KATHMANDU UNIVERSITY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

LAB REPORT 02

EEEG-321

Department of Electrical and Electronics Engineering

By:

Samyam Shrestha (31056)

To:

Santosh Shaha Sir

Date:

30th April, 2024

Title: Implement Multiplexer and Demultiplexer

Requirements: Vivado Software, Laptop

1) **8:1 Multiplexer**

VHDL Code

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

entity muxx_56 is
   Port ( d : in STD_LOGIC_vector(7 downto 0);
          sel : in STD_LOGIC_vector (2 downto 0);
          y : out STD_LOGIC);
end muxx_56;

architecture Behavioral of muxx_56 is
begin
process (sel, d)
begin
   case sel is
      when "000" => Y <= d(0);
      when  "001" => Y <= d(1);
      when "010" => Y <= d(2);
      when "011" => Y <= d(3);
      when "100" => Y <= d(4);
      when "101" => Y <= d(5);
      when "110" => Y <= d(6);
      when "111" => Y <= d(7);
      when others => Y <= '0';
end case;
end process;
end Behavioral;
```

Test Bench code:

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

entity muxx_56_tb is
end muxx_56_tb;

architecture Behavioral of muxx_56_tb is
component muxx_56
PORT ( d: in std_logic_vector ( 7 downto 0);
      sel: in std_logic_vector (2 downto 0);
      y: out std_logic);
 end component;

signal d: std_logic_vector(7 downto 0):= "00000000";
signal sel: std_logic_vector(2 downto 0);
signal y: std_logic;
begin

uut: muxx_56 Port Map (d => d,
                 sel => sel,
                 y => y);
  stimuli: process
        begin
        wait for 10ns ;
        d <= "10101011";
        sel <= "000";
        wait for 10ns;
        sel <= "001";
        wait for 10ns;
        sel <= "010";
        wait for 10ns;
        sel <= "011";
        wait for 10ns;
        sel <= "100";
        wait for 10ns;
        sel <= "101";
        wait for 10ns;
        sel <= "110";
        wait for 10ns;
        sel <= "111";
        wait for 10ns;
   end process;
 end behavioral;
```

RTL Schematic Diagram:

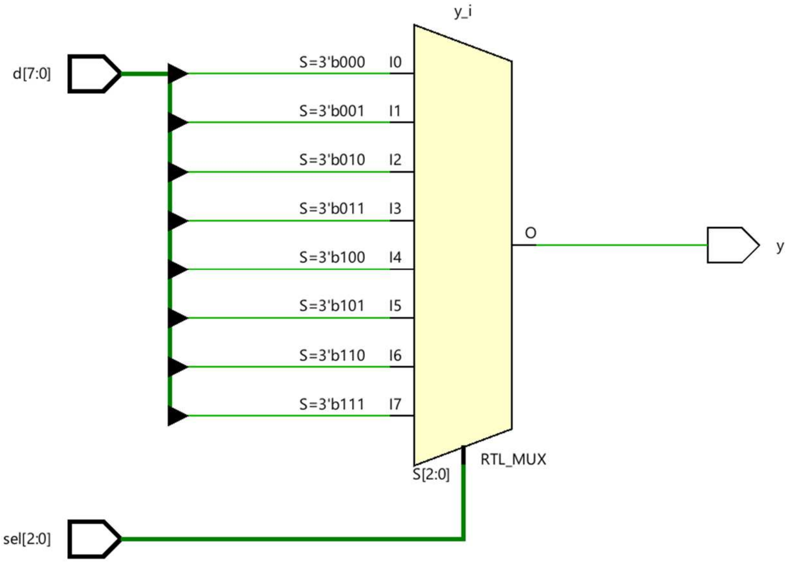


Figure 1: RTL Schematic of 8:1 Multiplexer

Simulation Model:

Here, three selection lines are given through sel variable. The output is shown in variable y.

