**DIGITAL LOGIC DESIGN LAB (EET1211)**

**LAB II: Examine & Analyze Advantages of Gate Level Minimization for Boolean Function Using HDL**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Branch:** Computer Science and Engineering  **Section:** D | | | |
| **S. No.** | **Name** | **Registration No.** | **Signature** |
| 2 | Saswat Mohanty | 1941012660 | **E:\sign.jpg** |

**Marks: \_\_\_\_\_\_/10**

**Remarks:**

**Teacher’s Signature**

1. **OBJECTIVE:**
2. Construct a circuit using basic gates that implements the Boolean function given below and record the output for all sets of input.

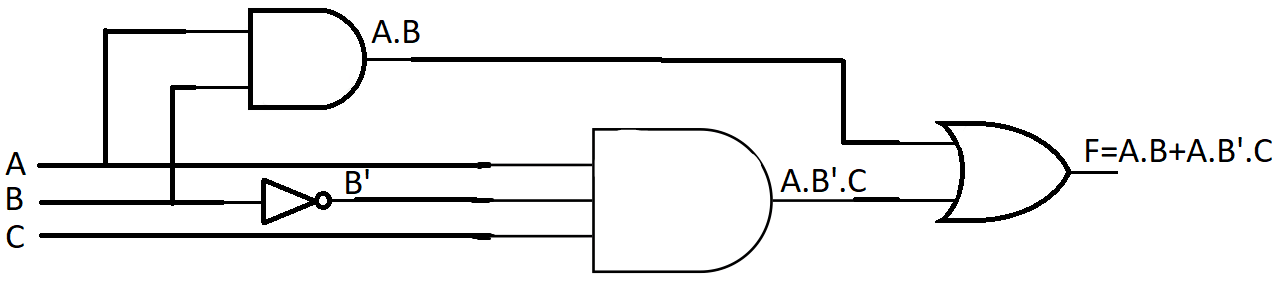
𝐹 = 𝐴𝐵 + 𝐴𝐵′𝐶

1. Simplify the Boolean function using minimization technique.
2. Construct the circuit for the simplified expression using basic gates & verify the truth table.
3. Write HDL code for the function F and for the simplified expression.
4. Simplify the following Boolean functions to a minimum number of literals and write HDL program for each function.
5. F= XY +X′Z+YZ
6. F= (X′Y′+Z) ′+Z+XY+WZ
7. Consider two Boolean functions in sum-of-min terms form:

F1 (A, B, C, D) = (0, 1, 2, 3, 4, 6, 8, 9, 10, 11)

F2 (A, B, C, D) = (3, 5, 7, 8, 10, 11, 13, 15)

1. Implement both the functions using a minimum number of NAND ICs & verify the truth tables.
2. Write HDL code for the functions
3. **PRE-LAB**
4. **For Obj. 1:**
5. **Draw the circuit diagram.**



1. **Obtain the truth table for F as function of 3 inputs.**

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

1. **Simplify the Boolean function using Boolean algebra rules & verify the correctness of the simplified expression using truth table.**

F = (A.B)+(A.B’.C)

= A.(B+(B’.C) (Taking A common from the both)

= A.((B+B’).(B+C)) (Distributive Law)

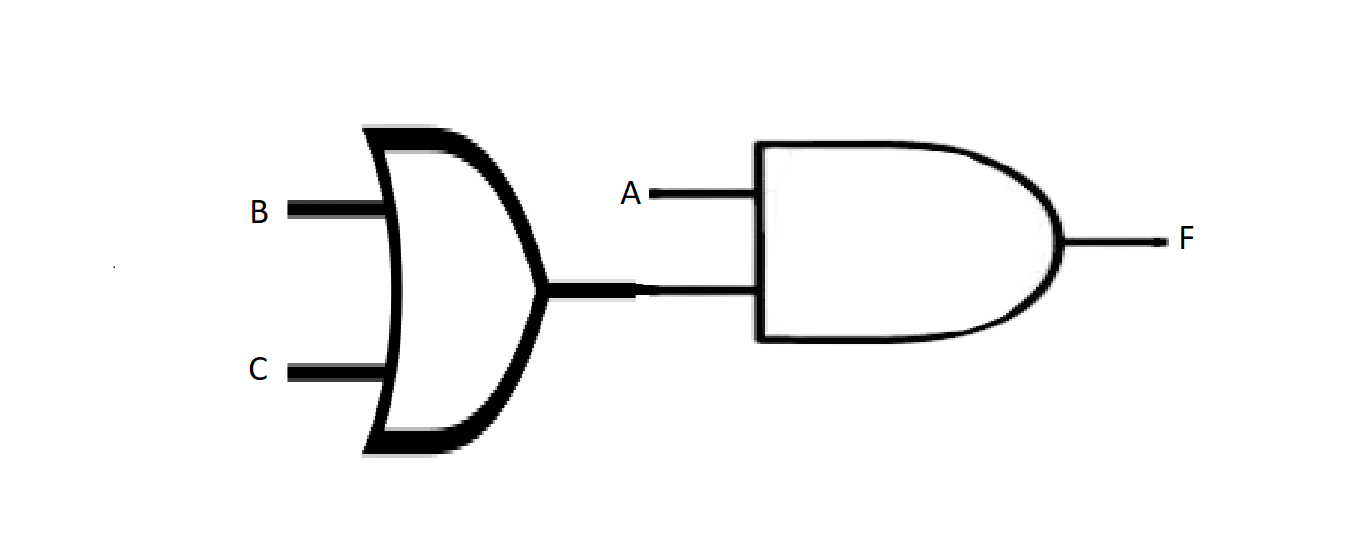
= A.((1).(B+C)) (Complement Law)

