Review Laboratory Exercise 2: MUX, DECODER, and ENCODER Basic Circuit Design and Testing

Zi Xuan Li

The Grove School of Engineering, The City College of New York

CSC 34300 5DE[43223]: Computer Systems Design Laboratory

Professor Gertner, TA Albi Arapi

March 6th, 2024

Table of Contents

5-to-1 Multiplexer (using AND, OR, and NOT gates)

Functionality and Specifications

What is the combinational logic function (Boolean function) of the circuit?

What is the VHDL code (Boolean function) of the circuit?

Include a screenshot of your design steps and of final the circuit.

Simulation

Include a screenshot of the vector waveform output file for all possible inputs.

Draw a truth table from the waveforms.

3-to-8 Decoder

Functionality and Specifications

What is the combinational logic function (Boolean function) of the circuit?

What is the VHDL code (Boolean function) of the circuit?

Include a screenshot of your design steps and of final the circuit.

Simulation

Include a screenshot of the vector waveform output file for all possible inputs.

Draw a truth table from the waveforms.

8-to-3 Priority Encoder

Functionality and Specifications

What is the combinational logic function (Boolean function) of the circuit?

What is the VHDL code (Boolean function) of the circuit?

Include a screenshot of your design steps and of final the circuit.

Simulation

Include a screenshot of the vector waveform output file for all possible inputs.

Draw a truth table from the waveforms.

1-to-2 Demultiplexer

Functionality and Specifications

What is the combinational logic function (Boolean function) of the circuit?

What is the VHDL code (Boolean function) of the circuit?

Include a screenshot of your design steps and of final the circuit.

Simulation

Include a screenshot of the vector waveform output file for all possible inputs.

Draw a truth table from the waveforms.

Generated VHDL code from Quartus

Conclusion

Objective: The goal of this exercise is to design and verify the functionality of our own 4:1 mux, 2:1 mux with an 8-bit input and output, 5:1 mux, 3:8 decoder, 8:3 encoder, and 1:2 demux.

4-to-1 Multiplexer

Functionality and Specifications:

What is the combinational logic function (Boolean function) of the circuit?

A 4-to-1 multiplexer (mux) has 4 data inputs (D0, D1, D2, D3), 1 output (Y), and 2 select lines (S1, S0). The output Y is determined by the select lines as follows:

$$Y = (D0 \cdot \overline{S1} \cdot \overline{S0}) + (D1 \cdot S1 \cdot \overline{S0}) + (D2 \cdot \overline{S1} \cdot S0) + (D3 \cdot S1 \cdot S0)$$

What is the VHDL code (Boolean function) of the circuit?

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

entity mux4to1_vhdl is
    Port (
        A, B, C, D : in STD_LOGIC;
        S : in STD_LOGIC_VECTOR(1 downto 0);
        Z : out STD_LOGIC);
end mux4to1_vhdl;

architecture Behavioral of mux4to1_vhdl is

begin
   process (S)
   begin
      case S is
         when "00" =>
            Z <= A;
         when "01" =>
            Z <= B;
         when "10" =>
            Z <= C;
         when "11" =>
            Z <= D;
         when others =>
            Z <= '0';
      end case;
   end process;
end Behavioral;
```

Figure 1: VHDL code snippet of 4 to 1 multiplexer.

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity mux4to1_TB is
end entity mux4to1_TB;

architecture TestBenchArchitecture of mux4to1_TB is
component mux4to1_vhdl is
    Port (
        A, B, C, D : in STD_LOGIC;
        S : in STD_LOGIC_VECTOR(1 downto 0);
        Z : out STD_LOGIC);
   end component;

   signal A_tb, B_tb, C_tb, D_tb, Z_tb: STD_LOGIC;

   signal S_tb : STD_LOGIC_VECTOR(1 downto 0);

begin
   uut: mux4to1_vhdl
      port map (
         A => A_tb,
         B => B_tb,
         C => C_tb,
         D => D_tb,
         S => S_tb,
         Z => Z_tb);

   stimulus_process: process
   begin
      wait for 10 ns;

      A_tb <= '1';
      B_tb <= '0';
      C_tb <= '1';
      D_tb <= '0';

      S_tb <= "00";
      wait for 10 ns;
      assert Z_tb = '1' report "The output is not correct" severity failure;

      S_tb <= "01";
      wait for 10 ns;
      assert Z_tb = '0' report "The output is not correct" severity failure;

      S_tb <= "10";
      wait for 10 ns;
      assert Z_tb = '1' report "The output is not correct" severity failure;

      S_tb <= "11";
      wait for 10 ns;
      assert Z_tb = '0' report "The output is not correct" severity failure;

   end process;

END;
```

Figure 2: VHDL code snippet of 4 to 1 multiplexer test bench.

Include a screenshot of your design steps and of final the circuit.

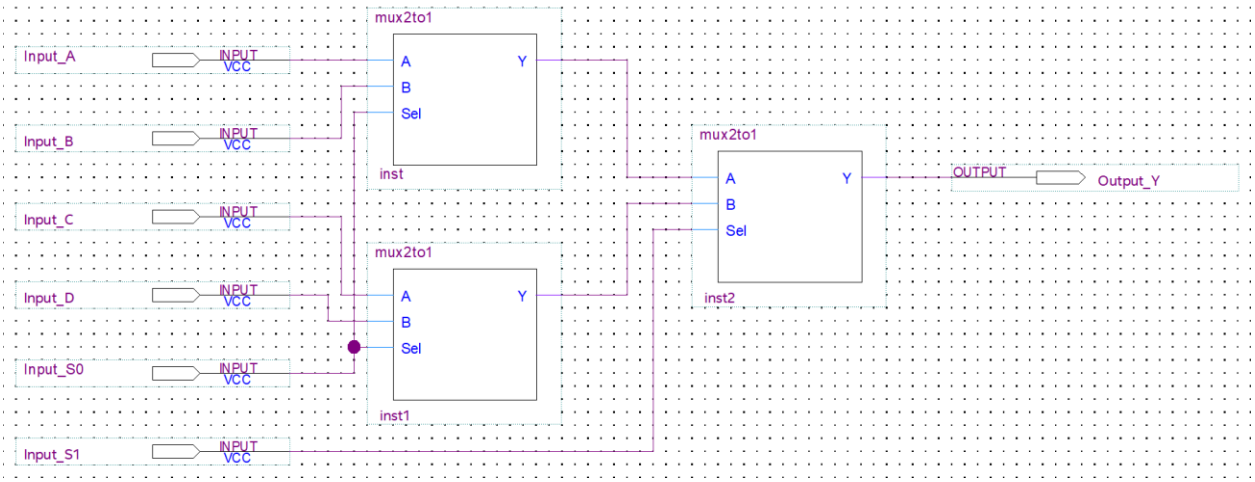

Figure 3: Block design file of 4 to 1 multiplexer using 2 to 1 multiplexers.

Simulation:

Include a screenshot of the vector waveform output file for all possible inputs.

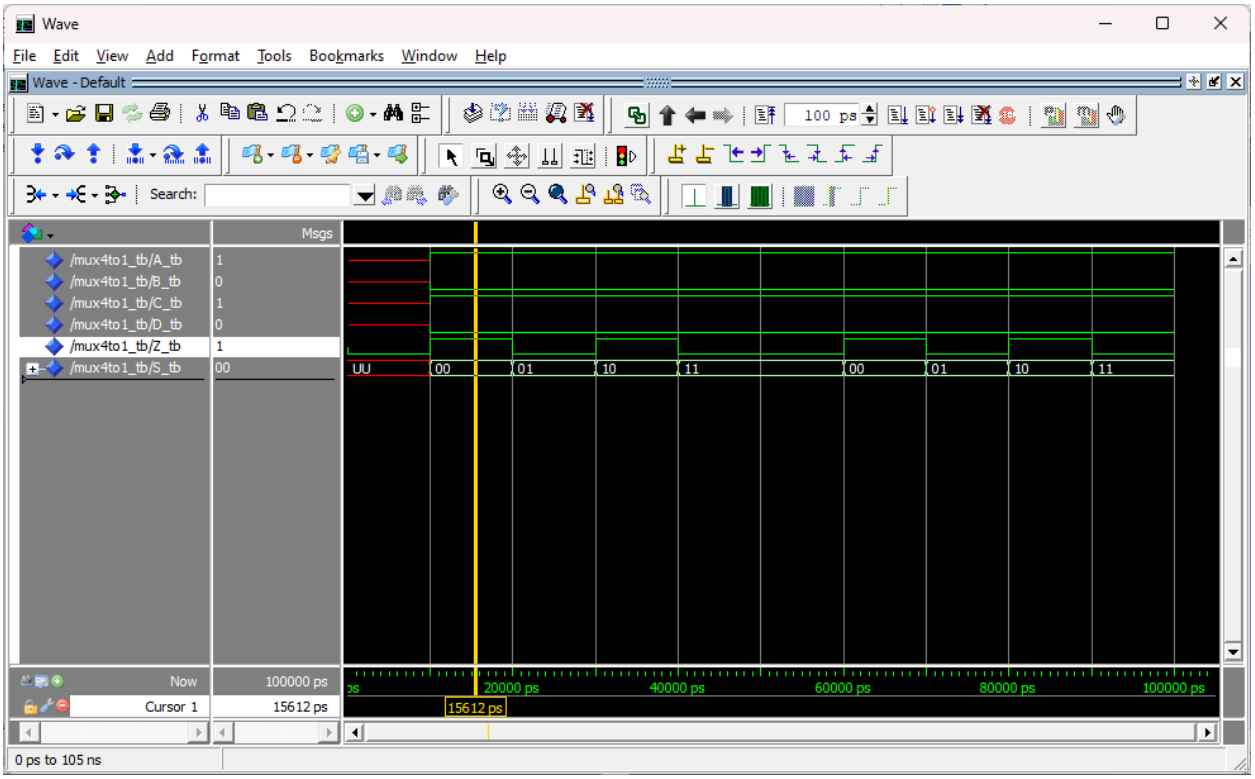

Figure 4: Waveform output of the 4 to 1 multiplexer testbench VHDL code.

Draw a truth table from the waveforms.

