|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 🔑 **Essential** 🔑 | | | |  | 🏆 **Enhanced** 🏆 | | | |
|  |  |  |  |  |  |  |  |  |
| \_\_\_\_\_ | \_\_\_\_\_ | \_\_\_\_\_ | \_\_\_\_\_ |  | \_\_\_\_\_ | \_\_\_\_\_ | \_\_\_\_\_ | \_\_\_\_\_ |

**Score: \_\_\_\_\_**

**Lab03 – Programmable Circuits**

COMP256 – Computing Abstractions

Dickinson College

Spring 2022

Prof. Grant Braught

**Name(s):**

**Introduction:**

In the first lab you built build one of the fundamental units of computation, a logical NOT gate, using transistors. In the second lab you used integrated circuit chips containing logic gates to build a circuit that could perform a simple computation – adding three 1-bit numbers. In this lab, you will build on what you know to create circuits that are “programmable.” These circuits will use a pattern of 1’s and 0’s that tells them what function to perform on their inputs. While this may not quite seem like “programming,” if you think of the pattern of 1’s and 0’s as code for an instruction (e.g. 01 meaning compute OR, or 11 meaning compute a sum) then it may seem more plausible. After all, any of the instructions we write in a language like Java (e.g. z = x + y) must be ultimately translated into 1’s and 0’s that tell the machine what to do. We’ll be learning more about that soon. But for now, you’ll get a little insight into how a pattern of 1’s and 0’s can act as an instruction to a machine.

**A Programmable 1-bit Machine:**

The programmable machine we will be building performs computations on two 1-bit inputs, A and B. The machine will be able to perform four different computations: NOT A, A OR B, A AND B, and A PLUS B. Which operation this computer performs is controlled by a 2-bit instruction. We’ll call the bits of the instruction S1 and S0 so that we can easily refer to them.

The bits of the instruction control the operation that is performed by the machine according to the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Instruction** | |  |  |
|  | S1 | S0 | **Operation** |  |
|  | 0 | 0 | NOT A |  |
|  | 0 | 1 | A OR B |  |
|  | 1 | 0 | A AND B |  |
|  | 1 | 1 | A PLUS B |  |

For example, the instruction 10 will cause the machine to perform the operation A AND B. So if the inputs to the machine are A=0, B=1 then the Instruction 10 (i.e. S1=1 and S0=0) would generate a result of 0 because 0 AND 1 is 0. A few additional notes on these operations:

* When NOT A is computed the input B is ignored.
* When A PLUS B is computed we will only care about the sum, any carry that would occur is ignored.

🔑 1. Fill in the Result column in the following table to show what result would be generated if the indicated instruction is performed on the given inputs.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  | **Instruction** | | **Inputs** | |  |  |
|  | S1 | S0 | A | B | **Result** |  |
|  | 0 | 0 | 1 | 0 |  |  |
|  | 0 | 1 | 1 | 0 |  |  |
|  | 1 | 1 | 1 | 0 |  |  |
|  | 1 | 0 | 1 | 1 |  |  |
|  | 1 | 1 | 1 | 1 |  |  |
|  |  |  |  |  |  |  |

**Making Choices:**

One of the fundamental operations in programmable machines is that of making a choice. For example, we will have to choose NOT, OR, AND or PLUS based on the bits in the instruction. Thus, to create a programmable machine we will need to build circuits that can make choices. The next several questions will lay the foundations for making choices in circuits and then you will construct a circuit that can make a choice.