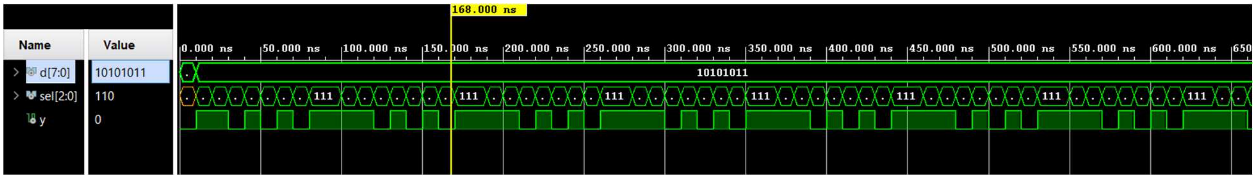
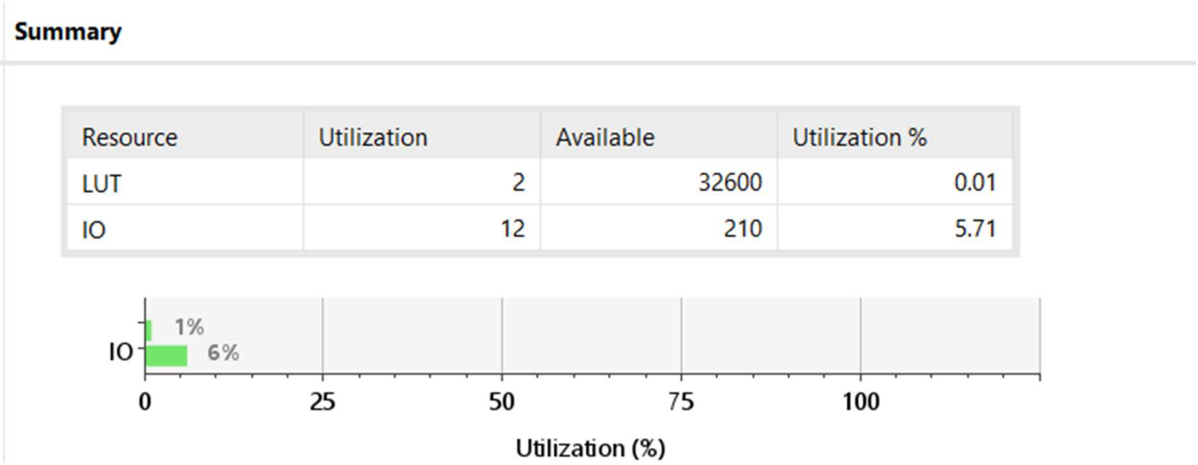


Figure 2: Simulation Model of 8:1 Multiplexer

Device Utilization Report:

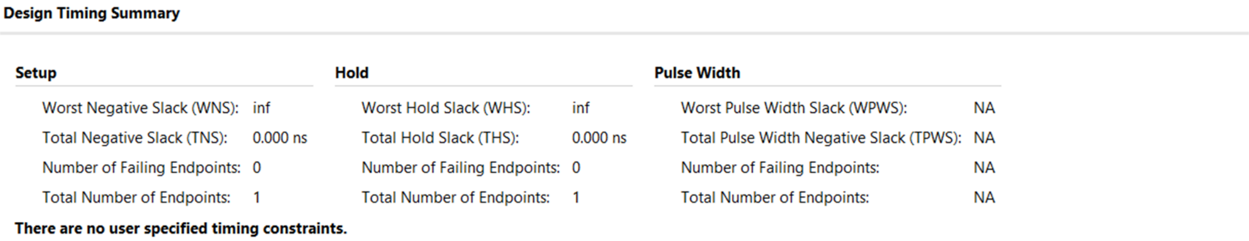


Summary

Resource	Utilization	Available	Utilization %
LUT	2	32600	0.01
IO	12	210	5.71

Figure 3: Device Utilization Report

Timing Analysis Report:



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	1	Total Number of Endpoints:	1	Total Number of Endpoints:	NA

There are no user specified timing constraints.