S0	S1	D0	D1	D2	D3	Y
0	0	0	x	x	x	0
0	0	1	x	x	x	1
0	1	x	0	x	x	0
0	1	x	1	x	x	1
1	0	x	x	0	x	0
1	0	x	x	1	x	1
1	1	x	x	x	0	0
1	1	x	x	x	1	1

2-to-1 Multiplexer with 8-bit input & output

Functionality and Specifications:

What is the combinational logic function (Boolean function) of the circuit?

A 2-to-1 multiplexer with 8-bit input and output has 2 data inputs (A, B), 1 output (Z), and 1 select line (S). The output Z is determined by the select line S as follows:

$$Z = (A \cdot \overline{S}) + (B \cdot S)$$

What is the VHDL code (Boolean function) of the circuit?

```
1   library ieee;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity mux2to1_8bit_vhdl is
5       Port ( A : in STD_LOGIC_VECTOR(7 downto 0);
6              B : in STD_LOGIC_VECTOR(7 downto 0);
7              Sel : in STD_LOGIC;
8              Y : out STD_LOGIC_VECTOR(7 downto 0)
9              );
10  end mux2to1_8bit_vhdl;
11
12  architecture Behavioral of mux2to1_8bit_vhdl is
13  component mux2to1 is
14     Port ( A : in STD_LOGIC;
15            B : in STD_LOGIC;
16            Sel : in STD_LOGIC;
17            Y : out STD_LOGIC);
18  end component;
19
20  begin
21     first_bit: mux2to1 port map(A(0), B(0), Sel, Y(0));
22     second_bit: mux2to1 port map(A(1), B(1), Sel, Y(1));
23     third_bit: mux2to1 port map(A(2), B(2), Sel, Y(2));
24     fourth_bit: mux2to1 port map(A(3), B(3), Sel, Y(3));
25     fifth_bit: mux2to1 port map(A(4), B(4), Sel, Y(4));
26     sixth_bit: mux2to1 port map(A(5), B(5), Sel, Y(5));
27     seventh_bit: mux2to1 port map(A(6), B(6), Sel, Y(6));
28     eigth_bit: mux2to1 port map(A(7), B(7), Sel, Y(7));
29  end Behavioral;
```

Figure 5: VHDL code snippet of 2 to 1 multiplexer with an 8-bit input and 8-bit output.

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity mux2to1_8bit_TB is
end entity mux2to1_8bit_TB;

architecture TBArchitecture of mux2to1_8bit_TB is
    signal A_tb, B_tb, Y_tb : STD_LOGIC_VECTOR(7 downto 0);
    signal Sel_tb : STD_LOGIC;

    component mux2to1_8bit_vhdl is
        Port ( A : in STD_LOGIC_VECTOR(7 downto 0);
               B : in STD_LOGIC_VECTOR(7 downto 0);
               Sel : in STD_LOGIC;
               Y : out STD_LOGIC_VECTOR(7 downto 0)
             );
    end component mux2to1_8bit_vhdl;

    begin
        DUT: mux2to1_8bit_vhdl
          port map (
              A => A_tb,
              B => B_tb,
              Sel => Sel_tb,
              Y => Y_tb
          );

    stimulus_process: process
        begin
          A_tb <= "00000000";
          B_tb <= "11111111";
          Sel_tb <= '0';
          wait for 10 ns;

          Sel_tb <= '1';
          wait for 10 ns;

          A_tb <= "00001111";
          B_tb <= "11110000";
          Sel_tb <= '0';
          wait for 10 ns;

          Sel_tb <= '1';
          wait for 10 ns;

          A_tb <= "11111111";
          B_tb <= "00000000";
          Sel_tb <= '0';
          wait for 10 ns;

          Sel_tb <= '1';
          wait for 10 ns;

        end process;

end architecture TBArchitecture;
```

Figure 6: VHDL code snippet of 2 to 1 multiplexer with an 8-bit input and 8-bit output test bench.

Include a screenshot of your design steps and of final the circuit.

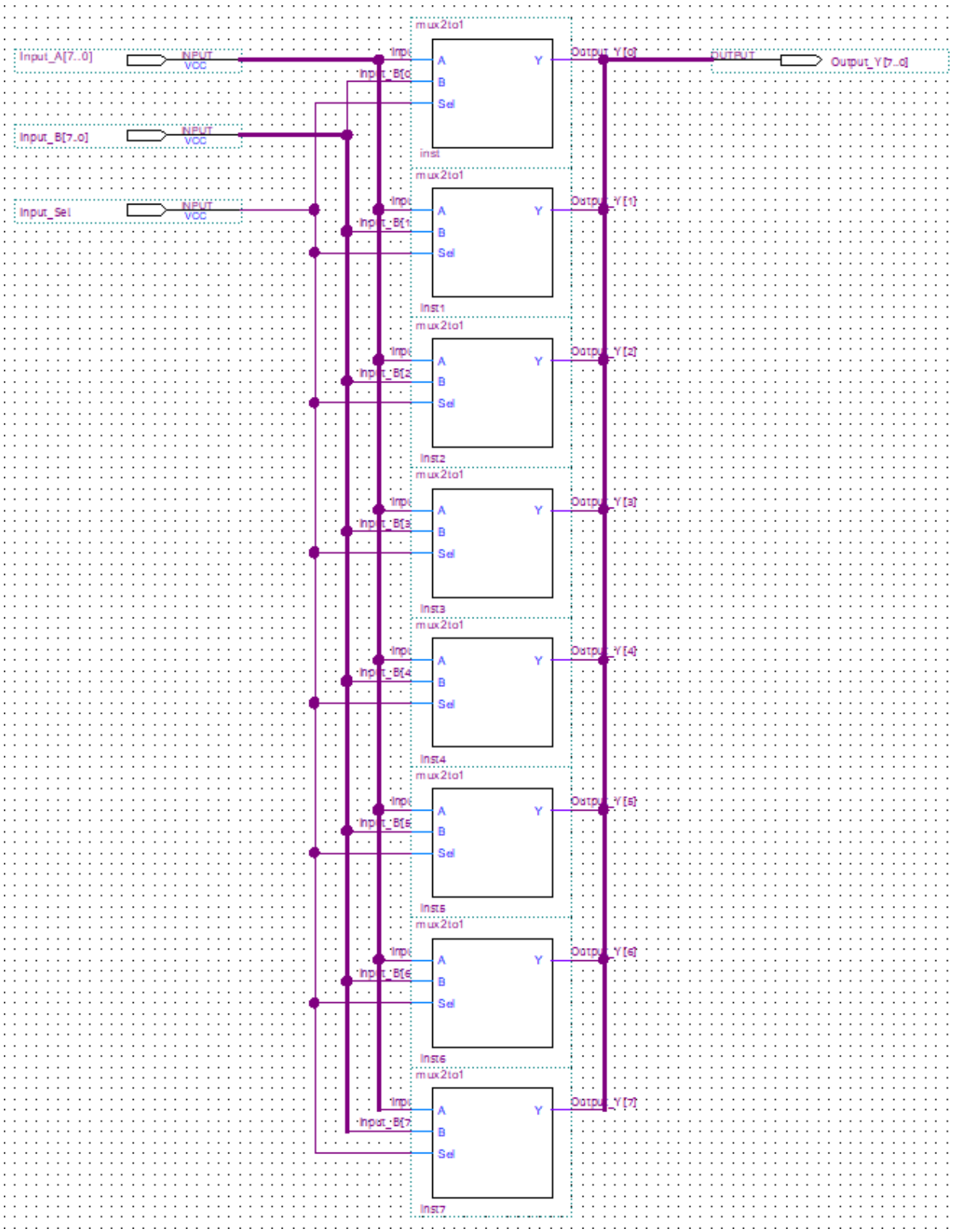
Input_A[7..0]
INPUT
VCC
Input_B[7..0]
INPUT
VCC
Input_Sel
INPUT
VCC
mux2to1
A
B
Sel
Y
inst
inst1
inst2
inst3
inst4
inst5
inst6
inst7
Output_Y[0]
Output_Y[1]
Output_Y[2]
Output_Y[3]
Output_Y[4]
Output_Y[5]
Output_Y[6]
Output_Y[7]
OUTPUT
Output_Y[7..0]

Figure 7: Block design file of 2 to 1 multiplexer with an 8-bit input & output using 2 to 1 multiplexers.

Simulation:

Include a screenshot of the vector waveform output file for all possible inputs.

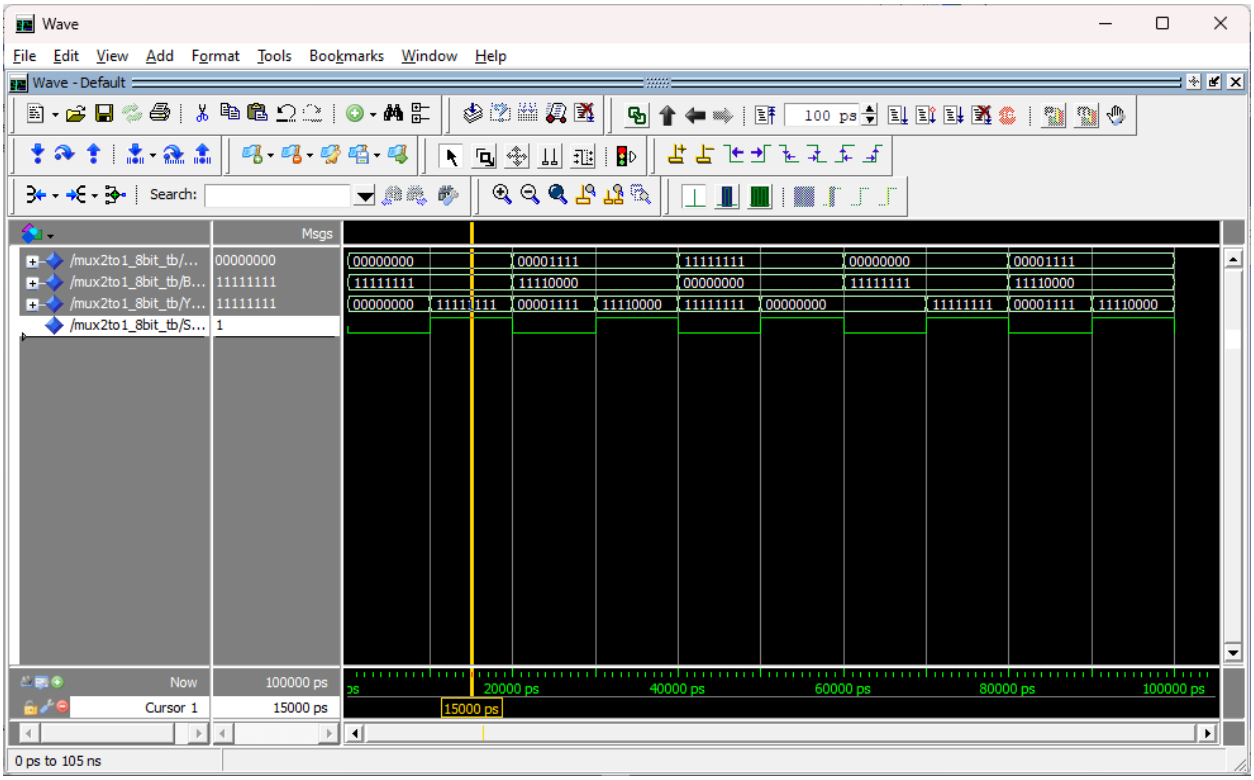

Figure 8: Waveform output of the 2 to 1 multiplexer with an 8-bit input & output testbench VHDL code.

Draw a truth table from the waveforms.

S	A	B	Y
0	0	0	0
0	0	1	0

0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

5-to-1 Multiplexer (using AND, OR, and NOT gates)

Functionality and Specifications:

What is the combinational logic function (Boolean function) of the circuit?

A 5-to-1 multiplexer with 8-bit input and output has 5 data inputs (A, B, C, D, E), 1 output (Z), and 3 select lines (S2, S1, S0). The output Z is determined by the select lines S2, S1, S0 as follows:

$$Z = (A \cdot \overline{S2} \cdot \overline{S1} \cdot \overline{S0}) + \left(B \cdot \overline{S2} \cdot \overline{S1} \cdot S0\right) + \left(C \cdot \overline{S2} \cdot S1 \cdot \overline{S0}\right) + \left(D \cdot \overline{S2} \cdot S1 \cdot S0\right) + \left(E \cdot S2 \cdot \overline{S1} \cdot S0\right)$$

What is the VHDL code (Boolean function) of the circuit?

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

entity mux5to1_vhdl is
    Port ( A, B, C, D, E : in STD_LOGIC;
           S : in STD_LOGIC_VECTOR(2 downto 0);
           Y : out STD_LOGIC);
end mux5to1_vhdl;

architecture Behavioral of mux5to1_vhdl is
begin
   process (S)
   begin
      case S is
         when "000" =>
            Y <= A;
         when "001" =>
            Y <= B;
         when "010" =>
            Y <= C;
         when "011" =>
            Y <= D;
         when "100" =>
            Y <= E;
         when others =>
            y <= E;
      end case;

   end process;

end Behavioral;
```

