**DIGITAL LOGIC DESIGN LAB (EET1211)**

**LAB I: Examine the Operation of Logic Gates Using HDL**

**Siksha ‘O’ Anusandhan Deemed to be University, Bhubaneswar**

|  |  |  |  |
| --- | --- | --- | --- |
| **Branch:** Computer Science & Engineering **Section:** D | | | |
| **S. No.** | **Name** | **Registration No.** | **Signature** |
| 1 | Saswat Mohanty | 1941012407 | **E:\sign.jpg** |

**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:

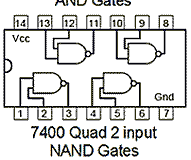
F=A (B+C)

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**

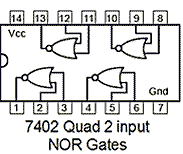
1. **Draw the IC diagram & obtain truth tables for obj. 1-**
2. **7400 quadruple two-input NAND gates**



|  |  |  |
| --- | --- | --- |
| **Truth Table** | | |
| **A** | **B** | **X=A.B** |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

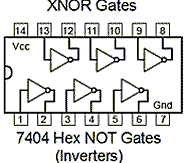
1. **7402 quadruple two-input NOR gates**

|  |  |  |
| --- | --- | --- |
| **Truth Table** | | |
| **A** | **B** | **X=A+B** |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |



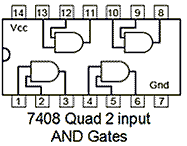
1. **7404 hex inverters**

|  |  |
| --- | --- |
| **Truth Table** | |
| **A** | **X=A** |
| 0 | 1 |
| 1 | 0 |



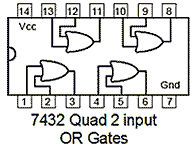
1. **7408 quadruple two-input AND gates**

|  |  |  |
| --- | --- | --- |
| **Truth Table** | | |
| **A** | **B** | ***X= A.B*** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



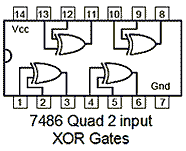
1. **7432 quadruple two-input OR gates**

|  |  |  |
| --- | --- | --- |
| **Truth Table** | | |
| **A** | **B** | ***X= A+B*** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

****

1. **7486 quadruple two-input XOR gates**

|  |  |  |
| --- | --- | --- |
| **Truth Table** | | |
| **A** | **B** | ***X= A* ⊕*B*** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

****

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

**Objective 2:-**

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

1. **An inverter.**

|  |  |
| --- | --- |
| **Truth Table** | |
| **A** | **Q=A.A** |
| 0 | 1 |
| 1 | 0 |



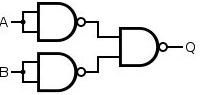
|  |  |  |  |
| --- | --- | --- | --- |
| **Truth Table** | | | |
| **A** | **B** | **C=A.B** | **Q=C.C** |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

1. **A two-input AND.**



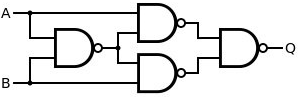
1. **A two-input OR.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Truth Table** | | | | |
| **A** | **B** | **C=A.A** | **D=B.B** | **Q=C.D** |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |



1. **A two-input XOR.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C=A.B** | **D=A.C** | **E=B.C** | **Q=D.E** |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |



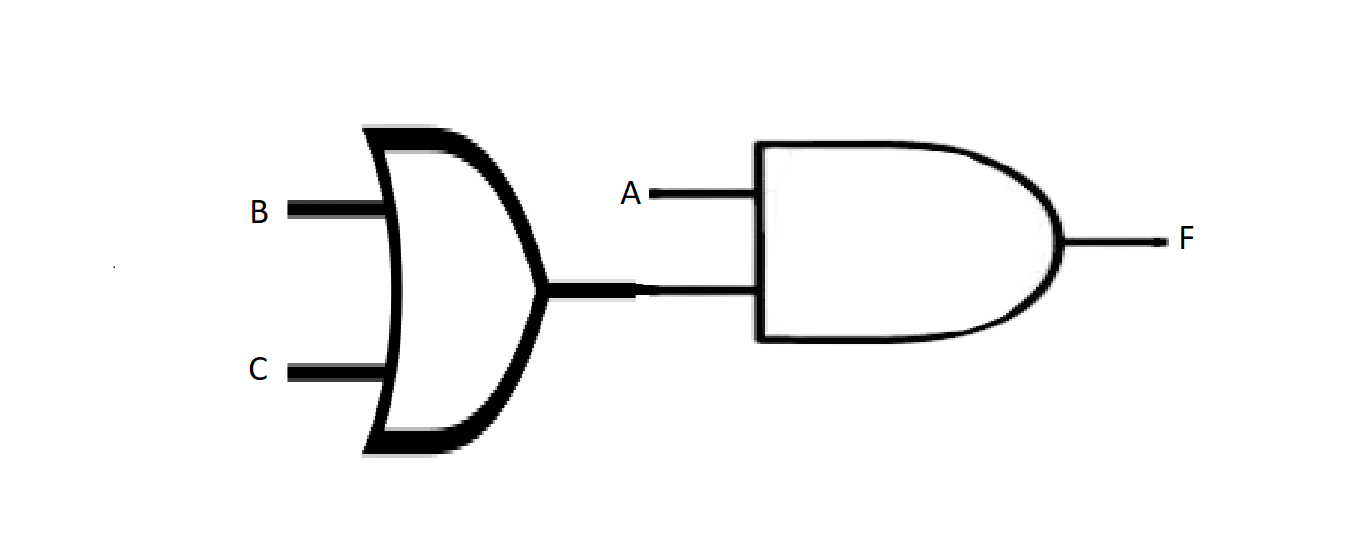
**Objective 3:-**

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

**F=A (B+C)**

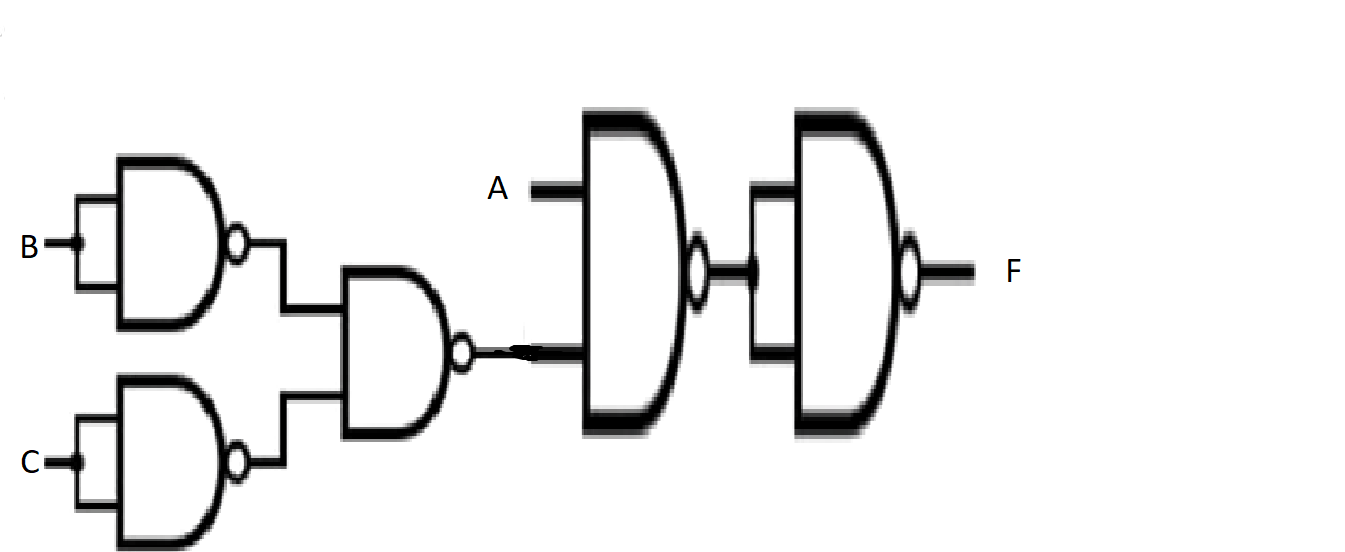
1. **Construct the circuit using Logic gates & verify the truth table.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |



1. **Construct the circuit using NAND gates only & verify the truth table.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Truth Table** | | | | | | | | |
| **A** | **B** | **C** | **X=B.B** | **Y=C.C** | **Z=X.Y** | **P=A.Z** | **F=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. **Investigation of the logic behavior of various gates using HDL:**
2. **7400 quadruple two-input NAND gates**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module mod (*

*input A,*

*input B,*

*output F*

*);*

*// dataflow model*

*assign F= ~(A && B);*

*// gate level model*

*//nand(F,A,B);*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module tb\_mod;*

