

CS 2110 Timed Lab 1: Arithmetic Logic Units

Shivani Vora, Isaac Kim, Rohit Rao, Rishi Nopany, Jessica Liu, Bryan Shook, Henry Bui, Andrew Garcl

v1.1

Contents

1	Timed Lab Rules—Please Read	2
1.1	General Rules	2
1.2	Submission Rules	2
1.3	Is collaboration allowed?	2
2	Overview	3
2.1	Tasks and Strategy	3
2.2	Allowed Components	3
3	Instructions	4
3.1	CircuitSim Information	4
3.2	Part A: ALU Components	4
3.3	Part B: Gate Substitution	4
4	Checking your work	5
5	Deliverables	5

1 Timed Lab Rules—Please Read

1.1 General Rules

1. You are allowed to submit this timed lab starting from the moment the assignment is released, until **75 minutes** after your lab period—no more and no less (unless you have accommodations or special circumstances that have already been discussed with your professor). Gradescope submissions will remain open but **you are not allowed to submit after the 75 minute period is over.**
 - You may submit to Gradescope as many times as you wish within your allotted test time period. We will grade your last submission.
 - **Submitting or resubmitting the assignment after this is a violation of the honor code—doing so will automatically incur a zero on the assignment and might be referred to the Office of Student Integrity.**
2. Although you may ask TAs for clarification, you are ultimately responsible for what you submit. **The information provided in this Timed Lab pdf takes precedence.** If in doubt, please make sure to indicate any conflicting information to your TAs.
3. Resources you are allowed to use during the timed lab:
 - Assignment files
 - Previous homework and lab submissions (this includes homework PDFs)
 - Class Notes (Open Net, Open Book)
 - Your mind!
4. Resources you are **NOT** allowed to use:
 - Email/messaging
 - Contact in any form with any other person besides TAs
 - Anything written with the use of AI-based code completion software such as GitHub Copilot or ChatGPT

1.2 Submission Rules

1. Follow the guidelines under the Deliverables section.
2. You are also responsible for ensuring that what you turned in is what you meant to turn in. After submitting you should be sure to download your submission into a brand new folder and test if it works. No excuses if you submit the wrong files, what you turn in is what we grade. In addition, your assignment must be turned in via Gradescope. Under no circumstances whatsoever will we accept any email submission of an assignment. Note: if you were granted an extension you will still turn in the assignment over Gradescope.
3. Do not submit links to files. We will not grade assignments submitted this way as it is easy to change the files after the submission period ends.

1.3 Is collaboration allowed?

Absolutely NOT. No collaboration is allowed for timed labs.

2 Overview

In this timed lab, you will be creating an ALU with four operations. This ALU will take in **two 8-bit inputs** and **one 2-bit op code**. It will output **one 8-bit output**. Additionally, you will be given a circuit that you will be tasked to recreate using a limited set of gates.

2.1 Tasks and Strategy

- Set up your MUX for the ALU output.
- Build your first operation as a subcircuit, and connect it to the MUX.
- Save your work and submit it to Gradescope. This will ensure you get credit for your work even if you do not finish on time.
- Repeat steps 2 and 3 for the other three operations.
- Build your logically equivalent circuit in part B

2.2 Allowed Components

When completing this assignment, you may only use:

1. basic logic gates (AND, OR, NOT, NAND, NOR),
2. decoders,
3. multiplexers,
4. the built-in Circuitsim adders (**NOT** the built-in subtractors),
5. splitters,
6. wires,
7. tunnels,
8. constants,
9. provided input and output pins (**do NOT** add any additional input or output pins to the original/provided subcircuits),
10. subcircuits you create (the circuits tab)

IMPORTANT NOTE 1: YOU DO NOT NEED TO BUILD THE GATES OUT OF TRANSISTORS. PLEASE, FOR YOUR OWN SAKE, DON'T DO IT. USE THE BUILT-IN GATES.

IMPORTANT NOTE 2: After you complete each operation, you may want to save a local copy of your file as a backup. That way, if you have a problem later on, you can revert to the prior working version quickly.