Figure 9: VHDL code snippet of 5 to 1 multiplexer.

```
1   library ieee;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity mux5to1_TB is
5   end entity mux5to1_TB;
6
7   architecture TestBenchArchitecture of mux5to1_TB is
8
9      component mux5to1_vhdl is
10        Port ( A, B, C, D, E : in STD_LOGIC;
11               S : in STD_LOGIC_VECTOR(2 downto 0);
12               Y : out STD_LOGIC);
13     end component mux5to1_vhdl;
14
15     signal A_tb, B_tb, C_tb, D_tb, E_tb, Y_tb : std_logic;
16     signal S_tb : std_logic_vector(2 downto 0);
17
18  begin
19     DUT: mux5to1_vhdl
20        port map (
21           A => A_tb,
22           B => B_tb,
23           C => C_tb,
24           D => D_tb,
25           E => E_tb,
26           S => S_tb,
27           Y => Y_tb
28        );
29     stimulus_process: process
30        begin
31           A_tb <= '1';
32           B_tb <= '0';
33           C_tb <= '1';
34           D_tb <= '0';
35           E_tb <= '1';
36           S_tb <= "000";
37           wait for 10ns;
38
39           S_tb <= "001";
40           wait for 10ns;
41
42           S_tb <= "010";
43           wait for 10ns;
44
45           S_tb <= "011";
46           wait for 10ns;
47
48           S_tb <= "100";
49           wait for 10ns;
50
51           S_tb <= "101";
52           wait for 10ns;
53
54           S_tb <= "110";
55           wait for 10ns;
56
57           S_tb <= "111";
58           wait for 10ns;
59
60        end process;
61
62  end architecture TestBenchArchitecture;
```

Figure 10: VHDL code snippet of 5 to 1 multiplexer test bench.

Include a screenshot of your design steps and of final the circuit.

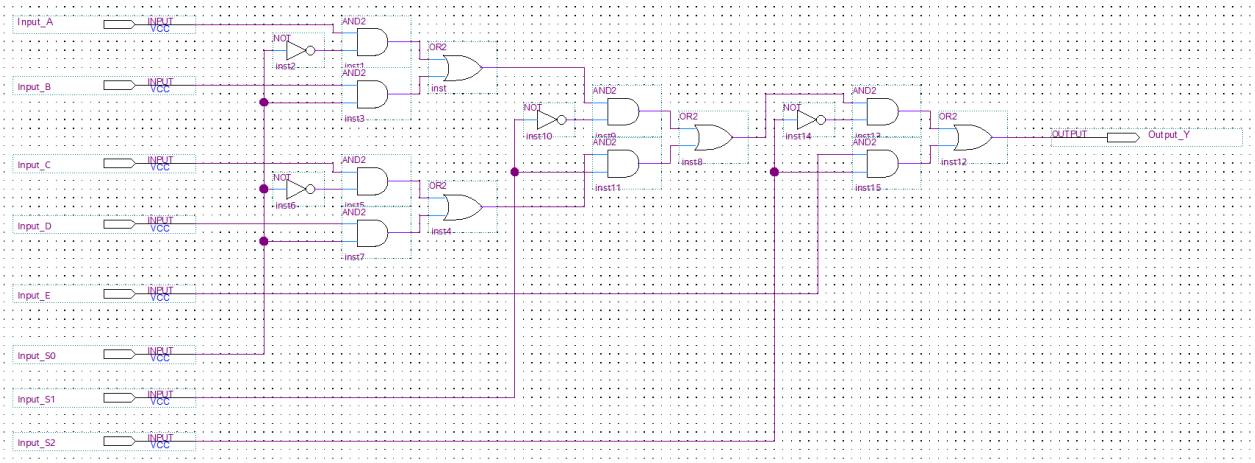

Figure 11: Block design file of 5 to 1 multiplexer using only AND, OR, and NOT gates.

Simulation:

Include a screenshot of the vector waveform output file for all possible inputs.

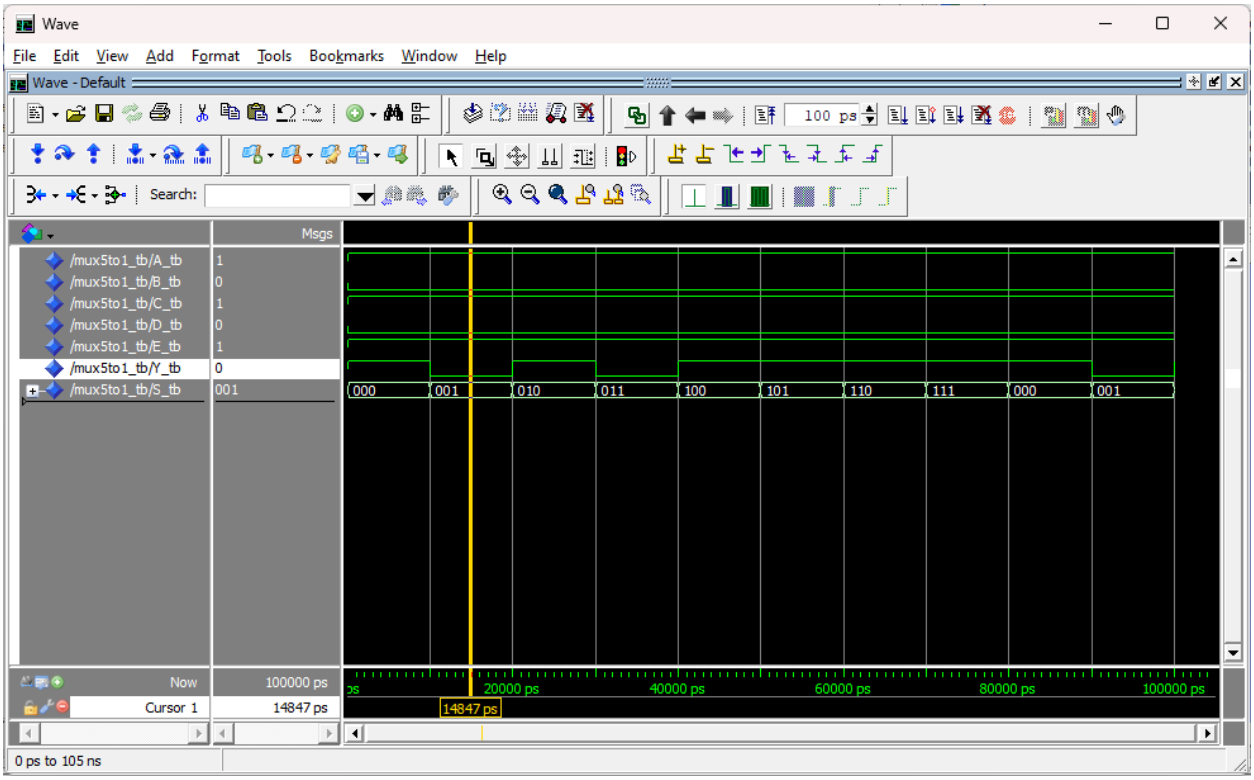

Figure 12: Waveform output of the 5 to 1 multiplexer testbench VHDL code.

Draw a truth table from the waveforms.

S2	S1	S0	Y
0	0	0	A
0	0	1	B
0	1	0	C
0	1	1	D
1	0	0	E
1	0	1	E
1	1	0	E
1	1	1	E

3-to-8 Decoder

Functionality and Specifications:

What is the combinational logic function (Boolean function) of the circuit?

