# EXPERIMENT 9

# Multiplexers & De-Multiplexers

*Department of Electrical & Computer Engineering*

## I. OBJECTIVES:

- Examine the functions of multiplexers (MUX) and demultiplexers (DEMUX).
- Create an 8-to-1 MUX in schematic mode and simulate the design.
- Create a 1-of-8 DEMUX in schematic mode and test the design on a target board.
- Compare the characteristics of the 1-of-8 DEMUX with the 3-to-8 decoder.

## II. MATERIALS:

- Xilinx Vivado software, student or professional edition V2018.2 or higher.
- IBM or compatible computer with Pentium III or higher, 128 M-byte RAM or more, and 8 G-byte Or larger hard drive.
- BASYS 3 Board.

## III. DISCUSSION:

A multiplexer (MUX) is a logic circuit that channels two or more input data lines to one output data line. A MUX is also called a data selector. The routing of a particular data input to the output is controlled by the SELECT (or ADDRESS) inputs. Generally speaking, a MUX has N select inputs (address bits), 2N data inputs, and one data output. For example, an 8-to-1 MUX has eight data inputs, three select inputs, and one output.

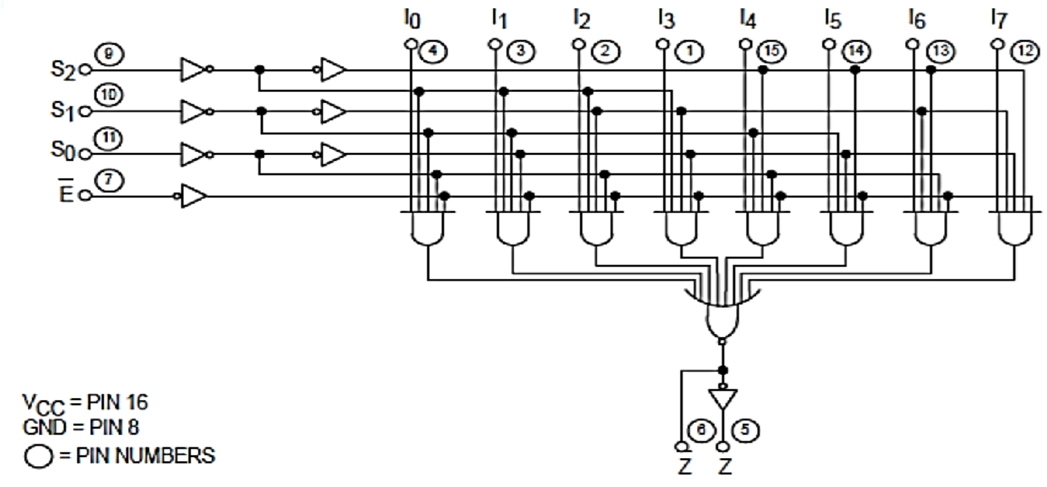
Multiplexers are widely used in digital and data communication systems. They can perform data selection, data routing, operation sequencing, parallel-to-serial conversion, waveform generation, and logic-function generation. Multiplexers make it possible for several streams of digital data to be sent over one physical cable in a system called TDM (time-division-multiplexing) or TDMA (time-division-multiple-access).

A demultiplexer (DEMUX) performs the reverse operation of a MUX. A DEMUX is also called a data distributor. It selects a single data stream (channel) out of the several coming in and routes it to the appropriate output. The channel is chosen by putting its binary address on the select (address) inputs. A DEMUX has one data input, N select inputs, and $2^N$ output lines. For instance, a 1-of-16 DEMUX has one data input, four select inputs, and 16 outputs.

## 1. <u>The 8-to-1 MUX (74151)</u>

The 8-to-1 MUX accepts one of the eight data inputs and passes the data to the output depending on the status of the select lines. Table 10.1 describes the function of this MUX.

**Figure 9.1 Diagram of 74151**



**TRUTH TABLE**

| $\overline{E}$ | $S_2$ | $S_1$ | $S_0$ | $I_0$ | $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ | $I_6$ | $I_7$ | $\overline{Z}$ | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H | X | X | X | X | X | X | X | X | X | X | X | H | L |
| L | L | L | L | L | X | X | X | X | X | X | X | H | L |
| L | L | L | L | H | X | X | X | X | X | X | X | L | H |
| L | L | L | H | X | L | X | X | X | X | X | X | H | L |
| L | L | L | H | X | H | X | X | X | X | X | X | L | H |
| L | L | H | L | X | X | L | X | X | X | X | X | H | L |
| L | L | H | L | X | X | H | X | X | X | X | X | L | H |
| L | L | H | H | X | X | X | L | X | X | X | X | H | L |
| L | L | H | H | X | X | X | H | X | X | X | X | L | H |
| L | H | L | L | X | X | X | X | L | X | X | X | H | L |
| L | H | L | L | X | X | X | X | H | X | X | X | L | H |
| L | H | L | H | X | X | X | X | X | L | X | X | H | L |
| L | H | L | H | X | X | X | X | X | H | X | X | L | H |
| L | H | H | L | X | X | X | X | X | X | L | X | H | L |
| L | H | H | L | X | X | X | X | X | X | H | X | L | H |
| L | H | H | H | X | X | X | X | X | X | X | L | H | L |
| L | H | H | H | X | X | X | X | X | X | X | H | L | H |

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

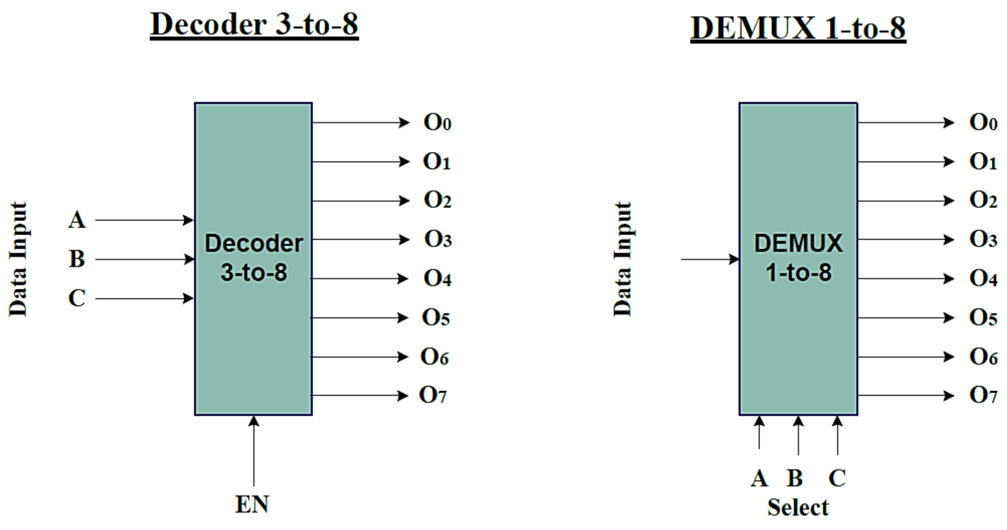
**Table 9.2** Truth Table for the 8-to-1 MUX (74151)

In the truth table, E is the gate (enable) input and A, B, and C are select inputs; $\mathbf{I_0}$ through $\mathbf{I_7}$ are data inputs. Since the enable input is active-low, an input channel can be selected only when the enable line is LOW. An input channel is selected by placing its binary address on the inputs A, B, and C. For example, when **ABC**=110, the output on **Y** will be the $\mathbf{D_6}$ bit stream.

