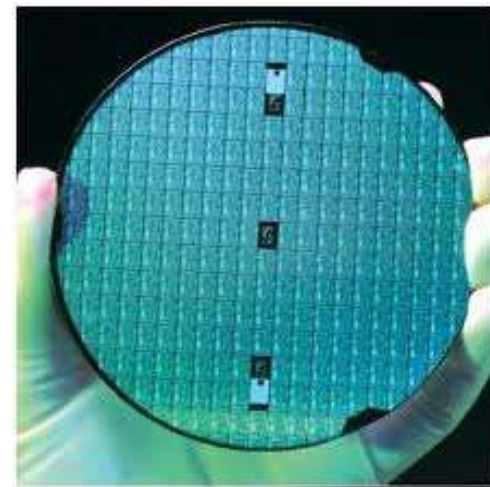
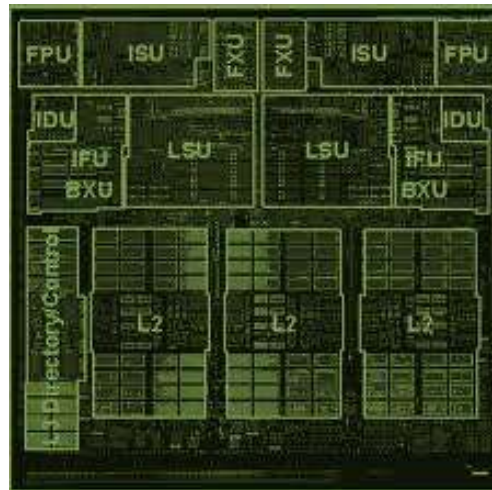
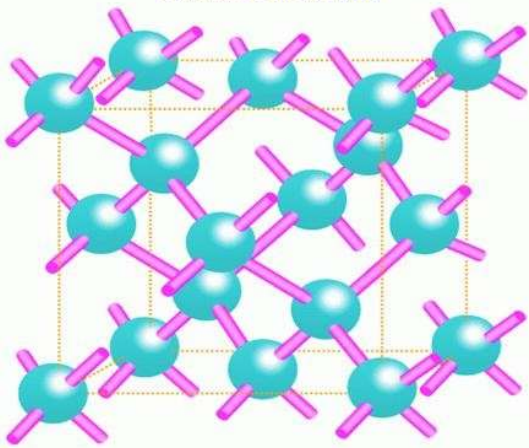


Structure of silicon crystal



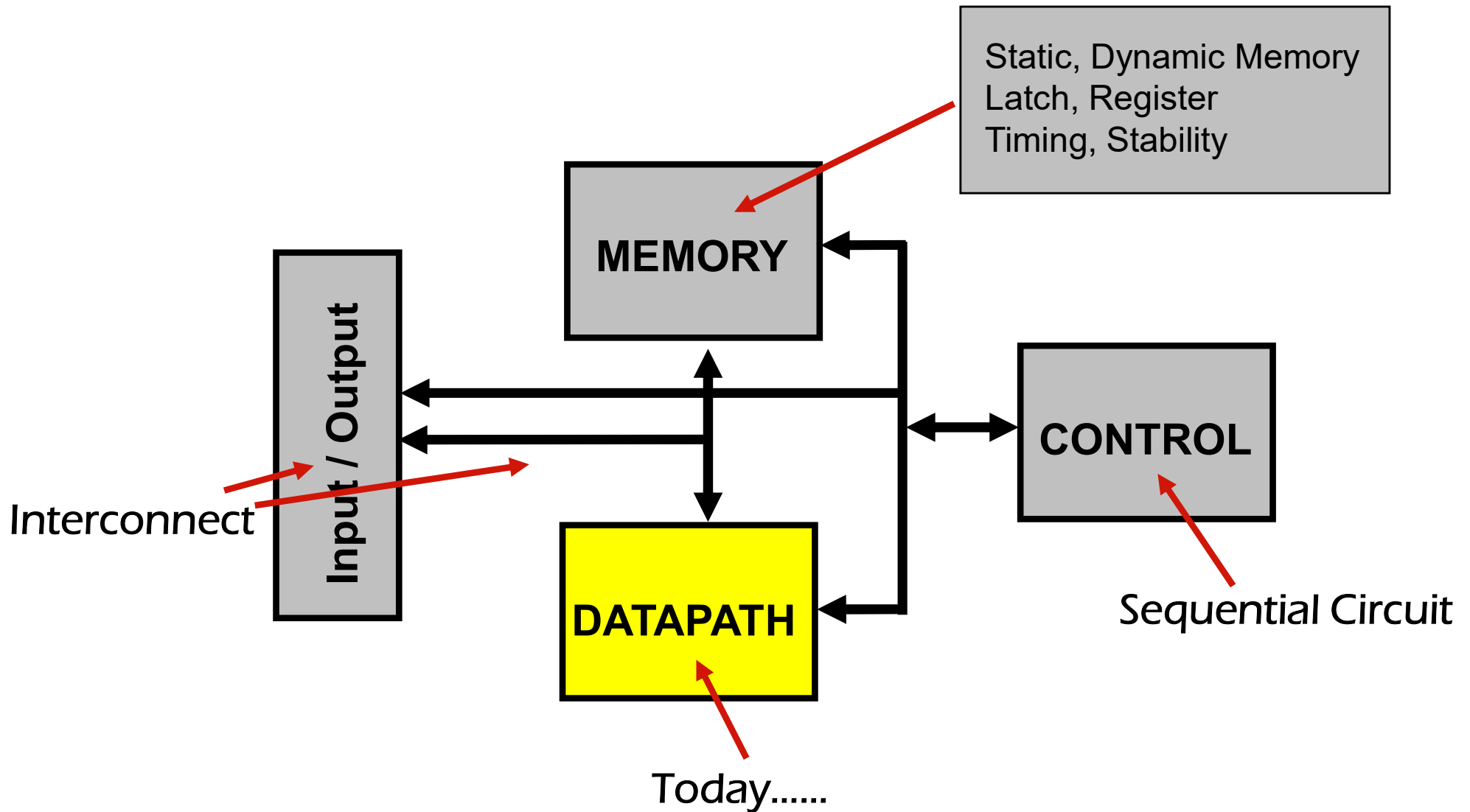
ECE 122A

VLSI Principles

Lecture 12

Prof. Kaustav Banerjee
Electrical and Computer Engineering
University of California, Santa Barbara
E-mail: kaustav@ece.ucsb.edu

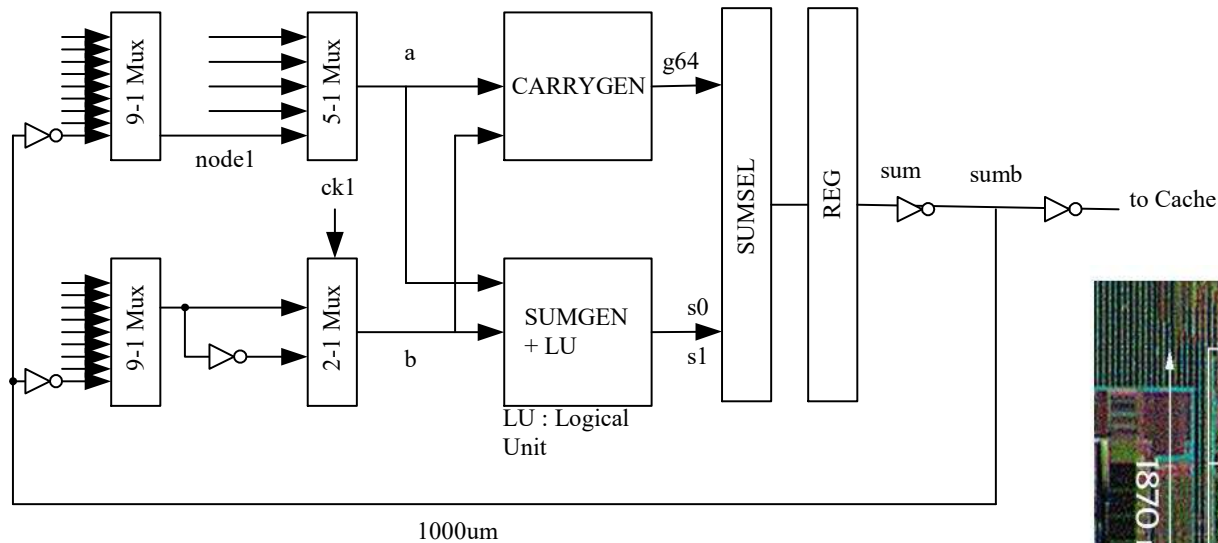
A Generic Digital Processor



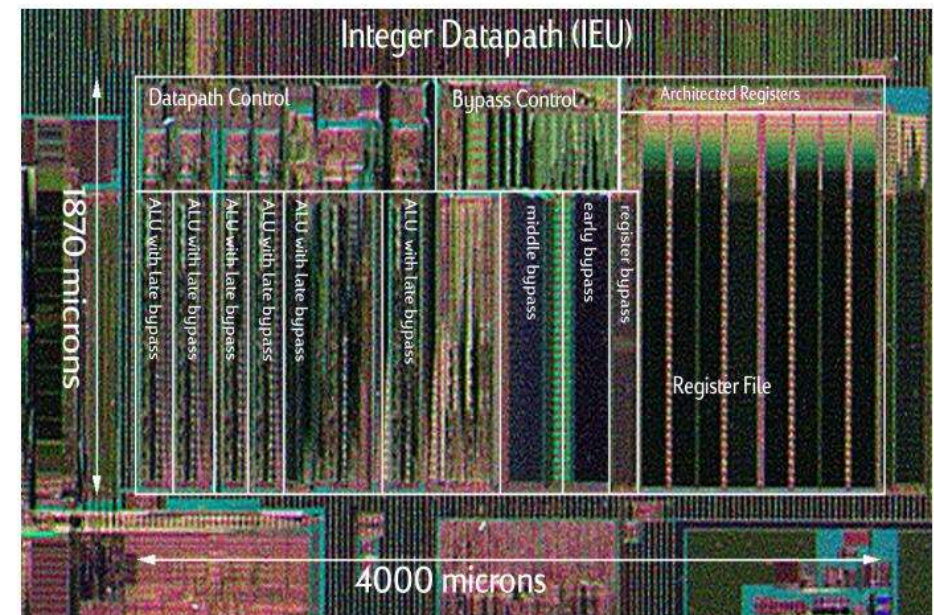
DATAPATH

- The core of a digital Processor
where all computations are performed
- Datapath consists
 - **Logic Blocks**
 - Combinational Logic Functions (AND, OR, XOR....)
 - **Arithmetic Blocks**
 - Addition
 - Multiplication
 - Comparison
 - Shift

