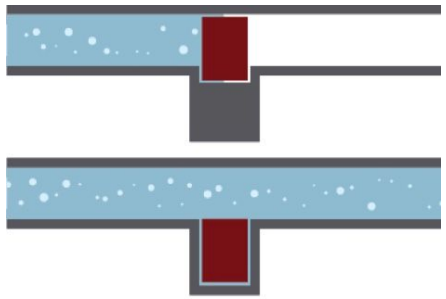
Gates and Boolean Logic

CS 3410: Computer System Organization and Programming

Lecture 1: Representing Numbers

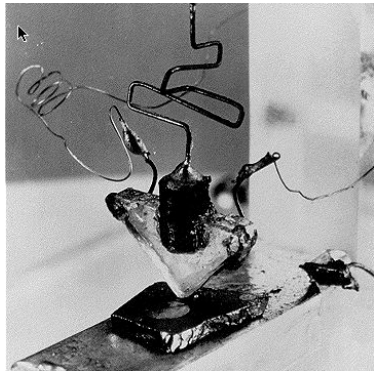


Analog

Water Flow (l/min)

Digital: 0 / 1

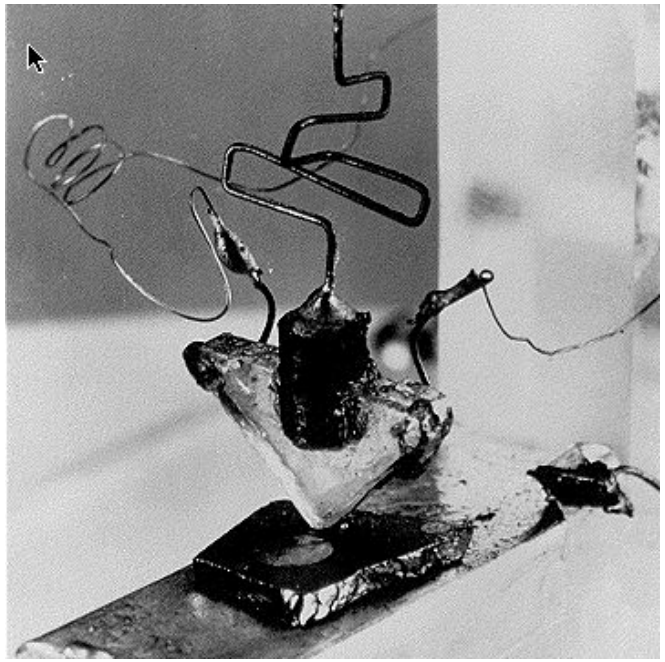
No Water / Water Flowing



Voltage (V)

Low / High Voltage

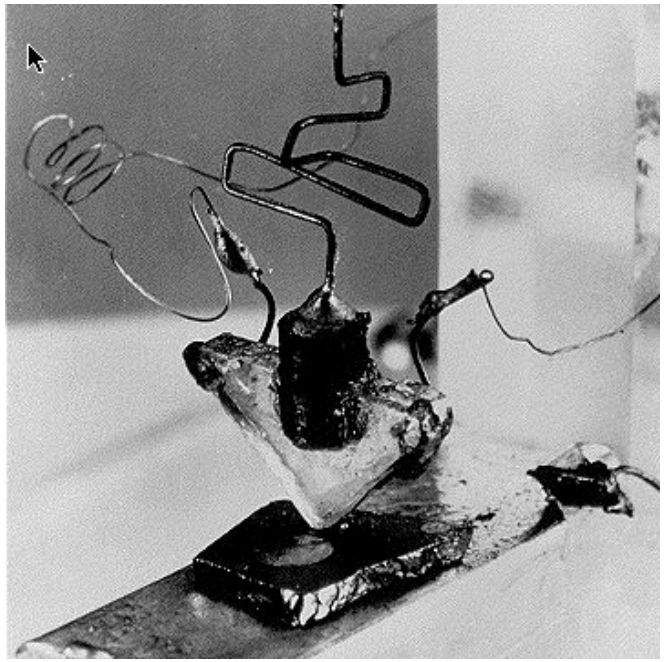
Then and Now

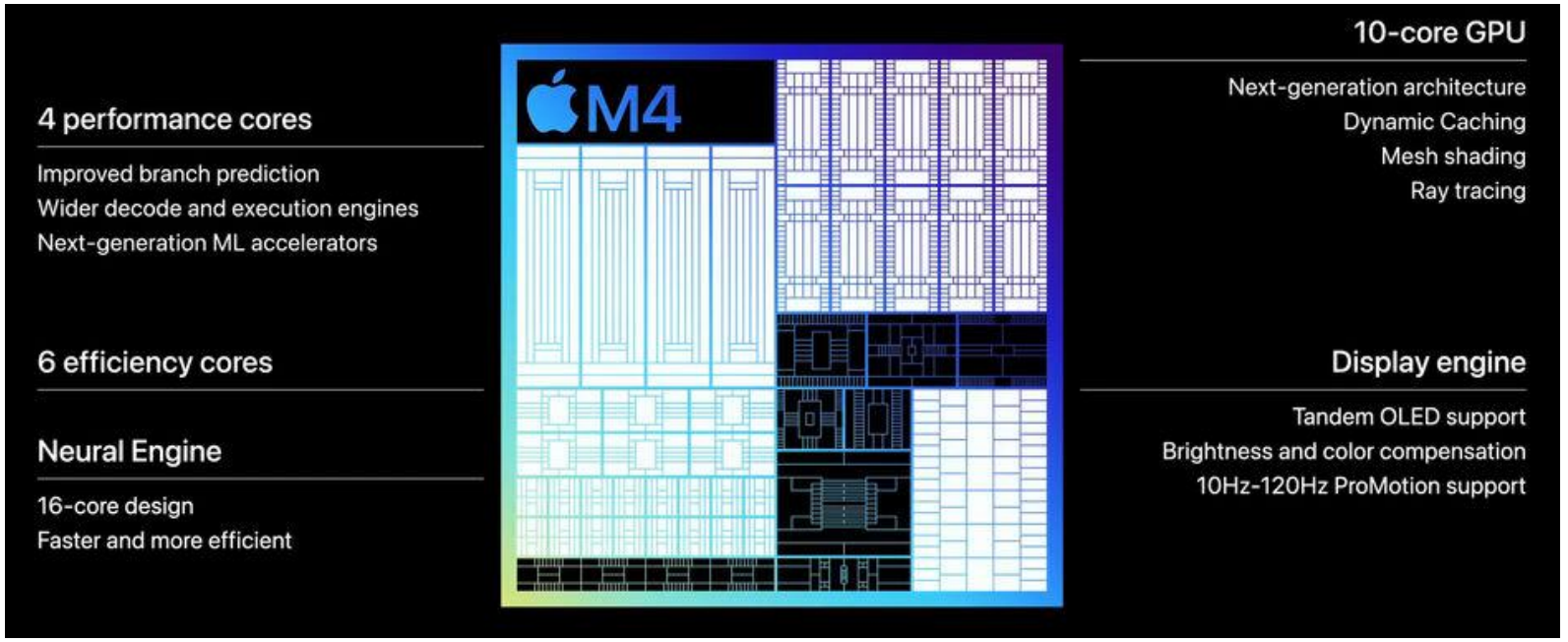


The first transistor

- One workbench at AT&T Bell Labs
- 1947
- Bardeen, Brattain, and Shockley

Then and Now





https://en.wikipedia.org/wiki/Apple_M4
https://en.wikipedia.org/wiki/Transistor_count

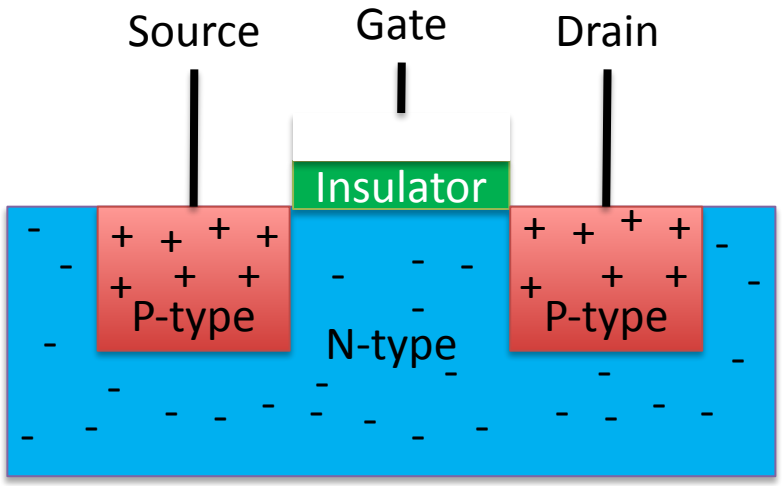
The first transistor

- One workbench at AT&T Bell Labs
- 1947
- Bardeen, Brattain, and Shockley

Apple M4

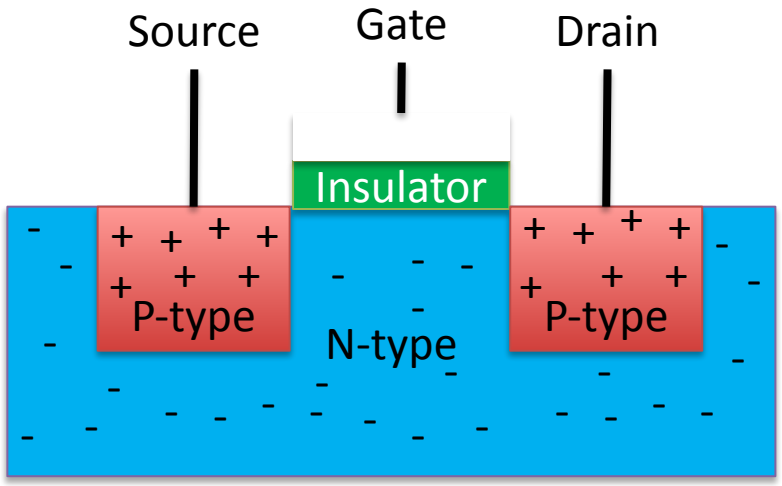
- 28 billion transistors, 3nm
- 177 square millimeters
- 4x-10x performance, 4x-6x efficiency, 8x-40x GPU, 16x Neural processing cores

Transistors 101

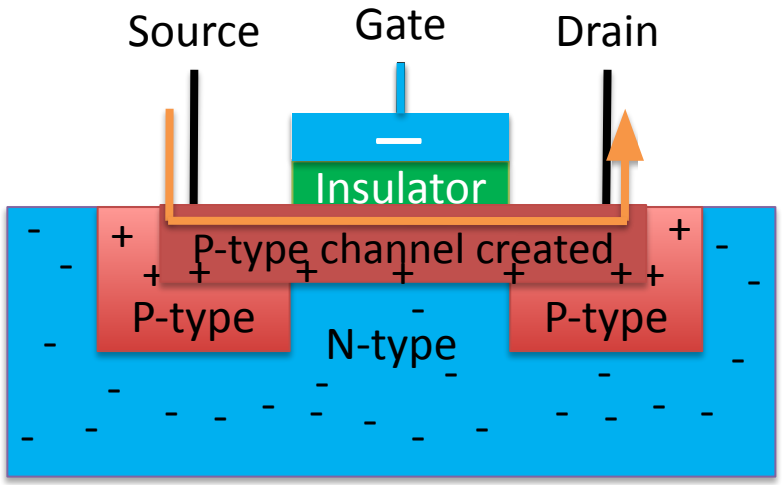


P-Transistor **Off**

Transistors 101



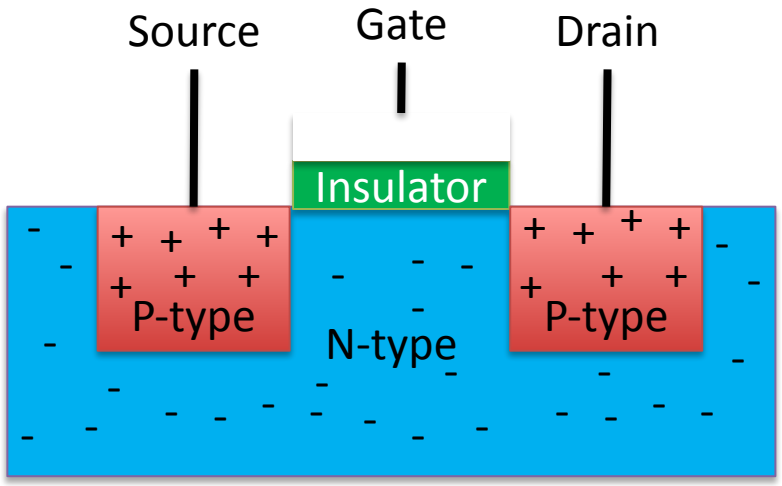
P-Transistor **Off**



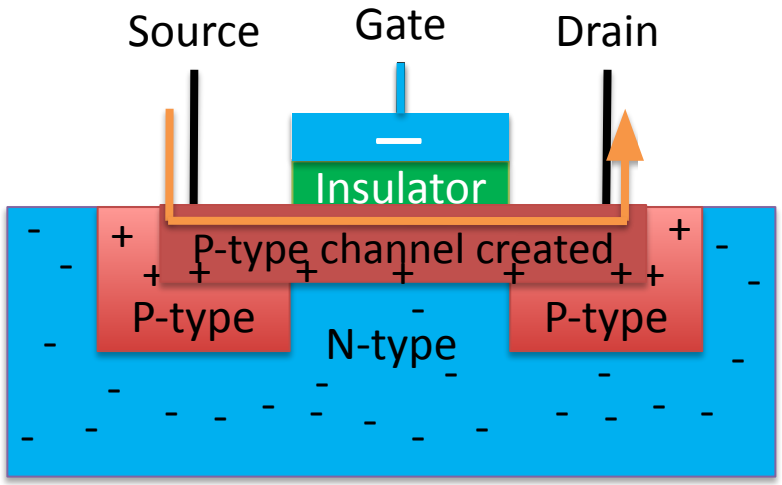
P-Transistor **On**

P-Transistor: **negative** charge (0!) creates conductive channel

Transistors 101



P-Transistor **Off**



P-Transistor **On**

P-Transistor: **negative** charge (0!) creates conductive channel

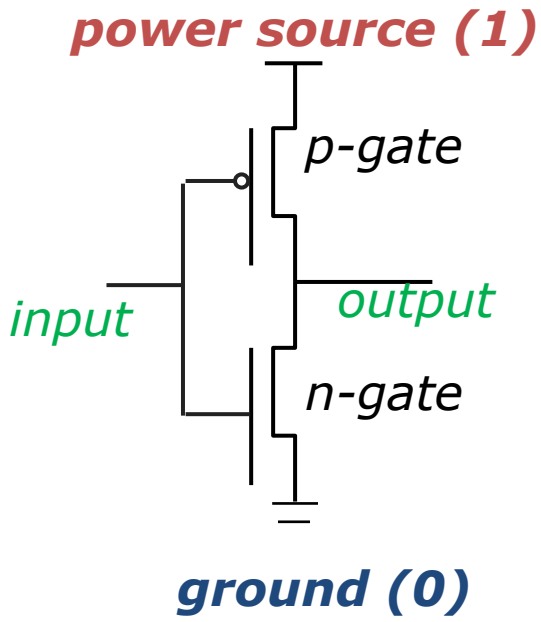
N-Transistor: **positive** charge (1!) creates conductive channel

2-Transistor Combination

- CMOS: combine P- and N-Transistors

Truth Table

Input	Output

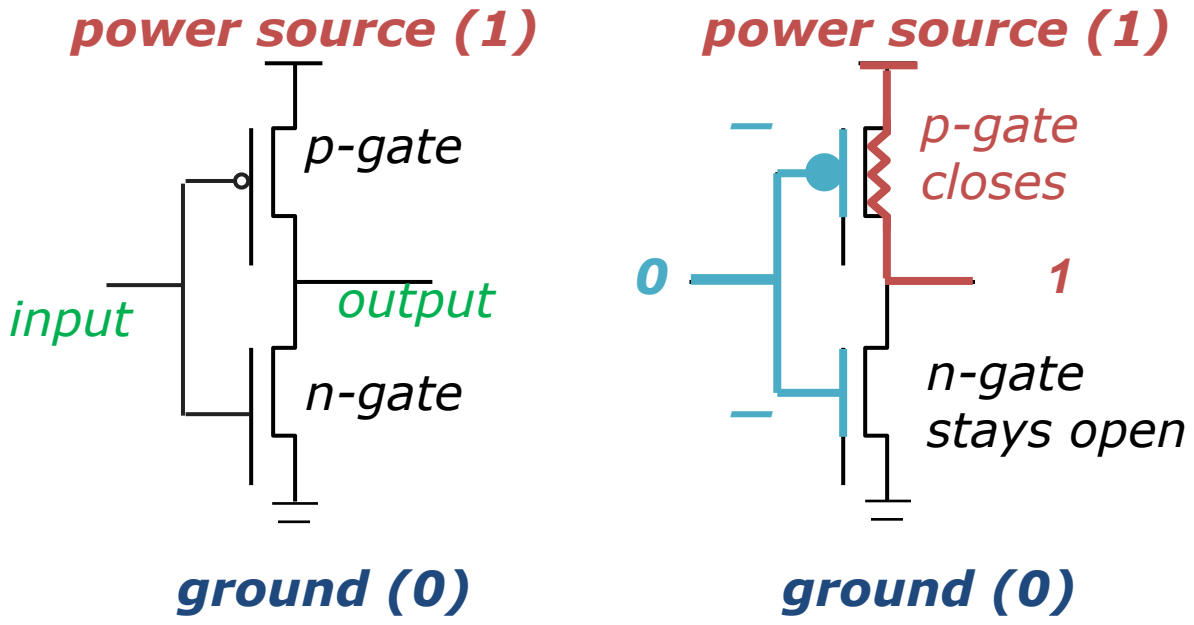


2-Transistor Combination

- CMOS: combine P- and N-Transistors

Truth Table

Input	Output
0	1

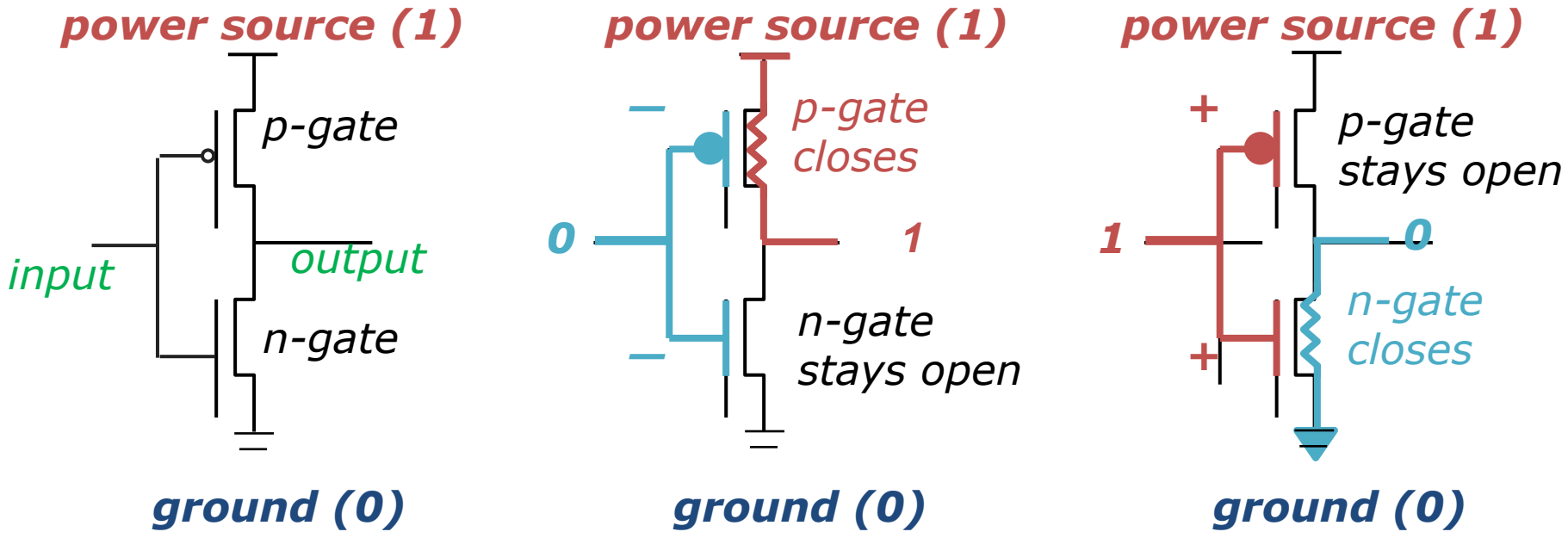


2-Transistor Combination

- CMOS: combine P- and N-Transistors

Truth Table

Input	Output
0	1
1	0

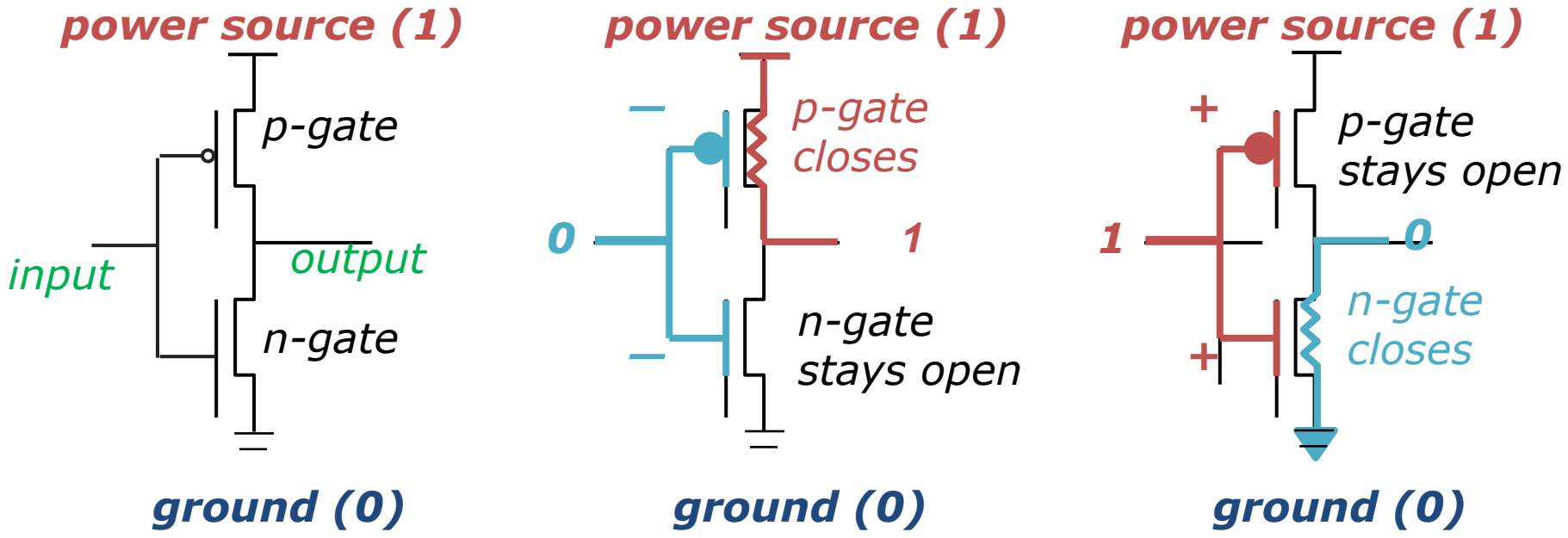


2-Transistor Combination

- CMOS: combine P- and N-Transistors
- f(a) = …?

