DIGITAL LOGIC DESIGN LAB (EET1211)

LAB I: Examine the Operation of Logic Gates Using HDL

Siksha ‘O’ Anusandhan Deemed to be University, Bhubaneswar

Branch:		**Section:**	
S. No.	**Name**	**Registration No.**	**Signature**
	Deepak Pattnayak	**1941012112**	Deepak Pattnayak

Marks: ______/10

Remarks:

Teacher’s Signature

I. OBJECTIVE:

1. Investigation of the logic behavior of various gates using HDL:

a) 7400 quadruple two-input NAND gates

b) 7402 quadruple two-input NOR gates

c) 7404 hex inverters

d) 7408 quadruple two-input AND gates

e) 7432 quadruple two-input OR gates

f) 7486 quadruple two-input XOR gates

2. Using a single 7400 IC, connect and implement a circuit using HDL that produces

a) An inverter.

b) A two-input AND.

c) A two-input OR.

d) A two-input XOR.

3. Construct & record the output of circuit using HDL that implements the Boolean function:

F=A (B+C)

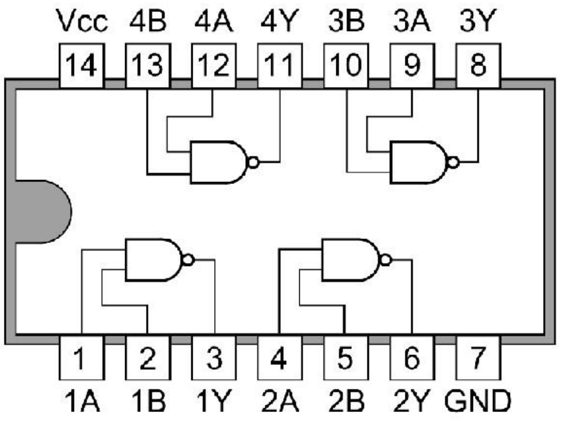
a) Construct the circuit using Logic gates & verify the truth table.

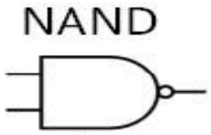
b) Construct the circuit using NAND gates only & verify the truth table.

