

|  |  |
| --- | --- |
| **ASSIGNMENT** | |
| **Course Code** | CSC203A |
| **Course Name** | Logic Design |
| **Programme** | B.Tech |
| **Department** | CSE |
| **Faculty** | FET |

#### 

|  |  |
| --- | --- |
| **Name of the Student** | Satyajit Ghana |
| **Reg. No** | 17ETCS002159 |
| **Semester/Year** | 03/2018 |
| **Course Leader/s** | Mr. Narasimha Murthy |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Declaration Sheet** | | | | | | | | |
| Student Name | Satyajit Ghana | | | | | | | |
| Reg. No | 17ETCS002159 | | | | | | | |
| Programme | B.Tech | | | | | Semester/Year | 03/2018 | |
| Course Code | CSC203A | | | | | | | |
| Course Title | Logic Design | | | | | | | |
| Course Date |  | | to | |  | | | |
| Course Leader | Mr. Narasimha Murthy | | | | | | | |
| **Declaration**  The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly. | | | | | | | | |
| Signature of the Student | |  | | | | | Date |  |
| Submission date stamp  (by Examination & Assessment Section) | |  | | | | | | |
| Signature of the Course Leader and date | | | | Signature of the Reviewer and date | | | | |
|  | | | |  | | | | |

# **Contents**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[**Declaration Sheet** ii](#_Toc525523637)

[**Contents** iii](#_Toc525523638)

[**List of Tables** **Error! Bookmark not defined.**](#_Toc525523639)

[**List of Figures** iv](#_Toc525523640)

[**Question No. 1** 5](#_Toc525523641)

[A 1.1 Introduction: 5](#_Toc525523642)

[A 1.2 Arithmetic subtraction using BCD and Excess-3 codes: 5](#_Toc525523643)

[A 1.3 Conclusion: 6](#_Toc525523644)

[**Question No. 2** 7](#_Toc525523645)

[B 1.1 Write the truth table for the combination circuit described above: 7](#_Toc525523646)

[B 1.2 Express the behavior of the circuit in the canonical Sum of Product (SoP) form 7](#_Toc525523647)

[B 1.3 Reduce the above SoP expression to the minimized form using K-Map: 8](#_Toc525523648)

[**Question No. 3** 10](#_Toc525523649)

[B 2.1 Write the truth table for the combination circuit described above: 10](#_Toc525523650)

[B 2.2 Derive the minimized Product of Sum (PoS) expression using K-Map: 10](#_Toc525523651)

[B 2.3 Analyze the importance of don’t care condition in the process of minimization: 11](#_Toc525523652)

# **List of Figures**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title of the figure** | **Pg.No.** |
| Figure B1.1 | Circuit for B1 | 8 |
| Figure B1.2 | Circuit for B1 with Don’t Care | 9 |
| Figure B1.3 | Circuit for B2 | 10 |
|  |  |  |

# 

# **Question No. 1**

**Solution to Question No. 1 Part A:**

## A 1.1 Introduction:

Computers and essentially any digital logic can be thought of a really dumb machine which is really good at following instructions, the downside of being *dumb* is that, first of all the data is composed of bits, which can be either 1, or 0, that’s only two states, and also that this dumb machine can only add, but is really good at it, subtraction can also be thought of as addition, where we are adding a number to another negative number, here we use complements to represent negative numbers.

I would like to think of complements as *rotation* of numbers, it’s essentially rotation in its core, suppose we would like to represent in complement and assuming we deal with only 1 digit, it will be like counting from the end times, . This seems intuitive and a very clever way to subtract without knowing the actual knowledge of negative numbers, Clever Humans, Dumb Computers, beautiful combination. Let’s go by this quote,

“There is a race between mankind and the universe. Mankind is trying to build bigger, better, faster and more foolproof machines. The universe is trying to build bigger, better, and faster fools. So far the universe is winning

– Albert Einstein”

## A 1.2 Arithmetic subtraction using BCD and Excess-3 codes:

To illustrate and compare the two forms of codes in terms of subtraction we will try to perform

Arithmetic subtraction using BCD

It can also be written as, so we convert into its complement which is, this number is then converted to BCD and then added to in BCD

The Binary Code is not a valid BCD, so we add to it to get,

Now the answer obtained is in BCD, or decimal which is complemented, and since we did not observe any the result obtained is negative

Hence, which was the required answer.

Using the complement in such a way seems more *human* than a *machine*, let’s look at a better way, using complement, which a machine can do quite easily, we’ll convert into it’s BCD complement and add it to BCD

Since there was no we know for sure that our answer is negative, now we do complement of our result and those individual groups that had a carry are added to them others are added with , any carries found henceforth are dropped.

