Datapath Components (2)

Prof. Usagi

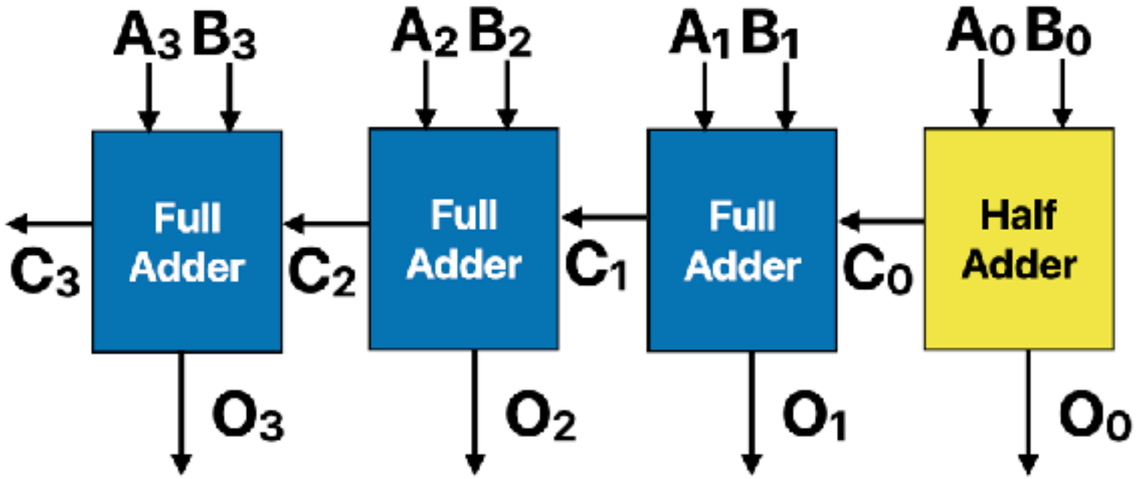
Recap: 2's complement

- Guidelines
  - ✓ Obvious representation of 0, 1, 2, ...
  - ✓ Efficient usage of number space
  - ✓ Equal coverage of positive and negative numbers
  - Easy hardware design
- 1's complement + 1 = 2's complement **Does not waste 1111 anymore**
  - Invert every bit, then + 1
  - -1 = b'1110 + b'1 = b'1111

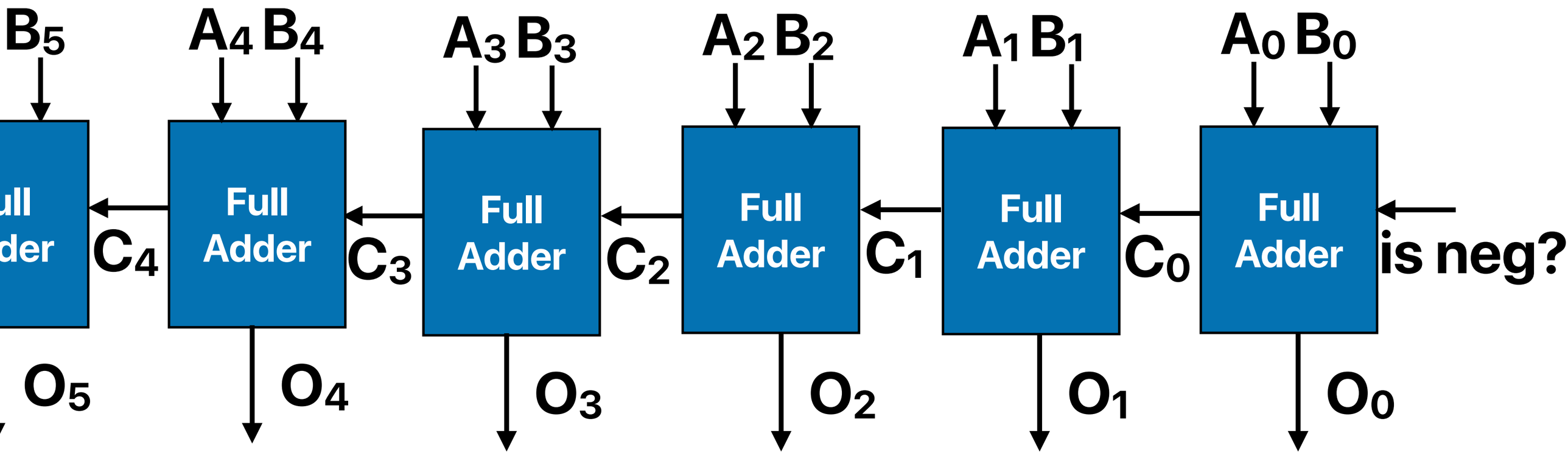
Decimal	Binary	Decimal	Binary
0	0000	-1	1111
1	0001	-2	1110
2	0010	-3	1101
3	0011	-4	1100
4	0100	-5	1011
5	0101	-6	1010
6	0110	-7	1001
7	0111	-8	1000

If we want to support subtraction?

- If we would like to extend the 4-bit adder that we've built before to support "A-B" with 2's complement, how many of the followings should we add at least?
  - ① Provide an option to use bitwise NOT A
  - ② Provide an option to use bitwise NOT B
  - ③ Provide an option to use bitwise A XOR B
  - ④ Provide an option to add 0 to the input of the half adder
  - ⑤ Provide an option to add 1 to the input of the half adder
  - A. 1
  - B. 2
  - C. 3
  - D. 4
  - E. 5



We can support more bits!

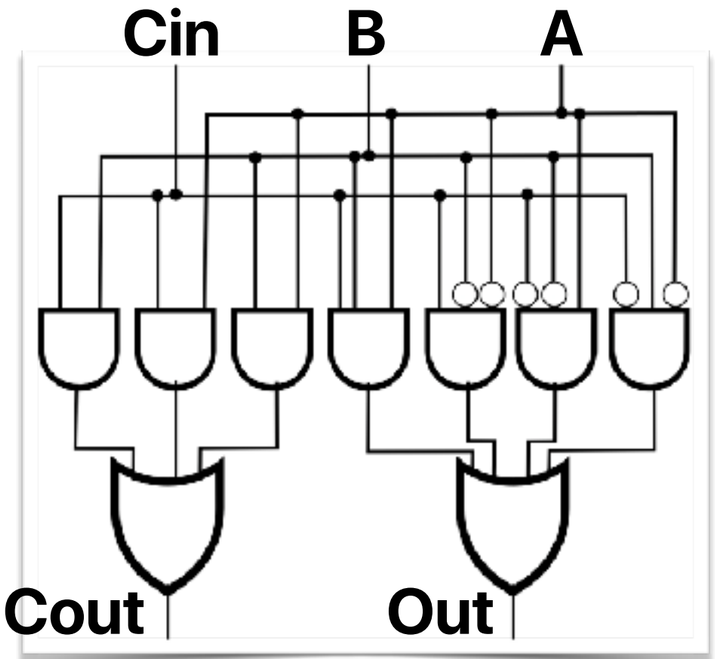


How efficient is the adder?