An Intel Microprocessor



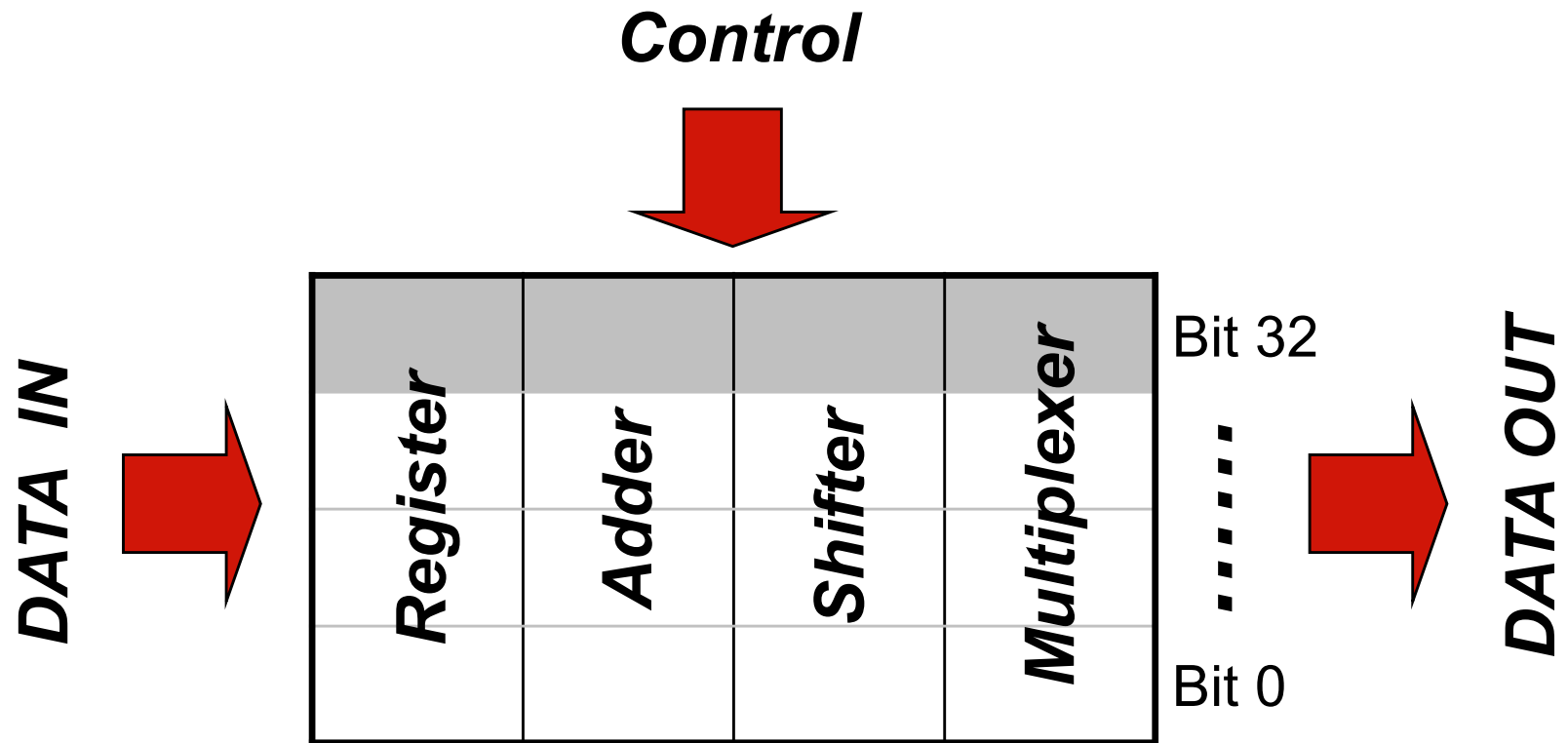
Intel Itanium® Integer Datapath



Itanium has 6 integer execution units like this

Fetzer, Orton, ISSCC'02

Bit-Sliced Design



Design a single bit datapath and repeat for all bits

Adders

Design an Adder

- Fundamental Arithmetic Building Block
- Performance
 - Logic Level Optimization
 - Optimize Boolean Functions
 - Carry Lookahead
 - Circuit Level Optimization
 - Transistor Sizing
- Power Consumption

Fundamental Logic of Adders

□ Truth table of a full adder

- Sum: $S = A \oplus B \oplus C_{in}$

- Carry out:

$$C_{out} = (A \cdot B) + (C_{in} \cdot (A \oplus B))$$

□ PGK (independent of C_{in})

- Generate: $G = A \cdot B$

- Propagate: $P = A \oplus B$

- Kill: $K = \sim A \cdot \sim B$

$$S = P \oplus C_{in}, C_{out} = G + C_{in} \cdot P$$

A	B	C	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Adder Overview

□ Different adders covered in the slides:

➤ **Single-bit adder**

❖ *Static*

- *Static CMOS*
- *Mirror design*
- *Transmission gate*

❖ *Dynamic*

- *Manchester chain*

➤ **Multi-bit adder**

❖ *Ripple-carry adder*

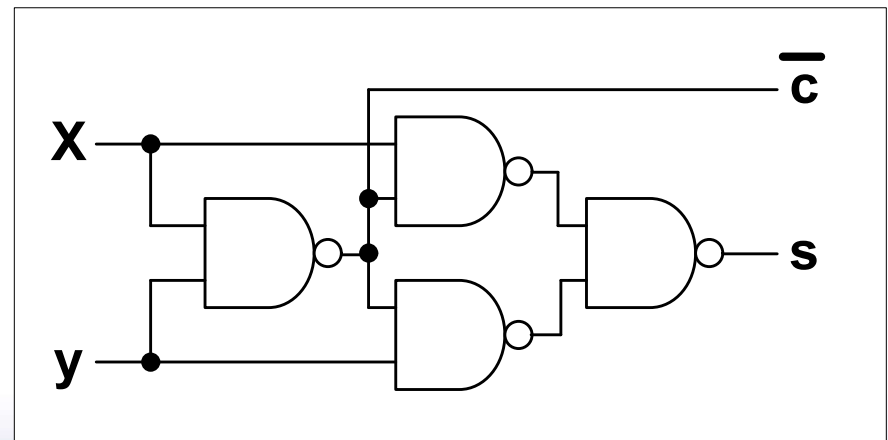
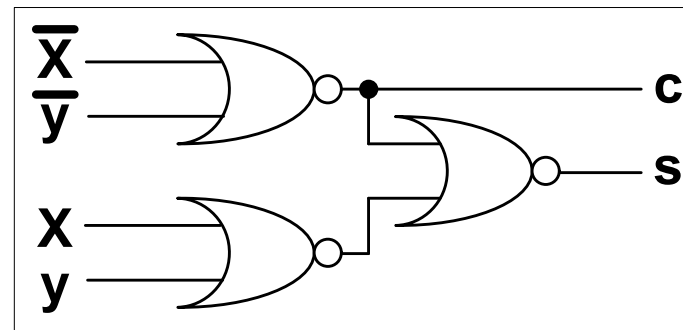
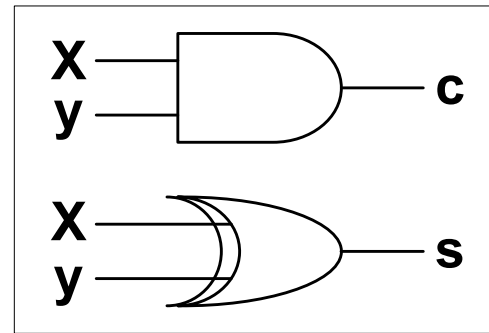
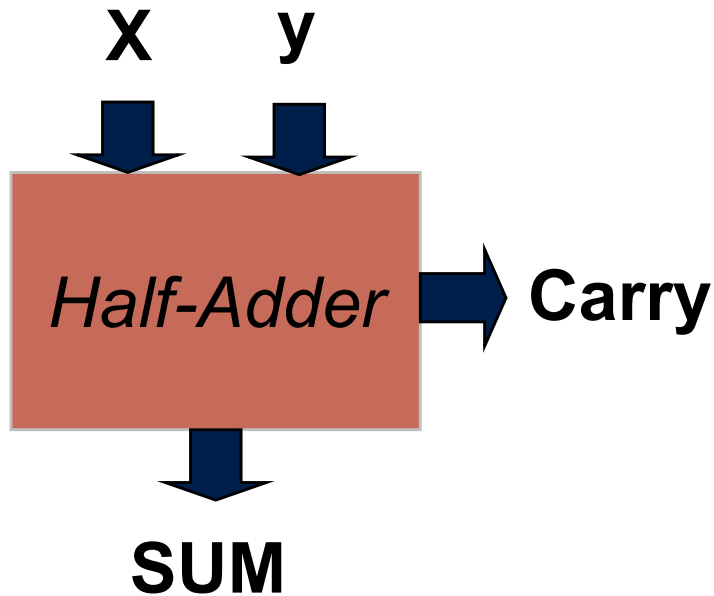
❖ *Carry bypass adder*

- *Linear carry select*
- *Square-root carry select*

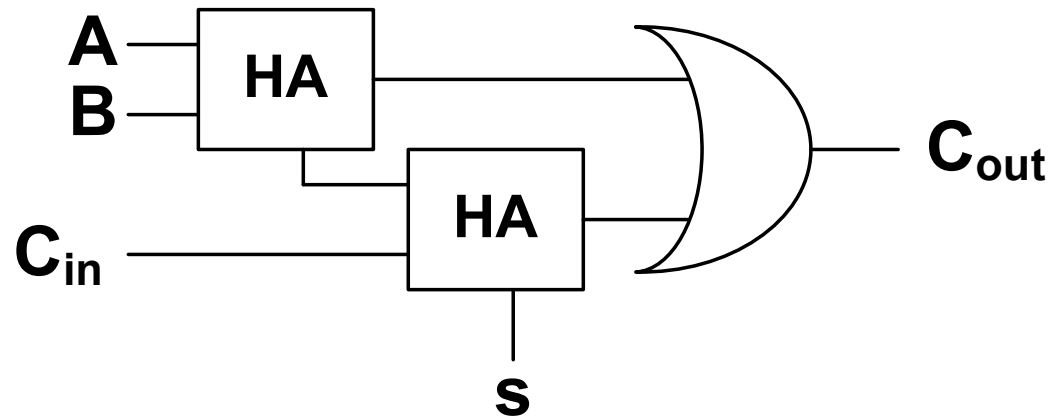
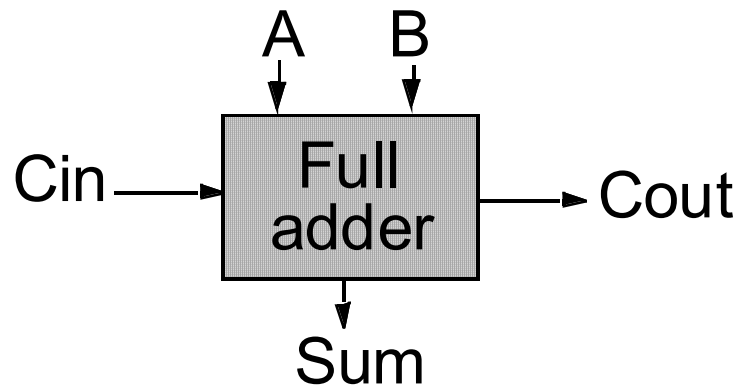
❖ *Carry look-ahead*

Half-Adder Implementations

Half-Adder



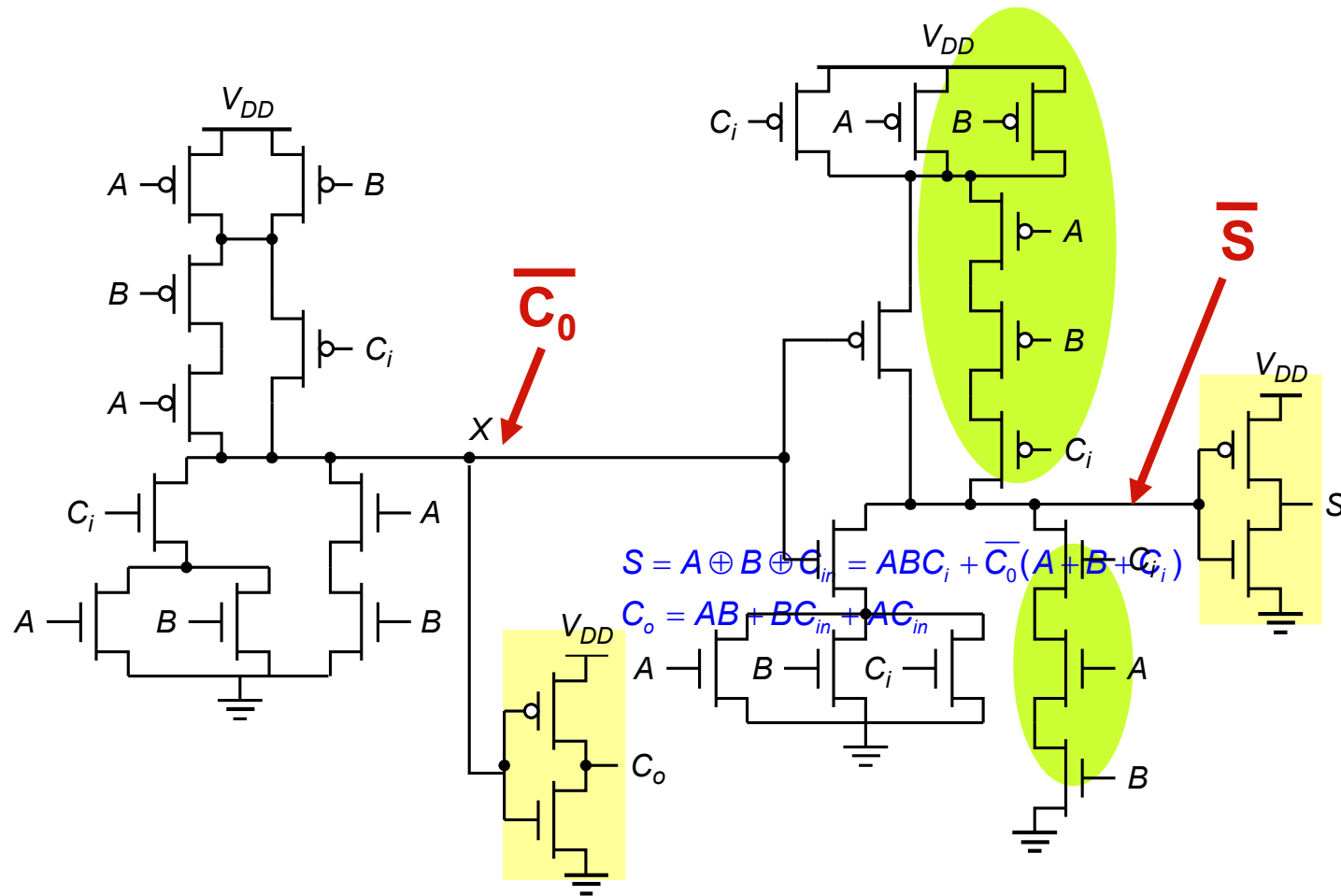
Full-Adder



$$S = A \oplus B \oplus C_{in} = \overline{A}\overline{B}\overline{C}_{in} + \overline{A}B\overline{C}_{in} + A\overline{B}\overline{C}_{in} + ABC_{in}$$

