UTKAL GOURAV MADHUSUDAN INSTITUTE OF TECHNOLOGY, RAYAGADA | Govt. of Odisha

ଉତ୍କଳ ଗୌରବ ମଧୁସୂଦନ ବୈଷୟିକ ଅନୁଷ୍ଠାନ, ରାୟଗଡ

VLSI LAB MANUAL

5TH SEMESTER E&TC ENGG.

DEPTT. OF ELECTRONICS & TELECOMMUNICATION ENGG.

U.G.M.I.T.

RAYAGADA

VISION OF THE INSTITUTE

Our vision is to develop highly skilled, well educated society where all access their potential and compatible to social and economical prosperity of the country.

MISSION OF THE INSTITUTE

To provide high quality education, develop technical skills and innovative capabilities so as to enable the product to serve the society better with ethical values and tomorrow's workforce.

VISION OF THE DEPARTMENT(Department of ETC)

To establish Electronics and Telecommunication Engineering Department as the center of excellence in Education, research, technology, producing skilled and ethical Electronics and Telecommunication Engineers who can meet the needs of current technological advancements and can adapt to the accelerating changes at state and national level.

MISSION OF THE DEPARTMENT(Department of ETC)

Mission No	Mission Statement
M1	To offer quality education through innovative teaching methods and practical orientations to prepare the students for areal-time design and development so as to pursue a successful career.
M2	To produce diploma graduates with technical expertise ,professional attitude and ethical values.
M3	To provide the best learning environment to the students, faculty and staff members conducive to create excellence in technical education.
M4	To encourage students to pursue higher studies, to appear various competitive examinations and other career enhancement courses.
M5	To improve department-industries collaboration through various internship and training programs .

LIST OF EXPERIMENTS

S.NO.	NAME OF EXPERIMENTS	PAGE NO.
1	Develop a VHDL test bench code for testing and implement addition, subtraction, multiplication and division on FPGA kit.	
2	Write a VHDL program for Buzzer Interface.	
3	Develop a VHDL test bench code for testing 7 segment LED display.	
4	Develop a VHDL test bench code for testing 4-bit binary counter.	
5	Develop a VHDL test bench code for testing simple gates.	
6	Develop a VHDL test bench code & implement of FPGA kit for Multiplexer and De-Multiplexer.	
7	Develop a VHDL test bench & implement of FPGA kit for Encoder and Decoder.	
8	Design and implementation of Half Adder and Full Adder.	
9	Design and implementation of D flip flop.	
10	Write a simple program with two separate LED blinking tasks.	

EXPERIMENT-1

AIM OF THE EXPERIMENT-

Develop a VHDL test bench code for testing and implement addition, subtraction, multiplication and division on FPGA kit.

EQUIPMENT REQUIRED:

1. PC
2. XILINX ISE software
3. FPGA / CPLD training Kit.

PROCEDURE:

· New project and type the project name and check the top level source type as HDL

·Enter the device properties and click Next

· Click New Source and Select the Verilog Module and then give the file name

·Give the Input and Output port names and click finish.

·Type the Verilog program and save it

·Double click the synthesize XST and check syntax

·Simulate the waveform by behavioral simulation

· For implementation Select User constraints and give input and output port pin number

·Click Implement design for Translate, map and place & route

·Generate .bit file using programming file

·Implement in FPGA through parallel-JTAG cable

·Check the behavior of design in FPGA by giving inputs

PROGRAM:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Arithmetic_operation is
   Port ( NUM1 : in STD_LOGIC_VECTOR (4 downto 0) := "00000";
        NUM2 : in  STD_LOGIC_VECTOR (4 downto 0) := "00000";
        SUM, SUB, MUL, DIV : out  STD_LOGIC_VECTOR (4 downto 0));
end Arithmetic_operation ;
architecture Behavioral of Arithmetic_operation is
begin
   SUM <= NUM1 + NUM2;
```