Figure 4: Timing Summary for 8:1 Multiplexer

2) Multiplexer 4:1

VHDL Code

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

entity muxx_56 is
   Port ( d : in STD_LOGIC_vector(3 downto 0);
          sel : in STD_LOGIC_vector (1 downto 0);
          y : out STD_LOGIC);
end muxx_56;

architecture Behavioral of muxx_56 is

begin
process (sel, d)
begin
   case sel is
      when "00" => Y <= d(0);
      when  "01" => Y <= d(1);
      when "10" => Y <= d(2);
      when "11" => Y <= d(3);
end case;
end process;
end Behavioral;
```

Test Bench Code

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

entity fourto1mux_tb is
end fourto1mux_tb;

architecture Behavioral of fourto1mux_tb is
component fourto1mux
PORT ( d: in std_logic_vector ( 3 downto 0);
      sel: in std_logic_vector (1 downto 0);
      y: out std_logic);
 end component;
signal d: std_logic_vector(3 downto 0):= "0000";
signal sel: std_logic_vector(1 downto 0);
signal y: std_logic;
begin

uut: fourto1mux Port Map (d => d,
                    sel => sel,
                    y => y);
```

```
  stimuli: process
     begin
              wait for 10ns ;
        d <= "1010";
        sel <= "00";
        wait for 10ns;
        sel <= "01";
        wait for 10ns;
        sel <= "10";
        wait for 10ns;
        sel <= "11";
        wait for 10 ns;
  end process;
end Behavioral;
```

RTL Schematic

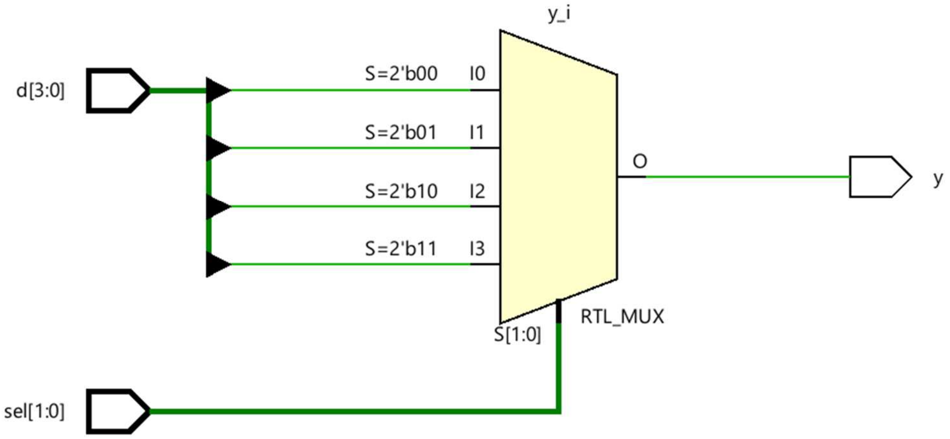


Figure 5: RTL Schematic of 4:1 Multiplexer

XDC File

```
set_property PACKAGE_PIN R15 [get_ports {d[3]}]
set_property PACKAGE_PIN M13 [get_ports {d[2]}]
set_property PACKAGE_PIN L16 [get_ports {d[1]}]
set_property PACKAGE_PIN J15 [get_ports {d[0]}]
set_property PACKAGE_PIN U8 [get_ports {sel[1]}]
set_property PACKAGE_PIN T8 [get_ports {sel[0]}]
set_property PACKAGE_PIN H17 [get_ports y]
set_property IOSTANDARD LVCMOS33 [get_ports {d[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {d[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {d[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sel[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sel[0]}]
```

set_property IOSTANDARD LVCMOS33 [get_ports y]

Simulation

The timing waveform graph for different possible combinations of input is given in the figure below. In a test-bench code, the different input combination was given and also the wait instruction for 10ns. So, while simulating, all the output can be seen in a single figure.

