Hardware:
Materials, Circuits and Combinational Logic

For use in CSE6010 only

Not for distribution

Outline

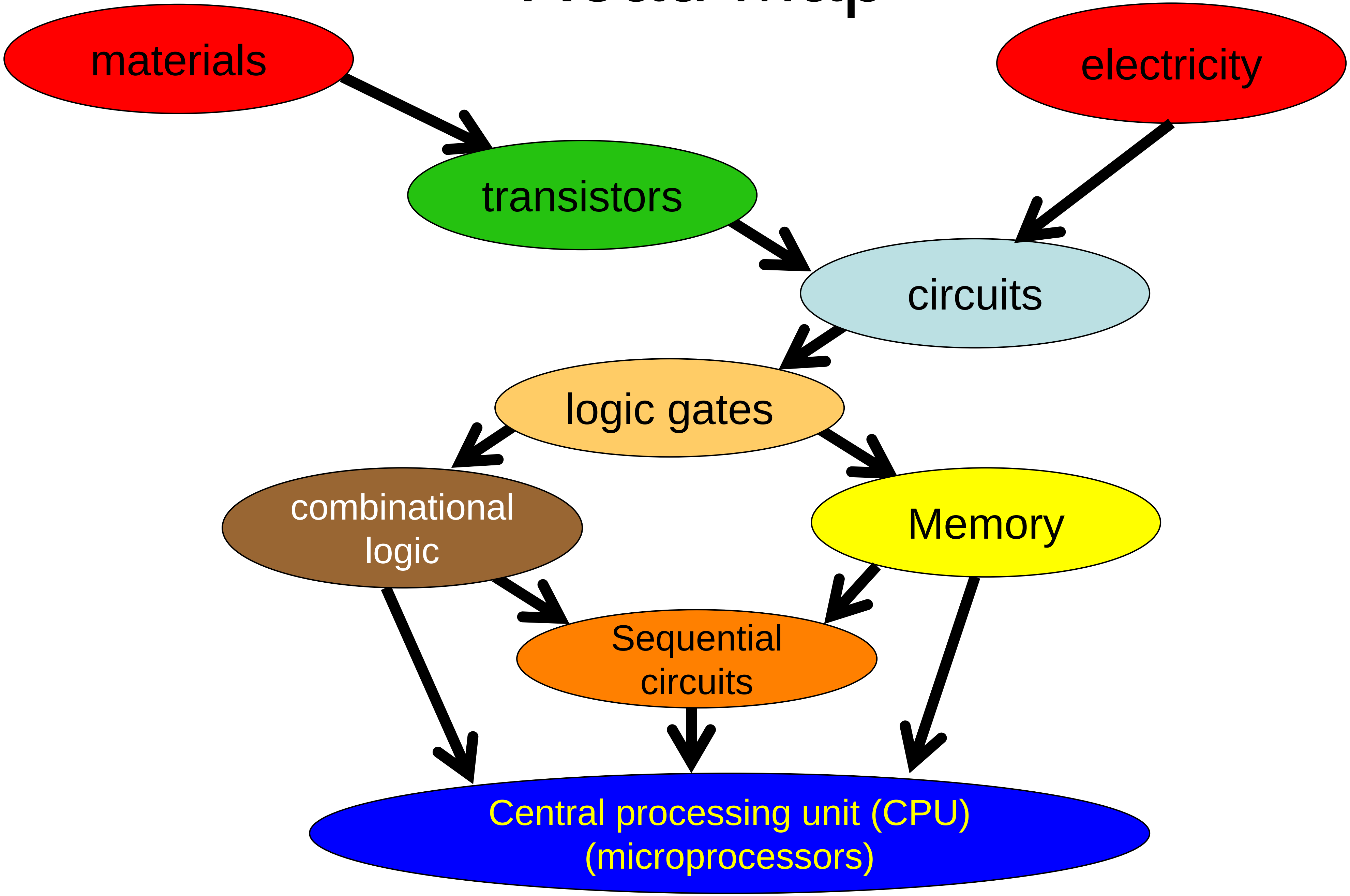
Application (word processor, simulator, web browser, …)
Algorithms (sorting, optimization, equation solver, …)
Programming Language (C, FORTRAN, Matlab, Java, …)
Operating System (UNIX, Windows, iOS, …)
Machine Instruction Set Architecture (Intel i86, ARM, …)
Machine Organization (Main memory, registers, adders, …)
Logic Gates (NAND, NOR, inverter, …)
Transistors (CMOS, NMOS, …)
Physics (Semiconductors) (electrons, holes, …)

Combinational Logic

Digital Circuits

Materials, Physics, Electricity

Road Map


materials
electricity
transistors
circuits
logic gates
combinational logic
Memory
Sequential circuits
Central processing unit (CPU) (microprocessors)

Hardware

We only need three components:

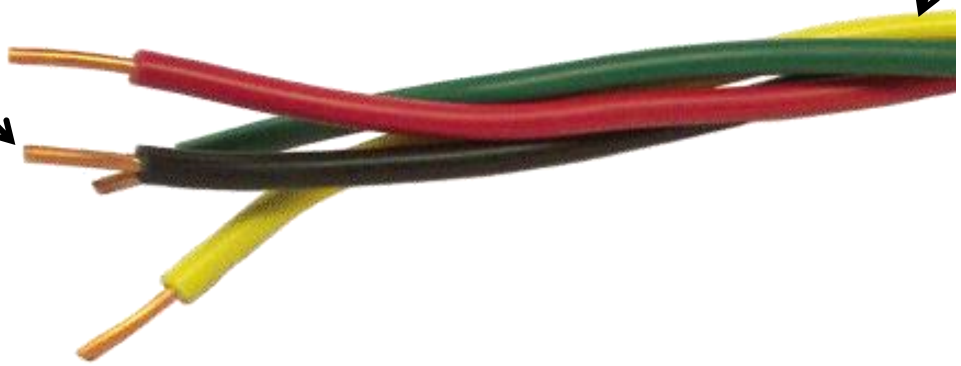
- Wire
- Power source
- Switch (transistor)

Wires

There are two types of materials:

Conductors: conduct electricity

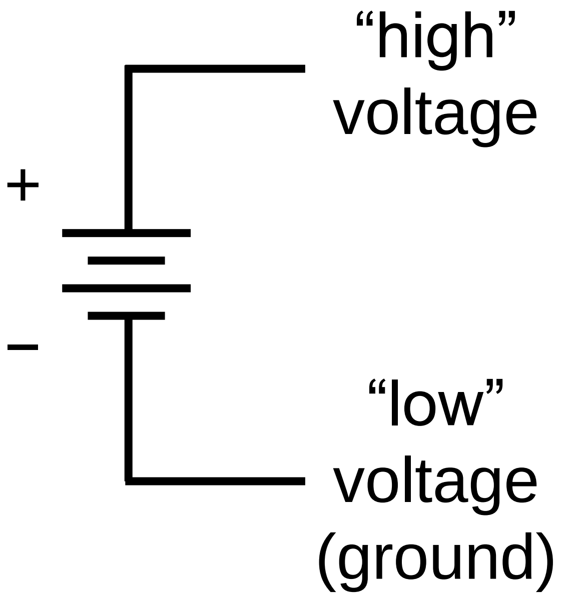
Insulators: do not conduct electricity



Electrical symbol (wire):

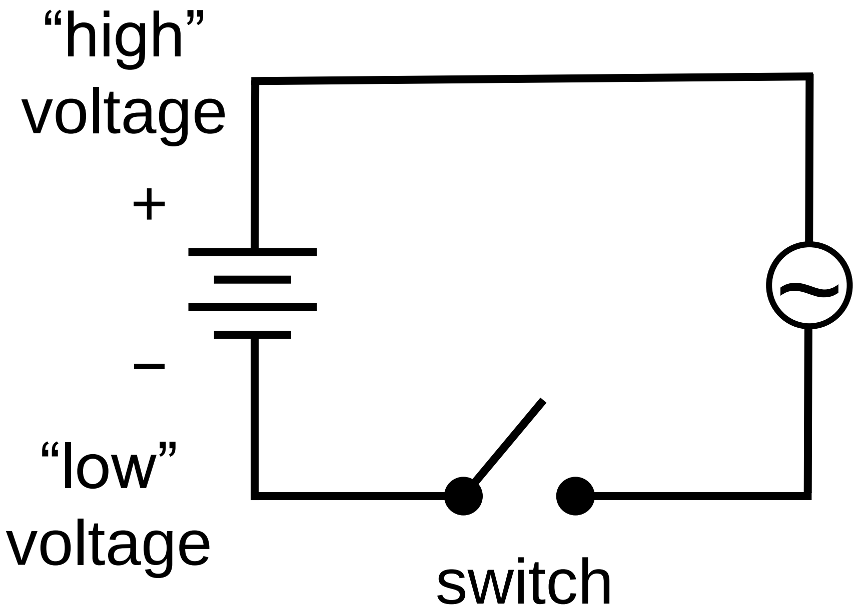
Power Source

Electrical symbol:



- Computers manipulate binary data
- Within a circuit, voltage is used to represent data
  - At any instant, a wire has either a high or low voltage
  - By convention, a high voltage represents binary ‘1’, and a low voltage (aka ground) represents binary ‘0’

Switch

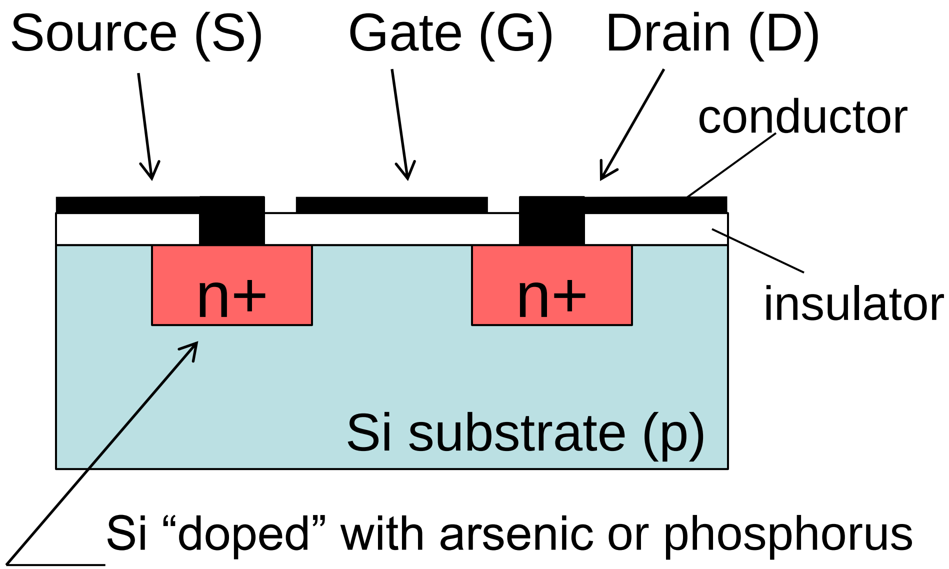


- Switch either open or closed
- We need an electronically controlled switch

Transistor (Switch)

Semiconductors (e.g., silicon): conduct electricity, sometimes

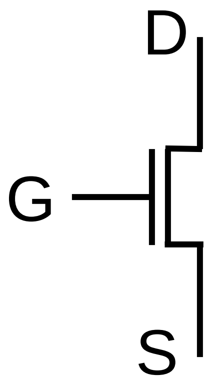
Metal Oxide Semiconductor, Field Effect Transistor (MOSFET)



Si "doped" with arsenic or phosphorus
effectively creates supply of extra electrons

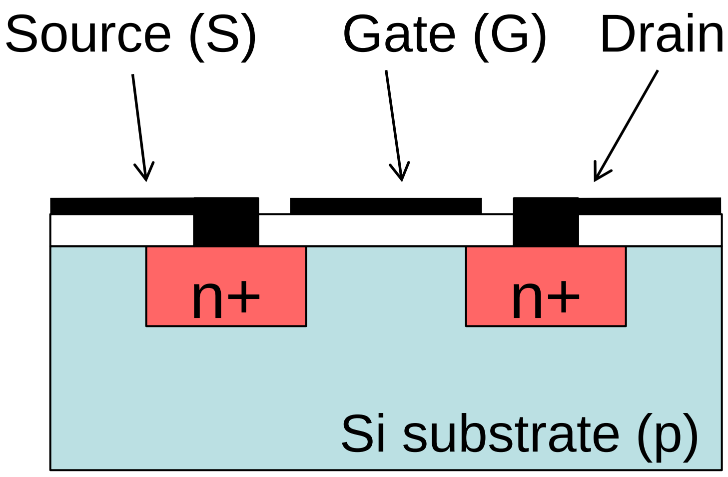
- Silicon (Si) in natural form does *not* conduct electricity
- It can be made into a conductor by implanting positive or negative ions in selected areas of the Si wafer (doping)

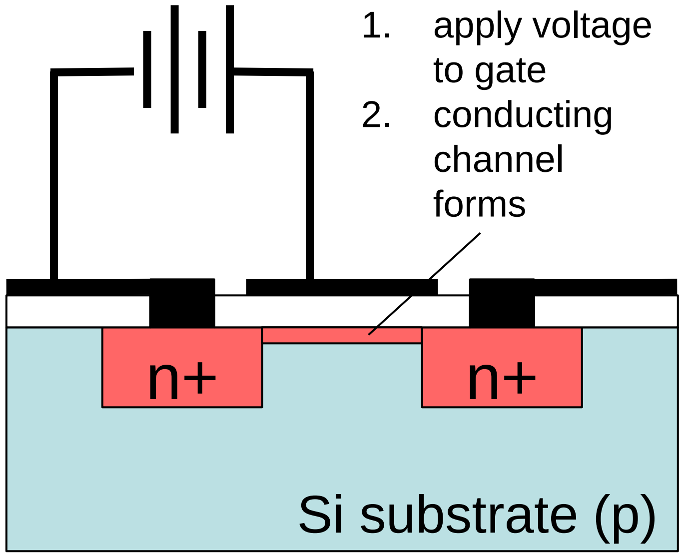
Electrical symbol:



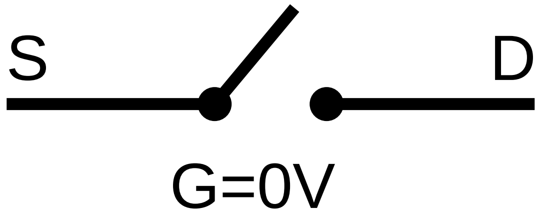
MOSFET: Operation

(metal-oxide-semiconductor field-effect transistor)

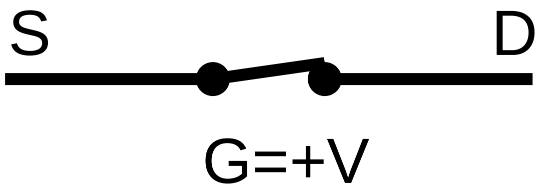




Conduct electricity (between source and drain)?



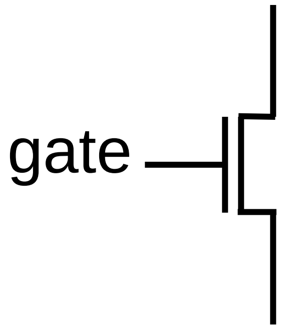
n-channel
MOS transistor



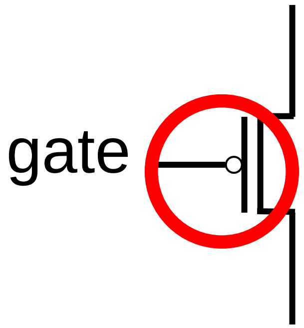
- Voltage-controlled switch
  - No voltage (logic 0) on gate leaves the switch open
  - Apply voltage (logic 1) to gate to close switch

MOSFET as a Circuit Component

n-type



p-type



Gate = “low” voltage (logic 0)

