**EE1005 – Digital Logic Design**

**Assignment – 1**

**Summer 2024**

**Instructor:** Muhammad Adeel Tahir **Sections:** BSE-9A, BSE-9B

|  |  |
| --- | --- |
| **Maximum Marks:** 155 marks | **Due Date:** 6th July 2024 |

* Partially or fully **copied assignments** will be marked as **zero**.
* Only **handwritten** solution on **A4 page** will be accepted.
* Submission on the GCR by the deadline is **Compulsory.**
* **Late submissions are not allowed. In case of late submission, assignment will not be accepted.**
* Clearly indicate all the calculations in your solution. No points will be awarded in case of missing calculations.
* You can submit your assignment **during the class** on due date. But submitting on GCR as mentioned is compulsory.
* **Proper calculations including k-map and circuit diagram labelling at each output, simplifications if any are to be implemented, missing steps will receive zero marks in that question straight away.**
* **Only eligible handwriting will be checked, the question shall be liable to receiving a 0 if the not readable at all.**
* **A viva of this assignment will take place and hence not being able to explain your questions will lead to 0 in that specific question.**

**Marks Distribution:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Questions** | **1** | **2** | **3** | **4** | **5** | **Total** |
| **Marks** | 40 | 30 | 20 | 45 | 20 | **155** |

**Question 1: Number Systems, Operations and Codes**

**Outcomes: You must have knowledge of the following after this section for your exam point of view:**

Number Systems and their conversions, Signed Numbers

Convert, express, and add BCD numbers, and convert BCD to decimal

Gray Code conversions

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Parts** | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** | **M** | **Total** |
| **Marks** | 3 | 5 | 4 | 3 | 2 | 2 | 3 | 1 | 3 | 2 | 3+2=5 | 4 | 3 | 40 |

1. Express the following decimal numbers as a sum of the values of each digit:
   1. 568.23
   2. 70
   3. 67.924
2. Perform the following base conversions (show proper working):
   1. to base 7
   2. a) 1001101.0101112 b) 1100101001010111 c) 111111000101101001 to base 16

* 1. to base 3
  2. to base 5
  3. to base 10

1. Convert to hexadecimal and then to binary:
2. Convert to base 6: 3BA.2514 (do all of the arithmetic in decimal)
3. Add −11 and −20 in 1’s complement (show proper working)
4. Add −8 and +19 in 2’s complement (show proper working and steps for your own understanding)
5. Express the decimal number -39 as an 8-bit number in the sign-magnitude, 1’s complement, and 2’s complement forms
6. Determine the decimal value of this signed binary number expressed in sign-magnitude: 10010101.
7. Subtract in binary. Place a 1 over each column from which it was necessary to borrow.
   1. (a) 111101 − 10011
   2. (b) 10000 − 11
   3. (c) 111001 – 1011
8. Add the signed numbers: 01000100, 00011011, 00001110, and 00010010 (show proper working) **Hint:** Proceed with step by step additions, first add two numbers and then third and so on.
9. Construct a table for 4-3-2-1 weighted code and write 9154 using this code. Is it possible to construct a 5-3-1-1 weighted code? A 6-4-1-1 weighted code? Justify your answers.
10. Add the following BCD numbers: (Show proper workings step by step). When are BCD numbers invalid, explain in 1 lines only. This is to only develop your concept)
    1. (a) 1001 + 0100
    2. (b) 1001 + 1001
    3. (c) 00010110 + 00010101
    4. (d) 01100111 + 01010011
11. The **Gray code** is unweighted and is not an arithmetic code; that is, there are no specific weights assigned to the bit positions. Imagine Gray code like a dimmer switch where you can only move one notch at a time. To convert it to regular binary:
12. Copy the leftmost bit.
13. For other bits, Add each binary code bit generated to the Gray code bit in the next adjacent position. Discard carries.

This keeps things smooth when switching between the two codes. A diagram of numbers and arrows

Description automatically generated

Now solve the following by showing working as above:

1. Convert the binary number 11000110 to Gray code.
2. Convert the Gray code 10101111 to binary.
3. Convert 11011,1001010, 111101110110 to Gray codes.