- One approach estimates transistors, assuming every gate input requires 2 transistors, and ignoring inverters for simplicity. A 2-input gate requires 2 inputs · 2 trans/input = 4 transistors. A 3-input gate requires 3 · 2 = 6 transistors. A 4-input gate: 8 transistors. Wires also contribute to size, but ignoring wires as above is a common approximation.
- Considering the shown 1-bit full adder and use it to build a 32-bit adder, how many transistor do we need?

A. 1152
B. 1600
C. 1664
D. 1792
E. 1984

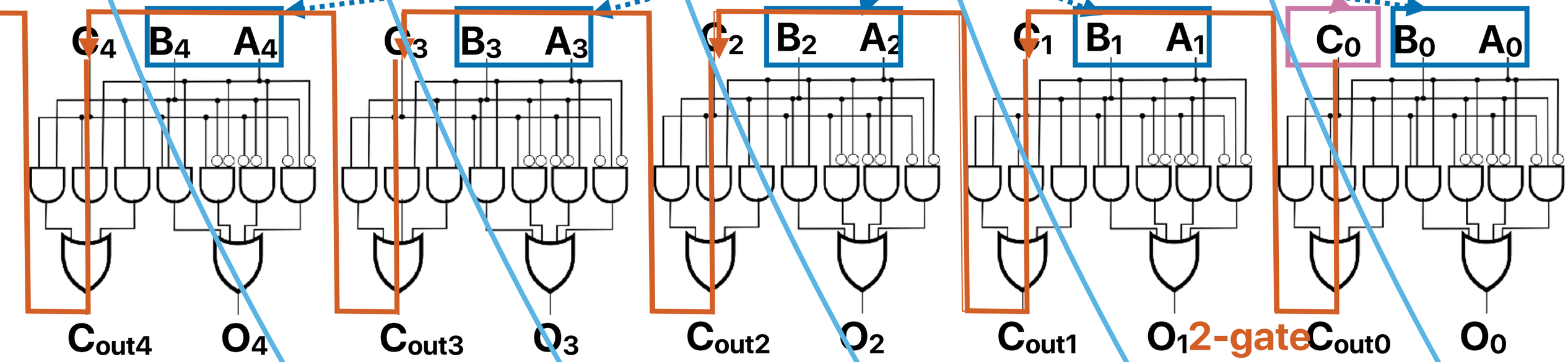
# of 2-inputs: 3
# of 3-inputs: 5
# of 4-inputs: 1
= 3*4 + 5*6 + 1*8 = 50 each



The delay is determined by the "critical path"

Available in the very beginning

Only this is available in the beginning



Carry-Ripple Adder

Outline

- Adders
- Multiplexer
- Multiplier
- Divisor

Carry-lookahead adder

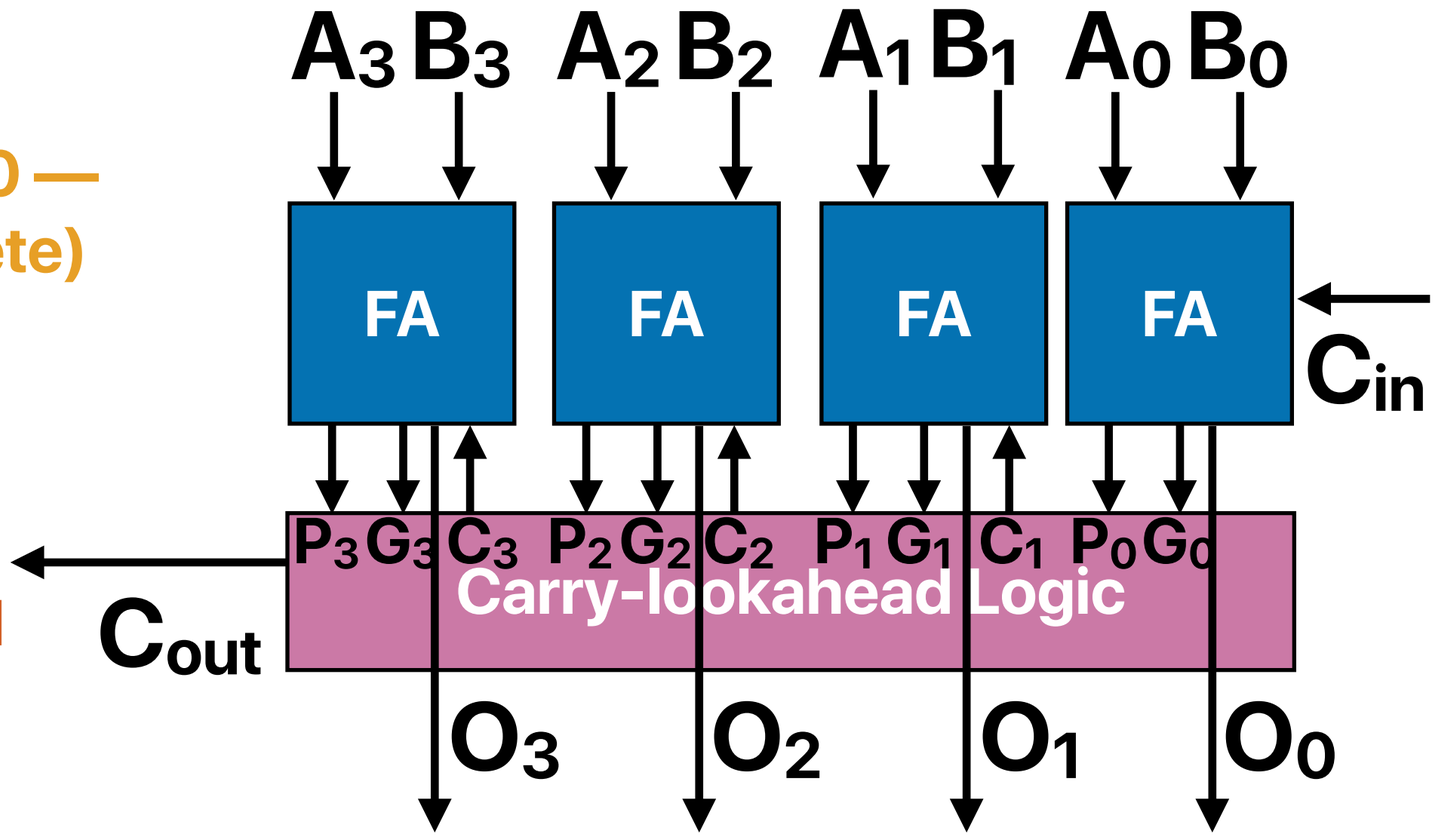
- Uses logic to quickly pre-compute the carry for each digit

Input			Output	
A	B	Cin	Out	Cout
0	0	0	**0**	**0**
0	0	1	**1**	**0**
0	1	0	**1**	**0**
0	1	1	**0**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**1**
1	1	0	**0**	**1**
1	1	1	**1**	**1**

Both A, B are 0 — no carry (Delete)

