School of Computer Science Engineering and Technology

Course- BTech
Course Code: CSET 105
Year- 2022
Name – Ayushman Pranav
Rollno. – E21CSEU0245
Batch – EB10

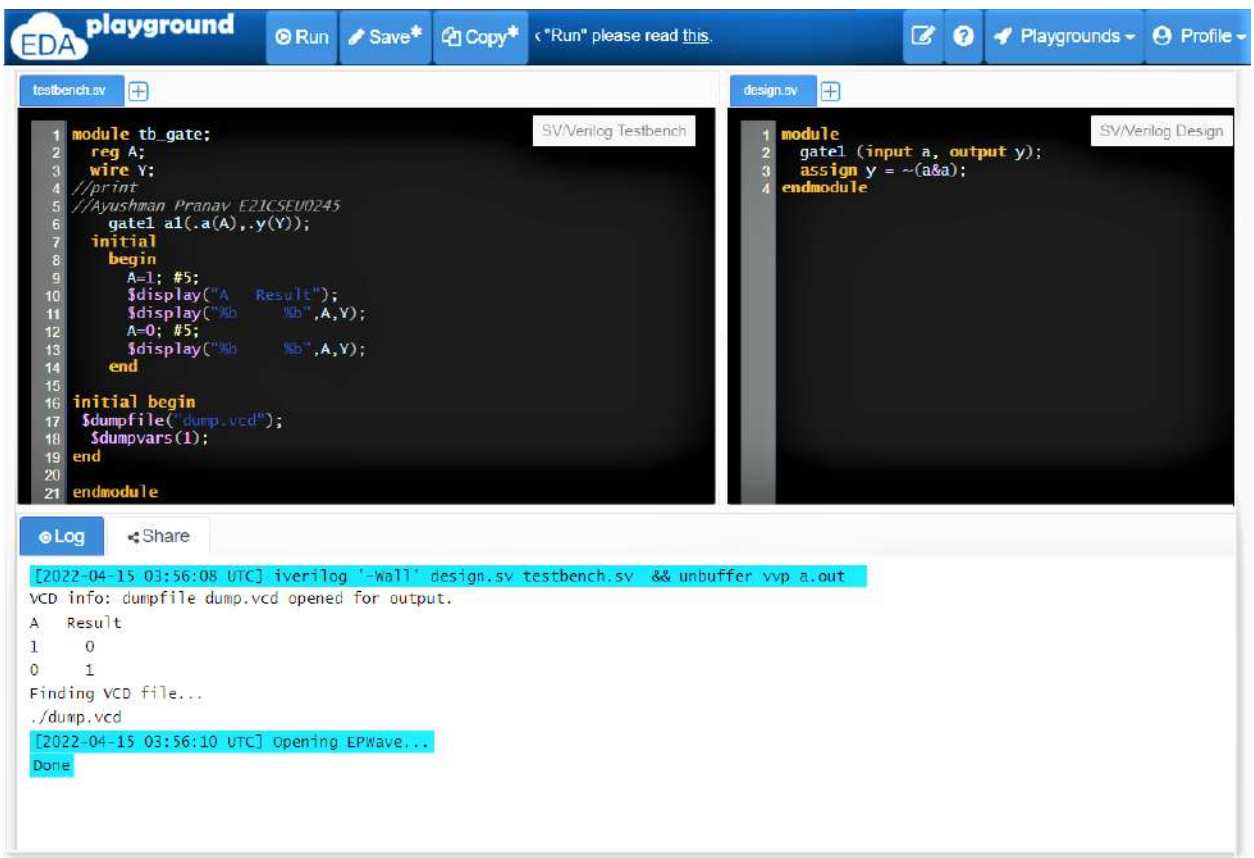
Type- Core
Course Name- Digital Logic Design Lab
Semester- Even

Lab Assignment 4.1

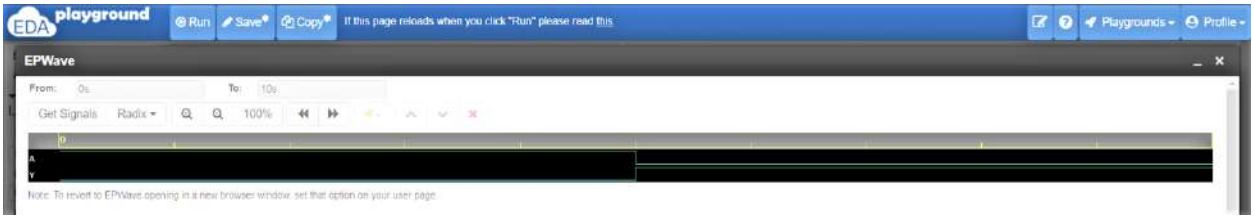
Perform the following operations:

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?

2. Represent the following gates using only NAND gates
   A. NOT



School of Computer Science Engineering and Technology



Testbench.sv

```
module tb_gate;
  reg A;
  wire Y;
//print
//Ayushman Pranav E21CSEU0245
    gate1 a1(.a(A),.y(Y));
  initial
    begin
      A=1; #5;
      $display("A   Result");
      $display("%b     %b",A,Y);
      A=0; #5;
      $display("%b     %b",A,Y);
    end

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

endmodule
```

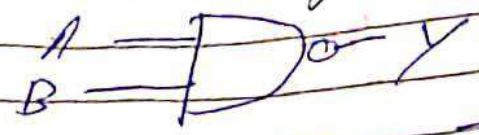
design.sv

```
module

  gate1 (input a, output y);

  assign y = ~(a&a);

endmodule
```

School of Computer Science Engineering and Technology

1a) Implement Not gate using NAND gate

NAND gate



$$Y = \overline{A \cdot B} \quad \text{--(i)}$$

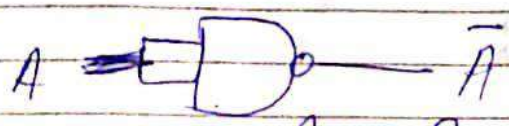
Not gate

--(ii)

let us $A = B$

put $A = B$ in (i)

