ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

DSD AND DICA LAB MANUAL

III B.Tech - I Sem.

Prepared By

Mr.P.Bhavani sankar,Assistant Professor

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

Affiliated to JNTUK, Kakinada, Approved by AICTE, New Delhi, Accredited by NAAC
Recognized by UGC under Section 2(f) of UGC Act 1956
Aditya Nagar,ADB Road, Surampalem-533437,E.G.Dt.,PH: 0884-2326212,99591 76665

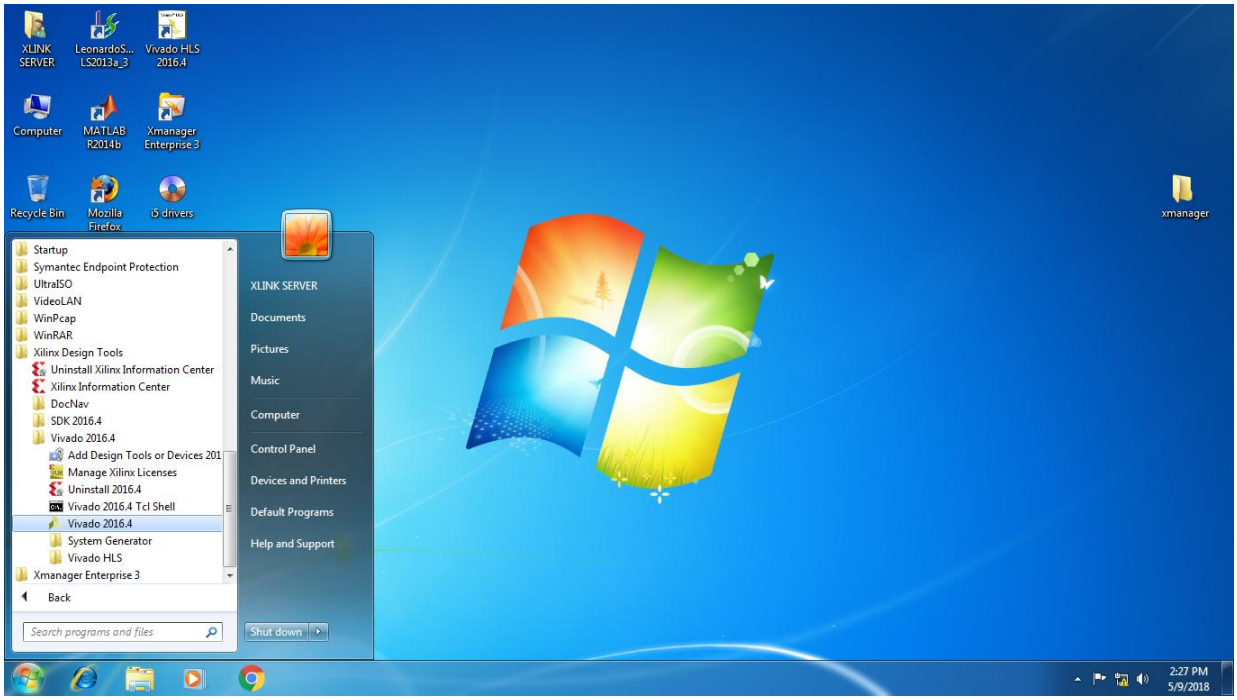
LIST OF EXPERIMENTS

ADDITIONAL EXPERIMENTS

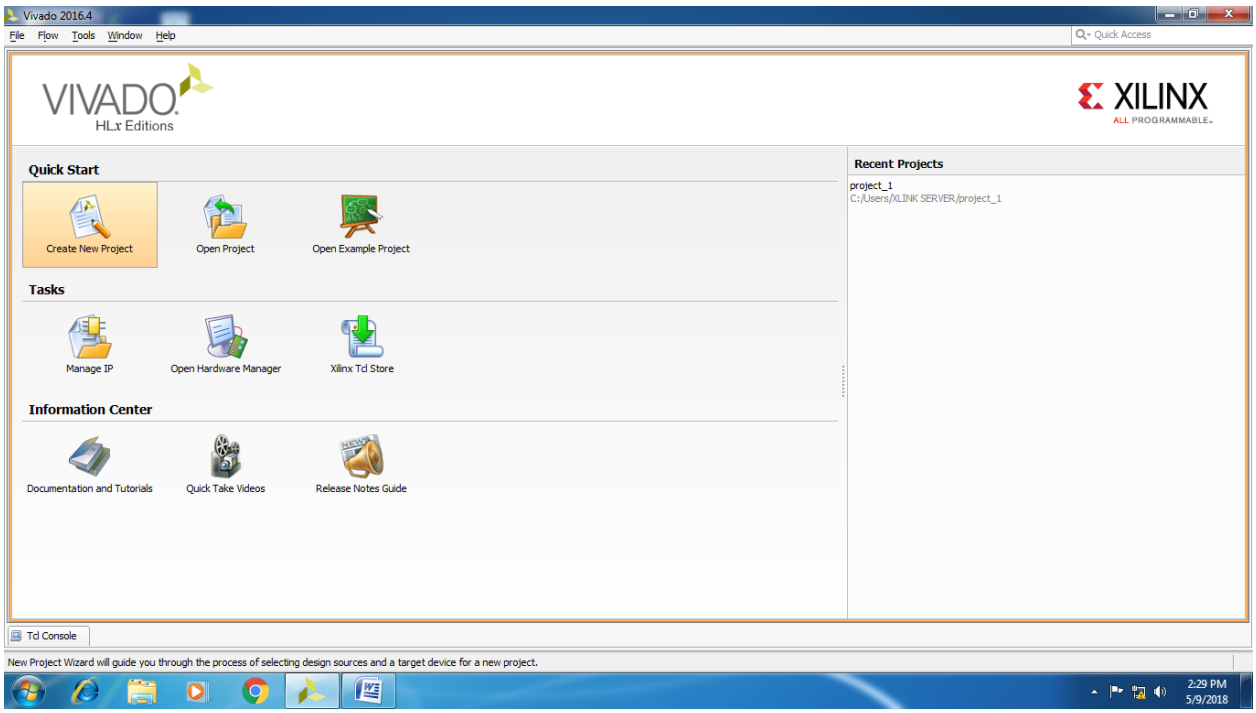
VHDL DESIGN FLOW

Creating a New Project:

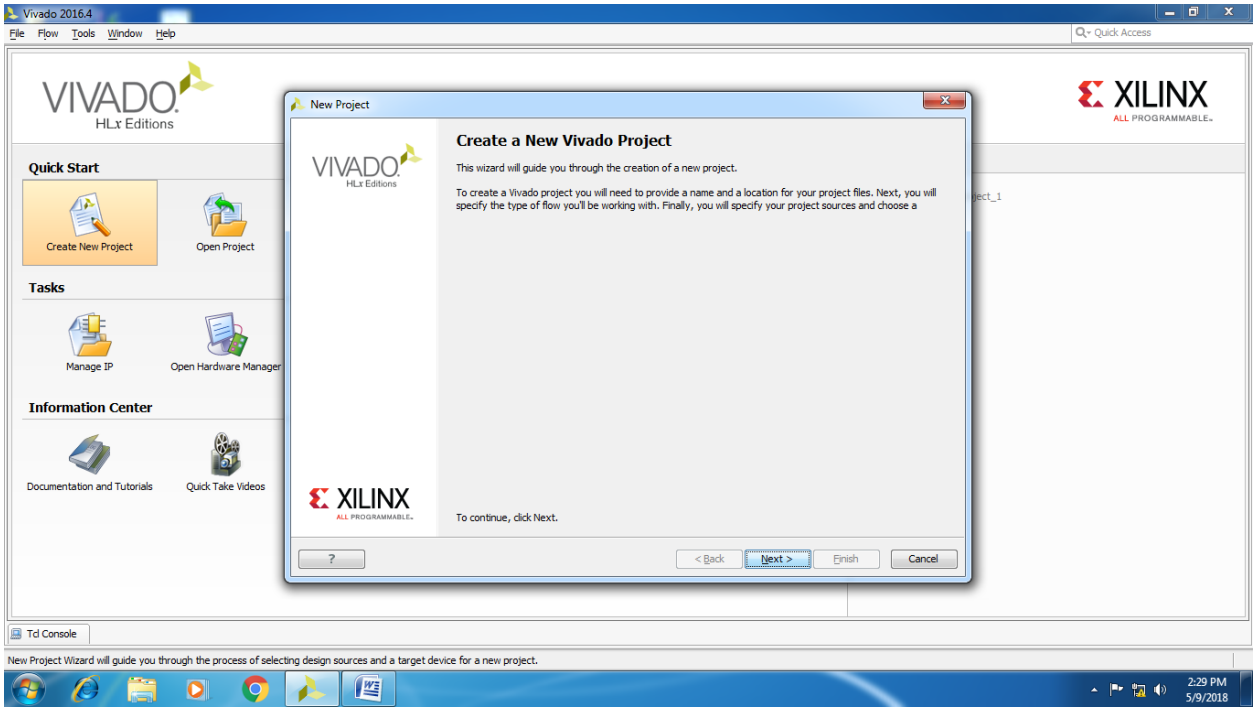
1.Select Vivado 2016.4 as shown below



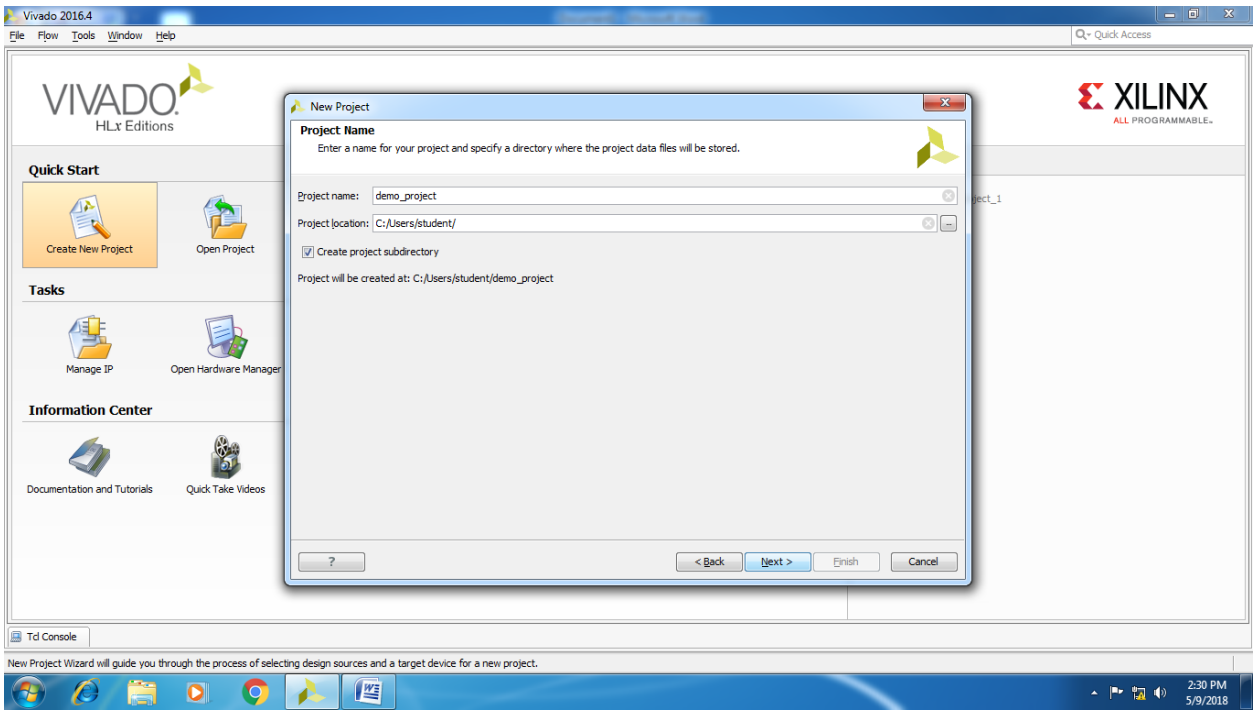
2. Click on 'Create New Project'



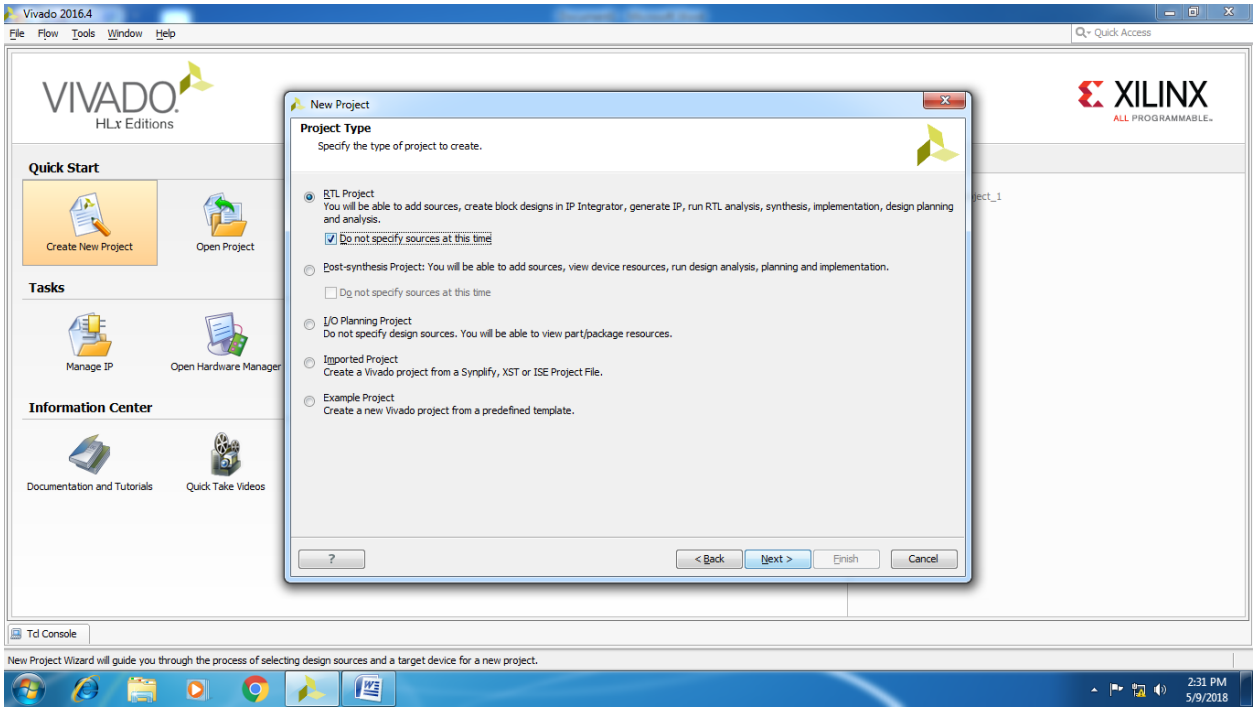
3. Select 'next'



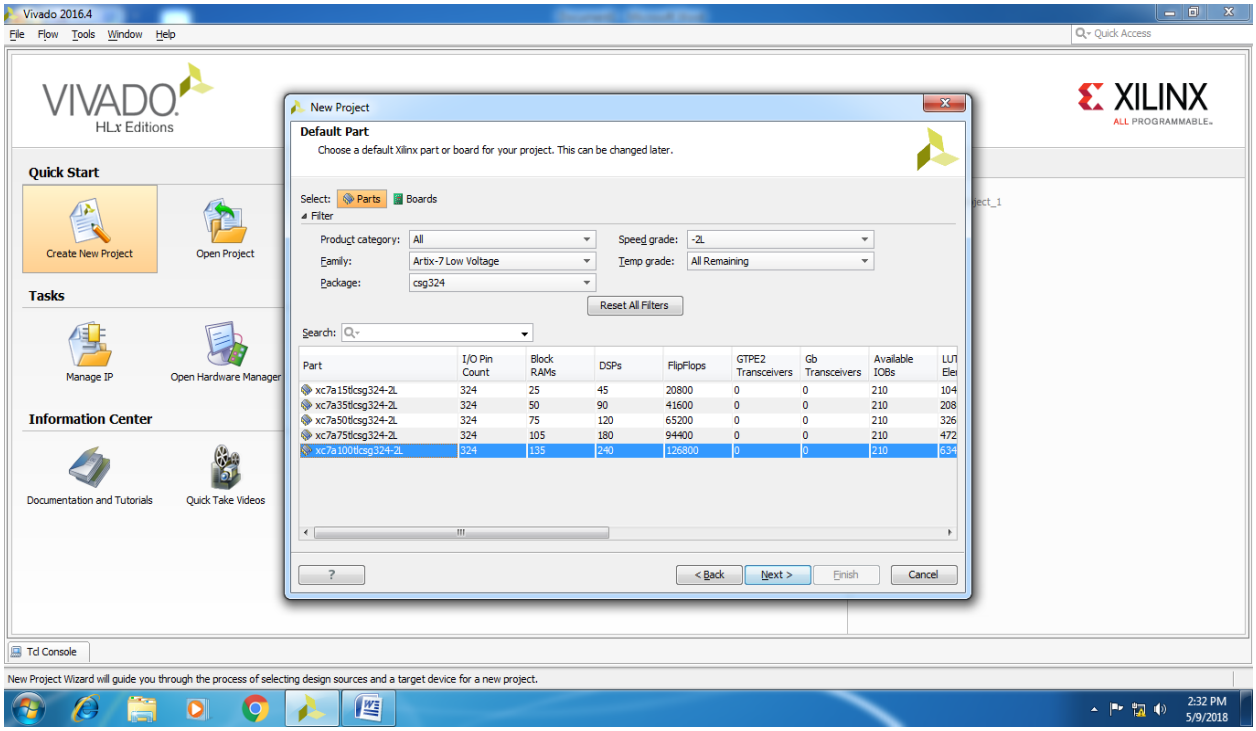
4. Choose appropriate project name and project location



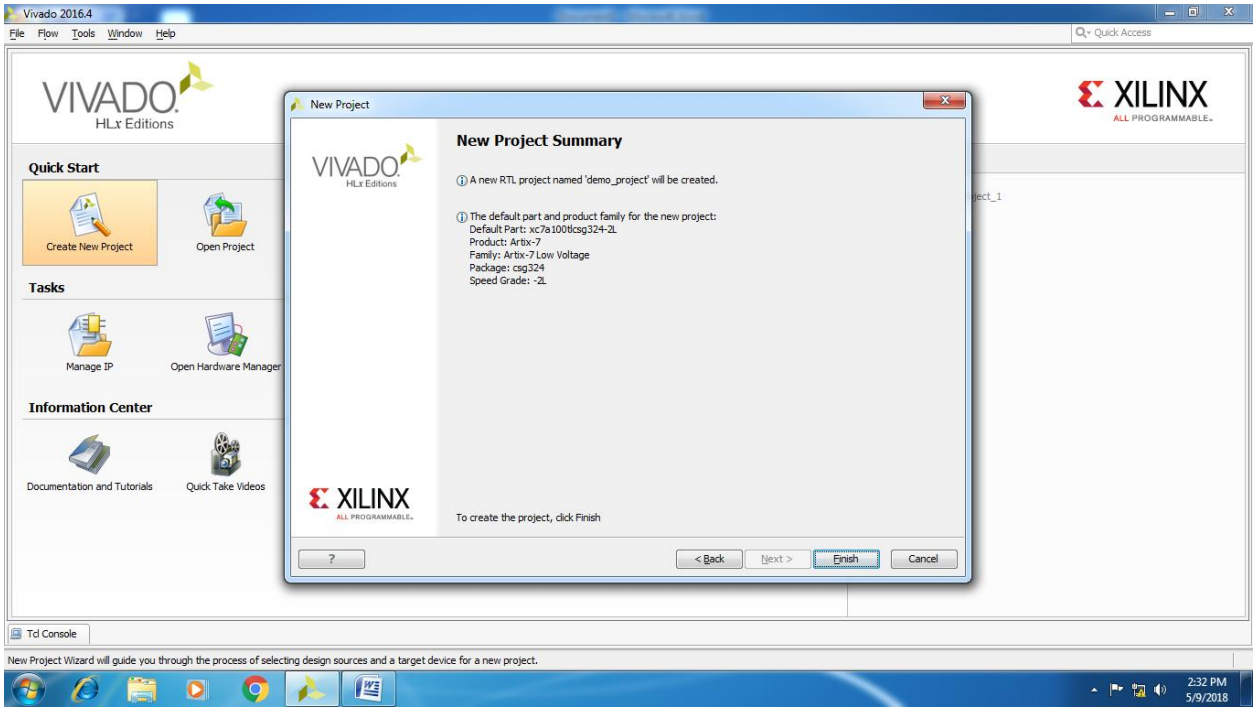
5. select RTL Project and tick on 'Do not specify sources at this time'



