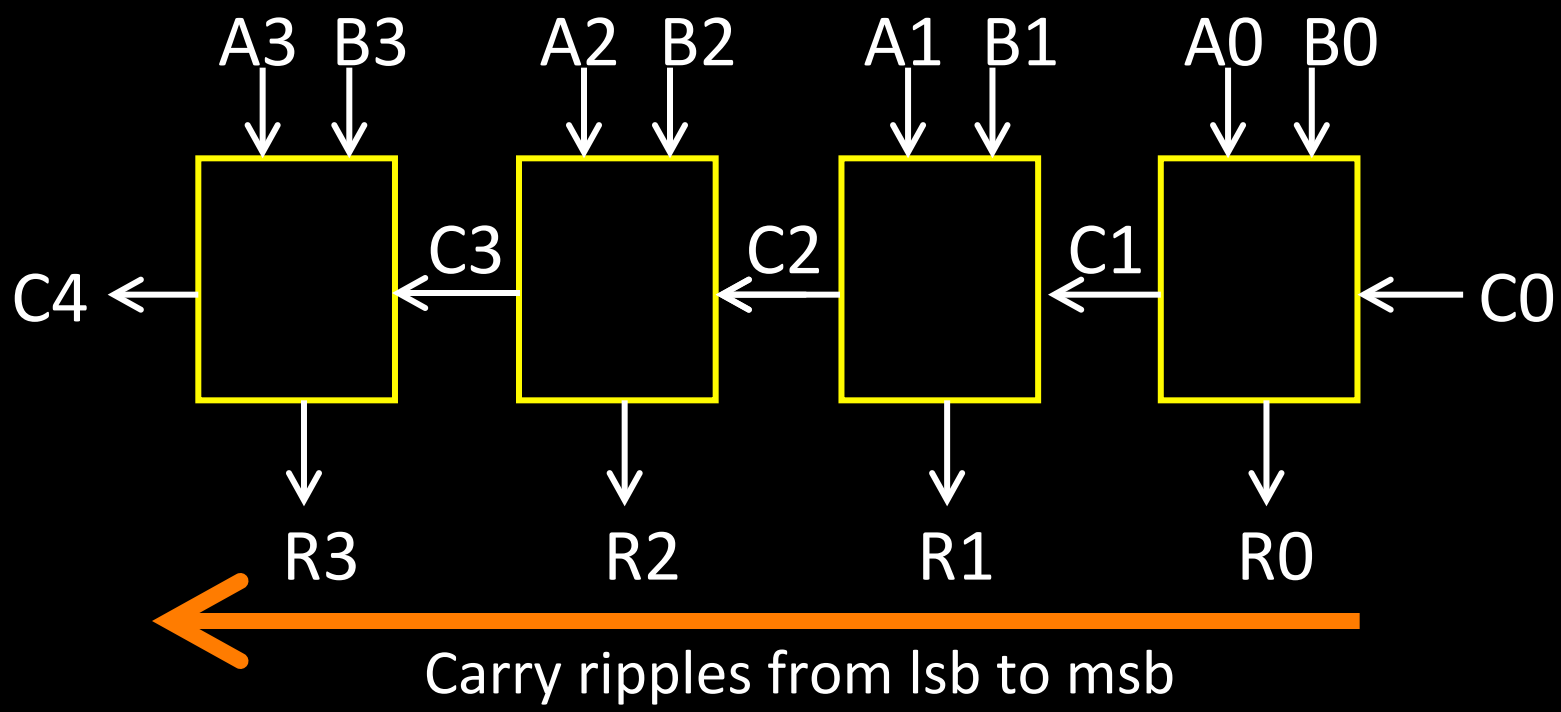
Fast Adders

See: P&H Chapter 3.1-3, C.5-6

Goals:

- serial to parallel conversion
- time vs. space tradeoffs
- design choices



- First full adder, 2 gate delay
- Second full adder, 2 gate delay
- …

Every bit needs to wait for carry in.

Q: Can we compute Cin earlier?

