Experiment 1 - Creating an Arithmetic Logic Unit (ALU)

Objectives

The purpose of this experiment is to practice your combinational design skills through one of the most fundamental structures used in computer architectures, an Arithmetic Logic Unit (ALU). As in the previous experiment, you will use the given cocotb testbenches to test your design. Finally, in the experiment sessions, you will expand upon your design and implement it onto the DE1-SoC Board, equipped with a field programmable gate array (FPGA) using several peripheral units namely the switch inputs, button inputs, LEDs, and 7-segment outputs.

Please take your time and familiarize yourself with everything present in this preliminary work as what is taught here will be used throughout the entire course.

Although the preliminary work grade is not included in the overall grade of the experiment,

1. **You need to get at least 50% from the report.**
2. **Your codes should pass at least half of the testbenches (2/3 for this experiment).**

Both are required to be eligible to attend the experiment session. We will run the testbenches automatically on your submitted codes for the second requirement.

1 Preliminary Work

In this experiment, you will design and implement an Arithmetic Logic Unit (ALU) using Verilog. The ALU consists of two submodules: a Logic Unit (LU) and an Arithmetic Unit (AU). Once completed, you will implement the ALU on the FPGA boards during the experiment session. The concepts introduced here are foundational and will recur throughout the course, so take the time to master them thoroughly.

For the preliminary work, you will submit two separate assignments: A .pdf report, and Verilog codes inside a .zip file.

To fulfill the requirements of this laboratory work, the following tasks should be performed. **Note: If necessary you should only include important parts of the code in the pdf report. The full codes should be submitted separately as a .zip file.**

Your report must include RTL schematic views of the specified components, along with screenshots showing the successful execution of the testbenches.

All designs must be written in Verilog HDL and included in your submission as a ".v" file wherever required.

1.1 Reading Assignment

The Laboratory Manual, where the regulations and other useful information exist, is available on the ODTUClass course page. **Read the Laboratory Manual thoroughly**. Refer to the combinational Verilog lecture notes Part 1 and 2 as they contain everything about Verilog you need **for this experiment**. We also assume you are familiar with Chapters 1-3 of the EE348 course as this experiment is complementary to that.

If you are unfamiliar with Verilog, Quartus, or Cocotb please refer to the Experiment 0 manual.

1.2 Design Question: Arithmetic Unit, Logic Unit, and Arithmetic Logic Unit (100% Credits)

In this part, you will learn how to make an ALU module that can perform addition, subtraction, bitwise OR, and bitwise AND operations on two W-bit input signals. This ALU is constructed using two smaller modules called an AU and LU which can perform addition/subtraction and AND/OR respectively. **For all parts, you must use the supplied template Verilog files, do not change the file names, module names, or module I/O names.**

1.2.1 Creating the Project

1. Download SystemBuilder from ODTUClass course page if you haven't already.
2. Open SystemBuilder (Section 4 of the DE1-SoC User manual.).
3. Set your project name to EE314exp1.
4. You only need 7-Segmentx6, LEDx10 and Switchx10 for this experiment. **Note:** 7-segment interface module is given on ODTUClass inside the "hexto7seg.v" file, it will also be supplied during the laboratory sessions.
5. Generate the project.

Notice that the SystemBuilder also creates a Verilog file with the same name as the project, "EE314exp1" in this case. This file will be useful when uploading your designs to the FPGA boards as described in the section 2.

1.2.2 Describing the AU using Verilog HDL (30% Credits)

Design a W-**bit AU** that performs 2's complement addition and subtraction, where W is a configurable parameter representing the data width of its operands. You must only utilize **assign** statements for the logic.

The AU features:

- **Data Inputs:**
  - Two W-bit operands `A` and `B`.
- **Control Input:** A 1-bit `control` signal selecting among 2 operations.
- **Outputs:**
  - A W-bit `OUT`.
  - Four status bits:
    - Carry Out (CO)
    - Overflow (OVF)
    - Negative (N)
    - Zero (Z)

The detailed specifications for the AU operations and status outputs are provided in Table 1a and Table 1b, respectively.

1. Add to your project and fill in the shared template file "Arithmetic_Unit.v" to implement the module. Only use **assign** statements for creating logic.
2. Generate RTL schematics for the "Arithmetic_Unit" using $W = 4$ and add the screenshot of the RTL schematic to your report as described in the Experiment 0 manual. **(20% Credits)**
3. Test the "Arithmetic_Unit" module using the testbench supplied on ODTUClass and add the screenshot of a successfully passed test (a single test screenshot suffices) to your report as described in the Experiment 0 manual. **You should test with multiple W values (4,8,12) to ensure correctness.** Testbench automatically adjusts for your W value so you only need to change your ".v" files. **(10% Credits)**

Table 1: AU Control and Status

(a) AU Operation Control

AU Control	**AU Operation**	**Symbol**
0	SubtractionAB	$A - B$
1	Addition	$A + B$

(b) AU Status Descriptions

Status	**Description**
CO	1 if there is a Carry Out from add or subtract operations, 0 otherwise
OVF	1 if the add or subtract operation results in overflow, 0 otherwise
N	1 if the result is negative, 0 otherwise
Z	1 if the result is zero, 0 otherwise

Note for overflow: If two positive numbers are added and the result appears negative (or vice versa), it indicates an overflow. Similarly, subtraction (A–B) is performed by adding A to the two's complement of B.