## 2. The 1-of-8 DEMUX

In CPLD Experiment 8, we examined decoders. Given a 3-to-8 decoder with an enable input, we had three data inputs, one enable input, and eight outputs. However, the same circuit can be used for a different function. Specifically, we can use the enable input as a data input, and the three data inputs as select inputs. The result is that the 3-to-8 decoder becomes a 1-of-8 demultiplexer, as illustrated in Figure 10.1 below.

**Figure 9.3**

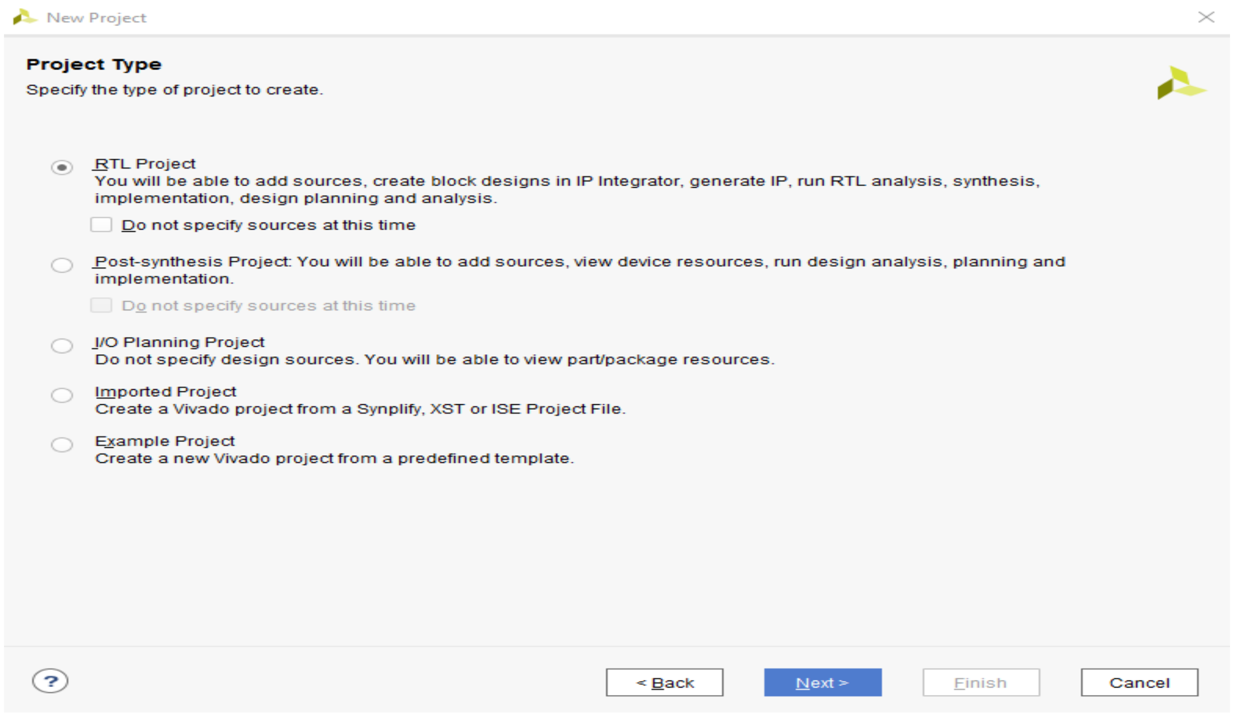


## IV. PROCEDURE:

## Section I. The 8-to-1 MUX

1. Open Xilinix Vivado and in the **Xilinx-Project Navigator** window, Quick start, **New Project**.

2. Choose "RTL Project" and check the "Do not specify sources at this time" as we will configure all the settings manually through the navigator from inside the project.



**Project Type**

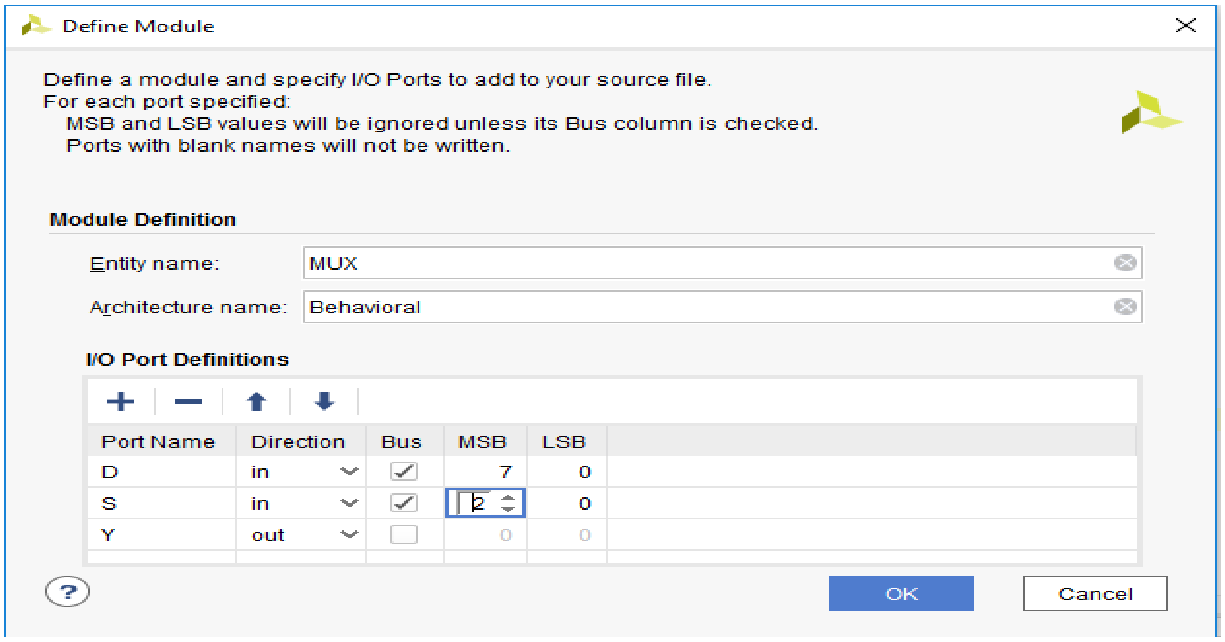
Specify the type of project to create.

