Logic Gates

and the Mechanics of Computers

Decimal: Base Ten

When we think of numbers, we usually think in **base ten**, which we represent with Arabic numerals:

0 1 2 3 4 5 6 7 8 9

For numbers greater than 9, we prepend more digits. Each place to the left is **ten times** the value of the preceding place to the right

Three hundred twenty one

321

three hundreds | two tens | one one

One hundred eleven thousand, one hundred eleven

111,111

Other Bases

im in ur base, countin ur numberz

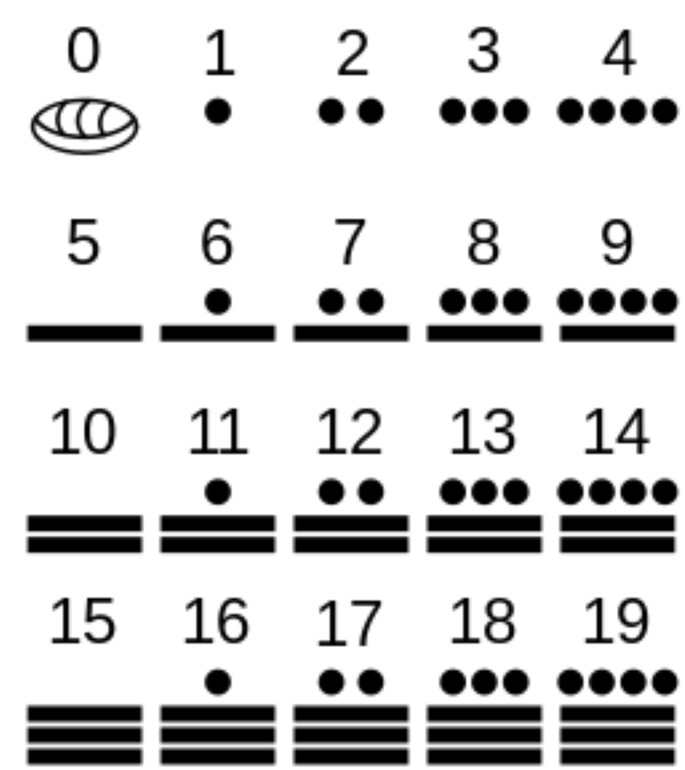
However, there’s no particular reason why we use base ten, any other base could be used just as well
(well, we do have ten fingers which are handy for counting with)

Hexadecimal (base 16):
0 1 2 3 4 5 6 7 8 9 A B C D E F

Duodecimal (base 12):
0 1 2 3 4 5 6 7 8 9 X E

Octal (base 8):
0 1 2 3 4 5 6 7

Vigesimal (base 20)
Mayan number system



Binary: Base Two

However, the way computers work on the most fundamental level is in binary - the smallest base possible, with two digits:

0 1

Bits - **Bi**nary Digi**ts**

Bits are particularly useful because you don’t need symbols (like 0 and 1) to represent them. All you need is something that can exist in two possible states

1	**0**
on	off
presence	absence
true	false
up	down

Binary: Base Two

Each place to the left is **two times** the value of the preceding place to the right. For example:

10101010

1	**0**	**1**	**0**	**1**	**0**	**1**	**0**
one twenty eight	sixty-four	thirty-two	sixteen	eight	four	two	one

128 + 32 + 8 + 2 = 170

10101010 (bin) = 170 (dec)

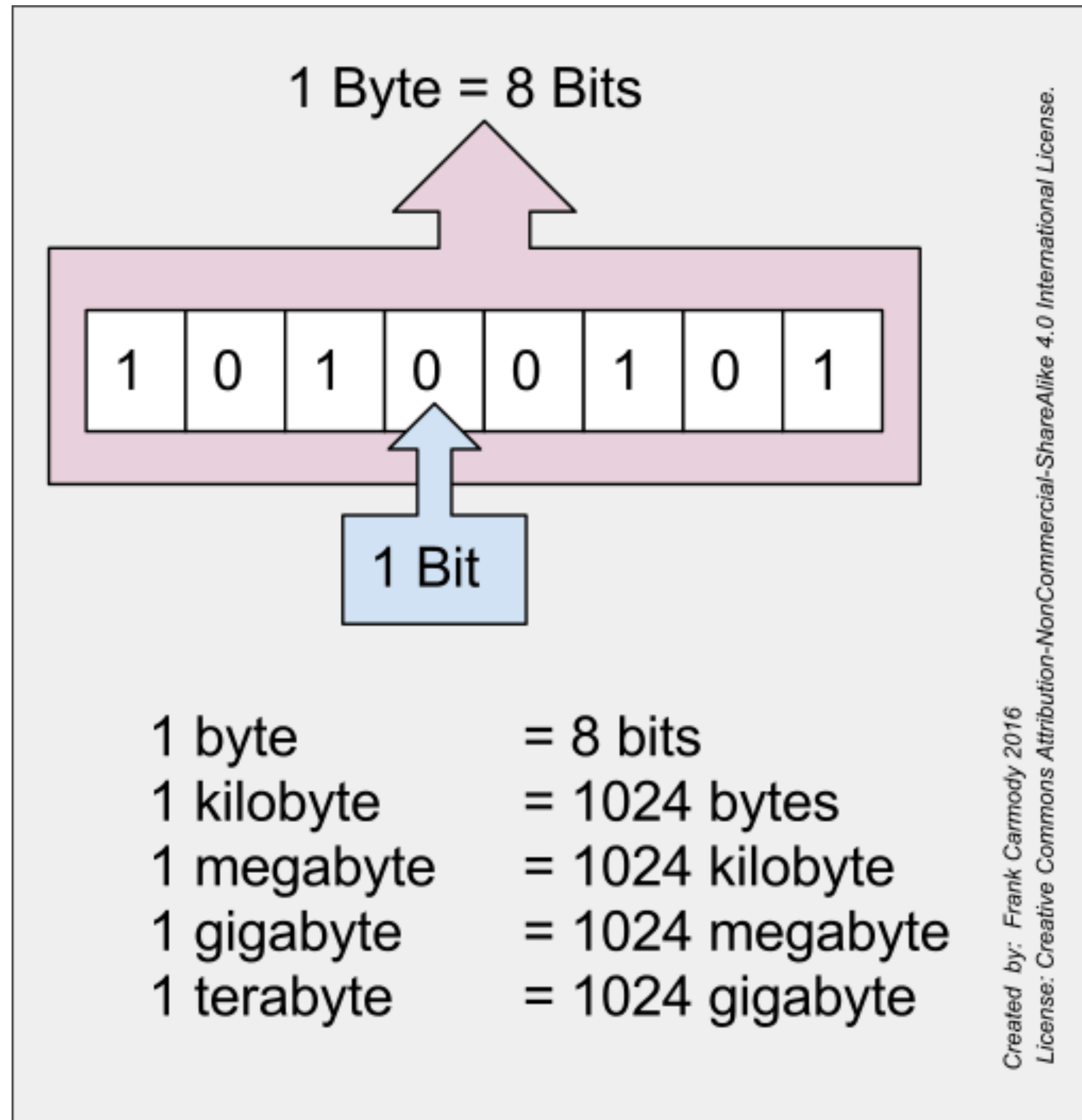
Decimal vs **Binary**

Largest digit: 9 Each place to the left is **ten times** 1he preceding place's value	Largest digit (bit): 1 Each place to the left is **two times** the preceding place's value
One hundred eleven thousand, one hundred eleven **111,111**	*thirty one* **111111**
10^3 1000(dec) 1111101000(bin)	**2^3** 1000(bin) 8(dec)

