|  |
| --- |
| **DIGITAL SYSTEM DESIGN LABORATORY** |
| **LAB 4** |

**IMPLEMENTATION OF BASIC COMBINATION LOGIC CIRCUIT USING VHDL**

### I. LAB OBJECTIVES

### This Lab experiments are intended to implement Basic Combination Logic and Sequential Circuit in VHDL. Students are require to write test bench to simulate the given example code and Top level module to implement these codes in DE2-115 FPGA Kit.

### a) For each experiment write the VHDL Code using one of three method ( dataflow, behavior and gate level)

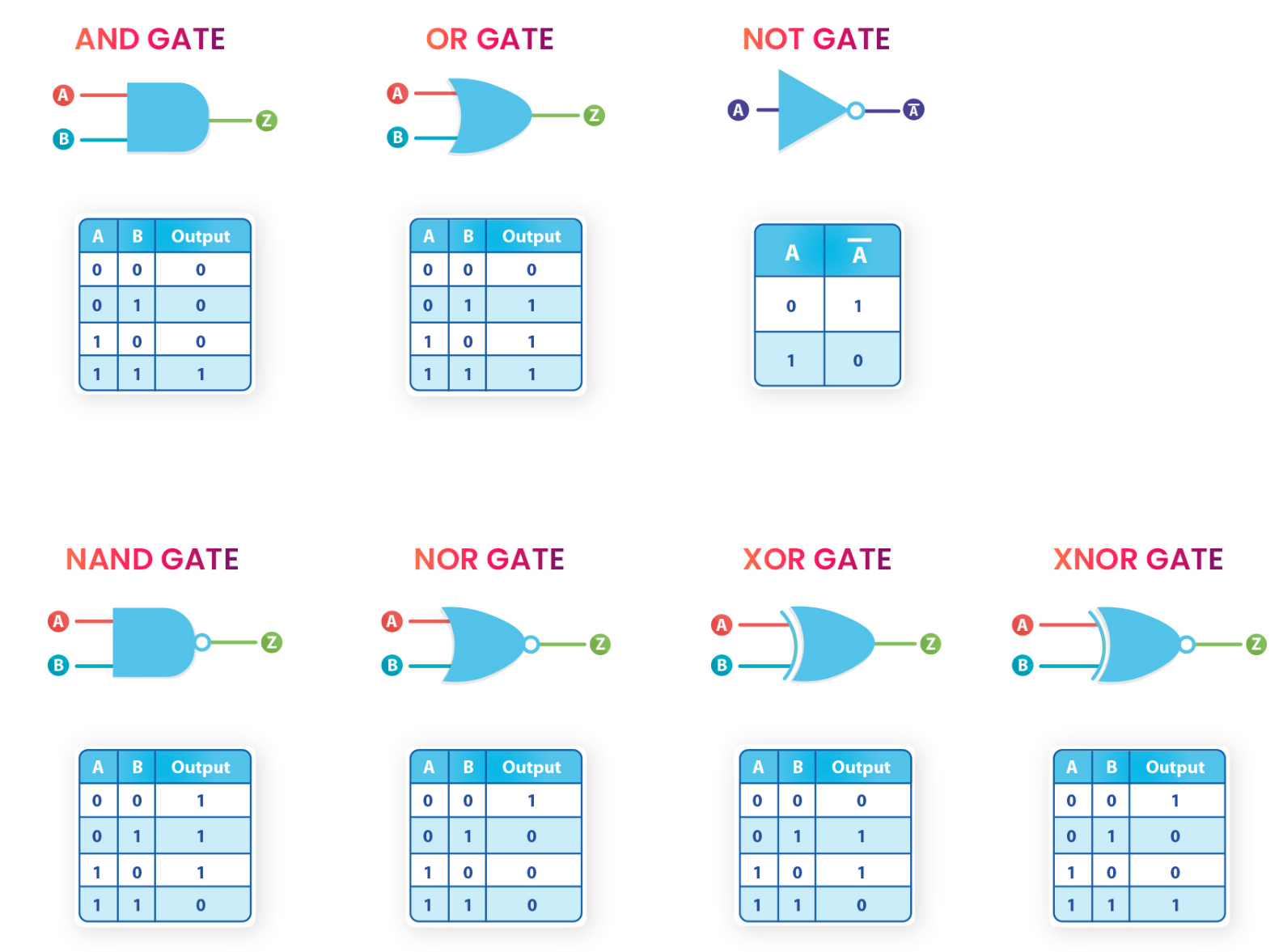
### b) Write the Top Level VHDL Code to implement the these modules in DE2-FPGA Kit

### (Show implementation results in Lab report)

### c) Analyze the FPGA implementation results for these model

### II. PROCEDURE

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



|  |  |  |  |
| --- | --- | --- | --- |
| **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;  use IEEE.STD\_LOGIC\_1164.ALL;  entity all\_gates is  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 all\_gates;  architecture structural of all\_gates is  -- Declare components for each gate  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 NOT\_GATE  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; | library ieee;  use ieee.std\_logic\_1164.all;  entity all\_gates is  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 all\_gates;  architecture dataflow of all\_gates is  begin  Y\_AND <= A and B;  Y\_OR<= A or B;  Y\_NOT\_A <= not A;  Y\_NAND <= A nand B;  Y\_NOR <= A nor B;  Y\_XOR <= A xor B;  Y\_XNOR <= A xnor B;  end dataflow; | library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  entity all\_gates is  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 all\_gates;  architecture behavioral of all\_gates is  begin  process(A, B)  begin  Y\_AND <= A and B;  Y\_OR <= A or B;  Y\_NOT\_A <= not A;  Y\_NAND <= A nand B;  Y\_NOR <= A nor B;  Y\_XOR <= A xor B;  Y\_XNOR <= A xnor B;  end process;  end behavioral; |
|  | | | |

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

A black line drawing of a circuit

Description automatically generated A table with numbers and symbols

Description automatically generated

|  |  |  |
| --- | --- | --- |
| **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** | **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 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;  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; | 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; | 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; |
|  | | |

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

**A diagram of a circuit

Description automatically generated A table of input output

Description automatically generated**

|  |  |  |
| --- | --- | --- |
| **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;  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 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; | 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; | 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 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; |
|  | | |

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

A diagram of a network

Description automatically generated A square with black text

Description automatically generated with medium confidence

|  |
| --- |
| **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:**

A diagram of a circuit

Description automatically generated A diagram of a diagram and a diagram of a diagram

Description automatically generated

|  |
| --- |
| **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:**

A diagram of a decoder

Description automatically generated

|  |
| --- |
| **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:**

A table with numbers and symbols

Description automatically generated

|  |
| --- |
| **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:**

A diagram of a block diagram

Description automatically generated

A table with numbers and symbols

Description automatically generated

|  |
| --- |
| **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; |
|  |

**IV. LAB REPORT GUIDELINES**

Students write up a report on the Verilog HDL implementation experiment projects created in this lab. The lab report should include Circuit Schematics, Truth Table, Verilog Module Codes, Verilog test bench codes, Top level module to implement the required circuit in FPGA KIT and evidences of data output evidences to validate the experiments (The Captured Screens, Photo of FPGA Kit implementation results).