II. PRE-LAB

1. Draw the IC diagram & obtain truth tables for obj. 1-

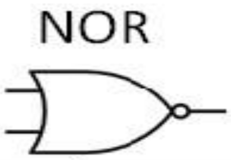
a. 7400 2 Input NAND Gate



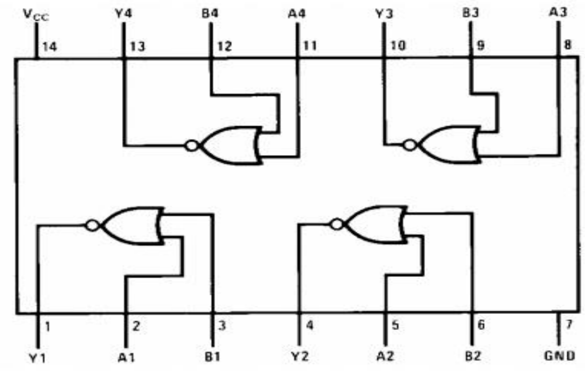


INPUT		OUTPUT
A	B	
0	0	1
1	0	1
0	1	1
1	1	0

b. 7402 2 Input NOR Gate

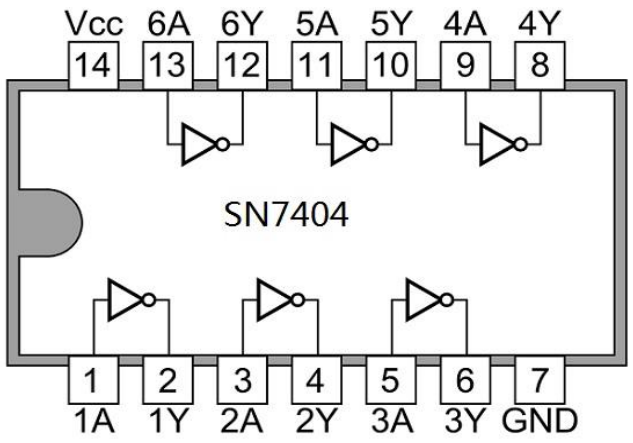


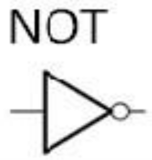
Connection Diagram



INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	0

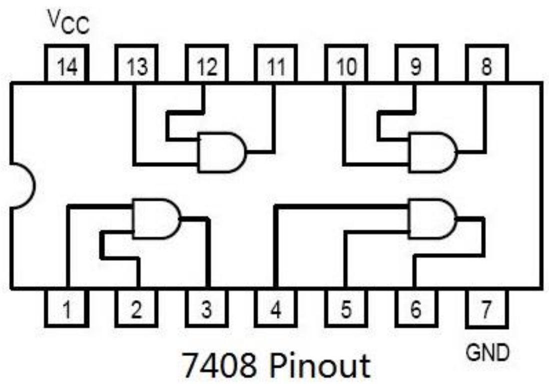
c. 7404 Hex Inverter

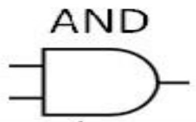




INPUT	OUTPUT
A	
0	1
1	0

d. 7408 2 Input AND Gate

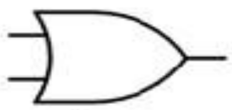


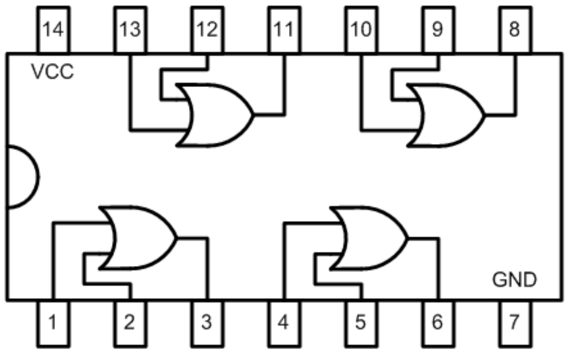


INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1

e. 7432 2 Input OR GATE

OR





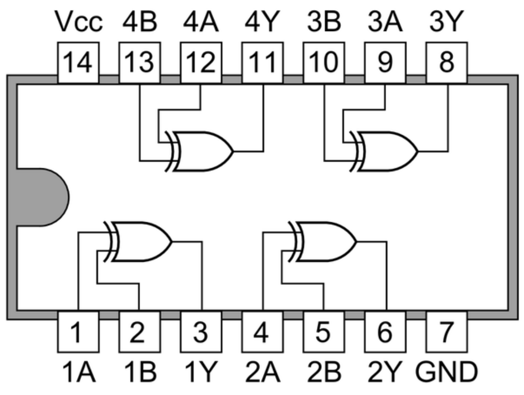
7432 Quad 2 Input OR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	1

f. 7486 2 Input XOR

XOR

7486 Quad 2-input ExOR Gates

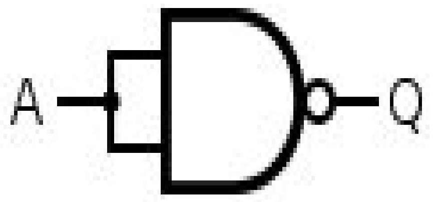


INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	0

2. Draw the circuit diagram & obtain truth tables for obj. 2 & 3-

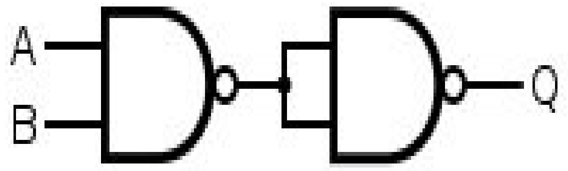
a) An inverter.

Truth Table



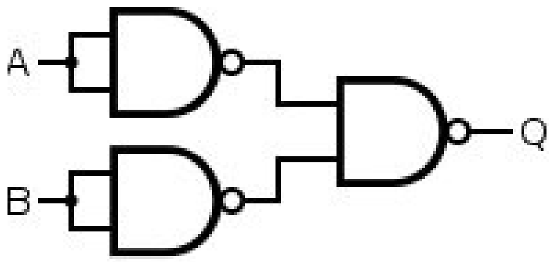
A	$Q=\overline{A.A}$
0	1
1	0

b) A two-input AND.



Truth Table			
A	B	$C=\overline{A.B}$	$Q=\overline{C.C}$
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

c) A two-input OR.

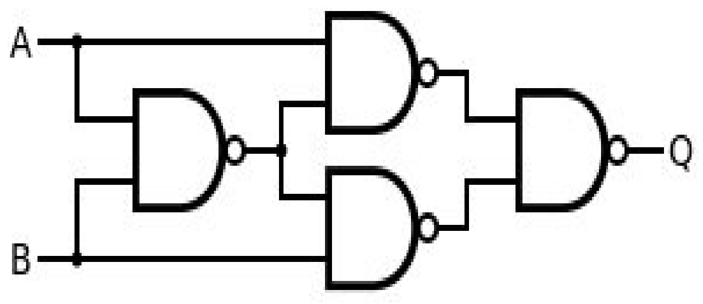


Truth Table				
A	B	$C=\overline{A.A}$	$D=\overline{B.B}$	$Q=\overline{C.D}$
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

