ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E

DIGITAL CIRCUITS LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	3
EXPERIMENT DATE	:	2.05.2023
LAB SESSION	:	FRIDAY - 10.30
GROUP NO	:	G8

GROUP MEMBERS:

150200919 : Abdullah Jafar Mansour Shamout

150220762 : Muhammed Yusuf Mermer

SPRING 2023

Contents

1 INTRODUCTION [10 points]

In this experiment we implemented a logic circuit for arithmetic operations on signed and unsigned binary numbers using 2's complement. we implemented a Half adder in experiment 1 then transformed it into a full adder in experiment 2 using Cin. In experiment 3 we used a 4-bit full adder IC to create an adder subtracter circuit for 4 bit values.

2 MATERIALS AND METHODS [40 points]

2.1 Preliminary

2.1.1 Signed and Unsigned Addition For Binary Numbers

For adding two binary numbers we start from LSB (Least significant bit). Using logic, 0 and 0 results in a 0, 1 and 0 results in 1, 1 and 1 results in a carry to the bit more significant that it by 1. After we continue to the next left bit, after we make the addition we add the carry value to the result. For two n-bit unsigned integers added there can be a resultant carry needing a (n+1)th bit. This results in no problems if we use n+1 bits to represent the number, or else we cant represent it. For n-bit signed integers, its the same as unsigned integers, however, their MSB represents the sign, and bits greater than the nth bit (n+1) are ignored. If a carry occurs there could be an overflow.it happens when we add two positive numbers and get a negative number, or when we add two negative numbers and get a positive one.

2.1.2 Signed and Unsigned Subtraction For Binary Numbers

In binary we do not use subtraction directly, we change the second operand to its negative value and add it to the first one, which is the same as subtraction. For that we use 2's complement: Flip all 1's and 0's to their counterpart, then add 1 to that value, this way we achieved that number's 2's complement. Then we add both numbers. For unsigned integer subtraction we can get either a carry or a borrow, if we get a carry then we just ignore it and the result is valid, if we get a borrow "no carry" then the result is invalid and the number can not be represented. For signed integer subtraction however,if there is a carry we just ignore it, but we need to check for the sign bits as an overflow can also occur here. if we have a negative - positive and get positive or if we have positive - negative and get negative then these are cases of overflow and we cannot represent them.

2.1.3 carry, borrow, and overflow summary

Carry: It can occur in the addition of unsiqned numbers. It indicates that the result cannot be represented with n bits, and an (n+l)st bit is necessary. Borrow: It can occur in the subtraction of unsigned numbers. It indicates that the first number is smaller than the second one (the number being subtracted). The result cannot be represented with unsigned numbers. If the result of the subtraction using 2's complement is a (n+l)-bit number,then there is no borrow, and the result is valid. Overflow: It can occur in the addition and subtraction operations only on signed numbers. It indicates that the result cannot be represented with n bits. Overflow can be detected by checking the signs of operands and the result. There is an overflow in the following cases:

pos + pos == neg

neg + neg == pos

pos - neg == neg

neg - pos == pos

2.2 Experiment 1

for this part we needed to implement a half adder. we were given the circuit design below to implement.

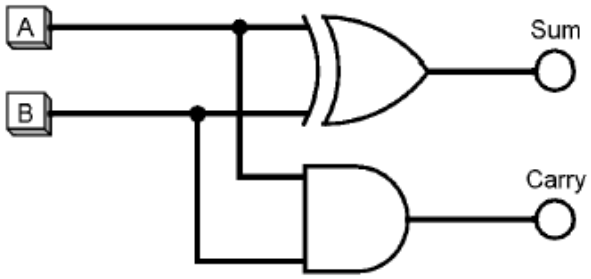

Figure 1: half adder design

as it can be seen the sum is calculated as an XOR operation between A and B while the carry is calculated as an AND operation. In the experiment we connected the input bits to switches were we controlled whether they are a 1 or 0. and for the outputs we used LEDs

2.3 Experiment 2

for this part we needed to create a full bit adder by adding a Cin input. To create this, we need to add three bits, thus by adding both A and B bits utilizing the half adder

from part 1. We take that sum and have it as input for another half adder with Cin as inputs. The final sum is the sum of the whole full bit adder, the carry is the OR of both of the half adder carrys, as if any of them create a carry, that means there is a total carry. the design that we implemented can be seen in the results section with its circuit design

2.4 Experiment 3

At first, we calculated all the required binary summations' and subtraction operations by our hands. When we are filling out the tables, we used arithmetic addition and subtraction.

