**DIGITAL LOGIC DESIGN LAB (EET1211)**

**LAB III: Design, Construct & Test the Combinational Circuit to Solve a Given Problem Using HDL**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Branch:** Computer Science and 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. Design a combinational circuit with four inputs A, B, C, and D and one output F. F is to be equal to 1 when (A = 1 and B = 0), or when (A=0, B = 1, provided that either C or D is equal to 1). F is also equal to 1 when (A=0, B=1, C=1 and D=1). Otherwise, the output is to be equal to 0.
2. Design and test a 3-input majority circuit using NAND gates with a minimum number of ICs. A majority logic is a digital circuit whose output is equal to 1 if the majority of the inputs are 1’s. The output is 0 otherwise.
3. Design, construct, and test a circuit that generates an even parity bit from four message bits.
4. Design a combinational circuit that compares two 2-bit numbers A and B to check if they are equal or not.

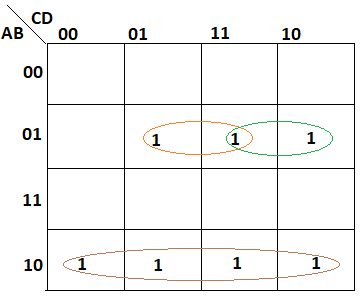
**II. PRE-LAB**

**For Obj. 1:**

1. **Obtain the truth table for output F as a function of 4 inputs (A, B, C & D) based on the given logic.**

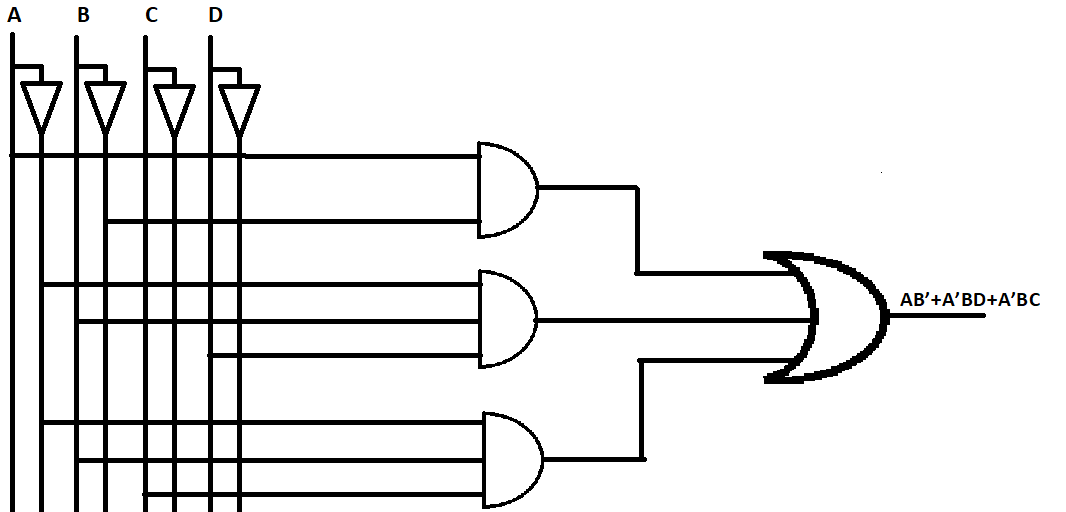
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **F** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0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

1. **Derive a simplified Boolean expression for F that satisfies the truth table.**



