Project 1 - Digital Logic

Prabhav Gupta, Rahul Narayanan, Alexander Latz, Vy Mai

Fall 2023

Contents

1 Introduction

This project is designed to strengthen your knowledge of basic hardware elements and circuitry. The sections build on each other, so work down the document linearly, otherwise you may not understand what is going on.

Please read through the entire document before starting. Oftentimes things are elaborated on below where they are introduced, so reading the entire document can give you a better grasp on things. **Start early** and if you get stuck, there's always Piazza or come visit us in office hours.

2 Setup

The software you will be using for this project and all future circuit based assignments is called CircuitSim - an interactive circuit simulation package.

CircuitSim is a powerful simulation tool designed for educational use. This gives it the advantage of being a little more forgiving than some of the more commercial simulators. However, it still requires some time and effort to be able to use the program efficiently.

Please follow the instructions in the file "CircuitSim Installation Guide.pdf" in Canvas, under "Files > Course Software > CircuitSim" to install CircuitSim.

All .sim files should be opened in CircuitSim.

3 Part 1 — Introduction to CircuitSim

The first part of this project is designed to get you up to speed with CircuitSim. If you do not have CircuitSim downloaded and running, see the setup section of this document to get there.

3.1 Read Resources

Read through the following resources

- CircuitSim Wires Documentation `https://ra4king.github.io/CircuitSim/docs/wires/`
- Tutorial 0: My First Circuit `https://ra4king.github.io/CircuitSim/tutorial/tut-1-beginner`

3.2 Complete Tutorial 1

In the existing file `tutorial1.sim` and subcircuit `Tutorial 1`, complete the following tutorial. **Do not rename the subcircuit or file**. Be sure to label your two inputs `a` and `b`, and your output as `c`.

Complete Tutorial 1 `https://ra4king.github.io/CircuitSim/tutorial/tut-2-xor`

3.3 Complete Tutorial 2

In the existing file `tutorial2.sim` and subcircuit `Tutorial 2`, complete the following tutorial. **Do not rename the file or subcircuit.** Name the input `in`, and the output `out`.

Complete Tutorial 2 `https://ra4king.github.io/CircuitSim/tutorial/tut-3-tunnels-splitters`

4 Part 2 — ALU

Now that we have the basics down, we can move on to more complex circuits and logic. All computer processors have a very important component known as the Arithmetic Logic Unit (ALU). This component allows the computer to do, as the name suggests, arithmetic and logical operations. For this part, you're going to build an ALU of your own, along with creating some of the gates.

For this part of the project you will:

1. Create the standard logic gates (NAND, NOR, NOT, AND, OR)
2. Create an 8-input multiplexer and an 8-output decoder
3. Create a 1-bit full adder
4. Create an 8-bit full adder using your 1-bit full adder
5. Use your 8-bit full adder and other components to construct an 8-bit ALU

When building these circuits, restrictions have been placed on what you can use. These restrictions are listed at the beginning of each section. **Using anything not listed will result in heavy deductions.** You also need to have everything in its correctly named sub-circuit. More information on the sub-circuits is given below.

Use tunnels where necessary to make your designs more readable, but do not overdo it! For gates, multiplexers, decoders, and adders you can often make clean circuits by placing your components strategically rather than using tunnels everywhere.

Throughout this assignment, do not change/delete any of the input/output pins (You can move them).

4.1 1-Bit Logic Gates

Allowed Components: Wiring Tab and Circuits Tab

All of the circuits are found in the `gates.sim` file.

For this part, you will create a transistor-level implementation of the NAND, NOT, NOR, AND, and OR logic gates.

Remember for this section that you are only allowed to use the components listed in the Wiring section, along with any of the logic gates you are implementing in CircuitSim. For example, once you implement the NOT gate you are free to use that subcircuit in implementing other logic gates. Implementing the gates in the order of the subcircuit tabs can be the easiest option.

Hint: Start by creating the NOT, NAND, and NOR gates from transistors (You may not use subcircuits for these 3 circuits). Then, use these gates as subcircuits for implementing the others.

