# CS 4341 Fall 2021
# Project, Part 2
# Assigned Week of September 13th
# Due October 11th, 11:59 pm

## 1 Objective

For this semester, you are to form a team, and build a simple processor, such as an ALU, in a hardware design language. For the second part of the project, cohorts must have combinational logic components incorporated into a circuit diagram and have a matching Verilog code and output.

## 2 Tasks

The following tasks to complete for this project will be:

- Submit an update of Part 1 of the Project
- Read the description of the circuit below, and design and build the circuit. This circuit will contain combinational logic components and a detailed description.
- Construct the circuit in Verilog
- Run the Verilog simulation and capture the output.
- Keep a journal of meeting, tasks, and progress
- Submit a Peer Participation Report

### 2.1 Update Part 1

Since Part 1 has been submitted, changes may have changed in the cohort membership, description, new ideas about the project, or even new software, including the name of the cohort and its description, its members, the software, and the description of what you are to build. Anything removed from Part 1 of the projects will be marked with a ~~strikethrough~~. Anything added to the update will be marked in **bolded, blue text**. As progress is made through the semester, the cohort should have some new ideas. At least one change of some kind must be in the update to receive credit. Turn this in as a file named <cohort>.update.pdf.

## 2.2 The System Design with Circuit Diagram

For this next part, the minimum required to be done is the following. Your cohort may be advancing at a faster pace and may clearly be larger. But the following must be designed and built:

### 2.2.1 What to build, the Given

A combinational logic circuit must be put together with the following components.

- Two inputs will represent 16-bit integers. A third input will represent a 4-bit operational code.
- One output will represent a 32-bit integer, and another output will represent a 2-bit error code.
- The inputs will feed into an four arithmetic modules: adder-subtractor, multiplication, division, and modulo.
    - The adder-subtractor module will be structured code.
    - The multiplication module will be structured code.
    - The division module will be behavioral code.
    - The modulo module will be behavioral code.
- The output of the arithmetic will be controlled by a 16-channel, 32-bit, multiplexor with a one-hot select.
    - The sum of the adder-subtractor module will feed into a multiplexor input channel
    - The difference of the adder-subtractor module will feed into a different multiplexor input channel.
    - The product of the multiplication module will feed into yet another multiplexor input channel.
    - The quotient of the division module will feed into yet another multiplexor input channel.
    - The remainder of the modulo module will feed into yet another multiplexor input channel.
- The errors of the arithmetic will be fed into a 2-bit error code.
    - The adder-subtractor module will have an overflow line that feeds into the least significant bit of the error output.
    - The division module and the modulo module will have divide-by-zero lines OR'd together, the result of which feeds into the most significant bit of the error module.
- Assign a 4-bit code to each of the 5 arithmetic functions.
- Use a 4-to-16 Decoder to translate the 4-bit codes into the **one-hot** the channel select value to the Multiplexor.

### 2.2.2 System Design Description

The System Design description has five tasks that need to be completed. The list of inputs, the list of outputs, the list of interfaces, the list of parts, and the top-level circuit diagram.

#### *2.2.2.1 The List of Inputs*

The description in section 2.2.1 gives three inputs. The cohort needs to assign identifier names to these inputs, and list them, including their bit-size and a basic description.

#### *2.2.2.2 The List of Outputs*

The description in section 2.2.1 gives two outputs. The cohort needs to assign identifier names to these outputs, and list them, including their bit-size and a basic description.

#### *2.2.2.3 The List of Interfaces*

The description in section 2.2.1 has modules and components that connect to each other. Math modules connect to the multiplexor, the Decoder to the channel select of the multiplexor. These internal connection lines are called interfaces. The cohort needs to assign identifier names to these interfaces, and list them, including their bit-size and a basic description.

#### *2.2.2.4 The List of Parts*

The description in section 2.2.1 lists out an adder-subtractor module, a multiplier, a divisor, a modulo, a multiplexor, and a decoder.  Each of these is a combinational logic component. The cohort needs to assign identifier names to these components, and list them, including what type of component and a basic description.

#### *2.2.2.5 The Top-Level Circuit Diagram*

Only the top-level circuit diagram is needed for this portion of the project.  *The interior workings of the arithmetic modules, the decoder, and the multiplexor are not required.* The overall layout of the circuit begins with the inputs far left, next the arithmetic modules near left, then the multiplexor and decoder near right, and finally the output on the far right. Each line should be connected to the correct component, labeled with their identifiers, and marked with a bus-size shown as a slash on the line, with the size beneath.

