Project 1
ALU Design

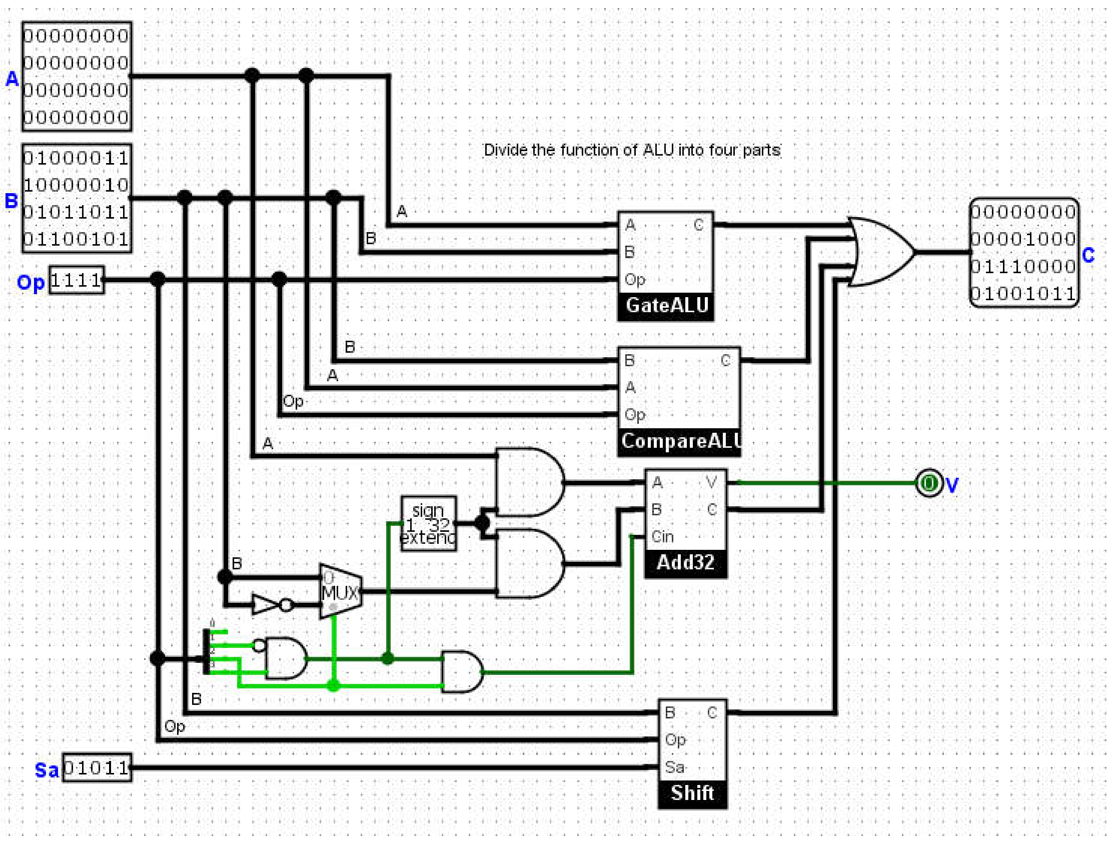
Shengqi Wang

sw979

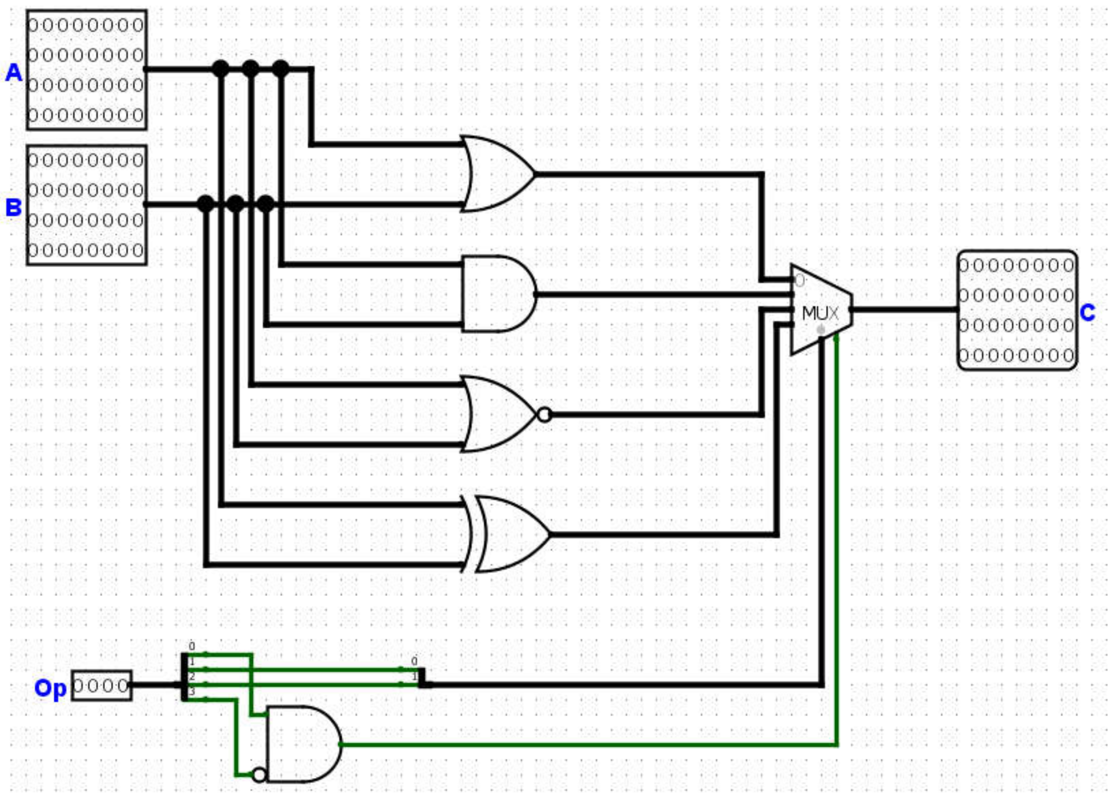
02/10/2019

1. Overview

This is a 32-bit ALU which can realize basic function for one or two 32 bits number. This