All of the logic gates must be within their named sub-circuits.

4.2 Muxes

Allowed Components: Wiring Tab, Gates Tab, and Circuits Tab

All of the circuits are found in the `muxes.sim` file.

4.2.1 Decoder

The decoder you will be creating has a single 3-bit selection input (`SEL`), and eight 1-bit outputs (labeled `A`, `B`, `C`, ..., `H`). The decoder uses the `SEL` input to raise a specific output line. `000` should correspond to `A`, `001` should correspond to `B`, etc.

4.2.2 Multiplexer, commonly abbreviated as "mux"

The multiplexer you will be creating has 8 1-bit inputs (labeled appropriately as `A`, `B`, `C`, ..., `H`), a single 3-bit selection input (`SEL`), and one 1-bit output (`OUT`). The multiplexer uses the `SEL` input to choose a specific input line for forwarding to the output. `000` should correspond to `A`, `001` should correspond to `B`, etc.

4.3 Adders

Allowed Components: Wiring Tab, Gates Tab, and Circuits Tab

All of the circuits are found in the `alu.sim` file. Throughout this assignment, you do **not need to account for overflow**.

4.3.1 1-Bit Adder

The full adder has three 1-bit inputs (`A`, `B`, and `CIN`), and two 1-bit outputs (`SUM` and `COUT`). The full adder adds `A + B + CIN` and places the sum in `SUM` and the carry-out in `COUT`.

For example:

```
A = 0, B = 1, CIN = 0 ==> SUM = 1, COUT = 0
A = 1, B = 0, CIN = 1 ==> SUM = 0, COUT = 1
```

Hint: Making a truth table of the inputs will help you.

4.3.2 8-Bit Adder

You should daisy-chain 8 of your 1-bit full adders together in order to make an 8-bit full adder.

This circuit should have two 8-bit inputs (`A` and `B`) for the numbers you're adding, and one 1-bit input for `CIN`. The reason for the `CIN` has to do with using the adder for purposes other than adding the two inputs.

There should be one 8-bit output for `SUM` and one 1-bit output for `COUT`.

4.4 8-Bit ALU

Allowed Components: Wiring Tab, Gates Tab, Plexer Tab, and Circuits Tab

You will create an 8-bit ALU with the following operations. Feel free to use the circuits you made in previous parts of this lab to implement the following operations. They will be under the **Circuits** tab in CircuitSim. You may also create subcircuits for each ALU operation. The inputs to all of these functions can be treated as **2's Complement** binary integers unless explicitly stated otherwise.

000 `add`	`[A + B]`
001 `sub`	`[A - B]`
010 `min`	`[min(A, B)]`

011	`mysteryFunction`	See below
100	`mostSignificantOne`	See below
101	`isMultipleOf32`	`[A % 32 == 0]`
110	`multiplyByNegative7`	`[A * -7]`
111	`isPositive`	`[A > 0]`

Remember, **do NOT account for overflow**.

`isMultipleof32` and `isPositive` should return either `00000000` for `false` or `00000001` for `true`.

`mysteryFunction:` For this operation, you will need to implement a circuit which calculates the result of the following formula for input `A`: $wordsize - \lfloor log_2 A \rfloor - 1$.

Note: For this function, the input can be treated as **unsigned**.

Edge Case: When A = 0, the output of this function should be equal to the wordsize.

Hint: Try applying this formula to the first 16 non-negative integers (0-15) using a wordsize of 4. Look for a pattern.

Examples:

```
A = 00011001 => return 00000011 (3)
A = 00000111 => return 00000101 (5)
```

`mostSignificantOne:` To elaborate, this operation should return the 8-bit representation of the value that is produced when all bits of A are cleared except for the most significant 1.

Examples:

```
A = 01100110 => return 01000000
A = 00001100 => return 00001000
```

Notice that `mysteryFunction`, `mostSignificantOne`, `isMultipleof32`, `multiplyByNegative7`, and `isPositive` only operate on the `A` input. They should **NOT** rely on `B` being a particular value.