- [✓ RTL Project]
  You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
  - ☐ Do not specify sources at this time
- ◯ Post-synthesis Project: You will be able to add sources, view device resources, run design analysis, planning and implementation.
  - ☐ Do not specify sources at this time
- ◯ I/O Planning Project
  Do not specify design sources. You will be able to view part/package resources.
- ◯ Imported Project
  Create a Vivado project from a Synplify, XST or ISE Project File.
- ◯ Example Project
  Create a new Vivado project from a predefined template.

3. Select **New Source…** and the **New** window appears. In the New window, choose Schematic, type your file name (such as ***MUX***) in the File Name editor box, click on OK, and then click on the Next button.

4. In the **Xilinx - Project Navigator** window, select the following
   - Category: "General Purpose"
   - Family: "Artix-7"
   - Package: "cpg236"
   - Speed: "-1"
   - Choose "xc7a35tcpg236-1" that corresponds to the board we are using.
   - Then choose Finish.

5. The Define Module Window that will appear, we will choose the input and output labels for the gates under investigation in this experiment as shown below.



Define Module

Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

**Module Definition**

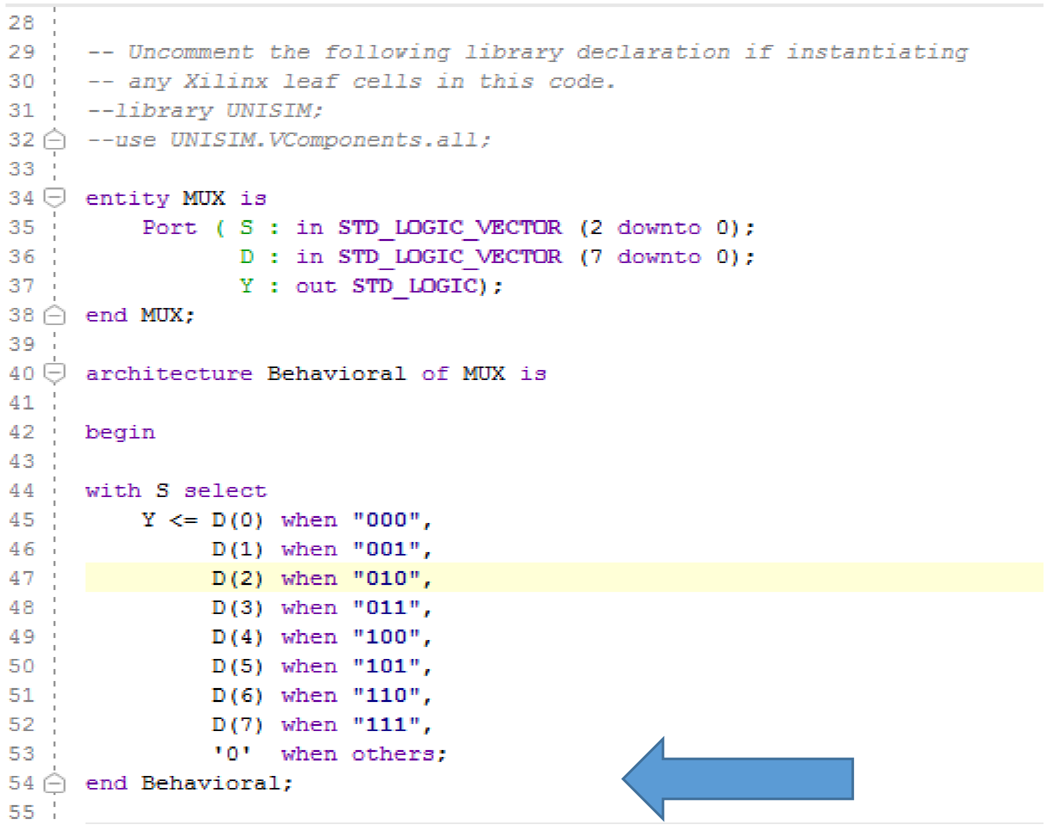
Entity name: MUX

Architecture name: Behavioral

**I/O Port Definitions**

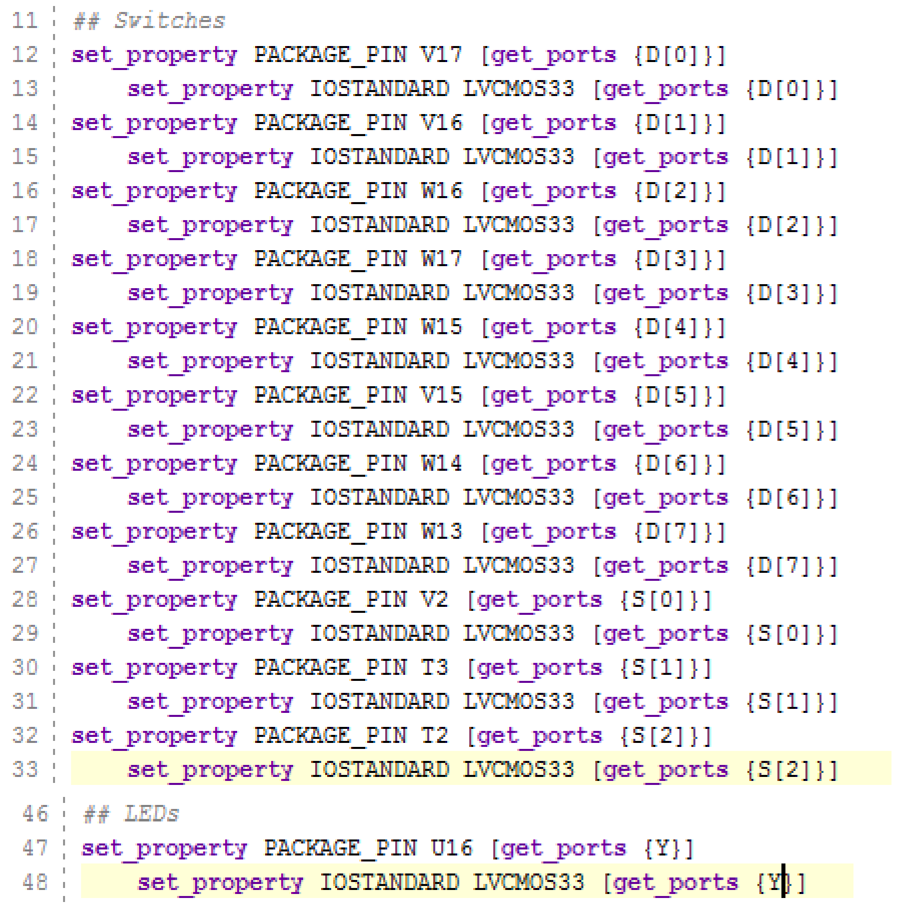
| Port Name | Direction | Bus | MSB | LSB |
|---|---|---|---|---|
| D | in | ✓ | 7 | 0 |
| S | in | ✓ | 2 | 0 |
| Y | out |  | 0 | 0 |

6. In the “MUX.vhd” created file, type the gates equivalent VHDL code for the 8 X 1 MUX between the “begin” and “end Behavioral” as follows and then save the file.



