Jack Weissenberger

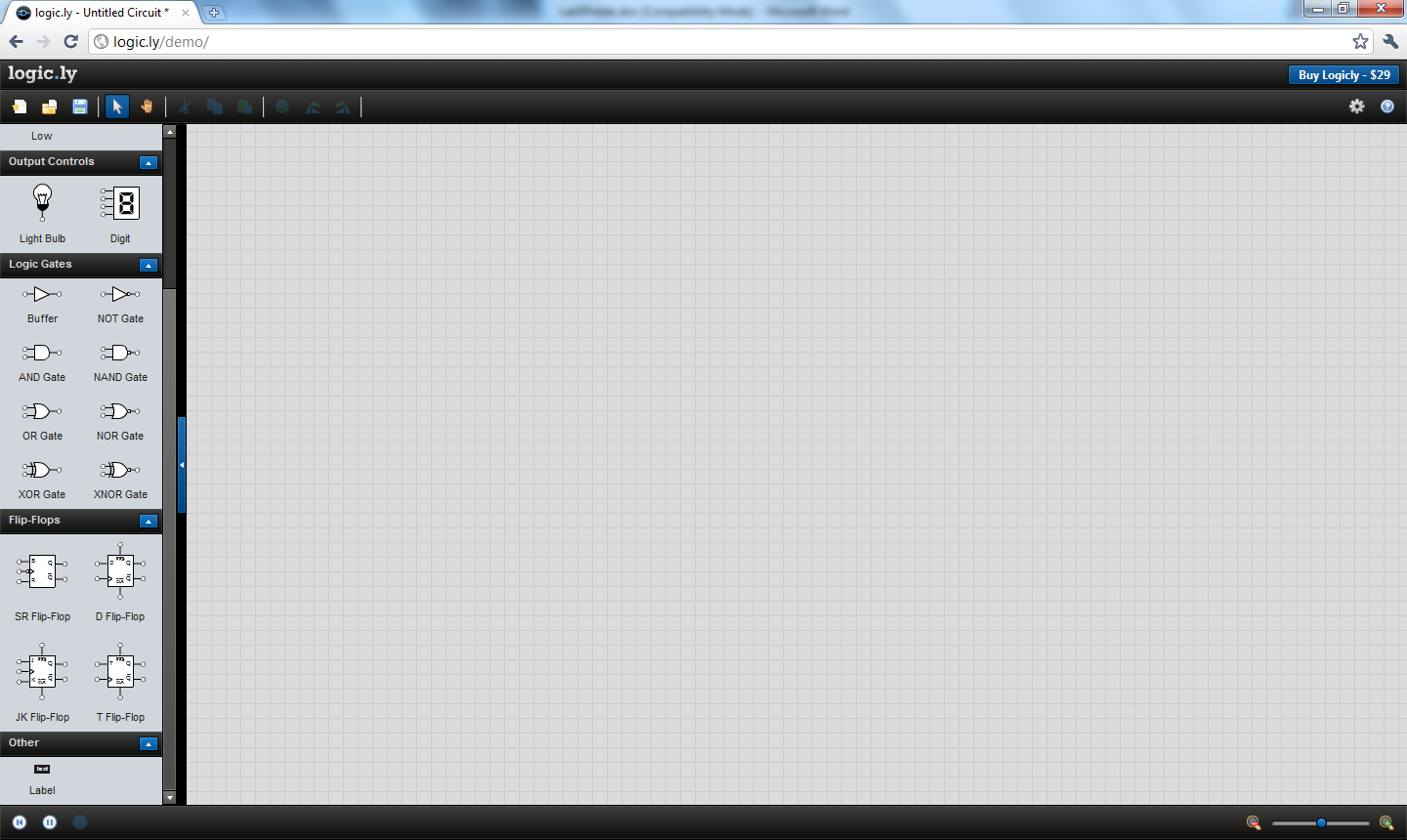
**Boolean Algebra, Truth Tables, Gates, and Circuits.**

**Part 0: Background and Instructions**

The purpose of this assignment is to reinforce and allow you to demonstrate what you have learned about logic design.

**Instructions**

* Create a folder called ***Logic\_Lab***.
* Download and start the Logicly software using the instructions on Sakai.
* Complete this assignment and submit to the assignment in Sakai using this document and any other material you are asked to submit. **See the end of this assignment for details.**

Hit Close on the small welcome window that may pop up when you first start the software. You should now see a window like the image at right:

Logicly is a program that allows you to simulate actual logic gate circuits. The interface provides a toolbox of logic gates and other components on the left side of the screen and a large, gray desktop area on the right. Components from the toolbox can be dragged onto the desktop and connected together.

