

18-100: Intro to Electrical and Computer Engineering

LAB02: Adder Lab

Writeup Due: Thursday, February 3rd, 2022 at 10 PM

Name: Kevin Wang

Andrew ID: kgw

How to submit labs:

Download from this file from *Canvas* and edit it with whatever PDF editor you're most comfortable with. Some recommendations from other students and courses that use Gradescope include:

pdfescape.com A web-based PDF editor that works on most, if not all, devices.

Preview Pre-installed default MacOS PDF Editor.

iAnnotate A cross-platform editor for mobile devices (iOS/Android).

*If you have difficulties inserting your image into the PDF, simply append them as an extra page to the END of your lab packet and mark the given box. **Do NOT insert between pages.***

If you'd prefer not to edit a PDF, you can print the document, write your answers in neatly and scan it as a PDF. (*Note: We do not recommend this as unreadable lab reports will not be graded!*). Once you've completed the lab, upload and submit it to *Gradescope*.

Note that while you may work with other students on completing the lab, this writeup is to be completed alone. Do not exchange or copy measurements, plots, code, calculations, or answer in the lab writeup.

Your lab grade will consist of two components:

1. Answers to all lab questions in your lab handout. The questions consist of measurements taken during the lab activities, calculations on those measurements and questions on the lab material.
2. A demonstration of your working lab circuits and conceptual understanding of the material. These demos are scheduled on an individual basis with your group TA.

Question:	1	2	3	4	Total
Points:	10	15	15	10	50
Score:					

Lab Outline

This lab aims to familiarize students with the principle of *Hierarchical Design*: complex digital logic circuits can be constructed bottom-up from fundamental gate-level building blocks.

1. The XOR Gate
2. Adder Circuits
3. 2-Bit Adder
4. 3-Bit Adder
5. Adder/Subtractor (Bonus)

Equipment Required

Breadboard
Digital Multimeter
Diagonal Cutters

Power Supply
Wire Strippers
Needle-nose Pliers

Bill of Materials

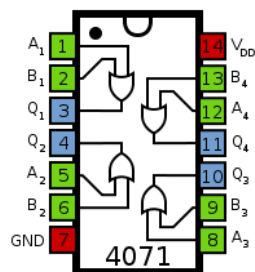
7x  10k Ω Resistor
1x OR Gate IC Chip (CD4071BE)
2x AND Gate IC Chip (CD4081BE)
3x XOR Gate IC Chip (CD4070BE)

3x Red LEDs
1x Yellow LEDs
2x 4-Ch DIP Switch
Jumper Wire

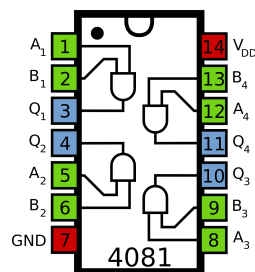
Small Group Check-off Circuits

- ☐ 3-Bit Adder (pg. 8)
- ☐ 3-Bit Adder/Subtractor (pg. 10, Bonus)

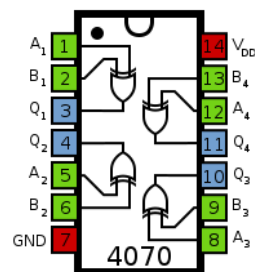
Pinouts



CD4071 OR Gate



CD4081 AND Gate



CD4070 XOR Gate

Note the U-shaped notch indicating the orientation on the pins.

1. XORs and Arithmetic Logic

The Exclusive OR gate is a logic function similar to an OR gate except outputs a zero when both inputs are high (hence the “exclusive”, A or B but not both). This gate forms the foundation of boolean arithmetic and we will use its properties extensively in this lab to build a 3-Bit Adder.

