ALU Implementation

Computer Architecture

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

CS3501 - 2023I
PROF.: JGONZALEZ@UTEC.EDU.PE
SRC: HARRIS, HARRIS - DIGITAL DESIGN AND COMPUTER ARCHITECTURE

Executive Summary

- Motivation: We need a working ALU for the microprocessor design.
- Problem: We need to put together and test the building blocks for the microprocessor design.
- Overview:
  - ALU implementation and testing.
  - Introduce Project Logistics.
- Conclusion: We can implement an ALU using logic building blocks, and we can evaluate its design. We are ready to work in our microprocessor!

Outline

ALU Implementation

Course Logistics

Conclusions

Outline

ALU Implementation

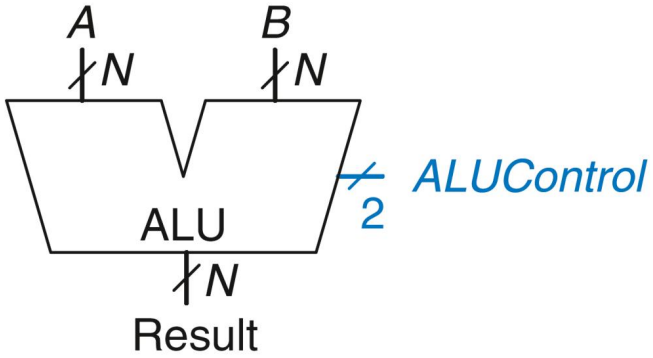
Course Logistics

Conclusions

Recall: Arithmetic Logic Unit

- **ALU is the heart of the processor.**
- **ALU should perform at least:**
  - Addition
  - Subtraction
  - AND
  - OR

$ALUControl_{1:0}$	Function
00	Add
01	Subtract
10	AND
11	OR



Example: Perform *A* + *B*

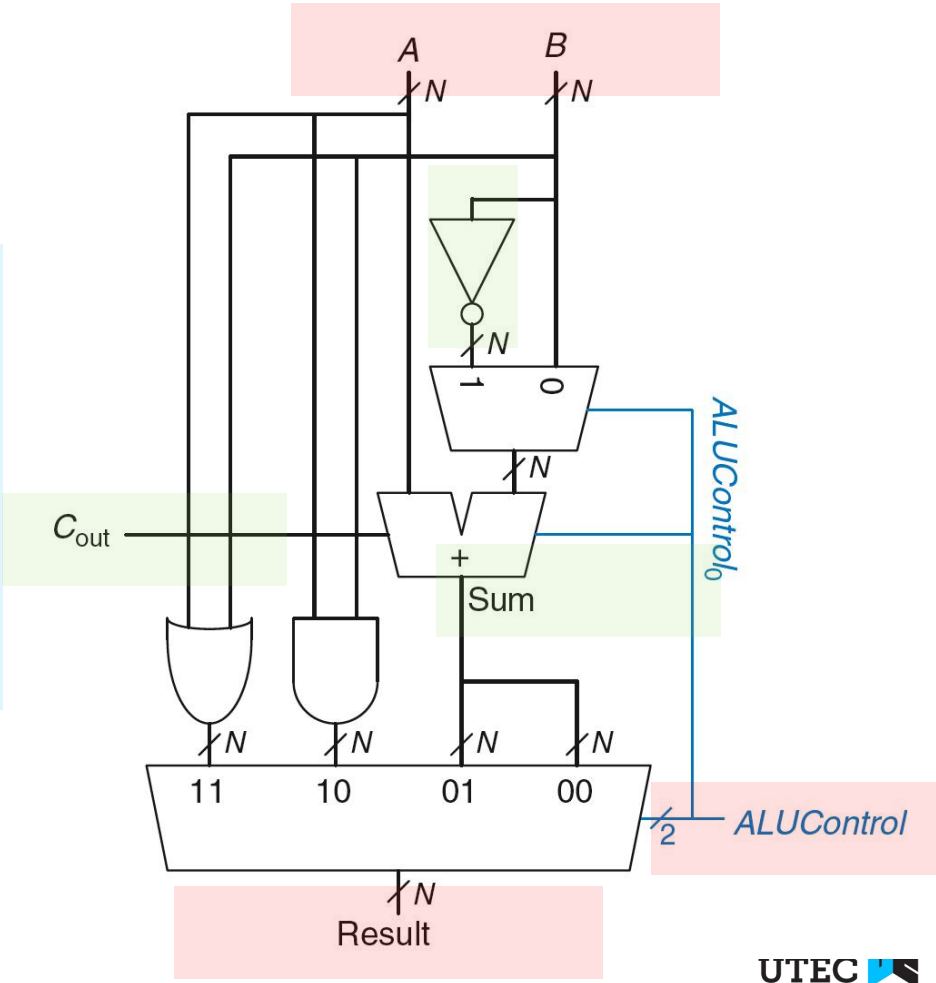
ALUControl = 00

Result = A + B

Inputs and Outputs

```
//32-bit ALU for ARM processor
module alu(input  [31:0] a, b,
           input  [1:0]  ALUControl,
           output reg [31:0] Result,
           output wire [3:0]  ALUFlags);

  wire        neg, zero, carry, overflow;
  wire [31:0] condinvb;
  wire [32:0] sum;
```