A 3-to-8 decoder has 3 input lines (A2, A1, A0) and 8 output lines (Y7, Y6, Y5, Y4, Y3, Y2, Y1, Y0). The output lines correspond to the binary representation of the input combination. The combinational logic function for a 3-to-8 decoder is as follows:

$$Y7 = (\overline{A2} \cdot \overline{A1} \cdot \overline{A0})$$

$$Y6 = \left(\overline{A2} \cdot \overline{A1} \cdot A0\right)$$

$$Y5 = \left(\overline{A2} \cdot A1 \cdot \overline{A0}\right)$$

$$Y4 = \left(\overline{A2} \cdot A1 \cdot A0\right)$$

$$Y3 = \left(A2 \cdot \overline{A1} \cdot \overline{A0}\right)$$

$$Y2 = \left(A2 \cdot \overline{A1} \cdot A0\right)$$

$$Y1 = \left(A2 \cdot A1 \cdot \overline{A0}\right)$$

$$Y0 = (A2 \cdot A1 \cdot A0)$$

What is the VHDL code (Boolean function) of the circuit?

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

entity decoder3to8_vhdl is
    Port ( S0, S1, S2 : in STD_LOGIC;
           Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7: out STD_LOGIC
         );
end decoder3to8_vhdl;

architecture Behavioral of decoder3to8_vhdl is
begin
    process (S0, S1, S2)
    variable input : std_logic_vector(2 downto 0);
    begin
        input:=S2&S1&S0;
        case input is
            when "000" =>
                Y0 <= '1';
                Y1 <= '0';
                Y2 <= '0';
                Y3 <= '0';
                Y4 <= '0';
                Y5 <= '0';
                Y6 <= '0';
                Y7 <= '0';
            when "001" =>
                Y0 <= '0';
                Y1 <= '1';
                Y2 <= '0';
                Y3 <= '0';
                Y4 <= '0';
                Y5 <= '0';
                Y6 <= '0';
                Y7 <= '0';
            when "010" =>
                Y0 <= '0';
                Y1 <= '0';
                Y2 <= '1';
                Y3 <= '0';
                Y4 <= '0';
                Y5 <= '0';
                Y6 <= '0';
                Y7 <= '0';
            when "011" =>
                Y0 <= '0';
                Y1 <= '0';
                Y2 <= '0';
                Y3 <= '1';
                Y4 <= '0';
                Y5 <= '0';
                Y6 <= '0';
                Y7 <= '0';
            when "100" =>
                Y0 <= '0';
                Y1 <= '0';
                Y2 <= '0';
                Y3 <= '0';
                Y4 <= '1';
                Y5 <= '0';
                Y6 <= '0';
                Y7 <= '0';
            when "101" =>
                Y0 <= '0';
                Y1 <= '0';
                Y2 <= '0';
                Y3 <= '0';
                Y4 <= '0';
                Y5 <= '1';
                Y6 <= '0';
                Y7 <= '0';
            when "110" =>
                Y0 <= '0';
                Y1 <= '0';
                Y2 <= '0';
                Y3 <= '0';
                Y4 <= '0';
                Y5 <= '0';
                Y6 <= '1';
                Y7 <= '0';
            when "111" =>
                Y0 <= '0';
                Y1 <= '0';
                Y2 <= '0';
                Y3 <= '0';
                Y4 <= '0';
                Y5 <= '0';
                Y6 <= '0';
                Y7 <= '1';
            when others =>
                Y0 <= '0';
                Y1 <= '0';
                Y2 <= '0';
                Y3 <= '0';
                Y4 <= '0';
                Y5 <= '0';
                Y6 <= '0';
                Y7 <= '0';
        end case;
    end process;
end Behavioral;
```

Figure 13: VHDL code snippet of 3 to 8 decoder.

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

entity decoder3to8_TB is
end entity decoder3to8_TB;

architecture TestBenchArchitecture of decoder3to8_TB is

    component decoder3to8_vhdl is
        Port (
            S0, S1, S2 : in STD_LOGIC;
            Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7: out STD_LOGIC
        );
    end component decoder3to8_vhdl;

    signal S0_tb, S1_tb, S2_tb : std_logic;
    signal Y0_tb, Y1_tb, Y2_tb, Y3_tb, Y4_tb, Y5_tb, Y6_tb, Y7_tb : std_logic;

begin
    DUT: decoder3to8_vhdl
        port map (
            S0 => S0_tb,
            S1 => S1_tb,
            S2 => S2_tb,
            Y0 => Y0_tb,
            Y1 => Y1_tb,
            Y2 => Y2_tb,
            Y3 => Y3_tb,
            Y4 => Y4_tb,
            Y5 => Y5_tb,
            Y6 => Y6_tb,
            Y7 => Y7_tb
        );

    stimulus_process: process
    begin
        S0_tb <= '0';
        S1_tb <= '0';
        S2_tb <= '0';
        wait for 10 ns;

        S0_tb <= '1';
        S1_tb <= '0';
        S2_tb <= '0';
        wait for 10 ns;

        S0_tb <= '0';
        S1_tb <= '1';
        S2_tb <= '0';
        wait for 10 ns;

        S0_tb <= '1';
        S1_tb <= '1';
        S2_tb <= '0';
        wait for 10 ns;

        S0_tb <= '0';
        S1_tb <= '0';
        S2_tb <= '1';
        wait for 10 ns;

        S0_tb <= '1';
        S1_tb <= '0';
        S2_tb <= '1';
        wait for 10 ns;

        S0_tb <= '0';
        S1_tb <= '1';
        S2_tb <= '1';
        wait for 10 ns;

        S0_tb <= '1';
        S1_tb <= '1';
        S2_tb <= '1';
        wait for 10 ns;

        wait;
    end process stimulus_process;

end architecture TestBenchArchitecture;
```

Figure 14: VHDL code snippet of 3 to 8 decoder test bench.

Include a screenshot of your design steps and of final the circuit.

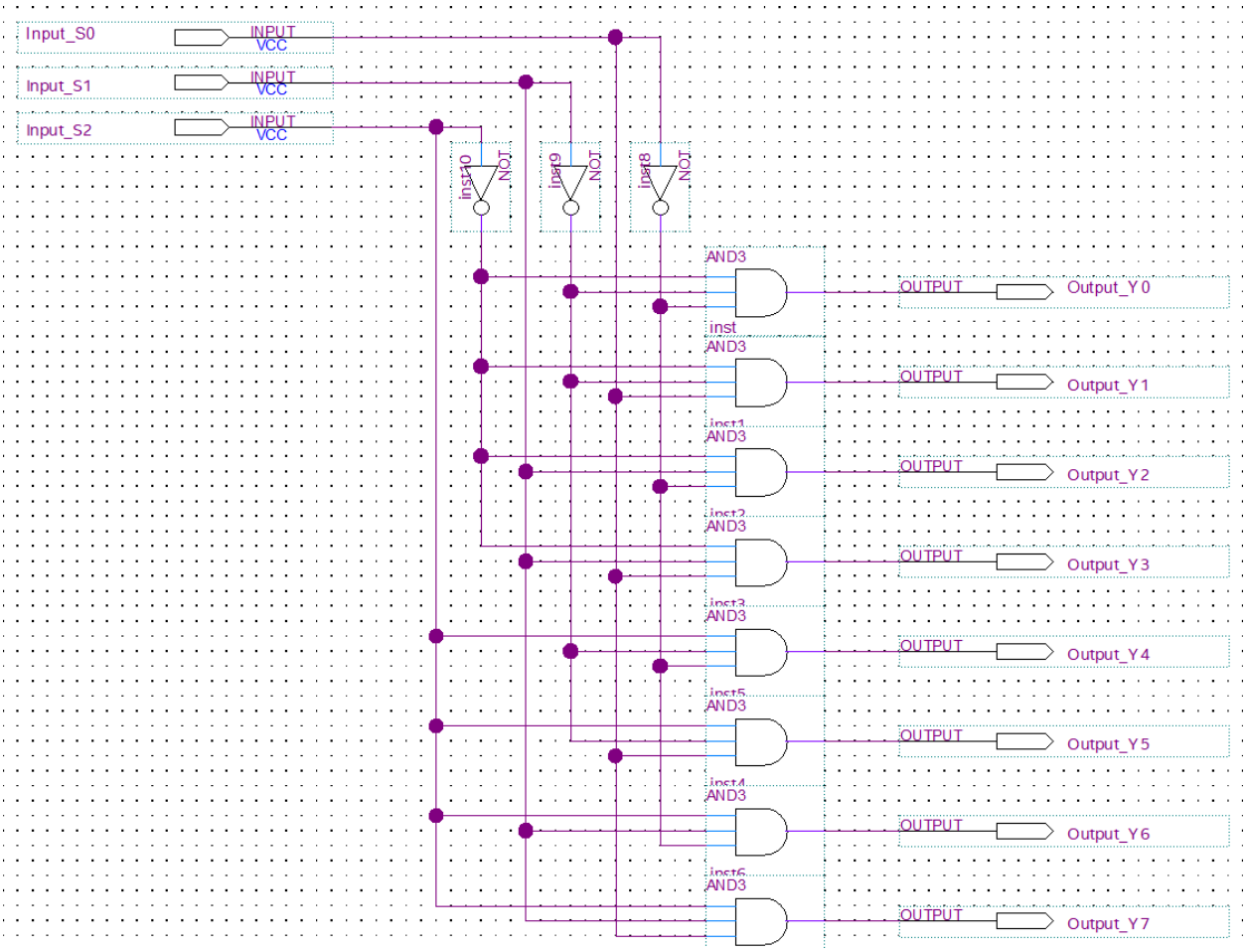

Figure 15: Block design file of 3 to 8 decoder.

Simulation:

Include a screenshot of the vector waveform output file for all possible inputs.