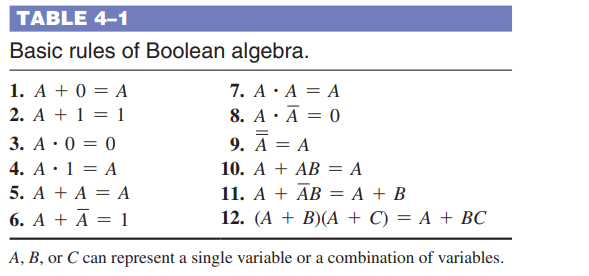
**Question 2: Boolean Algebra**

**Outcomes: You must have knowledge of the following after this section for your exam point of view:**

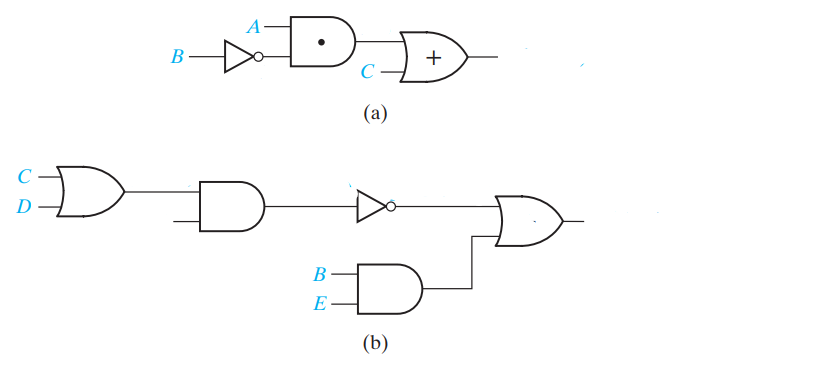
**Apply different Boolean laws to solve Boolean simplification and other related questions.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parts** | **A** | **B** | **C** | **D** | **E** | **Total** |
| **Marks** | 2+2=4 | 2+2+2+2+2=10 | 2.5+3.5=5 | 2.5+2.5=5 | 5 | 30 |

Before you proceed: You will need to remember these Boolean algebra laws in order to solve related questions in exam. (If any other are missing, you may consult the internet)



**Questions:**

1. Solve the following with mentioned laws:
   1. Apply the associative law of addition to the expression A + (B + C + D).
   2. Apply the distributive law to the expression A(B + C + D).
2. Apply DeMorgan’s theorems to the expressions:
   1. (XYZ)’
   2. (X + Y + Z)’
   3. (WXYZ)’
   4. (W + X + Y + Z)’
3. Obtain the equations from the following circuit diagrams, then implement their truth tables. 
4. Perform the following:
   1. Factor C′D + C′E′ + G′H and mention laws that were used in order to obtain the final equation. (So you can practice how factoring works during simplification)
   2. Find the inverse of the following function: F = A′B + AB′. Verify that your resultant Boolean equation is indeed the inverse to this equation via a truth table.
5. Simplify the following to obtain the Boolean equation result (**mention laws)**:
   * 1. A’BC + AB’C’ + A’B’C’ + AB’C + ABC
     2. (AB+AC)’ + A’B’C

**Question 3: K- Maps and POS, SOP Forms, Circuits**

**Outcomes: You must have knowledge of the following after this section for your exam point of view:**

**Be able to correctly use K-Maps and find out the POS SOP expressions.**

**Be able to find expressions from the circuits and simplify them.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parts** | **A** | **B** | **C** | **D** | **Total** |
| **Marks** | 1+1+1=3 | 2 | 1+3+2+1+1+5 | 4 | 20 |