*reg a, b;*

*wire f;*

*mod h\_dut(a,b,f);*

*initial*

*begin*

*$dumpfile("dump.vcd");*

*$dumpvars(0, h\_dut);*

*$display("TESTING orand example");*

*#1*

*a <= 0;*

*b <= 0;*

*#1*

*#1*

*a <= 0;*

*b <= 1;*

*#1*

*#1*

*a <= 1;*

*b <= 0;*

*#1*

*#1*

*a <= 1;*

*b <= 1;*

*#1*

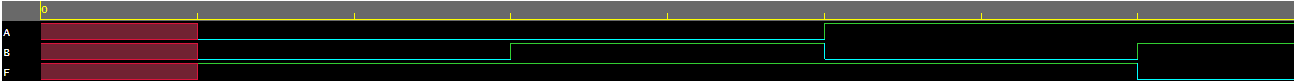
*$finish();*

*end*

*endmodule*

**Link:** <https://www.edaplayground.com/x/MUQ8>

**EP Waveform:-**



**Observation:-**

The following truth table was obtained from the above EP Wave

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **F** |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Hence, we observe that the generated truth table matches to that of a NAND gate.

1. **7402 quadruple two-input NOR gates**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module mod (*

*input A,*

*input B,*

*output F*

*);*

*// dataflow model*

*assign F= ~(A||B);*

*// gate level model*

*//nor(F,A,B);*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module tb\_mod;*

*reg a, b;*

*wire f;*

*mod h\_dut(a,b,f);*

*initial*

*begin*

*$dumpfile("dump.vcd");*

*$dumpvars(0, h\_dut);*

*$display("TESTING orand example");*

*#1*

*a <= 0;*

*b <= 0;*

*#1*

*#1*

*a <= 0;*

*b <= 1;*

*#1*

*#1*

*a <= 1;*

*b <= 0;*

*#1*

*#1*

*a <= 1;*

*b <= 1;*

*#1*

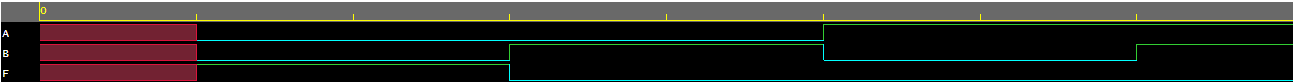
*$finish();*

*end*

*endmodule*

**Link:** <https://www.edaplayground.com/x/CudU>

**EP Waveform:-**



**Observation:-**

The following truth table was obtained from the above EP Wave

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **F** |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Hence, we observe that the generated truth table matches to that of a NOR gate.

1. **7404 hex inverters**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module mod (*

*input A,*

*output F*

*);*

*// dataflow model*

*assign F=~A;*

*// gate level model*

*//not(F,A);*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module tb\_mod;*

*reg i\_a;*

*wire out\_f;*

*mod h\_dut(i\_a,out\_f);*

*initial*

*begin*

*$dumpfile("dump.vcd");*

*$dumpvars(0, h\_dut);*

*$display("TESTING orand example");*

*#1*

*i\_a <= 0;*

*#1*

*#1*

*i\_a <= 1;*

*#1*

*$finish();*

*end*

*endmodule*

**Link:** <https://www.edaplayground.com/x/FbXQ>

**EP Waveform:-**



**Observation:-**

The following truth table was obtained from the above EP Wave

|  |  |
| --- | --- |
| **A** | **F** |
| 0 | 1 |
| 1 | 0 |

Hence, we observe that the generated truth table matches to that of an inverter.

1. **7408 quadruple two-input AND gates**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module mod (*

*input A,*

*input B,*

*output Y*

*);*

*// dataflow model*

*assign Y= A && B;*

*//gate level model*

*//and(Y,A,B);*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module tb\_mod;*

*reg a, b;*

*wire y;*

*mod h\_dut(a,b,y);*

*initial*

*begin*

*$dumpfile("dump.vcd");*

*$dumpvars(0, h\_dut);*

*$display("TESTING orand example");*

*#1*

*a <= 0;*

*b <= 0;*

*#1*

*#1*

*a <= 0;*

*b <= 1;*

*#1*

*#1*

*a <= 1;*

*b <= 0;*

*#1*

*#1*

*a <= 1;*

*b <= 1;*

*#1*

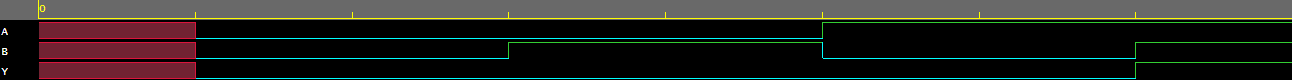
*$finish();*

*end*

*endmodule*

**Link:** <https://www.edaplayground.com/x/j9F6>

**EP Waveform:-**



**Observation:-**

The following truth table was obtained from the above EP Wave

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **F** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Hence, we observe that the generated truth table matches to that of an AND gate.

