School of Computer Science Engineering and Technology

Course- BTech
Course Code- CSET105
Year- 2022
Date-

Type- Core
Course Name- Digital Design
Semester- Odd
Batch-

1.2 Lab Assignment # 2

In this Lab, we shall start Verilog. To run Verilog, we shall use http://www.edaplayground.com. After opening you will find Design or Testbench window pane. In the Design window you need write the Verilog Code and in the Testbench window, the testbench verification code will be written.

1. Write a Verilog code to implement AND gate. Write the corresponding Testbench code for the verification of your Verilog code.

   Truth Table of AND Gate:

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

2. Write a Verilog code to implement OR gate. Write the corresponding Testbench code for the verification of your Verilog code.

   Truth Table of OR Gate:

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

3. Write a Verilog code to implement XOR gate. Write the corresponding Testbench code for the verification of your Verilog code.

   Truth Table of XOR Gate:

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

1	1	0

4. Write a Verilog code to implement NOR gate. Write the corresponding Testbench code for the verification of your Verilog code.

   Truth Table of NOR Gate:

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

5. Write a Verilog code to implement NAND gate. Write the corresponding Testbench code for the verification of your Verilog code.

   Truth Table of NAND Gate:

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

6. Write a Verilog code to implement XNOR gate. Write the corresponding Testbench code for the verification of your Verilog code.

   Truth Table of XNOR Gate:

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

Procedure to run the programs in EDAPlayground:

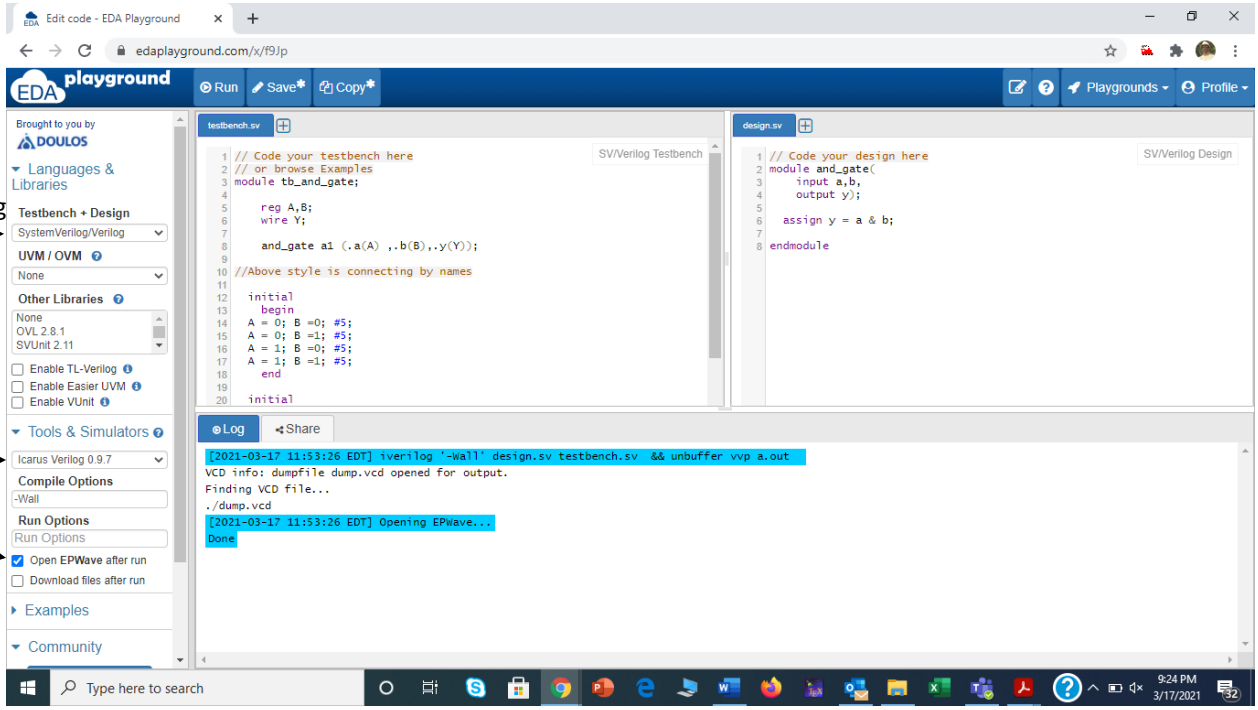
- **Open** http://www.edaplayground.com
- **Go to Design Window to write the Verilog Code.**
- **Go to TestBench Window to write the TestBench Verification Code.**
- **All the other setup should be done according to the following Screenshot**

School of Computer Science Engineering and Technology

(1) Select SystemVerilog

(2) Select Icarus Verilog 0.9.7 in Tools and Simulator

(3) Make it sure the check box has been clicked “Open EPWave after run”



- **Sample Verilog Code for AND Gate:**

```
module and_gate(
   input a,b,
   output y);
   assign y = a & b;
endmodule
```

- **Sample TestBench Code for the AND Gate:**

```
module tb_and_gate;

   reg A,B;
   wire Y;
   and_gate a1 (.a(A) ,.b(B),.y(Y));

initial
   begin
         A = 0; B =0; #5;
         A = 0; B =1; #5;
         A = 1; B =0; #5;
         A = 1; B =1; #5;
   end
initial
  begin
   $dumpfile("dump.vcd");
```

```
  $dumpvars(1);
 end

endmodule
```

Submission Instructions:

- Prepare the submission file according to the following process:
    1. Copy the Verilog code, the Test Bench Code in a Word File.
    2. Take the ScreenShot of Waveform and paste into the same word file.
    3. Repeat Step 1 and 2 for all the programs.
    4. Copy and Paste all the Verilog code, Testbench Code and Waveform into a single word file as 1_verilog, 1_TestBench, 1_Waveform, 2_verilog, 2_TestBench, 2_Waveform… etc.
    5. Convert it into pdf file, name it as **RollNo_Assignment# (Example: E20CSE001_ Assignment2.pdf)**.
    6. Submit your file on LMS **within the deadline**.

- Write your **Name and Roll No. as comment before starting of each program**. Keep in mind this is **Mandatory**. Failing which you may lose your marks.
- Make it sure that in each program, **you have mentioned enough comments** regarding the explanation of program instructions.
- **Each student will submit their assignment on their corresponding group slot only.**
- Late submission will lead to penalty.
- Any form of plagiarism/copying from peer or internet sources will lead penalty.
- Following of all instructions at submission time is mandatory. Missing of any instructions at submission time will lead penalty.

SOUMYA DUBEY

E21CSEU0760

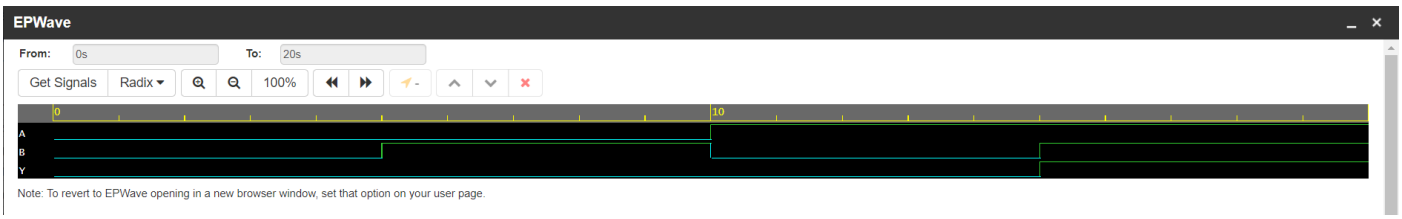
1. Write a Verilog code to implement AND gate. Write the corresponding Testbench code for the verification of your Verilog code.

testbench.sv

```
1  //Soumya Dubey
2  //E21CSEU0760
3
4
5  module tb_and_gate;
6  reg A,B;
7  wire Y;
8  and_gate a1 (.a(A) ,.b(B),.y(Y));
9  initial
10 begin
11 A = 0; B =0; #5;
12 A = 0; B =1; #5;
13 A = 1; B =0; #5;
14 A = 1; B =1; #5;
15 end
16 initial
17 begin
18 $dumpfile("dump.vcd");
19 $dumpvars(1);
20 end
21 endmodule
```

design.sv

```
1 module and_gate(
2 input a,b,
3 output y);
4   assign y = a & b;
5 endmodule
```



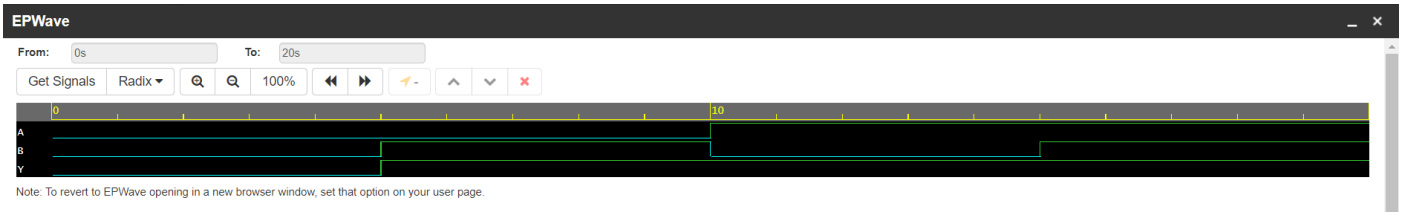
2. Write a Verilog code to implement OR gate. Write the corresponding Testbench code for the verification of your Verilog code.

testbench.sv

```
1  //Soumya Dubey
2  //E21CSEU0760
3
4
5  module tb_and_gate;
6  reg A,B;
7  wire Y;
8  and_gate a1 (.a(A) ,.b(B),.y(Y));
9  initial
10 begin
11 A = 0; B =0; #5;
12 A = 0; B =1; #5;
13 A = 1; B =0; #5;
14 A = 1; B =1; #5;
15 end
16 initial
17 begin
18 $dumpfile("dump.vcd");
19 $dumpvars(1);
20 end
21 endmodule
```

design.sv

```
1  module and_gate(
2  input a,b,
3  output y);
4    assign y = a | b;
5  endmodule
```



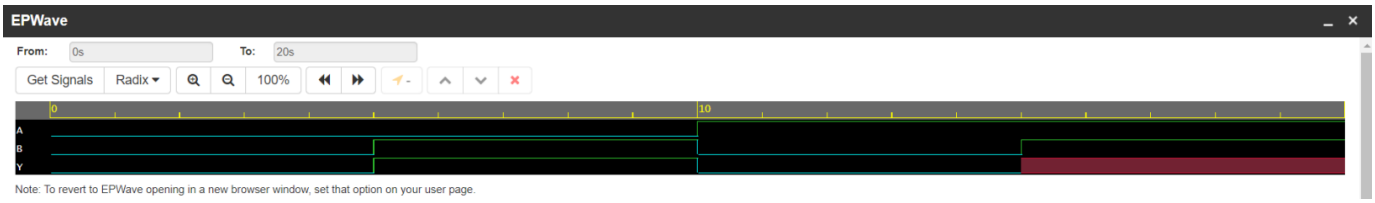
3. Write a Verilog code to implement XOR gate. Write the corresponding Testbench code for the verification of your Verilog code.

testbench.sv

```
//Soumya Dubey
//E21CSEU0760


module tb_and_gate;
reg A,B;
wire Y;
and_gate a1 (.a(A) ,.b(B),.y(Y));
initial
begin
A = 0; B =0; #5;
A = 0; B =1; #5;
A = 1; B =0; #5;
A = 1; B =1; #5;
end
initial
begin
$dumpfile("dump.vcd");
$dumpvars(1);
end
endmodule
```

design.sv

```
module and_gate(
input a,b,
output y);
  assign y = ((~a&b) | (a&(~b)));
endmodule
```



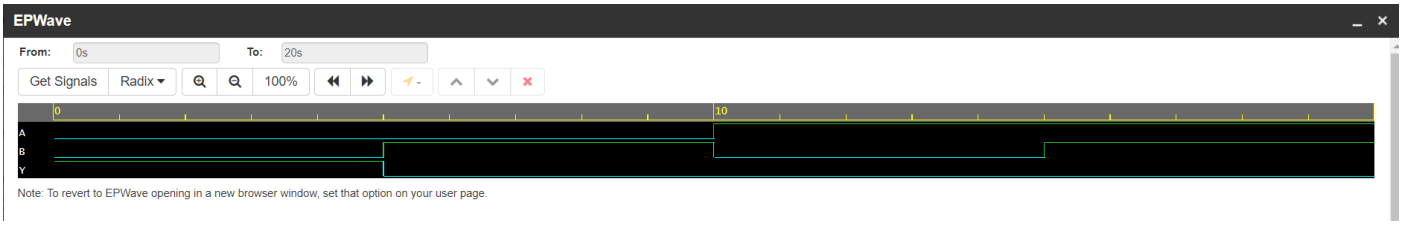
4. Write a Verilog code to implement NOR gate. Write the corresponding Testbench code for the verification of your Verilog code.

testbench.sv

```
//Soumya Dubey
//E21CSEU0760


module tb_and_gate;
reg A,B;
wire Y;
and_gate a1 (.a(A) ,.b(B),.y(Y));
initial
begin
A = 0; B =0; #5;
A = 0; B =1; #5;
A = 1; B =0; #5;
A = 1; B =1; #5;
end
initial
begin
$dumpfile("dump.vcd");
$dumpvars(1);
end
endmodule
```

design.sv

```
module and_gate(
input a,b,
output y);
  assign y = ~(a | b);
endmodule
```



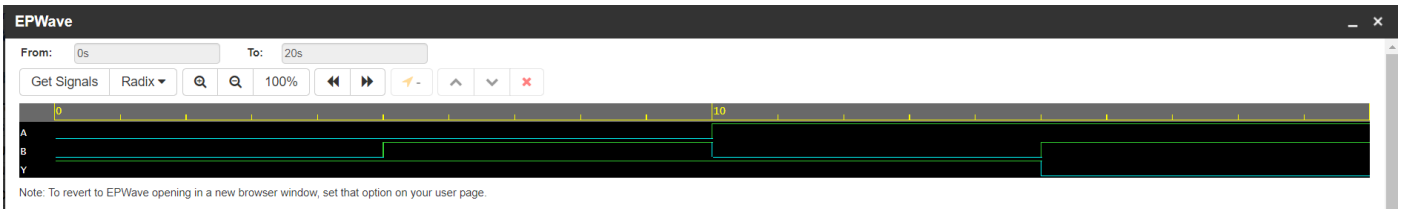
5. Write a Verilog code to implement NAND gate. Write the corresponding Testbench code for the verification of your Verilog code.

testbench.sv

```
//Soumya Dubey
//E21CSEU0760


module tb_and_gate;
reg A,B;
wire Y;
and_gate a1 (.a(A) ,.b(B),.y(Y));
initial
begin
A = 0; B =0; #5;
A = 0; B =1; #5;
A = 1; B =0; #5;
A = 1; B =1; #5;
end
initial
begin
$dumpfile("dump.vcd");
$dumpvars(1);
end
endmodule
```

design.sv

```
module and_gate(
input a,b,
output y);
  assign y = ~(a & b);
endmodule
```



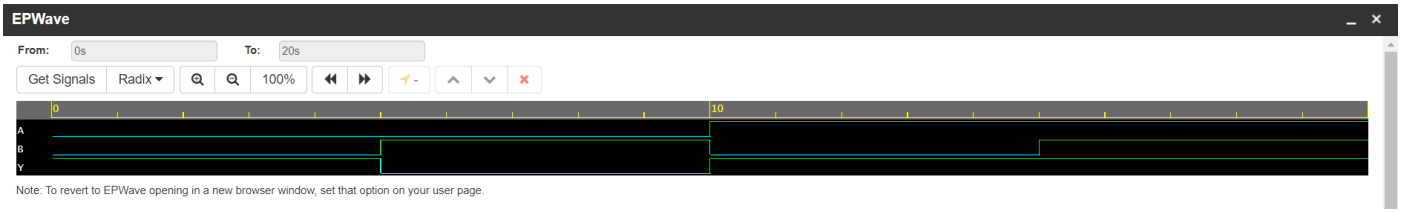
6. Write a Verilog code to implement XNOR gate. Write the corresponding Testbench code for the verification of your Verilog code.

testbench.sv

```
//Soumya Dubey
//E21CSEU0760


module tb_and_gate;
reg A,B;
wire Y;
and_gate a1 (.a(A) ,.b(B),.y(Y));
initial
begin
A = 0; B =0; #5;
A = 0; B =1; #5;
A = 1; B =0; #5;
A = 1; B =1; #5;
end
initial
begin
$dumpfile("dump.vcd");
$dumpvars(1);
end
endmodule
```

design.sv

```
module and_gate(
input a,b,
output y);
  assign y = ~((~a)&b)|(a&(~b));
endmodule
```



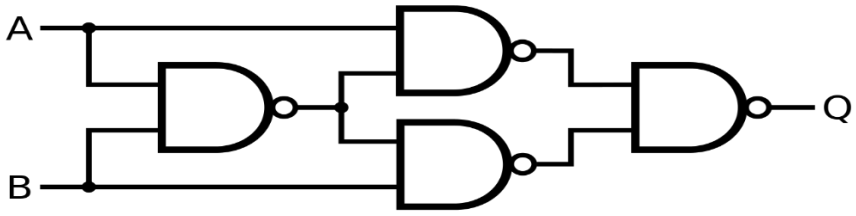
Lab Assignment 3

1. Perform the following operation on a given Boolean expression:
   F(ABC) = A'BC + A'B'C' + ABC
   a. Write a truth table for each Boolean expression.
   b. Draw a schematic diagram for each Boolean expression.
   c. Write a Verilog code for each Boolean expression and then test using wave form and compare with truth table whether your circuit produced same output or not?

2. Perform the following operation on a given Boolean expression:
   F(ABCD) = (A + B + C'+ D) (AB + BC + CA + D'C)
   a. Write a truth table for each Boolean expression.
   b. Draw a schematic diagram for each Boolean expression.
   c. Write a Verilog code for each Boolean expression and then test using wave form and compare with truth table whether your circuit produced same output or not?

3. Representation of XNOR gate using only NOR gates and perform the following operations:
   a. Derive the Boolean expression.
   b. Write the truth table for the above expression
   c. Write a Verilog code for each Boolean expression and then test using wave form and compare with truth table whether your circuit produced same output or not?

4. For the given circuit diagram do the following:
   a) Derive the Boolean expression.
   b) Write the truth table for the above expression
   c) Write a Verilog code for each Boolean expression and then test using wave form and compare with truth table whether your circuit produced same output or not?

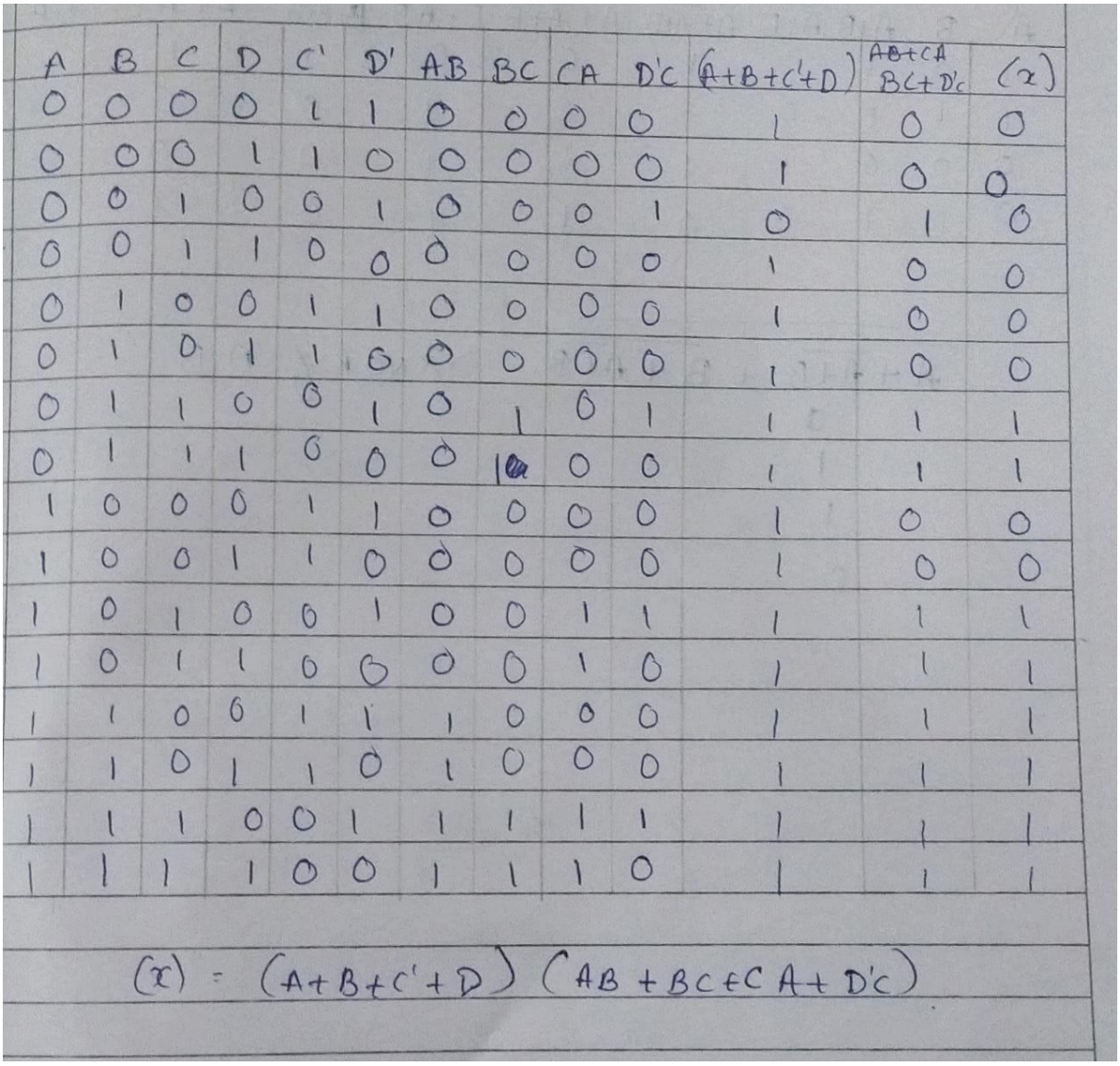


Lab Assignment 3

Name- Soumya Dubey
Roll No. – E21CSEU0760

1. **Perform the following operation on a given Boolean expression:**
   F(ABC) = A'BC + A'B'C' + ABC
   a. **Write a truth table for each Boolean expression.**
   b. **Draw a schematic diagram for each Boolean expression.**
   c. **Write a Verilog code for each Boolean expression and then test using wave form and compare with truth table whether your circuit produced same output or not?**

