SVR ENGINEERING COLLEGE

Approved by AICTE & Permanently Affiliated to JNTUA

Ayyalurmetta, Nandyal – 518503. Website: www.svrec.ac.in

Department of Electronics and Communication Engineering

(15A04712) VLSI & Embedded Systems Laboratory

IV B. Tech (ECE) I Semester 2021-22

STUDENT NAME	
ROLL NUMBER	
SECTION	

SVR ENGINEERING COLLEGE

Approved by AICTE & Permanently Affiliated to JNTUA

Ayyalurmetta, Nandyal – 518503. Website: www.svrec.ac.in

DEPARTMENT OF

ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

ACADEMIC YEAR: 2021-22

This is to certify that the bonafide record work done by Mr./Ms._____ bearing H.T.NO. _____ of IV B. Tech I Semester in the VLSI & Embedded Systems Laboratory.

Faculty In-Charge

Head of the Department

ECE DEPT VISION & MISSION PEOs and PSOs

Vision

To produce highly skilled, creative and competitive Electronics and Communication Engineers to meet the emergingneeds of the society.

Mission

- Impart core knowledge and necessary skills in Electronics and Communication Engineering Through innovative teaching and learning.
- Inculcate critical thinking, ethics, lifelong learning and creativity needed for industry and society.
- Cultivate the students with all-round competencies, for career, higher education and self-employability.

I. PROGRAMME EDUCATIONAL OBJECTIVES (PEOS)

PEO1: Graduates apply their knowledge of mathematics and science to identify, analyze and solve problems in the field of Electronics and develop sophisticated communication systems.

PEO2: Graduates embody a commitment to professional ethics, diversity and social awareness in theirprofessional career.

PEO3: Graduates exhibit a desire for life-long learning through technical training and professional activities.

II. PROGRAM SPECIFIC OUTCOMES (PSOS)

PSO1: Apply the fundamental concepts of electronics and communication engineering to design a variety of components and systems for applications including signal processing, image processing, communication, networking, embedded systems, VLSI and control system.

PSO2: Select and apply cutting-edge engineering hardware and software tools to solve complex Electronics and Communication Engineering problems.

III. PROGRAMME OUTCOMES (PO'S)

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, andin multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects andin multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

IV. COURSE OBJECTIVES:

- To design and draw the internal structure of the various digital integrated circuits
- To develop VHDL/Verilog HDL source code, perform simulation using relevant simulator and analyze the obtained simulation results using necessary synthesizer.
- To verify the logical operations of the digital ICs (Hardware) in the laboratory.
- Learn and understand how to configure EK-TM4C123GXL Launchpad digital I/O pins

V. COURSE OUTCOMES:

After the completion of the course students will be able to

Course Outcomes	Course Outcome statements	BTL
CO1	Design and draw the internal structure of the various digital integrated circuits.	L1
CO2	Develop VHDL/Verilog HDL source code, perform simulation using relevant simulator andanalyze the obtained simulation results using necessary synthesizer.	L2
CO3	Verify the logical operations of the digital IC"s (Hardware) in the laboratory.	L3
CO4	Learn and understand how to configure EK-TM4C123GXL Launchpad digital I/O pins, onboard LED, RTC (Real- Time Clock), Pulse Width Module, Launchpad to PC terminal and send an echo of the data input back to the PC using UART.	L4
CO5	Learn and understand interfacing of CC3100 WiFi module with EKTM4C123GXL Launchpad and configuration of static IP address for CC3100 booster pack.	L5

VI. COURSE MAPPING WITH PO'S AND PEO'S:

CourseTitle	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
VLSI & EMBEDDED SYSTEMS LABORATORY	3	3	2	3	3	2	2	1	2	2	2	2	3	2

VII. MAPPING OF COURSE OUTCOMES WITH PEO'S AND PO'S:

Course Title	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
CO1	3	3	3	3	3	2	1	1	1	2	1	2	3	2
CO2	2	2	2	2	2	1	2	1	2	2	1	2	2	2
CO3	3	3	3	3	3	3	2		2	2	1	2	3	2
CO4	2	3	2	3	2	2	2	1	2	1	2	2	2	2
CO5	3	3	2	2	3	2	1	1	1	2	2	1	3	1

LABORATORY INSTRUCTIONS

1. While entering the Laboratory, the students should follow the dress code. (Wear shoes and White apron,Female Students should tie their hair back).
2. The students should bring their observation book, record, calculator, necessary stationery items and graph sheets if any for the lab classes without which the students will not be allowed for doing the experiment.
3. All the Equipment and components should be handled with utmost care. Any breakage or damage will becharged.
4. If any damage or breakage is noticed, it should be reported to the concerned in charge immediately.
5. The theoretical calculations and the updated register values should be noted down in the observation book and should be corrected by the lab in-charge on the same day of the laboratory session.
6. Each experiment should be written in the record note book only after getting signature from the lab in-charge in the observation notebook.
7. Record book must be submitted in the successive lab session after completion of experiment.
8. 100% attendance should be maintained for the laboratory classes.

Precautions.

1. Check the connections before giving the supply.
2. Observations should be done carefully.

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

B. Tech IV-ISem. (ECE) **L T P C**

0 0 4 2

15A04712 VLSI & EMBEDDED SYSTEMS LABORATORY

Note: The students are required to perform any Six Experiments from each Part of the following.

Part-A: VLSI Lab

Course Objective:

- To design and draw the internal structure of the various digital integrated circuits
- To develop VHDL/Verilog HDL source code, perform simulation using relevant simulator and analyze the obtained simulation results using necessary synthesizer.
- To verify the logical operations of the digital ICs (Hardware) in the laboratory.

Course Outcome:

After completion of the course the students will be able to

- Design and draw the internal structure of the various digital integrated circuits
- Develop VHDL/Verilog HDL source code, perform simulation using relevant simulator andanalyze the obtained simulation results using necessary synthesizer.
- Verify the logical operations of the digital IC"s (Hardware) in the laboratory.

Note: For the following list of experiments students are required to do the following.

Target Device Specifications

- Simulation
- Synthesize the design
- Generate RTL Schematic.
- Generate Technology Map.
- Generate Synthesis report.
- Design Summary.

List of Experiments:

Note: Use VHDL/ Verilog HDL

1. Realization of Logic Gates.

2. 3- to - 8Decoder- 74138.

3. 8 x 1 Multiplexer-74151 and 2 x 4 De-multiplexer-74155.

4. 4-Bit Comparator-7485.

5. D Flip-Flop-7474.

6. Decade counter-7490.

7. Shift registers-7495.

8. ALU Design.

Part B : Embedded C Experiments using TM4C processor:

1. Learn and understand how to configure EK-TM4C123GXL Launchpad digital I/O pins. Write a C program for configuration of GPIO ports for Input and output operation (blinking LEDs, push buttons interface).

Exercises:

a) Modify the code to make the red LED of EK-TM4C123GXL Launchpad blink.

b) Modify the code to make the green and red LEDs blink: I. Together II. Alternately

c) Alter the code to turn the LED ON when the button is pressed and OFF when it is released.

d) Modify the delay with which the LED blinks.

e) Alter the code to make the green LED stay ON for around 1 second every time the button is pressed. f) Alter the code to turn the red LED ON when the button is pressed and the green LED ON when the button is released.

2. Learn and understand Timer based interrupt programming. Write a C program for EK-TM4C123GXL Launchpad and associated Timer ISR to toggle onboard LED using interrupt programming technique. Exercises:

a) Modify the code for a different timer toggling frequency.

b) Write the code to turn on interrupt globally.

3. Configure hibernation module of the TM4C123GH6PM microcontroller to place the device in low power state and then to wake up the device on RTC (Real- Time Clock) interrupt.

Exercises:

a) Write a program to configure hibernation mode and wake up the EK-TM4C123GXL Launchpad when onboard switch SW2 is pressed.

4. Configure in-build ADC of TM4C123GH6PM microcontroller and interface potentiometer with EK-TM4C123GXL Launchpad to observe corresponding 12- bit digital value.

Exercises:

a) Tabulate ten different position of the Potentiometer and note down the Digital value and calculate the equivalent analog value.

b) Use the ADC to obtain the analog value from the internal temperature sensor.

c) Configure Dual ADC modules to read from 2 analog input (could be from 2 potentiometers)

d) What are the trigger control mechanism for this ADC?

e) What does the resolution refer on ADC Specification?

f) The current sampling method is single ended sampling. This ADC could also be configured to do differential sampling. What is the difference between the two methods of sampling?

5. Learn and understand the generation of Pulse Width Module (PWM) signal by configuring and programming the in-build PWM module of TM4C123GH6PM microcontroller.

Exercises:

a) Change the software to output a set Duty Cycle, which can be user programmed.

b) Change the frequency of the PWM Output from 6.25 KHz to 10 KHz and do the tabulation again.

c) Generate Complementary signals, route it to two pins, and observe the waveforms.

d) What is dead band generation mean and where is it applied?

e) Is it possible to construct a DAC from a PWM? Identify the additional components and connection diagram for the same.

f) Sketch the gate control sequence of 3 phase Inverter Bridge and how many PWM generator blocks are required? Can we generate this from TIVA Launchpad?

6. Configure the PWM and ADC modules of TM4C123GH6PM microcontroller to control the speed of a DC motor with a PWM signal based on the potentiometer output.

Exercises:

a) With the same ADC input configure 2 PWM generator modules with 2 different frequencies.

b) Read the Internal temperature sensor and control a DC Motor that could be deployed in fan Controller by observing the unit or ambient temperature.

c) What is the resolution of the PWM in this experiment?

d) What would be the maximum frequency that can be generated from the PWM generator?

e) Briefly explain an integrated application of ADC and PWM based control.

7. Learn and understand to connect EK-TM4C123GXL Launchpad to PC terminal and send an echo of the data input back to the PC using UART.

Exercises:

a) Change the baud rate to 19200 and repeat the experiment.

b) What is the maximum baud rate that can be set in the UART peripheral of TIVA?

c) Modify the software to display “Switch pressed” by pressing a user input switch on the Launchpad.

8. Learn and understand interfacing of accelerometer in Sensor Hub Booster pack with EK-TM4C123GXL Launchpad using I2C.

Exercises:

a) Make a LED ON when the acceleration value in the x axis crosses a certain limit, say +5.

b) What is the precaution taken in this experiment in order to avoid the overflow of UART buffer?

c) Change the value of PRINT_SKIP_COUNT to 100 and see the difference in the output.

d) Change MPU9150_ACCEL_CONFIG_AFS_SEL_2G to MPU9150_ACCEL_CONFIG_AFS_SEL_4G on line 461 of the same source file and Observe the difference.

9. USB bulk transfer mode: Learn and understand to transfer data using bulk transfer mode with the USB2.0 peripheral of the TM4C123GH6PM device.

Exercises:

a) What are the different modes offered by USB 2.0?

b) What are the typical devices that use Bulk transfer mode?

10. Learn and understand to find the angle and hypotenuse of a right angle triangle using IQmath library of TivaWare.

Exercises:

a) Change the base and adjacent values in the program to other values, build the program and observe the values in the watch window.

b) Open IQmathLib.h and browse through the available functions. What function is to be used if the IQ number used in the program is to be converted to a string?

11. Learn and understand interfacing of CC3100 WiFi module with EKTM4C123GXL Launchpad and configuration of static IP address for CC3100 booster pack.

Exercises:

a) Try pinging the same IP address before connecting to the Access Point (AP) and note down the observation.

b) What is the difference between static IP address and dynamic IP address?

12. Configure CC3100 Booster Pack connected to EK-TM4C123GXL Launchpad as a Wireless Local Area Network (WLAN) Station to send Email over SMTP.

Exercises:

a) In the terminal output window, we have received a debug message “Pinging…!”. Search in the code and change the message to “Pinging the website”. Repeat the experiment to observe this change in the Serial Window.

b) In line no:62 of main. C replace www.ti.com with any non-existing web address and repeat the experiment and observe what happens

c) In line no: 62 of main. C replace again with www.ti.comand repeat the experiment.

d) Identify the code that helps in establishing connection over SMTP. Modify the code to trigger E-mail application based upon external analog input.

e) How to configure the AP WLAN parameters and network parameters (IP addresses and DHCP parameters) using CC3100 API.

13. Configure CC3100 Booster Pack connected to EK-TM4C123GXL Launchpad as a HTTP server.

Exercises:

a) Where are the webpages stored in the CC3100?

b) What happens if we try to access a webpage, which is not there inside the CC3100?

c) List 3 applications with a 3 to 4-line brief description that you think can be performed with this experimental setup.

INDEX

Mr./Ms._____ Roll Number:_____

S. No	Date	Title of the Experiment	Text Book	Page No.	Marks	Signature of the Staff
PART-A						
1		Introduction to ISE Xilinx Software	T1			
2		Introduction To TIVA MICROCONTROLLER	T2			
PART – A : LIST OF EXPERIMENTS						
1		Introduction to ISE Xilinx Software and Spartan 3E FPGA				
2		REALIZATION OF LOGIC GATES				
3		3 x 8 DECODER – IC74138				
4		8 X 1 MULTIPLEXER-74151 AND 1 X 4 DEMULTIPLEXER-74155				
5		4-BIT COMPARATOR-7485				
6		D Flip-Flop-7474				
7		ALU				
PART – B : LIST OF EXPERIMENTS						
8		BLINKING LED'S AND PUSH BUTTON INTERFACE USING TM4CGH6PM				
9		TIMER BASED INTERRUPT PROGRAMMING USING TM4C123GXL				
10		HIBERNATION MODULE FOR TM4C123GH6PM MICROCONTROLLER				

11		IN-BUILD ADC OF TM4C123GH6PM & POTENTIOMETER WITH TM4C123GXL				
12		PWM AND ADC MODULES OF TM4C123GH6PM MICROCONTROLLE R				
13		SENSORHUB BOOSTER PACK WITH TM4C123GXL				
PART C - Additional Experiment Beyond the Curriculum						
15		JK-FLIPFLOP USING VHDL				
16		ECHO OF THE DATA INPUT BACK TO THE PC USING UART				

TEXT BOOKS:

T1: 1. Kamran Eshraghian, Eshraghian Douglas and A. Pucknell, "Essentials of VLSI circuits and systems", PHI, 2013 Edition.

2. K.Lal Kishore and V.S.V. Prabhakar, "VLSI Design", IK Publishers

T2: 1. Embedded Systems: Real-Time Interfacing to ARM Cortex-M Microcontrollers, 2014, Create space publications ISBN: 978-1463590154.

2. Embedded Systems: Introduction to ARM Cortex - M Microcontrollers, 5th edition Jonathan W Valvano, Createspace publications ISBN-13: 978-1477508992

3. Embedded Systems 2E Raj Kamal, Tata McGraw-Hill Education, 2011 ISBN

0070667640, 9780070667648

Introduction to ISE Xilinx Software and Spartan 3E FPGA

AIM: TO Become familiar with the Xilinx ISE Foundation software package and Simulate and verify functionality of XOR Gate

SOFTWARE REQUIRED:

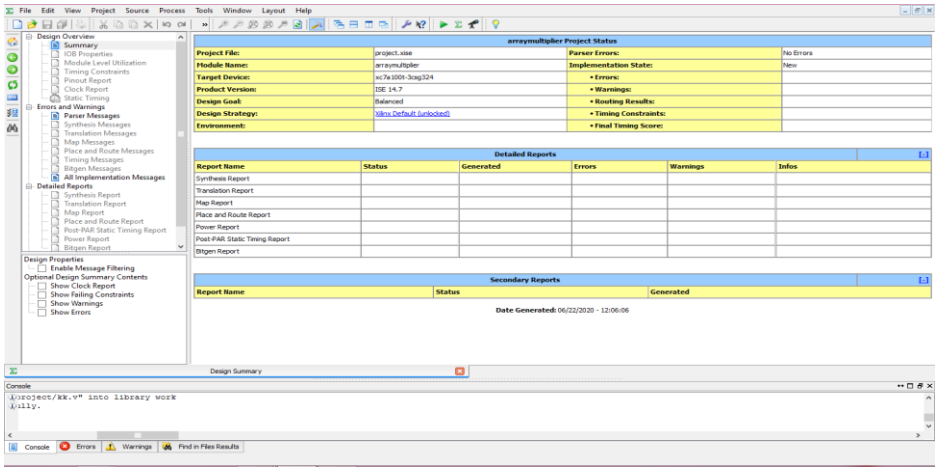
1. Xilinx ISE Foundation Series
2. Personal Computer
3. Spartan 3E

PROCEDURE:

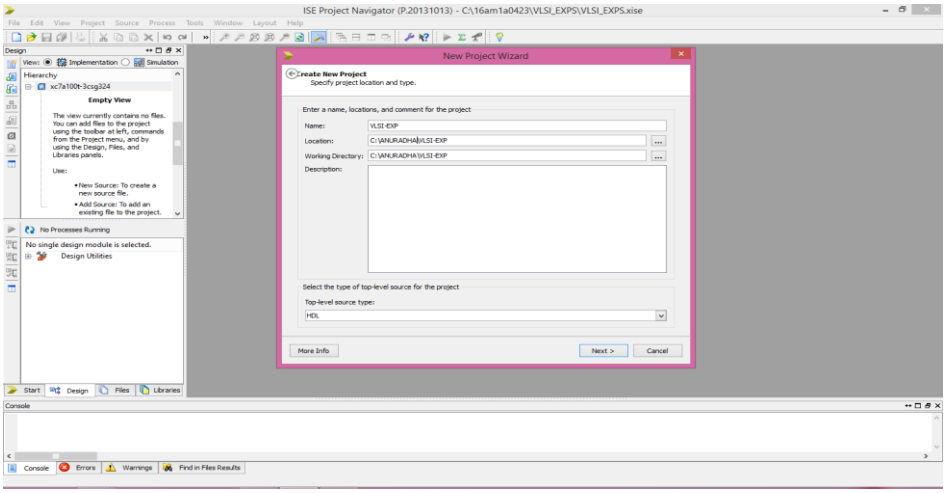
1. Read the Introductory Tutorial for Xilinx ISE Foundation v10.1
2. Go through the steps of the tutorial and implement the given logic function.
3. Implement the XOR Gate.

Introductory Tutorial for Xilinx ISE Foundation

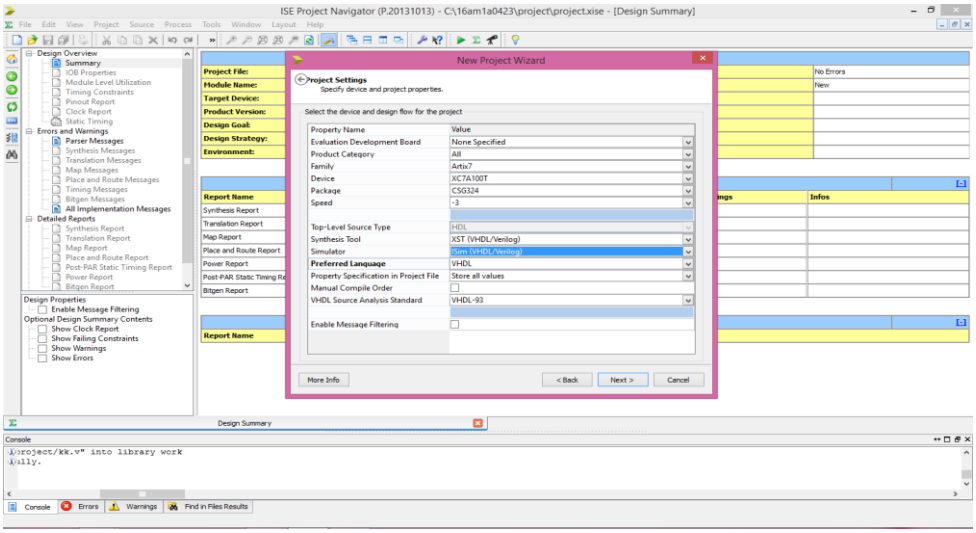
Open Xilinx Project Navigator.



Create new project.

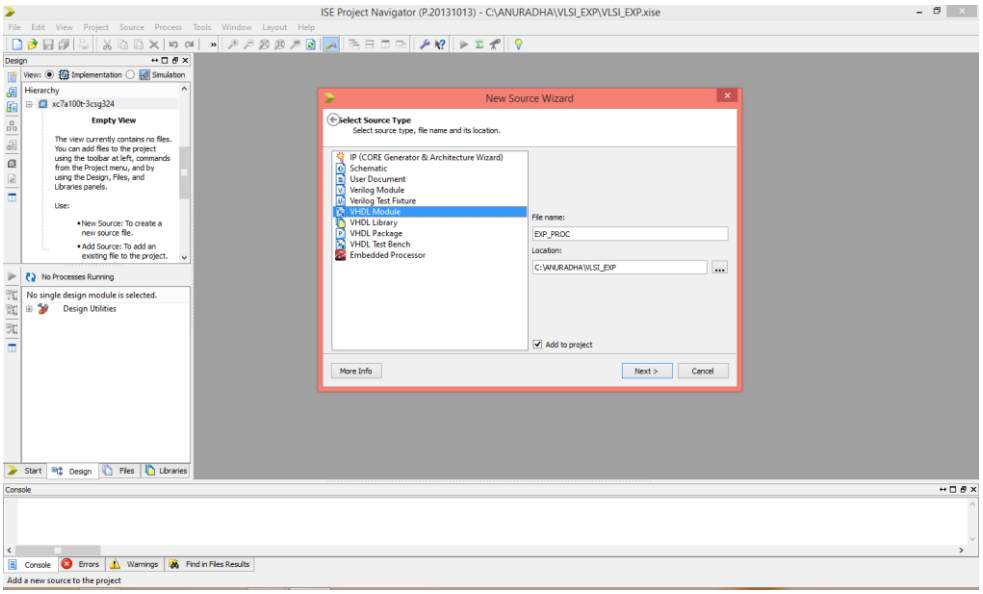


Specify device properties.

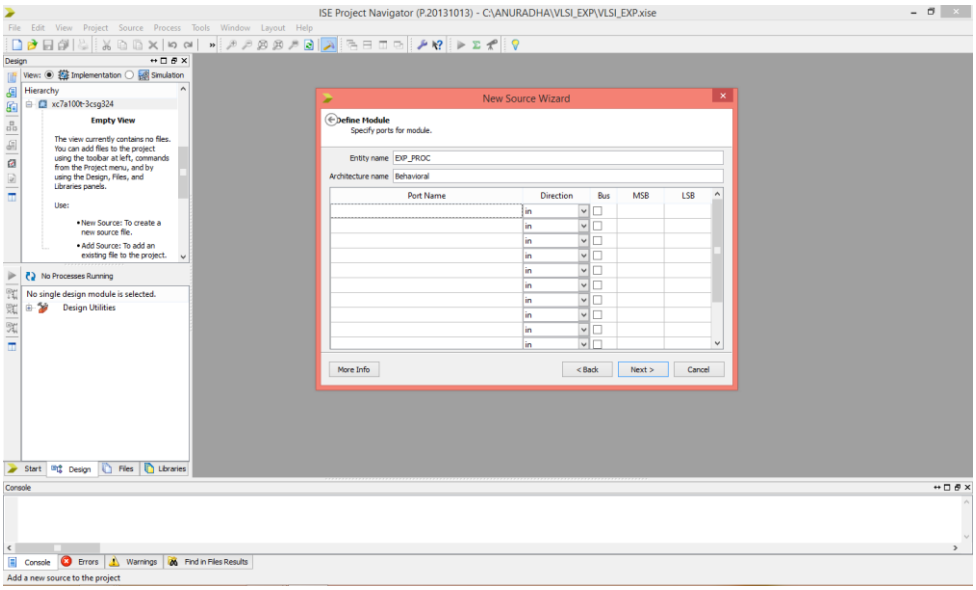


Add new source to the project.

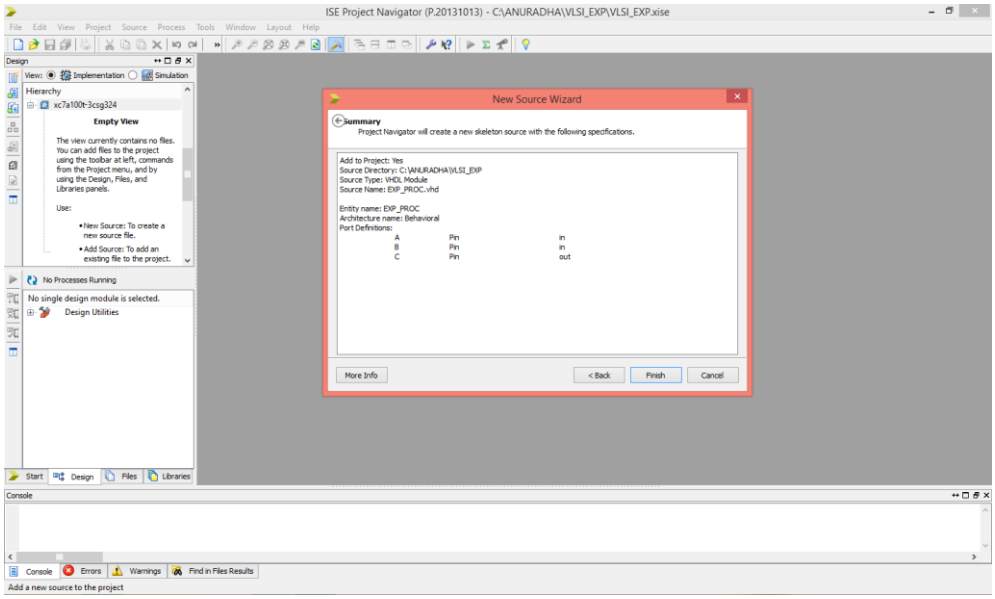
Select source type [VHDL or Verilog].



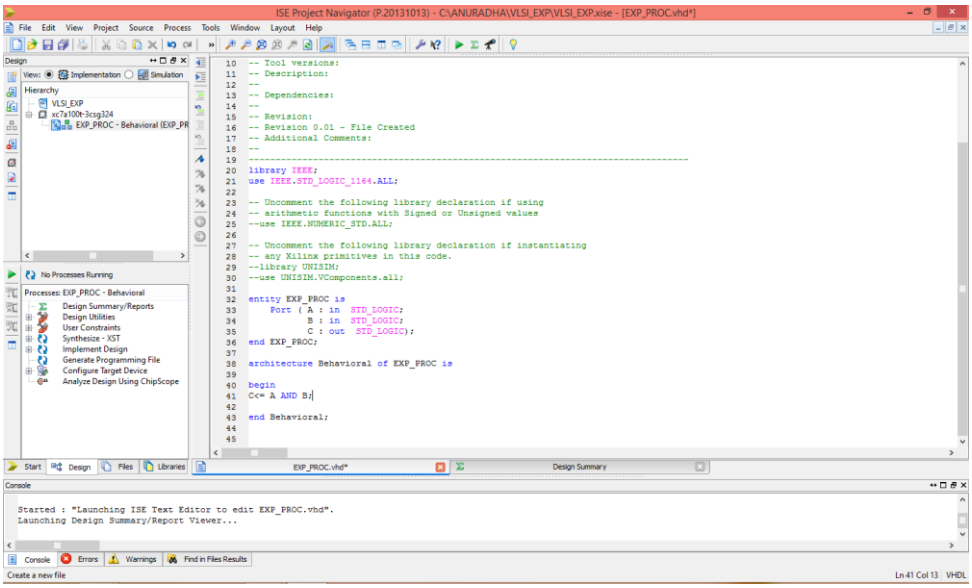
Specify input and output port names.



New source summary window will appear as shown below.

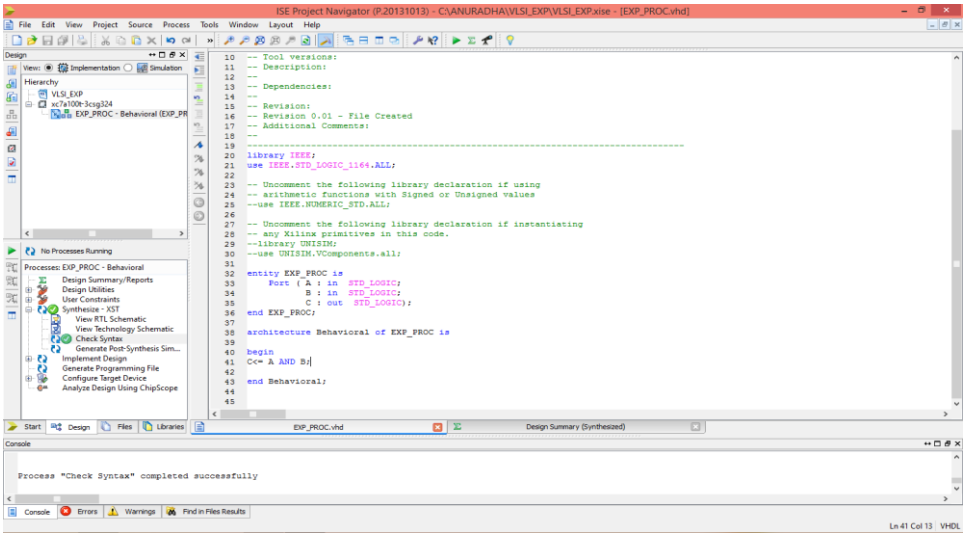


Enter program code in the HDL editor window.

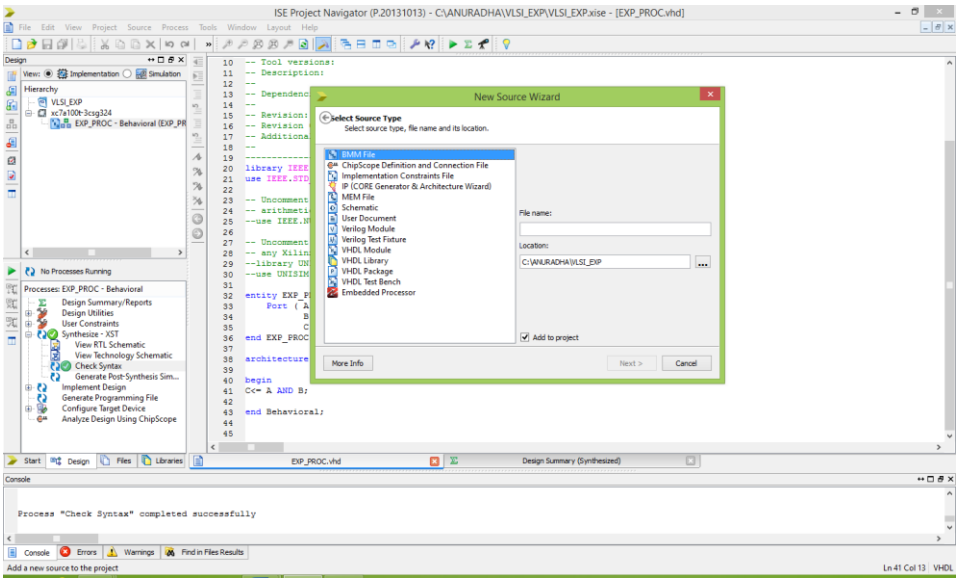


Save the program and synthesize the process.