ALU OR

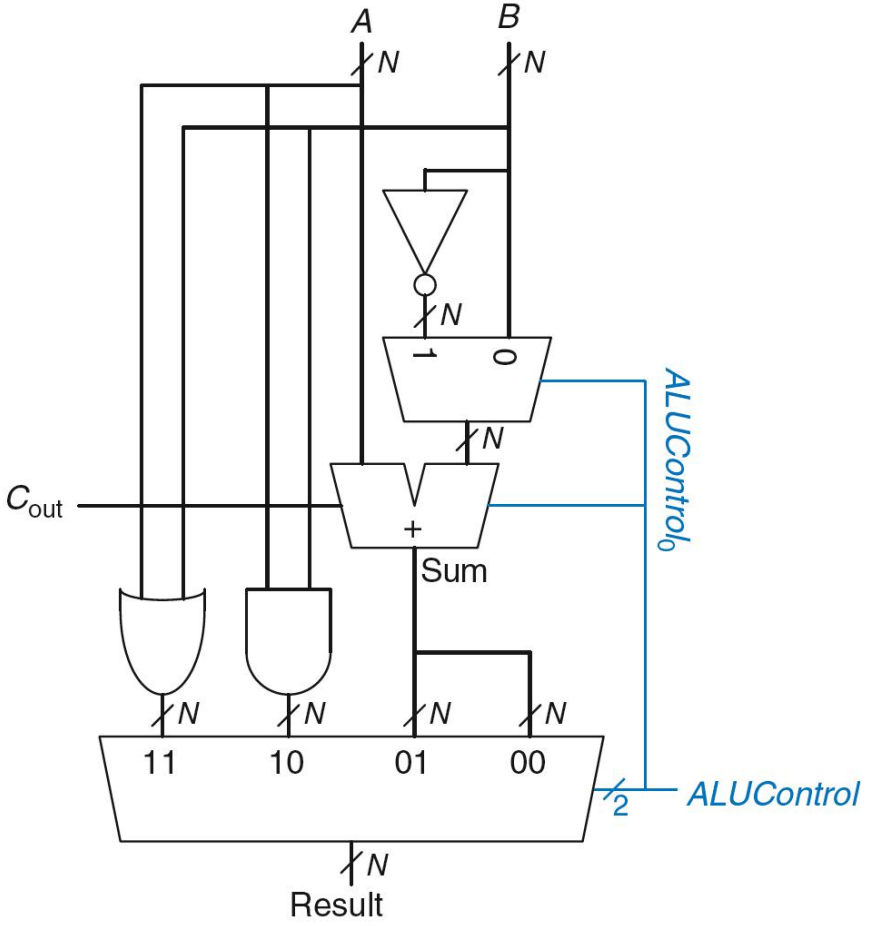
$ALUControl_{1:0}$	Function
00	Add
01	Subtract
10	AND
11	OR

