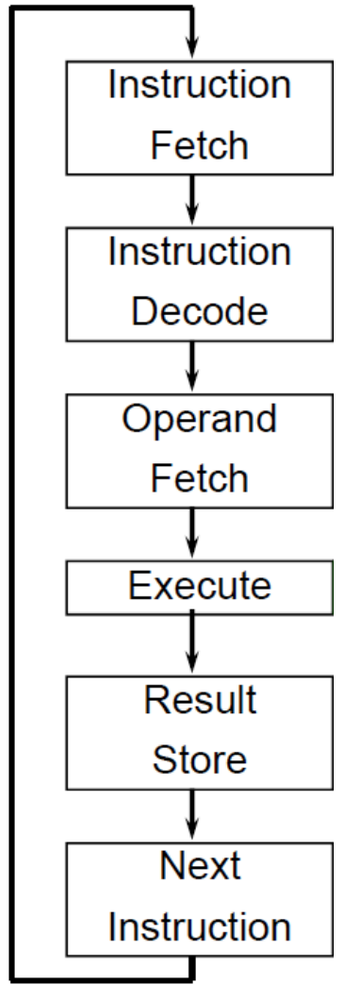
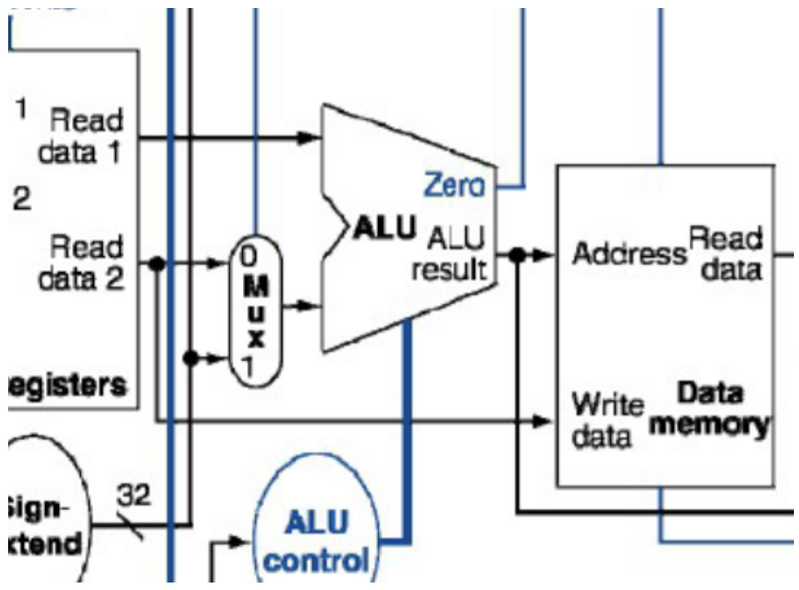
EE116C/CS151B
Fall 2017

Instructor: Professor Lei He
TA: Yuan Liang

Review

- **Arithmetic Logic Unit (ALU) Design**
  - Input: two operand (32-bits), ALU control signal
  - Output: Zero, ALU results, overflow, carryout





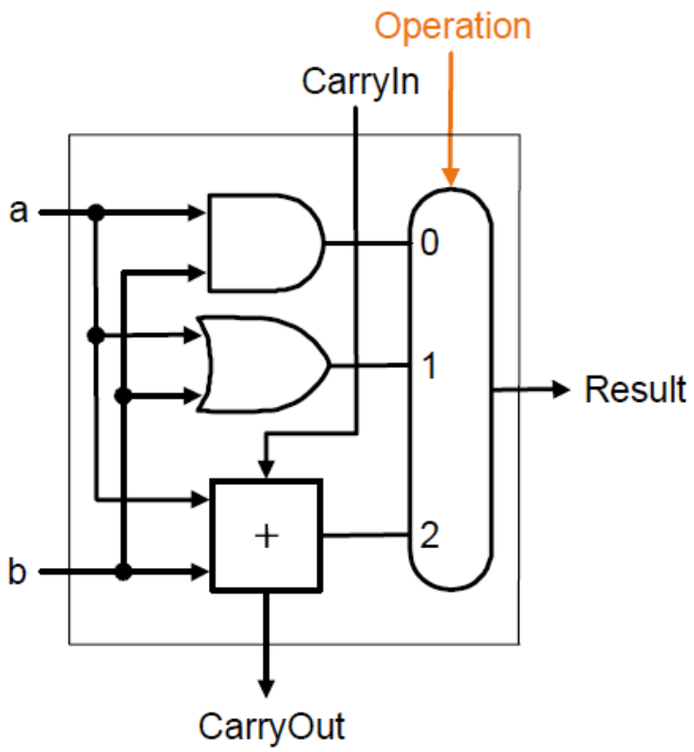
Review

- **Arithmetic Logic Unit (ALU) Design**
  - Function:
    - Add
    - Subtract
    - Overflow dealing
    - And
    - Or
    - Branch
    - Set-On-Less-Than (SLT)