6. Make New Project as shown below and select the last row and click Next

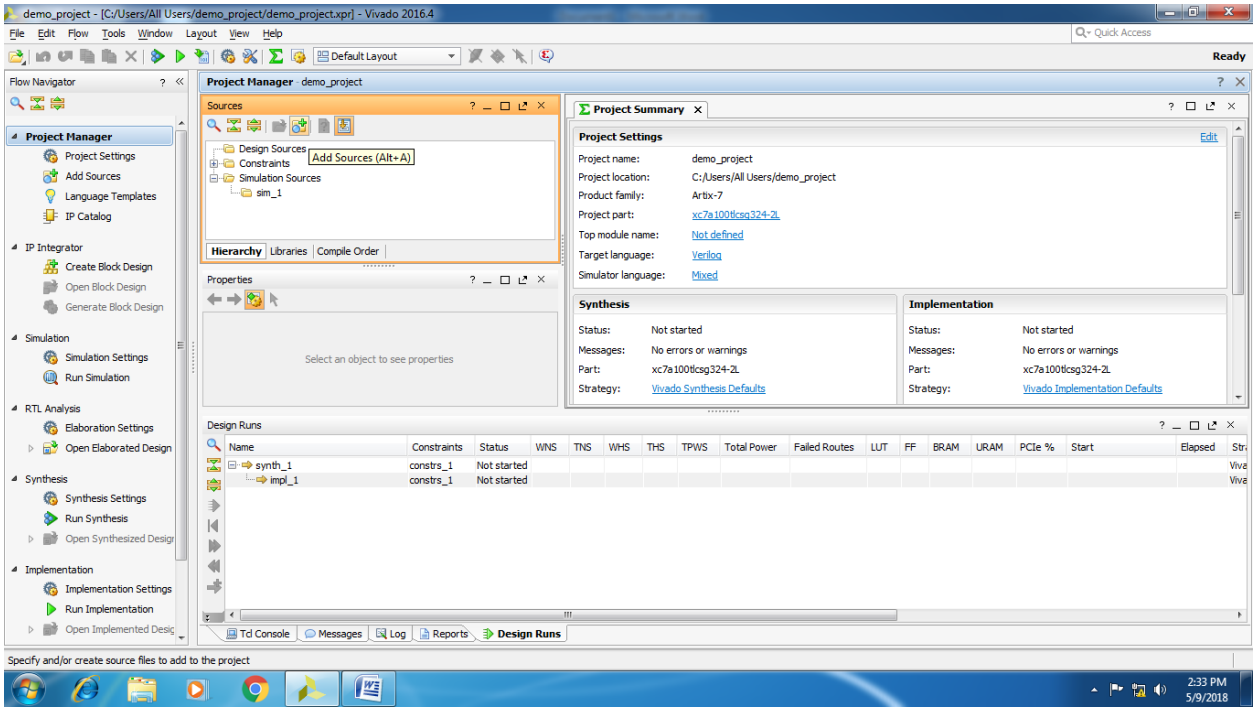


7. Click Finish

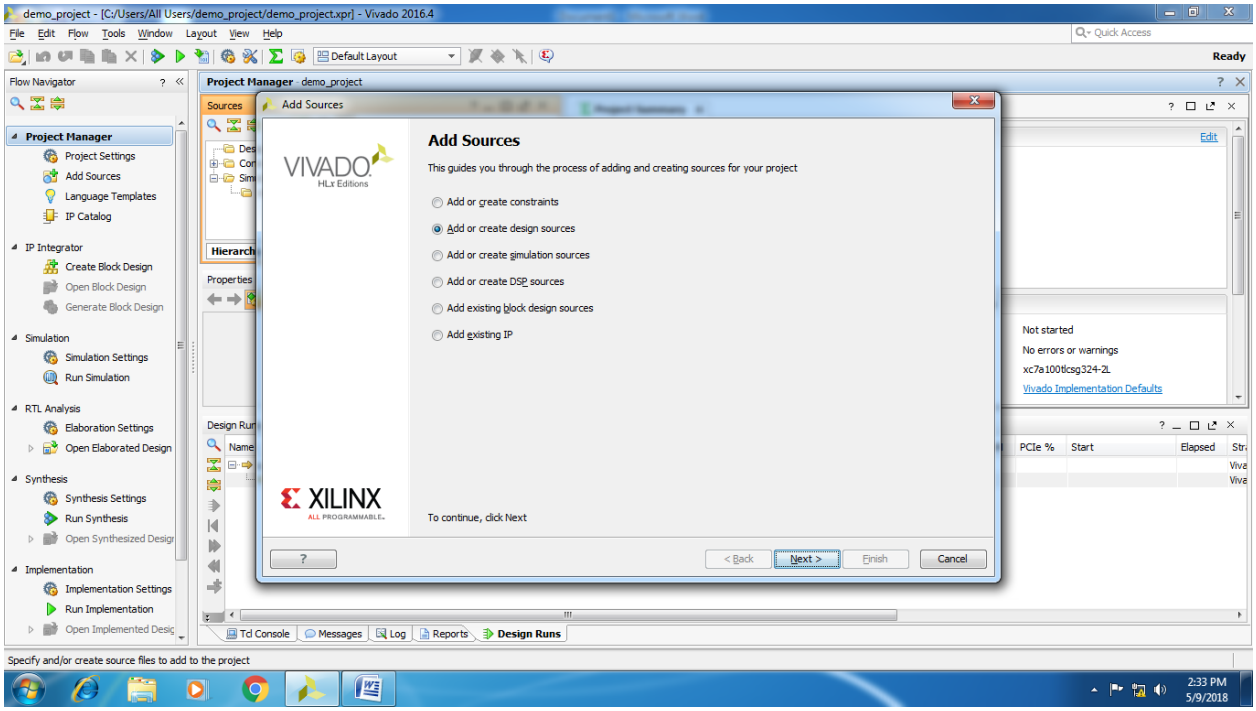


Adding a New VHDL Source:

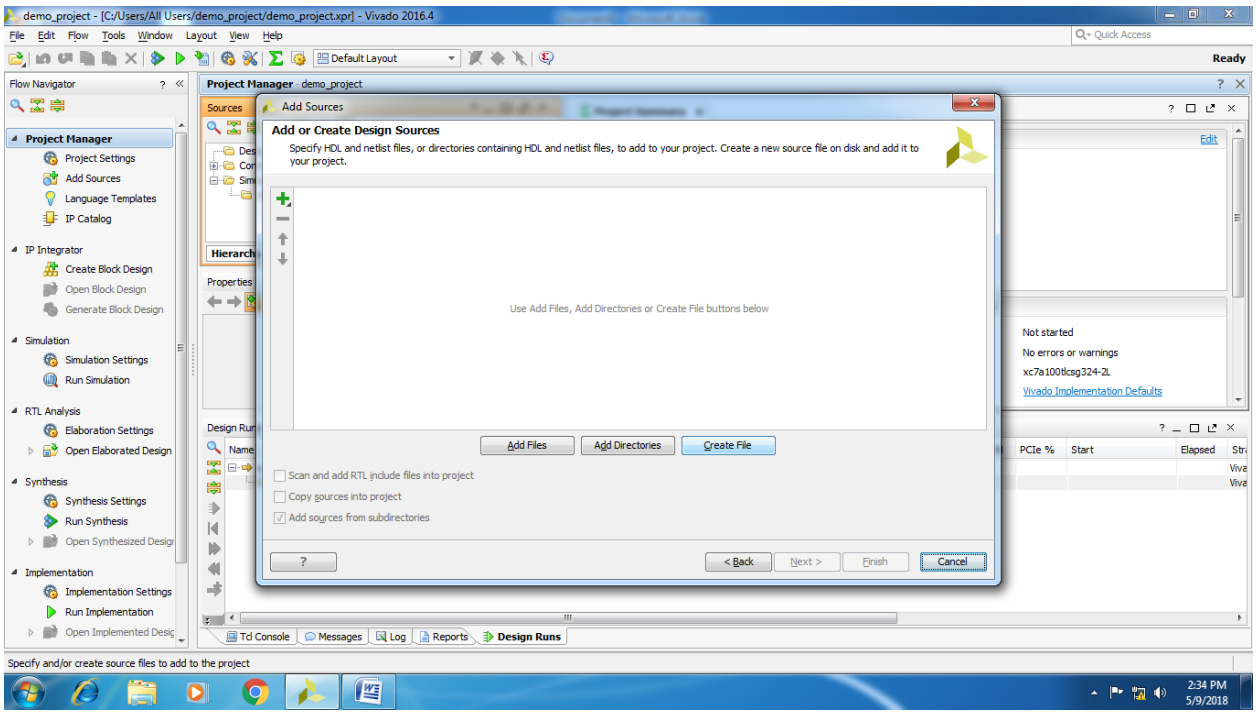
8. Select Add Sources



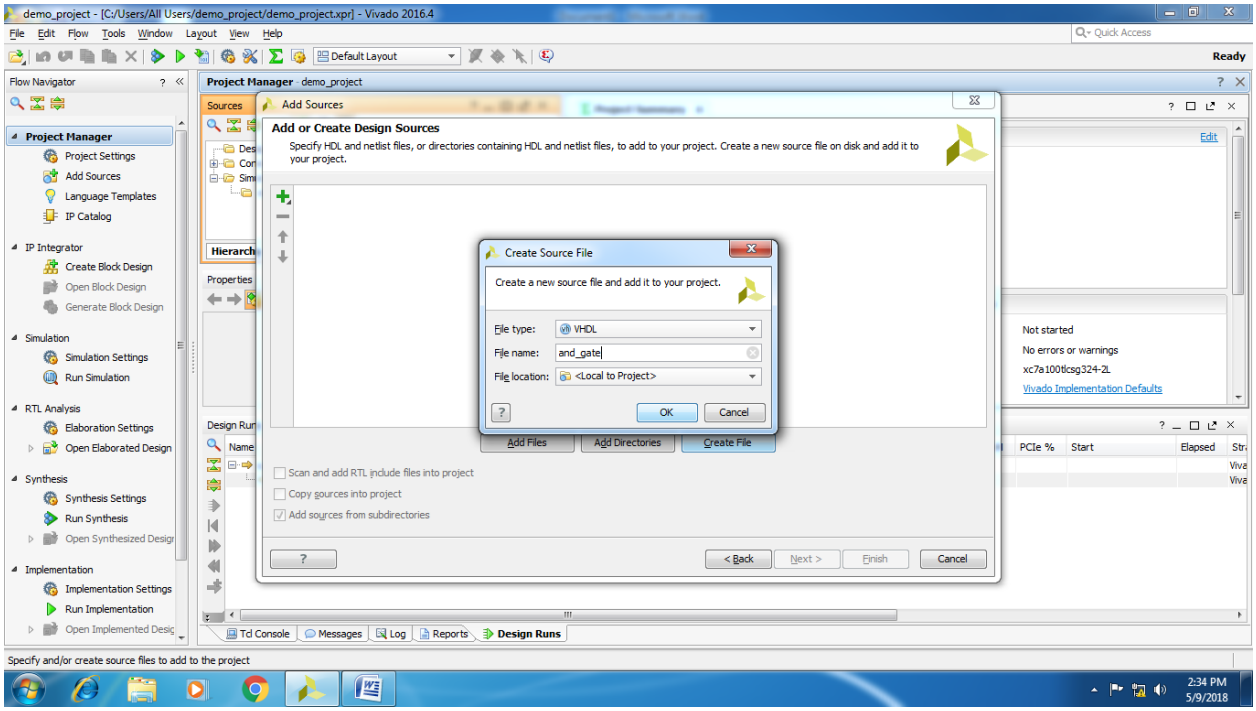
9. Select Add or create design sources and Next



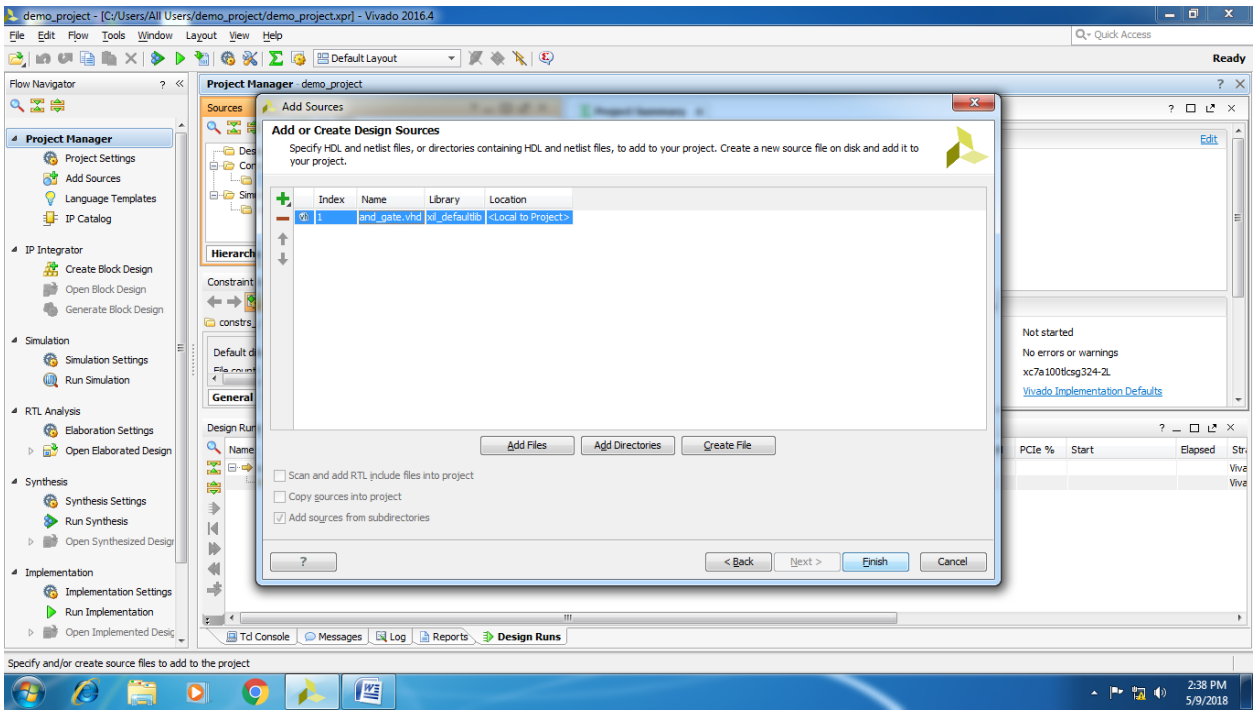
10. Select Create File



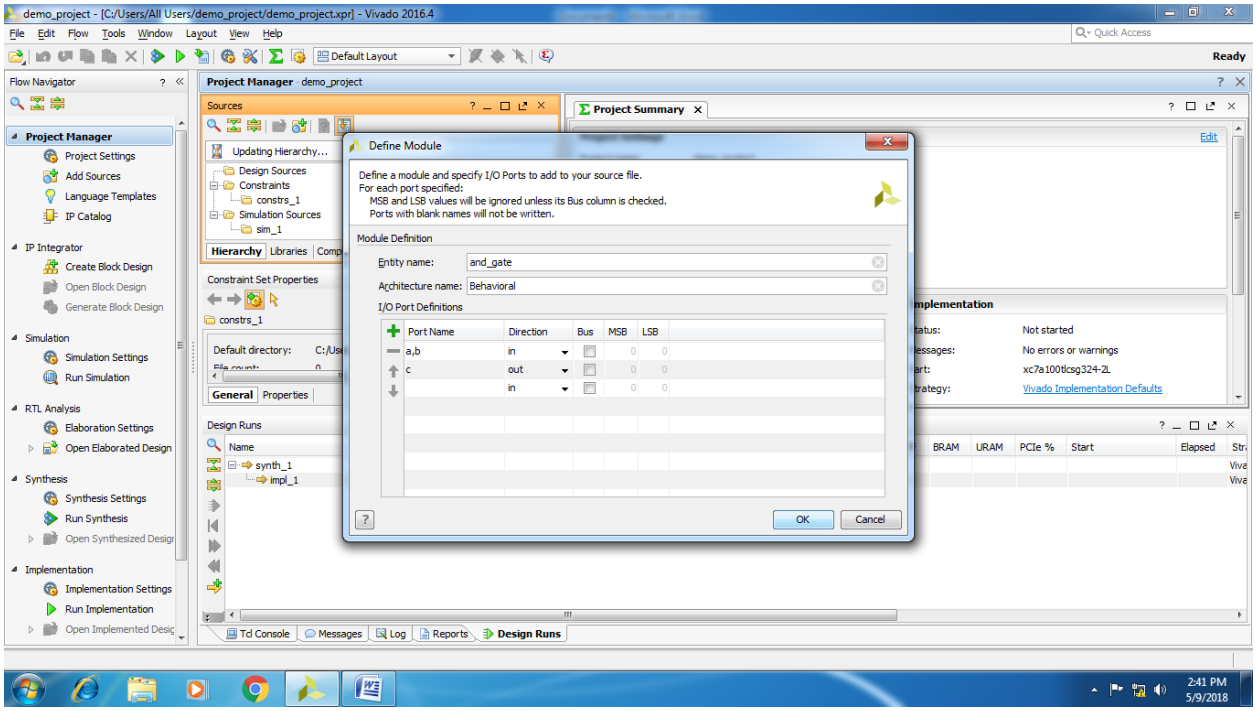
11. Select File type as VHDL and give appropriate 'File name'