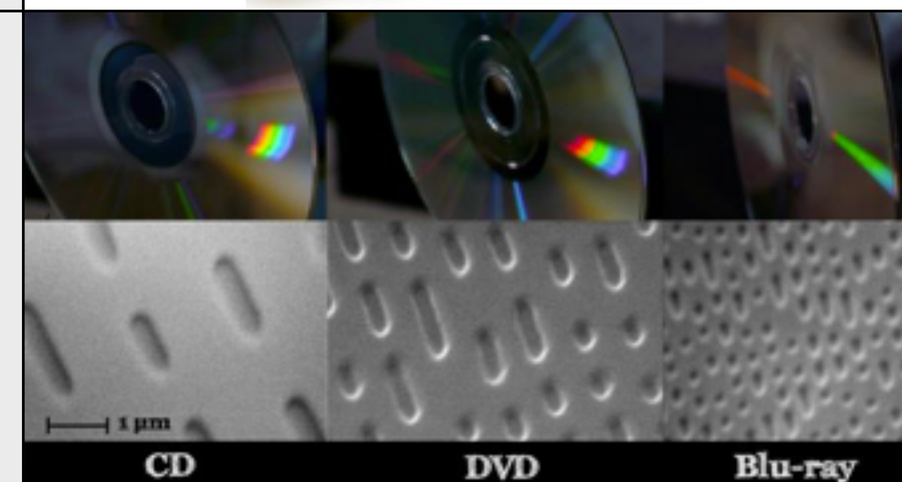
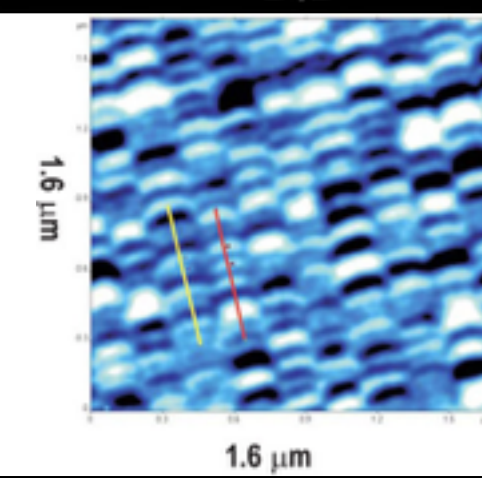
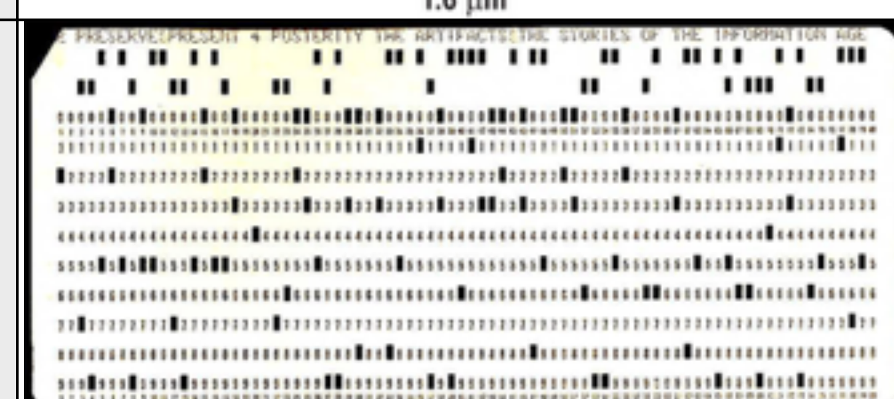
Binary Code

- Commonly, 8 bits to 1 byte
- Smallest addressable unit of memory in most computer architectures
- Value range: 0 - 255
- Each value can correspond to a character, symbol, or instruction
- ‘Wikipedia’ in ASCII binary code:

01010111 01101001 01101011
01101001 01110000 01100101
01100100 01101001 01100001



Bits in Media

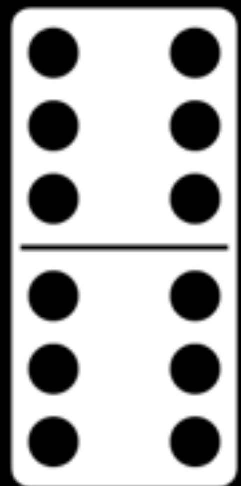
Media	Used in	Photo
Electricity	processors, RAM, flash memory, cable/DSL/Ethernet	
Light	fiber-optic cable, CD/DVD/Blu-Ray, <future> photonic processors </future>	
Magnetism	hard disk drives	
Paper	punched card	

but you don’t *need* to have electronics to store and process information

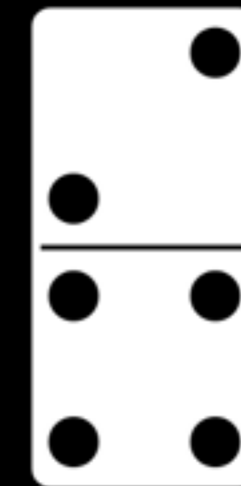
-so-

how about bits in…


DOMINOS



Logic Gates



Logic gates are the fundamental mechanisms by which computers process information

In place of trying to follow individual electrons through circuits, we can do the same with domino set ups to better grasp how logic gates work

First, lets go through the different logical operators and their *truth tables*, which form the basis of pure logic