🔑 2. Consider the following logic circuit:



a. Complete the following truth table for this circuit, including the intermediate variables X and Y and the output R:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  | **C** | **P** | **Q** | **X** | **Y** | **R** |  |
|  | 0 | 0 | 0 |  |  |  |  |
|  | 0 | 0 | 1 |  |  |  |  |
|  | 0 | 1 | 0 |  |  |  |  |
|  | 0 | 1 | 1 |  |  |  |  |
|  | 1 | 0 | 0 |  |  |  |  |
|  | 1 | 0 | 1 |  |  |  |  |
|  | 1 | 1 | 0 |  |  |  |  |
|  | 1 | 1 | 1 |  |  |  |  |
|  |  |  |  |  |  |  |  |

b. You should notice that when C is 0 the output (R) matches whatever P is and conversely that when C is 1 the output (R) matches whatever Q is. If this is not reflected in your table, revisit part a and resolve any errors. **If you are unsure if your table is correct, have your instructor check it at this point.**

c. To make it a little clearer that this circuit is making a choice based on an instruction, think of C as a 1-bit instruction and fill in the blank in each of the sentences below with either P or Q:

An instruction (C) of 0 causes the circuit to choose as its output (R).

An instruction (C) of 1 causes the circuit to choose as its output (R).

Based on the statements you just filled, the instruction (C) is causing this circuit to do different things. An instruction of 0 causes it to choose P, while an instruction of 1 causes it to choose Q. So, in some very real sense, this circuit is programmable!

It is worth mention that the circuit above is very common. It is called a *multiplexer* and there are integrated circuit chips (e.g. a 74HC157 chip) that abstract away the details of the logic gates that implement it. Unfortunately, TINKERCAD does not provide this chip, so we’ll be building it ourselves.

**Choosing Operations:**

We can now build on the above circuit to create a larger circuit that makes a choice between two different computations. We will start with just the first two operations that our computer can perform (NOT A, and A OR B).

🔑 3. Consider the schematic diagram shown below. It builds directly on the circuit from question #2, which is highlighted in the yellow box. This diagram just changes the labels a bit. C is changed to be S0, one of the bits of our instruction. R is changed to be Z0, the result of the computation indicated by the instruction S0.



Now we can build on your answers from question #2c to make it clearer that this circuit is doing a computation. Complete the following sentences using the input variables (A, B) and the operations (NOT, OR).

An instruction (S0) of 0 causes the circuit to compute as its output (Z0).

An instruction (S0) of 1 causes the circuit to compute as its output (Z0).

🔑 4. Create a new circuit in TINKERCAD named Lab03-NotOr. *Your TINKERCAD circuit will be your answer to this question.*

🔑 5. Setup a regular size (not small or mini) breadboard as shown below. Note that I have added labels to the diagram that I’ll be referring to throughout the lab. You do not need to create any labels in your circuit.

*Your TINKERCAD circuit will be your answer to this question.*



🔑 6. Using the components on this breadboard, construct the circuit illustrated by the schematic in question #3.

* Don’t forget to make the Vcc and ground connections to each chip.
* Use the slide switches A, B as the inputs.
* Use the slide switch S0 as the instruction.
* Connect the output of the circuit (Z0) to the right-hand lead of the LED.

References for the pin-out diagrams for all of the integrated circuit chips that you need can be found at the end of this lab.

*Your TINKERCAD circuit will be your answer to this question.*

🔑 7. Using your circuit complete the following table indicating if the LED is on or off for each input combination, the logical value of the output Z0 and a logic expression involving A and B that expresses the function that has been computed. Note that the same function will be listed multiple times because the same instruction is used multiple times (for different inputs A and B).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  | **S0** | **A** | **B** | **LED** | **Z0** | **Function** |  |
|  | 0 | 0 | 0 |  |  |  |  |
|  | 0 | 0 | 1 |  |  |  |  |
|  | 0 | 1 | 0 |  |  |  |  |
|  | 0 | 1 | 1 |  |  |  |  |
|  | 1 | 0 | 0 |  |  |  |  |
|  | 1 | 0 | 1 |  |  |  |  |
|  | 1 | 1 | 0 |  |  |  |  |
|  | 1 | 1 | 1 |  |  |  |  |
|  |  |  |  |  |  |  |  |

🔑 8. Compare the results you have in the above table with your answer to question #3 above. If the results are not consistent, revisit questions #3-7 and resolve any issues. *There is no answer required for this question, just ensure that your earlier answers are correct.* **If you are unsure, ask your instructor at this point.**

**The Other Two Operations:**

Notice that the above circuit handles the first two operations of our programmable 1-bit machine (NOT A and A OR B). You will now build a circuit that handles the second two operations (A AND B and A PLUS B).