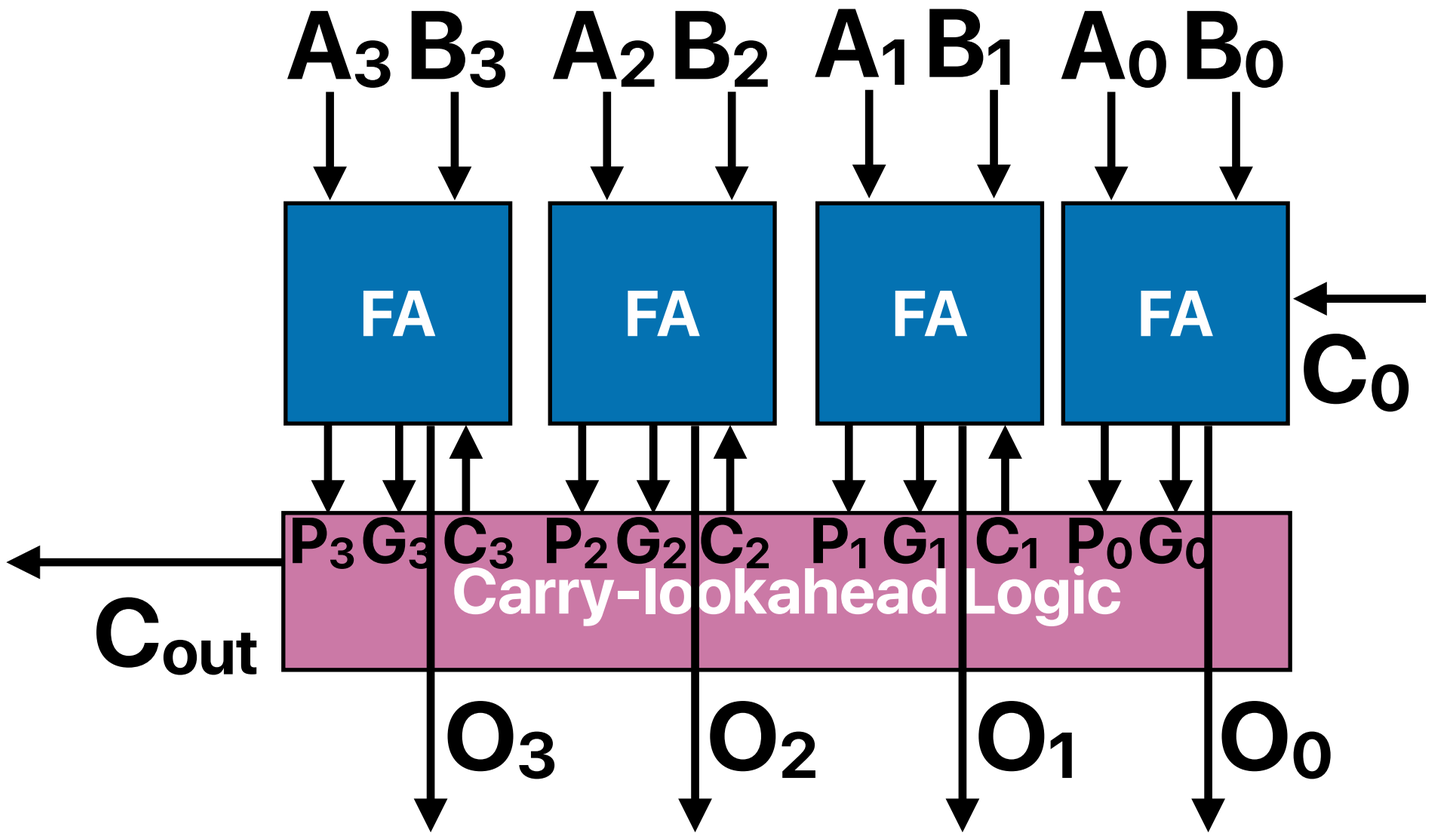
Needs to wait Cin (Propagate)

Both A, B are 1 — must carry (Generate)



CLA (cont.)

- All "G" and "P" are immediately available (only need to look over Ai and Bi), but "c" are not (except the c0).



$$G_i = A_i B_i$$

$$P_i = A_i \text{ XOR } B_i$$

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0)$$
$$= G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 C_2$$
$$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

$$C_4 = G_3 + P_3 C_3$$
$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$$

Poll close in 1:30

CLA's gate delay

- What's the gate-delay of a 4-bit CLA?
  A. 2
  B. 4
  C. 6
  D. 8
  E. 10

$$G_i = A_i B_i$$

$$P_i = A_i \text{ XOR } B_i$$

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0)$$
$$= G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 C_2$$
$$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

$$C_4 = G_3 + P_3 C_3$$
$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$$
$$+ P_3 P_2 P_1 P_0 C_0$$

CLA's gate delay

- What's the gate-delay of a 4-bit CLA?

A. 2

B. 4

C. 6

D. 8

E. 10

$$G_i = A_i B_i$$

$$P_i = A_i \text{ XOR } B_i$$

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0)$$
$$= G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 C_2$$
$$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

$$C_4 = G_3 + P_3 C_3$$
$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$$
$$+ P_3 P_2 P_1 P_0 C_0$$

Poll close in 1:30

CLA's size

- How many transistors do we need to implement a 4-bit CLA logic?
  A. 38
  B. 64
  C. 88
  D. 116
  E. 128

$$G_i = A_i B_i$$

$$P_i = A_i \text{ XOR } B_i$$

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0)$$
$$= G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 C_2$$
$$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

$$C_4 = G_3 + P_3 C_3$$
$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$$

CLA's size

- How many transistors do we need to implement a 4-bit CLA logic?
  - A. 38
  - B. 64
  - C. 88
  - D. 116
  - E. 128

$G_i = A_i B_i$

$P_i = A_i \text{ XOR } B_i$

$C_1 = G_0 + P_0 C_0$ 4 + 4 = 8

$C_2 = G_1 + P_1 C_1 = G_1 + P_1 (G_0 + P_0 C_0)$

$= G_1 + P_1 G_0 + P_1 P_0 C_0$

4 + 6 + 6 = 16

$C_3 = G_2 + P_2 C_2$

$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$

$C_4 = G_3 + P_3 C_3$ 4 + 6 + 8 + 8 =26

$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0$

$+ P_3 P_2 P_1 P_0 C_0$

4 + 6 + 8 + 10 + 10 = 38

CLA v.s. Carry-ripple

- Size:
  - 32-bit CLA with 4-bit CLAs — requires 8 of 4-bit CLA
    - Each requires 116 for the CLA 4*(4*6+8) for the A+B — 244 gates
    - 1952 transistors
  - 32-bit CRA
    - 1600 transistors **Win!**
- Delay
  - 32-bit CLA with 8 4-bit CLAs
    - 2 gates * 8 = 16 **Win!**
  - 32-bit CRA
    - 64 gates