```
    SUB <= NUM1 – NUM2;
    MUL <= NUM1 * NUM2;
    DIV <= NUM1 / NUM2;
end Behavioral;
```

CONCLUSION:

EXPERIMENT-2

AIM OF THE EXPERIMENT

Write a VHDL program for Buzzer Interface.

EQUIPMENT REQUIRED:

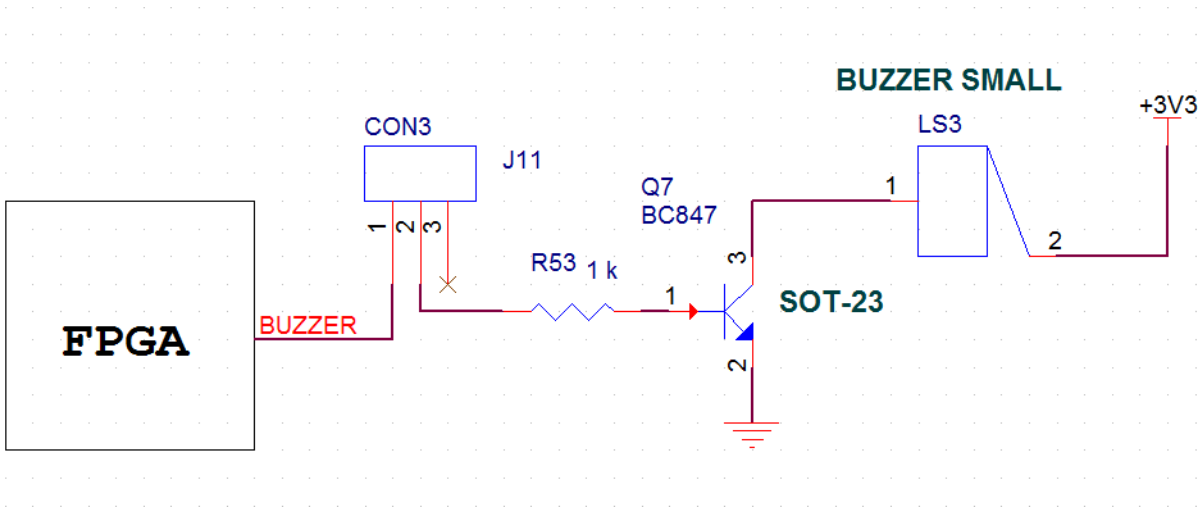
·PC
·XILINX
· FPGA / CPLD training Kit.

THEORY: Buzzer

Piezo buzzer is an electric component that comes in different shapes and sizes, which can be used to create sound waves when provided with electrical signal. piezo buzzer requires a square wave to produce a tone.

Interfacing Piezo buzzer with FPGA Development Kit

The FPGA Development Kit has Piezo buzzer, indicated as in Figure. Buzzer is driven by transistor Q1. FPGA can create sound by generating a PWM(Pulse Width Modulated) signal – a square wave signal, which is nothing more than a sequence of logic zeros and ones. Frequency of the square signal determines the pitch of the generated sound. To enable buzzer, place jumper JP at E label mark position and to disable buzzer place jumper JP at D Position.



ALGORITHM:

· New project and type the project name and check the top level source type as HDL

·Enter the device properties and click Next

· Click New Source and Select the Verilog Module and then give the file name

·Give the Input and Output port names and click finish.

·Type the Verilog program and save it

·Double click the synthesize XST and check syntax

·Simulate the waveform by behavioral simulation

· For implementation Select User constraints and give input and output port pin number

·Click Implement design for Translate, map and place & route

·Generate .bit file using programming file

·Implement in FPGA through parallel-JTAG cable

·Check the behavior of design in FPGA by giving inputs

VHDL Code Description:

The following VHDL Code demonstrates the functionality of piezo buzzer. PWM pulse is applied with 2s duty cycle. Buzzer produce beeps sound every 1 sec.

VHDL CODE-

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity buzz is

port ( clock : in std_logic;

    a    : out std_logic

         );

end buzz;

architecture Behavioral of buzz is

begin

process(clock)

variable i : integer := 0;

begin

if clock'event and clock = '1' then

if i <= 50000000 then

i := i + 1;

a <= '1';
```

```
elsif i > 50000000 and i < 100000000 then

i := i + 1;

a <= '0';

elsif i = 100000000 then

i := 0;

end if;

end if;

end process;

end Behavioral;
```

User Constraint File

```
NET "clock" LOC = "p185" ;
NET "a" LOC = "p116" ;
```

CONCLUSION-

Thus the vhdl code for buzzer was simulated successfully.

EXPERIMENT-3

AIM OF THE EXPERIMENT-

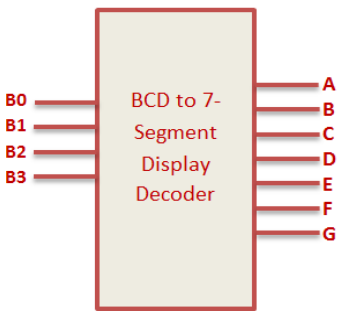
Develop a VHDL test bench code for testing 7 segment LED display.

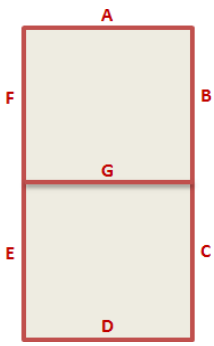
EQUIPMENT REQUIRED:

1. PC
2. XILINX ISE software
3. FPGA kit

THEORY-

Here is a program for BCD to 7-segment display decoder. The module takes 4 bit BCD as input and outputs 7 bit decoded output for driving the display unit. A seven segment display can be used to display decimal digits. They have LED or LCD elements which becomes active when the input is zero. The figure shows how different digits are displayed:





BCD to 7 segment display Decoder Truth Table:

B3 B2 B1 B0	A B C D E F G
0000	0000001
0001	1001111
0010	0010010
0011	0000110
0100	1001100
0101	0100100
0110	0100000
0111	0001111
1000	0000000
1001	0000100

PROCEDURE:

· New project and type the project name and check the top level source type as HDL.

·Enter the device properties and click Next.

· Click New Source and Select the Verilog Module and then give the file name.

·Give the Input and Output port names and click finish.

·Type the program and save it.

·Double click the synthesize XST and check syntax.

·Simulate the waveform by behavioral simulation.

PROGRAM:

VHDL PROGRAM:

```
library IEEE;
use
IEEE.STD_LOGIC_1164.AL
L; use
IEEE.STD_LOGIC_ARITH.A
LL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity
SEVEN
isport (
    clk : in std_logic;
     bcd : in std_logic_vector(3 downto 0);  --BCD input
     segment7 : out std_logic_vector(6 downto 0) -- 7 bit decoded output.
  );
end SEVEN;
--'a' corresponds to MSB of segment7 and g corresponds to LSB of segment7.
architecture Behavioral of SEVEN is
```

```
begin
process
(clk,bcd)
BEGIN
if (clk'event and
clk='1') thencase  bcd
is
when "0000"=> segment7 <="0000001";  -- '1'
when "0001"=> segment7 <="1001111";  -- '1'
when "0010"=> segment7 <="0010010";  -- '2'
when "0011"=> segment7 <="0000110";  -- '3'
when "0100"=> segment7 <="1001100";  -- '4'
when "0101"=> segment7 <="0100100";  -- '5'
when "0110"=> segment7 <="0100000";  -- '6'
when "0111"=> segment7 <="0001111";  -- '7'
when "1000"=> segment7 <="0000000";  -- '8'
when "1001"=> segment7 <="0000100";  -- '9'
 --nothing is displayed when a number more than 9 is given as
input.when others=> segment7 <="1111111";
e
n
d
c
a
s
e
;
e
n
d
if
;


end

process

; end

Behavio

ral;
```

