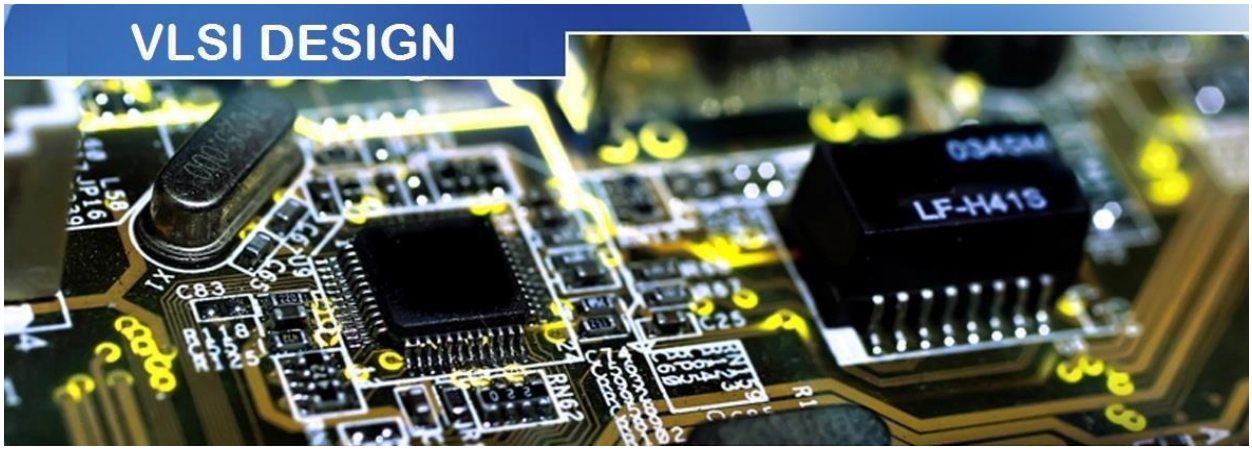
PHYSICAL DESIGN LAB MANUAL



Department of Electronics & Communication Engineering

J.N.N INSTITUTE OF ENGINEERING

90, USHAA GARDEN, KANNIGAIPAIR,THIRUVALLUR,TAMILNADU-601102

(NAAC 'A' Grade | Approved by AICTE | Affiliated to Anna University)

PHYSICAL DESIGN LAB MANUAL

PREPARED

BY

Department of Electronics & Communication Engineering

J.N.N INSTITUTE OF ENGINEERING

90, USHAA GARDEN, KANNIGAIPAIR,THIRUVALLUR,TAMILNADU-601102

(NAAC 'A' Grade | Approved by AICTE | Affiliated to Anna University)

J.N.N Institute Of Engineering
Department of Electronics and Communication Engineering

Vision of the institute

Lead the transformation of engineering and management learning experience to educate the next generation of innovators and entrepreneurs who want to make the world a better place.

Mission of the institute

Mission_1: To develop the required resources and infrastructure and to establish a conducive ambience for the teaching-learning process.

Mission_2: To nurture professional and ethical values in the students and to instil in them a spirit of innovation and entrepreneurship.

Mission_3: To encourage a desire for higher learning and research in the students and to equip them to face global challenges.

Mission_4: To provide opportunities for students to learn job-relevant skills to make them industry ready.

Mission_5: To interact with industries and other organisations to facilitate transfer of knowledge and know-how.

Vision of the department

Cultivating innovative and entrepreneurial Electronics and Communication Engineering graduates to ethically address global challenges through quality teaching and learning practices.

Mission of the department

Mission_1: To facilitate a state-of-the-art teaching-learning process, imparting comprehensive knowledge in electronics and communication engineering and related interdisciplinary areas.
Mission_2: To foster a sense of curiosity, critical thinking and ethical practices in students, preparing them for a continuous learning.
Mission_3: To instill innovative team work and industry collaboration for enhancing entrepreneurial skills, employability and research capabilities in graduates.
Mission_4: To inculcate ability for delivering novel solutions by taking social and environmental aspects into consideration.

Programme Outcomes(Pos)

PO_1	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO_2	**Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO_3	**Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO_4	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO_5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO_6	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO_7	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO_8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO_9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO_10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO_11	**Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO_12	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Programme Specific Outcome(PSOs)

PSO_1	**Electronic System Design/Analysis:** Apply the fundamental concepts of Electronics and Communication Engineering to design and analysis of Electronics Systems for applications including Signal Processing, Communication & Networking, Embedded Systems, VLSI design and Control Systems.
PSO_2	**Software Tools:** Proficiency in specialized software tools and computer programming useful for the design and analysis of complex electronic systems to meet challenges in contemporary business environment.

J.N.N INSTITUTE OF ENGINEERING

90, Ushaa Garden, Kannigaipair,Chennai-Periyapalayam Highway, Tiruvallur, Tamil Nadu 601102

Department of Electronics &Communication Engineering

PART A: FPGA Level Implementation

1. Realization of Logic Gates.
2. 4-bit ripple carry and carry look ahead adder using behavioral, dataflow and structural modeling
3. Design and Implementation of
   a) 16:1 mux through 4:1 mux
   b) 3:8 decoder realization through 2:4 decoder
4. Design and Implementation of 8:3 encoder
5. Design and Implementation of 8-bit parity generator checker
6. Design and Implementation of different Flip-Flops
7. Design and Implementation of 4-bit sequence detector through Mealy and Moore state machine

PART B*:* Back-end Level Design and Implementation

8. Design and Implementation of Universal Gates
9. Design and Implementation of an Inverter
10. Design and Implementation of Full Adder
11. Design and Implementation of Full Subtractor
12. Design and Implementation of D latch

PART C: Advanced Experiments

13. Design and Implementation of Differential Amplifier.
14. Design and Implementation of ALU

CONTENTS

DOS & DONTS IN LABORATORY

1. While entering the Laboratory, the students should follow the dress code Wear shoes, White Apron & Female students should tie their hair back).
2. The students should bring their observation note book, practical manual, record note book, calculator, necessary stationary items and graph sheets if any for the lab classes without which the students will not be allowed for doing the practical.
3. All the equipments and components should be handled with utmost care. Any breakage/damage will be charged.
4. If any damage/breakage is noticed, it should be reported to the instructor immediately.
5. If a student notices any short circuits, improper wiring and unusual smells immediately the same thing is to be brought to the notice of technician/lab in charge.
6. At the end of practical class the apparatus should be returned to the lab technician and take back the indent slip.
7. Each experiment after completion should be written in the observation note book and should be corrected by the lab in charge on the same day of the practical class.
8. Each experiment should be written in the record note book only after getting signature from the lab in charge in the observation note book.
9. Record should be submitted in the successive lab session after completion of the experiment.
10. 100% attendance should be maintained for the practical classes.

SCHEME OF EVALUATION

S.No	Program	Date	Marks Awarded				Total 30(M)
			Record (10M)	Obs. (10M)	Viva (5M)	Attd. (5M)	
PART-A							
1	Realization of Logic Gates.						
2	4-bit ripple carry and carry look ahead adder						
3.	Design and Implementation of A) 16:1 MUX through 4:1						
	B) 3:8 decoder realization 2:4 decoder						
4.	Design and Implementation of 8:3encoder						
5.	Design and Implementation of 8-bit parity generator and checker						
6.	Design and Implementation of different Flip-Flops						
7.	Design and Implementation of 4-bit sequence detector through Mealy and Moore state machine						
PART-B							
8.	Design and Implementation of Universal Gates						
9.	Design and Implementation of an Inverter						
10.	Design and Implementation of Full Adder						
11.	Design and Implementation of Full Subtractor						
12.	Design and Implementation of D latch						
Advanced Experiments							
13.	Design and Implementation of differentitional amplifier						
14.	Design and Implementation of ALU						

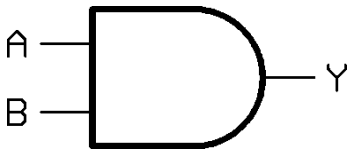
Signature of Lab In-charge

PART A

(FPGA Level Implementation)

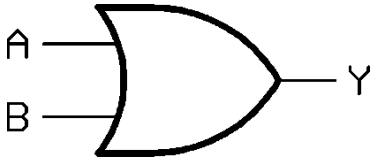
CIRCUIT DIAGRAM & TRUTH TABLES

AND GATE **TRUTH TABLE**



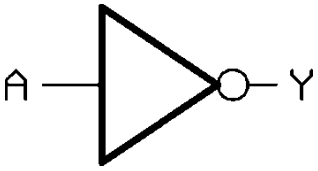
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR GATE



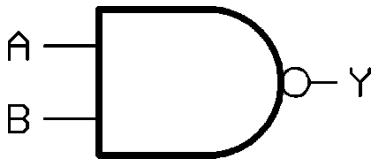
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT GATE



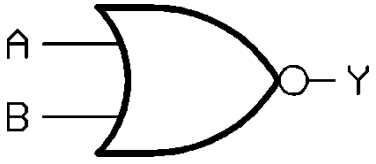
A	Y
0	1
1	0

NAND GATE



A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

NOR GATE



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

EXP NO.	REALIZATION OF ALL LOGIC GATES	DATE
1		

AIM:

To write a VHDL/Verilog code for All Logic Gates and to generate synthesis report, RTL schematic and to implement designs using FPGA (Spartan-3).

APPARATUS:

1. Synthesis Tool: Xilinx Project Navigator - ISE 9.1i.
2. Simulation Tool: Modelsim Simulator.

PROCEDURE:

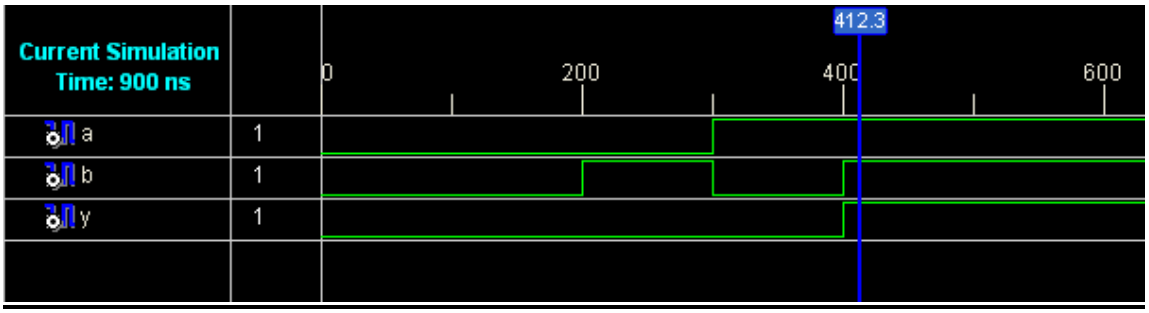
1. Open Xilinx ISE 9.1i.
2. Create a new source file in a new project with suitable name.
3. Create the file in VHDL/Verilog module.
4. Select the appropriate input and output ports according to the requirements.
5. Type the program and save it and synthesize the process.
6. Select Synthesize XST, check for syntax errors and generate report and RTL schematic.
7. Create another new source.
8. Select source type as Test bench wave form.
9. Associate the test bench to the source.
10. Assign clock and timing details.
11. Give the input waveforms for the source.
12. Save the input waveforms and perform behavioral simulation.
13. The simulated output waveforms window will be shown.

DUMPING PROCESS:

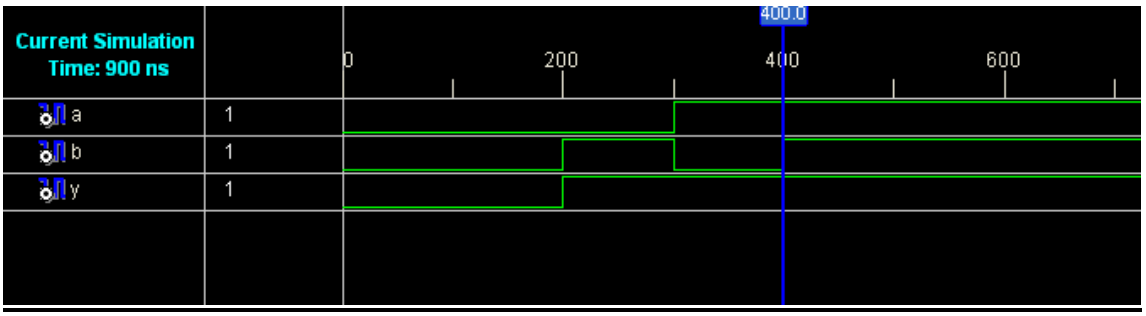
1. In the process window, go to 'user constraints' and select 'assign package pins' and after that double click on 'implement design'.
2. Select properties by right clicking on 'generate programming file'. Select 'JTAG' clock in startup options.
3. Select boundary scan in 'impact window' after double clicking on 'configure Device'.
4. In 'generate programming file' double clicking on 'programming file generation report. Bit file will be generated.
5. Xilinx boundary scan window will appear when the bit file is selected. Right click on Xilinx component and select program.
6. Programming properties will appear and finally program will be succeeded.
7. Thus the program can be dumped into FPGA kit and finally output can be seen on the kit.

SIMULATED MODEL OUTPUT WAVEFORMS

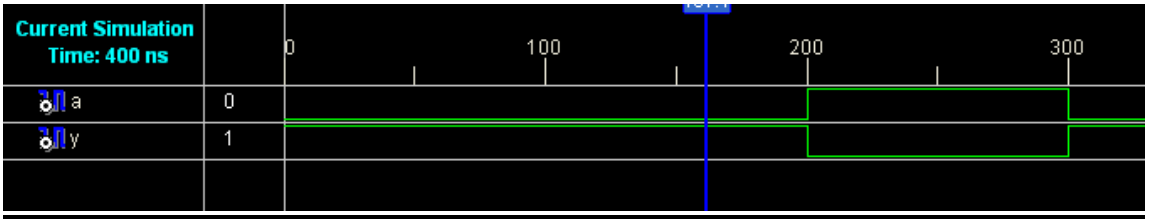
AND gate



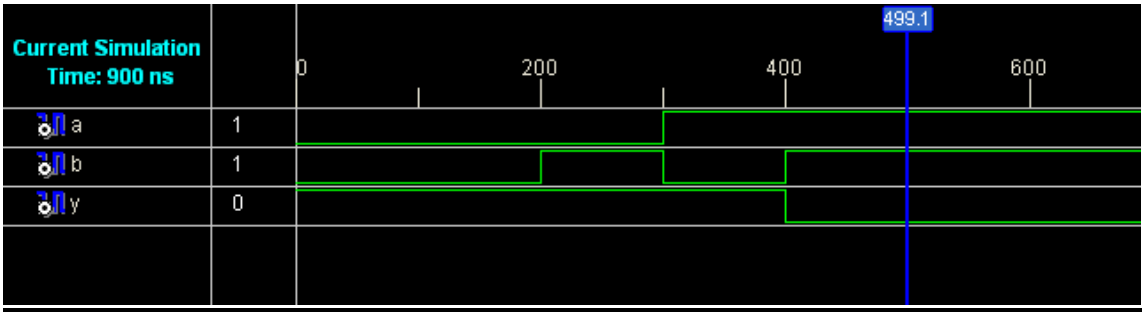
OR gate



NOT GATE



NAND gate



VHDL CODE:

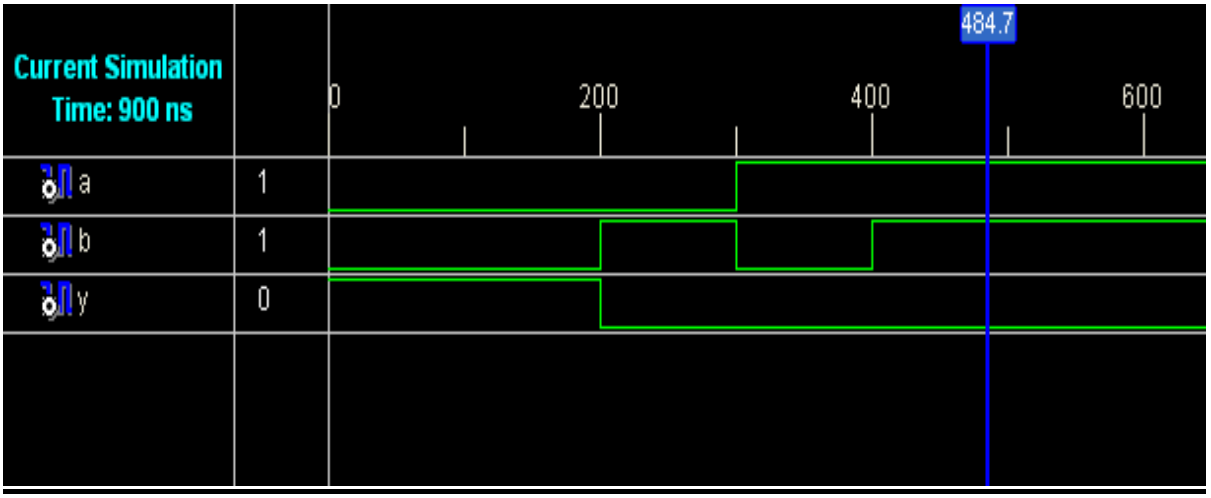
AND GATE :

```
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 : in  STD_LOGIC;
          b : in  STD_LOGIC;
          y : out  STD_LOGIC);
end Andgate  ;

architecture Behavioral of Andgate  is

begin

y <= a and b;

end Behavioral;
```

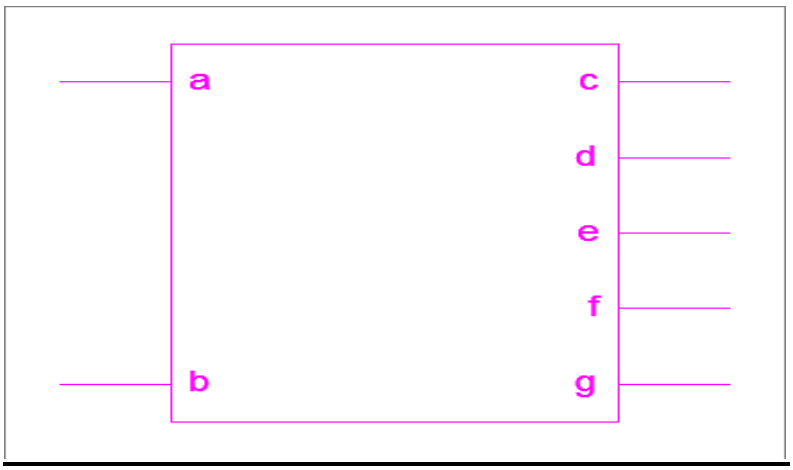
OR GATE :

```
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 : in  STD_LOGIC;
          b : in  STD_LOGIC;
          y : out  STD_LOGIC);
end orgate  ;

architecture Behavioral of  orgate  is

begin

y <= a or b;

end Behavioral;
```

NOR gate



BLOCK DIAGRAM:



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;
          y : out  STD_LOGIC);
end  notgate  ;
architecture Behavioral of notgate  is
begin
y <= not a;
end Behavioral;
```

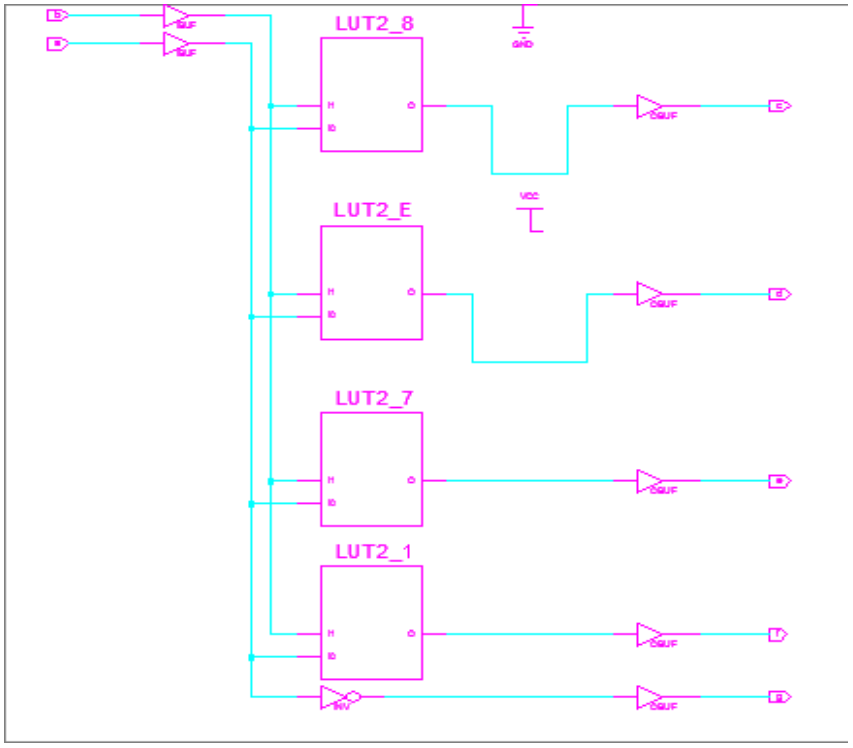
NAND GATE :

```
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 : in  STD_LOGIC;
          b : in  STD_LOGIC;
          y : out  STD_LOGIC);
end nandgate  ;
architecture Behavioral of nandgate  is
begin
y <= a nand b;
end Behavioral;
```

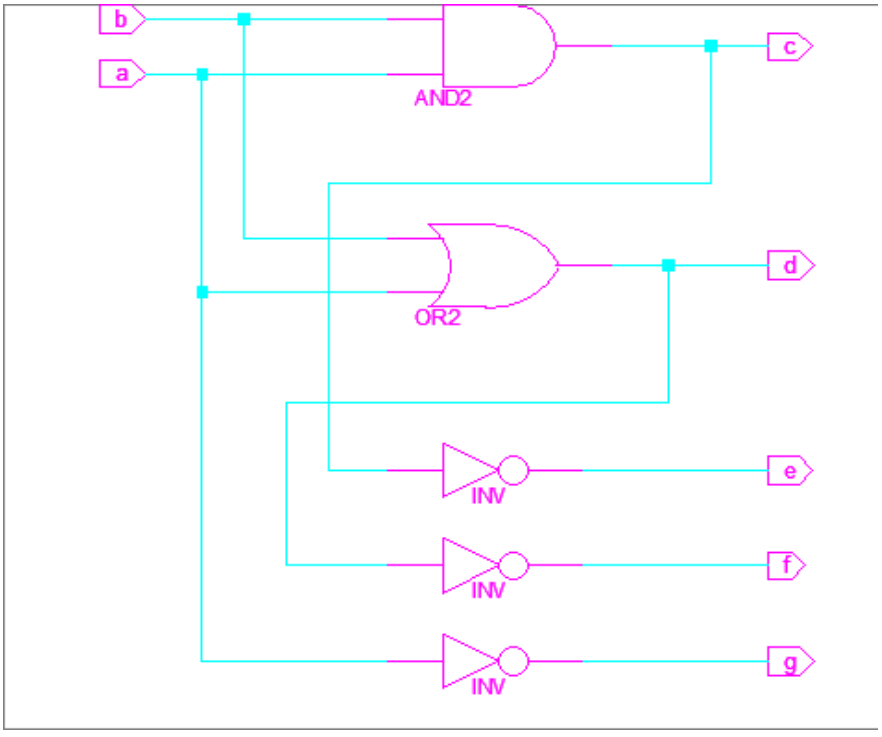
NOR GATE :

