CS 2110 Spring 2016

Homework 3

Created By Dan Shen

Rules and Regulations

General Rules

1. Starting with the assembly homeworks, Any code you write (if any) must be clearly commented and the comments must be meaningful. You should comment your code in terms of the algorithm you are implementing we all know what the line of code does.
2. 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.
3. Please read the assignment in its entirety before asking questions.
4. Please start assignments early, and ask for help early. Do not email us the night the assignment is due with questions.
5. 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.

Submission Conventions

1. All files you submit for assignments in this course should have your name at the top of the file as a comment for any source code file, and somewhere in the file, near the top, for other files unless otherwise noted.
2. When preparing your submission you may either submit the files individually to T-Square or you may submit an archive (zip or tar.gz only please) of the files (preferred). You can create an archive by right clicking on files and selecting the appropriate compress option on your system.
3. If you choose to submit an archive please don't zip up a folder with the files, only submit an archive of the files we want. (See **Deliverables**).
4. Do not submit compiled files that is .class files for Java code and .o files for C code. Only submit the files we ask for in the assignment.
5. 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.

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 T-Square. When you submit the assignment you should get an email from T-Square telling you that you submitted the assignment. If you do not get this email that means that you did not complete the submission process correctly. 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 T-Square.

3. There is a 6-hour grace period added to all assignments. You may submit your assignment without penalty up until 11:55PM, or with 25% penalty up until 5:55AM. *So what you should take from this is not to start assignments on the last day and plan to submit right at 11:54AM.* You alone are responsible for submitting your homework before the grace period begins or ends; neither T-Square, nor your flaky internet are to blame if you are unable to submit because you banked on your computer working up until 11:54PM. The penalty for submitting during the grace period (25%) or after (no credit) is non-negotiable.

Syllabus Excerpt on Academic Misconduct

Academic misconduct is taken very seriously in this class.
Quizzes, timed labs and the final examination are individual work.

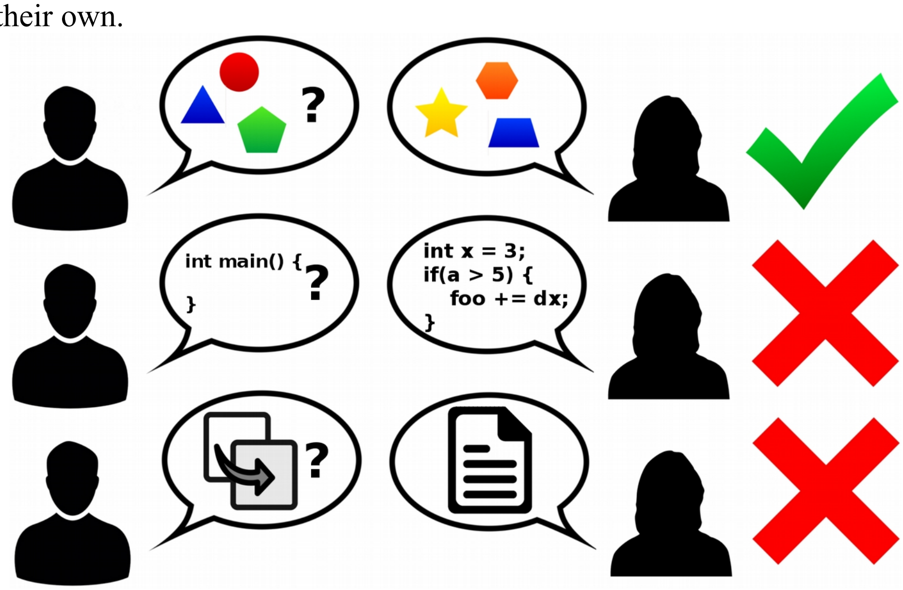
Homework assignments are collaborative, In addition many if not all homework assignments will be evaluated via demo or code review. During this evaluation, you will be expected to be able to explain every aspect of your submission. Homework assignments will also be examined using electronic computer programs to find evidence of unauthorized collaboration.