- **How?**

Review

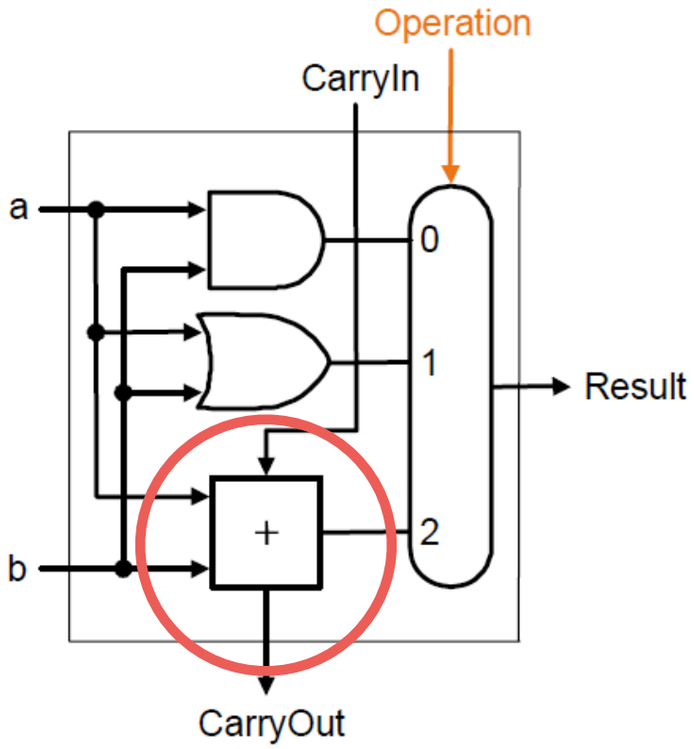
- **1-bit ALU**
  - Input: two operand (1-bits), ALU control signal, **carryin**
  - Output: Zero, ALU results, ~~**overflow**~~, carryout



Operation?
and,
or,
add with carryin,
etc.

Review

- **1-bit ALU**
  - Input: two operand (1-bits), ALU control signal, **carryin**
  - Output: Zero, ALU results, **overflow**, carryout



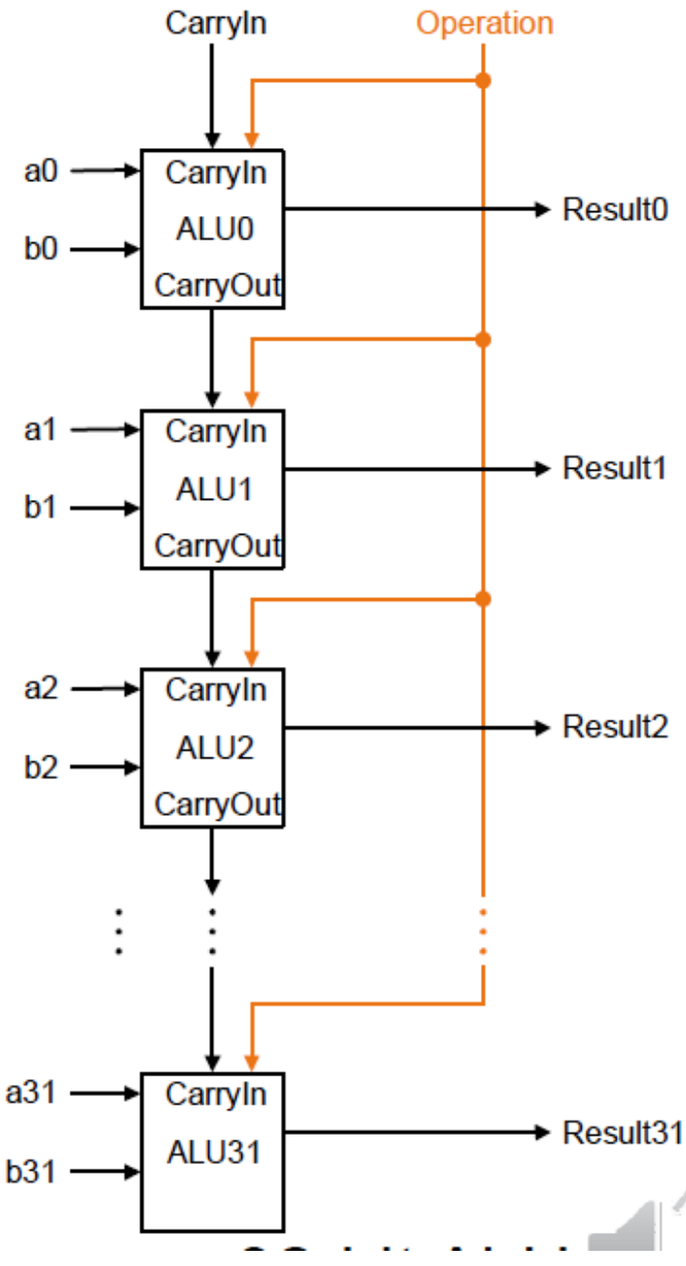
Composed of AND, OR, NOT gates.

