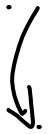
ALU Construction

(ALU) Arithmetic Logic Unit



does computation:

- add, sub
- or, and (logical)
- register comparison for branches
- address computation for memory instructions

Steps

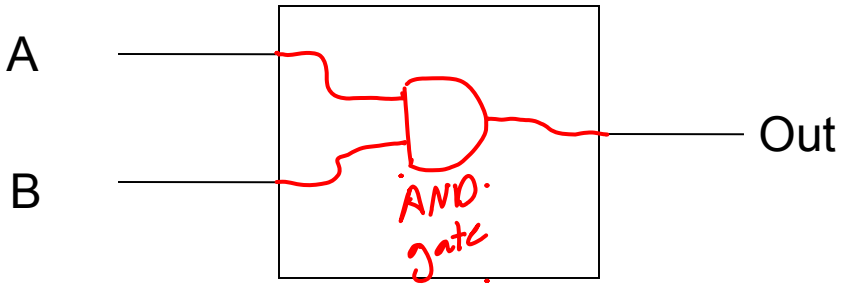
1. Pick operations to support.
2. Design a 1-bit ALU.

Decide which operations to support

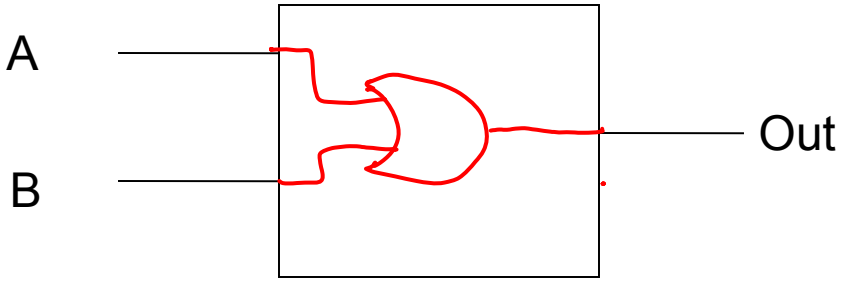
Operation	OpCode
AND	00
OR	01
Add	10
Sub	11

- Choose operations
- Assign op-codes

Design 1-bit for each operation And & Or



A & B



A | B

Design 1-bit for each operation

Add

inputs | outputs

A	B	Cin	Cout	Out
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

Carry in

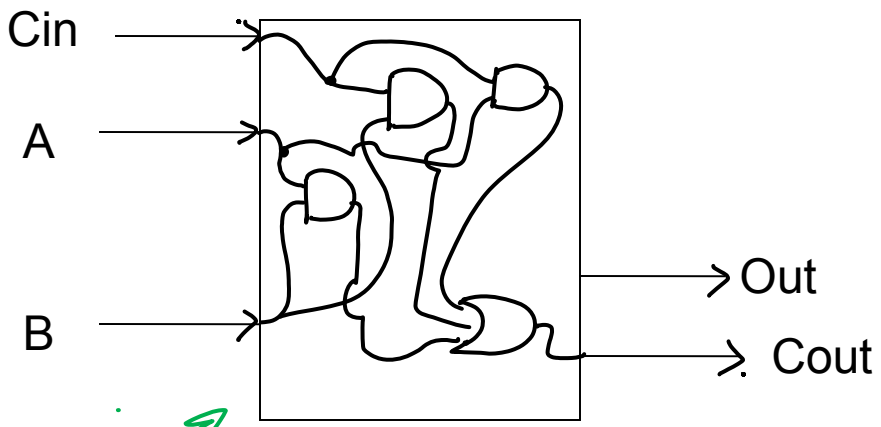
$$C_{out} = (A \cdot B) + (B \cdot C_{in}) + (A \cdot C_{in})$$

(AND, OR)

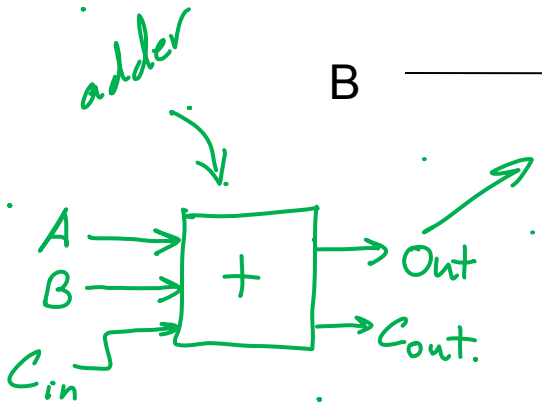
$$\begin{array}{r} 0\ 1 \\ +\ 1\ 1 \\ \hline 0 \end{array}$$