```
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 : in  STD_LOGIC;
          b : in  STD_LOGIC;
          y : out  STD_LOGIC);
end norgate  ;

architecture Behavioral of norgate  is
begin
y <= a nor b;
end Behavioral;
```

TECHNOLOGY SCHEMATIC:



RTL SCHEMATIC:



DESIGN SUMMARY:

Number of Slices	:	3 out of 960 0%
Number of 4 input LUTs	:	5 out of 1920 0%
Number of IOs	:	7
Number of bonded IOBs	:	7 out of 66 10%

SYNTHESIS REPORT:

RTL Top Level Output File Name : allgates.ngr
Top Level Output File Name : allgates
Output Format : NGC
Optimization Goal : Speed
Keep Hierarchy : NO
Design Statistics
IOs : 7
Cell Usage :
BELS : 5
INV : 1
LUT2 : 4
IO Buffers : 7
IBUF : 2
OBUF : 5

RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. Design all basic gates using 2:1 multiplexer?

2. Write the dataflow code for the logic gates

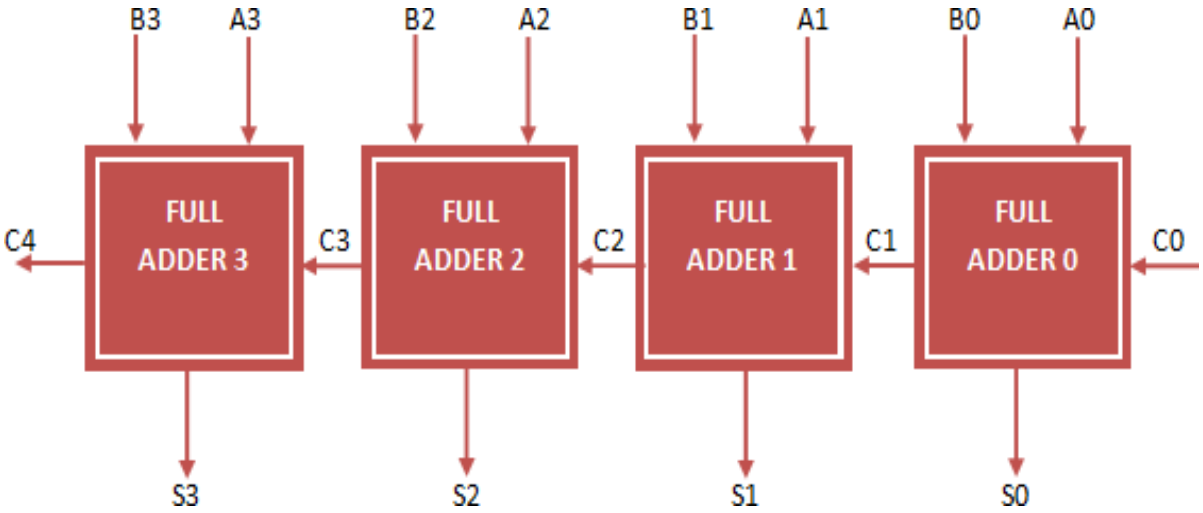
3. What are logic gates why the called so?

4. Which gates are called as universal gates? What are its advantages?

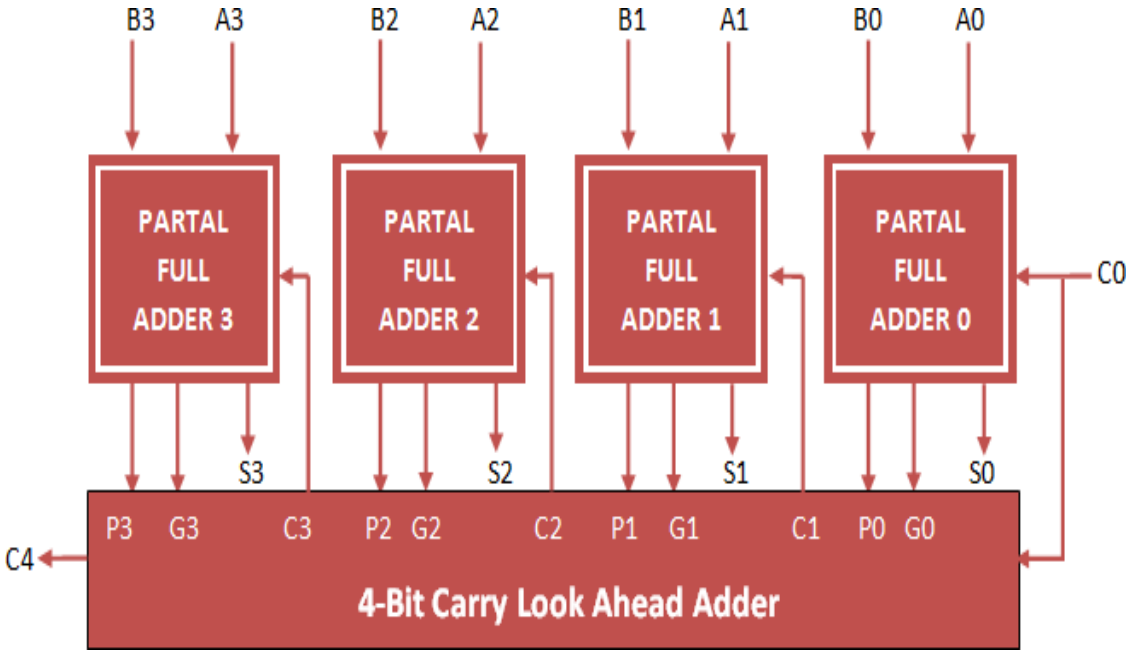
5. What are the applications of logic gates?

CIRCUIT DIAGRAM:

4-BIT RIPPLE CARRY ADDER



4-BIT CARRY LOOK AHEAD ADDER:



EXP NO.	4- BIT RIPPLE CARRYAND CARRY LOOK AHEAD ADDER	DATE
2		

AIM:

To write a VHDL/Verilog code for 4-bit ripple carry and carry look ahead adder and to generate synthesis report, RTL schematic and to implement designs using FPGA (Spartan-3).

APPARATUS:

1. Synthesis Tool: Xilinx Project Navigator - ISE 9.1i.
2. Simulation Tool: Modelsim Simulator.

PROCEDURE:

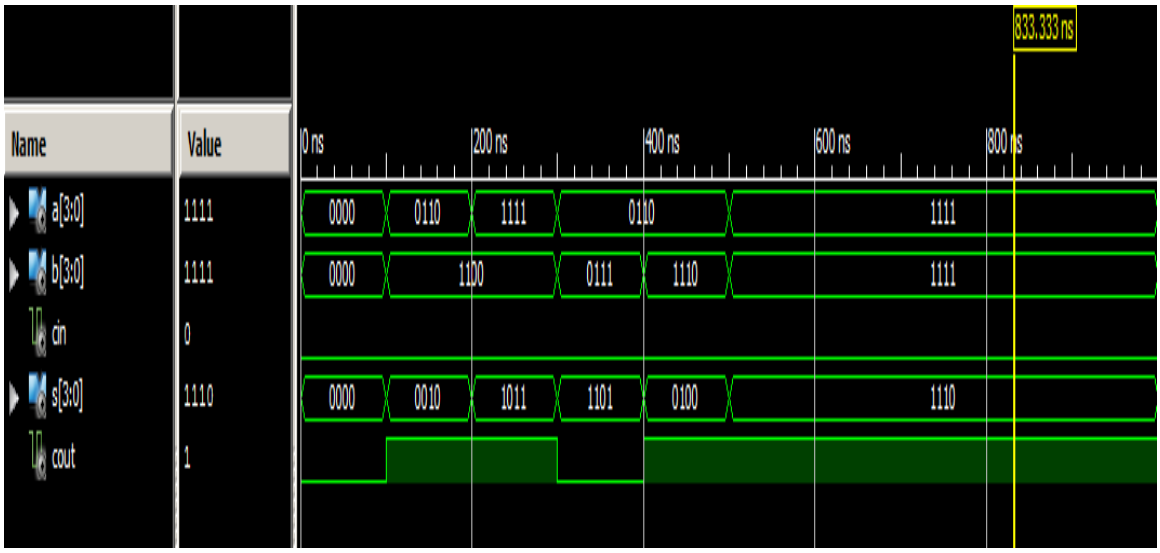
1. Open Xilinx ISE 9.1i.
2. Create a new source file in a new project with suitable name.
3. Create the file in VHDL/Verilog module.
4. Select the appropriate input and output ports according to the requirements.
5. Type the program and save it and synthesize the process.
6. Select Synthesize XST, check for syntax errors and generate report and RTL schematic.
7. Create another new source.
8. Select source type as Test bench wave form.
9. Associate the test bench to the source.
10. Assign clock and timing details.
11. Give the input waveforms for the source.
12. Save the input waveforms and perform behavioral simulation.
13. The simulated output waveforms window will be shown.

DUMPING PROCESS:

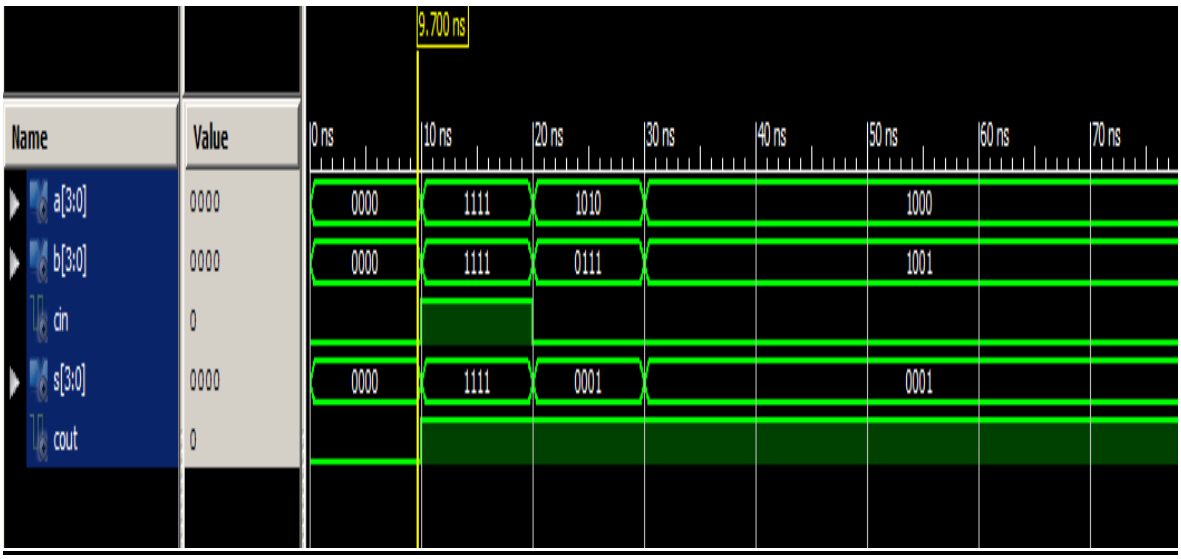
1. In the process window, go to ‘user constraints’ and select ‘assign package pins’ and after that double click on ‘implement design’.
2. Select properties by right clicking on ‘generate programming file’. Select ‘JTAG’ clock in startup options.
3. Select boundary scan in ‘impact window’ after double clicking on ‘configure device’.
4. In ‘generate programming file’ double clicking on ‘programming file generation report. Bit file will be generated.
5. Xilinx boundary scan window will appear when the bit file is selected. Right click on Xilinx component and select program.
6. Programming properties will appear and finally program will be succeeded.
7. Thus the program can be dumped into FPGA kit and finally output can be seen on the kit.

SIMULATED MODEL OUTPUT WAVEFORM:

4-BIT RIPPLE CARRY ADDER:



4-BIT CARRY LOOK AHEAD ADDER:



4-BIT RIPPLE CARRY ADDER VHDL CODE:

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

entity Ripple_Adder is
Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
 B : in STD_LOGIC_VECTOR (3 downto 0);
Cin : in STD_LOGIC;
S : out STD_LOGIC_VECTOR (3 downto 0); Cout : out STD_LOGIC);
end Ripple_Adder;

architecture Behavioral of Ripple_Adder is

component full_adder_vhdl_code Port ( A : in STD_LOGIC;
B : in STD_LOGIC;
Cin : in STD_LOGIC;
S : out STD_LOGIC;
Cout : out STD_LOGIC);
end component;
 signal c1,c2,c3: STD_LOGIC;

begin

FA1: full_adder_vhdl_code port map( A(0),B(0),Cin,S(0),c1);
FA2: full_adder_vhdl_code port map( A(1),B(1),c1,S(1),c2);
FA3: full_adder_vhdl_code port map( A(2),B(2),c2,S(2),c3);
FA4: full_adder_vhdl_code port map( A(3),B(3),c3,S(3),Cout);
end Behavioral;
```

4-BIT CARRY LOOK AHEAD ADDER VHDL CODE:

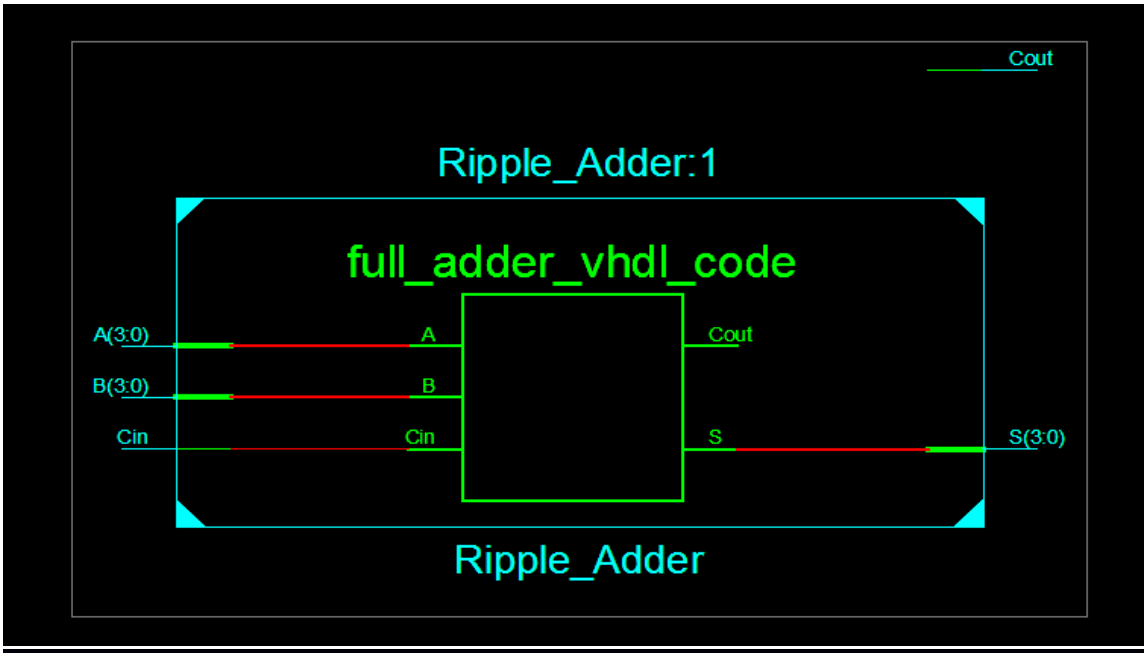
Partial Full Adder:

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

entity Partial_Full_Adder is
Port ( A : in STD_LOGIC;
B : in STD_LOGIC;
Cin : in STD_LOGIC;
S : out STD_LOGIC;
P : out STD_LOGIC;
G : out STD_LOGIC);
End  Partial_Full_Adder;

Architecture Behavioral of Partial_Full_Adder is
begin S <= A xor B xor Cin;
P <= A xor B;
G <= A and B;
end Behavioral;
```

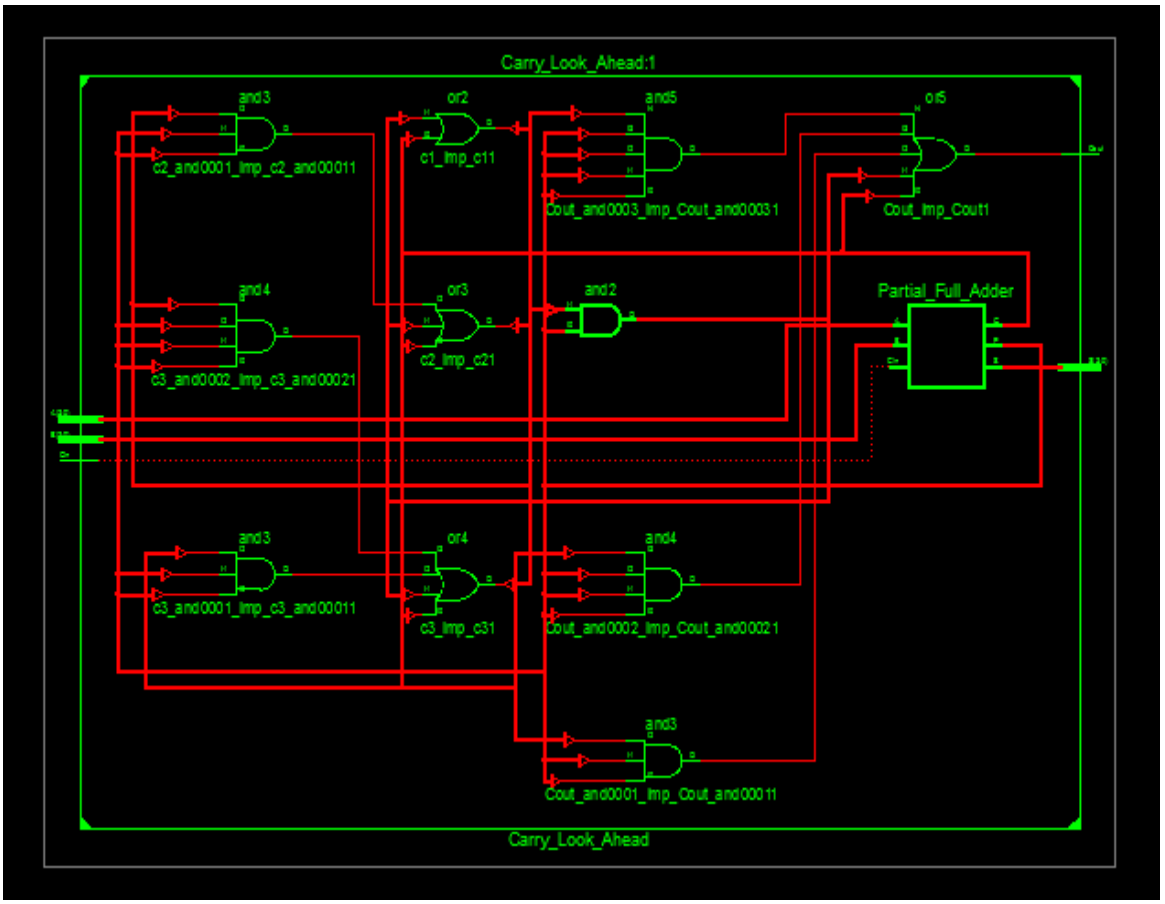
Carry Look Ahead Adder:

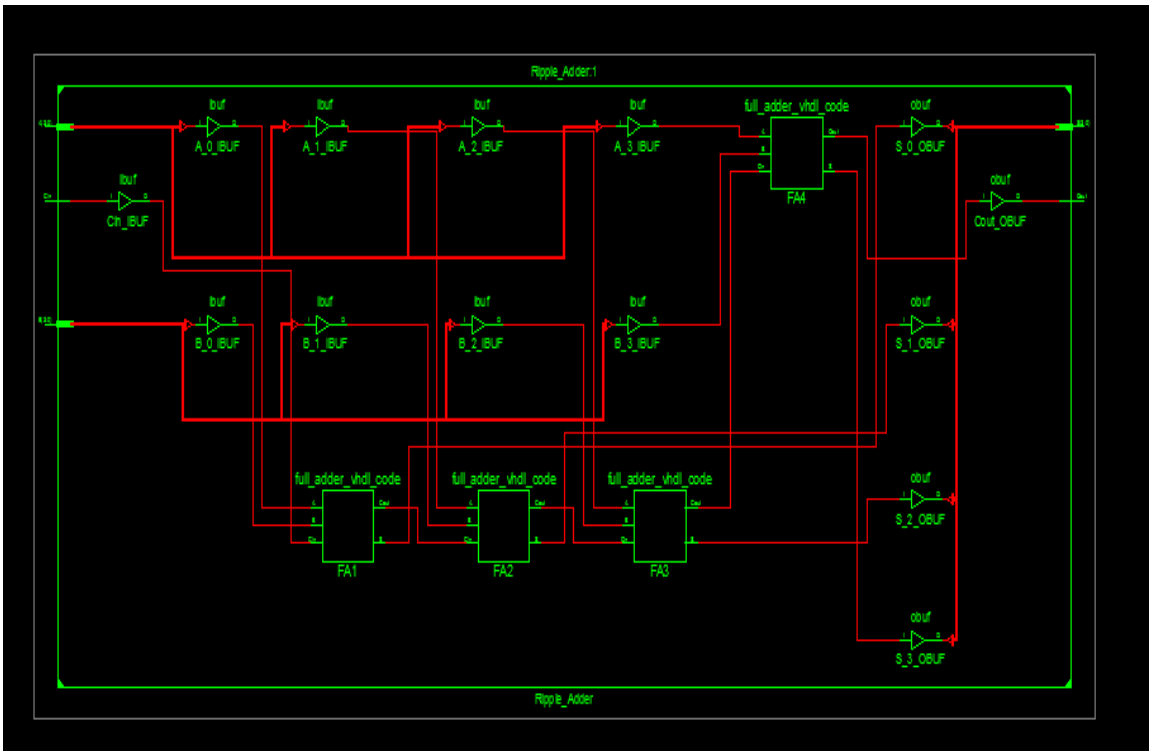
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Carry_Look_Ahead is
Port ( A : in STD_LOGIC_VECTOR (3 downto 0);
B: in STD_LOGIC_VECTOR (3 downto 0);
Cin : in STD_LOGIC;
S : out STD_LOGIC_VECTOR (3 downto 0);
Cout : out STD_LOGIC);
end Carry_Look_Ahead;
architecture Behavioral of Carry_Look_Ahead is component Partial_Full_Adder
Port ( A : in STD_LOGIC;
B : in STD_LOGIC;
Cin : in STD_LOGIC;
S : out STD_LOGIC;
P : out STD_LOGIC;
G : out STD_LOGIC);
end component;
signal c1,c2,c3: STD_LOGIC;
signal P,G: STD_LOGIC_VECTOR(3 downto 0);
begin
PFA1: Partial_Full_Adder port map( A(0), B(0), Cin, S(0), P(0), G(0));
PFA2: Partial_Full_Adder port map( A(1), B(1), c1, S(1), P(1), G(1));
PFA3: Partial_Full_Adder port map( A(2), B(2), c2, S(2), P(2), G(2));
PFA4: Partial_Full_Adder port map( A(3), B(3), c3, S(3), P(3), G(3));

c1 <= G(0) OR (P(0) AND Cin);
c2 <= G(1) OR (P(1) AND G(0)) OR (P(1) AND P(0) AND Cin);
c3 <= G(2) OR (P(2) AND G(1)) OR (P(2) AND P(1) AND G(0)) OR (P(2) AND P(1)
AND P(0) AND Cin);
Cout <= G(3) OR (P(3) AND G(2)) OR (P(3) AND P(2) AND G(1)) OR (P(3) AND
P(2) AND P(1) AND G(0)) OR (P(3) AND P(2) AND P(1) AND P(0) AND Cin);
end Behavioral;
```

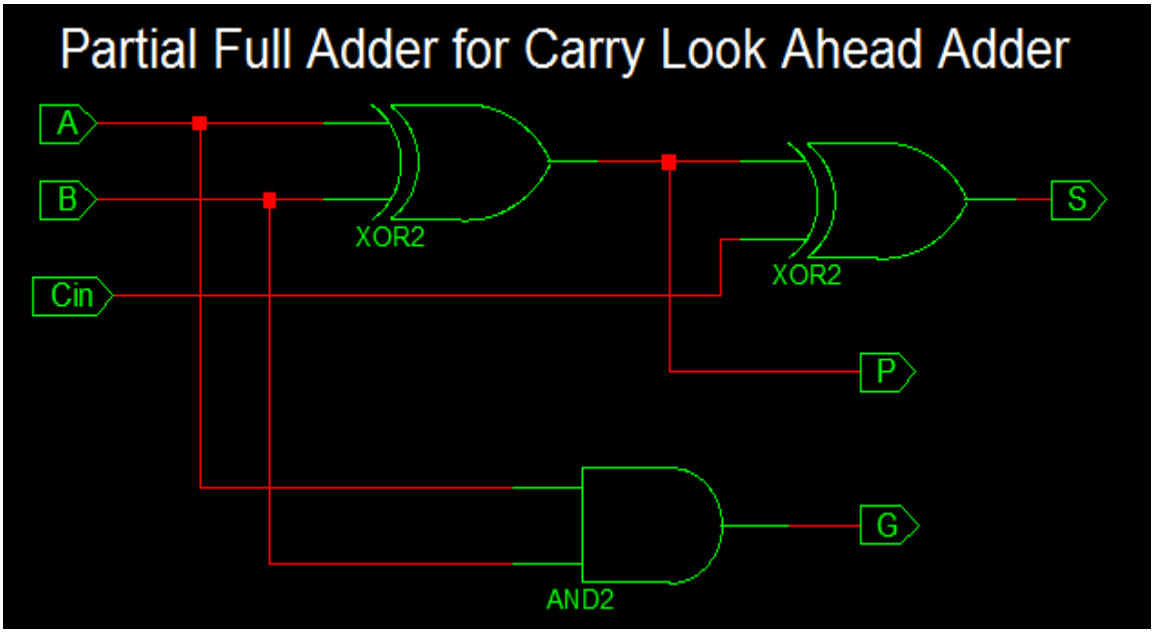
RTL SCHEMATIC:



TECHNOLOGY SCHEMATIC:




Ripple_Adder:1
Ibuf
A_0_IBUF
A_1_IBUF
A_2_IBUF
A_3_IBUF
Cin_IBUF
B_0_IBUF
B_1_IBUF
B_2_IBUF
B_3_IBUF
full_adder_vhdl_code
FA4
FA1
FA2
FA3
obuf
S_0_OBUF
Cout_OBUF
S_1_OBUF
S_2_OBUF
S_3_OBUF
Ripple_Adder


Partial Full Adder for Carry Look Ahead Adder
A
B
Cin
XOR2
XOR2
S
P
G
AND2

DEVICE UTILIZATION SUMMARY:

Selected Device: 3s400pq208-4

Number of Slices:	0	out of 3584	0%
Number of IOs:	14		
Number of bonded IOBs:	14	out of 141	9%

SYNTHESIS REPORT:

RTL Top Level Output File Name : Carry_Look_Ahead.ngr
Top Level Output File Name : Carry_Look_Ahead
Output Format : NGC
Optimization Goal : Speed
Keep Hierarchy : No

Design Statistics
IOs 14

Cell Usage :
BELS 4
LUT3 4
IO Buffers 14
IBUF 9
OBUF 5
Others 4
Partial_Full_Adder 4

Selected Device : 3s400pq208-4

Number of Slices	: 2	out of 3584	0%
Number of 4 input LUTs	: 4	out of 7168	0%
Number of IOs	: 14		
Number of bonded IOBs	: 14	out of 141	9%

SYNTHESIS REPORT

RTL Top Level Output File Name : Ripple_Adder.ngr
Top Level Output File Name : Ripple_Adder
Output Format : NGC
Optimization Goal : Speed
Keep Hierarchy : No

Design Statistics
IOs : 14

Cell Usage :
IO Buffers : 14
IBUF : 9
OBUF : 5
Others : 4
full_adder_vhdl_code : 4

RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. Why we use ripple carry adder?

