|  |
| --- |
| **DIGITAL SYSTEM DESIGN LABORATORY** |
| **LAB 3** |

**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 in three method ( dataflow, behavior and gate level)

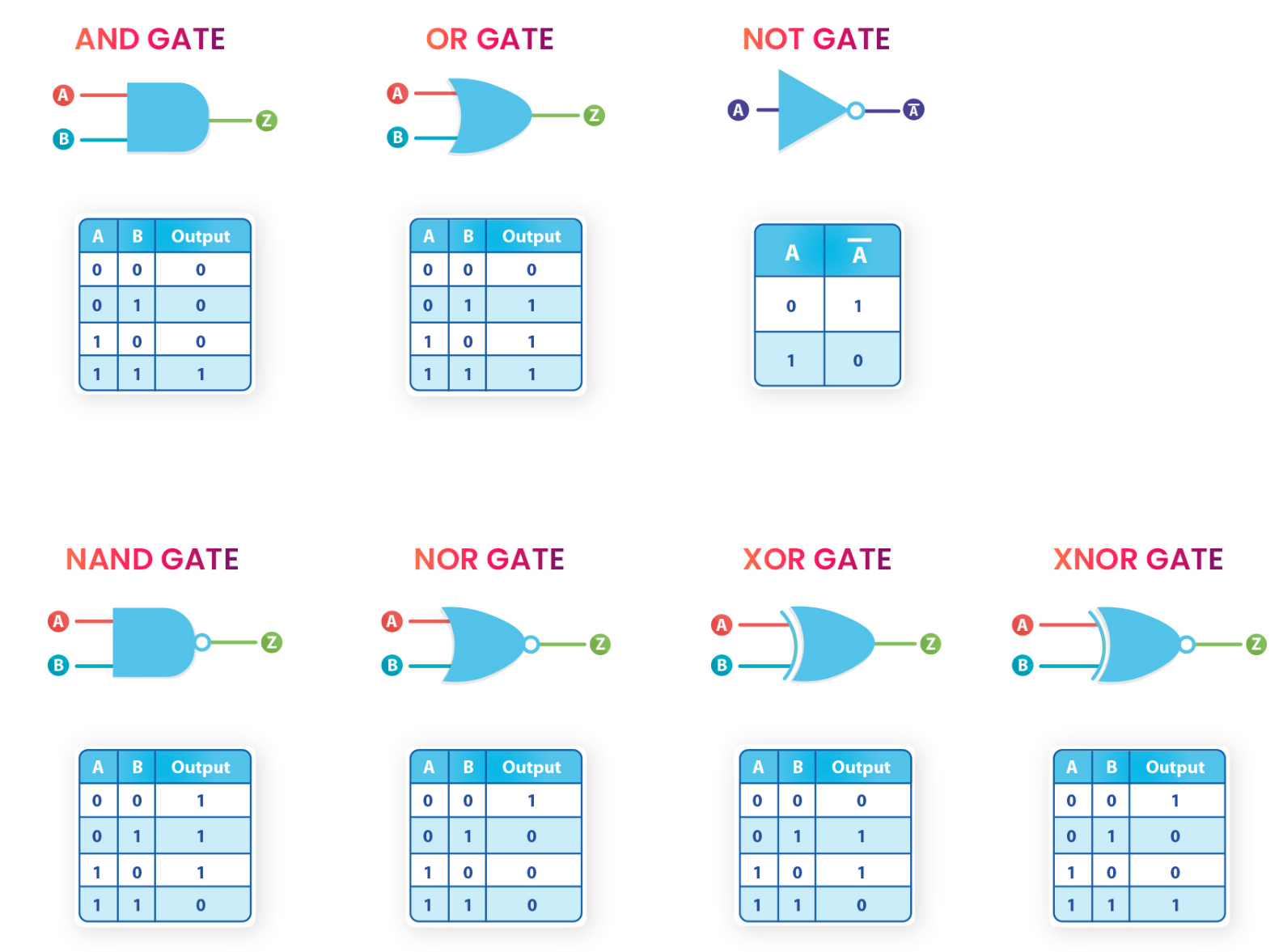
### b) For each model, write the Top Level VHDL Code to implement the these modules in DE2-FPGA Kit

