Design of Digital Systems: Lab Assignment 3

The objective of this assignment is to implement an arithmetic logic unit (ALU).

Background

Many of the operations performed by computers can be broken down into arithmetic or logic operations. It would be difficult, if not impossible, to have to re-implement such operations in software every time a program was written. Thus a dedicated piece of hardware, known as an ALU, was created and included in almost every processor since its inception.

Program Specification

The design must meet the following specifications and use all components listed at least once. Some might be used multiple times.

ALU

As shown in Figure 1, the ALU (name it `alu`) is a 6-bit arithmetic unit that implements addition, multiplication, logical, and shifting operations on its operand and outputs the result based on the select statement as shown in Table 1. It has the following entity:

- Inputs:
  - `sel` (4 bits): Selects the desired result.
  - `a` (6 bits): The first operand of the arithmetic operation.
  - `b` (6 bits): The second operand of the arithmetic operation.
- Outputs:
  - `r` (6 bits): The result of the arithmetic computation.

Notes

- The multiplexer can be written directly in the ALU unit. You do not need to create a separate unit for the multiplexer.

Table 1: ALU operations

Unit	Operation	Select	Description
Adder	Addition	`0000`	`a + b` without carry
	Addition's carry	`0001`	carry of a+b
	Subtraction	`0010`	`a-b` without carry
	Subtraction's borrow	`0011`	borrow of `a-b`
Multiplier	Multiplication low bits	`01X0`	Low `6` bits of `a`×`b`
	Multiplication high bits	`01X1`	High `6` bits of `a`×`b`
Logic Unit	Bitwise NOT	`1000`	NOT `a`
	Bitwise AND	`1001`	`a` AND `b`
	Bitwise OR	`1010`	`a` OR `b`
	Bitwise XOR	`1011`	a XOR `b`
Shifter	Shift left logical	`110X`	a << b
	Shift Right Logical	`1110`	a >> b
	Shift Right Artihmetic	`1111`	a >> b sign extended

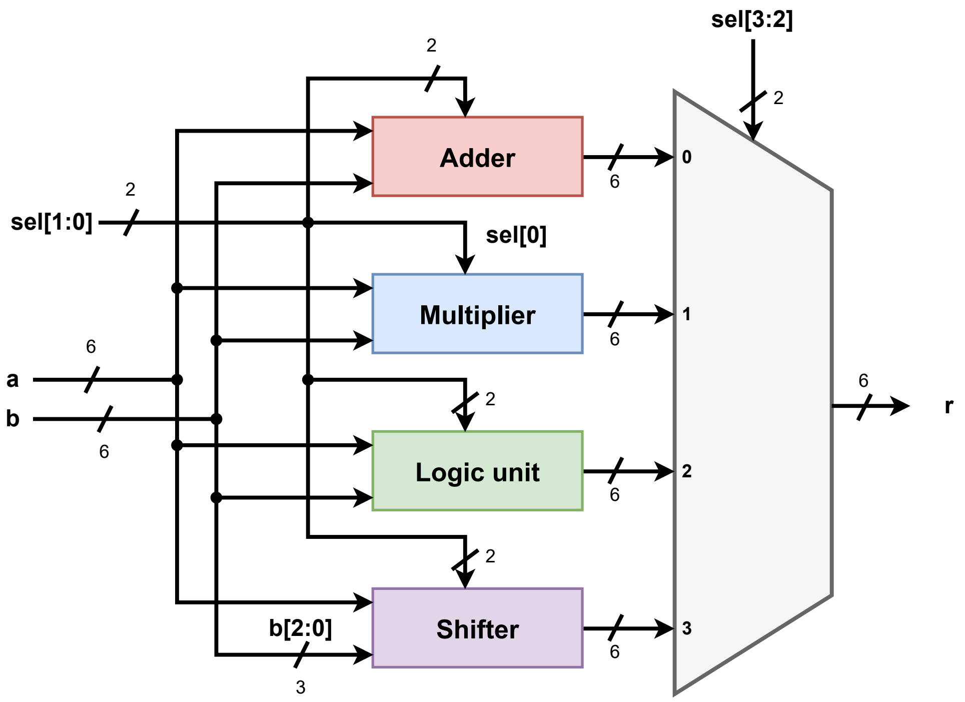

Figure 1: Arithmetic Logic Unit (ALU)

Adder Unit

The adder unit (name it `adder`) takes two 6-bit inputs a and b and outputs a 6-bit output r based on the 2-bit input `sel`.

- If `sel` is 00, then r is the sum of a and b, aka $a + b$, without the carry bit.

- If `sel` is 01, then r is the carry of the sum of a and b. The carry is 1 bit and will need to be extended to 6 bits by prepending 5 bits of zero.
- If `sel` is 10, then r is the difference of a and b, aka $a - b$, without the borrow bit.
- If `sel` is 11, then r is the borrow of the difference of a and b. The borrow is 1 bit and will need to be extended to 6 bits by prepending 5 bits of zero.

Hints

- The method to implement the adder is up to the student.
- The difference must be implemented by performing $a + (-b)$. This means that you will need to compute two's complement of b (invert b and add 1) and before adding it to a. The borrow becomes the carry of $a + (-b)$.

Multiplier unit

The multiplier unit (name it `mult`) takes two 6-bit inputs a and b and outputs a 6-bit output r based on the 1-bit input `sel`.

- If `sel` is 0, then r is the low 6 bits of $a \times b$
- If `sel` is 1, then r is the high 6 bits of $a \times b$.

Hints

- The method to implement the multiplier is up to the student.
- When performing multiplication of n-bit integer by n-bit integer, the output is a $2n$-bit integer.

Logic unit

The logic unit (name it `logic_unit`) takes two 6-bit inputs a and b and outputs a 6-bit output r based on the 2-bit input `sel`.

- If `sel` is 00, then r is NOT a.
- If `sel` is 01, then r is a AND b.
- If `sel` is 10, then r is a OR b.
- If `sel` is 11, then r is a XOR b.

Hints

- All operations are bitwise operations not logical operations even though the unit is called logic unit.

Shifter unit

The shifter unit (name it `shifter`) takes one 6-bit input a and one 3-bit input b and outputs a 6-bit output r based on the 2-bit input `sel`.

- If `sel` is 00 or 01, then r is a shifted left by b bits. The bits shifted in are 0s.
- If `sel` is 10, then r is a shifted right by b bits. The bits shifted in are 0s. This is commonly known as unsigned or logical shift right.
- If `sel` is 11, then r is a shifted right by b bits. The bits shifted in are the most significant bit of a. This is commonly known as signed or arithmetic shift right.

Hints

- The method to implement the shifter unit is up to the student.

Testbench

To ensure that the `ALU` is working properly, write a testbench that tests all 16 possible outputs (based on the 4-bit select) for the following operands:

- $a = 4_{10} = 000100_2$ and $b = 2_{10} = 000010_2$
- $a = 49_{10} = 110001_2$ and $b = 50_{10} = 110010_2$
- $a = 63_{10} = 111111_2$ and $b = 63_{10} = 111111_2$

Report

- Do NOT collect timing results for this lab
- For the demo, test all the scenarios covered in the testbench

FPGA port map

IO	**FPGA port**
sel	Switches 15-12
a	Switches 11-6
b	Switches 5-0
r	LEDs 5-0