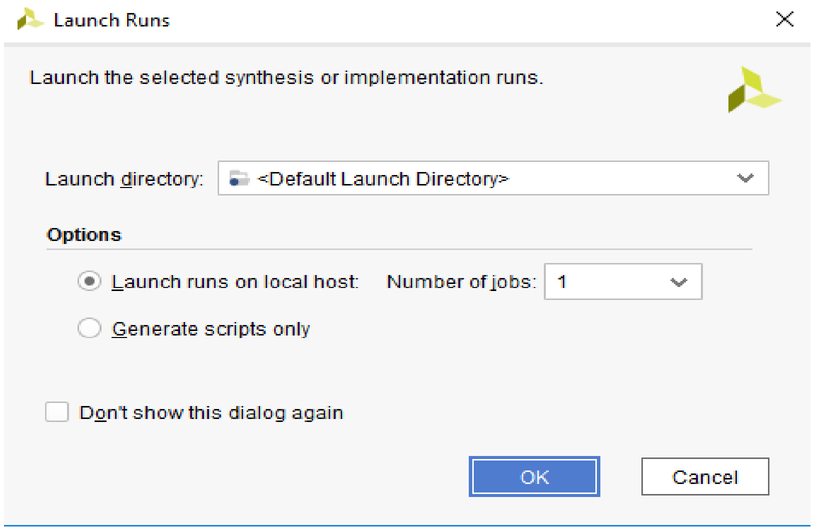
```

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

entity MUX is
    Port ( S : in STD_LOGIC_VECTOR (2 downto 0);
           D : in STD_LOGIC_VECTOR (7 downto 0);
           Y : out STD_LOGIC);
end MUX;

architecture Behavioral of MUX is

begin

with S select
    Y <= D(0) when "000",
         D(1) when "001",
         D(2) when "010",
         D(3) when "011",
         D(4) when "100",
         D(5) when "101",
         D(6) when "110",
         D(7) when "111",
         '0'  when others;
end Behavioral;

```

7. Next, we need to add To add a constraint file with the".xdc" extension, as following:



```
## Switches
set_property PACKAGE_PIN V17 [get_ports {D[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[0]}]
set_property PACKAGE_PIN V16 [get_ports {D[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[1]}]
set_property PACKAGE_PIN W16 [get_ports {D[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[2]}]
set_property PACKAGE_PIN W17 [get_ports {D[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[3]}]
set_property PACKAGE_PIN W15 [get_ports {D[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[4]}]
set_property PACKAGE_PIN V15 [get_ports {D[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[5]}]
set_property PACKAGE_PIN W14 [get_ports {D[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[6]}]
set_property PACKAGE_PIN W13 [get_ports {D[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[7]}]
set_property PACKAGE_PIN V2 [get_ports {S[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {S[0]}]
set_property PACKAGE_PIN T3 [get_ports {S[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {S[1]}]
set_property PACKAGE_PIN T2 [get_ports {S[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {S[2]}]

## LEDs
set_property PACKAGE_PIN U16 [get_ports {Y}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Y}]
```

8. From the tool tab choose the play button and then "Run Implementation". Select "Number of jobs" =1 and then press OK.



Launch Runs

Launch the selected synthesis or implementation runs.

Launch directory: <Default Launch Directory>

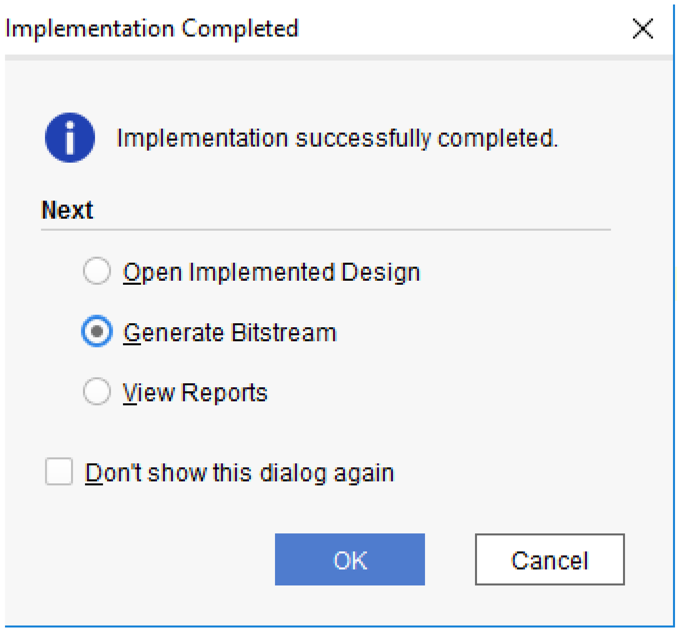
**Options**

[✓ Launch runs on local host:] Number of jobs: 1

Generate scripts only

Don't show this dialog again

9. The implementation errors window will appear if any or the successfully completed window. From this window select “Generate Bitstream” and then OK. This will make the software generate “.bin” file to be used in programing the hardware BAYAS 3.



Implementation Completed

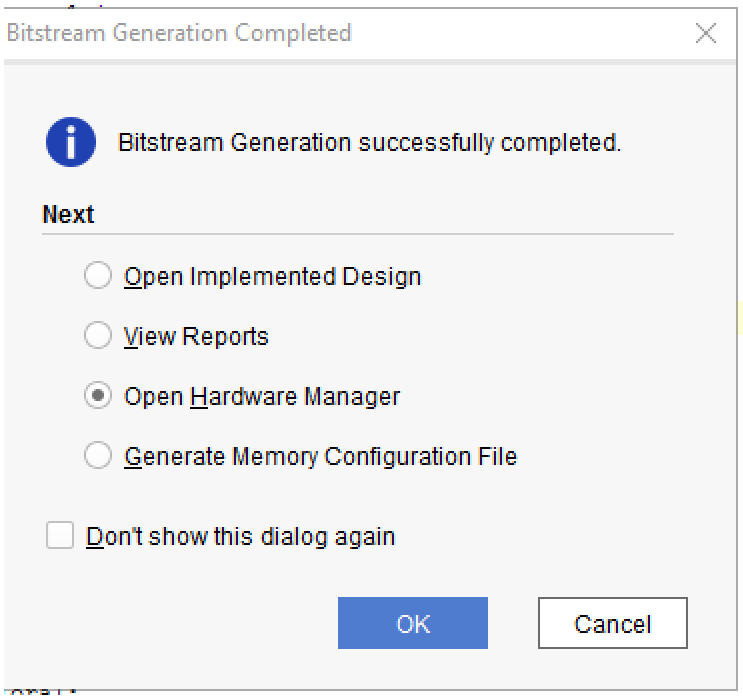
Implementation successfully completed.