AND

Both are true

p	*q*	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

OR

Either are true

p	*q*	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

XOR

Either are true, but not both

p	*q*	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

NAND

At least one is false

p	*q*	$p \uparrow q$
T	T	F
T	F	T
F	T	T
F	F	T

NOR

Both must be false

p	*q*	$p \downarrow q$
T	T	F
T	F	F
F	T	F
F	F	T

XNOR

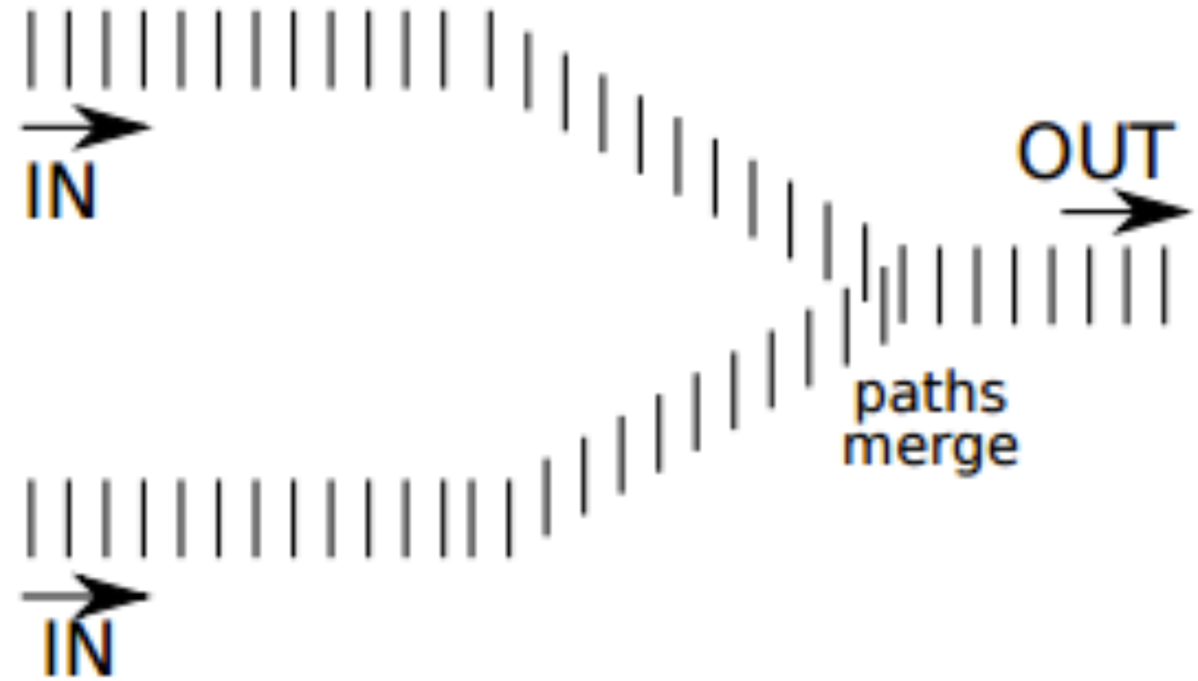
Logical equality

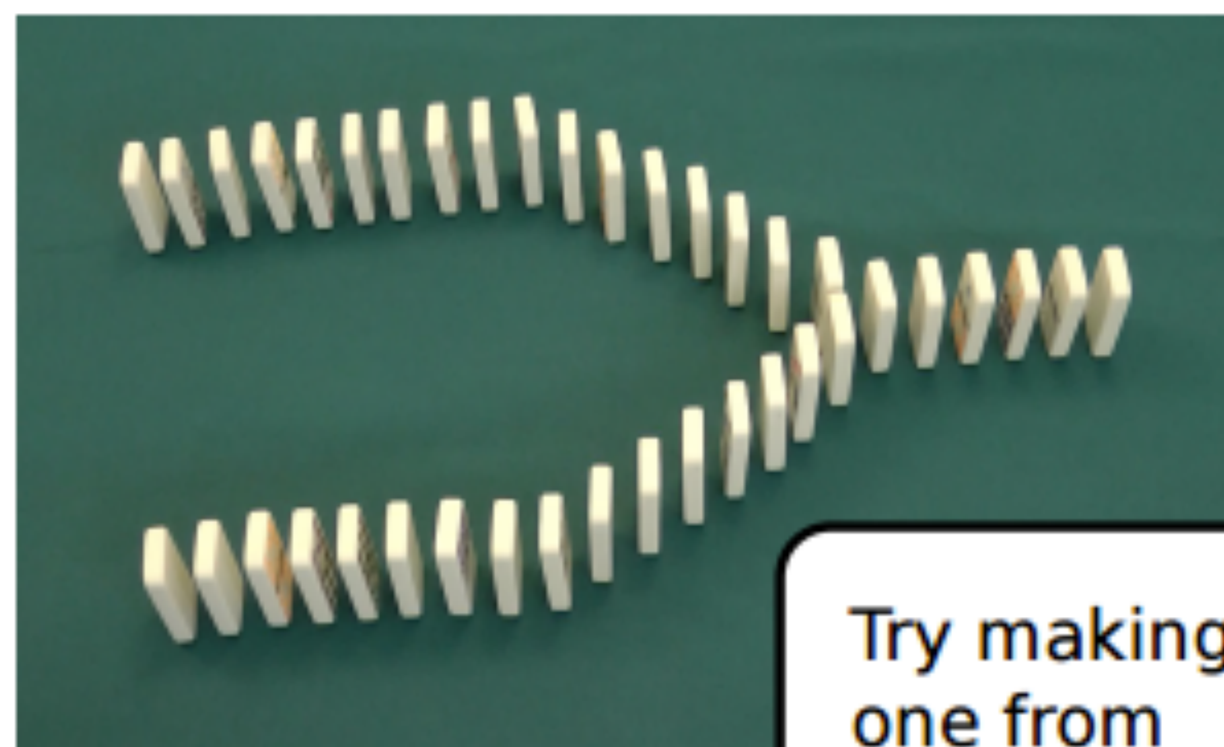
p	*q*	$p = q$
T	T	T
T	F	F
F	T	F
F	F	T

OR

INPUT		OUTPUT
A	B	A OR B
1	1	0
1	0	1
0	1	1
0	0	1

Sends output when it receives one or more inputs



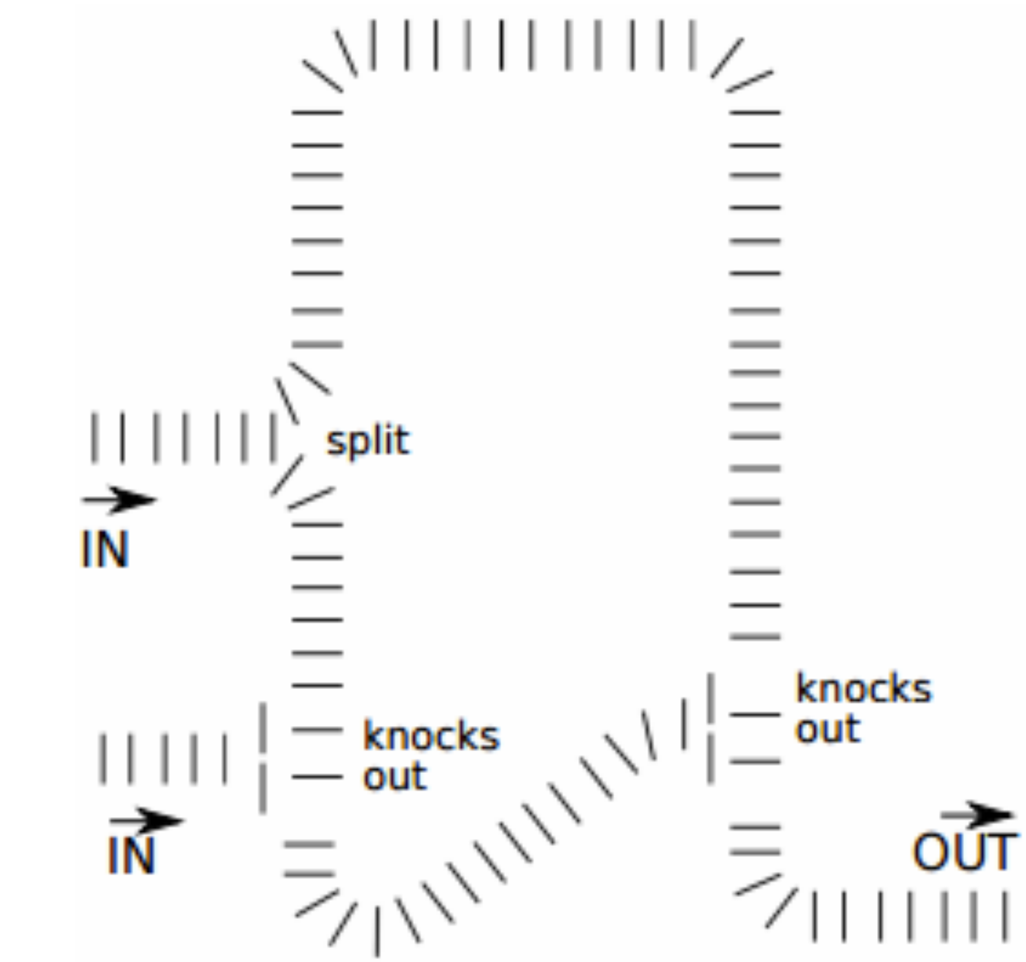


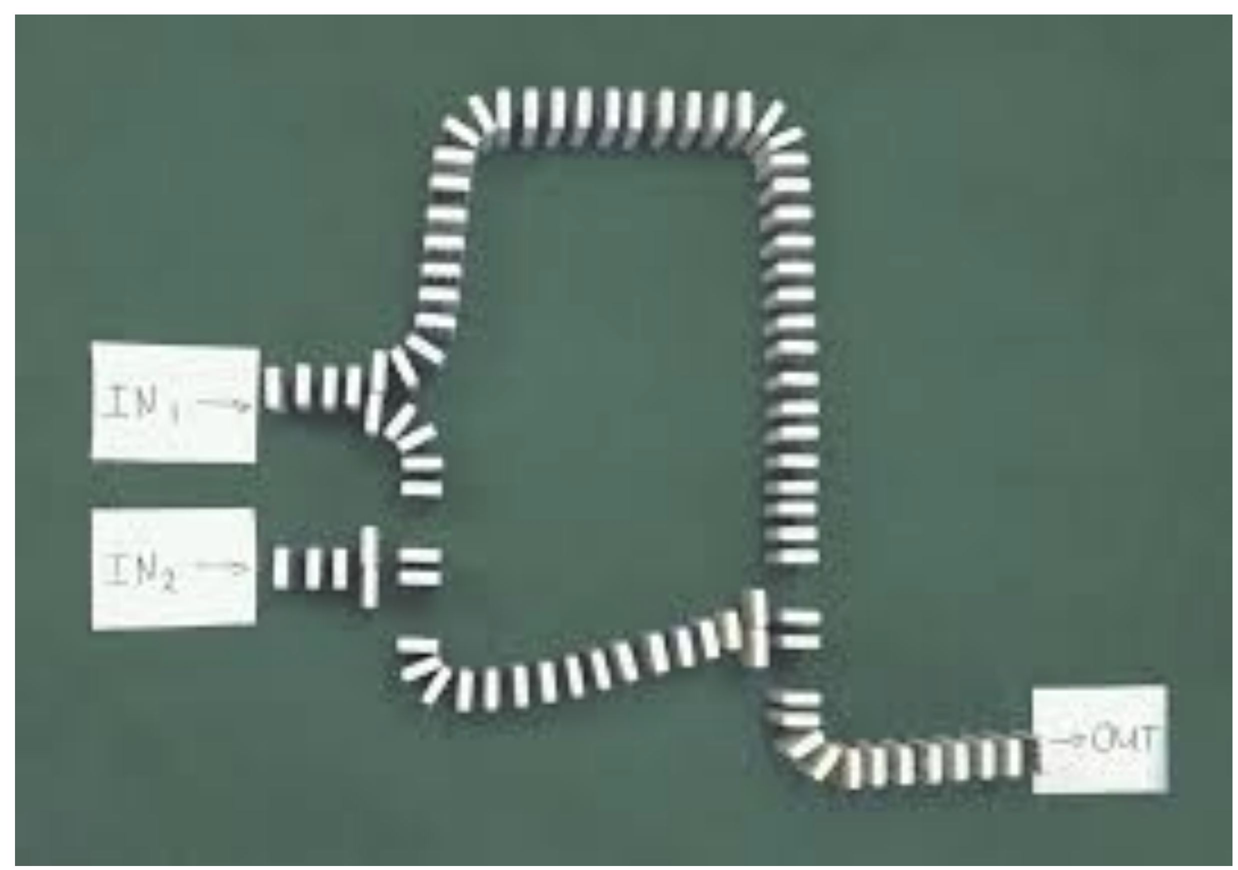
Try making one from dominoes!