Save this as a file called <cohort>.SystemDesign.pdf

## 2.3 Write the Verilog Code

The next part of the project is to write the Verilog code that matches the circuit you designed in section 2.2. As with Part 1 of the project, the code begins with a testbench and a breadboard. Also, modules will be created for the adder/subtractor, the multiplication, the division, the modulo, the decoder, and the multiplexor. ***The identifiers of these modules must match the identifiers chosen in the system design list of parts.***

All of these modules will be instantiated in the breadboard module and connected using parameters. ***The names of the parameters between those modules must match the identifiers chosen in the system design list of interfaces.***

The testbench module will control the input and output in the stimulus. The stimulus will have hard-coded values to be the inputs to the circuit. The stimulus will also display the matching output. ***The names of the input and output variables in the TestBench must match the identifiers chosen in the system design list of inputs and list of outputs.***

What will be seen in the Verilog code is the following. All the variables are created first. Next, the modules are instantiated with those variables being the parameters. Then down in the initial portion of the TestBench, chose the input, and the command, do a delay (such as #60; ), then display the output. Then do the next instruction.

Turn in a file named <cohort>.Part2.v

## 2.4 Run the Code and Capture the Output

Once the code is running, the cohort must demonstrate that the modules work, and that the output be clearly readable.

To do so, do the following 10 operations in the TestBench.

- Set the two inputs to integers less than 250 and add them.
- Set the two inputs to integers less than 250 and subtract them.
- Set the two inputs to integers less than 250 and multiply them.
- Set the two inputs to integers less than 250 and divide them.
- Set the two inputs to integers less than 250 and modulo them.
- Set the two inputs to integers greater than 16000 and add them.
- Set the two inputs to integers greater than 16000 and subtract them.
- Set the two inputs to integers greater than 16000 and multiply them.
- Set the two inputs to integers greater than 16000 and divide them.
- Set the two inputs to integers greater than 16000 and modulo them.

For the output, make your display something like this:

Let us assume the codes are: Add, 0001, Subtract, 0010, Multiply: 0011, Divide, 0100, and Modulo, 0101.

```
[Input A:   255, Input B:   127][Add:0001][Output:       382, Error: 00]
[Input A:   255, Input B:   127][Sub:0010][Output:       128, Error: 00]
[Input A:   255, Input B:   127][Mul:0011][Output:     32385, Error: 00]
[Input A:   255, Input B:   127][Div:0100][Output:         2, Error: 00]
[Input A:   255, Input B:     0][Mod:0101][Output:         X, Error: 10]
[Input A: 32000, Input B: 16000][Add:0001][Output:     48000, Error: 00]
[Input A: 32000, Input B: 16000][Sub:0010][Output:     16000, Error: 00]
[Input A: 32000, Input B: 16000][Mul:0011][Output: 512000000, Error: 00]
[Input A: 32000, Input B: 16000][Div:0100][Output:         2, Error: 00]
[Input A: 32000, Input B: 16000][Mod:0101][Output:         0, Error: 00]
```

Turn in a capture of this file in the form of <cohort>.part2.pdf

### 2.5 Journal

A calendar of events, meeting, discussion, choice of software, writing code, that has the date, a description of the work done, and who was doing the work. Turn in the file called <cohort>.Log2.pdf

| Date | Task | Who |
|---|---|---|
| September 20 | Working out design | Fred, Daphne, Velma, Norville |
| September 21 | Draft of circuit diagram | Velma, Fred |
| September 27 | Putting together System design | Daphne, Norville |
| September 28 | Final draft of circuit digram | Velma, Fred |
| October 2 | Start working on Verilog | Fred, Daphne, Velma, Norville |
| October 4 | We are sick of Verilog | Norville |
| October 5 | We got it to run! | Norville, Daphne |
| October 10 | Code tidied to match System Design | Fred, Daphne, Velma, Norville |

### 2.6 Peer Grade

Grade your Peers in this portion. Did they show up? Did they communicate? Did they do their work? If yes, say so. If no, say so. If it was bad, here is where you can tell the instructors. If it was good, here is a place to give praise. What is written in this portion will not be shared with other members of the cohort or the class or the public. This file will be a separate file for each individual member of the team, and should be submitted in the form of <netid>.peer.pdf

## 3 Turn-In

- The updated Cohort and Project description, named <cohort>.update.pdf
- A file called <cohort>.SystemDesign.pdf that contains
  - Description of all inputs
  - Description of all outputs
  - Description of all interfaces
  - Description of all combinational logic components
  - A correctly labelled Top-Level circuit diagram
- A text file, named <cohort>.Part2.v, that contains the Verilog code
- A text file, named <cohort>.Part2.txt, that contains the Verilog output
- The Work Log named <cohort>.Log2.pdf that contains the Journal
- A Peer Participation report, unique to each cohort member, named <netide>.peer.pdf