Truth Table

Input	Output
0	1
1	0

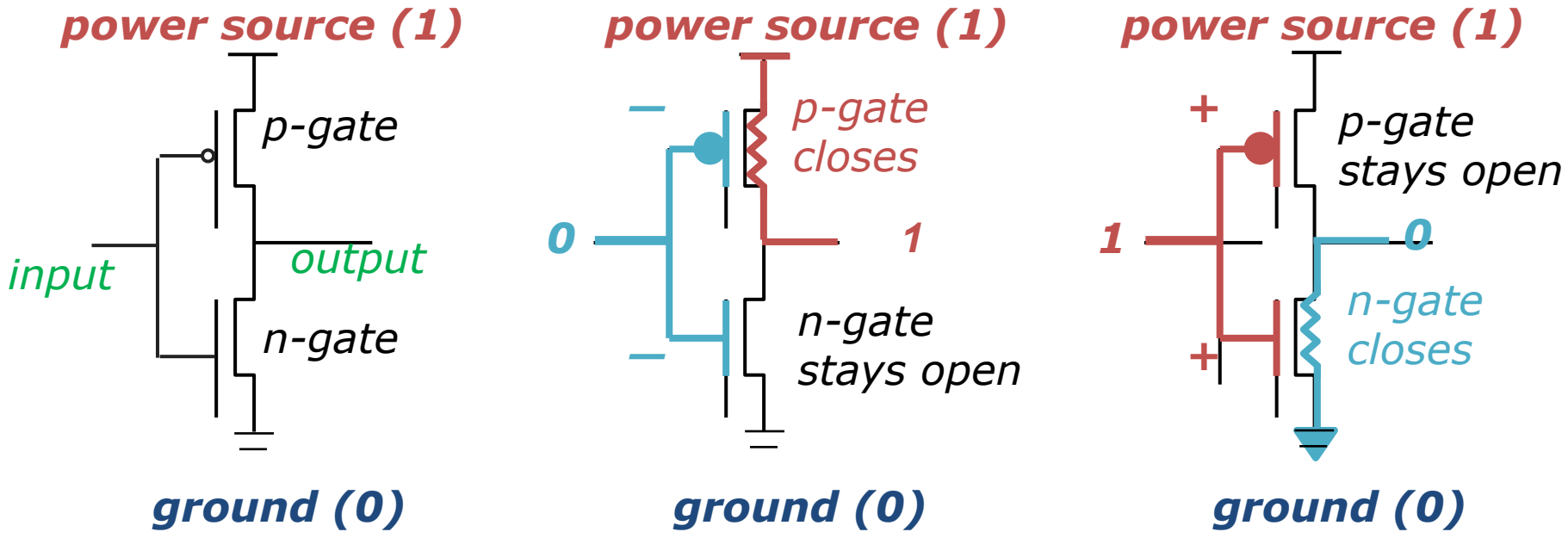


2-Transistor Combination

- CMOS: combine P- and N-Transistors
- f(a) = $\overline{a}$

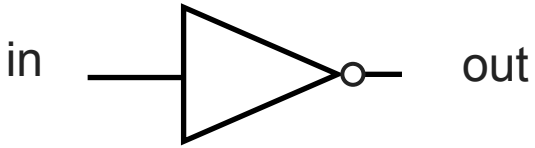
Truth Table

Input	Output
0	1
1	0



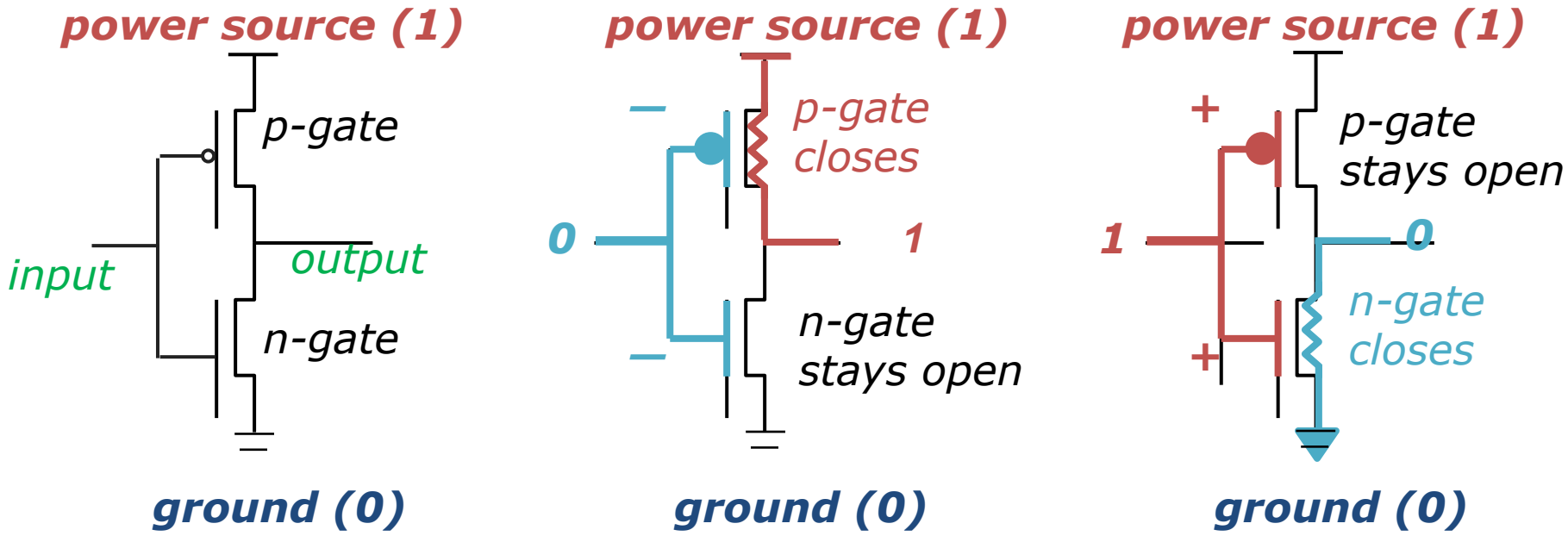
2-Transistor Combination

- CMOS: combine P- and N-Transistors
- $f(a) = \overline{a}$
- Inverter

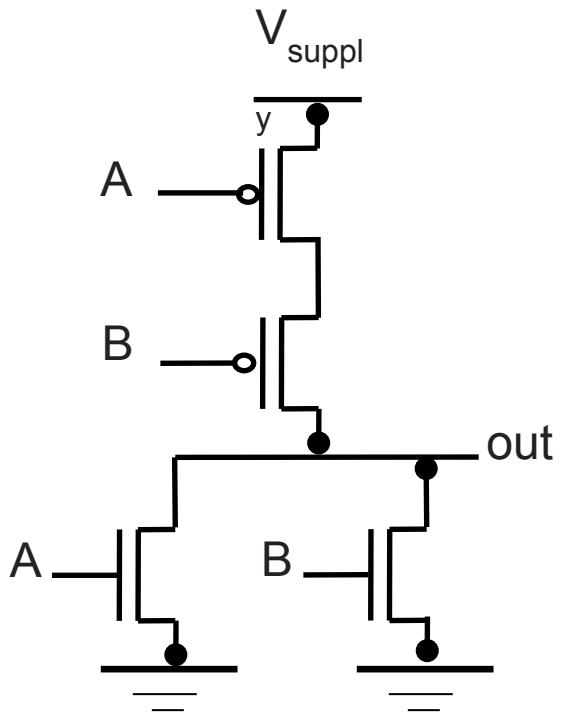


Truth Table

Input	Output
0	1
1	0



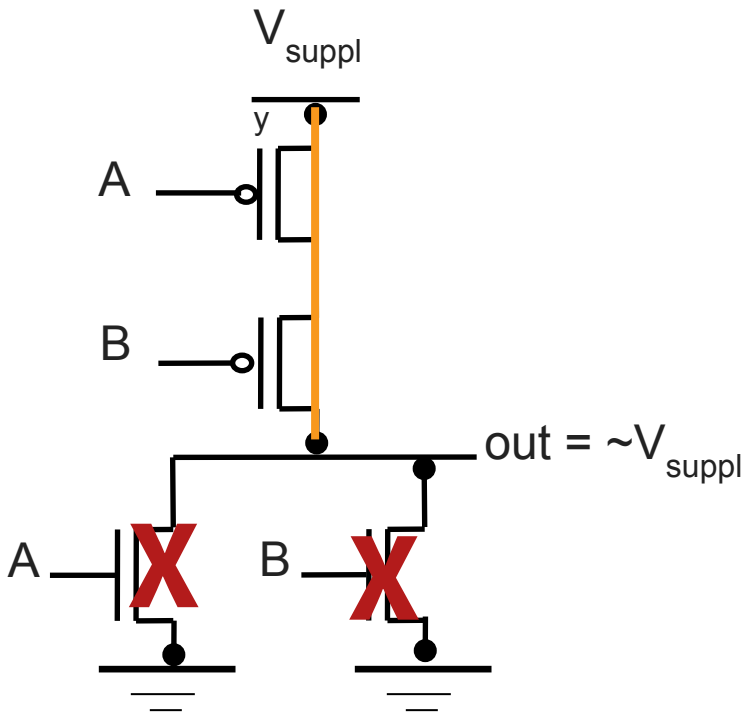
4-Transistor Combination



Truth Table

A	B	Output
0	0	
0	1	
1	0	
1	1	

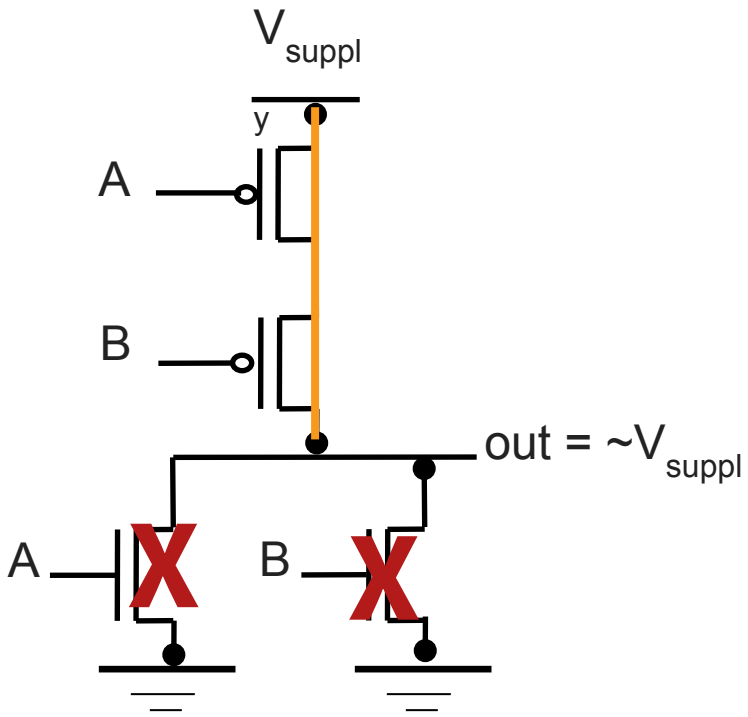
4-Transistor Combination



Truth Table

A	B	Output
0	0	1
0	1	
1	0	
1	1	

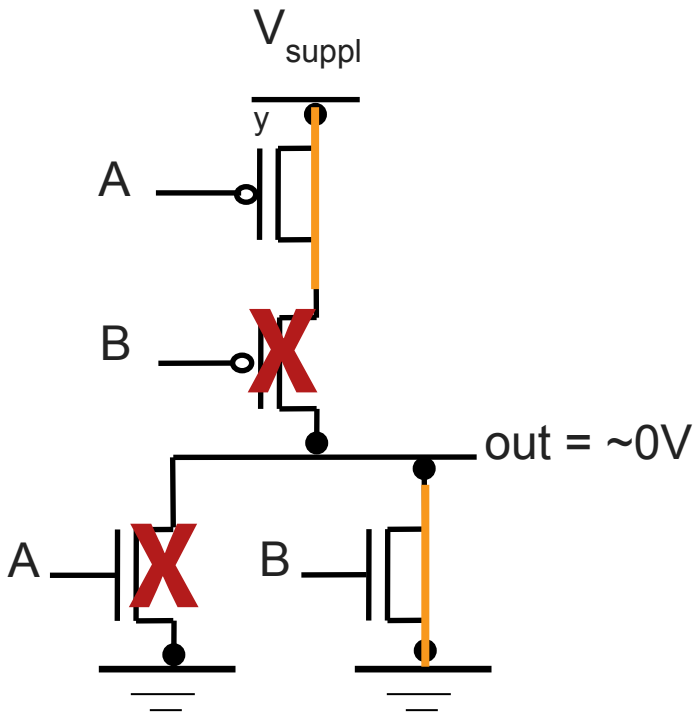
4-Transistor Combination



Truth Table

A	B	Output
0	0	1
0	1	
1	0	
1	1	

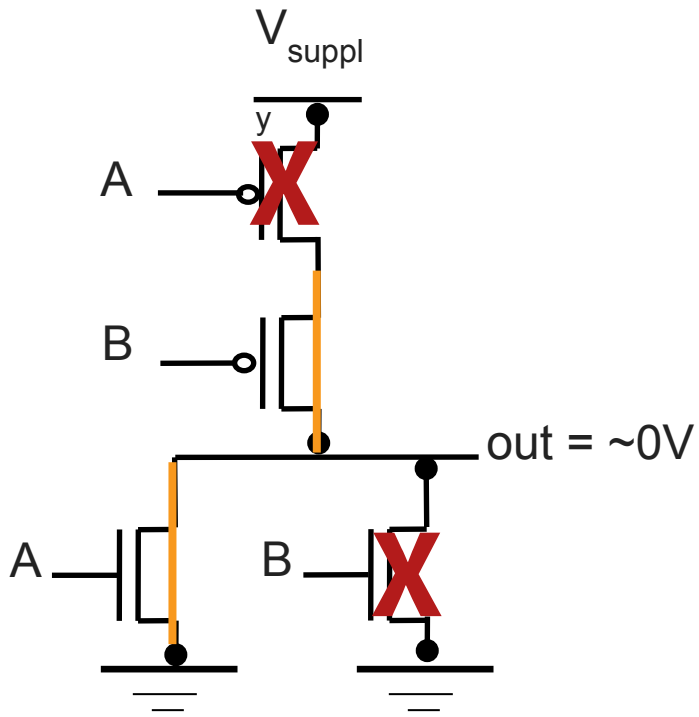
4-Transistor Combination



Truth Table

A	B	Output
0	0	1
0	1	0
1	0	
1	1	

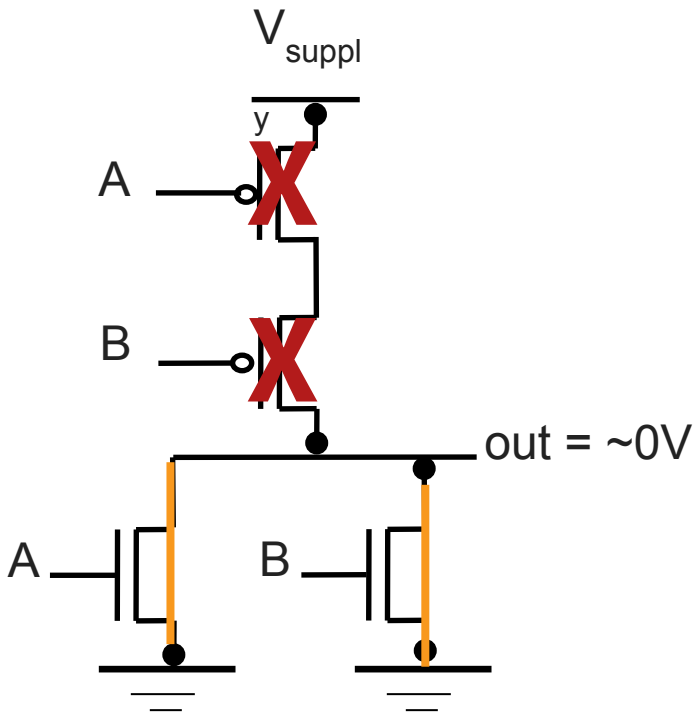
4-Transistor Combination



Truth Table

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	

4-Transistor Combination

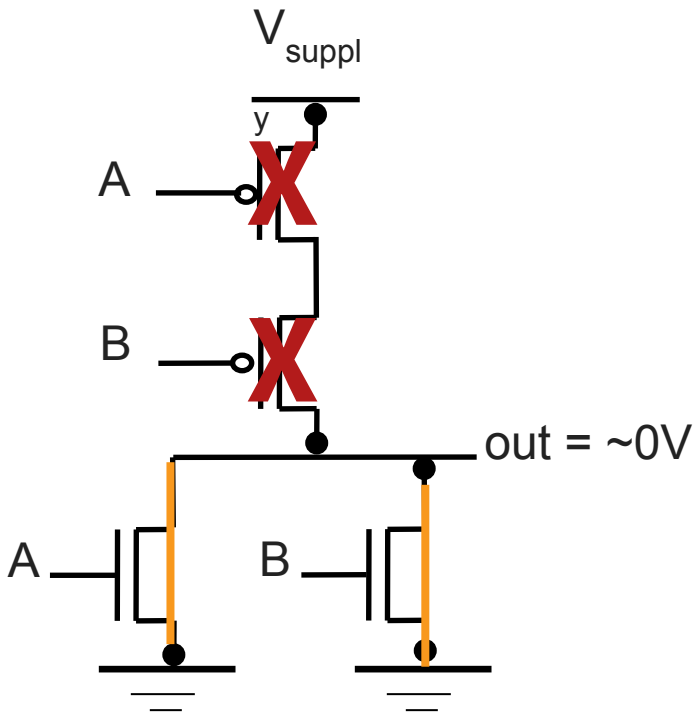


Truth Table

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

4-Transistor Combination

- What function is implemented?

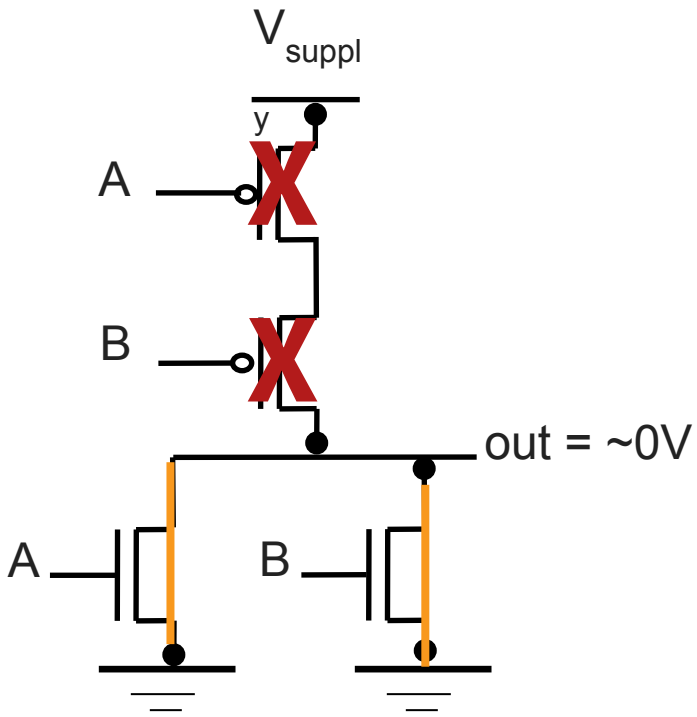


Truth Table

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

4-Transistor Combination

- What function is implemented?