This ALU has two 8-bit inputs (`A` and `B`) and one 3-bit input for `OP`, which is the op-code for the operation in the list above. It has one 8-bit output named `OUT`.

The provided autograder will check the op-codes according to the order listed above (add (`000`), sub (`001`), etc.) and thus it is important that the operations are in this exact order. **If you ever need a constant value, use a constant pin (not an input pin). Failing to do so will interfere with the autograder.**

No partial credit will be given for incorrect outputs for logic gates, plexers, or adders. However, for the ALU, partial credit will be awarded on a per-operation basis, wherein each operation must perform successfully to be awarded credit. Because of this, we urge you to check your score before the due date.

As always, do not change/delete any of the input/output pins (You can move them).

5 Part 3 — Advanced Combinational Logic

All of the circuits are found in the `project.sim` file.

For this part of the assignment, you are tasked with creating a **sum of products circuit** such that its behavior matches the expected behavior outlined by the truth table found in the `truthtable_kmap.xlsx` spreadsheet. In order to do so, you should use the techniques taught in class to create and reduce boolean expression (see 5.1 below).

As always, do not change/delete any of the input/output pins (You can move them).

5.1 Karnaugh Maps

While it is not a deliverable, making 2 K-maps and reducing boolean expressions will make this circuit significantly easier to design. To aid this, we have provided a template for both K-maps below and in the `truthtable_kmap.xlsx` spreadsheet:

A	$B'C'$	$B'C$	BC	BC'
$D'E'$				
$D'E$				
DE				
DE'				

A'	$B'C'$	$B'C$	BC	BC'
$D'E'$				
$D'E$				
DE				
DE'				

Once you fill out each K-map individually and make groups, you can stack the two K-maps on top of each other to create a 5-variable K-map. Then, you may group any overlapping groups together into one big group and get rid of the `A` variable in the resulting production.

Note: The `truthtable_kmap.xlsx` spreadsheet is NOT a deliverable. It only contains the truthtable that you need to use and sample K-Map templates.

5.2 Boolean Logic Details

For this part of the project, we are asking that you approach this problem in **sum of products format**. This means that after your reductions using the K-maps, you should have something in the format

$$OUT = A'B + C$$

before attempting to build the circuit. Failure to do so can lead to complications in your circuit that prevent the reduction we are looking for.

5.3 Circuit Details

To prevent trivialization of this assignment, we have placed a few restrictions:

- All of this circuit should be contained in the **project.sim** file
- You should try to reduce the amount of AND and OR gates as much as possible.
  Hint: The K-maps do this for you!

6 Autograder

To run the autograder, run

```
java -jar project1-tester-1.0.jar
```

at a command prompt in the same directory as all your `.sim` files. This is the same autograder that Gradescope uses, but is much easier and faster to use. **The autograder should be run from the terminal.**

7 Deliverables

Please submit the follow files:

1. `tutorial1.sim`
2. `tutorial2.sim`
3. `gates.sim` — `NOT, NAND, NOR, AND, OR`
4. `muxes.sim` — `Decoder, MUX`
5. `alu.sim` — `1-Bit Adder, 8-Bit Adder, 8-Bit ALU`
6. `project.sim` — `Advanced Combinational Logic`

to Gradescope under the assignment "Project 1 - Digital Logic".

Note: The autograder may not reflect your final grade on this assignment. We reserve the right to update the autograder as we see fit when grading.

8 Rules and Regulations

8.1 General Rules

1. Although you may ask TAs for clarification, you are ultimately responsible for what you submit.
2. Please read the assignment in its entirety before asking questions.
3. Please start assignments early, and ask for help early. Do not email us a few hours before the assignment is due with questions.
4. If you find any problems with the assignment, it would be greatly appreciated if you reported them to the author (which can be found at the top of the assignment). Announcements will be posted if the assignment changes.

8.2 Submission Conventions

1. In order to submit your assignment, submit the files individually to the Gradescope assignment.
2. Do not submit links to files. The autograder does not understand it, and we will not manually grade assignments submitted this way as it is easy to change the files after the submission period ends.

