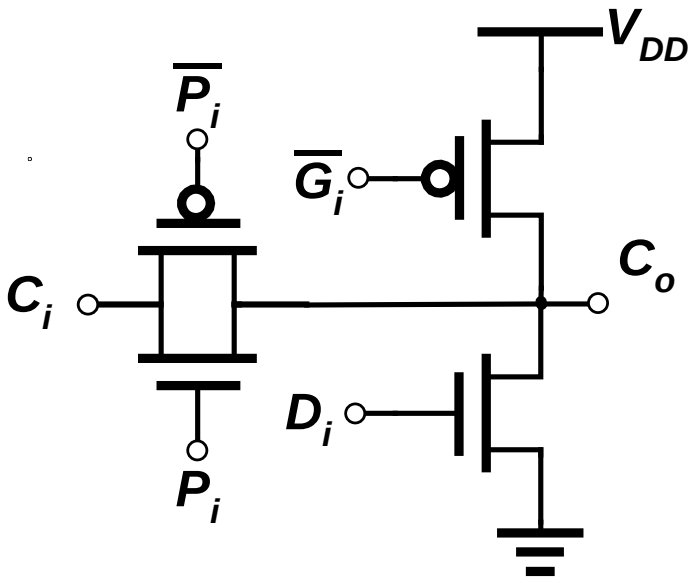
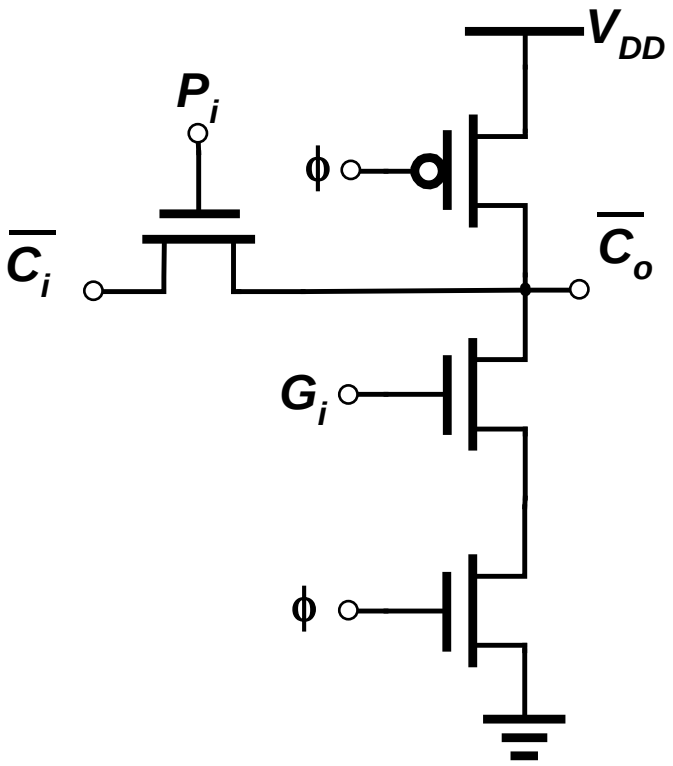
EE 466/586
VLSI Design

Partha Pande
School of EECS
Washington State University
pande@eecs.wsu.edu

Lecture 22
Arithmetic circuits (Cont'd)

