# EECS 427
# Lecture 9: Adders
Reading: 11.3.2 – 11.3.3 (limited carry lookahead coverage)



## Last Time

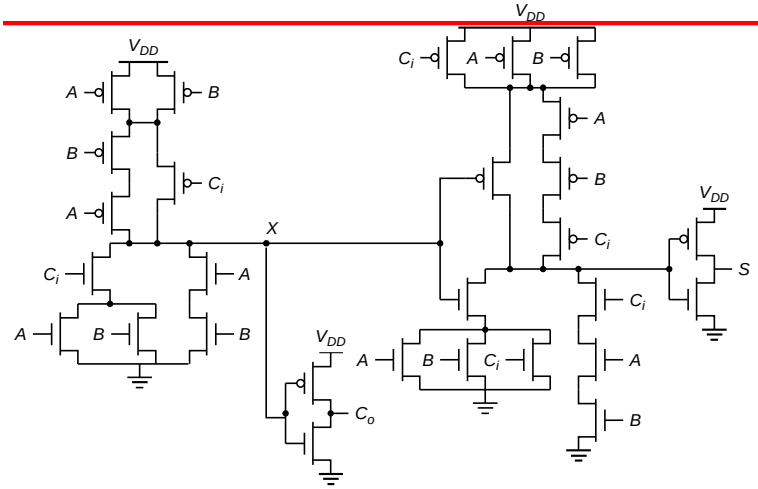
- Finished logical effort
- Adder intro
  - Ripple carry adder is simplest design but slow (delay grows linearly with # of bits)
  - Standard CMOS implementation is ugly
  - Key point: Inputs (A and B) are available at time zero, carry in is not

# Lecture Overview

- Better full adder design
- Better adder architectures vs. ripple carry adder (RCA)
- HW3 due Tuesday by beginning of class
  - Initial project proposal, only a brief description required
  - Can email it to me, or turn in hard copy in lecture
- Tuesday lecture on multipliers to be covered by Jerry Kao
  - ISSCC

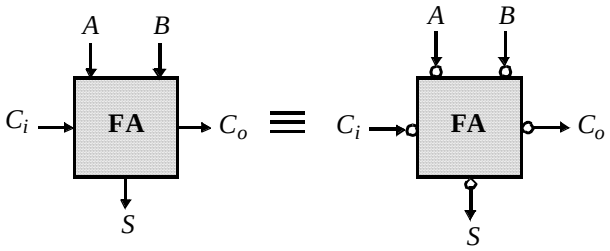


# Complementary Static CMOS Full Adder Revisited




28 Transistors

# Inversion Property

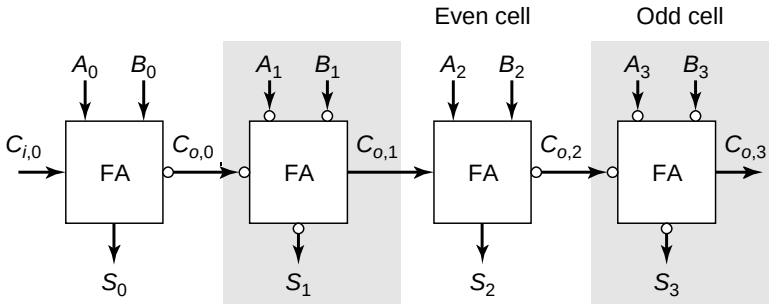



$$\bar{S}(A,B,C_i) = S(\bar{A},\bar{B},\bar{C_i})$$

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



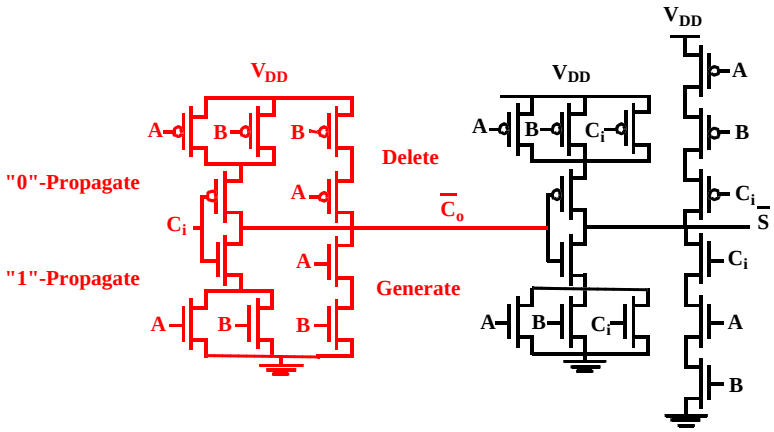
# Minimize Critical Path by Reducing Inverting Stages Along Carry Path




