CG2027 Transistor-Level Digital Circuits

Handout #4: CMOS Arithmetic Logic Unit

National University of Singapore

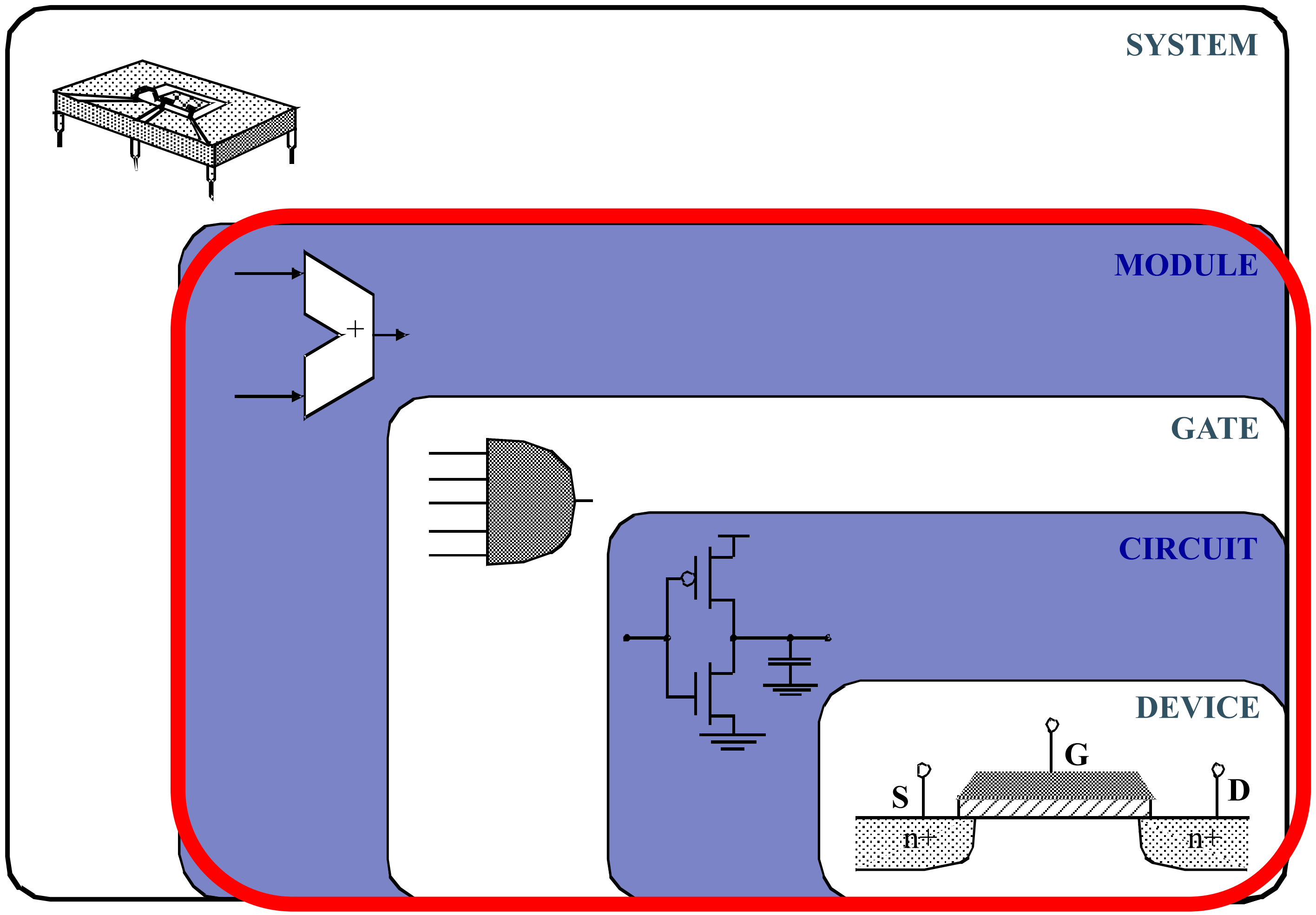
Kelvin Fong

Lecture Overview

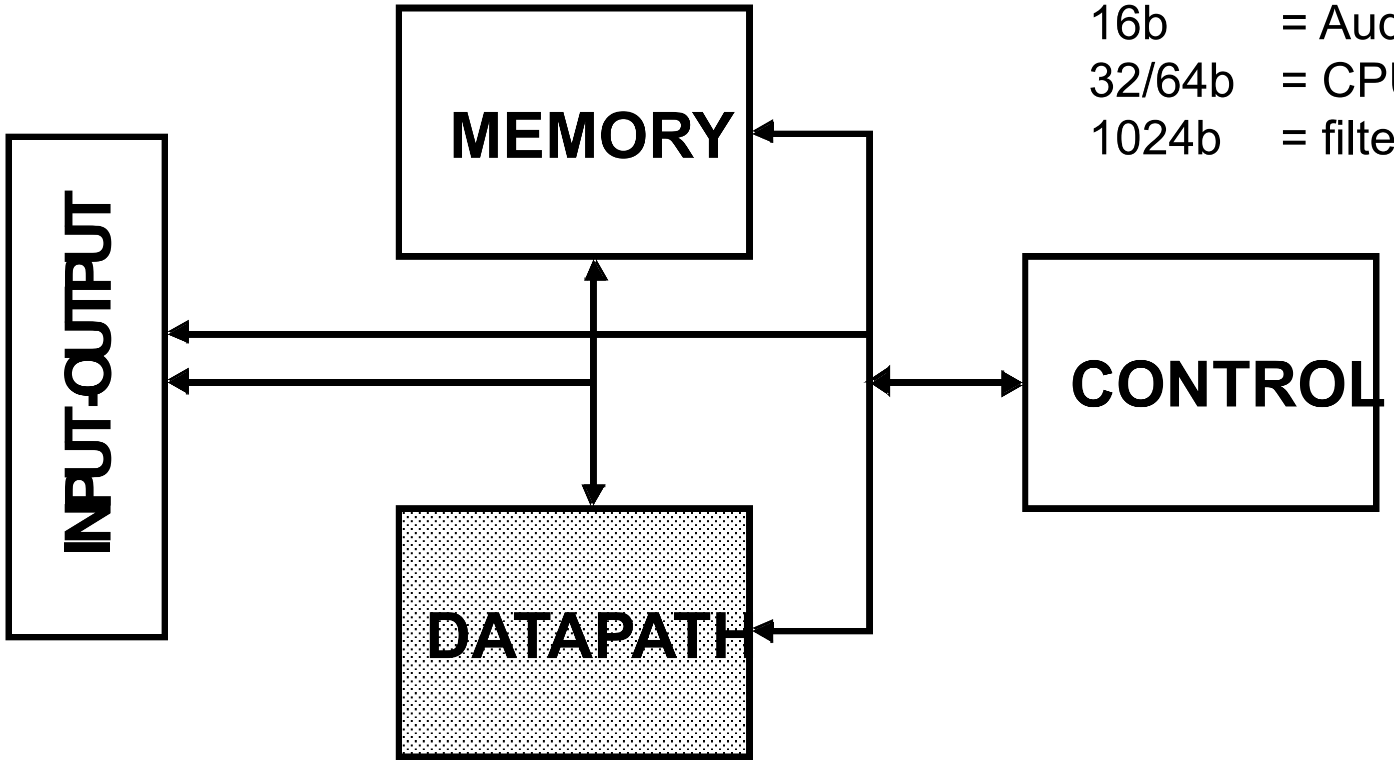
In this lecture, you will learn about

- CMOS Arithmetic Logic Unit
  - Generic Digital Processor Architecture
  - Adders (Ripple Carry, Carry Bypass, Carry Select)
  - Shifters

Design Abstraction Levels



A Generic Digital Processor



4b	= uController
8b	= Video (each RGB)
16b	= Audio
32/64b	= CPU
1024b	= filter, encryption

Building Blocks for Digital Architectures

Arithmetic unit

- Bit-sliced datapath (adder, multiplier, shifter, comparator, etc.)