ALU has 4 parts, including logic operation, compare operation, add/subtract operation and shift operation.



2. Logic operation

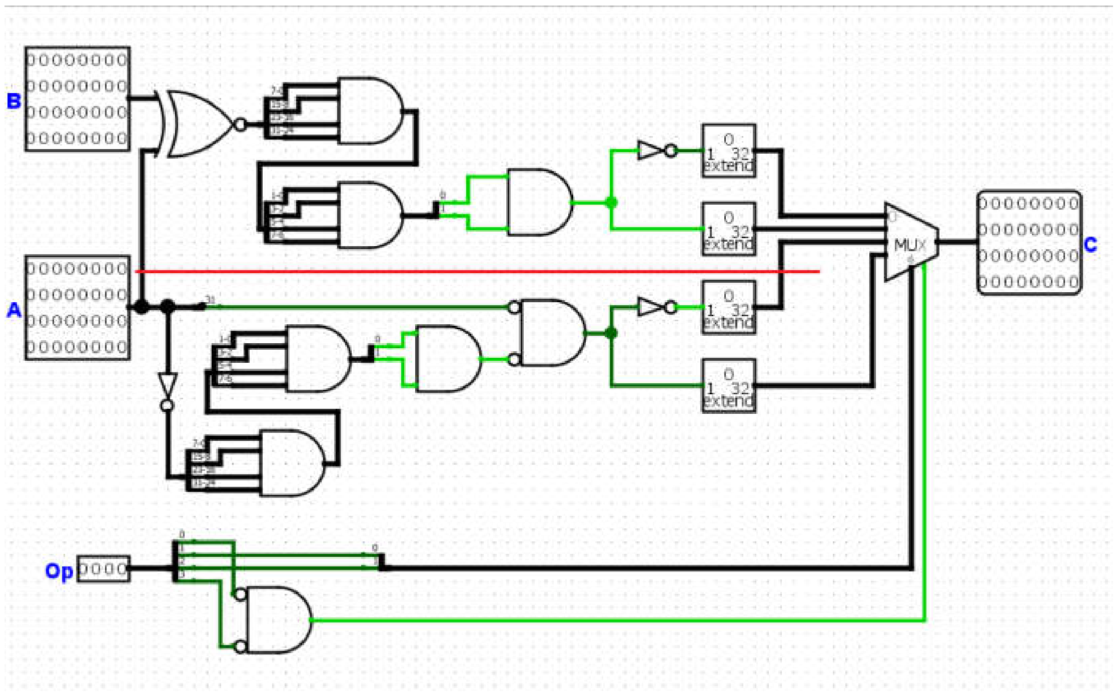


We use four 32-bit gates to realize AND, OR, NOR and XOR functions.