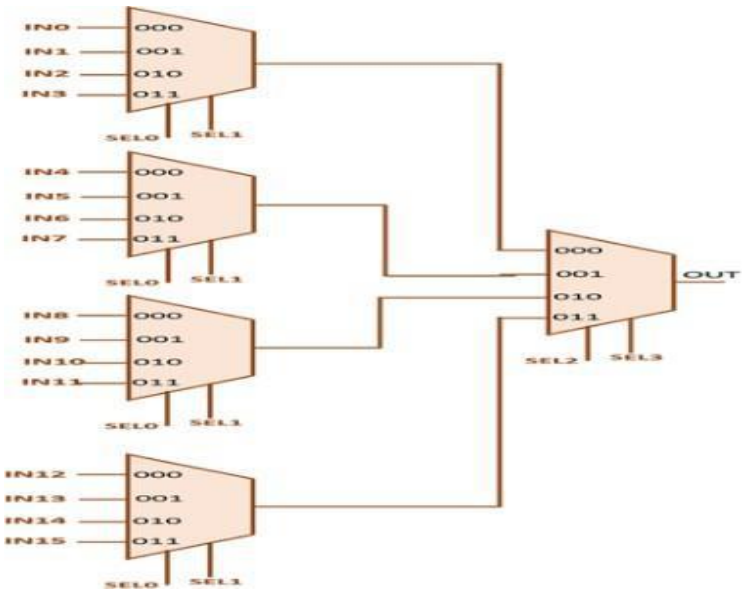
2. What is 4-bir carry look ahead adder?

3. What is delay of ripple carry adder?

4. Why CLA is better than RCA?

5. Is ripple carry adder and carry look ahead adder same?

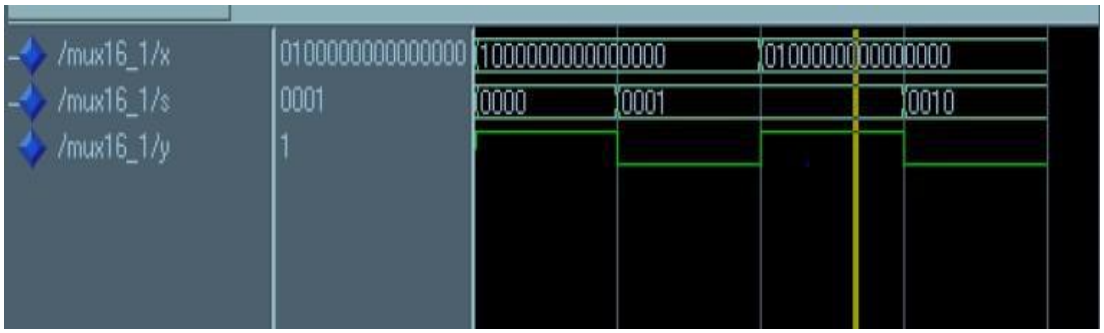
CIRCUIT16:1 MUX USING 4:1MUX



TRUTH TABLE:

INPUTS				Output
S_0	S_1	S_2	S_3	Y
0	0	0	0	A_0
0	0	0	1	A_1
0	0	1	0	A_2
0	0	1	1	A_3
0	1	0	0	A_4
0	1	0	1	A_5
0	1	1	0	A_6
0	1	1	1	A_7
1	0	0	0	A_8
1	0	0	1	A_9
1	0	1	0	A_{10}
1	0	1	1	A_{11}
1	1	0	0	A_{12}
1	1	0	1	A_{13}
1	1	1	0	A_{14}
1	1	1	1	A_{15}

SIMULATED MODEL OUTPUT WAVEFORM



EXP NO.	16:1 MUX USING 4:1MUX	DATE
3(a)		

AIM:

To write a VHDL/Verilog code for 16:1 Mux generate synthesis report, RTL schematic and to implement designs using FPGA (Spartan-2).

APPARATUS:

1. Synthesis Tool: Xilinx Project Navigator - ISE 9.1i.
2. Simulation Tool: Modelsim Simulator.

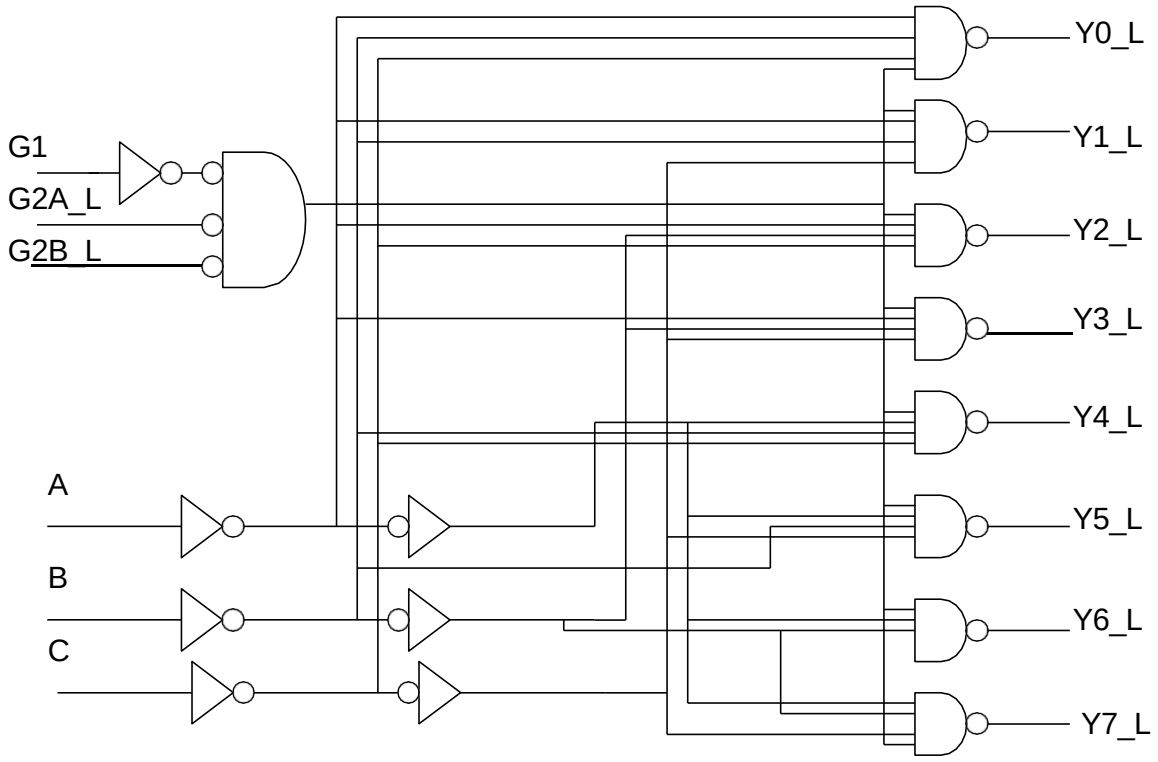
PROCEDURE:

1. Open Xilinx ISE 9.1i.
2. Create a new source file in a new project with suitable name.
3. Create the file in VHDL/Verilog module.
4. Select the appropriate input and output ports according to the requirements.
5. Type the program and save it and synthesize the process.
6. Select Synthesize XST, check for syntax errors and generate report and RTL schematic.
7. Create another new source.
8. Select source type as Test bench wave form.
9. Associate the test bench to the source.
10. Assign clock and timing details.
11. Give the input waveforms for the source.
12. Save the input waveforms and perform behavioral simulation.
13. The simulated output waveforms window will be shown.

DUMPING PROCESS:

1. In the process window, go to 'user constraints' and select 'assign package pins' and after that double click on 'implement design'.
2. Select properties by right clicking on 'generate programming file'. Select 'JTAG' clock in startup options.
3. Select boundary scan in 'impact window' after double clicking on 'configure Device'.
4. In 'generate programming file' double clicking on 'programming file generation report. Bit file will be generated.
5. Xilinx boundary scan window will appear when the bit file is selected. Right click on Xilinx component and select program.
6. Programming properties will appear and finally program will be succeeded.
7. Thus the program can be dumped into FPGA kit and finally output can be seen on the kit.

CIRCUIT DIAGRAM:



VHDL CODE FOR 16:1 using 4:1 mux

```
Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity kanhe_4x1mux is
port(a,b,c,d : in std_logic;
S0,s1 : in std_logic;
q : out std_logic);
end kanhe_4x1mux;

Architecture kanhe_4x1mux1 of kanhe_4x1mux is
Begin
Process(a,b,c,d,s0,s1)
Begin

If s0 ='0' and s1 ='0' then q <= a;
Elsif s0 ='1' and s1 ='0' then q <= b;
elsif s0 ='0' and s1='1' then q <= c;
else q <=d;
end if;
End process;
End kanhe_4x1mux1;
```

Main program:

```
Library ieee;
use ieee.std_logic_1164.all;
use  ieee.std_logic_arith.all;
entity kanhe_16x1mux is
port(a:in std_logic_vector(15 downto 0);
 s: in std_logic_vector(3 downto 0);
Z:out std_logic);
End kanhe_16x1mux;

Architecture kanhe_16x1mux1 of kanhe_16x1mux is
signal z1,z2,z3,z4:std_logic;

component kanhe_4x1mux is
port(a,b,c,d,s0,s1:in std_logic;
Q:out std_logic);

End component;

Begin
M1: kanhe_4x1mux port map(a(0),a(1),a(2),a(3),s(0),s(1),z1);
m2: kanhe_4x1mux port map(a(4),a(5),a(6),a(7),s(0),s(1),z2);
m3: kanhe_4x1mux port map(a(8),a(9),a(10),a(11),s(0),s(1),z3);
m4: kanhe_4x1mux port map(a(12),a(13),a(14),a(15),s(0),s(1),z4);
m5: kanhe_4x1mux port map(z1,z2,z3,z4,s(2),s(3),z);

End kanhe_16x1mux1;
```

EXP NO.	3:8 DECODER	DATE
3(b)		

AIM:

To write a VHDL/Verilog code for 3x8 Decoder and to generate synthesis report, RTL schematic and to implement designs using FPGA (Spartan-2).

APPARATUS:

1. Synthesis Tool: Xilinx Project Navigator - ISE 9.1i.
2. Simulation Tool: Modelsim Simulator.

PROCEDURE:

1. Open Xilinx ISE 9.1i.
2. Create a new source file in a new project with suitable name.
3. Create the file in VHDL/Verilog module.
4. Select the appropriate input and output ports according to the requirements.
5. Type the program and save it and synthesize the process.
6. Select Synthesize XST, check for syntax errors and generate report and RTL schematic.
7. Create another new source.
8. Select source type as Test bench wave form.
9. Associate the test bench to the source.
10. Assign clock and timing details.
11. Give the input waveforms for the source.
12. Save the input waveforms and perform behavioral simulation.
13. The simulated output waveforms window will be shown.

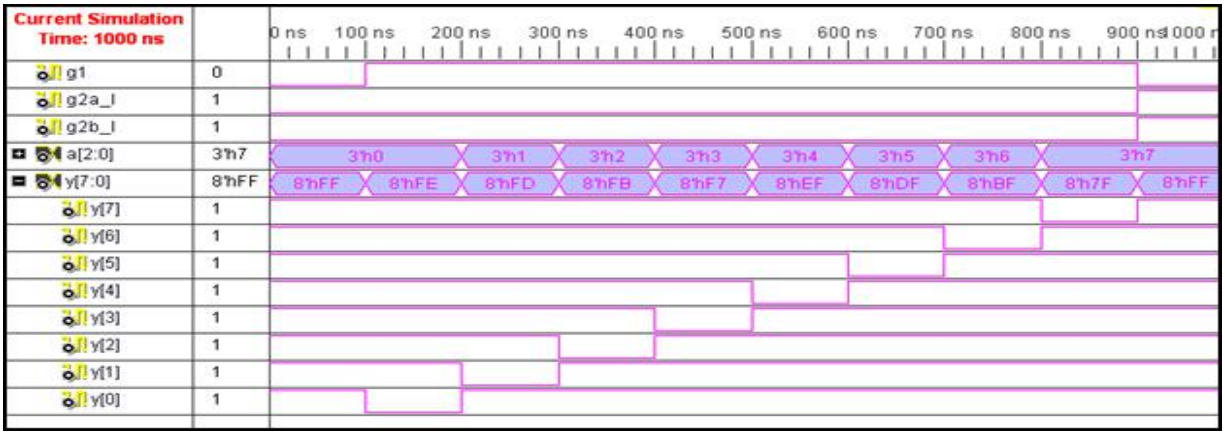
DUMPING PROCESS:

1. In the process window, go to ‘user constraints’ and select ‘assign package pins’ and after that double click on ‘implement design’.
2. Select properties by right clicking on ‘generate programming file’. Select ‘JTAG’ clock in startup options.
3. Select boundary scan in ‘impact window’ after double clicking on ‘configure device’.
4. In ‘generate programming file’ double clicking on ‘programming file generation report. Bit file will be generated.
5. Xilinx boundary scan window will appear when the bit file is selected. Right click on Xilinx component and select program.
6. Programming properties will appear and finally program will be succeeded.
7. Thus the program can be dumped into FPGA kit and finally output can be seen on the kit.

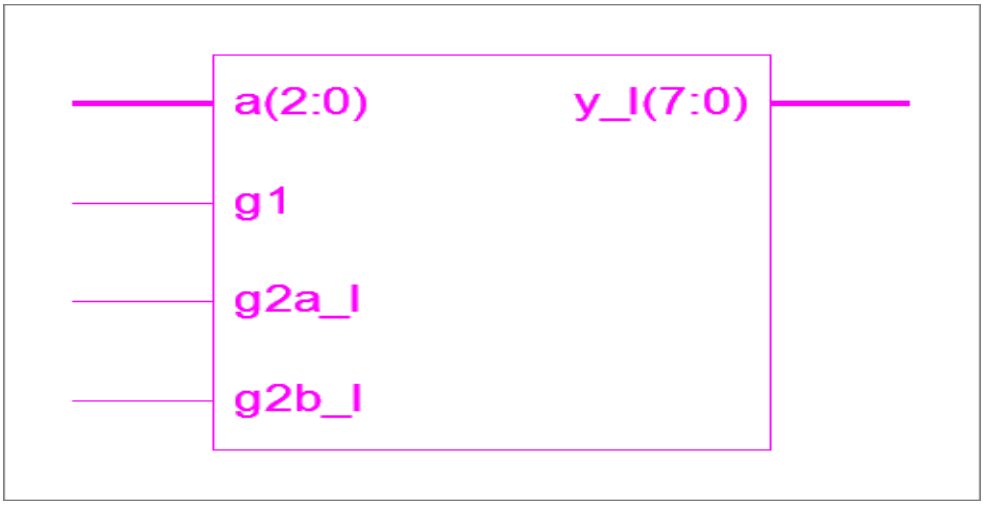
TRUTH TABLE:

Selected enable inputs			inputs			outputs							
G1	G2A_L	G2B_L	A(2)	A(1)	A(0)	Y(7)	Y(6)	Y(5)	Y(4)	Y(3)	Y(2)	Y(1)	Y(0)
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

SIMULATION MODEL OUTPUT WAVEFORM



BLOCK DIAGRAM:

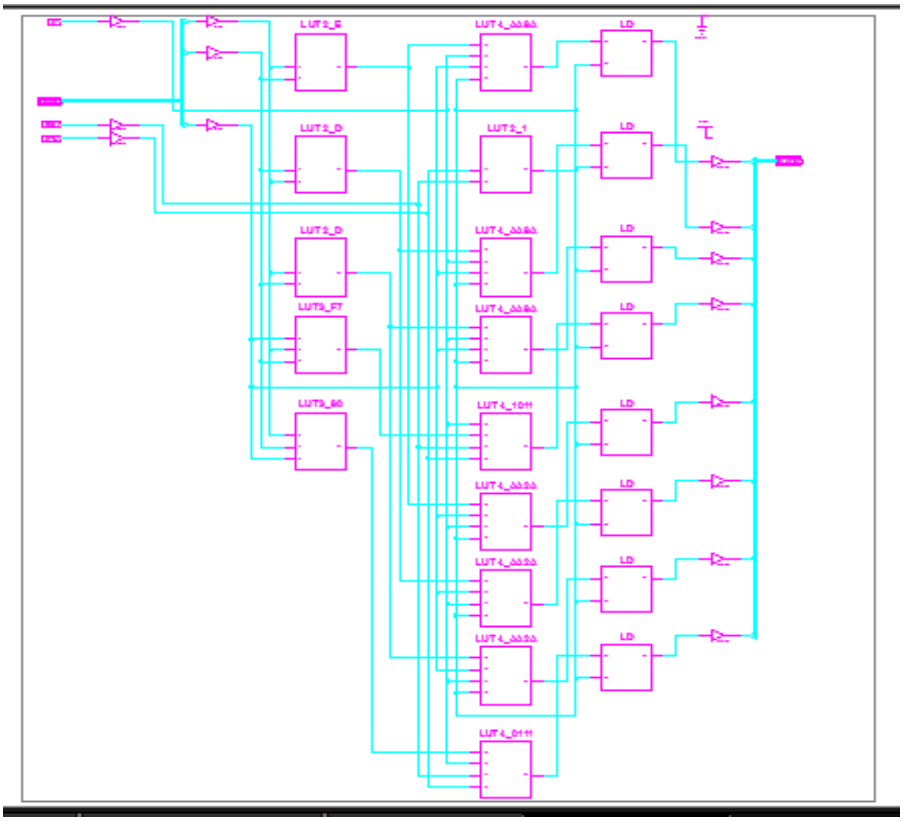


VHDL CODE:

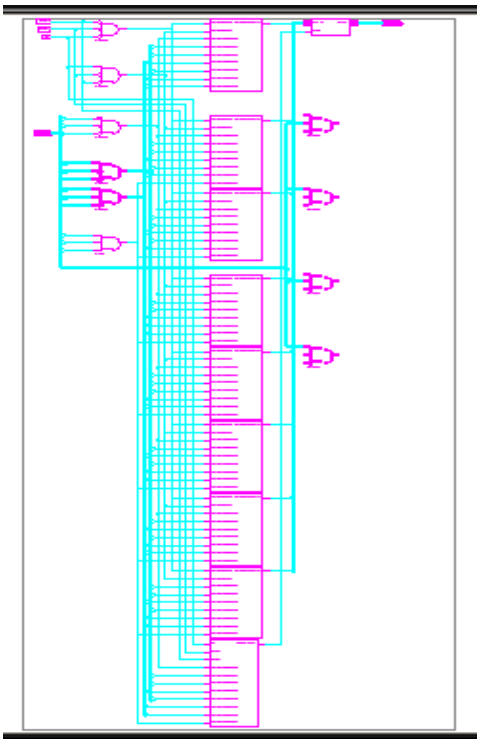
VHDL CODE FOR 3 t0 8 DECODER-BEHAVIORAL MODEL :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity dec3to8_beh is
port ( g1 : in  STD_LOGIC;
        g2a_l : in  STD_LOGIC;
        g2b_l : in  STD_LOGIC;
        a : in  STD_LOGIC_VECTOR(2 downto 0);
        y_l : out  STD_LOGIC_VECTOR(7 downto 0));
end dec3to8_beh;
architecture Behavioral of dec3to8_beh is
begin
process (g1,g2a_l,g2b_l,a)
begin
if (g1='0' and  g2a_l='0' and  g2b_l='0')then
y_l<= "11111111";
elsif(g1 ='1' and g2a_l='0' and g2b_l='0') then
if(a="000")then
y_l<="11111110";
elsif(a="001") then
y_l<="11111101";
elsif(a="010")then
y_l<="11111011";
elsif(a="011")then
y_l<="11110111";
elsif(a="100")then
y_l<="11101111";
elsif(a="101")then
y_l<="11011111";
elsif(a="110")then
y_l<="10111111";
elsif(a="111") then
y_l<="01111111";
end if;
end if;
end process;
end Behavioral;
```

TECHNOLOGY SCHEMATIC:



RTL SCHEMATIC:



DEVICE UTILIZATION SUMMARY:

Number of Slices:	8 out of	960	0%
Number of 4 input LUTs:	14 out of	1920	0%
Number of IOs:	14		
Number of bonded IOBs:	14 out of	66	21%
IOB Flip Flops:	8		

SYNTHESIS REPORT:

RTL Top Level Output File Name : decoder.ngr
Top Level Output File Name : decoder
Output Format : NGC
Optimization Goal : Speed
Keep Hierarchy : NO
Design Statistics
IOs : 14
Cell Usage:
BELS : 14
LUT2 : 4
LUT3 : 2
LUT4 : 8
FlipFlops/Latches : 8
LD : 8
IO Buffers : 14
IBUF : 6
OBUF : 8

RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. Write the behavioral code for the IC 74x138.

2. Write the VHDL code for the IC 74x138 using CASE statement.

3. What does priority encoder mean?

4. How many outputs will a decimal-to-BCD encoder have?

5. Can an encoder be a transducer?

EXP NO.	8:3 ENCODER	DATE
4		

AIM:

To write a VHDL/Verilog code for 8:3 Encoder and to generate synthesis report, RTL schematic and to implement designs using FPGA (Spartan-3).

APPARATUS:

1. Synthesis Tool: Xilinx Project Navigator - ISE 9.1i.
2. Simulation Tool: Modelsim Simulator.

PROCEDURE:

1. Open Xilinx ISE 9.1i.
2. Create a new source file in a new project with suitable name.
3. Create the file in VHDL/Verilog module.
4. Select the appropriate input and output ports according to the requirements.
5. Type the program and save it and synthesize the process.
6. Select Synthesize XST, check for syntax errors and generate report and RTL schematic.
7. Create another new source.
8. Select source type as Test bench wave form.
9. Associate the test bench to the source.
10. Assign clock and timing details.
11. Give the input waveforms for the source.
12. Save the input waveforms and perform behavioral simulation.
13. The simulated output waveforms window will be shown.

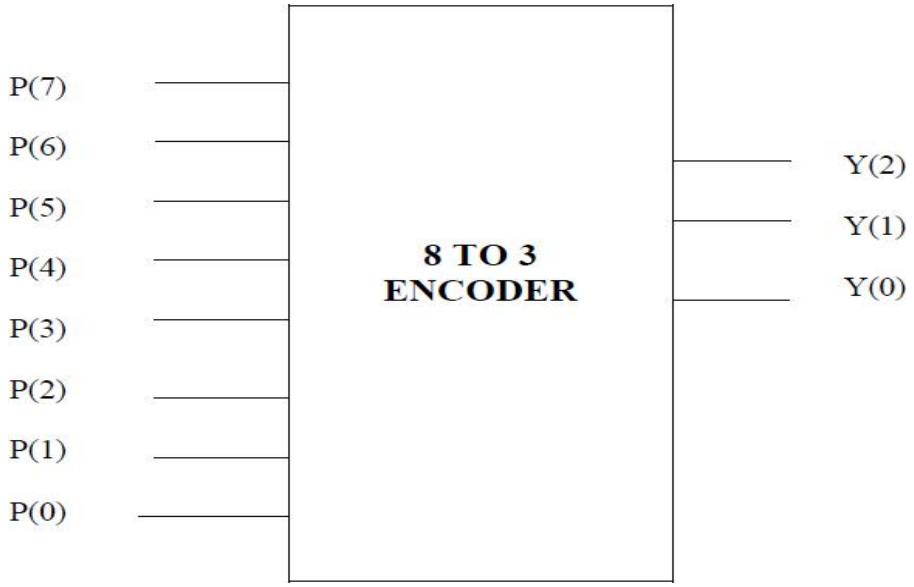
DUMPING PROCESS:

1. In the process window, go to 'user constraints' and select 'assign package pins' and after that double click on 'implement design'.
2. Select properties by right clicking on 'generate programming file'. Select 'JTAG' clock in startup options.
3. Select boundary scan in 'impact window' after double clicking on 'configure device'.
4. In 'generate programming file' double clicking on 'programming file generation report. Bit file will be generated.
5. Xilinx boundary scan window will appear when the bit file is selected. Right click on Xilinx component and select program.
6. Programming properties will appear and finally program will be succeeded.
7. Thus the program can be dumped into FPGA kit and finally output can be seen on the kit.

VHDL CODE FOR 8 to 3 ENCODER:

```
library ieee;

use ieee.std_logic_1164.all;

entity e83 is
port(p: in std_logic_vector(7 downto 0);
y: out std_logic_vector(2 downto 0));
end e83;
architecture beh of e83 is
begin
process(p)
begin
case p is


when "00000001"=>y<="000";
when "00000001"=>y<="001";
when "00000100"=>y<="010";
when "00001000"=>y<="011";
when "00010000"=>y<="100";
when "00100000"=>y<="101";
when "01000000"=>y<="110";
when "10000000"=>y<="111";
when others=>y<="UUU";

end case;
end process;
```

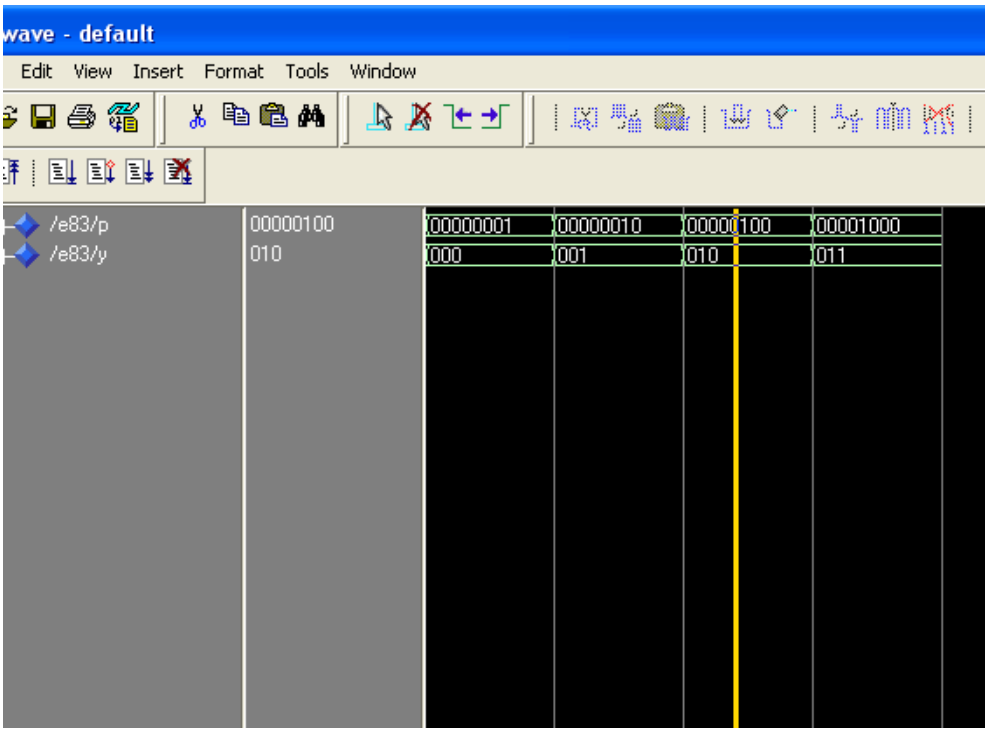
RTL SCHEMATIC:



TRUTH TABLE:

INPUTS								OUTPUTS		
P7	P6	P5	P4	P3	P2	P1	P0	Y(2)	Y(1)	Y(0)
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

SIMULATION MODEL OUTPUT WAVEFORM



RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. what is the use of 8:3 encoder?

2. What are the types of encoder?

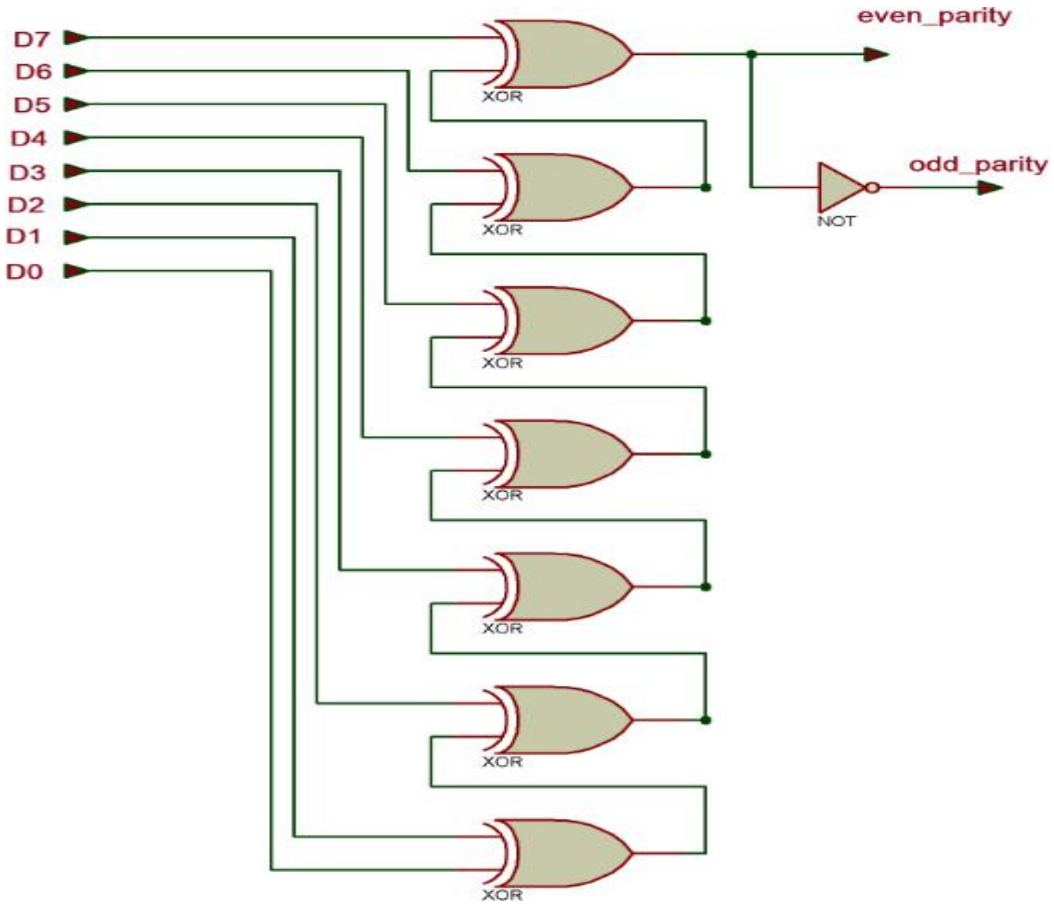
3. How is an encoder different from a decoder?

4. How many gates are required for a 8 to 3 encoder?

5. Where is encoder used?

CIRCUIT DIAGRAM:

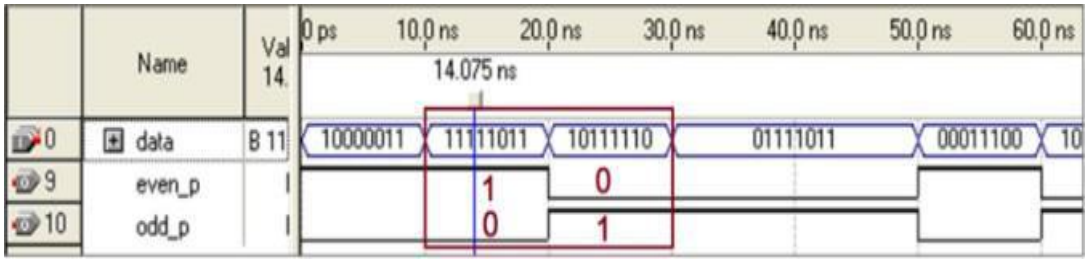
The 8-bit parity generator



TRUTH TABLE:

D7	D6	D5	D4	D3	D2	D1	D0	Even_parity	Odd_parity
1	0	1	1	0	0	1	0	0	1
1	1	0	0	1	0	0	0	1	0
1	1	1	1	1	0	1	1	1	0
1	0	1	1	1	1	1	0	0	1
0	0	1	0	1	0	1	0	1	0
0	1	1	1	0	1	0	1	1	0
0	1	0	1	0	0	1	1	0	1

SIMULATION MODEL OUTPUT WAVEFORM :



EXP NO.	PARITY GENERATOR AND CHECKER	DATE
5		

AIM:

To write a VHDL/Verilog code for Parity generator and checker synthesis report, RTL schematic and to implement designs using FPGA (Spartan-3).

APPARATUS:

1. Synthesis Tool: Xilinx Project Navigator - ISE 9.1i.
2. Simulation Tool: Modelsim Simulator.

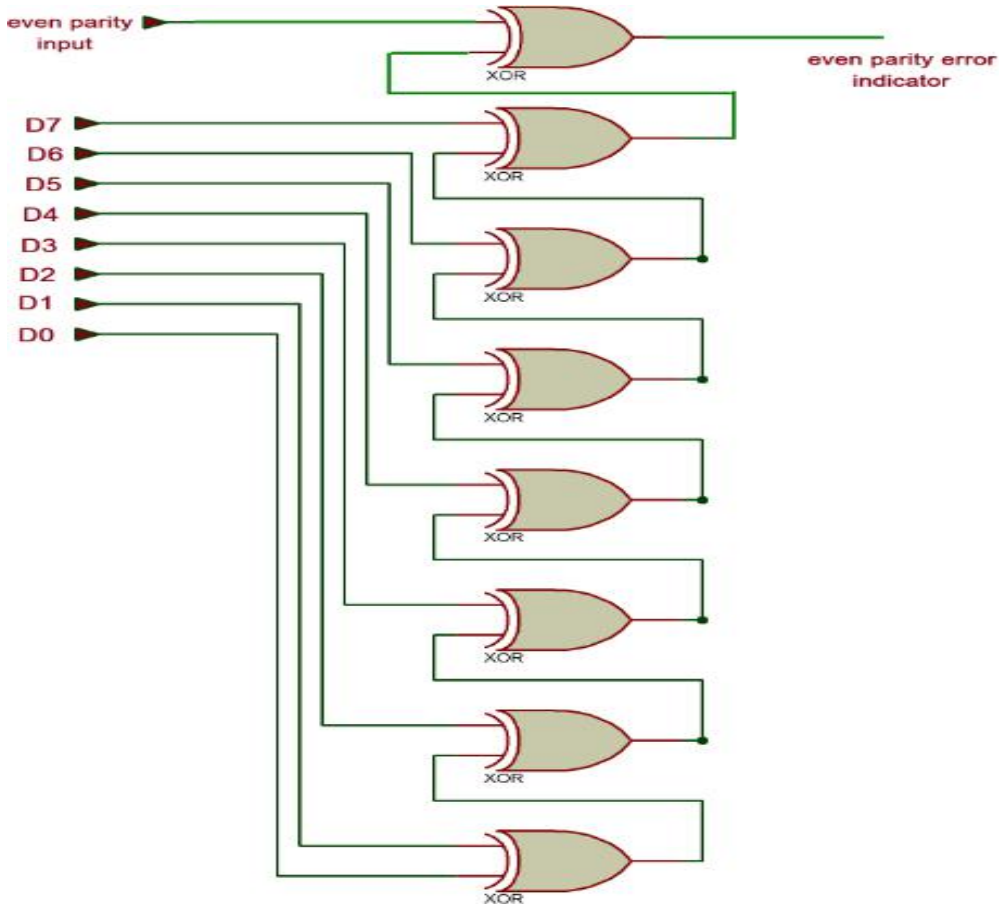
PROCEDURE:

1. Open Xilinx ISE 9.1i.
2. Create a new source file in a new project with suitable name.
3. Create the file in VHDL/Verilog module.
4. Select the appropriate input and output ports according to the requirements.
5. Type the program and save it and synthesize the process.
6. Select Synthesize XST, check for syntax errors and generate report and RTL schematic.
7. Create another new source.
8. Select source type as Test bench wave form.
9. Associate the test bench to the source.
10. Assign clock and timing details.
11. Give the input waveforms for the source.
12. Save the input waveforms and perform behavioral simulation.
13. The simulated output waveforms window will be shown.

DUMPING PROCESS:

1. In the process window, go to 'user constraints' and select 'assign package pins' and after that double click on 'implement design'.
2. Select properties by right clicking on 'generate programming file'. Select 'JTAG' clock in startup options.
3. Select boundary scan in 'impact window' after double clicking on 'configure device'.
4. In 'generate programming file' double clicking on 'programming file generation report. Bit file will be generated.
5. Xilinx boundary scan window will appear when the bit file is selected. Right click on Xilinx component and select program.
6. Programming properties will appear and finally program will be succeeded.
7. Thus the program can be dumped into FPGA kit and finally output can be seen on the kit.

CIRCUIT DIAGRAM:



TRUTH TABLE:

D7	D6	D5	D4	D3	D2	D1	D0	Parity input	Error
1	0	1	1	0	0	1	0	1	1
1	1	0	0	1	0	0	0	1	0
1	0	1	1	1	0	1	1	1	1
1	0	1	1	1	1	1	0	0	0
0	0	1	0	1	0	1	0	1	0
0	1	1	1	0	1	0	1	0	1
0	1	0	1	0	0	1	1	1	1

SIMULATION MODEL OUTPUT WAVEFORM :

VHDL CODE:

The 8-bit parity generator:

```
 library ieee;
use ieee.std_logic_1164.all;
entity parity is
      port( data:in bit_vector(7 downto 0);
       even_p,odd_p: out bit);
end parity;
architecture parity_gen of parity is
signal temp : bit_vector(5 downto 0);
begin
temp(0)<=data(0) xor data(1);
temp(1)<=temp(0) xor data(2);
temp(2)<=temp(1) xor data(3);
temp(3)<=temp(2) xor data(4);
temp(4)<=temp(3) xor data(5);
temp(5)<=temp(4) xor data(6);
even_p <= temp(5) xor data(7);
odd_p <= not(temp(5) xor data(7);
end parity_gen;
```

The 8-bit parity checker :

```
library ieee;
use ieee.std_logic_1164.all;
entity parity is
port( data:in bit_vector(7 downto 0);
even_p,odd_p: out bit);
end parity;
architecture parity_gen of parity is
signal temp : bit_vector(5 downto 0);
begin
temp(0)<=data(0) xor data(1);
temp(1)<=temp(0) xor data(2);
temp(2)<=temp(1) xor data(3);
temp(3)<=temp(2) xor data(4);
temp(4)<=temp(3) xor data(5);
temp(5)<=temp(4) xor data(6);
even_p <= temp(5) xor data(7);
odd_p <= not(temp(5) xor data(7);
end parity_arch;
```

RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. What is parity generator and checker?

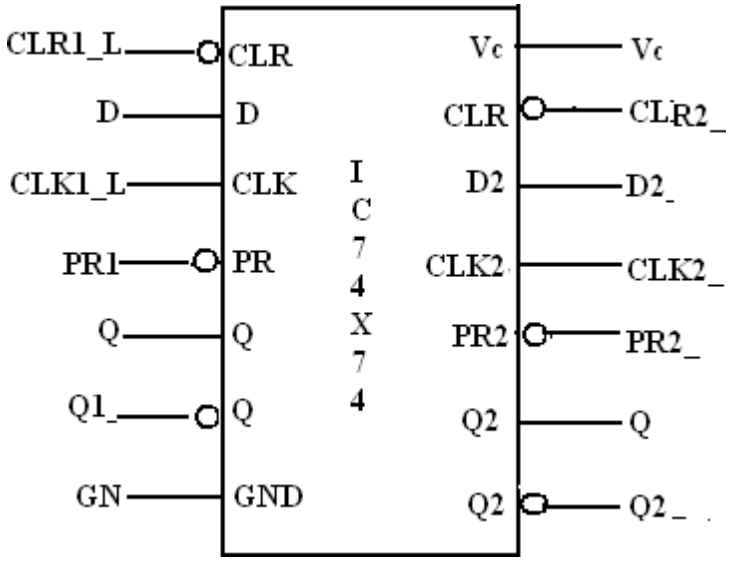
2. Which gate is ideal for checking the parity of data?

3. What is the purpose of parity checker?

4. What is a 3-bit parity checker?

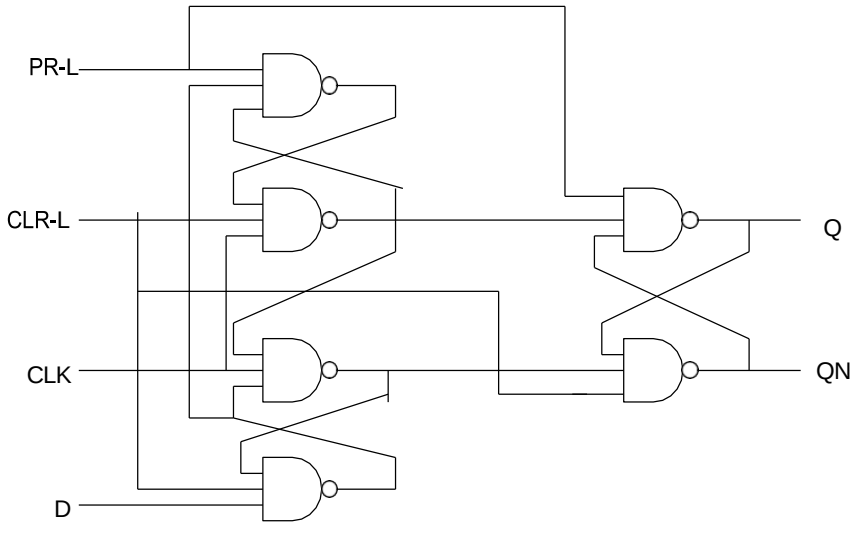
5. How is parity calaculated?

FLIP FLOPS:



LOGIC DIAGRAM OF IC 74X74

LOGIC DIARAM OF D FLIP FLOP



EXP NO.	FLIP FLOPS	DATE
6		

AIM:

To write a VHDL/Verilog code for Flip Flop and to generate synthesis report, RTL schematic and to implement designs using FPGA (Spartan-2).

APPARATUS:

1. Synthesis Tool: Xilinx Project Navigator - ISE 9.1i.
2. Simulation Tool: Modelsim Simulator.

PROCEDURE:

1. Open Xilinx ISE 9.1i.
2. Create a new source file in a new project with suitable name.
3. Create the file in VHDL/Verilog module.
4. Select the appropriate input and output ports according to the requirements.
5. Type the program and save it and synthesize the process.
6. Select Synthesize XST, check for syntax errors and generate report and RTL schematic.
7. Create another new source.
8. Select source type as Test bench wave form.
9. Associate the test bench to the source.
10. Assign clock and timing details.
11. Give the input waveforms for the source.
12. Save the input waveforms and perform behavioral simulation.
13. The simulated output waveforms window will be shown.

DUMPING PROCESS:

1. In the process window, go to 'user constraints' and select 'assign package pins' and after that double click on 'implement design'.
2. Select properties by right clicking on 'generate programming file'. Select 'JTAG' clock in startup options.
3. Select boundary scan in 'impact window' after double clicking on 'configure Device'.
4. In 'generate programming file' double clicking on 'programming file generation report. Bit file will be generated.
5. Xilinx boundary scan window will appear when the bit file is selected. Right click on Xilinx component and select program.
6. Programming properties will appear and finally program will be succeeded.
7. Thus the program can be dumped into FPGA kit and finally output can be seen on the kit.

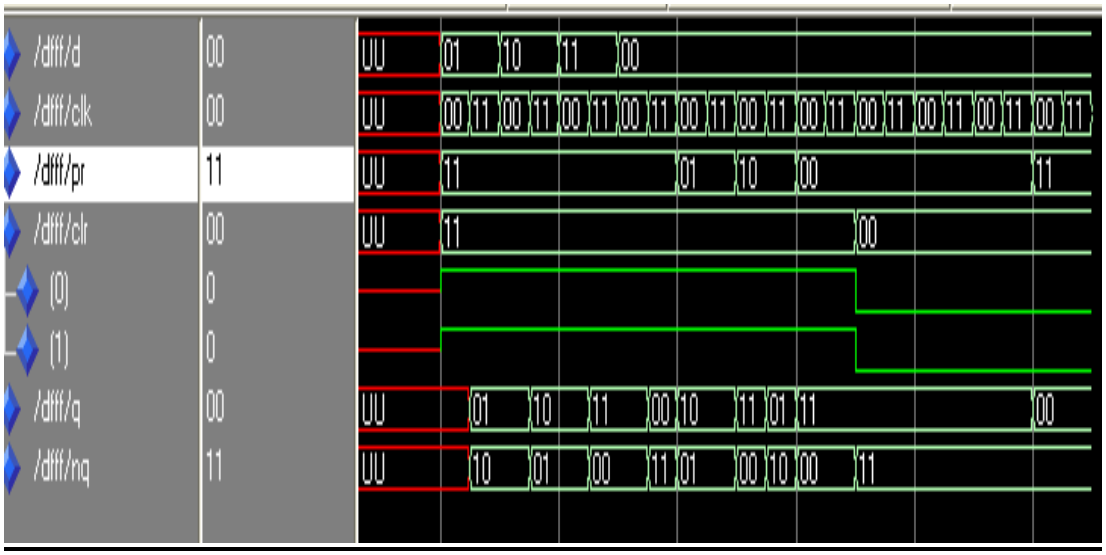
TRUTH TABLE:

TRUTH TABLE :

CLK	PR_L	CLR_L	D	Q	QN
x	0	1	x	1	0
x	1	0	x	0	1
x	0	0	x	1	1
↓	1	1	0	0	1
↑	1	1	1	1	0
↓	1	1	x	Q	QN

BLOCK DIAGRAM:

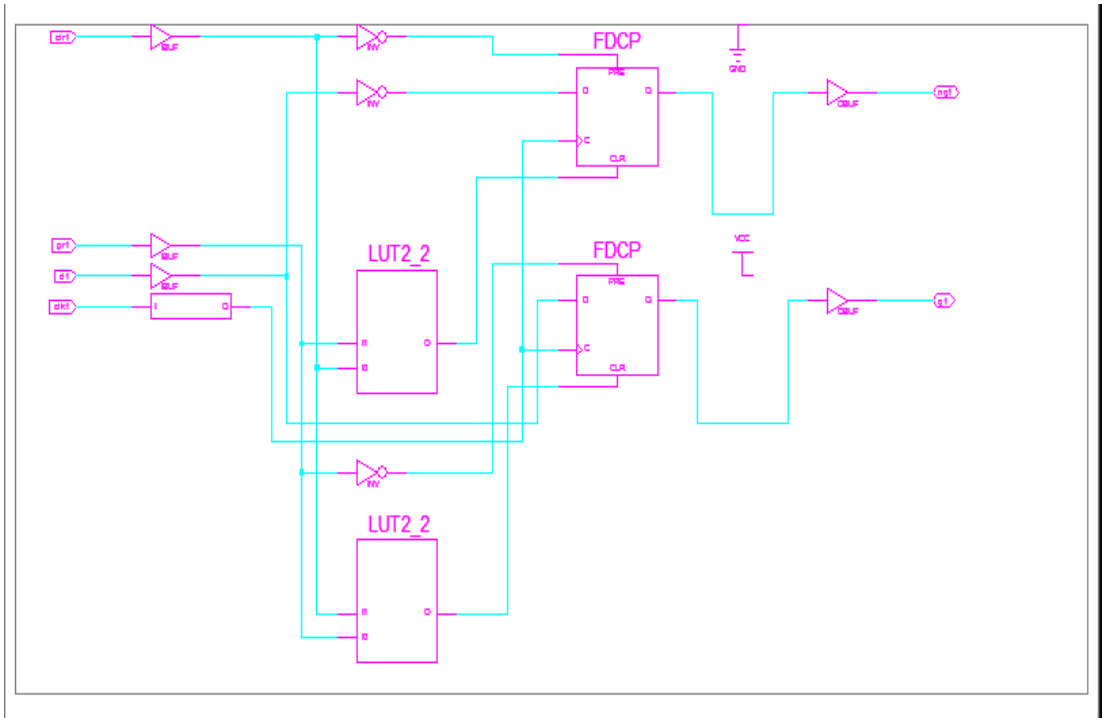
SIMULATED MODEL OUTPUT WAVEFORM:



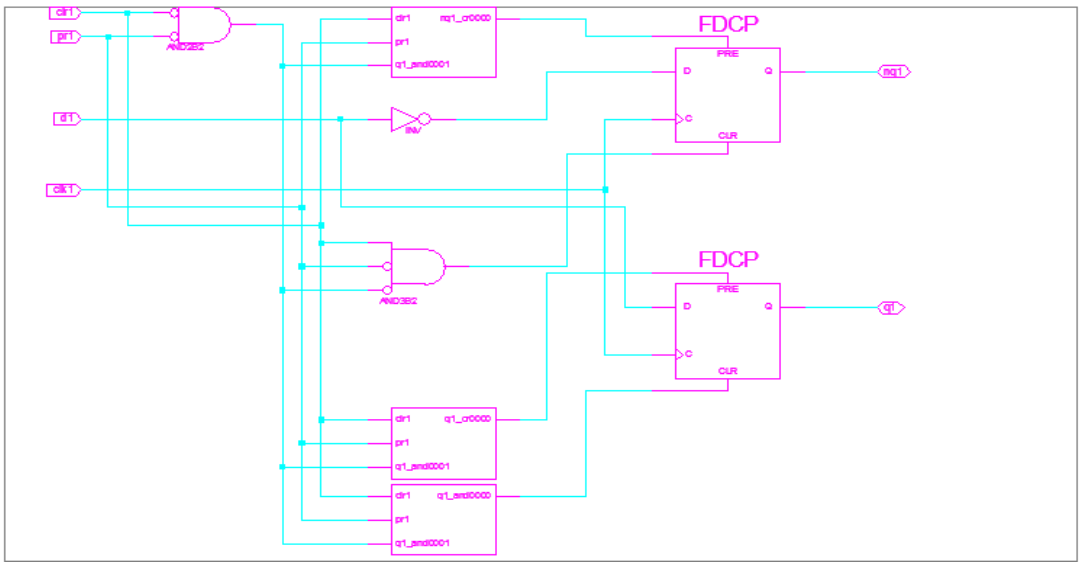
VHDL CODE:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity dfff is
Port ( d: in  STD_LOGIC_VECTOR(0 to 1);
       clk : in  STD_LOGIC_VECTOR;
       pr : in  STD_LOGIC_VECTOR(0 to 1);
       clr : in  STD_LOGIC_VECTOR(0 to 1);
       q : inout  STD_LOGIC_VECTOR(0 to 1);
       nq :inout  STD_LOGIC_VECTOR(0 to 1));
end DFFF;
architecture structural of DFFF is
component dff1
port(d1,clk1,pr1,clr1:in std_logic;
q1,nq1:inout std_logic);
end component;
begin
    D1:dff1 port  map(d(0),clk, pr(0),clr(0),q(0),nq(0));
    D2:dff1 port  map(d(1),clk, pr(1),clr(1),q(1),nq(1));
end structural;
```