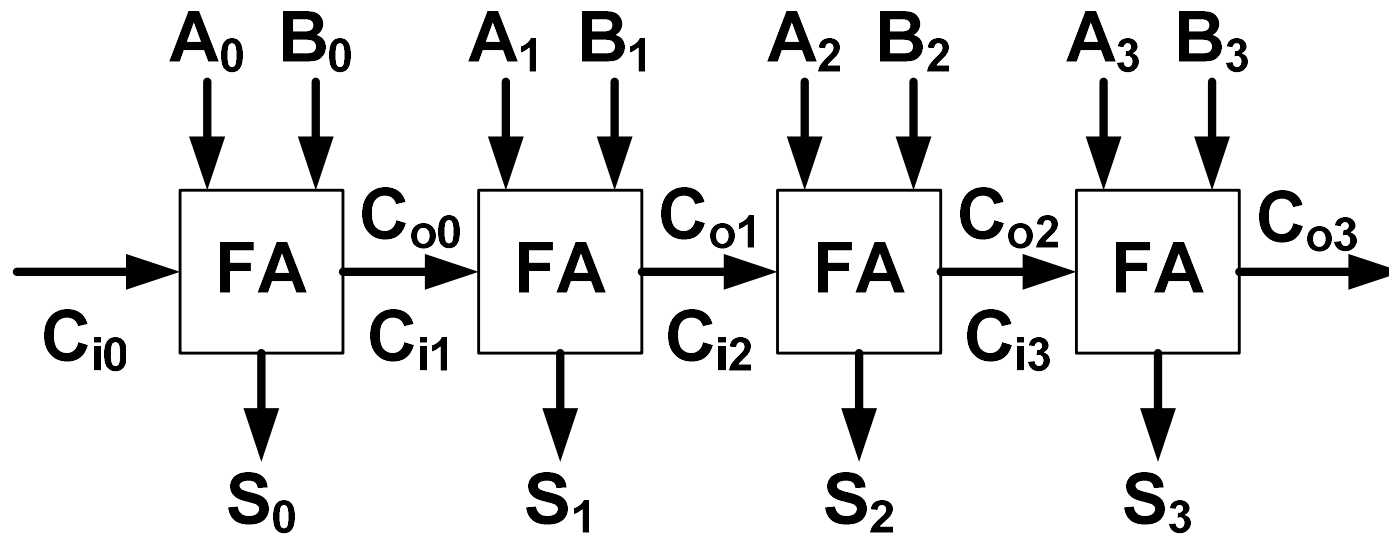
$$C_{out} = AB + BC_{in} + AC_{in}$$

Complimentary Static CMOS Full Adder



28 Transistors

The Ripple-Carry Adder

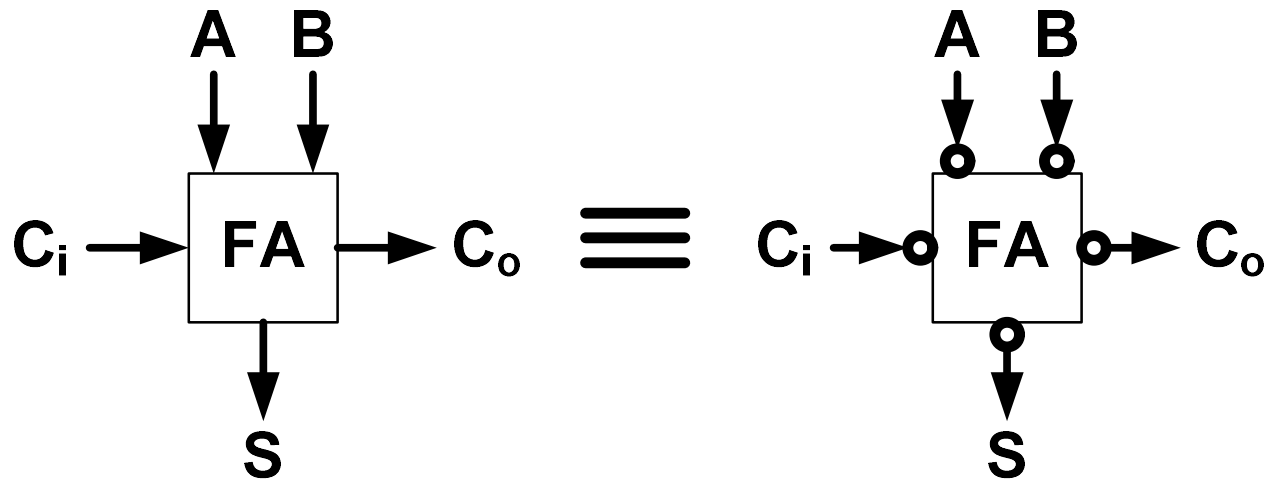


Worst case delay: linear with the number of bits $t_d = O(N)$

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

- ❑ Propagation Delay is proportional to N
- ❑ t_{carry} dominates the propagation delay

Inverting Property

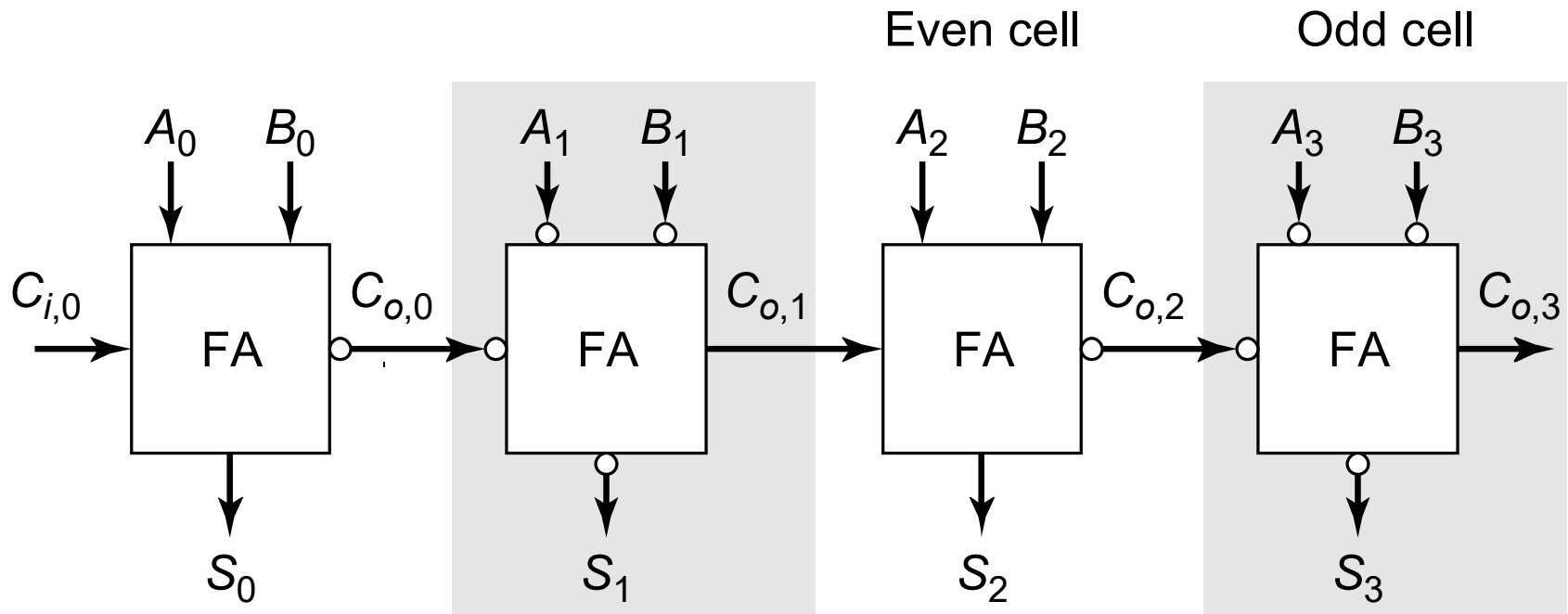


$$\overline{S}(A, B, C_i) = S(\overline{A}, \overline{B}, \overline{C_i})$$

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

Inverting all inputs to FA results in inverted values for all outputs

Minimize Critical Path by Reducing Inverting Stages

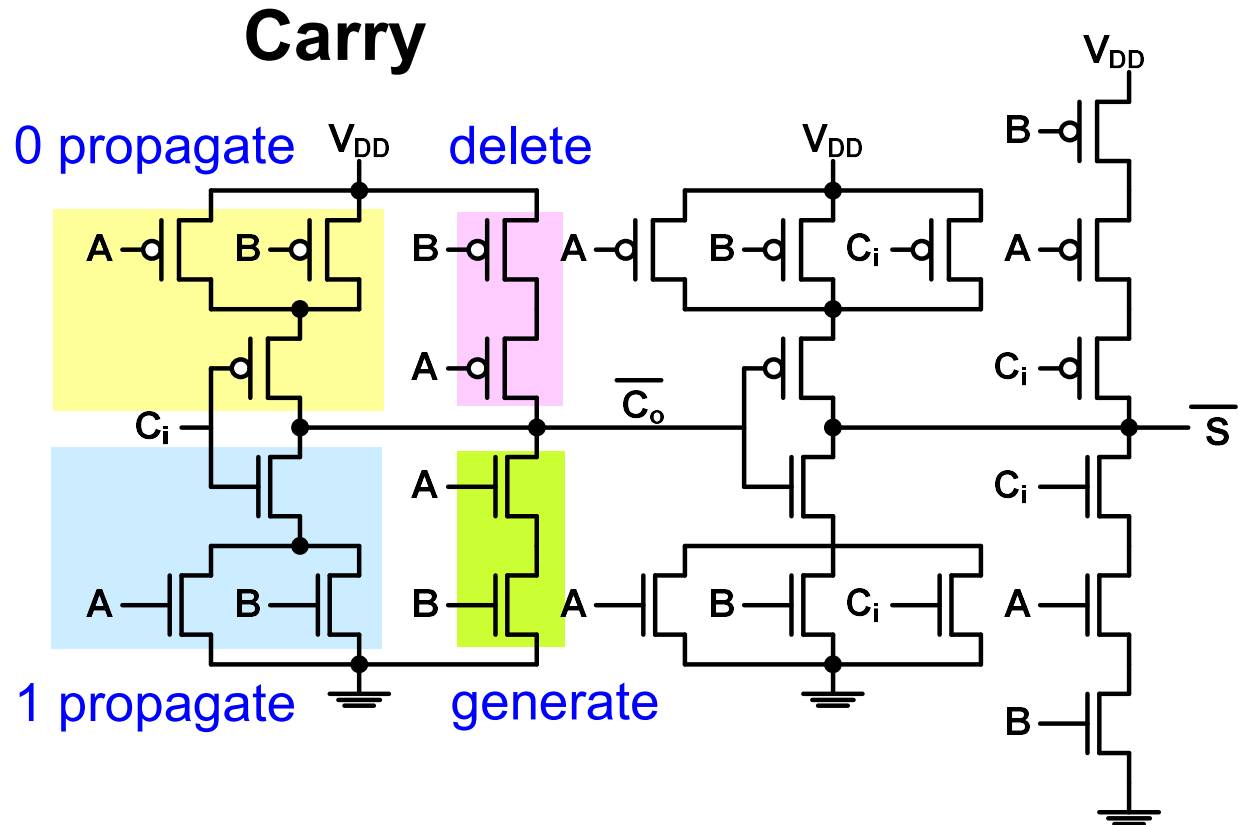


Exploit Inversion Property

Mirror Adder (1)

A	B	C_i	S	C_o	status
0	0	0	0	0	D
0	0	1	1	0	D
0	1	0	1	0	P
0	1	1	0	1	P
1	0	0	1	0	P
1	0	1	0	1	P
1	1	0	0	1	G
1	1	1	1	1	G

