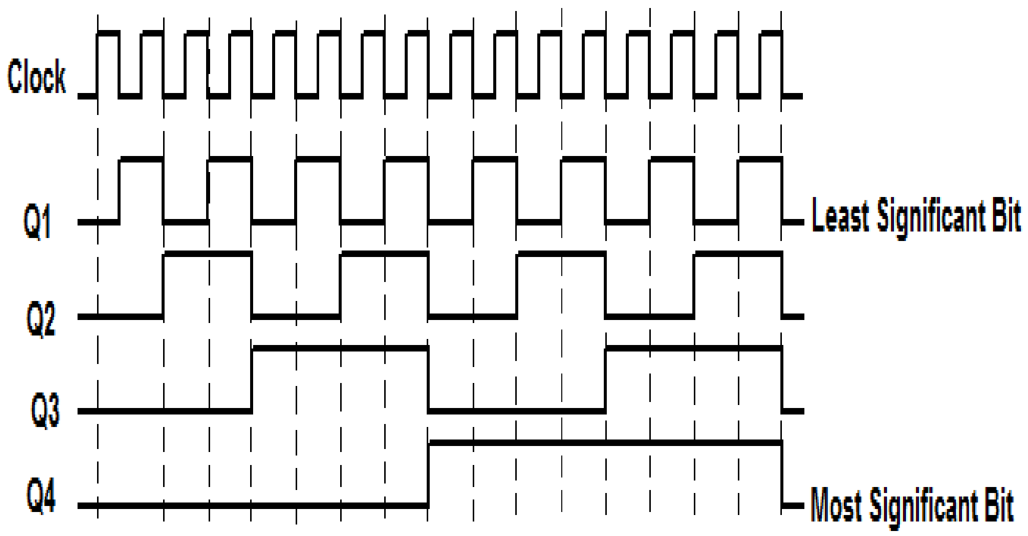
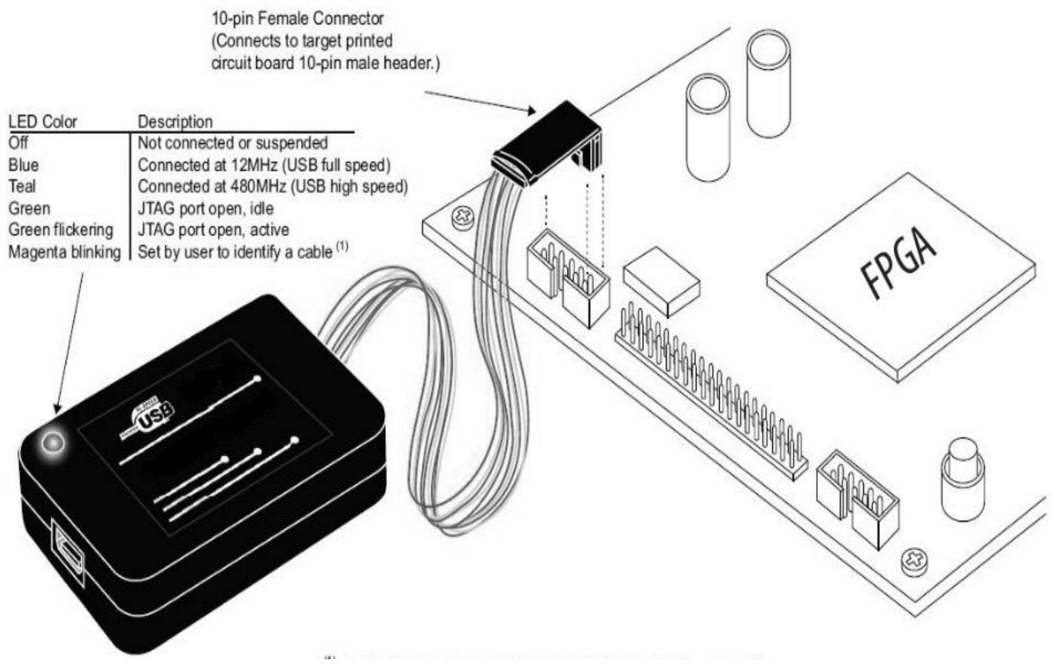


Figure 12.1: Four bit Up counter timing waveform.

VIII Practical Circuit diagram:

a) Practical JTAG cable setup:-



LED Color	Description
Off	Not connected or suspended
Blue	Connected at 12MHz (USB full speed)
Teal	Connected at 480MHz (USB high speed)
Green	JTAG port open, idle
Green flickering	JTAG port open, active
Magenta blinking	Set by user to identify a cable [1]

10-pin Female Connector
(Connects to target printed
circuit board 10-pin male header.)
USB
FPGA

Figure 12.2.a: Practical Setup with JTAG CABLE

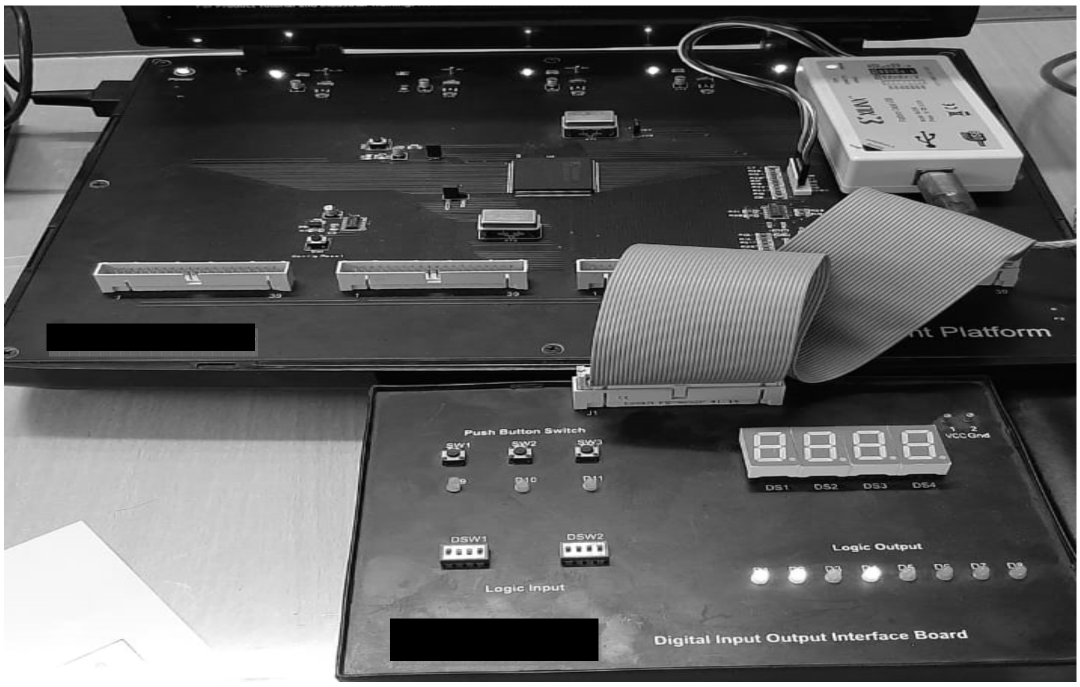
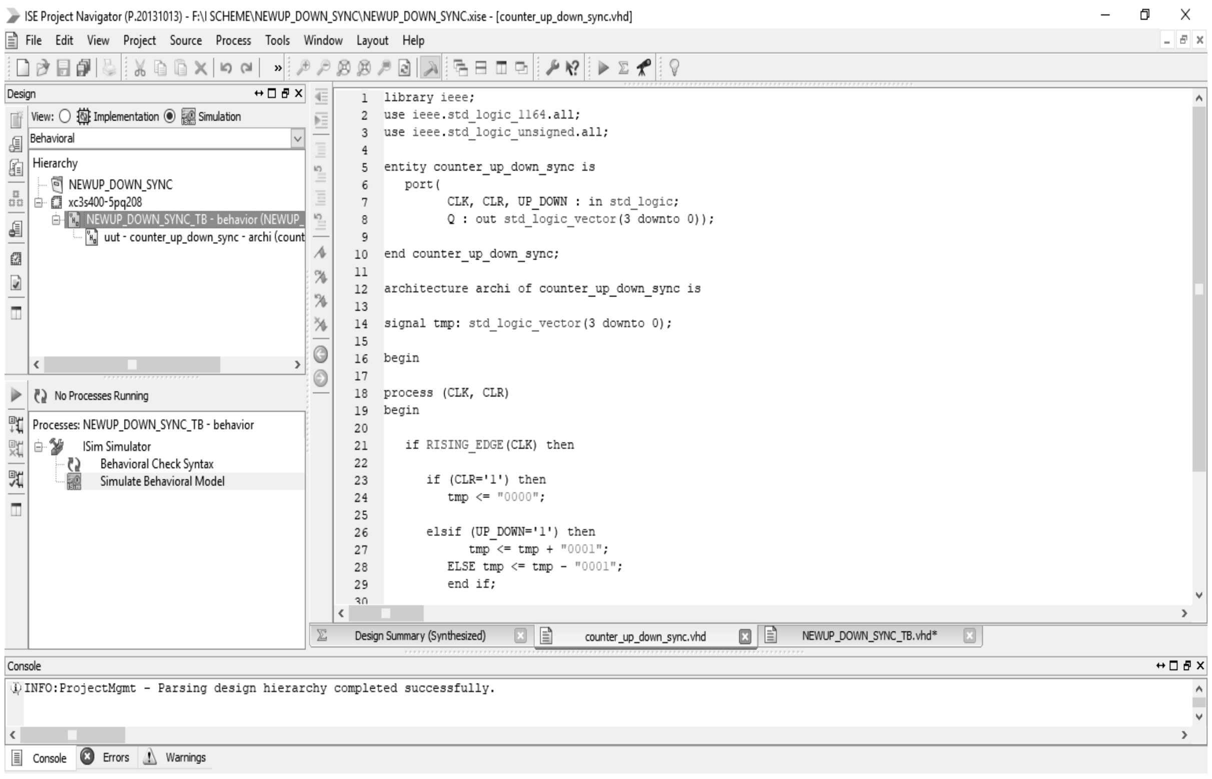


Figure 12.2.b: Lab Practical Setup

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



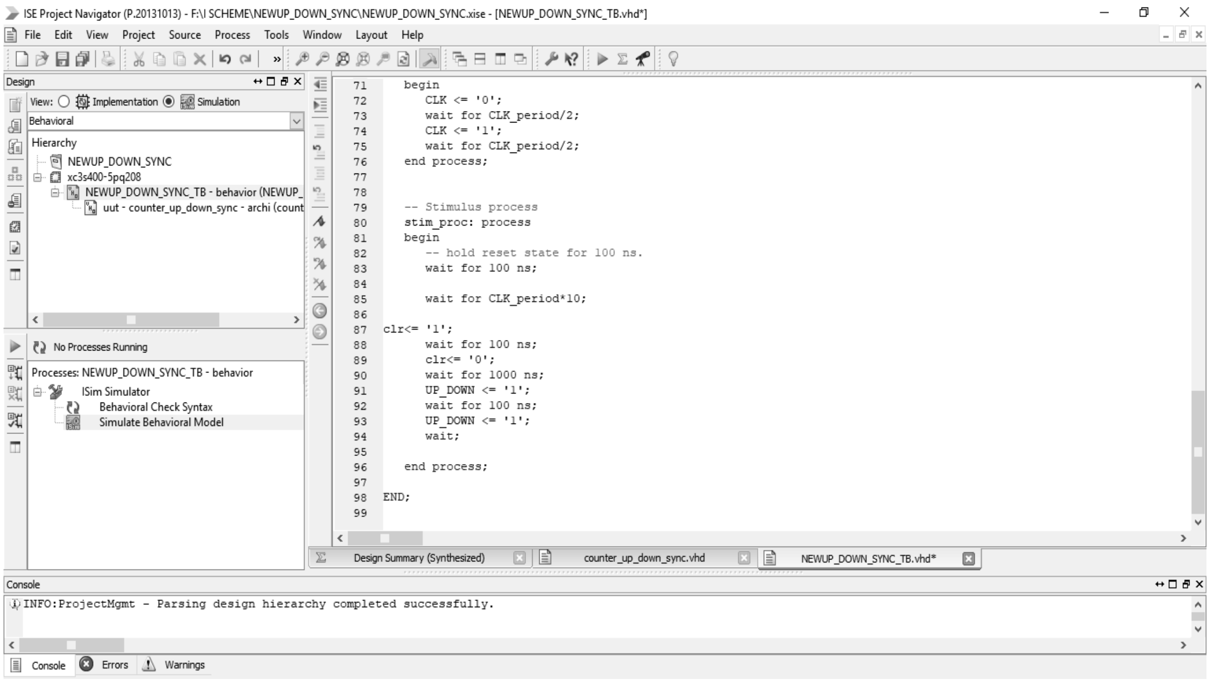
ISE Project Navigator (P.20131013) - F:\I SCHEME\NEWUP_DOWN_SYNC\NEWUP_DOWN_SYNC.xise - [counter_up_down_sync.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
NEWUP_DOWN_SYNC
xc3s400-5pq208
NEWUP_DOWN_SYNC_TB - behavior (NEWUP_
uut - counter_up_down_sync - archi (count
No Processes Running
Processes: NEWUP_DOWN_SYNC_TB - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Design Summary (Synthesized)
counter_up_down_sync.vhd
NEWUP_DOWN_SYNC_TB.vhd*
Console
Console
Errors
Warnings

```
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;
```

```
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
```

