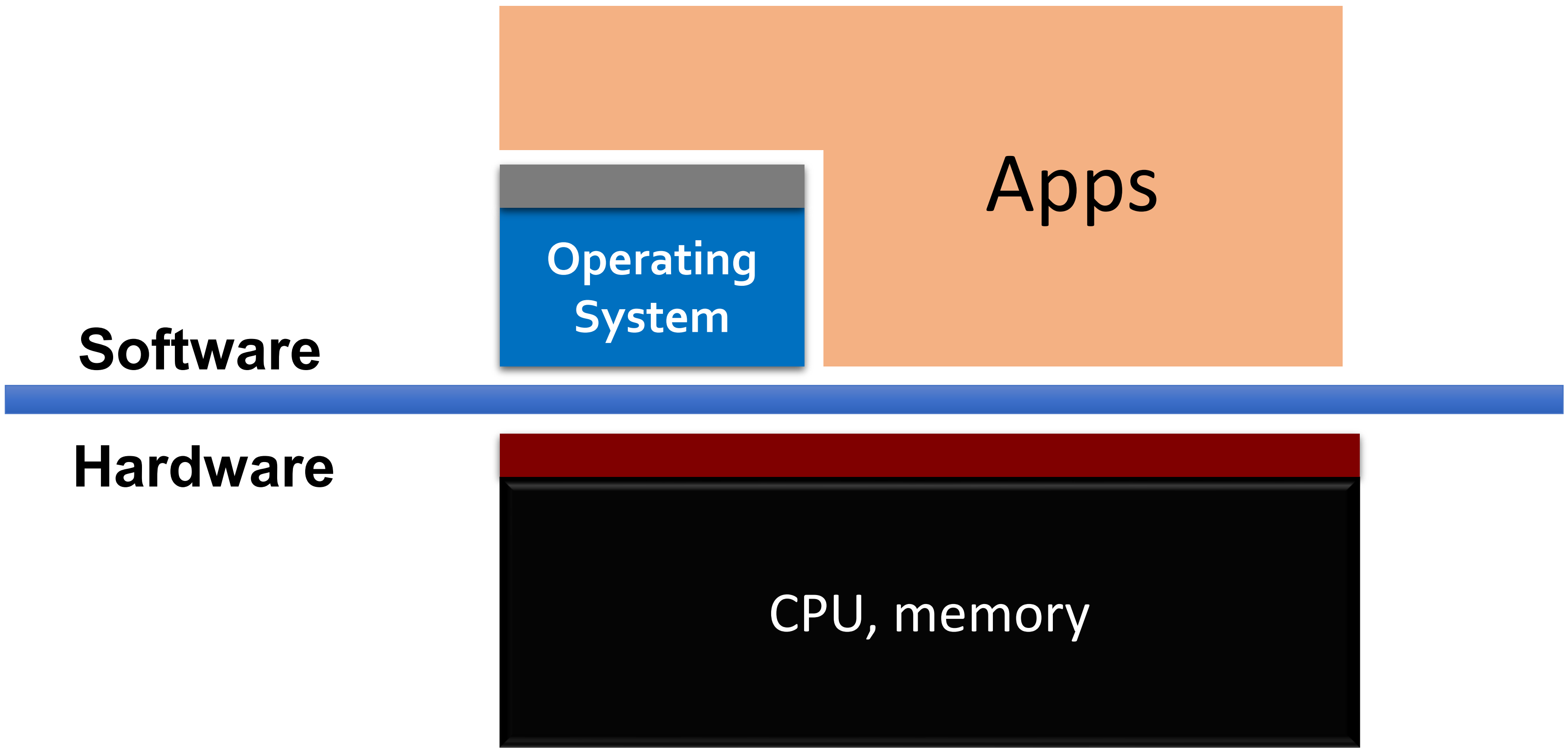
Architecture: Overview and Basic Logic Design

Jinyang Li

What we've learnt so far



What we’ve learnt so far

High level language program
(e.g. C)

```
tmp = *a
*a = *b
*b = tmp
```

Assembly program
(e.g. x86)

```
movq (%rdi), %rax
movq (%rsi), %rbx
movq %rbx, (%rdi)
movq %rax, (%rsi)
```

Machine language program
(e.g. x86)

```
1100110001010100110…
```

Software

Hardware

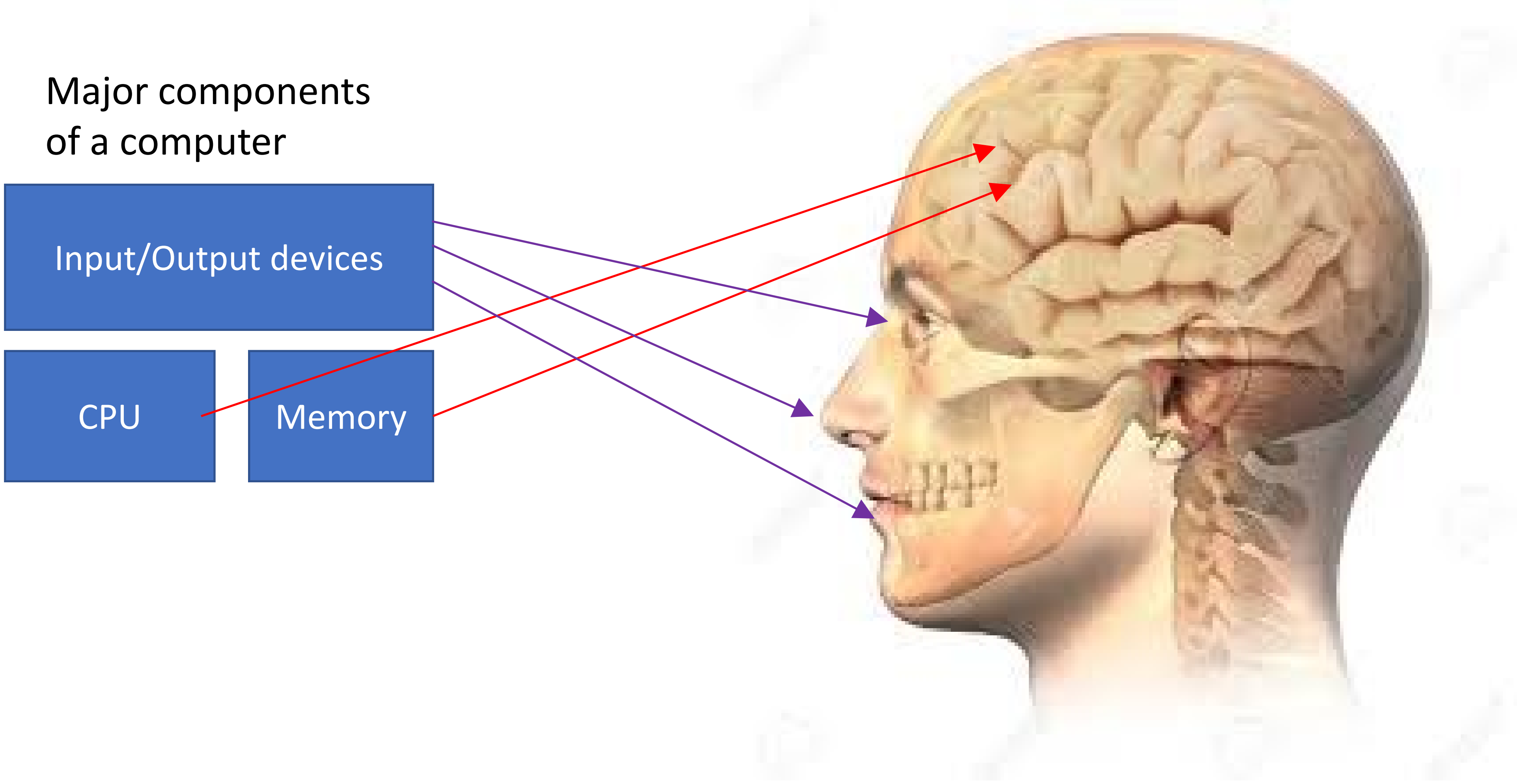
Today’s lecture plan

- Hardware Overview
  - CPU, memory, I/O devices
  - Moore’s law

- Basics of Logic design

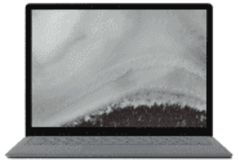
Computer hardware under the cover

Major components
of a computer



Computer hardware under the cover

Major components of a computer



rack-mountable server

Many racks full of rack-mountable servers
(data center)

Computer hardware under the cover

Major components
of a computer

Input/Output devices

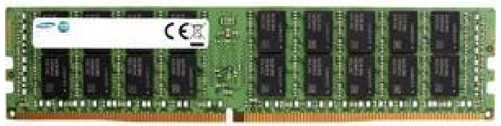
CPU

Memory

Network card

Solid state vs magnetic disk

LCD monitor
(various brands)