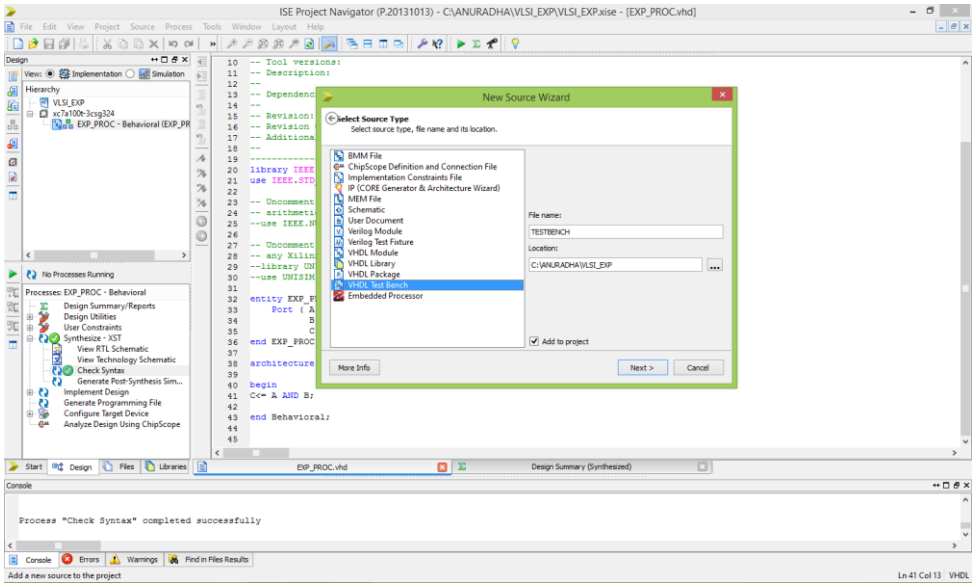
The design summary window after synthesis appears as shown below.



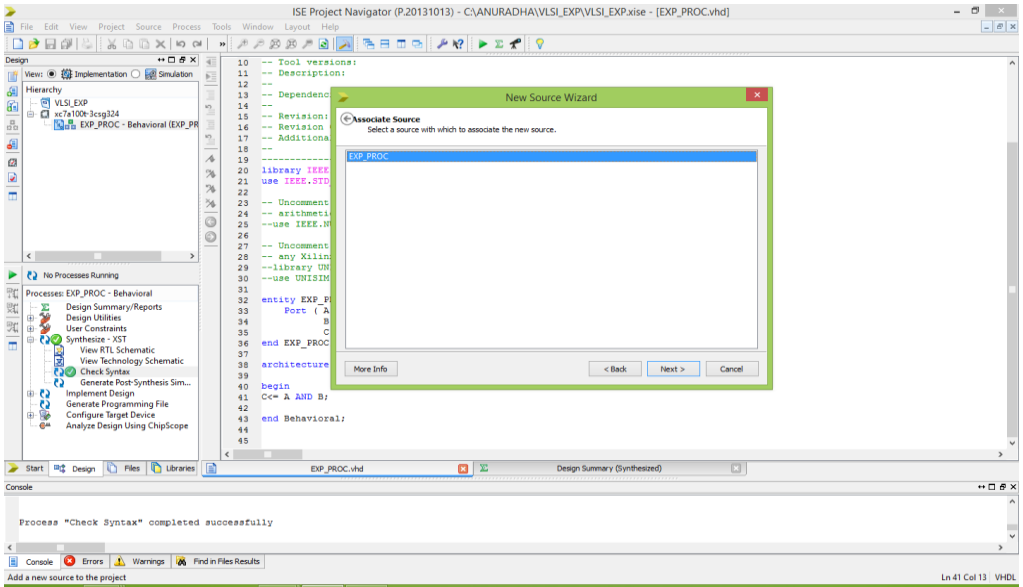
Create another new source.



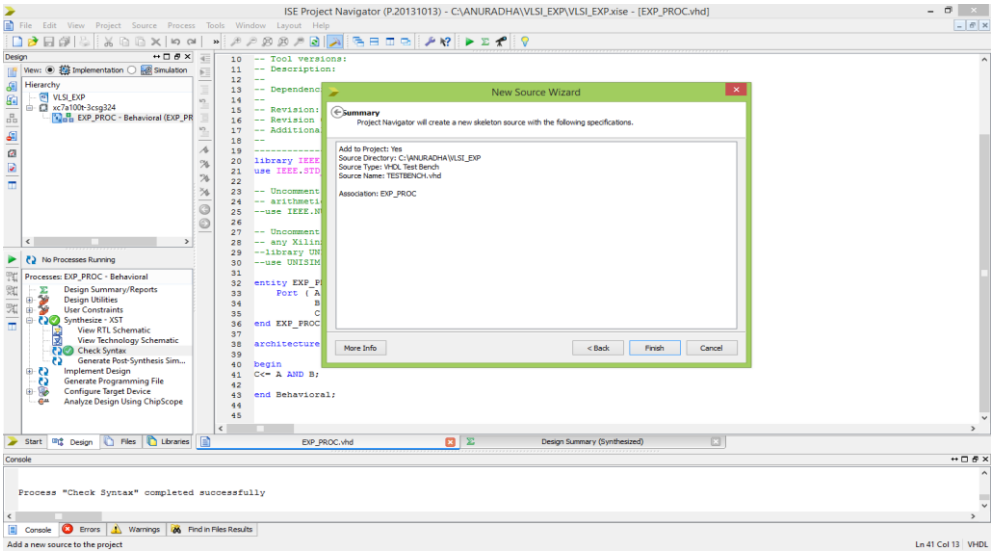
Select source type as Test bench wave form.



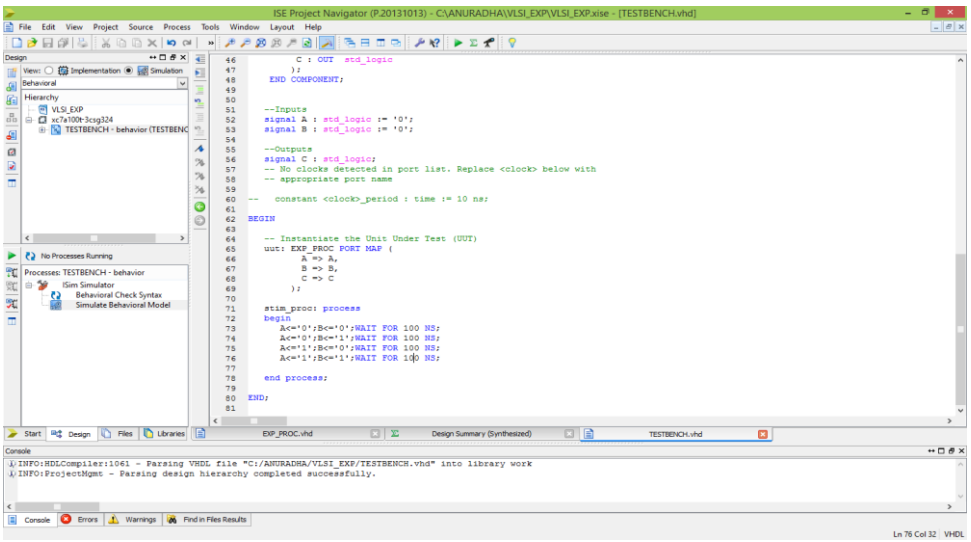
Associate the test bench to the source.



Test bench wave form summary window will appear as shown below.

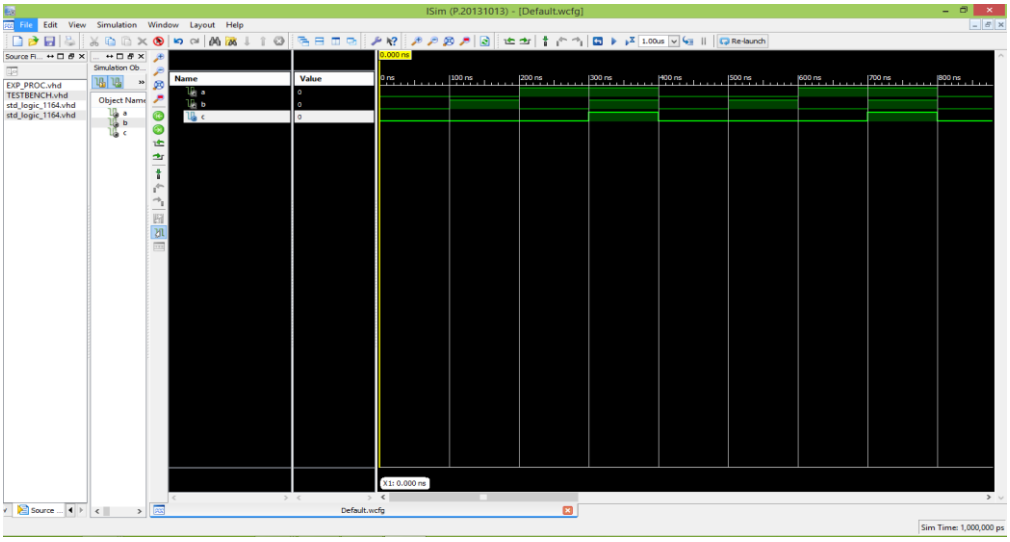


Enter program code for Assign clock and timing details Give the input for the source.

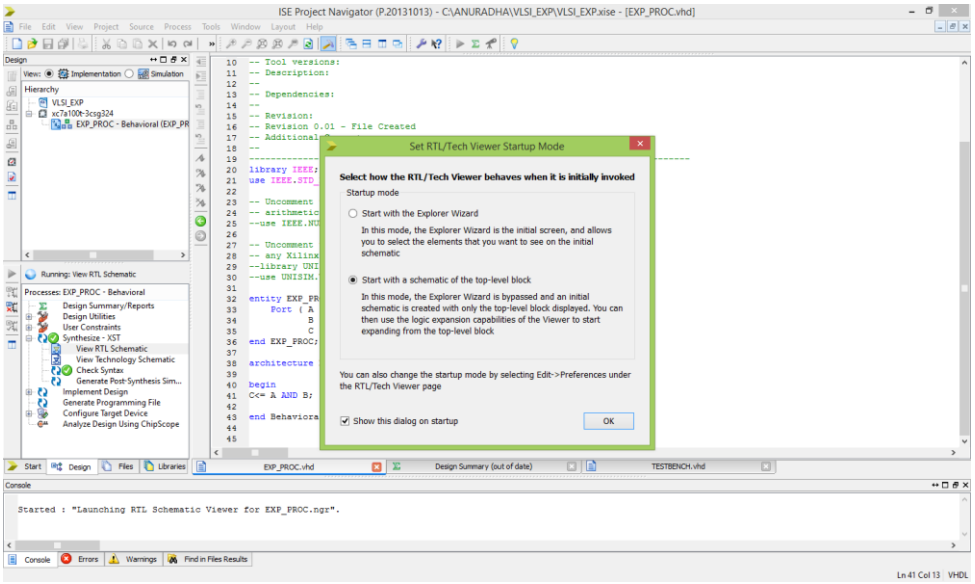


Save the input waveforms and perform behavioral simulation.

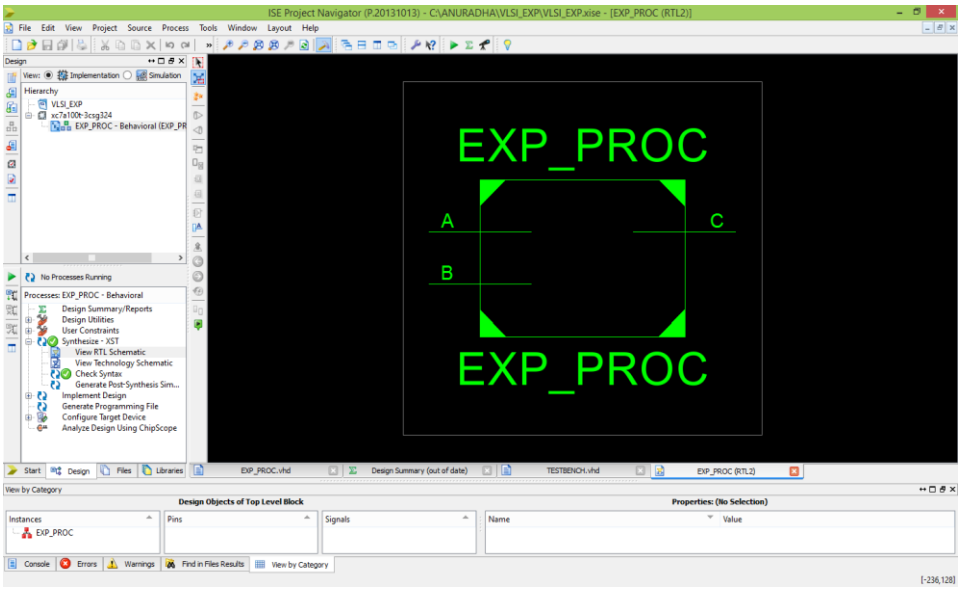
The simulated output waveforms window will be as shown below.

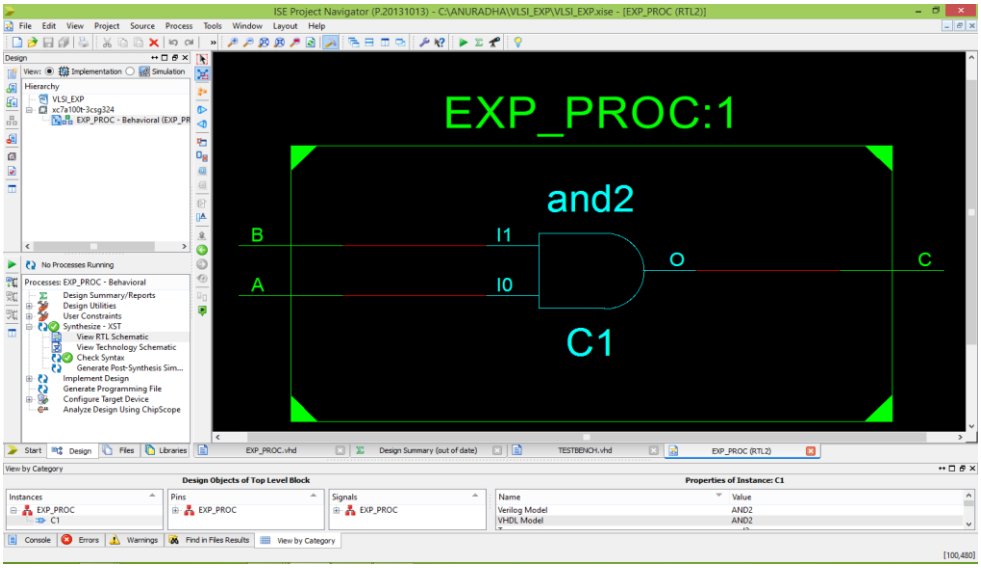


#Select View RTL Schematic for view



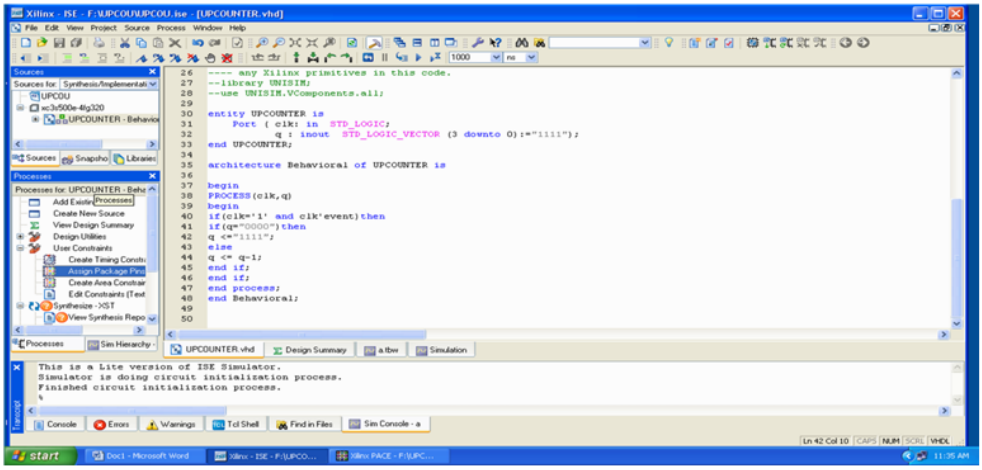
#RTL View for given testbench as sho in below.

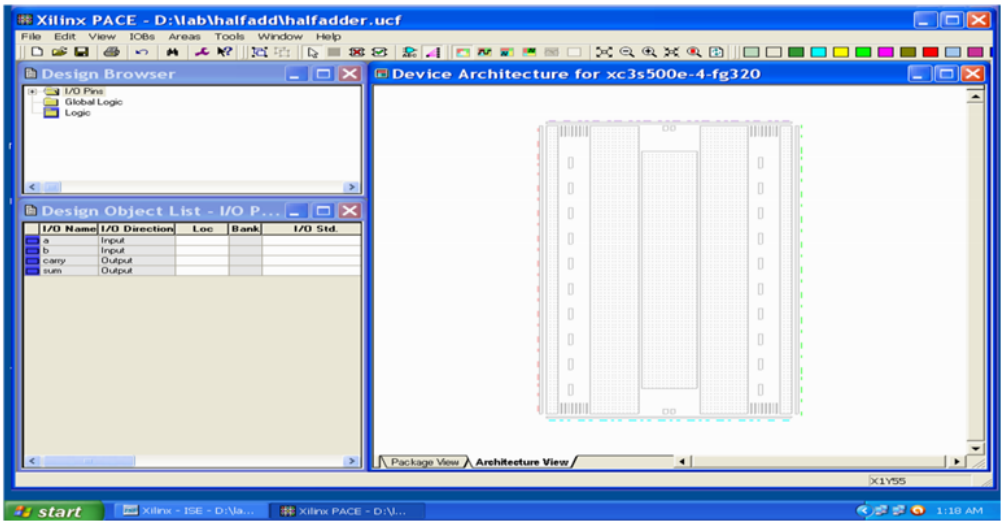


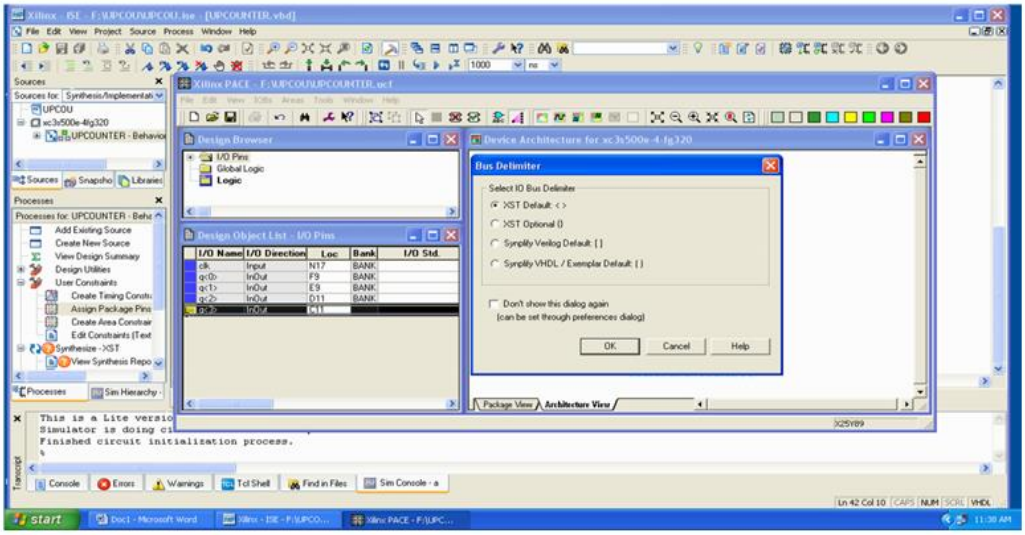


 # Dumping process:

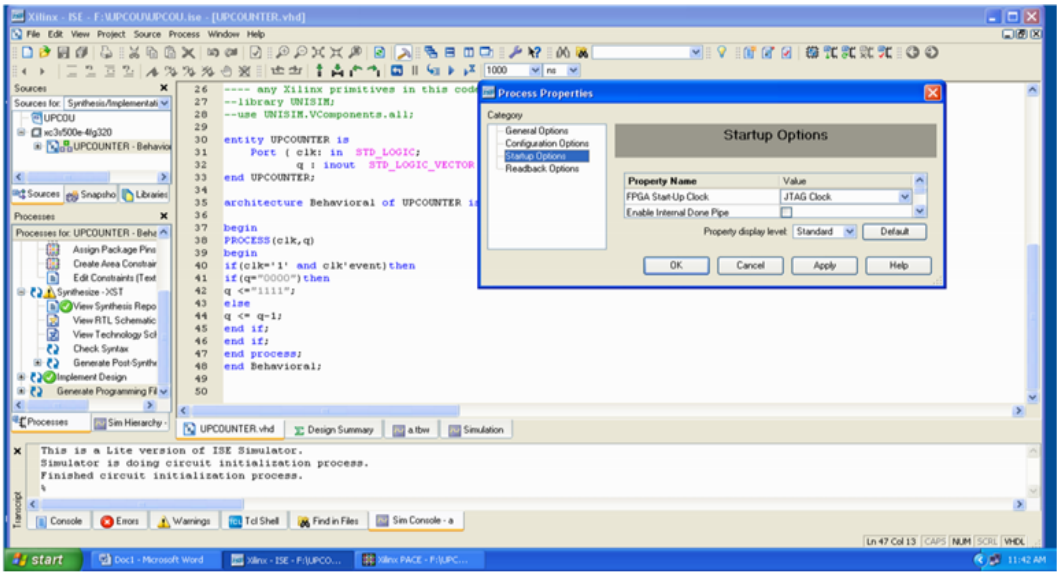
Assign pin numbers to input and output ports.







Select JTAG clock in the startup options of process properties by right clicking on ‘Generate programming file.



EXPERIMENT – 1
REALIZATION OF LOGIC GATES

AIM: To design and simulate logic gates using VHDL

SOFTWARE REQIURED:

1. Personal computer
2. ISE Xilinx software

HARDWARE REQUIRED:

1. SPARTAN – 3E KIT

THEORY:

A logic gate is an elementary building block of a digital circuit. Most logic gates have two inputs and one output. At any given moment, every terminal is in one of the two binary conditions *low* (0) or *high* (1), represented by different voltage levels. The logic state of a terminal can, and generally does, change often, as the circuit processes data. In most logic gates, the low state is approximately zero volts (0 V), while the high state is approximately five volts positive (+5 V). There are seven basic logic gates: AND, OR, XOR, NOT, NAND, NOR, and XNOR.

The **AND** gate acts in the same way as the logical "and" operator. The output is "true" when both inputs are "true." Otherwise, the output is "false."

The ***OR** gate* gets its name from the fact that it behaves after the fashion of the logical inclusive "or." The output is "true" if either or both of the inputs are "true." If both inputs are "false," then the output is "false."

A logical *inverter*, sometimes called a ***NOT** gate* to differentiate it from other types of electronic inverter devices, has only one input. It reverses the logic state.

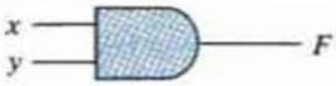
The ***NAND** gate* operates as an AND gate followed by a NOT gate. It acts in the manner of the logical operation "and" followed by negation. The output is "false" if both inputs are "true." Otherwise, the output is "true."

The ***NOR** gate* is a combination OR gate followed by an inverter. Its output is "true" if both inputs are "false." Otherwise, the output is "false."

The ***XOR (exclusive-OR)** gate* acts in the same way as the logical "either/or." The output is "true" if either, but not both, of the inputs are "true." The output is "false" if both inputs are "false" or if both inputs are "true." Another way of looking at this circuit is to observe that the output is 1 if the inputs are different, but 0 if the inputs are the same.

The ***XNOR (exclusive-NOR)** gate* is a combination XOR gate followed by an inverter. Its output is "true" if the inputs are the same, and "false" if the inputs are different.

The following figure/table shows the circuit symbols and logic combinations of different logic gates.

Name	Graphic symbol	Algebraic function	Truth table
AND		$F = xy$	$x\ y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 1$
OR		$F = x + y$	$x\ y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 1$
Inverter		$F = x'$	$x \mid F$ $0 \mid 1$ $1 \mid 0$
NAND		$F = (xy)'$	$x\ y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 0$
NOR		$F = (x + y)'$	$x\ y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 0$
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	$x\ y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 0$
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	$x\ y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 1$

PROGRAM IN VHDL:

AND GATE:

DATAFLOW:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity AND_LOGIC is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           C : out  STD_LOGIC);
end AND_LOGIC;
architecture Dataflow of AND_LOGIC is
begin
```

```
C<= A AND B;
end Dataflow;
```

BEHAVIORAL:

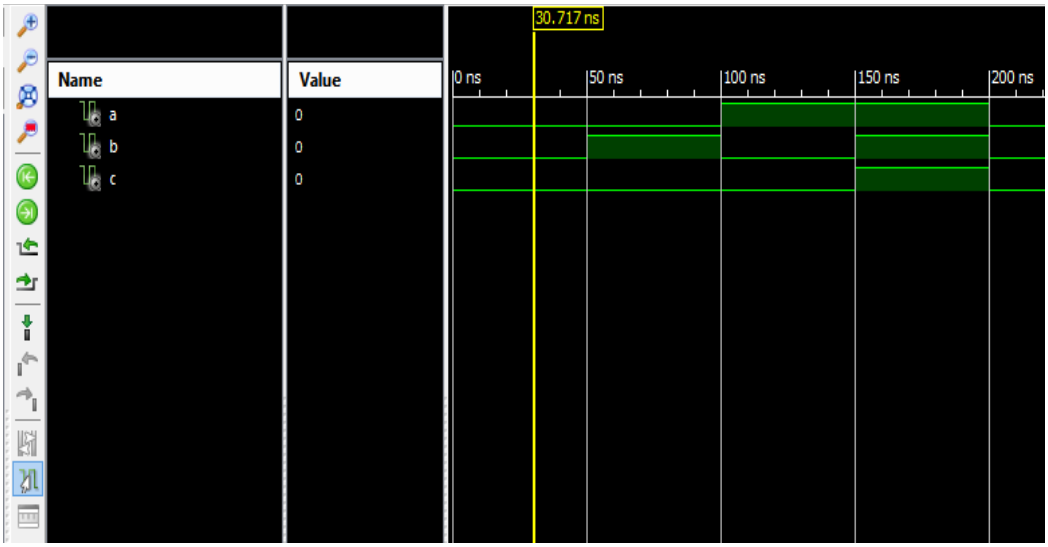
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity AND_LOGIC is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           C : out  STD_LOGIC);
end AND_LOGIC;
architecture Behavioral of AND_LOGIC is
begin
          process (A,B)
          begin
                    if A='1' and B='1' then C<='1';
                    else C<='0';
                    end if;
          end process;
end Behavioral;
```

VHDL TEST BENCH:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY AND_LOGIC_TB IS
END AND_LOGIC_TB;
          ARCHITECTURE behavior OF AND_LOGIC_TB IS
    COMPONENT AND_LOGIC
    PORT(
         A : IN  std_logic;
         B : IN  std_logic;
         C : OUT  std_logic
        );
    END COMPONENT;
   signal A : std_logic := '0';
   signal B : std_logic := '0';
   signal C : std_logic;
BEGIN
    uut: AND_LOGIC PORT MAP (
          A => A,
          B => B,
          C => C
        );
   stim_proc: process
   begin
     A<='0';B<='0';WAIT FOR 50 NS;
                    A<='0';B<='1';WAIT FOR 50 NS;
```

```
            A<='1';B<='0';WAIT FOR 50 NS;
            A<='1';B<='1';WAIT FOR 50 NS;
  end process;
END;
```

OUTPUT WAVEFORMS:



OR GATE:

DATAFLOW:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity OR_LOGIC is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           C : out  STD_LOGIC);
end OR_LOGIC;
architecture Dataflow of OR_LOGIC is
begin
C<= A OR B;
end Dataflow;
```

BEHAVIORAL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity OR_LOGIC is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           C : out  STD_LOGIC);
end OR_LOGIC;
architecture Behavioral of OR_LOGIC is
begin
          process (A,B)
          begin
                    if A='0' and B='0' then C<='0';
                    else C<='1';
                    end if;
```

```
        end process;
end Behavioral;
```

VHDL TEST BENCH:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY OR_LOGIC_TB IS
END OR_LOGIC_TB;
ARCHITECTURE behavior OF OR_LOGIC_TB IS
    COMPONENT OR_LOGIC
    PORT (
         A: IN std_logic;
         B: IN std_logic;
         C: OUT std_logic        );
    END COMPONENT;
   signal A : std_logic := '0';
   signal B : std_logic := '0';
   signal C : std_logic;
BEGIN
   uut: OR_LOGIC PORT MAP (
          A => A,
          B => B,
          C => C        );
   stim_proc: process
   begin
                    A<='0'; B<='0'; WAIT FOR 50 NS;
                    A<='0'; B<='1'; WAIT FOR 50 NS;
                    A<='1'; B<='0'; WAIT FOR 50 NS;
                    A<='1'; B<='1'; WAIT FOR 50 NS;
   end process;
END;
```

OUTPUT WAVEFORMS:

NOT GATE:

DATAFLOW:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity NOT_LOGIC is
   Port ( A : in  STD_LOGIC;
          B : out  STD_LOGIC);
end NOT_LOGIC;
architecture Dataflow of NOT_LOGIC is
begin
B<= NOT A ;
end Dataflow;
```

BEHAVIORAL:

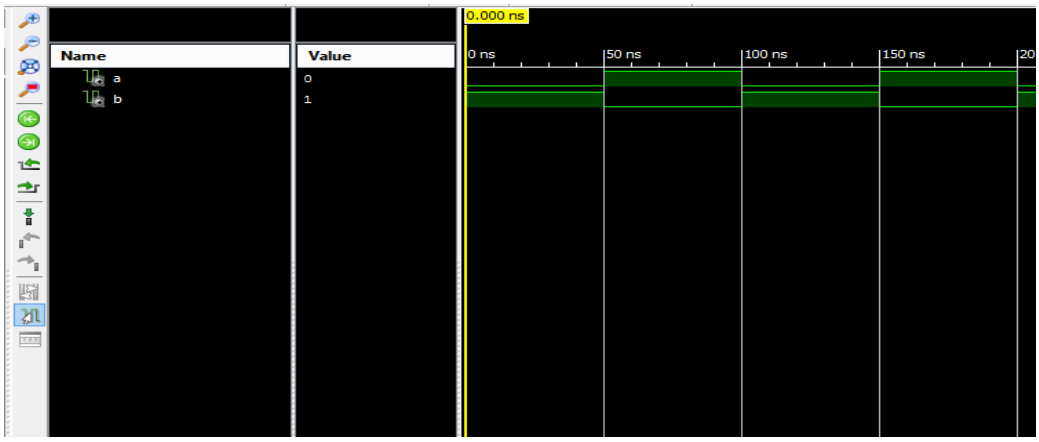
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity NOT_LOGIC is
   Port ( A : in  STD_LOGIC;
          B : out  STD_LOGIC);
end NOT_LOGIC;
architecture Behavioral of NOT_LOGIC is
begin
          process (A)
                    begin
                              if A='1' then B<='0';
                              else B<='1';
                              end if;
          end process;
end Behavioral;
```

