Experiment No. 1

Title : Verification of Truth Tables of Logic Gates

Software Used : Xilinx 14.7

Apparatus Required : FPGA Board

Theory :

Logic gates are the basic building blocks of any digital system. A logic gate is a simple switching circuit that determines whether an input pulse can pass through to the output in digital circuits. A logic gate is a digital gate that allows data to be transferred. Logic gates, use logic to determine whether or not to pass a signal. Logic gates, on the other hand, govern the flow of information based on a set of rules. The following types of logic gates are commonly used –

1) AND
2) OR
3) NOT
4) XOR
5) NAND
6) NOR
7) XNOR

Procedure :

1) Create a new Project, select new source as VHDL module and write the VHDL code for the intended design.
2) Perform Check syntax operation to verify that the code is syntactically correct.
3) View RTL schematic and observe the Block diagram and its detailed connection.
4) Study the synthesis report in detail.
5) Launch ISIM Simulator, force signals to the inputs in signal window, run the simulator and observe the output in Wave window.
6) Write VHDL testbench for the Design-under-test (DUT) and simulate it to observe the behaviour of design.
7) Select the target FPGA device.
8) For the VHDL file, create Implementation constraint file.
9) In UCF file, assign package pins to the input and output ports of design object list as per the selected target device. (The pin numbers can be referred from the Matrix II board manual.
10) Implement the design that includes translation, mapping, placement and routing.
11) Select the FPGA startup clock and configuration mode as per the selected target FPGA device (Spartan 6).
12) Add Xilinx device, select the appropriate bit file and program the FPGA.
13) Apply the input through onboard switches or interfaced Add-on Module and observe the output on the intended output device which may be LED/LCD/SSD/Add-on Module.

Module 1 : AND Gate

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    02:59:46 10/11/2023
-- Design Name:    Logic Gates
-- Module Name:    and_gate_48 - Behavioral
-- Project Name:   AND Gate
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

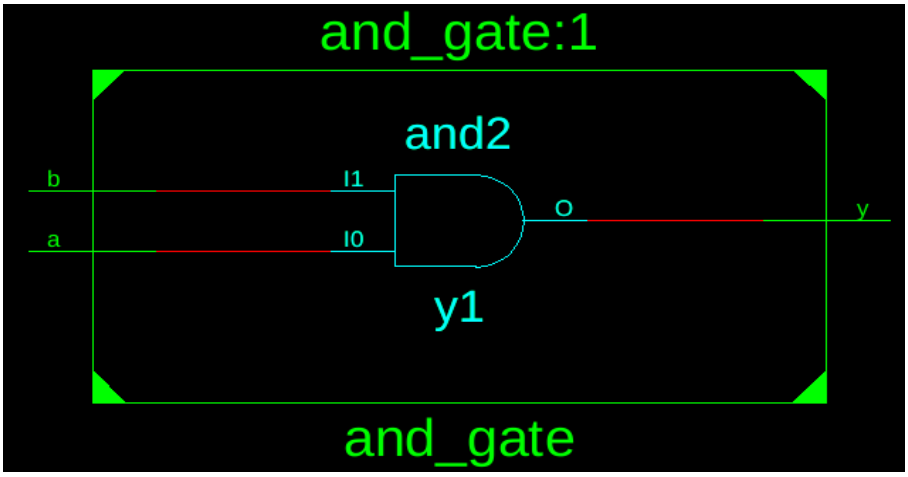
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity and_gate is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          y : out  STD_LOGIC);
end and_gate;
architecture and_gate_arch of and_gate is
begin
      y <= a and b;
end and_gate_arch;
```

Testbench :

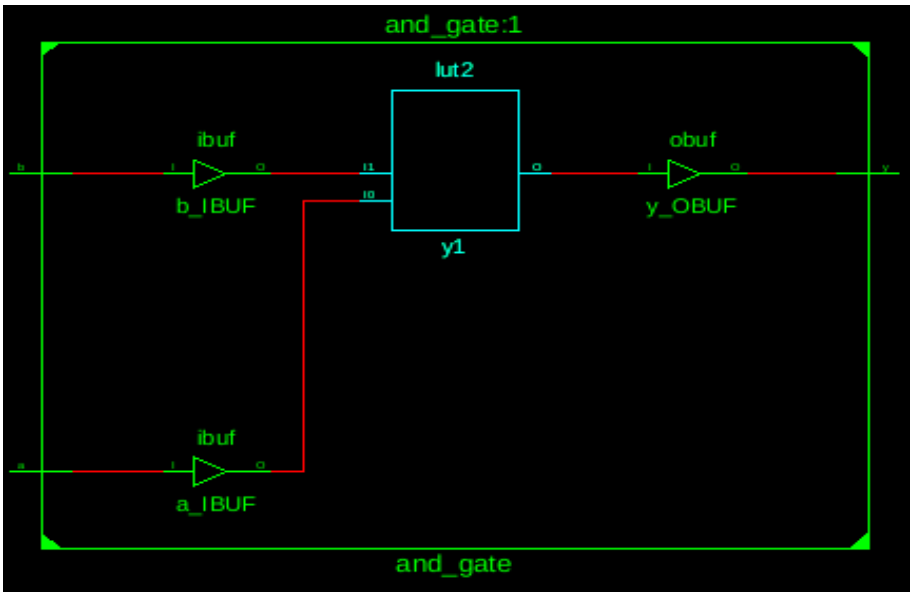
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY and_gate_tb IS
END and_gate_tb;
ARCHITECTURE and_gate_tb_arch OF and_gate_tb IS
COMPONENT and_gate
   PORT(  A : IN  std_logic;
          B : IN  std_logic;
          Y : OUT  std_logic
         );
END COMPONENT;
signal A : std_logic := '0';
signal B : std_logic := '0';
signal Y : std_logic;
BEGIN
uut : and_gate PORT MAP (
    A => A,
    B => B,
    Y => Y
  );
Stim_proc : process
```

```
begin
wait for 100 ns;
    A <= '0';
    B <= '0';
    wait for 100 ns;
    A <= '0';
    B <= '1';
    wait for 100 ns;
    A <= '1';
    B <= '0';
    wait for 100 ns;
    A <= '1';
    B <= '1';
    wait for 100 ns;
end process;
END;
```

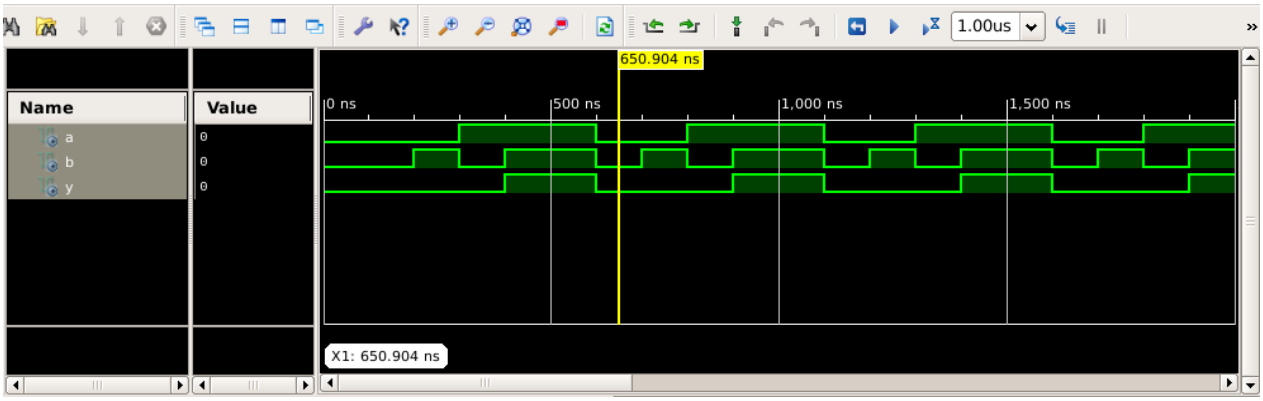
RTL Schematic :

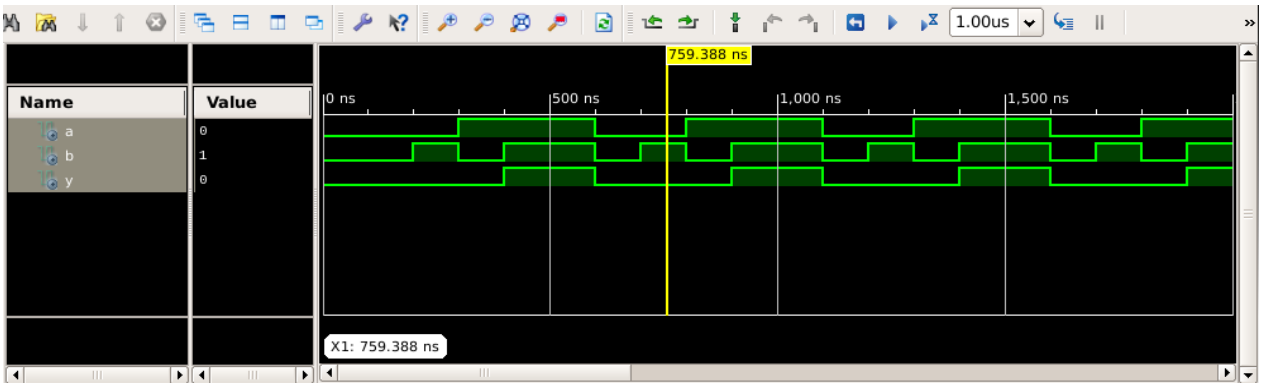


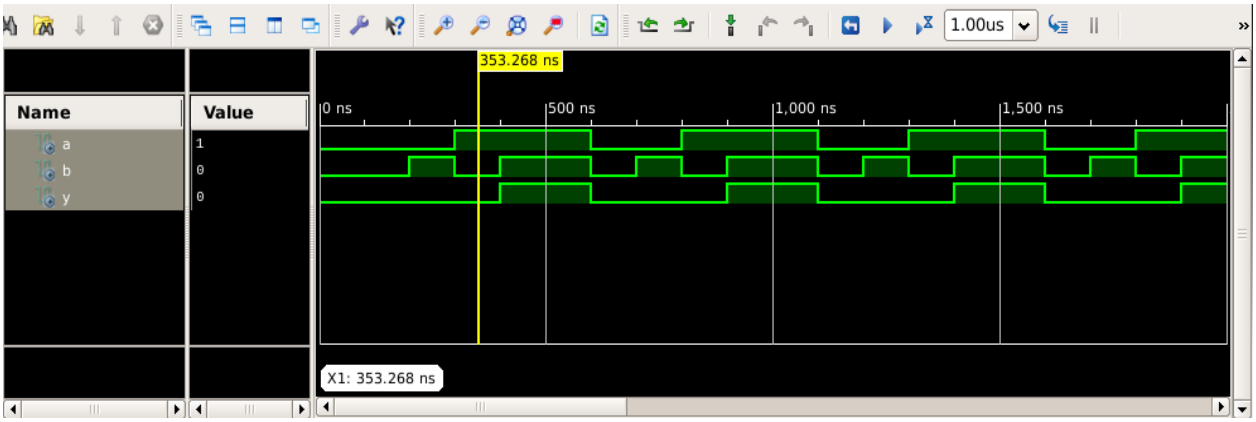
Technology Schematic :

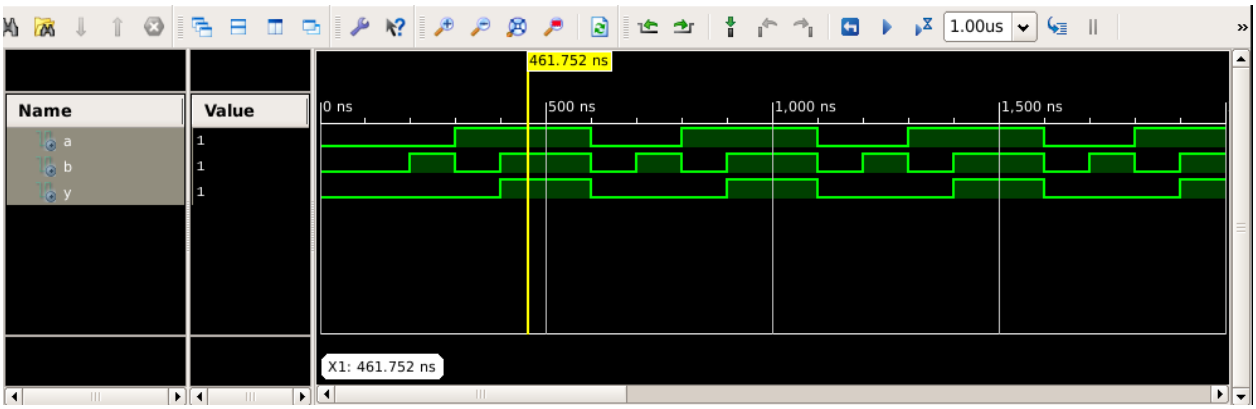


Waveform :


1.00us
650.904 ns
Name
Value
a
b
y
0
0
0
0 ns
500 ns
1,000 ns
1,500 ns
X1: 650.904 ns


1.00us
759.388 ns
Name
Value
a
b
y
0
1
0
0 ns
500 ns
1,000 ns
1,500 ns
X1: 759.388 ns


1.00us
353.268 ns
Name
Value
a
b
y
1
0
0
0 ns
500 ns
1,000 ns
1,500 ns
X1: 353.268 ns


1.00us
461.752 ns
Name
Value
a
b
y
1
1
1
0 ns
500 ns
1,000 ns
1,500 ns
X1: 461.752 ns

Design Summary :

```
=========================================================================
*                            Design Summary                             *
=========================================================================

Top Level Output File Name         : and_gate.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 1
#      LUT2                        : 1
# IO Buffers                       : 3
#      IBUF                        : 2
#      OBUF                        : 1
```

Module 2 : OR Gate

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    02:59:46 10/11/2023
-- Design Name:    Logic Gates
-- Module Name:    or_gate_48 - Behavioral
-- Project Name:   OR Gate
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

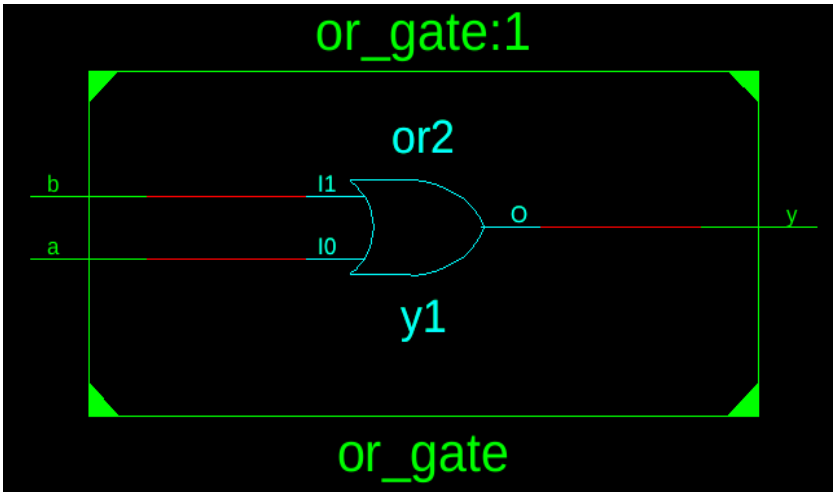
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity or_gate is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          y : out  STD_LOGIC);
end or_gate;
architecture or_gate_arch of or_gate is
begin
      y <= a or b;
end or_gate_arch;
```

Testbench :

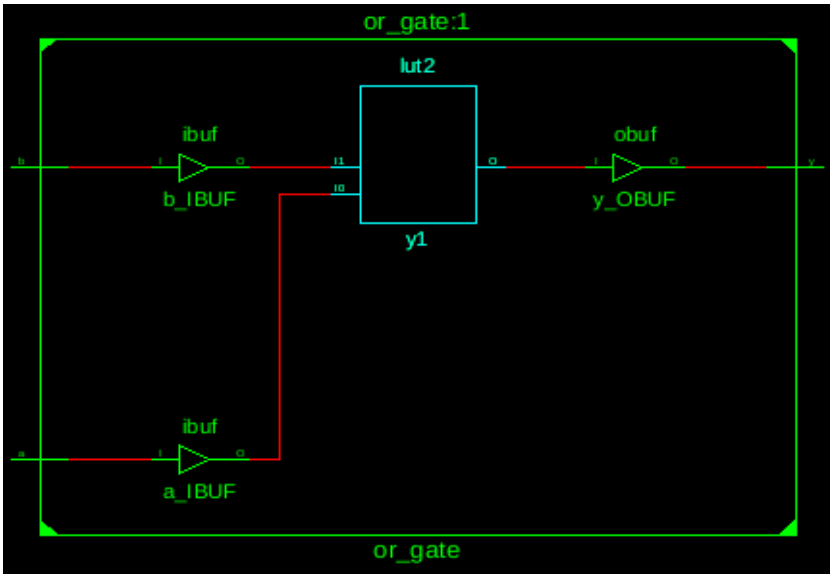
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY or_gate_tb IS
END or_gate_tb;
ARCHITECTURE or_gate_tb_arch OF or_gate_tb IS
COMPONENT or_gate
   PORT(  A : IN  std_logic;
          B : IN  std_logic;
          Y : OUT  std_logic
```

```
        );
END COMPONENT;
signal A : std_logic := '0';
signal B : std_logic := '0';
signal Y : std_logic;
BEGIN
uut : or_gate PORT MAP (
    A => A,
    B => B,
    Y => Y
  );
Stim_proc : process
begin
wait for 100 ns;
    A <= '0';
    B <= '0';
    wait for 100 ns;
    A <= '0';
    B <= '1';
    wait for 100 ns;
    A <= '1';
    B <= '0';
    wait for 100 ns;
    A <= '1';
    B <= '1';
    wait for 100 ns;
end process;
END;
```

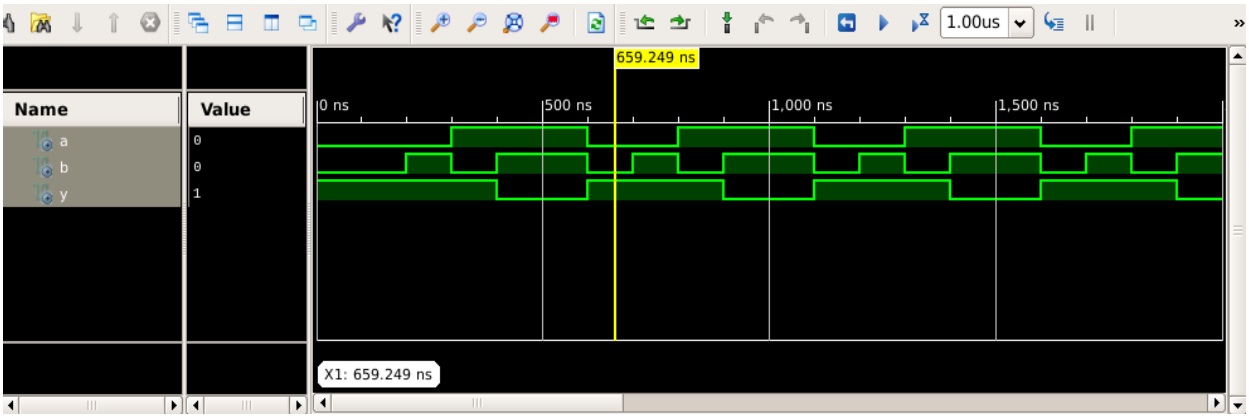
RTL Schematic :

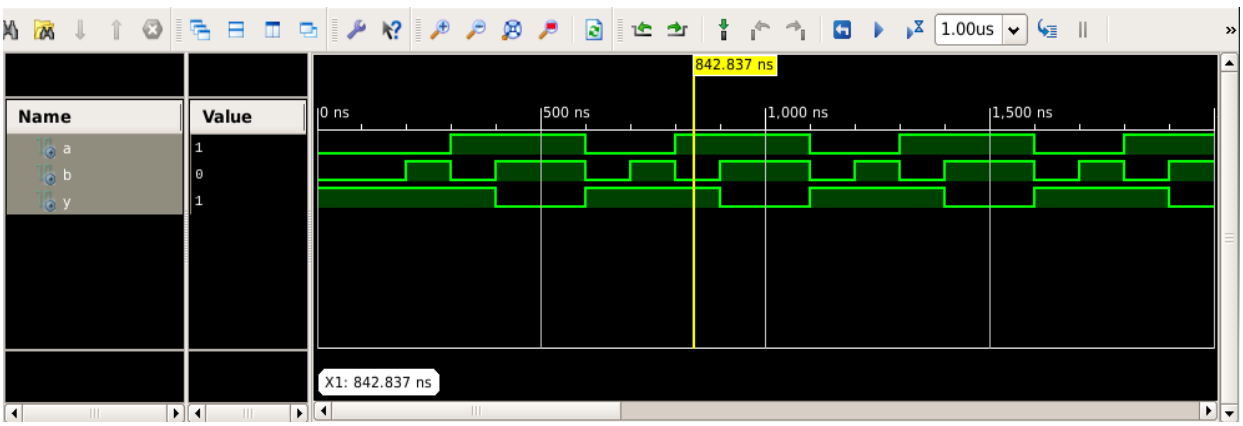


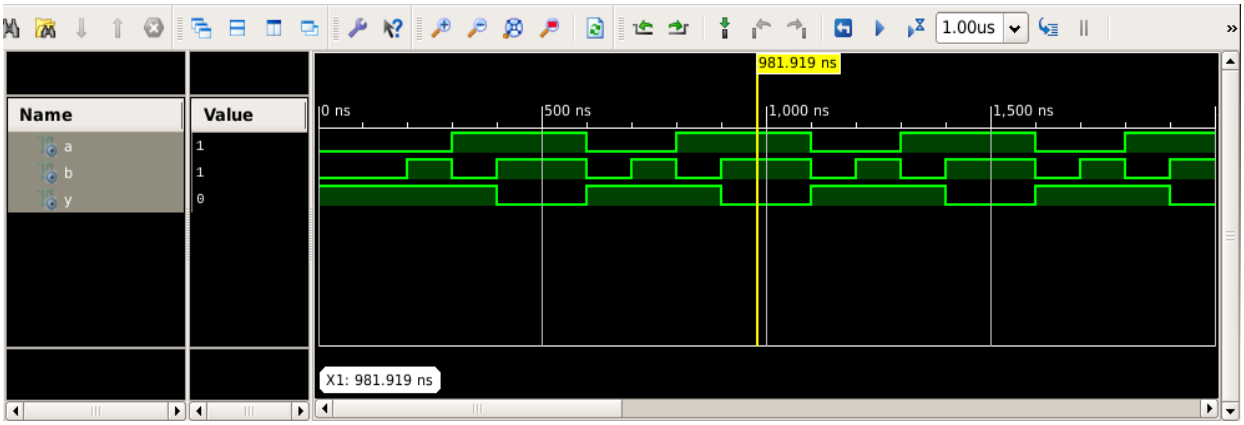
Technology Schematic :



Waveform :







Design Summary :

```
==========================================================================
*                            Design Summary                              *
==========================================================================

Top Level Output File Name         : or_gate.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 1
#      LUT2                        : 1
# IO Buffers                       : 3
#      IBUF                        : 2
#      OBUF                        : 1
```

Module 3 : NOT Gate

```
----------------------------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    02:59:46 10/11/2023
-- Design Name:    Logic Gates
-- Module Name:    not_gate_48 - Behavioral
-- Project Name:   NOT Gate
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------------------------
```

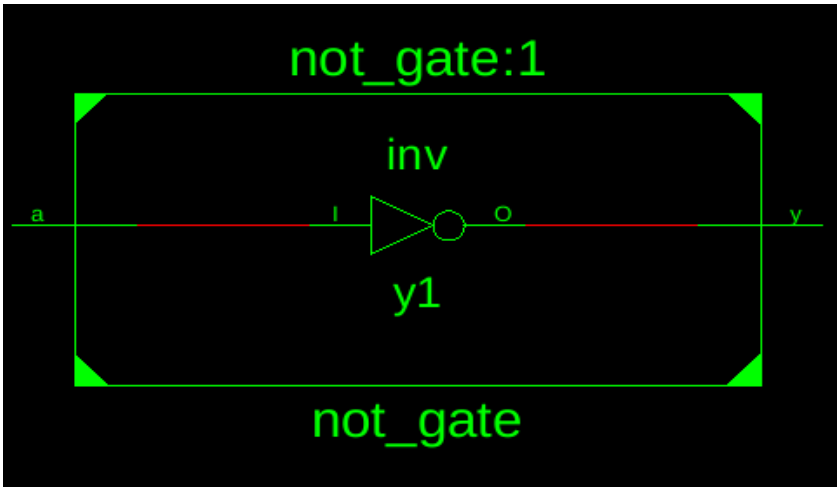
VHDL Program :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity not_gate is
   Port ( a : in  STD_LOGIC;
          y : out  STD_LOGIC);
end not_gate;
architecture not_gate_arch of not_gate is
begin
      y <=  not a;
end not_gate_arch;
```

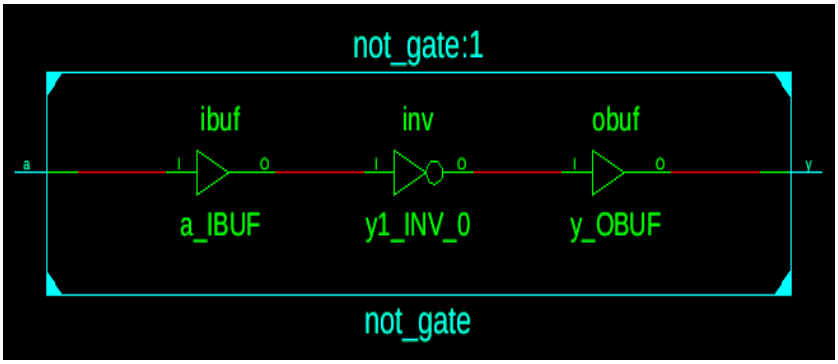
Testbench :

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY not_gate_tb IS
END not_gate_tb;
ARCHITECTURE not_gate_tb_arch OF not_gate_tb IS
COMPONENT not_gate
   PORT( A : IN  std_logic;
         Y : OUT  std_logic
        );
END COMPONENT;
signal A : std_logic := '0';
signal Y : std_logic;
BEGIN
uut: not_gate PORT MAP (
         A => A,
         Y => Y
       );
stim_proc: process
begin
wait for 100 ns;
      A <= '0';
      wait for 100 ns;
      A <= '1';
      wait for 100 ns;
end process;
END;
```

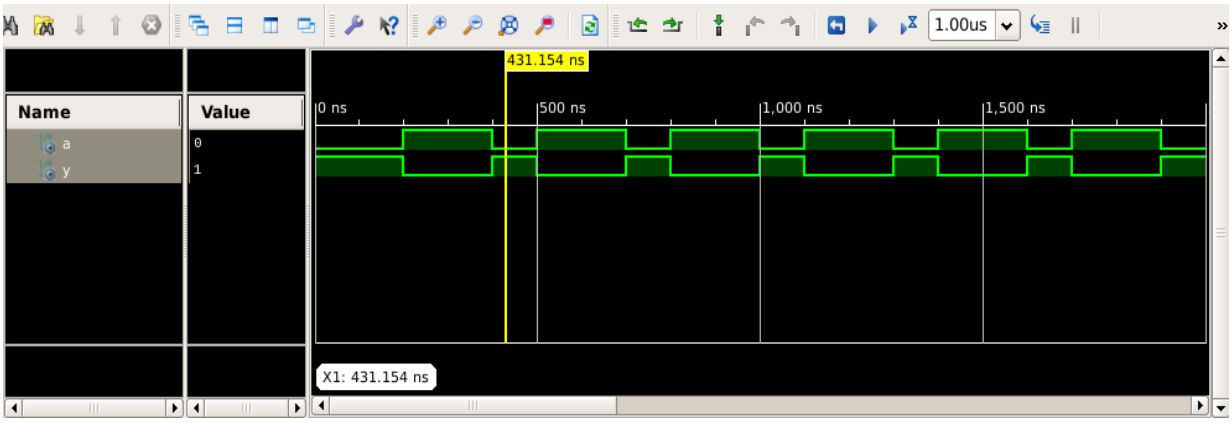
RTL Schematic :

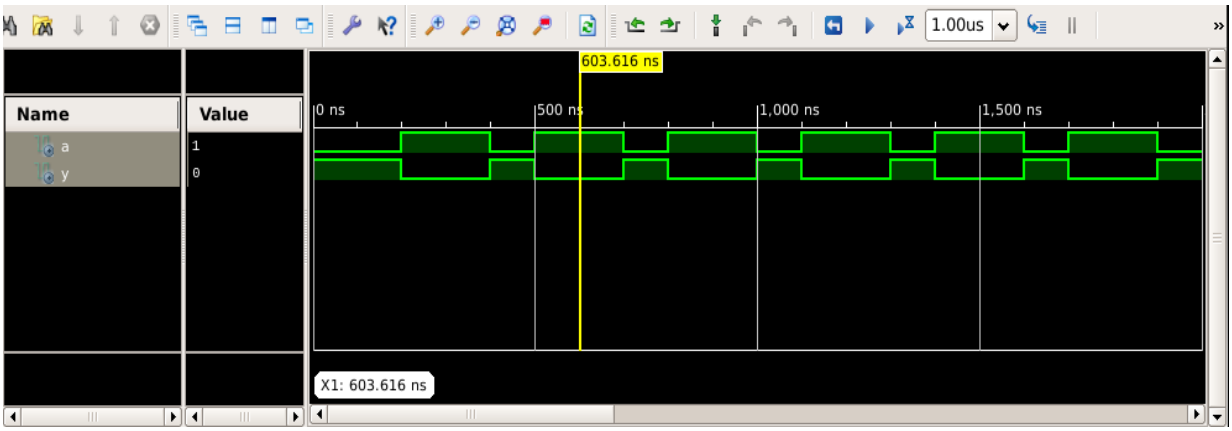


Technology Schematic :



Waveform :





Design Summary :

```
================================================================================
*                            Design Summary                                    *
================================================================================

Top Level Output File Name         : not_gate.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 1
#      INV                          : 1
# IO Buffers                        : 2
#      IBUF                         : 1
#      OBUF                         : 1
```

Module 4 : XOR Gate

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    02:59:46 10/11/2023
-- Design Name:    Logic Gates
-- Module Name:    xor_gate_48 - Behavioral
-- Project Name:   XOR Gate
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

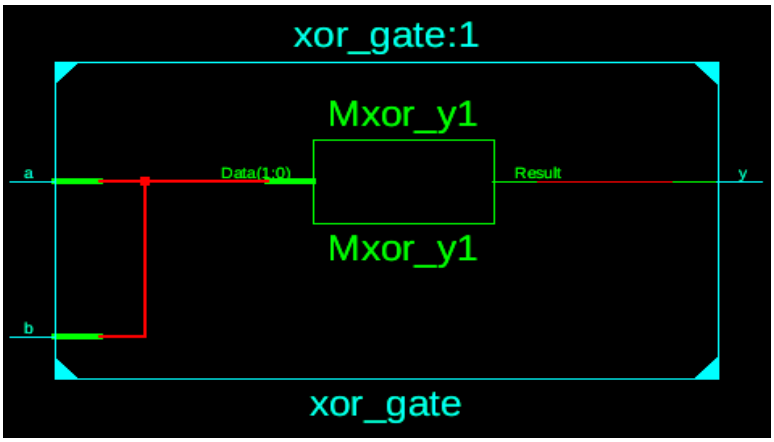
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity xor_gate is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          y : out  STD_LOGIC);
end xor_gate;
architecture xor_gate_arch of xor_gate is
begin
     y <= a xor b;
end  xor_gate_arch;
```

Testbench :

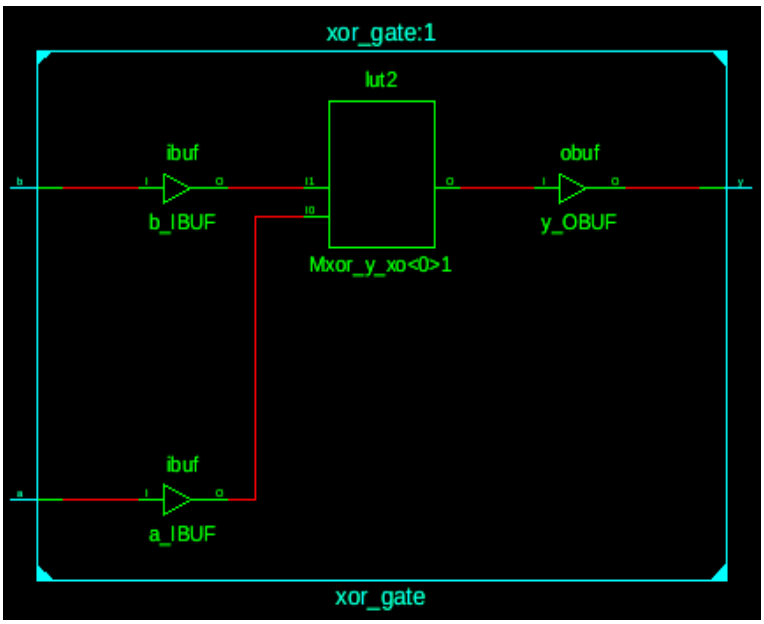
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY xor_gate_tb IS
END xor_gate_tb;
ARCHITECTURE xor_gate_tb_arch OF xor_gate_tb IS
COMPONENT xor_gate
   PORT(  A : IN  std_logic;
          B : IN  std_logic;
          Y : OUT  std_logic
         );
END COMPONENT;
signal A : std_logic := '0';
signal B : std_logic := '0';
signal Y : std_logic;
BEGIN
uut : xor_gate PORT MAP (
    A => A,
    B => B,
    Y => Y
  );
```

```
Stim_proc : process
begin
wait for 100 ns;
    A <= '0';
    B <= '0';
    wait for 100 ns;
    A <= '0';
    B <= '1';
    wait for 100 ns;
    A <= '1';
    B <= '0';
    wait for 100 ns;
    A <= '1';
    B <= '1';
    wait for 100 ns;
end process;
END;
```

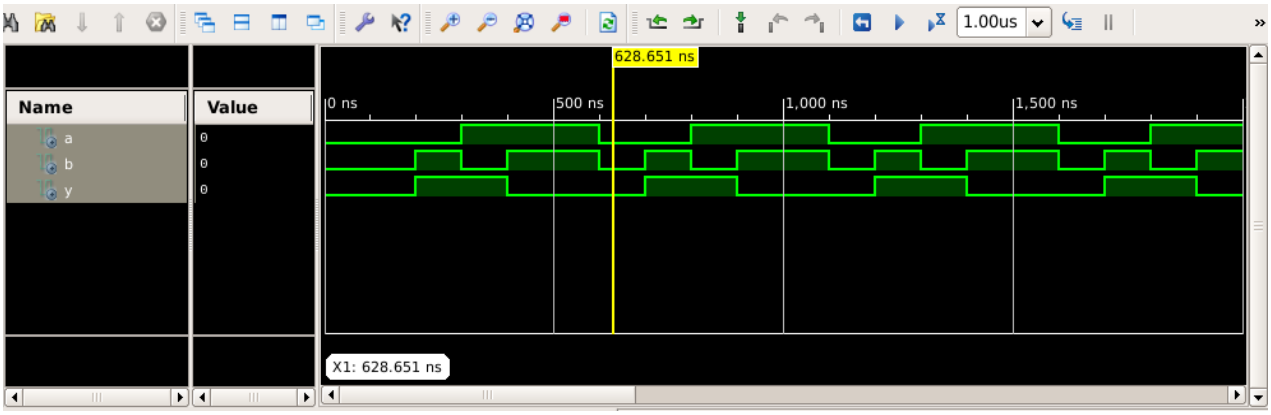
RTL Schematic :

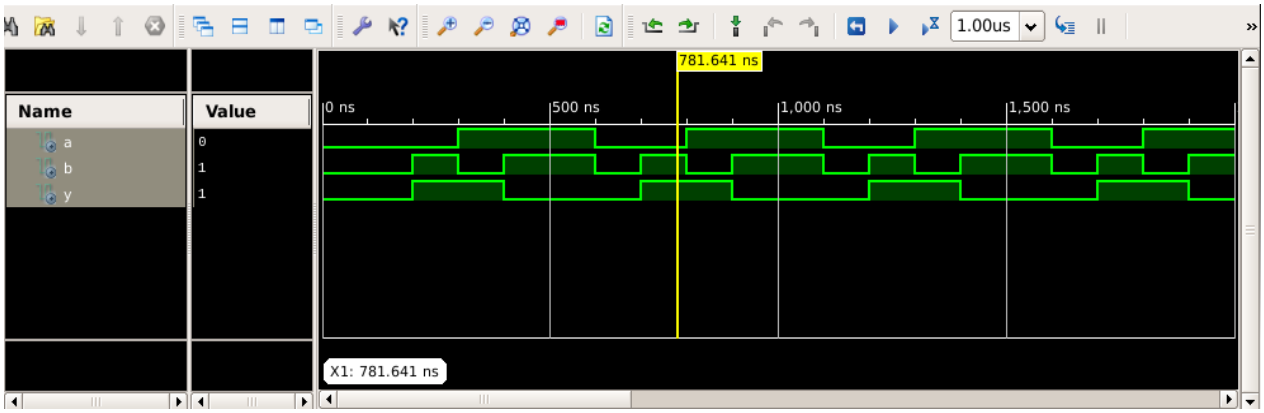


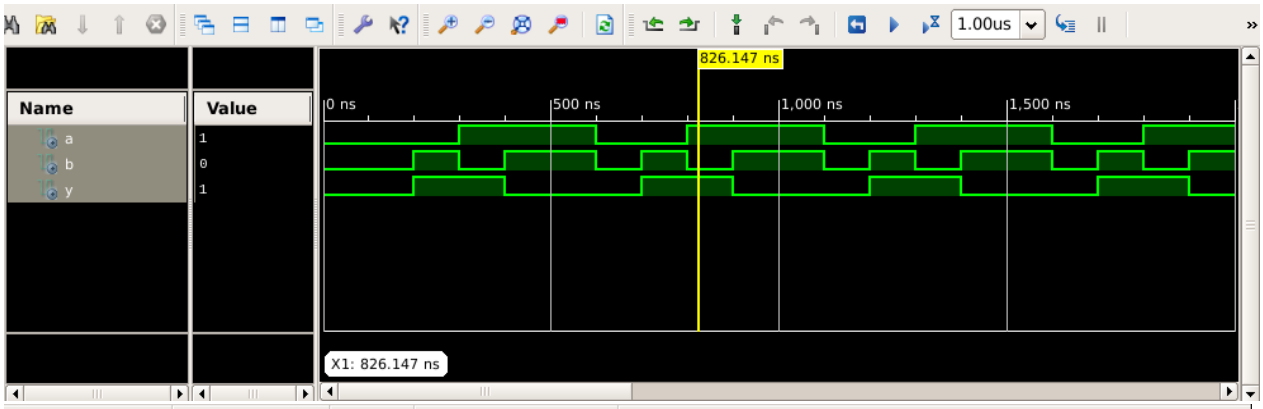
Technology Schematic :