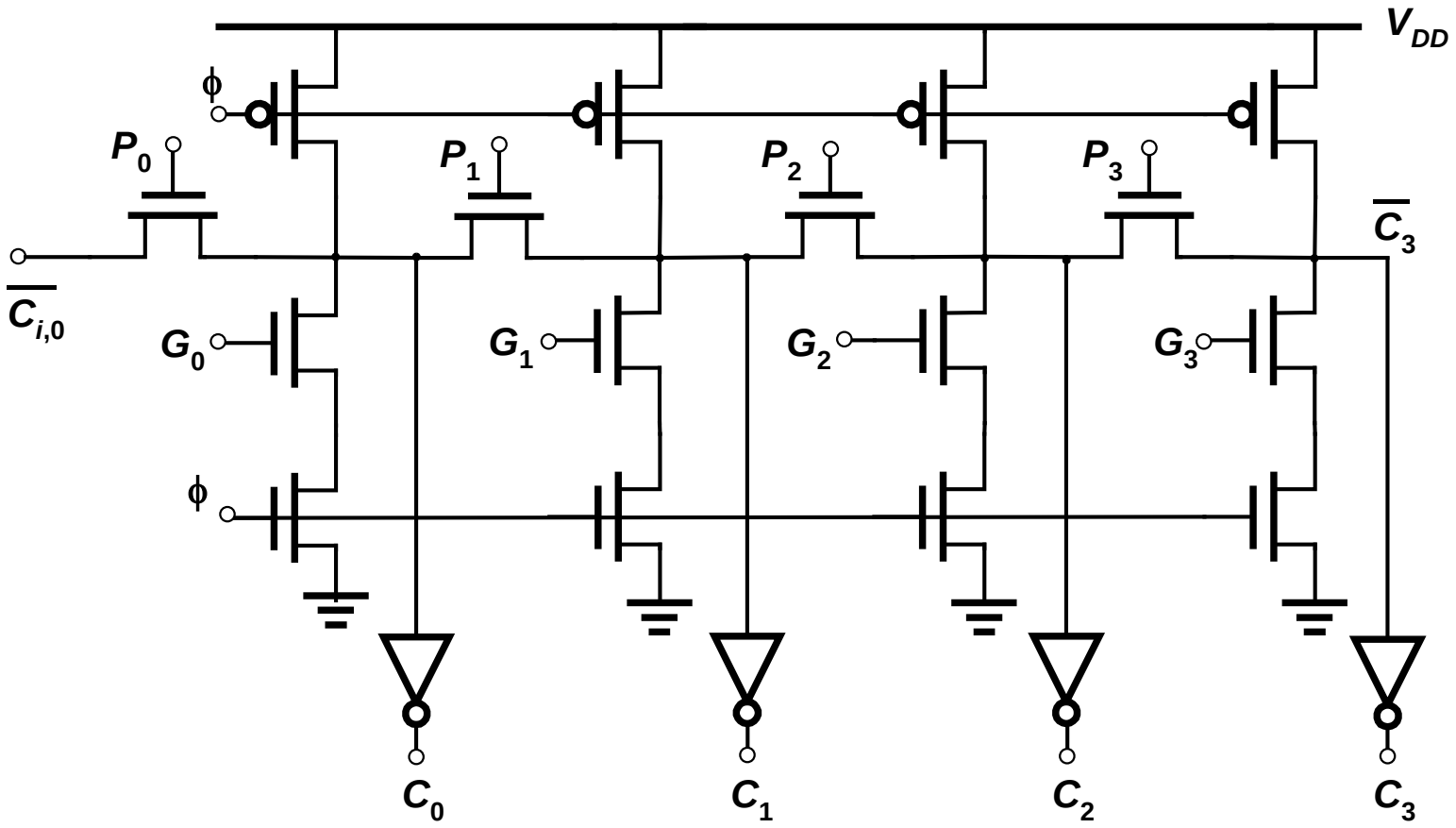
Manchester Carry Chain





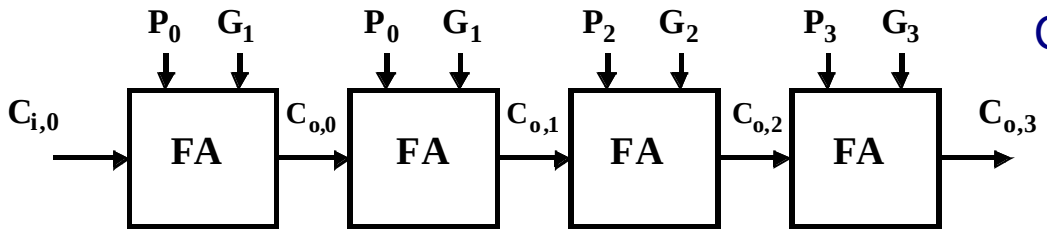
- *The propagate path passes C_i to C_o output if the propagate signal is true*
- *If the propagate condition is not satisfied, the output is either pulled low by the D_i signal or pulled up by* $\overline{G_i}$

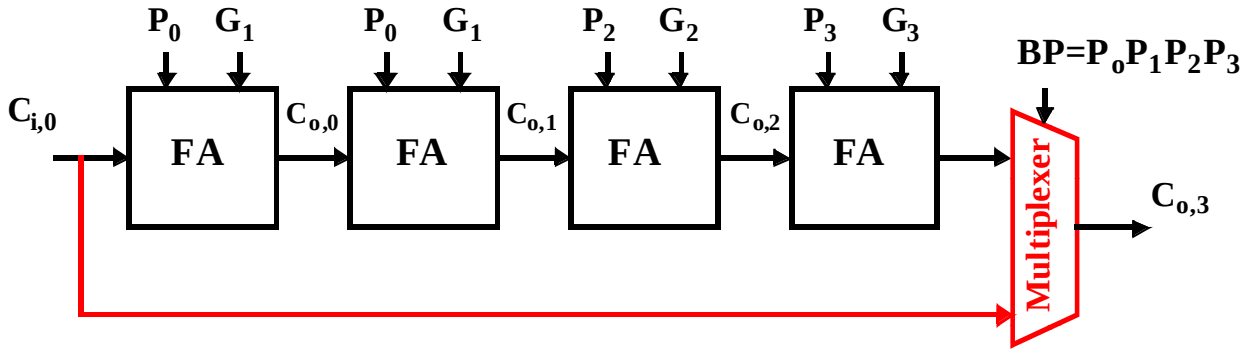
Manchester Carry Chain



Carry-Bypass Adder

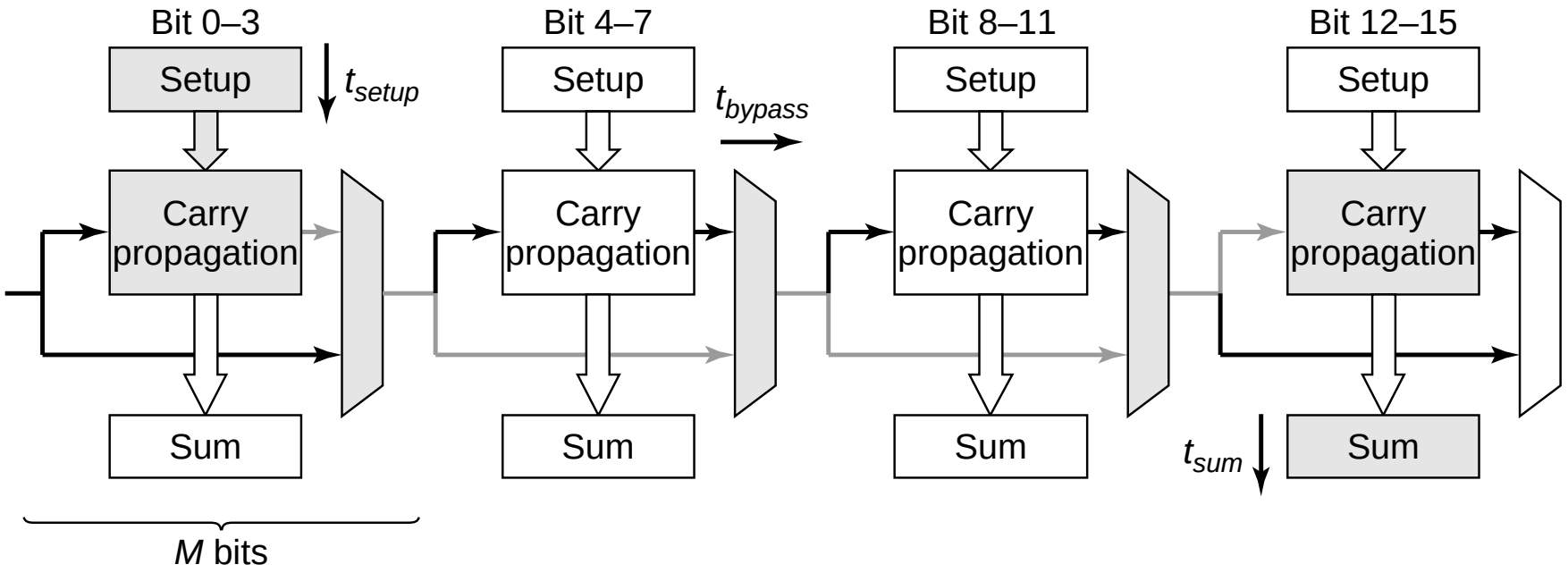
Also called
Carry-Skip





Idea: If (P0 and P1 and P2 and P3 = 1) then $C_{o3} = C_0$, else “kill” or “generate”.