Simplified expression: ***AB’ + A’BD + A’BC***

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



1. **Write HDL code.**

**design.sv:**

*`default\_nettype none*

*module lab3 (*

*input A,*

*input B,*

*input C,*

*input D,*

*output F*

*);*

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

*// dataflow model*

*assign F=A&&~B||~A&&B&&D||~A&&B&&C;*

*// gate-level model*

*not n1(P,A),*

*n2(Q,B);*

*and a1(R,A,Q),*

*a2(S,P,B,D),*

*a3(T,P,B,C);*

*or o1(F,R,S,T);*

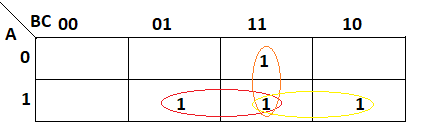
*endmodule*

**For Obj. 2:**

1. **Draw the truth table for 3 input majority circuits.**

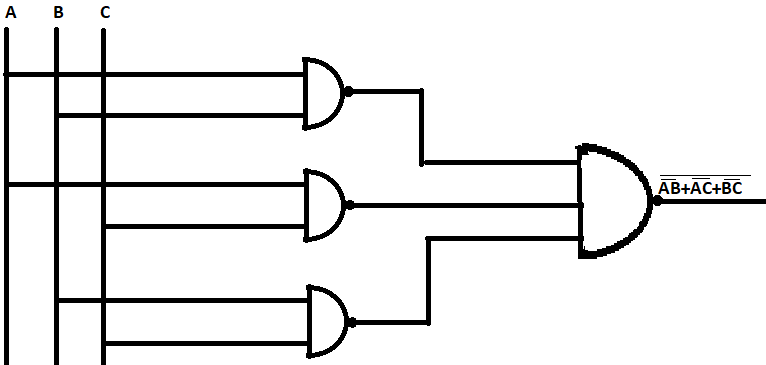
|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | F |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

1. **Write a simplified Boolean expression for the output of the given 3 input majority circuit.**



Simplified Expression: ***AB + AC + BC***

1. **Draw the circuit diagram using NAND gates according to the simplified Boolean expression.**



1. **Write HDL code.**

**design.sv:**

*`default\_nettype none*

*module lab3 (A,B,C,F);*

*input A,B,C;*

*output F;*

*wire P,Q,R;*

*// dataflow model*

*assign F=A&&B||A&&C||B&&C;*

*// gate-level model*

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

*a2(Q,A,C),*

*a3(R,B,C);*

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

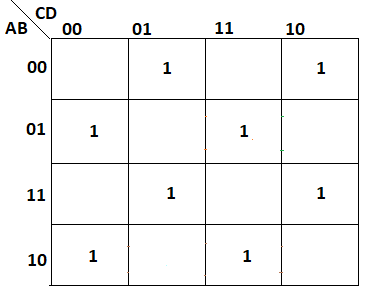
*endmodule*

**For Obj. 3:**

1. **Obtain the truth table for the parity bit as output of the circuit corresponding to 4 input message bits.**

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

1. **Derive a simplified Boolean function that defines the value of parity bit for given combination of 4 messages bits.**



= A’B’(C’D+CD’) + A’B(C’D’+CD) + AB(C’D+CD’) + AB’(C’D’+CD)

= A’B’(C **⊕** D) + A’B(C ⊙D) + AB(C **⊕** D) + AB’(C ⊙D)

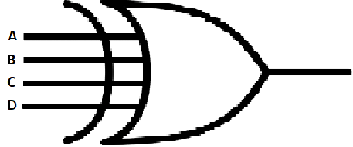
= (C **⊕** D)(A’B’ + AB) + (C ⊙D) + (A’B + AB’)

= (C **⊕** D) (A ⊙B) + (C ⊙D) ( A **⊕** B)

Let (A **⊕** B) & (C **⊕** D) be X & Y respectively therefore equation will be XY’+X’Y that equals to X **⊕** Y

=A **⊕** B **⊕** C **⊕** D

1. **Draw the logic diagram using logic gates according to derived Boolean function.**



1. **Write HDL code.**

**design.sv:**

*`default\_nettype none*

*module lab3 (*

*input A,*

*input B,*

*input C,*

*input D,*

*output F*

*);*

*// dataflow model*

*assign F=A^B^C^D;*

*//gate-level model*

*xor(F,A,B,C,D);*

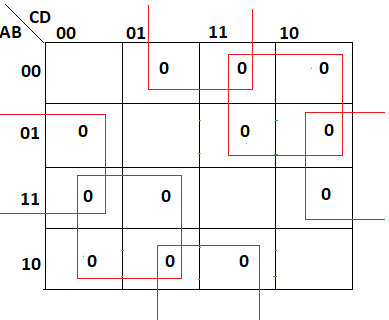
*endmodule*

**For Obj. 4:**

1. **Obtain the truth table to know for which set of data bits the output of the circuit is high /low.**

