Fast Adders

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

Goals:

serial to parallel conversion

time vs. space tradeoffs

design choices

A3 B3   A2 B2   A1 B1   A0 B0

C4 ←   ← C3   ← C2   ← C1   ← C0

R3   R2   R1   R0

Carry ripples from lsb to msb
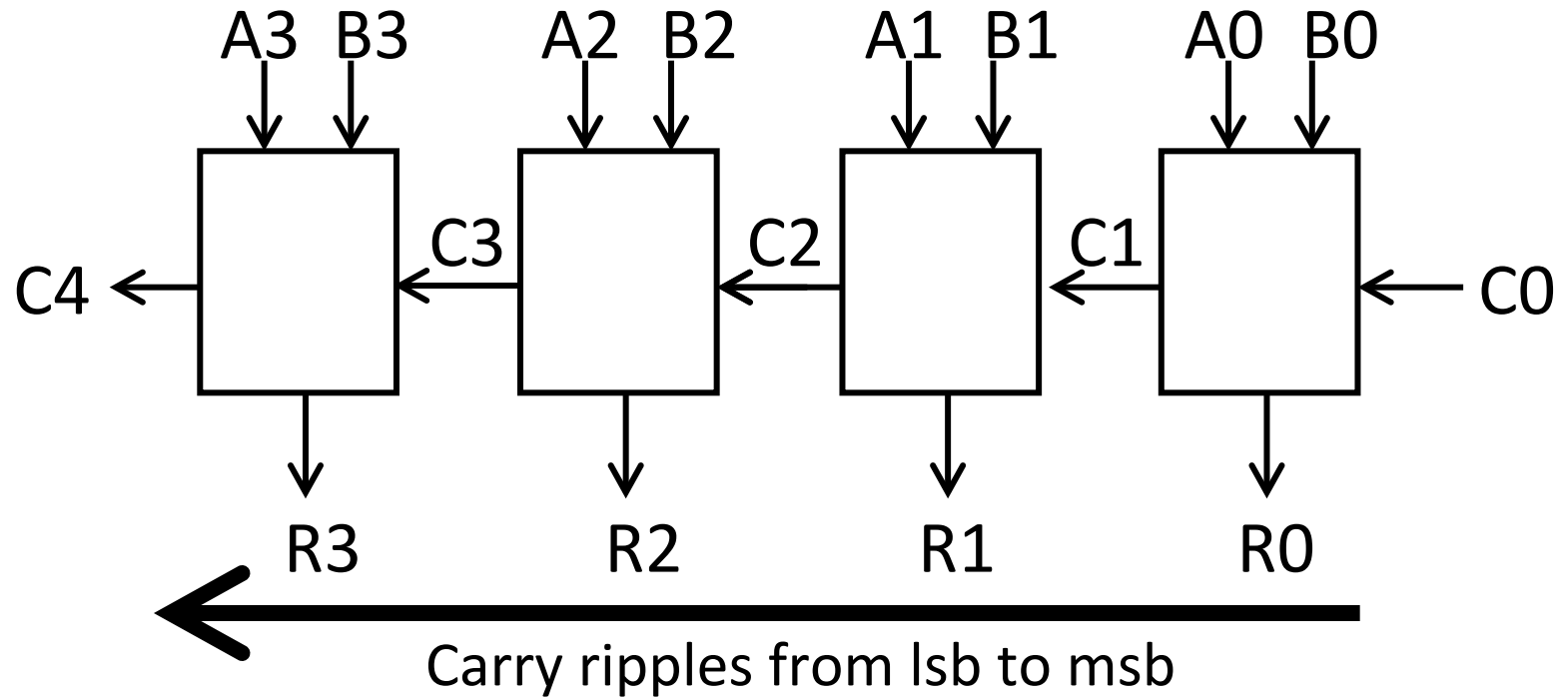
- First full adder, 2 gate delay
- Second full adder, 2 gate delay
- …

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?