AND

INPUT		OUTPUT
A	**B**	**A AND B**
1	1	1
1	0	0
0	1	0
0	0	0

Sends output only when it receives two inputs

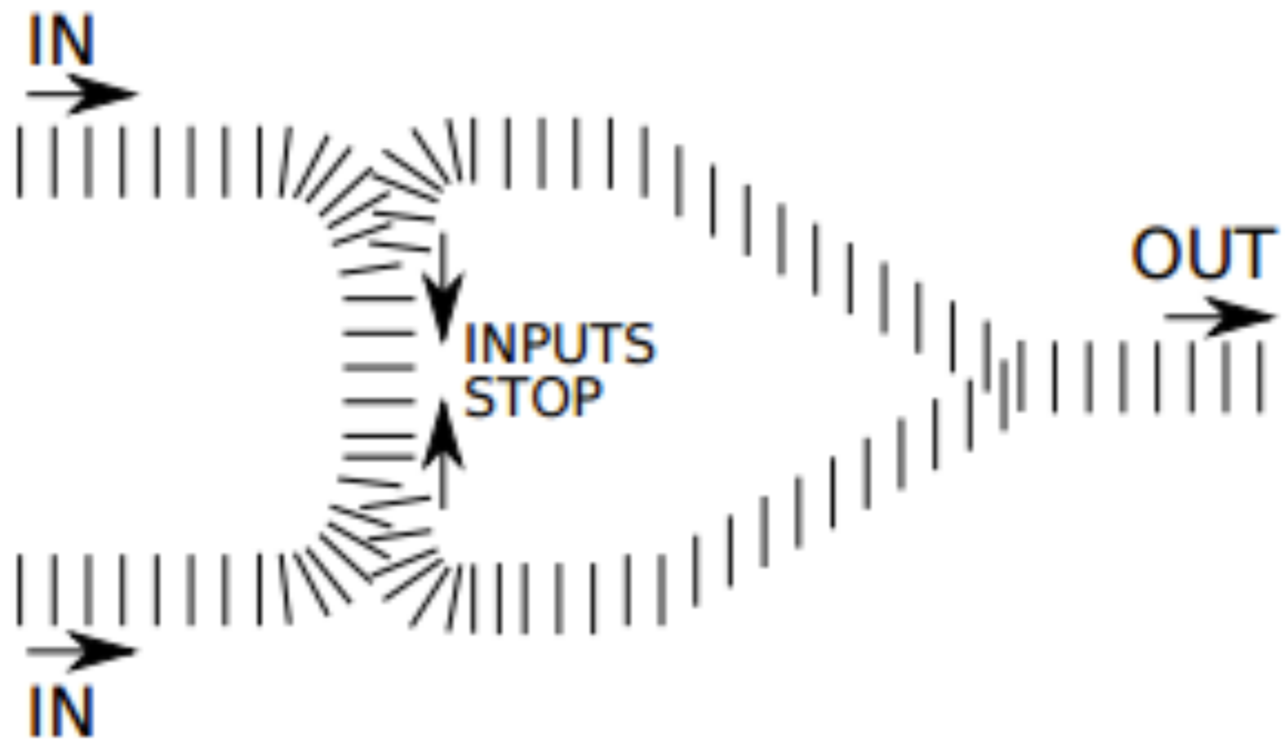


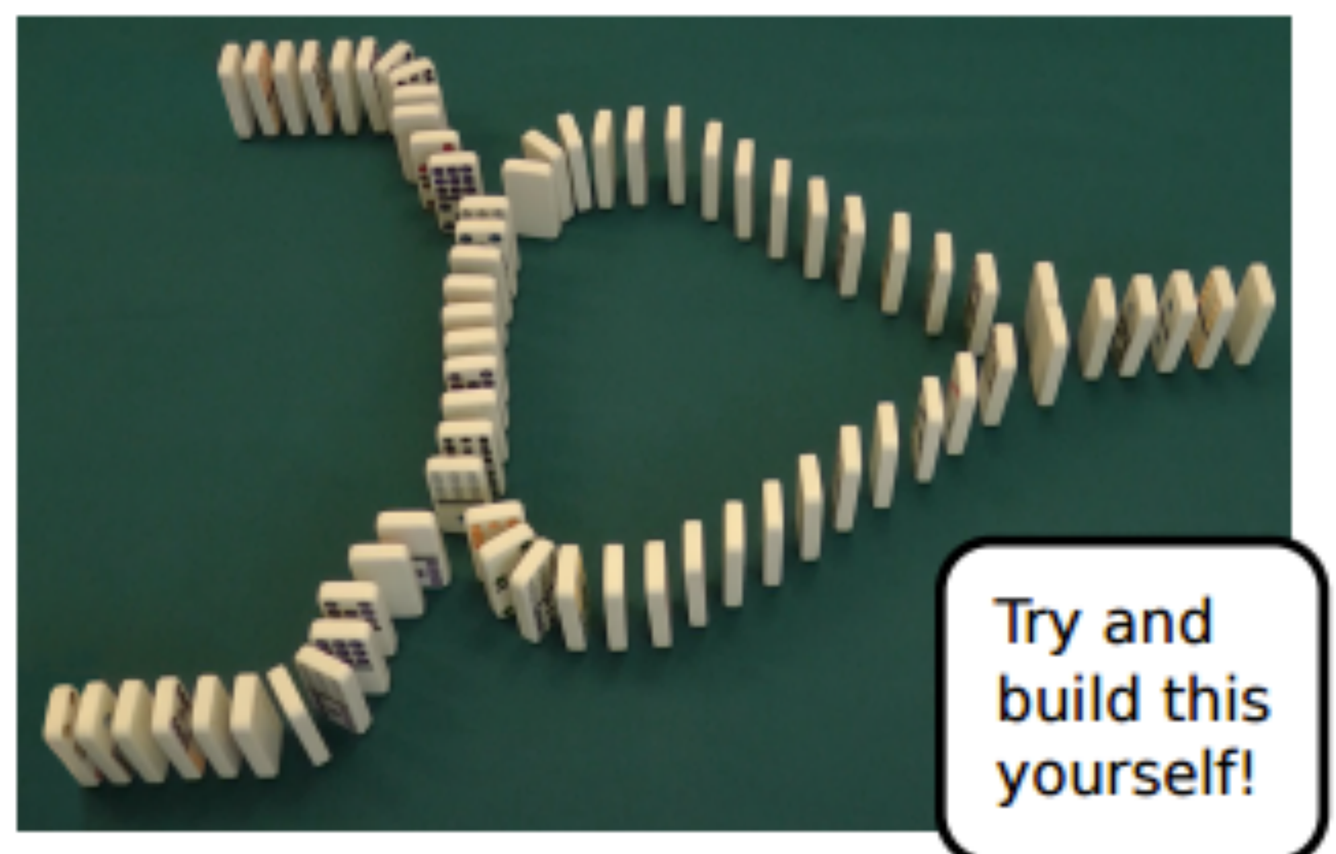


XOR

INPUT		OUTPUT
A	**B**	**A XOR B**
1	1	0
1	0	1
0	1	1
0	0	0

Sends output when it receives exactly one input





NAND

INPUT		OUTPUT
A	B	A NAND B
1	1	0
1	0	1
0	1	1
0	0	1