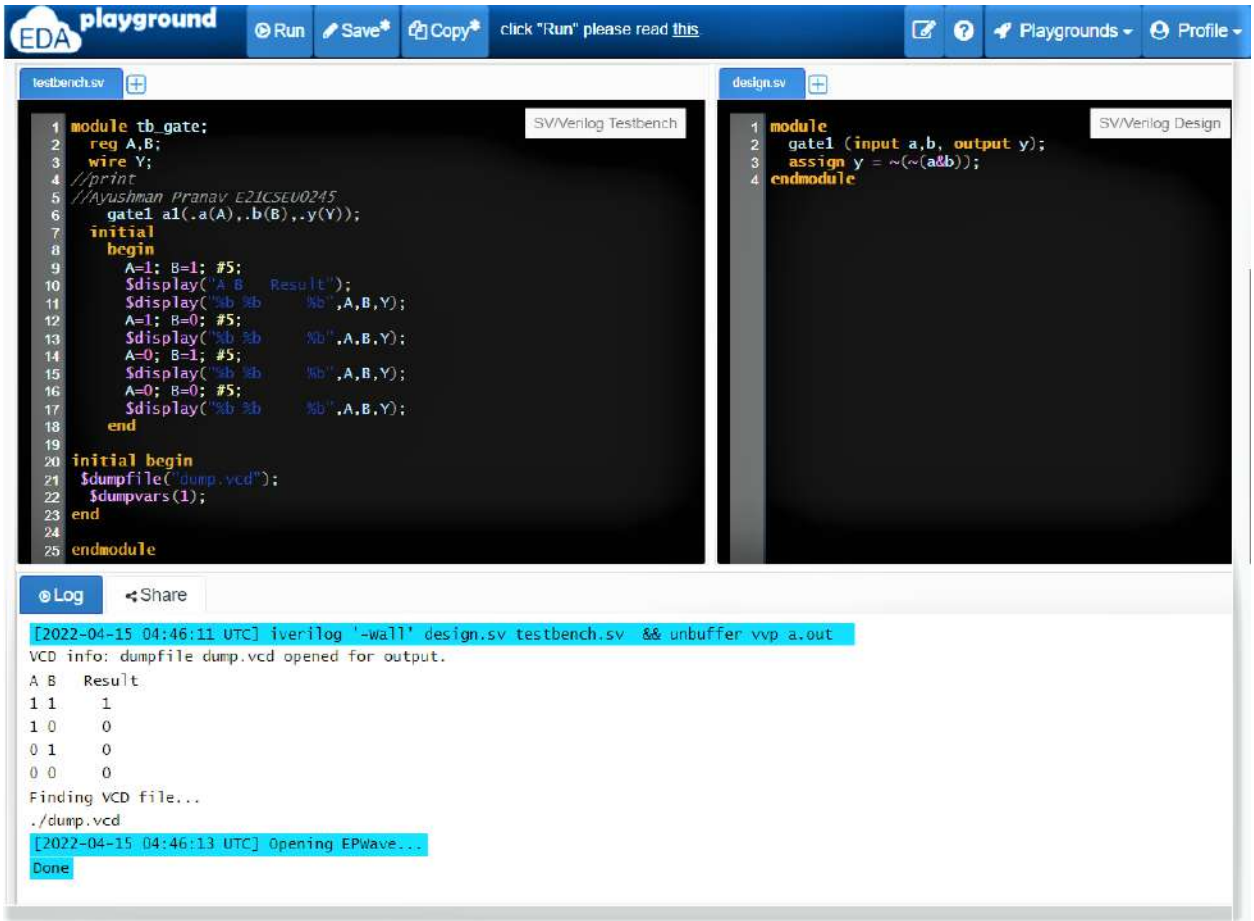
$$Y = \overline{A \cdot A} = \bar{A} \qquad \boxed{A \cdot A = A}$$

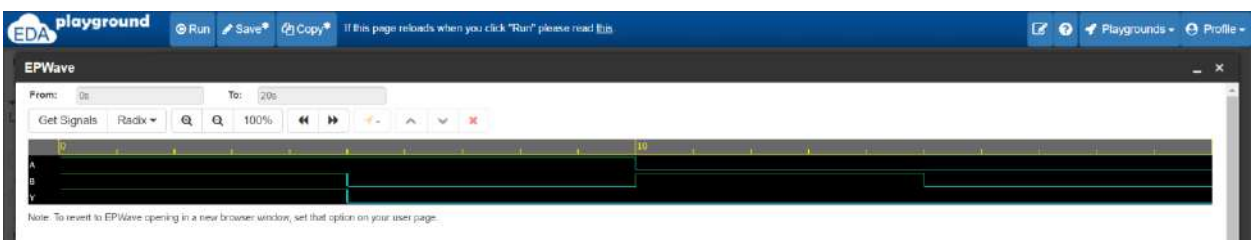


A	$Y = \bar{A}$
0	1
1	0

A	B	B = A	$\overline{AB}$	$\overline{A \cdot A}$
1	1	1	0	0
1	0	0	1	0
0	1	1	1	1
0	0	0	1	1

B. AND





Testbench.sv

```
module tb_gate;
  reg A,B;
  wire Y;
//print
//Ayushman Pranav E21CSEU0245
    gate1 a1(.a(A),.b(B),.y(Y));
  initial
```

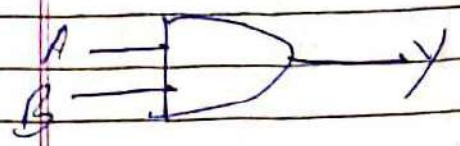
```
    begin
      A=1; B=1; #5;
      $display("A B   Result");
      $display("%b %b     %b",A,B,Y);
      A=1; B=0; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=1; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=0; #5;
      $display("%b %b     %b",A,B,Y);
    end

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

endmodule
```

Design.sv

```
module
  gate1 (input a,b, output y);
  assign y = ~(~(a&b));
endmodule
```

2) AND gate using NAND gate

And gate



And gate using NAND gate

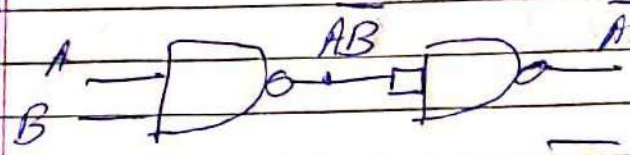
$Y = AB$

Expression $Y = AB$ using double inversion principle

A	B	AB
1	1	1
1	0	0
0	1	0
0	0	0

$\bar{\bar{Y}} = \overline{\overline{AB}}$ $\boxed{\because \bar{\bar{A}} = A}$

$Y = \overline{\overline{AB}}$



$\overline{\overline{AB}} = AB$

A	B	$\overline{AB}$	$\overline{\overline{AB}}$	AB
1	1	0	1	1
1	0	1	0	0
0	1	1	0	0
0	0	1	0	0

