HACETTEPE UNIVERSITY
ELECTRICAL AND ELECTRONICS ENGINEERING
ELE227 FUNDAMENTALS OF DIGITAL SYSTEMS

PRELIMINARY WORK 5

EGEMEN CAN AYDUĞAN - 21728036

1.A) Universal Shift Register

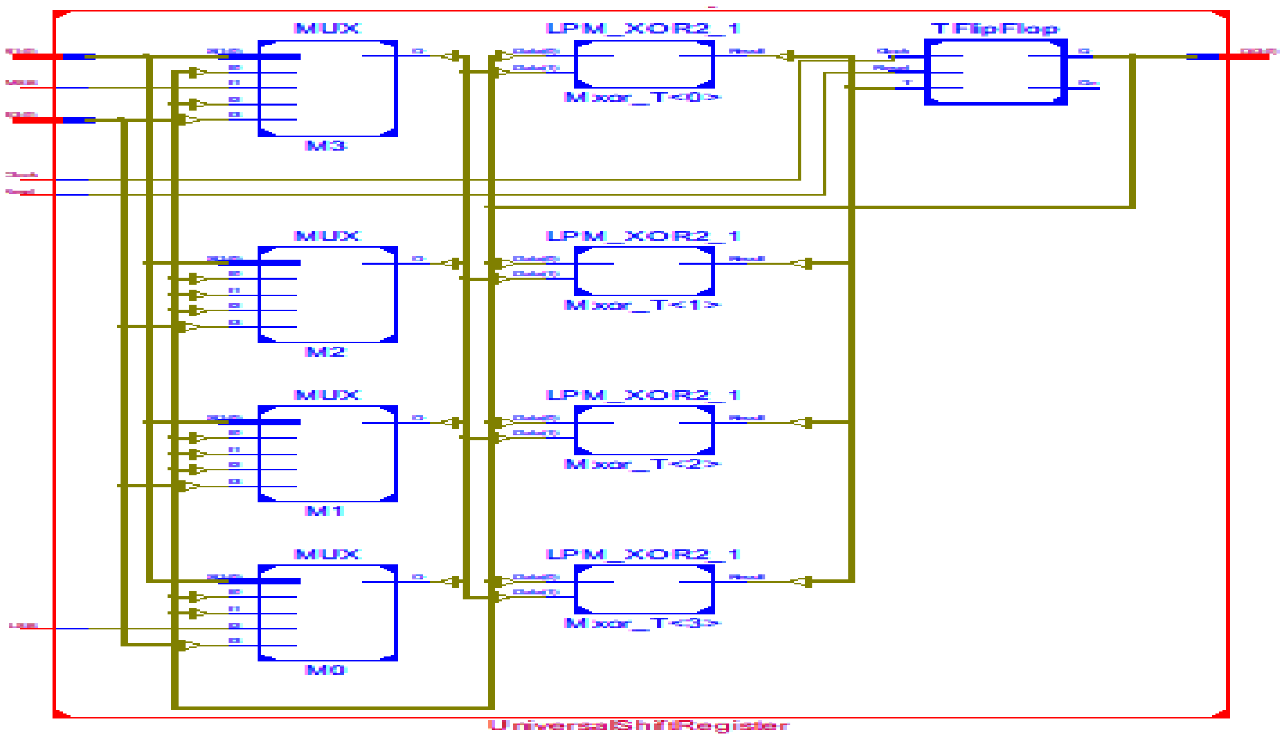
Circuit Diagram

VHDL Implementation:

```
entity UniversalShiftRegister is
    Port ( I : in  STD_LOGIC_VECTOR (3
downto 0);
           S : in  STD_LOGIC_VECTOR (1
downto 0);
           MSB : in  STD_LOGIC;
           LSB : in  STD_LOGIC;
           Clock : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           O : out  STD_LOGIC_VECTOR (3
downto 0));
end UniversalShiftRegister;

architecture Behavioral of
UniversalShiftRegister is

component TFlipFlop
    Port ( T : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           Clock : in  STD_LOGIC;
           Q : out  STD_LOGIC;
           Qn : out  STD_LOGIC);
end component;

component MUX
    Port ( I3 : in  STD_LOGIC;
              I2 : in  STD_LOGIC;
              I1 : in  STD_LOGIC;
              I0 : in  STD_LOGIC;
           S : in  STD_LOGIC_VECTOR (1
downto 0);
           O : out  STD_LOGIC);
end component;
```

```
signal A : STD_LOGIC_VECTOR (3 downto
0);
signal L : STD_LOGIC_VECTOR (3 downto
0);
signal T : STD_LOGIC_VECTOR (3 downto
0);

begin
M3 : MUX port
map(I(3),A(2),MSB,A(3),S,L(3));
M2 : MUX port
map(I(2),A(1),A(3),A(2),S,L(2));
M1 : MUX port
map(I(1),A(0),A(2),A(1),S,L(1));
M0 : MUX port
map(I(0),LSB,A(1),A(0),S,L(0));

T(3) <= L(3) xor A(3);
T(2) <= L(2) xor A(2);
T(1) <= L(1) xor A(1);
T(0) <= L(0) xor A(0);

FF3 : TFlipFlop port
map(T(3),Reset,Clock,A(3));
FF2 : TFlipFlop port
map(T(2),Reset,Clock,A(2));
FF1 : TFlipFlop port
map(T(1),Reset,Clock,A(1));
FF0 : TFlipFlop port
map(T(0),Reset,Clock,A(0));

O(3) <= A(3);
O(2) <= A(2);
O(1) <= A(1);
O(0) <= A(0);

end Behavioral;
```

RTL Schematic:

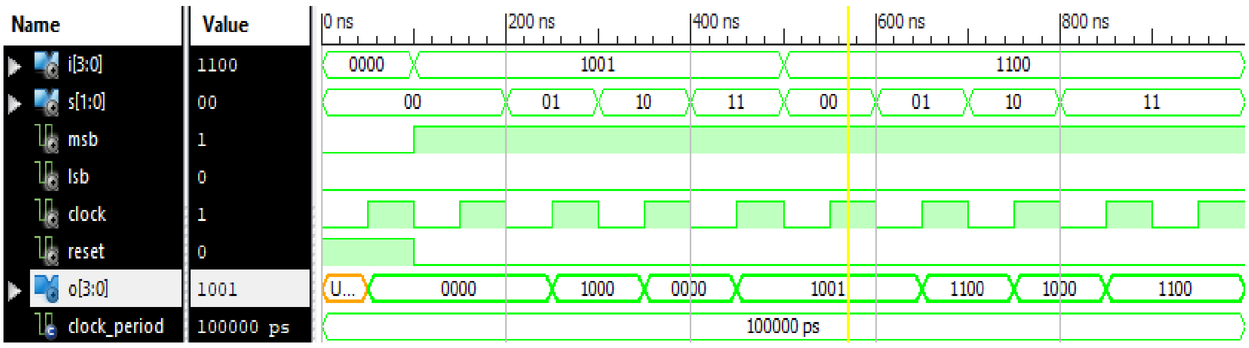


Test Bench Input:

```
signal I : std_logic_vector(3 downto 0)
:= (others => '0');
   signal S : std_logic_vector(1 downto
0) := (others => '0');
   signal MSB : std_logic := '0';
   signal LSB : std_logic := '0';
   signal Clock : std_logic := '0';
   signal Reset : std_logic := '1';


   signal O : std_logic_vector(3 downto
0);

   constant Clock_period : time := 100
ns;

   Clock_process :process
   begin
             Clock <= '0';
             wait for Clock_period/2;
             Clock <= '1';
             wait for Clock_period/2;
   end process;
```

```
 stim_proc: process
  begin
    Reset <= '1';
    wait for 100 ns;
           I <= "1001";
           S <= "00";
           MSB <= '1';
           LSB <= '0';
           Reset <= '0';
           wait for 100 ns;
           I <= "1001";
           S <= "01";
           wait for 100 ns;
           I <= "1001";
           S <= "10";
           wait for 100 ns;
           I <= "1001";
           S <= "11";
           wait for 100 ns;
           I <= "1100";
           S <= "00";
           wait for 100 ns;
           I <= "1100";
           S <= "01";
           wait for 100 ns;
           I <= "1100";
           S <= "10";
           wait for 100 ns;
           I <= "1100";
           S <= "11";
    wait;
 end process;
```

Test Bench Result:



Name	Value
i[3:0]	1100
s[1:0]	00
msb	1
lsb	0
clock	1
reset	0
o[3:0]	1001
clock_period	100000 ps

1.B) UpDown Counter

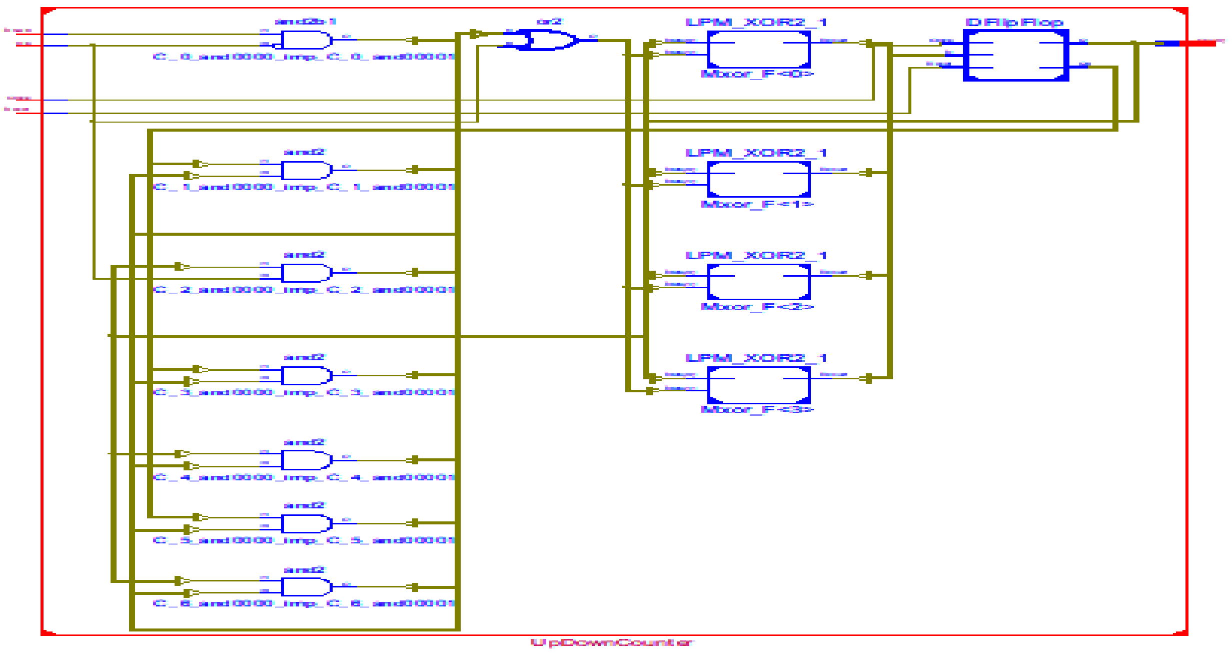
Circuit Diagram

VHDL Implementation:

```
entity UpDownCounter is
    Port ( Up : in  STD_LOGIC;
           Down : in  STD_LOGIC;
           Clock : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           O : out  STD_LOGIC_VECTOR (3
downto 0));
end UpDownCounter;

architecture Behavioral of UpDownCounter
is

component DFlipFlop
    Port ( D : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           Clock : in  STD_LOGIC;
           Q : out  STD_LOGIC;
           Qn : out  STD_LOGIC);
end component;

signal A : STD_LOGIC_VECTOR (3 downto
0);
signal B : STD_LOGIC_VECTOR (3 downto
0);
signal C : STD_LOGIC_VECTOR (6 downto
0);
signal F : STD_LOGIC_VECTOR (3 downto
0);
signal K : STD_LOGIC_VECTOR (3 downto
0);
begin
```

```
C(0) <= (not Up) and Down ;
K(0) <= Up or C(0);
F(0) <= K(0) xor A(0);
FF0 : DFlipFlop port map
(F(0),Reset,Clock,A(0),B(0));

C(1) <= C(0) and B(0);
C(2) <= Up and A(0);
K(1) <= C(1) or C(2);
F(1) <= K(1) xor A(1);
FF1 : DFlipFlop port map
(F(1),Reset,Clock,A(1),B(1));

C(3) <= C(1) and B(1);
C(4) <= C(2) and A(1);
K(2) <= C(3) or C(4);
F(2) <= K(2) xor A(2);
FF2 : DFlipFlop port map
(F(2),Reset,Clock,A(2),B(2));

C(5) <= C(3) and B(2);
C(6) <= C(4) and A(2);
K(3) <= C(5) or C(6);
F(3) <= K(3) xor A(3);
FF3 : DFlipFlop port map
(F(3),Reset,Clock,A(3),B(3));

O(0) <= A(0);
O(1) <= A(1);
O(2) <= A(2);
O(3) <= A(3);
end Behavioral;
```

RTL Schematic:

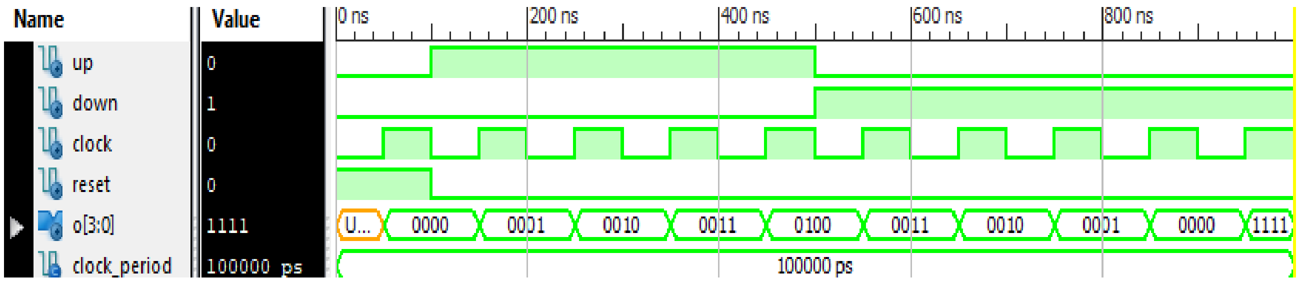

and2b1
C_0_and0000_imp_C_0_and00001
or2
LPM_XOR2_1
Mxor_F<0>
DFlipFlop
and2
C_1_and0000_imp_C_1_and00001
LPM_XOR2_1
Mxor_F<1>
and2
C_2_and0000_imp_C_2_and00001
LPM_XOR2_1
Mxor_F<2>
and2
C_3_and0000_imp_C_3_and00001
LPM_XOR2_1
Mxor_F<3>
and2
C_4_and0000_imp_C_4_and00001
and2
C_5_and0000_imp_C_5_and00001
and2
C_6_and0000_imp_C_6_and00001
UpDownCounter

Test Bench Input:

```
signal Up : std_logic := '0';
   signal Down : std_logic := '0';
   signal Clock : std_logic := '0';
   signal Reset : std_logic := '1';


   signal O : std_logic_vector(3 downto
0);


   constant Clock_period : time := 100
ns;

   Clock_process :process
   begin
             Clock <= '0';
             wait for Clock_period/2;
             Clock <= '1';
             wait for Clock_period/2;
   end process;
```

```
 stim_proc: process
   begin
      Reset <= '1';
      wait for 100 ns;
             Up <= '1';
      Reset <= '0';
      wait for 100 ns;
             Up <= '1';
      wait for 100 ns;
             Up <= '1';
      wait for 100 ns;
             Up <= '1';
      wait for 100 ns;
             Up <= '0';
             Down <= '1';
      wait for 100 ns;
             Up <= '0';
             Down <= '1';
      wait for 100 ns;
             Up <= '0';
             Down <= '1';
      wait for 100 ns;
             Up <= '0';
             Down <= '1';


      wait;
   end process;
```

Test Bench Result:



```
Name          Value         0 ns      200 ns     400 ns     600 ns     800 ns
up            0
down          1
clock         0
reset         0
o[3:0]        1111          U... 0000 0001 0010 0011 0100 0011 0010 0001 0000 1111
clock_period  100000 ps     100000 ps
```

1.C) Johnson Counter

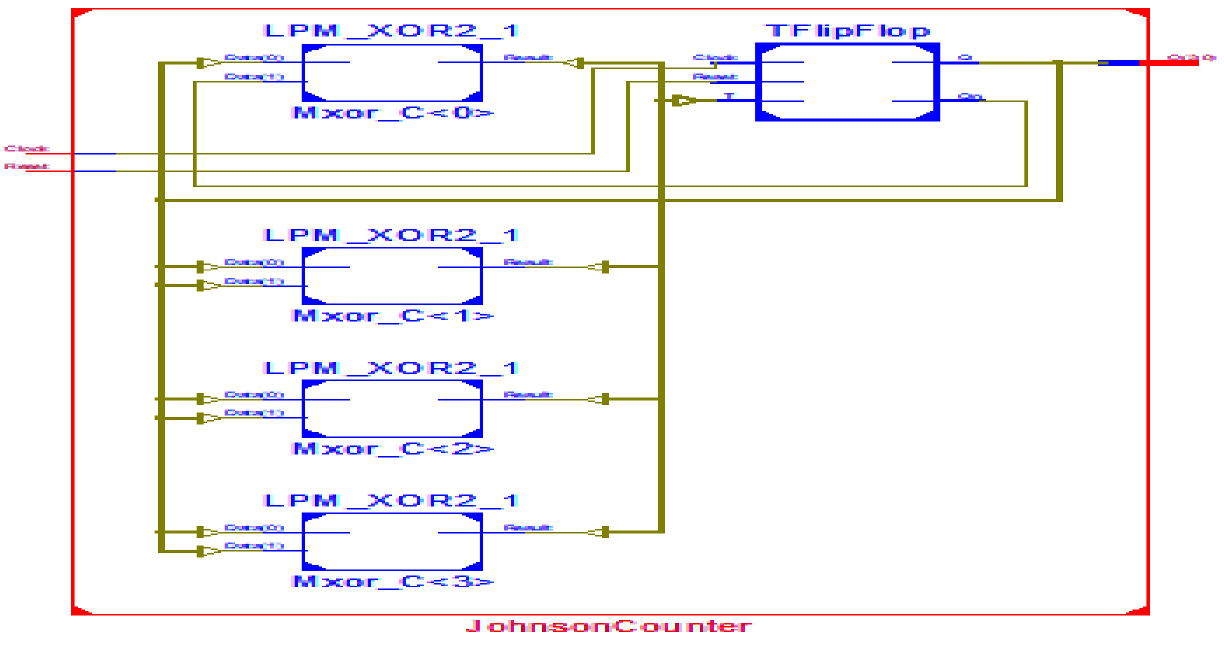
Circuit Diagram

VHDL Implementation

```
entity JohnsonCounter is
    Port ( Clock : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           O : out  STD_LOGIC_VECTOR (3
downto 0));
end JohnsonCounter;

architecture Behavioral of
JohnsonCounter is

component TFlipFlop
    Port ( T : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           Clock : in  STD_LOGIC;
           Q : out  STD_LOGIC;
           Qn : out  STD_LOGIC);
end component;

signal A : STD_LOGIC_VECTOR (3 downto
0);
signal B : STD_LOGIC_VECTOR (3 downto
0);
signal C : STD_LOGIC_VECTOR (3 downto
0);
```

```
begin
C(0) <= B(3) xor A(0);
FF0 : TFlipFlop port
map(C(0),Reset,Clock,A(0),B(0));

C(1) <= A(1) xor A(0);
FF1 : TFlipFlop port
map(C(1),Reset,Clock,A(1),B(1));

C(2) <= A(2) xor A(1);
FF2 : TFlipFlop port
map(C(2),Reset,Clock,A(2),B(2));

C(3) <= A(3) xor A(2);
FF3 : TFlipFlop port
map(C(3),Reset,Clock,A(3),B(3));

O(0) <= A(3);
O(1) <= A(2);
O(2) <= A(1);
O(3) <= A(0);

end Behavioral;
```