Carry-Bypass Adder (cont.)



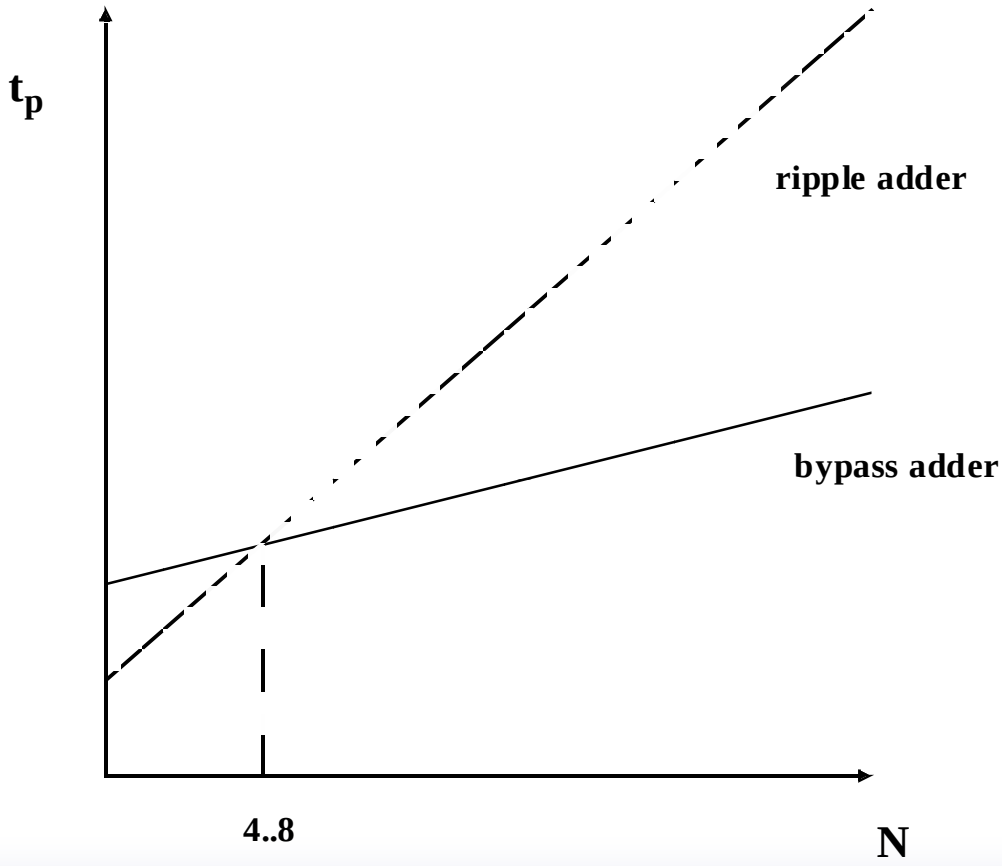
Carry-Bypass Adder (cont.)

- Total adder is divided into (N/M) length bypass stages, each of which contains M bits.

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

- t_{setup}= Fixed overhead time to create the generate and propagate signals
- t_{carry}= Propagation delay through a single bit. The worst case carry propagation delay through a single stage of M bits is approximately M times larger.
- t_{bypass}=the propagation delay through the bypass multiplexer of a single stage.
- t_{sum}= the time to generate the sum of the final stage.