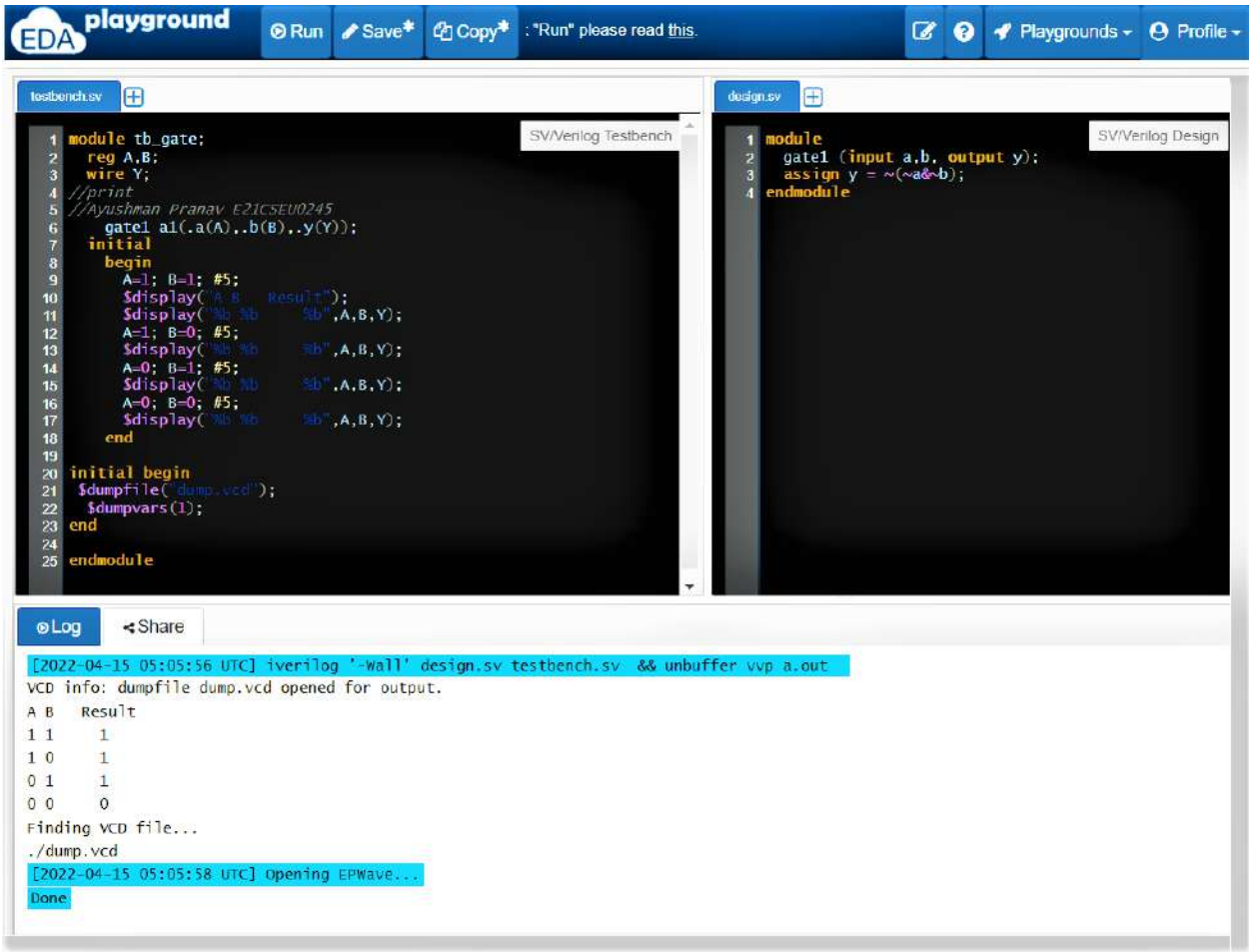
↳ Logically equivalent

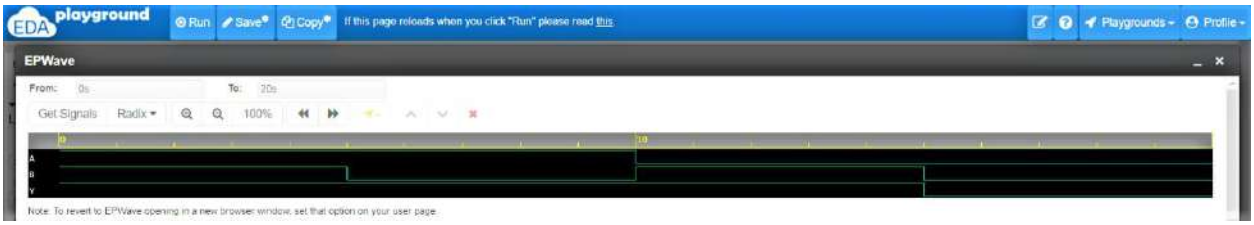
School of Computer Science Engineering and Technology

$A + B$

$\overline{AB} = \overline{A} + \overline{B}$

C. OR





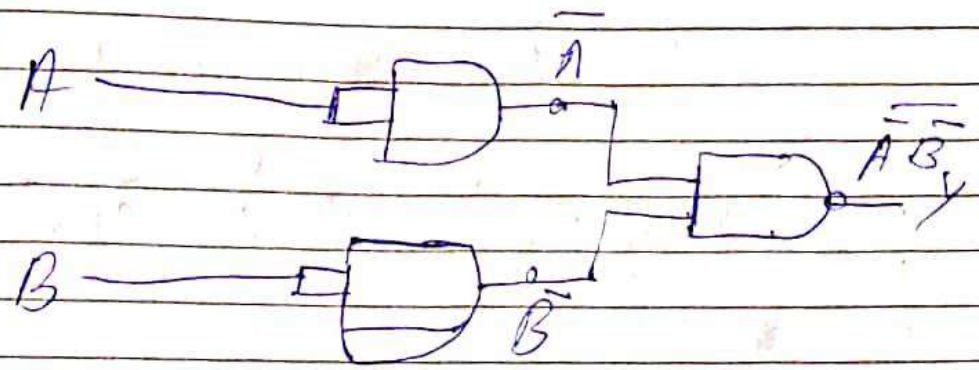
Testbench.sv

```
module tb_gate;
  reg A,B;
  wire Y;
//print
//Ayushman Pranav E21CSEU0245
    gate1 a1(.a(A),.b(B),.y(Y));
  initial
    begin
      A=1; B=1; #5;
      $display("A B   Result");
      $display("%b %b     %b",A,B,Y);
      A=1; B=0; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=1; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=0; #5;
      $display("%b %b     %b",A,B,Y);
    end

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

endmodule
```

Design.sv

```
module
  gate1 (input a,b, output y);
  assign y = ~(~a&~b);
endmodule
```

3) or gate using NAND or gate



$$Y = \overline{\bar{A}\bar{B}}$$
$$= A + B \text{ (Demorgan)}$$

A	B	$\bar{A}$	$\bar{B}$	$\bar{A}\bar{B}$	$\overline{\bar{A}\bar{B}}$	A+B
1	1	0	0	0	1	1
1	0	0	1	0	1	1
0	1	1	0	0	1	1
0	0	1	1	1	0	0

School of Computer Science Engineering and Technology

D. NOR

EDA playground | Run | Save* | Copy* | Playgrounds | Profile

testbench.sv (SV/Verilog Testbench)

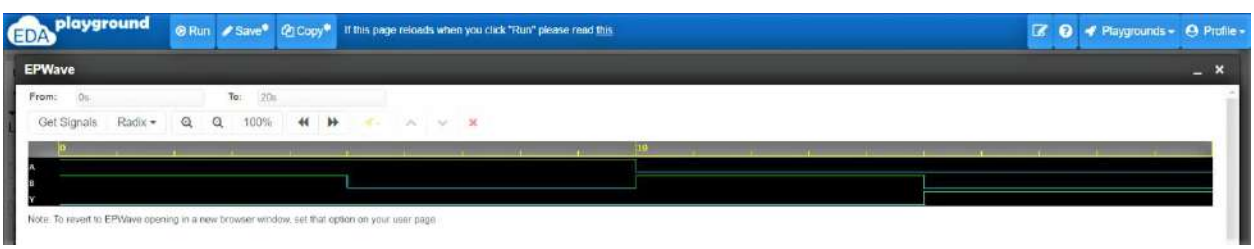
```
module tb_gate;
  reg A,B;
  wire Y;
//print
//Ayushman Pranav E21CSEU0245
    gate1 a1(.a(A),.b(B),.y(Y));
  initial
    begin
      A=1; B=1; #5;
      $display("A B   Result");
      $display("%b %b     %b",A,B,Y);
      A=1; B=0; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=1; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=0; #5;
      $display("%b %b     %b",A,B,Y);
    end

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

endmodule
```

design.sv (SV/Verilog Design)

```
module
  gate1 (input a,b, output y);
  assign y = ~(~(~a&~b));
endmodule
```