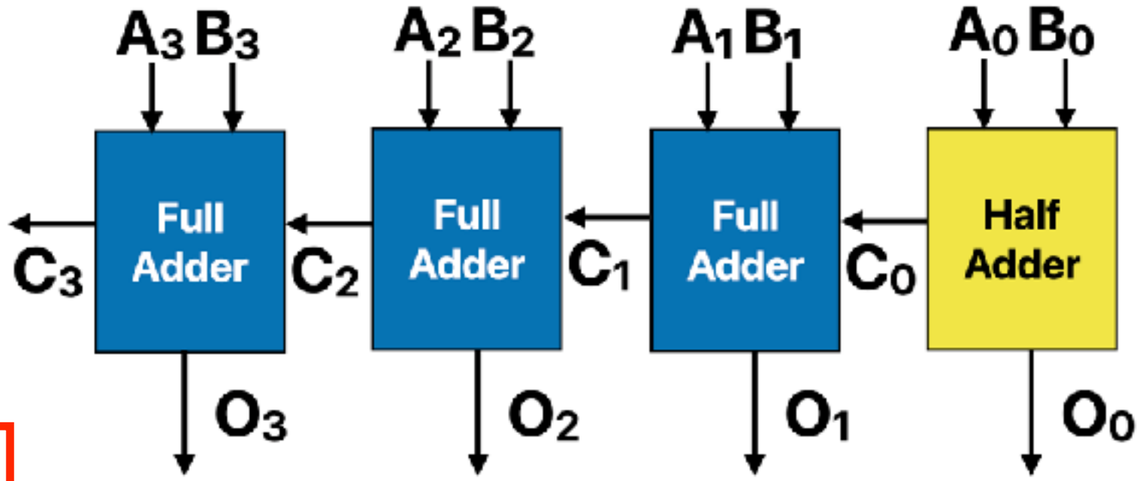
Area-Delay Trade-off!

Recap: If we want to support subtraction?

- If we would like to extend the 4-bit adder that we've built before to support "A-B" with 2's complement, how many of the followings should we add at least?
  - ① Provide an option to use bitwise NOT A
  - ② Provide an option to use bitwise NOT B
  - ③ Provide an option to use bitwise A XOR B
  - ④ Provide an option to add 0 to the input of the half adder
  - ⑤ Provide an option to add 1 to the input of the half adder

  A. 1
  B. 2
  C. 3
  D. 4
  E. 5

How to provide this option

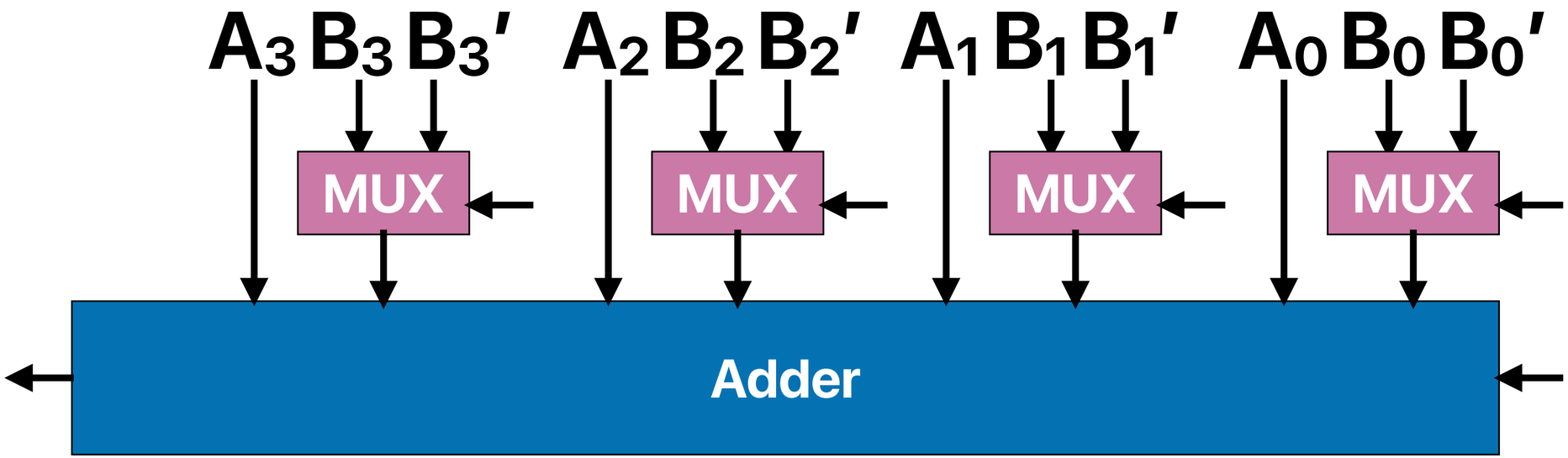


To "NOT" or not to "NOT", that's the question!

Multiplexer

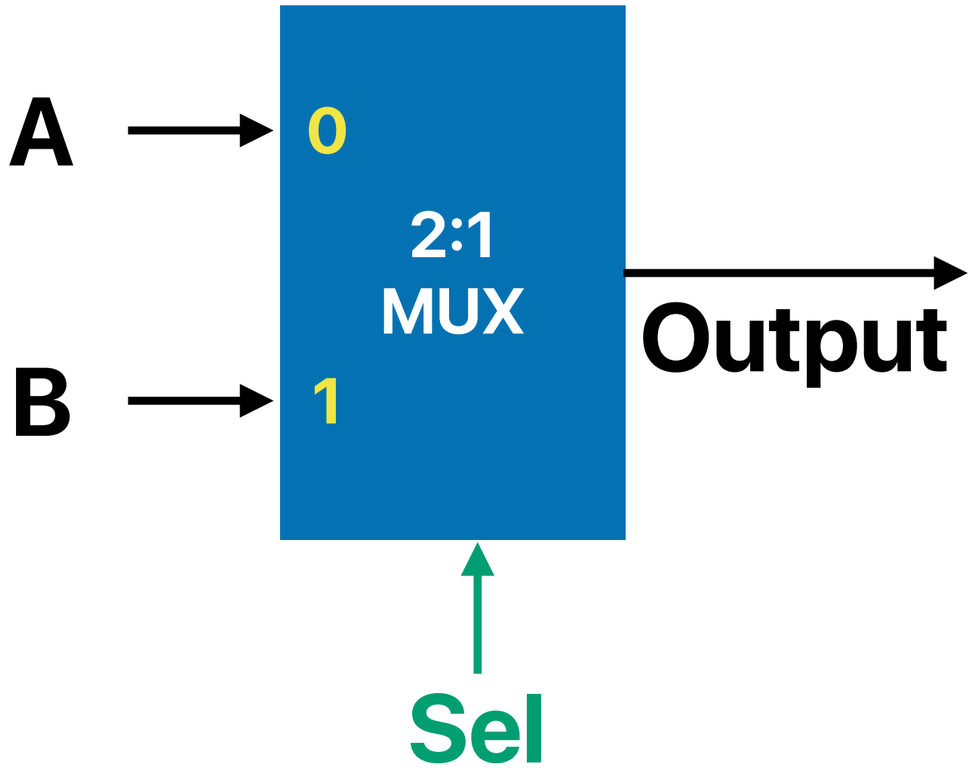
Multiplexer

- Problem — you have multiple possible inputs and you only want to use one of them
  - **N**-to-**M** MUX mean a MUX with **N** inputs, **M** outputs.
- Solution — you need a multiplexer (MUX) to control the output



Let's start with a 2-to-1 MUX

- The MUX has two input ports — numbered as 0 and 1
- To select from two inputs, you need a 1-bit control/select signal to indicate the desired input port



Input			Output
A	B	Sel	
0	0	0	**0**
0	1	0	**0**
1	0	0	**1**
1	1	0	**1**
0	0	1	**0**
0	1	1	**1**
1	0	1	**0**
1	1	1	**1**