= A.(B+C)

**Truth Table:-**

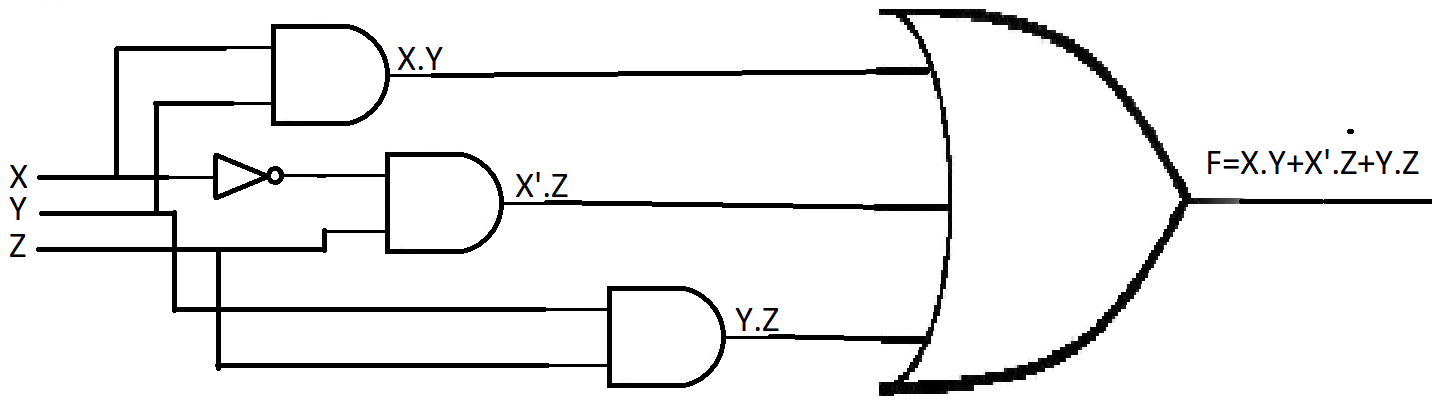
|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **F** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

1. **Draw the logic diagram of simplified Boolean expression**



1. **For Obj. 2:**
2. **Draw the circuit diagram and obtain the truth table for all input combinations.**

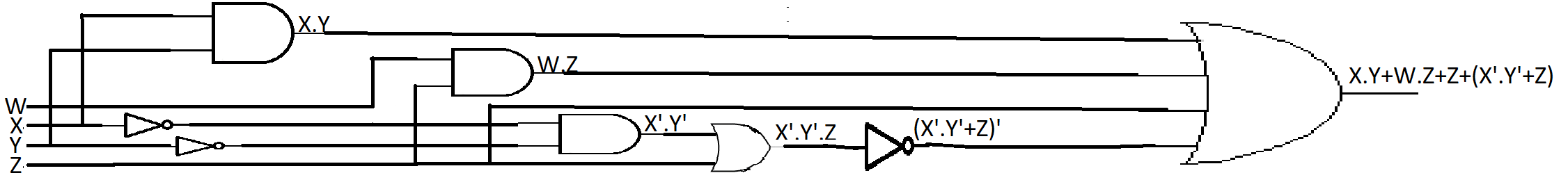
***For the function, F= XY +X′Z+YZ***



**Truth Table:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **F** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

***For the function, F= (X′Y′+Z) ′+Z+XY+WZ***



**Truth Table:-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **W** | **X** | **Y** | **Z** | **F** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

1. **Simplify the functions.**

F = X.Y+X′.Z+Y.Z

= X.Y+X’Z+Y.Z(X+X’) (Multiply 1 with Y.Z)

= X.Y+X’Z+X.Y.Z+X’.Y.Z (Distributive Law)

= X.Y.(1+Z)+X’.Z.(1+Y) (Taking X.Y and X’.Z common from the both)

= X.Y+X’.Z (Identity Law)

F = (X′.Y′+Z)′+Z+X.Y+W.Z

= (X′.Y′+Z)′+X.Y+Z(1+W) (Taking Z common from the both)

= (X′.Y′+Z)′+X.Y+Z (Identity Law)

= (X+Y)\*Z’+Z+X.Y (Commutative Law & De Morgan’s Theorem)

= (Z+Z’).(X+Y+Z)+X.Y (Distributive Law)

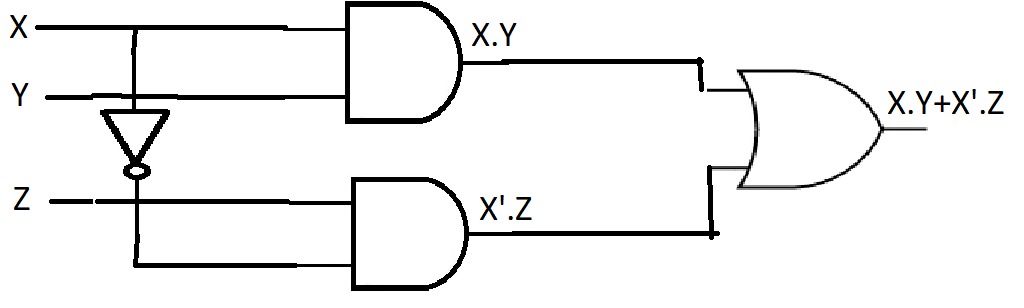
= X+Y+Z+X.Y (Distributive Law)

= X+Y+Z (Absorption Law)