**Exploit Inversion Property**

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

# A Better Structure: Mirror Adder




**24 transistors**

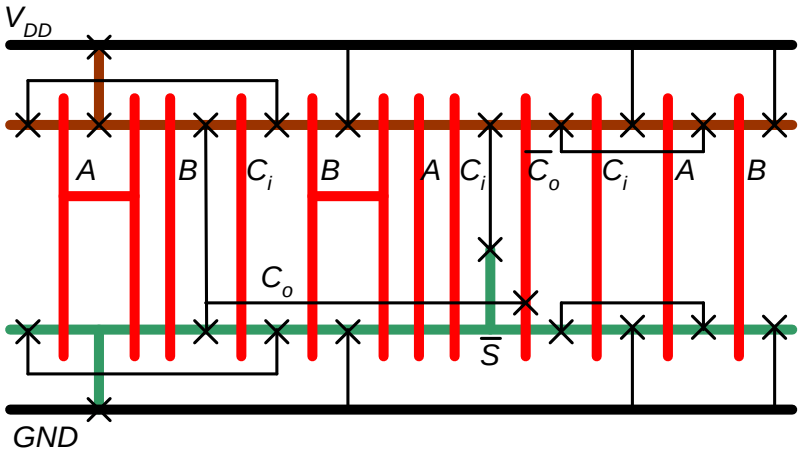


# Mirror Adder Details

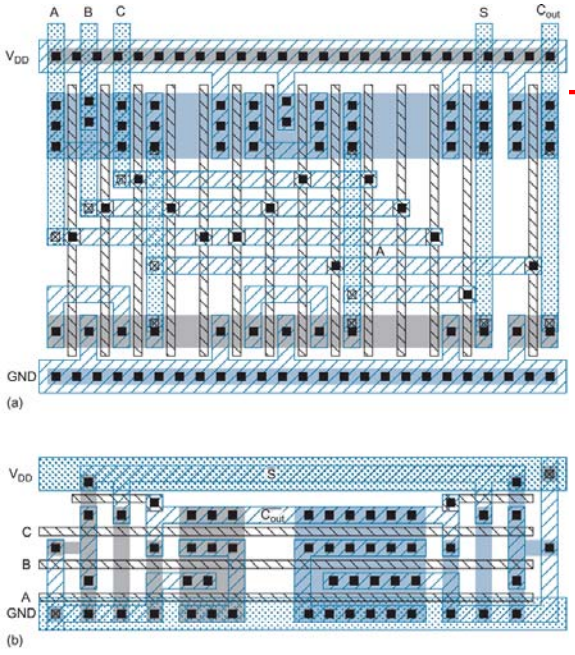
- NMOS and PMOS chains are completely symmetric. Maximum of 2 series transistors in carry generation
- In layout → critical to minimize capacitance at node $C_o$. Reduction of junction capacitances is particularly important
- Capacitance at node $C_o$ is composed of 4 junction capacitances, 2 internal 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

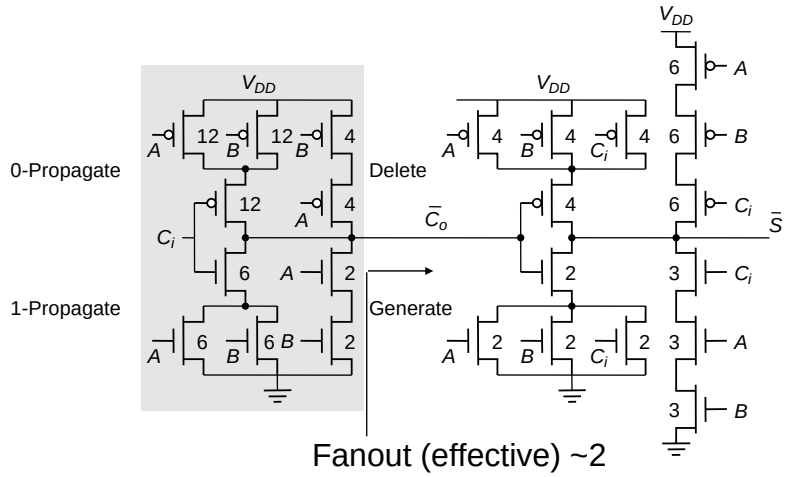





- 2 possible layouts of mirror adder
- (a) corresponds roughly to last slide's stick diagram
- Layout (b) is datapath-oriented (ex. M2 can easily run horizontally across cell)