VHDL TEST BENCH:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY NOT_LOGIC_TB IS
END NOT_LOGIC_TB;
ARCHITECTURE behavior OF NOT_LOGIC_TB IS
   COMPONENT NOT_LOGIC
   PORT(
        A : IN  std_logic;
        B : OUT  std_logic
       );
   END COMPONENT;
  signal A : std_logic := '0';
  signal B : std_logic;
BEGIN
  uut: NOT_LOGIC PORT MAP (
         A => A,
         B => B
       );
```

```
  stim_proc: process
  begin
                A<='0';WAIT FOR 50 NS;
                A<='1';WAIT FOR 50 NS;
  end process;
END;
```

OUTPUT WAVEFORMS:



NAND GATE:

DATAFLOW:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity NAND_LOGIC is
    Port (A: in STD_LOGIC;
          B: in STD_LOGIC;
          C: out STD_LOGIC);
end NAND_LOGIC;
architecture Dataflow of NAND_LOGIC is
begin
C<= A NAND B;
end Dataflow;
```

BEHAVIORAL:

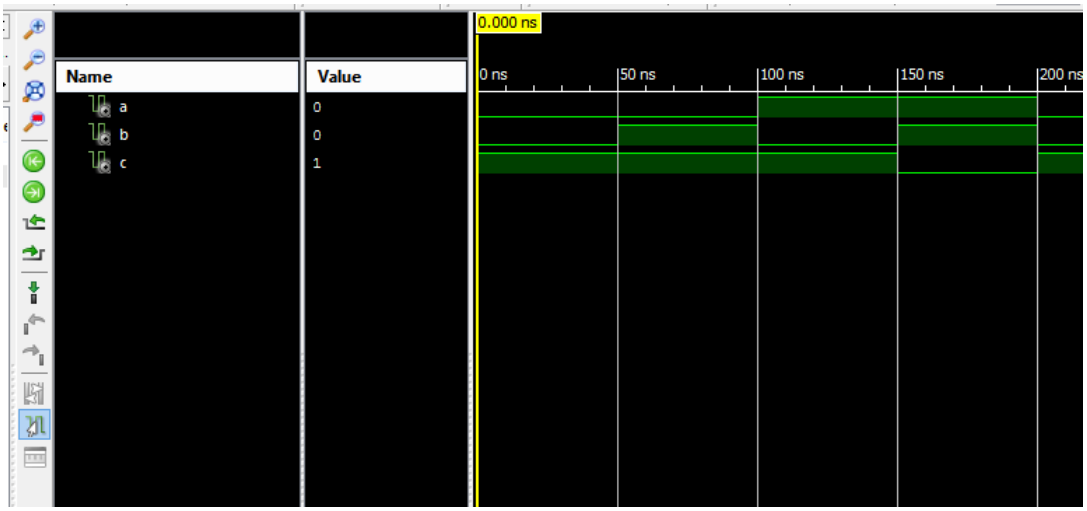
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity NAND_LOGIC is
    Port (A: in STD_LOGIC;
          B: in STD_LOGIC;
          C: out STD_LOGIC);
end NAND_LOGIC;
architecture Behavioral of NAND_LOGIC is
begin
          process (A,B)
                    begin
                              if A='1' AND B='1' then C<='0';
                              else C<='1';
```

```
                    end if;
        end process;
end Behavioral;
```

VHDL TEST BENCH:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY NAND_LOGIC _TB IS
END NAND_LOGIC _TB;
ARCHITECTURE behavior OF NAND_LOGIC _TB IS
   COMPONENT NAND_LOGIC
   PORT(
        A : IN  std_logic;
        B : IN  std_logic;
        C : OUT  std_logic
       );
   END COMPONENT;
  signal A : std_logic := '0';
  signal B : std_logic := '0';
  signal C : std_logic;
BEGIN
  uut: NAND_LOGIC PORT MAP (
         A => A,
         B => B,
         C => C
       );
  stim_proc: process
  begin
                A<='0';B<='0';WAIT FOR 50 NS;
                A<='0';B<='1';WAIT FOR 50 NS;
                A<='1';B<='0';WAIT FOR 50 NS;
                A<='1';B<='1';WAIT FOR 50 NS;
  end process;
END;
```

OUTPUT WAVEFORMS:



NOR GATE:

DATAFLOW:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity NOR_LOGIC is
   Port ( A : in  STD_LOGIC;
          B : in  STD_LOGIC;
          C : out  STD_LOGIC);
End NOR_LOGIC;
architecture Dataflow of NOR_LOGIC is
begin
C<= A NOR B;
end Dataflow;
```

BEHAVIORAL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity NOR_LOGIC is
   Port ( A : in  STD_LOGIC;
                        B : in  STD_LOGIC;
          C : out  STD_LOGIC);
end NOR_LOGIC;
architecture Behavioral of NOR_LOGIC is
begin
          process (A,B)
                    begin
                              if A='0' AND B='0' then C<='1';
                              else C<='0';
                              end if;
          end process;
end Behavioral;
```

VHDL TEST BENCH:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY NOR_LOGIC _TB IS
END NOR_LOGIC _TB;
ARCHITECTURE behavior OF NOR_LOGIC _TB IS
   COMPONENT NOR_LOGIC
   PORT(
        A : IN  std_logic;
        B : IN  std_logic;
        C : OUT  std_logic
       );
   END COMPONENT;
  signal A : std_logic := '0';
  signal B : std_logic := '0';
  signal C : std_logic;
```

```
BEGIN
  uut: NOR_LOGIC PORT MAP (
        A => A,
        B => B,
        C => C
      );
  stim_proc: process
  begin
    A<='0';B<='0';WAIT FOR 50 NS;
                A<='0';B<='1';WAIT FOR 50 NS;
                A<='1';B<='0';WAIT FOR 50 NS;
                A<='1';B<='1';WAIT FOR 50 NS;
  end process;
END;
```

OUTPUT WAVEFORMS:

XOR GATE:

DATAFLOW:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity XOR_LOGIC is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           C : out  STD_LOGIC);
end XOR_LOGIC;
architecture Dataflow of XOR_LOGIC is
begin
C<= A XOR B;
end Dataflow;
```

BEHAVIORAL:

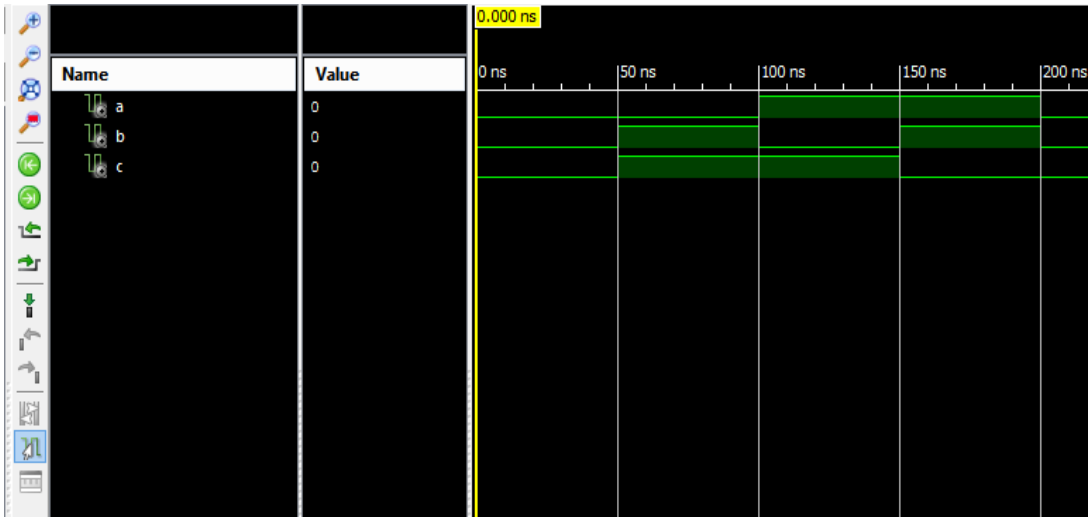
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity XOR_LOGIC is
```

```
    Port ( A : in  STD_LOGIC;
                              B : in  STD_LOGIC;
           C : out  STD_LOGIC);
end XOR_LOGIC;
architecture Behavioral of XOR_LOGIC is
begin
          process (A,B)
                    begin
                              if A=B then C<='0';
                              else C<='1';
                              end if;
          end process;
end Behavioral;
```

VHDL TEST BENCH:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY XOR_LOGIC _TB IS
END XOR_LOGIC _TB;
ARCHITECTURE behavior OF XOR_LOGIC _TB IS
    COMPONENT XOR_LOGIC
    PORT(
         A : IN  std_logic;
         B : IN  std_logic;
         C : OUT  std_logic
        );
    END COMPONENT;
   signal A : std_logic := '0';
   signal B : std_logic := '0';
   signal C : std_logic;
 BEGIN
   uut: XOR_LOGIC PORT MAP (
          A => A,
          B => B,
          C => C
        );
   stim_proc: process
   begin
      A<='0';B<='0';WAIT FOR 50 NS;
                    A<='0';B<='1';WAIT FOR 50 NS;
                    A<='1';B<='0';WAIT FOR 50 NS;
                    A<='1';B<='1';WAIT FOR 50 NS;
   end process;
END;
```

OUTPUT WAVEFORMS:



XNOR GATE:

DATAFLOW:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity XNOR_LOGIC is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           C : out  STD_LOGIC);
end XNOR_LOGIC;
architecture Dataflow of XNOR_LOGIC is
begin
C<= A XNOR B;
end Dataflow;
```

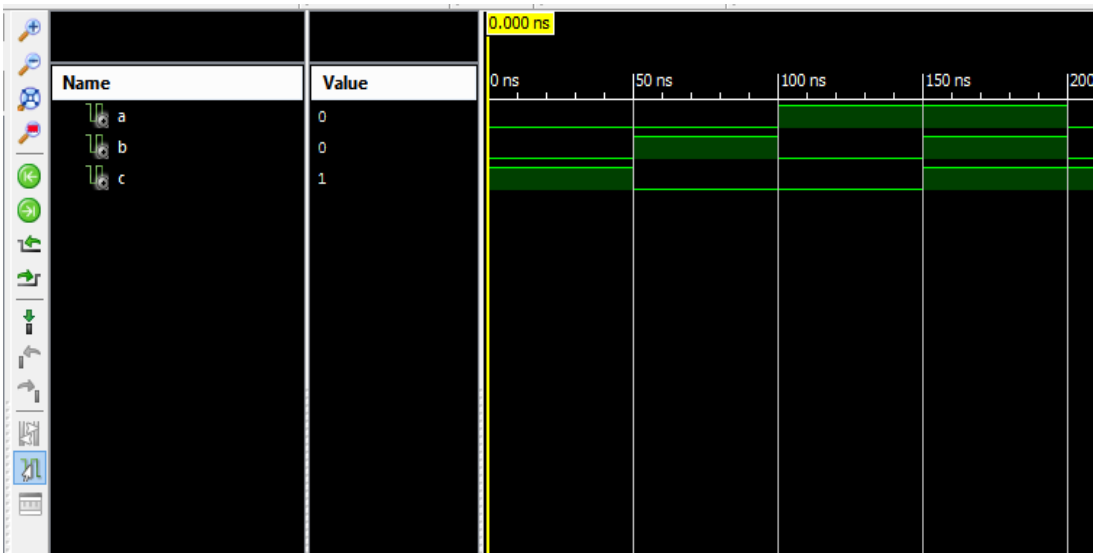
BEHAVIORAL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity XNOR_LOGIC is
    Port ( A : in  STD_LOGIC;
                              B : in  STD_LOGIC;
           C : out  STD_LOGIC);
end XNOR_LOGIC;
architecture Behavioral of XNOR_LOGIC is
begin
          process (A,B)
                    begin
                              if A=B then C<='1';
                              else C<='0';
                              end if;
          end process;
end Behavioral;
```

VHDL TEST BENCH:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY XNOR_LOGIC_TB IS
END XNOR_LOGIC_TB;
ARCHITECTURE behavior OF XNOR_LOGIC_TB IS    COMPONENT XNOR_LOGIC
    PORT(
         A : IN  std_logic;
         B : IN  std_logic;
         C : OUT  std_logic
        );
    END COMPONENT;
   signal A : std_logic := '0';
   signal B : std_logic := '0';
   signal C : std_logic;
BEGIN
   uut: XNOR_LOGIC PORT MAP (
          A => A,
          B => B,
          C => C
        );
   stim_proc: process
   begin
                    A<='0';B<='0';WAIT FOR 50 NS;
                    A<='0';B<='1';WAIT FOR 50 NS;
                    A<='1';B<='0';WAIT FOR 50 NS;
                    A<='1';B<='1';WAIT FOR 50 NS;
   end process;
END;
```

OUTPUT WAVEFORMS:



DEVICE UTILIZATION SUMMARY:

Number of Slices:	1 out of	4656	0%
Number of 4 input LUTs:	1 out of	9312	0%
Number of IOs:	3		
Number of bonded IOBs:	3 out of	232	1%

SYNTHESIS REPORT:

RTL Top Level Output File Name : ALL_LOGIC_GATES.ngr
Top Level Output File Name : ALL_LOGIC_GATES
Output Format : NGC
Optimization Goal : Speed
Keep Hierarchy : No

Design Statistics
IOs : 3

Cell Usage:
BELS : 1
LUT2 : 1
IO Buffers : 3
IBUF : 2
OBUF : 1

RESULT:

Thus the VHDL code for half adder is verified, synthesis report is generated and the design is implemented using FPGA.

VIVA QUESTIONS:

1. Implement the following function using VHDL coding. (Try to minimize if you can). F(A,B,C,D)=(A′+B+C) . (A+B′+D′). (B+C′+D′). (A+B+C+D)
2. What will be the no. of rows in the truth table of N variables?
3. What are the advantages of VHDL?
4. Design Ex-OR gate using behavioral model?
5. Implement the following function using VHDL code f=AB+CD.
6. What are the differences between half adder and full adder?
7. What are the advantages of minimizing the logical expressions?
8. What does a combinational circuit mean?
9. Implement the half adder using VHDL code?
10. Implement the full adder using two half adders and write VHDL program in structural model?

EXPERIMENT – 2
3 x 8 DECODER – IC74138

AIM: To design and simulate 3:8 Decoder - 74138 using VHDL.

SOFTWARE REQIURED:

1. Personal computer
2. ISE Xilinx software

HARDWARE REQUIRED:

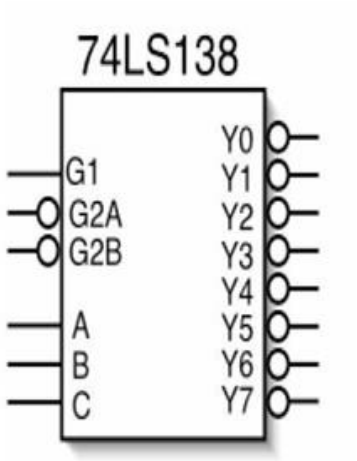
1. SPARTAN – 3E KIT

THEORY:

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

It uses all AND gates, and therefore, the outputs are active- high. For active- low outputs, NAND gates are used. It has 3 input lines and 8 output lines. It is also called as binary to octal decoder it takes a 3-bit binary input code and activates one of the 8(octal) outputs corresponding to that code.

PIN DIAGRAM:



TRUTH TABLE:

Inputs						Outputs							
G1	$\overline{G2A}$	$\overline{G2B}$	C	B	A	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

PROGRAM IN VHDL:

DATAFLOW:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity IC74138_3X8dec_DF is
    Port ( G,GA,GB : in  STD_LOGIC;
           A,B,C : in  STD_LOGIC;
           Y : out  STD_LOGIC_VECTOR (0 to 7));
end IC74138_3X8dec_DF;
architecture dataflow of  IC74138_3X8dec_DF  is
signal X:std_logic_vector (0 to 5);
begin
          X<=G&GA&GB&A&B&C;
                with X select
                      Y<="01111111" when "100000",
                          "10111111" when "100001",
                          "11011111" when "100010",
                          "11101111" when "100011",
                          "11110111" when "100100",
                          "11111011" when "100101",
                          "11111101" when "100110",
                          "11111110" when "100111",
                          "11111111" when others;
end dataflow;
```

BEHAVIORAL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity IC74138_3X8dec_BH is
    Port ( G,GA,GB : in  STD_LOGIC;
           A,B,C : in  STD_LOGIC;
           Y : out  STD_LOGIC_VECTOR (0 to 7));
end IC74138_3X8dec_BH;
architecture Behavioral of  IC74138_3X8dec_BH  is
signal X:std_logic_vector(2 downto 0);
begin
process (G,GA,GB,A,B,C,X)
          begin
                X<=A&B&C;
                      if (G and (not GA) and (not GB))='1' then
                            case X is
                                  when "000"=>Y<="01111111";
                                  when "001"=>Y<="10111111";
                                  when "010"=>Y<="11011111";
                                  when "011"=>Y<="11101111";
                                  when "100"=>Y<="11110111";
```

```
                                    when "101"=>Y<="11111011";
                                    when "110"=>Y<="11111101";
                                        when "111"=>Y<="11111110";
                                    when others=>Y<="11111111";
                                end case;
                            else Y<="11111111";
                            end if;
        end process;
end Behavioral;
```

STRUCTURAL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity IC74138_3X8dec_ST is
    Port ( G,GA,GB : in  STD_LOGIC;
           A,B,C : in  STD_LOGIC;
           Y : out  STD_LOGIC_VECTOR (0 to 7));
end IC74138_3X8dec_ST;
architecture Structural of  IC74138_3X8dec_ST  is
component not11
        port(n1:in std_logic;
                n0:out std_logic);
end component;
component and31
        port(a1,a2,a3:in std_logic;
                a0:out std_logic);
end component;
component nand41
        port(na1,na2,na3,na4:in std_logic;
                na0:out std_logic);
end component;
signal Abar,Bbar,Cbar,GAbar,GBbar,E: std_logic;
begin
L1:not11 port  map(A,Abar);
L2:not11 port  map(B,Bbar);
L3:not11 port  map(C,Cbar);
L4:not11 port  map(GA,GAbar);
L5:not11 port  map(GB,GBbar);
L6:and31 port  map(G,GAbar,GBbar,E);
L7:nand41 port  map(E,Abar,Bbar,Cbar,Y(0));
L8:nand41 port  map(E,Abar,Bbar,C,Y(1));
L9:nand41 port  map(E,Abar,B,Cbar,Y(2));
L10:nand41 port  map(E,Abar,B,C,Y(3));
L11:nand41 port  map(E,A,Bbar,Cbar,Y(4));
L12:nand41 port  map(E,A,Bbar,C,Y(5));
L13:nand41 port  map(E,A,B,Cbar,Y(6));
L14:nand41 port  map(E,A,B,C,Y(7));
end Structural;
```

SUB PROGRAMS:

NOT11:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity not11 is
        port(n1:in std_logic;
             n0:out std_logic);
end not11;
architecture dataflow of  not11 is
begin
        n0<= not  n1;
end dataflow;
```

AND31:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity and31 is
   Port ( a1,a2,a3 : in  STD_LOGIC;
                a0 : out  STD_LOGIC);
end and31;
architecture dataflow of and31 is
begin
        a0<=a1 and a2 and a3;
end dataflow;
```

NAND41:

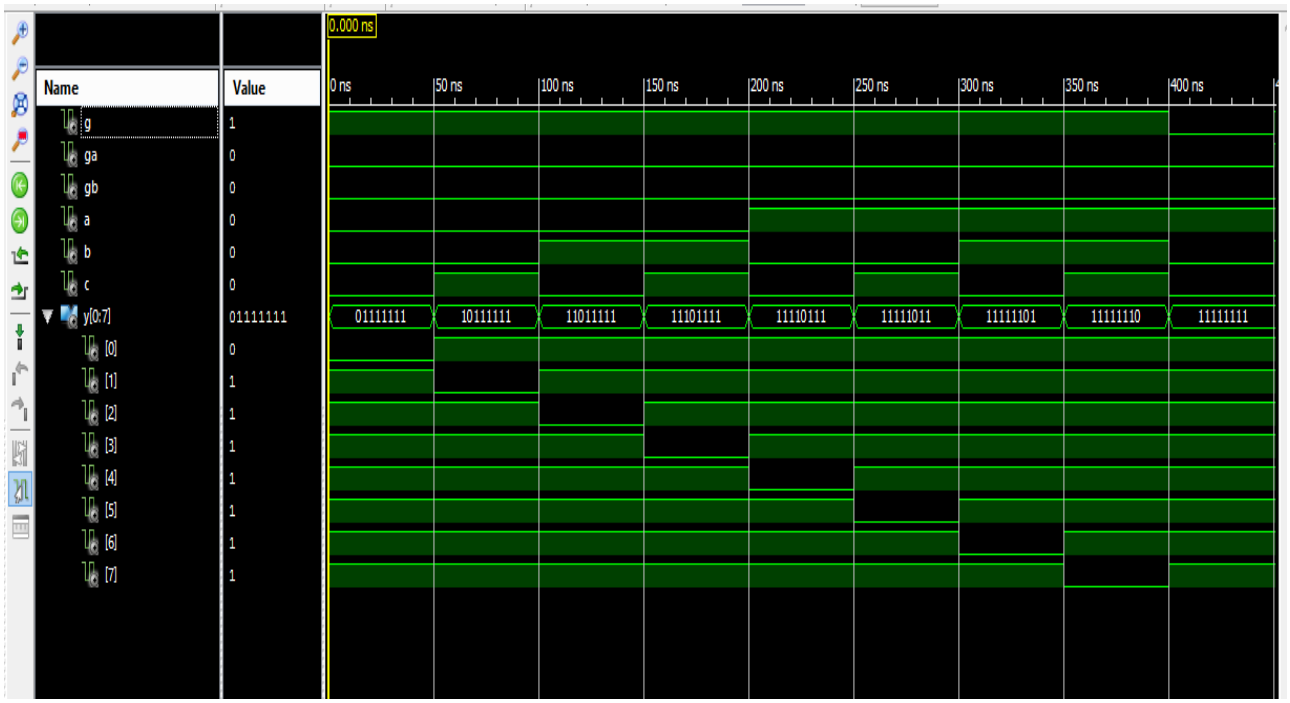
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity nand41 is
        port(na1,na2,na3,na4:in std_logic;
                na0:out std_logic);
end nand41;
architecture dataflow of nand41 is
begin
        na0<= not( na1 and na2 and na3 and na4);
end dataflow;
```

VHDL TEST BENCH:

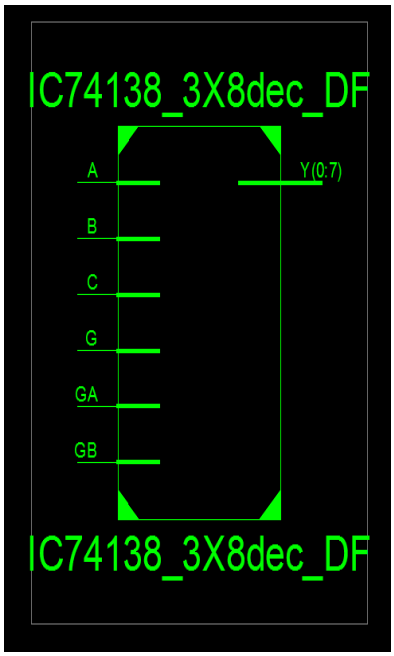
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY IC74138_3X8dec_TB IS
END IC74138_3X8dec_TB;
ARCHITECTURE behavior OF IC74138_3X8dec_TB IS
   COMPONENT IC74138_3X8dec
   PORT(
      G : IN  std_logic;
      GA : IN  std_logic;
```

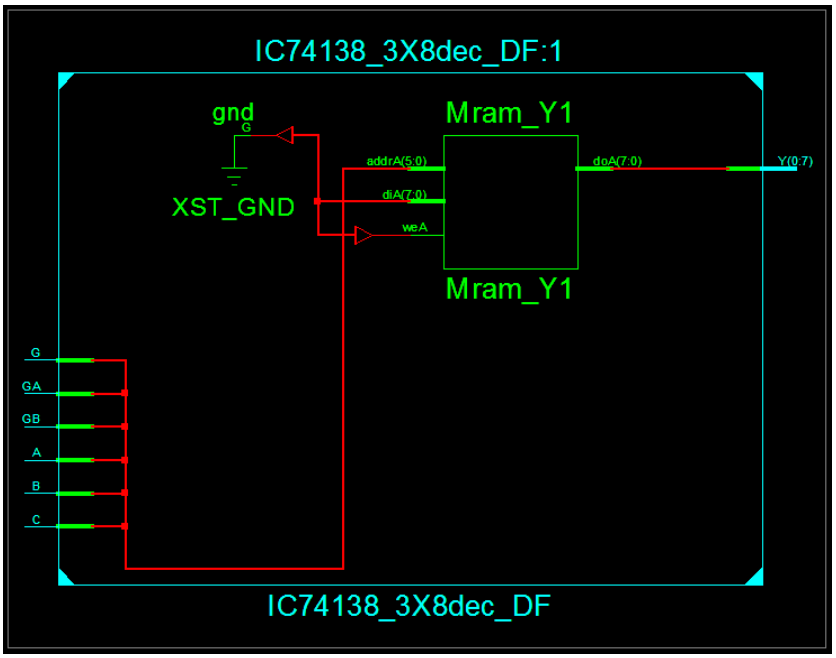
```
        GB : IN  std_logic;
        A : IN  std_logic;
        B : IN  std_logic;
        C : IN  std_logic;
        Y : OUT  std_logic_vector(0 to 7)
       );
    END COMPONENT;
   signal G : std_logic := '0';
   signal GA : std_logic := '0';
   signal GB : std_logic := '0';
   signal A : std_logic := '0';
   signal B : std_logic := '0';
   signal C : std_logic := '0';
   signal Y : std_logic_vector(0 to 7);
  BEGIN
   uut: IC74138_3X8dec  PORT MAP (
         G => G,
         GA => GA,
         GB => GB,
         A => A,
         B => B,
         C => C,
         Y => Y
        );
                    stim_proc: process
                   begin
                           G<='1';GA<='0';GB<='0';A<='0';B<='0';C<='0'; wait for 50 ns;
                           G<='1';GA<='0';GB<='0';A<='0';B<='0';C<='1'; wait for 50 ns;
                           G<='1';GA<='0';GB<='0';A<='0';B<='1';C<='0'; wait for 50 ns;
                           G<='1';GA<='0';GB<='0';A<='0';B<='1';C<='1'; wait for 50 ns;
                           G<='1';GA<='0';GB<='0';A<='1';B<='0';C<='0'; wait for 50 ns;
                           G<='1';GA<='0';GB<='0';A<='1';B<='0';C<='1'; wait for 50 ns;
                           G<='1';GA<='0';GB<='0';A<='1';B<='1';C<='0'; wait for 50 ns;
                           G<='1';GA<='0';GB<='0';A<='1';B<='1';C<='1'; wait for 50 ns;
                           G<='0';GA<='0';GB<='0';A<='1';B<='0';C<='0'; wait for 50 ns;
                           G<='1';GA<='1';GB<='0';A<='1';B<='1';C<='0'; wait for 50 ns;
                           G<='1';GA<='0';GB<='1';A<='1';B<='0';C<='1'; wait for 50 ns;
   end process;
END;
```

OUTPUT WAVEFORMS:

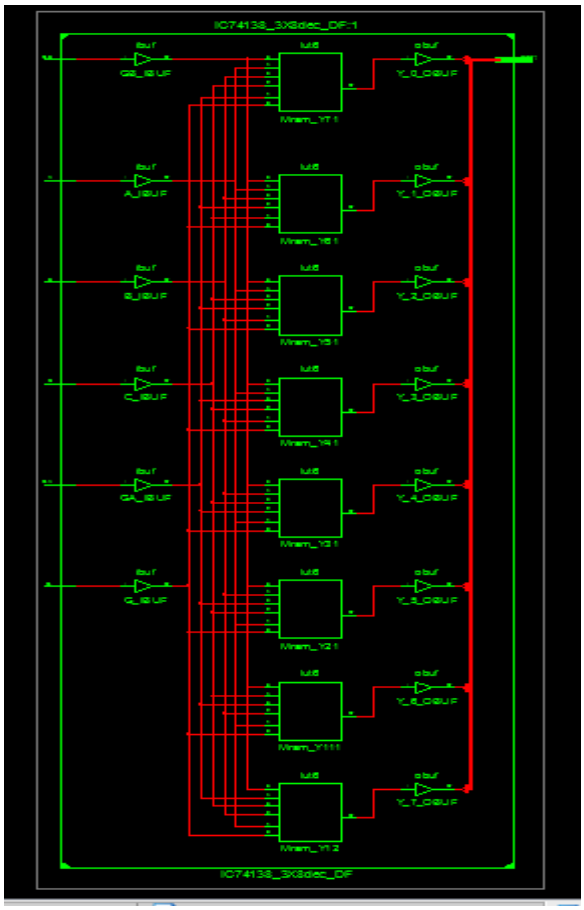


RTL SCHEMATIC:





INTERNAL DIAGRAM OF 3X8 DECODER:



DEVICE UTILIZATION SUMMARY:

Number of Slices:	5 out of	4656	0%
Number of 4 input LUTs:	9 out of	9312	0%
Number of IOs:	14		
Number of bonded IOBs:	14 out of	232	6%

SYNTHESIS REPORT:

RTL Top Level Output File Name : IC74138_3X8dec.ngr
Top Level Output File Name : IC74138_3X8dec
Output Format : NGC
Optimization Goal : Speed
Keep Hierarchy : No

Design Statistics
IOs : 14

Cell Usage:
BELS : 9
LUT3 : 1
LUT4 : 8
IO Buffers : 14
IBUF : 6
OBUF : 8

RESULT:

Thus the VHDL code for 3x8 Decoder 74138 is verified, synthesis report is generated and the design is implemented using FPGA.

VIVA QUESTIONS:

1. Write the behavioural code for the IC 74x138.
2. Write the VHDL code for the IC 74x138 using CASE statement.
3. Write the VHDL code for the IC 74x138 using WITH statement.
4. Write the VHDL code for the IC 74x138 using WHEN--ELSE statement.
5. Write the structural program for IC 74x138.
6. What does priority encoder mean?
7. How many decoders are needed to construct 4X16 decoder?
8. What is the difference between decoder and encoder?
9. Write the syntax for exit statement?
10. Explain briefly about next statement?
11. How to specify the delay in VHDL program?
12. Write the syntax for component declaration.

EXPERIMENT – 3
8 X 1 MULTIPLEXER-74151 AND
1 X 4 DEMULTIPLEXER-74155

AIM: To design and simulate MUX & DEMUX using VHDL.

SOFTWARE REQIURED:

1. Personal computer
2. ISE Xilinx software

HARDWARE REQUIRED:

1. SPARTAN – 3E KIT

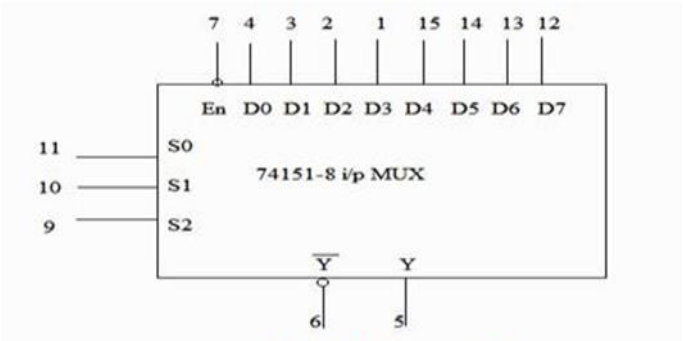
THEORY:

Multiplexing is defined as the process of feeding several independent signals to a common load, one at a time. The device or switching circuitry used to select and connect one of these several signals to the load at any one time is known as a multiplexer.

