CPT_S 260 Intro to Computer Architecture
Lecture 25

Digital Design IV
March 9, 2022

Ganapati Bhat
School of Electrical Engineering and Computer Science
Washington State University

Recap: Rules of Boolean Algebra

- **Associative Law of multiplication**

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

- **Distributive Law of multiplication**

$$A + BC = (A + B) \cdot (A + C)$$

- **Annulment law:**

$$A \cdot 0 = 0$$
$$A + 1 = 1$$

- **Identity law:**

$$A \cdot 1 = A$$
$$A + 0 = A$$

Recap: Rules of Boolean Algebra

- **Complement law:**

$$A + \bar{A} = 1$$
$$A \cdot \bar{A} = 0$$

- **Double negation law:**

$$\bar{\bar{A}} = A$$

- **Absorption law:**

$$A \cdot (A + B) = A$$
$$A + AB = A$$
$$A + \bar{A}B = A + B$$

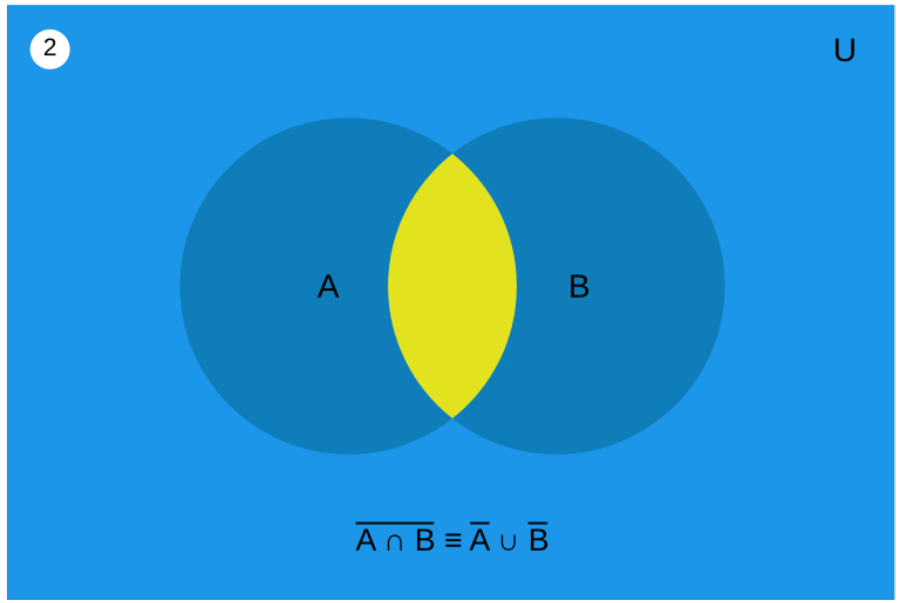
- **Idempotent law:**

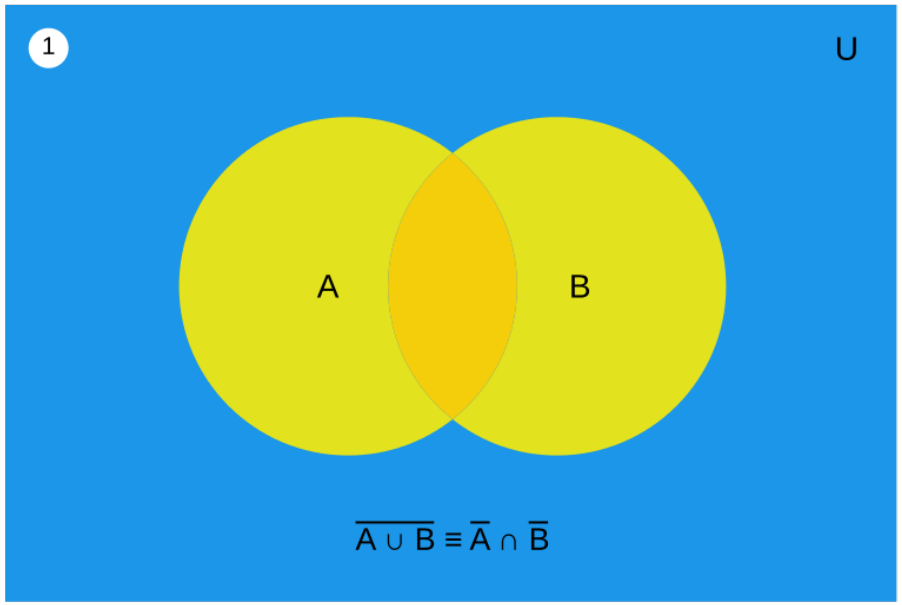
$$A + A = A$$
$$A \cdot A = A$$

Recap: De Morgan's Laws

- **Transformation rules that help simplification of negations**
- **Statement:**

$$\overline{AB} = \bar{A} + \bar{B}$$
$$\overline{(A + B)} = \bar{A} \cdot \bar{B}$$