Truth Table for Full Adder

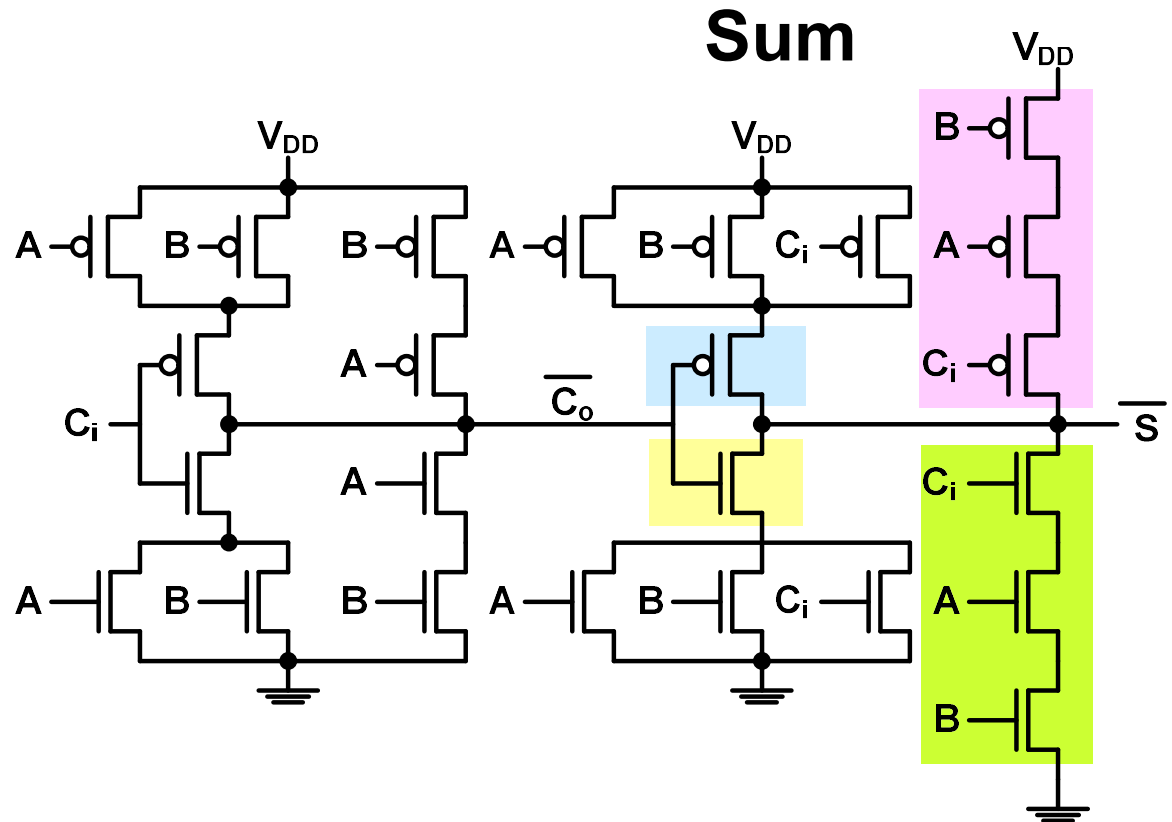


- ❑ Carry-inverting gate is eliminated
- ❑ PDN/PUN networks are not dual

Mirror Adder (2)

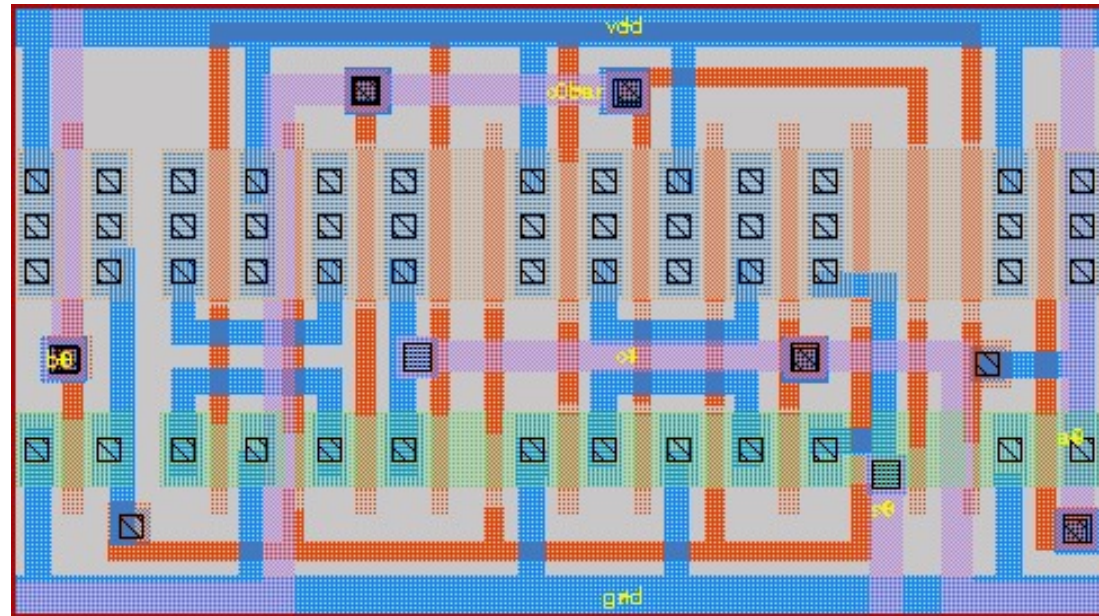
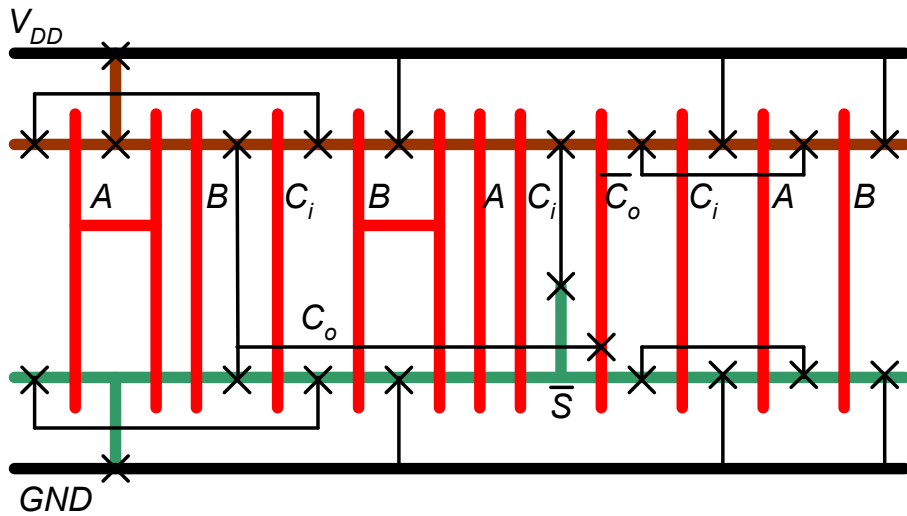
A	B	C_i	S	C_o	status
0	0	0	0	0	D
0	0	1	1	0	D
0	1	0	1	0	P
0	1	1	0	1	P
1	0	0	1	0	P
1	0	1	0	1	P
1	1	0	0	1	G
1	1	1	1	1	G

Truth Table for Full Adder



Mirror Adder Implementation

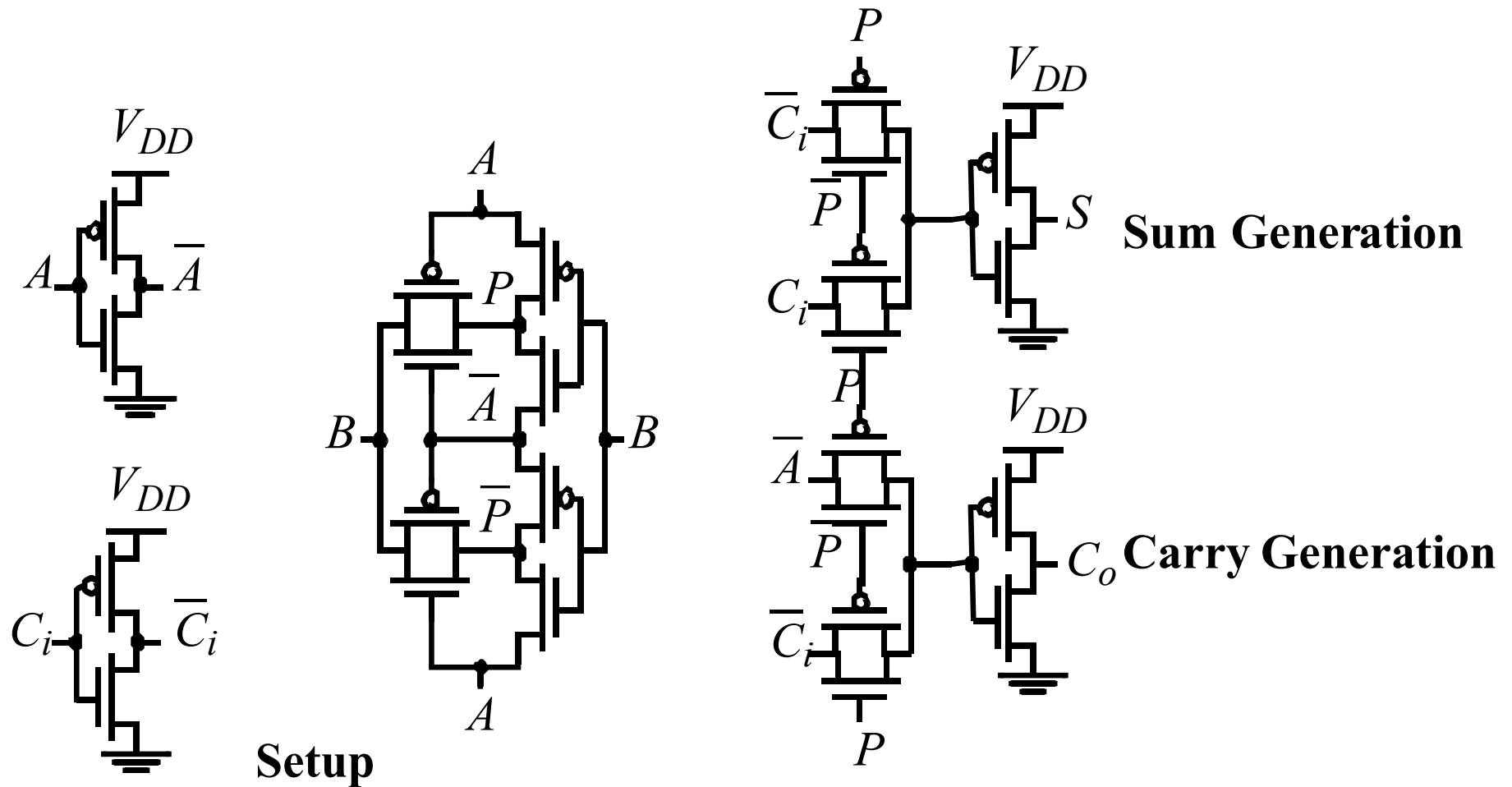
Stick Diagram



Mirror Adder Summary

- ❑ **24** Transistors
- ❑ The NMOS and PMOS chains are completely **symmetrical**
- ❑ A maximum of **2 series transistors** in carry-generation circuitry
- ❑ The transistors connected to C_i should be closest to the output
- ❑ Carry-Stage transistors have to be optimized for speed
- ❑ Sum-Stage transistors can be optimized for area
- ❑ The most critical issue is to minimize the capacitance at C_o
- ❑ C_o is composed of **4 diffusion capacitances**, **2 internal gate capacitances**, and **6 gate capacitances** in the connecting adder cell

