CS 2110 Timed Lab 1: Arithmetic Logic Units

Sophia Imhof, Adrien Cao, Richard Zhang, Eric Stuhr

v1.0

Contents

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

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 simplify to use the **least** amount of transistors possible.

2.1 Tasks and Strategy

- Set up your MUX for the ALU output.
- Build your first operation circuit, and connect it to the MUX. Then submit it to Gradescope.
- Build your second operation circuit and connect it to the MUX. Submit it to Gradescope.
- Rinse and repeat (do operations 3 and 4).
- Build your logically equivalent circuit in part B
- *Hint: 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.*

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. input pins,
10. output pins

IMPORTANT NOTE 1: You are allowed to and should make use of subcircuits. As a reminder, the 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 2: 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 3: 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 will be using CircuitSim. The version is the exact same as the one used in Homework 2 and 3, and can be found on the Docker image provided for this class. To ensure you are on the correct version, check to see if the title bar says "CircuitSim v1.8.4 2110 edition". **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.

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 op-code for the operation in the list above. 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 + B) / 16	`[Ex. A = 00010000, B = 00010000, out = 00000010]`
01. B * -1	`[Ex. B = 01011001, out = 10100111 ]`
10. A % 4	`[Ex. A = 01001110, out = 00000010]`
11. absOf(A - B)	`[Ex. A = 00100000, B = 01000000, out = 00100000]`

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 + B) / 16** operation, note that if A + B is not divisible by 16, round down to the *next lowest integer*. For example, -1/16 and -15/16 should both result in -1 while 19/16 and 31/16 should result in 1.
- Notice that **B * -1** depends solely on the `B` input. **It should NOT rely on `A` being a particular value.**
- For the **A % 4** operation, you need to return the result of A mod 4 (A modulo 4). If you are trying to verify your results, recall how a negative number is modded. Additionally, it is important to note that *all results of modding* are positive. Therefore, think about if your result will be signed or unsigned. Ex. -3 mod 8 = 5 (-3 = -1 * 8 + **5**). This operation also depends solely on the `A` input. **It should NOT rely on `B` being a particular value.**
- For the **absOf(A - B)** you need to return the absolute value of A - B. This means that you should **first** perform the subtraction of A - B and **then** output the absolute value of that result.
- The provided autograder will check the op-codes according to the order listed above: $(\texttt{00})_2 \rightarrow$ (A+B) / 16, $(\texttt{01})_2 \rightarrow$ B * -1, $(\texttt{10})_2 \rightarrow$ A % 4, and $(\texttt{11})_2 \rightarrow$ absOf(A - B) and thus it is important that the operations are in this exact order.

3.3 Part B: Gate Substitution

AB + A'B' is a sum of products (SOP) expression. A circuit for this expression can be found under the `simplification` tab in `tl1.sim`. Due to current supply-chain shortages, CS 2110 students only have access NAND and OR gates. Convert the given circuit to a logically equivalent one that contains only these gates.

4 Checking your work

You can run the autograder in Docker by navigating to the directory containing `tl1.sim` and running:

```
java -jar tl01-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. `tl1.sim`