Use K-Map

Input			Output
A	B	Sel	
0	0	0	**0**
0	1	0	**0**
1	0	0	**1**
1	1	0	**1**
0	0	1	**0**
0	1	1	**1**
1	0	1	**0**
1	1	1	**1**

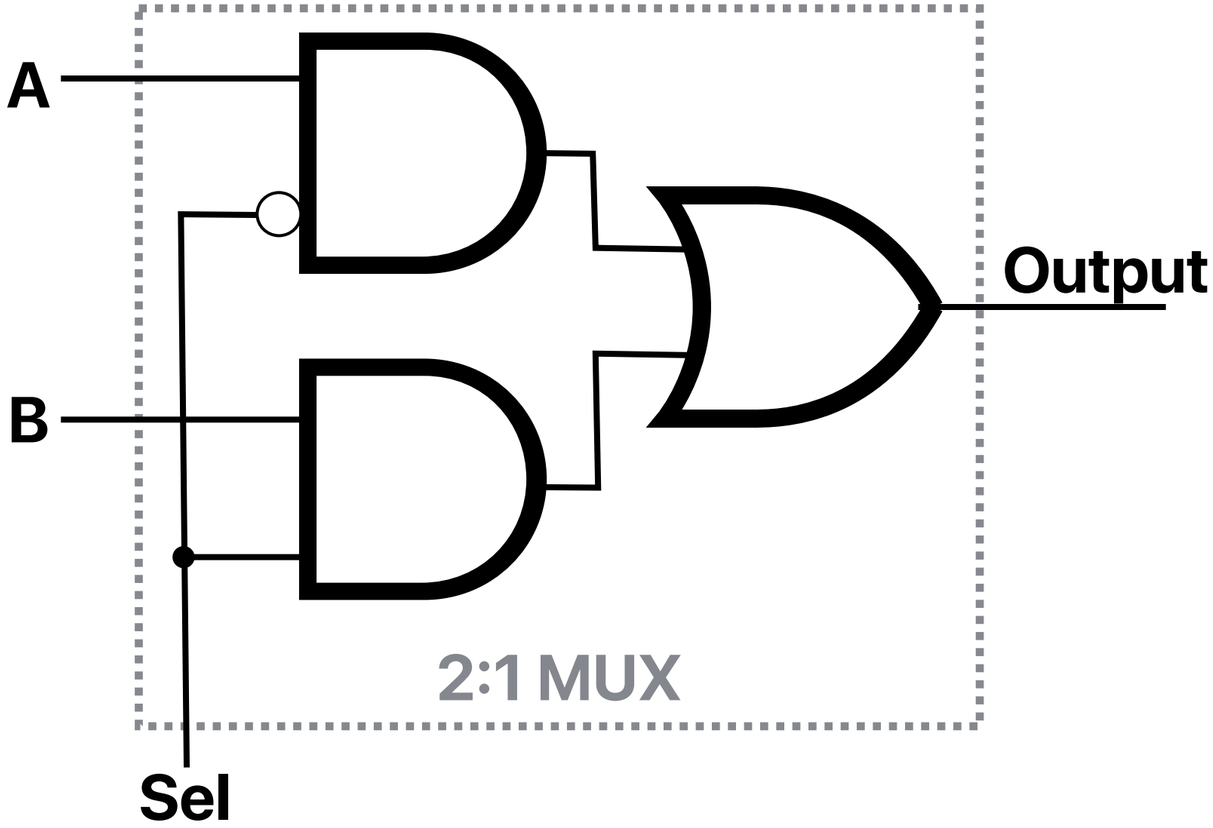
Sel′ means output A
Sel means output B

Output = ASel′ + BSel

(A, B) Sel	A′B′ 0,0	A′B 0,1	AB 1,1	AB′ 1,0
Sel′	0	0	1	1
Sel	0	1	1	0

ASel′

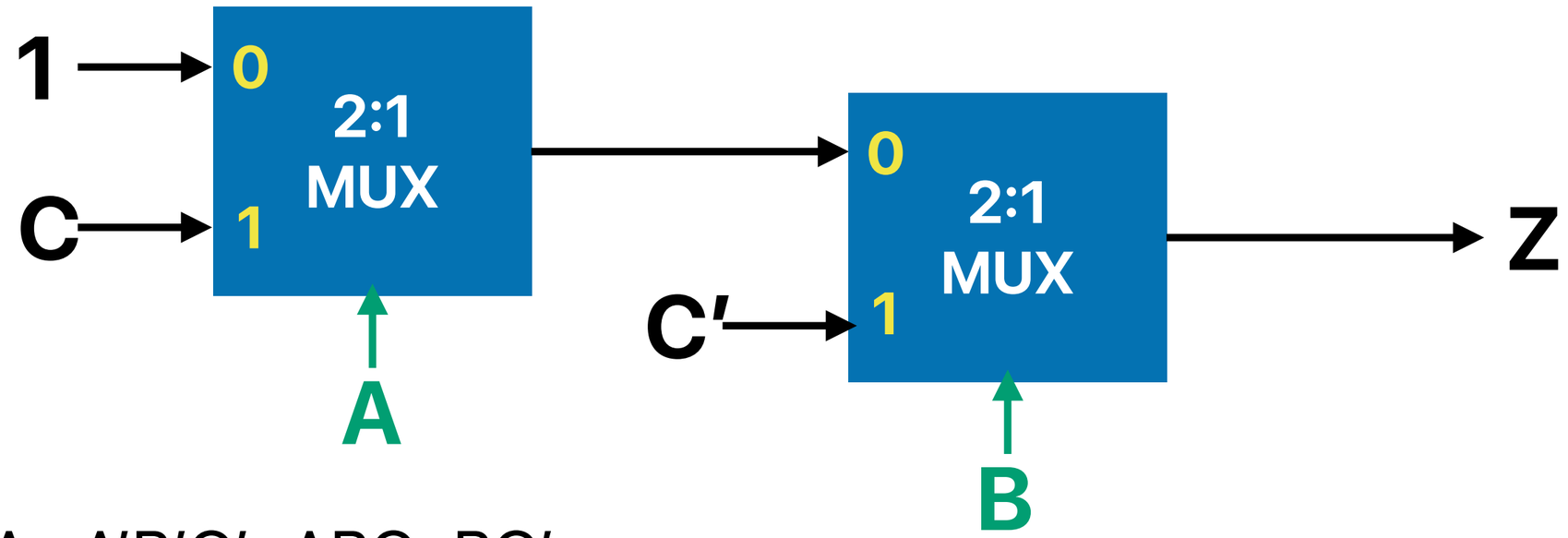
BSel



Poll close in 1:30

Cascading MUXes

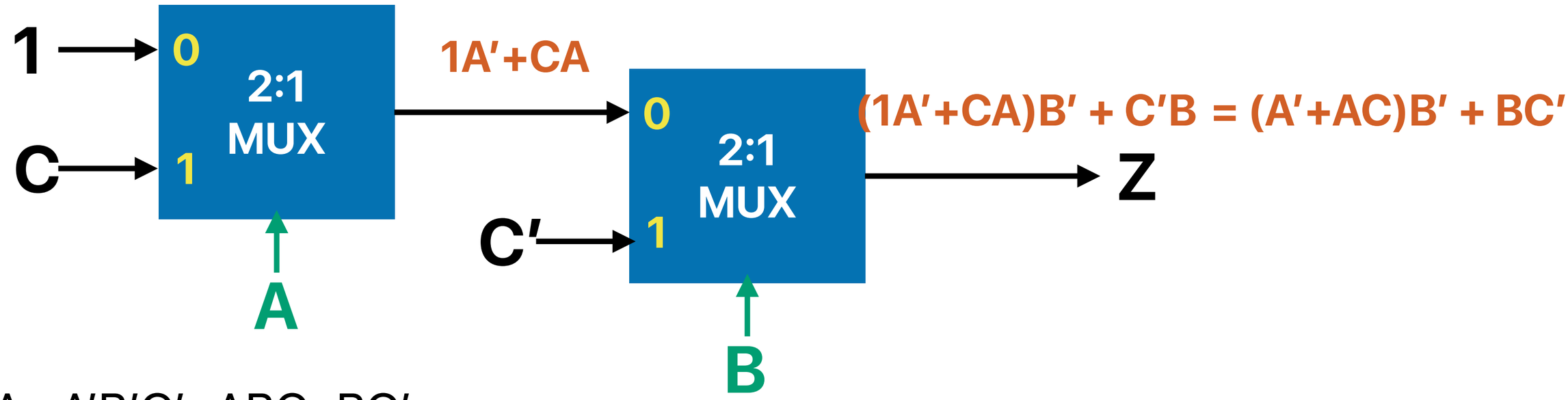
- Function Z(A,B,C) implemented by 2:1 Muxes above is:



A. A′B′C′+ABC+BC′
B. (A′+AC)B+B′C′
C. A′B′+B′C+BC′
D. (A′+AC)B′+BC′

