ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E

DIGITAL CIRCUITS LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	3
EXPERIMENT DATE	:	06.03.2020
LAB SESSION	:	FRIDAY - 08.30
GROUP NO	:	G6 - G12

GROUP MEMBERS:

150170039 : Fatih MURAT

150170044 : Bengisu ÖZBEK

SPRING 2020

Contents

1 INTRODUCTION

In this experiment, we were expected to implement and apply some boolean algebra operations such as addition and subtraction for signed and unsigned binary numbers using half adder and full adder; exclusive disjunction, multiplication, increment and discrement operations by using Arithmetic Logic Unit(ALU) and D Flip-Flop.

2 MATERIALS AND EXPERIMENT

Tools Used

- C.A.D.E.T
- 74000 series ICs
  - 74xx08 - Quadruple 2-input Positive AND Gates
  - 74xx32 - Quadruple 2-input Positive OR Gates
  - 74xx83 - 4-bit Binary Full Adder
  - 74xx86 - Quadruple 2-input Positive Exclusive Or (XOR) Gates
  - 74xx174 - Hex D-Type Flip-Flops
  - 74xx181 - 4-Bit Arithmetic Logic Unit

2.1 PART 1

In the first part of the experiment, we were expected to implement half adder given below. As it has been known, half adder takes 2 input and gives 2 outputs. One output is sum of the 2 inputs, and the other one is the carry output.

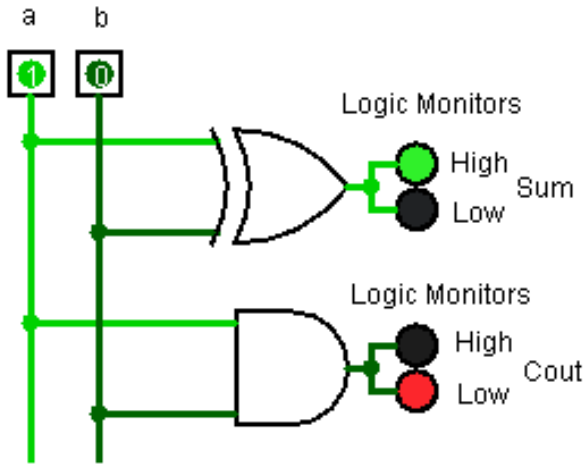

Figure 1: Half Adder Circuit Design

Truth table of the half adder that we implemented above is as below.

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 1: Truth table of half adder.

2.2 PART 2

In the second part of the experiment, we were expected to implement full adder. Difference between full adder and half adder is that full adder takes extra 1 input. While we can not use half adders with more than one bit operations, with that extra input, we can use full adders with n(n≥1) bit operations. The full adder implementation that we did as below.

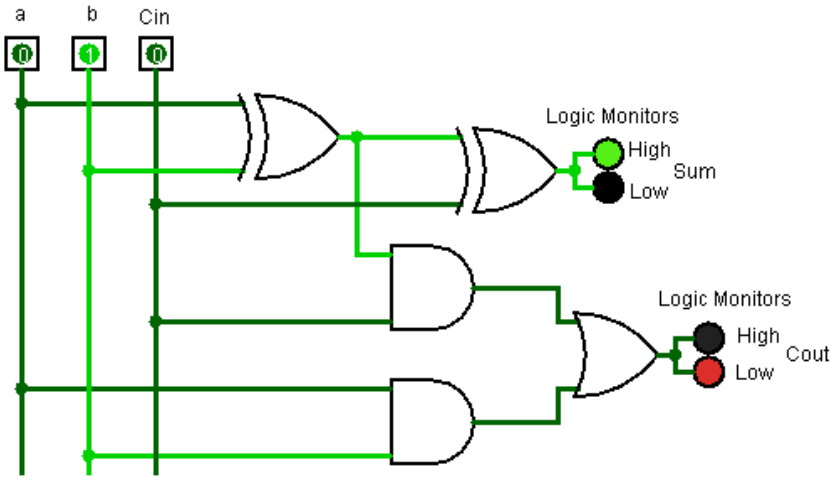

Figure 2: Full Adder Circuit Design

The truth table of the full adder is shown in Table 2 as below.

A	B	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table 2: Truth table of full adder.

2.3 PART 3

In the third part of the experiment, we were expected to implement 4-bit full adder using full adder and XOR gates to deliver addition and subtraction operations of 4-bit signed and unsigned binary numbers. The aim of using XOR gates is to determine which operation(addition/subtraction) we are going to use. When we give 0 to C_{in}, our circuit will add two 4-bit binary numbers together. Since 0 $\oplus$ x = x, there will not be any complementary operation and everything is going to done properly by the circuit. When we give 1 to C_{in}, our circuit will subtract B from A. In order to do that, we use 2's complementary for B(2's complement of B = B' + 1). Since 1 $\oplus$ x = x', B0, B1, B2 and B3 inputs will be get complemented one by one, and C_{in} value will go to the full adder as carry input again. As a result, our circuit will operate addition and subtraction operations for 4-bit signed and unsigned binary numbers successfully.

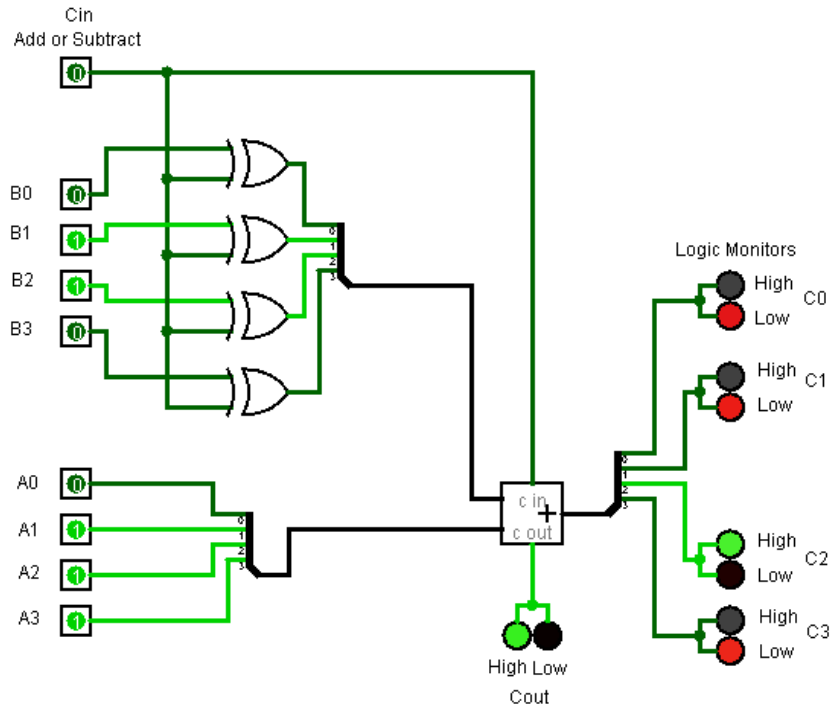

Figure 3: 4-Bit Full Adder Circuit Design

When we interpret A+B accepting them as unsigned binary numbers, we need to consider **carry** bit((n+1)th bit). If we get fifth bit as a result of addition, it means that we have carry bit and the result can not be represented. In other words, our numbers has to be interval of [0,15] in decimal, and any other number different from that interval can not be represented.

In Table 3, first case has not carry and it is representable. However other three cases have carry so they can not be represented.

A	B	Carry	Result in Binary	Result in Decimal
0101	0111	0	1100	12
1101	1001	1	0110	22
1111	1111	1	1110	30
0110	1101	1	0011	19

Table 3: Results of the unsigned sum $A + B$.

When we interpret A+B accepting them as signed binary numbers, we need to consider sign of the result and check if overflow exists or not. All possibilities that we need to consider for overflow situations;

- positive + positive -> negative
- negative + negative -> positive

In the Table 4, while third and fourth cases have not overflow and can be represented, first and second cases have overflow and can not be represented.

A	B	Overflow	Result sign	Result in Binary	Result in Decimal
0101	0111	1	-	1100	-4
1101	1001	1	+	0110	6
1111	1111	0	-	1110	-2
0110	1101	0	+	0011	3

Table 4: Results of the signed sum $A + B$.

When we interpret A-B accepting them as unsigned binary numbers, we need to consider borrow situation. While we subtracting two unsigned binary numbers, we have to obtain carry((n+1)th) bit. So, carry means no borrow and it is presentable while no carry means borrow and the result can not be represented.

In the Table 5, first and fourth cases have borrow, so they can not be represented while second and third cases can.

A	B	Borrow	Result in Binary	Result in Decimal
0101	0111	1	1110	14
1101	1001	0	0100	4
1111	1111	0	0000	0
0110	1101	1	1001	9

Table 5: Results of the unsigned subtraction $A - B$

When we interpret A-B accepting them as signed binary numbers, we need to check sign of the result and check if overflow exists or not as we did it for signed A+B. All possible cases for overflow;

- positive - negative -> negative
- negative - positive -> positive

In the Table 6, sign of the result of the fourth case must be positive since we subtract negative number from positive number. But we obtained negative sign, so there is overflow. Other three subtraction operation is what it has to be and there is no overflow.

A	B	Overflow	Result sign	Result in Binary	Result in Decimal
0101	0111	0	-	1110	-2
1101	1001	0	+	0100	4
1111	1111	0	+	0000	0
0110	1101	1	-	1001	-7

Table 6: Results of the signed subtraction $A - B$

2.4 PART 4

In the last part of the experiment, firstly we used Hex Inverter to use the inverse of B because of ALU outputs are in negative logic. We used 4-Bit Arithmetic Logic Unit(74xx181), a D type flipflop(74xx174), and a Hex Inverter for reaching the goal of this experiment. The ouputs are observed on the logic monitor. Input values are taken in the logic switches such as Bs and Ss. We used D type flipflop and we took the A inputs from these flipflops. Clock and M are taken from the SPDT switches. The Cn input of the ALU is taken from the power supply and is connected to ground.

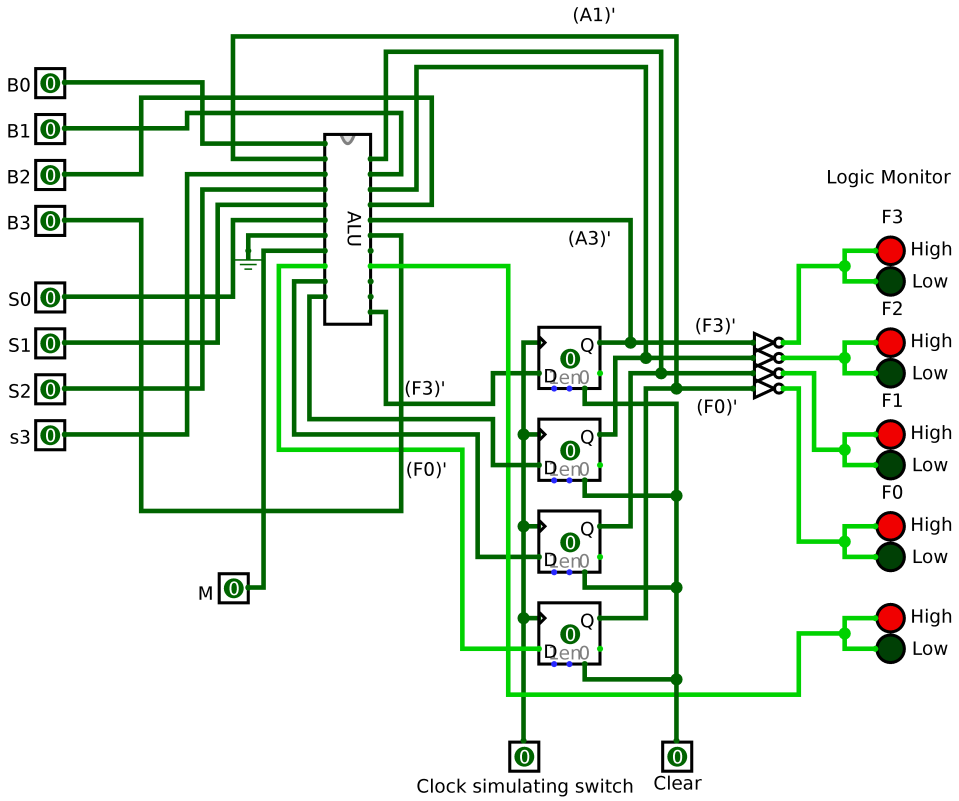

Figure 4: 4 bit full adder (in the References)

3 RESULTS

Results that we obtained from the first three part of the experiment were exact same with the expected results and we verified them with the related truth tables. Therefore, there was no contradiction with the results we got. However, in the last and fourth part of the experiment, the results we obtained from our circuit design that we used ALU and D flip-flop were not matching. Results we obtained during the experiment given in the related parts in detail.

4 DISCUSSION

In the first part, we simply implemented half adder using an XOR and an AND gate. In the second part, we developed full adder using half adder that we did in the first part. Only difference between them is that full adder takes carry input beside two input while half adder does not. And this property allows full adder to be used in the n-bit operations while half adder can be used only for one bit operations. With this information, we implemented 4-bit full adder in the third part of the experiment. Up to now, we verified all of our output results with truth related truth tables. In the last part, we were not be able to analyze and verify our implementation because of the limited time of the session.

5 CONCLUSION

As a result of the session, first three part were quite easy to handle since we had basic knowledge about the topics. As we mentioned in the results part, in the fourth part of the experiment, we implemented our design but we could not obtained expected outputs. Because of restricted time of the lab session, we could not analyze where we did something wrong and could not figure out the problem. But if we had time, we would have analyzed our implementation from starting to end carefully and could have been solve the problem. This experience taught us that we need to more and more take care of what we are going to face during the sessions.

REFERENCES

[1] Istanbul Technical University Department of Computer Engineering. Blg 242e logic circuits laboratory experiments booklet version 1.9.1, Spring 2020.

[2] Overleaf documentation https://tr.overleaf.com/learn.

[3] Part 4 logisim design is used from `https://github.com/fatihaltinpinar/logic-lab/blob/c4521c5ba1f486bbbecc662a0d56af2f90893276/exp4/Report_LaTeX/part4.png`.