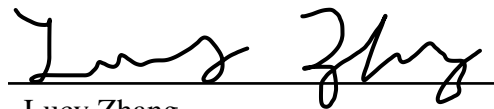


# **CMPE-160 Digital System Design 1**

## **Laboratory Exercise 6**

### **Binary Addition and Subtraction Circuits**

By submitting this report, I attest that its contents are wholly my individual writing about this exercise and that they reflect the submitted code. I further acknowledge that permitted collaboration for this exercise consists only of discussions of concepts with course staff and fellow students. Other than code provided by the instructor for this exercise, all code was developed by me.

A handwritten signature in black ink, appearing to read 'Lucy Zhang', is written over a horizontal line.

Lucy Zhang

Performed 27 September 2022

Submitted 4 October 2022

Lab Section: 1

Instructor: Purab Sutradhar

TA: Morgan Kreifels  
Karisha Pradham  
Colin Vo

Lecture Section: 1

Lecture Instructor: Richard Cliver

## Abstract

The purpose of this exercise was to study binary arithmetic circuit through the design and implementation of a 1-bit full adder and a 4-bit adder/subtractor. Using K-maps and Boolean algebra, the output expression of a 1-bit full adder was determined and a circuit was designed using logic gates. Using a 74LS283 IC chip, a 4-bit adder/subtractor was designed with combinational logic gates and constructed in hardware. A simulated waveform was generated to verify the circuit. Expected values from the 4-bit adder/subtractor were compared with the observed results. Discrepancies were seen between the expected and observed results due to overflow errors. This was expected to happen, so the experiment was a success.

## Design Methodology

A Karnaugh Map was used to show the expected outputs of a full adder. The inputs of a full adder are  $A$ ,  $B$ , and  $C_{in}$ . The input  $C_{in}$  represents the carry in and  $A$  and  $B$  represent the bits being added. The outputs of a full adder are  $Sum$  and  $C_{out}$ . The output  $C_{out}$  represents carry out, the bit that is carried out. The output  $Sum$  represents the sum of the bits.

A Karnaugh map was made for  $Sum$  for a 1-bit adder in Table 1.

Table 1: *Karnaugh Map of 1-bit Full Adder Sum*

$\begin{matrix} AB \\ C_{in} \end{matrix}$	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Table 1 shows the Karnaugh map for  $Sum$  in a 1-bit adder. The K-map does not simplify any further, so Boolean algebra was used to find the final expression.

Equation 1 was derived using Boolean algebra with the following steps.

$$\begin{aligned} &A'B'C + A'BC' + ABC + AB'C' \\ &A'(B'C + BC') + A(BC + B'C') \\ &A'(B \oplus C) + A(B \oplus C)' \end{aligned}$$

$$Sum = A \oplus (B \oplus C) \quad (1)$$

Equation 1 was simplified using distributive law and knowledge of XOR logic. In addition to the *Sum* expression, another expression for the *Cout* was derived.

Table 2 shows the K-map used for the *Cout* of the 1-bit adder.

Table 2: *Karnaugh Map of 1-bit Full Adder Carry Out*

$\begin{matrix} AB \\ C_{in} \end{matrix}$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Table 2 shows the groupings made on the K-map for *Cout*. Each grouping was used to simplify the expression.

The resulting expression is shown in Equation 2.

$$C_{out} = AB + BC + AC \quad (2)$$

Equation 2 shows the expression for *Cout*. Using (1) and (2), the 1-bit adder was designed.

The circuit schematic for the 1-bit adder is shown in Figure 1.

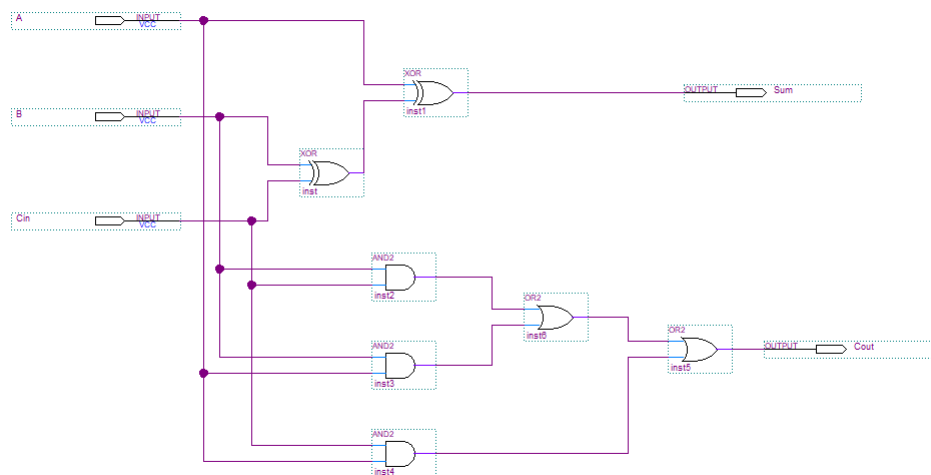


Figure 1: *Circuit Schematic for 1-bit Adder*

Figure 1 shows a circuit that has three inputs ( $A$ ,  $B$ , and  $Cin$ ) and two outputs ( $Sum$  and  $Cout$ ). Using a 74LS283 chip, a 4-bit adder/subtractor was designed in a similar fashion.

Figure 2 displays the circuit design for the 4-bit adder/subtractor.

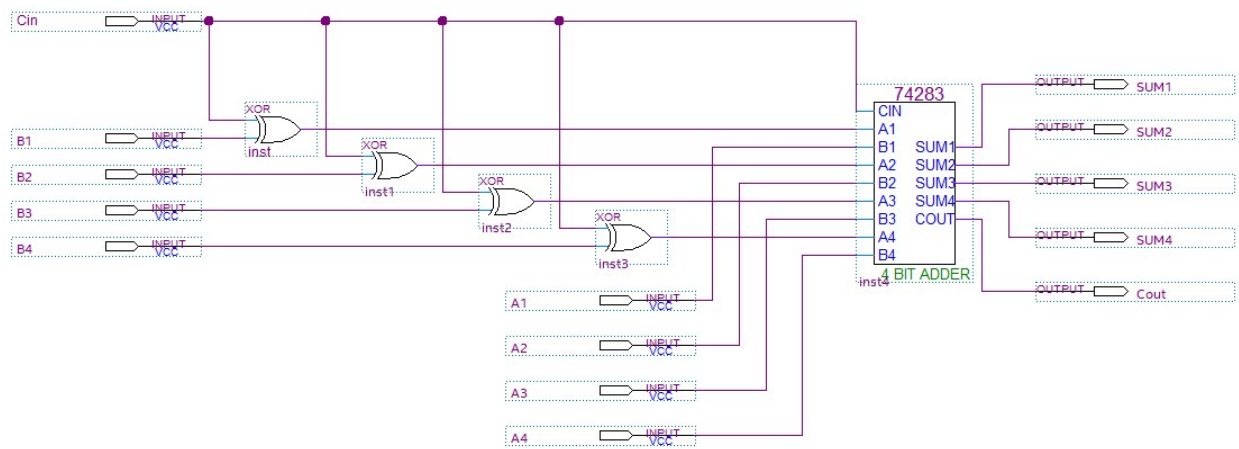


Figure 2: Circuit Schematic for 4-bit Adder/Subtractor

Figure 2 has nine inputs and five outputs. The inputs are made of  $Cin$ ,  $A1$ ,  $A2$ ,  $A3$ ,  $A4$ ,  $B1$ ,  $B2$ ,  $B3$ ,  $B4$ . The input  $Cin$  in this circuit is used as a control. Whenever,  $Cin$  is at logic 0, the circuit performs addition and outputs the sum of  $A$  and  $B$ . Whenever,  $Cin$  is at logic 1, the circuit performs subtraction and outputs the difference of  $A$  and  $B$ . The inputs beginning with "A" represent one 4-bit number with each A input equating to a bit. The inputs starting with "B" work in a similar fashion. The outputs are  $S1$ ,  $S2$ ,  $S3$ ,  $S4$ , and  $Cout$ . The outputs starting with "S" collectively represent the 4-bit sum of  $A$  and  $B$ . The output  $Cout$  represents any carried over bits.

## Results and Analysis

To verify the circuit shown in Figure 1, the circuit was simulated to generate a waveform.

The generated waveform is shown in Figure 3.

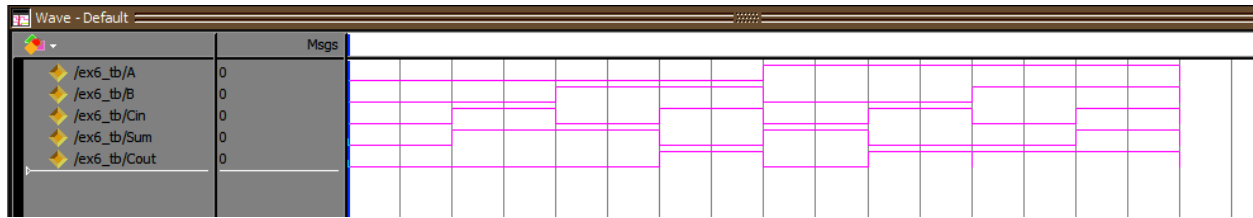


Figure 3: 1-Bit Full Adder Simulation

Figure 3 shows the results expected from the circuit in Figure 3. The waveform matches with the Karnaugh maps in Table 1 and Table 2 and verifies the circuit schematic. The circuit was then constructed in hardware and tested. The inputs were wired to switches and the outputs were connected to LED's. A four-bit adder chip (74LS283) was used.

The results of the hardware circuit when Control = 0 was recorded in Table 3.

Table 3: Results of 4-Bit Adder Addition

A3 A2 A1 A0	B3 B2 B1 B0	C4	F3 F2 F1 F0	$A + B = F$
S7 S6 S5 S4	S3 S2 S1 S0	L4	L3 L2 L1 L0	(Decimal)
1 0 1 0	0 0 1 1	0	1 1 0 1	$(-6) + (3) = 3$
1 1 1 1	1 1 1 1	1	1 1 1 0	$(-1) + (-1) = -2$
1 1 1 1	0 1 1 0	1	0 1 0 1	$(-1) + 6 = 5$
0 1 1 1	0 0 1 1	0	1 0 1 0	$(7) + (3) \neq -6$
0 1 0 1	1 0 1 0	0	1 1 1 1	$(-6) + (5) \neq -1$

Table 3 shows the results of the 4-bit adder/subtractor when it added the two 4-bit binary numbers  $A$  and  $B$ . There were two instances of overflow error in the data seen in the last two rows of the Table 1. The binary in the last row should theoretically add up to 15, but the adder read -1. The binary in the second to last row should add up to 10, but the *Sum* output read -6. These errors are due to the fact that the carry out bit (*Cout*) is not appended to the binary sum. The fifth bit found in column  $C4$  is needed to accurately represent the sum in these instances, but only the 4-bits from *Sum* is considered as the final number.

The control was switched to logic 0 and the 4-bit adder/subtractor performed subtraction. The output would be  $F = A - B$  where  $B$  is subtracted from  $A$ .

Table 4 shows the results from the subtraction.

Table 4: *Results of 4-Bit Adder Subtraction*

A3 A2 A1 A0	B3 B2 B1 B0	C4	F3 F2 F1 F0	A - B = F
S7 S6 S5 S4	S3 S2 S1 S0	L4	L3 L2 L1 L0	(Decimal)
1 0 1 0	0 0 1 1	1	0 1 1 1	$(-6) - (3) \neq 7$
1 1 1 1	1 1 1 1	1	0 0 0 0	$(-1) - (-1) = 0$
1 1 1 1	0 1 1 0	1	1 0 0 1	$(-1) - 6 = -7$
0 1 1 1	0 0 1 1	1	0 1 0 0	$(7) - (3) = 4$
0 1 0 1	1 0 1 0	0	1 0 1 1	$(-6) - (5) \neq -5$

Table 4 shows the same values for  $A$  and  $B$  as Table 3. The difference of the two values are taken instead of the sum. The column  $C4$  represent the carry out bit ( $Cout$ ). When subtracting two binary numbers in 2's complement, the second number is inverted to it's negative equivalent in binary and both numbers are added together. Once again, there were two instances of overflow error. These errors can be found in the first and last row of Table 4. For the first row, the expected output was -9, but the observed value was 7. In the last row, the expected output was 11, but the observed value was -5. These errors are due to the fact that the carry out bit ( $Cout$ ) is not appended to the binary difference. The fifth bit found in column  $C4$  is needed to accurately represent the difference in these instances, but only the 4-bits from  $Sum$  is considered as the final number. This is what causes an overflow error, when the numbers aren't accurately represented with the given number of bits.

## Conclusion

Using K-maps and Boolean algebra, expressions were derived to design a 1-bit full adder. A 4-bit adder subtracter was also designed using a similar method. The 4-bit adder/subtractor circuit was verified using a generated waveform simulation and then constructed in hardware. The observed results were as expected except for a few instances of overflow error. This exercise shows how overflow error can occur. One thing to note is that the carry out bit does not necessarily reflect whether or not there was overflow error. The carry out bit being a logic 0 or logic 1 does not show if there was overflow. The occurrence of overflow is based on whether the number represented is the expected value with the given number of bits. It is important for people using adders and subtracters to take into account the possibility of overflow error and have the appropriate number of bits prevent it. Overall, this exercise was successful.

## Questions

1. What would be the simplest way to combine two full adders to make a two-bit carry-ripple adder?

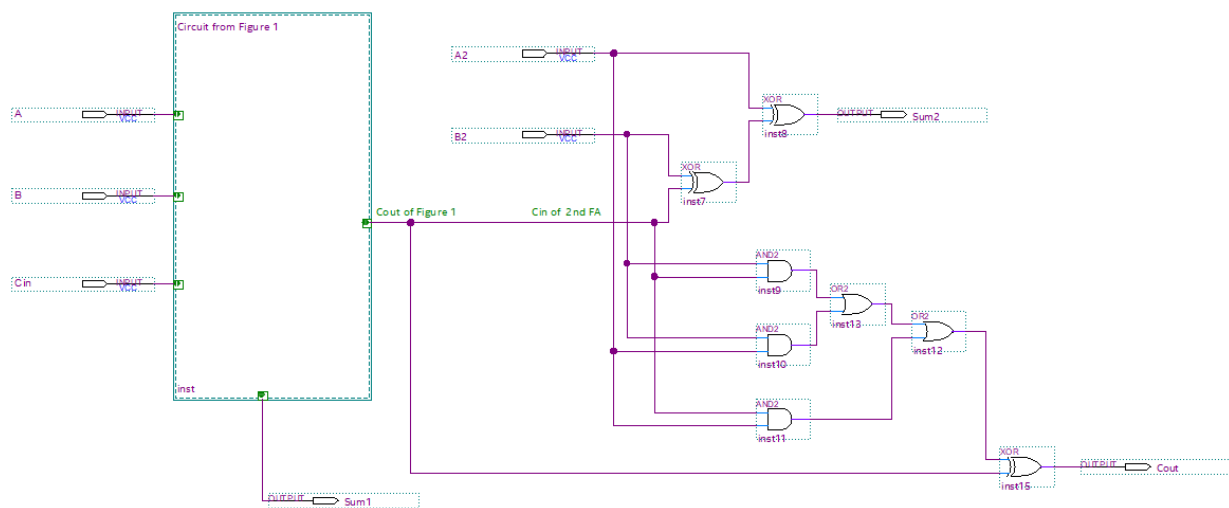


Figure 4: 2-bit Carry-Ripple Adder

2. How can overflow be detected for addition and subtraction of two's complement numbers?

$$C_n \text{out} \oplus C_{n-1} \text{out} \quad (3)$$

3. Why does the 4-bit adder/subtractor circuit produce incorrect two's complement results, even though the circuit is correct?

The 4-bit adder/subtractor in 2's complement can only show numbers between -8 and 7. If the resulting number from the operations in the 4-bit adder/subtractor was outside of the range, it resulted in incorrect two's complement results. The circuit does not account for overflow.

4. What happens to the "incorrect" results found in Question 3 if the result register is extended by one bit to use the carry-out bit as the MSB in the register? If this change were implemented, would the "correct" results found in Question 3 still be "correct" results?

If extended by one bit the "incorrect" results found in Question 3 would be correct. However, this does not extend to all the "correct" results. Some "correct" results would remain "correct" while others would be "incorrect". Due to the way that 2's complement works, the MSB (the first bit) can alter the number in certain cases. For example, the result in row 2 of Table 4 is 0000 in binary and 0 in decimal. Zero is the "correct" result. If the result was extended by one bit to include the carry out bit, the result would be 10000 in binary and -16 in decimal. Clearly, by appending the additional bit, the "correct" result is altered to be "incorrect".

5. Design an additional circuit for a 4-bit adder/subtractor so that the overall circuit will produce a 5-bit result that is always correct in the case of overflow.

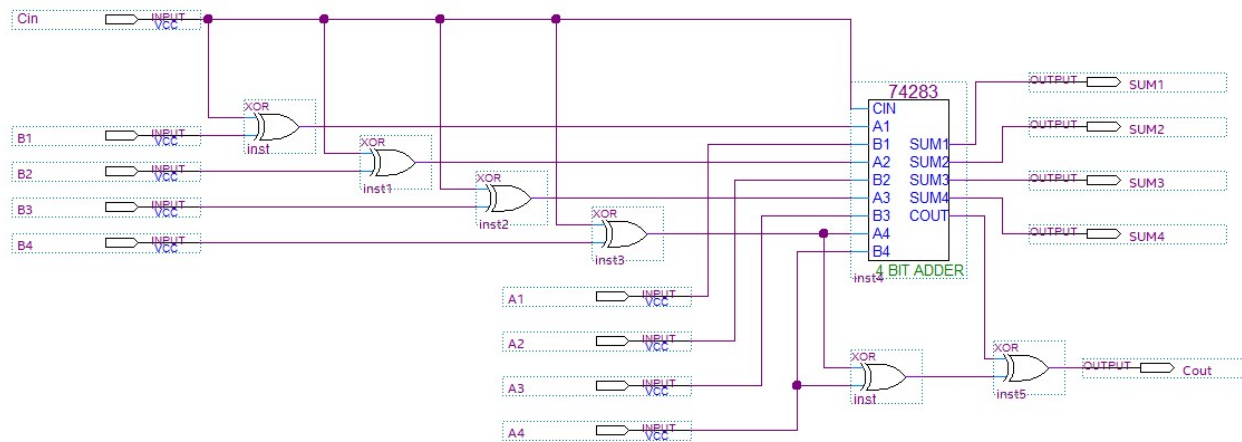


Figure 5: 4-bit Adder/Subtractor with Overflow Accounted For



### Exercise 6: Binary Addition and Subtraction Circuits

Student's Name: \_\_\_\_\_

Section: \_\_\_\_\_

Prelab		Point Value	Points Earned	Comments
Part 1	K-maps	2	2	Mick 9/27
	Boolean transformations	4	4	
	Correct Boolean expressions	2	2	
	Schematic diagram	4	4	
	Simulations	4	4	
Part 2	Schematic diagram	4	4	Mick 9/27

Demo		Point Value	Points Earned	Date
Demo	1-bit full adder	20	20	Mick 9/27
	4-bit adder/-subtractor	20	20	Mick 9/27

To receive any grading credit students must earn points for both the demonstration and the report.

### Exercise 6: Binary Addition and Subtraction Circuits

Report		Point Value	Points Earned	Comments
Abstract		4		
Design Methodology	K-maps / Boolean transformations	3		
	1-bit FA circuit diagram	2		
	Discussion of add/subtract control	3		
	4-bit adder/-subtractor circuit diagram	2		
Results and Analysis	1-bit full adder Simulation	2		
	4-bit adder/-subtractor result tables	3		
Conclusion		4		
Questions	Q1	4		
	Q2	2		
	Q3	3		
	Q4	2		
	Q5	3		
Writing Composition		3		
Total for prelab, demo, and report		100		