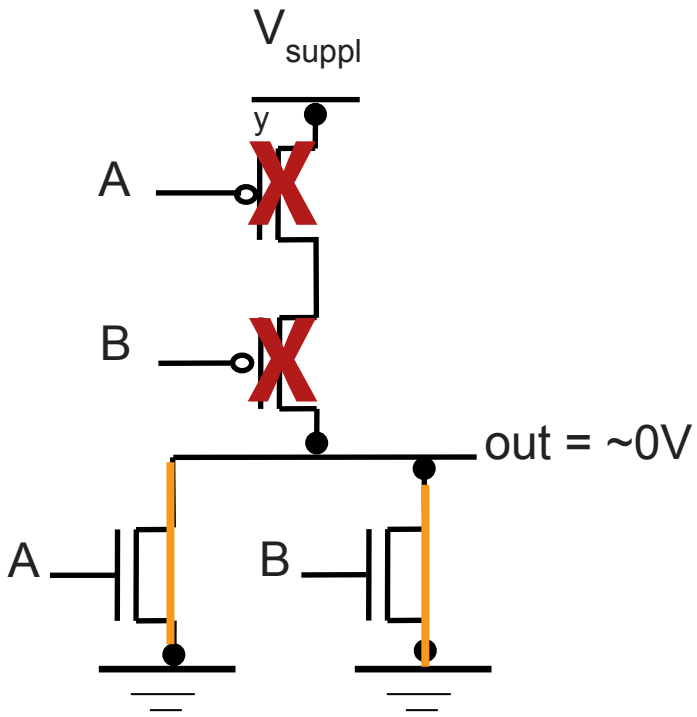


Truth Table

A	B	Out	¬Out
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

4-Transistor Combination

- $f(a, b) = \overline{a \vee b}$

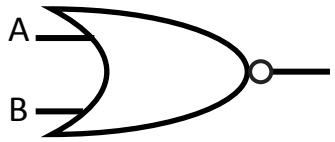


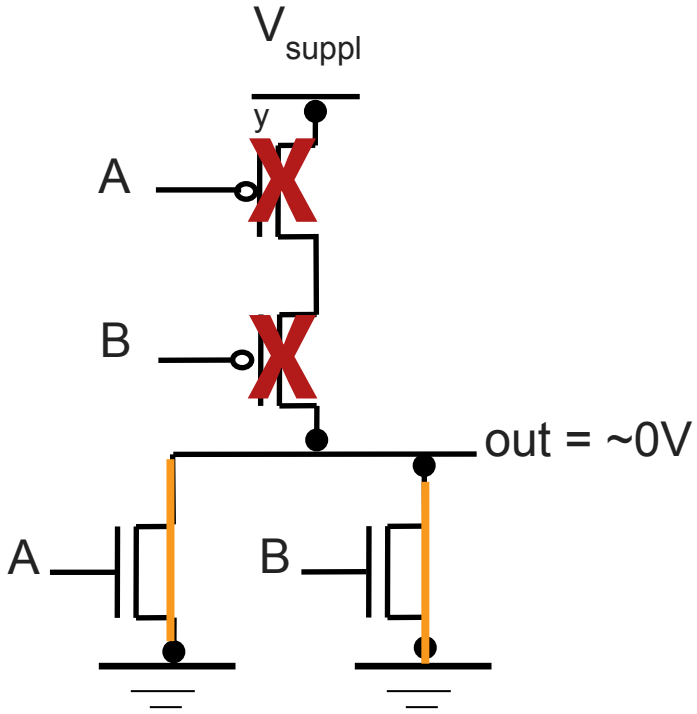
Truth Table

A	B	Out	¬Out
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

4-Transistor Combination

- $f(a, b) = \overline{a \vee b}$
- not or → NOR Gate

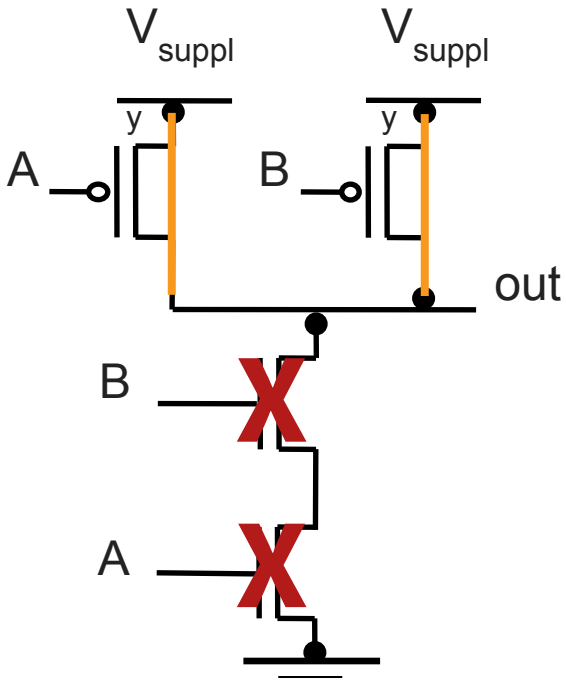




Truth Table

A	B	Out	¬Out
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

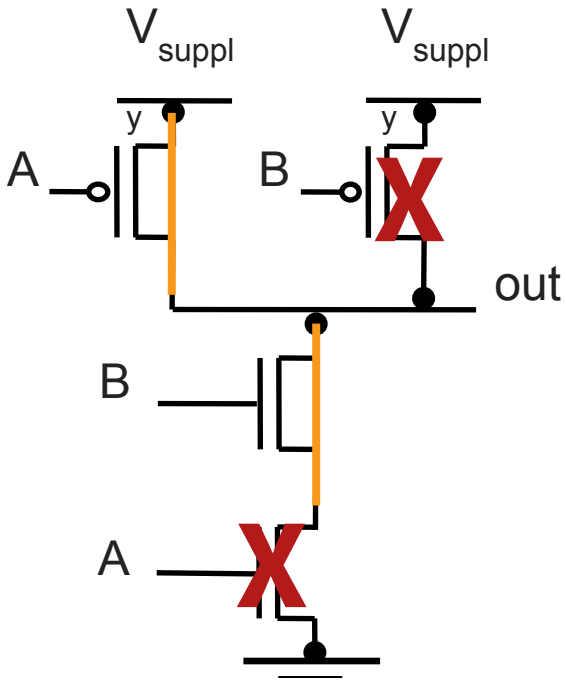
4-Transistor Combination



Truth Table

A	B	Output
0	0	1
0	1	
1	0	
1	1	

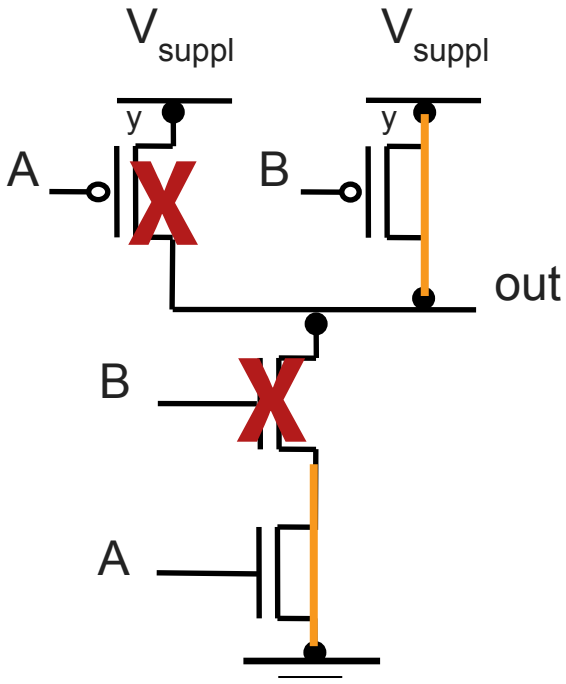
4-Transistor Combination



Truth Table

A	B	Output
0	0	1
0	1	1
1	0	
1	1	

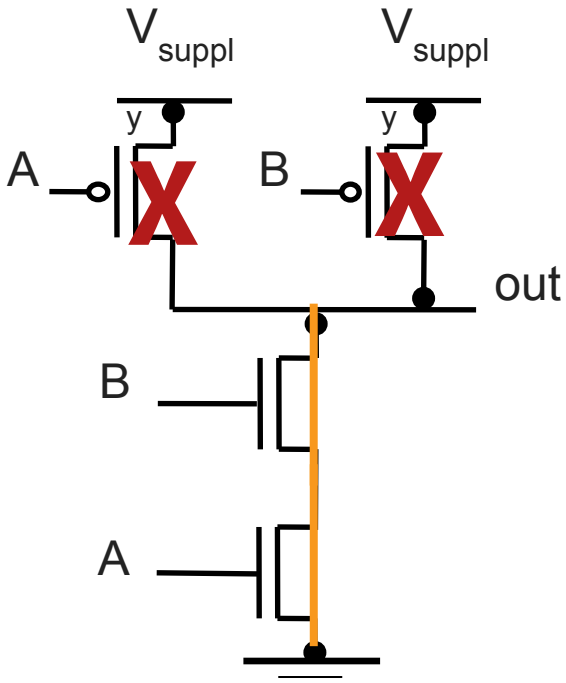
4-Transistor Combination



Truth Table

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	

4-Transistor Combination

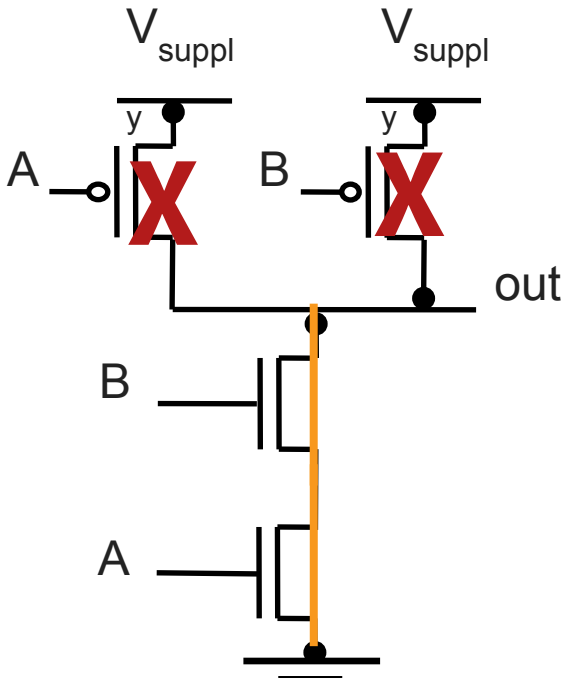


Truth Table

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

4-Transistor Combination

- What function is implemented?

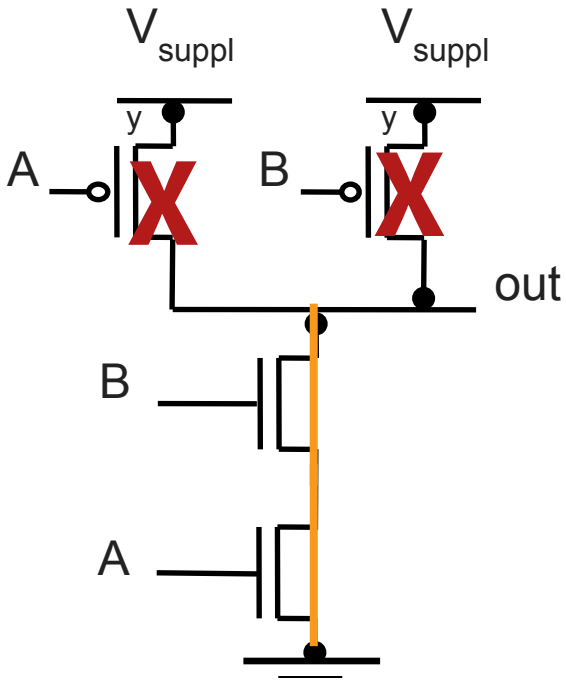


Truth Table

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

4-Transistor Combination

- What function is implemented?

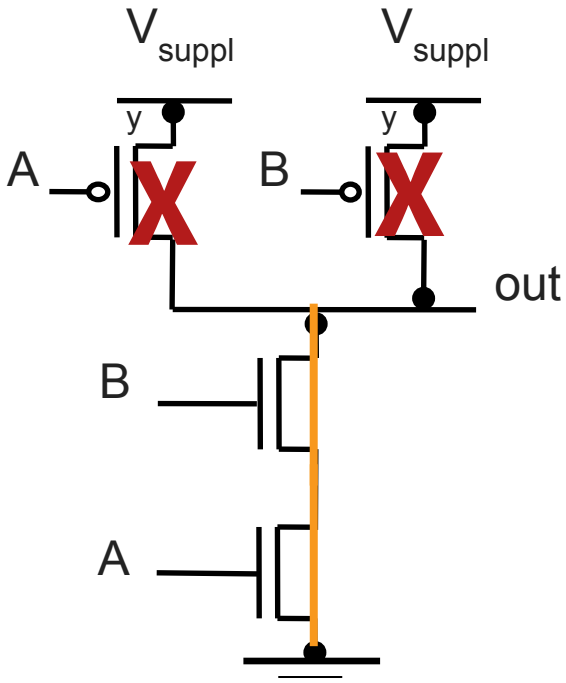


Truth Table

A	B	Out	¬Out
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

4-Transistor Combination

- $f(a, b) = \overline{a \wedge b}$

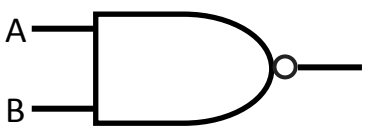


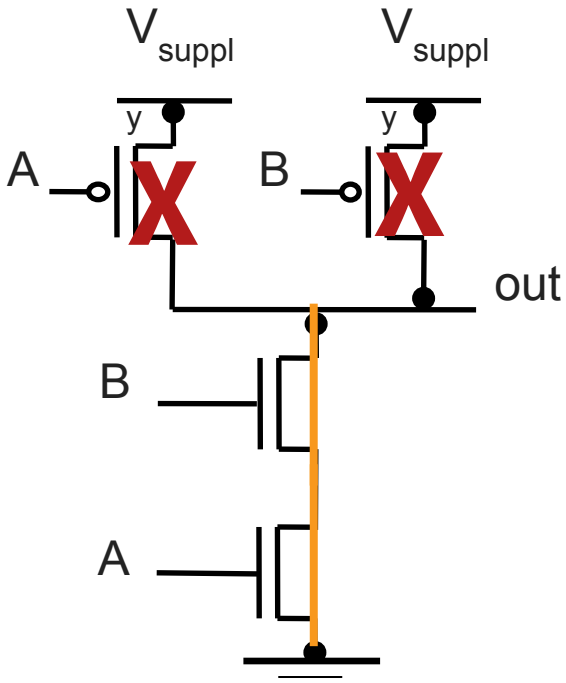
Truth Table

A	B	Out	¬Out
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

4-Transistor Combination

- $f(a, b) = \overline{a \wedge b}$
- not and → NAND gate

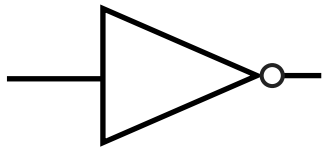




Truth Table

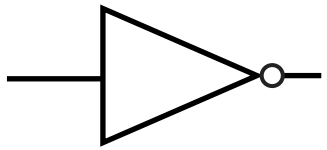
A	B	Out	¬Out
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

1 and 2 Input Logic Gates

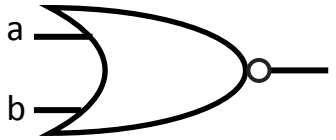
Inverter: a  $f(a) = \bar{a}$

$= \neg a$

1 and 2 Input Logic Gates

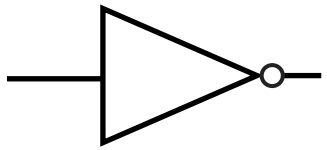
Inverter: a  $f(a) = \overline{a}$

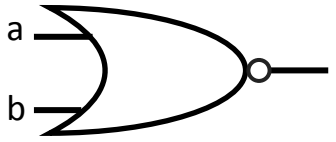
$= \neg a$

NOR: 

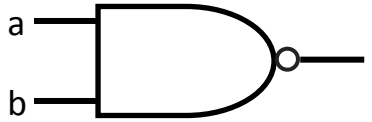
$f(a, b) = \overline{a \vee b}$

1 and 2 Input Logic Gates

Inverter: a  $f(a) = \overline{a}$ $= \neg a$

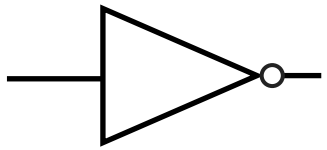
NOR: 

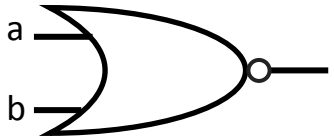
$f(a, b) = \overline{a \vee b}$

NAND: 

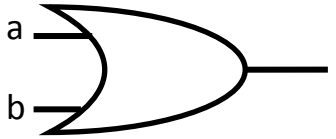
$f(a, b) = \overline{a \wedge b}$

1 and 2 Input Logic Gates

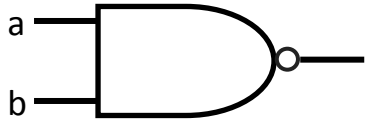
Inverter: a  $f(a) = \overline{a}$ $= \neg a$

NOR: 

$f(a, b) = \overline{a \vee b}$

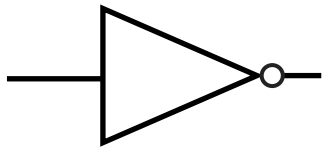
OR: 

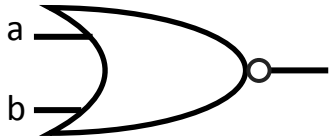
$f(a, b) = a \vee b$ $= a + b$

NAND: 

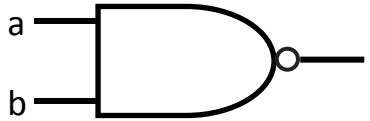
$f(a, b) = \overline{a \wedge b}$

1 and 2 Input Logic Gates

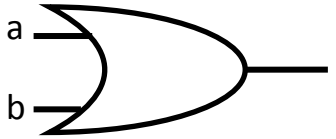
Inverter: a  $f(a) = \overline{a}$ $= \neg a$

NOR: 

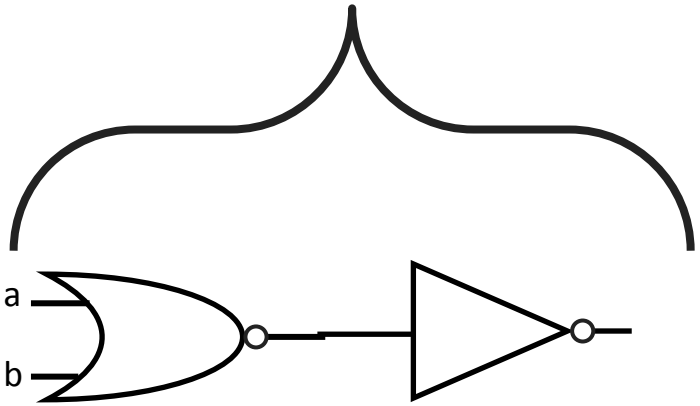
$f(a, b) = \overline{a \vee b}$

NAND: 

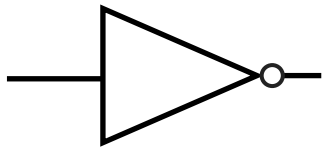
$f(a, b) = \overline{a \wedge b}$

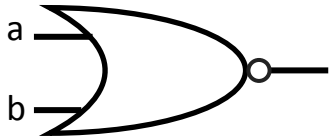
OR: 

$f(a, b) = a \vee b$ $= a + b$

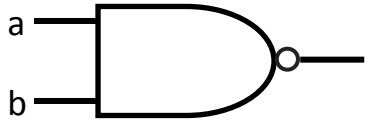


1 and 2 Input Logic Gates

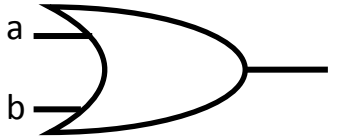
Inverter: a  $f(a) = \overline{a}$ $= \neg a$

NOR: 

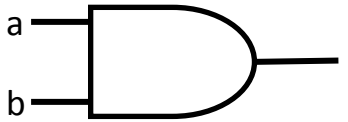
$f(a, b) = \overline{a \vee b}$

NAND: 

$f(a, b) = \overline{a \wedge b}$

OR: 

$f(a, b) = a \vee b$ $= a + b$

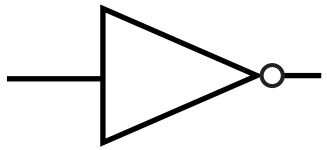
AND: 

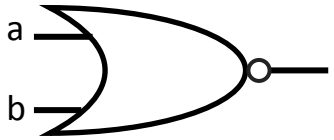
$f(a, b) = a \wedge b$

$= a \times b$

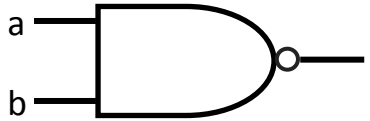
$= ab$

1 and 2 Input Logic Gates

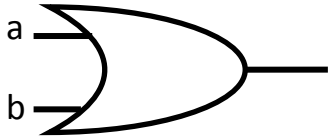
Inverter: a  $f(a) = \overline{a}$ $= \neg a$

NOR: 

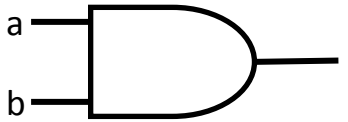
$f(a, b) = \overline{a \vee b}$

NAND: 

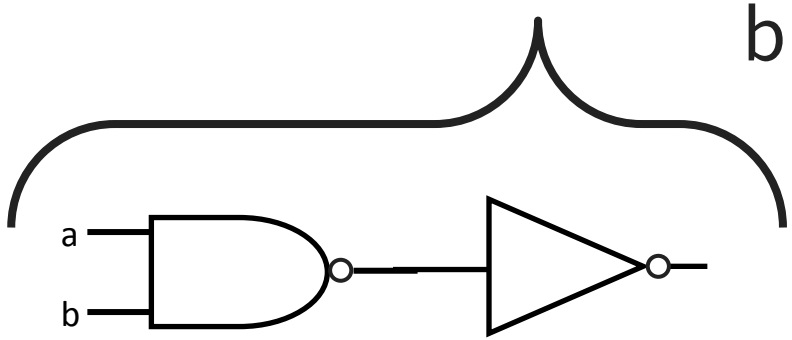
$f(a, b) = \overline{a \wedge b}$

OR: 

$f(a, b) = a \vee b$ $= a + b$

AND: 

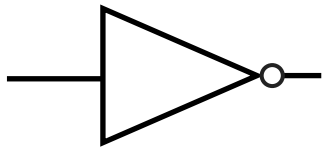
$f(a, b) = a \wedge b$

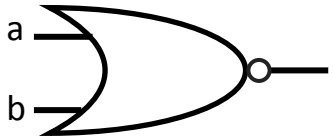


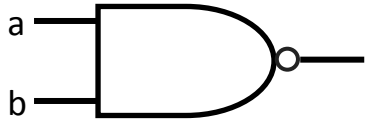
$= a \times b$

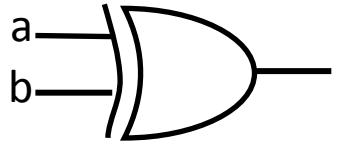
$= ab$

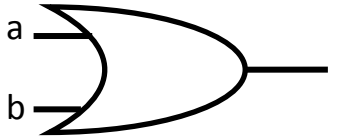
1 and 2 Input Logic Gates

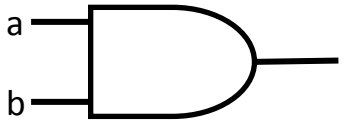
Inverter: a  $f(a) = \overline{a}$ $= \neg a$

NOR:  $f(a, b) = \overline{a \vee b}$

NAND:  $f(a, b) = \overline{a \wedge b}$

XOR:  $f(a, b) = a \oplus b$ $= \overline{a}b \vee a\overline{b}$

OR:  $f(a, b) = a \vee b$ $= a + b$

AND:  $f(a, b) = a \wedge b$ $= a \times b$ $= ab$

Identities

$a \vee 0 =$

$a \vee 1 =$

$a \vee \bar{a} =$

$a \wedge 0 =$

$a \wedge 1 =$

$a \wedge \bar{a} =$

Truth Table

a	$a \vee 0$
0	0
1	1

Identities

$a \vee 0 = a$

$a \vee 1 =$

$a \vee \bar{a} =$

$a \wedge 0 =$

$a \wedge 1 =$

$a \wedge \bar{a} =$

Truth Table

a	a ∨ 0
0	0
1	1

Identities

$a \vee 0 = a$

$a \vee 1 = 1$

$a \vee \bar{a} =$

$a \wedge 0 =$

$a \wedge 1 =$

$a \wedge \bar{a} =$

Identities

$a \vee 0 = a$

$a \vee 1 = 1$

$a \vee \bar{a} = 1$

$a \wedge 0 =$

$a \wedge 1 =$

$a \wedge \bar{a} =$

Identities

$a \vee 0 = a$

$a \vee 1 = 1$

$a \vee \bar{a} = 1$

$a \wedge 0 = 0$

$a \wedge 1 =$

$a \wedge \bar{a} =$

Truth Table

a	$a \wedge 1$
0	0
1	1

Identities

$a \vee 0 = a$

$a \vee 1 = 1$

$a \vee \bar{a} = 1$

$a \wedge 0 = 0$

$a \wedge 1 = a$

$a \wedge \bar{a} =$

Truth Table

a	$a \wedge 1$
0	0
1	1

Identities

$$a \vee 0 = a$$
$$a \vee 1 = 1$$
$$a \vee \bar{a} = 1$$

$$a \wedge 0 = 0$$
$$a \wedge 1 = a$$
$$a \wedge \bar{a} = 0$$

DeMorgan's Law

$$\overline{a \wedge b} = \bar{a} \vee \bar{b}$$

$$\overline{a \vee b} = \bar{a} \wedge \bar{b}$$

Distributive Property

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$$

Distributive Property

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$$

$$a\ (b + c) = ab + ac$$

Which Gate is this?

