CSCE-312 | Project 2

Combinational chips and the ALU (Heart of the CPU)

Project Submission: 100 points

Grading

Project Submission [100%]:
You will be graded for correctness of the chips you have designed. Grader will run tests on all the HDL codes downloaded from your Canvas submission using Nand2tetris software (Hardware Simulator). So, please make sure to test and verify your codes before finally submitting on Canvas.

Rubric:

(a) **For chips where the prebuilt test collateral (.tst and .cmp) is provided to you**: ***Each chip must pass all its test cases to get full credit,*** else you will receive a **0 point** on that chip.

(b) **For chips where you are constructing the test collateral:** We will test the chips with our own test collateral. **Partial credit will be provided for chips that do not pass the entire suite of tests**.

Deliverables & Submission

You are required to turn in **the completed HDL files for all the chips** implemented. This includes the 3 chips required as part of Project 2 and any other chips that you needed to implement to complete Project 2. You do **NOT** need to turn in your TST or CMP files.

Put your **full name** in the introductory comment present in each HDL code. Use relevant code comments and indentation to improve readability for your benefit. Zip all the required files into a compressed file ***FirstName-LastName-UIN.zip*** . **Submit this zip file on Canvas.**

Late Submission Policy: Refer to the Syllabus

I. Overview

The centerpiece of the computer's architecture is the CPU, or Central Processing Unit, and the centerpiece of the CPU is the ALU, or Arithmetic-Logic Unit. In this project you will gradually build a set of chips, culminating in the construction of the ALU chip of the Hack computer. All the chips built in this project are standard across a diverse set of computer architectures, except for the ALU itself, which differs from one computer architecture to another. The objective of this project is to build many of the chips discussed in Chapter 2 and corresponding class lectures, leading up to an Arithmetic Logic Unit - the Hack computer's ALU.

The only building block chips that you may use in this project are the built-in versions of (*or your own*) basic logic gate chips and adder building blocks (HalfAdder and FullAdder), the chips that you will gradually build in this project, and the chips you made in Lab Exercise LE2.

II. Instructions

- Download the collateral provided on Canvas and use the skeletal files provided. Do not change the name of the files.
- Add your full name to the introductory comment present in each .hdl file.
- Implement the .hdl for each chip.
- Test the ALU using the completed .tst and .cmp files provided.
- You will need to complete the starter .tst and .cmp files provided for the FastRCA12 and Combo Add/Subtract chips on your own.

III. Chips

You may find the starter files for the chips to be constructed in this project in ***P2Codes.zip***

Chips Name	**File Name**	**Description**
Subtraction: Implement .hdl, (only starter .tst, and .cmp are provided) (22 points)		
Combo 10-bit Add/Subtract	AddSub10.hdl	Combo design with 10-bit Add/Subtract capabilities in 2s complement. **(22 points)**
Special Chips: Implement .hdl, (for FastRCA12 only starter .tst, and .cmp are provided) (38 points)		
FastRCA12	FastRCA12.hdl	Chained addition with optimal carry propagation configuration **(20 points)**
ActiveCounter16	ActiveCounter16.hdl	Count the number of active bits in a 16-bit input (complete .tst and .cmp files are provided) **(18 points)**
Arithmetic Logic Unit Implement .hdl (complete .tst and .cmp files are provided) (40 points)		
ALU	ALU.hdl	Arithmetic Logic Unit for 16-bit data inputs

Table 1 - List of Chips to be implemented in Project 2
Further discussion on the four chips appears later in this document.

IV. Proposed Implementation Sequence

1. Build a **combo 10-bit add-subtract chip**. Assume the inputs are in 2's complement.
2. Build the **fast RCA12** chip which exploits parallelism in the computation to minimize carry propagation delay. More discussion on this appears later in this document.
3. Build the **active 16-bit counter** chip that counts the number of active bits in a 16-bit string and return the 5-bit value in binary.
4. Finally, construct the **HACK ALU chip** as covered in the lecture material and Chapter 2.

NOTE: The combo add/sub and ALU chips are the only exercises where you must assume inputs in 2's complement format. For the other exercise, assume the inputs to be unsigned.

V. Submission

- Make sure you have completed all the requirements.
- Zip all of the required files and the signed honor code into a compressed folder named FirstName-LastName-UIN.zip
- Submit the .zip file on Canvas by the deadline.
- Late Submission Policy: refer to the syllabus.

VI. How Things Work

This section covers several topics you should be familiar with before working on the project. Below we share with you some additional background on the Combo Add/Subtract, Fast RCA and the ALU.

A. **COMBO ADD/SUBTRACT (AddSub10):** In this exercise, you are expected to construct a combo 10-bit Add/Subtract chip.

The chip has two 10-bit data inputs 'a' and 'b' specified in 2's complement format (this means that they can both be positive or negative, or in between). The chip also has a control input 'sub' that tells it to perform addition (sub==0) or subtraction (sub==1). Depending on the control signal, the 'b' input will either be used directly or processed through the steps of 2's complement creation for achieving '-b'. Also note that you will properly need to account for the overflow generation conditions. Refer to the lecture and lab material for ideas on how to accomplish this design.

B. **FastRCA12:**

A Ripple Carry Adder (RCA) relies on carry propagation across bit slices. The performance impact of carry propagation worsens with increasing adder capacity i.e., increasing number of bits that need to be added. For example, in a 64-bit addition using the standard RCA, the MSB slice must wait for all previous 63 slices to compute (i.e., give time for the carry to ripple through) before it can do its work. In this exercise, we seek to reduce the impact of delay propagation using a mechanism we will designate as "FAST RCA". We will explore this to build a 12-bit adder that we call FastRCA12. The central intuition behind FastRCA12 is that you can arbitrarily duplicate and parallelize hardware to reduce the total computational delay in performing 12-bit addition. Carry propagation delay in a ripple adder is caused because you can't start calculating the next segment without the carry bit of the previous segment. The carry bit only has two possible states though, so **what if you just calculate for both carry = 0 and carry = 1 and select the one that matters**!

In this exercise, you are to design a RCA12 using RCA4 style modules configured to optimize the carry propagation delay between any two modules. In other words, the carry propagation inside the RCA4 can still be sequential, but carry propagation across any two RCA4 modules is optimized through a clever scheme hinted above in RED. Clearly, such an approach has area overhead in module duplication and muxing logic, but it optimizes carry propagation delay. Note that for simplicity you can construct your RCA4 with all full adder slices.

C. ActiveCounter16:

This chip takes in a 16-bit input ‘in’ and counts the number of bits that are 1s and output to a 5-bit ‘out’.

Hint: The chip does addition for each active bit.

D. The HACK ALU

The Hack ALU produces two kinds of outputs: a "main" 16-bit output resulting from operating on the two signed two’s complement 16-bit inputs, and two 1-bit "status outputs" named 'zr' and 'ng'.

Read about the HACK ALU (Page 35 and onwards):

https://b1391bd6-da3d-477d-8c01-38cdf774495a.filesusr.com/ugd/44046b_f0eaab042ba042dcb58f3e08b46bb4d7.pdf

Watch HACK ALU Video:

https://www.youtube.com/watch?v=PEs855FNCOw&list=PLrDd_kMiAuNmSb-CKWQqq9oBFN_KNMTaI&index=17

VII. Additional Resources

In addition to class lecture material, the following is a list of helpful resources for this project.

Youtube - Nand2Tetris - Building a Modern Computer

The Elements of Computing Systems - Appendix A

The Elements of Computing Systems - Chapter 2 - Boolean Arithmetic