IMPORTANT NOTE 3: You are allowed to and should make use of subcircuits. These subcircuits contain the circuits that you have built in other tabs of your open file. This is extremely useful to keep your ALU circuit clean and will reduce the amount of errors resulting from overlapping wires.

IMPORTANT NOTE 4: You're allowed to use CircuitSim's **default, built-in adders** (in the Arithmetic tab). So please don't try to make your own adders, just use that one. You're not allowed to use anything else from the Arithmetic tab (don't try to use a subtractor; if you need a subtractor, you will need to create your own using the above-mentioned components).

3 Instructions

3.1 CircuitSim Information

For this assignment, you must use the version of CircuitSim provided to you via Canvas. If you cannot access this version of CircuitSim, let a TA know as soon as possible. **If your file does not open in this version of CircuitSim you will receive a 0.**

All changes should be made in the *tl1.sim* file. Do not move or rename any of the input or output pins, and do not add any additional input or output pins.

3.2 Part A: ALU Components

You will create an 8-bit ALU with the following operations, using any of the gates listed above. All numbers should be interpreted as **2's complement**. **Remember:** This ALU has two **8-bit** inputs for A and B and one **2-bit** input for OP, the ALU op-code for the operation in the list below. It has one **8-bit** output named out. These are all provided and labeled correctly, so no need to modify them. If you do modify what is provided, you risk losing compatibility with the autograder!

00. $A / 8$	[Ex. $A = 01001000$ returns 00001001]
01. $(4A - 5B)$	[Ex. $A = 00010000$, $B = 00001010$ returns 00001110]
10. $A * 12$	[Ex. $A = 00000100$ returns 00110000]
11. $\text{isOdd}(B) ? B : A$	[Ex. $A = 00110000$, $B = 01001101$ returns 01001101]

Below are descriptions to elaborate more on the problems above. If you're still having trouble comprehending exactly what the question is asking after reading below, please ask one of your TAs for clarification.

- In the $A / 8$ operation, note that if A is not divisible by 8, round down to the *next lowest integer*. For example, the inputs -7 and -3 should both result in -1 while the inputs 8 and 13 should result in 1. This operation also depends **solely** on the A input.
- For the $\text{isOdd}(B) ? B : A$ operation, you must check if B is an odd number, and if so, you should return B. Else, you should return A. This operation depends solely on the **B being a particular value**.
- You do not have to account for or deal with overflow in any of these problems. Therefore it is possible that given a positive value for A, $A * 12$ results in a negative number due to a lack of bits, this will not cause any issues if the calculation is done correctly.
- The provided autograder will check the ALU op-codes according to the order listed above: $(00)_2 \rightarrow A / 8$, $(01)_2 \rightarrow (4A - 5B)$, $(10)_2 \rightarrow A * 12$, and $(11)_2 \rightarrow \text{isOdd}(B) ? B : A$ and thus it is important that the operations are assigned to the indicated ALU op-codes.

3.3 Part B: Gate Substitution

$AB + A'B + A'B'$ is a sum of products (SOP) expression. A circuit for this expression can be found under the **Provided** circuit tab in *tl1.sim*. You will need to simplify the expression, and create a new simplified circuit under the **student** tab in *tl1.sim*. Due to current supply-chain shortages, CS 2110 students only have access to basic **NAND** gates. These gates cannot be modified by negating their inputs with bubbles. Convert the given circuit to a logically equivalent one that contains only these gates. Use of bubbles to negate the inputs to these gates will result in a zero for this portion of the timed lab.

Hint: you can replace $\text{NOT}(A)$ with $\text{NAND}(A, A)$, i.e. use A as both inputs to a NAND gate.

4 Checking your work

You can run the autograder in Docker by navigating to the directory containing `t11.sim` and running:

```
java -jar t101-tester.jar
```

Note: CircuitSim autograders only show failing test cases and do not provide a results summary at the end. There are roughly 2,048 tests per operation, so the terminal output may get flooded. The final line may state something similar to `[8184 more failures omitted]`. If you want a quick estimate of whether or not your operation passed, just observe how many failures are left. **Upload to gradescope for a more detailed summary of test cases.**

5 Deliverables

Please upload the following files onto the assignment on Gradescope:

1. `t11.sim`