PollEV Question #1

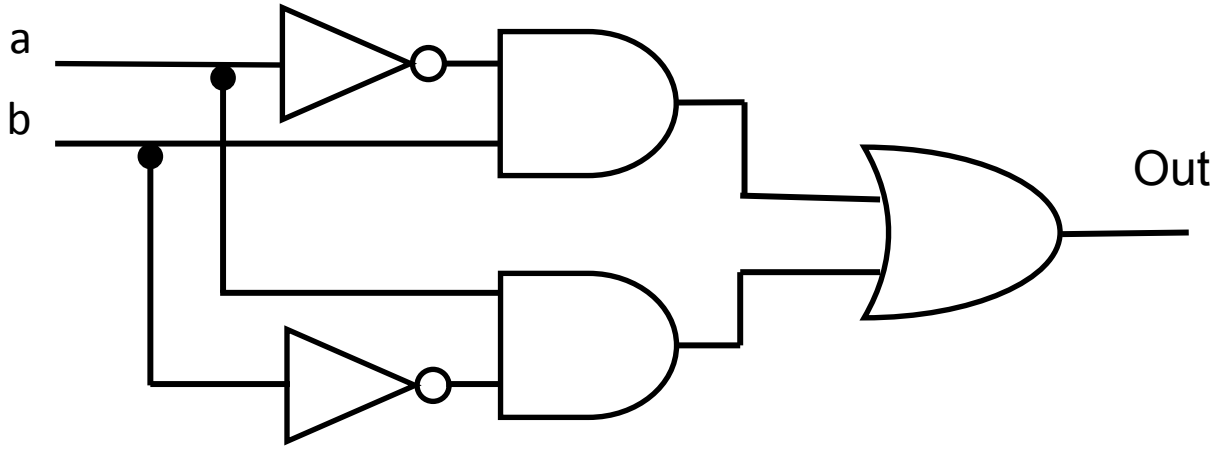
(A) NOT 

(B) OR 

(C) XOR 

(D) AND 

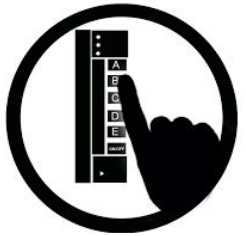
(E) NAND 



Truth Table

A	B	Out
0	0	
0	1	
1	0	
1	1	

PollEv.com/cs3410



Which Gate is this?

PollEV Question #1

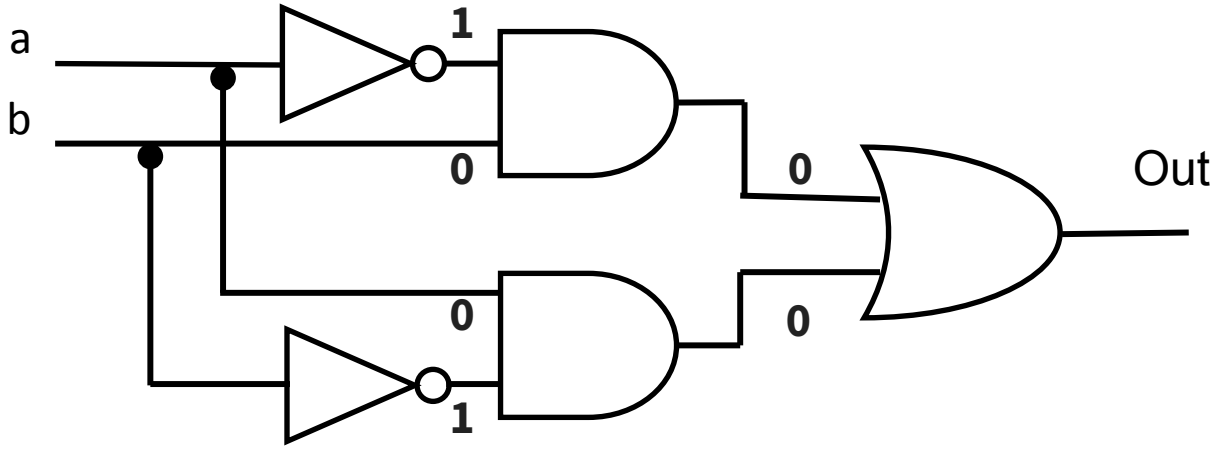
(A) NOT 

(B) OR 

(C) XOR 

(D) AND 

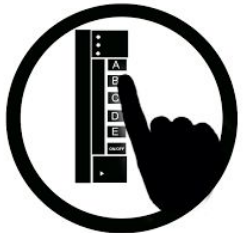
(E) NAND 



Truth Table

A	B	Out
0	0	0
0	1	
1	0	
1	1	

PollEv.com/cs3410



Which Gate is this?

PollEV Question #1

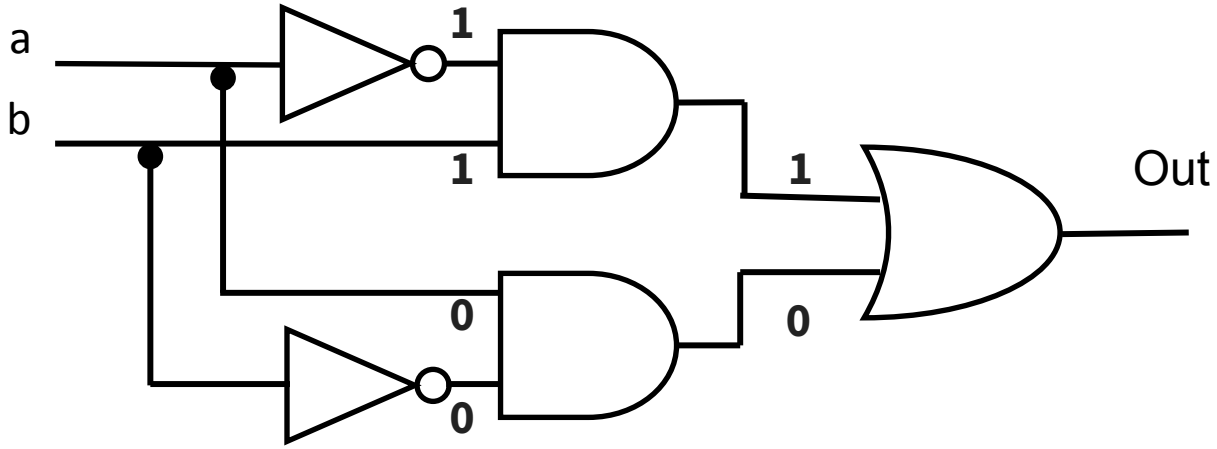
(A) NOT 

(B) OR 

(C) XOR 

(D) AND 

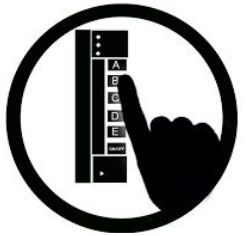
(E) NAND 



Truth Table

A	B	Out
0	0	0
0	1	1
1	0	
1	1	

PollEv.com/cs3410



Which Gate is this?

PollEV Question #1

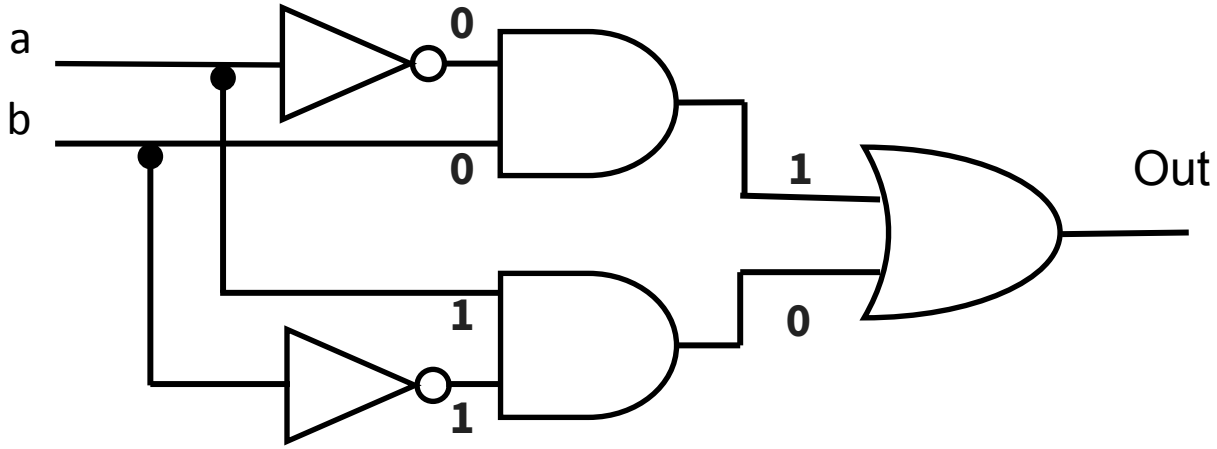
(A) NOT 

(B) OR 

(C) XOR 

(D) AND 

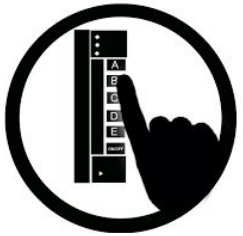
(E) NAND 



Truth Table

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	

PollEv.com/cs3410



Which Gate is this?

PollEV Question #1

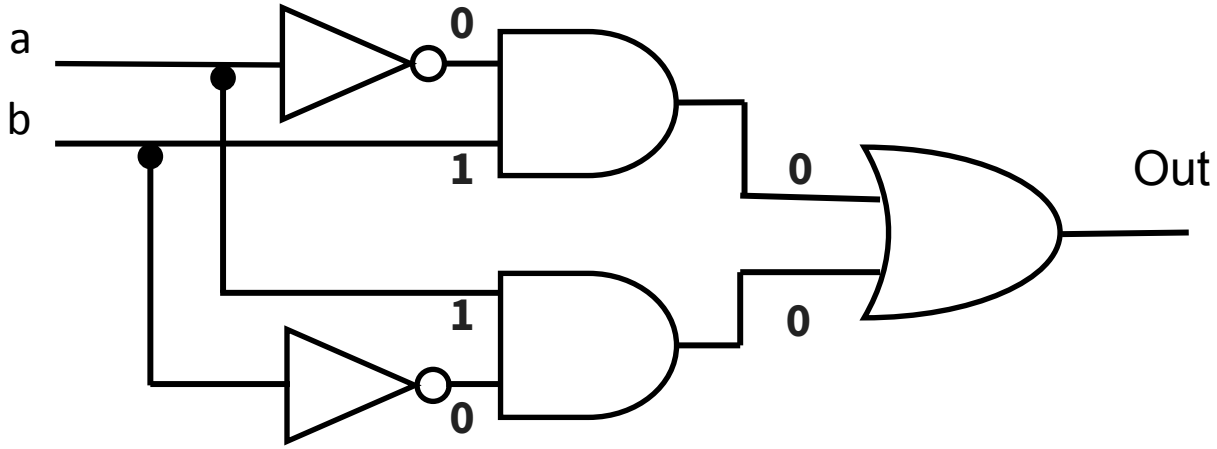
(A) NOT 

(B) OR 

(C) XOR 

(D) AND 

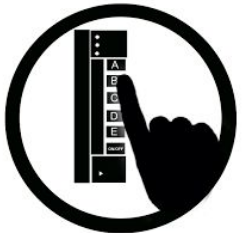
(E) NAND 



Truth Table

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

PollEv.com/cs3410



Which Gate is this?

PollEV Question #1

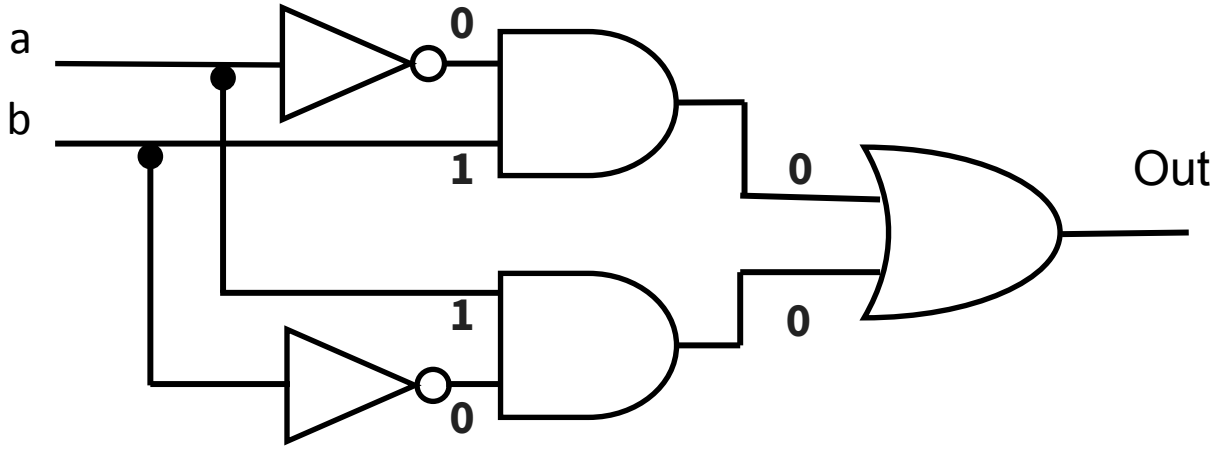
(A) NOT 

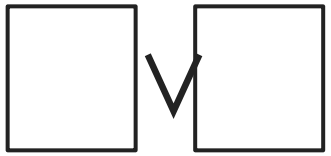
(B) OR 

(C) XOR 

(D) AND 

(E) NAND 

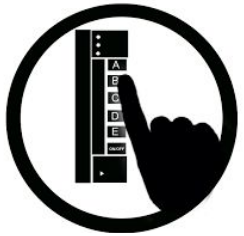




Truth Table

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

PollEv.com/cs3410



Cornell Bowers CIS
Computer Science

Which Gate is this?

PollEV Question #1

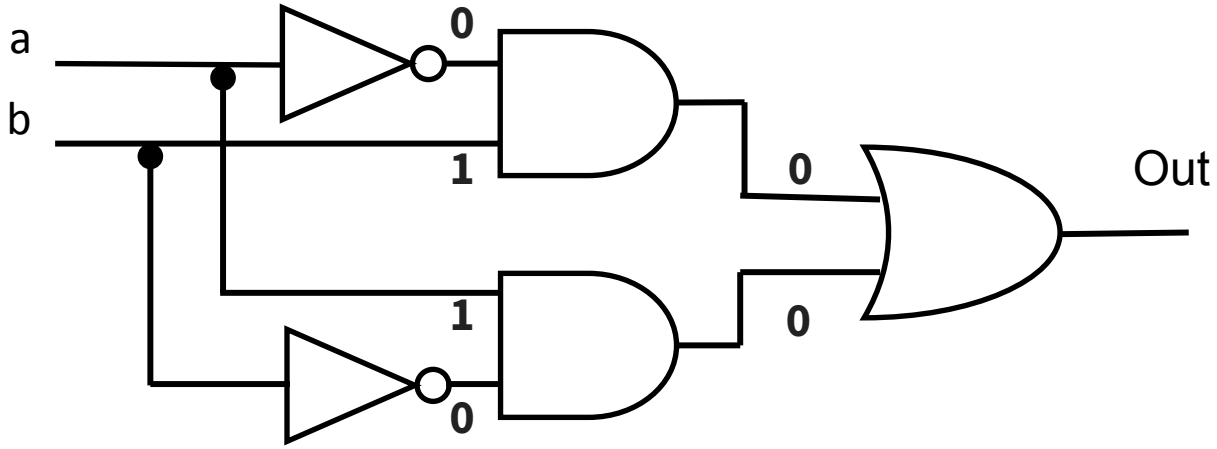
(A) NOT 

(B) OR 

(C) XOR 

(D) AND 

(E) NAND 

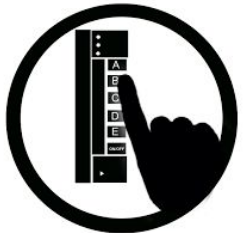


$\bar{a}b \vee$

Truth Table

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

PollEv.com/cs3410



Which Gate is this?

PollEV Question #1

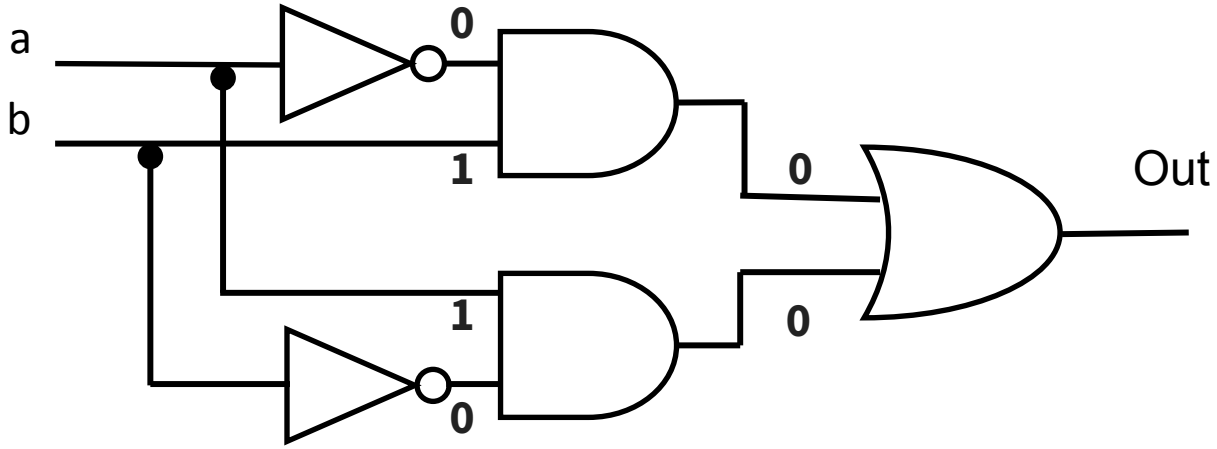
(A) NOT 

(B) OR 

(C) XOR 

(D) AND 

(E) NAND 

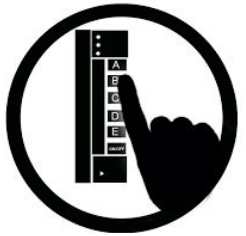


$\bar{a}b \vee a\bar{b}$

Truth Table

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

PollEv.com/cs3410



Which Gate is this?

PollEV Question #1

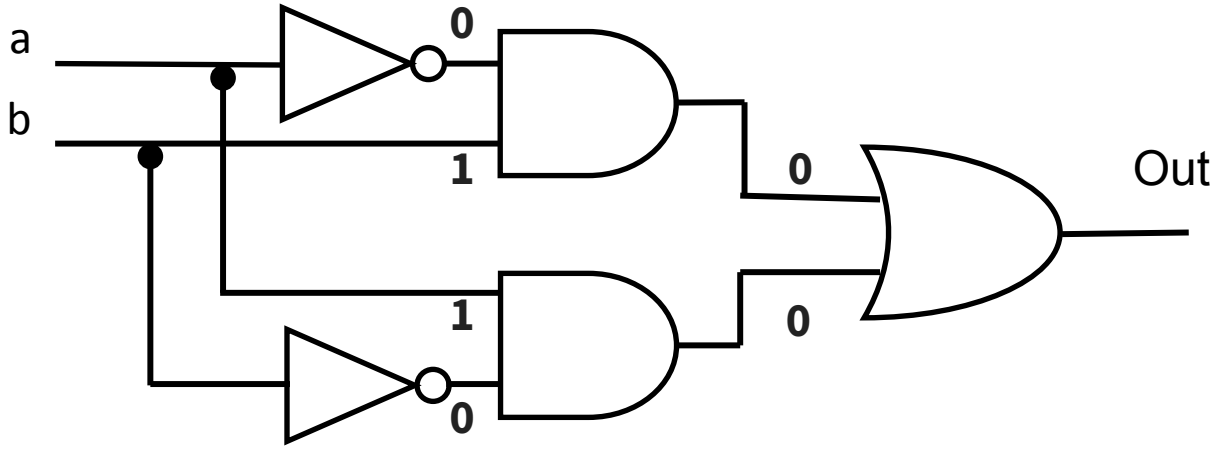
(A) NOT 

(B) OR 

(C) XOR 

(D) AND 

(E) NAND 



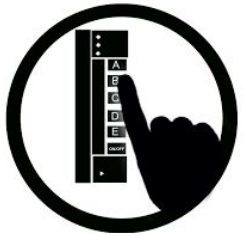
$\bar{a}b \vee a\bar{b}$

minterms!