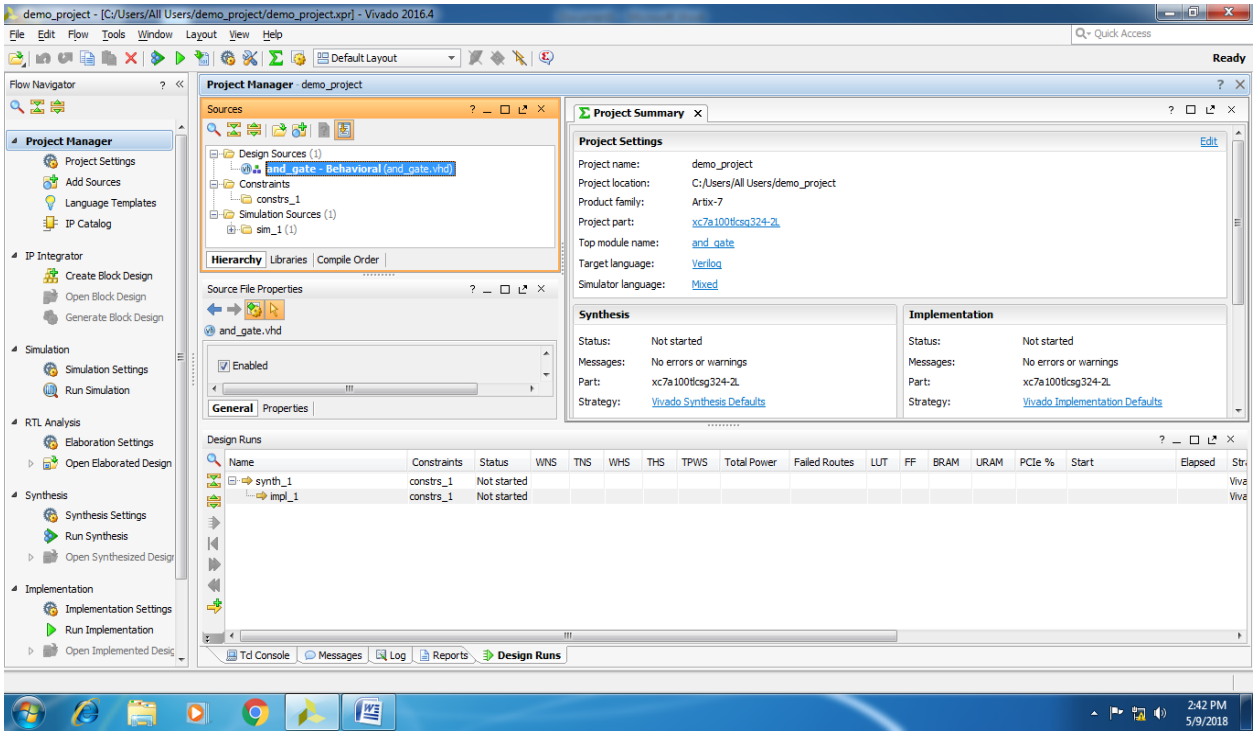
12. Select Finish



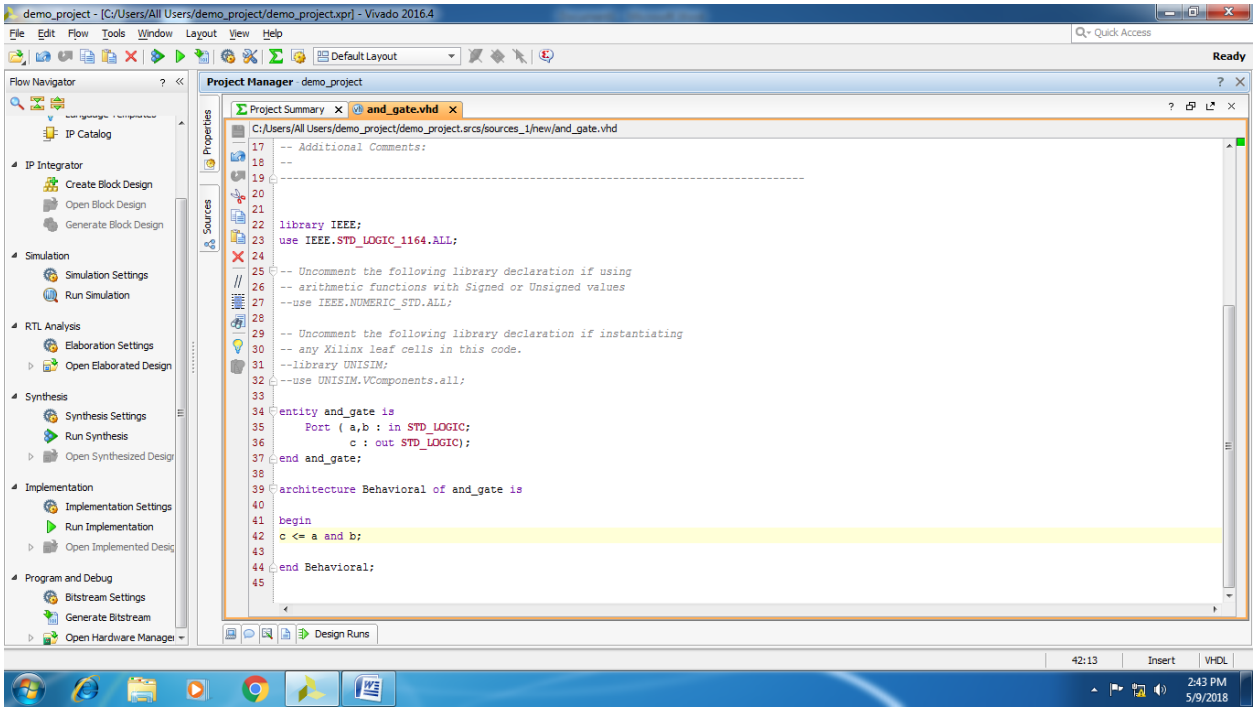
13. Add I/O ports and click OK



14. Double click on file created

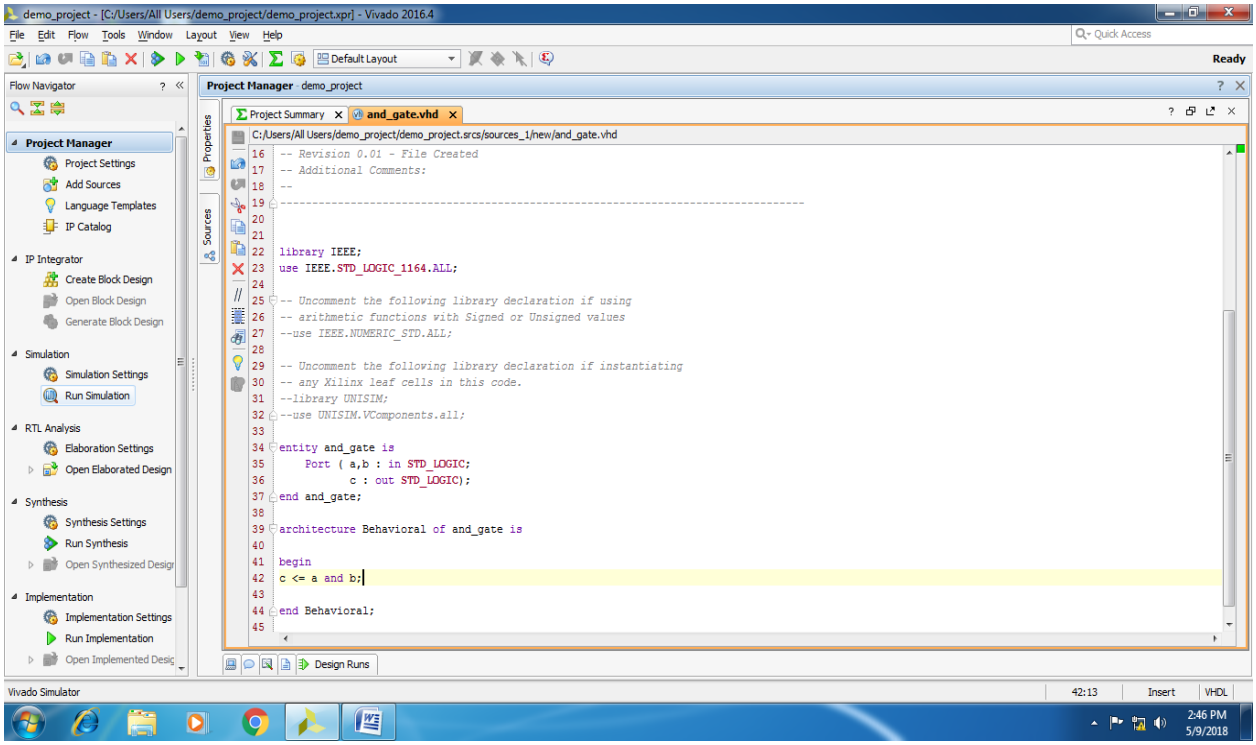


15. Complete the program

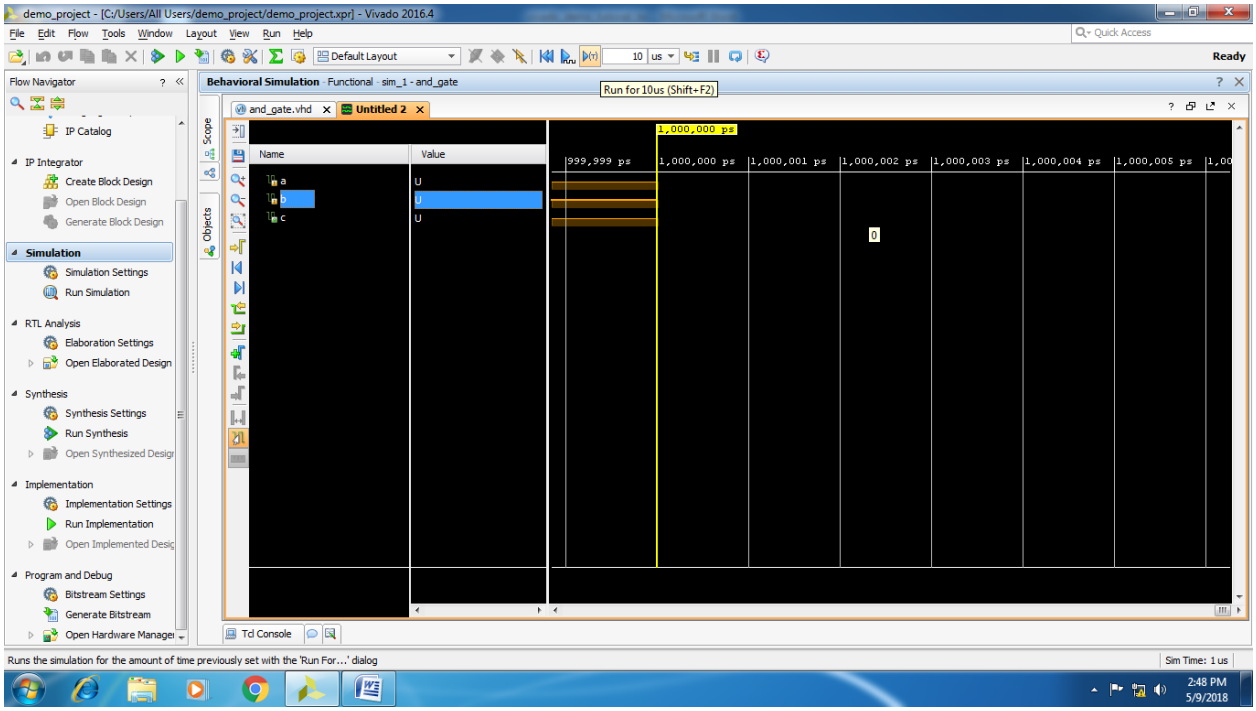


Simulation:

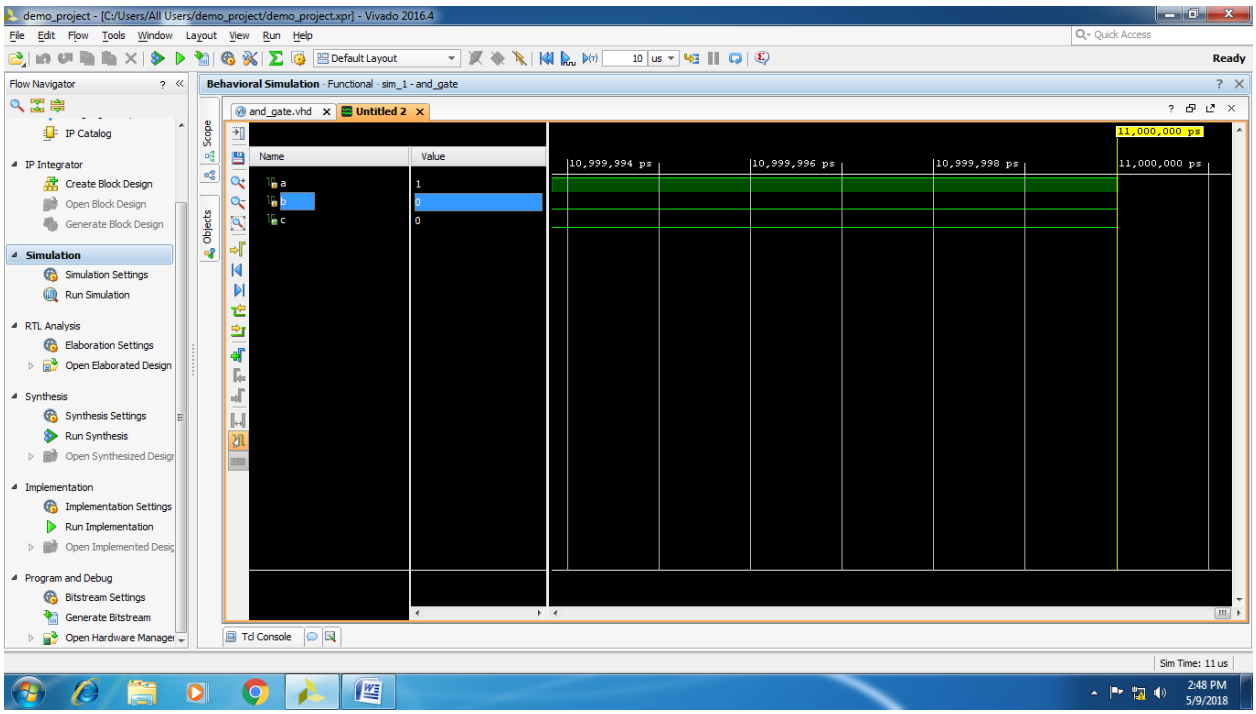
16. Click 'Run Simulation'



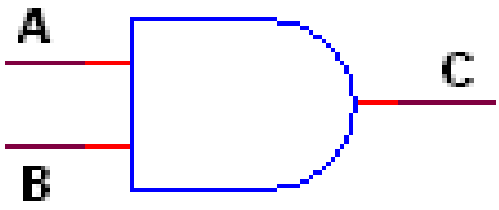
17. Force constant Input ports



18. Run Simulation

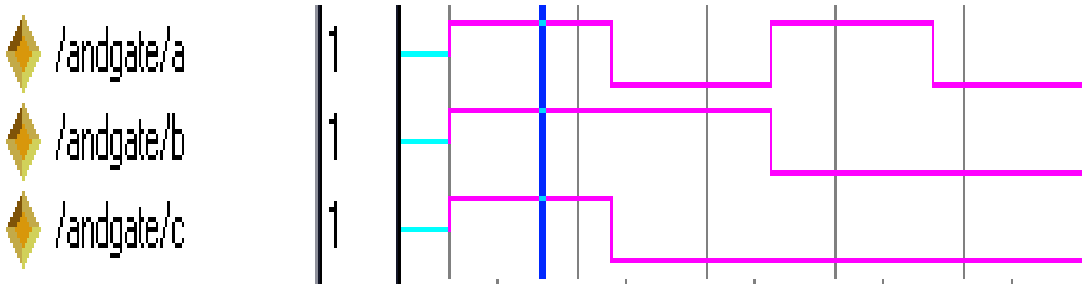


AND GATE



A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

TIMING DIAGRAM:



EXP NO:

Date:

REALIZATION OF LOGIC GATES

ABSTRACT: To study and simulate logic gates using VHDL.

THEORY: The AND_GATE performs the logical AND of two variables A and B produces an output C.

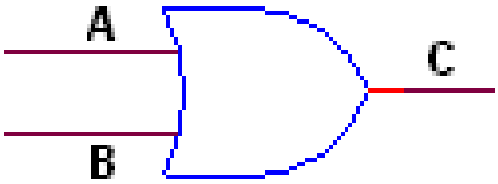
PROCEDURE:

- The AND gate Design is entered through VHDL.
- The design is simulated by applying test vectors-a,b and observe the output c.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR AND GATE:

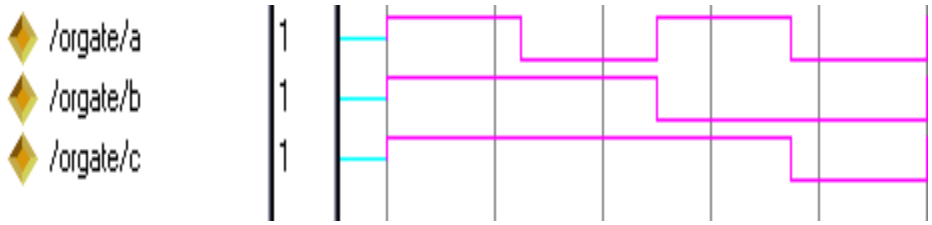
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity andgate is

   Port ( a,b : in  STD_LOGIC;

          c : out  STD_LOGIC);

end andgate;

architecture Behavioral of andgate is

begin

c<= a and b;

end Behavioral;
```

OR GATE



A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

TIMING DIAGRAM:



OR GATE

THEORY: The OR_GATE performs the logical OR of two variables A and B produces an output C.

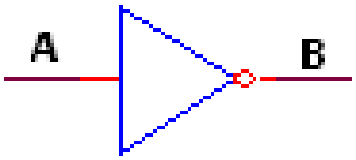
PROCEDURE:

- The OR gate Design is entered through VHDL.
- The design is simulated by applying test vectors-a,b and observe the output c.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR OR GATE:

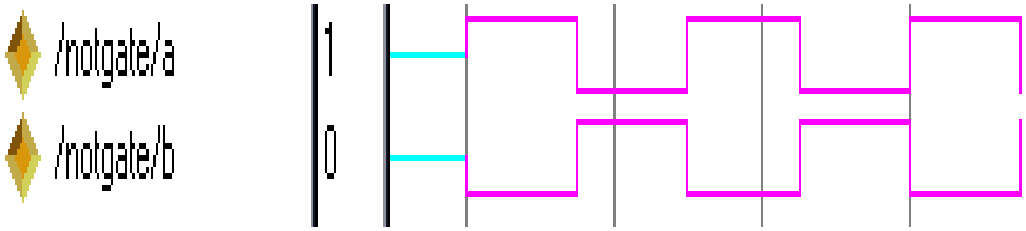
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;



entity orgate is

   Port ( a,b : in  STD_LOGIC;

          c : out  STD_LOGIC);

end orgate;



architecture Behavioral of orgate is

begin

c<= a or b;

end Behavioral;
```

NOT GATE



A	B
0	1
1	0

TIMING DIAGRAM:



NOT_GATE

THEORY: The NOT_GATE performs inversion.

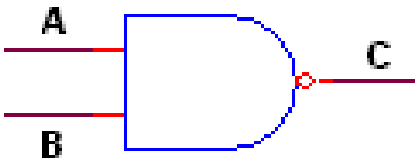
PROCEDURE:

- The NOT Design is entered through VHDL.
- The design is simulated by applying test vectors-a and observe the output b.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR NOT GATE:

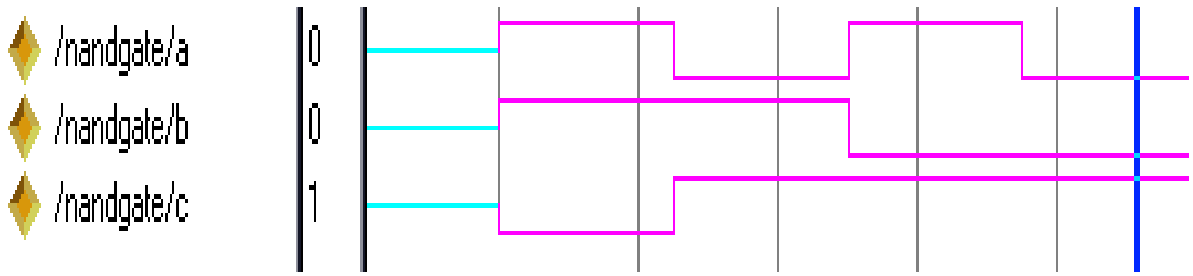
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity notgate is

Port ( a : in  STD_LOGIC;

       b : out  STD_LOGIC);

end notgate;

architecture Behavioral of notgate is

begin

b<= not a;

end Behavioral;
```

NAND GATE



A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

TIMING DIAGRAM:



NAND_GATE

THEORY: The NAND_GATE performs the logical NAND of two variables A and B produces an output C.

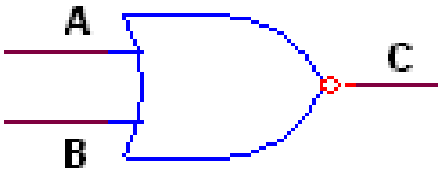
PROCEDURE:

- The NAND gate Design is entered through VHDL.
- The design is simulated by applying test vectors-a,b and observe the output c.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR NAND GATE:

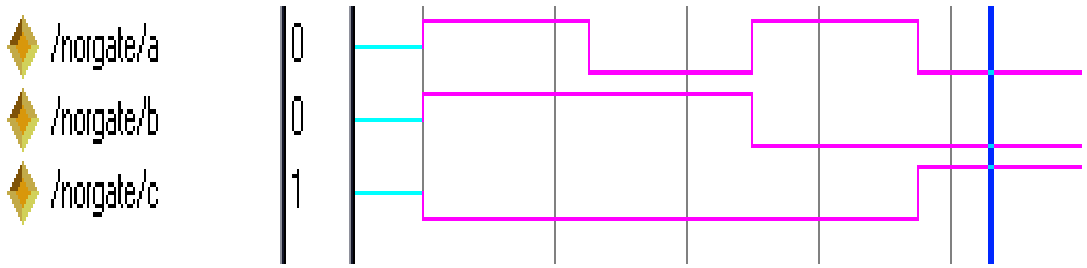
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity nandgate is

   Port ( a,b : in  STD_LOGIC;

          c : out  STD_LOGIC);

end nandgate;

architecture Behavioral of nandgate is

begin

c<= a nand b;

end Behavioral;
```

NOR GATE



A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

TIMING DIAGRAM:



NOR GATE

THEORY: The NOR_GATE performs the logical NOR of two variables A and B produces an output C.

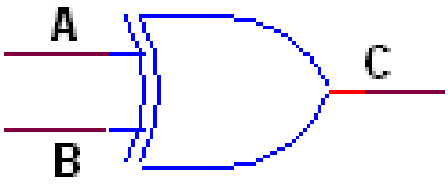
PROCEDURE:

- The NOR gate Design is entered through VHDL.
- The design is simulated by applying test vectors-a,b and observing output c.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR NOR GATE:

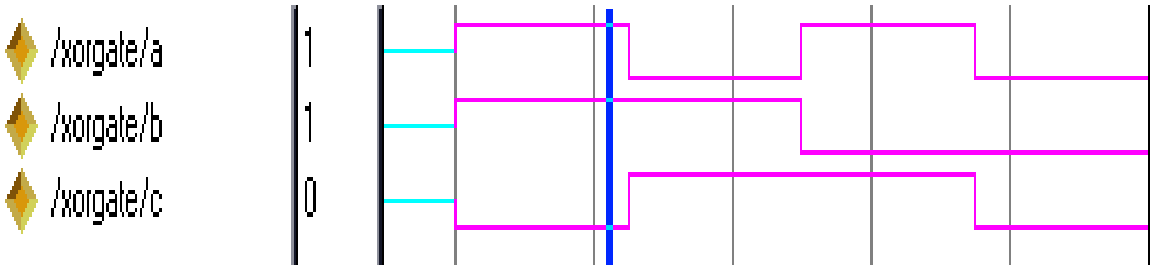
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity norgate is

   Port ( a,b : in  STD_LOGIC;

          c : out  STD_LOGIC);

end norgate;

architecture Behavioral of norgate is

begin

c<= a nor b;

end Behavioral;
```

EX-OR GATE



A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

TIMING DIAGRAM:



EX-OR GATE

THEORY: The EXOR_GATE performs the logical EXOR of two variables A and B produces an output C.

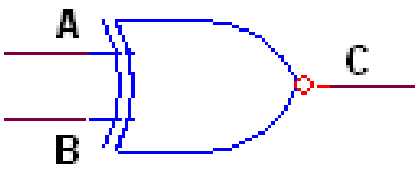
PROCEDURE:

- The EXOR gate Design is entered through VHDL.
- The design is simulated by applying test vectors-a,b and observing output c.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR EXOR GATE:

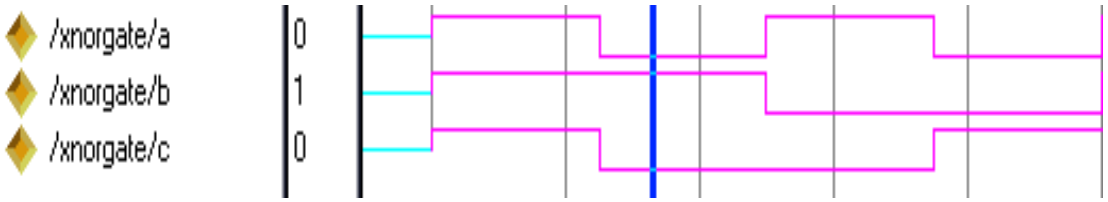
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity xorgate is

   Port ( a,b : in  STD_LOGIC;

          c : out  STD_LOGIC);

end xorgate;

architecture Behavioral of xorgate is

begin

c<= a xor b;

end Behavioral;
```

EX-NOR GATE



A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

TIMING DIAGRAM:



EX-NOR GATE

THEORY: The EXNOR_GATE performs the logical EXNOR of two variables A and B produces an output C.