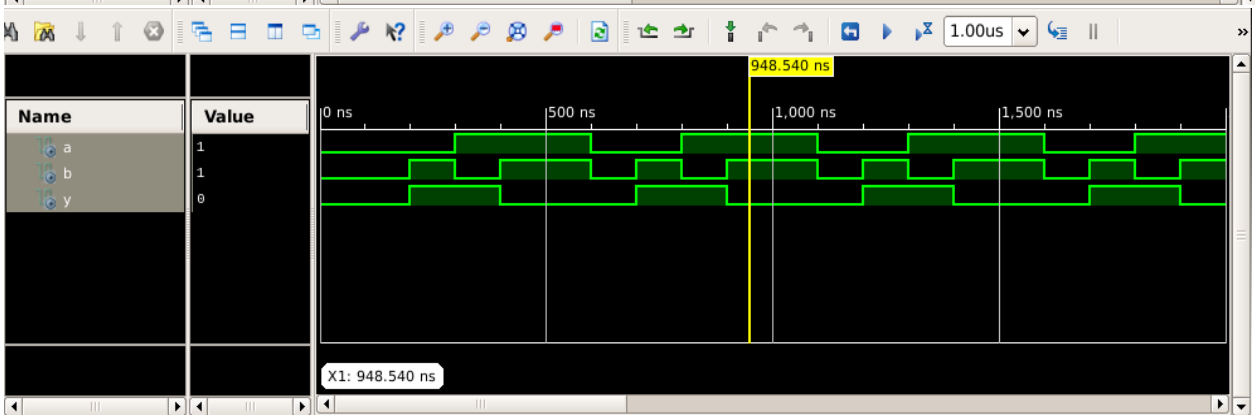


Waveform :


1.00us
628.651 ns
Name
Value
a
b
y
0
0
0
0 ns
500 ns
1,000 ns
1,500 ns
X1: 628.651 ns


1.00us
781.641 ns
Name
Value
a
b
y
0
1
1
0 ns
500 ns
1,000 ns
1,500 ns
X1: 781.641 ns


1.00us
826.147 ns
Name
Value
a
b
y
1
0
1
0 ns
500 ns
1,000 ns
1,500 ns
X1: 826.147 ns


1.00us
948.540 ns
Name
Value
a
b
y
1
1
0
0 ns
500 ns
1,000 ns
1,500 ns
X1: 948.540 ns

Design Summary :

```
=========================================================================
*                            Design Summary                             *
=========================================================================

Top Level Output File Name         : xor_gate.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 1
#      LUT2                        : 1
# IO Buffers                       : 3
#      IBUF                        : 2
#      OBUF                        : 1
```

Module 5 : NAND Gate

```
----------------------------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    02:59:46 10/11/2023
-- Design Name:    Logic Gates
-- Module Name:    nand_gate_48 - Behavioral
-- Project Name:   NAND Gate
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------------------------
```

VHDL Program :

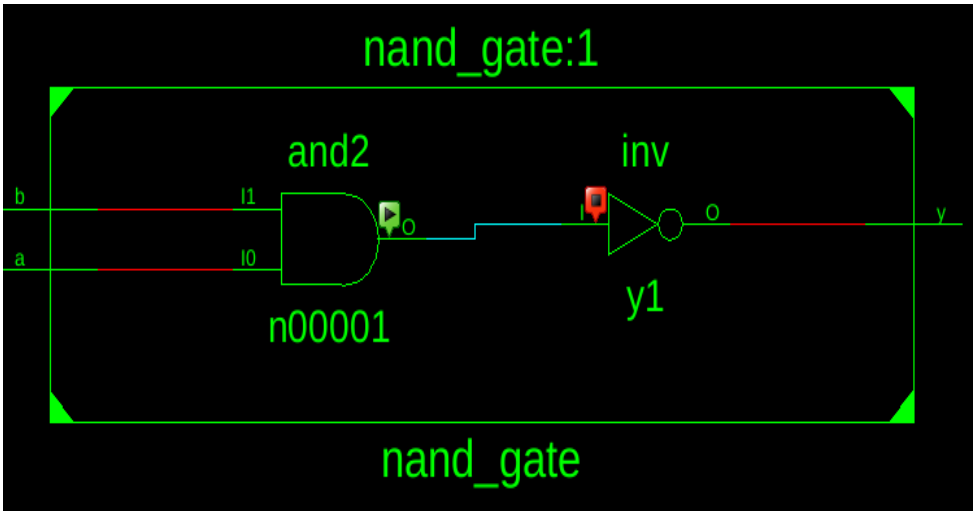
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity nand_gate is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          y : out  STD_LOGIC);
end nand_gate;
architecture nand_gate_arch of nand_gate is
begin
      y <= a nand b;
end nand_gate_arch;
```

Testbench :

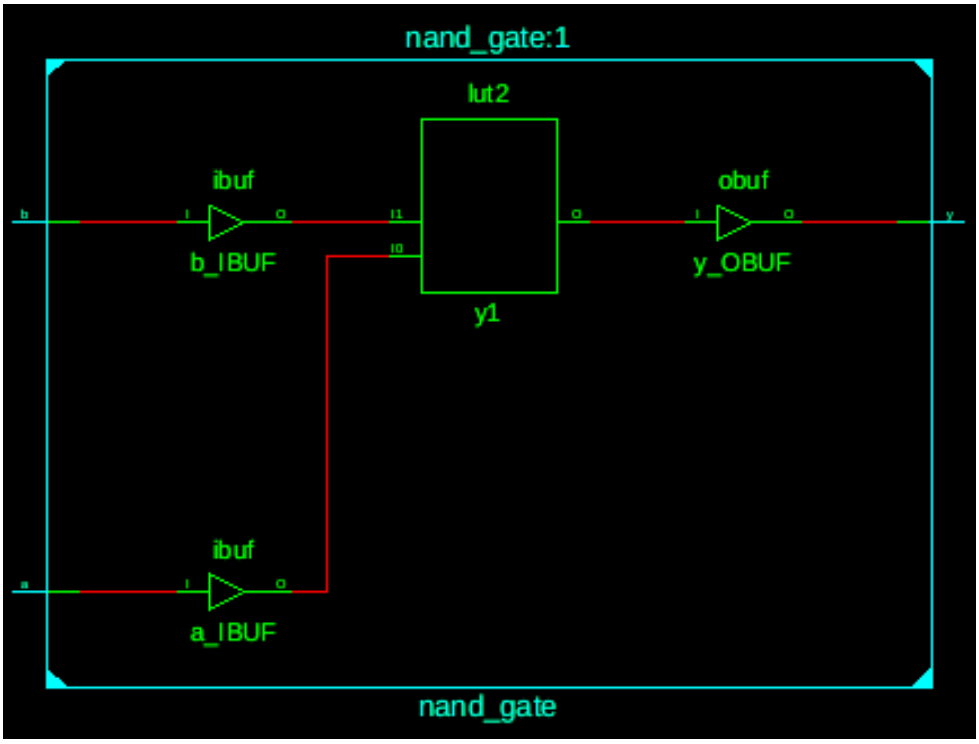
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY nand_gate_tb IS
END nand_gate_tb;
ARCHITECTURE nand_gate_tb_arch OF nand_gate_tb IS
COMPONENT nand_gate
   PORT(  A : IN  std_logic;
          B : IN  std_logic;
          Y : OUT  std_logic
```

```
        );
END COMPONENT;
signal A : std_logic := '0';
signal B : std_logic := '0';
signal Y : std_logic;
BEGIN
uut : nand_gate PORT MAP (
    A => A,
    B => B,
    Y => Y
  );
Stim_proc : process
begin
wait for 100 ns;
    A <= '0';
    B <= '0';
    wait for 100 ns;
    A <= '0';
    B <= '1';
    wait for 100 ns;
    A <= '1';
    B <= '0';
    wait for 100 ns;
    A <= '1';
    B <= '1';
    wait for 100 ns;
end process;
END;
```

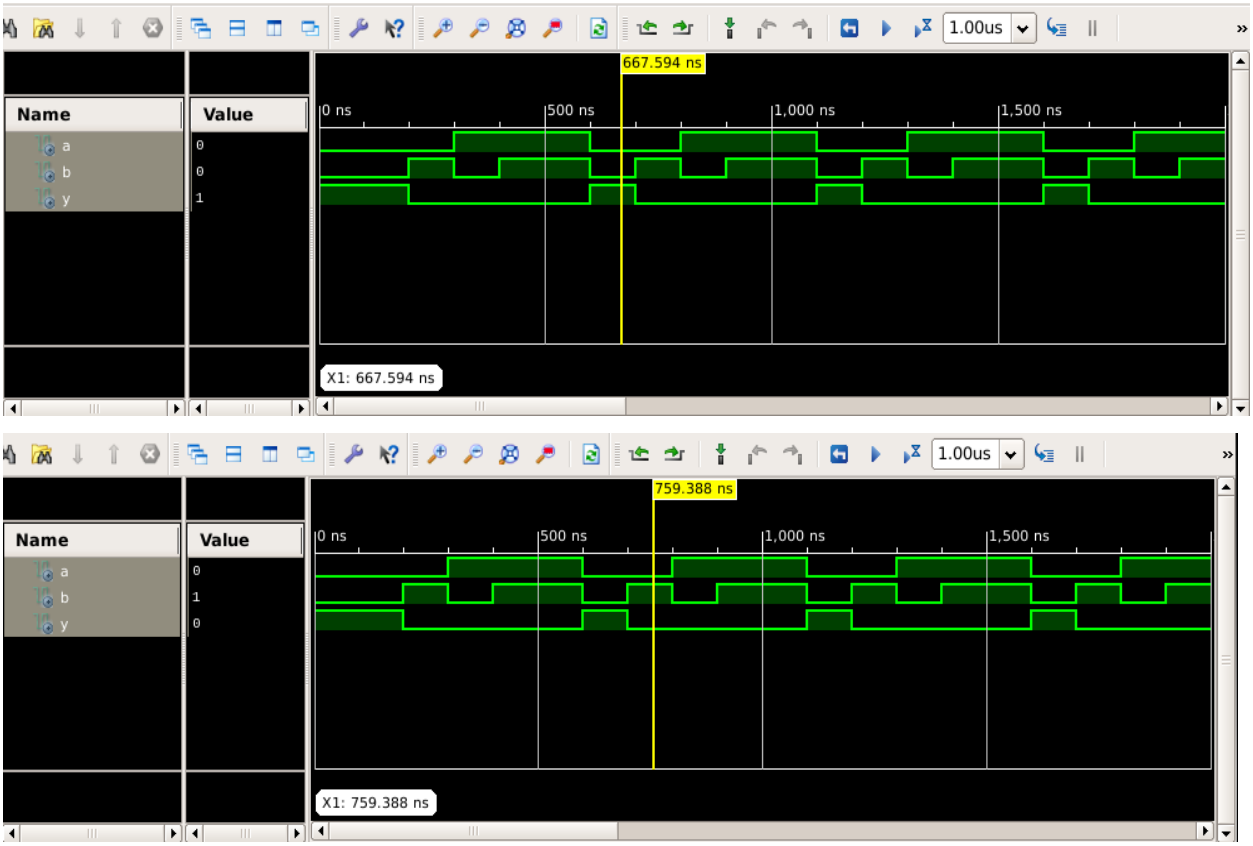
RTL Schematic :

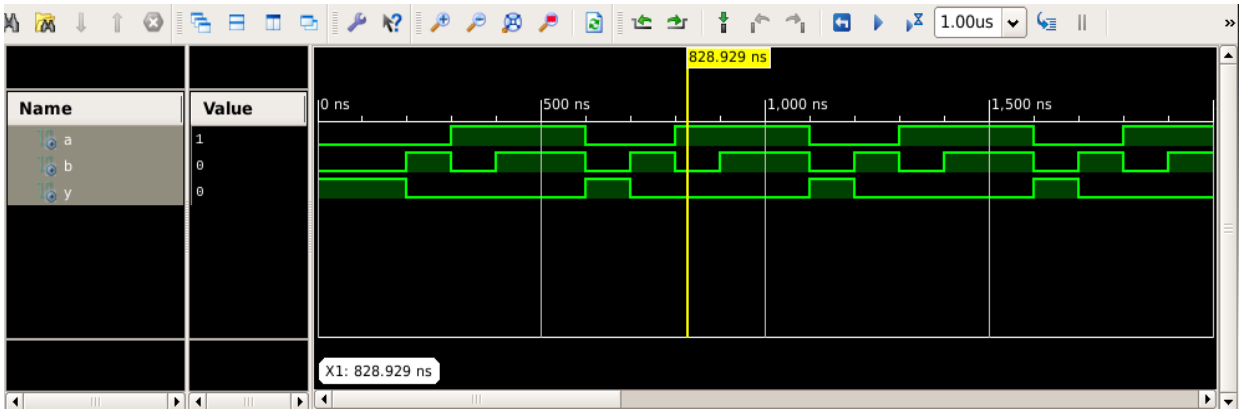


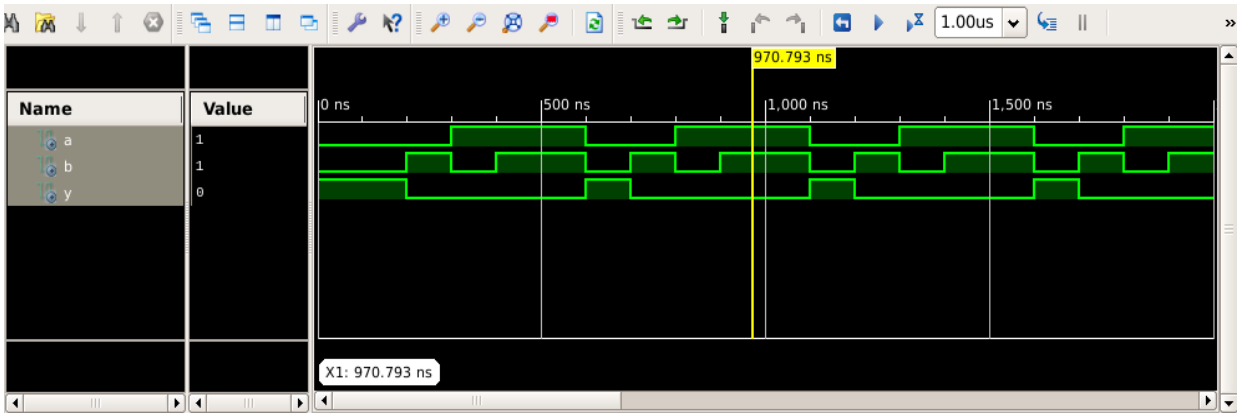
Technology Schematic :



Waveform :







Design Summary :

```
================================================================================
*                              Design Summary                                  *
================================================================================

Top Level Output File Name         : nand_gate.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 1
#      LUT2                        : 1
# IO Buffers                       : 3
#      IBUF                        : 2
#      OBUF                        : 1
```

Module 6 : NOR Gate

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer: Tanmay Vilas Pawar

-- Create Date:    11:27:26 11/04/2023
-- Design Name:    Logic Gates
-- Module Name:    nor_gate_48 - Behavioral
-- Project Name:   NOR Gate
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

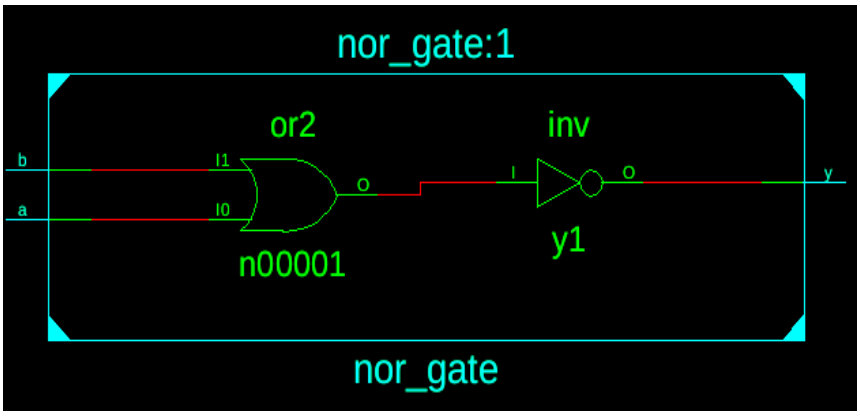
```
entity nor_gate is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          y : out  STD_LOGIC);
end nor_gate;
architecture nor_gate_arch of nor_gate is
begin
      y <= a nor b;
end nor_gate_arch;
```

Testbench :

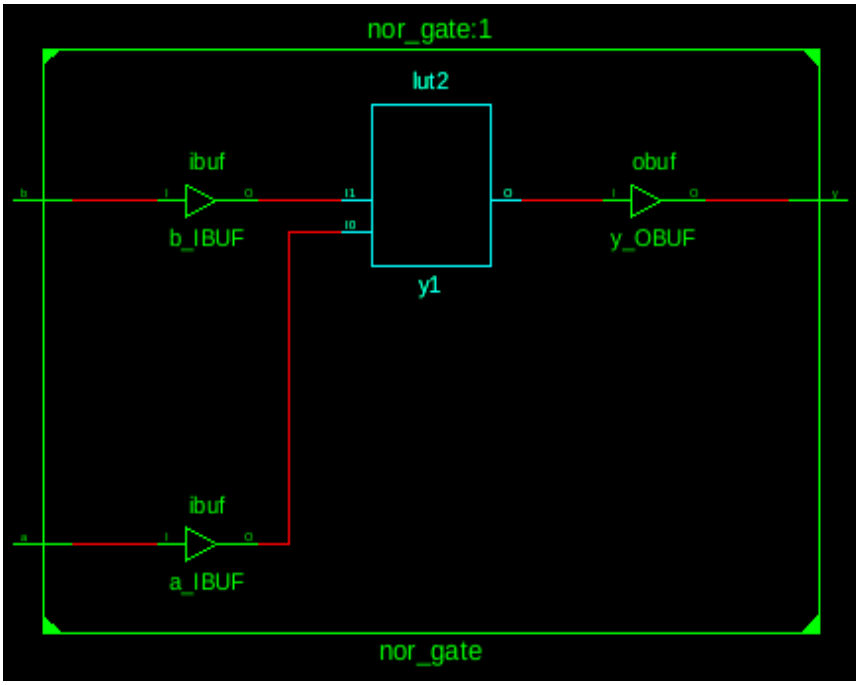
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY nor_gate_tb IS
END nor_gate_tb;
ARCHITECTURE nor_gate_tb_arch OF nor_gate_tb IS
COMPONENT nor_gate
   PORT(  A : IN  std_logic;
          B : IN  std_logic;
          Y : OUT  std_logic
         );
END COMPONENT;
signal A : std_logic := '0';
signal B : std_logic := '0';
signal Y : std_logic;
BEGIN
uut : nor_gate PORT MAP (
    A => A,
    B => B,
    Y => Y
  );
Stim_proc : process
begin
wait for 100 ns;
    A <= '0';
    B <= '0';
    wait for 100 ns;
    A <= '0';
    B <= '1';
    wait for 100 ns;
    A <= '1';
    B <= '0';
    wait for 100 ns;
    A <= '1';
    B <= '1';
```

```
    wait for 100 ns;
end process;
END;
```

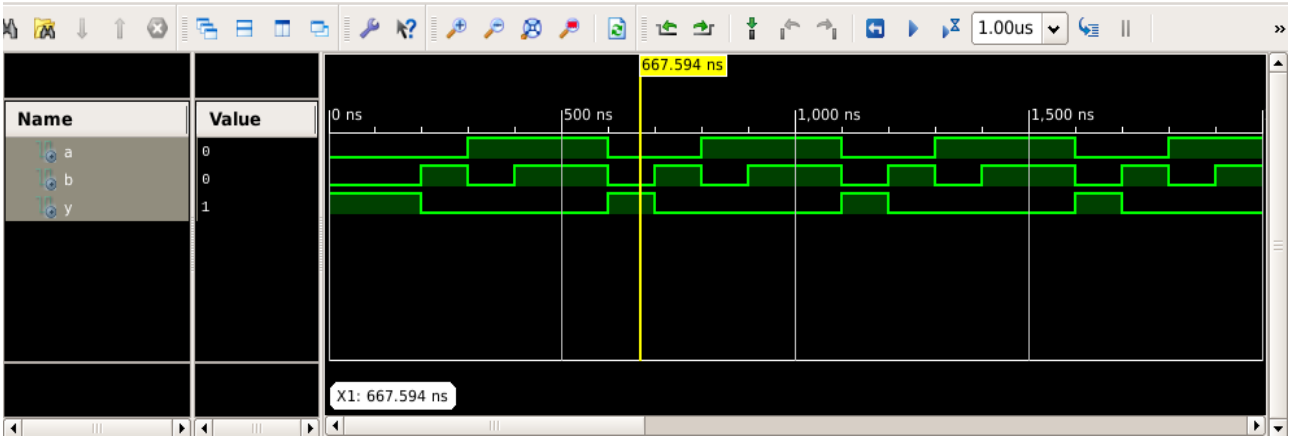
RTL Schematic :

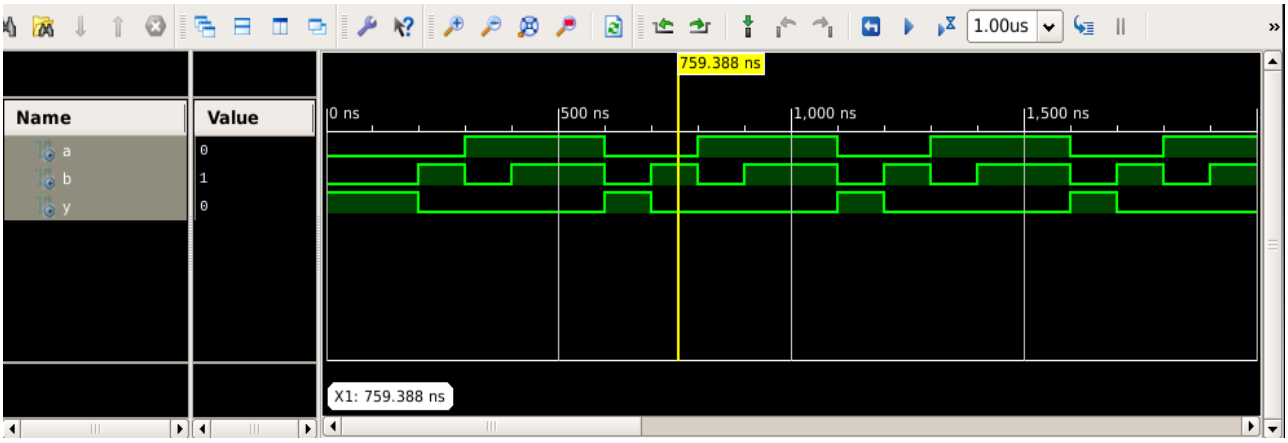


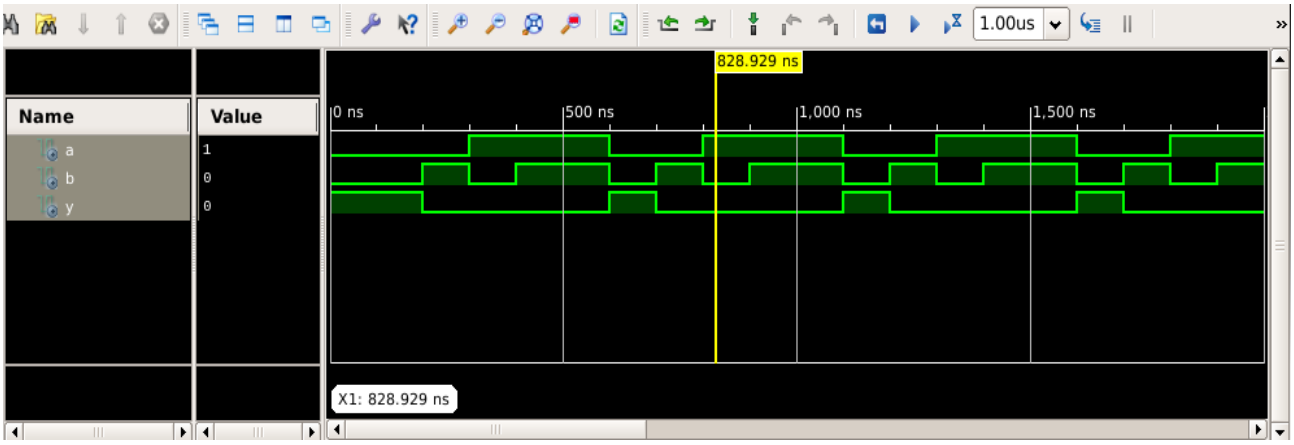
Technology Schematic :

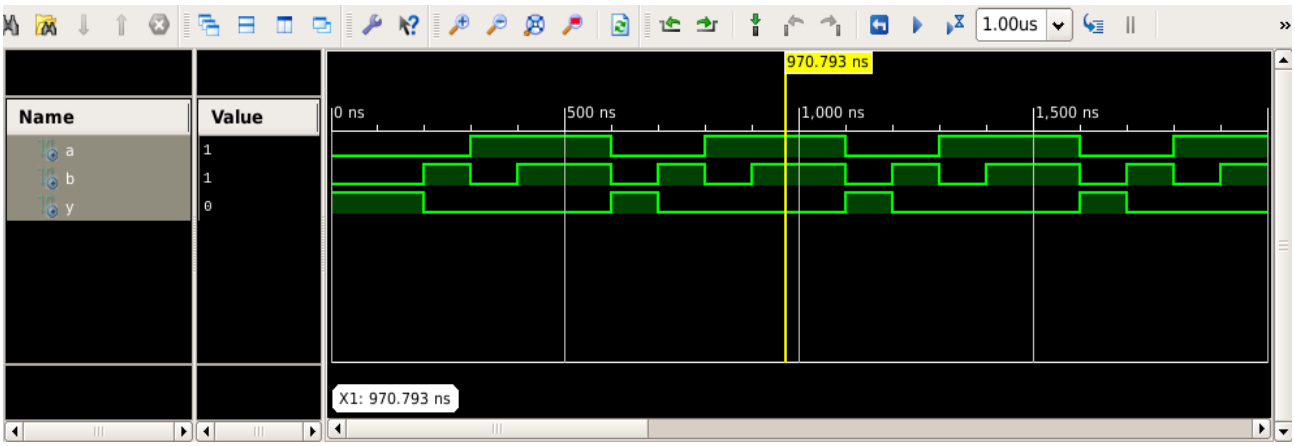


Waveform :









Design Summary :

```
================================================================================
*                           Design Summary                                     *
================================================================================

Top Level Output File Name         : nor_gate.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 1
#      LUT2                        : 1
# IO Buffers                       : 3
#      IBUF                        : 2
#      OBUF                        : 1
```

Module 7 : XNOR Gate

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    02:59:46 10/11/2023
-- Design Name:    Logic Gates
-- Module Name:    xnor_gate_48 - Behavioral
-- Project Name:   XNOR Gate
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

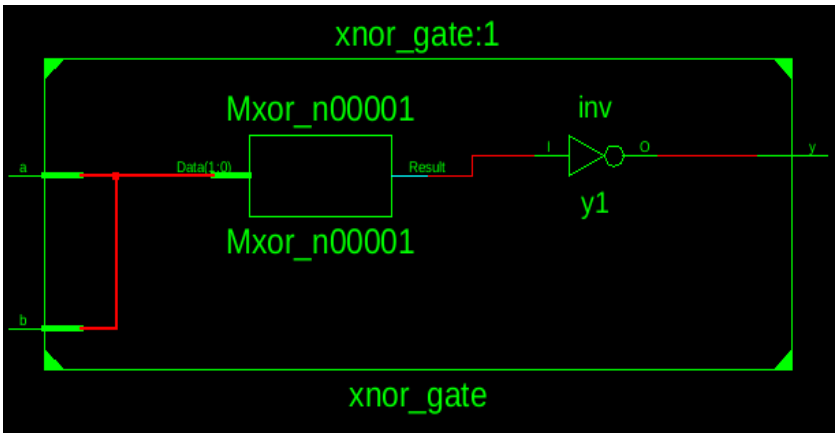
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity xnor_gate is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          y : out  STD_LOGIC);
end xnor_gate;
architecture xnor_gate_arch of xnor_gate is
begin
      y <= a xnor b;
end xnor_gate_arch;
```

Testbench :

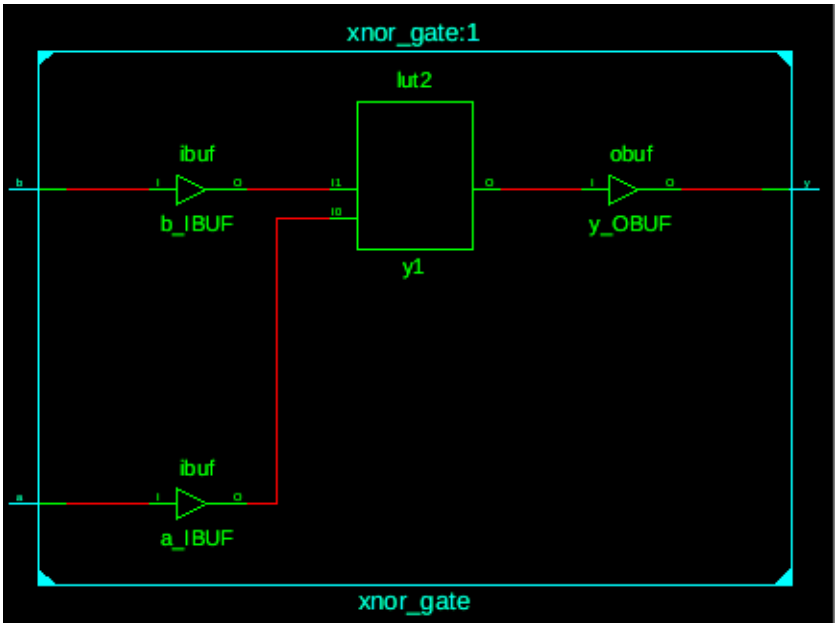
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY xnor_gate_tb IS
END xnor_gate_tb;
ARCHITECTURE xnor_gate_tb_arch OF xnor_gate_tb IS
COMPONENT xnor_gate
   PORT(  A : IN  std_logic;
          B : IN  std_logic;
          Y : OUT  std_logic
        );
END COMPONENT;
signal A : std_logic := '0';
signal B : std_logic := '0';
signal Y : std_logic;
BEGIN
uut : xnor_gate PORT MAP (
    A => A,
    B => B,
    Y => Y
  );
```

```
Stim_proc : process
begin
wait for 100 ns;
    A <= '0';
    B <= '0';
    wait for 100 ns;
    A <= '0';
    B <= '1';
    wait for 100 ns;
    A <= '1';
    B <= '0';
    wait for 100 ns;
    A <= '1';
    B <= '1';
    wait for 100 ns;
end process;
END;
```

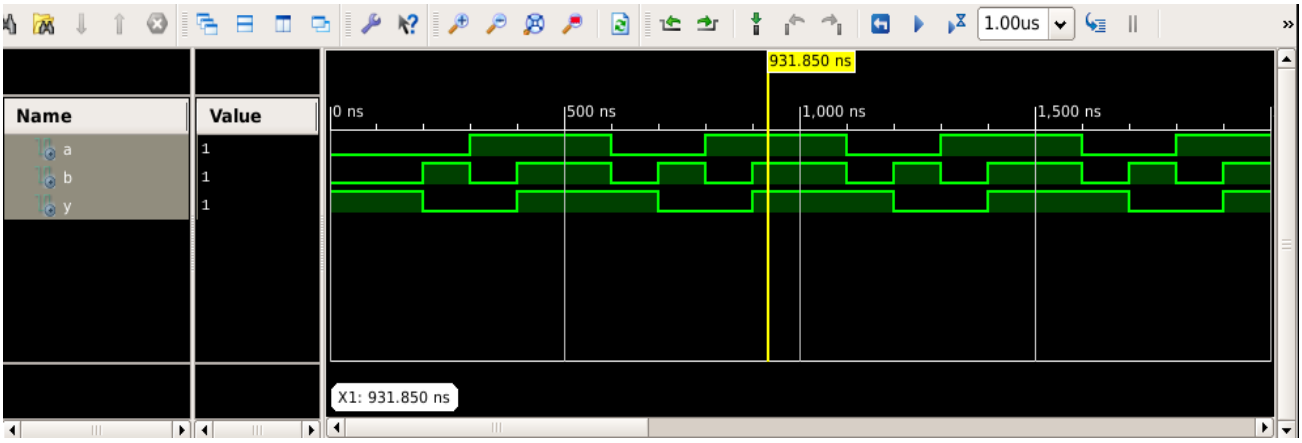
RTL Schematic :

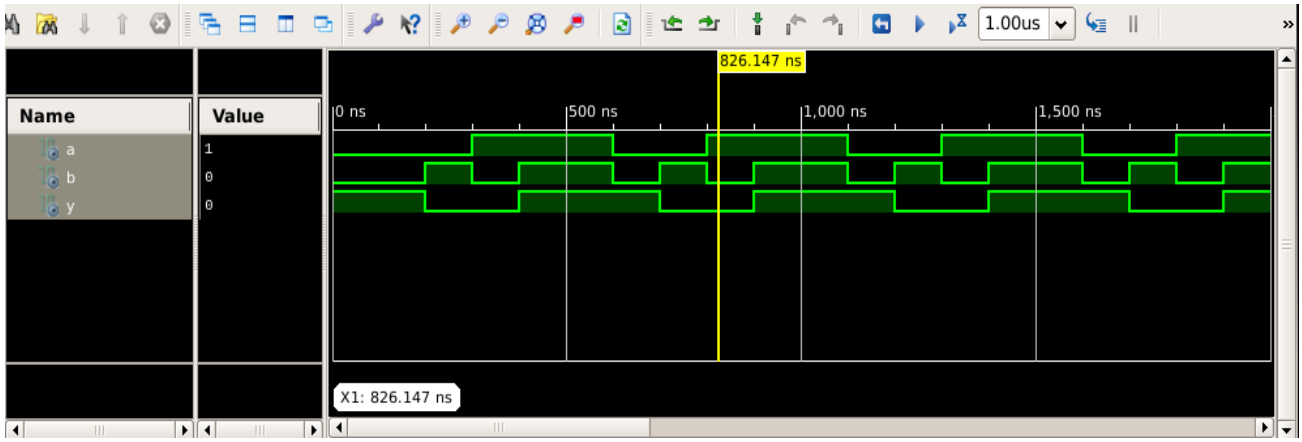


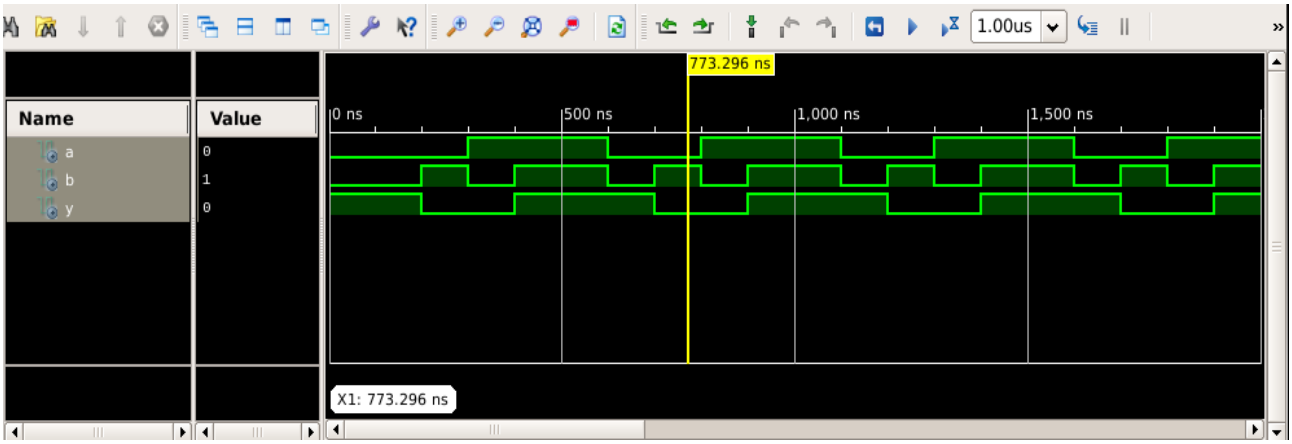
Technology Schematic :

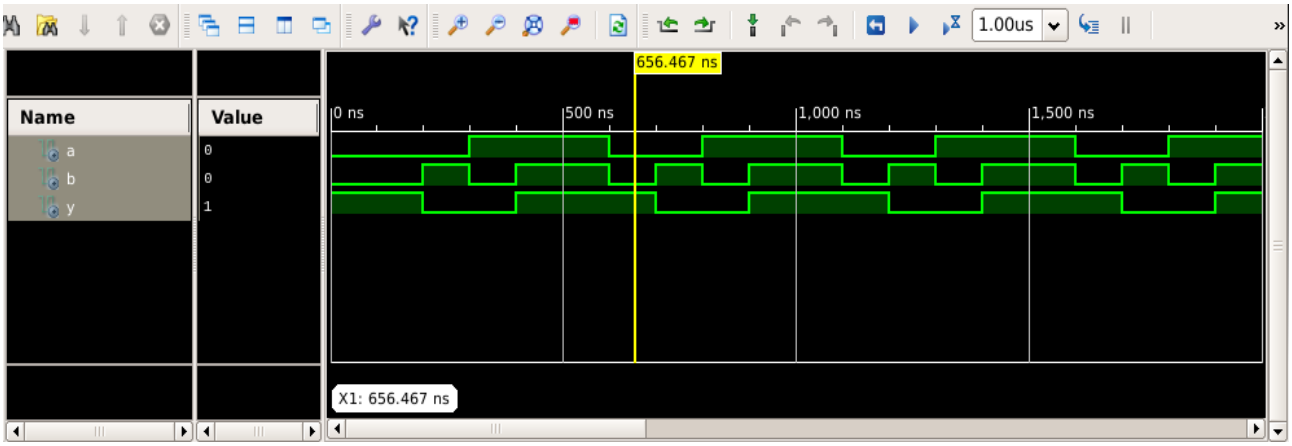


Waveform :


1.00us
931.850 ns
Name
Value
a
b
y
1
1
1
0 ns
500 ns
1,000 ns
1,500 ns
X1: 931.850 ns


1.00us
826.147 ns
Name
Value
a
b
y
1
0
0
0 ns
500 ns
1,000 ns
1,500 ns
X1: 826.147 ns


1.00us
773.296 ns
Name
Value
a
b
y
0
1
0
0 ns
500 ns
1,000 ns
1,500 ns
X1: 773.296 ns


1.00us
656.467 ns
Name
Value
a
b
y
0
0
1
0 ns
500 ns
1,000 ns
1,500 ns
X1: 656.467 ns

Design Summary :

