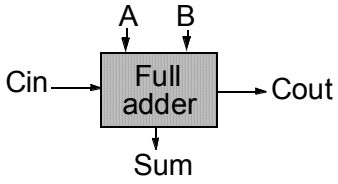
EECS 427
Lecture 7: Adders
Reading: 11.1 – 11.3.3

Last Time

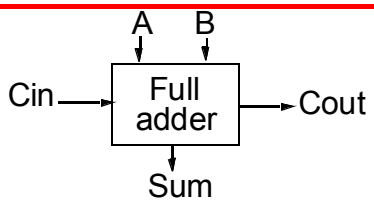
- Sizing & Interconnect
- Today:
  - Introduction to Adders.

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



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

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

$$C_o = AB + BC_i + AC_i$$

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

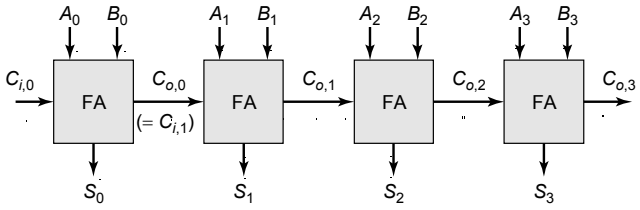
Delete = $\overline{A}\,\overline{B}$

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

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

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

The Ripple-Carry Adder



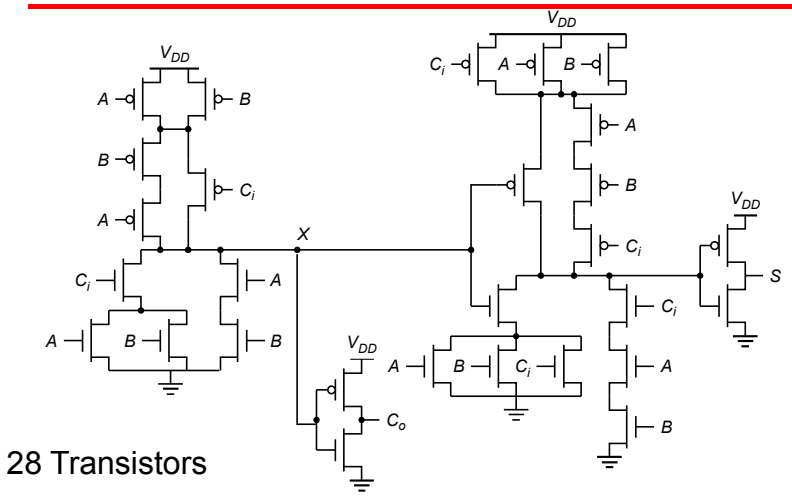
Worst case delay linear with the number of bits

$t_d = O(N)$

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

Goal: Make the fastest possible carry path circuit

Complementary Static CMOS Full Adder



28 Transistors

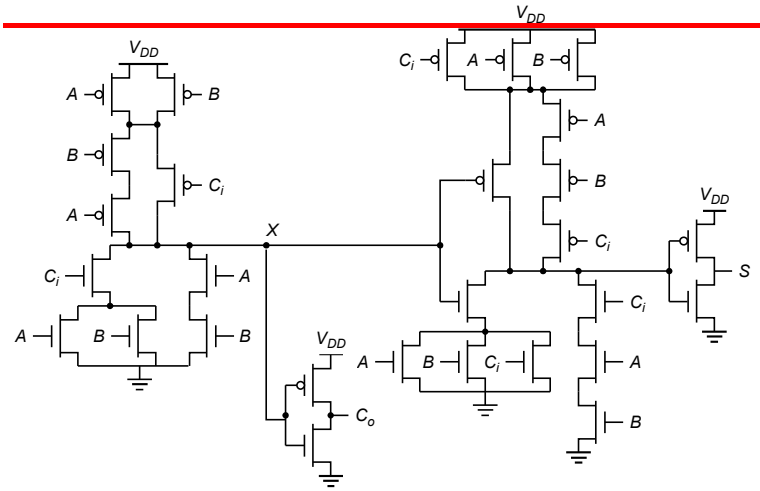
Summary So Far…

- Instruction set and general 2-stage pipeline structure of the baseline processor
  - Covered in discussion last time
- Adders are a critical part of any digital processor
  - Adder design requires an architecture/topology (ex: ripple carry) and a bit cell design (ex: std. static CMOS)
  - More architectures and cell designs…coming up

