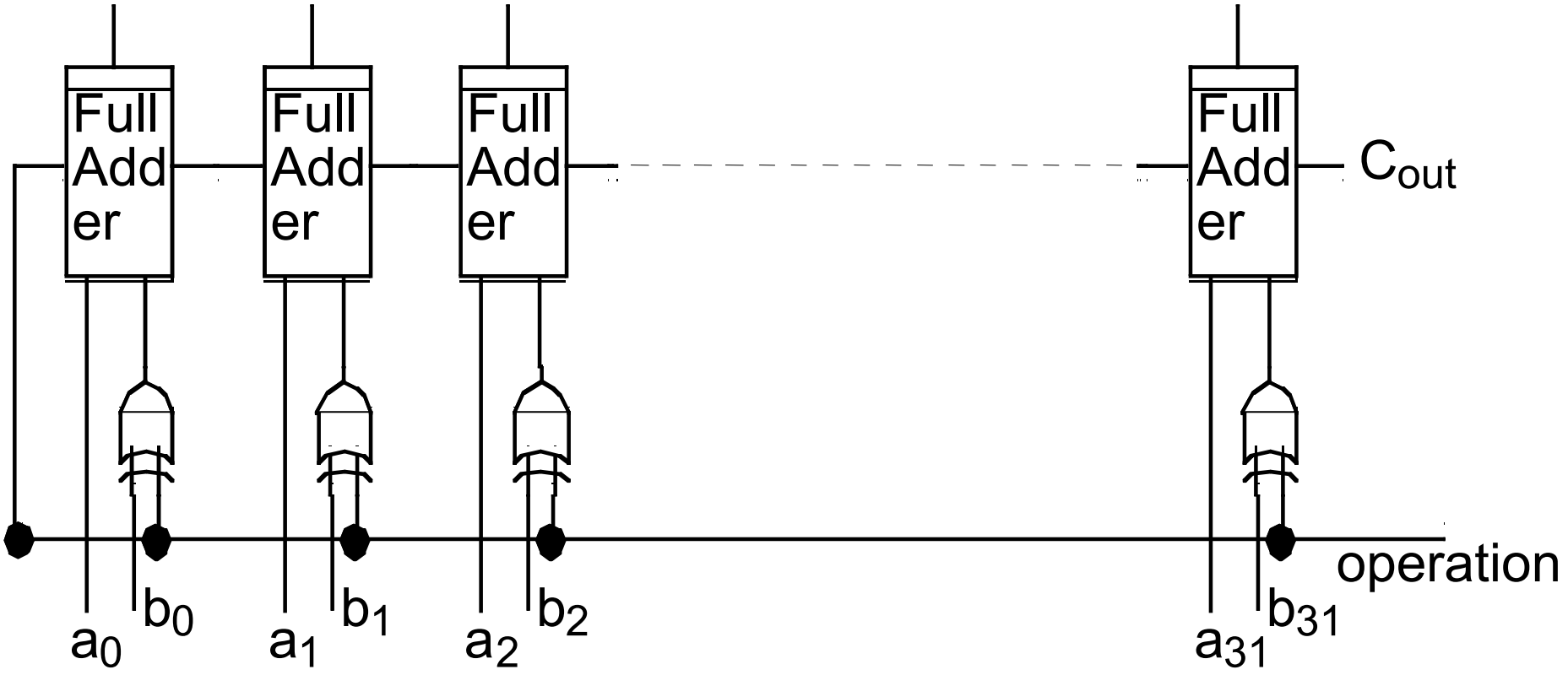
CS/ECE 552: Carry Lookahead

Prof. Matthew D. Sinclair

Lecture notes based in part on slides created by Mark Hill, David Wood, Mikko Lipasti, Guri Sohi, John Shen and Jim Smith

Combined Ripple-carry Adder/Subtractor



- The above ALU is too slow
  - Gate delays for add = 32 x FA + XOR ~= 64

Carry Lookahead

- Theoretically, in parallel
  - $Sum_0 = f(c_{in}, a_0, b_0)$
  - $Sum_i = f(c_{in}, a_i \ldots a_{0,}, b_i \ldots b_0)$
  - $Sum_{31} = f(c_{in}, a_{31} \ldots a_0, b_{31} \ldots b_0)$

- Any boolean function in two levels, right?
  - Wrong! Too much fan-in!

