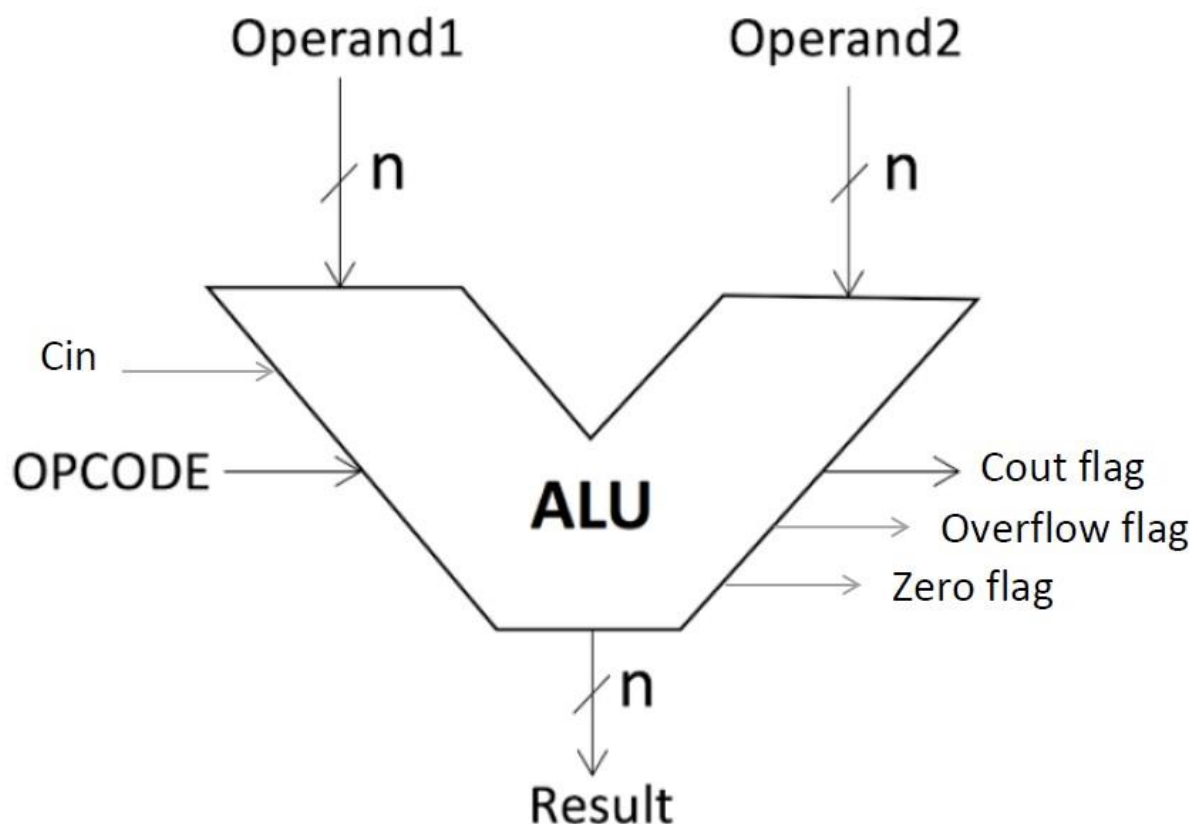


## [CSCI311-Fall22]Computer Architecture Project Phase 1

### Implement n-bits full ALU (10 points)

Description: The ALU performs simple addition, subtraction, multiplication, division, and logic operations, such as OR and AND. The memory stores the program's instructions and data.