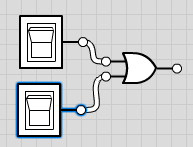
At this time, watch the short (3-minute) video found at the link given below. This video discusses a tool called LogicGateSimulator which is very similar to Logicly. The video provides an overview of how to go about designing circuits in such a tool. Don’t worry about the specific circuits the demo shows being built. Instead, focus on the processes being used: dragging and dropping gates, creating “wires” (links) between gates, and turning inputs on and off.

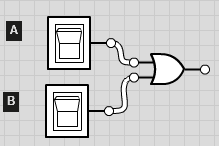
Video to watch: <http://www.kolls.net/gatesim/> (click on “Watch a 3 minute video demonstration”)

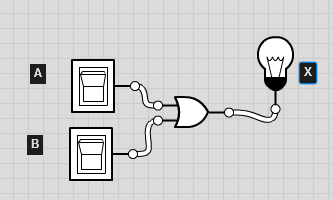
The major differences you will see when working with Logicly instead of LogicGateSimulator are that in Logicly the inputs are toggle switches and the outputs are light bulbs.

**Part 1: Practice with the Interface**

To learn more about Logicly, use it to verify the actions of the OR gate. Drag an *OR gate* onto the desktop. Drag two *Switch*es onto the desktop and connect them to the left-hand (input) side of the *OR* gate.



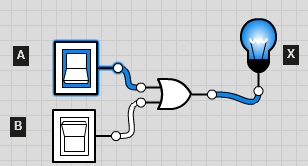
Note that you can drag *Labels* next to the switches and outputs to help me remember their purpose later on. Label the top input **A**, the bottom input **B**, and, in a little bit, the output **X**. In the rest of the lab, the inputs will be labeled in this same top-to-bottom {**A**, **B**, … } order.



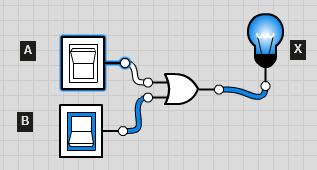
Add a *Light Bulb* output, connecting it to the right hand of the OR gate and labeling it **X**.

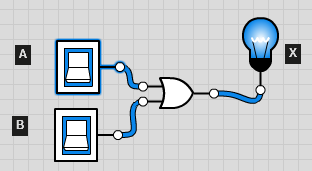
Test this gate setup. Notice that the input switches are off (white), indicating that they have the value **0**. The output light bulb is also off (white), indicating it has a **0** value. This is reasonable, since **0 + 0** (false OR false) should return the value **0** (false).

Try each of the other three combinations of inputs (the **+** symbol means Boolean **OR** operator):

**** **1 + 0**, **0 + 1**, and **1 + 1**. To turn an input “on” or “off”, just need to click on the appropriate switch and it will change color. “White” indicates **0** (off/False) and “Blue” indicates **1** (on/True).

(**True** **OR** **False**) 🡺 output is **True**

****(**False** **OR** **True**) 🡺 output is **True**

****

(**True** **OR** **True**) 🡺 output is **True**

From the pictures above, it appears the OR circuit is correct!

At this point, you should try to gain experience in working with the Logicly interface by experimenting. Some suggestions for experimenting are:

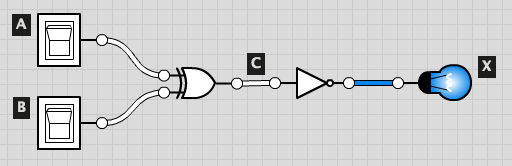
* Drag gates, inputs, and outputs onto the desktop and connect them. Rebuild the OR gate setup created above.
* Practice dragging between the connector hooks on inputs, outputs, and gates to make connections.
* **You can get wires to be straighter with right angles by clicking on the middle of the wire and moving the bubble around. Try it out.**
* Click on a wire connection to remove the connection.

**Exercise 1**

**1.1)** One of the circuits we will build in lab is called a NAND gate (“NAND” = “NOT‑AND”). A NAND gate is a combination of an *AND gate* and a *NOT gate*. The output of the AND becomes the input of the NOT, and the output of the NOT is the output of the entire NAND gate. (In words, a NAND gate outputs “true” when the two inputs are *not both* true.)

