CURIE Academy, Summer 2014
Lab 1: Computer Engineering – Hardware Perspective
Sign-Off Sheet

NAME: _____

NAME: _____

DATE: _____

Sign-Off Milestone		TA Initials
Part 1.A	Experiment with LED	
Part 1.B	Experiment with Inverters	
Part 1.C	Develop NAND Gate	
Part 2.A	Experiment with Logic Gates	
Part 2.B	Develop Parity Checker	
Part 3.A	Experiment with Half-Adder	
Part 3.B	Develop Full-Adder	
Part 3.C	Develop Multi-Bit Adder	
Optional Extension:		
Optional Extension:		
Optional Extension:		

CURIE Academy, Summer 2014
Lab Handout 1: Computer Engineering – Hardware Perspective

Prof. Christopher Batten
School of Electrical and Computer Engineering
Cornell University

In this lab assignment, you and your partner will explore the field of computer engineering from the hardware perspective by by gradually building a simple binary adder. Feel free to consult the lab notes as you work through the lab. In Part 1, you will experiment with basic circuits before developing a NAND gate. In Part 2, you will experiment with basic logic gates before developing a parity checker. In Part 3, you will gradually develop the full multi-bit ripple-carry adder unit described in the lab notes. For each part and various subparts you will need to have a teaching assistant observe the desired outcome and initial the appropriate line on the sign-off sheet. **Turn in your sign-off sheet at the end of the lab session.**

Before beginning, take a look at the materials on the lab bench which you will be using to complete the lab. You will notice that we have already inserted some transistors and logic gates into your breadboard for you.

- Breadboard-based prototyping platform
- Black power cable with barrel connector
- LED
- Resistor
- Wire cutter and stripper

Part 1. Understanding Basic Digital Circuits

In this part, you will wire up some simple circuits using the prototyping platform. You will need to cut some wire from the wire spools in the lab, and then strip the ends. Feel free to ask a TA for help. You can also use some pre-made wire jumpers from the supplies desk in the lab. Consult the lab notes for more information on the connectivity that is inside the breadboard. Before getting started make sure your prototyping platform is *not* plugged in (i.e., the barrel plug from the wall adapter should not be plugged into the breadboard power supply on the right of the prototyping platform).

Part 1.A Experiment with LED

Wire up the simple LED circuit shown in Figure 1 using the breadboard diagram in Figure 2 as an example. You will need to find some free space on the breadboard to insert the LED and resistor.

Sign-Off Milestone: Once you have wired everything up, have an instructor verify that things are connected correctly; then the instructor will demonstrate how to plug in the barrel connector, test the circuit, and turn the board on/off using the switch on the breadboard power supply. Try putting the LED in both directions.

Critical Thinking Questions: What do you think would happen if we used a resistor with higher resistance? This would be equivalent to using an even narrower pipe in our water circuit analogy. What do you think would happen if we used a resistor with lower resistance? Would would happen if we put two resistors in series or in parallel?