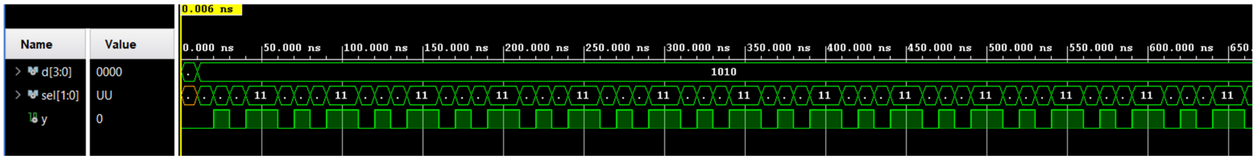
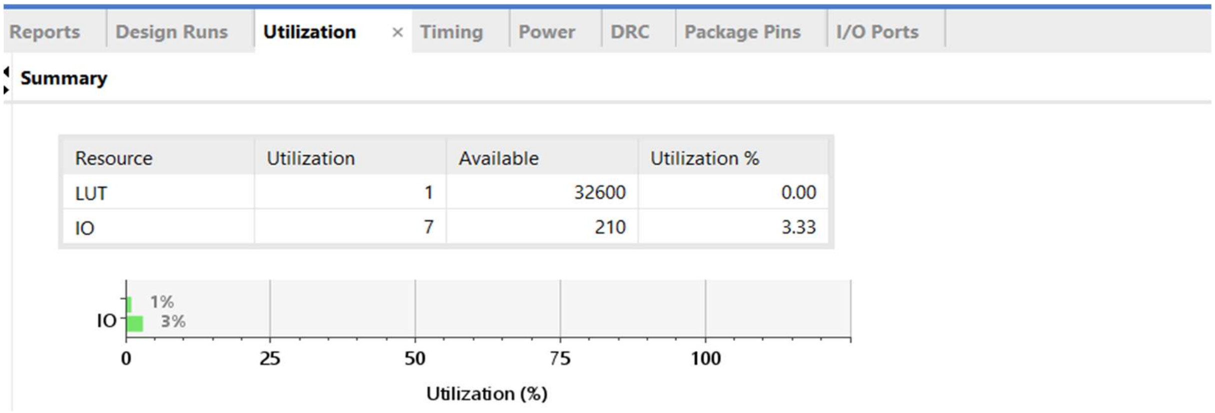


Figure 6: Timing Diagram for 4:1 Multiplexer

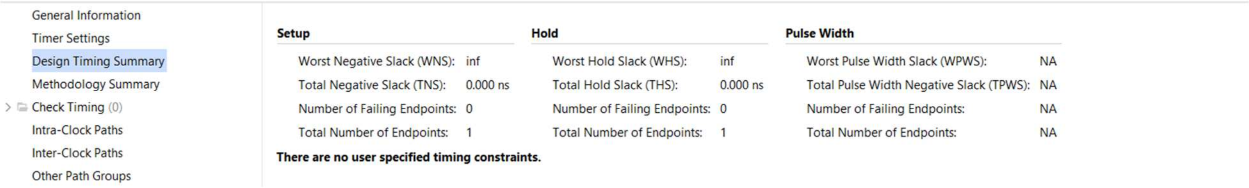
Device Utilization Report



Resource	Utilization	Available	Utilization %
LUT	1	32600	0.00
IO	7	210	3.33

Figure 7: Device Utilization Report for 4:1 Mux

Time Analysis Report



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	1	Total Number of Endpoints:	1	Total Number of Endpoints:	NA

There are no user specified timing constraints.

