21BDS0340

Abhinav Dinesh Srivatsa

Digital Systems Design Lab

Task 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S. No. | Components | Page No. | Student Check Mark | RA Check Mark |
| 1 | Aim | 1 | ✓ |  |
| 2 | Components Required and Tools Required | 1 | ✓ |  |
| 3 | Procedure and Theory | 1, 2 | ✓ |  |
| 4 | Truth Table | 2 | ✓ |  |
| 5 | SOP Boolean expression in canonical form | 2, 3 | ✓ |  |
| 6 | POS Boolean expression in canonical form | 2, 3 | ✓ |  |
| 7 | Boolean simplification using Boolean laws | 3, 4 | ✓ |  |
| 8 | Boolean simplification using KMAP simplification | 5, 6 | ✓ |  |
| 9 | Simplified SOP expression in Standard form | 6 | ✓ |  |
| 10 | Simplified POS expression in Standard form | 6 | ✓ |  |
| 11 | Circuit diagram using AOI logic | 6 | ✓ |  |
| 12 | Circuit diagram using OAI logic | 7 | ✓ |  |
| 13 | Circuit diagram using NAND logic | 7 | ✓ |  |
| 14 | Circuit diagram using NOR logic | 7 | ✓ |  |
| 15 | Multisim live / Circuitverse.org Simulation link for SOP F and F’ circuit | 8 | ✓ |  |
| 16 | Multisim live / Circuitverse.org Simulation link for POS F and F’ circuit | 8 | ✓ |  |
| 17 | Multisim live / Circuitverse.org Simulation link for NAND F and F’ circuit | 8 | ✓ |  |
| 18 | Multisim live / Circuitverse.org Simulation link for NOR F and F’ circuit | 8 | ✓ |  |
| 19 | Justification for optimized circuit NAND /NOR /XOR Logic | 8 | ✓ |  |
| 20 | Verilog code: Gate Level | 8 | ✓ |  |
| 21 | Verilog code: Data Flow | 9 | ✓ |  |
| 22 | Test Bench | 9 | ✓ |  |
| 23 | Snip of Output waveform with respective code | 10 | ✓ |  |
| 24 | Online Verilog code Simulation links | 10 | ✓ |  |
| 25 | Result | 10 | ✓ |  |
| 26 | Inference | 11 | ✓ |  |

**Aim**

Using Reg.no. formulate expressions in SOP and POS for F and F'. Use K-Map and Boolean laws to simplify the expressions. Write a Verilog code to implement F and F’ with a neat circuit diagram for all circuits designed using the following forms.

1. SOP – AND-OR-INV logic circuit
2. POS – OR-AND-INV logic circuit
3. SOP – NAND-NAND logic circuit
4. POS – NOR-NOR logic circuit

Use only two input logic elements for AND, OR, NAND and NOR logic gates.

**Components Required**

1. AND, OR, NOT, NAND and NOR gates
2. 5V voltage source
3. Led indicator

**Tools Required**

1. Multisim simulator
2. Charlie-Coleman KMAP solver
   * <https://charlie-coleman.com/experiments/kmap/>
3. Truth table display
   * <http://www.32x8.com/>
4. Boolean algebra simplifier
   * <https://www.boolean-algebra.com/>

**Theory**

1. K-maps

* Karnaugh maps (K-maps) are graphical representations of Boolean functions.

1. Don’t care conditions

* There may be a combination of input values which (i)will never occur, (ii)if they do occur, the output is of no concern.

1. Canonical Forms

* Any Boolean function F can be expressed as a unique sum of minterms and a unique product of maxterms (under a fixed variable ordering).
* In other words, every function F() has two canonical forms:
  + Canonical Sum-Of-Products (sum of minterms)
  + Canonical Product-Of-Sums (product of maxterms)

**Procedure**

1. Write down the SOP and POS expression using registration numbers.
2. Implement the expression in K-map using Logic minimiser or Logic Friday.
3. Then design circuits using these expressions in Multisim using AND OR INV and 2 input NAND logic for Σm expressions.
4. Similarly design ΠM expressions in OR AND INV Logic and 2- input NOR Logic.
5. Compare and contrast the circuits with the number of gates used.
6. Draw suitable conclusions and inference.
7. Implement one of these 8 circuits in hardware and cross check the results with the truth table.