Note for carry out: For both addition and subtraction carry out should always be taken as the MSB+1'th bit of the result. For example, for a 2-bit design carry out is the 3rd bit.

1.2.3 Describing the LU using Verilog HDL (30% Credits)

Design a W**-bit Logic Unit (LU)** that performs bitwise **AND** and **OR** operations, where W is a configurable parameter defining the data width of its operands. You must only utilize **assign** statements for the logic.

The LU features:

- **Data Inputs:**
  - Two W-bit operands `A` and `B`.
- **Control Input:** A 1-bit `control` signal selecting among 2 operations.
- **Outputs:**
  - A W-bit `OUT`.
  - Two status bits:
    - Negative (N)
    - Zero (Z)

The detailed specifications for LU operations and status outputs are provided in Table 2a and Table 2b.

1. Add to your project and fill in the shared template file "Logic_Unit.v" to implement the module. Only use **assign** statements for creating logic.
2. Generate RTL schematics for the "Logic_Unit" using $W = 4$ and add the screenshot of the RTL schematic to your report as described in the Experiment 0 manual. **(20% Credits)**
3. Test the "Logic_Unit" module using the testbench supplied on ODTUClass and add the screenshot of a successfully passed test (a single test screenshot suffices) to your report as described in the Experiment 0 manual. **You should test with multiple W values (4,8,12) to ensure correctness.** Testbench automatically adjusts for your W value so you only need to change your ".v" files. **(10% Credits)**

Table 2: LU Control and Status

(a) LU Operation Control

LU Control	**LU Operation**	**Symbol**
0	AND	$A \wedge B$
1	OR	$A \vee B$

(b) LU Status Descriptions

Status	**Description**
N	1 if the result is negative, 0 otherwise
Z	1 if the result is zero, 0 otherwise

1.2.4 Combining the Modules for the ALU (40% Credits)

Implement a W-bit Arithmetic Logic Unit (ALU) for 2's complement arithmetic, where the parameter W defines the operand data width. To achieve this, you may introduce new **assign** statements for the status flag logic. However, ensure that the arithmetic and logical operations are executed using a **single instantiation of the previously designed AU and LU modules each.**

The ALU features:

- **Data Inputs:**
  - Two W-bit operands.
- **Control Input:** A 2-bit control signal selecting among 4 operations.
- **Outputs:**
  - A W-bit `OUT`.
  - Four status bits:
    * Carry Out (CO)
    * Overflow (OVF)
    * Negative (N)
    * Zero (Z)

Note: The Negative (N) and Zero (Z) flags are affected by all ALU operations. The Carry Out (CO) and Overflow (OVF) flags are updated only during arithmetic operations (Addition and subtraction in this case), otherwise they take the value "0".

Detailed specifications for the ALU operations and status outputs are provided in Table 3 and Table 4, respectively.

1. Add to your project and fill in the shared template file "ALU.v" to implement the module.
2. Generate RTL schematics for the "ALU" using $W = 4$ and add the screenshot of the RTL schematic to your report as described in the Experiment 0 manual. **(25% Credits)**
3. Test the "ALU" module using the testbench supplied on ODTUClass and add the screenshot of a successfully passed test (a single test screenshot suffices) to your report as described in the Experiment 0 manual. **You should test with multiple W values (4,8,12) to ensure correctness.** Testbench automatically adjusts for your W value so you only need to change your ".v" files. **(15% Credits)**

Table 3: ALU Operation Control

ALU Control [1:0]	**ALU Operation**	**Symbol**
00	AND	$A \wedge B$
01	OR	$A \vee B$
10	SubtractionAB	$A - B$
11	Addition	$A + B$

Table 4: ALU Status Descriptions

Status	**Description**
CO	1 if there is a Carry Out from add or subtract operations; 0 for logic operations
OVF	1 if the add or subtract operation results in overflow; 0 for logic operations
N	1 if the result is negative
Z	1 if the result is zero

2 Experimental Work

In the experiment session, you will have two parts, a preliminary work demonstration and a design task.

2.1 Preliminary Work Demonstration (20% Credits)

Run the cocotb testbench for the ”ALU” module once again (with W specified by the lab instructor) and show it to your laboratory instructor.

2.2 Experiment Session Design Task (80% Credits)

The experiment sessions will require expansions on the ALU, details of which will only be given when the experiment sessions start.

A Parts List

`DE1-SoC Board`