Sum of Products

- **Minterm Expressions**
- **If input is 0 we take the complement of the variable**
- **If input is 1 we take the variable as is**
- **To get the desired canonical SOP expression we will add the minterms (product terms) for which the output is 1**

$$F = \bar{A}B + A\bar{B} + AB$$

A	B	F	Minterm
0	0	0	A'B'
0	1	1	A'B
1	0	1	AB'
1	1	1	AB

Product of Sums

- **Maxterm Expressions**
- **If input is 1, we take the complement of the variable**
- **If input is 0, we take the variable as is**
- **To get the desired canonical POS expression we will multiply the maxterms (sum terms) for which the output is 0**

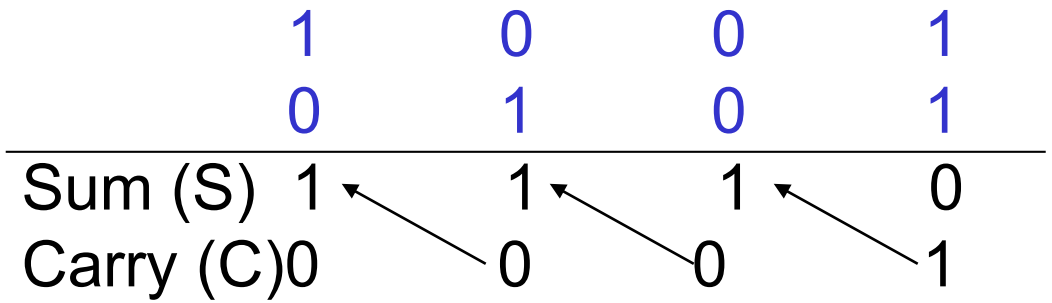
$$F = (A + B)$$

A	B	F	Maxterm
0	0	0	A + B
0	1	1	$A + \overline{B}$
1	0	1	$\overline{A} + B$
1	1	1	$\overline{A} + \overline{B}$

Designing a Simple ALU

(Arithmetic Logic Unit)

Adder Algorithm



Truth Table for the above operations:

A	B	Cin	Sum	Cout
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Adder Algorithm

	1	0	0	1
	0	1	0	1
Sum	1	1	1	0
Carry	0	0	0	1

Truth Table for the above operations:

A	B	Cin	Sum	Cout
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

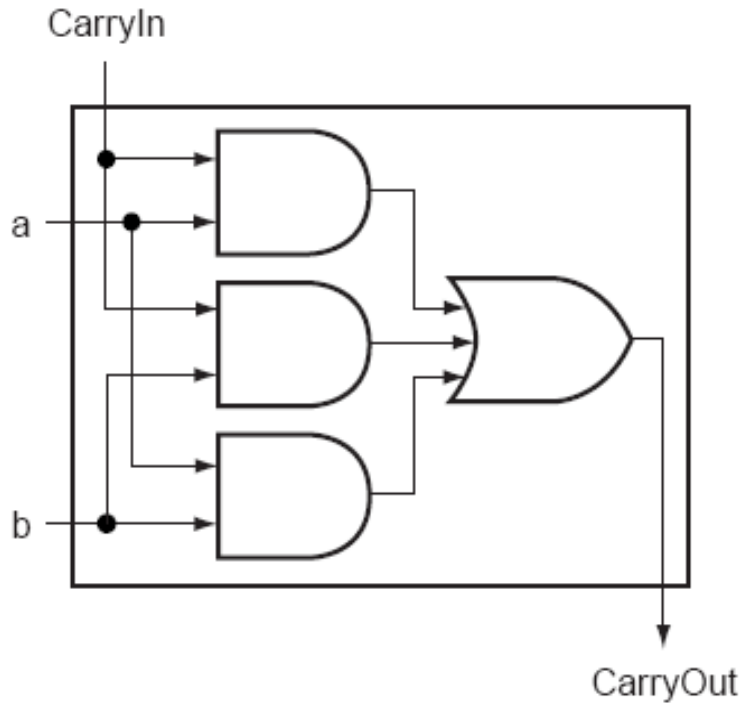
Equations:

$$Sum = Cin \cdot \overline{A} \cdot \overline{B} + B \cdot \overline{Cin} \cdot \overline{A} + A \cdot \overline{Cin} \cdot \overline{B} + A \cdot B \cdot Cin$$

$$Cout = A \cdot B \cdot Cin + A \cdot B \cdot \overline{Cin} + A \cdot Cin \cdot \overline{B} + B \cdot Cin \cdot \overline{A}$$

$$= A \cdot B + A \cdot Cin + B \cdot Cin$$

Carry Out Logic



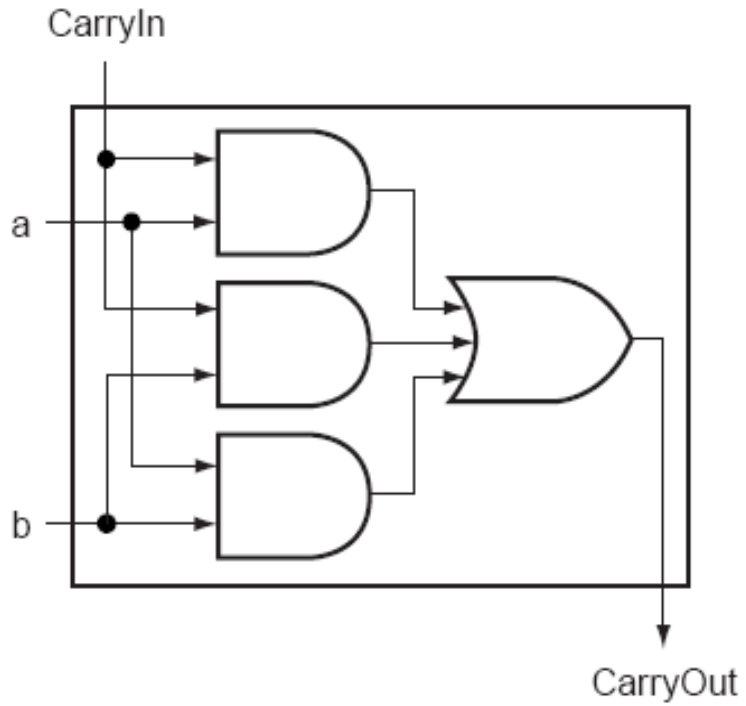
Equations:

$$\text{Sum} = Cin \cdot \overline{A} \cdot \overline{B} + B \cdot \overline{Cin} \cdot \overline{A} + A \cdot \overline{Cin} \cdot \overline{B} + A \cdot B \cdot Cin$$

$$\text{Cout} = A \cdot B \cdot Cin + A \cdot B \cdot \overline{Cin} + A \cdot Cin \cdot \overline{B} + B \cdot Cin \cdot \overline{A}$$

$$= A \cdot B + A \cdot Cin + B \cdot Cin$$

Carry Out Logic



$$C_{out} = A \cdot B \cdot Cin + A \cdot B \cdot \overline{Cin} + A \cdot Cin \cdot \bar{B} + B \cdot Cin \cdot \bar{A}$$

$$= A \cdot B \cdot (Cin + \overline{Cin}) + A \cdot Cin \cdot \bar{B} + A \cdot B \cdot Cin + B \cdot Cin \cdot \bar{A} + A \cdot B \cdot Cin$$

$$= A \cdot B + A \cdot Cin \cdot (B + \bar{B}) + B \cdot Cin \cdot (A + \bar{A})$$

Full-Adder (3 inputs, 2 outputs)