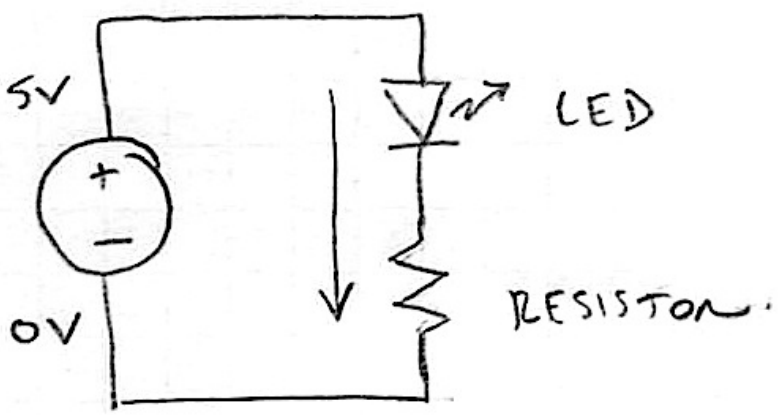

Figure 1: Simple LED Circuit

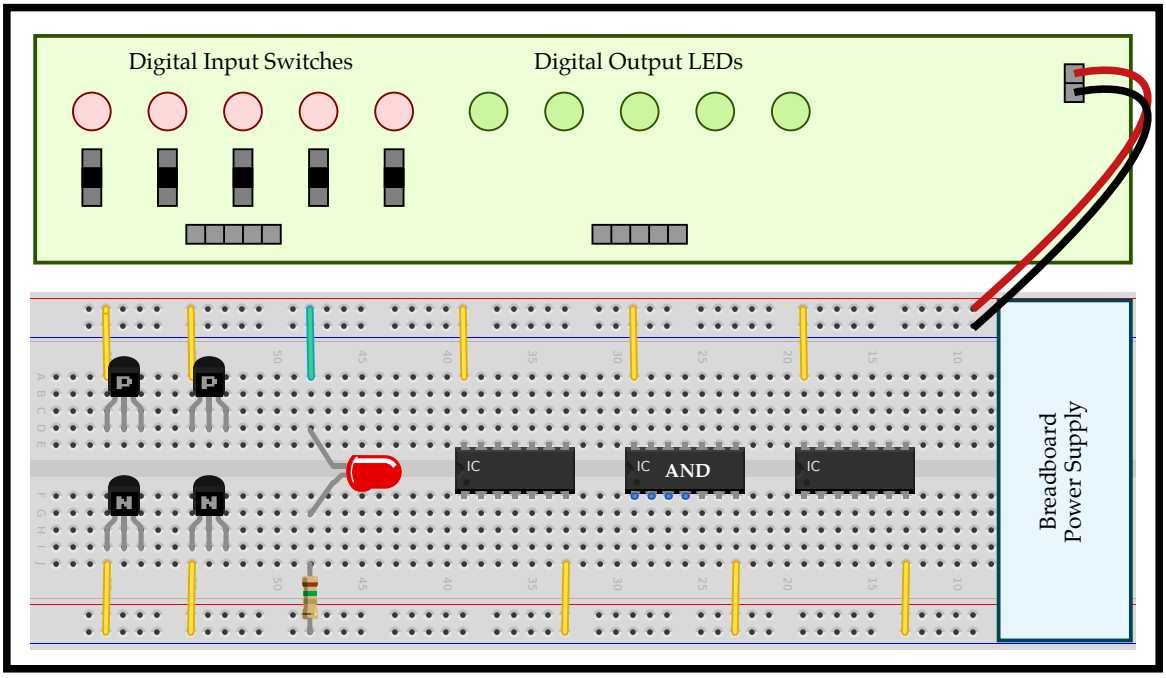

Figure 2: Simple LED Circuit Implemented on Prototyping Platform

Part 1.B Experiment with Inverters

Turn off your prototyping platform using the switch on the breadboard power supply. Wire up a PMOS and an NMOS transistor to create a single inverter. Use the circuit in Figure 3 and the breadboard diagram in Figure 5 as a guideline. Note that we have already inserted two PMOS transistors at the top of your breadboard and two NMOS transistors at the bottom of your breadboard. We have also already wired up VDD and ground to the transistors. We are using a digital input switch to send a digital value into the circuit and a digital output LED to observe the output of the circuit. Once you have double checked your wiring, turn on the board and try toggling the input to the inverter. When the input to the inverter is one is the output zero? When the input to the inverter is zero is the output one?

Turn off your prototyping platform using the switch on the breadboard power supply. Wire up four transistors to create two back-to-back inverters as shown in Figure 4. You will need to work on your own to figure out how to map this circuit to the actual breadboard.

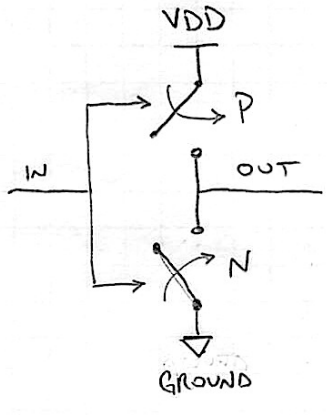

Figure 3: Inverter Circuit (P = PMOS transistor, N = NMOS transistor)

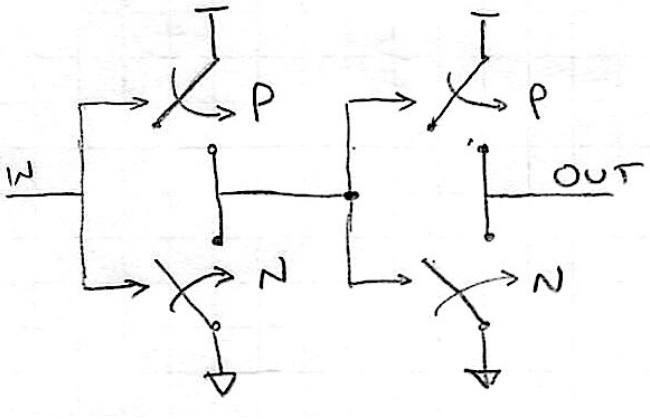

Figure 4: Back-to-Back Inverter Circuit (P = PMOS transistor, N = NMOS transistor)