CarryOut = (b & CarryIn) | (a & CarryIn) | (a & b)

Sum = (!a & !b & CarryIn) | (!a & b & !CarryIn)
| (a & !b & !CarryIn) | (a & b & CarryIn)

Review

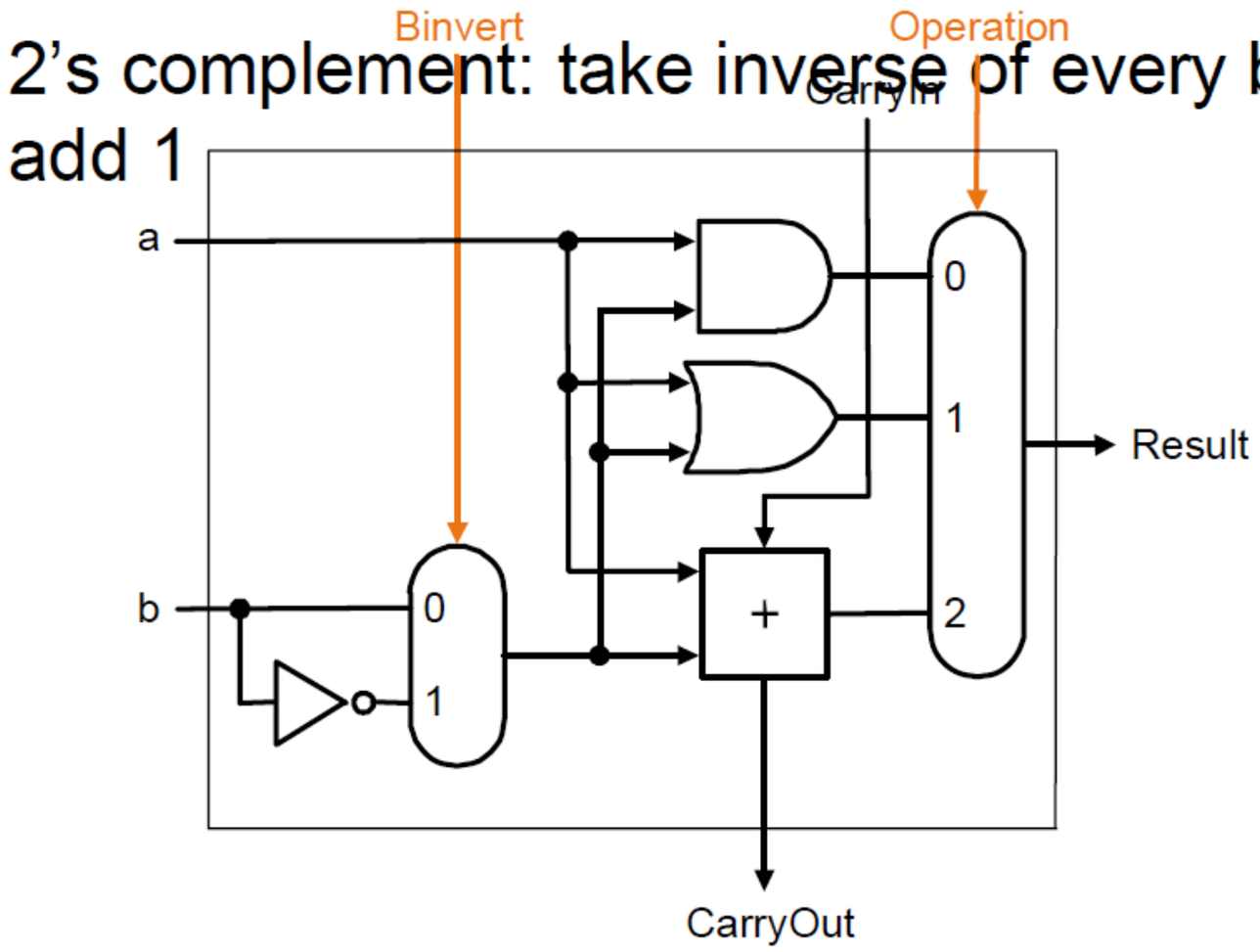
- **Style 1: Ripple Carry Adder (1-bit ALU -> 32-bit ALU)**



- **Function:**
  - **Add**
  - Subtract
  - Overflow dealing
  - **And**
  - **Or**
  - Branch
  - Set-On-Less-Than (SLT)