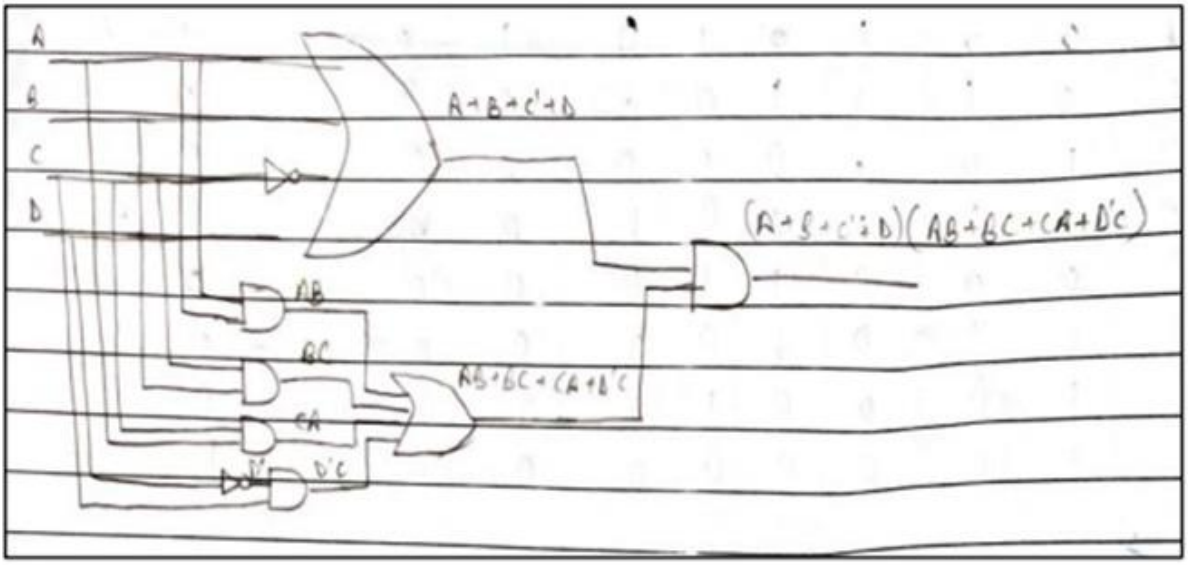
1.A



A	B	C	D	C'	D'	AB	BC	CA	D'C	(A+B+C'+D)	AB+CA / BC+D'C	(x)
0	0	0	0	1	1	0	0	0	0	1	0	0
0	0	0	1	1	0	0	0	0	0	1	0	0
0	0	1	0	0	1	0	0	0	1	0	1	0
0	0	1	1	0	0	0	0	0	0	1	0	0
0	1	0	0	1	1	0	0	0	0	1	0	0
0	1	0	1	1	0	0	0	0	0	1	0	0
0	1	1	0	0	1	0	1	0	1	1	1	1
0	1	1	1	0	0	0	0	0	0	1	1	1
1	0	0	0	1	1	0	0	0	0	1	0	0
1	0	0	1	1	0	0	0	0	0	1	0	0
1	0	1	0	0	1	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0	1	0	1	1	1
1	1	0	0	1	1	1	0	0	0	1	1	1
1	1	0	1	1	0	1	0	0	0	1	1	1
1	1	1	0	0	1	1	1	1	1	1	1	1
1	1	1	1	0	0	1	1	1	0	1	1	1

$(x) = (A+B+C'+D)(AB+BC+CA+D'C)$

1.B



1.C

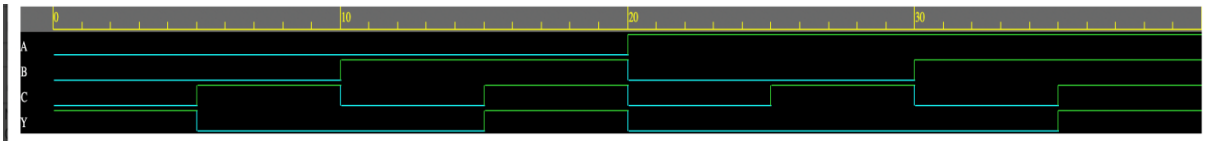
Verilog

```
module Ans1(
inputa,b,c,
    output y);
    assign y = ((~a&b& c) | (~a& ~b& ~c) | (a&b& c));
endmodule
```

TestBench

```
module Q1;
    reg A,B,C;
    wire Y;
  Ans1 a1 (.a(A) ,.b(B),.c(C),.y(Y));
initial
    begin
A = 0; B =0; C=0; #5;
A = 0; B =0; C=1; #5;
A = 0; B =1; C=0; #5;
A = 0; B =1; C=1; #5;
A = 1; B =0; C=0; #5;
A = 1; B =0; C=1; #5;
A = 1; B =1; C=0; #5;
A = 1; B =1; C=1; #5;
    end
initial
  begin
    $dumpfile("dump.vcd");
    $dumpvars(1);
  end
endmodule
```

Waveform



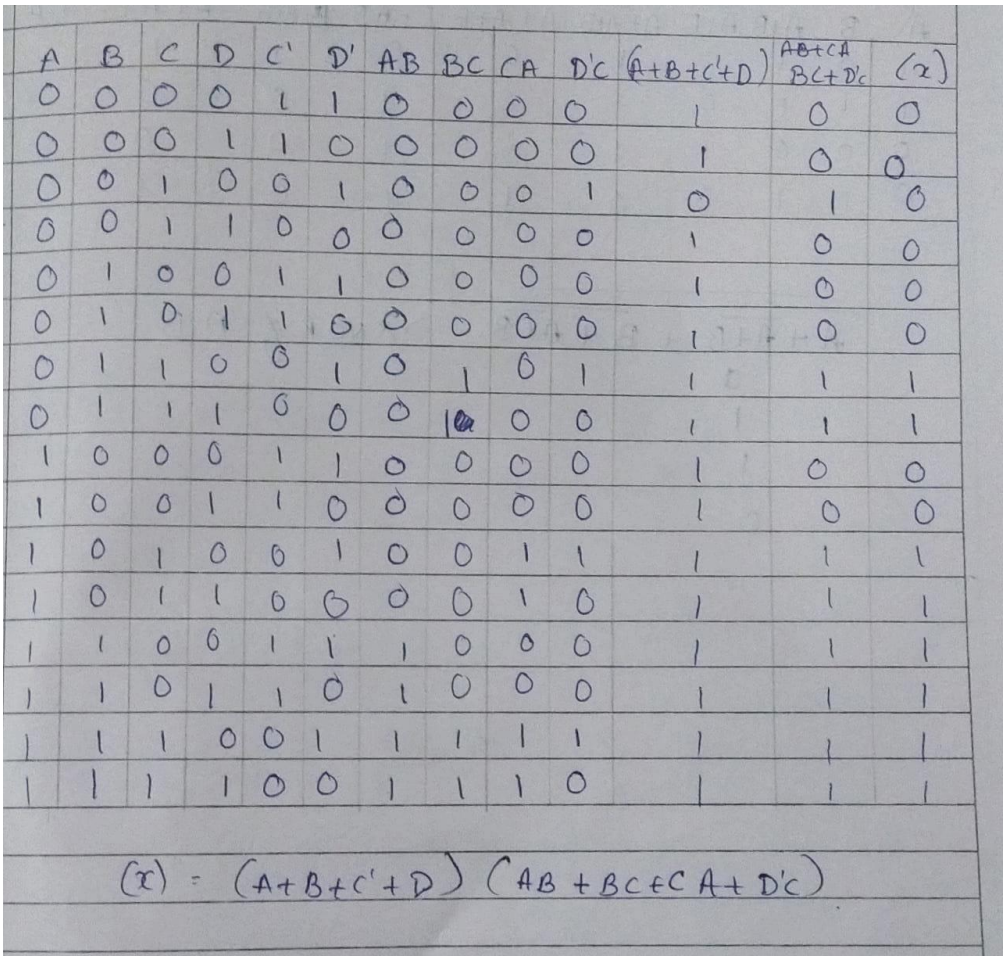
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

Comparison:

On comparison we got same result of waveform as our truth table output

2. **Perform the following operation on a given Boolean expression: F(ABCD) = (A + B + C'+ D) (AB + BC + CA + D'C)**
   a. **Write a truth table for each Boolean expression.**
   b. **Draw a schematic diagram for each Boolean expression.**
   c. **Write a Verilog code for each Boolean expression and then test using wave form and compare with truth table whether your circuit produced same output or not?**

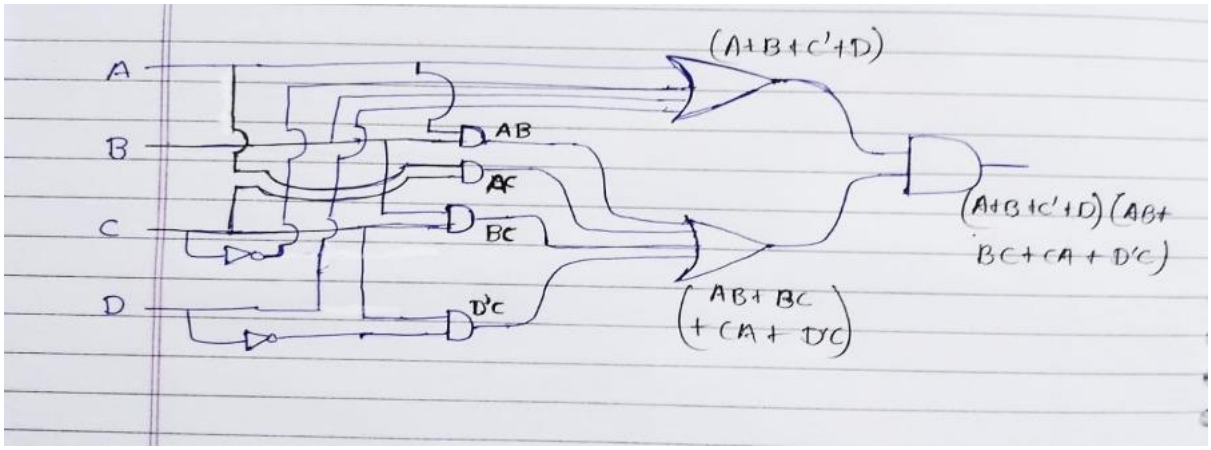
2.A



A	B	C	D	C'	D'	AB	BC	CA	D'C	(A+B+C'+D)	AB+CA / BC+D'C	(x)
0	0	0	0	1	1	0	0	0	0	1	0	0
0	0	0	1	1	0	0	0	0	0	1	0	0
0	0	1	0	0	1	0	0	0	1	0	1	0
0	0	1	1	0	0	0	0	0	0	1	0	0
0	1	0	0	1	1	0	0	0	0	1	0	0
0	1	0	1	1	0	0	0	0	0	1	0	0
0	1	1	0	0	1	0	1	0	1	1	1	1
0	1	1	1	0	0	0	0	0	0	1	1	1
1	0	0	0	1	1	0	0	0	0	1	0	0
1	0	0	1	1	0	0	0	0	0	1	0	0
1	0	1	0	0	1	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0	1	0	1	1	1
1	1	0	0	1	1	1	0	0	0	1	1	1
1	1	0	1	1	0	1	0	0	0	1	1	1
1	1	1	0	0	1	1	1	1	1	1	1	1
1	1	1	1	0	0	1	1	1	0	1	1	1

(x) = (A+B+C'+D) (AB + BC + CA + D'C)

2.B



2.C

Verilog

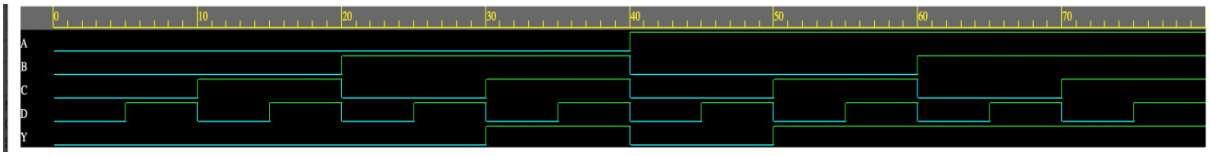
```
module Q2(
inputa,b,c,d,
    output y);
  assign y = ((a|b|~c|d) & ((a&b)|(b&c)|(c&a)|(~d&c)));
endmodule
```

TestBench

```
module tb_Q2;
    reg A,B,C,D;
    wire Y;
  Q2 a1 (.a(A) ,.b(B),.c(C),.d(D),.y(Y));
initial
    begin
A = 0; B =0; C=0; D=0; #5;
A = 0; B =0; C=0; D=1; #5;
A = 0; B =0; C=1; D=0; #5;
A = 0; B =0; C=1; D=1; #5;
A = 0; B =1; C=0; D=0; #5;
A = 0; B =1; C=0; D=1; #5;
A = 0; B =1; C=1; D=0; #5;
A = 0; B =1; C=1; D=1; #5;
A = 1; B =0; C=0; D=0; #5;
A = 1; B =0; C=0; D=1; #5;
A = 1; B =0; C=1; D=0; #5;
A = 1; B =0; C=1; D=1; #5;
A = 1; B =1; C=0; D=0; #5;
A = 1; B =1; C=0; D=1; #5;
A = 1; B =1; C=1; D=0; #5;
A = 1; B =1; C=1; D=1; #5;

    end
initial
```

```
  begin
    $dumpfile("dump.vcd");
    $dumpvars(1);
  end
endmodule
```

Waveform

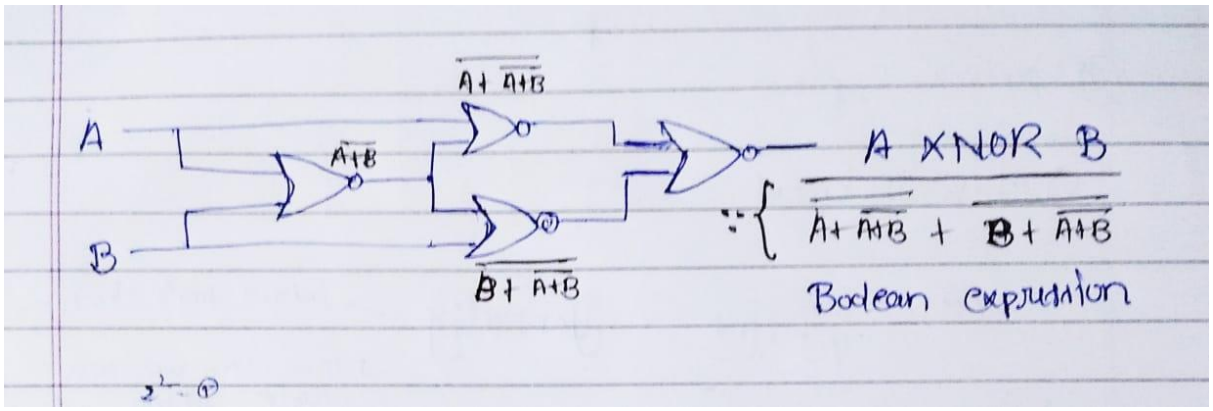


Comparison:

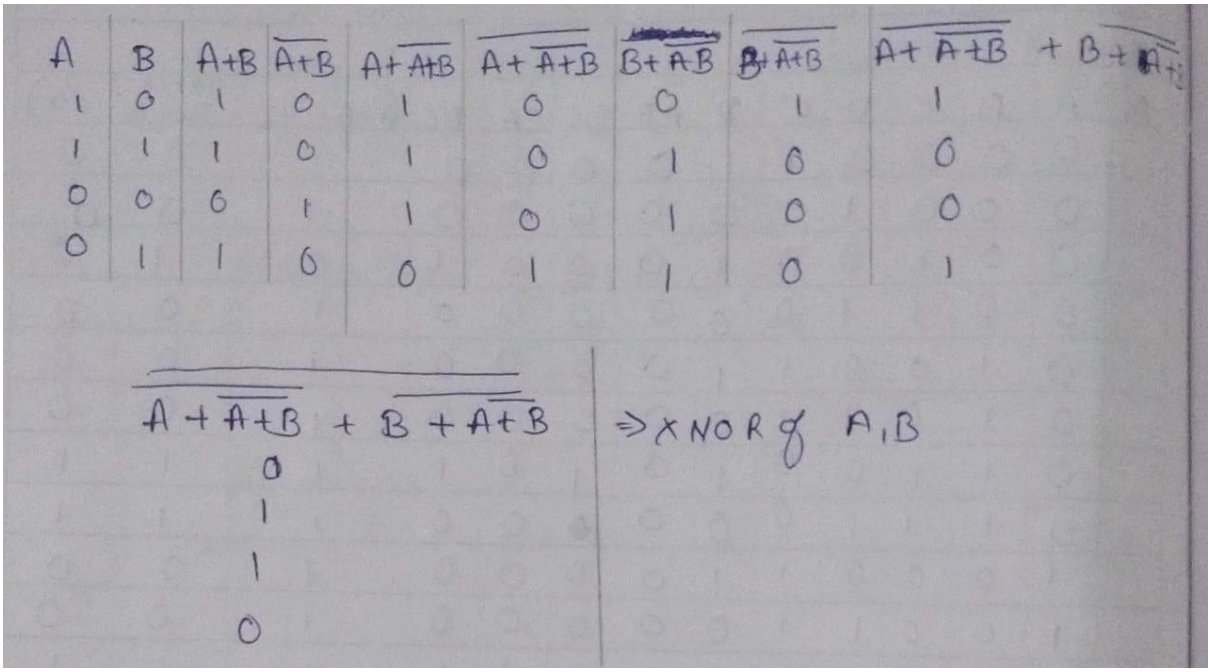
On comparison we got same result of waveform as our truth table output

3. **Representation of XNOR gate using only NOR gates and perform the following operations:**
    a. **Derive the Boolean expression.**
    b. **Write the truth table for the above expression**
    c. **Write a Verilog code for each Boolean expression and then test using wave form and compare with truth table whether your circuit produced same output or not?**

3A



3B



3C

Verilog

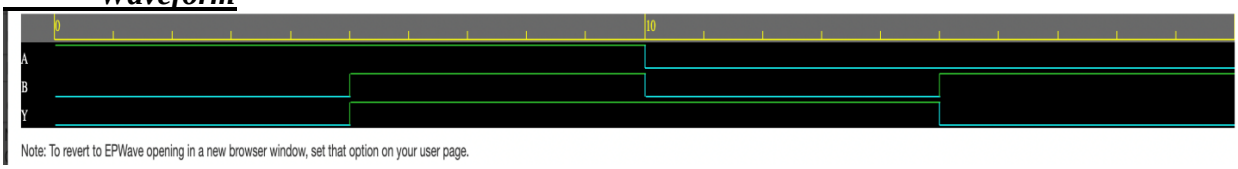
```
module Ans3(
inputa,b,
    output y);
  assign y = (~(~(a|~(a|b)) | ~(b|~(a|b))));
endmodule
```

TestBench

```
module Q3;
    reg A,B;
    wire Y;
  Ans3 a1 (.a(A) ,.b(B),.y(Y));
initial
    begin
A = 1; B =0; #5;
A = 1; B =1; #5;
A = 0; B =0; #5;
A = 0; B =1; #5;
    end
initial
  begin
    $dumpfile("dump.vcd");
    $dumpvars(1);
  end
```

```
endmodule
```

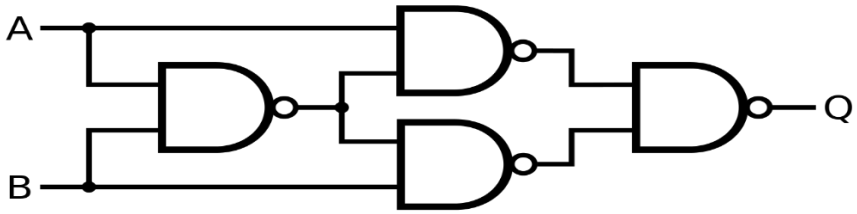
Waveform



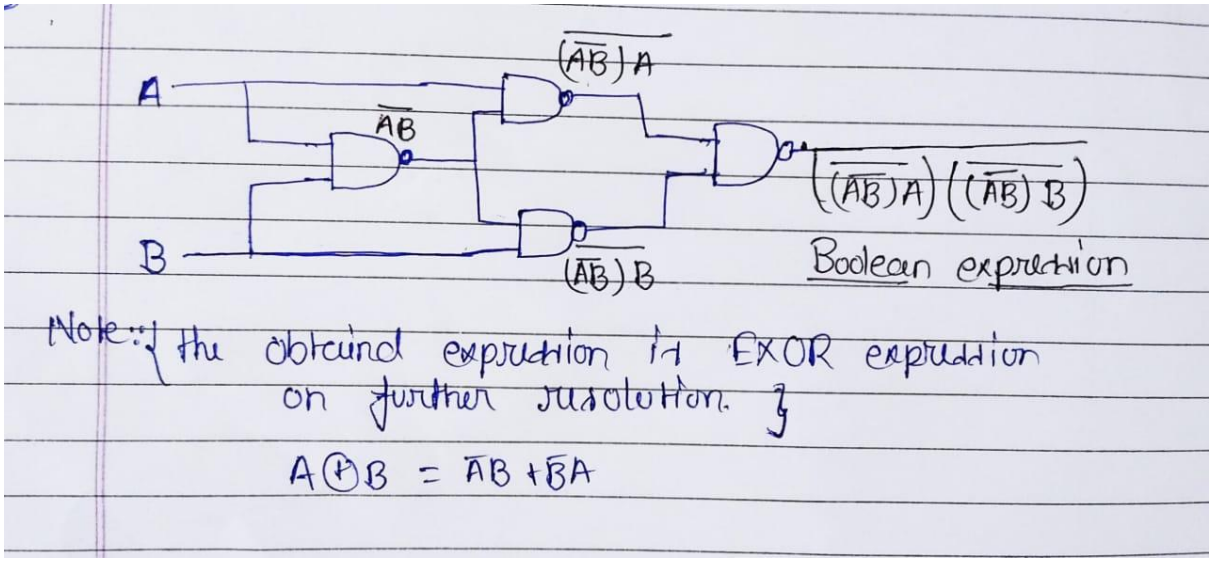
Comparison:

On comparison we got same result of waveform as our truth table output

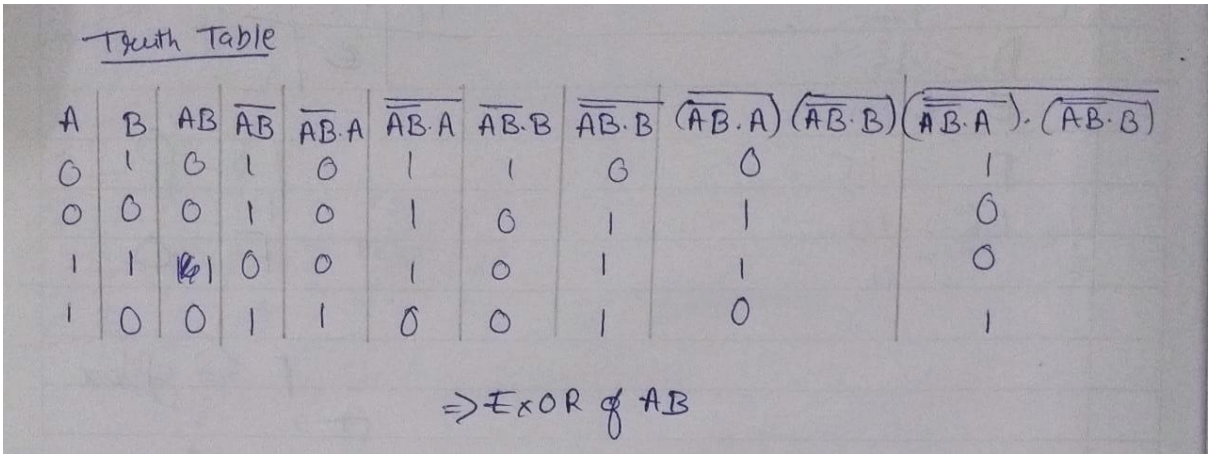
4. For the given circuit diagram do the following:
    a) Derive the Boolean expression.
    b) Write the truth table for the above expression
    c) Write a Verilog code for each Boolean expression and then test using wave form and compare with truth table whether your circuit produced same output or not?