Next…

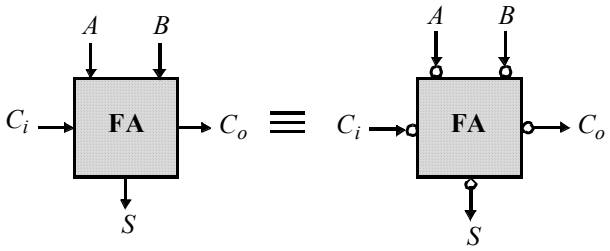
- Better full adder design
- Better adder architectures vs. ripple carry adder (RCA)
  - Carry skip adder
  - Linear Carry select adder
  - Square root carry select adder

Complementary Static CMOS Full Adder Revisited



28 Transistors

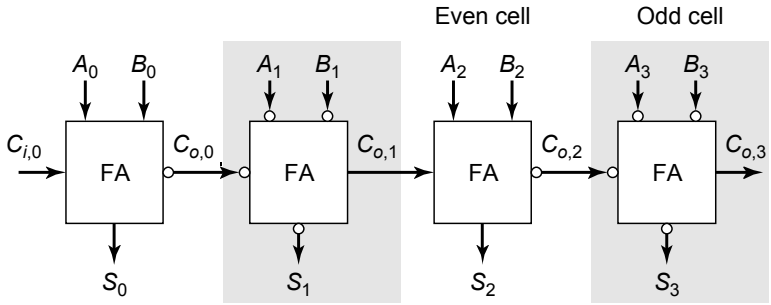
Inversion Property



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

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

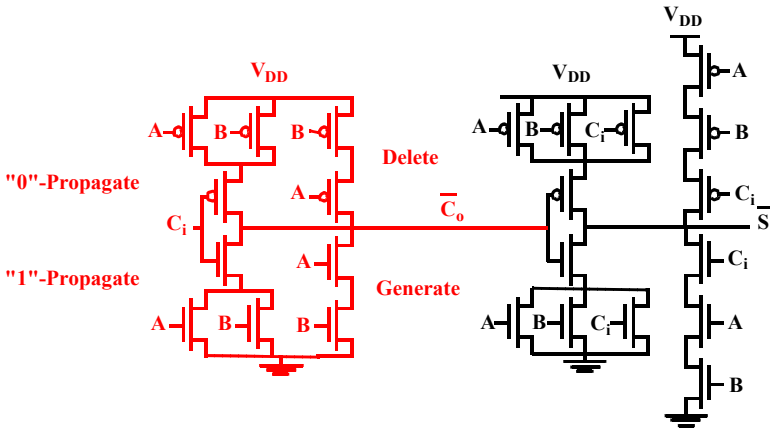
Minimize Critical Path by Reducing Inverting Stages Along Carry Path



