Hardware Lab 1: Building a Half and Full Adder

Prerequisites: Before beginning this laboratory experiment you must be able to:

- Communicate with the Arduino board using the Firmata Test application. Refer to the instructional videos for Week 2 on how to download and install the Arduino IDE, upload the firmware, download and test Firmata.
- Use a prototyping board and a voltmeter.
- Interpret a truth table.
- Read Boolean algebraic notation.
- Recognize the Boolean algebraic expressions characterizing a full adder.

Equipment: Personal computer, Arduino Uno (or equivalent) with Firmata firmware loaded, USB cable, breadboard, breadboarding wire bundle.

Integrated Circuits: You will need the following IC to complete this lab:

- (1) 7404 (Hex Inverters)
- (1) 7408 (Quad 2-input AND gate)
- (1) 7432 (Quad 2-input OR gate)
- (1) 7486 (Quad 2-input XOR gate)

Objective: In this laboratory you will gain some experience building and debugging combinational logic circuits using TTL IC's. You will also gain experience using canonical sum-of-products and product-of-sums forms along with the exclusive OR (XOR) operator.

Outcomes: When you have completed the tasks in this experiment you will be able to:

- Describe the truth tables that characterize the addition of two single bit numbers.
- Write the Boolean algebraic expressions that characterize the sum and carry functions for the half adder using both the product-of-sums and sum-of-products canonical forms.
- Realize the sum and carry functions using TTL hardware.
- Debug TTL circuits using a logic indicator and a voltmeter as testing instruments.
- Write the Boolean algebraic expressions that characterize the sum and carry functions for the full adder.
- Build and debug a full adder.

Introduction

In this laboratory exercise you will gain knowledge how to design and build a hardware digital logic circuit, starting from truth tables or Boolean algebraic equations. For this Hardware Lab 1 you will build and test two digital logic circuits, the half adder and the full adder, using TTL logic gates on a rapid hardware prototyping platform, consisting of a Breadboard and a microcontroller.

Developing a hardware "prototype" of a digital logic circuit requires a certain set of tasks to be completed in a step-by-step fashion. This sequence will be extremely helpful if modifications are required, for example because requirements change or the circuit stops working because of a hardware fault. The following list should provide a guideline on how to proceed in the development process:

1. Explain the function of the circuit. This is important so that you can document what the circuit is supposed to do.
2. Describe the logic in form of a truth table and convert the truth table into a set of canonical sum-of-products or product-of-sums algebraic equations.

3. Find out which logic gates you have available. Do you want to build it using AND/OR/NOT gates or do you only want to use one type of gate (NAND, NOR)?
4. Eventually reduce the canonical algebraic equations to a minimal form, which will save you hardware and reduce your build time. Based on the gate availability, you will decide on a sum-of-products or product-of-sums or convert the equation to be implemented with NAND or NOR gates only.
5. Build your circuit in a simulator. This way, you can check if it works correctly before going through the effort of building it in hardware. You can also develop a "testing protocol" to verify if your hardware works correctly after you finish building it.
6. After drawing the schematic diagram in the simulation software, label the input and output pins of all the logic gates with the pin numbers of the chip that you want to use, referring to the pin-out diagrams. This tremendously cuts down on build time and helps in testing the circuit.
7. Optional in our case, draw a layout of how the chips will be placed. While a breadboard allows flexible wiring, try to match the "flow" of the logic signals on your schematic with the actual "flow" of the wiring. This will minimize the length of the wires and frequent wire "crossings".
8. Build the circuit, preferably in separate blocks. This makes the next testing step easier.
9. Test your circuit (blocks) based on the testing protocol you have developed when simulating the logic circuit. If everything works, connect the blocks and test the complete circuit if it works as expected.
10. If you find any flaw in a circuit (block), you have to perform "debugging". The following paragraph will list the steps involved in debugging.

