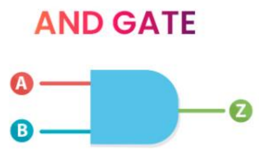
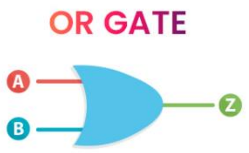
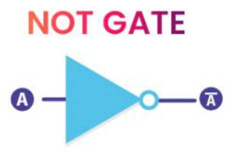
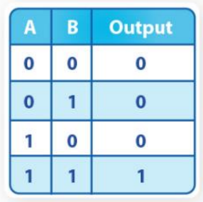
II. PROCEDURE

II.1 LAB EXPERIMENT 1 : WRITE VHDL CODE TO IMPLEMENT ALL LOGIC GATES IN FPGA KIT.

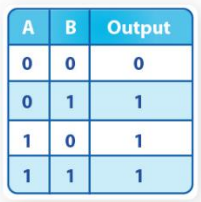




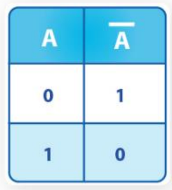




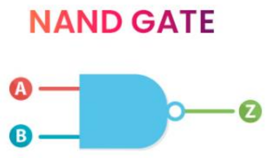
A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

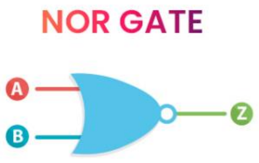


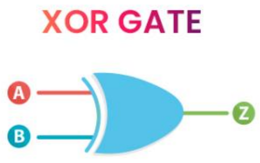
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

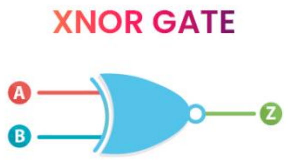


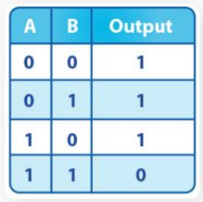
A	$\overline{A}$
0	1
1	0



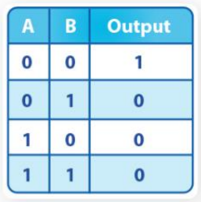




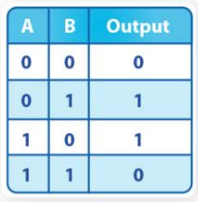




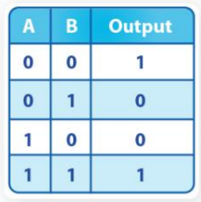
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Top module

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

-- Simple module that connects the SW switches to the LEDR lights
ENTITY lab4_ex1 IS
PORT (SW        : IN  STD_LOGIC_VECTOR(17 DOWNTO 0);
        LEDR   : OUT STD_LOGIC_VECTOR(17 DOWNTO 0);
        LEDG  : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));

END lab4_ex1;

ARCHITECTURE Structure OF lab4_ex1 IS
        COMPONENT all_gates
                        PORT (A: in std_logic;
                                   B: in std_logic;
                                   Y_AND: out std_logic;
                                   Y_OR: out std_logic;
                                   Y_NOT_A: out std_logic;
                                   Y_NAND: out std_logic;
                                   Y_NOR: out std_logic;
                                   Y_XOR: out std_logic;
                                   Y_XNOR: out std_logic
                              );

     END COMPONENT;