A: carry look-ahead adder (CLA)

For each bit, analyze situation independent of Cin

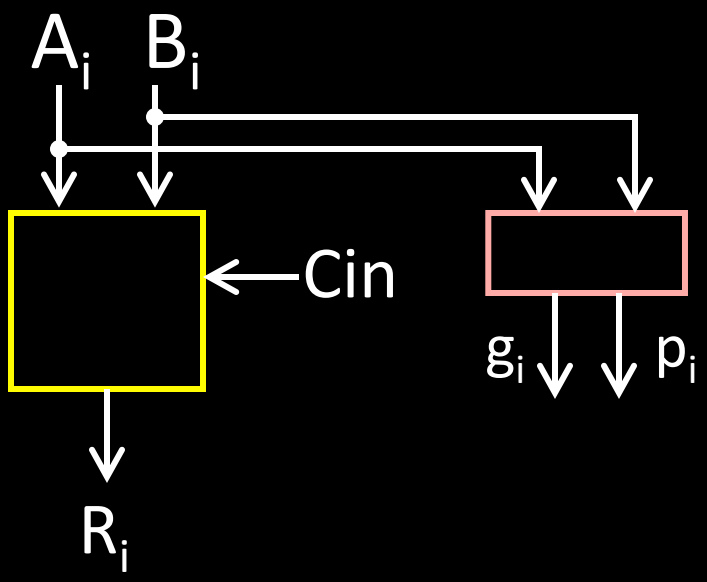
- Just based on (A,B) only

Q: When is Cout == 1, irrespective of Cin?

A: When A == 1 and B == 1
(this bit *generates* a carry, irrespective of Cin)

Q: When else might Cout == 1?

A: When A == 1 or B == 1, and Cin == 1
(this bit *propagates* carry from Cin to Cout)

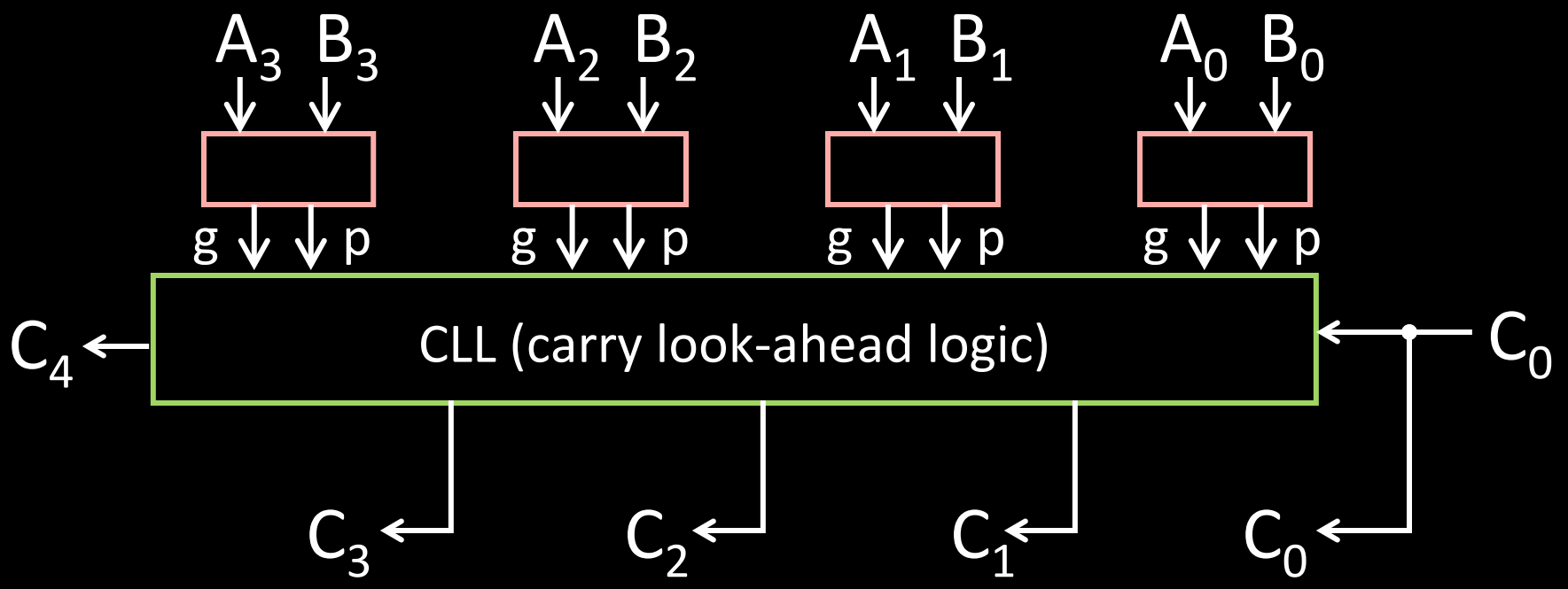


Invent two new terms: propagator, generator

- g == 1 means this bit generates carry, irrespective of Cin
  - g = AB
- p == 1 means this bit propagates Cin to Cout, but doesn't generate carry
  - p = A xor B

Performance?