1. **7432 quadruple two-input OR gates**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module mod (*

*input A,*

*input B,*

*output F*

*);*

*// dataflow model*

*assign F= A || B;*

*// gate level model*

*//or(F,A,B);*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module tb\_mod;*

*reg a, b;*

*wire f;*

*mod h\_dut(a,b,f);*

*initial*

*begin*

*$dumpfile("dump.vcd");*

*$dumpvars(0, h\_dut);*

*$display("TESTING orand example");*

*#1*

*a <= 0;*

*b <= 0;*

*#1*

*#1*

*a <= 0;*

*b <= 1;*

*#1*

*#1*

*a <= 1;*

*b <= 0;*

*#1*

*#1*

*a <= 1;*

*b <= 1;*

*#1*

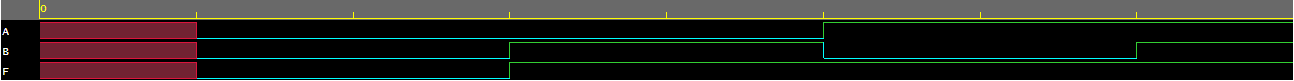
*$finish();*

*end*

*endmodule*

**Link:** <https://www.edaplayground.com/x/et6S>

**EP Waveform:-**



**Observation:-**

The following truth table was obtained from the above EP Wave

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **F** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Hence, we observe that the generated truth table matches to that of a OR gate.

1. **7486 quadruple two-input XOR gates**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module mod (*

*input A,*

*input B,*

*output F*

*);*

*// dataflow model*

*assign F= A ^ B;*

*// gate level model*

*//xor(F,A,B);*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module tb\_mod;*

*reg a, b;*

*wire f;*

*mod h\_dut(a,b,f);*

*initial*

*begin*

*$dumpfile("dump.vcd");*

*$dumpvars(0, h\_dut);*

*$display("TESTING orand example");*

*#1*

*a <= 0;*

*b <= 0;*

*#1*

*#1*

*a <= 0;*

*b <= 1;*

*#1*

*#1*

*a <= 1;*

*b <= 0;*

*#1*

*#1*

*a <= 1;*

*b <= 1;*

*#1*

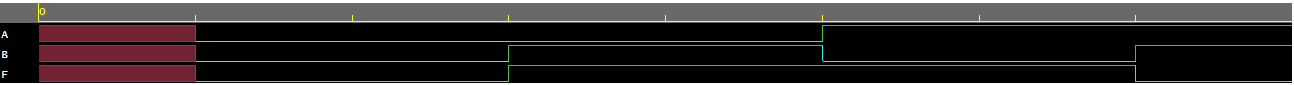
*$finish();*

*end*

*endmodule*

**Link:** <https://www.edaplayground.com/x/6uk6>

**EP Waveform:-**



**Observation:-**

The following truth table was obtained from the above EP Wave

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **F** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Hence, we observe that the generated truth table matches to that of a XOR gate.

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

**Code:-**

**design.sv:**

*`default\_nettype none*

*module mod (*

*input A,*

*output F*

*);*

*wire Z;*

*// dataflow model*

*assign Z=A && A;*

*assign F=~Z;*

*// gate level model*

*//and(Z,A,A);*

*//not(F,Z);*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module tb\_mod;*

*reg i\_a;*

*wire out\_f;*

*mod h\_dut(i\_a,out\_f);*

*initial*

*begin*

*$dumpfile("dump.vcd");*

*$dumpvars(0, h\_dut);*

*$display("TESTING orand example");*

*#1*

*i\_a <= 0;*

*#1*

*#1*

*i\_a <= 1;*

*#1*

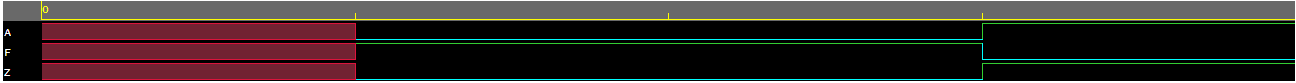
*$finish();*

*end*

*endmodule*

**Link:** <https://www.edaplayground.com/x/b6dj>

**EP Waveform:-**



**Observation:-**

The following truth table was obtained from the above EP Wave

|  |  |
| --- | --- |
| **A** | **F** |
| 0 | 1 |
| 1 | 0 |

Hence, we observe that the generated truth table matches to that of a inverter gate.