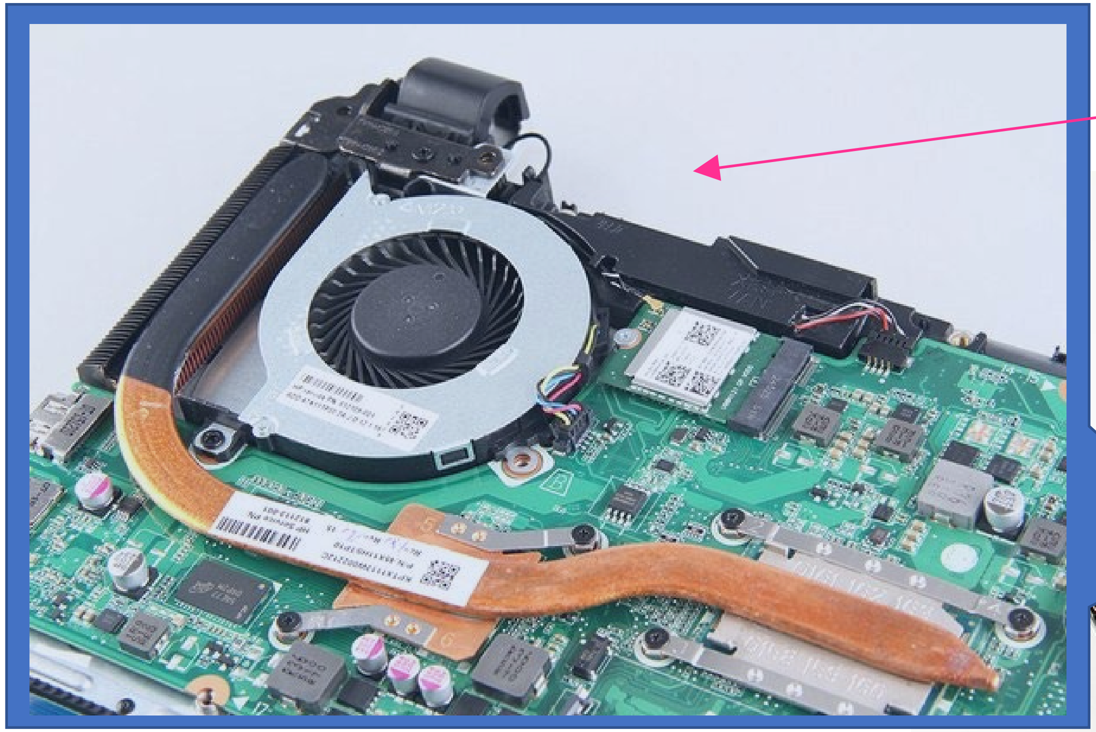


16GB for $100

Teardown of a packaged computer

Teardown of a packaged computer

Teardown of a packaged computer

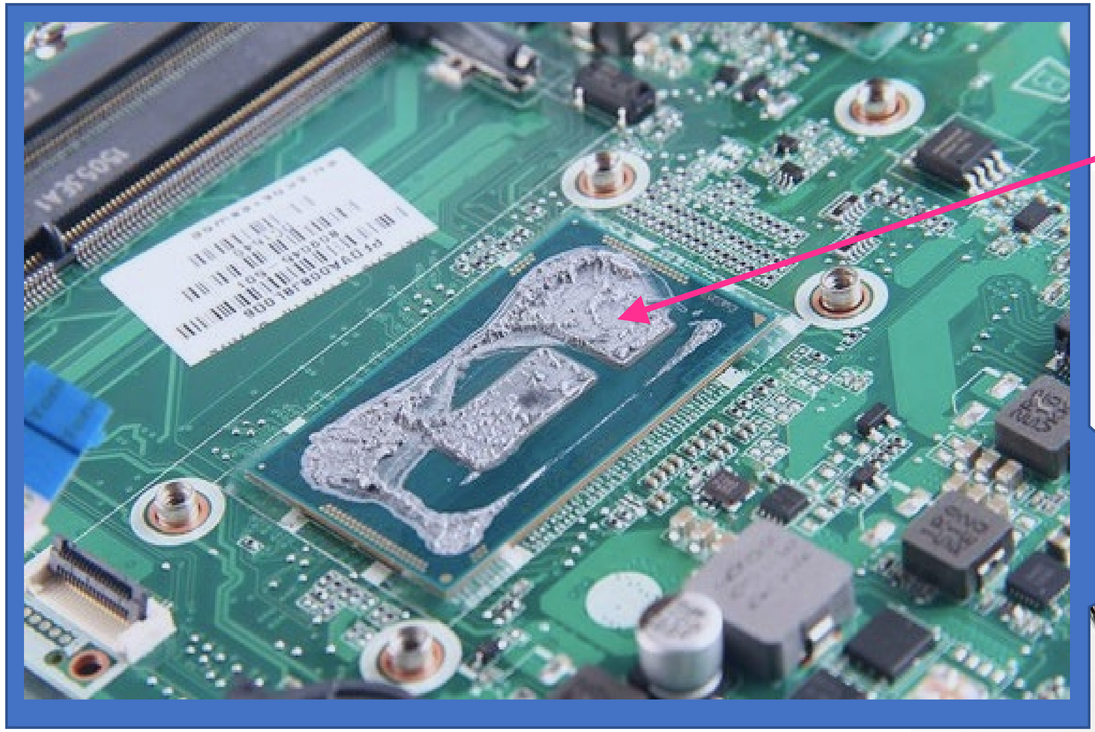


heat sink and cooling fan

If you peel away that fan….