### (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.



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY and\_test IS

PORT (SW: IN STD\_LOGIC\_VECTOR(15 DOWNTO 0);

LEDR: OUT STD\_LOGIC\_VECTOR(15 DOWNTO 0); -- red LEDs

LEDG: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0)); -- Green LEDs

END and\_test;

ARCHITECTURE Structure OF and\_test IS

COMPONENT my\_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: my\_gates PORT MAP (SW(1),SW(0),LEDG(6),LEDG(5),LEDG(4),LEDG(3),LEDG(2),LEDG(1),LEDG(0));

END Structure;

library ieee;

use ieee.std\_logic\_1164.all;

entity my\_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 my\_gates;

architecture dataflow of my\_gates is

begin

Y\_AND <= A and B;

Y\_OR <= A or B;

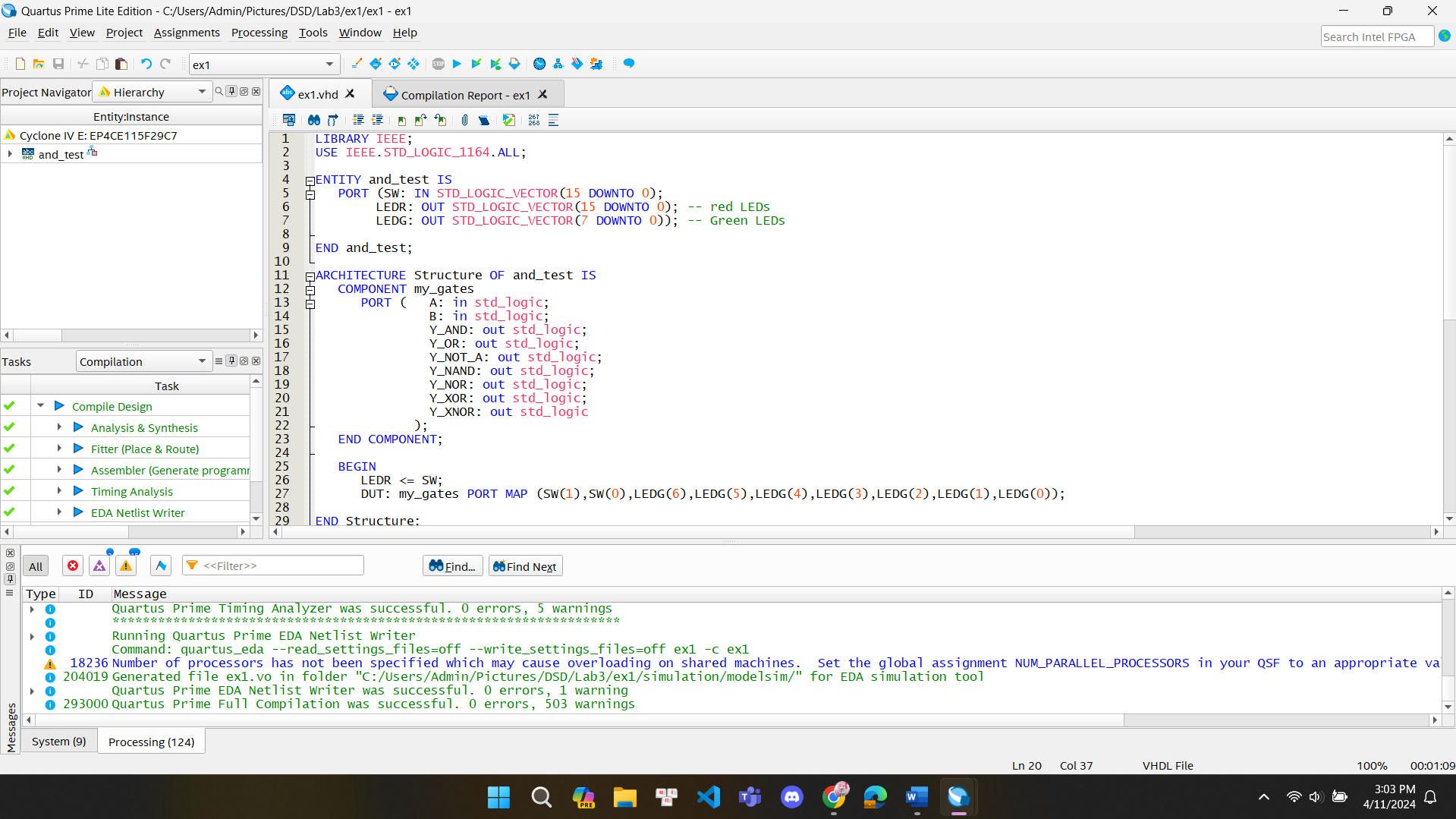
Y\_NOT\_A <= not A;