TEST_BENCH PROGRAM:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

ENTITY
SEVEN_TB IS
END
SEVEN_TB;
```

```
ARCHITECTURE behavior OF SEVEN_TB IS
  signal clk : std_logic := '0';
  signal bcd : std_logic_vector(3 downto 0) := (others
  => '0');signal segment7 : std_logic_vector(6 downto
  0);
  constant clk_period : time
:= 1 ns;BEGIN
  uut: entity work.SEVEN PORT MAP
  (clk,bcd,segment7);clk_process :process
  begin
           clk <= '0';
          wait for
           clk_period/2;
           clk <= '1';
           wait for
  clk_period/2;end
  process;
  stim_proc:
  process
  begin
   for i in 0 to 9 loop
       bcd <=
      conv_std_logic_vector(i,4);
      wait for 2 ns;
   end
  loop
  ; end
  proc
  ess;

END;
```

CONCLUSION:

EXPERIMENT - 4

AIM OF THE EXPERIMENT-

Develop a VHDL test bench code for testing 4-bit binary counter.

EQUIPMENT REQUIRED:

1. PC
2. XILINX ISE software

THEORY-

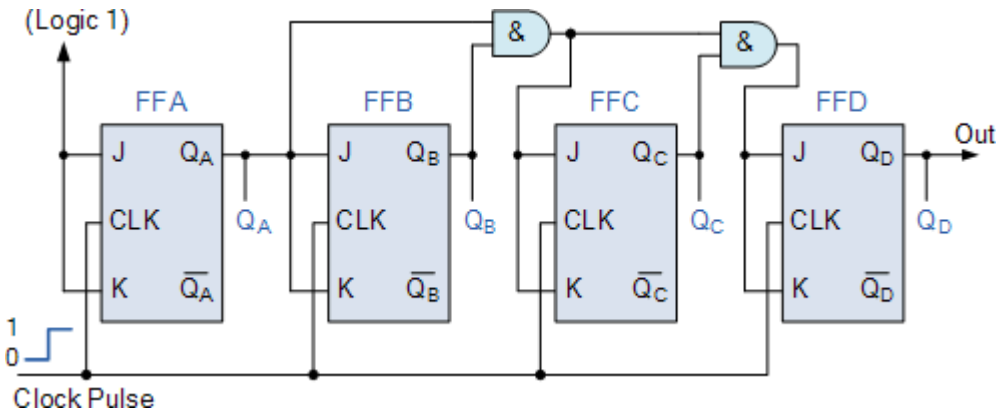
The external clock pulses (pulses to be counted) are fed directly to each of the J-K flipflop in the counter chain and that both the J and K inputs are all tied together in toggle mode, but only in the first flip-flop, flip-flop FFA (LSB) are they connected HIGH, logic "1" allowing the flip-flop to toggle on every clock pulse. Then the synchronous

counter follows a predetermined sequence of states in response to the common clock signal, advancing one state for each pulse.

The J and K inputs of flip-flop FFB are connected directly to the output Q_A of flip-flop FFA, but the J and K inputs of flip-flops FFC and FFD are driven from separate AND gates which are also supplied with signals from the input and output of the previous stage. These additional AND gates generate the required logic for the JK inputs of the next stage.

If we enable each JK flip-flop to toggle based on whether or not all preceding flip-flop outputs (Q) are "HIGH" we can obtain the same counting sequence as with the asynchronous circuit but without the ripple effect, since each flip-flop in this circuit will be clocked at exactly the same time.

Then as there is no inherent propagation delay in synchronous counters, because all the counter stages are triggered in parallel at the same time, the maximum operating frequency of this type of frequency counter is much higher than that for a similar asynchronous counter circuit.



PROCEDURE:

· New project and type the project name and check the top level source type as HDL.

·Enter the device properties and click Next.

· Click New Source and Select the Verilog Module and then give the file name.

·Give the Input and Output port names and click finish.

·Type the program and save it.

·Double click the synthesize XST and check syntax.

·Simulate the waveform by behavioral simulation.

PROGRAM:

VHDL PROGRAM:

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




entity UP_COUNTER is
   Port ( clk: in std_logic; -- clock input
        reset: in std_logic; -- reset input
        counter: out std_logic_vector(3 downto 0) -- output 4-bit counter
    );
end UP_COUNTER;

architecture Behavioral of UP_COUNTER is
signal counter_up: std_logic_vector(3 downto 0);
begin
-- up counter
process(clk)
begin
if(rising_edge(clk)) then
   if(reset='1') then
      counter_up <= x"0";
   else
      counter_up <= counter_up + x"1";
   end if;
 end if;
end process;
 counter <= counter_up;

end Behavioral;
```

TEST BENCH PROGRAM:

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

```
architecture Behavioral of tb_counters is

component UP_COUNTER
   Port ( clk: in std_logic; -- clock input
          reset: in std_logic; -- reset input
          counter: out std_logic_vector(3 downto 0) -- output 4-bit counter
    );
end component;
signal reset,clk: std_logic;
signal counter:std_logic_vector(3 downto 0);

begin
dut: UP_COUNTER port map (clk => clk, reset=>reset, counter => counter);
clock_process :process
begin
    clk  <= '0';
    wait for 10 ns;
    clk  <= '1';
    wait for 10 ns;
end process;

stim_proc: process
begin
    reset <= '1';
  wait for 20 ns;
  reset <= '0';
  wait;
end process;
end Behavioral;
```

CONCLUSION

EXPERIMENT-5

AIM OF THE EXPERIMENT-

Develop a VHDL test bench code for testing simple gates.