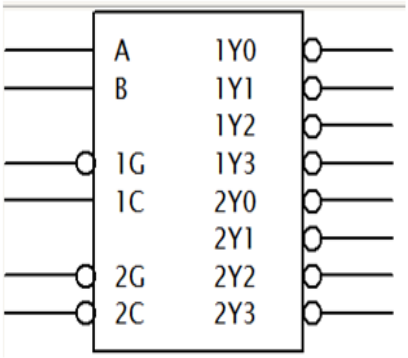
The reverse function of multiplexing, known as de-multiplexing, pertains to the process of feeding several independent loads with signals coming from a common signal source, one at a time. A device used for de-multiplexing is known as de- multiplexer.

Multiplexing and de-multiplexing, therefore, allow the efficient use of common circuits to feed a common load with signals from several signal sources, and to feed several loads form a single, common signal source, respectively.

PIN DIAGRAM:



8 X 1 MULTIPLEXER



2x4 Dual - DE MULTIPLEXER

TRUTH TABLE:

S.No	en_1	Data select lines A	B	C	Output Y
1	0	0	0	0	I(0)
2	0	0	0	1	I(1)
3	0	0	1	0	I(2)
4	0	0	1	1	I(3)
5	0	1	0	0	I(4)
6	0	1	0	1	I(5)
7	0	1	1	0	I(6)
8	0	1	1	1	I(7)
9	1	X	X	X	0

8 X 1 MULTIPLEXER

B	A	1G	1C	1Y0	1Y1	1Y2	1Y3	B	A	2G	2C	2Y0	2Y1	2Y2	2Y3
x	x	1	x	1	1	1	1	x	x	1	x	1	1	1	1
x	x	x	0	1	1	1	1	x	x	x	1	1	1	1	1
0	0	0	1	0	1	1	1	0	0	0	0	0	1	1	1
0	1	0	1	1	0	1	1	0	1	0	0	1	0	1	1
1	0	0	1	1	1	0	1	1	0	0	0	1	1	0	1
1	1	0	1	1	1	1	0	1	1	0	0	1	1	1	0

2x4 Dual - DE MULTIPLEXER

PROGRAM IN VHDL:

8 X 1 MULTIPLEXER:

DATAFLOW:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity IC74151_8X1MUX_DF is
    Port ( S : in  STD_LOGIC_VECTOR (2 downto 0);
           D : in  STD_LOGIC_VECTOR (0 to 7);
           Y : inout  STD_LOGIC;
           YBar : out  STD_LOGIC;
           ENBar : in  STD_LOGIC);
end IC74151_8X1MUX_DF;
architecture Dataflow of IC74151_8X1MUX_DF is
```

```
signal X:std_logic;
begin
        with S select
                X<=D(0) when "000",
                        D(1) when "001",
                        D(2) when "010",
                        D(3) when "011",
                        D(4) when "100",
                        D(5) when "101",
                        D(6) when "110",
                        D(7) when others;
        Y<=X when ENBar='0' else '0';
        YBar<= not Y;
end Dataflow;
```

BEHAVIORAL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity IC74151_8X1MUX_BF is
   Port ( S : in  STD_LOGIC_VECTOR (2 downto 0);
        D : in  STD_LOGIC_VECTOR (0 to 7);
        Y : inout  STD_LOGIC;
        YBar : out  STD_LOGIC;
        ENBar : in  STD_LOGIC);
end IC74151_8X1MUX_BF;
architecture Behavioral of  IC74151_8X1MUX_BF is
begin
PROCESS (S,D,ENBar,Y)
BEGIN
        IF ENBar='0' THEN
                CASE S IS
                        WHEN "000"=>Y<=D(0);
                        WHEN "001"=>Y<=D(1);
                        WHEN "010"=>Y<=D(2);
                        WHEN "011"=>Y<=D(3);
                        WHEN "100"=>Y<=D(4);
                        WHEN "101"=>Y<=D(5);
                        WHEN "110"=>Y<=D(6);
                        WHEN OTHERS=>Y<=D(7);
                END CASE;
        ELSE Y<='0';
        END IF;
        IF Y='0' THEN YBar<='1';
        ELSE YBar<='0';
        END IF;
END PROCESS;
end Behavioral;
```

STRUCTURAL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity IC74151_8X1MUX_ST is
   Port ( S : in  STD_LOGIC_VECTOR (2 downto 0);
```

```
        D : in  STD_LOGIC_VECTOR (0 to 7);
        ENBar : in  STD_LOGIC
        Y : inout  STD_LOGIC;
        YBar : out  STD_LOGIC);
end IC74151_8X1MUX_ST;
architecture Structural of IC74151_8X1MUX_ST is
component  not11
port ( n1: in std_logic;
         n0: out std_logic);
end component;
component  and51
port ( a1,a2,a3,a4,a5: in std_logic;
                  a0: out std_logic);
end component;
component  or81
port ( r1,r2,r3,r4,r5,r6,r7,r8: in std_logic;
                  r0: out std_logic);
end component;
signal S2,S1,S0,EN:std_logic;
signal R:std_logic_vector(0 to 7);
begin
L1:not11 port  map (ENBar,EN);
L2:not11 port  map (S(2),S2);
L3:not11 port  map (S(1),S1);
L4: not11 port  map (S(0),S0);
L5:and51 port  map (EN,D(0),S2,S1,S0,R(0));
L6:and51 port  map (EN,D(1),S2,S1,S(0),R(1));
L7:and51 port  map (EN,D(2),S2,S(1),S0,R(2));
L8:and51 port  map (EN,D(3),S2,S(1),S(0),R(3));
L9:and51 port  map (EN,D(4),S(2),S1,S0,R(4));
L10:and51 port  map (EN,D(5),S(2),S1,S(0),R(5));
L11:and51 port  map (EN,D(6),S(2),S(1),S0,R(6));
L12:and51 port  map (EN,D(7),S(2),S(1),S(0),R(7));
L13:or81 port  map (R(0),R(1),R(2),R(3),R(4),R(5),R(6),R(7),Y);
L14:not11 port  map (Y,YBar);
end Structural;
```

SUB PROGRAMS:

NOT11:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity not11 is
port ( n1: in std_logic;
         n0: out std_logic);
end not11;
architecture dataflow of not11 is
begin
        n0<= not n1;
end dataflow;
```

AND51:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity and51 is
```

```
port ( a1,a2,a3,a4,a5: in std_logic;
                a0: out std_logic);
end and51;
architecture dataflow of and51 is
begin
        a0<= a1 and a2 and a3 and a4 and a5;
end dataflow;
```

OR81:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity or81 is
port ( r1,r2,r3,r4,r5,r6,r7,r8: in std_logic;
                r0: out std_logic);
end or81;
architecture dataflow of or81 is
begin
        r0<=(r1 or r2 or r3 or r4 or r5 or r6 or r7 or r8);
end dataflow;
```

VHDL TEST BENCH:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY IC74151_8X1MUX_TB IS
END IC74151_8X1MUX_TB;
ARCHITECTURE behavior OF IC74151_8X1MUX_TB IS
   COMPONENT IC74151_8X1MUX
   PORT(
      S : IN  std_logic_vector(2 downto 0);
      D : IN  std_logic_vector(0 to 7);
      Y : INOUT  std_logic;
      YBar : OUT  std_logic;
      ENBar : IN  std_logic
     );
   END COMPONENT;
  signal S : std_logic_vector(2 downto 0) := (others => '0');
  signal D : std_logic_vector(0 to 7) := (others => '0');
  signal ENBar : std_logic := '0';
  signal Y : std_logic;
  signal YBar : std_logic;
BEGIN
  uut: IC74151_8X1MUX  PORT MAP (
       S => S,
       D => D,
       Y => Y,
       YBar => YBar,
       ENBar => ENBar
      );
  stim_proc: process
  begin
        ENBar<='0';D<="00011011";S<="000"; WAIT FOR 50 NS;
        ENBar<='0';D<="00011011";S<="001"; WAIT FOR 50 NS;
        ENBar<='0';D<="01011011";S<="010"; WAIT FOR 50 NS;
        ENBar<='0';D<="00011011";S<="011"; WAIT FOR 50 NS;
        ENBar<='0';D<="11011011";S<="100"; WAIT FOR 50 NS;
```

```
        ENBar<='0';D<="00011001";S<="101"; WAIT FOR 50 NS;
        ENBar<='0';D<="01111010";S<="110"; WAIT FOR 50 NS;
        ENBar<='0';D<="01111010";S<="111"; WAIT FOR 50 NS;
        ENBar<='1';D<="01111010";S<="110"; WAIT FOR 50 NS;
        ENBar<='1';D<="00111011";S<="010"; WAIT FOR 50 NS;
        ENBar<='1';D<="00011010";S<="100"; WAIT FOR 50 NS;
  end process;
END;
```

1x4 DE MULTIPLEXER:

DATAFLOW:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity IC74155_1X4DeMUX_DF is
        port (D,EN : in std_logic;
                S    : in std_logic_vector(1 downto 0);
                Y    : out std_logic_vector(0 to 3));
End  IC74155_1X4DeMUX_DF;
architecture Dataflow of  IC74155_1X4DeMUX_DF is
begin
Y(0)<= D and (not EN) and (not S(1)) and (not S(0));
Y(1)<= D and (not EN) and (not S(1)) and S(0);
Y(2)<= D and (not EN) and S(1) and (not S(0));
Y(3)<= D and (not EN) and S(1) and S(0);
end Dataflow;
```

BEHAVIORAL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity IC74155_1X4DeMUX_BF is
        port (D,EN : in std_logic;
                S    : in std_logic_vector(1 downto 0);
                Y    : out std_logic_vector(0 to 3));
end IC74155_1X4DeMUX_BF;
architecture Behavioral of IC74155_1X4DeMUX_BF is
begin
        process(D,EN,S)
begin
        Y(0)<= D and (not EN) and (not S(1)) and (not S(0));
        Y(1)<= D and (not EN) and (not S(1)) and S(0);
        Y(2)<= D and (not EN) and S(1) and (not S(0));
        Y(3)<= D and (not EN) and S(1) and S(0);
        end process;
end Behavioral;
```

STRUCTURAL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity IC74155_1X4DeMUX_ST is
        port (D,EN : in std_logic;
```

```
                S   : in std_logic_vector(1 downto 0);
                Y  : out std_logic_vector(0 to 3));
end IC74155_1X4DeMUX_ST;
architecture Structural of IC74155_1X4DeMUX_ST is
component NOT11
        port (n1 : in std_logic;
              n0 : out std_logic);
end component;
component AND41
        port (a1,a2,a3,a4 : in std_logic;
                    a0 : out std_logic);
end component;
signal s1,s0,E: std_logic;
begin
L1:NOT11 port  map(EN,E);
L2:NOT11 port  map(S(1),s1);
L3:NOT11 port  map(S(0),s0);
L4:AND41 port  map(E,D,s1,s0,Y(0));
L5:AND41 port  map(E,D,s1,S(0),Y(1));
L6:AND41 port  map(E,D,S(1),s0,Y(2));
L7:AND41 port  map(E,D,S(1),S(0),Y(3));
end Structural;
```

SUB PROGRAMS:

NOT11:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity NOT11 is
   Port ( n1 : in  STD_LOGIC;
          n0 : out  STD_LOGIC);
end NOT11;
architecture Dataflow of NOT11 is
begin
n0<=not n1;
end Dataflow;
```

AND41:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity AND41 is
        port (a1,a2,a3,a4 : in std_logic;
                    a0 : out std_logic);
end AND41;
architecture Dataflow of AND41 is
begin
a0<= a1 and a2 and a3 and a4;
end Dataflow;
```

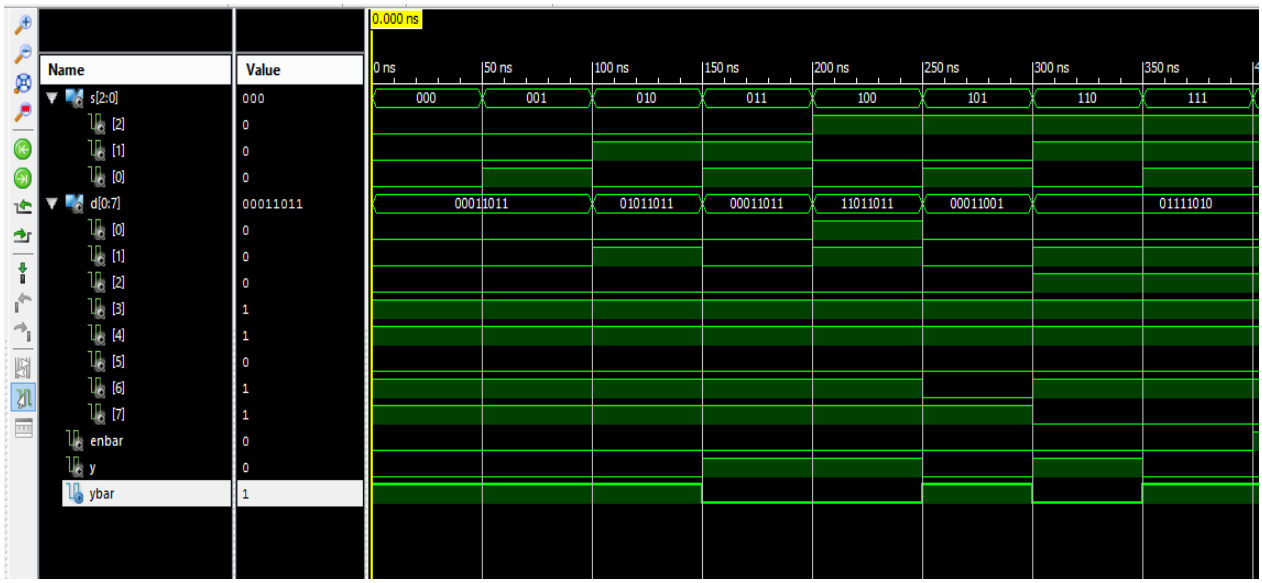
VHDL TEST BENCH:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY IC74155_1X4DeMUX_TB IS
END IC74155_1X4DeMUX_TB;
ARCHITECTURE behavior OF IC74155_1X4DeMUX_TB IS
```

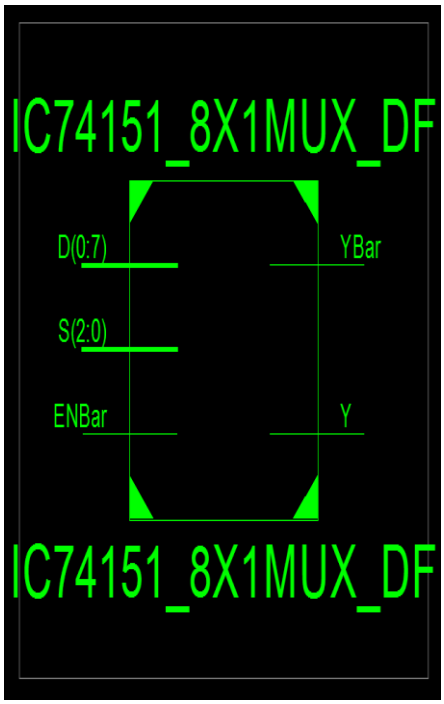
```
COMPONENT IC74155_1X4DeMUX_DF
    PORT(
         D : IN  std_logic;
         EN : IN  std_logic;
         S : IN  std_logic_vector(1 downto 0);
         Y : OUT  std_logic_vector(0 to 3) );
    END COMPONENT;
   signal D : std_logic := '0';
   signal EN : std_logic := '0';
   signal S : std_logic_vector(1 downto 0) := (others => '0');
   signal Y : std_logic_vector(0 to 3);
 BEGIN
    uut: IC74155_1X4DeMUX_DF PORT MAP (
         D => D,
         EN => EN,
         S => S,
         Y => Y );
   stim_proc: process
   begin
          EN<='0';D<='0';S<="00"; wait for 50 ns;
          EN<='0';D<='1';S<="00"; wait for 50 ns;
          EN<='0';D<='0';S<="01"; wait for 50 ns;
          EN<='0';D<='1';S<="01"; wait for 50 ns;
          EN<='0';D<='0';S<="10"; wait for 50 ns;
          EN<='0';D<='1';S<="10"; wait for 50 ns;
          EN<='0';D<='0';S<="11"; wait for 50 ns;
          EN<='0';D<='1';S<="11"; wait for 50 ns;
          EN<='1';D<='0';S<="00"; wait for 50 ns;
          EN<='1';D<='1';S<="01"; wait for 50 ns;
          EN<='1';D<='0';S<="10"; wait for 50 ns;
          EN<='1';D<='1';S<="11"; wait for 50 ns;
   end process;
END;
```

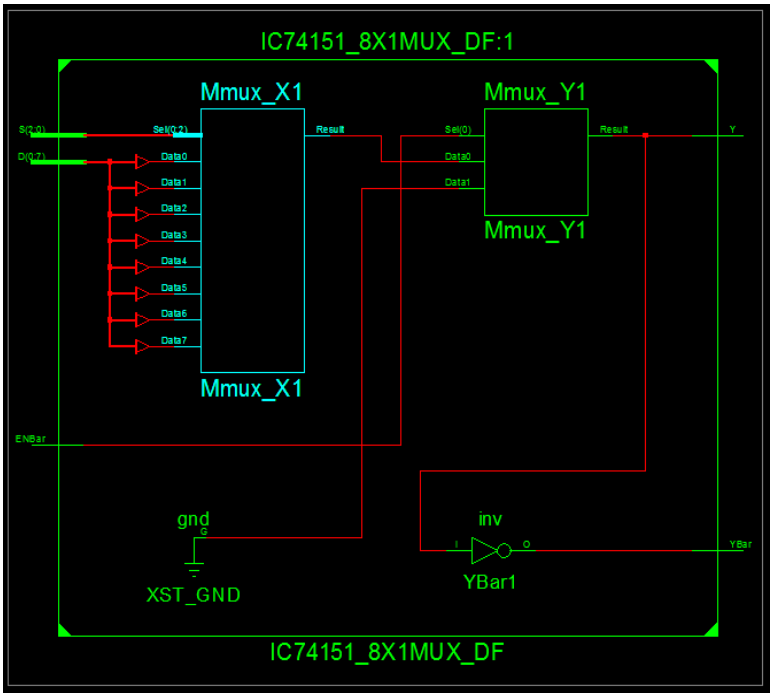
8 X 1 MULTIPLEXER:

OUTPUT WAVEFORMS:

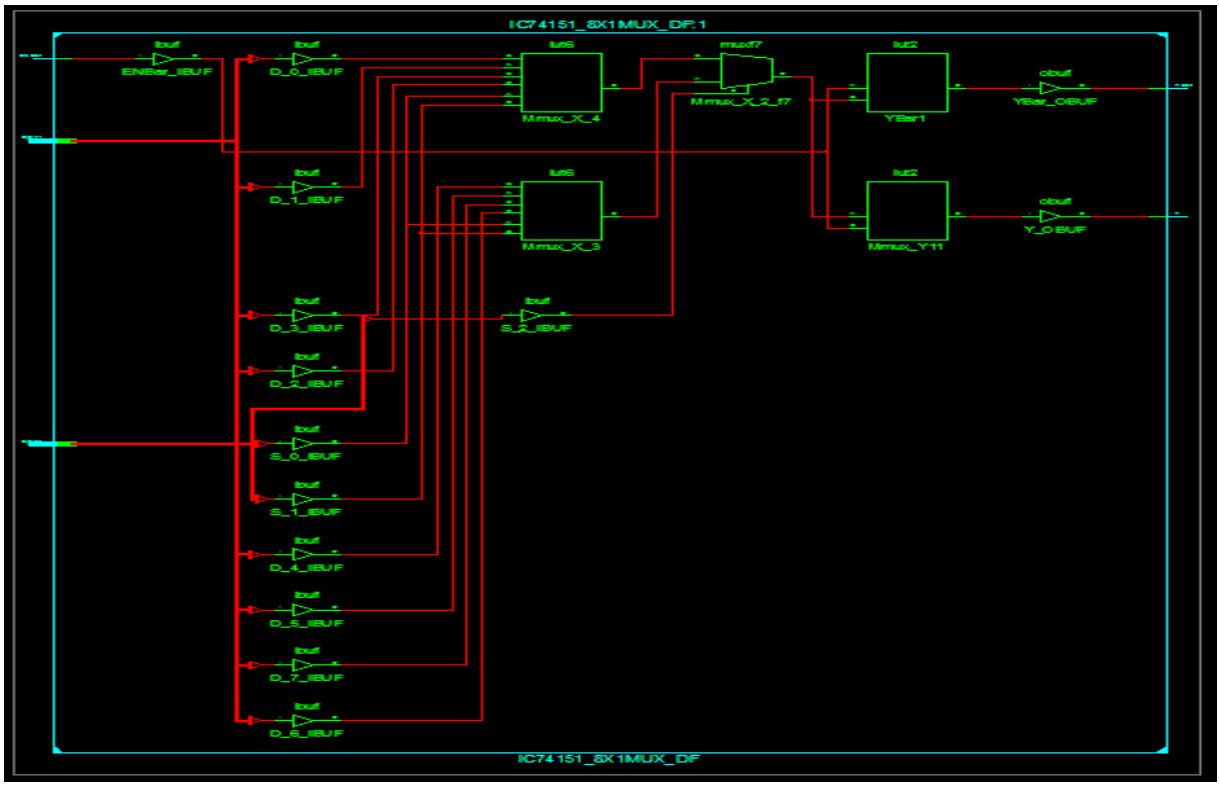


RTL SCHEMATIC:



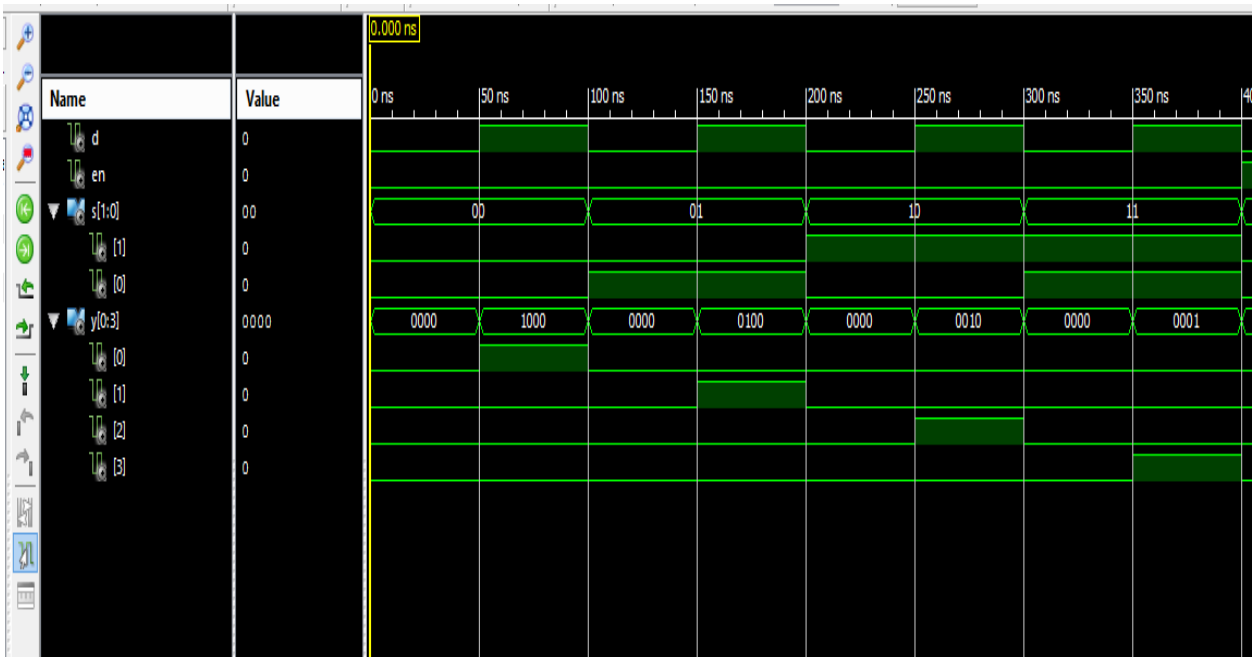


INTERNAL DIAGRAM OF 8X1 MUX:

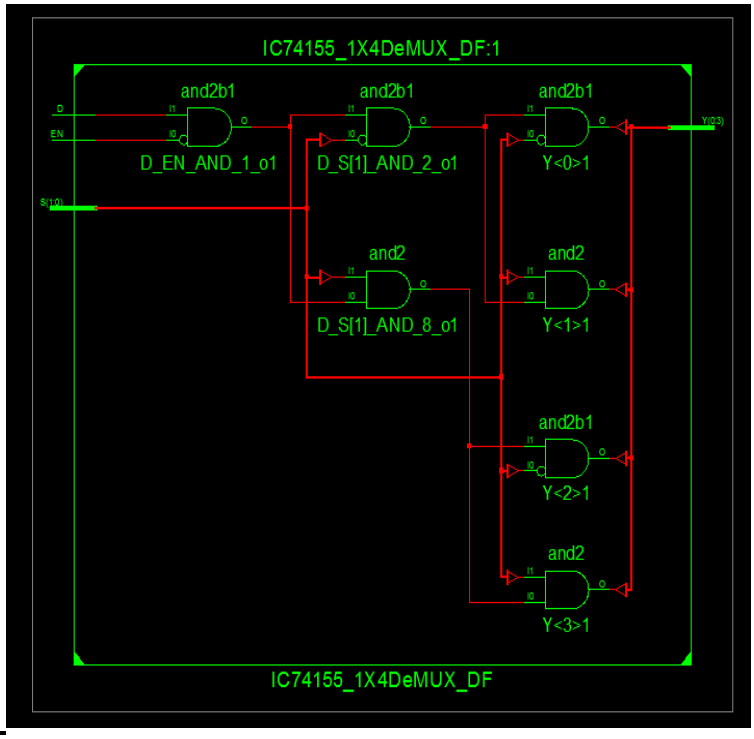


1x4 DE MULTIPLEXER:

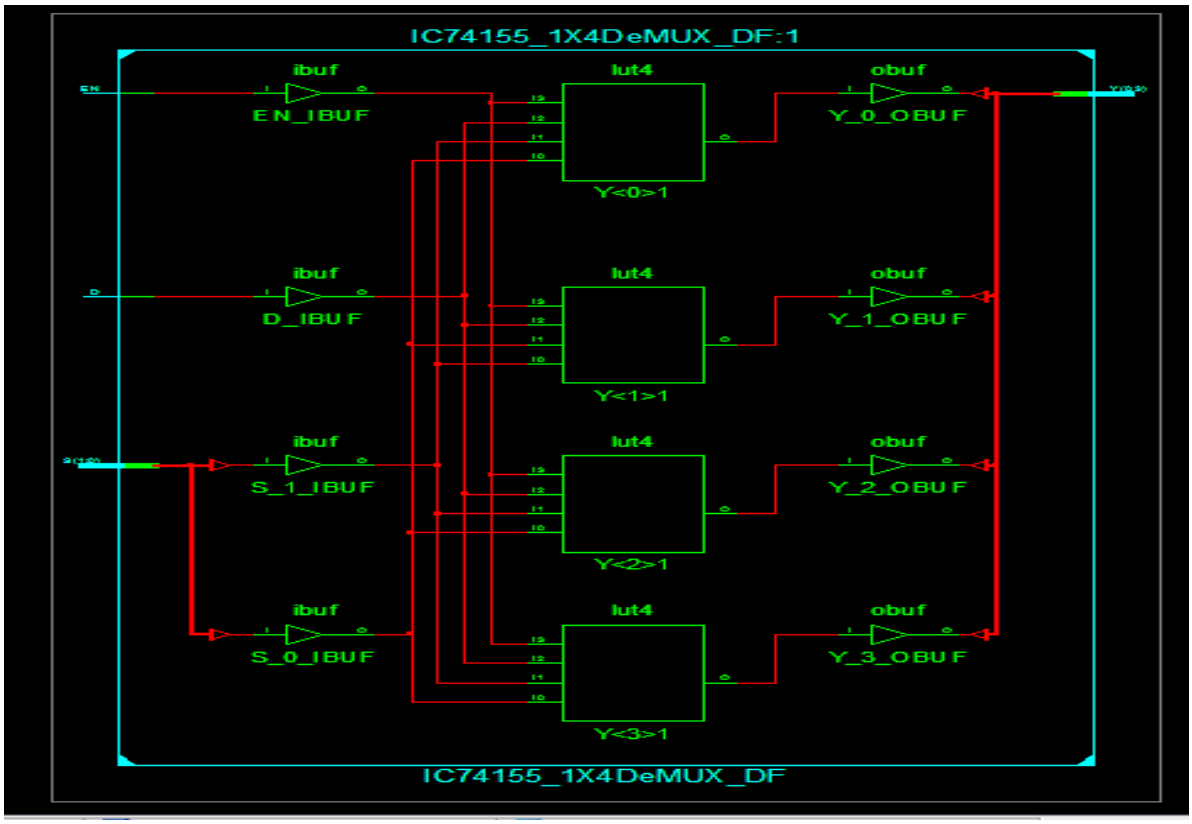
OUTPUT WAVEFORMS:



RTL SCHEMATIC:



INTERNAL DIAGRAM OF 1X4 DeMUX:



8 X 1 MULTIPLEXER:

DEVICE UTILIZATION SUMMARY:

Number of Slices:	4 out of	4656	0%
Number of 4 input LUTs:	7 out of	9312	0%
Number of IOs:	14		
Number of bonded IOBs:	14 out of	232	6%

SYNTHESIS REPORT:

RTL Top Level Output File Name : IC74151_8X1MUX.ngr

Top Level Output File Name : IC74151_8X1MUX

Output Format : NGC

Optimization Goal : Speed

Keep Hierarchy : No

Design Statistics

IOs : 14

Cell Usage:

# BELS	: 8
# INV	: 1
# LUT3	: 4
# LUT4	: 2
# MUXF5	: 1
# IO Buffers	: 14
# IBUF	: 12
# OBUF	: 2

1x4 DE MULTIPLEXER:

DEVICE UTILIZATION SUMMARY:

Number of Slices:	2 out of	4656	0%
Number of 4 input LUTs:	4 out of	9312	0%
Number of IOs:	8		
Number of bonded IOBs:	8 out of	232	3%

SYNTHESIS REPORT:

RTL Top Level Output File Name	: IC74155_1X4DeMUX.ngr
Top Level Output File Name	: IC74155_1X4DeMUX
Output Format	: NGC
Optimization Goal	: Speed
Keep Hierarchy	: No
Design Statistics	
# IOs	: 8
Cell Usage :	
# BELS	: 4
# LUT4	: 4
# IO Buffers	: 8
# IBUF	: 4
# OBUF	: 4

RESULT:

Thus the VHDL code for 8 X 1 MULTIPLEXER-74151 AND 2 X 4 DEMULTIPLEXER-74155 is verified, synthesis report is generated and the design is implemented using FPGA

VIVA QUESTIONS:

1. Write the behavioural code for the IC 74x151.
2. Write the VHDL code for the IC 74x151 using IF statement.
3. Write the VHDL code for the IC 74x151 using WITH statement.
4. Write the VHDL code for the IC 74x151 using WHEN--ELSE statement.
5. Write the structural program for IC 74x151.
6. What is meant by multiplexer?
7. What does demultiplexer mean?
8. How many 8X1 multiplexers are needed to construct 16X1 multiplexer?
9. Compare decoder with demultiplexer?
10. Design a full adder using 8X1 multiplexer?
11. What are the two kinds of subprograms?
12. What are the difference between function and procedure?
13. Explain briefly about subprogram overloading?

EXPERIMENT – 4
4-BIT COMPARATOR-7485

AIM: To design and simulate 4-BIT COMPARATOR using VHDL.

SOFTWARE REQIURED:

1. Personal computer
2. ISE Xilinx software

HARDWARE REQUIRED:

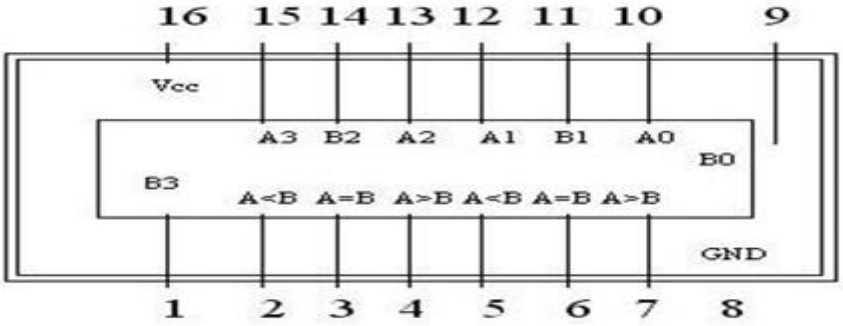
1. SPARTAN – 3E KIT

THEORY:

A comparator is a special combinational circuit designed primarily to compare the relative magnitudes of two binary numbers. If a comparator receives two n-bit numbers A and B as inputs and the outputs are A>B, A=B, A<B. Depending upon the relative magnitudes of the two numbers, one of the outputs will be high.

IC7485 is a bit comparator. It can be used to compare two 4-bit binary words. These ICs, can cascade to compare words of almost any length.

PIN DIAGRAM:



7485
4- Bit Magnitude Comparator

TRUTH TABLE:

Input Numbers				Cascading Inputs			Outputs		
A_3B_3	A_2B_2	A_1B_1	A_0B_0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$Y_{A>B}$	$Y_{A<B}$	$Y_{A=B}$
$A_3>B_3$	X	X	X	X	X	X	H	L	L
$A_3<B_3$	X	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2>B_2$	X	X	X	X	X	H	L	L
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	L	L	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	H	L	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	X	H	L	L	H
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	L	L	H	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	H	L	L	L	L

S.No.	Cascade inputs	Present input condition			AGTBOUT	AEQBOUT	ALTBOUT
		A>B	A=B	A<B			
1	AGTBIN=1	X	X	X	1	0	0
2	AEQBIN=1	1	0	0	1	0	0
		0	1	0	0	1	0
		0	0	1	0	0	1
5	ALTBIN=1	X	X	X	0	0	1

PROGRAM IN VHDL:

DATAFLOW:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity IC7485_Comparator_DF is
    Port ( A : in  STD_LOGIC_VECTOR (0 to 3);
           B : in  STD_LOGIC_VECTOR (0 to 3);
           AgtB : out  STD_LOGIC;
           AeqB : out  STD_LOGIC;
           AltB : out  STD_LOGIC);
end IC7485_Comparator_DF;
architecture Dataflow of IC7485_Comparator_DF is
begin
           AgtB<='1' when A>B else '0';
           AeqB<='1' when A=B else '0';
           AltB<='1' when A<B else '0';
end Dataflow;
```