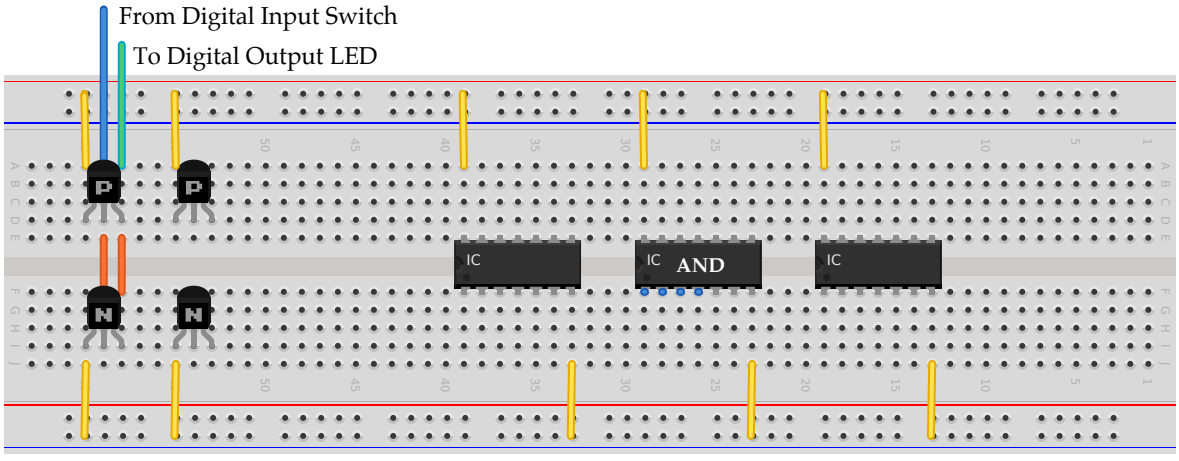

Figure 5: Breadboard Diagram for Testing Inverter Circuit

Sign-Off Milestone: Once you have wired the two back-to-back inverters up, have an instructor verify that things are connected correctly; then the instructor will turn on the board and we can observe the functionality of the circuit.

Critical Thinking Questions: What do you think would happen if we cascaded three inverters? four inverters? What would happen if we tied the output of a three inverter chain to the input of the chain?

Part 1.C Developing a NAND Gate

In this subpart, you will develop a more complex circuit that implements a new kind of logic gate. Before beginning, make sure that the instructor removed one of the yellow jumpers; if you still have four yellow jumpers (one per transistor) have an instructor come over to remove one. Turn off your prototyping platform using the switch on the breadboard power supply.

Consider the circuit shown in Figure 6. Before continuing can you determine what this circuit does using pencil-and-paper? Try all four possible combinations of input values and derive the expected truth table. Once you have studied the circuit using pencil-and-paper, go ahead and implement the circuit on the breadboard.

Sign-Off Milestone: Once you have wired the logic gate, have an instructor verify that things are connected correctly; then the instructor will turn on the board and we can observe the functionality of the circuit.

Critical Thinking Questions: Why is this gate called a NAND gate? Can we use a NAND and the inverter from the previous part to implement an AND gate? Can you think of how to implement a NOR gate?

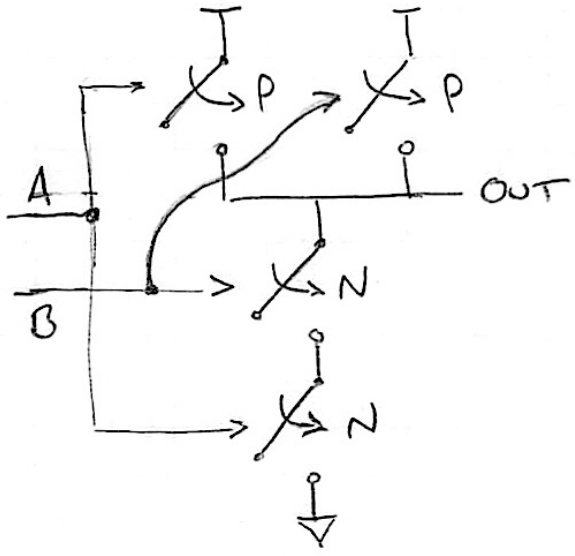

Figure 6: NAND Circuit (P = PMOS transistor, N = NMOS transistor)