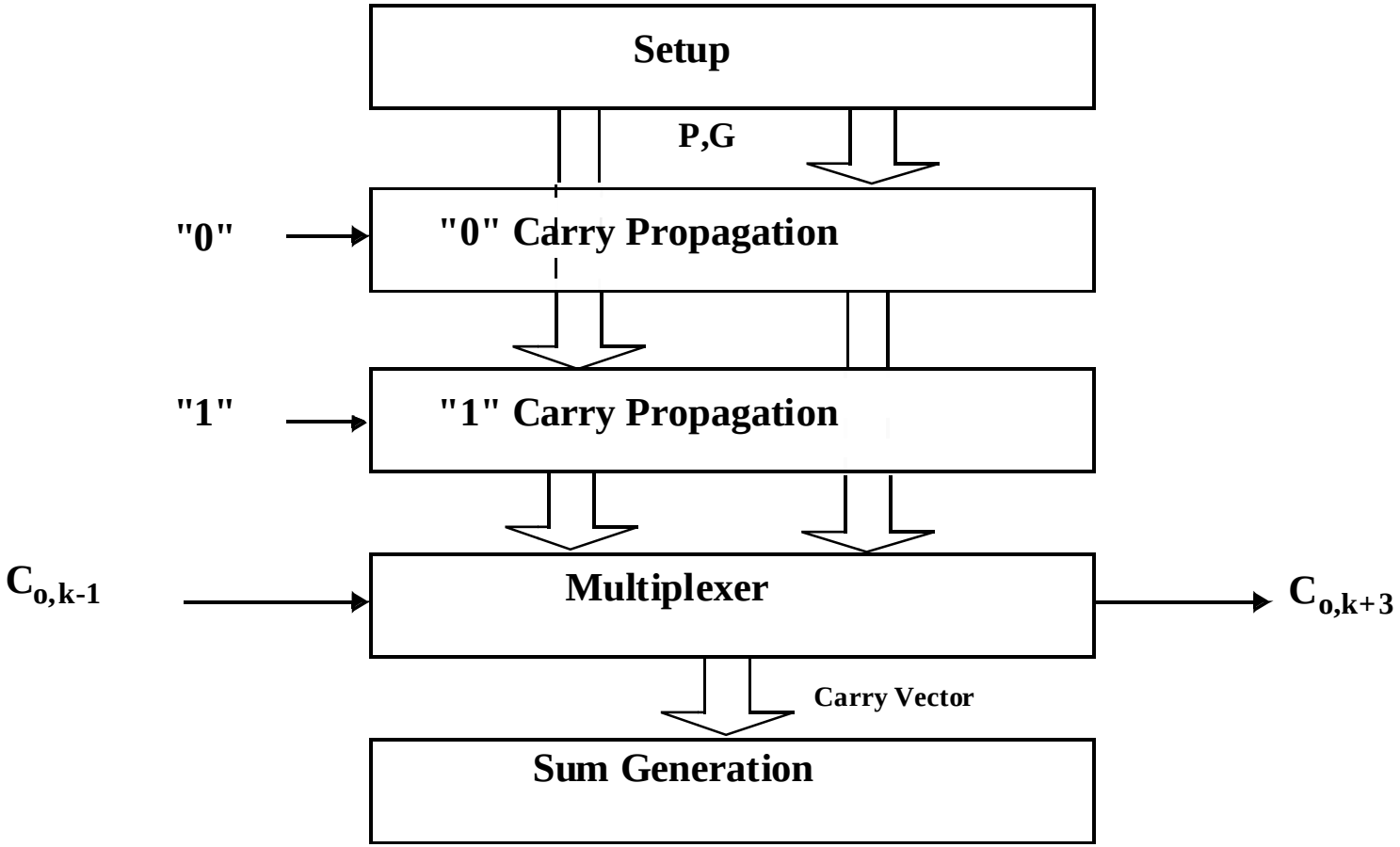
Carry Ripple versus Carry Bypass



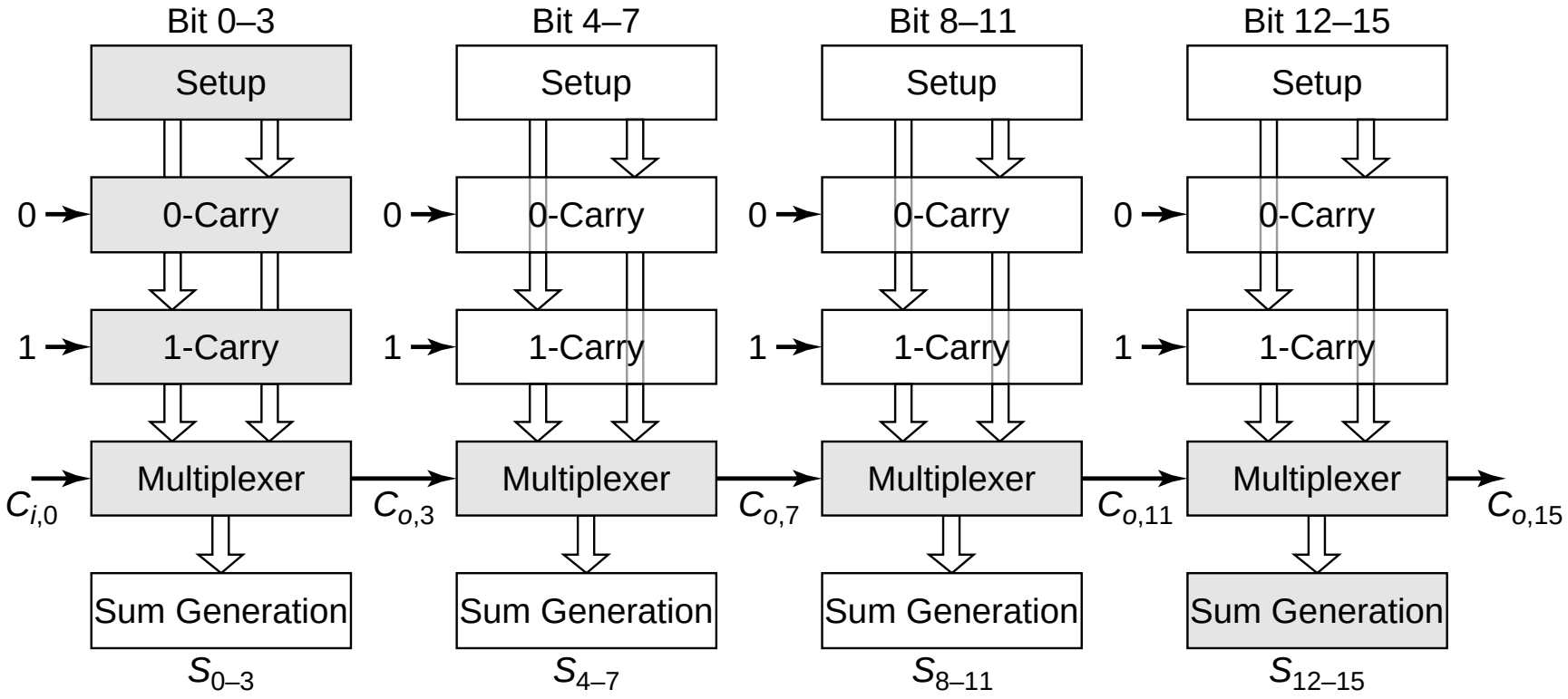
Linear Carry-Select Adder

- Anticipate both possible values of the carry input and evaluate the result for both possibilities in advance.
- Once the real value of the incoming carry is known, the correct result is selected with a multiplexer stage.