Example: Perform *A* OR *B*

$ALUControl_{1:0}$ = 11

Mux selects output of OR gate as *Result*, so

Result* = *A* OR *B



ALU ADD

$ALUControl_{1:0}$	Function
00	Add
01	Subtract
10	AND
11	OR

Example: Perform *A* + *B*

$ALUControl_{1:0}$ = 00

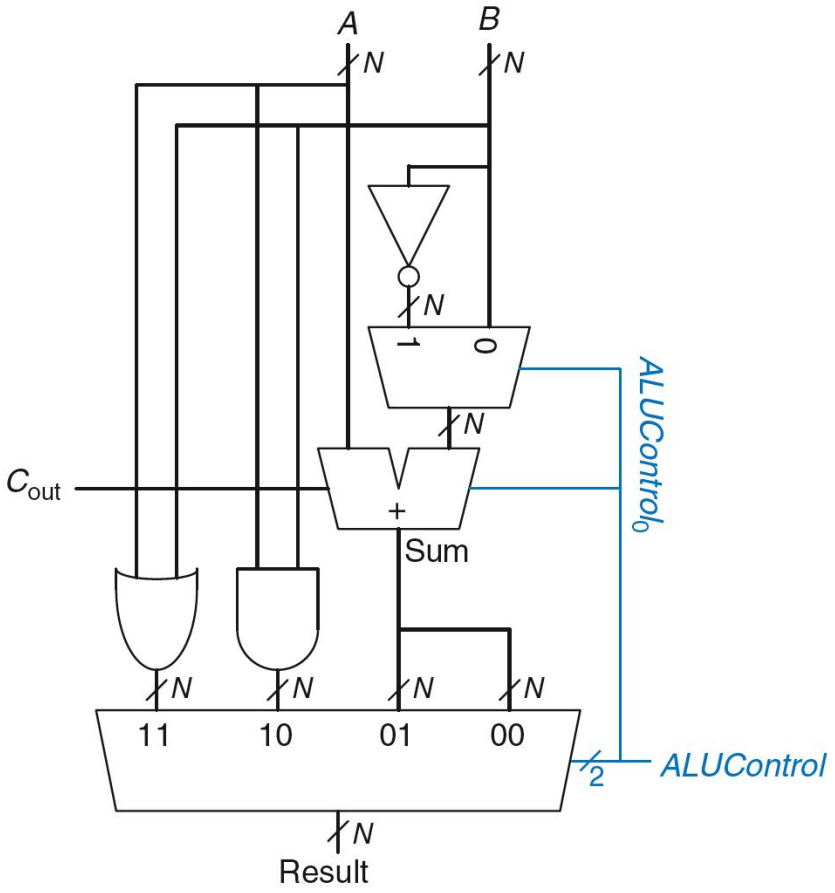
$ALUControl_0$ = 0, so:

Cin to adder = 0

2nd input to adder is *B*

Mux selects *Sum* as *Result*, so

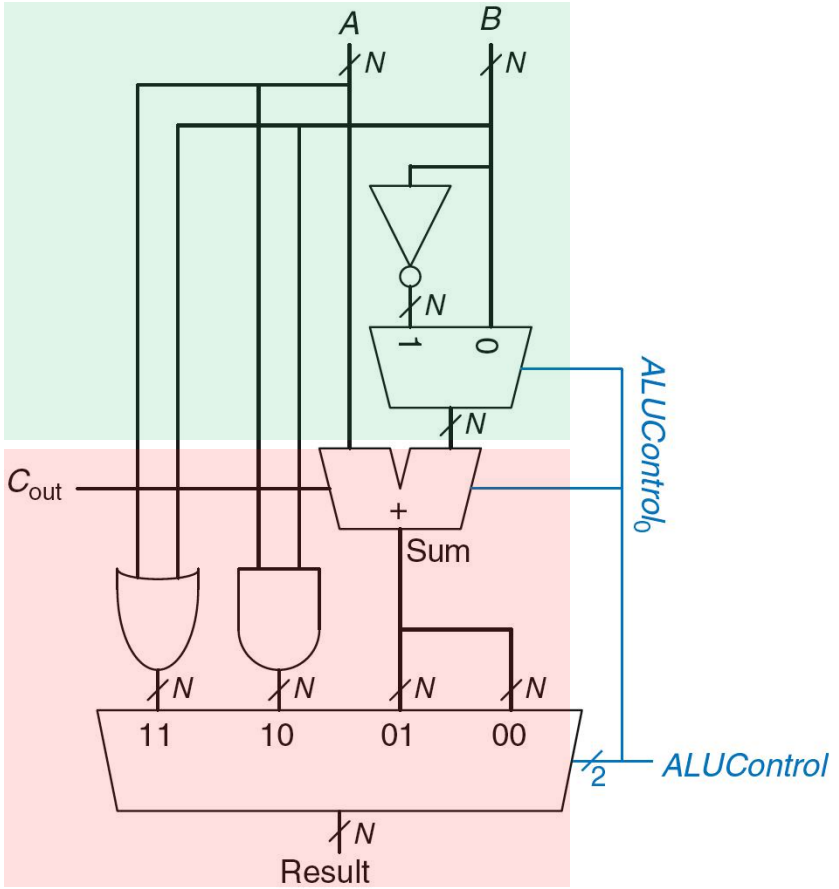
Result* = *A* + *B



Operation Logic

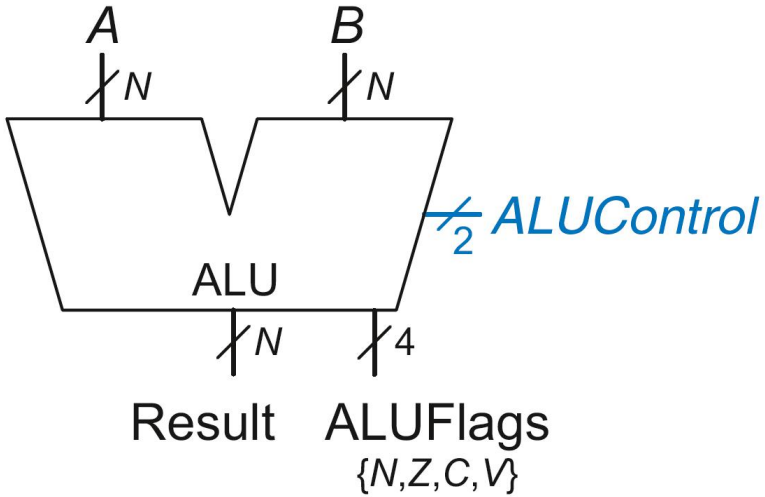
```
assign condinvb = ALUControl[0] ? ~b : b;
assign sum = a + condinvb + ALUControl[0];

always @(*)
  begin
    casex (ALUControl[1:0])
      2'b0?: Result = sum;
      2'b10: Result = a & b;
      2'b11: Result = a | b;
    endcase
  end
```