- **Full-Adder Truth Table:**

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

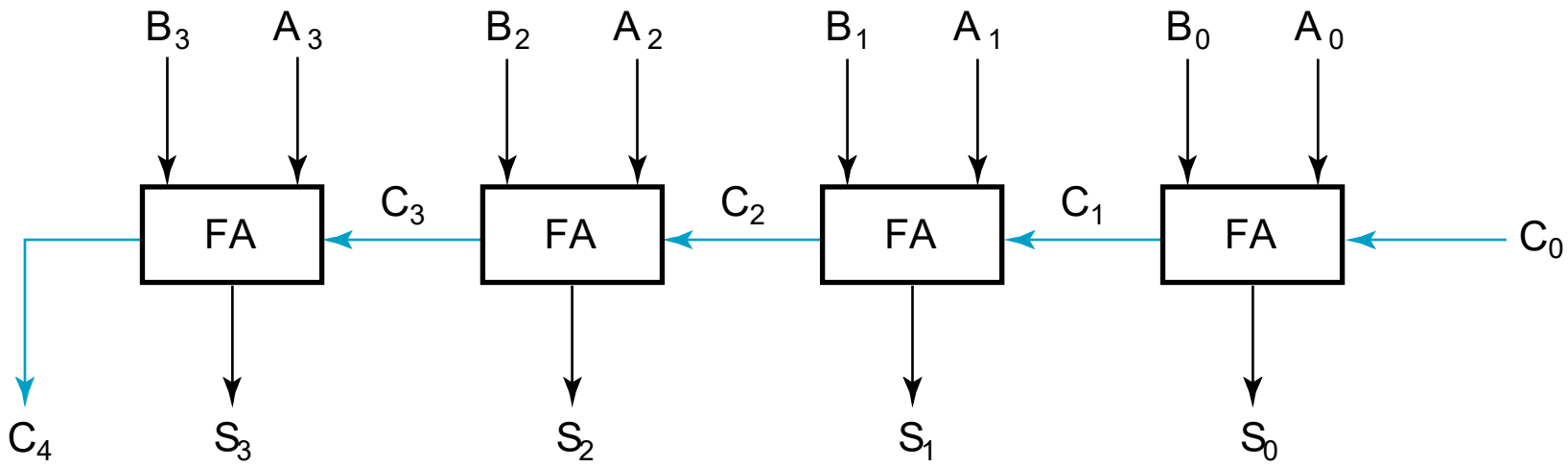
- **Same as the previous example:**

$$S = X\,\bar{Y}\,\bar{Z} + \bar{X}\,Y\,\bar{Z} + \bar{X}\,\bar{Y}\,Z + X\,Y\,Z$$