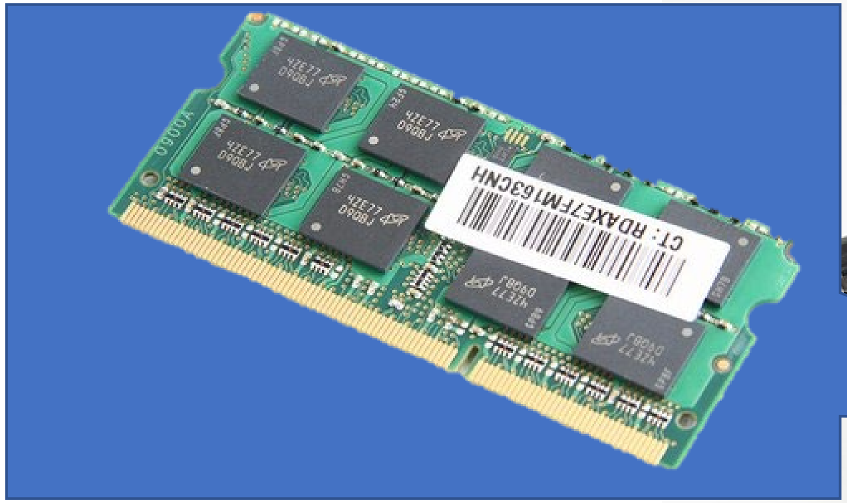
Teardown of a packaged computer

Thermal paste makes the CPU hard to see



Intel® Core™ i7-5500U 3GHz processor

Teardown of a packaged computer

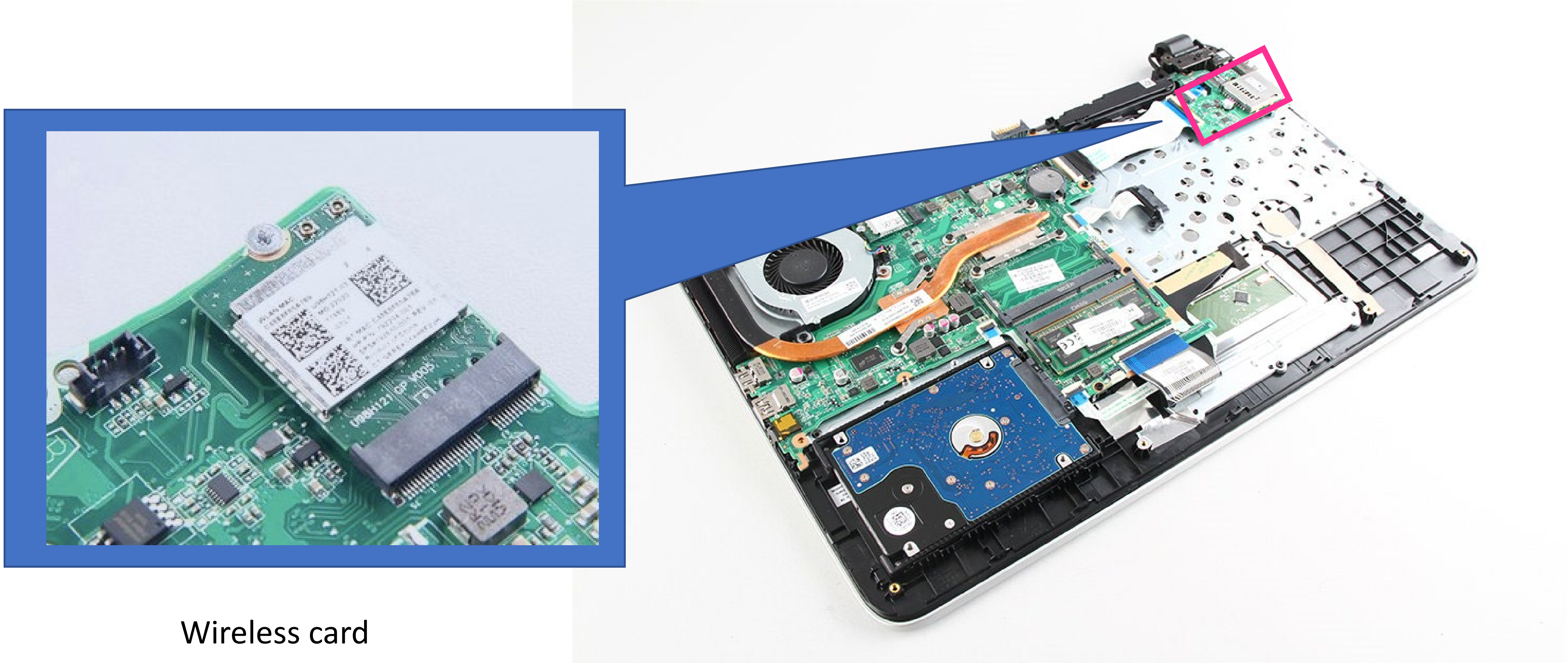


Micron 8GB memory

Teardown of a packaged computer

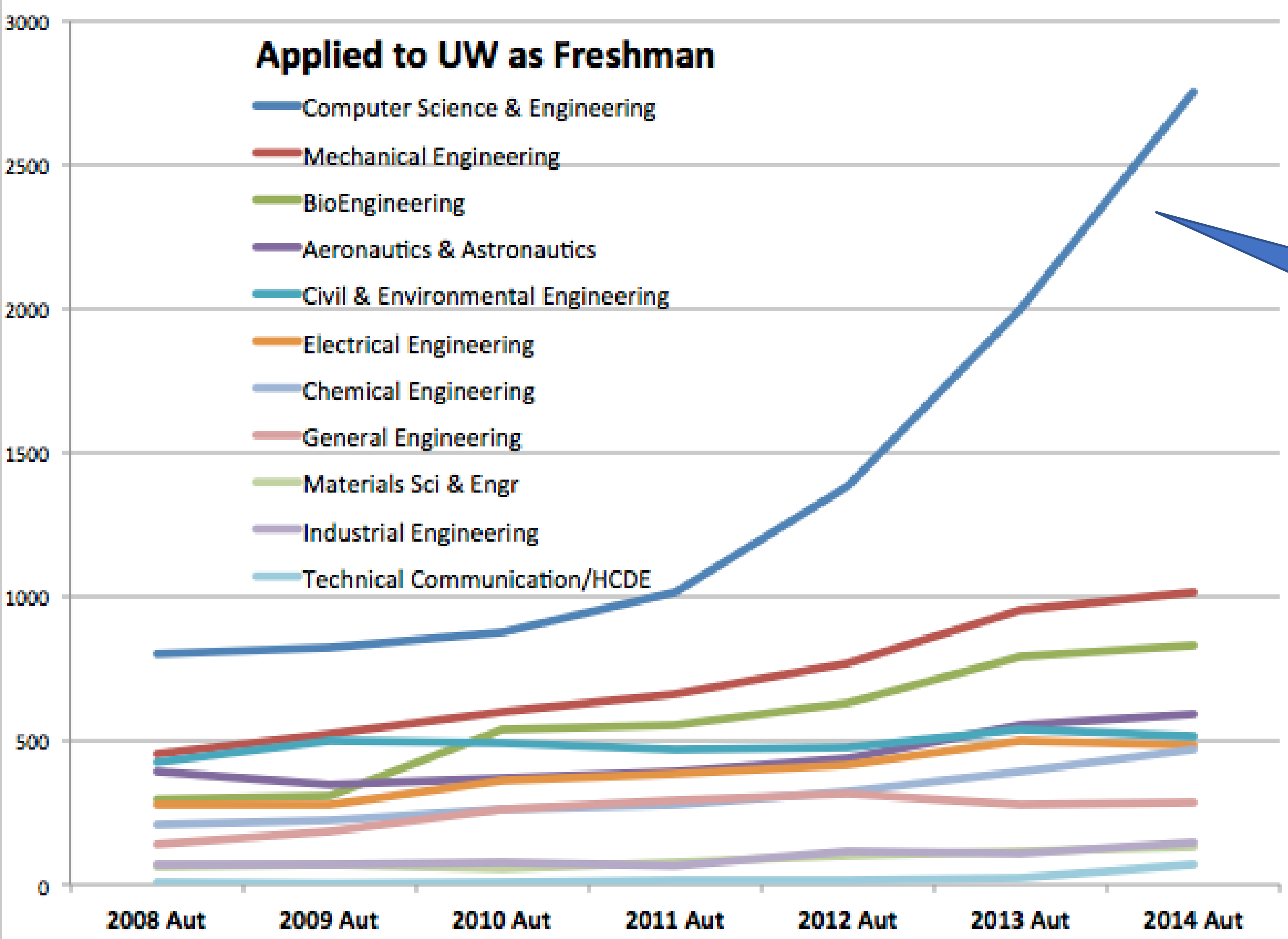
Hard-drive

Teardown of a packaged computer



Wireless card

We are in an exciting field



Why is CS a growth field?

Technological trend: exponential growth

An off/off switch controlled by electricity

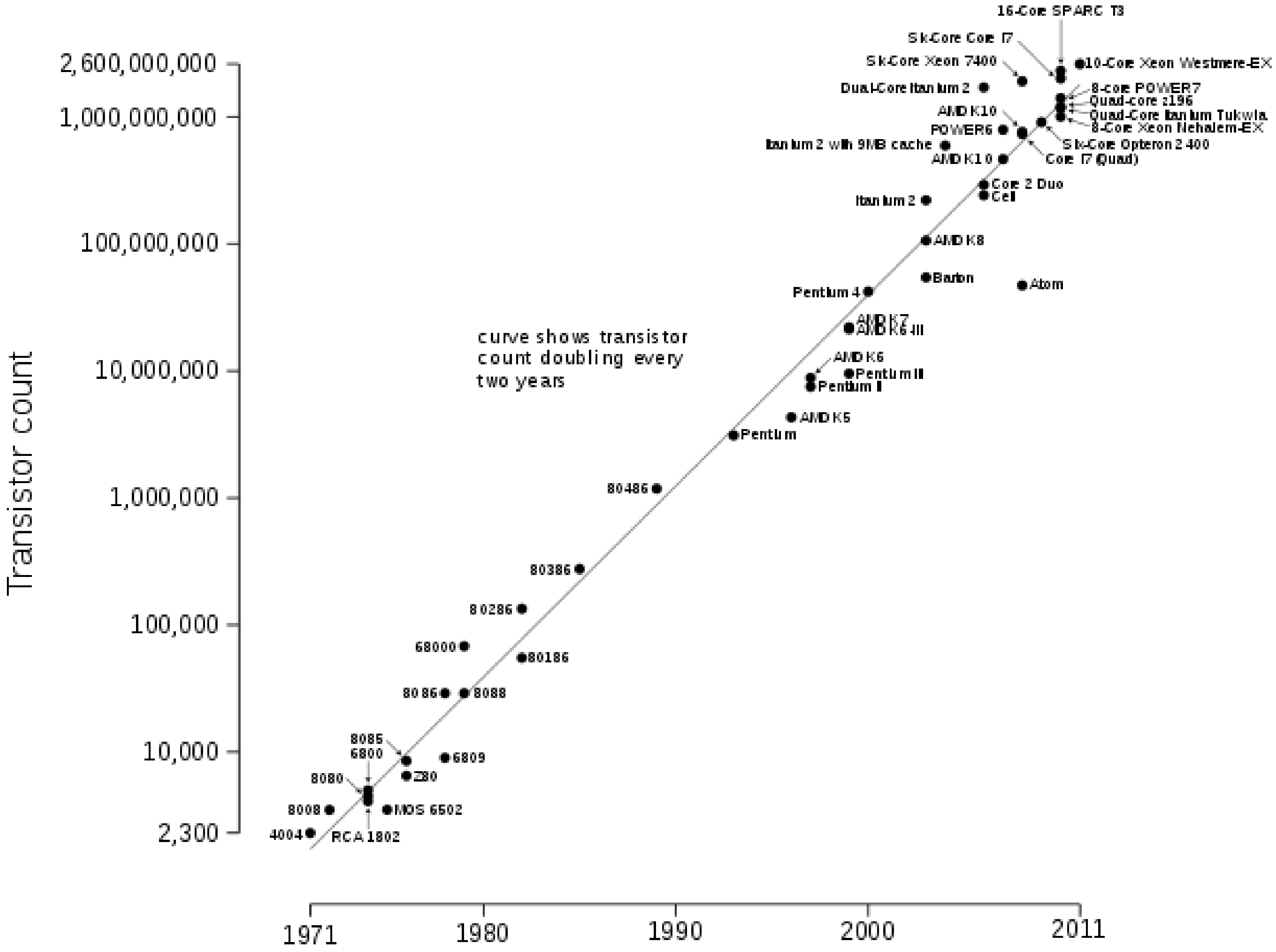
Combine numerous transistors on a single chip

Year	Technology used in computers	Relative performance/unit cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit	900
1995	Very large-scale integrated circuit	2,400,000
2013	Ultra large-scale integrated circuit	250,000,000,000