Exploit Inversion Property

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

A Better Structure: Mirror Adder



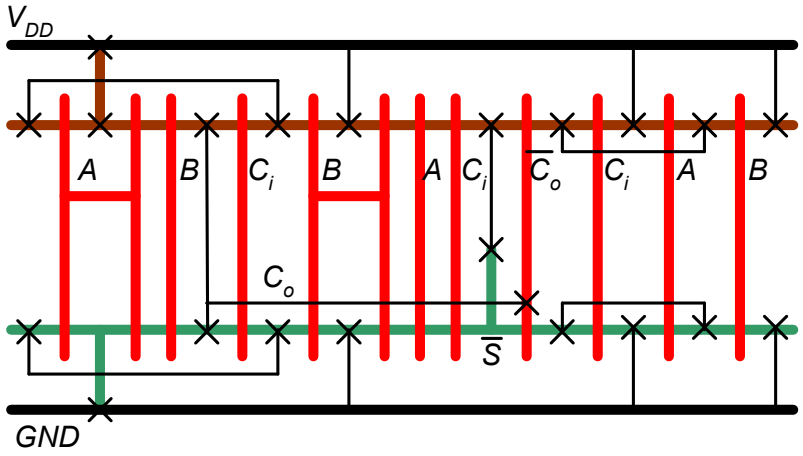
24 transistors

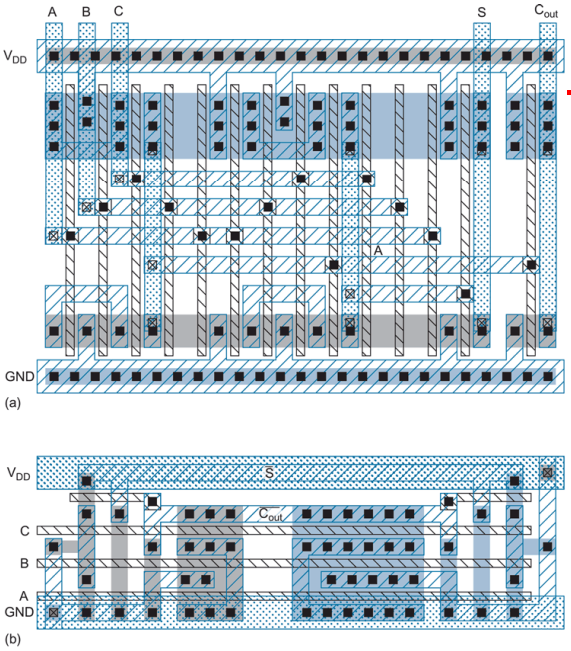
Mirror Adder Details

- 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

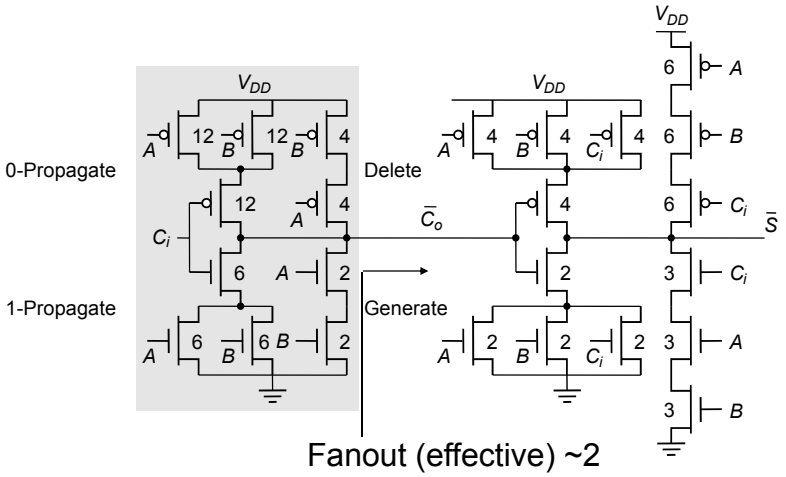
Stick diagram of layout



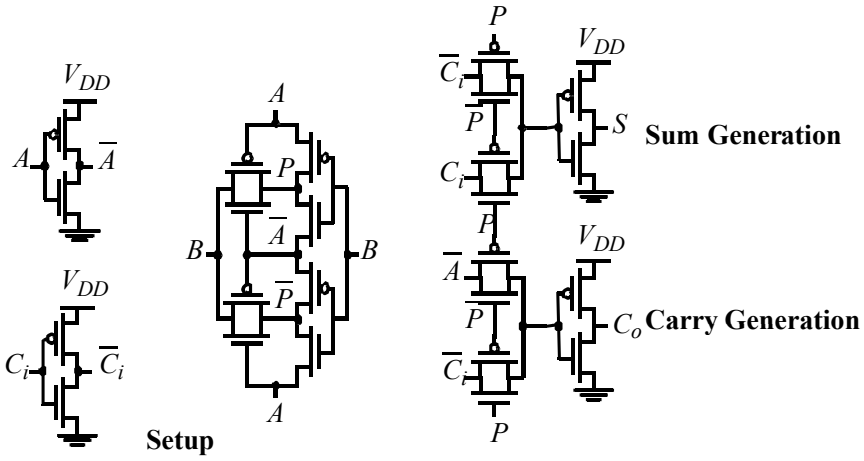


- 2 possible layouts of mirror adder
- (a) corresponds roughly to last slide's stick diagram
- Layout (b) is datapath-oriented (ex. M2 can easily run horizontally across cell)