Truth Table

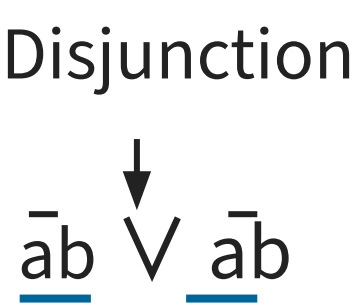
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

PollEv.com/cs3410



Cornell Bowers CIS Computer Science

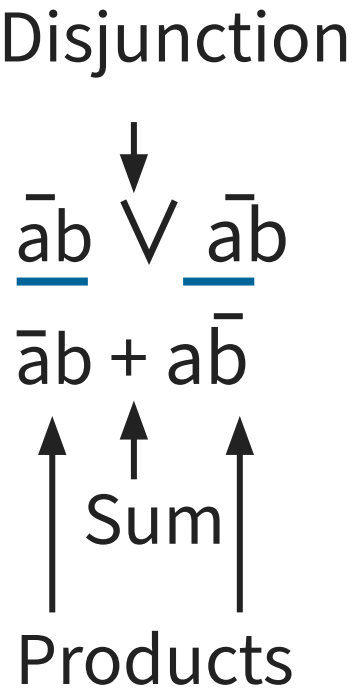
Truth Table to Boolean Formula with SOP



- **Sum-Of-Product (SOP)** or **Disjunctive Normal Form (DNF)**

Truth Table to Boolean Formula with SOP

- **Sum-Of-Product (SOP)** or **Disjunctive Normal Form (DNF)**

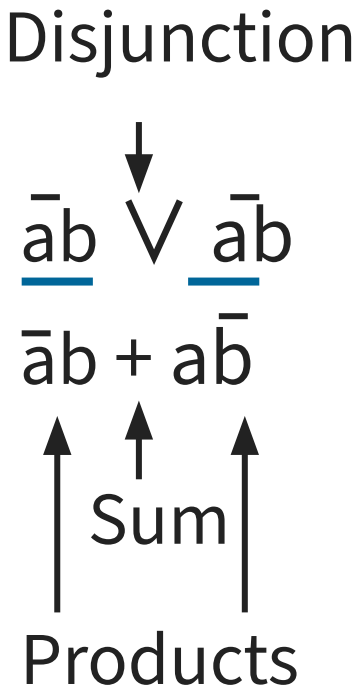


Truth Table to Boolean Formula with SOP

Truth Table

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

- **Sum-Of-Product (SOP)** or **Disjunctive Normal Form (DNF)**
- **minterms**: product of (inverted) inputs

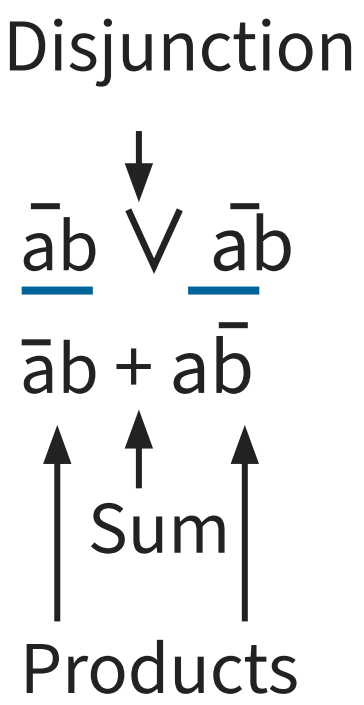


Truth Table to Boolean Formula with POS

Truth Table

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

- **Sum-Of-Product (SOP)** or **Disjunctive Normal Form (DNF)**
- **minterms**: product of (inverted) inputs
- **Product-Of-Sum (POS)** or **Conjunctive Normal Form (CNF)**
- **maxterms:** sum of (inverted) inputs



$(a + b)\ (\bar{a} + \bar{b})$

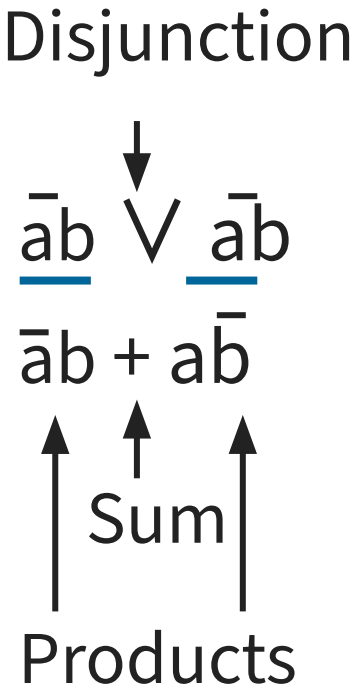
Truth Table to Boolean Formula with POS

Truth Table

A	B	Out	¬Out
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

$\overline{f(a, b)} = \bar{a}\bar{b} + ab$

- **Sum-Of-Product (SOP)** or **Disjunctive Normal Form (DNF)**
- **minterms**: find rows where output is 1
- **Product-Of-Sum (POS)** or **Conjunctive Normal Form (CNF)**
- **maxterms:** sum of (inverted) inputs



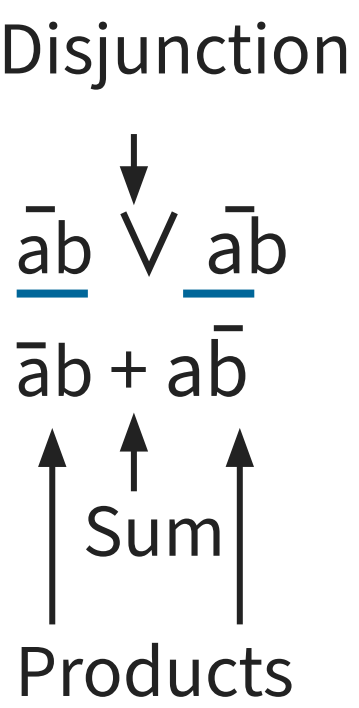
$(a + b)\ (\bar{a} + \bar{b})$

Truth Table to Boolean Formula with POS

Truth Table

A	B	Out	¬Out
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

- **Sum-Of-Product (SOP)** or **Disjunctive Normal Form (DNF)**
- **minterms**: find rows where output is 1
- **Product-Of-Sum (POS)** or **Conjunctive Normal Form (CNF)**
- **maxterms:** sum of (inverted) inputs



$\overline{f(a, b)} = \bar{a}\bar{b} + ab$

$f(a, b) = \overline{\bar{a}\bar{b} + ab} = (a + b)\,(\bar{a} + \bar{b})$

Sum-Of-Product Practice

Sum-Of-Product Practice

a	b	c	out	minterm
0	**0**	**0**	**0**	$\bar{a}\ \bar{b}\ \bar{c}$
0	**0**	**1**	**1**	$\bar{a}\ \bar{b}\ c$
0	**1**	**0**	**0**	$\bar{a}\ b\ \bar{c}$
0	**1**	**1**	**1**	$\bar{a}\ b\ c$
1	**0**	**0**	**0**	$a\ \bar{b}\ \bar{c}$
1	**0**	**1**	**1**	$a\ \bar{b}\ c$
1	**1**	**0**	**0**	$a\ b\ \bar{c}$
1	**1**	**1**	**0**	$a\ b\ c$

1) Write minterms
2) sum of products:
   - OR of all minterms where out=1

Logic Implementation

- How to implement a desired logic function?

a	b	c	out	minterm
0	0	0	0	$\bar{a}\,\bar{b}\,\bar{c}$
0	0	1	1	$\bar{a}\,\bar{b}\,c$
0	1	0	0	$\bar{a}\,b\,\bar{c}$
0	1	1	1	$\bar{a}\,b\,c$
1	0	0	0	$a\,\bar{b}\,\bar{c}$
1	0	1	1	$a\,\bar{b}\,c$
1	1	0	0	$a\,b\,\bar{c}$
1	1	1	0	$a\,b\,c$

1) Write minterms
2) sum of products:

- OR of all minterms where out=1
  - E.g. out = $\overline{abc} + \bar{a}bc + a\bar{b}c$

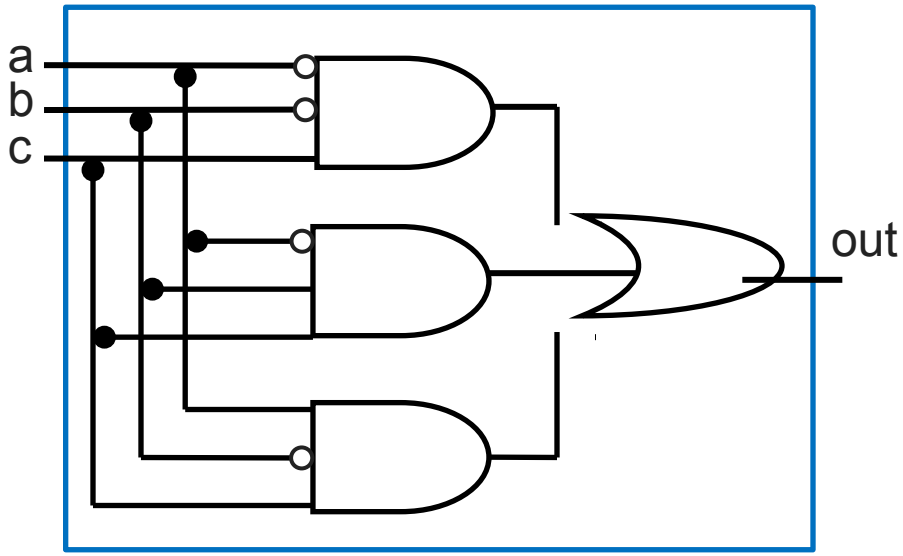
Logic Implementation

- How to implement a desired logic function?

a	b	c	out	minterm
0	0	0	0	$\bar{a}\,\bar{b}\,\bar{c}$
0	0	1	1	$\bar{a}\,\bar{b}\,c$
0	1	0	0	$\bar{a}\,b\,\bar{c}$
0	1	1	1	$\bar{a}\,b\,c$
1	0	0	0	$a\,\bar{b}\,\bar{c}$
1	0	1	1	$a\,\bar{b}\,c$
1	1	0	0	$a\,b\,\bar{c}$
1	1	1	0	$a\,b\,c$

1) Write minterms
2) sum of products:

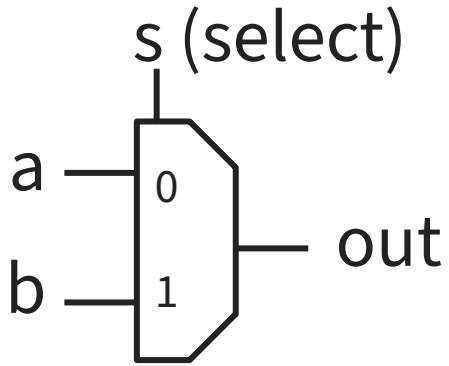
- OR of all minterms where out=1
  - E.g. out = $\overline{abc} + \bar{a}bc + a\bar{b}c$



corollary: *any* combinational circuit *can be* implemented in two levels of logic (ignoring inverters)

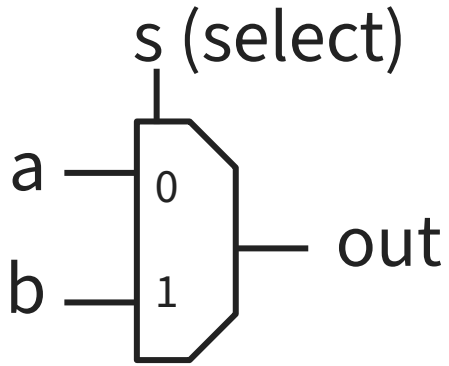
Multiplexor

- Select one of two inputs
- Ternary operator in C: `sel? b : a`



Multiplexor

- Select one of two inputs
- Ternary operator in C: `sel? b : a`

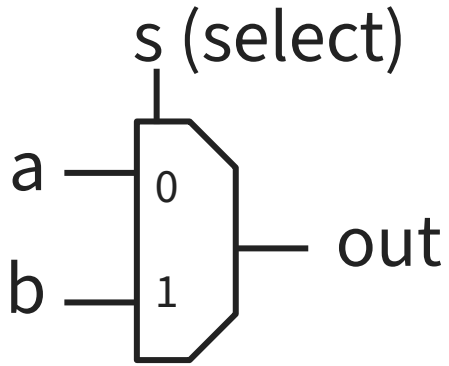


Truth Table

sel	b	a	Out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Multiplexor

- Select one of two inputs
- Ternary operator in C: `sel? b : a`



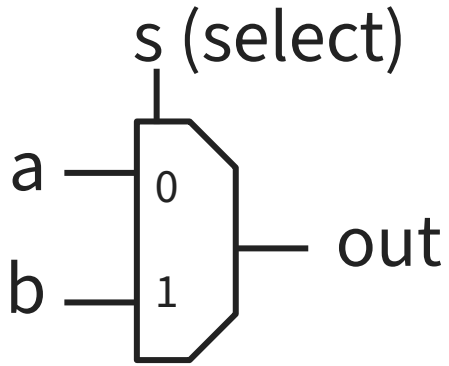
$\overline{s}\overline{b}a + \overline{s}b\overline{a} + sb\overline{a} + sba$

Truth Table

s	b	a	Out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Multiplexor

- Select one of two inputs
- Ternary operator in C: `sel? b : a`



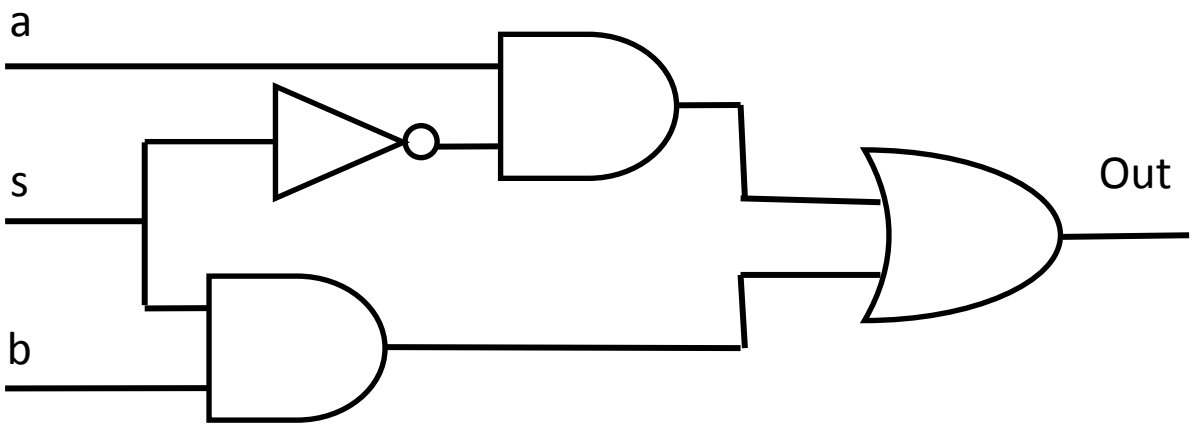
$\overline{s}\overline{b}a + \overline{s}ba + s\overline{b}\overline{a} + sba = \overline{s}a + sb$

Truth Table

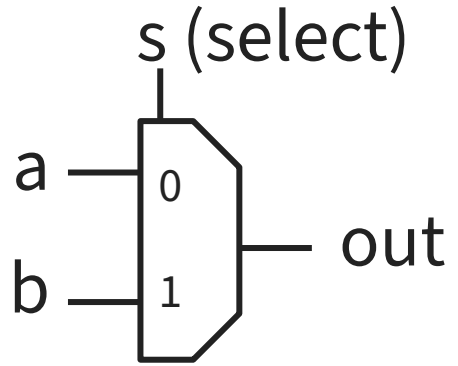
s	b	a	Out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Multiplexor

- Select one of two inputs
- Ternary operator in C: `sel? b : a`



$\bar{s}\bar{b}a + \bar{s}ba + sb\bar{a} + sba = \bar{s}a + sb$



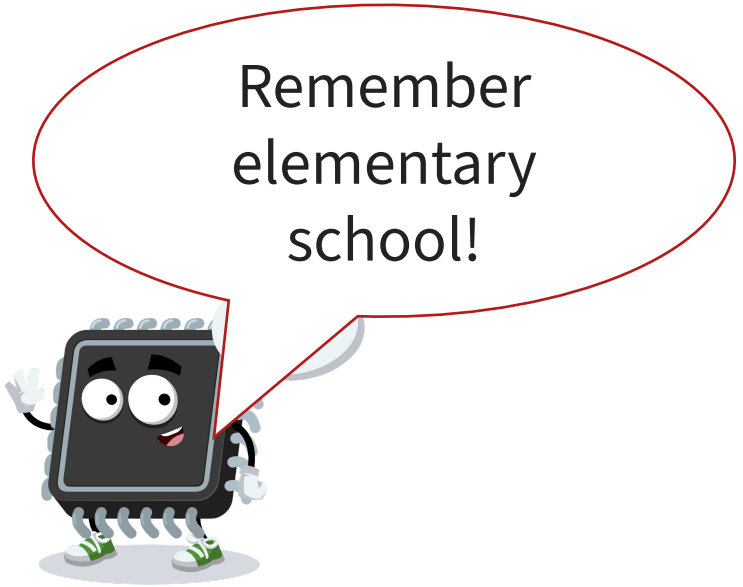
Truth Table

s	b	a	Out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Binary Addition

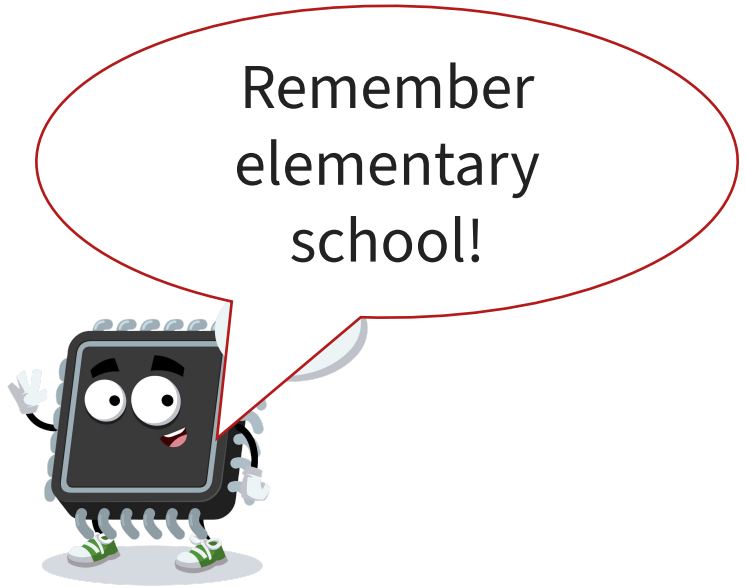
Adding Binary Numbers

$$\begin{array}{r} 24_{10} \\ +\ 17_{10} \\ \hline \_\_\_ \end{array}$$



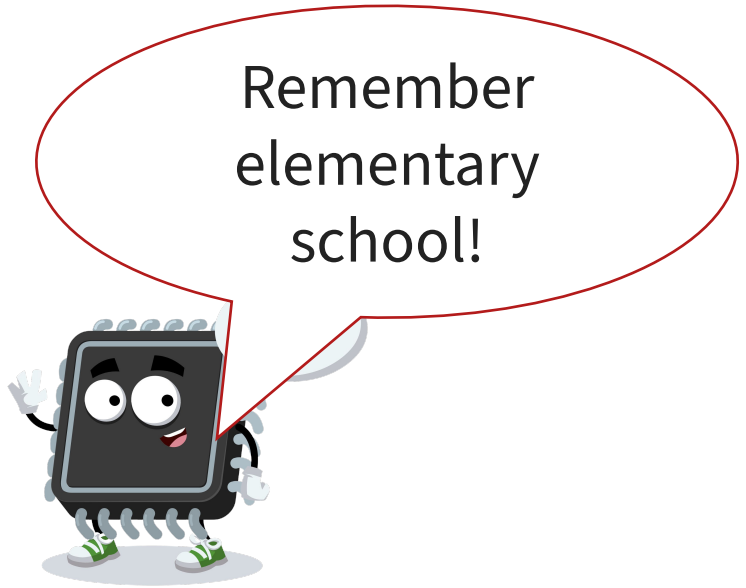
Adding Binary Numbers

$$
\begin{array}{r}
24_{10} \\
+\ 17_{10} \\
\quad (+1)\ _{10} \\
\hline
\\
1_{10}
\end{array}
$$



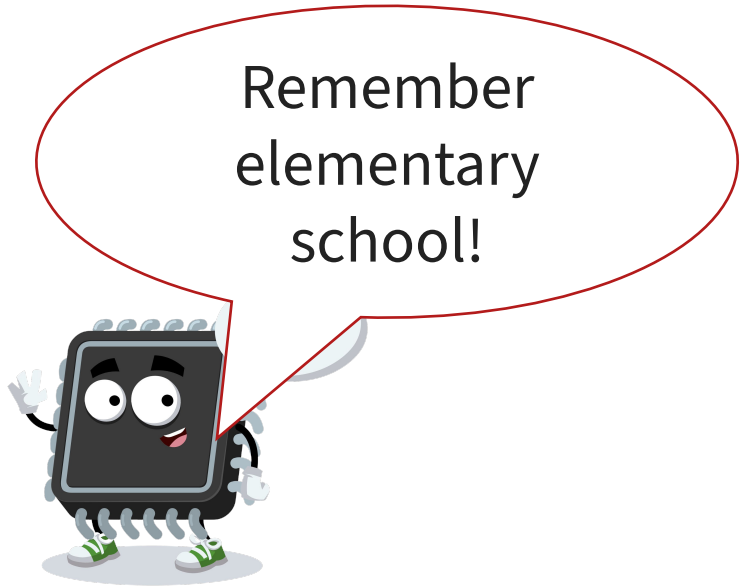
Adding Binary Numbers

$$
\begin{array}{r}
24_{10} \\
+\ 17_{10} \\
\quad +1 \\
\hline
41_{10}
\end{array}
$$



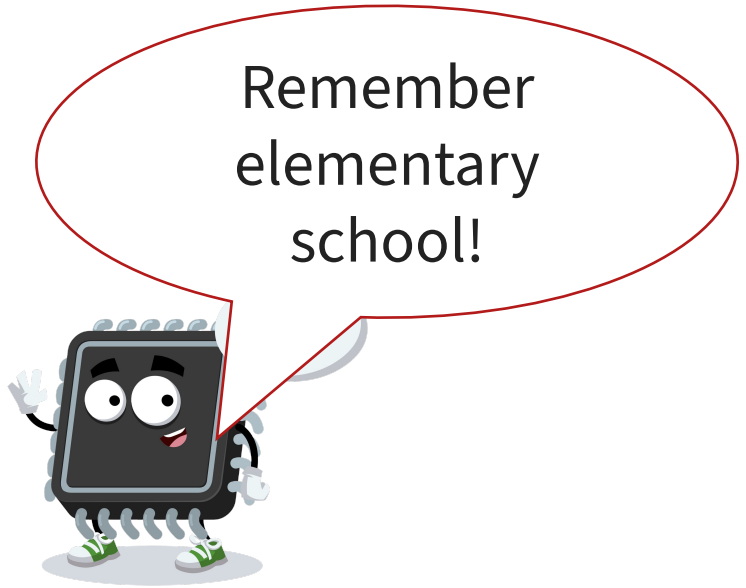
Adding Binary Numbers

$$\begin{array}{r} 24_{10} \\ +\ 17_{10} \\ \quad (+1) \\ \hline 41_{10} \end{array} \qquad \begin{array}{r} 11000_{2} \\ +\ 10001_{2} \\ \hline \end{array}$$