Steps Used in Debugging

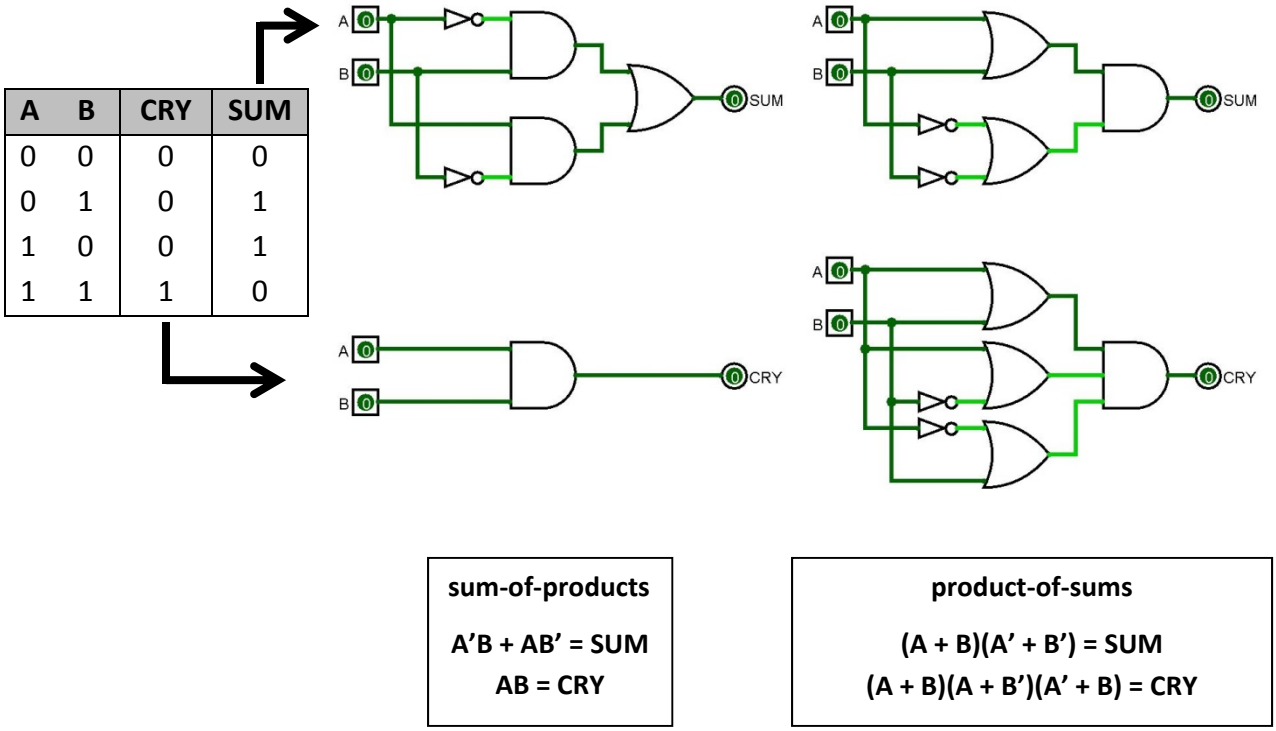
- Measure the V_{CC} and GND pin logic values for each IC used in your design using the Voltmeter. Visually trace all the wires supplying V_{CC} and GND to all the chips and make sure that all V_{CC} and GND connections are secure.
- Using a schematic with pin numbers of your circuit, write down the digital values that you would expect to measure at each pin if the circuit was working correctly. Probe the signals gate-by-gate, going from the output back to the input. Methodically observe the output for every combination of input variables for each function. At the first wrong output, measure the input and output voltages and compare them to the values you would expect from Hardware Lab 0.
- Your measurements might indicate one of the following problems:
    - If you make a measurement that has a voltage level that is neither within the logic 1 nor logic 0 voltage range or is near a limit of one of these ranges, either your input signal is disconnected (refer to Hardware Lab 0 where you tested the levels at open inputs) or you have two outputs connected to the same point. Correctly wire the circuit and complete the original testing.
    - If after re-wiring a missing input signal the voltage levels are still not within the expected range, remove the wire on the receiving end and test the signal directly at the wire. If the voltage level is correct, check all the other connections going to the same input. It can happen that another wire is connected to a wrong node or shorted to V_{CC} or GND. After checking these other connections, re-connect the receiving end and test again. It can also happen that receiving-end pin is shorted to V_{CC} or GND inside the IC. If checking the wiring does not help, try replacing the IC.
    - If a gate is defective, you will reach a point where the output of the gate is inconsistent with the input. If that is the case, replace the IC.