- p and g generated in 1 gate delay after A and B arrive
- R is 2 gate delay after Cin arrives



CLL inputs: p,g from all 4 bits, C_0

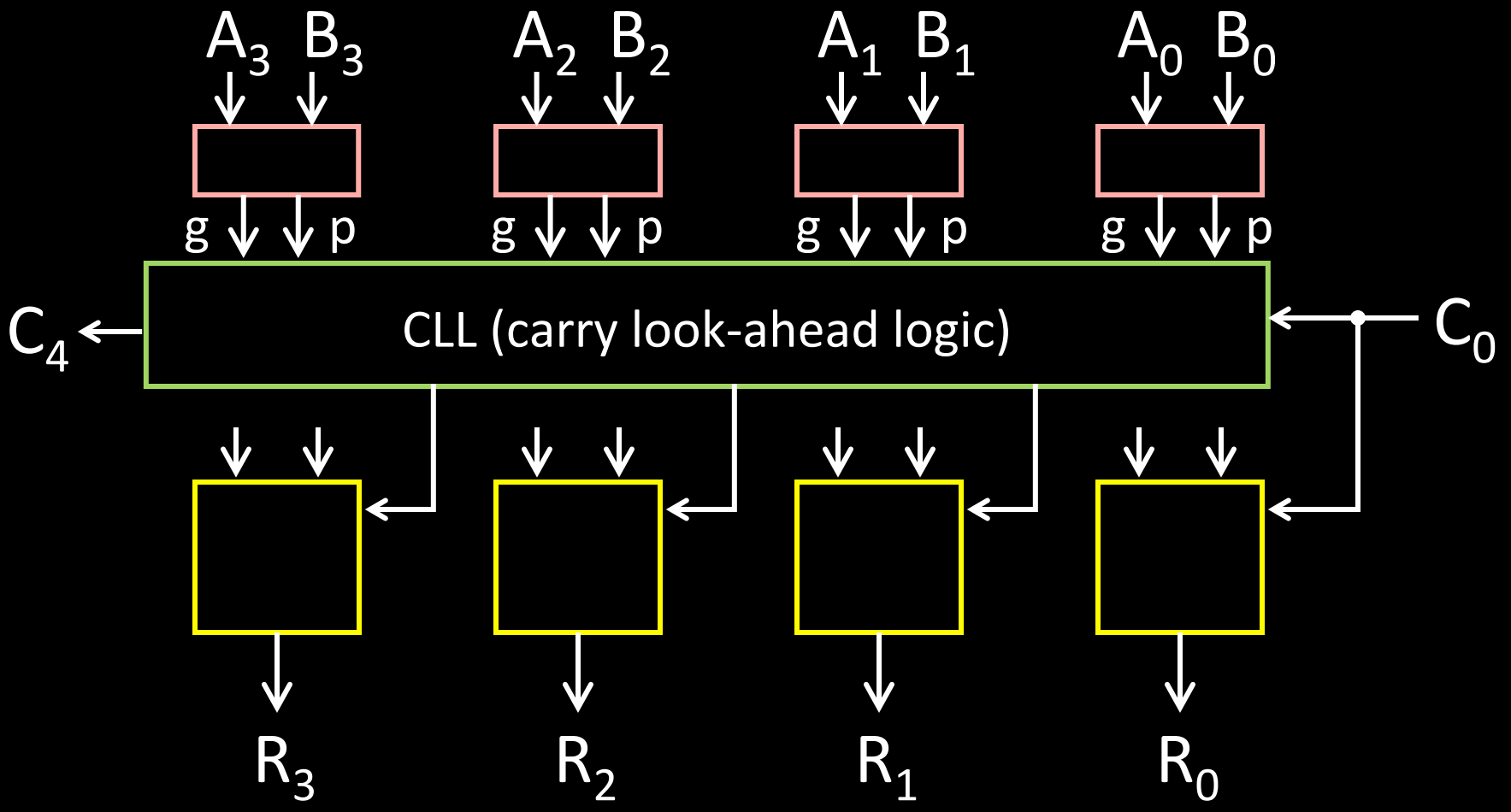
CLL outputs: all carry bits (using just 2 gate delays)