Figure 8: Timing Summary for Full Adder

3) 1:8 Demux

VHDL Code:

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

entity demux_56 is
   Port ( I : in STD_LOGIC;
          S : inout STD_LOGIC_vector (2 downto 0);
          D : out STD_LOGIC_vector(7 downto 0));
end demux_56;

architecture Behavioral of demux_56 is
begin
process (I, S)
begin
D <= "00000000";
case S is
       when "000" => d(0) <= i;
       when  "001" => d(1) <= i;
       when "010" => d(2) <= i;
       when "011" => d(3) <= i;
       when "100" => d(4) <= i;
       when "101" => d(5) <= i;
       when "110" => d(6) <= i;
       when "111" => d(7) <= i;
       when others => I <= '0';
end case;
end process;
end Behavioral;
```

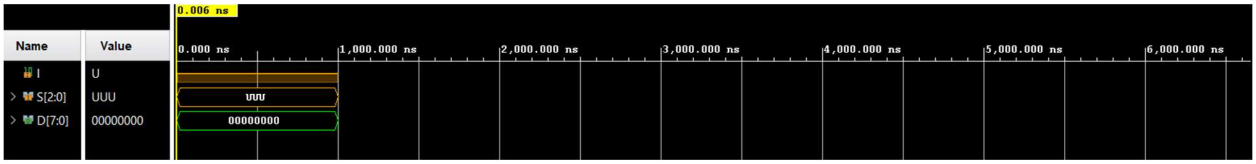
Test-bench Code:

```
library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.Numeric_Std.all;

entity demux_56_tb is
end demux_56_tb;

architecture bench of demux_56_tb is

 component demux_56
    Port ( I : in STD_LOGIC;
           S : inout STD_LOGIC_vector (2 downto 0);
           D : out STD_LOGIC_vector(7 downto 0);
 end component;

 signal I: STD_LOGIC;
 signal S: STD_LOGIC_vector (2 downto 0);
 signal D: STD_LOGIC_vector(7 downto 0;
begin

 uut: demux_56 port map ( I => I,
                         S => S,
                         D => D );

 stimulus: process
 begin
                           wait for 10ns ;
                          d => "10101011";
                          s => "000";
                          wait for 10ns;
                          s => "001";
                          wait for 10ns;
                          s => "010";
                          wait for 10ns;
                          s => "011";
                          wait for 10ns;
                          s => "100";
                          wait for 10ns;
                          s => "101";
                          wait for 10ns;
                          s => "110";
                          wait for 10ns;
                          s => "111";
                          wait for 10ns;