A list of common problems encountered in the lab are (roughly ordered from most to least frequent):

- V_{CC} and/or GND of an IC not connected.
- Wires making a poor or intermittent connection with the breadboard.
- Defective breadboard connection.
- Wires broken inside of the insulation.
- IC not properly inserted.
- Wires connected to the wrong pin.
- Two outputs connected.
- Defective IC's.

Task 1-1: Build and Test the SUM & CRY of the 1-Bit Half-Adder.

The 1-bit half adder should be a familiar circuit by now, since it was part of the lecture. To summarize, a 1-bit half adder has two inputs A and B, which represent two 1-bit inputs. The adder then performs an arithmetic sum and drives two outputs, SUM and CRY, with the latter being the carry-out. We can directly write down the truth table and the canonical algebraic equations in sum-of-product and product-of sum form, as shown in Figure 1.



A	B	CRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

sum-of-products

A'B + AB' = SUM

AB = CRY

product-of-sums

(A + B)(A' + B') = SUM

(A + B)(A + B')(A' + B) = CRY

Figure 1: Truth table, canonical sum-of-products and product-of-sums algebraic equations and logic gate schematics for a half adder.

Now we have to decide which of the circuits shown in Figure 1, we actually want to build. To make our life easier and improve the reliability of the circuit, we want to minimize the number of logic gates and the number of connections. While we could try to simplify the circuits, let's select the ones from Figure 1 directly. While the SOP implementation of CRY is the obviously simpler circuit, it is not obvious if the SOP or POS implementation of the SUM function is "better". For this lab, we chose the product-of-sum implementation of the SUM circuit, mainly because it is easier to remember the wiring and the overall output is coming from an AND gate, just like the output for the CRY function. After making the selection, we have to list the number and type of gates we need. That will in turn determine the chips we need to build the circuit on a breadboard.

- 2 logic NOT gates – one 74LS04 chip with 6 NOT gates per chip
- 2 logic OR gates with two inputs each – one 74LS32 chip with 4 OR gates per chip
- 2 logic AND gates with two inputs each – one 74LS08 chip with 4 AND gates per chip

While we could start building right away, we should look at step #6 in our first list again. It helps tremendously if we label the individual gates in the schematic we drew up in our simulation software (for example Logisim) with the pin numbers of the actual gates in the chip (see Figure 2) we want to use. Figure 3 shows how to accomplish this.

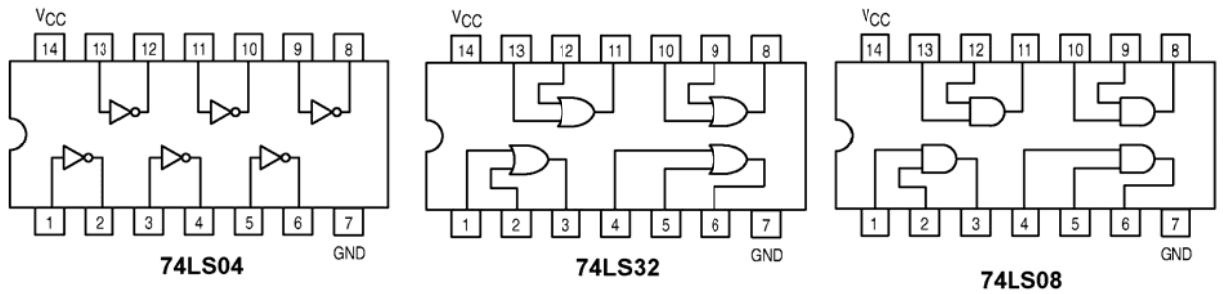


Figure 2: Pin-out diagrams of the logic chips used in Hardware Lab 1, Task 1-1. (Copyright of Motorola. Used with permission.)