4A



4B



4C

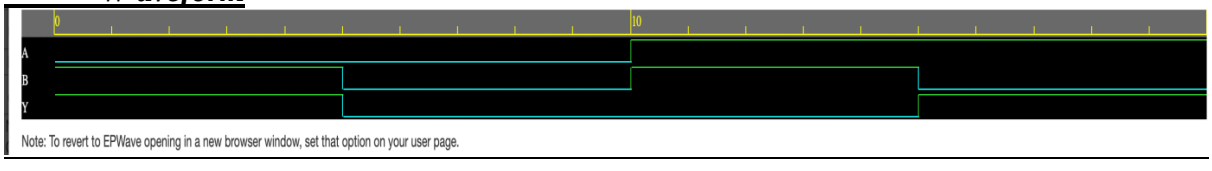
verilog

```
module Ans4(
inputa,b,
    output y);
  assign y = (~(~(a&~(a&b)) & ~(b&~(a&b))));
endmodule
```

TestBench

```
module Q4;
    reg A,B;
    wire Y;
  Ans4 a1 (.a(A) ,.b(B),.y(Y));
initial
    begin
A = 0; B =1; #5;
A = 0; B =0; #5;
A = 1; B =1; #5;
A = 1; B =0; #5;
    end
initial
  begin
    $dumpfile("dump.vcd");
    $dumpvars(1);
  end
endmodule
```

W aveform



Comparison:
On comparison we got same result of waveform as our truth table output

School of Computer Science Engineering and Technology

Course- BTech
Course Code: CSET 105
Year- 2022

Type- Core
Course Name- Digital Logic Design Lab
Semester- Even

Lab Assignment 4.1

1. You have given an assignment by your DLD teacher to buy and check the functionality of different gates (AND, OR, NOT, XOR, XNOR, and NOR), for that you went to a shop in Lajpat Rai Market, Chandani Chowk. You ask for all the above gates, but shop keeper says that he is out of these gates but having only NAND gate. Now one idea strikes in your mind that NAND is a universal gate, and you can make any gate using this one. Now complete your assignment by using NAND gate and perform the following task.
   A. Device the Boolean expression for the below mentioned gates using 2 inputs.
   B. Write the truth table for the above expression.
   C. Draw a semantic diagram using only NAND gates.
   D. Write down the Verilog code and match the derived expression with the truth table of the following gates.
      - NOT
      - AND
      - OR
      - NOR
      - XOR
      - XNOR
2. Write a Verilog code to verify Absorption Law and then test using wave form and compare with truth table whether your circuit produced same output or not?
3. Write a Verilog code to verify Transposition Law and then test using wave form and compare with truth table whether your circuit produced same output or not?
4. Write a Verilog code to verify Consensus Law and then test using wave form and compare with truth table whether your circuit produced same output or not?

Sameer Singh
E21CSEU0551 – EB22

Digital Design LAB – 4

Question1

You have given an assignment by your DLD teacher to buy and check the functionality of different gates (AND, OR, NOT, XOR, XNOR, and NOR), for that you went to a shop in Lajpat Rai Market, Chandani Chowk. You ask for all the above gates, but shop keeper says that he is out of these gates but having only NAND gate. Now one idea strikes in your mind that NAND is a universal gate, and you can make any gate using this one. Now complete your assignment by using NAND gate and perform the following task.

A. Device the Boolean expression for the below mentioned gates using 2 inputs.
B. Write the truth table for the above expression.
C. Draw a semantic diagram using only NAND gates.
D. Write down the Verilog code and match the derived expression with the truth table of the following gates.
   - NOT
   - AND
   - OR
   - NOR
   - XOR
   - XNOR

a) NOT GATE:

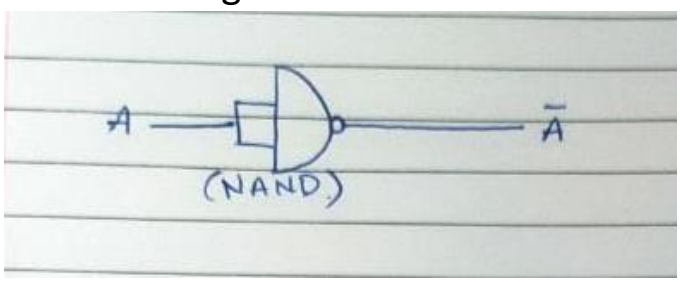
A. Boolean Expression:

~(A&A)

B. Truth Table

A	¬(A ∧ A)
T	F
F	T

C. Schematic Diagram:



D. Verilog Code and EP Wave:

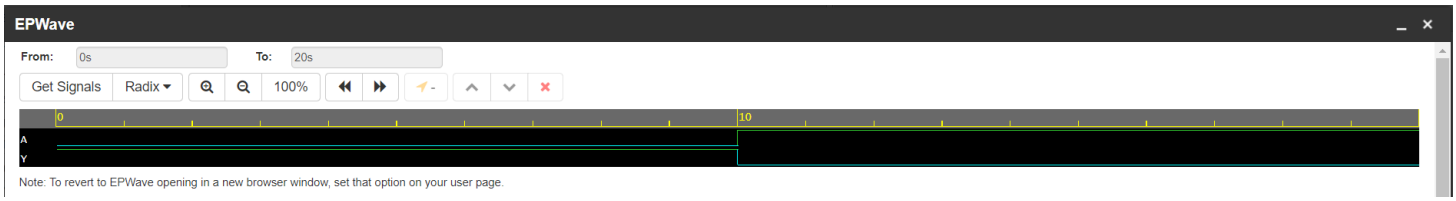
TestBench.sv

```
//SOUMYA DUBEY E21CSEU0760
//QUESTION 1 LAB4

module tb_and_gate;
 reg A;
 wire Y;

 and_gate a1(.a(A),.y(Y));
 initial
   begin
     A=0; #5;
     A=0; #5;
     A=1; #5;
     A=1; #5;
   end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end

endmodule
```

Digital.sv

```
//SOUMYA DUBEY E21CSEU0760
//QUESTION 1 LAB4

module and_gate (
 input a,
 output y);
 assign y= ~(a & a); //NOT Gate USING NAND Gate

   endmodule
```

EP Wave:



Explanation:

When Value of A is False, Output Y gives True and visa-versa.

b.) AND Gate:

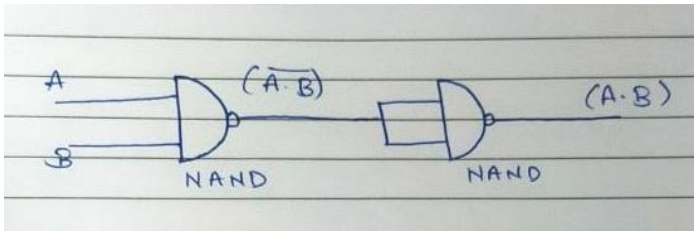
A. Boolean Expression:

~(~(A&B))

B. Truth Table:

A	B	¬¬(A ∧ B)
F	F	F
F	T	F
T	F	F
T	T	T

C. Schematic Diagram:



D. Verilog Code and EP Wave:

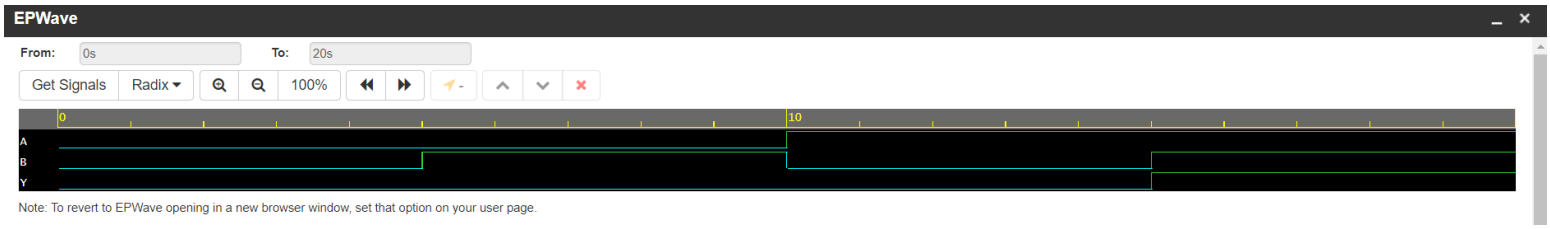
TestBench.sv

```
//SOUMYA DUBEY E21CSEU0760
//QUESTION 1 LAB4

module tb_and_gate;
 reg A, B;
 wire Y;

 and_gate a1(.a(A),.b(B),.y(Y));
 initial
   begin
     A=0; B=0;  #5;
     A=0; B=1;  #5;
     A=1; B=0;  #5;
     A=1; B=1;  #5;
   end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end

endmodule
```

Digital.sv

```
//SOUMYA DUBEY E21CSEU0760
//QUESTION 1 LAB4

module and_gate (
 input a,b,
 output y);
 assign y= ~(~(a&b)); //AND Gate using NAND Gate

  endmodule
```

EP Wave:



c.) OR Gate:

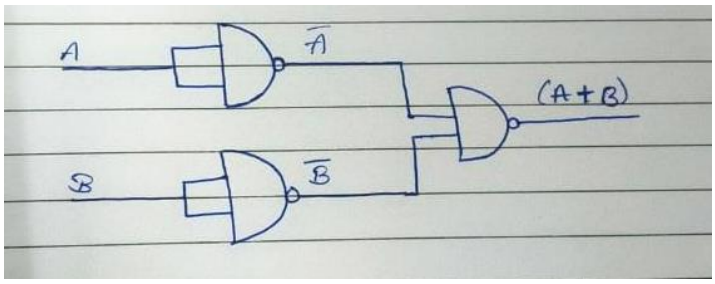
A. Boolean Expression:

~((~A)&(~B))

B. Truth Table:

A	B	¬(¬A ∧ ¬B)
F	F	F
F	T	T
T	F	T
T	T	T

C. Schematic Diagram:



D. Verilog Code and EP Wave:

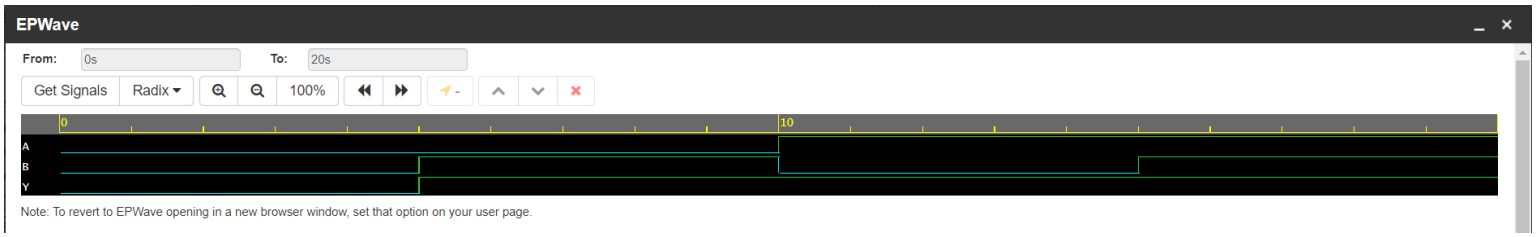
TestBench.sv

```
//SOUMYA DUBEY E21CSEU0760
//QUESTION 1 LAB4

module tb_and_gate;
 reg A, B;
 wire Y;

 and_gate a1(.a(A),.b(B),.y(Y));
 initial
   begin
     A=0; B=0;  #5;
     A=0; B=1;  #5;
     A=1; B=0;  #5;
     A=1; B=1;  #5;
   end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end

endmodule
```

Digital.sv

```
//SOUMYA DUBEY E21CSEU0760
//QUESTION 1 LAB4

module and_gate (
 input a,b,
 output y);
 assign y= ~((~a)&(~b)); //OR Gate using NAND Gate

  endmodule
```

EP Wave:



d.) NOR Gate:

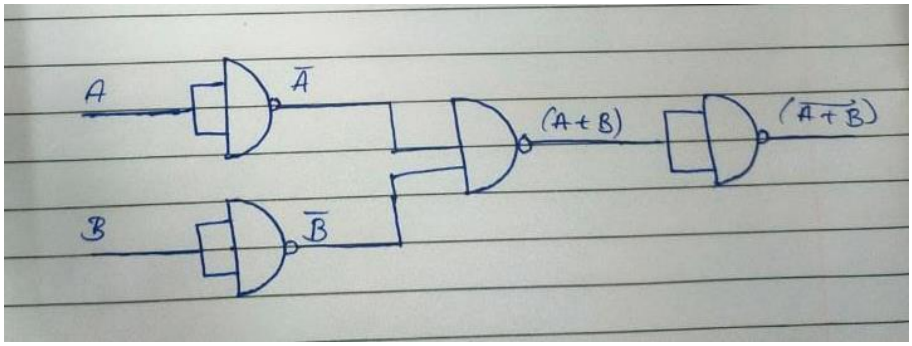
A. Boolean Expression:

~(~((~A)&(~B)))

B. Truth Table:

A	B	¬¬(¬A ∧ ¬B)
F	F	T
F	T	F
T	F	F
T	T	F

C. Schematic Diagram:



D. Verilog Code and EP Wave:

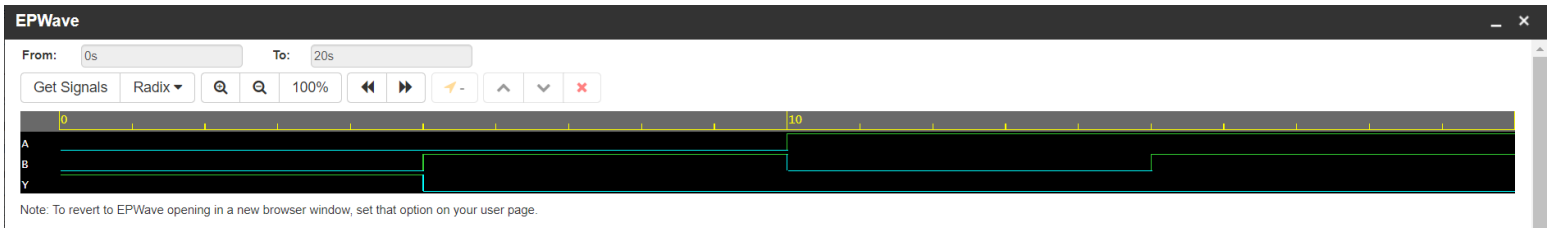
TestBench.sv

```
//SOUMYA DUBEY E21CSEU0760
//QUESTION 1 LAB4

module tb_and_gate;
 reg A, B;
 wire Y;

 and_gate a1(.a(A),.b(B),.y(Y));
 initial
   begin
     A=0; B=0;  #5;
     A=0; B=1;  #5;
     A=1; B=0;  #5;
     A=1; B=1;  #5;
   end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end

endmodule
```

Digital.sv

```
//SOUMYA DUBEY E21CSEU0760
//QUESTION 1 LAB4

module and_gate (
 input a,b,
 output y);
 assign y= ~(~((~a)&(~b))); //NOR Gate using NAND Gate

  endmodule
```

EP Wave:



E.) XOR Gate:

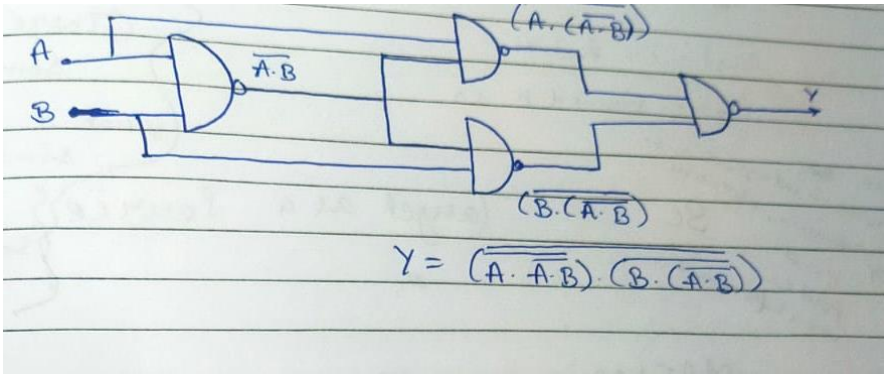
A. Boolean Expression:

~(~(A&~(A&B))&~(B&~(A&B)))

B. Truth Table:

A	B	¬(¬(A ∧ ¬(A ∧ B)) ∧ ¬(B ∧ ¬(A ∧ B)))
F	F	F
F	T	T
T	F	T
T	T	F

C. Schematic Diagram:



D. Verilog Code and EP Wave:

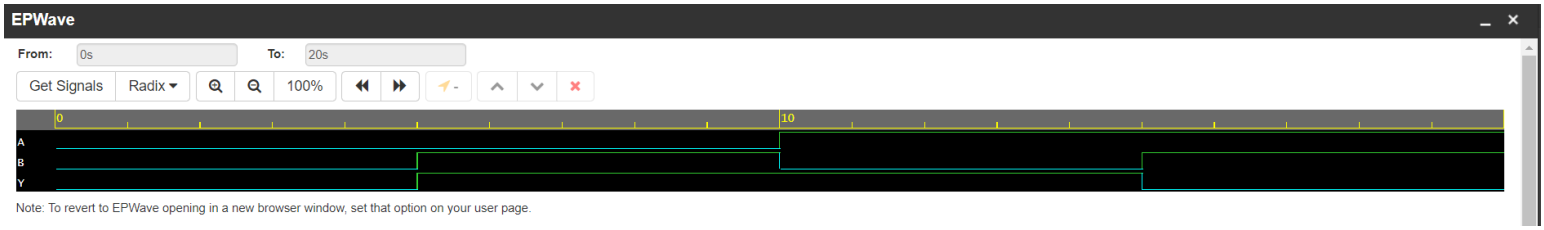
TestBench.sv

```
//SOUMYA DUEBY E21CSEU0760
//QUESTION 1 LAB4

module tb_and_gate;
 reg A, B;
 wire Y;

 and_gate a1(.a(A),.b(B),.y(Y));
 initial
   begin
     A=0; B=0;  #5;
     A=0; B=1;  #5;
     A=1; B=0;  #5;
     A=1; B=1;  #5;
   end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end

endmodule
```

Digital.sv

```
//SOUMYA DUBEY E21CSEU0760
//QUESTION 1 LAB4

module and_gate (
 input a,b,
 output y);
 assign y= ~(~(a&~(a&b))&~(b&~(a&b)))//XOR Gate using NAND Gate

  endmodule
```

EP Wave:



F.) XNOR Gate:

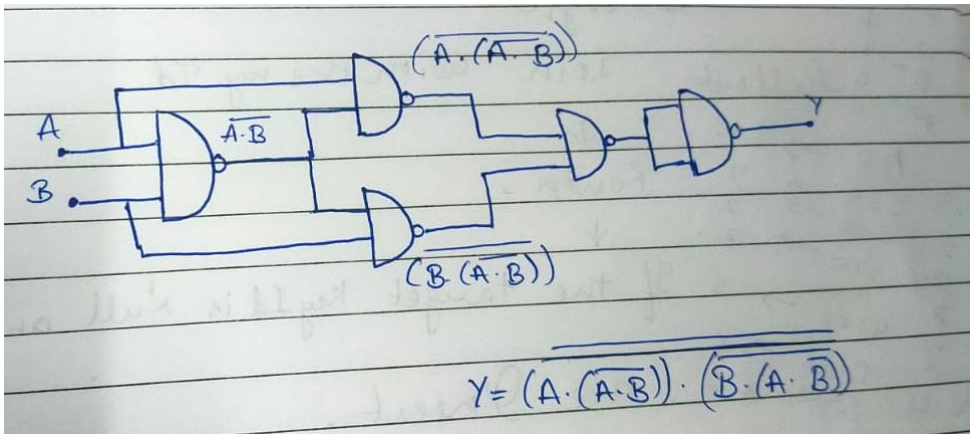
A. Boolean Expression:

~(~(~(A&~(A&B))&~(B&~(A&B))))

B. Truth Table:

A	B	¬¬(¬(A ∧ ¬(A ∧ B)) ∧ ¬(B ∧ ¬(A ∧ B)))
F	F	T
F	T	F
T	F	F
T	T	T

C. Schematic Diagram:



D. Verilog Code and EP Wave:

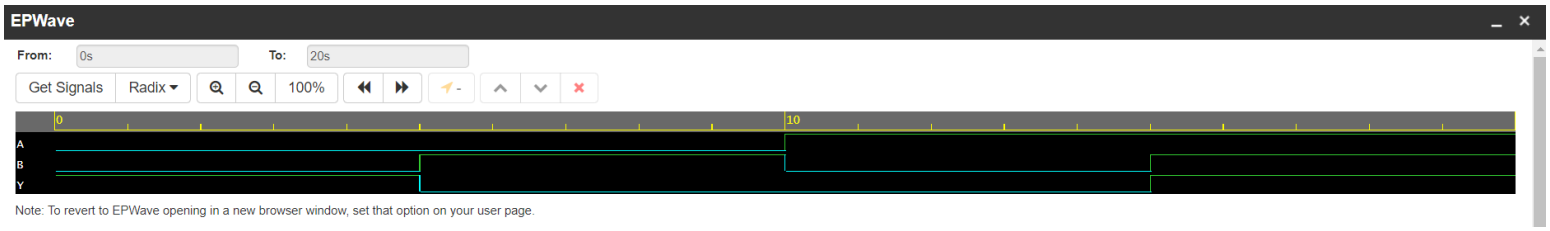
TestBench.sv

```
//SOUMYA DUEBY E21CSEU0760
//QUESTION 1 LAB4

module tb_and_gate;
 reg A, B;
 wire Y;

 and_gate a1(.a(A),.b(B),.y(Y));
 initial
   begin
     A=0; B=0;  #5;
     A=0; B=1;  #5;
     A=1; B=0;  #5;
     A=1; B=1;  #5;
   end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end

endmodule
```

Digital.sv

```
//SOUMYA DUEBY E21CSEU0760
//QUESTION 1 LAB4

module and_gate (
 input a,b,
 output y);
 assign y= ~(~(~(a&~(a&b))&~(b&~(a&b)))); //XNOR Gate using NAND Gate

  endmodule
```

EP Wave:



Question2

Write a Verilog code to verify Absorption Law and then test using wave form and compare with truth table whether your circuit produced same output or not?