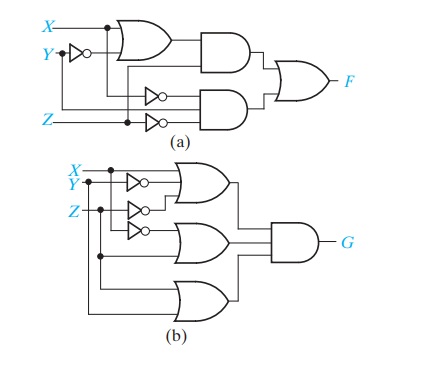
Before you proceed:

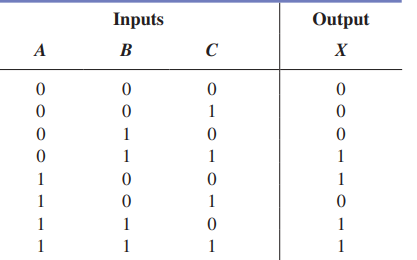
* “Don’t care”: A combination of input literals that cannot occur and can be used as a 1 or a 0 on
* a Karnaugh map for simplification.
* Minimization: The process that results in an SOP or POS Boolean expression that contains the fewest possible literals per term.
* Product-of-sums (POS) : A form of Boolean expression that is basically the ANDing of ORed terms.
* Product term The Boolean product of two or more literals equivalent to an AND operation.
* Sum-of-products (SOP): A form of Boolean expression that is basically the ORing of ANDed terms.

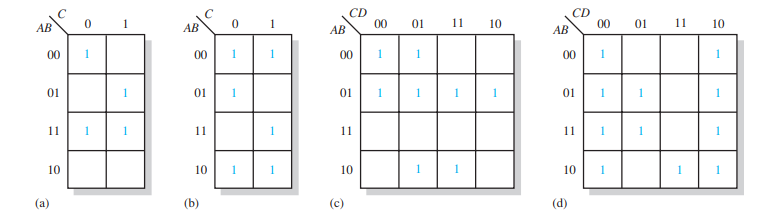
|  |  |
| --- | --- |
| **A 3-Variable K-Map** |  |
| **A 4-Variable K-Map** |  |

**Questions:**

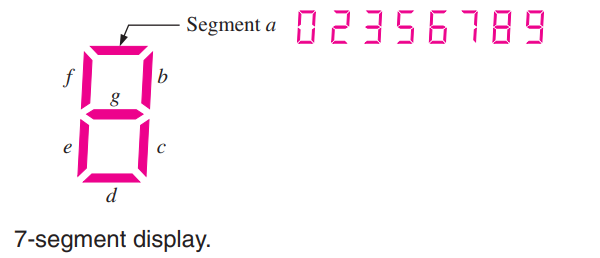
1. Convert each of the following Boolean expressions to SOP form:
   1. AB + B(CD + EF)
   2. (A + B)(B+C+D)
   3. ((A+B)’+C)’
2. Show (through simplification of Boolean equations) that the circuits below realize the same function:



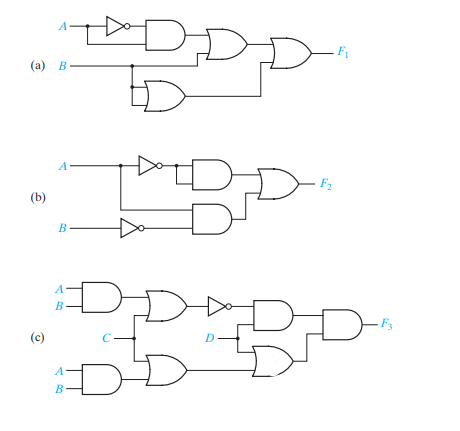
1. K-maps implementation:
   1. For the table given below, determine the SOP expression and the equivalent standard POS expression. After this use K-Maps to verify your expressions. 
   2. Draw K-map of following expression:
   3. For each of the following K-Maps that are given below, group all the 1’s correctly and write the expressions obtained after the grouping. Make sure you show group in a neat and clean manner as given in the example in the start of this section (Ref: Before you proceed):