1. **A two-input AND.**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module mod (*

*input A,*

*input B,*

*output Y*

*);*

*wire C;*

*wire D;*

*wire E;*

*// dataflow model*

*//assign C=A&&B;*

*//assign D=~C;*

*//assign E=D&&D;*

*//assign Y=~E;*

*// gate level model*

*and(C,A,B);*

*not(D,C);*

*and(E,D,D);*

*not(Y,E);*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module tb\_mod;*

*reg a, b;*

*wire y;*

*mod h\_dut(a,b,y);*

*initial*

*begin*

*$dumpfile("dump.vcd");*

*$dumpvars(0, h\_dut);*

*$display("TESTING orand example");*

*#1*

*a <= 0;*

*b <= 0;*

*#1*

*#1*

*a <= 0;*

*b <= 1;*

*#1*

*#1*

*a <= 1;*

*b <= 0;*

*#1*

*#1*

*a <= 1;*

*b <= 1;*

*#1*

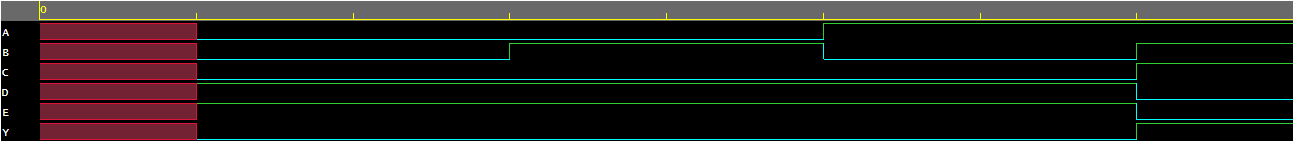
*$finish();*

*end*

*endmodule*

**Link:** <https://www.edaplayground.com/x/Ea45>

**EP Waveform:-**



**Observation:-**

The following truth table was obtained from the above EP Wave

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Y** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Hence, we observe that the generated truth table matches to that of an AND gate.

1. **A two-input OR.**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module mod (*

*input A,*

*input B,*

*output F*

*);*

*wire C;*

*wire D;*

*wire X;*

*wire Y;*

*wire Z;*

*wire P;*

*//dataflow model*

*assign C=A&&A;*

*assign X=~C;*

*assign D=B&&B;*

*assign Y=~D;*

*assign P=Y&&X;*

*assign F=~P;*

*// gate level model*

*//and(C,A,A);*

*//not(X,C);*

*//and(D,B,B);*

*//not(Y,D);*

*//and(P,Y,X);*

*//not(F,P);*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module tb\_mod;*

*reg a, b;*

*wire f;*

*mod h\_dut(a,b,f);*

*initial*

*begin*

*$dumpfile("dump.vcd");*

*$dumpvars(0, h\_dut);*

*$display("TESTING orand example");*

*#1*

*a <= 0;*

*b <= 0;*

*#1*

*#1*

*a <= 0;*

*b <= 1;*

*#1*

*#1*

*a <= 1;*

*b <= 0;*

*#1*

*#1*

*a <= 1;*

*b <= 1;*

*#1*

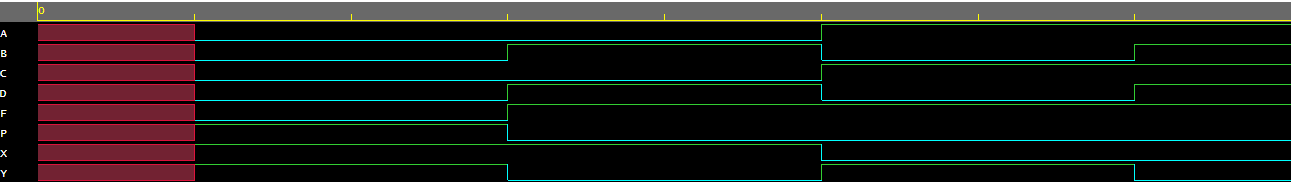
*$finish();*

*end*

*endmodule*

**Link:** <https://www.edaplayground.com/x/M4Ph>

**EP Waveform:-**



**Observation:-**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **F** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

The following truth table was obtained from the above EP Wave

Hence, we observe that the generated truth table matches to that of an OR gate.