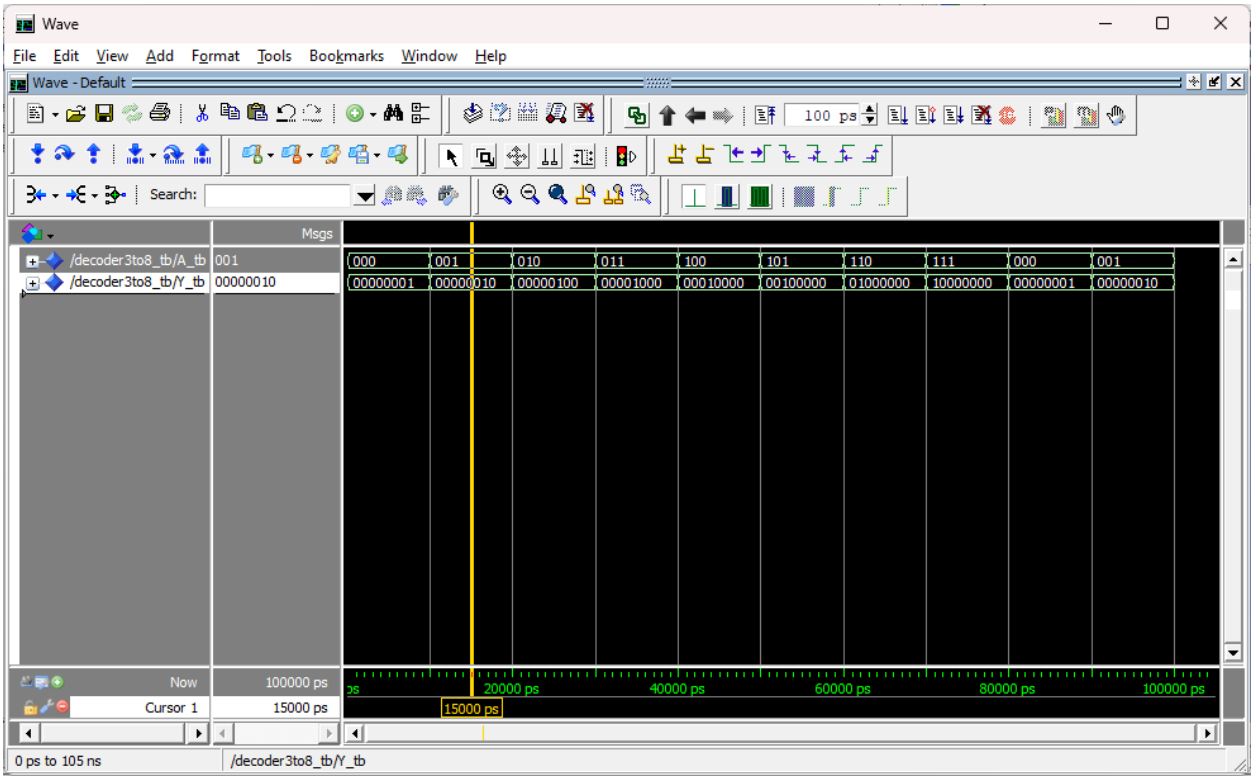

Figure 16: Waveform output of the 3 to 8 decoder testbench VHDL code.

Draw a truth table from the waveforms.

A2	A1	A0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

8-to-3 Priority Encoder

Functionality and Specifications:

What is the combinational logic function (Boolean function) of the circuit?

An 8-to-3 priority encoder with 8 inputs (I7, I6, I5, I4, I3, I2, I1, I0) and 3 outputs (Y2, Y1, Y0) works by encoding the highest priority active input into a 3-bit binary output. The output bits are determined as follows:

$$Y2 = (A7 + A6 + A5 + A4)$$

$$Y1 = (A7 + A6 + A3 + A2)$$

$$Y0 = (A7 + A5 + A3 + A1)$$

What is the VHDL code (Boolean function) of the circuit?

```
1   library ieee;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity encoder8to3_vhdl is
5       Port (
6           Y0, Y1, Y2 : out STD_LOGIC;
7           S0, S1, S2, S3, S4, S5, S6, S7: in STD_LOGIC
8       );
9   end encoder8to3_vhdl;
10
11  architecture Dataflow of encoder8to3_vhdl is
12  begin
13      Y0 <= (S1 or S3 or S5 or S7);
14      Y1 <= (S2 or S3 or S6 or S7);
15      Y2 <= (S4 or S5 or S6 or S7);
16  end Dataflow;
17
```

Figure 17: VHDL code snippet of 8 to 3 encoder.

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

entity encoder8to3_TB is
end entity encoder8to3_TB;

architecture TestBenchArchitecture of encoder8to3_TB is

    component encoder8to3_vhdl is
        Port ( Y0, Y1, Y2 : out STD_LOGIC;
               S0, S1, S2, S3, S4, S5, S6, S7: in STD_LOGIC
        );
    end component encoder8to3_vhdl;

    signal S0_tb, S1_tb, S2_tb, S3_tb, S4_tb, S5_tb, S6_tb, S7_tb : std_logic;
    signal Y0_tb, Y1_tb, Y2_tb : std_logic;

begin
    DUT: encoder8to3_vhdl
        port map (
            Y0 => Y0_tb,
            Y1 => Y1_tb,
            Y2 => Y2_tb,
            S0 => S0_tb,
            S1 => S1_tb,
            S2 => S2_tb,
            S3 => S3_tb,
            S4 => S4_tb,
            S5 => S5_tb,
            S6 => S6_tb,
            S7 => S7_tb
        );

    stimulus_process: process
    begin
        S0_tb <= '1'; S1_tb <= '0'; S2_tb <= '0'; S3_tb <= '0'; S4_tb <= '0'; S5_tb <= '0'; S6_tb <= '0'; S7_tb <= '0';
        wait for 10 ns;

        S0_tb <= '0'; S1_tb <= '1'; S2_tb <= '0'; S3_tb <= '0'; S4_tb <= '0'; S5_tb <= '0'; S6_tb <= '0'; S7_tb <= '0';
        wait for 10 ns;

        S0_tb <= '0'; S1_tb <= '0'; S2_tb <= '1'; S3_tb <= '0'; S4_tb <= '0'; S5_tb <= '0'; S6_tb <= '0'; S7_tb <= '0';
        wait for 10 ns;

        S0_tb <= '0'; S1_tb <= '0'; S2_tb <= '0'; S3_tb <= '1'; S4_tb <= '0'; S5_tb <= '0'; S6_tb <= '0'; S7_tb <= '0';
        wait for 10 ns;

        S0_tb <= '0'; S1_tb <= '0'; S2_tb <= '0'; S3_tb <= '0'; S4_tb <= '1'; S5_tb <= '0'; S6_tb <= '0'; S7_tb <= '0';
        wait for 10 ns;

        S0_tb <= '0'; S1_tb <= '0'; S2_tb <= '0'; S3_tb <= '0'; S4_tb <= '0'; S5_tb <= '1'; S6_tb <= '0'; S7_tb <= '0';
        wait for 10 ns;

        S0_tb <= '0'; S1_tb <= '0'; S2_tb <= '0'; S3_tb <= '0'; S4_tb <= '0'; S5_tb <= '0'; S6_tb <= '1'; S7_tb <= '0';
        wait for 10 ns;

        S0_tb <= '0'; S1_tb <= '0'; S2_tb <= '0'; S3_tb <= '0'; S4_tb <= '0'; S5_tb <= '0'; S6_tb <= '0'; S7_tb <= '1';
        wait for 10 ns;

    end process stimulus_process;