PROCEDURE:

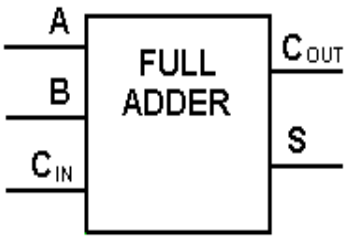
- The EXNOR gate Design is entered through VHDL.
- The design is simulated by applying test vectors-a,b and observing output c.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR EXNOR GATE:

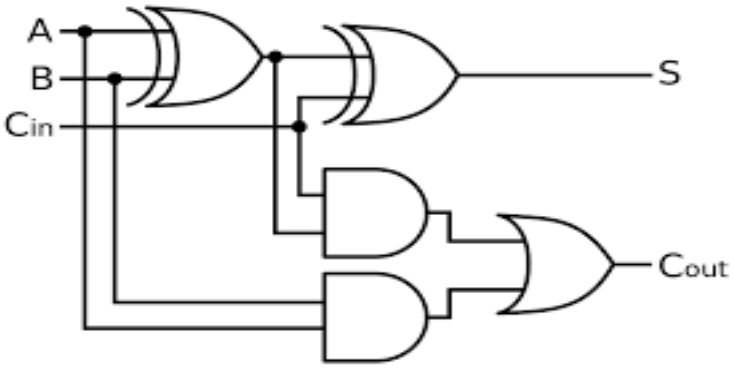
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity xnorgate is

   Port ( a,b : in  STD_LOGIC;

          c : out  STD_LOGIC);

end xnorgate;

architecture Behavioral of xnorgate is

begin

c<= a xnor b;

end Behavioral;
```

RESULT:

BLOCK DIAGRAM:



LOGIC DIAGRAM:



TRUTH TABLE:

Inputs			Outputs	
A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

EXP NO:

Design of Full Adder using 3 modeling styles

Date:

ABSTRACT: To study and simulate Full adder circuit in three different modeling VHDL.

THEORY: A full adder is a combinational circuit that forms the arithmetic sum of input; it consists of three inputs and two outputs. A full adder is useful to add three bits at a time but a half adder cannot do so. In full adder sum output will be taken from X-OR Gate, carry output will be taken from OR Gate.

PROCEDURE:

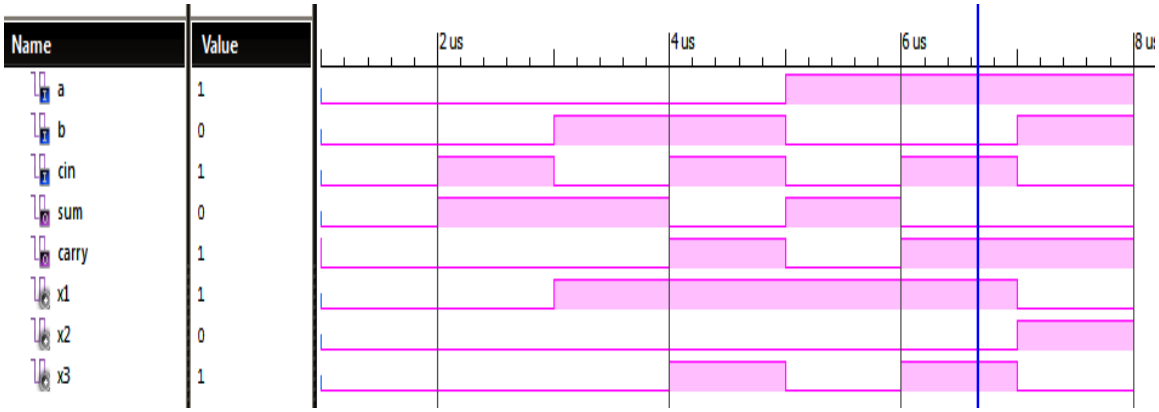
- The Full adder Design is entered through VHDL.
- The design is simulated by applying test vectors-a,b, c and observe the output sum, carry.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR FULLADDER:

BEHAVIORAL STYLE

```
entity  fulladderbeh  is

        Port ( a,b,c : in std_logic;

        sum,carry : out std_logic);

        end fulladderbeh;

architecture Behavioral of fulladderbeh  is

begin

        process( a,b,c)

begin

if(a='0' and b='0' and c='0') then sum<='0';

carry<='0';

elsif(a='0' and b='0' and c='1') then sum<='1';

carry<='0';

elsif(a='0' and b='1' and c='0') then sum<='1';

carry<='0';

elsif(a='0' and b='1' and c='1') then sum<='0';
```

TIMING DIAGRAM:



```
carry<='1';
elsif(a='1' and b='0' and c='0') then sum<='1';
carry<='0';
elsif(a='1' and b='0' and c='1') then sum<='0';
carry<='1';
elsif(a='1' and b='1' and c='0') then sum<='0';
carry<='1';
else
sum<='1'; carry<='1';
end if;
end process;
end Behavioral;
```

DATA FLOW MODELING

```
entity fulladder is
        Port ( a,b,c : in std_logic;
         s,cout : out std_logic);
        end fulladr;
architecture data of fulladr is
begin
 sum<=a xor b xor cin;
 cout<= ( a and b) or ( b and cin) or ( cin and
a);
end data;
```

STRUCTURAL STYLE

```
entity fullstru is
Port ( a,b,cin : in std_logic;
     sum,carry : out std_logic);
```

```
end fullstru;
architecture structural of fullstru is
 signal c1,c2,c3:std_logic;
 component xor_3
 port(x,y,z:in std_logic;
u:out std_logic);
end component;
 component and_2
 port(l,m:in std_logic;
 n:out std_logic);
end component;
component or_3
 port(p,q,r:in std_logic;
 s:out std_logic);
end component;
begin
X1: xor_3 port map ( a, b, cin,sum);
A1: and_2 port map (a, b, c1);
A2: and_2 port map (b,cin,c2);
A3: and_2 port map (a,cin,c3);
O1: or_3 port map (c1,c2,c3,carry);
end structural;
```

LINKUP FOR STRUCTURAL MODELLING

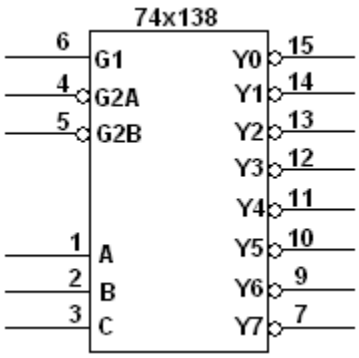
```
//and gate//
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity and_2 is
 Port ( l,m : in std_logic;
 n : out std_logic);
end and2;
architecture dataf of and2 is
begin
n<=l and m;
end dataf;
//or gate//
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity or_3 is
 Port ( p,q,r : in std_logic;
 s : out std_logic);
end or3;
architecture dat of or_3 is
begin
 s<= p or q or r;
end dat;
//xor gate//
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity xor_3 is
 Port ( x,y,z : in std_logic;
 u : out std_logic);
end xor_3;
architecture dat of xor_3 is
begin
```

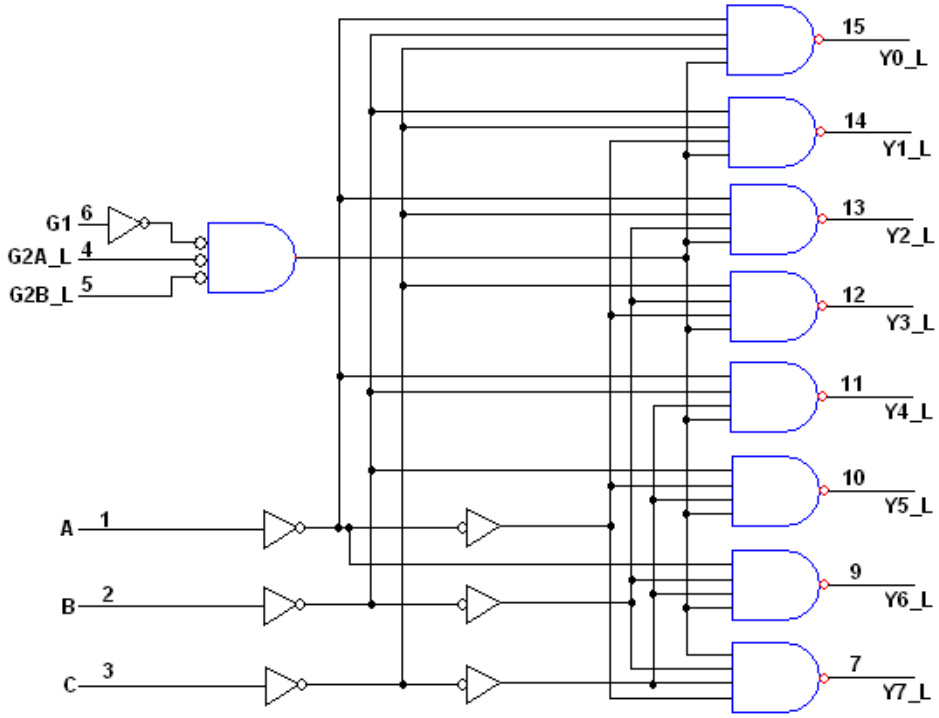
```
 u<=x xor y xor z;
end dat;
```

RESULT:

PIN DIAGRAM:



INTERNAL DIAGRAM:



EXP NO:

Date:

3 TO 8 DECODER (IC 74138)

ABSTRACT: To study and simulate IC 74138 using VHDL.

THEORY: The74138 is a commercially available MSI 3 to 8 decoder. The 74138 has3 enable inputs (G1, G2A_L, G2B_L), all of which must be asserted for the selected output it has 3 active inputs and 8 active low outputs. Based on the select inputs (n) one output among 2^n outputs.

PROCEDURE:

- The IC 74138 Design is entered through VHDL.
- The design is simulated by applying test vectors-G1, G2A_L, G2B_L ,I and observing output y_l.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

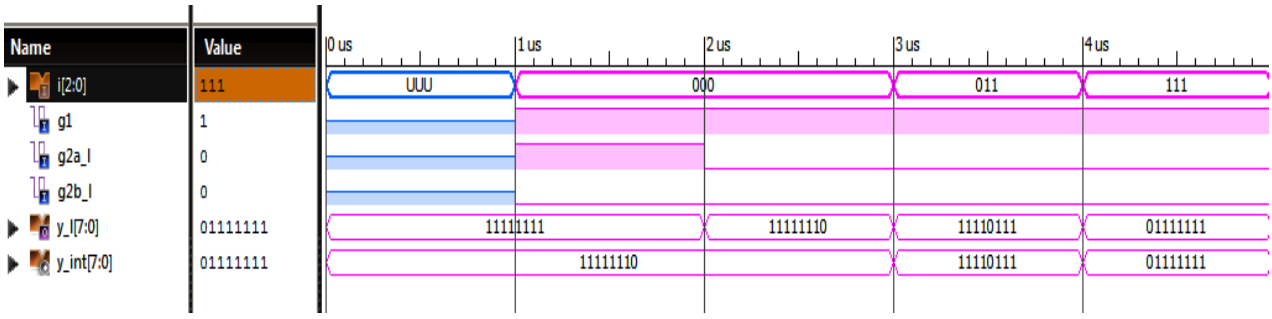
VHDL PROGRAM FOR 3 TO 8 DECODER IC 74138:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ic74138 is

   Port ( i : in  STD_LOGIC_VECTOR (2 downto 0);

          g1,g2a_l,g2b_l : in  STD_LOGIC;

          y_l : out  STD_LOGIC_VECTOR (7 downto 0));

end ic74138;

architecture dataflow of ic74138 is

           signal y_int:std_logic_vector(7 downto 0);
```

TRUTH TABLE OF IC74138:

INPUTS						OUTPUTS							
G1	G2A_L	G2B_L	C	B	A	Y7_L	Y6_L	Y5_L	Y4_L	Y3_L	Y2_L	Y1_L	Y0_L
0	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	1	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

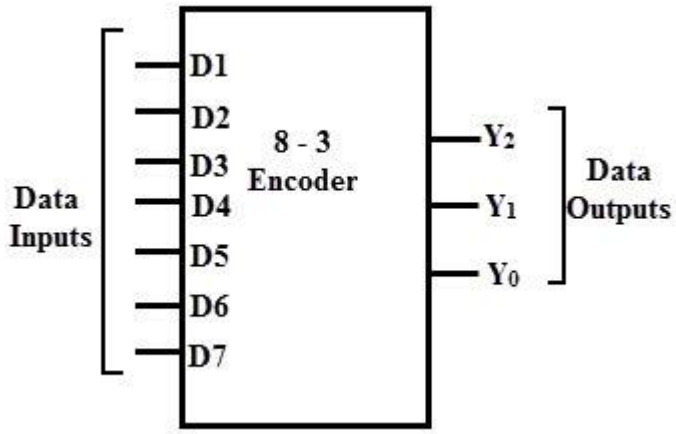
TIMING DIAGRAM OF IC 74138:



```
begin
with i select
y_int<="01111111" when "111",
"10111111" when "110",
"11011111" when "101",
"11101111" when "100",
"11110111" when "011",
"11111011" when "010",
"11111101" when "001",
"11111110" when others;
y_l<=y_int when(g1 and (not g2a_l) and( not g2b_l))='1' else "11111111";
end dataflow;
```

RESULT:

BLOCK DIAGRAM:



TRUTHTABLE:

INPUTS								OUTPUTS		
D0	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**Y2**	**Y1**	**Y0**
1	0	0	0	0	0	0	0	0	0	0
X	1	0	0	0	0	0	0	0	0	1
X	X	1	0	0	0	0	0	0	1	0
X	X	X	1	0	0	0	0	0	1	1
X	X	X	X	1	0	0	0	1	0	0
X	X	X	X	X	1	0	0	1	0	1
X	X	X	X	X	X	1	0	1	1	0
X	X	X	X	X	X	X	1	1	1	1

EXP NO:

Date:

8 to 3 Encoder (with and without priority)

ABSTRACT: To study and simulate design of 8 to 3 Encoder (with and without priority) using VHDL.

THEORY: 8 to 3 encoder has 8 inputs and only one output based on the select inputs (2^n) stress out one output n. Priority encoders are available in standard IC form and the TTL 74LS148 is an 8-to-3 bit priority encoder which has eight active LOW (logic "0") inputs and provides a 3-bit code of the highest ranked input at its output.

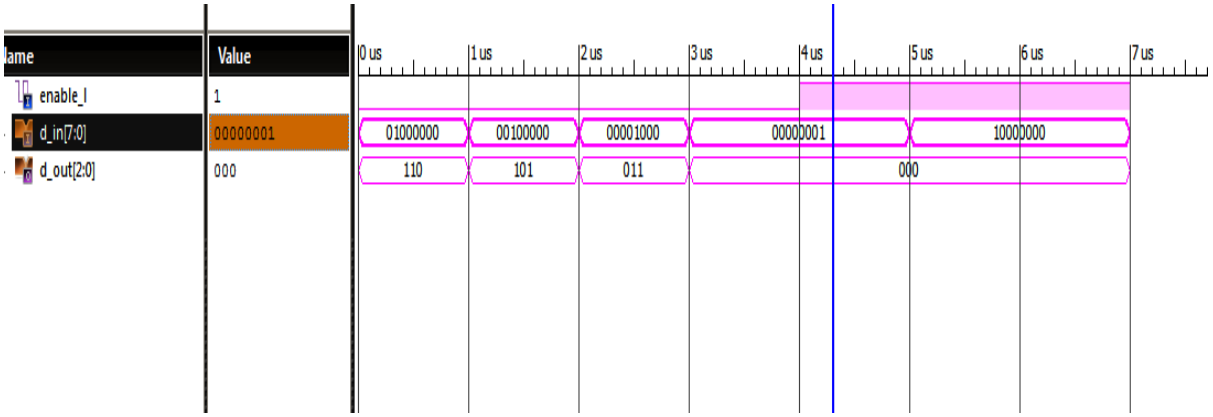
PROCEDURE:

- The 8 to 3 encoder Design is entered through VHDL.
- The design is simulated by applying test vectors- ENABLE_L, D_IN and observing output D-OUT.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR 8 TO 3 ENCODER (with out priority):

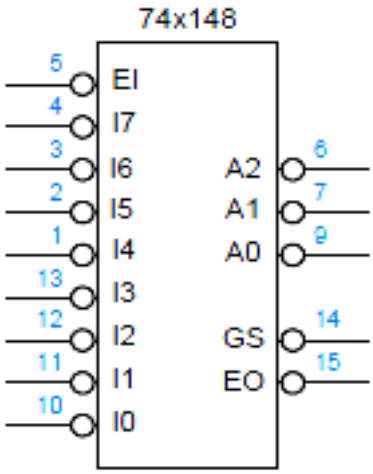
```
library IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY ENCODER8_3 IS
PORT ( ENABLE_L :  IN STD_LOGIC;

D_IN: IN STD_LOGIC_VECTOR(7 DOWNTO 0);

D_OUT: OUT STD_LOGIC_VECTOR(2 DOWNTO 0) );

END ENCODER8_3;

ARCHITECTURE ENCODER_ARCH OF ENCODER8_3 IS

BEGIN

PROCESS(ENABLE_L,D_IN)

BEGIN

IF ( ENABLE_L = '1') THEN

D_OUT <= "000";

ELSE
```

TIMING WAVEFORM:

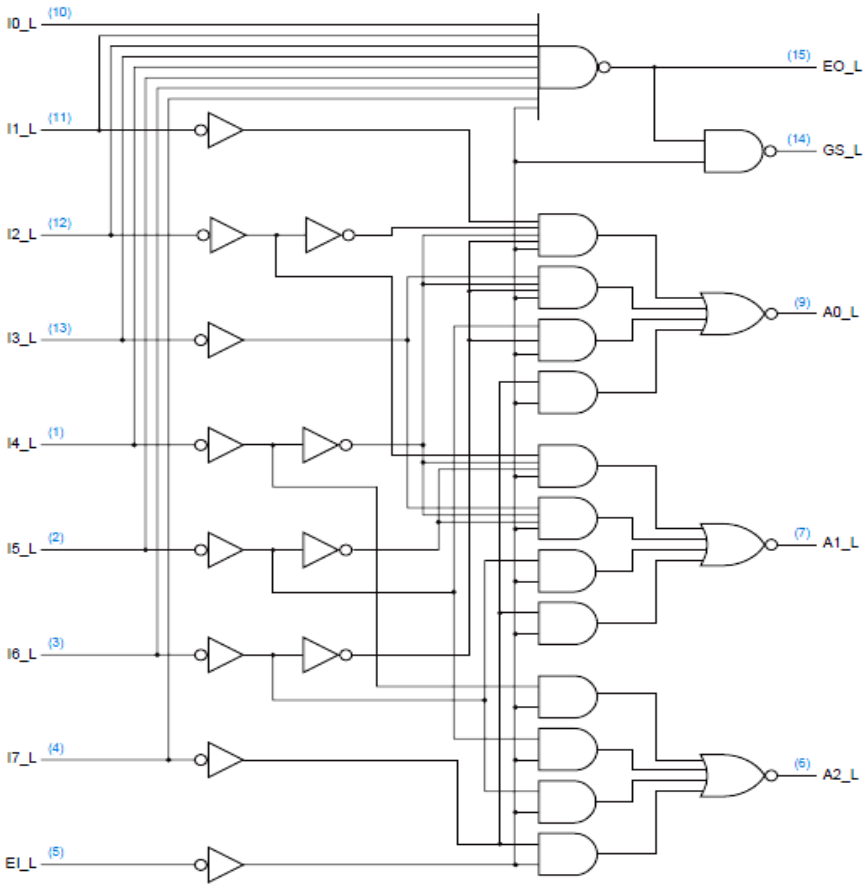


```
CASE D_IN IS

WHEN "00000001" => D_OUT <= "000";

WHEN "00000010" => D_OUT <= "001";

WHEN "00000100" => D_OUT <= "010";

WHEN "00001000" => D_OUT <= "011";

WHEN "00010000" => D_OUT <= "100";

WHEN "00100000" => D_OUT <= "101";

WHEN "01000000" => D_OUT <= "110";

WHEN "10000000" => D_OUT <= "111";

WHEN OTHERS => NULL;

END CASE;

END IF;

END PROCESS;

END ENCODER_ARCH;
```

LOGIC DIAGRAM:



INTERNAL DIAGRAM:



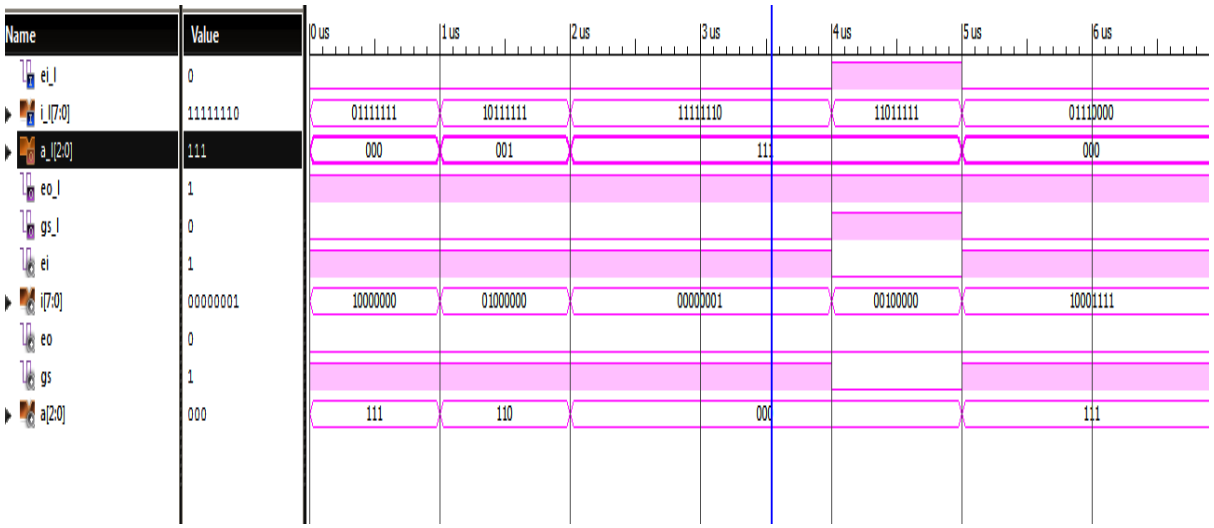
VHDL PROGRAM FOR 8 TO 3 ENCODER (with priority):

```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity V74x148 is
Port (    EI_L: in std_logic;
          I_L: in std_logic_vector(7 downto  0);
          A_L: out std_logic_vector(2 downto  0);
          EO_L,GS_L: out std_logic);
End V74x148;
Architecture behavioral of V74x148 id
Signal EI:  std_logic;
Signal I: std_logic_vector(7 downto 0);
Signal EO,GS: std_logic;
Signal A: std_logic_vector(2 downto 0);
Begin
process(EI_L,I_L,EI,EO,GS,I,A)
variable j;integer range 7 downto 0;
begin
EI<= not EI_L;
I<= not I_L;
EO<='1';
GS<='0';
A<="000";
If (EI)='0' then EO<='0';
Else for j in 7 downto 0 loop
If I(j)='1' then
GS<='1'; EO<='0'; A<=conv_std_logic_vector(j,3);
Exit;
```