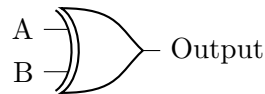
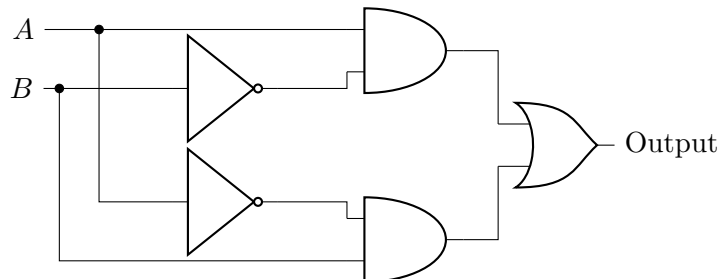


Figure 2: An XOR Gate

Given the equation for an XOR gate: $\text{Output} = A \cdot \bar{B} + \bar{A} \cdot B$, we can use this information to create an equivalent logic circuit using AND, OR and NOT.



Here's an example of XOR test circuit. **You do not have to make the circuit in Figure 3**, but you can use your kit's four-channel switch to recreate this circuit on your breadboard to get a better idea of how an XOR gate works. Don't forget about the pull-down resistors!

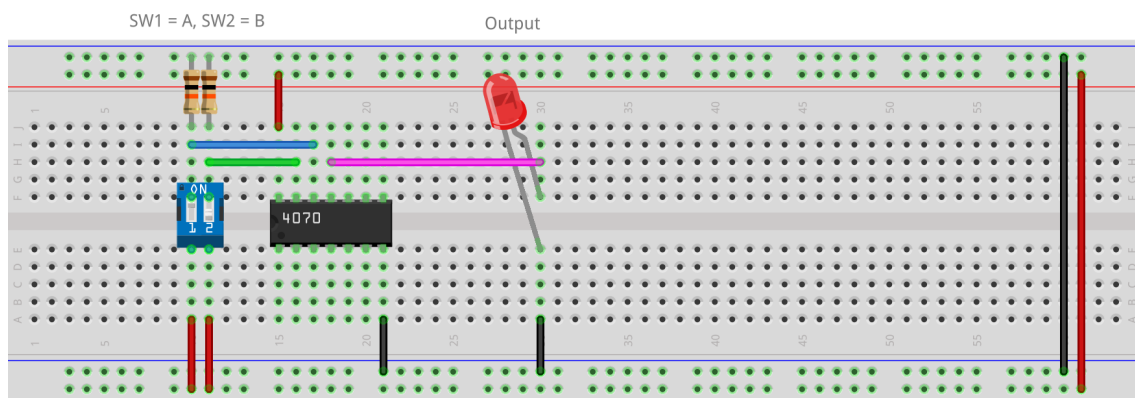


Figure 3: Single XOR Circuit Breadboard Diagram with 4070 IC chip

2 pts

1.1 Fill out the truth table for the XOR gate.

A	B	$\bar{A}$	$\bar{B}$	$A \cdot \bar{B}$	$\bar{A} \cdot B$	Output
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

2 pts

- 1.2 Add the following 1-bit binary numbers together (*Hint: think about how you would add these numbers in decimal*).

$$\begin{array}{rclcl}
 0_2 & + & 0_2 & = & \underline{0}_2 \\
 0_2 & + & 1_2 & = & \underline{1}_2 \\
 1_2 & + & 0_2 & = & \underline{1}_2 \\
 1_2 & + & 1_2 & = & \underline{10}_2
 \end{array}$$

3 pts

- 1.3 What's different about adding $1_2 + 1_2$ from the other 1-bit addition operands? What comparisons can you draw to decimal addition?

The sum has two digits. This is like when you carry over a 1 in decimal addition (i.e. $1+1=10$).

3 pts

- 1.4 Compare the operations you did to the truth table for the XOR. What is the same, and what is missing?

If we set high to 1 and low to 0, like it is in the XOR truth table, the sum of the value in binary is equivalent to the boolean expression of $A \oplus B$. However, the exception is when A and B are both true; instead of being 0, $1_2 + 1_2$ carries over to be 10_2 .

2. Adder Circuits

The solution to the above problem is the inclusion of a carry bit (c_{out}) on the output. This what we refer to as a *Half Adder*. In Figure 4, two 1-bit binary numbers a_0/b_0 are added and the sum is s_0 .