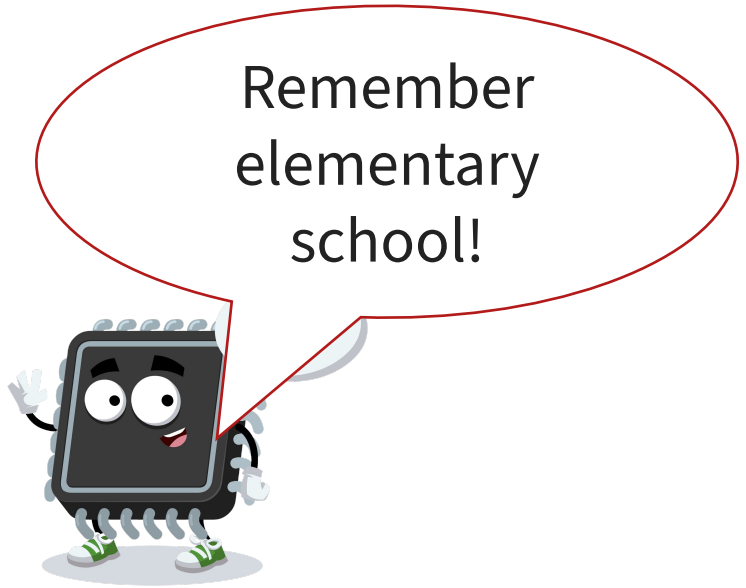
Adding Binary Numbers

$$\begin{array}{r} 24_{10} \\ +\ 17_{10} \\ \quad (+1) \\ \hline 41_{10} \end{array} \qquad \begin{array}{r} 11000_{2} \\ +\ 10001_{2} \\ \hline 1_{2} \end{array}$$



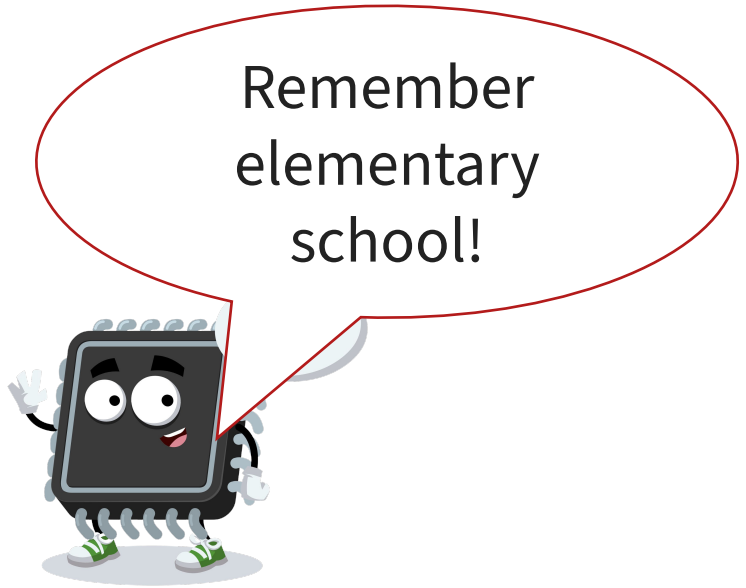
Adding Binary Numbers

$$\begin{array}{r} 24_{10} \\ +\ 17_{10} \\ \quad (+1) \\ \hline 41_{10} \end{array} \qquad \begin{array}{r} 11000_{2} \\ +\ 10001_{2} \\ \hline 01_{2} \end{array}$$



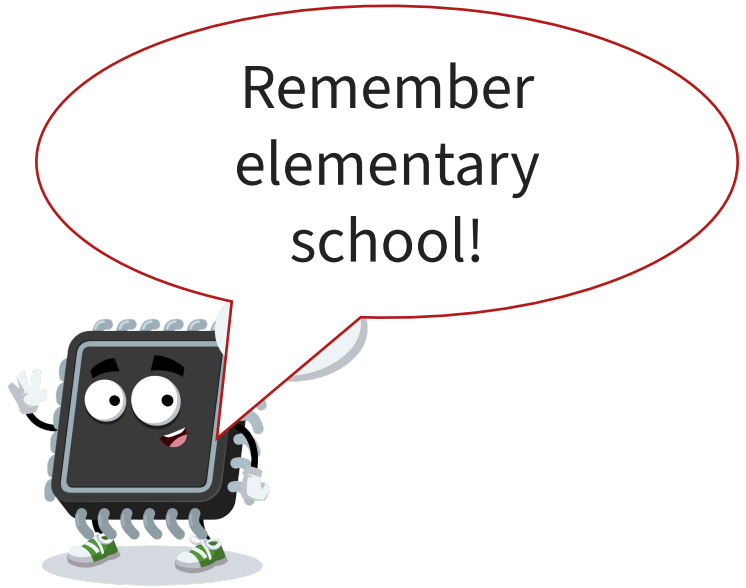
Adding Binary Numbers

$$\begin{array}{r} 24_{10} \\ +\ 17_{10} \\ \quad (+1) \\ \hline 41_{10} \end{array} \qquad \begin{array}{r} 11000_{2} \\ +\ 10001_{2} \\ \hline 001_{2} \end{array}$$



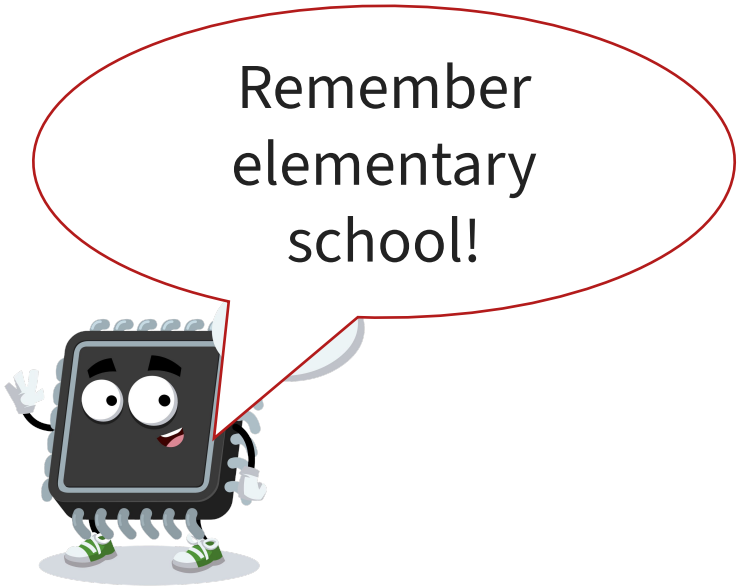
Adding Binary Numbers

$$\begin{array}{r} 24_{10} \\ +\ 17_{10} \\ \quad (+1) \\ \hline 41_{10} \end{array} \qquad \begin{array}{r} 11000_{2} \\ +\ 10001_{2} \\ \hline 1001_{2} \end{array}$$



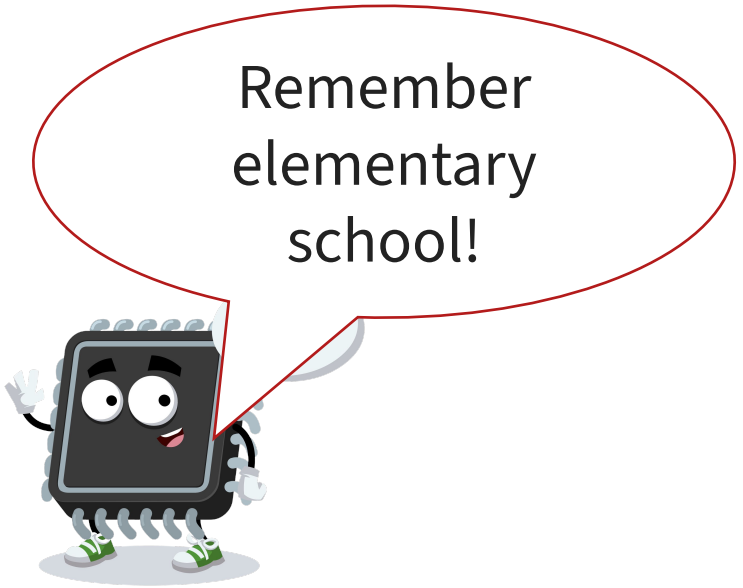
Adding Binary Numbers

$$\begin{array}{r} 24_{10} \\ +\ 17_{10} \\ \hline 41_{10} \end{array} \qquad \begin{array}{r} 11000_{2} \\ +\ 10001_{2} \\ \hline 01001_{2} \end{array}$$



Adding Binary Numbers

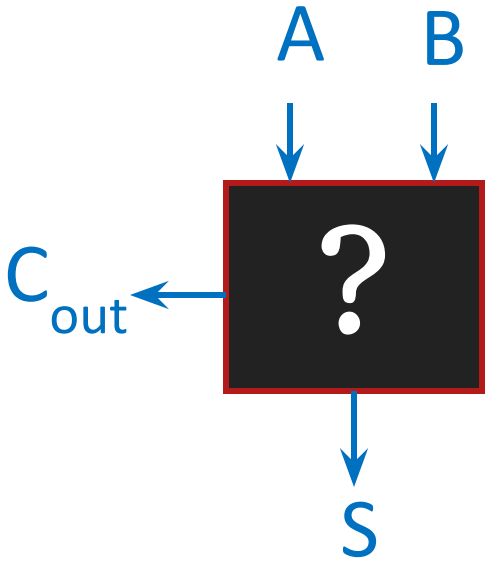
$$\begin{array}{r} 24_{10} \\ +\ 17_{10} \\ \hline 41_{10} \end{array} \qquad \begin{array}{r} 11000_{2} \\ +\ 10001_{2} \\ \hline 101001_{2} \end{array}$$



Binary Addition

- Binary addition requires
  - Add of ***two bits*** PLUS ***carry-in***
  - Also, ***carry-out*** if necessary

1-bit Half Adder



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

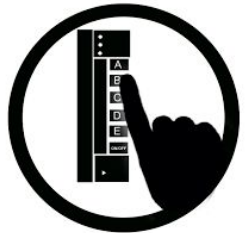
- Adds two 1-bit numbers
- Computes 1-bit result and 1-bit carry-out
- No carry-in

PollEv.com/cs3410

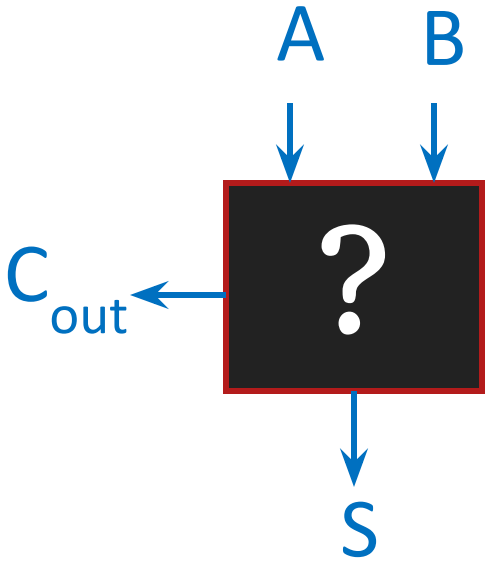
PollEV Question

What is the equation for C_{out}?

a) A + B
b) AB
c) A ⊕ B
d) A + !B
e) !A!B



1-bit Half Adder



- Adds two 1-bit numbers
- Computes 1-bit result and 1-bit carry-out
- No carry-in

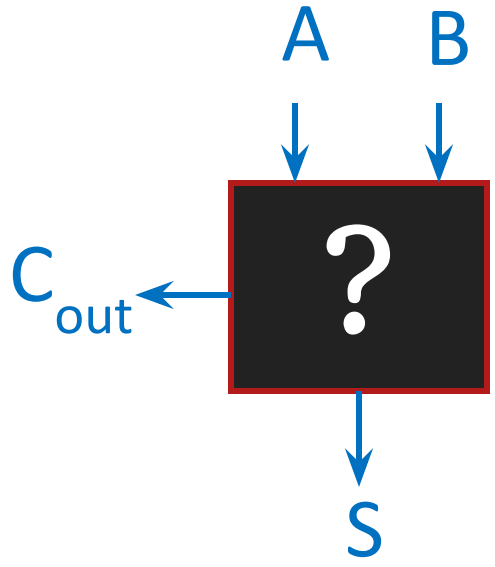
A	B	C_{out}	S
0	0		
0	1		
1	0		
1	1		

PollEV Question #3

What is the equation for C_{out}?

a) A + B
b) AB
c) A ⊕ B
d) A + !B
e) !A!B

1-bit Half Adder



- Adds two 1-bit numbers
- Computes 1-bit result and 1-bit carry-out
- No carry-in

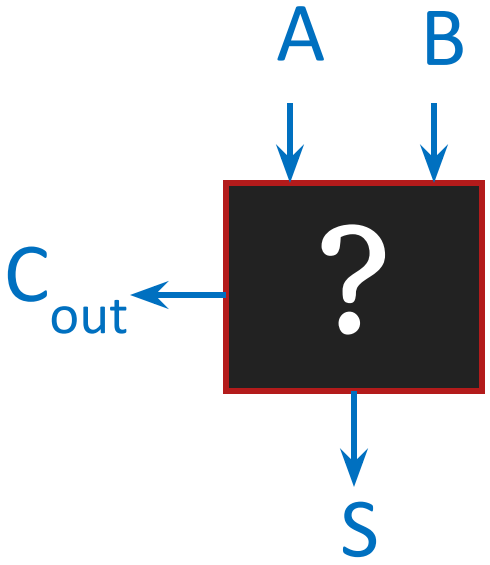
A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

0	0	0	1
0	0	1	1
+ 0	+ 1	+ 0	+ 1
0	1	1	0

S = one input equals 1

C_{out} = two inputs equal 1

1-bit Half Adder

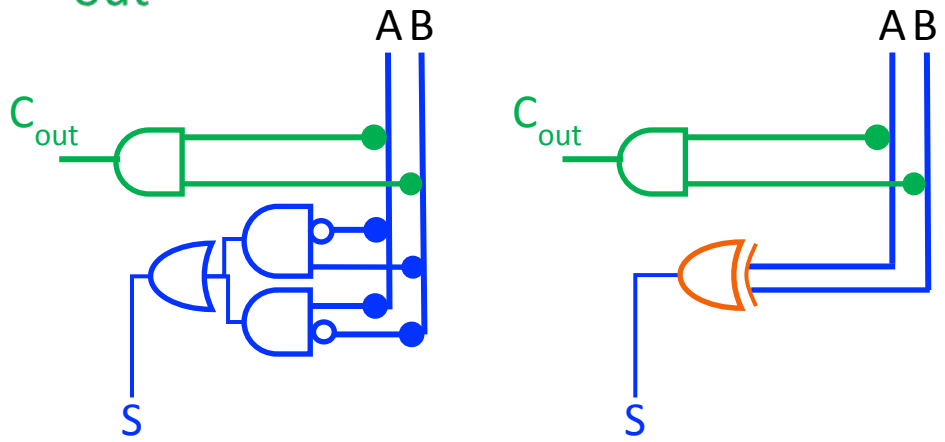


- Adds two 1-bit numbers
- Computes 1-bit result and 1-bit carry-out
- No carry-in

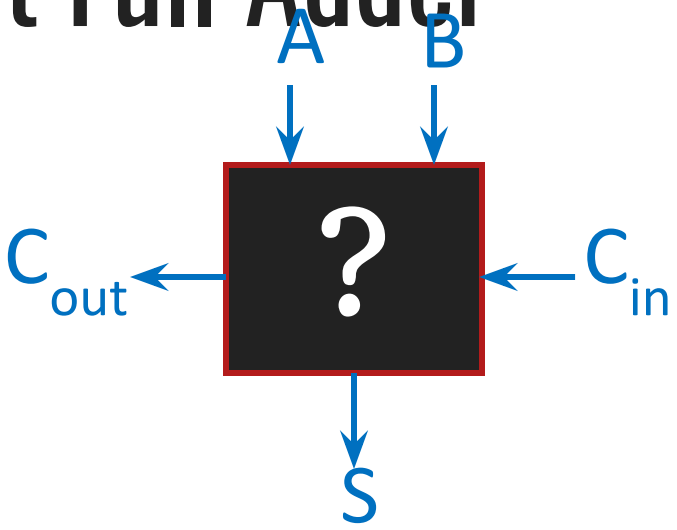
A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = \bar{A}B + A\bar{B} = A \oplus B$

$C_{out} = AB$



1-bit Full Adder

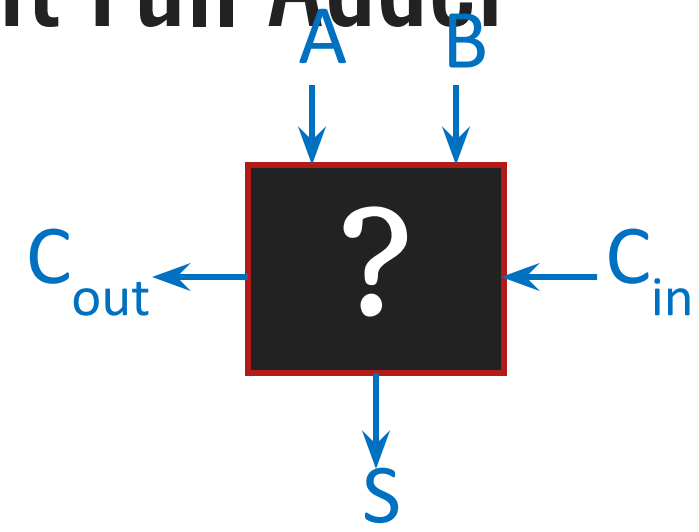


A	B	C_{in}	C_{out}	S
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

- Adds three 1-bit numbers
- Computes 1-bit result and 1-bit carry-out
- Can be cascaded

- Fill in Truth Table
- Create Sum-of-Product Form
- Draw the Circuits

1-bit Full Adder



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

- Adds three 1-bit numbers
- Computes 1-bit result and 1-bit carry-out
- Can be cascaded

PollEv.com/cs3410

PollEV Question #4

What is the equation for C_{out}?

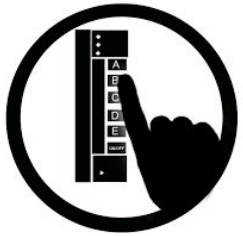
a) $A + B + C_{in}$

b) $!A + !B + !\,C_{in}$

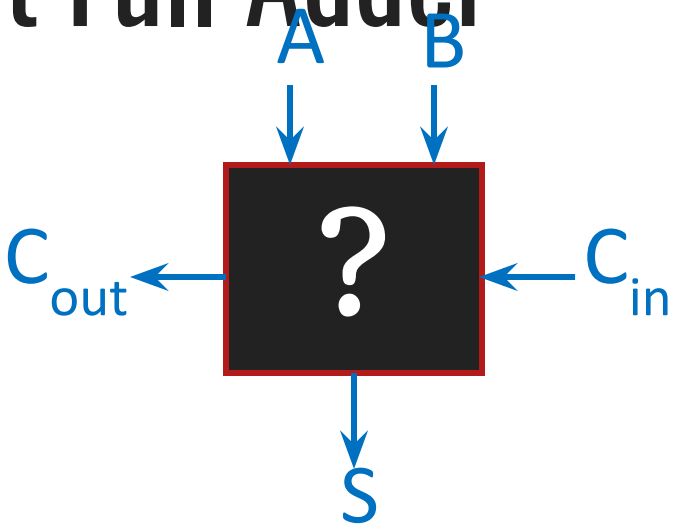
c) $A \oplus B \oplus C_{in}$

d) $\overline{A}BC + A\overline{B}\overline{C} + \overline{A}\overline{B}C + ABC$

e) $\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$



1-bit Full Adder



- Adds three 1-bit numbers
- Computes 1-bit result and 1-bit carry-out
- Can be cascaded

A	B	C_{in}	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

PollEV Question #4

What is the equation for C_{out}?

a) $A + B + C_{in}$

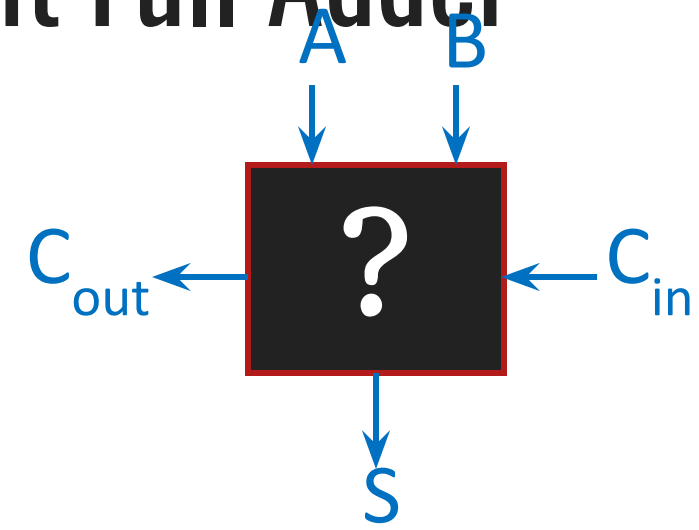
b) $!A + !B + !\,C_{in}$

c) $A \oplus B \oplus C_{in}$

d) $\overline{A}BC + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

e) $\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$

1-bit Full Adder

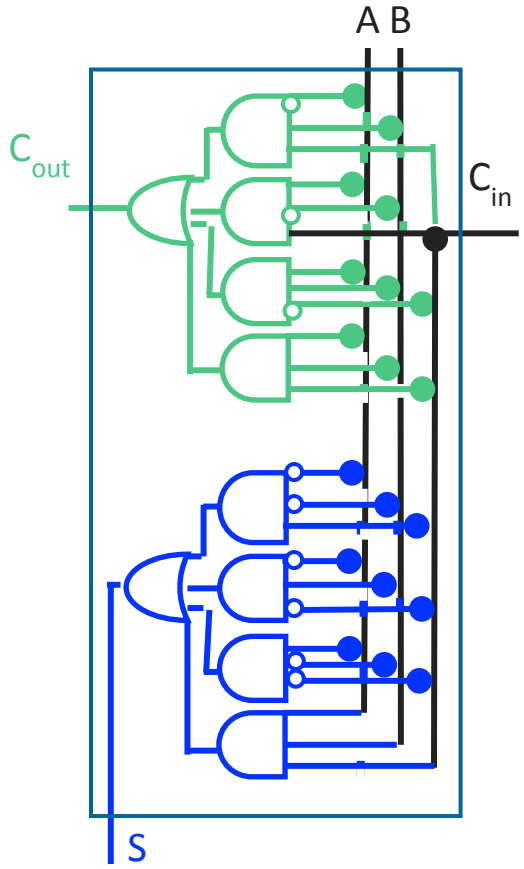


- Adds three 1-bit numbers
- Computes 1-bit result and 1-bit carry-out
- Can be cascaded