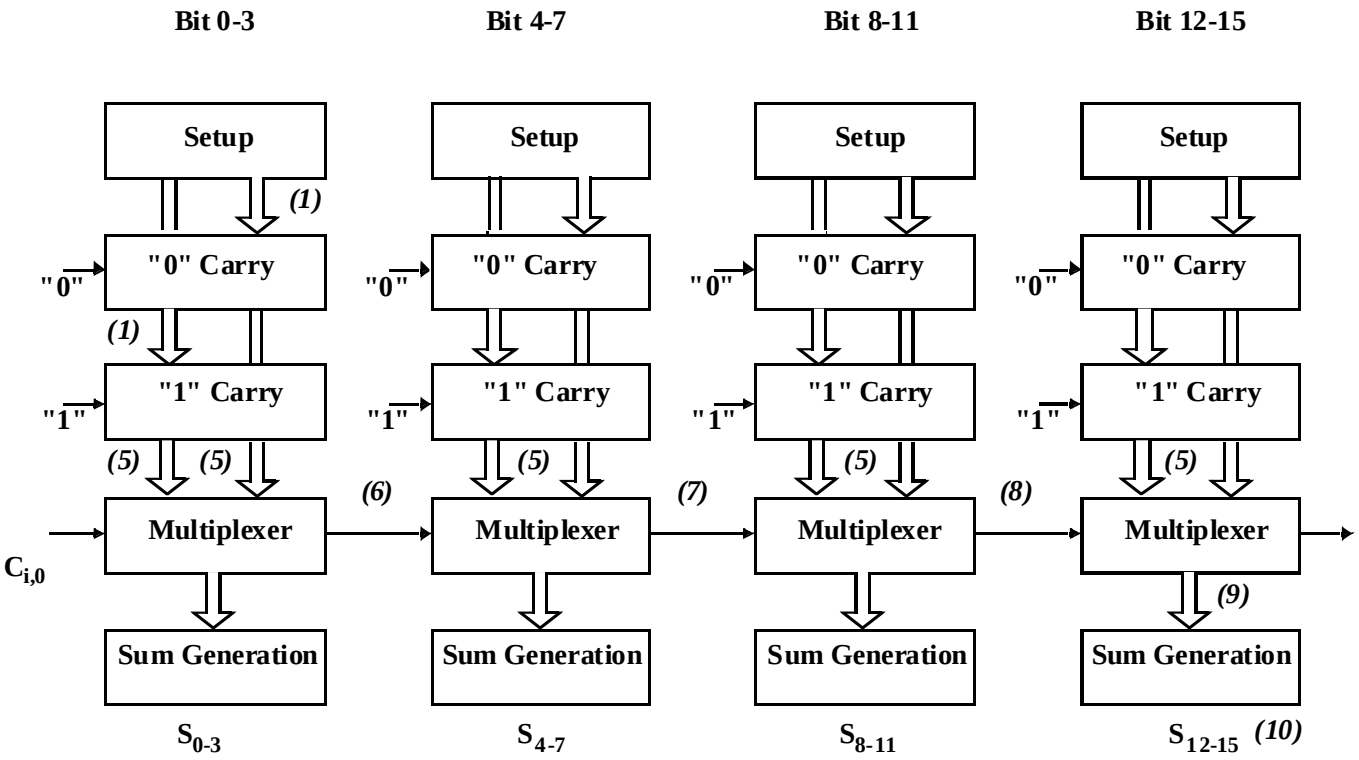
Carry-Select Adder



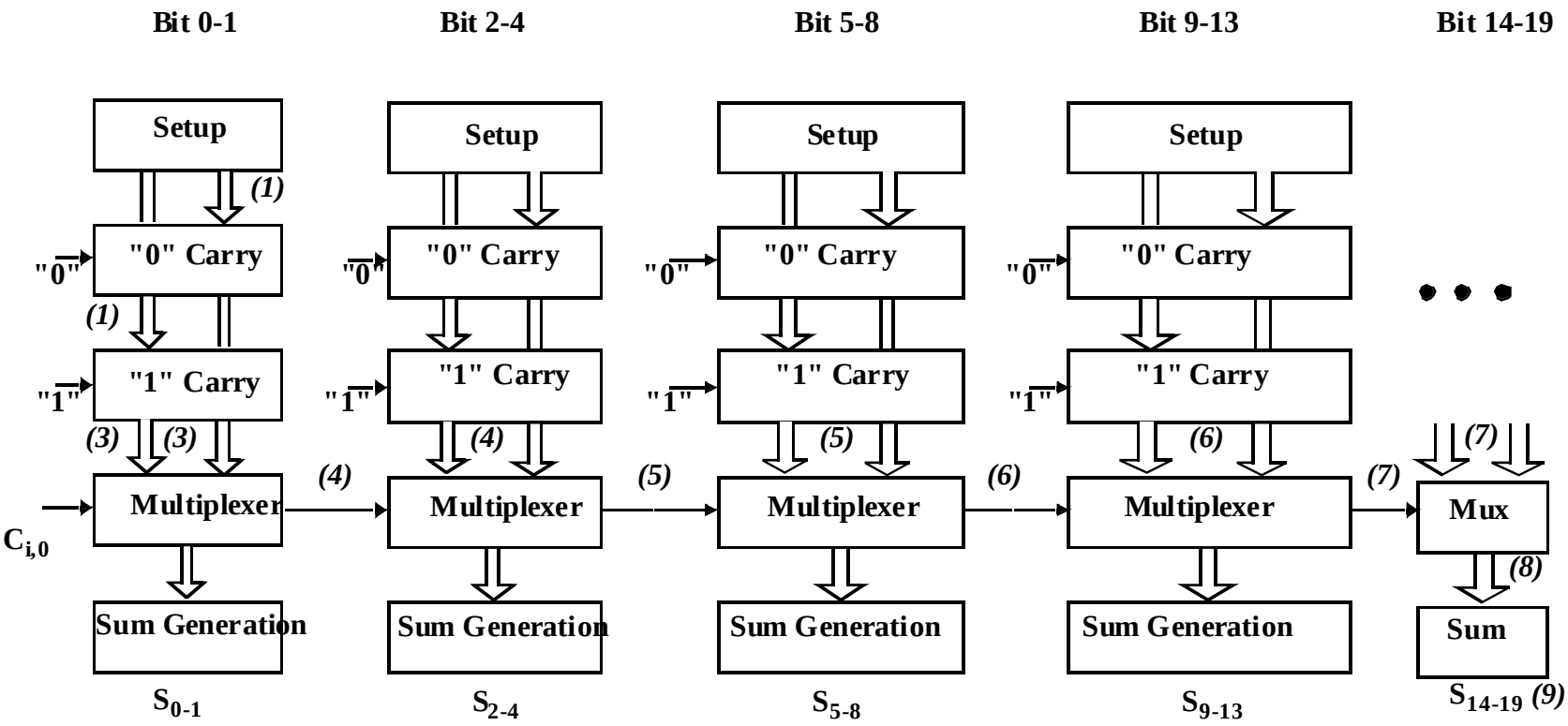
Carry Select Adder: Critical Path



Linear Carry Select

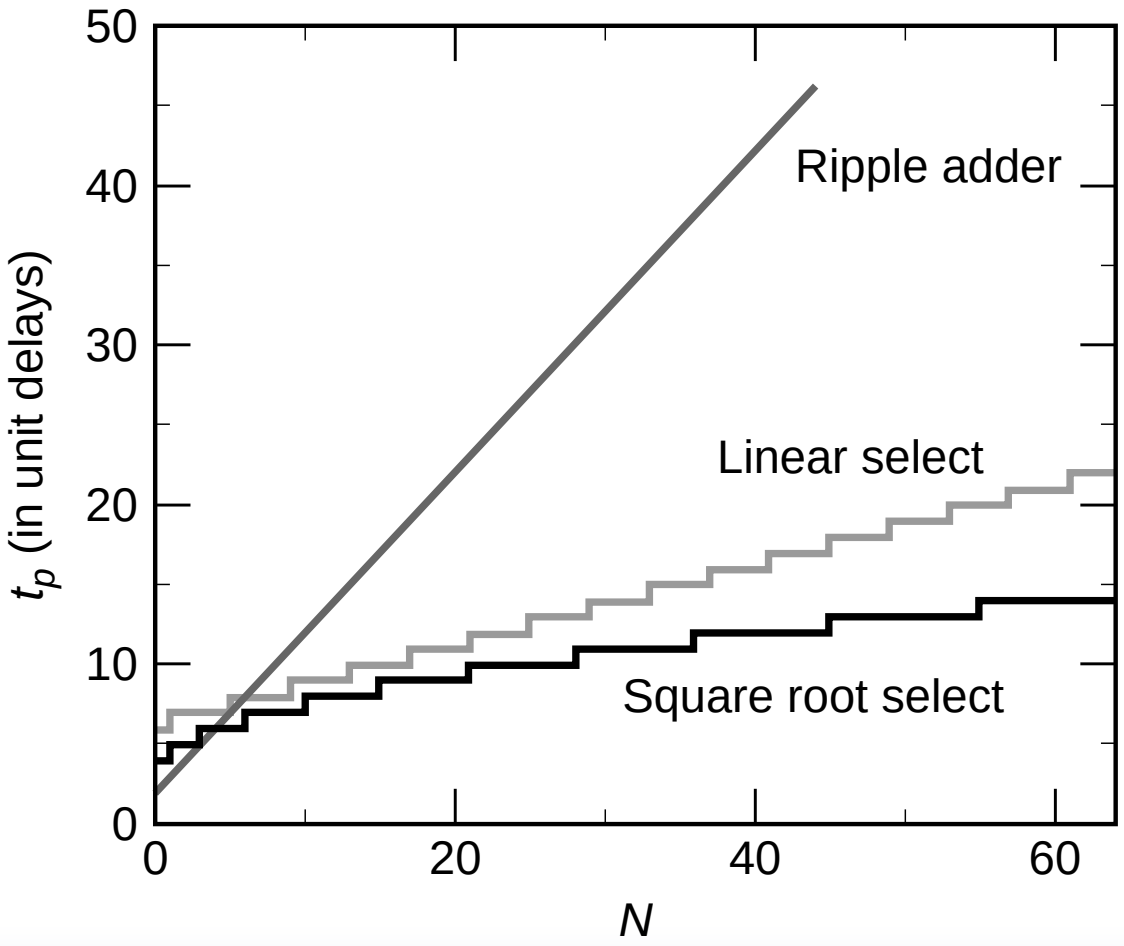


Square Root Carry Select

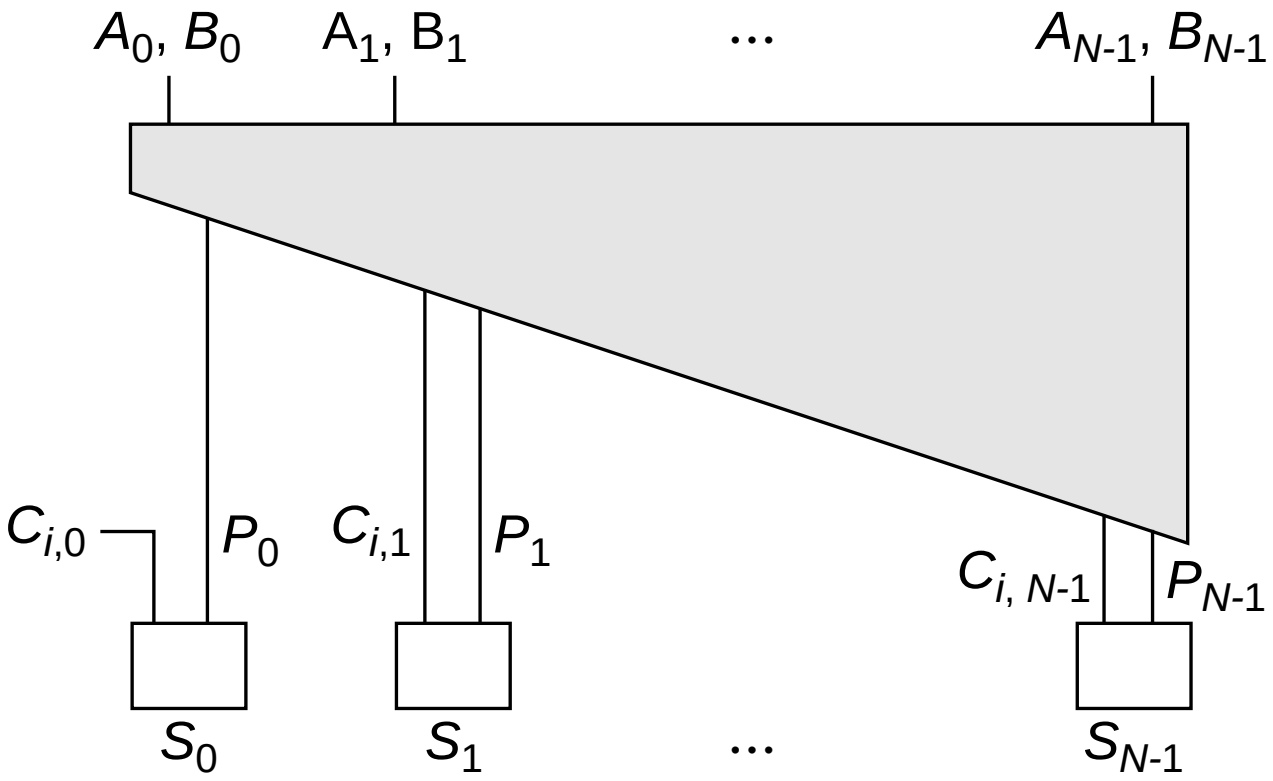