end architecture TestBenchArchitecture;

```

Figure 18: VHDL code snippet of 8 to 3 encoder test bench.

Include a screenshot of your design steps and of final the circuit.

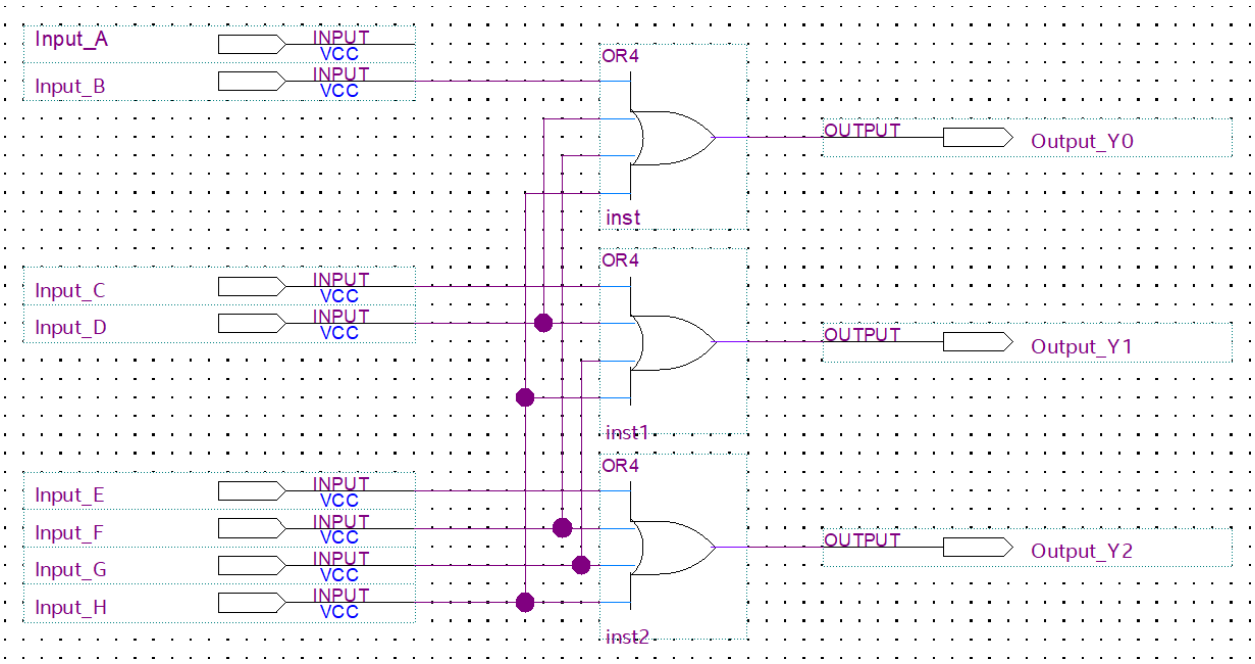

Figure 19: Block design file of 8 to 3 priority encoder.

Simulation:

Include a screenshot of the vector waveform output file for all possible inputs.

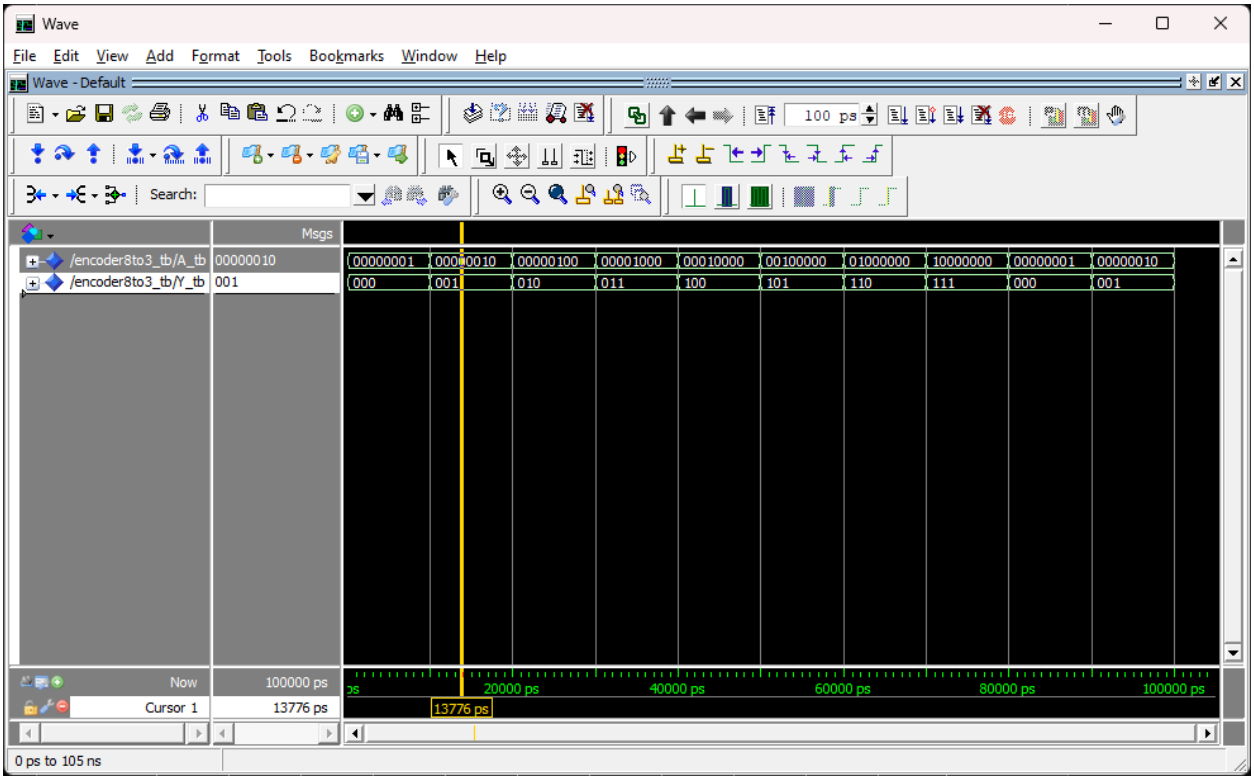

Figure 20: Waveform output of the 8 to 3 encoder testbench VHDL code.

Draw a truth table from the waveforms.

A7	A6	A5	A4	A3	A2	A1	A0	Y2	Y1	Y0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

1-to-2 Demultiplexer

Functionality and Specifications:

What is the combinational logic function (Boolean function) of the circuit?

A 1-to-2 demultiplexer with 1 input (A), 2 outputs (Y0, Y1), and 1 select line (S) works by selecting one of the two outputs based on the select line. The output bits are determined as follows:

$$Y0 = \left(A \cdot \overline{S}\right)$$

$$Y1 = (A \cdot S)$$

What is the VHDL code (Boolean function) of the circuit?

```
1   library ieee;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity demux1to2_vhdl is
5       Port ( S, A : in STD_LOGIC;
6              Y0, Y1 : out STD_LOGIC
7            );
8   end demux1to2_vhdl;
9
10  architecture Dataflow of demux1to2_vhdl is
11  begin
12      Y0 <= A and (not S);
13      Y1 <= A and S;
14  end Dataflow;
```

Figure 21: VHDL code snippet of 1 to 2 demux.

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

entity demux1to2_TB is
end entity demux1to2_TB;

architecture TestBenchArchitecture of demux1to2_TB is

    component demux1to2_vhdl is
         Port ( S, A : in STD_LOGIC;
                Y0, Y1 : out STD_LOGIC
              );
    end component demux1to2_vhdl;

    signal A_tb, S_tb, Y0_tb, Y1_tb : std_logic;

begin
    DUT: demux1to2_vhdl
        port map (
            A => A_tb,
            S => S_tb,
            Y0 => Y0_tb,
            Y1 => Y1_tb
        );

    stimulus_process: process
    begin
      A_tb <= '0';
      S_tb <= '0';
      wait for 10ns;
      S_tb <= '1';
      wait for 10ns;

      A_tb <= '1';
      S_tb <= '0';
      wait for 10ns;
      S_tb <= '1';
      wait for 10ns;

    end process stimulus_process;

end architecture TestBenchArchitecture;
```

Figure 22: VHDL code snippet of 1 to 2 demux test bench.

Include a screenshot of your design steps and of final the circuit.

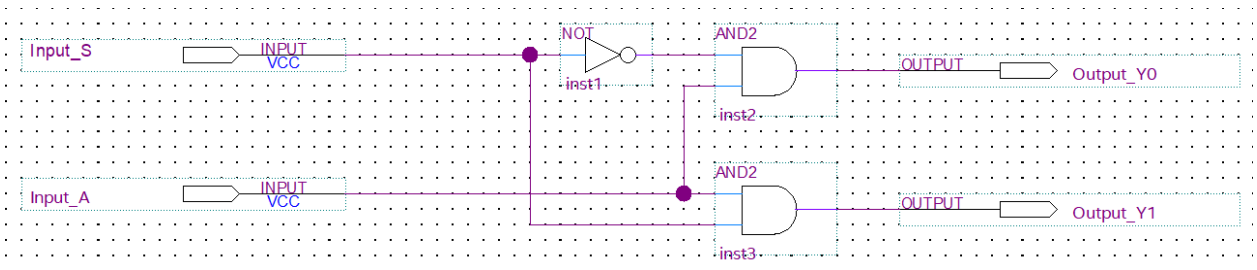

Figure 23: Block design file of a 1 to 2 demultiplexer.

Simulation:

Include a screenshot of the vector waveform output file for all possible inputs.

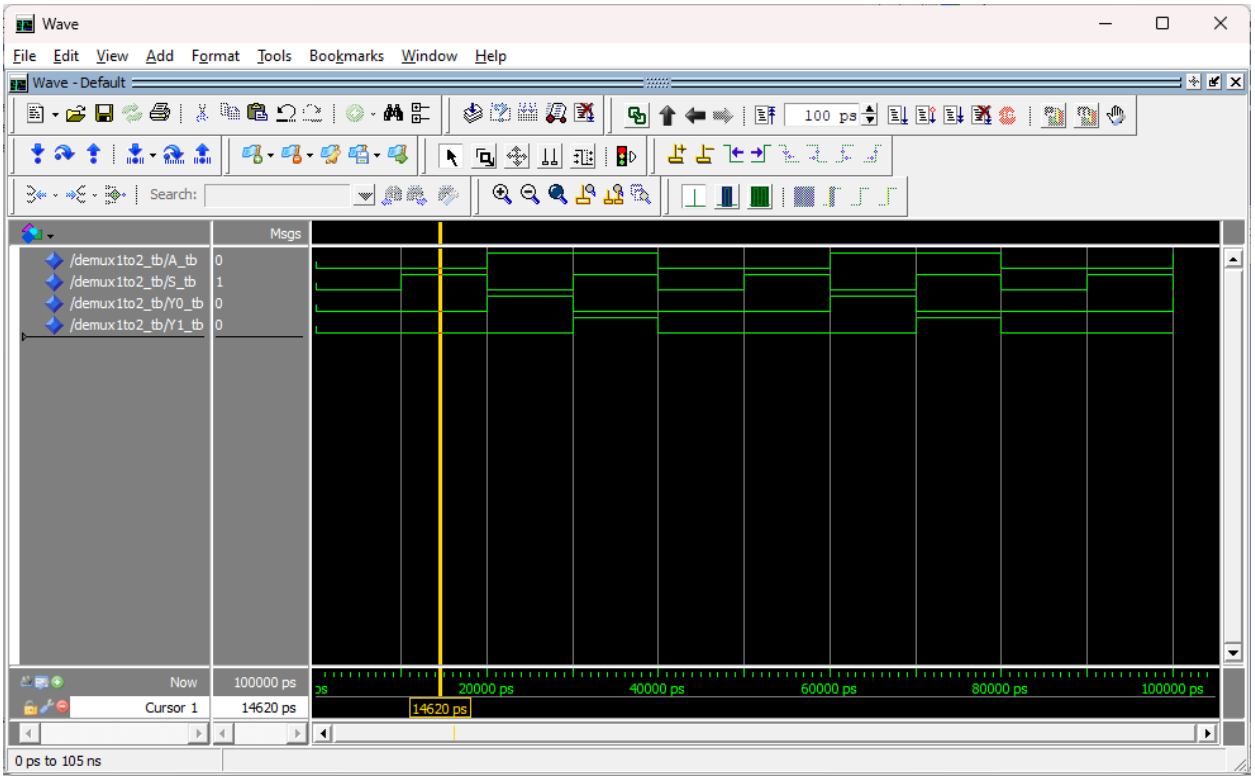

Figure 24: Waveform output of the 1 to 2 demultiplexer testbench VHDL code.

Draw a truth table from the waveforms.