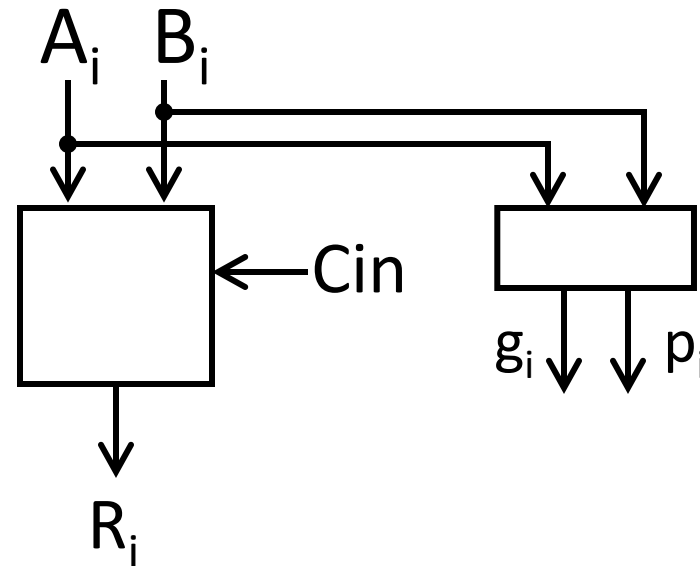A: When A == 1 and B == 1
   (this bit *generates* a carry, irrespective of Cin)

Q: When else might Cout == 1?

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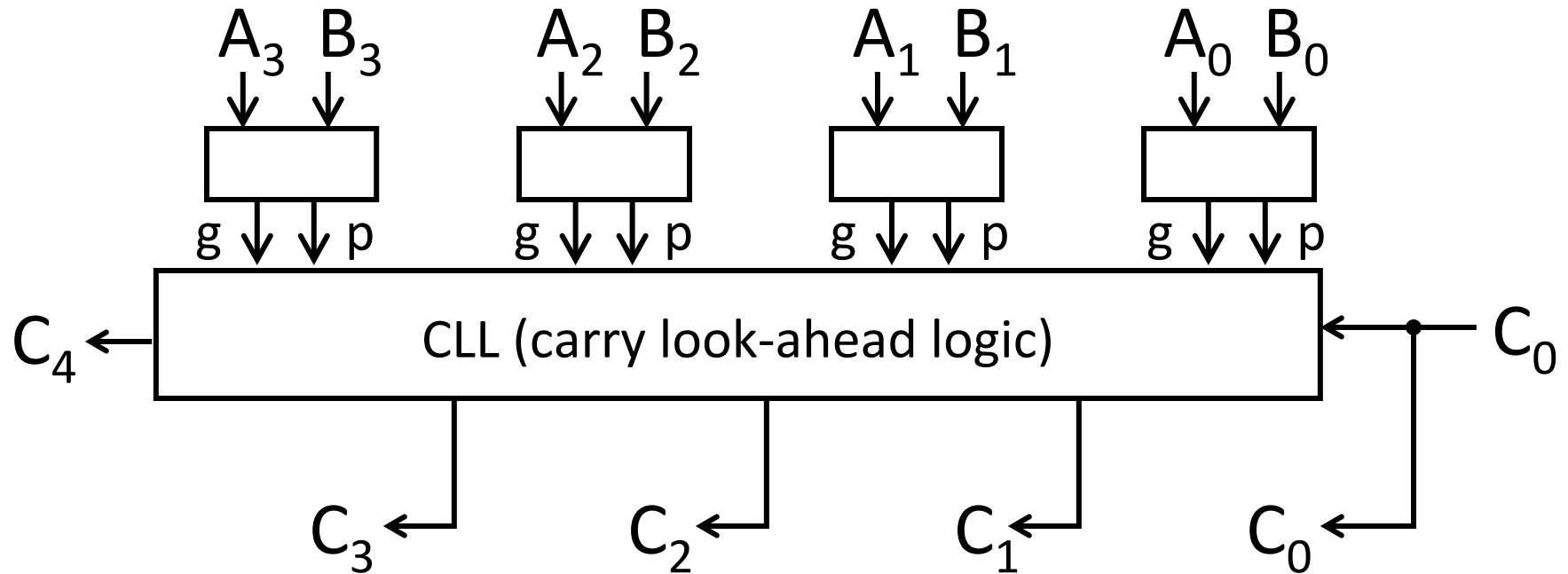   (this bit *propagates* carry from Cin to Cout)

