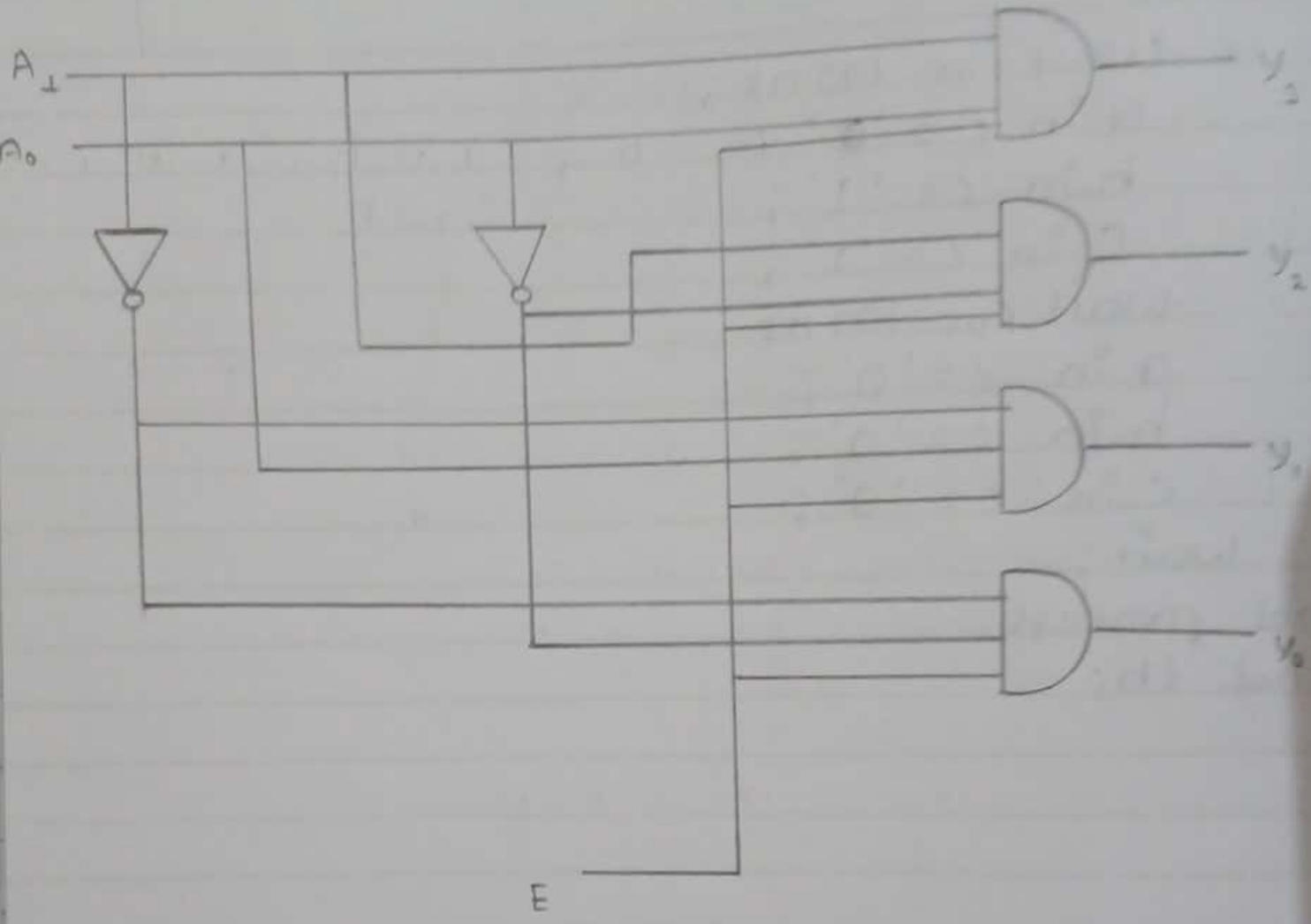
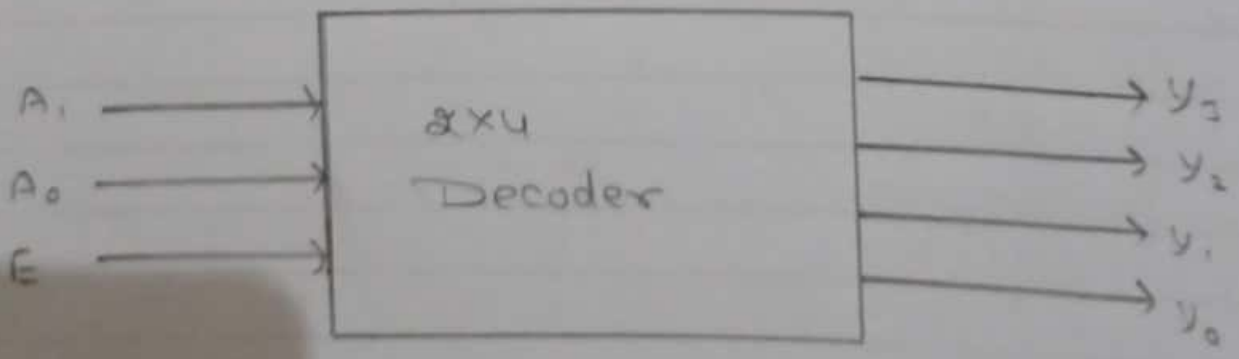

