CSCE-312 | Project 3

Sequential Chips

Project Submission: 100 points

Grading

Project Submission [100%]:
You will be graded for correctness of the chips you have designed and coded. TA's 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. **For all chips, the prebuilt test collateral (.tst and .cmp) is provided to you**, so each chip needs to pass all its test cases to get full credit. If any chip does not pass all of its test cases, you will receive **0 points** on that chip.

Deliverables & Submission

You are required to turn in **the completed HDL files for all the chips** implemented. This includes the 5 chips required as part of Project 3 and any other chips that you needed to implement in order to complete Project 3. 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 computer's main memory, also known as Random Access Memory (RAM), is an addressable sequence of n-bit registers, each designed to hold an n-bit value. In this project you will build a RAM module utilizing the skills and knowledge from the labs. This involves two main issues: (i) how to use gate logic to store bits persistently, over time, and (ii) how to use gate logic to locate the memory register ("using the address") on which we wish to operate. In addition, you will build several functions that are constructed with combinational and sequential logic design elements.

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 and UIN to the introductory comment present in each .hdl file.
- Implement the .hdl for each chip.
- Test the basic chips using the completed .tst and .cmp files provided.

III. Chips

You may find the starter files for the chips to be constructed in this project in ***P3Codes.zip***

Build all the chips described in the list below. **To build the RAM chip you need to have some understanding of how a bit and a register works. Details about these were discussed in the lectures and labs to develop a working level understanding of these chips, and you'll also build them yourself in Lab Exercise 3.**

Chips Name:	**File name**	**Description**
Basic Chips: **Implement .hdl (complete .tst and .cmp files are provided)**		
RAM64	RAM64.hdl	64 16-bit register memory **(20 pts)**
RAM512	RAM512.hdl	512 16-bit register memory **(20 pts)**
PC	PC.hdl	16-bit program counter **(20 pts)**
AggieCipher	AggieCipher.hdl	Cipher input by adding to a 4 bit counter sum **(20 pts)**
Fibonacci	Fibonacci.hdl	Fibonacci Sequence generator **(20 pts)**

IV. Proposed Implementation Sequence

1. Take a closer look at the *modular memory* chips constructed in the labs. You'll design **RAM64** and **RAM512** very similarly. Use modular construction techniques to build these chips. Use the *RAM16* as designed in the lab or design your own from the builtin *RAM8* as a helper chip in constructing **RAM64** and reuse **RAM64** for **RAM512**.
2. Build a Program Counter (**PC)** chip which outputs a monotonically increasing count on its output. The PC can be used as a timer and may be used in constructing the AggieCipher chip.
3. Build the **AggieCipher** chip which sums user input and a timer count.
4. Build the **Fibonacci** chip which outputs numbers in the Fibonacci sequence.

V. Submission

- Make sure you have completed all of the requirements.
- Zip all of the required files 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

A. RAM512

Basic principles: V11

Here you can learn how to construct RAM4 using FOUR 16-bit registers . Additionally, you may look at Recitation Video RV11 to see details of RAM16 built with RAM4. You may apply the principles learnt here to construct RAM64 and RAM512 as proposed in Section IV earlier.

B. PC:

Principles and Approach: Lecture Video V11 showcases the process of constructing the program counter. **Just remember a small error at timestamp 1:00:38 which is corrected in the accompanying slide in CL11** (LINK).

You may also refer to the textbook (page 51) for details on the workings of the Program Counter (PC) chip. The book is available on the nand2tetris.org page here: CHAPTER 3.

C. AggieCipher:

Design a simple cipher logic which generates a code which is equal to a user-provided 4-bit *input + the value generated from a counter*, where counter value starts from 0000, and increments by 1 every clock cycle. The counter wraps to 0000 when it reaches a count of 15. You may use your program counter (PC) chip to implement the Aggie Cipher logic.