* 1. Find SOP expression using the K-Map:
  2. Find SOP expression for: B’C’D’ + A’BC’D’ + A’B’CD+AB’CD+A’BCD’+ABCD’+AB’CD’ (Hint: Expand the first term before mapping it)
  3. **Don’t Care conditions:** In a 7 segment display, the segment a is activated for the 0,2,3,5,6,7,8,9 (Ref to diagram for your ease). For each digit where segment 'a' is active, write a product term with the variables ABCD (Hint: remember that BCD code is being used to represent each digits) and after obtaining the product expression, use K-Map to map the simplified expression for it. (keep in mind we will take those digits that are not important to us as Don’t Care while mapping)



1. Below are given circuits, find the output and design a simpler circuit that has the same output. (**Hint**: Find outputs F1,F2,F3 and then perform the simplifications and redesign the circuits)



**Question 4: Combinational Logic (Design & analysis)**

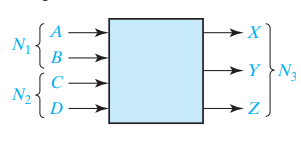
**Outcomes: You must have knowledge of the following after this section for your exam point of view:**

**Be able to design different combinational circuits from different statements.**

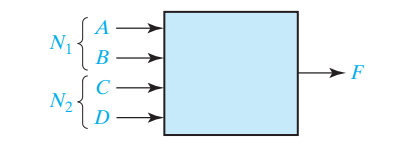
**Be able to analyze the given circuits as well.**

**Be able to design different circuits such as adders, subtractors, parity generators etc.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Parts** | **A** | **B** | **C** | **D** | **E** | **F** | **Total** |
| **Marks** | 5 | 5 | 10 | 5 | 10 | 10 | 45 |



1. An adder is to be designed which adds two 2-bit binary numbers to give a 3-bit binary sum. Find the truth table for the circuit. The circuit has four inputs and three outputs as shown in the diagram. Use k-maps to obtain the expressions and draw its logic circuit.
2. Design an error detector for 6-3-1-1 binary-coded-decimal digits. The output (F) is to be 1 iff the four inputs (A, B, C, D) represent an invalid code combination. (**Hint**: First find out the truth table and see which combinations are invalid i.e they don’t follow a pattern of course. Where an error occurs, we put F=1). Implement the truth table, find the SOP terms and then implement the circuit diagram. (Use k-maps for simplifications)
3. Many offices and buildings use combination locks to control entry. As the design engineer of the Wonderful Door Security Company, you are asked to implement a door security system by using a card reader. There are four inputs to the card reader: inputs X, Y, and Z are used to validate the correct door code, and input V is used to check if the card reader is still valid. After the card reader is being read by the system, there are three outputs to this system: alarm (A), door open (D), and Error (E). Door (D) will only open when the decimal value of the binary inputs (x, y, z) is odd AND the card reader is valid. The Error (E) signal goes on when the code on the card is correct (i.e. decimal value equal to odd) but the card is no longer valid. Finally, the alarm (A) will trigger when the code is incorrect. Show your final design in canonical product of sum form.
4. Develop a logic circuit with four input variables that will only produce a 1 output when exactly three input variables are 1s.
5. A gas detector works as follows: It outputs a 5-bit signal where all bits are zero if the gas concentration is below 10 parts per million (ppm). When the concentration reaches 10 ppm or more, the first bit becomes 1; when the concentration reaches 20 ppm or more, the first two bits become 1; when the concentration reaches 30 ppm or more, the first three bits become 1, and this pattern continues. The gas detector outputs a 5-bit signal, labeled as A, B, C, D, and E. These bits are inputs to a device that generates two outputs, Y and Z. **(a)** Write an equation for the output Y if Y should be 1 only when the gas concentration is less than 30 parts per million (ppm). **(b)** Write an equation for the output Z if Z should be 1 only when the gas concentration is at least 20 parts per million (ppm) but less than 50 parts per million (ppm). Draw the circuit diagram after implementing the truth table.
6. A switching circuit has four inputs as shown. A and B represent the first and second bits of a binary number N1. C and D represent the first and second bits of a binary number N2. The output is to be 1 only if the product N1 × N2 is less than or equal to 2. **(a)** Find the minterm expansion for F. **(b)** Find the maxterm expansion for F. Express your answers in both decimal notation and algebraic form. Draw the circuit diagram after implementing the truth table.