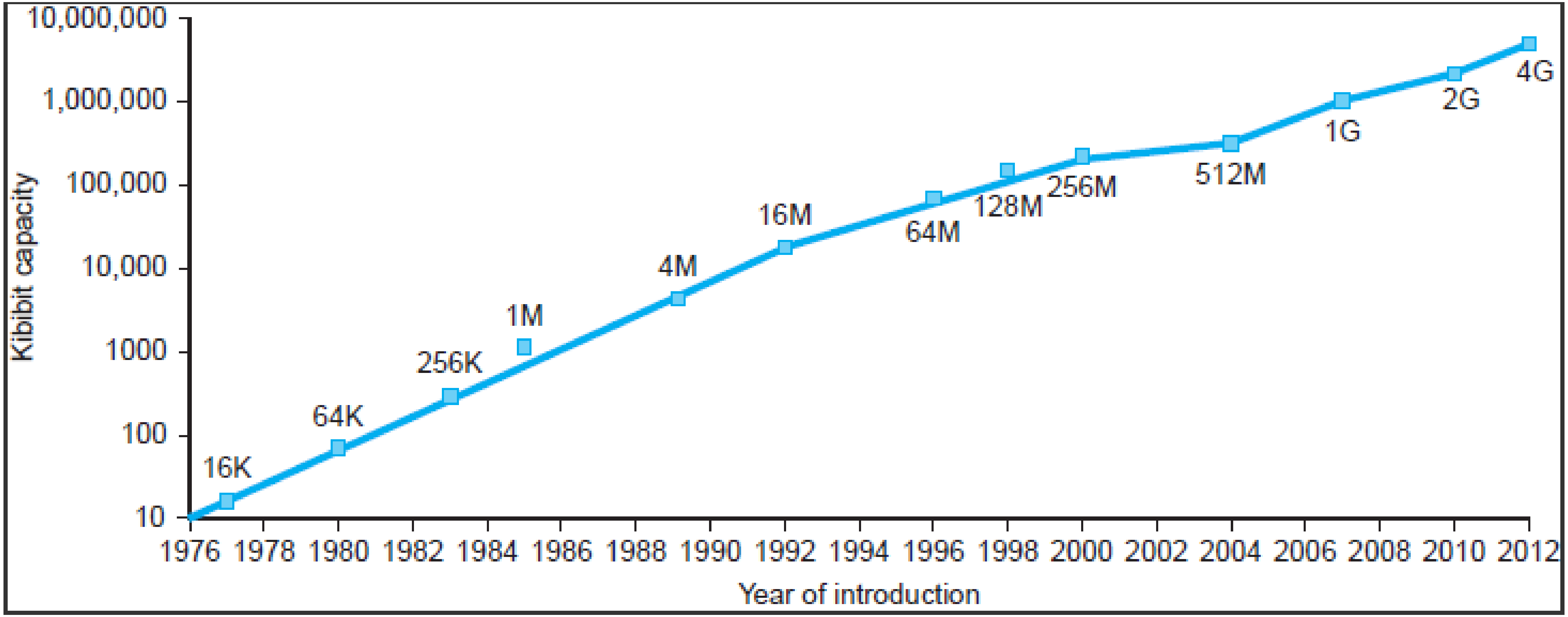
CSO will focus on CPU and memory

Moore's Law in CPU

Moore's Law:
Transistor count doubles every 2 years



Moore's Law in memory



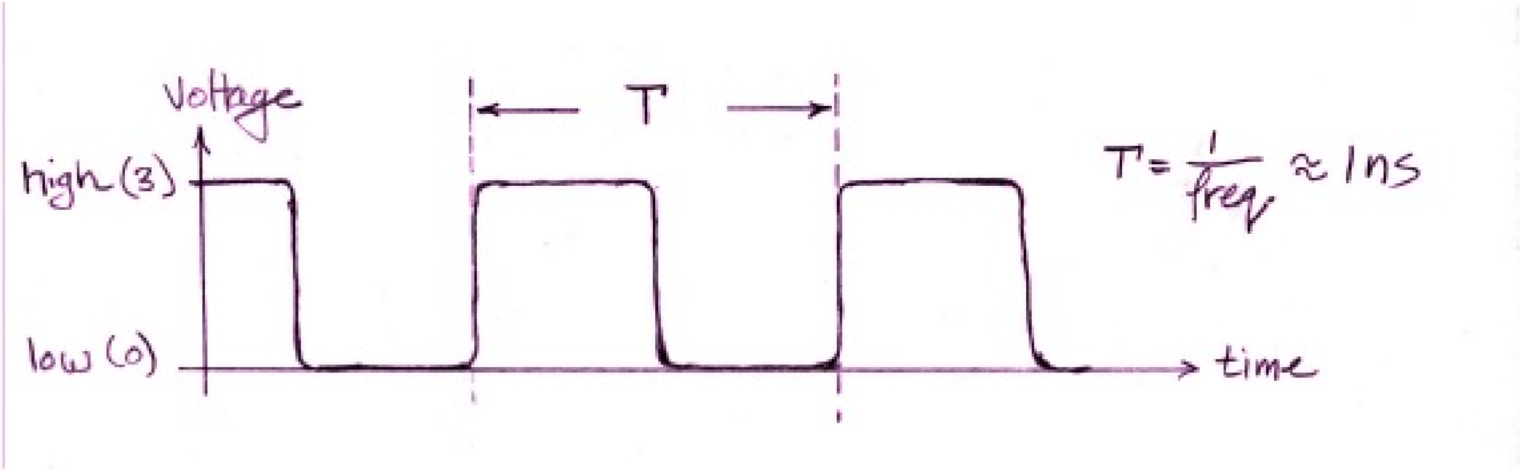
Today’s lecture plan

- Hardware Overview
  - CPU, memory, I/O devices
  - Moore’s law

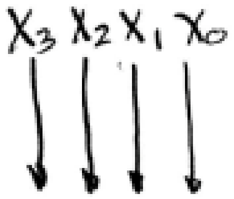
- Basics of Logic design

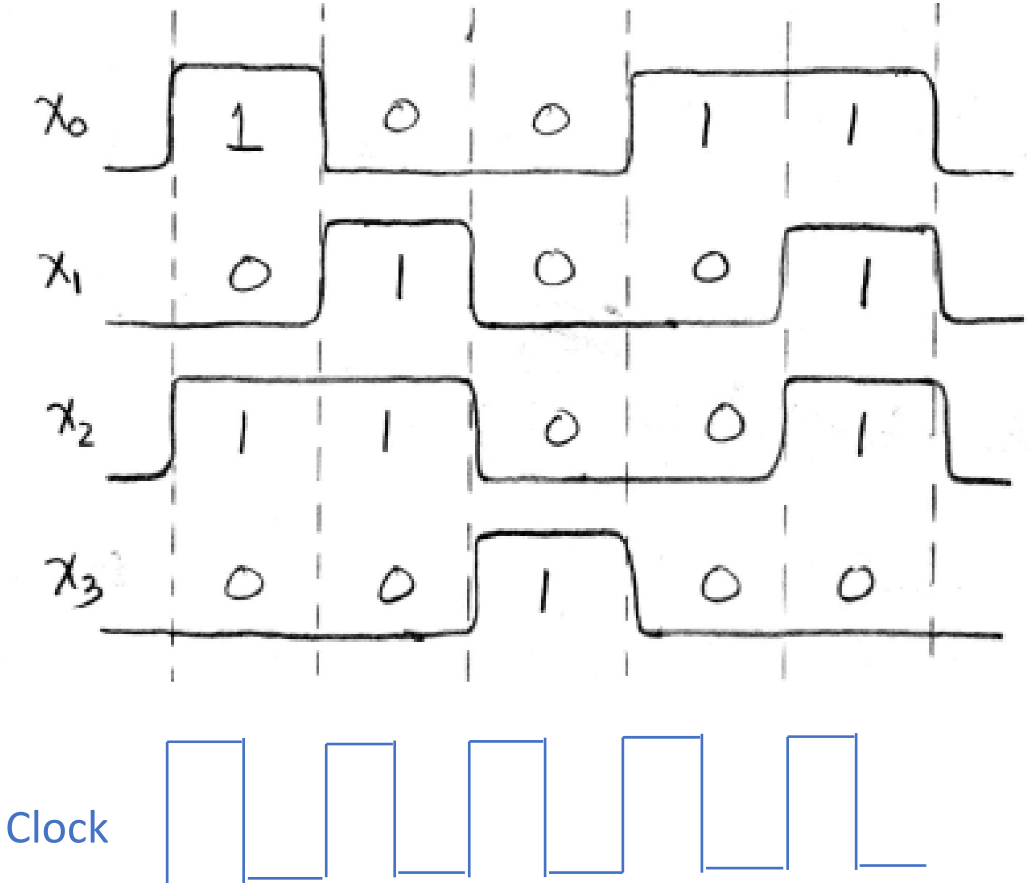
Digital Computer

- Operate with only two voltage levels: high/low
  - that's why computers work with binary numbers
- The clock signal
  - "heartbeat" of the system