$$t_{add} = t_{setup} + P \cdot t_{carry} + (\sqrt{2N})t_{mux} + t_{sum}$$

Adder Delays - Comparison



LookAhead - Basic Idea



$$C_{o,k} = f(A_k, B_k, C_{o,k-1}) = G_k + P_k C_{o,k-1}$$

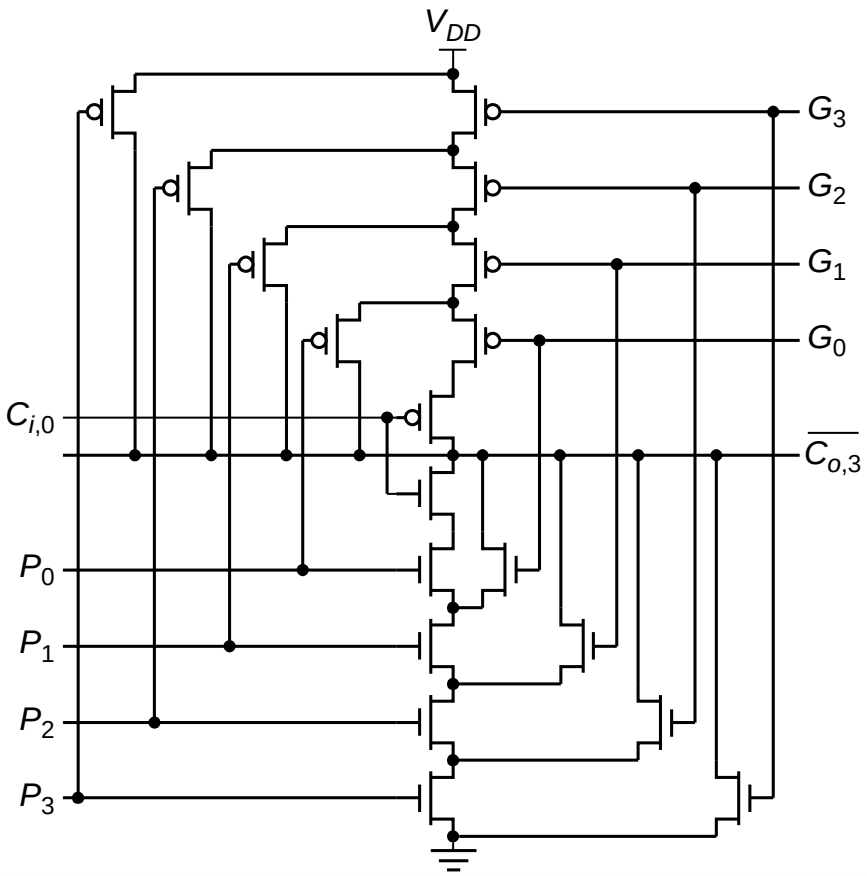
Look-Ahead: Topology

Expanding Lookahead equations:

$$C_{o,k} = G_k + P_k(G_{k-1} + P_{k-1}C_{o,k-2})$$