Part 2. Understanding Basic Logic Gates

In the previous part, we learned about basic circuits and how to use these circuits to implement logic gates. In this part, we will begin working with the logic gate abstraction. Using this abstraction, we no longer need to worry about the details involved with transistors. We can simply focus on the logical operation of each gate.

Part 2.A Experiment with AND, OR, XOR Gates

Turn off your prototyping platform using the switch on the breadboard power supply. Wire up the AND gate that is contained within the second chip as shown in Figure 7. Turn on the board and try all four input combinations. Does the output correspond to the expected truth table? Wire up the other two chips using the digital input switches and digital output LEDs in a similar fashion and test their functionality. Use a truth table and the information from the lab notes to determine which chip contains AND gates and which chip contains OR gates.

Sign-Off Milestone: Once you have determined which chip contains the each type of gate, show an instructor your truth tables and demonstrate the operation of either the OR or the AND gate.

Critical Thinking Questions: So far we have see four two-input logic gates: NAND, AND, OR, and XOR. How many different logic gates are possible if we limit ourselves to gates with just two inputs and one output?

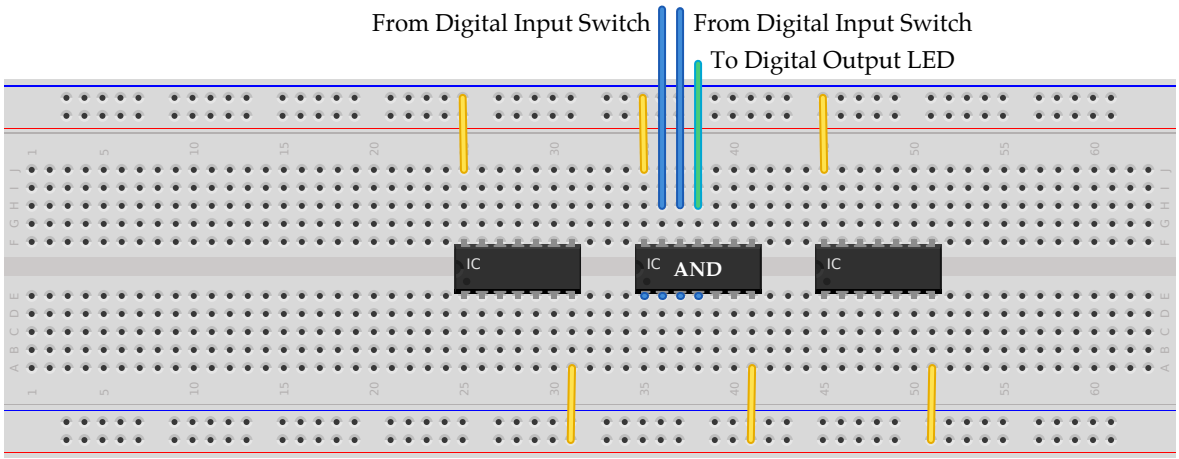

Figure 7: Breadboard Diagram for Testing AND Gate

Part 2.B Develop Parity Checker

In this subpart, you will develop a more complex *parity* hardware unit for a four-bit input. A parity unit should produce a zero when the number of ones in the input is even and should produce a one when the number of inputs in the input is odd. Parity units are actually quite common to help detect errors when sending messages between systems. The sender can calculate the parity of a message and send the parity bit along with the message. The receiver can then also calculate the parity of the message and compare it to the parity bit sent along with the message. If the parity bits do not match then it is likely that one of the bits in the message was corrupted.

Figure 8 illustrates how we can implement a parity hardware unit using three XOR gates. Wire up the three XOR gates on your breadboard, and verify its functionality by filling out the truth table below.

Sign-Off Milestone: Once you have wired up the parity unit, show an instructor its operation and verify the corresponding truth table.

Critical Thinking Questions: How would we implement a parity hardware unit for five, six, or more bits?