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
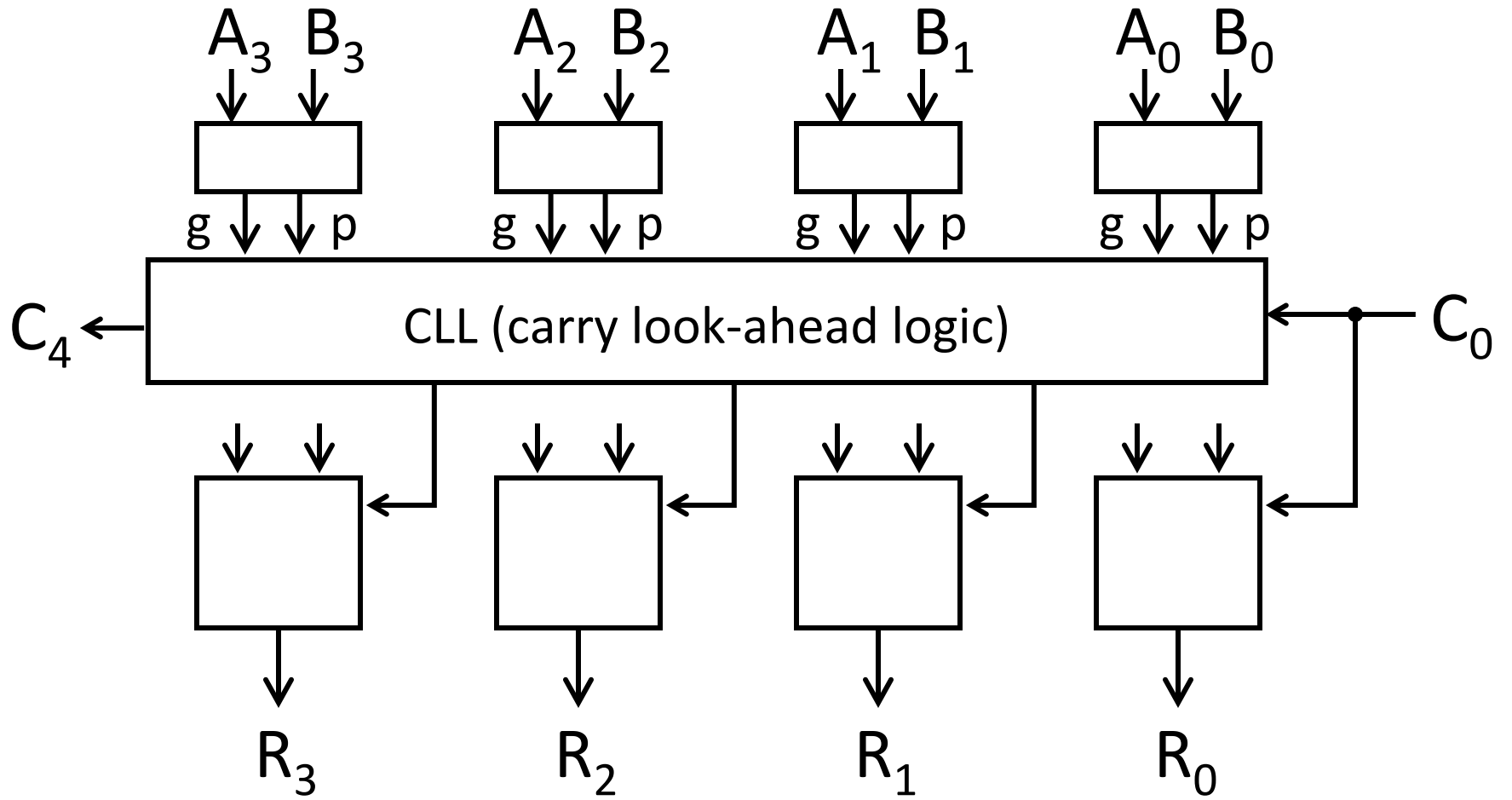- R is 2 gate delay after Cin arrives

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

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

$C_1 = g_0 + p_0 C_0$

$C_2 = g_1 + p_1 C_1 = g_1 + p_1(g_0 + p_0 C_0) = g_1 + p_1 g_0 + p_1 p_0 C_0$

$C_3 = g_2 + p_2 C_2 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 C_0$

$C_4 = g_3 + p_3 C_3 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 C_0$

A_3 B_3 A_2 B_2 A_1 B_1 A_0 B_0

g ↓ ↓ p   g ↓ ↓ p   g ↓ ↓ p   g ↓ ↓ p

C_4 ← CLL (carry look-ahead logic) ← C_0

R_3 R_2 R_1 R_0

	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

$C_{16} \leftarrow$ ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ $\leftarrow C_0$