Out

$Out = A \text{ XOR } B \text{ XOR } C_{in}$



01100
+00011



A + B

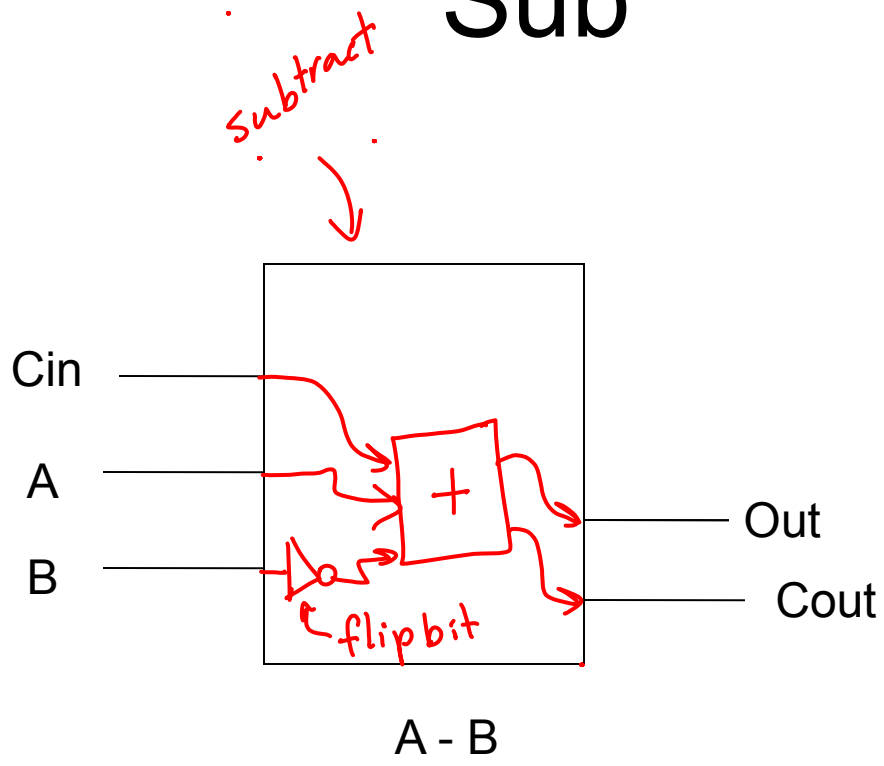
Design 1-bit for each operation

Sub

A - B

A + (-B)

