COMP2020 Project 1: ALU, Design Document

Nguyen Dinh Cuong, V202000025

1 Overview

This is a 32-bit ALU with the following features: equal, not equal, and, or, nor, xor, less than or equal to, greater than, add, subtract, right shift arithmetic, and left/right shift logical. The ALU is part of a RISC V processor, which is the latter project that we have to do.

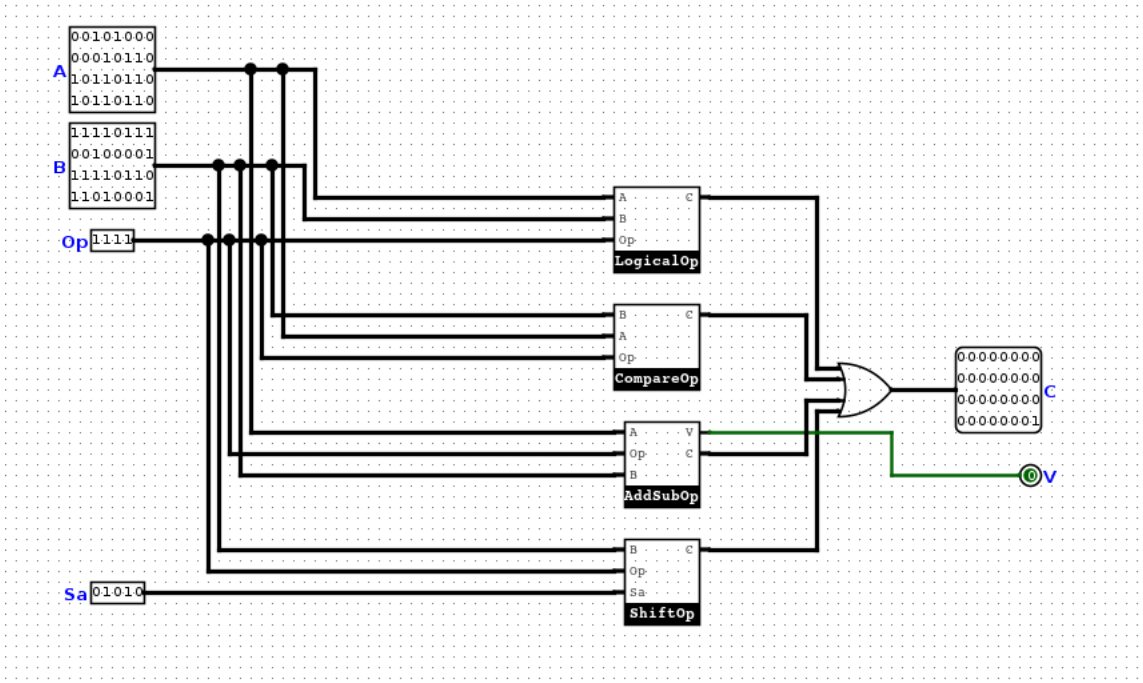

Figure 1: Final Product - The 32-bit ALU

2 Logical Operators

2.1 Implementation Details

AND, OR, XOR, and NOR are the four logical operators that the ALU must execute. Since NOR could be replaced using the NOT of OR, we can use three gates only. Op bits 0 and 1 represent the

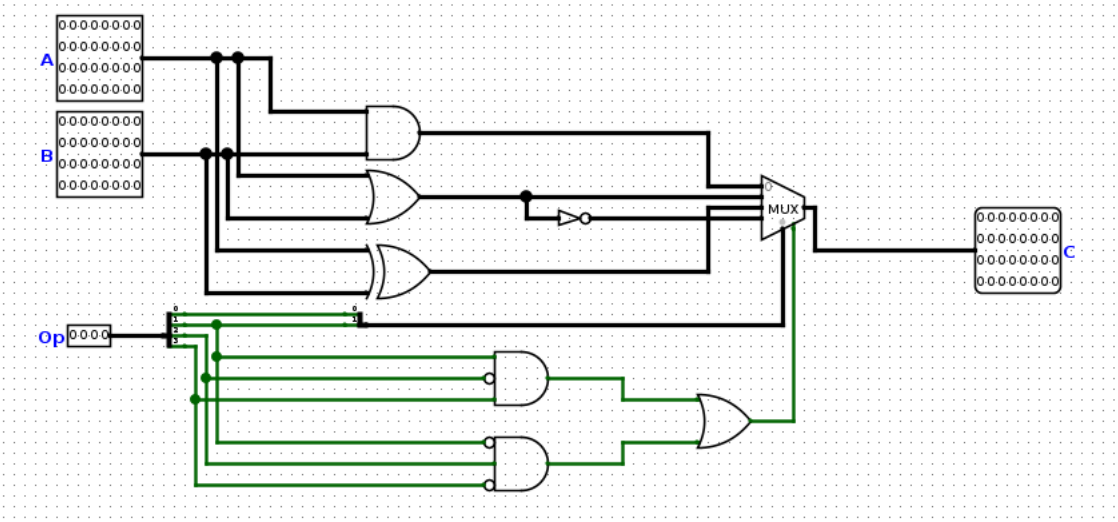

Figure 2: Logical Operators Implementation

control for these operations. The two Op bits are combined into a single wire and fed into a 2x4 mux as the input signal.