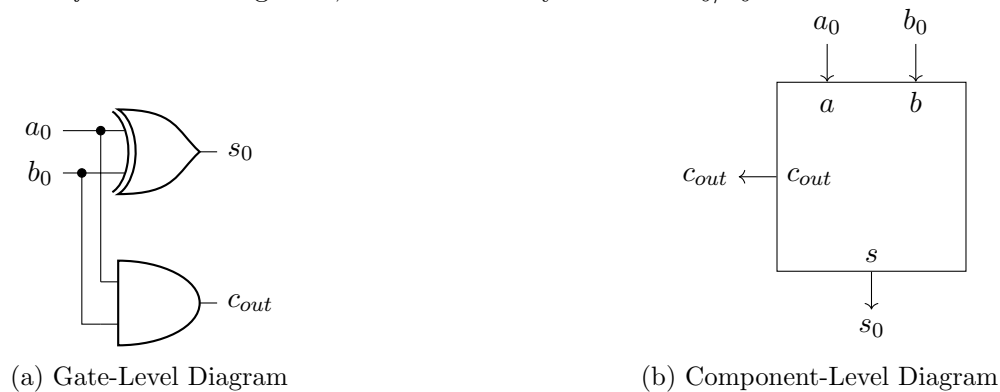


Figure 4: Half Adder Representations

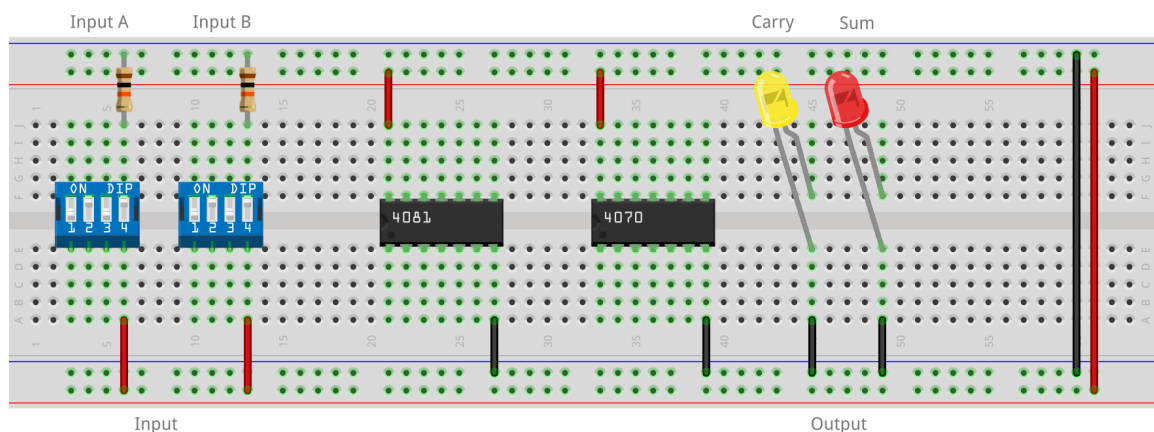
2 pts

2.1 Fill out the truth table for the Half Adder.

a_0	b_0	s_0	c_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

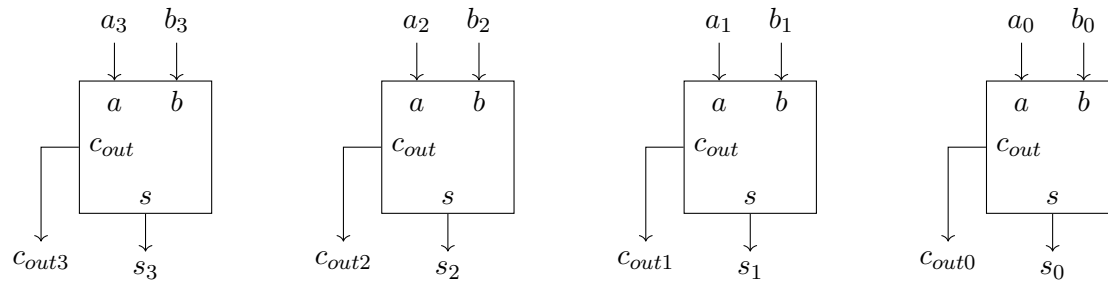
10 pts

2.2 Construct the half adder on your breadboard. The gate-level wiring is left up to you. **Your breadboard will be *completely filled* by the end. See the final diagram in Figure 9 as a reference. Follow the lab by building each diagram incrementally and check with TAs if you need help.**