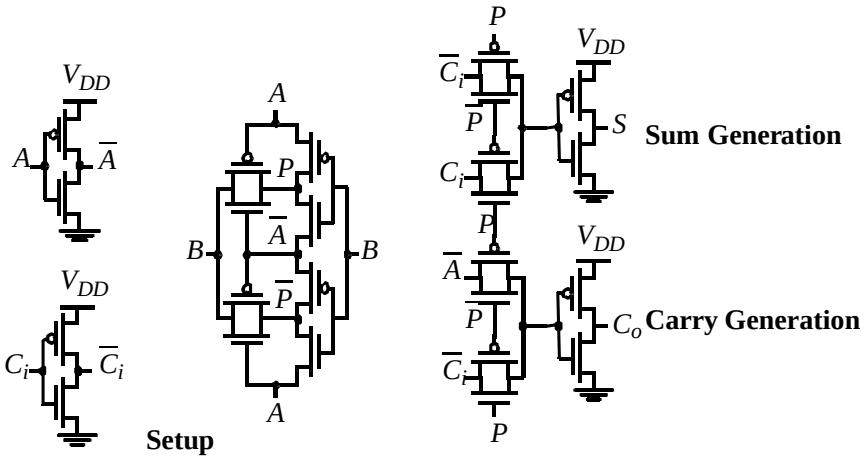
# Sizing Mirror Adder



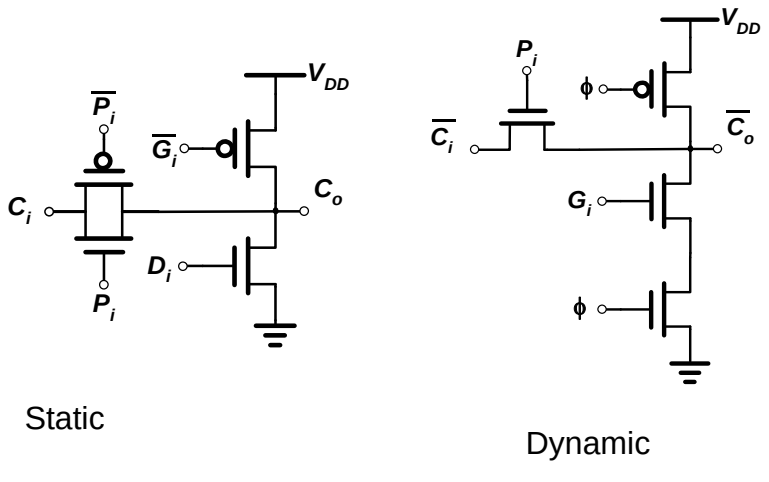

Fanout (effective) ~2



# Transmission Gate Full Adder



# Manchester Carry Chain



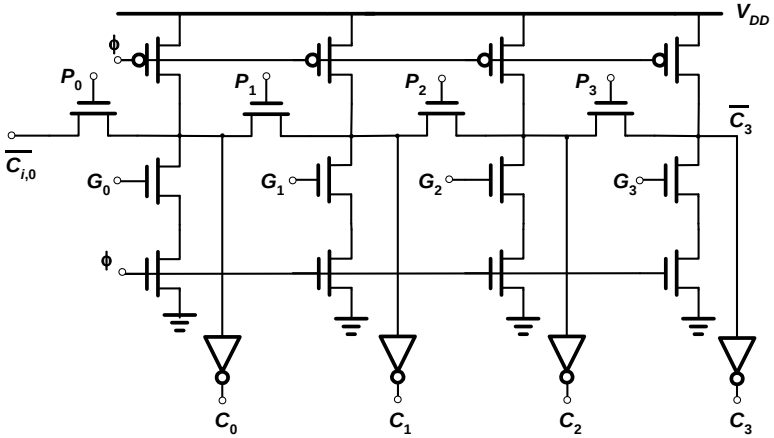

Static

Dynamic

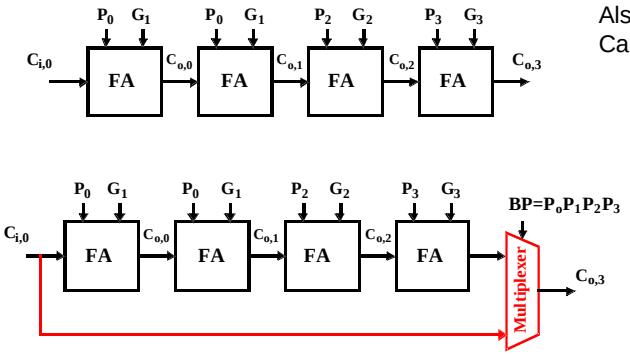


# Manchester Carry Chain

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



# Carry-Bypass Adder

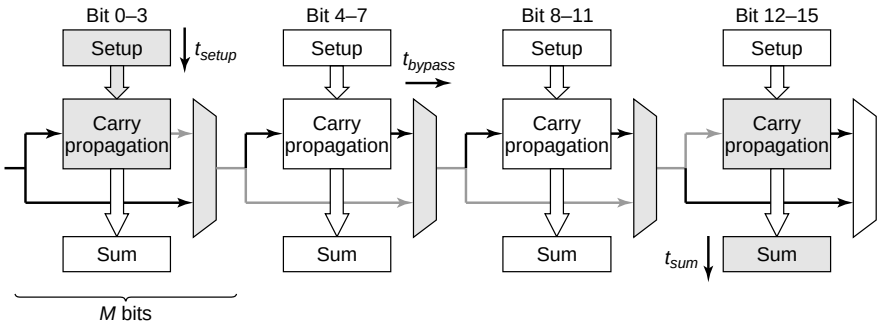