All the way:

$$C_{o,k} = G_k + P_k(G_{k-1} + P_{k-1}(\ldots + P_1(G_0 + P_0C_{i,0})))$$



Look-Ahead: Topology

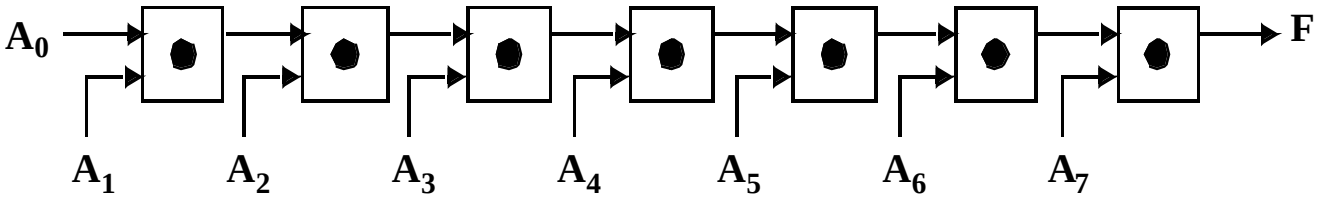
Expanding Lookahead equations:

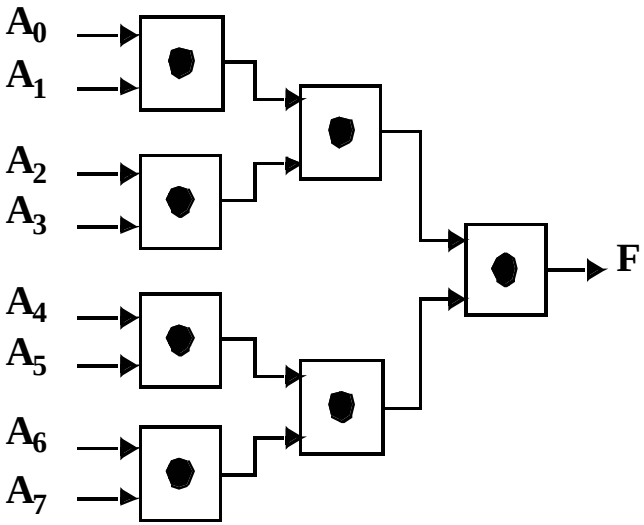
$$C_{o,k} = G_k + P_k(G_{k-1} + P_{k-1}C_{o,k-2})$$