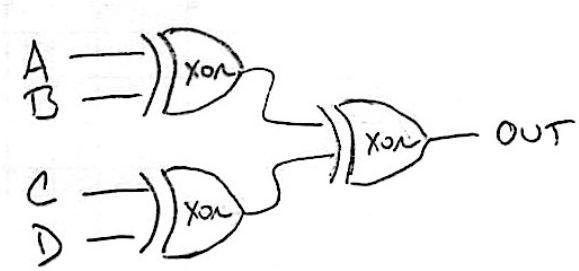

Figure 8: Four-Bit Parity Hardware Unit

A	B	C	D	out

Part 3. Building a Simple Calculator

In the previous part, we started building more interesting hardware units using logic gates. In this final part, we will incrementally build a simple binary adder unit. We will begin by implementing a half-adder, then a full-adder, and finally we will compose multiple full-adders to implement a multi-bit adder unit. You should review the background material on binary arithmetic and implementing adders in the lab notes.

Part 3.A Experiment with Half-Adder

Wire up the half-adder shown in Figure 9.

Sign-Off Milestone: Once you have wired up the half-adder unit, show an instructor its operation and verify the corresponding truth table.

Part 3.B Develop Full-Adder

Wire up the full-adder shown in Figure 10. This unit is complicated enough, that you should plan out your wiring ahead of time using the template in Figure 11.

Sign-Off Milestone: Once you have wired up the full-adder unit, show an instructor its operation and verify the corresponding truth table.

Part 3.C Develop Multi-Bit Adder

We could continue to add more and more logic gates to implement a multi-bit adder, but we will instead leverage modularity to simplify building a ripple-carry adder. Figure 12 illustrates a four-bit ripple-carry adder. As a first step, ask an instructor for an integrated full-adder board which is basically a printed circuit-board that contains the exact same circuit you implemented in the previous subpart. Take a close look at the integrated full-adder board. You will see that it has three input switches corresponding to the three one-bit inputs. It also includes a "mode switch" which can be set to either "independent mode" or "chain mode". Set the integrated full-adder board to "independent-mode" and test its operation. Power the integrated full-adder board using a 9 V battery. Verify that it produces the same truth table as the full-adder you implemented in the previous subpart.

Now find another lab group that is at the same step. Chain both of your integrated full-adder boards together to create a hardware unit capable of adding two two-bit numbers. Both full adders should be powered using their own batteries. The integrated full-adder board on the right should be set to "independent mode" and the integrated full-adder board on the left should be set to "chain mode". Verify that this simple two-bit ripple-carry adder works.

Sign-Off Milestone: Once you have verified the two-bit ripple-carry adder works, show its operation to an instructor. Pick two two-bit numbers, calculate the expected result, and then confirm that your ripple-carry adder produces the correct result.

Critical Thinking Questions: The delay of a digital signal is often measured in how many gates a signal must traverse to get from an input port to an output port. The performance of a hardware module is usually limited by the *worst case delay* through all possible paths. Can you reason about what might be the worst case delay through our two-bit ripple-carry adder?