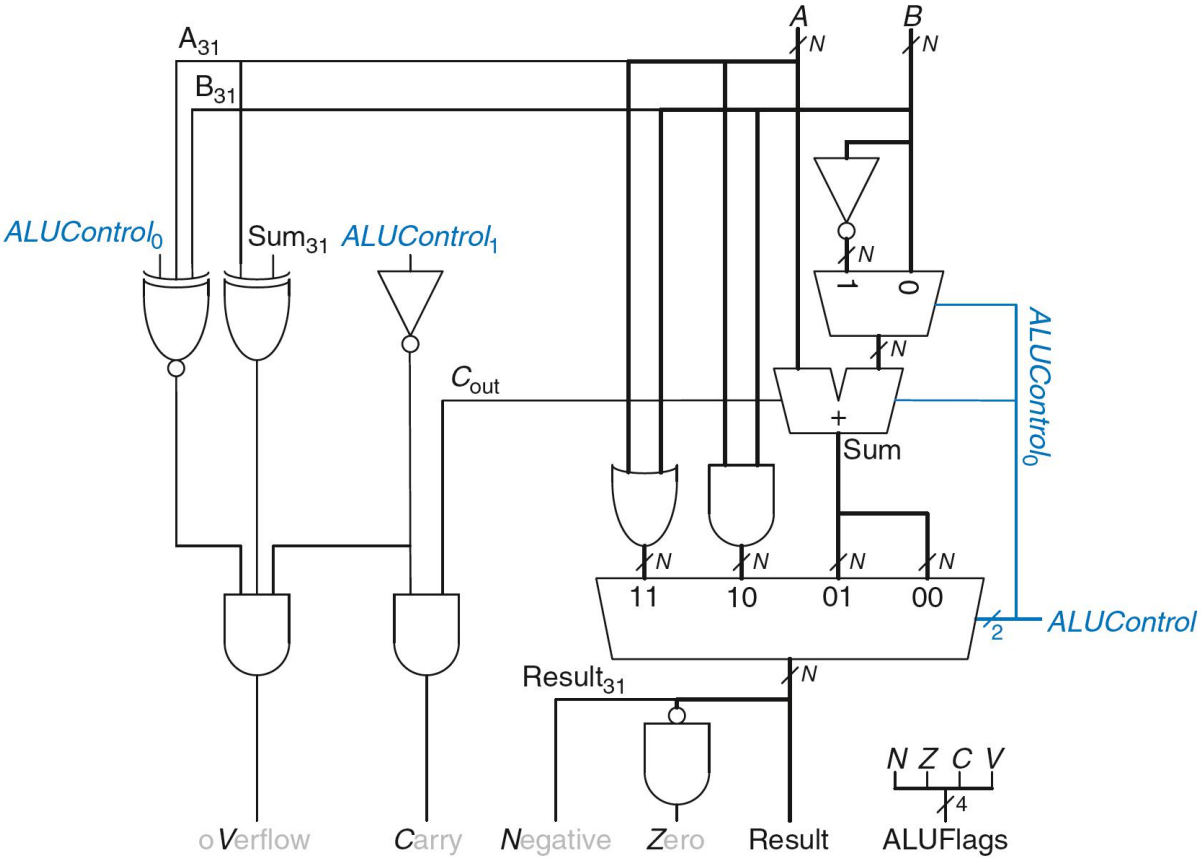


ALU with Status Flags

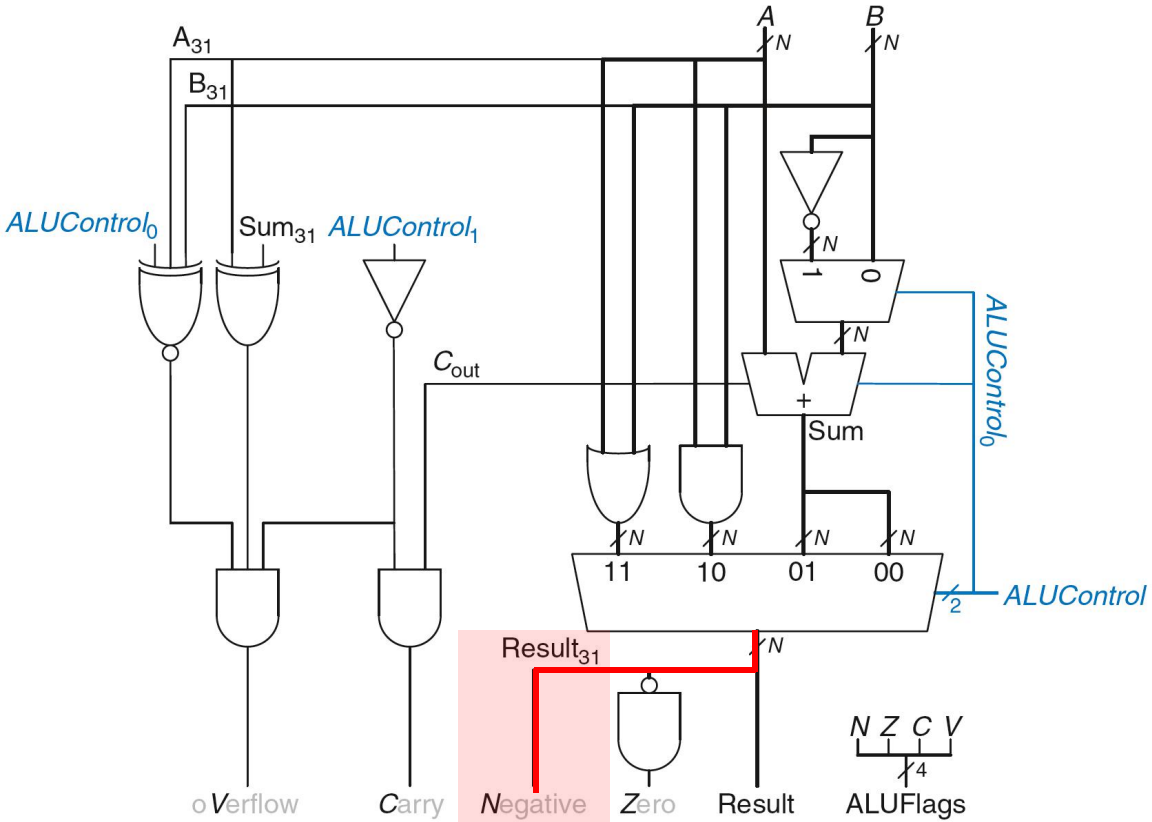
Flag	Description
N	*Result* is **N**egative
Z	*Result* is **Z**ero
C	Adder produces **C**arry out
V	Adder o**V**erflowed



ALU with Status Flags



Flag Negative (N)