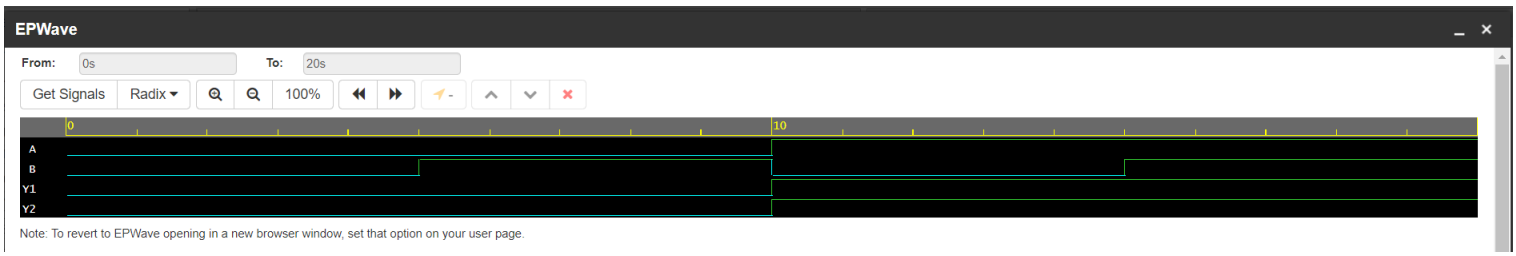
Design.sv

```
//SOUMYA DUBEY E21CSEU0760
//QUESTION 2 LAB4

module ABS(a,b,o1,o2);
 input a,b;
 output o1,o2;
 assign o1 = a;
 assign o2 = a|(a&b); //Proof of Absorption Law
endmodule
```

Testbench.sv

```
//SOUMYA DUEBY E21CSEU0760
//QUESTION 2 LAB4

module ABS1;
 reg A,B;
 wire Y1,Y2;

 ABS a1(.a(A),.b(B),.o1(Y1),.o2(Y2));
 initial
  begin
   A=1'b0; B=1'b0;  #5;
   A=1'b0; B=1'b1;  #5;
   A=1'b1; B=1'b0;  #5;
   A=1'b1; B=1'b1;  #5;
  end
 initial
  begin
   $dumpfile("dump.vcd");
   $dumpvars(1);
  end

endmodule
```

EP Wave



Truth Table

A	B	(A ∨ (A ∧ B))
F	F	F
F	T	F
T	F	T
T	T	T

The truth table matches with the Ep wave form.

Question3

Write a Verilog code to verify Transposition Law and then test using wave form and compare with truth table whether your circuit produced same output or not?

Design.sv

```
//SOUMYA DUEBY E21CSEU0760
//QUESTION 3 LAB4

module ABS(a,b,c,o1,o2);
 input a,b,c;
 output o1,o2;
 assign o1 = (a&b)|((~a)&c);//Transposition Law Proof
 assign o2 = (a|c)&((~a)|b);
endmodule
```

Testbench.sv

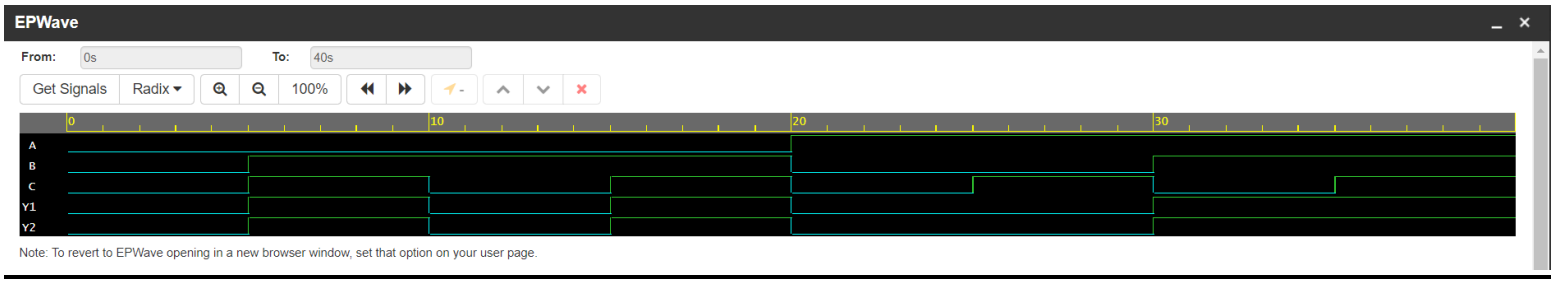
```
//SOUMYA DUEBY E21CSEU0760
//QUESTION 3 LAB4

module ABS1;
 reg A,B,C;
 wire Y1,Y2;

 ABS a1(.a(A),.b(B),.c(C),.o1(Y1),.o2(Y2));
 initial
   begin
     A=1'b0; B=1'b0; C=1'b0;  #5;
     A=1'b0; B=1'b1; C=1'b1;  #5;
     A=1'b0; B=1'b1; C=1'b0;  #5;
     A=1'b0; B=1'b1; C=1'b1;  #5;
     A=1'b1; B=1'b0; C=1'b0;  #5;
     A=1'b1; B=1'b0; C=1'b1;  #5;
     A=1'b1; B=1'b1; C=1'b0;  #5;
     A=1'b1; B=1'b1; C=1'b1;  #5;
   end
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end

endmodule
```

Truth Table

A	B	C	((A ∧ B) ∨ (¬A ∧ C))
F	F	F	F
F	F	T	T
F	T	F	F
F	T	T	T
T	F	F	F

T	F	T	F
T	T	F	T
T	T	T	T

EP Wave



The truth table matches with the EP wave form.

Question4

Write a Verilog code to verify Consensus Law and then test using wave form and compare with truth table whether your circuit produced same output or not?

Design.sv

```
//SOUMYA DUEBY E21CSEU0760
//QUESTION 4 LAB4

module ABS(a,b,c,o1,o2);
 input a,b,c;
 output o1,o2;
 assign o1 = (a&b)|((~a)&c);//Consensus Law Proof
 assign o2 = (a|c)&((~a)|b);
endmodule
```

Testbench.sv

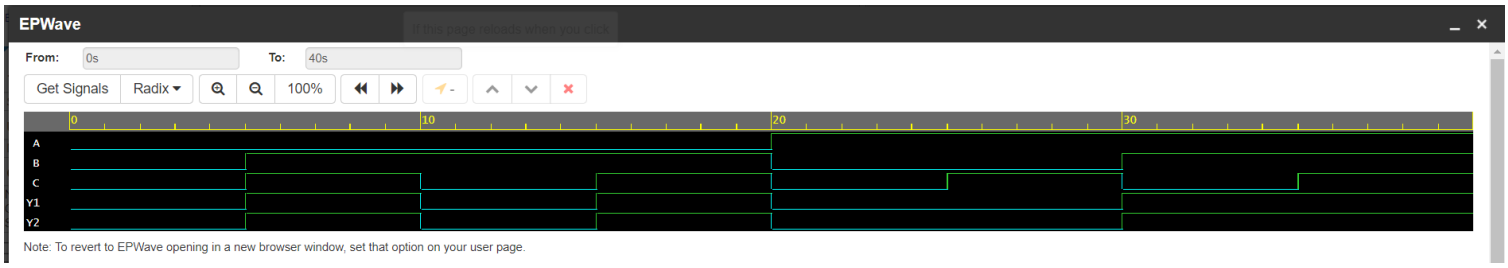
```
//SOUMYA DUEEBY E21CSEU0760
//QUESTION 4 LAB4

module ABS1;
 reg A,B,C;
 wire Y1,Y2;

 ABS a1(.a(A),.b(B),.c(C),.o1(Y1),.o2(Y2));
 initial
  begin
    A=1'b0; B=1'b0; C=1'b0;  #5;
    A=1'b0; B=1'b1; C=1'b1;  #5;
    A=1'b0; B=1'b1; C=1'b0;  #5;
    A=1'b0; B=1'b1; C=1'b1;  #5;
    A=1'b1; B=1'b0; C=1'b0;  #5;
    A=1'b1; B=1'b0; C=1'b1;  #5;
    A=1'b1; B=1'b1; C=1'b0;  #5;
    A=1'b1; B=1'b1; C=1'b1;  #5;
  end
 initial
```

```
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end

endmodule
```

Truth Table

A	B	C	((A ∧ B) ∨ ((¬A ∧ C) ∨ (B ∧ C)))
F	F	F	F
F	F	T	T
F	T	F	F
F	T	T	T
T	F	F	F
T	F	T	F
T	T	F	T
T	T	T	T

EP Wave



The truth table matches with the EP wave form.

School of Computer Science Engineering and Technology

Course- BTech
Course Code- CSET105
Year- 2022
Date-

Type- Core
Course Name- Digital Design
Semester- Even
Batch-

5.1 Lab Assignment # 5

In this lab, we shall learn about the procedures in Verilog, generation of modules from a truth table. We also further explore the "Module Instantiation", where a previously designed module will be used other modules.

Q1. Consider the following expressions. Here A, B, and C are acting as input.

X = A.B'.C + A'.B.C'
Y = A'.B' + B.C'
Z = X'.Y + Y'.X

i. Generate the truth table for each expression.
ii. Write down the modules for X and Y. use the instances of X and Y to implement the module for Z.
iii. Verify the same with the corresponding Testbench code.

Q2. Design a Verilog code for BCD to excess-3 code. Using behavioral design then test with test bench code.

a. Prepare truth table.
b. Identify Boolean function for each output using K-maps.
c. Write Verilog code for implement the module.
d. Check with test bench.

Q3. A half-adder is used to add two single bit inputs. It produces a single bit output and a possible carry bit. Below is the truth table for the same.

Input		**Output**	
A	B	Carry	Output
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(i) Develop the Boolean expression and logic circuit for the given truth table.
(ii) Write a Verilog module for Universal NOR gate. Utilize the instances of NOR gate only to write a structural Verilog code for Half adder.
(iii) Verify it with respective testbench code.

School of Computer Science Engineering and Technology

Q.4 The half adder in the previous question can only add two one-bit numbers when there is no carry bit, which is not sufficient in many cases. While a full-adder has two one-bit inputs, a carry-in input, a sum output, and a carry-out output. Below truth table represents a full adder.

Full Adder Truth Table				
Input			**Output**	
A	B	Cin	Cout	Sum
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

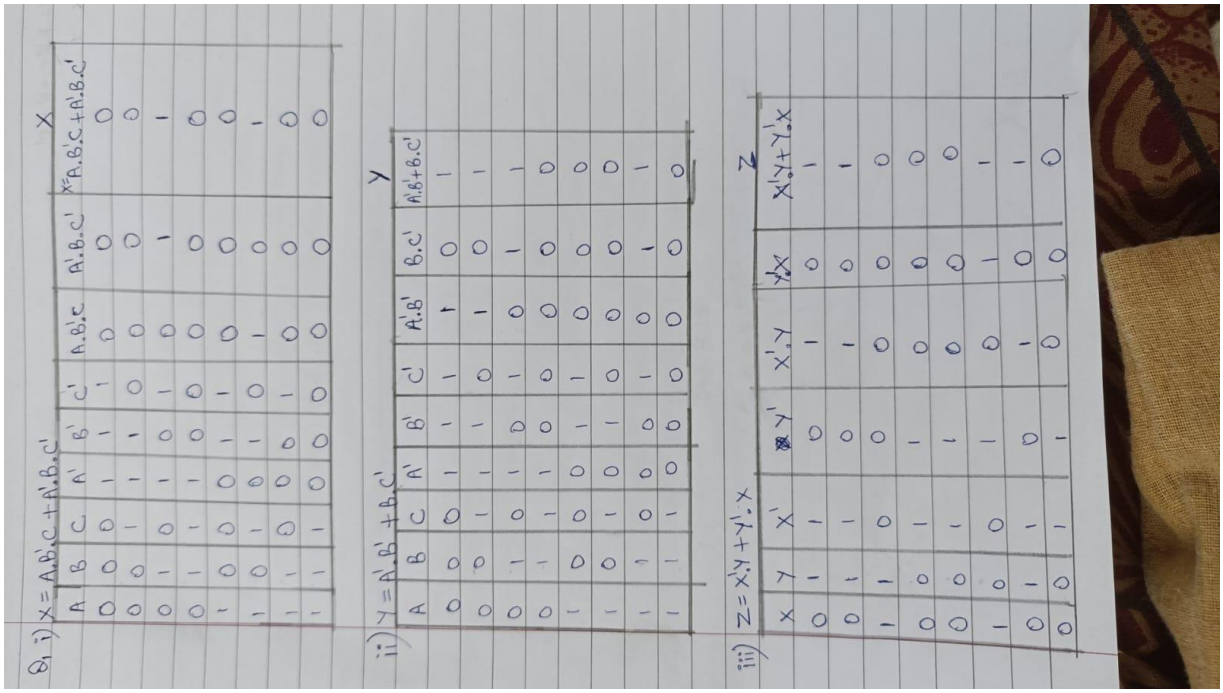
(i) Utilize the above truth table to design the Boolean expression and digital circuit for the full adder.
(ii) Write the behavioral Verilog code for the full adder.
(iii) Verify it with respective Testbench code.

Submission Instructions:

- Prepare the submission file according to the following process:
    1. Copy the Verilog code, the Test Bench Code in a Word File.
    2. Take the ScreenShot of Waveform and paste into the same word file.
    3. Repeat Step 1 and 2 for all the programs.
    4. Copy and Paste all the Verilog code, Testbench Code and Waveform into a single word file as 1_verilog, 1_TestBench, 1_Waveform, 2_verilog, 2_TestBench, 2_Waveform… etc.
    5. Convert it into pdf file, name it as **RollNo_Assignment# (Example: E20CSE001_ Assignment3.pdf)**.
    6. Submit your file on LMS **within the deadline**.
- Write your **Name and Roll No. as comment before starting of each program**. Keep in mind this is **Mandatory**. Failing which you may lose your marks.
- Make it sure that in each program, **you have mentioned enough comments** regarding the explanation of program instructions.
- **Each student will submit their assignment on their corresponding group slot only.**
- Late submission will lead to penalty.
- Any form of plagiarism/copying from peer or internet sources will lead penalty.
- Following of all instructions at submission time is mandatory. Missing of any instructions at submission time will lead penalty.

SOUMYA DUBEY
E21CSEU0760
EB27

Q1. I)TRUTH –TABLE



II) VERILOG CODE

```
//E21CSEU0760
//SOUMYA DUBEY
module
  x1(a,b,c,x);
  input a,b,c;
  output x;
  assign
x=(a&~b&c)|(~a&b&~c);
endmodule


module
  y1(a,b,c,y); input
  a,b,c;
```

```
 output y;
 assign
y=(~a&b)|(b&~c);
endmodule

module
 z1(a,b,c,x,y,z);
 input a,b,c;
 output x,y,z;
 assign
x=(a&~b&c)|(~a&b&~c);
assign  y=(~a&~b)|(b&~c);
assign z=(~x&y)|(~y&x);
endmodule
```

TESTBENCH

```
//E21CSEU0760
//SOUMYA DUBEY
module
 tb_z1(); reg
 A,B,C;
 wire X,Y,Z;
 z1 a1(.a(A),.b(B),.c(C),.x(X),.y(Y),.z(Z));
      initial begin
       A=0;B=0;C=0;
       #5;
       A=0;B=0;C=1;
       #5;
       A=0;B=1;C=0;
       #5;
       A=0;B=1;C=1;
```

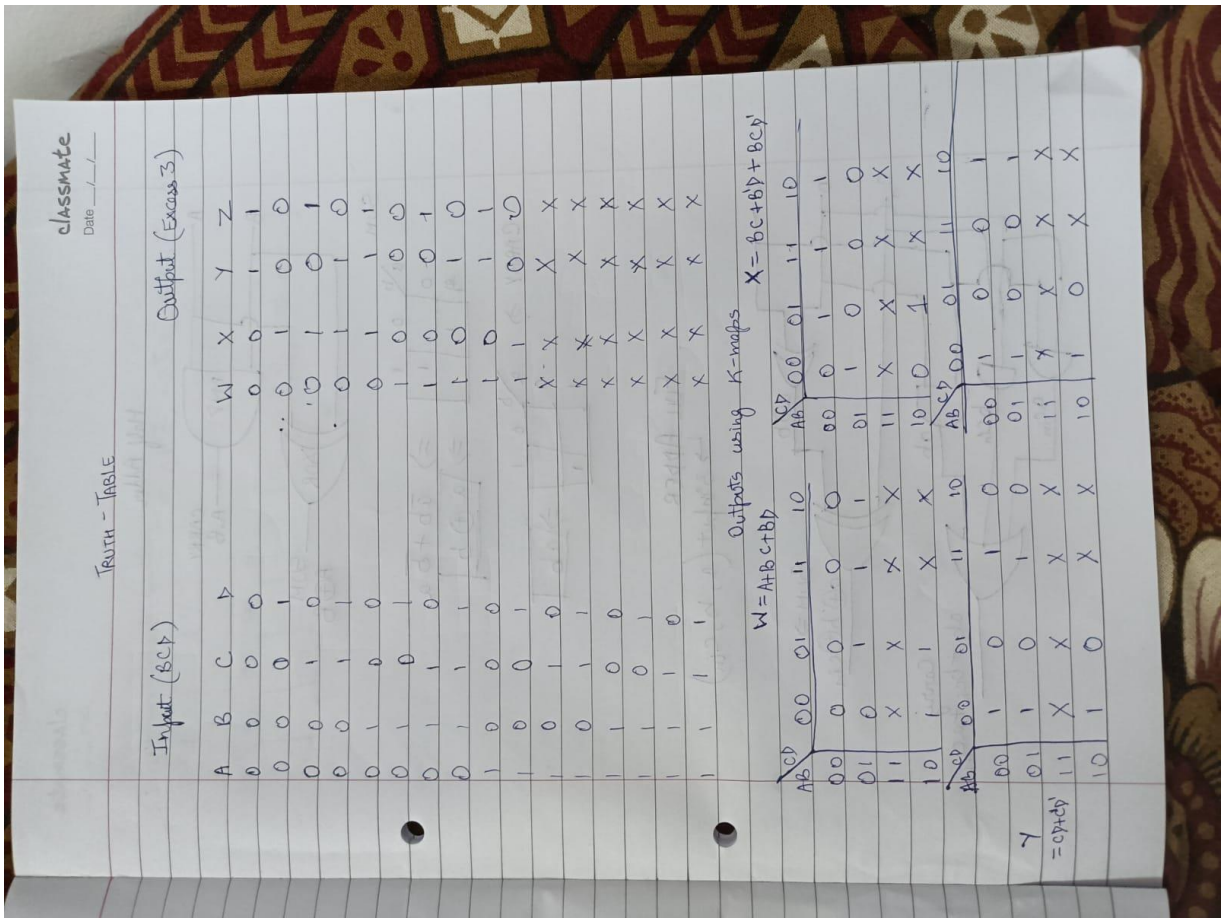
```
#5;
A=1;B=0;C=0;
#5;
A=1;B=1;C=0;
#5;
A=1;B=0;C=1;
#5;
```

```
  A=1;B=1;C=1;#5;
end
initial
begin
    $dumpfile("dump.vcd");
    $dumpvars(1
  ); end
endmodule
```

Waveform

Note: To revert to EPWave opening in a new browser window, set that option on your user page.