BEHAVIORAL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity IC7485_Comparator_BH is
    Port ( A : in  STD_LOGIC_VECTOR (0 to 3);
           B : in  STD_LOGIC_VECTOR (0 to 3);
           AgtB : out  STD_LOGIC;
           AeqB : out  STD_LOGIC;
           AltB : out  STD_LOGIC);
end IC7485_Comparator_BH;
architecture Behavioral of IC7485_Comparator_BH is
begin
           process (A,B)
           begin
                    if A>B THEN AgtB<='1';
                    else AgtB<='0';
                    end if;
                    if A=B THEN AeqB<='1';
```

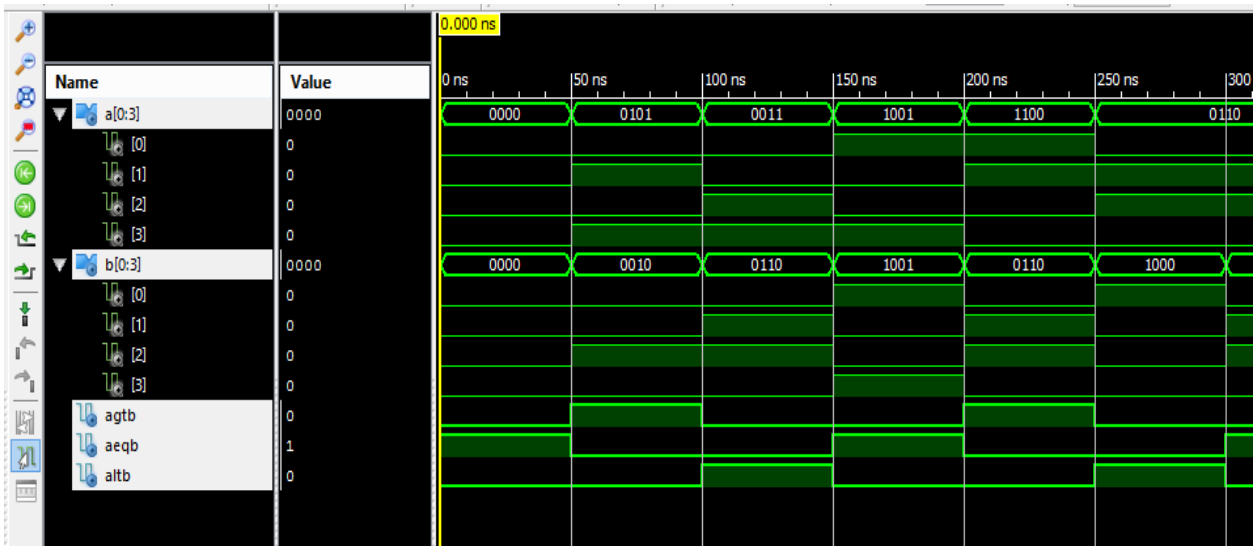
```
                else AeqB<='0';
                end if;
                if A<B THEN AltB<='1';
                else AltB<='0';
                end if;
        end process;
end Behavioral;
```

VHDL TEST BENCH:

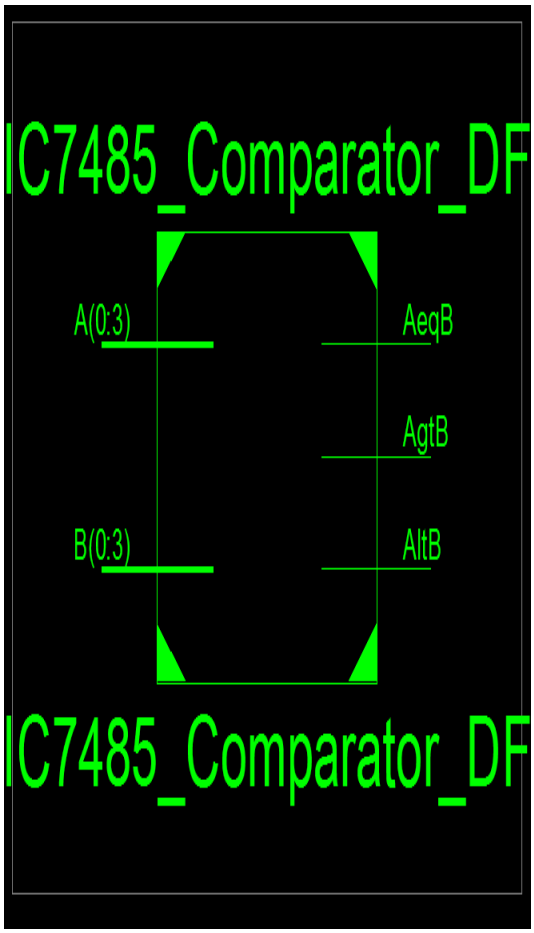
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY IC7485_Comparator_TB IS
END IC7485_Comparator_TB;
ARCHITECTURE behavior OF IC7485_Comparator_TB IS
   COMPONENT IC7485_Comparator
   PORT(
       A : IN  std_logic_vector(0 to 3);
       B : IN  std_logic_vector(0 to 3);
       AgtB : OUT  std_logic;
       AeqB : OUT  std_logic;
       AltB : OUT  std_logic
      );
   END COMPONENT;
  signal A : std_logic_vector(0 to 3) := (others => '0');
  signal B : std_logic_vector(0 to 3) := (others => '0');
  signal AgtB : std_logic;
  signal AeqB : std_logic;
  signal AltB : std_logic;
BEGIN
  uut: IC7485_Comparator PORT MAP (
       A => A,
       B => B,
       AgtB => AgtB,
       AeqB => AeqB,
       AltB => AltB
      );
  stim_proc: process
  begin
                A<="0000";B<="0000"; wait for 50 ns;
                A<="0101";B<="0010"; wait for 50 ns;
                A<="0011";B<="0110"; wait for 50 ns;
                A<="1001";B<="1001"; wait for 50 ns;
                A<="1100";B<="0110"; wait for 50 ns;
                A<="0110";B<="1000"; wait for 50 ns;
                A<="0110";B<="0110"; wait for 50 ns;
                A<="1100";B<="0110"; wait for 50 ns;
```

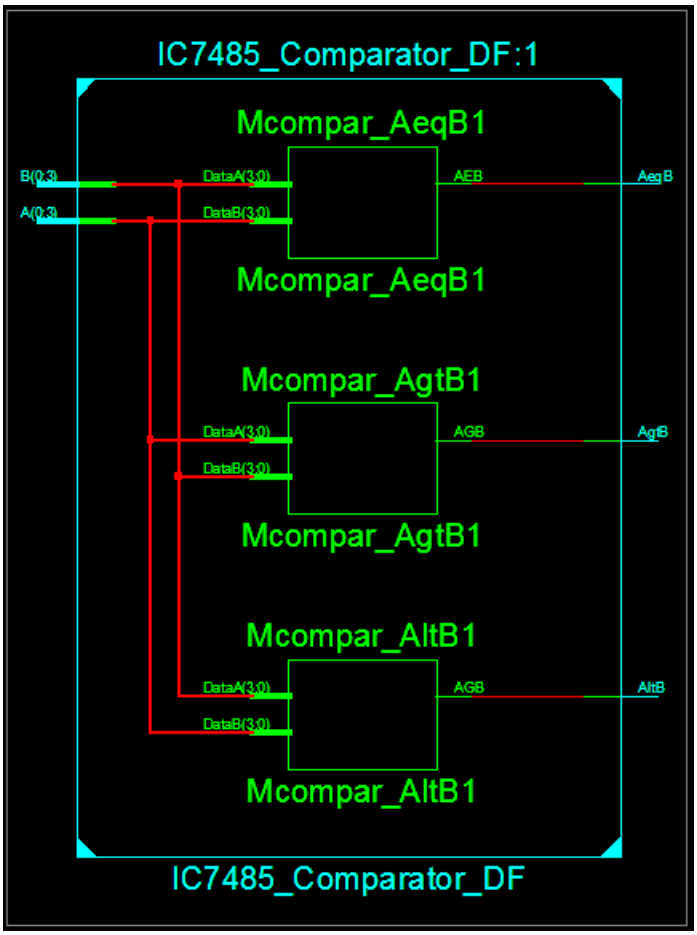
```
                A<="0011";B<="1000"; wait for 50 ns;
  end process;

END;
```

OUTPUT WAVEFORMS:

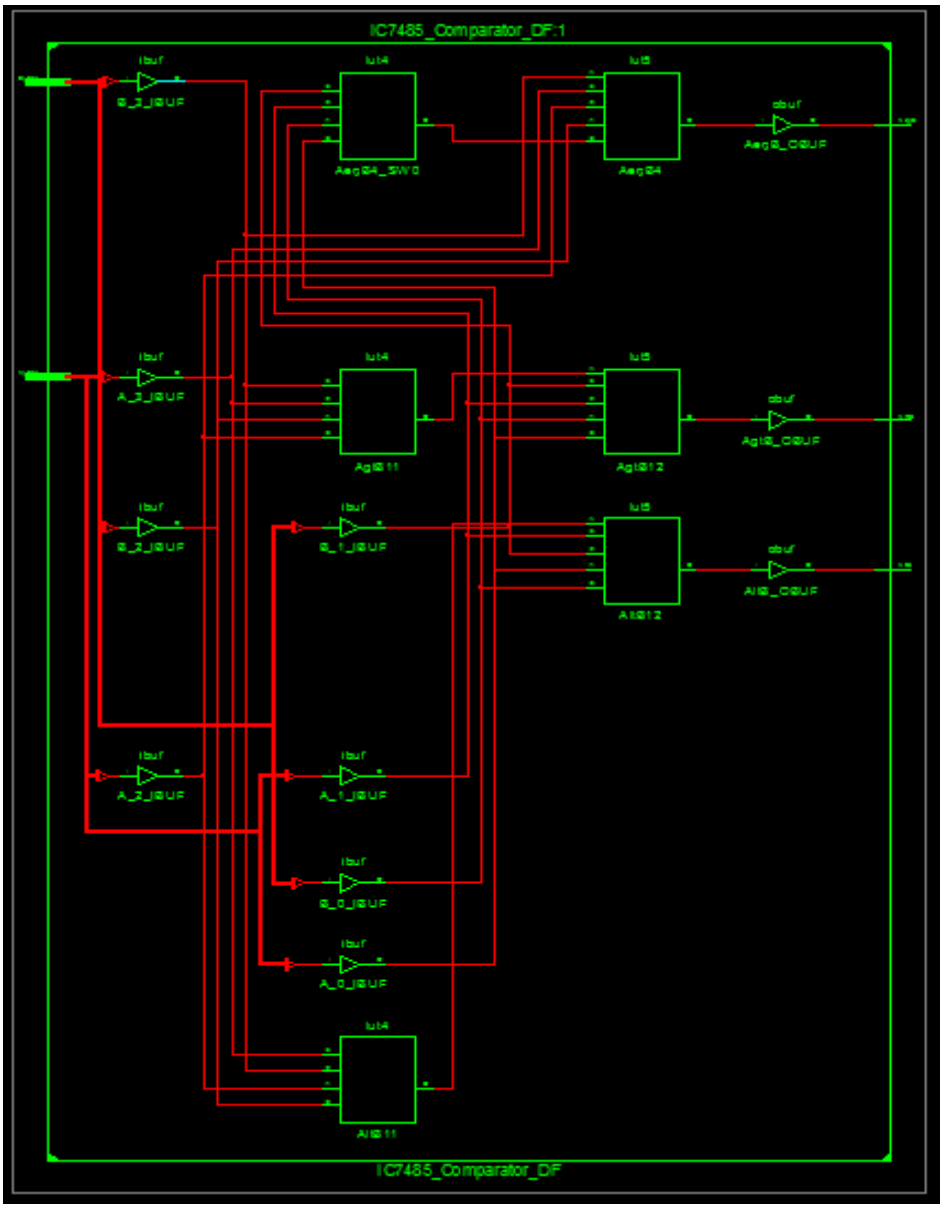


RTL SCHEMATIC:





INTERNAL DIAGRAM OF COMPARATOR:



DEVICE UTILIZATION SUMMARY:

Number of Slices:	5 out of	4656	0%
Number of 4 input LUTs:	9 out of	9312	0%
Number of IOs:	11		
Number of bonded IOBs:	11 out of	232	4%

SYNTHESIS REPORT:

RTL Top Level Output File Name	: IC7485_Comparator.ngr
Top Level Output File Name	: IC7485_Comparator
Output Format	: NGC
Optimization Goal	: Speed
Keep Hierarchy	: No

Design Statistics

# IOs	: 11

Cell Usage:

# BELS	: 11
# LUT2	: 1
# LUT4	: 8
# MUXF5	: 2
# IO Buffers	: 11
# IBUF	: 8
# OBUF	: 3

RESULT:

Thus the VHDL code for 4-BIT COMPARATOR **is** verified, synthesis report is generated and the design is implemented using FPGA.

VIVA QUESTIONS:

1. Write the dataflow model for the IC 74x85.
2. Write the VHDL code for the IC 74x85 using CASE statement.
3. Write the VHDL code for the IC 74x85 using WITH statement.
4. Write the VHDL code for the IC 74x85 using WHEN--ELSE statement.
5. Write the structural program for IC 74x85.
6. How many 4-bit comparators are needed to construct 12-bit comparator?
7. What does a digital comparator mean?
8. Design a 2-bit comparator using gates?
9. Explain the phases of a simulation?
10. Explain briefly about wait statement?

EXPERIMENT – 5
D Flip-Flop-7474

AIM: To design and simulate **D Flip-Flop-7474** using VHDL

SOFTWARE REQIURED:

1. Personal computer
2. ISE Xilinx software

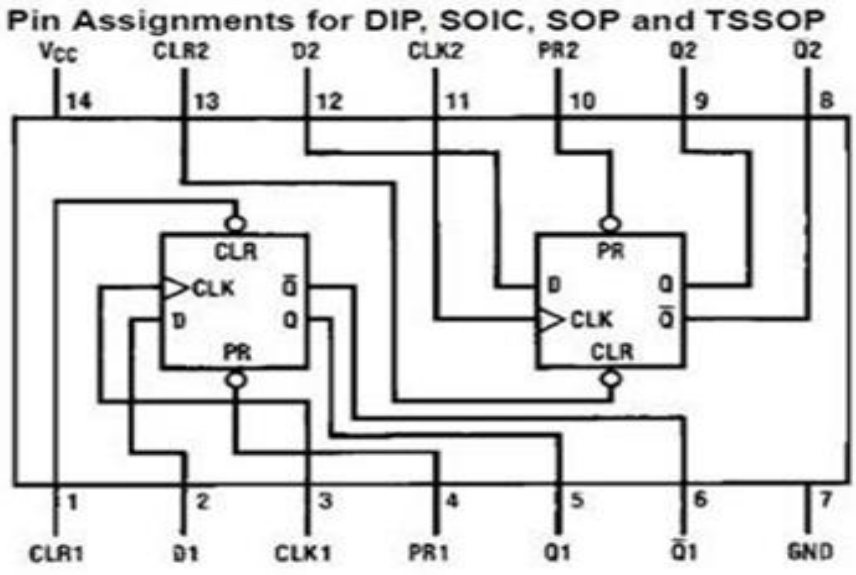
HARDWARE REQUIRED:

1. SPARTAN – 3E KIT

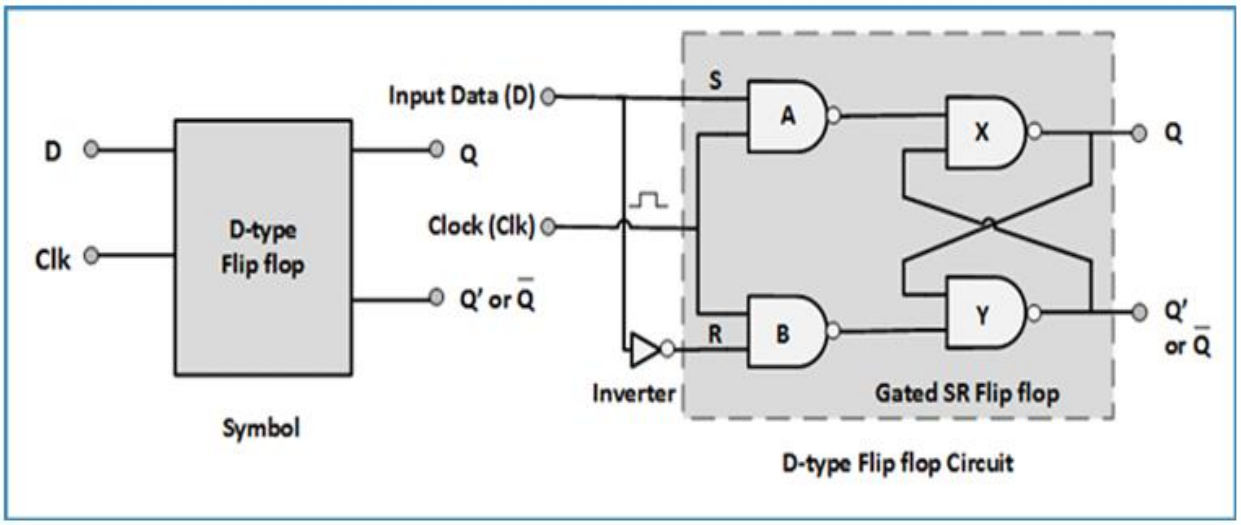
THEORY:

The D-flip flop has only a single data input .The data input is connected to the S input of an RS flip flop, while the inverse of D is connected to the R input. This prevents that the input combination ever occurs. To allow the flip flop to be in a holding state, a D-flip flop has a second input called "Enable." The Enable-input is AND with the D-input, such that when Enable=0, the R & S of the RS-flip flop are 0 and the state is held. When the Enable-input is 1, the S input of the RS flip flop equals the D input and R is the inverse of D determines the value of the output Q when Enable is 1. When Enable returns to 0, the most recent input D is "remembered."

PIN DIAGRAM:



CIRCUIT DIAGRAM:



TRUTH TABLE:

clr_1	pr_1	Clk	d	q	qn
0	0	X	X	1	1
0	1	X	X	0	1
1	0	X	X	1	0
1	1	↑	0	0	1
1	1	↑	1	1	0

PROGRAM IN VHDL:

BEHAVIORAL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity IC7474_DFF_BH is
    Port ( D : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           Pre : in  STD_LOGIC;
           Clr : in  STD_LOGIC;
           Q : inout  STD_LOGIC;
           Qbar : out  STD_LOGIC);
end IC7474_DFF_BH;
architecture Behavioral of IC7474_DFF_BH is
begin
```

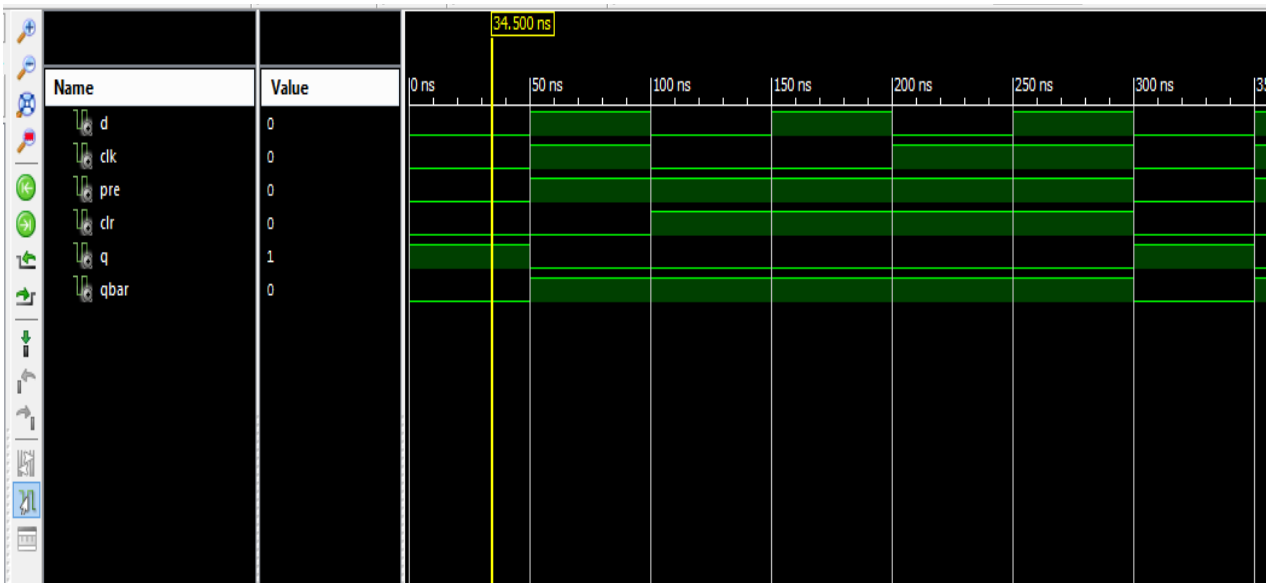
```
        process(D,Clk,Pre,Clr,Q)
        begin
                if Pre='0' then Q<='1';
                else if Clr='0' then Q<='0';
                else if (Clk'event and Clk='1') then Q<= D;
                else
                Q<=Q;
                end if;
                end if;
                end if;
                        Qbar<=not Q;
                end process;
end Behavioral;
```

VHDL TEST BENCH:

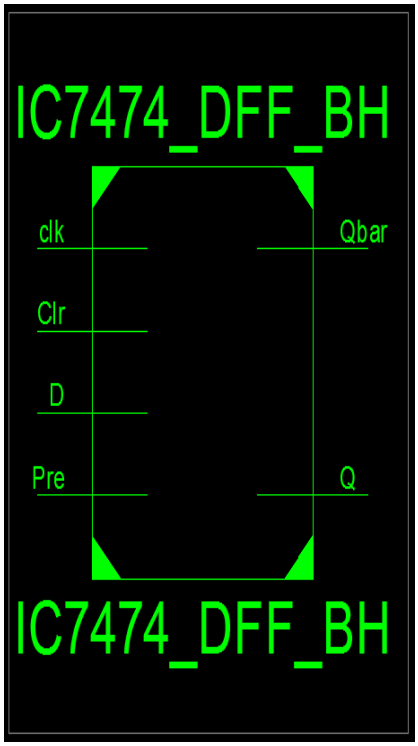
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY IC7474_DFF_TB IS
END IC7474_DFF_TB;
ARCHITECTURE behavior OF IC7474_DFF_TB IS
   COMPONENT IC7474_DFF_BH
   PORT(
        D : IN  std_logic;
        clk : IN  std_logic;
        Pre : IN  std_logic;
        Clr : IN  std_logic;
        Q : INOUT  std_logic;
        Qbar : OUT  std_logic
       );
   END COMPONENT;
  signal D : std_logic := '0';
  signal clk : std_logic := '0';
  signal Pre : std_logic := '0';
  signal Clr : std_logic := '0';
  signal Q : std_logic;
  signal Qbar : std_logic;
BEGIN
  uut: IC7474_DFF_BH PORT MAP (
        D => D,
        clk => clk,
        Pre => Pre,
        Clr => Clr,
        Q => Q,
        Qbar => Qbar
       );
  stim_proc: process
```

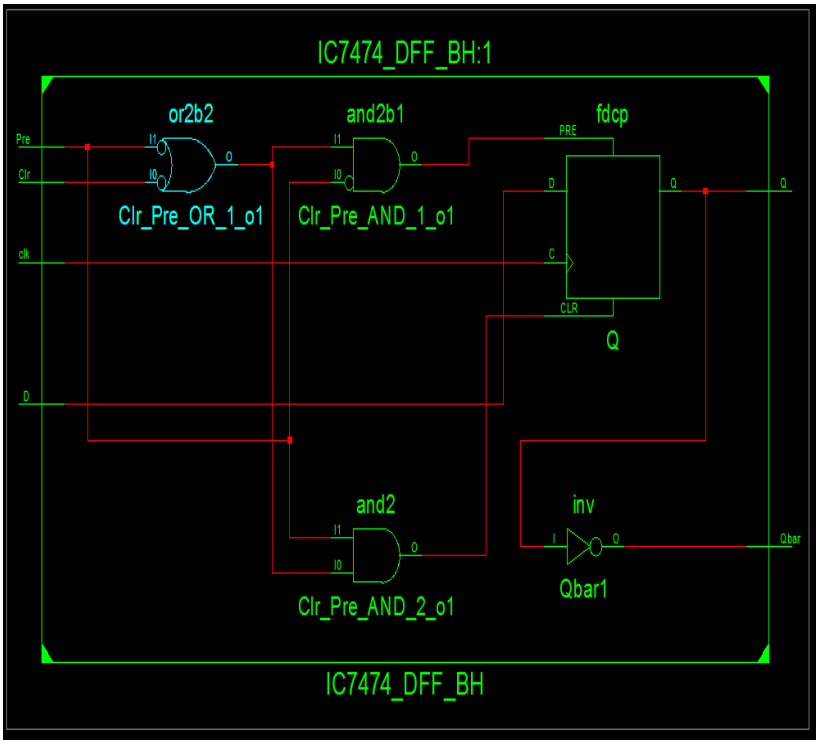
```
  begin
                Pre<='0';Clr<='0';Clk<='0';D<='0';wait for 50 ns;
                Pre<='1';Clr<='0';Clk<='1';D<='1';wait for 50 ns;
                Pre<='1';Clr<='1';Clk<='0';D<='0';wait for 50 ns;
                Pre<='1';Clr<='1';Clk<='0';D<='1';wait for 50 ns;
                Pre<='1';Clr<='1';Clk<='1';D<='0';wait for 50 ns;
                Pre<='1';Clr<='1';Clk<='1';D<='1';wait for 50 ns;
  end process;
END;
```

OUTPUT WAVEFORMS:

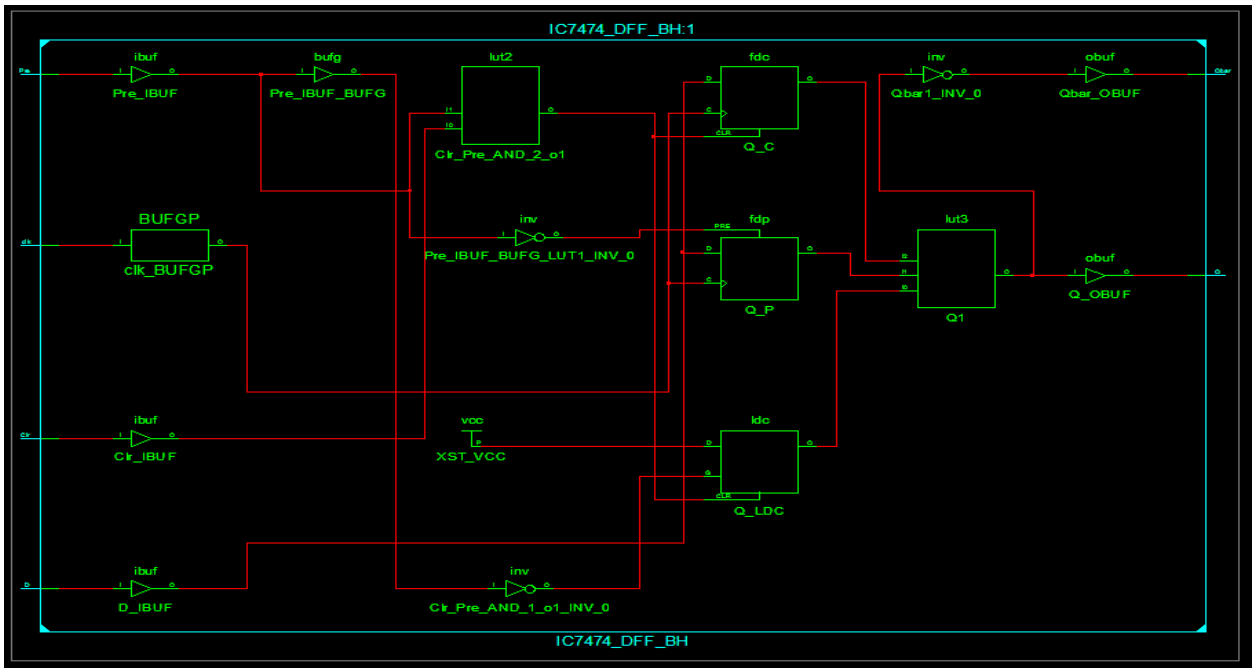


RTL SCHEMATIC:





INTERNAL DIAGRAM OF D FLIPFLOP:



DEVICE UTILIZATION SUMMARY:

Number of Slices:	2 out of	4656	0%
Number of 4 input LUTs:	3 out of	9312	0%
Number of IOs:	6		
Number of bonded IOBs:	6 out of	232	2%
IOB Flip Flops:	2		
Number of GCLKs:	1 out of	24	4%

SYNTHESIS REPORT:

RTL Top Level Output File Name: IC7474_DFF.ngr
Top Level Output File Name : IC7474_DFF
Output Format : NGC
Optimization Goal : Speed
Keep Hierarchy : No

Design Statistics
IOs : 6

Cell Usage:
BELS : 3
INV : 2
LUT2 : 1
Flip Flops/Latches : 2
FDCP : 2

# Clock Buffers	: 1
# BUFGP	: 1
# IO Buffers	: 5
# IBUF	: 3
# OBUF	: 2

RESULT:

Thus the VHDL code for **D Flip-Flop is** verified, synthesis report is generated and the design is implemented using FPGA.

