Philadelphia University

Faculty of Engineering

Microprocessors project

Project title: Designing 8 bits ALU

By:

المنذر محمدمهدي الدهني

201610825

رنيم عبدالعزيز غليون

201710671

Introduction:

ALU (Arithmetic Logic Unit)

A critical component of the microprocessor, the core component of central processing unit. ALU comprises the combinational logic
That implements logic operations such as AND and OR, and arithmetic operations such as Addition, Subtraction, and Multiplication.

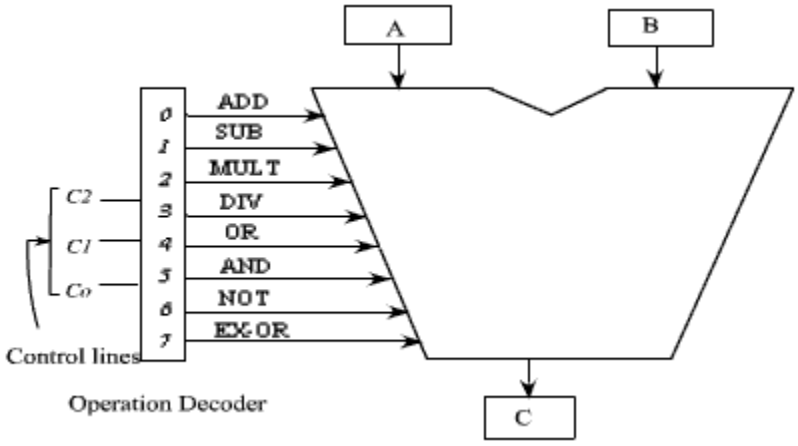

Figure 1

.

Furthermore.. The ALU takes input as the data to be operated on (called operands) and a code, from the control unit, indicating which operation to perform. The main operation of any processor is mainly fetching, decoding and executing an instructions. The fetching of instructions is perform by instruction fetch unit, the decoding of instructions is perform by decode unit (or control unit) which generate appropriate control signals for ALU to carry-out the operation of instruction. These control signals refer to as selection lines for ALU to select particular operation.

The block diagram below shows the ALU which we will be working on :