A	B	$C=\overline{A.B}$	$D=\overline{A.C}$	$E=\overline{B.C}$	$Q=\overline{D.E}$
0	0	1	1	1	0
0	1	1	1	0	1

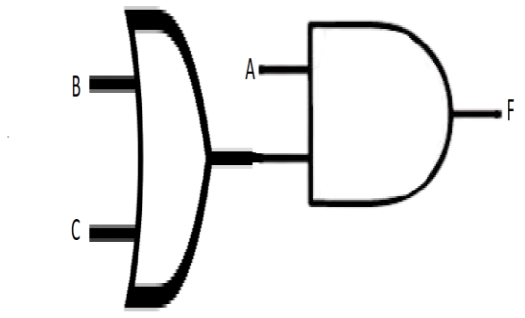
d) A two-input XOR.

1	0	1	0	1	1
1	1	0	1	1	0



<u>3</u>

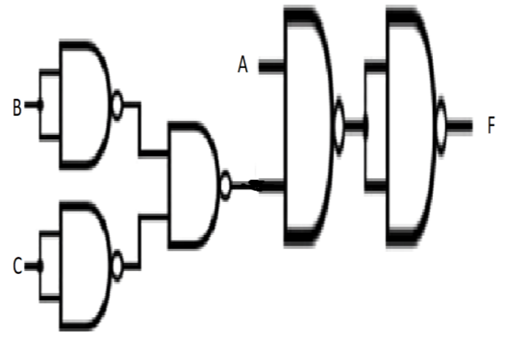
a) Construct the circuit using Logic gates & verify the truth table.



Truth Table				
A	B	C	B+C	F=A(B+C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

b) Construct the circuit using NAND gates only & verify the truth table.

Truth Table							
A	B	C	$X=\overline{B.B}$	$Y=\overline{C.C}$	$Z=\overline{X.Y}$	$P=\overline{A.Z}$	$F=\overline{P.P}$



0	0	0	1	1	0	1	0
0	0	1	1	0	1	1	0
0	1	0	0	1	1	1	0
0	1	1	0	0	1	1	0
1	0	0	1	1	0	1	0
1	0	1	1	0	1	0	1
1	1	0	0	1	1	0	1
1	1	1	0	0	1	0	1

III. LAB:

HDL Program:

1. a) 7400 quadruple two-input NAND gate

design.sv

```
`default_nettype none


module input_2_nand (
   input A,B,
   output X
);

  nand(X,A,B); //GATE LEVEL
  assign X = !(A && B); //DATAFLOW


endmodule
```

testbench.sv

```
`default_nettype none

module nand_2input;

   reg i_a,i_b;
   wire out_x;

  input_2_nand h_dut(i_a,i_b,out_x);

           // UPDATED UNIT TEST //

   initial
      begin
         $dumpfile("dump.vcd");
         $dumpvars(0, h_dut);


        $display("TESTING nand example");
         #1
         i_a <= 0;
         i_b <= 0;
         #1

         #1
         i_a <= 0;
         i_b <= 1;
         #1

         #1
         i_a <= 1;
         i_b <= 0;
         #1

         #1
         i_a <= 1;
         i_b <= 1;
         #1

         $finish();
      end

endmodule
```
LINK: https://www.edaplayground.com/x/TRFA

b) 7402 quadruple two-input NOR gate

design.sv

```
`default_nettype none


module input_2_nor (
  input A,B,
  output X
);

 nor(X,A,B); //GATE LEVEL
 assign X = !(A || B); //DATAFLOW

endmodule
```

testbench.sv

```
`default_nettype none

module nor_2input;

  reg i_a,i_b;
  wire out_x;

 input_2_nor h_dut(i_a,i_b,out_x);

       // UPDATED UNIT TEST //
```

```
    initial
        begin
            $dumpfile("dump.vcd");
            $dumpvars(0, h_dut);


          $display("TESTING nor example");
            #1
            i_a <= 0;
            i_b <= 0;
            #1

            #1
            i_a <= 0;
            i_b <= 1;
            #1

            #1
            i_a <= 1;
            i_b <= 0;
            #1

            #1
            i_a <= 1;
            i_b <= 1;
            #1

            $finish();
        end

endmodule
```

LINK: https://www.edaplayground.com/x/6rdY

c) 7404 hex inverters

design.sv

```
`default_nettype none


module hex_inverter (
    input A,
    output X
);
```

```
  not(X,A); //GATE LEVEL
  assign X = ~A; //DATAFLOW

endmodule
```

testbench.sv

```
`default_nettype none

module inverter_x;

    reg i_a;
    wire out_x;

  hex_inverter h_dut(i_a,out_x);

           // UPDATED UNIT TEST //

    initial
        begin
            $dumpfile("dump.vcd");
            $dumpvars(0, h_dut);


          $display("TESTING hex-inverter example");
            #1
            i_a <= 0;
            #1

            #1
            i_a <= 1;
            #1
            $finish();
        end

endmodule
```