Figure 12.3: .vhd file

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

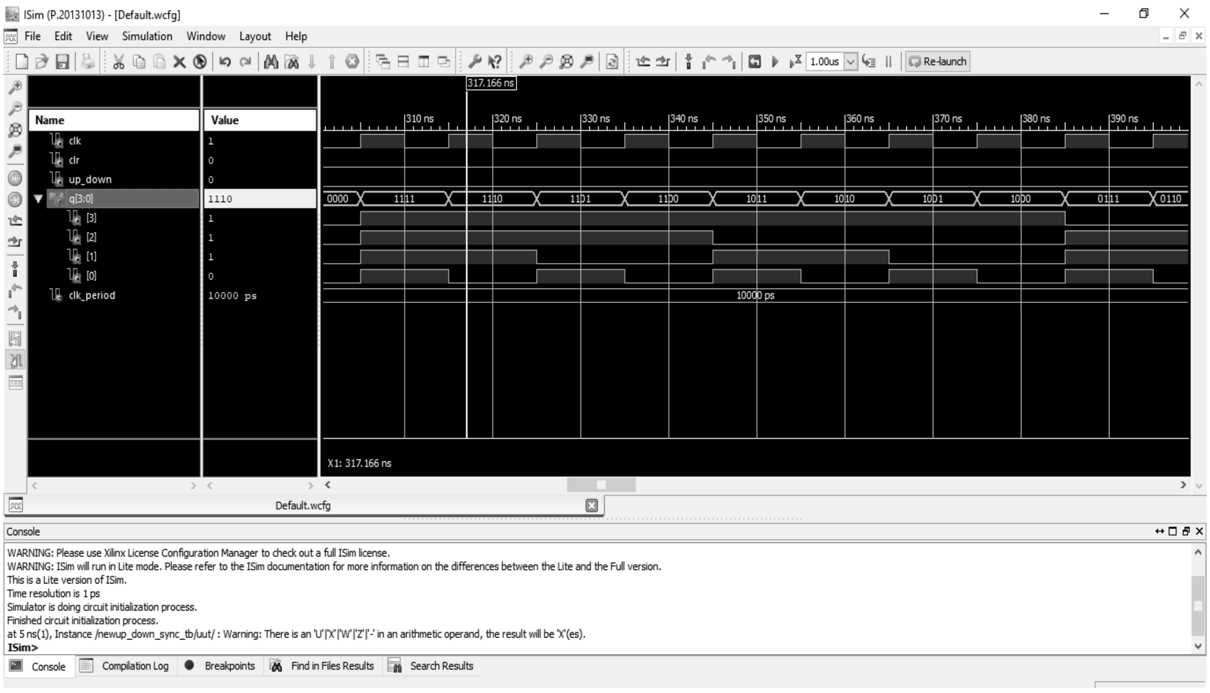


```
   begin
      CLK <= '0';
      wait for CLK_period/2;
      CLK <= '1';
      wait for CLK_period/2;
   end process;


   -- 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;
```

```
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
```

Figure 12.4: Test bench file

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



```
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
at 5 ns(1), Instance /newup_down_sync_tb/uut/ : Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
ISim>
```

Figure 12.5: Simulation output

e) Actual Experimental set up used in laboratory

IX 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	Loaded with open source IDE, simulation and program downloading software.	1 No.

X 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.

XI 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.
   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.12.3.
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.12.4.
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 "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 r test bench loaded.
9. ISim simulator window will open with simulation executed, as shown in Ref Fig no.12.5. where able to simulate designs and check for errors.
10. After simulation implement UP counter using FPGA development board.
11. 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.

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

XII Resources Used

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

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

....

....

....

....

....

XIV Precautions Followed (use blank sheet provided if space not sufficient)

....

....

....

....

....

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

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

XVI Result (Output of the Program)

....

....

....

....

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

....
....
....
....
....

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

....
....
....
....
....

XIX 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]

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

XX References / Suggestions for further reading

1. https://www.pantechsolutions.net/vhdl-code-to-simulate-4-bit-binary-counter-by-software-using-spartan-3-starter-kit
2. https://circuitdigest.com/tutorial/synchronous-counter
3. https://forums.xilinx.com/
4. https://www.geeksforgeeks.org/differences-between-synchronous-and-asynchronous-counter/

XXI Assessment Scheme

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

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 13: Implement binary to gray code converter using FPGA.

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 that is Binary to Gray code converter application on FPGA board using Xilinx ISE tools.

II Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronics circuits and equipments.'**

- Develop 'VHDL' code using EDA tools.
- Use Input/output port pins of FPGA.
- Interface FPGA KIT and desktop PC.

IV Relevant Course Outcome(s)

Debug VHDL programme for the given application.

V Practical Outcome

Implement binary to gray code converter using FPGA.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum 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:

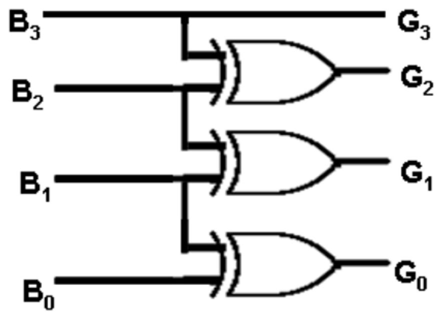
$G_3 = B_3$

$G_2 = B_3$ XOR B_2

$G_1 = B_2$ XOR B_1

$G_0 = B_1$ XOR B_0

A **binary to gray code converter** can be implemented using XOR gates. For n input, n-1 gates are required. As shown in fig.. 13.1 for 4 inputs, 3 XOR gates are used.

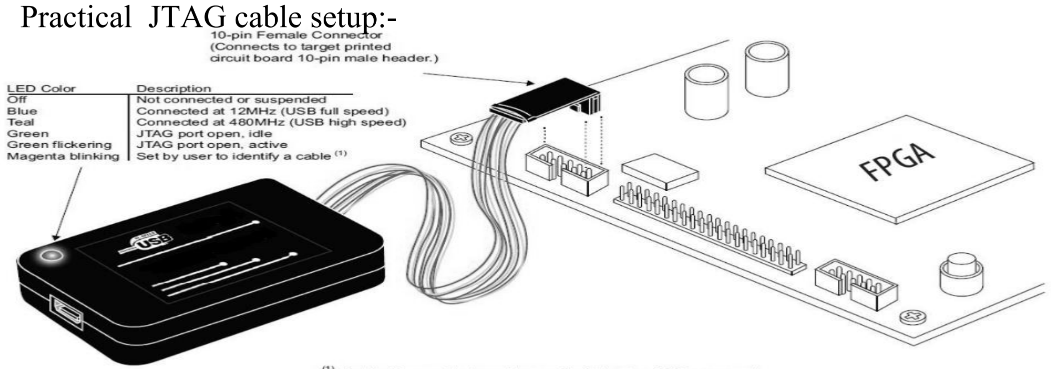


Binary				Gray Code			
b_3	b_2	b_1	b_0	g_3	g_2	g_1	g_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1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

Fig 13.1: Binary to Gray Code Converter **Table No. 13.1: Truth table**

VIII Practical Circuit diagram:

a) Practical JTAG cable setup:-


10-pin Female Connector (Connects to target printed circuit board 10-pin male header.)
USB
FPGA

LED Color	Description
Off	Not connected or suspended
Blue	Connected at 12MHz (USB full speed)
Teal	Connected at 480MHz (USB high speed)
Green	JTAG port open, idle
Green flickering	JTAG port open, active
Magenta blinking	Set by user to identify a cable [1]

Fig 13.2.a: Practical Setup with JTAG CABLE

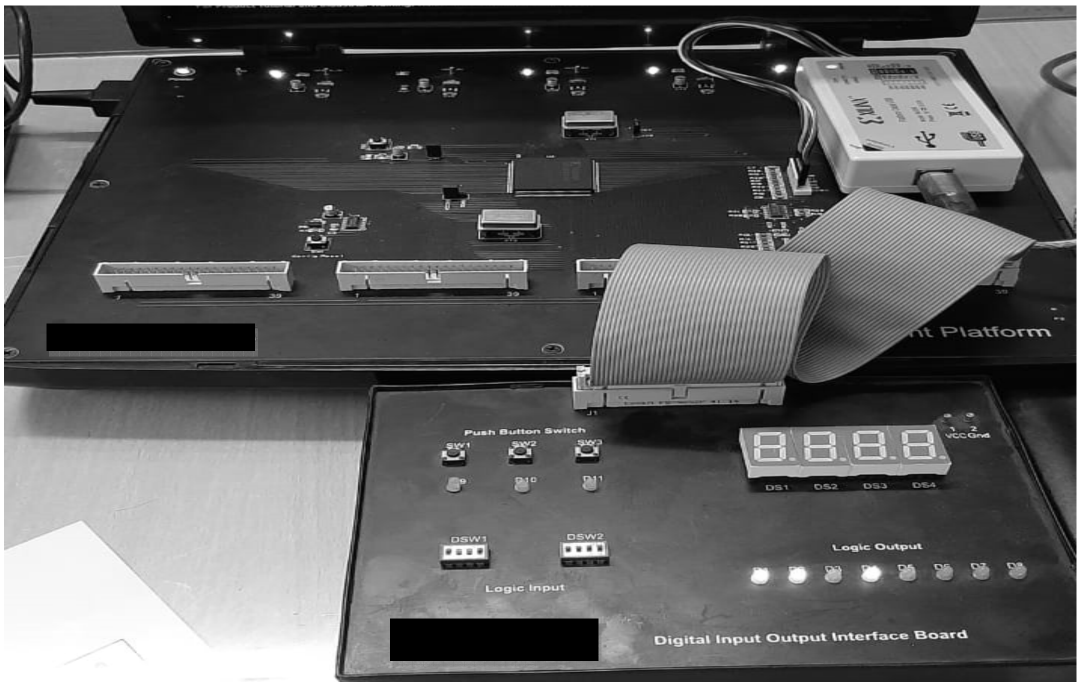
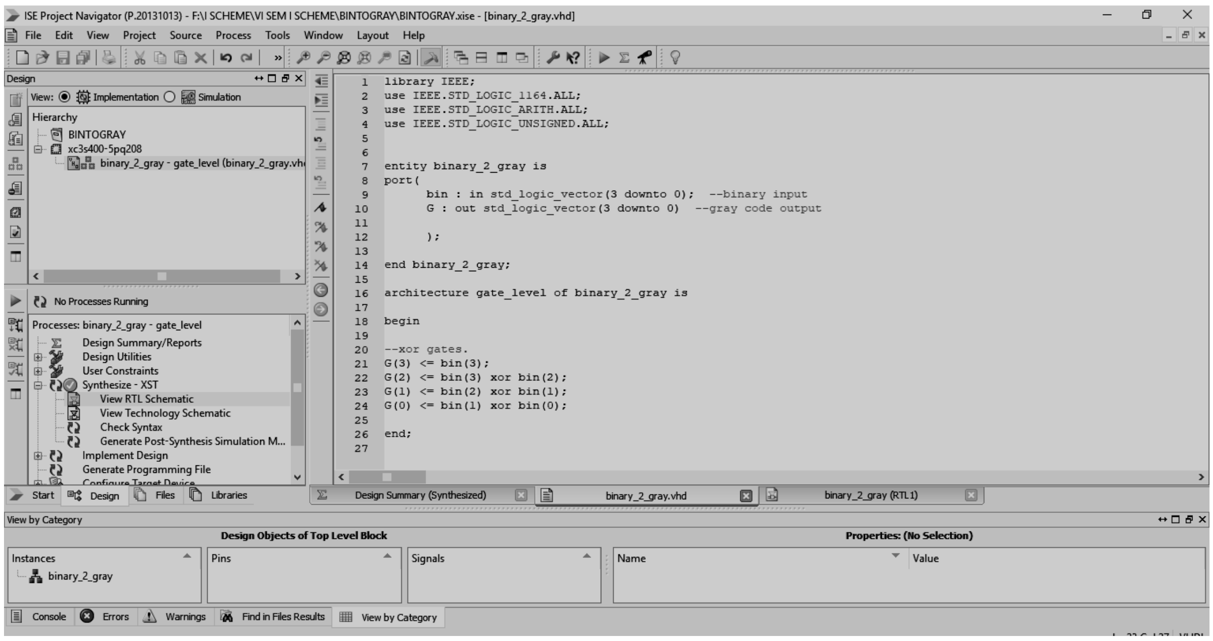