```
=========================================================================
*                            Design Summary                             *
=========================================================================

Top Level Output File Name         : xnor_gate.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 1
#      LUT2                        : 1
# IO Buffers                       : 3
#      IBUF                        : 2
#      OBUF                        : 1
```

Conclusion :

Experiment No. 2

Title : Verification of Truth Tables of Adders

Software Used : Xilinx 14.7

Apparatus Required :FPGA Board

Theory :

An adder is a fundamental digital circuit component that performs addition of binary numbers. It takes two binary numbers as input and produces their sum as output. Adders are crucial in various digital applications, including arithmetic operations in processors and the calculators. There are several types of adders commonly used in digital circuits, each with its own characteristics and applications. The most common types of adders include –

1) Half Adder
2) Full Adder

Procedure :

1) Create a new Project, select new source as VHDL module and write the VHDL code for the intendeddesign.
2) Perform Check syntax operation to verify that the code is syntacticallycorrect.
3) View RTL schematic and observe the Block diagram and its detailedconnection.
4) Study the synthesis report indetail.
5) Launch ISIM Simulator, force signals to the inputs in signal window, run the simulator and observe the output in Wavewindow.
6) WriteVHDLtestbenchfortheDesign-under-test(DUT)andsimulateittoobservethebehaviour ofdesign.
7) Select the target FPGA device.
8) For the VHDL file, create Implementation constraintfile.
9) In UCF file, assign package pins to the input and output ports of design object list as per the selected target device. (The pin numbers can be referred from the Matrix II boardmanual.
10) Implement the design that includes translation, mapping, placement and routing.
11) SelecttheFPGAstartupclockandconfigurationmodeasperttheselectedtargetFPGAdevice (Spartan6).
12) Add Xilinx device, select the appropriate bit file and program theFPGA.
13) Applytheinputthroughonboardswitchesorinterfaced Add-onModuleandobservetheoutput on the intended output device which may be LED/LCD/SSD/Add-onModule.

Module 1 :Half Adder

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    02:59:46 10/11/2023
-- Design Name:    Adders
-- Module Name:    half_adder_48 - Behavioral
-- Project Name:   Half Adder
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

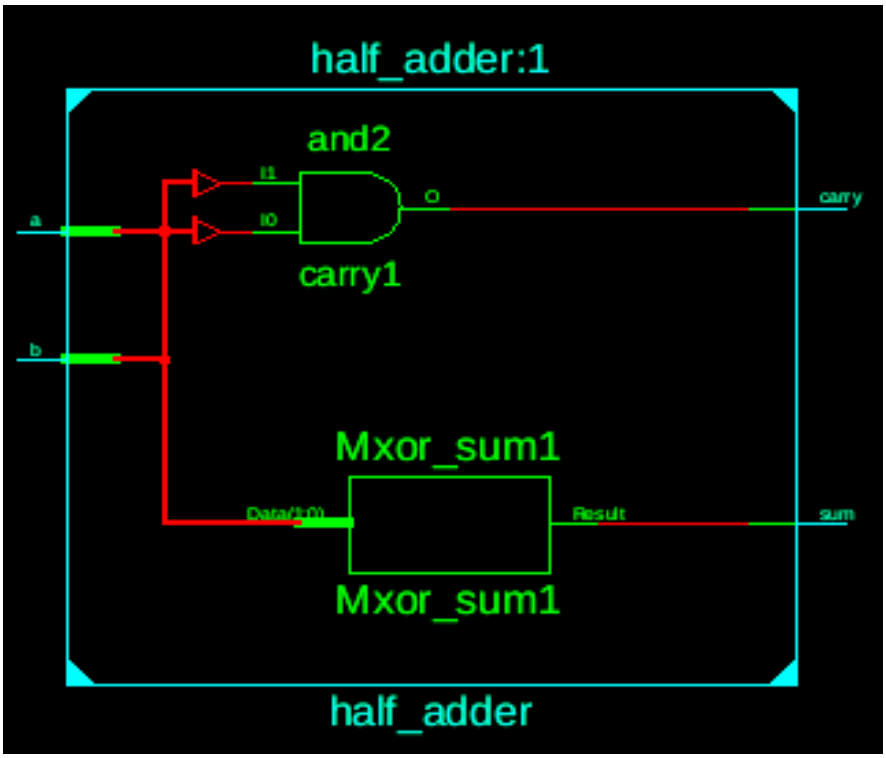
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity half_adder is
   Port ( a : in  STD_LOGIC;
b : in  STD_LOGIC;
sum : out  STD_LOGIC;
carry : out  STD_LOGIC);
end half_adder;
architecture half_adder_arch of half_adder is
begin
sum <= a xor b;
carry <= a and b;
end half_adder_arch;
```

Testbench :

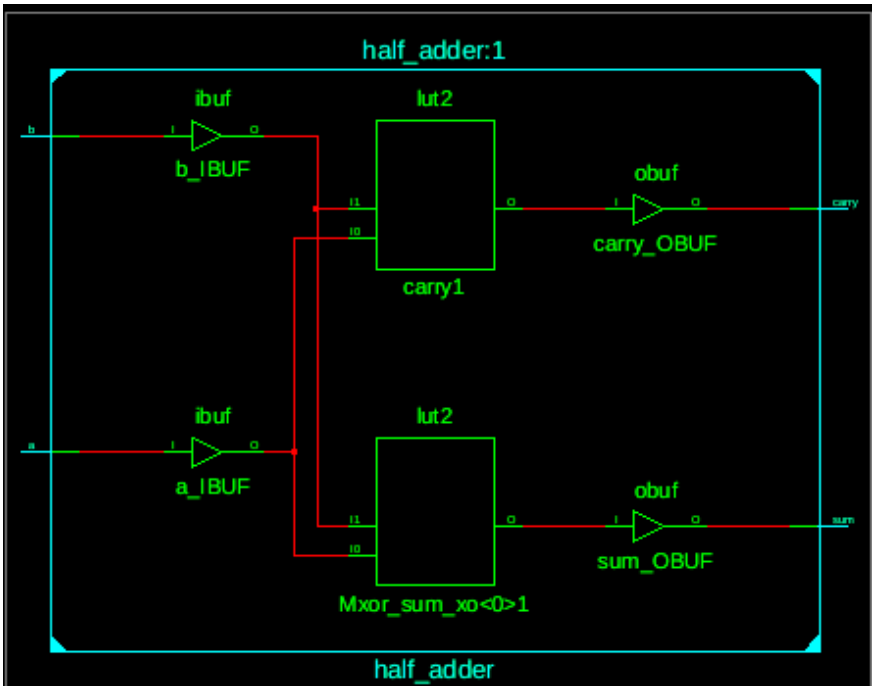
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY half_adder_tb IS
END half_adder_tb;
ARCHITECTURE half_adder_tb_arch OF half_adder_tb IS
COMPONENT half_adder
PORT(A : IN  std_logic;
       B : IN  std_logic;
       Sum : OUT  std_logic;
Carry : OUT  std_logic
 );
END COMPONENT;
signal A :std_logic := '0';
signal B :std_logic := '0';
signal Sum :std_logic;
signal Carry :std_logic;
BEGIN
uut: half_adder PORT MAP (
       A => A,
```

```
        B => B,
        Sum => Sum,
        Carry => Carry
    );
  stim_proc: process
  begin
  wait for 100 ns;
  A <= '0';
  B <= '0';
wait for 100 ns;
  A <= '0';
  B <= '1';
  wait for 100 ns;
  A <= '1';
  B <= '0';
  wait for 100 ns;
  A <= '1';
  B <= '1';
  wait for 100 ns;
  end process;
  END;
```

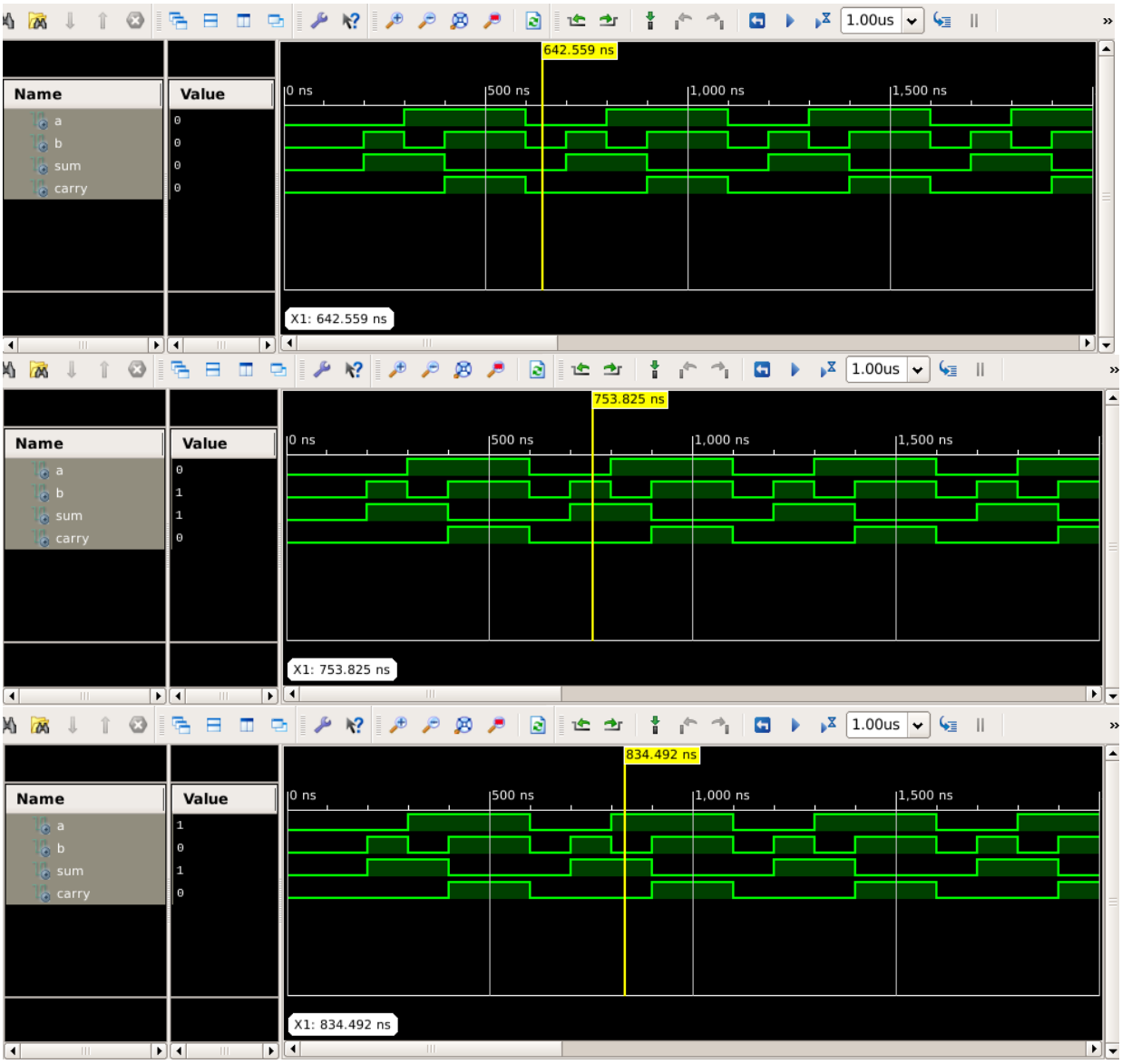
RTL Schematic:

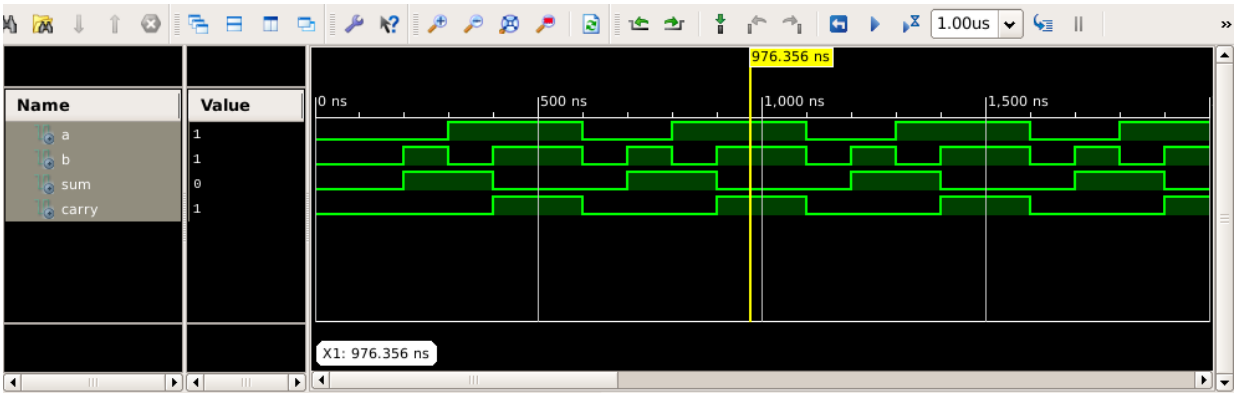


Technology Schematic :



Waveform :





Design Summary :

```
================================================================================
*                            Design Summary                                    *
================================================================================

Top Level Output File Name         : half_adder.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 2
#      LUT2                        : 2
# IO Buffers                       : 4
#      IBUF                        : 2
#      OBUF                        : 2
```

Module 2 :Full Adder

```
----------------------------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    02:59:46 10/11/2023
-- Design Name:    Adders
-- Module Name:    full_adder48 - Behavioral
-- Project Name:   Full Adder
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------------------------
```

VHDL Program :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity full_adder is
   Port ( a : in  STD_LOGIC;
b : in  STD_LOGIC;
c : in  STD_LOGIC;
sum : out  STD_LOGIC;
carry : out  STD_LOGIC);
end full_adder;
architecture full_adder_arch of full_adder is
begin
sum <= a xor b xor c;
```

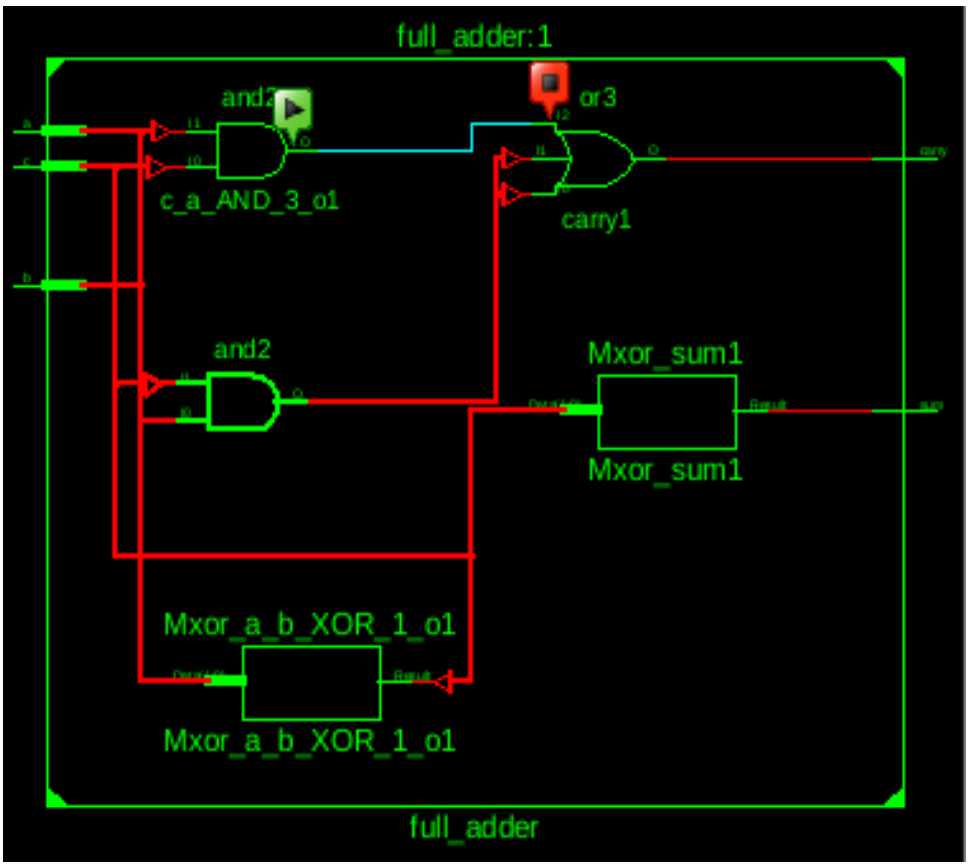
```
carry <= (a and b) or (b and c) or (c and a);
end full_adder_arch;
```

Testbench :

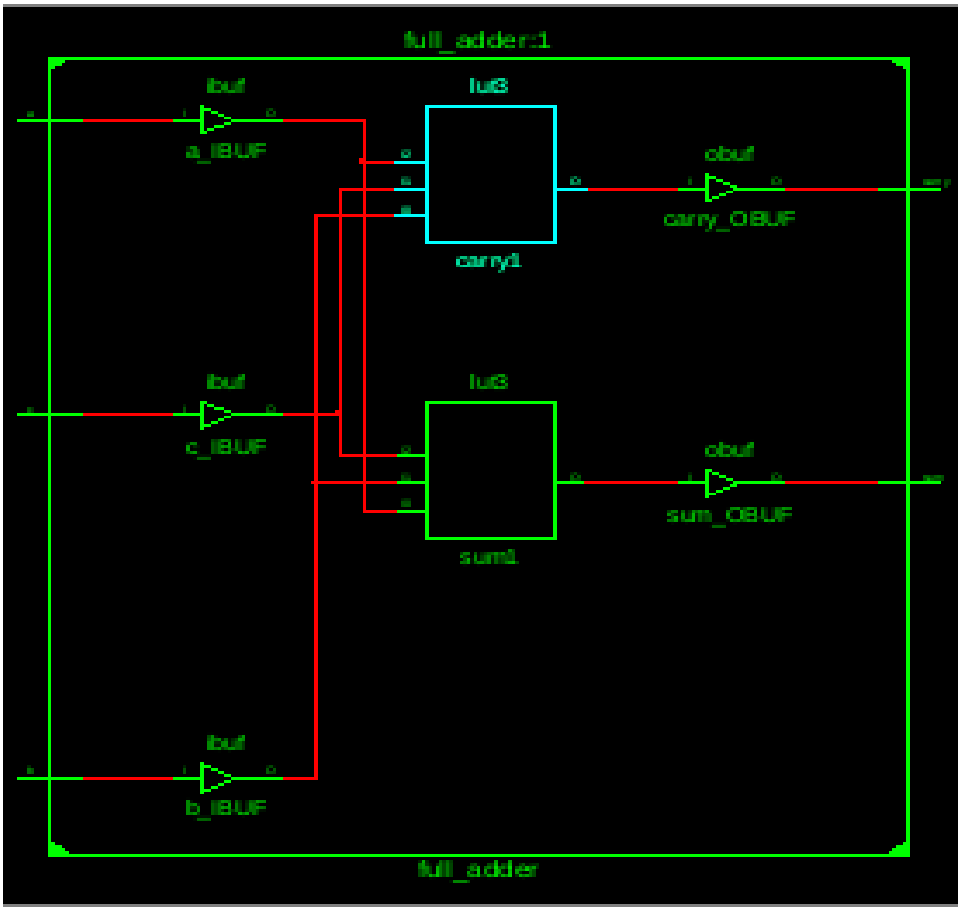
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY full_adder_tb IS
END full_adder_tb;
ARCHITECTURE behavior OF full_adder_tb IS
COMPONENT full_adder
PORT(A : IN  std_logic;
       B : IN  std_logic;
C : IN  std_logic;
Sum : OUT  std_logic;
Carry : OUT  std_logic
  );
END COMPONENT;
signal A :std_logic := '0';
signal B :std_logic := '0';
signal C :std_logic := '0';
signal Sum :std_logic;
signal Carry :std_logic;
BEGIN
uut: full_adder PORT MAP (
        A => A,
        B => B,
        C => C,
        Sum => Sum,
        Carry => Carry
     );
stim_proc: process
begin
wait for 100 ns;
    A <= '0';
B <= '0';
    C<= '0';
wait for 100 ns;
A <= '0';
B <= '0';
     C<= '1';
wait for 100 ns;
A <= '0';
B <= '1';
     C<= '0';
wait for 100 ns;
A <= '0';
B <= '1';
```

```
      C<= '1';
wait for 100 ns;
A <= '1';
B <= '0';
      C<= '0';
wait for 100 ns;
A <= '1';
       B <= '0';
       C<= '1';
wait for 100 ns;
       A <= '1';
       B <= '1';
       C<= '0';
       wait for 100 ns;
       A <= '1';
       B <= '1';
       C<= '1';
       wait for 100 ns;
end process;
END;
```

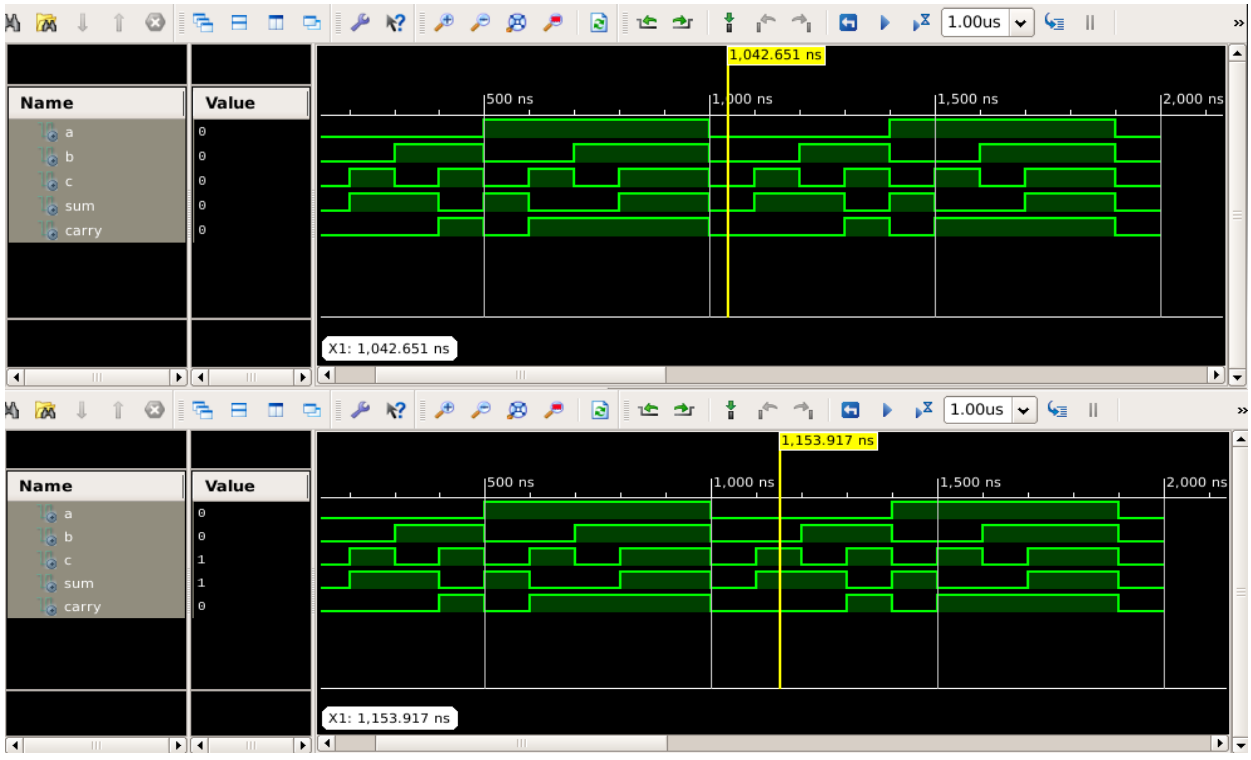
RTL Schematic :



Technology Schematic :



Waveform :




1.00us
Name
Value
a
b
c
sum
carry
0
1
0
1
0
500 ns
1,000 ns
1,500 ns
2,000 ns
1,251.274 ns
X1: 1,251.274 ns
1.00us
Name
Value
a
b
c
sum
carry
0
1
1
0
1
1,348.632 ns
500 ns
1,000 ns
1,500 ns
2,000 ns
X1: 1,348.632 ns
1.00us
Name
Value
a
b
c
sum
carry
1
0
0
1
0
1,459.898 ns
500 ns
1,000 ns
1,500 ns
2,000 ns
X1: 1,459.898 ns
1.00us
Name
Value
a
b
c
sum
carry
1
0
1
0
1
1,554.473 ns
500 ns
1,000 ns
1,500 ns
2,000 ns
X1: 1,554.473 ns
1.00us
Name
Value
a
b
c
sum
carry
1
1
0
0
1
1,654.612 ns
500 ns
1,000 ns
1,500 ns
2,000 ns
X1: 1,654.612 ns

Design Summary :

```
=========================================================================
*                            Design Summary                             *
=========================================================================

Top Level Output File Name         : full_adder.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 2
#      LUT3                        : 2
# IO Buffers                       : 5
#      IBUF                        : 3
#      OBUF                        : 2
```

Conclusion :

Experiment No. 3

Title : Verification of Truth Tables of Multiplexer

Software Used : Xilinx 14.7

Apparatus Required : FPGA Board

Theory :

Multiplexers are fundamental building blocks in digital circuits and play a crucial role in data routing and selection. They are used to select one of several input data lines and route it to a single output line based on the control signals. Multiplexers have a variety of applications in digital systems, including data transmission, data compression, and control logic.

Procedure :

1) Create a new Project, select new source as VHDL module and write the VHDL code for the intended design.
2) Perform Check syntax operation to verify that the code is syntactically correct.
3) View RTL schematic and observe the Block diagram and its detailed connection.
4) Study the synthesis report in detail.
5) Launch ISIM Simulator, force signals to the inputs in signal window, run the simulator and observe the output in Wave window.
6) Write VHDL testbench for the Design-under-test (DUT) and simulate it to observe the behaviour of design.
7) Select the target FPGA device.
8) For the VHDL file, create Implementation constraint file.
9) In UCF file, assign package pins to the input and output ports of design object list as per the selected target device. (The pin numbers can be referred from the Matrix II board manual.
10) Implement the design that includes translation, mapping, placement and routing.
11) Select the FPGA startup clock and configuration mode as per the selected target FPGA device (Spartan 6).
12) Add Xilinx device, select the appropriate bit file and program the FPGA.
13) Apply the input through onboard switches or interfaced Add-on Module and observe the output on the intended output device which may be LED/LCD/SSD/Add-on Module.

Module 1 : Multiplexer 4 to 1

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering and Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    10:23:14 21/09/2023
-- Design Name:    Multiplexer
-- Module Name:    Multiplexer_4to1_48 Behaviourial
-- Project Name:   Multiplexer 4:1
-- Target Device:  XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

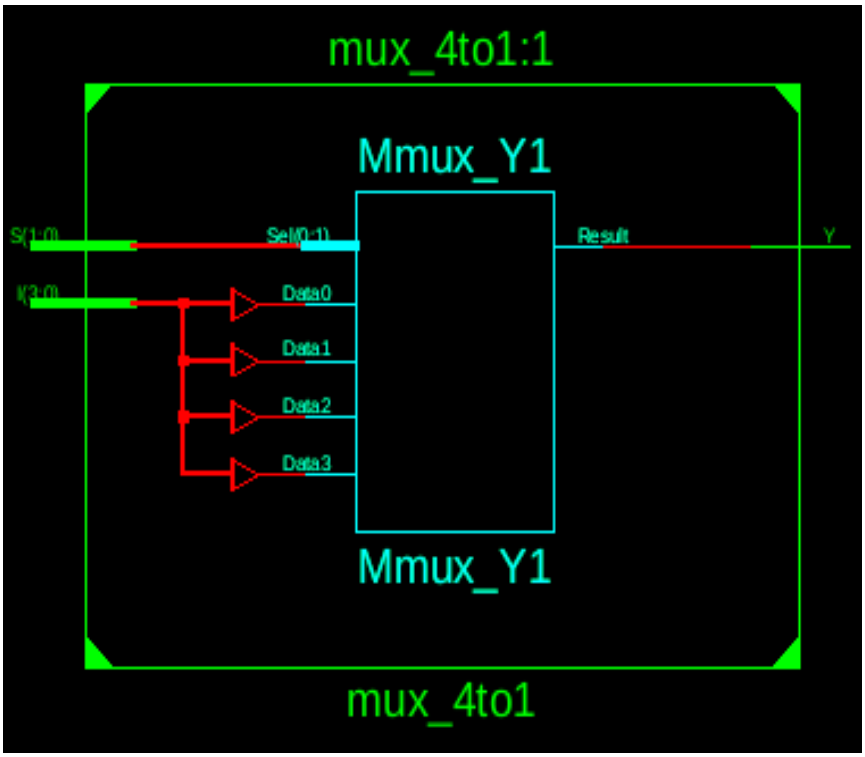
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity mux_4to1 is
   Port ( I : in  STD_LOGIC_VECTOR (3 downto 0);
          S : in  STD_LOGIC_VECTOR (1 downto 0);
          Y : out  STD_LOGIC);
end mux_4to1;
architecture mux_4to1_arch of mux_4to1 is
begin
with s select
            y <= I(0) when "00",
            I(1) when "01",
            I(2) when "10",
            I(3) when others;
end mux_4to1_arch;
```

Testbench :

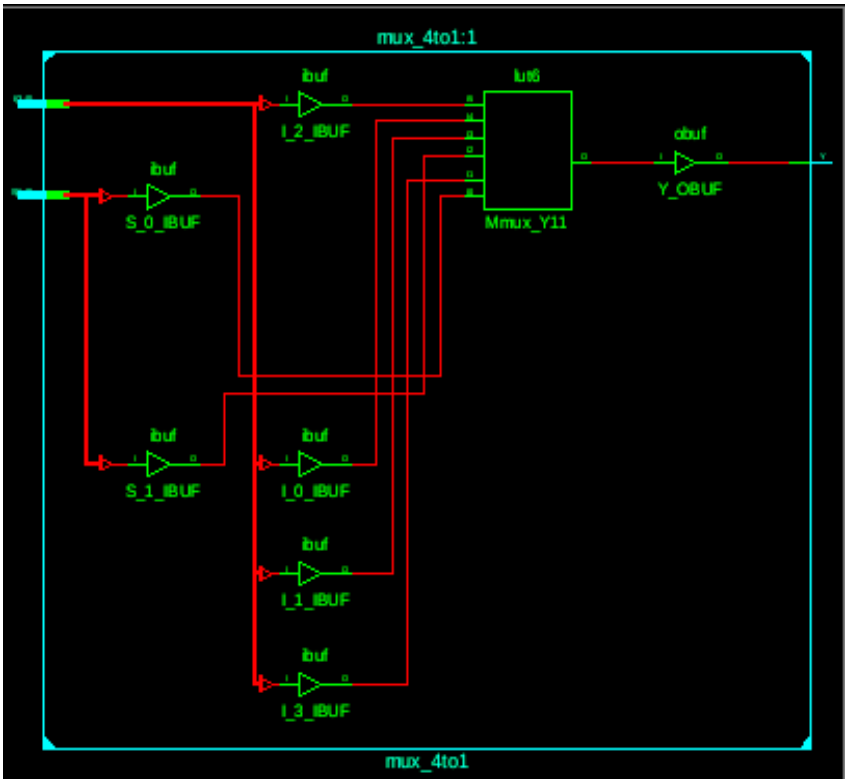
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity mux_4to1_tb is
end mux_4to1_tb;
architecture mux_4to1_tb_arch of mux_4to1_tb is
   signal I0, I1, I2, I3, Y : STD_LOGIC;
   signal S : STD_LOGIC_VECTOR(1 downto 0);
   component mux_4to1
      Port ( I0, I1, I2, I3 : in STD_LOGIC;
             S : in STD_LOGIC_VECTOR(1 downto 0);
             Y : out STD_LOGIC);
   end component;
begin
   UUT: mux_4to1 port map (I0, I1, I2, I3, S, Y);
   stimulus: process
   begin
      I0 <= '0';
      I1 <= '1';
      I2 <= '0';
      I3 <= '1';
      S <= "00";
      wait for 10 ns;
      S <= "01";
      wait for 10 ns;
```

```
        S <= "10";
        wait for 10 ns;
        S <= "11";
        wait for 10 ns;
          I0 <= '1';
          I1 <= '0';
          I2 <= '1';
          I3 <= '0';
        S <= "00";
        wait for 10 ns;
        S <= "01";
        wait for 10 ns;
        S <= "10";
        wait for 10 ns;
        S <= "11";
        wait for 10 ns;
        wait;
    end process;
end mux_4to1_tb_arch;
```

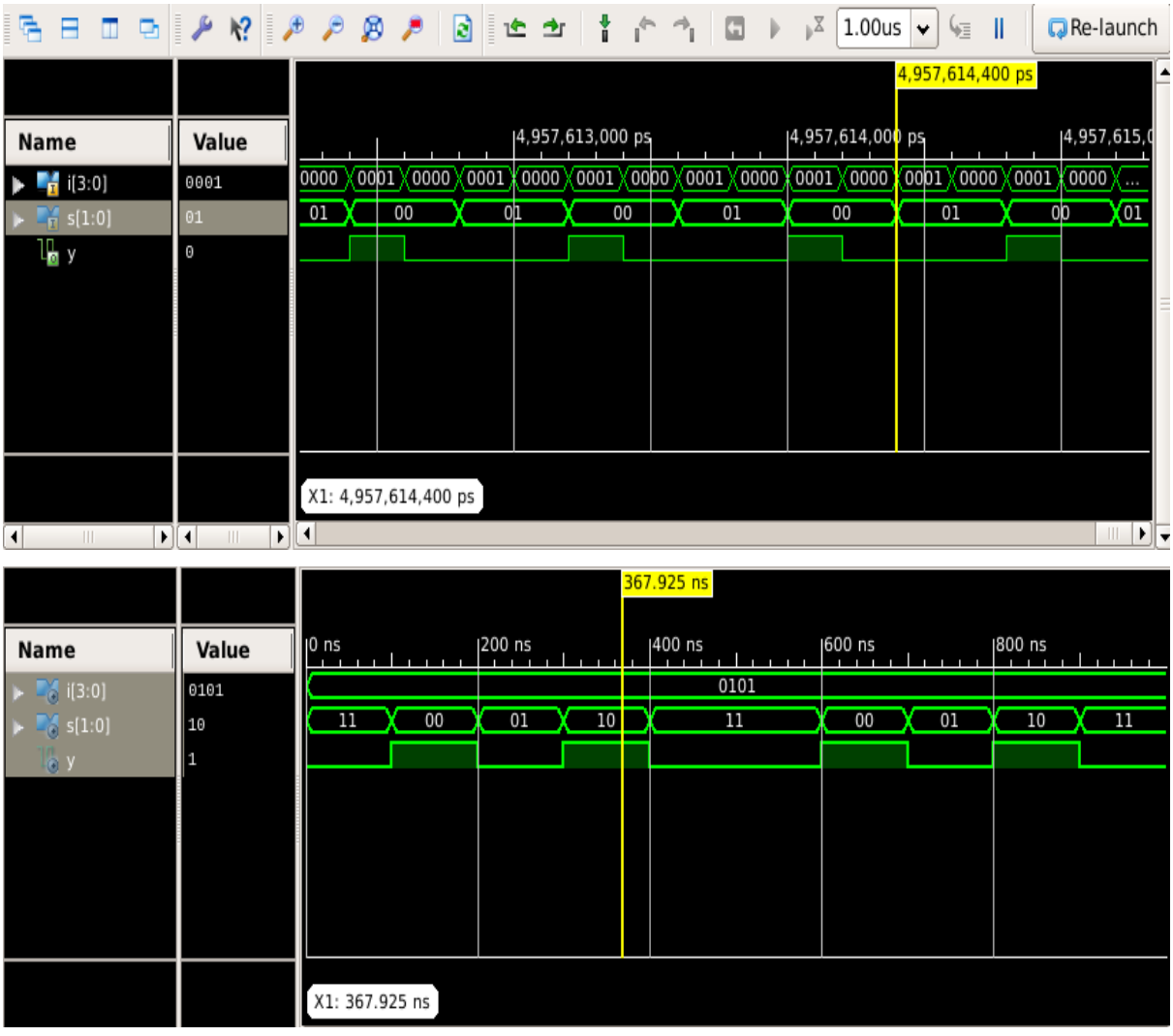
RTL Schematic :



Technology Schematic :



Waveform :



Design Summary :