LINK: https://www.edaplayground.com/x/sALM

d) 7408 quadruple two-input AND gates

design.sv

```
`default_nettype none


module input_2_and (
   input A,B,
   output X
);

  and(X,A,B); //GATE LEVEL
  assign X = (A && B); //DATAFLOW

endmodule
```

testbench.sv

```
`default_nettype none

module and_2input;

   reg i_a,i_b;
   wire out_x;

  input_2_and h_dut(i_a,i_b,out_x);

        // UPDATED UNIT TEST //

   initial
      begin
         $dumpfile("dump.vcd");
         $dumpvars(0, h_dut);


        $display("TESTING and example");
         #1
         i_a <= 0;
         i_b <= 0;
         #1

         #1
         i_a <= 0;
         i_b <= 1;
         #1

         #1
         i_a <= 1;
         i_b <= 0;
         #1

         #1
         i_a <= 1;
         i_b <= 1;
         #1

         $finish();
      end

endmodule
```

LINK: https://www.edaplayground.com/x/dg3V

e) 7432 quadruple two-input OR gates

design.sv

```
`default_nettype none


module input_2_or (
  input A,B,
  output X
);

 or(X,A,B); //GATE LEVEL
 assign X = (A || B); //DATAFLOW

endmodule
```

testbench.sv

```
`default_nettype none

module or_2input;

  reg i_a,i_b;
  wire out_x;

 input_2_or h_dut(i_a,i_b,out_x);

       // UPDATED UNIT TEST //

  initial
    begin
```

```
          $dumpfile("dump.vcd");
          $dumpvars(0, h_dut);


         $display("TESTING or example");
          #1
          i_a <= 0;
          i_b <= 0;
          #1

          #1
          i_a <= 0;
          i_b <= 1;
          #1

          #1
          i_a <= 1;
          i_b <= 0;
          #1

          #1
          i_a <= 1;
          i_b <= 1;
          #1

          $finish();
        end

endmodule
```

LINK: https://www.edaplayground.com/x/mr7G

f) 7486 quadruple two-input XOR gates

design.sv

```
`default_nettype none

module input_2_xor (
   input A,B,
   output X
);

  xor(X,A,B); //GATE LEVEL
  assign X = ~A&&B || A&&~B; //DATAFLOW
```

```
endmodule
```

testbench.sv

```
`default_nettype none
module xor_2input;

  reg i_a,i_b;
  wire out_x;

 input_2_xor h_dut(i_a,i_b,out_x);

      // UPDATED UNIT TEST //

  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(0, h_dut);


     $display("TESTING xor example");
      #1
      i_a <= 0;
      i_b <= 0;
      #1

      #1
      i_a <= 0;
      i_b <= 1;
      #1

      #1
      i_a <= 1;
      i_b <= 0;
      #1

      #1
      i_a <= 1;
      i_b <= 1;
      #1

      $finish();
    end

endmodule
```

LINK: https://www.edaplayground.com/x/tmfW

2.a) Using a single 7400 IC as an Inverter

design.sv

```
`default_nettype none

module NAND_Inverter(
  input A,
  output X
);
  nand(X,A,A) ; //GATE LEVEL
  assign X = !(A && A) ; //DATAFLOW

endmodule
```

testbench.sv

```
`default_nettype none

module nand_inverter ;
  reg i_a ;
  wire x ;

  NAND_Inverter calc(i_a, x) ;

           initial
      begin
        $dumpfile("dump.vcd") ;
        $dumpvars(0 , calc) ;

        $display("Using a 7400 IC as an inverter") ;
        #1
        i_a<=0 ;
        #1

        #1
        i_a<=1 ;
        #1

        $finish() ;
      end
 endmodule
/*Inverter by using a single 7400 IC*/
```

LINK: https://www.edaplayground.com/x/9LZD

b.) Using a single 7400 IC as two-input AND gate.