Q2.



Verilog code

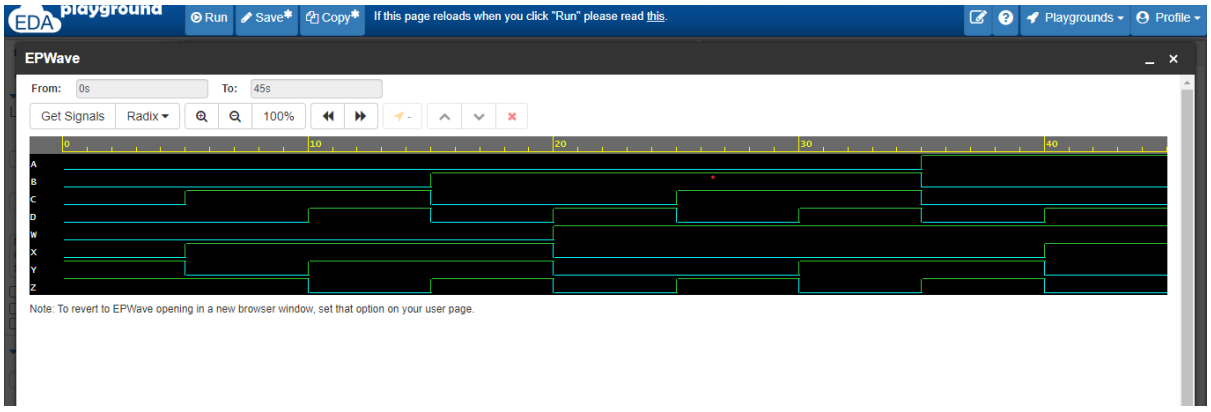
```
//E21CSEU0760
//SOUMYA DUBEY
module bcex3(input a, b, c, d,output w,
x,y,z); assign w = (a|(b & c)|(b & d));
assign x = (((~b) & c) | ((~b) & d) | (b & (~c) &
(~d))); assign y = ((c & d) | ((~c) & (~d)));
 assign z =
~d;
endmodule
```

testbench

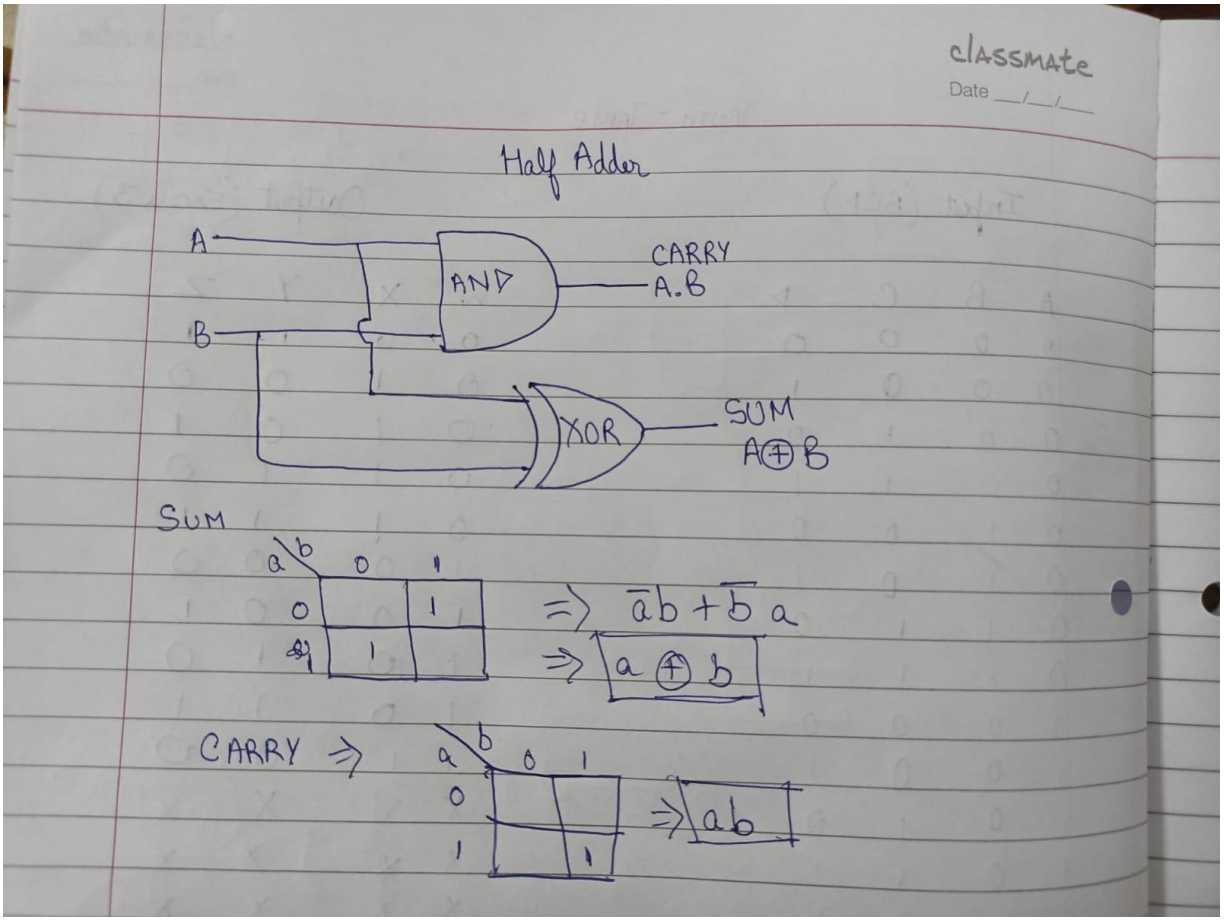
```
//E21CSEU0760
//SOUMYA DUBEY
module tb;
```

```
reg A,B,C,D;
 wire W,X,Y,Z;
 bcex3 x1 (.a(A) ,.b(B),.c(C),.d(D),.w(W),.x(X),.y(Y),.z(Z));
 initial begin
A = 0; B =0; C=0; D=0; #5;
 A = 0; B =0; C=1; D=0; #5;

A = 0; B =0; C=1; D=1; #5;
 A = 0; B =1; C=0; D=0; #5;
 A = 0; B =1; C=0; D=1; #5;
 A = 0; B =1; C=1; D=0; #5;
 A = 0; B =1; C=1; D=1; #5;
A = 1; B =0; C=0; D=0; #5;
 A = 1; B =0; C=0; D=1; #5;
 end
initial begin
$dumpfile("dump.vcd");
$dumpvars(1
); end
endmodule
```

Waveform



Q3.



Verilog code

```
  //E21CSEU0760
  //SOUMYA DUBEY
module norgate(input p,q,output
  c); assign c=~(p|q);
endmodule
module halfadder(input A,B,output
  sum,car); norgate n1
  (.p(A),.q(A),.c(~A));
norgate n2 (.p(B),.q(B),.c(~B));
```

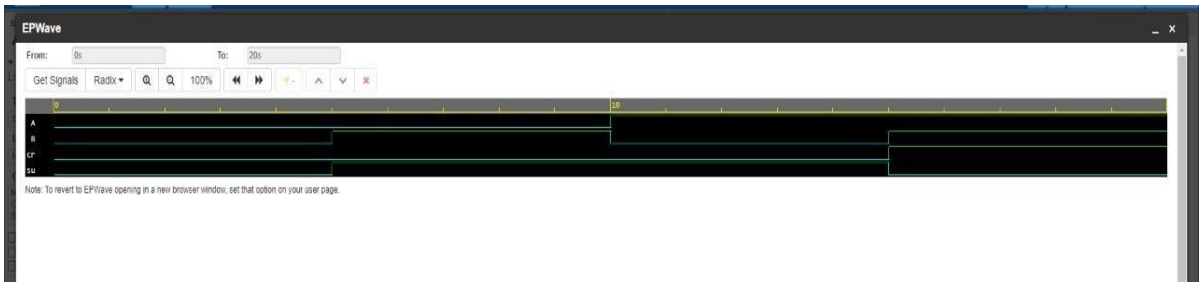
```
        norgate n3 (.p(~A),.q(~B),.c(car));
         norgate n4 (.p(A),.q(B),.c(~(A|B)));
        norgate n5
                (.p(A),.q(B),.c(sum));
                endmodule
```

testbench

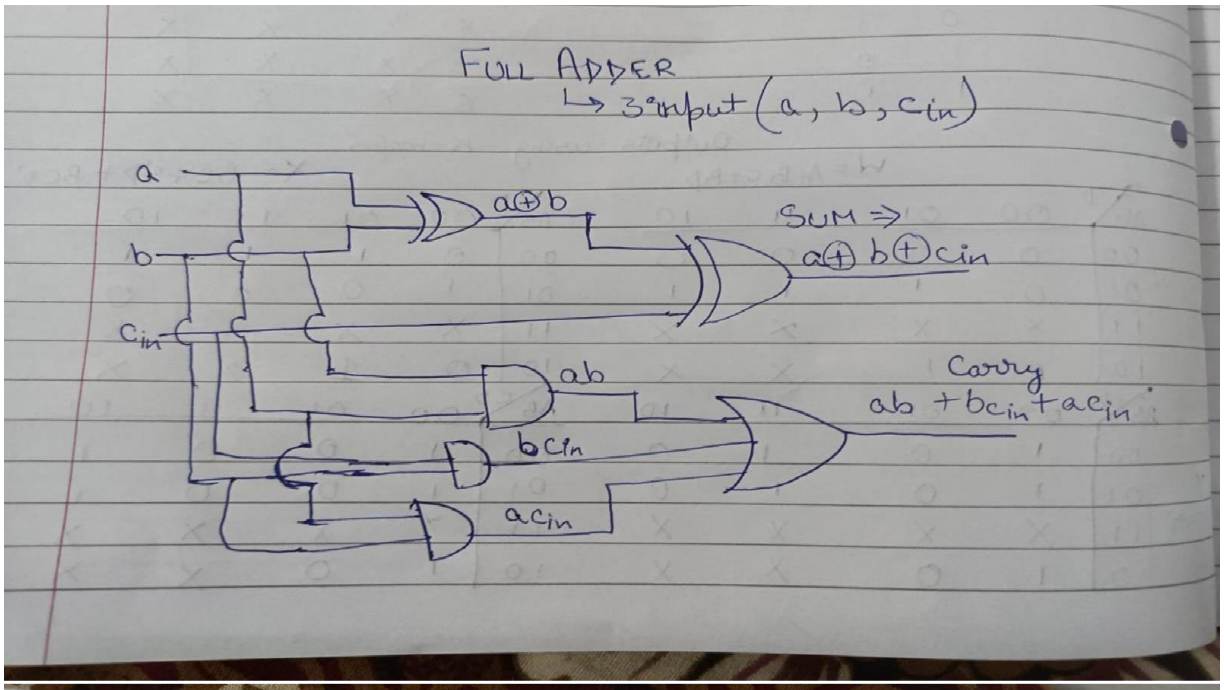
```
//E21CSEU0760
//SOUMYA DUBEY
module
         tb_halfadder;
         reg A,B;
         wire su,cr;
         halfadder
        ha(.a(A),.b(B),.su(sum),.car(car));   initial
        begin
         A=0;B=0;#
         5;
         A=0;B=1;#
         5;
         A=1;B=0;#
         5;
         A=1;B=1;#
         5;

        end
         initial
         begi
         n
          $dumpfile("dump.vcd");
          $dumpvars(1
```

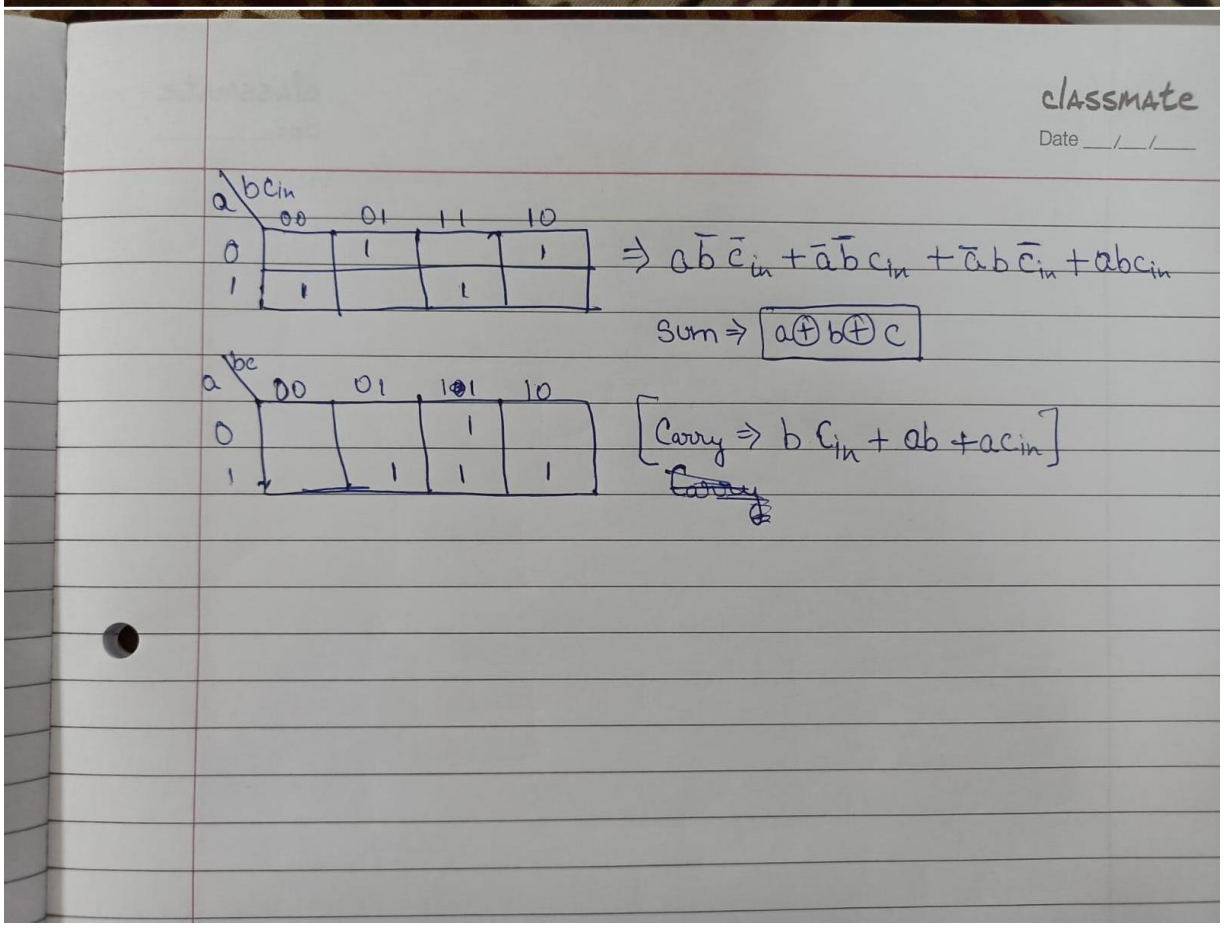
```
 ); end
endmodule
```

WaveForm


EPWave
From:
0s
To:
20s
Get Signals
Radix
100%
0
10
A
B
CI
SU
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

Q4.





VERILOG CODE

```
// Code your design here
module fsub(input p,q,r,output s,c);
```

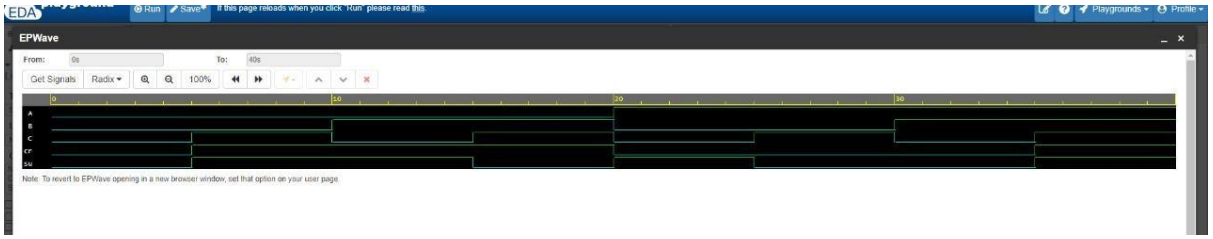
```
 assign s=(p^q)^r;
 assign
c=(r&~(p^q))|(~p&q);
endmodule
```

TESTBENCH

```
module
 tb_fsub; reg
 A,B,C;
 wire su,cr;
 fsub
x(.p(A),.q(B),.r(C),.s(su),.c(cr));
initial begin
 A=0;B=0;C=0;
 #5;
 A=0;B=0;C=1;
 #5;
 A=0;B=1;C=0;
 #5;
 A=0;B=1;C=1;
 #5;
 A=1;B=0;C=0;
 #5;
 A=1;B=0;C=1;
 #5;
 A=1;B=1;C=0;
 #5;
 A=1;B=1;C=1;
 #5;
end
 initial
```

```
 begi
 n
  $dumpfile("dump.vcd");
  $dumpvars(1
 ); end
endmodule
```

WAVEFORM


EDA
Run
Save
If this page reloads when you click "Run", please read this
Playgrounds
Profile
EPWave
From:
0s
To:
40s
Get Signals
Radix
100%
0
10
20
30
A
B
C
Cr
Sw
Note: To revert to EPWave opening in a new browser window, set that option on your user page

School of Computer Science Engineering and Technology

Course- BTech
Course Code- CSET105
Year- 2022
Date-

Type- Core
Course Name- Digital Design
Semester- Even
Batch-

Lab Assignment # 6

In this lab, we shall learn more types of Combinational circuits, how they work and about the procedures in Verilog

Q1. A Parallel adder can be used to find arithmetic sum of two n-bit binary numbers. n-bit parallel adder requires n-bit full adders. Based on this information, design a 4 -bit parallel adder in Verilog and perform the following operations:

a) Write Verilog code for implement the module.
b) Check with test bench.

Q2. A Parallel subtractor can be used to find arithmetic difference of two n -bit binary numbers. N-bit parallel subtractor can be designed in two different ways:

1. Using n-bit full adder and one complemented input.
2. Using n-bit full subtractors

Design a 4-bit parallel subtractor in Verilog based on above mentioned both designing ways and perform the following operations:

a) Write Verilog code for implement the module.
b) Check with test bench.

Submission Instructions:

- Prepare the submission file according to the following process:
    1. Copy the Verilog code, the Test Bench Code in a Word File.
    2. Take the ScreenShot of Waveform and paste into the same word file.
    3. Repeat Step 1 and 2 for all the programs.
    4. Copy and Paste all the Verilog code, Testbench Code and Waveform into a single word file as 1_verilog, 1_TestBench, 1_Waveform, 2_verilog, 2_TestBench, 2_Waveform… etc.
    5. Convert it into pdf file, name it as **RollNo_Assignment# (Example: E20CSE001_ Assignment3.pdf)**.
    6. Submit your file on LMS **within the deadline**.
- Write your **Name and Roll No. as comment before starting of each program**. Keep in mind this is **Mandatory**. Failing which you may lose your marks.
- Make it sure that in each program, **you have mentioned enough comments** regarding the explanation of program instructions.
- **Each student will submit their assignment on their corresponding group slot only.**
- Late submission will lead to penalty.
- Any form of plagiarism/copying from peer or internet sources will lead penalty.
- Following of all instructions at submission time is mandatory. Missing of any instructions at submission time will lead penalty.