Verify your circuit functions correctly by flipping the switches and comparing the output with the truth table. **Make sure to test your circuit thoroughly or you risk restarting *everything*! Any errors in this circuit will carry forward into the rest of the lab!**

So now, what if we want to add numbers bigger than one bit? We could just use multiple half adders however you'll see a problem appears pretty quickly:



What do we do with the carry bit? We need some way to include the carry data in our calculation. The *Full Adder*, shown below, takes in the carry-in bit c_{n-1} . In Figure 5(b), two 1-bit binary numbers a_n and b_n are added. The sum produced by addition is s_n . The final carry out bit is c_n .

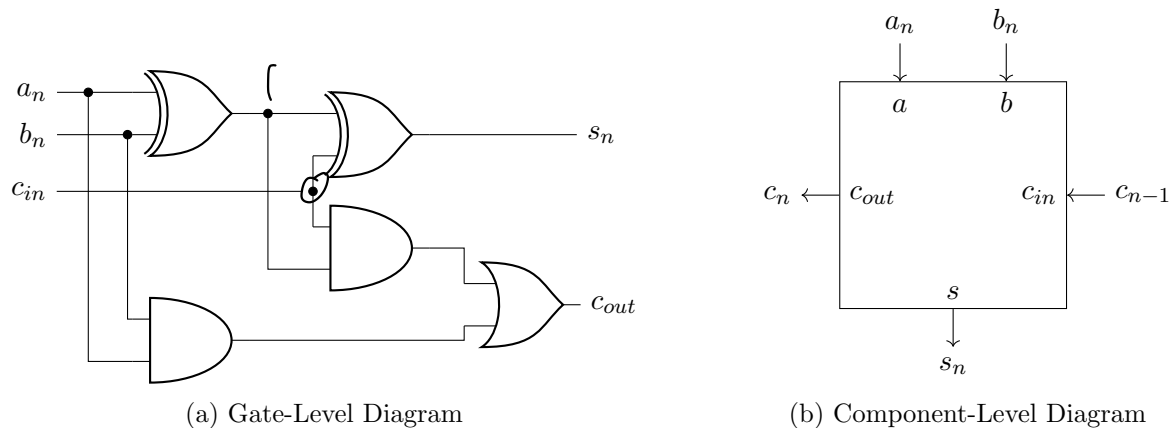


Figure 5: Full Adder Representations

3 pts

2.3 Fill out the truth table for the Full Adder.

a_n	b_n	c_{in}	c_{out}	s_n
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

3. 2-Bit Adder

The 2-Bit Adder comprises of a Half Adder and a Full Adder. In Figure 6, C_{out} , produced by the Half Adder, is connected to the C_{in} of the Full Adder. The final carry bit is generated from the Full Adder.