```
=========================================================================
*                            Design Summary                             *
=========================================================================

Top Level Output File Name         : mux_4to1.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 1
#      LUT6                        : 1
# IO Buffers                        : 7
#      IBUF                        : 6
#      OBUF                        : 1
```

Conclusion :

Experiment No. 4

Title : Verification of Truth Tables of Encoder

Software Used : Xilinx 14.7

Apparatus Required : FPGA Board

Theory :

In digital circuits, an encoder is a combinational logic circuit that converts a set of input signals into a binary code representation at its output. The primary purpose of an encoder is to encode multiple input lines into a smaller number of output lines. Encoders are commonly used in various applications, including data transmission, address encoding, and multiplexing/demultiplexing.

Procedure :

1) Create a new Project, select new source as VHDL module and write the VHDL code for the intended design.
2) Perform Check syntax operation to verify that the code is syntactically correct.
3) View RTL schematic and observe the Block diagram and its detailed connection.
4) Study the synthesis report in detail.
5) Launch ISIM Simulator, force signals to the inputs in signal window, run the simulator and observe the output in Wave window.
6) Write VHDL testbench for the Design-under-test (DUT) and simulate it to observe the behaviour of design.
7) Select the target FPGA device.
8) For the VHDL file, create Implementation constraint file.
9) In UCF file, assign package pins to the input and output ports of design object list as per the selected target device. (The pin numbers can be referred from the Matrix II board manual.
10) Implement the design that includes translation, mapping, placement and routing.
11) Select the FPGA startup clock and configuration mode as per the selected target FPGA device (Spartan 6).
12) Add Xilinx device, select the appropriate bit file and program the FPGA.
13) Apply the input through onboard switches or interfaced Add-on Module and observe the output on the intended output device which may be LED/LCD/SSD/Add-on Module.

Module 1 : Encoder 4 to 2

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering and Research
-- Engineer:Tanmay Vilas Pawar]

-- Create Date:    10:23:14 15/09/2023
-- Design Name:    Encoder
-- Module Name:    encodrer_4to2_48 - Behaviourial
-- Project Name:   Encoder 4:2
-- Target Device:  XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

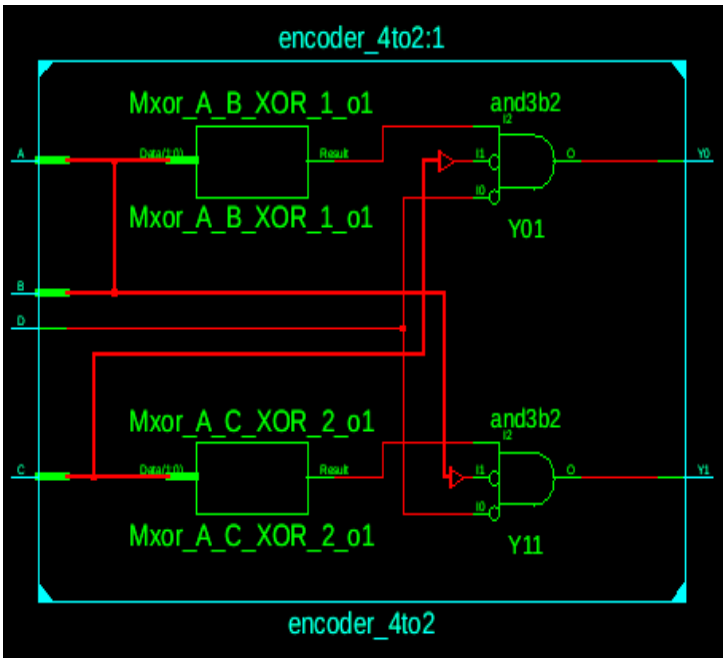
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity encoder_4to2 is
   Port ( A,B,C,D : in  STD_LOGIC;
            Y0,Y1 : out  STD_LOGIC);
end encoder_4to2;
architecture encoder_4to2_arch of encoder_4to2 is
begin
      Y0 <= ((not C)and(not D))and(A xor B);
      Y1 <= ((not B)and(not D))and(A xor C);
end encoder_4to2_arch;
```

Testbench :

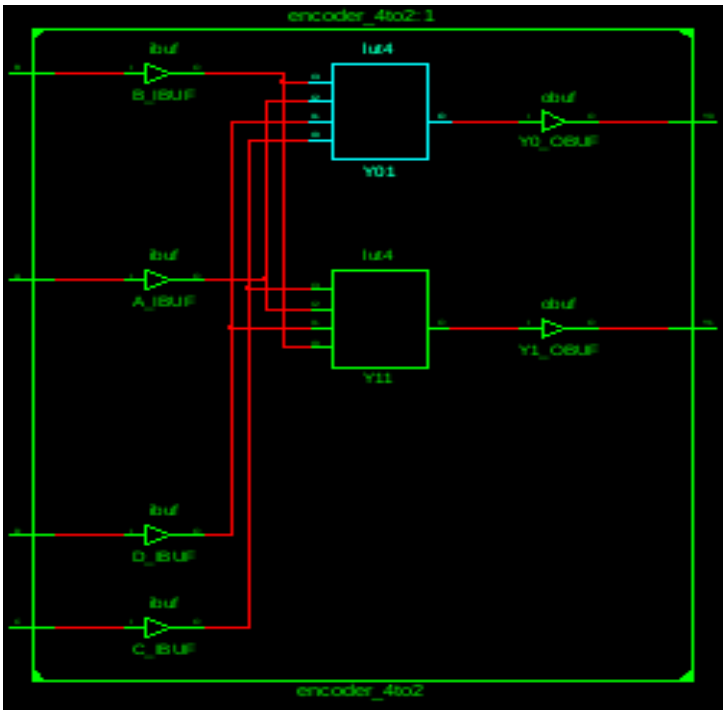
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity encoder_4to2_tb is
end encoder_4to2_tb;
architecture encoder_4to2_tb_arch of encoder_4to2_tb is
   signal I0, I1, I2, I3 : STD_LOGIC;
   signal Y : STD_LOGIC_VECTOR(1 downto 0);
   component encoder_4to2
      Port ( I0, I1, I2, I3 : in STD_LOGIC;
           Y : out STD_LOGIC_VECTOR(1 downto 0));
   end component;
        begin
   UUT: encoder_4to2 port map (I0, I1, I2, I3, Y);
   stimulus: process
   begin
      I0 <= '0';
      I1 <= '0';
      I2 <= '0';
      I3 <= '0';
      wait for 10 ns;
      I0 <= '1';
      wait for 10 ns;
      I1 <= '1';
      wait for 10 ns;
      I2 <= '1';
      wait for 10 ns;
                  I3 <= '1';
      wait for 10 ns;
      I0 <= '0';
      I1 <= '1';
```

```
        I2 <= '0';
        I3 <= '1';
        wait for 10 ns;
        I0 <= '1';
        I1 <= '0';
        I2 <= '1';
        I3 <= '0';
        wait for 10 ns;
        wait;
    end process;
end encoder_4to2_tb_arch;
```

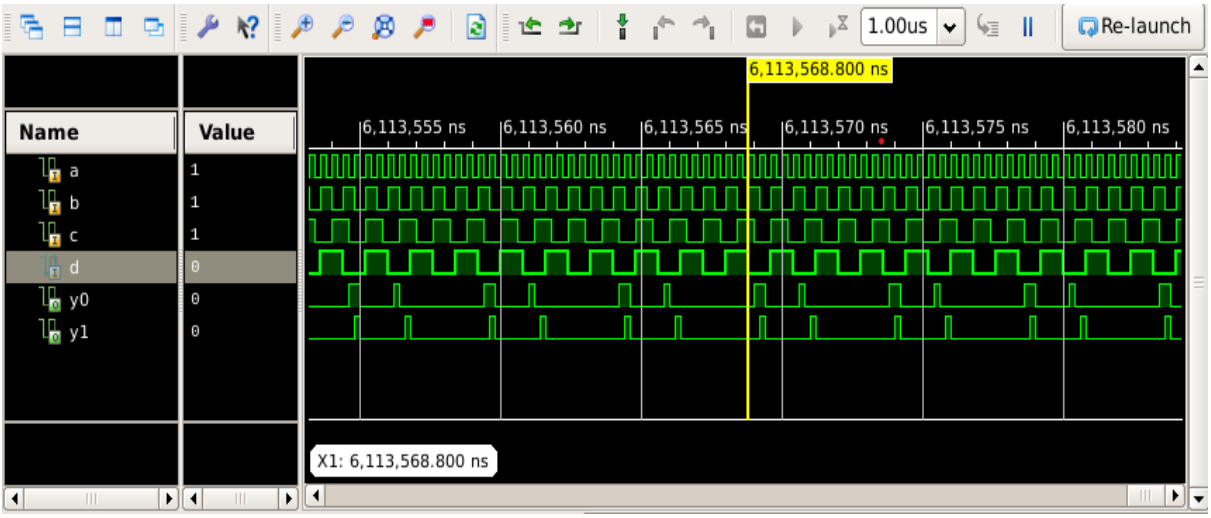
RTL Schematic :



Technology Schematic :



Waveform :



Design Summary :

```
================================================================================
*                            Design Summary                                    *
================================================================================

Top Level Output File Name         : encoder_4to2.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 2
#      LUT4                        : 2
# IO Buffers                       : 6
#      IBUF                        : 4
#      OBUF                        : 2
```

Conclusion :

Experiment No. 5

Title : Verification of Truth Tables of Decoder

Software Used : Xilinx 14.7

Apparatus Required : FPGA Board

Theory :

In digital circuits, a decoder is a combinational logic circuit that takes a binary input code and activates one specific output line based on that code. Decoders are the inverse of encoders, which take multiple input lines and produce a binary code at the output. Decoders are widely used in digital systems for tasks such as address decoding, data demultiplexing, and controlling multiple devices or operations based on specific input conditions.

Procedure :

1) Create a new Project, select new source as VHDL module and write the VHDL code for the intended design.
2) Perform Check syntax operation to verify that the code is syntactically correct.
3) View RTL schematic and observe the Block diagram and its detailed connection.
4) Study the synthesis report in detail.
5) Launch ISIM Simulator, force signals to the inputs in signal window, run the simulator and observe the output in Wave window.
6) Write VHDL testbench for the Design-under-test (DUT) and simulate it to observe the behaviour of design.
7) Select the target FPGA device.
8) For the VHDL file, create Implementation constraint file.
9) In UCF file, assign package pins to the input and output ports of design object list as per the selected target device. (The pin numbers can be referred from the Matrix II board manual.
10) Implement the design that includes translation, mapping, placement and routing.
11) Select the FPGA startup clock and configuration mode as per the selected target FPGA device (Spartan 6).
12) Add Xilinx device, select the appropriate bit file and program the FPGA.
13) Apply the input through onboard switches or interfaced Add-on Module and observe the output on the intended output device which may be LED/LCD/SSD/Add-on Module.

Module 1 : Decoder 2 to 4

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering and Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    10:23:14 18/09/2023
-- Design Name:    Decoder
-- Module Name:    decoder_2to4_48 Behaviourial
-- Project Name:   Decoder 2:4
-- Target Device:  XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

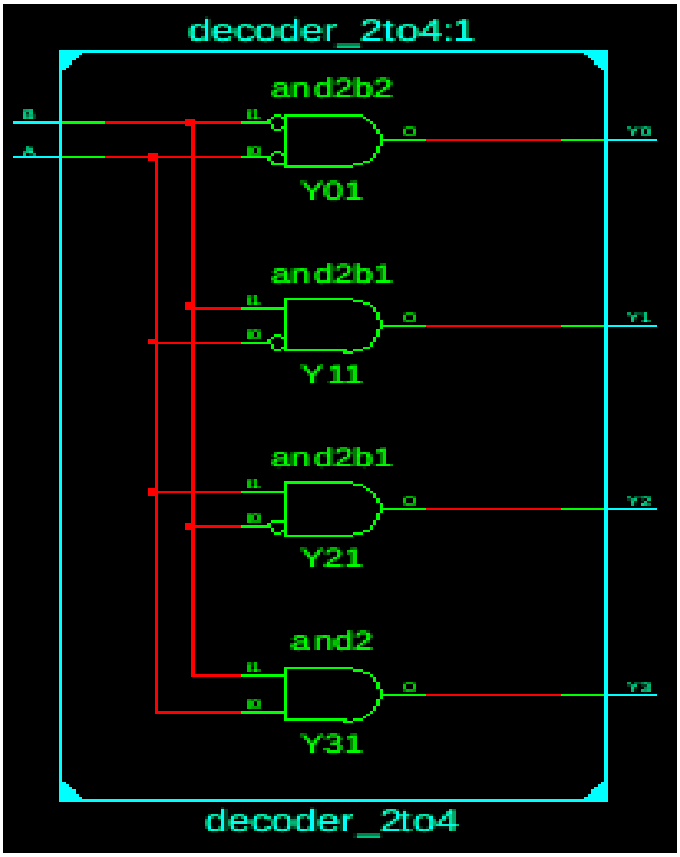
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity decoder_2to4 is
   Port ( A,B : in  STD_LOGIC;
            Y0,Y1,Y2,Y3 : out  STD_LOGIC);
end decoder_2to4;
architecture decoder_2to4_arch of decoder_2to4 is
begin
      Y0 <= ((not A)and(not B));
      Y1 <= ((not A) and B);
      Y2 <= (A and (not B));
      Y3 <= (A and B);
end decoder_2to4_arch;
```

Testbench :

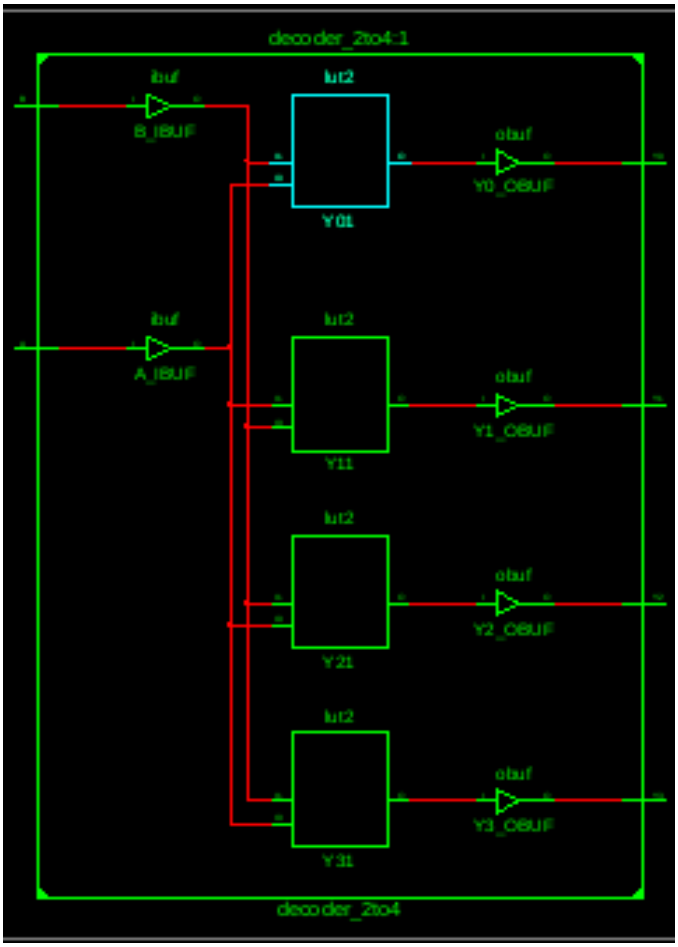
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity decoder_2to4_tb is
end decoder_2to4_tb;
architecture decoder_2to4_tb_arch of decoder_2to4_tb is
   signal A : STD_LOGIC_VECTOR(1 downto 0);
   signal Y : STD_LOGIC_VECTOR(3 downto 0);
   component decoder_2to4
      Port ( A : in STD_LOGIC_VECTOR(1 downto 0);
           Y : out STD_LOGIC_VECTOR(3 downto 0));
   end component;
begin
   UUT: decoder_2to4 port map (A, Y);
   stimulus: process
   begin
      A <= "00";
      wait for 10 ns;
      A <= "01";
      wait for 10 ns;
      A <= "10";
      wait for 10 ns;
      A <= "11";
      wait for 10 ns;
      wait;
   end process;
```

end decoder_2to4_tb_arch;

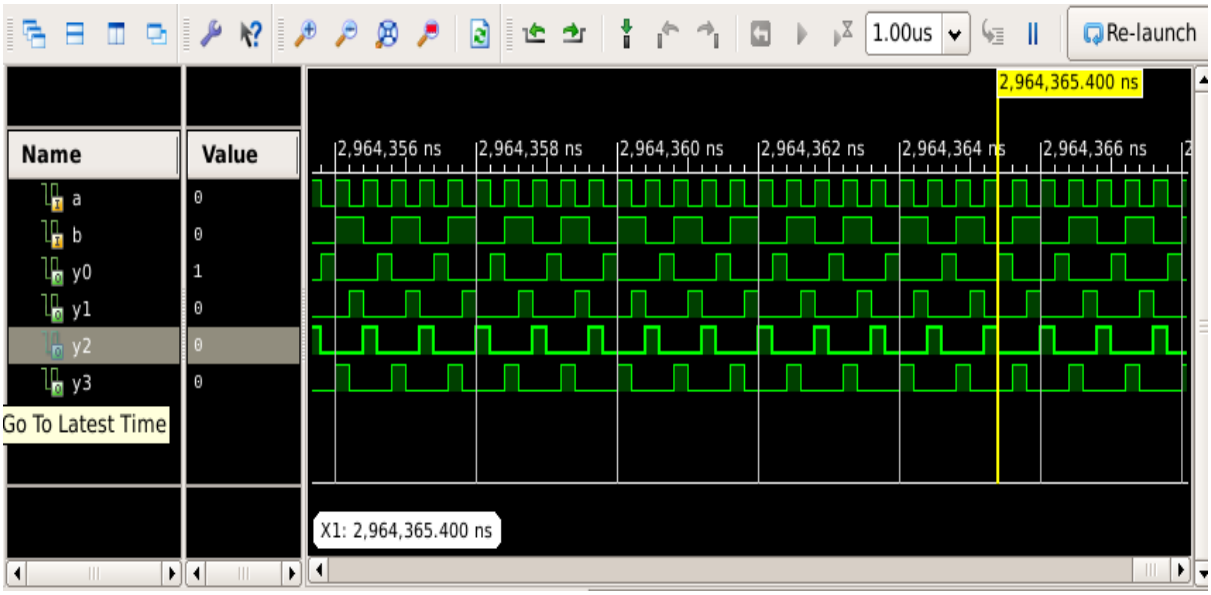
RTL Schematic :



Technology Schematic :



Waveform :



Design Summary :

```
================================================================================
*                            Design Summary                                    *
================================================================================

Top Level Output File Name         : decoder_2to4.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 4
#      LUT2                        : 4
# IO Buffers                       : 6
#      IBUF                        : 2
#      OBUF                        : 4
```

Conclusion :

Experiment No. 6

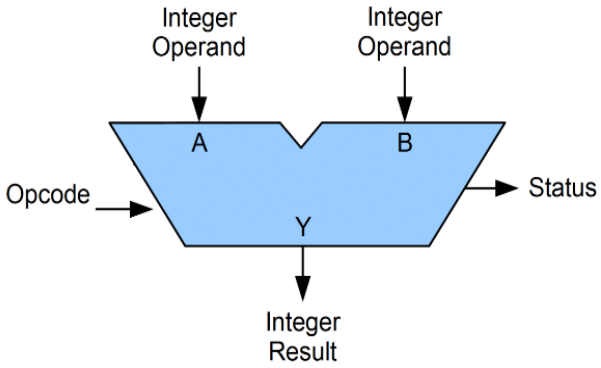
Title : To write VHDL code, simulate with test bench, synthesis, implement on PLD 4 bit ALU for Add, Subtract, AND, NAND, OR, XOR & XNOR.

Software Used : Xilinx 14.7

Apparatus Required :FPGA Board

Theory :

The VLSI (Very Large Scale Integration) ALU (Arithmetic Logic Unit) is an integral component of microprocessors, responsible for executing intricate arithmetic and logical operations on binary data. Controlled by specific inputs, it conducts tasks like addition, subtraction, AND, OR, and more. With two data inputs and a resulting output, the ALU's outcomes are crucial for program execution. Often associated with flags, indicating conditions like zero, carry, and overflow, it accommodates various data widths, from 8-bit to 64-bit. Advanced VLSI designs incorporate parallel processing techniques to bolster processing speed, and its meticulous design minimizes power consumption, ensuring efficiency in modern CPUs.



In ECL, TTL and CMOS, there are available integrated packages that are referred to as arithmeticlogicunits(ALU).Thelogiccircuitryinthisunitisentirelycombinational(i.e.consists of gates with no feedback and no flip-flops).The ALU is an extremely versatile and useful device since, it makes available, in single package, facility for performing many different logical and arithmetic operations. Arithmetic Logic Unit (ALU) is a critical component of a microprocessor andisthecorecomponentofcentralprocessingunit.ALU'scomprisethecombinationallogict hat implements logic operations such as AND, OR and arithmetic operations, such as ADD, SUBTRACT. Functionally, the operation of typical ALU is represented as shown in diagram below,

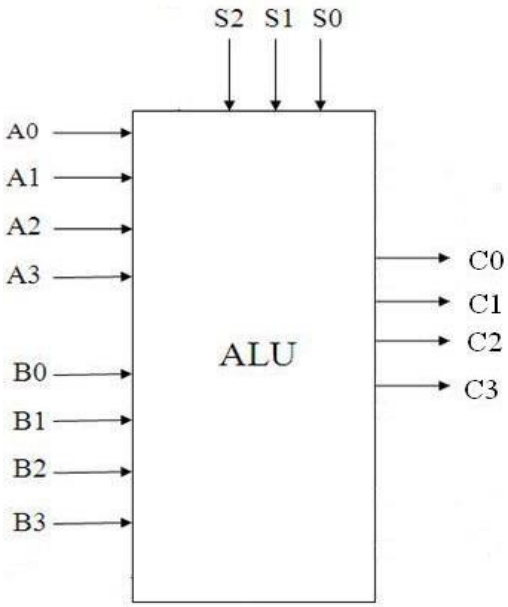


Figure 1: Functional representation of Arithmetic Logic Unit

Controlled by the three function select inputs (S0 to S2), ALU can perform all the 8 possible logicoperationsor8differentarithmeticoperationsonactiveHIGHoractiveLOWoperands.The function table lists the arithmetic operations that are performed inALU.

Module 1 :4 Bit ALU

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    02:59:46 10/11/2023
-- Design Name:    Arithematic Logic Unit
-- Module Name:    alu_48 - Behavioral
-- Project Name:   ALU
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity alu is
    Port ( A,B : in  STD_LOGIC_VECTOR (3 downto 0);
S : in  STD_LOGIC_VECTOR (2 downto 0);
Y : out  STD_LOGIC_VECTOR (3 downto 0));
end alu;
architecture alu_arch of alu is
begin
process(A,B,S)
begin
```

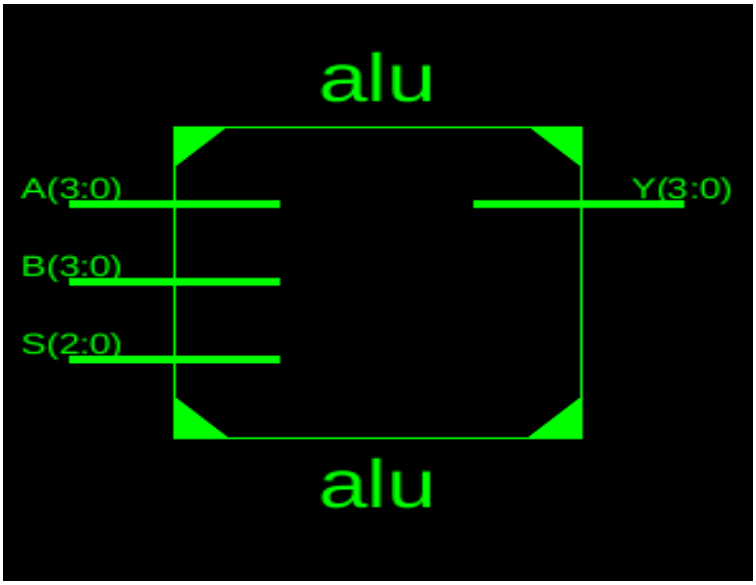
```
case S is
when "000" => Y <= A + B;
when "001" => Y <= A - B;
when "010" => Y <= A and B;
when "011" => Y <= A nand B;
when "100" => Y <= A or B;
when "101" => Y <= A xor B;
when "110" => Y <= A xnor B;
when "111" => Y <= "0000";
when others => Null;
end case;
end process;
end alu_arch;
```

Testbench :

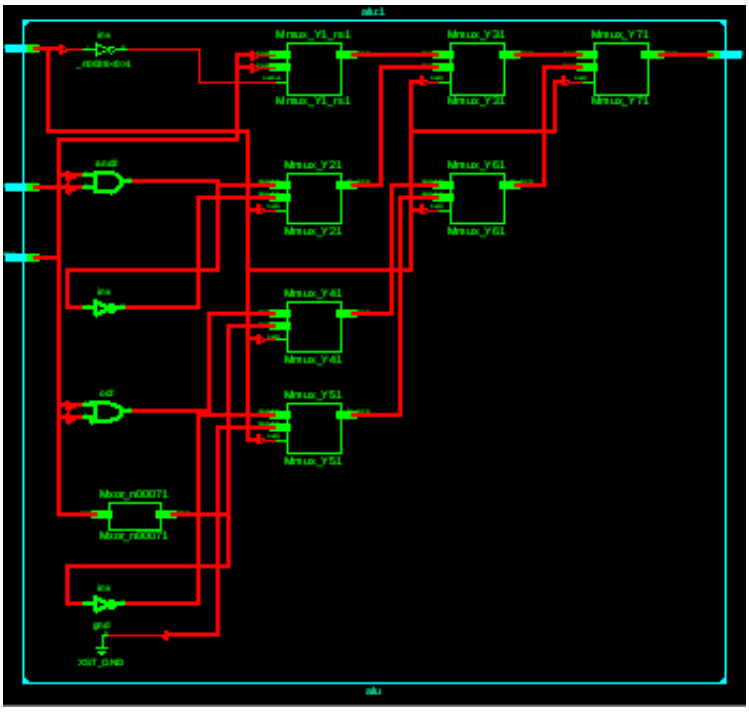
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY alu_tb IS
END alu_tb;
ARCHITECTURE alu_tb_arch OF alu_tb IS
COMPONENT alu
PORT(
A : IN  std_logic_vector(3 downto 0);
B : IN  std_logic_vector(3 downto 0);
S : IN  std_logic_vector(2 downto 0);
Y : OUT  std_logic_vector(3 downto 0)
    );
END COMPONENT;
signal A :std_logic_vector(3 downto 0) := 0101;
signal B :std_logic_vector(3 downto 0) := 1010;
signal S :std_logic_vector(2 downto 0) ;
signal Y :std_logic_vector(3 downto 0);
BEGIN
uut: alu PORT MAP(
            A => A,
            B => B,
            S => S,
            Y => Y
);
stim_proc: process
begin
        S<="000";
   wait for 100 ns;
        S<="001";
   wait for 100 ns;
        S<="010";
```

```
    wait for 100 ns;
        S<="011";
    wait for 100 ns;
        S<="100";
    wait for 100 ns;
        S<="101";
    wait for 100 ns;
        S<="110";
    wait for 100 ns;
        S<="111";
    wait for 100 ns;
end process;
END;
```

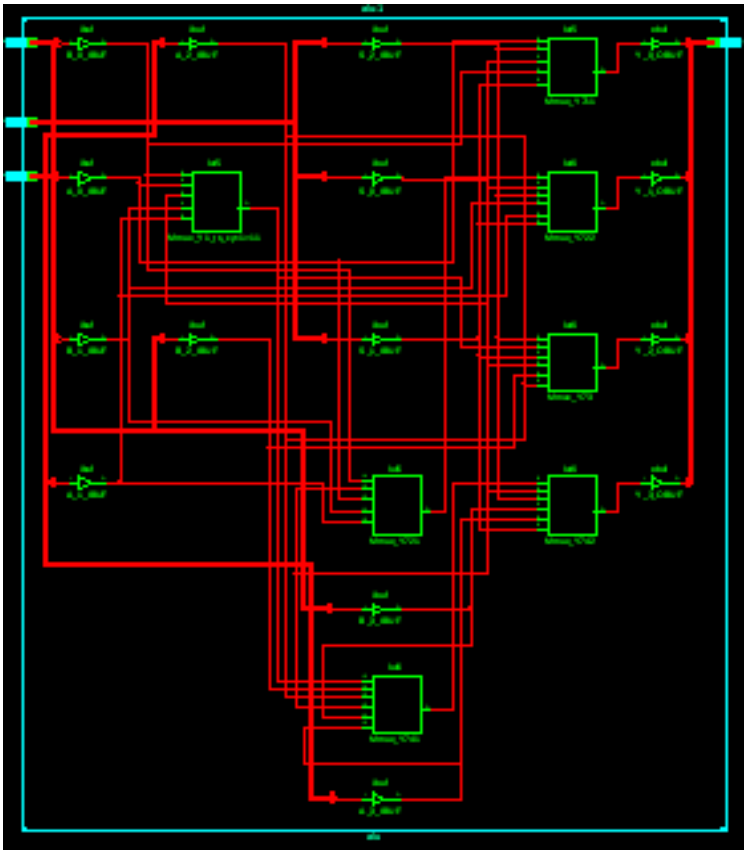
Block Diagram :



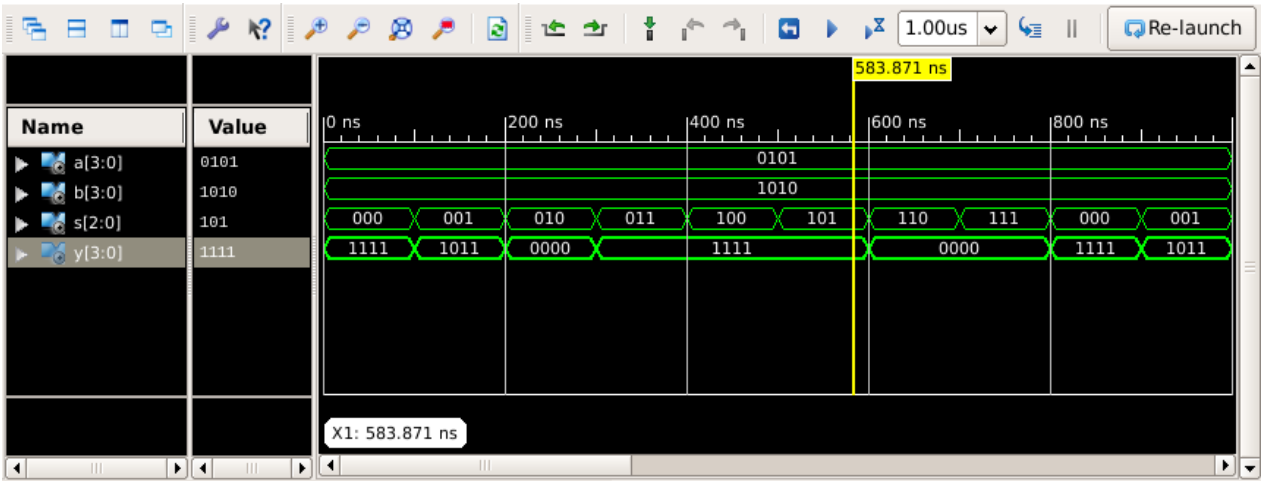
RTL Schematic:



Technology Schematic :



Waveform :



Design Summary :

```
================================================================================
*                            Design Summary                                    *
================================================================================