Fig 13.2.b: Lab Practical Setup

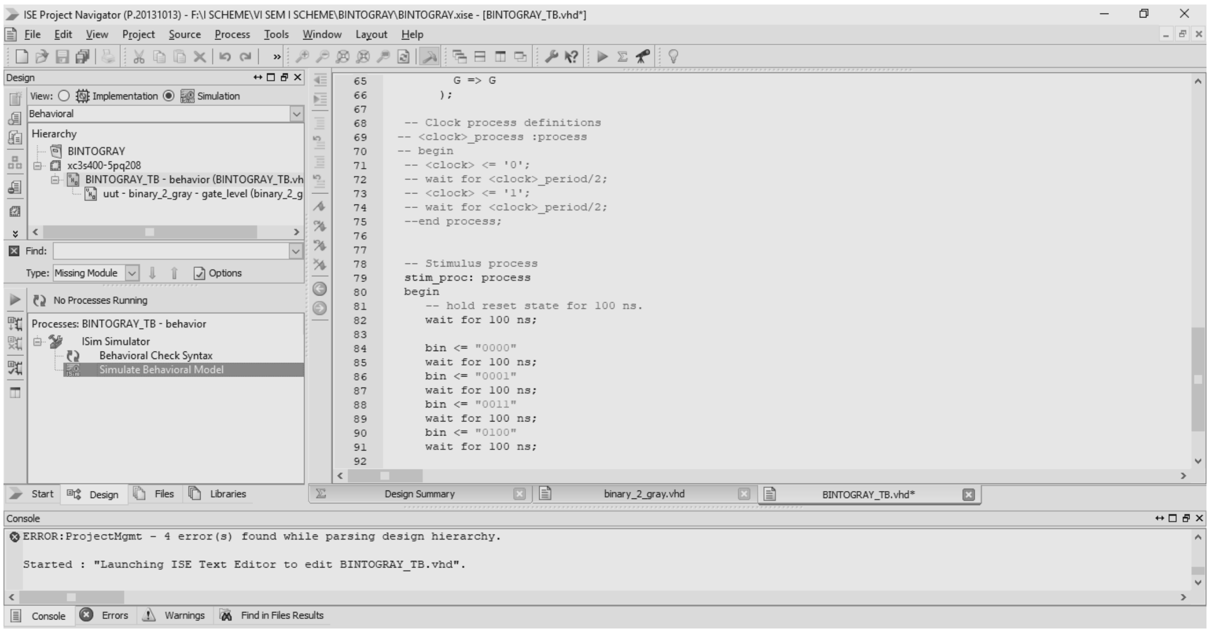
b) Create binary to gray code converter .vhd file:-


ISE Project Navigator (P.20131013) - F:\I SCHEME\VI SEM I SCHEME\BINTOGRAY\BINTOGRAY.xise - [binary_2_gray.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
BINTOGRAY
xc3s400-5pq208
binary_2_gray - gate_level (binary_2_gray.vhd)
No Processes Running
Processes: binary_2_gray - gate_level
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation M...
Implement Design
Generate Programming File
Configure Target Device
Start Design Files Libraries
Design Summary (Synthesized)
binary_2_gray.vhd
binary_2_gray (RTL1)
View by Category
Design Objects of Top Level Block
Properties: (No Selection)
Instances
binary_2_gray
Pins
Signals
Name
Value
Console Errors Warnings Find in Files Results View by Category

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


entity binary_2_gray is
port(
      bin : in std_logic_vector(3 downto 0);  --binary input
      G : out std_logic_vector(3 downto 0)  --gray code output

      );

end binary_2_gray;

architecture gate_level of binary_2_gray is

begin

--xor gates.
G(3) <= bin(3);
G(2) <= bin(3) xor bin(2);
G(1) <= bin(2) xor bin(1);
G(0) <= bin(1) xor bin(0);

end;
```

Fig. 13.3: .vhd file

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



ISE Project Navigator (P.20131013) - F:\I SCHEME\VI SEM I SCHEME\BINTOGRAY\BINTOGRAY.xise - [BINTOGRAY_TB.vhd*]

```
            G => G
          );

   -- 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;

      bin <= "0000"
      wait for 100 ns;
      bin <= "0001"
      wait for 100 ns;
      bin <= "0011"
      wait for 100 ns;
      bin <= "0100"
      wait for 100 ns;
```

```
ERROR:ProjectMgmt - 4 error(s) found while parsing design hierarchy.

Started : "Launching ISE Text Editor to edit BINTOGRAY_TB.vhd".
```

Fig. 13.4: Test bench file

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

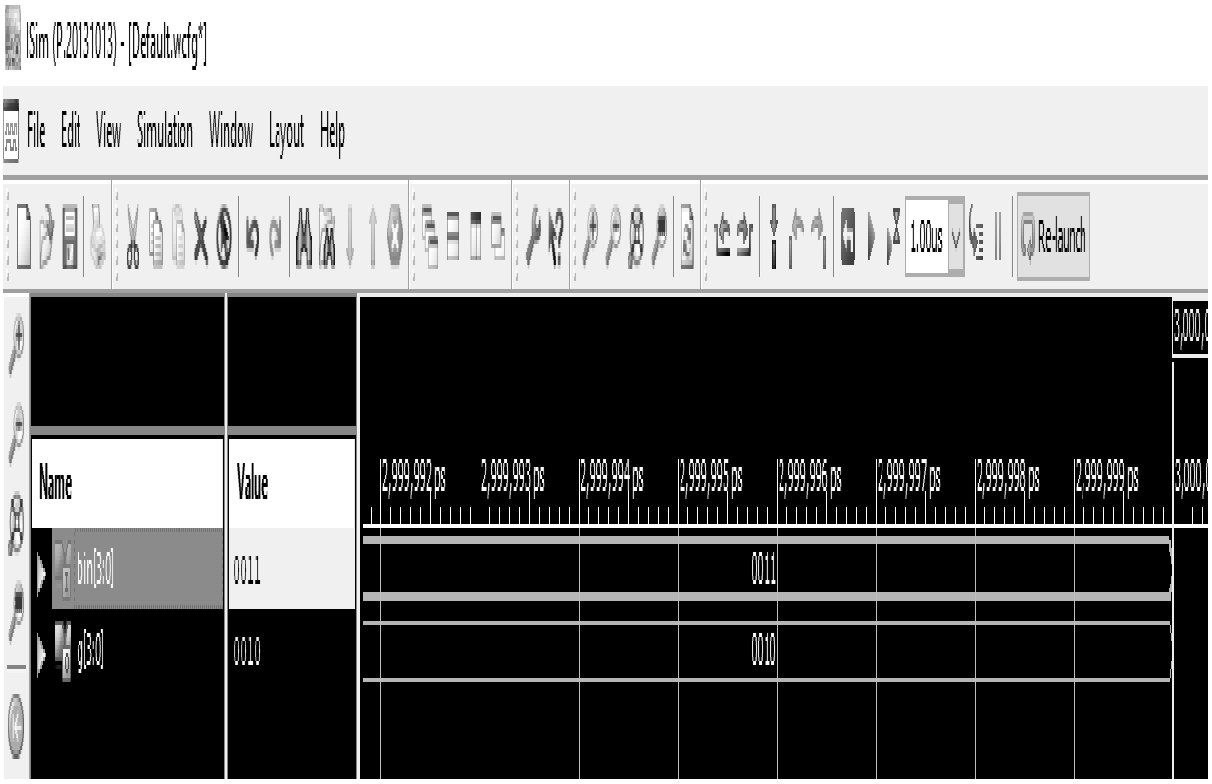


Fig. 13.5: Simulation output

e) Actual Experimental set up used in laboratory

IX 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	Loaded with open source IDE, simulation and program downloading software.	1 No.

X 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.

XI 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.

   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.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 binary to gray code converter 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 Config. 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;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;



entity binary_2_gray is
port(
                bin : in std_logic_vector(3 downto 0);  --binary input
                G : out std_logic_vector(3 downto 0)  --gray code output
                );
```

```
end binary_2_gray;

architecture gate_level of binary_2_gray is

begin

--xor gates.
G(3) <= bin(3);
G(2) <= bin(3) xor bin(2);
G(1) <= bin(2) xor bin(1);
G(0) <= bin(1) xor bin(0);

end;
```

Step 2: Develop Test Bench file for simulation.