design.sv

```
`default_nettype none

module Nand_as_and(
   input A,B,
   output X
);
  wire w1,w2;
  //GATE LEVEL MODELLING
  nand g1(w1,A,B);
  nand g2(X,w1,w1);
  //DATA FLOW MODELLING
  //assign X= !((!(A && B)) && (!(A && B)));

endmodule
```

testbench.sv

```
`default_nettype none

module nand_as_and;

   reg i_a,i_b;
   wire out_x;

  Nand_as_and h_dut(i_a,i_b,out_x);

           // UPDATED UNIT TEST //

   initial
      begin
         $dumpfile("dump.vcd");
         $dumpvars(0, h_dut);


        $display("TESTING: A two-input AND, Using a single 7400 IC");
        $monitor(i_a,i_b,out_x);
         #1
         i_a <= 0;
         i_b <= 0;
         #1

         #1
         i_a <= 0;
         i_b <= 1;
         #1

         #1
         i_a <= 1;
         i_b <= 0;
         #1

         #1
         i_a <= 1;
         i_b <= 1;
         #1

         $finish();
      end

endmodule
```

LINK: https://www.edaplayground.com/x/72LX

c.) Using a single 7400 IC as two-input OR gate.

design.sv

```
`default_nettype none

module Nand_as_or (
  input A,B,
  output X
);
 wire w1,w2;
 //GATE LEVEL MODELLING
 nand g1(w1,A,A);
 nand g2(w2,B,B);
 nand g3(X,w1,w2);
 //DATA FLOW MODELLING
 //assign w1 = !(A && A);
 //assign w2 = !(B && B);
 //assign X = !(w1 && w2);

endmodule
```

testbench.sv

```
`default_nettype none

module nand_as_or;
```

```
    reg i_a,i_b;
    wire out_x;

  Nand_as_or h_dut(i_a,i_b,out_x);

            // UPDATED UNIT TEST //

    initial
        begin
          $dumpfile("dump.vcd");
          $dumpvars(0, h_dut);


        $display("TESTING: A two-input OR, Using a single 7400 IC");
        $monitor(i_a,i_b,out_x);
          #1
          i_a <= 0;
          i_b <= 0;
          #1

          #1
          i_a <= 0;
          i_b <= 1;
          #1

          #1
          i_a <= 1;
          i_b <= 0;
          #1

          #1
          i_a <= 1;
          i_b <= 1;
          #1

          $finish();
        end

endmodule
```

LINK: https://www.edaplayground.com/x/kVFj

d.) Using a single 7400 IC as two-input XOR gate.

design.sv

```
`default_nettype none
```

```
module Nand_as_xor (
    input A,B,
    output X
);
  wire C,D,E;
  //GATE LEVEL
  nand(C,A,B);
  nand(D,A,C);
  nand(E,C,B);
  nand(X,D,E);
  //DATA FLOW MODELLING
 // wire C=!(A&&B);
 // wire D=!(A&&C);
 // wire E=!(C&&B);
 // assign X=!(D&&E);
endmodule
```

testbench.sv

```
`default_nettype none

module nand_as_xor;

    reg i_a ,i_b;
    wire out_x;

  Nand_as_xor h_dut(i_a,i_b,out_x);

            // UPDATED UNIT TEST //

    initial
        begin
            $dumpfile("dump.vcd");
            $dumpvars(0, h_dut);


          $display("TESTING: Nand as XOR using single 7400IC");


             #1
            i_a <= 0;
            i_b <= 0;


            #1
            i_a <= 0;
            i_b <= 1;
            #1
            i_a <= 1;
            i_b <= 0;


            #1
            i_a <= 1;
            i_b <= 1;
            #1
            $finish();
        end

endmodule
```

LINK: https://www.edaplayground.com/x/n7au

3. Construct & record the output of circuit using HDL that implements the Boolean function:

F=A (B+C)

a) Construct the circuit using Logic gates & verify the truth table.

b) Construct the circuit using NAND gates only & verify the truth table.

design.sv

```
`default_nettype none
module obj_3(
  input A, B, C,
  output F
);
  wire w1,w2;

  //A. Using Logic Gates

  //GATE LEVEL
  or g1(w1,B,C);
  and g2(F,A,w1);
  //DATA FLOW
  //assign F = (A && (B || C)) ;
  //B. Using NAND GATES ONLY

  //B or C
  //nand g1(w1,B,B);
  //nand g2(w2,C,C);
  //nand g3(X,w1,w2);

  //A and (B or C)
  //nand g4(w3,A,X);
  //nand g5(F,w3,w3);
  //DATA FLOW
  //assign F = (A && (B || C)) ;
endmodule
```

testbench.sv

```
`default_nettype none

module Obj_3 ;
  reg i_A, i_B, i_C ;
  wire oF;

  obj_3 OUTPUT(i_A, i_B, i_C, oF) ;

        initial
     begin
      $dumpfile("dump.vcd") ;
      $dumpvars(0 , OUTPUT) ;

      $display("HDL program for objective 3, F=A(B+C)") ;
      #1
      i_A<=0 ;
      i_B<=0 ;
      i_C<=0 ;
      #1

      #1
      i_A<=0 ;
      i_B<=0 ;
      i_C<=1 ;
      #1

      #1
      i_A<=0 ;
      i_B<=1 ;
      i_C<=0 ;
      #1
```