**Truth Table**

**A screenshot of a computer

Description automatically generated with low confidenceA screenshot of a computer

Description automatically generated with low confidenceF:** **F’:**

**Functional Expression**

F(A, B, C, D) = Σm(0, 1, 2, 3, 4, B, D) = Σm(0, 1, 2, 3, 4, 11, 13)

= ΠM(5, 6, 7, 8, 9, 10, 12, 14, 15)

F = A’B’C’D’ + A’B’C’D + A’B’CD’ + A’B’CD + A’BC’D’ + AB’CD + ABC’D **(SOP form)**

F = (A+B’+C+D’) (A+B’+C’+D) (A+B’+C’+D’) (A’+B+C+D) (A’+B+C+D’) (A’+B+C’+D) (A’+B’+C+D) (A’+B’+C’+D) (A’+B’+C’+D’) **(POS form)**

F’(A, B, C, D) = Σm(5, 6, 7, 8, 9, 10, 12, 14, 15)

= ΠM(0, 1, 2, 3, 4, 11, 13)

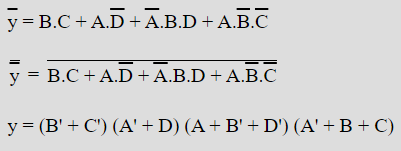
F’ = A’BC’D + A’BCD’ + A’BCD + AB’C’D’ + AB’C’D + AB’CD’ + ABC’D’ + ABCD’ + ABCD **(SOP form)**