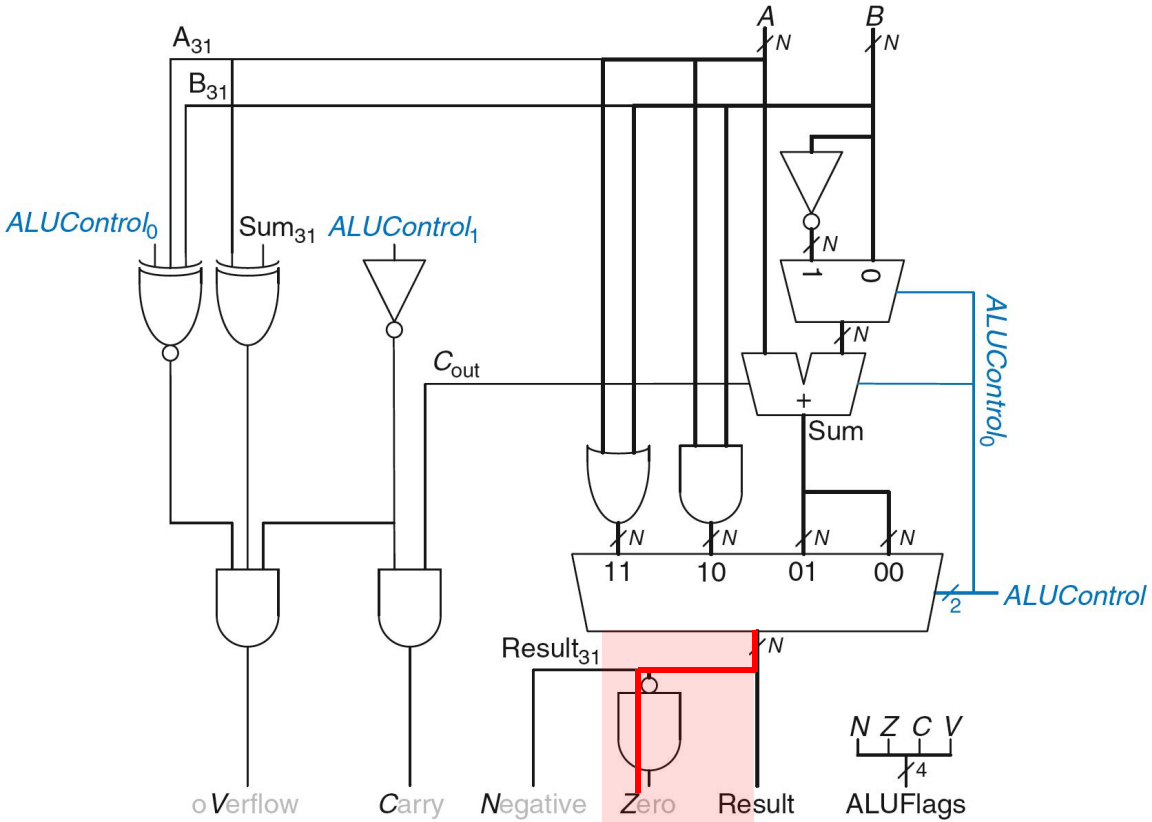


***N* = 1** if:

Result is **negative**

So, *N* is connected to most significant bit (msb) of *Result*

Flag Zero (Z)



Z = 1 if:

all of the bits of *Result* are 0

E.g.: Result = 0101

~Result = 1010

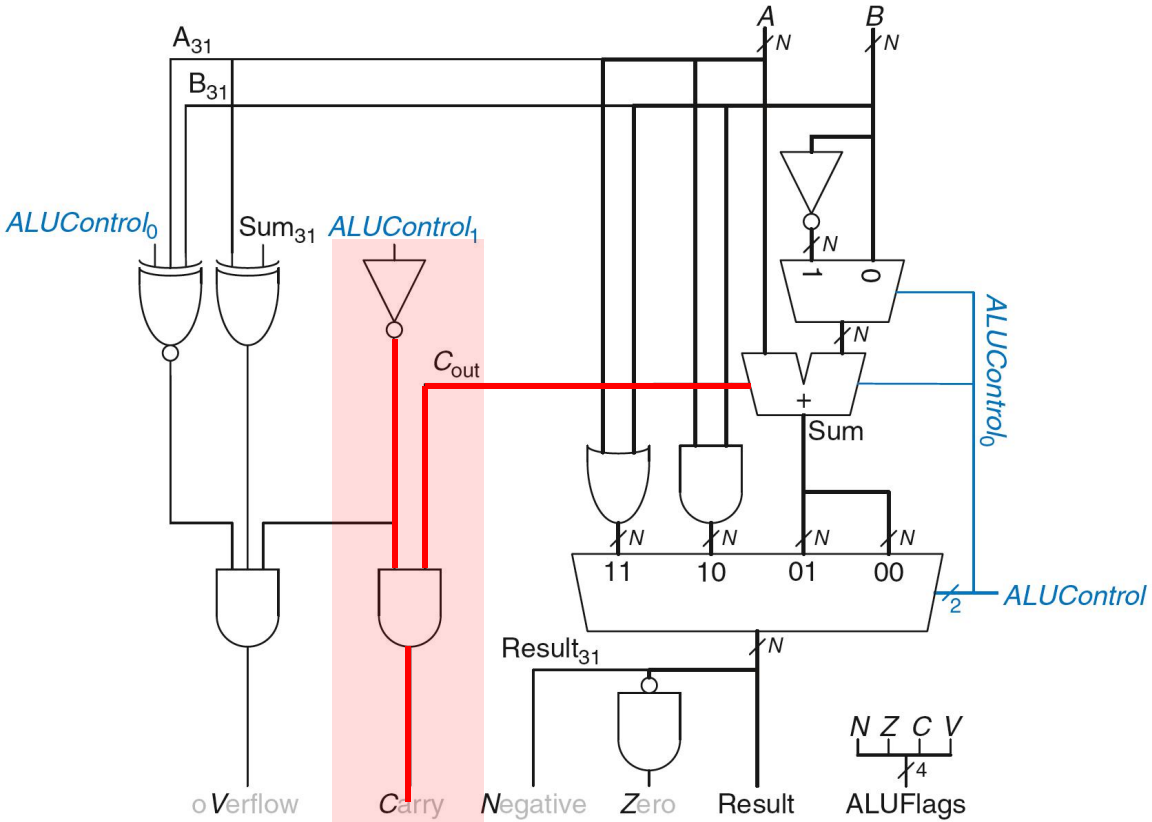
Z=AND(Result bits) = 0

E.g.: Result = 0000

~Result = 1111

Z=AND(Result bits) = 1

Flag Carry (C)