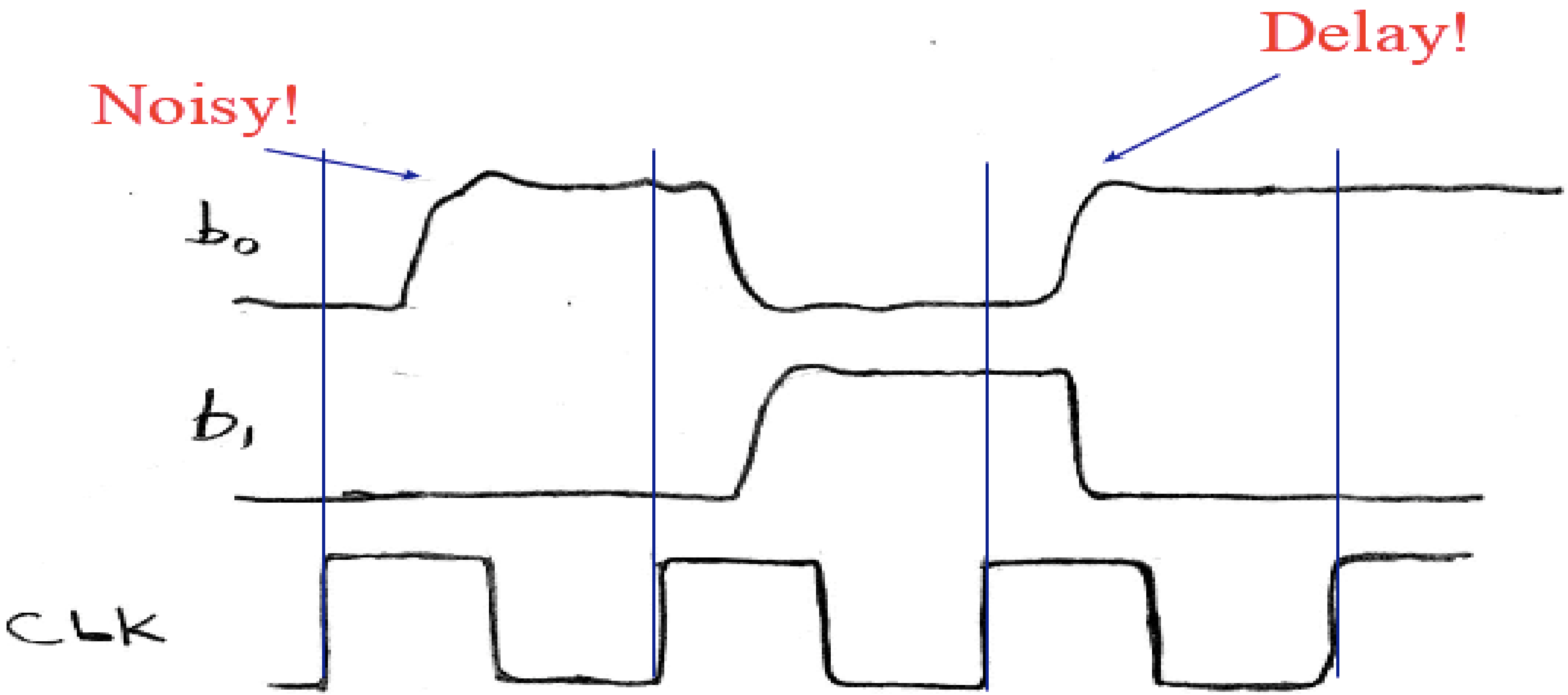


Digital Signals

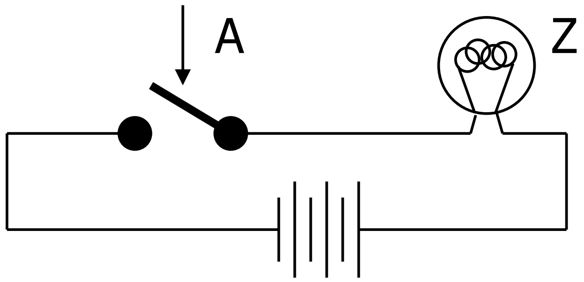




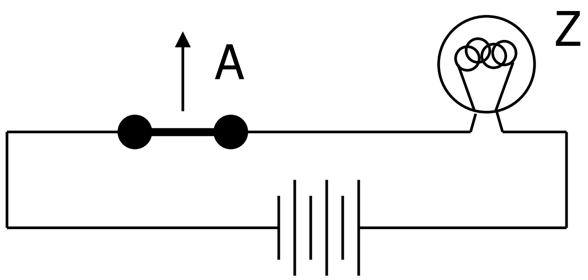
Digital signals



Basic element of implementation: switches



Close switch (if A is “1” or asserted)
→ light bulb is on (Z)

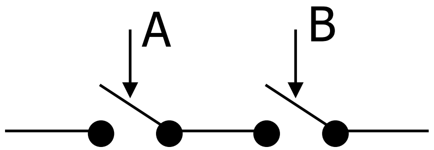


Open switch (if A is “0” or unasserted)
→ light bulb is off (Z)

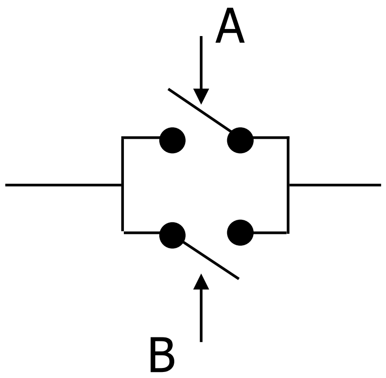
$$\mathbf{Z} \equiv \mathbf{A}$$

Basic element of implementation: switches

- Compose switches into more complex ones (Boolean functions):

AND 

Z ≡ A and B

OR 

Z ≡ A or B

Modern digital systems are done in CMOS transistors

Transistors act as switches

- Modern digital systems are implemented using CMOS transistors
- MOS transistors act as voltage-controlled switches

Bardeen, Shockley and Brattain at Bell Labs, 1948

Historical Note

- Early computer are built from ad hoc circuits
- Common patterns emerge: ANDs, ORs, …
- Claude Shannon made the link to work of 19th century mathematician George Boole
  - Called it "Boolean" in his honor

Shannon

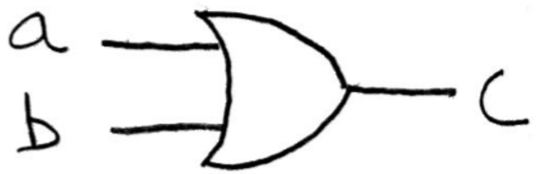
Boole

Gates: building blocks made from small groups of transistors

AND

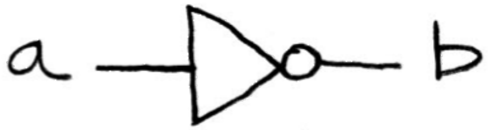
ab	c
00	0
01	0
10	0
11	1

OR



ab	c
00	0
01	1
10	1
11	1

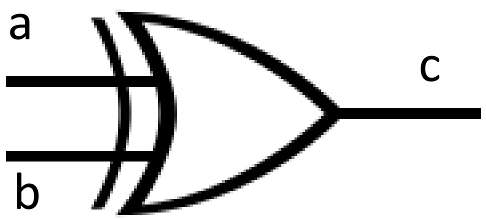
NOT



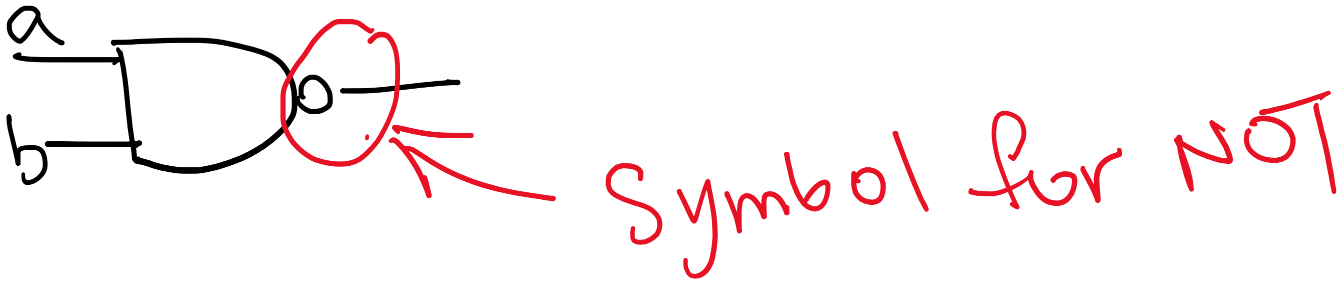
a	b
0	1
1	0

More gates

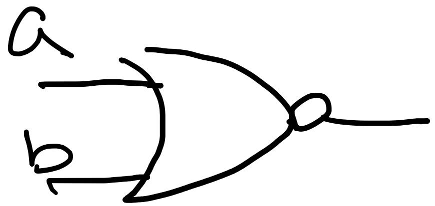
XOR



NAND



NOR



Boolean Algebra

- Express (logic) functions with (logic) equations
  - A•B (AND), A+B (OR), $\bar{A}$ (NOT)
- Laws of Boolean algebra:
  - Basic: $A+0=A$ $\quad A+1=1$ $\quad A\cdot 0=0$ $\quad A\cdot 1=A$
  - Inverse: $A+\bar{A}=1$ $\quad A\cdot\bar{A}=0$ $\quad \overline{A+B}=\bar{A}\cdot\bar{B}$ $\quad \overline{A\cdot B}=\bar{A}+\bar{B}$
  - Commutativity: $A+B=B+A$ $\quad A\cdot B=B\cdot A$
  - Associativity: $A+(B+C)=(A+B)+C$ $\quad A\cdot(B\cdot C)=(A\cdot B)\cdot C$
  - Distribution: $A\cdot(B+C)=A\cdot B+A\cdot C$ $\quad A+(B\cdot C)=(A+B)\cdot(A+C)$