Fill out the truth table below with expected results, then, using Logicly, design a complete NAND circuit by using two *Switch* inputs, an *AND gate*, a *NOT gate*, and a *LightBulb* output connected in the manner described above. Complete the truth table below with the actual values for the output of the NAND gate as implemented above.

|  |  |  |  |
| --- | --- | --- | --- |
| **Input A** | **Input B** | **Expected Output** | **LogicLy Output (A *NAND* B)** |
| **0** | **0** | 1 | 1 |
| **0** | **1** | 1 | 1 |
| **1** | **0** | 1 | 1 |
| **1** | **1** | 0 | 0 |

**Exercise 2**

**2.1)** Consider the circuit shown at right, built in Logicly from a selection of components from the Logicly toolbox:

There are various different types of components used to create this circuit. Complete the following about this circuit:

1. This circuit has 2 (#) inputs **A B**
2. This circuit has 1(#) outputs. **X**
3. This circuit has 1(#) intermediate signals. Underline the name(s): **X**
4. This circuit has 1 (#) NOT gates.
5. This circuit has 0(#) NAND gates. It has an XOR
6. Write an equivalent Boolean expression for this circuit:

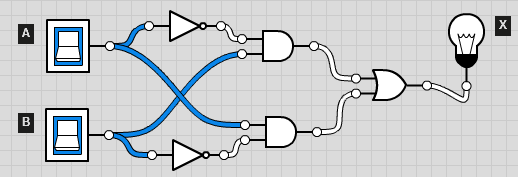
(A and B) or (Not A and Not B)

1. Complete the truth table for this circuit (include the expressions for **C** and **X**):

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** = \_\_\_\_\_\_\_\_\_\_\_ | **X** = \_\_\_\_\_\_\_\_\_\_\_ |
| **0** | **0** | 0 | 1 |
| **1** | **0** | 1 | 0 |
| **0** | **1** | 1 | 0 |
| **1** | **1** | 0 | 1 |

**Part 2: Determining the function of an unknown circuit**

The purpose of this portion of the lab is to provide hands-on experience in manipulating and designing simple Boolean logic circuits using a logic simulator.

**Step 1**

Implement this Boolean logic circuit in Logicly:

Fill out the table below. Based upon those observed results, which fundamental Boolean operation does this circuit implement?

XOR

|  |  |  |
| --- | --- | --- |
| **Input A** | **Input B** | **Observed Output X** |
| **0** | **0** | **0** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **0** |

### Step 2

In Logicly, go to File -> Save Document As. Name the file **UNKNOWN** and save the file to your Lab folder that you created above.

**Step 3**

Clear your Logicly workspace by selecting Edit->Select All and then Edit->Delete

**Part 3: Building a half-adder**

The truth table for a half-adder, as derived in class, is shown below. (Be sure to understand how to determine the correct Sum and Carry outputs for this truth table.)

|  |  |  |  |
| --- | --- | --- | --- |
| **Half Adder** | | | |
| **Input A** | **Input B** | **Sum** | **Carry-out** |
| **0** | **0** | **0** | **0** |
| **0** | **1** | **1** | **0** |
| **1** | **0** | **1** | **0** |
| **1** | **1** | **0** | **1** |

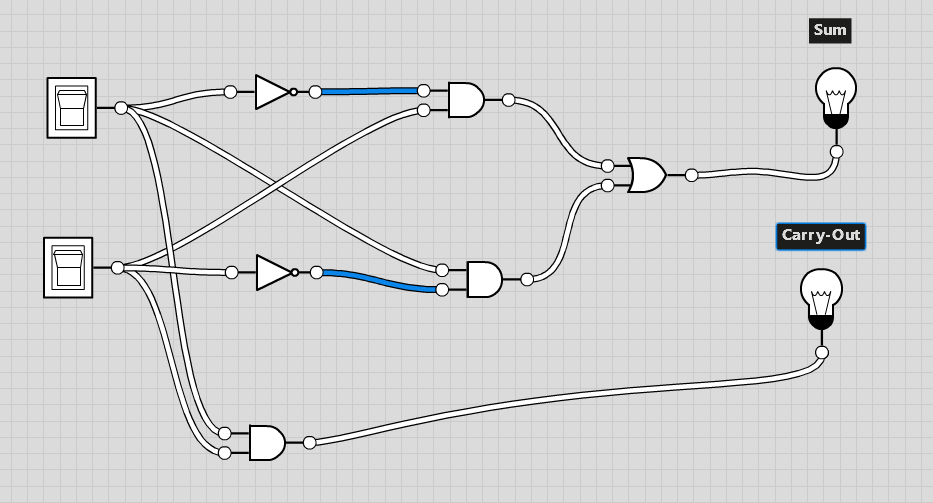
**Step 1**

### In Logicly, using only AND, NOT, and OR gates (do not use any XOR gates), design, build, and test a half-adder circuit. Use the Label component in the Other component section (panel on left, use scroll bar to go to near the bottom) to label your inputs and outputs (see example labels in figure above). Are both outputs (SUM and Carry-out) correct for all inputs?

Put the SOP Boolean expression for both outputs here:

(Not A and B) or (A and Not B)

Paste a screen shot of your circuit here:



**Step 2**

In Logicly, go to File -> Save Document As. Name the file **HALF** and save the file to your Lab folder that you created above.