TRUTH TABLE :

INPUTS									OUTPUTS				
EI_L	**I0_L**	**I1_L**	**I2_L**	**I3_L**	**I4_L**	**I5_L**	**I6_L**	**I7_L**	**A2_L**	**A1_L**	**A0_L**	**GS_L**	**EO_L**
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	0	0	1	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	0	1	1	0	1
0	X	X	X	0	1	1	1	1	1	0	0	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0

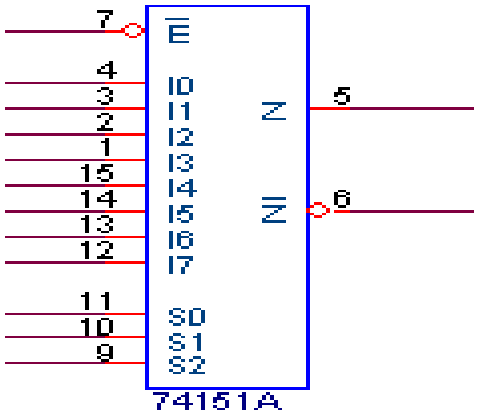
TIMING WAVEFORM:



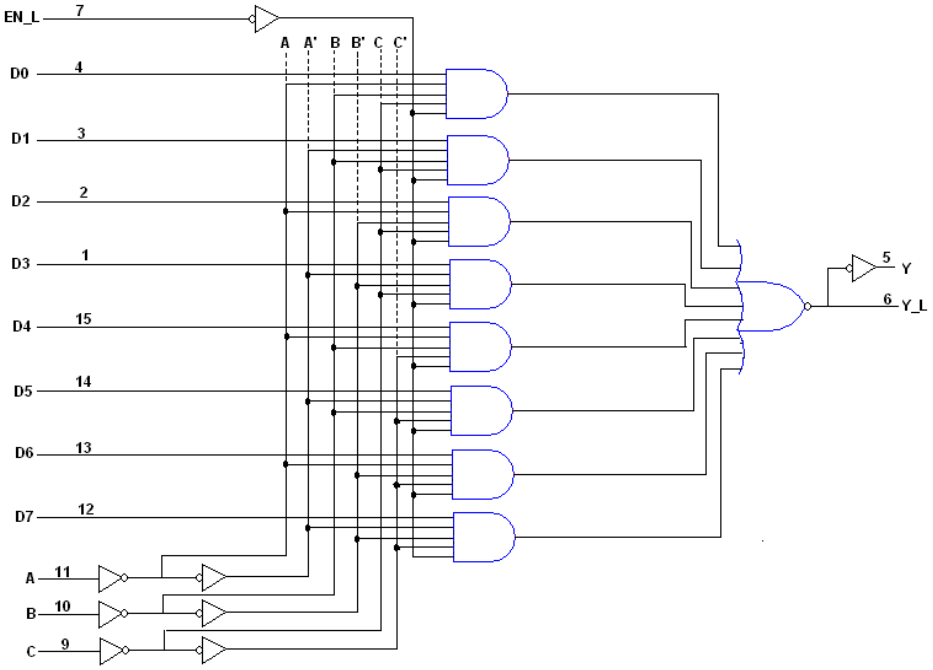
```
End if;

End loop;

End if;

EO_L<= not EO;

GS_L<= not GS;

A_L<= not A;

End process;

End behavioral;
```

RESULT:

PIN DIAGRAM:



INTERNAL DIAGRAM:



EXP NO:

Date:

8 TO 1 MULTIPLEXER (IC 74151) &

1 TO 4 DEMULTIPLEXER (IC 74155)

ABSTRACT: To study and simulate design of IC 74151 using VHDL.

THEORY: Multiplexer IC 74151 is 8 to 1 multiplexer. It has 8 inputs and only one output based on the select inputs A, B, C it steers one of the input to the output Y.

PROCEDURE:

- The IC 74151 Design is entered through VHDL.
- The design is simulated by applying test vectors-s,en_l,i and observing output y, y_l.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

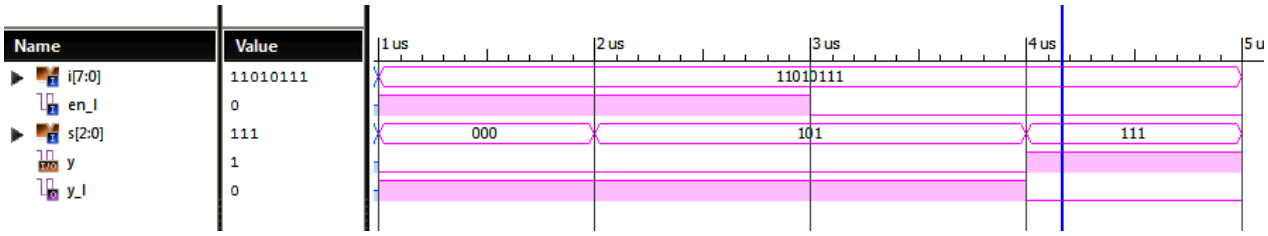
VHDL PROGRAM FOR 8 TO 1 MULTIPLEXER (IC 74151):

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ic74151 is

   Port ( i : in  STD_LOGIC_VECTOR (7 downto 0);

          en_l : in  STD_LOGIC;

          s : in  STD_LOGIC_VECTOR (2 downto 0);

          y : inout  STD_LOGIC;

          y_l : out  STD_LOGIC);

end ic74151;

architecture Behavioral of ic74151 is

begin
```

TRUTH TABLE OF IC74151:

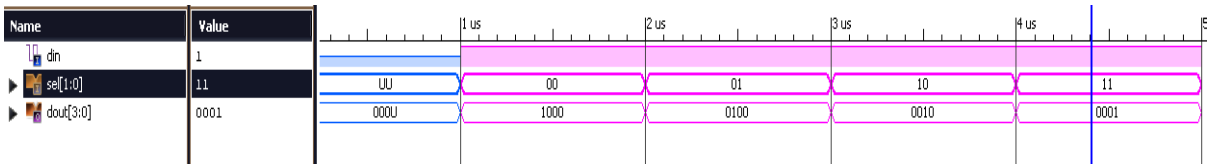
INPUTS				OUTPUTS	
EN_L	C	B	A	Y	Y_L
1	X	X	X	0	1
0	0	0	0	D0	D0’
0	0	0	1	D1	D1’
0	0	1	0	D2	D2’
0	0	1	1	D3	D3’
0	1	0	0	D4	D4’
0	1	0	1	D5	D5’
0	1	1	0	D6	D6’
0	1	1	1	D7	D7’

TIMING DIAGRAM OF IC 74151:

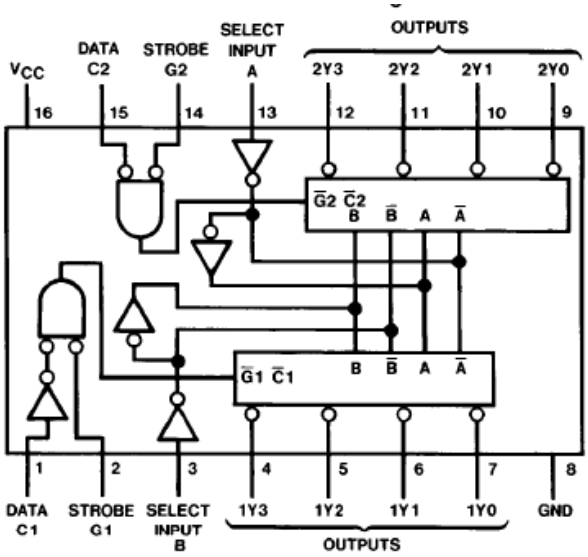


```
process(i,en_l,s)
begin
if(en_l='1') then y<='0';
else
case s is
when"000"=>y<= i(0);
when"001"=>y<= i(1);
when"010"=> y <= i(2);
when"011"=> y <= i(3);
when"100"=> y <= i(4);
when"101"=> y <= i(5);
when"110"=> y <= i(6);
when others=> y <= i(7);
end case;
end if;
end process;
y_l<=not y;
end Behavioral;
```

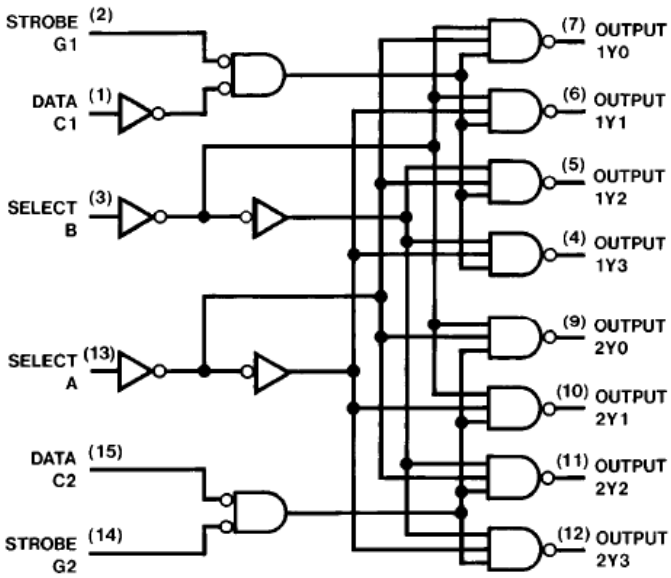
TIMING DIAGRAM OF IC 74155:



PIN DIAGRAM:



CIRCUIT DIAGRAM:

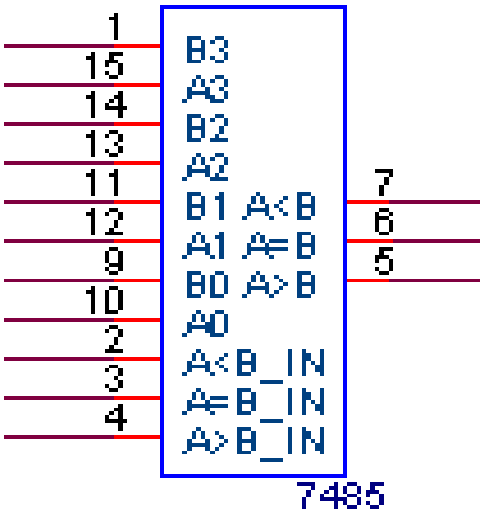


VHDL PROGRAM FOR 1 X 4 DEMULTIPLEXER:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity demultiplexer is
    port(
       din : in STD_LOGIC;
       sel : in STD_LOGIC_VECTOR(1 downto 0);
       dout : out STD_LOGIC_VECTOR(3 downto 0)
       );
end demultiplexer;
architecture demultiplexer_arch of demultiplexer is
begin
 process (din,sel)
   begin
      case sel is
         when "00" => dout <= din & "000";
         when "01" => dout <= '0' & din & "00";
         when "10" => dout <= "00" & din & '0';
         when others => dout <= "000" & din;
      end case;
   end process;
end demultiplexer_arch;
```

RESULT:

PIN DIAGRAM:



TRUTH TABLE:

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A3, B3	A2,B2	A1,B1	A0, B0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	A>B	A<B	A=B
A3>B3	X	X	X	X	X	X	H	L	L
A3<B3	X	X	X	X	X	X	L	H	L
A3=B3	A2>B2	X	X	X	X	X	H	L	L
A3=B3	A2<B2	X	X	X	X	X	L	H	L
A3=B3	A2=B2	A1>B1	X	X	X	X	H	L	L
A3=B3	A2=B2	A1<B1	X	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	H	L	L
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	H	L	L	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	L	L	L	H	H	L

EXP NO:

4 BIT COMPARATOR (IC 7485)

Date:

ABSTRACT: To study and simulate design of 4bit comparator IC 7485 using VHDL.

THEORY: The7485 is a 4bit magnitude comparator that can be expanded to almost any length. It has three cascade inputs A>Bin, A<Bin and A=Bin. The four bit inputs are weighted (A0-A3) and (B0-B3), where A3&B3 are the most significant bits. The cascading outputs are A>Bout, A<Bout and A equ Bout.

AGTBOUT=(A>B)+(A=B). AGTBIN

AEQBOUT=(A=B). AEQBIN

ALTBOUT=(A<B)+(A=B). ALTBIN

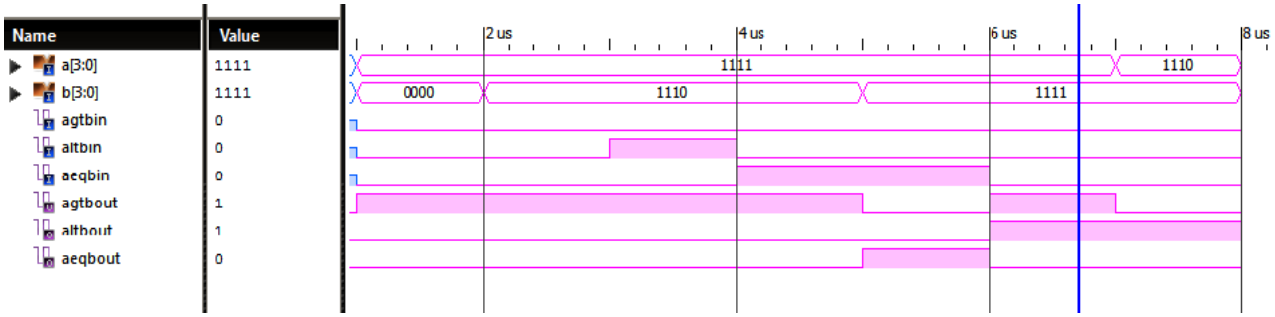
PROCEDURE:

- The IC 7485 Design is entered through VHDL.
- The design is simulated by applying test vectors-a, b, agtbin, aeqbin, altbin and observe the output for agtbout, altbout and aeqbout.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output of the implementation can be directly programmed into target device FPGA.

VHDL PROGRAM FOR 4 BIT COMPARATOR (IC 7485):

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ic7485 is

   Port ( a,b : in  STD_LOGIC_VECTOR (3 downto 0);

        agtbin, altbin, aeqbin : in  STD_LOGIC;
```

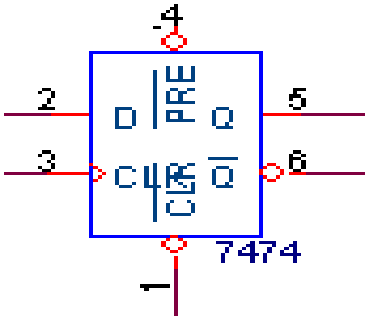
TIMING DIAGRAM OF IC 7485:


Name
Value
a[3:0]
1111
b[3:0]
1111
agtbin
0
altbin
0
aeqbin
0
agtbout
1
altbout
1
aeqbout
0
2 us
4 us
6 us
8 us
1111
1110
0000
1110
1111

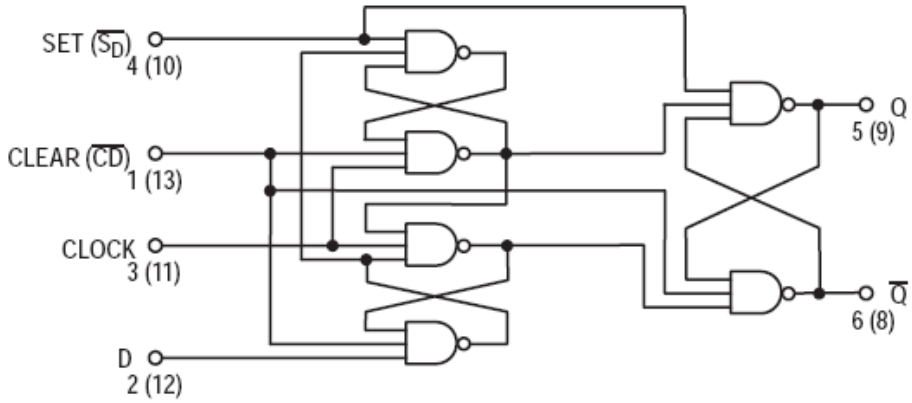
```
agtbout, altbout, aeqbout : out  STD_LOGIC);
end ic7485;
architecture behavioral of ic7485 is
begin
process(a,b,agtbin,aeqbin,altbin)
begin
if(A>B or (A=B and agtbin='1' and altbin='0' and aeqbin='0')) then
agtbout<='1';
elsif(A=B and altbin='0' and agtbin='0' and aeqbin='0') then
agtbout<='1';
else agtbout<='0';
end if;
if(A=B and agtbin='0' and altbin='0' and aeqbin='1') then
aeqbout<='1';
elsif(A=B and altbin='0' and agtbin='0' and aeqbin='0') then
aeqbout<='0';
else aeqbout<='0';
end if;
if(A<B or (A=B and agtbin='0' and altbin='1' and aeqbin='0')) then
altbout<='1';
elsif(A=B and altbin='0' and agtbin='0' and aeqbin='0') then
altbout<='1';
else altbout<='0';
end if; end process;
end behavioral;
```

RESULT:

PIN DIAGRAM:



CIRCUIT DIAGRAM:



TRUTH TABLE OF IC7474:

Operating mode	inputs				outputs	
	Pr_l	clr_L	CLK	D	Q	Qn
Asynchronus set	L	H	X	X	H	L
Asynchronus reset	H	L	X	X	L	H
Load “1”	H	H	CLK	H	H	L
Load “0”	H	H	CLK	L	L	H
indeterminate	L	L	X	X	H	H

EXP NO:

D FLIP-FLOP(IC 7474)

Date:

ABSTRACT: To study and simulate D FLIP-FLOP using VHDL.

THEORY: The 74 is a dual positive edge triggered D type flip flop featuring individual data, clock, set and reset inputs; also complementary Qn, Qn bar. The asynchronous set and reset pins are PR_L and CLR_L these inputs operate independent of clock. Information on the data D input is transferred to the Qn output on the low to high transition of the clock pulse.

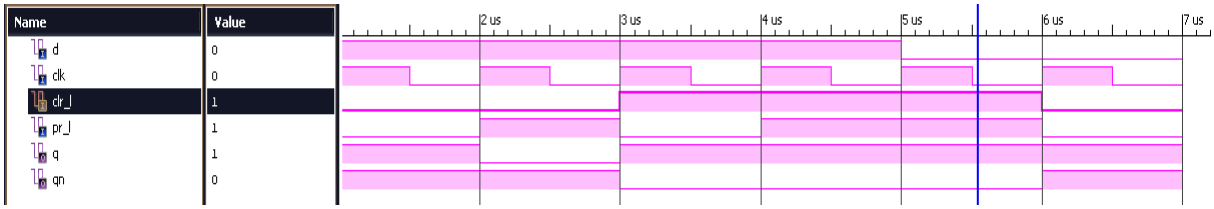
PROCEDURE:

- Logical gate D Flip-flop Design is entered through VHDL.
- The design is simulated by applying test vectors-pr_l, clr_l, clk, d and observing outputs q and qn.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR D FLIP-FLOP IC 7474:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity dff is

   Port ( d,clk,clr_l,pr_l : in  STD_LOGIC;

          q,qn : out  STD_LOGIC);

end dff;

architecture Behavioral of dff is

begin
```

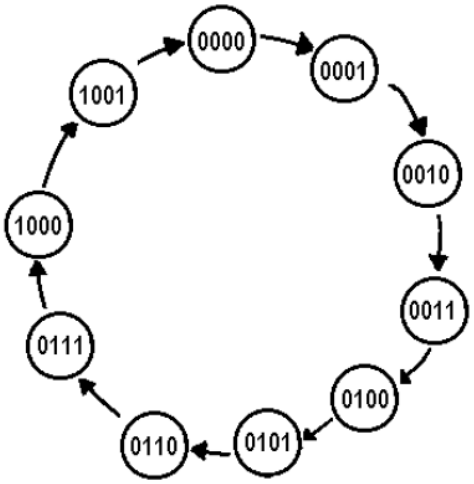
TIMING DIAGRAM OF IC 7474:



```
process(pr_l,clr_l,d,clk)
begin
if (clr_l='0' and pr_l='0') then q<='1';qn<='1';
elsif clr_l='0' then q<='0';qn<='1';
elsif pr_l='0' then q<='1';qn<='0';
elsif(clk'event and clk='1') then q<=d;
qn<=not d;
end if;
end process;
end Behavioral;
```

RESULT:

State diagram:



TRUTH TABLE OF IC7490:

COUNT	Q_0	Q_1	Q2	Q3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	H	H	L
7	H	H	H	L
8	L	L	L	H
9	H	L	L	H

EXP NO:

Date:

DECADE COUNTER (IC 7490)

ABSTRACT: To study and simulate DECADE COUNTER using VHDL.