Log | Share

```
[2022-04-15 05:24:27 UTC] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dump.vcd opened for output.
A B   Result
1 1     0
1 0     0
0 1     0
0 0     1
Finding VCD file...
./dump.vcd
[2022-04-15 05:24:28 UTC] Opening EPWave...
Done
```



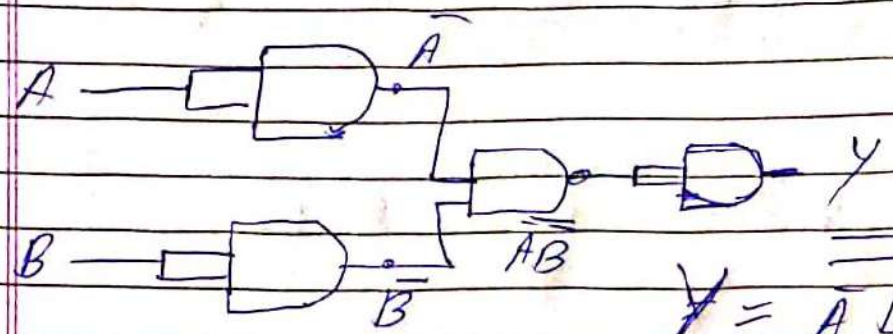
Testbench.sv

```
module tb_gate;
  reg A,B;
  wire Y;
//print
//Ayushman Pranav E21CSEU0245
    gate1 a1(.a(A),.b(B),.y(Y));
  initial
    begin
      A=1; B=1; #5;
      $display("A B   Result");
      $display("%b %b     %b",A,B,Y);
      A=1; B=0; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=1; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=0; #5;
      $display("%b %b     %b",A,B,Y);
    end

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

endmodule
```

Design.sv

```
module
  gate1 (input a,b, output y);
  assign y = ~(~(~a&~b));
endmodule
```

4) Nor gate using ~~AND~~ NAND

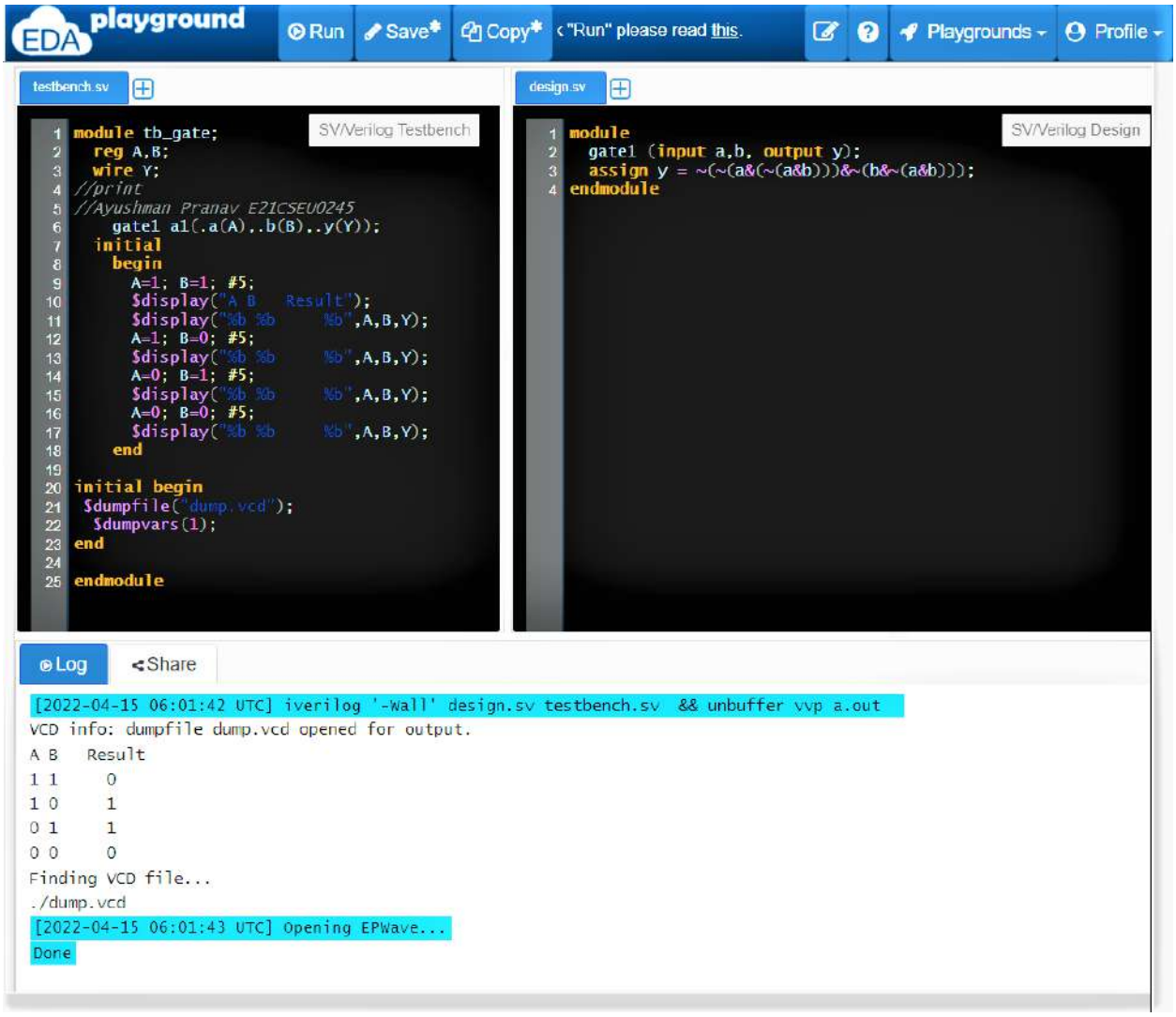


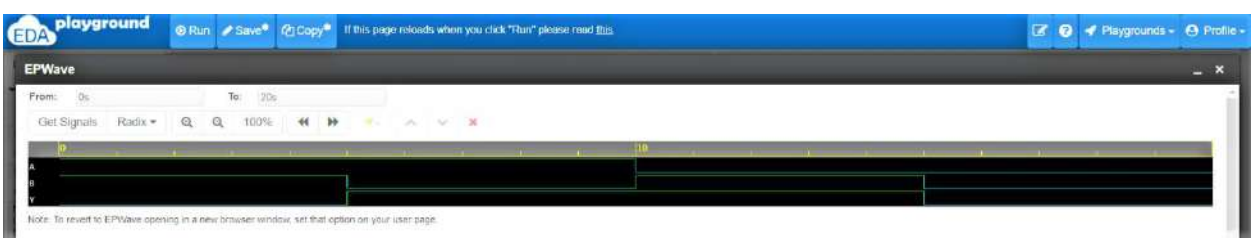
$$Y = \overline{\overline{\overline{A}\,\overline{B}}}$$
$$= \overline{\overline{A}}\;\overline{\overline{B}}$$
$$= \overline{(\overline{A+B})}$$

(Demorgan's Law).