Memory

- RAM, ROM, Buffers, Shift registers

Control

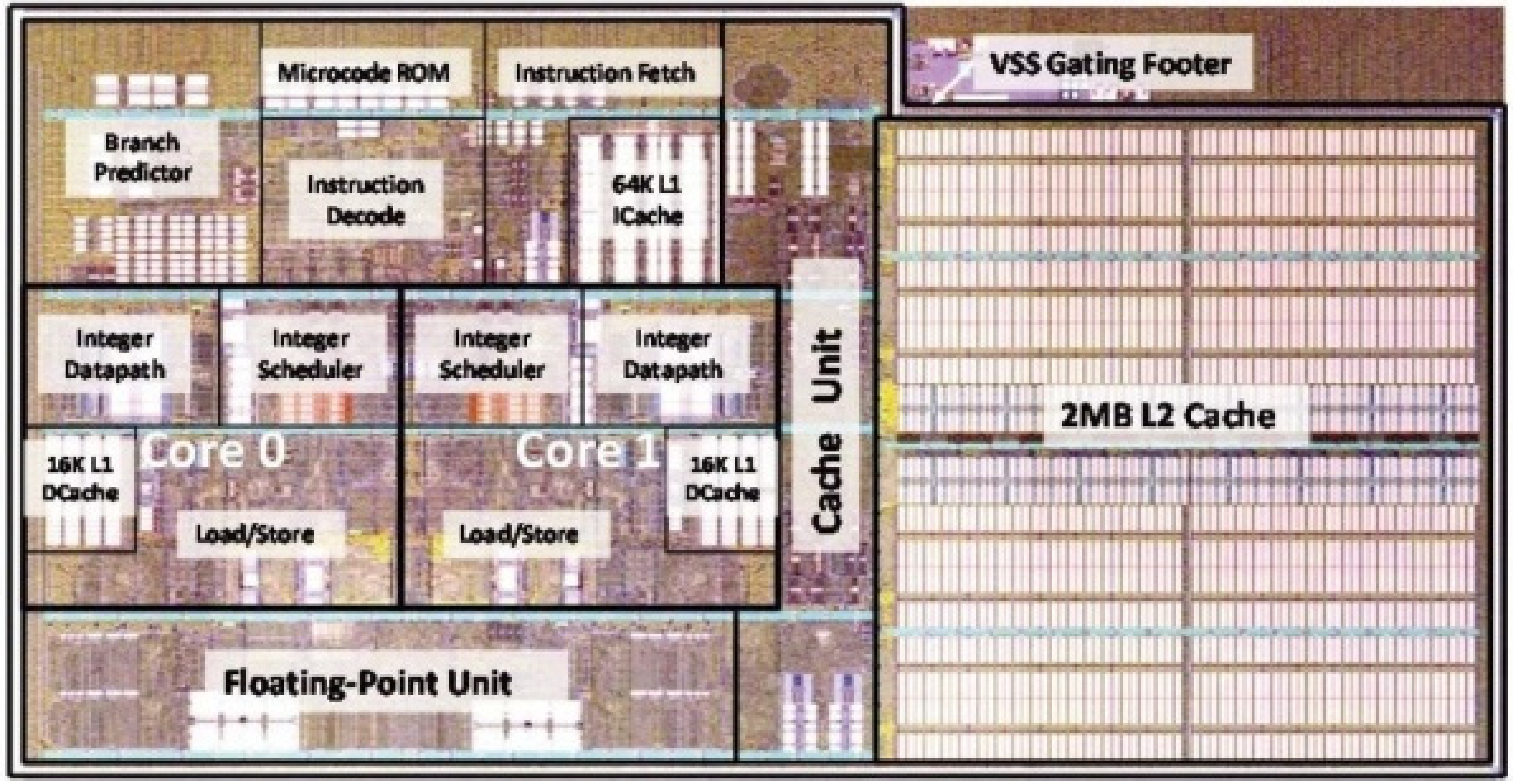
- Finite state machine (PLA, random logic.)
- Counters

Interconnect

- Switches
- Arbiters
- Bus

NUS National University of Singapore

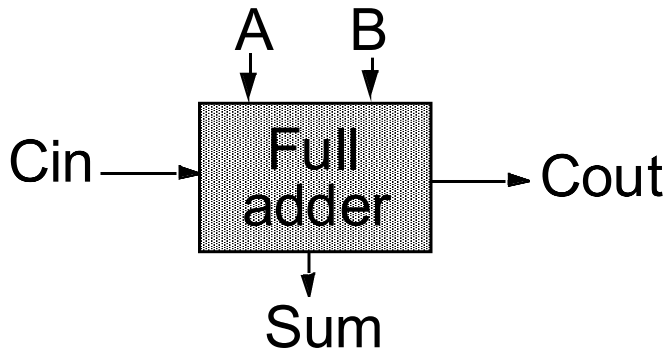
X86 datapath



ALU and UHP logic
➔ custom design is necessary here

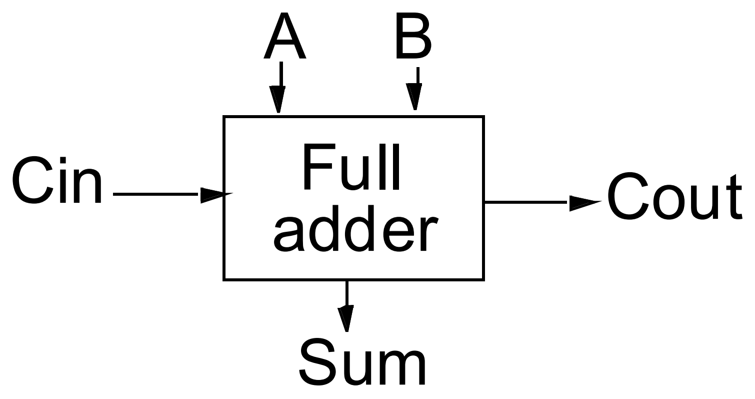
ADDERS

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



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

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

$$C_o = AB + BC_i + AC_i$$

$$= AB + C_i(A \oplus B)$$

Express Sum and Carry as f(P, G, D)

Define 3 new variable which ONLY depend on A, B

Generate (G) = AB

Propagate (P) = $A \oplus B$

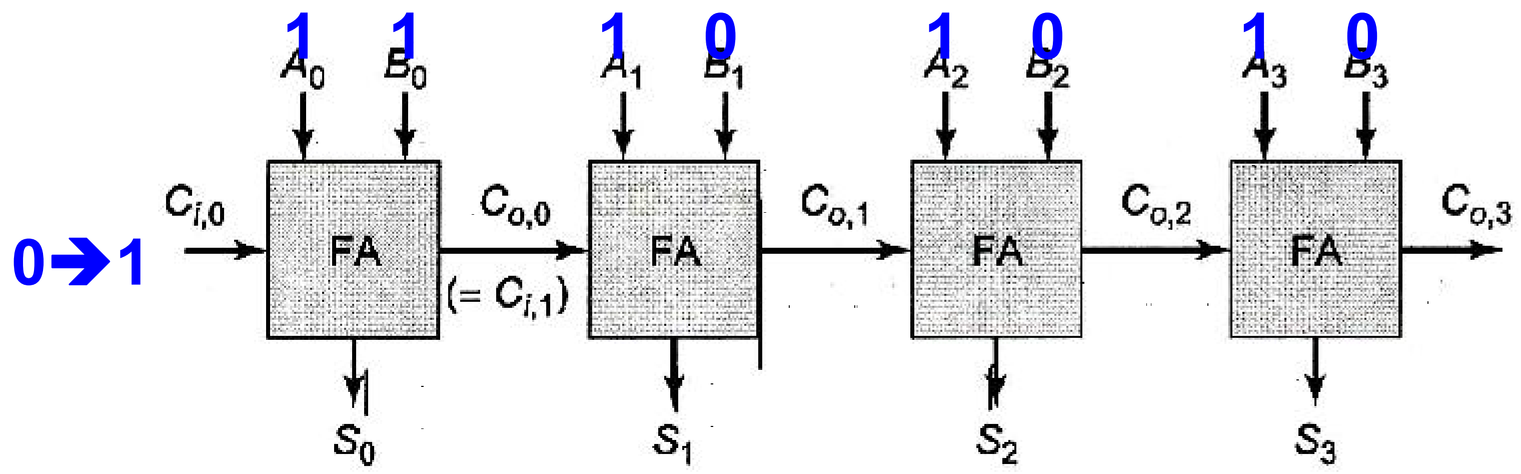
Delete (D) = $\overline{A}\,\overline{B}$