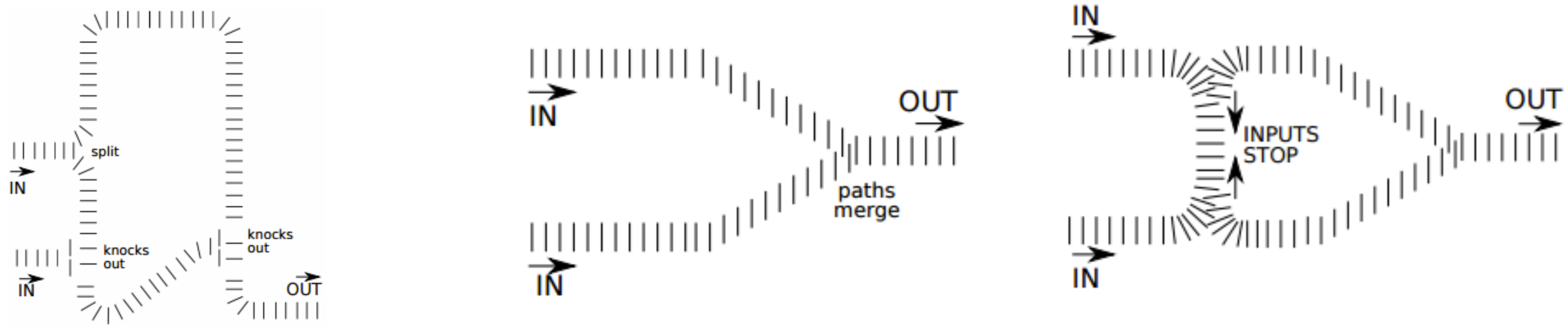
NOR

INPUT		OUTPUT
A	B	A NOR B
1	1	0
1	0	0
0	1	0
0	0	1

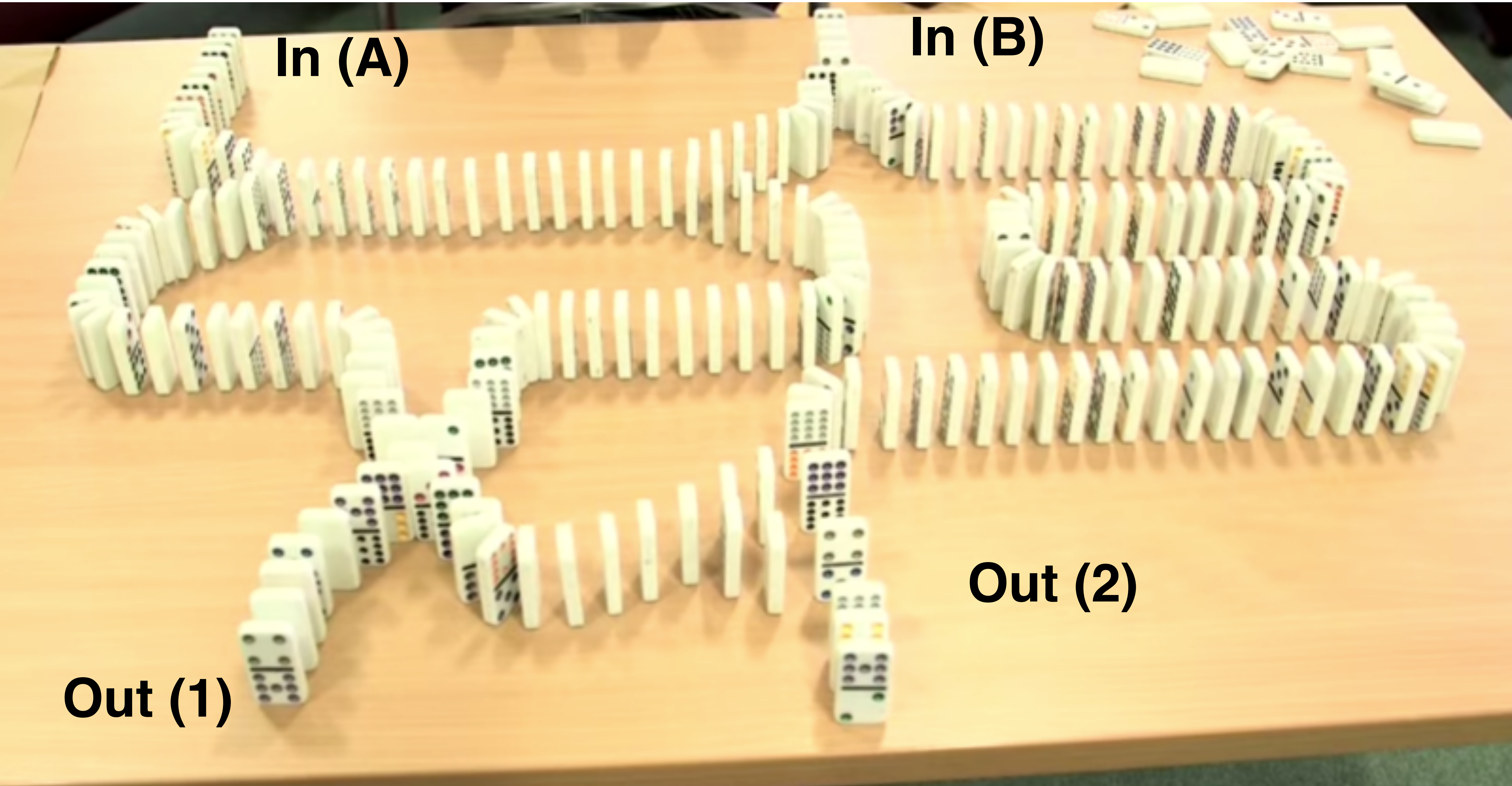
XNOR

INPUT		OUTPUT
A	B	A XNOR B
1	1	1
1	0	0
0	1	0
0	0	1

just throw a NOT before!



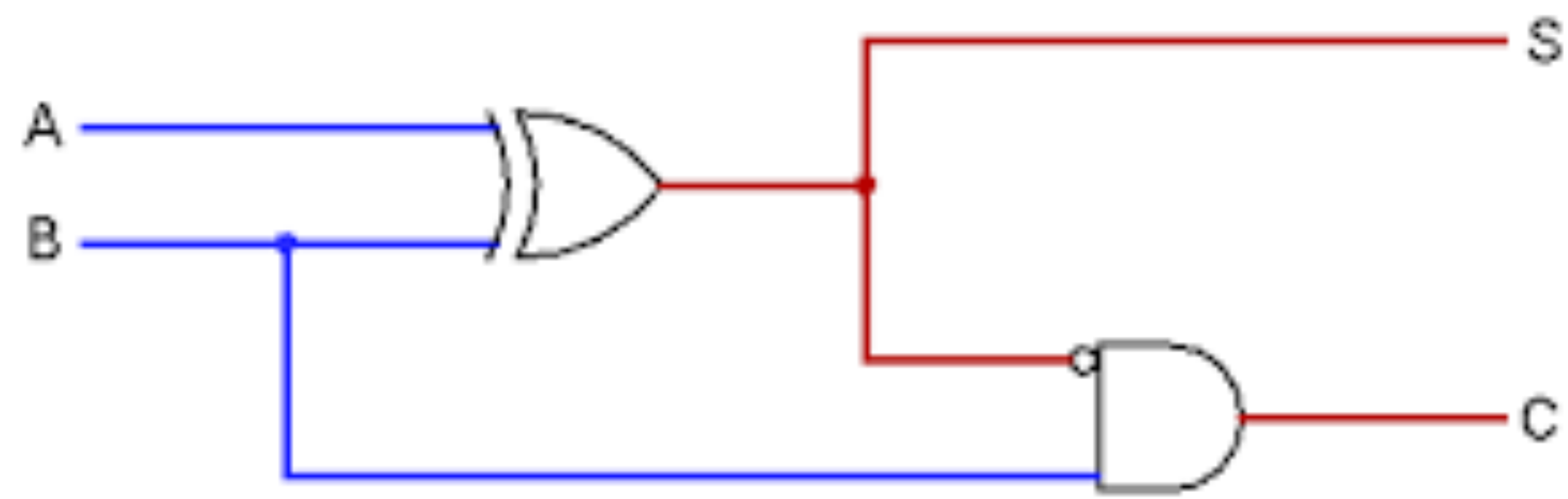
split
IN
knocks out
IN
knocks out
OUT
IN
OUT
IN
paths merge
IN
OUT
INPUTS STOP
IN



In (A)
In (B)
Out (1)
Out (2)