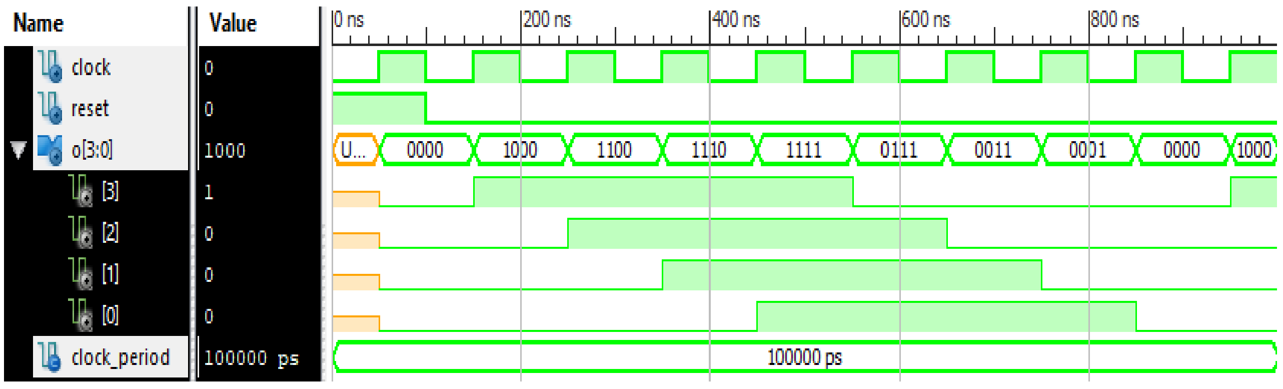
RTL Schematic:



Test Bench Input:

```
   signal Clock : std_logic := '0';
   signal Reset : std_logic := '1';

   signal O : std_logic_vector(3 downto 0);

   constant Clock_period : time := 100 ns;

 Clock_process :process
   begin
             Clock <= '0';
             wait for Clock_period/2;
             Clock <= '1';
             wait for Clock_period/2;
   end process;

   stim_proc: process
   begin
      Reset <= '1';
      wait for 100 ns;
      Reset <= '0';
      wait;
   end process;

END;
```

Test Bench Result:



1.D) Asynchronous Binary Ripple Counter

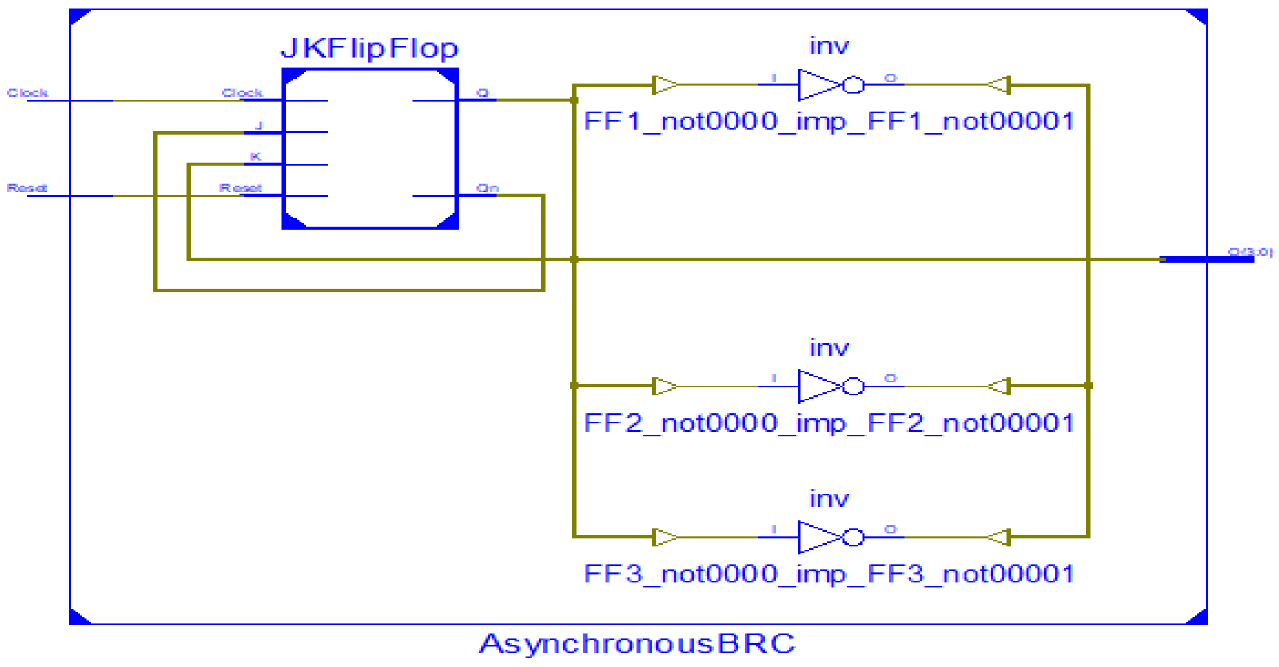
Circuit Diagram

VHDL Implementation:

```
entity AsynchronousBRC is
    Port ( Clock : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           O : out  STD_LOGIC_VECTOR (3
downto 0));
end AsynchronousBRC;

architecture Behavioral of
AsynchronousBRC is

component JKFlipFlop
    Port ( J : in  STD_LOGIC;
           K : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           Clock : in  STD_LOGIC;
           Q : out  STD_LOGIC;
           Qn : out  STD_LOGIC);
end component;

signal A : STD_LOGIC_VECTOR (3 downto
0);
signal B : STD_LOGIC_VECTOR (3 downto
0);
```

```
begin

FF0 : JKFlipFlop port map
(B(0),A(0),Reset,Clock,A(0),B(0));

FF1 : JKFlipFlop port map
(B(1),A(1),Reset,not A(0),A(1),B(1));

FF2 : JKFlipFlop port map
(B(2),A(2),Reset,not A(1),A(2),B(2));

FF3 : JKFlipFlop port map
(B(3),A(3),Reset,not A(2),A(3),B(3));

O(0) <= A(0);
O(1) <= A(1);
O(2) <= A(2);
O(3) <= A(3);

end Behavioral;
```