What is unauthorized collaboration? Each individual programming assignment should be coded by you. You may work with others, but each student should be turning in their own version of the assignment. Submissions that are essentially identical will receive a zero and will be sent to the Dean of Students' Office of Academic Integrity. Submissions that are copies that have been superficially modified to conceal that they are copies are also considered unauthorized collaboration.

You are expressly forbidden to supply a copy of your homework to another student via electronic means. This includes simply e-mailing it to them so they can look at it. If you supply an electronic copy of your homework to another student and they are charged with copying, you will also be charged. This includes storing your code on any site which would allow other parties to obtain your code such as but not limited to public repositories (Github), pastebin, etc. If you would like to use version control, use github.gatech.edu

Is collaboration allowed?

Collaboration is allowed on a high level, meaning that you may discuss design points and concepts relevant to the homework with your peers, as well as help each other debug code. What you shouldn't be doing, however, is paired programming where you collaborate with each other on a low level. Furthermore, sending an electronic copy of your homework to another student for them to look at and figure out what is wrong with their code is not an acceptable way to help them, and it is often the case that the recipient will simply modify the code and submit it as their own.



Note About Logisim version

You must use Brandonsim 2.7.4 for this homework, as well as timed labs. Despite the warning given in the HW01 PDF, **13 people submitted HW01 having used old versions of Logisim**. Deductions weren't made for HW01 submissions this time, but for HW03:

If you use a different version of Logisim you will be deducted 5 points from HW03!

Yes, that means that if you are retaking this class, you must download the version of Logisim that was distributed this semester with HW01 on Canvas, and use that to complete this homework. You have been warned!

Objectives

1. To understand digital logic
2. To use logic gates to perform various operations
3. To learn how to use sub-circuits

Overview

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 assignment, you're going to build an ALU of your own.

DO NOT USE TRANSISTORS!

1. Create a 1-bit full adder
2. Create a 4-bit full adder using the 1-bit full adder
3. Use your 4-bit full adder and other components to construct a 4-bit ALU
4. Create a 16-bit ALU (highly recommended that you create a 16-bit adder to assist, by using the 4-bit full adder)

This assignment will be demoed. More information on this and the sign-up schedule will be posted on the Canvas Sign-Up tool. An announcement will be sent out and it will also be announced in Lecture/Lab when the schedule is up. You have to be present for the demo in order to get credit for this assignment.

Requirements

You may use anything from the Base and Wiring sections, basic gates (AND, OR, XOR, NOT, NAND, NOR, XNOR), multiplexers, and decoders. Use of anything not listed above will result in heavy deductions. **Your designs for the first three problems must each be a sub-circuit**.

More information on sub-circuits is given below

Use tunnels where necessary to make your designs more readable

Sub-circuit tutorial

As you build circuits that are more and more sophisticated, you will want to build smaller circuits that you can use multiple times within larger circuits. In Logisim, this is called a sub-circuit. Sub-circuits behave like classes in Object-Oriented languages. Any changes made in the design of a sub-circuit are automatically reflected wherever it is used. The direction of the IO pins in the sub-circuit correspond to their locations on the representation of the sub-circuit.

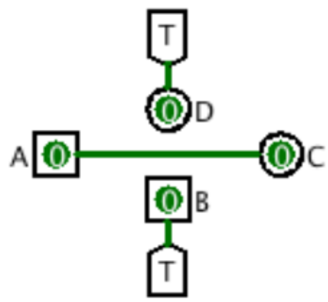


Fig 1. Sub-circuit SC

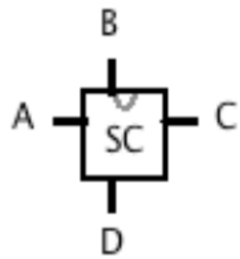


Fig 2. Sub-circuit SC used in another circuit

To create a sub-circuit:

1. Go to the “Project” menu and choose “Add Circuit…”
2. Name your sub-circuit

To use a sub-circuit:

1. Click the sub-circuit you want to use from the sidebar.
2. Place it in your design.

To set a sub-circuit as the main circuit:

1. Right-click the sub-circuit and choose "Set As Main Circuit".

Part 1: 1-bit Full Adder

The full adder has three 1-bit inputs (A, B, and CarryIn), and two 1-bit outputs (Answer and CarryOut). The full adder adds A+B+CarryIn and places the answer in Answer and the carry-out in CarryOut.