Top Level Output File Name         : alu.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 7
#      LUT5                        : 3
#      LUT6                        : 4
# IO Buffers                       : 15
#      IBUF                        : 11
#      OBUF                        : 4
```

Conclusion :

Experiment No. 7

Title : Write a VHDL code for Universal shift register with mode selection input for SISO, SIPO, PISO, & PIPO.

Software Used : Xilinx 14.7

Apparatus Required :FPGA Board

Theory :

Shiftregisters,likecounters,areaformofsequentiallogic.Sequentiallogic,unlikecombinational logic is not only affected by the present inputs, but also, by the prior history. In other words, sequential logic remembers past events. Shift registers produce a discrete delay of a digital signal or waveform. A waveform synchronized to a clock, a repeating square wave, is delayed by "n" discrete clock times, where "n" is the number of shift register stages. Thus, a four stage shift registerdelays"datain"byfourclocksto"dataout".Thestagesinashiftregisteraredelaystages, typically type "D" Flip-Flops or type "JK" Flip-flops. Basic shift registers are classified by structure according to the followingtypes:

1) Serial-in/serial-out
2) Parallel-in/serial-out
3) Serial-in/parallel-out
4) Parallel-in/parallel-out

Universal Shift Register: A shift register that can perform with any combination of serial and parallel inputs and outputs (i.e. Serial-in/serial-out, Parallel-in/serial-out, Serial-in/parallel-out, parallel-in/parallel-out). A universal shift register is often a bi-directional as well.

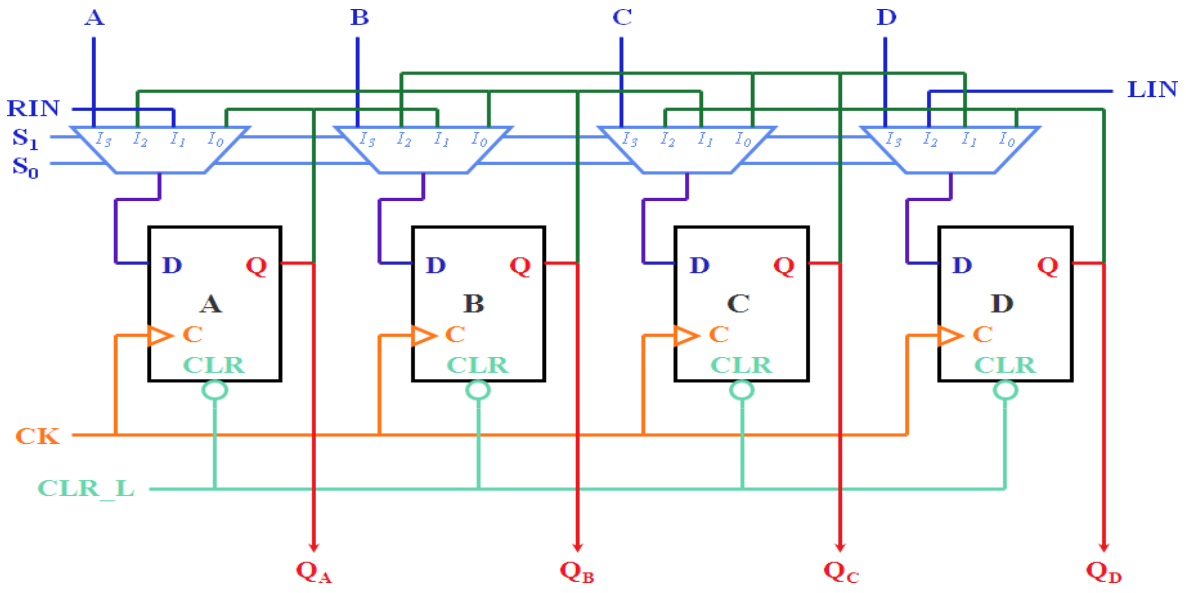


Figure 1: Universal shift register

Serial-in/serial-out shift register: It has a clock input, a data input, and a data output from the last stage. In general, the other stage outputs are not available otherwise; it would be a serial-in, parallel-out shift register. Three **D** Flip-Flops are cascaded Q to D and the clocks paralleled to form a three stage shift register as shown in following figure.

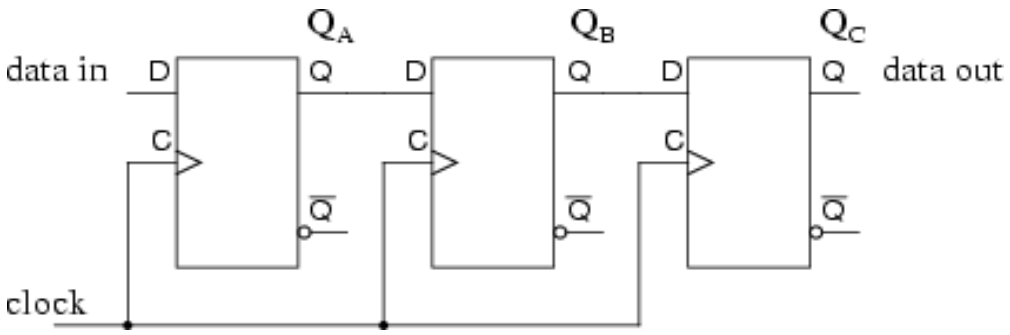


Figure 2: In-Out Shift Register using type "D" Storage Elements

Thewaveformsbelowareapplicabletoserial-in,serial-outshiftregister.Thethreepairsofarrows show that a three-stage shift register temporarily stores 3-bits of data and delays it by three clock periods from input tooutput.

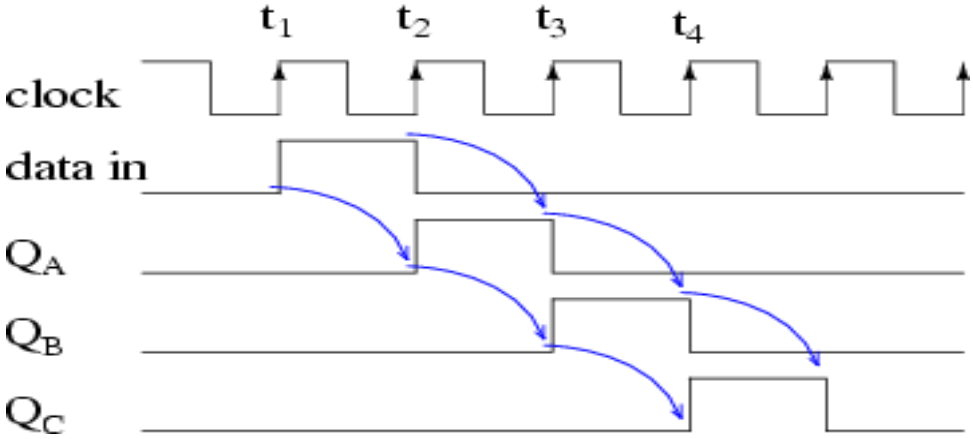
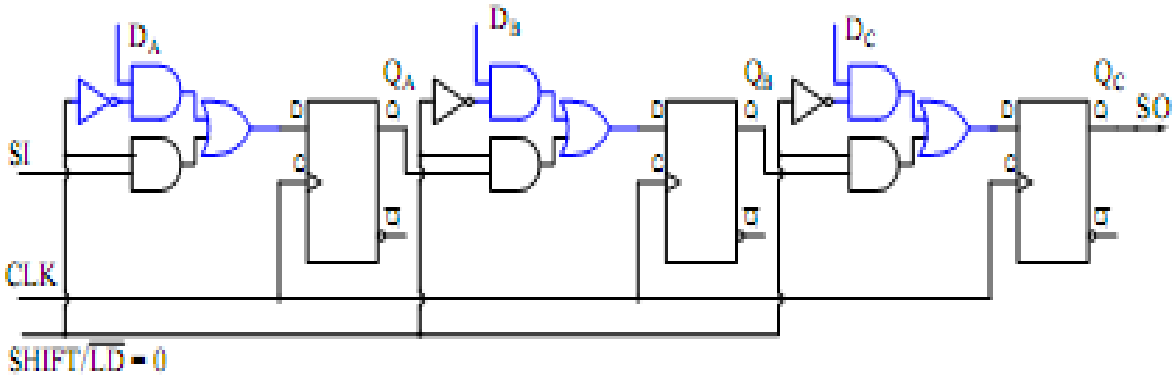


Figure 3: Waveforms

Parallel-in/serial-outshiftregister: Itdoeseverythingthatthepreviousserial-in/serial-outshift registers does plus input data to all stages simultaneously. The parallel-in/ serial-out shift register storesdata,shiftsitonaclockbyclockbasis,anddelaysitbythenumberofstagestimestheclock period. In addition, parallel-in/ serial-out really means that we can load data in parallel



into all stages before any shifting everbegins

Figure 4: Parallel-in, Serial-out Shift Register showing Parallel Load Pattern

In above figure, we show the parallel load path when SHIFT/LD' is logic low. The upper NAND gates serving DA DB DC are enabled, passing data to the D inputs of type D Flip-Flops respectively. At the next positive going clock edge, the data will be clocked from D

to Q of the threeFFs.TheshiftpathisshownabovewhenSHIFT/LD'islogichigh.ThelowerANDgatesof thepairsfeedingtheORgateareenabled givingusashiftregister connectionofSItoDA,QAto DB,QBtoDC,QCtoSO.Clockpulseswillcausedatatoberightshiftedouttosoonsuccessive pulses.Thewaveformsbelowshowbothparallelloadingofthreebitsofdataandserialshifting of this data. Parallel data at DA DB DC is converted to serial data atSO.

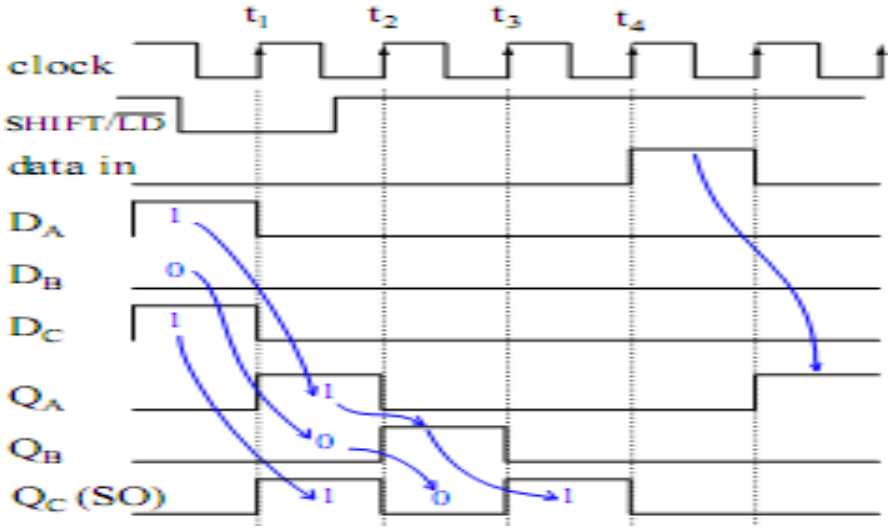


Figure 5: Parallel-in, Serial-out Shift Register Load / Shift Waveforms

Serial-in/parallel-out shift register: The details of the serial-in/parallel-out shift register are fairlysimple. Itlookslikeaserial-in/serial-outshiftregisterwithtapsaddedtoeachstageoutput. Serial data shifts in at **SI** (Serial Input). After a number of clocks equal to the number of stages, the first data bit in appears at SO (QD) in the figure. In general, there is no SO pin. The last stage (QD above) serves as SO and is cascaded to the next package if itexists.

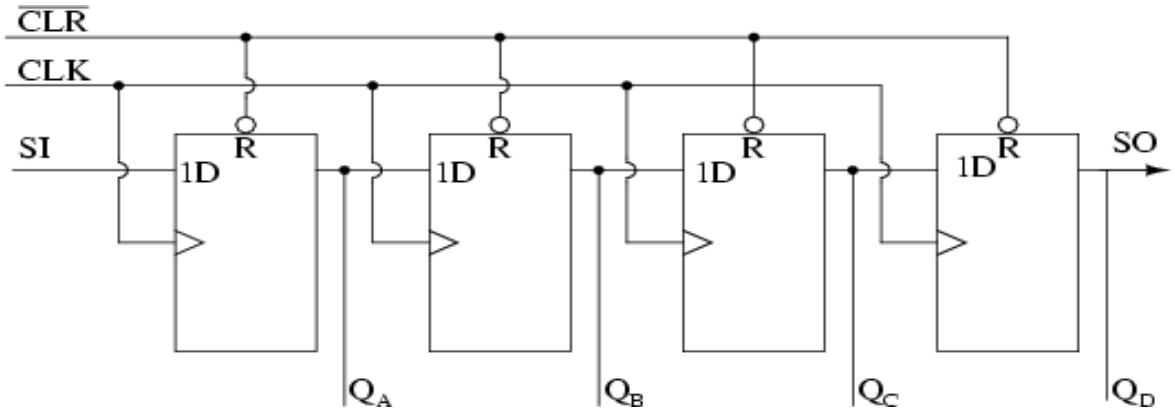


Figure 6: Serial-in, Parallel-out Shift Register Details

The shift register has been cleared prior to any data by CLR', an active low signal, which clears all type D Flip-Flops within the shift register. Following figure shows waveforms for this mode. Note that serial data 1011 pattern is presented at the SI input. This data is synchronized with the clock CLK.

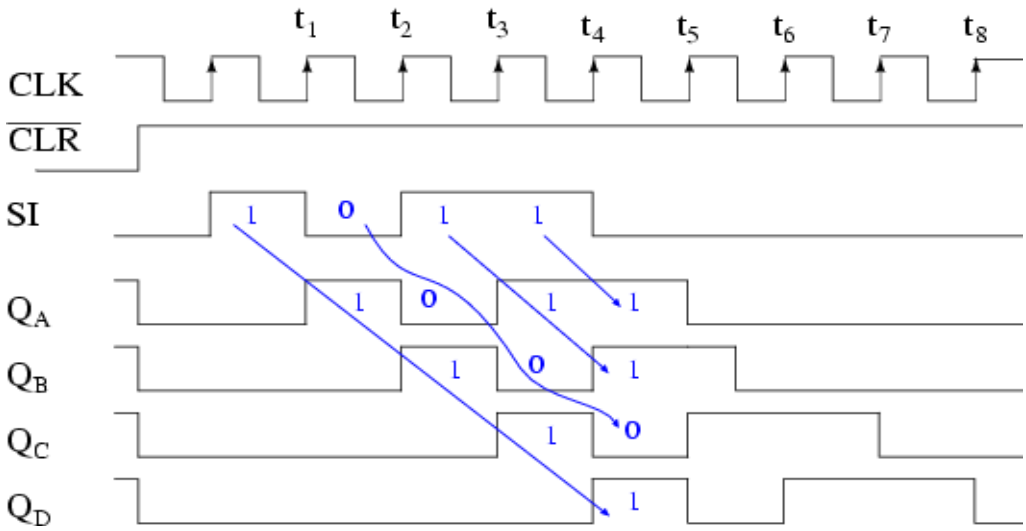


Figure 7: Serial-in, Parallel-out Shift Register Waveform

Parallel-in/parallel-outshiftregister: Theinternaldetailsofarightshiftingparallel-in/parallel- outshiftregisterareshownbelow.Thetri-statebuffersarenotstrictlynecessarytotheparallel-in/ parallel-out shift register, but are part of the real-world device shownbelow,

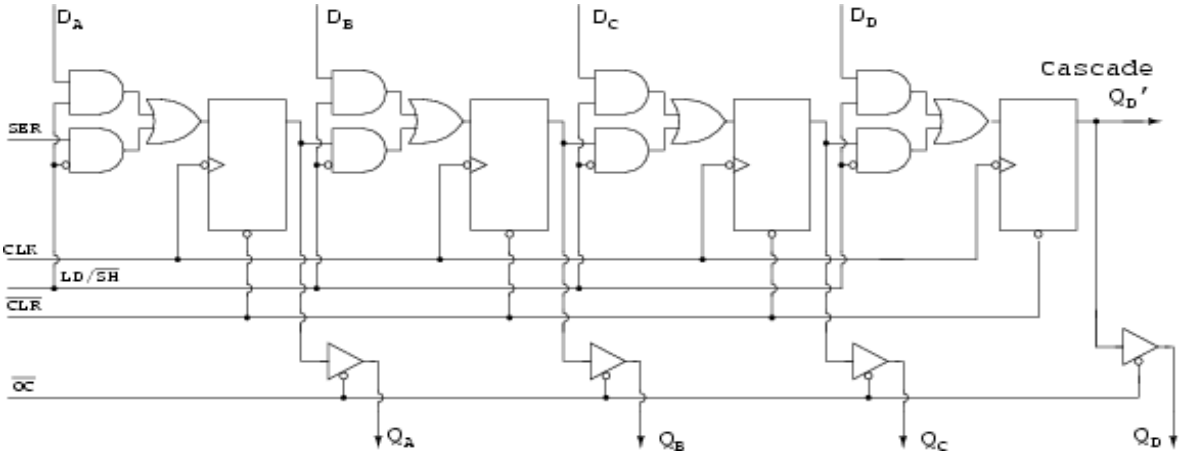


Figure 8: Parallel-in, Parallel-out Shift Register with tri-state Output

If**LD/SH'=1**,theupperfourANDgatesareenabledallowingapplicationofparallelinputs**DAD B DC DD**to the four FF data inputs. The four bits of data will be clocked in parallel from **DA DB DC DD**to **QA QB QC QD** at the next negative going clock. In this "real part", **OC'** must be low if the data needs to be available at the actual output pins as opposed to only on the internalFFs.

Procedure :

1) Create a new Project, select new source as VHDL module and write the VHDL code for the intendeddesign.
2) Perform Check syntax operation to verify that the code is syntacticallycorrect.
3) View RTL schematic and observe the Block diagram and its detailedconnection.
4) Study the synthesis report indetail.
5) Launch ISIM Simulator, force signals to the inputs in signal window, run the simulator and observe the output in Wavewindow.

6) WriteVHDLtestbenchfortheDesign-under-test(DUT)andsimulateittoobservethebehaviour ofdesign.
7) Select the target FPGA device.
8) For the VHDL file, create Implementation constraintfile.
9) In UCF file, assign package pins to the input and output ports of design object list as per the selected target device. (The pin numbers can be referred from the **Matrix II boardmanual**).
10) Implement the design that includes translation, mapping, placement and routing.
11) SelecttheFPGAstartupclockandconfigurationmodeasperttheselectedtarget FPGAdevice (Spartan6).
12) Add Xilinx device, select the appropriate bit file and program theFPGA.
13) ApplytheinputthroughonboardswitchesorinterfacedAdd-onModuleand observetheoutput on the intended output device that may be LED/LCD/SSD/Add-onModule.

Module 1 :Shift Register – SISO, SIPO, PISO, PIPO

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering and Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    10:23:14 10/09/2023
-- Design Name:    Universal Shift Register
-- Module Name:    universal_shift_register1_48 Behaviourial
-- Project Name:   universal_shift_register1_48
-- Target Device:  XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;
use ieee.std_logic_unsigned.all;
entity shift_register is
   Port ( clk : in  STD_LOGIC;
E : in  STD_LOGIC;
rst : in  STD_LOGIC;
pin : in  STD_LOGIC_VECTOR (3 downto 0);
sout : out  STD_LOGIC;
                        sin : in  STD_LOGIC;
pout : out  STD_LOGIC_VECTOR (3 downto 0);
sel : in  STD_LOGIC_VECTOR (1 downto 0));
end shift_register;
architecture shift_register_arch of shift_register is
signal temp:std_logic_vector(3 downto 0);
```

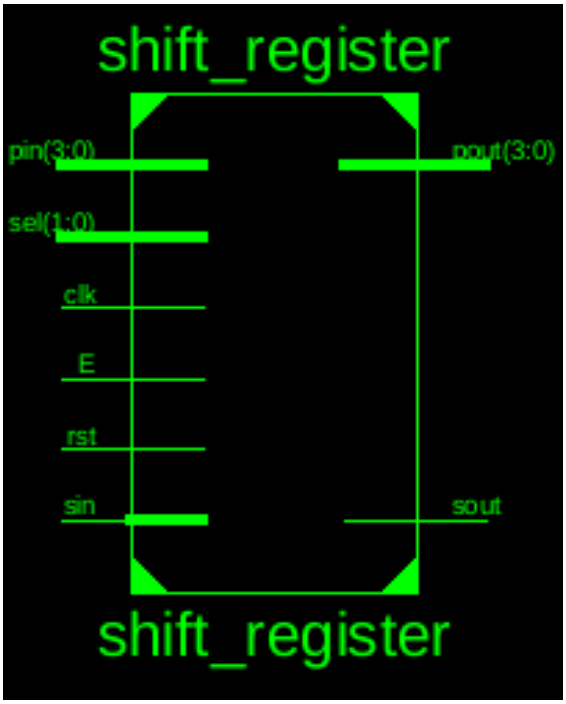
```
begin
process(clk,rst,sel,temp,sin,pin)
begin
if (rst='1') then
        temp<=(others=>'0');
elsif (clk'event and clk='1') then
        case sel is
when "00" =>
        temp <=(temp(2 downto 0) & sin);
        sout<= temp(3);
when "01" =>
        temp<=(temp(2 downto 0) & sin);
        pout<=temp;
when "10" =>
        if (E = '1' ) then
                temp(0) <= pin (0);
                temp(1) <= pin (1);
                temp(2) <= pin (2);
                temp(3) <= pin (3);
        else
                temp (0) <= sin;
                temp (1) <= temp (0);
                temp (2) <= temp (1);
                temp (3) <= temp (2);
                sout<=temp(3);
        end if;
when "11" =>
        temp <= pin;
        pout <= temp;
when others=>
        sout<='0';
        pout<="0000";
end case;
end if;
end process;
end shift_register_arch;
```

Testbench :

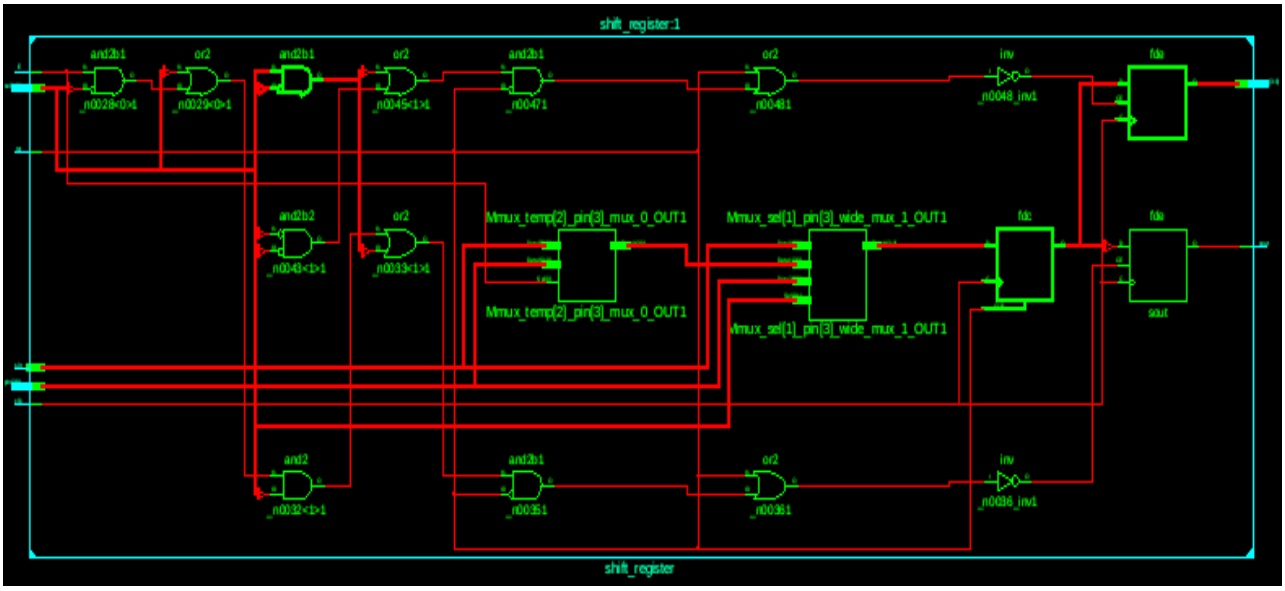
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity shift_register is
   Port ( CLR : in  STD_LOGIC;
SEL : in  STD_LOGIC_VECTOR (1 downto 0);
         SHF_LOAD : in  STD_LOGIC;
```

```
Serial_in : in  STD_LOGIC;
Clk : in  STD_LOGIC;
D : in  STD_LOGIC_VECTOR (3 downto 0);
Q :inout  STD_LOGIC_VECTOR (3 downto 0));
end shift_register;
architecture shift_register_arch of shift_register is
begin
        process(CLK,clr)
begin
if(CLR='0')then
        Q<="0000";
elsif(clk'event and clk='1')then
if(SHF_LOAD='0')then
        Q<=D;
else
if(SEL="00")then
        Q(3)<=Serial_in;
        Q(2)<=Q(3);
        Q(1)<=Q(2);
        Q(0)<=Q(1);
end if;
if(SEL="01")then
        Q(3)<=Serial_in;
        Q(1)<=Q(0);
        Q(2)<=Q(1);
        Q(3)<=Q(2);
end if;
if(SEL="10")then
        Q<=D(2downto 0)&D(3);
end if;
if(SEL="11")then
        Q<=D(0)&D(3downto 1);
end if;
end if;
end if;
end process;
end shift_register_arch;
```

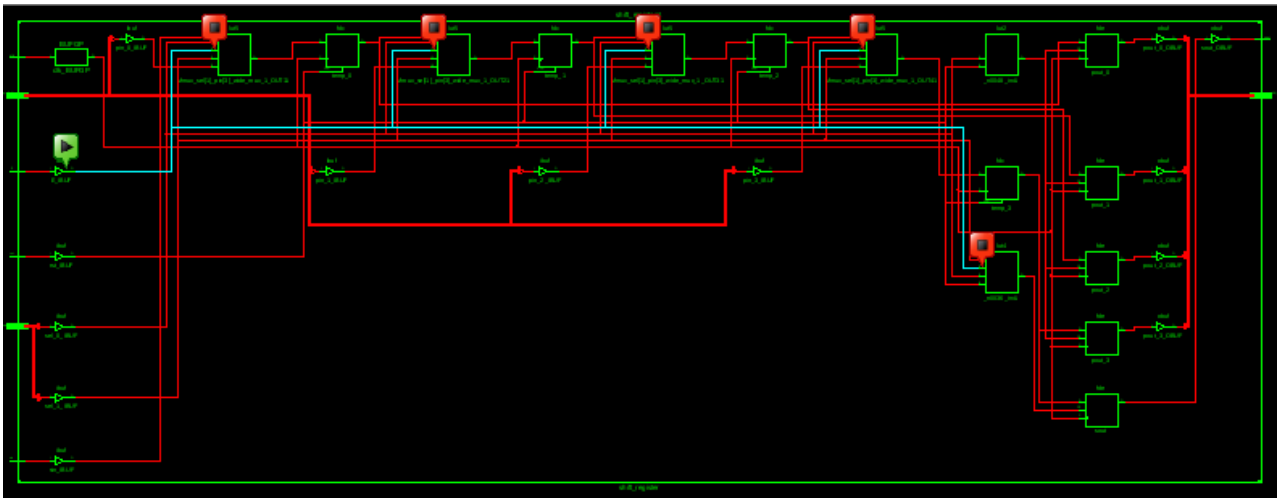
Block Diagram :



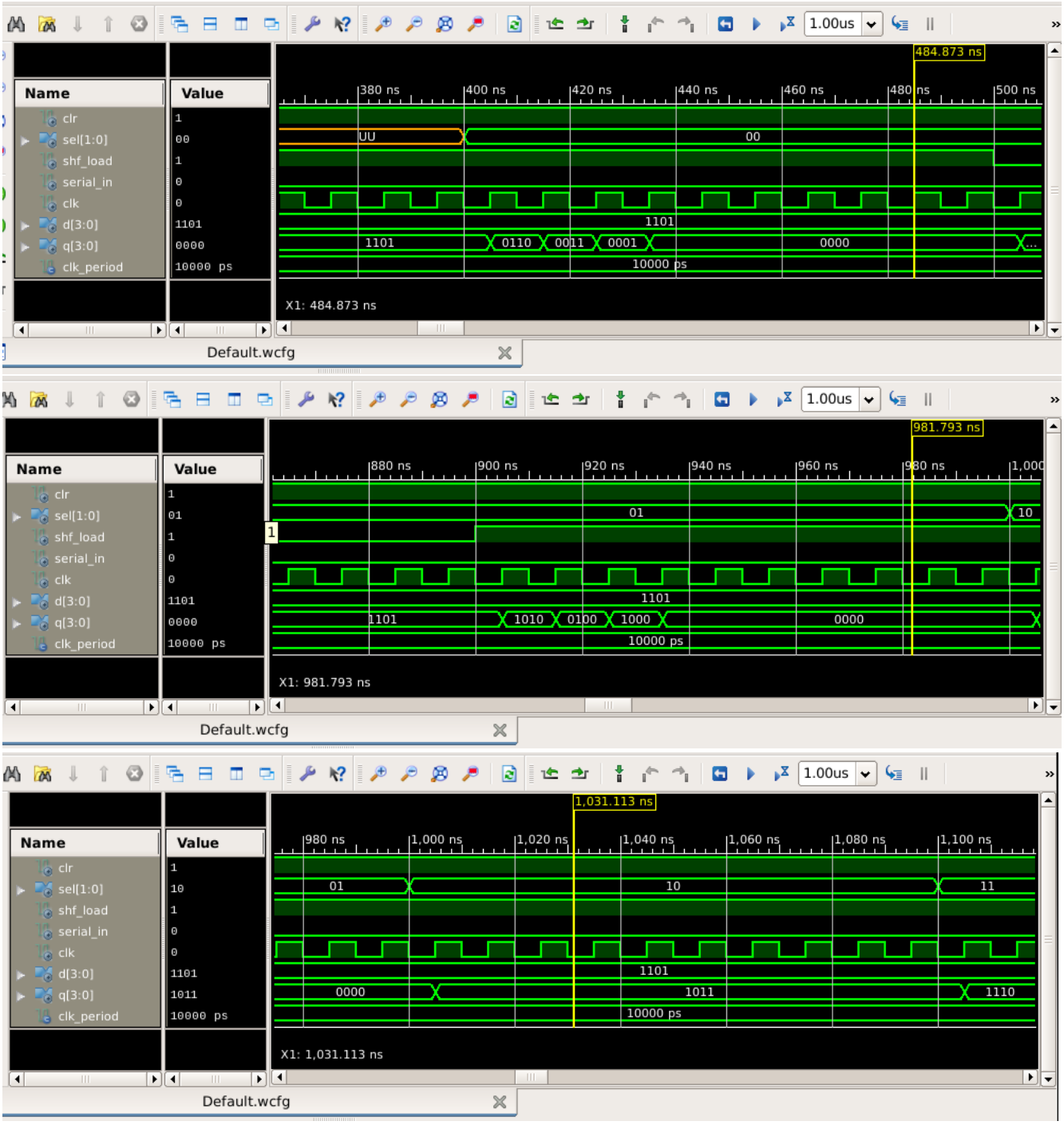
RTL Schematic:

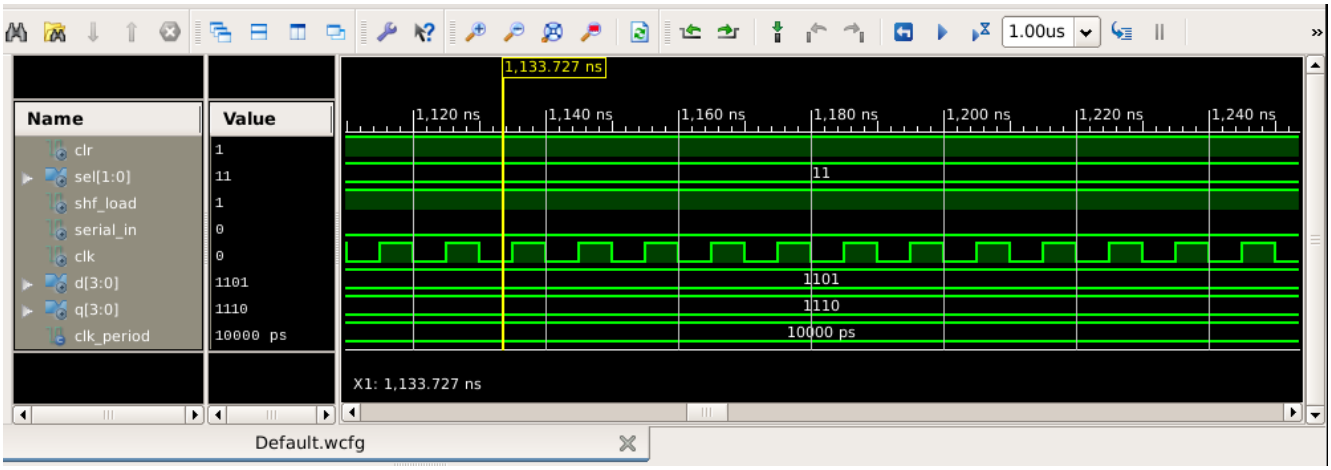


Technology Schematic :



Waveform :





Design Summary :

```
================================================================================
*                              Design Summary                                  *
================================================================================

Top Level Output File Name         : shift_register.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 6
#      LUT2                        : 1
#      LUT4                        : 1
#      LUT5                        : 4
# FlipFlops/Latches                : 9
#      FDC                         : 4
#      FDE                         : 5
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 14
#      IBUF                        : 9
#      OBUF                        : 5
```

Conclusion :

Experiment No. 8

Title : To write VHDL code, simulate with test bench, synthesis, implement on PLD for Mod - N Counter.

Software Used : Xilinx 14.7

Apparatus Required : FPGA Board

Theory :

A mod-N counter is a digital counter circuit that counts from 0 to N-1 and then wraps around to 0. It can be implemented using both up counters and down counters.

UP Counter –

An up counter is a digital counter that counts upwards from 0 to its maximum value (N-1) and then resets to 0. To implement a modulo-N up counter, you can use a binary counter with N flip-flops.

Down Counter –

A down counter is a digital counter that counts downwards from its maximum value (N-1) to 0 and then wraps around to the maximum value. To implement a modulo-N down counter, you can use a similar approach as the up counter, but you decrement the count with each clock pulse instead of incrementing it.

Diagram -

Procedure :

1) Create a new Project, select new source as VHDL module and write the VHDL code for the intended design.
2) Perform Check syntax operation to verify that the code is syntactically correct.
3) View RTL schematic and observe the Block diagram and its detailed connection.
4) Study the synthesis report in detail.
5) Launch ISIM Simulator, force signals to the inputs in signal window, run the simulator and observe the output in Wave window.
6) Write VHDL testbench for the Design-under-test (DUT) and simulate it to observe the behavior of design.
7) Select the target FPGA device.
8) For the VHDL file, create Implementation constraint file.
9) In UCF file, assign package pins to the input and output ports of design object list as per the selected target device.(The pin numbers can be referred from the Matrix II board manual.
10) Implement the design which includes translation, mapping, placement and routing.
11) Select the FPGA startup clock and configuration mode as per the selected target FPGA device.
12) Add Xilinx device, select the appropriate bit file and program the FPGA.

13) Apply the input through onboard switches or interfaced Add-on Module and observe the output on the intended output device which may be LED/LCD/SSD/Add-on Module.

Module 1 : UP Counter

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer: Tanmay Vilas Pawar

-- Create Date:    11:27:26 11/04/2023
-- Design Name:    Counter
-- Module Name:    mod_n_counter_48 - Behavioral
-- Project Name:   Mod-N Counter
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

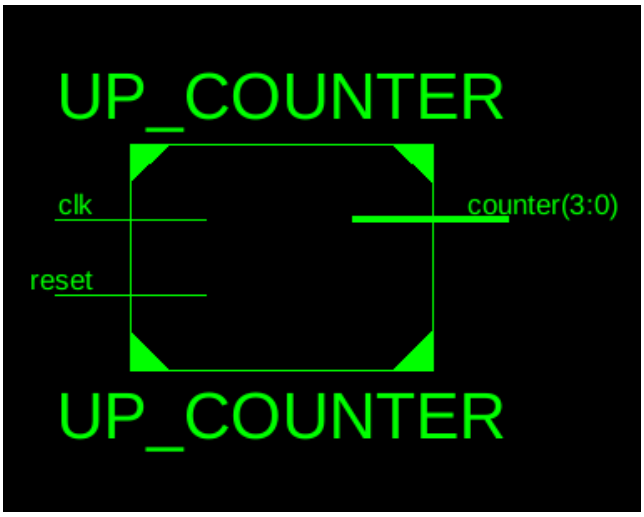
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity UP_COUNTER is
   Port ( clk: in std_logic; -- clock input
          reset: in std_logic; -- reset input
          counter: out std_logic_vector(3 downto 0) -- output 4-bit counter
    );
end UP_COUNTER;
architecture Behavioral of UP_COUNTER is
signal counter_up: std_logic_vector(3 downto 0);
begin
-- up counter
process(clk)
begin
if(rising_edge(clk)) then
   if(reset='1') then
       counter_up <= x"0";
   else
      counter_up <= counter_up + x"1";
   end if;
 end if;
end process;
 counter <= counter_up;
end Behavioral;
```

TestBench :

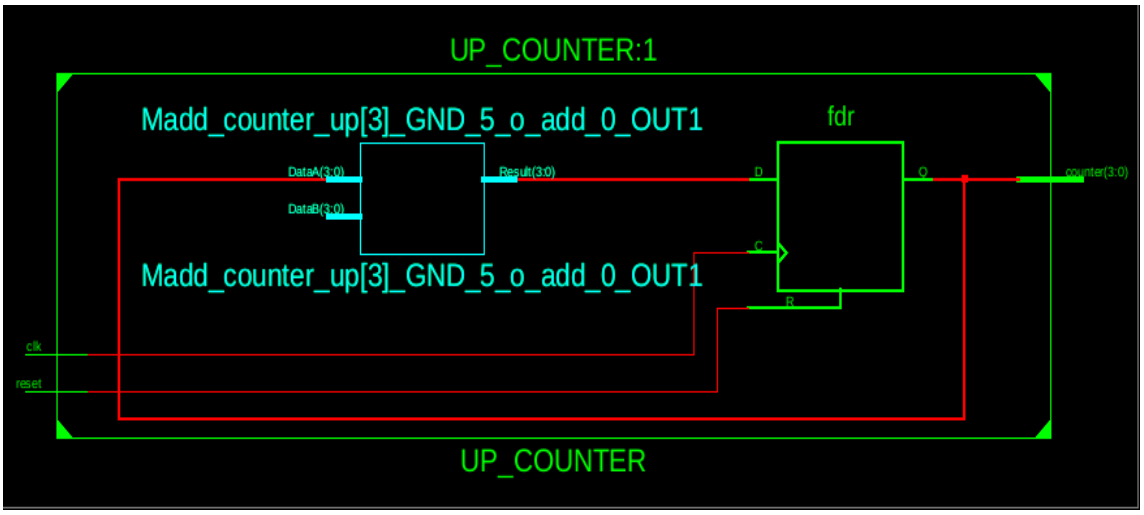
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity tb_counters is
end tb_counters;
architecture Behavioral of tb_counters is
```