🔑 9. To handle the second two operations, we will need to compute A AND B, which is pretty easy using an AND gate. But we also need to compute A PLUS B. It turns out that this is pretty easy as well.

a. Complete the following truth table with the value of A PLUS B. Note: We are ignoring the carry that might result for this lab so (1 PLUS 1 is just 0).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | A | B | A PLUS B |  |
|  | 0 | 0 |  |  |
|  | 0 | 1 |  |  |
|  | 1 | 0 |  |  |
|  | 1 | 1 |  |  |
|  |  |  |  |  |

b. Which logic gate computes A PLUS B as shown in the table above?

🔑 10. Now we can create a circuit that will compute the second two functions.

a. Draw a schematic diagram for a circuit that will use the instruction bit (S0) to choose either A AND B or A PLUS B as its output. Label the output of this circuit Z1 to differentiate it from the earlier output Z0. Hint: Your schematic here should be very similar to the one from question #3.

b. Complete the following sentences using the input variables (A, B) and the operations (AND, PLUS).

An instruction (S0) of 0 causses this circuit to compute as its output (Z1).

An instruction (S0) of 1 causes this circuit to compute as its output (Z1).

🔑 11. Create a copy of your Lab03-NotOr circuit in TINKERCAD and rename it Lab03-AndPlus. *Your TINKERCAD circuit will be your answer to this question.*

🔑 12. Add a second full-size breadboard to your Lab03-AndPlus circuit, leaving the original one unchanged. **Do not add a second battery. Connect the rails of your new breadboard to the same battery as the first.**

On this new breadboard, implement the circuit you drew in question #10.

* Do not add any new slide switches:
  + Use the A and B slide switches from your original bread board as the inputs for this circuit.
  + Use the S0 slide switch from your original bread board as the instruction to for circuit.
* Include an LED in your new circuit to show the output Z1, similar to how the LED in your original circuit showed the output Z0.

*Your TINKERCAD circuit will be your answer to this question.*

🔑 13. Using your new circuit complete the following table indicating if the LED is on or off, the logical value of Z1 for each input combination and a logic expression involving A and B that expresses the function that has been computed. Note that the same function will be listed multiple times because the same instruction is used multiple times (for different inputs A and B).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  | **S0** | **A** | **B** | **LED** | **Z1** | **Function** |  |
|  | 0 | 0 | 0 |  |  |  |  |
|  | 0 | 0 | 1 |  |  |  |  |
|  | 0 | 1 | 0 |  |  |  |  |
|  | 0 | 1 | 1 |  |  |  |  |
|  | 1 | 0 | 0 |  |  |  |  |
|  | 1 | 0 | 1 |  |  |  |  |
|  | 1 | 1 | 0 |  |  |  |  |
|  | 1 | 1 | 1 |  |  |  |  |
|  |  |  |  |  |  |  |  |

🔑 14. Compare the results you have in the above table with your answer to question #10b above. If the results are not consistent, revisit questions #9-13 and resolve any issues. *There is no answer required for this question, just ensure that your earlier answers are correct.* **If you are unsure, ask your instructor at this point.**

**Putting it all Together:**

Now looking at the table of instructions again:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Instruction** | |  | **Circuit** |  |
|  | S1 | S0 | **Operation** | **Output** |  |
|  | 0 | 0 | NOT A | Z0 |  |
|  | 0 | 1 | A OR B |  |
|  | 1 | 0 | A AND B | Z1 |  |
|  | 1 | 1 | A PLUS B |  |

We can see that S0 casues Z0 to be either NOT A (when S0=0) or A OR B (when S0=1). But notice that S0 also causes Z1 to be either A AND B (when S0=0) or A PLUS B (S0=1). This corresponds exactly to the two circuits that you have constructed thus far.