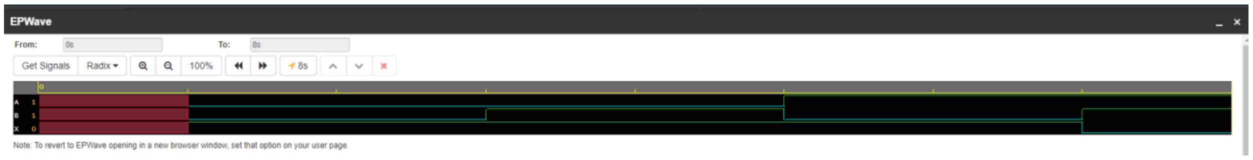
```
        #1
        i_A<=0 ;
        i_B<=1 ;
        i_C<=1 ;
        #1

        #1
        i_A<=1 ;
        i_B<=0 ;
        i_C<=0 ;
        #1

        #1
        i_A<=1 ;
        i_B<=0 ;
        i_C<=1 ;
        #1

        #1
        i_A<=1 ;
        i_B<=1 ;
        i_C<=0 ;
        #1

        #1
        i_A<=1 ;
        i_B<=1 ;
        i_C<=1 ;
        #1

        $finish() ;
    end
endmodule
```

LINK: https://www.edaplayground.com/x/NV4G

Observation:

EP Wave for Obj.1

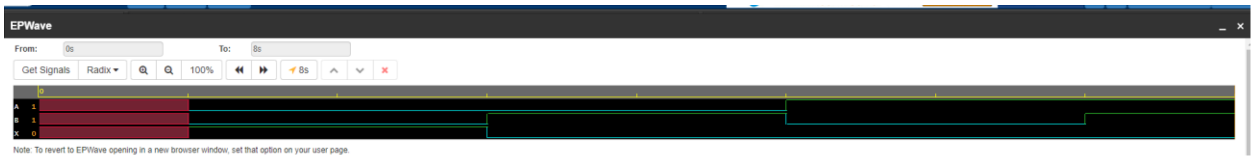
a.NAND Gate



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

The above EP Wave shows the truth table of NAND gate. Here when input A is 1 and B is 1 the output X is 0 which satisfies the working of the Nand Gate.

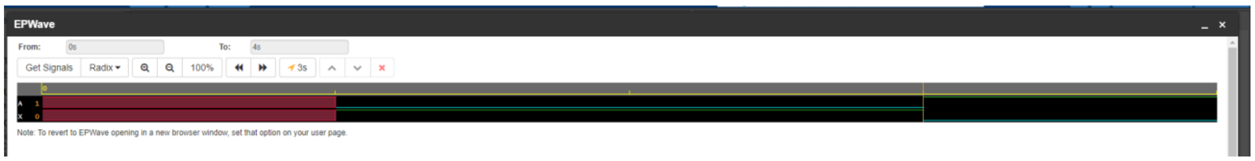
b.NOR Gate



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

The above EP Wave shows the truth table of NOR gate. Here when input A is 1 and B is 1 the output X is 0 which satisfies the working of the NOR Gate.

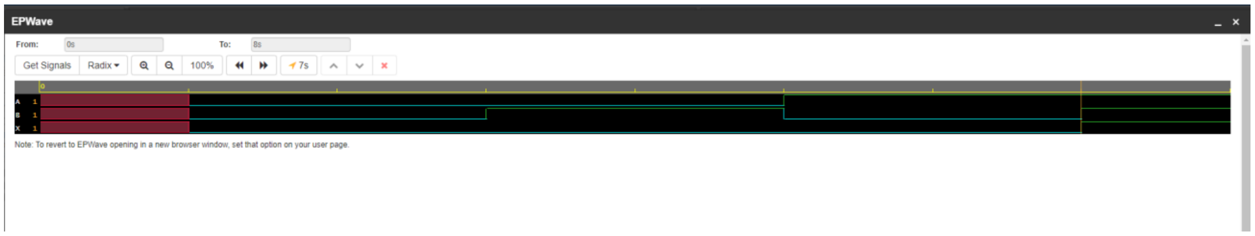
c.Hex Inverter



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

The above EP Wave shows the truth table of NOT gate. Here when input A is 1 the output X is 0 which satisfies the working of the Hex Inverter.