```
component UP_COUNTER
   Port ( clk: in std_logic; -- clock input
          reset: in std_logic; -- reset input
          counter: out std_logic_vector(3 downto 0) -- output 4-bit counter
    );
end component;
signal reset,clk: std_logic;
signal counter:std_logic_vector(3 downto 0);
begin
dut: UP_COUNTER port map (clk => clk, reset=>reset, counter => counter);
  -- Clock process definitions
clock_process :process
begin
    clk <= '0';
    wait for 10 ns;
    clk <= '1';
    wait for 10 ns;
end process;
-- Stimulus process
stim_proc: process
begin
  -- hold reset state for 100 ns.
    reset <= '1';
  wait for 20 ns;
   reset <= '0';
  wait;
end process;
end Behavioral;
```

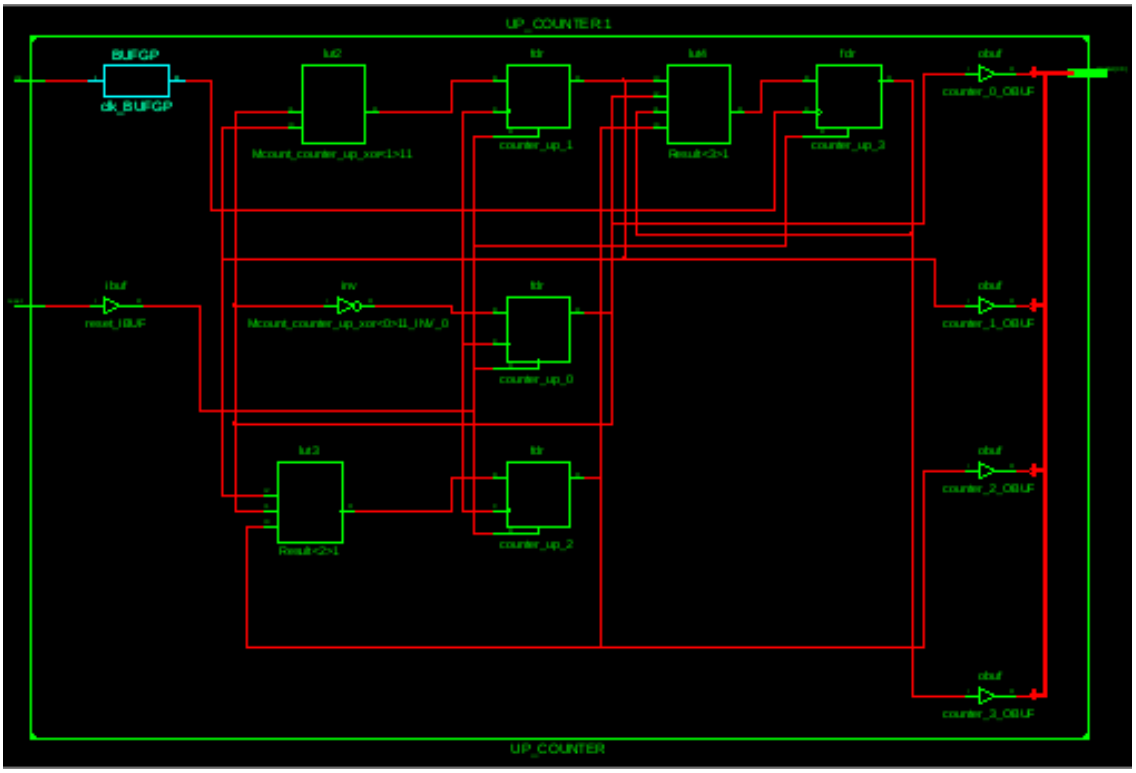
Block Diagram :



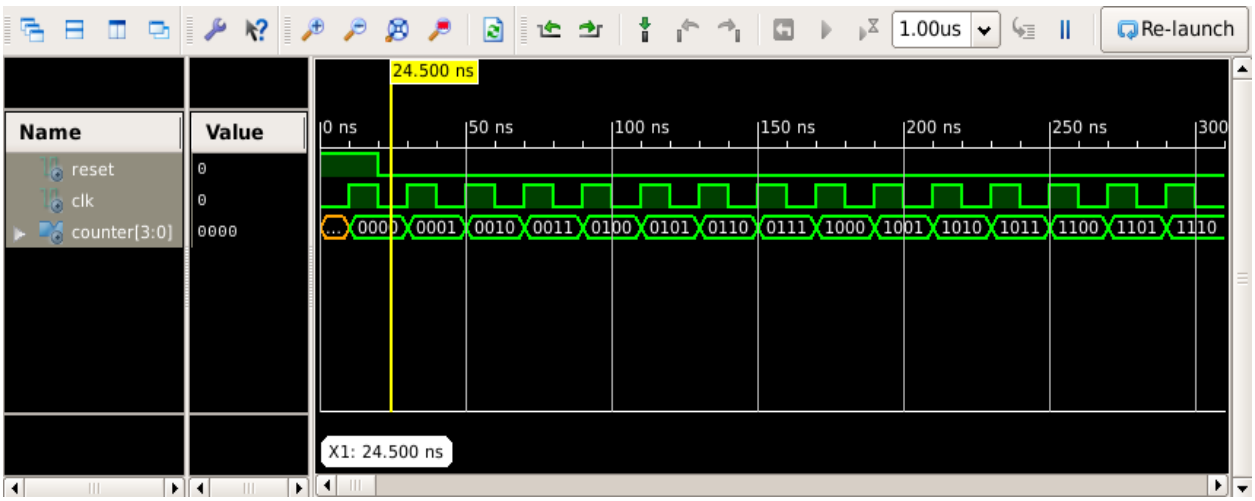
RTL Schematic :

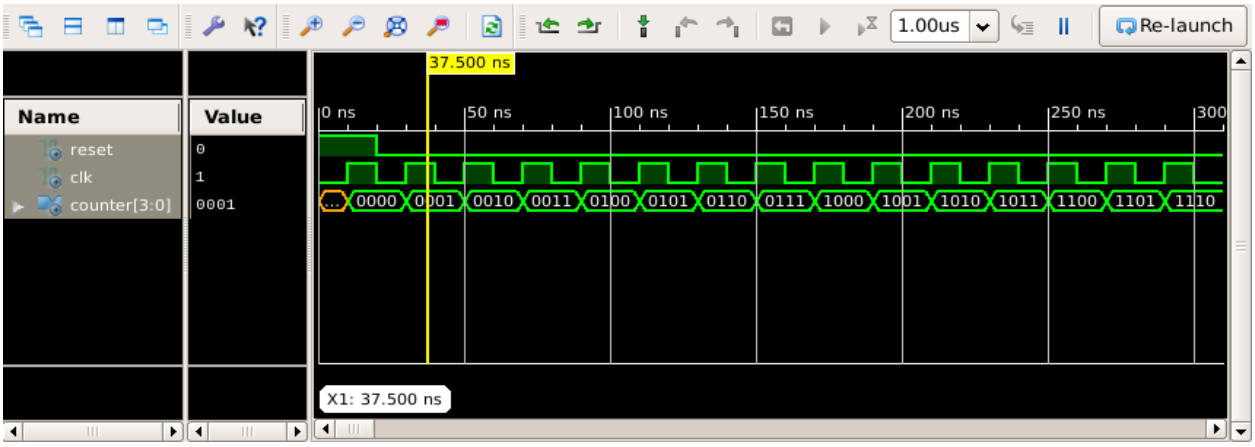


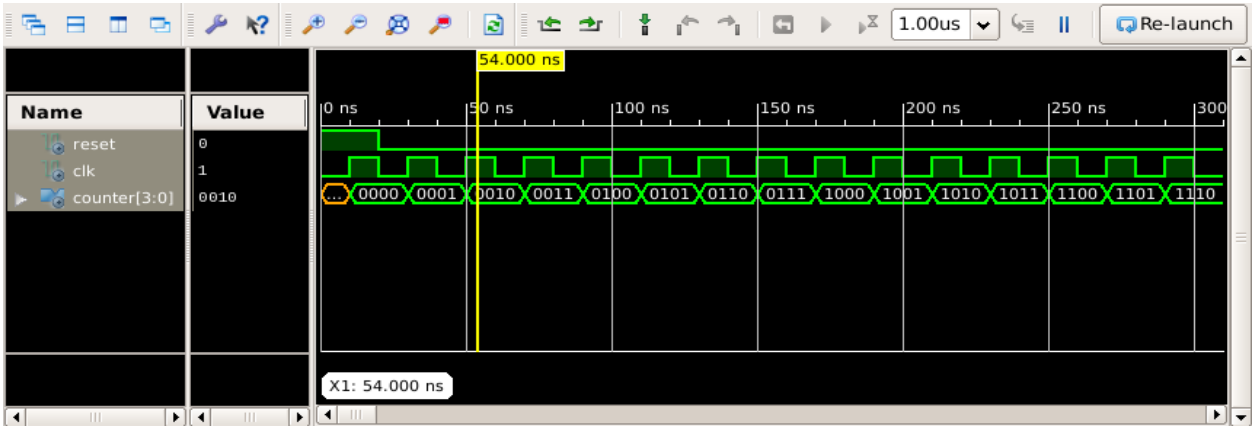
Technology Schematic :

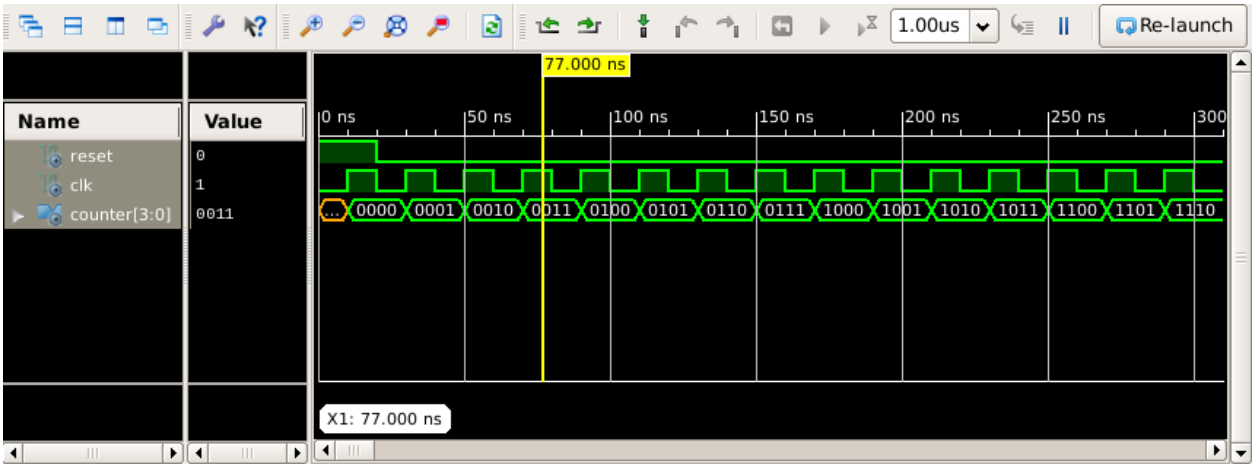


Waveform :









Design Summary :

```
================================================================================
*                           Design Summary                                     *
================================================================================

Top Level Output File Name         : UP_COUNTER.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 4
#      INV                         : 1
#      LUT2                        : 1
#      LUT3                        : 1
#      LUT4                        : 1
# FlipFlops/Latches                : 4
#      FDR                         : 4
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 5
#      IBUF                        : 1
#      OBUF                        : 4
```

Module 2 : DOWN Counter

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer: Tanmay Vilas Pawar

-- Create Date:    11:27:26 11/04/2023
-- Design Name:    Counter
-- Module Name:    mod_n_counter_48 - Behavioral
-- Project Name:   Mod-N Counter
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

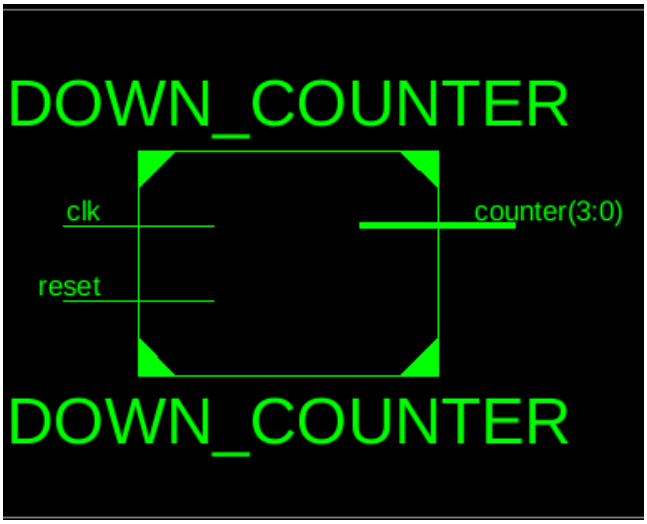
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity DOWN_COUNTER is
   Port ( clk: in std_logic; -- clock input
          reset: in std_logic; -- reset input
          counter: out std_logic_vector(3 downto 0) -- output 4-bit counter
    );
end DOWN_COUNTER;
architecture Behavioral of DOWN_COUNTER is
signal counter_down: std_logic_vector(3 downto 0);
begin
-- down counter
process(clk)
begin
if(rising_edge(clk)) then
   if(reset='1') then
       counter_down <= x"F";
   else
      counter_down <= counter_down - x"1";
   end if;
 end if;
end process;
 counter <= counter_down;
end Behavioral;
```

TestBench :

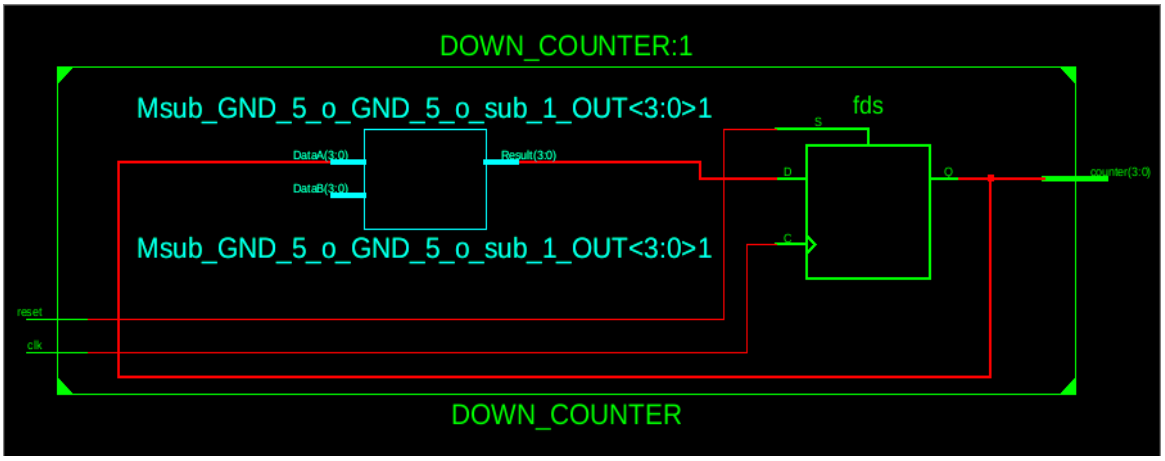
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity tb_counters is
end tb_counters;
architecture Behavioral of tb_counters is
component DOWN_COUNTER
   Port ( clk: in std_logic; -- clock input
          reset: in std_logic; -- reset input
```

```
        counter: out std_logic_vector(3 downto 0) -- output 4-bit counter
    );
end component;
signal reset,clk: std_logic;
signal counter:std_logic_vector(3 downto 0);
begin
dut: DOWN_COUNTER port map (clk => clk, reset=>reset, counter => counter);
  -- Clock process definitions
clock_process :process
begin
    clk <= '0';
    wait for 10 ns;
    clk <= '1';
    wait for 10 ns;
end process;
-- Stimulus process
stim_proc: process
begin
  -- hold reset state for 100 ns.
    reset <= '1';
  wait for 20 ns;
   reset <= '0';
  wait;
end process;
end Behavioral;
```

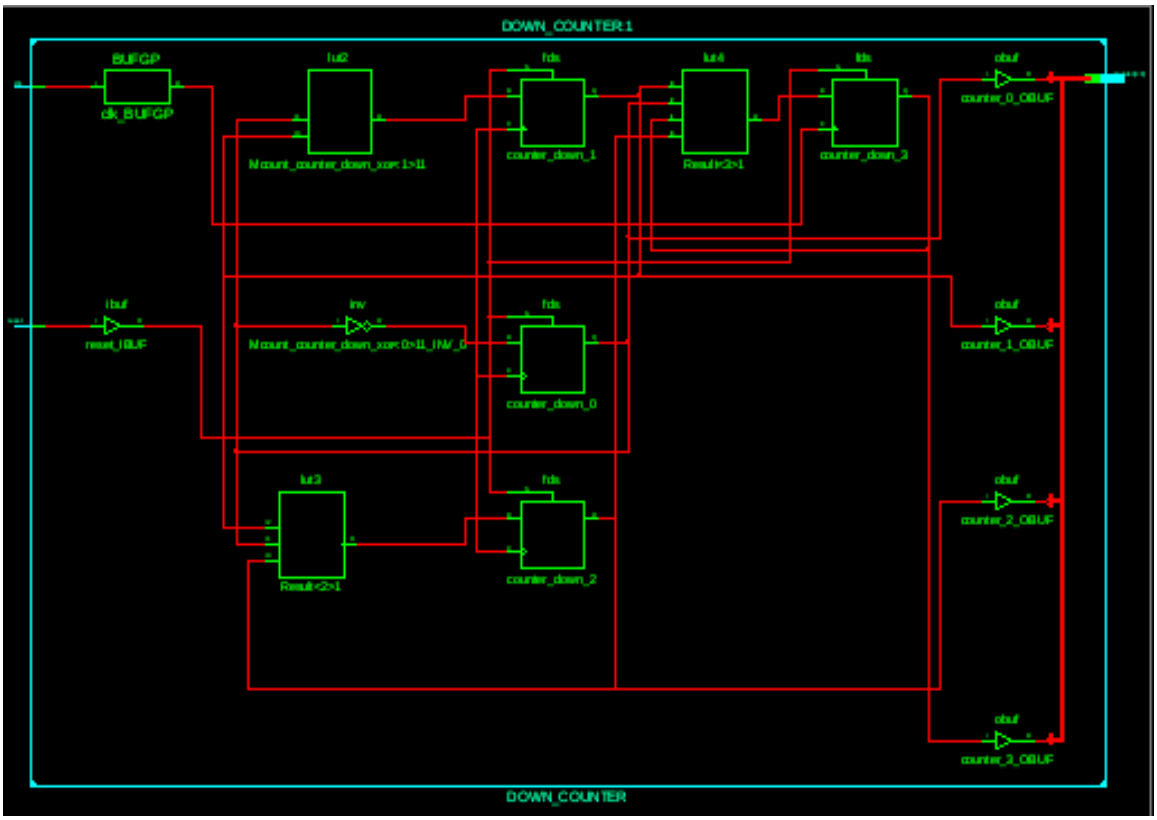
Block Diagram :



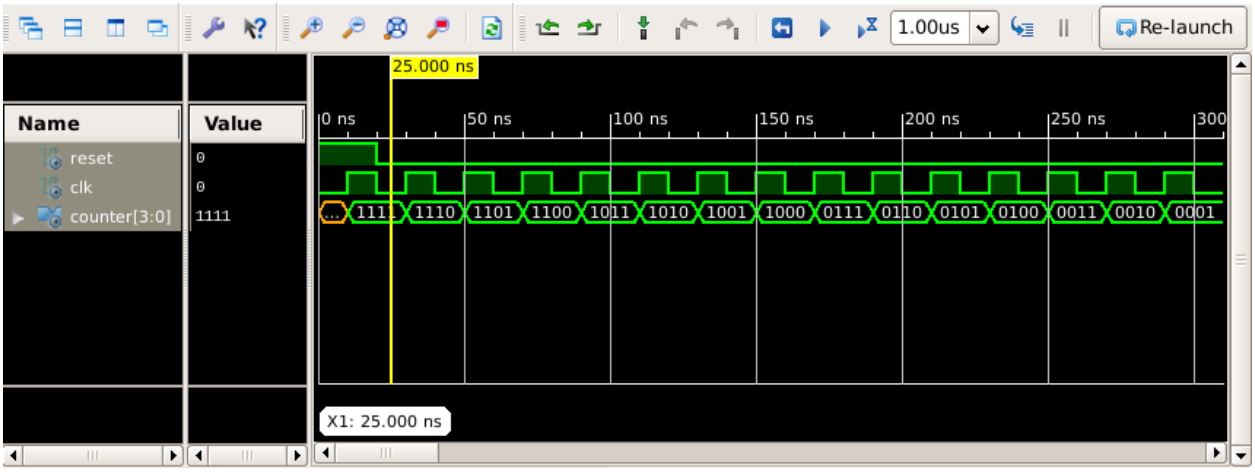
RTL Schematic :

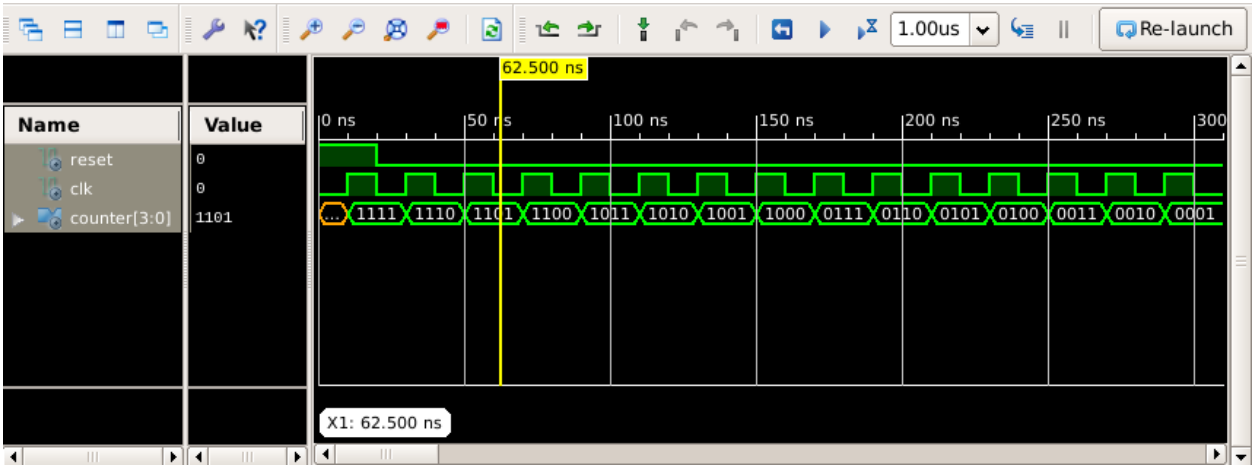


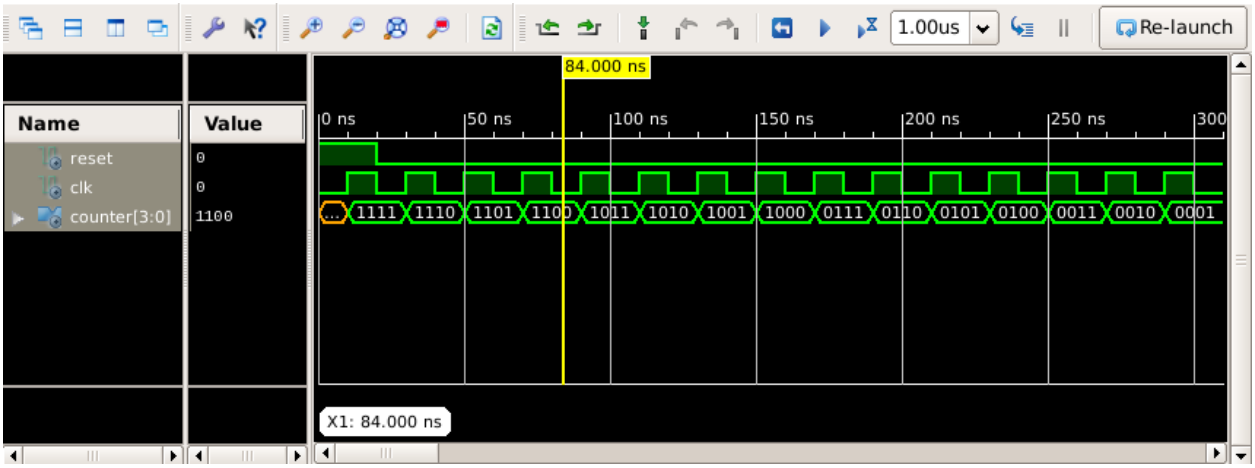
Technology Schematic :



Waveform :







Design Summary :

```
================================================================================
*                            Design Summary                                    *
================================================================================

Top Level Output File Name         : DOWN_COUNTER.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 4
#      INV                         : 1
#      LUT2                        : 1
#      LUT3                        : 1
#      LUT4                        : 1
# FlipFlops/Latches                : 4
#      FDS                         : 4
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 5
#      IBUF                        : 1
#      OBUF                        : 4
```

Conclusion :

Experient No. 9

Title : To write VHDL code, simulate, synthesis, implement for LCD Interfacing.

Software Used : Xilinx 14.7

Apparatus Required : FPGA Board, Spartan 6

Theory :
LCD is a practical way to display a variety of information using a standard ASCII and characters. However this display is not fast scrolling the display at half adder interval test the practical limit of clarity.
ASCII Codes for LCD Interface :

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Command to LCD instruction Register :

Code (hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Shift cursor to left
5	Shift display right
6	Shift cursor to right
7	Shift display left
8	Display off, Cursor off
A	Display off, Cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
38	2 lines and 5x7 matrix

Procedure :

1) Create a new Project, select new source as VHDL module and write the VHDL code for the intended design.
2) Perform Check syntax operation to verify that the code is syntactically correct.
3) View RTL schematic and observe the Block diagram and its detailed connection.
4) Study the synthesis report in detail.
5) Launch ISIM Simulator, force signals to the inputs in signal window, run the simulator and observe the output in Wave window.
6) Write VHDL testbench for the Design-under-test (DUT) and simulate it to observe the behavior of design.
7) Select the target FPGA device.
8) For the VHDL file, create Implementation constraint file.
9) In UCF file, assign package pins to the input and output ports of design object list as per the selected target device.(The pin numbers can be referred from the Matrix II board manual.
10) Implement the design which includes translation, mapping, placement and routing.
11) Select the FPGA startup clock and configuration mode as per the selected target FPGA device.
12) Add Xilinx device, select the appropriate bit file and program the FPGA.
13) Apply the input through onboard switches or interfaced Add-on Module and observe the output on the intended output device which may be LED/LCD/SSD/Add-on Module.

Module 1 : LCD Interfacing

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    02:59:46 10/11/2023
-- Design Name:    LCD
-- Module Name:    lcd_interface_48 - Behavioral
-- Project Name:   LCD Interface
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;
entity LCD_interface is
   Port ( rst : in  STD_LOGIC;
          clk : in  STD_LOGIC;
          RS : out  STD_LOGIC;
          RW : out  STD_LOGIC;
          EN : out  STD_LOGIC;
          LCD : out  STD_LOGIC_VECTOR(7 downto 0));
end LCD_interface;
```

```
architecture arch_lcd_interface of LCD_interface is
signal count: integer range 0 to 51;
signal clock: std_logic;
signal flag : std_logic:='0';
begin
process(rst,clk)
variable temp: integer range 0 to 999999;
begin
if(not rst='1')then
  temp:=0;
  clock<='0';
elsif(clk='1' and clk' event)then
  temp:=temp+1;
  if(temp =250000)then
    clock<= not clock;
    temp:=0;
 end if;
end if;
end process;
process (rst,clock)
begin
 if (not rst ='1') then
 COUNT <=0;
 elsif (clock' event and clock = '1') then
if (count < 51 and flag='0' ) then
count <= count+ 1;
end if;
end if;
END PROCESS;
Process(count, rst, clock)
begin
if (NOT rst ='1') then
RW<='0';
RS<='0';
EN<='0';
LCD <= "00000000";
flag<='0';
elsif(rising_edge(clock)) then
case count is
when 0 => RW <= '0';
RS <= '0';
LCD <= x"38"; -- init in 2 lin mode
EN <= '1';
when 1 => EN <= '0';
when 2 => RW <= '0';
RS <= '0';
```

```
LCD <= x"0E"; ------- display on, cursor blink
EN <= '1';
when 3 => EN <= '0';
when 4 => RW <= '0';
RS <= '0';
LCD <= x"01"; ------- clear display
EN <= '1';
when 5 => EN <= '0';
when 6 => RW <= '0';
RS <= '0';
LCD <= x"06"; -------- shift cursor to right
EN <= '1';
when 7 => EN <= '0';
when 8=> RW <= '0';
RS <= '0';
LCD <= x"80"; --------- initialize to 1st line, 2nd position
EN <= '1';
when 9=> EN <= '0';
when 10 => RW <= '0';
RS <= '1';
LCD <=x"57"; -------- W
EN <= '1';
when 11 => EN <= '0';
when 12 => RW <= '0';
RS<= '1';
LCD<= x"45"; -------- E
EN <= '1';
when 13 => EN <= '0';
when 14 => RW <= '0';
RS<= '1';
LCD<= x"4C";  -------- L
EN <= '1';
when 15 => EN <= '0';
when 16 => RW <= '0';
RS<= '1';
LCD<= x"43"; -------- C
EN <= '1';
when 17 => EN <= '0';
when 18 => RW <= '0';
RS<= '1';
LCD<= x"4F"; -------- O
EN <= '1';
when 19 => EN <= '0';
when 20 => RW <= '0';
RS<= '1';
LCD<= x"4D"; -------- M
```

```
EN <= '1';
when 21 => EN <= '0';
when 22 => RW <= '0';
RS<= '1';
LCD<= x"45"; -------- E
EN <= '1';
when 23 => EN <= '0';
when 24 => RW <= '0';
RS<= '1';
LCD<= x"20"; -------- space
EN <= '1';
when 25 => EN <= '0';
when 26 => RW <= '0';
RS<= '1';
LCD<= x"20"; -------- space
EN <= '1';
when 27 => EN <= '0';
when 28 => RW <= '0';
RS<= '1';
LCD<= x"54"; -------- T
EN <= '1';
when 29=> EN <= '0';
when 30 => RW <= '0';
RS<= '1';
LCD<= x"4F"; -------- O
EN <= '1';
when 31=> EN <= '0';
when 32 => RW <= '0';
RS<= '0';
LCD<= x"C6"; -------- second line(0xC5)
EN <= '1';
when 33=> EN <= '0';
when 34 => RW <= '0';
RS<= '1';
LCD<= x"56"; --------- V
EN <= '1';
when 35=> EN <= '0';
when 36 => RW <= '0';
RS<= '1';
LCD<= x"4C"; ---------- L
EN <= '1';
when 37 => EN <= '0';
when 38 => RW <= '0';
RS<= '1';
LCD<= x"53"; --------- S
EN <= '1';
```

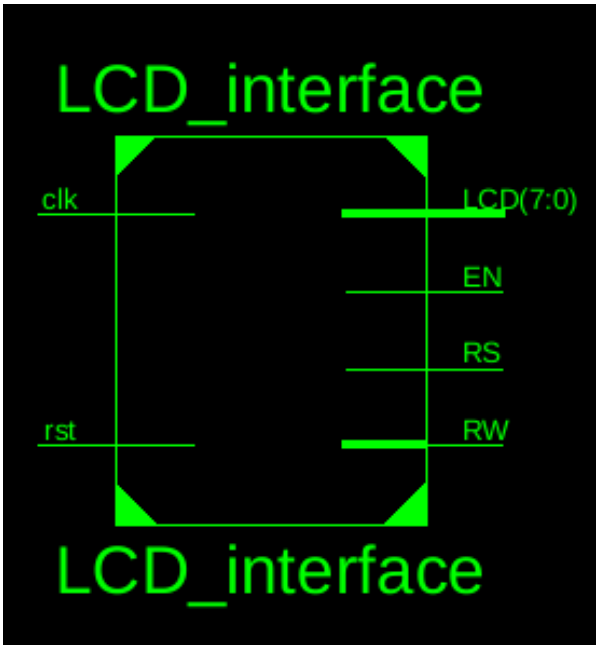
```
when 39=> EN <= '0';
when 40 => RW <= '0';
RS<= '1';
LCD<= x"49"; ---------- I
EN <= '1';
when 41 => EN <= '0';
when 42 => RW <= '0';
RS<= '1';
LCD<= x"20"; --------- space
EN <= '1';
when 43 => EN <= '0';
when 44 => RW <= '0';
RS<= '1';
LCD<= x"4C"; --------- L
EN <= '1';
when 45 => EN <= '0';
when 46 => RW <= '0';
RS<= '1';
LCD<= x"41"; ---------- A
EN <= '1';
when 47 => EN <= '0';
when 48 => RW <= '0';
RS<= '1';
LCD<= x"42"; ---------- B
EN <= '1';
when 49 => EN <= '0';
when 50 => RW <= '0';
RS<= '1';
LCD<= x"20"; -------- space
EN <= '1';
when 51 => EN <= '0';
flag<='1';
END CASE;
end if;
END PROCESS;
end arch_lcd_interface;
```

LCD_interfce.ucf File :

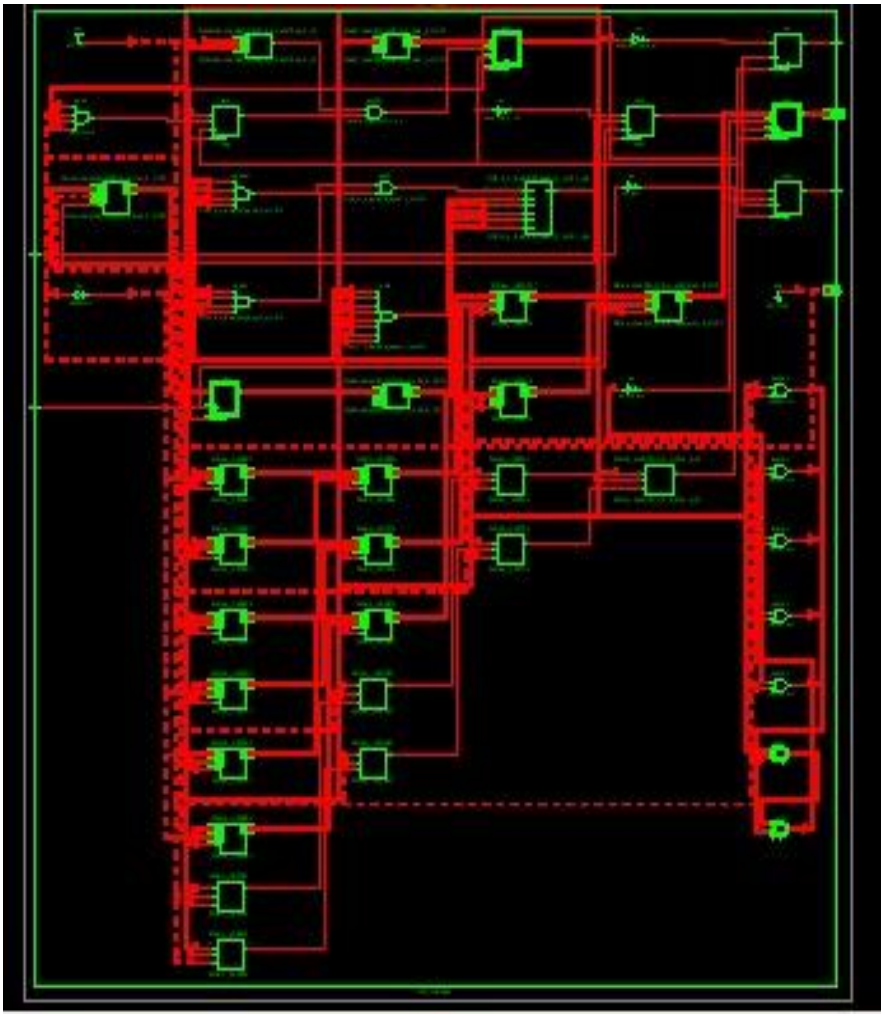
```
NET "clk" LOC="P56";#BANK = 0,Signal name=MCLK
NET "rst" LOC="P22";#BANK = 2,Signal name=RESET
NET "rst" CLOCK_DEDICATED_ROUTE =FALSE;
NET "LCD[0]"LOC = "P85";
NET "LCD[1]"LOC = "P82";
NET "LCD[2]"LOC = "P83";
NET "LCD[3]"LOC = "P80";
NET "LCD[4]"LOC = "P81";
NET "LCD[5]"LOC = "P78";
```

```
NET "LCD[6]"LOC = "P79";
NET "LCD[7]"LOC = "P74";
NET "RS" LOC="P88";
NET "EN" LOC="P84";
NET "RW" LOC="P2";
```

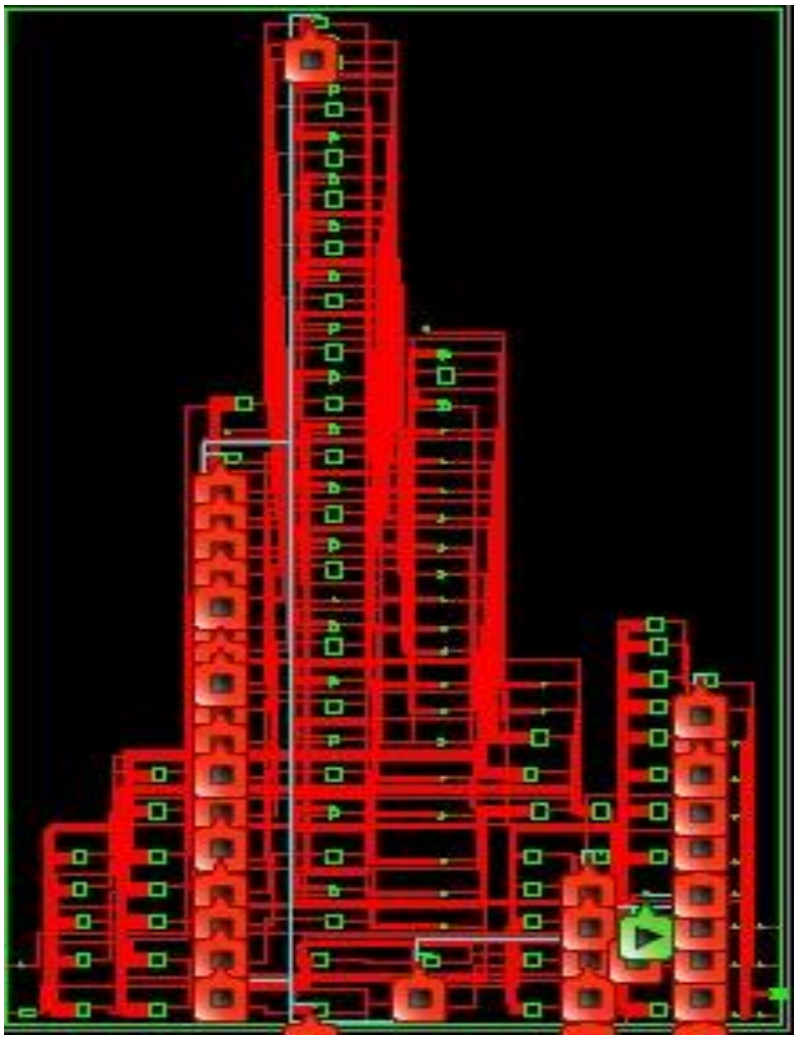
Block Diagram :



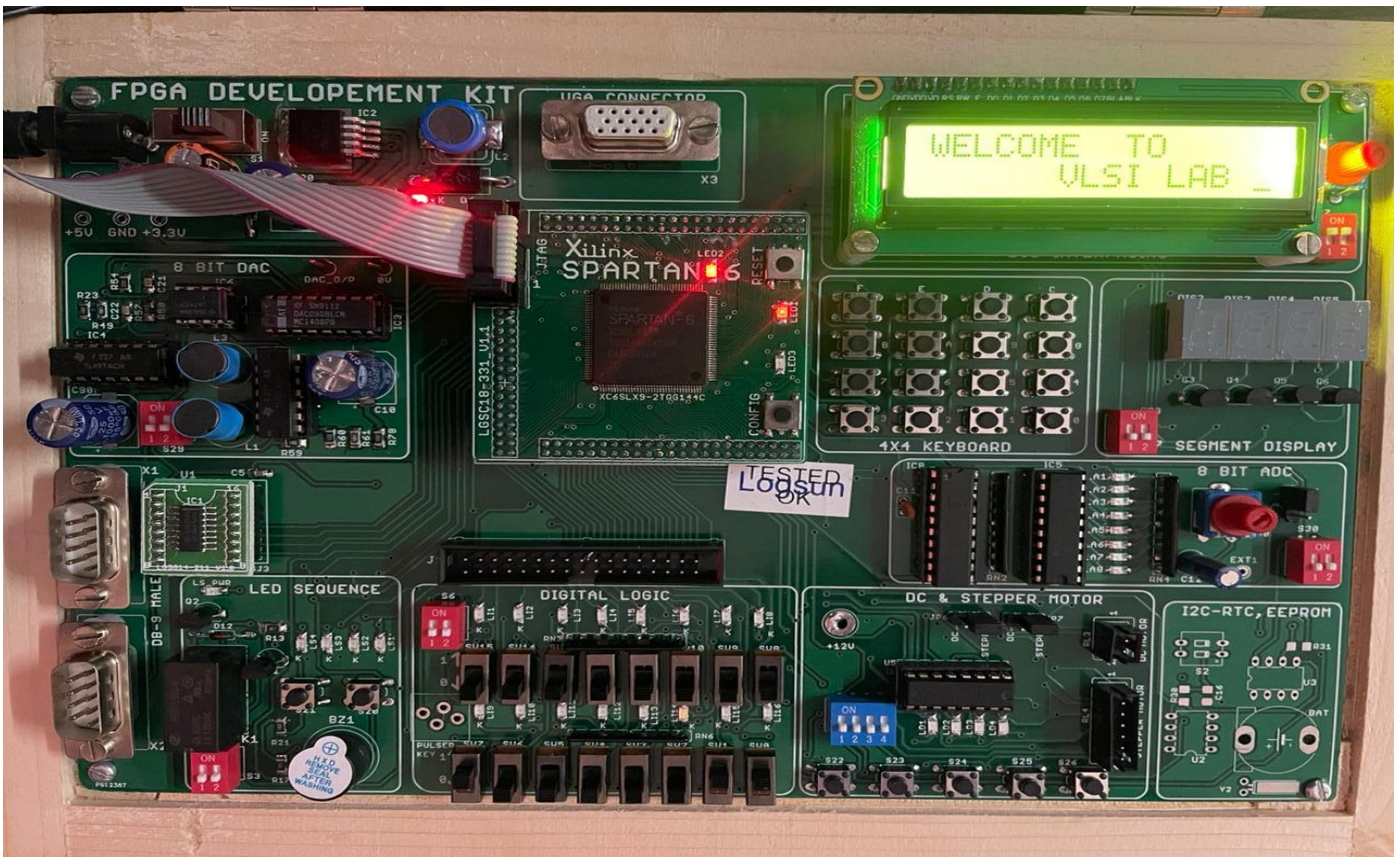
RTL Schematic :



Technology Schematic :



Output :



Design Summary :

```
=========================================================
*                            Design Summary
=========================================================

Top Level Output File Name         : LCD_interface.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 93
#      GND                         : 1
#      INV                         : 3
#      LUT1                        : 17
#      LUT2                        : 2
#      LUT3                        : 1
#      LUT4                        : 2
#      LUT5                        : 16
#      LUT6                        : 13
#      MUXCY                       : 18
#      VCC                         : 1
#      XORCY                       : 19
# FlipFlops/Latches                : 36
#      FDC                         : 21
#      FDCE                        : 15
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 12
#      IBUF                        : 1
#      OBUF                        : 11
```

Timing Report :

```
Timing Report

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
-----------------------------------+------------------------+-------+
Clock Signal                       | Clock buffer(FF name)  | Load  |
-----------------------------------+------------------------+-------+
clock                              | NONE(EN)               | 17    |
clk                                | BUFGP                  | 19    |
-----------------------------------+------------------------+-------+
```

Conclusion:

Experiment No. 10

Title : To write VHDL code, simulate, synthesis, implement for Keypad Interfacing with 7 segment Display.