***out=in+counter**, where counter={0,1,2,3,4,5,6,....,15,0,1,2,.....}*

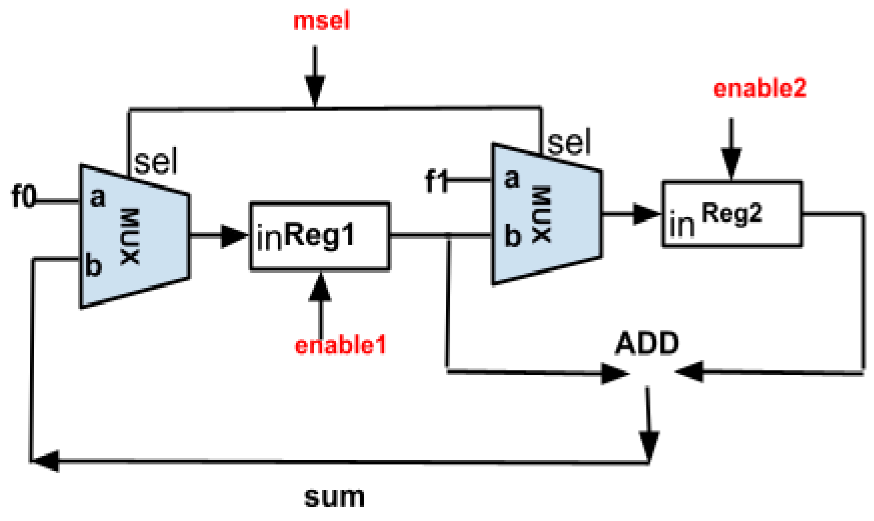
For this problem you only need to know the background of an adder (you may use the builtin adder Add16) and the program counter as a timer (you may use the builtin Program Counter, PC) in order to construct the AggieCipher. One of the key realizations here is how to achieve the 'wraparound' effect of 0,1,2,3,4,5,6,....,15,0,1,2,..... with a 16-bit counter. When in doubt, simulate the PC chip and watch the lower 4 bits 0..3.

D. Fibonacci:

The general Fibonacci sequence is a sequence that starts with f0=0 and f1=1 . The next number in the sequence is the sum of the previous two numbers. So the Fibonacci number sequence generated in our circuit will be:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89…

Below is a schematic logic diagram to accomplish the construction of the Fibonacci sequence. Draw the clocked waveforms for the output to convince yourself of the chip's working before you implement the HDL. It'll be much more fun that way.

The ADD operation in the circuit diagram is the addition of the two previous Fibonacci numbers and you can use any 16 bit adder chip (including the builtin Add16) for this purpose.



MUX behavior: if sel==0 then out=a else out=b

REG behavior: if load(t) then out($t+1$) = in(t)

e

else out($t+1$) = out(t)

ADD performs addition of its 2 inputs to generate sum. Carry output is ignored.

To use this circuit, you have to control the enable signals, namely, enable1, enable2 and msel.

- *msel=0* will select the starting values *f0* and *f1* of the Fibonacci Sequence
- *msel=1* will keep running the Fibonacci sequence with *sum(t+1)* ← *sum(t)* + *sum(t-1)* for a given clock cycle *t*
- *enable1=1 and enable2=1* activate their respective registers by **loading** the corresponding input values to be visible at their corresponding register outputs

The test file Fibonacci.tst assigns the values to these control signals.
Notice how the output in the Fibonacci.cmp file changes while changing those signals.

VII. Background Resources

LECTURE CONTENT (CANVAS):

1. **L10-V10-N10-CL10 + Video and Practice Questions** to firm up understanding of the working of DFF
2. **L11-V11-N11-CL11 + Video and Practice Questions** to understand design approach for Modular Memory and Program Counter

LAB CONTENT (CANVAS):

1. Slides, hdl files and notes for sequential chips

BOOK Resources are as follows:

Chapter 3
https://docs.wixstatic.com/ugd/44046b_1801b5682e4d4a67bd05e14235665d8b.pdf
Appendix A
https://docs.wixstatic.com/ugd/44046b_d715f80dca2a43af926131a52e3d3d90.pdf