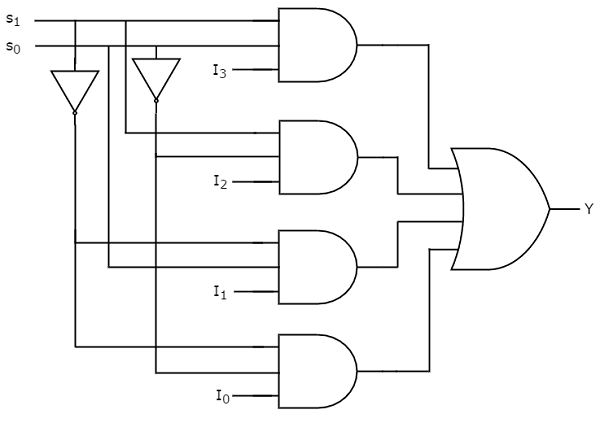
EXPERIMENT 4

**AIM:** To design and simulate MUX and DEMUX using VHDL in ModelSim.

**Theory:** In electronics, a **multiplexer** (or **mux**) is a device that selects between several analog or digital input signals and forwards it to a single output line. A multiplexer of n 2 n {\displaystyle 2^{n}} inputs has 2n n {\displaystyle n} select lines, which are used to select which input line to send to the output. Multiplexers are mainly used to increase the amount of data that can be sent over the network within a certain amount of time and bandwidth. A multiplexer is also called a **data selector**. Multiplexers can also be used to implement Boolean functions of multiple variables.

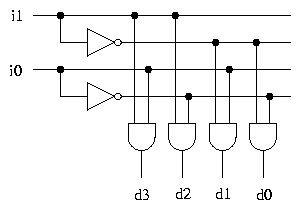


|  |  |  |
| --- | --- | --- |
| **INPUT** | | **OUTPUT** |
| **S1** | **S0** | **Y** |
| **0** | **0** | **D0** |
| **0** | **1** | **D1** |
| **1** | **0** | **D2** |
| **1** | **1** | **D3** |

The process of getting information from one input and transmitting the same over one of many outputs is called demultiplexing. A demultiplexer is a combinational logic circuit that receives the information on a single input and transmits the same information over one of 2n possible output lines.

The bit combinations of the select lines control the selection of specific output line to be connected to the input at given instant.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **DATA** | **SELECT LINES** | | **OUTPUTS** | | | |
| **D** | **S1** | **S0** | **D0** | **D1** | **D2** | **D3** |
| **D** | **0** | **0** | **0** | **0** | **0** | **D** |
| **D** | **0** | **1** | **0** | **0** | **D** | **0** |
| **D** | **1** | **0** | **0** | **D** | **0** | **0** |
| **D** | **1** | **1** | **D** | **0** | **0** | **0** |



**VHDL CODE: 4-to-1 Multiplexer**

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

USE ieee.std\_logic\_arith.all;

ENTITY MULTIPLEXER\_4\_1 IS

port(A,B,C,D,S0,S1: in STD\_LOGIC;Z: out STD\_LOGIC);

END ENTITY MULTIPLEXER\_4\_1;

--

ARCHITECTURE MULTIPLEXER\_4\_1\_DATAFLOW OF MULTIPLEXER\_4\_1 IS

BEGIN

Z<=(((NOT S1)AND(NOT S0)AND A) OR ((NOT S1)AND S0 AND B) OR(S1 AND (NOT S0)AND C) OR(S1 AND S0 AND D));

END ARCHITECTURE MULTIPLEXER\_4\_1\_DATAFLOW;

ARCHITECTURE MULTIPLEXER\_4\_1\_BEHAV OF MULTIPLEXER\_4\_1 IS

BEGIN

PROCESS (A,B,C,D,S0,S1) IS

BEGIN

if (S0 ='0' and S1 = '0') then

Z <= A;

elsif (S0 ='1' and S1 = '0') then

Z <= B;

elsif (S0 ='0' and S1 = '1') then

Z <= C;

else

Z <= D;

end if;

