VHDL AND CAD Laboratory
(EC-452) Lab Manual

Prepared by
Y.Sri Chakrapani, M.Tech
(Lecturer)
&
S.K.M.Subhani, M.Tech
(Lecturer)

DEPARTMENT OF ECE
BAPATLA ENGINEERING COLLEGE
BAPATLA

LIST OF EXPERIMENTS

1. LOGIC GATES 2

2. MODULO SYNCHRONOUS UP-DOWN COUNTER 11

3. MULTIPLERXERS/DECODERS 13

4. DESIGN OF ALU 17

5. PRIORITY ENCODER 19

6. 4-BIT MAGNITUDE COMPARATOR 21

7. PARITY GENERATOR 23

8. LINEAR CONVOLUTION 25

9. CIRCULAR CONVOLUTION 26

10. AUTO-CORRELATION 28

11. CROSS-CORRELATION 30

12. FFT USING DIF 31

13. IFFT USING DIF 33

14. FFT USING DIT 34

15. IFFT USING DIT 36

16. DESIGN OF BUTTER WORTH FILTER USING
IMPUSEINVARIENT METHOD 37

18 DESIGN OF LPF USING HAMMING WINDOWS 38

19. DFT 39

1.LOGIC GATES

1.1 AND GATE

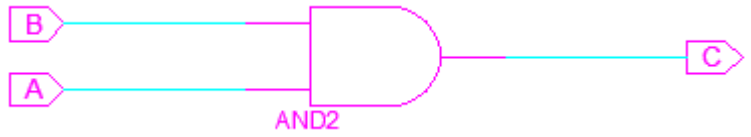
Aim:

To implement And Gate using VHDL.

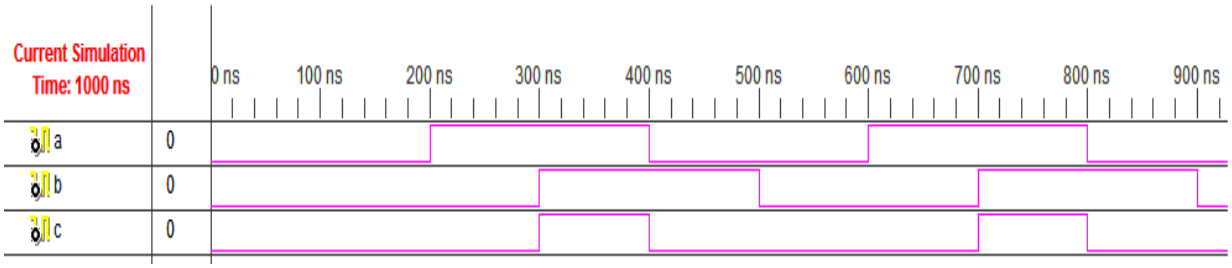
Code:

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

entity and1 is
     port(A,B : in std_logic;
               C : out std_logic);
end and1;

architecture archi of and1 is
begin
     C <= A and B;
end archi;
```

RTL SCHEMATIC:



Simulation Waveforms



1.2 OR GATE

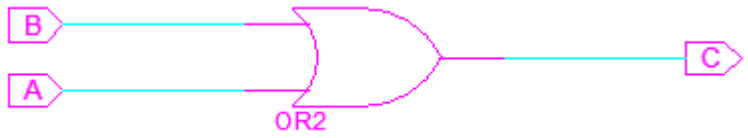
Aim:

To implement And Gate using VHDL.

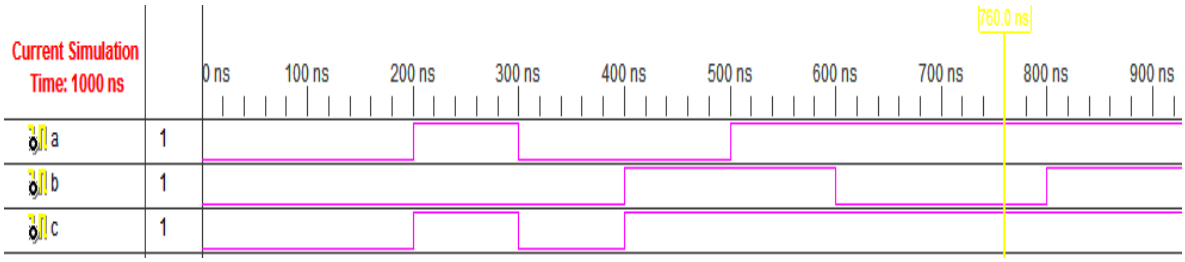
Code:

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

entity or1 is
     port(A,B : in std_logic;
             C : out std_logic);
end or1;

architecture archi of or1 is
begin
     C <= A or B;
end archi;
```

RTL Schematic:



Simulation Waveforms:



1.3 NAND GATE

Aim:

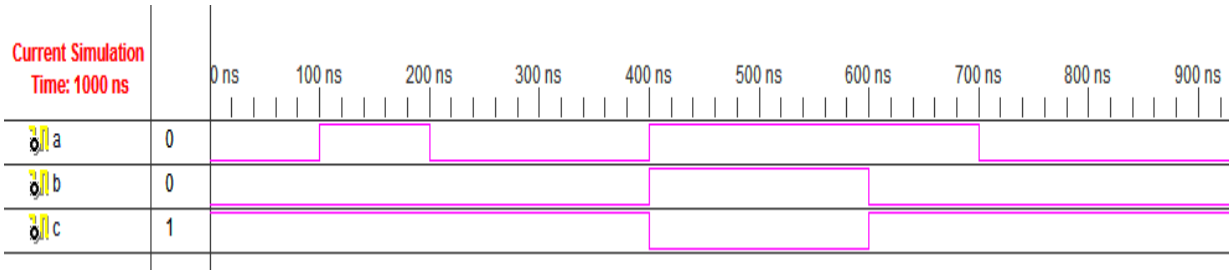
To implement Nand Gate using VHDL.