1. **A two-input XOR.**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module mod (*

*input A,*

*input B,*

*output F*

*);*

*wire C;*

*wire X;*

*wire D;*

*wire Y;*

*wire E;*

*wire Z;*

*wire P;*

*//gate level model*

*//and(C,A,B);*

*//not(X,C);*

*//and(D,A,X);*

*//not(Y,D);*

*//and(E,B,X);*

*//not(Z,E);*

*//and(P,Y,Z);*

*//not(F,P);*

*//dataflow model*

*assign C=A&&B;*

*assign X=~C;*

*assign D=A&&X;*

*assign Y=~D;*

*assign E=B&&X;*

*assign Z=~E;*

*assign P=Y&&Z;*

*assign F=~P;*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module tb\_mod;*

*reg a, b;*

*wire f;*

*mod h\_dut(a,b,f);*

*initial*

*begin*

*$dumpfile("dump.vcd");*

*$dumpvars(0, h\_dut);*

*$display("TESTING orand example");*

*#1*

*a <= 0;*

*b <= 0;*

*#1*

*#1*

*a <= 0;*

*b <= 1;*

*#1*

*#1*

*a <= 1;*

*b <= 0;*

*#1*

*#1*

*a <= 1;*

*b <= 1;*

*#1*

*$finish();*

*end*

*endmodule*

**Link:** <https://www.edaplayground.com/x/6xDy>

**EP Waveform:-**



**Observation:-**

The following truth table was obtained from the above EP Wave

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **F** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Hence, we observe that the generated truth table matches to that of a XOR gate.

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

**F=A (B+C)**

1. **Construct the circuit using Logic gates & verify the truth table.**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module lab1 (A,B,C,F);*

*input A,B,C;*

*output F;*

*wire Y;*

*// dataflow model*

*//assign F= A && (B || C);*

*// gate level model*

*or(Y,B,C);*

*and(F,A,Y);*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module dl\_lab1;*

*reg i\_a, i\_b, i\_c;*

*wire out\_f;*

*lab1 h\_dut(i\_a,i\_b,i\_c,out\_f);*

*initial*

*begin*

*$dumpfile("dump.vcd");*

*$dumpvars(0, h\_dut);*

*$display("TESTING orand example");*

*#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*

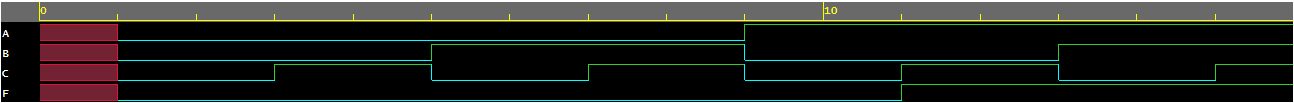
*$finish();*

*end*

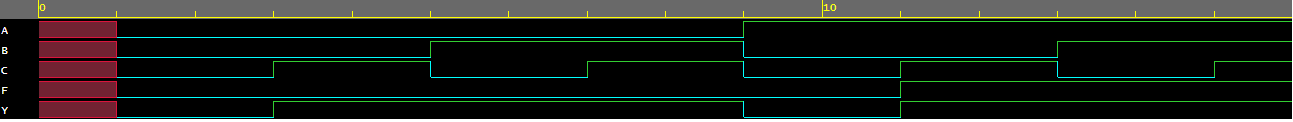
*endmodule*

**Link:** <https://www.edaplayground.com/x/XuAA>

**EP Waveform:-**



*Dataflow model*



*Gate level model*

**Observation:-**

The following truth table was obtained from the above EP Wave

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **F** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Hence, we observe that the generated truth table matches to that of the function F=A(B+C).