In the first part, we simply make addition operation in the base 2. To do this operation, if the digit exceed 1, we subtracted 2 for current digit and give carry as 1 to next digit. In some results carry are 1 in some of them carry is 0. To calculate real value it is important to also use carry but due to limited storage of digits, we only store 4 fixed digits. Decimal values written according to these 4 bits.

In the second part of this question, we considered MSB as sign of the number. If it 1, it is negative and if 0, then it is positive. However, in here if we sum positive + positive or negative + negative, sometimes result becomes in the reverse sign, which impossible by summing to same sign numbers. For positive + positive this occurs because carry bit occurred in the previous digit makes MSB 1, but if it becomes 1, this means MSB makes number negative, which cannot be true from summing two positive numbers. This situation is called overflow. Same overflow situation can be happened between two negative numbers. When we sum two negative numbers and sum in the MSB becomes 0 and gives carry, because we don't care carry of the MSB and number seen in the MSB is 0, result interpreted as positive which cannot be true from the sum of two negative numbers.

In third part to find unsigned A - B, we simply take 2's complement of B and sum it with A. To find 2's complement, take 1's complement and add 1 to it. However, there is difference in the interpretation of the result. Due to it is unsigned, result cannot show negative numbers, means you cannot subtract in the form $A < B$. If you do so, carry bit will be 0, which means barrow and result cannot be represented as unsigned 4-bits. If equation is in the form $A \geq B$, then one can make subtraction without any hesitation. Carry bit will be 1, which means there is no barrow.

In the forth part, as in the previous part, we first made A - B via summation of A and 2's complement of B. In here carry bit has no importance. In contrast, in the signed notation, we need to consider also overflow situation like in the part 2. Overflow happens when positive - negative = negative or negative - positive = positive. Reasons are same for the overflow with part 2. For positive - negative (same as positive + positive) this

occurs because carry bit occurred in the previous digit makes MSB 1, but if it becomes 1, this means MSB makes number negative, which cannot be true from subtraction positive - negative. Another overflow situation can be happened for negative - positive (or summation of two negative numbers). After the 2' complement of B operation, when we sum two negative numbers and sum in the MSB becomes 0 and gives carry, because we don't care carry of the MSB and number seen in the MSB is 0, result interpreted as positive which cannot be true from the subtraction of negative - positive.

Click here to see tables created for this part.