$$C = X\,Y + X\,Z + Y\,Z$$

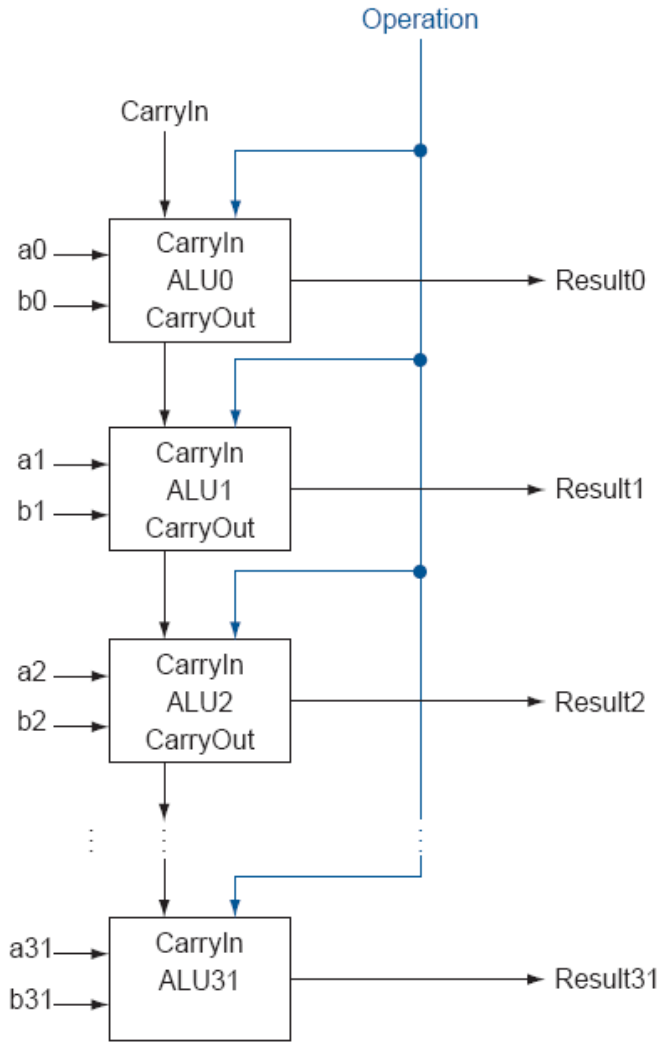
4-bit Ripple-Carry Binary Adder

- **A four-bit Ripple Carry Adder made from four 1-bit Full Adders:**



32-bit Ripple Carry Adder

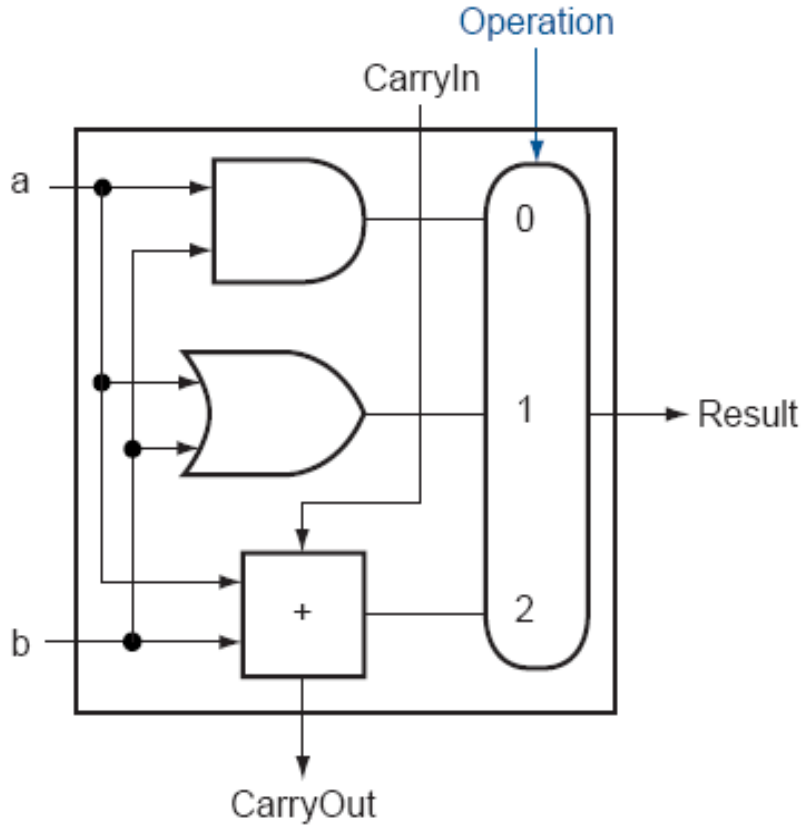
1-bit ALUs are connected "in series" with the carry-out of 1 box going into the carry-in of the next box



Designing a Simple ALU (Arithmetic Logic Unit)