A1
A0
E


A1
A0
E
2x4
Decoder

Decoder

Objective:- To design and simulate the decoder circuit.

- **Description:-**

A decoder is a combinational circuit with n inputs and upto 2^n outputs, where each output corresponds to a unique binary input. It uses logic gates (AND, OR, NOT) to decode binary inputs and activate a single output line while deactivating others. Some decoders include an enable input to control activation.

- **Truth Table:-**

Inputs		Outputs			
A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Equations:-

$D_0 = \bar{A}_1 \cdot \bar{A}_0$

$D_1 = \bar{A}_1 \cdot A_0$

$D_2 = A_1 \cdot \bar{A}_0$

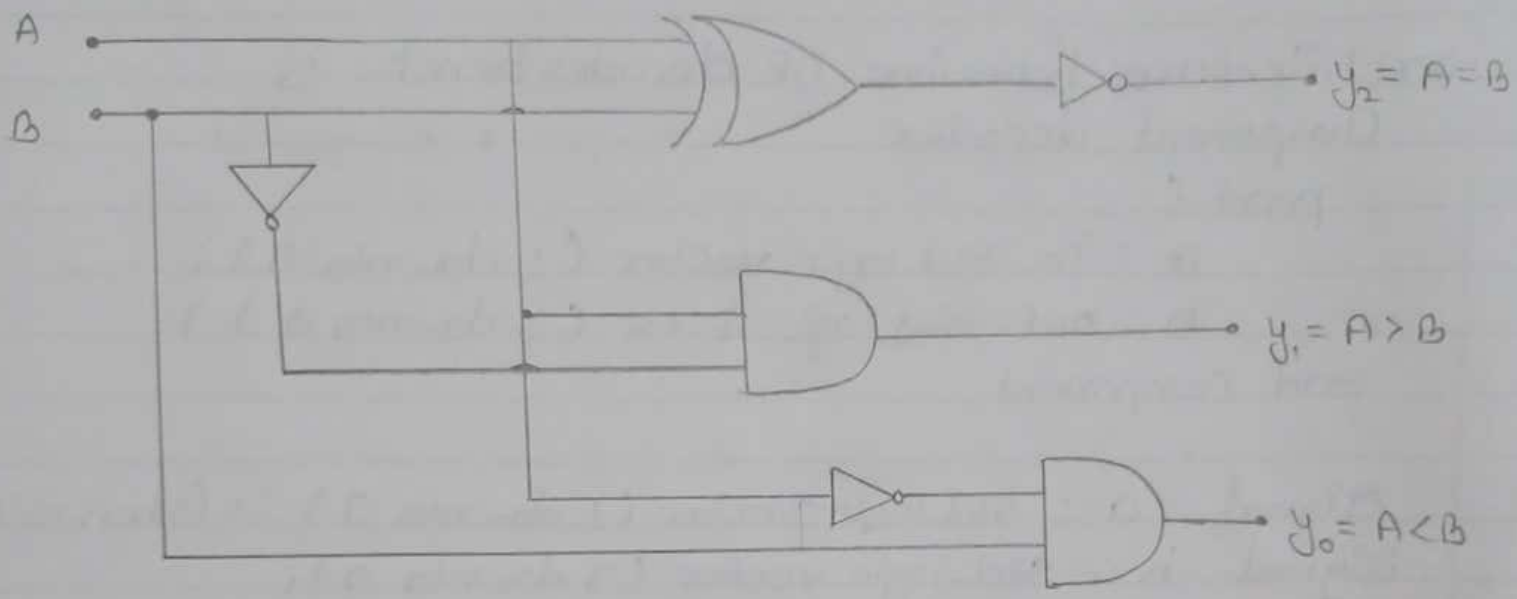
$D_3 = A_1 \cdot A_0$

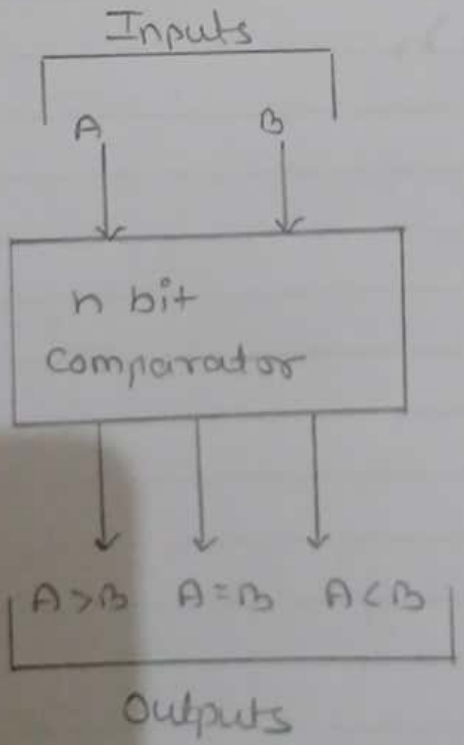
Teacher's Signature

- VHDL Code :-

```
library IEEE;
use IEEE.STD_Logic_1164.all;

entity decoder is
  port (
        a : in STD_logic_VECTOR (1 downto 0);
        b : out STD_Logic_VECTOR (3 downto 0));
end decoder;

architecture bhv of decoder is
begin
      b(0) <= not a(0) and not a(1);
      b(1) <= not a(0) and a(1);
      b(2) <= a(0) and not a(1);
      b(3) <= a(0) and a(1);
end bhv;


-- Test Bench --

library IEEE;
Use IEEE.Std_logic_1164.All;

entity decoderbench is
end decoderbench;
```

```
architecture behaviour of decoderbench is