**Step 3**

Outside your circuit, left click and hold to paint all components. Right click and select ‘Create Integrated Circuit’. In the Full Name box, enter HALF-ADDER. . In the Symbol Label box, enter HA.

For each input and output, click on it and name it. For example, click on one of your light bulbs, you should get a pop-up window that has the words ‘Export Name’ followed by a box. In the box, name the output what it is, i.e., SUM or Carry-out. Name the inputs in the same way. Note that you will need to click on the border of the light switch to get the pop-up window. Otherwise, you will just flick the switch.

Then select ‘Create IC’. You IC will be put in the bottom of the left screen below the symbols. **Once you are done creating ICs, you must export your IC library. Next time you want to use the library or add new ICs to it, import the library first, then you can export it with new ICs. Play with this before you commit to anything to sure you don't lose work. At the end of the lab, you should have a single IC library named IC\_LIB.**

**Step 4**

Clear your Logicly workspace by selecting Edit->Select All and then Edit->Delete

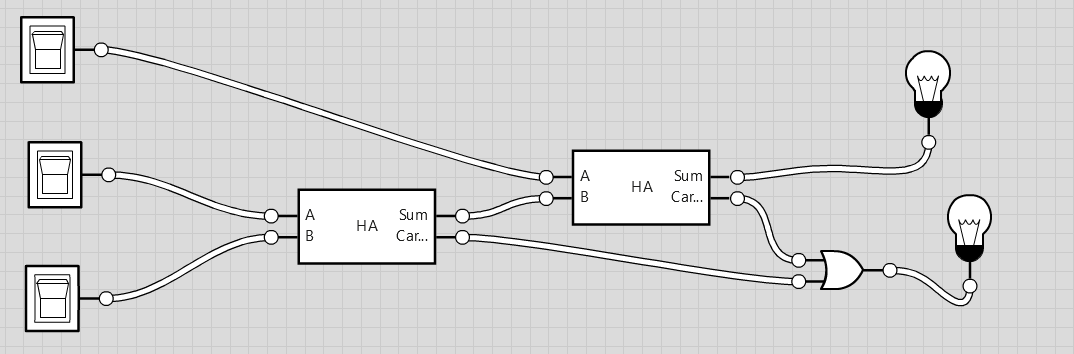
**Part 4: Building a full-adder**

The HALF-ADDER integrated circuit that you created above will appear in the Custom component section in Logicly (panel on left, use scroll bar to go to bottom).

**Step 1**

Review the design of a full-adder given in the book. Using two of the HALF-ADDER integrated circuits that you created above and additional components as necessary, create a full-adder circuit. Use the Label component in the Other component section (panel on left, use scroll bar to go to near the bottom) to label your inputs and outputs (see example labels in figure above). Are both outputs (SUM and Carry-out) correct for all inputs?

Paste a screen shot of your circuit here showing only the IC for the half adders, not the internal circuits for them. I.e., show how the half adders are connected and any additional gates and connections.



**Step 2**

In Logicly, go to File -> Save Document As. Name the file **FULL** and save the file to your Lab folder that you created above.

**Step 3**

As you did above, outside your circuit, left click and hold to paint all components. Right click and select ‘Create Integrated Circuit’. In the Full Name box, enter FULL-ADDER. In the Symbol Label box, enter FA. Then select ‘Create IC’.

**Step 4**

For each input and output, click on it and name it. For example, click on one of your light bulbs, you should get a pop-up window that has the words ‘Export Name’ followed by a box. In the box, name the output what it is, i.e., SUM or Carry-out. Name the inputs in the same way (for example, A, B, and Carry-IN). Note that you will need to click on the border of the light switch to get the pop-up window. Otherwise, you will just flick the switch.

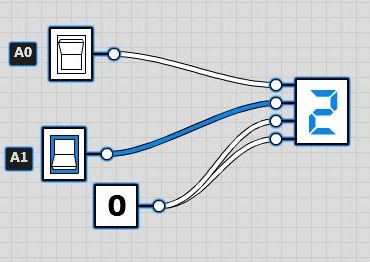
**Step 5**

Clear your Logicly workspace by selecting Edit->Select All and then Edit->Delete

**Part 5: Building a circuit to display, in decimal, the result of adding two, two-bit binary numbers.**

**Step 1**

Using two of the full-adder integrated circuits that you created above and a Digit display component (in the Output Controls component section), build and test a circuit for adding two, two-bit binary numbers (A and B. Thus, you will have the following input bits: A­­­0, A1, B0, and B1, where the zero subscripts indicate the low-order bits). You can think of these as the A and B registers. Use 3 of the Digit display components to display both addends and the sum in decimal. (Note: Before doing this, you should implement the following circuit to familiarize yourself with the Digit display component. Try adding switches instead of the Low (zero) input; see what happens.) The screenshot below is what your two-adder should look like. You have to put in the components and wires. I used two full adders and some zero constants.