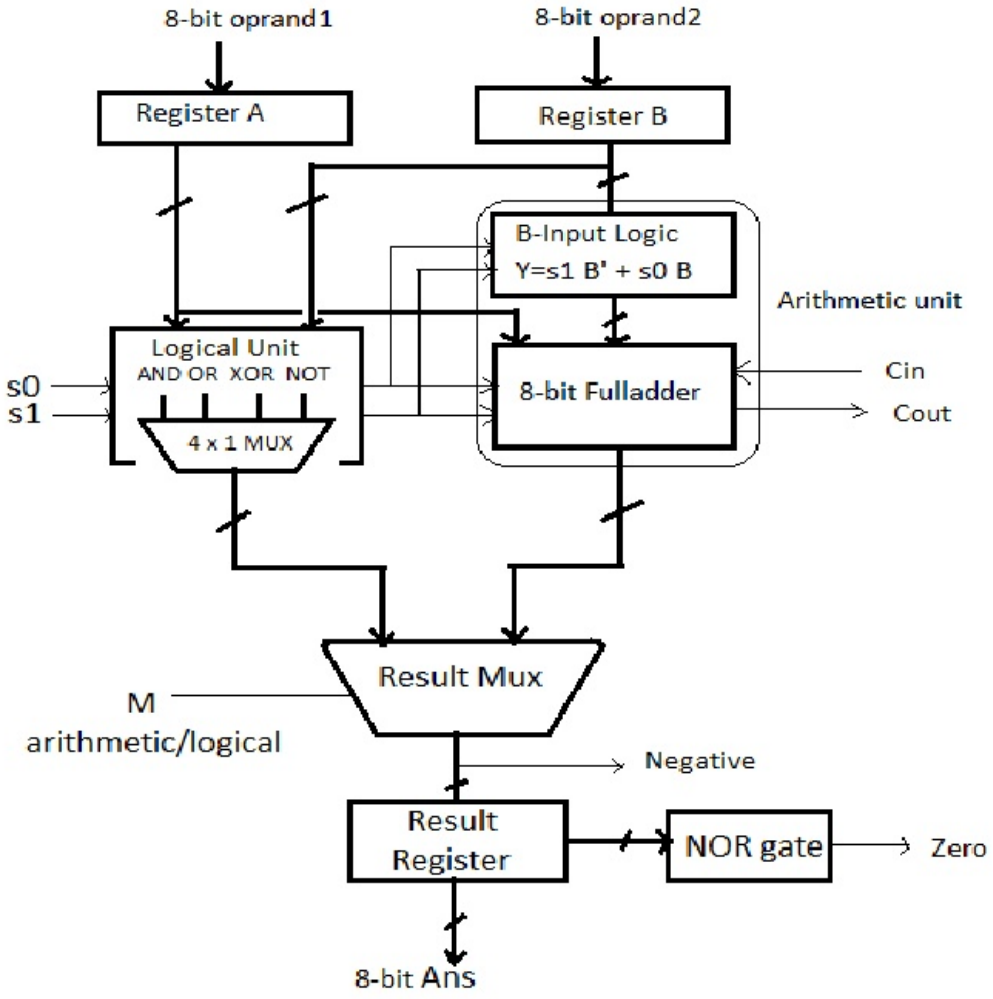

Figure 2

1-Arithmetic Unit:

Arithmetic operations performed are 8-bit addition and subtraction.
ALU also calculate 1's and 2's complement for the 8-bit input for subtraction.

Arithmetic unit contains two blocks:
1) B-input logic
2) 8-bit Full-adder.

B-input logic is used to obtain 2's complement of input B when the subtraction is going to perform. The B-input logic reduces the complexity of the circuit.
And Full-adder is used to get addition of two operand with input carry and 8-bit sum and 1-bit carry.

