
Computer Science 141

Computing Hardware

Fall 2009

Harvard University

Instructor: Prof. David Brooks

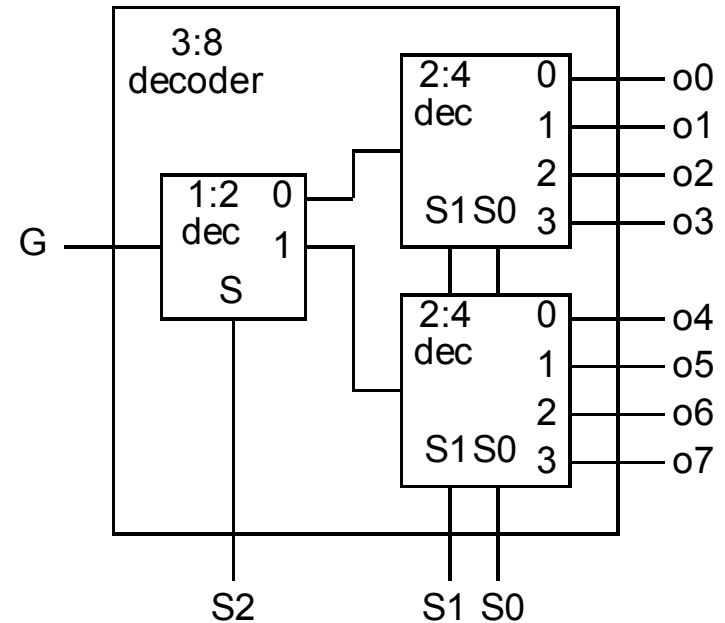
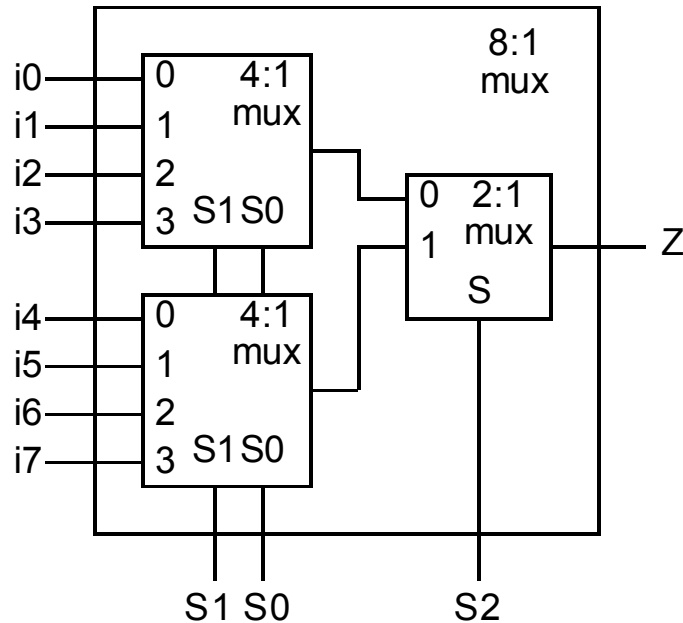
dbrooks@eecs.harvard.edu

Today's Lecture

- Muxes and Fast Decoders
- Tri-state devices and bus design
- Arithmetic Circuits
 - Simple Adders
 - *Fast* Adders
- Next Monday: More complex arithmetic (shifters, multipliers)

multiplexer/decoder

- using small multiplexers to build bigger multiplexers
- using small decoders to build bigger decoders

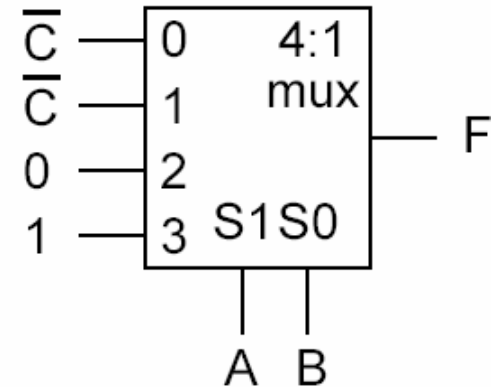
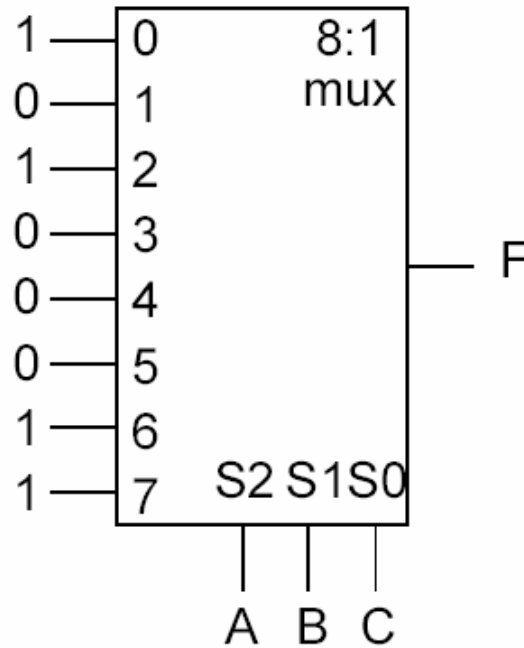


multiplexers

- using multiplexers as logic building blocks

$$F(A,B,C) = m_0 + m_2 + m_6 + m_7$$

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



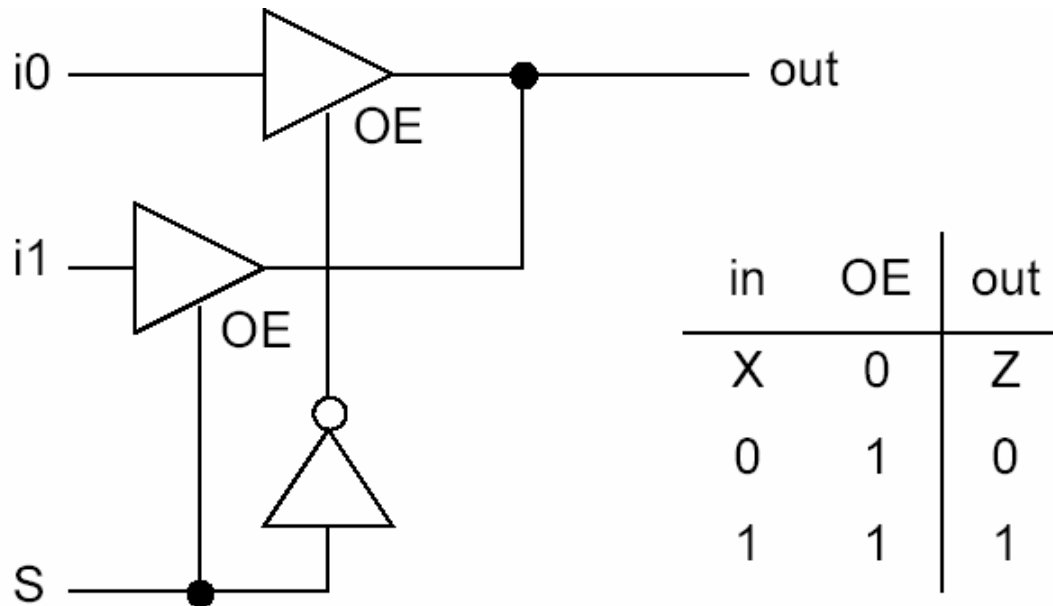
multiplexers

- General strategy to map truth table to multiplexer input mapping

I0	I1	...	In	F			
...			0	0	0	1	1
...			1	0	1	0	1
				↓	↓	↓	↓
				0	In	$\overline{\text{In}}$	1

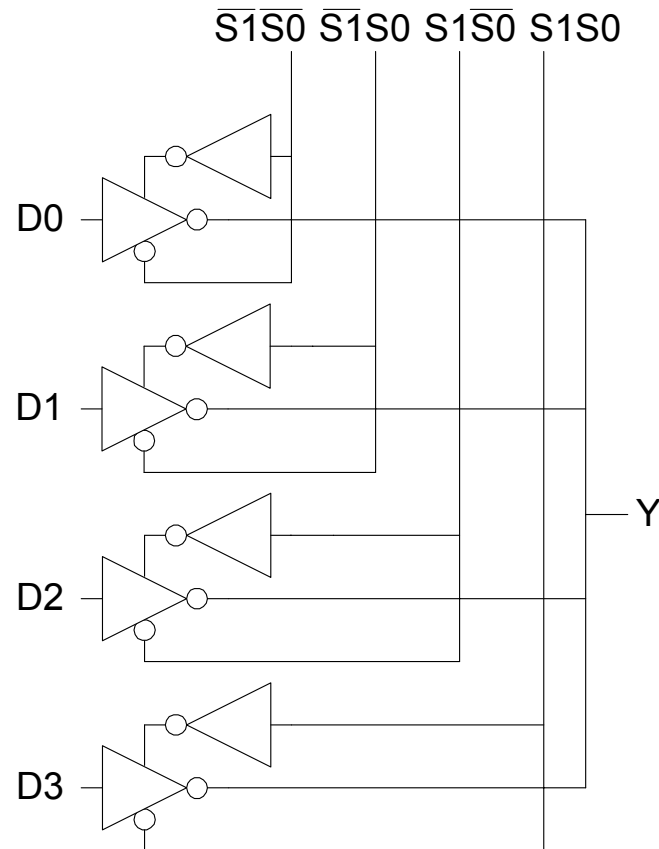
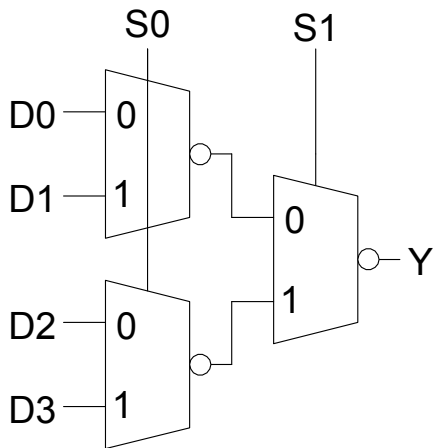
Tri-state gates

- Used to implement multiple outputs connected to a single net
 - without these circuits, possibility of connecting power to ground!
- gate has 3 output values – 0, 1, Z
- Z is high-impedance state: output floats because both pull-up and pull-down devices off



Tri-state example

- 4:1 mux chooses one of 4 inputs using two selects
 - Two levels of 2:1 muxes
 - Or four tristates

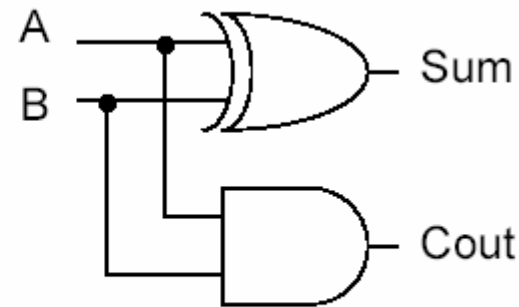


Bus example

Binary adders

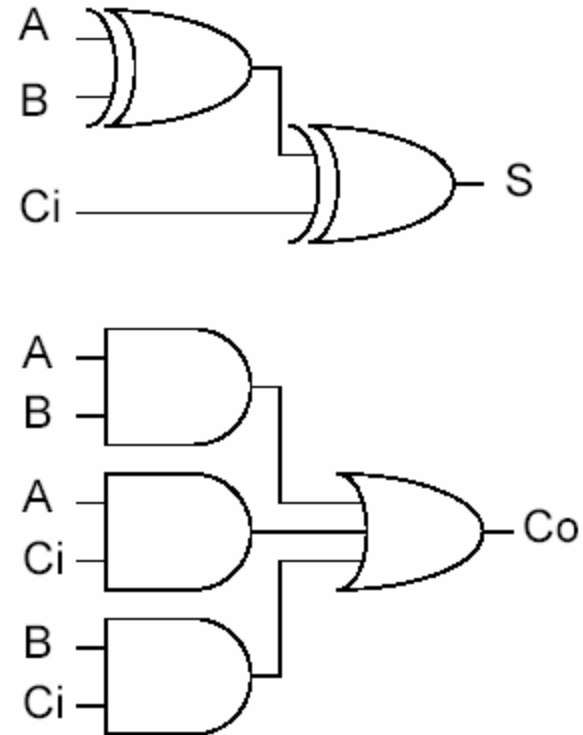
- most modern computers work with 32-bit/64-bit binary numbers
 - need 32-bit/64-bit adder
- simplest way to build 32-bit adder is with 32 1-bit adders
- 1-bit half adder

A	B	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

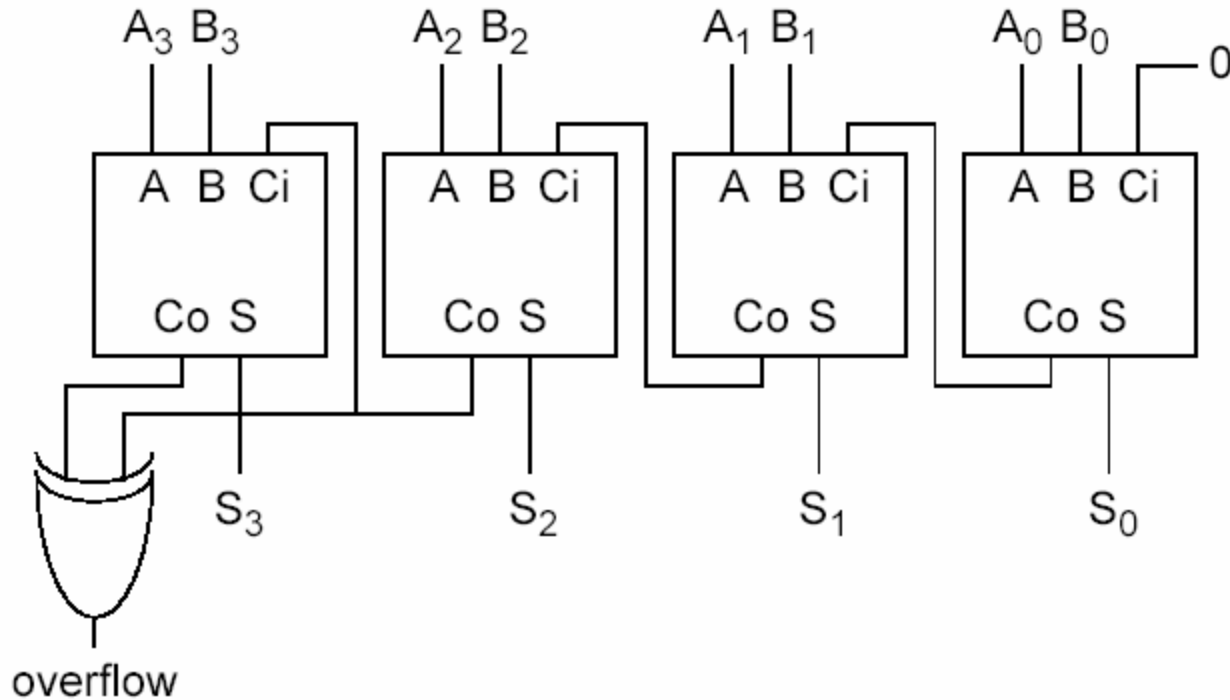


1-bit full adder

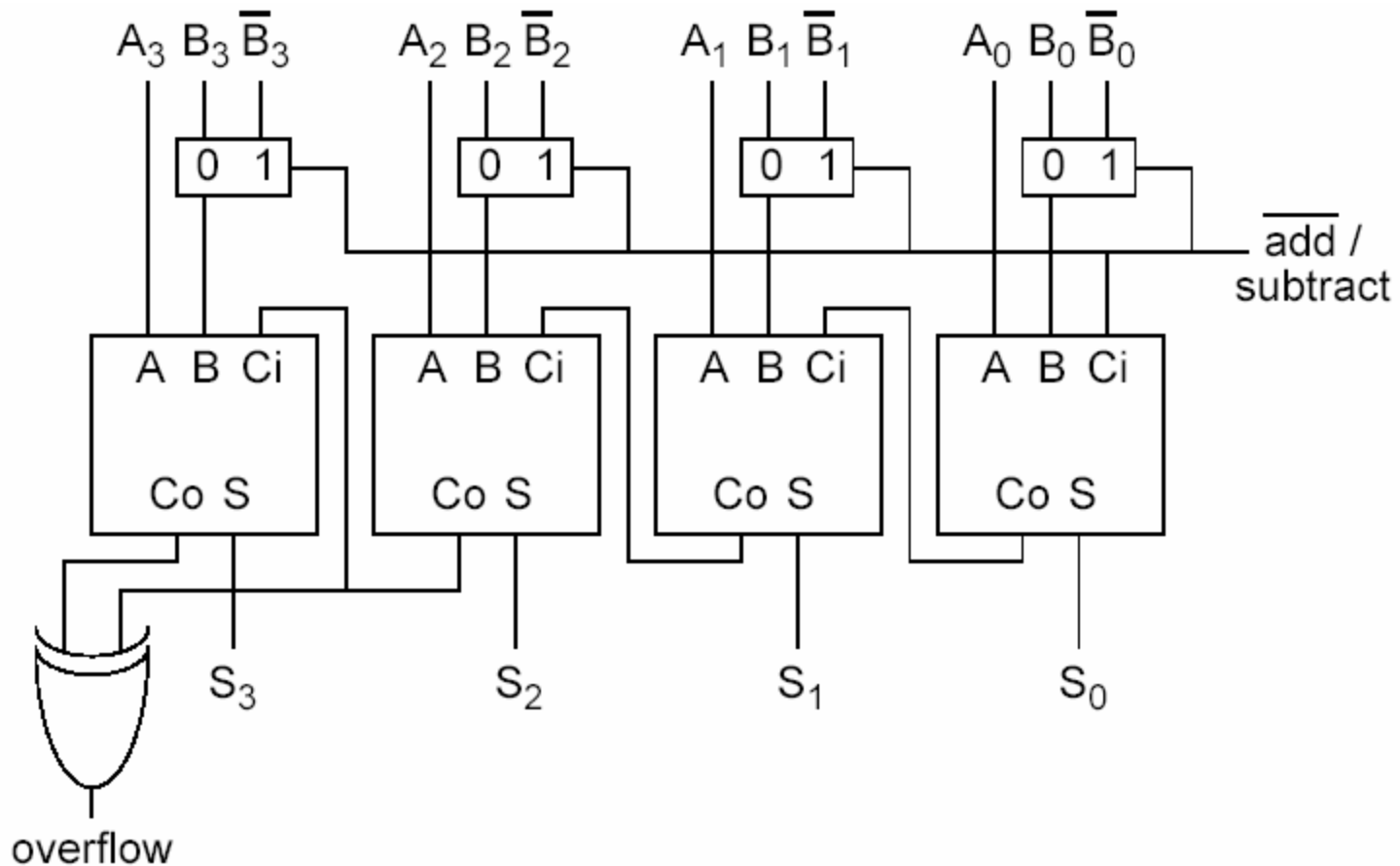
A	B	Ci	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



4-bit ripple-carry adder



4-bit ripple-carry adder/subtractor



$$A - B = A + (-B) = A + (\bar{B} + 1)$$

Delay in ripple-carry adder

- assume all gates have delay of 1, inputs applied at time 0
- for 1-bit full adder
 - Sum after 2 gate delays — $(A \oplus B) \oplus C_i$
 - Cout after 2 gate delays — $A \cdot B + A \cdot C_i + B \cdot C_i$
- for 2-bit ripple-carry adder
 - S0 @ 2 C1 @ 2
 - S1 @ 3 C2 @ 4 (worse case – wait for carry)
- for 4-bit ripple-carry adder
 - S0 @ 2 C1 @ 2
 - S1 @ 3 C2 @ 4
 - S2 @ 5 C3 @ 6
 - S3 @ 7 C4 @ 8
- in worse case, $2N$ gate delays for N -bit ripple-carry adder

Faster adders

- Adder is simply a combinational logic circuit, faster adder requires only 2 gate delays (express directly in 2-level logic form)

⇒ translate $S_N = f(A_N, B_N, C_N)$ into $S_N = f(A_N, \dots, A_0, B_N, \dots, B_0, C_0)$

	Time	Space
ripple carry	$O(n)$	$O(n)$
carry lookahead	$O(\log n)$	$O(n \log n)$
carry skip	$O(n^{1/2})$	$O(n)$
carry select	$O(n^{1/2})$	$O(n)$

Carry-lookahead adder (CLA)

- rewrite sum and carry functions

$$S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i$$

$$C_{i+1} = A_i B_i + A_i C_i + B_i C_i = A_i B_i + C_i (A_i + B_i)$$

$$= A_i B_i + C_i (A_i \bar{B}_i + \bar{A}_i B_i + A_i B_i) = A_i B_i + C_i (A_i \oplus B_i) = G_i + P_i C_i$$

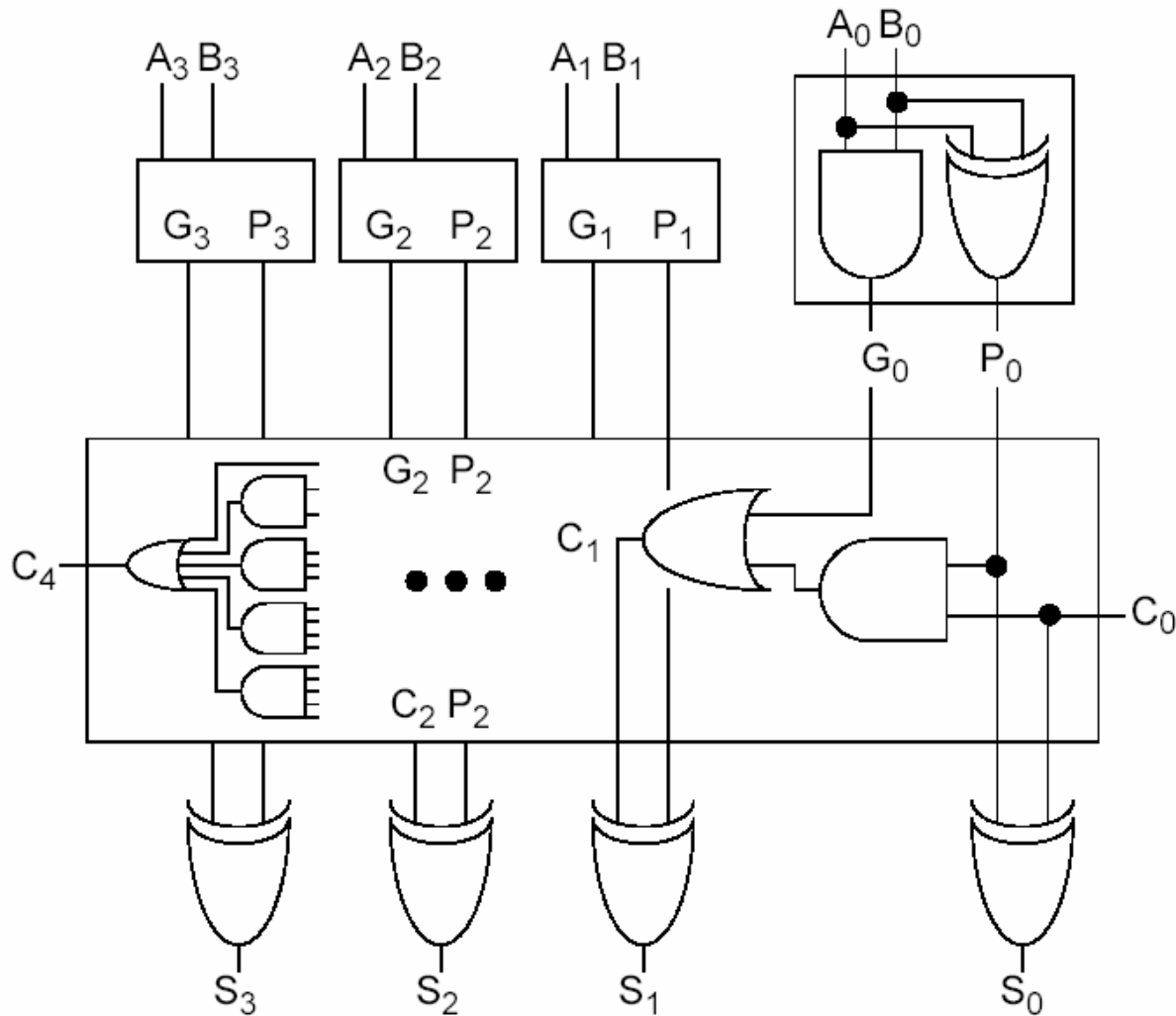
- thus $P_i = A_i \oplus B_i$ $G_i = A_i \cdot B_i$
- (carry **P**ropagate, carry **G**enerate)
- looking at carry calculation now

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_0 \text{ (etc.)}$$

- i th carry has $i+1$ product terms and an $i+1$ literal product
∴ size of lookahead limited to about 4
(Fan-in for C_4 gates?)

Carry-lookahead adder (CLA)



Hierarchical approach to wider adders

- use 4-bit carry-lookahead adders
- each 4-bit adder computes its own group P and G

$$P_G = P_3 \cdot P_2 \cdot P_1 \cdot P_0$$

$$G_G = G_3 + G_2P_3 + G_1P_3P_2 + G_0P_3P_2P_1$$

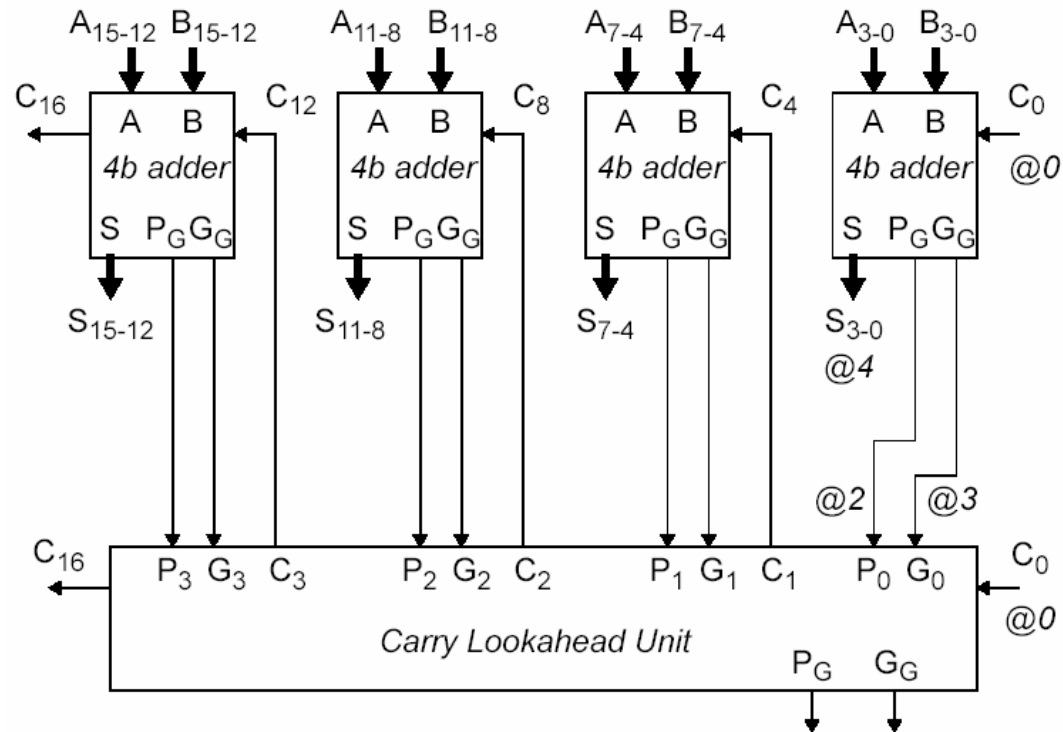


Figure inspired by Figure A.15 in J. Hennessy and D. Patterson, Computer Architecture: A Quantitative Approach,