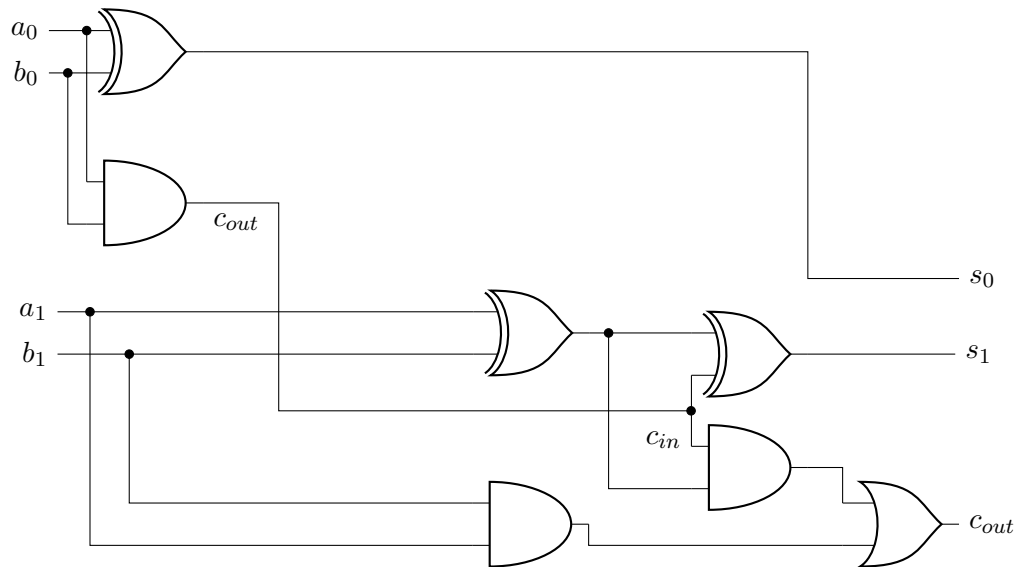


Figure 6: Gate-Level Diagram of 2-Bit Adder

In Figure 7, two 2-bit binary numbers a_1a_0 and b_1b_0 are added. The sum produced by addition is s_1s_0 . The final carry out bit is c .

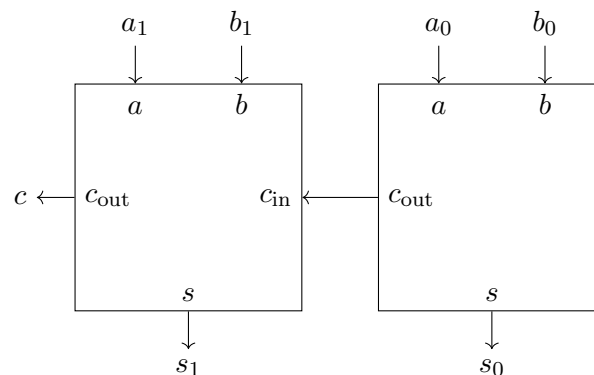


Figure 7: Component Diagram of 2-Bit Adder

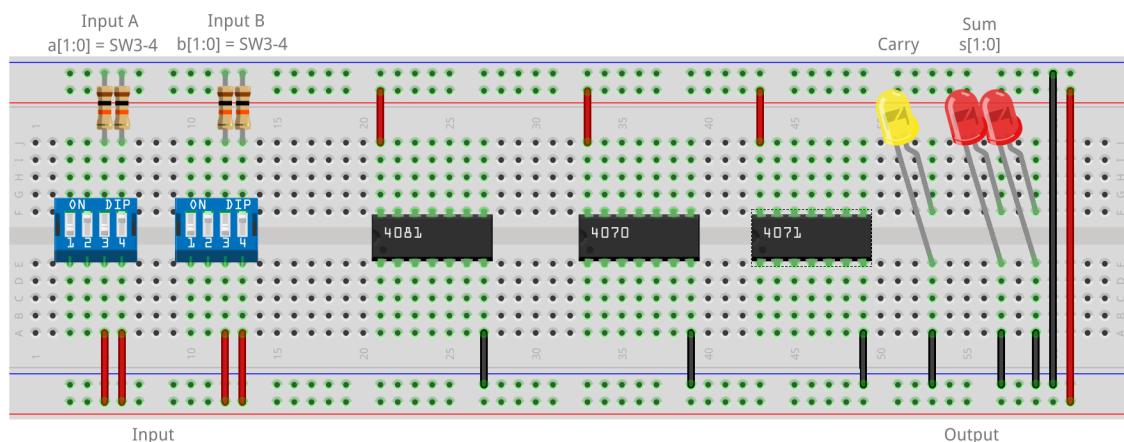
5 pts

3.1 Fill out the truth table for the 2-bit adder. (*Hint: you might want to use these entries later to test your circuit implementation*).