     BEGIN
          LEDR <= SW;
          DUT: all_gates PORT
MAP(SW(1),SW(0),LEDG(6),LEDG(5),LEDG(4),LEDG(3),LEDG(2),LEDG(1),LEDG(0));
     END Structure;
```

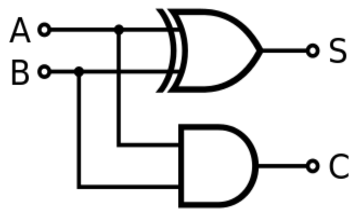
Structural	**Dataflow**	**Behavior**
library IEEE;	library ieee;	library IEEE;
use IEEE.STD_LOGIC_1164.ALL;	use ieee.std_logic_1164.all;	use
	entity all_gates is	IEEE.STD_LOGIC_1164.ALL;
entity all_gates is	port(A: in std_logic;	entity all_gates is
port(A: in std_logic;	B: in std_logic;	port(A: in std_logic;
B: in std_logic;	Y_AND: out std_logic;	B: in std_logic;
Y_AND: out std_logic;	Y_OR: out	Y_AND: out std_logic;
Y_OR: out std_logic;	std_logic;	Y_OR: out std_logic;
Y_NOT_A: out	Y_NOT_A: out	Y_NOT_A: out std_logic;
std_logic;	std_logic;	Y_NAND: out std_logic;
Y_NAND: out	Y_NAND: out	Y_NOR: out std_logic;
std_logic;	std_logic;	Y_XOR: out std_logic;
Y_NOR: out std_logic;	Y_NOR: out	Y_XNOR: out std_logic
Y_XOR: out std_logic;	std_logic;	);
Y_XNOR: out	Y_XOR: out	end all_gates;
std_logic	std_logic;	
);	Y_XNOR: out	architecture behavioral of
end all_gates;	std_logic	all_gates is
	);	begin
architecture structural of all_gates is	end all_gates;	process(A, B)
-- Declare components for each gate		begin
component AND_GATE		Y_AND <= A and B;
port (A, B: in std_logic; Y: out	architecture dataflow of all_gates is	Y_OR <= A or B;
std_logic);	begin	Y_NOT_A <= not A;
end component;	Y_AND <= A and B;	Y_NAND <= A nand B;
	Y_OR<= A or B;	Y_NOR <= A nor B;
component OR_GATE	Y_NOT_A <= not A;	Y_XOR <= A xor B;
port (A, B: in std_logic; Y: out	Y_NAND <= A nand B;	Y_XNOR <= A xnor B;
std_logic);	Y_NOR <= A nor B;	end process;
end component;	Y_XOR <= A xor B;	end behavioral;
	Y_XNOR <= A xnor B;	
component NOT_GATE	end dataflow;	
port (A: in std_logic; Y: out		
std_logic);		
end component;		

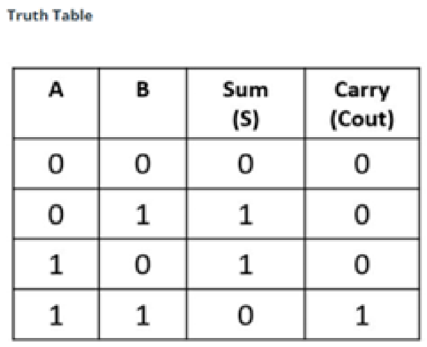
component NAND_GATE
port (A, B: in std_logic; Y: out std_logic);
end component;
 	
component NOR_GATE
port (A, B: in std_logic; Y: out std_logic);
end component;
 	
component XOR_GATE
port (A, B: in std_logic; Y: out std_logic);
end component;
 	
component XNOR_GATE
port (A, B: in std_logic; Y: out std_logic);
end component;
 	
begin
-- Instantiate the components
u1: AND_GATE port map(A => A, B => B, Y => Y_AND);
u2: OR_GATE port map(A => A, B => B, Y => Y_OR);
u3: NOT_GATE port map(A => A, Y => Y_NOT_A);
u4: NAND_GATE port map(A => A, B => B, Y => Y_NAND);
u5: NOR_GATE port map(A => A, B => B, Y => Y_NOR);
u6: XOR_GATE port map(A => A, B => B, Y => Y_XOR);
u7: XNOR_GATE port map(A => A, B => B, Y => Y_XNOR);
 	
end structural;
 	
-- AND Gate Component
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
 	
entity AND_GATE is
port (A, B: in std_logic;
Y: out std_logic);
end AND_GATE;
 	
architecture dataflow of AND_GATE is
begin
Y <= A and B;
end dataflow;
 	
-- OR Gate Component
library IEEE;		

```
use IEEE.STD_LOGIC_1164.ALL;

entity OR_GATE is
    port (A, B: in std_logic;
          Y: out std_logic);
end OR_GATE;

architecture dataflow of OR_GATE is
begin
    Y <= A or B;
end dataflow;

-- NOT Gate Component
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity NOT_GATE is
    port (A: in std_logic;
          Y: out std_logic);
end NOT_GATE;

architecture dataflow of NOT_GATE is
begin
    Y <= not A;
end dataflow;

-- NAND Gate Component
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity NAND_GATE is
    port (A, B: in std_logic;
          Y: out std_logic);
end NAND_GATE;

architecture dataflow of NAND_GATE
is
begin
    Y <= A nand B;
end dataflow;

-- NOR Gate Component
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity NOR_GATE is
    port (A, B: in std_logic;
          Y: out std_logic);
end NOR_GATE;

architecture dataflow of NOR_GATE is
begin
    Y <= A nor B;
end dataflow;
```

```
-- XOR Gate Component
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity XOR_GATE is
    port (A, B: in std_logic;
          Y: out std_logic);
end XOR_GATE;

architecture dataflow of XOR_GATE is
begin
    Y <= A xor B;
end dataflow;

-- XNOR Gate Component
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity XNOR_GATE is
    port (A, B: in std_logic;
          Y: out std_logic);
end XNOR_GATE;

architecture dataflow of XNOR_GATE
is
begin
    Y <= A xnor B;
end dataflow;
```

II.2 LAB EXPERIMENT 2 : WRITE VHDL CODE TO IMPLEMENT THE HALF ADDER CIRCUIT IN FPGA KIT:




Truth Table

A	B	Sum (S)	Carry (Cout)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Top module

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

entity lab4_ex2 is
        port (SW: in std_logic_vector(17 downto 0);
                    LEDR: out std_logic_vector(17 downto 0);
                    LEDG: out std_logic_vector(7 downto 0));
end lab4_ex2;

architecture structure of lab4_ex2 is
        component half_adder
                port(a,b: in std_logic;
                        s,c: out std_logic);
        end component;
        begin
                LEDR <= SW;
                DUT: half_adder port map(SW(1),SW(0),LEDG(1),LEDG(0));
        end structure;
```