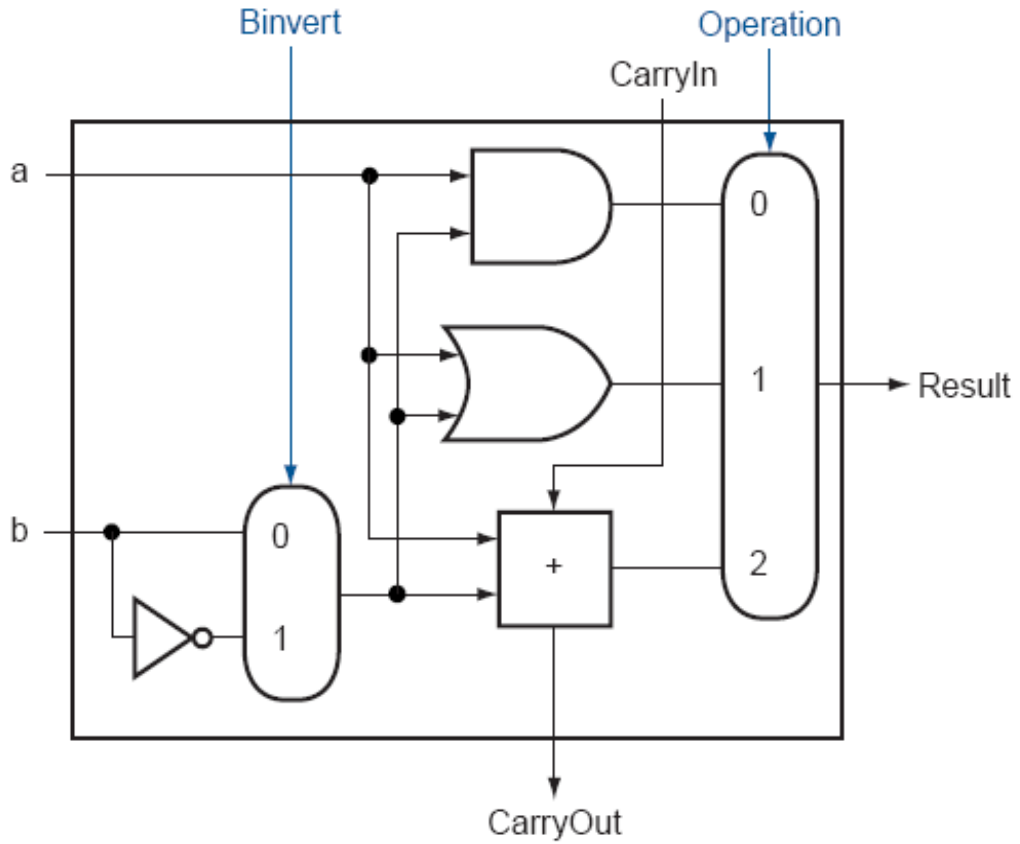
1-Bit ALU with Add, Or, And

- **Multiplexor selects between Add, Or, And operations**

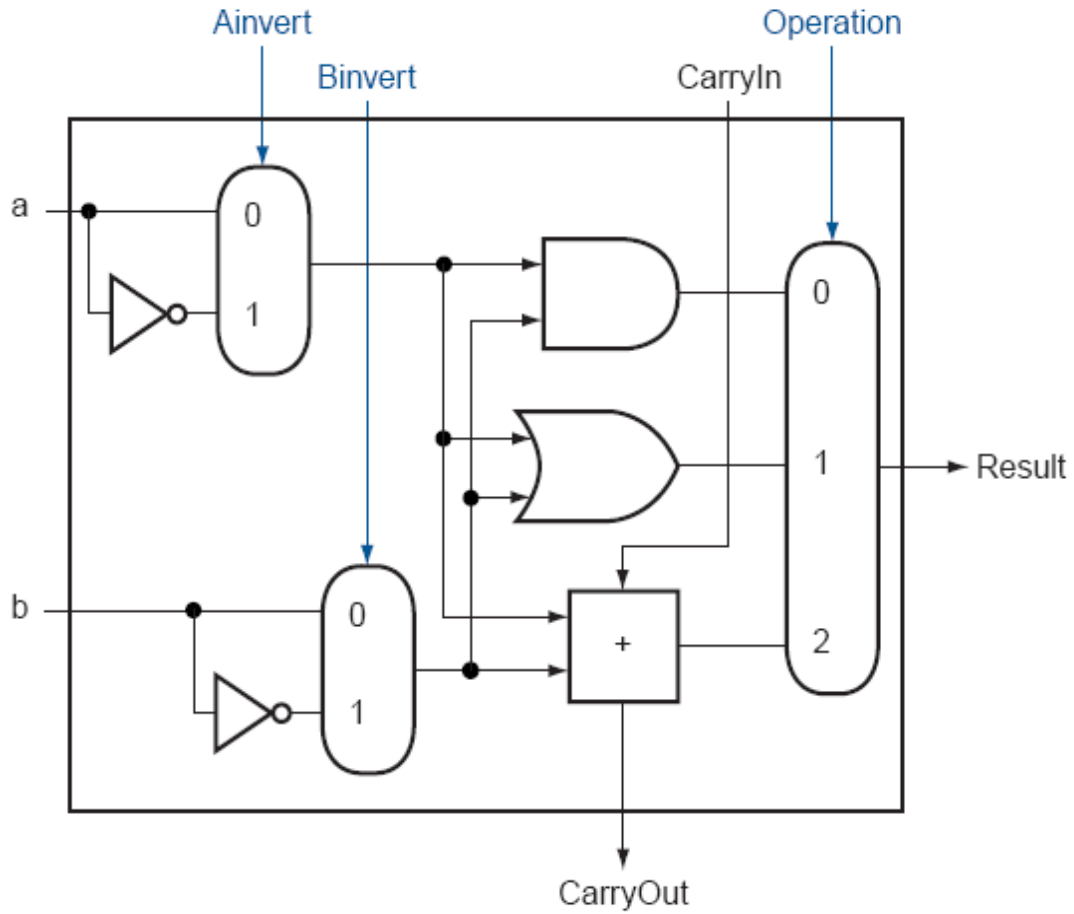


Incorporating Subtraction

- **Must invert bits of B and add a 1**
  - Include an inverter
- **CarryIn for the first bit is 1**
- **The CarryIn signal (for the first bit) can be the same as the Binvert signal**

