Assignment 01
ALU design

1 Objectives

- To understand the basic concept of an Arithmetic and Logic Unit (ALU).
- To design and simulate an ALU.
- To understand how the ALU provides a status of the result of each operation.

2 Deadline

23:59 hours on Friday November 19th 2021.

3 Teamwork policy

This is an individual assignment.

4 Background

An Arithmetic and Logic Unit (ALU) combines a variety of both mathematical, boolean, and logical operations into a single unit. The schematic symbol of an ALU is shown in Figure 1.

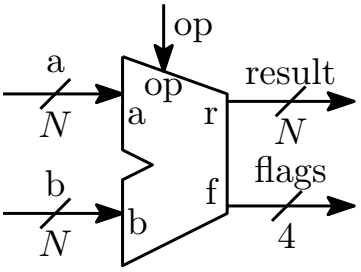

Figure 1: ALU symbol.

In Figure 1, the ALU receives two N-bit operands, `a` and `b`, as well as a control signal `op`, which indicates the type of operation the ALU must perform. More specifically, the value of `op` specifies whether the ALU should perform an addition, a subtraction or any bitwise logical operation between the two input operands. The ALU delivers an N-bit output, labelled as `result` in Figure 1, which corresponds to the result of the arithmetic or logical operation. In addition to this, the ALU delivers a 4-bit output, labelled as `flags` in Figure 1, which provides a status regarding the result of each ALU operation. More specifically, the output

`flags` indicates whether the output `result` is zero or negative, as well as whether an arithmetic operation produced an overflow or a carry out condition.

Table 1: ALU ports.

Port name	Direction	Description
`a`	`input`	Operand a.
`b`	`input`	Operand b.
`op`	`input`	Operation to be performed.
`result`	`output`	Result from the arithmetic or logical operation.
`z`	`output`	Zero flag. Active high when result is exactly zero.
`n`	`output`	Negative flag. Active high when result is negative.
`v`	`output`	Overflow. Active high when an overflow occurs in an arithmetic operation.
`c`	`output`	Carry flag. Active high when a carry out is produced in an arithmetic operation.

Note that the output `flags` of Figure 1 is split into outputs `z`, `n`, `v`, and `c` in Table 1.

5 ALU design exercise

In this assignment, you are required to design and simulate an ALU. In order to allow flexibility in this assignment, you may choose to design your ALU using either Logisim simulator or SystemVerilog Hardware Description Language (HDL). You are required to submit **ONLY ONE** ALU design. More specifically, you might submit the design using Logisim **OR** SystemVerilog.

IMPORTANT! Please note that SystemVerilog is not covered as a topic in this course. Hence, the ALU design with SystemVerilog is completely optional. However, this challenge might be a great opportunity to learn a sought-after industry skill, if you are interested in pursuing a career in the design and verification of Integrated Circuits (ICs).

Section 5.1 and Section 5.2 describe the design specifications for designing the ALU in Logisim and SystemVerilog, respectively.

5.1 ALU design in Logisim

The ALU designed using Logisim must be a 4-bit ALU with flags generation. The input and output ports of the ALU are specified in the list below.

- Two 4-bit inputs, namely `a` and `b`. These inputs are the operands of the ALU.
- Input `op`, which will be used by the ALU in order to select which operation will be performed. Note that the number of bits for the input `op` must be selected according to the number of operation supported by the ALU. It is your task to determine the number of bits for `op`.
- 4-bit output `result`. This output contains the result of the arithmetic or logical operation.
- Output port `z`. Zero flag.
- Output port `n`. Negative flag.
- Output port `v`. Overflow flag. This flag should only be enabled when the ALU performs an arithmetic operation. For logical operations, this flag should always be 0.
- Output port `c`. Carry flag. This flag should only be enabled when the ALU performs an arithmetic operation. For logical operations, this flag should always be 0.

5.2 ALU design in SystemVerilog

Previous work

If you opt for designing and simulating the ALU in SystemVerilog, I strongly suggest that you to complete the SystemVerilog tutorial that is included with this assignment. After completing the tutorial, you might come back to complete the rest of this section.

ALU design exercise

You are provide with `alu_pkg.sv`, `alu.sv`, and `tb_alu.sv` SystemVerilog files. Each of these files is explained below.

- `alu_pkg.sv` contains a basic definition of the ALU operations. SystemVerilog package files are somewhat analogous to `.h` header files in C language. SystemVerilog package files are useful to declare and share parameters and user-defined data types among several

SystemVerilog files and modules that are included in a single design. For this assignment, you are not required to modify `alu_pkg.sv`.

- `alu.sv` contains a basic skeleton of an ALU design. However, this file is incomplete and you are required to include SystemVerilog code for completing the ALU design. More specifically, you are required to complete the SystemVerilog code that will perform each of the ALU operations, as well as the logic required for generating the ALU flags `z`, `n`, `v`, and `c`.
- `tb_alu.sv` is a testbench file for the ALU module declared in `alu.sv`. More specifically, `tb_alu.sv` generates stimuli and basic functionality tests for the ALU input ports, to which the ALU must respond correctly. This file instantiates (uses) one copy of the ALU design and connects its ports to signals declared in the testbench. You are not required to modify this file. However, you are encouraged to have a look at it and understand the code provided. You are also encouraged to modify this file, in order to apply different test scenarios, **after** you have completed this assignment.

ALU simulation in ModelSim

In order to verify the correct operation of your ALU, you are required to perform a simulation using ModelSim. For this purpose, you might follow the steps provided in the tutorial files for compiling and simulating your design. When working with SystemVerilog packages, it is recommended to specify the order in which the SystemVerilog files will be compiled. More specifically, in ModelSim, you should select **Compile** → **Compile order**, and make sure that `alu_pkg.sv` is the first file in the list.

After successfully compiling all your SystemVerilog files, you might begin the simulation, as stated in the tutorial. This include the steps necessary for adding and viewing waveforms into the simulation. Note that once you have waveforms available in your simulation, you can change the radix (format) in which the signals are displayed. This is particularly useful when working with multi-bit width signals, such as the ALU operands. For example, right-click on the input signal `a` in the **Wave** pane and select **Radix** → **Decimal**.

ALU synthesis in Quartus

Quartus might be used for performing logic synthesis of the ALU. Synthesis is the process of generating logic gates from SystemVerilog (or any HDL). In other words, a synthesizer, such as Quartus, will translate your SystemVerilog file into simple logic gates that will be later used for implementing the design in a Field-Programmable Gate Array (FPGA) or an Application Specific Integrated Circuit (ASIC). **A design that can not be synthesized is a useless design.** Use the tutorial as a reference in order to synthesize and observe the equivalent Register-Transfer Level (RTL) view of the ALU using Quartus. Note that Quartus must only be used to synthesize design units. Testbenches are never synthesized.

6 Deliverables and submission instructions

This section details the deliverable and submission instructions for this assignment.

6.1 Deliverables

Please follow the delivery instructions below depending on your choice of ALU design. Remember that you are required to submit **ONLY ONE** ALU design, either in Logisim or SystemVerilog.

ALU in Logisim

Submit your ALU `.circ` Logisim file.

ALU in SystemVerilog

Submit the following items.

- `alu.sv`. This file should contain the ALU module definition and operation.
- A screenshot of your ModelSim simulation.
- A screenshot of the ALU's RTL view after synthesis in Quartus.

6.2 Submission

Submit your assignment through Canvas **no later** than 23:59 hours on Friday November 19th 2021.

IMPORTANT! This is a desing exercide.Therefore, there may be **multiple** correct answers.

Please send any questions to isaac.perez.andrade@tec.mx.