Structural

```
library ieee;
use ieee.std_logic_1164.all;
entity half_adder is
        port(a,b: in std_logic;
                 s,c: out std_logic);
end half_adder;

architecture structural of half_adder is
        component and_gate
                port(a,b: in std_logic;
y: out std_logic);
        end component;

        component xor_gate
                port(a,b: in std_logic;
y: out std_logic);
        end component;

        begin
                u1: and_gate port
map(a => a,b=>b,y=>c);
                u2: xor_gate port
map(a =>a,b=>b,y=>s);
        end structural;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity and_gate is
   port (a, b: in std_logic;
       y: out std_logic);
end and_gate;

architecture dataflow of and_gate is
begin
   y <= a and b;
end dataflow;

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

Dataflow

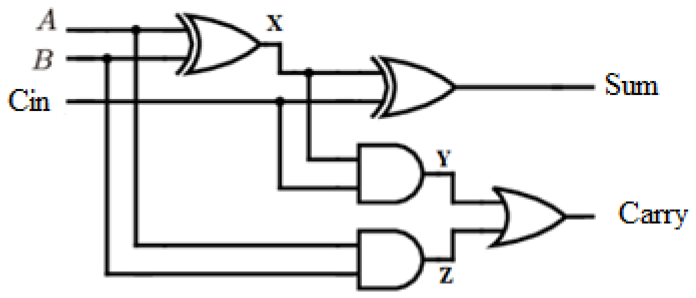
```
library ieee;
use ieee.std_logic_1164.all;
entity half_adder is
        port(a,b: in std_logic;
                 s,c: out std_logic);
end half_adder;

architecture dataflow of half_adder is
        begin
                s <= a xor b;
                c <= a and b;
        end dataflow;
```

Behavior

```
library ieee;
use ieee.std_logic_1164.all;
entity half_adder is
        port(a,b: in std_logic;
                 s,c: out std_logic);
end half_adder;

architecture behavior of half_adder is
begin
        process(a,b)
        begin
                s <= a xor b;
                c <= a and b;
        end process;
end behavior;
```

```
entity xor_gate is
    port (a, b: in std_logic;
          y: out std_logic);
end xor_gate;

architecture dataflow of xor_gate is
begin
    y <= a xor b;
end dataflow;
```

II.3 EXPERIMENT 3: WRITE VHDL CODE TO SIMULATE AND IMPLEMENT THE FULL ADDER CIRCUIT IN FPGA KIT:



Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Top module

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

entity lab4_ex3 is
    port (
        SW   : in  std_logic_vector(15 downto 0);
        LEDR : out std_logic_vector(15 downto 0);
        LEDG : out std_logic_vector(7 downto 0)
    );
end lab4_ex3;

architecture structural of lab4_ex3 is
    component full_adder is
        port (
            a, b, cin : in  std_logic;
            sum, carry : out std_logic
        );
    end component;

begin
    LEDR <= SW;
    DUT: full_adder port map (SW(2), SW(1), SW(0), LEDG(1), LEDG(0));
end structural;
```

Structural	**Dataflow**	**Behavior**
library ieee; use ieee.std_logic_1164.all;	library ieee; use ieee.std_logic_1164.all;	library ieee; use ieee.std_logic_1164.all;

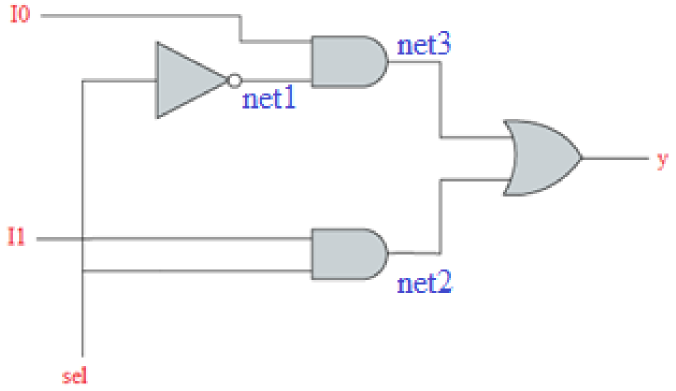
```
entity full_adder is
    port (
        a, b, cin : in  std_logic;
        sum, carry : out std_logic
    );
end full_adder;

architecture structural of full_adder is
    signal x, y, z : std_logic;

    component and_gate
        port (a, b : in  std_logic; y : out
std_logic);
    end component;

    component or_gate
        port (a, b : in  std_logic; y : out
std_logic);
    end component;

    component xor_gate
        port (a, b : in  std_logic; y : out
std_logic);
    end component;

begin
    x1: xor_gate port map (a => a, b =>
b, y => x);
    x2: xor_gate port map (a => x, b =>
cin, y => sum);
    a1: and_gate port map (a => x, b =>
cin, y => y);
    a2: and_gate port map (a => a, b =>
b, y => z);
    a3: or_gate port map (a => y, b => z,
y => carry);
end structural;

library ieee;
use ieee.std_logic_1164.all;

entity and_gate is
    port (a, b : in  std_logic; y : out
std_logic);
end and_gate;

architecture dataflow of and_gate is
begin
    y <= a and b;
end dataflow;

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

```
entity full_adder is
    port (
        a, b, cin : in  std_logic;
        sum, carry : out std_logic
    );
end full_adder;

architecture dataflow of full_adder is
begin
     sum <= (a xor b) xor cin;
     carry <= (a and b) or ((a xor b) and
cin);
end dataflow;
```