3 RESULTS [15 points]

3.1 Experiment 1

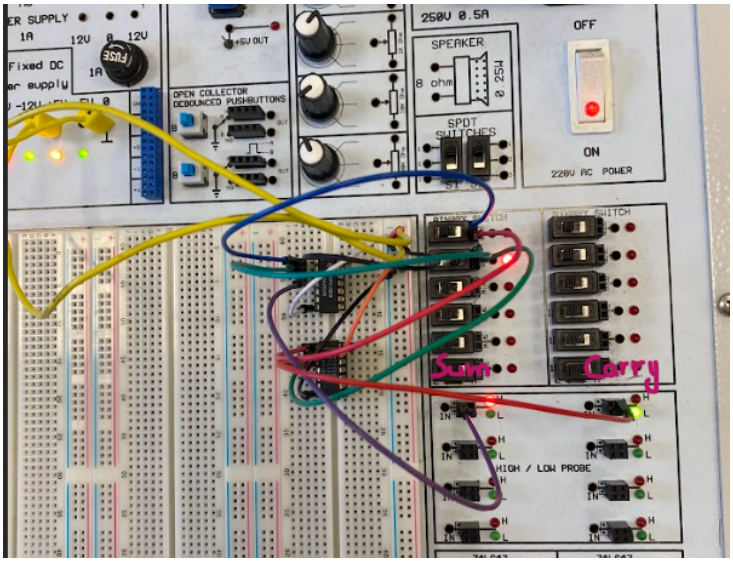

Figure 2: implementation of 0 + 1

3.2 Experiment 2

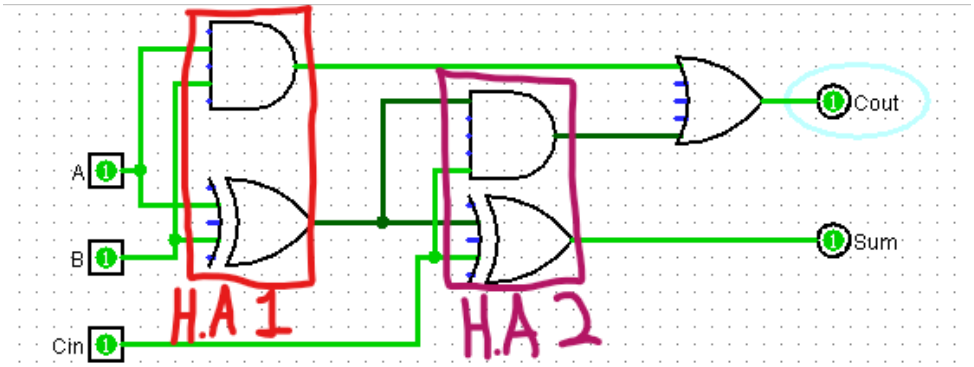

Figure 3: circuit design of a full adder using half adders

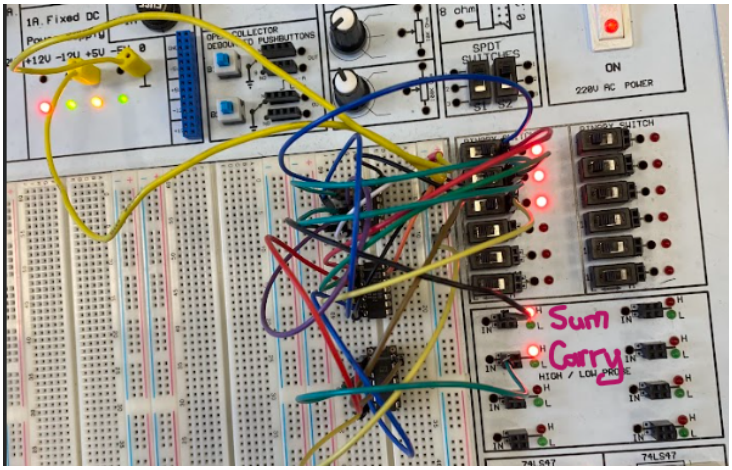

Figure 4: implementation of our full adder

a	b	Cin	sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 1: Truth Table for a+b+Cin using a full adder

3.3 Experiment 3

A	B	Carry	Result in Binary	Result in Decimal
0101	0111	0	1100	12
1101	1001	1	0110	6
1111	1111	1	1110	14
0110	1101	1	0011	3

Table 2: A+B unsigned

A	B	Overflow	Result Sign	Binary	Decimal
0101	0111	yes	-	1100	-4 / avoid
1101	1001	yes	+	0110	6 / avoid
1111	1111	no	-	1110	-2
0110	1101	no	+	0011	3

Table 3: A+B signed

A	B	Barrow	Binary	Decimal
0101	0111	yes	1110	avoid
1101	1001	no	0100	4
1111	1111	no	0000	0
0110	1101	yes	1001	avoid

Table 4: A-B unsigned

A	B	Overflow	Sign	Binary	Decimal
0101	0111	no	-	1110	-2
1101	1001	no	+	0100	4
1111	1111	no	+	0000	0
0110	1101	yes	-	1001	-7 / avoid