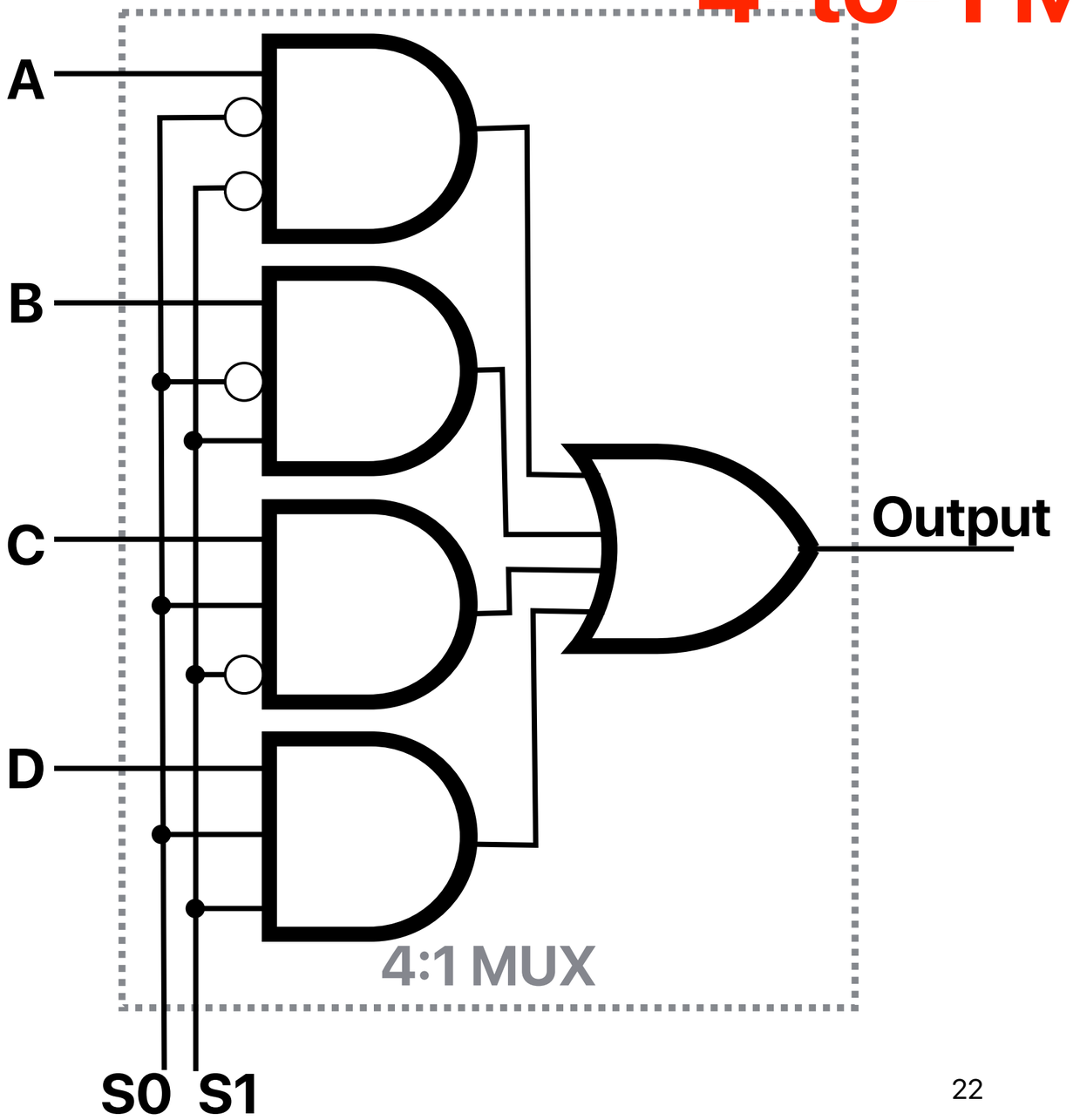
Cascading MUXes

- Function Z(A,B,C) implemented by 2:1 Muxes above is:



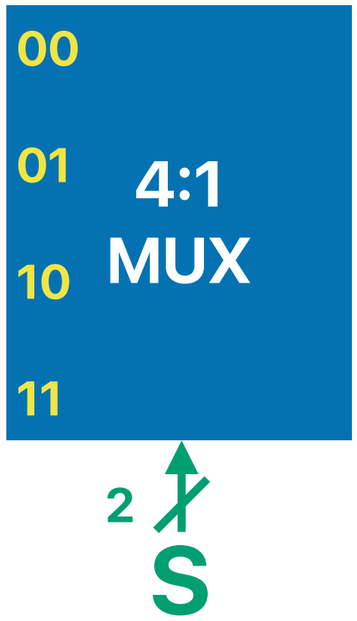
A. A′B′C′+ABC+BC′
B. (A′+AC)B+B′C′
C. A′B′+B′C+BC′
D. (A′+AC)B′+BC′

4-to-1 MUX



S0==0 && S1==0 output A
S0==0 && S1==1 output B
S0==1 && S1==0 output C
S0==1 && S1==1 output D

Output = AS0′S1′ + BS0′S1 + CS0S1′ + DS0S1



Poll close in 1:30

Gate delay of 8:1 MUX

- What's the estimated gate delay of an 8:1 MUX?
  A. 1
  B. 2
  C. 4
  D. 8
  E. 16

Gate delay of 8:1 MUX

- What's the estimated gate delay of an 8:1 MUX?