```
entity full_adder is
    port (
        a, b, cin : in  std_logic;
        sum, carry : out std_logic
    );
end full_adder;

architecture behavior of full_adder is
begin
             process(a,b)
             begin
                            sum <= (a xor b) xor
cin;
                            carry <= (a and b) or
((a xor b) and cin);
             end process;
end behavior;
```

entity or_gate is
    port (a, b : in std_logic; y : out std_logic);
end or_gate;
 	
architecture dataflow of or_gate is
begin
    y <= a or b;
end dataflow;
 	
library ieee;
use ieee.std_logic_1164.all;
 	
entity xor_gate is
    port (a, b : in std_logic; y : out std_logic);
end xor_gate;
 	
architecture dataflow of xor_gate is
begin
    y <= a xor b;
end dataflow;		

II.4 EXPERIMENT 4 :WRITE VHDL CODE TO IMPLEMENT 2:1 MULTIPLEXER CIRCUIT IN FPGA KIT :



Truth Table

sel	y
0	i_0
1	i_1

Top module

```
library ieee;
use ieee.std_logic_1164.all;
entity lab4_ex4 is
            port (SW            : in std_logic_vector(17 downto 0);
                        LEDR   : out std_logic_vector(17 downto 0);
                        LEDG  : out std_logic_vector(7 downto 0));
end lab4_ex4;
```

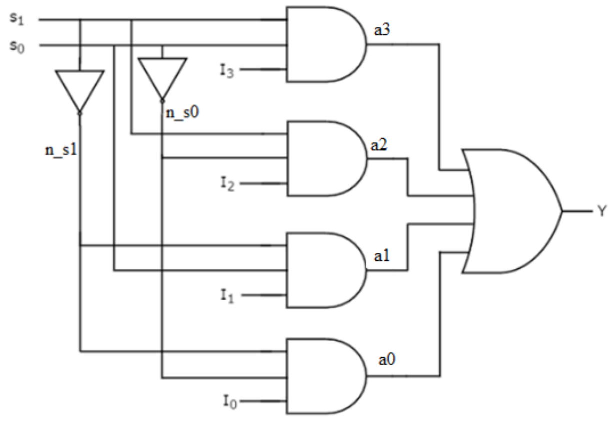
```
architecture structural of lab4_ex4 is
        component mux21
                port(i: in std_logic_vector(1 downto 0);
                        sel: in std_logic;
                        y: out std_logic);
        end component;

        begin
                LEDR <= SW;
                DUT: mux21 port map(SW(1 downto 0),SW(2),LEDG(7));
        end structural;
```

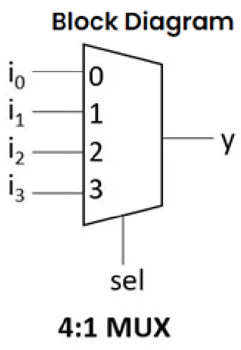
Code

```
library ieee;
use ieee.std_logic_1164.all;
entity mux21 is
        port(i: in std_logic_vector(1 downto 0);
                sel: in std_logic;
                y: out std_logic
        );
end mux21;

architecture structural of mux21 is
signal x: std_logic_vector(3 downto 0);
component and_gate
        port(a,b: in std_logic;
                y:out std_logic );
end component;

component or_gate
        port(a,b: in std_logic;
                y:out std_logic );
end component;

component intervert
        port(a: in std_logic;
                y: out std_logic);
end component;

begin
  -- Implement the 2-to-1 multiplexer logic
  g1: intervert port map(a => sel, y => x(0));
  g2: and_gate port map(a => i(0), b => x(0), y => x(1));
  g3: and_gate port map(a => i(1), b => sel, y => x(2));
  g4: or_gate port map(a => x(1), b => x(2), y => x(3));

  -- Connect z to output y
  y <= x(3);
end structural;

library ieee;
use ieee.std_logic_1164.all;
entity and_gate is
port(   a: in std_logic;
          b: in std_logic;
```

```
                 y: out std_logic
);
end and_gate;

architecture dataflow of and_gate is
begin
    y <= a and b;
end dataflow;

library ieee;
use ieee.std_logic_1164.all;
entity or_gate is
port(     a: in std_logic;
                b: in std_logic;
                y: out std_logic
);
end or_gate;
architecture dataflow of or_gate is
begin
    y <= a or b;
end dataflow;

library ieee;
use ieee.std_logic_1164.all;
entity intervert is
port(     a: in std_logic;
                 y: out std_logic
);
end intervert;
architecture dataflow of intervert is
begin
    y <= not a;
end dataflow;
```

II.5 EXPERIMENT 5 : WRITE VHDL CODES TO IMPLEMENT 4:1 MULTIPLEXER CIRCUIT IN FPGA KIT:





Truth Table

sel[0]	sel[1]	y
0	0	i_0
0	1	i_1
1	0	i_2
1	1	i_3

Top module

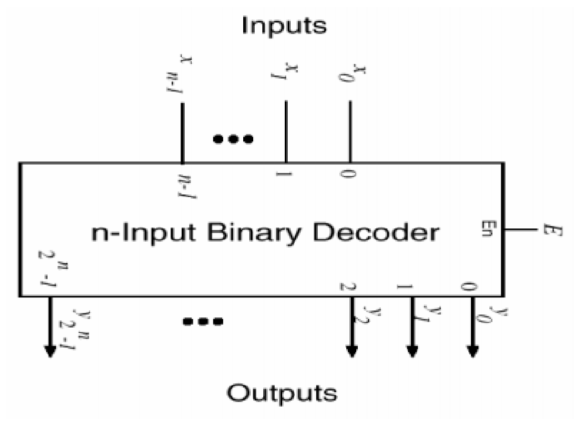
```
library ieee;
```