1. **Draw the circuit & write truth table for minimized functions.**

The function*, F= XY +X′Z+YZ* after minimization becomes,

***F=X.Y+X’.Z***

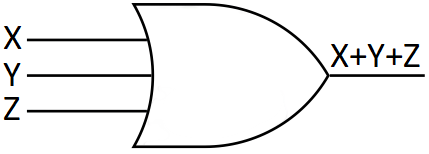


**Truth Table:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **F** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

The function*, F= (XʹYʹ+Z) ʹ+Z+XY+WZ* after minimization becomes,

***F=X+Y+Z***

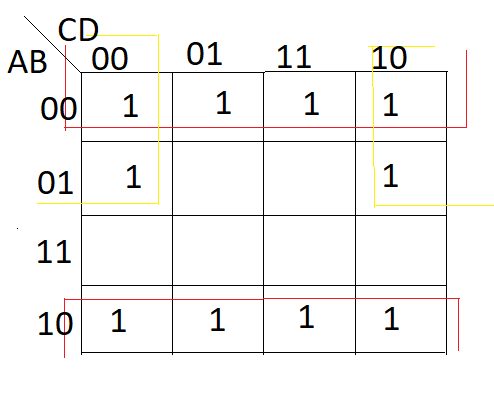


**Truth Table:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **F** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

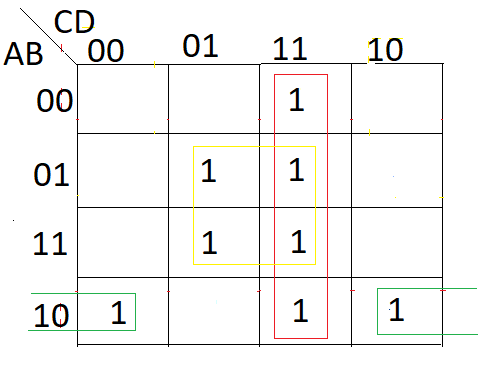
1. **For Obj. 3:**
2. **Simplify the given expressions using k-map.**

**F1 (A, B, C, D) = ∑(0, 1, 2, 3, 4, 6, 8, 9, 10, 11)**



*Simplified Expression: - B’+A’D’*

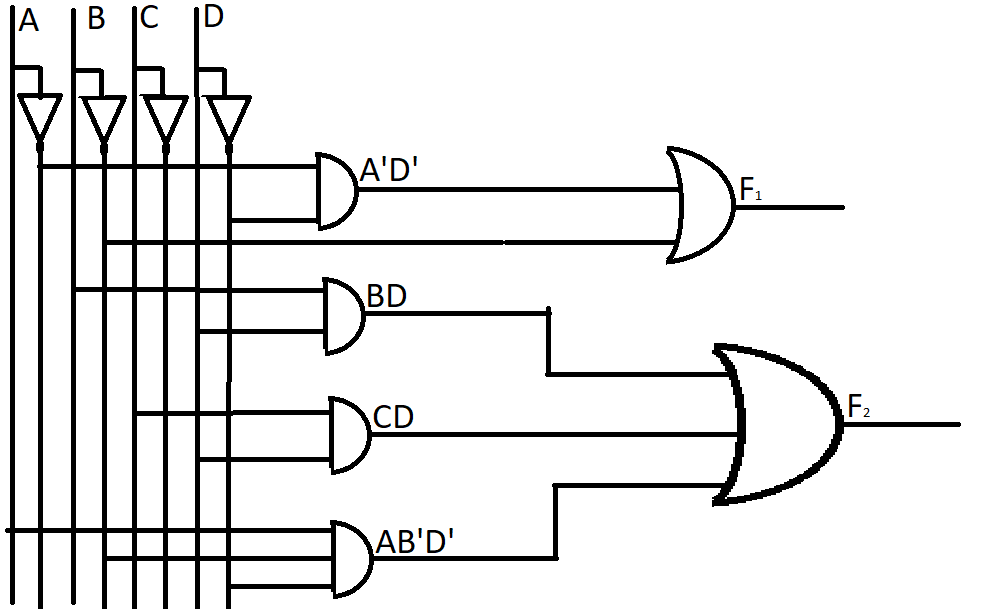
**F2 (A, B, C, D) = ∑(3, 5, 7, 8, 10, 11, 13, 15)**



*Simplified Expression: - CD+BD+AB’D’*

1. **Obtain a composite logic diagram with four inputs, A, B, C and D, and two outputs, F1, and F2. Obtain truth tables for both the functions.**

F1= B’+A’D’ F2= CD+BD+AB’D’



**Truth Table:-**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **F1= B’+A’D’** | **F2= CD+BD+AB’D’** |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 |

**III. LAB:**

1. **Construct a circuit using basic gates that implements the Boolean function given below and record the output for all sets of input.**

**𝐹 = 𝐴𝐵 + 𝐴𝐵′𝐶**

1. **Simplify the Boolean function using minimization technique.**

F = (A.B) + (A.B’.C)

=A.(B + (B’.C) (Taking A common from the both)

=A.((B + B’).(B + C)) (Distributive Law)