Review

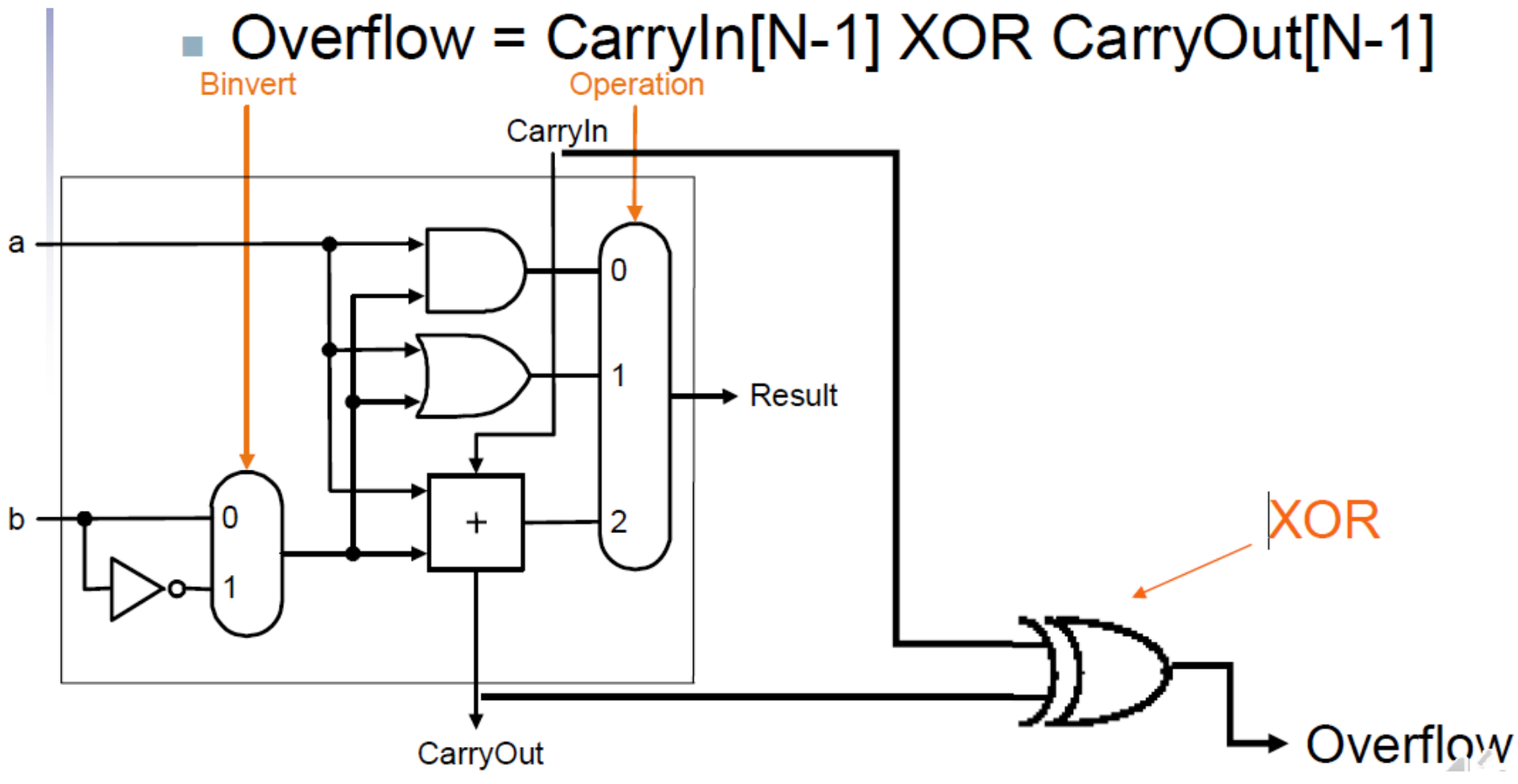
- **Style 1: Ripple Carry Adder (1-bit ALU -> 32-bit ALU)**



- **Function:**
  - **Add**
  - **Subtract**
  - Overflow dealing
  - **And**
  - **Or**
  - Branch
  - Set-On-Less-Than (SLT)

Review

- **Style 1: Ripple Carry Adder (1-bit ALU -> 32-bit ALU)**



only for the most significant bit ALU

- **Function:**
  - **Add**
  - **Subtract**
  - **Overflow dealing**
  - **And**
  - **Or**
  - Branch
  - Set-On-Less-Than (SLT)

Review

- **Style 1: Ripple Carry Adder (1-bit ALU -> 32-bit ALU)**
  - bneq r1, r2, L
  - beq r1, r2, L
  - if (a == b)
  - if (a != b)
  - -> a - b ?= 0

One big NOR gate including all N results.

if any result(i) == 1 -> return 0
if all result(i) == 0 -> return 1

- **Function:**
  - **Add**
  - **Subtract**
  - **Overflow dealing**
  - **And**
  - **Or**
  - **Branch**
  - Set-On-Less-Than (SLT)