Truth Table of Logic Operators		
Op Bit 0	Op Bit 1	Output
0	0	A AND B
0	1	A XOR B
1	0	A OR B
1	1	A NOR B

2.2 Evaluation

The most significant design decision was to divide the wire after OR rather than implementing a separate NOR gate. While adding another NOR gate to the circuit may improve its clarity, employing a NOT gate is more efficient.

3 Value Comparisons

3.1 Implementation Details

This circuit is divided into two sections, the upper half comparing A and B. A XOR B equals a 32-bit zero if A is equal to B, thus, A XNOR B equals a 32-bit one if A equals B. Check if A XOR B equals zero using AND gates. If the answer is yes, A equals B; if the answer is no, A does not equal B. We verify that A is greater than zero in the lower section. If A is positive, the 31st bit will be zero, and

at least one of the remaining bits will not be zero. A is positive if it meets this criterion; else, it's less than or equal to zero.

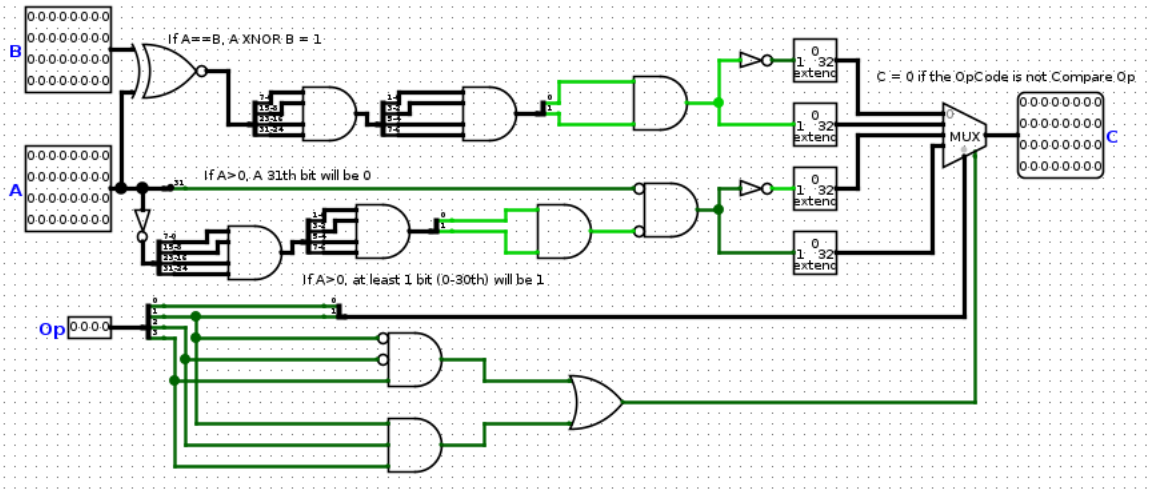

Figure 3: Comparing Operation Implementation

The control for these operations is likewise represented by Op bits 0 and 1. As the input signal, the two Op bits are merged into a single wire and sent through a 2x4 mux.

Truth Table of Comparing Operation		
Op Bit 0	Op Bit 1	Output
0	0	(A != B) ?
0	1	(A == B) ?
1	0	(A $\leq$ 0) ?
1	1	(A $>$ 0) ?

3.2 Evaluation

The function of this circuit gets apparent and the implementation becomes simpler by splitting it into two smaller components. Using those bit extenders for calculating and gates for Op bits handling are effectively-sufficient methods in this case.

4 Adder/Subtractor

4.1 Implementation Details

4.2 Evaluation

5 Shifter

5.1 Implementation Details

5.2 Evaluation

6 Output C

Any portion of this ALU will only produce a non-zero result if and only if the operation is corresponding to this component, as shown in the figure below. As a consequence, the only non-zero result is output into C through an OR gate.

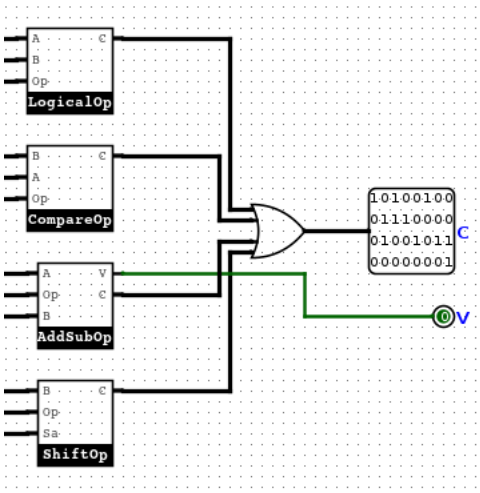

Figure 4: Comparing Operation Implementation