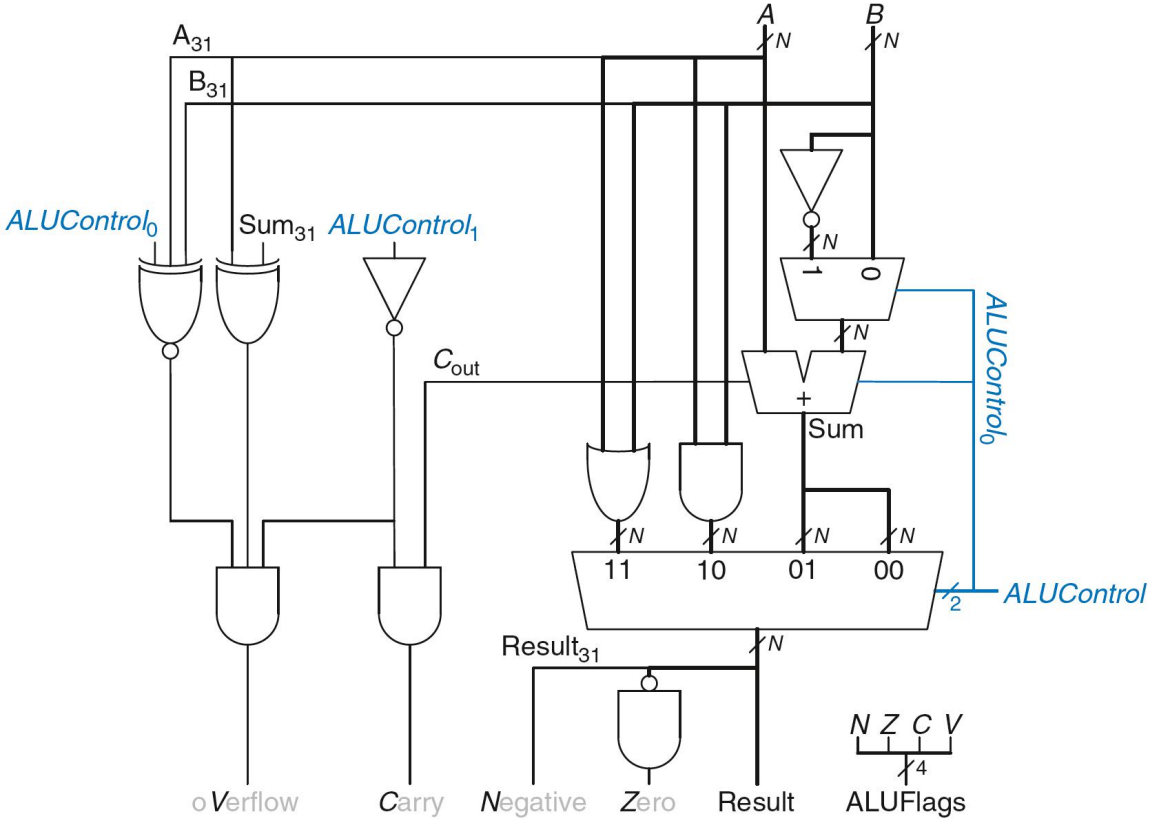
C = 1 if:

C_{out} of Adder is 1

AND

ALU is adding or subtracting (ALUControl is 00 or 01)

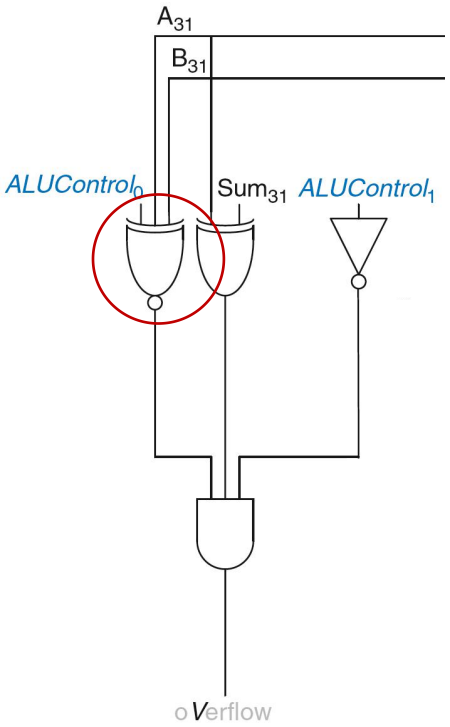
Flag Overflow (V)



V = 1 if:

The addition of 2 **same-signed numbers** produces a result with the **opposite sign**