Code:

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

entity nand1 is
     port(A,B : in std_logic;
             C : out std_logic);
end nand1;

architecture archi of nand1 is
begin
     C <= A nand B;
end archi;
```

RTL Schematic:

Simulation Waveforms:



1.4 EX-OR GATE

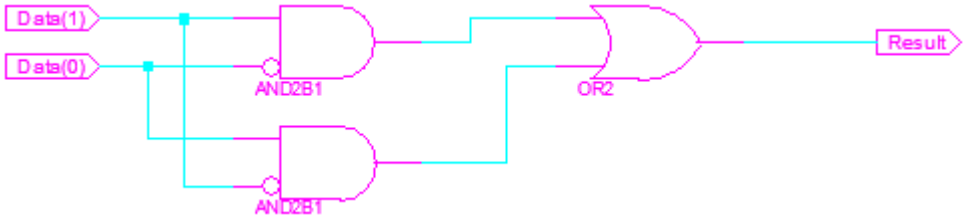
Aim:

To implement Nand Gate using VHDL.

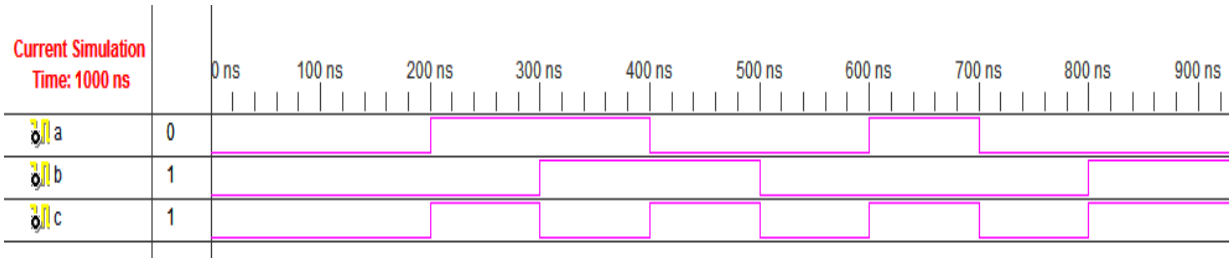
Code:

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

entity exor1 is
     port(A,B : in std_logic;
             C : out std_logic);
end exor1;

architecture archi of exor1 is
begin
     C <= A xor B;
end archi;
```

RTL Schematic:



Simulation Waveforms:



1.5 NOT GATE

Aim:

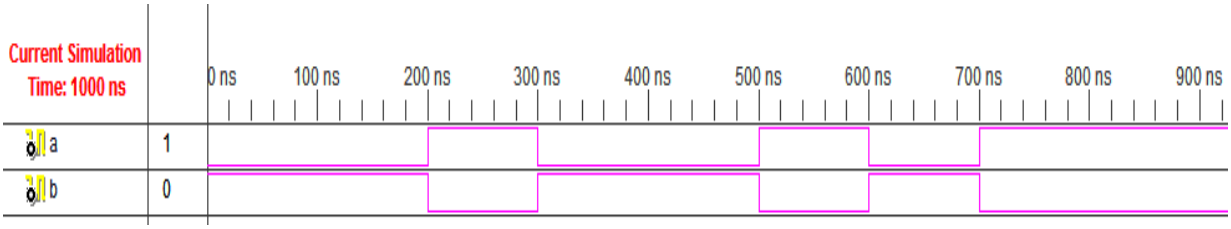
To implement Not Gate using VHDL.

Code:

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

entity exor1 is
     port(A : in std_logic;
          B : out std_logic);
end exor1;

architecture archi of exor1 is
begin
     B<=  not A;
end archi;
```

RTL Schematic:

Simulation Waveforms:



1.6 HALF ADDER

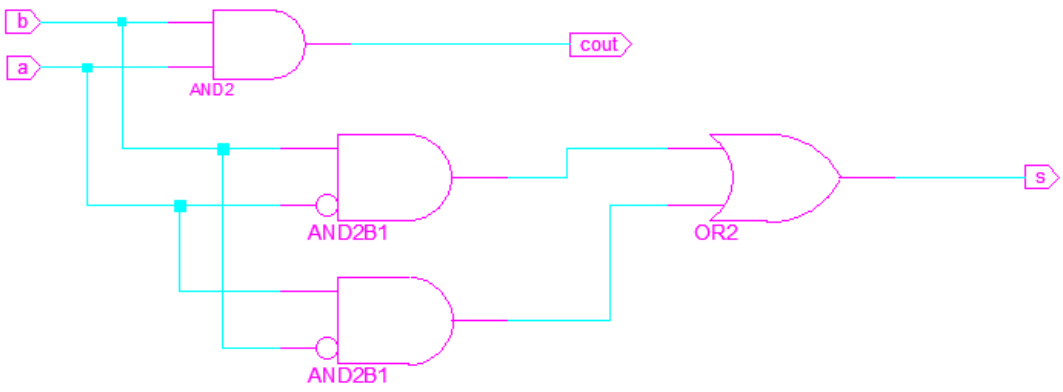
Aim:

To implement Half Adder using VHDL.

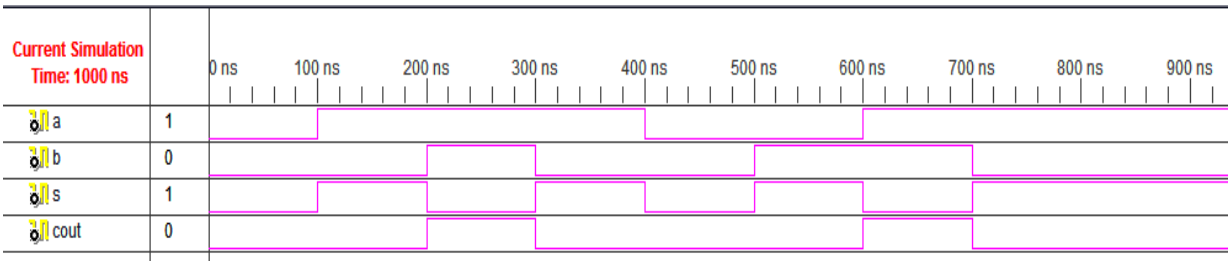
Code:

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

entity halfadder is
     port(      a     : in  std_logic;
                b     : in  std_logic;
                s     : out std_logic;
                cout  : out std_logic);
end halfadder;

architecture Behavioral of halfadder is

begin
  s <= a xor b;
  cout<= a and b;

end Behavioral;
```

RTL Schematic:



Simulation Waveforms:



1.7 FULL ADDER

Aim:

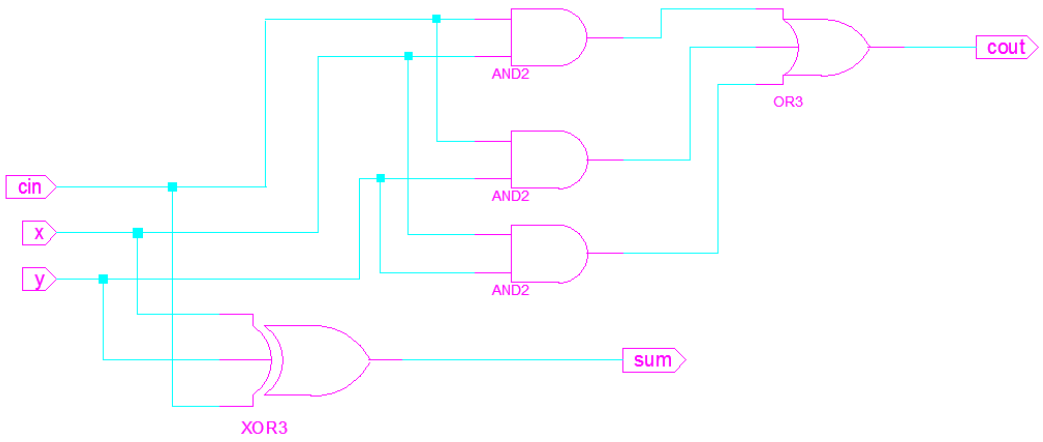
To implement Full Adder using VHDL.