Y\_NAND <= A nand B;

Y\_NOR <= A nor B;

Y\_XOR <= A xor B;

Y\_XNOR <= A xnor B;

end dataflow; 

**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

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY ex2 IS

PORT (SW: IN STD\_LOGIC\_VECTOR(15 DOWNTO 0);

LEDR: OUT STD\_LOGIC\_VECTOR(15 DOWNTO 0); -- red LEDs

LEDG: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0)); -- Green LEDs

END ex2;

ARCHITECTURE Structure OF ex2 IS

COMPONENT test

PORT ( A: in std\_logic;

B: in std\_logic;

SUM: out std\_logic;

Carry: out std\_logic

);

END COMPONENT;

BEGIN

LEDR <= SW;

DUT: test PORT MAP (SW(1),SW(0),LEDG(1),LEDG(0));

END Structure;

library ieee;

use ieee.std\_logic\_1164.all;

entity test is

port( A: in std\_logic;

B: in std\_logic;

SUM: out std\_logic;

Carry: out std\_logic

);

end test;

architecture dataflow of test is

begin

SUM <= A xor B;

Carry <= A and B;

end dataflow;

A screenshot of a computer

Description automatically generated

**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**

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY ex3 IS

PORT (SW: IN STD\_LOGIC\_VECTOR(15 DOWNTO 0);

LEDR: OUT STD\_LOGIC\_VECTOR(15 DOWNTO 0); -- red LEDs

LEDG: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0)); -- Green LEDs

END ex3;

ARCHITECTURE Structure OF ex3 IS

COMPONENT test

PORT ( A: in std\_logic;

B: in std\_logic;

C: in std\_logic;

SUM: out std\_logic;

Carry: out std\_logic

);

END COMPONENT;

BEGIN

LEDR <= SW;

DUT: test PORT MAP (SW(2),SW(1),SW(0),LEDG(1),LEDG(0));

END Structure;

library ieee;

use ieee.std\_logic\_1164.all;

entity test is

port( A: in std\_logic;

B: in std\_logic;

C: in std\_logic;

SUM: out std\_logic;

Carry: out std\_logic

);

end test;

architecture dataflow of test is

begin

SUM <= (A xor B) xor C;

Carry <= ((A xor B) and C) or (A and B);

end dataflow;

**Structural style**

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY ex1 IS

PORT (SW: IN STD\_LOGIC\_VECTOR(15 DOWNTO 0);

LEDR: OUT STD\_LOGIC\_VECTOR(15 DOWNTO 0); -- red LEDs

LEDG: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0) -- Green LEDs

);

END ex1;

ARCHITECTURE Structure OF ex1 IS

COMPONENT Full\_adder

PORT ( A: in std\_logic;

B: in std\_logic;

Cin: in std\_logic;

S: out std\_logic;

C: out std\_logic);

END COMPONENT;

BEGIN

LEDR <= SW;

DUT: Full\_adder PORT MAP (SW(2),SW(1),SW(0),LEDG(1),LEDG(0));

END Structure;

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity Full\_adder is

port(A,B,Cin : in STD\_LOGIC;

S, C:out STD\_LOGIC);

end Full\_adder;

architecture struct\_Full of Full\_adder is

signal X,Y,Z :std\_logic;

component my\_AND\_gate

port( A: in std\_logic;

