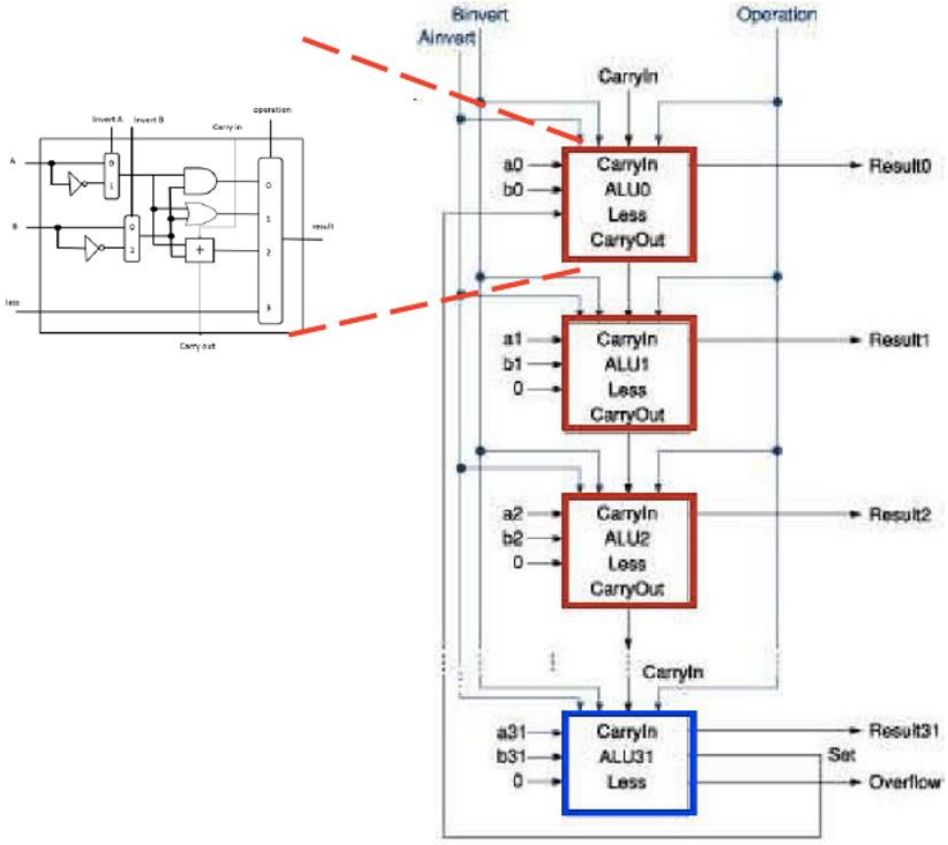
Computer Organization

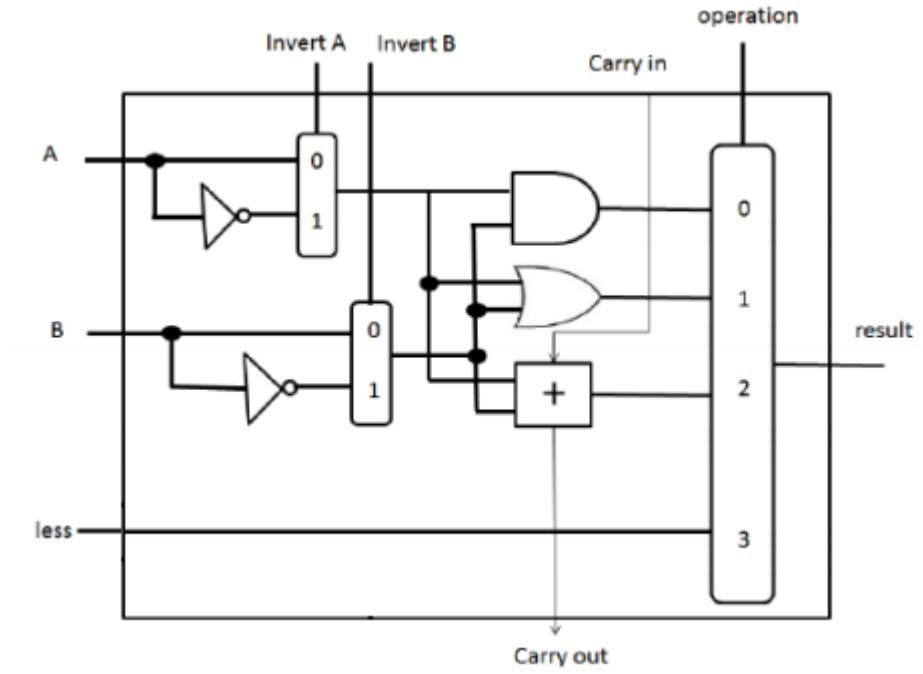
Architecture diagram:

1-bit ALU **32-bit ALU**



Blue 1-bit ALUtop

Detailed description of the implementation:



Intuition on ALU hardware implementation:

- **Use switches and MUXs to build 1-bit ALU depends on basic operations needed.**
- **Control input is like the condition to do different type of operation which is why we wired it into MUXs which are acting as selector for different operations.**
- **Use wires to connect 32 1-bit ALUs to build a 32-bit ALU, where last 2 bit of control input is wired to 4x1 MUXs as operation selector, and first 2 bit of control input is wired to 2 2x1 MUXs as selector for input inverting.**

ALUtop:

4 Operations

Operation Code	Operation	Implementation
00	Use by AND/NOR	Using & operator
01	Use by OR/NAND	Using | operator
10	Use by ADD/SUB	Adding A, B and Cin together
11	Use by SLT	Set result to less (more will be explained later).

32-bit ALU:

ALU action	Name	ALU control input
And	And	0000
Or	Or	0001
Add	Addition	0010
Sub	Subtract	0110
Nor	Nor	1100
Nand	Nand	1101
Slt	Set less than	0111

Control input consist of 4-bit A B C D.

- **A is wired to A_invert MUX.**
- **B is wired to B_invert MUX.**
- **CD is wired to 2-bit opcode MUX.**

AND and NOR share same op code 00:

- **AND is trivial.**
- **(A OR B)' = A' AND B' by De Morgan's Law**

OR and NAND share same op code 01:

- **OR is trivial.**
- **(A AND B)' = A' NOR B' by De Morgan's Law**

ADD and SUB share same op code 10:

- **A SUB B is just A ADD B'. (A+ -B)**

SLT implementation:

- **Perform A SUB B and check the sign. If A < B set less to 0 for all ALU except the least significant bit ALU is set to 1.**
- **If A >= B set less = 0 for all ALU.**
- **Then if opcode is 11, less is set to result as per ALUtop implementation.**

In ALU.V, connect 32 ALU_top using for loop to form 32bit ALU:

- **EXCEPT for least significant bit ALU (ALU0):**
  - **Cin is determined by ALU control input**
  - **Less will be determined by the result of the minus operation. (assign 'SET' of MSB to less of LSB).**

- **The rest 32 ALU_top is generate and connected by for loop.**
    - **Pass Cout of previous bit to Cin of current bit.**

ZCV overflow detection:
Check most significant bit (MSB):

- **Set zero to 1 if result is 0, otherwise set zero to 0.**
- **Set cout to 1 if carry_out is 1, otherwise set cout to 0.**
- **Set overflow to 1 if cin != cout, otherwise set overflow to 0.**

Problems encountered and solutions:

At first, we do not have any slightest idea on what is Verilog, as this is the very first time that we encounter a Verilog assignment. We are from ECE department and even though we had taken Logic Design course, we were never required to write any HDL so we have to learn HDL from scratch.
Therefore, we try to find tutorials from the internet (mainly YouTube) and learn from those tutorials.

Lesson learnt (if any):

1. **How to install ModelSim and request for a license.**
2. **How to write Verilog (or HDL, in general).**
3. **What is an ALU, what can it do, and how does it use logic gate and MUXs to do arithmetic and logical computation.**
4. **How to write a 32-bit ALU in Verilog using 1-bit ALUs.**
5. **We recognized that the 1-bit ALUtop has different configurations compared to the other 31 1-bit ALUs.**