Name of Main Module: ALU



## Options of OpCode:

OpCode	Operation
0000	Op1 AND Op2
1111	Op1 OR Op2
1001	Op1 + Op2
0110	Op1 – Op2

## Notes on variables of the ALU:

- **N**: the size of the input/output of the ALU. It should be variable set to 32 in the ALU module.
- **OpCode**: chooses which operation to apply on the operands
- **Op1**: first input, n-bit in length
- **Op2**: second input, n-bit in length
- **Cin**: Carry input (to be considered in addition/subtraction operations) • **Cout**: Carry output (to be considered in addition/subtraction operations) • **Overflow flag**: if in addition, oFlag is turned on when:
  - The sum of two numbers with the sign bit off yields a result number with the sign bit on. ✦  $0100 + 0100 = 1000 \Rightarrow$  overflow flag is turned on.
  - The sum of two numbers with the sign bit on yields a result number with the sign bit off.
    - ✦  $1000 + 1000 = 0000 \Rightarrow$  overflow flag is turned on.
  - Mixed-sign addition never turns on the overflow flag.
  - For more details to understand it, check the link [here](#).
  - If in subtraction, same rules apply between Op1 and 2's complement of Op2.
- **Zero Flag**: set to 1 in case the output is zero
- **Result**: output of the ALU, n-bit in length

## Grading Schema:

Code Structure	4 points
Wave Window	1 point
RTL Design	1 point
Test case 1	0.5
Test case 2	0.5
Test case 3	0.5
Test case 4	0.5
Test case 5	1
Test case 6	1
<b>(Total)</b>	<b>10</b>

## Testing Bench (Main):

```
module main();
  reg [31:0] op1;
  reg [31:0] op2;

  reg [3:0] opCode;
  reg Cin

  wire [31:0] result;
  wire cflag, zflag, oflag;

  ALU my_alu(OpCode, op1, op2, cin, result, cflag, zflag, oflag);

  initial
  begin
    $display("Hello, World");
    $finish ;
  end
endmodule
```

## Test Cases:

1.

Type	Var. Name	Value
Input	Op1	00110011001100110011001100110011
Input	Op2	11001100110011001100110011001100
Input	OpCode	1111
Input	Cin	1
Output	Result	11111111111111111111111111111111
Output	cFlag	0
Output	zFlag	0
Output	oFlag	0

2.

Type	Var. Name	Value
Input	Op1	00110011001100110011001100110011
Input	Op2	11001100110011001100110011001100
Input	OpCode	0000
Input	Cin	0
Output	Result	00000000000000000000000000000000
Output	cFlag	0
Output	zFlag	1
Output	oFlag	0

3.

Type	Var. Name	Value
Input	Op1	00000000000000001111111111111111
Input	Op2	00000000000000000000000000000001
Input	OpCode	1001
Input	Cin	1
Output	Result	00000000000000001000000000000001
Output	cFlag	0
Output	zFlag	0
Output	oFlag	0

4.

Type	Var. Name	Value
Input	Op1	00000000000000000000000000001000
Input	Op2	00000000000000000000000000000001
Input	OpCode	0110
Input	Cin	0
Output	Result	0000000000000000000000000000111
Output	cFlag	0
Output	zFlag	0
Output	oFlag	0

5.

Type	Var. Name	Value
Input	Op1	11110000111100001111000011110001
Input	Op2	00001111000011110000111100001111
Input	OpCode	1001
Input	Cin	0
Output	Result	00000000000000000000000000000000
Output	cFlag	1
Output	zFlag	1
Output	oFlag	0

6.

Type	Var. Name	Value
Input	Op1	0011001100110011001100110011
Input	Op2	0100110011001100110011001100
Input	OpCode	0110
Input	Cin	1
Output	Result	1110011001100110011001100110
Output	cFlag	1
Output	zFlag	0
Output	oFlag	1

### Rules: • Teams are from 3-5.

- **No plagiarism! Both teams would get a zero, and a penalty for next phases, no discussion.**
- **Any submission after the deadline will not be accepted**
- **No copying from the internet (still counts as plagiarism!)**

### Deliverables:

Only one member of the team should deliver a zip file including the following:

- PDF Files including:
  - Team members names & ID's
  - Screenshots of simulation in all 6 test cases
    - ✦ Please show the values in binary radix and set the pointer (the yellow column) in each test case in a separate screenshot, so it's visible right away on the left side bar.
  - Workload distribution between team members
- Verilog files ○ **Bonus point** if you wrote your code in separate files & the test bench in a separate "main.v" file.
  - ✦ To explain more, like OOP concepts. We want a Verilog file for adder modules, subtractor modules, one for ALU which would include from both adder & subtractor, and finally a main file having the test bench which would import the ALU.