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

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

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

$$t_d = O(N)$$

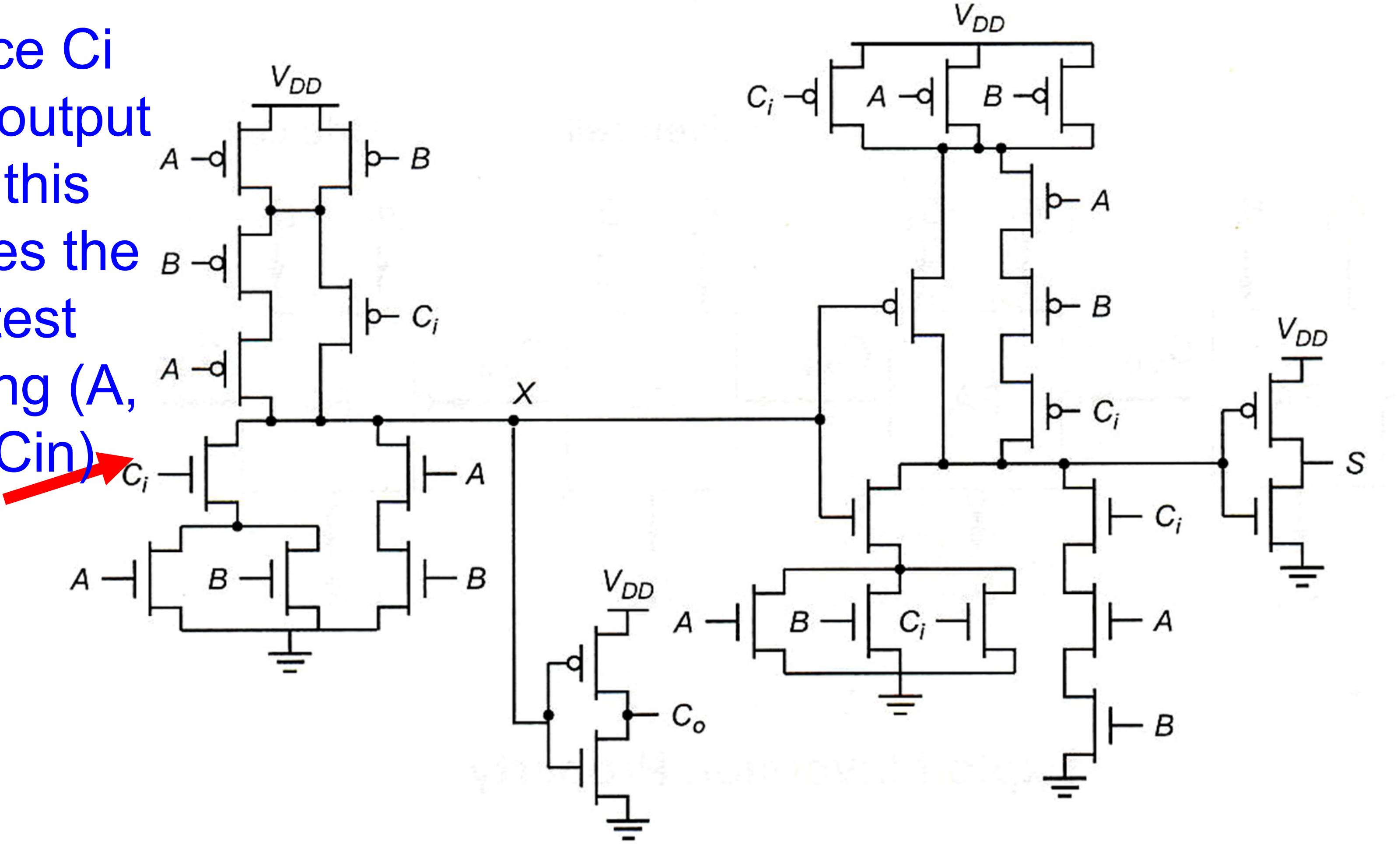
$$t_{add,ripple-carry} = (N-1)t_{carry} + t_{sum}$$

Goal: Make the fastest possible carry path circuit

Q) Subtractor?
Comparator?

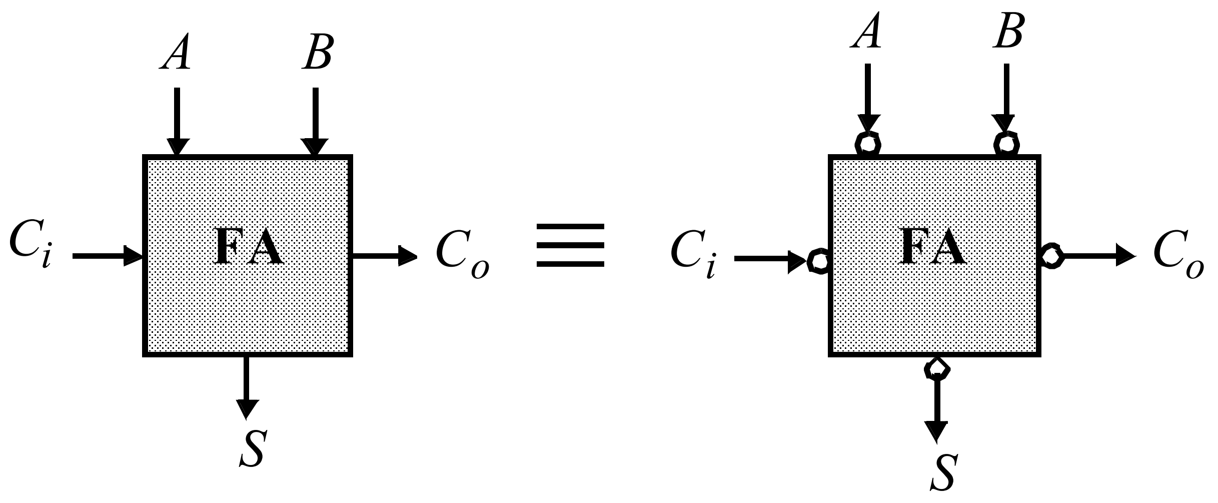
Complementary Static "1b" CMOS Full Adder

Place Ci near output as this arrives the latest among (A, B, Cin)