The next step is to choose either the output of your first circuit (Z0) or the output of your second circuit (Z1) as the final result. Notice in the table that an output of Z0 corresponds to S1=0 and that an output of Z1 corresponds to S1=1. Thus, what we need is a circuit that uses S1 to choose either Z0 or Z1. That should sound pretty familiar by now.

🏆 15. Create a copy of your Lab03-AndPlus circuit in TINKERCAD and rename it

Lab03-1BitComputer. *Your TINKERCAD circuit will be your answer to this question.*

🏆 16. Without changing the circuits you already have, add another breadboard (the small is probably fine this time) to your Lab03-1BitComputer circuit. On this breadboard, construct a circuit that chooses between Z0 and Z1 based upon the value of S1. Be sure to include an LED to show the final output, which we will call Z.

* Be sure to connect the rails on your new breadboard to one of the others. Do not add a new battery.
* Add a new slide switch for S1.
* Use the A, B and S0 slide switches on your original breadboard.

*Your TINKERCAD circuit will be your answer to this question.*

🏆 17. Using your circuit complete the following truth table where Z is the final output:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  | **S1** | **S0** | **A** | **B** | **LED** | **Z** | **Function** |  |
|  | 0 | 0 | 0 | 0 |  |  |  |  |
|  | 0 | 0 | 0 | 1 |  |  |  |  |
|  | 0 | 0 | 1 | 0 |  |  |  |  |
|  | 0 | 0 | 1 | 1 |  |  |  |  |
|  | 0 | 1 | 0 | 0 |  |  |  |  |
|  | 0 | 1 | 0 | 1 |  |  |  |  |
|  | 0 | 1 | 1 | 0 |  |  |  |  |
|  | 0 | 1 | 1 | 1 |  |  |  |  |
|  | 1 | 0 | 0 | 0 |  |  |  |  |
|  | 1 | 0 | 0 | 1 |  |  |  |  |
|  | 1 | 0 | 1 | 0 |  |  |  |  |
|  | 1 | 0 | 1 | 1 |  |  |  |  |
|  | 1 | 1 | 0 | 0 |  |  |  |  |
|  | 1 | 1 | 0 | 1 |  |  |  |  |
|  | 1 | 1 | 1 | 0 |  |  |  |  |
|  | 1 | 1 | 1 | 1 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

🏆 18. Double check each of the rows of the above truth table to confirm that your circuit is working correctly. If anything is amiss, revisit questions #15-17 and resolve any issues. *There is no answer required for this question, just ensure that your earlier answers are correct.*

**Some Final Thoughts:**

So, I don’t know about you, but the first time I saw that I thought it was a little like cheating… the instruction (the bits S1S0) didn’t really tell the computer what to compute. Rather, the computer just went ahead and computed everything (NOT A, A OR B, A AND B and A PLUS B) and then the instruction was just used to pick the right one as the output.

While that seems wasteful, it turns out that is in fact the way the CPU in your current machine operates as well. The reason being is that it is less complex to build it that way than to build it so that it only does the desired operation. This is actually the case for lots of things in computing (and we’ll see more later). Further, not only is it less complex, but it is also faster. A good analogy right now is the way that Corona virus vaccines were created. Many of the possible vaccines were being mass produced before we even knew which, if any, of them would work. The idea being that if one or more did work, then since we produced them ahead of time, we only needed to pick the one(s) that did. That way, the vaccines that worked were ready to be distributed much more quickly (like the chosen output), and those we didn’t use were just ignored (like the other outputs).

**Chip Pin-Out Reference:**

A picture containing clock

Description automatically generated

Optional: To help me improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this Lab (include the 2-hour lab period in your total)?

b. Please comment on any particular challenges you faced in completing this Lab.