Code:

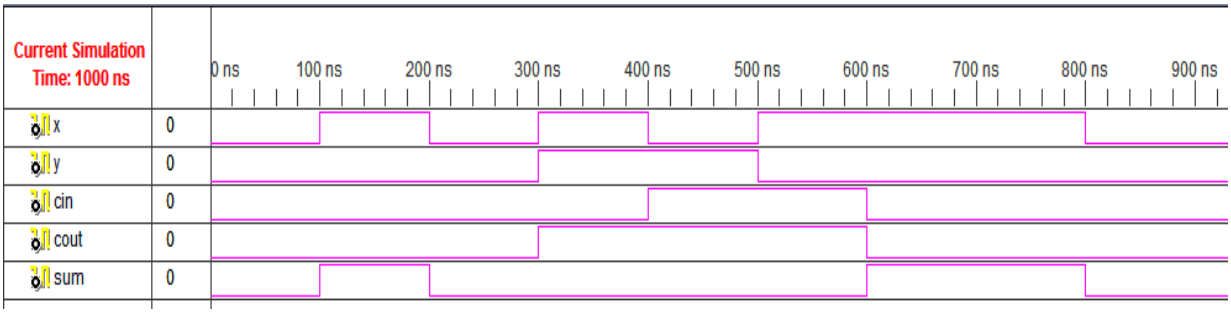
```
LIBRARY  ieee ;
USE ieee.std_logic_1164.all ;
```

```
ENTITY full_adder IS
   PORT(x:    IN   std_logic ;
        y:    IN   std_logic ;
        cin:  IN   std_logic ;
        cout: OUT  std_logic ;
        sum:  OUT  std_logic
   ) ;
END full_adder ;

ARCHITECTURE equations OF full_adder IS
BEGIN
   sum  <= x XOR y XOR cin ;
   cout <= (x AND y) OR (x AND cin) OR (y AND cin) ;
END ;
```

RTL Schematic:



Simulation waveforms:



2. MODULO SYNCRONOUS UP-DOWN COUNTER

Aim:

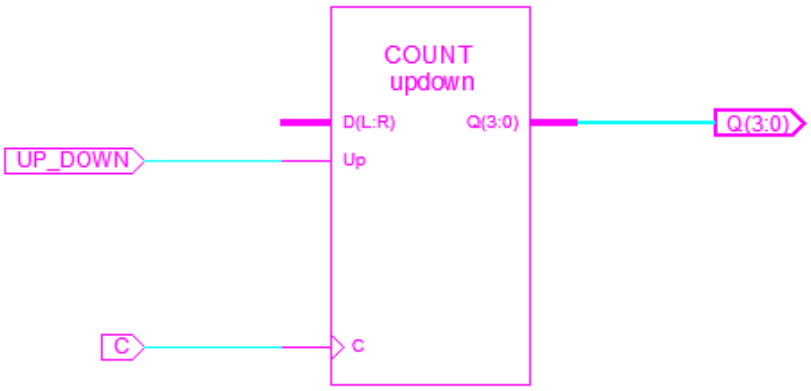
To implement Modulo Syncronous Up-Down Counter using VHDL.

Code:

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

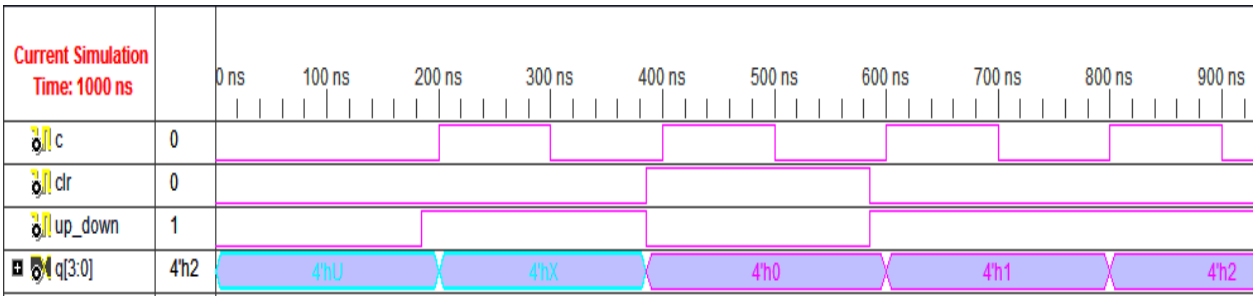
```
entity counter is
     port(C, CLR, UP_DOWN : in std_logic;
          Q : out std_logic_vector(3 downto 0));
end counter;

architecture archi of counter is
     signal tmp: std_logic_vector(3 downto 0);
begin
     process (C, CLR)
     begin
          if (CLR='1') then
               tmp <= "0000";
          elsif (C'event and C='1') then
               if (UP_DOWN='1') then
                    tmp <= tmp + 1;
               else
                    tmp <= tmp - 1;
               end if;
          end if;
     end process;
Q <= tmp;
end archi;
```

RTL Schematic:



.

Simulation Waveforms:



3.1 MULTIPLEXER 8x1

Aim:

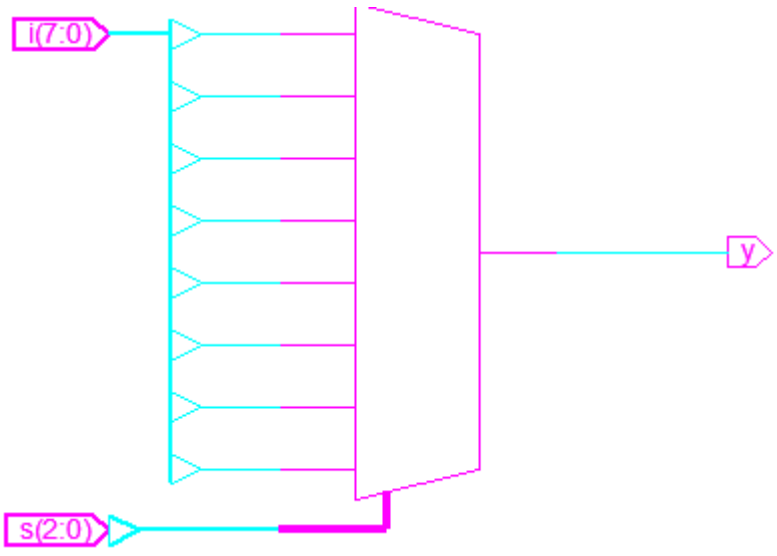
To implement Multiplexer 8x1using VHDL.