28 Transistors

Inversion Property

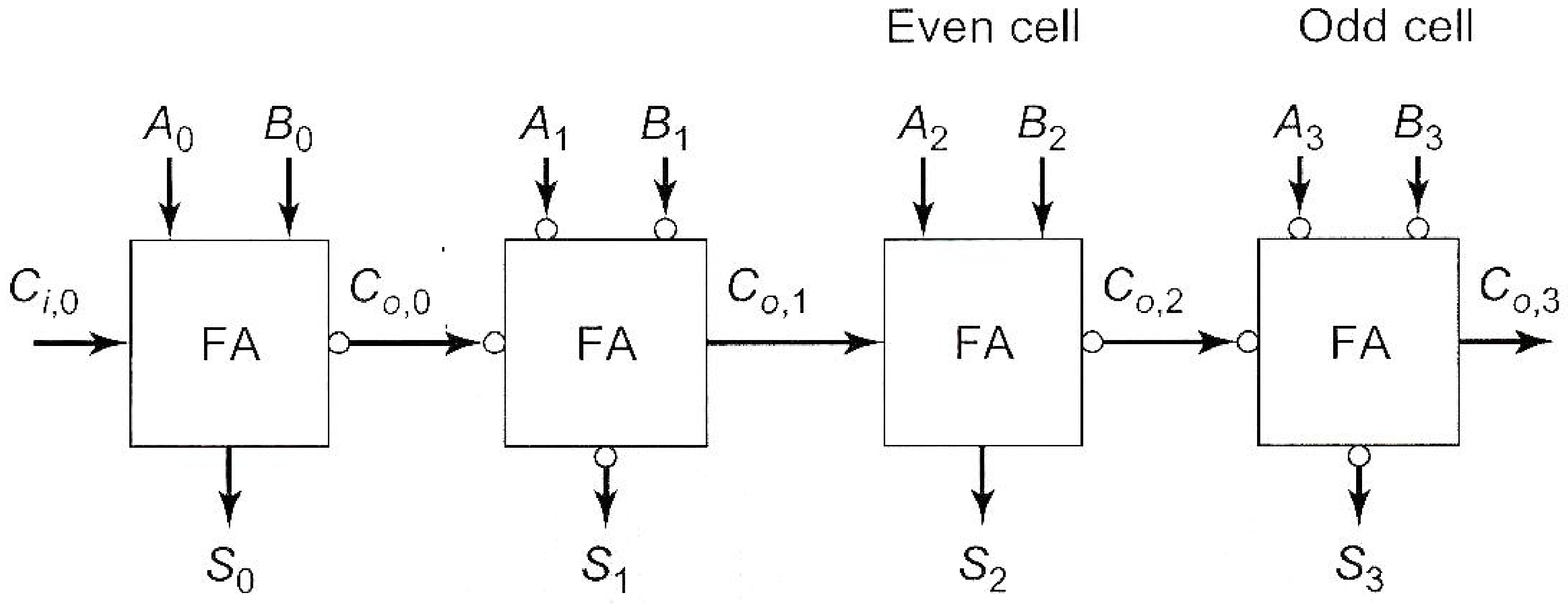


$$\bar{S}(A, B, C_i) = S(\bar{A}, \bar{B}, \overline{C_i})$$

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

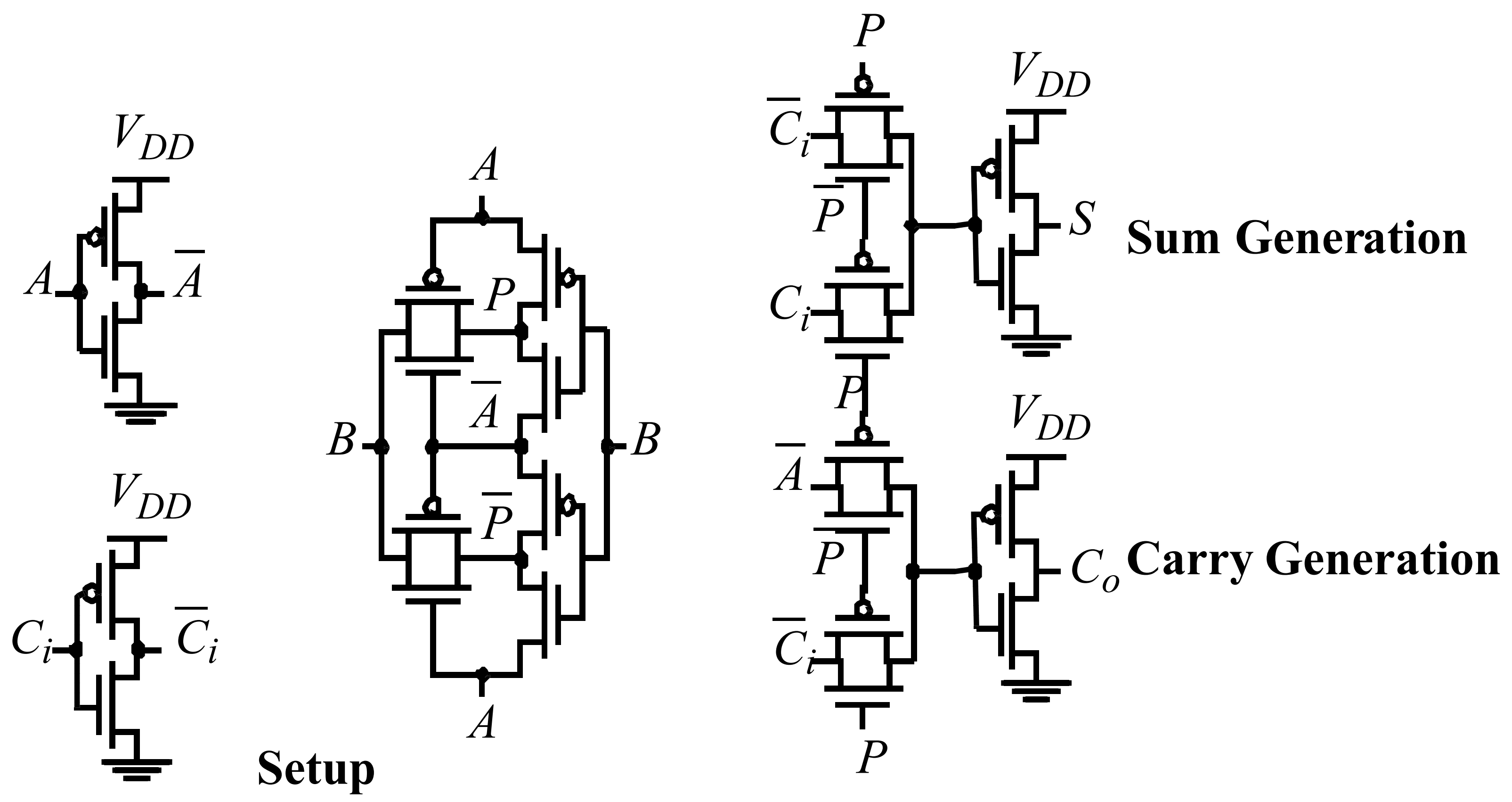
Minimize Critical Path

by Reducing Inverting Stages

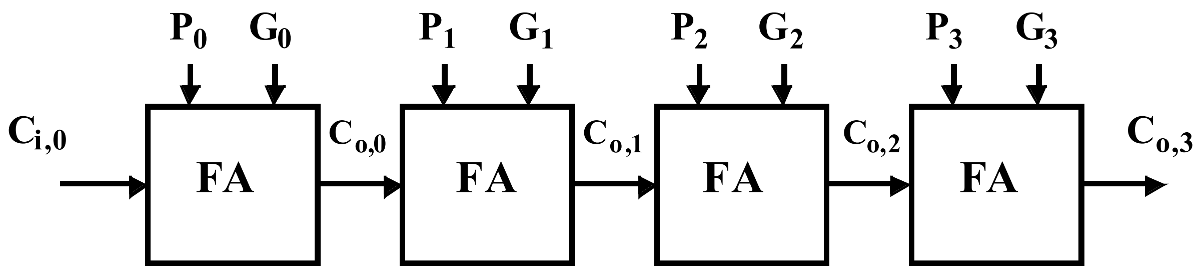


Exploit Inversion Property

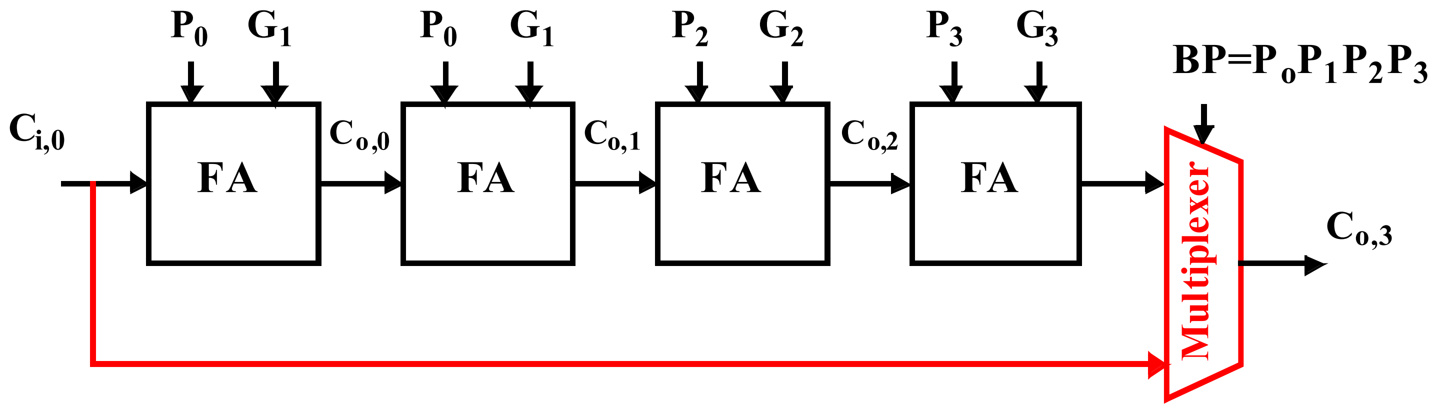
Transmission Gate Full Adder



Carry-Bypass Adder



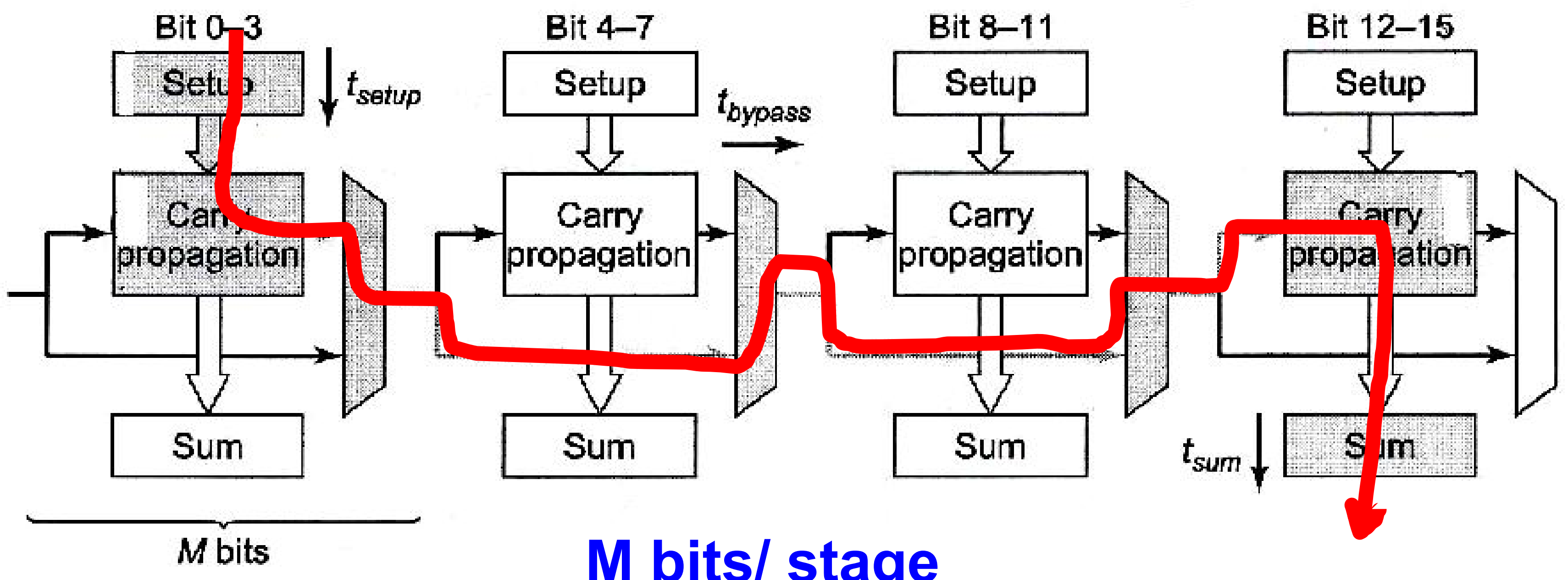
Also called Carry-Skip



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

Carry-Bypass Adder (cont.)