$C_1 = g_0 + p_0C_0$

$C_2 = g_1 + p_1C_1 = g_1 + p_1(g_0 + p_0C_0) = g_1 + p_1g_0 + p_1p_0C_0$

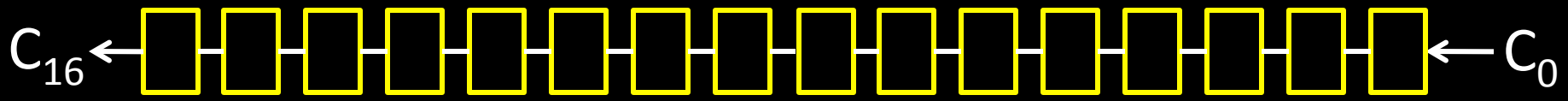
$C_3 = g_2 + p_2C_2 = g_2 + p_2g_1 + p_2p_1g_0 + p_2p_1p_0C_0$

$C_4 = g_3 + p_3C_3 = g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1g_0 + p_3p_2p_1p_0C_0$


A3 B3
A2 B2
A1 B1
A0 B0
g p
g p
g p
g p
C4
CLL (carry look-ahead logic)
C0
R3
R2
R1
R0

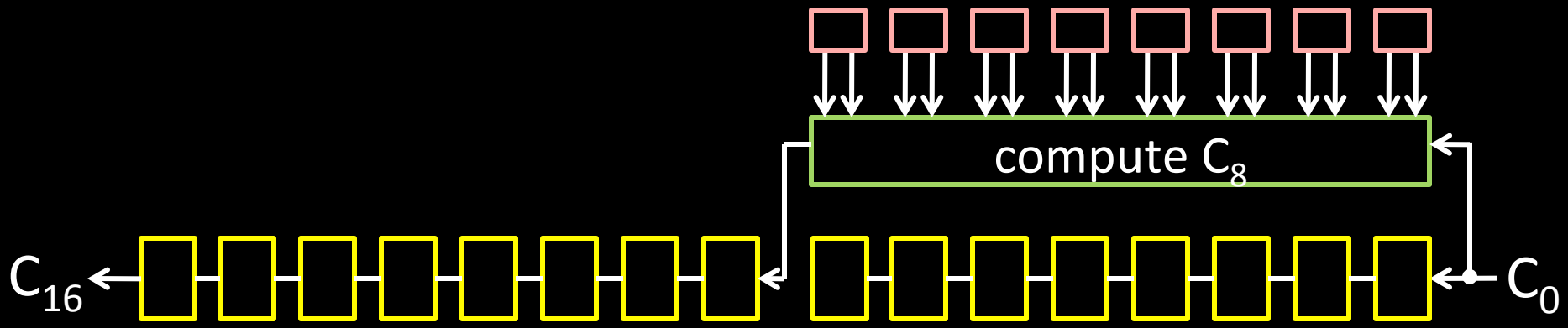
	Space	Time
4 Bit Ripple	4 x 9 = **36 gates** (4 input)	4 x 2 = **8 gate delays**
4 Bit Look-Ahead	4 x 11 + 14 = **58 gates** (5 input)	**5 gate delays**
16 Bit Ripple	**16 x 9 = 144 gates (4 input)**	**16 x 2 = 32 gate delays**
16 Bit Look-Ahead	16 x 11 + 152 = **328 gates** (17 input)	**5 gate delays**
64 Bit Ripple	**64 x 9 = 576 gates (4 input)**	**64 x 2 = 128 gate delays**
64 Bit Look-Ahead	64 x 11 + 2144 = **2848 gates** (65 input)	**5 gate delays**