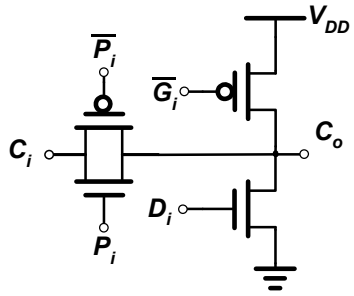
Sizing Mirror Adder



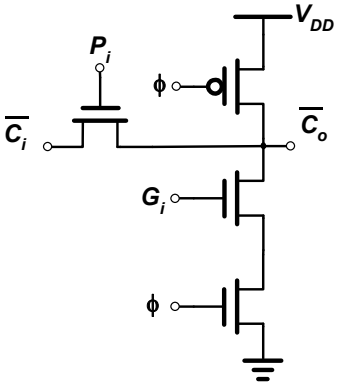
Transmission Gate Full Adder



Manchester Carry Chain



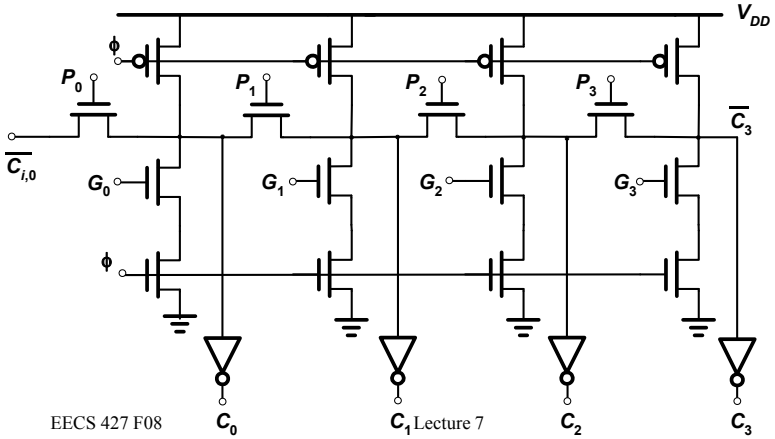
Static



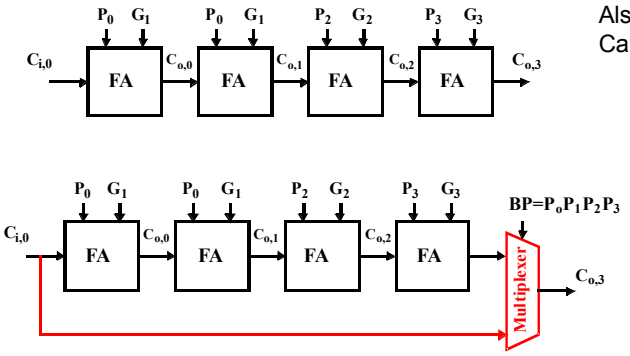
Dynamic

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



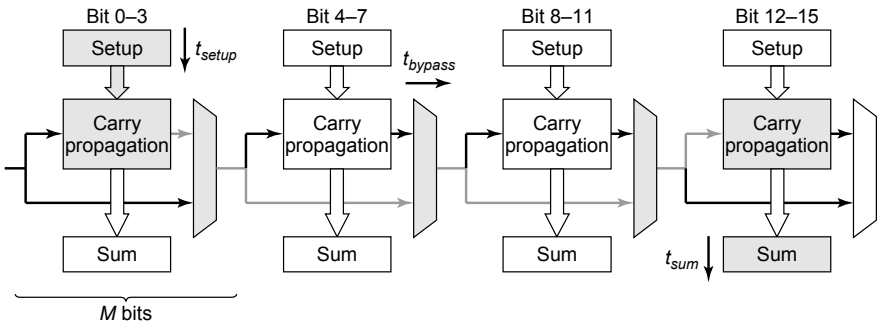
Carry-Bypass Adder



Also called Carry-Skip