Flag Overflow (V)



***V* = 1** if:

ALU is performing addition or subtraction

($ALUControl_1 = 0$)

AND

A and Sum have opposite signs

AND

{ A and B have same signs upon addition

($ALUControl_0 = 0$)

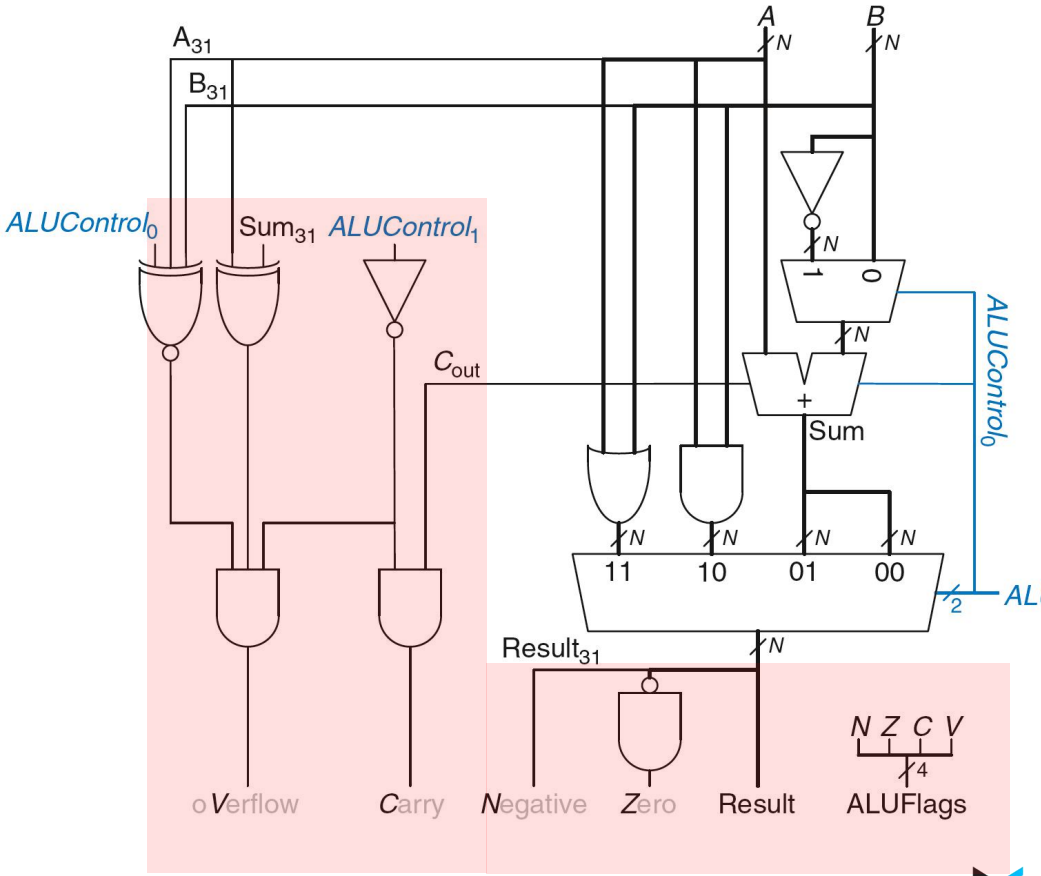
OR

A and B have different signs upon subtraction

($ALUControl_0 = 1$) }

Flags Assignment

```
  assign neg      = Result[31];
  assign zero     = (Result == 32'b0);
  assign carry    = (ALUControl[1] == 1'b0)
                    & sum[32];
  assign overflow = (ALUControl[1] == 1'b0)
                    & ~(a[31] ^ b[31] ^
                    ALUControl[0]) & (a[31]
                    ^ sum[31]);
  assign ALUFlags = {neg, zero, carry,
                     overflow};
endmodule
```



Outline

ALU Implementation

Course Logistics

Conclusions

Arch Project

- **Define teamworks for the final Project.**
  - **Up to 3 members.**
- **Collaboration also for:** Laboratory 04, 05 and 06.
- **Final project details will be revealed during next week.**

- **For starting:**
  - Recall: Verilog and ASM.
  - Recall: Topics from ISA to Microarchitecture.
  - Recall: Floating Point Arithmetic from Discrete Structures II.
  - **First job: complete Lab 04 (next lab)**

Outline

ALU Implementation

Course Logistics

Conclusions

Conclusions

- **We implemented an ALU** that can be used as building blocks for our microprocessor.
- **We evaluated the ALU using the testvector testbench.**
- **We conclude that the ALU enables our microprocessor design.**

ALU Implementation

Computer Architecture

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

CS3501 - 2023I
PROF.: JGONZALEZ@UTEC.EDU.PE