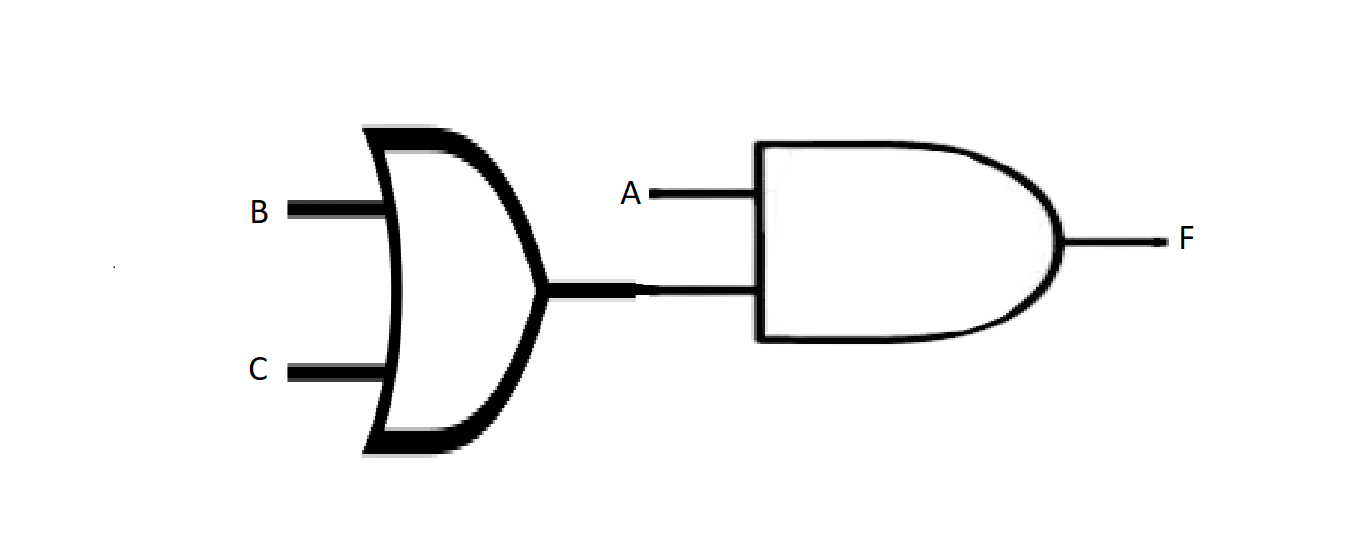
=A.((1).(B + C)) (Complement Law)

=A.(B + C)

1. **Construct the circuit for the simplified expression using basic gates & verify the truth table.**

Truth table:-

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



1. **Write HDL code for the function F and for the simplified expression.**

*For the original function,* **𝐹 = 𝐴𝐵 + 𝐴𝐵′𝐶**

**Code:-**

**design.sv:**

*`default\_nettype none*

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

*input A,B,C;*

*output F;*

*wire P,Q,R,S;*

*// dataflow model*

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

*// gate level model*

*//and a1(P,A,B);*

*//and a2(Q,A,C);*

*//not n1(R,B);*

*//and a3(S,R,Q);*

*//or(F,P,S);*

*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*

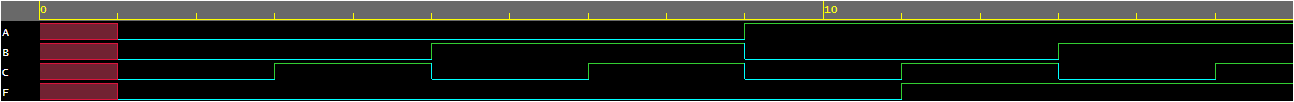
*$finish();*

*end*

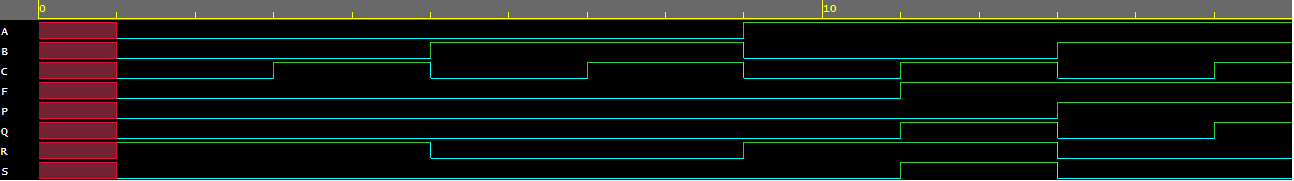
*endmodule*

**Link: -** <https://www.edaplayground.com/x/Ff76>

**EP Waveform:-**



***Dataflow model***



***Gate-level model***

**Observation:-**

The following truth table was obtained from the above EP Waveforms:

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

*For the simplified function,* **𝐹 = *(*𝐵+*C)***

**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*

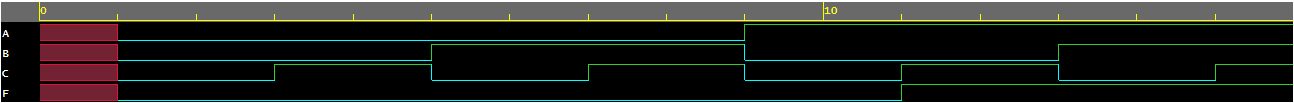
*$finish();*

*end*

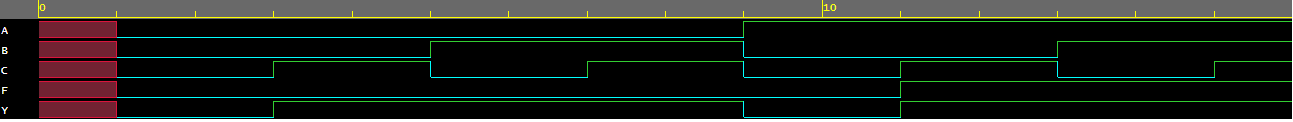
*endmodule*

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

**EP Waveform:-**



***Dataflow model***



***Gate-level model***

**Observation:-**

The following truth table was obtained from the above EP Waveforms:

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

1. **Simplify the following Boolean functions to a minimum number of literals and write HDL program for each function.**
2. **F= XY +X′Z+YZ**

F= XY +X′Z+YZ

=X.Y+X’Z+Y.Z(X+X’) (Multiply 1 with Y.Z)

=X.Y+X’Z+X.Y.Z+X’.Y.Z (Distributive Law)

=X.Y.(1+Z)+X’.Z.(1+Y) (Taking X.Y and X’.Z common from the both)

=X.Y+X’.Z (Identity Law)

*For the original function,* ***F= XY + X′Z + YZ***

**Code:-**

**design.sv:**

*`default\_nettype none*

*module lab1 (X,Y,Z,F);*

*input X,Y,Z;*

*output F;*

*wire P,Q,R,S,T;*

*// dataflow model*

*assign F= (X&&Y)||(~X&&Z)||(Y&&Z);*

*//gate level model*

*//and a1(P,X,Y);*

*//not n1(Q,X);*

*//and a2(R,Q,Z);*

*//and a3(S,Y,Z);*

*//or o1(T,P,R);*

*//or o2(F,T,S);*

*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*

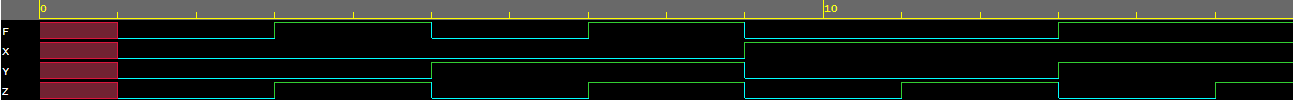
*$finish();*

*end*

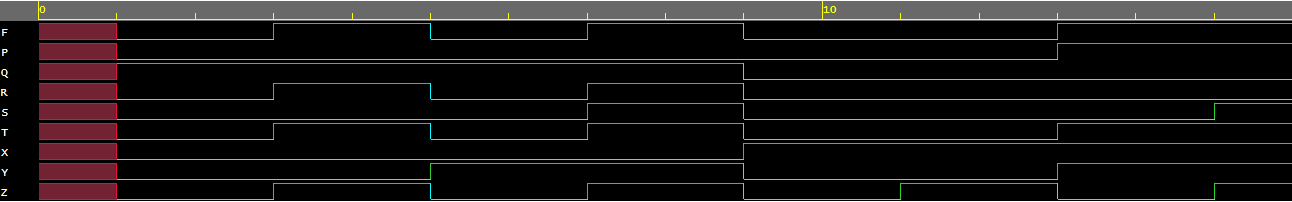
*endmodule*

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

**EP Waveform:-**



***Dataflow model***



***Gate-level model***

**Observation:-**

The following truth table was obtained from the above EP Waveforms:

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **F** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

*For the simplified expression,* ***F= XY + X’Z***

**Code:-**

**design.sv:**

*`default\_nettype none*

*module lab1 (X,Y,Z,F);*

*input X,Y,Z;*

*output F;*

*wire P,Q,R;*

*// dataflow model*

*assign F= (X&&Y)||(~X&&Z);*

*// gate level model*

*//and a1(P,X,Y);*

*//not n1(Q,X);*

*//and a2(R,Q,Z);*

*//or o1(F,P,R);*

*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*

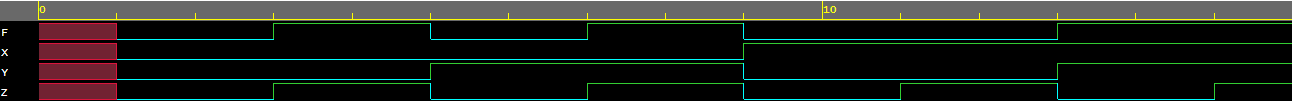
*$finish();*

*end*

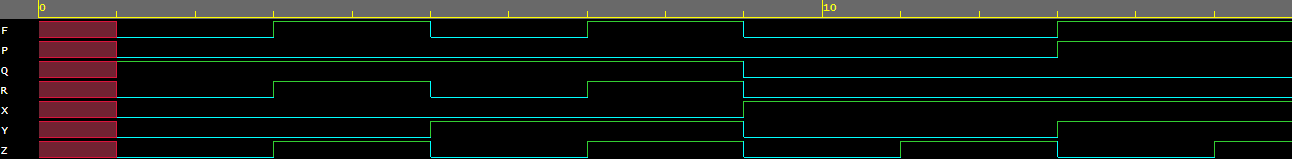
*endmodule*

**Link:-** <https://www.edaplayground.com/x/r9P2>

**EP Waveform:-**



***Dataflow model***



***Gate-level model***

**Observation:-**

The following truth table was obtained from the above EP Waveforms:

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **F** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

1. **F= (X′Y′+Z) ′+Z+XY+WZ**

F = (X′Y′+Z) ′+Z+XY+WZ

= (X′.Y′+Z)′+X.Y+Z(1+W) (Taking Z common from the both)

= (X′.Y′+Z)′+X.Y+Z (Identity Law)

= (X+Y)\*Z’+Z+X.Y (Commutative Law & DeMorgans Theorem)

= (Z+Z’).(X+Y+Z)+X.Y (Distributive Law)

= X+Y+Z+X.Y (Distributive Law)

= X+Y+Z (Absorption Law)