Boolean Algebra

$$A\overline{B} + \overline{A} \stackrel{?}{=} \overline{B} + \overline{A}B$$

Boolean Algebra

$$A\bar{B} + \bar{A} \stackrel{?}{=} \bar{B} + \bar{A}B$$

Method 1: Compare Truth Table

A	B	$A\bar{B}+\bar{A}$	$\bar{B}+\bar{A}B$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Boolean Algebra

$$A\overline{B} + \overline{A} \stackrel{?}{=} \overline{B} + \overline{A}B$$

Method 2: Simplify using basic laws

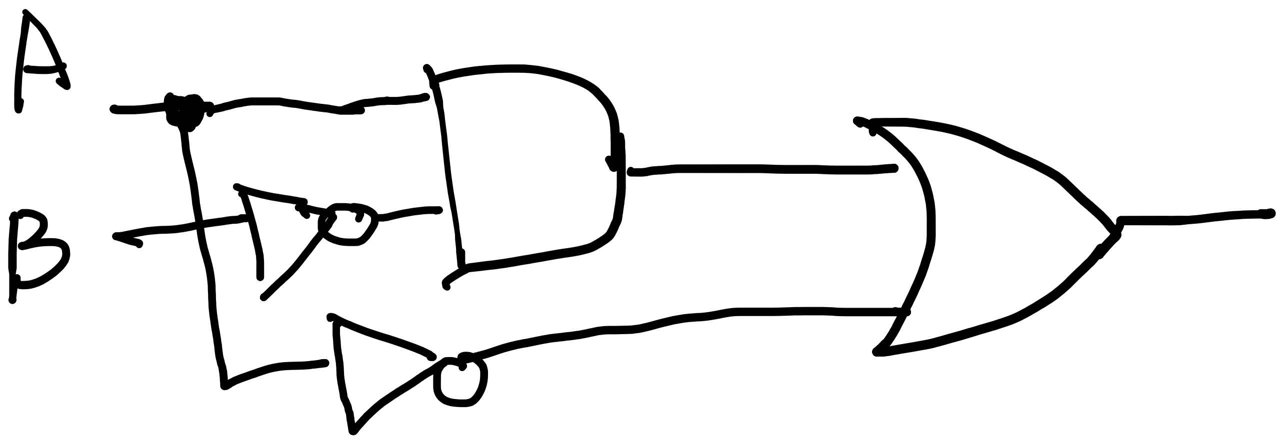
$$A\overline{B} + \overline{A} = A\overline{B} + \overline{A}\cdot 1 = A\overline{B} + \overline{A}\cdot(B + \overline{B})$$

$$= A\overline{B} + \overline{A}B + \overline{A}\overline{B}$$

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

$$= \overline{B} + \overline{A}\cdot B$$

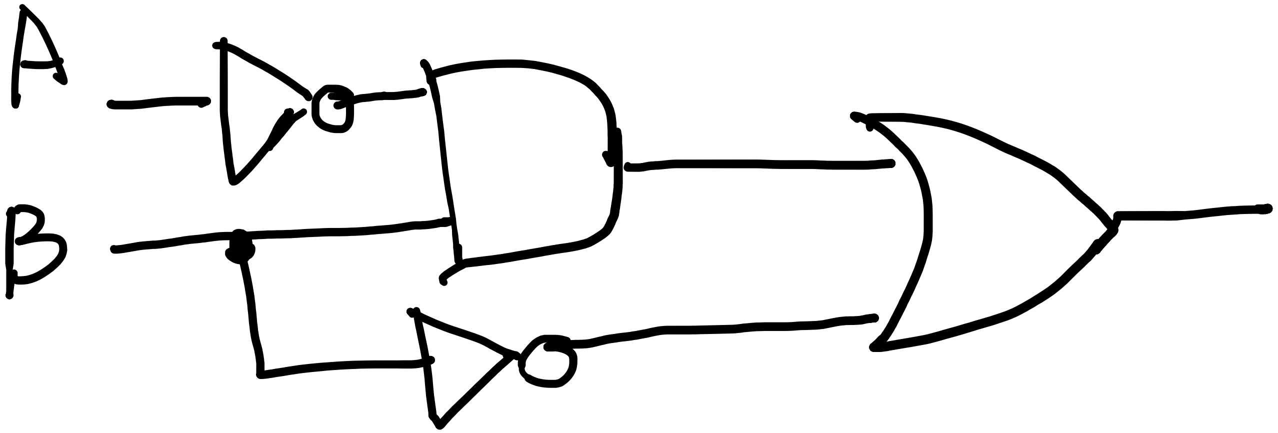
Logic circuits == Boolean algebra



What is the Boolean expression?

$$A\bar{B} + \bar{A}$$

Logic circuits == Boolean algebra



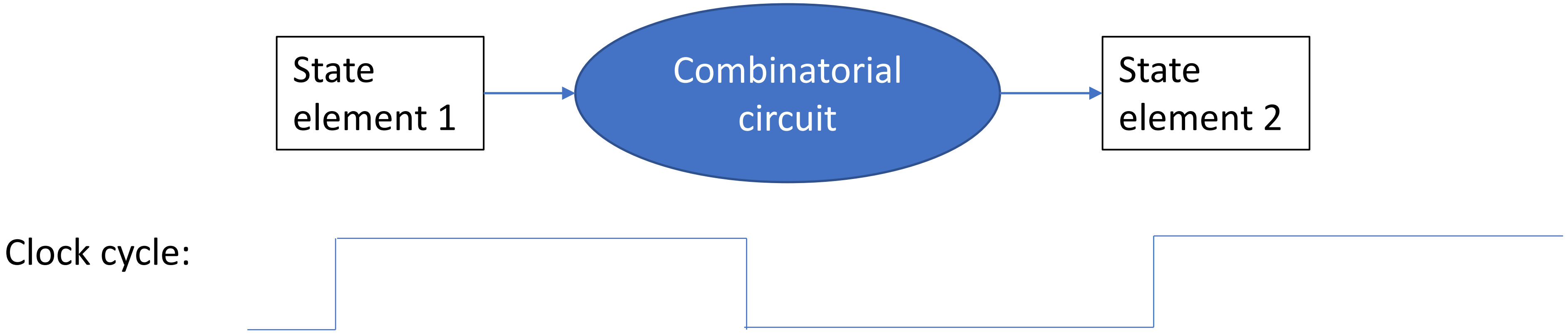
What is the Boolean expression?

$$\bar{A}B + \bar{B}$$

Two types of logic circuit

- Combinatorial circuit
  - output is dependent only on the input

- Sequential circuit
  - output is dependent on both input and state (memory)