A	B	$\overline{A}$	$\overline{B}$	$\overline{A}\,\overline{B}$	$A+B$	$\overline{A+B}$
1	1	0	0	0	1	0
1	0	0	1	0	1	0
0	1	1	0	1	1	0
0	0	1	1	1	0	1

E. XOR





Testbench.sv

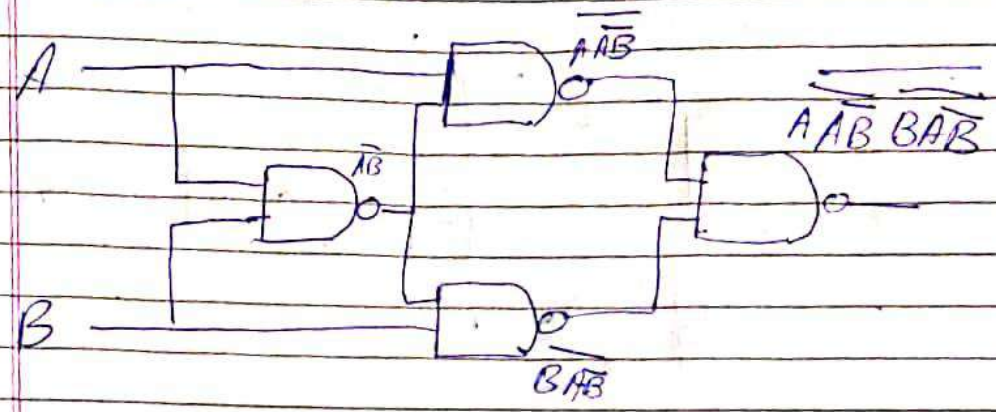
```
module tb_gate;
  reg A,B;
  wire Y;
//print
```

```
//Ayushman Pranav E21CSEU0245
    gate1 a1(.a(A),.b(B),.y(Y));
  initial
    begin
      A=1; B=1; #5;
      $display("A B   Result");
      $display("%b %b     %b",A,B,Y);
      A=1; B=0; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=1; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=0; #5;
      $display("%b %b     %b",A,B,Y);
    end

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

endmodule
```

Design.sv

```
module
  gate1 (input a,b, output y);
  assign y = ~(~(a&(~(a&b)))&~(b&~(a&b)));
endmodule
```

5) XOR gate using NAND

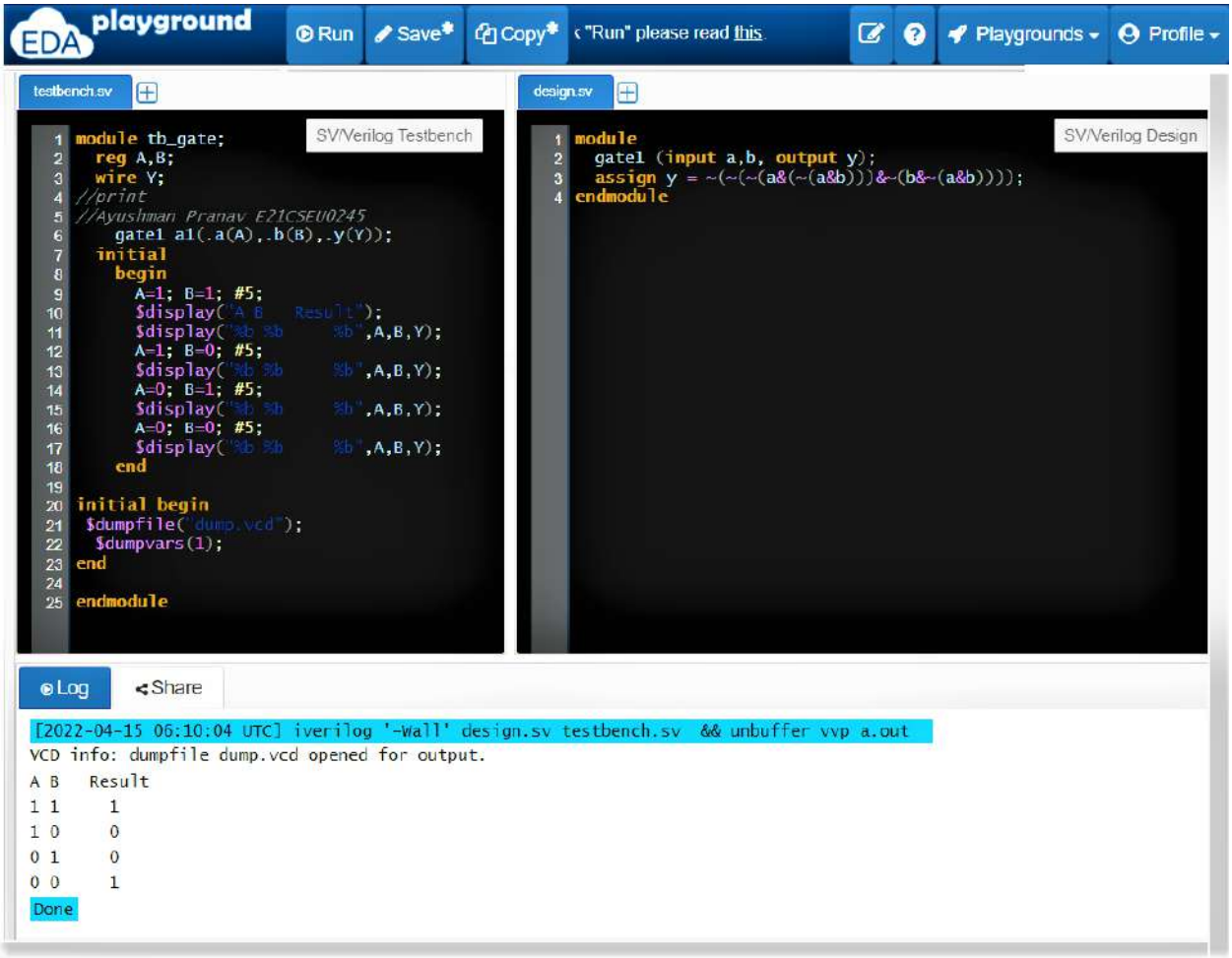


$$F(ABC) = \overline{\overline{A\overline{AB}}\ \overline{B\overline{AB}}}$$
$$= A\cdot\overline{A\overline{B}} + B\cdot\overline{A\overline{B}}$$
$$= A(\bar{A}+\bar{B}) + B(\bar{A}+\bar{B})$$
$$= A\bar{A} + A\bar{B} + B\bar{A} + B\bar{B}$$
$$= A\bar{B} + \bar{A}B$$
$$= A \oplus B$$