Idea: If (P0 and P1 and P2 and P3 = 1) then $C_{o3} = C_{i,0}$, else "delete" or "generate"

Carry-Bypass Adder (cont.)

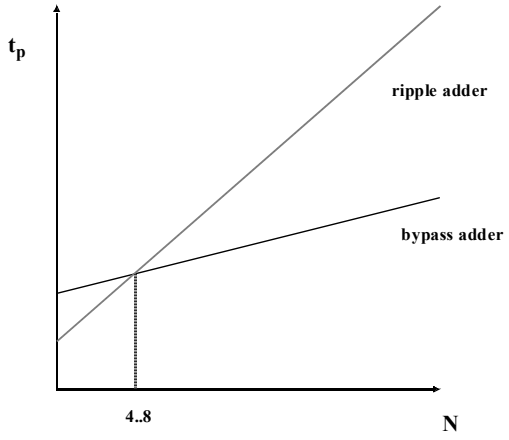


$$t_{adder} = t_{setup} + M_{tcarry} + (N/M\text{-}1)t_{bypass} + (M\text{-}1)t_{carry} + t_{sum}$$

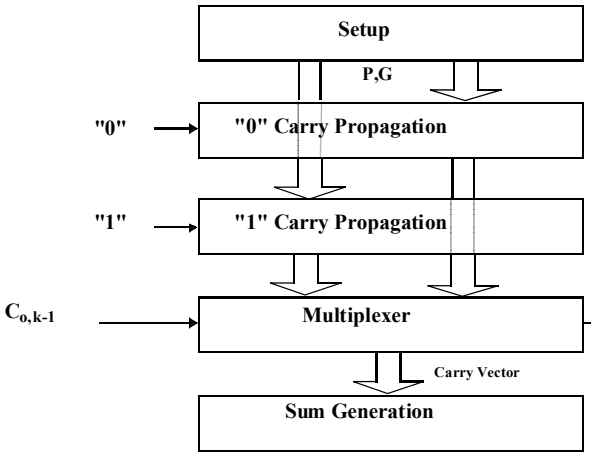
Inner blocks do not contribute to worst-case delay since they have time to compute while bits 0-3 are propagating (assuming they have a generate or delete)