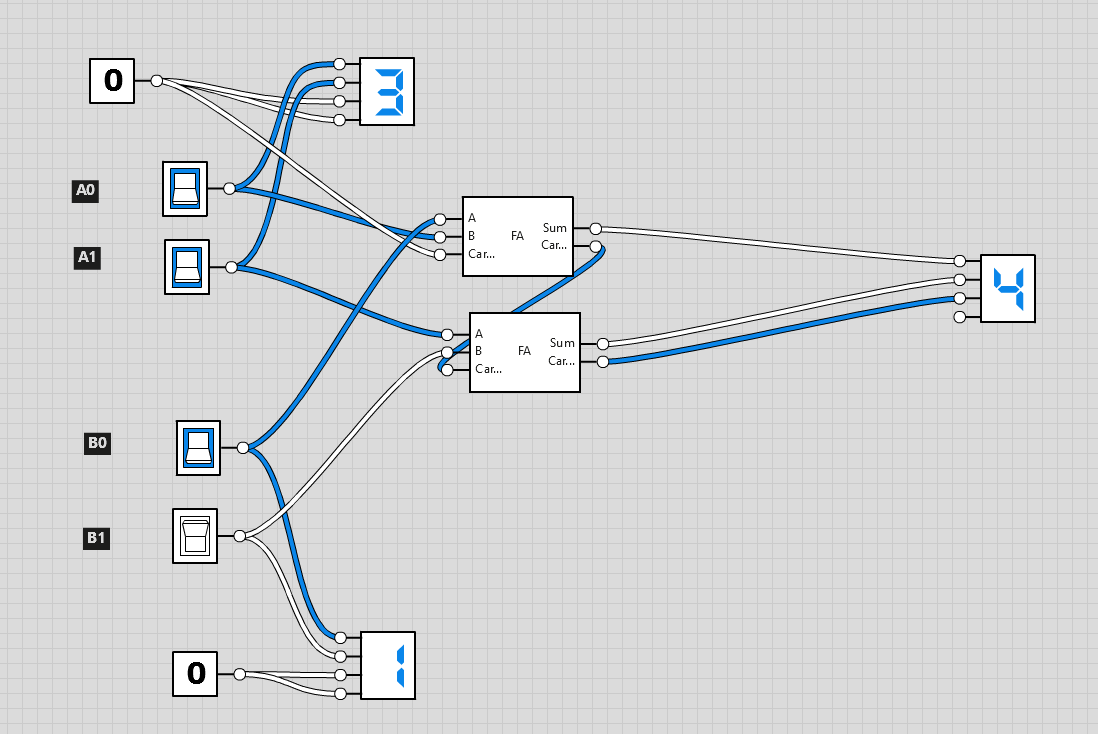


**Step 2**

In Logicly, go to File -> Save Document As. Name the file **TWO** and save the file to your Lab folder that you created above.

**Step 3**

Paste a screenshot of your circuit here like the one above but show the full circuit. Replace my screenshot with yours.