$\bar{A}B+\bar{B}A$	A	B	$\bar{A}$	$\bar{B}$	$\overline{AB}$	$\overline{A\overline{AB}}$	$\overline{B\overline{AB}}$	$\overline{A\overline{AB}}\cdot\overline{B\overline{AB}}$	F(ABC)
0	1	1	0	0	0	0	0	1	1
1	1	0	0	1	1	1	0	0	0
1	0	1	1	0	1	0	1	0	0
0	0	0	1	1	1	0	0	1	1

↳ logically equivalent

F. XNOR



Testbench.sv

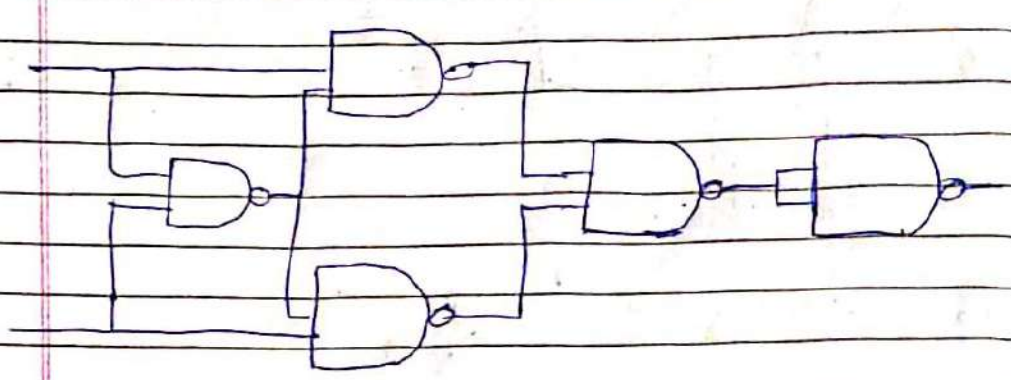
```
module tb_gate;
  reg A,B;
  wire Y;
//print
//Ayushman Pranav E21CSEU0245
    gate1 a1(.a(A),.b(B),.y(Y));
```

```
  initial
    begin
      A=1; B=1; #5;
      $display("A B   Result");
      $display("%b %b     %b",A,B,Y);
      A=1; B=0; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=1; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=0; #5;
      $display("%b %b     %b",A,B,Y);
    end

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

endmodule
```

Design.sv

```
module
  gate1 (input a,b, output y);
  assign y = ~(~(a&(~(a&b)))&~(b&~(a&b)));
endmodule
```

6) XNOR using NAND



$$= \overline{\overline{A\overline{AB}}\;\overline{B\overline{AB}}}$$

$$= \overline{A\cdot\overline{AB} + B\overline{AB}}$$

$$= \overline{A(\bar{A}+\bar{B}) + B(\bar{A}+\bar{B})}$$

$$= \overline{A\bar{A} + A\bar{B} + \bar{A}B + B\bar{B}}$$

$$= \overline{A\bar{B} + \bar{A}B} = \overline{A \oplus B}$$

$$= \overline{A\bar{B}}\cdot\overline{\bar{A}B}$$

$$= (A+\bar{B})(\bar{A}+B)$$

$$= A\bar{A} + AB + \bar{A}\bar{B}\;\bar{B}\bar{B}$$

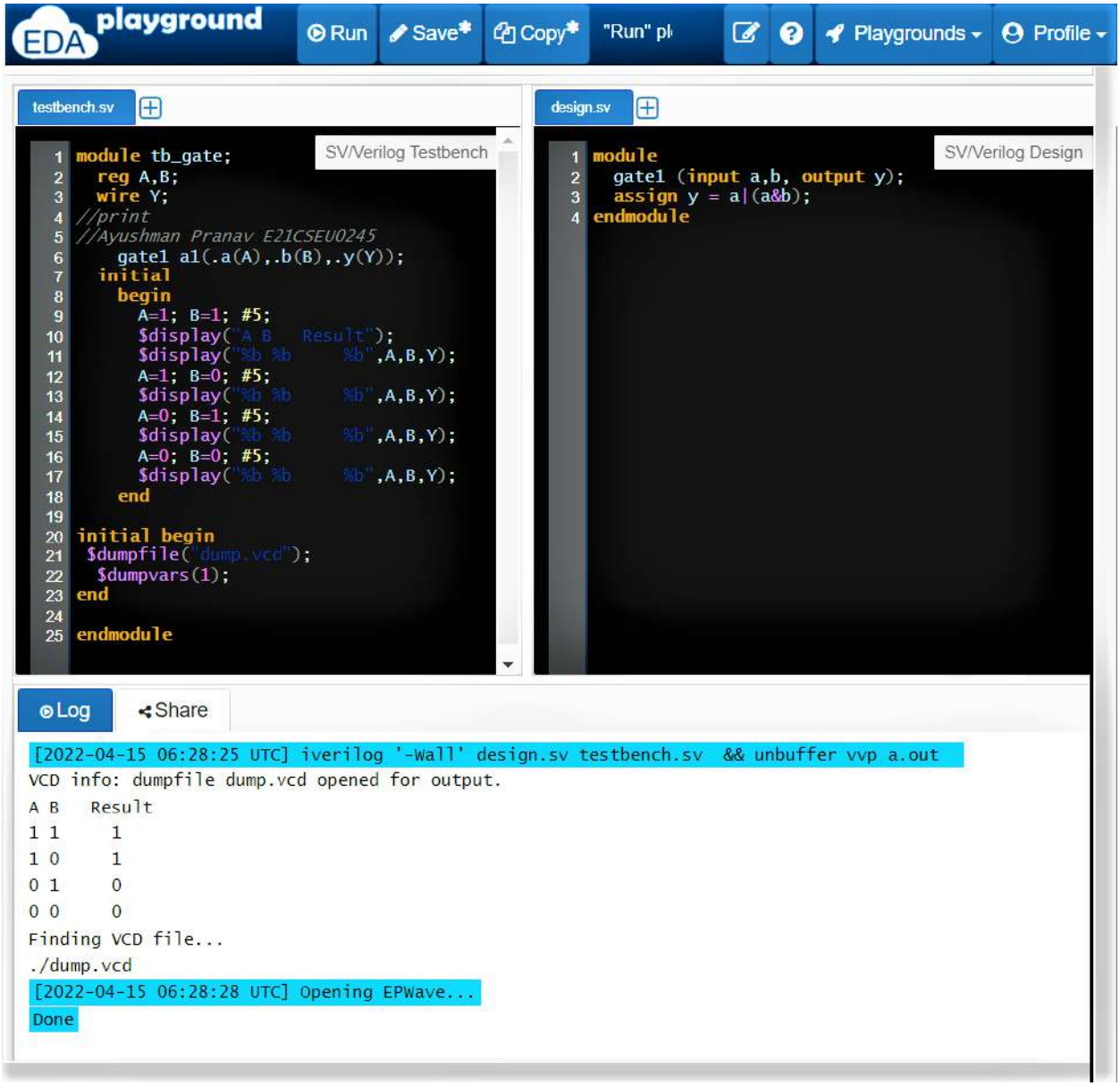
$$= AB + \bar{A}\bar{B}$$

$$= A \odot B$$

A	B	$\bar{A}$	$\bar{B}$	AB	$\overline{AB}$	$A\odot B$ $AB+\bar{A}\bar{B}$	$\overline{A\oplus B}$
1	1	0	0	1	0	1	1
1	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0
0	0	1	1	0	1	1	1

School of Computer Science Engineering and Technology

3. 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?