Only compute some fast carry signals



Carry-Skip Adder

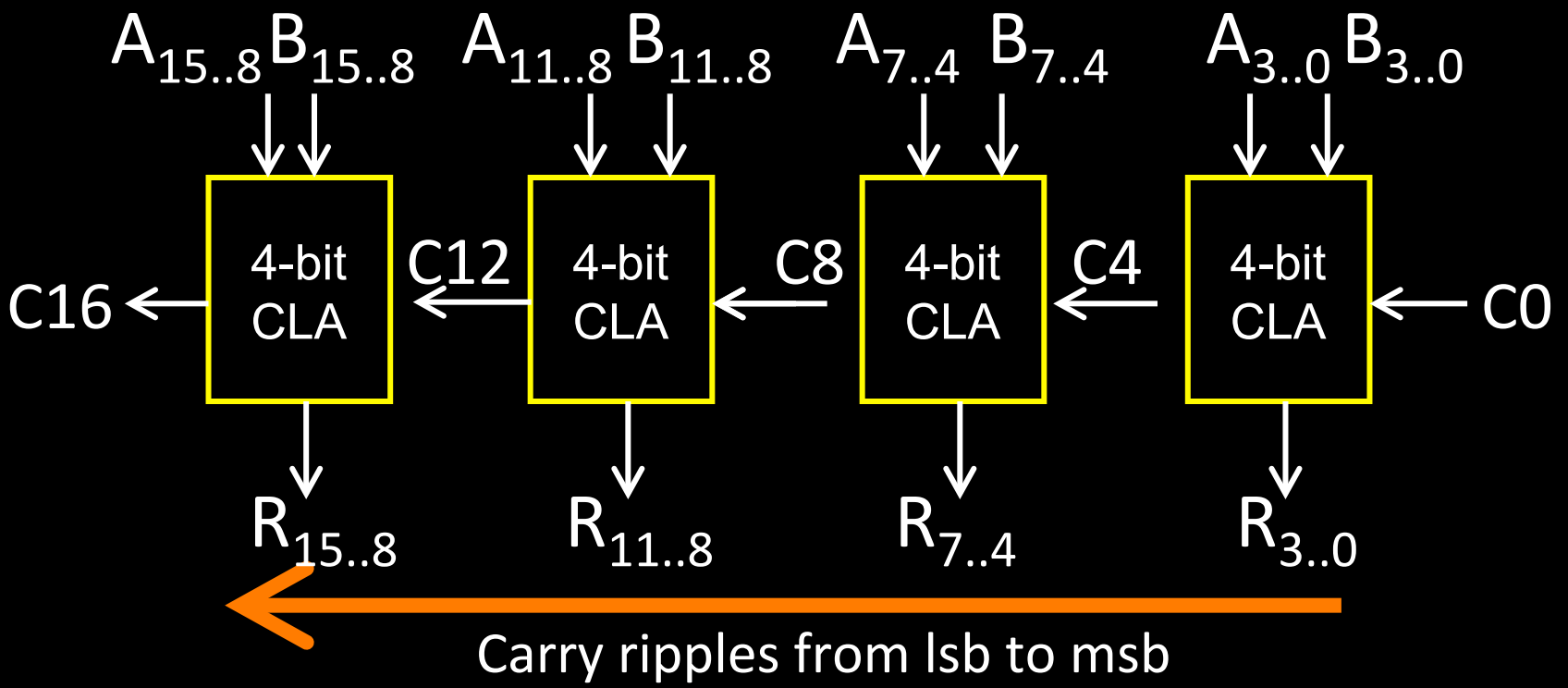
Only compute some fast carry signals