School of Computer Science Engineering and Technology

Course- BTech
Course Code- CSET105
Year- 2022

Type- Core
Course name- Digital Design
Semester- Even
Batch- EB27

SOUMYA DUBEY

E21CSEU0760

Lab Assignment # 6

Q1. A Parallel adder can be used to find arithmetic sum of two n-bit binary numbers. n-bit parallel adder requires n-bit full adders. Based on this information, design a 4 -bit parallel adder in Verilog and perform the following operations:

a) **Write Verilog code for implement the module.**
b) **Check with test bench.**

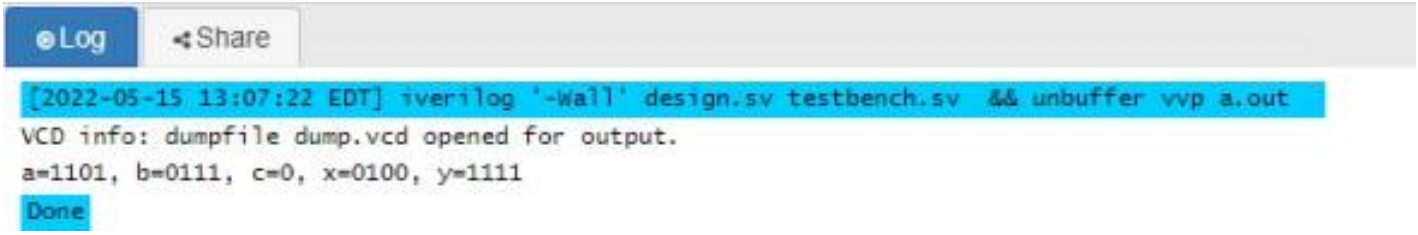
1_verilog:

```
//E21CSEU0760
//Soumya dubey

module parallel_adder(a, b, c, s, o);
  input [3:0] a;
  input [3:0] b;
  input c;
  output [3:0] s;
  output [3:0] o;
  full_adder f1(a[0], b[0], c, s[0], o[0]);
  full_adder f2(a[1], b[1], o[0], s[1], o[1]);
  full_adder f3(a[2], b[2], o[1], s[2], o[2]);
  full_adder f4(a[3], b[3], o[2], s[3], o[3]);
endmodule
module full_adder(a, b, c, x, y);
  input a, b, c;
  output x, y;
  assign x = ((~a&~b&c)|(a&b&c)|(a&~b&~c)|(~a&b&~c));
  assign y = (a&c)|(b&c)|(a&b);
endmodule
```

1_testbench:

```
//E21CSEU0760
//Soumya dubey

testbench();
reg [3:0]a1, b1;
reg c;
wire [3:0]y1, y2;
```

```
parallel_adder pa_ds(a1, b1, c, y1, y2);
initial begin
 a1 = 4'b1101;
 b1 = 4'b0111;
 c = 1'b0;
 #1
 $display("a=%b, b=%b, c=%b, x=%b, y=%b", a1, b1, c, y1,y2);
end
initial begin
 $dumpfile("dump.vcd");
 $dumpvars;
end
endmodule
```



Q2. A Parallel subtractor can be used to find arithmetic difference of two n -bit binary numbers. N-bit parallel subtractor can be designed in two different ways:

1. **Using n-bit full adder and one complemented input.**
2. **Using n-bit full subtractors**

Design a 4-bit parallel subtractor in Verilog based on above mentioned both designing ways and perform the following operations:

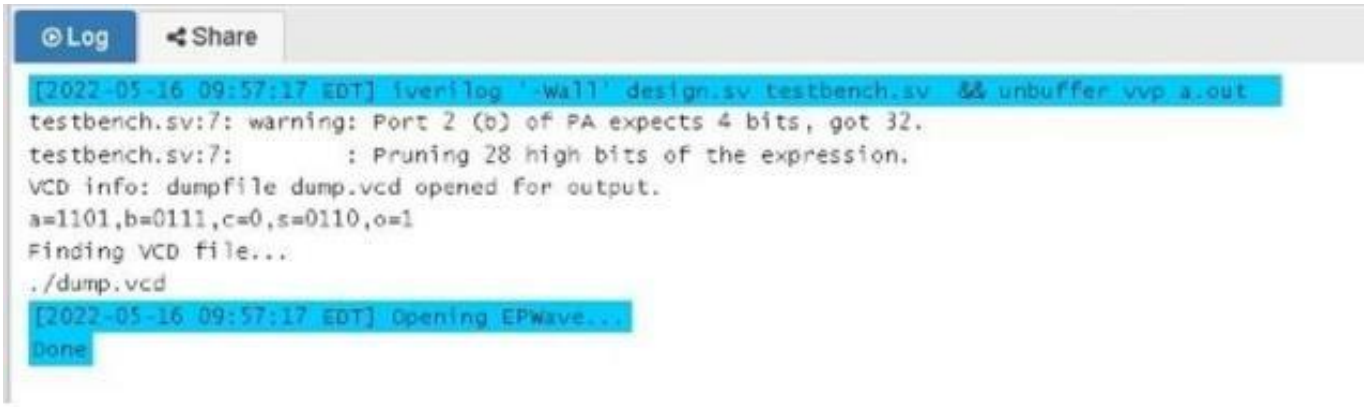
a) **Write Verilog code for implement the module.**
b) **Check with test bench.**

2a_verilog:

```
//E21CSEU0760
//Soumya dubey

module PA(a,b,c,s,o);
 input [3:0] a;
 input [3:0] b;
 input c;
 output [3:0] s;
 output [3:0] o;
 FA f1(a[0],b[0],c,s[0],o[0]);
 FA f2(a[1],b[1],o[0],s[1],o[1]);
 FA f3(a[2],b[2],o[1],s[2],o[2]);
 FA f4(a[3],b[3],o[2],s[3],o[3]);
endmodule
module FA(A,B,C,S,O);
 input A,B,C;
 output S,O;
 assign S = ((~A&~B&C)|(A&B&C)|(A&~B&~C)|(~A&B&~C));
 assign O = (A&C)|(B&C)|(A&B);
endmodule
```

2a_testbench:

```
//E21CSEU0760
//Soumya dubey

module tb_gate;
  reg [3:0]a,b;
  reg c;
  wire [3:0]s,o;
  PA padd(a,~b+1,c,s,o);
  initial begin
    a = 4'b1101;b=4'b0111;c=1'b0;
    #1
    $display("a=%b,b=%b,c=%b,s=%b,o=%b",a,b,c,s,o[3]);
    end
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars();
    end
endmodule
```



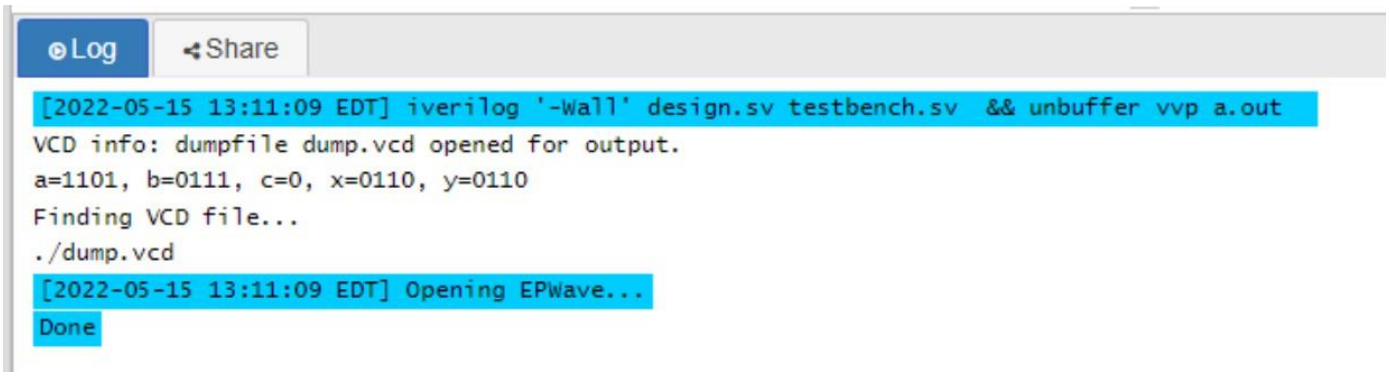
2b_verilog:

```
//E21CSEU0760
//Soumya dubey

module parallel_substracter(a, b, c, s, o);
  input [3:0] a;
  input [3:0] b;
  input c;
  output [3:0] s;
  output [3:0] o;
  full_adder f1(a[0], b[0], c, s[0], o[0]);
  full_adder f2(a[1], b[1], o[0], s[1], o[1]);
  full_adder f3(a[2], b[2], o[1], s[2], o[2]);
  full_adder f4(a[3], b[3], o[2], s[3], o[3]);
  endmodule
module full_adder(a, b, c, x, y);
  input a, b, c;
  output x, y;
```

```
  assign x = ((~a&~b&c)|(a&b&c)|(a&~b&~c)|(~a&b&~c));
  assign y = (~a&c)|(b&c)|(~a&b);
endmodule
```

2b_testbench:

```
//E21CSEU0760
//Soumya dubey

testbench();
reg [3:0]a1, b1;
reg c;
wire [3:0]y1, y2;
parallel_substracter pa_ds(a1, b1, c, y1, y2);
initial begin
  a1 = 4'b1101;
  b1 = 4'b0111;
  c = 1'b0;
  #1
  $display("a=%b, b=%b, c=%b, x=%b, y=%b", a1, b1, c, y1,y2);
  end
initial begin
  $dumpfile("dump.vcd");
  $dumpvars;
end
endmodule
```



Assignment 7

Multiplexing is the process of combining one or more signals and transmitting on a single channel. In analog communication systems, a communication channel is a scarce quantity, which must be properly used. For cost-effective and efficient use of a channel, the concept of Multiplexing is very useful as it allows multiple users to share a single channel in a logical way.

The three common types of Multiplexing approaches are:

- Time
- Frequency
- Space

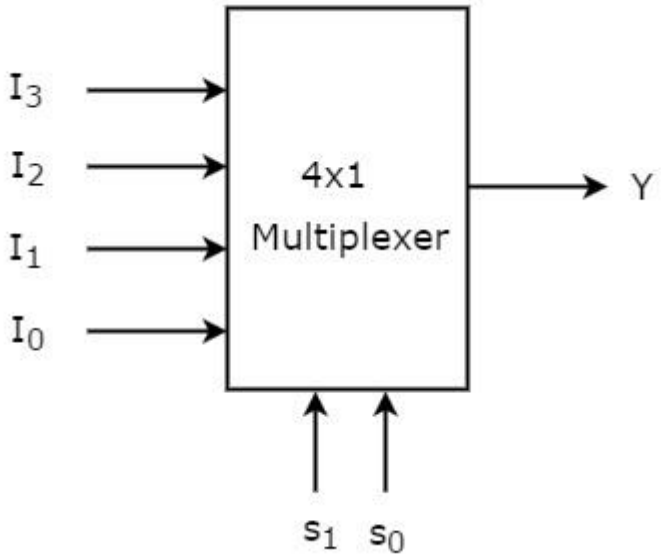
Eg: landline telephone network and the Cable TV.

Multiplexer is a combinational circuit that has maximum of 2^n data inputs, 'n' selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.

Since there are 'n' selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination will select only one data input.

4x1 Multiplexer

4x1 Multiplexer has four data inputs I_3, I_2, I_1 & I_0, two selection lines s_1 & s_0 and one output Y. The **block diagram** of 4x1 Multiplexer is shown in the following figure.



One of these 4 inputs will be connected to the output based on the combination of inputs present at these two selection lines. **Truth table** of 4x1 Multiplexer is shown below.

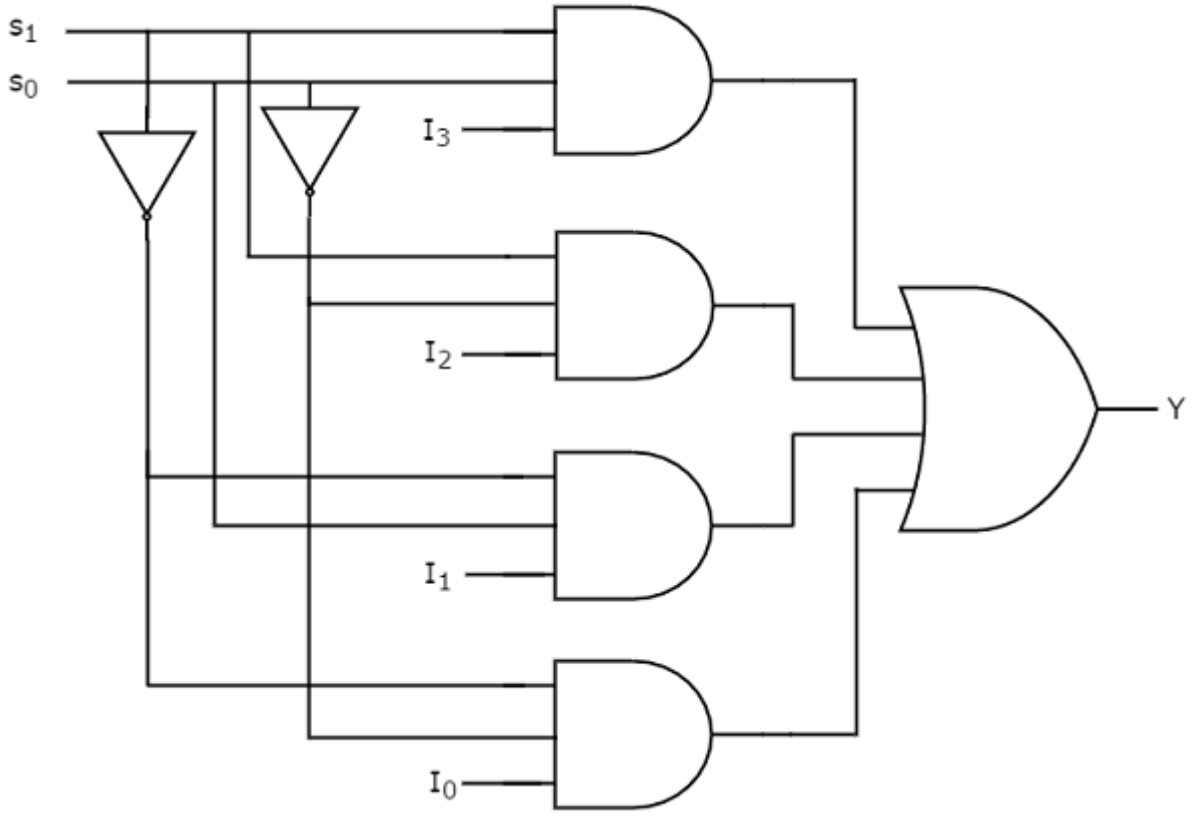
Selection Lines	Output

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

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

Y=S1′S0′I0+S1′S0I1+S1S0′I2+S1S0I3Y=S1′S0′I0+S1′S0I1+S1S0′I2+S1S0I3

We can implement this Boolean function using Inverters, AND gates & OR gate. The **circuit diagram** of 4x1 multiplexer is shown in the following figure.



We can easily understand the operation of the above circuit. Similarly, you can implement 8x1 Multiplexer and 16x1 multiplexer by following the same procedure.

Implementation of Higher-order Multiplexers.

Now, let us implement the following two higher-order Multiplexers using lower-order Multiplexers.

- 8x1 Multiplexer
- 16x1 Multiplexer

8x1 Multiplexer:

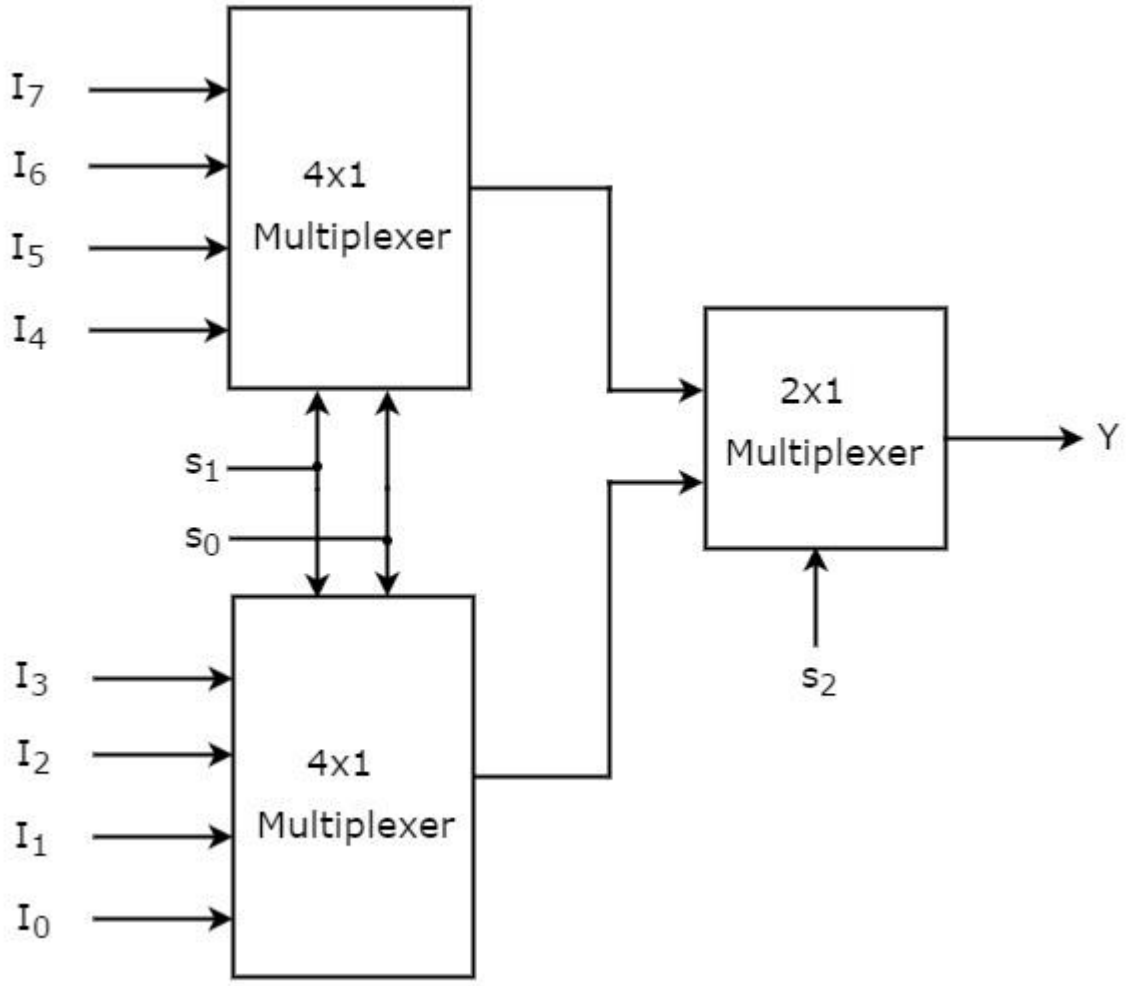
In this section, let us implement 8x1 Multiplexer using 4x1 Multiplexers and 2x1 Multiplexer. We know that 4x1 Multiplexer has 4 data inputs, 2 selection lines and one output. Whereas, 8x1 Multiplexer has 8 data inputs, 3 selection lines and one output.

So, we require two **4x1 Multiplexers** in first stage in order to get the 8 data inputs. Since, each 4x1 Multiplexer produces one output, we require a **2x1 Multiplexer** in second stage by considering the outputs of first stage as inputs and to produce the final output.

Let the 8x1 Multiplexer has eight data inputs I_7 to I_0, three selection lines s_2, s_1 & s0 and one output Y. The **Truth table** of 8x1 Multiplexer is shown below.

Selection Inputs			**Output**
S_2	**S_1**	**S_0**	**Y**
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

We can implement 8x1 Multiplexer using lower order Multiplexers easily by considering the above Truth table. The **block diagram** of 8x1 Multiplexer is shown in the following figure.



The same **selection lines, s_1 & s_0** are applied to both 4x1 Multiplexers. The data inputs of upper 4x1 Multiplexer are I_7 to I_4 and the data inputs of lower 4x1 Multiplexer are I_3 to I_0. Therefore, each 4x1 Multiplexer produces an output based on the values of selection lines, s_1 & s_0.

The outputs of first stage 4x1 Multiplexers are applied as inputs of 2x1 Multiplexer that is present in second stage. The other **selection line, s_2** is applied to 2x1 Multiplexer.

- If s_2 is zero, then the output of 2x1 Multiplexer will be one of the 4 inputs I_3 to I_0 based on the values of selection lines s_1 & s_0.
- If s_2 is one, then the output of 2x1 Multiplexer will be one of the 4 inputs I_7 to I_4 based on the values of selection lines s_1 & s_0.

Therefore, the overall combination of two 4x1 Multiplexers and one 2x1 Multiplexer performs as one 8x1 Multiplexer.

Read the above document and revise the concept of multiplexers. Now implement the following on EDA Playgrounds.

1. **Write the Verilog code for a 4-to-1 multiplexer design and test bench.**
2. **Write the Verilog code for a Demultiplexer design and test bench.**

LAB 7

DIGITAL DESIGN

E21CSEU0760

SOUMYA DUBEY

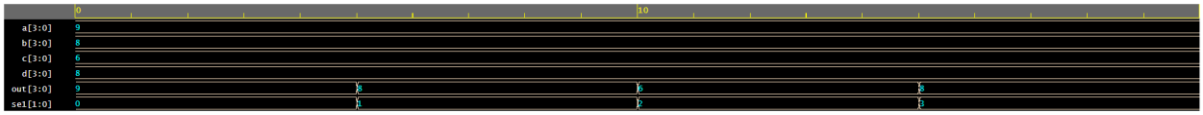
Q1)

SV/Verilog Testbench

```
//Soumya Dubey
//E21CSEU0760
module tb_4x1_mux;
 reg [3:0] a;
 reg [1:0] sel;
 reg [3:0] b;
 reg [3:0] c;
 reg [3:0] d;
 wire [3:0] out;


 mux_4x1_case mux0 (.a(a), .b(b), .c(c), .d(d), .out(out),
.sel(sel));
 initial begin


 a = 4'b1001;
 b = 4'b1000;
 c = 4'b0110;
 d = 4'b1000;


 sel[1] = 1'b0; sel[0] = 1'b0;
 #5
 sel[1] = 1'b0; sel[0] = 1'b1;
 #5
 sel[1] = 1'b1; sel[0] = 1'b0;
 #5
 sel[1] = 1'b1; sel[0] = 1'b1;
 #5
 $finish;

 end
 initial begin
 $dumpfile("dump.vcd");
 $dumpvars;
 end
endmodule
```

```
module mux_4x1_case(input [3:0] a, input [1:0] sel, input [3:0]b, input[3:0]c, input [3:0]d,
output reg[3:0] out);
 always @(a or b or c or d or sel) begin
 case(sel)
 2'b00 : out<=a;
 2'b01 : out<=b;
 2'b10 : out<=c;
 2'b11 : out<=d;
 endcase
 end
endmodule
```

Waveform:



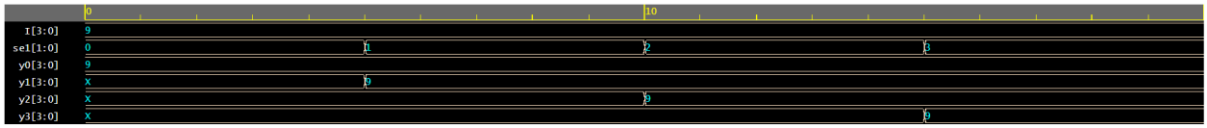
2)

SV/Verilog Testbench

```
//Soumya Dubey
//E21CSEU0760
module tb_1to4_demux;
 reg [3:0] I;
 reg [1:0] sel;
 wire [3:0] y0;
 wire [3:0] y1;
 wire [3:0] y2;
 wire [3:0] y3;


 demux_1x4 demux0 (.I(I), .y0(y0), .y1(y1), .y2(y2),
.y3(y3), .sel(sel));
 initial begin

 I = 4'b1001;


 sel[1] = 1'b0; sel[0] = 1'b0;
 #5
 sel[1] = 1'b0; sel[0] = 1'b1;
 #5
 sel[1] = 1'b1; sel[0] = 1'b0;
 #5
 sel[1] = 1'b1; sel[0] = 1'b1;
 #5
 $finish;

 end
 initial begin
 $dumpfile("dump.vcd");
 $dumpvars;
 end
endmodule
```

```
module demux_1x4(input [3:0] I, input [1:0] sel, output reg[3:0] y0, output reg[3:0]
y1,
output reg[3:0] y2, output reg[3:0] y3);
 always @(I or sel) begin
 case(sel)
 2'b00 : y0<=I;
 2'b01 : y1<=I;
 2'b10 : y2<=I;
 2'b11 : y3<=I;
 endcase
 end
endmodule
```