VIVA QUESTIONS:

1. Write the behavioral code for the IC 74x74.
2. Write the dataflow code for the IC 74x74.
3. What is the difference between sequential and combinational circuit?
4. What is a flip-flop?
5. Explain the functions of preset and clear inputs in flip-flop?
6. What is meant by a clocked flip-flop?
7. What is meant by excitation table?
8. What is the difference between flip-flop and latch?
9. What are the various methods used for triggering flip-flops?
10. Explain level triggered flip-flop?
11. Write the behavioral code for IC 74X74.
12. Write the syntax of IF statement?

EXPERIMENT – 6

ALU

AIM: To design and simulate 16 bit ALU

SOFTWARE REQIURED:

1. Personal computer
2. ISE Xilinx software

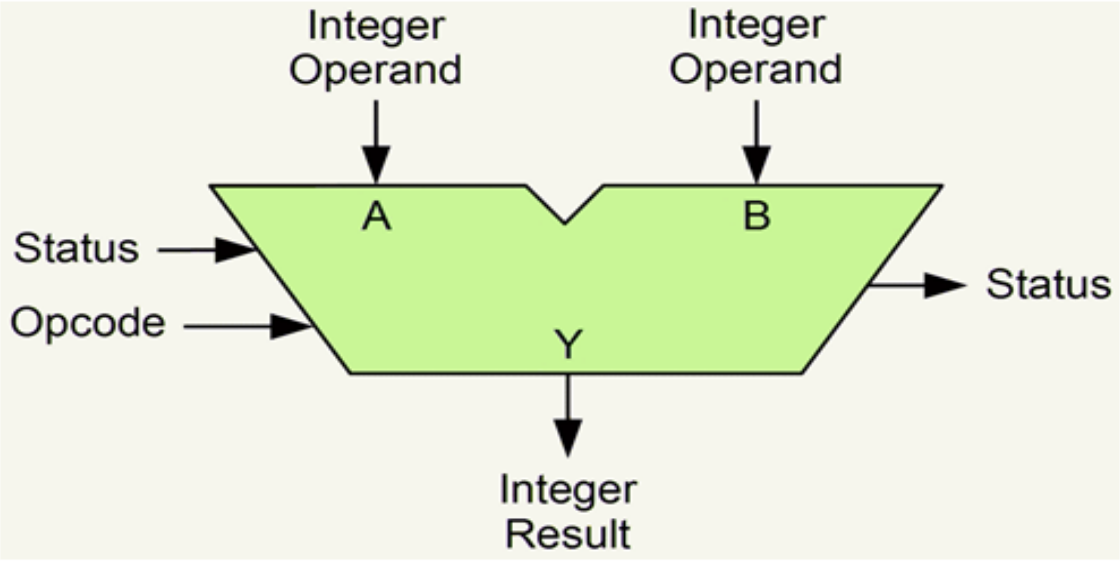
HARDWARE REQUIRED:

1. SPARTAN – 3E KIT

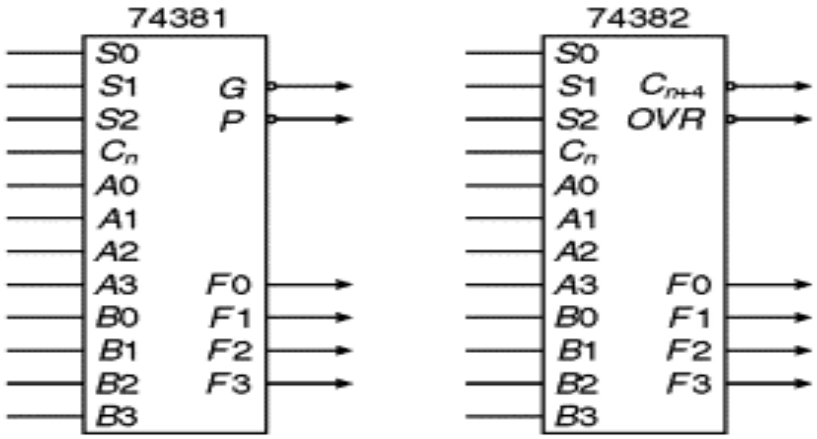
THEORY:

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.



PIN DIAGRAM:



TRUTH TABLE:

Selection			Arithmetic/logic function
S2	S1	S0	
0	0	0	Clear
0	0	1	B minus A
0	1	0	A minus B
0	1	1	A plus B
1	0	0	$A \oplus B$
1	0	1	$A + B$
1	1	0	AB
1	1	1	Preset

PROGRAM IN VHDL:

DATAFLOW:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
entity ALU_74381IC_DF is
    Port ( S : in  STD_LOGIC_VECTOR (2 downto 0);
           A,B : in  STD_LOGIC_VECTOR (3 downto 0);
           F : out  STD_LOGIC_VECTOR (3 downto 0);
                          Cin: in std_logic);
end ALU_74381IC_DF;
architecture Dataflow of ALU_74381IC_DF is
begin
          with S select
                    F<="0000" when "000",
                         B-A-1+Cin when "001",
                        A-B-1+Cin when "010",
```

```
                    A+B+Cin when "011",
                    A xor B when "100",
                    A or B when "101",
                    A and B when "110",
                   "1111" when others;
end Dataflow;
```

BEHAVIORAL:

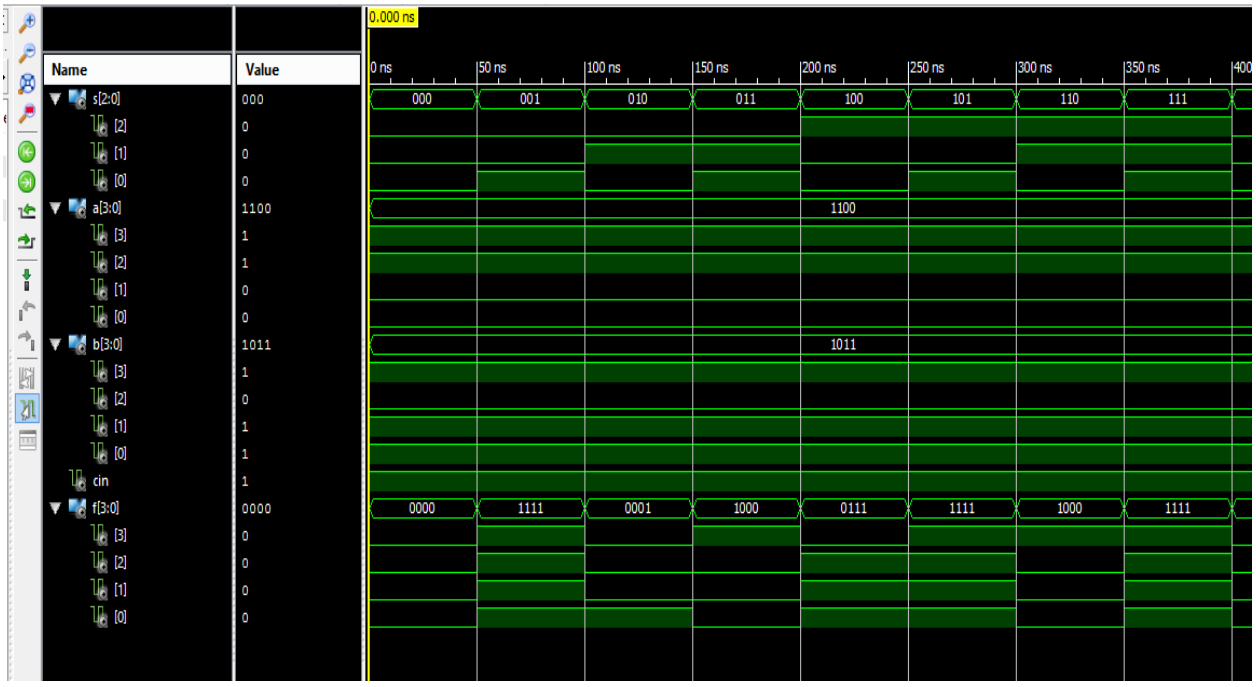
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
entity ALU_74381IC_BH is
   Port ( S : in  STD_LOGIC_VECTOR (2 downto 0);
          A,B : in  STD_LOGIC_VECTOR (3 downto 0);
          F : out  STD_LOGIC_VECTOR (3 downto 0);
                    Cin: in std_logic);
end ALU_74381IC_BH;

architecture Behavioral of ALU_74381IC_BH is
begin
        process (S,A,B,Cin)
                begin
                        case S is
                                when "000"=>F<="0000";
                                when "001"=>F<=B-A-1+Cin;
                                when "010"=>F<=A-B-1+Cin;
                                when "011"=>F<=A+B+Cin;
                                when "100"=>F<=A xor B;
                                when "101"=>F<=A or B;
                                when "110"=>F<=A and B;
                                when others=>F<="1111";
                        end case;
        end process;
end Behavioral;
```

VHDL TEST BENCH:

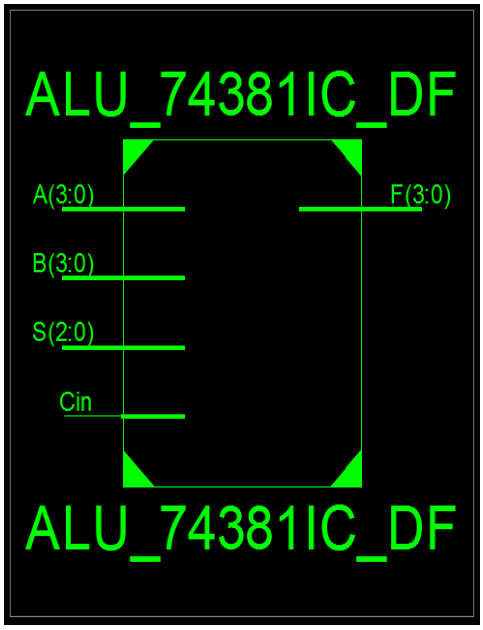
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY ALU_74381IC_TB IS
END ALU_74381IC_TB;
ARCHITECTURE behavior OF ALU_74381IC_TB IS
   COMPONENT ALU_74381IC
   PORT(
      S : IN  std_logic_vector(2 downto 0);
      A : IN  std_logic_vector(3 downto 0);
      B : IN  std_logic_vector(3 downto 0);
      F : OUT  std_logic_vector(3 downto 0);
```

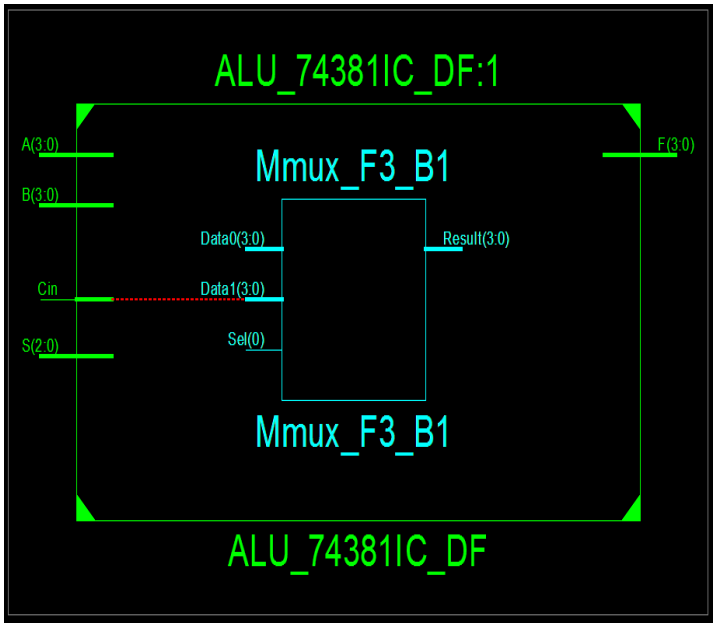
```
        Cin : IN  std_logic
       );
    END COMPONENT;
   signal S : std_logic_vector(2 downto 0) := (others => '0');
   signal A : std_logic_vector(3 downto 0) := (others => '0');
   signal B : std_logic_vector(3 downto 0) := (others => '0');
   signal Cin : std_logic := '0';
   signal F : std_logic_vector(3 downto 0);
BEGIN
   uut: ALU_74381IC PORT MAP (
         S => S,
         A => A,
         B => B,
         F => F,
         Cin => Cin
       );
   stim_proc: process
   begin
            S<="000";A<="1100";B<="1011";Cin<='1';wait for 50 ns;
            S<="001";A<="1100";B<="1011";Cin<='1';wait for 50 ns;
            S<="010";A<="1100";B<="1011";Cin<='1';wait for 50 ns;
            S<="011";A<="1100";B<="1011";Cin<='1';wait for 50 ns;
            S<="100";A<="1100";B<="1011";Cin<='1';wait for 50 ns;
            S<="101";A<="1100";B<="1011";Cin<='1';wait for 50 ns;
            S<="110";A<="1100";B<="1011";Cin<='1';wait for 50 ns;
            S<="111";A<="1100";B<="1011";Cin<='1';wait for 50 ns;
   end process;
END;
```

OUTPUT WAVEFORMS:

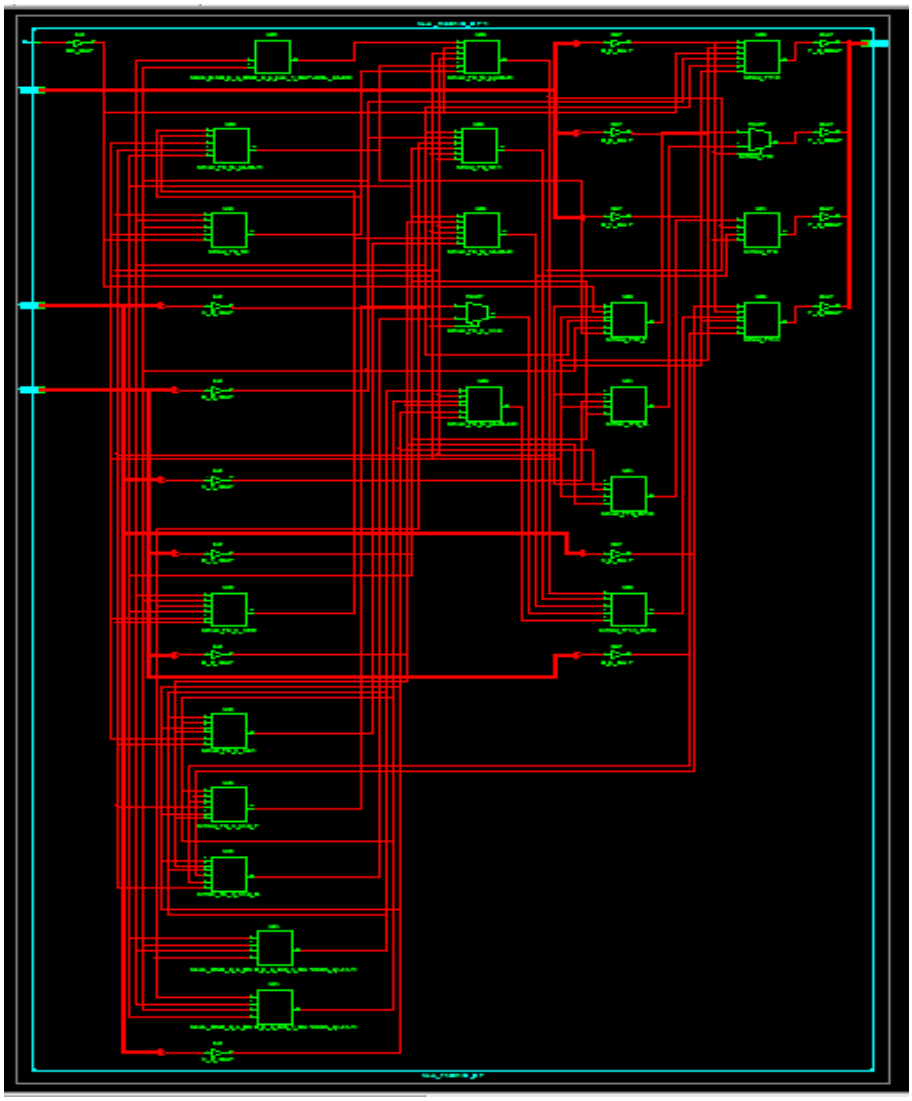


RTL SCHEMATIC:





INTERNAL DIAGRAM OF DECADECOUNTER:



DEVICE UTILIZATION SUMMARY:

Number of Slices:	13 out of 4656 0%
Number of 4 input LUTs:	23 out of 9312 0%
Number of IOs:	16
Number of bonded IOBs:	16 out of 232 6%

SYNTHESIS REPORT:

RTL Top Level Output File Name : ALU_74381IC.ngr
Top Level Output File Name : ALU_74381IC
Output Format : NGC
Optimization Goal : Speed
Keep Hierarchy : No

Design Statistics
IOs : 16

Cell Usage:
BELS : 37
LUT3 : 11
LUT4 : 12
MULT_AND : 2
MUXCY : 3
MUXF5 : 5
XORCY : 4
IO Buffers : 16
IBUF : 12
OBUF : 4

RESULT:

Thus the VHDL code for ALU is verified, synthesis report is generated and the design is implemented using FPGA

VIVA QUESTIONS:

1. Write the behavioral code for the IC 74x74.
2. Write the dataflow code for the IC 74x74.
3. What is the difference between sequential and combinational circuit?
4. What is a flip-flop?
5. Explain the functions of preset and clear inputs in flip-flop?
6. What is meant by a clocked flip-flop?
7. What is meant by excitation table?
8. What is the difference between flip-flop and latch?
9. What are the various methods used for triggering flip-flops?
10. Explain level triggered flip-flop?
11. Write the behavioral code for IC 74X74.
12. Write the syntax of IF statement?

EXPERIMENT – 7

BLINKING LED'S AND PUSH BUTTON INTERFACE USING TM4CGH6PM

AIM:

To Write a C program for configuration of GPIO ports for Input and output operation (blinking LEDs, push buttons interface) using TM4C123GXL Launch Pad.

a) Modify the code to make the red LED of EK-TM4C123GXL Launchpad blink.
b) Modify the code to make the green and red LEDs blink: I. Together II. Alternately
c) Alter the code to turn the LED ON when the button is pressed and OFF when it is released.
d) Modify the delay with which the LED blinks.
e) Alter the code to make the green LED stay ON for around 1 second every time the button is pressed.
f) Alter the code to turn the red LED ON when the button is pressed and the green LED ON when the button is released.

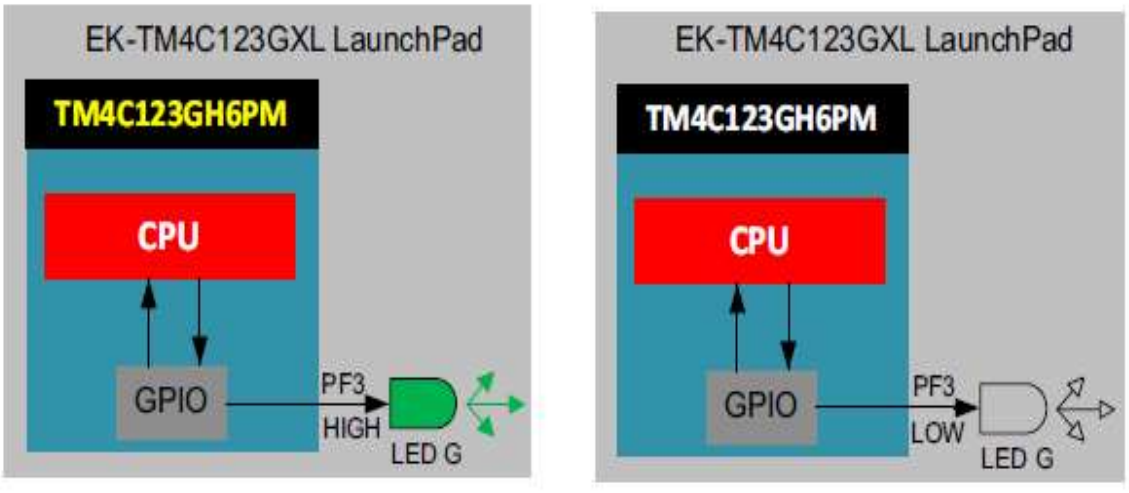
APPARATUS:

1. Software Required: Code Composer Studio (CCS)
2. Hardware Required: TIVA TM4C123GXL Launch Pad

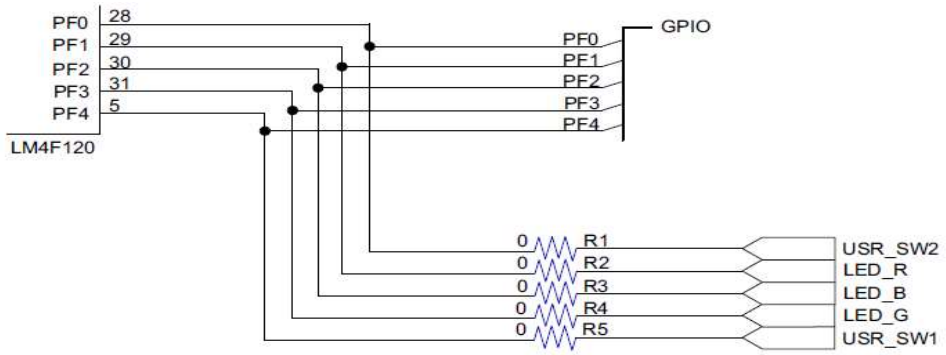
PROCEDURE:

1. Connect the TM4C123GXL to the PC using the USB cable supplied.
2. Build, program and debug the code into the Launch Pad using CCS to view the status of the green LED.

BLOCK DIAGRAM:



Launch Pad Schematics for GPIO Connected to LEDs



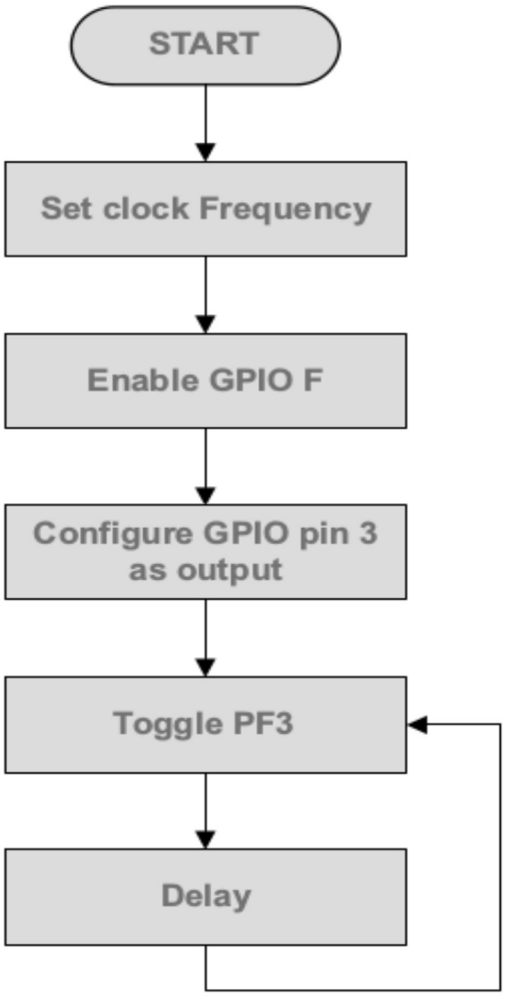
GPIO Pin Functions and USB Device Connected

GPIO Pin	Pin Function	USB Device
PF4	GPIO	SW1
PF0	GPIO	SW2
PF1	GPIO	RGB LED (Red)
PF2	GPIO	RGB LED (Blue)
PF3	GPIO	RGB LED (Green)

PROGRAM :

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
int main(void)
{
SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|
SYSCTL_OSC_MAIN);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
while(1){
GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x08);
SysCtlDelay(20000000);
GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00);
SysCtlDelay(20000000);
}
}
```

FLOW CHART:



RESULT:

VIVA:

1. What is TM4C123GH6PM?
2. How many GPIO pins are there in TM4C123GH6PM?
3. What is the maximum clock frequency of TM4C123GH6PM?
4. What is the basic function of GPIO control?
5. What are the various development tools in the TIVA C series programming?

EXPERIMENT – 8

TIMER BASED INTERRUPT PROGRAMMING USING TM4C123GXL

AIM:

To write a C program for EK-TM4C123GXL Launchpad and associated Timer ISR to toggle onboard LED using interrupt programming technique.
a) Modify the code for a different timer toggling frequency.
b) Write the code to turn on interrupt globally.

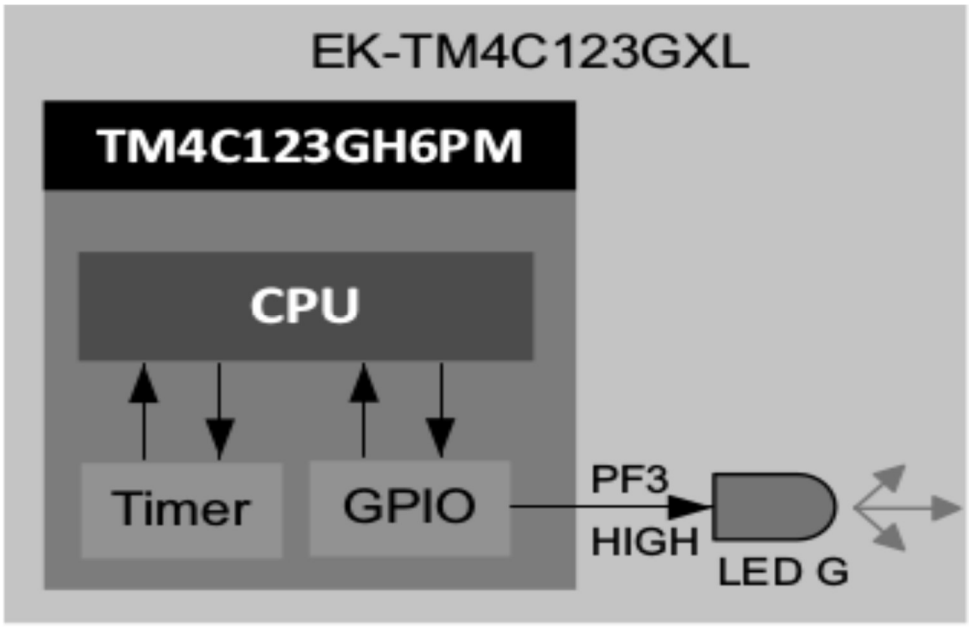
APPARATUS:

1. Software Required: Code Composer Studio (CCS)
2. Hardware Required: TIVA TM4C123GXL Launch Pad

PROCEDURE:

1. Connect the EK-TM4C123GXL to the PC using the USB cable supplied.
2. Build, program and debug the code into the EK-TM4C123GXL using CCS to View the status of the green LED.

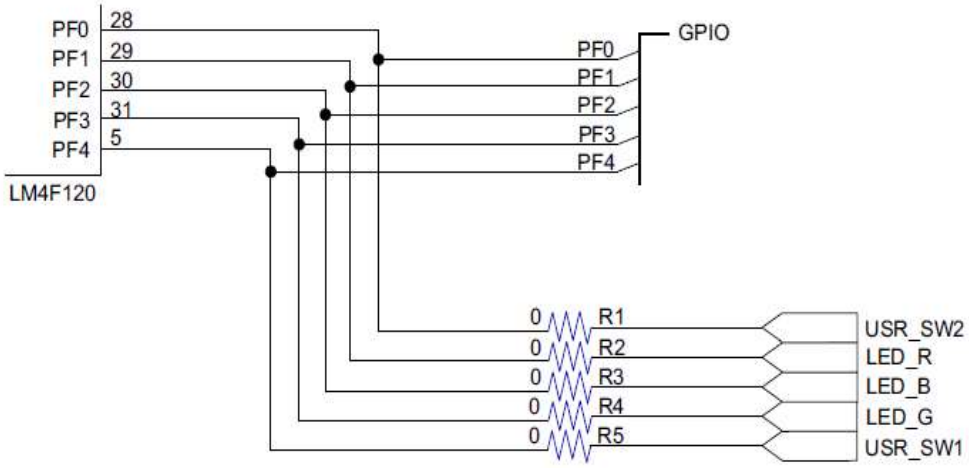
BLOCK DIAGRAM:



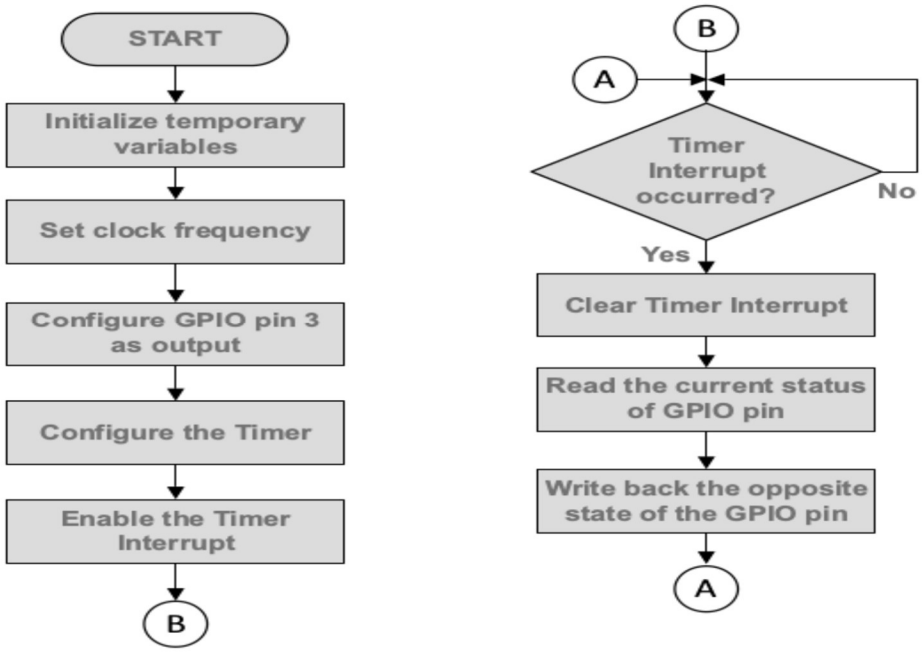
Launch Pad Schematics for GPIO Connected to LEDs

GPIO Pin	Pin Function	USB Device
PF4	GPIO	SW1
PF0	GPIO	SW2
PF1	GPIO	RGB LED (Red)
PF2	GPIO	RGB LED (Blue)
PF3	GPIO	RGB LED (Green)

GPIO Pin Functions and USB Device Connected



FLOW DIAGRAM:



Calculation of timer period ui32Period

The timer counts for every clock cycle of system frequency.

