CMSC 132: Computer Architecture
First Semester, AY 2020-2021
Exercise 03: Combinational Circuits
Prepared by: Clinton Poserio

Arithmetic and Logic Unit or ALU is a digital combinational circuit which is responsible for computing basic arithmetic operations and bitwise operations, specifically on integer binary inputs. ALUs are often designed with two sub-units: an arithmetic unit and logic unit to separate the aforementioned operations. For this exercise, your task is to create a 2-bit ALU which accepts a total of seven bits as input and produces three bits as output. Let $s_0, s_1, s_2, i_0, i_1, j_0, j_1$ be the input bits where:

- s_x are the selection bits which indicates the type of operation to be executed; $x = 0$ being the most significant bit and $x = 2$ the least significant bit and
- i_y and j_y are two bit integer inputs; for both inputs (and the output), $y = 0$ is the most significant bit.

and k_0, k_1, c_{out} be the output bits where:

- k_y is a two bit integer; the output of the specified operation and
- c_{out} is the "carry out" bit from arithmetic operations

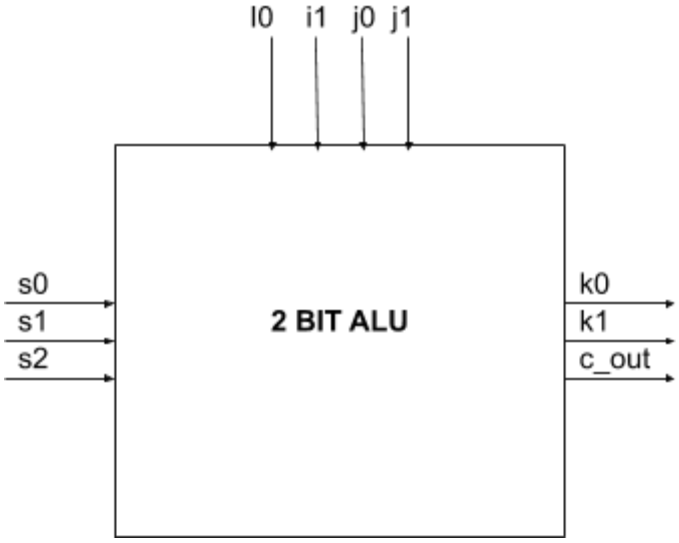

Figure 1. Block Diagram of the 2-bit ALU

The ALU is illustrated in the figure above together with its inputs and outputs. The ALU should be able to do the following operations:

- Arithmetic operations
  - Addition - add the two inputs i.e. $k = i + j$
  - Subtraction in 1's Complement - subtract the second input from the first input i.e. $k = i - j$ using 1's complement representation of negative numbers
  - Subtraction in 2's Complement - subtract the second input from the first input i.e. $k = i - j$ using 2's complement representation of negative numbers
  - Increment - increment the first input by 1 (ignore second input) i.e $k = i + 1$
- Logical operations
  - AND - the bitwise AND operation i.e. $k = i \; AND \; j$
  - OR - the bitwise OR operation i.e. $k = i \; OR \; j$

- XOR - the bitwise XOR operation i.e. $k = i\ XOR\ j$
- NOT - flip the bits of the first input (ignore second input) i.e. $k = \tilde{}i$

Table 1. A few sample inputs and outputs

s_0	s_1	s_2	ACTION	i_0	i_1	j_0	j_1	k_0	k_1	c_{out}
0	0	0	AND	1	0	1	1	1	0	0
0	0	1	OR	1	0	1	1	1	1	0
0	1	0	XOR	1	0	1	1	0	1	0
0	1	1	NOT	1	0	1	1	0	1	0
1	0	0	ADD	1	0	1	1	0	1	1
1	0	1	SUB (1)	1	0	1	1	1	0	0
1	1	0	SUB (2)	1	0	1	1	1	1	0
1	1	1	INC	1	0	1	1	1	1	0

The table above shows a few sample test cases for the ALU. Take note that above is not a comprehensive list; thus, take note of other possible input and output cases. Use the assigned selection lines above for each operation.

CMSC 132: Computer Architecture
First Semester, AY 2020-2021
Exercise 03: Combinational Circuits
Prepared by: Clinton Poserio

TASKS:

1. Design the "inner workings" of the ALU; that is, provide the circuit diagram for every component needed for your ALU and its wiring connections (Please minimize wire intersections in your diagram; put necessary marks e.g. arrow heads to easily follow and trace the signal flow). Format your design using any editor you are comfortable with.
2. Create your ALU model, together with its components, using VHDL. You may implement your units using either behavioral or structural approach, with the following conditions:
    a. If implemented in behavioral approach, provide truth tables for any formula you used. Include proofs and derivations if necessary.
    b. If implemented in structural approach, provide design diagrams for each component.

    Components created for atomic gates are exempted from any conditions mentioned above.
3. Create a test bench for your ALU that should test all supported operations. You may opt to create multiple test benches to easily check each operation.

SUBMISSION:

- **Deadline** is the day before the next laboratory meeting.
- **Submission format:**
    - Submit a single **.zip** file named Surname1_Surname2Ex03.zip in our Google Classroom **before the deadline**. (Yep, you read it right! You'll work on pairs.)
    - The file must contain the following:
        - *model* - folder containing your files for your ALU.
        - *test* - folder containing your test bench files.
        - *others* - a folder containing either your truth tables or component diagrams. You may use scanned documents for this.
        - *design.jpg* - your ALU diagram
        - README - a text file containing your names, a brief background on your individual works, and citations to the references you have used.

The exercise is worth **30 points**.

HINTS: You only need several gates, muxes, and adders to create a basic ALU. #glhf

REFERENCES you may use**:**

Mano, M. M. (2002). *Digital design*. EBSCO Publishing, Inc..

Areibi, S. (2001). A first course in digital design using VHDL and programmable logic. In *Frontiers in Education Conference, 2001. 31st Annual* (Vol. 1, pp. TIC-19). IEEE.

`