N bits total



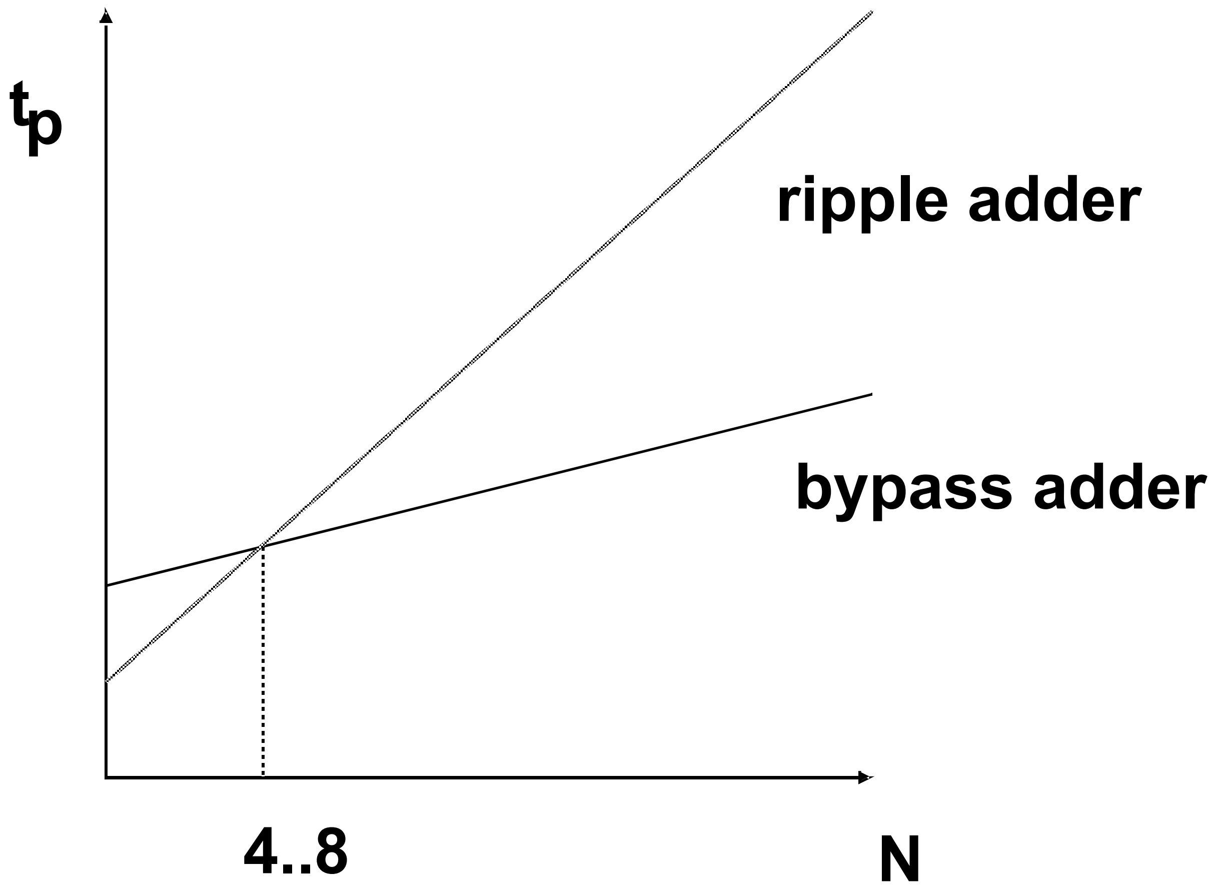
M bits/ stage

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

Delay through a single bit

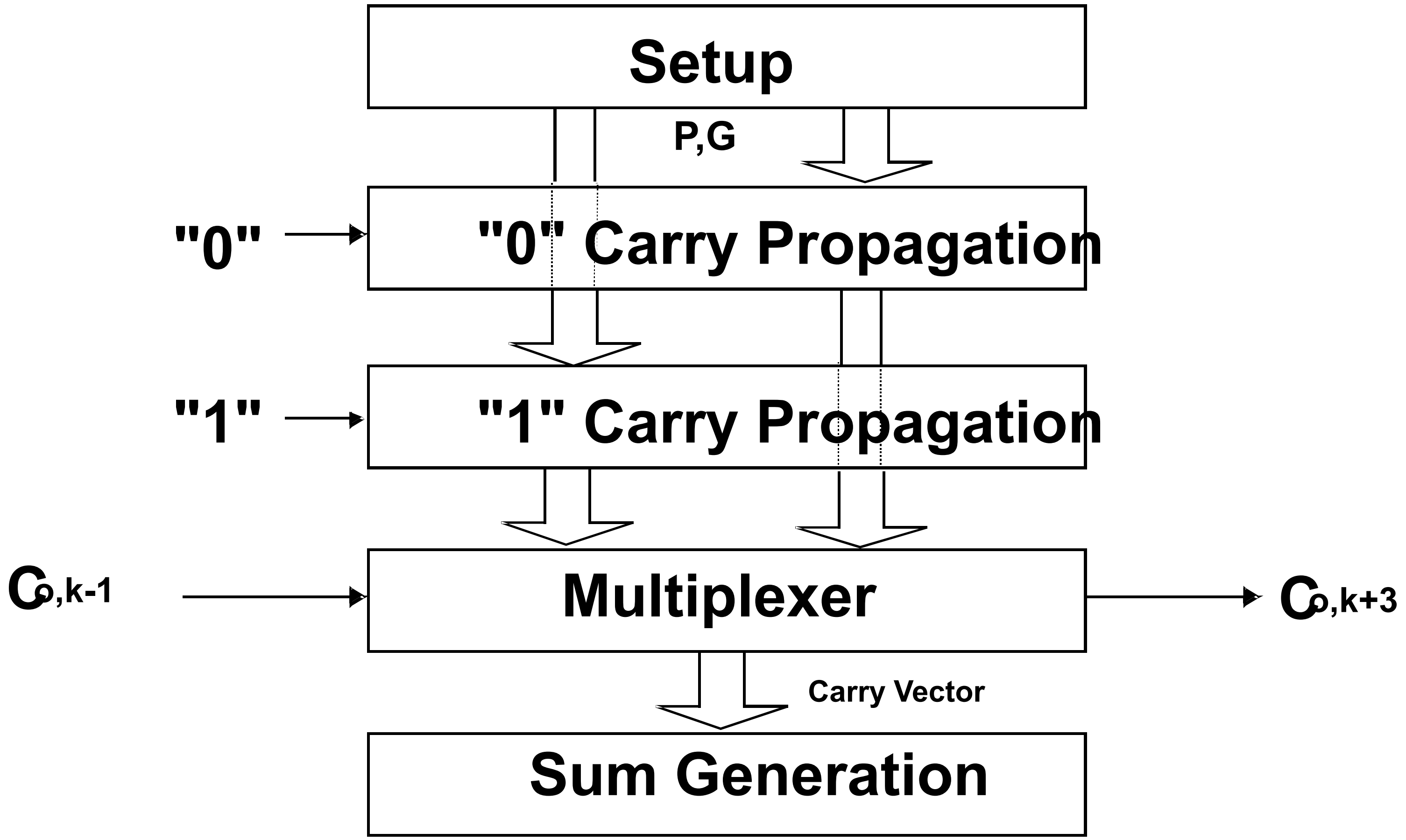
- t_{setup} : the fixed overhead time to create the generate and propagate signals
- t_{carry} : the propagation delay through a single bit. The worst-case propagation delay through a single stage of M bits is approximately M times larger.
- t_{bypass} : the propagation delay through the bypass multiplexer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