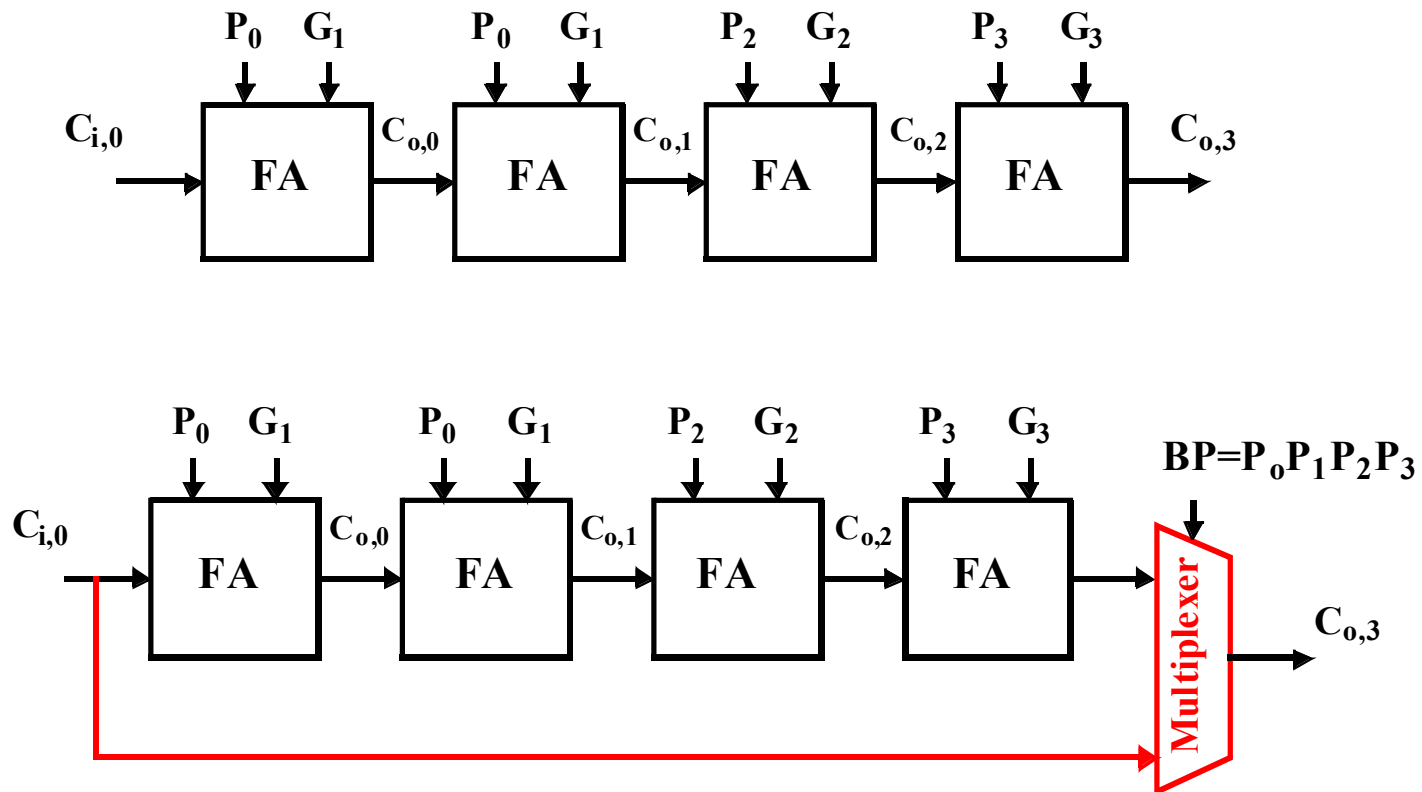
Transmission Gate Full Adder



This implementation has similar sum and carry output delay

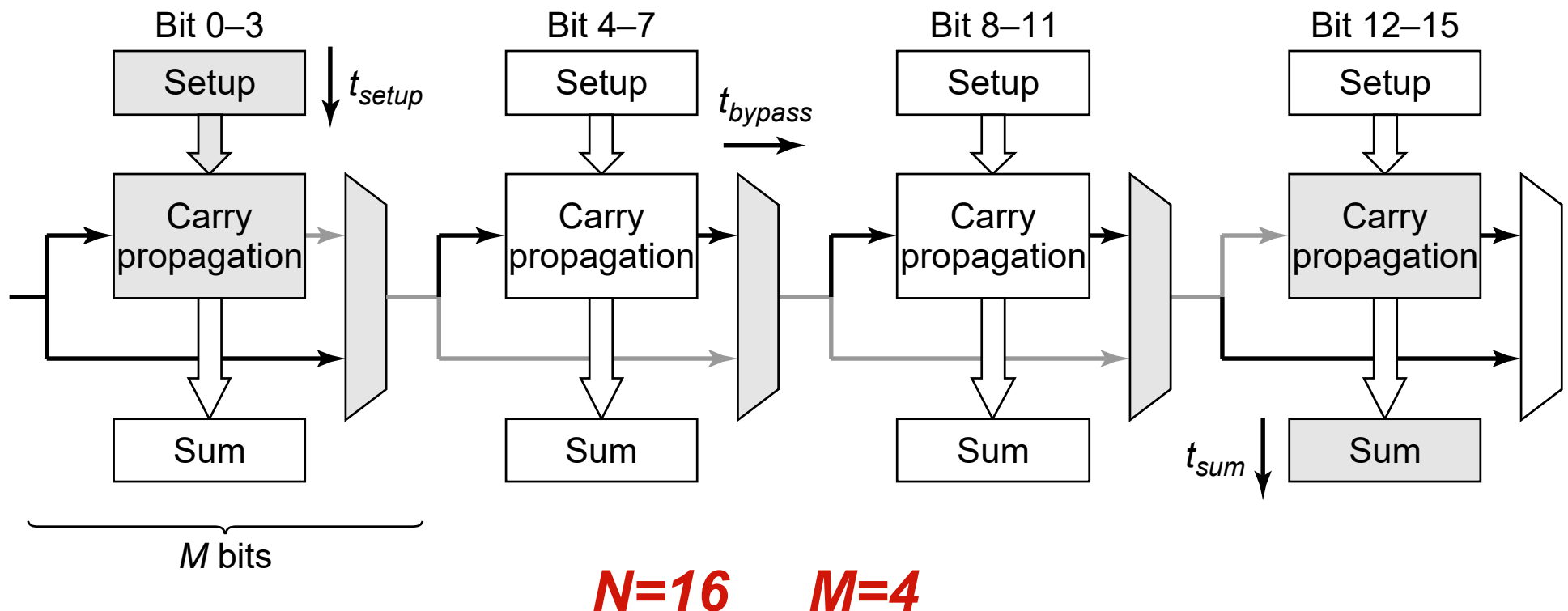
Carry-Bypass Adder

Carry-Skip Adder



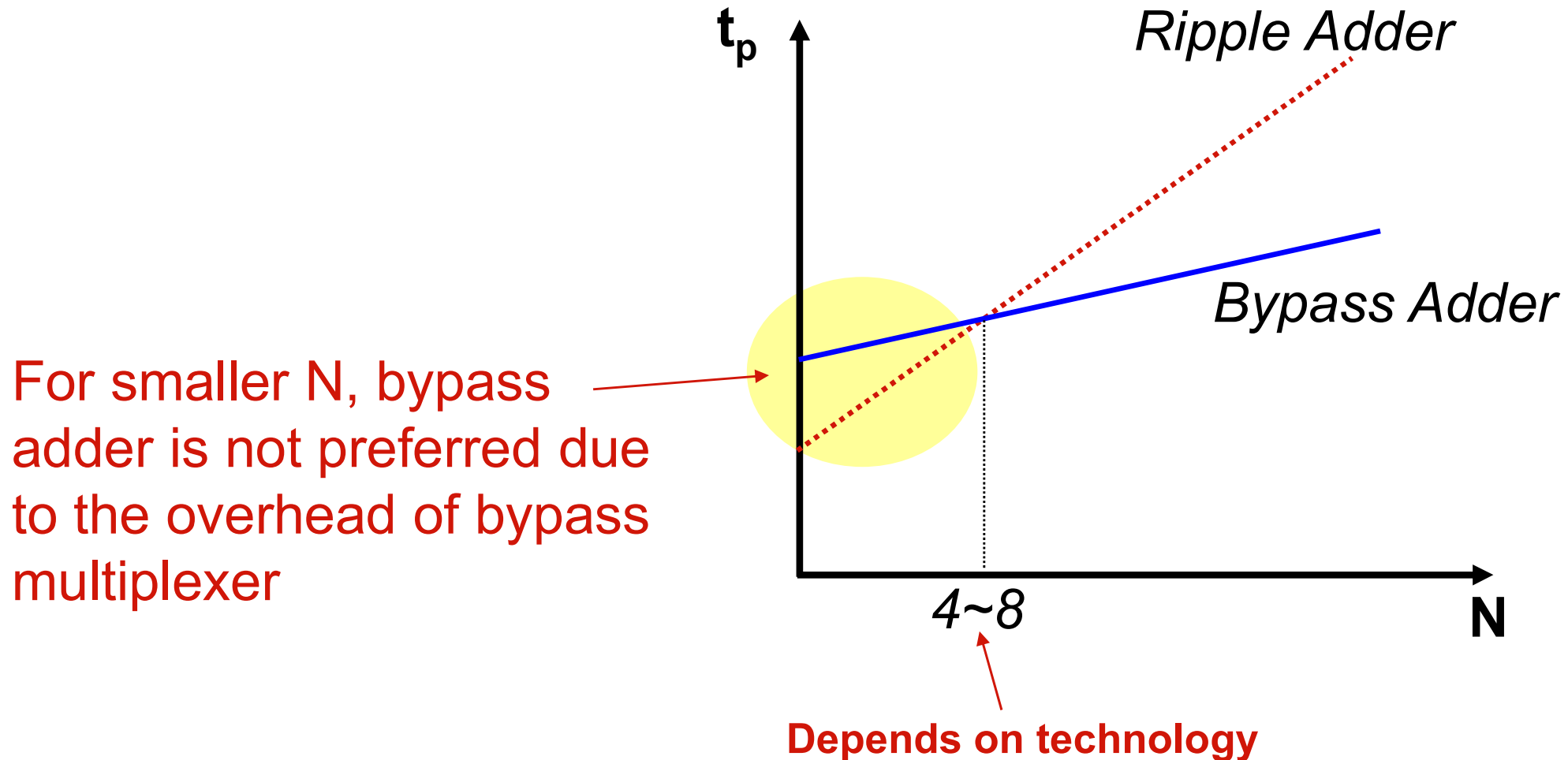
Idea: If (P_0 and P_1 and P_2 and $P_3 = 1$)
then $C_{o3} = C_0$, else “kill” or “generate”.

16-bit Carry-Bypass Adder

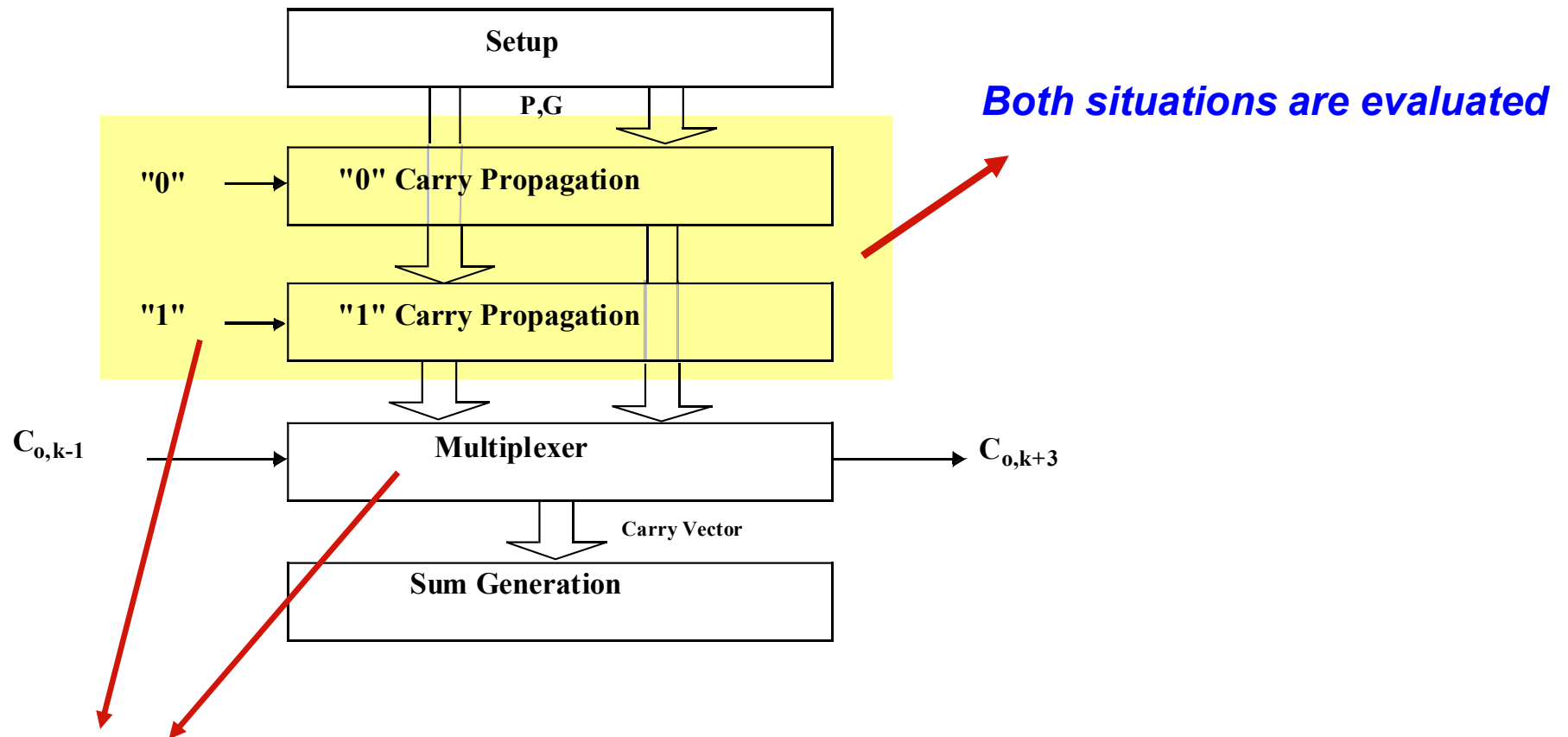


$$t_{adder} = t_{setup} + Mt_{carry} + (N/M-1)t_{bypass} + (M-1)t_{carry} + t_{sum}$$

