Computer Organization and Systems

Assignment 1

Rutam Kathale

21 August 2023

Problem 1

Question: Subtract 314 from 402 using 7's complement method. Solve in two ways:
(a) assuming that both are base 8 numbers, and
(b) assuming that both are base 7 numbers.

Solution:
(a) First let's assume both numbers are base 8: To subtract 314 from 463, we have to essentially take the 7's complement of 314 and add it with 402 to get our desired answer. If a carry during addition is generated, we add the carry to our sum. To take 7's complement of 314, we subtract it with 777. That would be 463. So, now we add 463 with 402.

$$\begin{array}{r} 463 \\ +\ 402 \\ \hline 1065 \end{array}$$

The result we get is 1065. Since a carry bit of 1 is generated, we add 1 to 065 which becomes 066.

(b) Now, let's assume that both numbers are base 7 Taking the 7's complement of a number in a particular base is basically subtracting that number from the largest possible number in that base. For base 7, we have digits 0-6 and so, here we subtract 314 with 666. So, subtracting 314 from 666 will be 352. Then we add this number with 402.

$$\begin{array}{r} 352 \\ +\ 402 \\ \hline 1054 \end{array}$$

The sum we get is 1054. Since a carry is generated, we add 1 back to 054 and then, we get 055. So, the answer for base 7 is 055.

Problem 2

Question: Convert the decimal number 85.125 into IEEE 754 Single Precision format.
Solution: To convert 85.125, we basically convert 85.125 to binary, normalize it, determine sign bit, mantissa, and exponent (and add with bias term) and pad them to fit the determined bits according to convention.

Part1. 0.125 -> binary
$0.125 * 2 = 0.25$ (Integer part - 0)
$0.25 * 2 = 0.5$ (Integer part - 0)
$0.5 * 2 = 1.0$ (Integer part - 1)
Therefore, the binary representation of the decimal part is 0.001.

Part2: 85 in binary is 1010101 and so, 85.125 is 1010101.001

Part 3: Now, we normalize the expression (decimal after 1 place from left) into scientific notation and get the exponent. So, 1010101.001 becomes $1.010101001 * 2^6$ (the exponent is 6)

Part 4: For IEEE 754, bias is 127 added to exponent which becomes 133 (10000101 in binary).

Part 5: Since the number is positive, sign bit is 0. Mantissa is just the decimal part of expression (010101001). By IEEE 754, mantissa is 23 bits, and so we convert our mantissa to 23 bits giving us: 01010100100000000000000
So, our number (combining sign bit, exponent, and mantissa) becomes:
0 10000101 01010100100000000000000