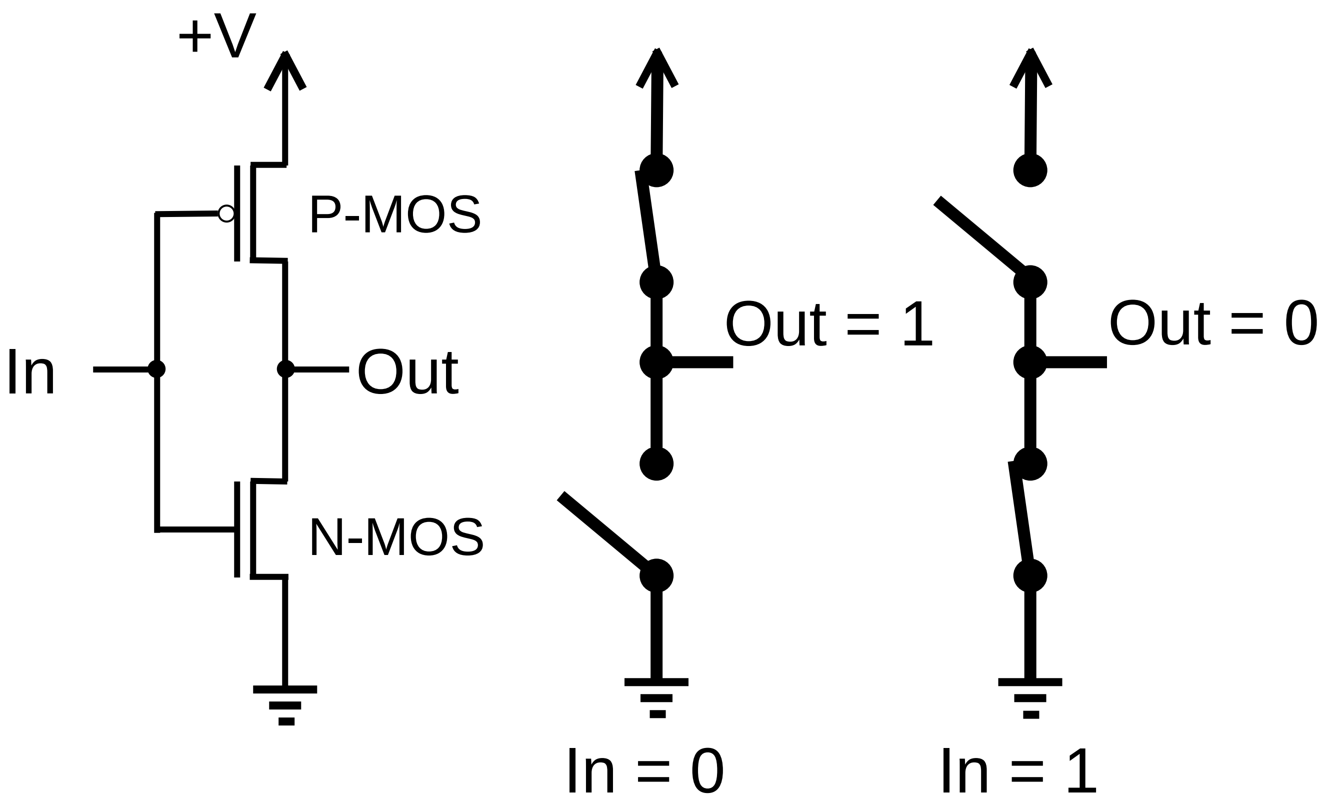
Gate = “high” voltage (logic 1)

Gate = “low” voltage (logic 0)

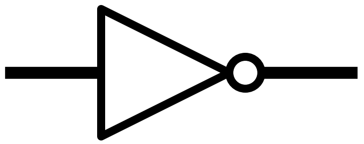
Gate = “high” voltage (logic 1)

	n-type transistor	**p-type transistor**
Low voltage (0)	Open (Non-conducting)	Closed (Conducting)
High voltage (1)	Closed (Conducting)	Open (Non-conducting)

Complementary MOS (CMOS)



Inverter

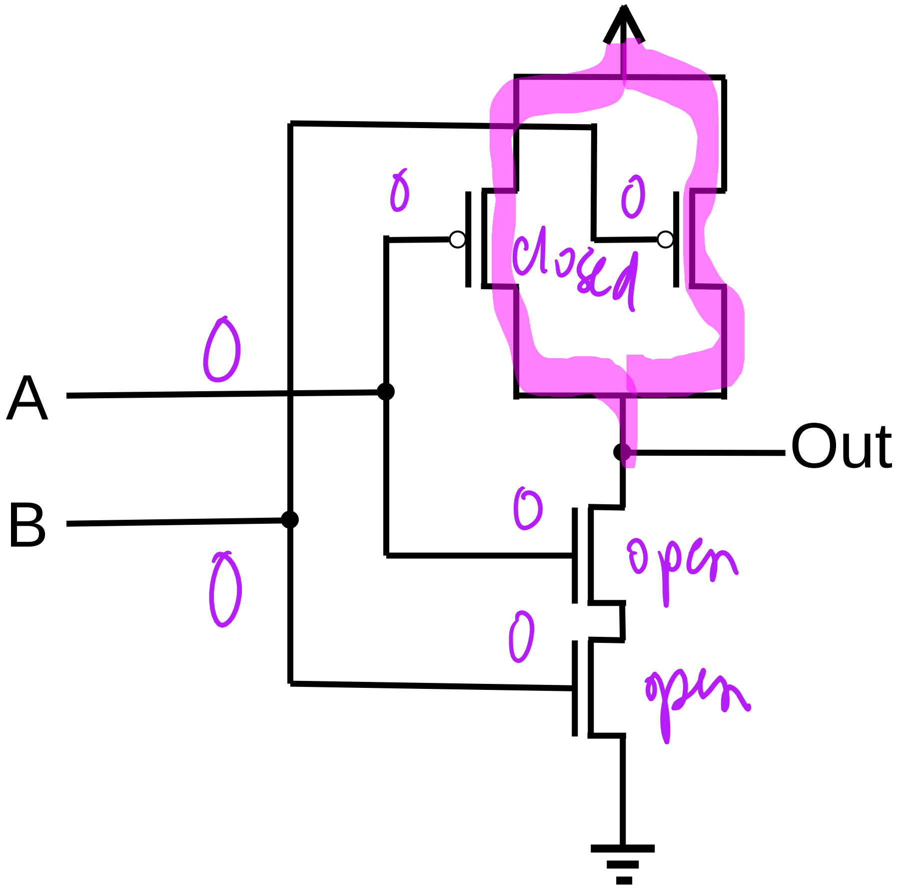


In	Out
0	1
1	0

Truth table

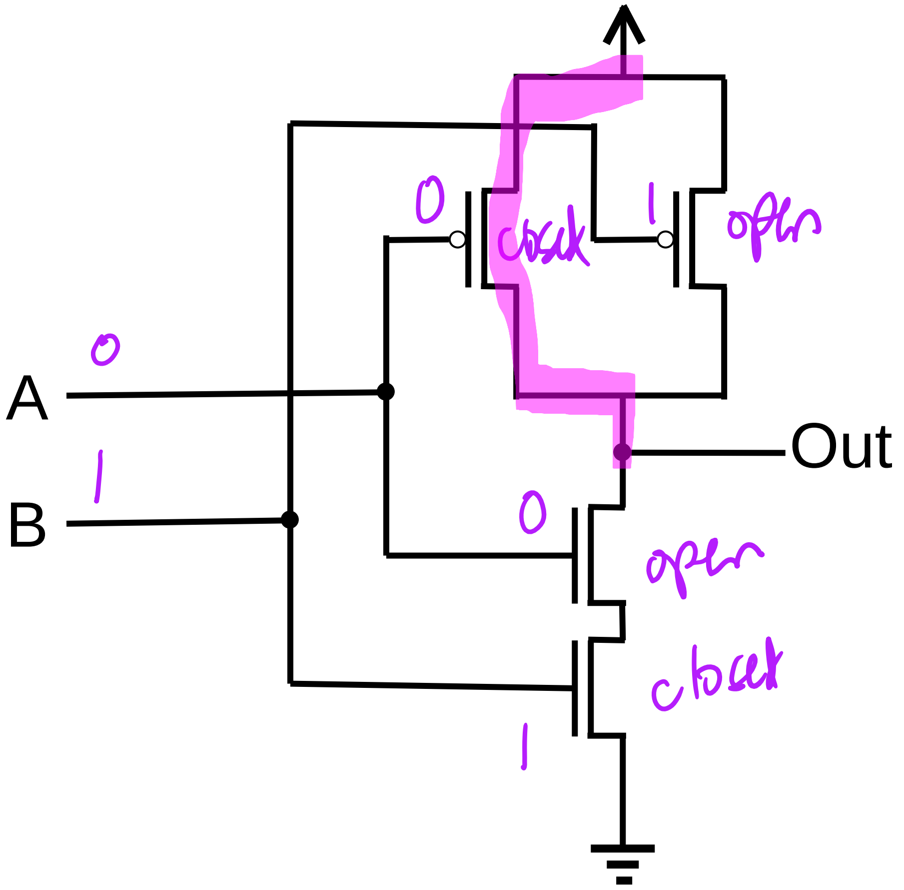
- CMOS is the dominant technology used for integrated circuits today
- Low power consumption