END PROCESS;

END ARCHITECTURE MULTIPLEXER\_4\_1\_BEHAV;

ARCHITECTURE MULTIPLEXER\_4\_1\_STRUCTURAL OF MULTIPLEXER\_4\_1 IS

component AND\_GATE

port(A,B:in std\_logic;Z: out std\_logic);

end component;

component OR\_GATE

port(A,B:in std\_logic;Z: out std\_logic);

end component;

component NOT\_GATE

port(A:in std\_logic;Z: out std\_logic);

end component;

SIGNAL var,var1: std\_logic\_vector(4 downto 0);

SIGNAL S0BAR,S1BAR: std\_logic;

BEGIN

N1: NOT\_GATE port map (S1,S1BAR);

N2: NOT\_GATE port map (S0,S0BAR);

A1: AND\_GATE port map (S1BAR,S0BAR,var(0));

A2: AND\_GATE port map (var(0),A,var1(0));

A3: AND\_GATE port map (S1BAR,S0,var(1));

A4: AND\_GATE port map (var(1),B,var1(1));

A5: AND\_GATE port map (S1,S0BAR,var(2));

A6: AND\_GATE port map (var(2),C,var1(2));

A7: AND\_GATE port map (S1,S0,var(3));

A8: AND\_GATE port map (var(3),D,var1(3));

O1: OR\_GATE port map (var1(0),var1(1),var(4));

O2: OR\_GATE port map (var1(2),var1(3),var1(4));

O3: OR\_GATE port map (var(4),var1(4),Z);

END ARCHITECTURE MULTIPLEXER\_4\_1\_STRUCTURAL;

**DEMUX CODE:**

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

USE ieee.std\_logic\_arith.all;

ENTITY DEMUX\_1\_4 IS

port ( A,S1,S0: in std\_logic;Z:out std\_logic\_vector(3 downto 0) );

END ENTITY DEMUX\_1\_4;

--

ARCHITECTURE DEMUX\_1\_4\_STRUCTURAL OF DEMUX\_1\_4 IS

component AND\_GATE

port(A,B:in std\_logic;Z: out std\_logic);

end component;

component NOT\_GATE

port(A:in std\_logic;Z: out std\_logic);

end component;

SIGNAL var: std\_logic\_vector(3 downto 0);

SIGNAL S0BAR,S1BAR: std\_logic;

BEGIN

N1: NOT\_GATE port map (S1,S1BAR);

N2: NOT\_GATE port map (S0,S0BAR);

A1: AND\_GATE port map (S1BAR,S0BAR,var(0));

A2: AND\_GATE port map (var(0),A,Z(0));

A3: AND\_GATE port map (S1BAR,S0,var(1));

A4: AND\_GATE port map (var(1),A,z(1));

A5: AND\_GATE port map (S1,S0BAR,var(2));

A6: AND\_GATE port map (var(2),A,Z(2));

A7: AND\_GATE port map (S1,S0,var(3));

A8: AND\_GATE port map (var(3),A,Z(3));

END ARCHITECTURE DEMUX\_1\_4\_STRUCTURAL;

ARCHITECTURE DEMUX\_1\_4\_DATAFLOW OF DEMUX\_1\_4 IS

BEGIN

Z(0)<=(NOT S0)AND (NOT S1) AND A;

Z(1)<=(NOT S0)AND S1 AND A;

Z(2)<= S0 AND (NOT S1) AND A;

Z(3)<= S0 AND S1 AND A;

END ARCHITECTURE DEMUX\_1\_4\_DATAFLOW;