Waveform:



Lab Assignment 8

1. Understand the working of a 8:3 encoder and perform the following operations
   a) Find the truth table of the above digital circuit
   b) Minimize the expressions formed from the truth table in a) part
   c) Draw the systematic circuit diagram from the expressions formed in b) part
   d) Write Verilog coding from the expressions obtained in b) part and test your code using testbench module
2. Understand the working of a 3:8 Decoder and perform the following operations
   a) Find the truth table of the above digital circuit
   b) Minimize the expressions formed from the truth table in a) part
   c) Draw the systematic circuit diagram from the expressions formed in b) part
   d) Write Verilog coding from the expressions obtained in b) part and test your code using testbench module

LAB ASSIGNMENT 8
DIGITAL DESIGN
SOUMYA DUBEY
E21CSEU0760

1_VERILOG

```
//E21CSEU0760
//SOUMYA DUBEY
module Encoder(a0,a1,a2,a3,a4,a5,a6,a7,d,e,f);
input a0,a1,a2,a3,a4,a5,a6,a7;
output d,e,f;
or(d,a4,a5,a6,a7);
or(e,a2,a3,a6,a7);
or(f,a1,a3,a5,a7);
endmodule
```

1_TESTBENCH

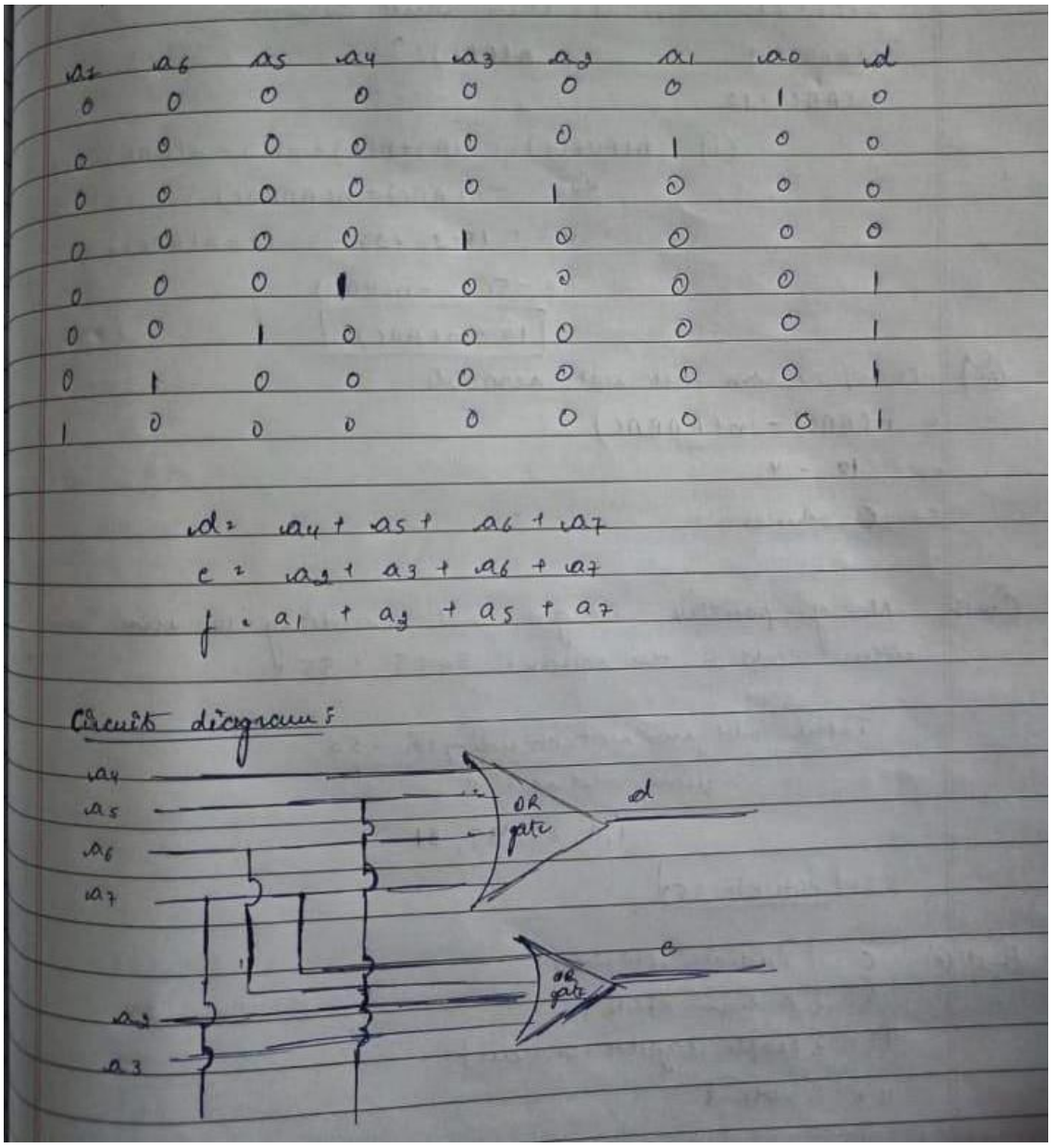
```
module testbench;
reg A0;
reg A1;
reg A2;
reg A3;
reg A4;
reg A5;
reg A6;
reg A7;
wire D;
wire  E;
wire F;
Encoder uut (.a0(A0), .a1(A1), .a2(A2), .a3(A3), .a4(A4), .a5(A5), .a6(A6),
.a7(A7), .d(D), .e(E), .f(F));
initial begin
A0 = 1;
A1 = 0;
A2 = 0;
A3 = 0;
A4 = 0;
A5 = 0;
A6 = 0;
A7 = 0;
$display("A0 = %b, A1 = %b, A2 = %b,A3 = %b, A4 = %b, A5 = %b,A6 = %b, A7 =
%b, D = %b,E = %b, F = %b",A0,A1,A2,A3,A4,A5,A6,A7,D,E,F);
#100;
A0 = 0;
A1 = 1;
A2 = 0;
A3 = 0;
```

```
A4 = 0;
A5 = 0;
A6 = 0;
A7 = 0;
$display("A0 = %b, A1 = %b, A2 = %b,A3 = %b, A4 = %b, A5 = %b,A6 = %b, A7 =
%b, D = %b,E = %b, F = %b",A0,A1,A2,A3,A4,A5,A6,A7,D,E,F);
#100;
A0 = 0;
A1 = 0;
A2 = 1;
A3 = 0;
A4 = 0;
A5 = 0;
A6 = 0;
A7 = 0;
$display("A0 = %b, A1 = %b, A2 = %b,A3 = %b, A4 = %b, A5 = %b,A6 = %b, A7 =
%b, D = %b,E = %b, F = %b",A0,A1,A2,A3,A4,A5,A6,A7,D,E,F);
#100;
A0 = 0;
A1 = 0;
A2 = 0;
A3 = 1;
A4 = 0;
A5 = 0;
A6 = 0;
A7 = 0;
$display("A0 = %b, A1 = %b, A2 = %b,A3 = %b, A4 = %b, A5 = %b,A6 = %b, A7 =
%b, D = %b,E = %b, F = %b",A0,A1,A2,A3,A4,A5,A6,A7,D,E,F);
#100;
A0 = 0;
A1 = 0;
A2 = 0;
A3 = 0;
A4 = 1;
A5 = 0;
A6 = 0;
A7 = 0;
$display("A0 = %b, A1 = %b, A2 = %b,A3 = %b, A4 = %b, A5 = %b,A6 = %b, A7 =
%b, D = %b,E = %b, F = %b",A0,A1,A2,A3,A4,A5,A6,A7,D,E,F);
#100;
A0 = 0;
A1 = 0;
A2 = 0;
A3 = 0;
A4 = 0;
A5 = 1;
A6 = 0;
```

```
A7 = 0;
$display("A0 = %b, A1 = %b, A2 = %b,A3 = %b, A4 = %b, A5 = %b,A6 = %b, A7 =
%b, D = %b,E = %b, F = %b",A0,A1,A2,A3,A4,A5,A6,A7,D,E,F);
#100;
A0 = 0;
A1 = 0;
A2 = 0;
A3 = 0;
A4 = 0;
A5 = 0;
A6 = 1;
A7 = 0;
$display("A0 = %b, A1 = %b, A2 = %b,A3 = %b, A4 = %b, A5 = %b,A6 = %b, A7 =
%b, D = %b,E = %b, F = %b",A0,A1,A2,A3,A4,A5,A6,A7,D,E,F);
#100;
A0 = 0;
A1 = 0;
A2 = 0;
A3 = 0;
A4 = 0;
A5 = 0;
A6 = 0;
A7 = 1;
$display("A0 = %b, A1 = %b, A2 = %b,A3 = %b, A4 = %b, A5 = %b,A6 = %b, A7 =
%b, D = %b,E = %b, F = %b",A0,A1,A2,A3,A4,A5,A6,A7,D,E,F);
#100
$display("A0 = %b, A1 = %b, A2 = %b,A3 = %b, A4 = %b, A5 = %b,A6 = %b, A7 =
%b, D = %b,E = %b, F = %b",A0,A1,A2,A3,A4,A5,A6,A7,D,E,F);
end
initial
begin
$dumpfile("dump.vcd");
$dumpvars(1);
end
endmodule
```

1_WAVEFORM

```
[2022-05-27 09:43:57 EDT] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
A0 = 1, A1 = 0, A2 = 0,A3 = 0, A4 = 0, A5 = 0,A6 = 0, A7 = 0,  D = z,E = z, F = z
VCD info: dumpfile dump.vcd opened for output.
A0 = 0, A1 = 1, A2 = 0,A3 = 0, A4 = 0, A5 = 0,A6 = 0, A7 = 0,  D = 0,E = 0, F = 0
A0 = 0, A1 = 0, A2 = 1,A3 = 0, A4 = 0, A5 = 0,A6 = 0, A7 = 0,  D = 0,E = 0, F = 1
A0 = 0, A1 = 0, A2 = 0,A3 = 1, A4 = 0, A5 = 0,A6 = 0, A7 = 0,  D = 0,E = 1, F = 0
A0 = 0, A1 = 0, A2 = 0,A3 = 0, A4 = 1, A5 = 0,A6 = 0, A7 = 0,  D = 0,E = 1, F = 1
A0 = 0, A1 = 0, A2 = 0,A3 = 0, A4 = 0, A5 = 1,A6 = 0, A7 = 0,  D = 1,E = 0, F = 0
A0 = 0, A1 = 0, A2 = 0,A3 = 0, A4 = 0, A5 = 0,A6 = 1, A7 = 0,  D = 1,E = 0, F = 1
A0 = 0, A1 = 0, A2 = 0,A3 = 0, A4 = 0, A5 = 0,A6 = 0, A7 = 1,  D = 1,E = 1, F = 0
A0 = 0, A1 = 0, A2 = 0,A3 = 0, A4 = 0, A5 = 0,A6 = 0, A7 = 1,  D = 1,E = 1, F = 1
Finding VCD file...
./dump.vcd
[2022-05-27 09:43:58 EDT] Opening EPWave...
Done
```



2_VERILOG

```
//E21CSEU0760
//SOUMYA DUBEY
module decoder (x,y,z,d0,d1,d2,d3,d4,d5,d6,d7);
input x,y,z;
output d0,d1,d2,d3,d4,d5,d6,d7;
assign d0 = (~x)&(~y)&(~z);
assign d1 = (~x)&(~y)&(z);
assign d2 = (~x)&(y)&(~z);
assign d3 = (~x)&(y)&(z);
assign d4 = (x)&(~y)&(~z);
assign d5 = (x)&(~y)&(z);
assign d6 = (x)&(y)&(~z);
assign d7 = (x)&(y)&(z);
endmodule
```

2_TESTBENCH

```
module testbench();
reg x,y,z;
wire d0,d1,d2,d3,d4,d5,d6,d7;
decoder obj (x,y,z,d0,d1,d2,d3,d4,d5,d6,d7);
initial begin
x = 1'b0;
y = 1'b0;
z = 1'b0;
#1
$display(" x = %b, y = %b, z = %b, d0 = %b,d1 = %b,d2 = %b,d3= %b,d4=
%b,d5= %b,d6= %b,d7= %b",x,y,z,d0,d1,d2,d3,d4,d5,d6,d7);
x = 1'b0;
y = 1'b0;
z = 1'b1;
#1
$display(" x = %b, y = %b, z = %b, d0 = %b,d1 = %b,d2 = %b,d3= %b,d4=
%b,d5= %b,d6= %b,d7= %b",x,y,z,d0,d1,d2,d3,d4,d5,d6,d7);
x = 1'b0;
y = 1'b1;
z = 1'b0;
#1
$display(" x = %b, y = %b, z = %b, d0 = %b,d1 = %b,d2 = %b,d3= %b,d4=
%b,d5= %b,d6= %b,d7= %b",x,y,z,d0,d1,d2,d3,d4,d5,d6,d7);
x = 1'b0;
y = 1'b1;
z = 1'b1;
#1
$display(" x = %b, y = %b, z = %b, d0 = %b,d1 = %b,d2 = %b,d3= %b,d4=
%b,d5= %b,d6= %b,d7= %b",x,y,z,d0,d1,d2,d3,d4,d5,d6,d7);
x = 1'b1;
y = 1'b0;
```

```
z = 1'b0;
#1
$display(" x = %b, y = %b, z = %b, d0 = %b,d1 = %b,d2 = %b,d3= %b,d4=
%b,d5= %b,d6= %b,d7= %b",x,y,z,d0,d1,d2,d3,d4,d5,d6,d7);
x = 1'b1;
y = 1'b0;
z = 1'b1;
#1
$display(" x = %b, y = %b, z = %b, d0 = %b,d1 = %b,d2 = %b,d3= %b,d4=
%b,d5= %b,d6= %b,d7= %b",x,y,z,d0,d1,d2,d3,d4,d5,d6,d7);
x = 1'b1;
y = 1'b1;
z = 1'b0;
#1
$display(" x = %b, y = %b, z = %b, d0 = %b,d1 = %b,d2 = %b,d3= %b,d4=
%b,d5= %b,d6= %b,d7= %b",x,y,z,d0,d1,d2,d3,d4,d5,d6,d7);
x = 1'b1;
y = 1'b1;
z = 1'b1;
#1
$display(" x = %b, y = %b, z = %b, d0 = %b,d1 = %b,d2 = %b,d3= %b,d4=
%b,d5= %b,d6= %b,d7= %b",x,y,z,d0,d1,d2,d3,d4,d5,d6,d7);
end
initial begin
$dumpfile("dump.vcd");
$dumpvars;
end
endmodule
```

2_WAVEFORM

```
[2022-05-29 05:21:50 EDT] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dump.vcd opened for output.
 x = 0, y = 0, z = 0, d0 = 1,d1 = 0,d2 = 0,d3= 0,d4= 0,d5= 0,d6= 0,d7= 0
 x = 0, y = 0, z = 1, d0 = 0,d1 = 1,d2 = 0,d3= 0,d4= 0,d5= 0,d6= 0,d7= 0
 x = 0, y = 1, z = 0, d0 = 0,d1 = 0,d2 = 1,d3= 0,d4= 0,d5= 0,d6= 0,d7= 0
 x = 0, y = 1, z = 1, d0 = 0,d1 = 0,d2 = 0,d3= 1,d4= 0,d5= 0,d6= 0,d7= 0
 x = 1, y = 0, z = 0, d0 = 0,d1 = 0,d2 = 0,d3= 0,d4= 1,d5= 0,d6= 0,d7= 0
 x = 1, y = 0, z = 1, d0 = 0,d1 = 0,d2 = 0,d3= 0,d4= 0,d5= 1,d6= 0,d7= 0
 x = 1, y = 1, z = 0, d0 = 0,d1 = 0,d2 = 0,d3= 0,d4= 0,d5= 0,d6= 1,d7= 0
 x = 1, y = 1, z = 1, d0 = 0,d1 = 0,d2 = 0,d3= 0,d4= 0,d5= 0,d6= 0,d7= 1
Finding VCD file...
./dump.vcd
[2022-05-29 05:21:51 EDT] Opening EPWave...
Done
```

D	E	F	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$d_0 = \bar{D}.\bar{E}.\bar{F}$

$d_1 = \bar{D}.\bar{E}.F$

$d_2 = \bar{D}.E.\bar{F}$

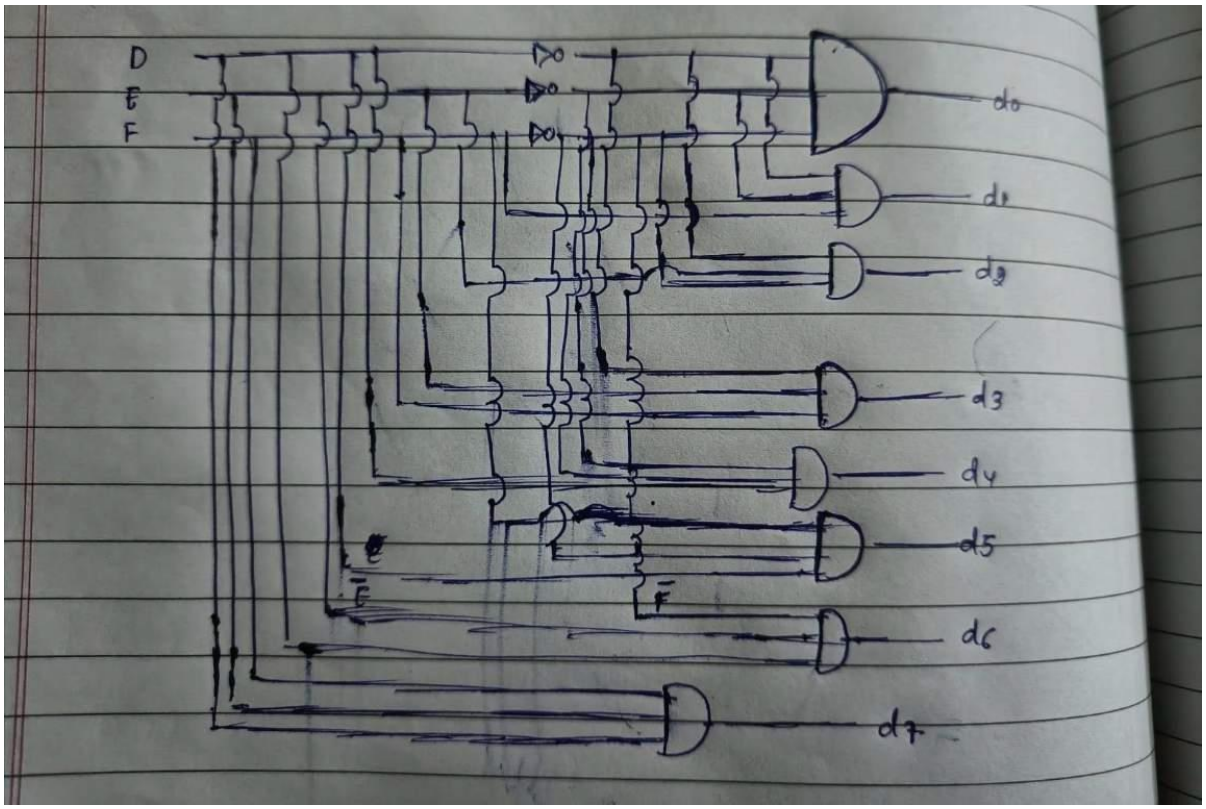
$d_3 = \bar{D}.E.F$

$d_4 = D.\bar{E}.\bar{F}$

$d_5 = D.\bar{E}.F$

$d_6 = D.E.\bar{F}$

$d_7 = D.E.F$



Lab Assignment 9

1. Understand the working of SR flip flop and perform the following operations
   a) Find the truth table of the above digital circuit
   b) Minimize the expressions formed from the truth table in a) part
   c) Draw the systematic circuit diagram from the expressions formed in b) part
   d) Write Verilog coding from the expressions obtained in b) part and test your code using testbench module
2. Understand the working of JK flip flop and perform the following operations
   a) Find the truth table of the above digital circuit
   b) Minimize the expressions formed from the truth table in a) part
   c) Draw the systematic circuit diagram from the expressions formed in b) part
   d) Write Verilog coding from the expressions obtained in b) part and test your code using testbench module.
3. Understand the working of D flip flop and perform the following operations
   a) Find the truth table of the above digital circuit
   b) Minimize the expressions formed from the truth table in a) part
   c) Draw the systematic circuit diagram from the expressions formed in b) part
   d) Write Verilog coding from the expressions obtained in b) part and test your code using testbench module

LAB-9

Soumya Dubey

E21CSEU0760

1 TESTBENCH—

```
//E21CSEU0760
//Soumya Dubey
module tb_exp;
  reg S,R,Q;
  wire QB;
  exp a1(.s(S),.r(R),.q(Q),.qb(QB));
  initial
    begin
      S=0 ; R=0 ; Q=0 ; #5;
      $display("S R Q QB");
      $display("%b %b %b %b", S, R, Q, QB);
      S=0 ; R=0 ; Q=1 ; #5;
      $display("%b %b %b %b", S, R, Q, QB);
      S=0 ; R=1 ; Q=0 ; #5;
      $display("%b %b %b %b", S, R, Q, QB);
      S=0 ; R=1 ; Q=1 ; #5;
      $display("%b %b %b %b", S, R, Q, QB);
      S=1 ; R=0 ; Q=0 ; #5;
      $display("%b %b %b %b", S, R, Q, QB);
      S=1 ; R=0 ; Q=1 ; #5;
```

```
    $display("%b %b %b %b", S, R, Q, QB);
    S=1 ; R=1 ; Q=0 ; #5;
    $display("%b %b %b %b", S, R, Q, QB);
    S=1 ; R=1 ; Q=1 ; #5;
    $display("%b %b %b %b", S, R, Q, QB);
    S=1 ; R=1 ; Q=0 ; #5;
    $display("%b %b %b %b", S, R, Q, QB);
    S=1 ; R=1 ; Q=1 ; #5;
    $display("%b %b %b %b", S, R, Q, QB);

  end

 initial
  begin
    $dumpfile("dump.vcd");
    $dumpvars(1);
  end
endmodule
```