**Next**

- ( ) Open Implemented Design
- [✓ Generate Bitstream]
- ( ) View Reports

☐ Don't show this dialog again

10. The next window will appear in which choose “Open Hardware Manger”, connect the Hardware Kit to the USB port and then press OK.



Bitstream Generation Completed

Bitstream Generation successfully completed.

**Next**

- ( ) Open Implemented Design
- ( ) View Reports
- [✓ Open Hardware Manager]
- ( ) Generate Memory Configuration File

☐ Don't show this dialog again

11. From the window appears, select the ".bin" file from the Project you create by browsing for the generated ".bit file" under the ".runs" folder and program the board then press OK.

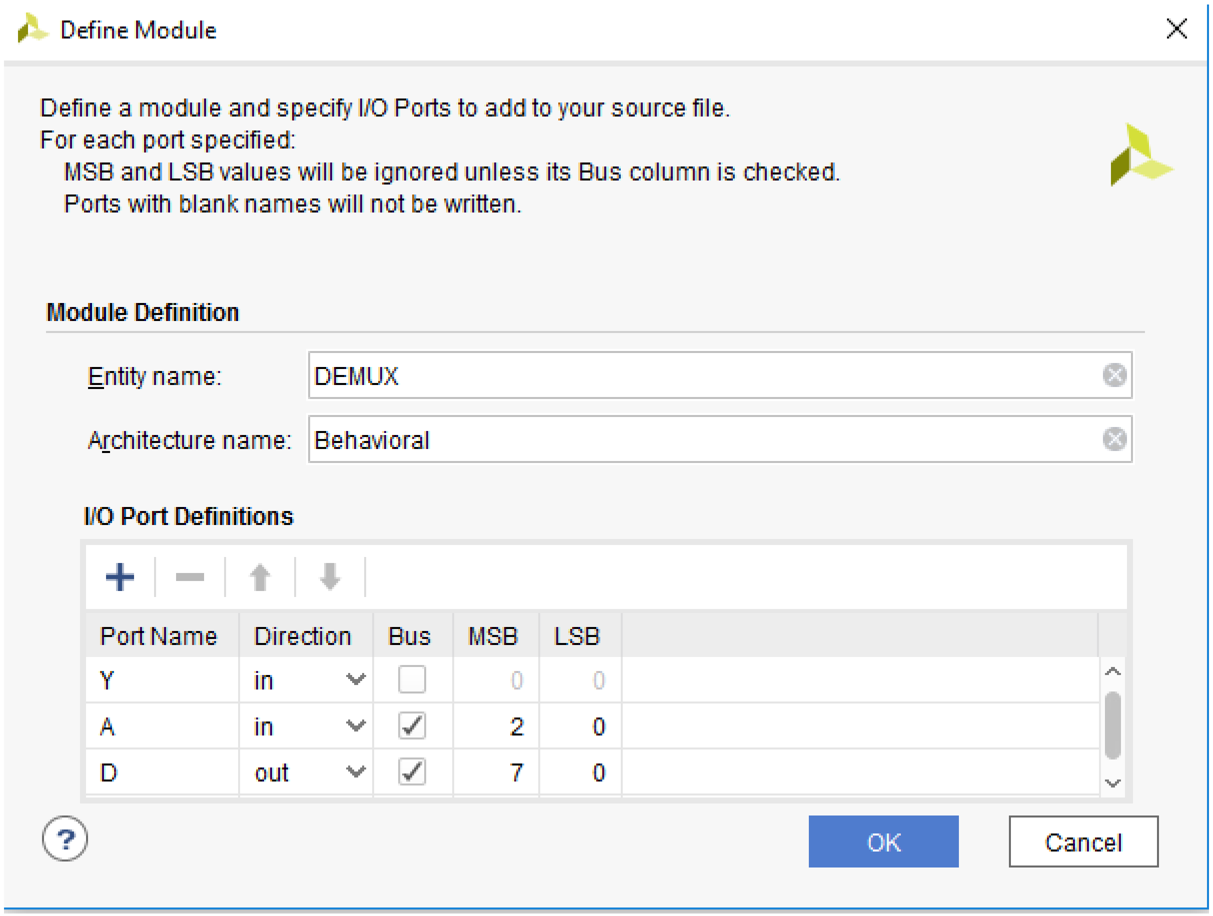
12. Complete the following truth table based on the output from your board

| Selection | | | Data Inputs | | | | | | | | Outputs | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S2** | **S1** | **S0** | **D0** | **D1** | **D2** | **D3** | **D4** | **D5** | **D6** | **D7** | **Y** | **W=Y'** |
| x | x | x | x | x | x | x | x | x | x | x | 0 | 1 |
| 0 | 0 | 0 | 0 | X | X | X | X | X | X | X | 0 | 1 |
| 0 | 0 | 0 | 1 | X | X | X | X | X | X | X | 1 | 0 |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |

Checked by_______________________________ Date ____________

## Section II. The 1-to-8 DEMUX

1. Repeat steps from 1 to 5 in section I
2. The Define Module Window that will appear, we will choose the input and output labels for the gates under investigation in this experiment as shown below.