Carry-Skip Adder
Only compute some fast carry signals

compute C_8

C_{16}

C_0
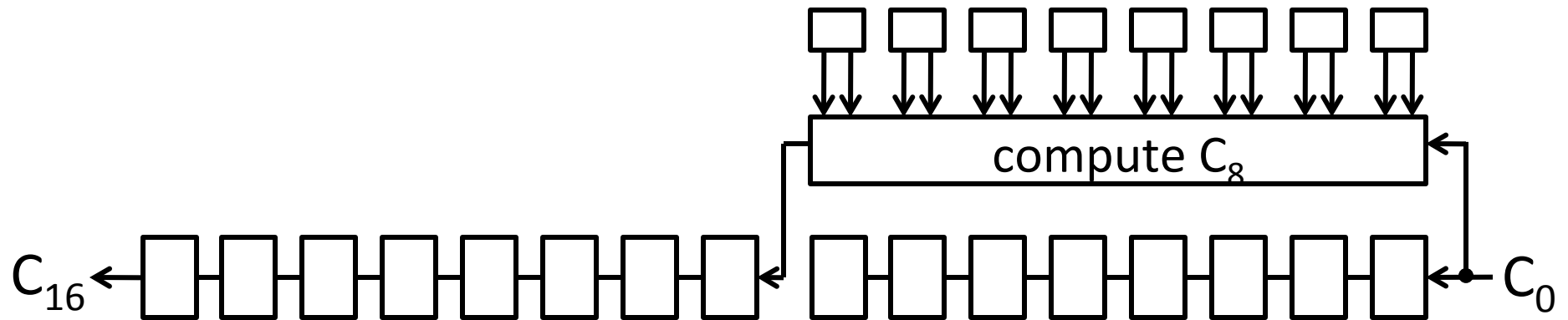
Time: ~ 2x faster than ripple
Space: O(N) extra gates, O(N) gate inputs