VHDL Code

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

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

ENTITY BINTOGRAY_TB IS
END BINTOGRAY_TB;

ARCHITECTURE behavior OF BINTOGRAY_TB IS

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

   COMPONENT binary_2_gray
   PORT(
       bin : IN  std_logic_vector(3 downto 0);
       G : OUT  std_logic_vector(3 downto 0)
      );
   END COMPONENT;


  --Inputs
  signal bin : std_logic_vector(3 downto 0) := (others => '0');

         --Outputs
  signal G : std_logic_vector(3 downto 0);
  -- No clocks detected in port list. Replace <clock> below with
  -- appropriate port name
  --constant <clock>_period : time := 10 ns;

BEGIN

         -- Instantiate the Unit Under Test (UUT)
  uut: binary_2_gray PORT MAP (
       bin => bin,
       G => G
      );

  -- 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;

    bin <= "0000"
                wait for 100 ns;
                bin <= "0001"
                wait for 100 ns;
                bin <= "0011"
                wait for 100 ns;
    bin <= "0100"
                wait for 100 ns;

                wait;

  end process;

END;
```

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

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

VHDL Code using behavioral method statement

Step 2: Develop Test Bench file for simulation.

VHDL Code

XII Resources Used

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

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

....
....
....
....
....

XIV Precautions Followed (use blank sheet provided if space not sufficient)

....
....
....
....
....

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

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

XVI Result (Output of the Program)

....
....
....
....
....

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

....
....
....
....
....

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

....
....
....
....
....

XIX 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 Gray to Binary using When statement.
2. Design VHDL Code using FPGA board for Gray to Binary using Dataflow method statement.

[Space for Answers]

XX References / Suggestions for further reading

1. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_usb_blstr_ii_cable.pdf
2. https://www.xilinx.com/support/documentation/data_sheets/ds610.pdf
3. https://forums.xilinx.com/
4. https://www.geeksforgeeks.org/digital-logic-code-converters-binary-gray-code/

XXI Assessment Scheme

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

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 14: Build / Test DAC using FPGA.

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 and Digital to Analog converter. 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 Relevant Program Outcomes (POs)

- Discipline knowledge: Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- Experiments and practice: Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- Engineering tools: Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- Lifelong learning: Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronics circuits and equipments.'**

- Develop 'VHDL' code using EDA tools.
- Use Input/output port pins of FPGA.
- Interface FPGA KIT and desktop PC.

IV Relevant Course Outcome(s)

Debug VHDL programme for the given application.

V Practical Outcome

Build / Test DAC using FPGA.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum 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.

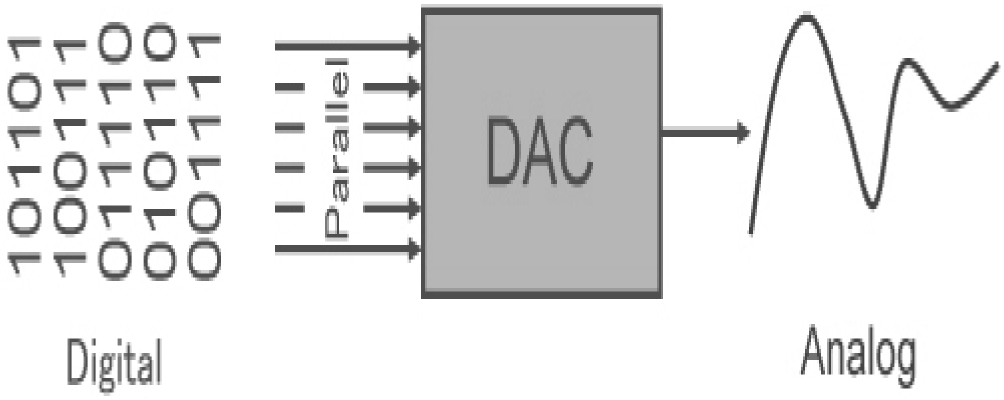


Fig 14.1.a: Basic Digital to Analog signal representation.

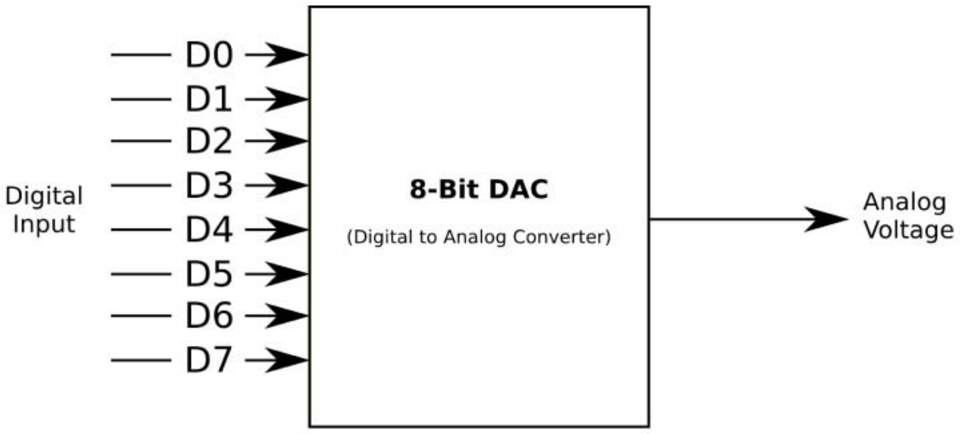


Fig 14.1.b: 8 Bit Digital to Analog converter.

VIII Practical Circuit diagram:

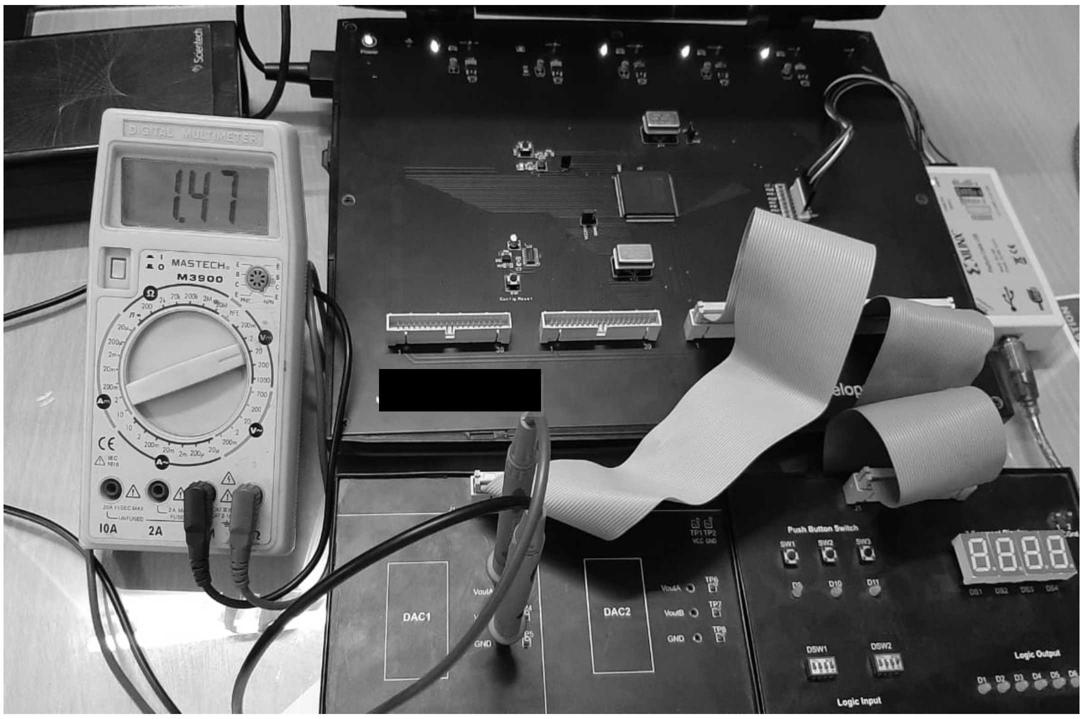

[redacted]

Fig 14.2.a: Lab Practical Setup by set LSB data Bit high Analog output is low voltage

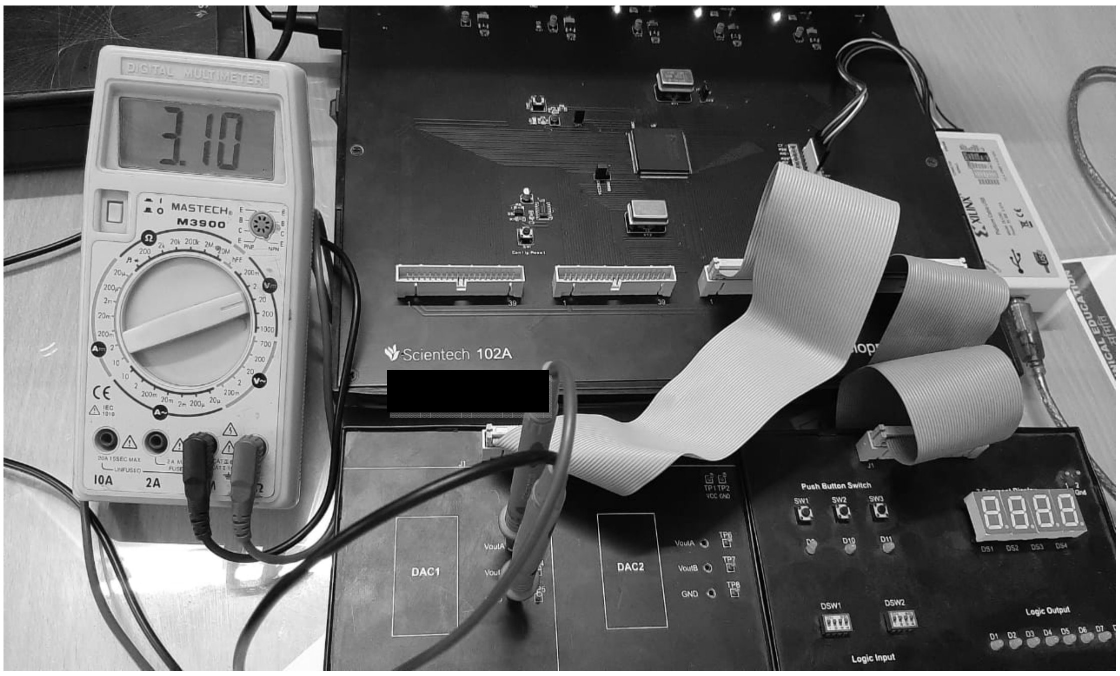
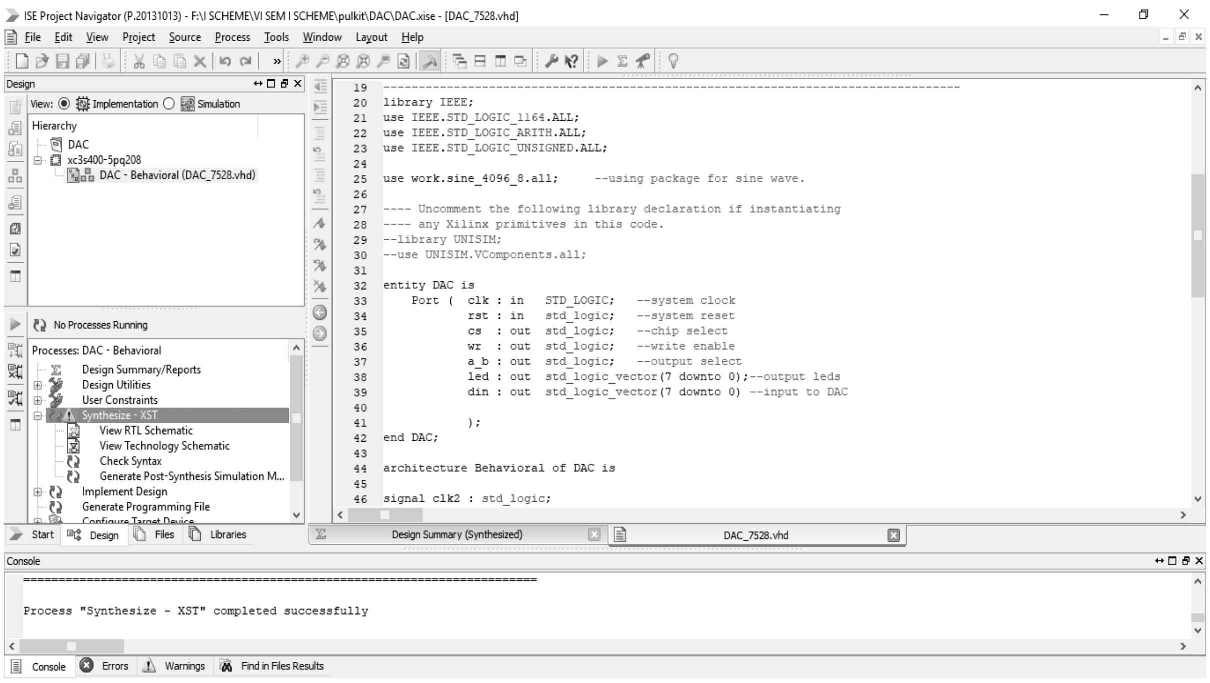

[redacted]

Fig 14.2.b: Lab Practical Setup by set MSB data Bit high Analog output is high voltage

Create ADC .vhd file:-

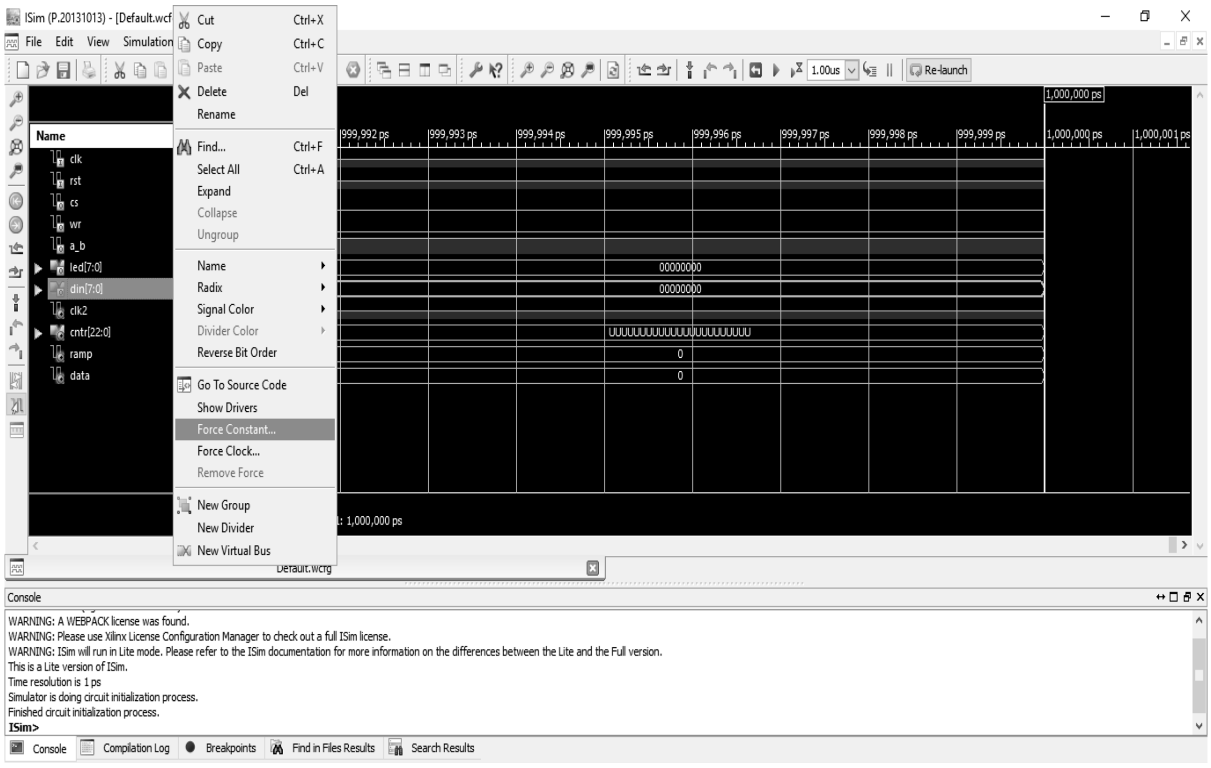

ISE Project Navigator (P.20131013) - F:\I SCHEME\VI SEM I SCHEME\pulkit\DAC\DAC.xise - [DAC_7528.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
DAC
xc3s400-5pq208
DAC - Behavioral (DAC_7528.vhd)
No Processes Running
Processes: DAC - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation M...
Implement Design
Generate Programming File
Start Design Files Libraries
Design Summary (Synthesized)
DAC_7528.vhd
Console
Console Errors Warnings Find in Files Results