*For the original function,* ***F = (X′Y′ + Z)′ + Z + XY + WZ***

**Code:-**

**design.sv:**

*`default\_nettype none*

*module lab1 (*

*input W,*

*input X,*

*input Y,*

*input Z,*

*output F*

*);*

*wire P;*

*wire Q;*

*wire R;*

*wire S;*

*wire T;*

*wire U;*

*wire V;*

*wire A;*

*wire B;*

*// dataflow model*

*assign F=~(~X&&~Y||Z)||Z||X&&Y||W&&Z;*

*// gate level model*

*//not(P,X);*

*//not(Q,Y);*

*//and(R,P,Q);*

*//or(S,R,Z);*

*//not(T,S);*

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

*//and(V,W,Z);*

*//or(A,T,Z);*

*//or(B,U,V);*

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

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module dl\_lab1;*

*reg a, b, c, d;*

*wire f;*

*lab1 parms(a,b,c,d,f);*

*initial*

*begin*

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

*$dumpvars(0, parms);*

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

*#1*

*a<=0;*

*b<=0;*

*c<=0;*

*d<=0;*

*#1*

*#1*

*a<=0;*

*b<=0;*

*c<=0;*

*d<=1;*

*#1*

*#1*

*a<=0;*

*b<=0;*

*c<=1;*

*d<=0;*

*#1*

*#1*

*a<=0;*

*b<=0;*

*c<=1;*

*d<=1;*

*#1*

*#1*

*a<=0;*

*b<=1;*

*c<=0;*

*d<=0;*

*#1*

*#1*

*a<=0;*

*b<=1;*

*c<=0;*

*d<=1;*

*#1*

*#1*

*a<=0;*

*b<=1;*

*c<=1;*

*d<=0;*

*#1*

*#1*

*a<=0;*

*b<=1;*

*c<=1;*

*d<=1;*

*#1*

*#1*

*a<=1;*

*b<=0;*

*c<=0;*

*d<=0;*

*#1*

*#1*

*a<=1;*

*b<=0;*

*c<=0;*

*d<=1;*

*#1*

*#1*

*a<=1;*

*b<=0;*

*c<=1;*

*d<=0;*

*#1*

*#1*

*a<=1;*

*b<=0;*

*c<=1;*

*d<=1;*

*#1*

*#1*

*a<=1;*

*b<=1;*

*c<=0;*

*d<=0;*

*#1*

*#1*

*a<=1;*

*b<=1;*

*c<=0;*

*d<=1;*

*#1*

*#1*

*a<=1;*

*b<=1;*

*c<=1;*

*d<=0;*

*#1*

*#1*

*a<=1;*

*b<=1;*

*c<=1;*

*d<=1;*

*#1*

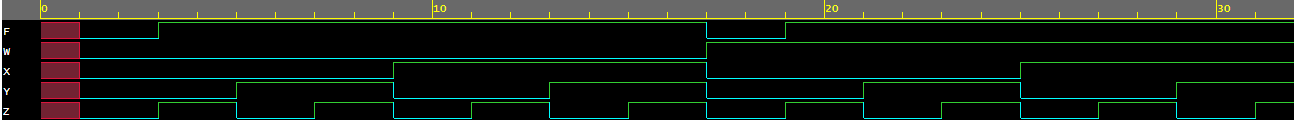
*$finish();*

*end*

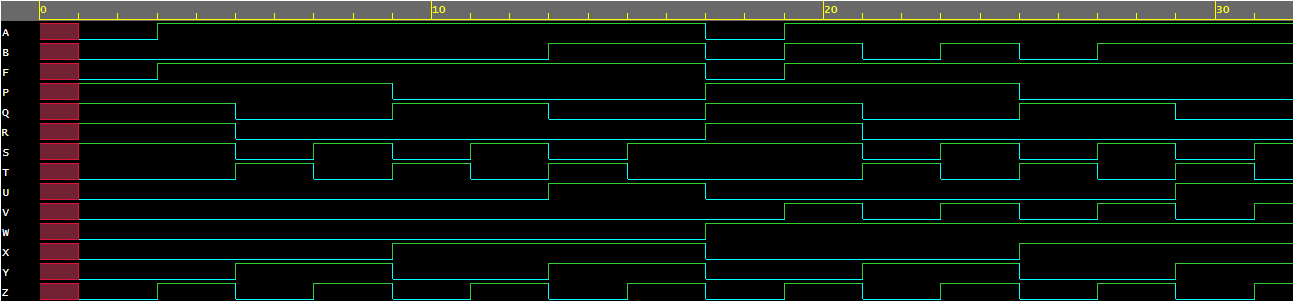
*endmodule*

**Link:-** <https://www.edaplayground.com/x/HtJ5>

**EP Waveform:-**



***Dataflow model***



***Gate-level model***

**Observation:-**

The following truth table was obtained from the above EP Waveforms:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **W** | **X** | **Y** | **Z** | **F** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

