Project Checkpoint 2

Full ALU Checkpoint 2

Logistics

This is the first Project Checkpoint for our processor. We will post clarifications, updates, etc. on Sakai.

- Due: **Friday, September 30, 2022,** by **11:59 PM** **(Duke time)**
    - Late policy can be found on the course webpage/syllabus

Introduction

Design and simulate an ALU using Verilog. You must support:

- a **non-RCA** adder with support for addition & subtraction (that you have done in the last checkpoint)
- bitwise AND, OR **without** the built-in &, &&, |, and || operators
- 32-bit barrel shifter with SLL (Logical Left Shift) and SRA (Arithmetic Right Shift) **without** the <<, <<<, >>, and >>> operators

Module Interface

Designs which do not adhere to the following specification will incur significant penalties.

Your module must use the following interface (n.b. It is the template provided to you in alu.v):

```
module alu(data_operandA, data_operandB, ctrl_ALUopcode,
ctrl_shiftamt, data_result, isNotEqual, isLessThan, overflow);

   input [31:0] data_operandA, data_operandB;
   input [4:0] ctrl_ALUopcode, ctrl_shiftamt;

   output [31:0] data_result;
   output isNotEqual, isLessThan, overflow;

endmodule
```

Each operation should be associated with the following ALU opcodes:

Operation	ALU Opcode	Description
ADD	00000	Performs `data_operandA + data_operandB`
SUBTRACT	00001	Performs `data_operandA - data_operandB`
AND	00010	Performs (bitwise) `data_operandA & data_operandB`
OR	00011	Performs (bitwise) `data_operandA | data_operandB`
SLL	00100	Logical left-shift on `data_operandA`
SRA	00101	Arithmetic right-shift on `data_operandA`

Control Signals (In)

- `ctrl_shiftamt`
    - Shift amount for SLL and SRA operations
    - Only needs to be used in SLL and SRA operations

Information Signals (Out)

- `isNotEqual`
    - Asserts true **iff** `data_operandA` and `data_operandB` are not equal
    - Only needs to be correct after a SUBTRACT operation
- `isLessThan`
    - Asserts true **iff** `data_operandA` is **strictly** less than `data_operandB`
    - Only needs to be correct after a SUBTRACT operation
- `overflow`
    - Asserts true **iff** there is an overflow in ADD or SUBTRACT
    - Only needs to be correct after an ADD or SUBTRACT operation

Permitted and Banned Verilog

Designs that do not adhere to the following specifications cannot receive a score.

No "megafunctions."

- *Tip: think about whether your codes can specify only one design!*

Use structural Verilog like:

- `and and_gate(output_1, input_1, input_2 ... );`

Not allowed to use SystemVerilog or syntactic sugar like:

- `+, -, *, /, %, **, ==, >=, &, ^, |, &&, ||, !, <<, <<<, etc`
- ***`if`***`, `***`else`***`, and `***`case`***` statements, `***`for`***` loop, etc`

except in constructing your DFFE (i.e. you can use whatever you need to construct a DFFE).

However, feel free to use the following syntactic sugar and primitives:

- **`Bitwise not(~)`**
- **`assign ternary_output`**` = cond ? High : Low;`
  - The ternary operator is a simple construction that passes on the “High” wire if the cond wire is asserted and “Low” wire if the cond wire is not asserted
- **`generate if, generate for,`** and/or **`genvar`**
  - It could reduce the repeated lines but maintain the structural design
  - Any expression to specify the range, e.g., a[(i+24)%7]

Grading Breakdown

Test	Points	Scoring Method
Less Than	10	Proportional (-0.2 pts / test case fail until zero)
Not Equal	10	Proportional (-0.2 pts / test case fail until zero)
Or	20	Proportional (-0.2 pts / test case fail until zero)
And	20	Proportional (-0.2 pts / test case fail until zero)
Logical Left Shift	20	Proportional (-0.2 pts / test case fail until zero)
Arithmetic Right Shift	20	Proportional (-0.2 pts / test case fail until zero)

Other Specifications

Designs which do not adhere to the following specifications will incur significant penalties.

Your design must operate correctly with a 50 MHz clock. Also, please remember that we are ultimately deploying these modules on our FPGAs. Therefore, when setting up your project in Quartus, be sure to pick the correct device.

Submission Instructions

Designs that do not adhere to the following specifications will incur significant penalties.

Writing Code

- Keep all of your source files in the top-level directory.
- Make sure you structure your codes so that alu.v is the top-level entity and it contains the provided alu interface.
- Change how your repo is configured at your own risk.
- You can choose to use the GitHub repository to manage your codebase if you are familiar with that. A few suggestions:
    - Branch off of main to implement your projects and merge changes back into main when you've completed a feature or you want to test.
    - Be sure to only put files into version control that are source files (*.v).
    - Modify .gitignore at your own risk.

Submission Requirements

- When using **Gradescope** to submit your design, please submit **one .zip file** and the file should include your code and a README.md file. For **Github** submission, click 'connect GitHub', link your account, and select the correct repo and branch you want to submit.
- The submitted codes should contain **all necessary *.v modules** to execute your alu. The autograder will read and examine all .v files in the .zip file; therefore, you may be able to include subfolders but you should be aware that if you submit unnecessary .v files it could cause compile errors.
- **Make sure the name of all testbench files ends with '_tb.v'**; otherwise, it will be involved in the style check and negatively affect your submission grade.
- A README.md (written in markdown, Github flavor) should include
    - Your name and netID,
    - A text description of your design implementation (e.g., "I used X,Y,Z to ..."),
    - If there are bugs or issues, descriptions of what they are and what you think caused them.

Resources

Test cases in this project have already been included in your alu_tb.v in Project 1, you can simply comment back the related statements. However, the testbench used for grading will be more extensive than the one presented here. **Passing the included testbench does not ensure that you will pass the grading testbench.**