  Component decoder
    port (
        a : in std_logic_vector (1 downto 0);
        b : out std_logic_vector (3 downto 0) );
  end component;

  signal a : std_logic_vector (1 downto 0) := (others => '0');
  signal b : std_logic_vector (3 downto 0);

  Begin
  uut : decoder port map (
              a => a,
              b => b );

  stim_proc : process
  begin
      wait for 10 ns;
      a <= "00";
      wait for 10 ns;
      a <= "01";
      wait for 10 ns;
      a <= "10";
      wait for 10 ns;
      a <= "11";
    wait;
    end process;
  End;
```


A
B
$y_2 = A = B$
$y_1 = A > B$
$y_0 = A < B$


Inputs
A
B
n bit
Comparator
A>B
A=B
A<B
Outputs

Date

Expt. No. Page No. 34

Comparator :-

- Objective :-

To design a comparator circuit that compares two binary numbers and outputs their relationship, such as greater than ($A > B$), less than ($A < B$), or equal ($A = B$), for use in decision-making in digital systems.

- Description :-

A comparator is a combinational logic circuit that compares two binary input (A and B) and produces outputs indicating their relationship. It uses logics gates to evaluate the comparison.

- Inputs : Two binary Numbers, A and B.
- Outputs :
  - $A > B$: High if A is Greater.
  - $A < B$: High if A is Smaller.
  - $A = B$: High if A and B are equal.