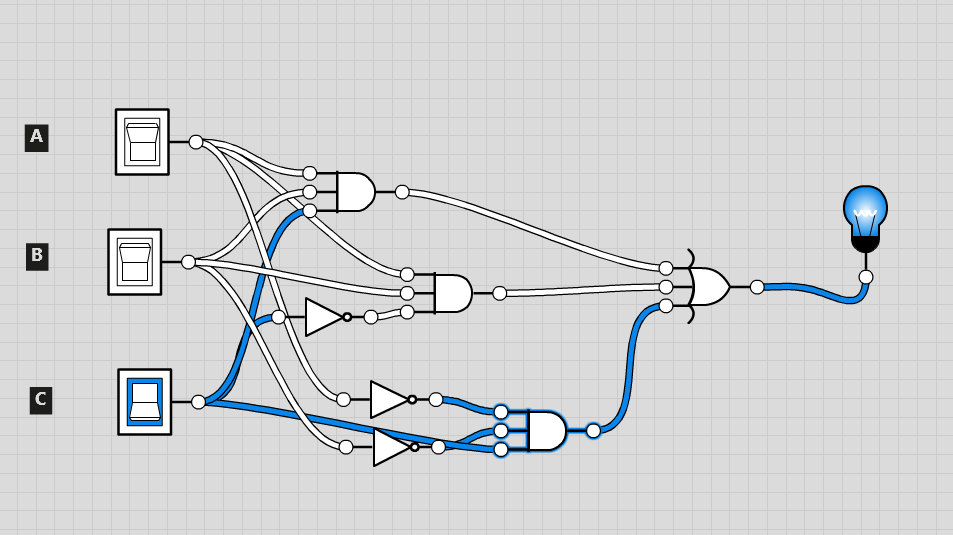


**Part 6. Building a circuit using the Sum-of-Products algorithm.**

**Step 1**

Using Logicly, implement the Sum-of-Products circuit for the following truth table. Do not simplify the circuit even if possible, but show the full SOP form. When you use AND and OR gates, only use the two-input gates. Test your circuit to be sure it is working correctly. In Logicly, go to File -> Save Document As. Name the file **SOP** and save the file to your Lab folder that you created above.

|  |  |  |  |
| --- | --- | --- | --- |
| **Input A** | **Input B** | **Input C** | **Output X** |
| **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **0** |
| **0** | **1** | **1** | **0** |
| **1** | **0** | **0** | **0** |
| **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** |
| **1** | **1** | **1** | **1** |



**Step 2**

This is a 3-input circuit. How many unique circuits can be built with 3 inputs (the answer is a number)? Unique means that the combination of outputs is unique. The above truth table is one such unique circuit.

9

If you have an N-input circuit, how many unique circuits can be built (the answer is a mathematical expression with N in it)?

N^2

**Step 3**

Write the Boolean expression in full SOP form for the truth table. Full SOP form will show the full subexpression for each row whose output is a 1 and then ORs those subexpressions to produce the final output. Using Boolean algebra rules (properties), show that the SOP form can be simplified to the following: (I.e, show each step of applying a rule. State the rule (property) used at each step.)

(C and A’ and B’) or ( A and C and B’) or (A and B and C)

**C(B’ + AB)**

**Part 7. Multiplexor (Mux).**

Multiplexors are everywhere. Check out figures 4.2, B.3.6, and B.8.8 in the text. In this exercise, you will design a mux (we call it a mux because this is cool and what the big girls do). Your mux will be a 4x1. That is, there are 4 inputs and one output. With 4 inputs, you need two selector lines to choose one of the inputs to route to the output. Think of the inputs as the result of some operation and the selectors as an opcode. So, you may be routing a bit from one of the following operations: AND, OR, XOR, NOR or similar. These operations may map to selector bits: 00, 01, 10, 11. See how this works? The input may be one bit of a register, so you would need 32 of these, or however many bits you have in your register.

Study the truth table below for the Mux. See Step 2 for column name meanings. Note how the output is not a fixed zero or 1. Also, there are 6 inputs to the mux but only 4 lines in the truth table. Think about this. There is a screenshot of my circuit below, without the internals, which includes the IC symbol for my mux. Why is the light bulb on with S0 and S1 off?

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A0** | **B0** | **C0** | **D0** | **S1** | **S0** | **R0** |
| **?** | **?** | **?** | **?** | **0** | **0** | **A0** |
| **?** | **?** | **?** | **?** | **0** | **1** | **B0** |
| **?** | **?** | **?** | **?** | **1** | **0** | **C0** |
| **?** | **?** | **?** | **?** | **1** | **1** | **D0** |

**Step 1**

Design and implement a 4x1 mux using Logicly. Label the inputs A0, B0, C0, D0. Label the selector lines as S0, S1. Label the output R0. All inputs are toggle switches, and the output is a light bulb. You may use AND and OR gates with more than two inputs.