Code:

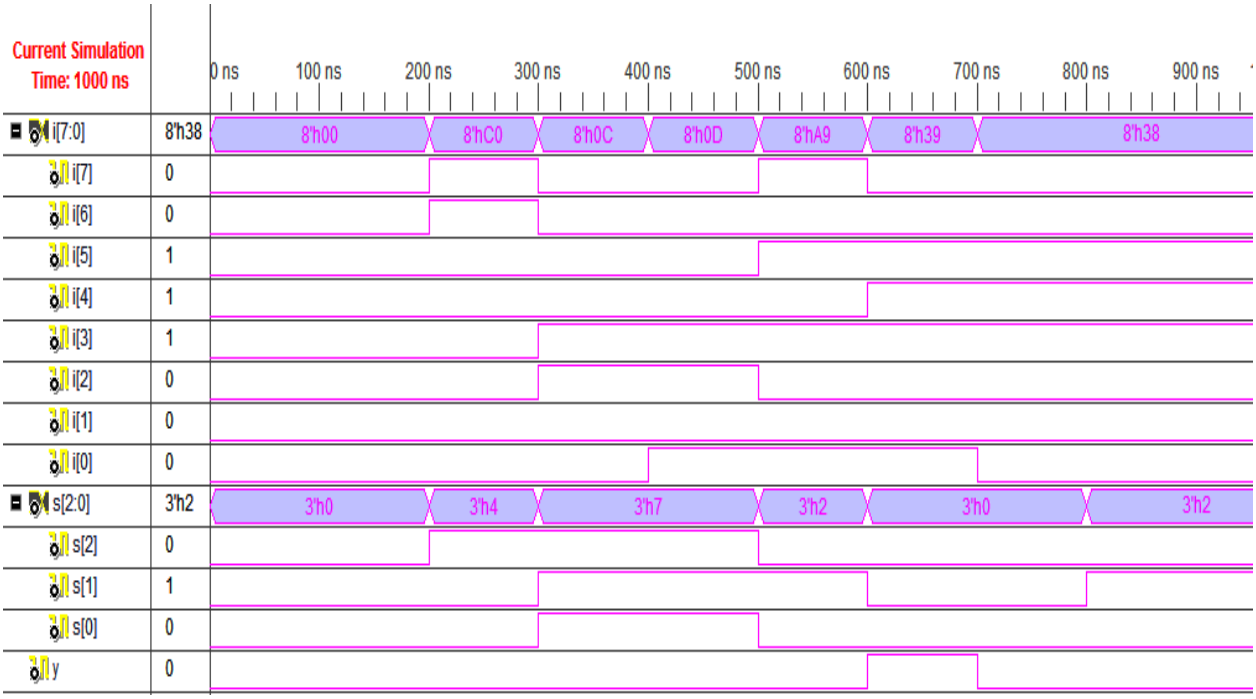
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
entity mux8x1 is
              Port (
              i : in std_logic_vector(7 downto 0);
                 s : in std_logic_vector (2 downto 0);
                 y : out std_logic
               );

end mux8x1;

architecture Behavioral of mux8x1 is

begin

              process (i,s)
              begin
                   if (s = "000") then y <= i(0);
                   elsif (s = "001") then y <= i(1);
                   elsif (s = "010") then y <= i(2);
                   elsif (s = "011") then y <= i(3);
                   elsif (s = "100") then y <= i(4);
                   elsif (s = "101") then y <= i(5);
                   elsif (s = "110") then y <= i(6);
                   else y <= i(7);
                   end if;
                   end process;


end Behavioral;
```

RTL Schematic:



Simulation Waveforms:



3.2 DECODER 2x4

Aim:

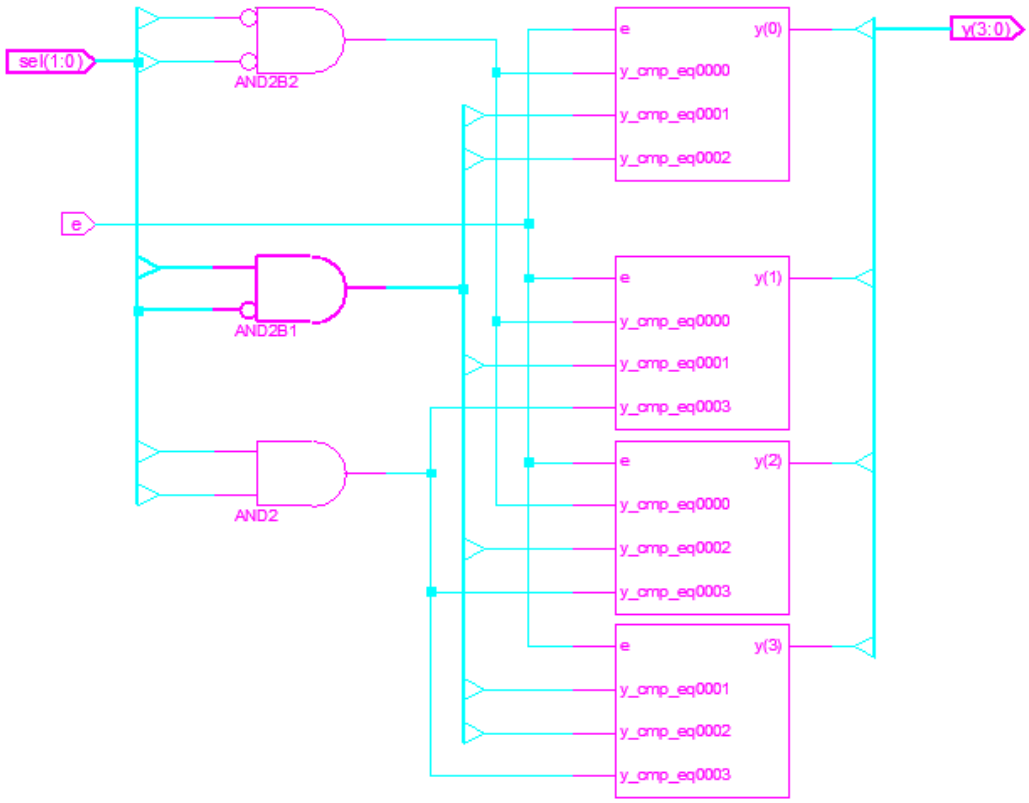
To implement Decoder 2x4 using VHDL.

Code:

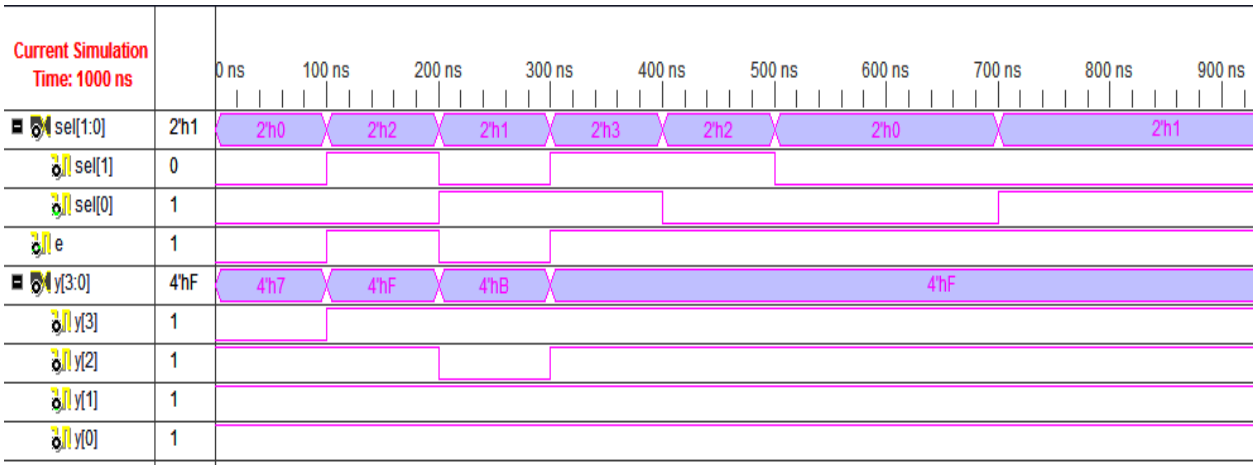
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity decoder2x4 is
```

```
port (sel: in std_logic_vector (1 downto 0);
           e: in std_logic;
           y: out std_logic_vector (3 downto 0));

end decoder2x4;

architecture Behavioral of decoder2x4 is
begin
          process(e,sel)
          begin
          if(e='0') then
              if(sel = "00") then
                    y <= "0111" ;
              elsif(sel = "01") then
                    y <= "1011" ;
              elsif(sel = "10") then
                    y <= "1101" ;
              else
                    y <= "1110" ;
              end if;

          else
              y <= "1111";
          end if;
         end process;
end Behavioral;
```