Also called Carry-Skip

Idea: If (P0 and P1 and P2 and P3 = 1) then $C_{o3} = C_0$, else "delete" or "generate"



# Carry-Bypass Adder (cont.)

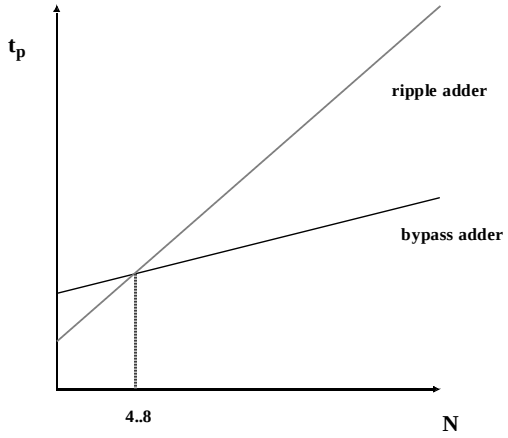



$$t_{adder} = t_{setup} + Mt_{carry} + (N/M\text{-}1)t_{bypass} + (M\text{-}1)t_{carry} + t_{sum}$$

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)

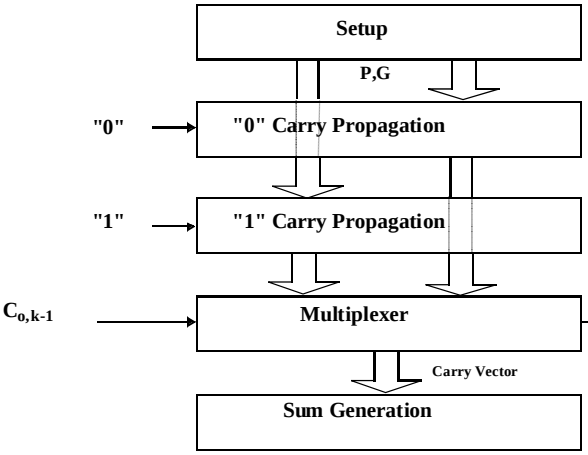
Block sizes can be made non-uniform (HOW?)