- Truth Table :

Inputs		outputs		
A	B	Y_0	Y_1	Y_2
0	0	0	0	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

$Y_0 = A < B$
$Y_1 = A > B$
$Y_2 = A = B$

Teacher's Signature

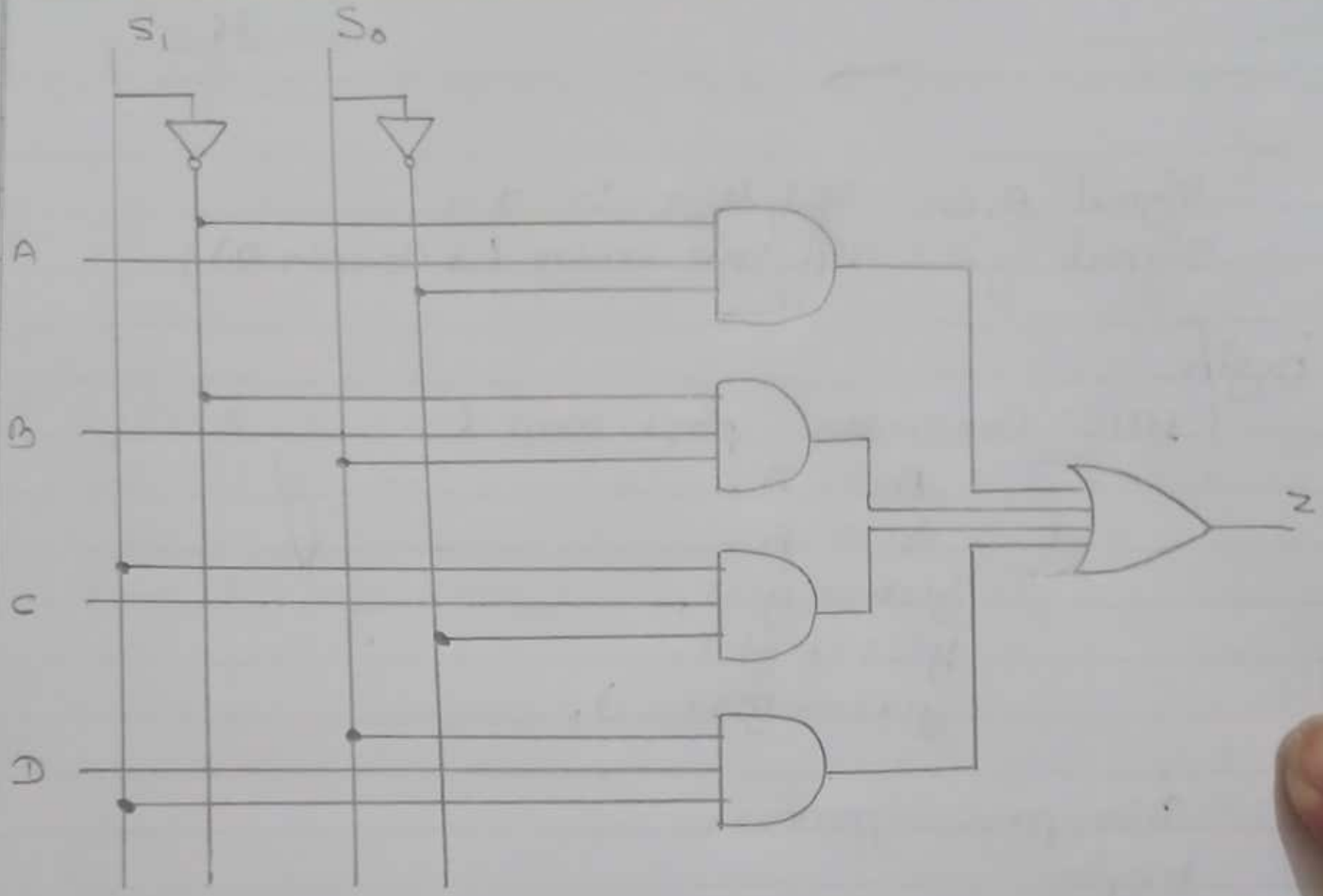
- VHDL code :-

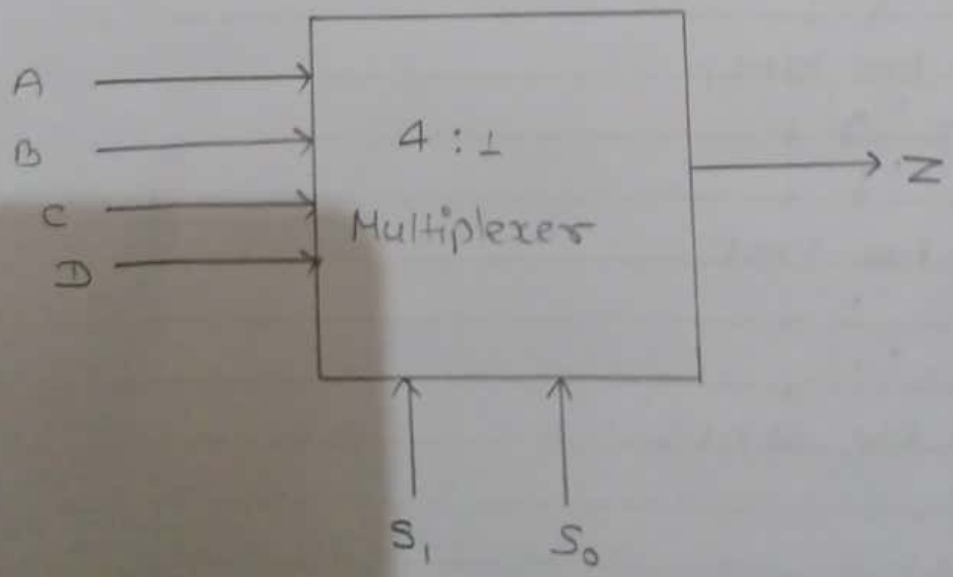
```
library IEEE;
use IEEE.Std_logic_1164.all;
entity comarator is
  port (
        A,B : in Std_logic;
  y(0),y(1),y(2)  : out Std_logic );
end ~~std~~ Comperator;

Architecture Comp_arch OF Comperator is
  begin
      y(0) <= (not A) and B;
      y(1) <= A and (not B);
      y(2) <= A xnor B;
end Comp_arch;
```

- -- Test Bench :- --

```
library IEEE;
use IEEE.Std_logic_1164.all;
entity tb_Comparator is
end tb_Comperator;

Architecture testbench of tb_Comperator is
    Component Comperator
        port ( A,B : in std_logic;
               y   : out Std_logic_vector (2 downto 0));
    end Component;
```

```
Signal A, B : std_logic := '0';
Signal y : std_logic_vector (2 downto 0);

begin
    uut : Comparator port map (
            A => A,
            B => B,
            y(0) => y(0),
            y(1) => y(1),
            y(2) => y(2)  );

    Stim_proc : process
    begin
            A <= '0';
            B <= '0';
            wait for 10 ns;
            A <= '1';
            B <= '0';
            wait for 10ns;
            A <= '0';
            B <= '1';
            wait for 10ns
            A <= '1';
            B <= '1';
            wait for 10 ns;
            wait;
    end process;
end testbench;
```


S1
S0
A
B
C
D
Z


A
B
C
D
4 : 1
Multiplexer
Z
S1
S0

Date

Expt. No.

Page No. 37

Multiplexer :

- Objective :- To design a multiplexer (MUX) circuit that selects one input from multiple data input based on control signals and forwards it to a single output line, optimizing data selection and routing in digital systems.

- Description :-

A multiplexer is a combinational circuit with Input lines, Control signals, and a single output line. Based on the binary value of the control signals, the multiplexer routes one specific input to the output.

  - Inputs : 2^n inputs, where n is the number of control lines.
  - Control Signals : Select which input is routed to the output.
  - Output : A single line carrying the select input.