Review

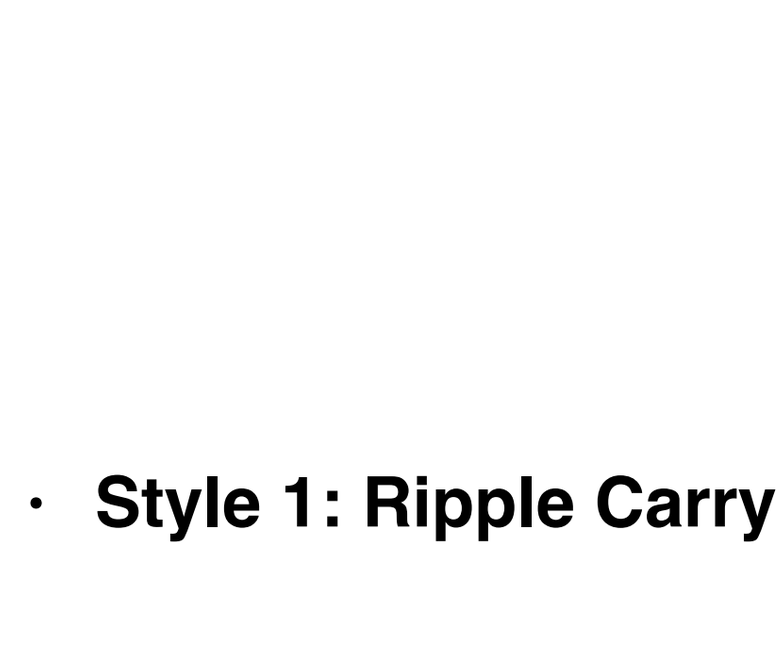
- **Style 1: Ripple Carry Adder (1-bit ALU -> 32-bit ALU)**
  - slt rd, rs, rt
    - if (rs < rt) rd = 1
    - else rd = 0

If R[rs] is input A, and R[rt] is input B, we can subtract, and then take the most significant bit of the result.
- If MSB = 1, then the difference is negative and thus R[rs] is less than R[rt]

- **Function:**
  - **Add**
  - **Subtract**
  - **Overflow dealing**
  - **And**
  - **Or**
  - **Branch**
  - **Set-On-Less-Than (SLT)**

Review

- **Style 1: Ripple Carry Adder (1-bit ALU -> 32-bit ALU)**
  - **Function:**
    - **Add**
    - **Subtract**
    - **Overflow dealing**
    - **And**
    - **Or**
    - **Branch**
    - **Set-On-Less-Than (SLT)**