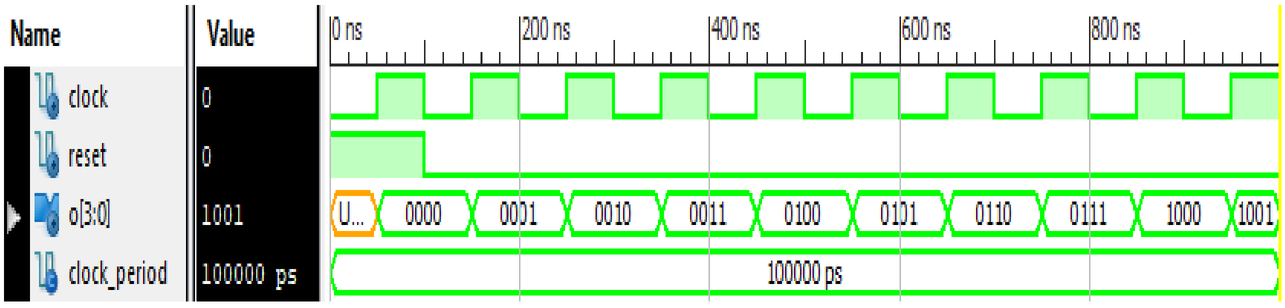
RTL Schematic:


JKFlipFlop
inv
Clock
Clock
J
K
Q
Reset
Reset
Qn
I
O
FF1_not0000_imp_FF1_not00001
O(3:0)
inv
FF2_not0000_imp_FF2_not00001
inv
FF3_not0000_imp_FF3_not00001
AsynchronousBRC

Test Bench Input:

```
   signal Clock : std_logic := '0';
   signal Reset : std_logic := '1';


   signal O : std_logic_vector(3 downto 0);

   constant Clock_period : time := 100 ns;


   Clock_process :process
   begin
            Clock <= '0';
            wait for Clock_period/2;
            Clock <= '1';
            wait for Clock_period/2;
   end process;


   stim_proc: process
   begin
      Reset <= '1';
      wait for 100 ns;
      Reset <= '0';
      wait;
   end process;
END;
```

Test Bench Result:



2.) Binary Divider

Circuit Diagram

VHDL Implementation:

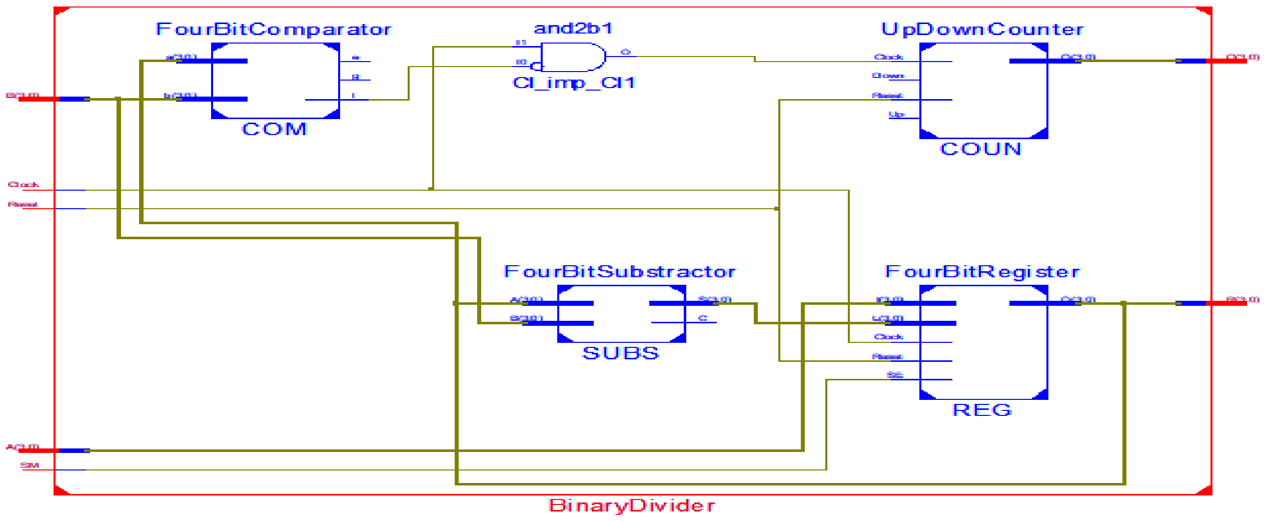
```
entity BinaryDivider is
    Port ( A : in  STD_LOGIC_VECTOR (3
downto 0);
           B : in  STD_LOGIC_VECTOR (3
downto 0);
                      SM: in  STD_LOGIC;
           Clock : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           Q : out  STD_LOGIC_VECTOR (3
downto 0);
           R : out  STD_LOGIC_VECTOR (3
downto 0));
end BinaryDivider;

architecture Behavioral of BinaryDivider
is

component UpDownCounter
    Port ( Up : in  STD_LOGIC;
           Down : in  STD_LOGIC;
           Clock : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           O : out  STD_LOGIC_VECTOR (3
downto 0));
end component;
```

```
component FourBitSubstractor
    Port ( A : in  STD_LOGIC_VECTOR (3
downto 0);
           B : in  STD_LOGIC_VECTOR (3
downto 0);
           S : out  STD_LOGIC_VECTOR
(3 downto 0);
           C : out  STD_LOGIC);
end component;

component FourBitRegister
    Port ( I : in  STD_LOGIC_VECTOR (3
downto 0);
               SE : in  STD_LOGIC;
           L :  in  STD_LOGIC_VECTOR
(3 downto 0);
           Clock : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           O : out  STD_LOGIC_VECTOR
(3 downto 0));
end component;
```

```
component FourBitComparator
    Port ( a : in  STD_LOGIC_VECTOR (3
downto 0);
           b : in  STD_LOGIC_VECTOR (3
downto 0);
           g : out  STD_LOGIC;
           e : out  STD_LOGIC;
           l : out  STD_LOGIC);
end component;

signal GR,EQ,LE : STD_LOGIC;
signal RA : STD_LOGIC_VECTOR (3 downto
0);
signal QAB : STD_LOGIC_VECTOR (3 downto
0);
signal Cl : STD_LOGIC ;
signal x : STD_LOGIC ;
```

```
 begin

COM : FourBitComparator port
map(RA,B,GR,EQ,LE);
Cl <= (not LE) and Clock ;

COUN : UpDownCounter port
map('1','0',Cl,Reset,Q);

REG : FourBitRegister port
map(A,SM,QAB,Clock,Reset,RA);

SUBS : FourBitSubstractor port
map(RA,B,QAB,x);

R <= RA ;

end Behavioral;
```