```
----------------------------------------------------------------------------------
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.

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

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;
```

```
========================================================================

Process "Synthesize - XST" completed successfully
```

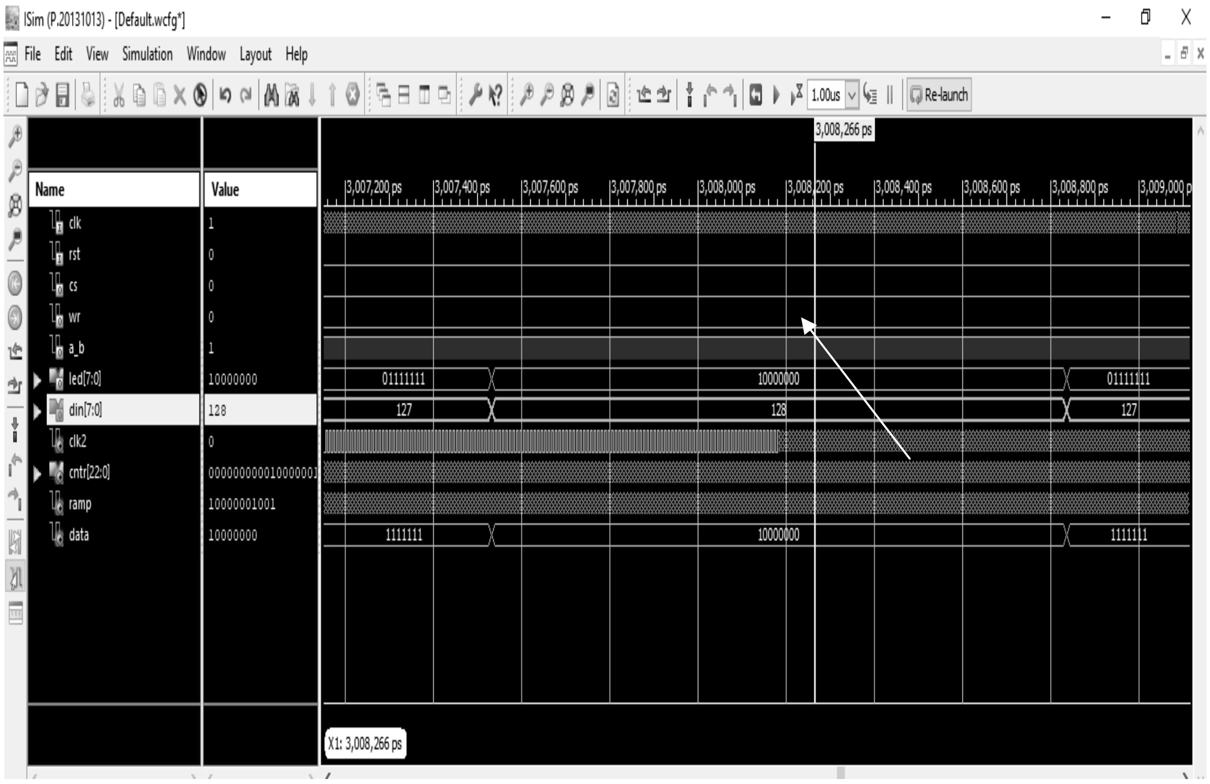
Fig 14.3: .vhd file

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



```
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Fig 14.4.a: Simulation diagram


ISim (P.20131013) - [Default.wcfg*]

Name	Value
clk	1
rst	0
cs	0
wr	0
a_b	1
led[7:0]	10000000
din[7:0]	128
clk2	0
cntr[22:0]	00000000001000000
ramp	10000001001
data	10000000

Fig 14.4.b: Simulation diagram (arrow shows peak level of converted analog signal)

c) Actual Experimental set up used in laboratory

IX 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	Loaded with open source IDE, simulation and program downloading software.	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.

X 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.

XI 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.
   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.14.3.
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
   Directly go to simulation tab – Process DAC behavioural – ISim simulator—select input entity bit – right click – select Force constant. Ref Fig no.14.4.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 Behavioral 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.14.4.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 (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 Config. 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 converter.**

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 = "11111111111111111111111")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.

XII Resources Used

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

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

....
....
....
....
....

XIV Precautions Followed (use blank sheet provided if space not sufficient)

....
....
....
....
....

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

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	

XVI Result (Output of the Program)

....
....
....
....
....

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

....
....
....
....
....

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

....
....
....
....
....

XIX 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 for ADC using FPGA board..

[Space for Answers]

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

XX References / Suggestions for further reading

1. https://www.pantechsolutions.net/vhdl-code-to-simulate-4-bit-binary-counter-by-software-using-spartan-3-starter-kit
2. https://www.xilinx.com/support/documentation/data_sheets/ds610.pdf
3. https://forums.xilinx.com/
4. https://www.rohm.com/electronics-basics/ad-da-converters/what-are-ad-da-converters
5. https://www.elprocus.com/digital-to-analog-converter-dac-applications/

XXI Assessment Scheme

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

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 15: Implement Stepper motor controller using FPGA.

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 Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronics circuits and equipments.'**

- Develop 'VHDL' code using EDA tools.
- Use Input/output port pins of FPGA.
- Interface FPGA KIT and desktop PC.

IV Relevant Course Outcome(s)

Maintain FPGA based circuits.

V Practical Outcome

Implement Stepper motor controller using FPGA.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum 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.

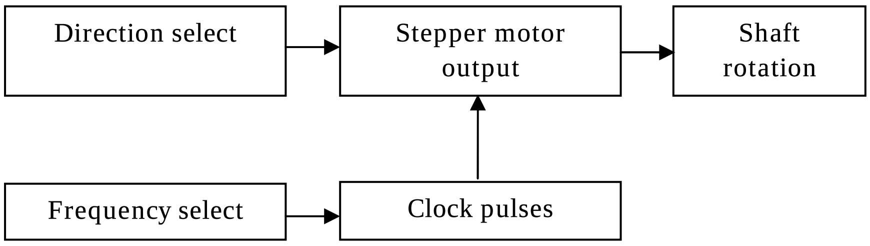


Fig. 15.1: Basic Stepper motor system

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.

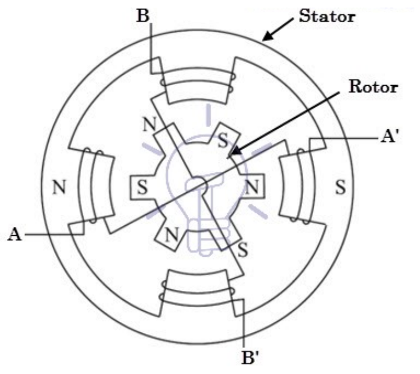


Fig. 15.2: Stepper motor stator and rotator

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 15.1 and 15.2 and the controller very well executes this function.

Table No. 15.1: Clockwise Rotation

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. 15.2: Anti Clock Wise Rotation

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.