Testbench.sv

module tb_gate;

```
  reg A,B;
  wire Y;
//print
//Ayushman Pranav E21CSEU0245
    gate1 a1(.a(A),.b(B),.y(Y));
  initial
    begin
      A=1; B=1; #5;
      $display("A B   Result");
      $display("%b %b     %b",A,B,Y);
      A=1; B=0; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=1; #5;
      $display("%b %b     %b",A,B,Y);
      A=0; B=0; #5;
      $display("%b %b     %b",A,B,Y);
    end

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

endmodule
```

School of Computer Science Engineering and Technology

Design.sv

```
module
  gate1 (input a,b, output y);
  assign y = a|(a&b);
endmodule
```

Am-2) Absorbtion law

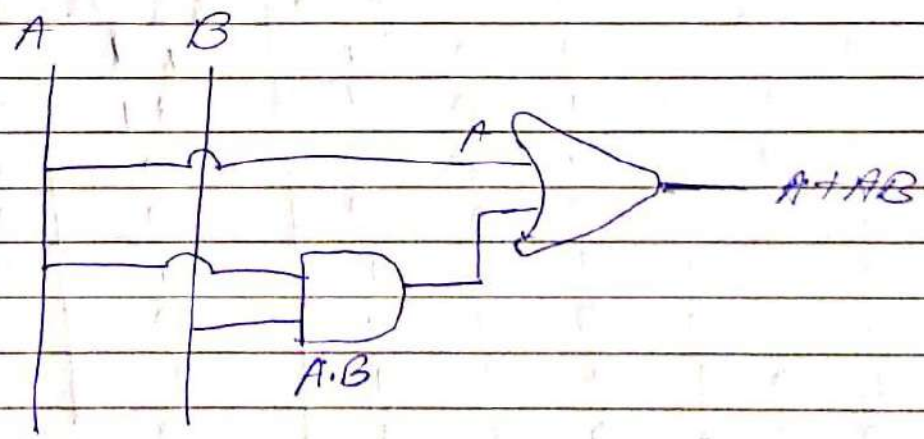
$= A + AB = A$

$= A(1+B)$

$= A \cdot 1$

$= A$

A	B	AB	A+AB
1	1	1	1
1	0	0	1
0	1	0	0
0	0	0	0



4. 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?

School of Computer Science Engineering and Technology

EDA playground | Run | Save | Copy | Playgrounds | Profile

testbench.sv — SV/Verilog Testbench

```sv
module tb_gate();
  reg A,B,C;
  wire Y;
//print
//Ayushman Pranav E21cseu0245
  gate1 a1(.a(A),.b(B),.c(C),.y(Y));
  initial begin
    A=1;B=1;C=1;#5;
    $display("A B C Result");
        $display("%b %b %b   %b",A,B,C,Y);
  A=1;B=1;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=1;B=0;C=1;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=1;B=0;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=1;C=1;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=1;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=0;C=1;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=0;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);


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

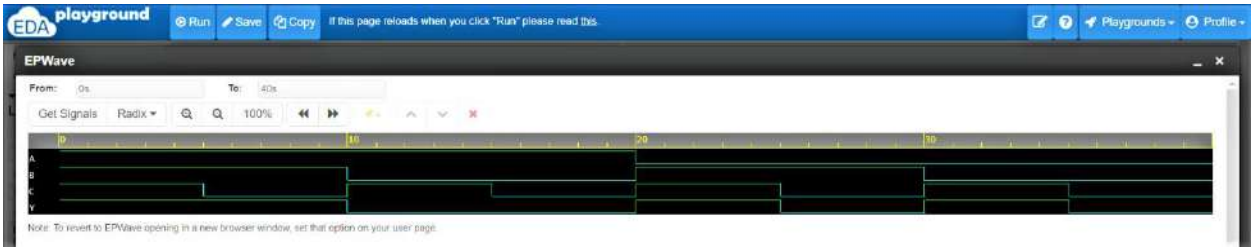
design.sv — SV/Verilog Design

```sv
module
  gate1 (input a,b,c, output y);
  assign y = (a&b)|(~a&c);
endmodule
```

Log | Share

```
[2022-04-15 13:04:18 UTC] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dump.vcd opened for output.
A B C Result
1 1 1   1
1 1 0   1
1 0 1   0
1 0 0   0
0 1 1   1
0 1 0   0
0 0 1   1
0 0 0   0
Finding VCD file...
./dump.vcd
[2022-04-15 13:04:20 UTC] Opening EPWave...
Done
```

School of Computer Science Engineering and Technology



Testbench.sv

```
module tb_gate();
  reg A,B,C;
  wire Y;
//print
//Ayushman Pranav E21cseu0245
  gate1 a1(.a(A),.b(B),.c(C),.y(Y));
  initial begin
    A=1;B=1;C=1;#5;
    $display("A B C Result");
        $display("%b %b %b   %b",A,B,C,Y);
  A=1;B=1;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=1;B=0;C=1;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=1;B=0;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=1;C=1;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=1;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=0;C=1;#5;
    $display("%b %b %b   %b",A,B,C,Y);
```

```
   A=0;B=0;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);



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

Design.sv

```
module
  gate1 (input a,b,c, output y);
  assign y = (a&b)|(~a&c);
endmodule
```

Aim-3) <u>Transposition law</u>

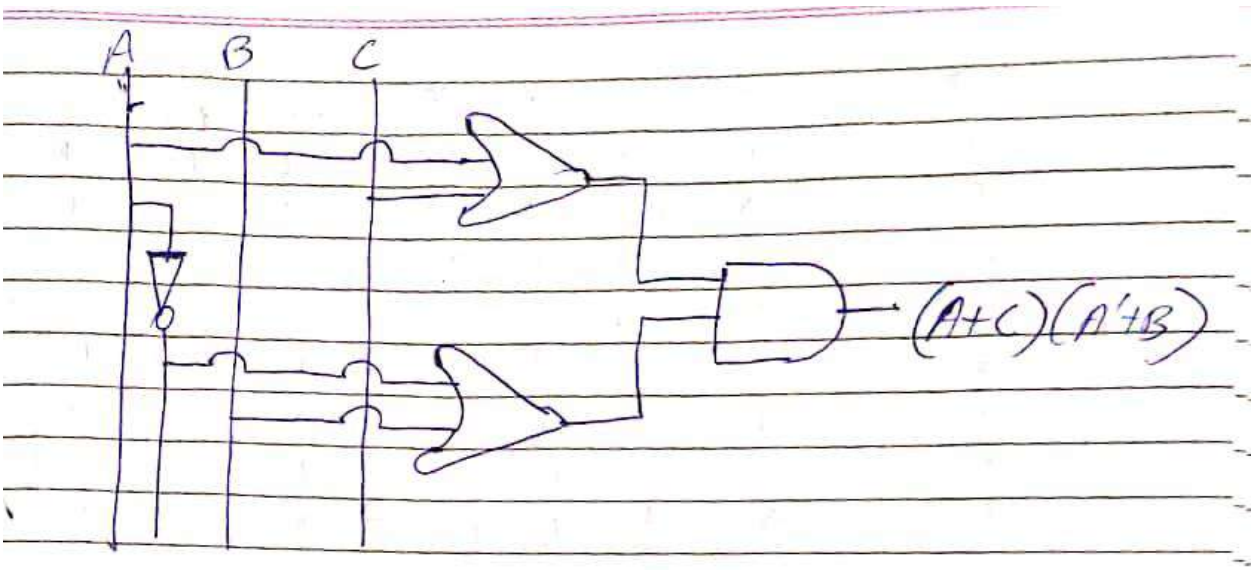
$$AB + A'C = (A+C)(A'+B)$$
$$= AA' + AB + CA' + CB$$
$$= AB + CA' + CB$$
$$= \underbrace{AB + BC}_{\text{Consensus law}} + A'C$$