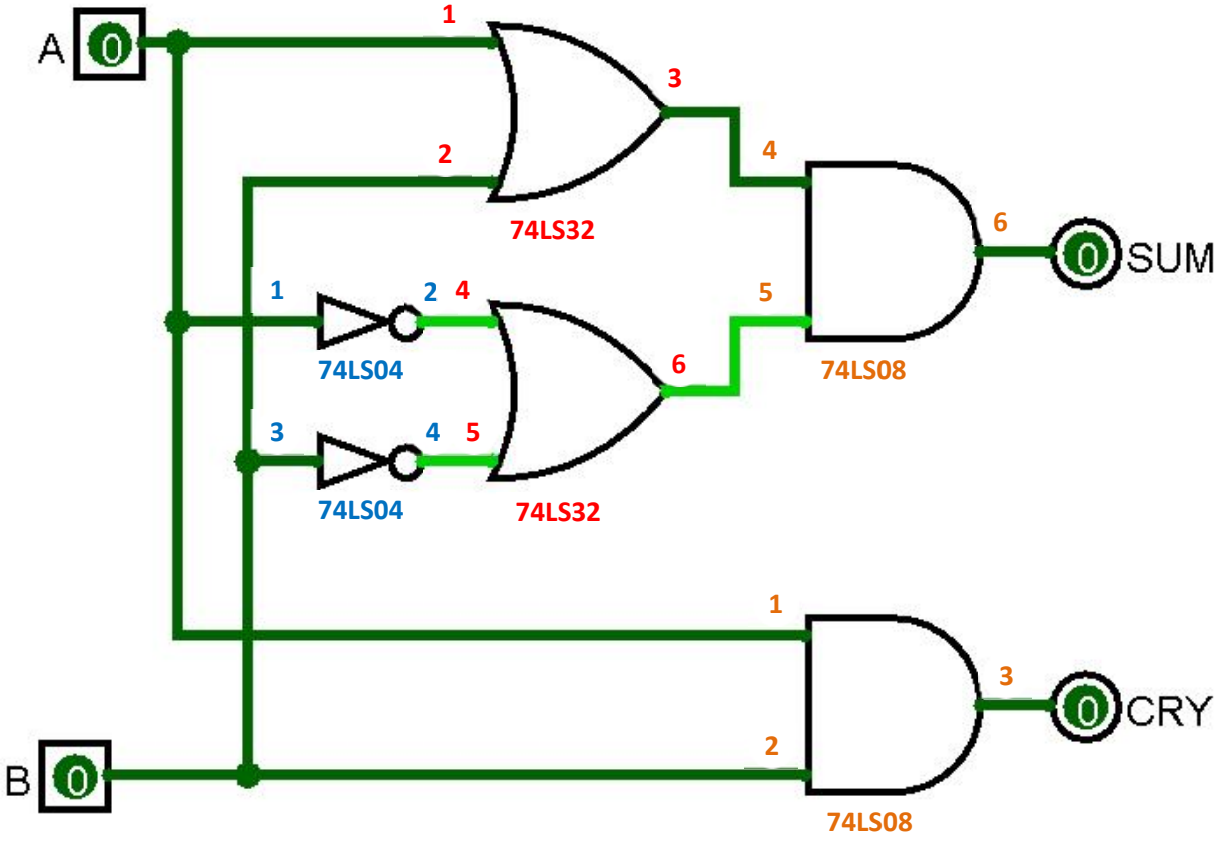


Figure 3: Schematic diagram of the half adder with chip numbers and pin numbers added.