To choose the result correspond to the operation (Op), we use a mux to choose what to output. If Op is not 0xx1, the mux will not be disable and only zero will be output. Otherwise, this mux will choose the output correspond to the operation (Op).

Op	Mux
0001	0
0101	1
0011	2
0111	3
Otherwise	Disable

3. Compare operation



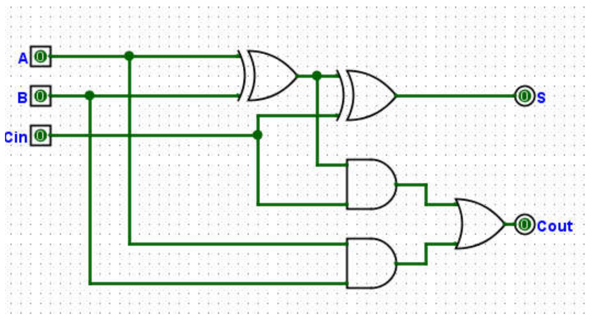
Above the red line, we compare A and B. If A is equal to B, A^B equals to a 32-bit zero. Use AND gates to check if A^B is zero. If yes, A is equal to B, otherwise A is not equal to B. Below the red line, we check whether A is bigger than zero. If A is positive, its 31th bit will be zero, while there exits at least one of all other bits is not zero. If A satisfy this condition, A is positive, otherwise A is less or equal to zero.

For now, we compare A and B and check if A is positive. In the right of the diagram we get our four answers for comparation part. We use a mux to choose the answer correspond to our operation (Op). If Op is not 0xx0, the mux will be disable and only 0 will be output. Otherwise this mux will choose the output that correspond to our operation.

Op	Mux
0010	0
0000	1
0110	2
0110	3
Otherwise	Disable

4. Add/subtract operation

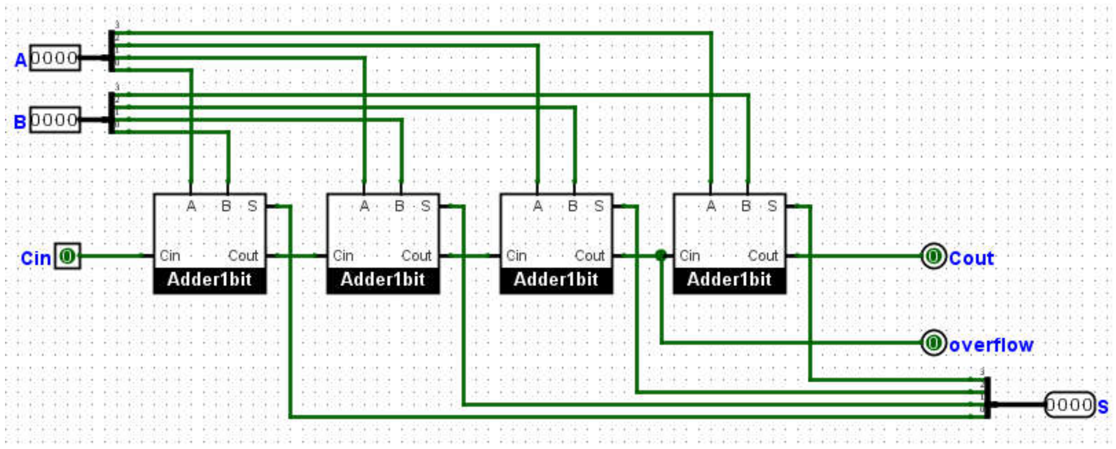
a) 1-bit full adder:



Shown in above diagram, A and B are the 1-bit inputs, and Cin is the carry-in bit. S is the output bit, and Cout is the carry-out bit.

