COMPUTER ORGANISATION – COMP2020 – ARITHMETIC LOGICAL UNIT PROJECT (ALU) 32-BIT

VINUNIVERSITY – COLLEGE OF ENGINEERING AND COMPUTER SCIENCE

NGUYEN TRONG NHAN – V202000224

1. Brief overview:

In this project, I design a 32-bit arithmetic logical unit (ALU). This ALU can perform main functions on either one or two 32-bit numbers.

This ALU contains a gate circuit, a compare circuit, an add and subtract circuit, and a shift circuit. The operations of these circuits will be described as below.

Here is the ALU:

Diagram

Description automatically generated

1. Operations:
2. Gate circuit:

+ This gate circuit serves a logic operation which consists of AND, OR, XOR, NOR.

+ Here, to create a value that fits the Op, I use a mux to pick the output. Besides, by combining the bit zero and bit one from the Op into a single wire, the input will be directed to a 2x4 multiplexer.

Diagram, schematic

Description automatically generated

Here is the truth table:

|  |  |  |
| --- | --- | --- |
| 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 |

1. Shift circuit:

a/ Mirror:

This mirror will mimic the input and output the input through the SignBit. This SignBit is used to send carry-in value into the LeftShift32 when operating a right shift arithmetic.

Diagram, schematic

Description automatically generated

b/ LeftShift32:

Here is my LeftShift32 which consists of 1 Shift1bit, 1 Shift2bit, 1 Shift4bit, 1 Shift8bit, and 1 Shift16bit and 5 muxes.

Diagram

Description automatically generated

c/ Shift circuit:

This shift circuit has the below truth table:

|  |  |
| --- | --- |
| Op bit 2 | Output |
| 0 | A << B |
| 1 | A >> B |

Here, we depend on the two muxes to determine the shift type. While the first mux uses the value of B and inverse of B, the second one will re-reverse B if there is a right shift performance.

It also uses bit two from Op to control the bit. If the shift moves to the right, B is inverted. If the shift moves to the left, B remains unaltered.

Diagram

Description automatically generated

1. Add and subtract circuit:

a/ 1-bit Adder:

Here, Cin is known as the carry-in bit, S is the output, Cout is the carry-out bit while A and B are 1-bit input values.

Diagram, schematic

Description automatically generated

b/ 4-bit Adder:

Here, I create a 4-bit adder by combining four 1-bit adders. In this 4-bit adder, “overflow” serves as the output value that helps us oversee whether there is an overflow.

Diagram

Description automatically generated with medium confidence

c/ 16-bit adder:

Likewise, I combine four 4-bit adders to make a 16-bit adder. In this 16-bit adder, “overflow” serves as the output value that helps us oversee whether there is an overflow.

Diagram

Description automatically generated

d/ 32-bit adder:

Just like the preliminary adders, I use four 16-bit adders to create a 32-bit adder. Diagram

Description automatically generated

e/ Add and subtract circuit:

Here, to process the add and subtract circuit, bit two will be used from the OpCode output. If it is equal to zero, the sum of A and B will be processed. If the Op Bit is zero, then the subtract of A and B will be processed. Besides, one bit extender and a XOR gate are implemented to take the inverse of B.

To manage the OpCode bits, I utilise one AND gate with two inputs: one bit extender and a negated input.

Diagram, schematic

Description automatically generated

1. Compare circuit:

Here is the truth table of this compare circuit.

|  |  |  |
| --- | --- | --- |
| Op bit 0 | Op bit 1 | Output |
| 0 | 0 | (A != B) ? |
| 0 | 1 | (A == B) ? |
| 1 | 0 | (A ≤ 0) ? |
| 1 | 1 | (A > 0) ? |

In this circuit, A and B is put in comparison. If A is equal to B, then A^B is a 32-bit zero.

In order to know whether A^B is equal to zero, I will utilise AND gates. If the output is yes, then A=B. If not, then A!=B.

To check whether A>0, if A>0, A’s 31st bit is equal to zero while there exists at least one bit that is not zero from the remaining bits. A>0 is confirmed if the condition is checked. If not, A<=0.

The input is directed into a 2x4 mux by merging bit zero and bit one from Op into a single wire.

Diagram

Description automatically generated

1. Conclusion:

With all of the above, I create my own ALU 32-bit. Here is the final product (again). Thank you!

Diagram

Description automatically generated