THEORY: The decade counter counts 0 to 9.When tenth pulse reaches it reset back to 0

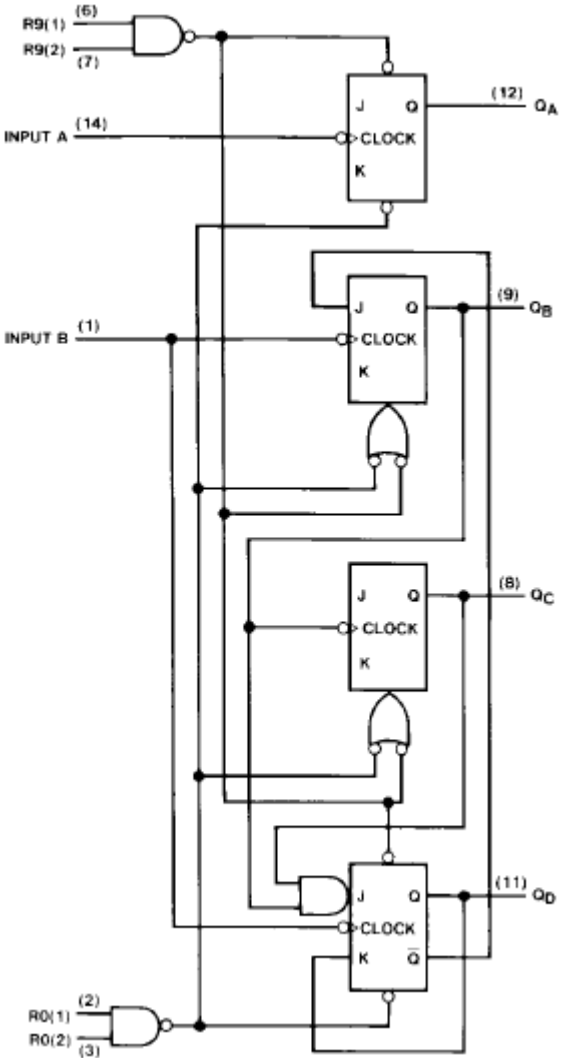
PROCEDURE:

- The decade counter Design is entered through VHDL.
- The design is simulated by applying test vectors-reset, clk and observing output d.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

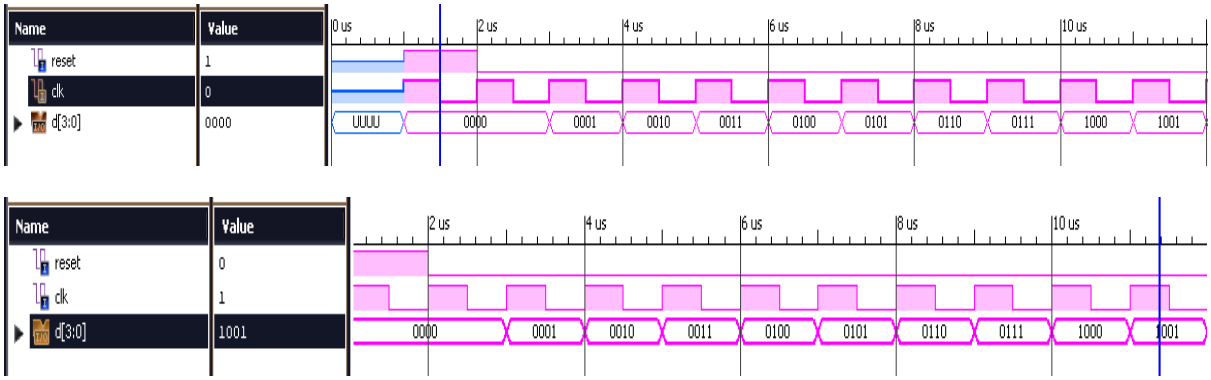
VHDL PROGRAM FOR DECADE COUNTER (IC7490):

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity counter10 is

   Port ( reset : in  STD_LOGIC;

          clk : in  STD_LOGIC;

            d : inout  STD_LOGIC_VECTOR (3 downto 0));

end counter10;

architecture Behavioral of counter10 is

begin

process(reset,clk,d)

begin
```

SCHEMATIC DIAGRAM:



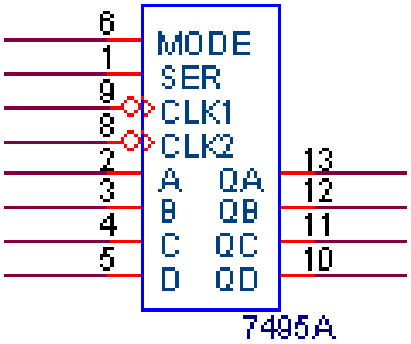
TIMING DIAGRAM:



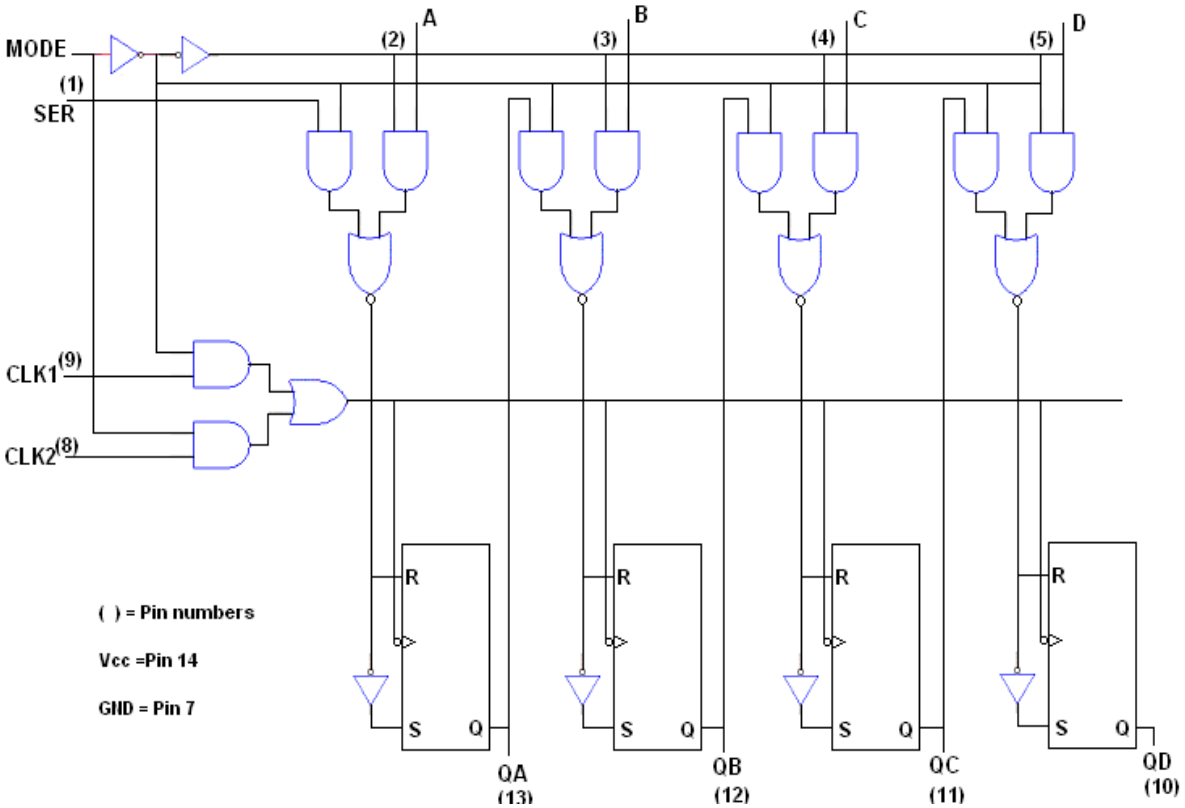
```
if reset='1' then
d<="0000";
elsif(clk'event and clk='1')then
d<=d+1;
if(d="1001") then
d<="0000";
end if;
end if;
end process;
end Behavioral;
```

RESULT:

PN DIAGRAM:



SCHEMATIC DIAGRAM:



EXP NO:

Date:

SHIFT REGISTER (IC 7495)

ABSTRACT: To study and simulate IC 7495using VHDL.

THEORY: The IC 7495 is a parallel to serial shift register. After four clock pulses the data is serially shifted into the parallel output.

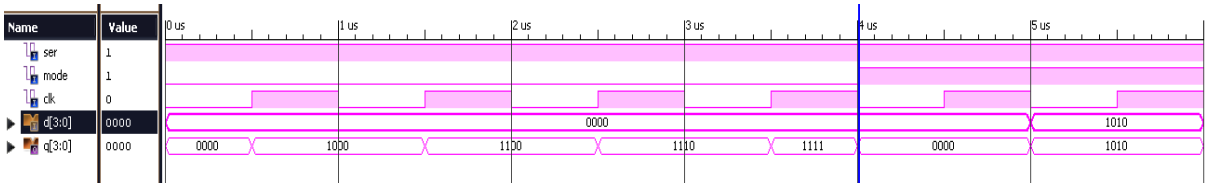
PROCEDURE:

- The IC 7495 Design is entered through VHDL.
- The design is simulated by applying test vectors- i, clk and observing output d.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR SHIFT REGISTER (IC 7495):

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity IC7495 is

Port ( ser,mode : in  STD_LOGIC;

          clk : in  STD_LOGIC;

            d : in  STD_LOGIC_VECTOR (3 downto 0);

            q : out  STD_LOGIC_VECTOR (3 downto 0));

end IC7495;
```

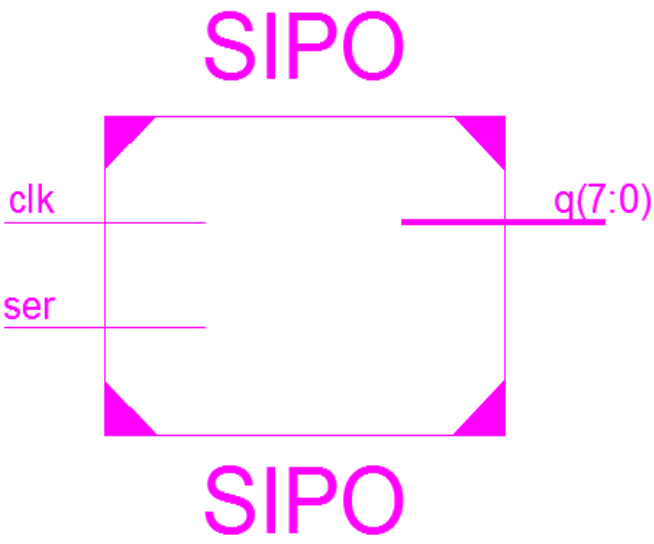
TIMING DIAGRAM:



```
architecture Behavioral of IC7495 is
begin
process( ser, mode, d, clk)
variable b:std_logic_vector(3 downto 0):="0000";
begin
if(mode='1' )then
b:=d;
elsif ((clk' event and clk='1')) then
b(0):=b(1);
b(1) :=b(2);
b(2) :=b(3);
b(3) :=ser;
end if;
q<=b;
end process;
end Behavioral;
```

RESULT:

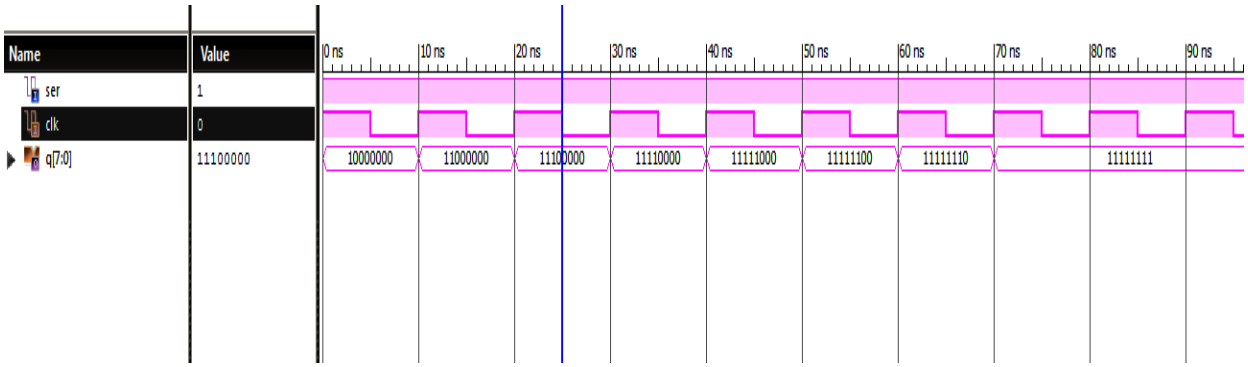
BLOCK DIAGRAM:



TRUTH TABLE:

CLK	SER	Q_7-Q_0
0	X	00000000
1	X	SER&Q_7-Q_1

TIMING WAVEFORM:



EXP NO:

Date:

8-BIT SERIAL IN-PARALLEL OUT AND PARALLEL IN-SERIAL OUT

ABSTRACT: To study and simulate SIPO and PISO using VHDL.

THEORY: In **Serial In Parallel Out (SIPO) shift registers**, the data is stored into the register serially while it is retrieved from it in parallel-fashion. In **Parallel In serial Out (PISO) shift registers**, the data is stored into the register parallel while it is retrieved from it in Serial-fashion.

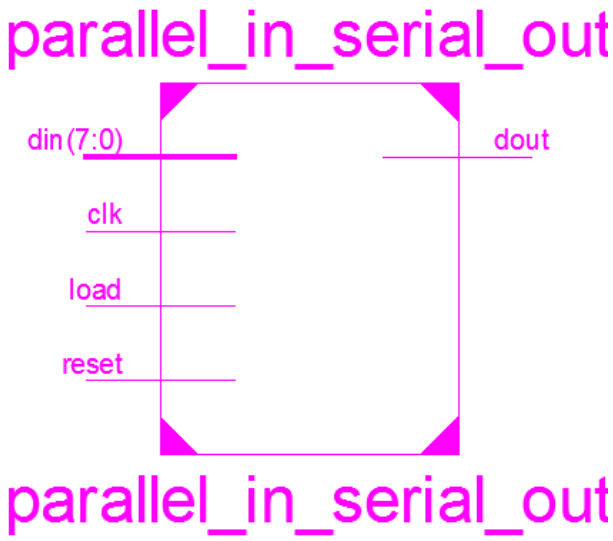
PROCEDURE:

- The SIPO AND PISO Design is entered through VHDL.
- The design is simulated by applying test vectors- i, clk and observing output Dataout.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR SIPO:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity SIPO  is
Port ( ser: in  STD_LOGIC;
                clk : in  STD_LOGIC;
                  q : out  STD_LOGIC_VECTOR (7 downto 0));
end SIPO;
architecture Behavioral of SIPO is
begin
process( ser, clk)
variable b: std_logic_vector(7 downto 0):="00000000";
begin
If ((clk' event and clk='1')) then
b(0):=b(1);
b(1) :=b(2);
b(2) :=b(3);
b(3) :=b(4);
b(4):=b(5);
b(5) :=b(6);
b(6) :=b(7);
b(7) :=ser;
end if;
q<=b;
end process;
end Behavioral;
```

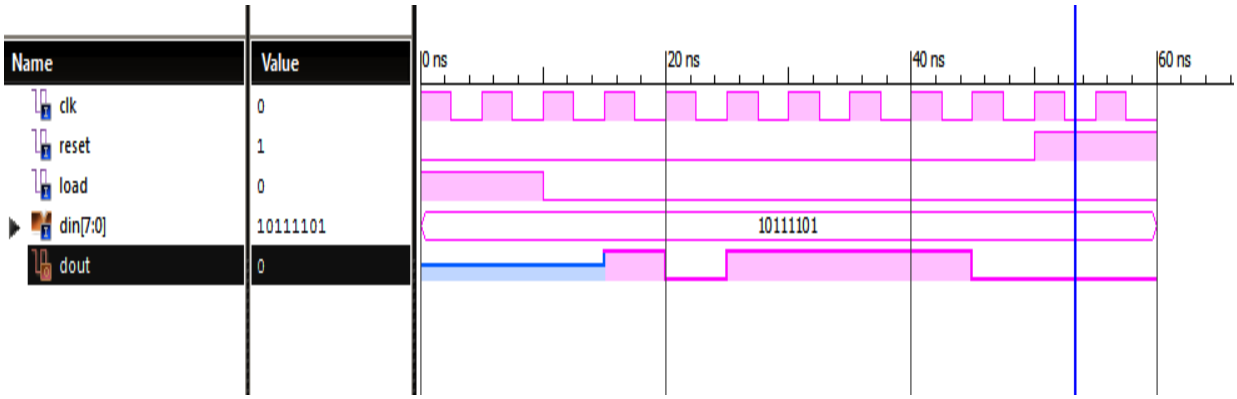
BLOCK DIAGRAM:



TRUTH TABLE:

CLK	RESET	LOAD	$D_{in}(D_0\text{-}D_7)$	D_{OUT}
0	X	X	X	0
1	1	X	X	0
1	0	1	D_{in}	0
1	0	0	D_{in}	D_7

TIMING WAVEFORMS:

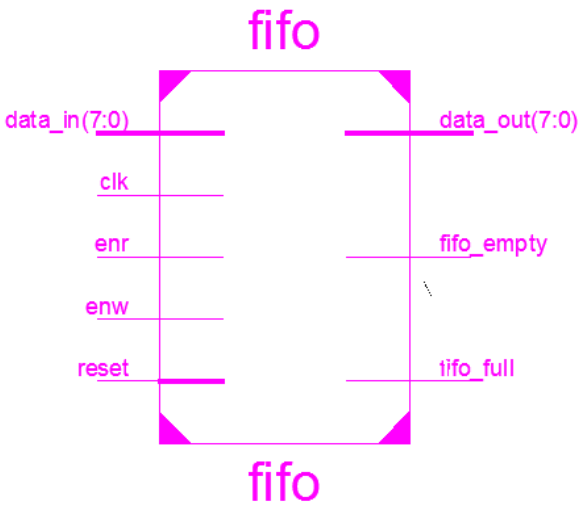


VHDL PROGRAM FOR PISO:

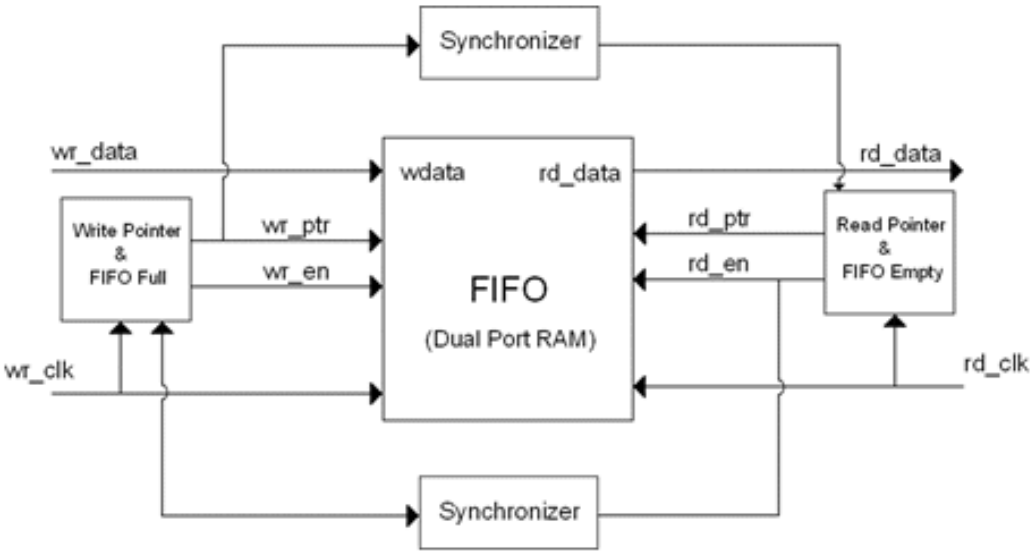
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity parallel_in_serial_out is
    port(
       clk : in STD_LOGIC;
       reset : in STD_LOGIC;
       load : in STD_LOGIC;
       din : in STD_LOGIC_VECTOR(7 downto 0);
       dout : out STD_LOGIC );
end parallel_in_serial_out;

architecture piso_arc of parallel_in_serial_out is
begin
   piso : process (clk,reset,load,din) is
   variable temp : std_logic_vector (din'range);
   begin
      if (reset='1') then
         temp := (others=>'0');
      elsif (load='1') then
         temp := din ;
      elsif (rising_edge (clk)) then
         dout <= temp(7);
         temp := temp(6 downto 0) & '0';
      end if;
   end process piso;
end piso_arc;
```

RESSULT:

PIN DIAGRAM:



INTERNAL ARCHITECTURE OF FIFO:



EXP NO:

Date:

FIFO
(FIRST IN FIRST OUT)

ABSTRACT: To study and simulate FIFO using VHDL.

THEORY: FIFOs (First In, First Out) are essentially memory buffers used to temporarily store data until another process is ready to read it. As their name suggests the first byte written into a FIFO will be the first one to appear on the output. Typically FIFOs are used when you have two processes that operate and a different rate. A common example is a high speed communications channel that writes a burst of data into a FIFO and then a slower communications channel that read the data as need to send it at a slower rate. The FIFO module has two settings that can be configured to adjust the width and depth of the FIFO. The *DATA_WIDTH* variable adjusts the size of the *DataIn* and *DataOut* buses so that you can write different sizes of bytes if needed and the *FIFO_DEPTH* variable adjusts how big the internal memory of the FIFO is.