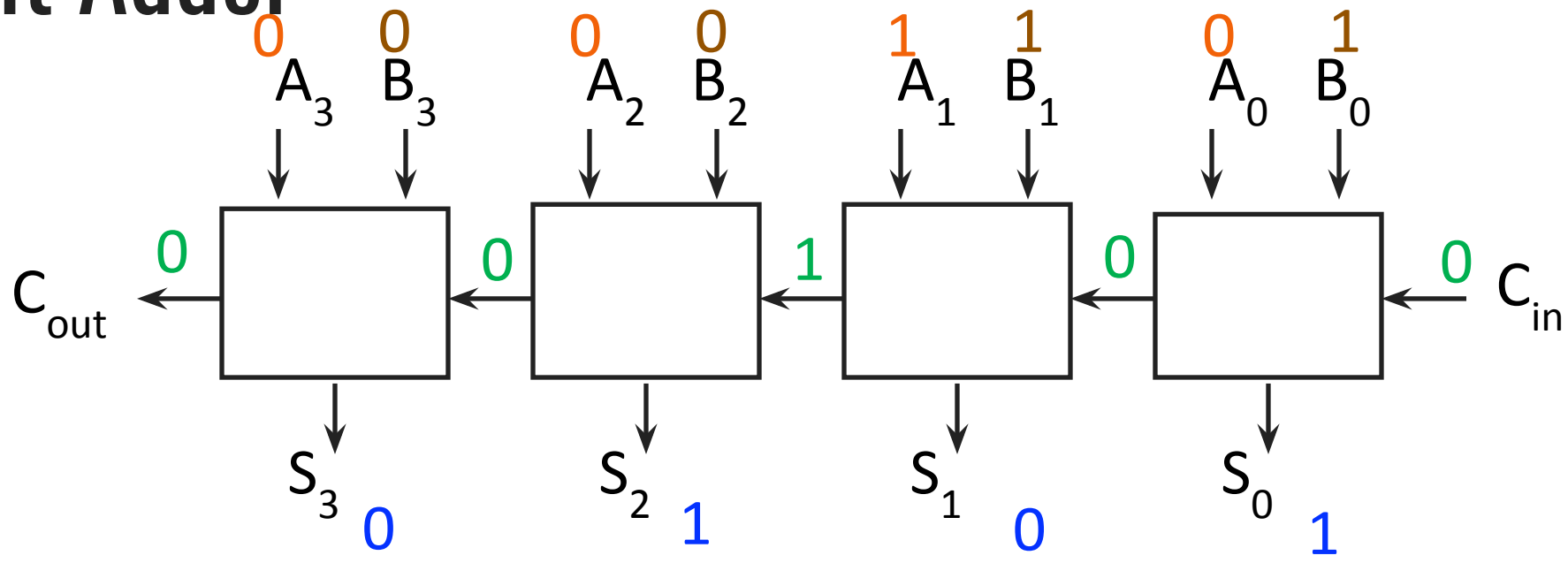
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$$

$$C_{out} = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

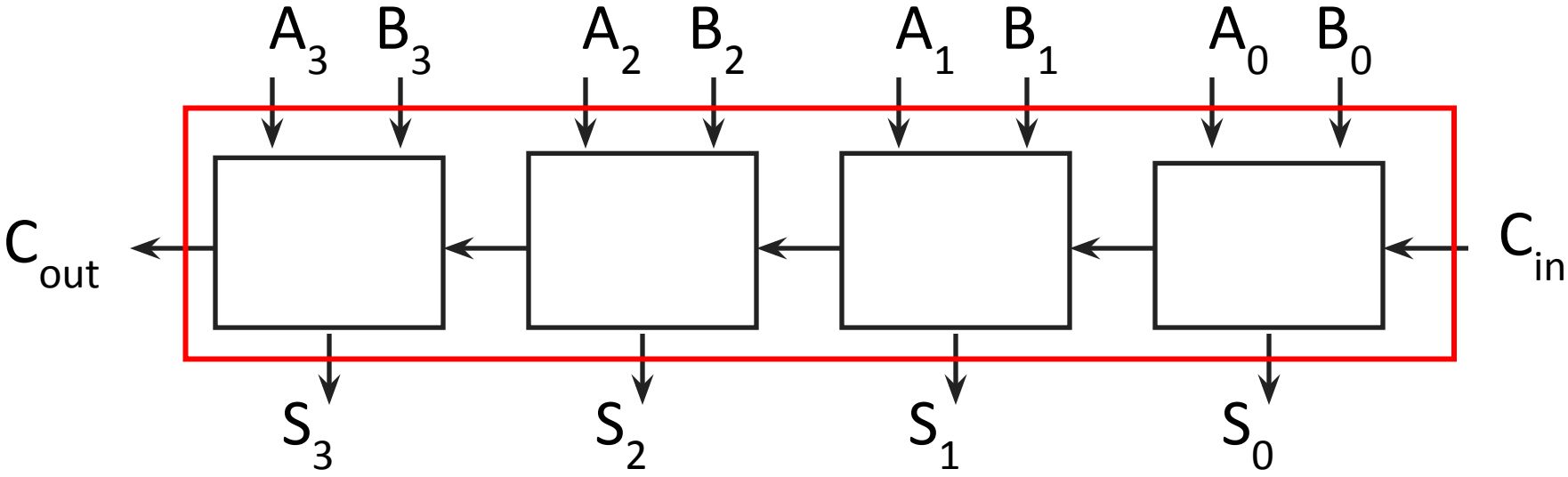


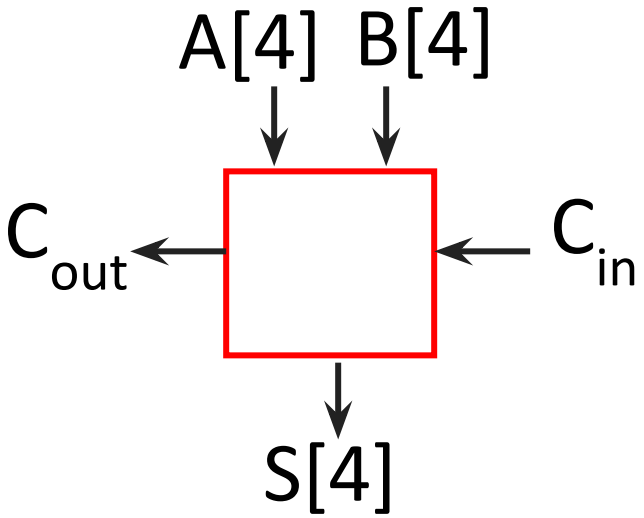
4-bit Adder



- Adds two 4-bit numbers, along with carry-in
- Computes 4-bit result and carry out
- 3 + 2 = 5
- Carry-out □ result > 4 bits

4-bit Adder





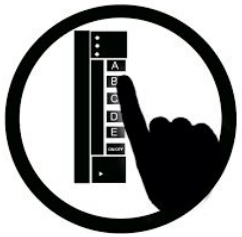
Build it and box it!

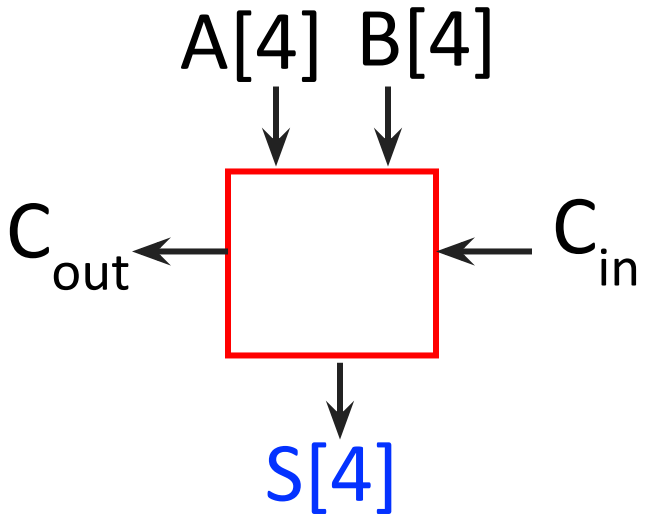
(Theme of 3410)

PollEV Question #5

What's the largest sum you can calculate with a 4 bit adder?
(Give your answer in base 10. Assume unsigned numbers)

a) 4
b) 1,111
c) 15
d) 16
e) 4000



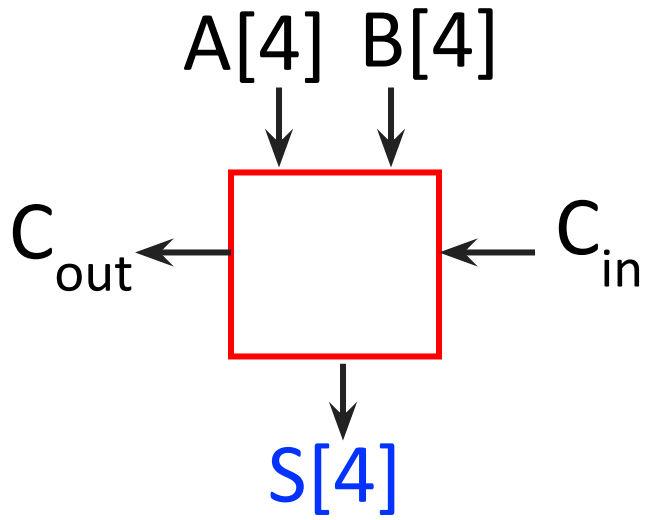


PollEv.com/cs3410

PollEV Question #5

What's the largest sum you can calculate with a 4 bit adder?
(Give your answer in base 10. Assume unsigned numbers)

a) 4
b) 1,111
c) 15
d) 16
e) 4000



Binary Subtraction

Why create a new circuit?

Just use addition using two's complement math

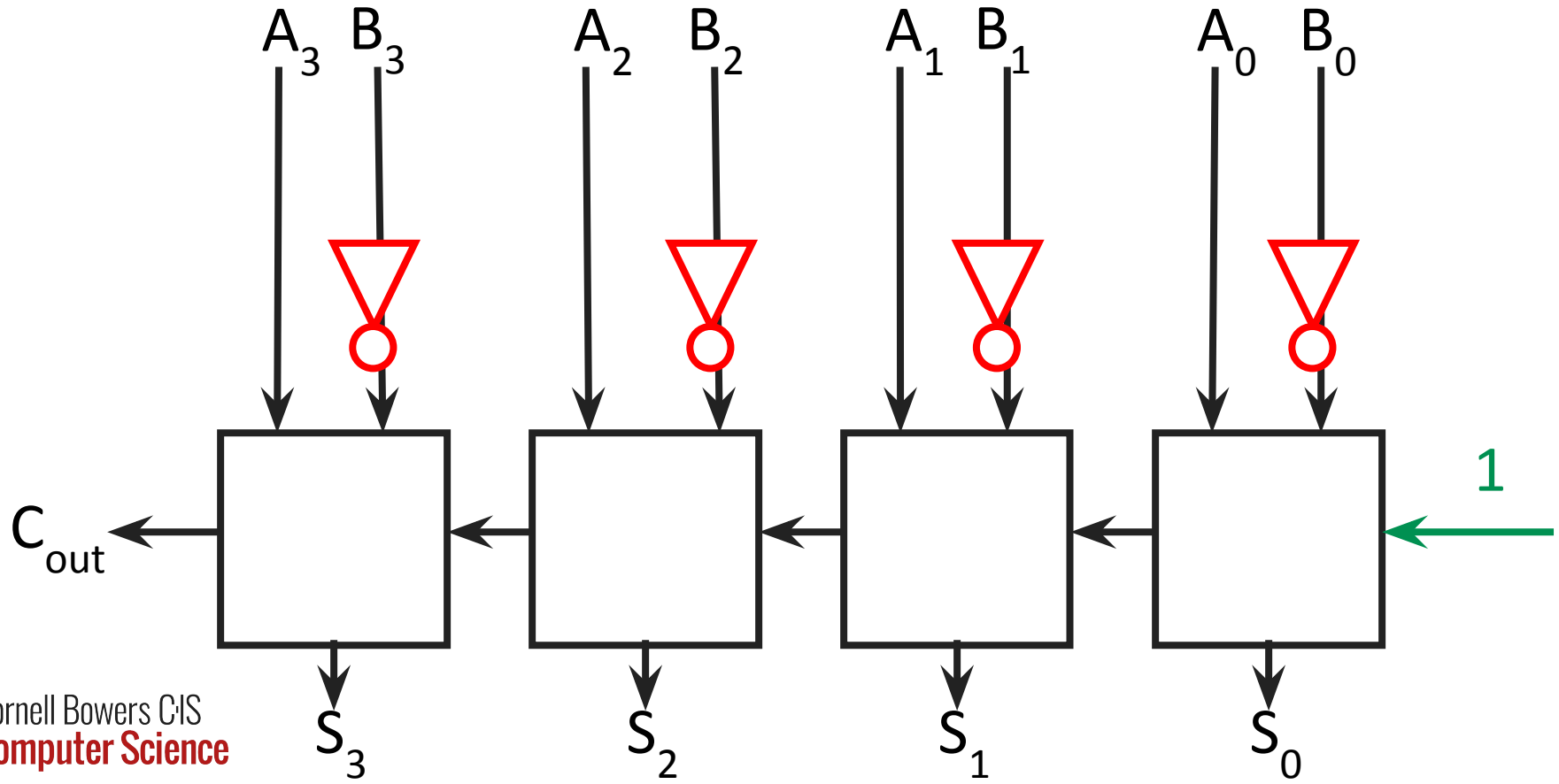
- How?

Binary Subtraction

Two's Complement Subtraction

- Subtraction is addition with a negated operand
  - Negation is done by inverting all bits and adding one

$$A - B = A + (-B) = A + (\overline{B} + 1)$$

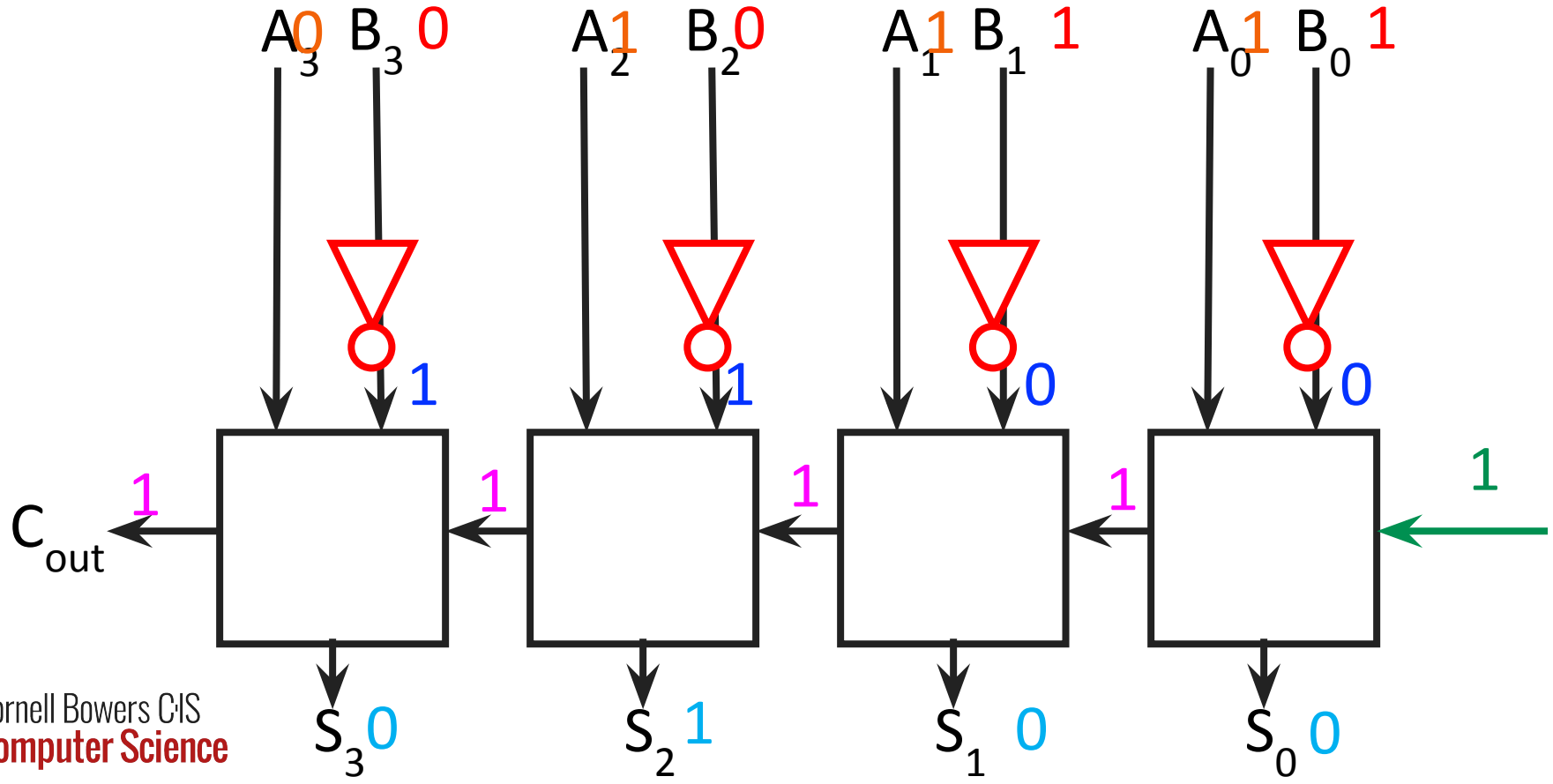


Binary Subtraction

Two's Complement Subtraction

- Subtraction is addition with a negated operand
  - Negation is done by inverting all bits and adding one

$A - B = A + (-B) = A + (\overline{B} + 1)$ E.g. $7 - 3 = 4 \rightarrow 7 + (-3) = 4$

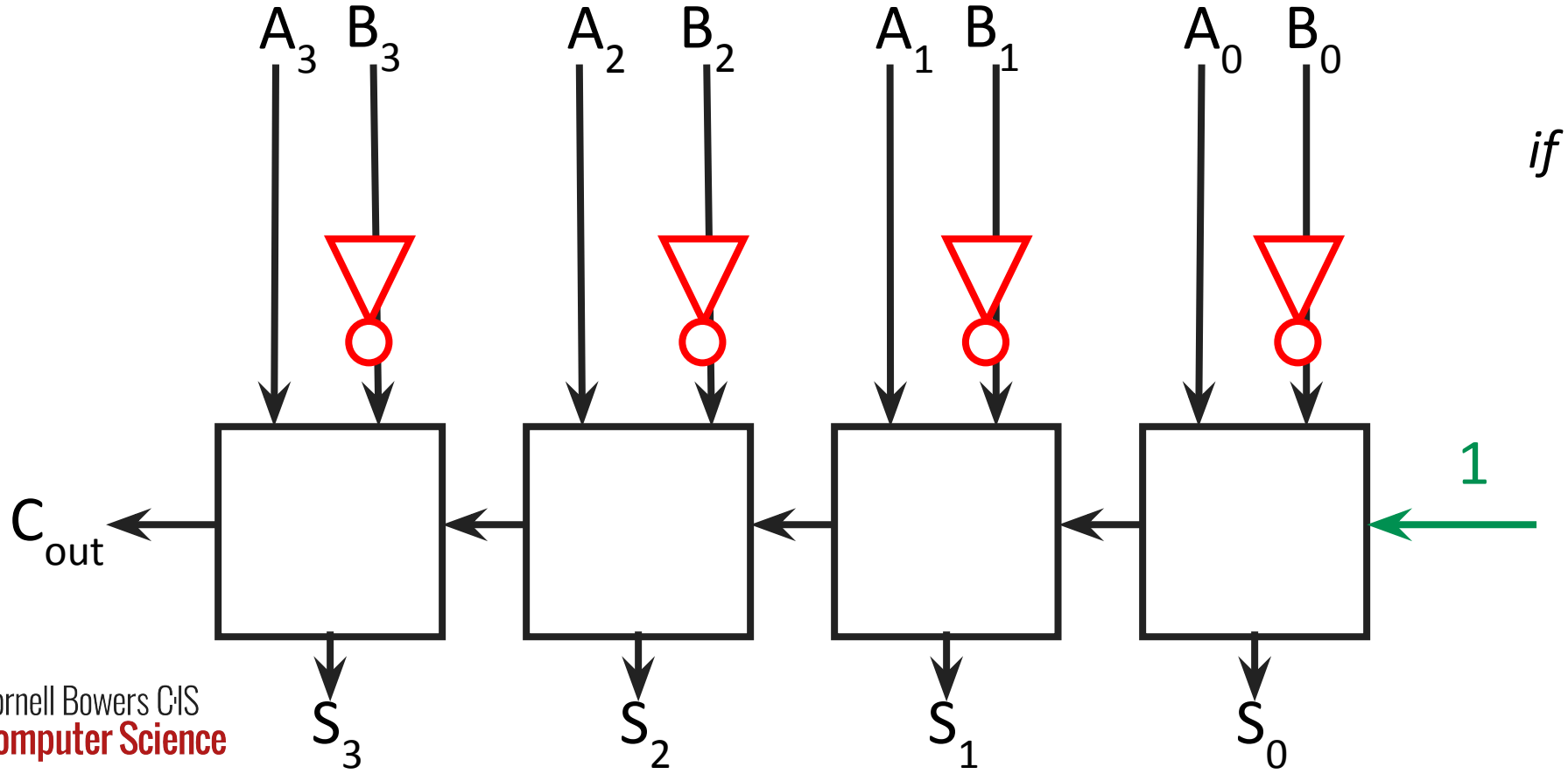


Binary Subtraction

Two's Complement Subtraction

- Subtraction is addition with a negated operand
  - Negation is done by inverting all bits and adding one