VIII Practical Circuit diagram:

a) Practical JTAG cable setup:

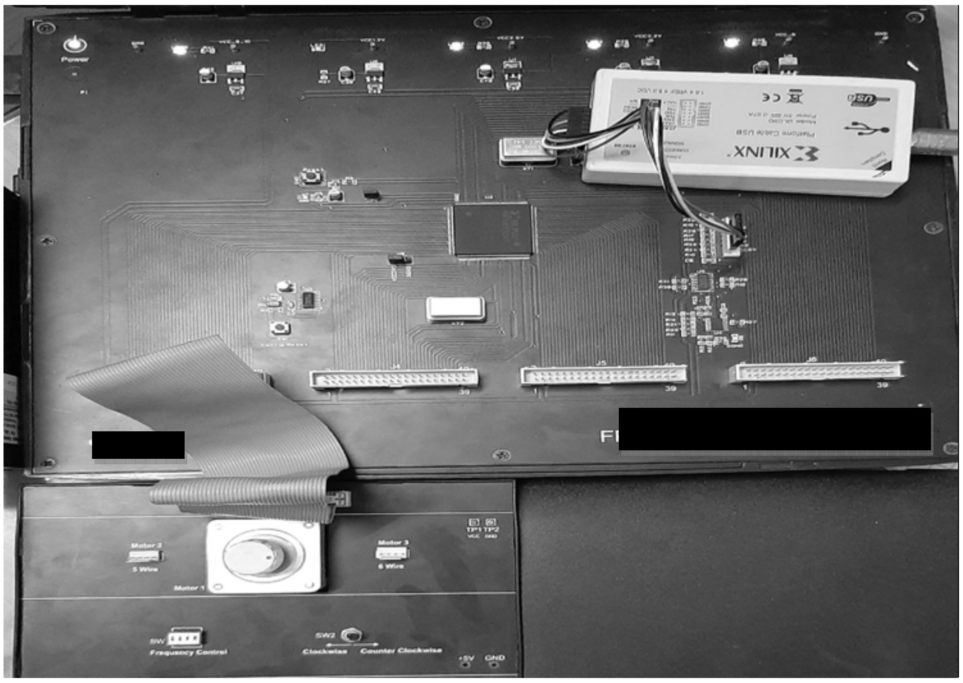
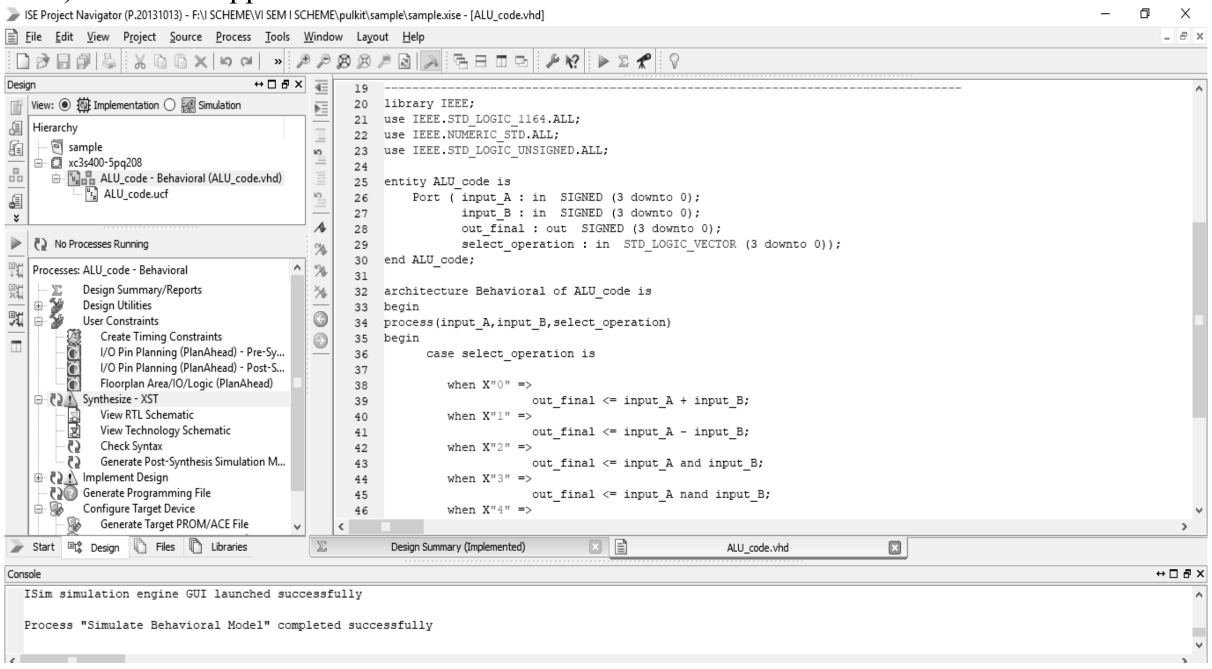

[redacted] [redacted]

Fig. 15.2: Lab Practical Setup

b) Create stepper motor .vhd file :-


ISE Project Navigator (P.20131013) - F:\I SCHEME\VI SEM I SCHEME\pulkit\sample\sample.xise - [ALU_code.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
sample
xc3s400-5pq208
ALU_code - Behavioral (ALU_code.vhd)
ALU_code.ucf
No Processes Running
Processes: ALU_code - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Create Timing Constraints
I/O Pin Planning (PlanAhead) - Pre-Sy...
I/O Pin Planning (PlanAhead) - Post-S...
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation M...
Implement Design
Generate Programming File
Configure Target Device
Generate Target PROM/ACE File
Start Design Files Libraries
Design Summary (Implemented)
ALU_code.vhd
Console

```
--------------------------------------------------------------------------------
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" =>
```

```
ISim simulation engine GUI launched successfully

Process "Simulate Behavioral Model" completed successfully
```

Fig. 15.3: .vhd file

c) Create View technology schematic diagram :-

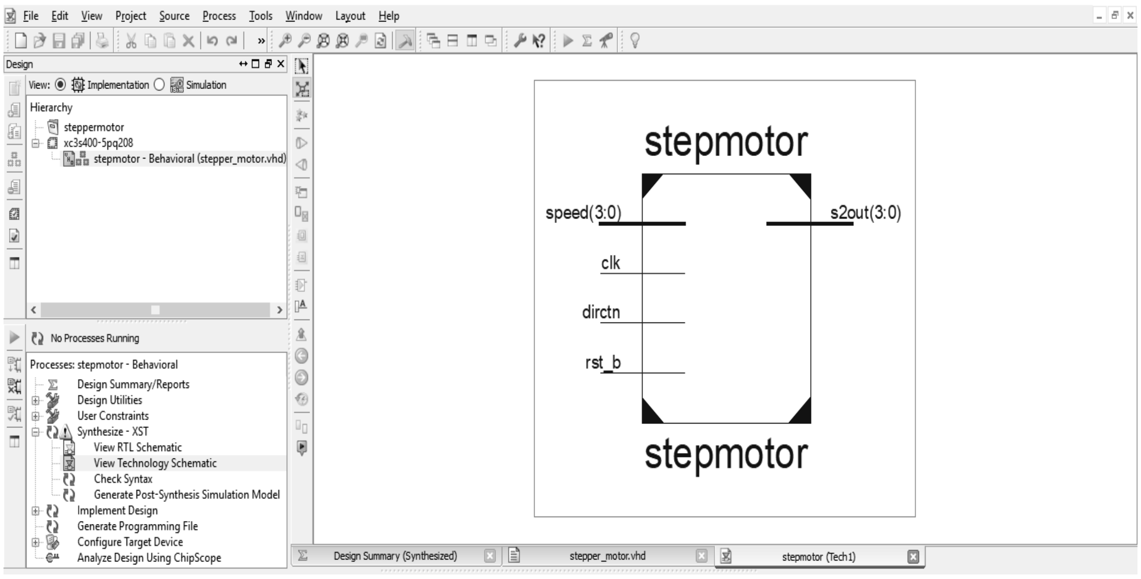
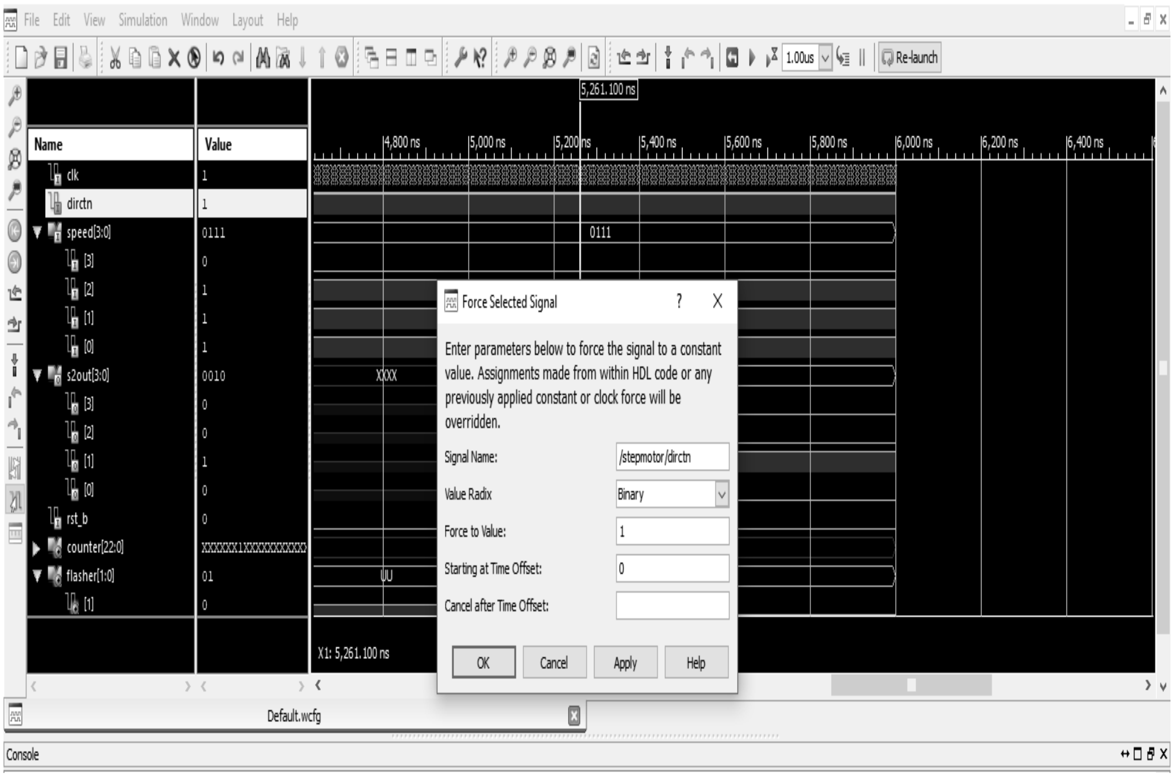

File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
steppermotor
xc3s400-5pq208
stepmotor - Behavioral (stepper_motor.vhd)
No Processes Running
Processes: stepmotor - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
stepmotor
speed(3:0)
s2out(3:0)
clk
dirctn
rst_b
stepmotor
Design Summary (Synthesized)
stepper_motor.vhd
stepmotor (Tech1)

Fig. 15.4: View Technology schematic diagram

d) Stepper motor Simulation waveforms-



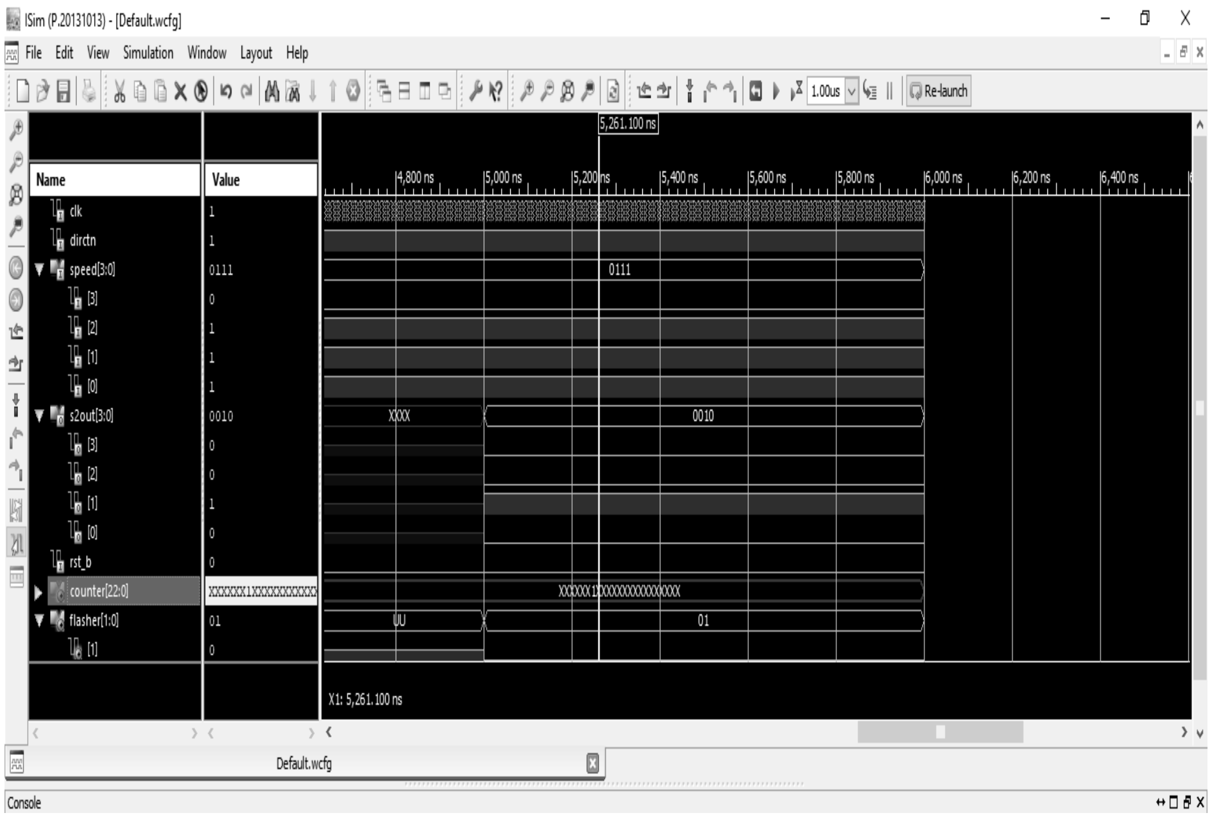
Force Selected Signal

Enter parameters below to force the signal to a constant value. Assignments made from within HDL code or any previously applied constant or clock force will be overridden.