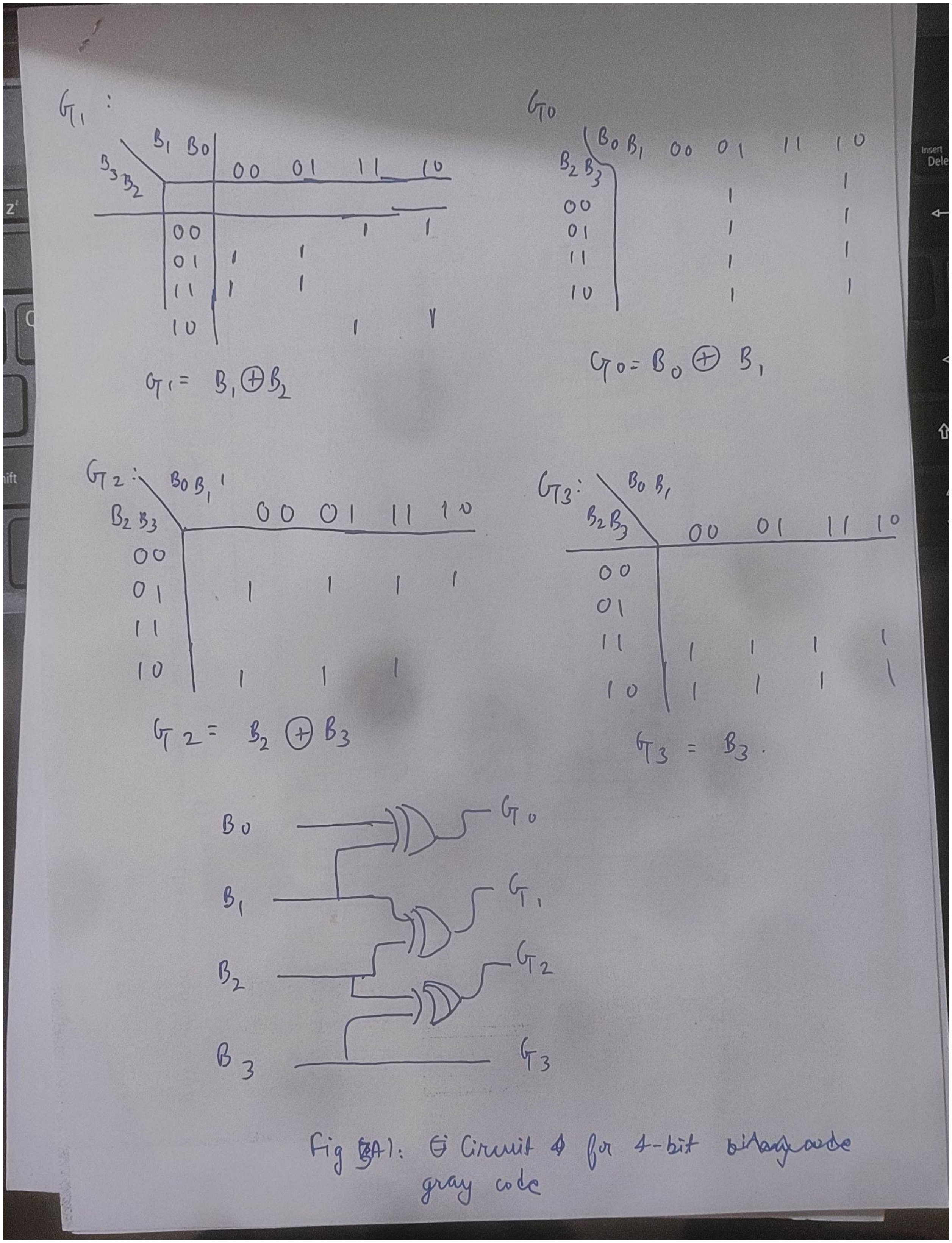
Problem 3

Question: Draw the circuit for 4-bit binary to Gray code converter using Karnaugh map simplification method.

Solution: To draw the circuit, we have to first convert the given 4-bit code to gray code.

B0	**B1**	**B2**	**B3**	**G0**	**G1**	**G2**	**G3**
0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0
0	1	0	0	1	1	0	0
1	1	0	0	0	1	0	0
0	0	1	0	0	1	1	0
1	0	1	0	1	1	1	0
0	1	1	0	1	0	1	0
1	1	1	0	0	0	1	0
0	0	0	1	0	0	1	1
1	0	0	1	1	0	1	1
0	1	0	1	1	1	1	1
1	1	0	1	0	1	1	1
0	0	1	1	0	1	0	1
1	0	1	1	1	1	0	1
0	1	1	1	1	0	0	1
1	1	1	1	0	0	0	1

We can determine relation between $B_0...B_3$ and $G_0,....G_3$ using k-maps and draw circuits to represent those relations. (Both can be found on next page).



G_1 :

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00			1	1
01	1	1		
11	1	1		
10			1	1

$G_1 = B_1 \oplus B_2$

G_0

$B_2B_3 \backslash B_0B_1$	00	01	11	10
00		1		1
01		1		1
11		1		1
10		1		1

$G_0 = B_0 \oplus B_1$

G_2 :

$B_2B_3 \backslash B_0B_1$	00	01	11	10
00				
01	1	1	1	1
11				
10	1	1	1	

$G_2 = B_2 \oplus B_3$

G_3 :

$B_2B_3 \backslash B_0B_1$	00	01	11	10
00				
01				
11	1	1	1	1
10	1	1	1	1

$G_3 = B_3$.

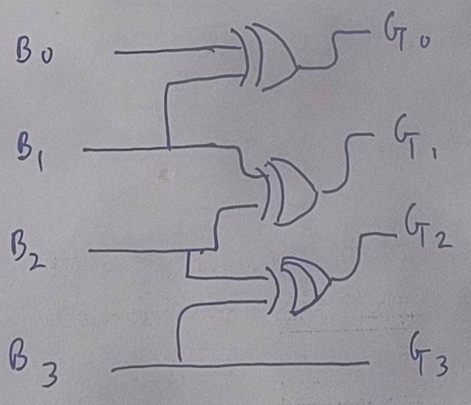


Fig 3A1: ~~G~~ Circuit ~~4~~ for 4-bit ~~binary code~~ gray code

Problem 4

Question: Use Quine-McCluskey Tabulation method to find the minimal SOP representation of f(A, B, C, D, E) = (1, 2, 3, 5, 9, 10, 11, 18, 19, 20, 21, 23, 25, 26, 27).
Solution: First we group the minterms based on the number of 1s

Table 1: Step: Grouping

Group 1	
1	00001
2	00010
Group 2	
3	00011
5	00101
9	01001
10	01010
18	10010
20	10100
Group 3	
11	01011
19	10011
21	10101
25	11001
26	11010
Group 4	
23	10111
27	11011

Now we group pairs based on differences in their bit values. If the difference is 1, we group them, else ignore them.