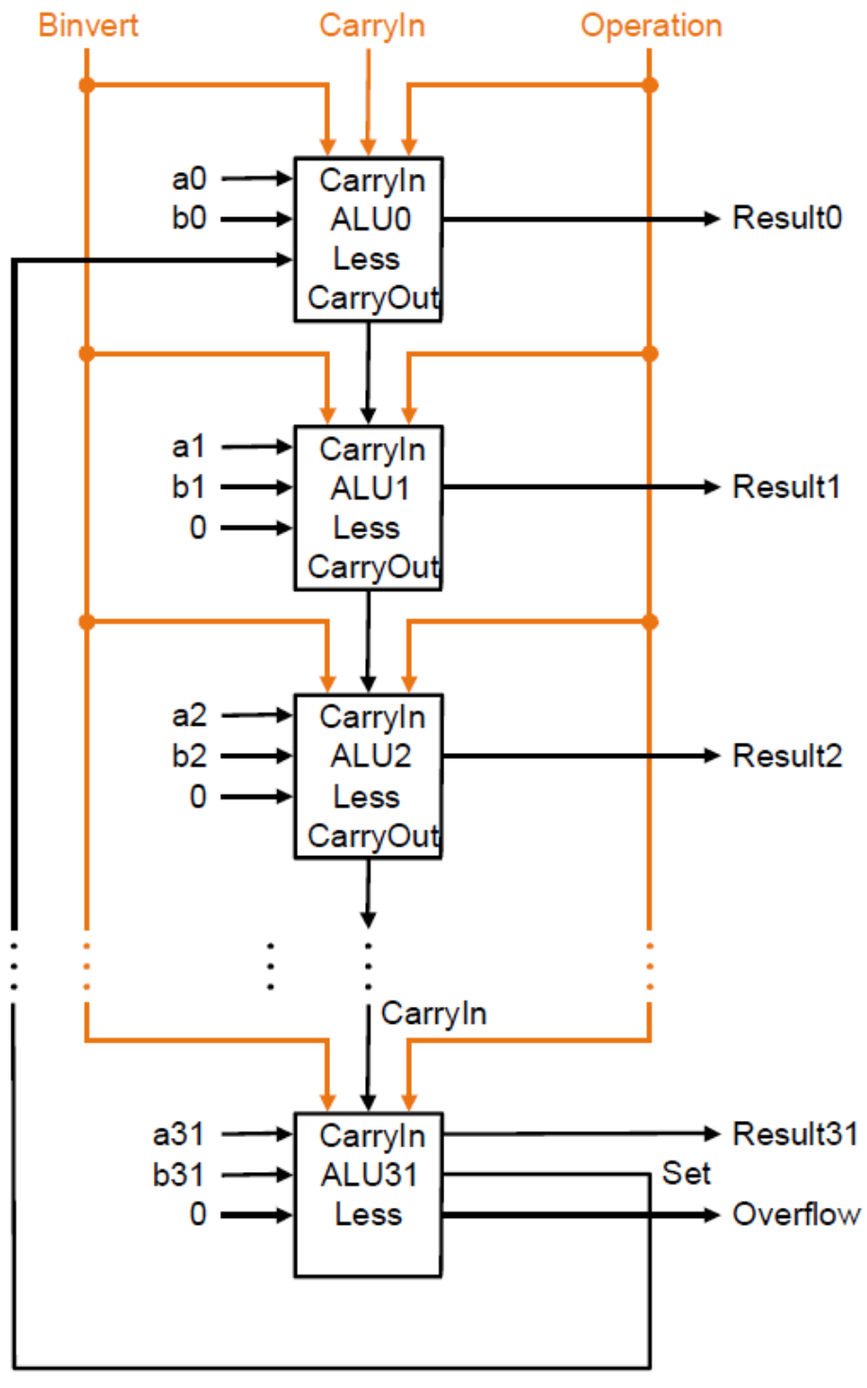


All but MSB

Most Significant Bit

Review

- **Style 1: Ripple Carry Adder (1-bit ALU -> 32-bit ALU)**

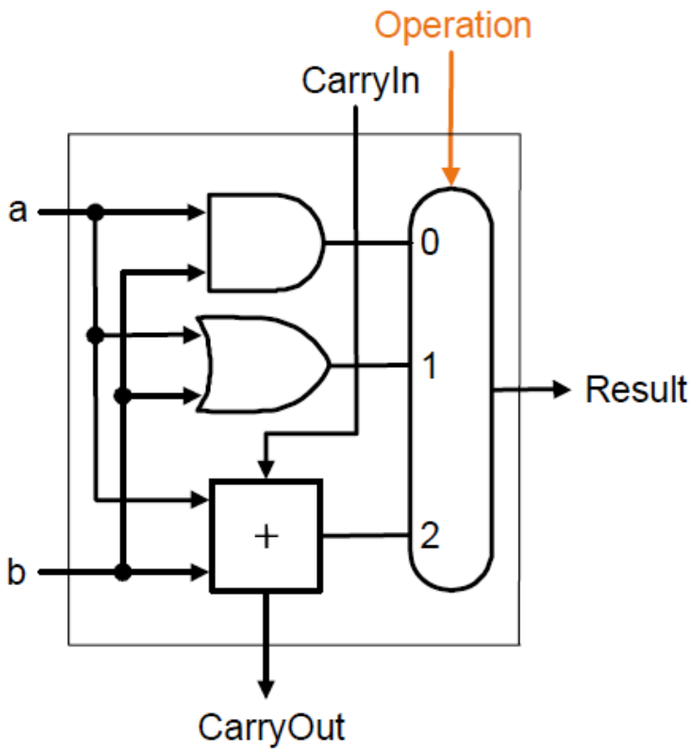


- **Function:**
  - **Add**
  - **Subtract**
  - **Overflow dealing**
  - **And**
  - **Or**
  - **Branch**
  - **Set-On-Less-Than (SLT)**

Review

- **Discussion: ripple style 32-bit adder**

- Worst case delay for N-bit Ripple Carry Adder



CarryOut = (b & CarryIn) | (a & CarryIn) | (a & b)

Delay = 2N

Sum = (!a & !b & CarryIn) | (!a & b & !CarryIn)
| (a & !b & !CarryIn) | (a & b & CarryIn)

Delay = 2N

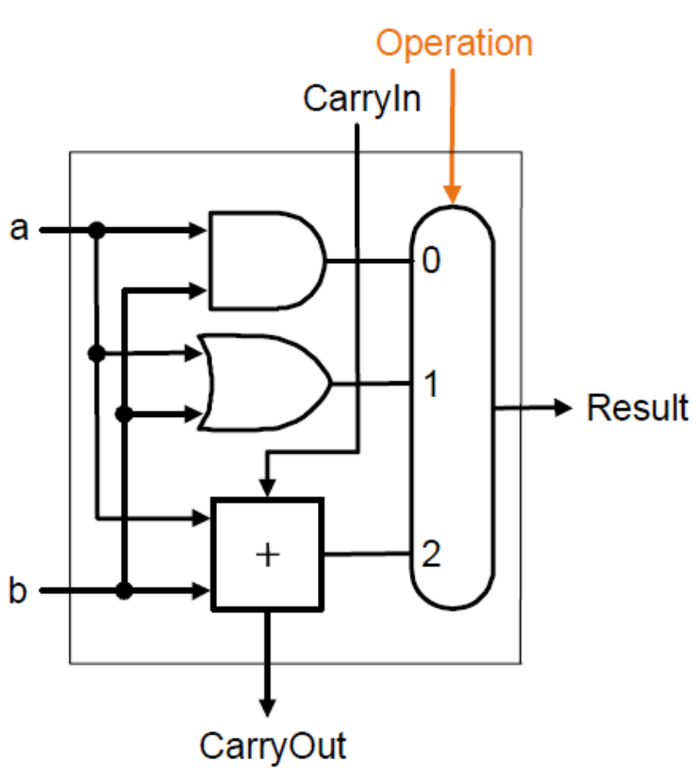
Total delay = 32 * 2N

Review

- **Style 2: Carry Lookahead Adder (1-bit ALU -> 32-bit ALU)**
    - The principle of the Carry Lookahead Adder is to sacrifice space and complexity in the interest of speed.
    - For calculating Cout, we will use two additional signals at 1-bit adder:
        - Generate (Gn) = An * Bn
        - Propagate (Pn) = An ^ Bn