RTL Schematic:

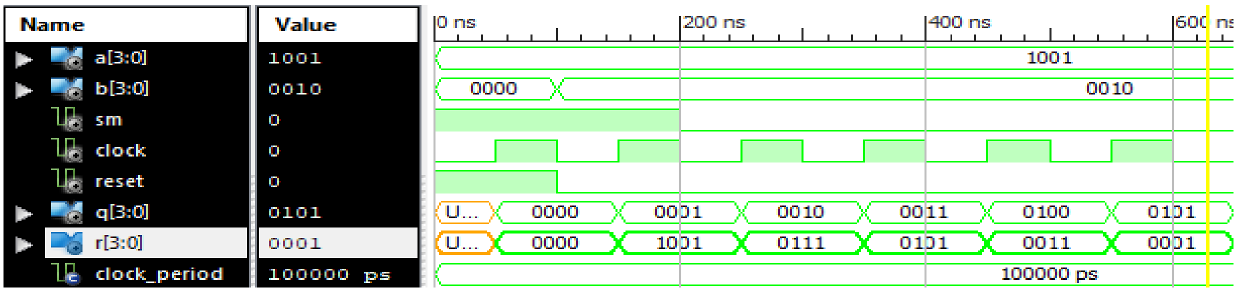


Test Bench Input:

```
signal A : std_logic_vector(3 downto 0)
:= (others => '0');
   signal B : std_logic_vector(3 downto
0) := (others => '0');
   signal SM : std_logic := '0';
   signal Clock : std_logic := '1';
   signal Reset : std_logic := '1';


   signal Q : std_logic_vector(3 downto
0);
   signal R : std_logic_vector(3 downto
0);


   constant Clock_period : time := 100
ns;

   Clock_process :process
   begin
             Clock <= '0';
             wait for Clock_period/2;
             Clock <= '1';
             wait for Clock_period/2;
   end process;
```

```
stim_proc: process
   begin

      A <= "1001";
      Reset <= '1';
      SM <= '1';
      wait for 100 ns;
      A <= "1001";
      B <= "0010";
      Reset <= '0';

      wait for 100 ns ;
      SM <= '0';


      wait;
   end process;

END;
```

Test Bench Result:



Name	Value
a[3:0]	1001
b[3:0]	0010
sm	0
clock	0
reset	0
q[3:0]	0101
r[3:0]	0001
clock_period	100000 ps

4.A) Simple Binary Cell

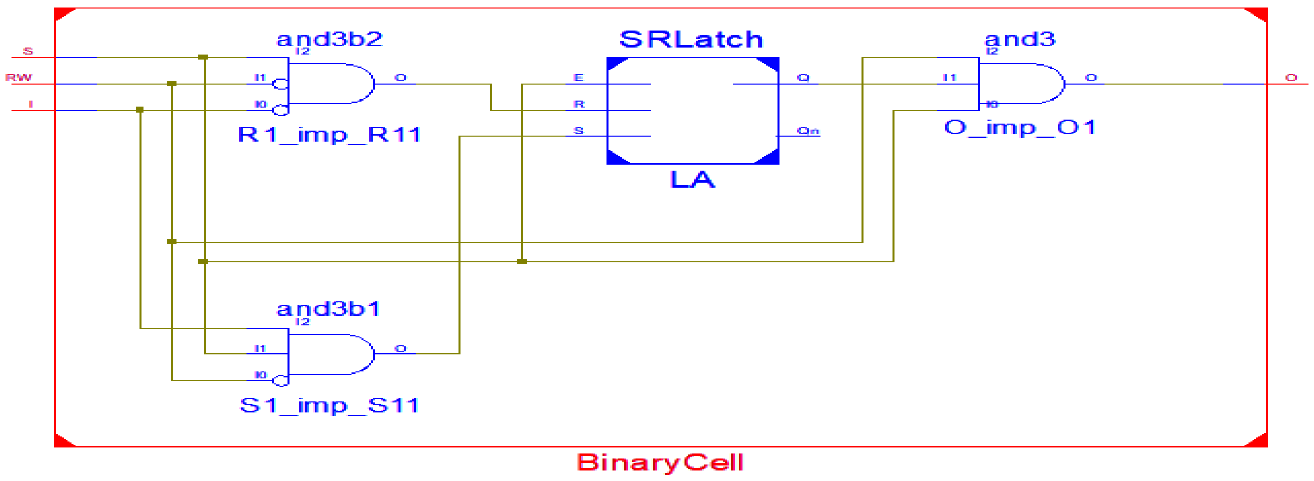
Circuit Diagram

VHDL Implementation:

```
entity BinaryCell is
    Port ( I : in  STD_LOGIC;
           S : in  STD_LOGIC;
           RW : in  STD_LOGIC;
           O : out  STD_LOGIC);
end BinaryCell;

architecture Behavioral of BinaryCell is

component SRLatch
    Port ( S : in  STD_LOGIC;
           R : in  STD_LOGIC;
           E : in  STD_LOGIC;
           Q : out  STD_LOGIC;
           Qn : out  STD_LOGIC);
end component;
```

```
signal S1 : STD_LOGIC;
signal R1 : STD_LOGIC;
signal Q1,Qn1 : STD_LOGIC;

begin
S1 <= I and (not RW) and S ;
R1 <= (not I) and (not RW) and S ;

LA : SRLatch port map(S1,R1,S,Q1,Qn1);

O <= S and Q1 and RW ;

end Behavioral;
```

RTL Schematic :

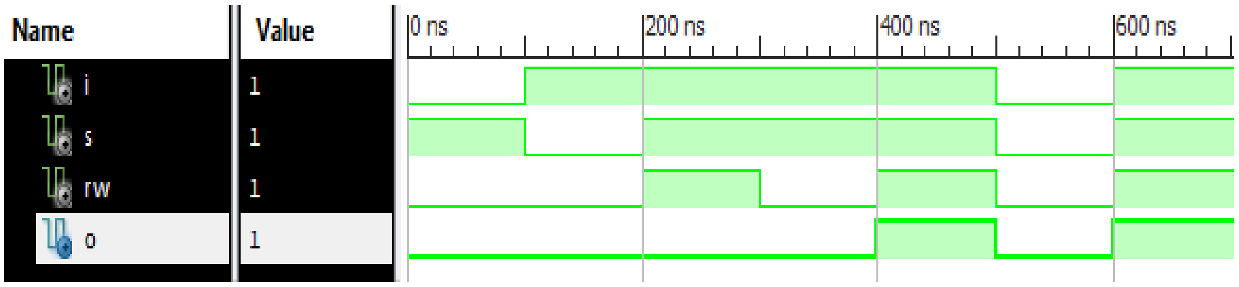


Test Bench Input:

```
stim_proc: process
   begin
      I <= '0';
             S <= '1';
             RW <= '0';
      wait for 100 ns;
             I <= '1';
             S <= '0';
             RW <= '0';
      wait for 100 ns;
             I <= '1';
             S <= '1';
             RW <= '1';
      wait for 100 ns;
             I <= '1';
             S <= '1';
             RW <= '0';
```

```
       wait for 100 ns;
              I <= '1';
              S <= '1';
              RW <= '1';
       wait for 100 ns;
              I <= '0';
              S <= '0';
              RW <= '0';
       wait for 100 ns;
              I <= '1';
              S <= '1';
              RW <= '1';
       wait;
    end process;

END;
```