Define Module

Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

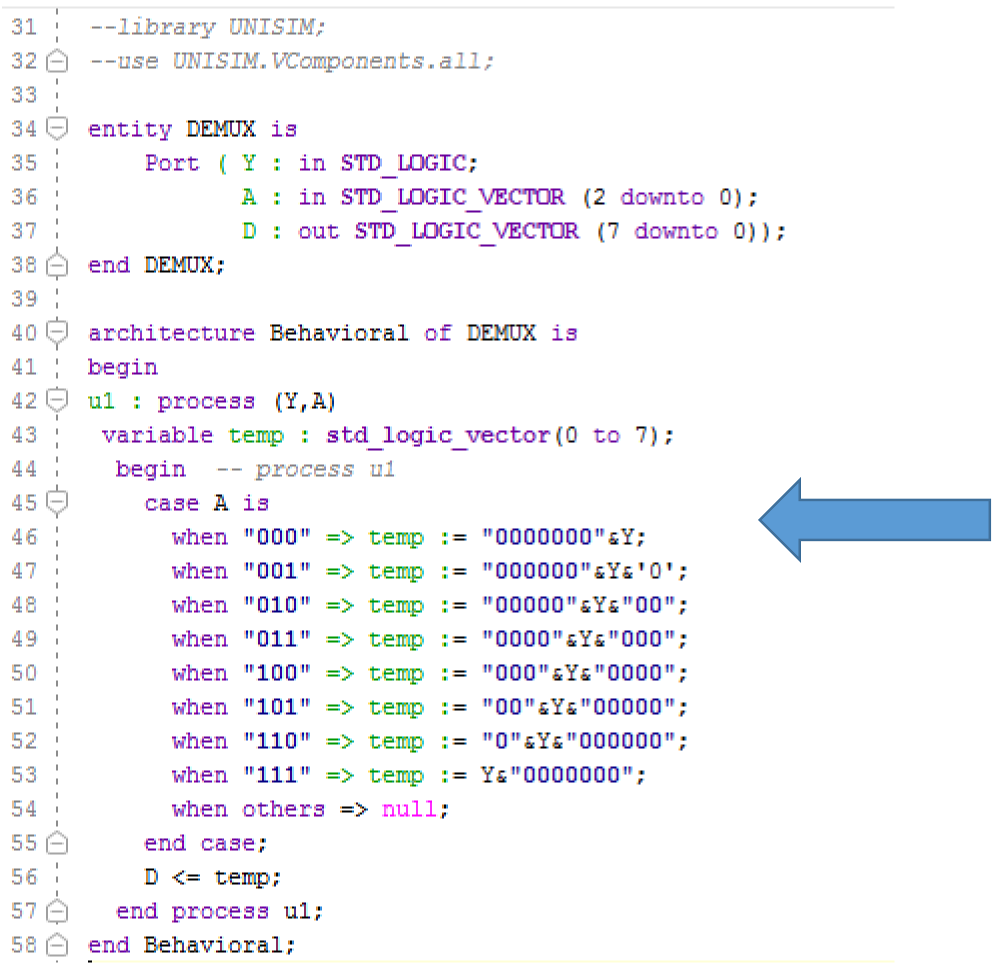
Entity name: DEMUX

Architecture name: Behavioral

I/O Port Definitions

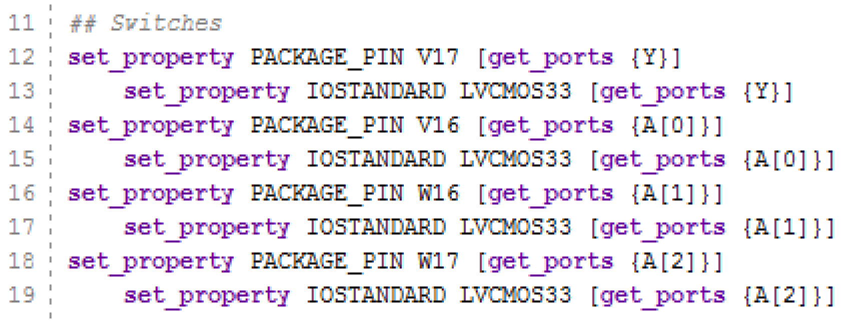
| Port Name | Direction | Bus | MSB | LSB |
|---|---|---|---|---|
| Y | in | ☐ | 0 | 0 |
| A | in | ✓ | 2 | 0 |
| D | out | ✓ | 7 | 0 |

3. In the "DEMUX.vhd" created file, type the gates equivalent VHDL code for the 1-to-8 DEMUX between the "begin" and "end Behavioral" as follows and then save the file.

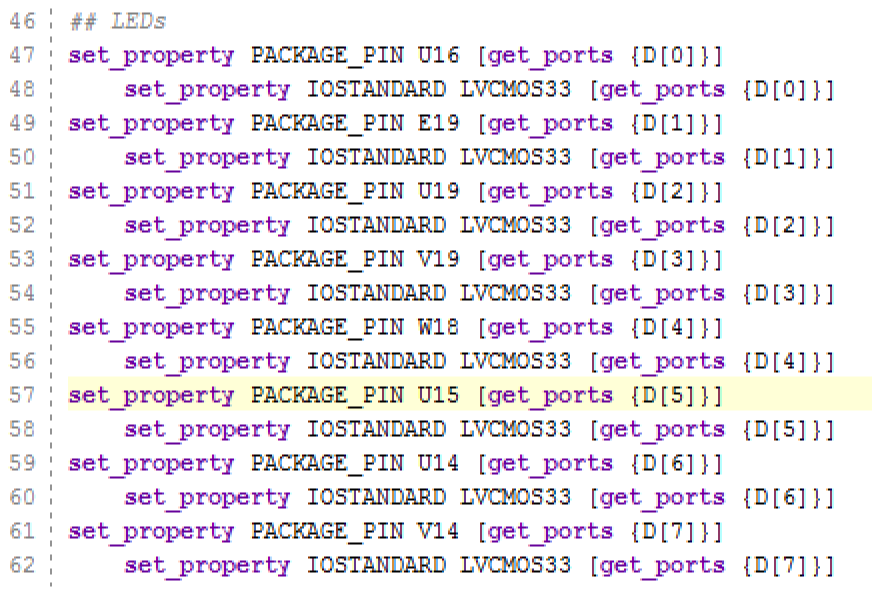


```vhdl
--library UNISIM;
--use UNISIM.VComponents.all;

entity DEMUX is
    Port ( Y : in STD_LOGIC;
           A : in STD_LOGIC_VECTOR (2 downto 0);
           D : out STD_LOGIC_VECTOR (7 downto 0));
end DEMUX;

architecture Behavioral of DEMUX is
begin
u1 : process (Y,A)
 variable temp : std_logic_vector(0 to 7);
  begin  -- process u1
    case A is
      when "000" => temp := "0000000"&Y;
      when "001" => temp := "000000"&Y&'0';
      when "010" => temp := "00000"&Y&"00";
      when "011" => temp := "0000"&Y&"000";
      when "100" => temp := "000"&Y&"0000";
      when "101" => temp := "00"&Y&"00000";
      when "110" => temp := "0"&Y&"000000";
      when "111" => temp := Y&"0000000";
      when others => null;
    end case;
    D <= temp;
  end process u1;
end Behavioral;
```

4. Next, we need to add To add a constraint file with the".xdc" extension, as following:



```
## Switches
set_property PACKAGE_PIN V17 [get_ports {Y}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Y}]
set_property PACKAGE_PIN V16 [get_ports {A[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
set_property PACKAGE_PIN W16 [get_ports {A[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]
set_property PACKAGE_PIN W17 [get_ports {A[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[2]}]
```



```
## LEDs
set_property PACKAGE_PIN U16 [get_ports {D[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[0]}]
set_property PACKAGE_PIN E19 [get_ports {D[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[1]}]
set_property PACKAGE_PIN U19 [get_ports {D[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[2]}]
set_property PACKAGE_PIN V19 [get_ports {D[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[3]}]
set_property PACKAGE_PIN W18 [get_ports {D[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[4]}]
set_property PACKAGE_PIN U15 [get_ports {D[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[5]}]
set_property PACKAGE_PIN U14 [get_ports {D[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[6]}]
set_property PACKAGE_PIN V14 [get_ports {D[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[7]}]
```

5. Then “Run Implementation”. Select ”Number of jobs” =1 and then press OK. The implementation errors window will appear if any or the successfully completed window. From this window select “Generate Bitstream” and then OK. This will make the software generate “.bin” file to be used in programing the hardware BAYAS 3.
6. The next window will appear in which choose “Open Hardware Manger”, connect the Hardware Kit to the USB port and then press OK.From the window appears, select the “.bin” file from the Project you create by browsing for the generated “.bit file” under the “.runs” folder and program the board then press OK.