ARCHITECTURE DEMUX\_1\_4\_BEHAV OF DEMUX\_1\_4 IS

BEGIN

process (F,S0,S1) is

begin

if (S0 ='0' and S1 = '0') then

Z(0) <= A;

elsif (S0 ='1' and S1 = '0') then

Z(1) <= A;

elsif (S0 ='0' and S1 = '1') then

Z(2) <= A;

else

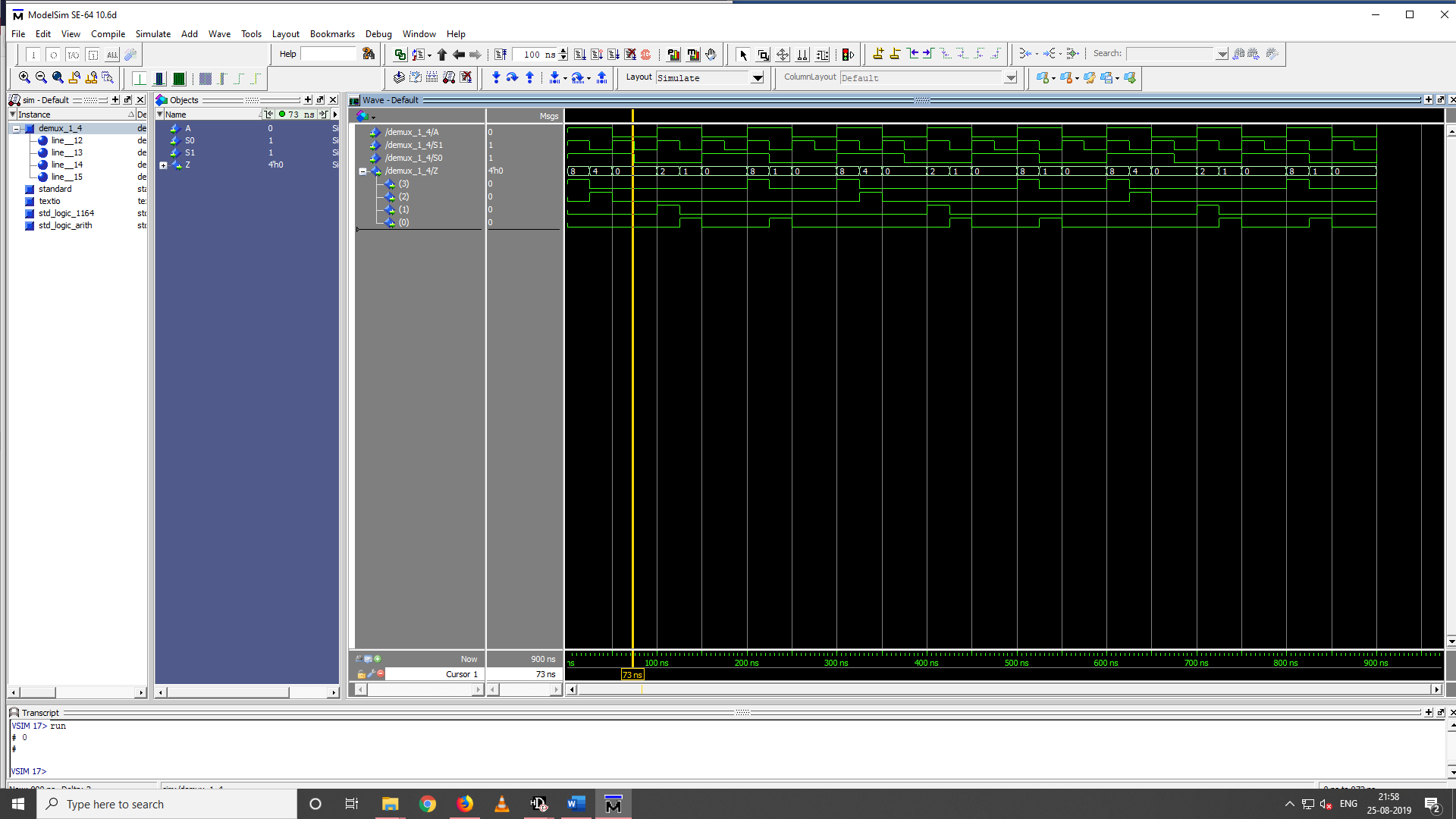
Z(3) <= A;

