ESKISEHIR TECHNICAL UNIVERSITY
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

LAB-2
32-BIT MIPS ALU DESIGN

1 Introduction

In this second lab, you will implement a 32-bit MIPS ALU. ALU is an example of a combinational circuit inside the microprocessor and is one of the main component inside a microprocessor that is responsible for performing arithmetic and logic operations such as addition, subtraction, logical AND, and logical OR.

2 ALU Functions

Arithmetic circuits are the central building blocks of computers. Computers and digital logic perform many arithmetic functions. This section describes hardware implementations for all of these operations.

A simple ALU has two inputs for the operands, one input for a control signal that selects the function to perform, and one output for the function result. Following figure is commonly used to represent this simple ALU.

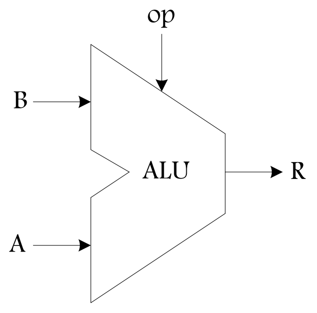

Figure 1: A simple ALU

For this lab you have to implement a 32-bit ALU. The ALU includes 4 internal components: add/sub, comparator, logical unit and shift unit. In the following table, available functions and corresponding encoding with Opcode are given.

The 6-bit Opcode is a control signal which can select one of the above functions. As you may notice that, the 2 most significant bits can select the operation type (e.g. Add / Sub, Comparison, Logical, Shift / Rotate). Opcode(3) bit is used to activate the subtraction mode of the Add/Sub unit. Notice that the subtraction mode is always activated for the comparisons and ignored for logical and shift / rotate unit. The remaining bits can select a specific operation of the units. The following figure shows the connections between internal components inside the ALU.

Operation	Type	Opcode
$A + B$	Add / Sub	000XXX
$A - B$		001XXX
$A \geq B$ (signed)	Comparison	011001
$A < B$ (signed)		011010
$A \neq B$		011011
$A = B$		011100
$A \geq B$ (unsigned)		011101
$A < B$ (unsigned)		011110
A *nor* B	Logical	10XX00
A *and* B		10XX01
A *or* B		10XX10
A *xor* B		10XX11
A *rol* B	Shift / Rotate	11X000
A *ror* B		11X001
A *sll* B		11X010
A *srl* B		11X011
A *sra* B		11X111