CMOS Circuit



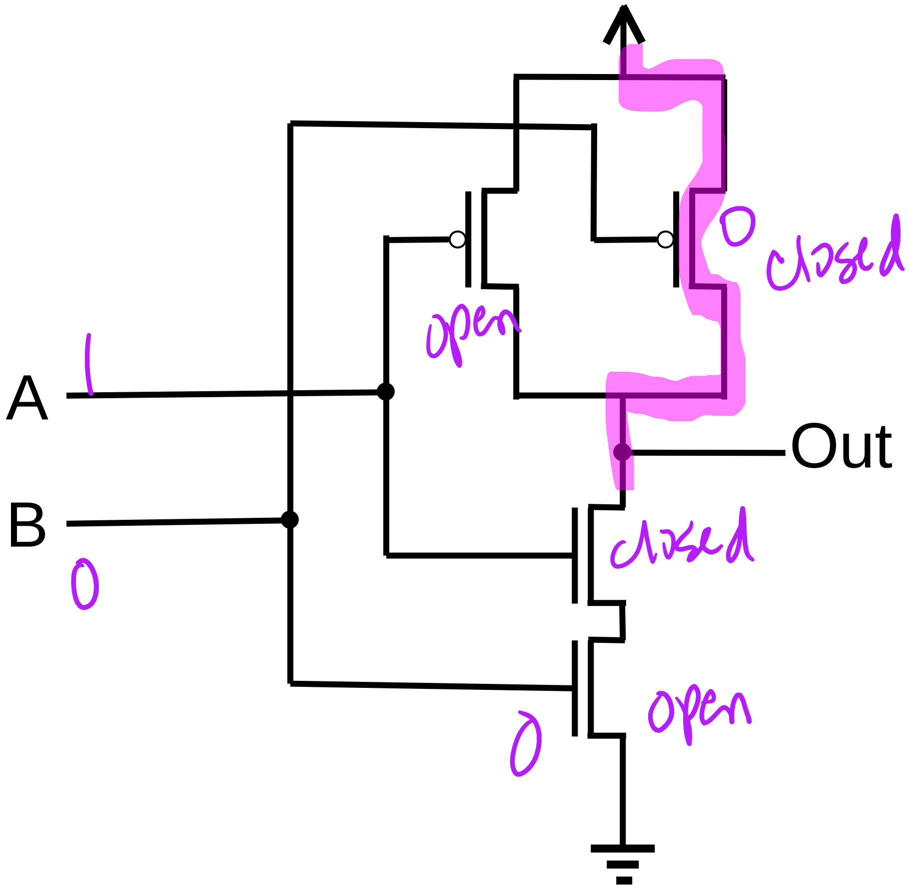
A	B	Out	$\overline{\text{Out}}$
0	0	1	0

CMOS Circuit



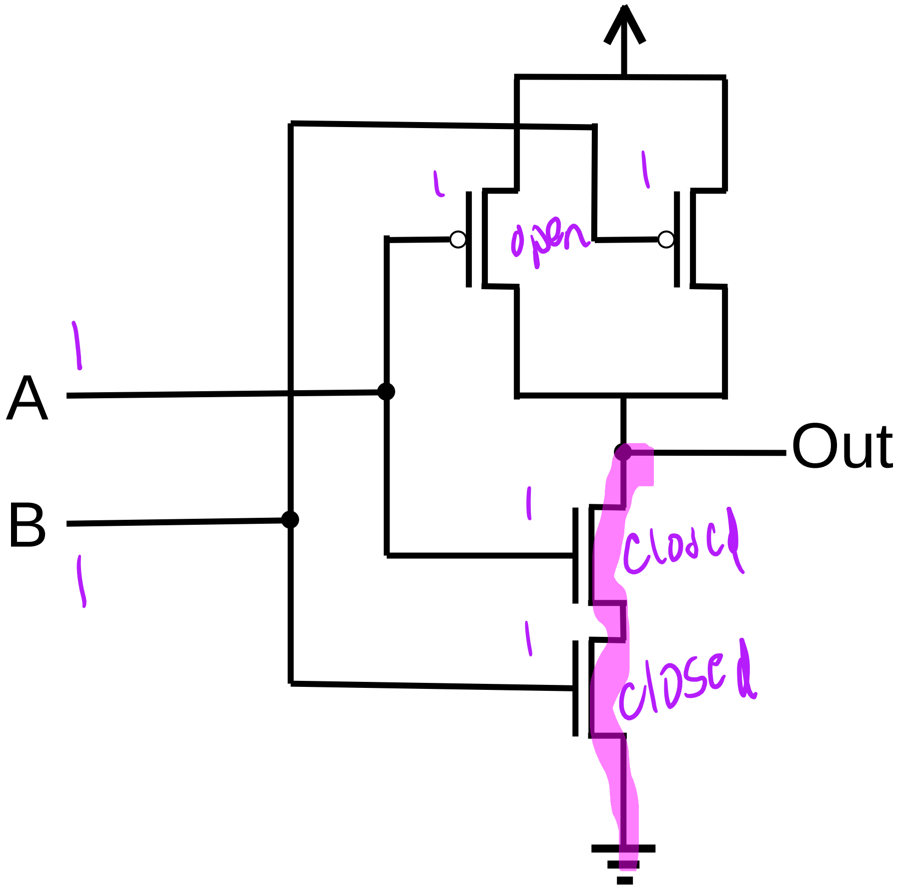
A	B	Out	$\overline{\text{Out}}$
0	0	1	0
0	1	1	0

CMOS Circuit

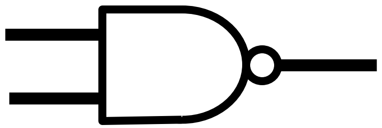


A	B	Out	$\overline{\text{Out}}$
0	0	1	0
0	1	1	0
1	0	1	0

CMOS Circuit



NAND (not-and)



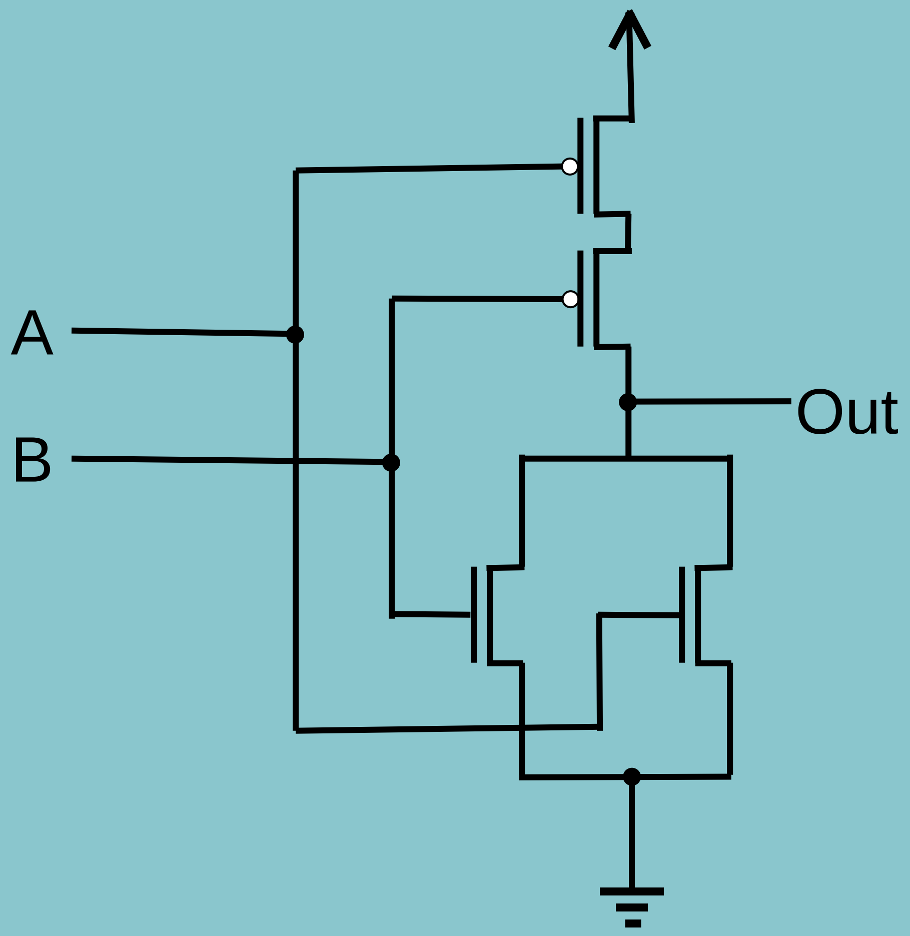
A	B	Out	$\overline{\text{Out}}$
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

Out = 0 if A and B are 1; otherwise, Out = 1

$\overline{\text{Out}}$ (complement Out) implements “AND” function

Circuit can extend to more inputs (e.g., 3-input NAND)