- Truth Table :-

Input		Output
S_1	S_0	Z
0	0	A
0	1	B
1	0	C
1	1	D

Teacher's Signature

VHDL Code :-

```
library IEEE;
Use IEEE.std_logic_1164.all;
entity multiplexer is
    port (
        A, B C, d : in std_logic;
        S0, S1 : in std_logic;
        Z : out std_logic );
end multiplexer;

architecture bhv of multiplexer 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 bhv;
```

```
-- Test Bench --
library ieee;
use ieee.std_logic_1164.all;
entity multibench is
end multibench;

Architecture behavior of multibench is
    Component multiplexer
    port (
            A : in std_logic;
            B : in std_logic;
            C : in std_logic;
            D : in std_logic;
            S0 : in std_logic;
            S1 : in std_logic;
            Z : out std_logic );
    end component;

    Signal A : std_logic := '0';
    Signal B : std_logic := '0';
    Signal C : std_logic := '0';
    Signal D : std_logic := '0';
    Signal S0 : std_logic := '0';
    Signal S1 : std_logic := '0';
    Signal z : std_ulogic;

Begin
    uut : multiplexer port map (
```

```
A => A,
B => B,
C => C,
D => D,
S0 => S0,
S1 => S1,
Z => Z );

Stim_proc: process
begin
    wait for 100 ns;
    A <= '1';
    B <= '0';
    C <= '1';
    D <= '0';
    S0 <= '0';
    S1 <= '0';
    wait for 100ns;
    S0 <= '1';
    S1 <= '0';
    wait for 100 ns;
    S0 <= '0';
    S1 <= '1';
    wait for 100 ns;
    S0 <= '1';
    S1 <= '1';
    wait for 100ns;
end: process;
End;
```

Date

Expt. No. Page No. 41

ALU :-

- Objective :- To design an Arithmetic Logic Unit (ALU) that performs arithmetic operations (e.g., addition, subtraction) and logic operations (e.g., AND, OR, NOT) on binary data, serving as the core computational component of a processor.

- Description : An ALU is a combinational circuit within a CPU that processes data by executing arithmetic and logical operations. It takes binary inputs, processes them based on a control signal (operation code), and generates the result as output.

  - Inputs : Operands (binary numbers) and control signals to specify the operation.
  - Outputs : Result of the specified operation.
  - Operations : Addition, subtraction, bitwise AND/OR, shifts, comparisons.

- VHDL Code :

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.Numeric_Std.all;
```

Teacher's Signature

```
entity alu is
    port (inp_a : in signed (3 downto 0);
          inp_b : in signed (3 downto 0);
          sel : in std_logic_vector (2 downto 0);
          out_alu : out signed (3 downto 0));
end alu;

architecture Behavioral of alu is
begin
    process (inp_a, inp_b, sel)
    begin
        case sel is
            when "000" =>
                out_alu <= inp_a+inp_b;
            when "001" =>
                out_alu <= inp_a - inp_b;
            when "010" =>
                out_alu <= inp_a - 1;
            when "011" =>
                out_alu <= inp_a+1;
            when "100" =>
                out_alu <= inp_a and inp_b;
            when "101" =>
                out_alu <= inp_a or inp_b;
            when "110" =>
                out_alu <= not inp_a;
            when "111" =>
                out_alu <= inp_a xor inp_b;
```

```
            When others =>
              Null ;
            end case;
        end process;
end Behavioral;


-- Test bench --

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Tb_alu is
end Tb_alu;

Architecture behavior of Tb_alu is
    Component alu
        port (
            inp_a : in signed (3 downto 0);
            inp_b : in signed (3 downto 0);
            sel : in std_logic_vector (2 downto 0);
            out_alu : out signed (3 downto 0) );
    end component;

Signal inp_a : signed (3 downto 0) := (others => '0');
Signal inp_b : signed (3 downto 0) := (others => '0');
Signal sel : std_logic_vector (2 downto 0) := (others => '0');
```

```
Signal out_alu : Signed (3 downto 0);