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | | **B** | |
| **A0** | **A1** | **B0** | **B1** | **F** |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

1. **Derive a simplified Boolean expression that justifies the given logic.**



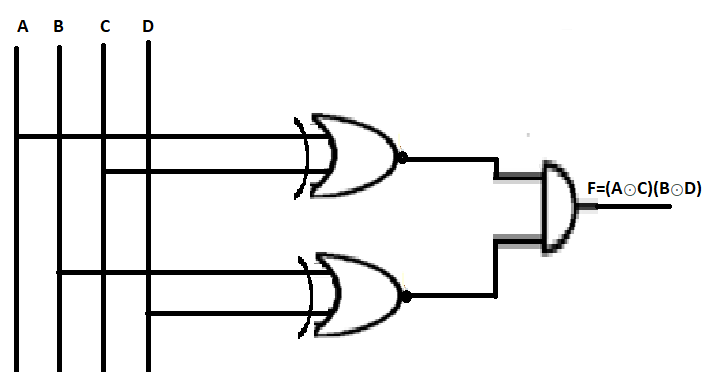
Simplified expression: ***F’ = AC’ + A’C + BD’ + B’D***

***F = (AC’ + A’C + BD’ + B’D)’***

***= (A⊕C + B⊕D)’***

***= (A⊙C)(B⊙D)***

1. **Draw the logic diagram using logic gates according to derived Boolean function.**



1. **Write HDL code.**

**design.sv:**

*`default\_nettype none*

*module lab3 (*

*input A,*

*input B,*

*input C,*

*input D,*

*output F*

*);*

*wire P,Q;*

*// dataflow model*

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

*// gate-level model*

*xnor x1(P,A,C);*

*xnor x2(Q,B,D);*

*and a1(F,P,Q);*

*endmodule*

**III. LAB:**

1. **Design a combinational circuit with four inputs A, B, C, and D and one output F. F is to be equal to 1 when (A = 1 and B = 0), or when (A=0, B = 1, provided that either C or D is equal to 1). F is also equal to 1 when (A=0, B=1, C=1 and D=1). Otherwise, the output is to be equal to 0.**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module lab3 (*

*input A,*

*input B,*

*input C,*

*input D,*

*output F*

*);*

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

*// dataflow model*

*assign F=A&&~B||~A&&B&&D||~A&&B&&C;*

*// gate-level model*

*not n1(P,A),*

*n2(Q,B);*

*and a1(R,A,Q),*

*a2(S,P,B,D),*

*a3(T,P,B,C);*

*or o1(F,R,S,T);*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module dl\_lab3;*