Another CMOS Circuit

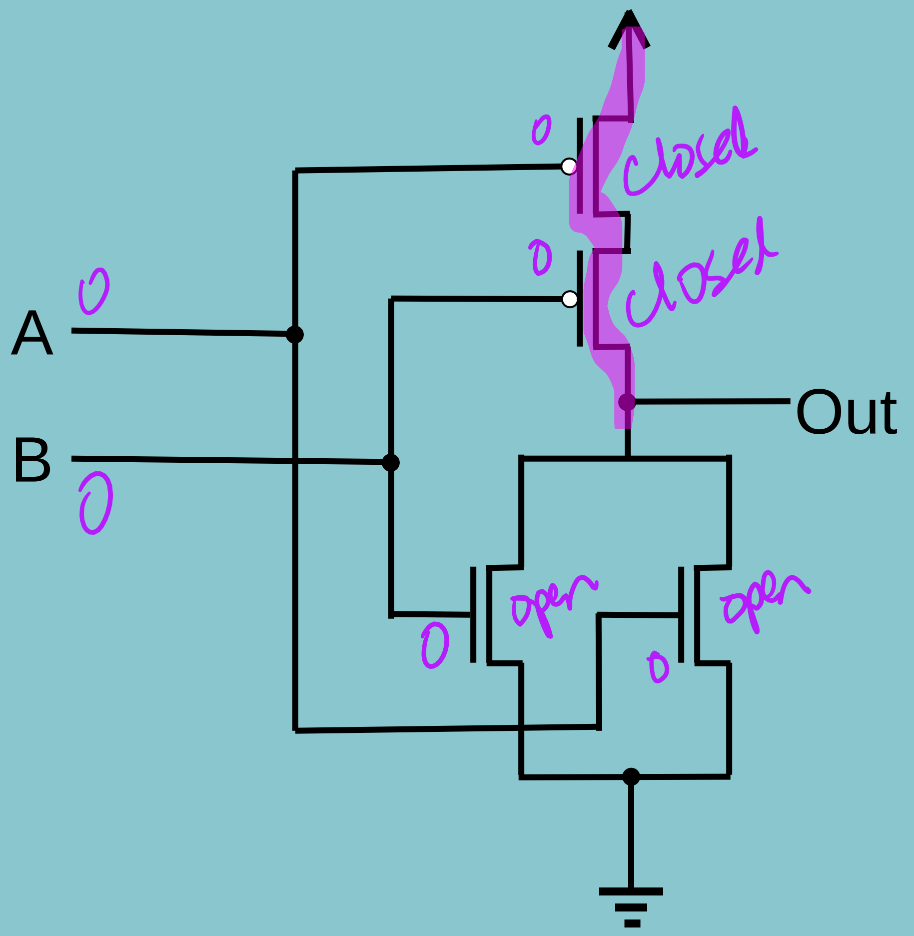


A	B	Out
0	0	
0	1	
1	0	
1	1	

Fill in the truth table for this circuit.

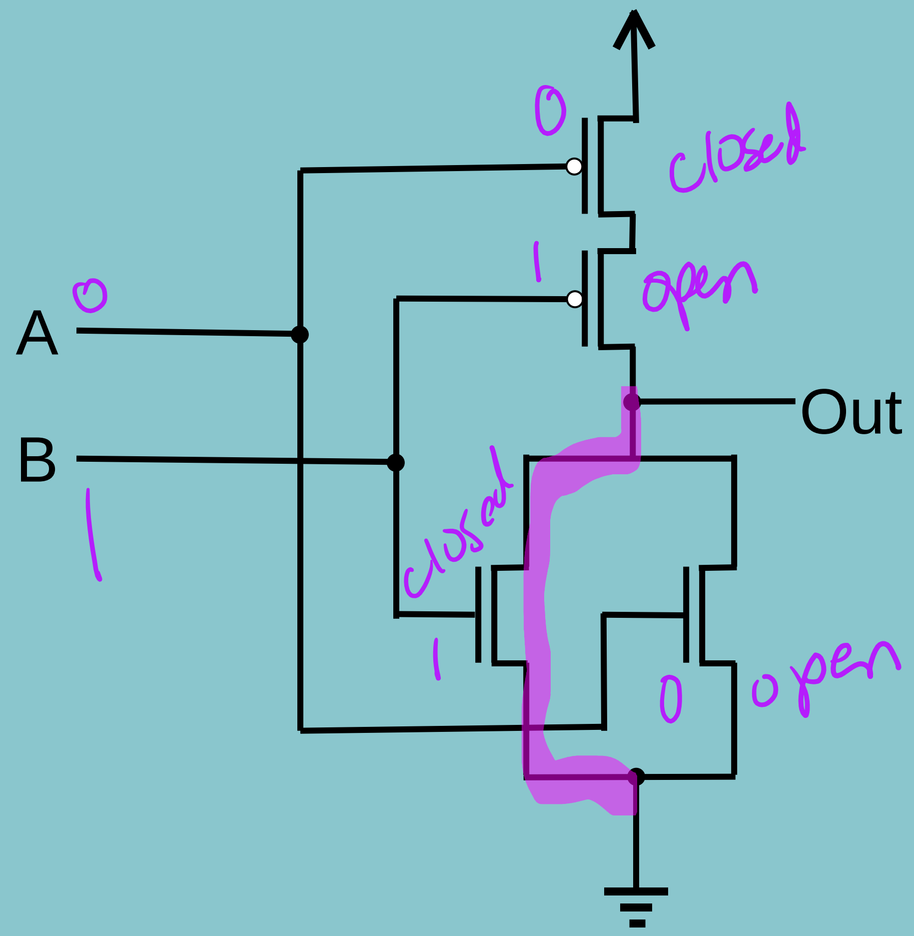
What does this circuit do?