RTL Schematic:



Simulation Waveforms:



4.DESIGN OF ALU

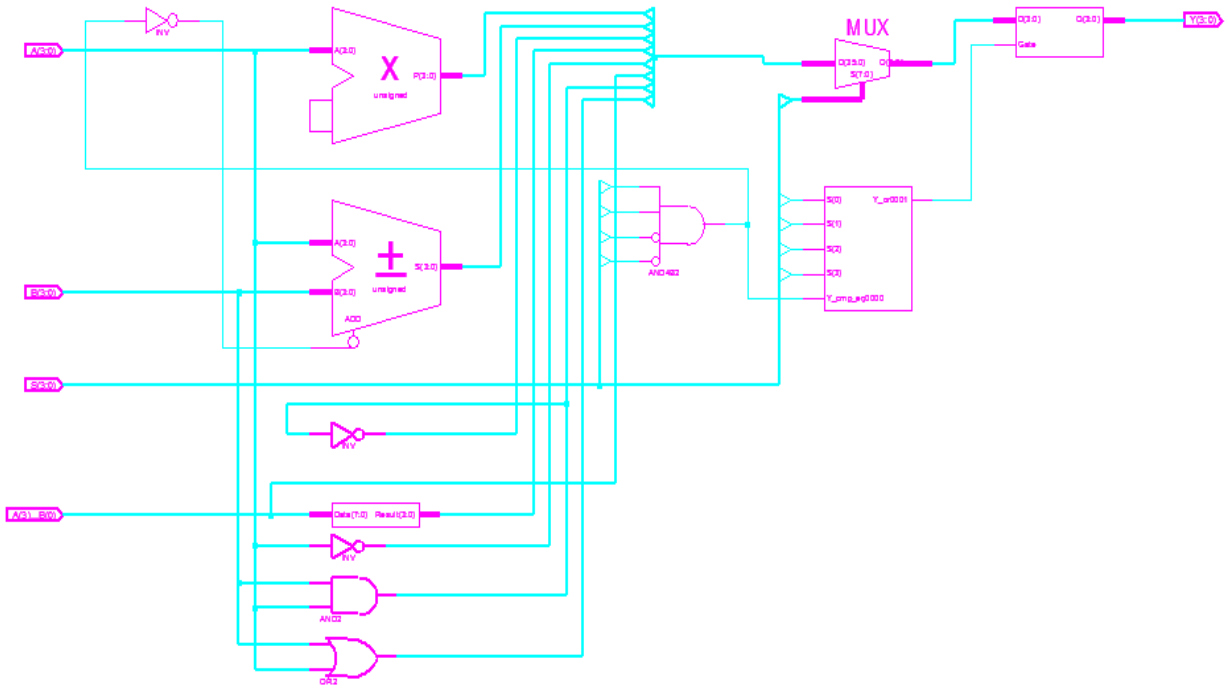
Aim:

To implement 4-Bit ALU using VHDL.

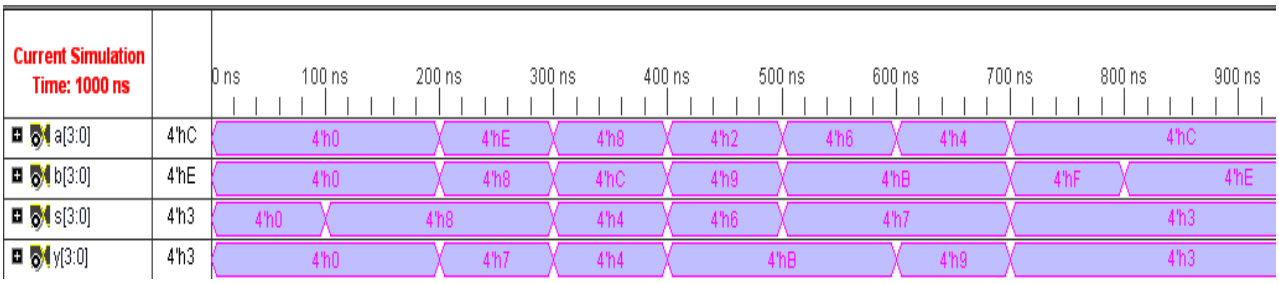
Code:

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


entity LC2_ALU is
   port( A: in std_logic_vector (3 downto 0);
      B: in std_logic_vector (3 downto 0);
      S: in std_logic_vector (3 downto 0);
      Y: out std_logic_vector (3 downto 0));
end LC2_ALU;


architecture bhv of LC2_ALU is
begin
   process(A, B, S)
   begin

   case S is
      when "0000" => Y <= A or B;
      when "0001" =>  Y<= A and B;
      when "0010" => Y <= A;
      when "0011" => Y <= not A;
      when "0100" => Y <= A xor B;
      when "0101" => Y <= A nand B;
      when "0110" => Y <= A + B;
      when "0111" => Y <= A - B;
      when "1000" => Y <= A*B;
      when "1001" => Y <= A/B;
      when others => null;
   end case;

end process;
end bhv;
```

RTL Schematic:



Simulation Waveforms:



5. PRORITY ENCODER

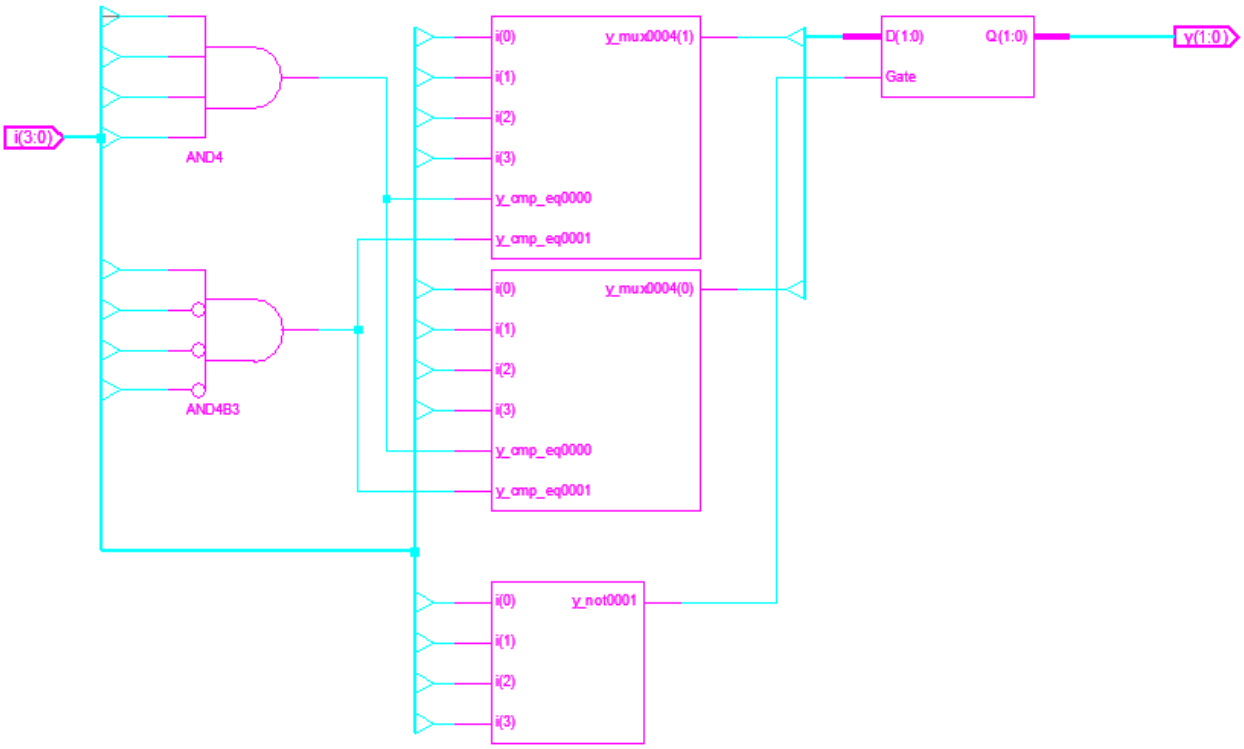
Aim:

To implement Priority Encoder 4x2 using VHDL.

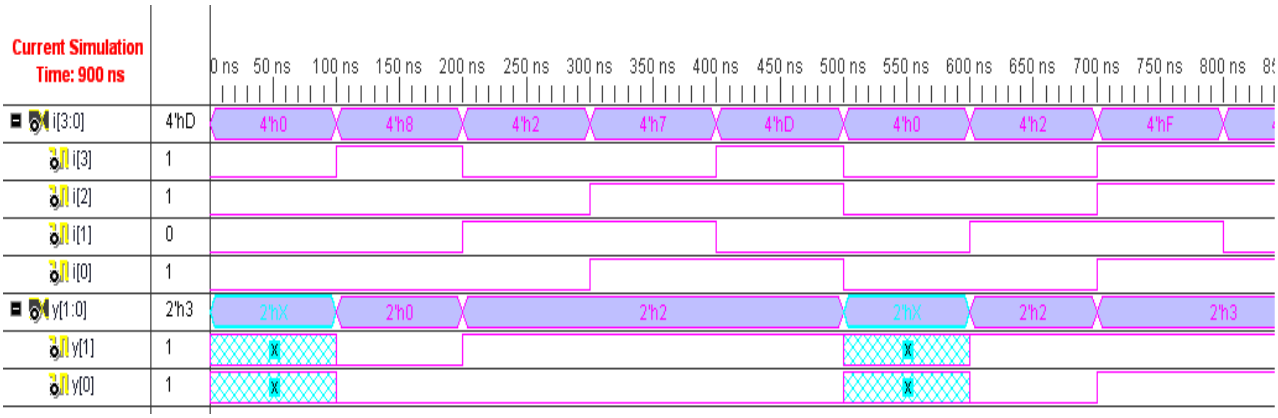
Code:

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

entity priority_encoder4x2 is
    Port (    i : in std_logic_vector(3 downto 0);
                  y : out std_logic_vector(1 downto 0)
     );
end priority_encoder4x2;

architecture Behavioral of priority_encoder4x2 is

begin

     process (i)
     begin
          if (i = "0000") then
               y <="XX”;
          else
                if (i = "0001")or (i = "1111") then y <=
"11";
               elsif (i = "0010")or (i = "1110")    then
y <= "10";
               elsif (i = "0100")or (i = "1100")    then
y <= "01";
               elsif (i = "1000")              then y <=
"00";
               end if;

          end if;

     end process;
end Behavioral;
```

RTL Schematic:



Simulation Waveforms:



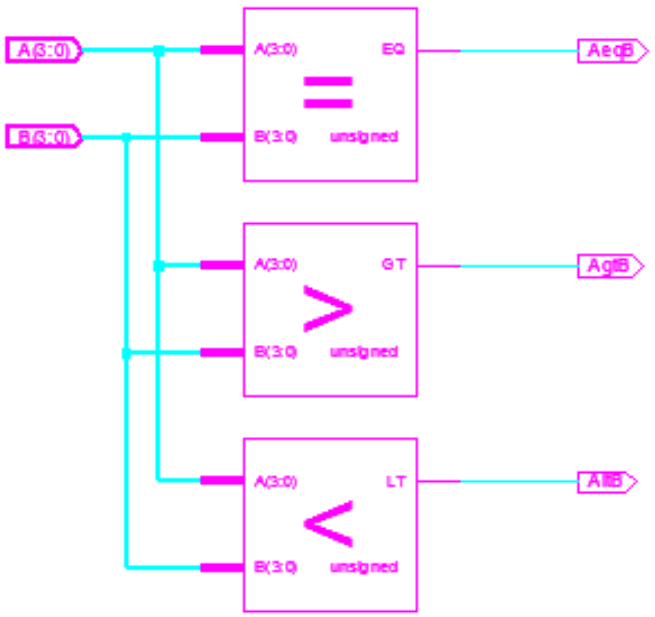
6. MAGNITUDE COMPARATOR

Aim:

To implement 4-Bit Magnitude Comparator using VHDL.

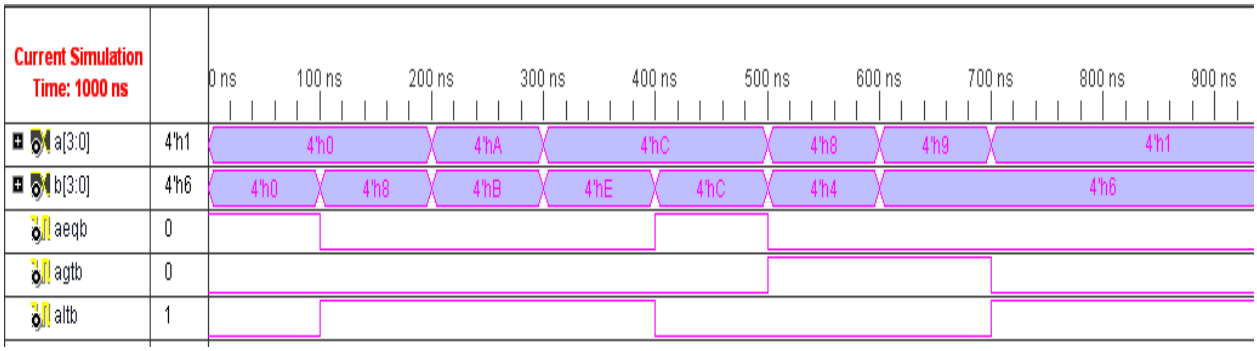
Code:

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;

ENTITY compare IS
     PORT ( A, B : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
          AeqB, AgtB, AltB : OUT STD_LOGIC ) ;
END compare ;


ARCHITECTURE Behavior OF compare IS
BEGIN
     AeqB <= '1' WHEN A = B ELSE '0' ;
     AgtB <= '1' WHEN A > B ELSE '0' ;
     AltB <= '1' WHEN A < B ELSE '0' ;
END Behavior ;
```



RTL Schematic:

Simulation Waveforms:



7. PARITY GENERATOR

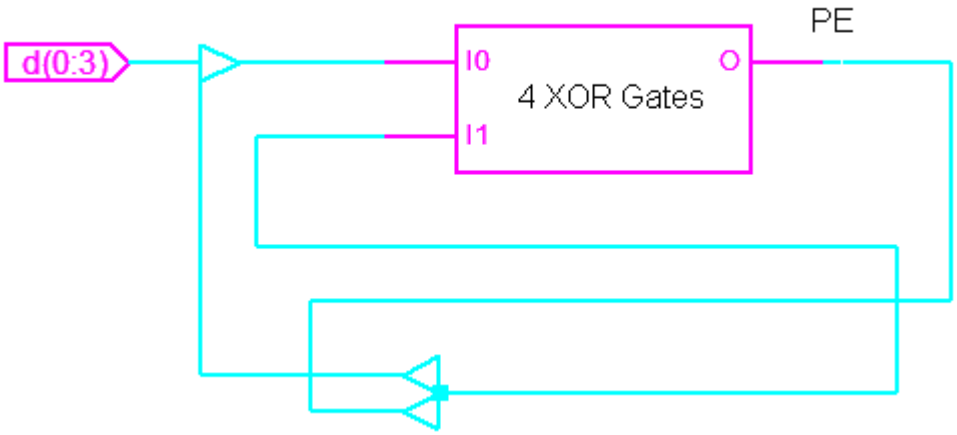
Aim:

To implement 4-Bit Parity Generator using VHDL.

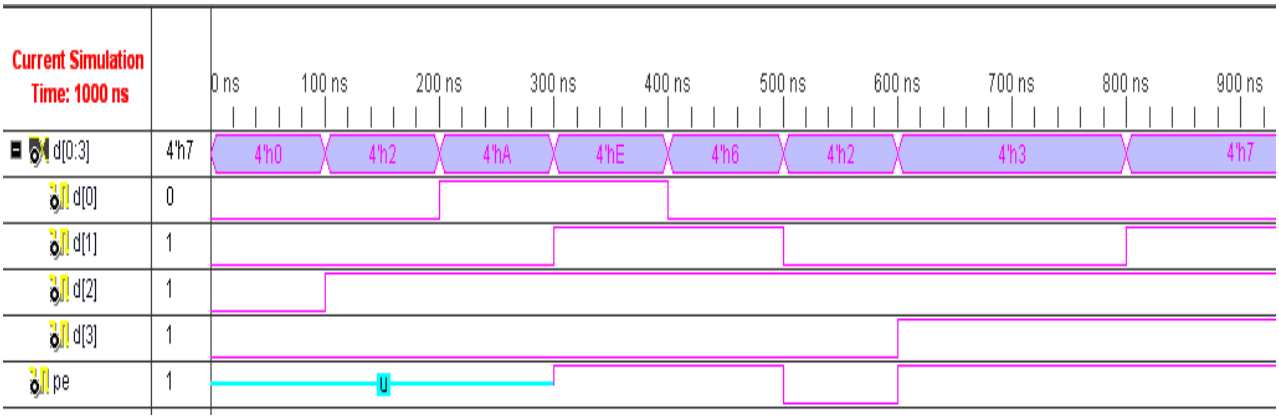
Code:

```
LIBRARY ieee;
USE ieee.std_logic1164.ALL;

ENTITY parity4_gen IS
PORT(d : IN STD_LOGIC_VECTOR (0 to3);
            pe : OUT STD_LOGIC);
END parity4_gen;