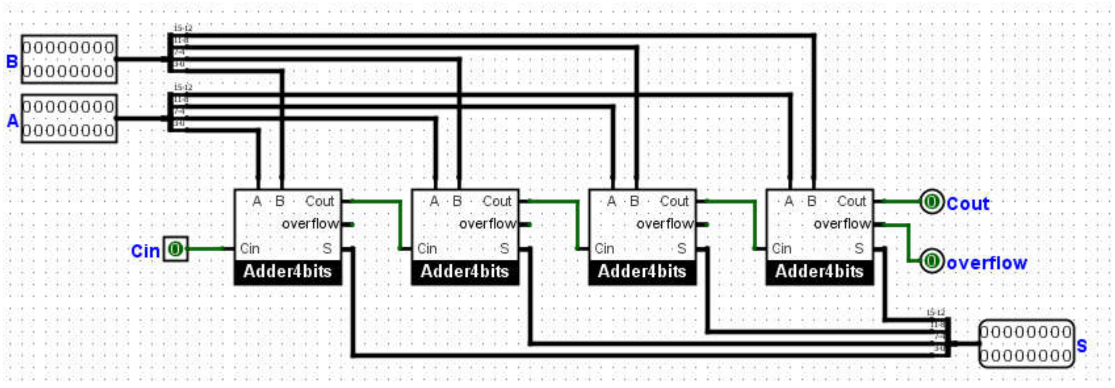
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

b) 4-bit full adder:



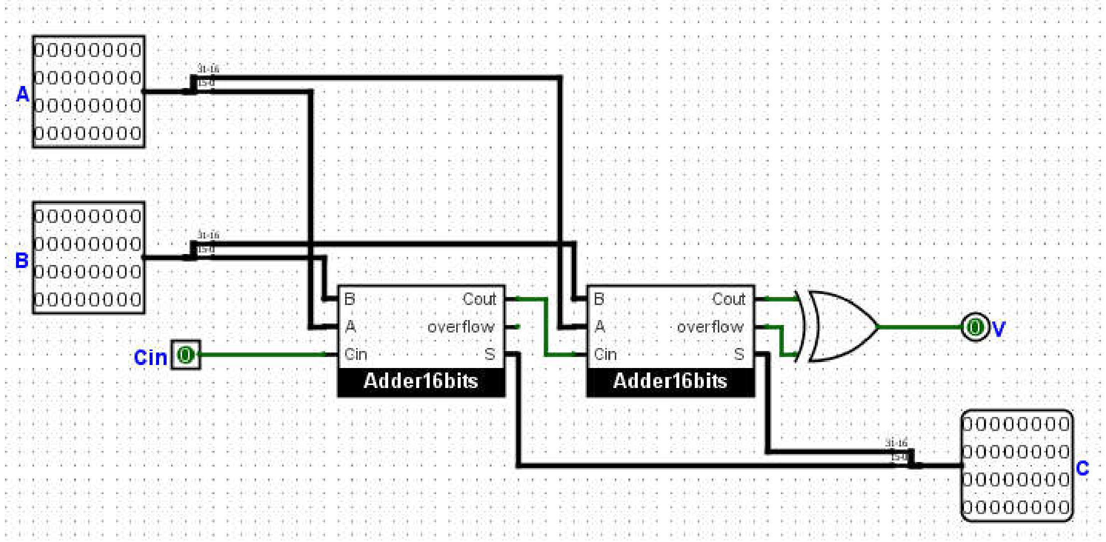
Combine four 1-bit full adder, we can get a 4-bit full adder. Pin overflow is an output to help us detect whether there is an overflow happens.

c) 16-bit full adder:



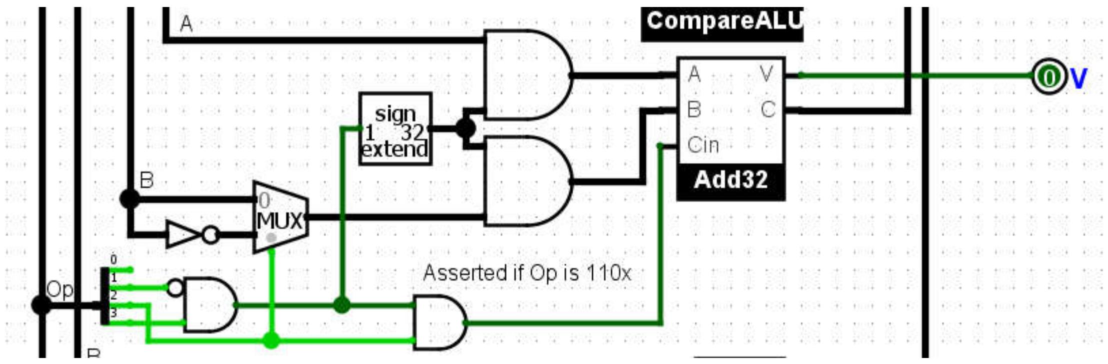
Combine four 4-bit full adder, we can get a 16-bit full adder. Pin overflow is an output to help us detect whether there is an overflow happens.