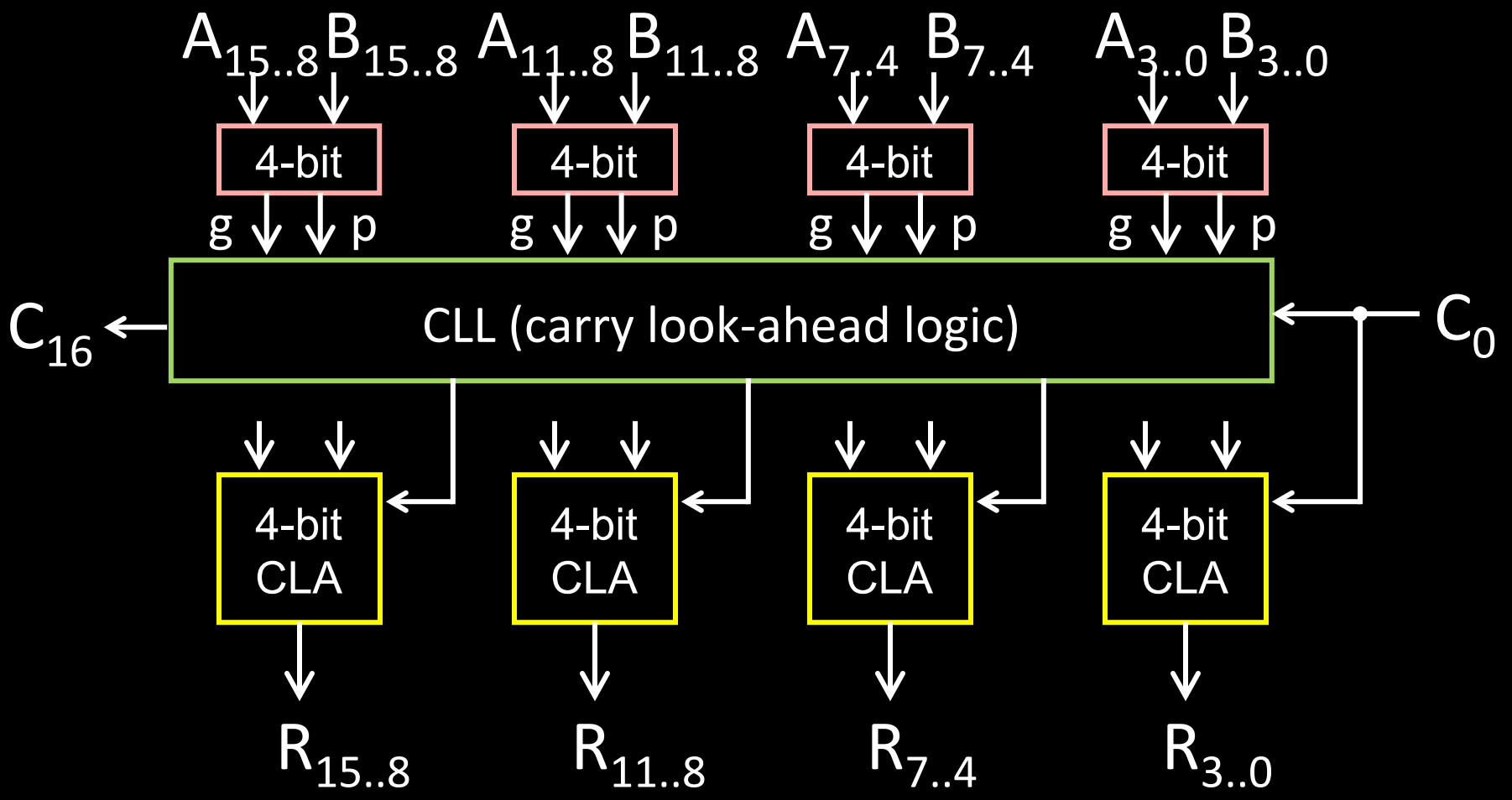
Time: ~ 2x faster than ripple

Space: O(N) extra gates, O(N) gate inputs

Hybrid Approach



Hierarchical Approach



r4 = (r1 + r2) | r3