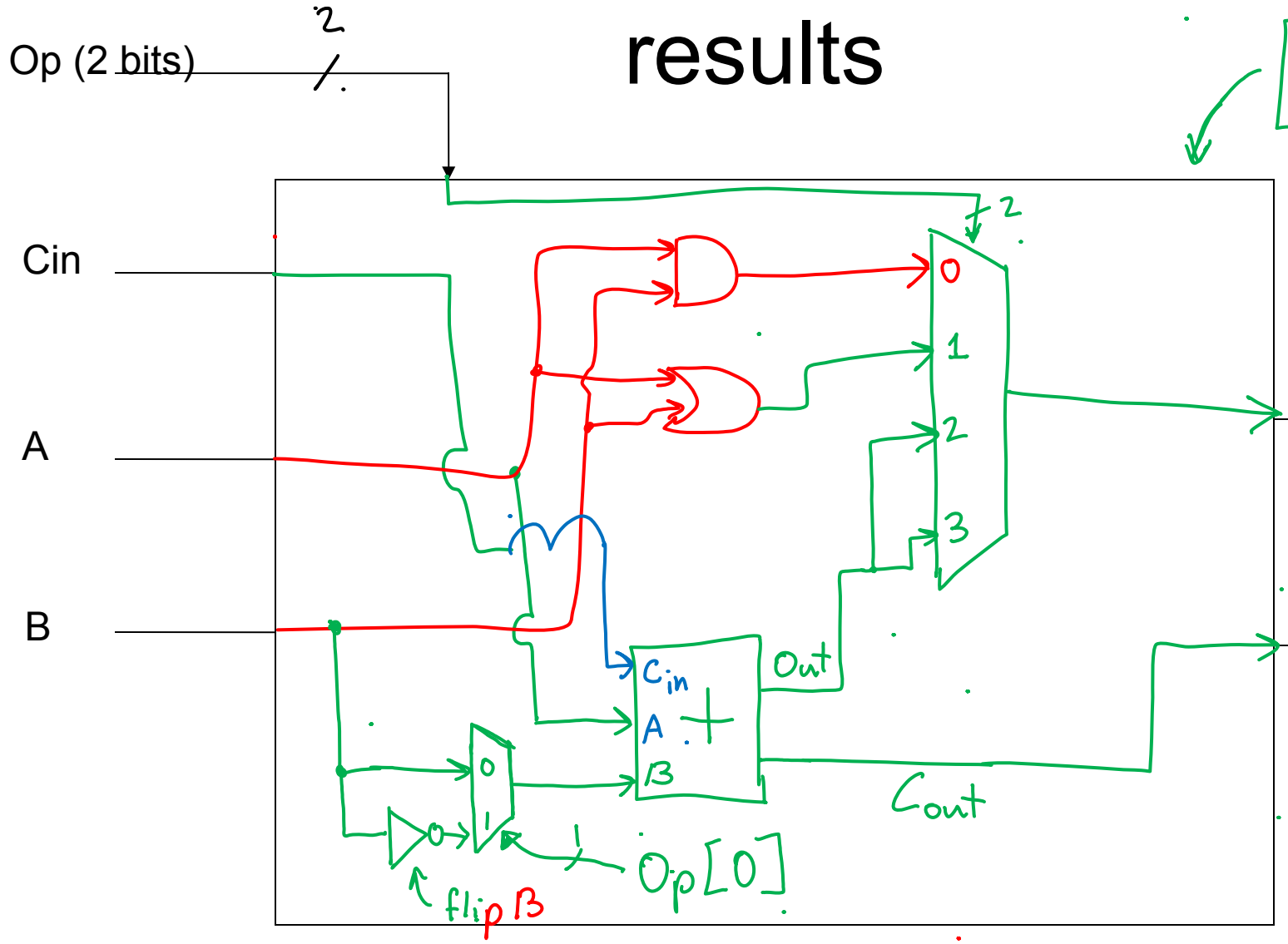
1. flip bits
2. add 1



0110

- 0011 ➔

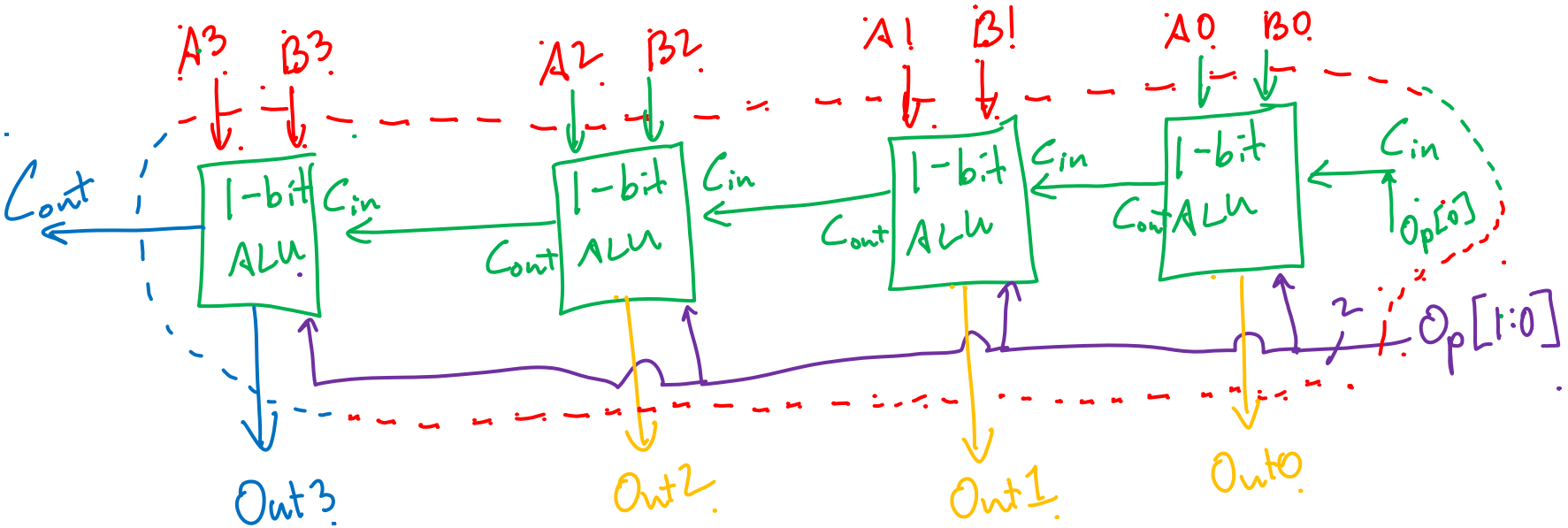
Use MUX to choose between results

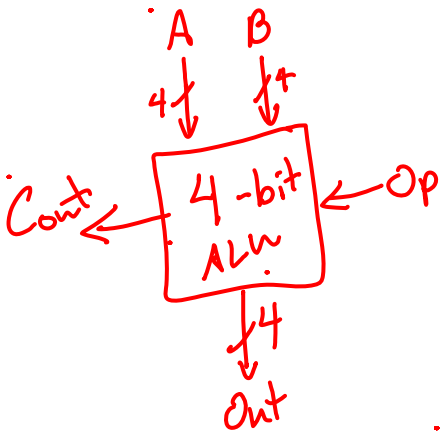


Op[1], Op[0]

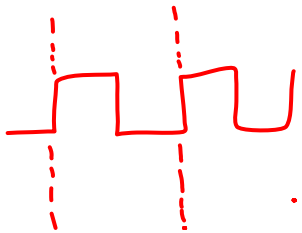
Op	Code
AND	00
OR	01
Add	10
Sub	11

Design logic to connect ALU





∴ Ripple Carry Adder



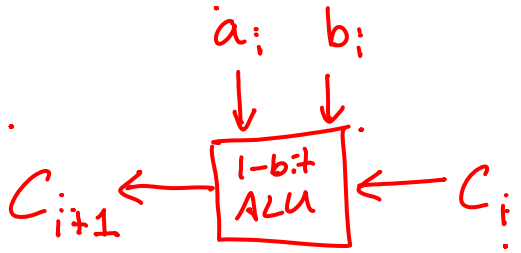
Adder Design

Problem: Ripple Carry adder is slow

Solution: compute carry information as quickly as possible

Carry Look-ahead (CLA)

We already know:



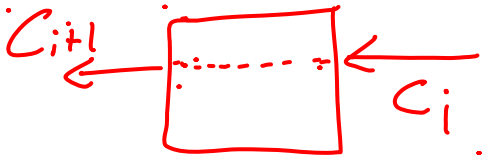
$$c_{i+1} = \overline{a_i \cdot b_i} + \overline{a_i \cdot c_i} + \overline{b_i \cdot c_i}$$