Based on the pin-out diagrams in Figure 2 and the labeled schematic in Figure 3 you can now start building the circuit on the breadboard:

1. Insert the three chips over the gutter of the breadboard, leaving a couple of columns in between the chips. Make sure that the notch of all the chips is pointing to your left.
2. Connect the Arduino GND to the GND bus.
3. Connect the GND pins (Pin 7 on each chip) to the GND bus.
4. Connect the Vcc pins (Pin 14 on each chip) to the +5V bus.
5. Complete the connections as shown in Figure 3.
6. Connect the Arduino board to your computer and start the Firmata Test Panel.
7. Once the Firmata test panel starts up, select Pins 2 and 3 to be outputs and Pins 6 and 7 to be inputs. Set all outputs to “Low” as shown in Figure 4.
8. Connect Arduino Pin 2 to the area of the breadboard that drives the “A” inputs of the chips.
9. Connect Arduino Pin 3 to the area of the breadboard that drives the “B” inputs of the chips.

10. Connect Arduino Pin 6 to Pin 6 on the 74LS08 IC. This will be the SUM signal.
11. Connect Arduino Pin 7 to Pin 3 on the 74LS08 IC. This will be the CRY signal.
12. Connect the Arduino +5 V supply to the +5V bus.
13. Test the circuit using the Firmata test panel on the computer. The indicators for Arduino Pins 6 and 7 should show “Low”. Now, set Arduino Pin 2 to “High”. If the circuit works correctly, Arduino Pin 6 should indicate “High”.
14. Go through all four possible settings of Arduino Pins 2 and 3. **Record the signal at Arduino Pins 6 and 7 for all input combinations on the Hardware Lab 1 Report Template**. Pin 6 corresponds to the SUM signal and Pin 7 to the CRY signal.