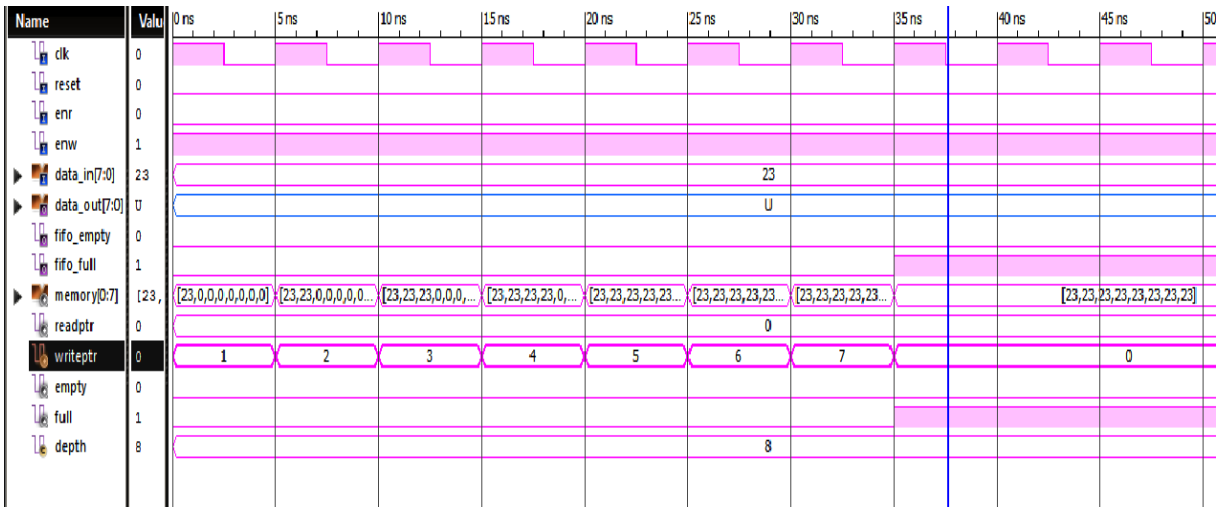
PROCEDURE:

- The FIFO Design is entered through VHDL.
- The design is simulated by applying test vectors- i, clk and observing output Dataout.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

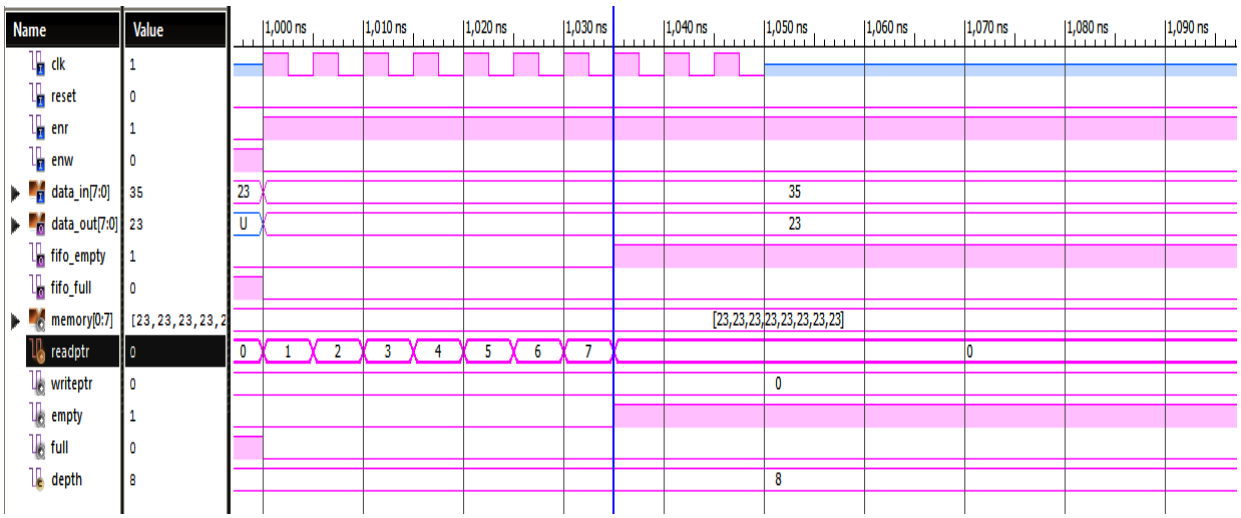
VHDL PROGRAM FOR FIFO:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;

entity fifo is

generic (depth : integer := 8);  --depth of fifo

port (    clk : in std_logic;

        reset : in std_logic;

        enr : in std_logic;   --enable read,should be '0' when not in use.

        enw : in std_logic;    --enable write,should be '0' when not in use.

        data_in : in std_logic_vector (7 downto 0);     --input data

        data_out : out std_logic_vector(7 downto 0);    --output data

        fifo_empty : out std_logic;     --set as '1' when the queue is empty

        fifo_full : out std_logic     --set as '1' when the queue is full     );
```

Timing diagram for WRITE operation:

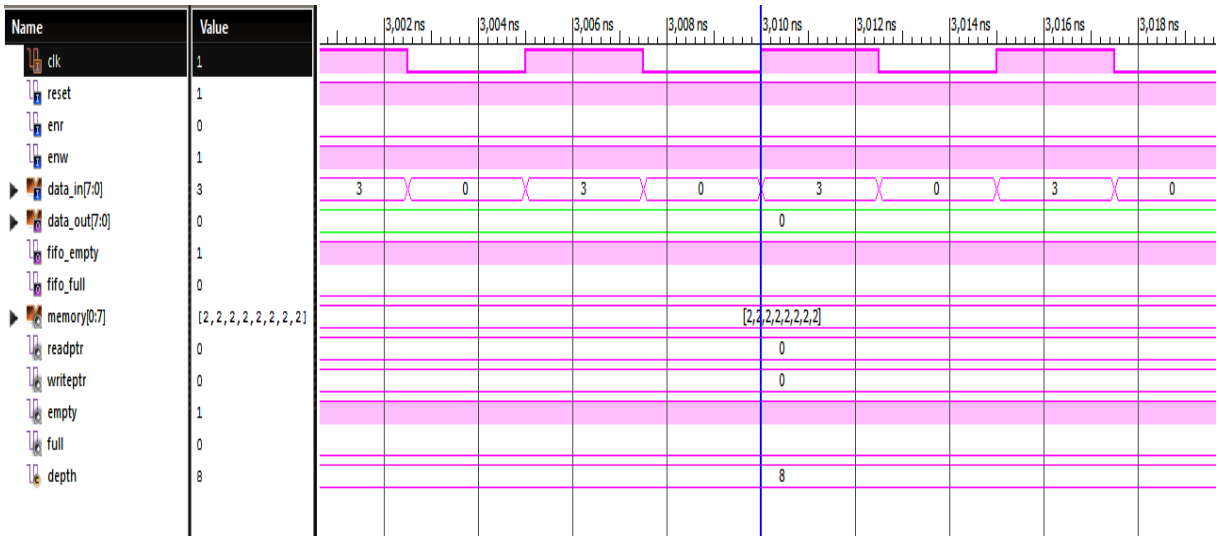


Timing diagram for READ operation:



```
end fifo;
architecture Behavioral of fifo is
type memory_type is array (0 to depth-1) of std_logic_vector(7 downto 0);
signal memory : memory_type :=(others => (others => '0'));   --memory for queue.
signal readptr,writeptr : integer := 0;  --read and write pointers.
signal empty,full : std_logic := '0';
begin
fifo_empty <= empty;
fifo_full <= full;
process(Clk,reset)
--this is the number of elements stored in fifo at a time.
--this variable is used to decide whether the fifo is empty or full.
variable num_elem : integer := 0;
begin
if(clk'event and clk='1') then
if(reset = '1') then
   data_out <= (others => '0');
   empty <= '0';
   full <= '0';
   readptr <= 0;
   writeptr <= 0;
   num_elem := 0;
elsif(enr = '1' and empty = '0') then  --read
       data_out <= memory(readptr);
       readptr <= readptr + 1;
       num_elem := num_elem-1;
   end if;
   if(enw ='1' and full = '0') then    --write
       memory(writeptr) <= data_in;
       writeptr <= writeptr + 1;
       num_elem := num_elem+1;
```

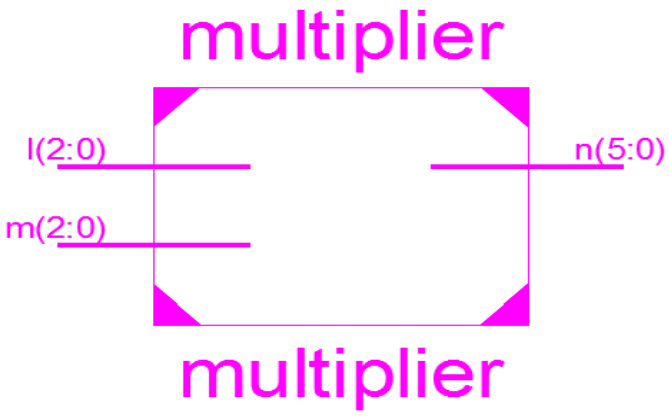
Timing diagram for reset operation :



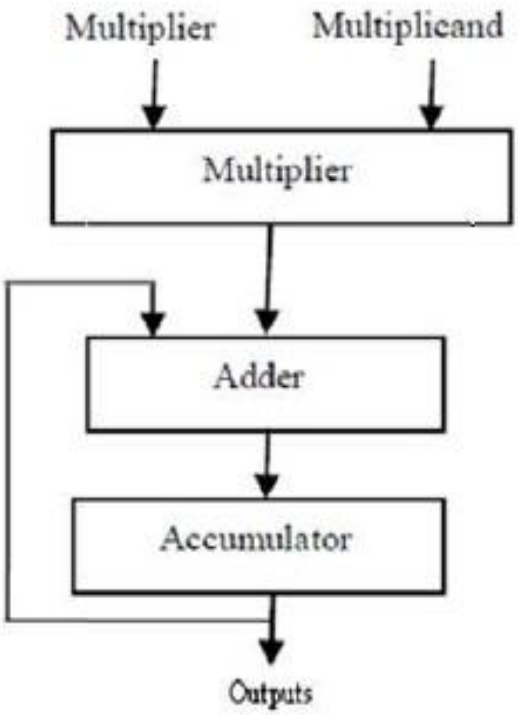
```
end if;
   --rolling over of the indices.
   if(readptr = depth-1) then      --resetting read pointer.
      readptr <= 0;
   end if;
   if(writeptr = depth-1) then       --resetting write pointer.
      writeptr <= 0;
   end if;
   --setting empty and full flags.
   if(num_elem = 0) then
      empty <= '1';
   else
      empty <= '0';
   end if;
   if(num_elem = depth) then
      full <= '1';
   else
      full <= '0';
   end if;
end if;
end process;
end Behavioral;
```

RESULT :

BLOCK DIAGRAM:



INTERNAL DIAGRAM:



EXP NO:

MAC (Multiplier & Accumulator)

Date:

ABSTRACT: To study and simulate MAC using VHDL.

THEORY: the multiply–accumulate operation is a common step that computes the product of two numbers and adds that product to an accumulator. The hardware unit that performs the operation is known as a multiplier–accumulator (MAC, or MAC unit)

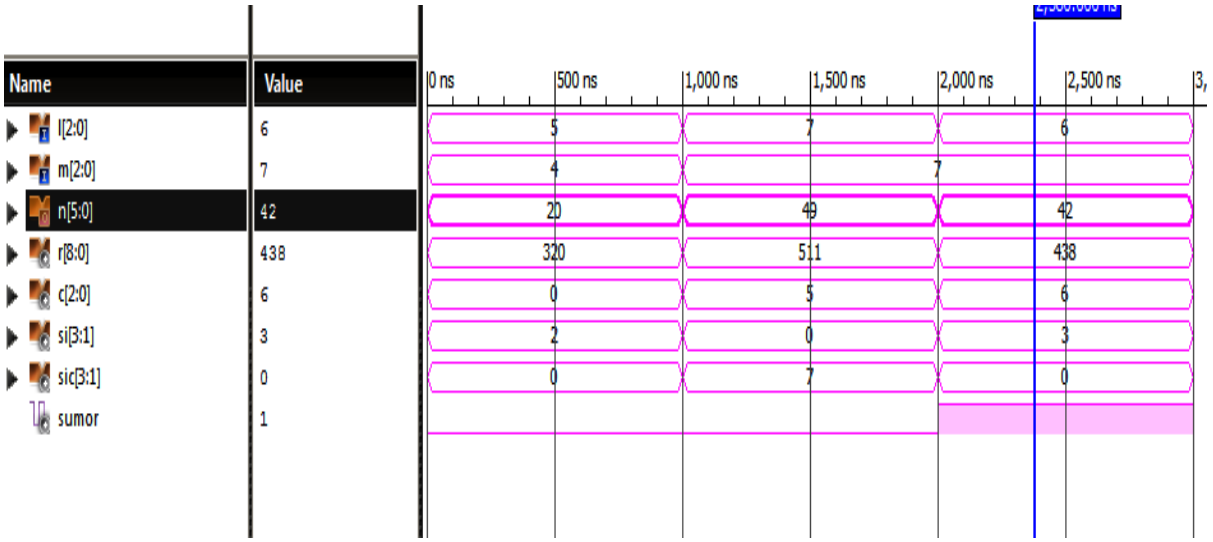
PROCEDURE:

- The MAC Design is entered through VHDL.
- The design is simulated by applying test vectors- l, m and observing output n.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR MAC:

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

entity mac is
Port ( l,m : in STD_LOGIC_VECTOR (2 downto 0);
       n : out STD_LOGIC_VECTOR (5 downto 0));
end mac;
architecture Behavioral of mac is
component and1
Port (  a,b : in STD_LOGIC;
c : out STD_LOGIC);
end component;
component hadder
port( i,j : in STD_LOGIC;
      su,ca : out STD_LOGIC);
end component;
component fulladd
Port (  d,e,f : in STD_LOGIC;
sum,carry : out STD_LOGIC);
end component;
component or1
Port ( a1,b1 : in STD_LOGIC;
c1 : out STD_LOGIC);
end component;
--signal p,q: std_logic;
signal r : std_logic_vector( 8 downto 0);
signal c : std_logic_vector ( 2 downto 0);
signal si: std_logic_vector(3 downto 1);
```

TIMING WAVEFORM:



```
signal sic: std_logic_vector(3 downto 1);
signal sumor: std_logic;
begin
l0: and1 port map ( l(0),m(0),r(0));
 l1: and1 port map ( l(1),m(0),r(1));


l2: and1 port map ( l(2),m(0),r(2));
l3: and1 port map ( l(0),m(1),r(3));
l4: and1 port map ( l(1),m(1),r(4));
l5:and1 port map  ( l(2),m(1),r(5));
l6: and1 port map ( l(0),m(2),r(6));
l7: and1 port map ( l(1),m(2),r(7));
l8: and1 port map ( l(2),m(2),r(8));
n(0)<= r(0);
m1: hadder port map (r(1),r(3),n(1),c(0));
m2: hadder port map (c(0),r(2),si(1),sic(1));
m3: hadder port map (r(4),r(6),si(2),sic(2));
m4: hadder port map (si(1),si(2),n(2),c(1));
m5: hadder port map (sic(1),sic(2),si(3),sic(3));m8: or1 port map (c(1),si(3),sumor);
m6: fulladd port map (sumor,r(5),r(7),n(3),c(2));
m7: fulladd port map (c(2),sic(3),r(8),n(4),n(5));
end Behavioral;
```

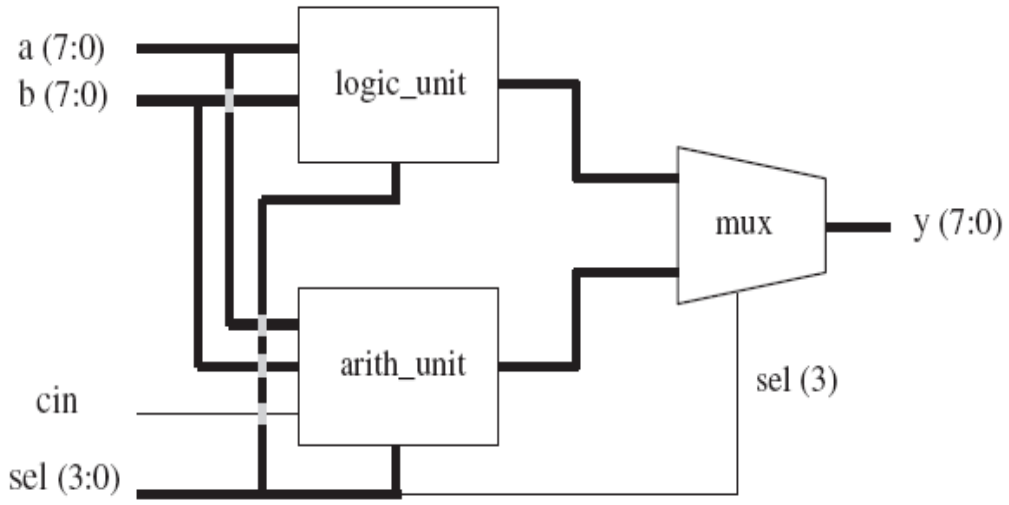
LINKUP FOR STRUCTURAL MODELLING

```
//AND GATE//

entity and1 is
Port ( a,b : in STD_LOGIC;c : out STD_LOGIC);
end and1;
architecture Behavioral of and1 is
begin
c<= a and b;
end Behavioral;

//OR GATE//
entity or1 is
Port ( a1,b1 : in STD_LOGIC;c1 : out STD_LOGIC);
end or1;
architecture Behavioral of or1 is
begin
c1<= a1 or b1;
end Behavioral;

//HALF ADDER//
entity hadder is
Port ( i,j : in STD_LOGIC;
       su,ca : out STD_LOGIC);
end hadder;
```

```
architecture Behavioral of hadder is
begin
su<= i xor j;ca<= i and j;
end Behavioral;

//FULL ADDER//
entity fulladd is
Port ( d,e,f : in STD_LOGIC;sum,carry : out STD_LOGIC);

end fulladd;
architecture Behavioral of fulladd is
component and1Port ( a,b : in STD_LOGIC;c : out STD_LOGIC);
end component;
begin
sum <= d xor e xor f;carry <= ((d and e) or ( e and f ) or ( f and d));
end Behavioral;
```

RESULT:

LOGIC DIAGRAM:



TRUTH TABLE:

sel	Operation	Function	Unit
0000	y <= a	Transfer a	
0001	y <= a+1	Increment a	
0010	y <= a-1	Decrement a	
0011	y <= b	Transfer b	Arithmetic
0100	y <= b+1	Increment b	
0101	y <= b-1	Decrement b	
0110	y <= a+b	Add a and b	
0111	y <= a+b+cin	Add a and b with carry	
1000	y <= NOT a	Complement a	
1001	y <= NOT b	Complement b	
1010	y <= a AND b	AND	
1011	y <= a OR b	OR	Logic
1100	y <= a NAND b	NAND	
1101	y <= a NOR b	NOR	
1110	y <= a XOR b	XOR	
1111	y <= a XNOR b	XNOR	

EXP NO:

Date:

ARITHMETIC AND LOGIC UNIT

ABSTRACT: To develop the source code for arithmetic and logic unit by using VHDL and obtain the simulation, synthesis, place and route and implement into FPGA.

THEORY:

The 74181 can be modelled as above. Recognizing the logic that makes up a CLA block in this case, the circled elements in the gate-level schematic is the key step in unravelling the secrets of the 74181. The four boxed circuits in the gate-level schematic are represented above by the single module M1 with 4-bit I/O buses. The second quadruplicated circuit in the 74181 leads to the high-level module M2. The various XOR gates are also grouped into 4-bit word gates as indicated above. Further analysis shows that the 74181 is original designers cleverly constructed the M_1 and M_2 logic so that with input line $M = 1$, each setting of the S (function select) bus produces one of the 16 possible Boolean functions of the form $F(A,B)$.