Consensus law $= AB + A'C$

A	B	C	A+C	A'	A'+B	(A+C)(A'+B)	AB	A'C	AB+A'C
1	1	1	1	0	1	1	1	0	1
1	1	0	1	0	1	1	1	0	1
1	0	1	1	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0
0	1	1	1	1	1	1	0	1	1
0	1	0	0	1	1	0	0	0	0
0	0	1	1	1	1	1	0	1	1
0	0	0	0	1	1	0	0	0	0

logically equivalent

School of Computer Science Engineering and Technology



5. 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?

EDA playground | Run | Save | Copy | Playgrounds | Profile

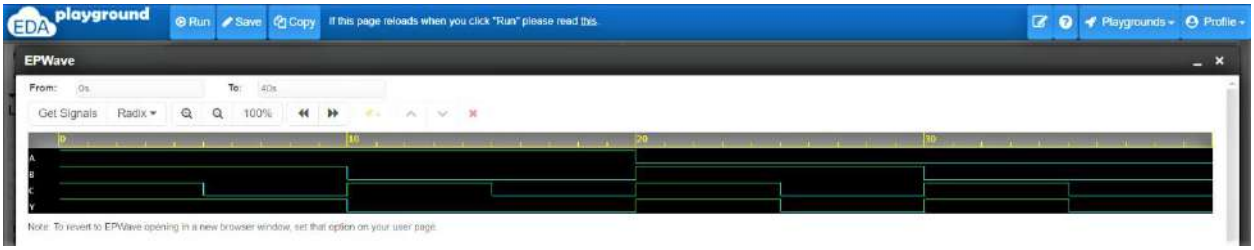
testbench.sv — SV/Verilog Testbench

```
module tb_gate();
  reg A,B,C;
  wire Y;
//print
//Ayushman Pranav E21cseu0245
  gate1 a1(.a(A),.b(B),.c(C),.y(Y));
  initial begin
    A=1;B=1;C=1;#5;
    $display("A B C Result");
        $display("%b %b %b   %b",A,B,C,Y);
  A=1;B=1;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=1;B=0;C=1;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=1;B=0;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=1;C=1;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=1;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=0;C=1;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=0;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);


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

design.sv — SV/Verilog Design

```
module
  gate1 (input a,b,c, output y);
  assign y = (a&b)|(~a&c);
endmodule
```

```
Log   Share
[2022-04-15 13:04:18 UTC] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dump.vcd opened for output.
A B C Result
1 1 1   1
1 1 0   1
1 0 1   0
1 0 0   0
0 1 1   1
0 1 0   0
0 0 1   1
0 0 0   0
Finding VCD file...
./dump.vcd
[2022-04-15 13:04:20 UTC] Opening EPWave...
Done
```



Testbench.sv

```
module tb_gate();
  reg A,B,C;
  wire Y;
//print
//Ayushman Pranav E21cseu0245
  gate1 a1(.a(A),.b(B),.c(C),.y(Y));
  initial begin
    A=1;B=1;C=1;#5;
    $display("A B C Result");
```

```
      $display("%b %b %b   %b",A,B,C,Y);
  A=1;B=1;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=1;B=0;C=1;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=1;B=0;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=1;C=1;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=1;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=0;C=1;#5;
    $display("%b %b %b   %b",A,B,C,Y);
   A=0;B=0;C=0;#5;
    $display("%b %b %b   %b",A,B,C,Y);



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

Design.sv

```
module
  gate1 (input a,b,c, output y);
```

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

Ans-4) Consensus law

$= AB + A'C + BC = AB + A'C$

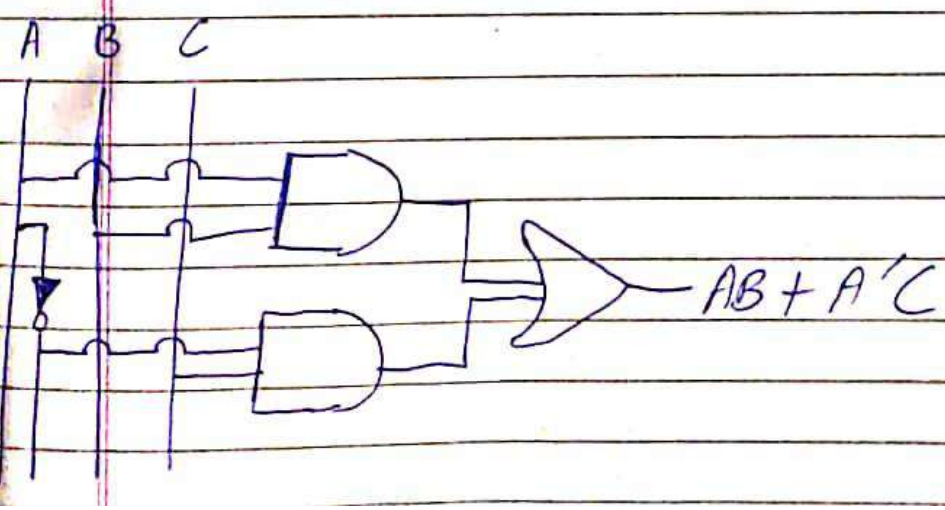
$= AB + A'C + BC(A+A')$

$= AB + A'C + BCA + BCA'$

$= ABC(1+C) + A'C(1+B)$

$= AB + A'C$ $\quad 1 \leftarrow$

Negation Law



A	B	C	$\bar{A}$	$\bar{B}$	$\bar{C}$	AB	$A'C$	BC	$AB+A'C$	$AB+A'C+BC$
1	1	1	0	0	0	1	0	1	1	1
1	1	0	0	0	1	1	0	0	1	1
1	0	1	0	1	0	0	0	0	0	0
1	0	0	0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
0	1	0	1	0	1	0	0	0	0	0
0	0	1	1	1	0	0	1	0	1	1
0	0	0	1	1	1	0	0	0	0	0

logically equivalent