$$= a_i \cdot b_i + c_i \cdot (a_i + b_i)$$

$$= G_i + c_i \cdot P_i$$

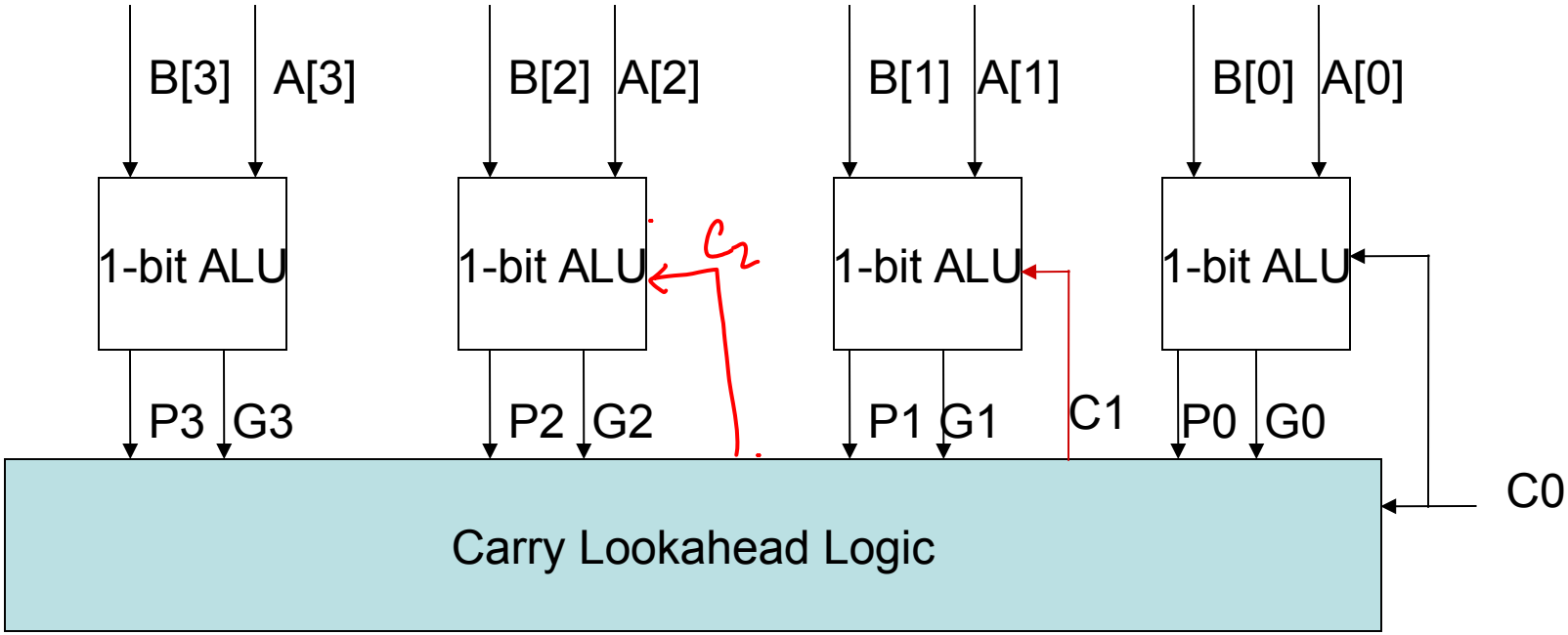
G_i: generate (this alu will generate a carry)

P_i: propagate (this alu will propagate a carry in)