Table 1: ALU Function table

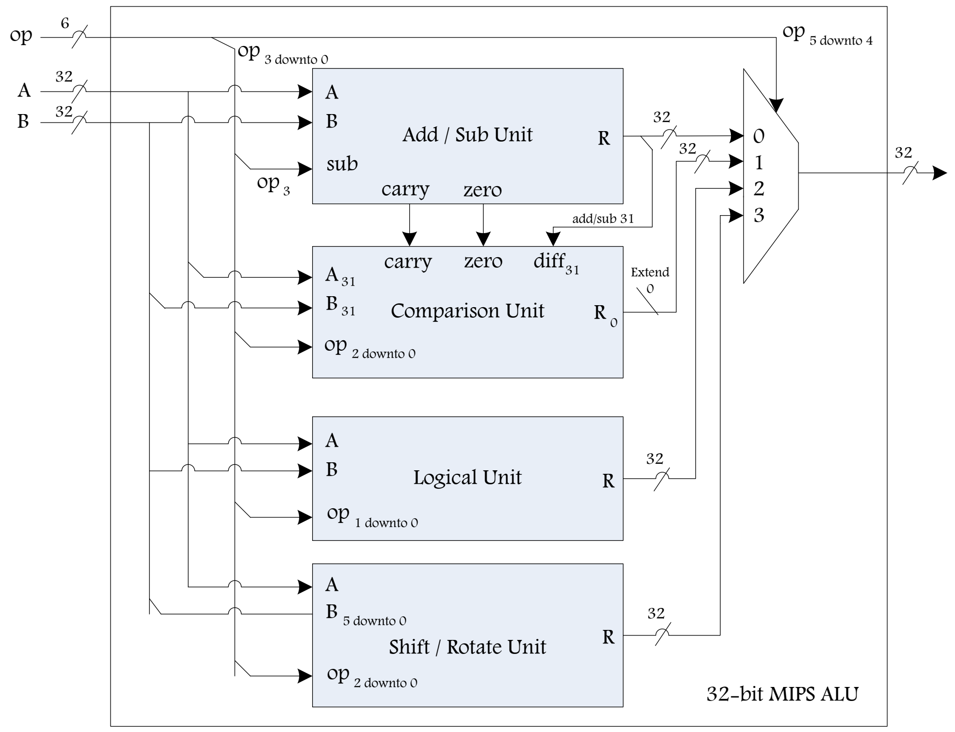

Figure 2: A 32-bit MIPS ALU

2.1 Add/Sub Unit

The Add/Sub unit performs 32-bit additions and subtractions. The sub input signal activates the subtraction mode. The carry output signal gives the carry out from the adder inside the unit. The zero output signal indicates whether the result is equal to 0.

When the subtraction mode is activated, we have to replace the B operand by its two's complement. According to the sub signal value, the two's complement of B input can be taken as followed. When the sub signal is high, B input is inverted; otherwise it keeps its original value. In case of a subtraction mode, Increment by 1 due to 2's complement can be done by connecting the sub signal directly to the carry in of the adder, then we will have $A + \bar{B} + 1$ which is equivalent to $A - B$.

2.2 Comparison Unit

The Comparison unit performs equal, not equal, signed or unsigned greater or equal and less than comparisons. It uses the subtraction result to compare the 2 operands. Therefore the Add/Sub unit has to be in subtraction mode to calculate the difference between these operands.

The 3-bit input op can select the type of comparison. The 1-bit output R gives the result of the comparison as 0=false, 1=true. In order to be compatible with 32-bit multiplexer inside the ALU, The result of the comparison unit should be extended to 32-bit. The 31 most significant bit is set to 0. The zero, carry, A_{31}, B_{31}, and $diff_{31}$ input signals are used to do the comparison.

2.2.1 Equal and not equal

The equal and not equal comparisons result is directly driven by the zero input signal. If A equals B, then subtraction result will be zero.

2.2.2 Unsigned greater or equal and less than

The greater or equal and less than unsigned comparisons depend only on the carry signal. If the carry is high, then A is greater or equal to B; otherwise A is less than B. The result of these comparisons is driven by carry input signal.

2.2.3 Signed greater or equal and less than

For these two signed comparisons, we need a little more logic to calculate the result. We have A31, B31, and diff31 the most significant bits of the ALU inputs A and B, and of the subtraction result.

If A is positive and B is negative, quickly we can say that A ≥ B. If subtraction result is positive and A and B signs are the same, we have also that A ≥ B. In any other case, A is less than B.

Operation	Opcode
$A \geq B$ (signed)	001
$A < B$ (signed)	010
$A \neq B$	011
$A = B$	100
$A \geq B$ (unsigned)	101
$A < B$ (unsigned)	110

Table 2: Comparison Unit functions

2.3 Logical Unit

The Logical Unit performs *and, or, nor* and *xor* logical bitwise operations. The op 2-bit signal can select the operation according to the following table.

Operation	Opcode
A *nor* B	00
A *and* B	01
A *or* B	10
A *xor* B	11

Table 3: Logic Unit functions

2.4 Shift / Rotate Unit

The Shift / Rotate unit performs 5 different shift operations on the operand A by B bits. The input B defines by how many position we should shift A. Only the 5 least significant bits of B are used in this shift unit. The 3-bit input op selects the operation according to the following table.

Operation	Opcode
A *rol* B	000
A *ror* B	001
A *sll* B	010
A *srl* B	011
A *sra* B	111

Table 4: Shift Unit functions

When shift operations are performed, operand A is shifted to the left or right by a number of position defined by B. The bits shifted out are dismissed. For logical shifts (e.g. sll, srl) zeros are shifted in. In right arithmetic shifts (sra) the sign bit is replicated to keep the sign of the operand. For the rotation operations (rol and ror) the bits shifted out are reinjected at the other end of the operand. As you may notice that, A rotated left by B is equivalent to A rotated right by (-B).

3 Procedure

In this second lab, you will design a pure combinational system to perform different arithmetic and logical operations.

- Write the VHDL description of the 32-bit MIPS ALU whose details are described above.
- You should simulate your design using ISE or another VHDL simulator to prove the correctness of your design. (Prepare a short report with the VHDL codes and the simulation results after lab session)