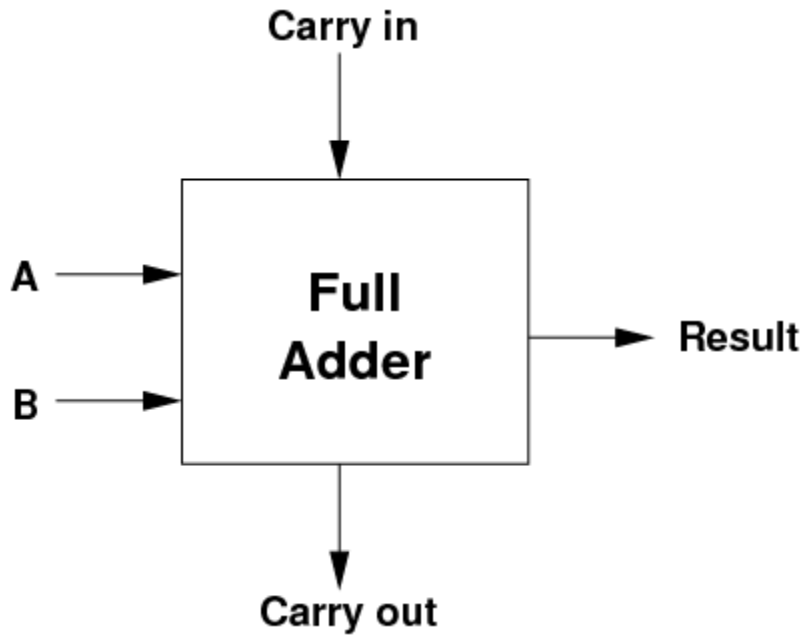

Figure 3

Internally, here is what a full adder looks like, just 5 logic gates

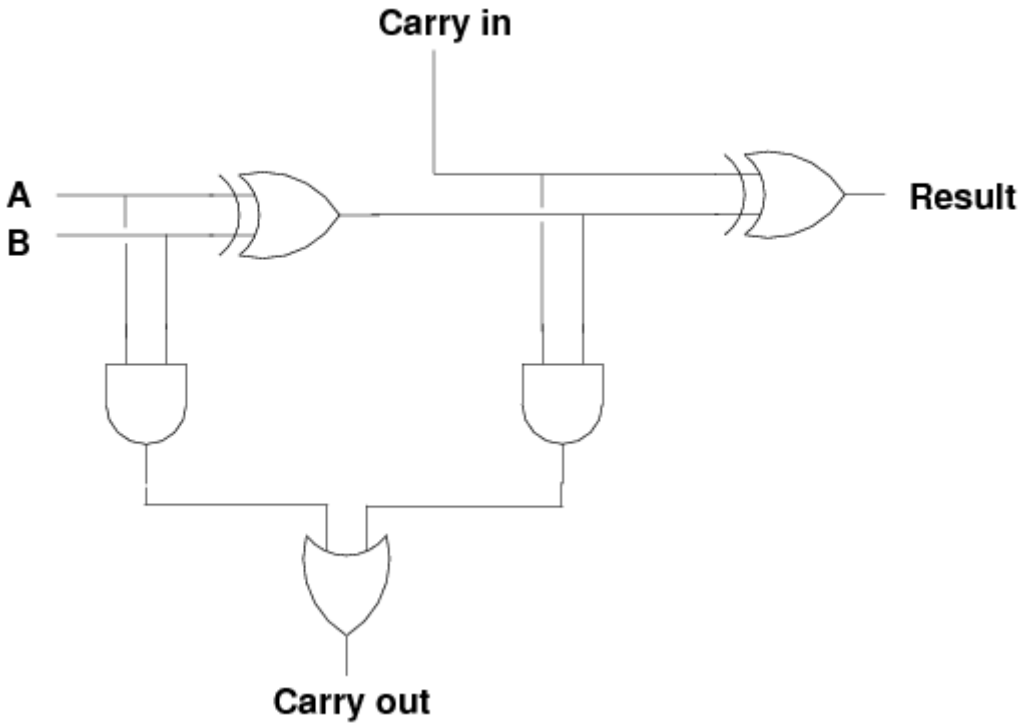

Figure 4

And here is the truth table for the full adder:

Cin	A	B	Result	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

What about Subtraction?
To save time,space and money we can use some maths with our full adder
Instead of using a subtractor
NOW according to math 3–2=3+(–2)

$$A - B = A + (-B) = A + \sim B + 1$$

So basically we can negate one of the inputs using our full adder.
twos complement binary integer is perferct for this operation ,just invert every bit in the number, then add 1.
we can NOT gate for each bit in B to do that.

But now we need to do $A + \sim B + 1$.
How can we do this?
Easy!
Set the initial carry-in to 1 instead of 0, thus adding an extra 1 to the sum.
And instead of using NOT gates, we will use XOR gates.

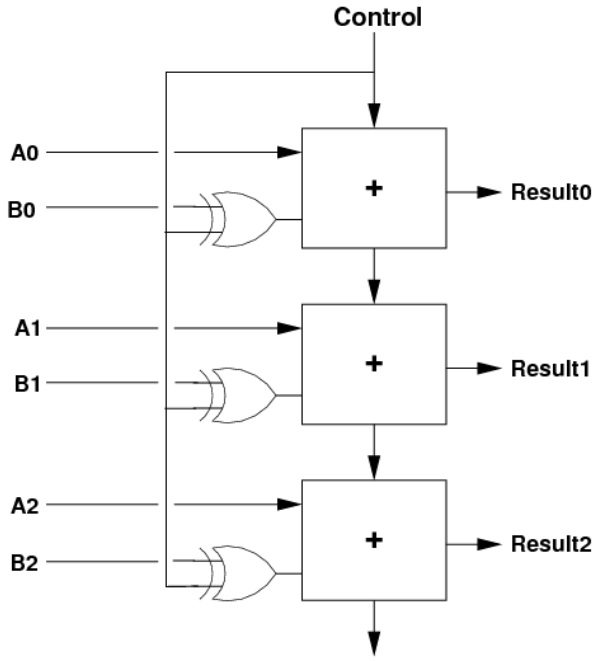

Figure 5

2-Logical Unit :

Logical operations performed are AND, OR, XOR and NOT.
4 to1 MUX selects the logic operations based on the select lines in the logic unit.
Both arithmetic and logical unit are performed the operation in parallel. Finally a 2 to1 MUX selects between arithmetic and logic unit.
Zero flag is obtain by applying all the lines of result to Nor gate. And Negative flag is obtain by taking the 7 th bit of the result

Inputs and outputs of an ALU!

Inputs are:
1. Two 8-bit Operands A &B
2. Operation Selection lines : S0, S1 : Cin (carry-in)
3. Result selection line: M = 1 : Arithmetic result = 0 : Logical result

Outputs are:
1. One 8-bit result operand
2. Z (Zero), Cout (carry-out),N(negative flags) and overflow.

Remember:
Carry: was there a carry in the most-significant bit which we could not output in the given number of bits .
Overflow: does the sign of the output differ from the inputs, indicating (for example) that a sum of two positive numbers has overflowed and is now a negative result!