F’ = (A+B+C+D)(A+B+C+D’) (A+B+C’+D) (A+B+C’+D’) (A+B’+C+D) (A’+B+C’+D’) (A’+B’+C+D’) **(POS form)**

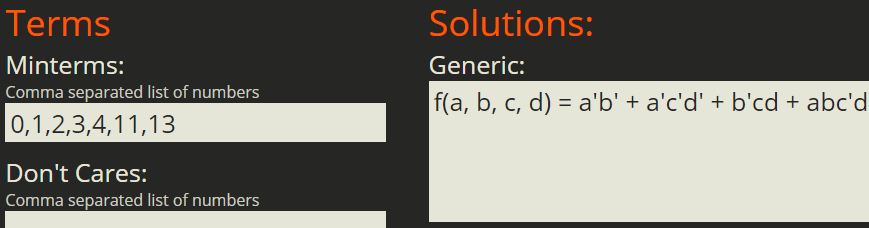
**Simplified SOP Form**

****

**Simplified POS Form**

****

**K-Map Simplification**

**F:**

A picture containing text, indoor

Description automatically generated

**F’:**

**A picture containing qr code

Description automatically generated**

**A picture containing text, scoreboard

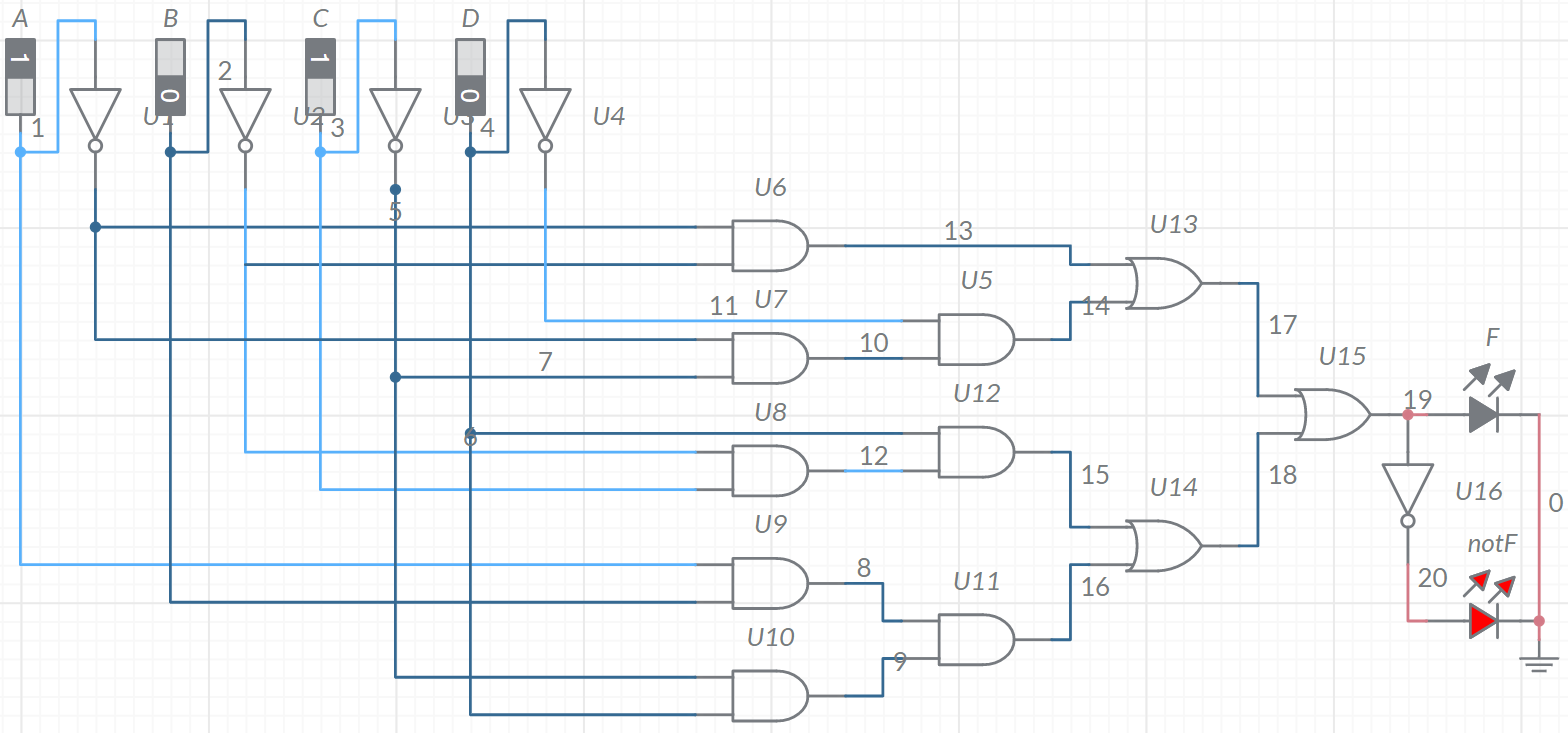
Description automatically generated**

**Simplified Forms**

F = A'B' + A'C'D' + B'CD + ABC'D = (B' + C') (A' + D) (A + B' + D') (A' + B + C)

F’ = BC + AD' + A'BD + AB'C' = (A + B) (A + C + D) (B + C' + D') (A' + B' + C + D')