The number of timer counts required to obtain a given frequency is calculated by

Number of clock cycles = System Clock Frequency / Desired Frequency

In the program, to toggle the GPIO at 10Hz and 50% duty cycle,

ui32Period = Number of clock cycles * Duty cycle = (40 MHz /10 Hz) / 2 =2 * 10^6 counts

PROGRAM:

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
int main(void)
{
uint32_t ui32Period;
SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|
SYSCTL_OSC_MAIN);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
ui32Period = (SysCtlClockGet() / 10) / 2;
TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
IntEnable(INT_TIMER0A);
TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
IntMasterEnable();
}
TimerEnable(TIMER0_BASE, TIMER_A);
while(1) { }
void Timer0IntHandler(void)
{ // Clear the timer interrupt
TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
// Read the current state of the GPIO pin and write back the opposite state
if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
{ GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
}
else
{ GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
}
}
```

RESULT:

VIVA:

1. What is watchdog timer?
2. What is an interrupt?
3. What are the various hardware and software type of interrupts in detail?
3. What is pooling?
4. What is predefined interrupts?

EXPERIMENT – 9

HIBERNATION MODULE FOR TM4C123GH6PM MICROCONTROLLER

AIM:

To configure hibernation module of the TM4C123GH6PM microcontroller to place the device in low power state and then to wake up the device on RTC (Real- Time Clock) Interrupt.

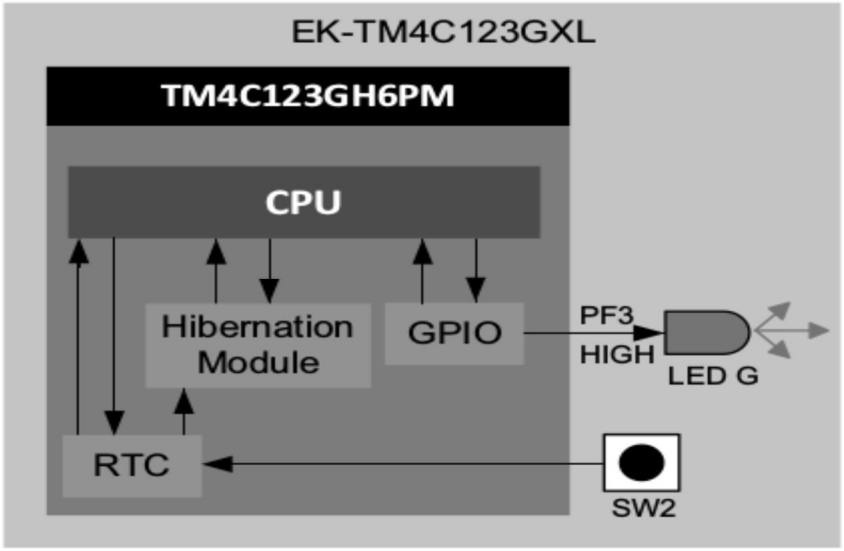
APPARATUS:

1. Software Required: Code Composer Studio (CCS)
2. Hardware Required: TIVA TM4C123GH6PM Launch Pad

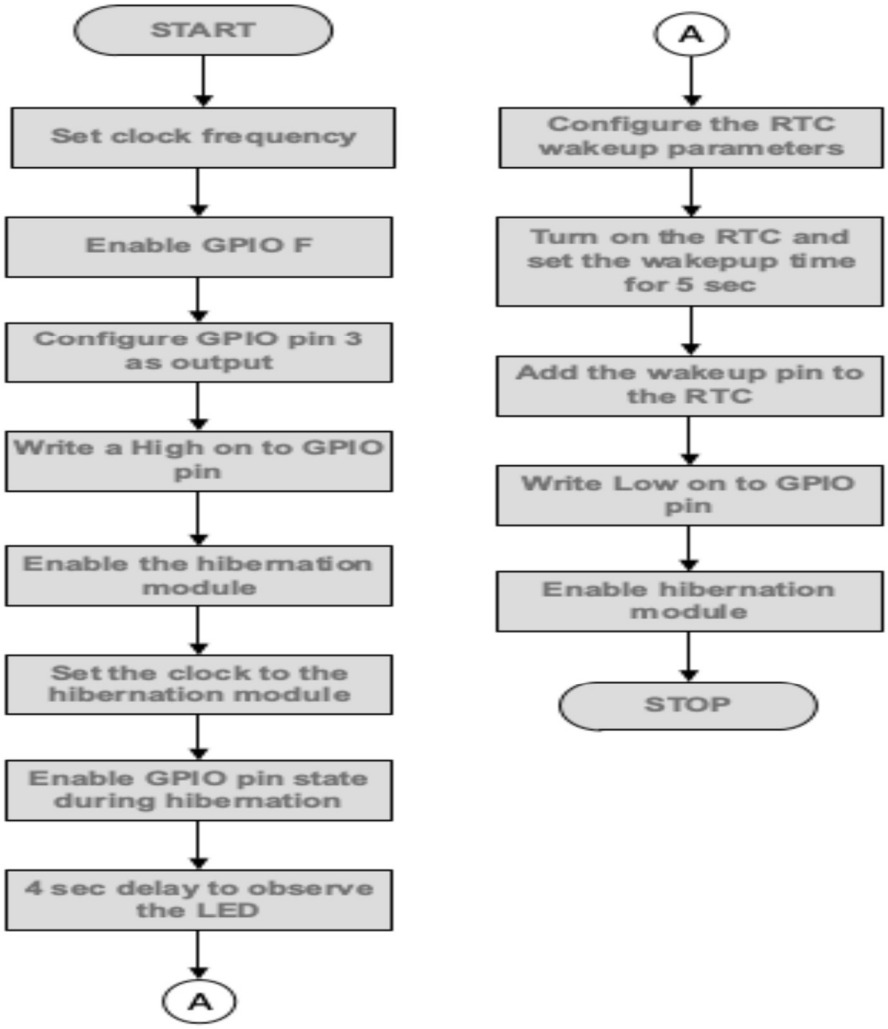
PROCEDURE:

1. Connect the EK-TM4C123GXL to the PC using the USB cable supplied.
2. Build, program and debug the code to view the status of the green LED.
3. After 4 seconds, the green LED will switch off, indicating that the TM4C123GH6PM device has gone into hibernation.
4. Observe the status of the LED. After 5 seconds (RTC wake up time set in the code), the LED turns ON, indicating the RTC has woken the processor.
5. Also you can press and hold the SW2 button located at the lower right corner of the EKTM4C123GXL to wake up the processor at any time.
6. On wake up the green LED will turn ON again.

BLOCK DIAGRAM:



FLOW CHART:



PROGRAM :

```
#include <stdint.h>
#include <stdbool.h>
#include "utils/ustdlib.h"
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/debug.h"
#include "driverlib/hibernate.h"
#include "driverlib/gpio.h"
int main(void)
{
SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|
```

```
SYSCTL_OSC_MAIN);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x08);
SysCtlPeripheralEnable(SYSCTL_PERIPH_HIBERNATE);
HibernateEnableExpClk(SysCtlClockGet());
HibernateGPIORetentionEnable();
SysCtlDelay(64000000);
HibernateRTCSet(0);
HibernateRTCEnable();
HibernateRTCMatchSet(0,5);
HibernateWakeSet(HIBERNATE_WAKE_PIN | HIBERNATE_WAKE_RTC);
GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_3, 0x00);
HibernateRequest();
while(1){}
}
```

RESULT:

VIVA:

1. What is the Hibernation Module on Tiva Microcontrollers?
2. What is RTC?
3. What are the two mechanisms of Hibernation Module for power control?
4. What are the power modes of TM4C123GH6PM?
5. How Hibernation module power source is determined dynamically?

EXPERIMENT – 10

IN-BUILD ADC OF TM4C123GH6PM & POTENTIOMETER WITH TM4C123GXL

AIM:

To configure in build ADC of TM4C123GH6PM microcontroller and interface potentiometer with EK-TM4C123GXL Launchpad to observe corresponding 12-bit digital value.

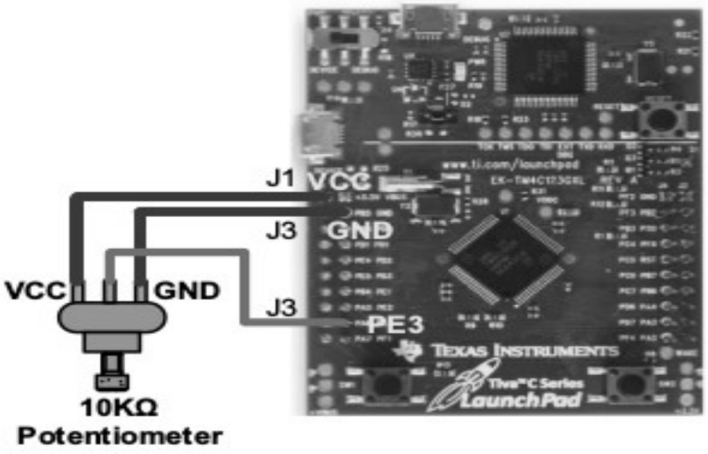
APPARATUS:

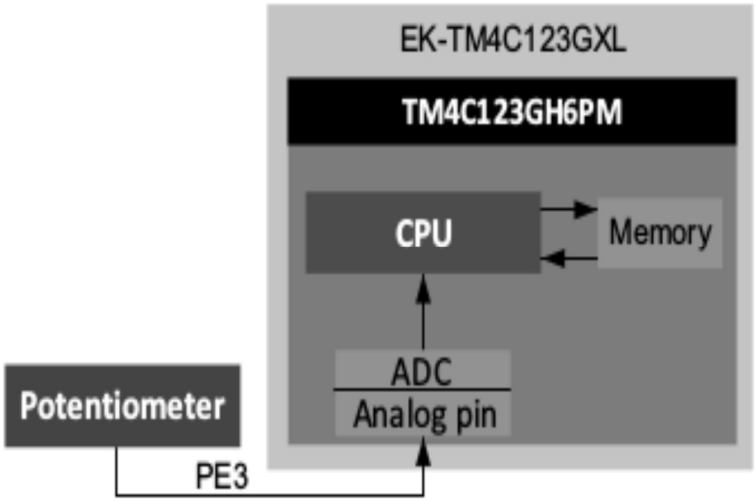
1. Software Required: Code Composer Studio (CCS)
2. Hardware Required: TIVA TM4C123GH6PM Launch Pad

PROCEDURE:

1. Connect one lead of the potentiometer (Vcc) to +3.3V DC Supply Voltage (J1 connector Pin1).
2. Connect the other lead of the potentiometer to the GND pin (J3 connector, Pin 2).
3. Connect the center lead of the potentiometer to pin PE3 which is the Analog Channel AN0 (J3 connector, Pin 9).
4. The Millimeter can be probed at the center lead of the potentiometer to observe the analog voltage input.
5. This configuration varies the Analog Voltage from 0V to 3.3V depending on the wiper position on PE3 which is the AN0 or Analog Input 0 of the TM4C123GH6PM.
6. The current configuration connects the VREFP of ADC to VDDA which is 3.3V and VREFP to GND which is 0V.
7. Build, program and debug the code.
8. Vary the position of the potentiometer wiper and observe the corresponding digital output stored in the register ui3on the CCS window.

BLOCK DIAGRAM:





Design Calculations

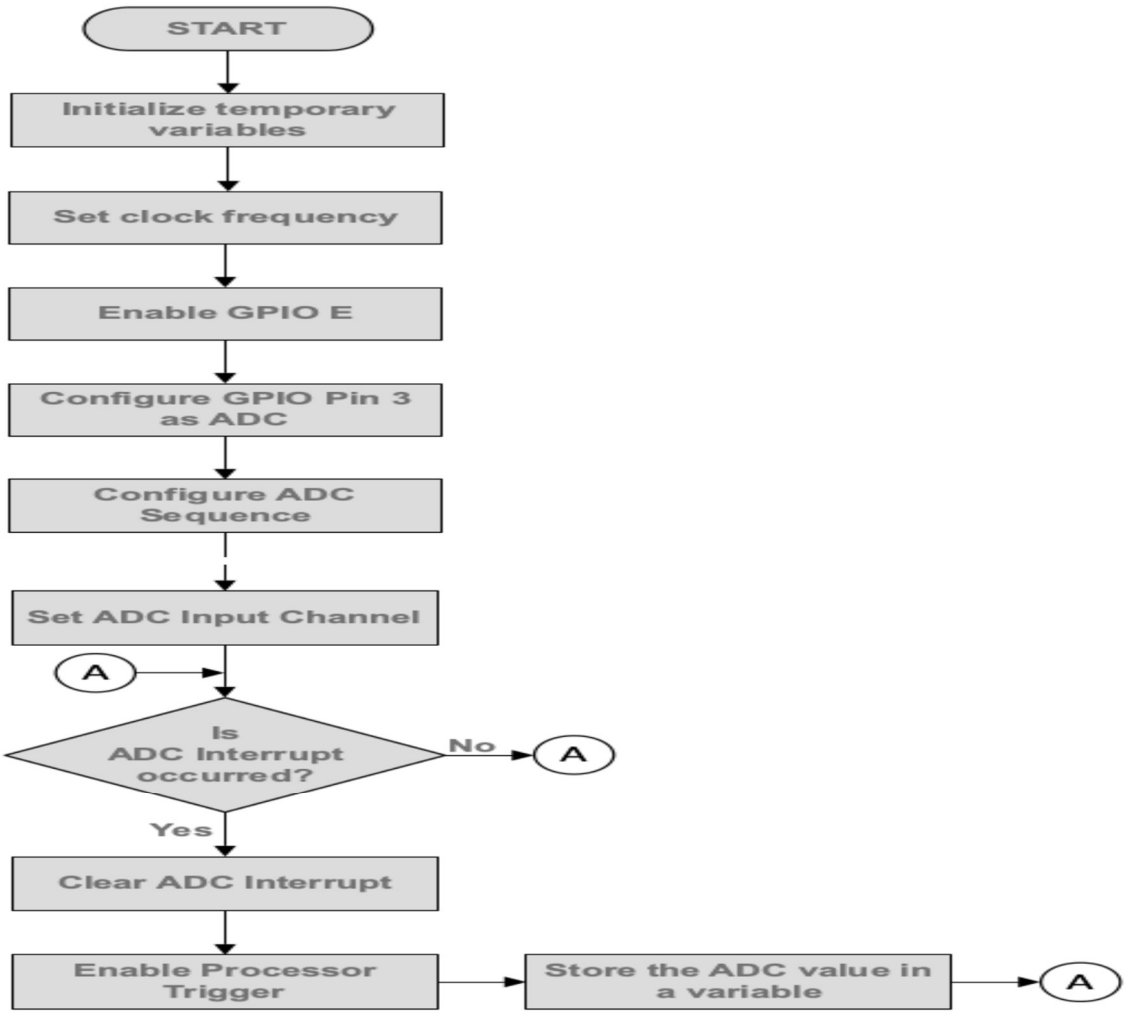
For a 12-bit ADC, the range of the conversion values is from 0 to 4095 (0 to 0xfff in Binary).
With the Conversion voltage span and the bit range we can calculate the following
Resolution = 3.3V/ 4096 = 3300mV / 4096 = 0.8mV
The resolution is the change in voltage per unit change in ADC code
For example, if the digital code is equivalent to 1000 in decimal,
The equivalent analog voltage for 1000 = Resolution * decimal equivalent of the digital code = 0.8mV *1000 = 800 mV = 0.8V

PROGRAM :

```
#include<stdint.h>
#include<stdbool.h>
#include"inc/hw_memmap.h"
#include"driverlib/gpio.h"
#include"inc/hw_types.h"
#include"driverlib/debug.h"
#include"driverlib/sysctl.h"
#include"driverlib/adc.h"
// TO STORE THE VALUE IN VARIABLE ui32ADC0Value FOR EVERY SAMPLING
uint32_tui32ADC0Value[1];
intmain(void)
{
SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN
SYSCTL_XTAL_16MHZ); // SYSTEM CLOCK AT 40MHZ
SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0); // ENABLE ADC0 MODULE
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
MODULE // ENABLE GPIO for ADC0
GPIOPinTypeADC(GPIO_PORTE_BASE,GPIO_PIN_3);// ENABLE AN0 OF ADC0
```

```
MODULE
// ADC0 MODULE, TRIGGER IS PROCESSOR EVENT, SEQUENCER 0 IS
CONFIGURED ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
// ADC0 MODULE, SEQUENCER 0 , FOR 1 SAMPLING, INPUT IS FROM CHANNEL 0
PE3 ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_CH0);
// ENABLE THE SEQUENCE 1 FOR ADC0
ADCSequenceEnable(ADC0_BASE, 1);
while(1) {
// CLEAR INTERRUPT FLAG FOR ADC0, SEQUENCER 1
ADCIntClear(ADC0_BASE, 1);
// TRIGGER IS GIVEN FOR ADC 0 MODULE, SEQUENCER 1
ADCProcessorTrigger(ADC0_BASE, 1);
// STORE THE CONVERTED VALUE FOR ALL DIFFERENT SAMPLING IN ARRAY
//ui32ADC0Value
ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value); }
}
```

FLOW CHART:



RESULT:

VIVA:

1. What is ADC**?**
2. What is potentiometer**?**
3. What is Successive Approximation Register (SAR) architecture**?**
4. What are the TM4C123GH6PM ADC module features?
5. What is the use of Digital Comparator Unit in ADC module?

EXPERIMENT – 11

PWM AND ADC MODULES OF TM4C123GH6PM MICROCONTROLLER

AIM:

To Configure the PWM and ADC Modules of TM4C123GH6PM Microcontroller for control the speed of DC Motor with Potentiometer Output.

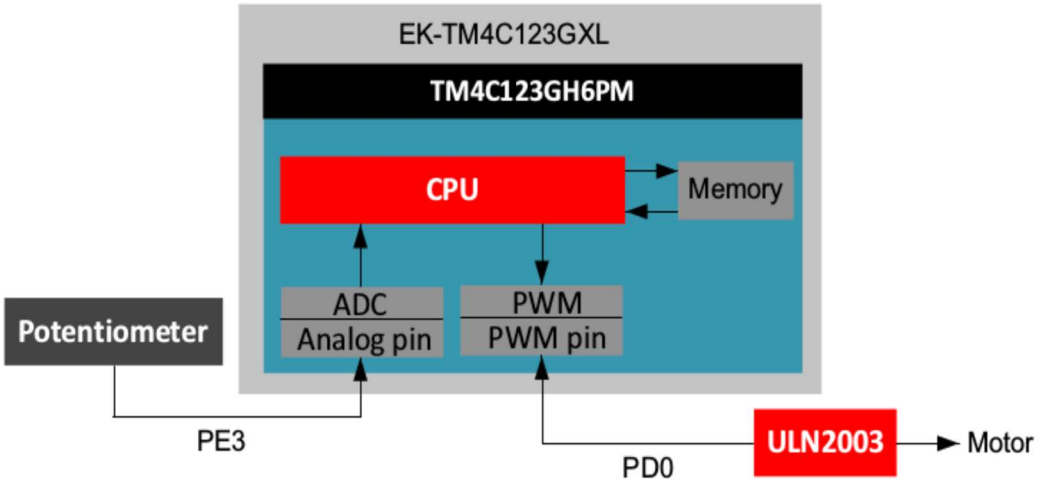
APPARATUS:

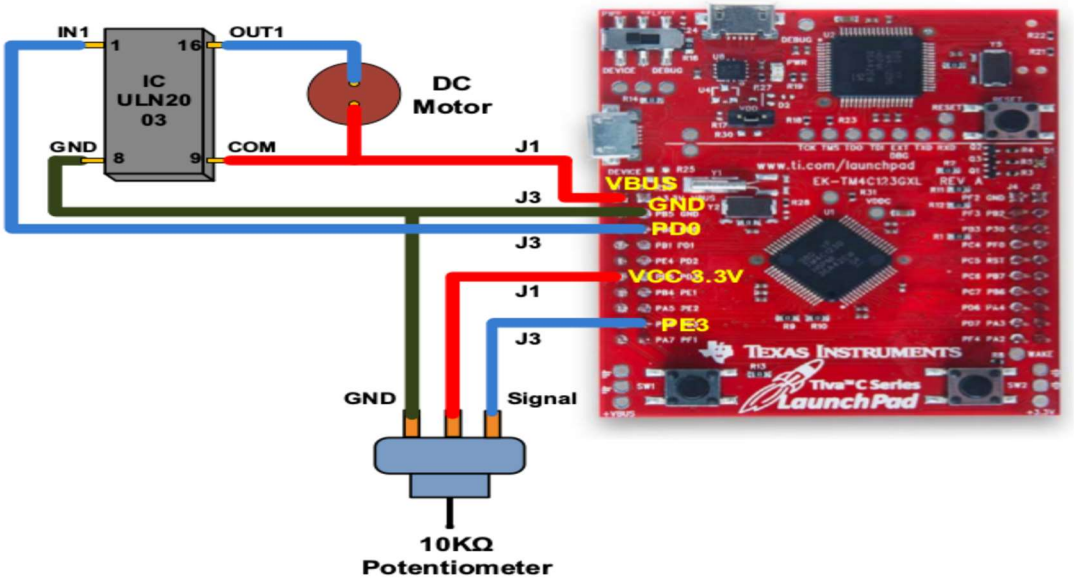
1. Software Required: Code Composer Studio (CCS)
2. Hardware Required: TIVA TM4C123GH6PM Launch Pad

PROCEDURE:

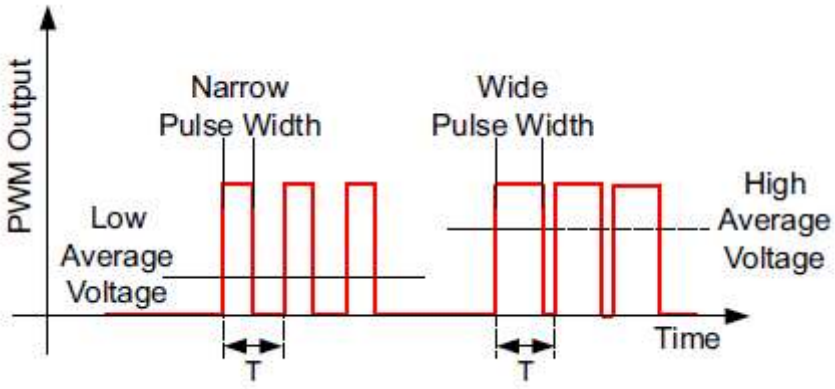
1. Position DIP IC ULN2003A on a Breadboard.
2. Connect one terminal of the DC motor to the Common pin (IC Pin 9) of ULN2003.
3. Connect the junction of the above 2 terminals to the VBUS (5V Power from USB) (Available at J3 Connector Pin 1).
4. Connect the other terminal of the DC motor to the Drive Pin (IC Pin 16) of ULN 2003.
5. Connect J3 Connector Pin 2 (GND) to the Ground Pin (IC Pin 8) of ULN2003.
6. Connect the PWM Output PD0 (J3 connector Pin 3) to Input Signal (IC Pin 1) of ULN2003.
7. Connect one lead of the Potentiometer to the +3.3V Supply Voltage (J1 connector, Pin 1).
8. Connect other lead of the Potentiometer to the GND Pin of ULN2003 (IC Pin 8).
9. Connect the center lead of the Potentiometer (variable analog output) to PE3 (J3 Connector Pin 9) which is the AN0 or Analog Input 0 of the Tiva TM4C123GH6PM.
10. Build, program and debug the code.
11. Vary the position of the potentiometer wiper and observe the corresponding digital output on the CCS window.
12. Also, observe the PWM waveform at J3 connector pin 3 using an oscilloscope.
13. Observe the motion of the DC motor connected to the PWM output.

BLOCK DIAGRAM:





Analog Output for PWM Signal



Design Calculations

- The calculations for the configuration of ADC and PWM are as given below.
- This configuration varies the Analog Voltage from 0V to 3.3V depending on the wiper position on PE3 which is the AN0 or Analog Input 0 of the microcontroller.
- The current configuration connects the VREFP of ADC to VDDA which is 3.3V and VREFP to GND which is 0V.
- The range of the conversion values is from 0 to 4095 which is 0 to 0xfff in Binary since it is a 12- bit ADC.
- With the Conversion voltage span and the bit range we can calculate the following
• Resolution = 3.3V/ 4096 = 3300mV / 4096 = 0.8mV
• Voltage per ADC code = 3.3V / 4096
• If Digital Code = 1000 in decimal, multiply this value by resolution
• Equivalent Voltage for 1000 = 0.8mV *1000 = 800 mV = 0.8V
- Varying the Potentiometer changes the Digital Value between 0 and 4095.
- The PWM period is set to 4095 to match the resolution of the 12-bit ADC.
- The ADC value is read and directly passed onto the duty cycle configuration of the PWM register. It is also possible to have other values, but necessary scaling has to be done.

Calculation for PWM

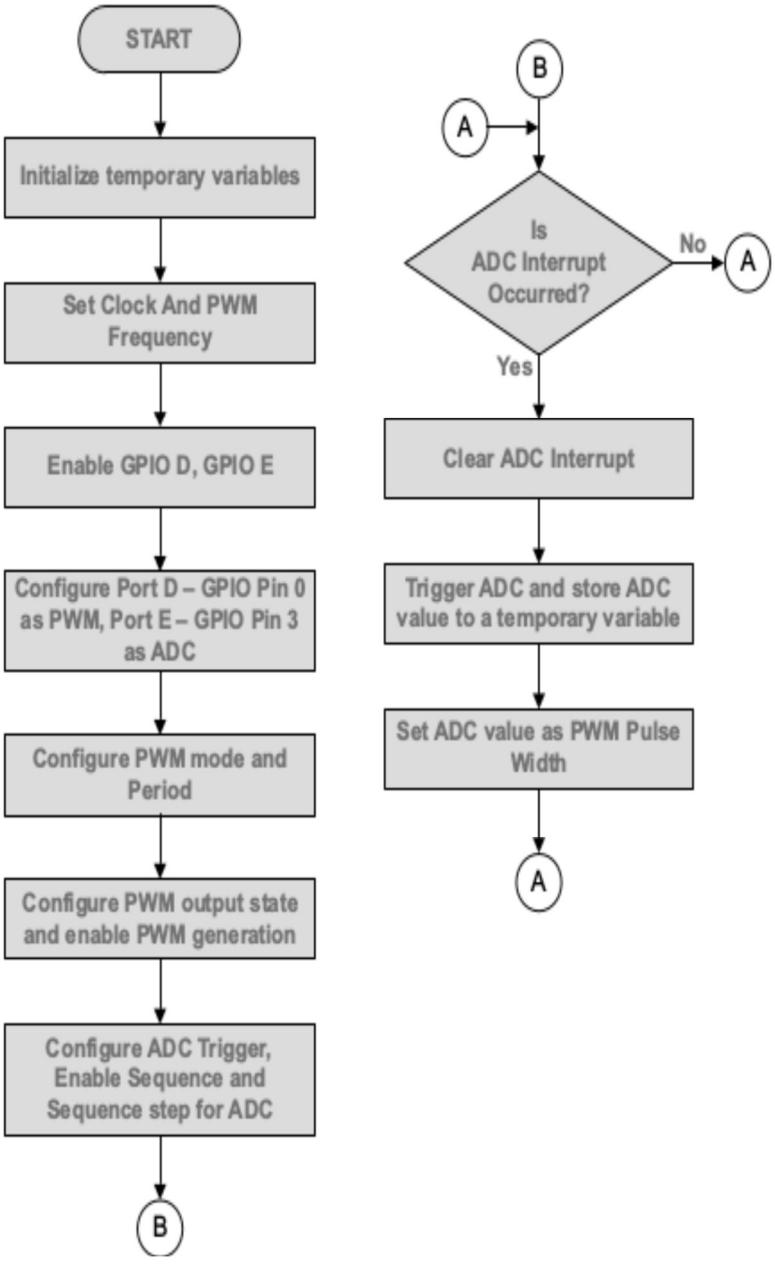
• System Clock = 40MHz
• PWM Clock = System Clock /64 = 625kHz
• PWM Period = 4095
• PWM frequency = PWM Clock / PWM Period = 152.63Hz
- The PWM can be varied as the ADC varies with a 12-bit resolution, and the PWM frequency is 152.63Hz.

PROGRAM :

```
#include<stdint.h>
#include<stdbool.h>
#include"inc/hw_memmap.h"
#include"inc/hw_types.h"
#include"driverlib/debug.h"
#include"driverlib/sysctl.h"
#include"driverlib/adc.h"
#include"inc/hw_types.h"
#include"driverlib/gpio.h"
#include"driverlib/pwm.h"
#include"driverlib/pin_map.h"
#include"inc/hw_gpio.h"
#include"driverlib/rom.h"
uint32_tui32ADC0Value[1];
// TO STORE THE VALUE IN VARIABLE ui32ADC0Value FOR EVERY SAMPLING
intmain(void)
```

```
{
SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|
SYSCTL_XTAL_16MHZ); // SET SYSTEM CLOCK AT 40MHZ
SysCtlPWMClockSet(SYSCTL_PWMDIV_64); //SET PWM CLOCK AT SYSTEM CLOCK
DIVIDED BY 64
SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1); //ENABLE PWM1 MODULE
SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0); / /ENABLE ADC0 MODULE
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
MODULE //ENABLE GPIO FOR ADC0
GPIOPinTypeADC(GPIO_PORTE_BASE,GPIO_PIN_3); //CONFIGURE PE3 AS AN0
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
MODULE //ENABLE GPIO FOR PWM1
GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);
OUTPUT //CONFIGURE PD0 AS PWM
GPIOPinConfigure(GPIO_PD0_M1PWM0); //SET PD0 AS M1PWM0
PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN);
//SET PWM GENERATOR WITH MODEOF OPERATION AS COUNTING
PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0,4095);
//SET THE PERIOD OF PWM GENERATOR
PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true); //ENABLE BIT0 OUTPUT
PWMGenEnable(PWM1_BASE, PWM_GEN_0); //ENABLE PWM GENERATOR
ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
// ADC0 MODULE, TRIGGER IS PROCESSOR EVENT, SEQUENCER 0 IS
CONFIGURED
ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_CH0);
// ADC0 MODULE, SEQUENCER 0 , FOR 1 SAMPLING, INPUT IS FROM CHANNEL 0
PE3
ADCSequenceEnable(ADC0_BASE, 1);
// ENABLE THE SEQUENCE 1 FOR ADC0
while(1)
{
ADCIntClear(ADC0_BASE, 1); // CLEAR INTERRUPT FLAG FOR ADC0, SEQUENCER1