S	A	Y1	Y0
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

Generated VHDL code from Quartus

```
1  -- Copyright (C) 2022  Intel Corporation. All rights reserved.
2  -- Your use of Intel Corporation's design tools, logic functions
3  -- and other software and tools, and any partner logic
4  -- functions, and any output files from any of the foregoing
5  -- (including device programming or simulation files), and any
6  -- associated documentation or information are expressly subject
7  -- to the terms and conditions of the Intel Program License
8  -- Subscription Agreement, the Intel Quartus Prime License Agreement,
9  -- the Intel FPGA IP License Agreement, or other applicable license
10 -- agreement, including, without limitation, that your use is for
11 -- the sole purpose of programming logic devices manufactured by
12 -- Intel and sold by Intel or its authorized distributors.  Please
13 -- refer to the applicable agreement for further details, at
14 -- https://fpgasoftware.intel.com/eula.
15
16 -- PROGRAM     "Quartus Prime"
17 -- VERSION     "Version 22.1std.0 Build 915 10/25/2022 SC Lite Edition"
18 -- CREATED     "Thu Mar  7 11:25:21 2024"
19
20 LIBRARY ieee;
21 USE ieee.std_logic_1164.all;
22
23 LIBRARY work;
24
25 ENTITY mux4to1 IS
26    PORT
27    (
28       Input_A :  IN  STD_LOGIC;
29       Input_B :  IN  STD_LOGIC;
30       Input_C :  IN  STD_LOGIC;
31       Input_D :  IN  STD_LOGIC;
32       Input_S0 :  IN  STD_LOGIC;
33       Input_S1 :  IN  STD_LOGIC;
34       Output_Y :  OUT  STD_LOGIC
35    );
36 END mux4to1;
37
38 ARCHITECTURE bdf_type OF mux4to1 IS
39
40 COMPONENT mux2to1
41    PORT(A : IN STD_LOGIC;
42         B : IN STD_LOGIC;
43         Sel : IN STD_LOGIC;
44         Y : OUT STD_LOGIC
45    );
46 END COMPONENT;
47
48 SIGNAL   SYNTHESIZED_WIRE_0 :  STD_LOGIC;
49 SIGNAL   SYNTHESIZED_WIRE_1 :  STD_LOGIC;
50
51
52 BEGIN
53
54
55
56 b2v_inst : mux2to1
57 PORT MAP(A => Input_A,
58        B => Input_B,
59        Sel => Input_S0,
60        Y => SYNTHESIZED_WIRE_0);
61
62
63 b2v_inst1 : mux2to1
64 PORT MAP(A => Input_C,
65        B => Input_D,
66        Sel => Input_S0,
67        Y => SYNTHESIZED_WIRE_1);
68
69
70 b2v_inst2 : mux2to1
71 PORT MAP(A => SYNTHESIZED_WIRE_0,
72        B => SYNTHESIZED_WIRE_1,
73        Sel => Input_S1,
74        Y => Output_Y);
75
76
77 END bdf_type;
```

Figure 25: Generated VHDL code of the 4 to 1 multiplexer bdf.

```
1   -- Copyright (C) 2022  Intel Corporation. All rights reserved.
2   -- Your use of Intel Corporation's design tools, logic functions
3   -- and other software and tools, and any partner logic
4   -- functions, and any output files from any of the foregoing
5   -- (including device programming or simulation files), and any
6   -- associated documentation or information are expressly subject
7   -- to the terms and conditions of the Intel Program License
8   -- Subscription Agreement, the Intel Quartus Prime License Agreement,
9   -- the Intel FPGA IP License Agreement, or other applicable license
10  -- agreement, including, without limitation, that your use is for
11  -- the sole purpose of programming logic devices manufactured by
12  -- Intel and sold by Intel or its authorized distributors.  Please
13  -- refer to the applicable agreement for further details, at
14  -- https://fpgasoftware.intel.com/eula.
15
16  -- PROGRAM     "Quartus Prime"
17  -- VERSION     "Version 22.1std.0 Build 915 10/25/2022 SC Lite Edition"
18  -- CREATED     "Thu Mar  7 11:25:51 2024"
19
20  LIBRARY ieee;
21  USE ieee.std_logic_1164.all;
22
23  LIBRARY work;
24
25  ENTITY mux2to1_8bit IS
26     PORT
27     (
28        Input_Sel :  IN  STD_LOGIC;
29        Input_A :  IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
30        Input_B :  IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
31        Output_Y :  OUT  STD_LOGIC_VECTOR(7 DOWNTO 0)
32     );
33  END mux2to1_8bit;
34
35  ARCHITECTURE bdf_type OF mux2to1_8bit IS
36
37  COMPONENT mux2to1
38     PORT(A : IN STD_LOGIC;
39          B : IN STD_LOGIC;
40          Sel : IN STD_LOGIC;
41          Y : OUT STD_LOGIC
42     );
43  END COMPONENT;
44
45  SIGNAL   Output_Y_ALTERA_SYNTHESIZED :  STD_LOGIC_VECTOR(7 DOWNTO 0);
46
47
48  BEGIN
49
50
51
52  b2v_inst : mux2to1
53  PORT MAP(A => Input_A(0),
54         B => Input_B(0),
55         Sel => Input_Sel,
56         Y => Output_Y_ALTERA_SYNTHESIZED(0));
57
58
59  b2v_inst1 : mux2to1
60  PORT MAP(A => Input_A(1),
61         B => Input_B(1),
62         Sel => Input_Sel,
63         Y => Output_Y_ALTERA_SYNTHESIZED(1));
64
65
66  b2v_inst2 : mux2to1
67  PORT MAP(A => Input_A(2),
68         B => Input_B(2),
69         Sel => Input_Sel,
70         Y => Output_Y_ALTERA_SYNTHESIZED(2));
71
72
73  b2v_inst3 : mux2to1
74  PORT MAP(A => Input_A(3),
75         B => Input_B(3),
76         Sel => Input_Sel,
77         Y => Output_Y_ALTERA_SYNTHESIZED(3));
78
79
80  b2v_inst4 : mux2to1
81  PORT MAP(A => Input_A(4),
82         B => Input_B(4),
83         Sel => Input_Sel,
84         Y => Output_Y_ALTERA_SYNTHESIZED(4));
85
86
87  b2v_inst5 : mux2to1
88  PORT MAP(A => Input_A(5),
89         B => Input_B(5),
90         Sel => Input_Sel,
91         Y => Output_Y_ALTERA_SYNTHESIZED(5));
92
93
94  b2v_inst6 : mux2to1
95  PORT MAP(A => Input_A(6),
96         B => Input_B(6),
97         Sel => Input_Sel,
98         Y => Output_Y_ALTERA_SYNTHESIZED(6));
99
100
101 b2v_inst7 : mux2to1
102 PORT MAP(A => Input_A(7),
103        B => Input_B(7),
104        Sel => Input_Sel,
105        Y => Output_Y_ALTERA_SYNTHESIZED(7));
106
107 Output_Y <= Output_Y_ALTERA_SYNTHESIZED;
108
109 END bdf_type;
```

Figure 26: Generated VHDL code of the 2 to 1 multiplexer with 8-bit input & output bdf.

```
1   -- Copyright (C) 2022  Intel Corporation. All rights reserved.
2   -- Your use of Intel Corporation's design tools, logic functions
3   -- and other software and tools, and any partner logic
4   -- functions, and any output files from any of the foregoing
5   -- (including device programming or simulation files), and any
6   -- associated documentation or information are expressly subject
7   -- to the terms and conditions of the Intel Program License
8   -- Subscription Agreement, the Intel Quartus Prime License Agreement,
9   -- the Intel FPGA IP License Agreement, or other applicable license
10  -- agreement, including, without limitation, that your use is for
11  -- the sole purpose of programming logic devices manufactured by
12  -- Intel and sold by Intel or its authorized distributors.  Please
13  -- refer to the applicable agreement for further details, at
14  -- https://fpgasoftware.intel.com/eula.
15
16  -- PROGRAM     "Quartus Prime"
17  -- VERSION     "Version 22.1std.0 Build 915 10/25/2022 SC Lite Edition"
18  -- CREATED     "Thu Mar  7 11:26:06 2024"
19
20  LIBRARY ieee;
21  USE ieee.std_logic_1164.all;
22
23  LIBRARY work;
24
25  ENTITY mux5to1 IS
26     PORT
27     (
28        Input_S0 :  IN  STD_LOGIC;
29        Input_S1 :  IN  STD_LOGIC;
30        Input_S2 :  IN  STD_LOGIC;
31        Input_A :  IN  STD_LOGIC;
32        Input_B :  IN  STD_LOGIC;
33        Input_C :  IN  STD_LOGIC;
34        Input_D :  IN  STD_LOGIC;
35        Input_E :  IN  STD_LOGIC;
36        Output_Y :  OUT  STD_LOGIC
37     );
38  END mux5to1;
39
40  ARCHITECTURE bdf_type OF mux5to1 IS
41
42  SIGNAL   SYNTHESIZED_WIRE_0 :  STD_LOGIC;
43  SIGNAL   SYNTHESIZED_WIRE_1 :  STD_LOGIC;
44  SIGNAL   SYNTHESIZED_WIRE_2 :  STD_LOGIC;
45  SIGNAL   SYNTHESIZED_WIRE_3 :  STD_LOGIC;
46  SIGNAL   SYNTHESIZED_WIRE_4 :  STD_LOGIC;
47  SIGNAL   SYNTHESIZED_WIRE_5 :  STD_LOGIC;
48  SIGNAL   SYNTHESIZED_WIRE_6 :  STD_LOGIC;
49  SIGNAL   SYNTHESIZED_WIRE_7 :  STD_LOGIC;
50  SIGNAL   SYNTHESIZED_WIRE_8 :  STD_LOGIC;
51  SIGNAL   SYNTHESIZED_WIRE_9 :  STD_LOGIC;
52  SIGNAL   SYNTHESIZED_WIRE_10 :  STD_LOGIC;
53  SIGNAL   SYNTHESIZED_WIRE_11 :  STD_LOGIC;
54  SIGNAL   SYNTHESIZED_WIRE_12 :  STD_LOGIC;
55  SIGNAL   SYNTHESIZED_WIRE_13 :  STD_LOGIC;
56  SIGNAL   SYNTHESIZED_WIRE_14 :  STD_LOGIC;
57
58
59  BEGIN
60
61
62
63  SYNTHESIZED_WIRE_13 <= SYNTHESIZED_WIRE_0 OR SYNTHESIZED_WIRE_1;
64
65
66  SYNTHESIZED_WIRE_1 <= Input_A AND SYNTHESIZED_WIRE_2;
67
68
69  SYNTHESIZED_WIRE_14 <= NOT(Input_S1);
70
71
72
73  SYNTHESIZED_WIRE_11 <= SYNTHESIZED_WIRE_3 AND Input_S1;
74
75
76  Output_Y <= SYNTHESIZED_WIRE_4 OR SYNTHESIZED_WIRE_5;
77
78
79  SYNTHESIZED_WIRE_5 <= SYNTHESIZED_WIRE_6 AND SYNTHESIZED_WIRE_7;
80
81
82  SYNTHESIZED_WIRE_7 <= NOT(Input_S2);
83
84
85
86  SYNTHESIZED_WIRE_4 <= Input_E AND Input_S2;
87
88
89  SYNTHESIZED_WIRE_2 <= NOT(Input_S0);
90
91
92
93  SYNTHESIZED_WIRE_0 <= Input_B AND Input_S0;
94
95
96  SYNTHESIZED_WIRE_3 <= SYNTHESIZED_WIRE_8 OR SYNTHESIZED_WIRE_9;
97
98
99  SYNTHESIZED_WIRE_9 <= Input_C AND SYNTHESIZED_WIRE_10;
100
101
102 SYNTHESIZED_WIRE_10 <= NOT(Input_S0);
103
104
105
106 SYNTHESIZED_WIRE_8 <= Input_D AND Input_S0;
107
108
109 SYNTHESIZED_WIRE_6 <= SYNTHESIZED_WIRE_11 OR SYNTHESIZED_WIRE_12;
110
111
112 SYNTHESIZED_WIRE_12 <= SYNTHESIZED_WIRE_13 AND SYNTHESIZED_WIRE_14;
113
114
115 END bdf_type;
```