All the way:

$$C_{o,k} = G_k + P_k(G_{k-1} + P_{k-1}(\ldots + P_1(G_0 + P_0C_{i,0})))$$

Logarithmic Look-Ahead Adder





$t_p \sim N$

$t_p \sim \log_2(N)$

Carry Lookahead Trees

$$C_{o,0} = G_0 + P_0 C_{i,0}$$

$$C_{o,1} = G_1 + P_1 G_0 + P_1 P_0 C_{i,0}$$

$$C_{o,2} = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{i,0}$$

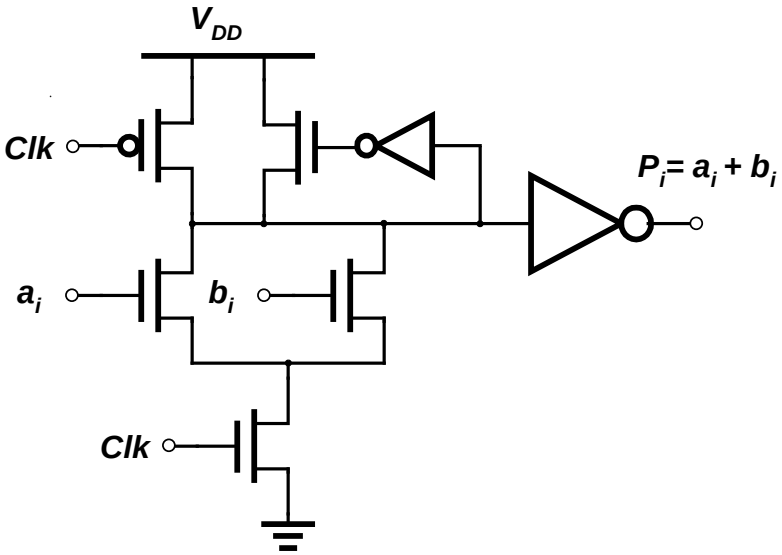
$$= (G_2 + P_2 G_1) + (P_2 P_1)(G_0 + P_0 C_{i,0}) = G_{2:1} + P_{2:1} C_{o,0}$$

Can continue building the tree hierarchically.

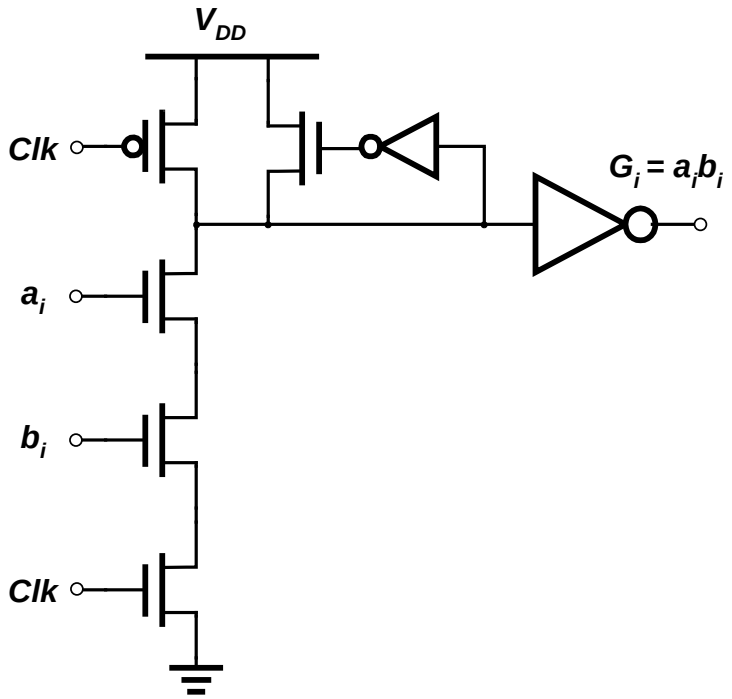
Carry Lookahead Trees

- The carry-propagation process is decomposed into subgroups of two bits
  - $G_{i:j}$ and $P_{i:j}$ denote the generate and propagate functions, respectively, for a group of bits for positions I to j
  - Block generate and propagate signals.
  - $G_{i:j}$ equals 1 if the group generates a carry, independent of the incoming carry
  - The block propagate $P_{i:j}$ is true if an incoming carry propagates through the complete group.

Example: Domino Adder

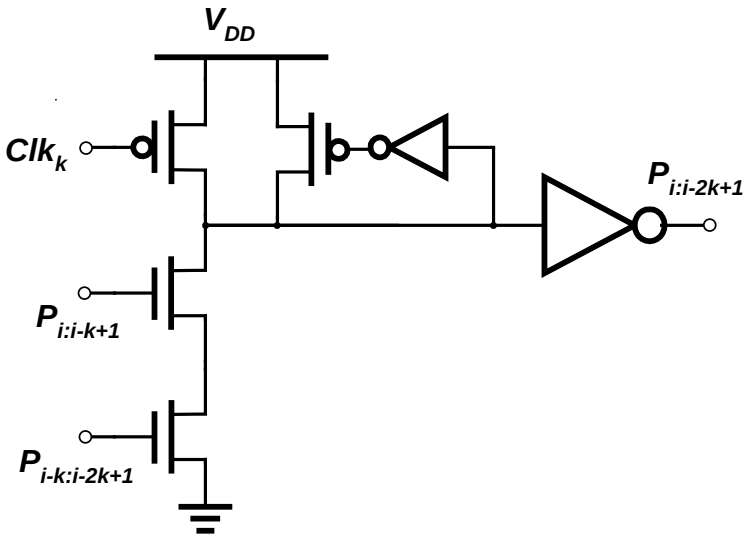


Propagate

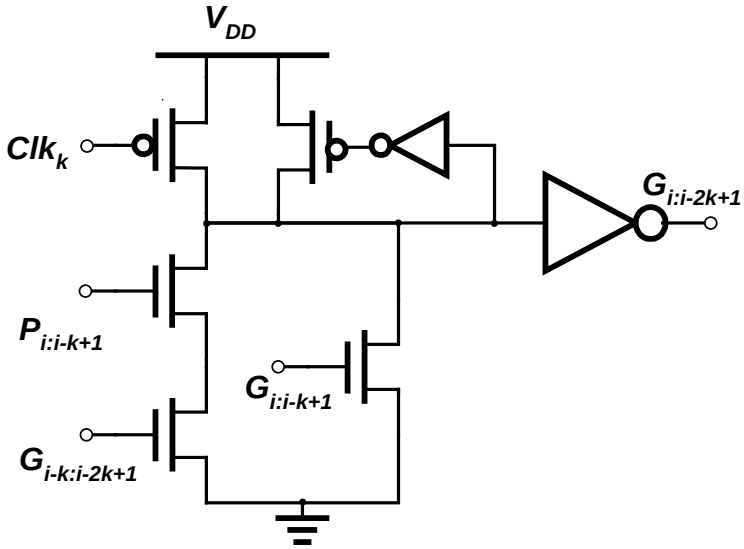


Generate

Example: Domino Adder



Propagate



Generate

Example: Domino Sum