Review

- **Style 2: Carry Lookahead Adder (1-bit ALU -> 32-bit ALU)**



Ripple Carry Adder:

C1 = (A0 * B0) + (A0 * C0) + (B0 * C0)

2T + 3T

Carry Lookahead Adder:

G0 = A0 * B0

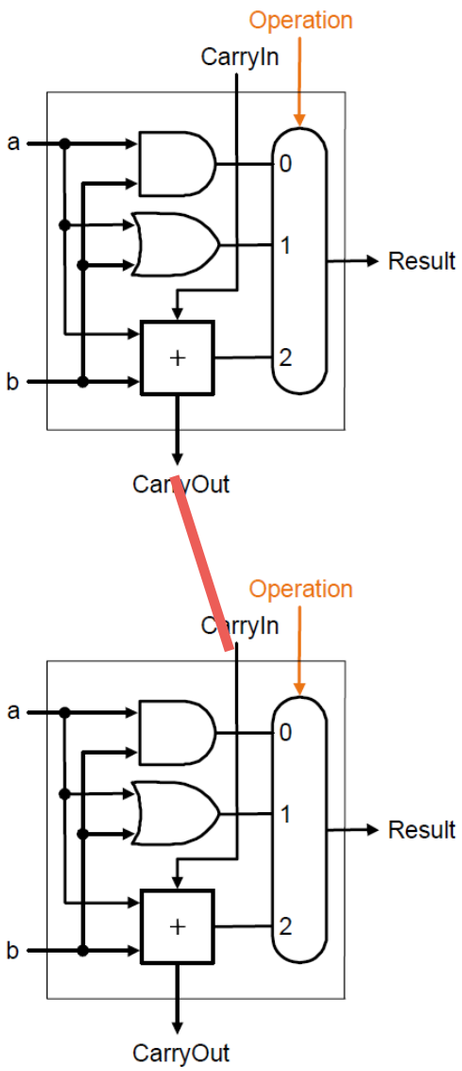
P0 = A0 ^ B0

C1 = G0 + C0 * P0

2T + 2T + 2T

Review

- **Style 2: Carry Lookahead Adder (1-bit ALU -> 32-bit ALU)**



Ripple Carry Adder:

C1 = (A0 * B0) + (A0 * C0) + (B0 * C0)

C2 = (A1 * B1) + (A1 * C1) + (B1 * C1)

2T + 3T + 2T + 3T

Carry Lookahead Adder:

G0 = A0 * B0

P0 = A0 ^ B0

C1 = G0 + C0 * P0

G1 = A1 * B1

P1 = A1 ^ B1

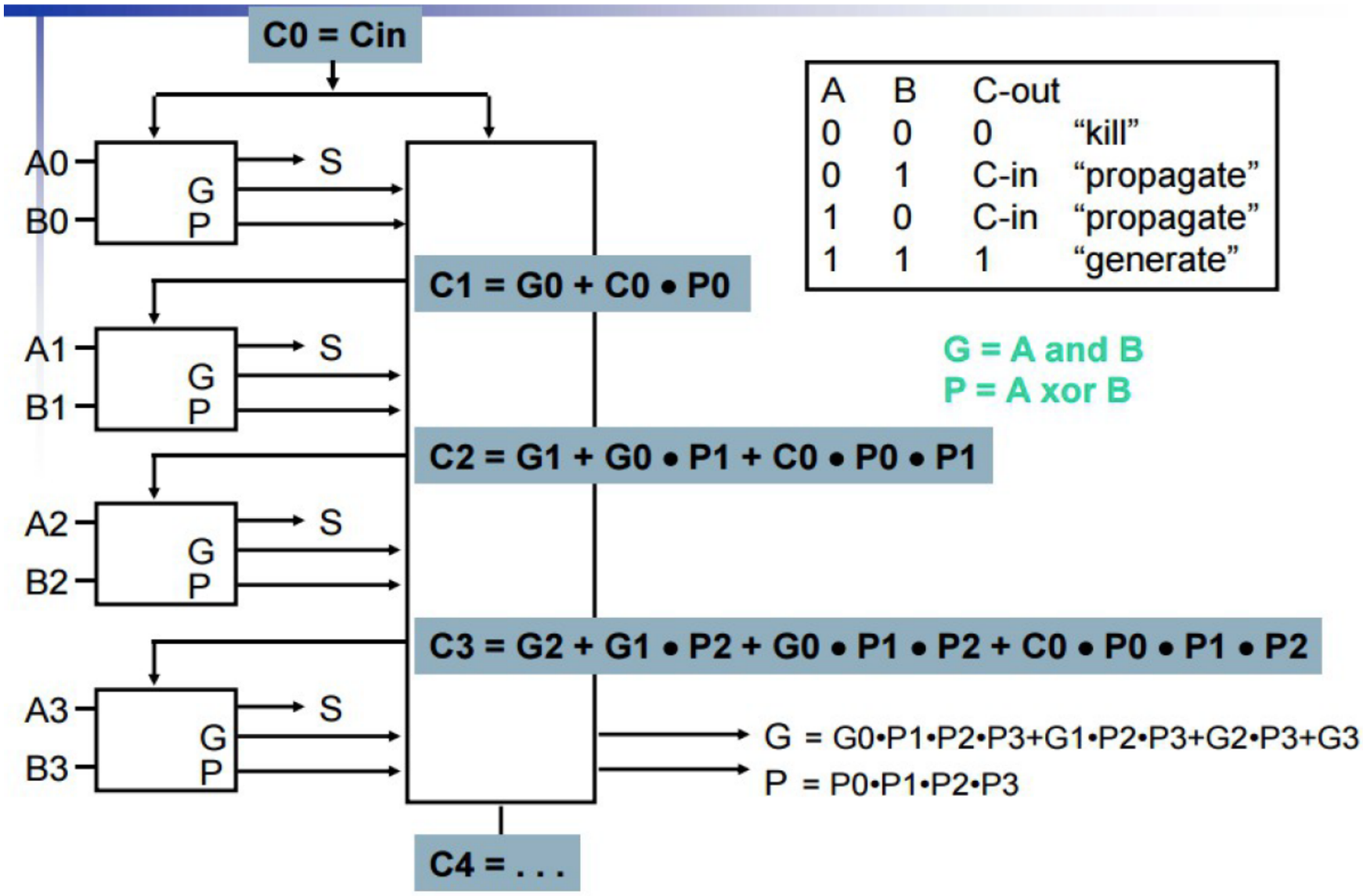
C1 = G1 + C1 * P1 = G1 + (G0 + C0 * P0) * P1