```
use ieee.std_logic_1164.all;
entity lab4_ex5 is
        port (SW        : in std_logic_vector(17 downto 0);
                LEDR   : out std_logic_vector(17 downto 0);
                LEDG  : out std_logic_vector(7 downto 0));
end lab4_ex5;

architecture structural of lab4_ex5 is
        component mux41
                port(i: in std_logic_vector(3 downto 0);
                        sel: in std_logic_vector(1 downto 0);
                        y: out std_logic);
        end component;

        begin
                LEDR <= SW;
                DUT: mux41 port map(SW(3 downto 0),SW(5 downto 4),LEDG(7));
        end structural;
```

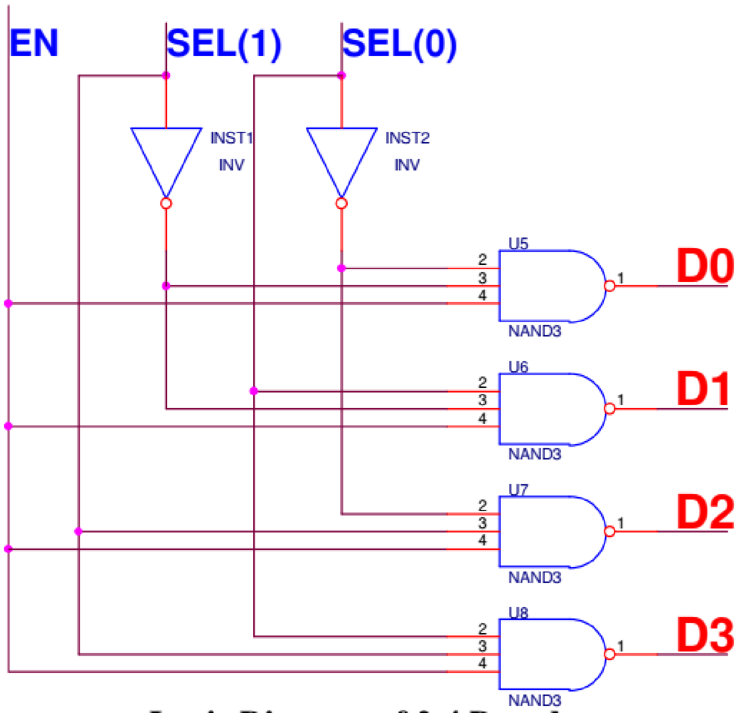
Code

```
library ieee;
use ieee.std_logic_1164.all;
entity mux41 is
        port(i: in std_logic_vector(3 downto 0);
                        sel: in std_logic_vector(1 downto 0);
                        y: out std_logic);
end mux41;

architecture structural of mux41 is
        signal n_s: std_logic_vector(1 downto 0);
        signal x:  std_logic_vector(3 downto 0);
        component and_gate
                port (a,b,c: in std_logic;
                        y:out std_logic);
        end component;

        component or_gate
                port (a,b,c,d: in std_logic;
                        y:out std_logic);
        end component;

        component not_gate
                port (a: in std_logic;
                        y:out std_logic);
        end component;
        begin
        g1: not_gate port map(a=>sel(0), y=>n_s(0));
        g2: not_gate port map(a=>sel(1), y=>n_s(1));

        g3: and_gate port map(a=>sel(1),b=>sel(0),c=>i(3),y=>x(3));
        g4: and_gate port map(a=>sel(1),b=>n_s(0),c=>i(2),y=>x(2));
        g5: and_gate port map(a=>n_s(1),b=>sel(0),c=>i(1),y=>x(1));
        g6: and_gate port map(a=>n_s(1),b=>n_s(0),c=>i(0),y=>x(0));

        g7: or_gate port map(a=>x(3),b=>x(2),c=>x(1),d=>x(0),y=>y);
        end structural;

library ieee;
use ieee.std_logic_1164.all;
entity and_gate is
        port (a,b,c: in std_logic;
                        y:out std_logic);
end and_gate;

architecture dataflow of and_gate is
        begin
                y <= a and b and c;
end dataflow;

library ieee;
use ieee.std_logic_1164.all;
entity or_gate is
        port (a,b,c,d: in std_logic;
                        y:out std_logic);
end or_gate;

architecture dataflow of or_gate is
        begin
                y <= a or b or c or d;
end dataflow;

library ieee;
use ieee.std_logic_1164.all;
entity not_gate is
        port (a: in std_logic;
                        y:out std_logic);
end not_gate;

