Project 1 - Hardware Basics

Todd Hayes, Christopher Turko, Aarsh Patel, Ammar Ratnani, Zilong Huang, and Jason Ng

Fall 2021

Contents

1 Introduction

This project is designed to strengthen your knowledge of basic hardware elements and circuitry. The assignment is designed to 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. Often times 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.

In order to use CircuitSim, you must have Docker up and running. If you do not have Docker, follow the instructions laid out in the installation guide found under Canvas → *Files* → *Docker*.

CircuitSim comes pre-installed on the Docker image, and should be accessible via the desktop. Please only use the CircuitSim through Docker to build your circuits as it is the correct version. **CircuitSim downloaded elsewhere may not be compatible with our grader. You have been warned.**

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.

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 running through Docker, 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.

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 NAND and NOT gates from transistors. Then use this gate as a subcircuit 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.

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.

000	`add`	`[A + B]`
001	`sub`	`[A - B]`
010	`max`	`[max(A, B)]`
011	`popCounter`	`See below`
100	`mostSignificantOne`	`See below`
101	`isMultipleOf16`	`[A % 16 == 0]`
110	`multiplyBy5`	`[A * 5]`
111	`isNegative`	`[A < 0]`

`isMultipleof16` and `isNegative` should return either `00000000` for `false` or `00000001` for `true`

`popCounter:` For this operation, you will need to implement a population counter. At it's core, a population counter is a circuit that counts the number of 1s in a given bit vector using only combinational logic. For the population counter in this project, you will need to create an circuit that will count the number of 1s in input A.

Examples:

```
A = 00101111 => return 00000101
A = 11110000 => return 00000100
```

`mostSignificantOne:` To elaborate, this operation should return the 8-bit representation of the value that is produced when all bits of C are cleared except for the most significant 1. Keep in mind that C is given in a 4-bit representation.

Examples:

```
C = 0110 => return 00000100
C = 1100 => return 00001000
```

Notice that `popCounter`, `mostSignificantOne`, `isMultipleOf16`, `multiplyBy5`, and `isNegative` only operate on the `A` or `C` input. They should **NOT** rely on `B` being a particular value.

This ALU has two 8-bit inputs for `A` and `B`, one 4-bit input for `C`, 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.

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.

5 Part 3 — Advanced Combinational Logic

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

(Note: The following does not represent an accurate Project 1 experience)

You are in an alternate universe where you are a procrastinator and have just begun working on your first project for CS 2110. It is currently 8 AM and the project is due at midnight, meaning you have 16 hours left to complete your project. From talking to your friends you have a rough idea about how long each part will take. You would like to know what possible combinations of parts you will be able to complete before the deadline and what better tool to use than combinational logic to figure it out! There are five parts to the project: Gates, Muxes, Decoders, Adders, and the ALU. Each part takes the following amount of time:

G	$Gates$	$3hrs$
M	$Muxes$	$4hrs$
D	$Decoders$	$7hrs$
R	$Adders$	$5hrs$
A	ALU	$13hrs$

(Note: These do not represent the actual amount of time it will take to finish Project 1)

Your goal is to create a circuit which can tell you whether a given combination of parts can be completed within the 16 hours. An input value of 1 indicates that you are trying to complete the part and that it will take the time specified in the table. A input value of 0 indicates that the part will not be attempted and will take 0 hrs. An output value of 1 indicates the given combination of parts can be completed in the 16 hours and an output value of 0 indicates the given combination of parts **cannot** be completed in 16 hours.

Ex: If $\mathbf{G = 1}$, $\mathbf{M = 0}$, $\mathbf{D = 0}$, $\mathbf{R = 0}$, and $\mathbf{A = 1}$ then the output should equal 1 since Gates and the ALU will take 16 hours.

As always, do not change/delete any of the input/output pins.

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:

A	$M'G'$	$M'G$	MG	MG'
$D'R'$				
$D'R$				
DR				
DR'				

A'	$M'G'$	$M'G$	MG	MG'
$D'R'$				
$D'R$				
DR				
DR'				

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

$$D = A'B + C$$

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.

7 Deliverables

Please submit the follow files:

1.	`tutorial1.sim`	
2.	`tutorial2.sim`	`umbrella`
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`
7.	`collaborators.txt`	

to Gradescope under the assignment "Project 1".

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 Demos

This project will be demoed. Demos are designed to make sure that you understand the content of the project and related topics. They may include technical and/or conceptual questions.

- Sign up for a demo time slot via Canvas **before** the beginning of the first demo slot. This is the only way you can ensure you will have a slot.
- If you cannot attend any of the predetermined demo time slots, e-mail the Head TA **before** the beginning of the first demo slot.
- If you know you are going to miss your demo, you can cancel your slot on Canvas with no penalty. However, you are **not** guaranteed another time slot. You cannot cancel your demo within 24 hours or else it will be counted as a missed demo.
- Your overall project score will be `((project_score * 0.5) + (demo_score * 0.5))`, meaning if you received a 90% on your project, but a 30% on the demo you would receive an overall score of 60%. **If you miss your demo you will not receive any of these points and the maximum you can receive on the project is 50%.**
- You will be able to makeup one of your demos at the end of the semester for half credit.

9 Rules and Regulations

9.1 General Rules

1. Although you may ask TAs for clarification, you are ultimately responsible for what you submit. This means that (in the case of demos) you should come prepared to explain to the TA how any piece of code you submitted works, even if you copied it from the book or read about it on the internet.
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.

9.2 Submission Conventions

1. When preparing your submission you may either submit the files individually to Canvas/Gradescope or you may submit an archive (zip or tar.gz only please) of the files. Both ways (uploading raw files or an archive) are exactly equivalent, so choose whichever is most convenient for you.
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.

9.3 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 Canvas/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 Canvas/Gradescope.
3. You are additionally responsible for ensuring that the collaborators list you have provided in your submission is accurate.
4. Projects turned in late receive partial credit within the first 48 hours. We will take off 30% of the points for a project submitted between 0 and 24 hours late, and we will take off 50% of the points for a project submitted between 24 and 48 hours late. We will not accept projects turned in over 48 hours late. This late policy is also in the syllabus.
5. 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.

9.4 Syllabus Excerpt on Academic Misconduct

Academic misconduct is taken very seriously in this class.

1. Students are expected to have read and agreed to the Georgia Tech Honor Code, see http://osi.gatech.edu/content/honor-code.
2. Suspected plagiarism will be reported to the Division of Student Life office. It will be prosecuted to the full extent of Institute policies.
3. A student must submit an assignment or project as his/her own work (this is what is expected of the students).

4. Using code from GitHub, via Googling, from Stack Overflow, etc., is plagiarism and is not permitted. Do not publish your assignments on public repositories (i.e., accessible to other students). This is also a punishable offense.

5. Although discussion among the students through piazza and other means are encouraged, the sharing of work is plagiarism. If you are not sure about it, please ask a TA or stop by the instructor's office during the office hours.

6. You must list any student with whom you have collaborated in your submission. Failure to list collaborators is a punishable offense.

7. TAs and Instructor determine whether the project is plagiarized. Trust us, it is really easy to determine this....

9.5 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").
- Publishing your assignments on public repositories accessible to other students is unauthorized collaboration and thus Academic Misconduct.

Any of your peers with whom you collaborate in a conceptual manner with must be properly added to a **collaborators.txt** file. Collaborating with another student without listing that student as a collaborator is considered plagiarism.