Carry Ripple versus Carry Bypass

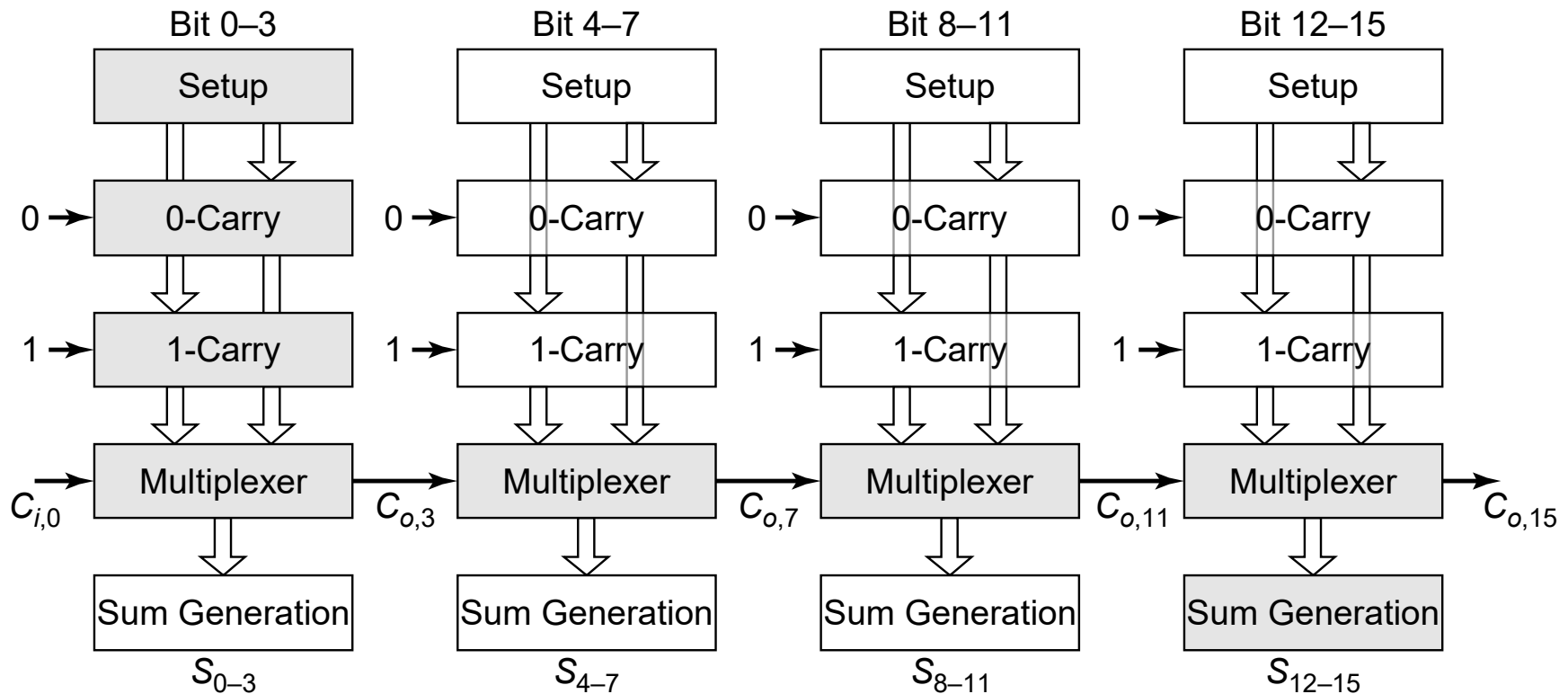


Linear Carry-Select Adder



~ 30 % Hardware overhead

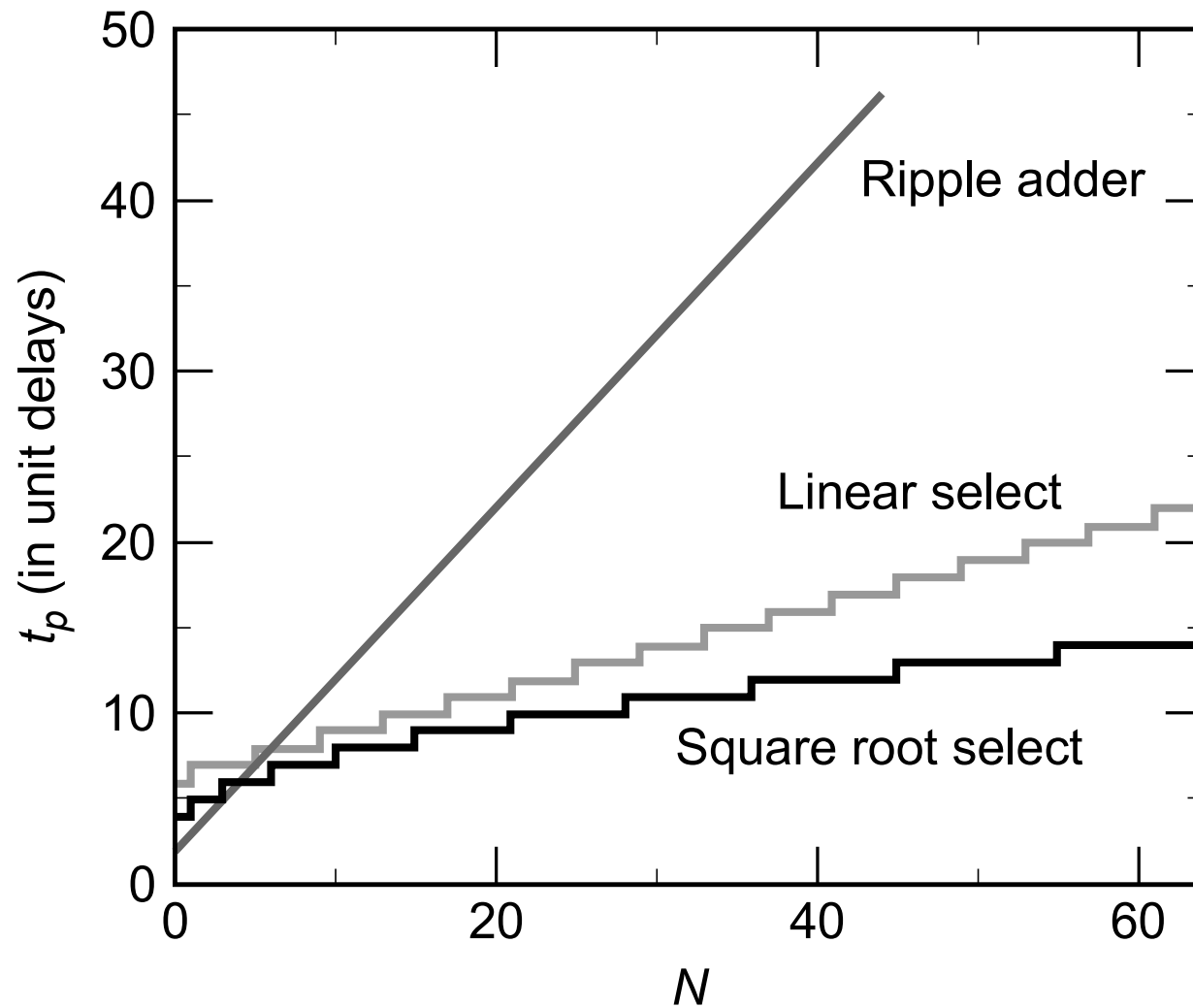
16-bit Linear Carry-Select Adder



$$N=16 \quad M=4$$

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

Adder Delays - Comparison

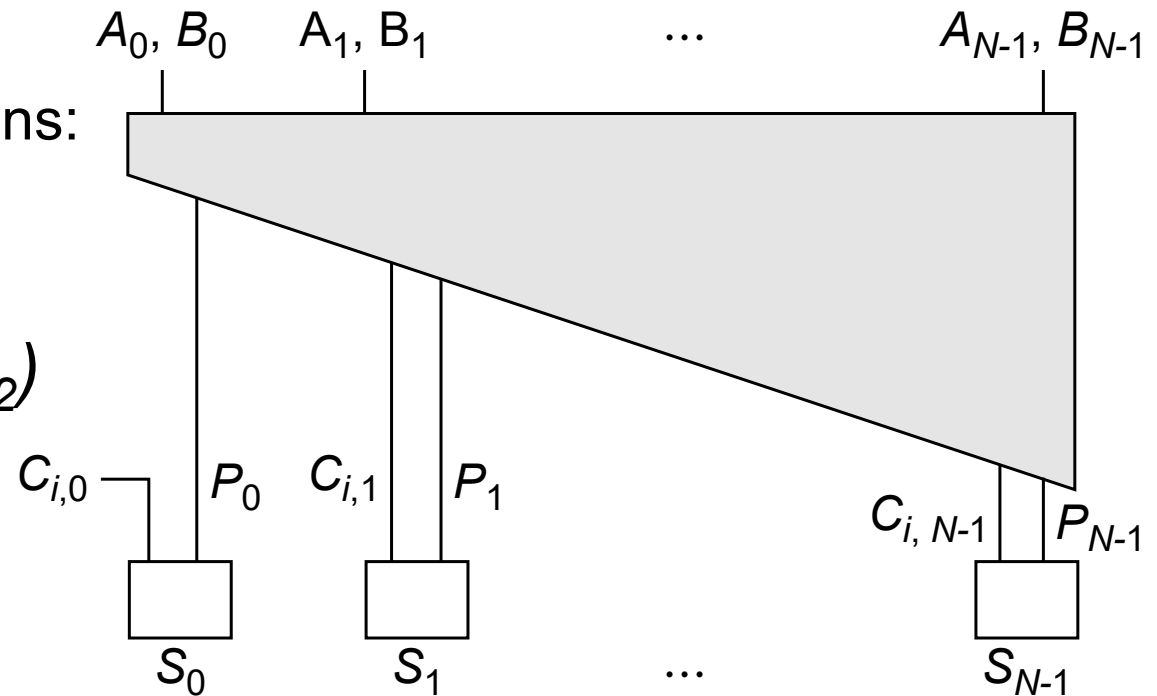


Look-Ahead - Basic Idea

Expanding Lookahead equations:

$$C_{0,k} = G_k + P_k C_{0,k-1}$$

$$C_{0,k} = G_k + P_k (G_{k-1} + P_{k-1} C_{0,k-2})$$



All the way:

$$C_{0,k} = G_k + P_k (G_{k-1} + P_{k-1} (\dots + P_1 (G_0 + P_0 C_{i,0}))))))$$

Carry Determination

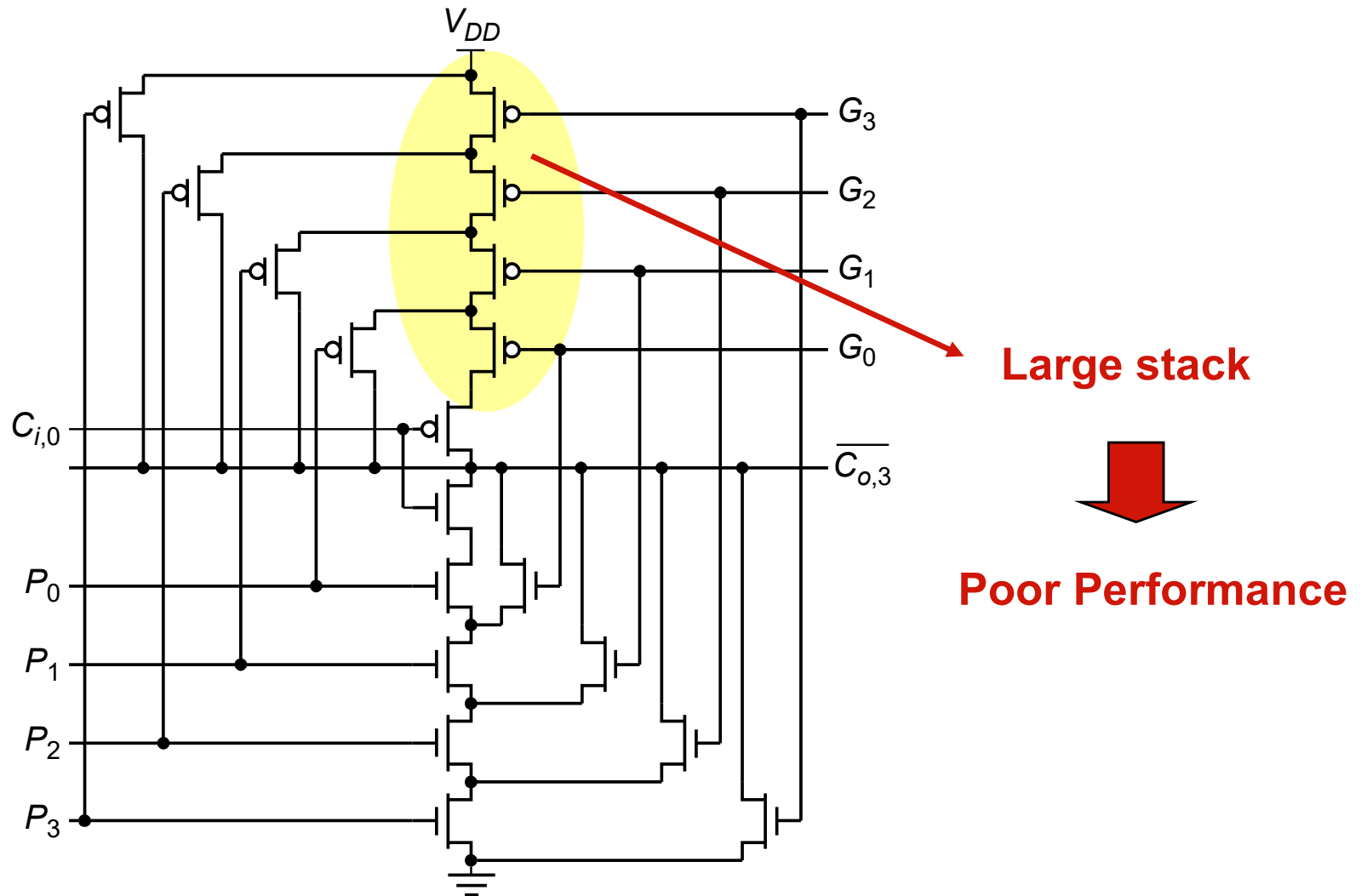
$$C_{o,0} = G_0 + P_0 C_{i,0}$$

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

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

$$C_{o,3} = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{i,0}$$

4-bit Look-Ahead



Multipliers

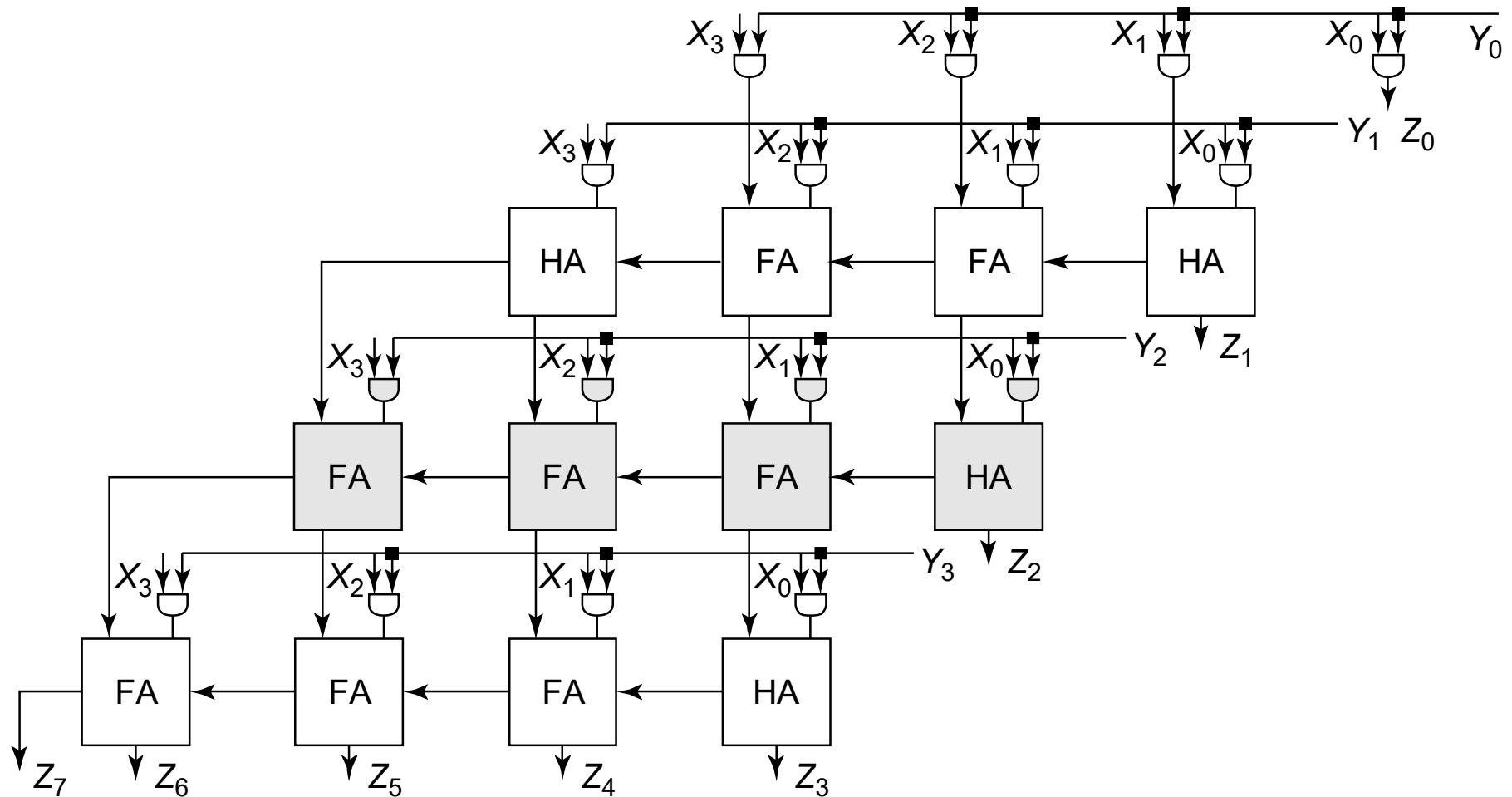
The Binary Multiplication

				1	0	1	0	1	0		Multiplicand
x					1	0	1	1			Multiplier
				1	0	1	0	1	0		Partial products
			1	0	1	0	1	0			
		0	0	0	0	0	0				
	+	1	0	1	0	1	0				
		1	1	1	0	0	1	1	1	0	Result

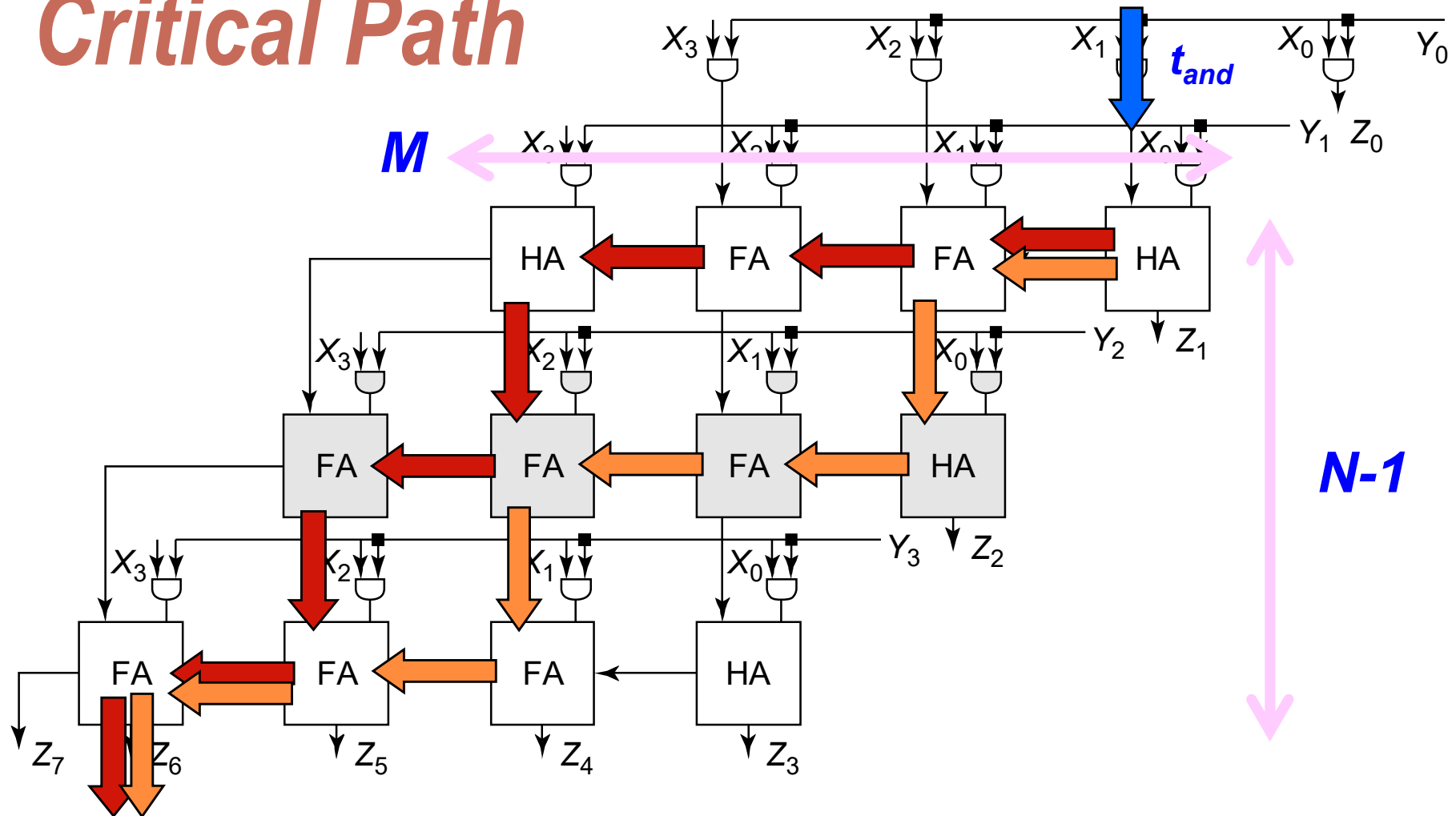
Reduce Partial Products

- Partial products can be reduced by multiplier transformation
(Booth's Recording)
- Reduce number of partial products is equivalent to reducing the number of additions