Signal Name:	/stepmotor/dirctn
Value Radix	Binary
Force to Value:	1
Starting at Time Offset:	0
Cancel after Time Offset:	

OK Cancel Apply Help

Fig. 15.5: Forcefully to generate simulation output



Name	Value
clk	1
dirctn	1
speed[3:0]	0111
[3]	0
[2]	1
[1]	1
[0]	1
s2out[3:0]	0010
[3]	0
[2]	0
[1]	1
[0]	0
rst_b	0
counter[22:0]	XXXXXX1XXXXXXXXXXXX
flasher[1:0]	01
[1]	0

Fig. 15.6: Simulation output

e) Actual Experimental set up used in laboratory

IX 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.	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	Loaded with open source IDE, simulation and program downloading software.	1 No.

X 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.

XI 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.
   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.15.3 and 15.4.
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 2 as 4 decoder Using FPGA development board.
11. 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 Config. 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 :- Step 1.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
------------------------------serial port------------------------------
                              rst_b   : in std_logic                  -- external reset
          );
  end stepmotor;

architecture Behavioral of stepmotor is

signal counter:std_logic_vector(22 downto 0);             --Signal for main counter
signal flasher:std_logic_vector(1 downto 0);             --signal for stepper sequence

begin
process(clk,dirctn,speed,rst_b)
begin

if(rst_b = '1')then
        counter <= (others => '0');

elsif(clk'event and clk='1')then
        counter<=counter+1;                                --counter increment

                                if(speed="0000")then
                                        if(counter(20)='1')then
                                                flasher<=flasher+1;
                                                counter<="00000000000000000000000";
                                        end if;
                                elsif(speed="0001")then
                                        if(counter(19)='1')then
                                                flasher<=flasher+1;
                                                counter<="0000000000000000000000";
                                        end if;
                                elsif(speed="0011")then
                                        if(counter(18)='1')then
                                                flasher<=flasher+1;
                                                counter<="0000000000000000000000";
                                        end if;
                                elsif(speed="0111")then
                                        if(counter(17)='1')then
                                                flasher<=flasher+1;
                                                counter<="0000000000000000000000";
                                        end if;
                                elsif(speed="1111")then
                                        if(counter(16)='1')then
                                                flasher<=flasher+1;
                                                counter<="0000000000000000000000";
                                        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";

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

Problem statement for student: Design VHDL Code for Stepper Motor using if else statement

Step 1.Develop VHDL code for Stepper Motor.

VHDL Code using if else statement

XII Resources Used

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

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

....
....
....
....
....

XIV Precautions Followed (use blank sheet provided if space not sufficient)

....
....
....
....
....

XV 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.

XVI Result (Output of the Program)

....
....
....
....
....

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

....
....
....
....
....

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

....
....
....
....
....

XIX 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 calculate the step angle for this motor.

[Space for Answers]

XX References / Suggestions for further reading

1. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_usb_blstr_ii_cable.pdf
2. https://www.xilinx.com/support/documentation/data_sheets/ds610.pdf
3. https://forums.xilinx.com/
4. https://www.researchgate.net/publication/258507854_FPGA_based_stepper_motor_controller
5. https://www.pantechsolutions.net/interfacing-stepper-motor-with-spartan3-fpga-development-kit

XXI Assessment Scheme

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

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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

Practical No. 16: 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 that is four Bit ALU or sequence generator application on FPGA board using Xilinx ISE tools.

II Relevant Program Outcomes (POs)

- **Discipline knowledge:** Apply Electronics and Telecommunication engineering knowledge to solve broad-based Electronics and Telecommunications engineering related problems.
- **Experiments and practice:** Plan to perform experiments and practices to use the results to solve broad-based Electronics and Telecommunication engineering problems.
- **Engineering tools:** Apply relevant Electronics and Telecommunications technologies and tools with an understanding of the limitations.
- **Lifelong learning:** Engage in independent and life-long learning activities in the context of technological changes also in the Electronics and Telecommunication engineering and allied industry

III Competency and Practical Skills

This practical is expected to develop the following skills for the industry-identified competency: '**Maintain VLSI based electronics circuits and equipments.'**

- Develop 'VHDL' code using EDA tools.
- Use Input/output port pins of FPGA.
- Interface FPGA KIT and desktop PC.

IV Relevant Course Outcome(s)

Maintain FPGA based circuits.

V Practical Outcome

Implement four Bit ALU or sequence generator using FPGA.

VI Relevant Affective domain related Outcome(s)

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

VII Minimum Theoretical Background

a) Arithmetic logic unit (ALU) :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.

Fig 16.1 & 16.2 shows how AND operation and OR operation is performed. Four bit two inputs and four bit one output.

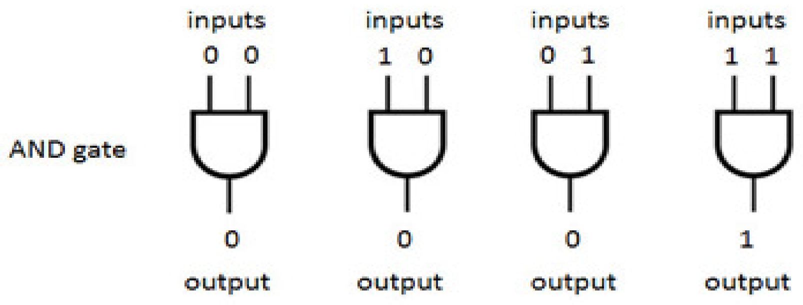


Fig. 16.1: AND operation.

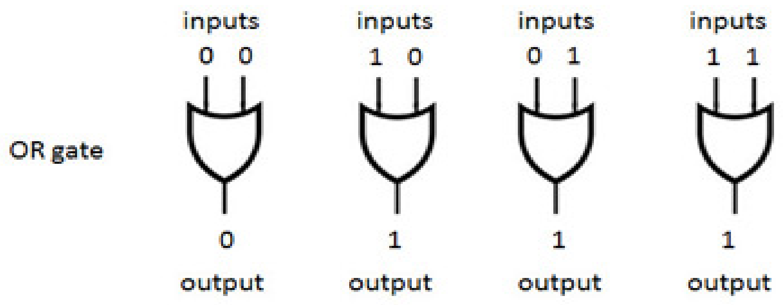


Fig. 16.2: OR operation.

Table 16.1 shows that ALU operation performed according to select signal. With this logic user can develop the VHDL code.

Table 16.1: Signal and operation.

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

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

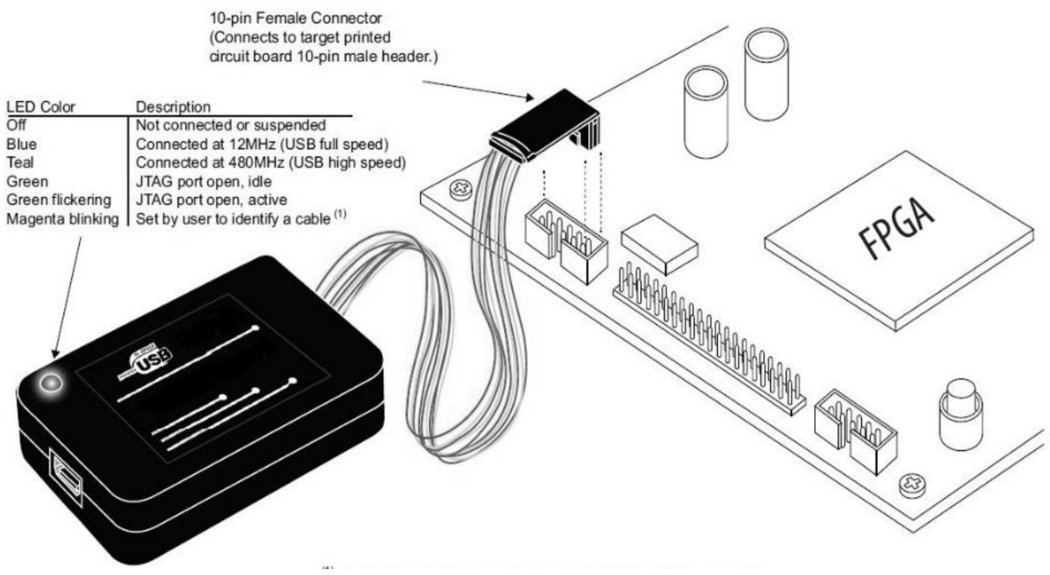
There are two types:

i. Moore machine: In a Moore machine, output depends only on the present state and not dependent on the input (x).

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

VIII Practical Circuit diagram:

a) Practical JTAG cable setup:-


10-pin Female Connector (Connects to target printed circuit board 10-pin male header.)
USB
FPGA

LED Color	Description
Off	Not connected or suspended
Blue	Connected at 12MHz (USB full speed)
Teal	Connected at 480MHz (USB high speed)
Green	JTAG port open, idle
Green flickering	JTAG port open, active
Magenta blinking	Set by user to identify a cable (1)