Figure 27: Generated VHDL code of the 5 to 1 multiplexer bdf.

```
1  -- Copyright (C) 2022  Intel Corporation. All rights reserved.
2  -- Your use of Intel Corporation's design tools, logic functions
3  -- and other software and tools, and any partner logic
4  -- functions, and any output files from any of the foregoing
5  -- (including device programming or simulation files), and any
6  -- associated documentation or information are expressly subject
7  -- to the terms and conditions of the Intel Program License
8  -- Subscription Agreement, the Intel Quartus Prime License Agreement,
9  -- the Intel FPGA IP License Agreement, or other applicable license
10 -- agreement, including, without limitation, that your use is for
11 -- the sole purpose of programming logic devices manufactured by
12 -- Intel and sold by Intel or its authorized distributors.  Please
13 -- refer to the applicable agreement for further details, at
14 -- https://fpgasoftware.intel.com/eula.
15
16 -- PROGRAM     "Quartus Prime"
17 -- VERSION     "Version 22.1std.0 Build 915 10/25/2022 SC Lite Edition"
18 -- CREATED     "Thu Mar  7 11:26:16 2024"
19
20 LIBRARY ieee;
21 USE ieee.std_logic_1164.all;
22
23 LIBRARY work;
24
25 ENTITY decoder3to8 IS
26    PORT
27    (
28       Input_S0 :  IN  STD_LOGIC;
29       Input_S1 :  IN  STD_LOGIC;
30       Input_S2 :  IN  STD_LOGIC;
31       Output_Y0 :  OUT  STD_LOGIC;
32       Output_Y1 :  OUT  STD_LOGIC;
33       Output_Y2 :  OUT  STD_LOGIC;
34       Output_Y3 :  OUT  STD_LOGIC;
35       Output_Y4 :  OUT  STD_LOGIC;
36       Output_Y5 :  OUT  STD_LOGIC;
37       Output_Y6 :  OUT  STD_LOGIC;
38       Output_Y7 :  OUT  STD_LOGIC
39    );
40 END decoder3to8;
41
42 ARCHITECTURE bdf_type OF decoder3to8 IS
43
44 SIGNAL   SYNTHESIZED_WIRE_12 :  STD_LOGIC;
45 SIGNAL   SYNTHESIZED_WIRE_13 :  STD_LOGIC;
46 SIGNAL   SYNTHESIZED_WIRE_14 :  STD_LOGIC;
47
48
49 BEGIN
50
51
52
53 Output_Y0 <= SYNTHESIZED_WIRE_12 AND SYNTHESIZED_WIRE_13 AND SYNTHESIZED_WIRE_14;
54
55
56 Output_Y1 <= SYNTHESIZED_WIRE_12 AND SYNTHESIZED_WIRE_13 AND Input_S0;
57
58
59 SYNTHESIZED_WIRE_12 <= NOT(Input_S2);
60
61
62
63 Output_Y2 <= SYNTHESIZED_WIRE_12 AND Input_S1 AND SYNTHESIZED_WIRE_14;
64
65
66 Output_Y3 <= SYNTHESIZED_WIRE_12 AND Input_S1 AND Input_S0;
67
68
69 Output_Y5 <= Input_S2 AND SYNTHESIZED_WIRE_13 AND Input_S0;
70
71
72 Output_Y4 <= Input_S2 AND SYNTHESIZED_WIRE_13 AND SYNTHESIZED_WIRE_14;
73
74
75 Output_Y6 <= Input_S2 AND Input_S1 AND SYNTHESIZED_WIRE_14;
76
77
78 Output_Y7 <= Input_S2 AND Input_S1 AND Input_S0;
79
80
81 SYNTHESIZED_WIRE_14 <= NOT(Input_S0);
82
83
84
85 SYNTHESIZED_WIRE_13 <= NOT(Input_S1);
86
87
88
89 END bdf_type;
```

Figure 28: Generated VHDL code of the 3 to 8 decoder bdf.

```
1  -- Copyright (C) 2022  Intel Corporation. All rights reserved.
2  -- Your use of Intel Corporation's design tools, logic functions
3  -- and other software and tools, and any partner logic
4  -- functions, and any output files from any of the foregoing
5  -- (including device programming or simulation files), and any
6  -- associated documentation or information are expressly subject
7  -- to the terms and conditions of the Intel Program License
8  -- Subscription Agreement, the Intel Quartus Prime License Agreement,
9  -- the Intel FPGA IP License Agreement, or other applicable license
10 -- agreement, including, without limitation, that your use is for
11 -- the sole purpose of programming logic devices manufactured by
12 -- Intel and sold by Intel or its authorized distributors.  Please
13 -- refer to the applicable agreement for further details, at
14 -- https://fpgasoftware.intel.com/eula.
15
16 -- PROGRAM     "Quartus Prime"
17 -- VERSION     "Version 22.1std.0 Build 915 10/25/2022 SC Lite Edition"
18 -- CREATED     "Thu Mar  7 11:26:28 2024"
19
20 LIBRARY ieee;
21 USE ieee.std_logic_1164.all;
22
23 LIBRARY work;
24
25 ENTITY encoder8to3 IS
26    PORT
27    (
28       Input_A :  IN  STD_LOGIC;
29       Input_B :  IN  STD_LOGIC;
30       Input_C :  IN  STD_LOGIC;
31       Input_D :  IN  STD_LOGIC;
32       Input_E :  IN  STD_LOGIC;
33       Input_F :  IN  STD_LOGIC;
34       Input_G :  IN  STD_LOGIC;
35       Input_H :  IN  STD_LOGIC;
36       Output_Y0 :  OUT  STD_LOGIC;
37       Output_Y1 :  OUT  STD_LOGIC;
38       Output_Y2 :  OUT  STD_LOGIC
39    );
40 END encoder8to3;
41
42 ARCHITECTURE bdf_type OF encoder8to3 IS
43
44
45
46 BEGIN
47
48
49
50 Output_Y0 <= Input_B OR Input_F OR Input_H OR Input_D;
51
52
53 Output_Y1 <= Input_C OR Input_G OR Input_H OR Input_D;
54
55
56 Output_Y2 <= Input_E OR Input_G OR Input_H OR Input_F;
57
58
59 END bdf_type;
```

Figure 29: Generated VHDL code of the 8 to 3 encoder bdf.

```
1  -- Copyright (C) 2022  Intel Corporation. All rights reserved.
2  -- Your use of Intel Corporation's design tools, logic functions
3  -- and other software and tools, and any partner logic
4  -- functions, and any output files from any of the foregoing
5  -- (including device programming or simulation files), and any
6  -- associated documentation or information are expressly subject
7  -- to the terms and conditions of the Intel Program License
8  -- Subscription Agreement, the Intel Quartus Prime License Agreement,
9  -- the Intel FPGA IP License Agreement, or other applicable license
10 -- agreement, including, without limitation, that your use is for
11 -- the sole purpose of programming logic devices manufactured by
12 -- Intel and sold by Intel or its authorized distributors.  Please
13 -- refer to the applicable agreement for further details, at
14 -- https://fpgasoftware.intel.com/eula.
15
16 -- PROGRAM     "Quartus Prime"
17 -- VERSION     "Version 22.1std.0 Build 915 10/25/2022 SC Lite Edition"
18 -- CREATED     "Thu Mar  7 11:26:37 2024"
19
20 LIBRARY ieee;
21 USE ieee.std_logic_1164.all;
22
23 LIBRARY work;
24
25 ENTITY demux1to2 IS
26    PORT
27    (
28       Input_S :  IN  STD_LOGIC;
29       Input_A :  IN  STD_LOGIC;
30       Output_Y0 :  OUT  STD_LOGIC;
31       Output_Y1 :  OUT  STD_LOGIC
32    );
33 END demux1to2;
34
35 ARCHITECTURE bdf_type OF demux1to2 IS
36
37 SIGNAL   SYNTHESIZED_WIRE_0 :  STD_LOGIC;
38
39
40 BEGIN
41
42
43
44 SYNTHESIZED_WIRE_0 <= NOT(Input_S);
45
46
47
48 Output_Y0 <= SYNTHESIZED_WIRE_0 AND Input_A;
49
50
51 Output_Y1 <= Input_A AND Input_S;
52
53
54 END bdf_type;
```

Figure 30: Generated VHDL code of the 1 to 2 demultiplexer bdf.

Conclusion: This exercise offered a practical exploration of digital circuit design and simulation techniques through the creating and testing of muxes, encoders, and decoders. By constructing these in Quartus and verifying its functionality in ModelSim, we gained an understanding of the behavior of muxes, encoders, and decoders, implemented them using both gates and VHDL code, as well as verifying the correctness of designs through waveform simulation sin ModelSim.