1. **Construct the circuit using NAND gates only & verify the truth table.**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module lab1 (A,B,C,R);*

*input A,B,C;*

*output R;*

*wire W;*

*wire D;*

*wire X;*

*wire E;*

*wire Y;*

*wire F;*

*wire Z;*

*wire G;*

*wire P;*

*// gatelevel model*

*and(W,B,B);*

*not(D,W);*

*and(X,C,C);*

*not(E,X);*

*and(Y,D,E);*

*not(F,Y);*

*and(Z,A,F);*

*not(G,Z);*

*and(P,G,G);*

*not(R,P);*

*// dataflow model*

*//assign X=~(B&&B);*

*//assign Y=~(C&&C);*

*//assign Z=~(X&&Y);*

*//assign P=~(A&Z);*

*//assign R=~(P&&P);*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module dl\_lab1;*

*reg i\_a, i\_b, i\_c;*

*wire out\_f;*

*lab1 h\_dut(i\_a,i\_b,i\_c,out\_f);*

*initial*

*begin*

*$dumpfile("dump.vcd");*

*$dumpvars(0, h\_dut);*

*$display("TESTING orand example");*

*#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*

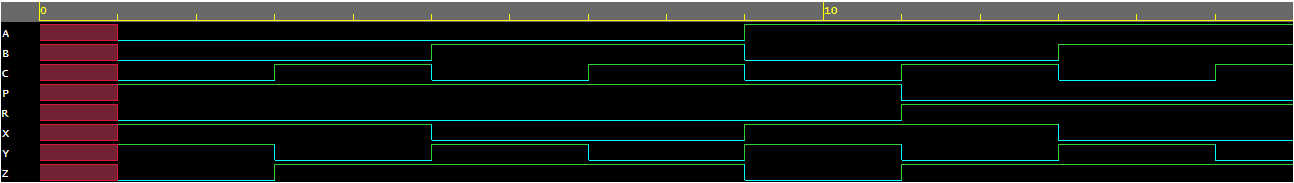
*$finish();*

*end*

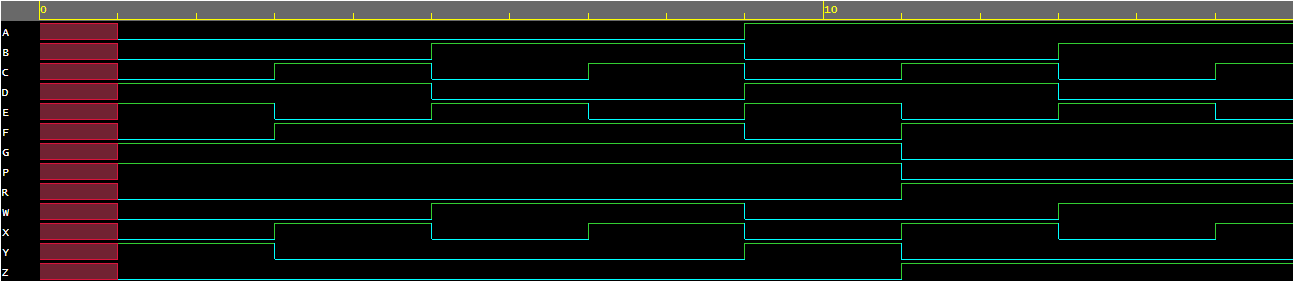
*endmodule*

**Link:** <https://www.edaplayground.com/x/DxNv>

**EP Waveform:-**



*Dataflow model*



*Gate level model*

**Observation:-**

The following truth table was obtained from the above EP Wave

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **R** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Hence, we observe that the generated truth table matches to that of the function F=A(B+C).

**Conclusion:**

* In objective 1 we verified the functioning of various logic gates.
* From objective 2 we conclude that NAND gate is an universal gate because we are able to connect and implement an Inverter, AND gate, OR gate, XOR gate from a single 7400IC.
* In objective 3 we have constructed a circuit using logic gates and 7400IC only respectively that implements the function F = A(B + C) and found that the circuit seems to be working fine.

**IV. POST LAB:**

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

**Ans:** “Acceptable” input signal voltages range from 0 volts to 0.8 volts for a “low” logic state, and 2 volts to 5 volts for a “high” logic state.

1. **What is the significance of ground and VCC connection?**

**Ans:** **VCC** (Voltage Common Collector) is the higher voltage with respect to GND (**ground**). **VCC** is the power input of a device. It may be positive or negative with respect to GND. When the only positive power supply is used then VSS (Voltage Source Supply) means **ground** or zero.

1. **Which gates are known as universal gates & why?**

**Ans:** A universal gate is a gate which can implement any Boolean function without need to use any other **gate type**. The **NAND** and NOR gates are universal gates.

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

**Ans:** An **XOR gate** circuit can be made from four **NAND gates**.