a_1	a_0	b_1	b_0	s_1	s_0	c_{out}
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	1	1	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	0	0	1
1	0	1	1	0	1	1
1	1	0	0	1	1	0
1	1	0	1	0	0	1
1	1	1	0	0	1	1
1	1	1	1	1	0	1

10 pts

3.2 Build upon your existing circuit implementations of a half adder, construct a full adder and connect the carry line to form a 2-bit adder. (the breadboard layout has been updated below)



4. 3-Bit Adder

In Figure 8, two 3-bit binary numbers $a_2a_1a_0$ and $b_2b_1b_0$ are added. The sum produced by addition is $s_2s_1s_0$. The final carry out bit is c .

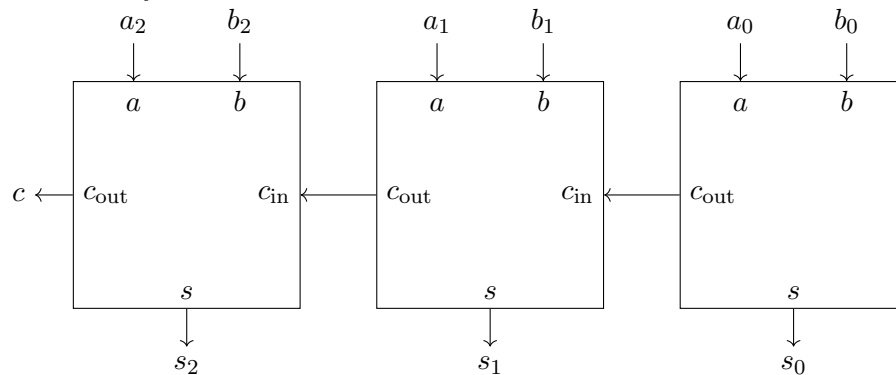


Figure 8: Component Diagram of 3-Bit Adder

10 pts

- 4.1 Build upon your understanding of the 2-Bit Adder and construct the 3-Bit Adder on breadboard. (Hint: *Connect another full adder to your previous circuit.*)