Carry Lookahead

- Need compromise
  - Build tree so delay is $O(\log_2 n)$ for n bits
  - E.g. 2 x 5 gate delays for 32 bits
- We will consider basic concept with
  - 4 bits
  - 16 bits
- Warning: a little convoluted

Carry Lookahead

0101 0100
0011 0110

Need:

both 1 to generate carry

one or both 1s to propagate carry

Define: $g_i = a_i * b_i$ ## carry generate

$p_i = a_i + b_i$ ## carry propagate

Recall: $c_{i+1} = a_i * b_i + a_i * c_i + b_i * c_i$

$= a_i * b_i + (a_i + b_i) * c_i$

$= g_i + p_i * c_i$

Carry Lookahead

- Therefore

$c_1 = g_0 + p_0 * c_0$

$c_2 = g_1 + p_1 * c_1 = g_1 + p_1 * (g_0 + p_0 * c_0)$

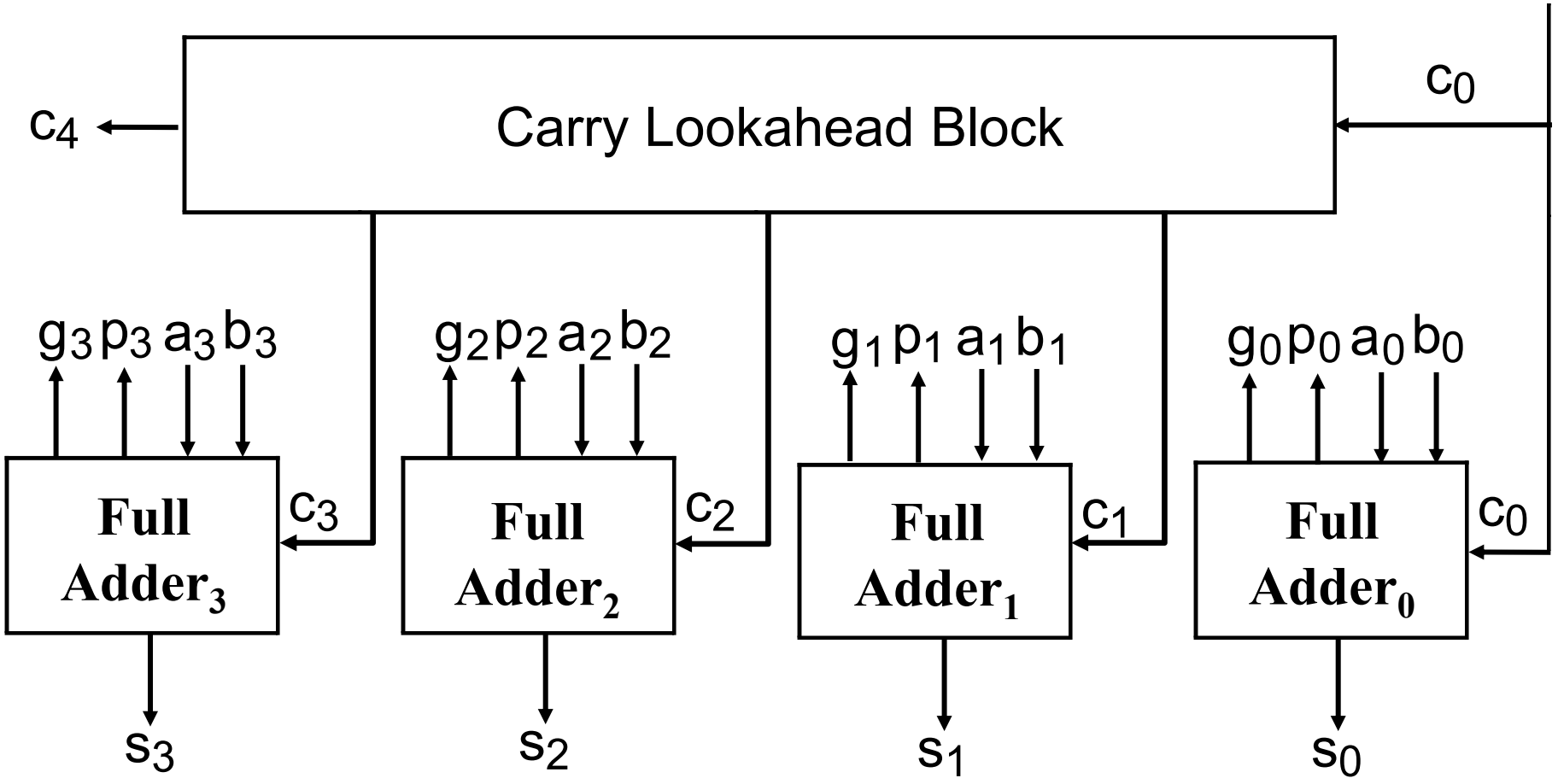
$= g_1 + p_1 * g_0 + p_1 * p_0 * c_0$

$c_3 = g_2 + p_2 * g_1 + p_2 * p_1 * g_0 + p_2 * p_1 * p_0 * c_0$

$c_4 = g_3 + p_3*g_2 + p_3*p_2*g_1 + p_3*p_2*p_1*g_0 + p_3*p_2*p_1*p_0*c_0$

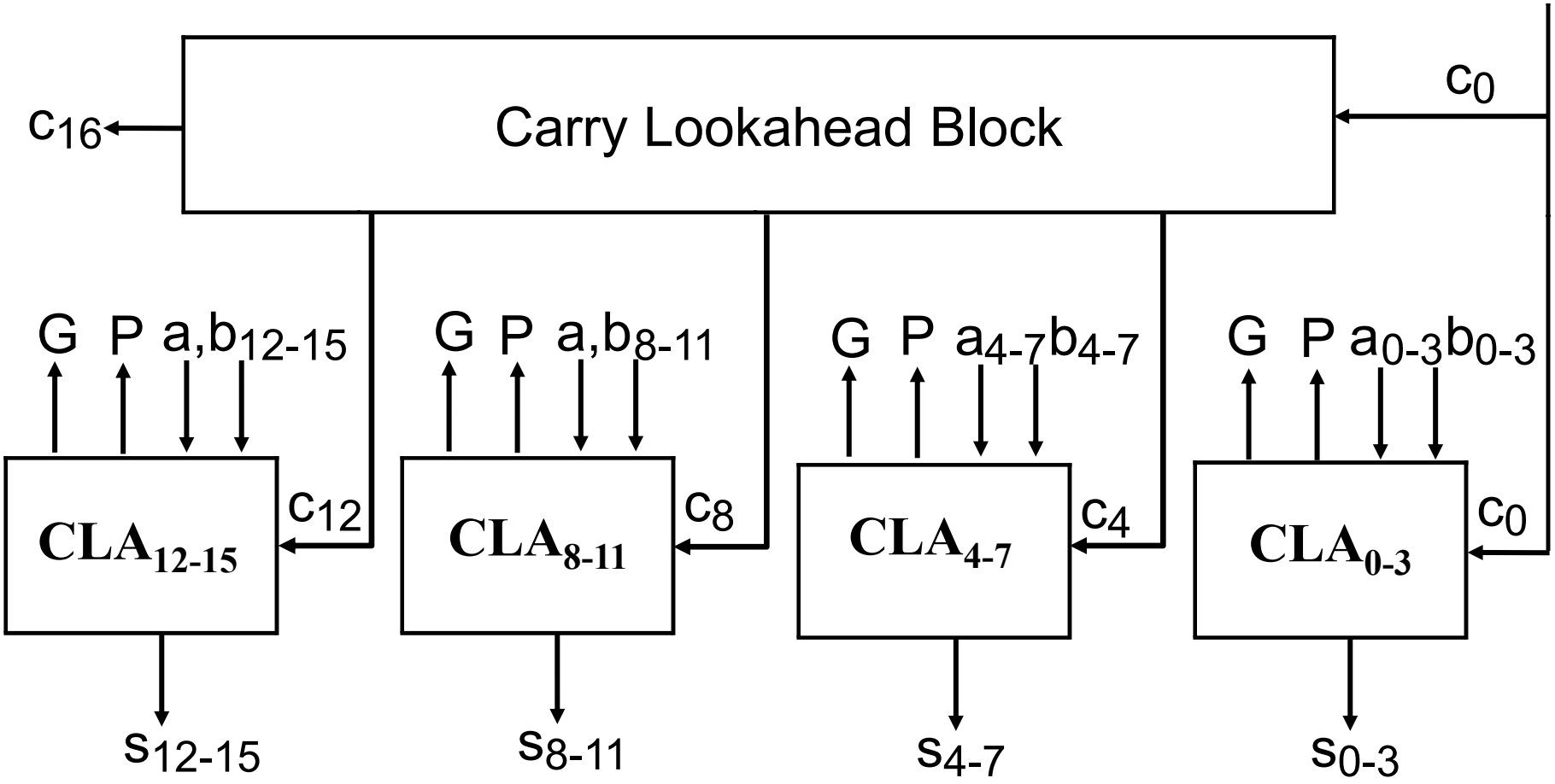
- Uses one level to form p_i and g_i, two levels for carry
- But, this needs n+1 fanin at the OR and the rightmost AND