architecture dataflow of not_gate is
        begin
                y <= not a;
end dataflow;
```

II.6 EXPERIMENT 6 : WRITE VHDL CODE TO IMPLEMENT 2 to 4 DECODER CIRCUIT IN FPGA KIT:



Block Diagram of Decoder



Logic Diagram of 2:4 Decoder

EN	Inputs		Output
	Sel(1)	Sel(0)	D
1	X	X	0
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3

Truth table

Top module

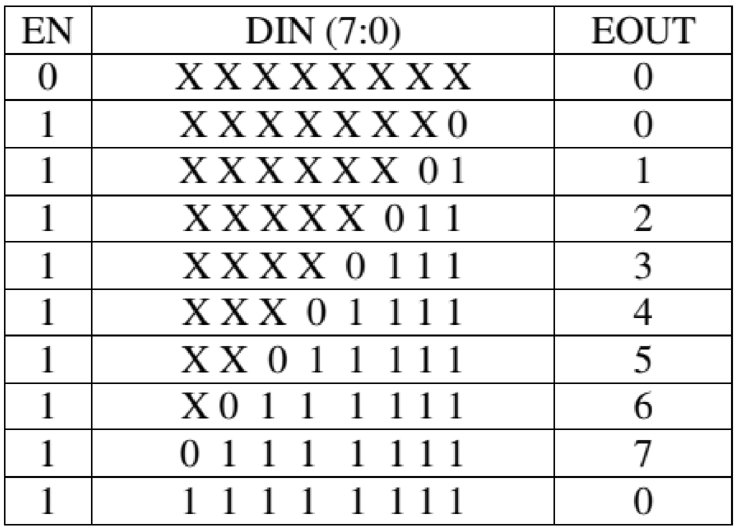
```
library ieee;
use ieee.std_logic_1164.all;
entity lab4_ex6 is
        port (SW        : in std_logic_vector(17 downto 0);
                LEDR   : out std_logic_vector(17 downto 0);
                LEDG  : out std_logic_vector(7 downto 0));
end lab4_ex6;

architecture structural of lab4_ex6 is
        component dec24
                port(en: in std_logic;
                        sel: in std_logic_vector(1 downto 0);
                        d: out std_logic_vector(3 downto 0));
        end component;

        begin
                LEDR <= SW;
                DUT: dec24 port map(SW(2),SW(1 downto 0),LEDG(7 downto 4));
        end structural;
```

Code

```
library ieee;
use ieee.std_logic_1164.all;
entity dec24 is
   port(en: in std_logic;
        sel: in std_logic_vector(1 downto 0);
        d: out std_logic_vector(3 downto 0));
end dec24;

architecture behavioral of dec24 is
begin
   process(en, sel)
   begin
       if (en = '0') then
           d <= "0000";
       else
           case sel is
               when "00" => d <= "0001";
               when "01" => d <= "0010";
               when "10" => d <= "0100";
               when "11" => d <= "1000";
               when others => d <= "0000";
           end case;
       end if;
   end process;
end behavioral;
```

II.7 EXPERIMENT 7 : WRITE VHDL CODE TO IMPLEMENT 8 TO 3 ENCODER WITH PRIORITY CIRCUIT IN FPGA KIT:

Truth table for 8-input priority encoder



EN	DIN (7:0)	EOUT
0	X X X X X X X X	0
1	X X X X X X X 0	0
1	X X X X X X 0 1	1
1	X X X X X 0 1 1	2
1	X X X X 0 1 1 1	3
1	X X X 0 1 1 1 1	4
1	X X 0 1 1 1 1 1	5
1	X 0 1 1 1 1 1 1	6
1	0 1 1 1 1 1 1 1	7
1	1 1 1 1 1 1 1 1	0

Top module

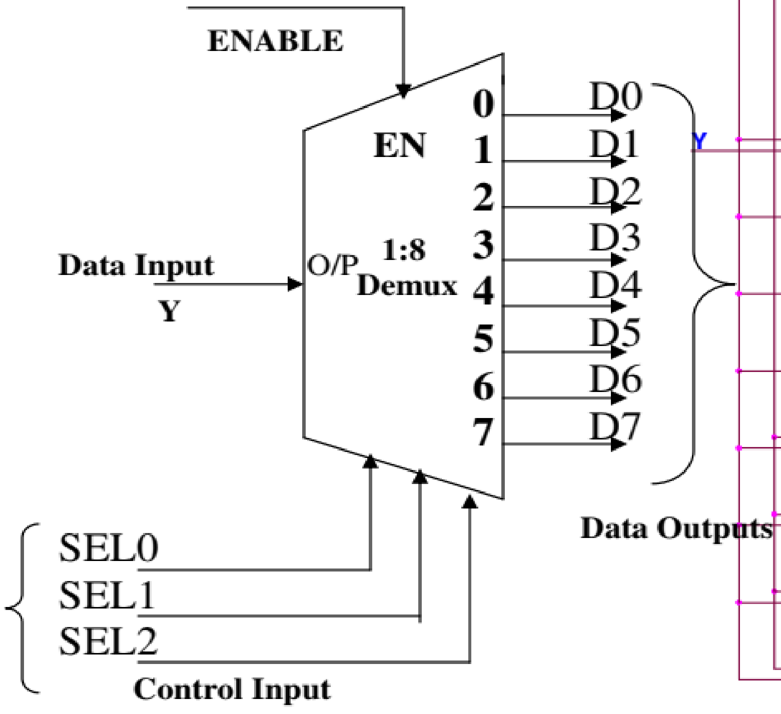
```
library ieee;
use ieee.std_logic_1164.all;
```