*For the simplified function,* ***F=X+Y+Z***

**Code:-**

**design.sv:**

*`default\_nettype none*

*module lab1 (X,Y,Z,F);*

*input X,Y,Z;*

*output F;*

*wire P,Q;*

*// datafloe model*

*assign F=X||Y||Z ;*

*// gate level model*

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

*//or(Q,P,Z);*

*//or(F,P,Q);*

*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*

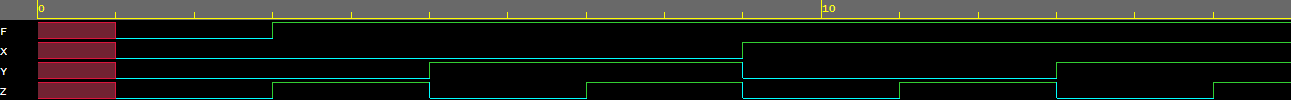
*$finish();*

*end*

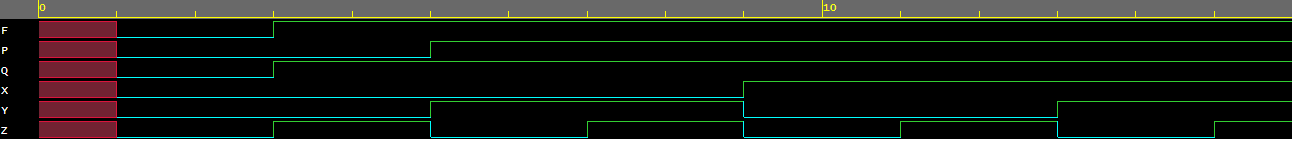
*endmodule*

**Link:-** <https://www.edaplayground.com/x/a5ee>

**EP Waveform:-**



***Dataflow model***



***Gate-level model***

**Observation:-**

The following Truth Table was obtained from the above EP Waveforms:

|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **Z** | **F** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

1. **Consider two Boolean functions in sum-of-min terms form:**

**F1 (A, B, C, D) = (0, 1, 2, 3, 4, 6, 8, 9, 10, 11)**

**F2 (A, B, C, D) = (3, 5, 7, 8, 10, 11, 13, 15)**

1. **Implement both the functions using a minimum number of NAND ICs & verify the truth tables.**

**F1 (A, B, C, D) = (0, 1, 2, 3, 4, 6, 8, 9, 10, 11)**

F1 = (A’B’C’D’) + (A’B’C’D) + (A’B’CD’) + (A’B’CD) + (A’BC’D’) + (A’BCD’) + (AB’C’D’) + (AB’C’D) + (AB’CD’) + (AB’CD)

Simplified Expression: - B’+A’D’

Nand Implementation: - (B.(A’.D’)’)’ = (B.((A’)’+(D’)’)’ = (B.(A+D))’

Truth Tables:-

|  |  |  |  |
| --- | --- | --- | --- |
| **F1 = B’ + A’.D’** | | | |
| **A** | **B** | **D** | **F1** |
| F | F | F | T |
| F | F | T | T |
| F | T | F | T |
| F | T | T | F |
| T | F | F | T |
| T | F | T | T |
| T | T | F | F |
| T | T | T | F |

|  |  |  |  |
| --- | --- | --- | --- |
| **F1 = (B.(A + D))’** | | | |
| **A** | **B** | **D** | **F1** |
| F | F | F | T |
| F | F | T | T |
| F | T | F | T |
| F | T | T | F |
| T | F | F | T |
| T | F | T | T |
| T | T | F | F |
| T | T | T | F |

**F2 (A, B, C, D) = (3, 5, 7, 8, 10, 11, 13, 15)**

F2 = (A’B’CD) + (A’BC’D) + (A’BCD) + (AB’C’D’) + (AB’CD’) + (AB’CD) + (ABC’D) + (ABCD)

Simplified Expression: - CD+BD+AB’D’

Nand Implementation: - ((CD)’.(BD)’.(AB’D’)’)’

Truth Tables:-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **F2 = ((CD)’.(BD)’.(AB’D’)’)’** | | | | |
| **A** | **B** | **C** | **D** | **F2** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **F2 = CD + BD + AB’D’** | | | | |
| **A** | **B** | **C** | **D** | **F2** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