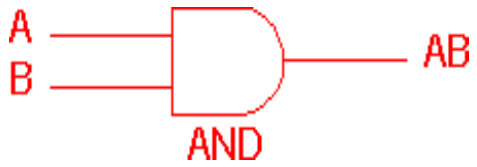
EQUIPMENT REQUIRED:

1. PC
2. XILINX ISE software

THEORY-

AND GATE-

The AND gate is an electronic circuit that gives a high output (1) only if all its inputs are high. A dot (.) is used to show the AND operation i.e. A.B. Bear in mind that this dot is sometimes omitted i.e. AB.



2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

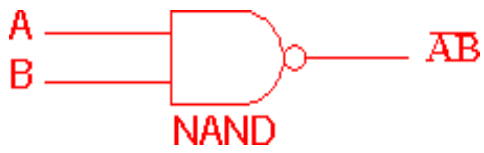
OR Gate:

The OR gate is an electronic circuit that gives a high output (1) if one or more of its inputs are high. A plus (+) is used to show the OR operation.

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NAND GATE:

This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if any of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

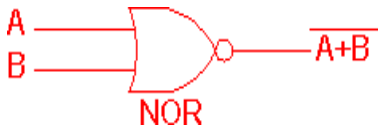


2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR GATE:

This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs

of all NOR gates are low if any of the inputs are high.
The symbol is an OR gate with a small circle on the output. The small circle represents inversion.



2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

EXOR GATE:
The 'Exclusive-OR' gate is a circuit which will give a high output if either, but not both, of its two inputs are high. An encircled plus sign () is used to show the EXOR operation.

2 Input EXOR gate		
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

PROCEDURE:

· New project and type the project name and check the top level source type as HDL.

·Enter the device properties and click Next.

· Click New Source and Select the Verilog Module and then give the file name.

·Give the Input and Output port names and click finish.

·Type the program and save it.

·Double click the synthesize XST and check syntax.

·Simulate the waveform by behavioral simulation.

PROGRAM:

VHDL PROGRAM:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity gates is

   Port ( a,b : in STD_LOGIC;

          p,q,r,s,t : out  STD_LOGIC);

end gates;

architecture Behavioral of gates is

begin

p<= a and b;
```

```
q<= a or b;
r<= a nand b;
s<= a nor b;
t<= a xor b;

end Behavioral;
```

TEST BENCH PROGRAM:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY gates_tb IS
END gates_tb;
ARCHITECTURE behavior OF gates_tb IS
   COMPONENT gates
   PORT(
       a : IN std_logic;
       b : IN  std_logic;
       p : OUT std_logic;
       q : OUT std_logic;
       r : OUT std_logic;
       s : OUT  std_logic;
       t : OUT  std_logic
      );
   END COMPONENT;
  signal a : std_logic := '0';
  signal b : std_logic := '0';
  signal p : std_logic;
  signal q : std_logic;
  signal r : std_logic;
  signal s : std_logic;
  signal t : std_logic;
BEGIN
  uut: gates PORT MAP (
        a => a,
        b => b,
```

```
        p => p,
        q => q,


        r => r,
        s => s,
        t => t
       );
     stim_proc: process
  begin
          wait for 20 ns;
            a<= '0';
             b<= '0';
             wait for 20 ns;
             a<= '0';
             b<= '1';
             wait for 20 ns;
             a<= '1';
             b<= '0';
             wait for 20 ns;
             end process;
END;
```

CONCLUSION

EXPERIMENT-6

AIM OF THE EXPERIMENT-

Develop a VHDL test bench code & implement of FPGA kit for Multiplexer and De-Multiplexer.

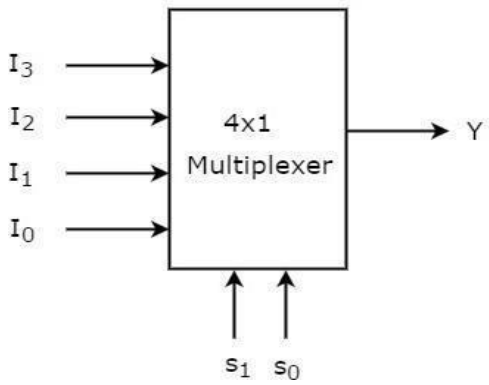
EQUIPMENT REQUIRED:

1. PC
2. XILINX ISE software
3. FPGA KIT

THEORY-

Multiplexer-

4x1 Multiplexer has four data inputs I_3, I_2, I_1 & I_0, two selection lines s_1 & s_0 and one output Y.



One of these 4 inputs will be connected to the output based on the combination of inputs present at these two selection lines.