How to implement any CL

- Implement a function with AND/OR/NOT

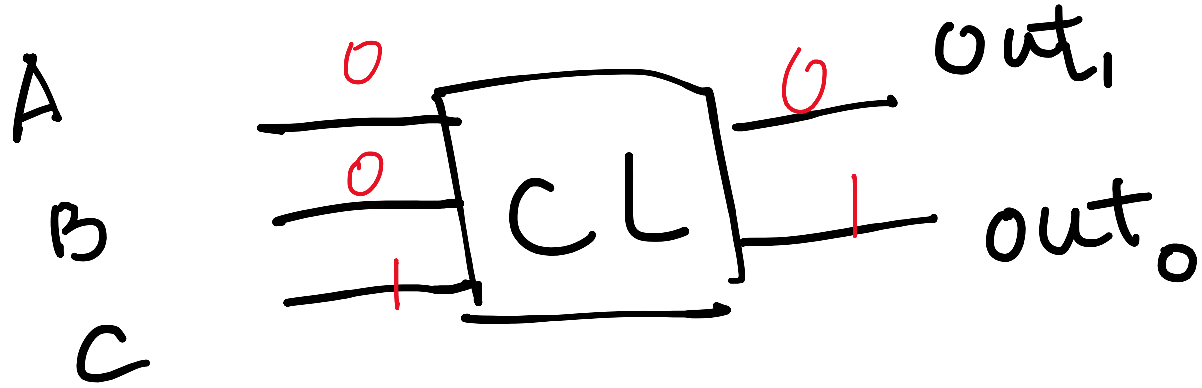
E.g.: Count # of 1's in 3-bit inputs

Step1: obtain truth tables

How to implement any CL

- Implement a function with AND/OR/NOT

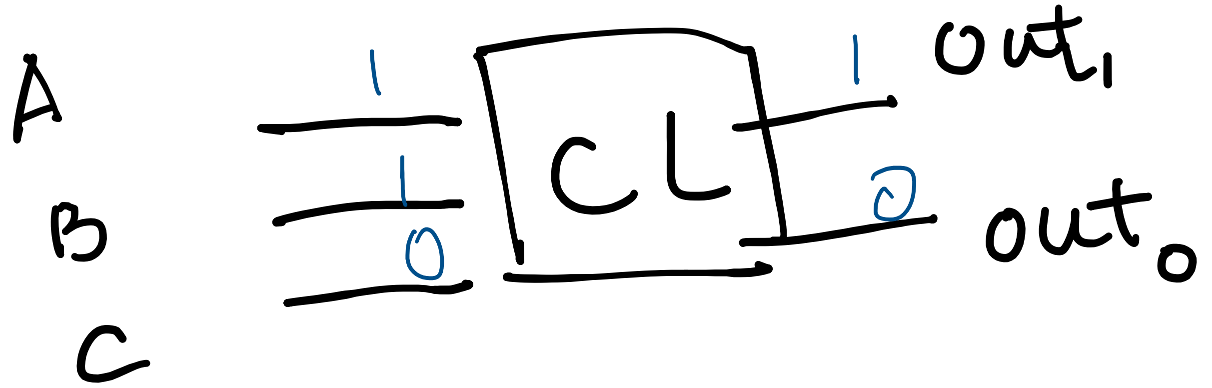
E.g.: Count # of 1's in 3-bit inputs



How to implement any CL

- Implement a function with AND/OR/NOT

E.g.: Count # of 1's in 3-bit inputs



How to implement any CL

- Implement a function with AND/OR/NOT

E.g.: Count # of 1's in 3-bit inputs

Step 1: Write truth table

A	B	C	OUT_1	OUT_0
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

How to implement any CL

- Implement a function with AND/OR/NOT

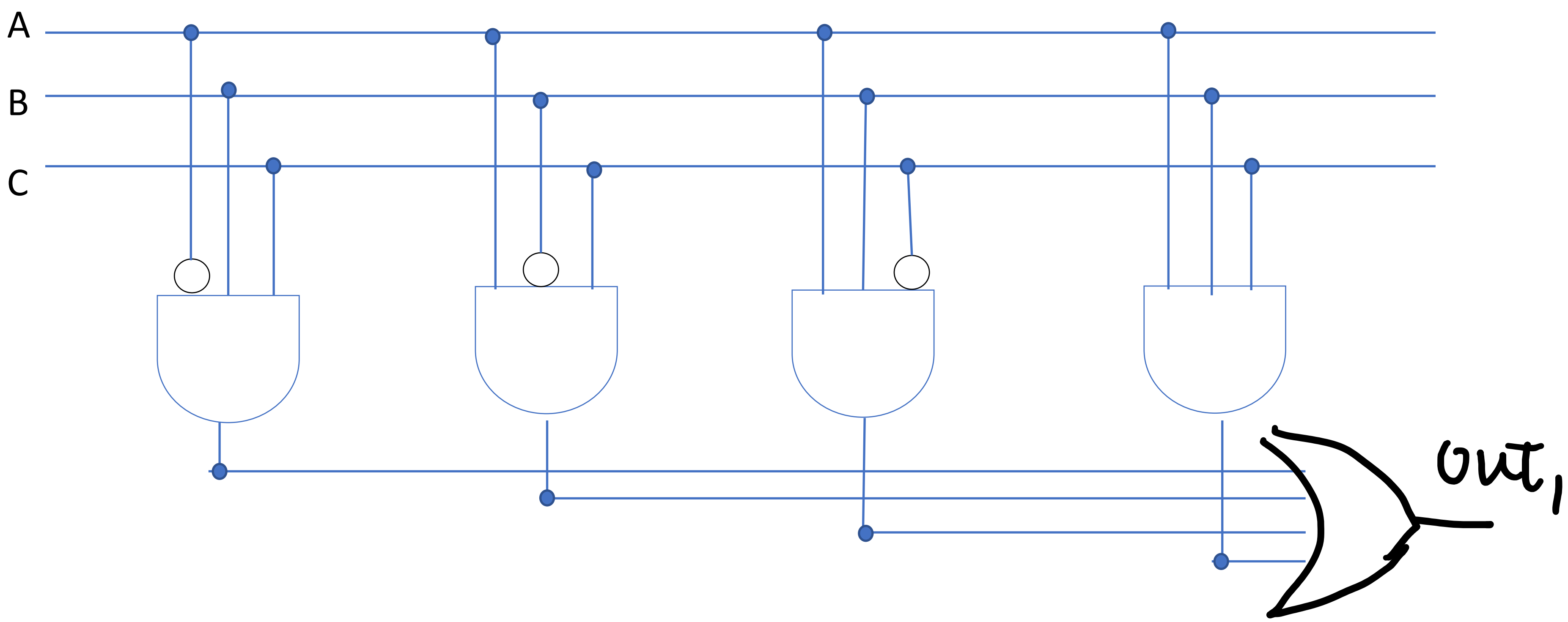
E.g.: Count # of 1's in 3-bit inputs

Step 2: Find sum of product

A	B	C	out_1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$out_1 = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