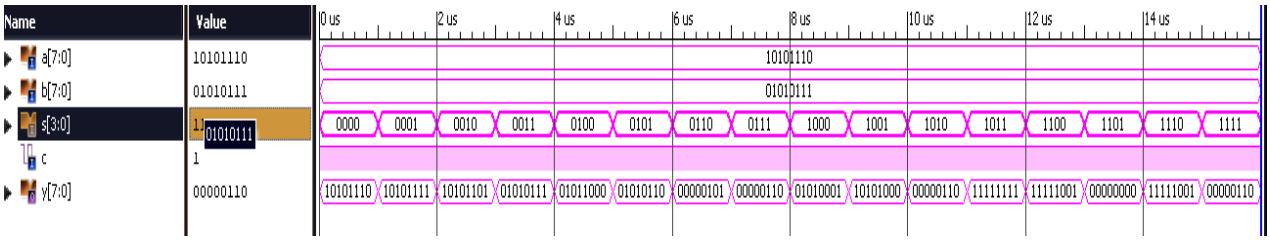
PROCEDURE:

- The ALU Design is entered through VHDL.
- The design is simulated by applying test vectors- a,b, s and observing output y.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR ALU:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity alu is

   Port ( a,b : in  STD_LOGIC_VECTOR (7 downto 0);

          s : in  STD_LOGIC_VECTOR (3 downto 0);

          c : in  STD_LOGIC;

          y : out  STD_LOGIC_VECTOR (7 downto 0));

end alu;
```

TIMING DIAGRAM:

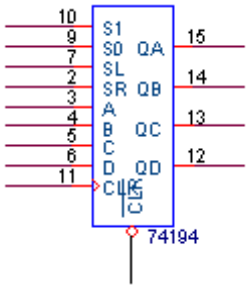


```
architecture dataflow of alu is
begin
with s select
y<=     a when "0000",
                a+1 when "0001",
                a-1 when "0010",
                b when "0011",
                b+1 when "0100",
                b-1 when "0101",
                a+b when "0110",
                a+b+c when "0111",
                not a when "1000",
                not b when "1001",
                a and b when "1010",
                a or b when "1011",
                a nand b when "1100",
                a nor b when "1101",
                a xor b when "1110",
                a xnor b when "1111",
                "00000000" when others;
end dataflow;
```

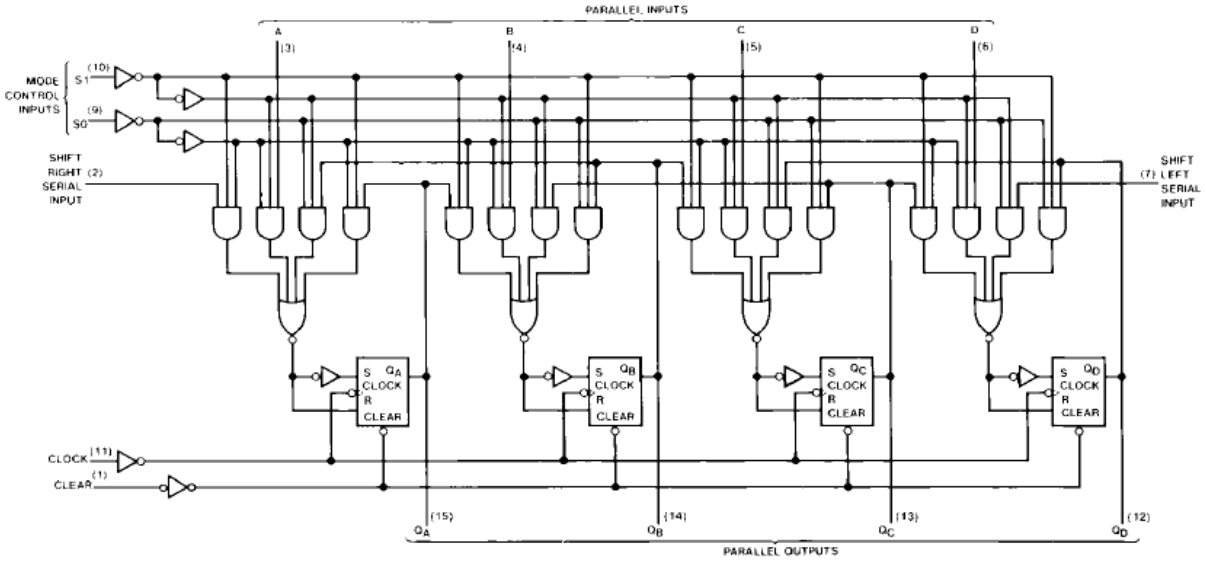
RESULT:

ADDITIONAL EXPERIMENTS

PIN DIAGRAM:



CIRCUIT DIAGRAM:



TRUTH TABLE OF IC 74194:

INPUTS			NEXT STATE			
FUNCTION	S1	S0	QA*	QB*	QC*	QD*
HOLD	0	0	QA	QB	QC	QD
SHIFT RIGHT	0	1	RIN	QA	QB	QC
SHIFT LEFT	1	0	QB	QC	QD	LIN
LOAD	1	1	A	B	C	D

EXP NO:

UNIVERSAL SHIFT REGISTER (IC 74194)

Date:

ABSTRACT: To study and simulate IC 74194using VHDL.

THEORY: The IC 74194 is an MSI 4-bit bi-directional parallel-in and parallel-out shift-register. The `194 is called bi-directional shift-register because they shift in either of two directions, depending on the control input. The directions are called left and right. In the'194the left means "in the direction from Q_A to Q_D, Right means "in the direction from Q_D to Q_A.

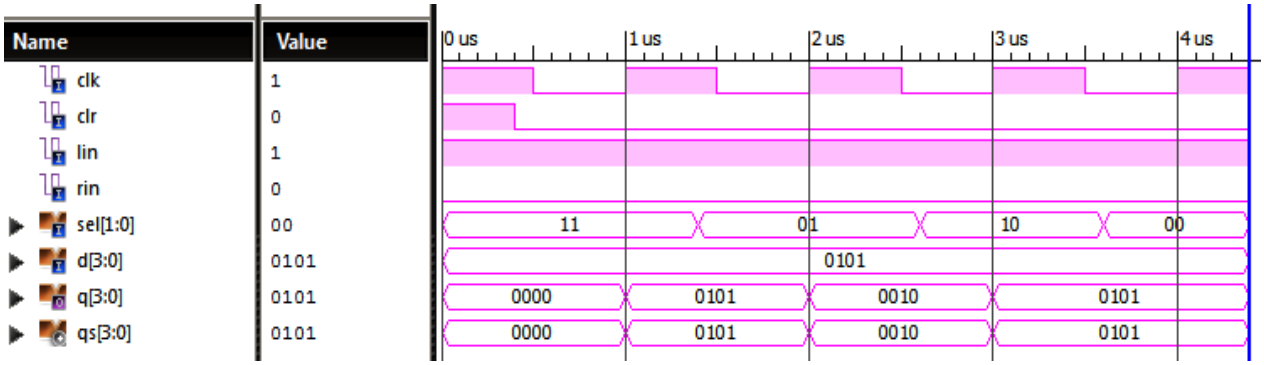
PROCEDURE:

- The IC 74194 Design is entered through VHDL.
- The design is simulated by applying test vectors- i, clk and observing output d.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR UNIVERSAL SHIFT REGISTER (IC 74194):

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ic74194 is

   Port ( clk,clr,lin,rin: in  STD_LOGIC;

          Sel:in std_logic_vector (1 downto 0);

         d : in  STD_LOGIC_VECTOR (3 downto 0);

         q : out  STD_LOGIC_VECTOR (3 downto 0));

end ic74194;

architecture Behavioral of ic74194 is

signal qs:std_logic_vector(3 downto 0);
```

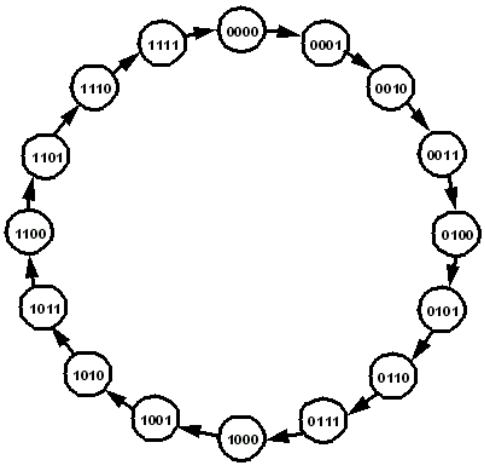
TIMING DIAGRAM:



```
begin
process(clk,clr,rin,lin,sel,d,qs)
begin
if (clr='1') then qs<=(others=>'0');
elsif(clk'event and clk='1') then
case sel is
 when "00"=>null;
 when "01"=>qs<=rin & qs(3 downto 1);
 when "10"=>qs <=qs(2 downto 0) & lin;
 when "11"=>qs<=d;
 when others=>null;
end case;
end if;
q<=qs;
end process;
end Behavioral;
```

RESULT:

State Diagram



TRUTH TABLE OF IC7493:

COUNT	Q_0	Q_1	Q2	Q3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	H	H	L
7	H	H	H	L
8	L	L	L	H
9	H	L	L	H
10	L	H	L	H
11	H	H	L	H
12	L	L	H	H
13	H	L	H	H
14	L	H	H	H
15	H	H	H	H

EXP NO:

Date:

4-BIT COUNTER (IC 7493)

ABSTRACT: To study and simulate IC 7493 using VHDL.

THEORY: the IC 7493 counts 0 to 15.When sixteenth pulse reaches it reset back to 0.

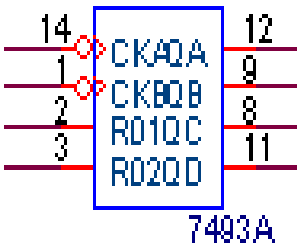
PROCEDURE:

- The IC 7493 Design is entered through VHDL.
- The design is simulated by applying test vector-clk and observing output count.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

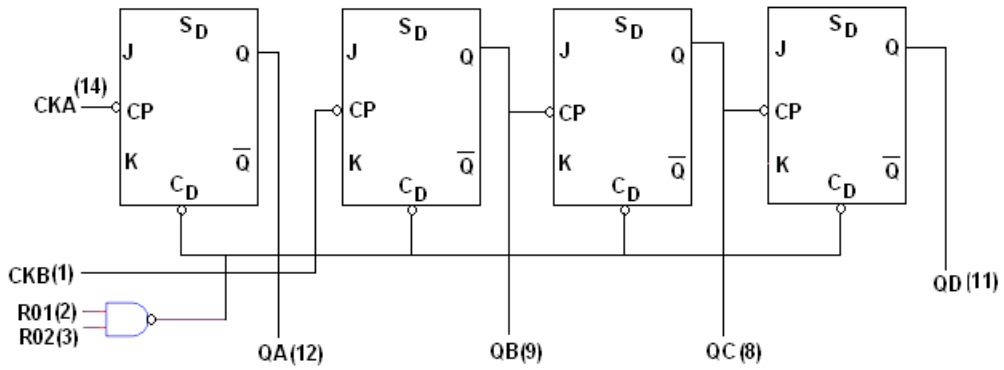
VHDL PROGRAM FOR 4-BIT BINARY COUNTER (IC7493):

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ic7493_a is

generic(n:NATURAL:=4);

port (clk : in  STD_LOGIC;

     reset : in  STD_LOGIC;

    count : out STD_LOGIC_VECTOR (n-1 downto 0));

end ic7493_a;

architecture Behavioral of ic7493_a is
```

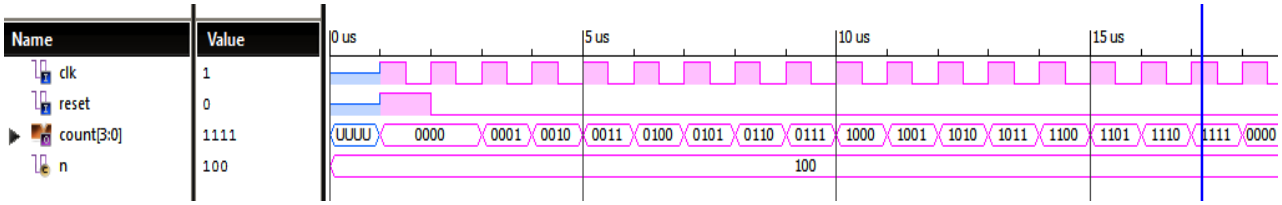
PIN DIAGRAM:



SCHEMATIC DIAGRAM:



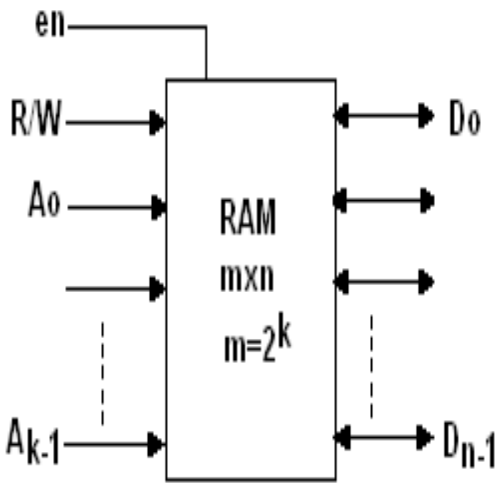
TIMING DIAGRAM:



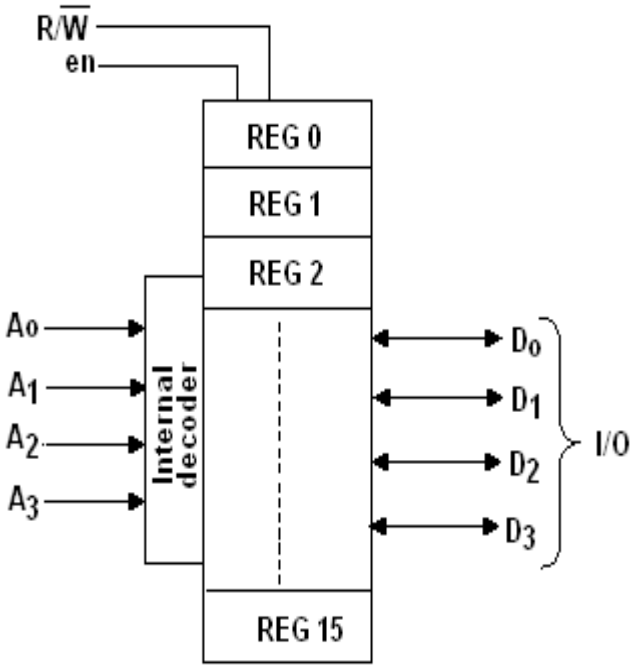
```
begin
        p0:process(clk,reset) is
        variable cnt: unsigned (n-1 downto 0);
        begin
        if reset='1' then
        cnt:=(others=>'0');
        elsif rising_edge(clk)then
        cnt:=cnt + 1;
        end if;
        count <= std_logic_vector(cnt);
        end process p0;
        end Behavioral;
```

RESULT:

BLOCK DIAGRAM:



16X4 RAM CHIP :



EXP NO:

Date:

RAM 16X4 (IC 74189)

ABSTRACT: To study and simulate design of RAM 16X4 using VHDL.

THEORY: The R/W memory is made of registers that store bits of information. The no.of bits stored in a register is called a memory word. This memory is used as a writing pad to write user programs and data. The information stored in this memory can be read and altered easily.

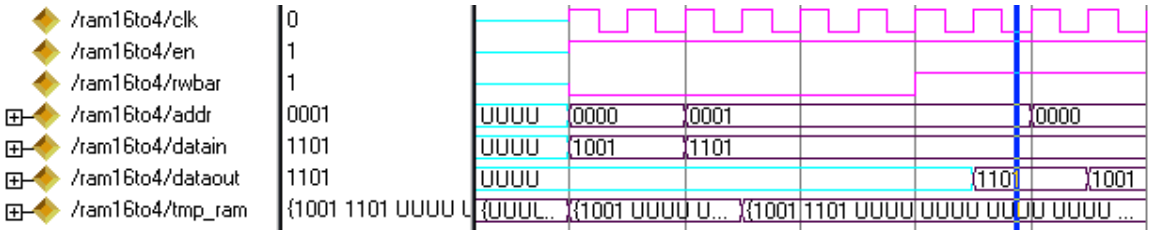
PROCEDURE:

- The RAM16X4 Design is entered through VHDL.
- The design is simulated by applying test vectors-clk,en,rwbar,addr,datain and observing output dataout.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

VHDL PROGRAM FOR 16 X 4 RAM:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ram16to4 is

   Port ( clk,en,rwbar : in  STD_LOGIC;

                  addr : in  STD_LOGIC_VECTOR (3 downto 0);

                datain : in  STD_LOGIC_VECTOR (3 downto 0);

               dataout : out  STD_LOGIC_VECTOR (3 downto 0));

end ram16to4;

architecture Behavioral of ram16to4 is
```

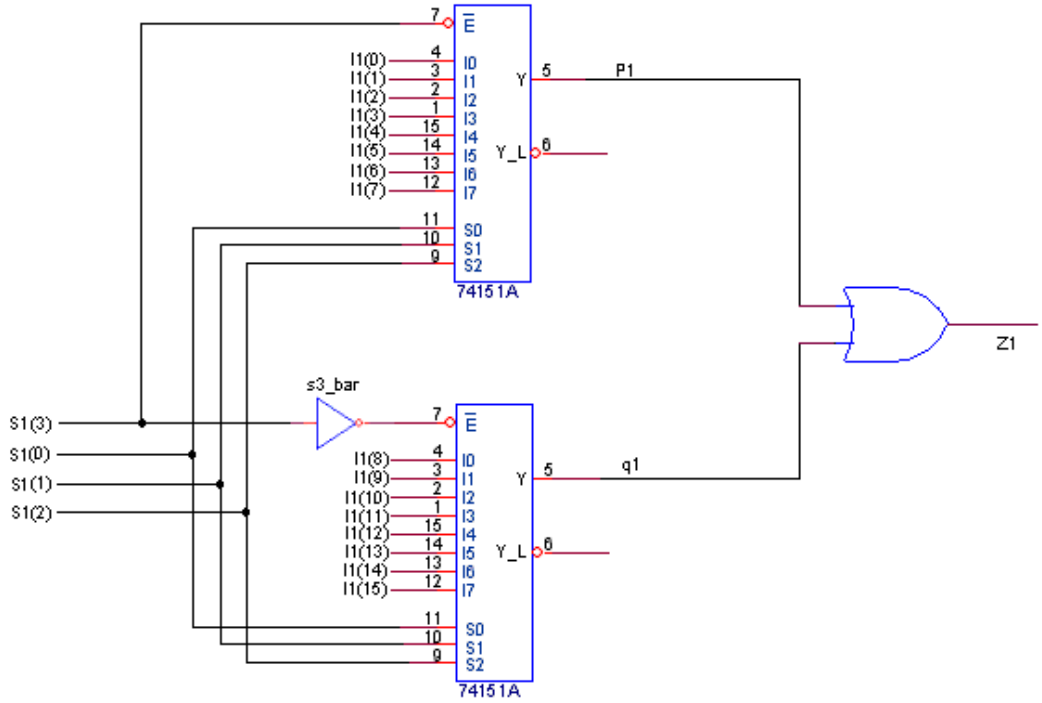
TIMING DIAGRAM:



```
type ram_type is array (0 to 15) of std_logic_vector(3 downto 0);
        signal tmp_ram:ram_type;
        begin
        process(clk,rwbar)
        begin
        if(clk='1' and clk'event) then
                if(en='1') then
                        if(rwbar='1') then
                                dataout<= tmp_ram(conv_integer(addr));
                        elsif(rwbar='0')then
                        tmp_ram(conv_integer(addr))<=datain;
                else
                        dataout<=(dataout'range=>'Z');
                end if;
        else
                dataout<=(dataout'range=>'Z');
        end if;
        end if;
        end process;
        end Behavioral;
```

RESULT:

CIRCUIT DIAGRAM:



EXP NO:

Date:

16 TO 1 MULTIPLEXER USING IC 74151(IC 74150)

ABSTRACT: To study and simulate design of IC 74150 using VHDL.

THEORY: Multiplexer IC 74151 is 8 to 1 multiplexer. It has 8 inputs and only one output based on the select inputs A, B, C it steers one of the input to the output Y. In 16 to 1 multiplexer design we use two 74151 IC's. The design uses four select i/p's $S_1(0)$, $S_1(1)$, $S_1(2)$, $S_1(3)$.The three select i/p's $S_1(0)$, $S_1(1)$, $S_1(2)$ are connected to each individual IC74151. The fourth select i/p & its complement are given to the top mux is selected & otherwise the bottom mux is selected.

PROCEDURE:

- The IC 74150 Design is entered through VHDL.
- The design is simulated by applying test vectors-s1, i1 and observing output z1.
- After simulation obtain the RTL, technology schematics and synthesis report.
- It is required to lock the pins and give timing constraints.
- Implement the design by passing the design by various stages by mapping, time analysis and bit stream. For locking the pins write UCF file before implementation and guide the same through option set control files. Output can be directly programmed into target device FPGA.

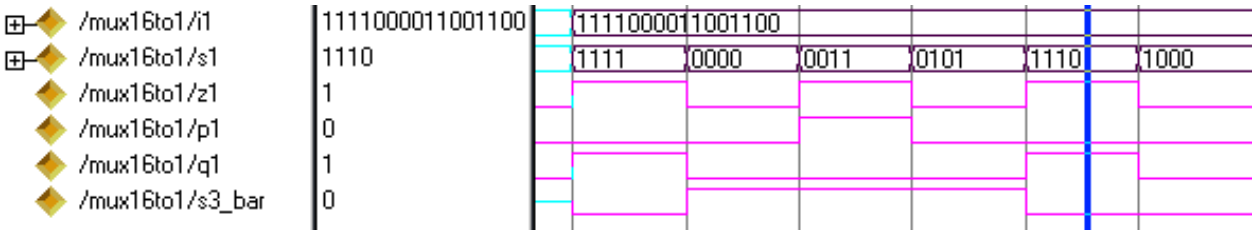
VHDL PROGRAM FOR 16 TO 1 MULTIPLEXER (IC74150):

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity mux16to1 is

   Port ( i1 : in  STD_LOGIC_VECTOR (15 downto 0);

          s1 : in  STD_LOGIC_VECTOR (3 downto 0);

          z1 : out  STD_LOGIC);

end mux16to1;

architecture structural of mux16to1 is

signal p1,q1, s3_bar:std_logic;
```

TRUTH TABLE OF IC74150:

INPUTS				OUTPUT
S1(3)	S1(2)	S1(1)	S1(0)	Z1
0	0	0	0	I1(0)
0	0	0	1	I1(1)
0	0	1	0	I1(2)
0	0	1	1	I1(3)
0	1	0	0	I1(4)
0	1	0	1	I1(5)
0	1	1	0	I1(6)
0	1	1	1	I1(7)
1	0	0	0	I1(8)
1	0	0	1	I1(9)
1	0	1	0	I1(10)
1	0	1	1	I1(11)
1	1	0	0	I1(12)
1	1	0	1	I1(13)
1	1	1	0	I1(14)
1	1	1	1	I1(15)

TIMING DIAGRAM:




```
  component ic74151 is
Port (    i : in  STD_LOGIC_VECTOR (7 downto 0);
        en_l : in  STD_LOGIC;
           s : in  STD_LOGIC_VECTOR (2 downto 0);
           y : out  STD_LOGIC;
        y_l : inout  STD_LOGIC);
end component;
component orgate is
   Port ( a,b : in  STD_LOGIC;
            c : out  STD_LOGIC);
end component;
component notgate is
   Port ( a : in  STD_LOGIC;
            b : out  STD_LOGIC);
end component;
begin
l0: notgate port map (s1(3),s3_bar);
l1: ic74151 port map (i1(7 downto 0),s1(3),s1(2 downto 0),p1,open);
```

```
l2: ic74151 port map (i1(15 downto 8),s3_bar,s1(2 downto 0),q1,open);

l3: orgate  port map (p1,q1,z1);
end structural;
```

RESULT: