Laboratory Exercise #5
Simple Arithmetic Logic Unit

ECEN 248: Introduction to Digital Design

Department of Electrical and Computer Engineering
Texas A&M University

ENGINEERING
TEXAS A&M UNIVERSITY

1 Introduction

In lab this week, you will design, implement, and test a simple 4-bit Arithmetic Logic Unit (ALU) which will perform elementary computations such as addition, subtraction, and bit-wise AND. To do so, you will learn about Two's Complement arithmetic and multiplexers.

2 Background

This section will provide you with the necessary background to complete this laboratory assignment. Topics that will be discussed in-depth include 2's Complement Arithmetic, Multiplexers, and Arithmetic Logic Units.

2.1 Two's Complement Arithmetic

In the previous lab, we briefly introduced addition of unsigned numbers but made no mention of subtraction. In order to perform subtraction, we need a way to represent negative numbers in binary. Therefore, this week, we will introduce a binary signed number representation and show how to use it for subtraction. Note that a binary signed number is simply a binary number that can take on either a postive or negative value. Although there are many ways to represent signed numbers in binary, we will only touch on one such representation in this lab assignment, namely *Two's Complement*.

Table 1: 3-bit Binary Representation

Binary	Decimal (Unsigned)	Decimal (2's Complement)
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-4
101	5	-3
110	6	-2
111	7	-1

For a moment, consider a 3-bit binary number, which can represent $2^3 = 8$ numbers. Table 1 displays the unsigned decimal values as well as the signed (two's complement) decimal values for all eight binary numbers. One thing to notice is that the positive signed numbers share the same binary numbers as their unsigned counterparts. Likewise, the value '0' is the same in both the signed and unsigned case. When the most-significant-bit (msb) of the binary number is '1', the signed number is negative, which is why the msb is often referred to as the sign bit. In mathematical terms, the decimal value of a two's complement binary number, X, is described in EQ 1, such that x_i is the i^{th} binary digit and $-2^{n-1} \leq X \leq 2^{n-1} - 1$.

$$X = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i \tag{1}$$

The primary advantage of two's complement binary representation is that the unsigned adders discussed previously can be used to perform arithmetic on two's complement binary numbers as well. Take a moment to convince yourself of this using Table 1 as a guide. Note that if two numbers are added such that their result is not listed in the table, overflow has occurred! Figure 1 illustrates the addition of two's complement numbers from Table 1. The situation on the left does not cause overflow while the situation on the right does. Notice in each of these scenarios, the carry-out bit is discarded. Later on, you will see that the carry-out bit can be used to detect overflow.

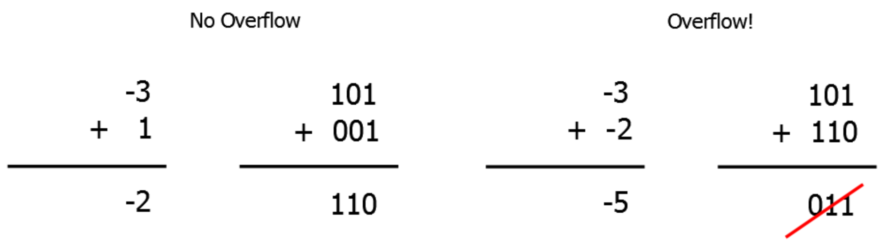
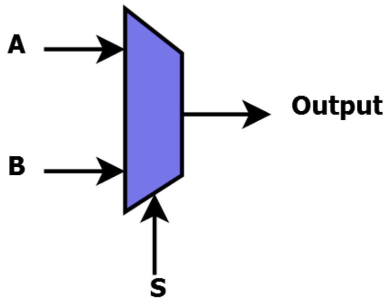


Figure 1: Addition of Two's Complement Numbers

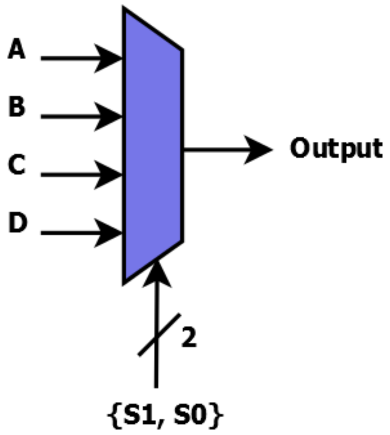
At this point, it should be obvious that two's complement subtraction can be performed by negating the subtrahend and adding it to the minuend (i.e. $A - B = A + (-B)$). To complement a two's complement number, simply invert the binary digits and add one. Convince yourself of this with the numbers in Table 1. Please note that a mathematical proof of this phenomena is beyond the scope of this laboratory course.

2.2 Multiplexers

A Multiplexer (MUX for short) is a digital circuit which selects an output from more than one input. Figure 2 provides two such examples, where A, B, C, D, $Output$, and S are all 1-bit wide. Note that $\{S1, S0\}$ represents the concatenation of $S1$ and $S0$ and is a 2-bit bus (i.e. grouping of signals noted by the diagonal line crossed through the arrow and marked with a "2").
The 2:1 MUX shown in Figure 2(a) outputs A if the selection line, S, is logic 0 and outputs B otherwise. Figure 2(b) depicts a 4:1 MUX which selects from four inputs, A, B, C, or D, based on the binary value



(a) 2:1 MUX



(b) 4:1 MUX

Figure 2: Multiplexers

of $\{S1, S0\}$. For example, when $S1 = 0$ and $S0 = 0$ (i.e. $\{S1, S0\}$="00"), the value of input A is passed through to $output$. Likewise, when $S1 = 0$ and $S0 = 1$ (i.e $\{S1, S0\}$="01"), the value of input B is pass through. The previous examples depicted 1-bit wide multiplexers; however, it is often necessary to multiplex buses of signals. Figure 3 shows how a 2-bit MUX can be constructed from two 1-bit MUXs. The drawing on the left shows the two MUXs superimposed to create the 2-bit wide MUX on the right. Note that the selection signals, S, have been tied together. We can easily create n-bit wide MUXs in this manner.

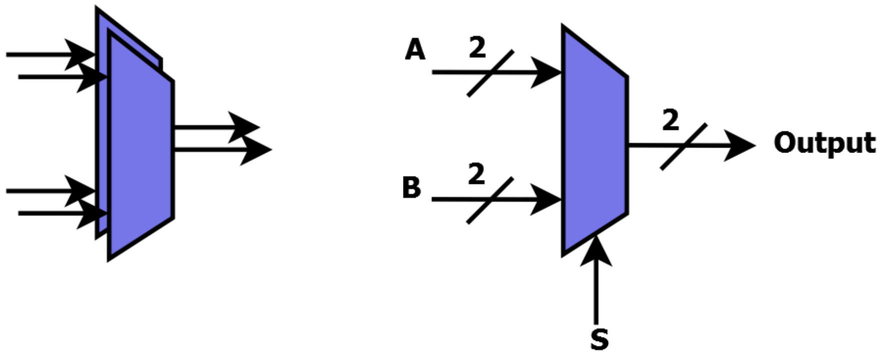


Figure 3: 2-bit Wide 2:1 MUX

2.3 Arithmetic Logic Unit

The ALU is a digital circuit capable of performing arithmetic and logical operations and is a fundamental building block of any sort of computational device. Simple microprocessors such as those found in home appliances often contain only one ALU, while the modern Central Processing Units (CPUs) and Graphics Processing Units (GPUs) found in today's desktops and mobile devices contain many very powerful ALUs. The symbol typically used to represent an ALU is shown in Figure 4.

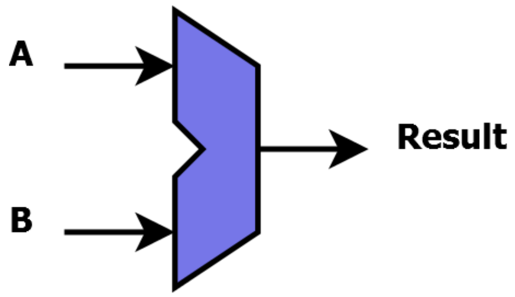


Figure 4: ALU Symbol

A and B represent the n-bit input buses, while $Result$ represents the output bus, which can be n-bits in size or greater. Not shown in this simple drawing are the control and status signals, which will be expounded upon shortly. One way to create a simple ALU is to build several computational blocks and multiplex the output buses. The selection signals for the multiplexer then serve as the control signals for the ALU. We will do something similar to this in the design section of the pre-lab.

3 Pre-lab

This section of the laboratory assignment will guide you through the design of a simple ALU capable of adding, subtracting, and bit-wise ANDing.

3.1 Adder/Subtractor Design

The ALU design discussed earlier can be inefficient because certain ALU operations could share logic. For example, as we learned in the background section of this manual, addition and subtraction both utilize an adder circuit. If we naively designed our ALU, we would create a separate adder circuit for addition and another one for subtraction. However, we can do better by constructing an addition/subtraction circuit as shown in Figure 5.

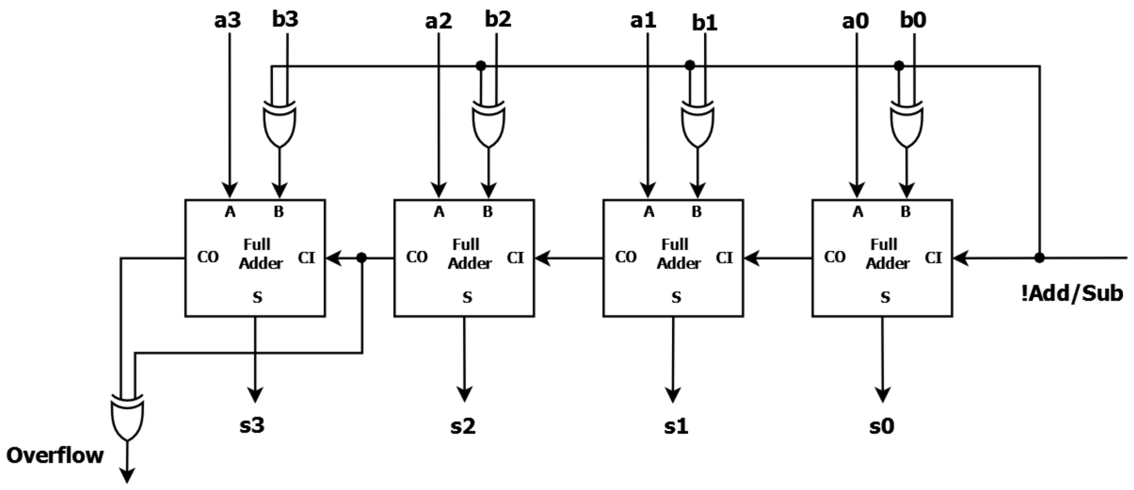


Figure 5: 4-bit Addition/Subtraction Circuit with Overflow Detection

Remember that subtraction is simply addition in which the B operand is complemented, and to complement a two's complement binary number, we must invert the binary digits and add one to the result. Base on your knowledge of full adder circuits and XOR gates, convince yourself that the diagram in Figure 5 is properly subtracting when **!Add/Sub** is HIGH. Similarly, use Figure 1 and Table 1 to help prove that the overflow detection logic in Figure 5 works as expected. Create the gate-level schematic for the 4-bit addition/subtraction circuit with overflow detection.

3.2 2:1 Multiplexer Design

If you have not already done so, read over the background section on multiplexers. Using your new found knowledge, create a truth table for a 1-bit wide, 2:1 multiplexer, utilize a Karnaugh Map to solve for the minimized boolean expression, and draw the gate-level schematic. Extend the design to a 4-bit wide, 2:1 multiplexer. Be sure to provide the full gate-level schematic in your pre-lab submission.

3.3 4-bit ALU Design

At this point, we have acquired the individual pieces we need to finish our simple ALU design. Examine Figure 6 and determine how the control signals, **c1** and **c0**, affect **Result**. Create a table with three columns, c1, c0, and Op, such that c1 and c0 correspond to the ALU control signals and op is the operation the ALU will perform (i.e. Add, Subtract, or AND). Unfortunately, the design in Figure 6 has an error in it. To be specific, the AND operation should not signal an Overflow but could depending on the inputs, A and B. Convince yourself that the aforementioned error exists and suggest a solution to fix the issue. Create the gate-level schematic of your final ALU design, incorporating your suggested modification. Note the the bit-wise AND functional block is nothing more than an array of AND gates.

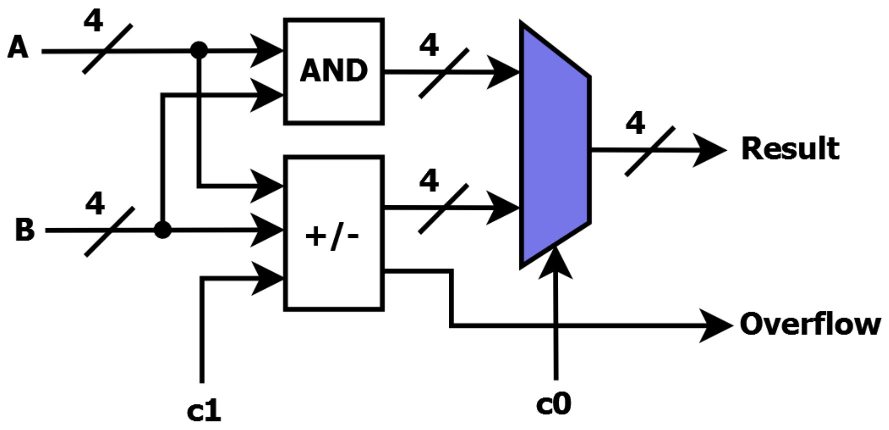


Figure 6: Simple ALU Design

3.4 Pre-lab Deliverables

Please include the following items in your pre-lab write-up.

1. Examples demonstrating how the circuit in Figure 5 adds and subtracts and how the overflow detection circuit works.
2. Truth table and minimized boolean expression for a 1-bit wide, 2:1 multiplexer.
3. Gate-level schematics for the Addition/Subtraction unit, 1-bit and 4-bit 2:1 multiplexers, and final ALU design.
4. Table of ALU operations and suggested design modification.

4 Lab Procedure

For the laboratory experiments below, consult the previous labs for wiring up the DIP switches and LEDs. Please note that the following experiments build on each other so be conservative with your bread-board space and leave each experiment constructed until you have completed the entire lab assignment.

4.1 Experiment 1

Implement and test your addition/subtraction unit. You may use the DIP switches to supply the input operands, A and B; however, you will have to use a jumper wire for the control signal, **!Add/Sub**. Display the output of the addition/subtraction unit using the LEDs in your lab kit and ensure your circuit is operating properly using the examples your created for the pre-lab. Do not forget to verify the operation of the Overflow detection logic. Demonstrate your progress to the TA once complete.

4.2 Experiment 2

Implement and test the 4-bit wide, 2:1 MUX design. As before, you may use the DIP switches for the inputs, A and B, and hard-wire the selection line. The LEDs should display the output of the MUX. Demonstrate your progress to the TA once complete.

4.3 Experiment 3

Using the circuits you built previously, construct and test the final ALU design. Wire the input operands to the DIP switches and the LEDs to the outputs (i.e. **Result** and **Overflow**). Vary the control signals, **c1** and **c0**, by manually connecting them to either power or ground. Verify the circuit operation matches the Op table you constructed in the pre-lab. Fix any issues your circuit may have including those associated with the **Overflow** signal. Once the final design is complete, demonstrate your progress to the TA.

5 Post-lab Deliverables

Please include the following items in your post-lab write-up in addition to the deliverables mentioned in the *Policies and Procedures* document.

1. Provide all design items found in the pre-lab deliverables. If you found that a design needed corrections while executing the lab, supply the updated version of that material.
2. Create a table with the following columns: $Control$, A, B, $Result$, and $Overflow$. Fill in sixteen rows of the table with values from your bread-boarded circuit, where twelve of the rows show the result of the addition/subtraction unit with atleast two rows demonstrate overflow.
3. Determine the maximum gate delay through your final ALU circuit assuming each gate has a delay of 1 unit. Highlight the critical path on the gate-level schematic.
4. Draw a diagram showing how to construct an 8:1 multiplexer from 4:1 and 2:1 multiplexers. Do **NOT** draw a gate-level schematic.

6 Important Student Feedback

The last part of lab requests your feedback. We are continually trying to improve the laboratory exercises to enhance your learning experience, and we are unable to do so without your feedback. Please include the following post-lab deliverables in your lab write-up.

Note: If you have any other comments regarding the lab that you wish to bring to your instructor's attention, please feel free to include them as well.

1. What did you like most about the lab assignment and why? What did you like least about it and why?
2. Were there any section of the lab manual that were unclear? If so, what was unclear? Do you have any suggestions for improving the clarity?
3. What suggestions do you have to improve the overall lab assignment?