1. **Write HDL code for the functions.**

**Code:-**

**design.sv:**

*module lab2obj3 (*

*input A,*

*input B,*

*input C,*

*input D,*

*output F1,*

*output F2*

*);*

*wire w1,w2,w3,w4,w5,w6,w7,w8;*

*// gate-level model*

*nand(w1,A,A);*

*nand(w2,D,D);*

*nand(w3,w1,w2);*

*nand(F1,B,w3);*

*nand(w4,C,D);*

*nand(w5,B,D);*

*nand(w6,B,B);*

*nand(w7,D,D);*

*nand(w8,w6,w7,A);*

*nand(F2,w8,w5,w4);*

*//data level modelling*

*//assign F1=(!B)|| (!A && !D);*

*//assign F2=(C&&D)||(B&&D)||(A&&!B&&!D);*

*endmodule*

**testbench.sv:**

*module lab2obj\_3;*

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

*wire out\_x,out\_y;*

*lab2obj3 h\_dut(i\_a,i\_b,i\_c,i\_d,out\_x,out\_y);*

*// UPDATED UNIT TEST //*

*initial*

*begin*

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

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

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

*#1*

*i\_a <= 0;*

*i\_b <= 0;*

*i\_c <= 0;*

*i\_d <= 0;*

*#1*

*#1*

*i\_a <= 0;*

*i\_b <= 0;*

*i\_c <= 0;*

*i\_d <= 1;*

*#1*

*#1*

*i\_a <= 0;*

*i\_b <= 0;*

*i\_c <= 1;*

*i\_d <= 0;*

*#1*

*#1*

*i\_a <= 0;*

*i\_b <= 0;*

*i\_c <= 1;*

*i\_d <= 1;*

*#1*

*#1*

*i\_a <= 0;*

*i\_b <= 1;*

*i\_c <= 0;*

*i\_d <= 0;*

*#1*

*#1*

*i\_a <= 0;*

*i\_b <= 1;*

*i\_c <= 0;*

*i\_d <= 1;*

*#1*

*#1*

*i\_a <= 0;*

*i\_b <= 1;*

*i\_c <= 1;*

*i\_d <= 0;*

*#1*

*#1*

*i\_a <= 0;*

*i\_b <= 1;*

*i\_c <= 1;*

*i\_d <= 1;*

*#1*

*#1*

*i\_a <= 1;*

*i\_b <= 0;*

*i\_c <= 0;*

*i\_d <= 0;*

*#1*

*#1*

*i\_a <= 1;*

*i\_b <= 0;*

*i\_c <= 0;*

*i\_d <= 1;*

*#1*

*#1*

*i\_a <= 1;*

*i\_b <= 0;*

*i\_c <= 1;*

*i\_d <= 0;*

*#1*

*#1*

*i\_a <= 1;*

*i\_b <= 0;*

*i\_c <= 1;*

*i\_d <= 1;*

*#1*

*#1*

*i\_a <= 1;*

*i\_b <= 1;*

*i\_c <= 0;*

*i\_d <= 0;*

*#1*

*#1*

*i\_a <= 1;*

*i\_b <= 1;*

*i\_c <= 0;*

*i\_d <= 1;*

*#1*

*#1*

*i\_a <= 1;*

*i\_b <= 1;*

*i\_c <= 1;*

*i\_d <= 0;*

*#1*

*#1*

*i\_a <= 1;*

*i\_b <= 1;*

*i\_c <= 1;*

*i\_d <= 1;*

*#1*

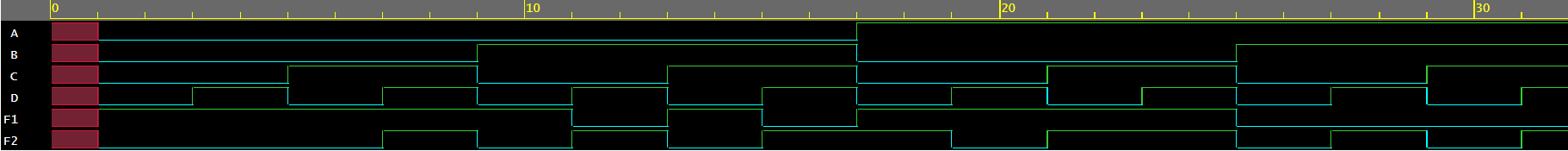
*$finish();*

*end*

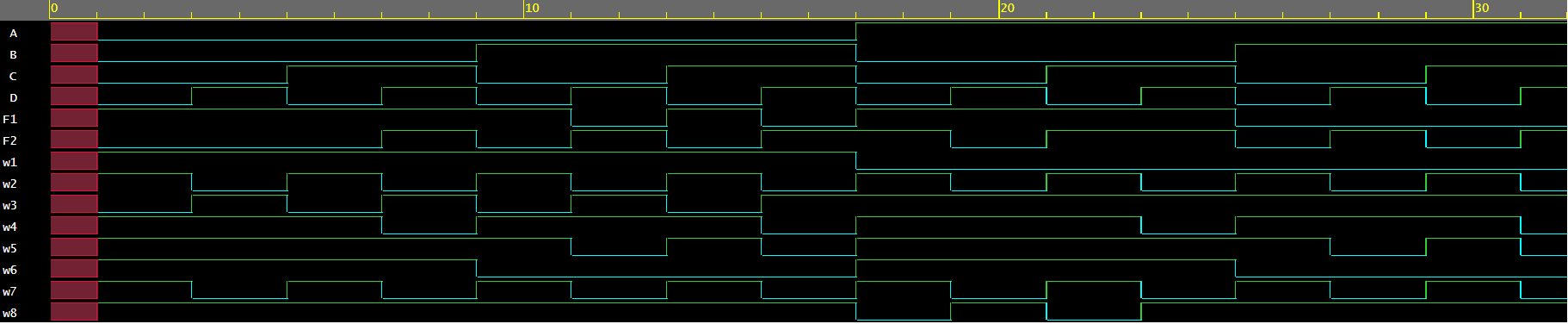
*endmodule*

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

**EP Waveform:-**



***Dataflow model***



***Gate-level model***

**Observation:-**

The following Truth Table was obtained from the above EP Waveforms:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **F1** | **F2** |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 |

Hence we observe that the above truth table matches to that of the one we came up with.

**Conclusion:**

* **Objective 1:**

From this objective it can be concluded that using Boolean law the given function 𝐴𝐵 + 𝐴𝐵′𝐶 can be simplified to A.(B+C) and can also be reconstructed.

* **Objective 2:**

From this objective it can be concluded that using Boolean law the given functions can be simplified to minimum number of literals.

* **Objective 3:**

From this objective it can be concluded that using K-map the given functions can be simplified to minimum number of literals and can also be implemented using NAND ICs.

**IV. POST LAB:**

1. **What is the advantage of Boolean laws?**

**Ans:-** Using Boolean law we can minimize a function which helps us to reduce the number of literals and construct the circuit using minimum number of gates.

1. **What is a K-map? What are its advantages and**

**disadvantages?**

**Ans:-** Karnaugh maps take truth tables and provide a visual way to produce a much simpler formula for expressing the same logic.

Advantages -

* A k map use reduces the number of logic gates to be used to solve a logic expression.
* reduces the error

Disadvantage -

* It is not suitable when the number of variables involved exceed four.
* It is not suitable for computer reduction.

1. **What ways can Boolean expressions be represented?**

**Ans:-** We can represent the Boolean function using 2 forms i.e.

* Standard form
* Canonical form

1. **Implement two-input Ex-NOR gate using minimum number of two input NOR gates.**

**Ans:-** F = A’B’ + AB