# Carry Ripple versus Carry Bypass



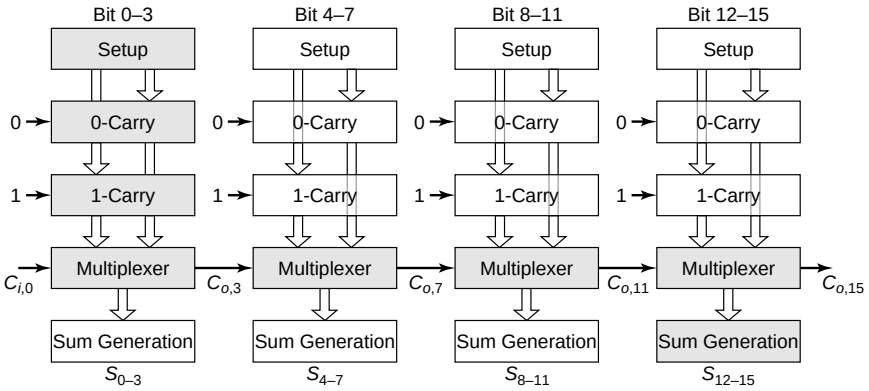



# Carry-Select Adder



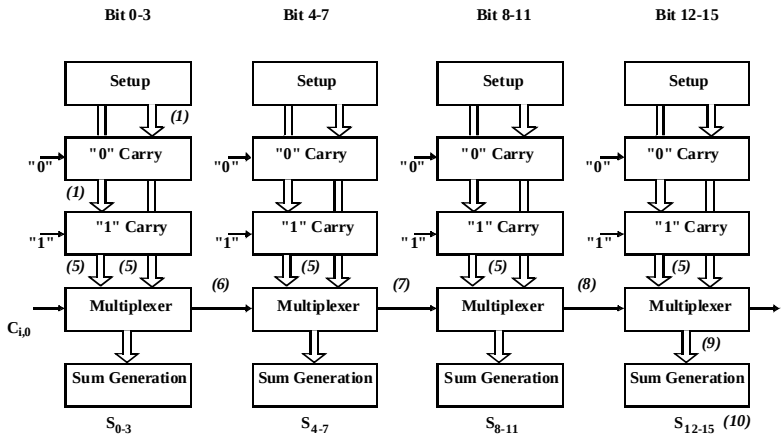

Basic idea:
Precompute both possibilities, choose when $C_{in}$ available

# Carry Select Adder: Critical Path



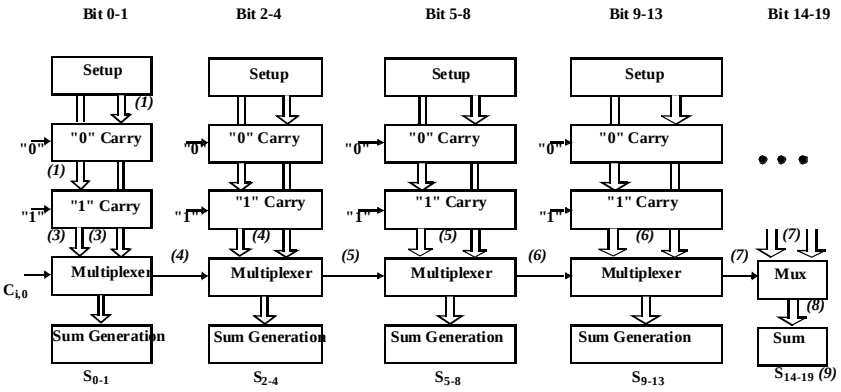



# Linear Carry Select



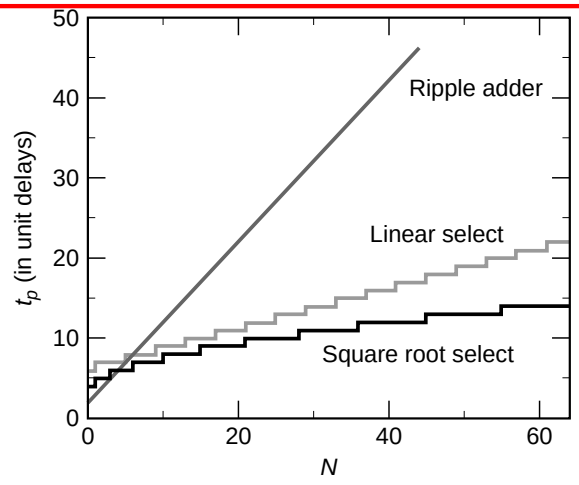

$$t_{add} = t_{setup} + M*t_{carry} + (N/M)t_{mux} + t_{sum}$$

# Square Root Carry Select

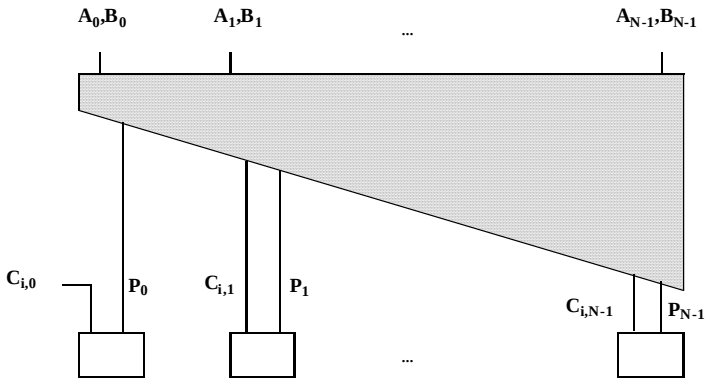





# Adder Delays - Comparison



# Carry Lookahead Adder




Weinberger, Smith, 1958.



# Lookahead Adder

Lookahead Equations

Position *i*: $c_i = g_i + p_i c_{i-1}$

Position *i* + 1:
$$c_{i+1} = g_{i+1} + p_{i+1} c_i$$
$$= g_{i+1} + p_{i+1}(g_i + p_i c_{i-1})$$
$$= g_{i+1} + p_{i+1} g_i + p_{i+1} p_i c_{i-1}$$

Carry exists if:

- generated in stage *i* + 1
- generated in stage *i* and propagated through *i* + 1
- propagated through both *i* and *i* + 1

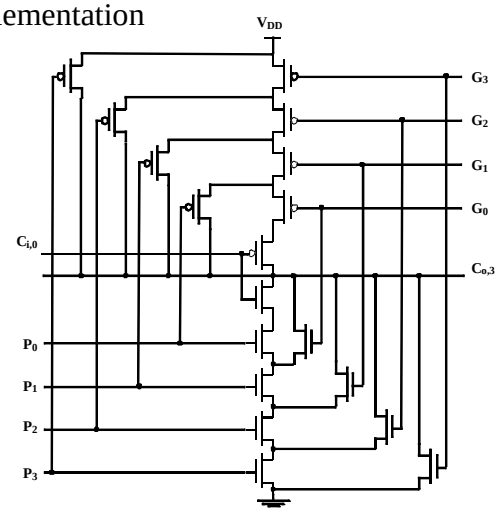
# Lookahead Adder

- Unrolling of carry recurrence can be continued
- If unrolled to level *k*, resulting in two-level AND-OR structure