Carry Ripple versus Carry Bypass

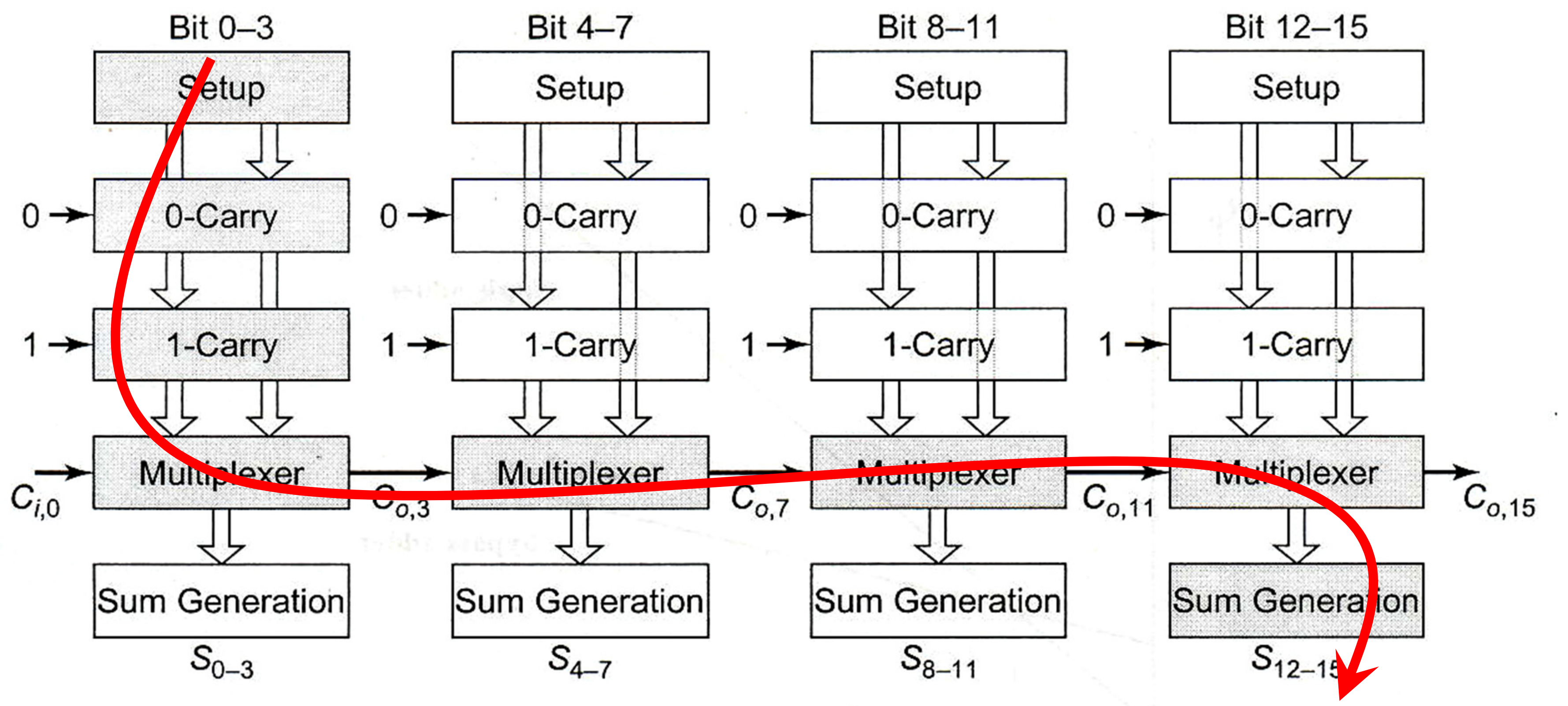


Note: carry-bypass adder will have advantage only if N=4~8b or greater

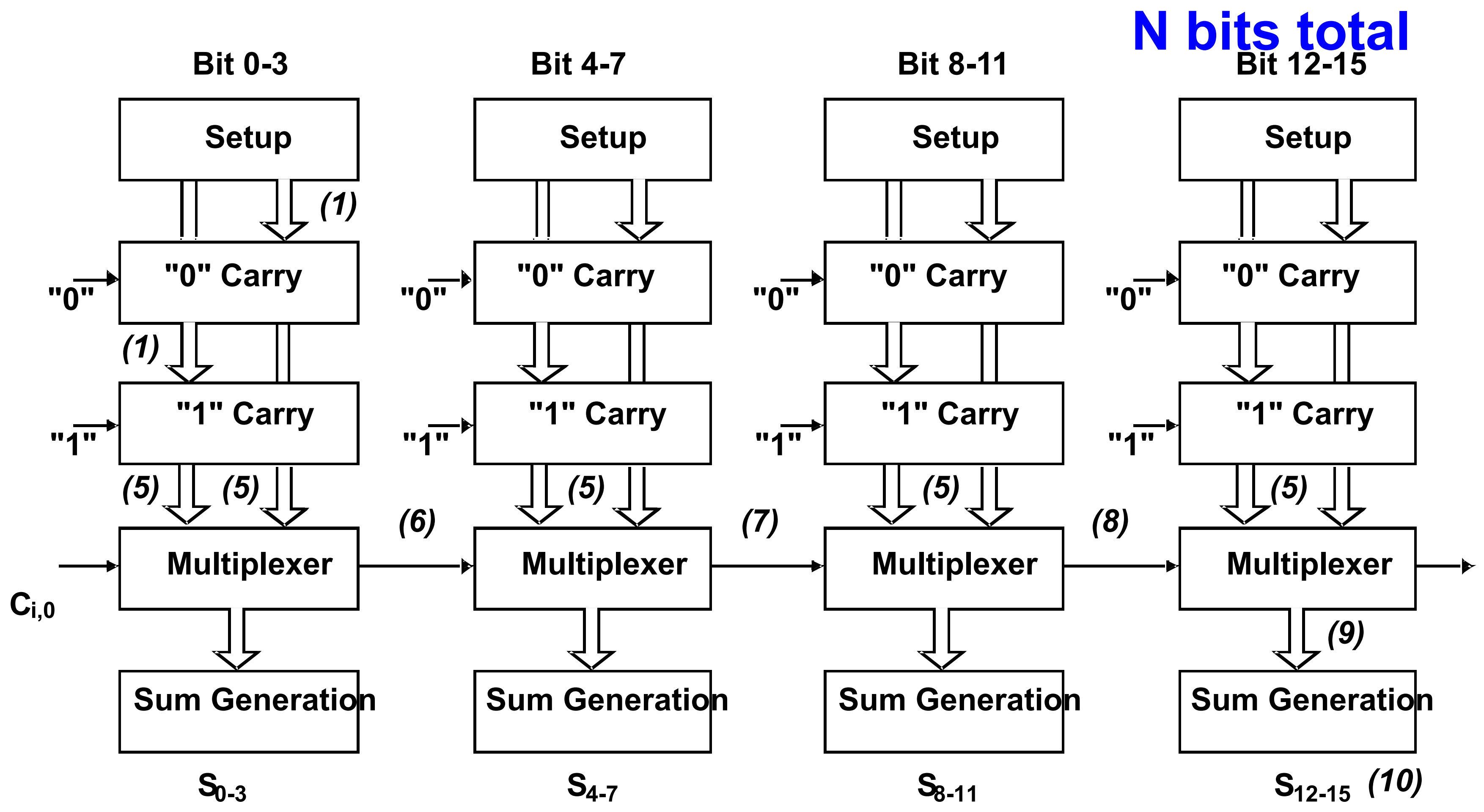
Carry-Select Adder



Carry Select Adder: Critical Path



Linear Carry Select

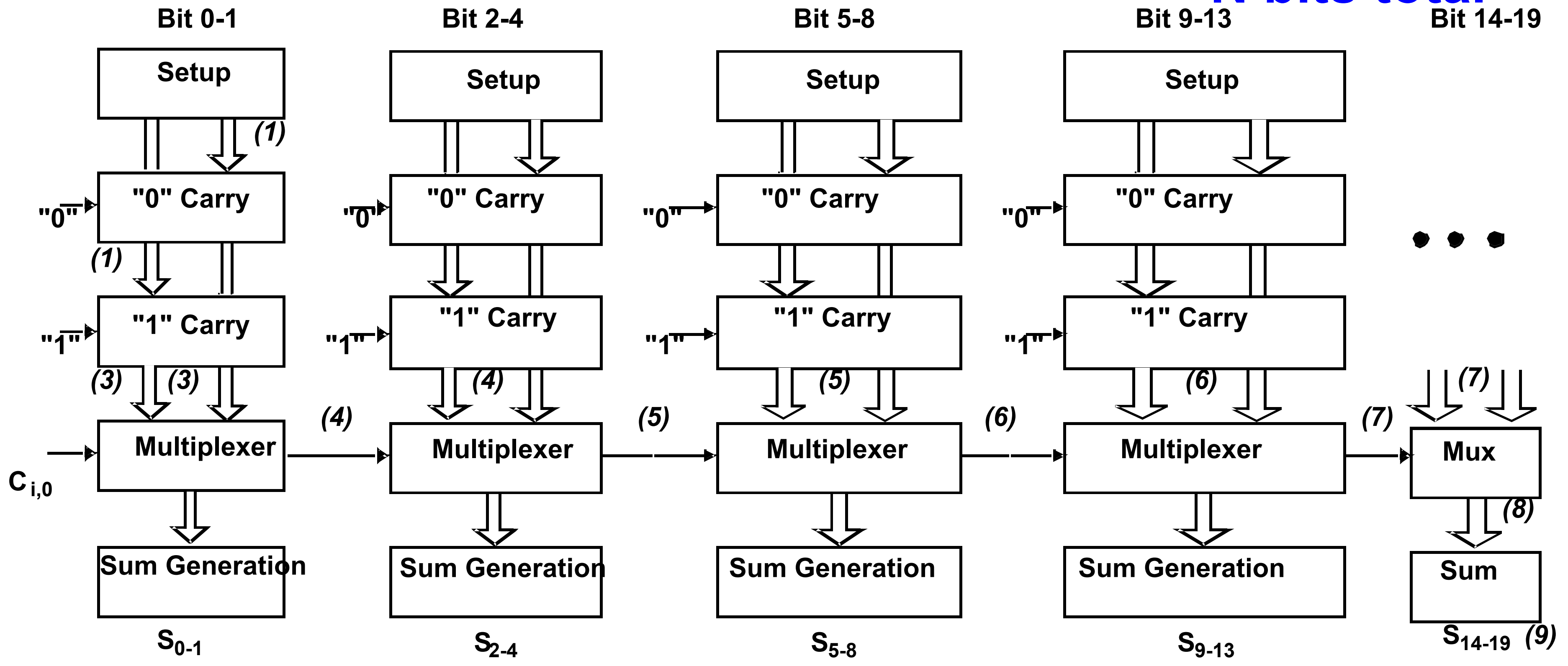


M bits/ stage