   wait;
 end process;
end behavioral;
```

XDC File

```
set_property PACKAGE_PIN R13 [get_ports {D[7]}]
set_property PACKAGE_PIN J18 [get_ports {D[6]}]
set_property PACKAGE_PIN T18 [get_ports {D[5]}]
set_property PACKAGE_PIN R17 [get_ports {D[4]}]
set_property PACKAGE_PIN R15 [get_ports {D[3]}]
set_property PACKAGE_PIN M13 [get_ports {D[2]}]
set_property PACKAGE_PIN L16 [get_ports {D[1]}]
set_property PACKAGE_PIN J15 [get_ports {D[0]}]
set_property PACKAGE_PIN R16 [get_ports {S[2]}]
set_property PACKAGE_PIN U8 [get_ports {S[1]}]
set_property PACKAGE_PIN T8 [get_ports {S[0]}]
set_property PACKAGE_PIN H17 [get_ports I]
```

Simulation Model



Name	Value
I	U
S[2:0]	UUU
D[7:0]	00000000

Figure 9: Simulation Diagram for 1:8 demultiplexer

RTL Schematic

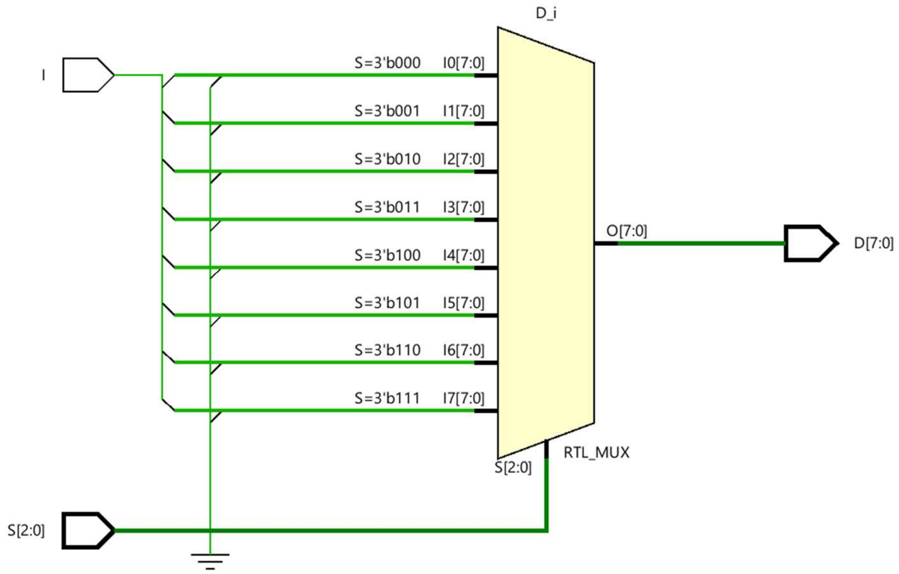
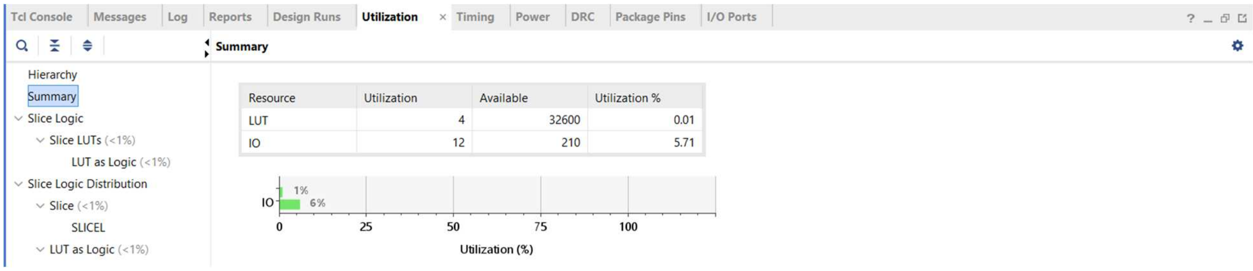


Figure 10: RTL Schematic for 1:8 De-multiplexer

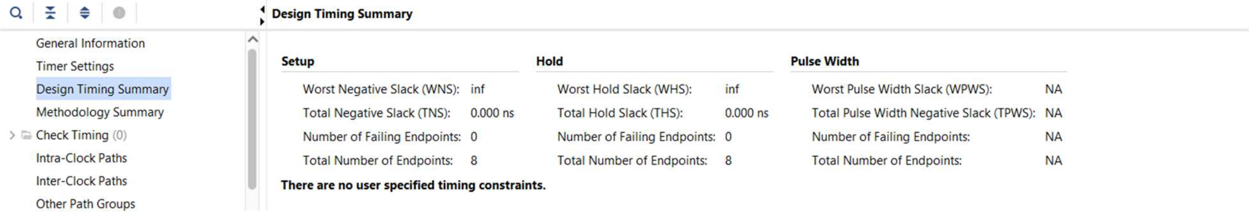
Device Utilization Report



Resource	Utilization	Available	Utilization %
LUT	4	32600	0.01
IO	12	210	5.71

Figure 11: Utilization Report for 1:8 Demultiplexer

Time Analysis Report



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	8	Total Number of Endpoints:	8	Total Number of Endpoints:	NA

There are no user specified timing constraints.

Figure 12: Timing Summary for 1:8 Demultiplexer

Conclusion

In this way, multiplexer, and de-multiplexer was designed in VIVADO software in VHDL language. In VHDL file, under architecture, between begin and end, the obtained expression was assigned to the respective outputs. But before that, selection lines S0, S1 and S2 was assigned. Then, another file was added under "Add Sources" that contained the test bench file. So, the test bench file was created. In the test bench file, all the possible combinations for input was given with each input, a delay of 10ns. Then synthesis along with the implementation was done. After that, simulation was done and the result was obtained in the form of graph. Also, schematic from the RTL Analysis was taken. Then, the device utility report and time analysis report was obtained.