Begin
    uut : alu port map (
            inp_a => inp_a,
            inp_b => inp_b,
            sel => sel,
            out_alu => out_alu );

stim_proc : process
begin
        wait for 100 ns;
        inp_a <= "10001";
        inp_b <= "1111";
        sel <= "000";
        wait for 100 ns;
        sel <= "001";
        wait for 100 ns;
        sel <= "010";
        wait for 100 ns;
        sel <= "011";
        wait for 100 ns;
        Sel <= "100";
        wait for 100 ns,
        Sel <= "101";
        wait for 100ns;
        Sel <= "110";
        wait for 100 ns;
```

```
        Sel <= "111";
        end process
end;
```

- Truth Table:

Expt. No. _____ Date _____

Page No. 46

Multiplier :

- Objective : To design a multiplier circuit that performs binary multiplication of two numbers, producing a product efficiently for use in digital systems such as processors, signal processing, and arithmetic units.

- Description :-

A multiplier is a combinational circuit that computes the product of two binary numbers. It uses adders and shift operations to perform the multiplication process, either in parallel or sequentially.

- Truth Table :-

Term	Inputs				Outputs			
	A_0	A_1	B_0	B_1	P_0	P_1	P_2	P_3
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0
2	0	0	1	0	0	0	0	0
3	0	0	1	1	0	0	0	0
4	0	1	0	0	0	0	0	0
5	0	1	0	1	0	0	0	1
6	0	1	1	0	0	0	1	0
7	0	1	1	1	0	0	1	1

Teacher's Signature _____

8	1	0	0	0	0	0	0	0
9	1	0	0	1	0	0	1	0
10	1	0	1	0	0	1	0	0
11	1	0	1	1	0	1	1	0
12	1	1	0	0	0	0	0	0
13	1	1	0	1	0	0	1	1
14	1	1	1	0	0	1	1	0
15	1	1	1	1	1	0	0	1

- VHDL Code :

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

entity multiply_behav is
    port (
        A, B : in bit_vector (1 downto 0);
        P    : out bit_vector (3 downto 0) );

end multiply_behav;

architecture behavioral of multiply_behav is
begin
    process (A, B) is
    begin
        Case A is
```

```
when "00" =>
    if B="00" then P<="0000";
    elsif B="01" then P<="0000";
    elsif B="10" then P<="0000";
    else P<="0000";
    end if
when "01" =>
    if B="00" then P<="0000";
    elsif B="01" then P<="0001";
    elsif B="10" then P<="0010";
    else P<="0011";
    end if;
when "10" =>
    if B="00" then P<="0000";
    elsif B="01" then P<="0010";
    elsif B="10" then P<="0100";
    else P<="0110";
    end if
when "11" =>
    if B="00" then P<="0000";
    elsif B="01" then P<="0011";
    elsif B="10" then P<="0110";
    else P<="1001";
    end if;
end case;
end process;
end architecture;
```

```
-- Test bench --

library ieee;
use ieee.std_logic_1164.all;

entity multiply_behav_tb is
end multiply_behav_tb;

architecture tb of multiply_behav_tb is
    component multiply_behav is
        port (
            A,B : in bit_vector (1 downto 0);
            P : out bit_vector ( 3 downto 0) );
    end component;

Signal A, B : bit_vector (1 downto 0);
Signal P : bit_vector (3 downto 0);
begin
    uut : multiply_behav port map (
            A => A;
            B => B
            P => P );

Force : process
Constant period : time := 20 ns;
begin
        A <= "00";
        B <= "00";
        wait for period;
```

```
A <= "00";
B <= "01";
Wait for period;
A <= "00";
B <= "10";
Wait for period;
A <= "00";
B <= "11";
Wait for period;
A <= "01";
B <= "00";
Wait for period;
A <= "01";
B <= "01";
Wait for period;
A <= "01"
B <= "10";
Wait for period;
A <= "01"
B <= "11"
Wait for period;
A <= "10";
B <= "00";
Wait for period
A <= "10";
B <= "01";;
Wait for period;
```

```
A <= "10";
B <= "10";
Wait for period;
A <= "10";
B <= "11";
Wait for period;
A <= "11";
B <= "00";
Wait for period;
A <= "11";
B <= "01";
Wait for period;
A <= "11";
B <= "10";
Wait for period;
A <= "11"
B <= "11";
Wait for period;
Wait;
end process;
end th;
```