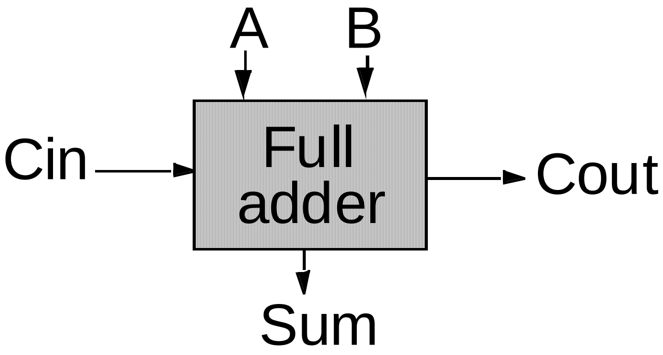
EECS 427
Lecture 8: Intro to adders

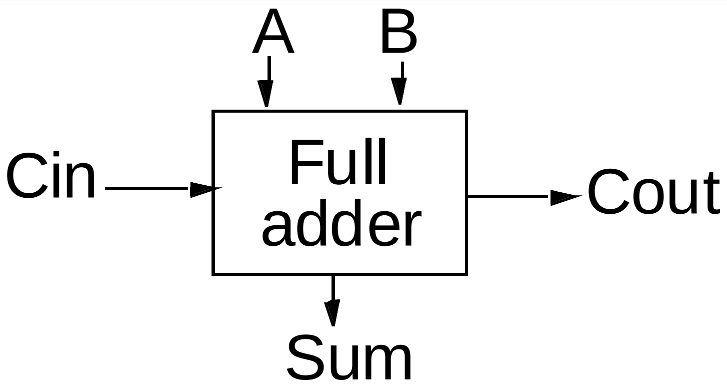
Reading: 11.1 – 11.3.1

Full Adder



A	B	C_i	S	C_o	*Carry status*
0	0	0	0	0	delete
0	0	1	1	0	delete
0	1	0	1	0	propagate
0	1	1	0	1	propagate
1	0	0	1	0	propagate
1	0	1	0	1	propagate
1	1	0	0	1	generate
1	1	1	1	1	generate

The Binary Adder



A

B

Cin

Full adder

Cout

Sum

$$S = A \oplus B \oplus C_i$$

$$= A\overline{B}\overline{C}_i + \overline{A}B\overline{C}_i + \overline{A}\overline{B}C_i + ABC_i$$

$$C_o = AB + BC_i + AC_i$$

Express Sum and Carry as a function of P, G, D

Define 3 new variables which ONLY depend on A, B   WHY?

Generate (G) = AB

Propagate (P) = A ⊕ B

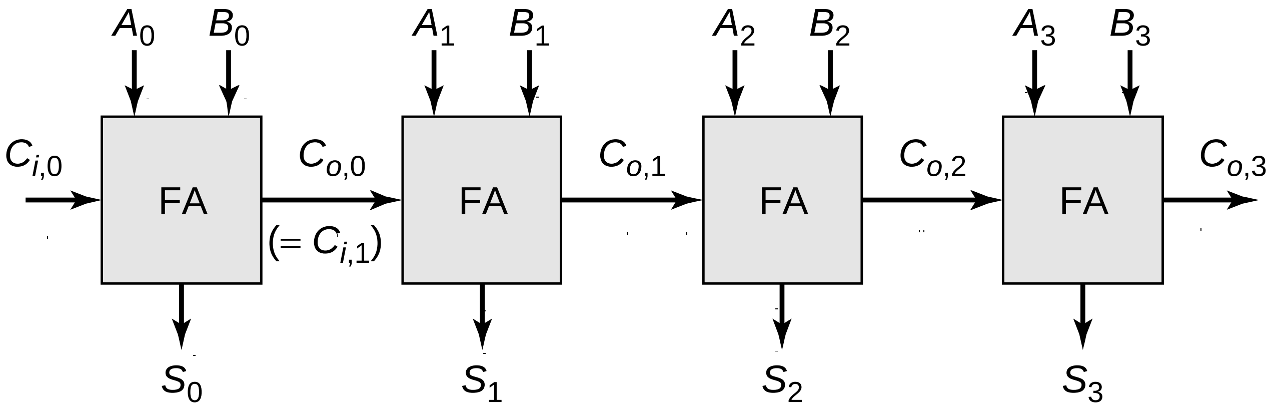
Delete = $\overline{A}\,\overline{B}$

$$C_o(G, P) = G + PC_i$$

$$S(G, P) = P \oplus C_i$$

Can also derive expressions for *S* and C_o based on *D* and *P*

The Ripple-Carry Adder



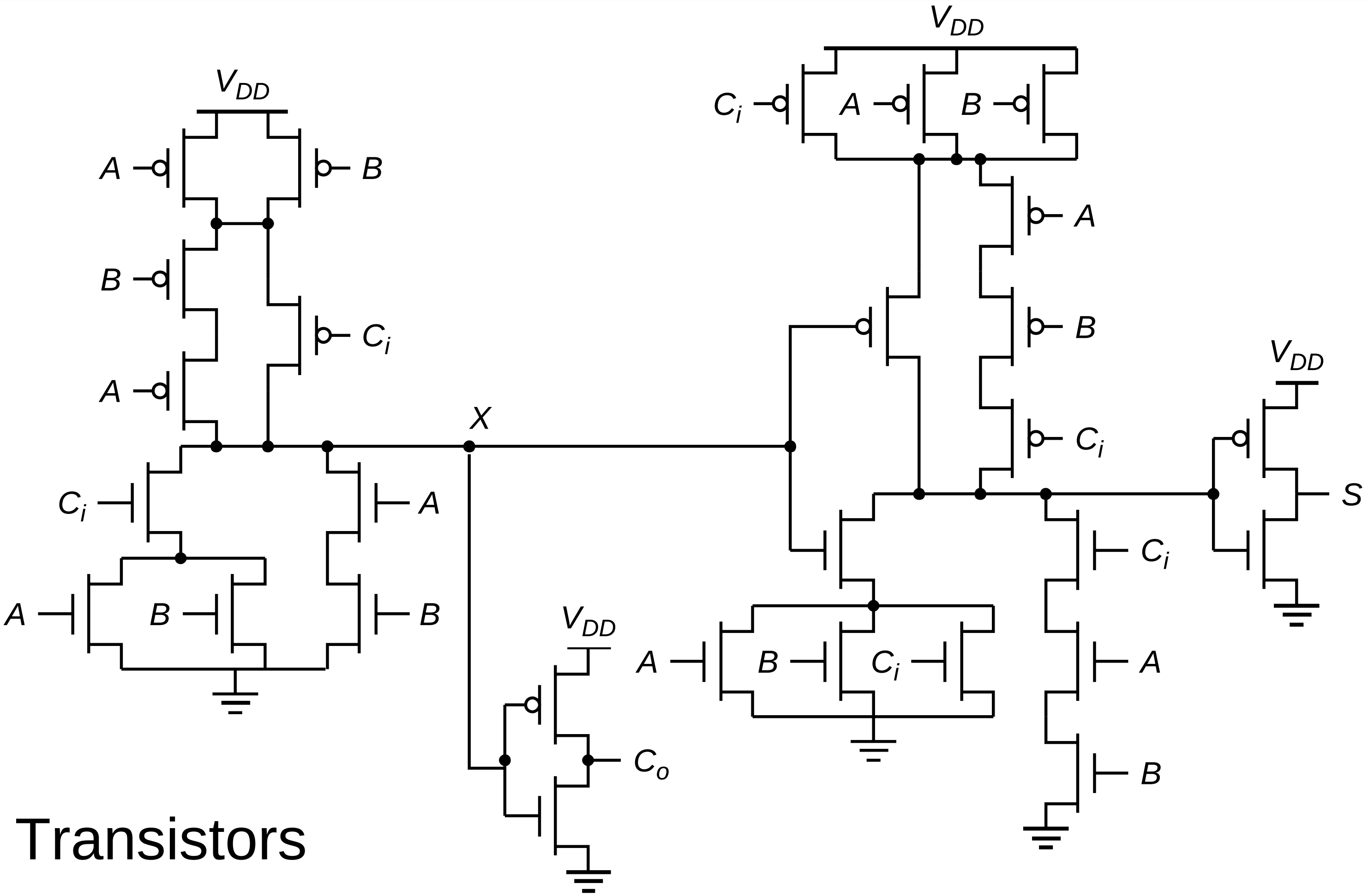
Worst case delay linear with the number of bits

$$t_d = \mathrm{O}(N)$$

$$t_{adder} = (N\text{-}1)t_{carry} + t_{sum}$$

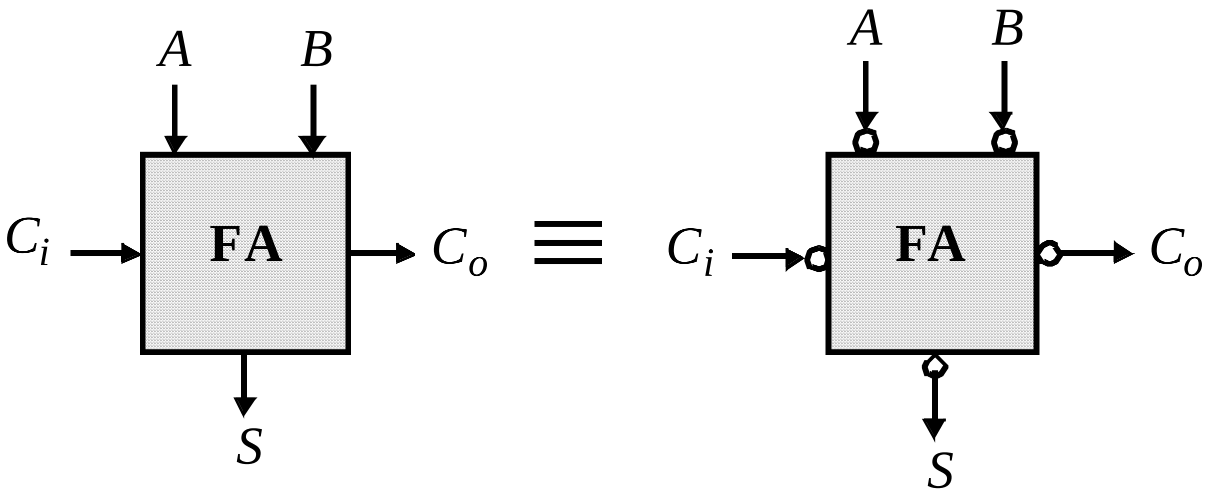
Goal: Make the fastest possible carry path circuit

Complementary Static CMOS Full Adder



28 Transistors

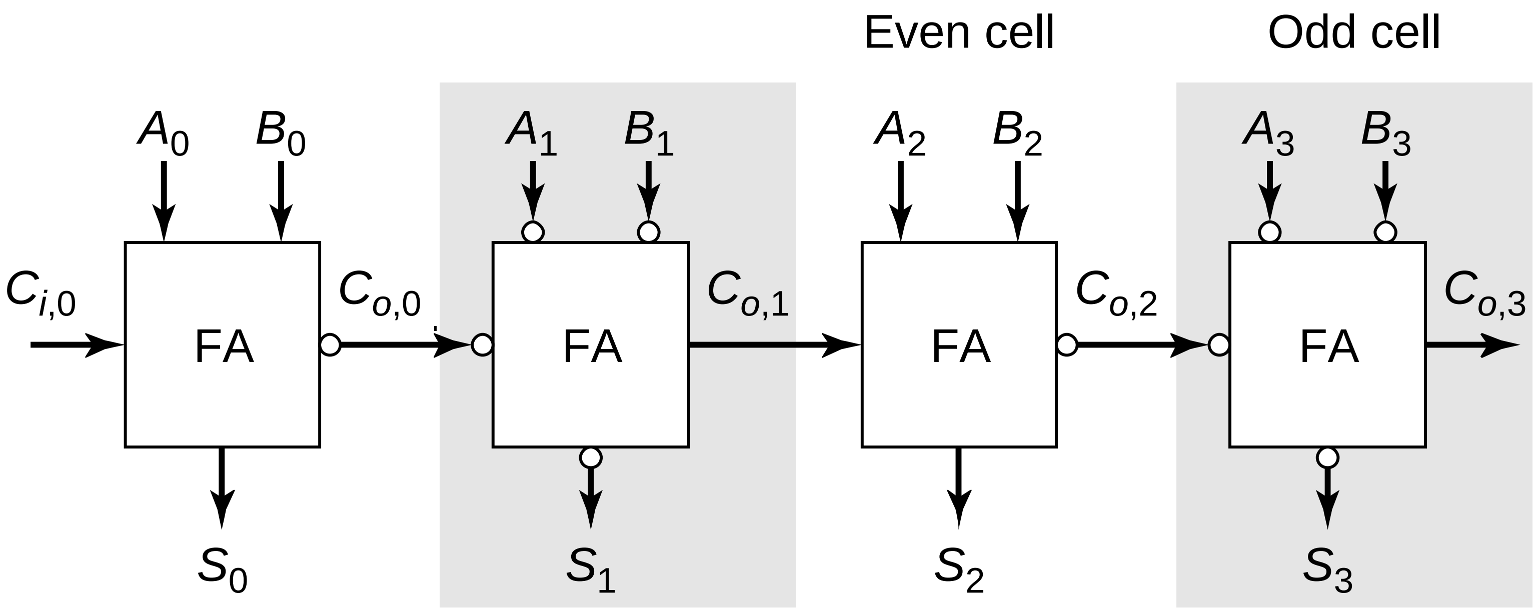
Inversion Property



$$\bar{S}(A,B,C_i) = S(\bar{A},\bar{B},\overline{C_i})$$

$$\overline{C_o}(A,B,C_i) = C_o(\bar{A},\bar{B},\overline{C_i})$$

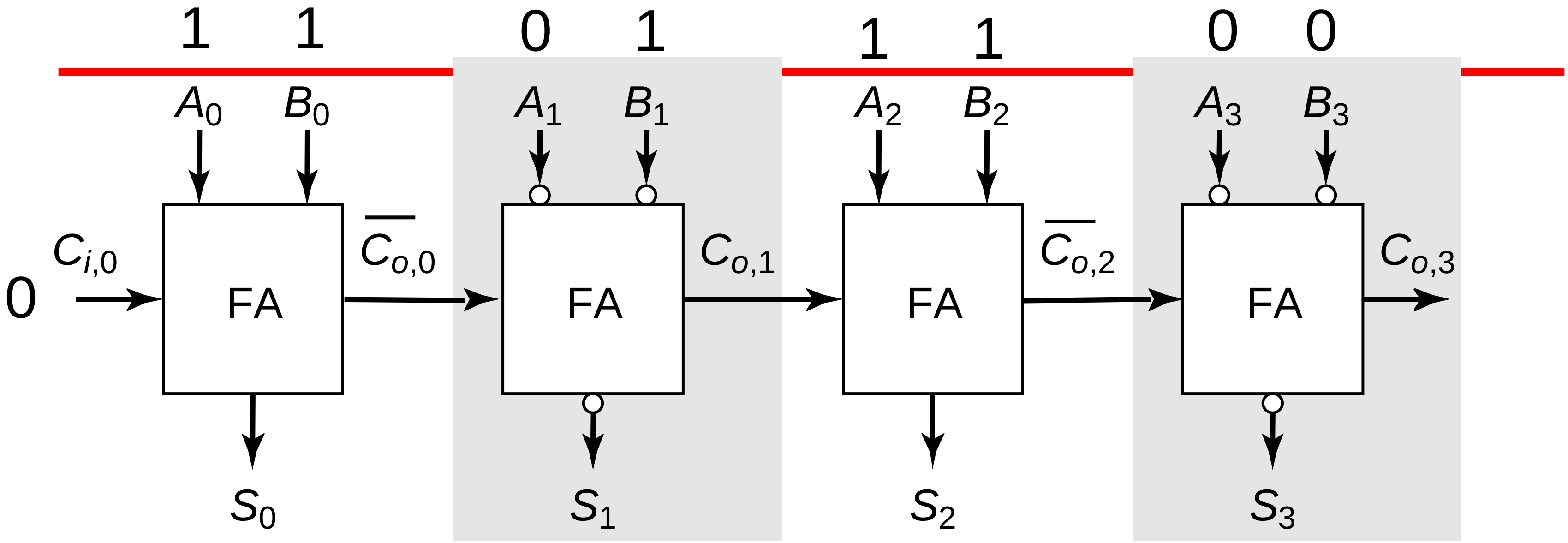
Minimize Critical Path by Reducing Inverting Stages Along Carry Path



Exploit Inversion Property

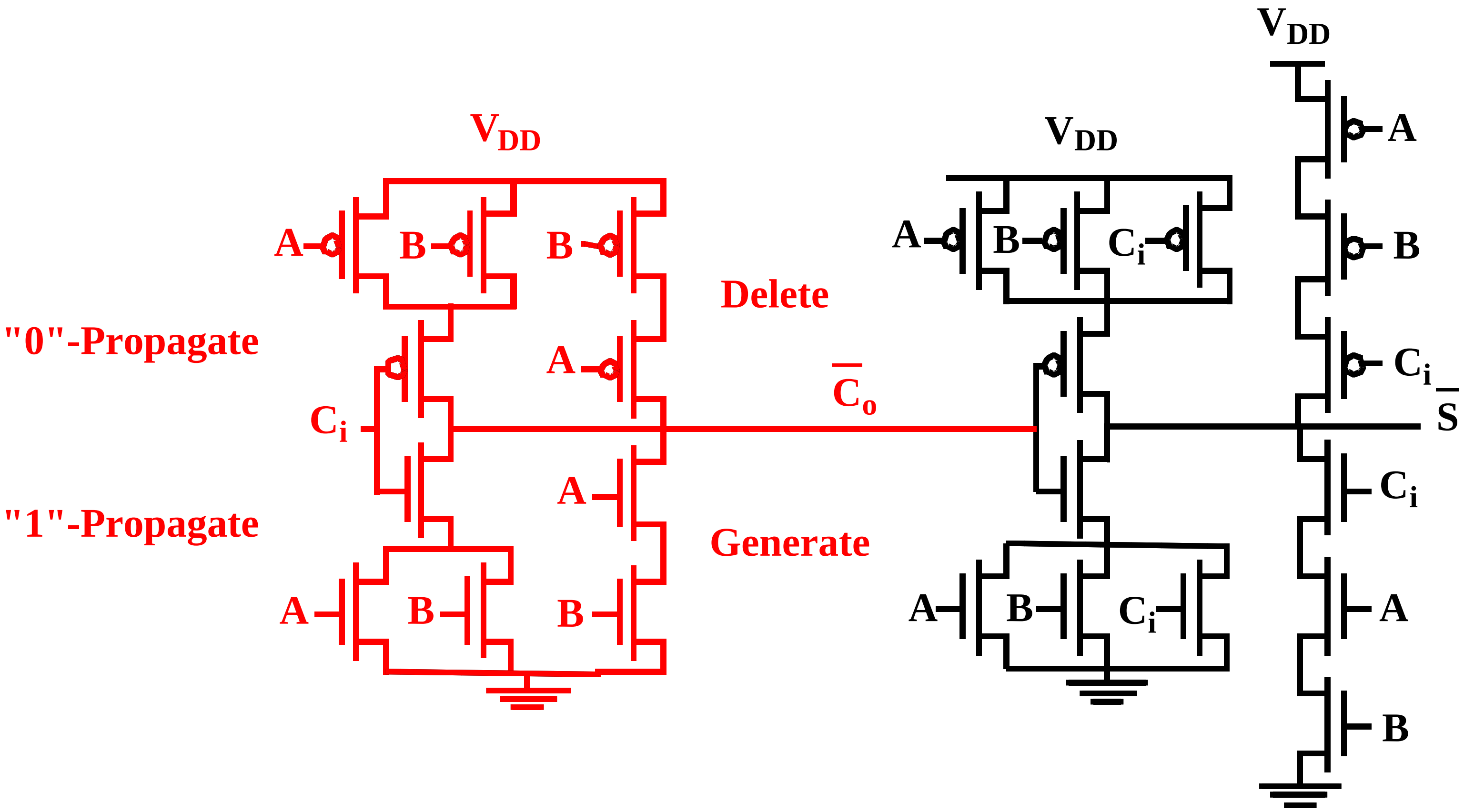
Allows us to remove inverter in carry chain → at what cost?

Example



1
1
0
1
1
1
0
0
A_0
B_0
A_1
B_1
A_2
B_2
A_3
B_3
0
$C_{i,0}$
FA
$\overline{C_{o,0}}$
FA
$C_{o,1}$
FA
$\overline{C_{o,2}}$
FA
$C_{o,3}$
S_0
S_1
S_2
S_3

A Better Structure: Mirror Adder



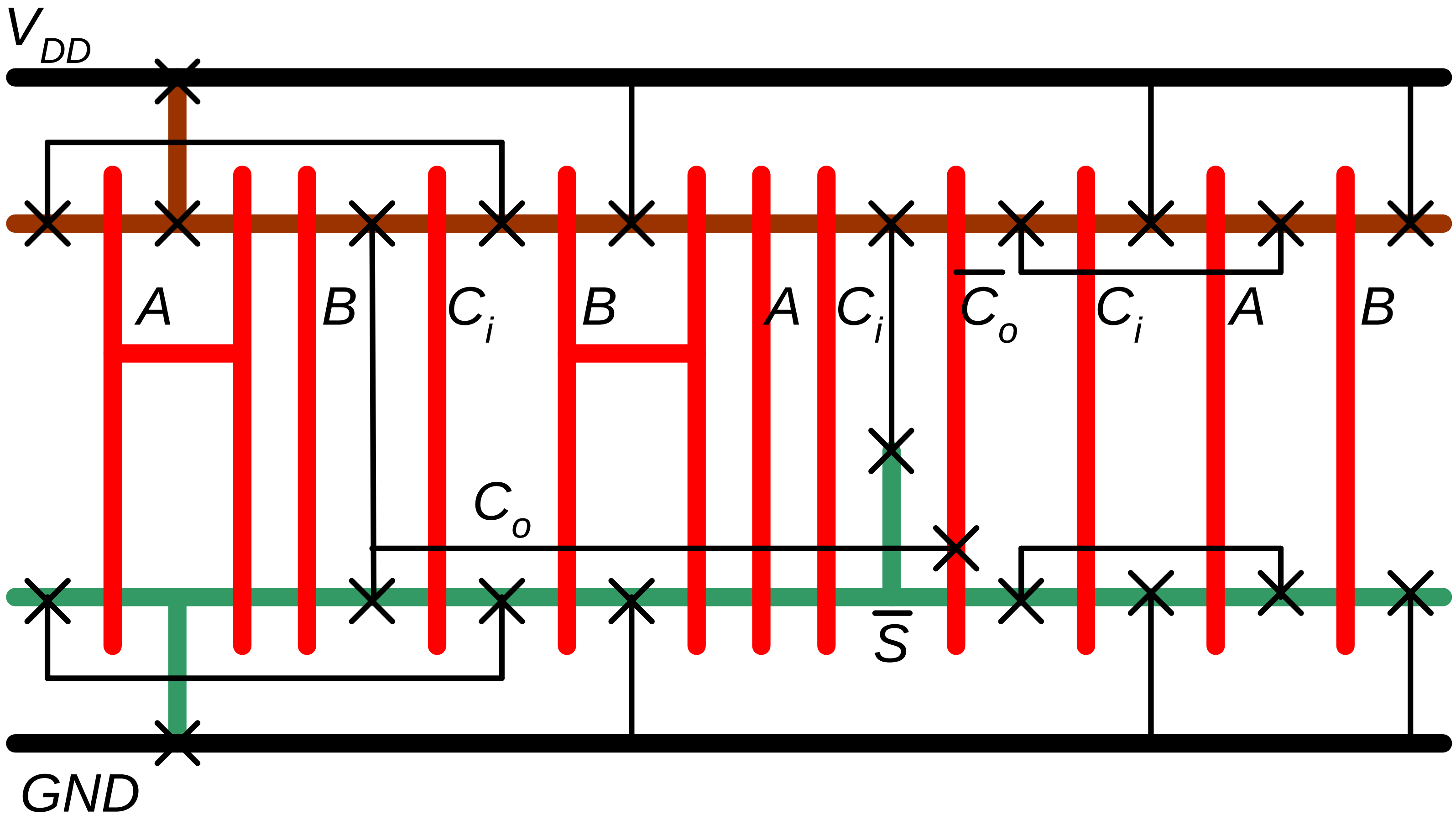
24 transistors

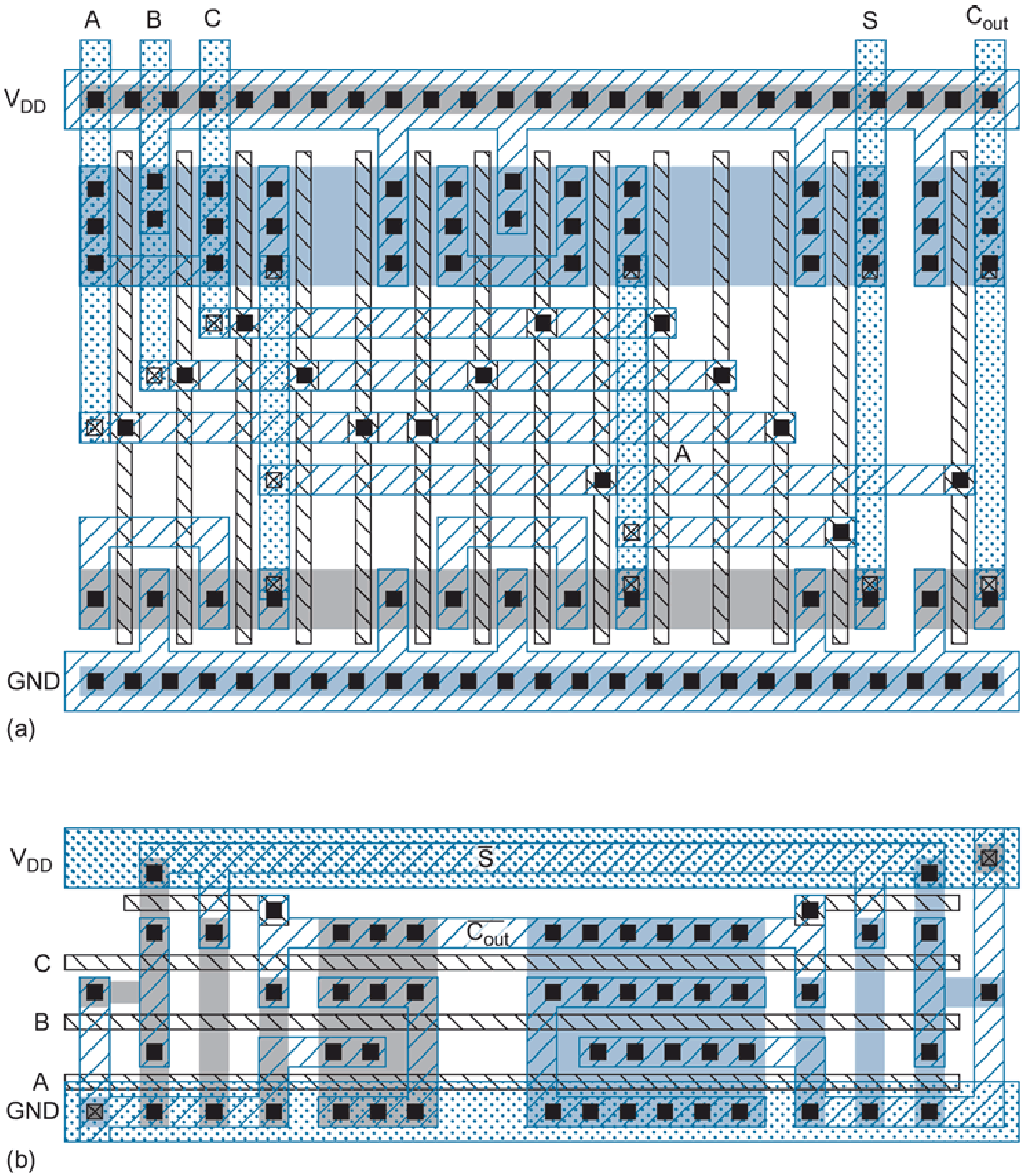
Mirror Adder Details

- NMOS and PMOS chains are completely symmetric. Maximum of 2 series transistors in carry generation
- In layout → critical to minimize capacitance at node C_o. Reduction of junction capacitances is particularly important
- Capacitance at node C_o is composed of 4 junction capacitances, 2 internal gate capacitances, and 6 gate capacitances in connecting adder cell
- Transistors connected to C_i are closest to output
- Only optimize transistors in carry stage for speed Transistors in sum stage can be small

Mirror Adder

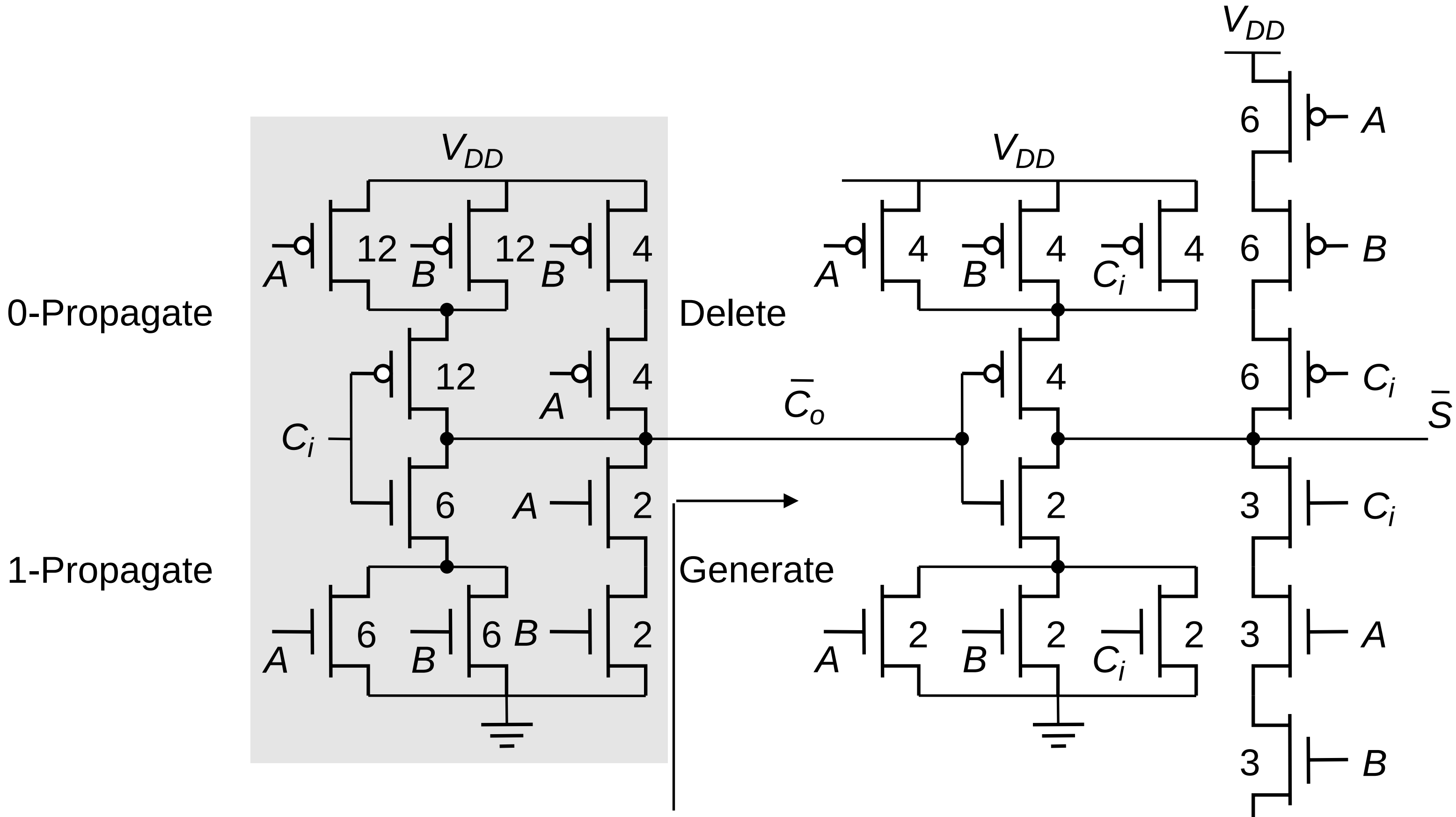
Stick diagram of layout





- 2 possible layouts of mirror adder
- (a) corresponds roughly to last slide’s stick diagram
- Layout (b) is datapath-oriented (ex. M2 can easily run horizontally across cell)

Sizing Mirror Adder



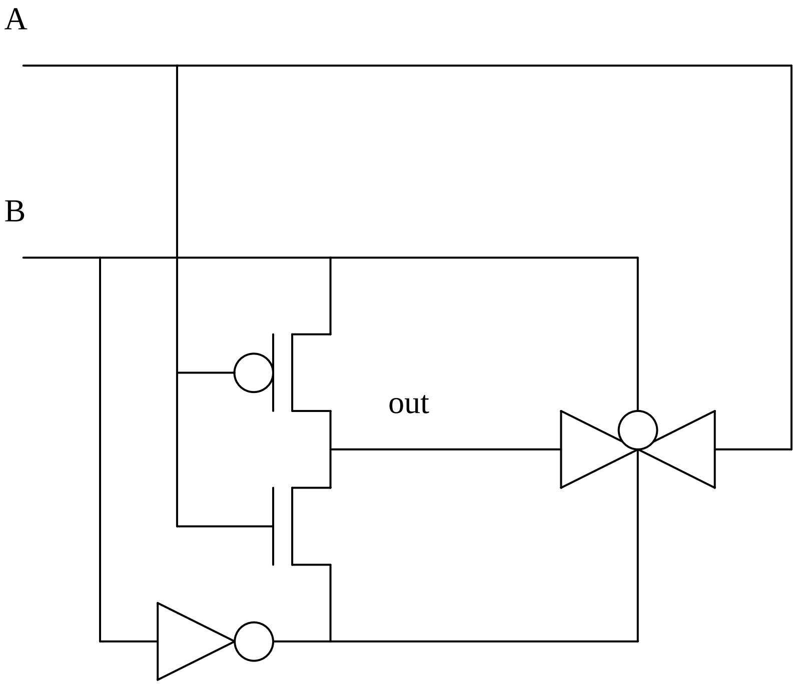
Fanout (effective) ~2

Building from smaller gates

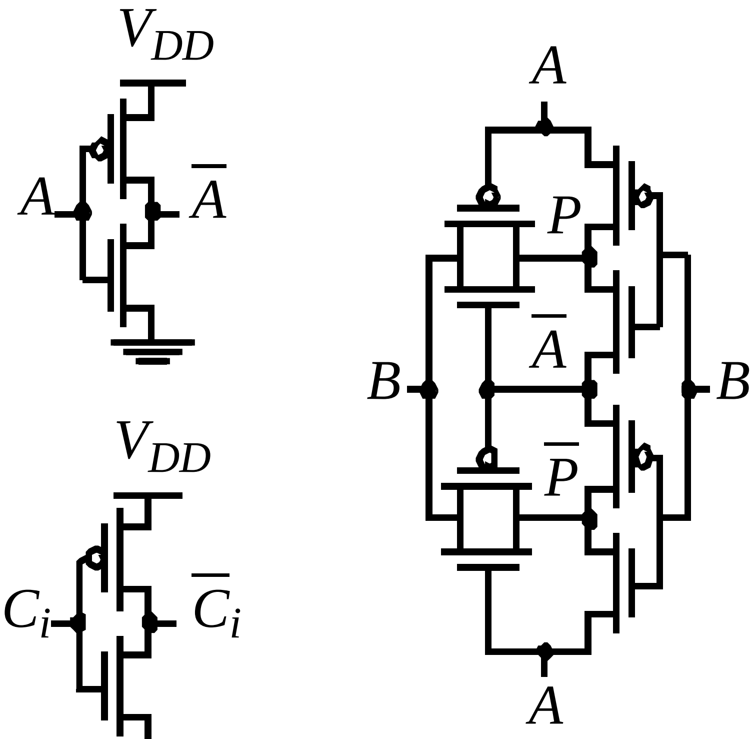
- Your ALU will need to do more than just add
- There are opportunities for sharing gates with other alu operations
- Do not feel the need to implement addition in giant monolithic gates with muxes. For example, what would mirror adder look like with P,G,D as inputs?
- XOR is the tough basic gate to make

Transmission gate XOR

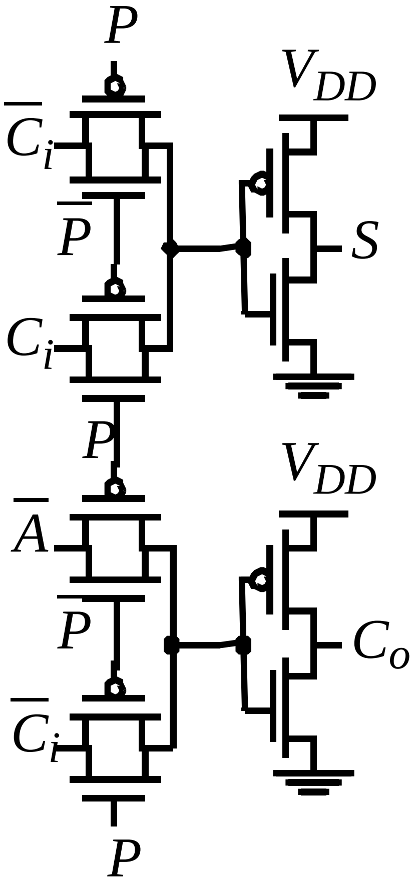
- Swap inverter to top for xnor gate



Transmission Gate Full Adder



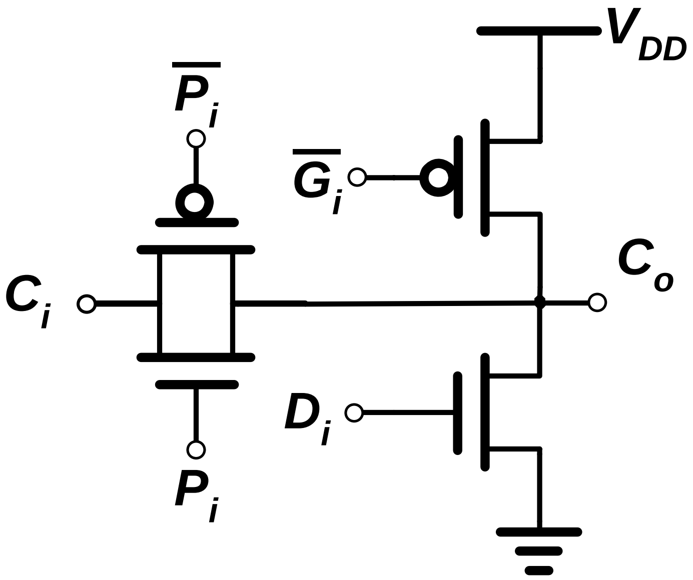
Setup



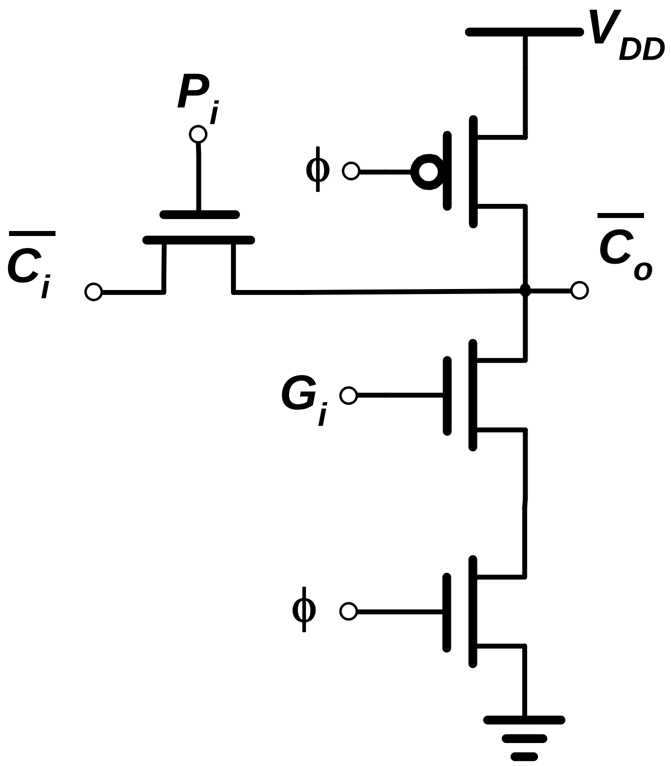
Sum Generation

Carry Generation

Manchester Carry Chain



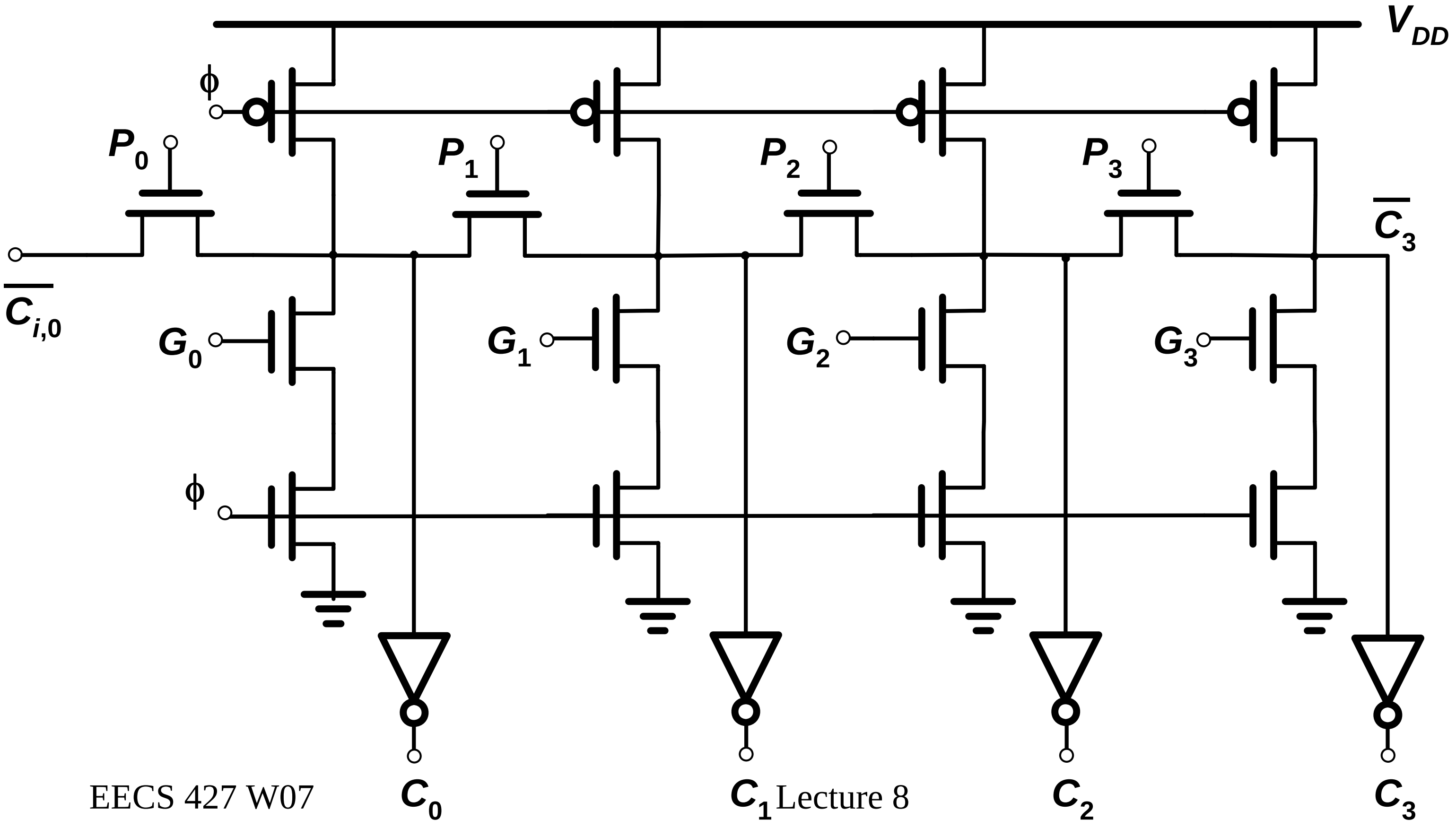
Static



Dynamic

Manchester Carry Chain

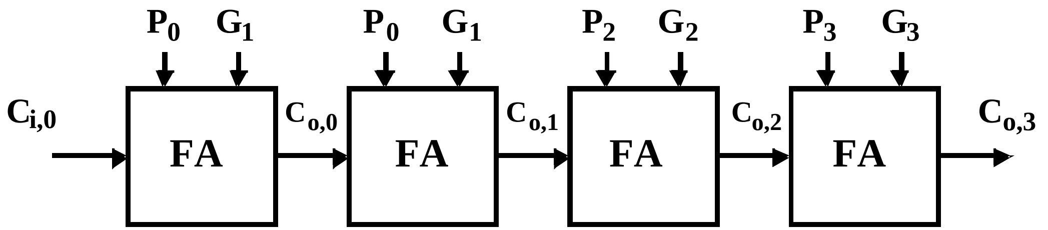
- •Implement P with pass-transistors
- •Implement G with pull-up OR kill (delete) with pull-down (note inversion)
- •Use dynamic logic to reduce complexity and speed up



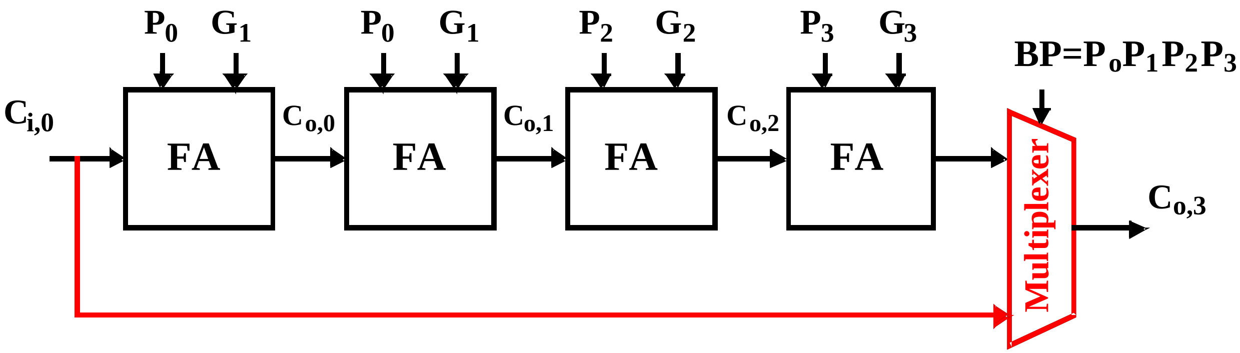
Change the Topology

- Instead of speeding up carry chain, try to shorten it
- Topologies to discuss:
  - Carry bypass (skip)
  - Carry select
  - Carry lookahead
  - Log lookahead
    - Brent-kung
    - Kogge-stone

Carry-Bypass Adder



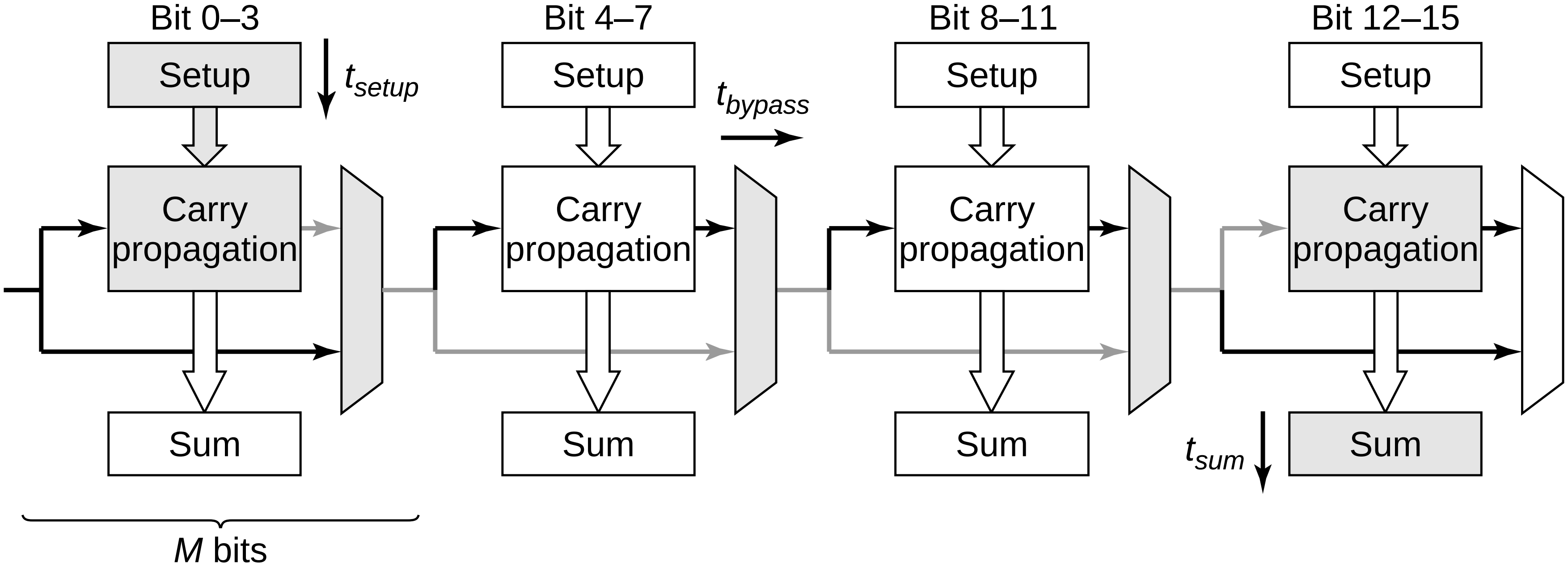
Also called Carry-Skip



Idea: If (P0 and P1 and P2 and P3 = 1)

then $C_{o3} = C_0$, else “delete” or “generate”

Carry-Bypass Adder (cont.)

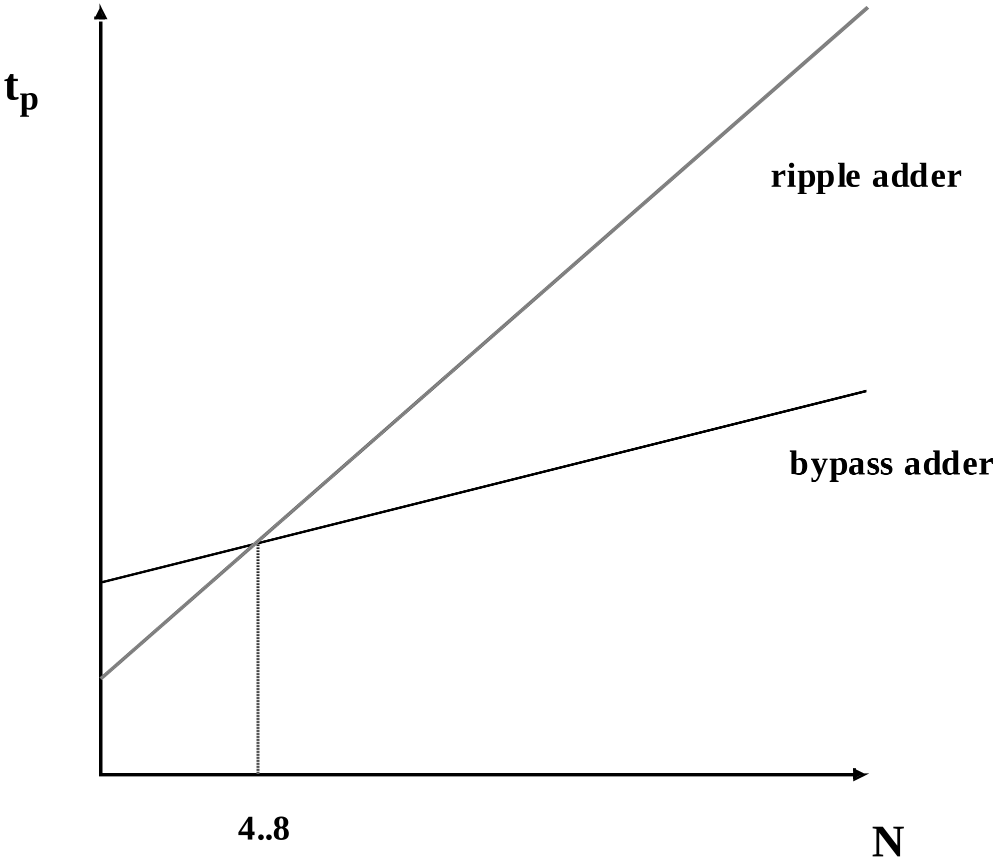


$$t_{adder} = t_{setup} + M_{tcarry} + (N/M\text{-}1)t_{bypass} + (M\text{-}1)t_{carry} + t_{sum}$$

Inner blocks do not contribute to worst-case delay since they have time to compute while bits 0-3 are propagating (assuming they have a generate or delete)

Block sizes can be made non-uniform (HOW?)

Carry Ripple versus Carry Bypass


t_p
ripple adder
bypass adder
N