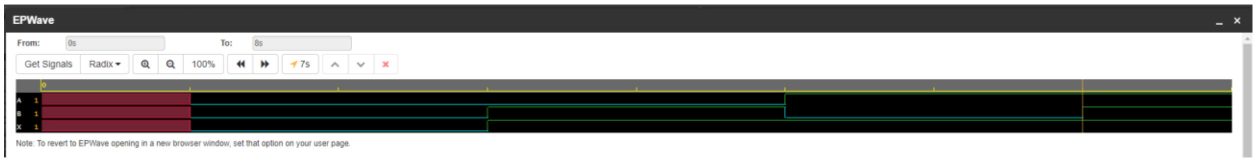
d.AND Gate



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

The above EP Wave shows the truth table of AND gate. Here when input A is 1 and B is 1 the output X is 1 which satisfies the working of the AND Gate.

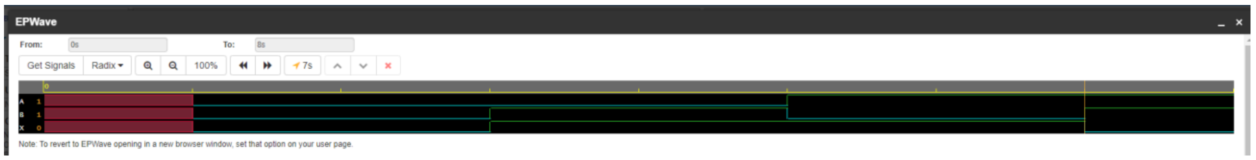
e.OR Gate



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

The above EP Wave shows the truth table of OR gate. Here when input A is 1 and B is 1 the output X 1 is A which satisfies the working of the OR Gate.

f. XOR Gate

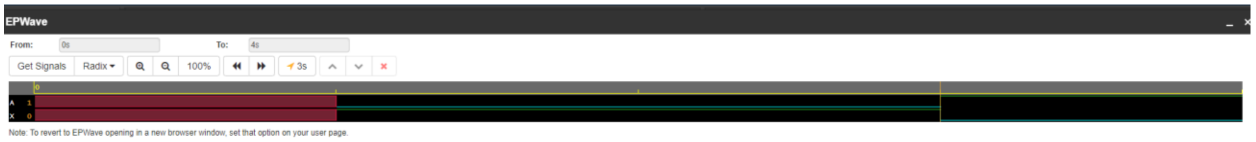


Note: To revert to EPWave opening in a new browser window, set that option on your user page.

The above EP Wave shows the truth table of XOR gate. Here when input A is 1 and B is 1 the output X is 0 which satisfies the working of the XOR Gate.

EP Wave for Obj.2

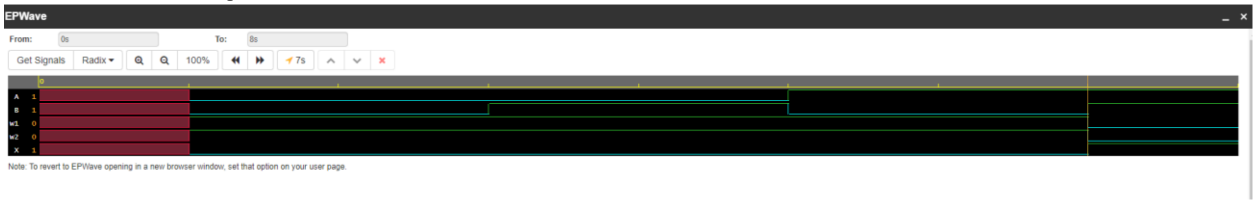
a. An inverter



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

The above EP Wave shows the truth table of Hex Inverter using a NAND gate of 7400 IC, here when input A is 1 and B is 0 which satisfies the working of the Hex Inverter gate.

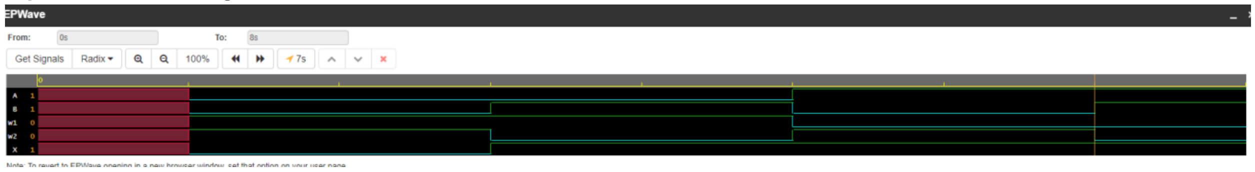
b. A two-input AND



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

The above EP Wave shows the truth table of AND gate using 2 NAND gates of a 7400 IC, here when input A is 1 and B is 1 the output X is 1 which satisfies the working of the AND gate.