  - AND Fan-In = *k* + 1, OR Fan-In = *k* + 1
  - *k* + 1 transistors in the MOS stack
  - Limits *k* to 3 -4



# Lookahead Adder

Mirror Implementation



# Block Lookahead

4th bit carry:

$$c_{i+3} = g_{i+3} + p_{i+3}g_{i+2} + p_{i+3}p_{i+2}g_{i+1} + p_{i+3}p_{i+2}p_{i+1}g_i + p_{i+3}p_{i+2}p_{i+1}p_i c_{i-1}$$

Block generate and block propagate:

$$G_{i,i+3} = g_{i+3} + p_{i+3}g_{i+2} + p_{i+3}p_{i+2}g_{i+1} + p_{i+3}p_{i+2}p_{i+1}g_i$$

$$P_{i,i+3} = p_{i+3}p_{i+2}p_{i+1}p_i$$

$$c_{i+3} = G_{i,i+3} + P_{i,i+3}c_{i-1}$$

# Block Lookahead

Can create groups of groups, or 'super-groups':

$$G^*_{j+3:j} = G_{j+3} + P_{j+3}G_{j+2} + P_{j+3}P_{j+2}G_{j+1} + P_{j+3}P_{j+2}P_{j+1}G_j$$

$$P^*_{j+3:j} = P_{j+3}P_{j+2}P_{j+1}p_j$$

$$t_d \propto \log N$$

An overview of hierarchical tree-style carry lookahead structures:

http://fourier.eng.hmc.edu/e85/lectures/arithmetic_html/node7.html

# Summary

- Many topologies for adders
  - Extensions of carry lookahead 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
  - Can string together 4-bit lookahead structures in a ripple fashion to achieve decent performance
- Adder cells may use mirror structure or transmission gates