*reg a, b, c, d;*

*wire f;*

*lab3 parms(a,b,c,d,f);*

*initial*

*begin*

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

*$dumpvars(0, parms);*

*$display("Lab 3 Obj 1");*

*#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*

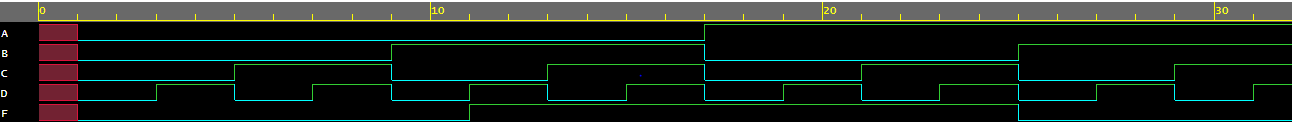
*$finish();*

*end*

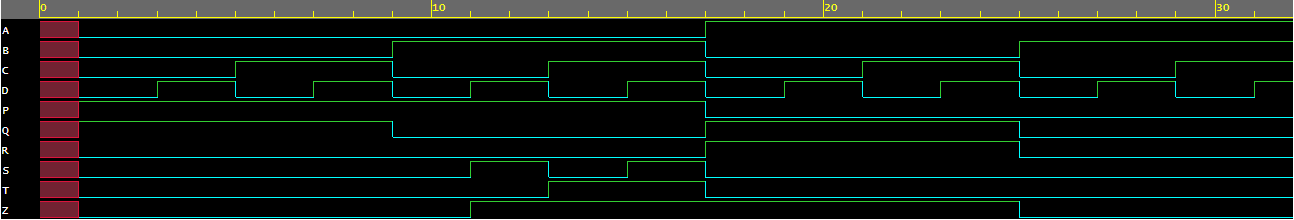
*endmodule*

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

**EP Waveform:-**



***Dataflow model***



***Gate-level model***

**Observation:-**

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

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

1. **Design and test a 3-input majority circuit using NAND gates with a minimum number of ICs. A majority logic is a digital circuit whose output is equal to 1 if the majority of the inputs are 1’s. The output is 0 otherwise.**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module lab3 (A,B,C,F);*

*input A,B,C;*

*output F;*

*wire P,Q,R;*

*// dataflow model*

*assign F=A&&B||A&&C||B&&C;*

*// gate-level model*

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

*a2(Q,A,C),*

*a3(R,B,C);*

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

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module dl\_lab1;*

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

*wire out\_f;*

*lab3 h\_dut(i\_a,i\_b,i\_c,out\_f);*

*initial*

*begin*

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

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

*$display("Lab 3 Obj 2");*

*#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*

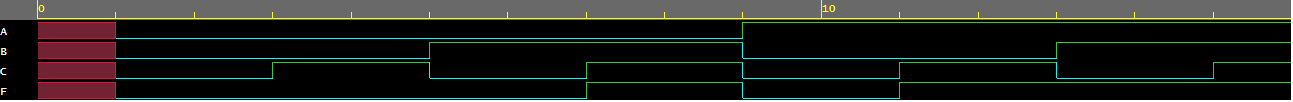
*$finish();*

*end*

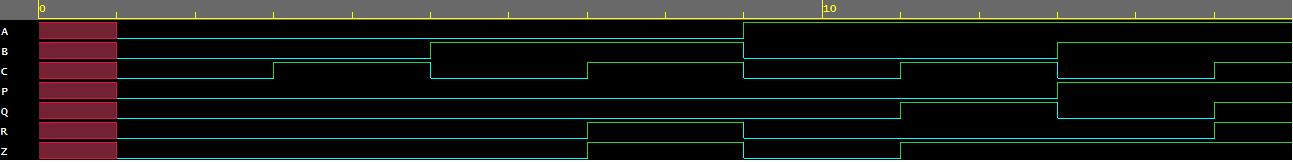
*endmodule*

**Link:-** <https://www.edaplayground.com/x/a4YV>

**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 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

1. **Design, construct, and test a circuit that generates an even parity bit from four message bits.**

**Code:-**

**design.sv:**

*`default\_nettype none*

*module lab3 (*

*input A,*

*input B,*

*input C,*

*input D,*

*output F*

*);*

*// dataflow model*

*assign F=A^B^C^D;*

*//gate-level model*

*xor(F,A,B,C,D);*

*endmodule*

**testbench.sv:**

*`default\_nettype none*

*module dl\_lab3;*

*reg a, b, c, d;*

*wire f;*

*lab3 parms(a,b,c,d,f);*

*initial*

*begin*

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

*$dumpvars(0, parms);*

*$display("Lab 3 Obj 3");*

*#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*

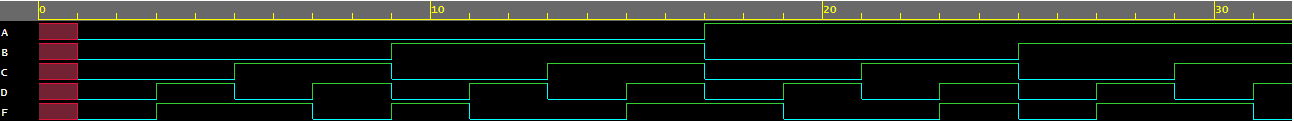
*$finish();*

*end*

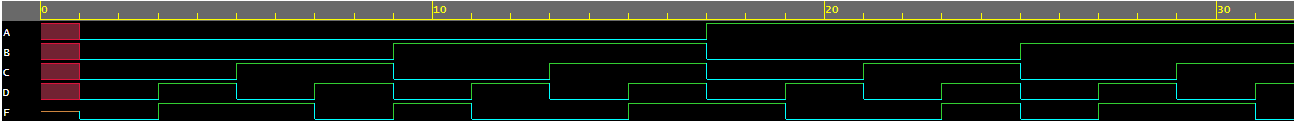
*endmodule*

**Link:-** <https://www.edaplayground.com/x/L2JT>

**EP Waveform:-**



***Dataflow model***



***Gate-level model***

**Observation:-**

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

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

1. **Design a combinational circuit that compares two 2-bit numbers A and B to check if they are equal or not**.