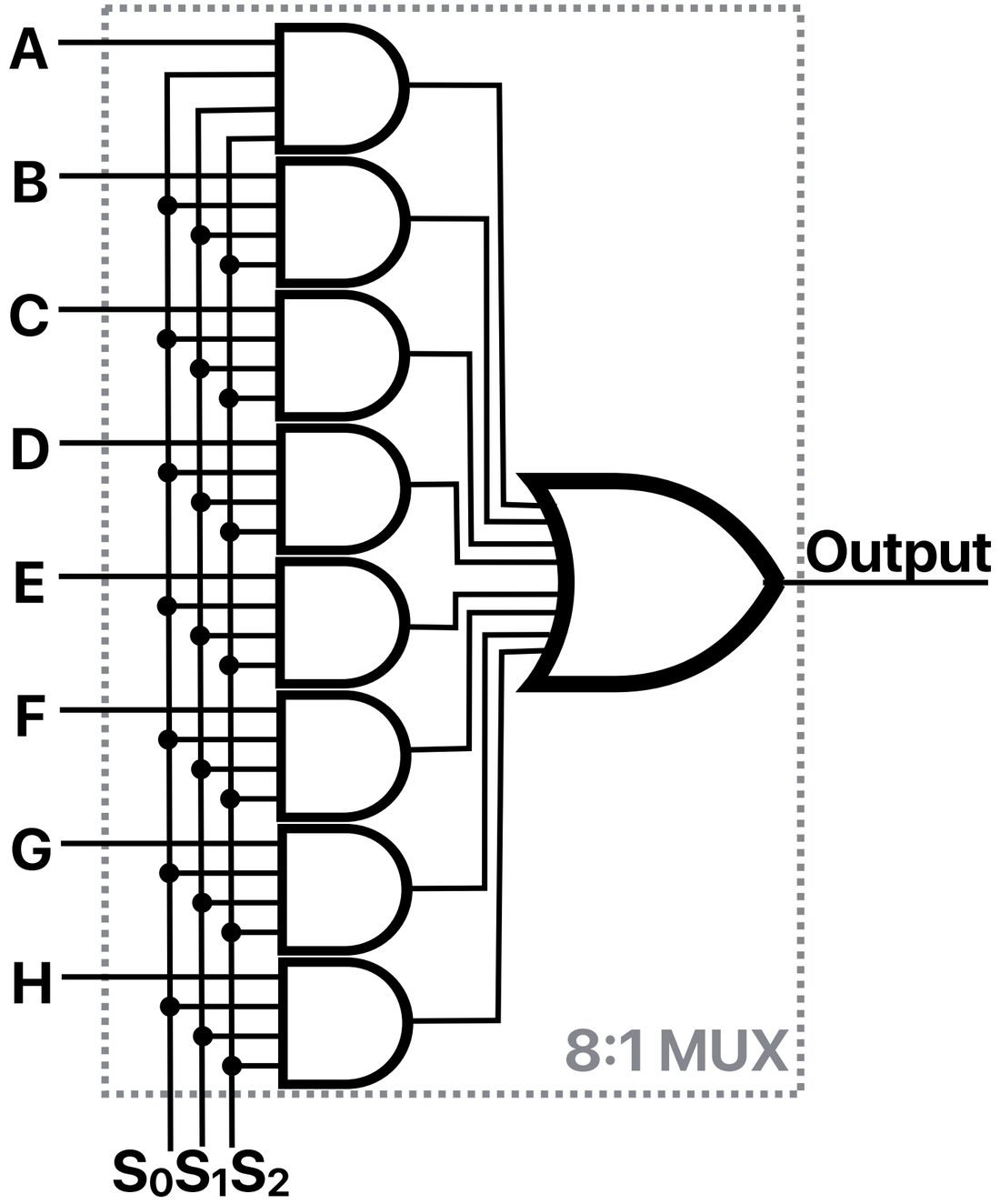
A. 1

B. 2

C. 4

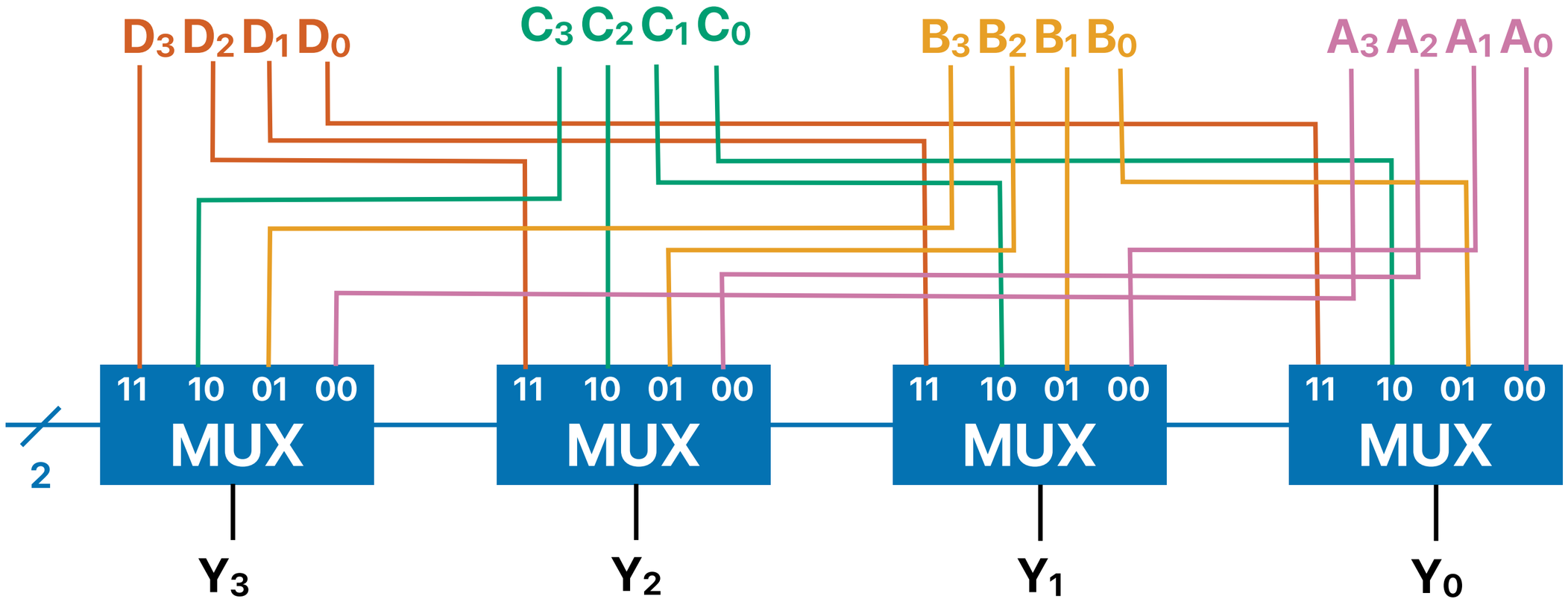
D. 8

E. 16



N-bit MUX

- What if we need to output an N-bit (say 4-bit) number from the input set?