**Question 4: Decoders**

**Outcomes: You must have knowledge of the following after this section for your exam point of view:**

**Be able to work with decoders**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parts** | **A** | **B** | **C** | **D** | **Total** |
| **Marks** | 2+2+1=5 | 5 | 5 | 2.5+2.5=5 | 20 |

**Part a:**

Design a combinational circuit that takes 3-bit input and at the output it multiplies it by 3 and adds 1 to have the final output. Design this circuit using only 2 × 4 decoders and basic logic gates if necessary.

a) Properly label and fill the truth table in neat and clean manner for this design.

b) Design the circuit diagram for this problem.

c) Explain the approach in your own words (5-8 lines max). Wrong explanation leads to 0.

**Part b:**

Using a decoder and external gates, design the combinational circuit defined by the following

three Boolean functions:

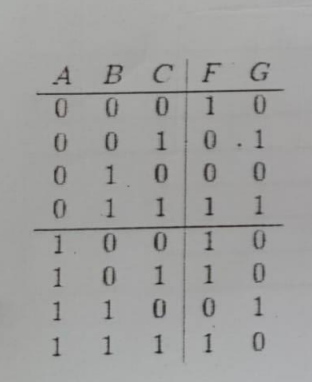
F1 = x'yz' + xz

F2 = xy'z' + x'y

F3 = x'y'z' + xy

**Part c:**

Using two 2:4 decoders-with-enable. Add wires, one not gate, and two or gates to implement the functions F and G given in the truth table.



**Part d:**

1. Implement using 3:8 Decoder and an OR gate.
2. F1 (x,y,z) = ∑(2,5,7) , F2 (x,y,z) = ∑(2,3,4), F3 = ∑(0,6,7) . Implement using appropriate decoder and OR gate.

**Question 5: NAND & NOR Implementation**

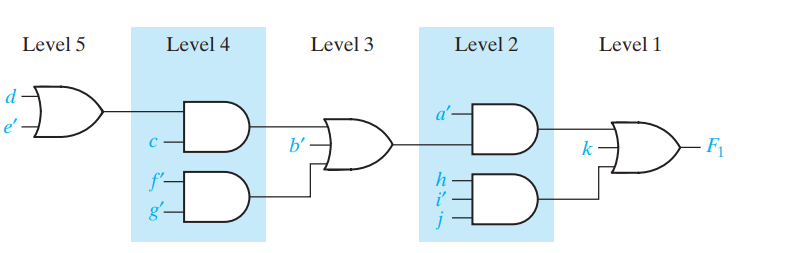
**Outcomes: You must have knowledge of the following after this section for your exam point of view:**

**Be able to implement circuits with NAND NOR gates**

**Be able to re-design the given circuits using NAND NOR implementation**

**Hint**: Most of the questions involve simplifying them like you did in previous questions and then simply implement them using the NAND/NOR gates.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Parts** | **A** | **B** | **C** | **D** | **E** | **F** | **Total** |
| **Marks** | 5 | 5 | 5 | 5 | 5 | 5 | 30 |

1. Simplify the following function, and implement them with two-level NAND gate circuits:
2. Implement the following using 2 Input NOR gates only: F =∑(0,3,12,15)
3. Convert the following circuit to NAND only implementation by re-drawing its circuit diagram. Label properly. 
4. Realize Z = AE + BDE + BCEF using only two-input NOR gates. Use as few gates as possible.
5. Implement f(x, y, z) = Σ m(0, 1, 3, 4, 7) as a two-level gate circuit, using a minimum number of gates. Use **NAND** gates only.
6. Implement f(a, b, c, d) = Σ m(3, 4, 5, 6, 7, 11, 15) as a two-level gate circuit, using a minimum number of gates. Use **NOR** gates only