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

Square Root Carry Select

N bits total



P stages

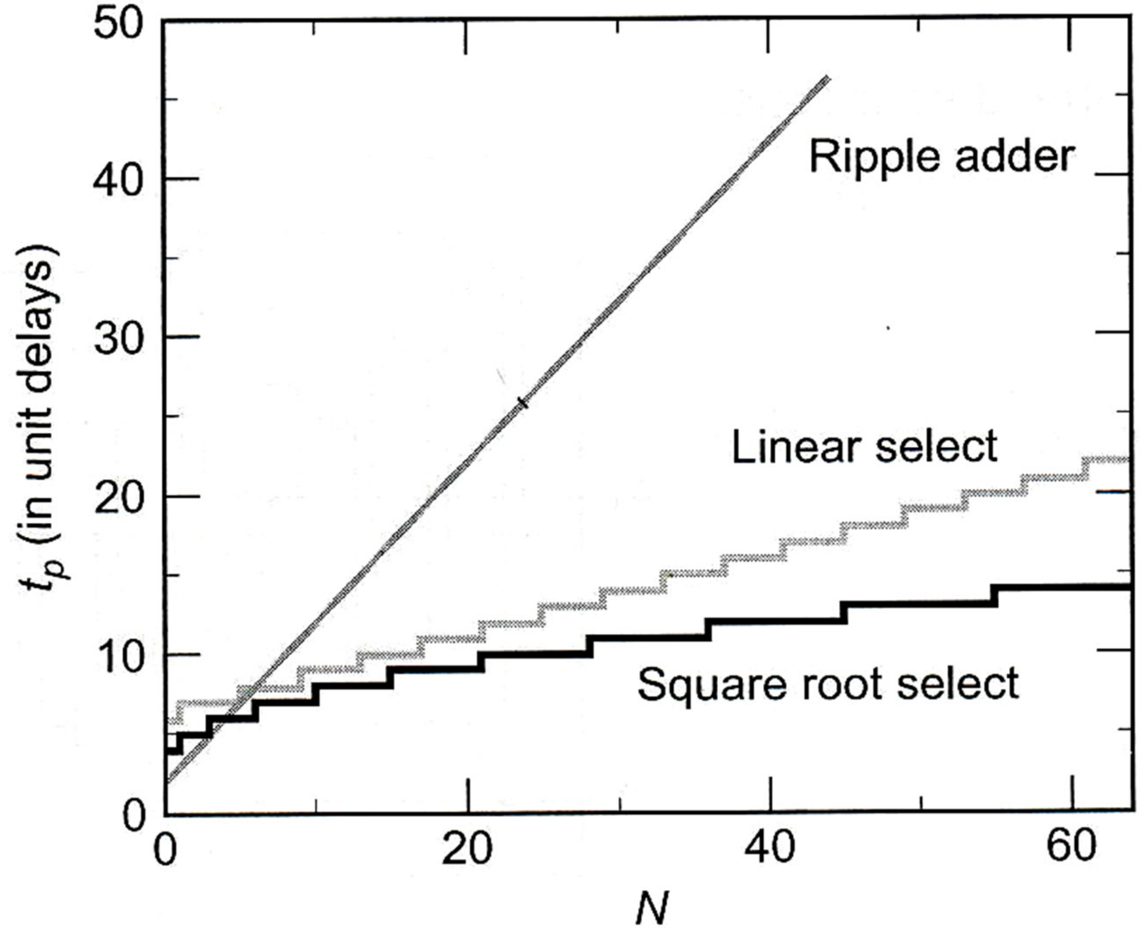
$$N = M + (M+1) + (M+2) + \cdots + (M+P-1)$$

$$= \mathrm{MP} + \frac{\mathrm{P(P-1)}}{2} = \frac{\mathrm{P}^2}{2} + \mathrm{P}\left(\mathrm{M} - \frac{1}{2}\right) \approx \frac{P^2}{2}$$