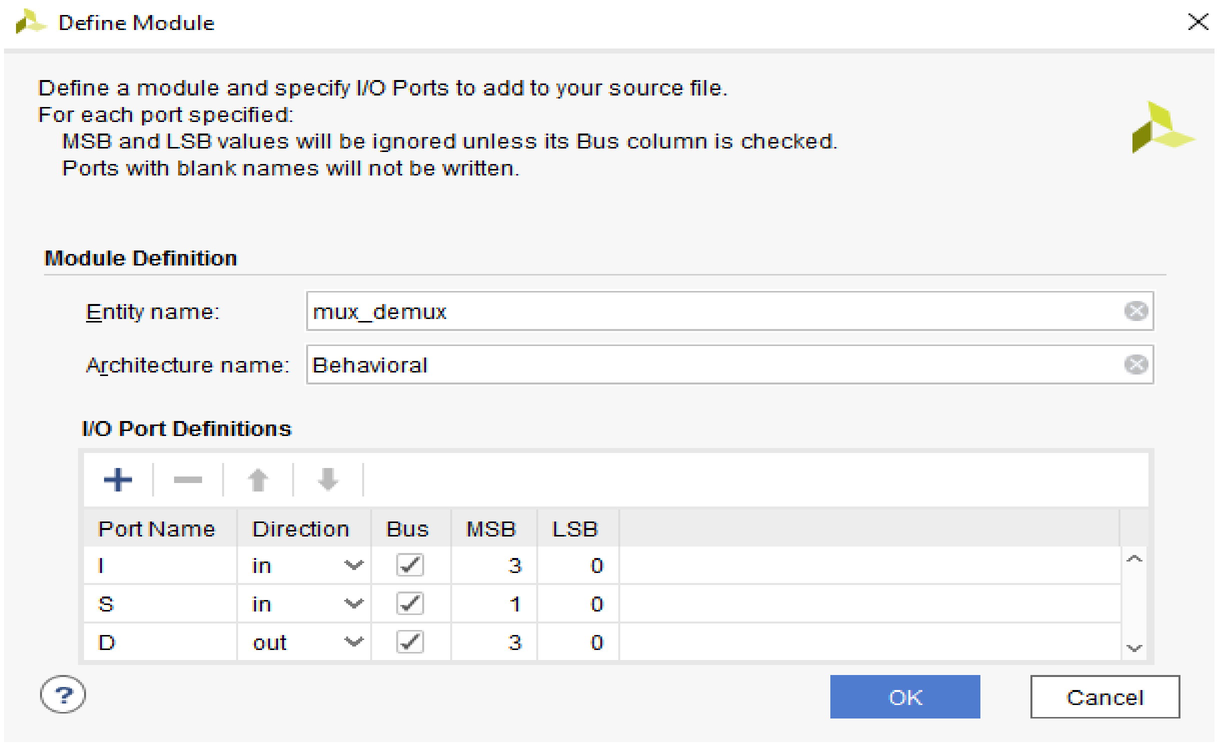
7. Complete the following truth table based on the output from your board

| Data in | Selection | | | Data Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Y | A2 | A1 | A0 | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
| 0 | 0 | 0 | 0 | 0 | X | X | X | X | X | X | X |
| 1 | 0 | 0 | 0 | 1 | X | X | X | X | X | X | X |
| 0 | 0 | 0 | 1 | X | 0 | X | X | X | X | X | X |
| 1 | 0 | 0 | 1 | X | 1 | X | X | X | X | X | X |
| 0 | | | | | | | | | | | |
| 1 | | | | | | | | | | | |
| 0 | | | | | | | | | | | |
| 1 | | | | | | | | | | | |
| 0 | | | | | | | | | | | |
| 1 | | | | | | | | | | | |
| 0 | | | | | | | | | | | |
| 1 | | | | | | | | | | | |
| 0 | | | | | | | | | | | |
| 1 | | | | | | | | | | | |
| 0 | | | | | | | | | | | |
| 1 | | | | | | | | | | | |

Checked by________________________________ Date ____________

## Section III. MUX and DEMUX in One Circuit

1. Repeat steps from 1 to 5 in section I
2. The Define Module Window that will appear, we will choose the input and output labels for the gates under investigation in this experiment as shown below.



Define a module and specify I/O Ports to add to your source file.
For each port specified:
  MSB and LSB values will be ignored unless its Bus column is checked.
  Ports with blank names will not be written.