= G1 + G0 * P1 + C0 * P0 * P1

2T + 3T + 3T

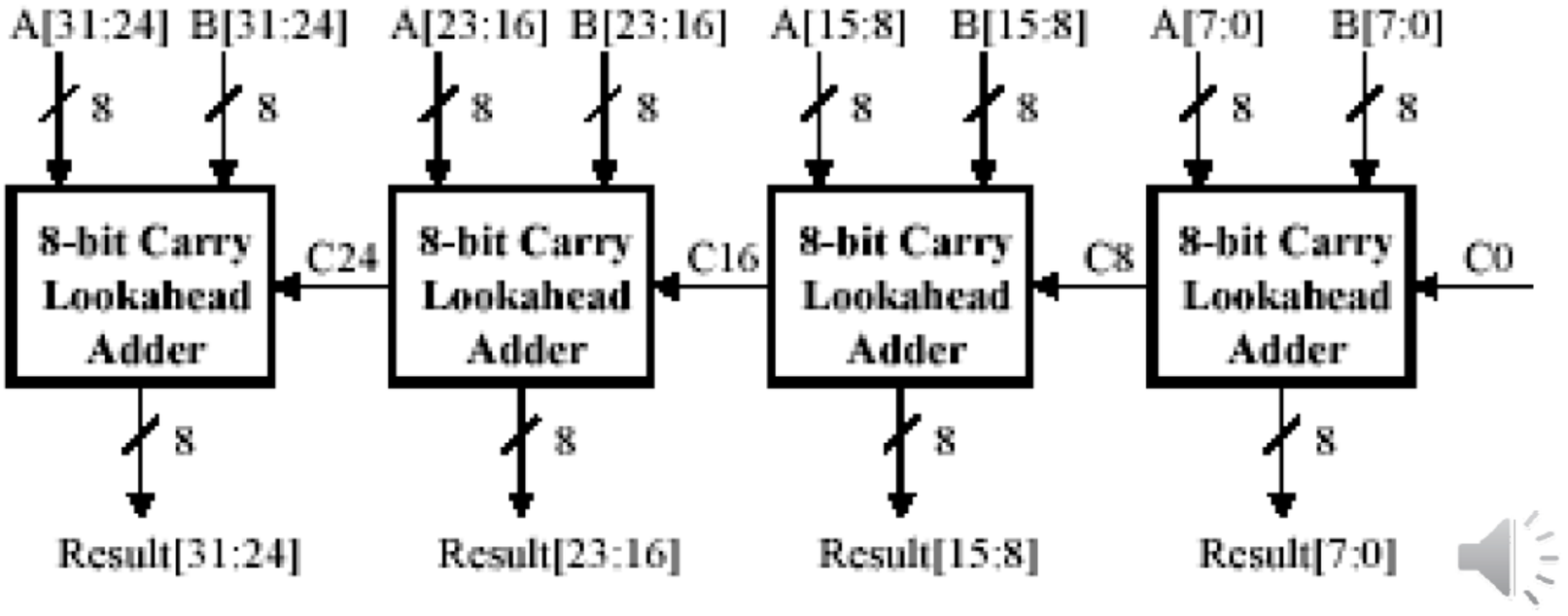
Review

- **Style 2: Carry Lookahead Adder (1-bit ALU -> 32-bit ALU)**



Review

- **Style 2.1: [Partial] Carry Lookahead Adder**
  - Connect several N-bit Lookahead Adders together
  - Four 8-bit carry lookahead adders can form a 32-bit partial carry lookahead adder



Review

- **Style 2.2: [Hierarchical] Carry Lookahead Adder**