Another CMOS Circuit



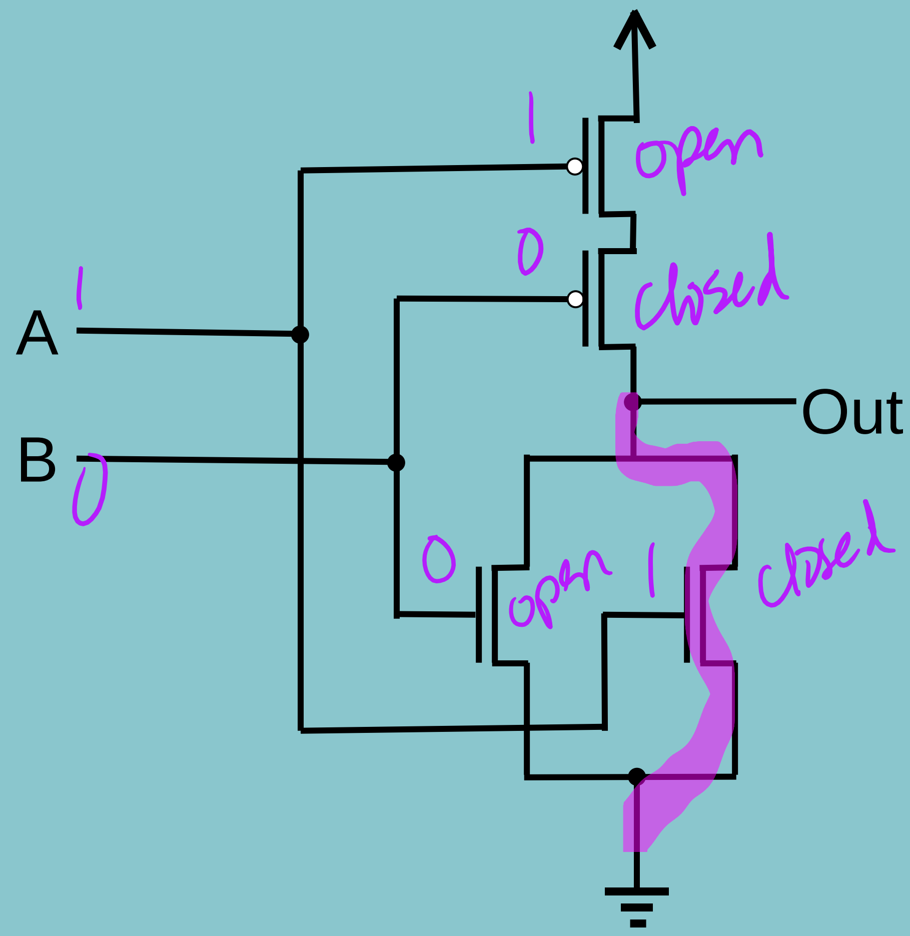
A	B	Out
0	0	1
0	1	
1	0	
1	1	

Another CMOS Circuit



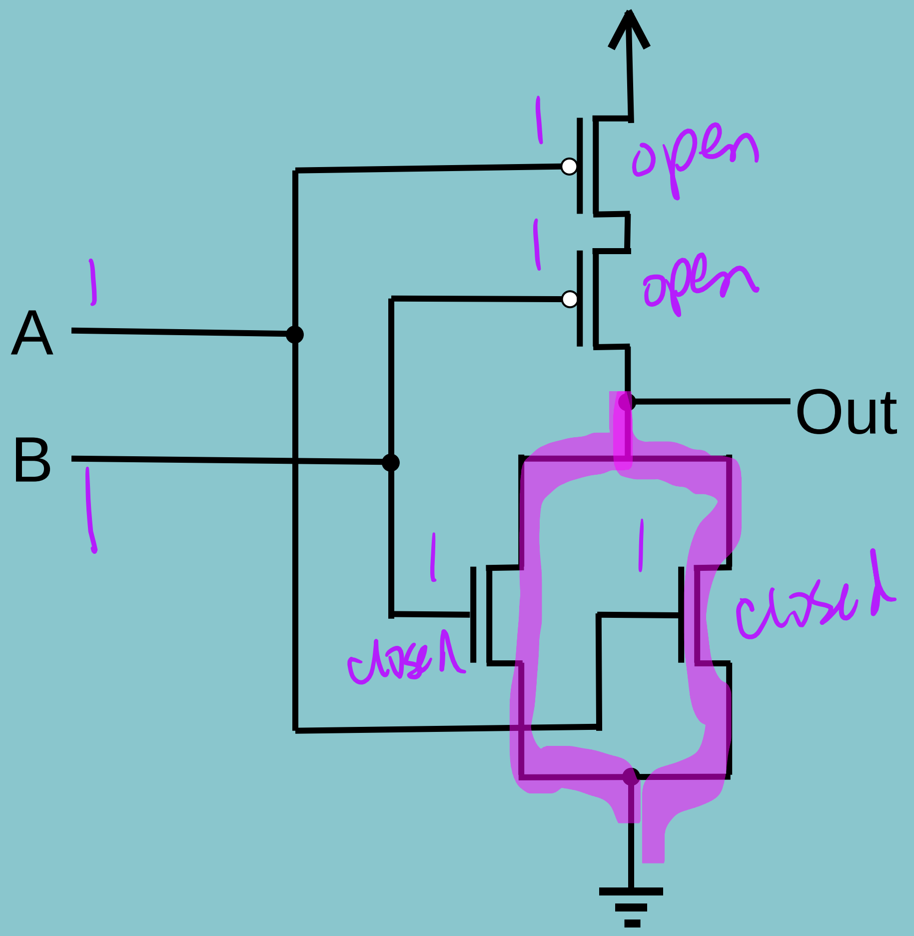
A	B	Out
0	0	1
0	1	0
1	0	
1	1	

Another CMOS Circuit

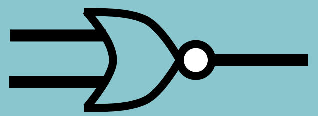


A	B	Out
0	0	1
0	1	0
1	0	0
1	1	

Another CMOS Circuit



NOR (not-or)



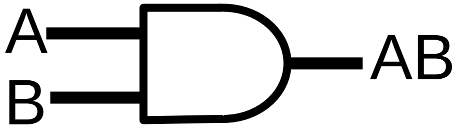
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Out = 0 if A or B are 1; otherwise, Out = 1

Circuit can extend to more inputs (e.g., 3-input NOR)

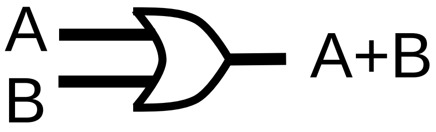
AND / OR Functions

- For circuits, it is easier to implement the NAND and NOR functions
- For people, it is easier to work with the AND and OR functions



AND

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

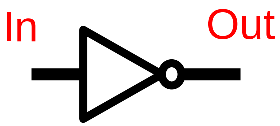


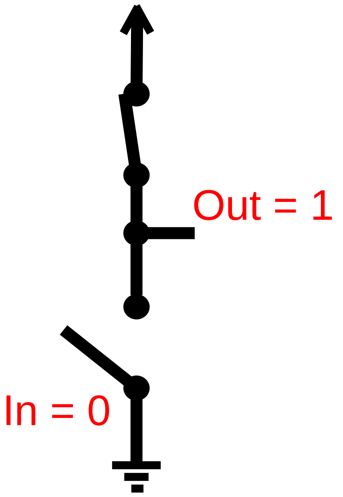
OR

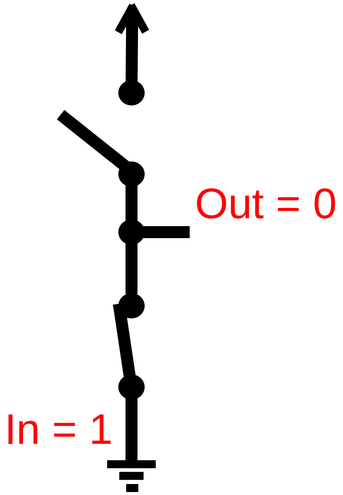
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

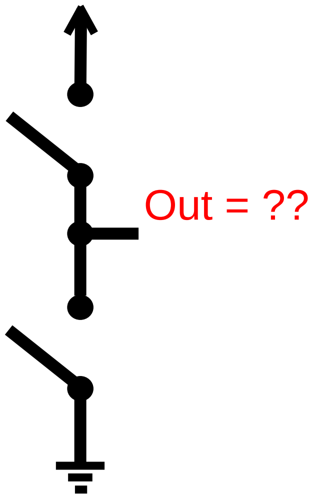
- AND (OR) can be implemented with NAND (NOR) plus an inverter

Circuits: Other Cases

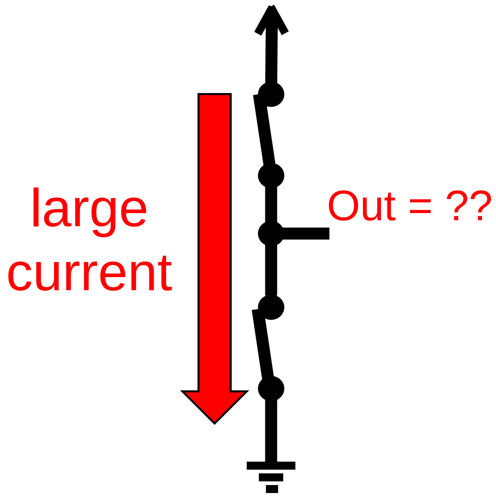




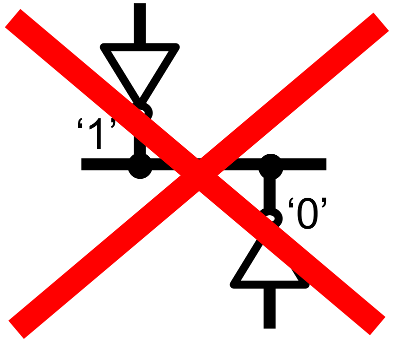


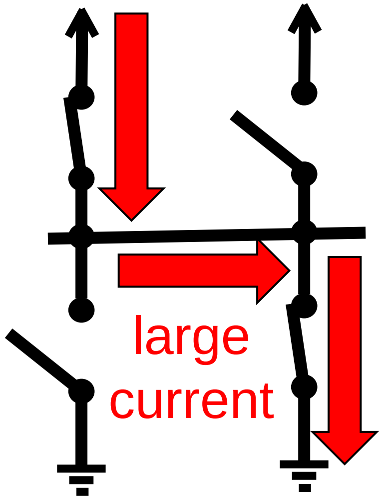


- Cannot occur with gates discussed so far, but if it did occur
  - Output neither ‘1’ nor ‘0’
  - Output said to be “floating” or “high impedance” (Hi-Z)
  - Hi-Z input to gate gives unpredictable results
- Sometimes useful in circuit design