r8 = 4*r3 + r4 – 1

r9 = 9

ADDU rd, rs, rt

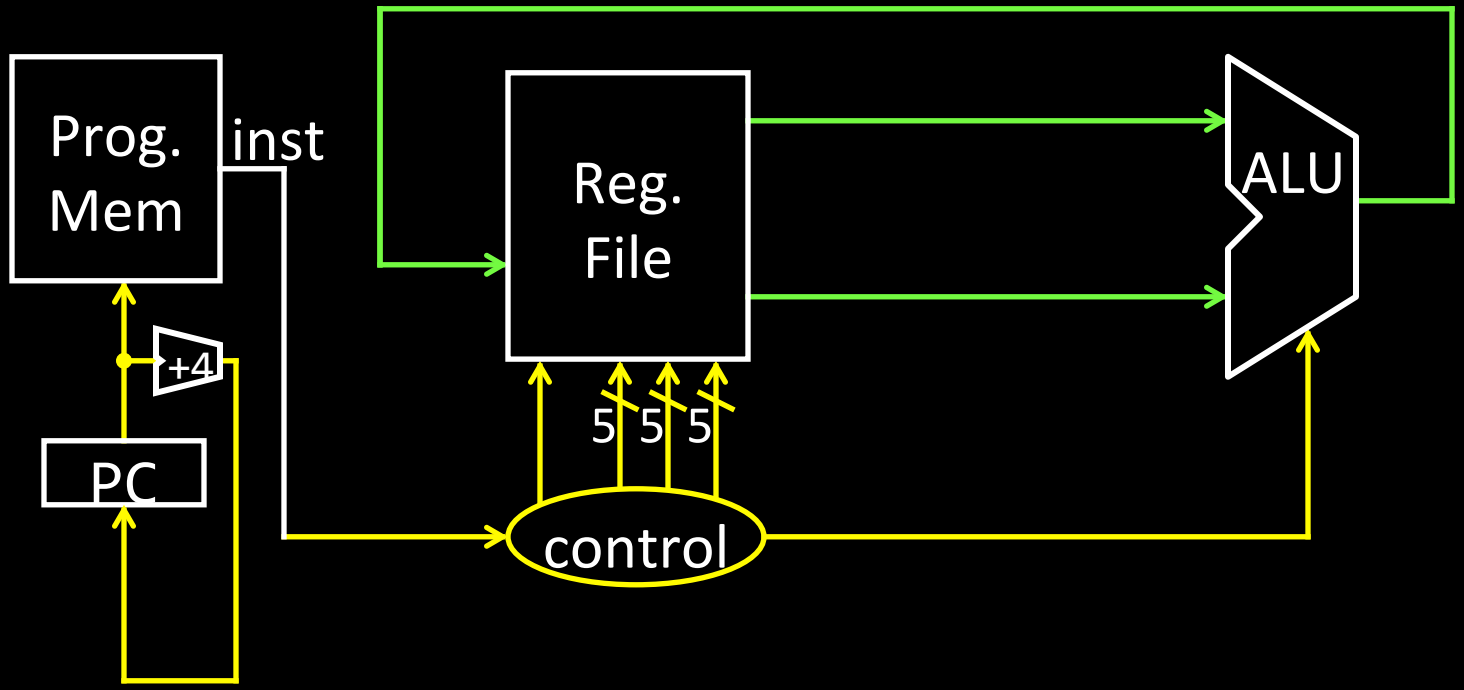
SUBU rd, rs, rt

OR rd, rs, rt

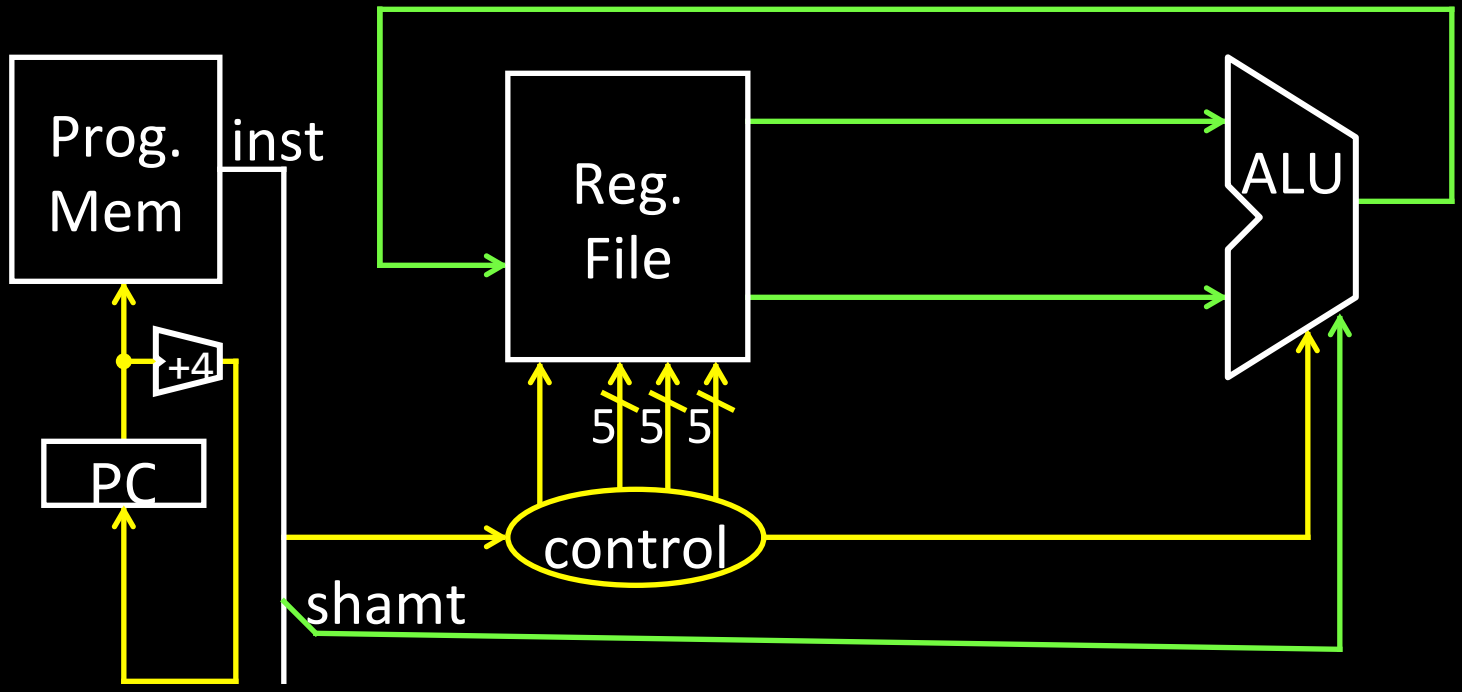