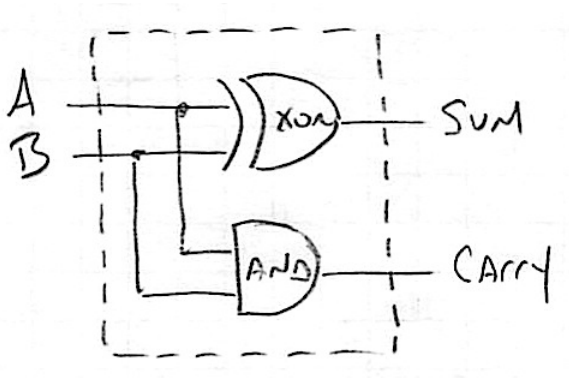

Figure 9: Half-Adder

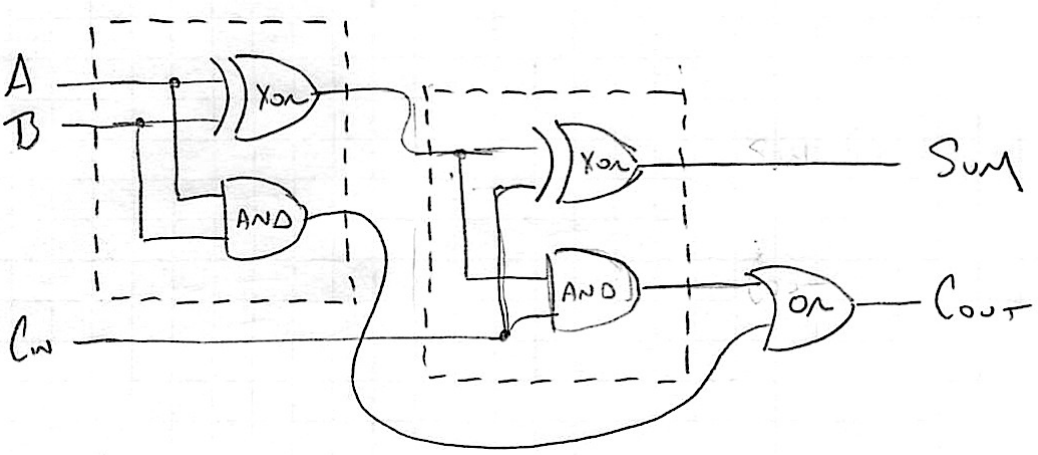

Figure 10: Full-Adder

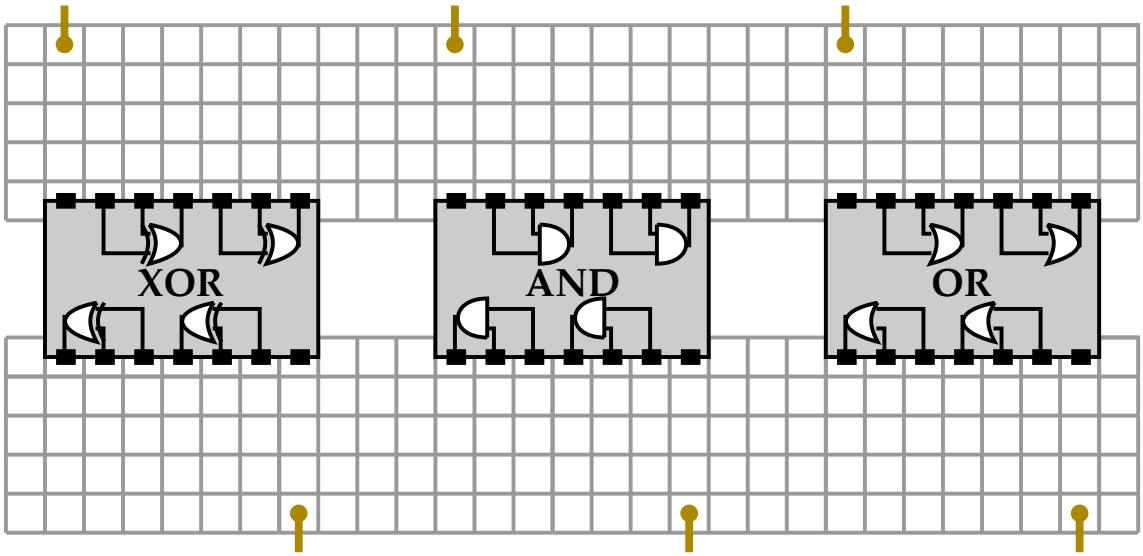

Figure 11: Planning Diagram for Breadboard Implementation of Full-Adder

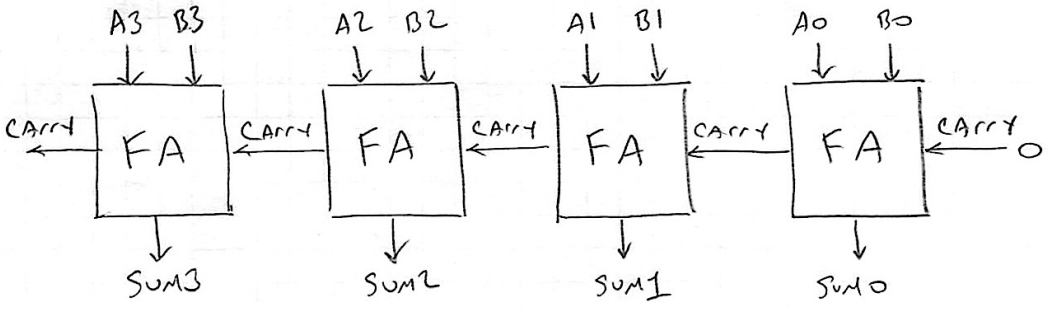

Figure 12: Four-Bit Ripple-Carry Adder (FA = full-adder)

Optional Extensions

Parts 1–3 are the required portions of the lab. If you complete these parts quickly, you are free to try one or more of the following extensions which are roughly arranged in order of difficulty. These extensions are purely optional; do not feel like you must attempt them.