1-bit binary adder

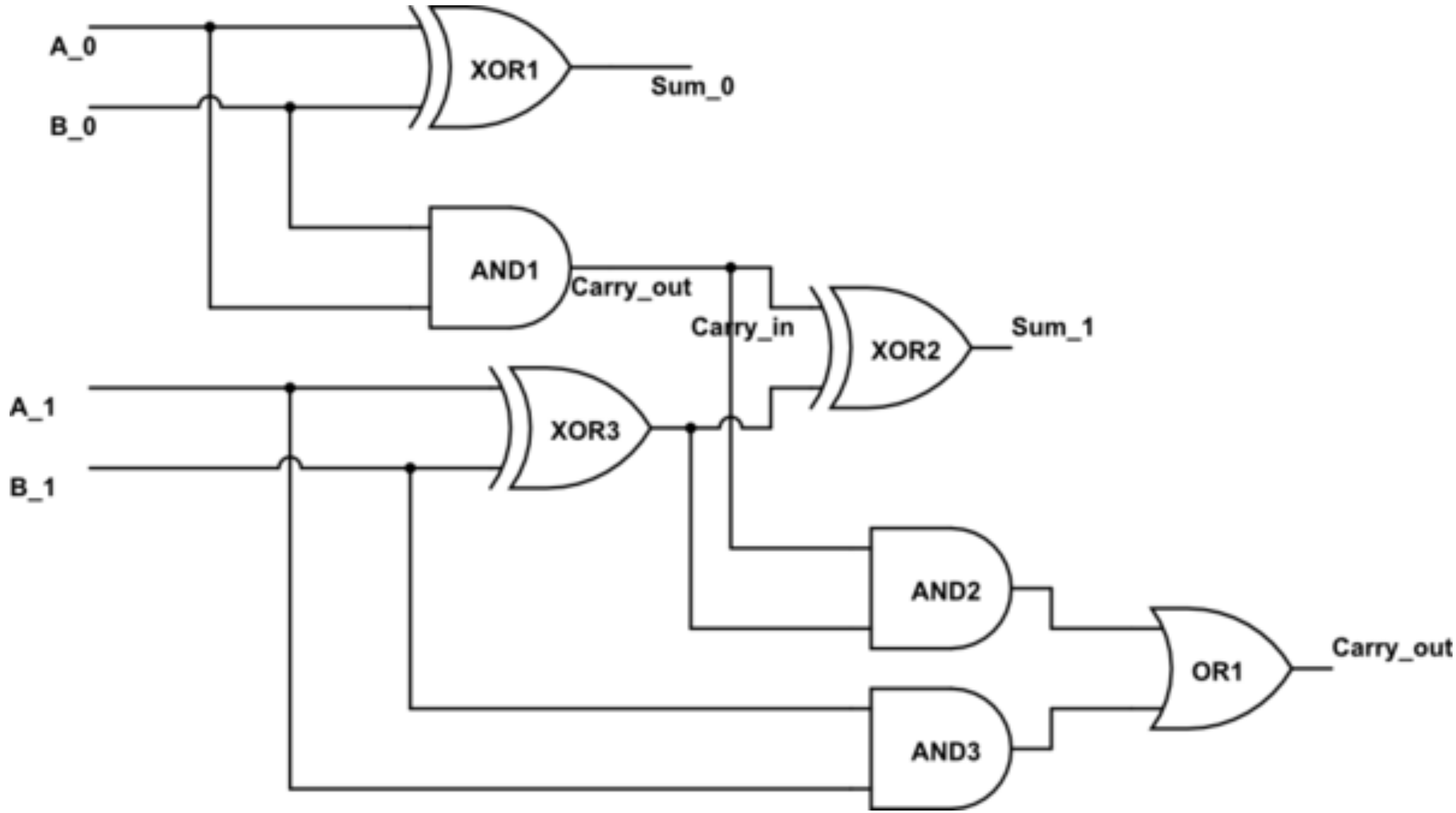
https://youtu.be/lNuPy-r1GuQ?t=10m1s



2-bit binary adder

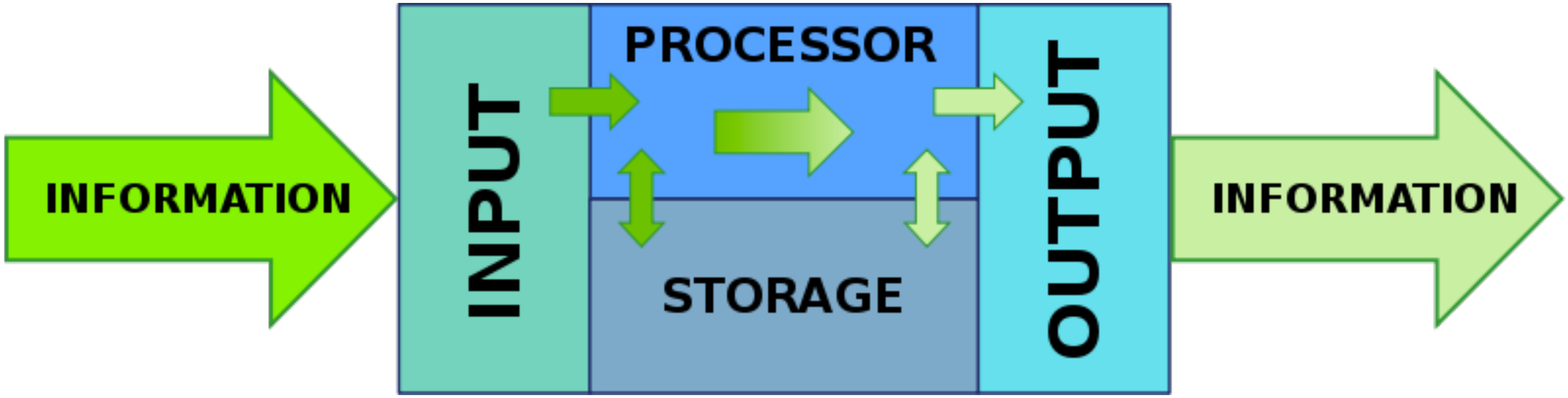
One's
place

Two's
place



https://youtu.be/lX-O3tyTncA?t=25s

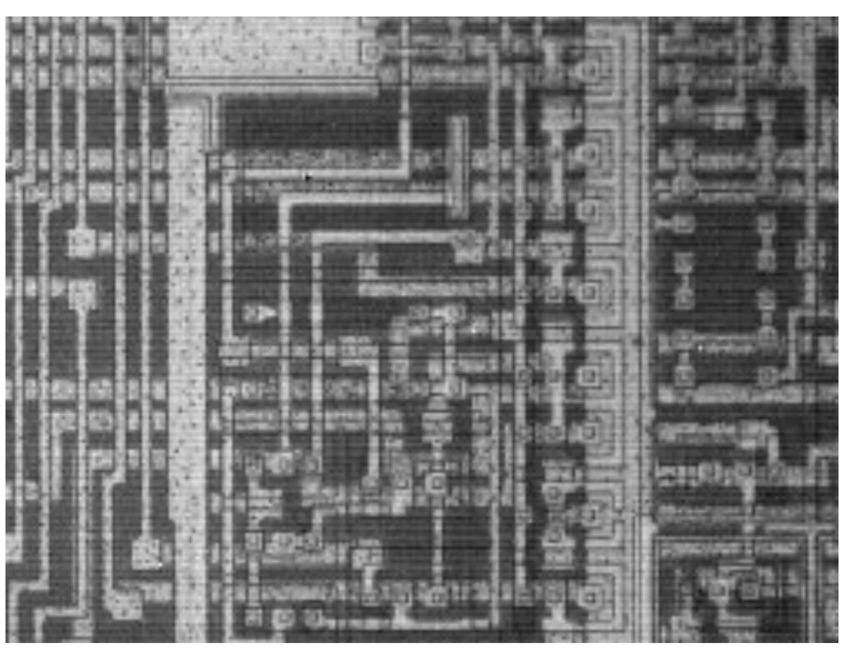
https://youtu.be/OpLU__bhu2w?t=15m28s

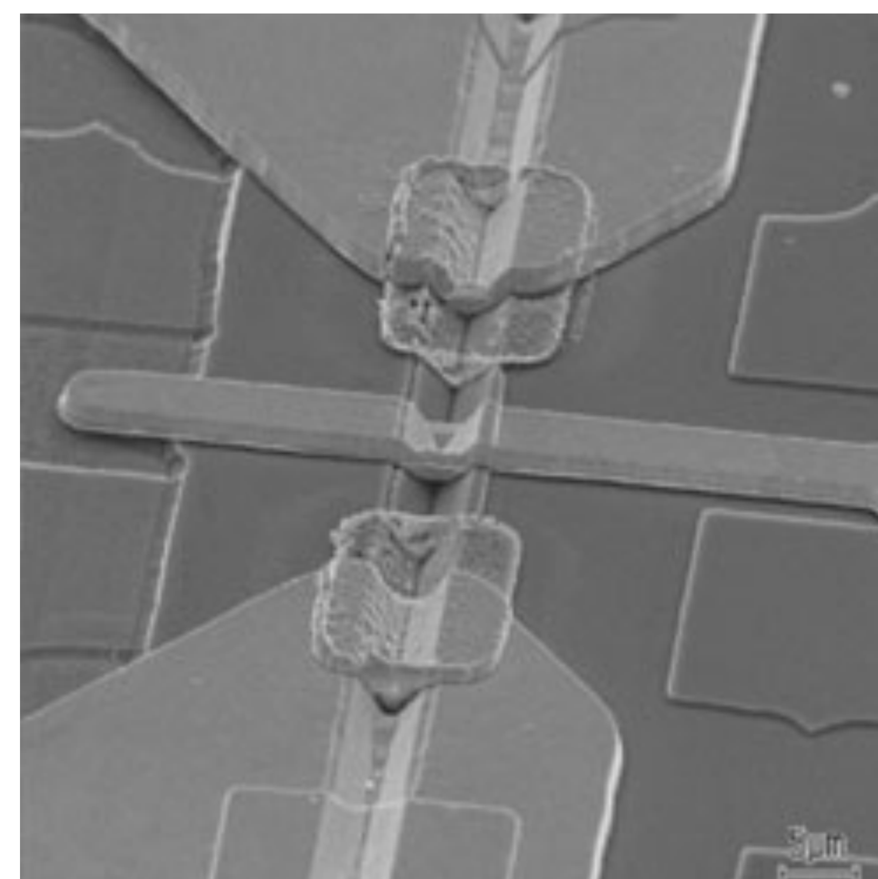


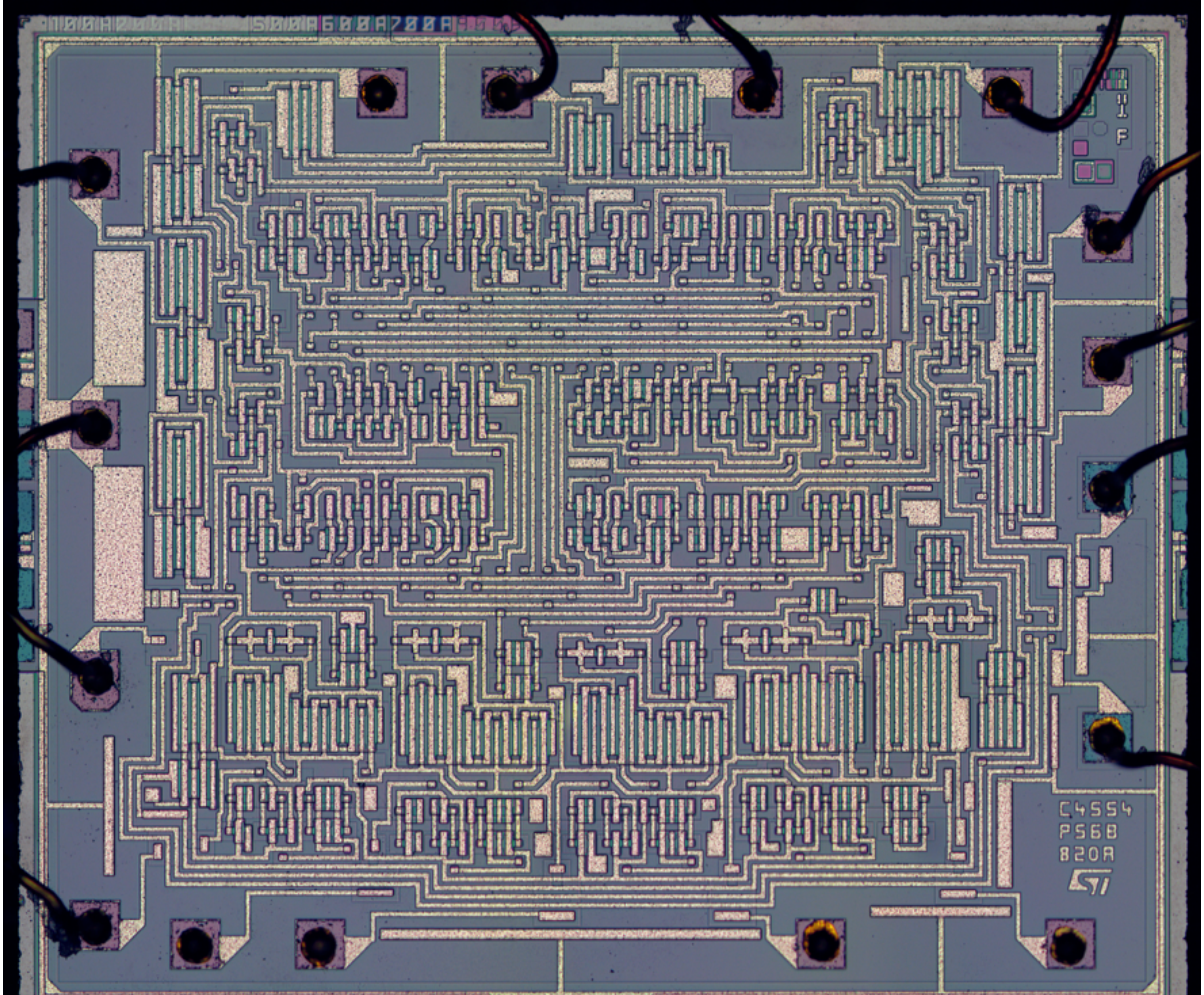
Year	Name	Operation	Input	Output
1833	Analytical Engine - Charles Babbage	Mechanical	Punched cards	Gear displays
1940s	Cryptography machines (bombes) - Alan Turing	Electro-Mechnical	Dials	Rotary displays
1950s	ENIAC	Electric, **vacuum tubes**	Punch cards	Card punch
1960s	Harwell CADET	Electric, **transistors**	Punch cards	Card punch
1970s	Integrated circuit computers	Electric, semiconductors	Punch cards, keyboards	Punch cards, monitors displays