$$A - B = A + (-B) = A + (\overline{B} + 1)$$



Add or subtract with XOR gate

sub?	B_0	newB_0
0	0	0
0	1	1
1	0	1
1	1	0

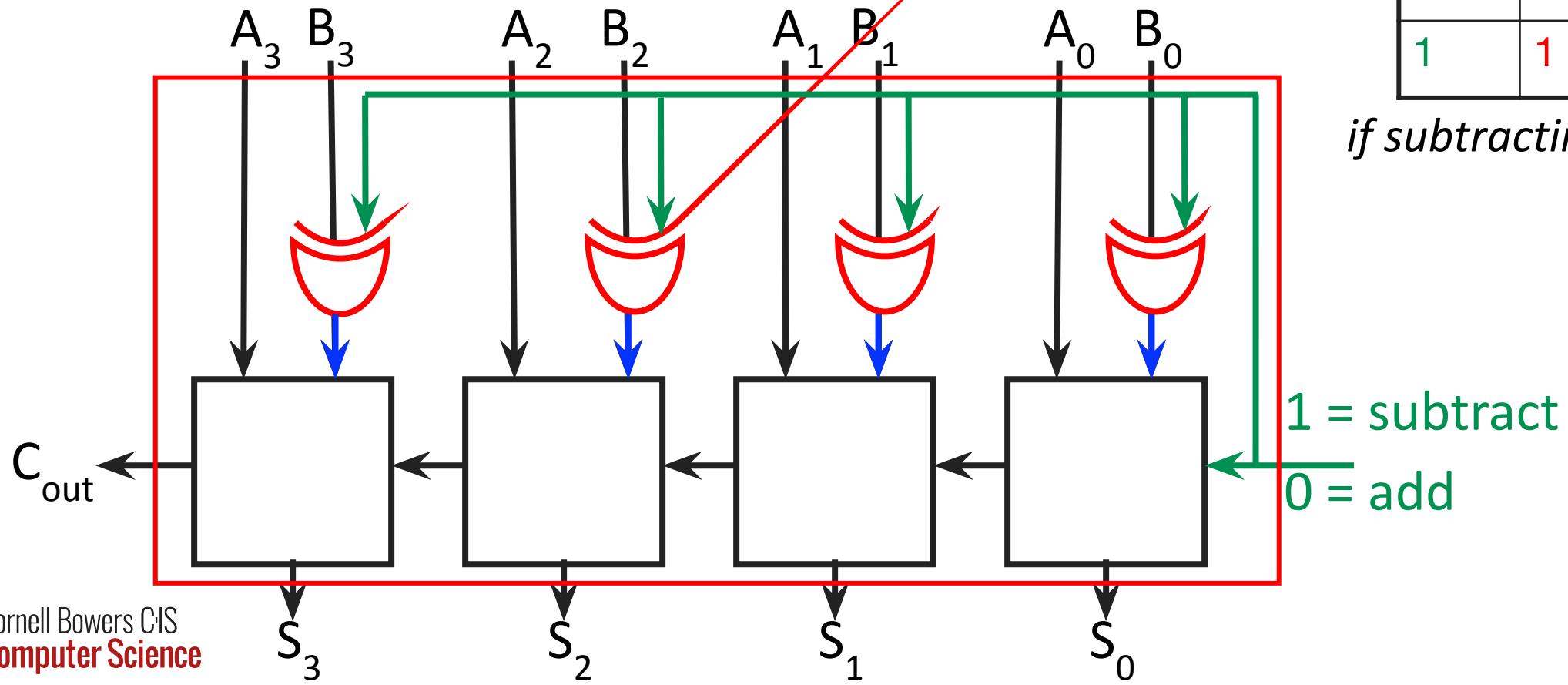
if subtracting, invert B_0

Binary Subtraction

Two's Complement Subtraction

- Subtraction is addition with a negated operand
  - Negation is done by inverting all bits and adding one

$$A - B = A + (-B) = A + (\overline{B} + 1)$$



Add or subtract with XOR gate

sub?	B_0	newB_0
0	0	0
0	1	1
1	0	1
1	1	0

if subtracting, invert B_0

Binary Subtraction

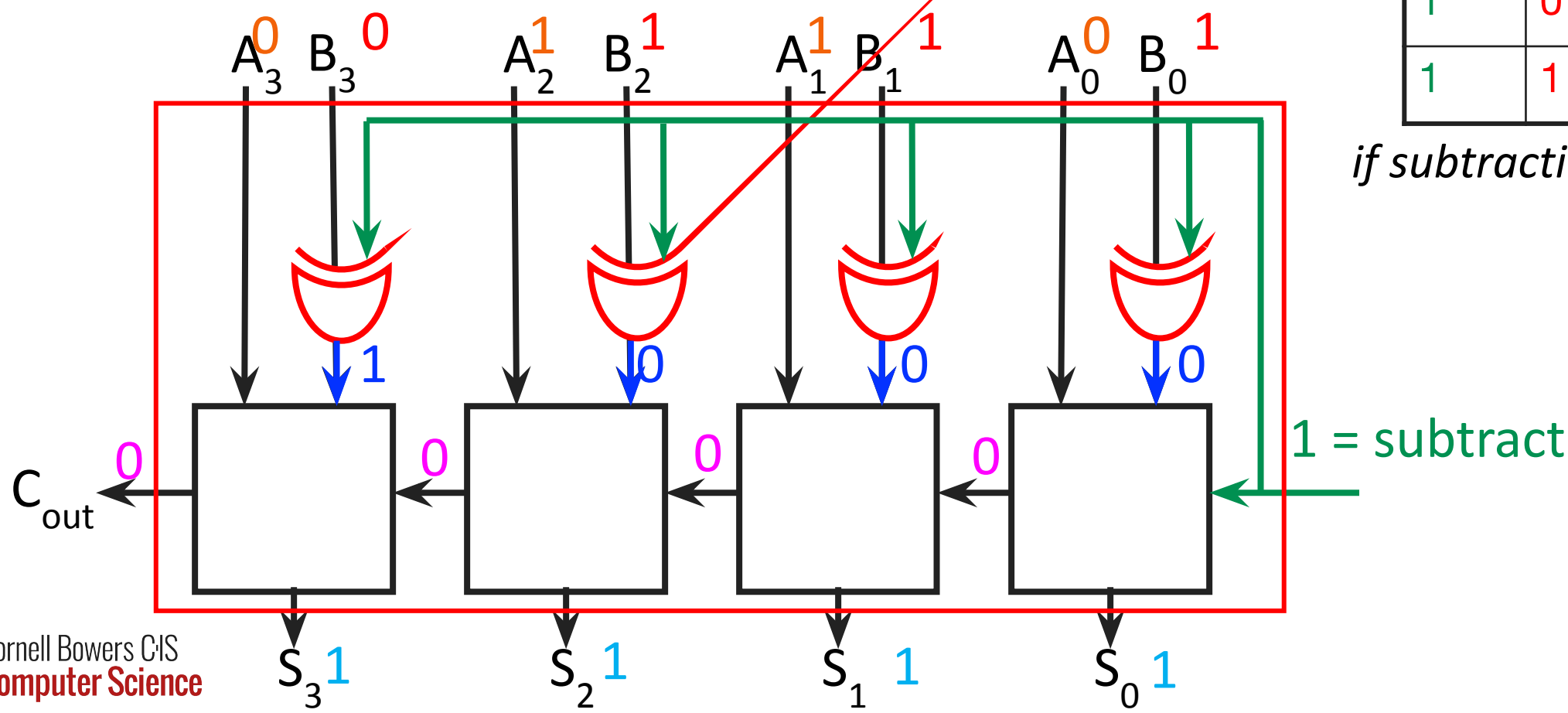
Two's Complement Subtraction

- Subtraction is addition with a negated operand
  - E.g. 6 – 7 = -1 □ 6 + (-7) = -1

Add or subtract with XOR gate

sub?	B_0	newB_0
0	0	0
0	1	1
1	0	1
1	1	0

if subtracting, invert B_0



Binary Subtraction

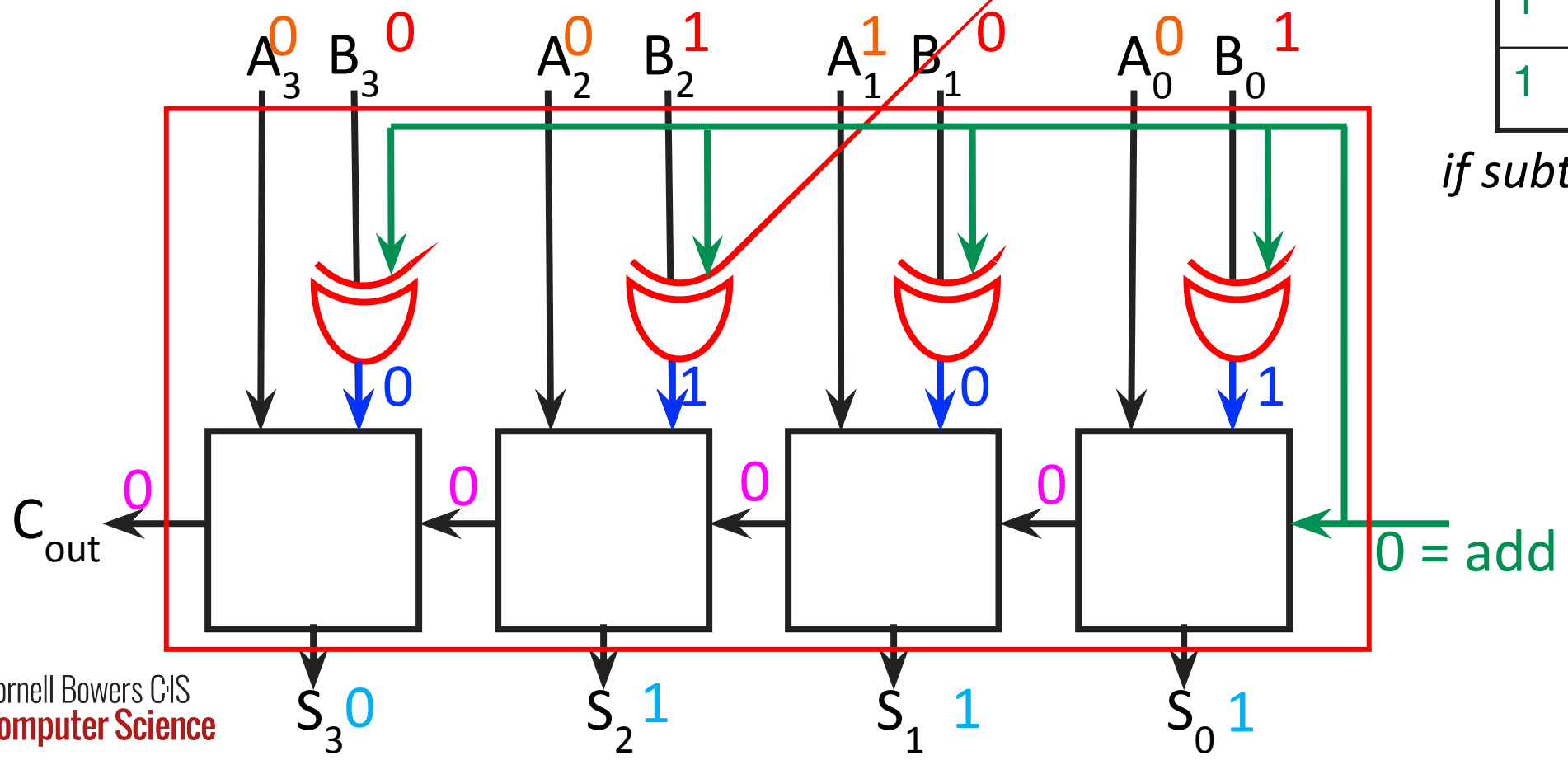
Two's Complement Subtraction

- Subtraction is addition with a negated operand
  - Addition still works! E.g. 2 + 5 = 7

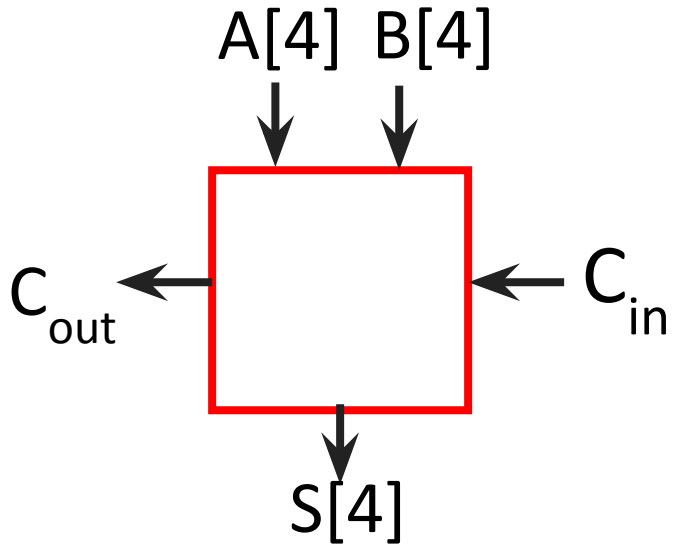
Add or subtract with XOR gate

sub?	B_0	newB_0
0	0	0
0	1	1
1	0	1
1	1	0

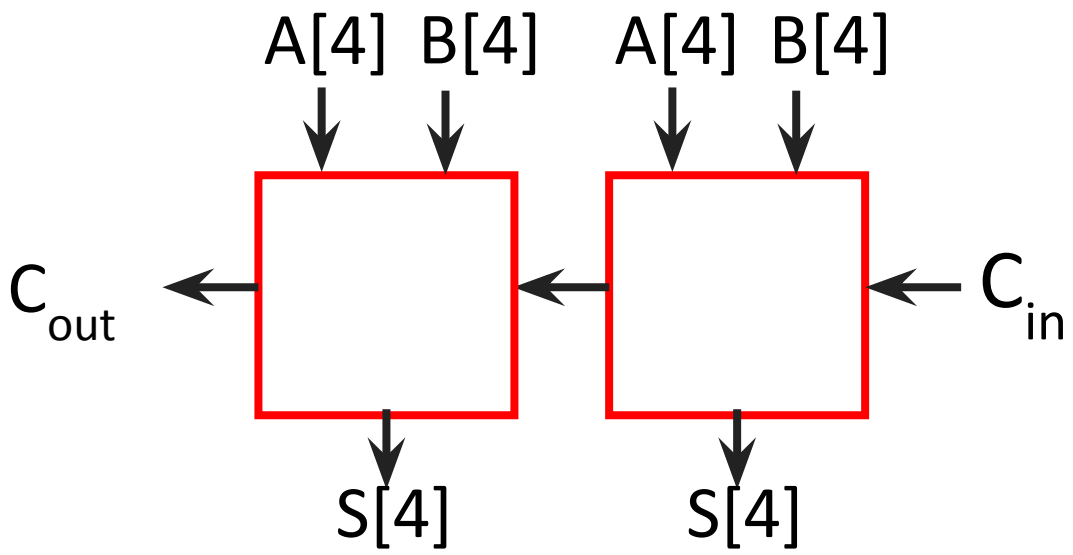
if subtracting, invert B_0



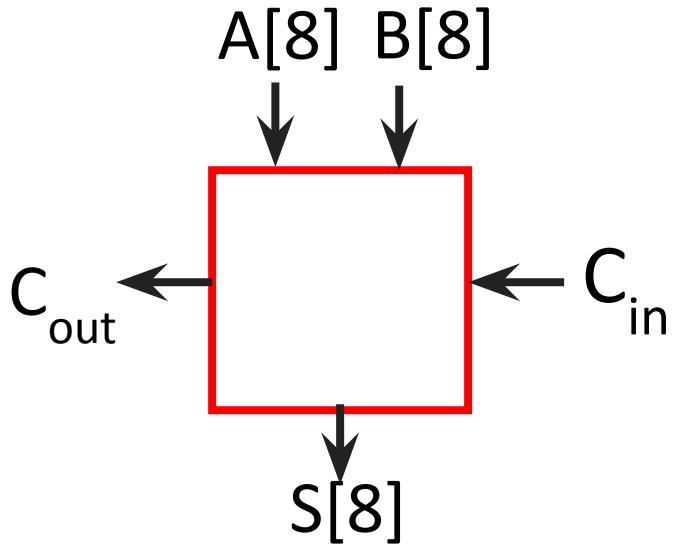
4-bit Adder with Two's Complement



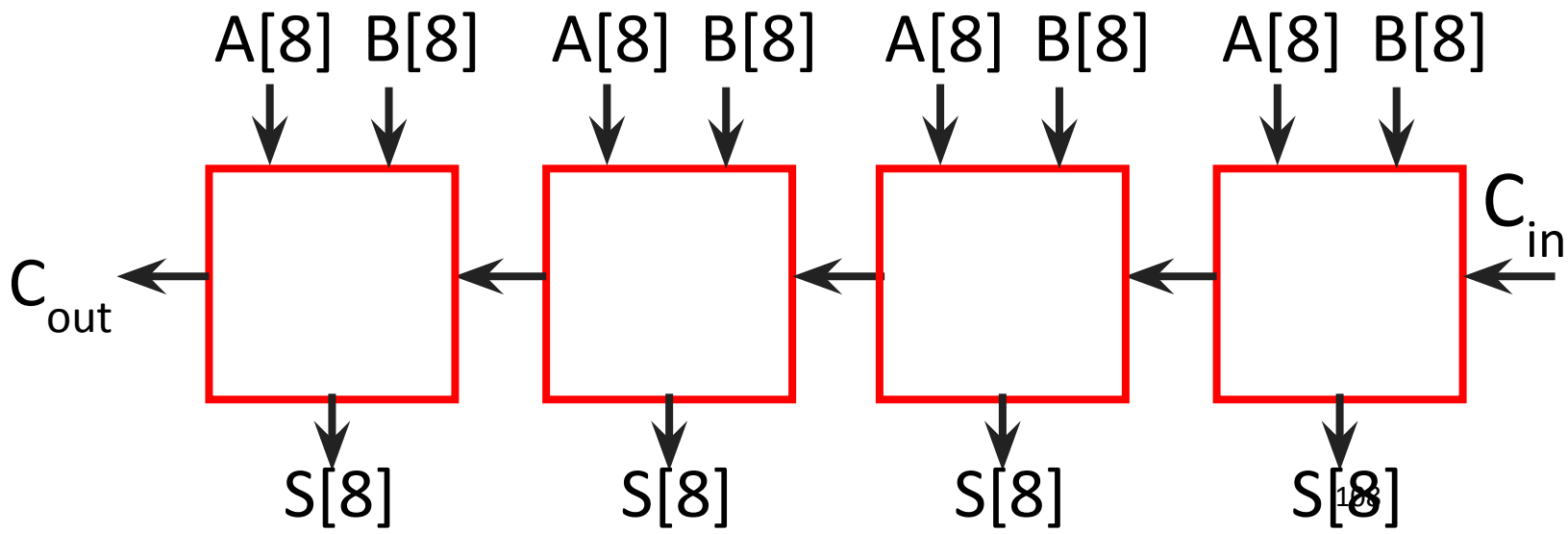
8-bit Adder with Two's Complement



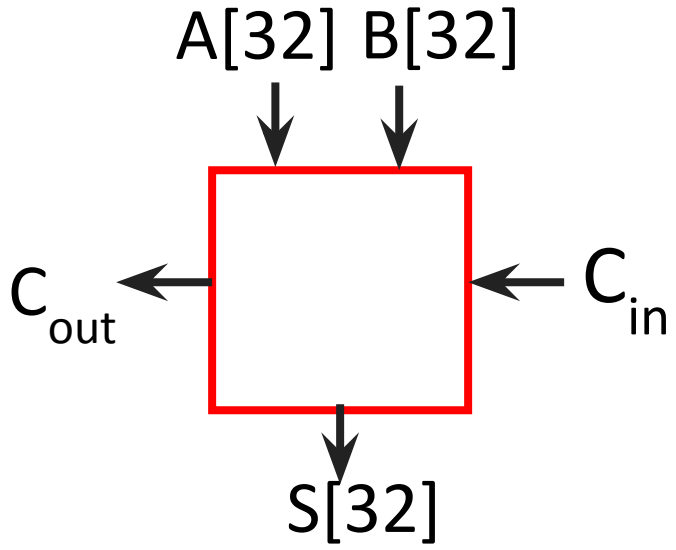
8-bit Adder with Two's Complement



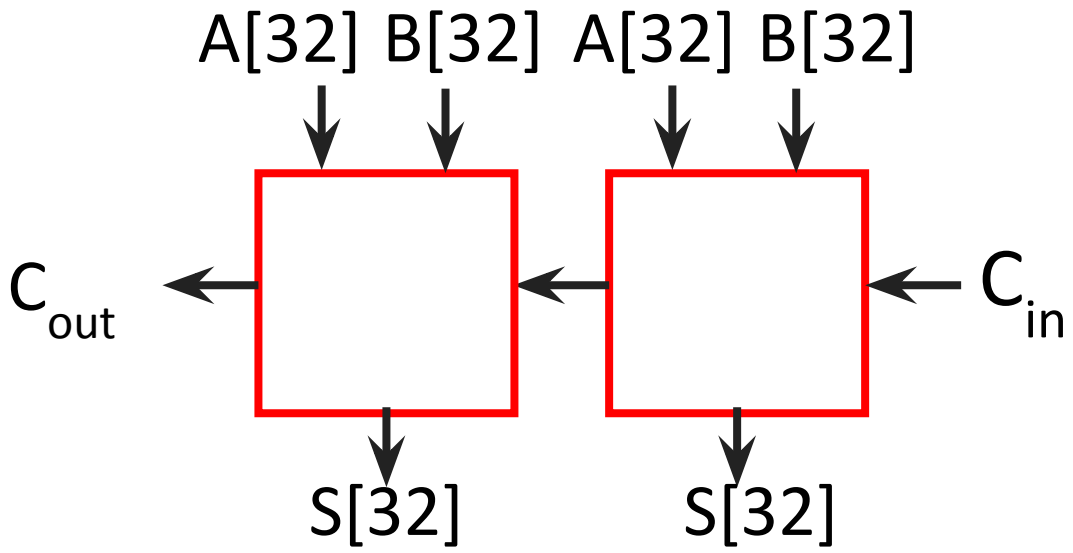
32-bit Adder with Two's Complement



32-bit Adder with Two's Complement



64-bit Adder with Two's Complement



64-bit Adder with Two's Complement