Software Used : Xilinx 14.7

Apparatus Required : FPGA Board, Spartan 6

Theory :

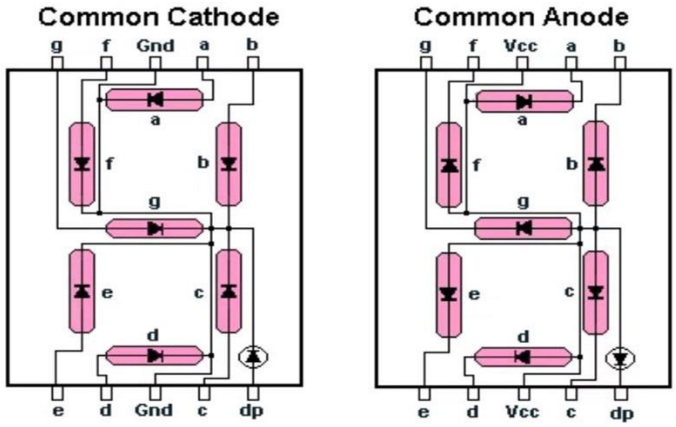
Matrix Keypad –

Matrix keypad consists of a set of buttons similar to alphanumeric keyboard provided with keys usually marked with letters or numbers and various extra keys. Embedded systems which require user interaction must be interfaced with devices that accept user input such as a keypad.

Interfacing Matrix Keypad with FPGA UDB –

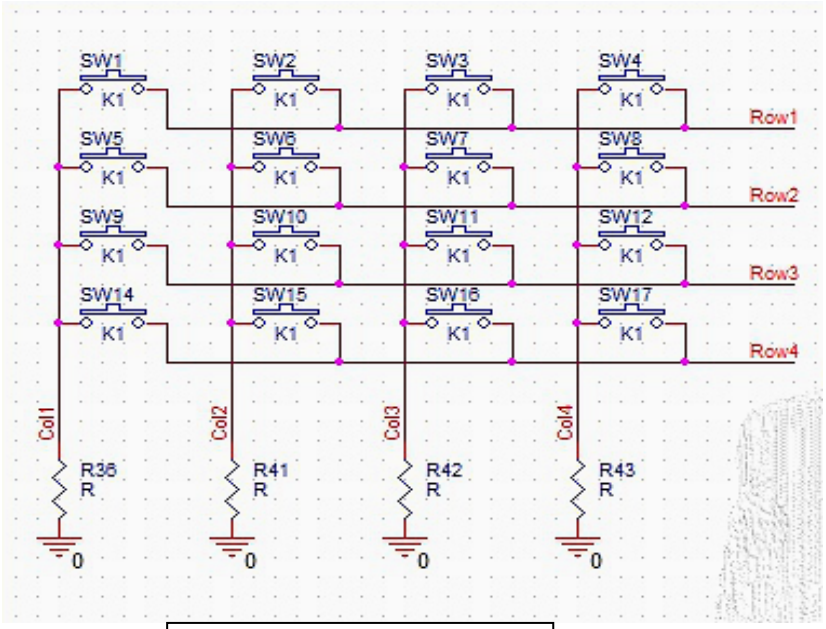
The Spartan-6 Primer board has 4×4 Matrix Keypad, indicated as in the Figure. Keypads arranged by a matrix format, each row and column section pulled by high, all row lines and column lines connected directly by the I/O pins.

Seven Segment Display Diagram –



SSD Configuration

Diagram :



4x4 Keypad

Procedure :

1) Create a new Project, select new source as VHDL module and write the VHDL code for the intended design.
2) Perform Check syntax operation to verify that the code is syntactically correct.
3) View RTL schematic and observe the Block diagram and its detailed connection.
4) Study the synthesis report in detail.
5) Launch ISIM Simulator, force signals to the inputs in signal window, run the simulator and observe the output in Wave window.
6) Write VHDL testbench for the Design-under-test (DUT) and simulate it to observe the behavior of design.
7) Select the target FPGA device.
8) For the VHDL file, create Implementation constraint file.
9) In UCF file, assign package pins to the input and output ports of design object list as per the selected target device.(The pin numbers can be referred from the Matrix II board manual.
10) Implement the design which includes translation, mapping, placement and routing.
11) Select the FPGA startup clock and configuration mode as per the selected target FPGA device.
12) Add Xilinx device, select the appropriate bit file and program the FPGA.
13) Apply the input through onboard switches or interfaced Add-on Module and observe the output on the intended output device which may be LED/LCD/SSD/Add-on Module.

Module 1 : Keypad Interfacing

```
----------------------------------------------------------------------------------
-- Company:Zeal College Of Engineering & Research
-- Engineer:Tanmay Vilas Pawar

-- Create Date:    02:59:46 10/11/2023
-- Design Name:    Keypad
-- Module Name:    keypad_interface_48 - Behavioral
-- Project Name:   Keypad Interface
-- Target Devices: XC6SLX9
----------------------------------------------------------------------------------
```

VHDL Program :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity keypad is
   Port ( clk : in  STD_LOGIC;
          rst : in  STD_LOGIC;
          row : inout  STD_LOGIC_VECTOR (3 downto 0);
          col : inout STD_LOGIC_VECTOR (3 downto 0);
        digit : out STD_LOGIC_VECTOR (3 downto 0);
      display : out STD_LOGIC_VECTOR (7 downto 0));
```

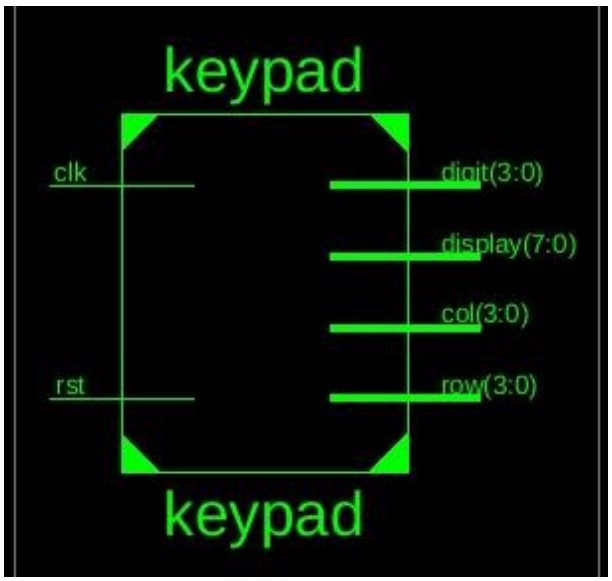
```
end keypad;
architecture keypad_arch of keypad is
TYPE STATE_TYPE IS ( Col1Set, Col2Set, Col3Set, Col4Set );
SIGNAL coltest : STATE_TYPE;
SIGNAL data:STD_LOGIC_VECTOR (7 downto 0);
SIGNAL rowm:STD_LOGIC_VECTOR (3 downto 0);
SIGNAL clock:STD_LOGIC;
begin
--------------------Process for clock divide by 10----------------------
Process(clk,rst)
variable temp:integer range 1 to 10;
begin
if(rst='0') then
temp:=1;
elsif(rising_edge(clk)) then
temp:=temp+1;
if(temp=10) then
clock<= not clock;
temp:=1;
end if;
end if;
end process;
------------------Process for Keypad scan & Display---------------------
process(clock,rst)
begin
if (rst='0') then
coltest<=col1set;
rowm<="1111";
data<=x"00";
elsif rising_edge (clock) then
digit<="1111";
case coltest is
when col1set=>rowm<="0111";
case col is
  when "0111"=>data<= x"C6";  -----------C
  when "1011"=>data<= x"A1";  -----------D
  when "1101"=>data<= x"86";  -----------E
  when "1110"=>data<= x"8E";  -----------F
  when others=>coltest<=col2set;
end case;
when col2set=>  rowm<="1011";
case col is
  when "0111"=>data<= x"C0";  -----------0
  when "1011"=>data<= x"F9";  -----------1
  when "1101"=>data<= x"A4";  -----------2
  when "1110"=>data<= x"B0";  -----------3
```

```
  when others=>coltest<=col3set;
end case;
when col3set=>   rowm<="1101";
case col is
  when "0111"=>data<= x"99";  ----------4
  when "1011"=>data<= x"92";  ----------5
  when "1101"=>data<= x"82";  ----------6
  when "1110"=>data<= x"F8";  ----------7
  when others=>coltest<=col4set;
end case;
  when col4set=>   rowm <="1110";
case col is
  when "0111"=>data<= x"80";  ----------8
  when "1011"=>data<= x"90";  ----------9
  when "1101"=>data<= x"88";  ----------A
  when "1110"=>data<= x"83";  ----------B
  when others=>coltest<=col1set;
end case;
end case;
end if;
end process;
display<=data;
row<=rowm;
end keypad_arch;
```

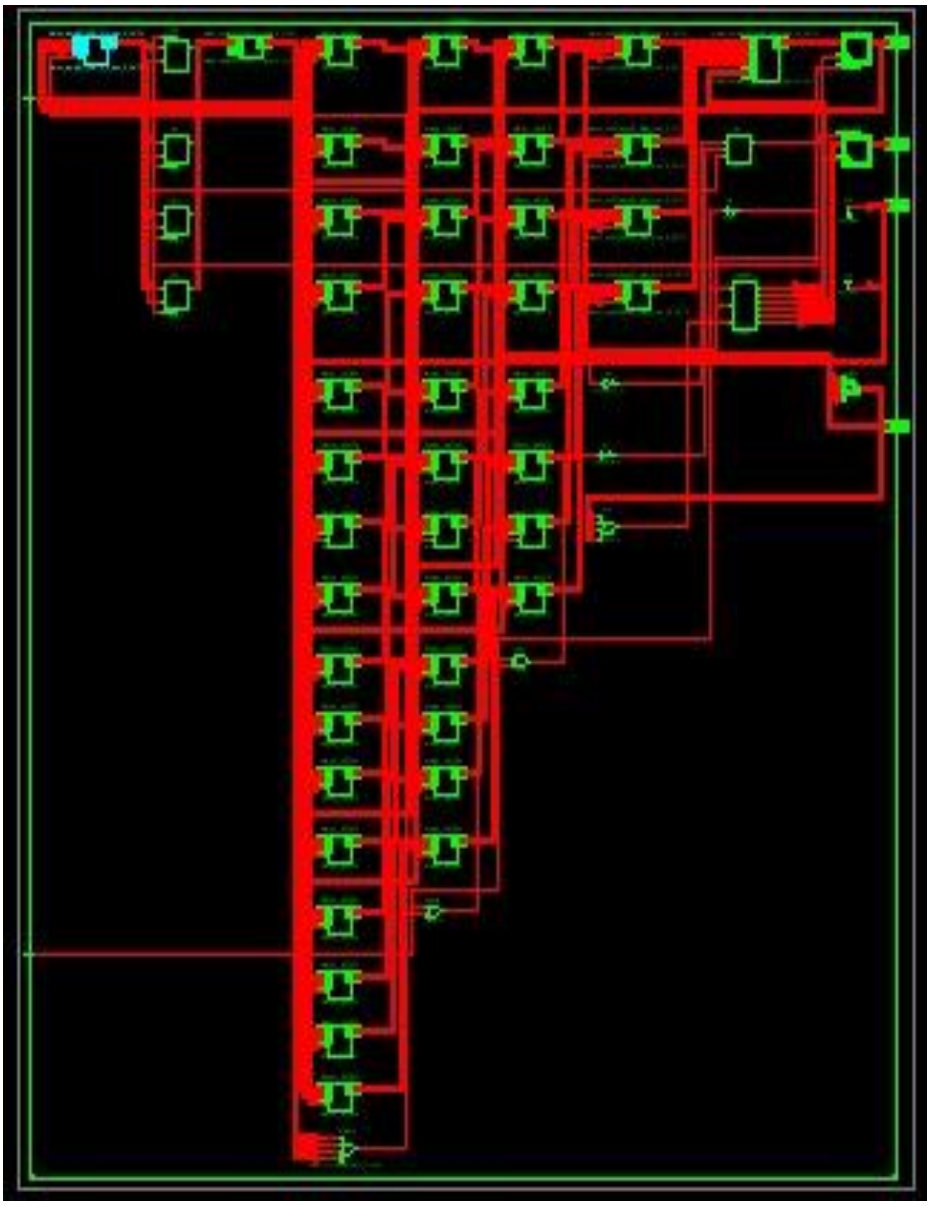
Keypad_interfce.ucf File :

```
NET "clk" LOC = "P56";
NET "rst" LOC = "P22";
NET "clk" CLOCK_DEDICATED_ROUTE = true;
NET "rst" CLOCK_DEDICATED_ROUTE = FALSE;
NET "display[0]" LOC = "P48";
NET "display[1]" LOC = "P45";
NET "display[2]" LOC = "P51";
NET "display[3]" LOC = "P47";
NET "display[4]" LOC = "P57";
NET "display[5]" LOC = "P50";
NET "display[6]" LOC = "P59";
NET "display[7]" LOC = "P55";
NET "col[0]" LOC = "P58"  | PULLUP | DRIVE = 2;
NET "col[1]" LOC = "P62"  | PULLUP | DRIVE = 2;
NET "col[2]" LOC = "P61"  | PULLUP | DRIVE = 2;
NET "col[3]" LOC = "P66"  | PULLUP | DRIVE = 2;
NET "row[0]" LOC = "P64";
NET "row[1]" LOC = "P16";
NET "row[2]" LOC = "P117";
NET "row[3]" LOC = "P75";
```

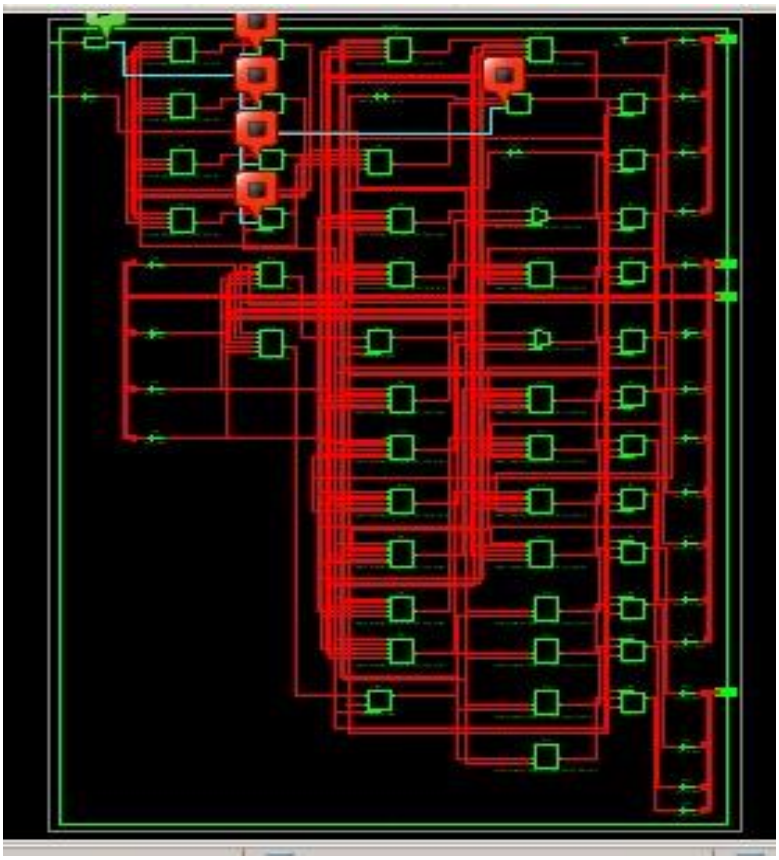
Block Diagram :



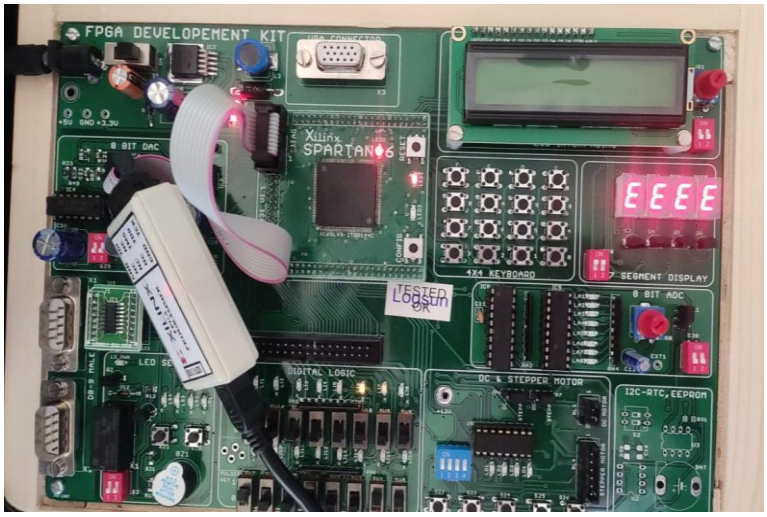
RTL Schematic :

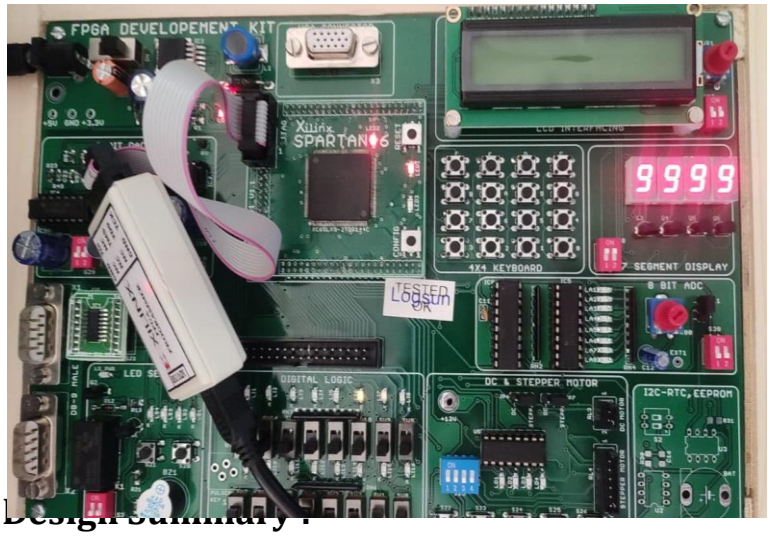


Technology Schematic :



Output :





Design Summary :

```
=========================================================
*                            Design Summary
=========================================================

Top Level Output File Name         : keypad.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 31
#      INV                         : 2
#      LUT2                        : 4
#      LUT3                        : 1
#      LUT4                        : 3
#      LUT5                        : 6
#      LUT6                        : 12
#      MUXF7                       : 2
#      VCC                         : 1
# FlipFlops/Latches                : 19
#      FDC                         : 13
#      FDE                         : 1
#      FDP                         : 5
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 21
#      IBUF                        : 5
#      OBUF                        : 16
```

Timing Report :

```
---------------------------------------------------------------------------
Timing Report

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
-----------------------------------+------------------------+-------+
Clock Signal                       | Clock buffer(FF name)  | Load  |
-----------------------------------+------------------------+-------+
clk                                | BUFGP                  | 5     |
clock                              | NONE(rowm_0)           | 14    |
-----------------------------------+------------------------+-------+
INFO:Xst:2169 - HDL ADVISOR - Some clock signals were not automatically buffered
```

Conclusion:

Experiment No.1

Title: Write a JavaScript program to calculate area of triangle, area of rectangle and area of circle.

Software Requirement:

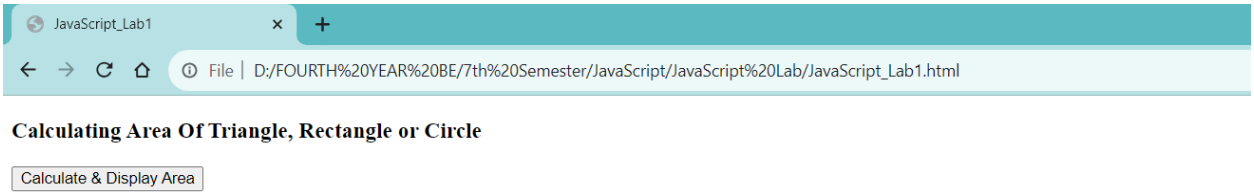
- VS code
- Google Crome (Browser)

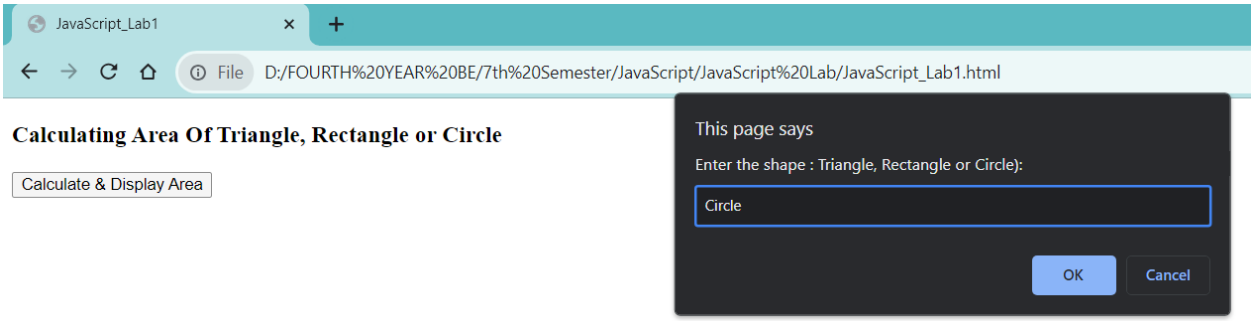
Program:

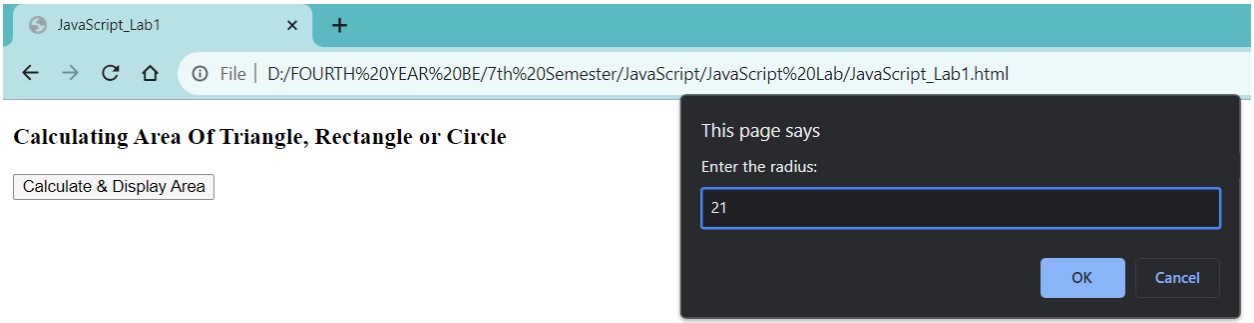
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JavaScript_Lab1</title>
</head>
<body>
  <h3>
    Calculating Area Of Triangle, Rectangle or Circle
  </h3>
  <script>
    // Function to calculate the area of a triangle
    function calculateTriangleArea(base, height) {
      return 0.5 * base * height;
    }

    // Function to calculate the area of a rectangle
    function calculateRectangleArea(length, width) {
      return length * width;
    }

    // Function to calculate the area of a circle
    function calculateCircleArea(radius) {
      return Math.PI * Math.pow(radius, 2);
    }

    // Function to handle user input and display results
    function calculateAndDisplay() {
      const shape = prompt("Enter the shape : Triangle, Rectangle or Circle):").toLowerCase();

      if (shape === "triangle") {
      const base = parseFloat(prompt("Enter the base length:"));
      const height = parseFloat(prompt("Enter the height:"));
      const triangleArea = calculateTriangleArea(base, height);
      alert(`The area of the triangle is: ${triangleArea}`);
      } else if (shape === "rectangle") {
      const length = parseFloat(prompt("Enter the length:"));
```

```
        const width = parseFloat(prompt("Enter the width:"));
        const rectangleArea = calculateRectangleArea(length, width);
        alert(`The area of the rectangle is: ${rectangleArea}`);
        } else if (shape === "circle") {
        const radius = parseFloat(prompt("Enter the radius:"));
        const circleArea = calculateCircleArea(radius);
        alert(`The area of the circle is: ${circleArea}`);
        } else {
        alert("Invalid shape entered. Please choose triangle, rectangle, or circle.");
        }
      }

      // Call the function to start the program
      // calculateAndDisplay();
    </script>
    <button onclick="calculateAndDisplay()">Calculate & Display Area</button>
</body>
</html>
```

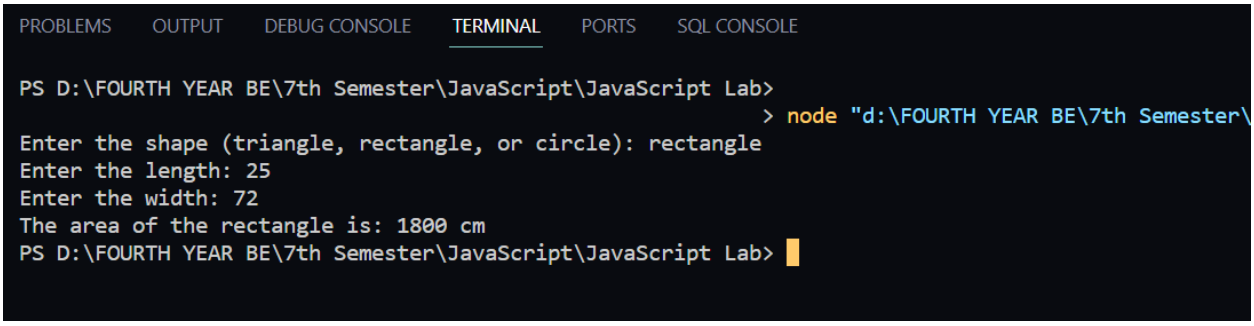
Output:








JavaScript_Lab1
File | D:/FOURTH%20YEAR%20BE/7th%20Semester/JavaScript/JavaScript%20Lab/JavaScript_Lab1.html
Calculating Area Of Triangle, Rectangle or Circle
Calculate & Display Area
This page says
The area of the circle is: 1385.4423602330987
OK


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SQL CONSOLE
PS D:\FOURTH YEAR BE\7th Semester\JavaScript\JavaScript Lab>
> node "d:\FOURTH YEAR BE\7th Semester\
Enter the shape (triangle, rectangle, or circle): rectangle
Enter the length: 25
Enter the width: 72
The area of the rectangle is: 1800 cm
PS D:\FOURTH YEAR BE\7th Semester\JavaScript\JavaScript Lab>

Experiment No.2

Title: Write a JavaScript program to generate the multiplication table of a given number.

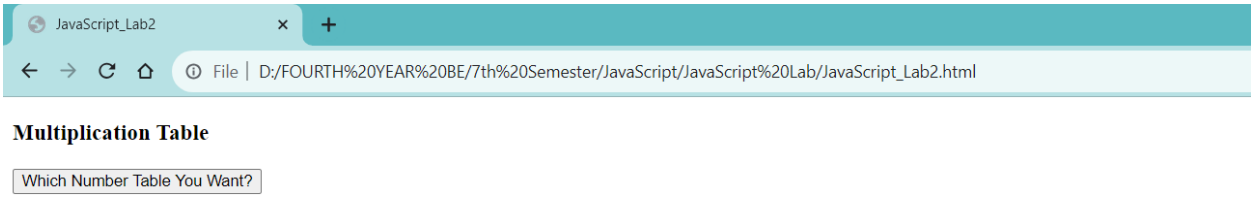
Software Requirement:

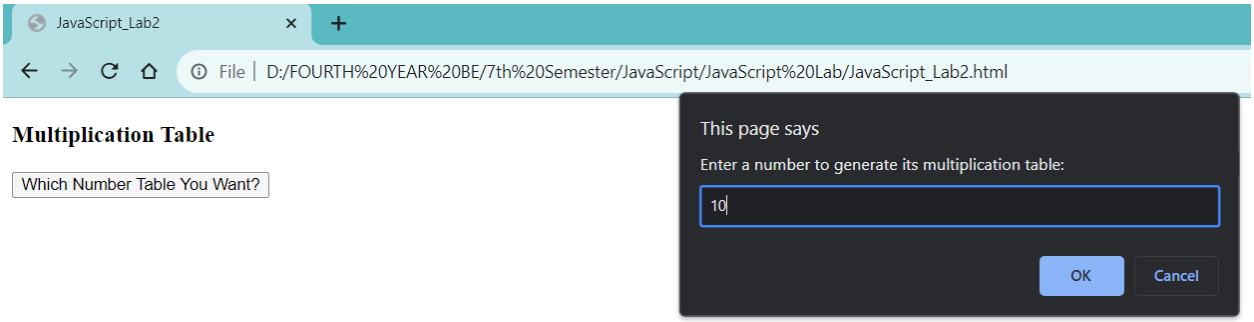
- VS code
- Google Crome (Browser)

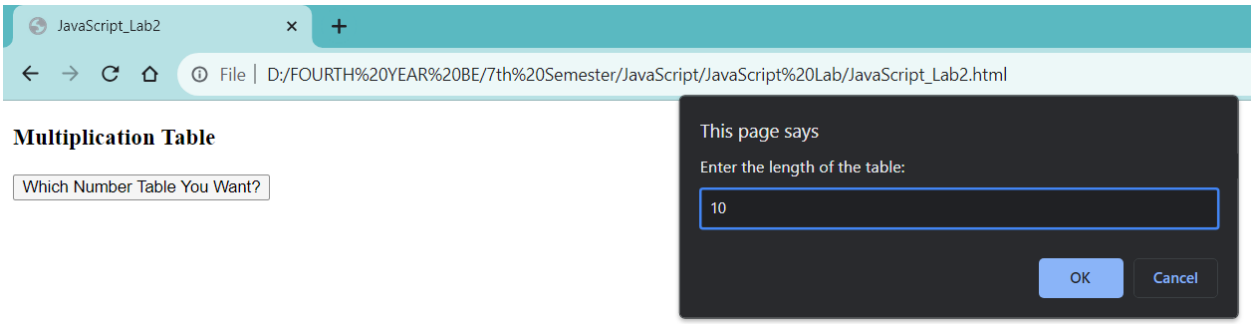
Program:

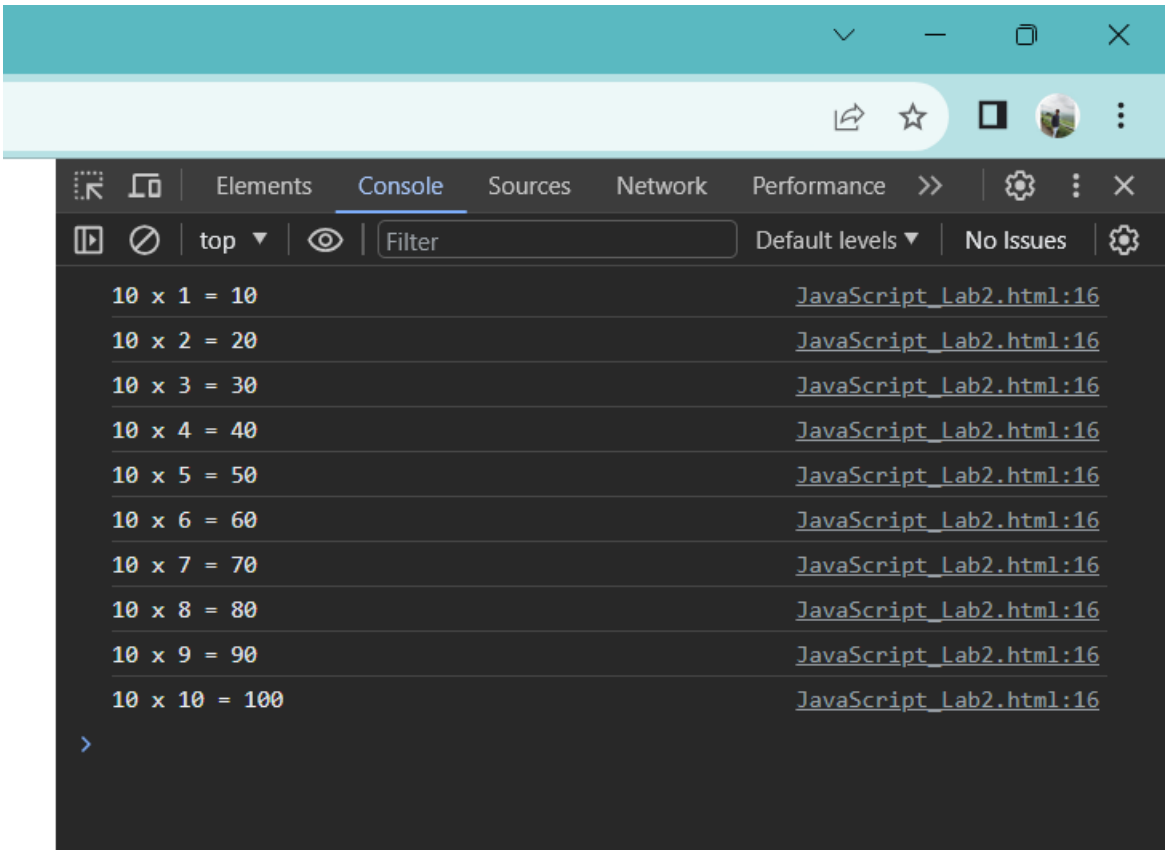
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JavaScript_Lab2</title>
</head>
<body>
  <h3>
    Multiplication Table
  </h3>
  <script>
    function generateMultiplicationTable(number, length) {
      for (let i = 1; i <= length; i++) {
      const product = number * i;
      console.log(`${number} x ${i} = ${product}`);
      }
    }

    function enterValues() {
      const inputNumber = parseInt(prompt("Enter a number to generate its multiplication table:"));
      const tableLength = parseInt(prompt("Enter the length of the table:"));

      if (!isNaN(inputNumber) && !isNaN(tableLength)) {
        generateMultiplicationTable(inputNumber, tableLength);
      } else {
      console.log("Invalid input. Please enter valid numbers.");
      }
    }
  </script>
  <button onclick="enterValues()">Which Number Table You Want?</button>
</body>
</html>
```

Output:


JavaScript_Lab2
File | D:/FOURTH%20YEAR%20BE/7th%20Semester/JavaScript/JavaScript%20Lab/JavaScript_Lab2.html
Multiplication Table
Which Number Table You Want?


JavaScript_Lab2
File | D:/FOURTH%20YEAR%20BE/7th%20Semester/JavaScript/JavaScript%20Lab/JavaScript_Lab2.html
Multiplication Table
Which Number Table You Want?
This page says
Enter a number to generate its multiplication table:
10
OK
Cancel


JavaScript_Lab2
File | D:/FOURTH%20YEAR%20BE/7th%20Semester/JavaScript/JavaScript%20Lab/JavaScript_Lab2.html
Multiplication Table
Which Number Table You Want?
This page says
Enter the length of the table:
10
OK
Cancel


Elements
Console
Sources
Network
Performance
top
Filter
Default levels
No Issues
10 x 1 = 10 JavaScript_Lab2.html:16
10 x 2 = 20 JavaScript_Lab2.html:16
10 x 3 = 30 JavaScript_Lab2.html:16
10 x 4 = 40 JavaScript_Lab2.html:16
10 x 5 = 50 JavaScript_Lab2.html:16
10 x 6 = 60 JavaScript_Lab2.html:16
10 x 7 = 70 JavaScript_Lab2.html:16
10 x 8 = 80 JavaScript_Lab2.html:16
10 x 9 = 90 JavaScript_Lab2.html:16
10 x 10 = 100 JavaScript_Lab2.html:16

Experiment No.3

Title: Write a JavaScript program to following operations on a given string –

a. Reverse string.
b. Replace characters of a string.
c. String is palindrome.

Software Requirement:

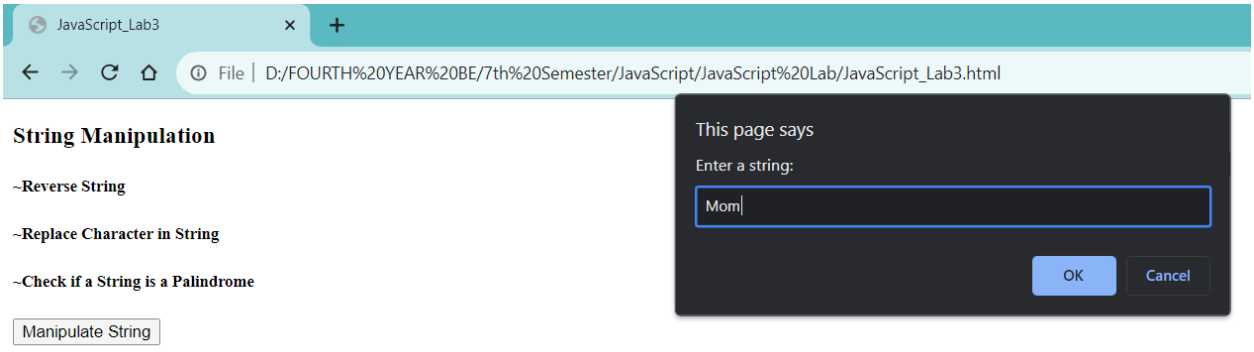
- VS code
- Google Crome (Browser)

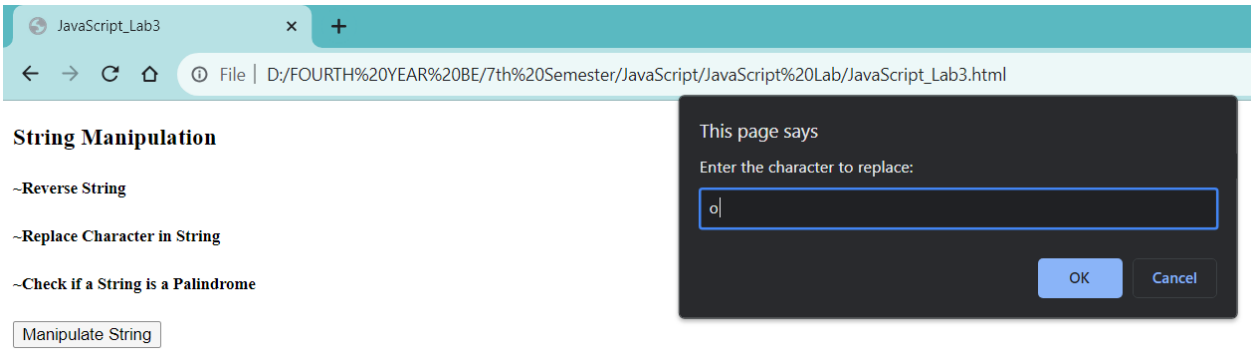
Program:

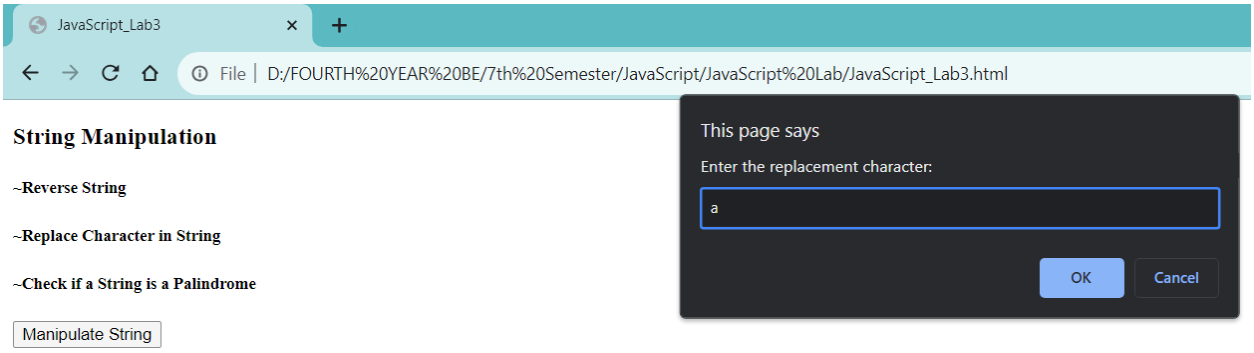
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JavaScript_Lab3</title>
</head>
<body>
  <h3>String Manipulation</h3>
  <h5>~Reverse String</h5>
  <h5>~Replace Character in String</h5>
  <h5>~Check if a String is a Palindrome</h5>
  <script>
    // Function to reverse a string
    function reverseString(inputString) {
      return inputString.split('').reverse().join('');
    }

    // Function to replace characters in a string
    function replaceCharacters(inputString, charToReplace, replacementChar) {
      return inputString.replace(new RegExp(charToReplace, 'g'), replacementChar);
    }

    // Function to check if a string is a palindrome
    function isPalindrome(inputString) {
      const cleanedString = inputString.toLowerCase().replace(/[^a-zA-Z0-9]/g, '');
      const reversedString = cleanedString.split('').reverse().join('');
      return cleanedString === reversedString;
    }

    function stringManipulation() {
      const userString = prompt("Enter a string:");

      if (userString) {
        // Reverse the string
        const reversed = reverseString(userString);
        console.log(`Reversed string: ${reversed}`);

        // Replace characters
        const charToReplace = prompt("Enter the character to replace:");
        const replacementChar = prompt("Enter the replacement character:");
```