How to implement any CL



$$out_1 = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

How to implement any CL

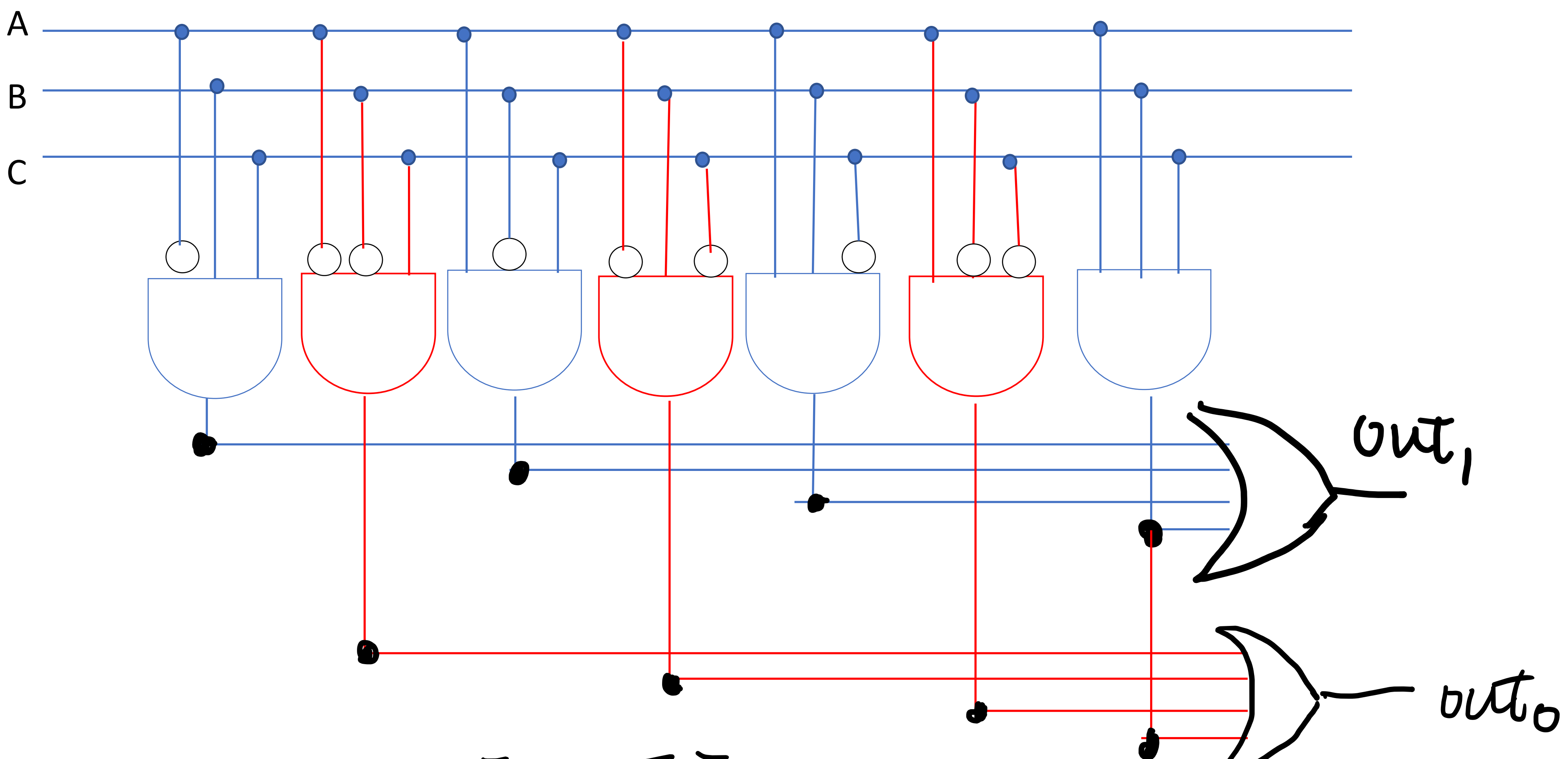
- Implement a function with AND/OR/NOT

E.g.: Count # of 1's in 3-bit inputs

Step 2: Find sum of product

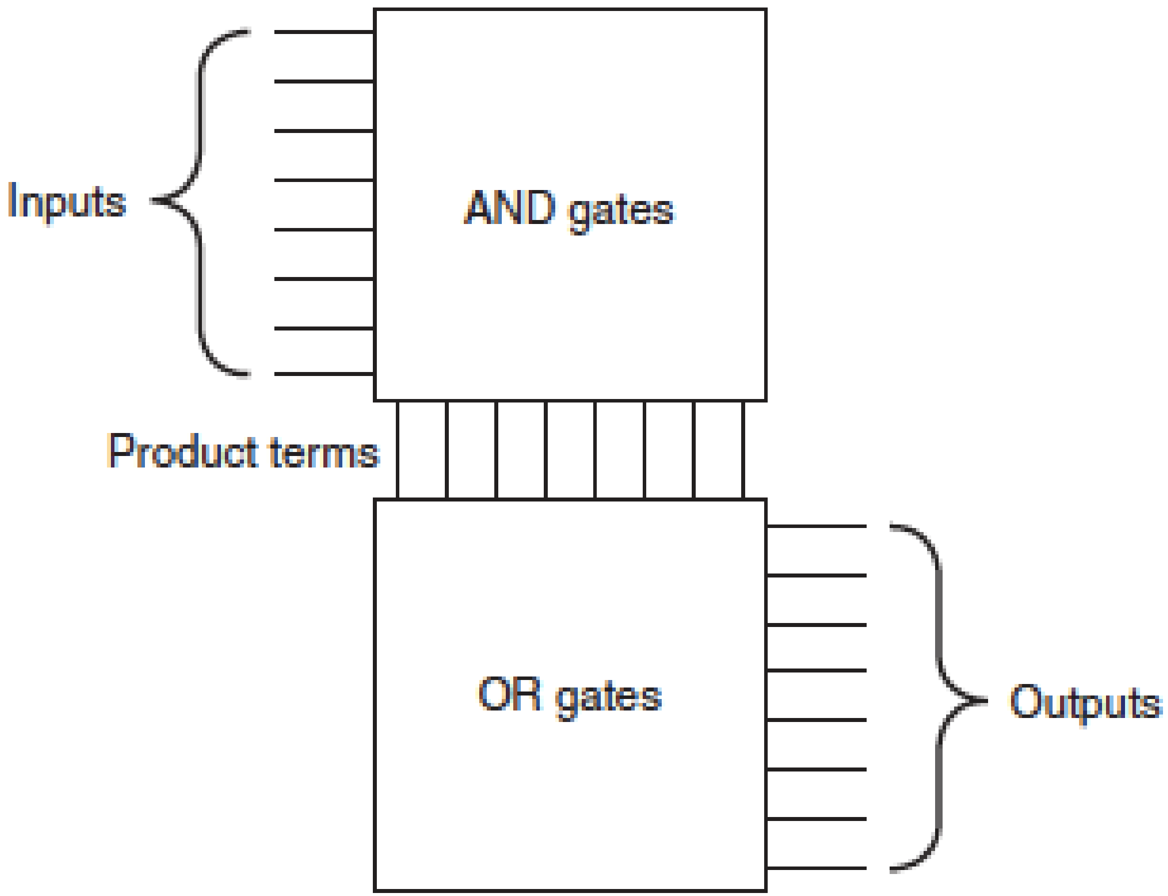
A	B	C	OUT_0
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

How to implement any CL

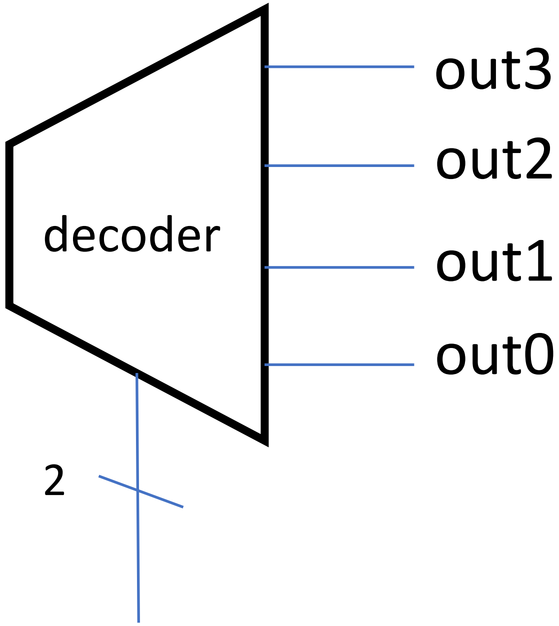


$$out_0 = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

PLA (Programmable Logic Array)

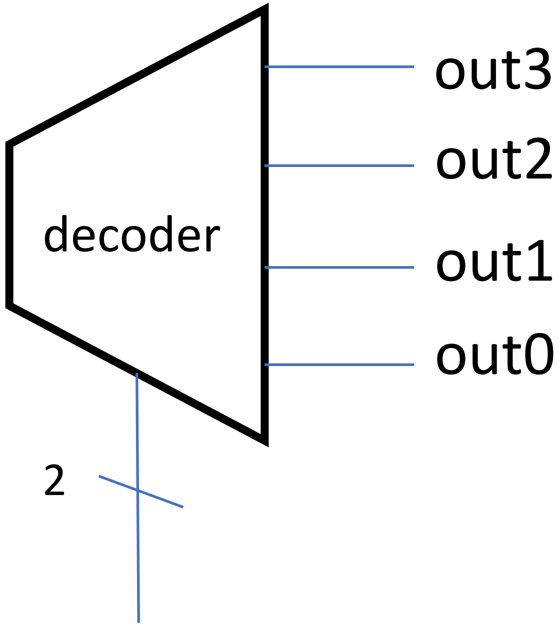


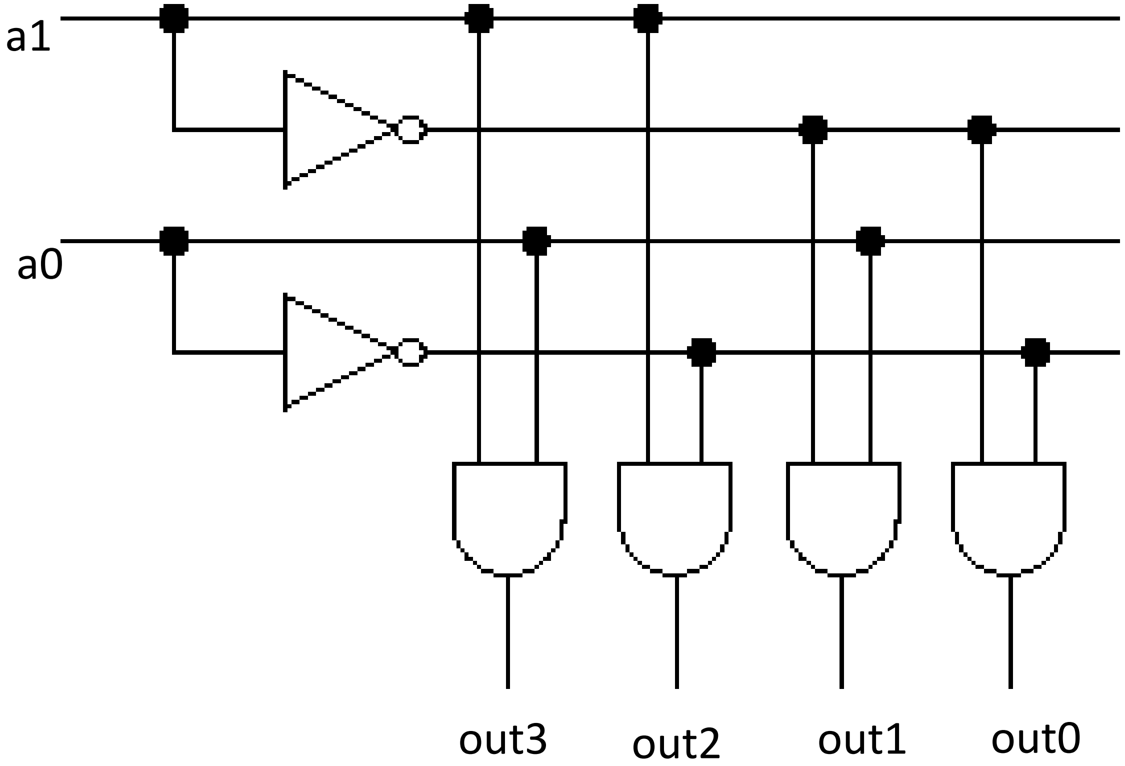
Common CL: decoder



a1	a0	out3	out2	out1	out0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

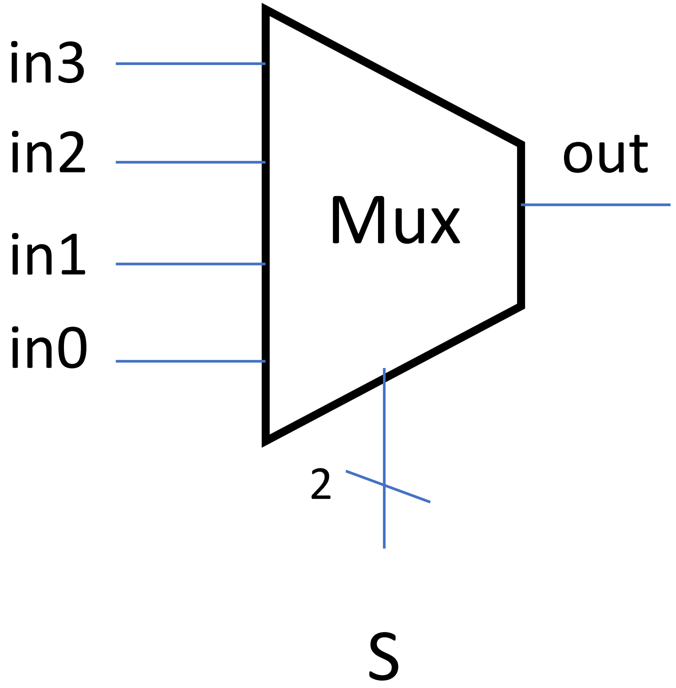
Common CL: decoder





Common CL: Multiplexor

- Select among a set of inputs based on control



s1	s0	in3	in2	in1	in0	out
0	0	x	x	x	0	0
0	0	x	x	x	1	1
0	1	x	x	0	x	0
0	1	x	x	1	x	1
1	0	x	0	x	x	0
1	0	x	1	x	x	1
1	1	0	x	x	x	0
1	1	1	x	x	x	1

Common CL: Multiplexor

- Select among a set of inputs based on control