Problem 4 continued on next page. . .

Table 2: Step: Merging

Group 11	
(1,3)	000-1
(1,5)	00-01
(1,9)	0-001
(2,3)	0001-
(2,10)	0-010
(2,18)	-0010
Group 12	
(3,11)	0-011
(3,19)	-0011
(5,21)	-0101
(9,11)	010-1
(9,25)	-1001
(10,11)	0101-
(10,26)	-1010
(18,19)	1001-
(18,26)	1-010
(20,21)	1010-
Group 13	
(11,27)	-1011
(19,23)	10-11
(19,27)	1-011
(21,23)	101-1
(25,27)	110-1
(26,27)	1101-

Since even in these pairs, we have common pairs with difference 1, we can again group them.

We can actually again group them due to the presence of terms with difference 1.

Problem 4 continued on next page. . .

Table 3: Step: Merging

Group 111	
(1,3,9,11)	0-0-1
(2,3,10,11)	0-01-
(2,3,18,19)	-001-
(2,10,18,26)	–010
Group 112	
(3,11,19,27)	–011
(9,11,25,27)	-10-1
(10,11,26,27)	-101-
(18,19,26,27)	1-01-

Table 4: Step: Merging

Group 1111	
(2,3,10,11,18,19,26,27)	–01-

We can use these tables to create a prime implicant chart (note that 0s are just placeholders in the below tables).

Minterms	1	2	3	5	9	10	11	18	19	20	21	23	25	26	27	A, B, C, D, E
1,5	X	0	0	X	0	0	0	0	0	0	0	0	0	0	0	00-01
5,21	0	0	0	X	0	0	0	0	0	0	X	0	0	0	0	-0101
20,21	0	0	0	0	0	0	0	0	0	X	X	0	0	0	0	1010-
19,23	0	0	0	0	0	0	0	0	X	0	0	X	0	0	0	10-11
21,23	0	0	0	0	0	0	0	0	0	0	X	X	0	0	0	101-1
1,3,9,11	X	0	X	0	X	0	X	0	0	0	0	0	0	0	0	0-0-1
9,11,25,27	0	0	0	0	X	0	X	0	0	0	0	0	X	0	X	-10-1
2,3,10,11,18,19,26,27	0	X	X	0	0	X	X	X	X	0	0	0	0	X	X	–01-

Now, we extract the A,B,C,D,E for columns with just 1 'X' and remove the column and row.
essential prime implicants : –01-,1010-,-10-1
We reduce the chart and find more implicants again:

Problem 4 continued on next page. . .

Minterms	1	5	23	A, B, C, D, E
1,5	X	X	0	00-01
5,21	0	X	0	-0101
19,23	0	0	X	10-11
21,23	0	0	X	101-1
1,3,9,11	X	0	0	0-0-1

Similarly, we extract: 00-01 and reduce further.

Minterms	23	A, B, C, D, E
5,21	0	-0101
19,23	X	10-11
21,23	X	101-1
1,3,9,11	0	0-0-1

Here, we extract 10-11 and so, we have –01-,1010-,-10-1,00-01,10-11
Then, the representation becomes: c’d + ab’cd’ + bc’e + a’b’d’e + ab’de

Problem 5

Question: Can a decoder be used to implement any Boolean function? Justify. Design a full subtractor circuit in two ways: using (a) logic gates, and (b) decoder. Compare it with the full adder circuit.

Solution: A decoder can theoretically be used to implement most Boolean functions. The output lines of the decoder can provide the minterms of input variables. Since Boolean functions can be expressed as a sum of minterms, a decoder can be used to form a circuit of any Boolean function. For example, if we need to implement the logic of a full adder, we need a 3:8 decoder and OR gates. The input to the full adder, first and second bits and carry bit, are used as input to the decoder and by using external OR gates to form the logical sums of the minterms, we can form a circuit that implements most Boolean functions. We can select specific lines based on how we want the input-output relation. But it probably won't be able to support all boolean functions. A factor among the limitations is that it has limited inputs and if we have more variables than the supported inputs, we will have to nest multiple decoders and manage the input-output lines to generate the desired outputs. But this 'arrangement' may not generalize well for all functions. It works the same way for output lines.