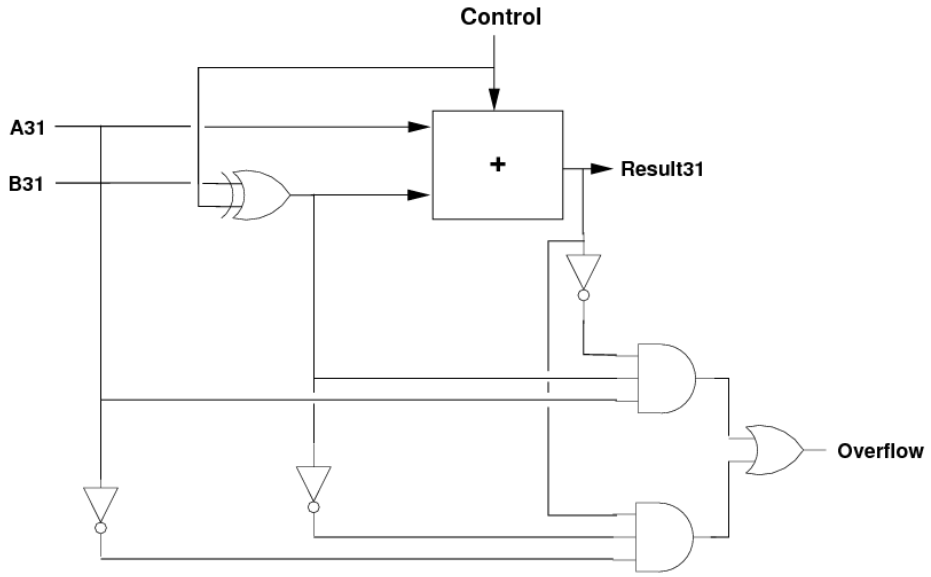

Figure 6

Negative Output:

When is the result negative? When its most-significant bit is 1.
We can wire this bit directly out of the ALU, so that it indicates if the result is negative.

Zero Output:

This is only true if all of the bits of the result are zero. We can do this for the logical and the maths units.
To do this, we can use OR gate followed by NOT gate:

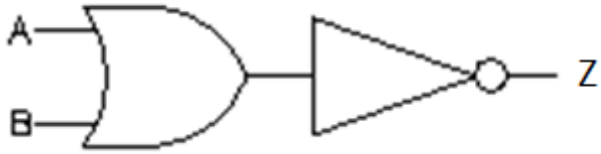

Figure 7

The OR gate outputs a 0 only if all input bits are 0. If any input bit is a 1, the OR's output is a 1.
The NOT gate simply inverts this, giving the result:

if any result bit is on, the output is 0, i.e. not zero.
if all result bits are off, the output is 1, i.e. zero.

Putting everything together..

We now have three main units:

- a 8-bit bitwise AND unit,
- a 8-bit bitwise OR unit, and
- a 8-bit ADD/SUBTRACT unit with a control line.
- the logic to output carry, overflow, zero and negative.

We can pass the inputs A and B to all three units, and perform all three operations in parallel:

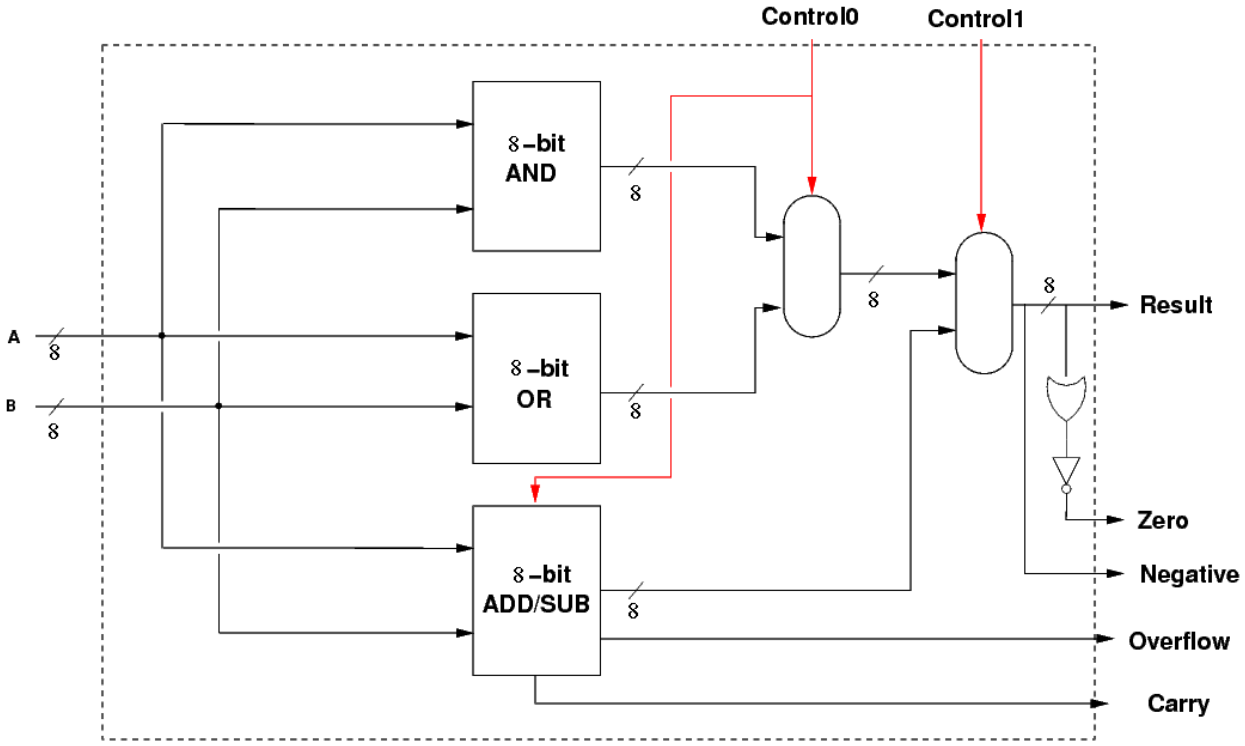

Figure 8

After putting everything together using Logisim simulation program
We will get our 8 bit ALU

Here is a link to download Logisim:
https://sourceforge.net/projects/circuit/

Here is a link to our Logisim ALU file uploaded to Google drive:
https://drive.google.com/open?id=1iLJVidLOP7HybB0Nw5EOrX3X6WO8OFeX

Download the file
Open Logisim
File >> open >>choose the downloaded file

It should look like this

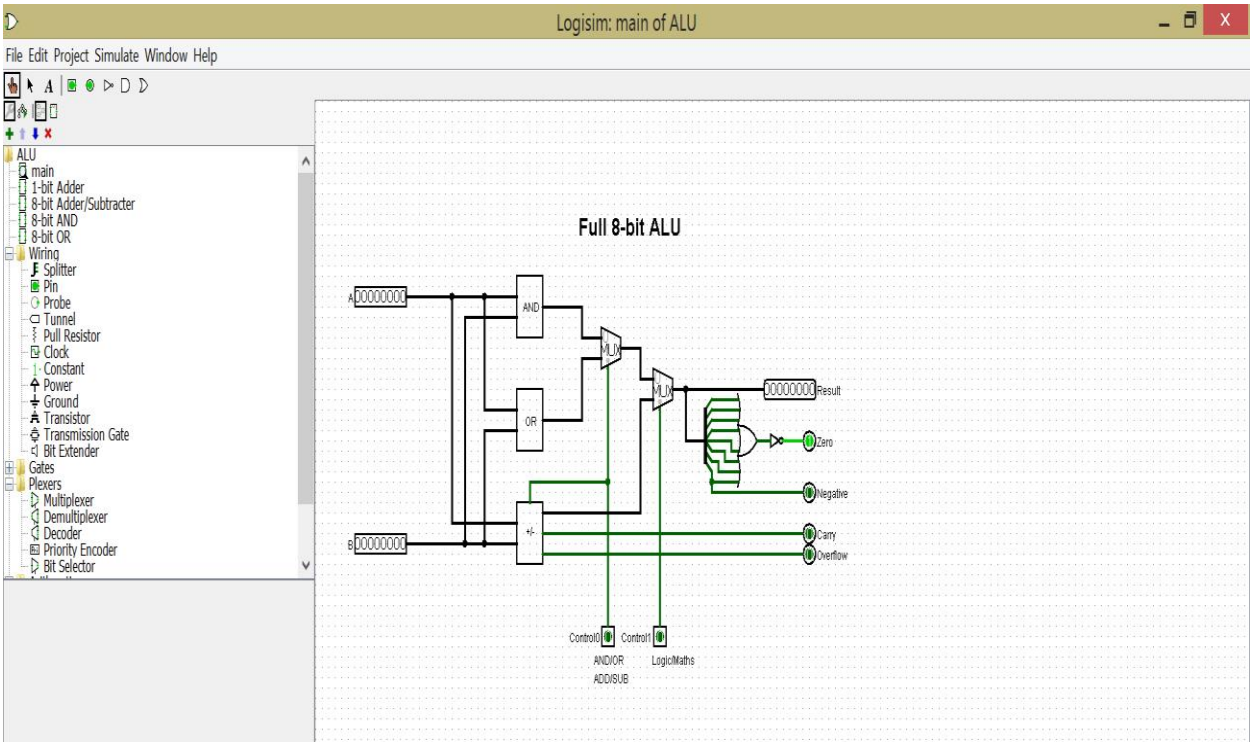

Figure 9