4-bit Carry Lookahead Adder



Aside: Do we still need c_{out}'s from Full Adders?

Hierarchical Carry Lookahead for 16 bits



Aside: Do we still need c_{out}'s from CLA4's?

Hierarchical CLA for 16 bits

Build 16-bit adder from four 4-bit adders
Figure out G and P for 4 bits together

$G_{0,3} = g_3 + p_3 * g_2 + p_3 * p_2 * g_1 + p_3 * p_2 * p_1 * g_0$

$P_{0,3} = p_3 * p_2 * p_1 * p_0$ (Notation a little different from the book)

$G_{4,7} = g_7 + p_7 * g_6 + p_7 * p_6 * g_5 + p_7 * p_6 * p_5 * g_4$

$P_{4,7} = p_7 * p_6 * p_5 * p_4$

$G_{12,15} = g_{15} + p_{15} * g_{14} + p_{15} * p_{14} * g_{13} + p_{15} * p_{14} * p_{13} * g_{12}$

$P_{12,15} = p_{15} * p_{14} * p_{13} * p_{12}$

Carry Lookahead Basics

Fill in the holes in the G's and P's

$G_{i,k} = G_{j+1,k} + P_{j+1,k} * G_{i,j}$ (assume $i < j+1 < k$)

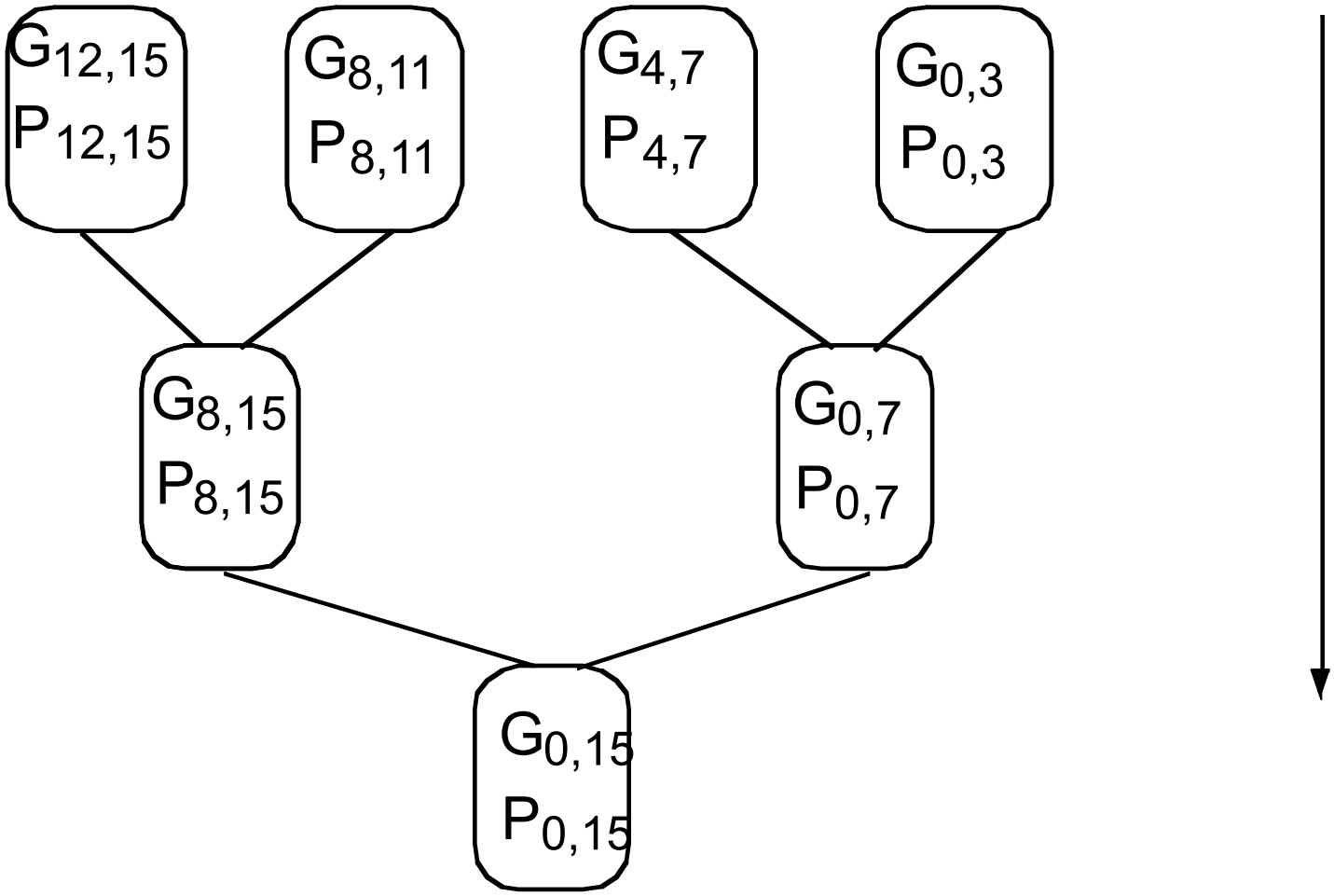
$P_{i,k} = P_{i,j} * P_{j+1,k}$

$G_{0,7} = G_{4,7} + P_{4,7} * G_{0,3}$ $P_{0,7} = P_{0,3} * P_{4,7}$

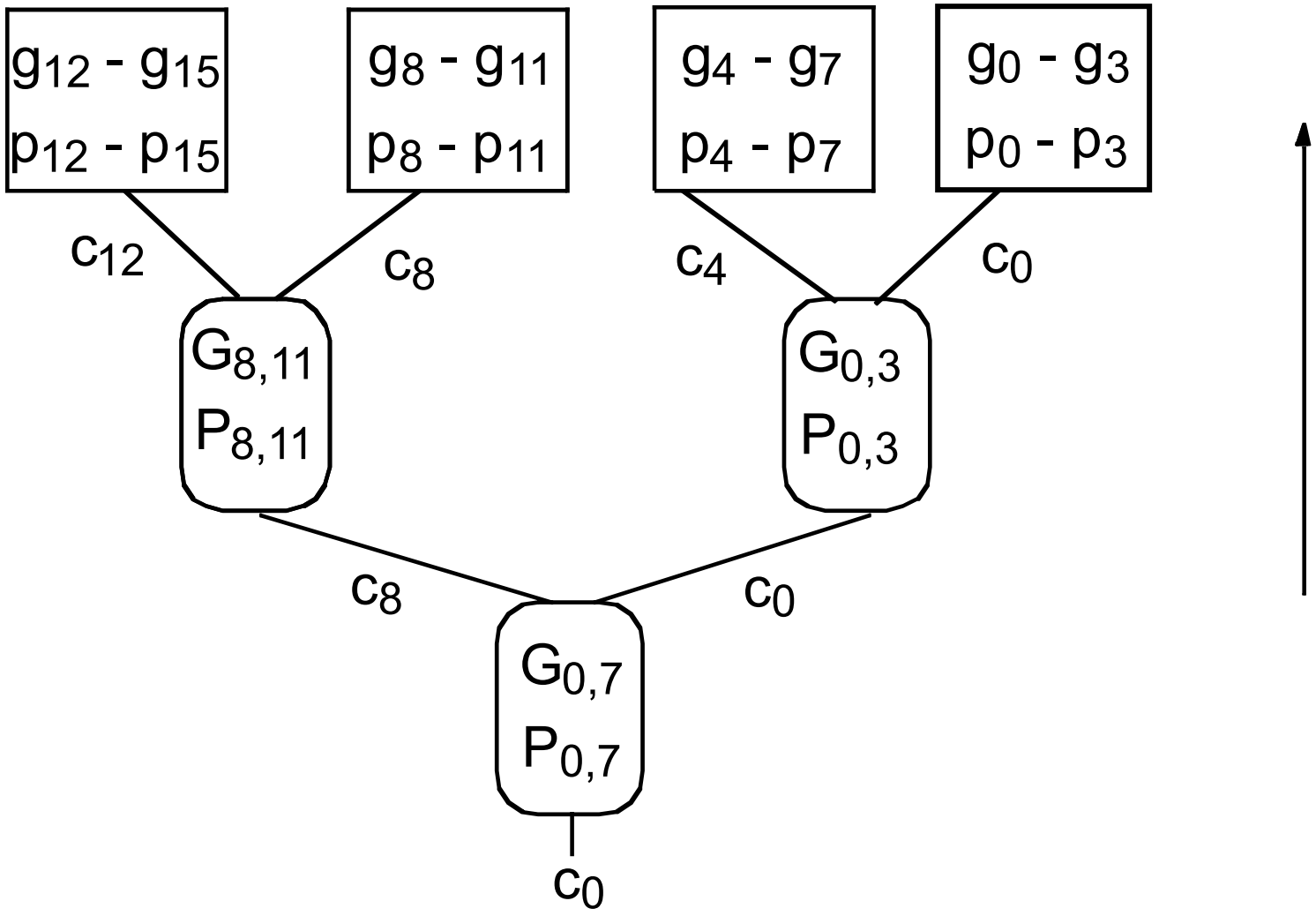
$G_{8,15} = G_{12,15} + P_{12,15} * G_{8,11}$ $P_{8,15} = P_{8,11} * P_{12,15}$