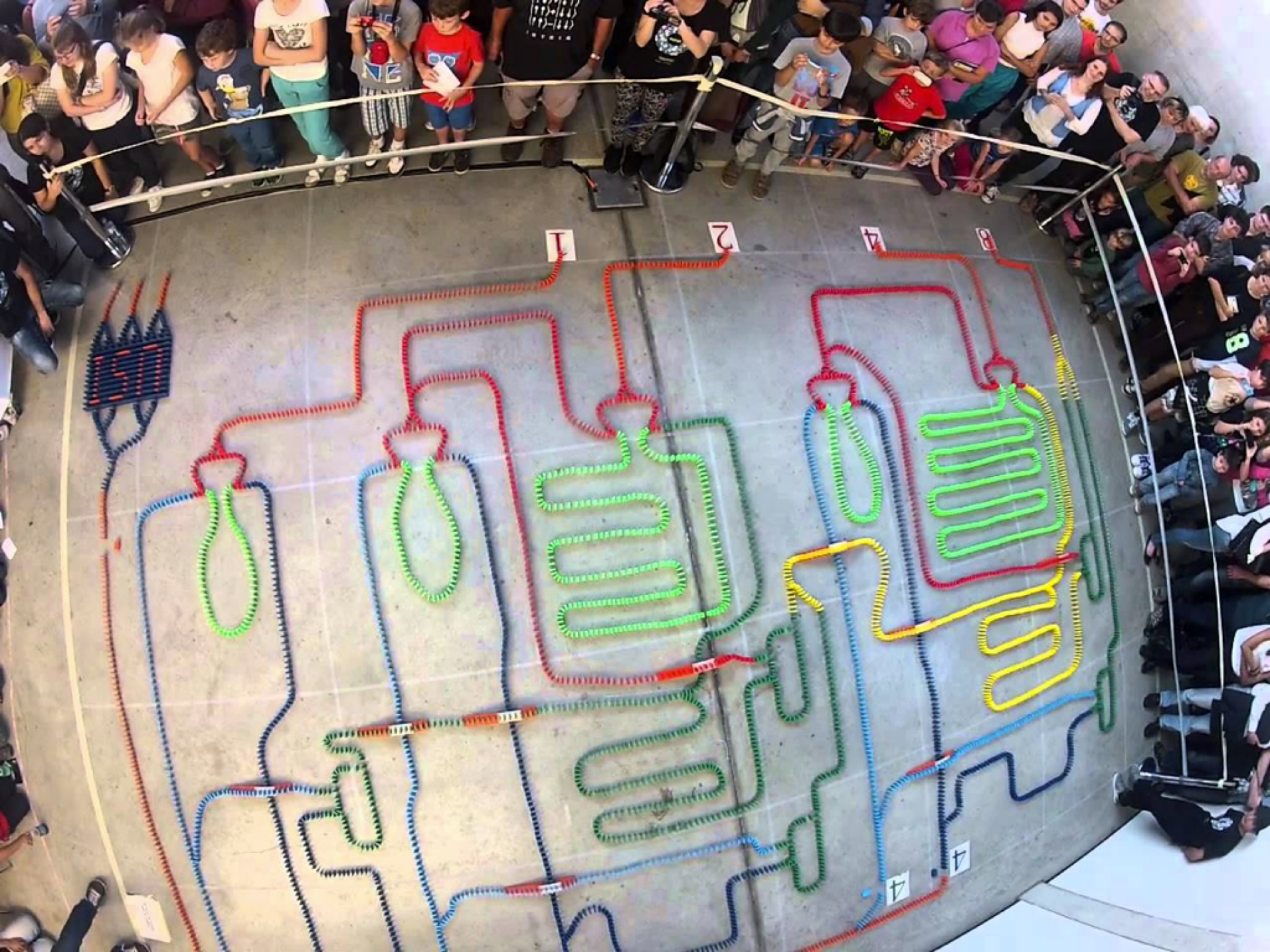
Semiconductors

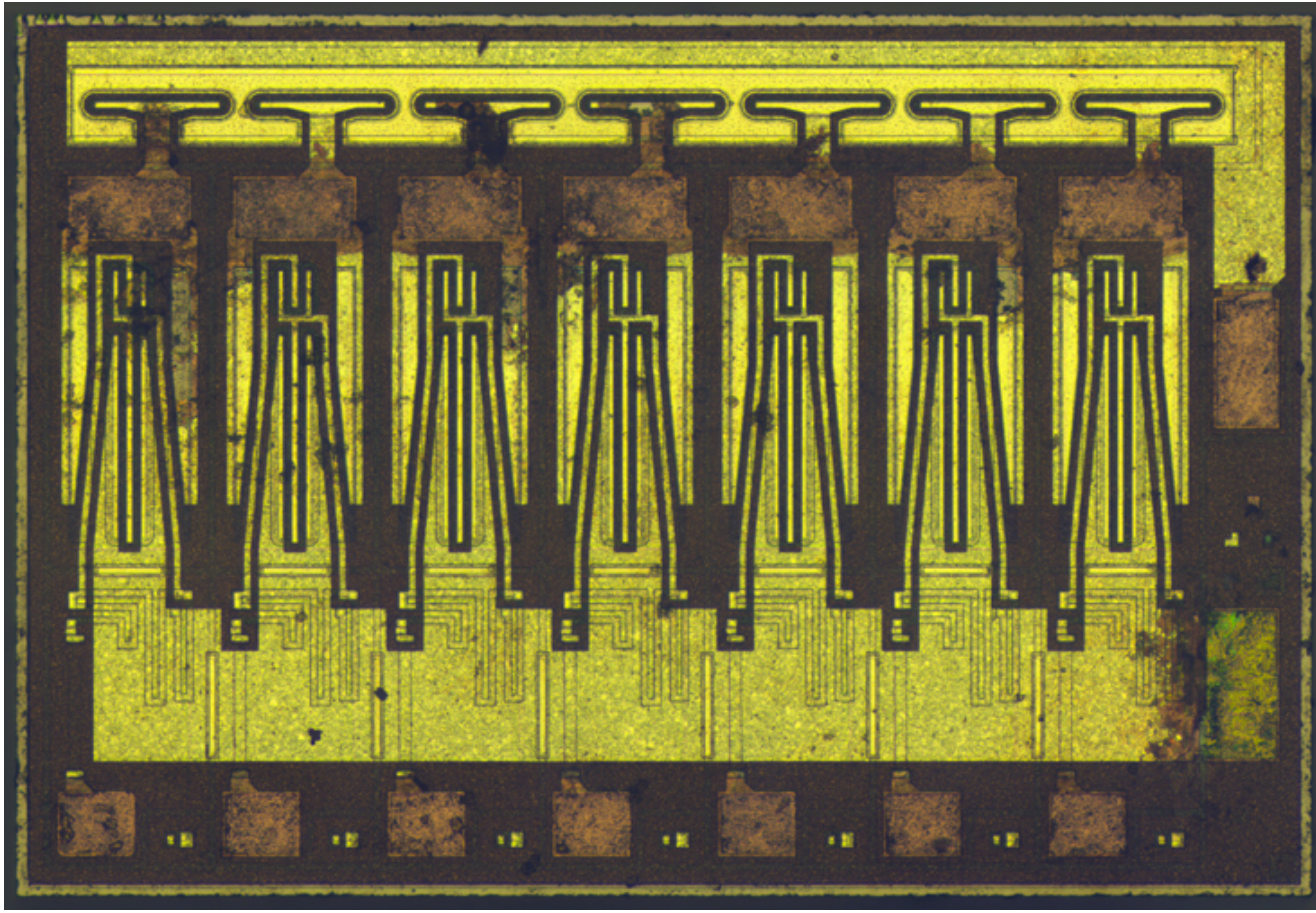
- They are of high resistance - higher than typical resistance materials, but still of much lower resistance than insulators
- Their resistance decreases as their temperature increases (opposite of metal)
- Doping with impurities can alter behavior in useful ways, including:
    - passing current more easily in one direction than the other
    - showing variable resistance
    - sensitivity to light or heat

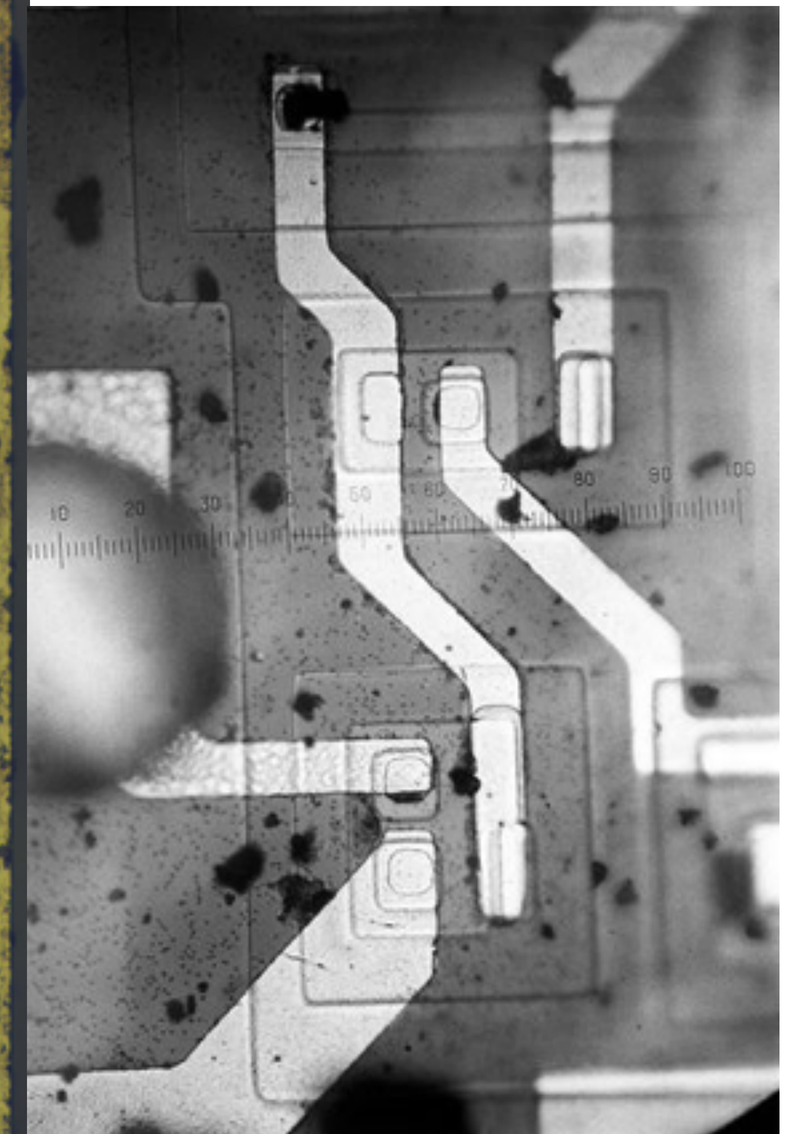


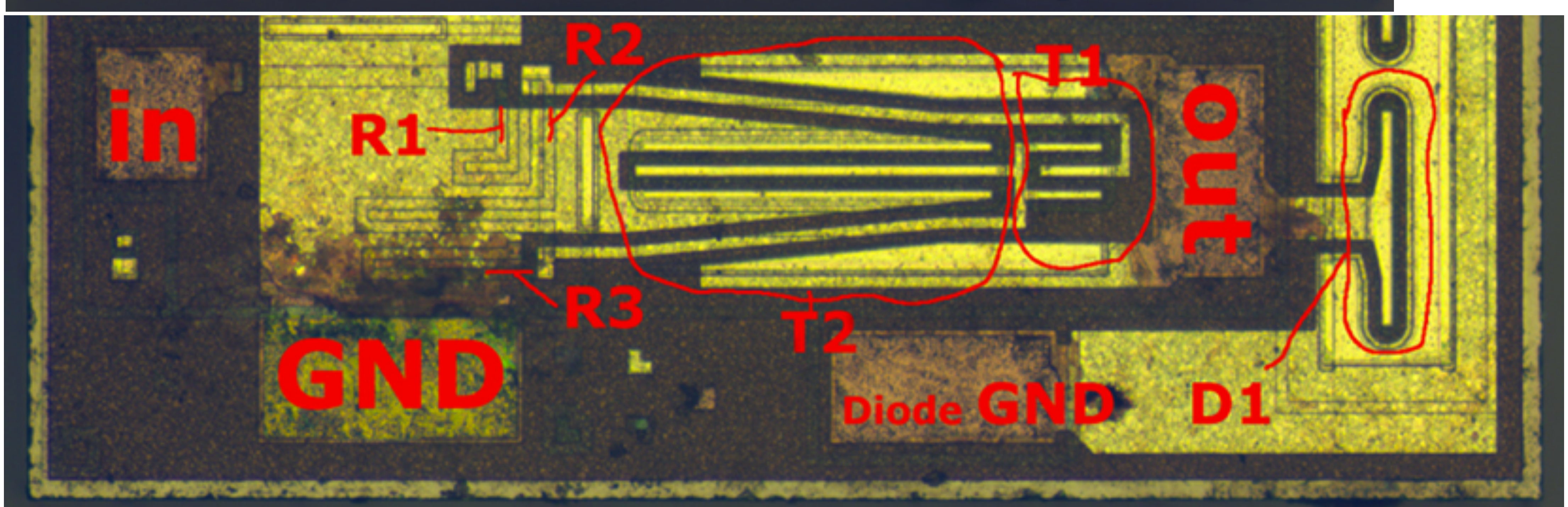



C4554
PS6B
820A


1
2
4
8
4
4






in
R1
R2
R3
T1
T2
out
GND
Diode GND
D1

But remember, though we are using insanely small circuits shuttling electrons around (soon to be light and/or quantum states) - you can still make a computer out of pretty much anything.

Like pulleys:

https://vimeo.com/93042377