XOR rd, rs, rt

NOR rd, rs rt

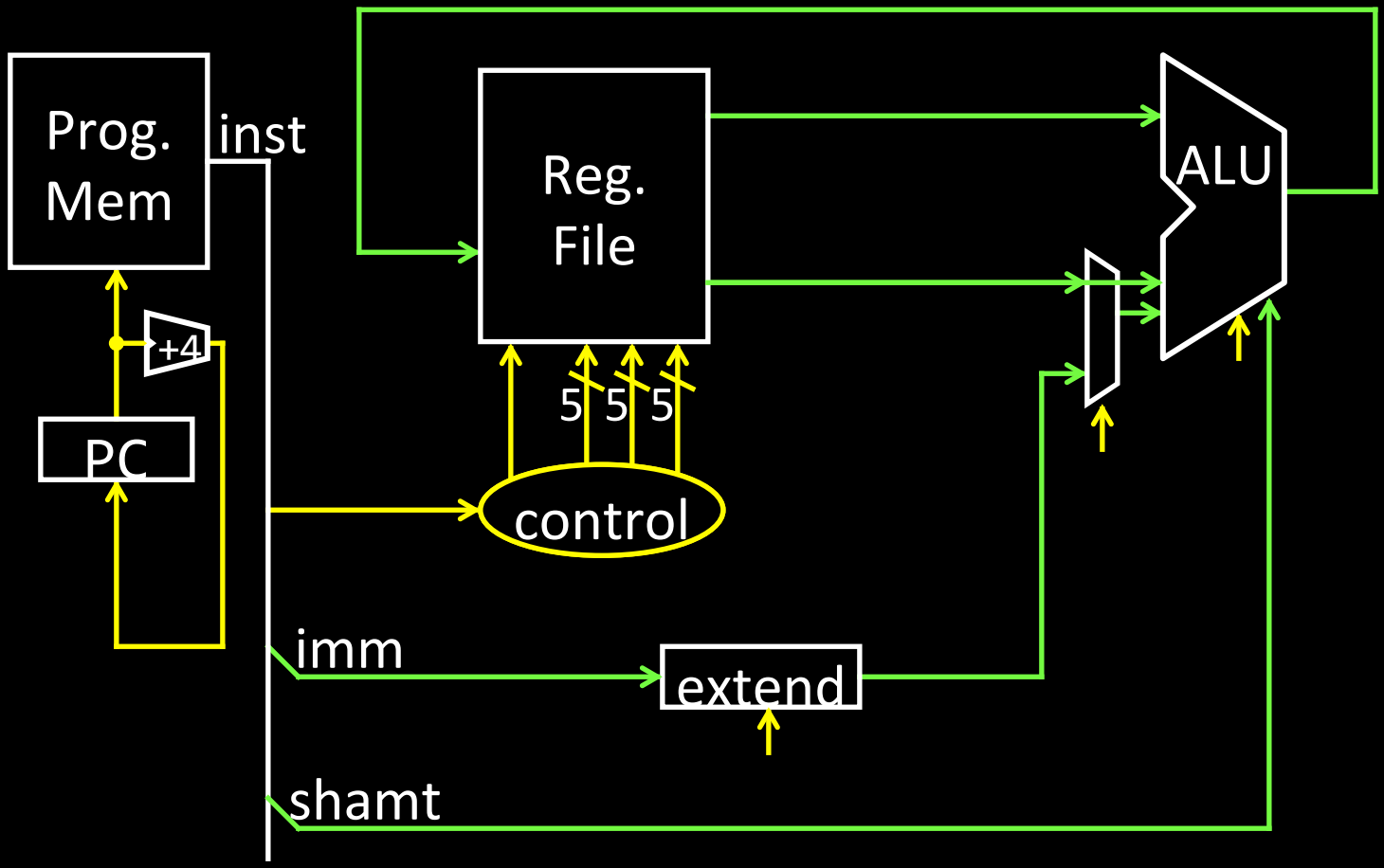
R-type instruction



R-type instruction



I-type instruction



I-type Instruction