B: in std\_logic;

Y: out std\_logic

);

end component;

component my\_OR\_gate

port( A: in std\_logic;

B: in std\_logic;

Y: out std\_logic

);

end component;

component my\_XOR\_gate

port( A: in std\_logic;

B: in std\_logic;

Y: out std\_logic

);

end component;

begin

X1: my\_XOR\_gate port map(A,B,X);

X2: my\_XOR\_gate port map(X,Cin,S);

A1: my\_AND\_gate port map(X,Cin,Y);

A2: my\_AND\_gate port map(A,B,Z);

A3: my\_OR\_gate port map(Y,Z,C);

end struct\_Full;

library ieee;

use ieee.std\_logic\_1164.all;

entity my\_AND\_gate is

port( A: in std\_logic;

B: in std\_logic;

Y: out std\_logic

);

end my\_AND\_gate;

architecture df\_and of my\_AND\_gate is

begin

Y <= A and B;

end df\_and;

library ieee;

use ieee.std\_logic\_1164.all;

entity my\_OR\_gate is

port( A: in std\_logic;

B: in std\_logic;

Y: out std\_logic

);

end my\_OR\_gate;

architecture df\_or of my\_OR\_gate is

begin

Y <= A or B;

end df\_or;

library ieee;

use ieee.std\_logic\_1164.all;

entity my\_XOR\_gate is

port( A: in std\_logic;

B: in std\_logic;

Y: out std\_logic

);

end my\_XOR\_gate;

architecture df\_xor of my\_XOR\_gate is

begin

Y <= A xor B;

end df\_xor;

**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

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY ex4 IS

PORT (SW: IN STD\_LOGIC\_VECTOR(15 DOWNTO 0);

LEDR: OUT STD\_LOGIC\_VECTOR(15 DOWNTO 0); -- red LEDs

LEDG: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0) -- Green LEDs

);

END ex4;

ARCHITECTURE Structure OF ex4 IS

COMPONENT my\_gates

PORT (I0: IN STD\_LOGIC;

I1: IN STD\_LOGIC;

Sel: IN STD\_LOGIC;

Y: OUT STD\_LOGIC

);

END COMPONENT;

BEGIN

LEDR <= SW;

DUT: my\_gates PORT MAP (SW(0),SW(1),SW(2),LEDG(1));

END Structure;

library ieee;

use ieee.std\_logic\_1164.all;

ENTITY my\_gates IS

PORT (I0: IN STD\_LOGIC;

I1: IN STD\_LOGIC;

Sel: IN STD\_LOGIC;

Y: OUT STD\_LOGIC

);

END my\_gates;

ARCHITECTURE dataflow OF my\_gates IS

BEGIN

Y<=(((NOT Sel) AND I0) OR (I1 AND Sel));

END dataflow;

A screenshot of a computer

Description automatically generated

**STRUCTURAl  
library ieee;**

**use ieee.std\_logic\_1164.all;**

**entity my\_gates is**

**port (**

**I0 : in std\_logic;**

**I1 : in std\_logic;**

**Sel : in std\_logic;**

**Y : out std\_logic**

**);**

**end my\_gates;**

**architecture dataflow of my\_gates is**

**begin**

**Y <= (((not Sel) and I0) or (I1 and Sel));**

**end dataflow;**

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**entity ex4 is**

**port (**

**SW : in std\_logic\_vector(15 downto 0);**

**LEDR : out std\_logic\_vector(15 downto 0); -- red LEDs**

**LEDG : out std\_logic\_vector(7 downto 0) -- Green LEDs**

**);**

**end ex4;**

**architecture Structure of ex4 is**

**component my\_gates**

**port (**

**I0 : in std\_logic;**

**I1 : in std\_logic;**

**Sel : in std\_logic;**

**Y : out std\_logic**

**);**

**end component;**

**begin**

**LEDR <= SW;**

**DUT : my\_gates port map (I0 => SW(0), I1 => SW(1), Sel => SW(2), Y => LEDG(1));**

**end Structure;  
A screenshot of a computer

Description automatically generated**

**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

**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

**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

**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

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

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 exor 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).