Table 5: A-B signed

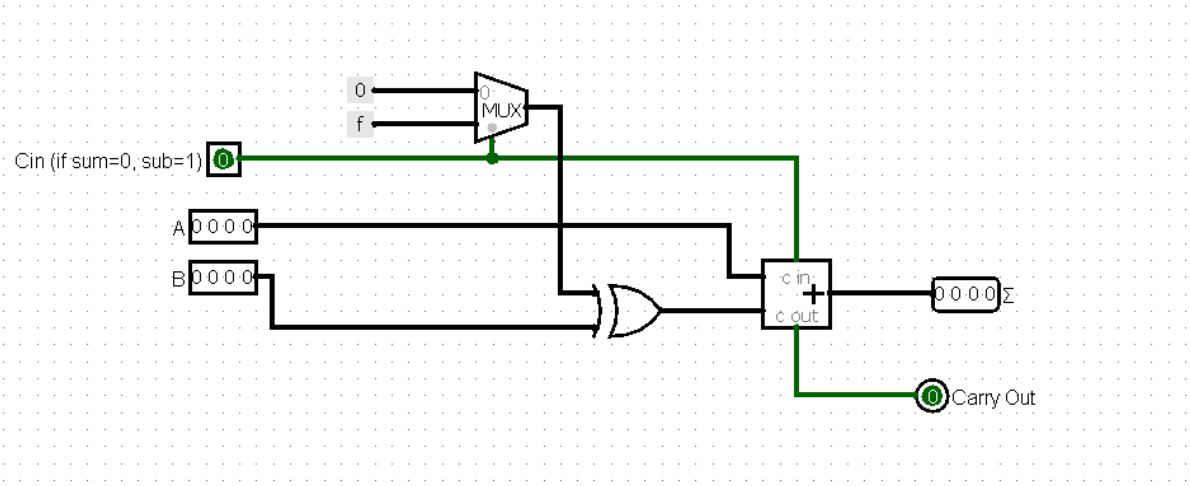

Figure 5: circuit design of 4 bit-adder-subtractor

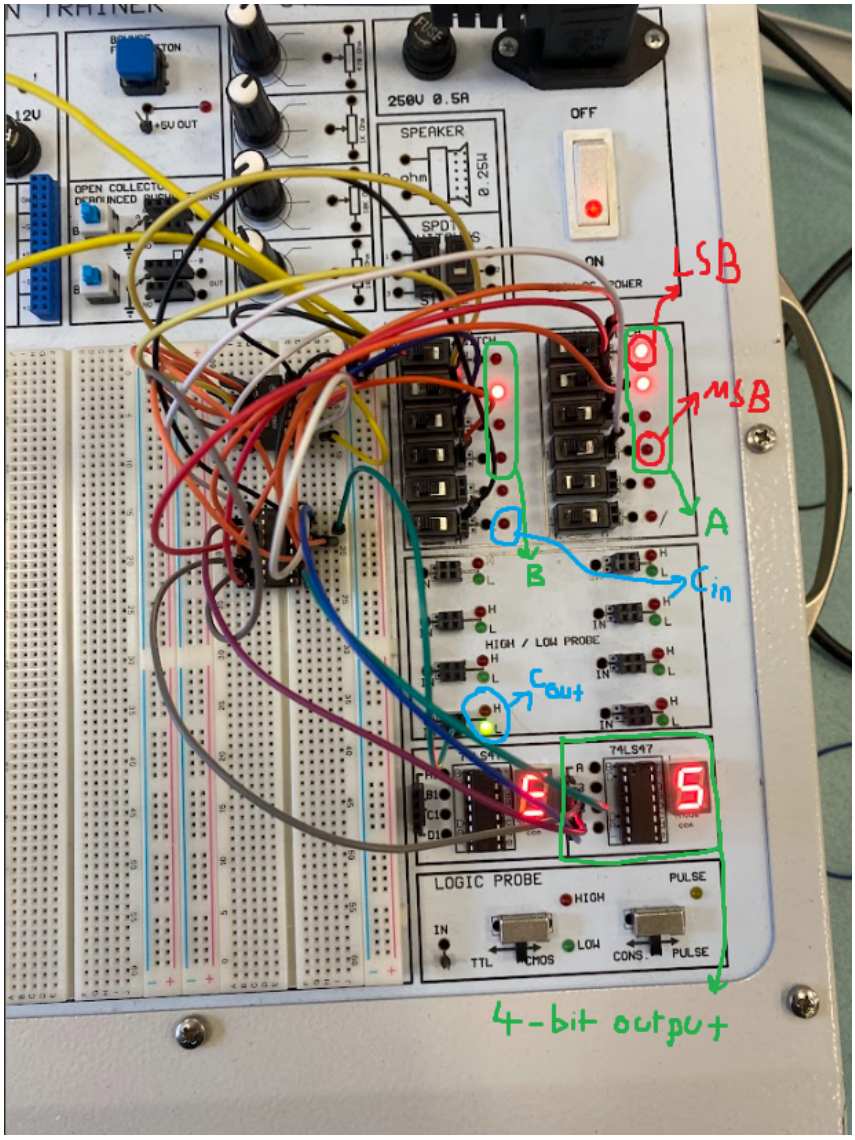

Figure 6: implementation of 3+2

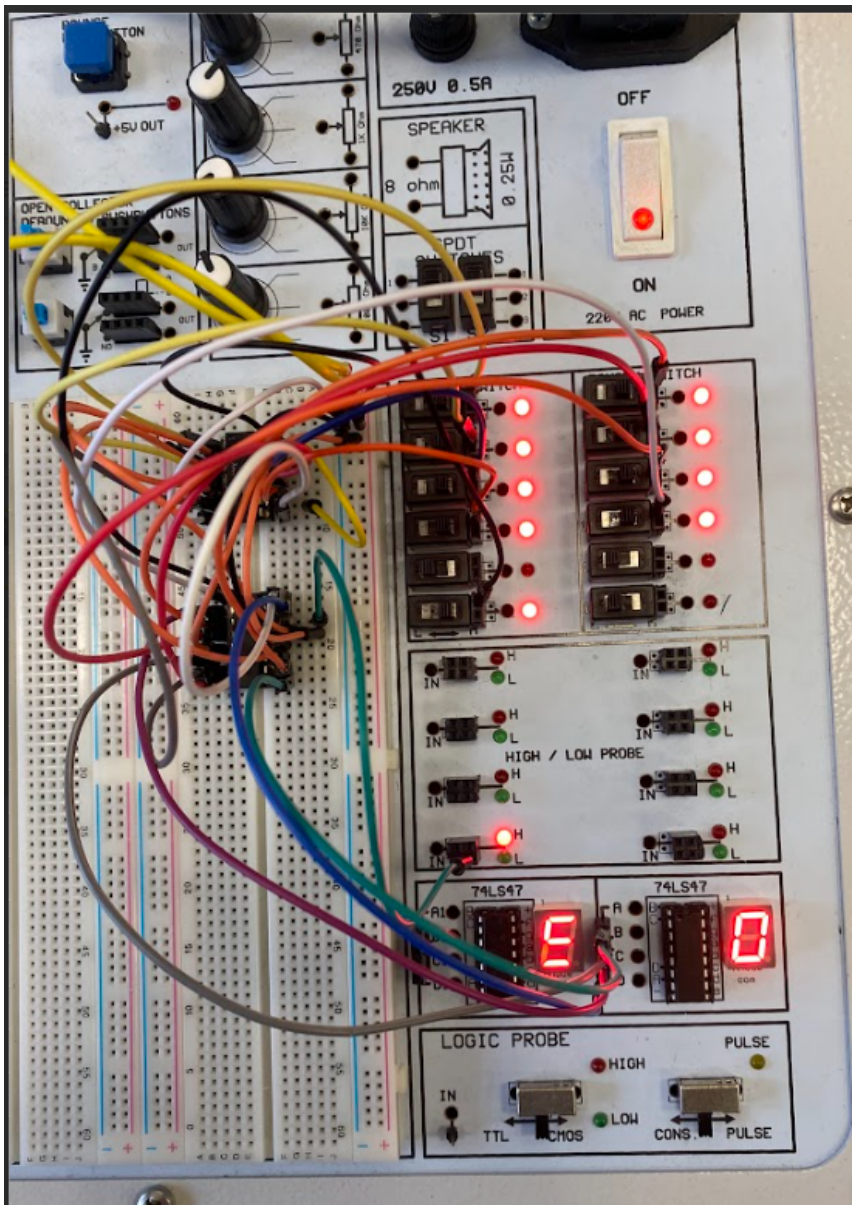

Figure 7: implementation of 15-15 or -7-(-7)

⋆ You can see all the photos from the experiment from here.

4 DISCUSSION [25 points]

4.1 Experiment part 1

for this experiment after we implemented our design as it is evident in the results section, we created a truth table to control whether our results were correct.

a	b	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 6: Truth Table for half adder

we tested all possibilities in class and found out that our implementation is correct as a sum would only occur when one bit is 1 and a carry would occur when both are, thus they use AND and XOR gates respectively

4.2 Experiment part 2

for this experiment, we utilized the half adder in our experiment 1, we also created another similar combination from the given ICs. Thus we used to half bit adders, they were used to find the total final sum. For the total final Carry, we used an OR gate to test for Carry at both half adders. As it can be seen by the table we calculated in our results section, we applied our circuit design and tested it for all the values. We found out that our design is correct and functions properly.

4.3 Experiment part 3

Before the experiment, we calculated all the necessary things to implement this part. Despite to our preparation, we little struggled in this part because there are too many connections between components, at first we couldn't be able find which part is MSB and which part is LSB. After doing some test we found it out, without any further problems. We interpreted result according to unsigned and signed representations. Besides, overflow and barrow were minded while interpreting the results. Explained well to the assistant how our system work, what are the possible errors etc. At the end we can say that we satisfied our expectations from this experiment.

5 CONCLUSION [10 points]

With this laboratory session, we learned to implement HA(Half Adder).Using two of the HA implemented FA(Full Adder) which contains additional carry in bit. In the last part, we used a component for 4 bit addition. We connected XOR gates to B inputs of 4-bit Full Adder. As one of the inputs we sent bits of B and in the other we send carry-in value. This is required because for the subtraction by using nature of XOR gate which

is inverting a input bit when the other input is 1. Not only that but also adding +1 as carry in helped us to achieve 2's complement form.

In the all implementations, we achieved our expected results, and can be say that finished without any further problems.