The truth table for building a subtractor is:

Table 5: Truth Table for subtractor

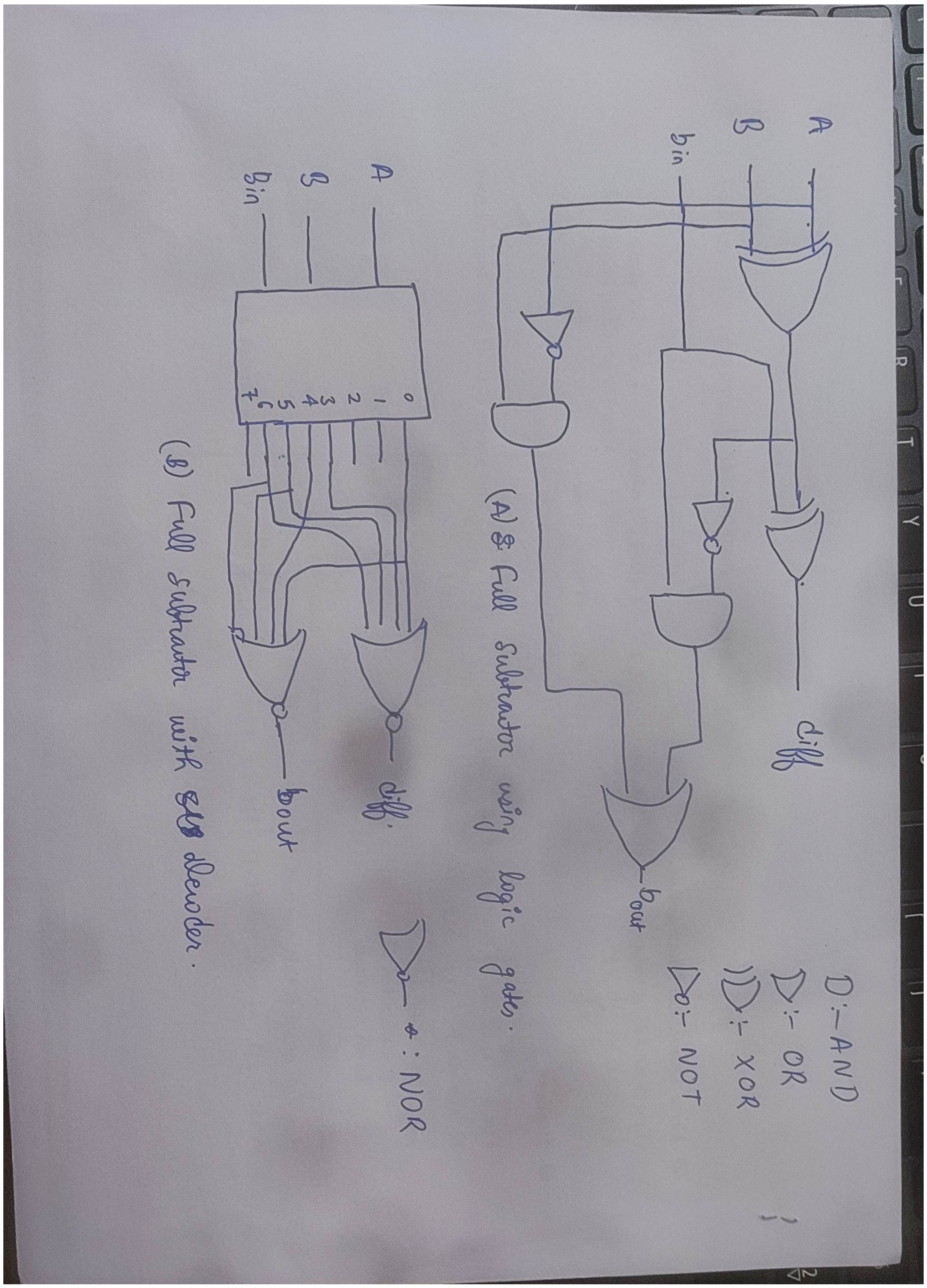
A	**B**	**Bin**	**Difference (D)**	**Borrow Out (Bout)**
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

The scanned image containing Figure (A): Full subtractor using logic gates and Figure (B): Full subtractor using decoder is attached. Note that I have consulted to online videos, webpages (like these) and forums to understand how exactly the minterms might be used to generate desired output for a subtractor, and so, Figure (B) will be similar to those shown on these websites and resources . Click here to watch the image in landscape mode.

Comparing the full subtractor with the full adder the difference lies in the input values. Rest all being same, we pass the negation of A in an AND gate to get b_{out} and the negation of the result of the first XOR operation between A and B. This is because subtraction is essentially the addition of the minuend and the two's complement of the subtrahend. The two's complement of a number is

Problem 5 continued on next page. . .

obtained by inverting all bits and adding 1 to it. So, when we negate the inputs and add them, we essentially are subtracting.


A
B
bin
diff
bout
D :- AND
D :- OR
D :- XOR
D :- NOT
(A) Full subtractor using logic gates.
A
B
Bin
0
1
2
3
4
5
6
7
diff.
bout
: NOR
(B) Full subtractor with decoder.