TECHNOLOGY SCHEMATIC:



RTLSCHEMATIC:



VHDL CODE FOR COMPONENT D FLIPFLOP:

```
entity dff1 is
    Port ( d1: in  STD_LOGIC;
           Clk1 : in  STD_LOGIC;
           Pr1 : in  STD_LOGIC;
           Clr1 : in  STD_LOGIC;
           Q1 : inout  STD_LOGIC;
           Nq1 :inout  STD_LOGIC);
end dff1;
architecture Behavioral of dff1 is
begin
process(d1,pr1,clr1,clk1)
begin
     if(pr1='0' and clr1='0')then
      q1<='1';nq1<='1';
 elsif(pr1='0' and clr1='1')then
  q1<='1';nq1<='0';
  elsif(pr1='1' and clr1='0')then
  q1<='0';nq1<='1';
  else
           if(clk1='1' and clk1'event)then
           q1<=d1;nq1<=not d1;
           else
           q1<=q1;nq1<=nq1;
           end if;
  end if;
  end process;
end Behavioral;
```

DEVICE UTILIZATION SUMMARY:

Number of Slices	: 3 out of	960	0%
Number of Slice Flip Flops	: 2 out of	1920	0%
Number of 4 input LUTs	: 5 out of	1920	0%
Number of IOs	: 6		
Number of bonded IOBs	: 6 out of	66	9%
IOB Flip Flops	: 2		
Number of GCLKs	: 1 out of	24	4%

SYNTHESIS REPORT:

RTL Top Level Output File Name	: dff.ngr
Top Level Output File Name	: dff
Output Format	: NGC
Optimization Goal	: Speed
Keep Hierarchy	: NO
Design Statistics	
# IOs	: 6
Cell Usage :	
# BELS	: 5
# INV	: 3
# LUT2	: 2
# FlipFlops/Latches	: 2
# FDCP	: 2
# Clock Buffers	: 1
# BUFGP	: 1
# IO Buffers	: 5
# IBUF	: 3
# OBUF	: 2

RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. What are the applications of Flip flops?

2. The truth table for an S-R flip-flop has how many VALID entries?

3. When both inputs of a J-K flip-flop cycle, the output will?

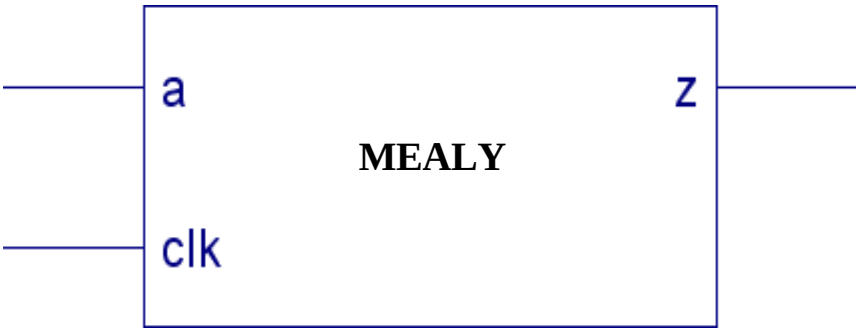
4. How many types of sequential circuits are?

5. In D flip-flop, if clock input is LOW, the D input?

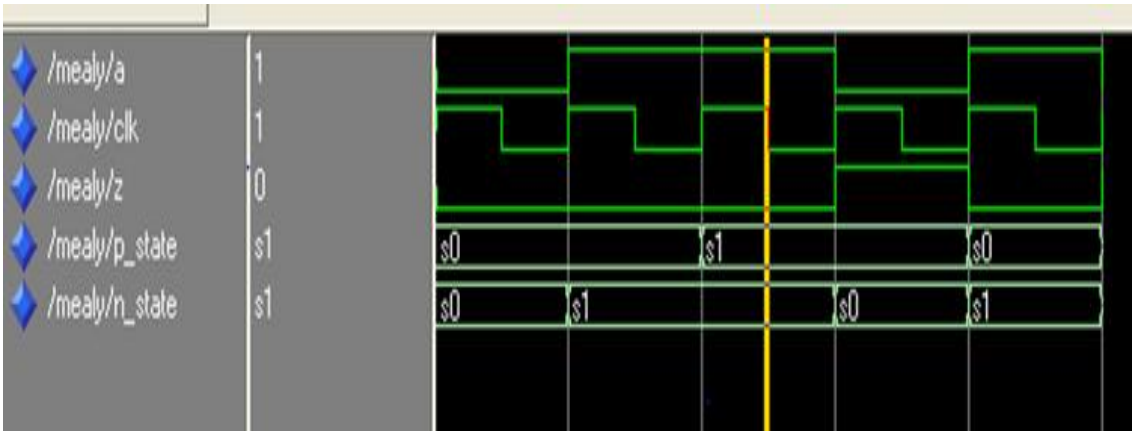
TRUTH TABLE:

PRESENT STATE	NEXT STATE		OUTPUT(Z)	
	a=0	a=1	a=0	a=1
S0	S0	S1	0	0
S1	S0	S1		0

RTL SCHEMATIC:



SIMULATED MODEL OUTPUT WAVEFORM:



EXP NO.	4 BIT SEQUENCE DETECTOR THROUGH MEALY &MOORE STATE MACHINES	DATE
7		

AIM:

To write a VHDL/Verilog code and to generate synthesis report, RTL schematic and to implement designs using FPGA (Spartan-2).

APPARATUS:

1. Synthesis Tool: Xilinx Project Navigator - ISE 9.1i.
2. Simulation Tool: Modelsim Simulator.

PROCEDURE:

1. Open Xilinx ISE 9.1i.
2. Create a new source file in a new project with suitable name.
3. Create the file in VHDL/Verilog module.
4. Select the appropriate input and output ports according to the requirements.
5. Type the program and save it and synthesize the process.
6. Select Synthesize XST, check for syntax errors and generate report and RTL schematic.
7. Create another new source.
8. Select source type as Test bench wave form.
9. Associate the test bench to the source.
10. Assign clock and timing details.
11. Give the input waveforms for the source.
12. Save the input waveforms and perform behavioral simulation.
13. The simulated output waveforms window will be shown.

DUMPING PROCESS:

1. In the process window, go to 'user constraints' and select 'assign package pins' and after that double click on 'implement design'.
2. Select properties by right clicking on 'generate programming file'. Select 'JTAG' clock in startup options.
3. Select boundary scan in 'impact window' after double clicking on 'configure Device'.
4. In 'generate programming file' double clicking on 'programming file generation report. Bit file will be generated.
5. Xilinx boundary scan window will appear when the bit file is selected. Right click on Xilinx component and select program.
6. Programming properties will appear and finally program will be succeeded.
7. Thus the program can be dumped into FPGA kit and finally output can be seen on the kit.

VHDL CODE FOR MEALY MACHINE:

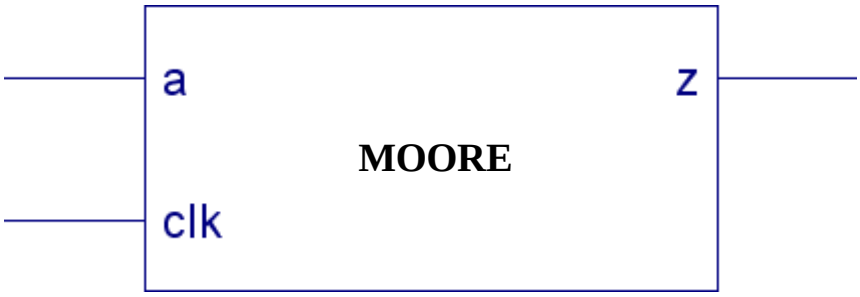
```
library ieee;
use ieee.std_logic_1164.all;
entity
mealyis
port(a, clk: in std_logic;z: out std_logic);
end mealy;
architecture beh of mealy is
typestate is( s0, s1);
signal p_state,n_state: state;
begin
sm: process(clk)
 begin
if rising_edge(clk) then

p_state<=n_state;
 end if;
end process sm;
cm: process(p_state,a)
begin
case p_state is
when s0=> if(a='0') then z<='0';
n_state<=p_state;
elsez<='0'; n_state<=s1;
end if;
whens1 =>if(a='1')thenz<='0';
n_state<=p_state;
elsez<='1';
n_state<=s0;
```

```
end if;
whenothers=>z<='0';
n_state<=s0;
end case;
endprocess cm;
end beh;
```

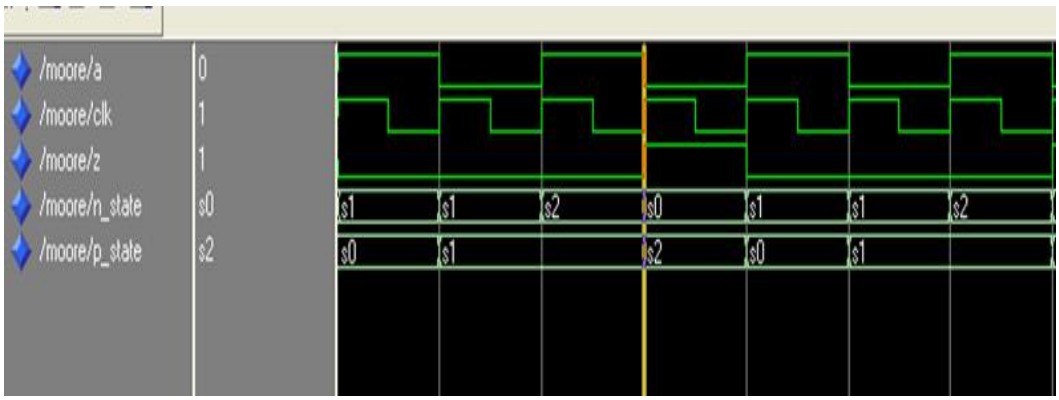
TRUTH TABLE:

PRESENT STATE	NEXT STATE		OUTPUT(Z)	
	a=0	a=1	a=0	a=1
S0	S0	S1	0	0
S1	S2	S1	0	0
S2	S0	S1	1	0

RTL SCHEMATIC:



SIMULATED MODEL OUTPUT WAVEFORM:



VHDL CODE FOR MOORE MACHINE:

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

entity moore is
port(a, clk:instd_logic; z:outstd_logic);
end moore;

 architecture beh of moore is
type state is(s0,s1,s2);
signal n_state,p_state: state;
begin