ARCHITECTURE parity OF parity4_gen IS
     SIGNAL p : STD_LOGIC_VECTOR (1 to 3);
BEGIN
     p(1) <= d(0) xor d(1);
          FOR i IN 2 to 3 GENERATE
          p(i) <= p(i-1) xor d(i);
     END GENERATE;
     pe <= p(3);
END parity;
```

RTL Schematic:



Simulation Waveforms:



8.LINEAR CONVOLUTION

```
x=input ('Enter first sequence;')
h=input ('Enter Second sequence;')
m=length(x);
n=length (h);
l=m+n-1;
for i=1:l
    r(i)=0;
end
for k=1:l
    for j=max(1,k+1-n):min(k, m)
        r(k)=r(k)+x(j)*h(k+1-j);
    end
end
```

……

```
Enter first sequence; [1 2 3]
Enter Second sequence; [1 2 3 4]

r = [ 1     4    10    16    17    12]
```

9.CIRCULAR CONVOLUTION

```
a=input('Enter first sequence:');
b=input('Enter Second sequence:');
x=length(a);
y=length(b);
z=max(x,y);
if x>y
for i=y+1:x
    b(i)=0;
end
elseif x<y
            for j=x+1:y
    a(j)=0
        end
end
for i=1:z
     c(i)=0;
    for k=1:z
    j=(i-k)+1;
```

```
    if (j<=0)
        j=z+j;
    end
    c(i)=c(i)+(a(k)*b(j));
    end
end
```

……..

```
Enter first sequence: [1 2 2 2]

Enter Second sequence: [1 2 3]

c =[   11    10     9    12]
```

10.AUTO-CORRELATION

```
a=input('Enter the sequence:');
n1=length (a);
 n=2*n1-1;
b=zeros(1,n);
c=zeros(1,n);
d=zeros(1,n);
n2=n-n1;
 for i=1:n1
b(i)=a(i);
end
 for i=n2:n-1
c(i+1)=a(i+1-n2);
end
for i=1:n
d(i)=sum(b.*c);
for j=n:-1:2
b(j)=b(j-1);
end
```

```
b(i)=0;
end
Enter first sequence: [1 2 3 4]

Ans = [4  11  20  30  20  11  4]
```

11. CROSS-CORRELATION

```
x=input('Enter first sequence:');
h1=input('Enter Second sequence:');
m=length(x);
n=length(h1);
h(n:-1:1)=h1(1:n);
l=m+n-1;
for i=1:l
    r(i)=0;
end
for k=1:l
    for j=max(1,k+1-n):min(k,m)
        r(k)=r(k)+x(j)*h(k+1-j);
    end
end
```

…….

```
Enter first sequence:[1 2 2 2]

Enter Second sequence:[2 2 3]
ans: r =[  3     8    12    14     8     4]
```

12.FFT USING DIF

```
 function a=diffft(a)
N=length(a);
l=log2(N);
for     m=l:-1:1
for t=0:1:((2^(m-1))-1)
    k(t+1)=(N*t)/2^m;
    wn(t+1)=exp((-2*i*pi*k(t+1))/N);
end
for p=1:2^m:(N-2^(m-1))
    r=1;
    x=p;
    for q=x+2^(m-1):1:(x+2^m)-1
        b(p)=a(p)+a(q);
        b(q)=(a(p)-a(q))*wn(r);
        p=p+1;
        r=r+1;
    end
end
```

```
a=b;

end

a=bitrevorder(a);

return;
```
…….

```
 a=[1 2 3 4]

X=fftdif(a)

x =  10.0000             -2.0000 + 2.0000i  -2.0000
-2.0000 - 2.0000i
```

13.IFFT USING DIF

```
function r =idftdif(a)

a=conj(a);

a=diffft(a);

a=(a/length(a));

r=conj(a);

return;
```

…….

```
 A =  10.0000             -2.0000 + 2.0000i  -2.0000
-2.0000 - 2.0000i

r =idftdif(a)

a=[1  2  3  4]
```

14.FFT USING DIT

```
function b=ditfft(a)
a=bitrevorder(a);
N=length(a);
l=log2(N);
for     m=1:1:l
for t=0:1:((2^(m-1))-1)
    k(t+1)=(N*t)/2^m;
    wn(t+1)=exp((-2*i*pi*k(t+1))/N);
end
for p=1:2^m:(N-2^(m-1))
    r=1;
    x=p;
    for q=x+2^(m-1):1:(x+2^m)-1
        b(p)=a(p)+a(q)*wn(r);
        b(q)=a(p)-a(q)*wn(r);
        p=p+1;
        r=r+1;
    end
```

```
end
end
return;
```

……

```
 a=[1 2 2 2]
 b=ditfft(a)
b=[7    -1    -1    -1]
```

15.IFFT USING DIT

```
function a =idftdit(a)
a=conj(a);
a=ditfft(a);
a=(a/length(a));
a=conj(a);
return;
```

```
 a=[7 -1 -1 -1];
b=idftdit(a)

     b= 1     2     2     2
```

16. DESIGN OF BUTTERWORTH FILTER USING IMPULSEINVARIENT METHOD

```
function [b,a] =butteriit(wp,ws,ap,as,t);

p=(10^(0.1*ap)-1)^0.5;

s=(10^(.1*as)-1)^0.5;

n=ceil(log(p/s)/log(wp/ws));

wc=wp/(p^(1/n)*t);

[u,v]=butter(n,wc,'s');

[r,p,k]=residue(u,v);

p=exp(p*t);

[b,a]=residue(r,p,k)
```

```
[b,a] =butteriit(.2*pi,.3*pi,7,16,1)

 a =

    1.0000   -2.0233    1.4675   -0.3690


 b =

         0    0.0438    0.0314
```

17.DESIGN OF LPF USING HAMMING WINDOW

```
Function   hamm(wp,ws)
T=ws-wp;
wc=ws-wp;
wc=(ws+wp)/2;
M=ceil(6.6*pi/t)+1;
hd=ideal-lp(wc,M);
function hd=ideal-lp(wc,M)
x=(M-1)/2;
n=[0:M-1];
m=n-x+eps;
hd=sine(wc*m)./(pi*m);
for i=1:m
w-ham(1)=0.54-.46*cos(2*pi*()i-1)/M;
end
M=hd.*w-ham;
subplot(2,2,1);stem(h,hd);titll('Ideal Imp
Responce');
axis([0 M-1-.1 0.4]);Xlable('n');Ylble('hd(n));
subplot(2,2,2);steam(n,w-ham);
title('Hamming window')
axis([0 M-1 0 1.1);Xlable('n');Ylable('hd(n));
subplot(2,2,3);stem(n,H);title('Actual Imp
Responce')
axis([0 N-1 .1 .4]);Xlable('n');
Y lable('H(n)');
```