Test Bench Result:



Name	Value
i	1
s	1
rw	1
o	1

4.B) 4X4 RAM

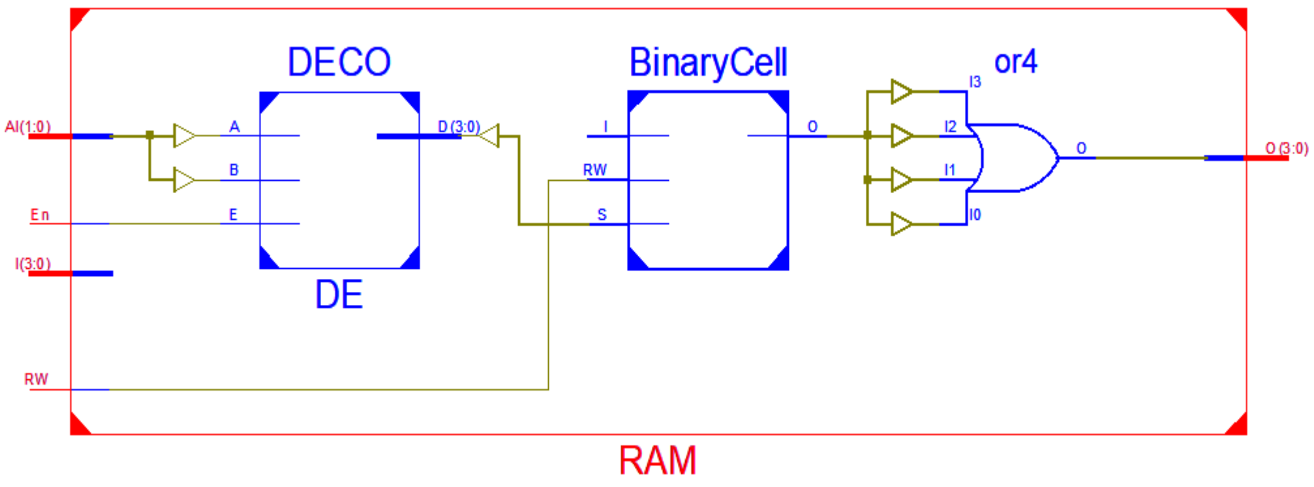
Circuit Diagram

VHDL Implementation:

```vhdl
entity RAM is
    Port ( I : in  STD_LOGIC_VECTOR (3
downto 0);
           AI : in  STD_LOGIC_VECTOR (1
downto 0);
           O : out  STD_LOGIC_VECTOR (3
downto 0);
           En : in  STD_LOGIC;
           RW : in  STD_LOGIC);
end RAM;

architecture Behavioral of RAM is

component DECO
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
          E : in  STD_LOGIC;
           D : out  STD_LOGIC_VECTOR (3
downto 0));
end component;

component BinaryCell
    Port ( I : in  STD_LOGIC;
           S : in  STD_LOGIC;
           RW : in  STD_LOGIC;
           O : out  STD_LOGIC);
end component;

signal W :STD_LOGIC_VECTOR (3 downto 0);
signal T :STD_LOGIC_VECTOR (15 downto 0);
```

```vhdl
begin

DE : DECO port map(AI(1),AI(0),En,W);

BC0 : BinaryCell portmap(I(3),W(0),RW,T(0));
BC1 : BinaryCell portmap(I(2),W(0),RW,T(1));
BC2 : BinaryCell portmap(I(1),W(0),RW,T(2));
BC3 : BinaryCell portmap(I(0),W(0),RW,T(3));

BC4 : BinaryCell portmap(I(3),W(1),RW,T(4));
BC5 : BinaryCell portmap(I(2),W(1),RW,T(5));
BC6 : BinaryCell portmap(I(1),W(1),RW,T(6));
BC7 : BinaryCell portmap(I(0),W(1),RW,T(7));

BC8 : BinaryCell portmap(I(3),W(2),RW,T(8));
BC9 : BinaryCell portmap(I(2),W(2),RW,T(9));
BC10 : BinaryCell port map(I(1),W(2),RW,T(10));
BC11 : BinaryCell port map(I(0),W(2),RW,T(11));

BC12 : BinaryCell port map(I(3),W(3),RW,T(12));
BC13 : BinaryCell port map(I(2),W(3),RW,T(13));
BC14 : BinaryCell port map(I(1),W(3),RW,T(14));
BC15 : BinaryCell port map(I(0),W(3),RW,T(15));

O(3) <= T(0) or T(4) or T(8) or T(12);
O(2) <= T(1) or T(5) or T(9) or T(13);
O(1) <= T(2) or T(6) or T(10) or T(14);
O(0) <= T(3) or T(7) or T(11) or T(15);

end Behavioral;
```

RTL Schematic :