- **Larger Adders –** Work with other groups to build a three-bit or four-bit adder using the integrated full-adder boards. What is the largest adder you can build? Don't forget to work through the expected result on paper first so you can verify that your multi-bit adder works.
- **Using Your Breadboard Implementation –** If you are confident that your breadboard implementation of the full-adder module works, try connecting it to the integrated full-adder board to create a two-bit adder.
- **Larger Parity Checkers –** Build a larger parity checker that can calculate the parity for an eight-bit input. You will need to use a total of seven XOR gates. Since each XOR chip only has four XOR gates, you will need to work with another group to build this circuit.
- **Implement a NOR Gate –** Use PMOS and NMOS transistors to implement a NOR gate. A NOR gate should produce a one when both inputs are zero, otherwise it should always produce a zero.
- **Subtraction Unit –** We can use your ripple-carry adder to subtract two binary numbers. To calculate A - B, we need to modify B before using the ripple-carry adder. We first *invert* every bit in B (i.e., every one becomes a zero and every zero becomes a one). Then we need to add one to the inverted version of B. If we add the result to A, it will have the same effect as subtracting B from A. In other words, using binary arithmetic A - B = A + !B + 1, where !B means invert every bit. Figure 13 illustrates how to implement a subtractor. You can implement the subtractor either by working out the inversion and increment by one on paper and then using the ripple-carry adder from Part 3. For the truly ambitious, you could implement a two-bit subtractor by working with three other groups. Combine each groups' breadboard implementation of the full-adder module along with an inverter implemented using NMOS/PMOS transistors to create the entire hardware module.

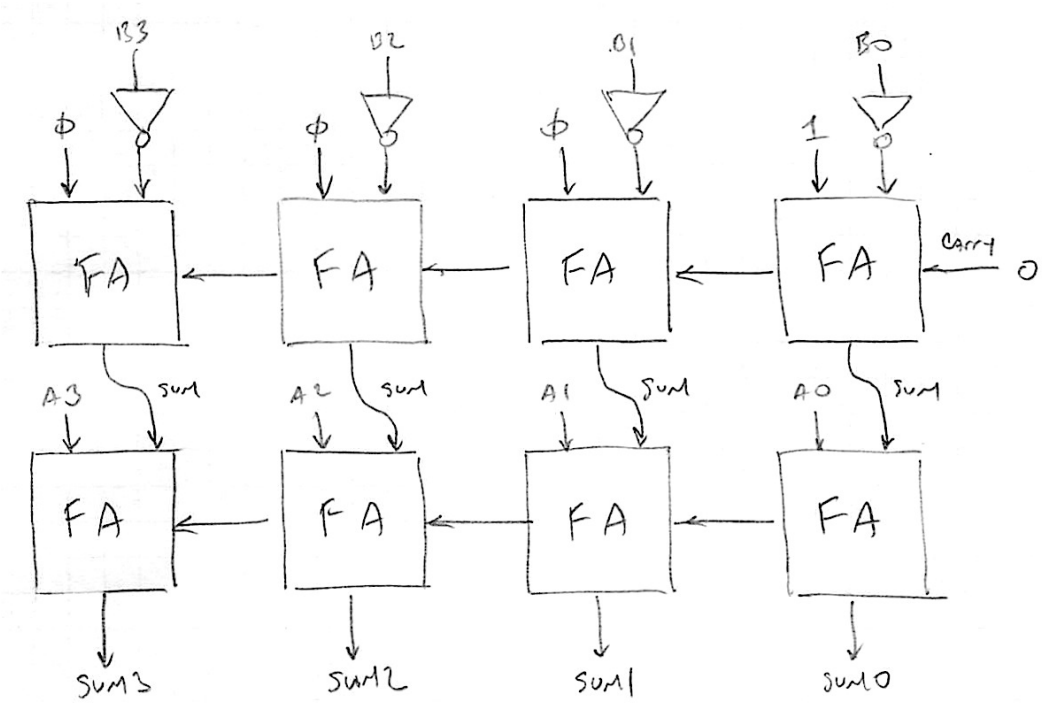

Figure 13: Four-Bit Ripple-Carry Subtractor (FA = full-adder)