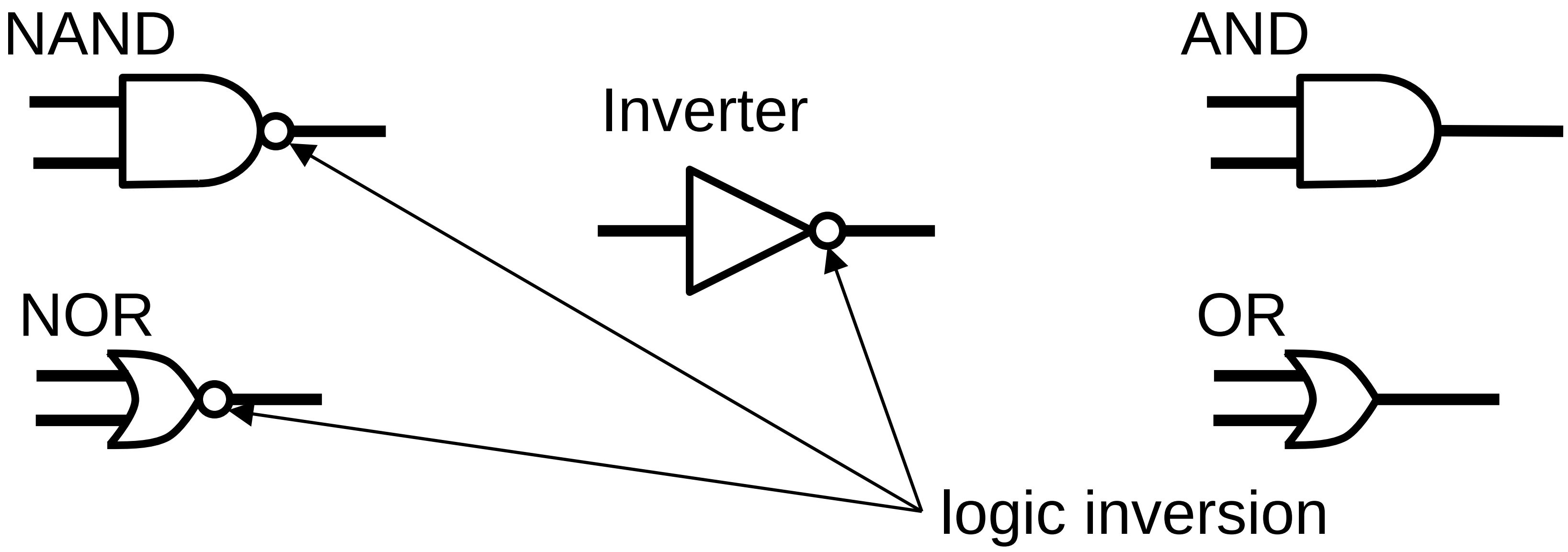
- “Short circuit”
- Permanent damage to circuit !





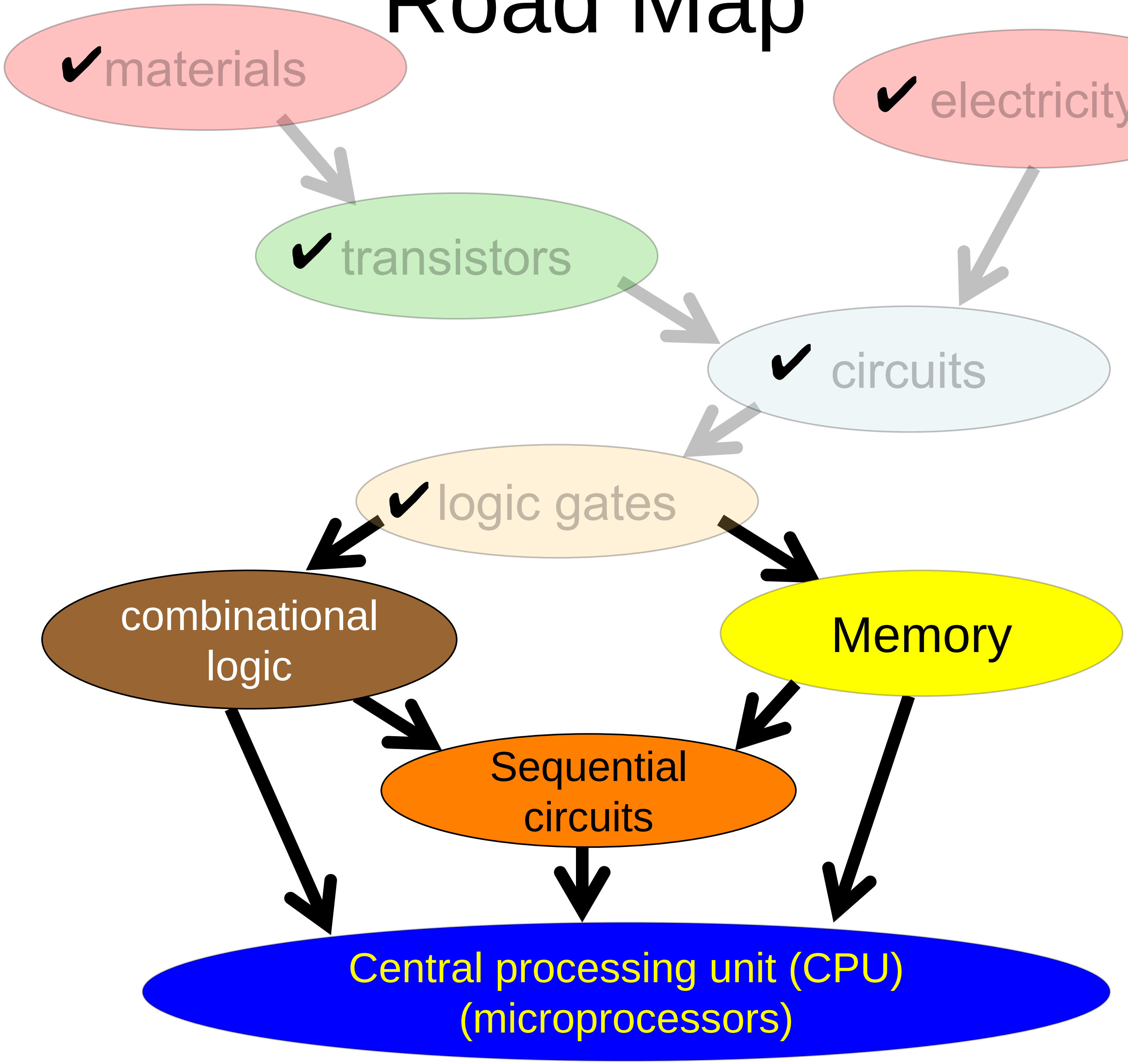
Gate outputs should never be directly wired together !

Summary: Gates
(commit this to your memory!)



- We can ignore lower level abstractions (electricity/materials, circuits, switches) and concentrate on logic gates
- Behavior completely specified by truth table
- Circle (or “bubble”) denotes logic inversion/complement, i.e., 0 becomes 1, 1 becomes 0)

Road Map



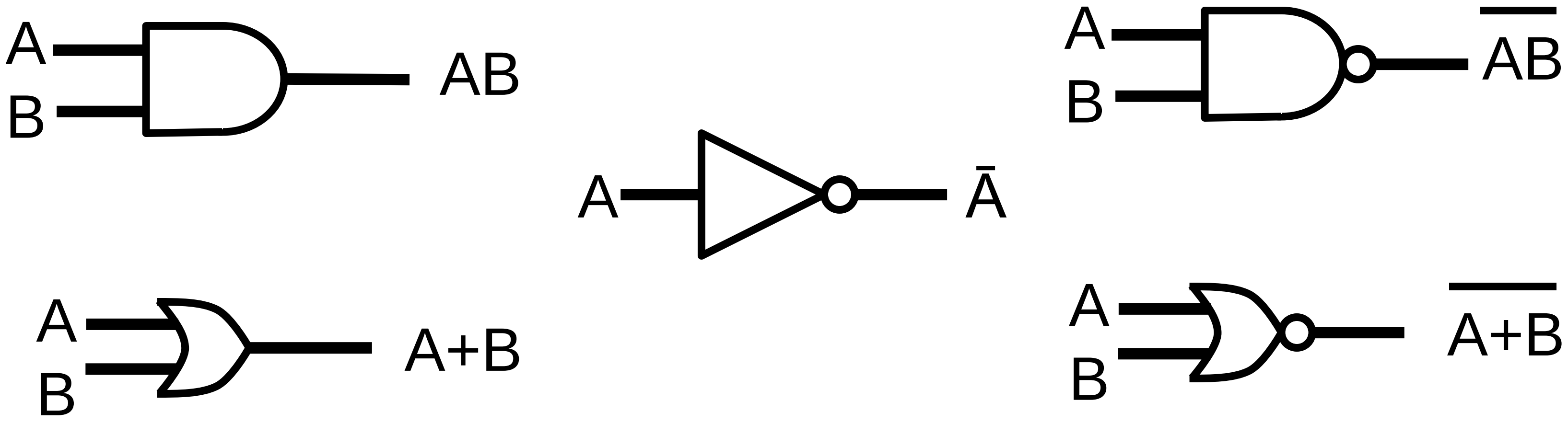
✔materials
✔ electricity
✔ transistors
✔ circuits
✔ logic gates
combinational logic
Memory
Sequential circuits
Central processing unit (CPU) (microprocessors)

Boolean Algebra

Need to be able to manipulate binary data (e.g., add)

Boolean Algebra: a mathematical system including

- Two values: 0 (false), 1 (true)
- Binary variables (one-bit signal): A, B, etc.
- Operators on binary variables
  - AND: AB (A and B, multiplication)
  - OR: A+B (A or B, addition)
  - NOT: $\bar{A}$ (not A, complementing)



Useful Boolean Algebra Laws

Given the listed Boolean algebra laws for AND, fill in the corresponding laws for “OR”.