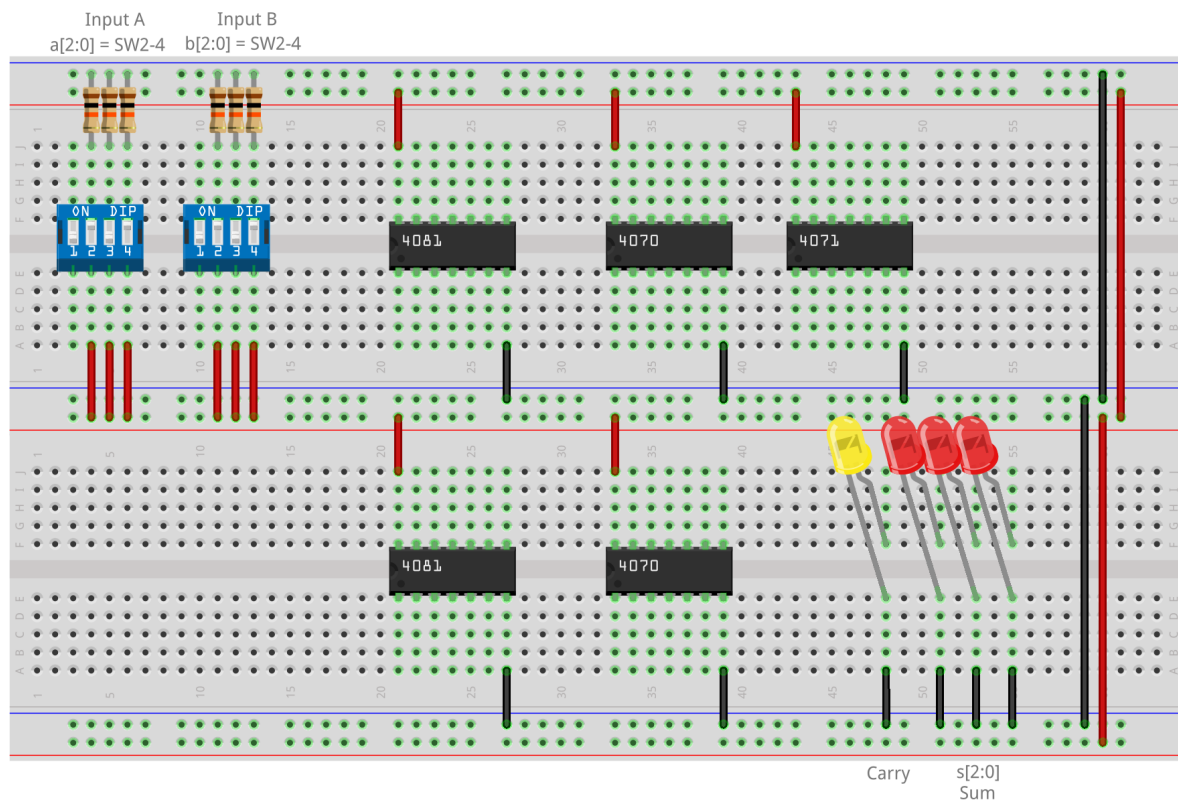


Figure 9: 3-bit Adder Breadboard Layout Diagram

Verify the circuit that you built by flipping the switches and comparing state of the LEDs to the theoretical outputs.

⚠ Do NOT take your circuit apart yet! You will need it for lab checkoff!

A 3-bit adder has over 60 different possible inputs; we can't expect you to test them all! However, we have provided the table below as a resource to help you ensure your circuit functions properly. Try to come up with a few example inputs/outputs for the 3-Bit Adder to test.

a_2	a_1	a_0	b_2	b_1	b_0	s_2	s_1	s_0	c_{out}
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	1	0
0	0	0	0	1	1	0	1	1	0
0	0	0	0	1	0	0	1	0	0
1	1	1	1	1	1	1	1	0	1
1	0	1	1	1	0	1	0	1	1
1	1	0	0	0	1	1	1	1	0
1	1	1	1	0	1	1	0	1	1

5. Adder/Subtractor Circuit

Now that you've mastered the ripple-carry adder, let's modify our circuit to support subtraction as well.

5 bonus

5.1 Build upon your 3-bit adder to support two's complement subtraction.

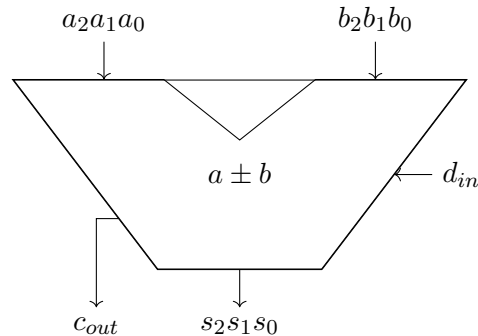


Figure 10: 3-Bit Adder/Subtractor

Input d is the toggle to switch from addition to subtraction. When d is 0, the circuit should behave the same as in *Part 4*. However, when d is 1, the circuit should perform 3-bit 2's complement subtraction (a.k.a. $a - b$). Some examples are shown below. **Hint: Think about what logic gate you can place at the beginning of your circuit that will "reverse" addition.**

d_{in}	a_2	a_1	a_0	b_2	b_1	b_0	C_{out}	s_2	s_1	s_0
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1
1	1	1	1	1	1	1	0	0	0	0
1	0	0	1	0	1	0	1	1	1	1
1	0	1	0	1	1	0	1	1	0	0

You will need to make a modification to your original 3-bit adder and then expand upon that circuit.

⚠ Warning: This circuit will require you to modify your 3-bit adder circuit. Talk to your TA before attempting.

⚠ Do NOT take your circuit apart yet! You will need it for lab checkoff!