**Step 2**

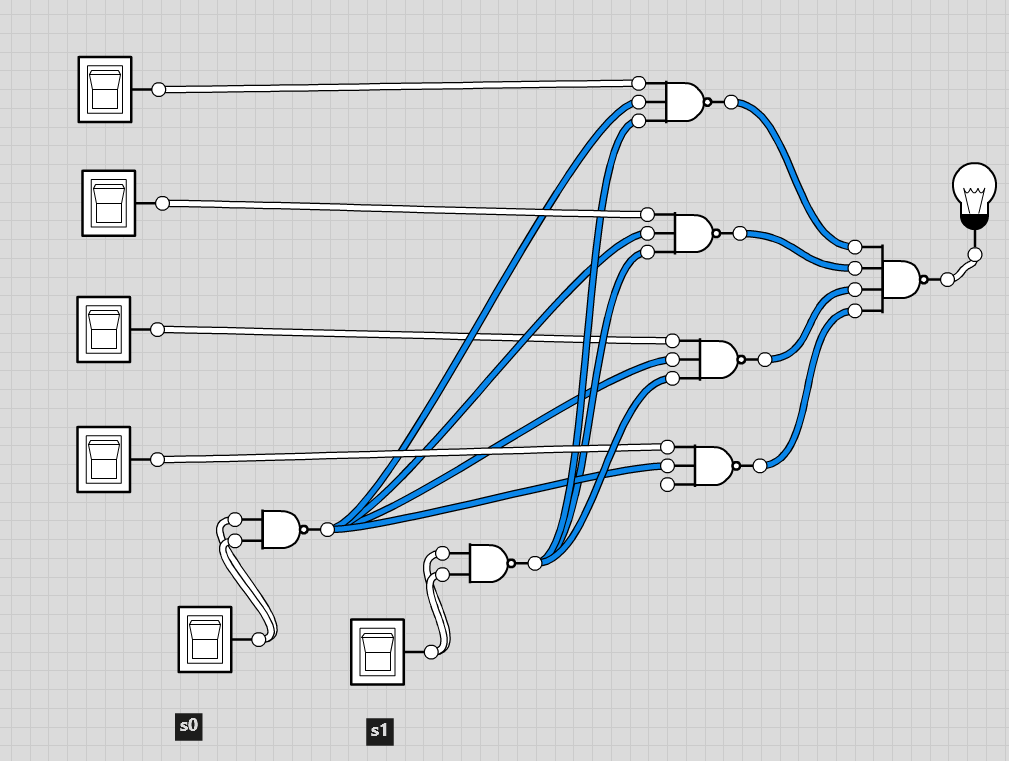
In Logicly, go to File -> Save Document As. Name the file **MUX\_4x1** and save the file to your Lab folder that you created above.

**Step 3**

As you did above, outside your circuit, left click and hold to paint all components. Right click and select ‘Create Integrated Circuit’. In the Full Name box, enter MUX\_4x1. In the Symbol Label box, enter MX4x1. Then select ‘Create IC’.

**Step 4**

Paste a screenshot of your circuit here like the one above but show the full circuit. Replace my screenshot with yours.



**Part 8: 1-bit ALU**

You are now ready to implement a 1-bit ALU. Review, understand, and implement the 1-bit ALU in figure B.5.6. Use your 4x1 mux and your full-adder. Label to match the figure. This is so cool. Below is a screenshot of my ALU without internals. (I included the IC for my final ALU just so you could see what it looks like. It is not part of the circuit.) Note: My mux did not order the inputs exactly as designed (C0 comes before B0), so be careful when you wire this up. The labels are correct, they just are not in the order I expected.

**Step 1**

Implement the 1-bit ALU with two changes. 1) Use the operation 3 (binary 11) for an XOR. 2) The carry out should always be zero unless ADD is selected, in which case the carry out should be whatever is calculated for this bit. Your operations should be: 00: AND; 01: OR; 10: ADD; 11: XOR.

**Step 2**

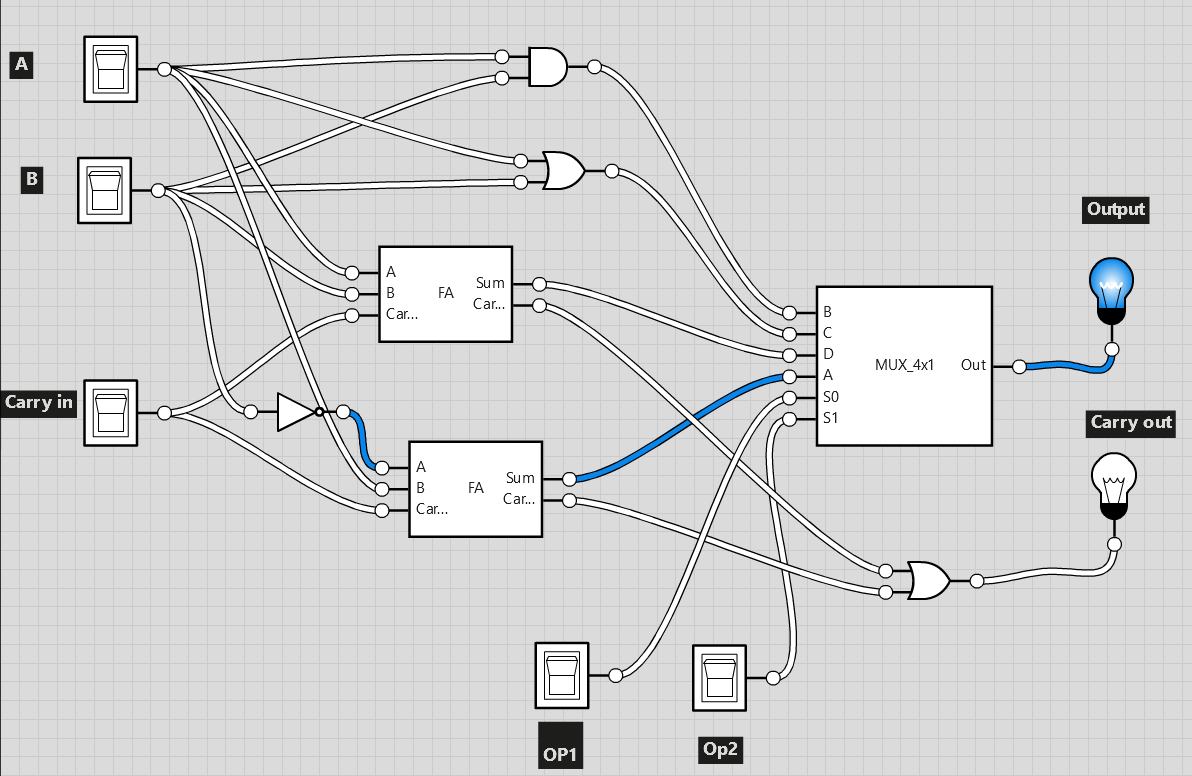
In Logicly, go to File -> Save Document As. Name the file **ALU\_1\_bit** and save the file to your Lab folder that you created above.