Fig. 16.3.a: Practical Setup with JTAG CABLE

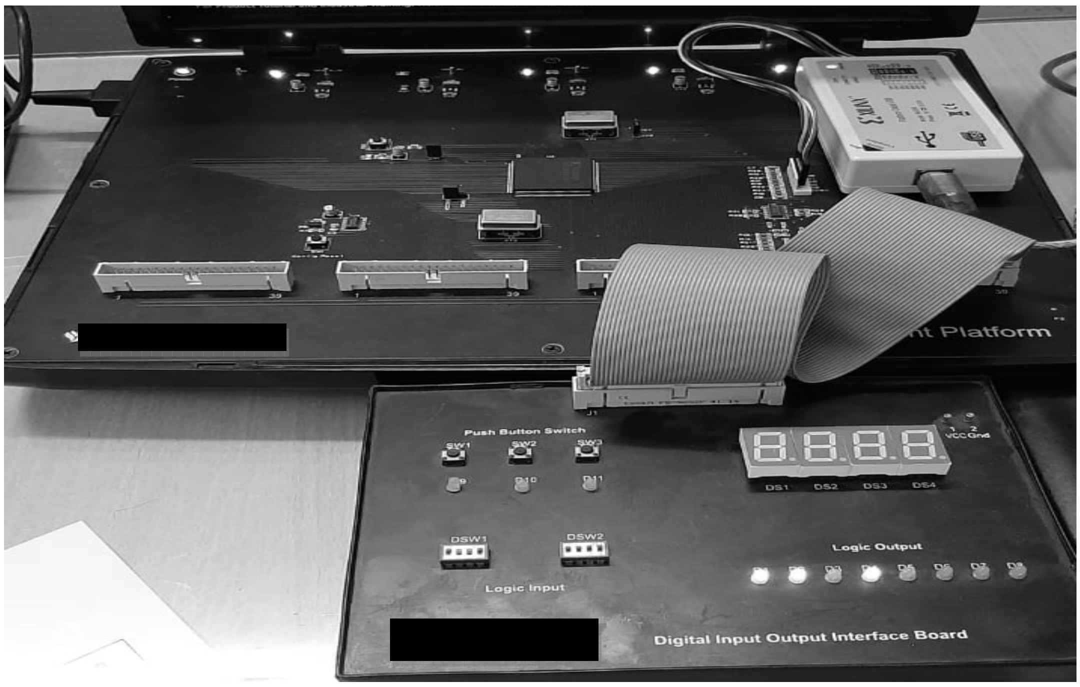
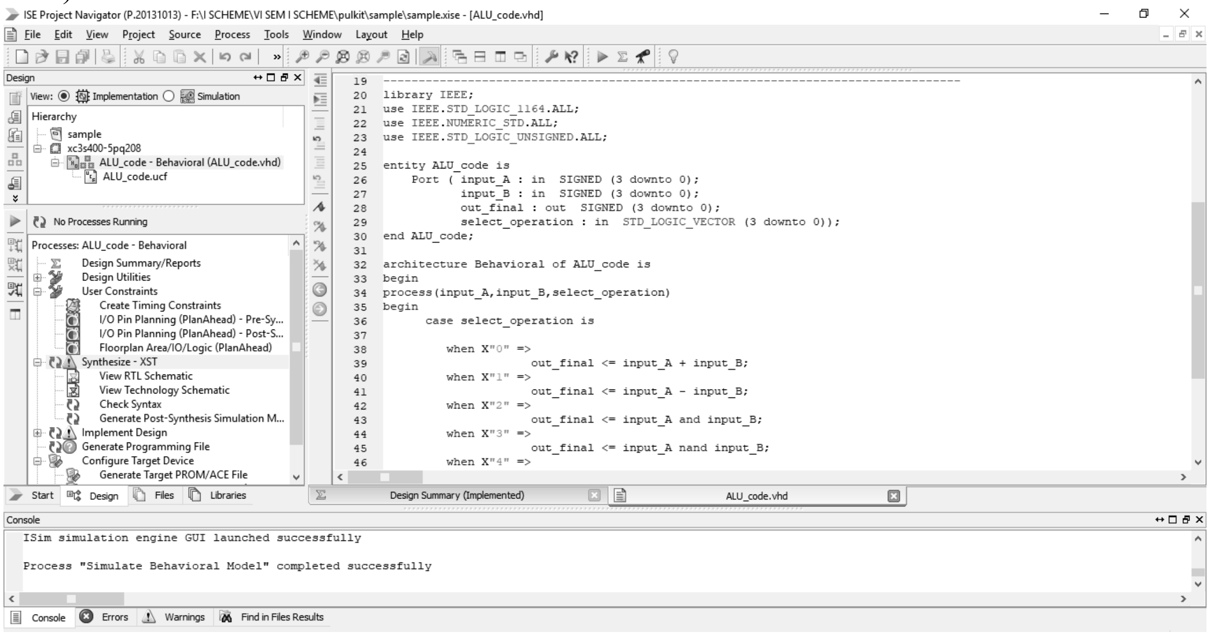


Fig. 16.3.b: Lab Practical Setup

b) Create four Bit ALU .vhd file :-

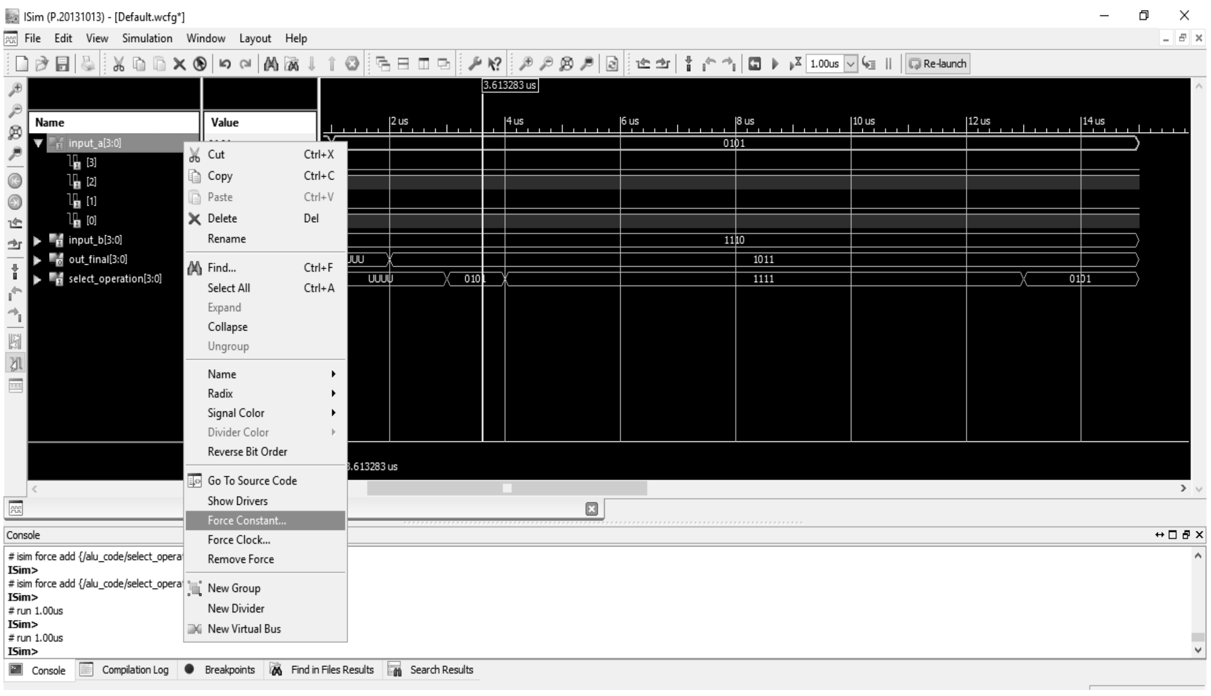

ISE Project Navigator (P.20131013) - F:\I SCHEME\VI SEM I SCHEME\pulkit\sample\sample.xise - [ALU_code.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
sample
xc3s400-5pq208
ALU_code - Behavioral (ALU_code.vhd)
ALU_code.ucf
No Processes Running
Processes: ALU_code - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Create Timing Constraints
I/O Pin Planning (PlanAhead) - Pre-Sy...
I/O Pin Planning (PlanAhead) - Post-S...
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation M...
Implement Design
Generate Programming File
Configure Target Device
Generate Target PROM/ACE File
Start Design Files Libraries
Design Summary (Implemented)
ALU_code.vhd
Console
Console Errors Warnings Find in Files Results

```
--------------------------------------------------------------------------------
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" =>
```

```
ISim simulation engine GUI launched successfully

Process "Simulate Behavioral Model" completed successfully
```

Fig. 16.4: .vhd file

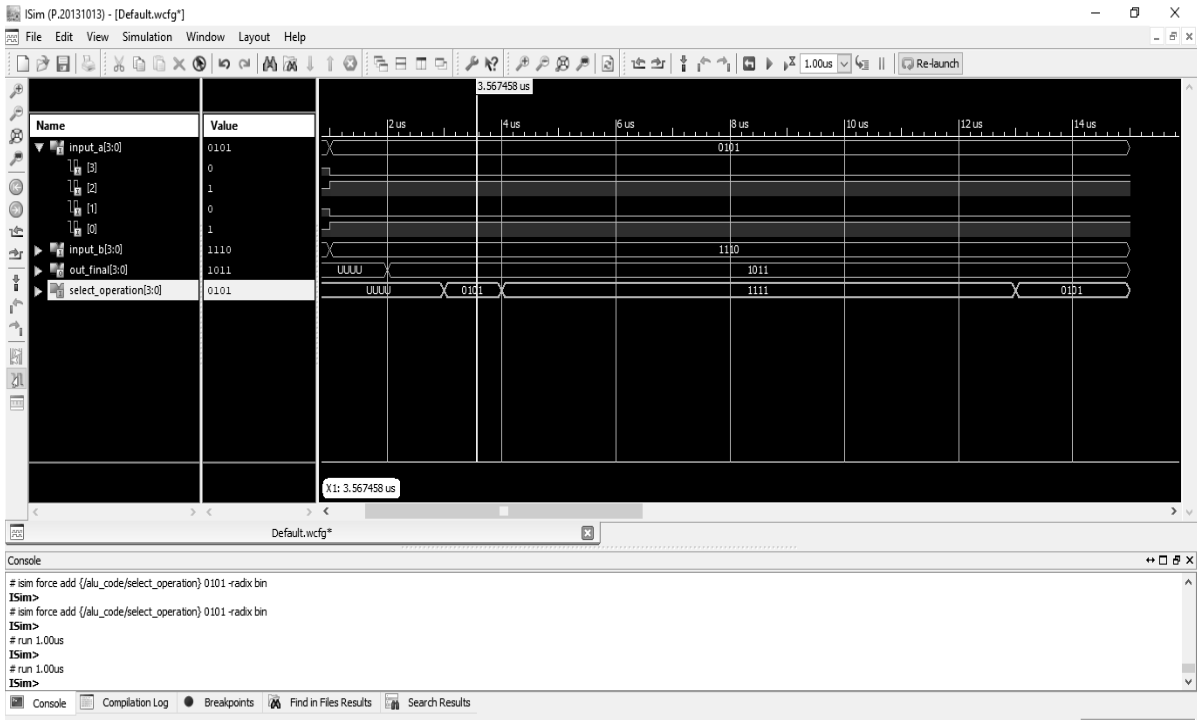
c) Apply entity values forcefully to generate simulation waveforms.



```
# isim force add {/alu_code/select_opera
ISim>
# isim force add {/alu_code/select_opera
ISim>
# run 1.00us
ISim>
# run 1.00us
ISim>
```

Fig. 16.5: Forceful Simulation

d) Bit ALU Simulation waveforms-



```
# isim force add {/alu_code/select_operation} 0101 -radix bin
ISim>
# isim force add {/alu_code/select_operation} 0101 -radix bin
ISim>
# run 1.00us
ISim>
# run 1.00us
ISim>
```

Fig. 16.6: Simulation Output

e) Actual Experimental set up used in laboratory

IX 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	Loaded with open source IDE, simulation and program downloading software.	1 No.

X 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.

XI 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.
   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.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. 16.5.
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 "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.16.6. where are able to simulate designs and check for errors.
10. After simulation implement 4-bit ALU using FPGA development board.
11. 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 Config. 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 4-bit ALU using if else statement.

Step 1: Develop VHDL code for four Bit ALU.

VHDL Code using if else statement

XII Resources Used

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

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

....
....
....
....
....

XIV Precautions Followed (use blank sheet provided if space not sufficient)

....
....
....
....
....

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

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

XVI Result (Output of the Program)

....
....
....
....
....

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

....
....
....
....
....

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

....
....
....
....
....

XIX 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 four Bit ALU using Case statement.
2. Design VHDL Code using FPGA board for sequence generator using if else statement.

[Space for Answers]

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

XX References / Suggestions for further reading

1. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_usb_blstr_ii_cable.pdf
2. https://www.xilinx.com/support/documentation/data_sheets/ds610.pdf
3. https://forums.xilinx.com/
4. https://en.wikipedia.org/wiki/Arithmetic_logic_unit
5. https://study.com/academy/lesson/how-to-design-sequence-detectors-steps-example.html

XXI Assessment Scheme

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%**
4	Correct corrections to FPGA Board	20%
5	Relevance of output of the problem definition.	15%
6	Timely Submission of report, Answer to sample questions.	05%
	Total (25 Marks)	**100%**

Name of student Team Member

1. …………………………………..
2. …………………………………..
3. …………………………………..
4. …………………………………..

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