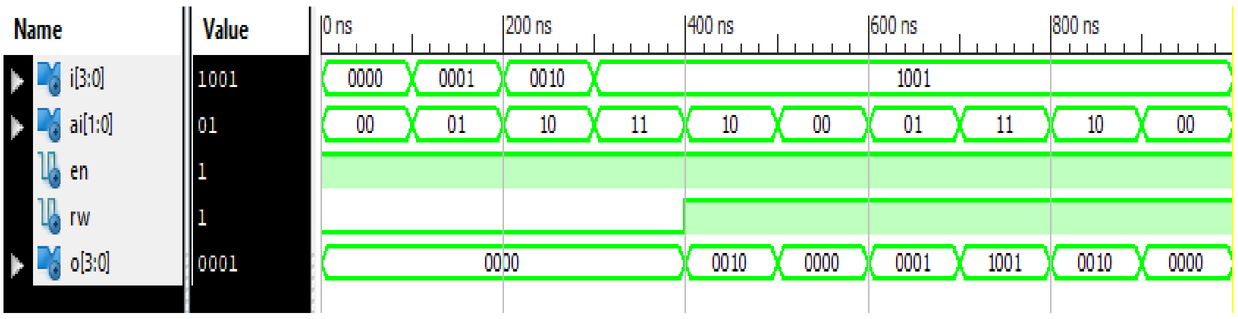


Test Bench Input:

```
stim_proc: process
   begin
             I <="0000";
             AI <= "00";
             En <= '1';
             RW <= '0';
      wait for 100 ns;
             AI <= "01";
             I<="0001";
      wait for 100 ns;
             AI <= "10";
             I<="0010";
      wait for 100 ns;
             AI <= "11";
             I<="1001";
      wait for 100 ns;
             RW <= '1';
             AI<= "10";
      wait for 100 ns;
             AI<= "00";
      wait for 100 ns;
             AI<= "01";
```

```
        wait for 100 ns;
               AI<= "11";
        wait for 100 ns;
               AI<= "10";
        wait for 100 ns;
               AI<= "00";
        wait for 100 ns;
               AI<= "01";
        wait for 100 ns;
               AI<= "11";

       wait;
   end process;

END;
```

Test Bench Result:



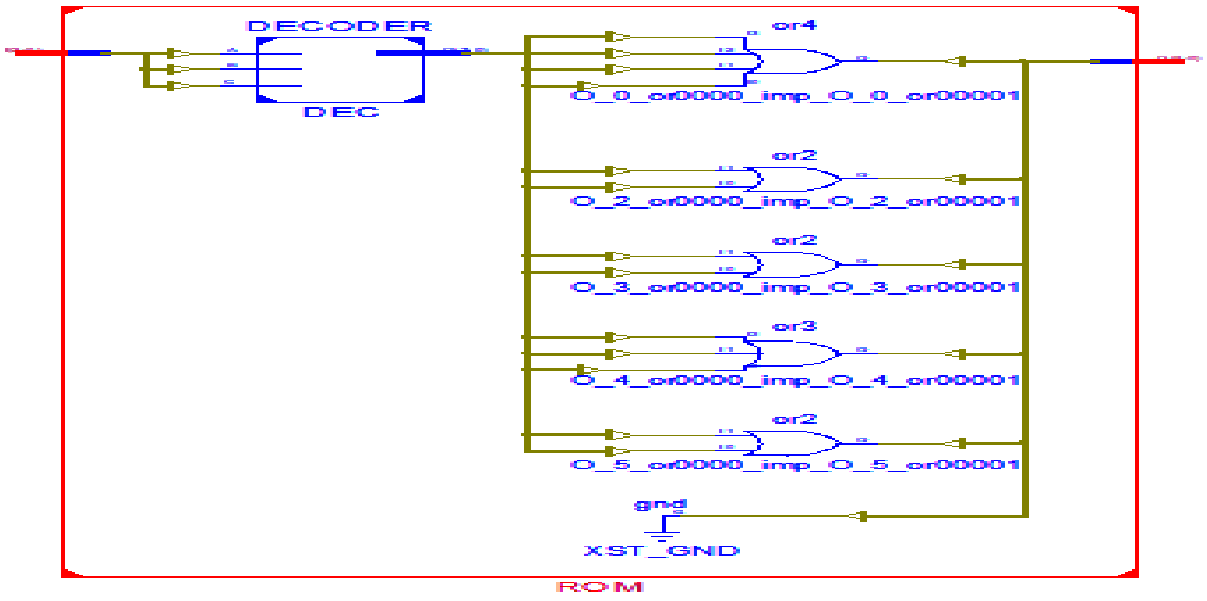
5.) Circuit Diagram

VHDL Implementation:

```
entity ROM is
    Port ( I : in  STD_LOGIC_VECTOR (2 downto
0);
           O : out  STD_LOGIC_VECTOR (5 downto
0));
end ROM;

architecture Behavioral of ROM is

component DECODER
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           C : in  STD_LOGIC;
           D : out  STD_LOGIC_VECTOR (7 downto
0));
end component;
```

```
signal R : STD_LOGIC_VECTOR (7 downto 0);

begin

DEC : DECODER port map(I(2),I(1),I(0),R);

O(0) <= R(1) or R(3) or R(5) or R(7);
O(1) <= '0';
O(2) <= R(2) or R(6);
O(3) <= R(3) or R(5);
O(4) <= R(4) or R(5) or R(7);
O(5) <= R(6) or R(7);

end Behavioral;
```

RTL Schematic:

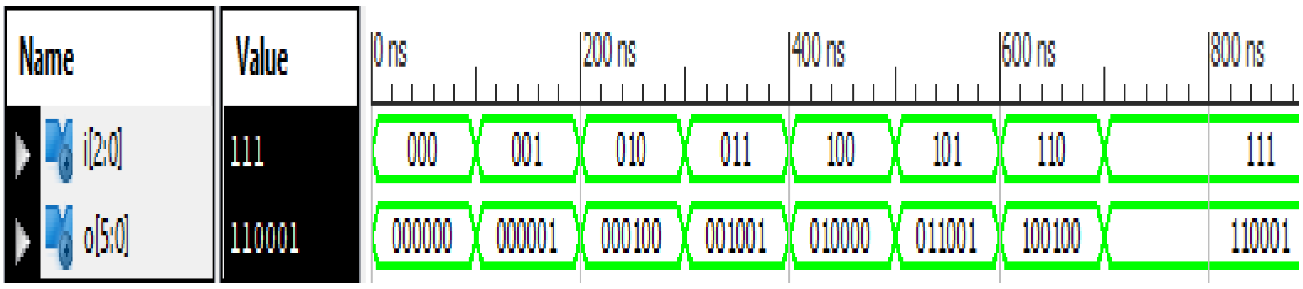

DECODER
DEC
or4
O_0_or0000_imp_O_0_or00001
or2
O_2_or0000_imp_O_2_or00001
or2
O_3_or0000_imp_O_3_or00001
or3
O_4_or0000_imp_O_4_or00001
or2
O_5_or0000_imp_O_5_or00001
gnd
XST_GND
ROM

Test Bench Inputs :

```
stim_proc: process
   begin
      I <= "000";
      wait for 100 ns;
             I <= "001";
      wait for 100 ns;
             I <= "010";
      wait for 100 ns;
             I <= "011";
      wait for 100 ns;
             I <= "100";
```

```
      wait for 100 ns;
             I <= "101";
      wait for 100 ns;
             I <= "110";
      wait for 100 ns;
             I <= "111";

      wait;
   end process;

END;
```

Test Bench Results :


Name
Value
0 ns
200 ns
400 ns
600 ns
800 ns
i[2:0]
o[5:0]