```
entity lab4_ex7 is
        port (SW        : in std_logic_vector(17 downto 0);
                LEDR   : out std_logic_vector(17 downto 0);
                LEDG  : out std_logic_vector(7 downto 0));
end lab4_ex7;

architecture structural of lab4_ex7 is
        component enc83
                port(en: in std_logic;
                        d: in std_logic_vector(7 downto 0);
                        y: out std_logic_vector(2 downto 0));
        end component;

        begin
                LEDR <= SW;
                DUT: enc83 port map(SW(8),SW(7 downto 0),LEDG(7 downto 5));
        end structural;
```

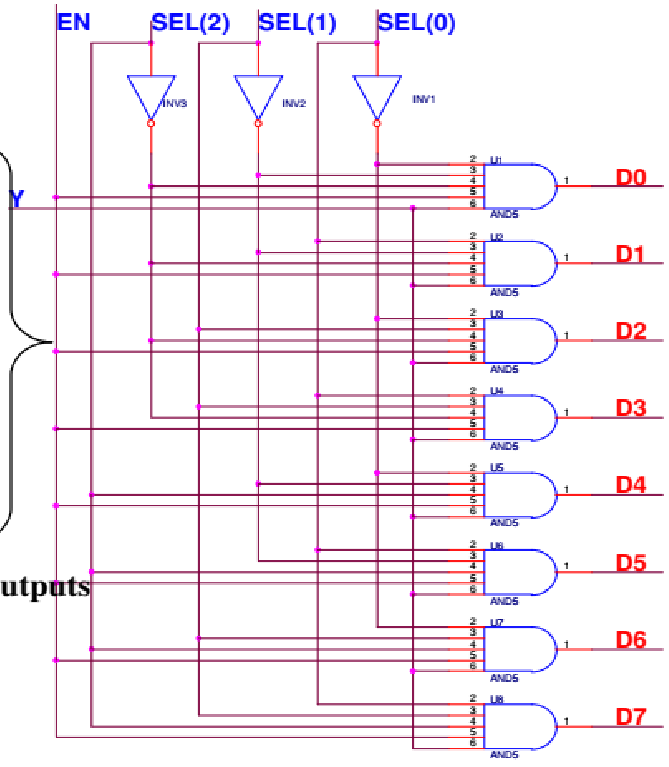
Code

```
library ieee;
use ieee.std_logic_1164.all;
entity enc83 is
        port(en: in std_logic;
                d: in std_logic_vector(7 downto 0);
                y: out std_logic_vector(2 downto 0));
end enc83;


architecture behavioral of enc83 is
begin
  process(en, d)
  begin
     if (en = '0') then
        y <= "000";
     else
        case d is
                                when "00000001" => y <= "000";
                                when "00000011" => y <= "001";
                                when "00000111" => y <= "010";
                                when "00001111" => y <= "011";
                                when "00011111" => y <= "100";
                                when "00111111" => y <= "101";
                                when "01111111" => y <= "110";
                                when "11111111" => y <= "111";
                                when others => y <= "000";
                        end case;
     end if;
  end process;
end behavioral;
```

II.8 EXPERIMENT 8 : WRITE VHDL CODES TO IMPLEMENT 1 TO 8 DEMULTIPLEXER CIRCUIT IN FPGA KIT:



Block Diagram of 1:8 Demux



Logic Diagram

Truth Table

EN	CONTROL INPUTS			OUTPUTS
	SEL(3)	SEL(3)	SEL(3)	
0	X	X	X	0
1	0	0	0	D0=Y
1	0	0	1	D1=Y
1	0	1	0	D2=Y
1	0	1	1	D3=Y
1	1	0	0	D4=Y
1	1	0	1	D5=Y
1	1	1	0	D6=Y
1	1	1	1	D7=Y

Top module

```
library ieee;
use ieee.std_logic_1164.all;
entity lab4_ex8 is
        port (SW        : in std_logic_vector(17 downto 0);
                LEDR    : out std_logic_vector(17 downto 0);
                LEDG   : out std_logic_vector(7 downto 0));
end lab4_ex8;
```

```
architecture structural of lab4_ex8 is
        component demux18
                port(i: in std_logic;
                         en: in std_logic;
                         sel: in std_logic_vector(2 downto 0);
                         d: out std_logic_vector(7 downto 0));
        end component;

        begin
                LEDR <= SW;
                DUT: demux18 port map(SW(4),SW(3),SW(2 downto 0),LEDG(7 downto 0));
        end structural;
```

Code

```
library ieee;
use ieee.std_logic_1164.all;
entity demux18 is
        port(i: in std_logic;
                 en: in std_logic;
                 sel: in std_logic_vector(2 downto 0);
                 d: out std_logic_vector(7 downto 0));
end demux18;


architecture structural of demux18 is
        signal n_s: std_logic_vector(2 downto 0);
        component not_gate
                port(a: in std_logic;
                         y: out std_logic);
        end component;

        component and_gate
                port(a,b,c,d,e: in std_logic;
                         y:out std_logic );
        end component;

        begin
        g1: not_gate port map(a=>sel(0), y=>n_s(0));
        g2: not_gate port map(a=>sel(1), y=>n_s(1));
        g3: not_gate port map(a=>sel(2), y=>n_s(2));