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

t_{mux}: delay for multiplexer

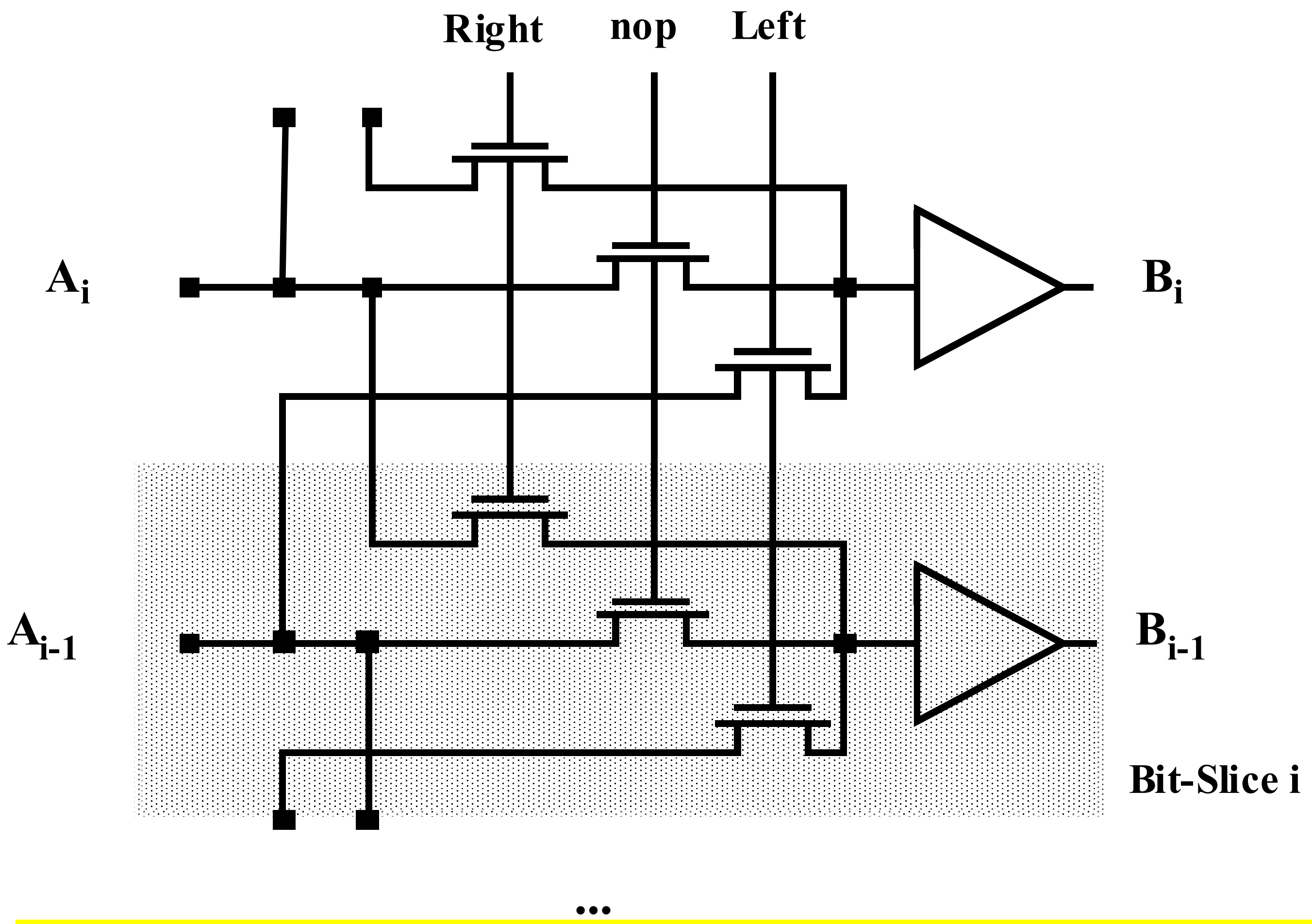
Adder Delays - Comparison



Note: carry-select adder will have advantage if N becomes greater

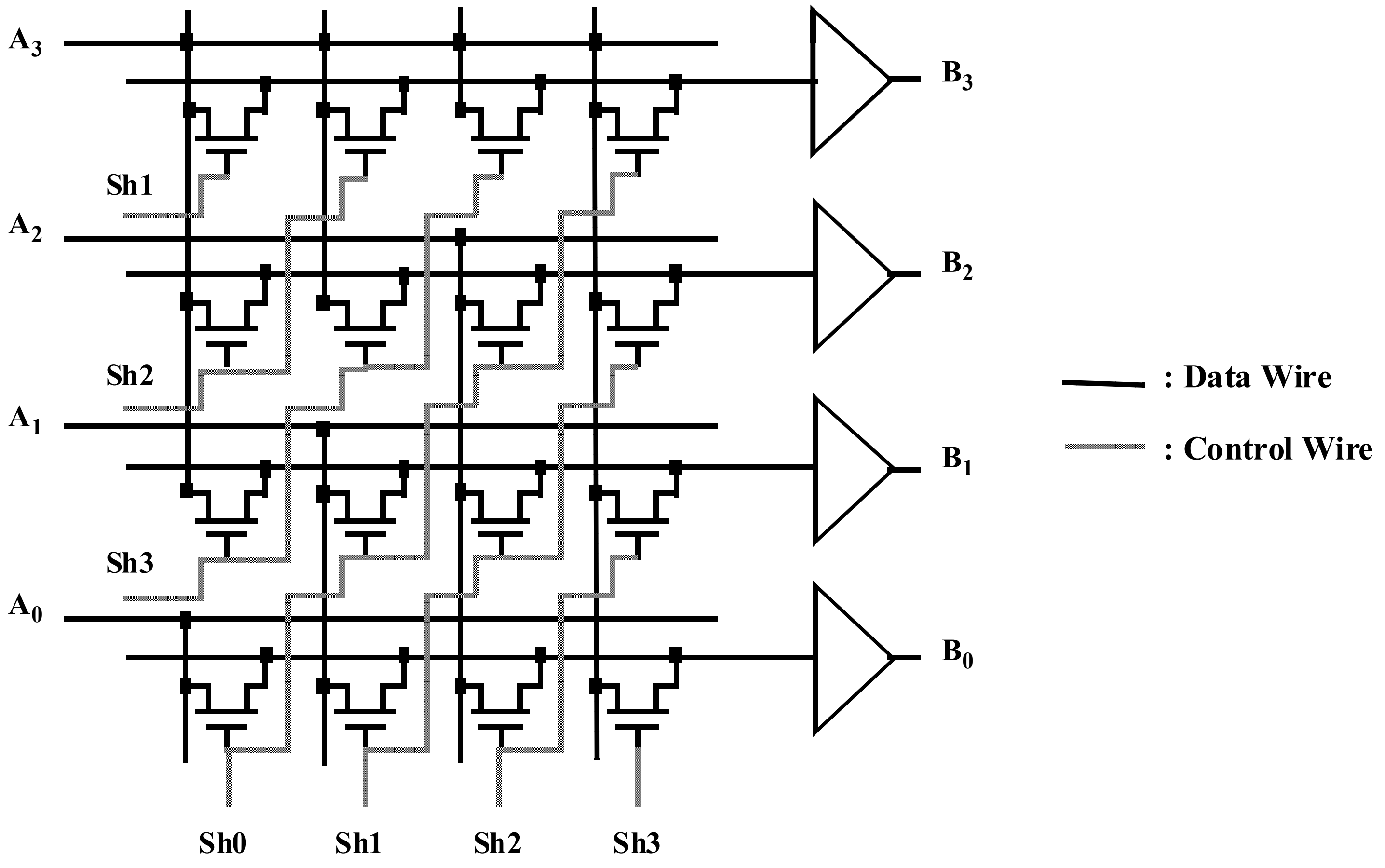
SHIFTERS

The Binary Shifter



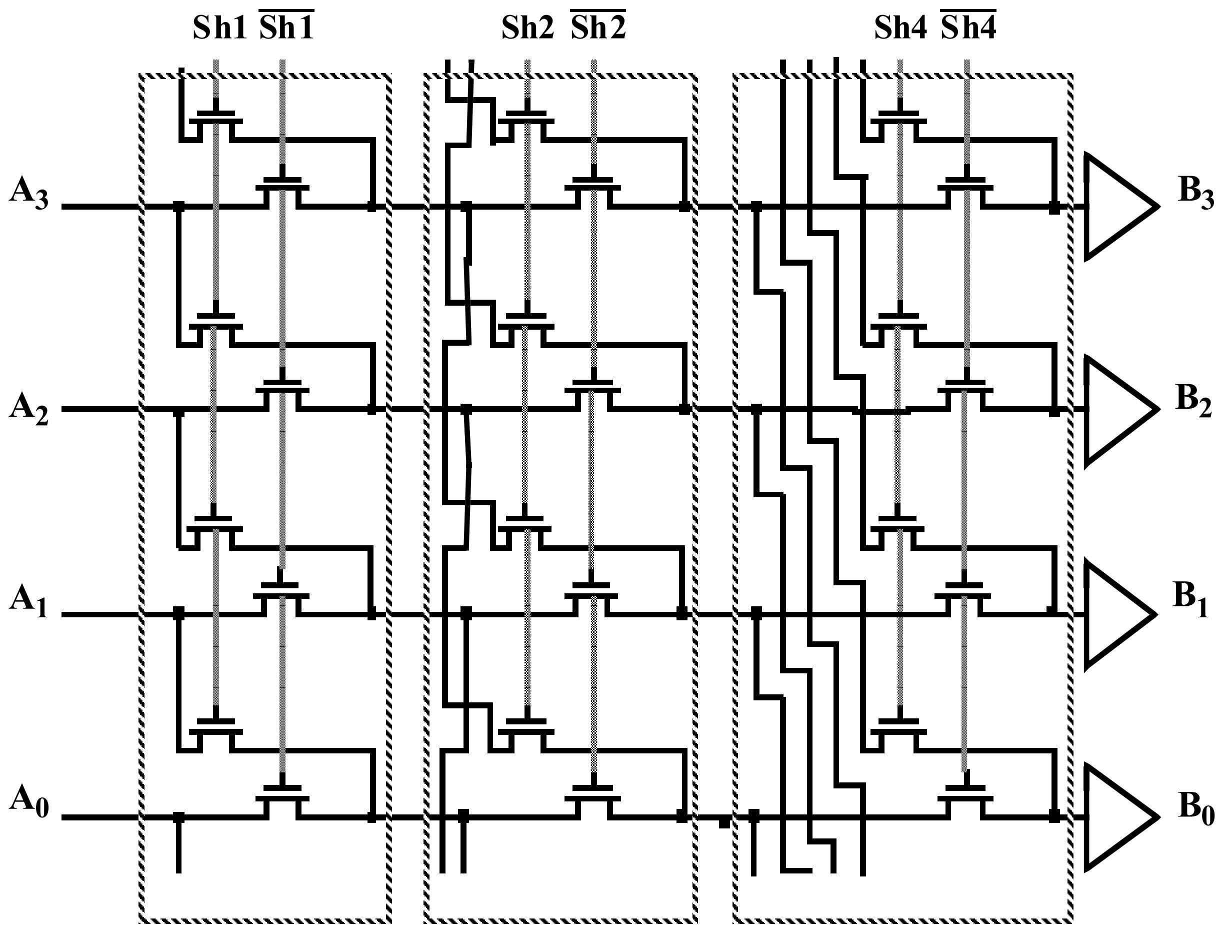
Note: in a binary number, shift right == divide
shift left == multiply

The Barrel Shifter



Area Dominated by Wiring

Logarithmic Shifter


Sh1 $\overline{\text{Sh1}}$
Sh2 $\overline{\text{Sh2}}$
Sh4 $\overline{\text{Sh4}}$
A_3
A_2
A_1
A_0
B_3
B_2
B_1
B_0

Note: reduce the wiring overhead by introducing “logarithmic” shifting.