The result obtained is in BCD, converted to Decimal is and since we know its negative the final answer is , which was our desired result.

Arithmetic subtraction using Excess-3

Taking the same example, we convert our decimal into Excess-3 code, the operands are converted to their respective code. in XS-3 Code is complemented using complement since it’s a negative number and then they are added.

Now we invert the bit’s,

is not a valid XS-3 code, so we add to it, so our final result becomes , which is and since there was no the number is negative, hence

## A 1.3 Conclusion:

Excess-3 Code is self-complementing, if we compare the procedure used for BCD and XS-3, there is that less extra steps, i.e. the number in BCD should be 9’s complemented from decimal and then converted back to BCD, which in XS-3, the 9’s complement is same as the 1’s complement, i.e. changing to and vice-versa, which makes it easier to perform the calculation.

Another conclusion drawn is that the number systems which are weighted codes, if the sum of the weights is equal to 9, they are self-complementing, in XS-3 the sum of weights is , , similarly in 2421 the sum of weights is, .

# **Question No. 2**

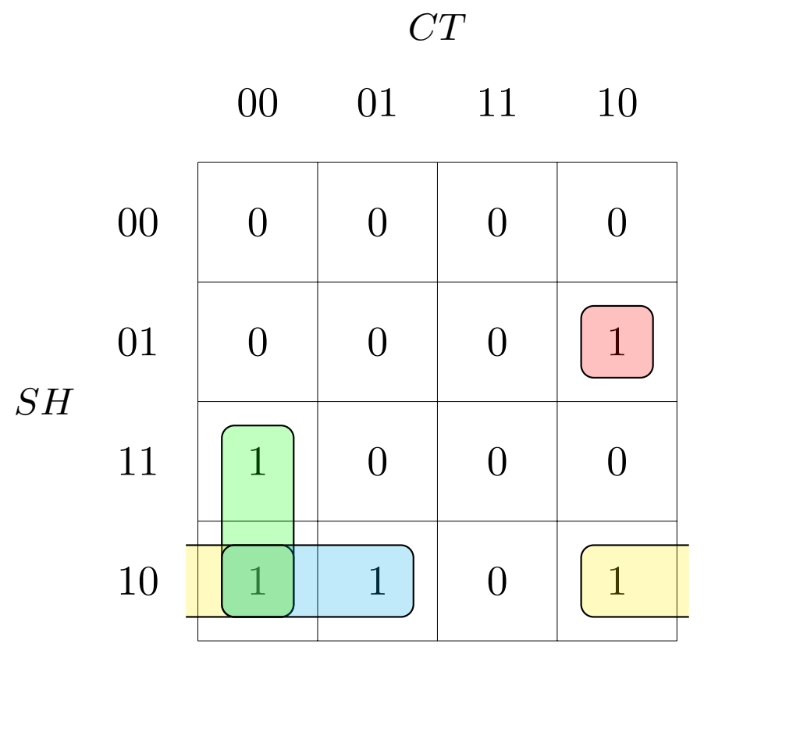
**Solution to Question No. 1 Part B:**

## B 1.1 Write the truth table for the combination circuit described above:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S** | **H** | **C** | **T** | **X** |
| 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 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

## B 1.2 Express the behavior of the circuit in the canonical Sum of Product (SoP) form

## B 1.3 Reduce the above SoP expression to the minimized form using K-Map:

Without using Don’t Care:

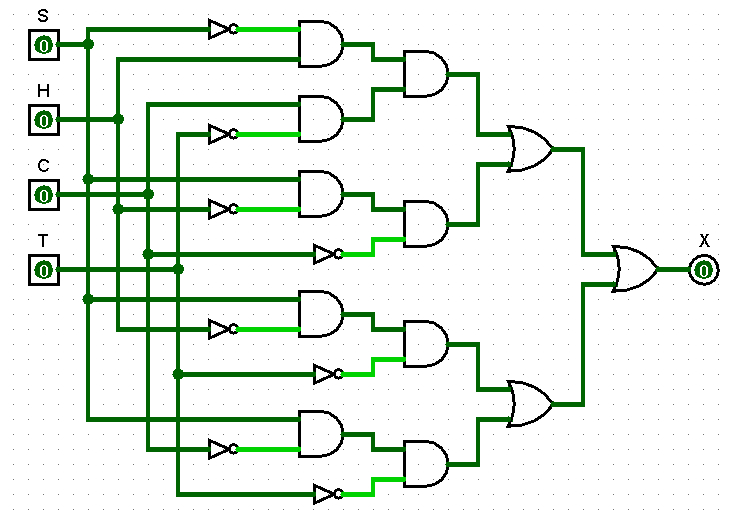
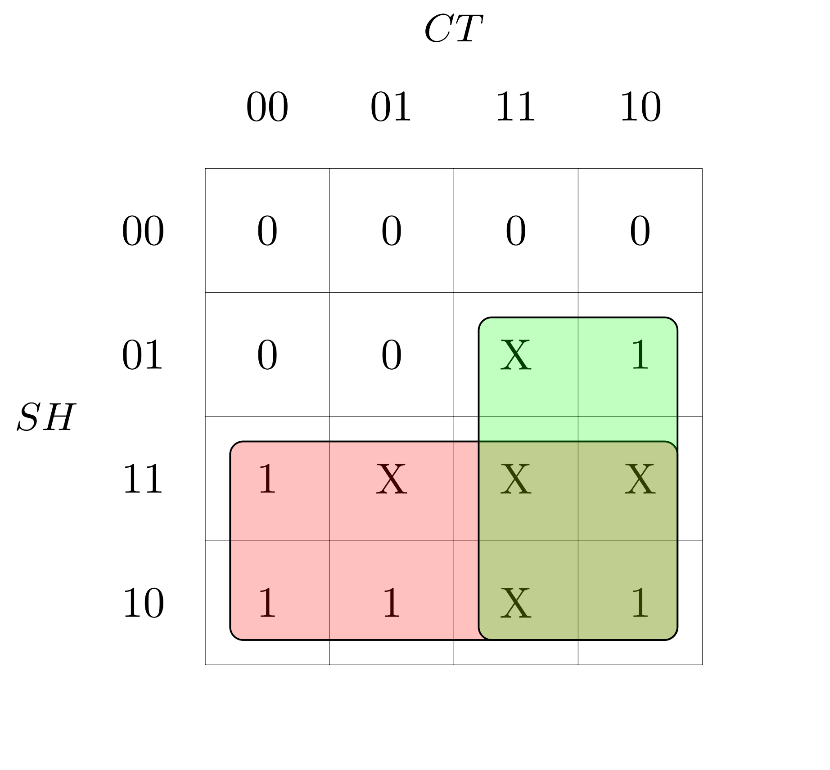
Reduced form:

Figure B1.1 Circuit for B1

Using Don’t Care:

Truth Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S** | **H** | **C** | **T** | **X** |
| 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 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | X |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | X |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | X |
| 1 | 1 | 1 | 0 | X |
| 1 | 1 | 1 | 1 | X |



Reduced form:

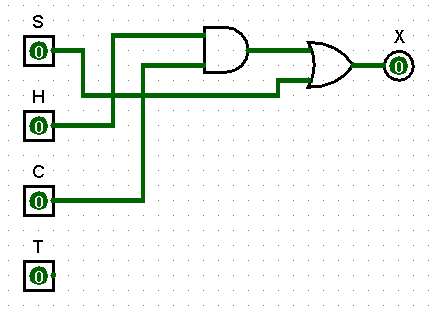


Figure B1.2 Circuit for B1 with Don’t Care

# **Question No. 3**

**Solution to Question No. 2 Part B:**

Data:

Here A = 2, B = 3, C = 4, D = 5,

Given Minimum Credits = 6

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B 2.1 Write the truth table for the combination circuit described above:  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **A** | **B** | **C** | **D** | **F** | | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 1 | 0 | | 0 | 0 | 1 | 0 | 0 | | 0 | 0 | 1 | 1 | X | | 0 | 1 | 0 | 0 | 0 | | 0 | 1 | 0 | 1 | X | | 0 | 1 | 1 | 0 | X | | 0 | 1 | 1 | 1 | X | | 1 | 0 | 0 | 0 | 0 | | 1 | 0 | 0 | 1 | X | | 1 | 0 | 1 | 0 | 1 | | 1 | 0 | 1 | 1 | X | | 1 | 1 | 0 | 0 | 0 | | 1 | 1 | 0 | 1 | X | | 1 | 1 | 1 | 0 | X | | 1 | 1 | 1 | 1 | X | | B 2.2 Derive the minimized Product of Sum (PoS) expression using K-Map:   Reduced form:    Figure B1.3 Circuit for B2 |

## B 2.3 Analyze the importance of don’t care condition in the process of minimization:

The don’t care terms are important enough to be taken into consideration since it helps in reducing the expression even further by helping in grouping the terms together. Don’t care is defined as those inputs for which the designer does not care, and the output for them is arbitrary.

The way it works is that each of these don’t care conditions can be represent 2 values, (0 or 1), assuming that a function has don’t care conditions or don’t care bits, there can be distinct functions, each of the bit gives, so, and increases exponentially, now since there are so many combinations that can be used to obtain the minimal expression, thereby it decreases the number of inputs for the circuit making implementation faster.

**Bibliography**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_