**Module Definition**

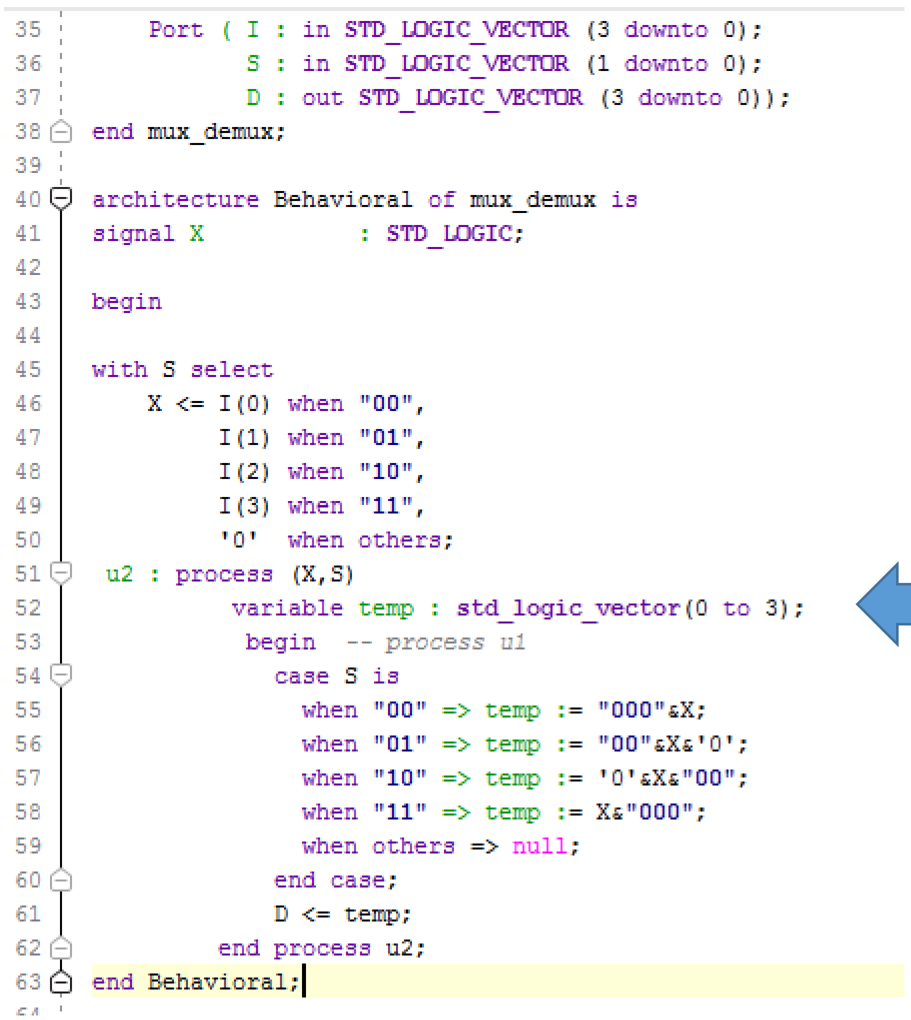
Entity name: mux_demux

Architecture name: Behavioral

**I/O Port Definitions**

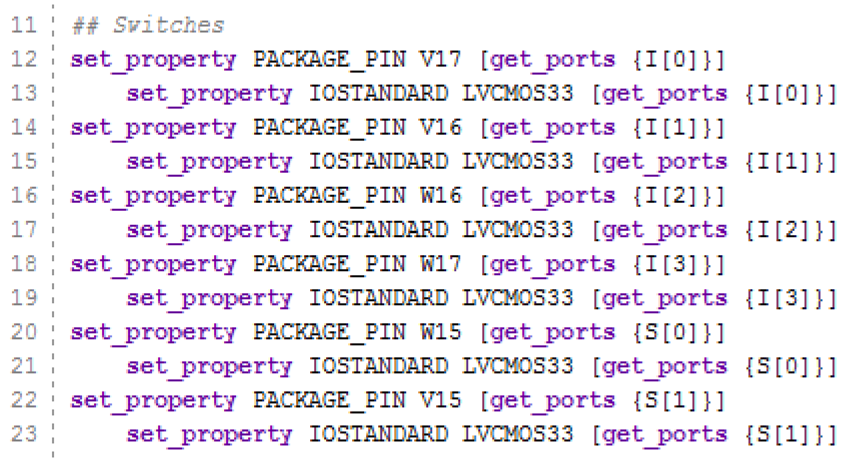
| Port Name | Direction | Bus | MSB | LSB |
|---|---|---|---|---|
| I | in | ✓ | 3 | 0 |
| S | in | ✓ | 1 | 0 |
| D | out | ✓ | 3 | 0 |

3. In the “mux_demux.vhd” created file, type the gates equivalent VHDL code for the 1-to-8 DEMUX between the “begin” and “end Behavioral” as follows and then save the file.

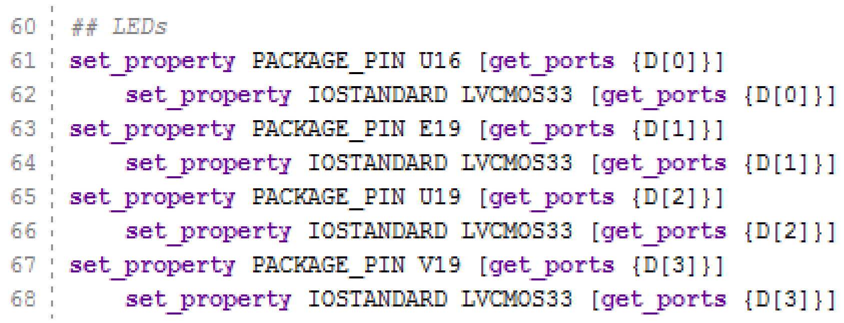


```vhdl
    Port ( I : in STD_LOGIC_VECTOR (3 downto 0);
           S : in STD_LOGIC_VECTOR (1 downto 0);
           D : out STD_LOGIC_VECTOR (3 downto 0));
end mux_demux;

architecture Behavioral of mux_demux is
signal X            : STD_LOGIC;

begin

with S select
    X <= I(0) when "00",
         I(1) when "01",
         I(2) when "10",
         I(3) when "11",
         '0'  when others;
 u2 : process (X,S)
          variable temp : std_logic_vector(0 to 3);
           begin  -- process u1
             case S is
               when "00" => temp := "000"&X;
               when "01" => temp := "00"&X&'0';
               when "10" => temp := '0'&X&"00";
               when "11" => temp := X&"000";
               when others => null;
             end case;
             D <= temp;
         end process u2;
end Behavioral;
```

4. Next, we need to add To add a constraint file with the".xdc" extension, as following:



```
## Switches
set_property PACKAGE_PIN V17 [get_ports {I[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {I[0]}]
set_property PACKAGE_PIN V16 [get_ports {I[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {I[1]}]
set_property PACKAGE_PIN W16 [get_ports {I[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {I[2]}]
set_property PACKAGE_PIN W17 [get_ports {I[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {I[3]}]
set_property PACKAGE_PIN W15 [get_ports {S[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {S[0]}]
set_property PACKAGE_PIN V15 [get_ports {S[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {S[1]}]
```



```
## LEDs
set_property PACKAGE_PIN U16 [get_ports {D[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[0]}]
set_property PACKAGE_PIN E19 [get_ports {D[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[1]}]
set_property PACKAGE_PIN U19 [get_ports {D[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[2]}]
set_property PACKAGE_PIN V19 [get_ports {D[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[3]}]
```

5. Then “Run Implementation”. Select ”Number of jobs” =1 and then press OK. The implementation errors window will appear if any or the successfully completed window. From this window select “Generate Bitstream” and then OK. This will make the software generate “.bin” file to be used in programing the hardware BAYAS 3.
6. The next window will appear in which choose “Open Hardware Manger”, connect the Hardware Kit to the USB port and then press OK.From the window appears, select the “.bin” file from the Project you create by browsing for the generated “.bit file” under the “.runs” folder and program the board then press OK.

7. List the function table (truth table) and analyze the operation.

Checked by______________________________ Date ___________

## QUESTIONS

1. Name two applications of MUXs.

2. Given a 4-to-16 decoder with an enable line. Draw a block diagram showing how this decoder can be used as a 1-to-16 DEMUX.

3. Given two 8-to-1 MUXs with enable lines, how do you make a 16-to-1 MUX? Draw the block diagram.

4. Given a 4-to-1 MUX (M4_1E in the symbol library), how do you obtain the Boolean function: ,

$$O = \overline{S1}\ S0 + S1\ \overline{S0}$$ ,

Show the external connection to the MUX. (Tips: Try to add the control circuits at the output, instead of inputs.)