For example:
A = 0, B = 1, CarryIn = 0 then Answer = 1, CarryOut = 0
A = 1, B = 0, CarryIn = 1 then Answer = 0, CarryOut = 1
A = 1, B = 1, CarryIn = 1 then Answer = 1, CarryOut = 1

Hint: making a truth table of the inputs will help you

Make your 1-bit full adder a sub-circuit. You will use it in Part 2.

Part 2: 4-bit Full Adder

For this part of the assignment, you will daisy-chain together 4 of your 1-bit full adders together in order to make a 4-bit full adder.

This circuit should have two 4-bit inputs (A and B) for the numbers you're adding, and one 1-bit input for CarryIn. The reason for the CarryIn has to do with using the adder for purposes other than adding the two inputs. You'll see this when you do part 4.

There should be one 4-bit output for the answer and one 1-bit output for CarryOut.

Make your 4-bit full adder a sub-circuit; you will use it in Part 3.

Part 3: 4-bit ALU

Using your 4-bit full adder you will create a 4-bit ALU with the following operations (same as before):

1. Addition [A + B]
2. Subtraction [A – B]
3. isOddNumber [A%2 == 1]
4. 2's complement Negation [-A]
5. Multiply by 2 [A * 2]
6. AND [A & B]
7. OR [A | B]
8. XOR [A ^ B]

Notice that isOddNumber, 2's compliment Negation, and Multiply by 2 only operate on the A input. They should NOT rely on B being a particular value.

Note that you are doing 2's complement negation, NOT simply flipping the bits.

Disregard any carry-over that may result by multiplying by 2.

This ALU has two **4-bit** inputs for A and B and three 1-bit inputs for S0, S1, and S2 (the selectors for the op-code of your ALU's functions)

This ALU should have one **4-bit** output for the answer.

You may assign the op-codes to the operations any way that you want as long as you implement every operation and each op-code only corresponds to one operation.

Add a label to your circuit that lists which operation each op-code corresponds to.

Part 4: 16-bit ALU

With this part you will need to make a helper subcircuit to assist you. Daisy-chain your 4-bit adders into a 16-bit adder then use that to build your 16-bit ALU. You will make the following operations (same as before):

1. Addition	[A + B]
2. Subtraction	[A – B]
3. isOddNumber	[A%2 == 1]
4. 2's complement Negation	[-A]
5. Multiply by 2	[A * 2]
6. AND	[A & B]
7. OR	[A | B]
8. XOR	[A ^ B]

Notice that isOddNumber, 2's compliment Negation, and Multiply by 2 only operate on the A input. They should NOT rely on B being a particular value.

Again, note that you are doing 2's complement negation, NOT simply flipping the bits.

Disregard any carry-over that may result by multiplying by 2.

This ALU has two **16-bit** inputs for A and B and three 1-bit inputs for S0, S1, and S2 (the selectors for the op-code of your ALU's functions)

This ALU should have one **16-bit** output for the answer.

You may assign the op-codes to the operations any way that you want as long as you implement every operation and each op-code only corresponds to one operation.

Add a label to your circuit that lists which operation each op-code corresponds to.

Set this sub-circuit as the main circuit.

Deliverables

Save the file as hw3.circ and turn it in through Canvas

Once again, your designs for the four problems must be contained in the same .circ file as subcircuits

You may also include a README file if there is anything you wish your grading TA to know about your designs. This would be a good place to discuss your choice of op-codes or other concerns.

Once again, this assignment will be demoed! More information on this and the sign-up schedule will be posted on the Canvas Sign-Up tool. An announcement will be sent out and it will also be announce in Lecture/Lab when the schedule is up. You have to be present for the demo in order to get credit for this assignment.