Truth Table-

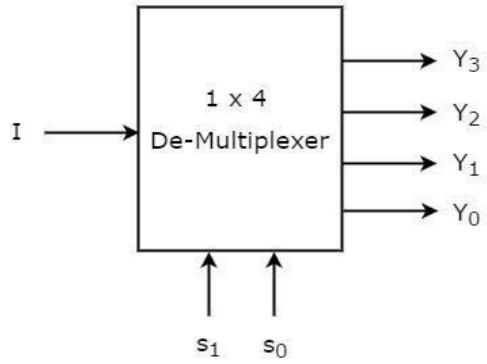
Selection Lines		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

From Truth table, we can directly write the **Boolean function** for output, Y as

$$Y=S1'S0'I0+S1'S0I1+S1S0'I2+S1S0I3$$

DE- MULTIPLEXER-

1x4 De-Multiplexer has one input I, two selection lines, s_1 & s_0 and four outputs Y_3, Y_2, Y_1 &Y_0.



The single input 'I' will be connected to one of the four outputs, Y_3 to Y_0 based on the values of selection lines s_1 & s0.

Truth Table-

Selection Inputs		**Outputs**			
S_1	**S_0**	**Y_3**	**Y_2**	**Y_1**	**Y_0**
0	0	0	0	0	**I**
0	1	0	0	**I**	0
1	0	0	**I**	0	0
1	1	**I**	0	0	0

From the above Truth table, we can directly write the **Boolean functions** for each output as

Y3=s1s0I

Y2=s1s0′I

Y1=s1′s0I

Y0=s1′s0′I

PROCEDURE:

· New project and type the project name and check the top level source type as HDL

·Enter the device properties and click Next

· Click New Source and Select the Verilog Module and then give the file name

·Give the Input and Output port names and click finish.

·Type the Verilog program and save it

·Double click the synthesize XST and check syntax

·Simulate the waveform by behavioral simulation

· For implementation Select User constraints and give input and output port pin number

·Click Implement design for Translate, map and place & route

·Generate .bit file using programming file

·Implement in FPGA through parallel-JTAG cable

·Check the behavior of design in FPGA by giving inputs

PROGRAM:

VHDL Program:

4:1 MULTIPLEXER

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity Mux is

   Port ( I0, I1, I2, I3,S0,S1 : in STD_LOGIC;

          Y : out  STD_LOGIC);

end Mux;

architecture Behavioral of Mux is

begin

Y<= ((Not S0) and (NOT S1) and I0) or ((not S0 and S1 and I1) or (S0 and (not S1) and I2) or (S0 and S1 and I3);

end Behavioral;
```

TEST BENCH PROGRAM:

```
LIBRARY ieee;

USE ieee.std_logic_1164.ALL;

 ENTITY Mux_TB IS

END Mux_TB;

ARCHITECTURE behavior OF Mux_TB IS
```

```
    COMPONENT Mux
    PORT(
        I0 : IN std_logic;
        I1 : IN std_logic;
        I2 : IN std_logic;
        I3 : IN std_logic;
        S0 : IN std_logic;
        S1 : IN std_logic;
        Y : OUT std_logic
       );
    END COMPONENT;
   signal I0 : std_logic := '0';
   signal I1 : std_logic := '0';
   signal I2 : std_logic := '0';
   signal I3 : std_logic := '0';
   signal S0 : std_logic := '0';
   signal S1 : std_logic := '0';
   signal Y : std_logic;
BEGIN
   uut: Mux PORT MAP (
         I0 => I0,
         I1 => I1,
         I2 => I2,
         I3 => I3,
         S0 => S0,
         S1 => S1,
         Y => Y
        );
   stim_proc: process
```

```
  begin
   I0<= '0';
   I1<= '1';
   I2<= '0';
   I3<= '1';
  S0<= '0';
   S1<= '0';
          wait for 100ns;
                  S0<= '0';
                  S1<= '1';
                  wait for 100ns;
                  S0<= '1';
                  S1<= '0';
                  wait for 100ns;
                  S0<= '1';
                  S1<= '1';
                  wait for 100ns;
     wait;
  end process;
END;
```

DE-MULTIPLEXER

VHDL Program:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Demux is
   Port ( Y0,Y1,Y2,Y3 : out STD_LOGIC;
        I, s0,s1 : in  STD_LOGIC);
end Demux;
architecture Behavioral of Demux is
begin
```

```
Y0<= (not s0) and (not s1) and I;
Y1<= (not s0) and s1 and I;
Y2<= s0 and (not s1) and I;
Y3<= s0 and s1 and I;
end Behavioral;
```

Test Bench Program:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY Demux_TB IS
END Demux_TB;
ARCHITECTURE behavior OF Demux_TB IS
   COMPONENT Demux
   PORT(
        Y0 : OUT std_logic;
        Y1 : OUT std_logic;
        Y2 : OUT std_logic;
        Y3 : OUT std_logic;
        I : IN std_logic;
        S0 : IN  std_logic;
        S1 : IN  std_logic
       );
   END COMPONENT;
  signal I : std_logic := '0';
  signal s0 : std_logic := '0';
  signal s1 : std_logic := '0';
  signal Y0 : std_logic;
  signal Y1 : std_logic;
  signal Y2 : std_logic;
  signal Y3 : std_logic;
```