 s:process(clk)
begin
if rising_edge(clk) then
 p_state<=n_state;
 end if;end process;
 d: process(a,p_state)
 begin
case p_state is
when s0=>z<='0';
if(a='0') then
n_state<= p_state;
 else
n_state<=s1;
 end if;
when s1=>z<='0';
if(a='1') then
 n_state<=s2;
else
n_state<=p_state;
end if;
when s2=>z<='1';
 n_state<=s0;
 endcase;
 endprocess;
 end beh;
```

RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. What is the use of sequence detector?

2. What is sequence detector moore?

3. What is the differenc between mealy and moore?

4. What is VHDL?

5. What is overlapping sequence detector?

PART B

(Back-end Level Design and Implementation)

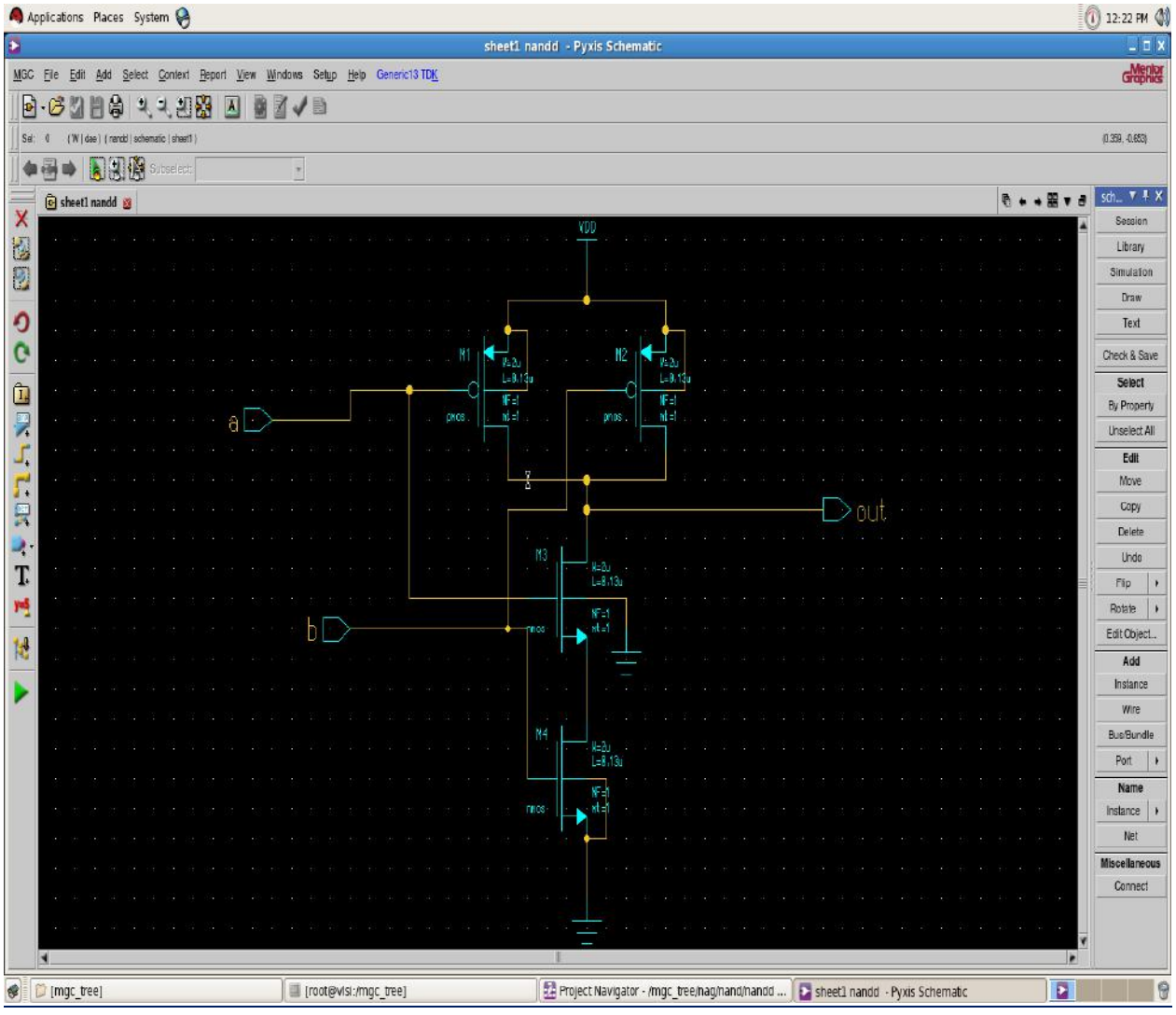
EXP NO.	Design and Implementation Universal Gates	DATE
8		

Aim: To design and implementation of universal gates

Software Required: Mentor Graphics-Pyxis, AMS, Calibre

(i). NAND Gate:

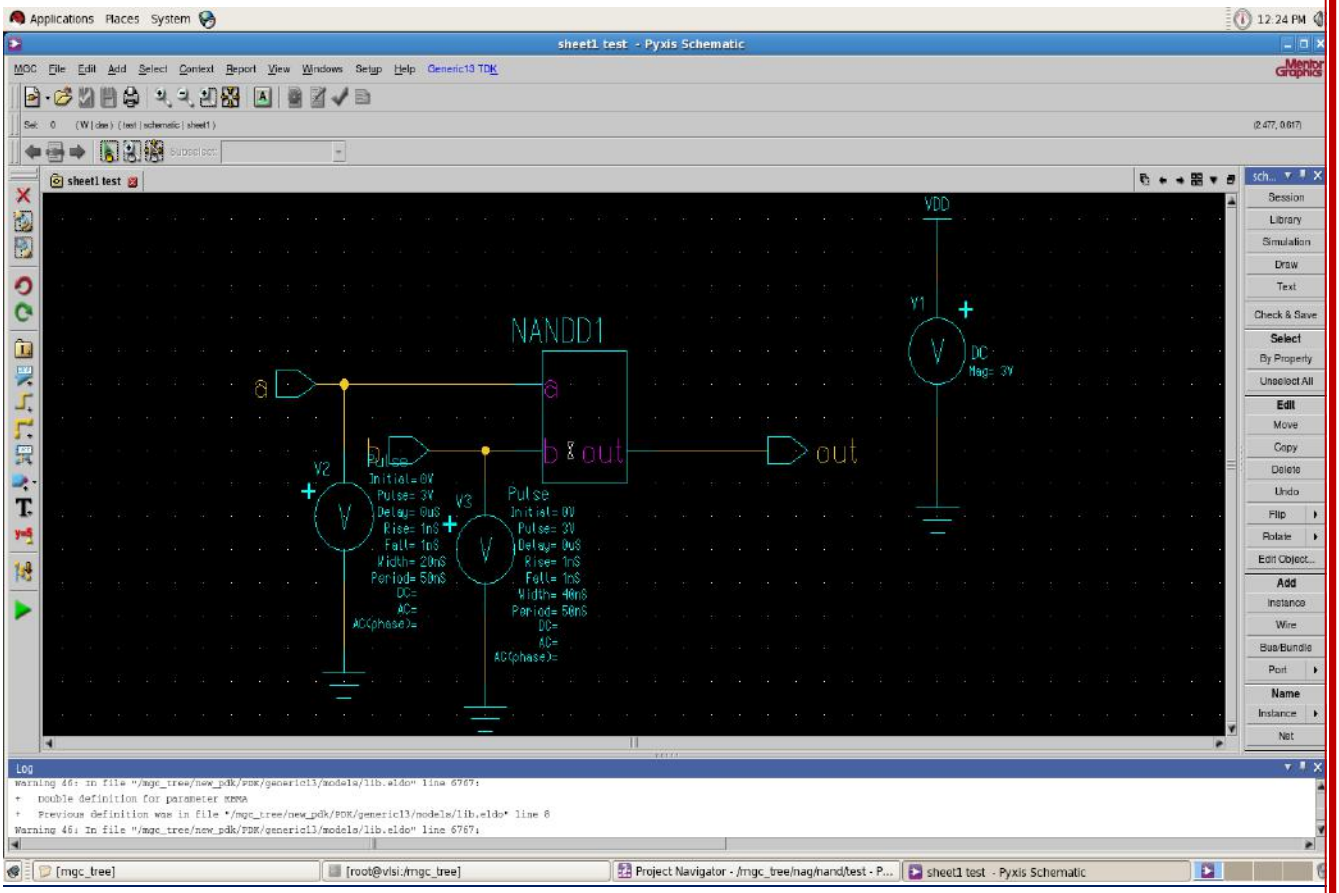
Circuit Diagram:



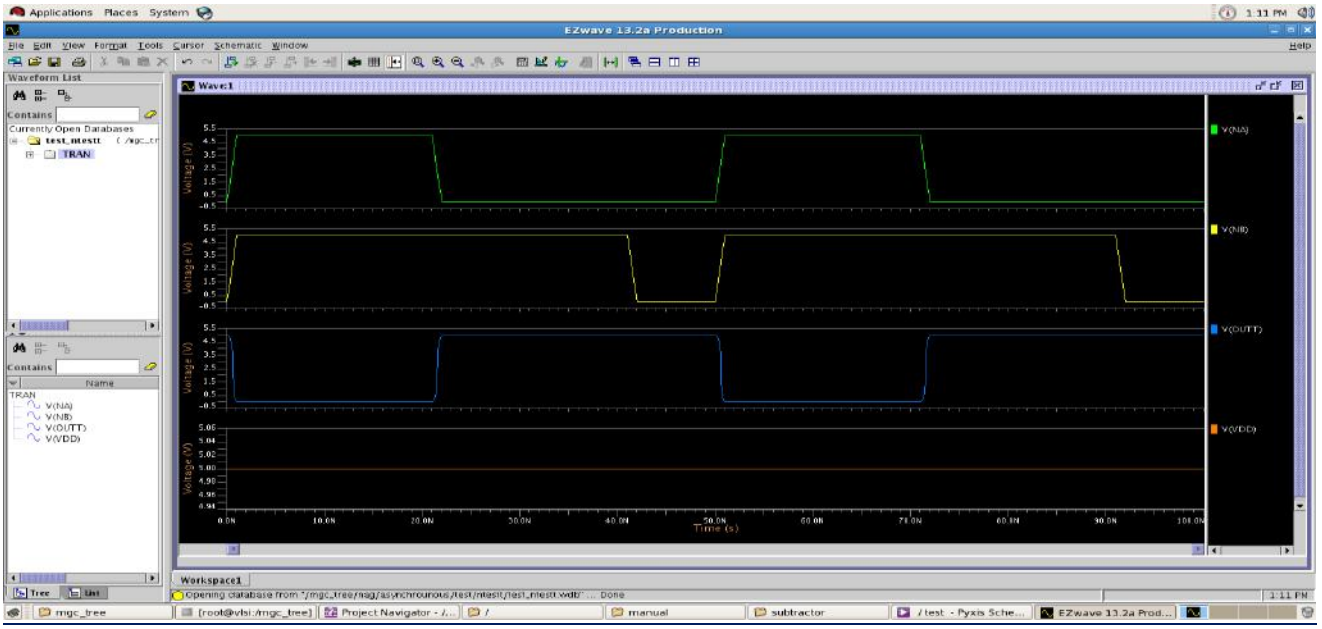
PROCEDURE:

1. Connect the Circuit as shown in the circuit diagram using Pyxis Schematic tool
2. Enter into Simulation mode.
3. Setup the Analysis and library.
4. Setup the required analysis.
5. Probe the required Voltages
6. Run the simulation.
7. Observe the waveforms in EZ wave.

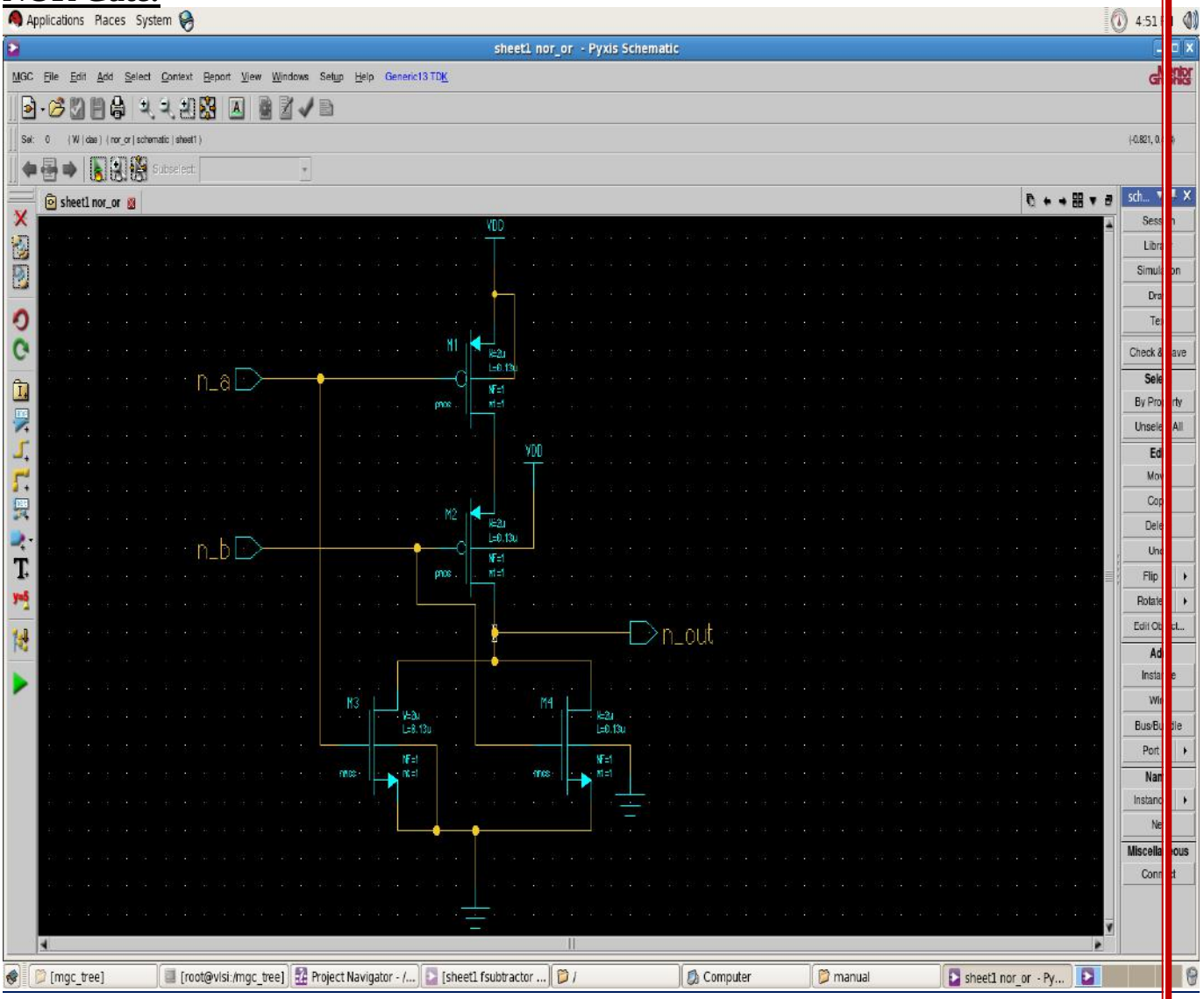
Testbench:



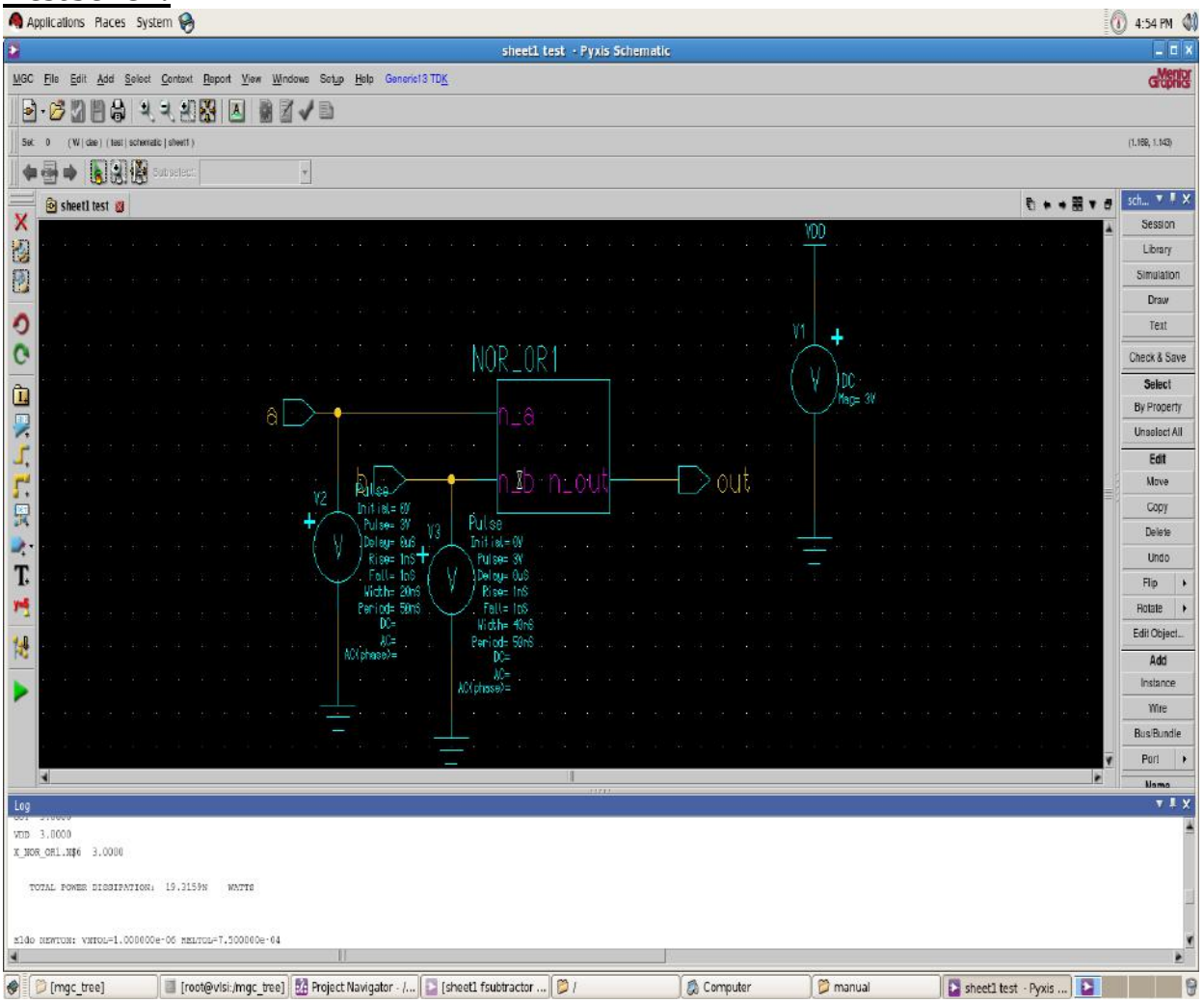
Output waveforms:



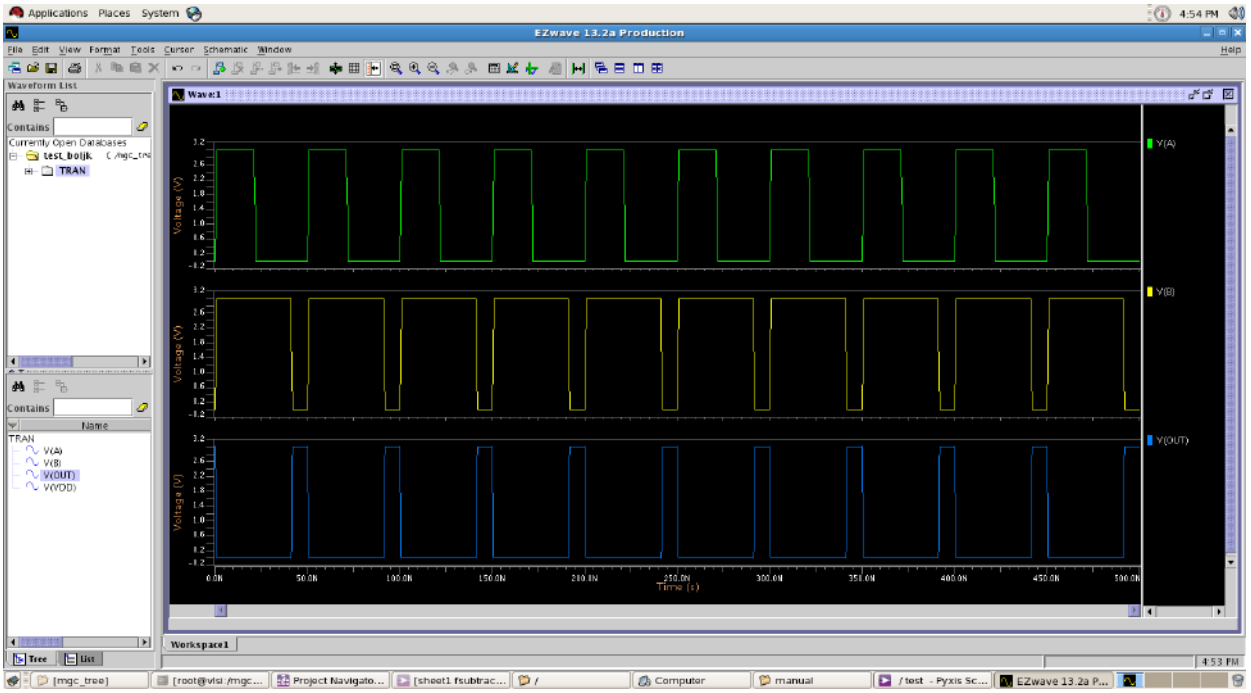
NOR Gate:



Testbench:



Output waveforms:



Result:

Conclusion:

Viva Questions:

1. How are universal gates implemented?

2. What is the difference between basic gates and universal gates?

3. What is the logic equation for a gate?

4. How many transistor are in Ex-or gate?

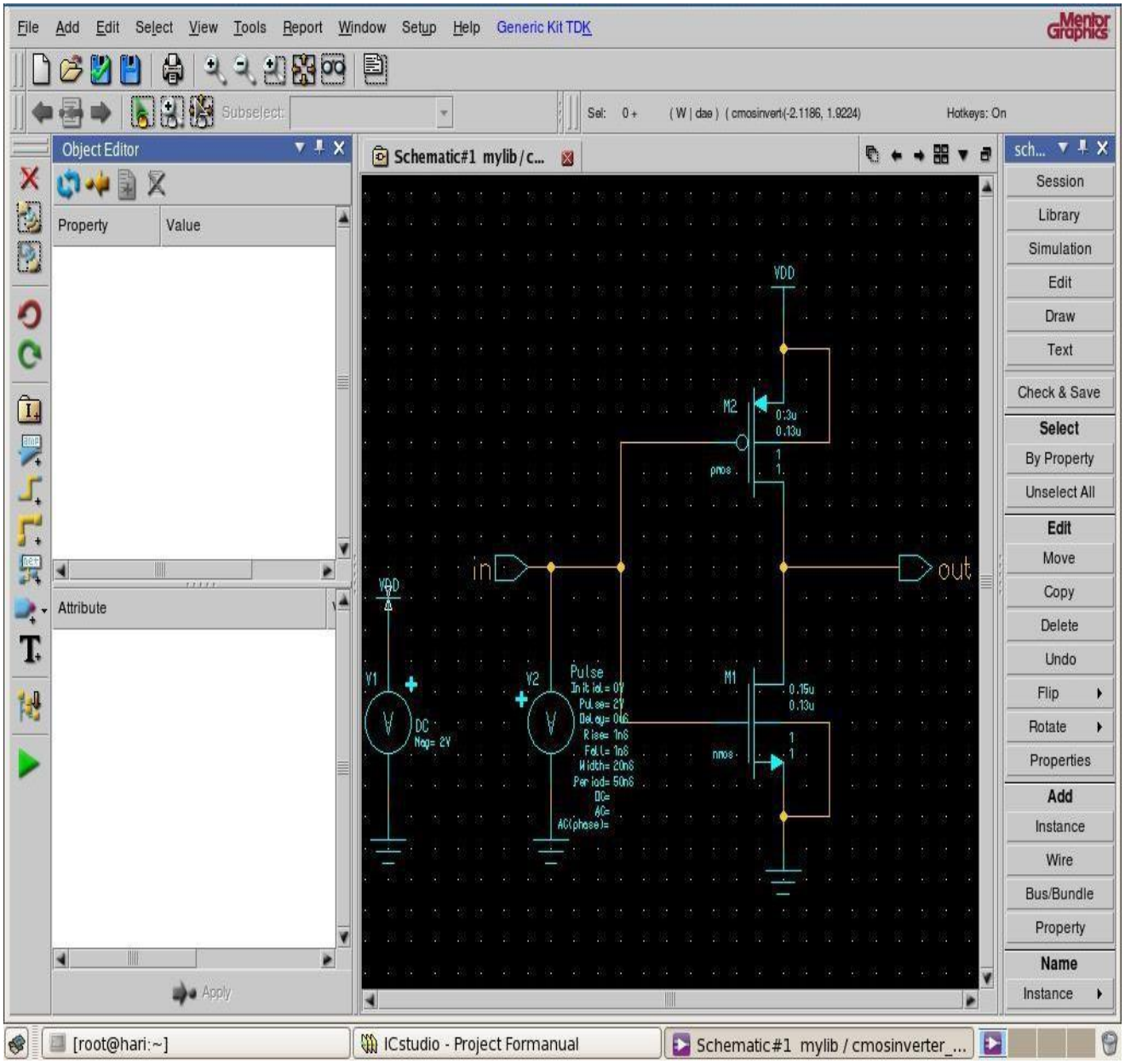
5. Which gate is known as equality detector?

EXP NO.	Design and Implementation of an Inverter	DATE
9		

AIM: To design and Implementation of an Inverter

Software Required: Mentor Graphics - Pyxis, AMS, Calibre.

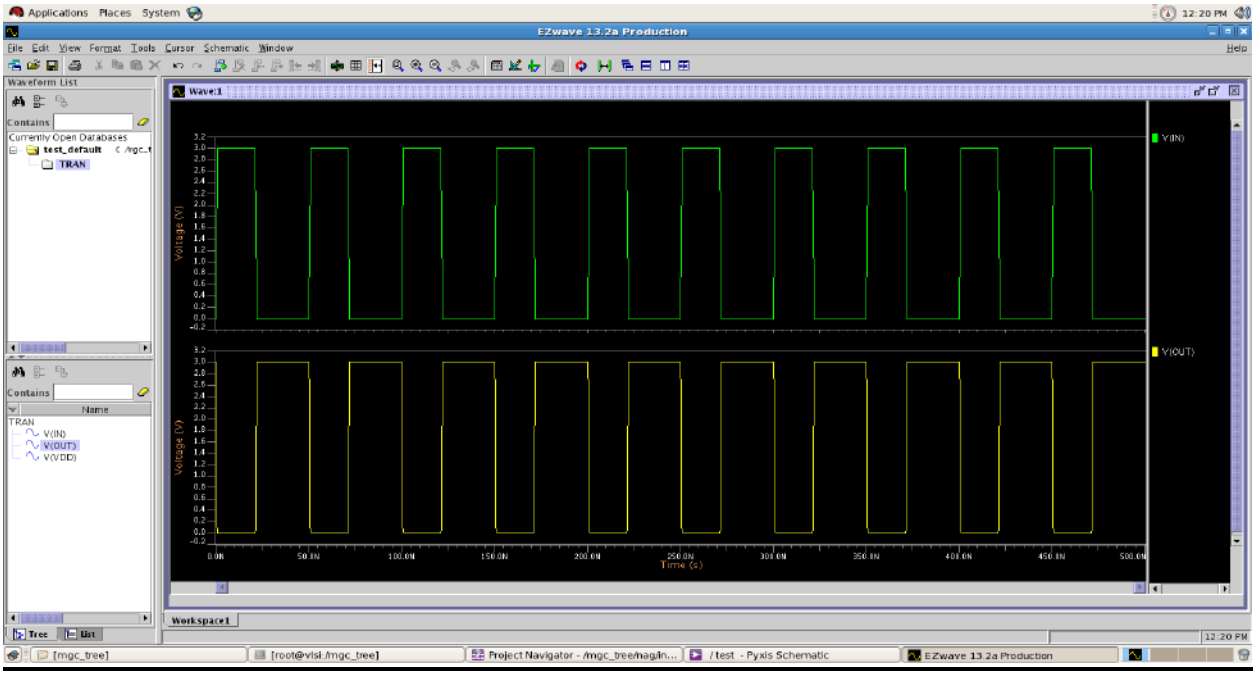
CIRCUIT DIAGRAM:



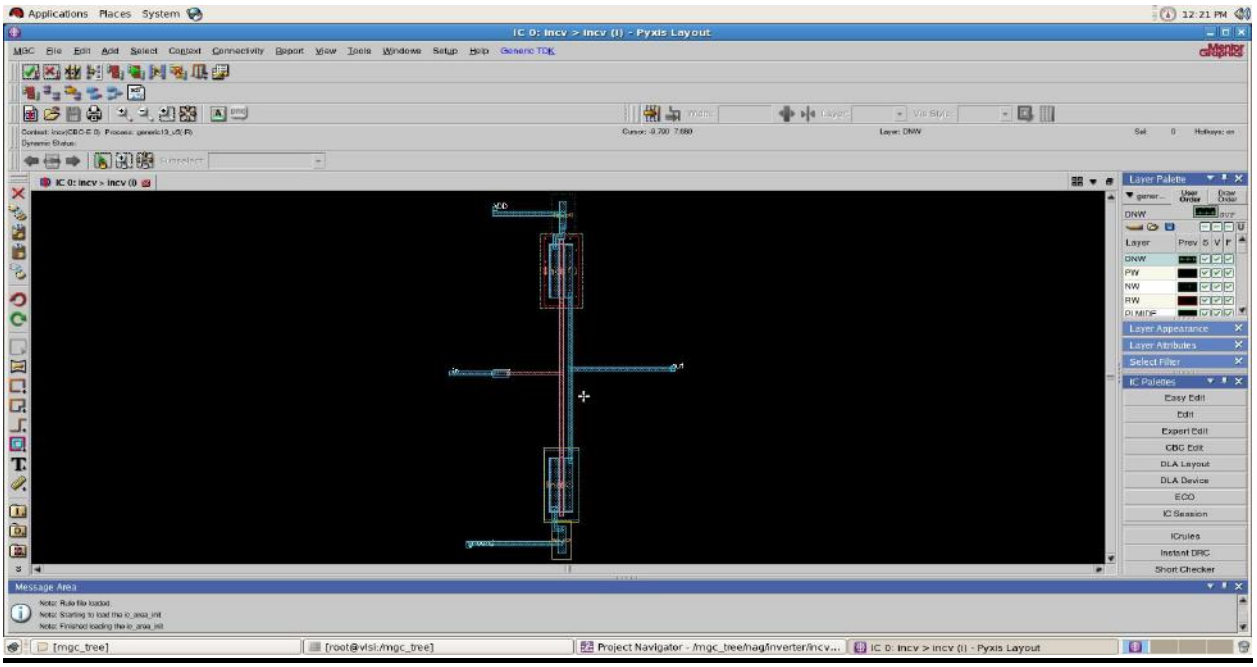
PROCEDURE:

1. Connect the Circuit as shown in the circuit diagram using Pyxis Schematic tool
2. Enter into Simulation mode.
3. Setup the Analysis and library.
4. Setup the required analysis.
5. Probe the required Voltages
6. Run the simulation.
7. Observe the waveforms in EZ wave.
8. Draw the layout using Pysis Layout.
9. Perform Routing using IRoute
10. Perform DRC, LVS, PEX.

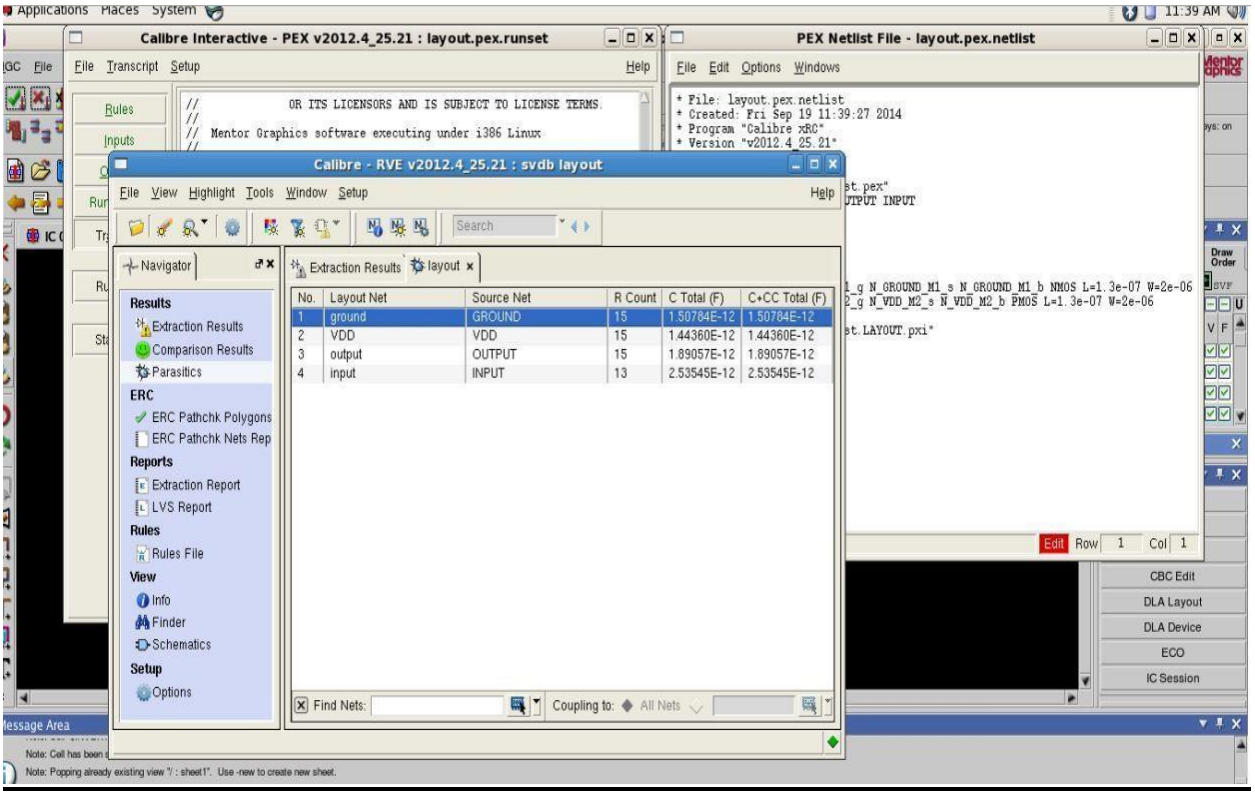
OUTPUT WAVEFORMS:



Layout:



Result Verification Environment:



Result:

Conclusion:

Viva Questions:

1. why Ex-or gate called an inverter?

2. what is the principle of inverter?

3. What is the basic inverter circuit of CMOS?

4. What are the characteristics of a CMOS inverter?

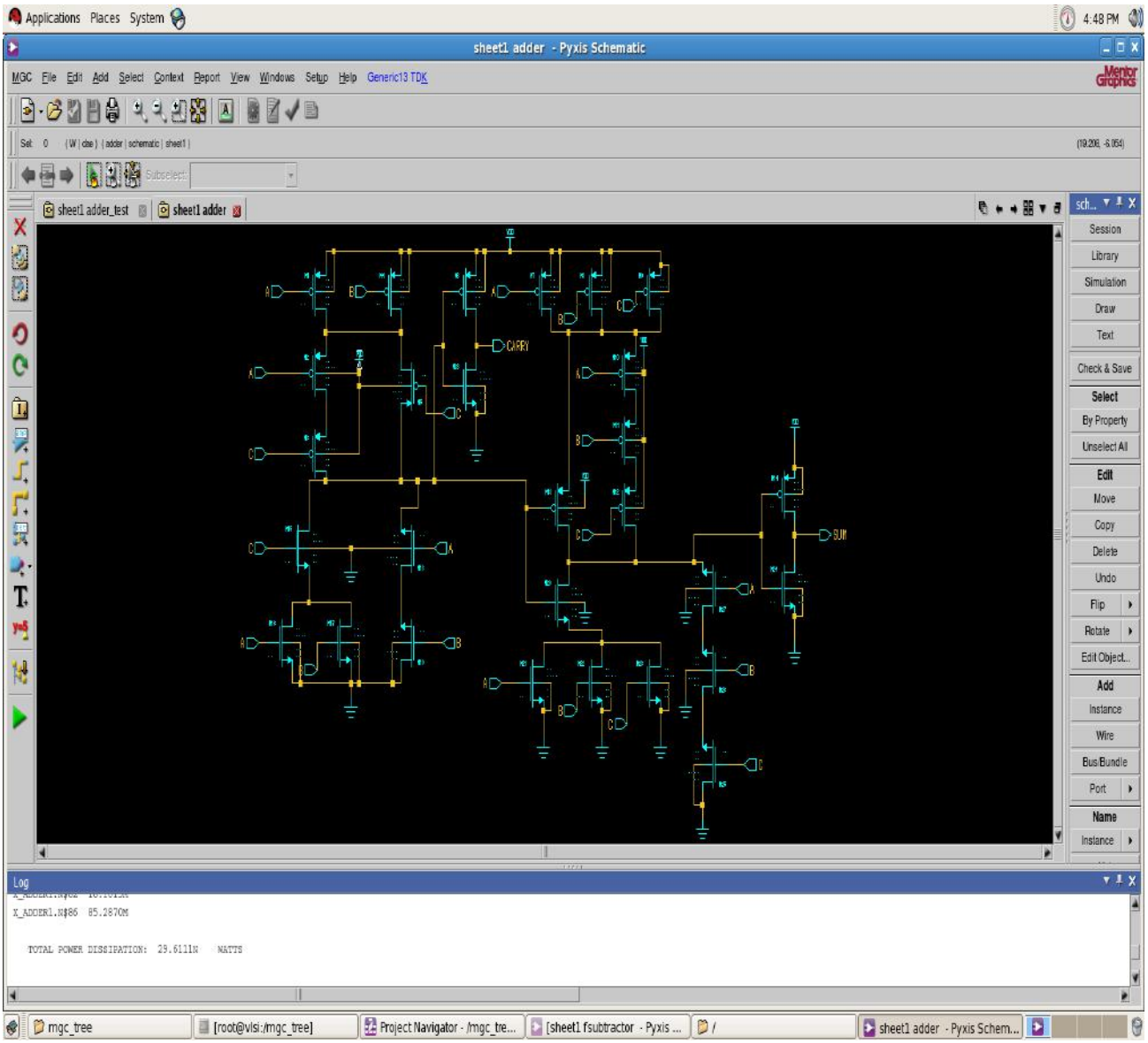
5. What is threshold voltage of CMOS inverter?

EXP NO.	Design and Implementation of Full Adder	DATE
10		

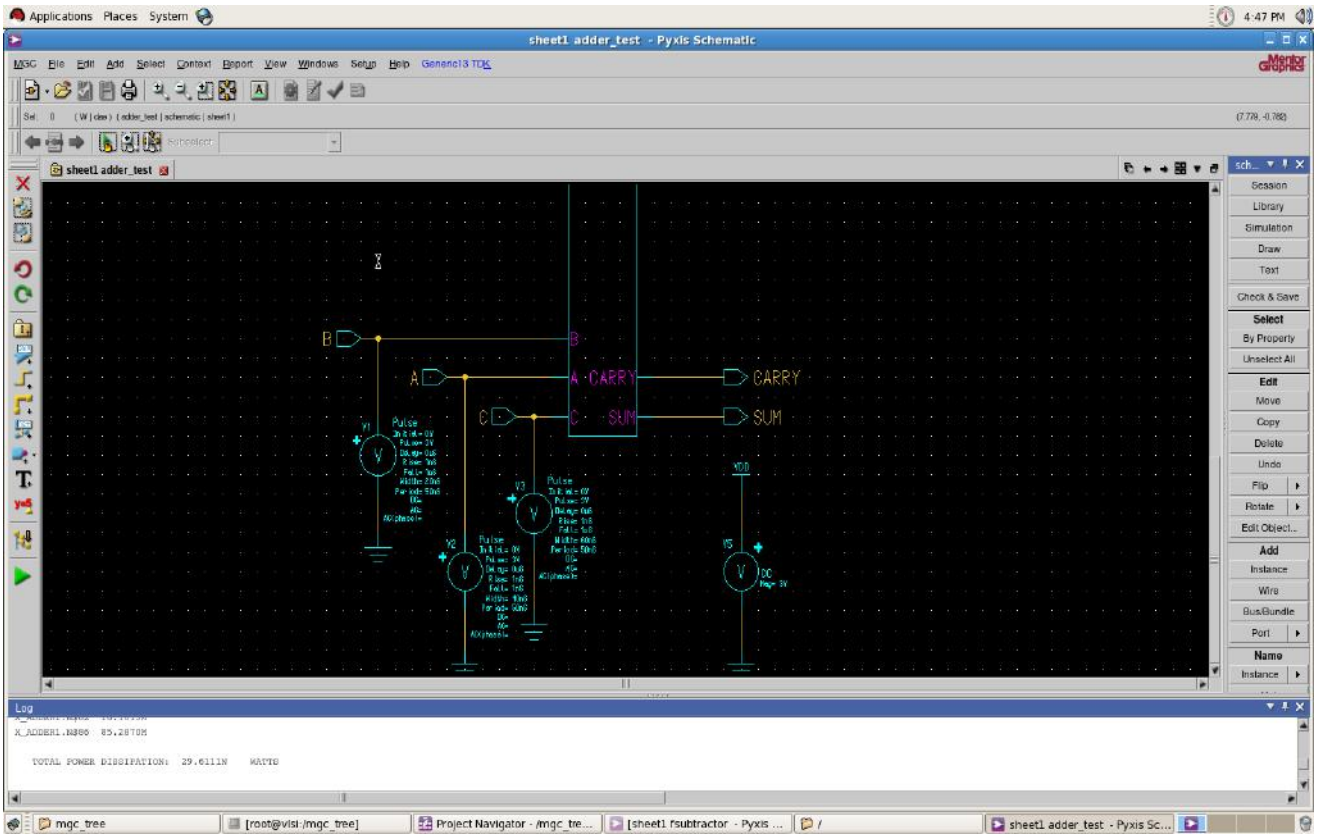
AIM: To design and Implementation of an Fulladder

Software Required: Mentor Graphics - Pyxis, AMS, Calibre.

CIRCUIT DIAGRAM:



Testbench:



output Waveforms:

Result:

Conclusion:

Viva Questions:

1. Which gate is used to design full adder?

2. What are the types of adders ?

3. How many MUX are required for full adder?

4. How do you implement a full adder using half adder?

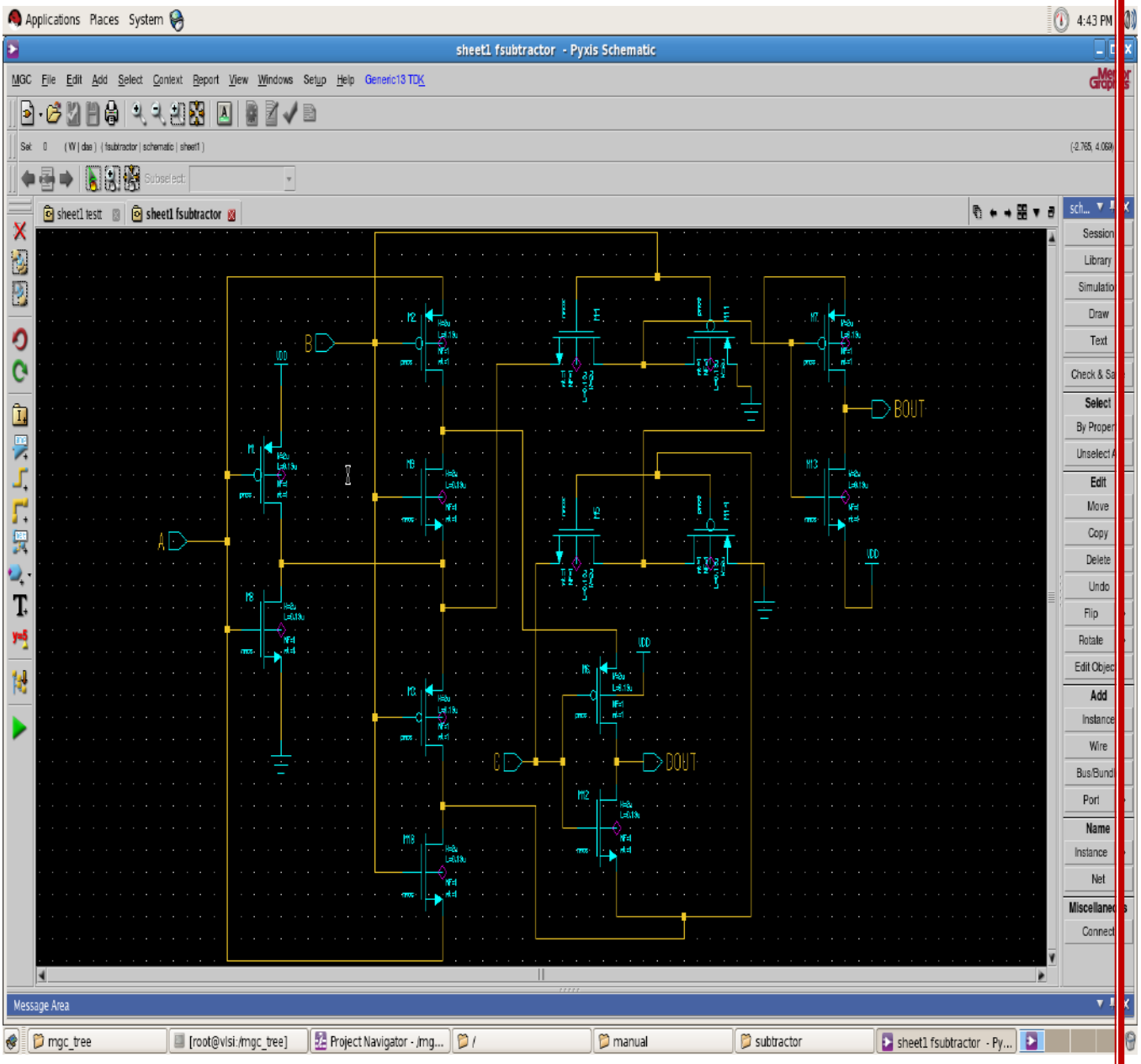