8.3 Submission Guidelines

1. You are responsible for turning in assignments on time. This includes allowing for unforeseen circumstances. If you have an emergency let us know **IN ADVANCE** of the due time supplying documentation (i.e. note from the dean, doctor's note, etc). Extensions will only be granted to those who contact us in advance of the deadline and no extensions will be made after the due date.
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 we will accept any email submission of an assignment. Note: if you were granted an extension you will still turn in the assignment over Gradescope.

3. Projects turned in late receive partial credit within the first 48 hours, as defined in the syllabus. Between 0 and 24 hours late, you can receive a maximum score of 70%. Between 24 and 48 hours late, you can receive a maximum score of 50%. We will not accept projects turned in over 48 hours late.

4. You alone are responsible for submitting your project before the assignment is due; neither Canvas/Gradescope, nor your flaky internet are to blame if you are unable to submit because you banked on your computer working up until the deadline.

8.4 Is collaboration allowed?

From the syllabus:

- You must submit an assignment or project as your own work. **No collaboration on answers is permitted. Absolutely no code or answers may be copied from others. Such copying is Academic Misconduct.**
- Using code from GitHub, via Googling, from Stack Overflow, etc., is Academic Misconduct (Honor Code: Academic Misconduct includes "submission of material that is wholly or substantially identical to that created or published by another person or persons").
- Publishing your assignments on public repositories, github, etc, that is accessible to other students is unauthorized collaboration and thus Academic Misconduct.

8.5 Syllabus Excerpt on Academic Misconduct

The goal of all assignments in CS 2110 is for you to learn. Learning requires thought and hard work. Copying answers thus prevents learning. More importantly, it gives an unfair advantage over students who do the work and follow the rules.

1. **As a Georgia Tech student, you have read and agreed to the Georgia Tech Academic Honor Code.** The Academic Honor Code defines Academic Misconduct as "***any act that does or could improperly distort Student grades or other Student academic records.***"
2. You must submit an assignment or project as your own work. **No collaboration on answers is permitted. Absolutely no code or answers may be copied from others. Such copying is Academic Misconduct**.
3. Using code from GitHub, via Googling, from Stack Overflow, etc., is Academic Misconduct (Honor Code: Academic Misconduct includes "*submission of material that is wholly or substantially identical to that created or published by another person or persons*").
4. Publishing your assignments on public repositories, github, etc, that is accessible to other students is unauthorized collaboration and thus Academic Misconduct.
5. Suspected Academic Misconduct will be reported to the Division of Student Life Office of Student Integrity. It will be prosecuted to the full extent of Institute policies.
6. Students suspected of Academic Misconduct are informed **at the end of the semester**. Suspects receive an *Incomplete* final grade until the issue is resolved.
7. We use accepted forensic techniques to determine whether there is copying of a coding assignment.
8. Submitting an assignment with code or text from an AI assistant (e.g., ChatGPT) is academic misconduct.
9. **If you are not sure about any aspect of this policy, please ask your lecturer**.

Using AI Assistants

Anything you did not write in your assignment will be treated as an academic misconduct case. If you are unsure where the line is between collaborating with AI and copying AI, we recommend the following heuristics:

Heuristic 1: Never hit "Copy" within your conversation with an AI assistant. You can copy your own work into your own conversation, but do not copy anything from the conversation back into your assignment. Instead, use your interaction with the AI assistant as a learning experience, then let your assignment reflect your improved understanding.

Heuristic 2: Do not have your assignment and the AI agent open at the same time. Similar to the above, use your conversation with the AI as a learning experience, then close the interaction down, open your assignment, and let your assignment reflect your revised knowledge. This heuristic includes avoiding using AI directly integrated into your composition environment: just as you should not let a classmate write content or code directly into your submission, so also you should avoid using tools that directly add content to your submission.

Deviating from these heuristics does not automatically qualify as academic misconduct; however, following these heuristics essentially guarantees your collaboration will not cross the line into misconduct.