A, B, and C are Boolean variables

Law	AND	OR
Identity	A 1 = A	
Null	A 0 = 0	
Idempotent	AA = A	
Complement	$\overline{\overline{A}} = A$ (not related to AND or OR)	
Commutative	AB = BA	
Associative	(AB)C = A(BC)	

Useful Boolean Algebra Laws

Given the listed Boolean algebra laws for AND, fill in the corresponding laws for “OR”.

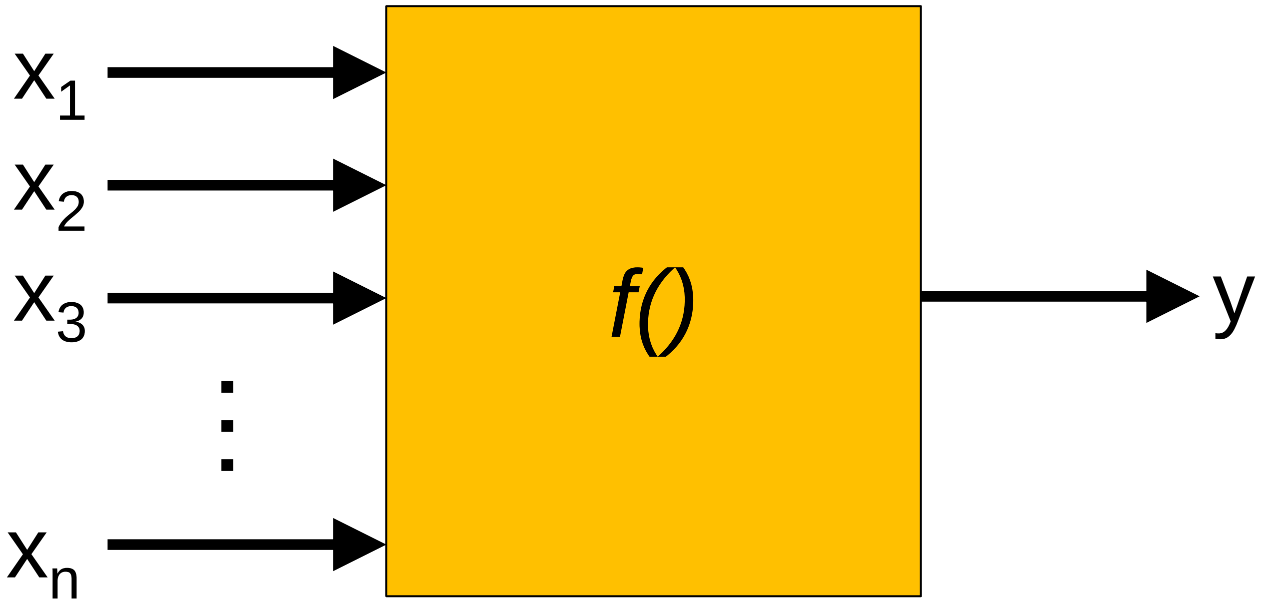
A, B, and C are Boolean variables

Law	AND	OR
Identity	A 1 = A	A + 0 = A
Null	A 0 = 0	A + 1 = 1
Idempotent	AA = A	A + A = A
Complement	$\bar{\bar{A}} = A$ (not related to AND or OR)	
Commutative	AB = BA	A + B = B + A
Associative	(AB)C = A(BC)	(A+B)+C = A+(B+C)

Boolean Functions

Goal: Design a circuit to implement an *arbitrary* Boolean function on n binary inputs:

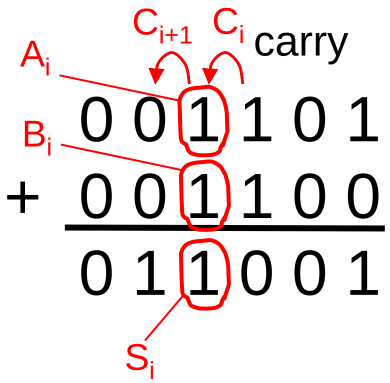
$$y = f(x_1, x_2, \ldots\ x_n)$$



Example: Adder Circuit

Binary addition: Add two n-bit numbers

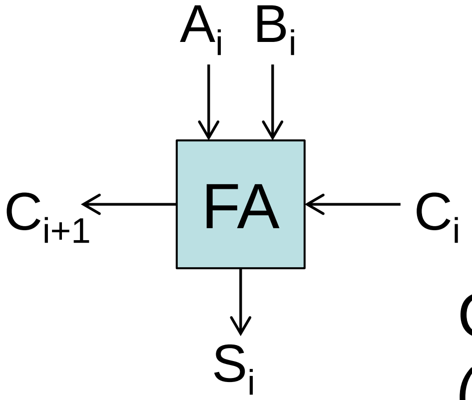
$A_{n-1} \ldots A_1A_0 + B_{n-1} \ldots B_1B_0 \;\text{->}\; S_{n-1} \ldots S_1S_0$



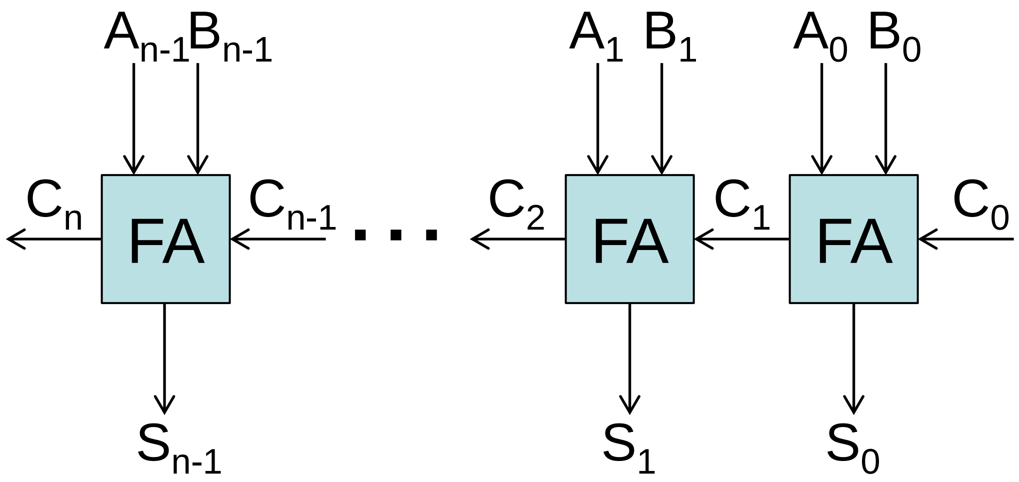
1 bit adder: full adder (FA)

Inputs: A_i, B_i, C_i

Outputs: S_i, C_{i+1}



Ripple Carry Adder



Assuming 2’s complement:

What is C_0?

What do we do with C_n?

Our strategy for addition is to develop a Boolean function (and then a circuit) whose output is equivalent to addition

Example: Sum Output

Consider addition with inputs A_i and B_i along with a carry bit C_i.
Determine the corresponding sum bit S_i and the next carry bit C_{i+1}.

A_i	B_i	C_i	S_i	C_{i+1}

Example: Sum Output

Consider addition with inputs A_i and B_i along with a carry bit C_i.
Determine the corresponding sum bit S_i and the next carry bit C_{i+1}.

A_i	B_i	C_i	S_i	C_{i+1}
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

Summary

- Start with wires, power source
- Semiconductors enable us to create transistors (voltage controlled switch)
- Switches enable us to create basic logic gates (NOT, NAND, NOR)
- Boolean Algebra enables us to create arbitrary Boolean functions from logic gates (logical completeness)