- G_i and P_i can be computed in parallel

Carry Look-Ahead



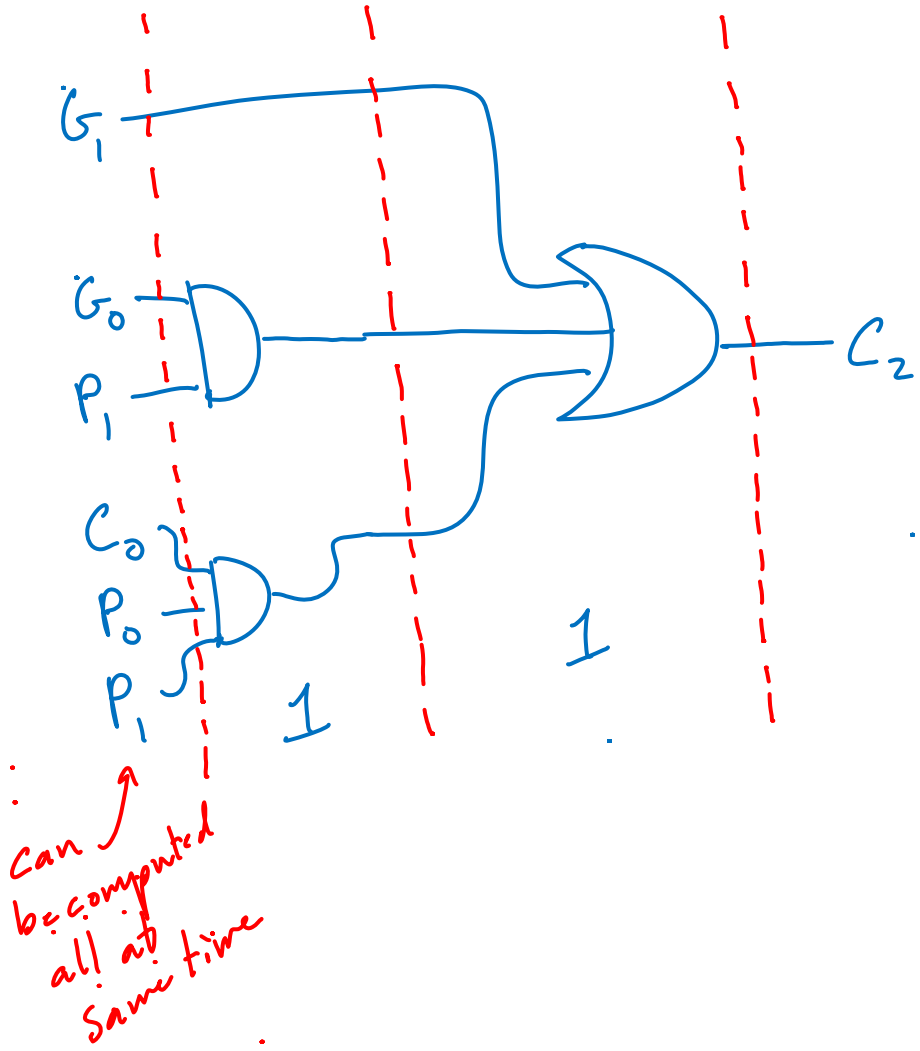
$C_1 = G_0 + C_0 \cdot P_0$

$C_2 = G_1 + C_1 \cdot P_1$

$= G_1 + (G_0 + C_0 \cdot P_0) \cdot P_1 = \overline{G_1 + G_0 \cdot P_1 + C_0 \cdot P_0 \cdot P_1}$

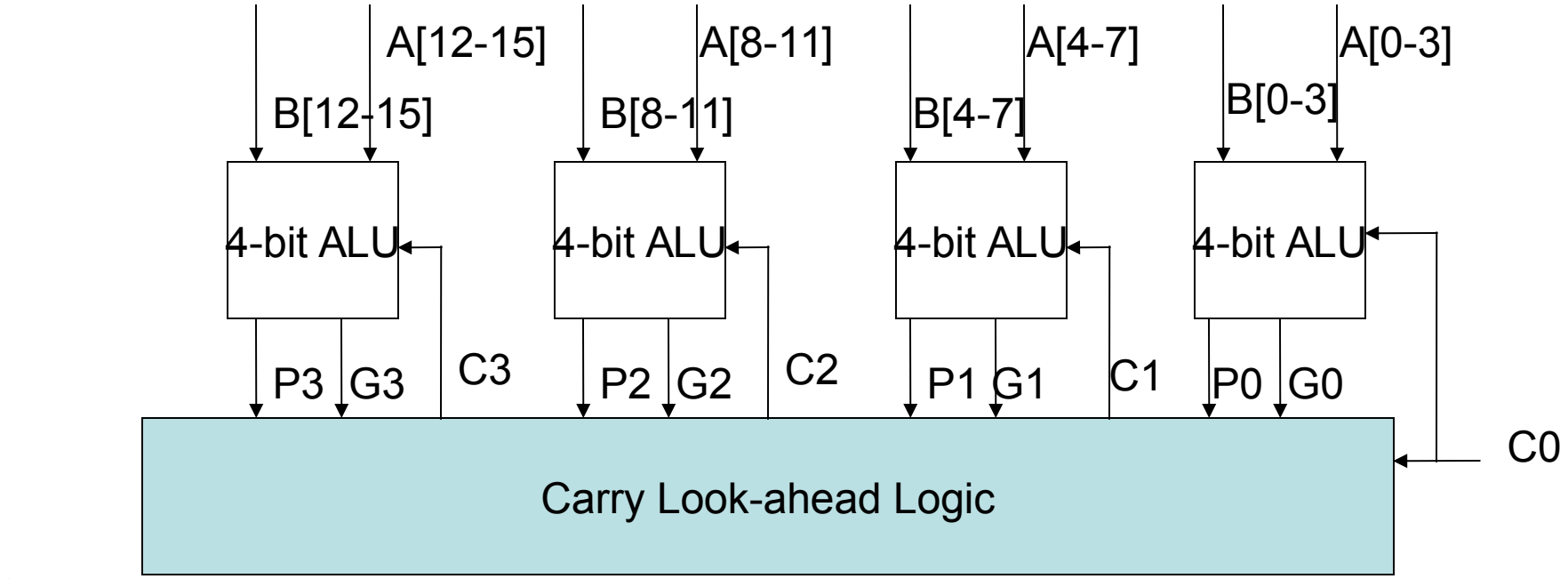
OR

Why is it faster?



- after G_i / P_i values are computed, roughly 2 gate delays

16-bit CLA Adder



Carry Select Adder.