**Step 3**

As you did above, outside your circuit, left click and hold to paint all components. Right click and select ‘Create Integrated Circuit’. In the Full Name box, enter ALU\_1\_bit. In the Symbol Label box, enter ALU\_1. Then select ‘Create IC’.

**Step 4**

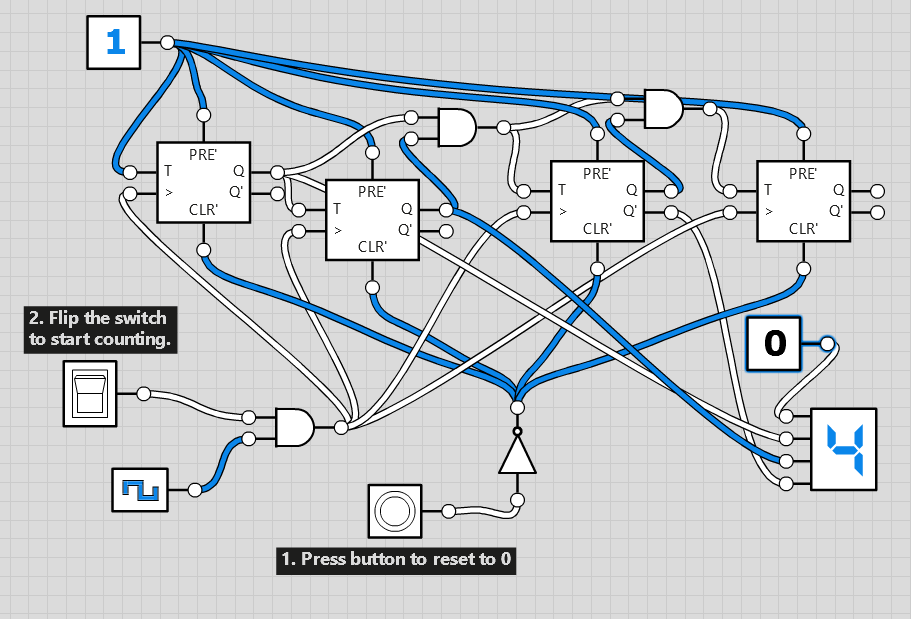
Paste a screenshot of your circuit here like the one above but show the full circuit. Replace my screenshot with yours.



**Part 9: Ripple Counter:**

Download the ripple counter Logicly file from the assignment on Sakai. Modify the circuit to count by two each clock cycle. I.e., LCD should show: 0, 2, 4, 6, 8, A, C, E, 0, 2 … Change the light bulbs so they show the same as the LCD. Hint: you may not need all the components, and you may need to add a component or two - maybe. I.e., it is not that hard but takes some thought. Remember, it should change each clock cycle. Understand how the T flip-flop works. Just another memory element. Do you notice that the output of the T flip-flops are clock pulses but at different frequencies? Save this circuit as RIPPLE\_2.

Paste a screenshot of your circuit here:



You don't have to do this step, just some ideas: Can you change the circuit to count backwards? Can you change to count the odd numbers? You don’t have to do these things, just tossing ideas around.

**Do the following to complete this lab.**

1. Upload your completed Lab Report document (this document) to Sakai.

Name the file: assign\_6\_*lastname*.docx.

2. Upload your HALF, FULL, TWO, SOP, MUX\_4x1, ALU\_1\_bit, and RIPPLE\_2 Logicly files to Sakai. Also upload your IC library. Be sure all files are properly named.