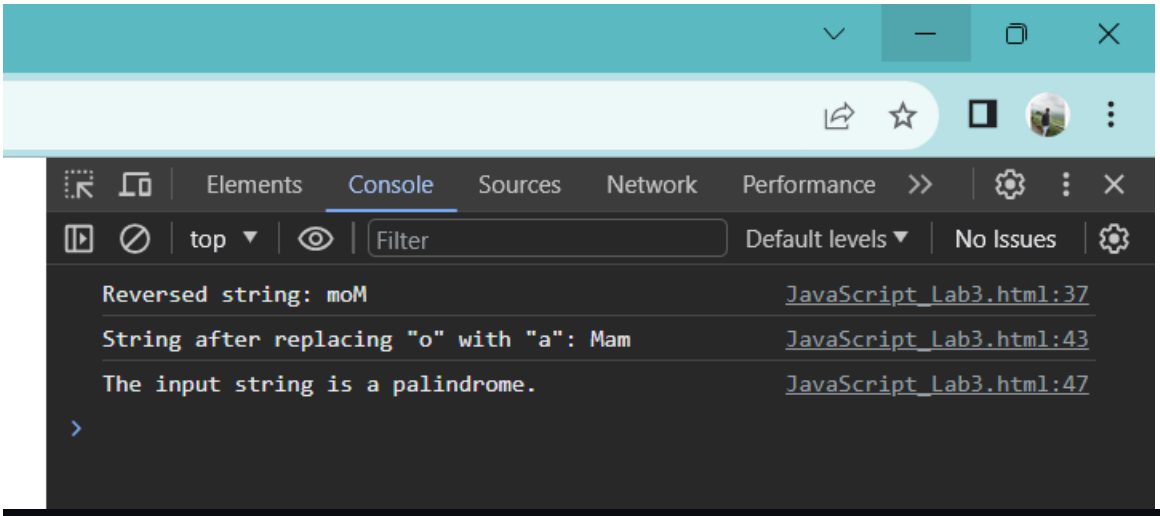
```
    const replacedString = replaceCharacters(userString, charToReplace, replacementChar);
    console.log(`String after replacing "${charToReplace}" with "${replacementChar}":
${replacedString}`);

    // Check if the string is a palindrome
    if (isPalindrome(userString)) {
      console.log("The input string is a palindrome.");
    } else {
    console.log("The input string is not a palindrome.");
    }
   } else {
   console.log("Invalid input. Please enter a valid string.");
   }
  }
 </script>
 <button onclick="stringManipulation()">Manipulate String</button>
</body>
</html>
```

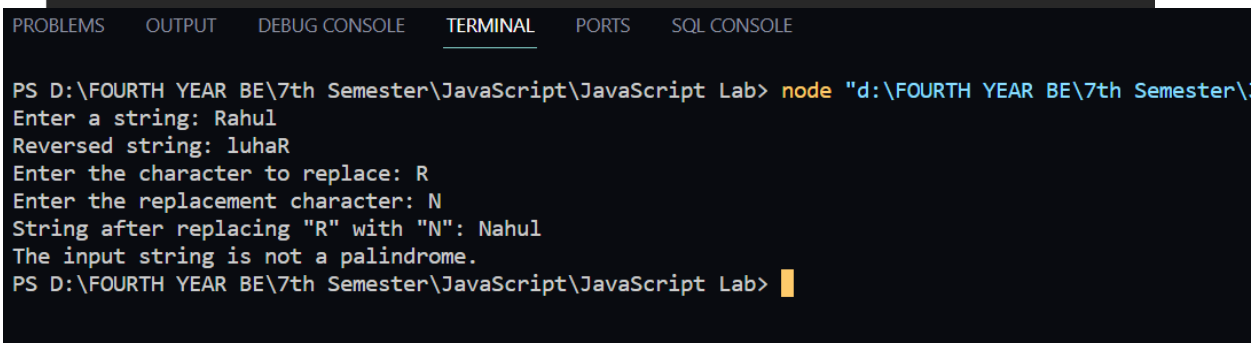
Output:






JavaScript_Lab3
File | D:/FOURTH%20YEAR%20BE/7th%20Semester/JavaScript/JavaScript%20Lab/JavaScript_Lab3.html
String Manipulation
~Reverse String
~Replace Character in String
~Check if a String is a Palindrome
Manipulate String
This page says
Enter the replacement character:
a
OK
Cancel


Elements
Console
Sources
Network
Performance
top
Filter
Default levels
No Issues
Reversed string: moM
JavaScript_Lab3.html:37
String after replacing "o" with "a": Mam
JavaScript_Lab3.html:43
The input string is a palindrome.
JavaScript_Lab3.html:47


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SQL CONSOLE
PS D:\FOURTH YEAR BE\7th Semester\JavaScript\JavaScript Lab> node "d:\FOURTH YEAR BE\7th Semester\
Enter a string: Rahul
Reversed string: luhaR
Enter the character to replace: R
Enter the replacement character: N
String after replacing "R" with "N": Nahul
The input string is not a palindrome.
PS D:\FOURTH YEAR BE\7th Semester\JavaScript\JavaScript Lab>

Experiment No.4

Title: Write a JavaScript program to compare two strings using various methods.

Software Requirement:

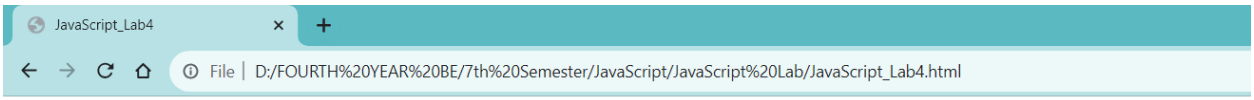
- VS code
- Google Crome (Browser)

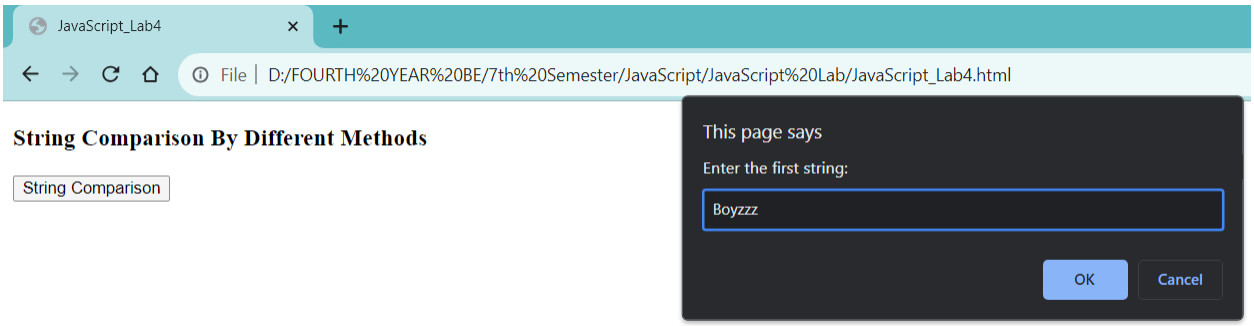
Program:

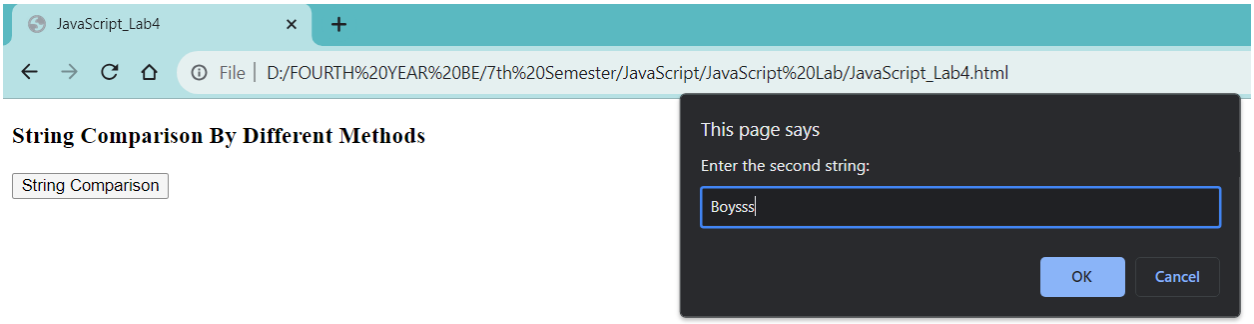
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JavaScript_Lab4</title>
</head>
<body>
  <h3>String Comparison By Different Methods</h3>
  <script>
    function compareStrings(string1, string2) {
      // Using equality (==) to compare strings
      const equalityComparison = string1 == string2;

      // Using strict equality (===) to compare strings
      const strictEqualityComparison = string1 === string2;

      // Using locale-based comparison
      const localeComparison = string1.localeCompare(string2);

      // Using case-insensitive comparison
      const caseInsensitiveComparison = string1.toLowerCase() === string2.toLowerCase();

      return {
        equalityComparison,
        strictEqualityComparison,
        localeComparison,
        caseInsensitiveComparison
      };
    }

    function stringComparison() {
      const string1 = prompt("Enter the first string:");
      const string2 = prompt("Enter the second string:");

      const comparisons = compareStrings(string1, string2);

      console.log("Equality (==) comparison:", comparisons.equalityComparison);
```

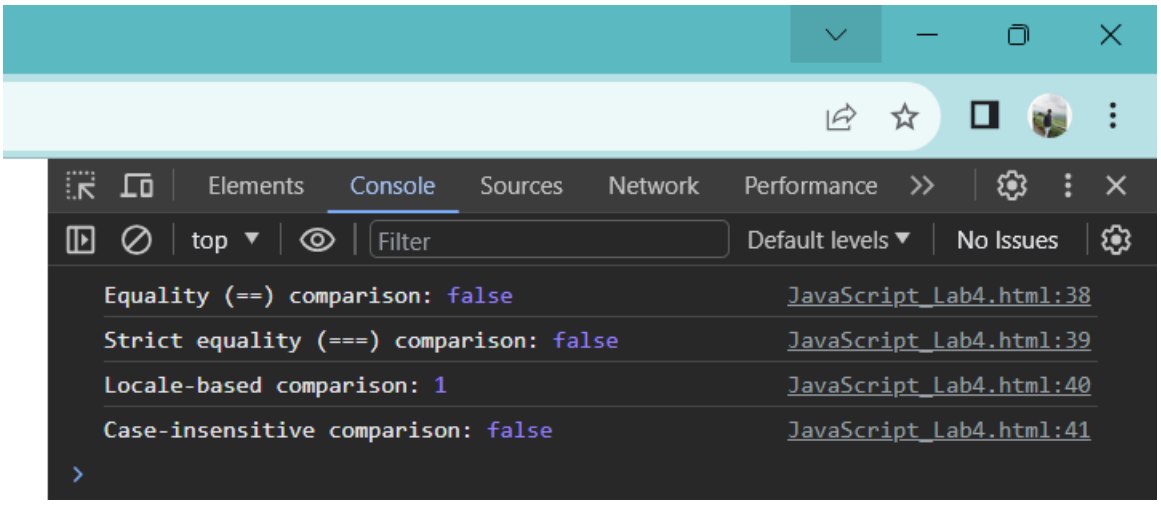
```
        console.log("Strict equality (===) comparison:", comparisons.strictEqualityComparison);
        console.log("Locale-based comparison:", comparisons.localeComparison);
        console.log("Case-insensitive comparison:", comparisons.caseInsensitiveComparison);
      }
  </script>
  <button onclick="stringComparison()">String Comparison</button>
</body>
</html>
```

Output:









Experiment No.5

Title: Write a JavaScript program that will create a countdown timer.

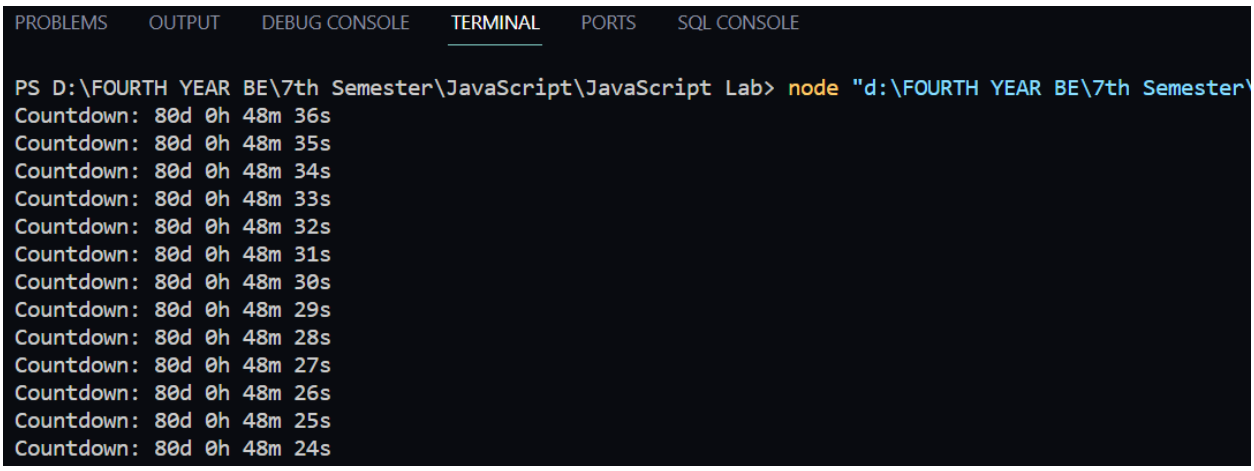
Software Requirement:

- VS code

Program:

```
function startCountdown(targetDate) {
  function updateTimer() {
    const currentDate = new Date();
    const timeDifference = targetDate - currentDate;

    if (timeDifference <= 0) {
      clearInterval(timerInterval);
      console.log("Countdown expired!");
      return;
    }

    const days = Math.floor(timeDifference / (1000 * 60 * 60 * 24));
    const hours = Math.floor((timeDifference % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60));
    const minutes = Math.floor((timeDifference % (1000 * 60 * 60)) / (1000 * 60));
    const seconds = Math.floor((timeDifference % (1000 * 60)) / 1000);

    console.log(`Countdown: ${days}d ${hours}h ${minutes}m ${seconds}s`);
  }

  const timerInterval = setInterval(updateTimer, 1000);
  updateTimer();
}

const targetDate = new Date("2023-12-31 23:59:59").getTime();
startCountdown(targetDate);
```

Output:



Experiment No.6

Title: Write a JavaScript program that will create an array and perform following operations –

a. To remove specific element from the array.
b. Check if an array contains a specified value.
c. To empty an array.

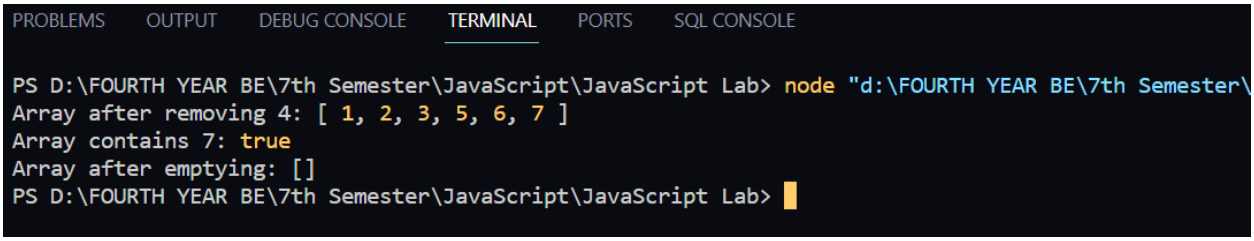
Software Requirement:

- VS code

Program:

```
// Create an array
const myArray = [1, 2, 3, 4, 5, 6, 7];

// Function to remove a specific element from the array
function removeElementFromArray(arr, element) {
  const index = arr.indexOf(element);
  if (index !== -1) {
    arr.splice(index, 1);
  }
}

// Function to check if an array contains a specified value
function arrayContainsValue(arr, value) {
  return arr.includes(value);
}

// Function to empty an array
function emptyArray(arr) {
  arr.length = 0;
}

// Remove the element 4 from the array
removeElementFromArray(myArray, 4);
console.log("Array after removing 4:", myArray);

// Check if the array contains the value 7
const containsValue = arrayContainsValue(myArray, 7);
console.log("Array contains 7:", containsValue);

// Empty the array
emptyArray(myArray);
console.log("Array after emptying:", myArray);
```

Output:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SQL CONSOLE
PS D:\FOURTH YEAR BE\7th Semester\JavaScript\JavaScript Lab> node "d:\FOURTH YEAR BE\7th Semester\
Array after removing 4: [1, 2, 3, 5, 6, 7]
Array contains 7: true
Array after emptying: []
PS D:\FOURTH YEAR BE\7th Semester\JavaScript\JavaScript Lab>

Experiment No.7

Title: Write a JavaScript program that will append an object to an array and will check if an object is an array.

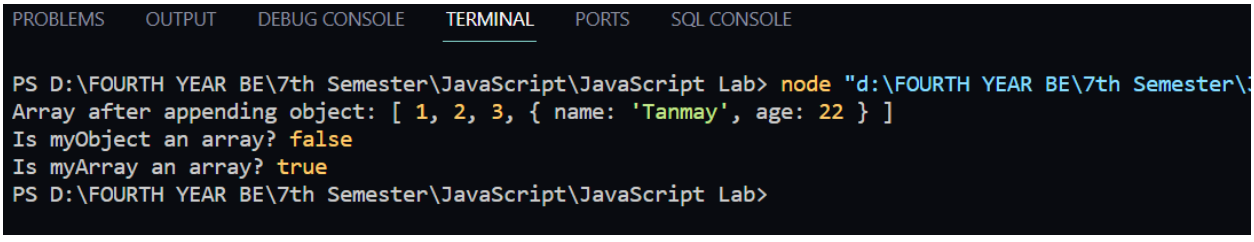
Software Requirement:

- VS code

Program:

```
// Create an array
const myArray = [1, 2, 3];

// Function to append an object to an array
function appendObjectToArray(arr, obj) {
  arr.push(obj);
}

// Function to check if an object is an array
function isObjectArray(obj) {
  return Array.isArray(obj);
}

// Append an object to the array
const myObject = { name: "Tanmay", age: 22 };
appendObjectToArray(myArray, myObject);

console.log("Array after appending object:", myArray);

// Check if an object is an array
console.log("Is myObject an array?", isObjectArray(myObject));
console.log("Is myArray an array?", isObjectArray(myArray));
```

Output:



Experiment No.8

Title: Write a JavaScript program to create a Home page of any website and change background color using –

a. On mouse over event
b. On focus event

Software Requirement:

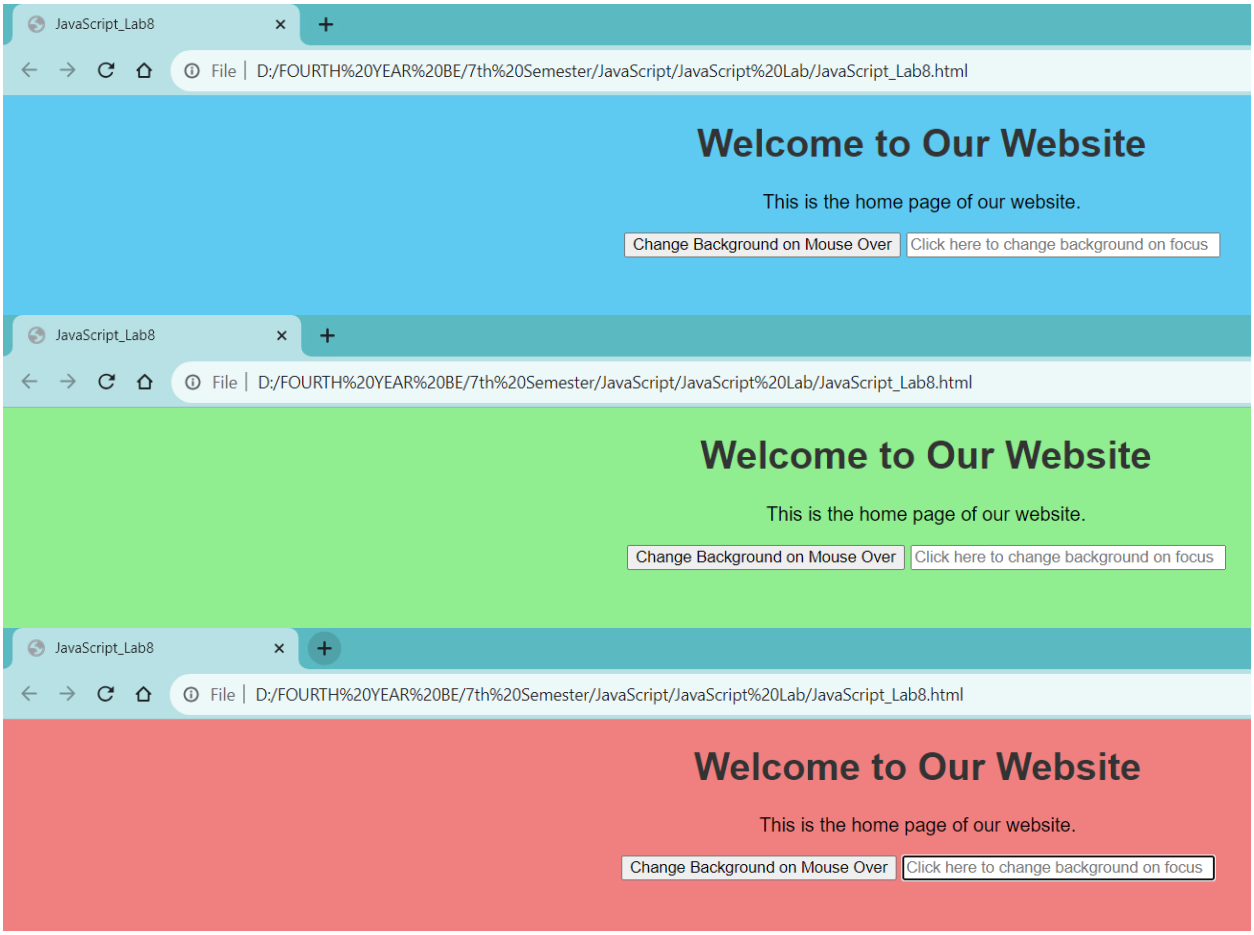
- VS code
- Google Crome (Browser)

Program:

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript_Lab8</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      background-color: rgb(95, 202, 241);
    }
    h1 {
      color: #333;
    }
  </style>
</head>
<body>
  <h1>Welcome to Our Website</h1>
  <p>This is the home page of our website.</p>

  <!-- Button to change background color on mouseover event -->
  <button id="changeBackgroundColorButton"
onmouseover="changeBackgroundColorOnMouseOver()">Change Background on Mouse
Over</button>

  <!-- Input to change background color on focus event -->
  <input type="text" id="backgroundColorInput" size="34" onfocus="changeBackgroundColorOnFocus()"
placeholder="Click here to change background on focus">

  <script>
    function changeBackgroundColorOnMouseOver() {
      document.body.style.backgroundColor = 'lightgreen';
    }

    function changeBackgroundColorOnFocus() {
```

```
    document.body.style.backgroundColor = 'lightcoral';
  }
 </script>
</body>
</html>
```

Output:



Experiment No.9

Title: Create a student information Form to accept information like Name, Address, City, State Gender, Mobile Number, and email id. Perform validations for –

a. Correct Names
b. Mobile Names
c. Email ID's
d. If no entered value
e. Re-display for wrongly entered values with message
f. Congratulation and Welcome page upon successful entries

Software Requirement:

- VS code
- Google Crome (Browser)

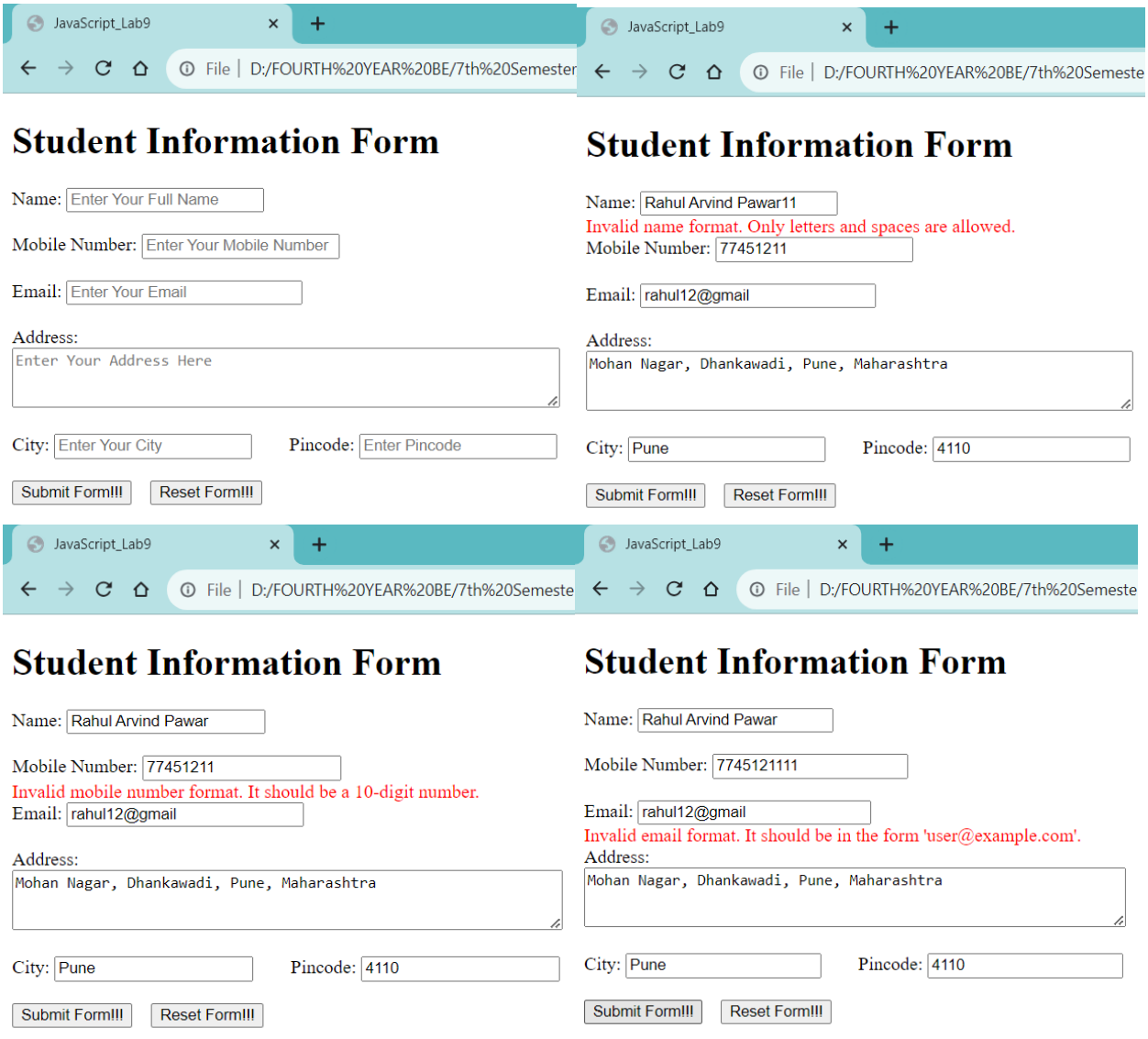
Program:

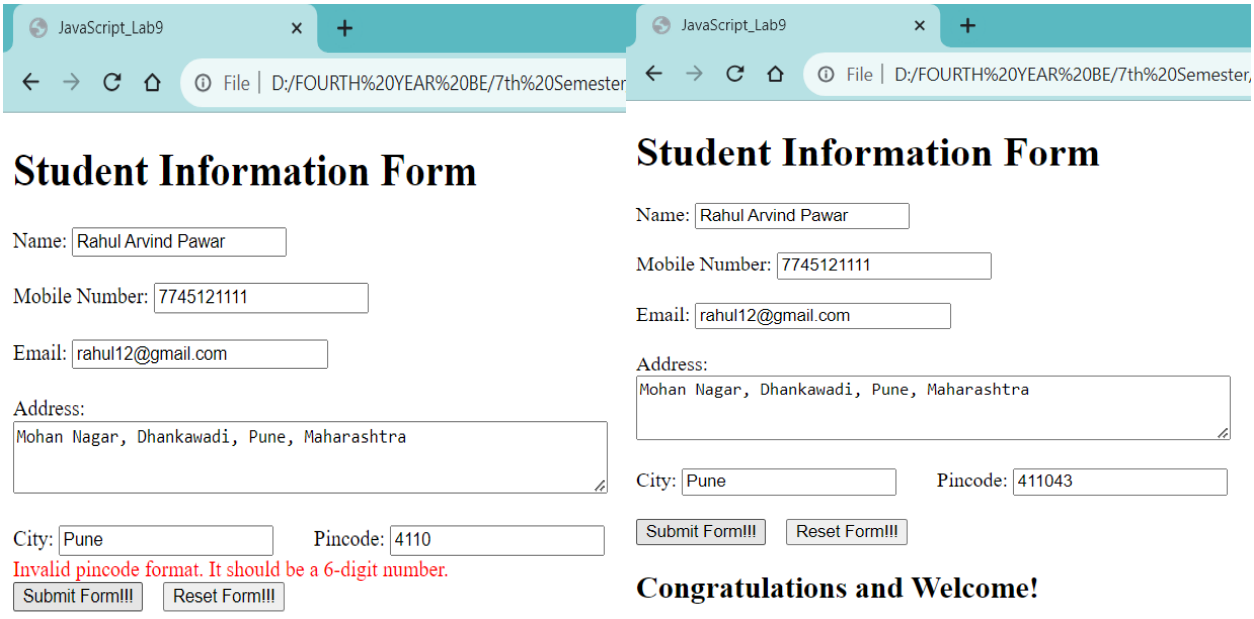
```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript_Lab9</title>
  <style>
    .error {
      color: red;
    }
  </style>
</head>
<body>
  <h1>Student Information Form</h1>
  <form id="studentForm" onsubmit="return validateForm()">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" placeholder="Enter Your Full Name" required><br>
    <span id="nameError" class="error"></span><br>

    <label for="mobile">Mobile Number:</label>
    <input type="text" id="mobile" name="mobile" placeholder="Enter Your Mobile Number"
required><br>
    <span id="mobileError" class="error"></span><br>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" size="25" placeholder="Enter Your Email"
required><br>
    <span id="emailError" class="error"></span><br>

    <label for="address">Address:</label><br>
    <textarea id="address" name="address" cols="61" rows="3" placeholder="Enter Your Address Here"
required></textarea><br>
    <span id="addressError" class="error"></span><br>

    <label for="city">City: </label>
```

```
  <input type="text" name="city" id="city" placeholder="Enter Your City" required>   
   
  <span id="cityError" class="error"></span>
  <label for="pincode">Pincode:</label>
  <input type="text" id="pincode" name="pincode" placeholder="Enter Pincode" required><br>
  <span id="pincodeError" class="error"></span><br>

  <input type="submit" value="Submit Form!!!">   
  <input type="reset" value="Reset Form!!!">

 </form>

 <div id="successMessage" style="display: none;">
  <h2>Congratulations and Welcome!</h2>
 </div>

 <script>
  function validateForm() {
   const name = document.getElementById("name").value;
   const mobile = document.getElementById("mobile").value;
   const email = document.getElementById("email").value;
   const pincode = document.getElementById("pincode").value;
   const address = document.getElementById("address").value;
   const namePattern = /^[A-Za-z\s]+$/;
   const mobilePattern = /^\d{10}$/;
   const emailPattern = /^\S+@\S+\.\S+$/;
   const pincodePattern = /^\d{6}$/;

   const nameError = document.getElementById("nameError");
   const mobileError = document.getElementById("mobileError");
   const emailError = document.getElementById("emailError");
   const pincodeError = document.getElementById("pincodeError");
   const addressError = document.getElementById("addressError");
   const successMessage = document.getElementById("successMessage");

   nameError.textContent = "";
   mobileError.textContent = "";
   emailError.textContent = "";
   pincodeError.textContent = "";
   addressError.textContent = "";

   if (!name.match(namePattern)) {
    nameError.textContent = "Invalid name format. Only letters and spaces are allowed.";
    return false;
   }

   if (!mobile.match(mobilePattern)) {
    mobileError.textContent = "Invalid mobile number format. It should be a 10-digit number.";
    return false;
   }

   if (!email.match(emailPattern)) {
    emailError.textContent = "Invalid email format. It should be in the form 'user@example.com'.";
```

```
      return false;
    }

    if (!pincode.match(pincodePattern)) {
      pincodeError.textContent = "Invalid pincode format. It should be a 6-digit number.";
      return false;
    }

    if (address.trim() === "") {
      addressError.textContent = "Address is required.";
      return false;
    }

    successMessage.style.display = "block";
    return false;
  }
 </script>
</body>
</html>
```

Output:




JavaScript_Lab9
File | D:/FOURTH%20YEAR%20BE/7th%20Semester
Student Information Form
Name: Rahul Arvind Pawar
Mobile Number: 7745121111
Email: rahul12@gmail.com
Address:
Mohan Nagar, Dhankawadi, Pune, Maharashtra
City: Pune
Pincode: 4110
Invalid pincode format. It should be a 6-digit number.
Submit Form!!!
Reset Form!!!
JavaScript_Lab9
File | D:/FOURTH%20YEAR%20BE/7th%20Semester/
Student Information Form
Name: Rahul Arvind Pawar
Mobile Number: 7745121111
Email: rahul12@gmail.com
Address:
Mohan Nagar, Dhankawadi, Pune, Maharashtra
City: Pune
Pincode: 411043
Submit Form!!!
Reset Form!!!
Congratulations and Welcome!