Hybrid Approach

$A_{15..8} B_{15..8}$ $A_{11..8} B_{11..8}$ $A_{7..4}$ $B_{7..4}$ $A_{3..0}$ $B_{3..0}$

C16 ← 4-bit CLA ← C12 ← 4-bit CLA ← C8 ← 4-bit CLA ← C4 ← 4-bit CLA ← C0

$R_{15..8}$ $R_{11..8}$ $R_{7..4}$ $R_{3..0}$

Carry ripples from lsb to msb

Hierarchical Approach

$A_{15..8}$ $B_{15..8}$ $A_{11..8}$ $B_{11..8}$ $A_{7..4}$ $B_{7..4}$ $A_{3..0}$ $B_{3..0}$

| 4-bit | 4-bit | 4-bit | 4-bit |

g p   g p   g p   g p

C_{16} ← CLL (carry look-ahead logic) ← C_0

| 4-bit CLA | 4-bit CLA | 4-bit CLA | 4-bit CLA |

$R_{15..8}$ $R_{11..8}$ $R_{7..4}$ $R_{3..0}$
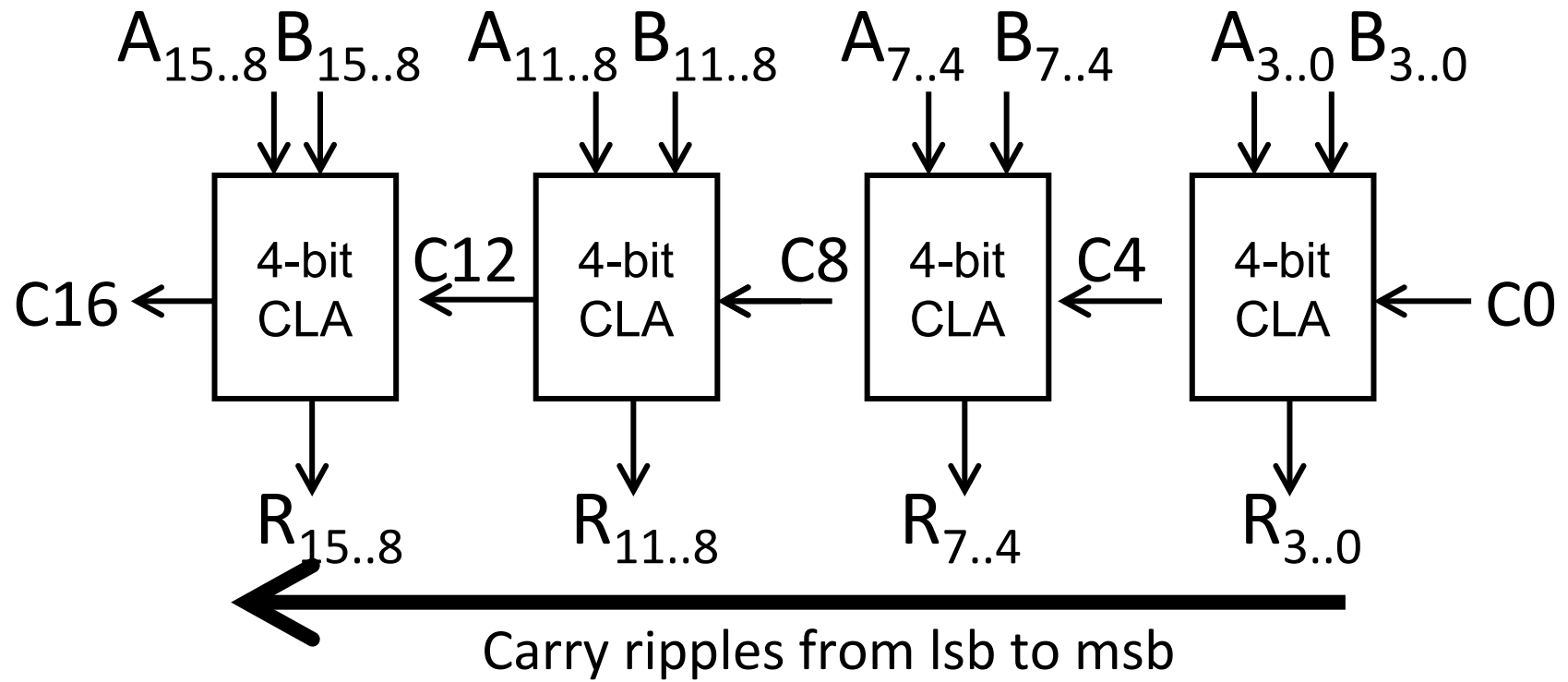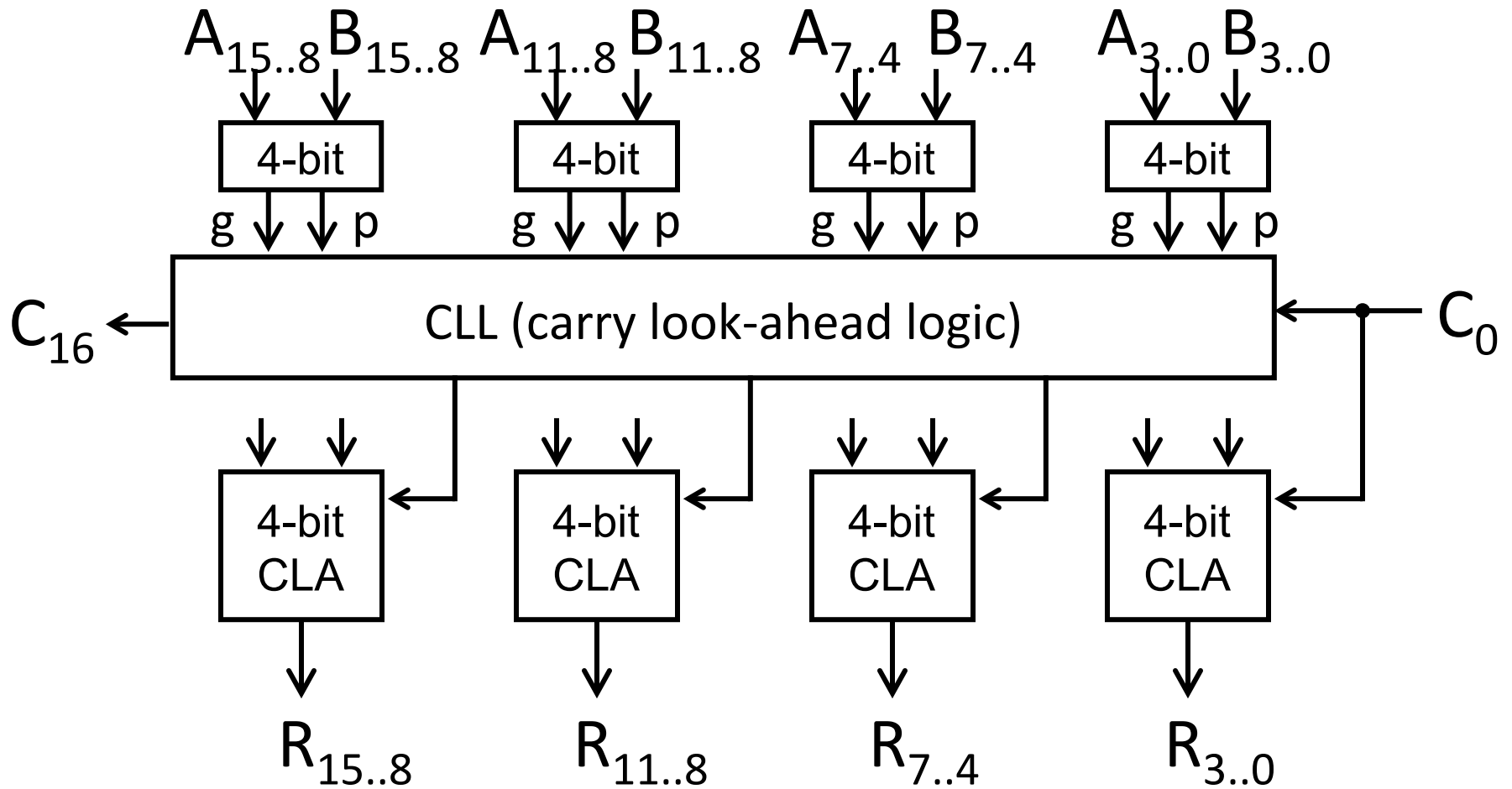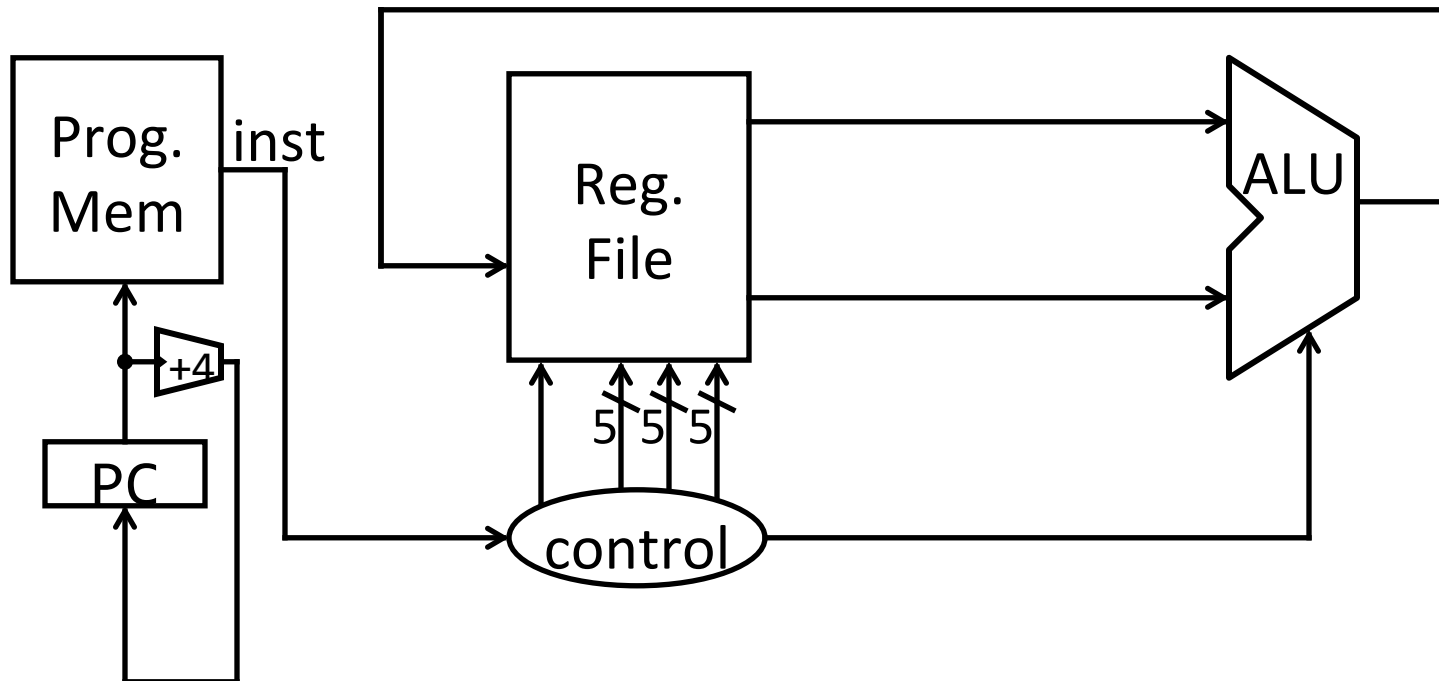
r4 = (r1 + r2) | r3