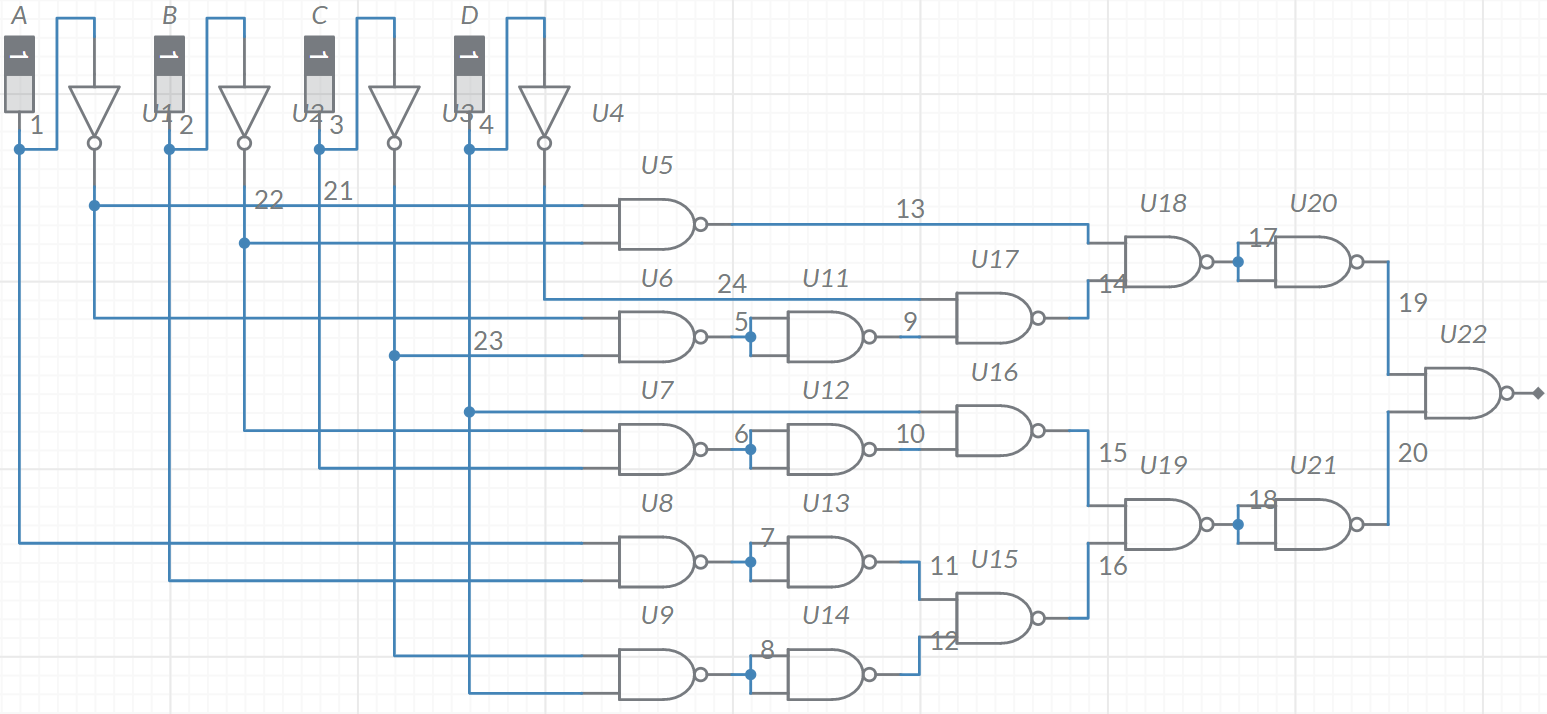
**SOP – AND-OR-INV Logic Circuit**

****

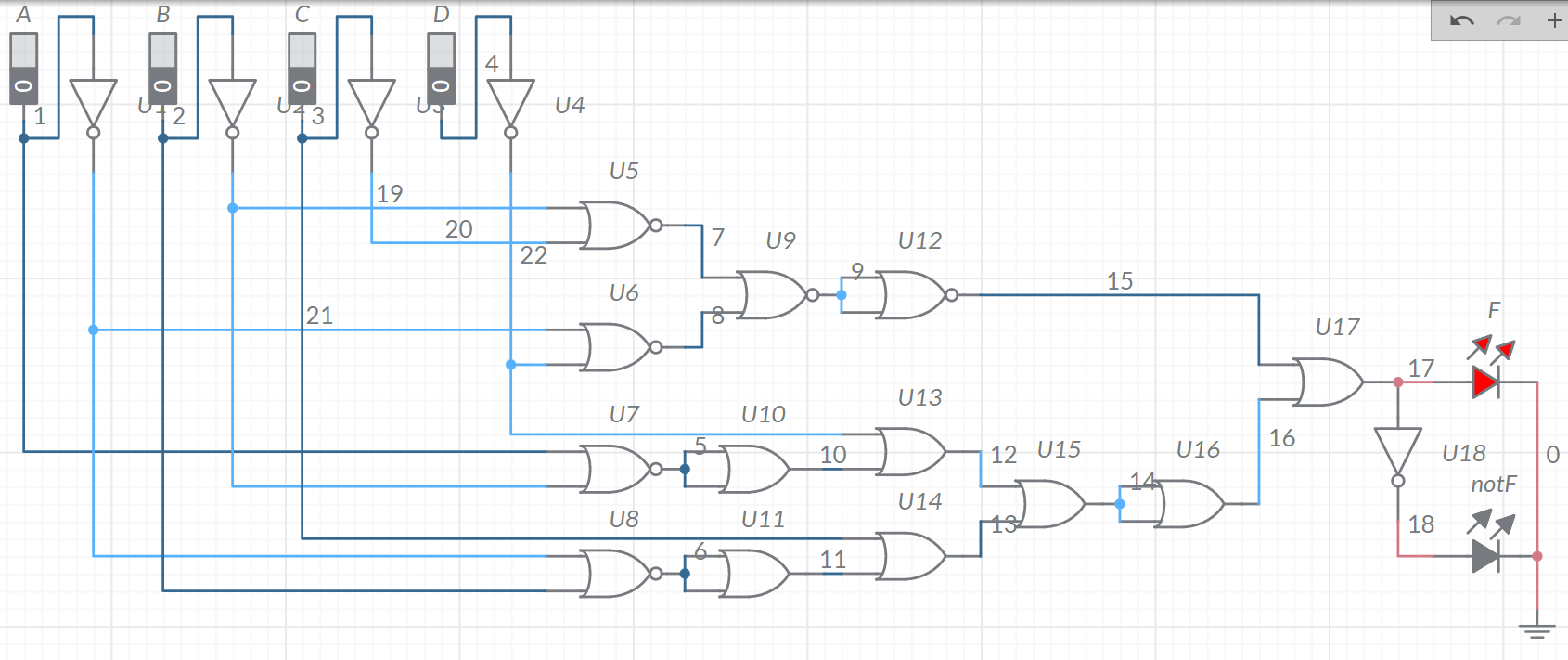
**POS – AND-OR-INV Logic Circuit**Diagram

Description automatically generated

**SOP – NAND-NAND Logic Circuit**



**POS – NOR-NOR Logic Circuit**



**Multisim Links to Circuit Diagrams**

1. SOP – AND-OR-INV logic circuit
   * <https://www.multisim.com/content/Dh2E5WYcPVSahgB6owGejF/21bds0340-sop-aoi-f-and-f/open/>
2. POS – OR-AND-INV logic circuit
   * <https://www.multisim.com/content/jr8yKr5ADsHAasgvYdj5Fo/21bds0340-pos-oai-fand-f/open/>
3. SOP – NAND-NAND logic circuit
   * <https://www.multisim.com/content/qTEa9pUv3tVrEoDRh5juhG/21bds0340-sop-nand-f/open/>
4. POS – NOR-NOR logic circuit
   * <https://www.multisim.com/content/DtaT8SneEbofyLkTobGae7/21bds0340-pos-nor-f/open/>

**Justification for Optimized Circuit**

The circuits constructed by NAND and NOR gates are optimised as those gates are cheaper and universal, meaning that they can replace any other gates.