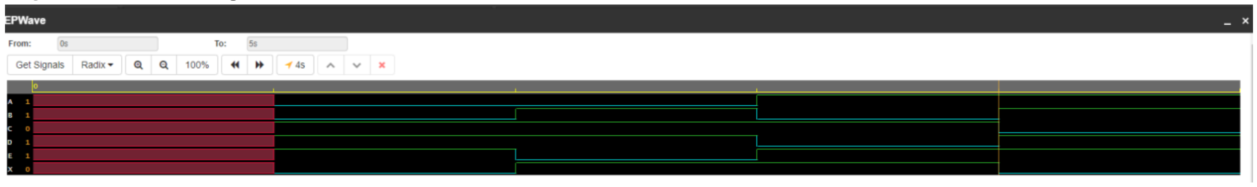
c) A two-input OR.



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

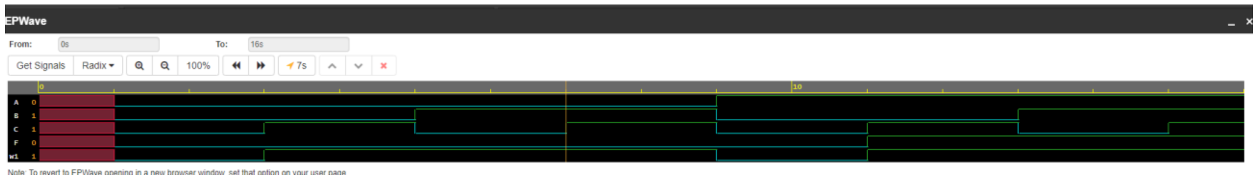
The above EP Wave shows the truth table of OR gate using 3 NAND gates of a 7400 IC, here when input A is 1 and B is 1 the output X is 1 which satisfies the working of the OR gate.

d) A two-input XOR.



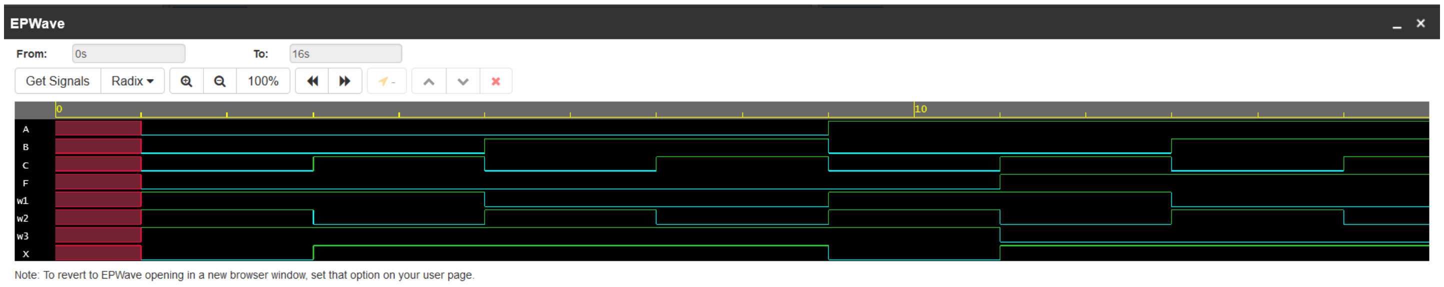
The above EP Wave shows the truth table of XOR gate using 4 NAND gates of a 7400 IC, here when input A is 1 and B is 1 the output X is 1 which satisfies the working of the XOR gate.

EP Wave for Obj.3



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

a. To verify the above function F=A(B+C) we have used 1 or gate and 1 AND gate. The above EP Wave verifies the boolean function, when input A is 0 , B is 1 & C is 1 and output F is 0.



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

b. To verify the above function F=A(B+C) we have used 5 NAND gates. The above EP Wave verifies the boolean function, when input A is 1 , B is 1 & C is 0 and output F is 1.

Conclusion:

Hence, the logical behaviours of all the gates have been verifired using HDL program in **Objective 1**. In **Objective 2**, we have concluded that all other gates such as AND,OR,XOR etc can be produced by using a single **7400IC Nand Gate**. In **Objective 3**, it can be concluded that output F can be constructed using NAND Gate. Thus the final output F has been recorded and truth table is also verified.

IV. POST LAB:

1. What is the voltage range for operation of digital circuits?

Ans: Voltage range for operations of Digital Circuits is 0V – 5V.

2. What is the significance of ground and VCC connection?

Ans: VCC is the higher voltage with respect to ground. VCC is the power input of a device. It may be positive or negative with respect to Ground. Ground is normally at zero volts or the zero voltage point for a power supply and circuit.

3. Which gates are known as universal gates & why?

Ans: NAND and NOR gates are known as universal gates which can be combined to form any other logic gates.

4. **What is the minimum number of NAND gates used to realize an EXOR gate?**

Ans: 4 NAND gates are used to realize an EXOR gate.