$G_{0,15} = G_{8,15} + P_{8,15} * G_{0,7}$ $P_{0,15} = P_{0,7} * P_{8,15}$

CLA: Compute G's and P's

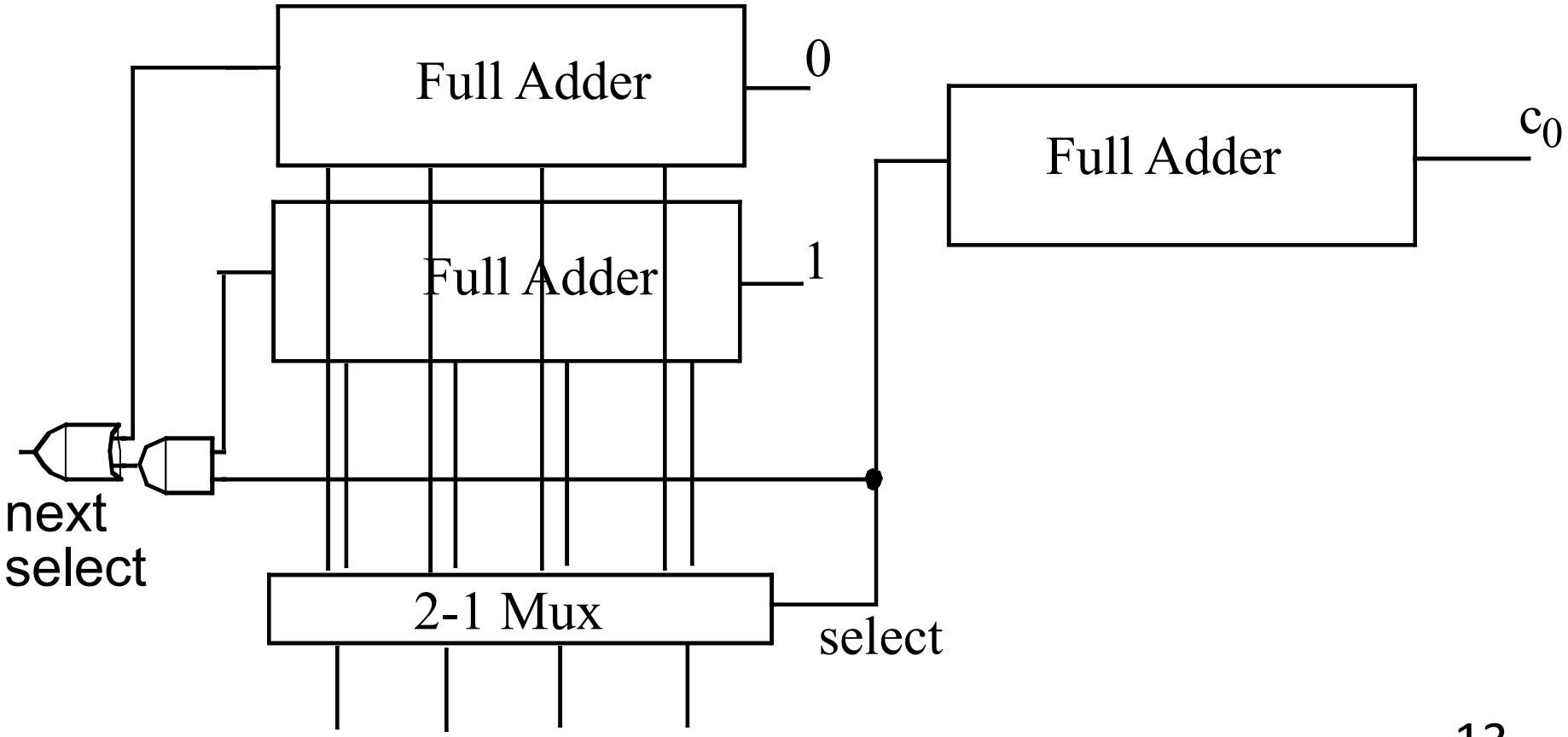


CLA: Compute Carries



Other Adders: Carry Select

- Two adds in parallel; with and without c_{in}
  - When C_{in} is done, select correct result



Other Adders: Carry Save

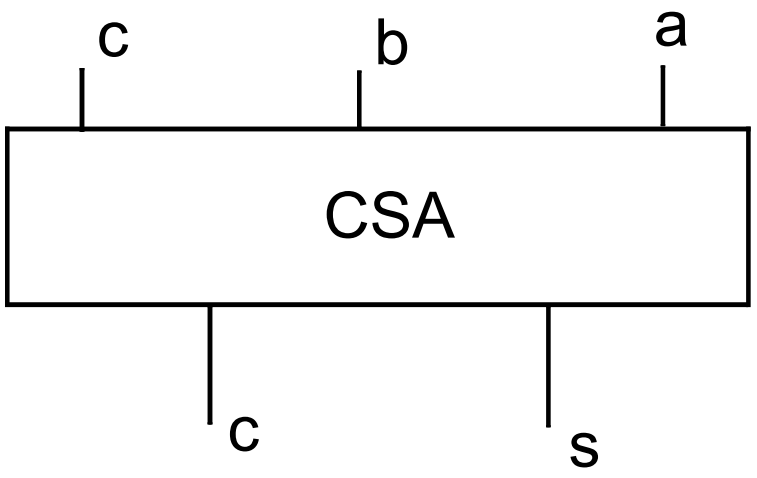
		7	9
	+	1	8
C,S		0,8	1,7
Final		8+1=9	7

A + B => S

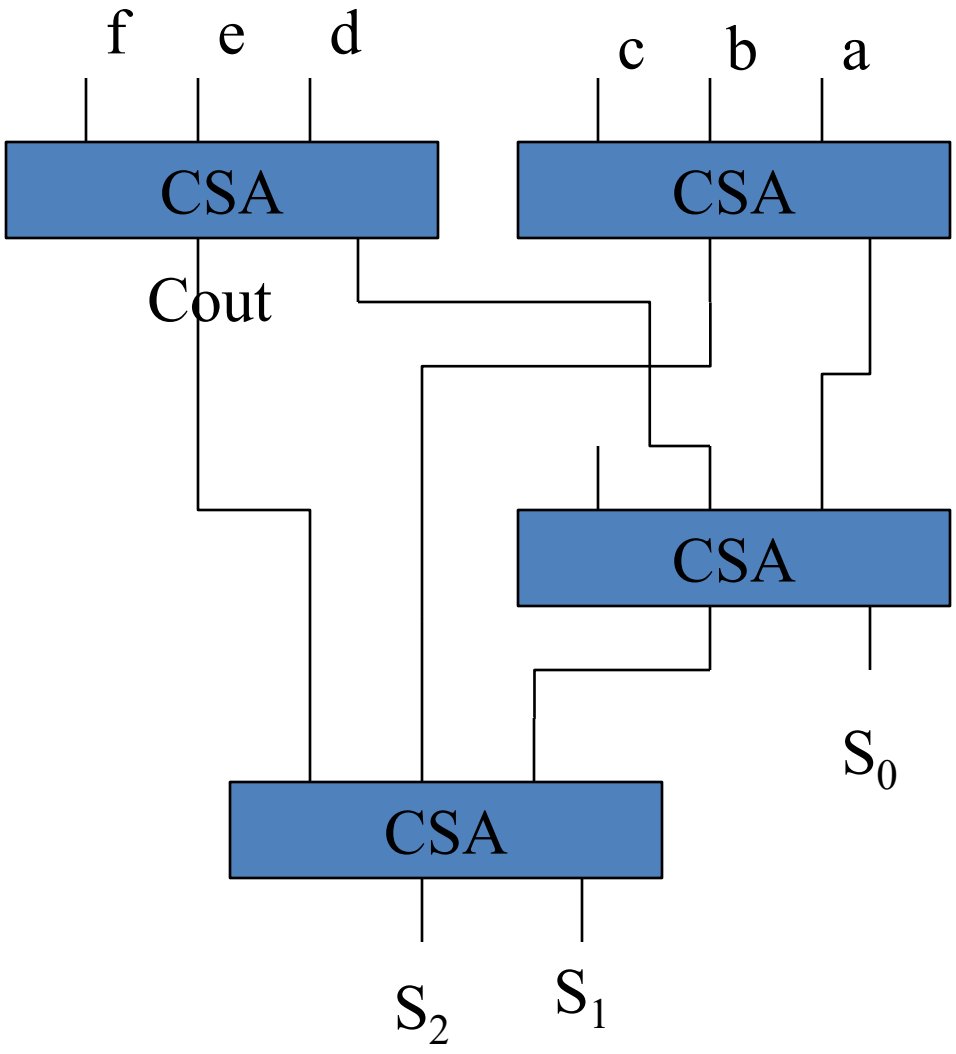
Save carries A + B => S, C_{out}

Use C_{in} A + B + C => S1, S2 (3# to 2# in parallel)

Used in combinational multipliers by building a Wallace Tree



Adding Up Many Bits



Summary

- Carry lookahead
- Carry-select, Carry-save
- State of the art: parallel prefix adders
  - aka. Brent-Kung, Kogge-Stone, …
  - Generalization of CLA
  - Physical design (e.g. wiring) of primary concern
  - Covered in ECE 555