```
BEGIN
   uut: Demux PORT MAP (
        Y0 => Y0,
        Y1 => Y1,
        Y2 => Y2,
        Y3 => Y3,
        I => I,
        S0 => s0,
        S1 => s1
      );
  stim_proc: process
  begin
   I<= '1';
 S0<= '0';
S1<= '0';
wait for 100ns;
s0<= '0';
s1<= '1';
wait for 100ns;
s0<= '1';
s1<= '0';
wait for 100ns;
s0<= '1';
s1<= '1';
wait for 100ns;
  wait;
 end process;
END;
```

CONCLUSION:

EXPERIMENT-7

AIM OF THE EXPERIMENT-

Develop a VHDL test bench & implement of FPGA kit for Encoder and Decoder.

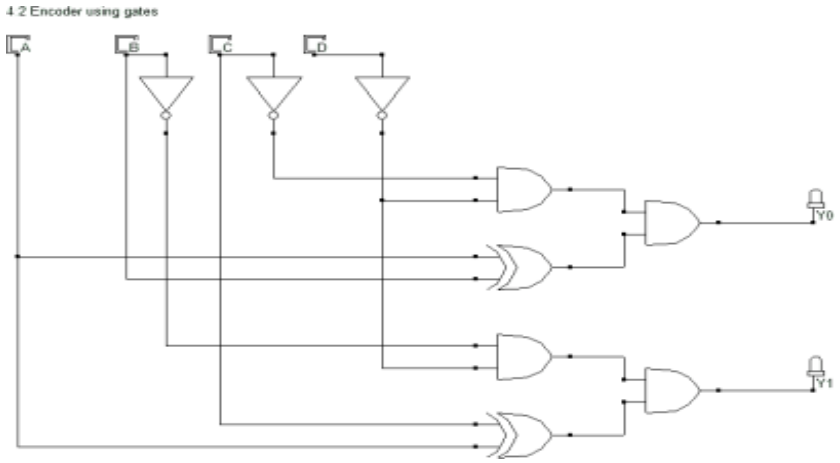
EQUIPMENT REQUIRED:

1. PC
2. XILINX ISE software
3. FPGA KIT

THEORY-

ENCODER-

An encoder is a combinational logic circuit that takes in multiple inputs, encodes them, and outputs an encoded version with fewer bits. A 4:2 encoder has four input ports and two output ports.



Y0 = A'BC'D' + AB'C'D' = C'D'($A \oplus B$)

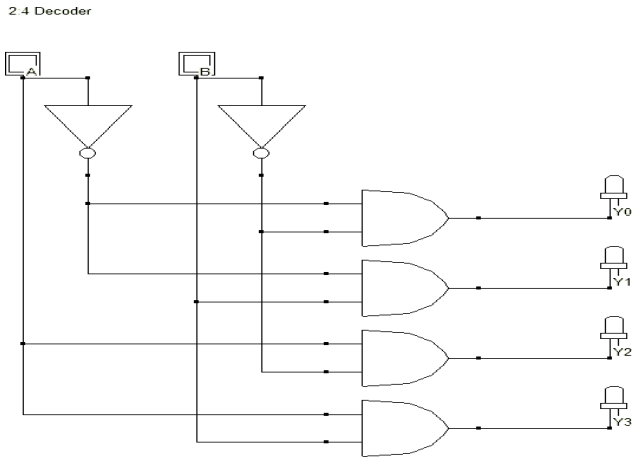
Y1 = A'B'CD' + AB'C'D' = B'D'($A \oplus C$)

Truth table of a 4:2 encoder

A	B	C	D	Y0	Y1
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

DECODER-

A decoder is a combinational logic circuit that does the opposite job of an encoder. It takes in a coded binary input and decodes it to give a higher number of outputs. 2:4 decoder has two input ports and four output ports.



Truth table for a 2:4 decoder

A	B	Y3	Y2	Y1	Y0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Y0 = A'B'

Y1 = A'B

Y2 = AB'

Y3 = AB

PROCEDURE:

· New project and type the project name and check the top level source type as HDL

·Enter the device properties and click Next

· Click New Source and Select the Verilog Module and then give the file name

·Give the Input and Output port names and click finish.

·Type the Verilog program and save it

·Double click the synthesize XST and check syntax

·Simulate the waveform by behavioral simulation

· For implementation Select User constraints and give input and output port pin number

·Click Implement design for Translate, map and place & route

·Generate .bit file using programming file

·Implement in FPGA through parallel-JTAG cable

·Check the behavior of design in FPGA by giving inputs

PROGRAM:
ENCODER-

VHDL Program:

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;
Use IEEE.STD_LOGIC_ARITH.ALL;
Use IEEE.STD_LOGIC_UNSIGNED.ALL;
Entity ENCODER_SOURCE is
   Port (A, B, C, D: in STD_LOGIC;
         Y0, Y1: out STD_LOGIC);
End ENCODER_SOURCE;
Architecture dataflow of ENCODER_SOURCE is
Begin
Y0 <= ((not C) and (not D)) and (A xor B);
Y1 <= ((not B) and (not D)) and (A xor C);
End dataflow;
```

TEST BENCH PROGRAM:

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;
Use IEEE.STD_LOGIC_ARITH.ALL;
Use IEEE.STD_LOGIC_UNSIGNED.ALL;
Entity encoder_dataflow_tb is
End entity;
Architecture tb of encoder_dataflow_tb is
Component ENCODER_SOURCE is
Port (A, B, C, D: in STD_LOGIC;
Y0, Y1: out STD_LOGIC);
End component;
Signal A, B, C, D, Y0, Y1: STD_LOGIC;
Begin
uut: ENCODER_SOURCE port map(
A => A, B => B,
C => C, D => D,
Y0 => Y0, Y1 => Y1);
stim: process
```