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

Carry Ripple versus Carry Bypass

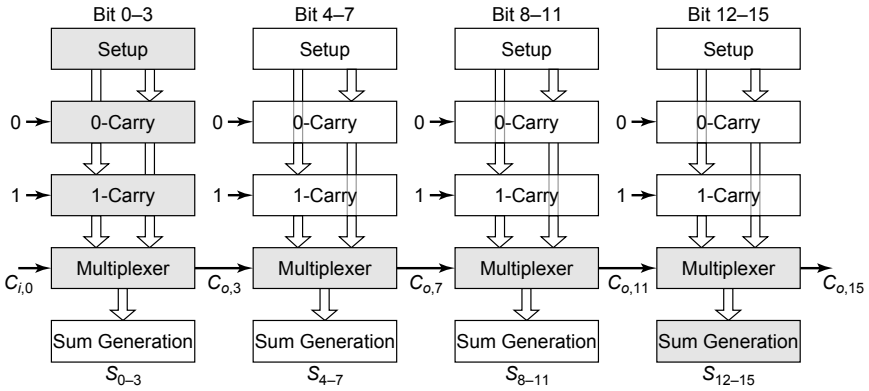


Carry-Select Adder



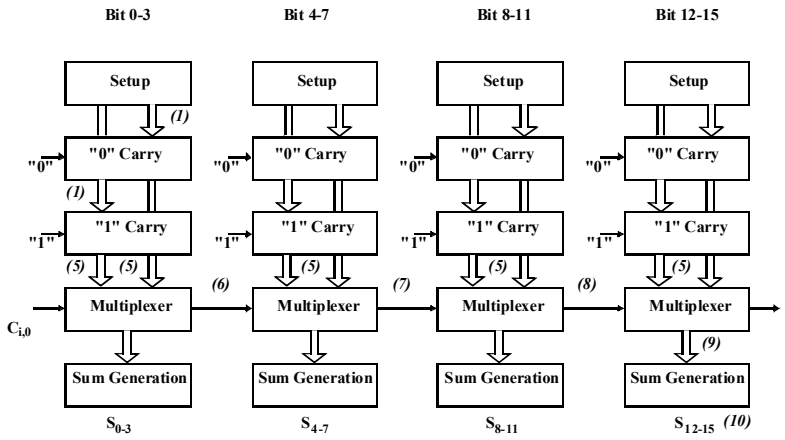
Basic idea: Precompute both possibilities, choose when C_{in} available

Carry Select Adder: Critical Path



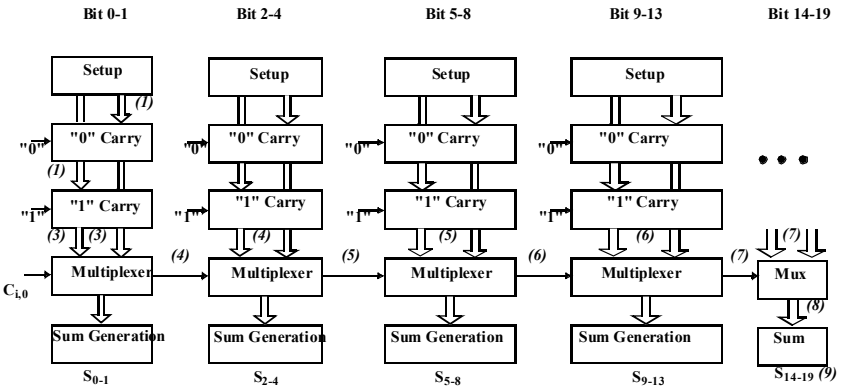
$$t_{add} = t_{setup} + Mt_{carry} + (N/M)t_{mux} + t_{sum}$$

Linear Carry Select



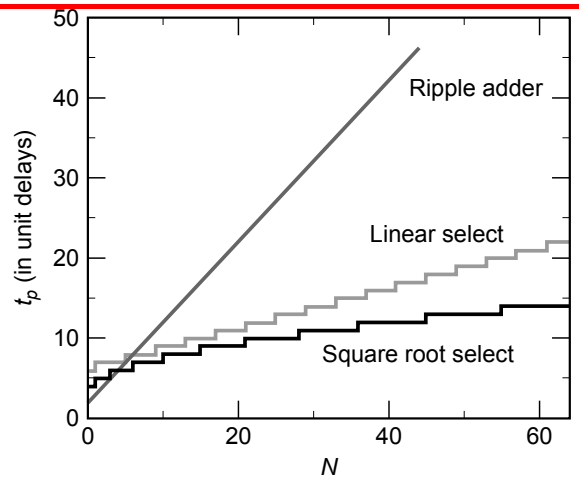
$$t_{add} = t_{setup} + M^{*}t_{carry} + (N/M)t_{mux} + t_{sum}$$

Square Root Carry Select



$$t_{add} = t_{setup} + Mt_{carry} + \left(\sqrt{2N}\right)t_{mux} + t_{sum}$$

Adder Delays - Comparison



Summary

- Many topologies for adders
  - Variants on carry lookahead (not discussed) are dominant today
- For 16-bit addition, complex techniques such as carry lookahead do not offer much benefit
  - Carry select and carry bypass yield good performance in this case
- Adder cells may use mirror structure or transmission gates