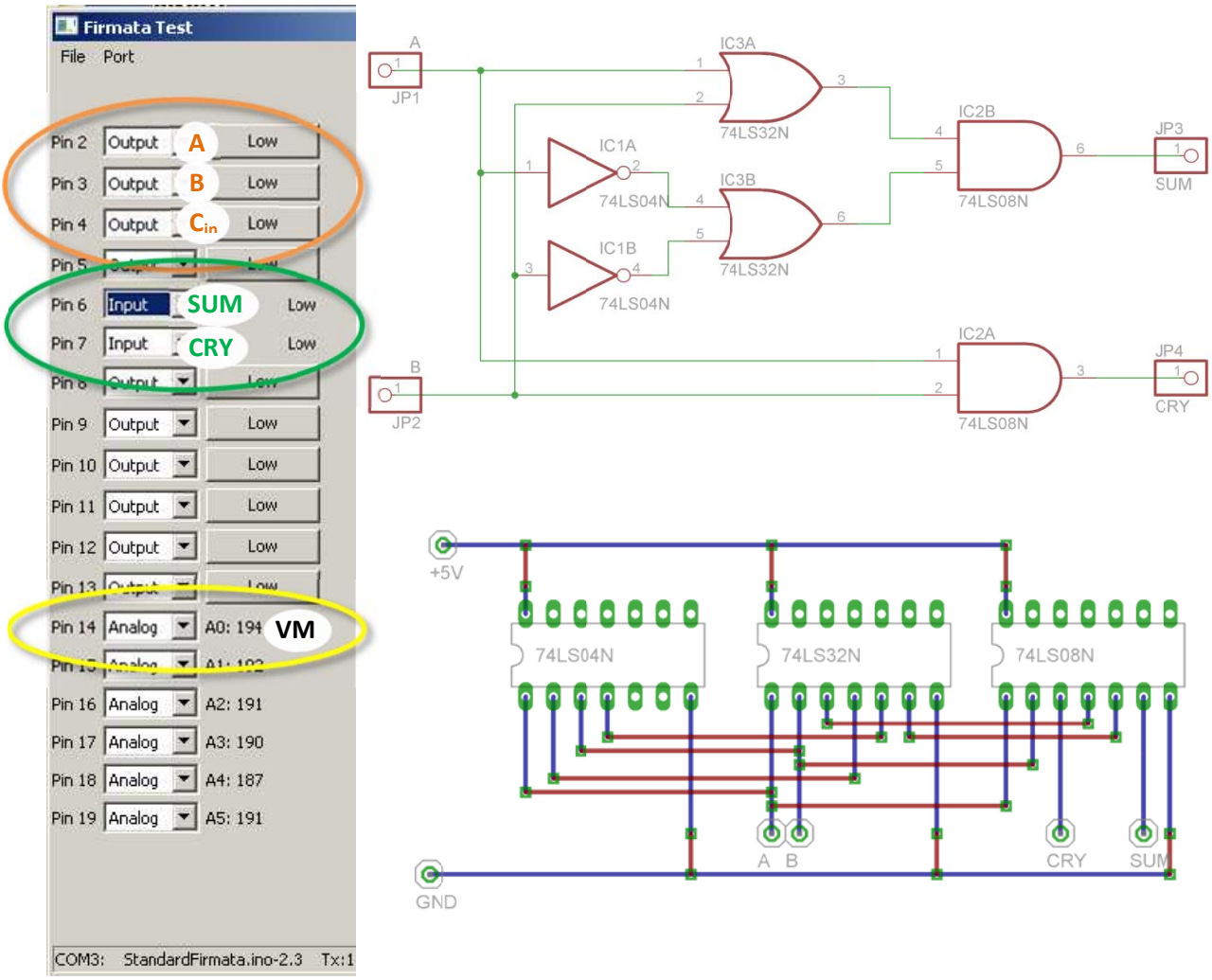


Figure 4: (Left) Firmata test panel used to interact with the digital logic circuit. The Analog Input A0 on Pin 14, labeled VM is only used for Task 1-2 “Debugging”. The output Pin 4 labeled C_{in} is only used for Task 1-3 “Building and Testing the Full Adder”. (Right) The top image shows a circuit diagram generated by professional circuit design software (Cadsoft Eagle), showing the similarity between the approach taken in Figure 3 and a professionally designed schematic, used to generate the layout for a printed circuit board. The lower image shows a wiring layout generated by Cadsoft Eagle, mimicking a breadboard-based design. The blue wires indicate the internal electrical connections, while the red wires indicate connections that have to be made in order to complete the circuit. Green dots indicate contact holes with wires or pins being plugged into the breadboard.

Task 1-2: Debugging

After you completed Task 1-1 and completed building a working 1-bit half adder, you should introduce a random error and try to find and correct it. You can for example ask somebody else to remove a wire or change some of the internal wiring. Alternatively, you can just randomly unplug one wire. Following the debugging steps describe in the introduction to this lab, observe how the outputs are behaving by looking at the logic indicators Pins 6 and 7 on the Firmata control panel when you switch through all four possible input combinations for signals A and B. If one or both of the output signals are wrong, apply the input combination for A and B for which you notice a wrong output.

Then, use a wire connected to the Arduino Analog Input A0 to probe the actual voltage levels at that output and trace the signal back to the input(s), following the pin diagram shown in Figure 3. When debugging, it is helpful to have the simulation software up and running to compare the values you "measure" with the logic values you expect to see in a working circuit. A table listing the IC pins from Figure 3 in which you enter the expected values and the measured values might also be helpful.

Briefly describe the approach you took and the steps involved in your debugging process on the Hardware Lab 1 Report Template. A few sentences are sufficient, just to convey the idea how you dealt with the problem and if you were actually successful to find the fault.

Task 1-3: Build and Test the SUM & CRY of the 1-Bit Full-Adder.

Moving on to the 1-bit full adder, we have to take care of a third input called C_{in}, while the number of outputs does not change. We can write down the truth table and the canonical expressions, however, building the canonical versions of the circuit would take a lot of effort. Here, we choose to simplify the circuit first, including using XOR gates (74LS86) to generate the SUM signal. Since we only have two-input gates available, we need to "stack" gates accordingly. Figure 5 shows the truth table and the minimized sum-of-product implementation of the 1-bit full adder, from which we see that NOT gates are no longer required.