ADCProcessorTrigger(ADC0_BASE, 1);
SEQUENCER1 // TRIGGER IS GIVEN FOR ADC0 MODULE,
// STORE THE CONVERTED VALUE FOR ALL DIFFERENT SAMPLING IN ARRAY
ui32ADC0Value
ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, ui32ADC0Value[0]);
// SET THE PULSE WIDTH
} }
```

FLOW CHART:



RESULT:

VIVA:

1. What is PWM (Pulse Width Modulation)?
2. How many PWM Modules are there in TM4C123GH6PM controller?
3. What are the two clock source options of PWM?
4. What are the two modes of PWM GENERATOR?
5. What is the difference between Count-Down mode and Count-Up/Down mode.?

EXPERIMENT – 12

SENSORHUB BOOSTER PACK WITH TM4C123GXL

AIM:

To Learn and understand interfacing of accelerometer in Sensor Hub Booster pack with TM4C123GXL Launch pad Using I2C.

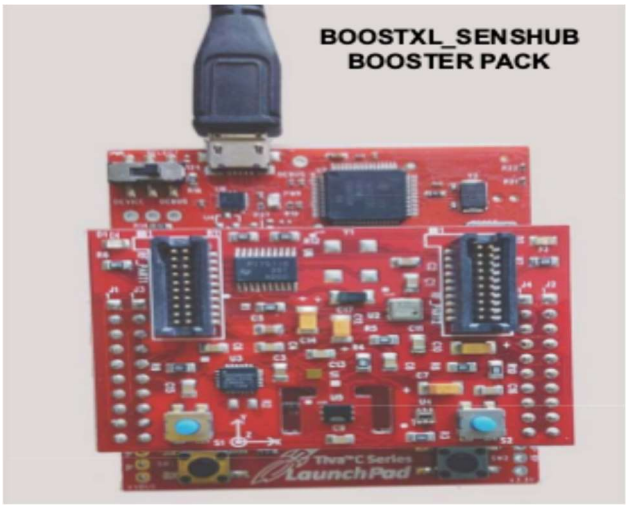
APPARATUS:

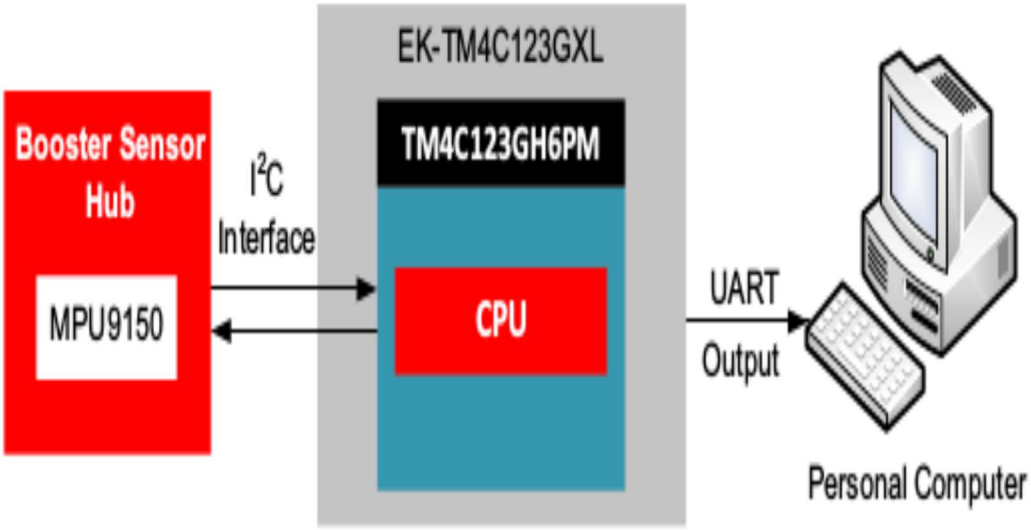
1. Software Required: Code Composer Studio (CCS)
2. Hardware Required: TIVA TM4C123GXL Launch Pad

PROCEDURE:

1. Install TivaWare.
2. Open CCS and create a new workspace.
3. Choose the Import Project link on the TI Resource Explorer page.
4. Import compdcm_mpu9150 project from TivaWare using the following steps:
    a. Choose the Import Project link on the TI Resource Explorer page
    b. Select the Browse button in the Import CCS Eclipse Projects dialog box
    c. Select the compdcm_mpu9150 directory within **C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\ek-tm4c123gxl-boostx-SENSHUB\compdcm_mpu9150.**
5. Open compdcm_mpu9150.c and comment the portion from the Line No. 659 to 690.
    The printing of all these values are commented and hence prevented to be sent to the serial terminal.
6. Add the 3 lines of code given below after the commented portion. This is the code to print the x, y and z values in the serial terminal. The code is now modified to print only the 3 axis values as x, y and z.
7. Save, Build, Debug the Project and Run.
8. Configure Tera Term serial terminal for Baud rate 115200, Data bits 8, Stop Bit 1 and No Parity.
9. Build, program and debug the code into the LaunchPad using CCS.
10. The accelerometer sensor measurements are printed to the terminal Window
11. The RGB LED begins to blink at 1Hz after initialization is completed.
12. Tilt the LaunchPad with BoosterPack in different axes and observe the change in the x, y and z in the terminal software.

BLOCK DIAGRAM:

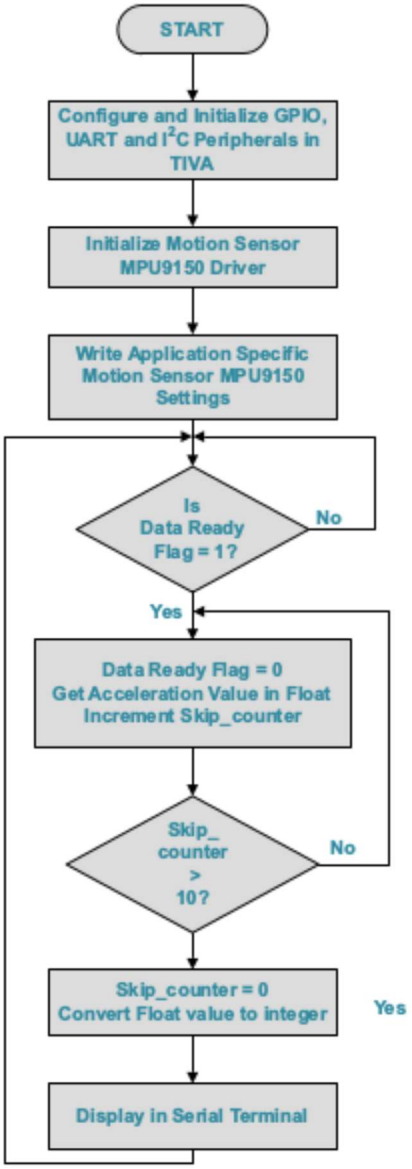




PROGRAM:

```
// Print the x,y,z measured in a table format.
UARTprintf("x=%3d.%03d\n", i32IPart[0], i32FPart[0]);
UARTprintf("y=%3d.%03d\n", i32IPart[1], i32FPart[1]);
UARTprintf("z=%3d.%03d\n", i32IPart[2], i32FPart[2]);
```

FLOW CHART:



RESULT:

VIVA:

1. What are the features of BOOSTXL-SENSHUB Booster Pack?
2. What is accelerometer sensor?
3. What is gyroscope sensor?
4. What is pressure and temperature sensor?
5. What is application of this experiment?

EXPERIMENT – 1
JK-FLIPFLOP USING VHDL

AIM: To design and simulate **JK Flip-Flop** using VHDL

SOFTWARE REQIURED:

1. Personal computer
2. ISE Xilinx software

HARDWARE REQUIRED:

1. SPARTAN – 3E KIT

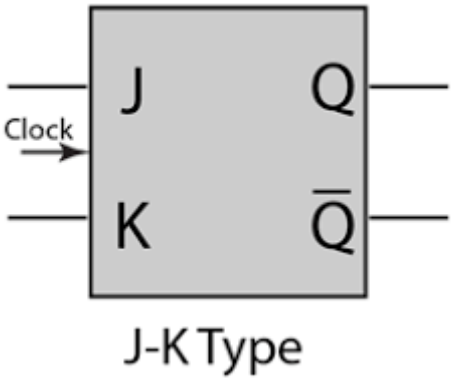
THEORY:

This simple **JK flip Flop** is the most widely used of all the flip-flop designs and is considered to be a universal flip-flop circuit. The two inputs labelled "J" and "K" are not shortened abbreviated letters of other words, such as "S" for Set and "R" for Reset, but are themselves autonomous letters chosen by its inventor Jack Kilby to distinguish the flip-flop design from other types.

The sequential operation of the JK flip flop is exactly the same as for the previous SR flip-flop with the same "Set" and "Reset" inputs. The difference this time is that the "JK flip flop" has no invalid or forbidden input states of the SR Latch even when S and R are both at logic "1".

The **JK flip flop** is basically a gated SR flip-flop with the addition of a clock input circuitry that prevents the illegal or invalid output condition that can occur when both inputs S and R are equal to logic level "1". Due to this additional clocked input, a JK flip-flop has four possible input combinations, "logic 1", "logic 0", "no change" and "toggle". The symbol for a JK flip flop is similar to that of an *SR Bistable Latch* as seen in the previous tutorial except for the addition of a clock input.

CIRCUIT DIAGRAM:



J-K Type

TRUTH TABLE:

Input		Present Output	Next Output
J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

PROGRAM IN VHDL:

BEHAVIORAL:

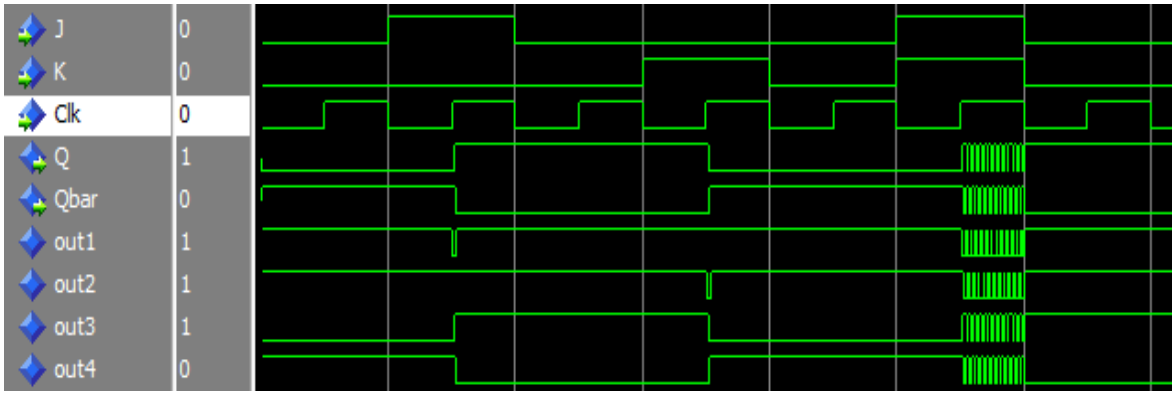
```
library ieee;
use ieee. std_logic_1164.all;
use ieee. std_logic_arith.all;
use ieee. std_logic_unsigned.all;
        entity JK_FF is
                PORT (J, K, CLOCK: in std_logic;
                                Q, QB: out std_logic);
        nd JK_FF;
        Architecture behavioral of JK_FF is
        begin
                PROCESS (CLOCK)
                variable TMP: std_logic;
        begin
                if (CLOCK='1' and CLOCK'EVENT) then
                        if (J='0' and K='0')then TMP:=TMP;
                        elsif (J='1' and K='1')then      TMP:= not TMP;
                        elsif (J='0' and K='1')then      TMP:='0';
                        else     TMP:='1';
                        end if;
                end if;
        Q<=TMP;
        QB <=not TMP;
                end PROCESS;
        end behavioral;
```

VHDL TEST BENCH:

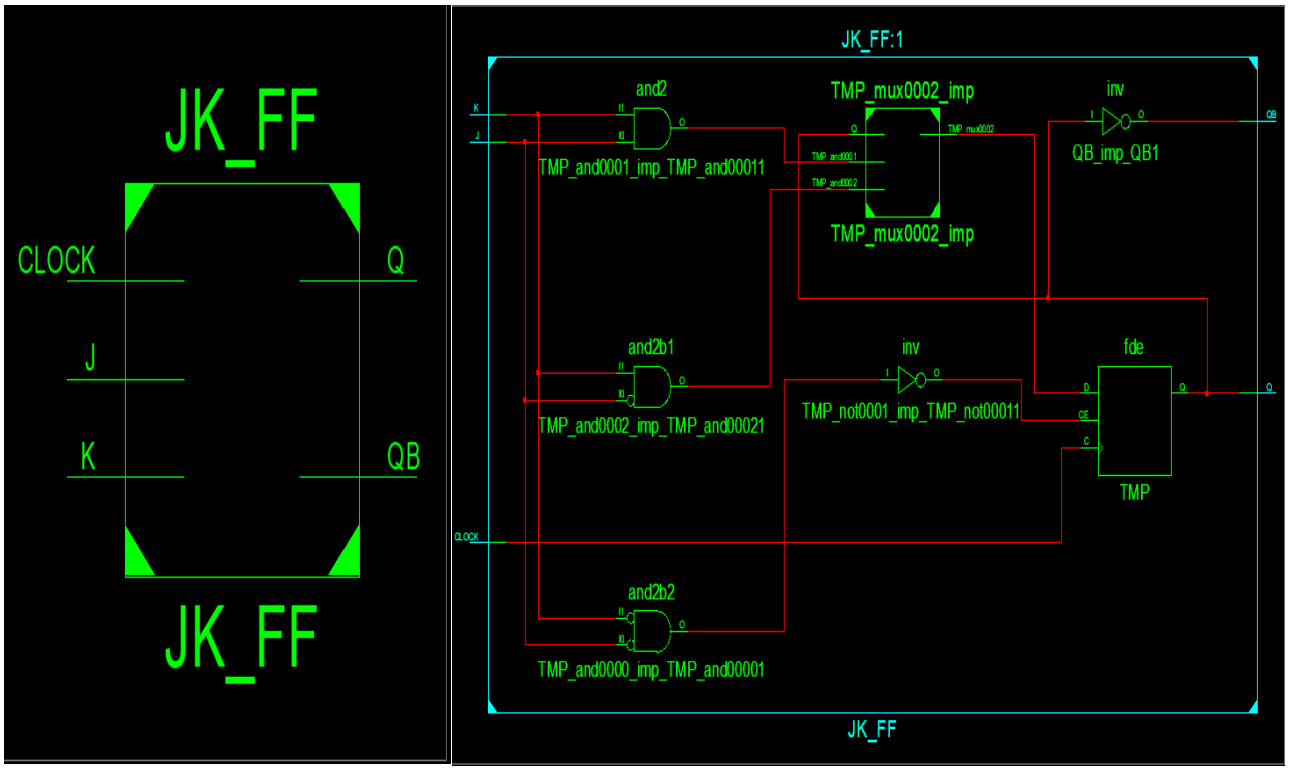
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
 ENTITY JK_FF_TB IS
END JK_FF_TB;
 ARCHITECTURE behavior OF JK_FF_TB IS
    COMPONENT JK_FF
```

```
    PORT(
         J : IN  std_logic;
         K : IN  std_logic;
         CLOCK : IN  std_logic;
         Q : OUT  std_logic;
         QB : OUT  std_logic
        );
    END COMPONENT;
   signal J : std_logic := '0';
   signal K : std_logic := '0';
   signal CLOCK : std_logic := '0';
          signal Q : std_logic;
   signal QB : std_logic;
BEGIN
    uut: JK_FF PORT MAP (
         J => J,
         K => K,
         CLOCK => CLOCK,
         Q => Q,
         QB => QB
        );
   stim_proc: process
   begin
      J<='0'; K<='1'; CLOCK<='1';WAIT FOR 100 NS;
                    J<='0'; K<='1'; CLOCK<='0'; WAIT FOR 100 NS;
                    J<='1'; K<='0'; CLOCK<='1';WAIT FOR 100 NS;
                    J<='1'; K<='0'; CLOCK<='0'; WAIT FOR 100 NS;
                    J<='0'; K<='0'; CLOCK<='1';WAIT FOR 100 NS;
                    J<='0'; K<='0'; CLOCK<='0'; WAIT FOR 100 NS;
                    J<='1'; K<='1'; CLOCK<='1'; WAIT FOR 100 NS;
                    J<='1'; K<='1'; CLOCK<='0'; WAIT FOR 100 NS;
    end process;
END;
```

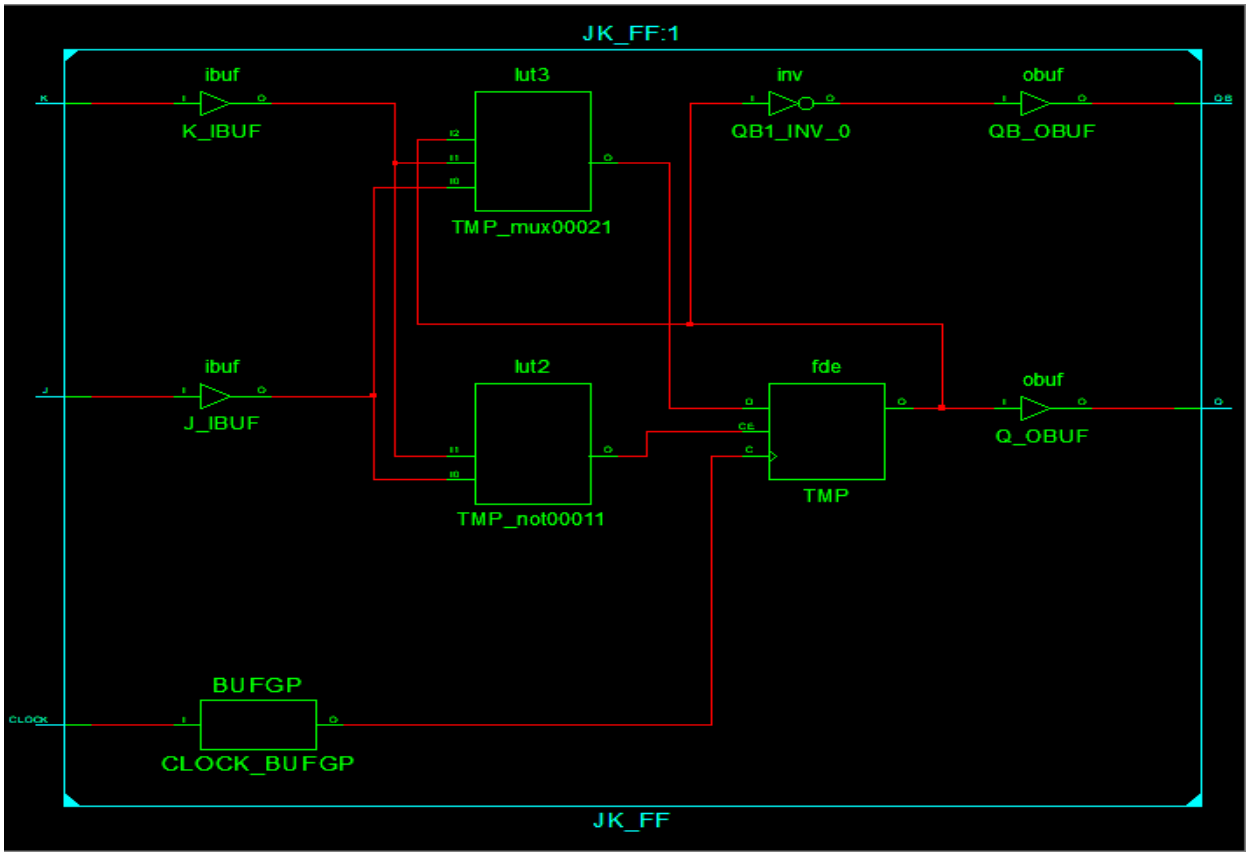
OUTPUT WAVEFORMS:



RTL SCHEMATIC:



INTERNAL DIAGRAM OF D FLIPFLOP:



DEVICE UTILIZATION SUMMARY:

Number of Slices:	2 out of	4656	0%
Number of Slice Flip Flops:	1 out of	9312	0%
Number of 4 input LUTs:	3 out of	9312	0%
Number of IOs:	5		
Number of bonded IOBs:	5 out of	232	2%
Number of GCLKs:	1 out of	24	4%

SYNTHESIS REPORT:

RTL Top Level Output File Name : JK_FF.ngr
Top Level Output File Name : JK_FF
Output Format : NGC
Optimization Goal : Speed
Keep Hierarchy : No
Design Statistics
IOs : 5
Cell Usage:
BELS : 3
INV : 1
LUT2 : 1
LUT3 : 1
Flip-flops/Latches : 1
FDE : 1
Clock Buffers : 1
BUFGP : 1
IO Buffers : 4
IBUF : 2
OBUF : 2

RESULT:

Thus the VHDL code for **JK Flip-Flop is** verified, synthesis report is generated and the design is implemented using FPGA.

VIVA QUESTIONS:

1. What is the use of JK flip flop?
2. What is the significance of the J and K terminals on the JK flip flop?
3. What does the triangle on the clock input of a JK flip flop mean?
4. What is toggle State in JK flip flop?
5. What are the advantages and disadvantages of JK flip flop?

EXPERIMENT – 2
ECHO OF THE DATA INPUT BACK TO THE PC USING UART

AIM:

To connect TM4C123GXL Launch pad to the PC Terminal and send an echo of the data input back to the PC Using UART

APPARATUS:

1. Software Required: Code Composer Studio (CCS)
2. Hardware Required: TIVA TM4C123GXL Launch Pad

PROCEDURE:

1. Connect the EK-TM4C123GXL to the PC using the USB cable supplied.
2. Build, program and debug the code.
3. Open Tera Term UART terminal window and configure which is explained.
4. Type characters on the keyboard and observe the terminal window.

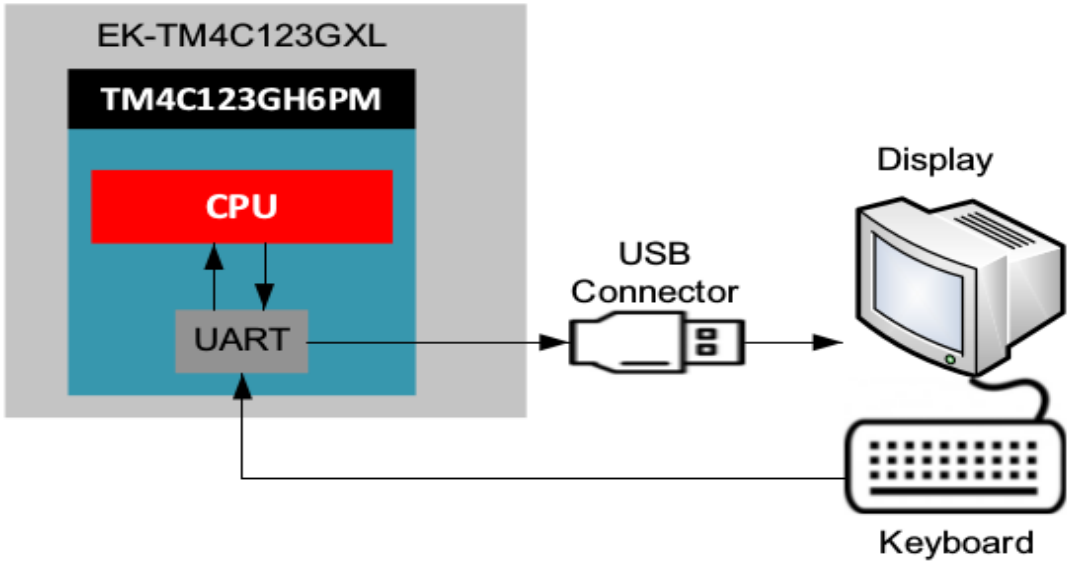
THEORY:

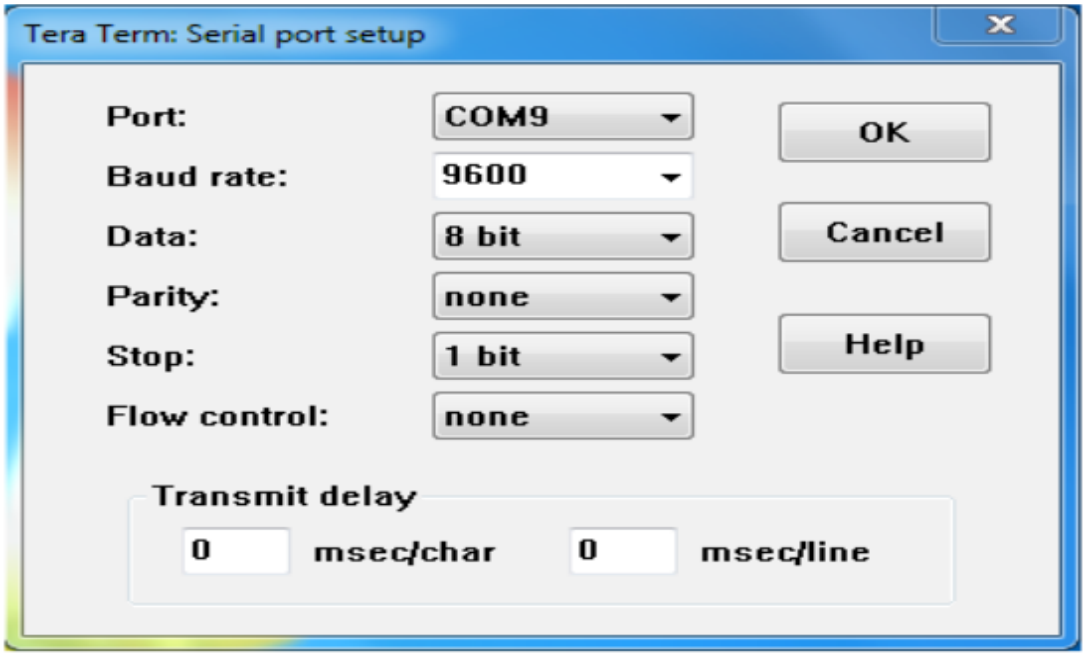
The universal asynchronous receiver-transmitter (UART) takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Serial transmission of digital information (bits) through a single wire or other medium is less costly than parallel transmission through multiple wires.

The UART usually does not directly generate or receive the external signals used between different items of equipment. Separate interface devices are used to convert the logic level signals of the UART to and from the external signalling levels, which may be standardized voltage levels, current levels, or other signals.

Communication may be *simplex* (in one direction only, with no provision for the receiving device to send information back to the transmitting device), *full duplex* (both devices send and receive at the same time) or *half duplex* (devices take turns transmitting and receiving).

BLOCK DIAGRAM:



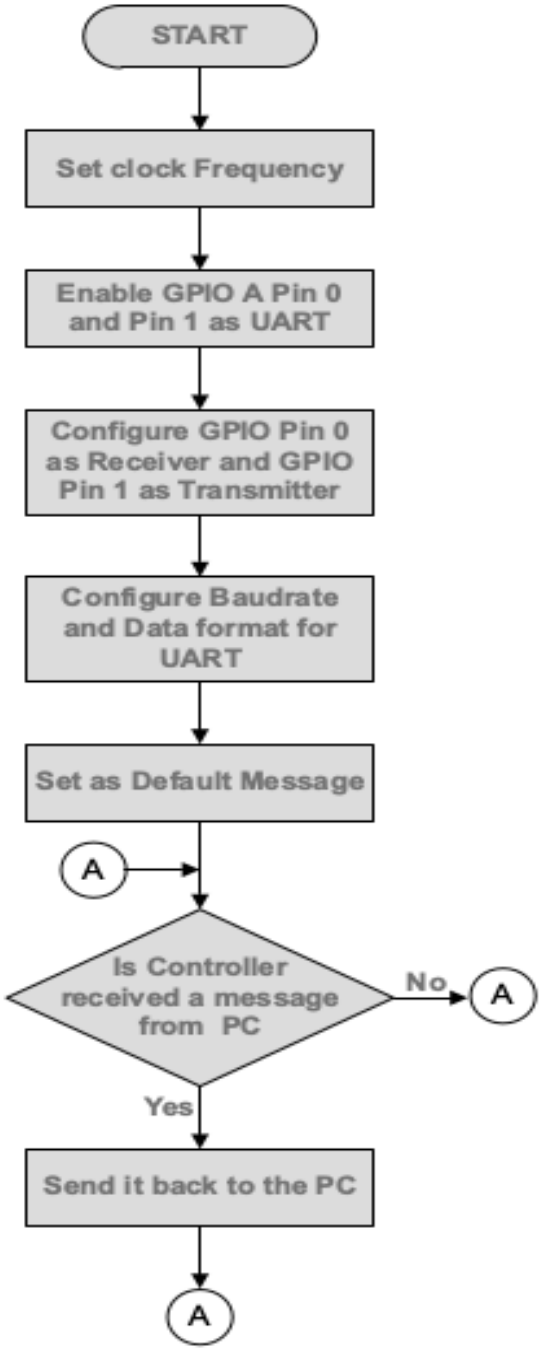


PROGRAM :

```
#include<stdint.h>
#include<stdbool.h>
#include"inc/hw_memmap.h"
#include"inc/hw_types.h"
#include"driverlib/gpio.h"
#include"driverlib/pin_map.h"
#include"driverlib/sysctl.h"
#include"driverlib/uart.h"
#define GPIO_PA0_U0RX 0x00000001 // UART PIN ADDRESS FOR UART RX
#define GPIO_PA1_U0TX 0x00000401 // UART PIN ADDRESS FOR UART TX
{ // SYSTEM CLOCK AT 40 MHZ
SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|
SYSCTL_XTAL_16MHZ); // ENABLE PERIPHERAL UART
0
SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA); // ENABLE GPIO PORT A,FOR
UART
GPIOPinConfigure(GPIO_PA0_U0RX); // PA0 IS CONFIGURED TO UART RX
GPIOPinConfigure(GPIO_PA1_U0TX); // PA1 IS CONFIGURED TO UART TX
GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
// CONFIGURE UART, BAUD RATE 115200, DATA BITS 8, STOP BIT 1, PARITY NONE
UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200,
(UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |
UART_CONFIG_PAR_NONE));
UARTCharPut(UART0_BASE, 'E'); UARTCharPut(UART0_BASE, 'c');
UARTCharPut(UART0_BASE, 'h'); UARTCharPut(UART0_BASE, 'o');
// SEND "Echo Output: " IN UART
```

```
UARTCharPut(UART0_BASE, ' '); UARTCharPut(UART0_BASE, 'O');
UARTCharPut(UART0_BASE, 'u'); UARTCharPut(UART0_BASE, 't');
UARTCharPut(UART0_BASE, 'p'); UARTCharPut(UART0_BASE, 'u');
UARTCharPut(UART0_BASE, 't'); UARTCharPut(UART0_BASE, ': ');
UARTCharPut(UART0_BASE, ' '); UARTCharPut(UART0_BASE, '\n');
while (1)
{ //UART ECHO -
what is received is transmitted back //
if (UARTCharsAvail(UART0_BASE)) UARTCharPut(UART0_BASE,
UARTCharGet(UART0_BASE)); } }
```

FLOW CHART:



RESULT:

VIVA:

1. What is UART?
2. What are different modes for the communication of UART?
3. What is UART baud rate?
4. What is a serial and parallel communication?
5. How may UART modules are there in TM4C123GH6PM controller?