5. What are the advantages and disadvantages of full adder?

EXP NO.	Design and Implementation of Full Subtractor	DATE
11		

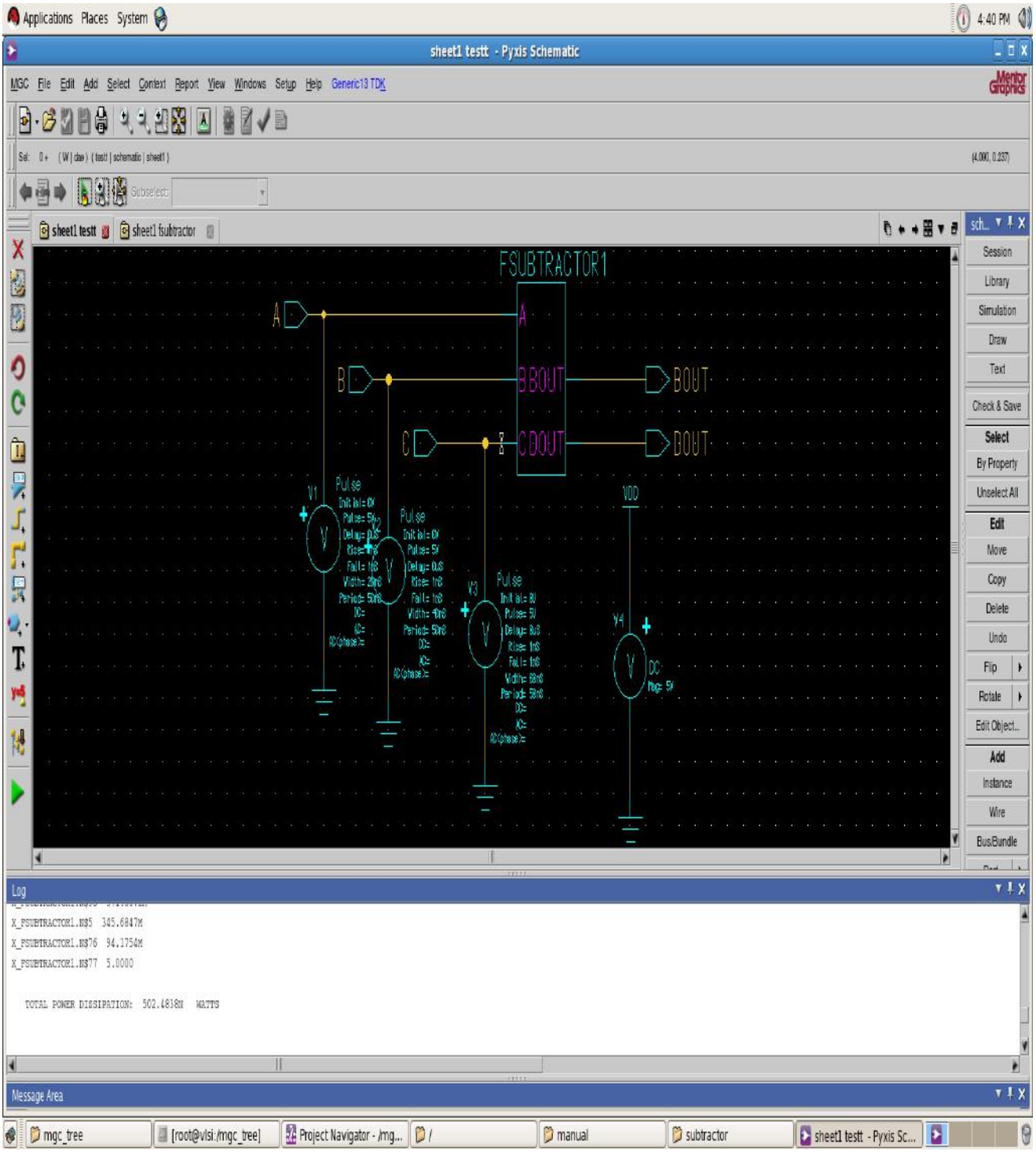
AIM: To design and Implementation of an Full-subtractor

Software Required: Mentor Graphics - Pyxis, AMS, Calibre.

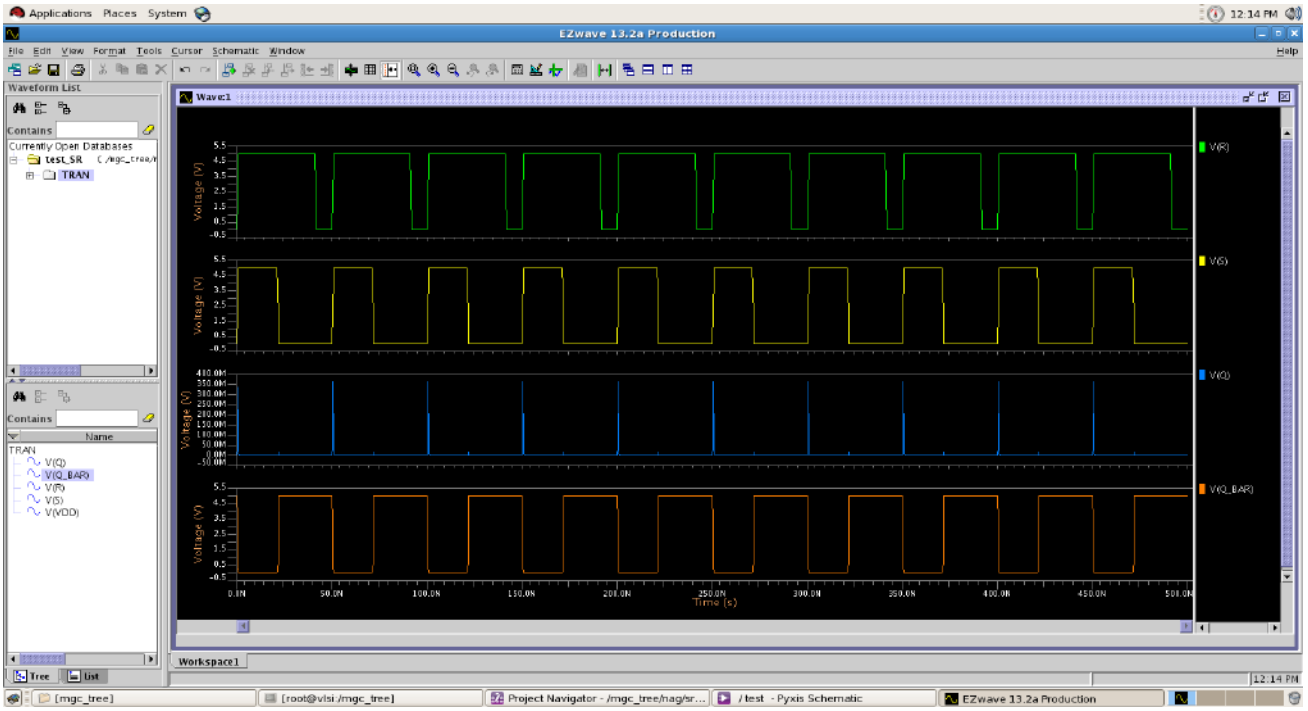
CIRCUIT DIAGRAM:



Testbench:


Applications Places System
4:40 PM
sheet1 testt - Pyxis Schematic
MGC File Edit Add Select Context Report View Windows Setup Help Generic13 TDK
sheet1 testt
sheet1 fsubtractor
FSUBTRACTOR1
A
B
C
BBOUT
CDOUT
BOUT
BOUT
VDD
V1
V3
V4
Pulse
DC
Session
Library
Simulation
Draw
Text
Check & Save
Select
By Property
Unselect All
Edit
Move
Copy
Delete
Undo
Flip
Rotate
Edit Object...
Add
Instance
Wire
Bus/Bundle
Log
X_FSUBTRACTOR1.M5 345.6847M
X_FSUBTRACTOR1.M76 94.1754M
X_FSUBTRACTOR1.M77 5.0000
TOTAL POWER DISSIPATION: 502.4818M WATTS
Message Area
mgc_tree
[root@vlsi:/mgc_tree]
Project Navigator - /mg...
/
manual
subtractor
sheet1 testt - Pyxis Sc...

Output waveforms:



Result:

Conclusion:

Viva Questions:

1. What are the advantages and disadvantages of full subtractor?

2. What is the difference between a half and full subtractor?

3. Which decoder is needed for full subtractor?

4. How many NAND gates make a full subtractor and design it?

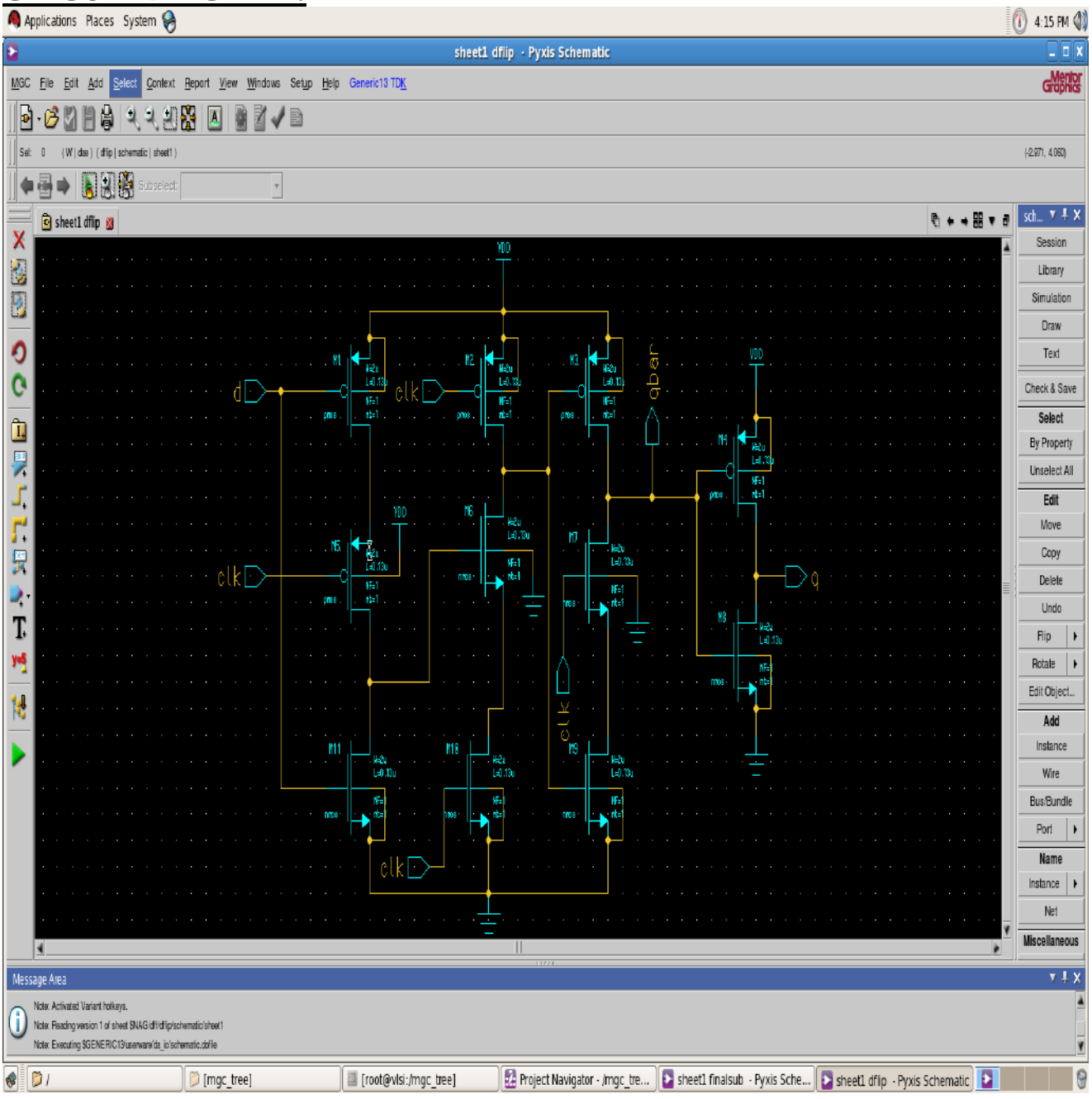
5. what is the expression for full subtractor?

EXP NO.	Design and Implementation of D-Latch	DATE
12		

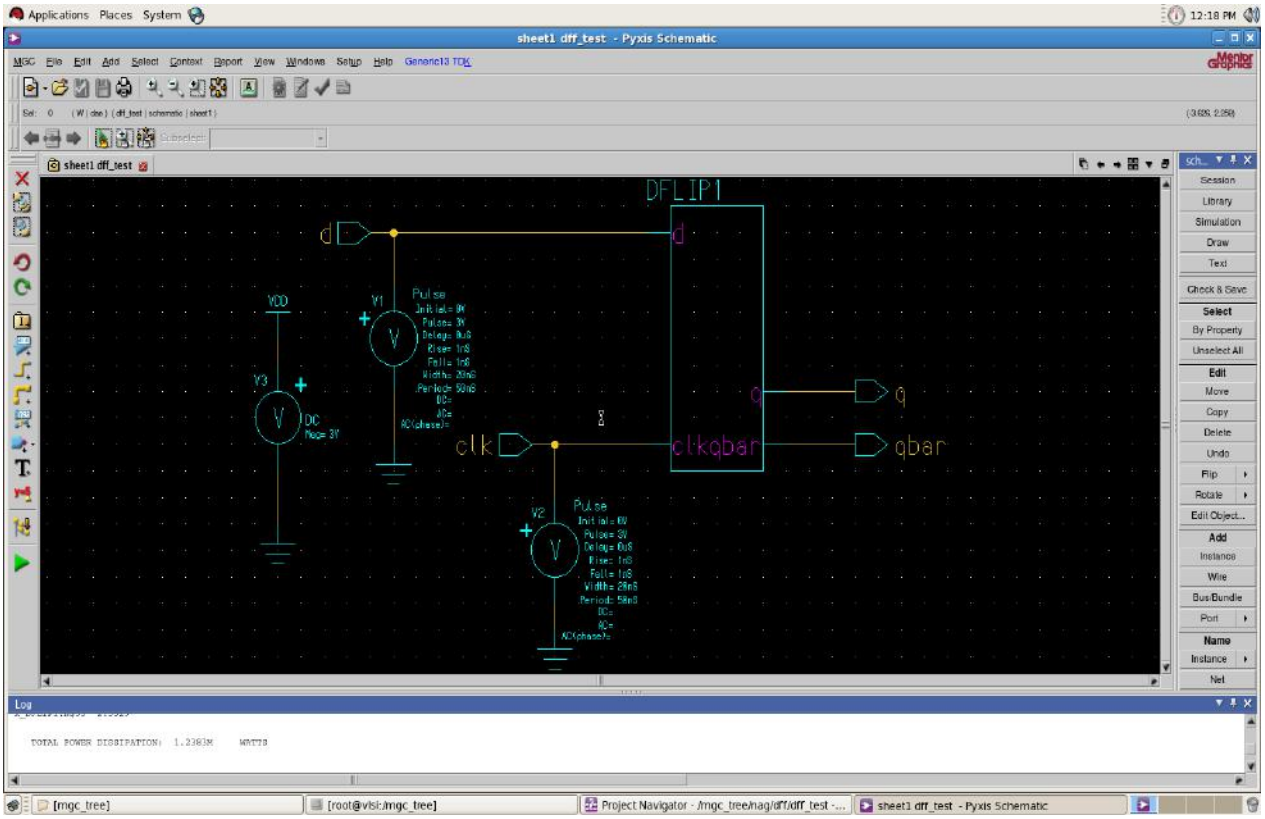
AIM: To design and Implementation of an D-Latch

Software Required: Mentor Graphics - Pyxis, AMS, Calibre.

CIRCUIT DIAGRAM:



Testbench:



Output Waveform:

Result:

Conclusion:

Viva Questions:

1. What is latch and its types?

2. What are the applications of D-Latch?

3. Why d flip-flop called data latch?

4. How do you convert SR latch to D latch?

5. Write the differences between latch and flip-flop?

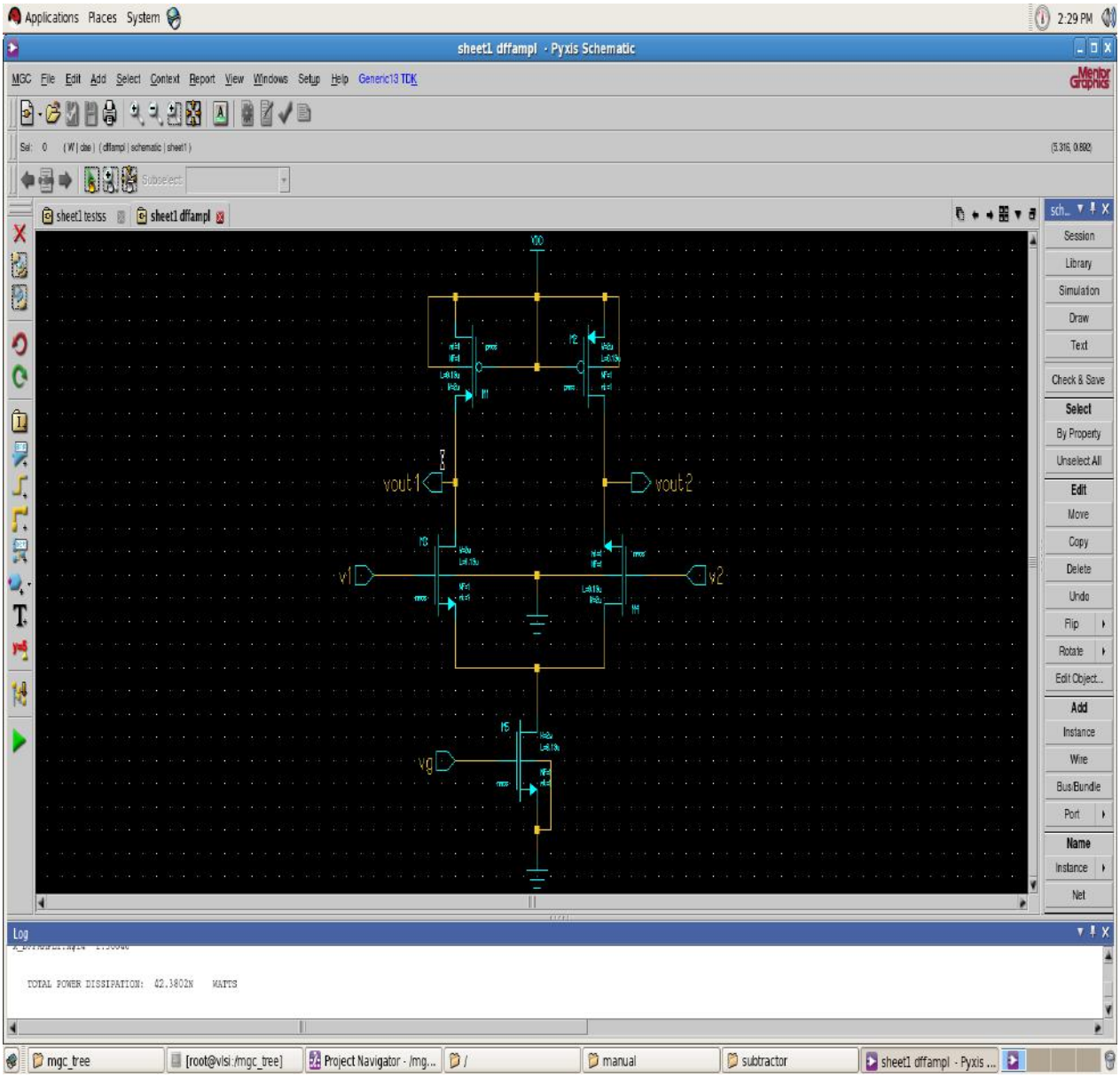
ADVANCED EXPERIMENT

EXP NO.	DESIGN AND IMPLEMENTATION DIFFERENTIAL AMPLIFIER	DATE
13		

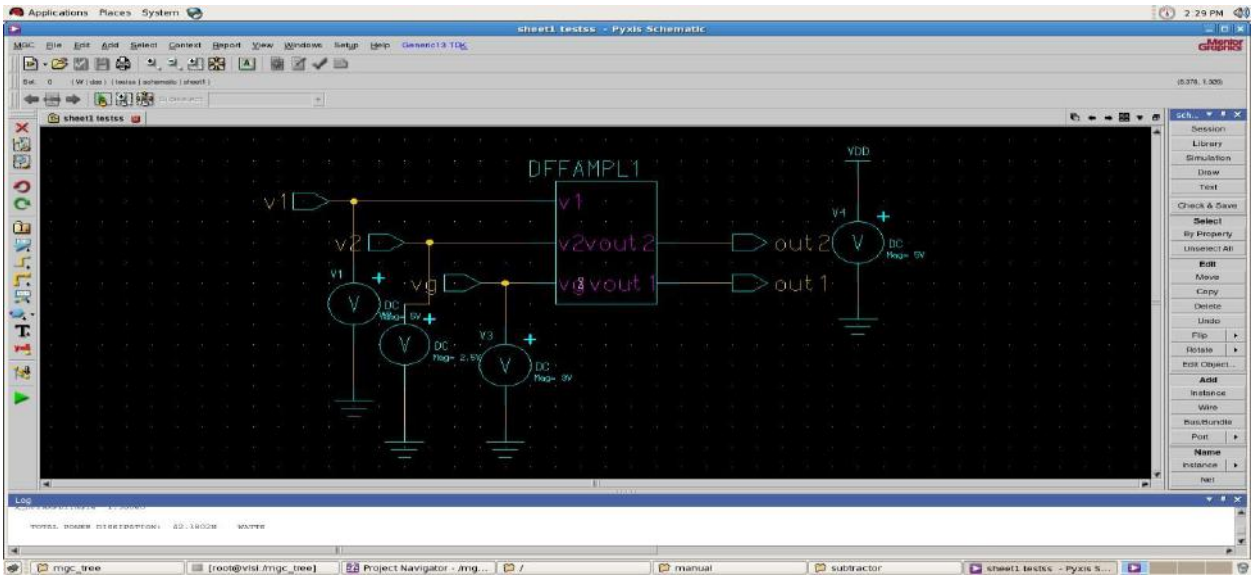
AIM: To design and Implementation of an Differential Amplifier

Software Required: Mentor Graphics - Pyxis, AMS, Calibre.

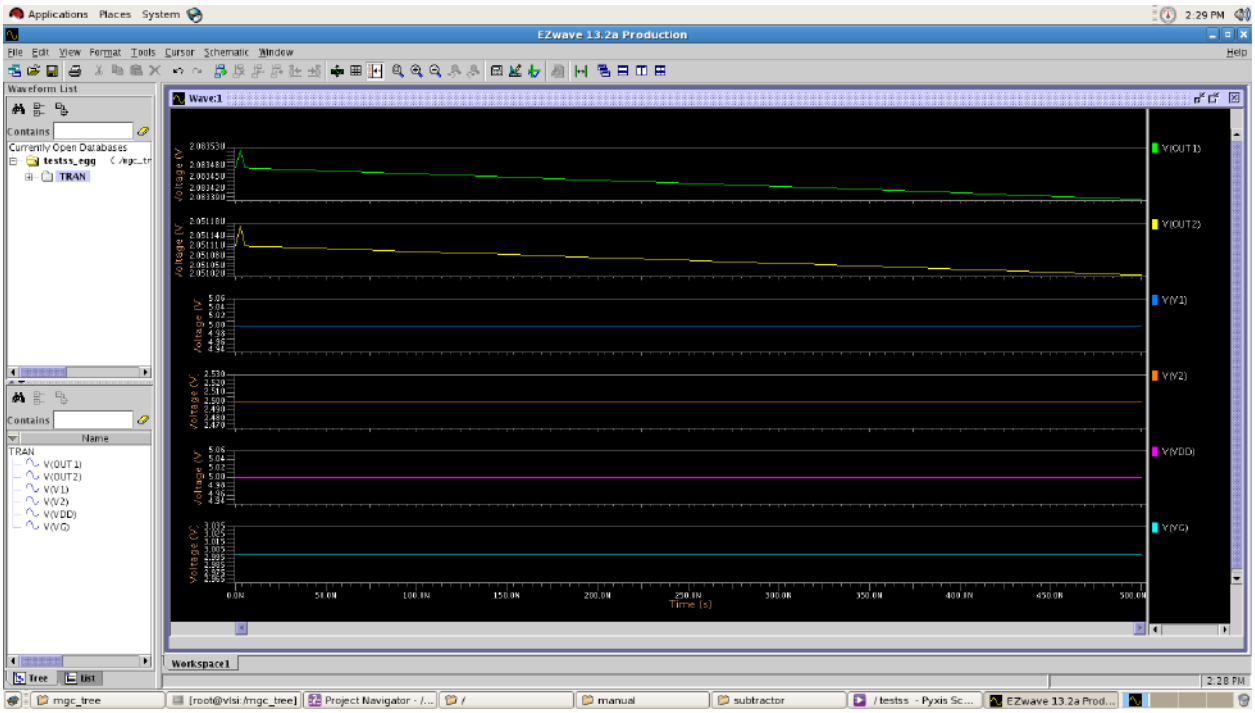
CIRCUIT DIAGRAM:



Testbench:



OUTPUT WAVEFORMS:



Result:

Conclusion:

Viva Questions:

1. Abbreviate CMOS and NMOS?

2. What are applications of Difference amplifier?

3. What Does Differential amplifier do?

4. What are the types of differential amplifier?

5. What are the advantages of differential amplifier?

BLOCK DIAGRAM:

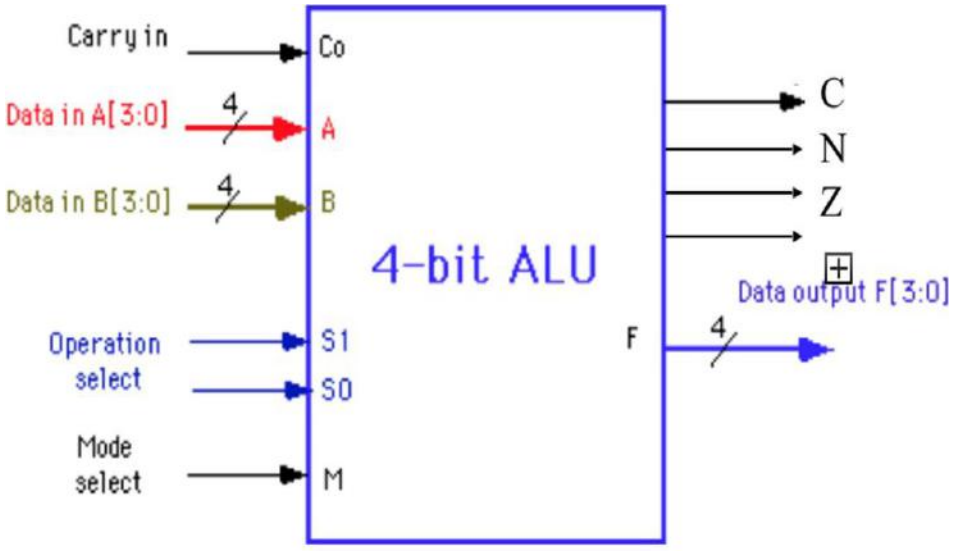
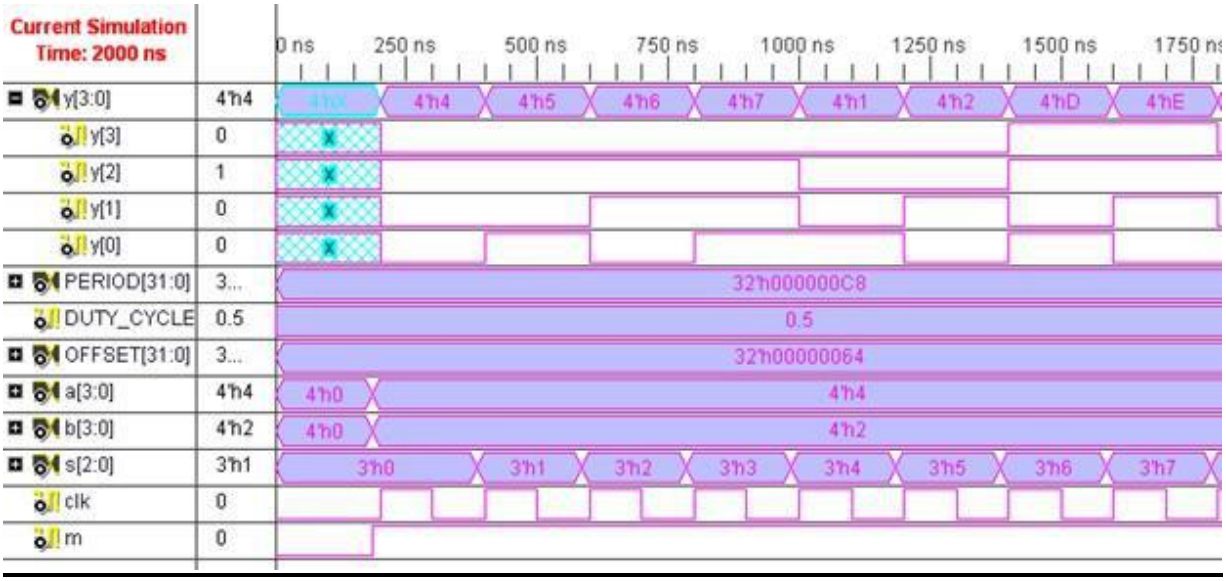


Figure 1: Block diagram of the 4-bit ALU.

OUTPUT WAVEFORMS:



EXP NO.	DESIGN AND IMPLEMENTATION OF ALU	DATE
14		

AIM:

To write a VHDL/Verilog code for ALU Design and to generate synthesis report, RTL schematic and to implement designs using FPGA (Spartan-2).

APPARATUS:

1. Synthesis Tool: Xilinx Project Navigator - ISE 9.1i.
2. Simulation Tool: Modelsim Simulator.

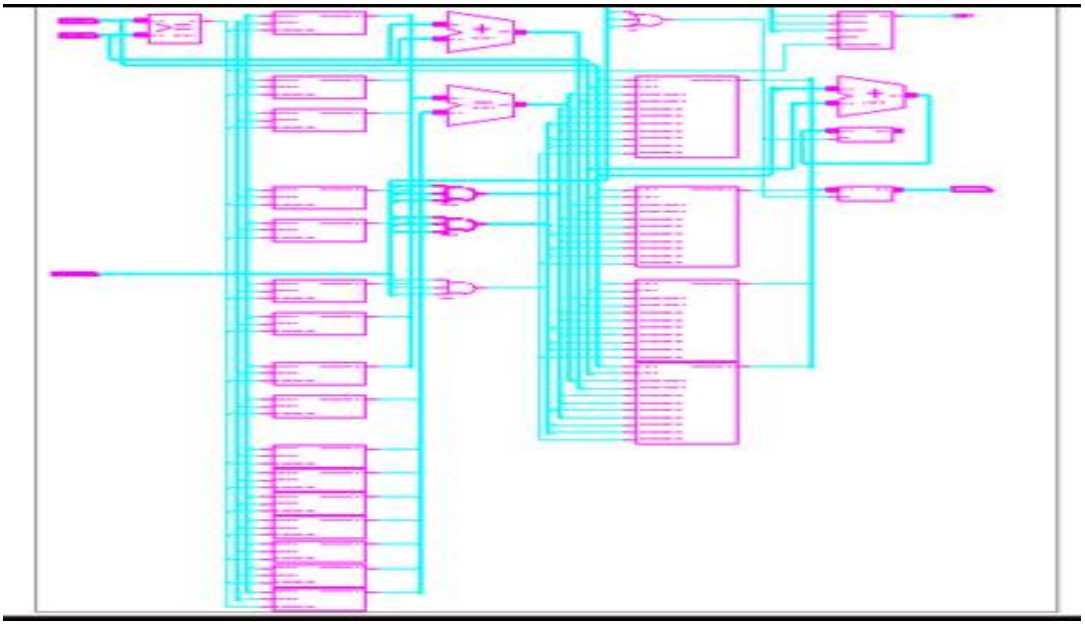
PROCEDURE:

1. Open Xilinx ISE 9.1i.
2. Create a new source file in a new project with suitable name.
3. Create the file in VHDL/Verilog module.
4. Select the appropriate input and output ports according to the requirements.
5. Type the program and save it and synthesize the process.