d) 32-bit full adder:



Combine two 16-bit full adder, we can get a 32-bit full adder. If Cout is not equal to overflow, V will be asserted to show that overflow happens.

e) Operation part:



A and B will be pushed into Add32 if and only if Op is 1x0x. Otherwise only 0 will be pushed into Add32. Also, Cin will be asserted if and only if Op is 110x, otherwise Cin will be 0. These make sure that when Op is not supposed to do adding or subtracting, Add32 circuit will always output 0.

If Op is 100x(add), the mux in the diagram will choose B to push into Add32 circuit, and Cin will be 0.

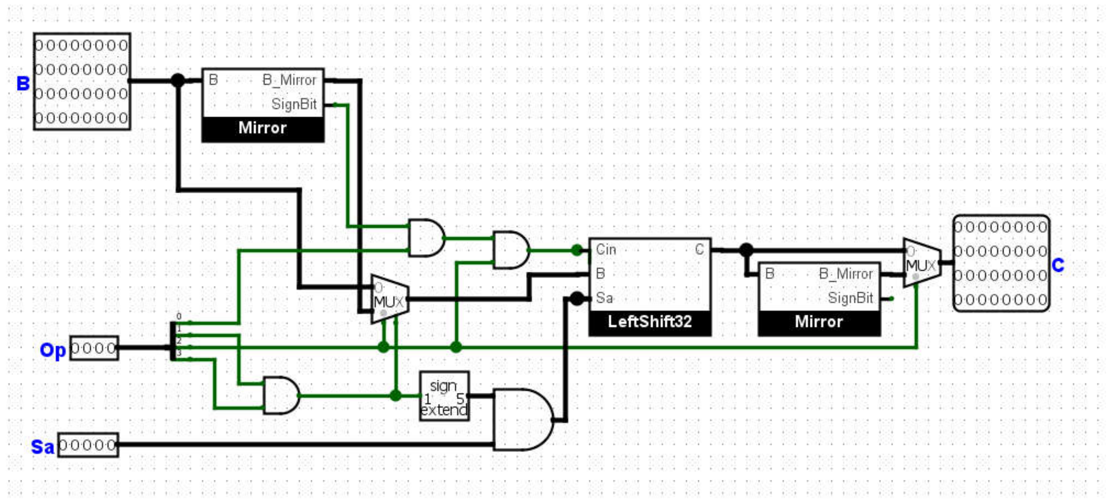
If Op is 110x(subtract), the mux in the diagram will choose ~B to push into Add32 circuit and Cin will be 1.

As a result, our operation part makes sure that add/subtract part can output the outcome we want.

Op	Cin	Mux	**A**	**B**
100x	0	0	A	B
110x	1	1	A	~B
Otherwise	0	0/1	0	0

A, B means values input into Add32 circuit as A and B.

5. Shift operation



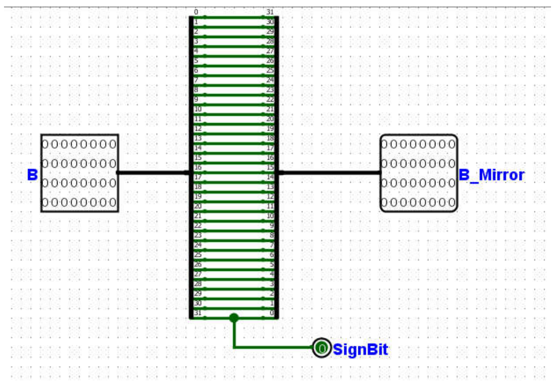
If Op is not 1x1x, the left mux will be disable and zero will be sent into LeftShift32 circuit. Also, the biggest AND gate will send zero as Sa into LeftShift32 circuit and finally only zero will be sent out and C will be zero.