A	B	C_{in}	CRY	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

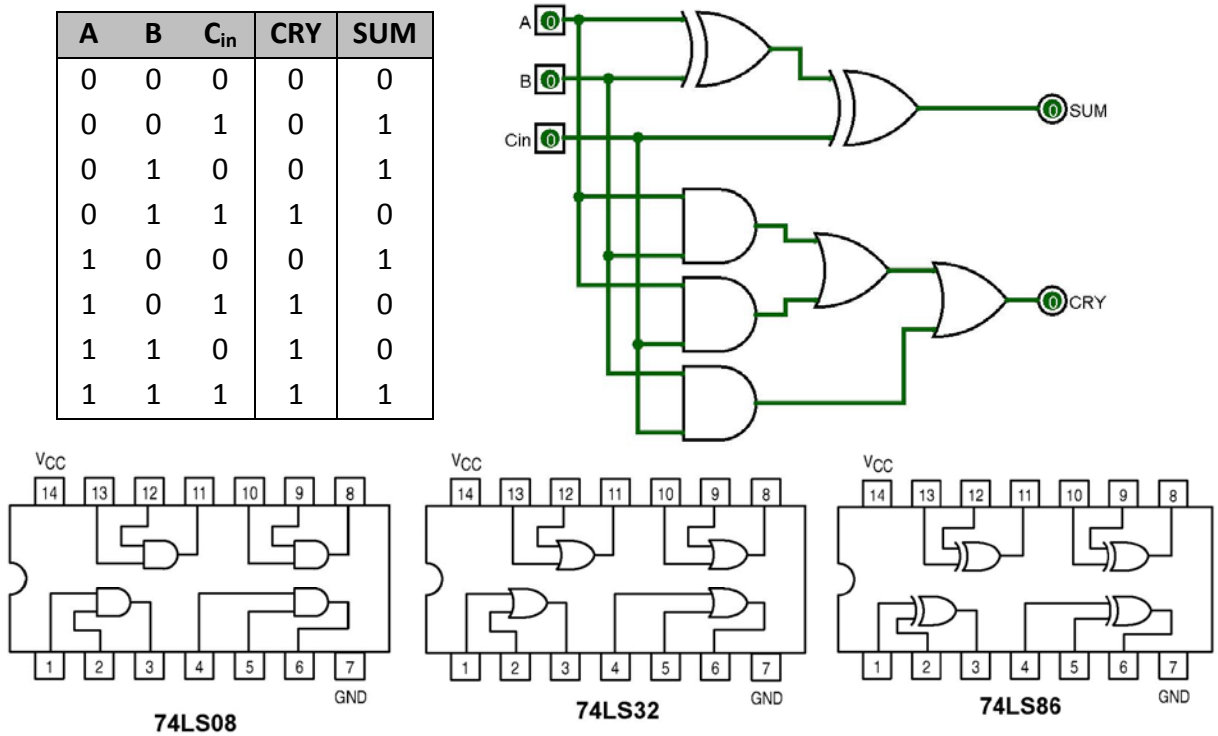


Figure 5: Truth table and logic gate diagram for a 1-bit full adder. The logic gate diagram is based on XOR gates for SUM and the minimum SOP for CRY. The figures on the bottom show the pin-outs for the integrated circuits used for building the 1-bit full adder in Task 1-3. (Copyright of Motorola. Used with permission.)

In this particular task, you have to repeat the procedure that we did in Task 1-1 when we built the 1-bit half adder. Specifically, you have to complete the following action items:

- Add the chip numbers and pin numbers in the schematic diagram of the full adder in Figure 5 and **insert the completed diagram into the Hardware Lab 1 Report Template.**
- Build the circuit on the breadboard, using Arduino Pin 4 to the area of the breadboard that drives the "C_{in}" inputs of the chips (see Figure 4).
- Test your circuit using the Firmata control panel for all possible input signal combinations and **record your results in the table on the Hardware Lab 1 Report Template.**

Task 1-4: Take a Photo of your Completed Circuit

Before you rip everything apart, take a photo of your completed circuit and attach it to the lab template. This will serve as a replacement for the "attendance stamp" that the in-person students have to get on their paper lab submission.