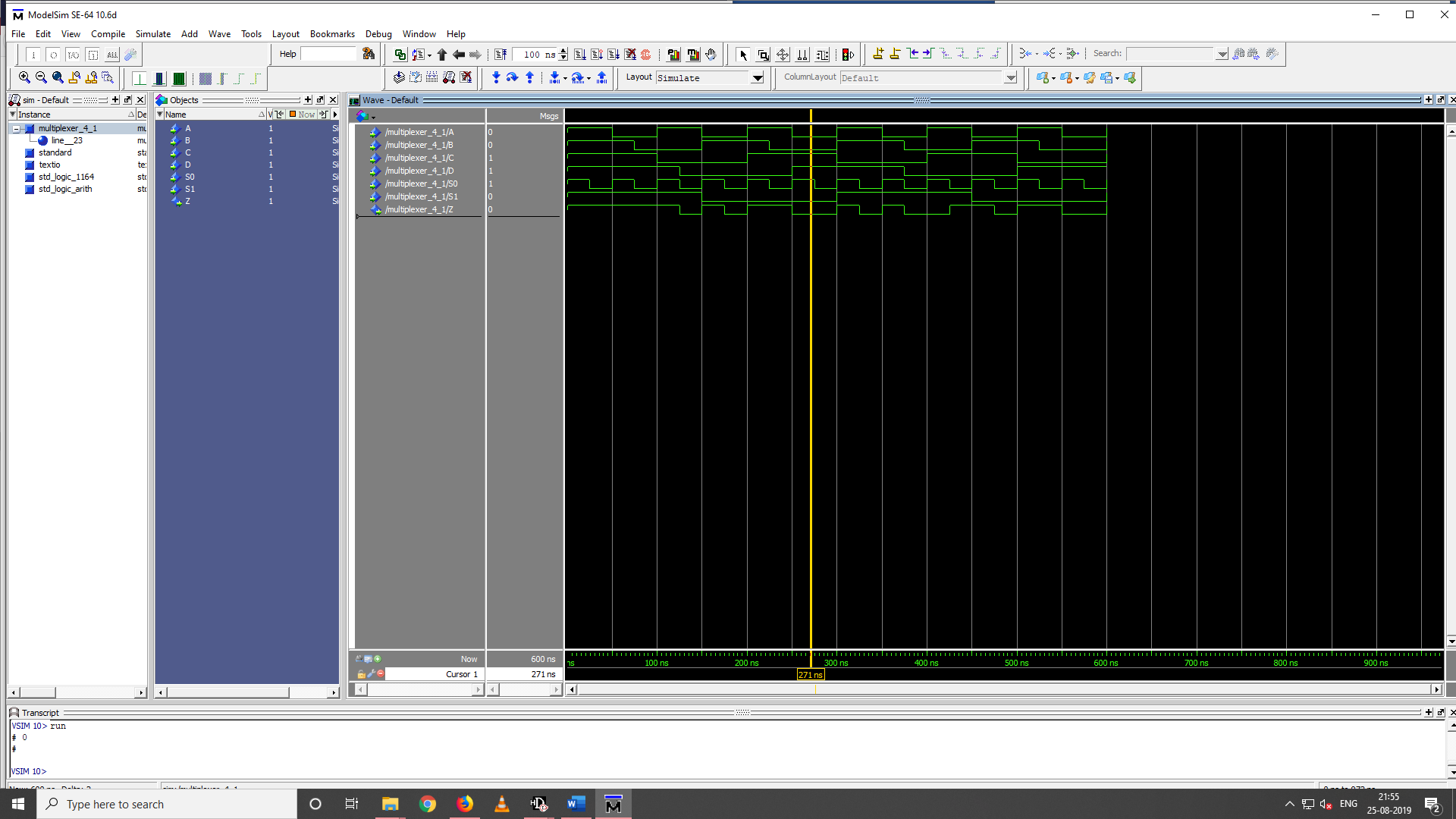
end if;

end process;

END ARCHITECTURE DEMUX\_1\_4\_BEHAV;

**OUTPUTS:**



**DEMUX**

**MUX**

**RESULT:** **MUX and DEMUX had been successfully simulated using VHDL in**

**ModelSim.**