RAMAIAH UNIVERSITY OF APPLIED SCIENCES

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

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 $-n$ in $10's$ complement and assuming we deal with only 1 digit, it will be like counting from the end n times, $9, 8, 7, \ . \ . \ . \ , n^{th}$. 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

$$323 - 414 = -091$$

Arithmetic subtraction using BCD

It can also be written as $323 + (-414)$, so we convert -414 into its $9's$ complement which is $999 - 414 = 585$, this number is then converted to BCD and then added to 323 in BCD

$$\begin{array}{ccc} 0011 & 0010 & 0011 \\ 0101 & 1000 & 0101 \\ \\ 1000 & 1010 & 1000 \end{array}$$

The Binary Code 1010 is not a valid BCD, so we add 0110 to it to get,

$$\begin{array}{ccc} 1000 & 1010 & 1000 \\ & 0110 & \\ 0001 & & \\ 1001 & 0000 & 1000 \end{array}$$

Now the answer obtained is in BCD, or decimal 908 which is $9's$ complemented, and since we did not observe any $end - arround - carry$ the result obtained is negative

$$\begin{array}{cccc} & 9 & 9 & 9 \\ - & 9 & 0 & 8 \\ = & 0 & 9 & 1 \end{array}$$

Hence $323 - 414 = -91$, which was the required answer.

Using the $9's$ complement in such a way seems more *human* than a *machine*, let's look at a better way, using $1's$ complement, which a machine can do quite easily, we'll convert 414 into it's BCD $1's$ complement and add it to BCD 323

$$\begin{array}{ccc} 0011 & 0010 & 0011 \\ 1011 & 1110 & 1011 \\ \\ 1110 & 0000 & 1110 \\ 0001 & & \\ \\ 1111 & 0000 & 1110 \end{array}$$

Since there was no $end - around - carry$ we know for sure that our answer is negative, now we do $1's$ complement of our result and those individual groups that had a carry are added 1010 to them others are added with 0000, any carries found henceforth are dropped.

$$\begin{array}{cccl} 1111 & 0000 & 1110 & \\ 0000 & 1111 & 0001 & \leftarrow 1's\ complemented \\ 0000 & 1010 & 0000 & \\ \\ 0000 & 1001 & 0001 & \end{array}$$

The result obtained is in BCD, converted to Decimal is 91 and since we know its negative the final answer is -91, which was our desired result.

<u>Arithmetic subtraction using Excess-3</u>

Taking the same example, we convert our decimal into Excess-3 code, the operands are converted to their respective $XS - 3$ code. 414 in XS-3 Code is complemented using $9's$ complement since it's a negative number and then they are added.

$$\begin{array}{ccc} 0110 & 0101 & 0110 \\ 1000 & 1011 & 1000 \\ \\ 1110 & 0000 & 1110 \\ 0001 & & \\ \\ 1111 & 0000 & 1110 \end{array}$$

Now we invert the bit's,

$$\begin{array}{ccc} 0000 & 1111 & 0001 \end{array}$$

1111 is not a valid XS-3 code, so we add 0110 to it, so our final result becomes $0000 \quad 1001 \quad 0001$, which is 91 and since there was no $end - around - carry$ the number is negative, hence -91

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 $0s$ to $1s$ 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 , $8 + 4 - 2 - 1 = 9$, similarly in 2421 the sum of weights is, $2 + 4 + 2 + 1 = 9$.

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

$$X = \bar{S}HC\bar{T} + S\bar{H}\bar{C}\bar{T} + S\bar{H}\bar{C}T + S\bar{H}C\bar{T} + SH\bar{C}\bar{T}$$

$$X = \sum(6, 8, 9, 10, 12)$$

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

Without using Don't Care:

SH \ CT	00	01	11	10
00	0	0	0	0
01	0	0	0	1
11	1	0	0	0
10	1	1	0	1

Reduced form:

$$X = \bar{S}HC\bar{T} + S\bar{H}\bar{C} + S\bar{H}\bar{T} + S\bar{C}\bar{T}$$

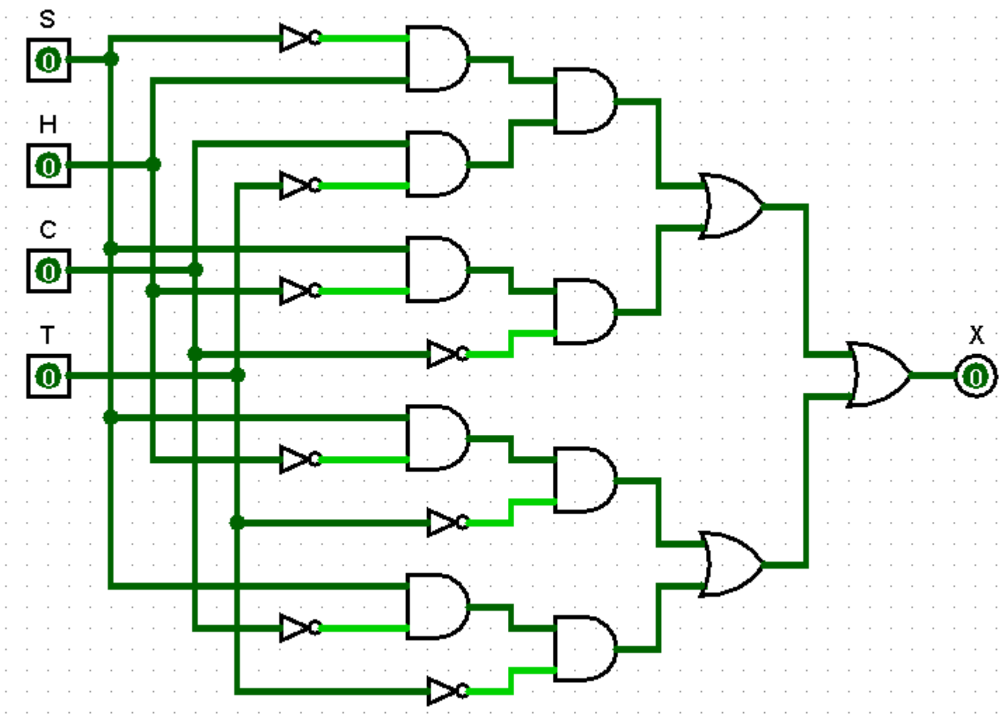

Figure B1.1 Circuit for B1

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

$$X = \sum m(6,8,9,10,12) + \sum \phi(7,11,13,14,15)$$

$SH \backslash CT$	00	01	11	10
00	0	0	0	0
01	0	0	X	1
11	1	X	X	X
10	1	1	X	1

Reduced form:

$$X = S + CH$$

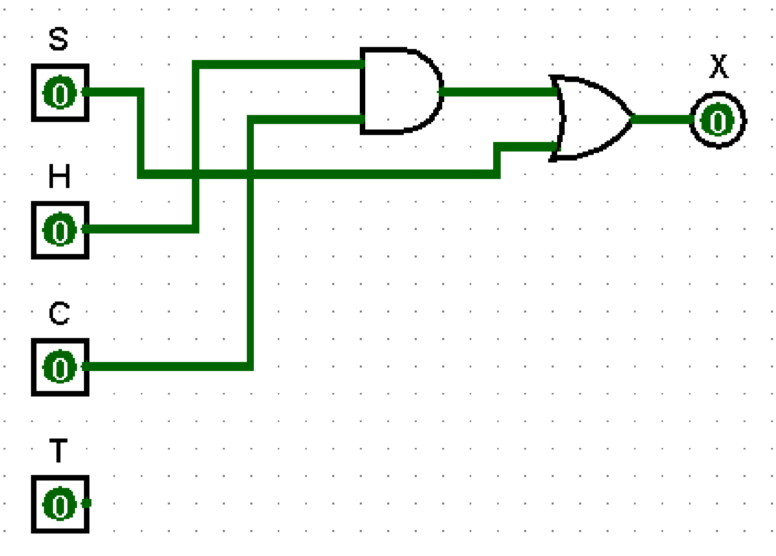

Figure B1.2 Circuit for B1 with Don’t Care

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:

AB \ CD	00	01	11	10
00	0	0	X	0
01	0	X	X	X
11	0	X	X	X
10	0	X	X	1

Reduced form:

$$F = (A) \cdot (C)$$

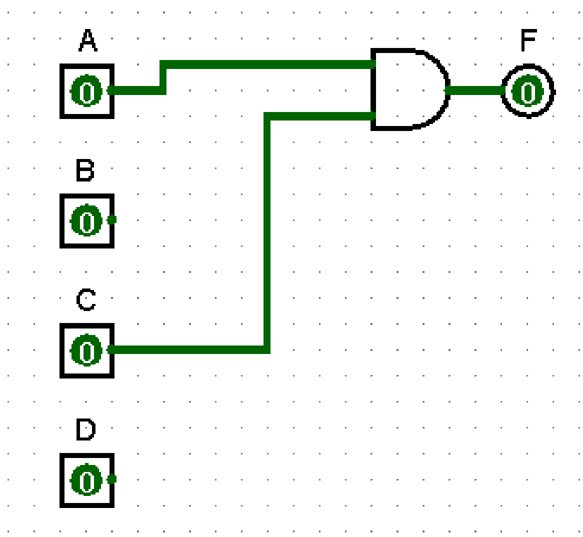

Figure B1.3 Circuit for B2

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 $'k'$ don't care conditions or $'k'$ don't care bits, there can be 2^k distinct functions, each of the k bit gives 2, so $2 \times 2 \times . . . k\ times = 2^k$, 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