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
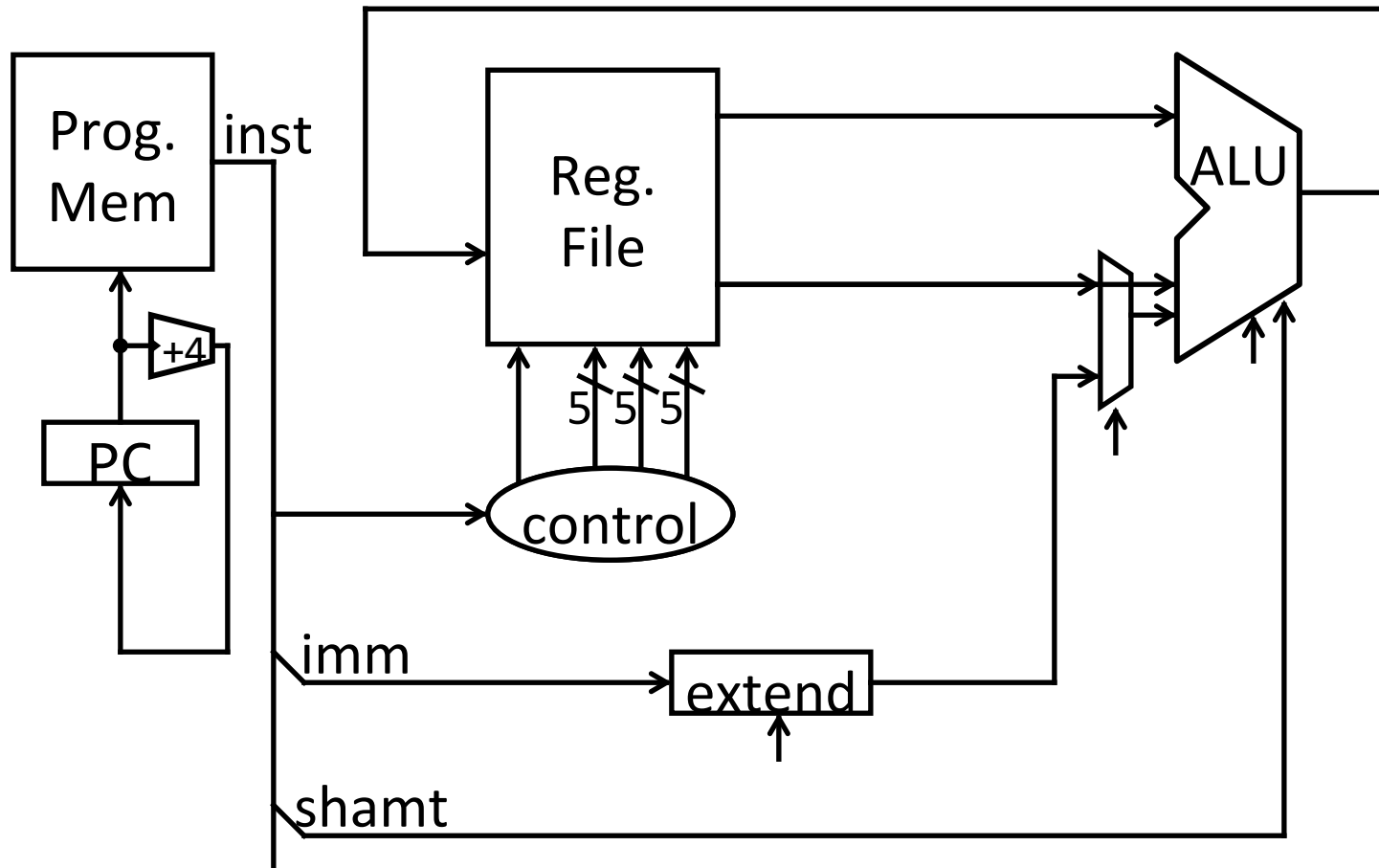
R-type instruction