        g4: and_gate port map(a=>n_s(0), b=>n_s(1), c=>n_s(2),d=>en,e=>i, y=>d(0));
        g5: and_gate port map(a=>sel(0), b=>n_s(1), c=>n_s(2),d=>en,e=>i, y=>d(1));
        g6: and_gate port map(a=>n_s(0), b=>sel(1), c=>n_s(2),d=>en,e=>i, y=>d(2));
        g7: and_gate port map(a=>sel(0), b=>sel(1), c=>n_s(2),d=>en,e=>i, y=>d(3));
        g8: and_gate port map(a=>n_s(0), b=>n_s(1), c=>sel(2),d=>en,e=>i, y=>d(4));
        g9: and_gate port map(a=>sel(0), b=>n_s(1), c=>sel(2),d=>en,e=>i, y=>d(5));
        g10: and_gate port map(a=>n_s(0), b=>sel(1), c=>sel(2),d=>en,e=>i, y=>d(6));
        g11: and_gate port map(a=>sel(0), b=>sel(1), c=>sel(2),d=>en,e=>i, y=>d(7));
        end structural;

library ieee;
use ieee.std_logic_1164.all;
entity and_gate is
```

```
port(     a: in std_logic;
              b: in std_logic;
              c: in std_logic;
              d: in std_logic;
              e: in std_logic;
              y: out std_logic
);
end and_gate;

architecture dataflow of and_gate is
begin
   y <= a and b and c and d and e;
end dataflow;

library ieee;
use ieee.std_logic_1164.all;
entity not_gate is
port(     a: in std_logic;
             y: out std_logic
);
end not_gate;

architecture dataflow of not_gate is
begin
   y <= not a;
end dataflow;
```

II.9 EXPERIMENT 9 : WRITE VHDL CODES TO IMPLEMENT N-BIT COMPARATOR CIRCUIT IN FPGA KIT WITH FOLLOWING VHDL REFERENED CODE

Top module

```
library ieee;
use ieee.std_logic_1164.all;
entity lab4_ex9 is
generic(n: natural :=4);
        port (SW          : in std_logic_vector(17 downto 0);
                  LEDR   : out std_logic_vector(17 downto 0);
                  LEDG  : out std_logic_vector(7 downto 0));
end lab4_ex9;

architecture structural of lab4_ex9 is
        component comparator
                  port(     A:        in std_logic_vector(n-1 downto 0);
                                      B:        in std_logic_vector(n-1 downto 0);
                                      less:                 out std_logic;
                                      equal:                out std_logic;
                                      greater:  out std_logic);
        end component;

        begin
                  LEDR <= SW;
                  DUT: comparator port map(SW(2*n-1 downto n),SW(n-1 downto 0),LEDG(7),LEDG(6),LEDG(5));
        end structural;
```

Code

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

entity comparator is
generic(n: natural :=4);
port(     A:        in std_logic_vector(n-1 downto 0);
          B:        in std_logic_vector(n-1 downto 0);
          less:              out std_logic;
          equal:             out std_logic;
          greater:  out std_logic
);
end comparator;
architecture behv of comparator is
begin
   process(A,B)
   begin
      if (A>B) then
             less <= '0';
             equal <= '0';
             greater <= '1';
          elsif (A=B) then
             less <= '0';
             equal <= '1';
             greater <= '0';
          else
             less <= '1';
             equal <= '0';
             greater <= '0';
          end if;
   end process;

end behv;
```

II.10 EXPERIMENT 10 : WRITE VHDL CODES TO IMPLEMENT N-BIT ALU CIRCUIT IN FPGA KIT WITH FOLLOWING VHDL REFERENED CODE

Top module

```
library ieee;
use ieee.std_logic_1164.all;
entity lab4_ex10 is
generic(N: natural :=4);
          port (SW          : in std_logic_vector(17 downto 0);
                    LEDR   : out std_logic_vector(17 downto 0);
                    LEDG  : out std_logic_vector(7 downto 0));
end lab4_ex10;

architecture structural of lab4_ex10 is
          component ALU
                    port(     A:        in std_logic_vector(N-1 downto 0);
                    B:        in std_logic_vector(N-1 downto 0);
                    Op:       in std_logic_vector(2 downto 0);
                    Res:      out std_logic_vector(N-1 downto 0));
          end component;
```

```
        begin
                LEDR <= SW;
                DUT: ALU
                                port map(
                                        SW(2*N-1 downto N),
                                        SW(N-1 downto 0),
                                        SW(2*N+2 downto 2*N),
                                        LEDG(N-1 downto 0));
        end structural;
```

Code

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

entity ALU is
generic(N: natural :=4);

port(   A:      in std_logic_vector(N-1 downto 0);
                B:      in std_logic_vector(N-1 downto 0);
                Op:     in std_logic_vector(2 downto 0);
                Res:    out std_logic_vector(N-1 downto 0));
end ALU;

architecture behv of ALU is
begin

  process(A,B,Op)
  begin

        case Op is
          when "000" =>
                 Res <= A + B;
          when "001" =>
              Res <= A - B;
       when "010" =>
                Res <= not A ;
          when "011" =>
                Res <= not (A and B);
          when "100" =>
                Res <= not (A or B);
          when "101" =>
                Res <= A and B;
          when "110" =>
                Res <= A or B;
          when "111" =>
                Res <= A xnor B;
      end case;

  end process;

end behv;
```