When Op is 101x, the second highest bit of Op let the left mux to choose B (not the mirror of B) to push into LeftShift32 circuit. And it will also let the right mux to choose the outcome of LeftShift32 circuit to become C, but not the mirror of the outcome.

When Op is 1110 or 1111, the second highest bit of Op will let B become mirror B and outcome of LeftShift32 circuit become mirror outcome. We know that when we want to right shift any number, we can first mirror it, then left shift it and finally mirror the result. As a result, we can use this circuit to control whether left or right shift.

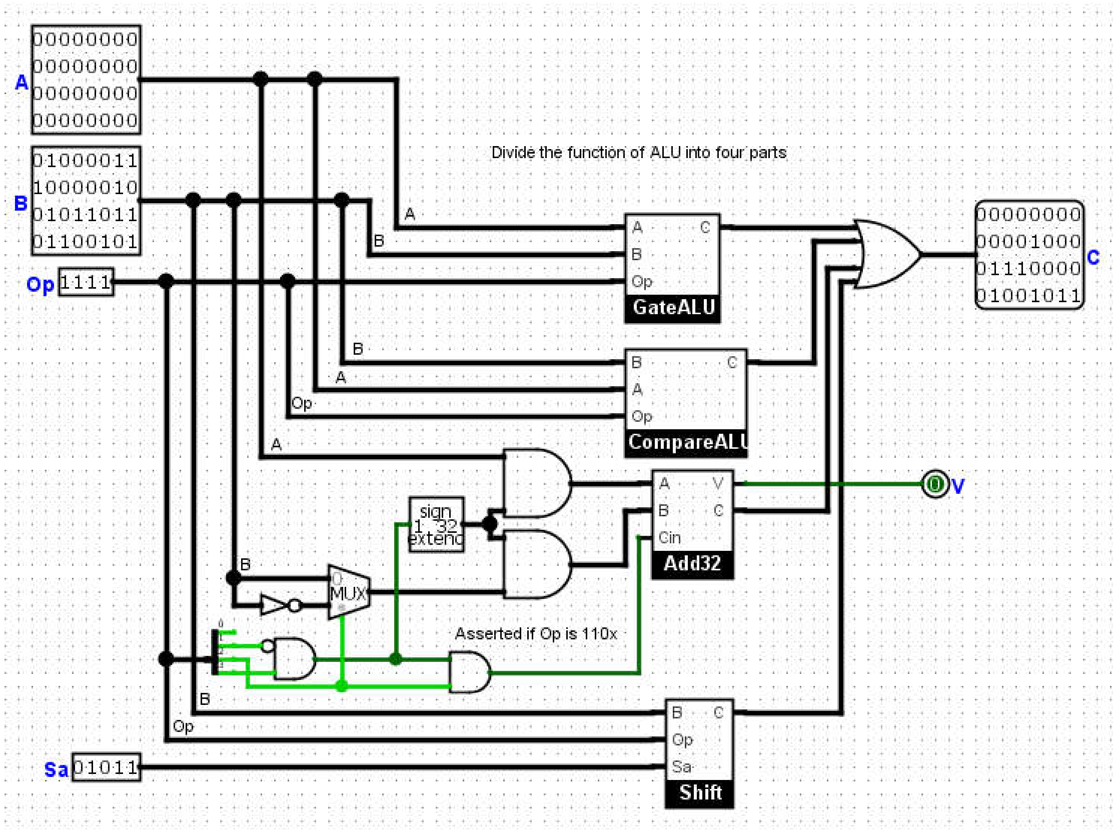
Op	Left-Mux	Right-Mux	**Cin**	**B**	**Sa**
101x	0	0	0	B	Sa
1110	1	1	0	MirrorB	Sa
1111	1	1	0/1	MirrorB	Sa
Otherwise	Disable	0/1	0	0	0

Cin, B, Sa means values input into LeftShift32 circuit as Cin, B and Sa.



The mirror circuit can not only mirror the input, but also can output the Sign bit of the input. When we need to do an arithmetic right shift, we can use this sign bit to send Cin into LeftShift32 circuit.

6. Gathering four parts



Based on above, any part of this ALU will only output a non-zero result if and only if the operation (Op) is corresponding to this part. As a result, we use an OR gate to output the only non-zero result into C.