6. Select Synthesize XST, check for syntax errors and generate report and RTL schematic.
7. Create another new source.
8. Select source type as Test bench wave form.
9. Associate the test bench to the source.
10. Assign clock and timing details.
11. Give the input waveforms for the source.
12. Save the input waveforms and perform behavioral simulation.
13. The simulated output waveforms window will be shown.

DUMPING PROCESS:

1. In the process window, go to ‘user constraints’ and select ‘assign package pins’ and after that double click on ‘implement design’.
2. Select properties by right clicking on ‘generate programming file’. Select ‘JTAG’ clock in startup options.
3. Select boundary scan in ‘impact window’ after double clicking on ‘configure Device’.
4. In ‘generate programming file’ double clicking on ‘programming file generation report. Bit file will be generated.
5. Xilinx boundary scan window will appear when the bit file is selected. Right click on Xilinx component and select program.
6. Programming properties will appear and finally program will be succeeded.
7. Thus the program can be dumped into FPGA kit and finally output can be seen on the kit.

RTL SCHEMATIC:



TRUTH TABLE:

m=0 Logic				
s(2)	**s(1)**	**s(0)**	**Function**	**Operation (bit wise)**
0	0	0	~a	NOT
0	0	1	~b	NOT
0	1	0	a&b	AND
0	1	1	a|b	OR
1	0	0	~(a&b)	NAND
1	0	1	~(a|b)	NOR
1	1	0	a^b	EXOR
1	1	1	~(a^b)	EXNOR
m=1 Arithmetic				
0	0	0	a	Transfer a
0	0	1	a+1	Increment a by 1
0	1	0	a+b	Add a and b
0	1	1	a+b+1	Increment the sum of a and b by 1
1	0	0	a+(~b)	a plus one's complement of b
1	0	1	a-b	Subtract b from a (i.e. ~b+a+1)
1	1	0	(~a)+b	b plus one's compliment of a
1	1	1	b-a	Subtract a from b (i.e. ~a+b+1)

VHDL CODE:

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

entity alu4b is
Port ( a,b : in  STD_LOGIC_VECTOR (3 downto 0);
          s : in  STD_LOGIC_VECTOR (2 downto 0);
          m,clk : in  STD_LOGIC;
          y: out  STD_LOGIC_VECTOR (3 downto 0));
end alu4b;
architectureBehavioral of alu4b is
begin
process(a,b,s,m)
begin
if(clk'event and clk='1')then
elsif(m='0')then
case s is
when"000"=>y<=not a;
when"001"=>y<=not b;
when"010"=>y<=a and b;
when"011"=>y<=a or b;
when"100"=>y<=a nand b;
when"101"=>y<=a nor b;
when"110"=>y<=a xor b;
when"111"=>y<=a xnor b;
when others=>null;
end case;
else
case s is
when"000"=>y<=a;
when"001"=>y<=a+1;
when"010"=>y<=a+b;
when"011"=>y<=a+b+1;
when"100"=>y<=a+(not b);
when"101"=>y<=a-b;
when"110"=>y<=(not a)+b;
when"111"=>y<=b-a;
when others=>y<=b;
end case;
end if;
end process;
end Behavioral;
```

EVICE UTILIZATION SUMMARY:

Number of Slices	: 2 out of	960	0%
Number of Slice Flip Flops	: 4 out of	1920	0%
Number of 4 input LUTs	: 4 out of	1920	0%
Number of IOs	: 6		
Number of bonded IOBs	: 6 out of	66	9%
Number of GCLKs	: 1 out of	24	4%

SYNTHESIS REPORT:

RTL Top Level Output File Name : alu.ngr
Top Level Output File Name : alu
Output Format : NGC
Optimization Goal : Speed
Keep Hierarchy : NO
Design Statistics
IOs : 6

Cell Usage :

BELS : 4
INV : 1
LUT3 : 1
LUT4 : 2
FlipFlops/Latches : 4
FDC : 4
Clock Buffers : 1
BUFGP : 1
IO Buffers : 5
IBUF : 1
OBUF : 4

RESULT:

CONCLUSION:

VIVA QUESTIONS:

1. What is the purpose of ALU?

2. What are the functional blocks of ALU?

3. In a 16-bit ALU, what does the number ‘16’ indicates?

4. Draw the schematic of Subtractor- using adder circuit.

5. What are the advantages of ALU Design?