4X4 Bit-Array Multiplier



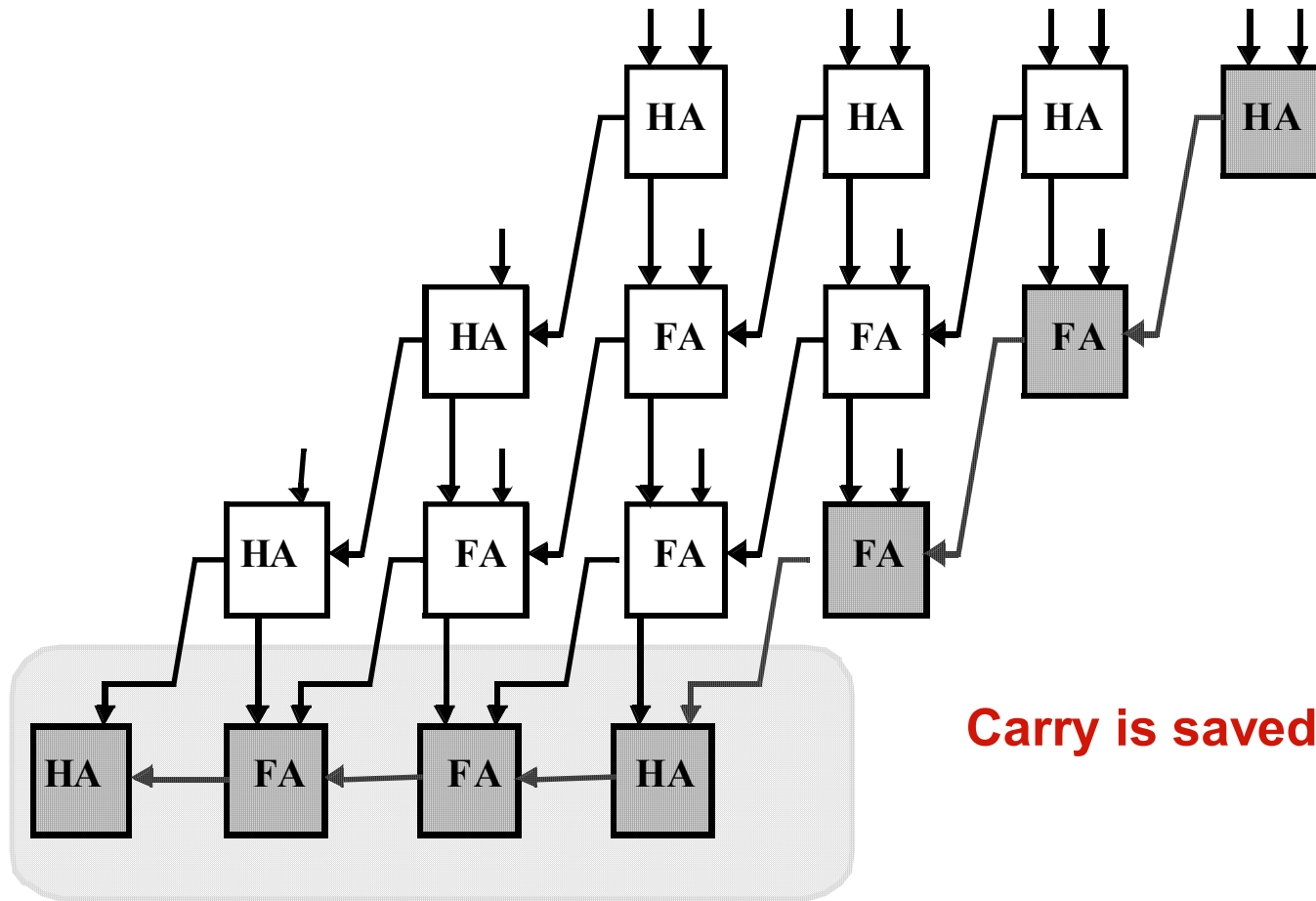
Critical Path



For $M \times N$ Bit-Array Multiplier

$$t_{multiplier} = [(M-1) + (N-2)] t_{carry} + (N-1) t_{sum} + t_{and}$$

Carry-Save Multiplier

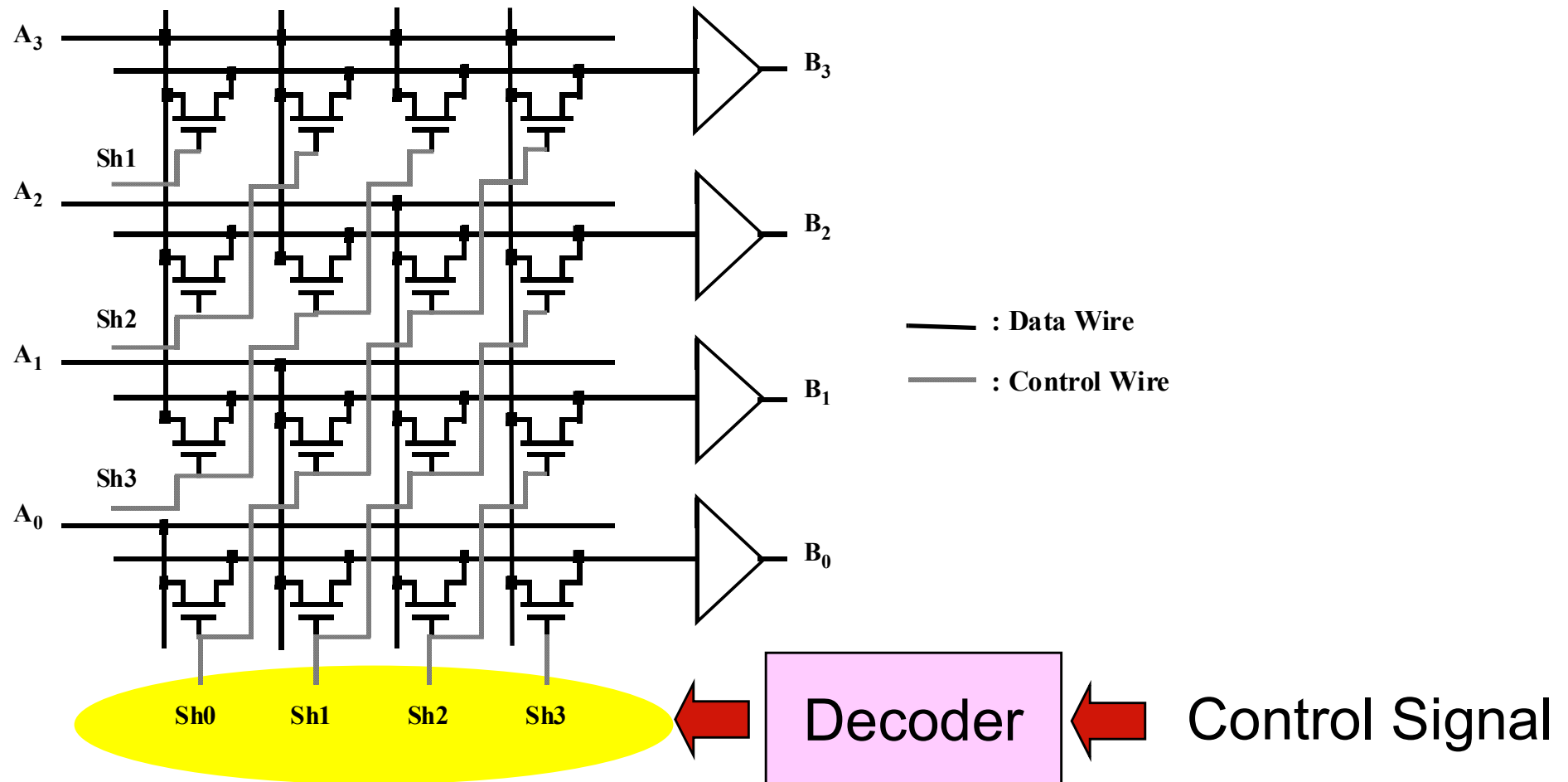


Vector Merging Adder

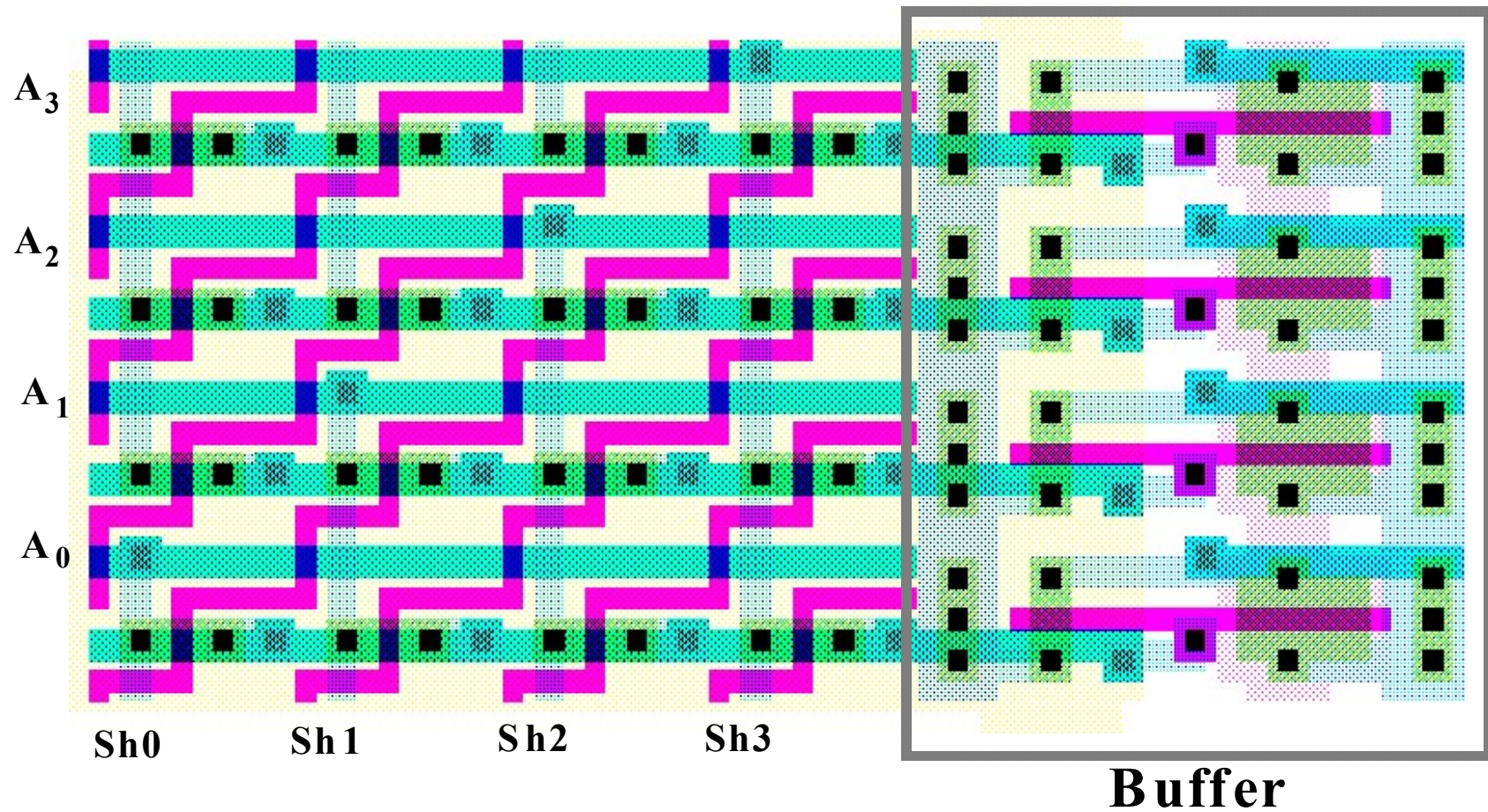
$$t_{multiplier} = (N-1) t_{carry} + t_{and} + t_{merge}$$

The Barrel Shifter

Area Dominated by Control Wiring

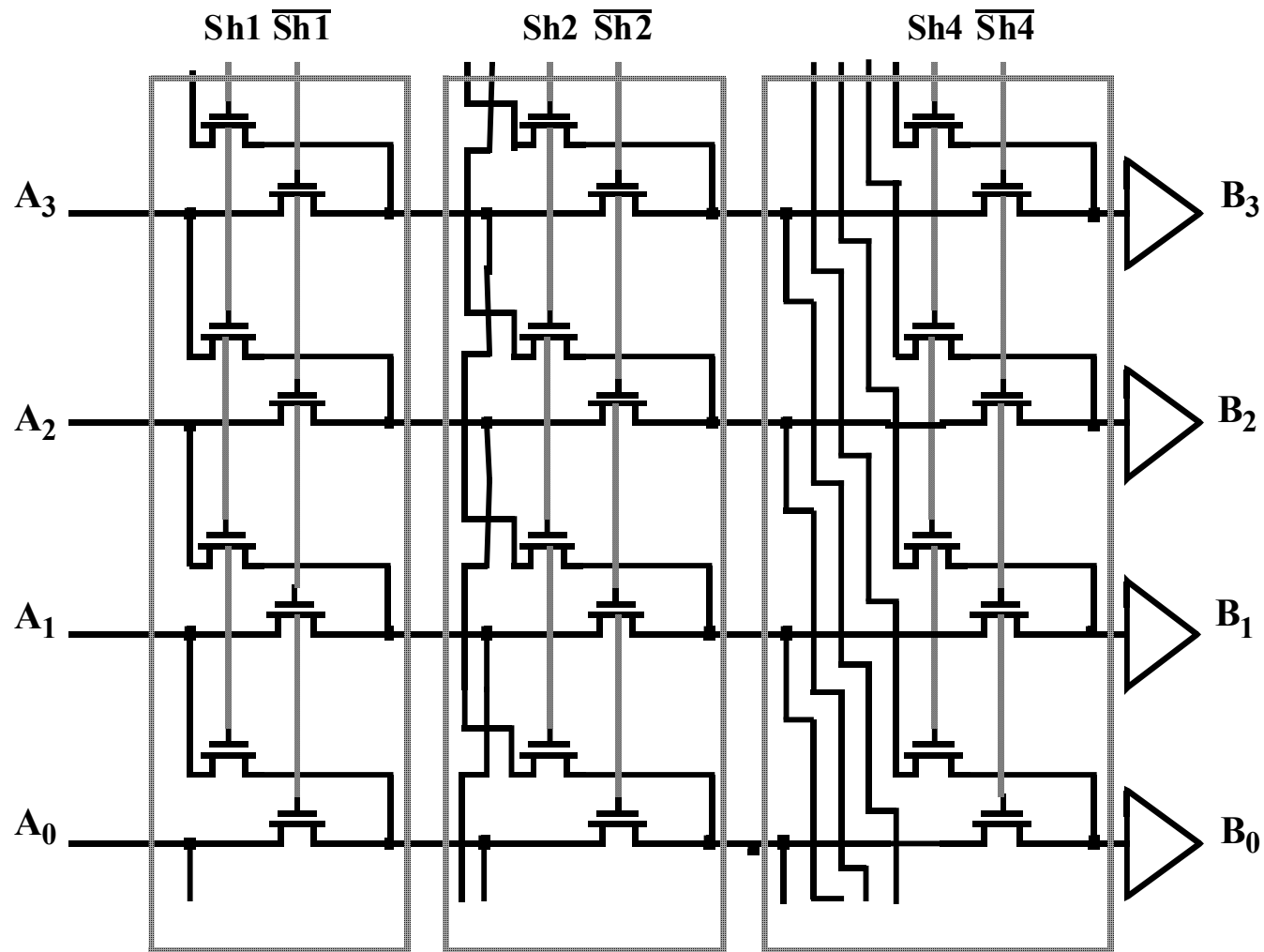


4x4 barrel shifter



$$\text{Width}_{\text{barrel}} \sim 2 p_m M$$

Logarithmic Shifter



0-7 bit Logarithmic Shifter