Prog.
Mem

inst

+4

PC

Reg.
File

ALU

5 5 5

control

shamt
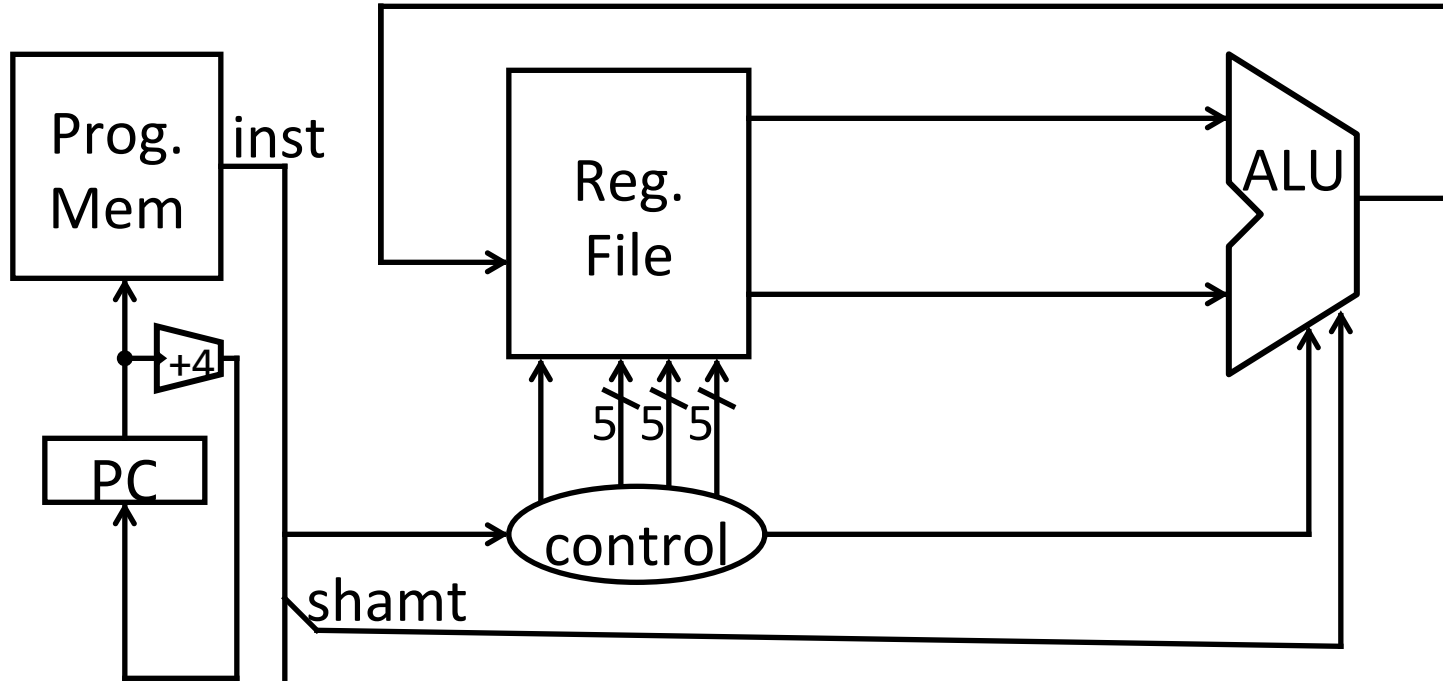
I-type instruction

I-type Instruction