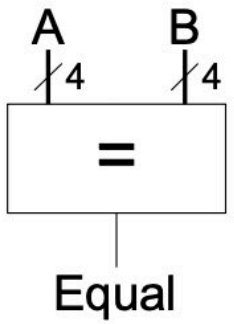


Incorporating NOR

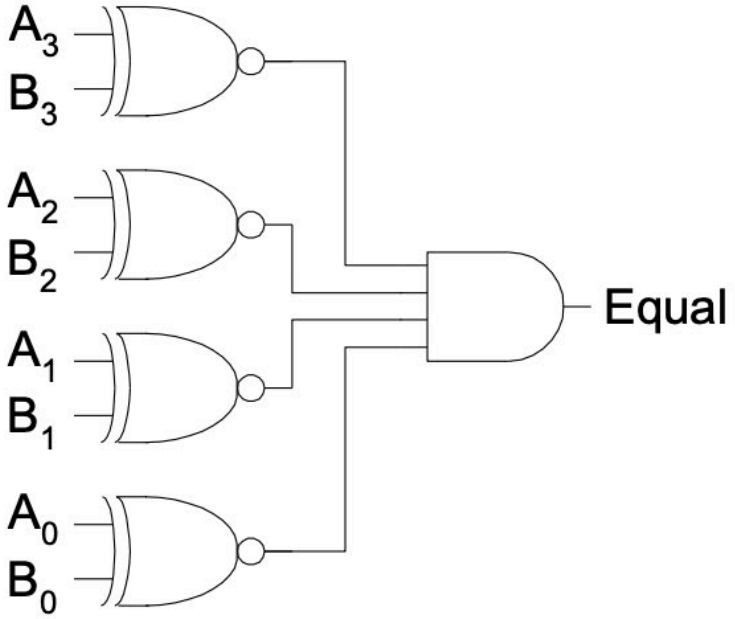


Comparator: Equality

Symbol



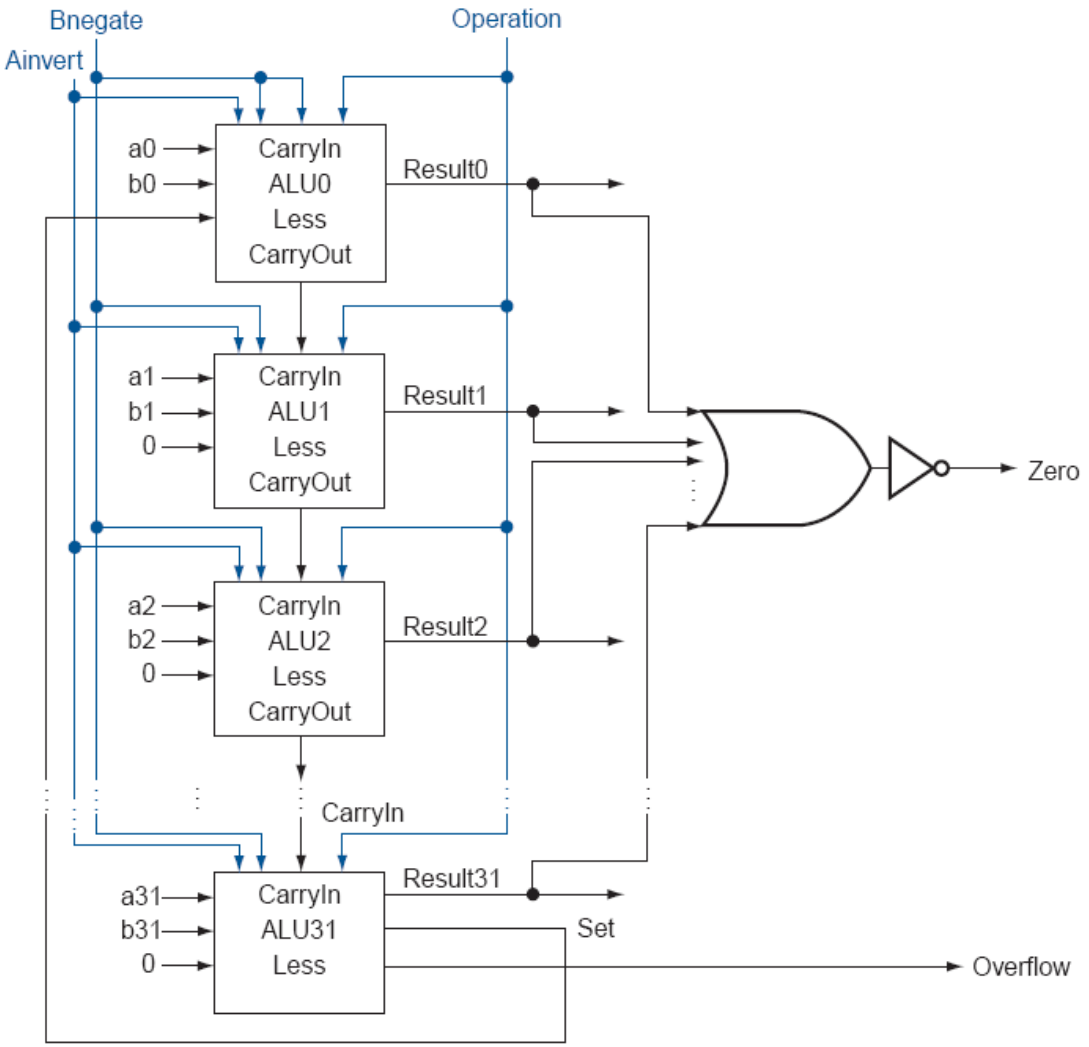
Implementation



- **Perform a – b and confirm that the result is all zero's**

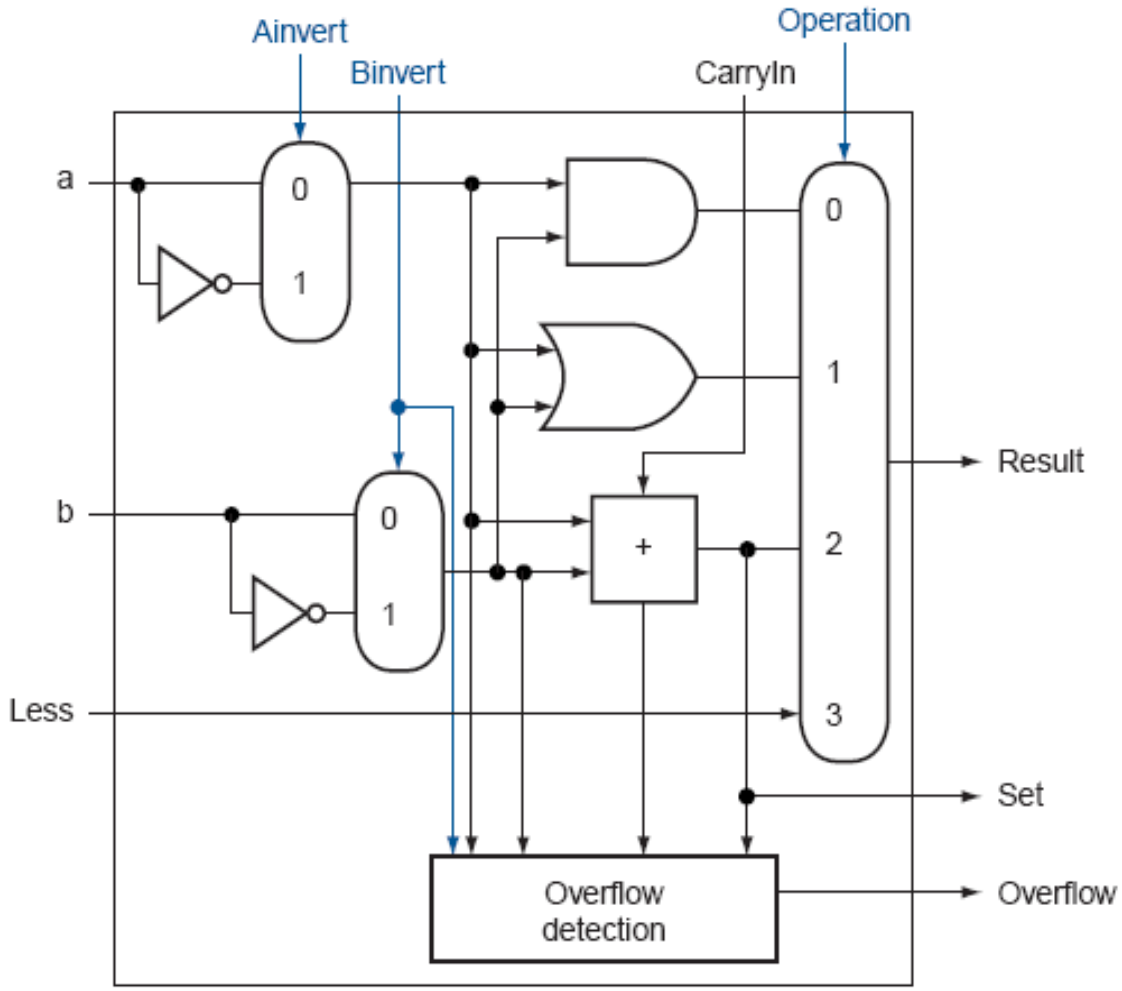
Incorporating beq

- **Perform a – b and**
  - Confirm that the
  - Result is all zero's



Incorporating slt

- Perform a – b and check the sign
- New signal (Less) that is zero for ALU boxes 1-31
- The 31st box has a unit to detect overflow and sign – the sign bit serves as the Less signal for the 0th box



Incorporating slt