1 DESIGN CODE—

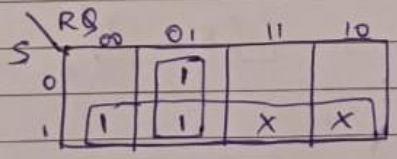
```
//E21CSEU0760
//Soumya Dubey
module exp(input s,r,q, output qb);
 assign qb  =  (s | (~r&q));
endmodule
```

1 TRUTH TABLE—

1) Characteristic truth table :-

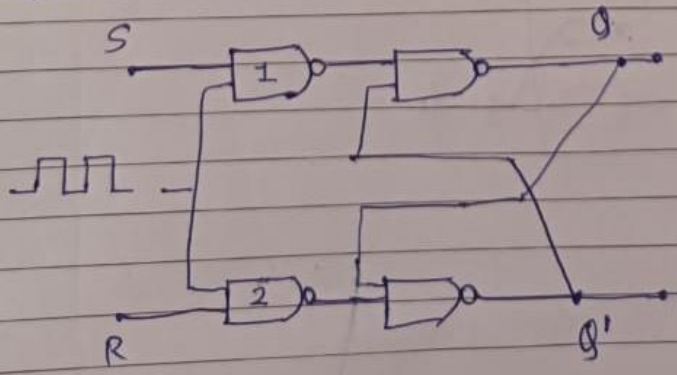
Clk	S	R	Q	Q_B
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	x
1	1	1	1	x

(b) Expression :-



$$Q_B = S + R'Q$$

(c) Systematic Diagram :-



(d) Excitation table :—

Present state Q	Next state QB	Present Input S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

S =

Q \ QB	0	1
0		1
1		X

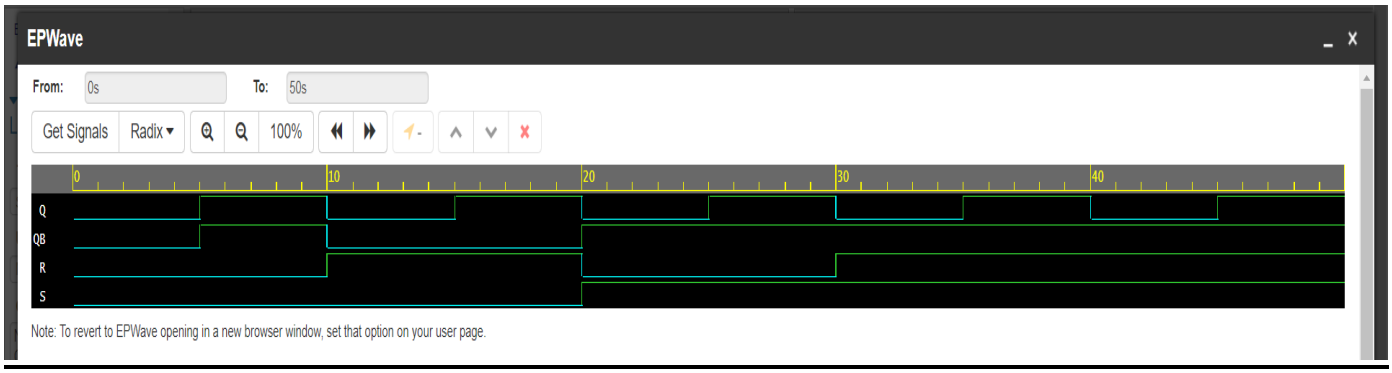
S = QB

R =

Q \ QB	0	1
0	X	
1	1	

R = QB'

1 WAVEFORM—



2 TESTBENCH—

```
//E21CSEU0760
//Soumya Dubey
module tb_exp;
  reg J,K,Q;
  wire QB;

  exp a1(.j(J),.k(K),.q(Q),.qb(QB));

  initial
    begin
      J=0 ; K=0 ; Q=0 ; #5;
      $display("J K Q QB ");
      $display("%b %b %b %b", J, K, Q, QB);
      J=0 ; K=0 ; Q=1 ; #5;
      $display("%b %b %b %b", J, K, Q, QB);
      J=0 ; K=1 ; Q=0 ;#5;
      $display("%b %b %b %b", J, K, Q, QB);
      J=0 ; K=1 ; Q=1 ;#5;
      $display("%b %b %b %b", J, K, Q, QB);
      J=1 ; K=0 ; Q=0 ;#5;
      $display("%b %b %b %b", J, K, Q, QB);
      J=1 ; K=0 ; Q=1 ;#5;
      $display("%b %b %b %b", J, K, Q, QB);
      J=1 ; K=1 ; Q=0 ;#5;
      $display("%b %b %b %b", J, K, Q, QB);
```

```
    J=1 ; K=1 ; Q=1 ;#5;
    $display("%b %b %b %b", J, K, Q, QB);

  end

 initial
  begin
    $dumpfile("dump.vcd");
    $dumpvars(1);
  end
endmodule
```

2 DESIGN CODE—

```
//E21CSEU0760
//Soumya Dubey
module exp(input j,k,q, output qb);
 assign qb = ((j&~q) | (~k&q));
endmodule
```

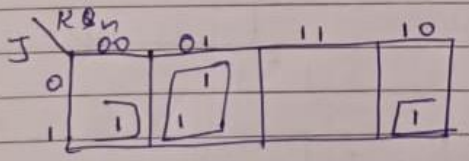
2 TRUTHTABLE—

2)

CLK	J	K	Q_B
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Q_n'

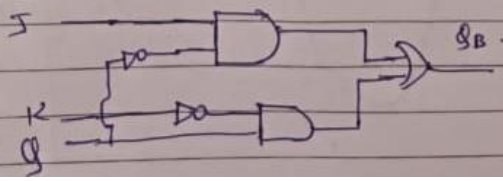
Characteristic Truth table :-

J	K	Q	Q_B
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



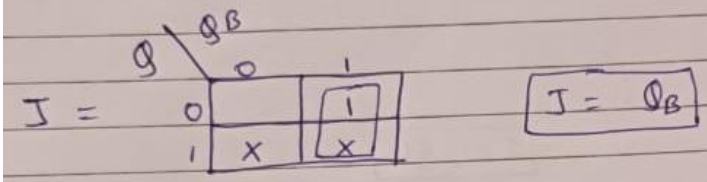
So, $Q_B = JQ' + K'Q$

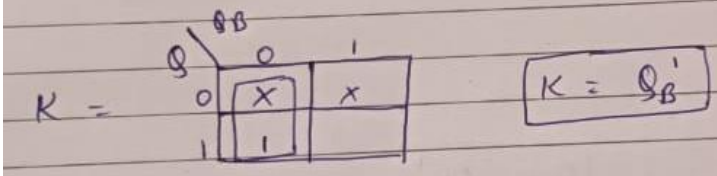
(c) Systematic Diagram :-



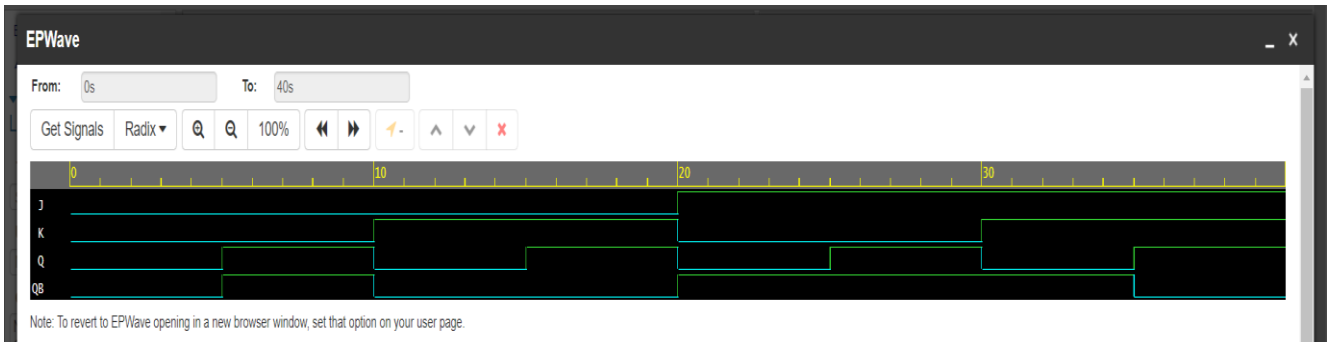
(d) Excitation table :-

Present state Q	Next state Q_B	Present Input J	Input K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0





2 WAVEFORM—



3 TESTBENCH—

```
//E21CSEU0760
//Soumya Dubey
module tb_exp;
  reg D,Q;
  wire QB;

  exp a1(.d(D),.q(Q),.qb(QB));

  initial
    begin
      D=0 ; Q=0 ; #5;
      $display("D Q QB");
      $display("%b %b %b ", D, Q, QB);
      D=0 ; Q=1 ; #5;
      $display("%b %b %b", D, Q, QB);
      D=1 ; Q=0 ; #5;
      $display("%b %b %b", D, Q, QB);
      D=1 ; Q=1 ; #5;
      $display("%b %b %b", D, Q, QB);

    end

  initial
```

```
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
endmodule
```

3 DESIGN CODE—

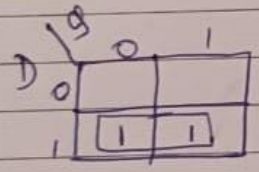
```
//E21CSEU0760
//Soumya Dubey

module exp(input d,q, output qb);
  assign qb = (d);
endmodule
```

3 TRUTHTABLE —

3) (a) Truth Table :-

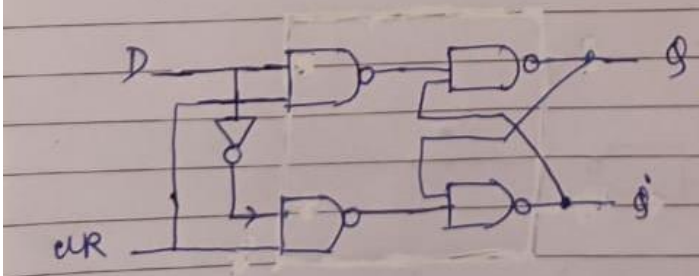
D	Q	Q_B
0	0	0
0	1	0
1	0	1
1	1	1

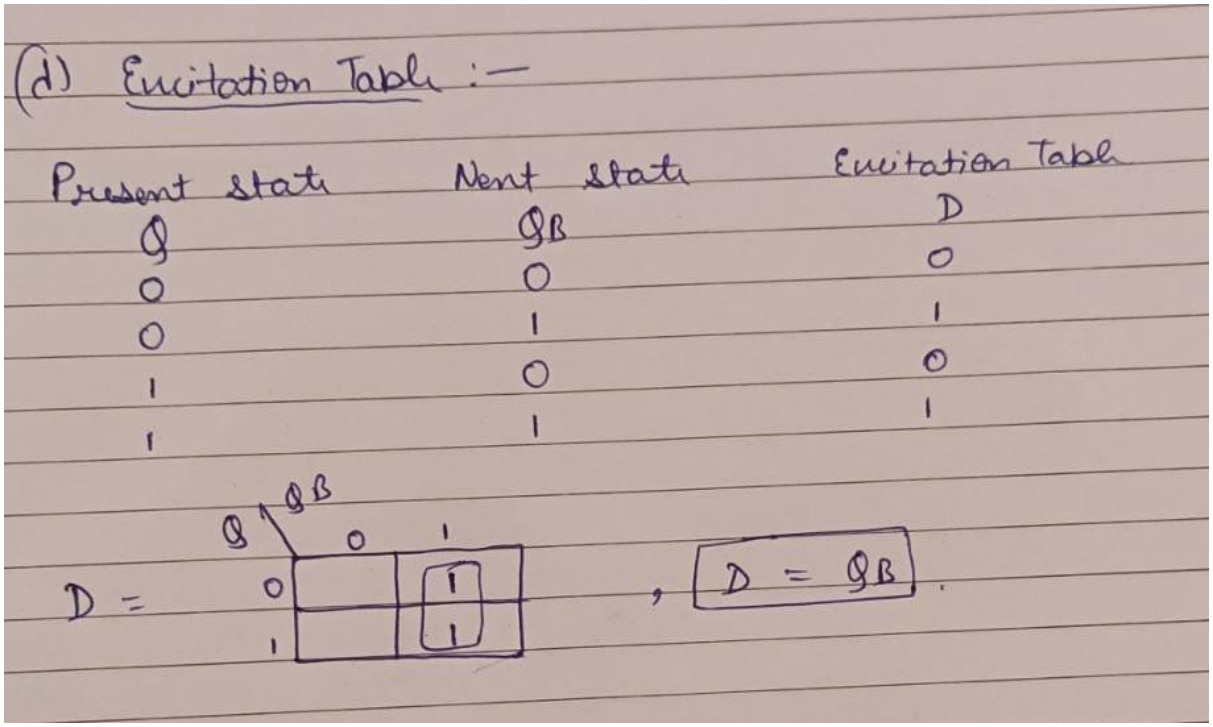
(b) Expression :-



$Q_B = D$

(c) Systematic Diagram :-




(d) Excitation Table :—
Present state
Q
0
0
1
1
Next state
QB
0
1
0
1
Excitation Table
D
0
1
0
1
D =
Q
QB
0
1
0
1
1
1
, D = QB

3 WAVEFORM—


EPWave
From:
0s
To:
20s
Get Signals
Radix
100%
0
10
D
Q
QB
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

Lab Assignment 10

1. Design a 4-bit Parallel In Parallel Out (PIPO) shift register and perform the following operations:
   A) Draw the diagram to show the working of PIPO shift register using D flipflops.
   B) Design the Verilog code using D flipflops.
   C) Load the value 1001 as the final output of given shift register by giving inputs in the testbench part.
2. Design a 4-bit Serial In Parallel Out (SIPO) shift register and perform the following operations:
   A) Draw the diagram to show the working of SIPO shift register using D flipflops.
   B) Design the Verilog code using D flipflops.
   C) Load the value 1001 as the final output of given shift register by giving inputs in the testbench part.
3. Design a 4-bit Serial In Serial Out (SISO) shift register and perform the following operations:
   A) Draw the diagram to show the working of SISO shift register using D flipflops.
   B) Design the Verilog code using D flipflops.
   C) Load the value 1001 as the final output of given shift register by giving inputs in the testbench part.

Question 1

//E21cseu0760

//Soumya Dubey

Design1

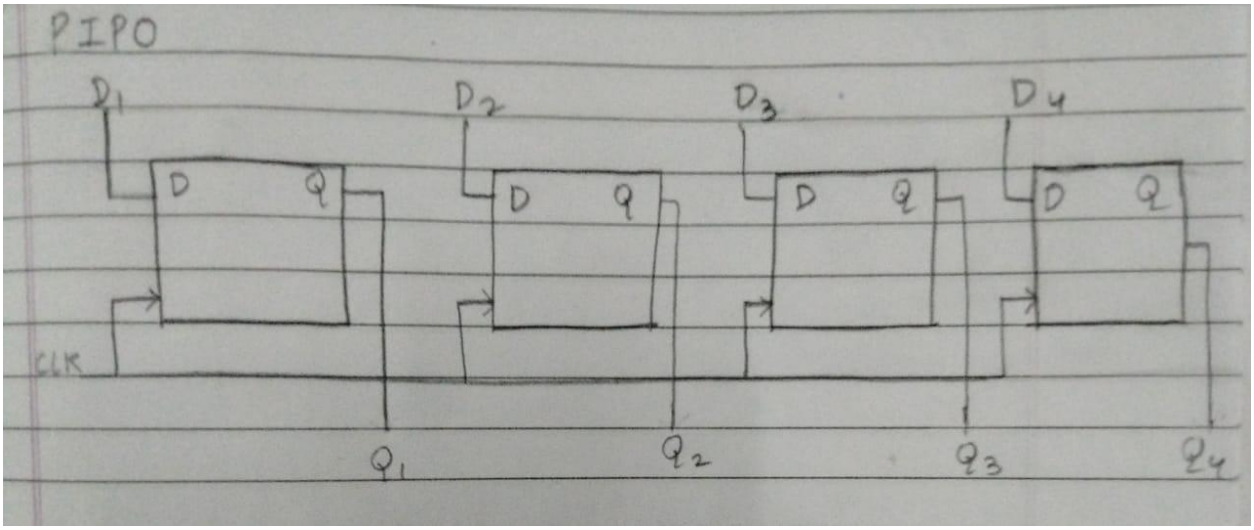
```
module PIPO(D, clk, Q);
  input [3:0]D;
  input clk;
  output reg [3:0]Q;
  always@(posedge clk) begin
    Q = D;
  end
endmodule
```

Testbench1

```
module testbench;
  reg clk;
  reg [3:0]D;
  wire [3:0]Q;
  PIPO p1 (D , clk, Q);

  initial begin
    forever
      #2
      clk = ~clk;
  end
```

```
 initial begin
   clk = 1'b0;
   D = 4'b1001;
   #4
   $display("D = %b , CLK = %b, Q = %b", D, clk, Q);

   $finish();
 end

 initial begin
   $dumpfile("dump.vcd");
   $dumpvars(1);
 end
endmodule
```

Diagram



Question 2

//E21CSEU0760

//Soumya Dubey

Design2

```
module SIPO(D, clk, Q);
  input D;
  input clk;
  output reg [3:0]Q;
  always@(posedge clk) begin
    Q[0] = Q[1];
    Q[1] = Q[2];
    Q[2] = Q[3];
    Q[3] = D;
  end
endmodule
```

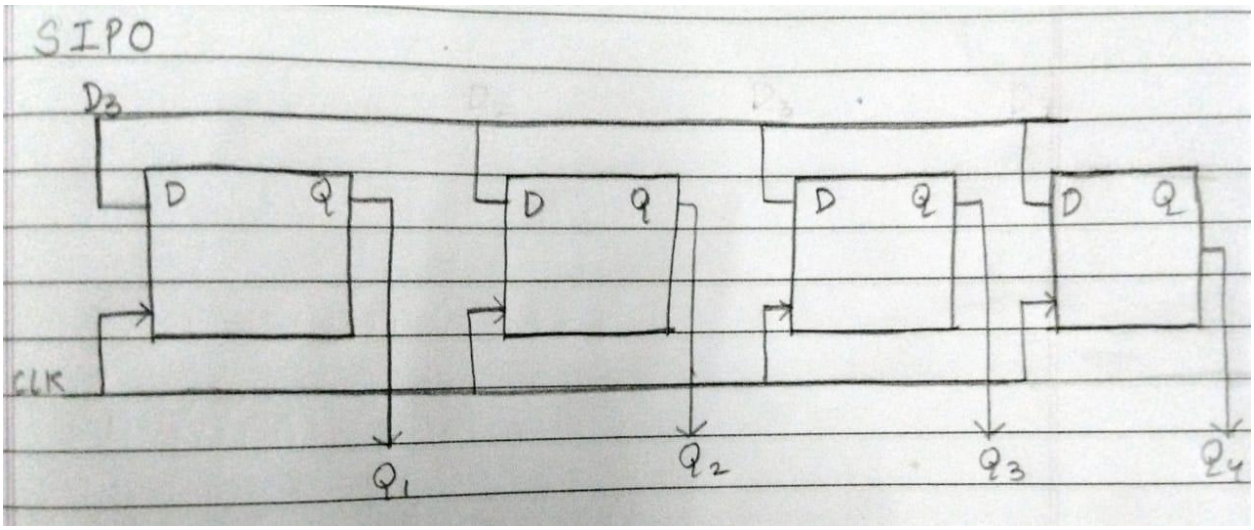
Testbench2

```
module testbench;
  reg clk;
  reg D;
  wire [3:0]Q;
  SIPO p1 (D , clk, Q);

  initial begin
    forever
      #2
      clk = ~clk;
```

```
end

initial begin
  clk = 1'b0;
  D = 1'b1;
  #4
  $display("D = %b , CLK = %b, Q = %b", D, clk, Q);

  D = 1'b0;
  #4
  $display("D = %b , CLK = %b, Q = %b", D, clk, Q);

  D = 1'b0;
  #4
  $display("D = %b , CLK = %b, Q = %b", D, clk, Q);

  D = 1'b1;
  #4
  $display("D = %b , CLK = %b, Q = %b", D, clk, Q);

  $finish();
end

initial begin
  $dumpfile("dump.vcd");
  $dumpvars(1);
end
```

endmodule

Diagram



Question 3

//E21CSEU0760

//Soumya Dubey

Design3

```
module SISO(D, clk, sout);
  input D;
  input clk;
  output reg [3:0]Q, sout;

  always@(posedge clk) begin
    sout = Q[3];
    Q[3] = Q[2];
    Q[2] = Q[1];
    Q[1] = Q[0];
```

```
   Q[0] = D;
  end
endmodule
```

Testbench3

```
module testbench;
  reg clk;
  reg D;
  wire Sout;
  SISO p1 (D , clk, sout);

  initial begin
    forever
      #2
      clk = ~clk;
  end

  initial begin
    clk = 1'b0;
    D = 1'b1;
    #4
    $display("D = %b , CLK = %b, Sout = %b", D, clk, sout);

    D = 1'b0;
    #4
    $display("D = %b , CLK = %b, Sout = %b", D, clk, sout);

    D = 1'b0;
```

```
  #4
  $display("D = %b , CLK = %b, Sout = %b", D, clk, sout);

  D = 1'b1;
  #4
  $display("D = %b , CLK = %b, Sout = %b", D, clk, sout);

  D = 1'bx;
  #4
  $display("D = %b , CLK = %b, Sout = %b", D, clk, sout);

  D = 1'bx;
  #4
  $display("D = %b , CLK = %b, Sout = %b", D, clk, sout);

  D = 1'bx;
  #4
  $display("D = %b , CLK = %b, Sout = %b", D, clk, sout);

  D = 1'bx;
  #4
  $display("D = %b , CLK = %b, Sout = %b", D, clk, sout);


  $finish();
end
```

```
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(1);
  end
endmodule
```

Diagram