**Verilog Code: Gate Level**

// Gate Level Model

module regno(a, b, c, d, f);

    input a, b, c, d;

    output f;

    not(nota, a);

    not(notb, b);

    not(notc, c);

    not(notd, d);

    and(prod1, nota, notb);

    and(prod2\_1, nota, notc);

    and(prod2, prod2\_1, notd);

    and(prod3\_1, notb, c);

    and(prod3, prod3\_1, d);

    and(prod4\_1, a, b);

    and(prod4\_2, notc, d);

    and(prod4, prod4\_1, prod4\_2);

    or(sum1, prod1, prod2);

    or(sum2, prod3, prod4);

    or(f, sum1, sum2);

endmodule

**Verilog Code: Data Flow**

// Data Flow Model

module design(a, b, c, d, f);

    input a, b, c, d;

    output f;

    assign nota = ~a;

    assign notb = ~b;

    assign notc = ~c;

    assign notd = ~d;

    assign prod1 = nota & notb;

    assign prod2\_1 = nota & notc;

    assign prod2 = prod2\_1 & notd;

    assign prod3\_1 = notb & c;

    assign prod3 = prod3\_1 & d;

    assign prod4\_1 = a & b;

    assign prod4\_2 = notc & d;

    assign prod4 = prod4\_1 & prod4\_2;

    assign sum1 = prod1 | prod2;

    assign sum2 = prod3 | prod4;

    assign f = sum1 | sum2;

endmodule

**Test Bench**

module testbench;

reg a, b, c, d;

wire f;

regno r(a, b, c, d, f);

initial begin

a = 0; b = 0; c = 0; d = 0;

#50

a = 0; b = 0; c = 0; d = 1;

#50

a = 0; b = 0; c = 1; d = 0;

#50

a = 0; b = 0; c = 1; d = 1;

#50

a = 0; b = 1; c = 0; d = 0;

#50

a = 0; b = 1; c = 0; d = 1;

#50

a = 0; b = 1; c = 1; d = 0;

#50

a = 0; b = 1; c = 1; d = 1;

#50

a = 1; b = 0; c = 0; d = 0;

#50

a = 1; b = 0; c = 0; d = 1;

#50

a = 1; b = 0; c = 1; d = 0;

#50

a = 1; b = 0; c = 1; d = 1;

#50

a = 1; b = 1; c = 0; d = 0;

#50

a = 1; b = 1; c = 0; d = 1;

#50

a = 1; b = 1; c = 1; d = 0;

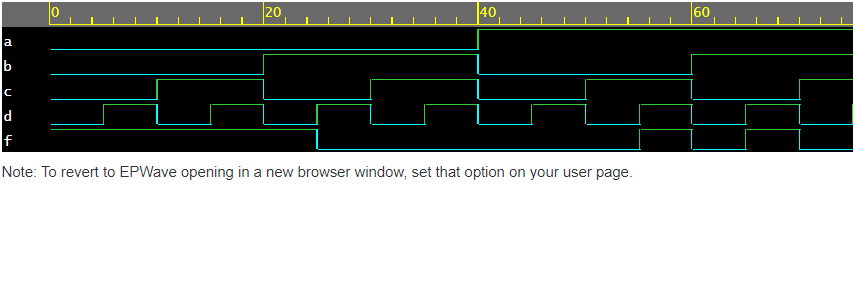
#50

a = 1; b = 1; c = 1; d = 1;

end

endmodule

**Snip of Output Waveform**



**Online Verilog Code Simulation Link**

<https://www.edaplayground.com/x/9BLy>

**Result**

I have successfully completed and simulated the combinational circuit. The result of the circuit was as expected from the truth table. I have derived functional expressions from the truth table using Boolean laws and K-map and got the same results in both techniques. I have simulated the circuit using a Multisim online simulator.

17 two input NAND gates are used in SOP F circuit simulation and 13 two input NOR gates are used in POS F circuit simulation. Both SOP and POS are using the same number of logic gates so, we can use any one of them for circuit simulation

**Inference**

I have derived functional expressions using my registration number (21BDS0340) both in SOP and POS form. I have learnt to derive a truth table from functional expressions. I learnt Boolean expressions simplification using Boolean laws and K-map technique.