**Code:-**

**design.sv:**

`default\_nettype none

module lab3 (

input A,

input B,

input C,

input D,

output F

);

wire P,Q;

// dataflow model

assign F=~(A^C)&&~(B^D);

// gate-level model

xnor x1(P,A,C);

xnor x2(Q,B,D);

and a1(F,P,Q);

endmodule

**testbench.sv:**

*`default\_nettype none*

*module dl\_lab3;*

*reg a, b, c, d;*

*wire f;*

*lab3 parms(a,b,c,d,f);*

*initial*

*begin*

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

*$dumpvars(0, parms);*

*$display("Lab 3 Obj 4");*

*#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*

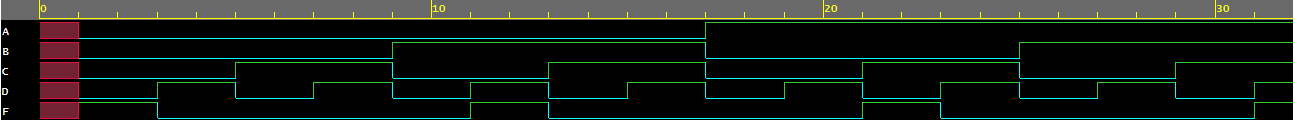
*$finish();*

*end*

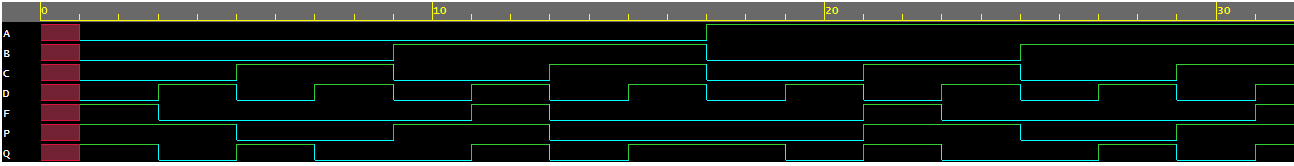
*endmodule*

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

**EP Waveform:-**



***Dataflow model***



***Gate-level model***

**Observation:-**

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

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

**Conclusion:**

**Objective 1:**

From this objective it can be concluded that for getting the output as per the instruction the combinational circuit leads to the function

***AB’+A’BD+A’BC***

**Objective 2:**

From this objective it can be concluded that 3 ­input majority circuit leads to the function

***AB+AC+BC***

**Objective 3:**

From this objective it can be concluded that even parity bit from four message bits leads to the function

***A ⊕ B ⊕ C ⊕ D***

**Objective 4:**

From this objective it can be concluded that equality condition leads to the XNOR function.

**IV. POST LAB:**

What do you understand by the term ‘majority logic’?

**Ans:-** A majority logic is a digital circuit whose output is equal to 1 if the majority of the inputs are 1’s i.e. (>50%). The output is 0 otherwise.

**Suggest a suitable modification to be made in existing even parity circuit that can be used to generate bit for odd parity.**

**Ans:-** Take XOR of it with high input i.e. 1

**What is the function of a magnitude comparator circuit?**

**Ans:-** The function of a magnitude comparator circuit is to determines whether one number is greater than, less than or equal to the other number

**V. HDL PROGRAM LINK:**

**Objective 1:** <https://www.edaplayground.com/x/jBhc>

**Objective 2:** <https://www.edaplayground.com/x/a4YV>

**Objective 3:** <https://www.edaplayground.com/x/L2JT>

**Objective 4:** <https://www.edaplayground.com/x/BNrH>