Poll close in 1:30

How big is the 4-bit 4:1 MUX?

- How many estimated transistors are there in the 4-bit 4:1 MUX?
  A. 48
  B. 64
  C. 80
  D. 128
  E. 192

How big is the 4-bit 4:1 MUX?

- How many estimated transistors are there in the 4-bit 4:1 MUX?
  A. 48
  B. 64
  C. 80
  D. 128
  E. 192

4:1 MUX —
each AND gate would need 2 inputs for control and one for number
an OR gate collects 4 results from AND gates
— 4 3-input AND gates and one 4-input OR gate
— 4*6 + 8 = 32
We need 4 of these = 32*4 = 128

Shifters

Poll close in 1:30

What's after shift?

- Assume we have a data type that stores 8-bit unsigned integer (e.g., unsigned char in C). How many of the following C statements and their execution results are correct?

	Statement	C = ?
I	`c = 3; c = c >> 2;`	1
II	`c = 255; c = c << 2;`	252
III	`c = 256; c = c >> 2;`	64
IV	`c = 128; c = c << 1;`	1

A. 0
B. 1
C. 2
D. 3
E. 4

What's after shift?

- Assume we have a data type that stores 8-bit unsigned integer (e.g., unsigned char in C). How many of the following C statements and their execution results are correct?

	Statement	C = ?
I	`c = 3; c = c >> 2;`	**0**
II	`c = 255; c = c << 2;`	252
III	`c = 256; c = c >> 2;`	**0**
IV	`c = 128; c = c << 1;`	**0**

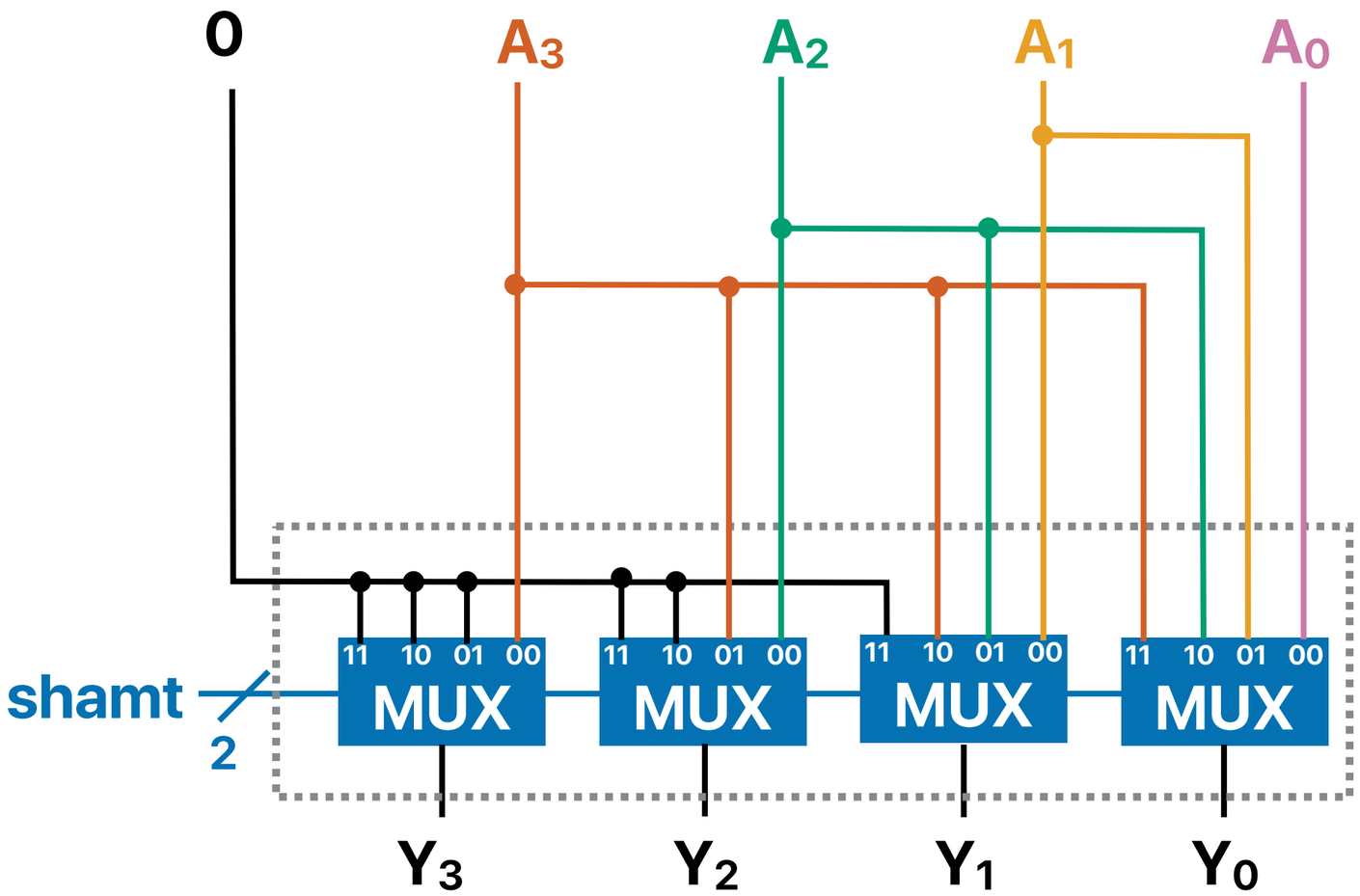
A. 0
B. 1
C. 2
D. 3
E. 4

Shifters

- Logical shifter: shifts value to left or right and fills empty spaces with 0's
  - 11001 >> 2 = 00110
  - 11001 << 2 = 00100
- Arithmetic shifter: same as logical shifter, but on right shift, fills empty spaces with the old most significant bit
  - Ex: 11001 >>> 2 = 11110
  - Ex: 11001 <<< 2 = 00100
- Rotator: rotates bits in a circle, such that bits shifted off one end are shifted into the other end
  - Ex: 11001 ROR 2 = 01110
  - Ex: 11001 ROL 2 = 00111

https://en.wikipedia.org/wiki/Circular_shift

Shift "Right"



Example:
if S = 11
then
Y3 = 0
Y2 = 0
Y1 = 0
Y0 = A3

Example:
if S = 10
then
Y3 = 0
Y2 = 0
Y1 = A3
Y0 = A2

Example:
if S = 01
then
Y3 = 0
Y2 = A3
Y1 = A2
Y0 = A1

The "chain" of multiplexers determines how many bits to shift

Based on the value of the selection input (shamt = shift amount)

Announcement

- Assignment 2 due 4/23
  - Submit on zyBooks.com directly — all challenge questions 2.3-3.5
- Reading quiz 5 due 4/28 **BEFORE** the lecture
  - Under iLearn > reading quizzes
- Lab 3 due 4/30
  - Watch the video and read the instruction BEFORE your session
  - There are links on both course webpage and iLearn lab section
  - Submit through iLearn > Labs
- Midterm on 5/7 during the lecture time, access through iLearn — no late submission is allowed — make sure you will be able to take that at the time
- Check your grades in iLearn

Electrical
Computer Science
Engineering

120A

つづく