```
begin
A <= '0';
B <= '0';
C <= '0';
D <= '1';
wait for 20 ns;
A <= '0';
B <= '0';
C <= '1';
D <= '0';
wait for 20 ns;
A <= '0';
B <= '1';
C <= '0';
D <= '0';
wait for 20 ns;
A <= '1';
B <= '0';
C <= '0';
D <= '0';
wait for 20 ns;
wait;

end process;
end tb;
```

DECODER

VHDL Program:

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

entity DECODER_SOURCE is
   Port ( A,B : in  STD_LOGIC;
          Y3,Y2,Y1,Y0 : out STD_LOGIC);
end DECODER_SOURCE;

architecture dataflow of DECODER_SOURCE is

begin

Y0 <= ((not A)and(not B));
Y1 <= ((not A) and B);
Y2 <= (A and (not B));
Y3 <= (A and B)

end dataflow;
```

Test Bench Program:

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

entity decoder_tb is
end entity;

architecture tb of decoder_tb is
component DECODER_SOURCE is
Port ( A,B : in STD_LOGIC;
Y3,Y2,Y1,Y0 : out STD_LOGIC);
end component;

signal A, B, Y3, Y2, Y1, Y0 : STD_LOGIC;

begin

uut: DECODER_SOURCE port map(
A => A, B => B,
Y0 => Y0, Y1 => Y1,
Y2 => Y2, Y3 => Y3);

stim: process
begin

A <= '0';
B <= '0';
wait for 20 ns;

A <= '0';
B <= '1';
wait for 20 ns;

A <= '1';
B <= '0';
wait for 20 ns;

A <= '1';
B <= '1';
wait for 20 ns;

wait;

end process;

end tb;
```

CONCLUSION:

EXPERIMENT-8

AIM OF THE EXPERIMENT-

Design and implementation of Half Adder and Full Adder.

EQUIPMENT REQUIRED:

1. PC
2. XILINX ISE software
3. FPGA KIT

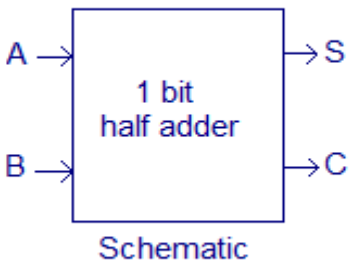
THEORY-

Half Adder-

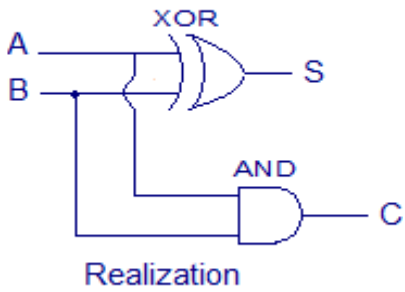
Half adder is a combinational arithmetic circuit that adds two numbers and produces a sum bit (S) and carry bit (C) as the output. If A and B are the input bits, then sum bit (S) is the X-OR of A and B and the carry bit (C) will be the AND of A and B.

Inputs		Outputs	
A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Truth table



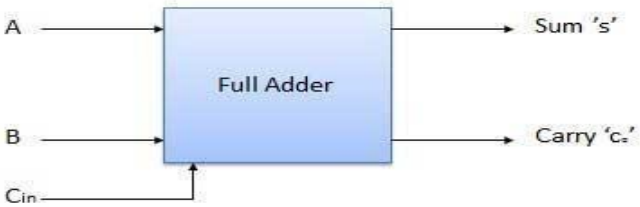
Schematic

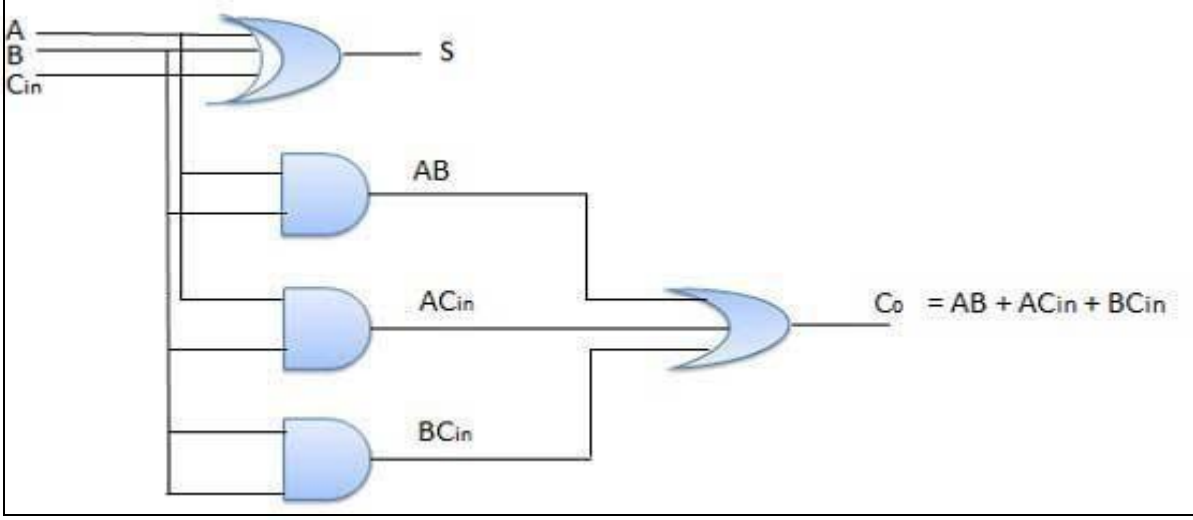


Realization

Full Adder-

The full adder is a three-input and two output combinational circuit.





Inputs			Output	
A	B	Cin	S	Co
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

PROCEDURE:

· New project and type the project name and check the top level source type as HDL

·Enter the device properties and click Next

· Click New Source and Select the Verilog Module and then give the file name

·Give the Input and Output port names and click finish.

·Type the Verilog program and save it

·Double click the synthesize XST and check syntax

·Simulate the waveform by behavioral simulation

· For implementation Select User constraints and give input and output port pin number

·Click Implement design for Translate, map and place & route

·Generate .bit file using programming file

·Implement in FPGA through parallel-JTAG cable

·Check the behavior of design in FPGA by giving inputs

PROGRAM:
HALF ADDER
VHDL Program:

```
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 Behavioral of Half_Adder is
begin
S<= A xor B;
C<= A and B;
```

end Behavioral;

FULL ADDER

VHDL Program:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Full_Adder is
   Port ( A,B,Cin : in STD_LOGIC;
          Co, S : out STD_LOGIC);
end Full_Adder;
architecture Behavioral of Full_Adder is
begin
S<= A xor B xor Cin;
Co<= (A and B)or (B and Cin )or (Cin and A);
end Behavioral;
```

CONCLUSION:

EXPERIMENT-9

AIM OF THE EXPERIMENT-

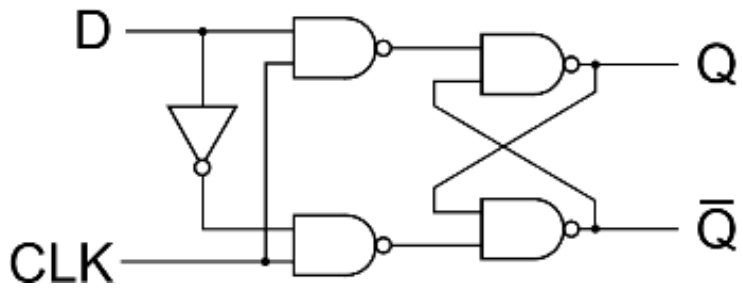
Design and implementation of D flip flop.

EQUIPMENT REQUIRED:

1. PC
2. XILINX ISE software

THEORY:

Latch is an electronic device that can be used to store one bit of information.
The D latch is used to capture, or 'latch' the logic level which is present on the Data line when the clock input is high.
If the data on the D line changes state while the clock pulse is high, then the output, Q, follows the input, D. When the CLK input falls to logic 0, the last state of the D input is trapped and held in the latch.



PROCEDURE:

· New project and type the project name and check the top level source type as HDL

·Enter the device properties and click Next

· Click New Source and Select the Verilog Module and then give the file name

·Give the Input and Output port names and click finish.

·Type the Verilog program and save it

·Double click the synthesize XST and check syntax

·Simulate the waveform by behavioral simulation

· For implementation Select User constraints and give input and output port pin number

·Click Implement design for Translate, map and place & route

·Generate .bit file using programming file

·Implement in FPGA through parallel-JTAG cable

·Check the behavior of design in FPGA by giving inputs

PROGRAM:

VHDL Program:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity D_flipflop is
   Port ( D,CLK : in  STD_LOGIC;
          q,qbar : inout STD_LOGIC);
end D_flipflop;
architecture Behavioral of D_flipflop is
signal d1, d2:STD_LOGIC;
begin
d1<= D nand  CLK;
d2<= (not D) nand CLK;
q<= d1 nand qbar;
qbar<= d2 nand q;
end Behavioral;
```

CONCLUSION:

EXPERIMENT-10

AIM OF THE EXPERIMENT:

Write a simple program with two separate LED blinking tasks.

EQUIPMENTS REQUIRED:

- **Nvis 5004B** board
- Adapter 5volt/1Amp
- USB Cable

THEORY:

- LED is a semiconductor device used in many electronic devices, mostly used for indication purposes. It is used widely as indicator during test for checking the validity of results at different stages.
- It is very cheap and easily available in variety of shape, color and size. The LEDs are also used in designing of message display boards and traffic control signal lights etc.

PROCEDURE:

- Connect the USB cable to USB port of your PC and UART0 (B Type USB) port of the board (in PC interface & ISP section) provided on the board.
- Change the position of Run/ISP switch placed in" PC Interface & ISP Section" block on the ISP mode.
- Turn ON switch no 1, 2 placed in" I2C & SPI" block.
- Connect the power cable to the board and switch 'ON' the power switch.
- Start the Philips flash utility (available in the all programs on the Start menu of Windows OS: Start menu/All programs/Philips semiconductor/Flash utility/Launch LPC210x_ISP.exe.) and select the appropriate port settings (use baud rate 9600).
- Program" Blinky Task.hex" (CD-drive\ RTOS Program\1.Blinky\ Blinky Task.hex).
- Switch 'OFF' the power supply and change the position of Run/ISP switch placed in" PC Interface & ISP Section" block on the RUN mode.
- Put all the switches provided in the 'LED Interface' block in the ON position.
- Switch 'On' the supply, then press reset switch.
- Observe simultaneously glowing of two LED's.

PROGRAM:

```
#include <LPC214x.H>
/*****************************************************************************
*                   Delay
* Description : This function provide Delay in Mili Sec.
*****************************************************************************/
void MSdelay(unsigned int rTime)
{
        unsigned int i,j;
        for(i=0;i<=rTime;i++)
        for(j=0;j<4867;j++);
}

/*****************************************************************************
*******************
*                                   Main:
* Description : This function used to interface 8 LEDs.
*****************************************************************************
******************/
int main(void)
{
        IO1DIR = 0x00FF0000;                            /* Define Port1 pin P1.16 to
P1.23 as output */
        while (1)
        {
                IO1SET = 0x00FF0000;                    /* Glow All LEDs */
                MSdelay (100);                  /* Delay */
                IO1CLR = 0x00FF0000;
                MSdelay (100);                  /* Delay */



}
```

CONCLUSION: