

Variables

Review

- A variable is a named location in memory.
 - A variable has a type (thereby size) and a location in memory.
- A variable needs to be *declared* before use. Syntax: `type name;`
 - The first assignment to a variable is called *initialization*.
 - A variable contains junk between declaration and initialization.
- An *array* is a *contiguous* block of elements of the *same type*.
Syntax: `type name[number];`
 - Fixed size
 - Access/modify by index, syntax: `name[index]`. This index is not checked.
- A *string* is a NUL-terminated array of characters.

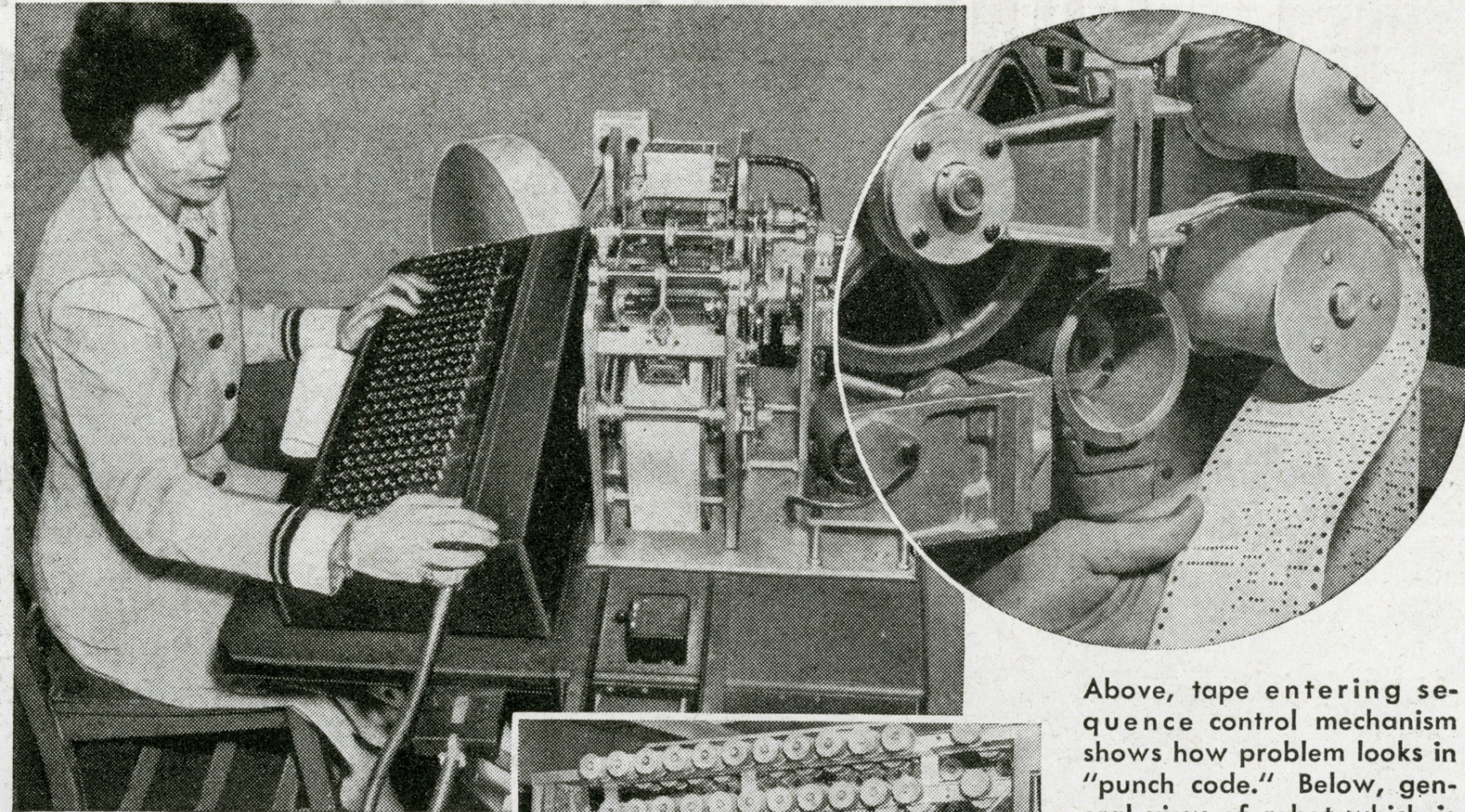
Relays, Vacuum Tubes, Transistors, and Computation

CS143: lecture 2 (cont)

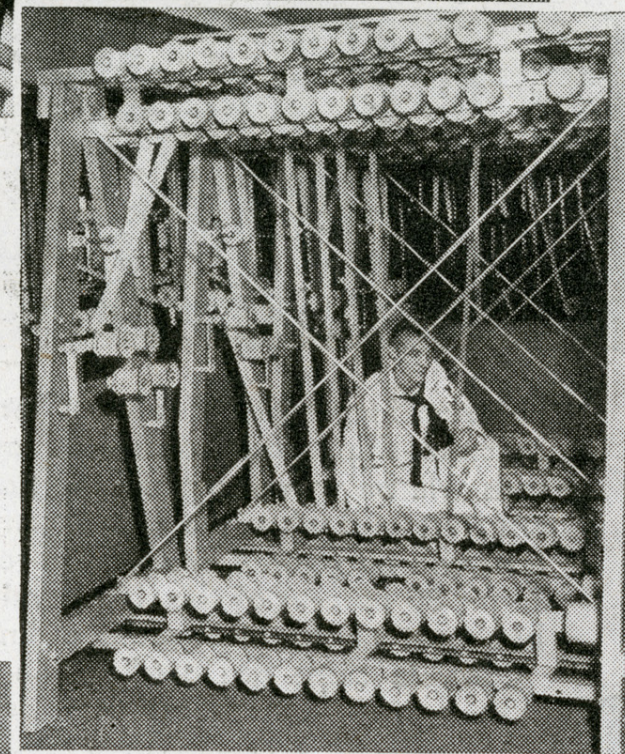


IBM Automatic Sequence Controlled Calculator (Harvard Mark I)

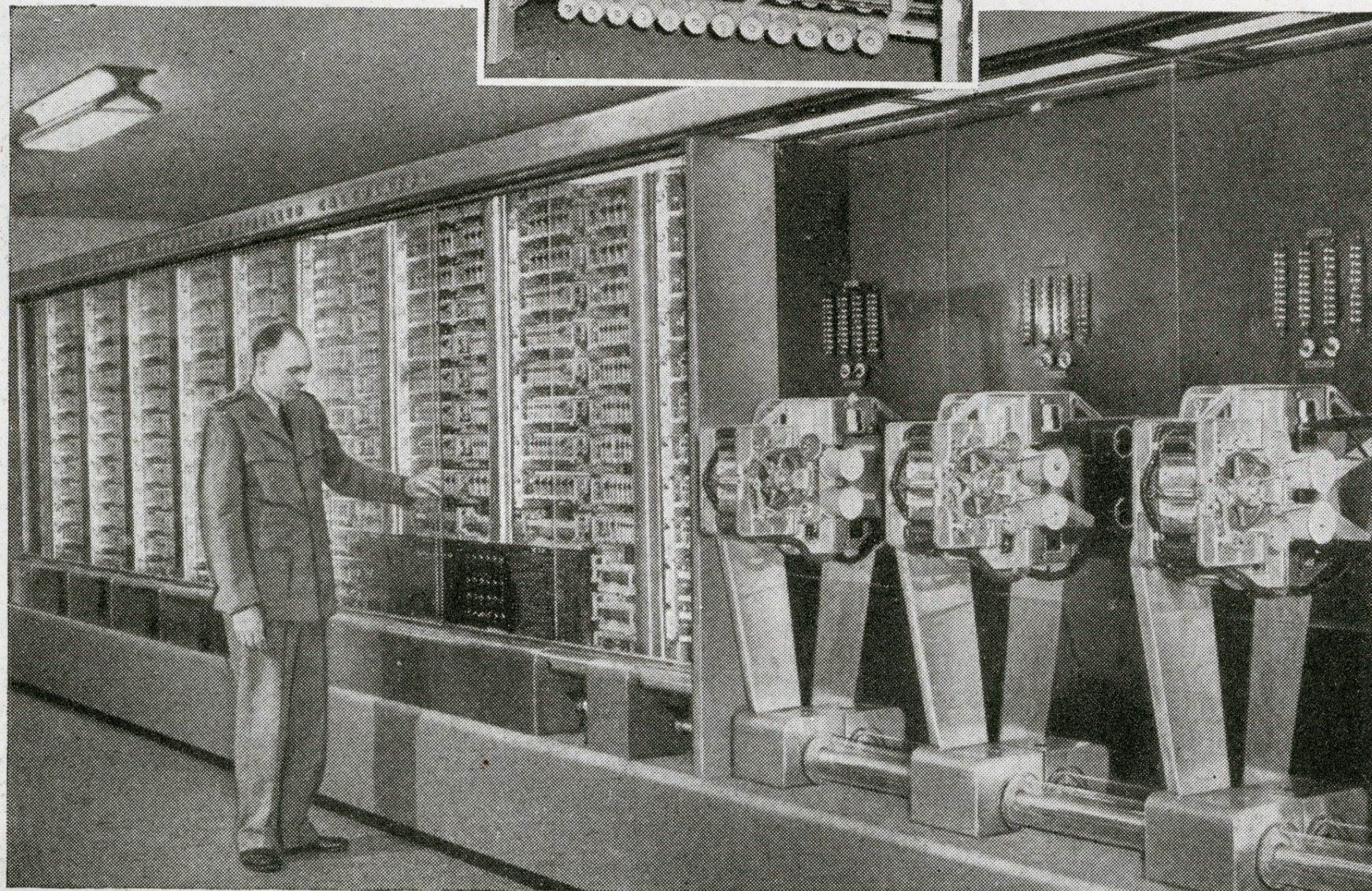
Robot Works Problems Never Before Solved



Solving problems which stumped mathematicians throughout history, is all in the day's work to the "world's greatest calculating machine" which gives accurate answers in 23 figures. Above, preparing a problem for the machine on manual tape punch which dictates operation of the "superbrain" with coded perforations. Center, system of holders on which tape moves

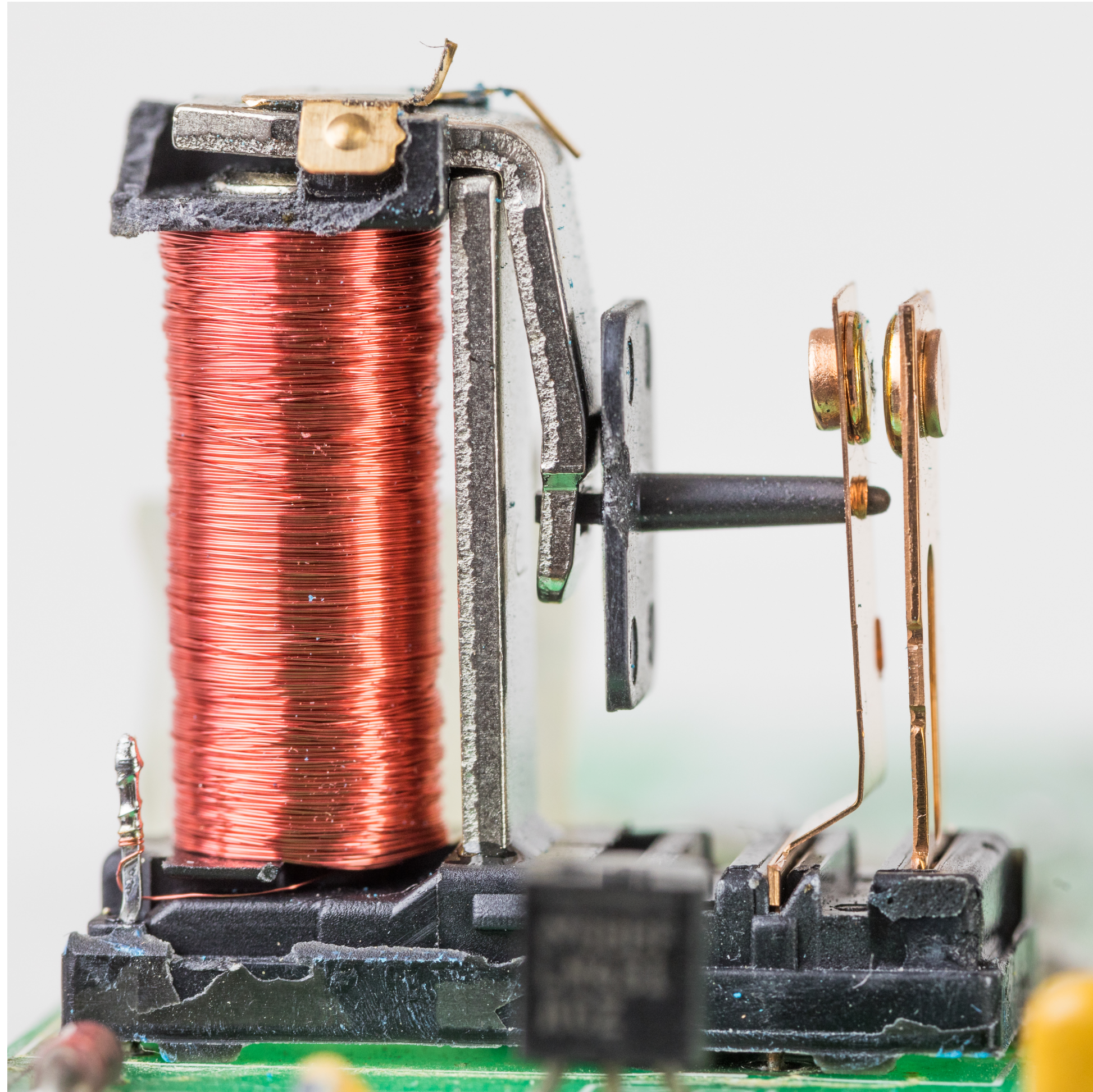


Above, tape entering sequence control mechanism shows how problem looks in "punch code." Below, general view of robot which is 51 feet long, 8 feet high. It has 500 miles of wire, 3,000,000 connections, tiers of 72 adding machines. Invented by Cmdr. H. H. Aiken, U.S.N.R., it was built by International Business Machines and presented to Harvard University for use by the Navy. After the war it will solve problems of star movements and algebraic equations hitherto unsolved

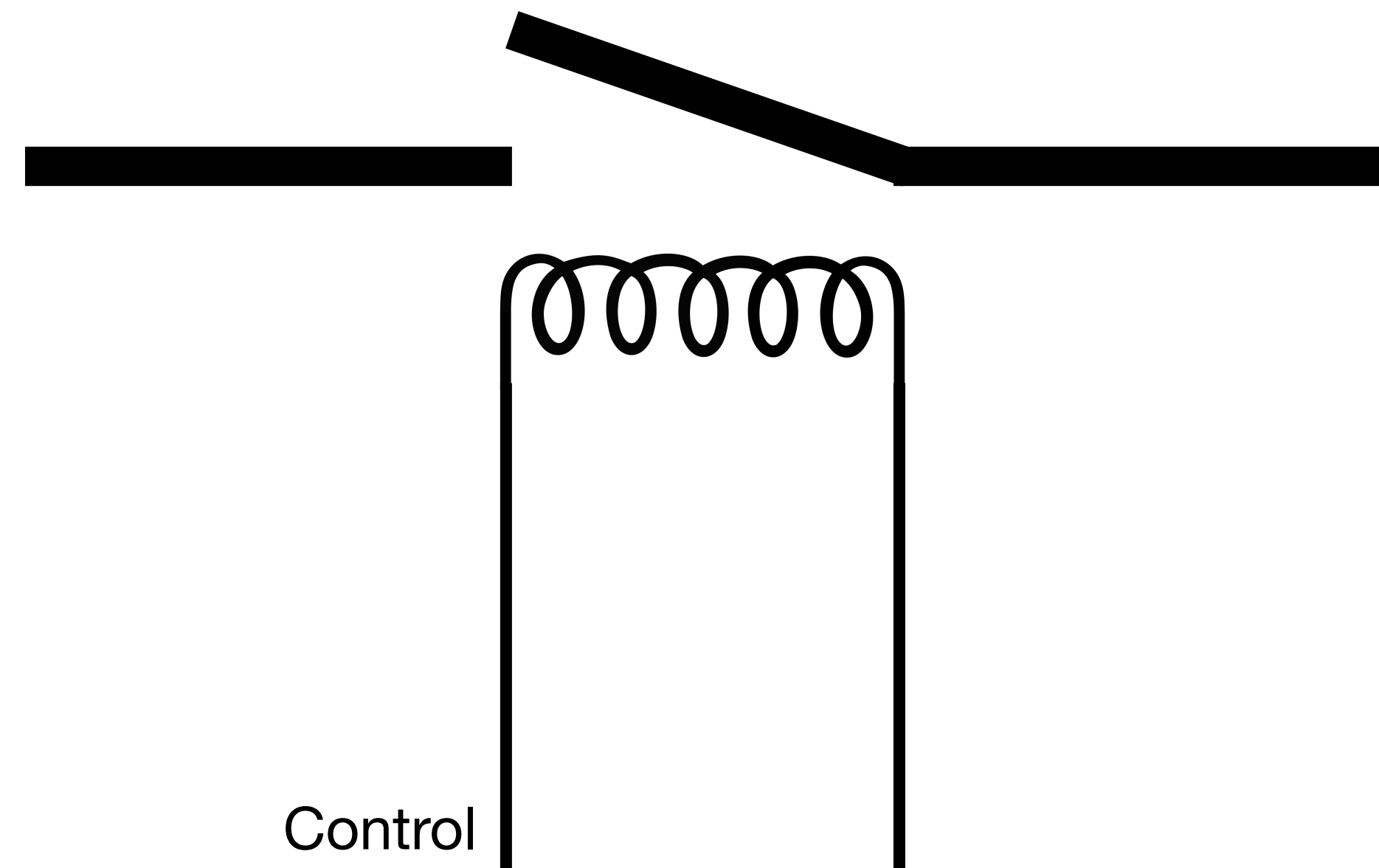


- Designed by Howard H. Aiken in 1937
- Built by IBM and delivered to Harvard in 1944, in operation until 1959
- 765,000 electromechanical components
- 3 million connections
- 500 miles of wire
- Does numerical calculations
- "Computer" used to be a job title

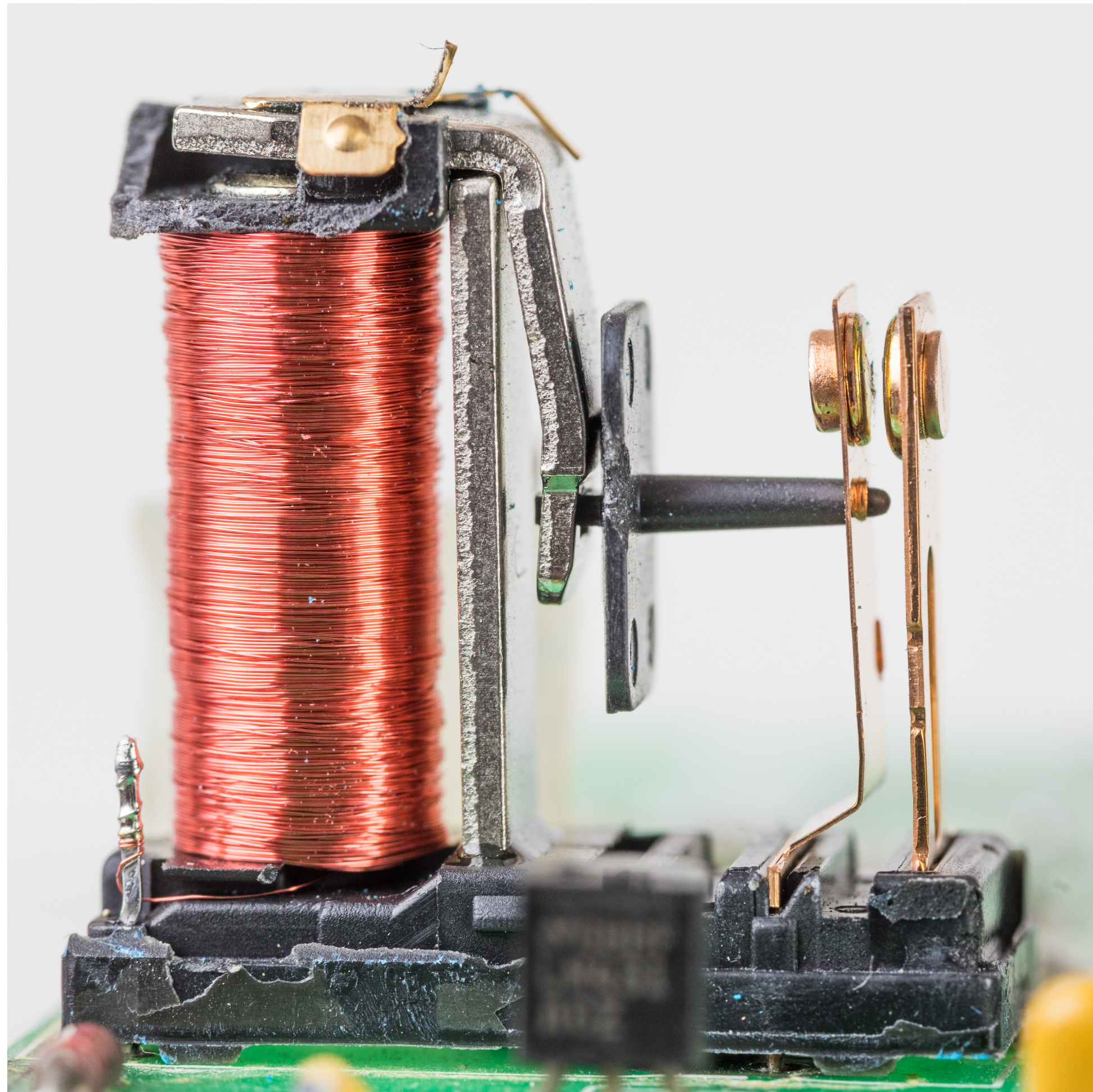
Relay



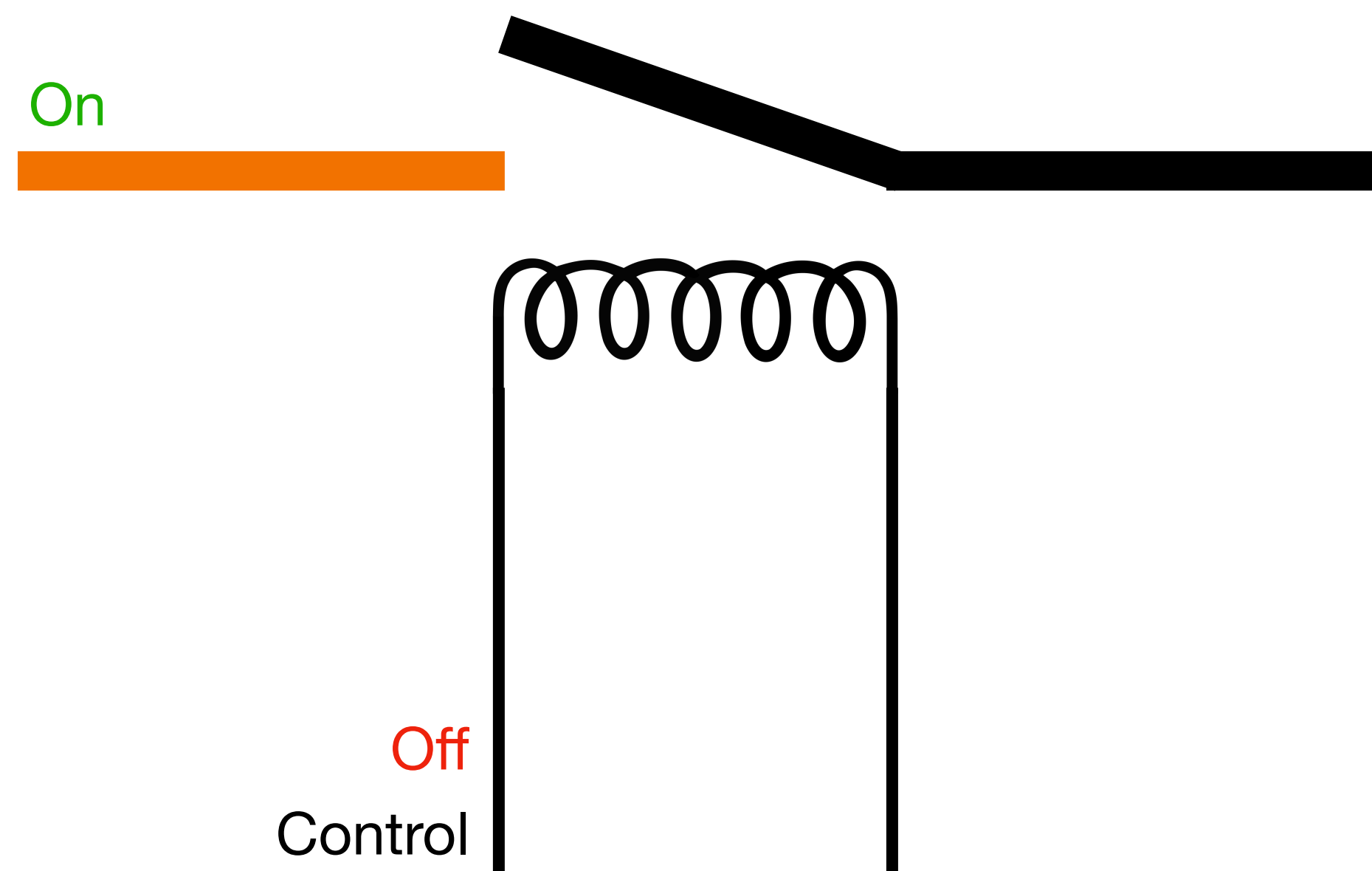
a relay switch



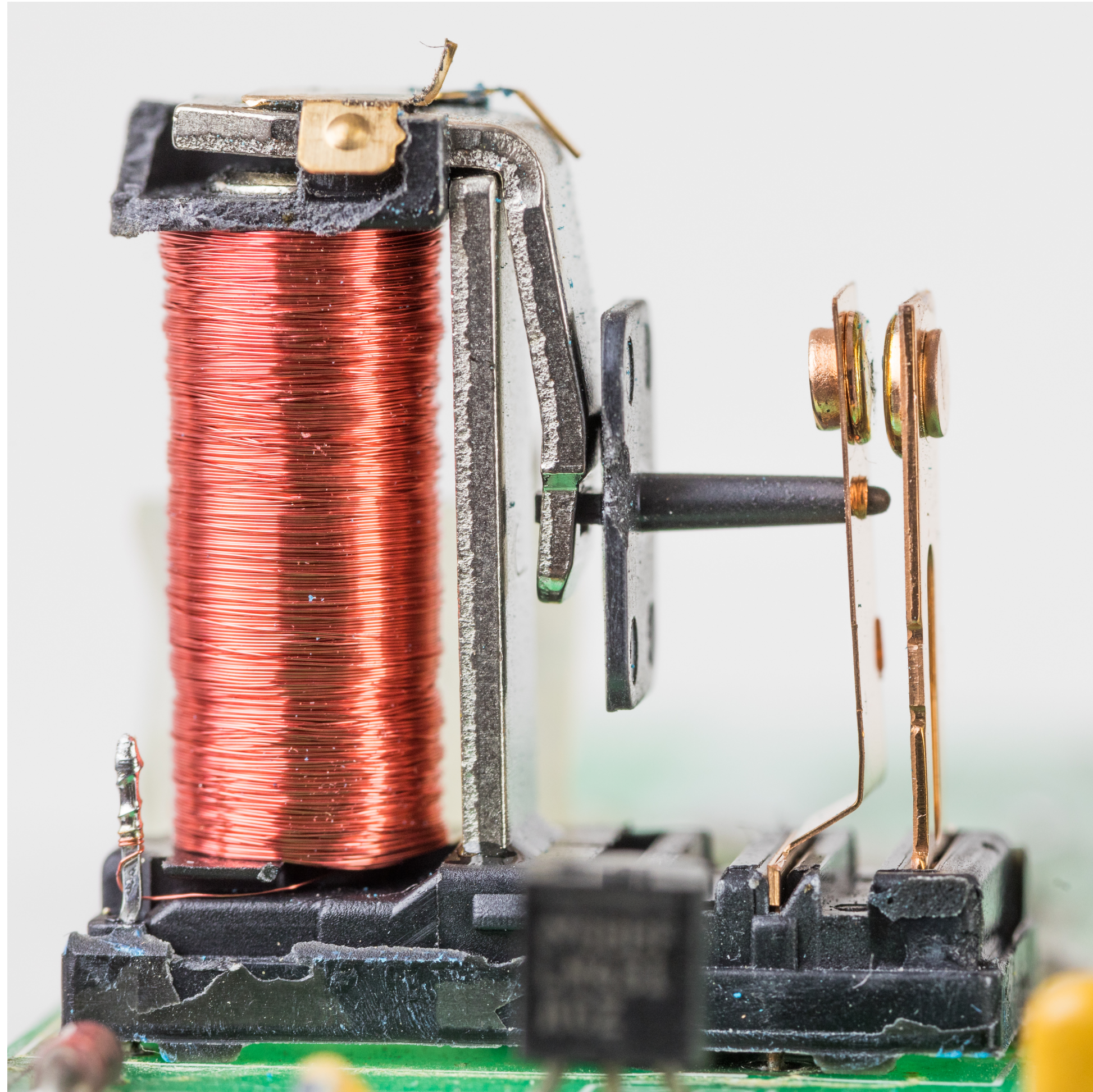
Relay



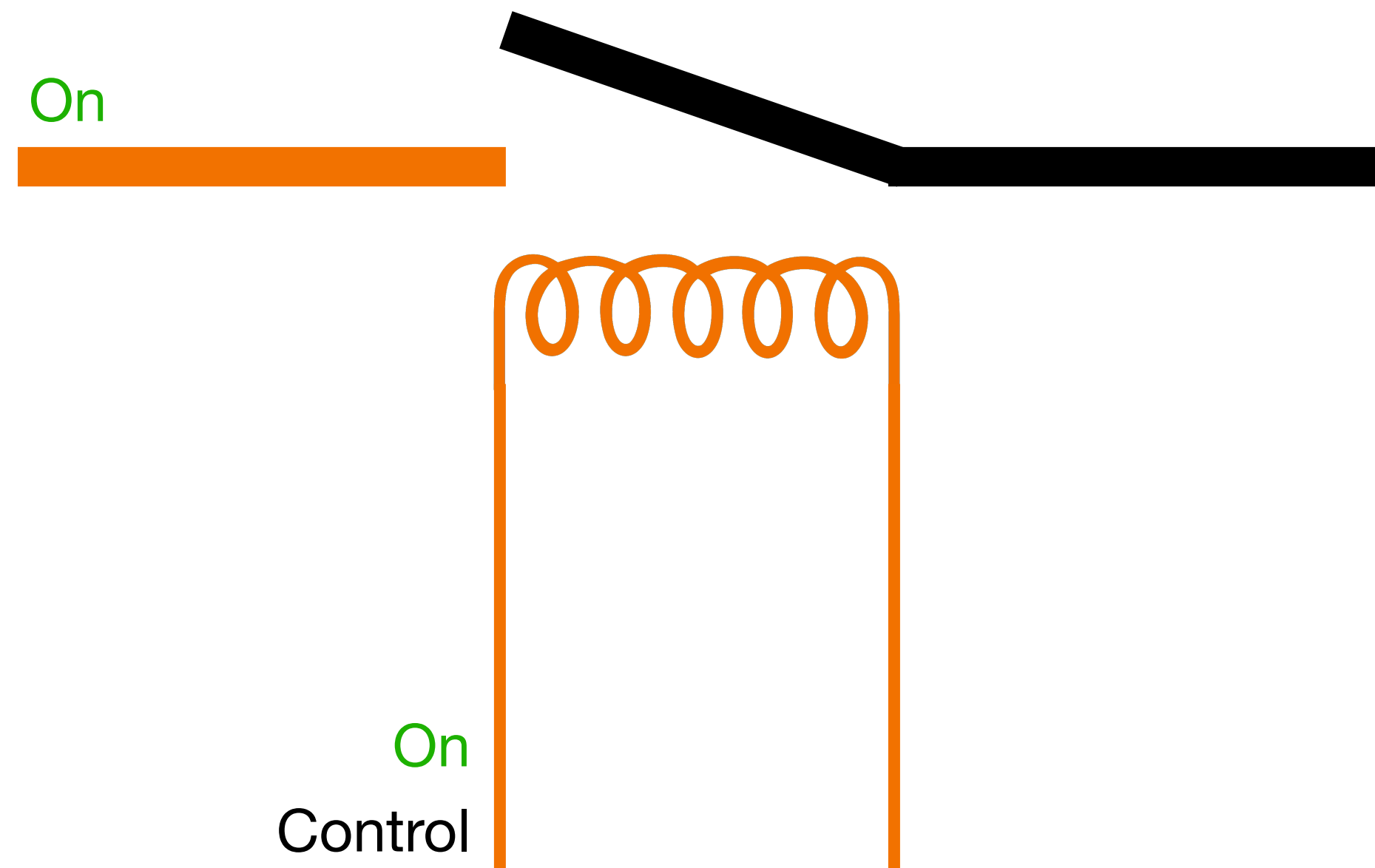
a relay switch



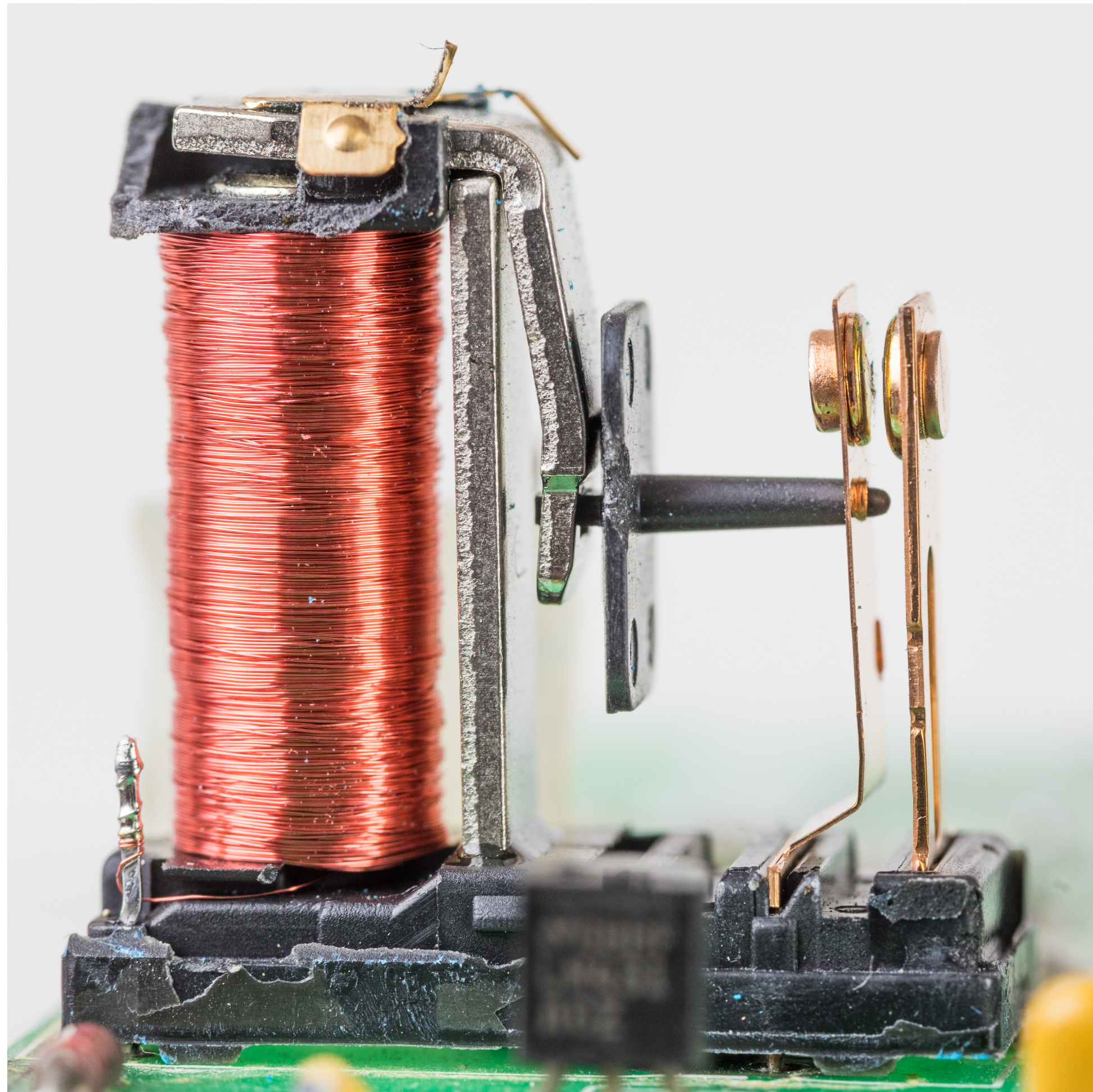
Relay



a relay switch



Relay



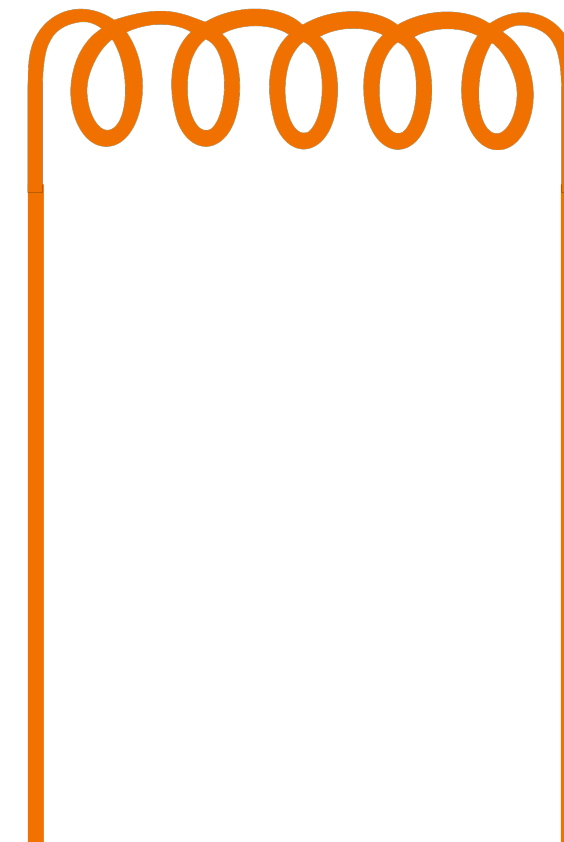
a relay switch

On

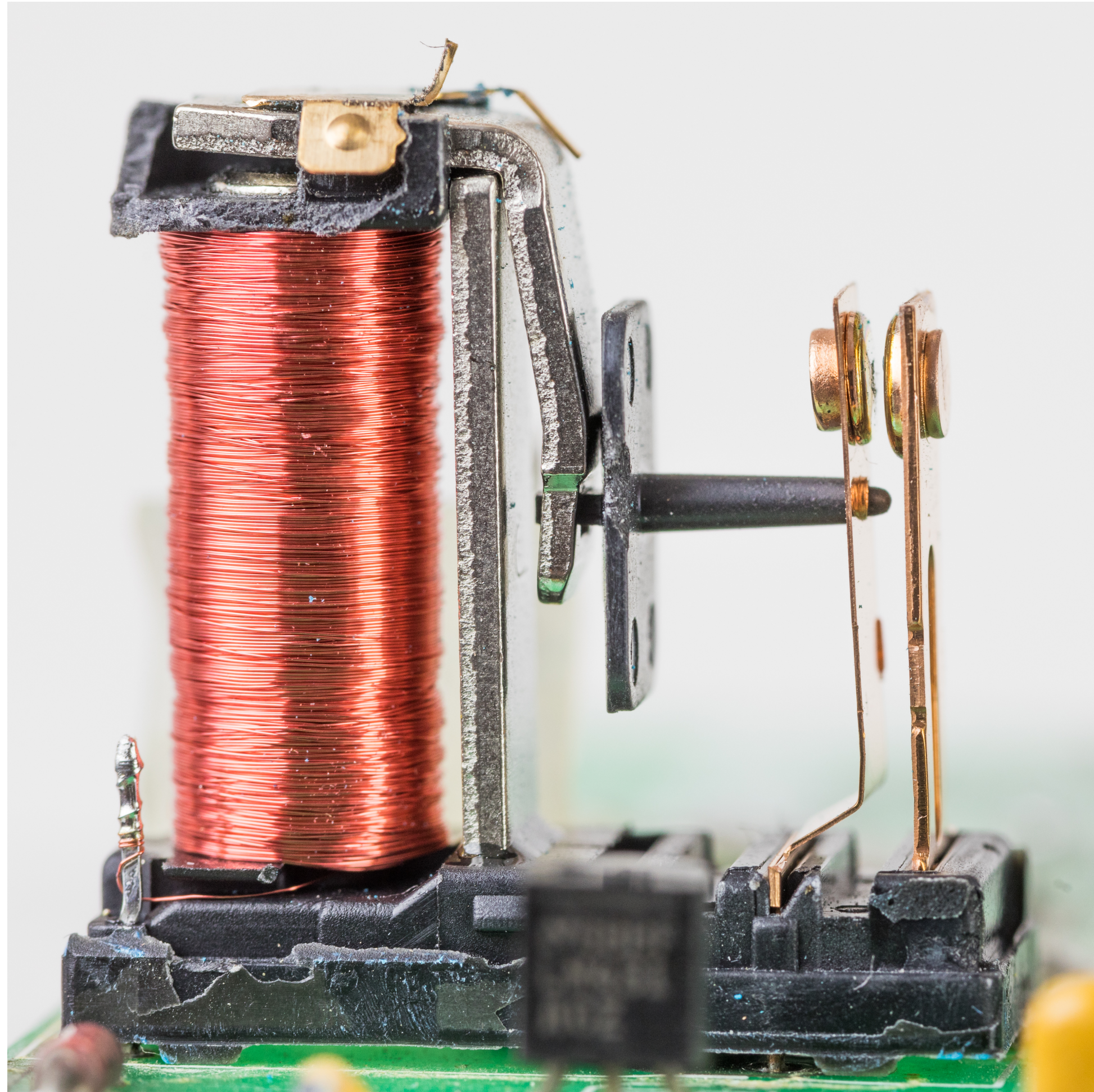


On

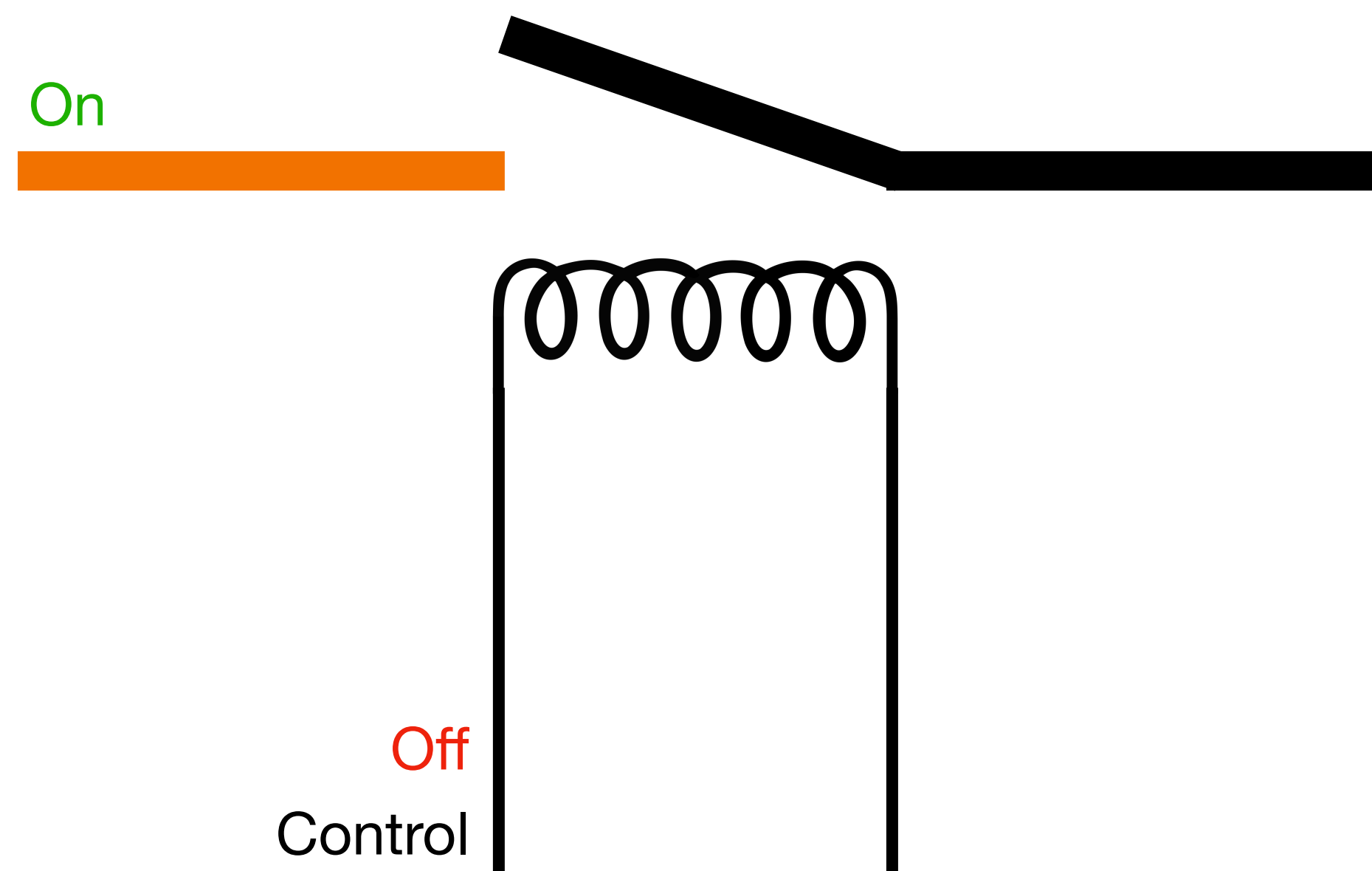
Control



Relay

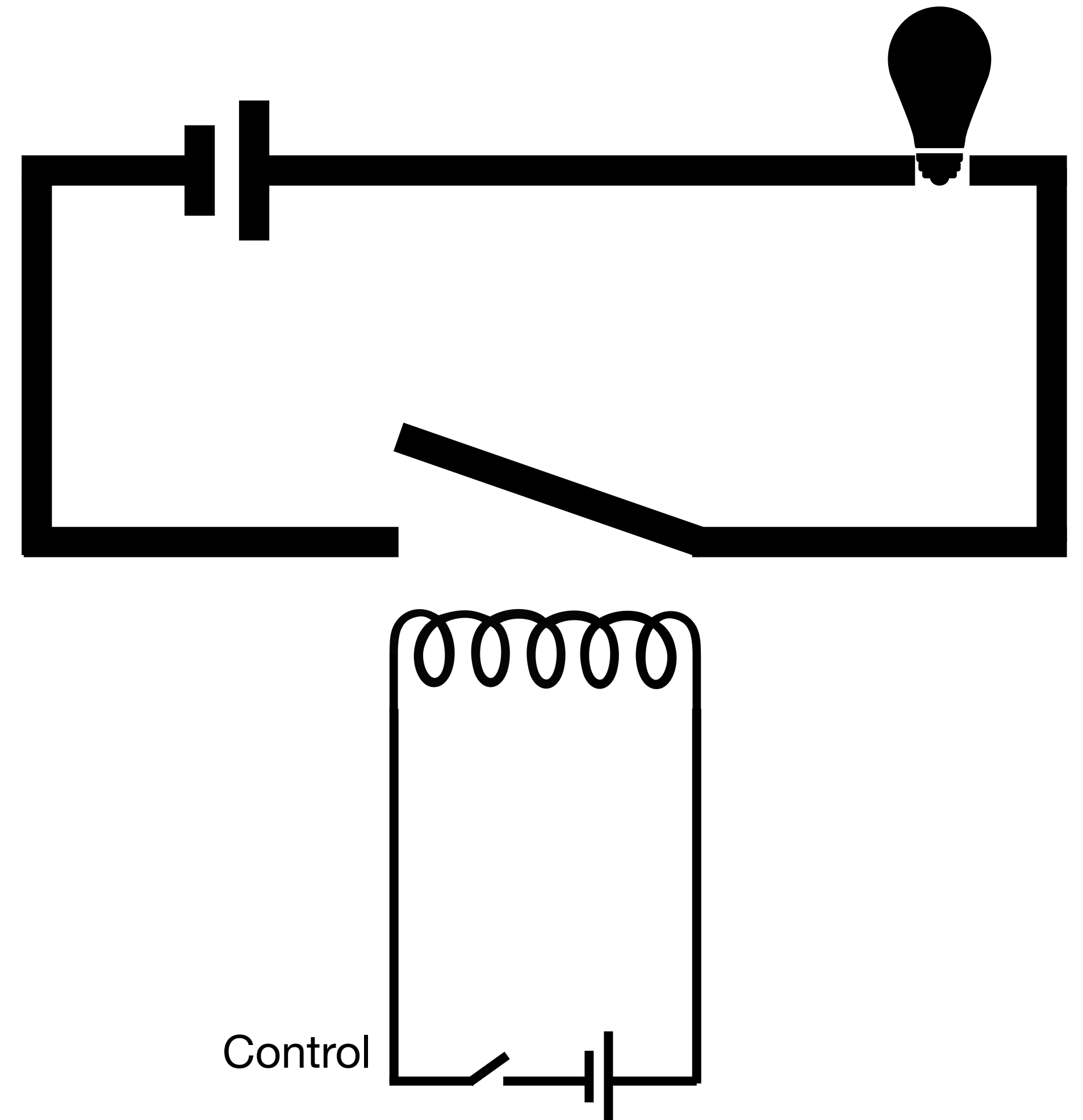


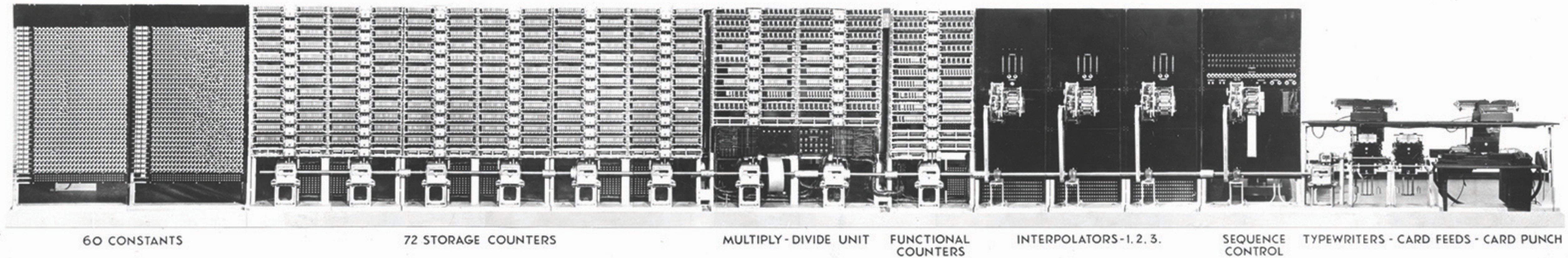
a relay switch



Relay

- Relay telegraph signal over long distance
- Mechanical things that have mass
- Slow: open and close ~15 times a second
- Wear and tear

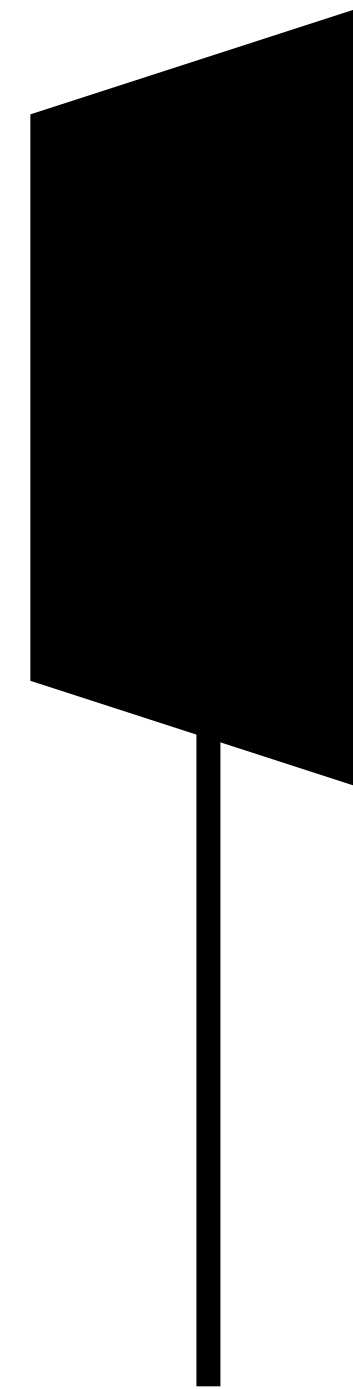
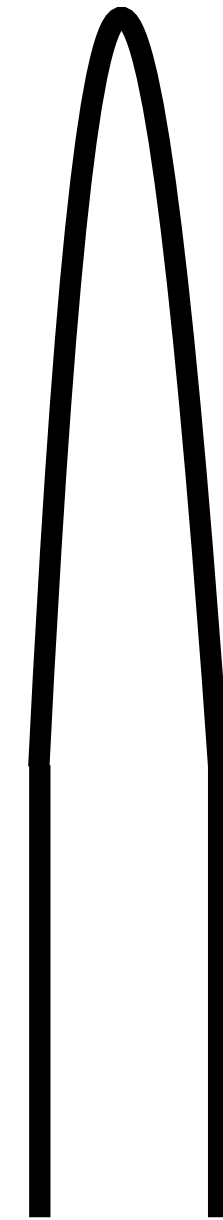
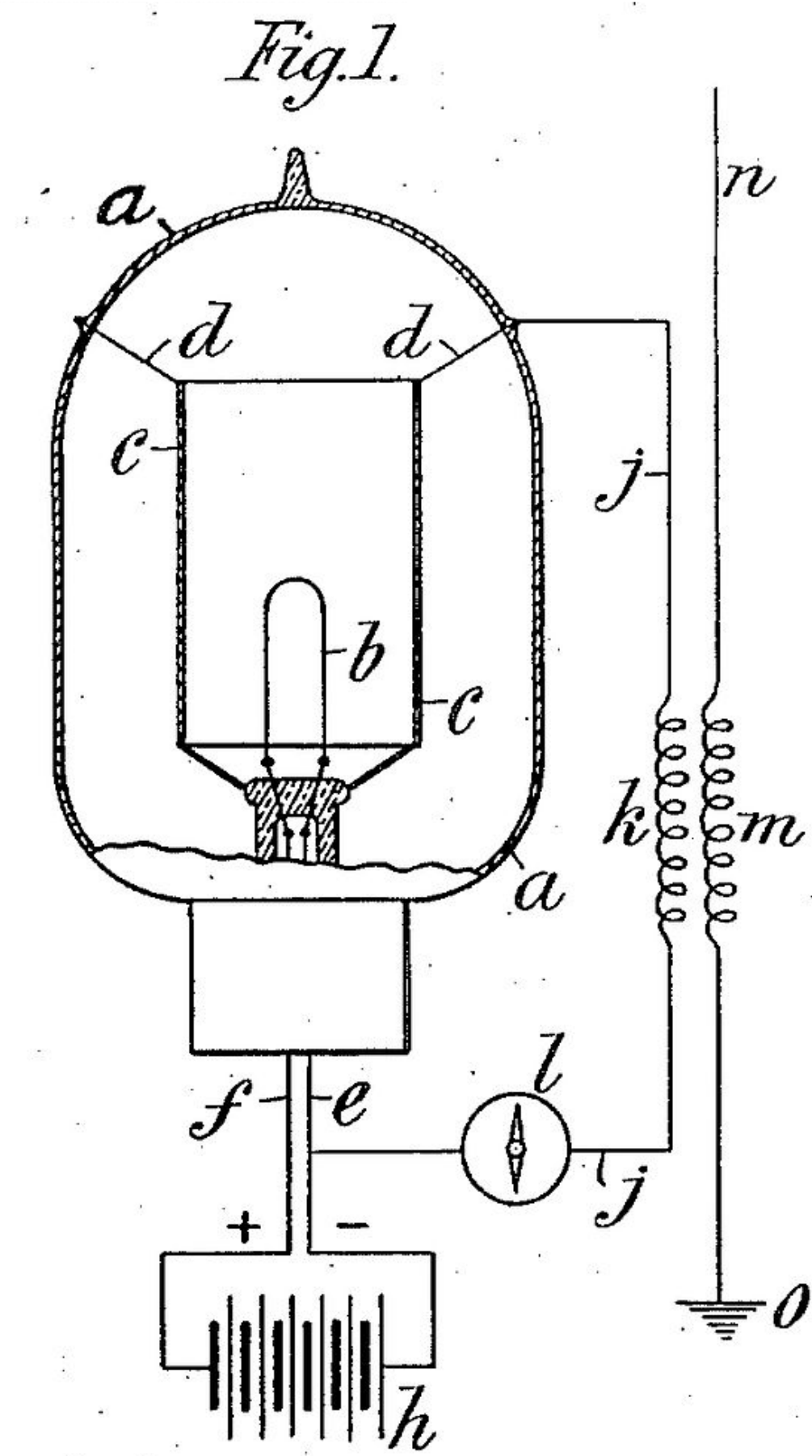




- 3 additions or subtractions per second
- 1 multiplication took 6 seconds; a division took 15.3 seconds
- logarithm or trigonometric functions took over a minute

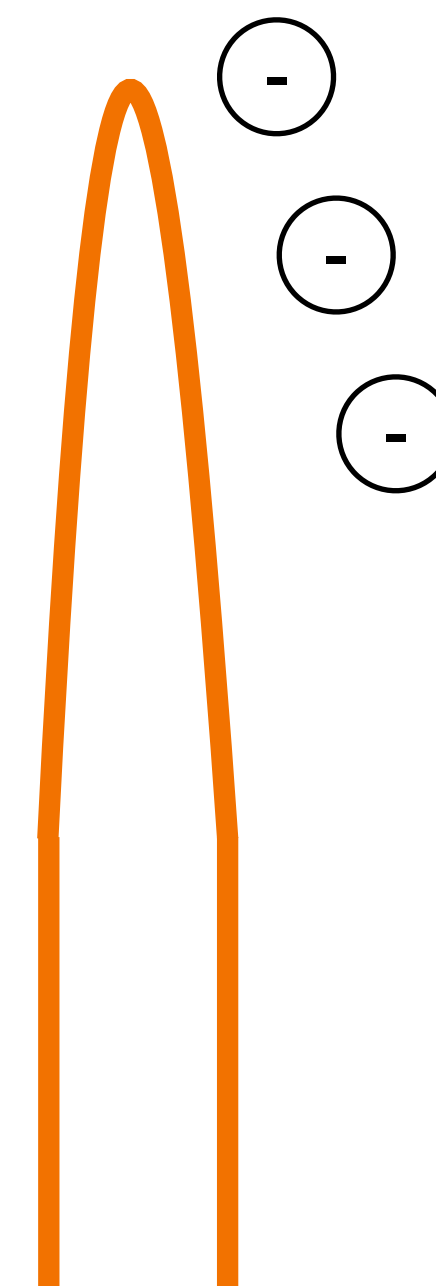
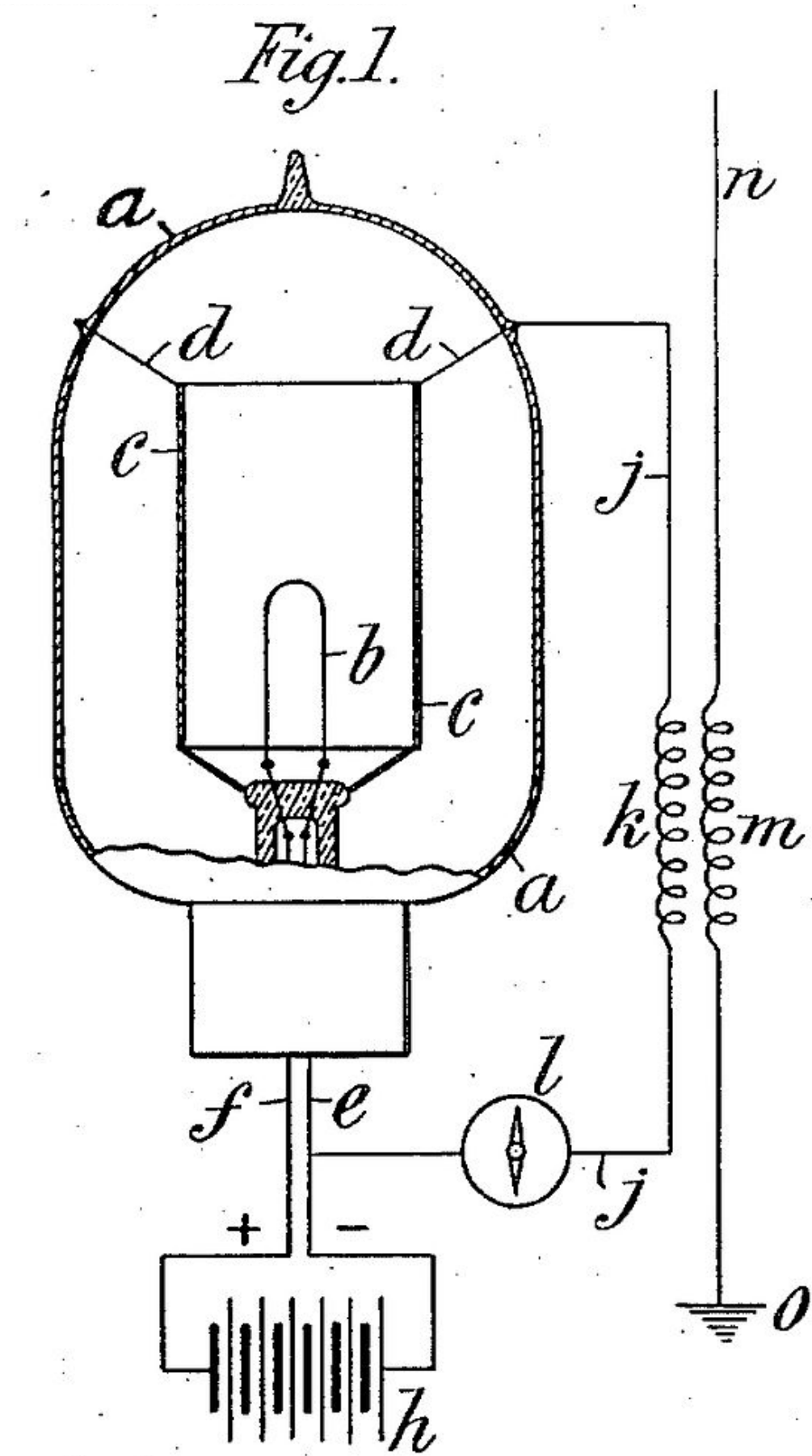
Vacuum Tube

Fleming Valve, 1904



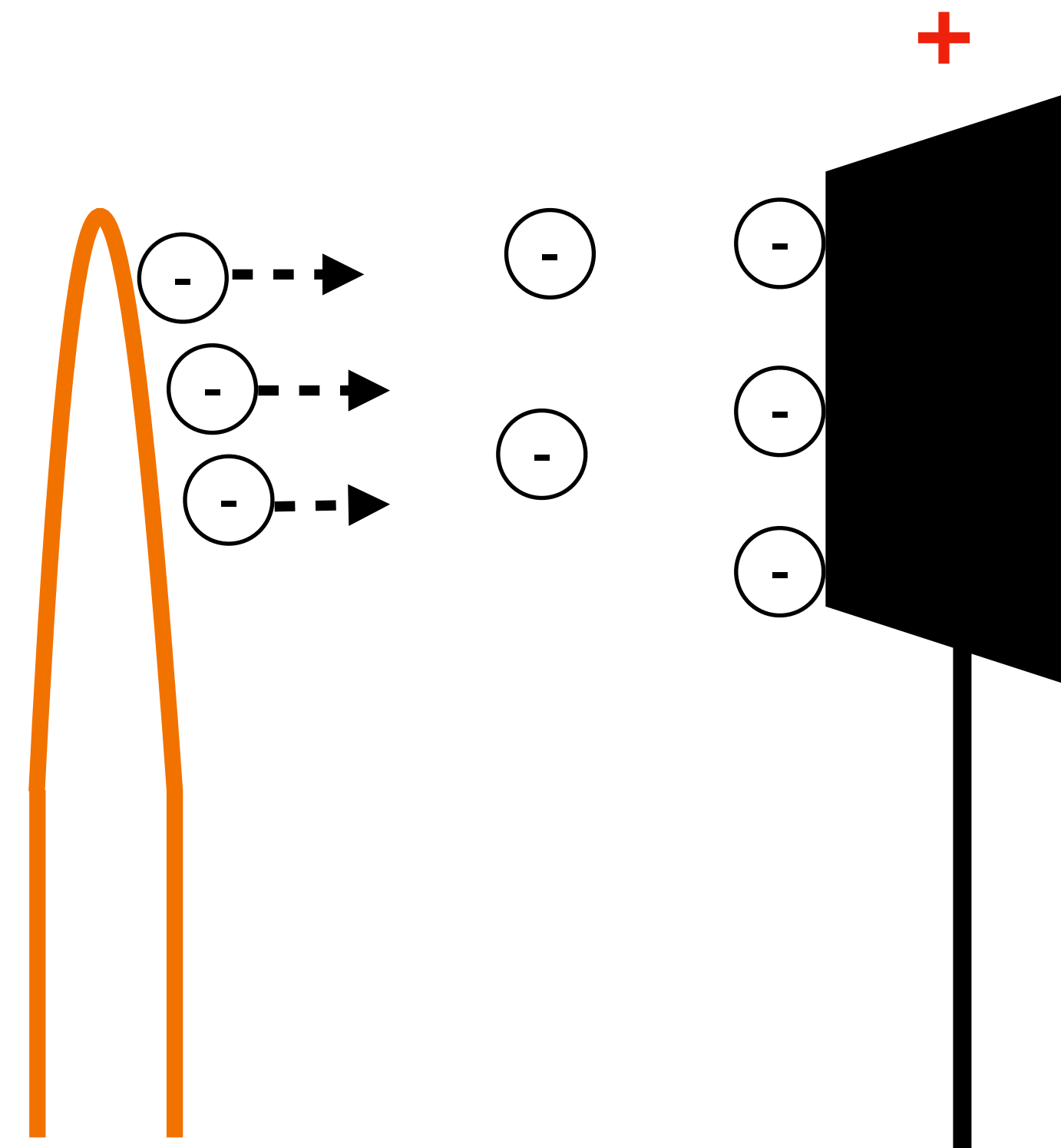
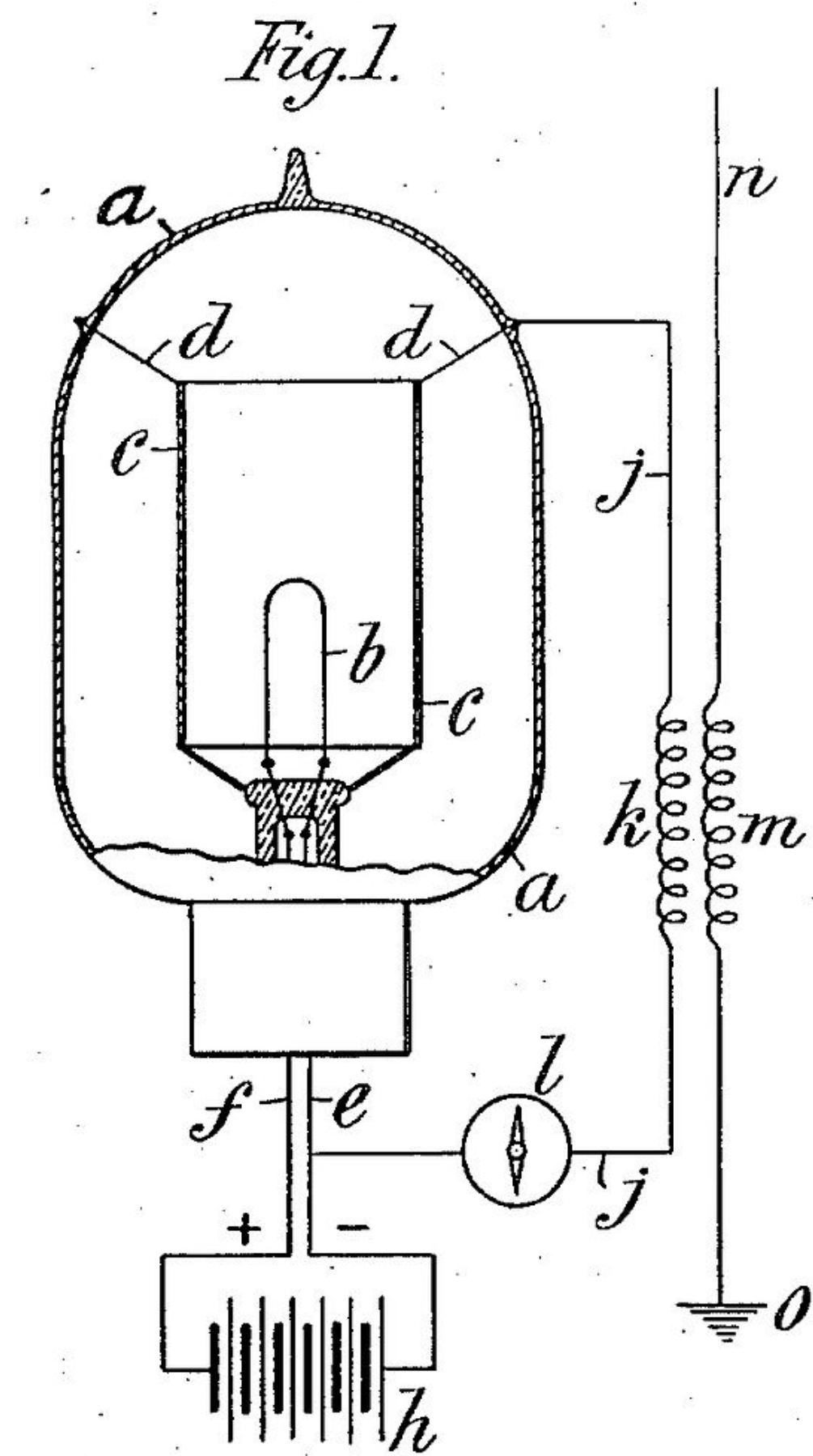
Vacuum Tube

Fleming Valve, 1904



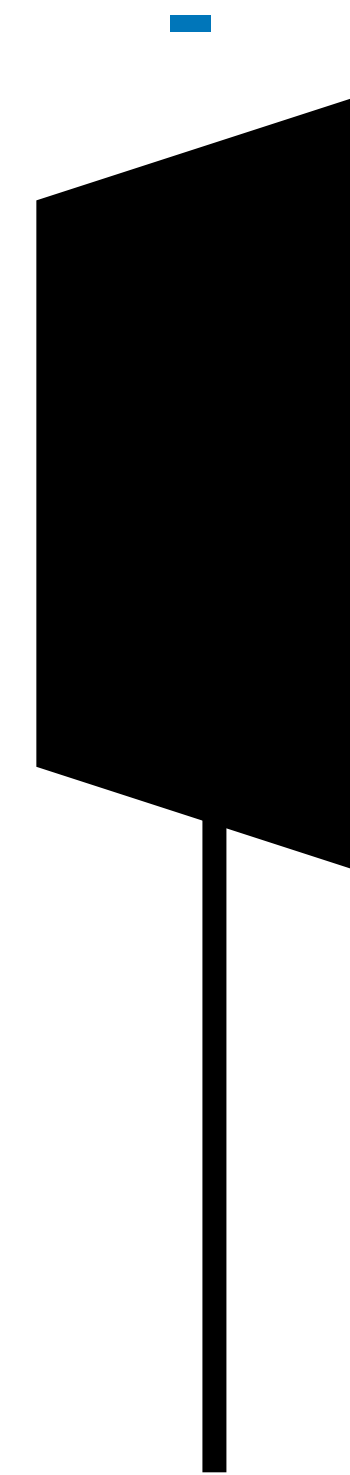
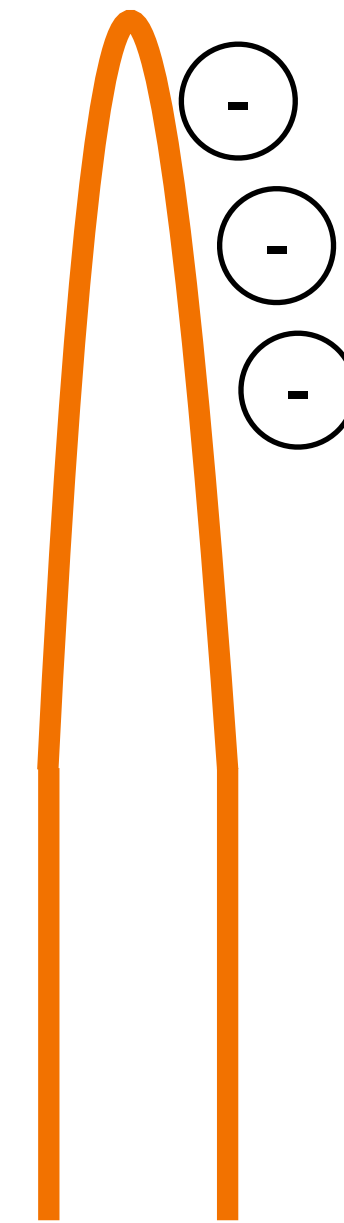
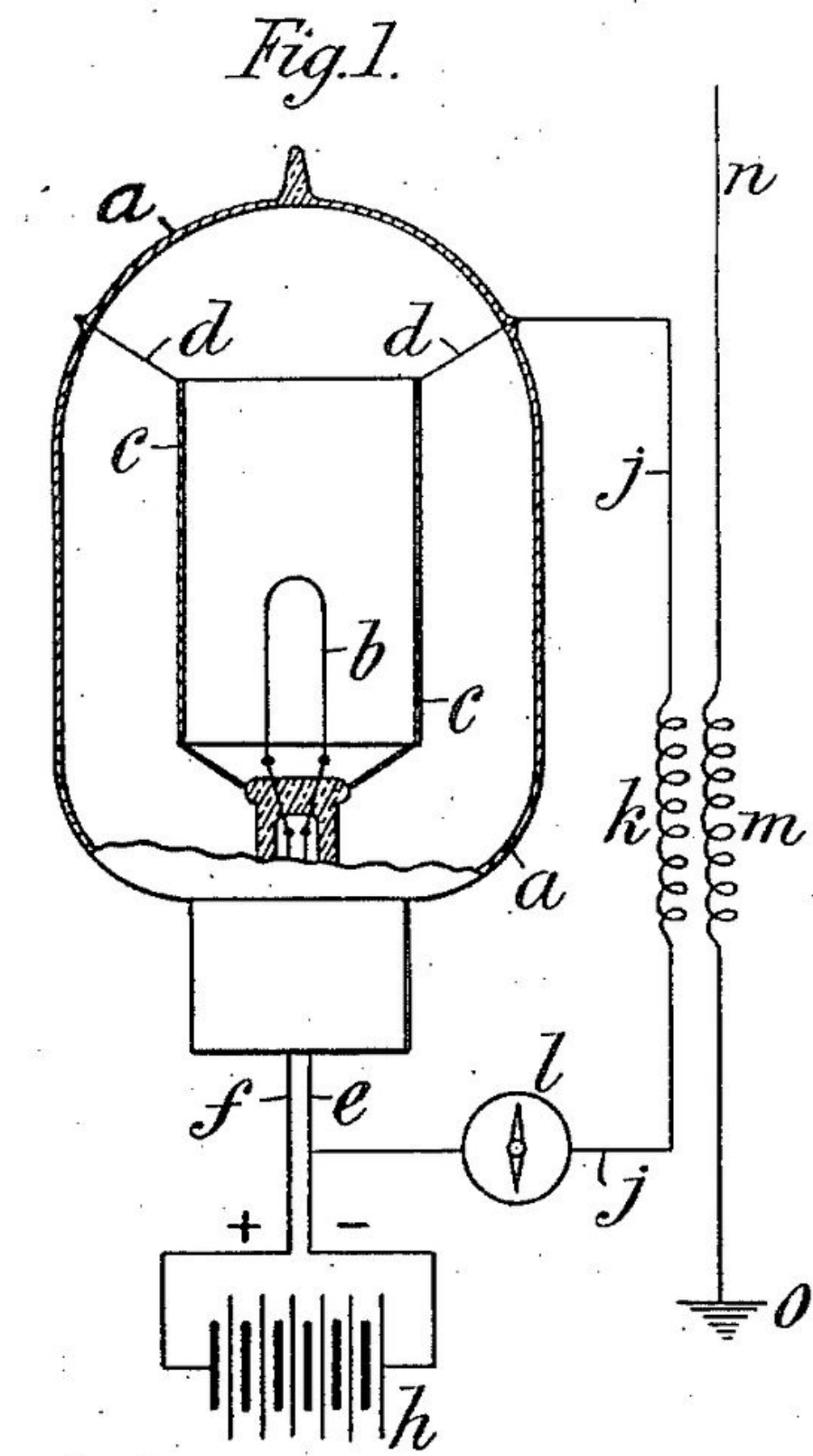
Vacuum Tube

Fleming Valve, 1904



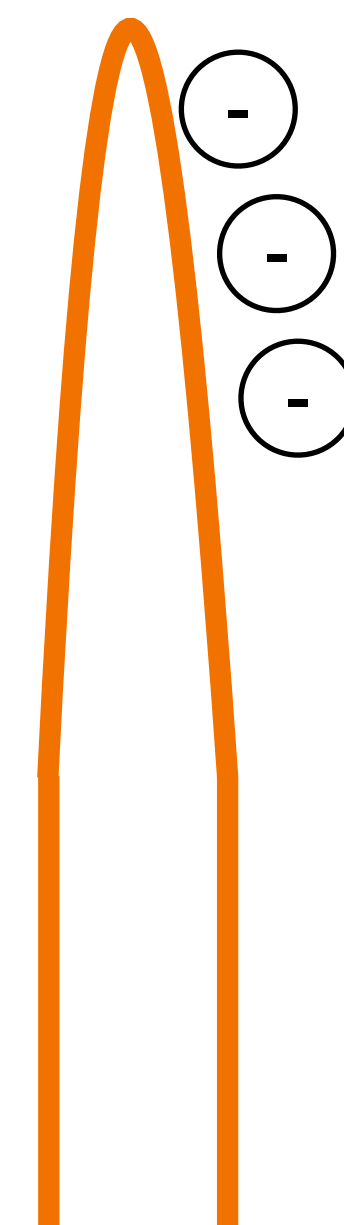
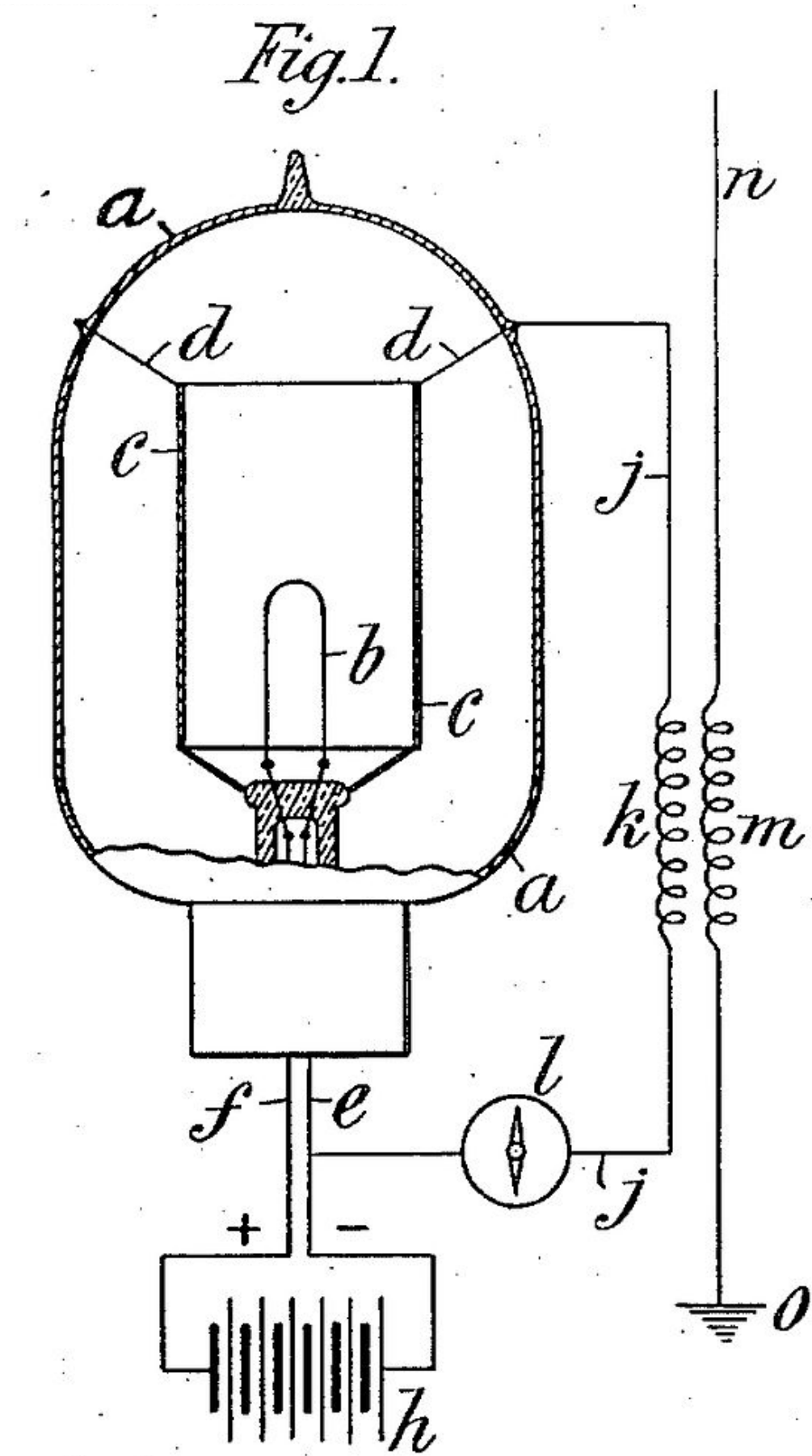
Vacuum Tube

Fleming Valve, 1904



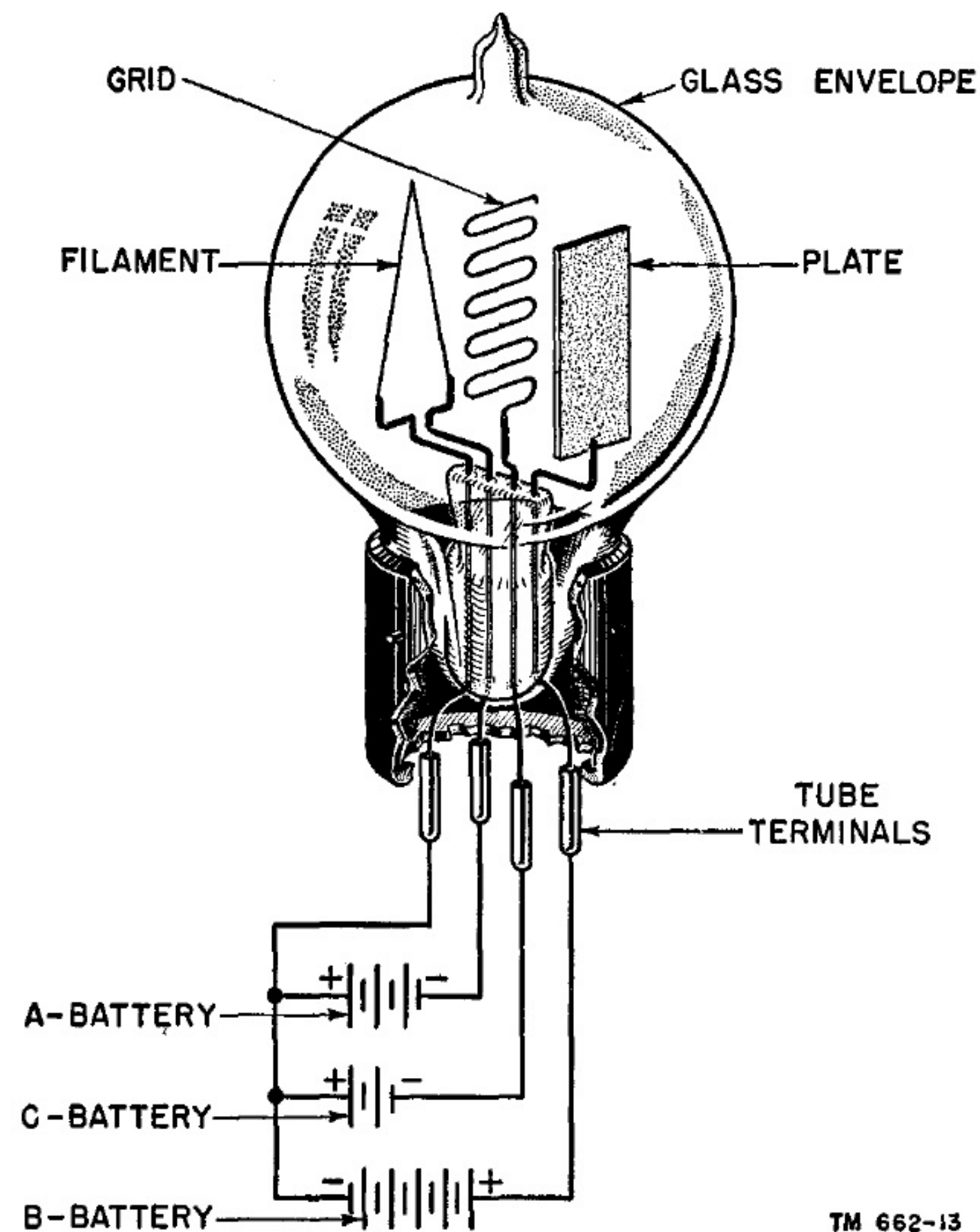
Vacuum Tube

Fleming Valve, 1904



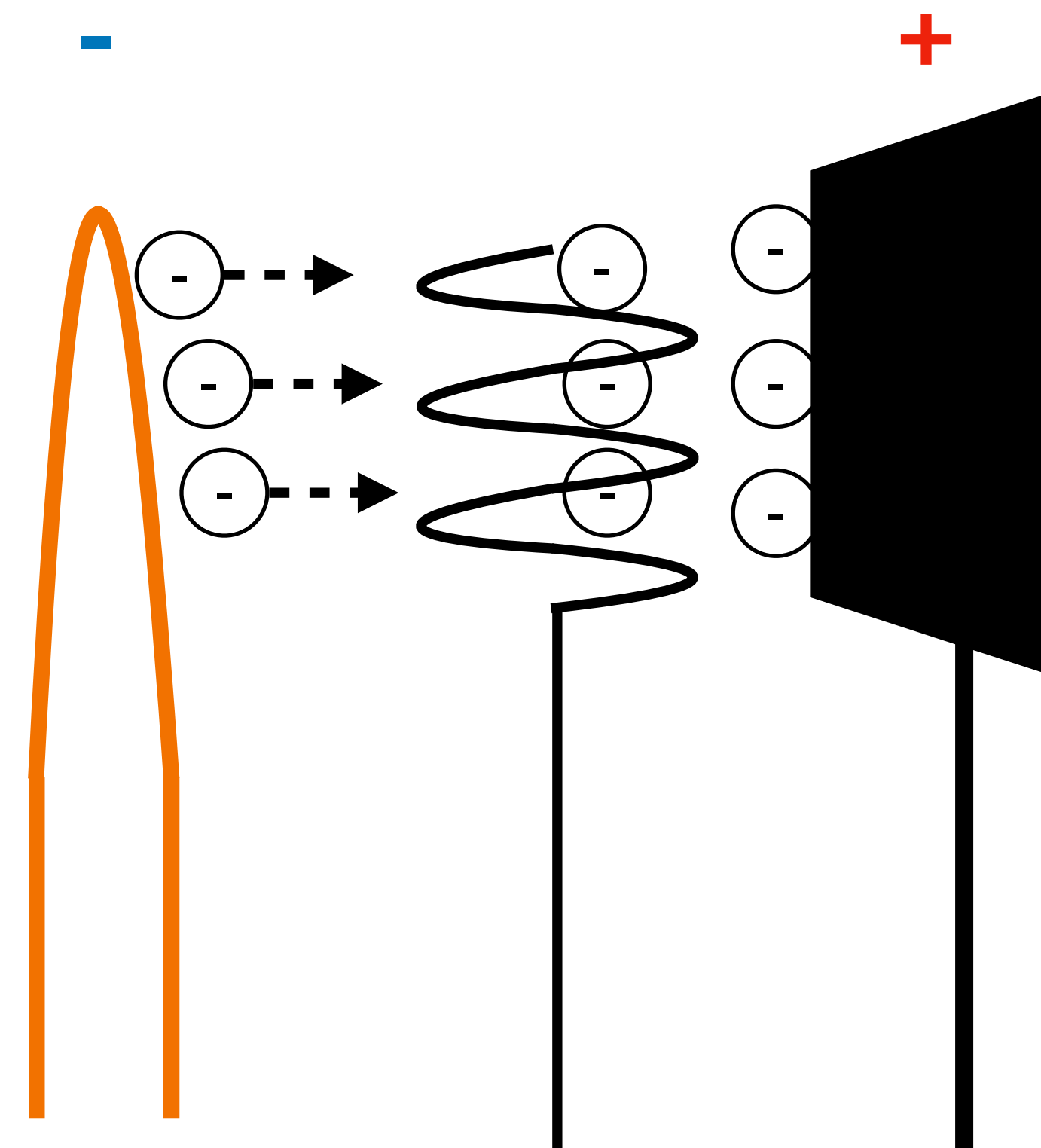
Vacuum Tube

Lee de Forest's Triode, 1906



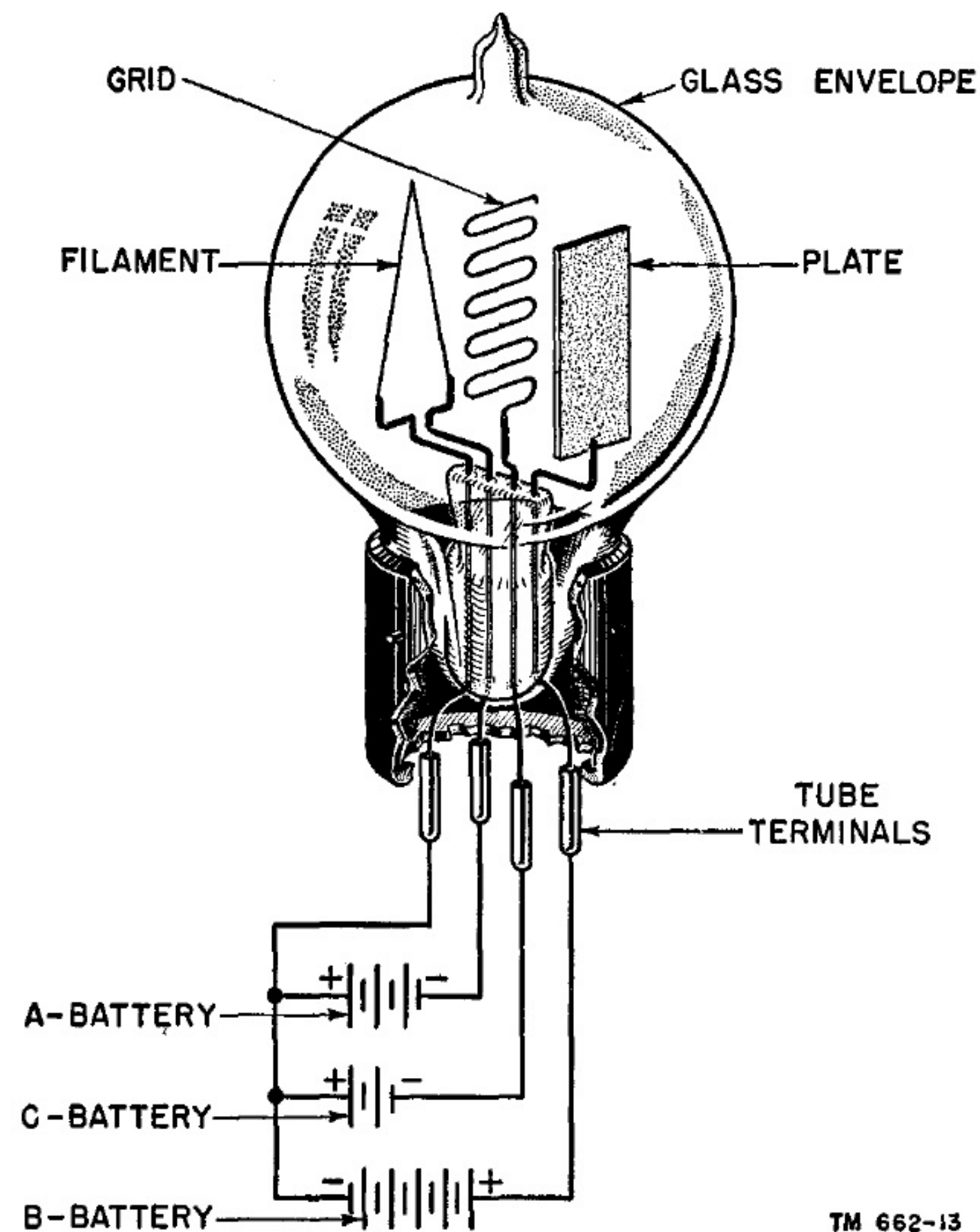
TM 662-13

Figure 4. Construction of DeForest's three-element tube, or triode.



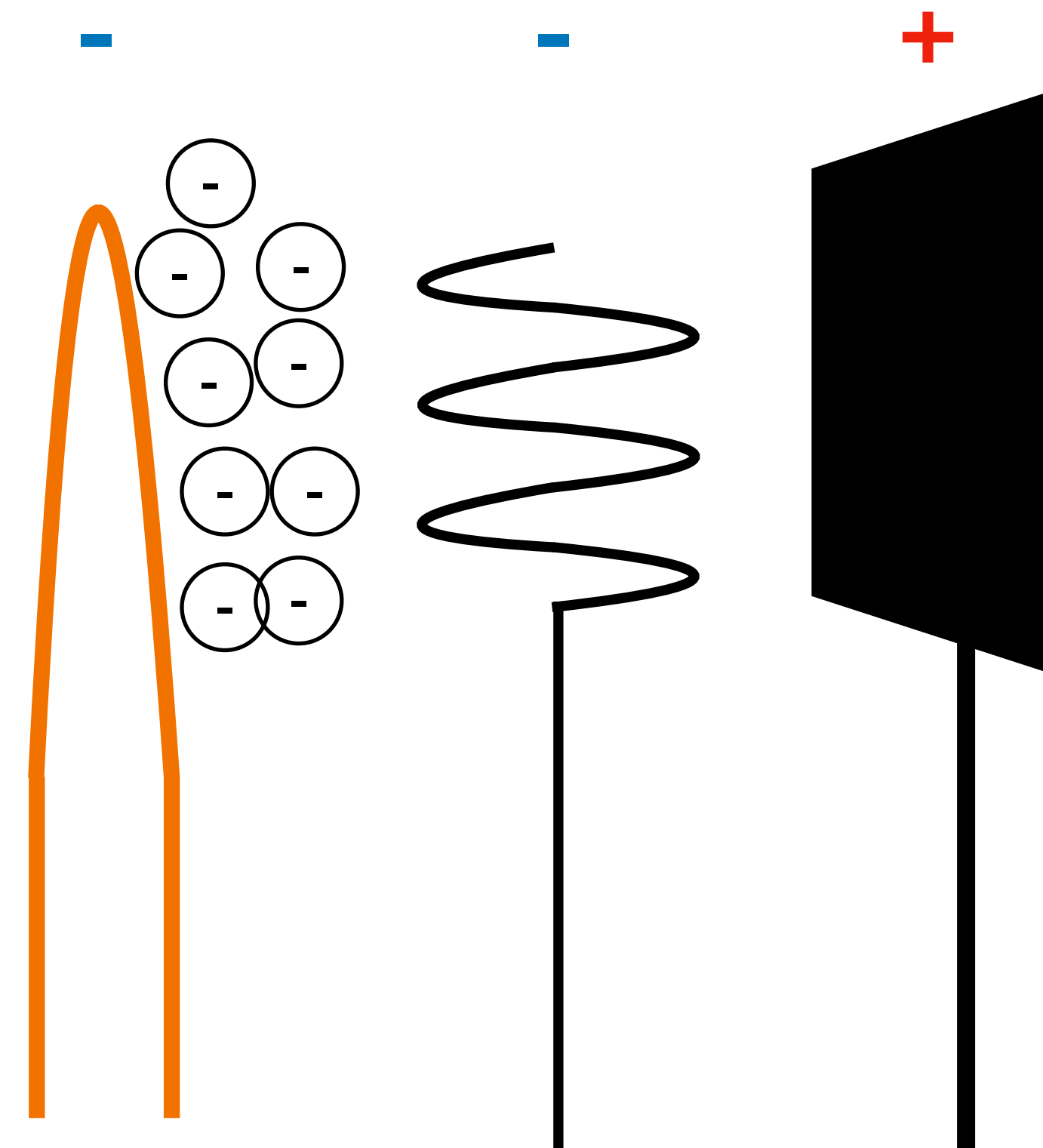
Vacuum Tube

Lee de Forest's Triode, 1906



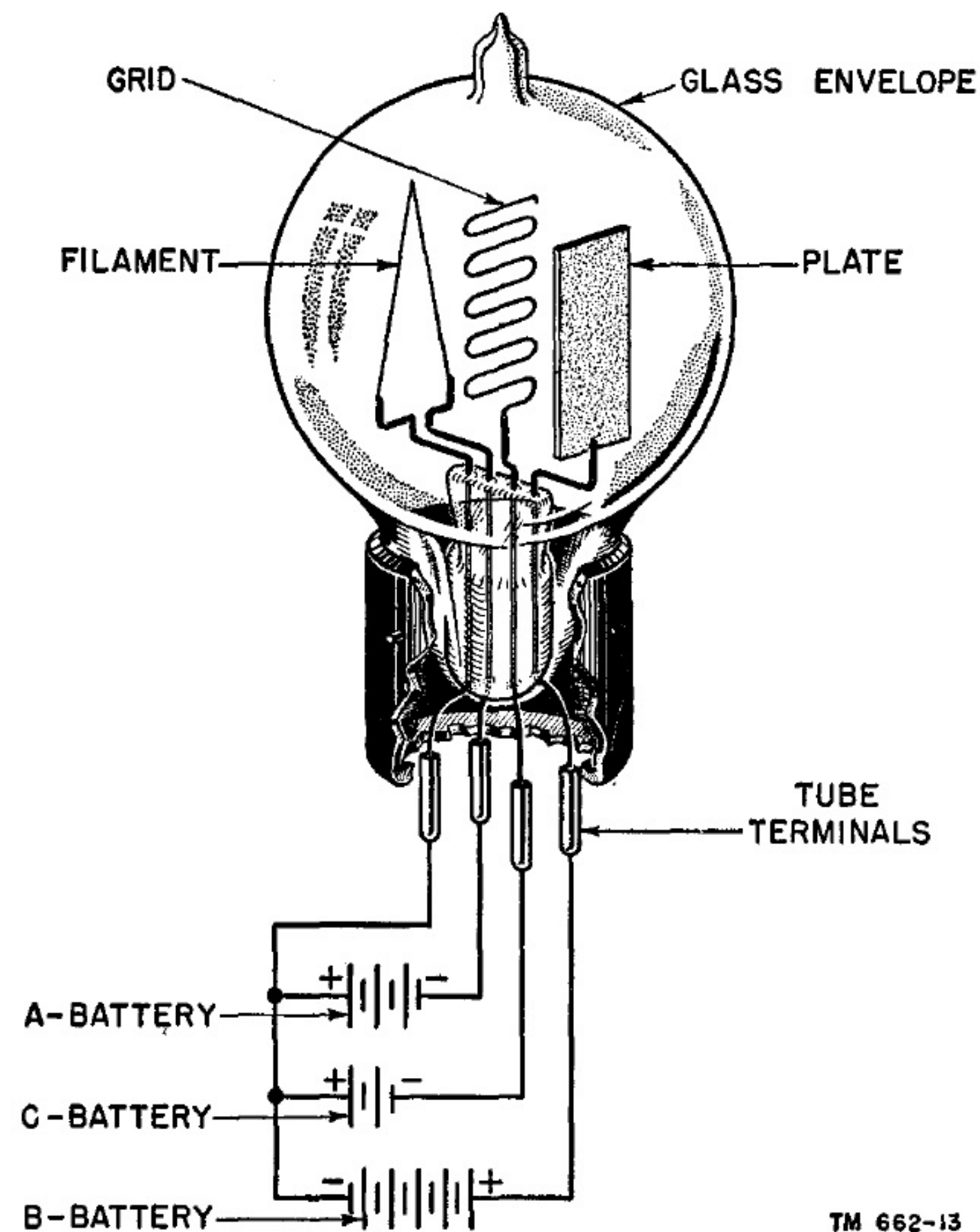
TM 662-13

Figure 4. Construction of DeForest's three-element tube, or triode.



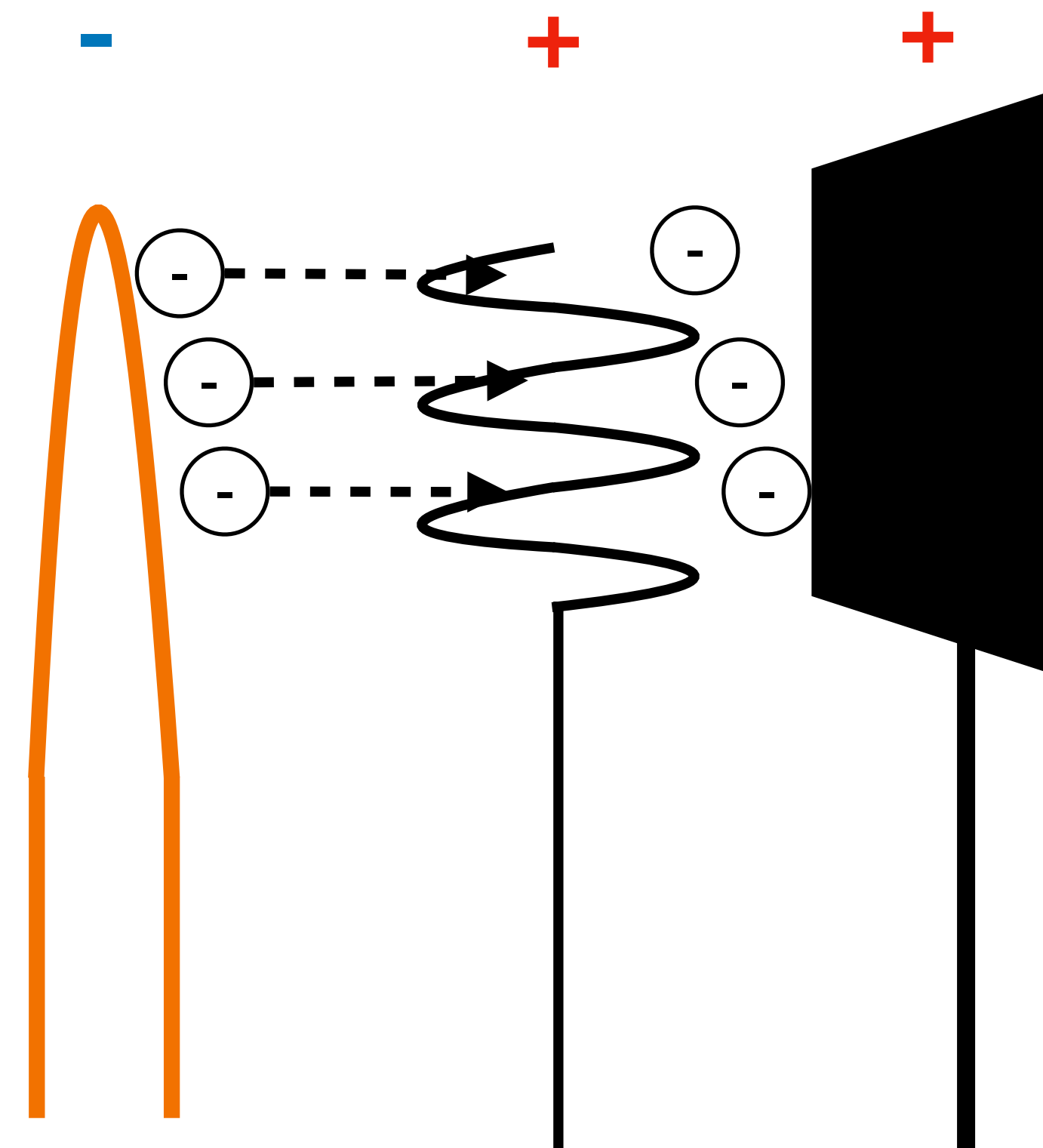
Vacuum Tube

Lee de Forest's Triode, 1906



TM 662-13

Figure 4. Construction of DeForest's three-element tube, or triode.



Vacuum Tube

Lee de Forest's Triode, 1906

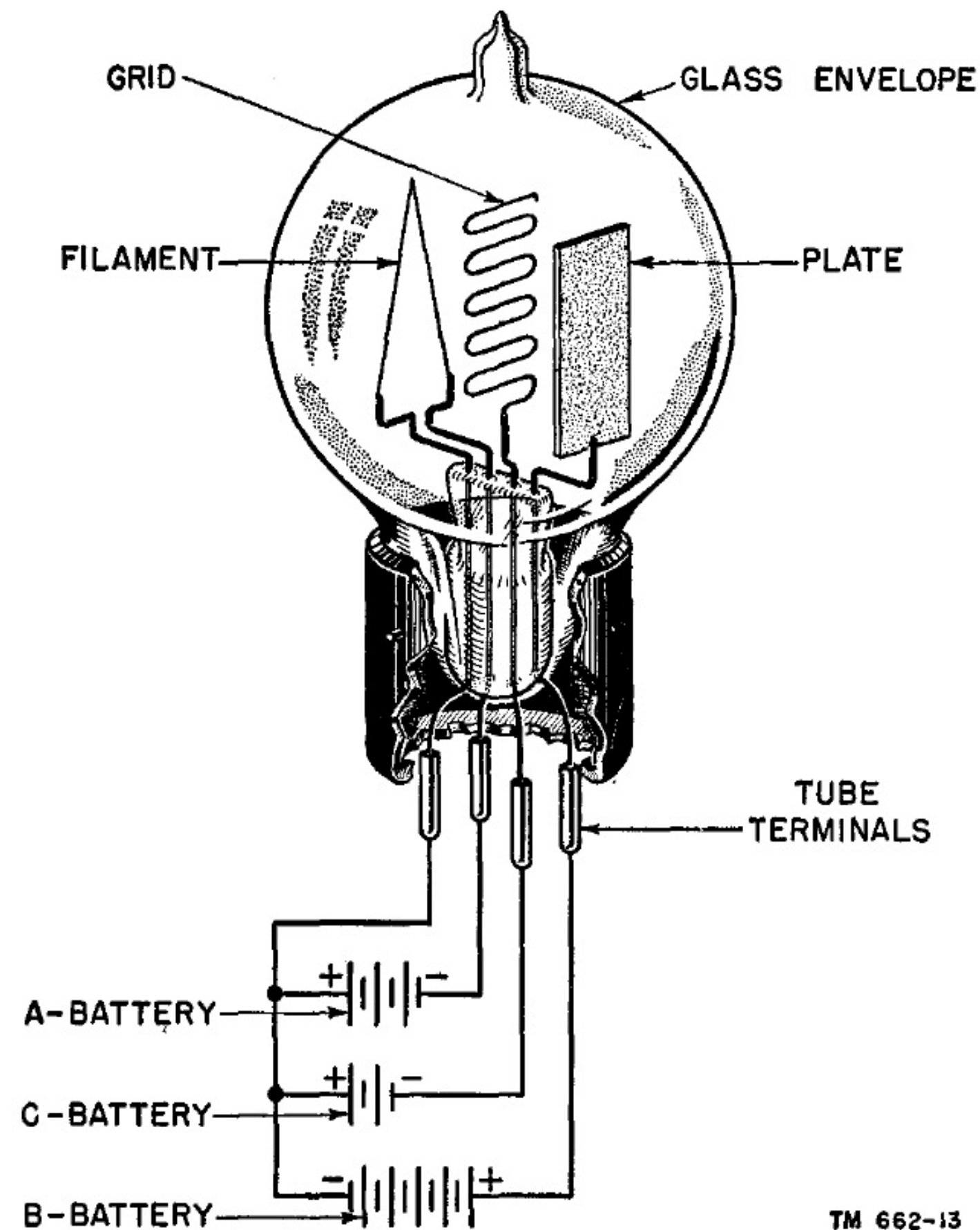
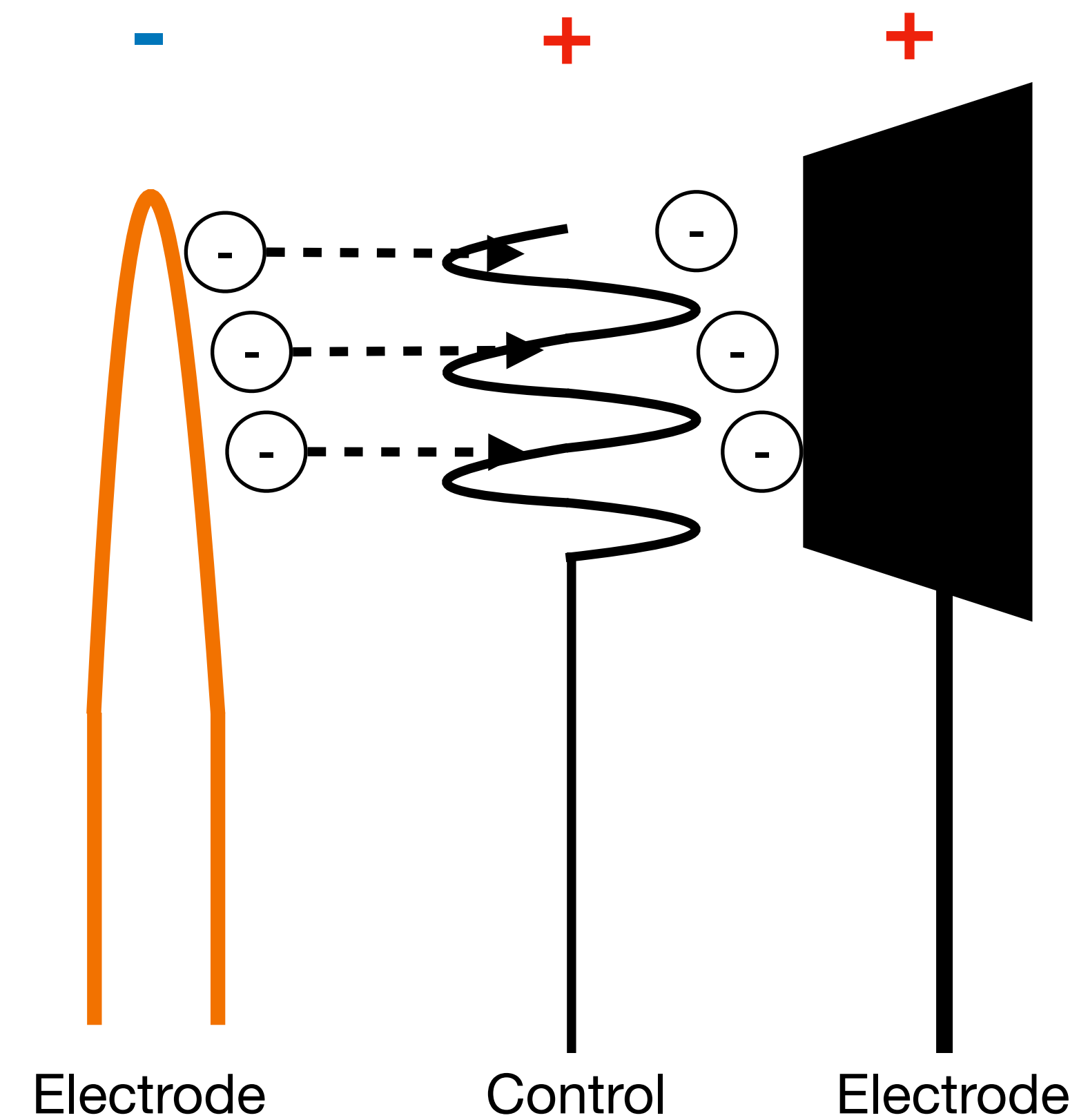


Figure 4. Construction of DeForest's three-element tube, or triode.



Vacuum Tube

- Just like a relay, but *no moving parts*
- Response is rapid -- kHz level
- Feasible for use in computers by 1940s
- Purely electronic

For those who want the best

EASIEST TO DRIVE!

6 OUTSTANDING FEATURES INSURE LONGER LIFE

- Short, direct connections to both sides of plate.
- Mica snubbers support elements in dome of bulb — increase strength — absorb shock.
- Lavo low-loss insulation used exclusively.
- Heavy, non-warping, over-size SPEER graphite anode.
- Filament heat radiators to reduce stem temperature.
- Dual grid leads to halve grid current in each wire, further reducing stem heating and subsequent glass electrolysis.

**HY40
HY40Z
\$3.50 NET**

Characteristics of HY40 & HY40Z

Plate dissipation	40 max. watts
Plate input	1000 max. watts and 115 max. ma.
Class C output at max. input (75% eff.)	86 watts

Above ratings, like all other Hytron, are for continuous-duty operation. The HY51 series having the above same outstanding features carries the following continuous-duty ratings.

Plate dissipation	65 max. watts
Plate input	1000 max. volts and 175 max. ma.
Class C output at max. input (75% eff.)	131 watts

HY51A	HY51B	HY51Z	\$4.50 net
-------	-------	-------	------------

All Hytron transmitting tubes are economically priced and have been designed for extremely efficient operation with low-cost radio parts. Thus the amateur makes a dual saving. Further advantage is that Hytron tubes require no voltages in excess of 1000 for full output.

Hytron transmitting and special purpose tubes are backed by more than 18 years continuous experience in the exclusive manufacture of radio tubes. The Hytron Corporation, the originator of the now popular *Bantam** "GT" series tubes as well as the pioneer of the ultra-small low-drain *Bantam Junior** pentodes for wearable hearing aids.

*Trade-mark registered.

Hytron tubes available at leading distributors.

HYTRONIC LABS.
25 New Derby St., Salem, Mass.

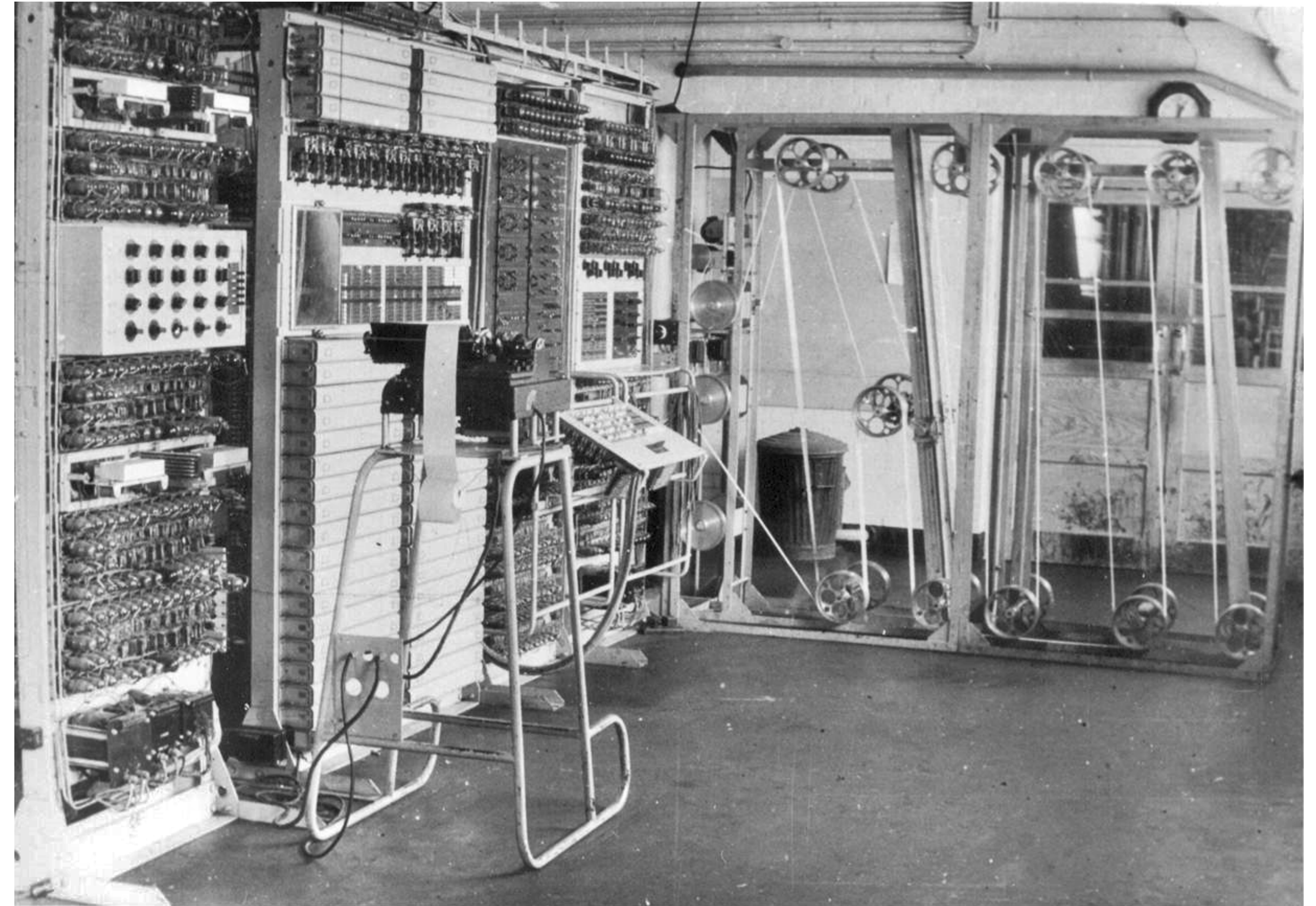
HYTRON
A DIVISION OF THE
HYTRON CORP.

Manufacturers of Radio Tubes Since 1921

Colossus

Bletchley Park, 1944

- Built by Tommy Flowers in 1944
- Used to decrypt German encrypted messages
- Used 1,600 tubes

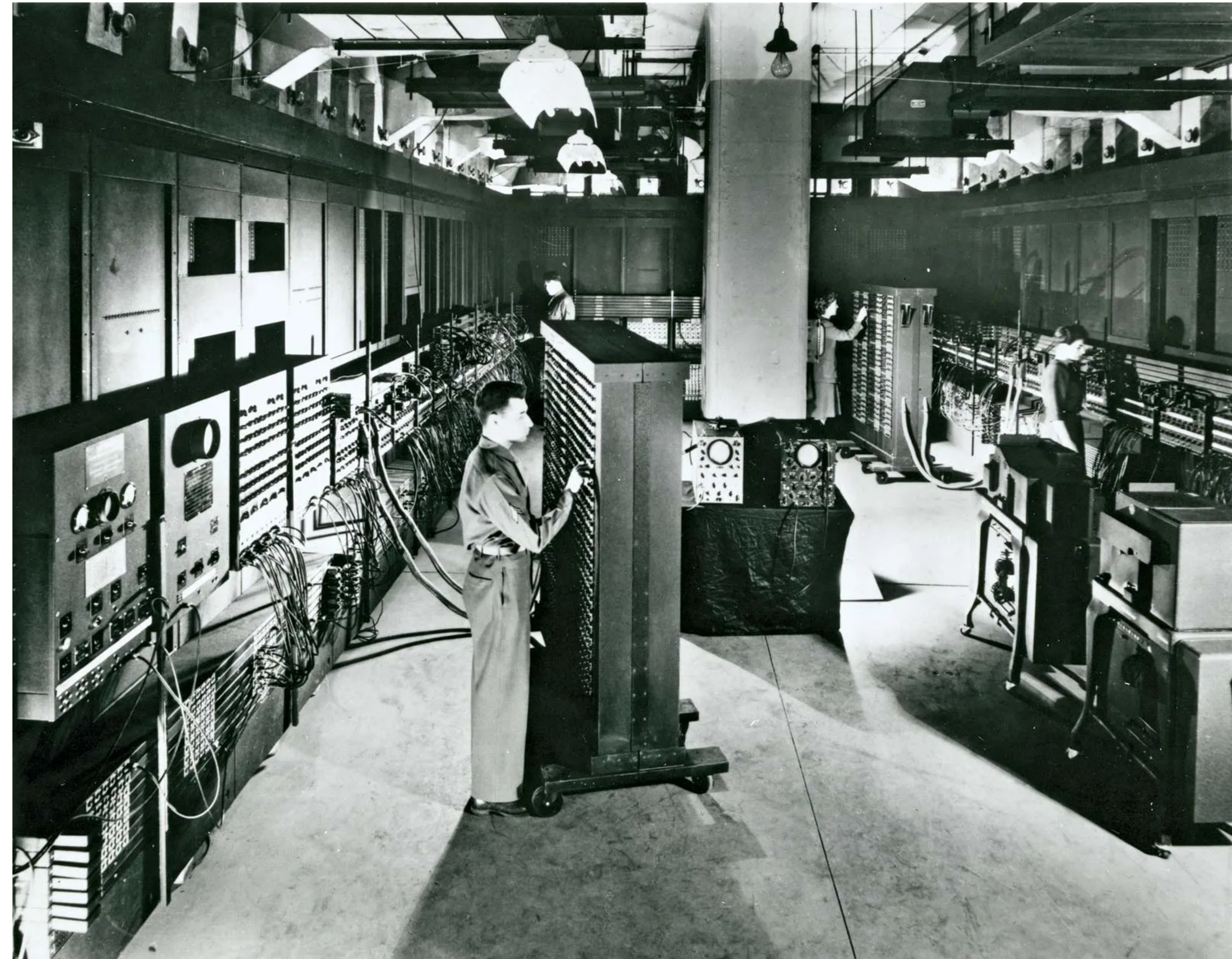


Colossus 10

ENIAC

University of Pennsylvania, 1945

- **E**lectronic **N**umerical **I**ntegrator and **C**omputer
- 18,000 vacuum tubes
- 5,000 additions/subtractions per second
- 357 multiplications per second
- Operational until 1955



Vacuum Tube

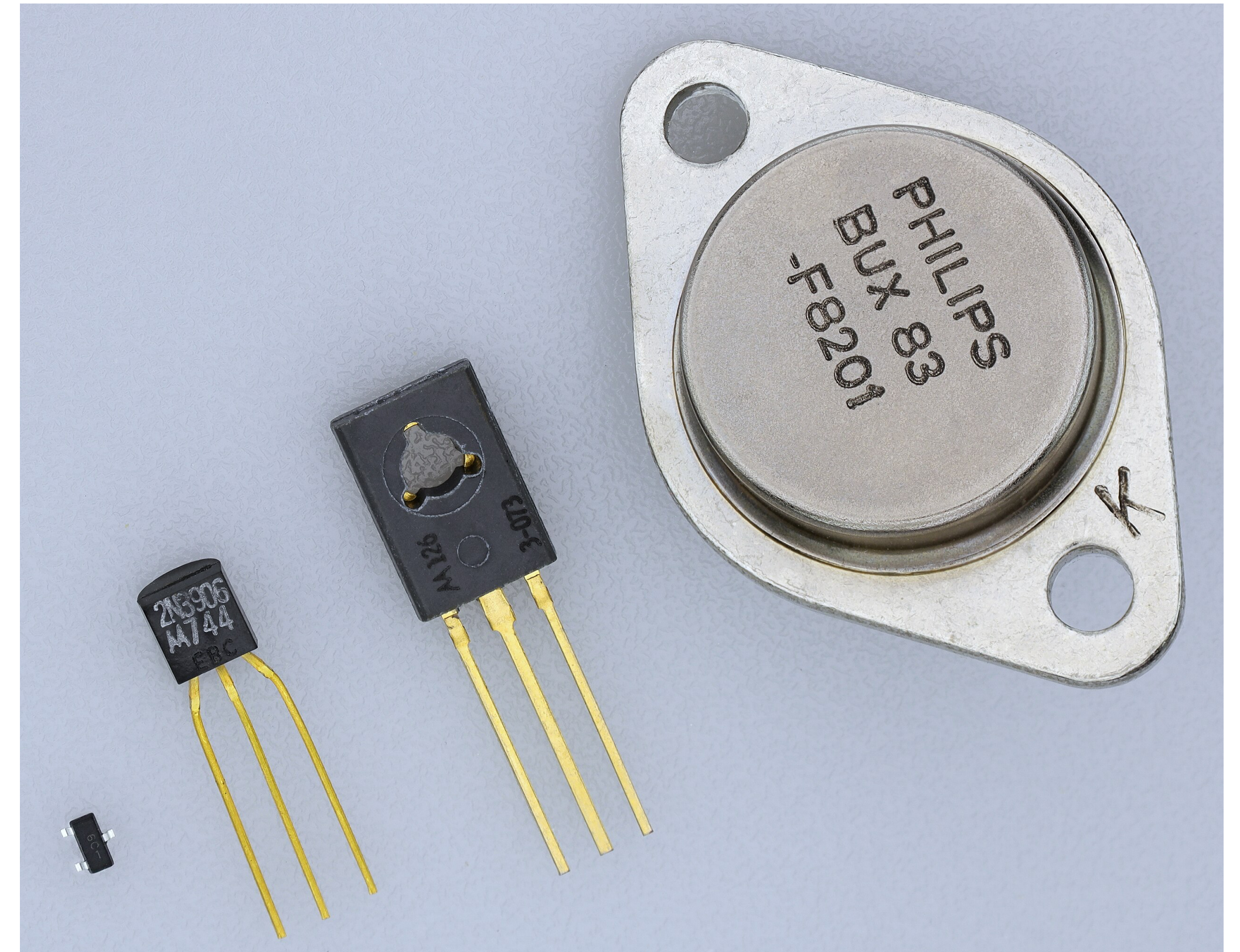
Limitations

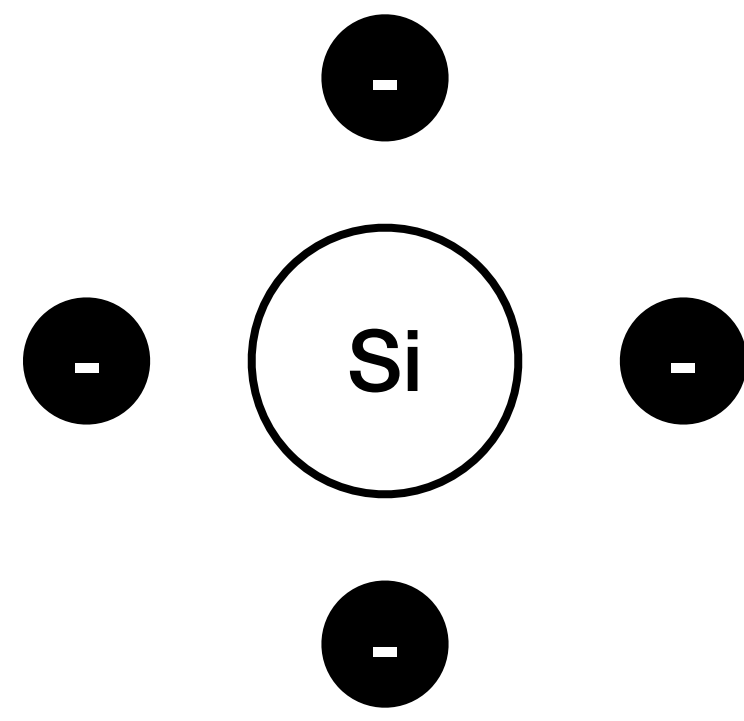
- Bulky, high voltage, inefficient
- Failure rate was high -- ENIAC was usually only operational for half a day
- *Transistors*

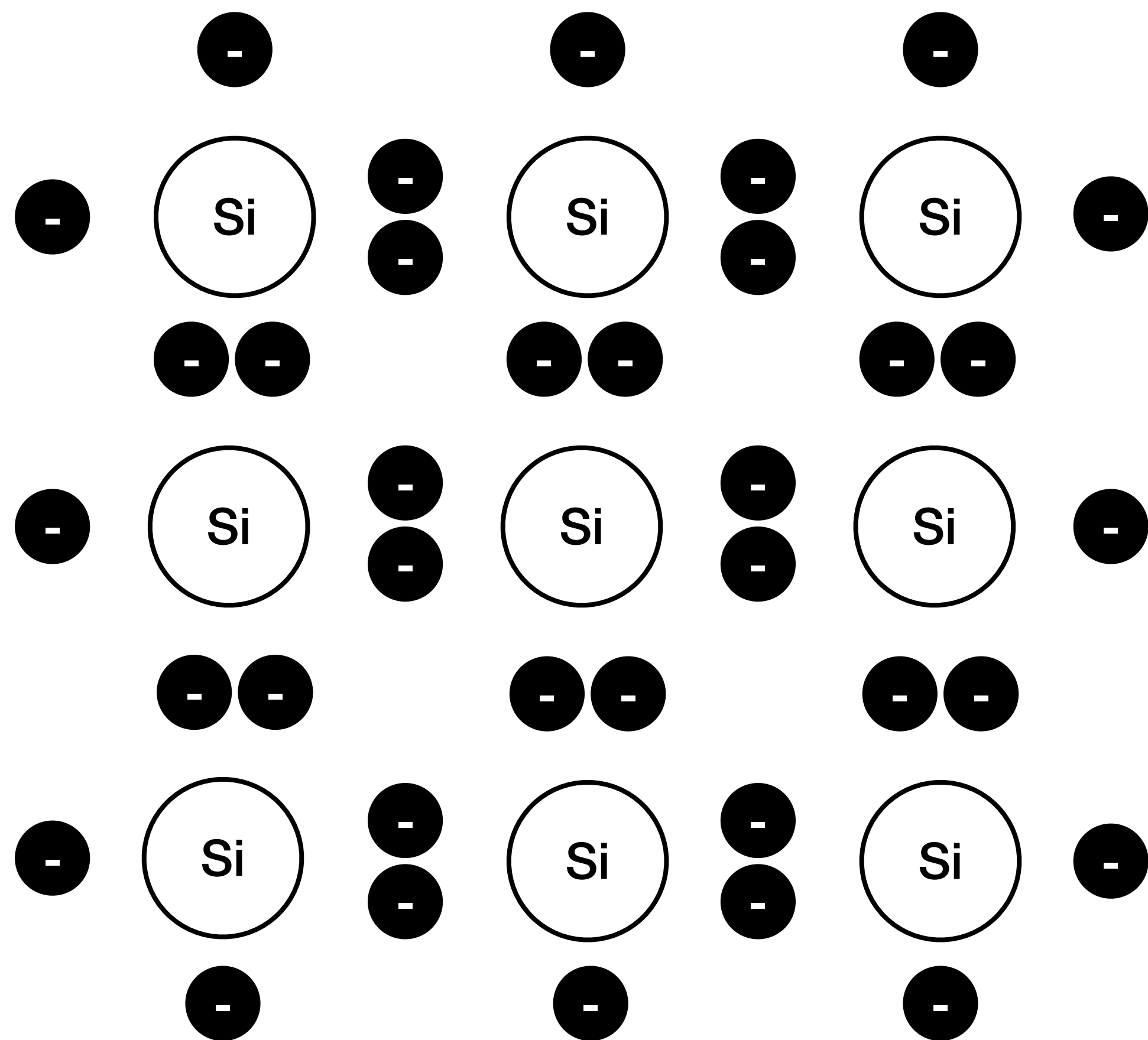
Transistors

Bell Lab, 1947

- John Bardeen, Walter Brattain, William Shockley
- Prerequisites:
 - Protons -- positive charge
 - Electrons -- negative charge
 - Opposites attract, likes repel
 - A flow of electrons makes an electric current



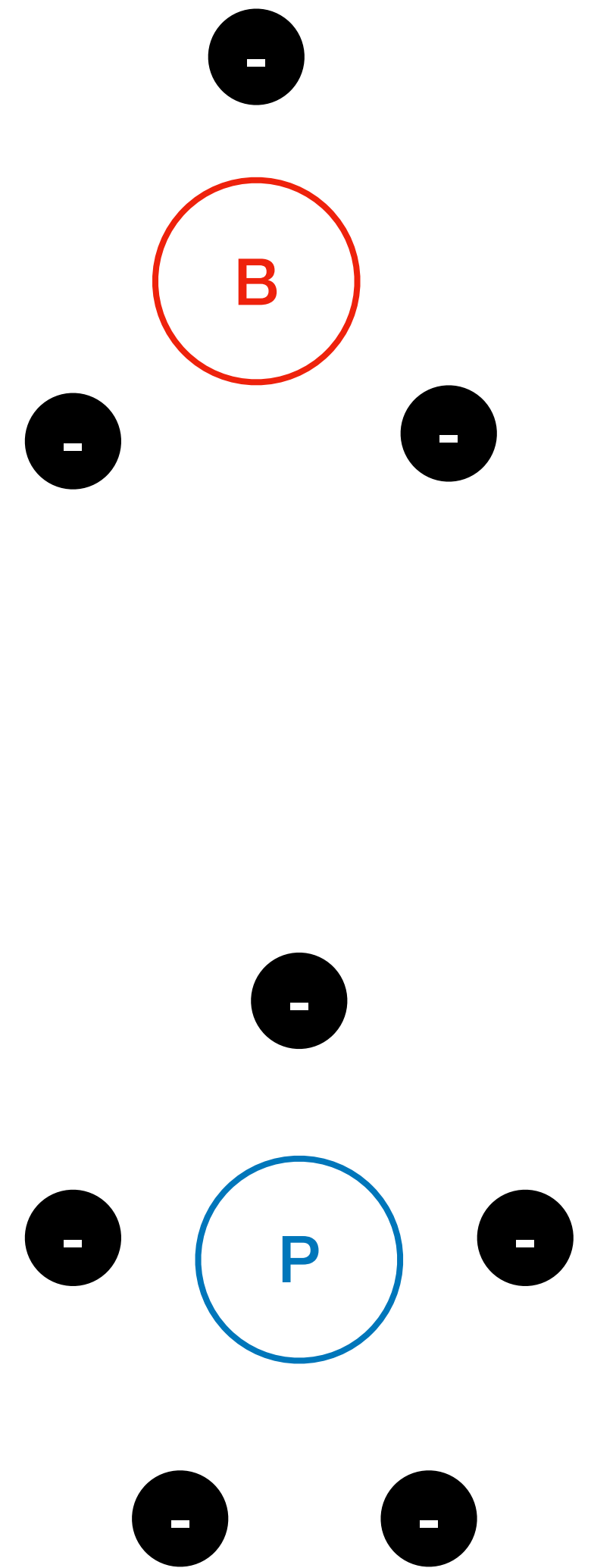
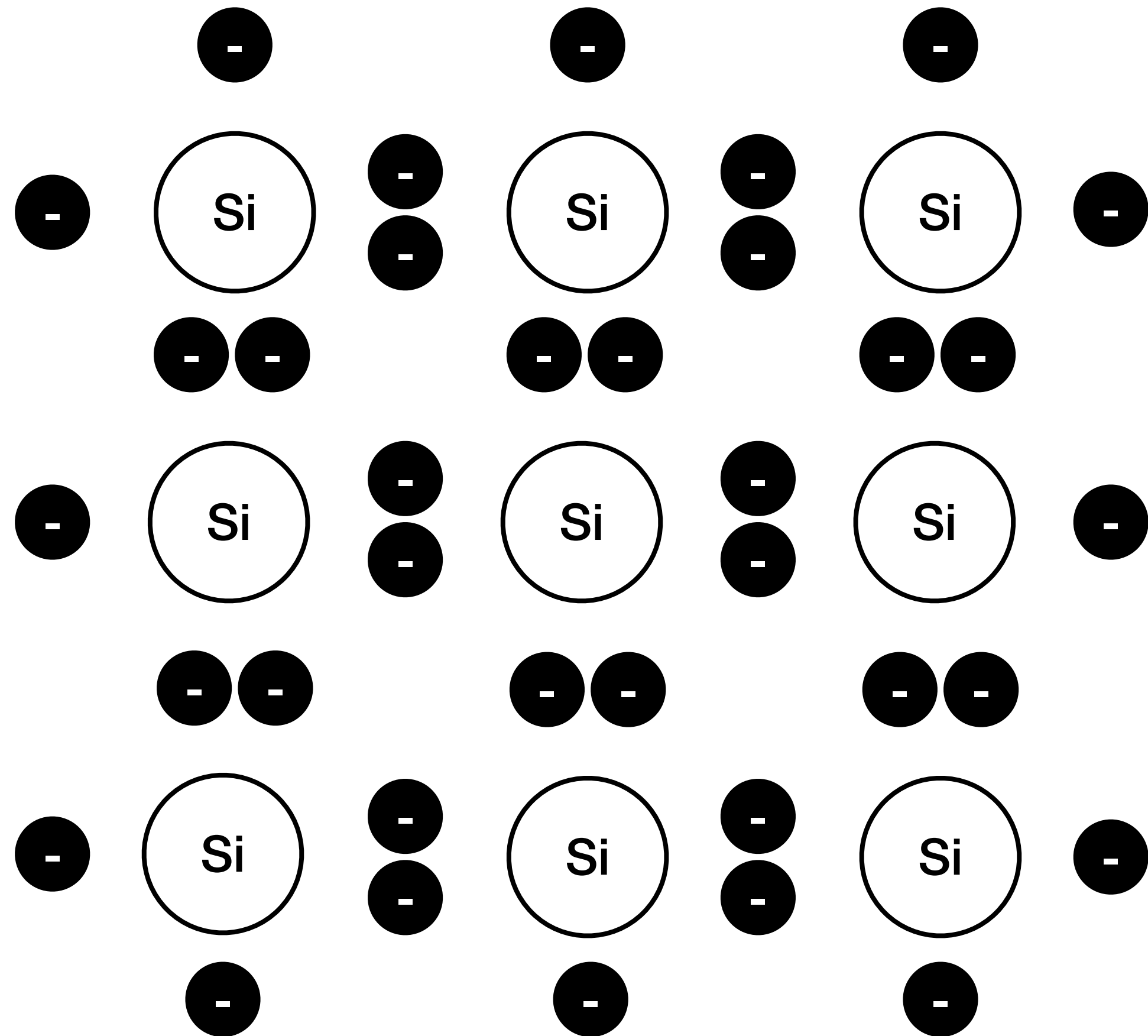




* Actually, they form a tetrahedron.

13 14 15

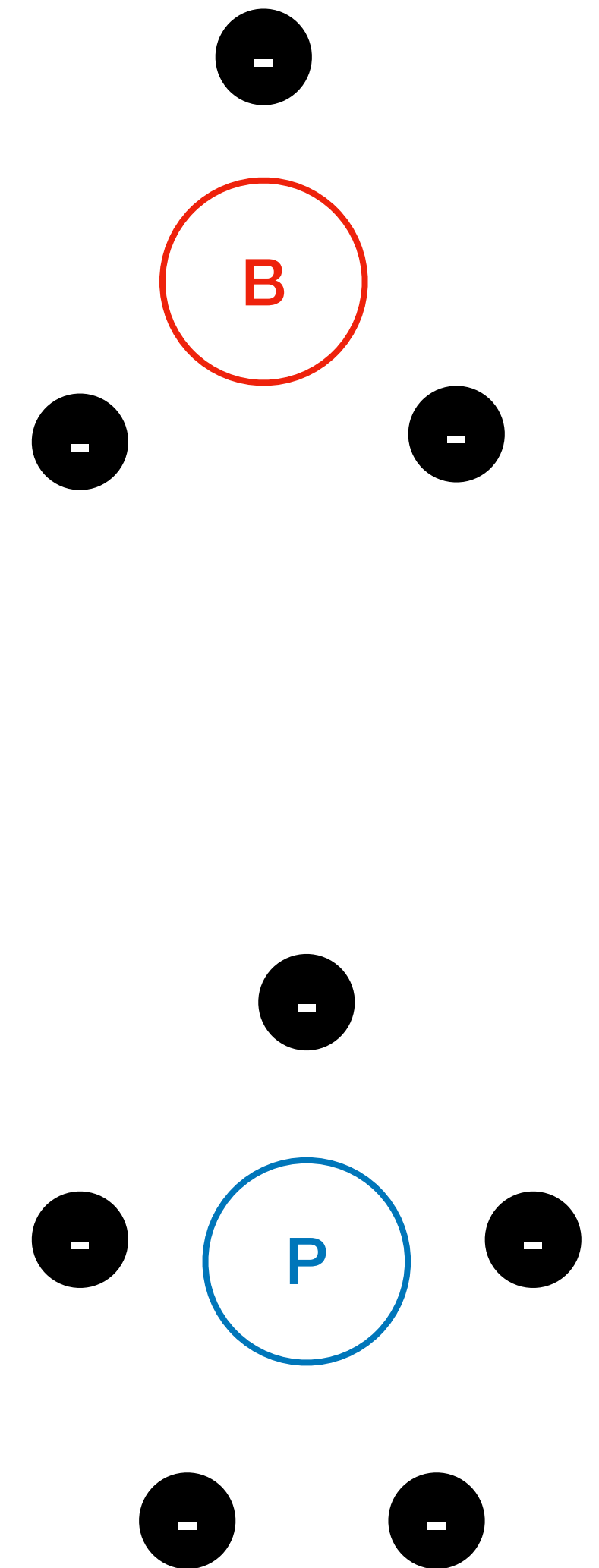
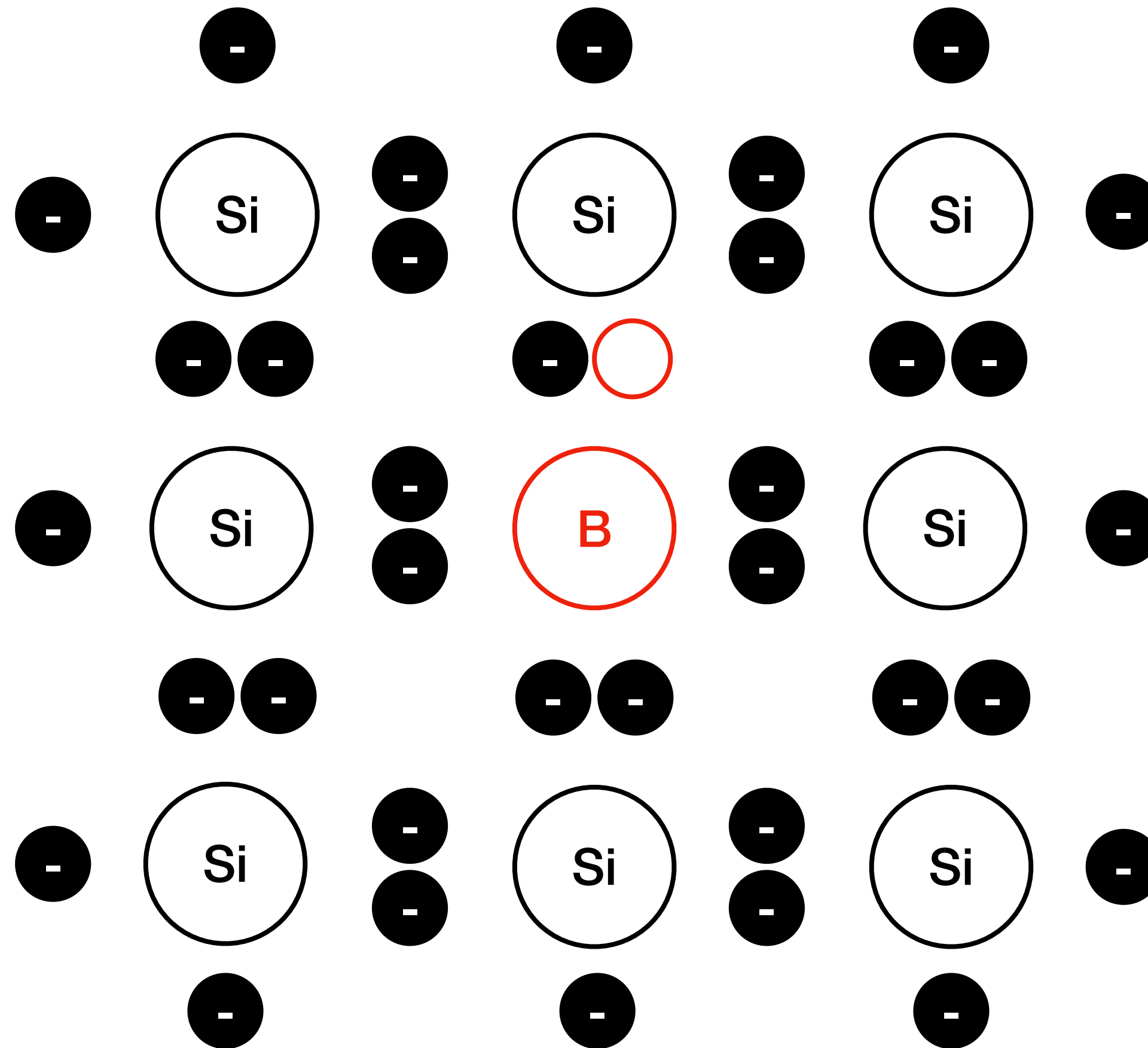
5 B	6 C	7 N
13 Al	14 Si	15 P
31 Ga	32 Ge	33 As
49 In	50 Sn	51 Sb
81 Tl	82 Pb	83 Bi
113 Nh	114 Fl	115 Mc
67 Ho	68 Er	69 Tm
99 Es	100 Fm	101 Md



* Actually, they form a tetrahedron.

13 14 15

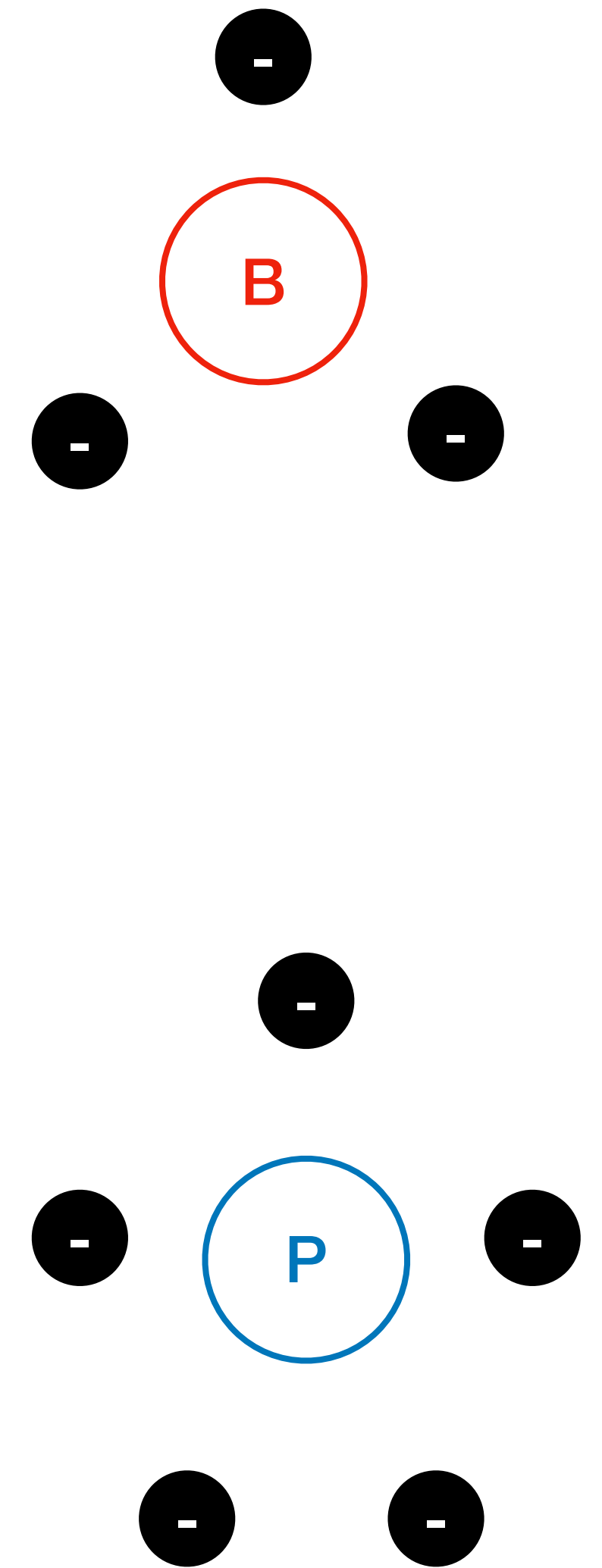
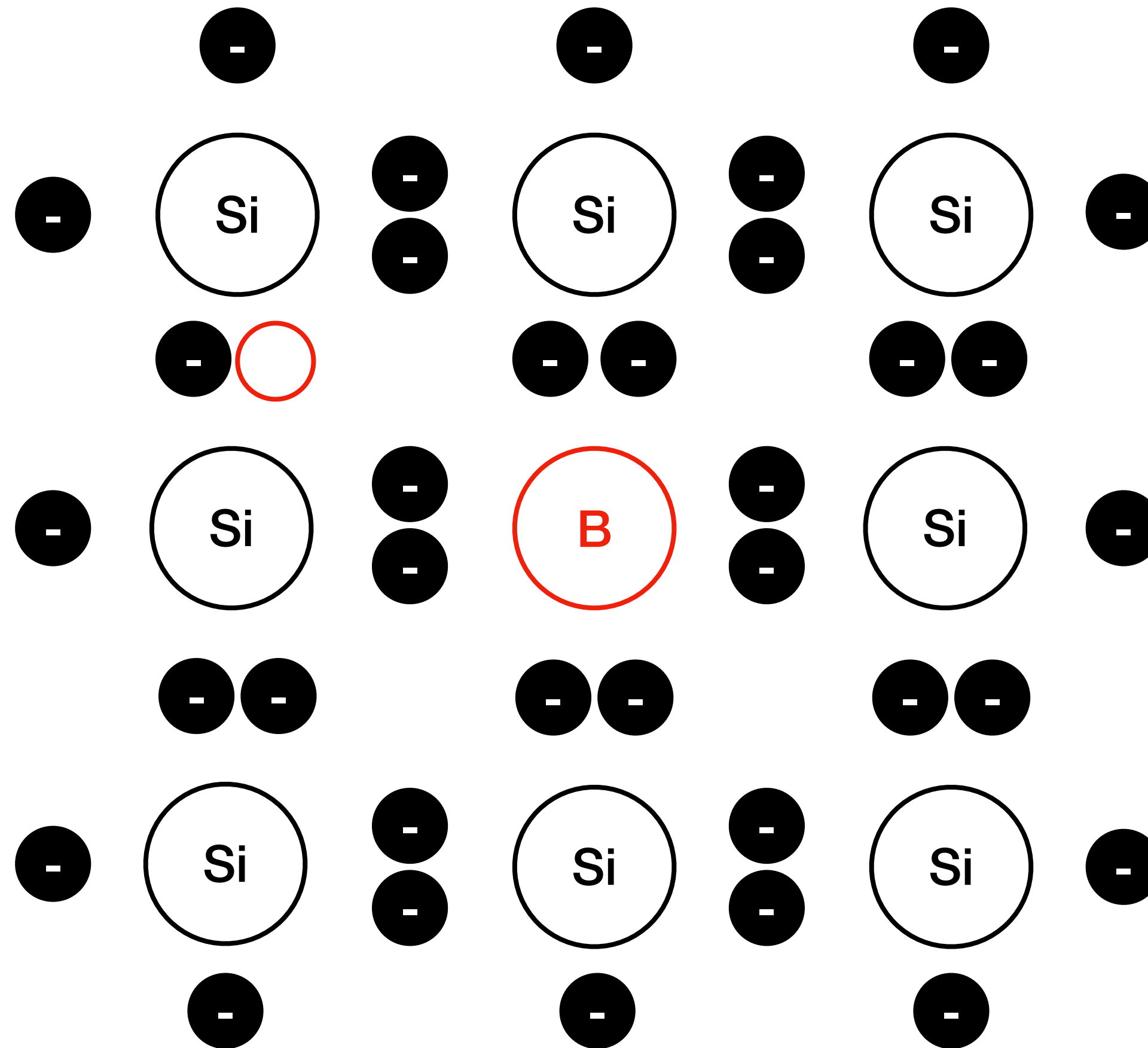
5 B	6 C	7 N
13 Al	14 Si	15 P
31 Ga	32 Ge	33 As
49 In	50 Sn	51 Sb
81 Tl	82 Pb	83 Bi
113 Nh	114 Fl	115 Mc
67 Ho	68 Er	69 Tm
99 Es	100 Fm	101 Md



* Actually, they form a tetrahedron.

13 14 15

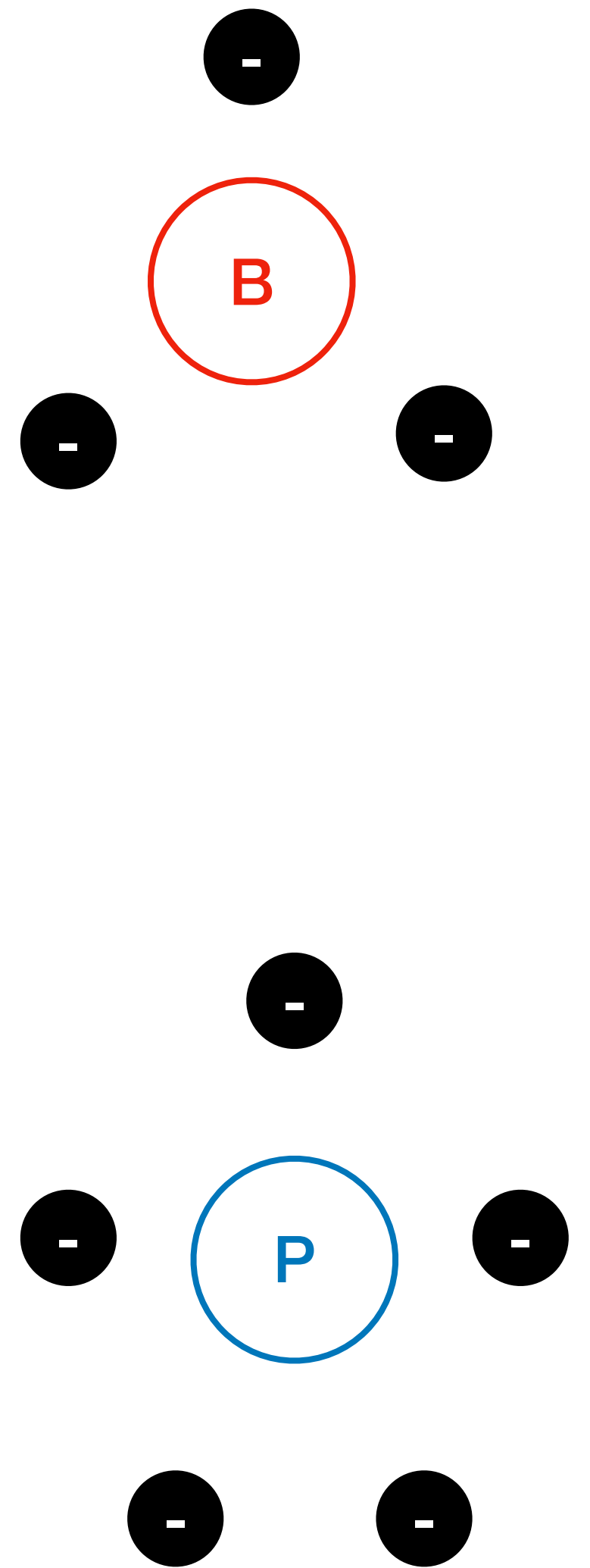
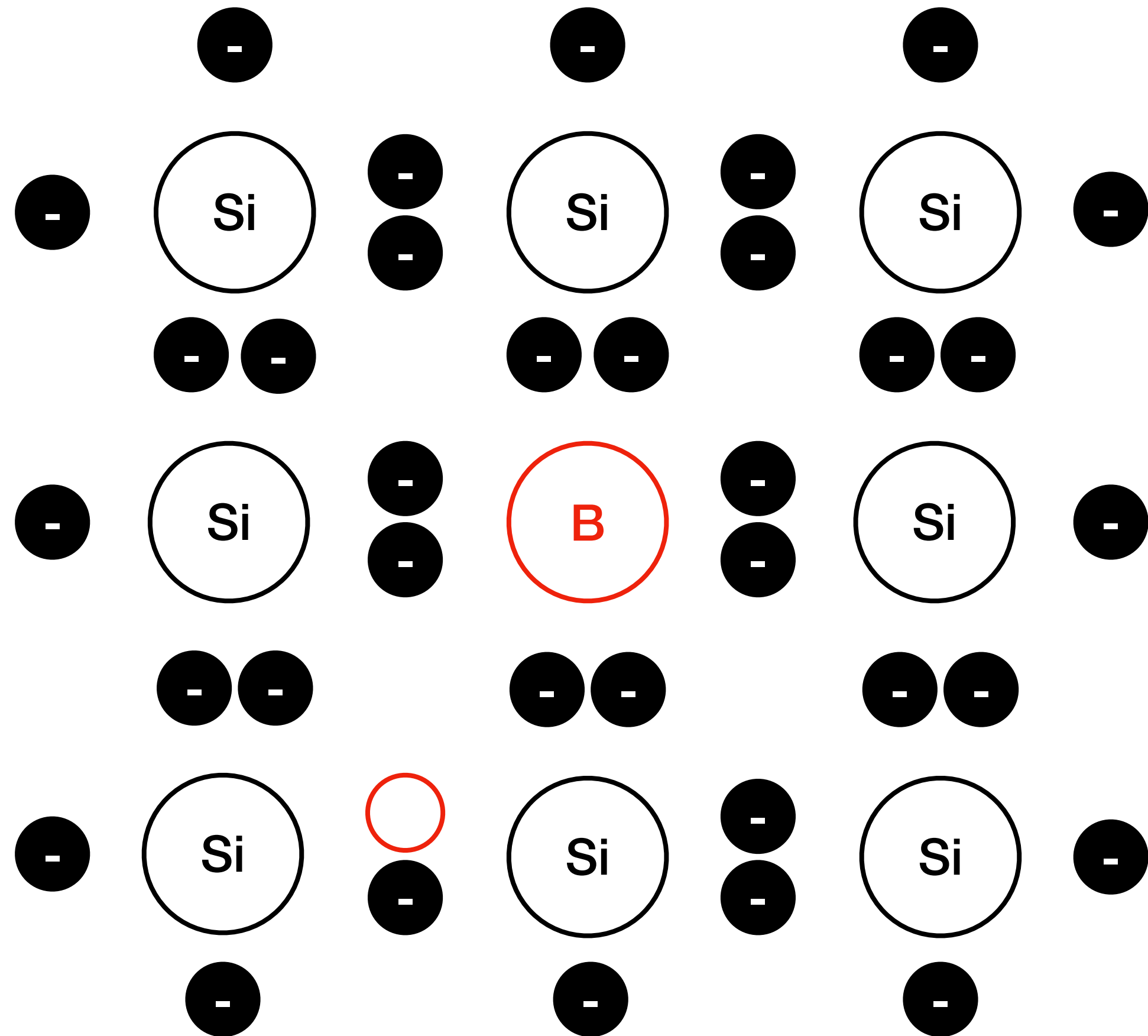
5 B	6 C	7 N
13 Al	14 Si	15 P
31 Ga	32 Ge	33 As
49 In	50 Sn	51 Sb
81 Tl	82 Pb	83 Bi
113 Nh	114 Fl	115 Mc
67 Ho	68 Er	69 Tm
99 Es	100 Fm	101 Md



* Actually, they form a tetrahedron.

13 14 15

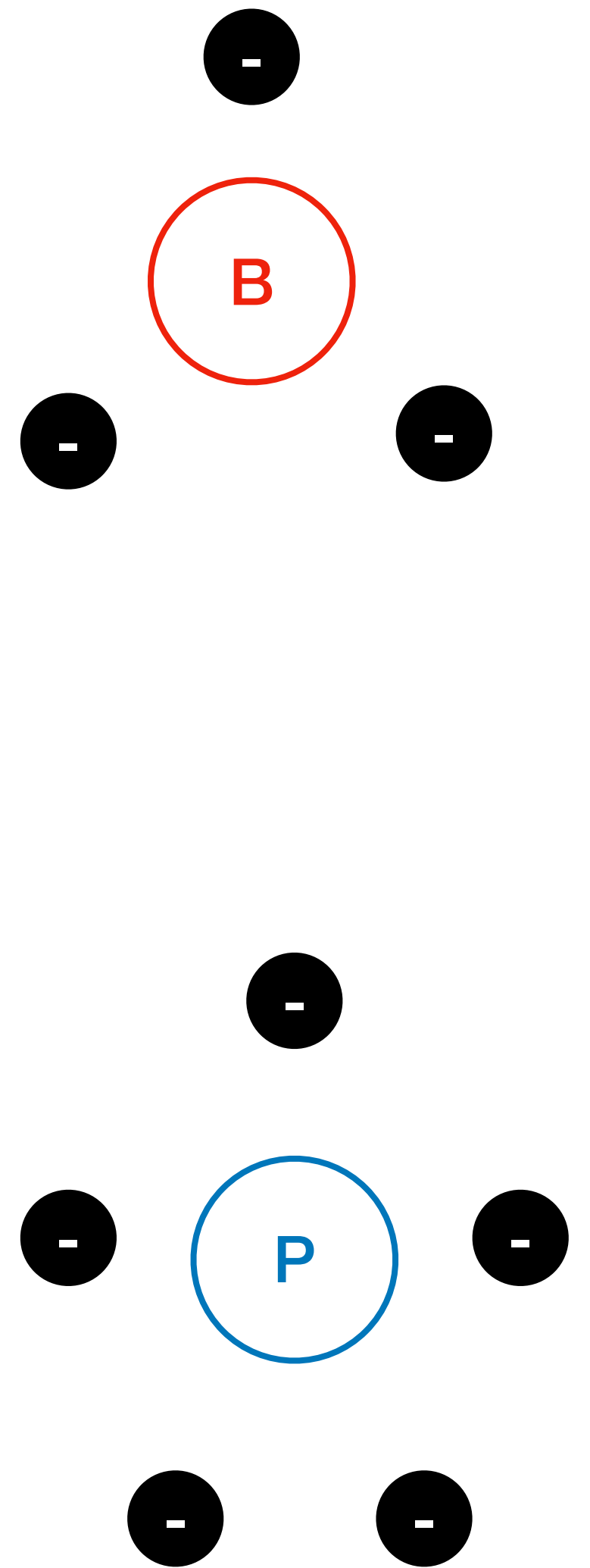
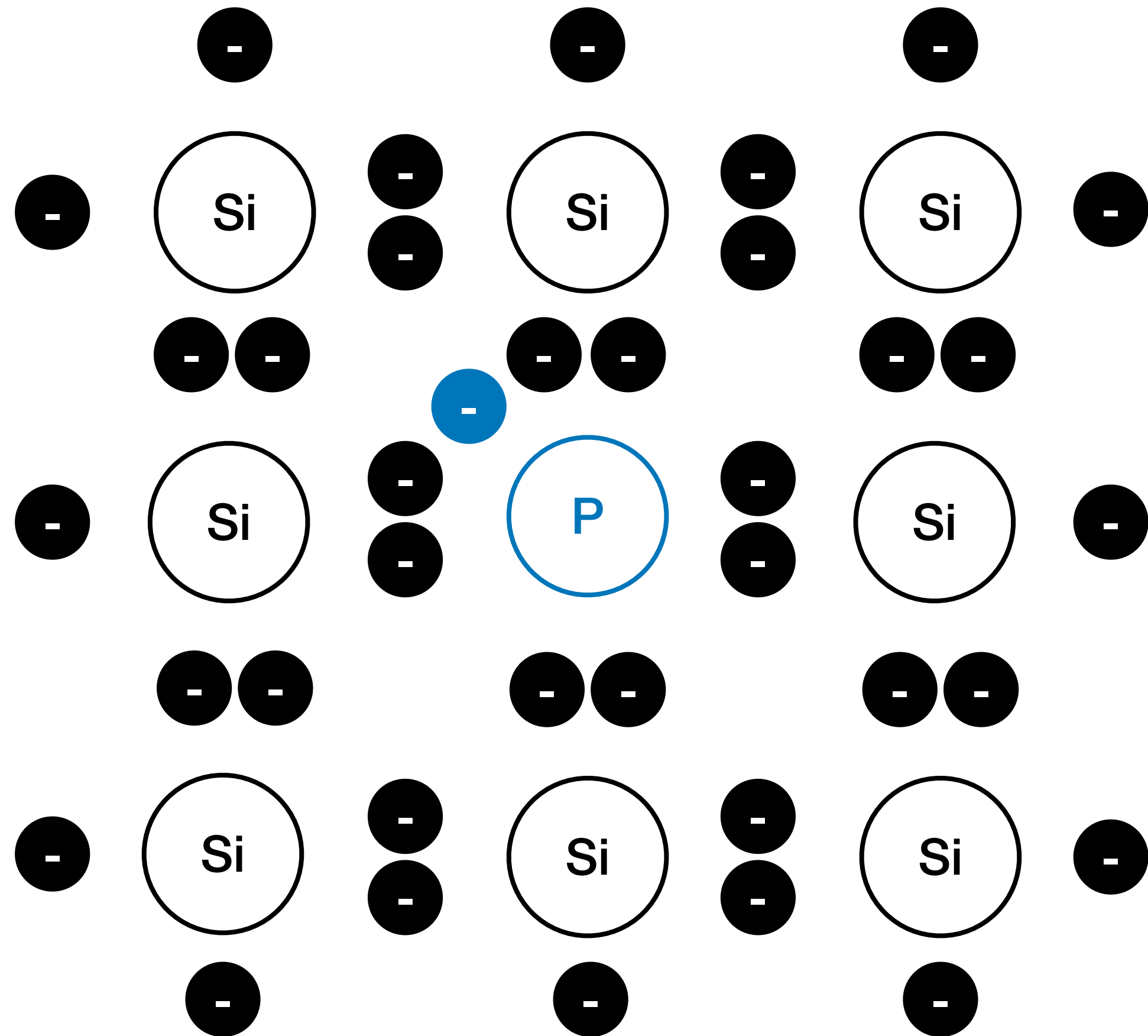
5 B	6 C	7 N
13 Al	14 Si	15 P
31 Ga	32 Ge	33 As
49 In	50 Sn	51 Sb
81 Tl	82 Pb	83 Bi
113 Nh	114 Fl	115 Mc
67 Ho	68 Er	69 Tm
99 Es	100 Fm	101 Md



* Actually, they form a tetrahedron.

13 14 15

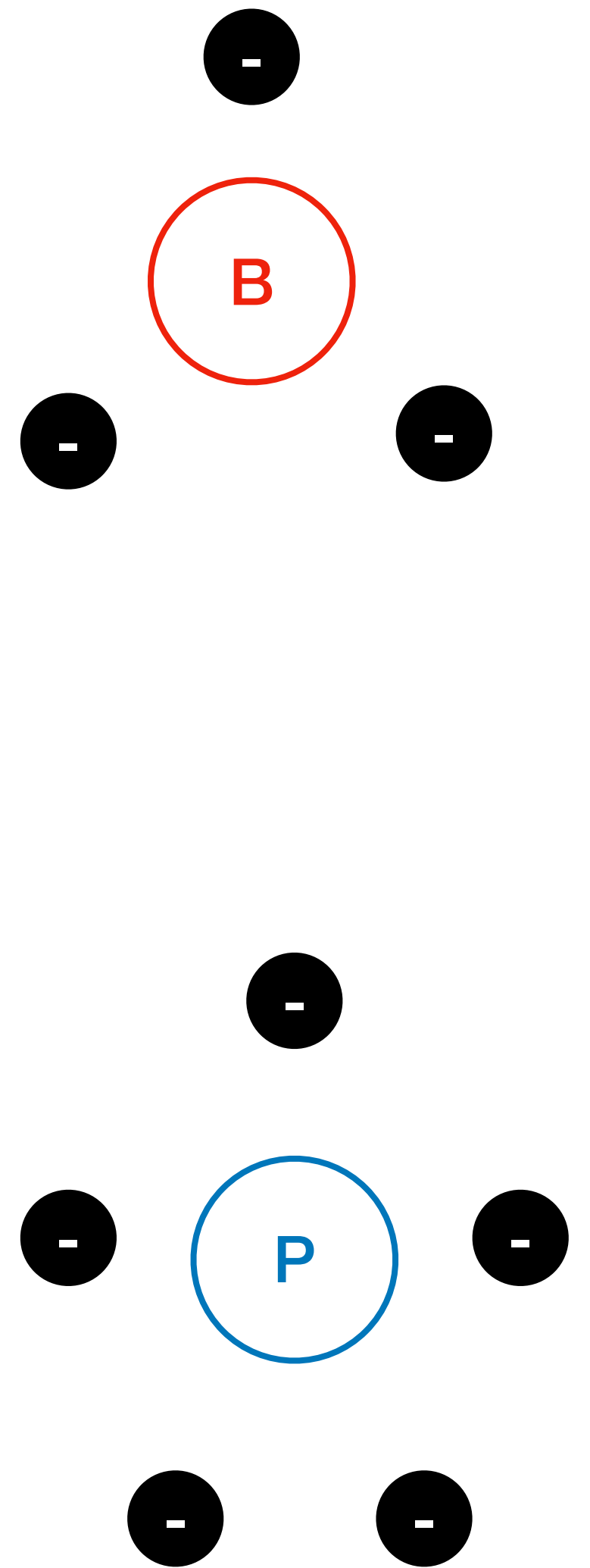
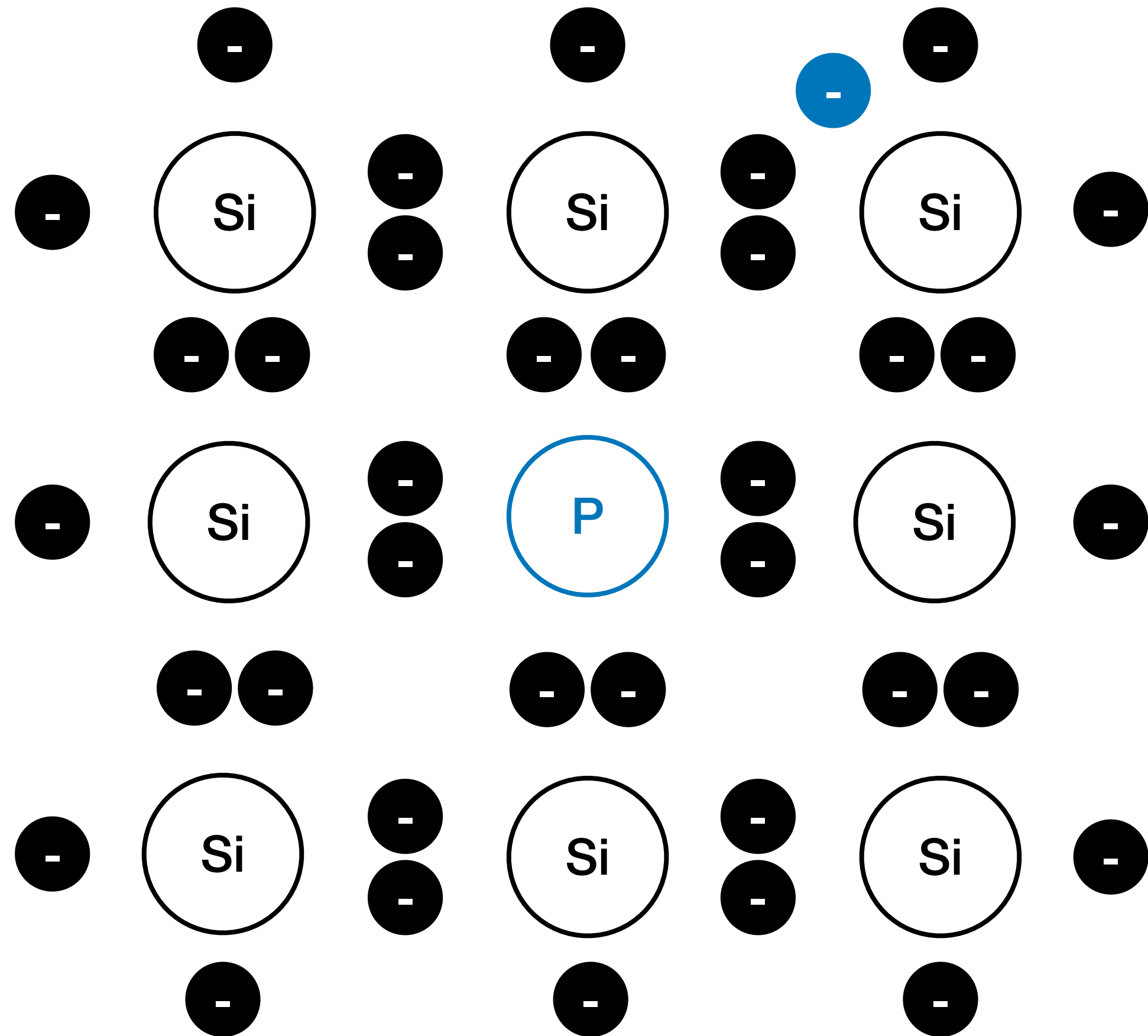
5 B	6 C	7 N
13 Al	14 Si	15 P
31 Ga	32 Ge	33 As
49 In	50 Sn	51 Sb
81 Tl	82 Pb	83 Bi
113 Nh	114 Fl	115 Mc
67 Ho	68 Er	69 Tm
99 Es	100 Fm	101 Md



* Actually, they form a tetrahedron.

13 14 15

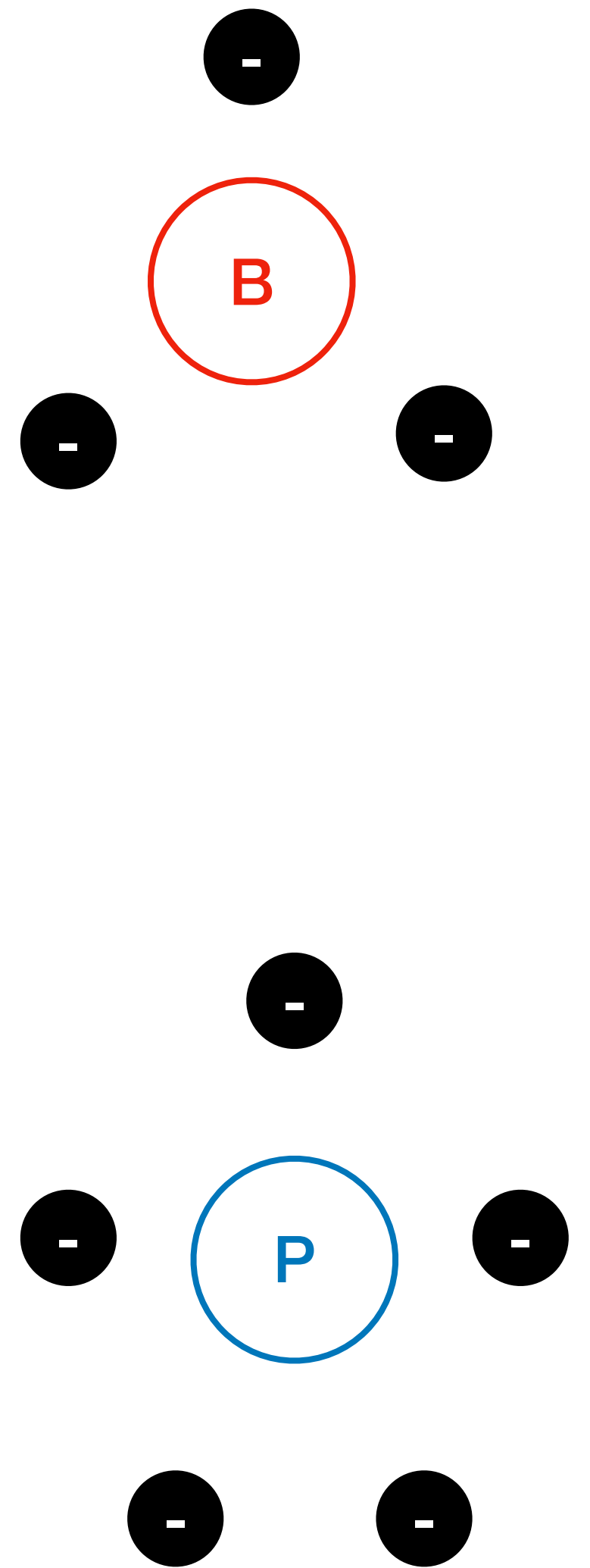
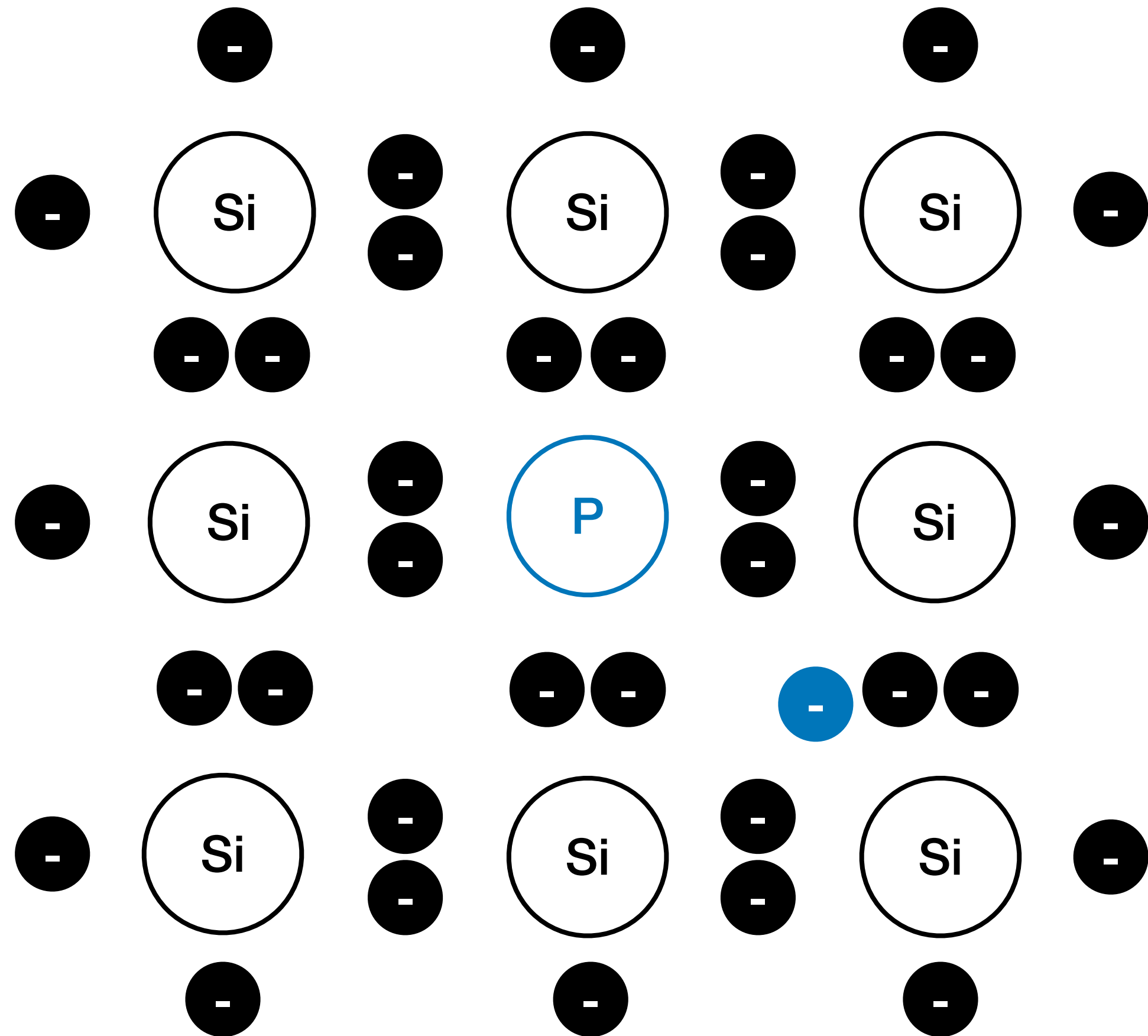
5 B	6 C	7 N
13 Al	14 Si	15 P
31 Ga	32 Ge	33 As
49 In	50 Sn	51 Sb
81 Tl	82 Pb	83 Bi
113 Nh	114 Fl	115 Mc
67 Ho	68 Er	69 Tm
99 Es	100 Fm	101 Md



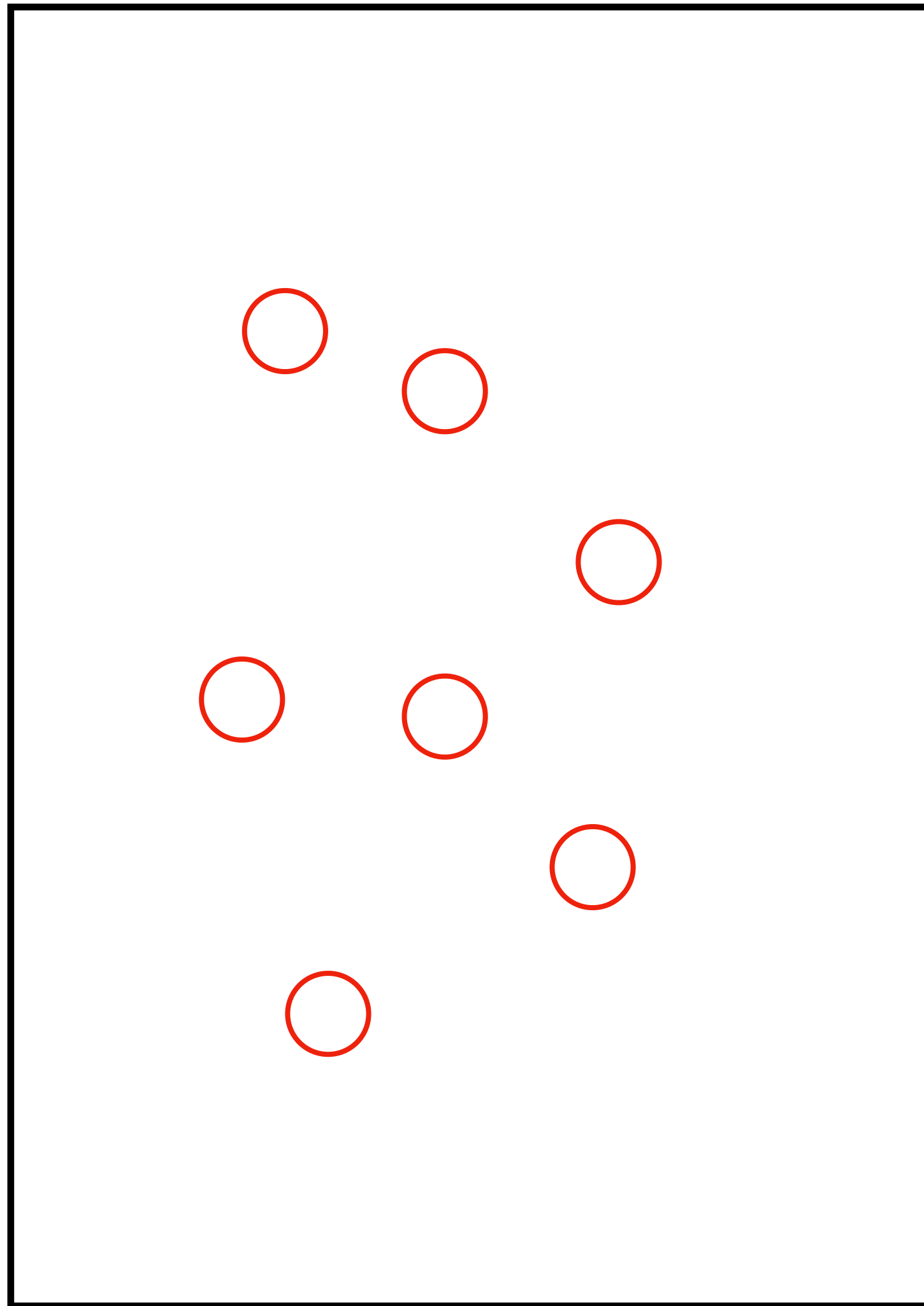
* Actually, they form a tetrahedron.

13 14 15

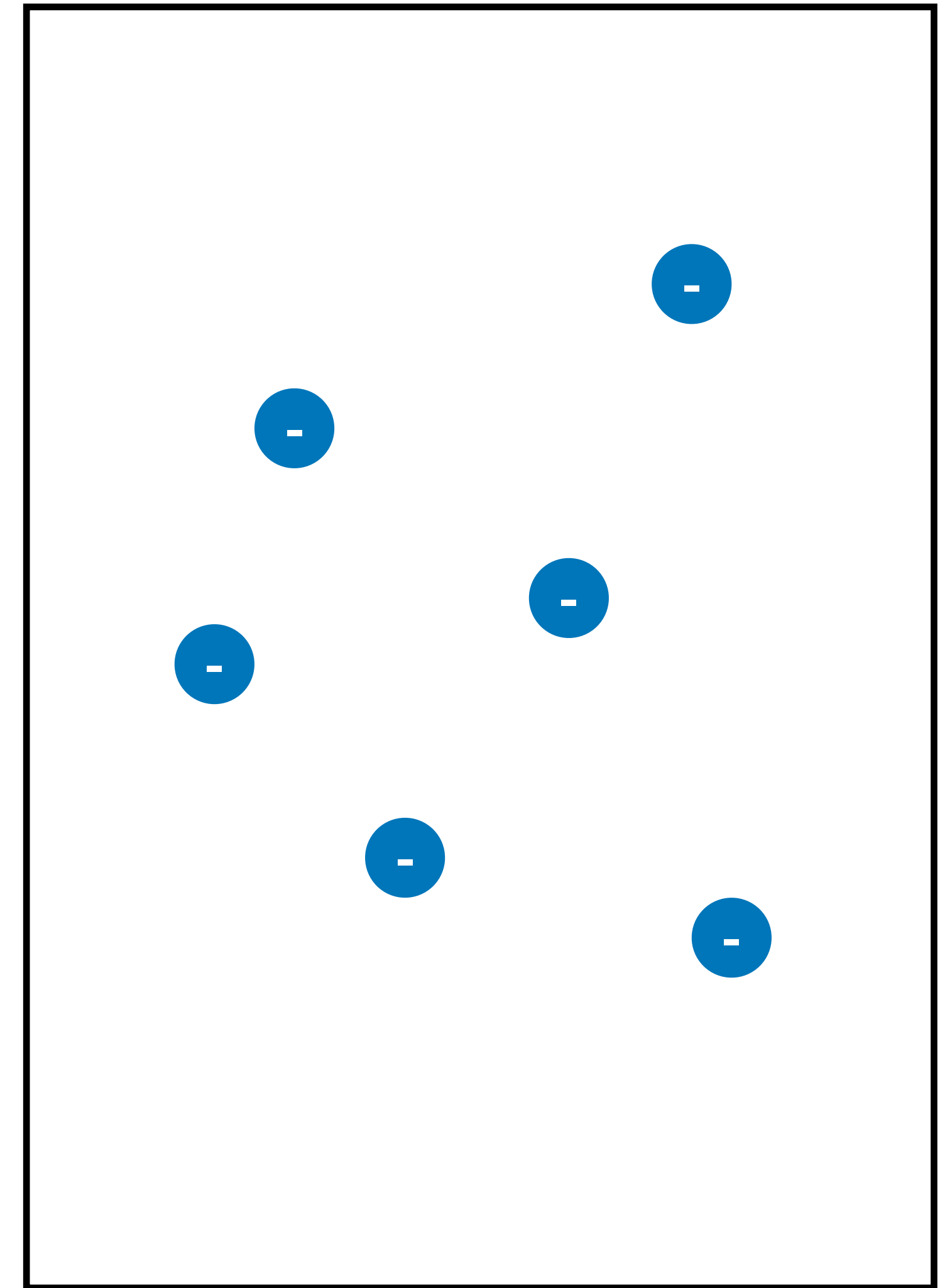
5 B	6 C	7 N
13 Al	14 Si	15 P
31 Ga	32 Ge	33 As
49 In	50 Sn	51 Sb
81 Tl	82 Pb	83 Bi
113 Nh	114 Fl	115 Mc
67 Ho	68 Er	69 Tm
99 Es	100 Fm	101 Md



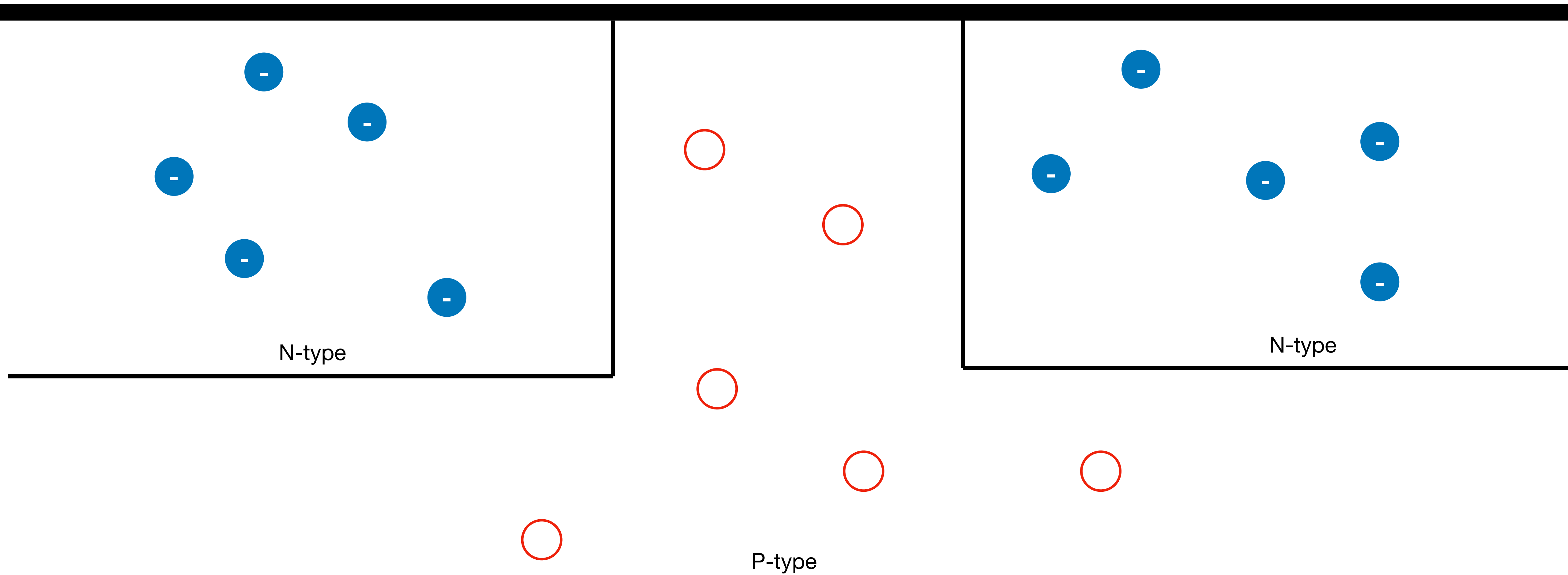
* Actually, they form a tetrahedron.

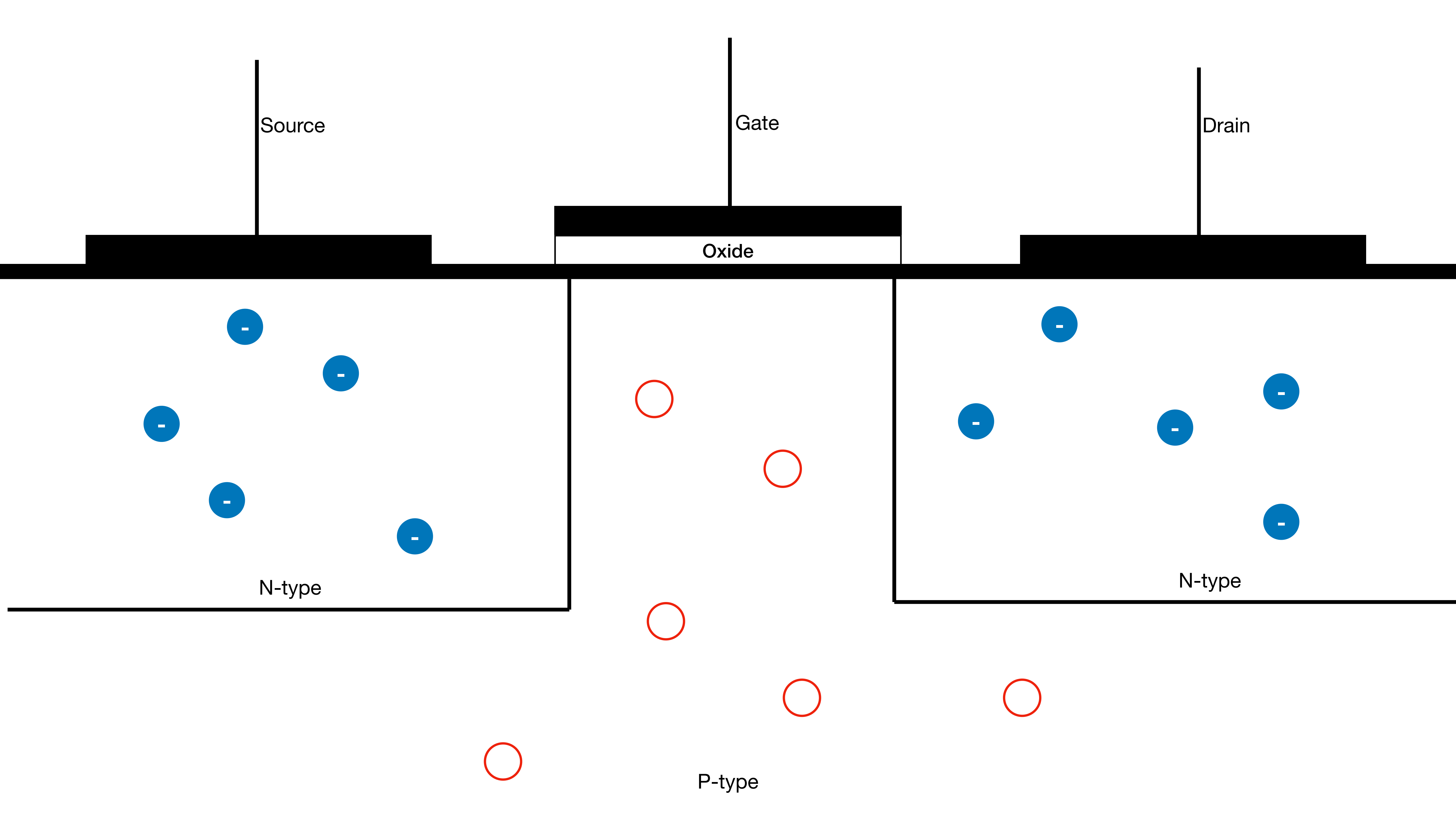


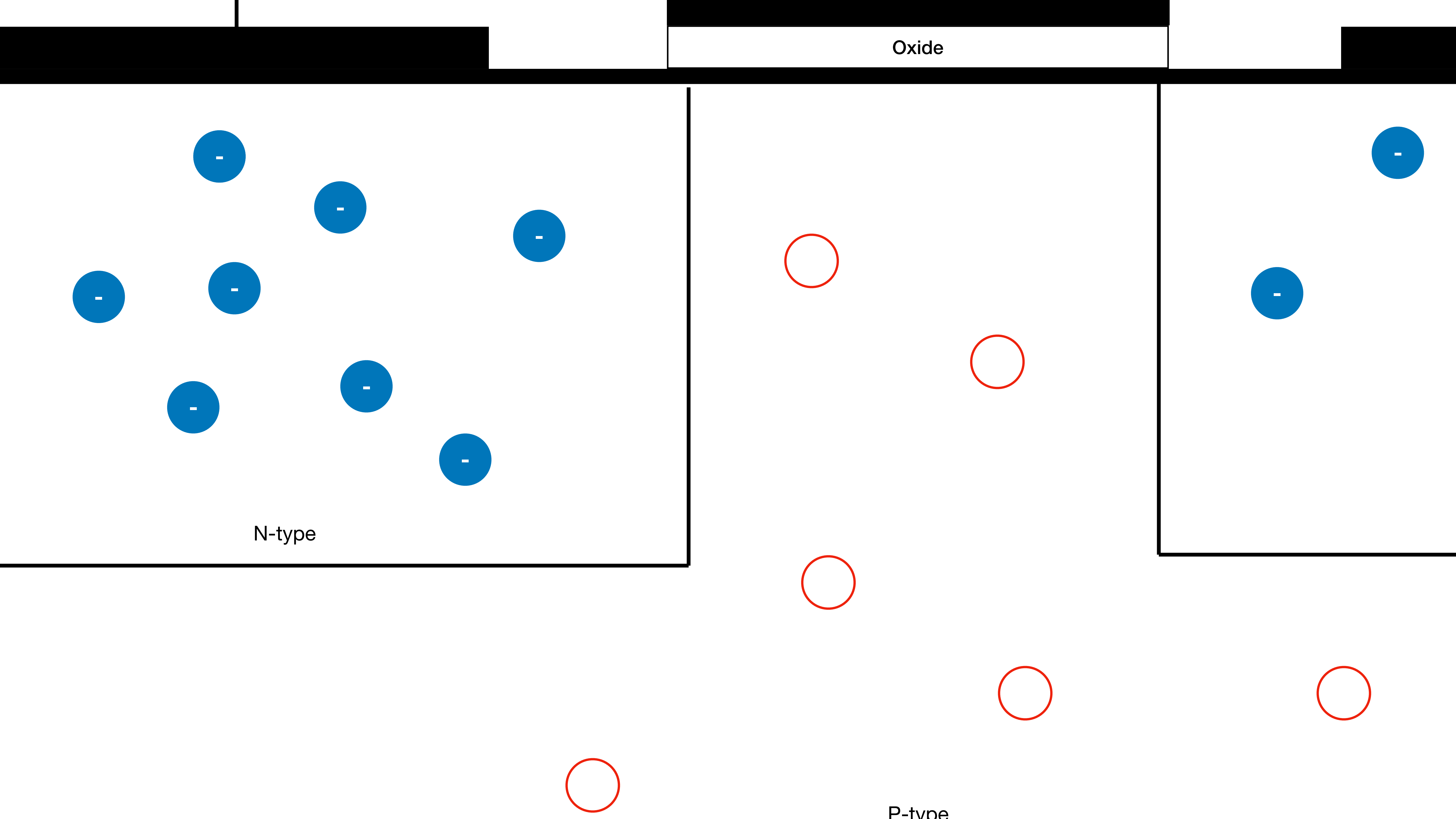
P-type Semiconductor

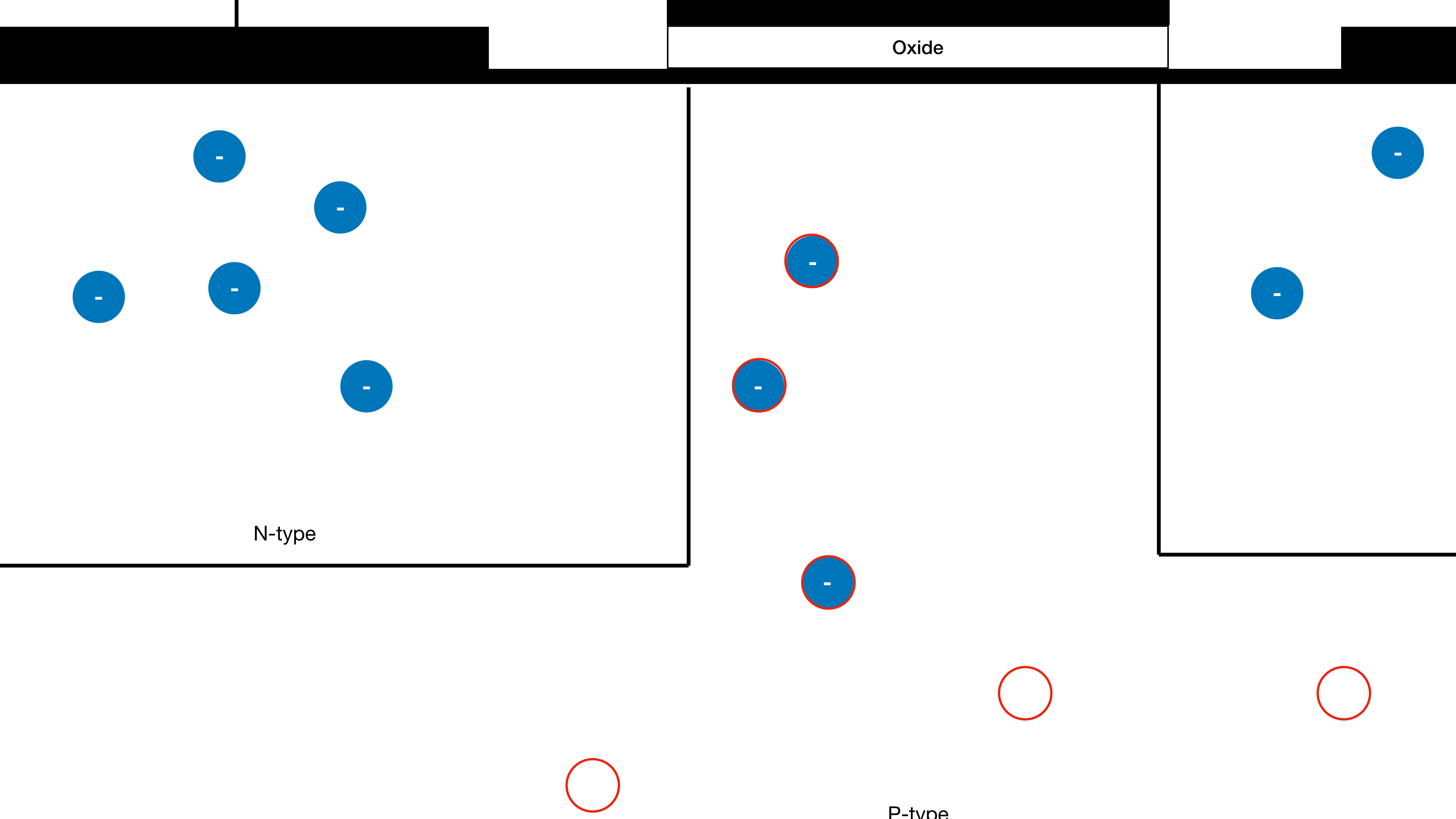


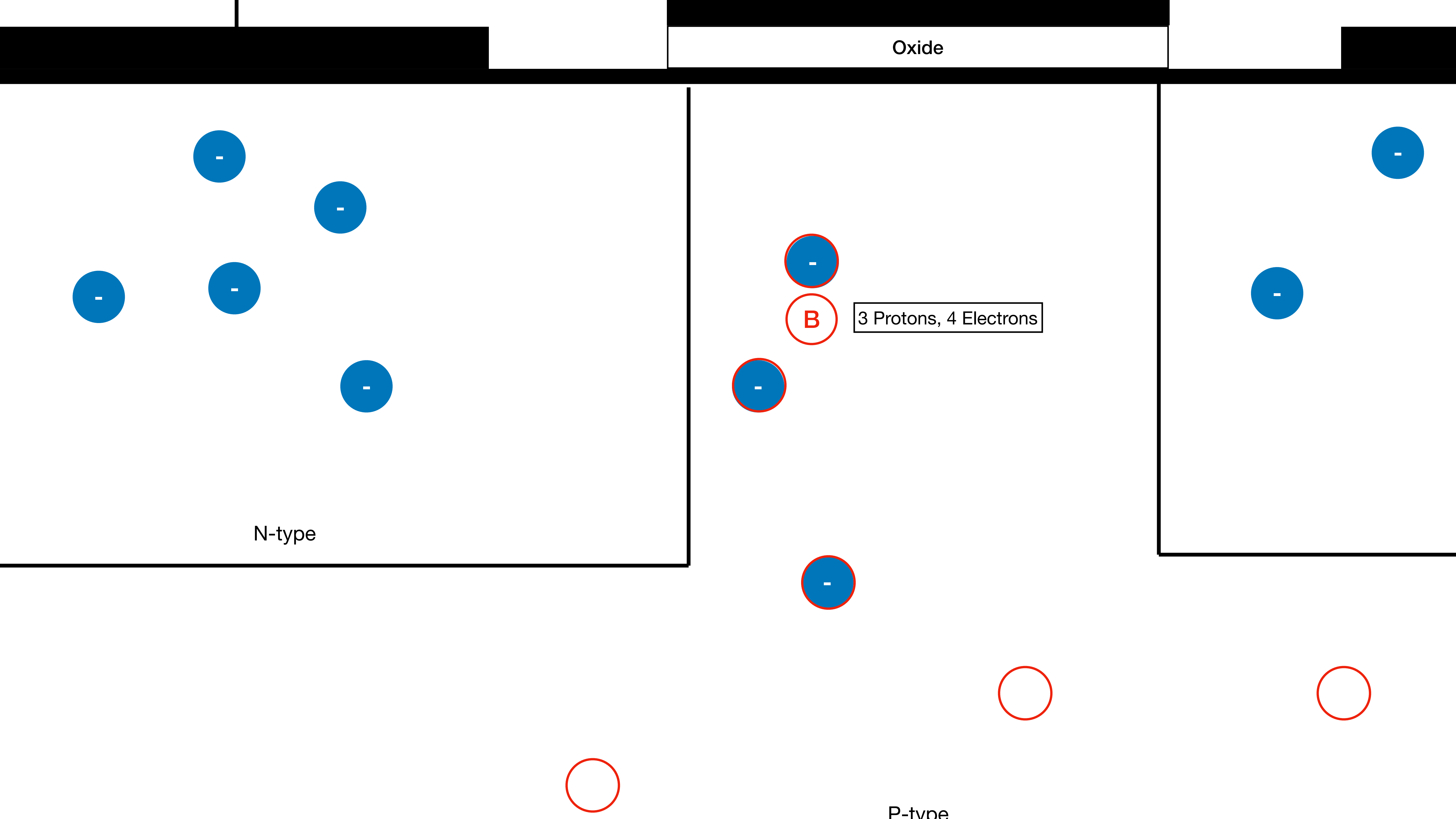
N-type Semiconductor

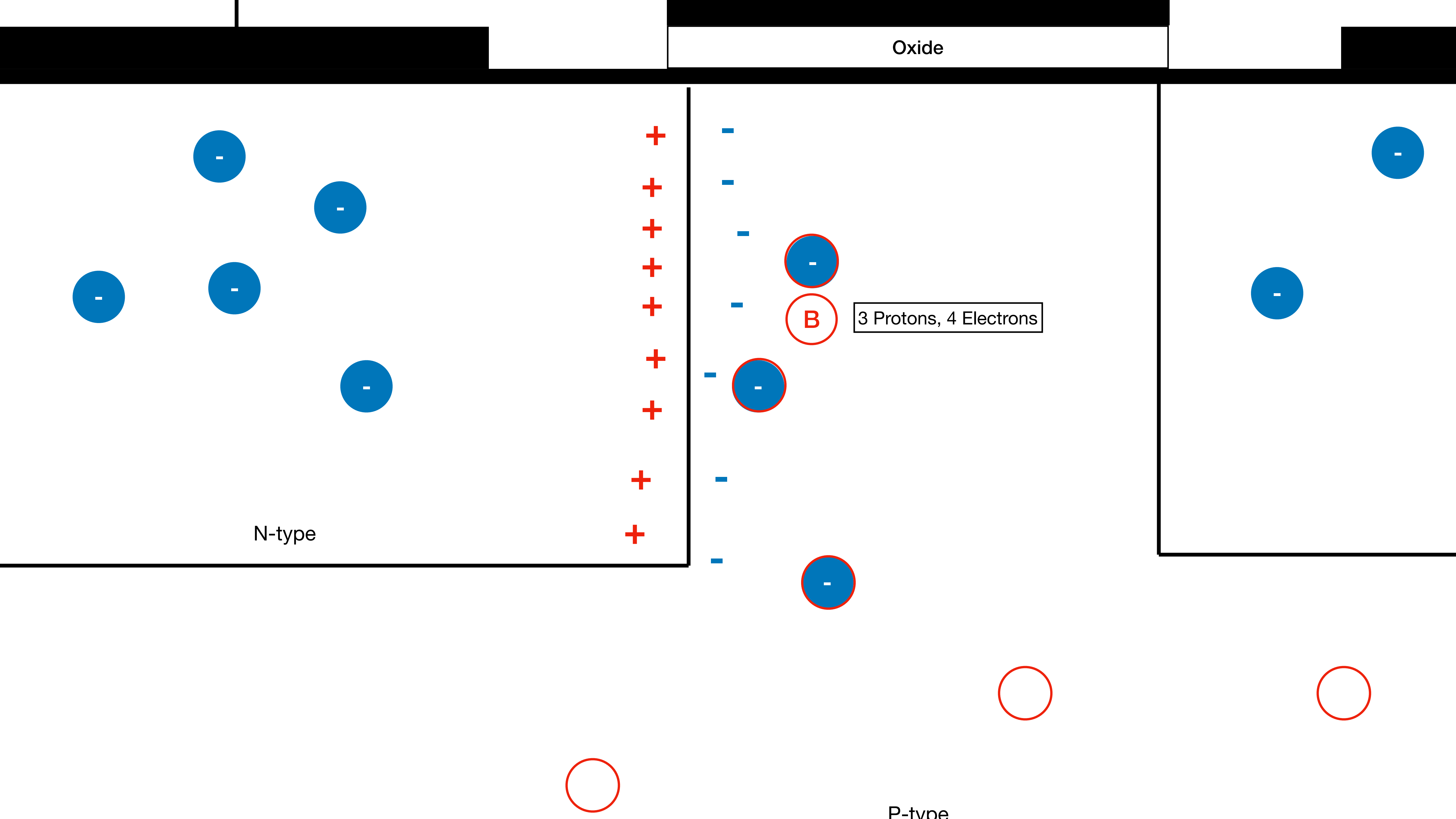


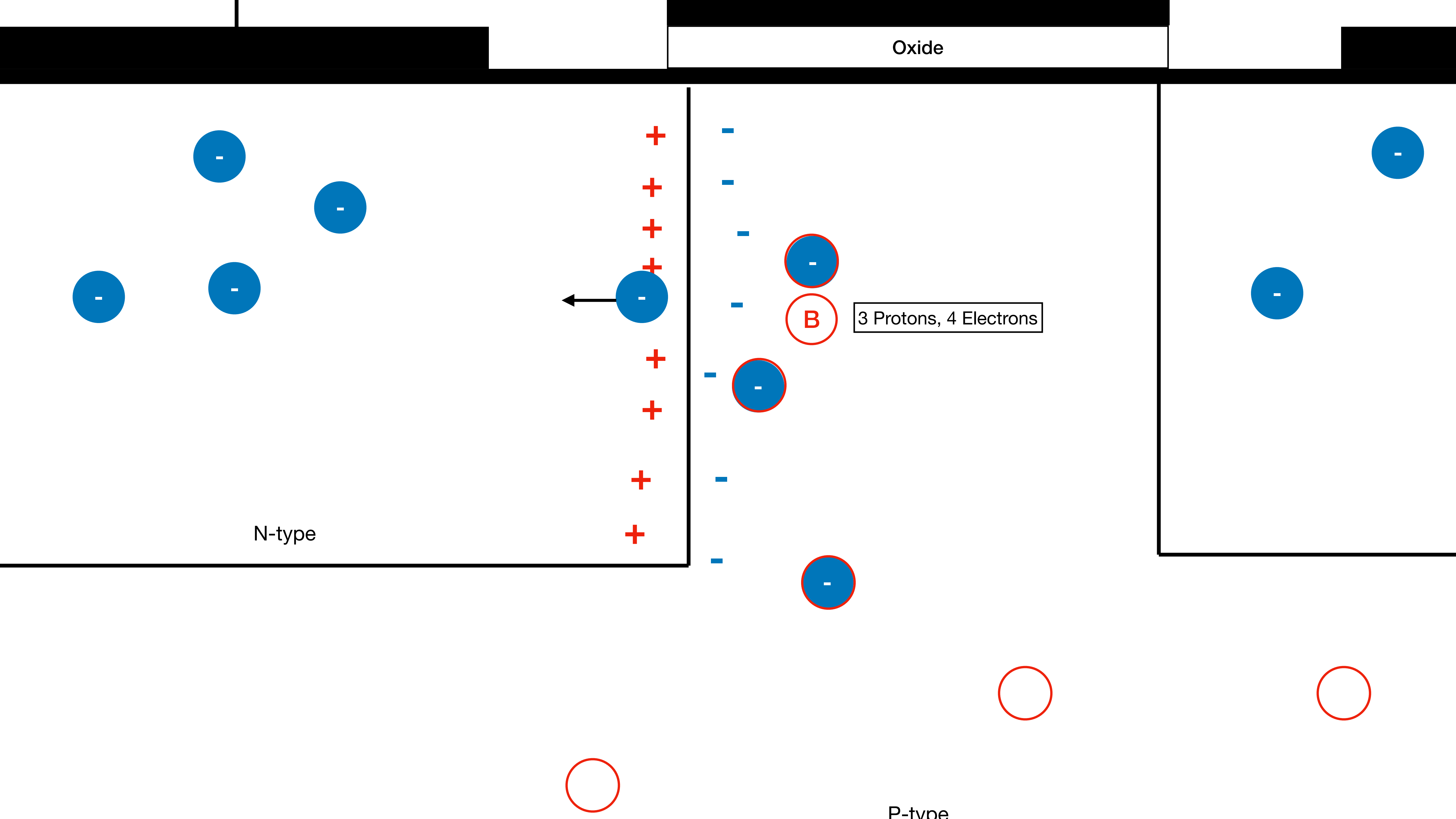


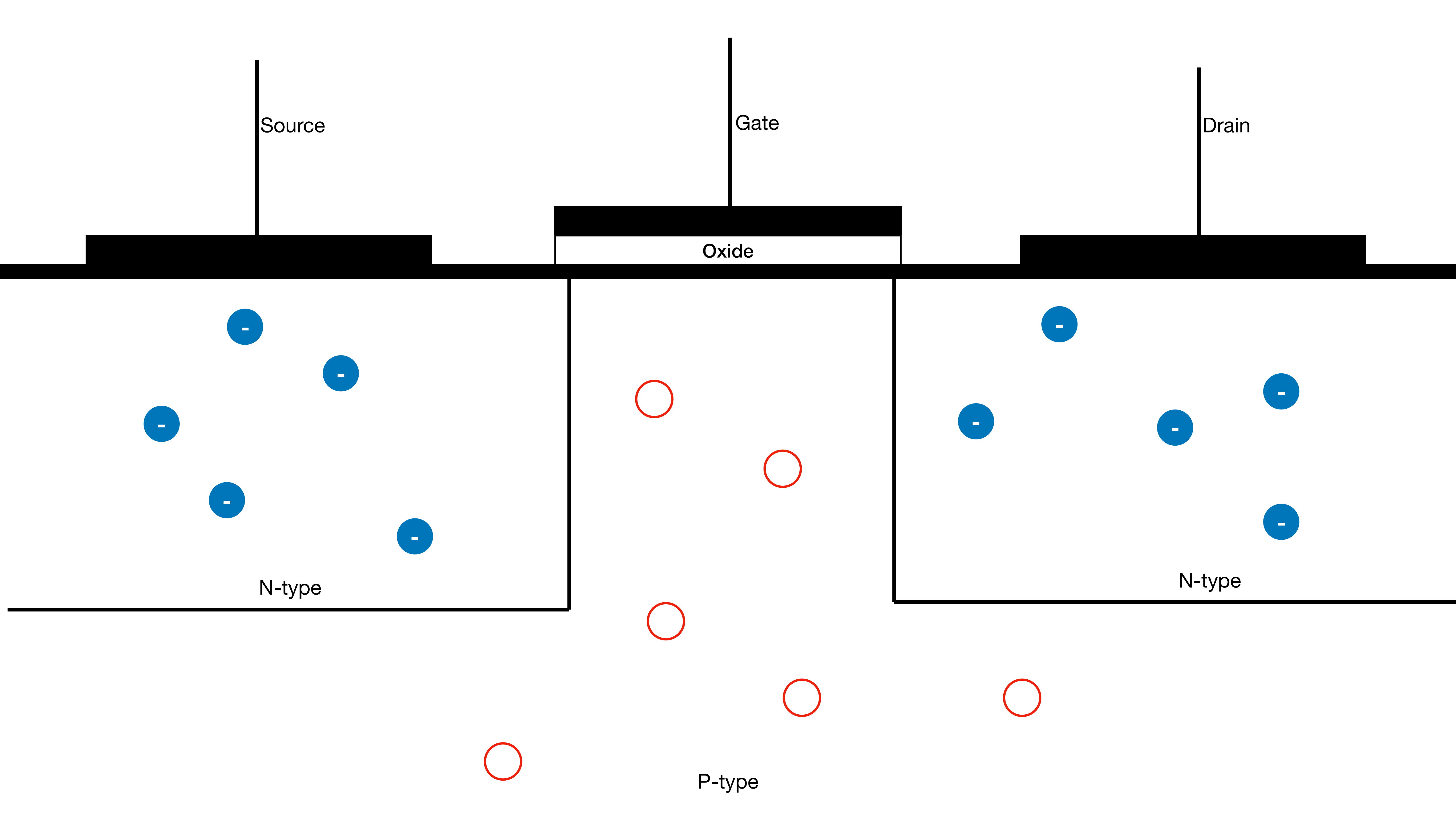


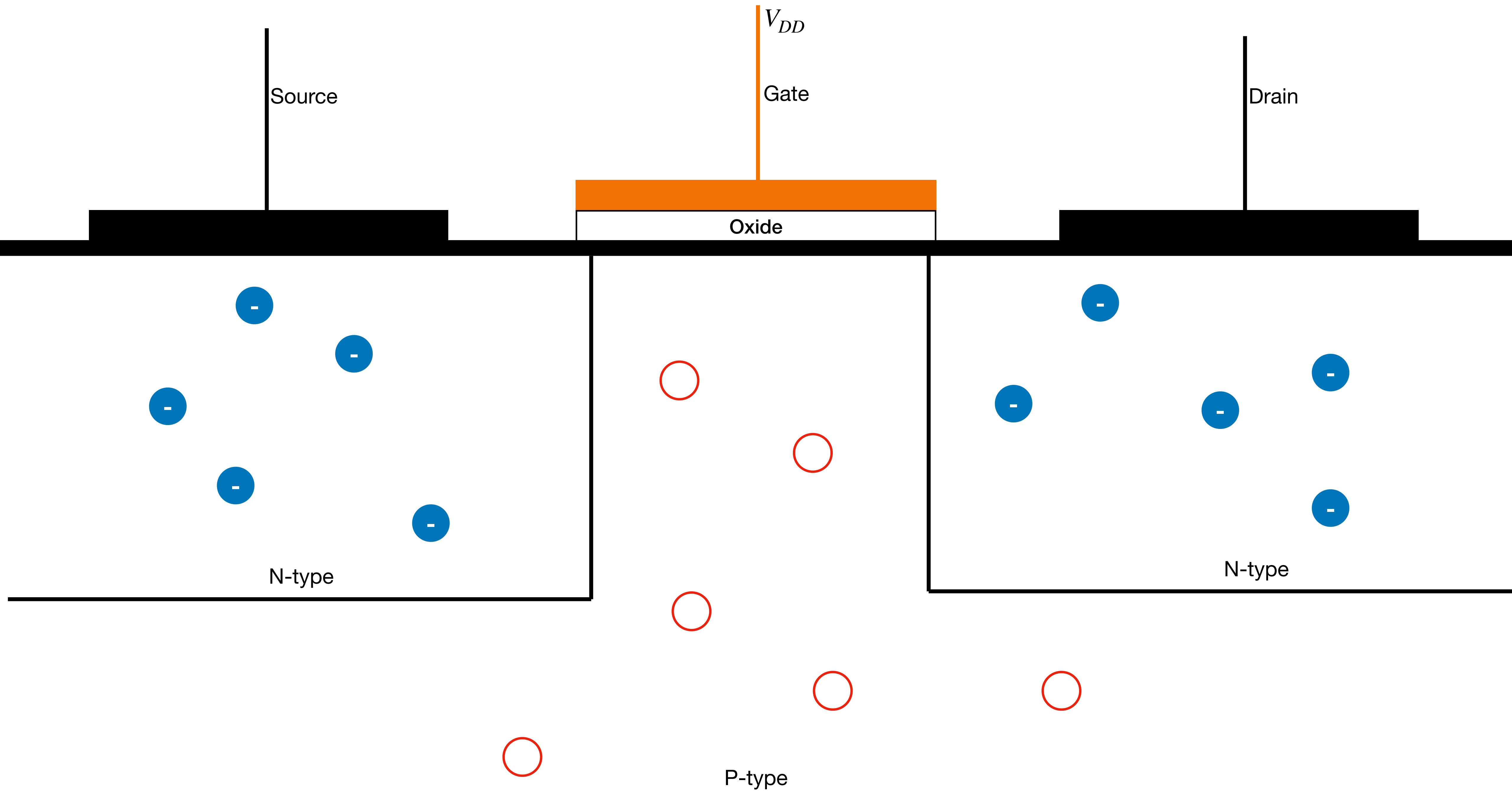


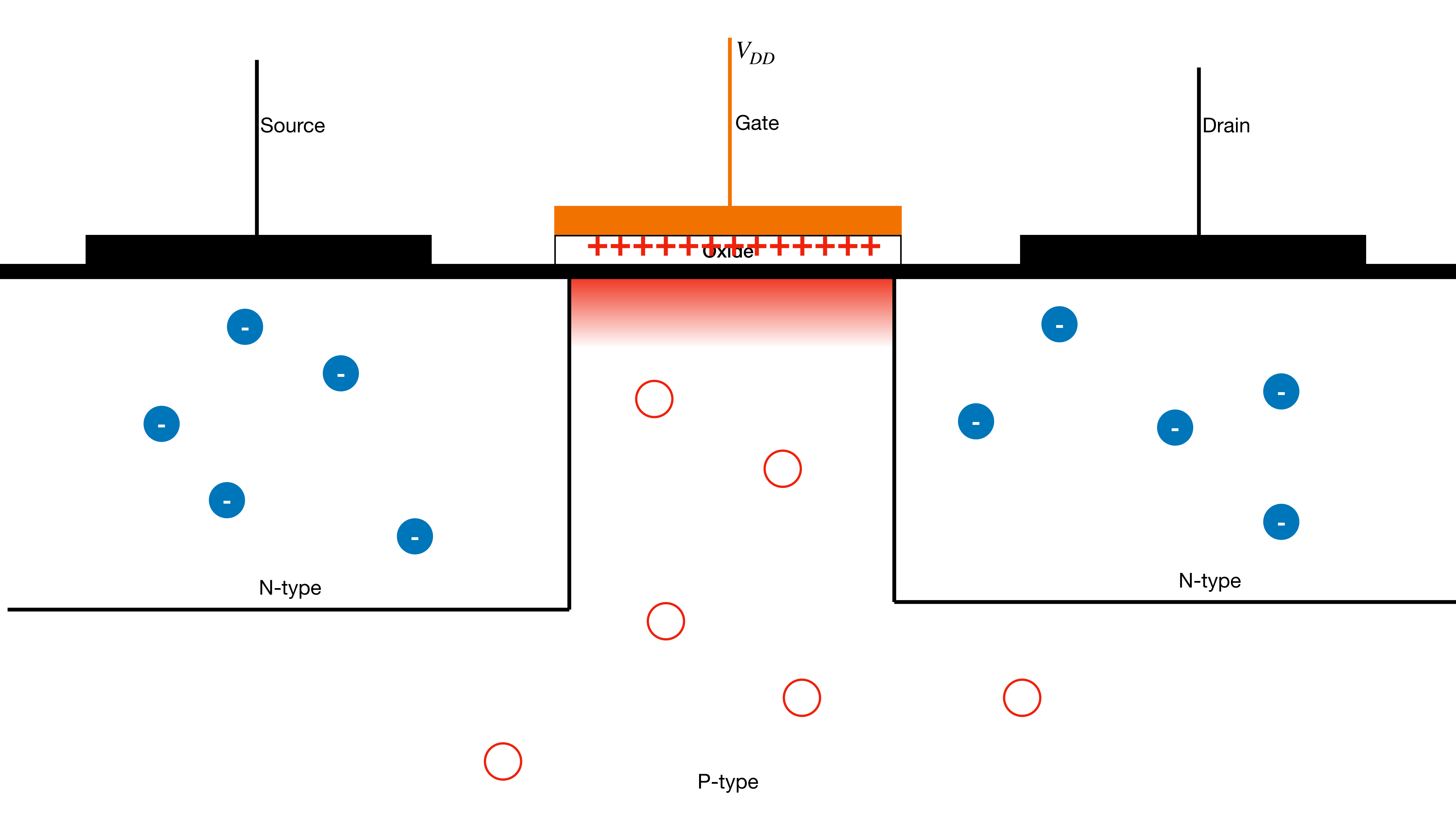


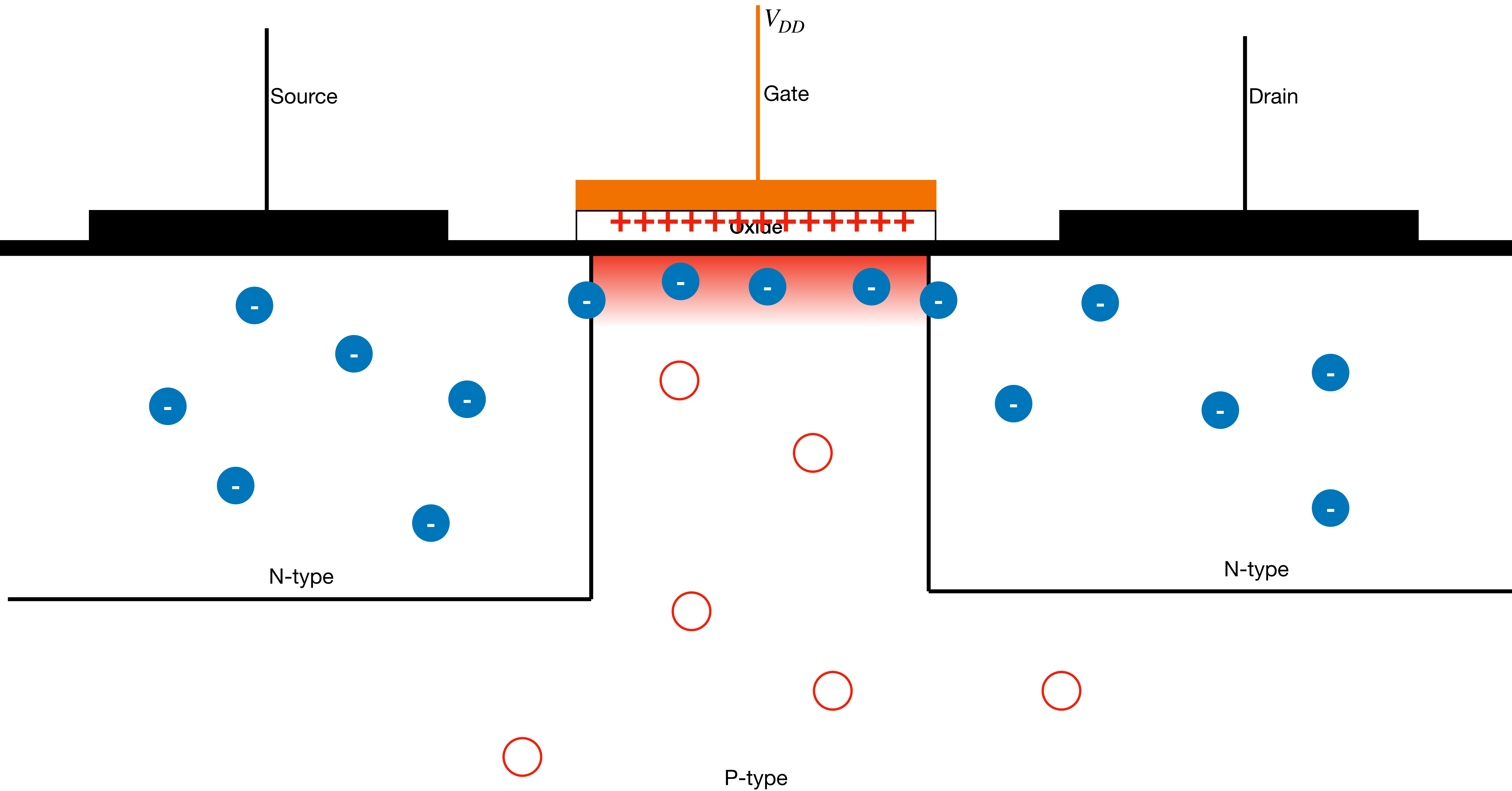


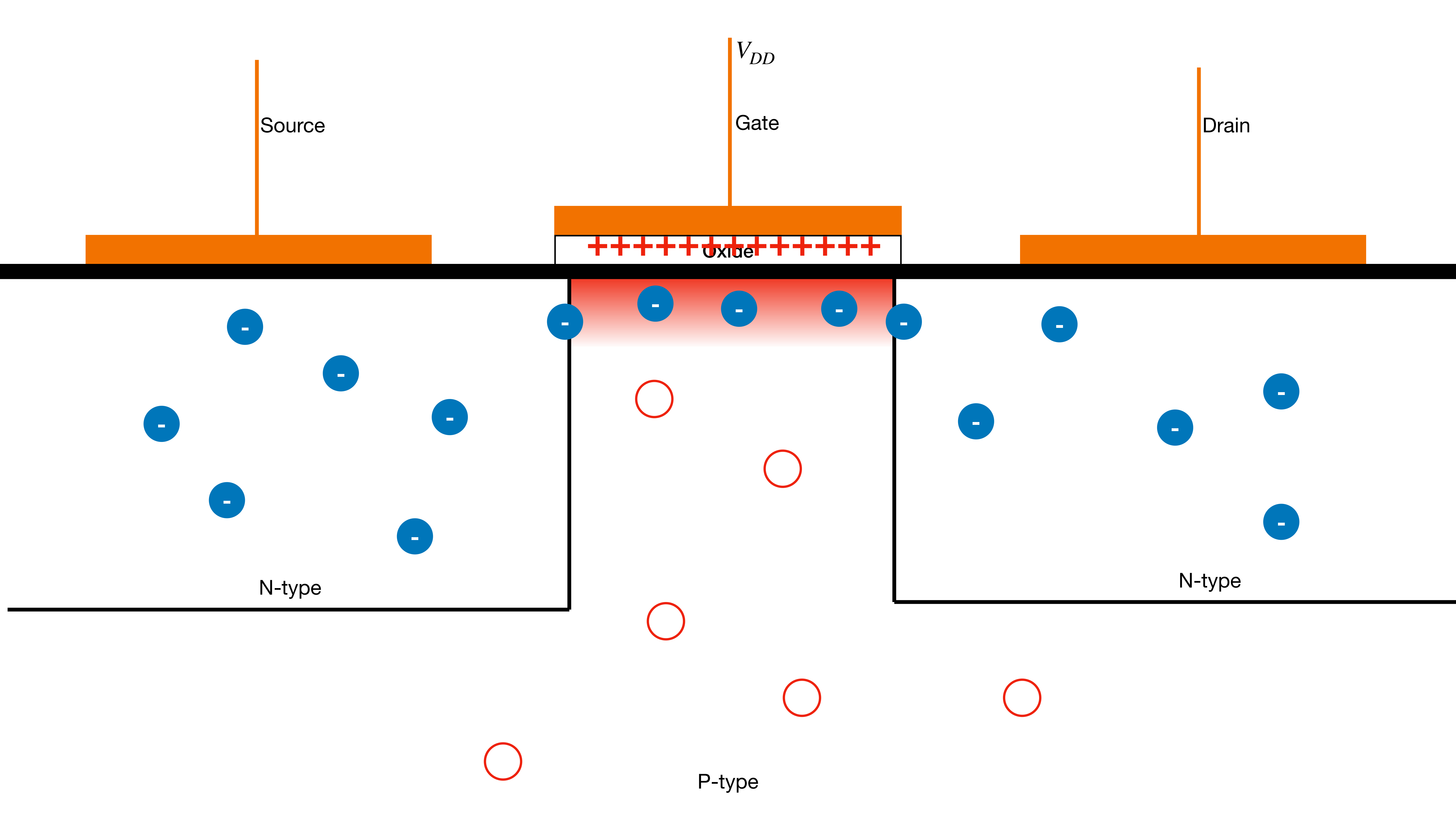


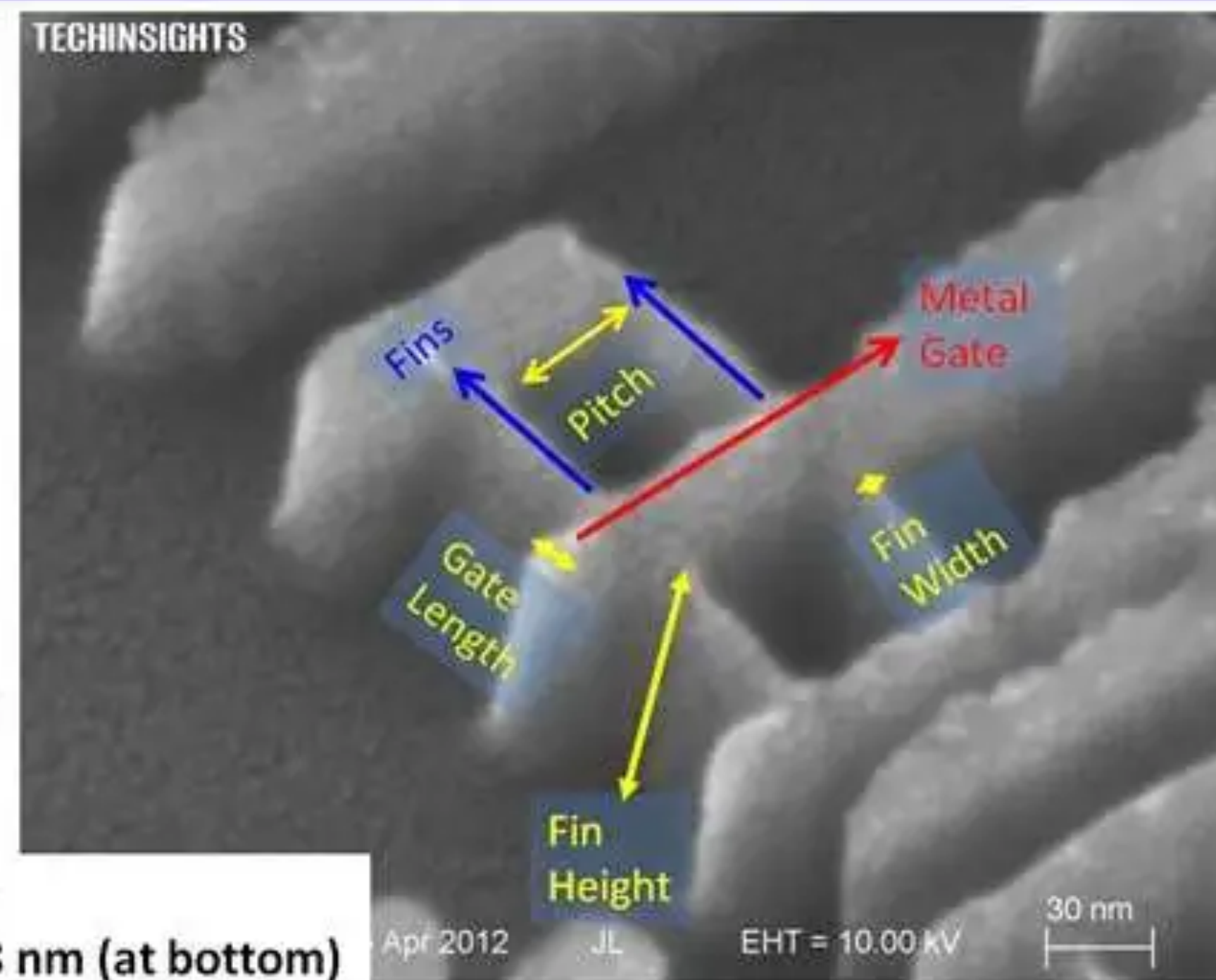
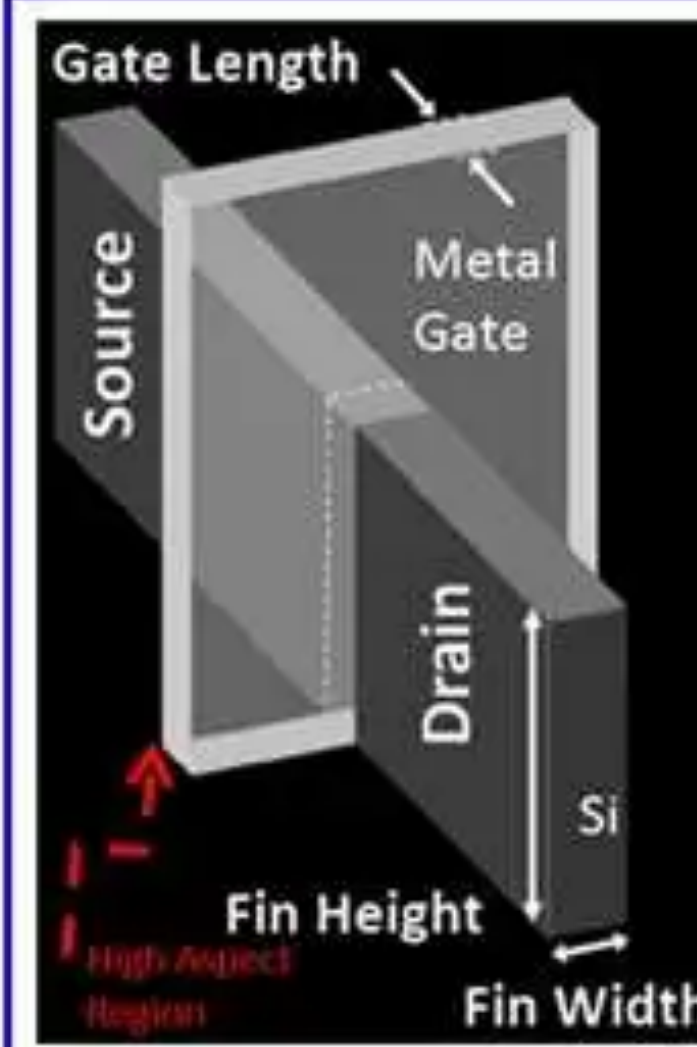
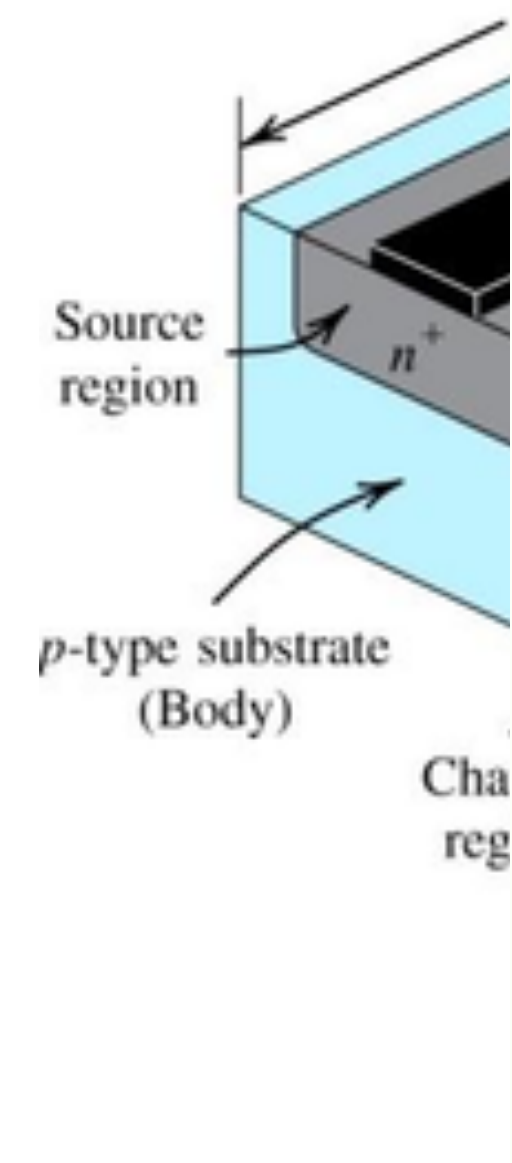






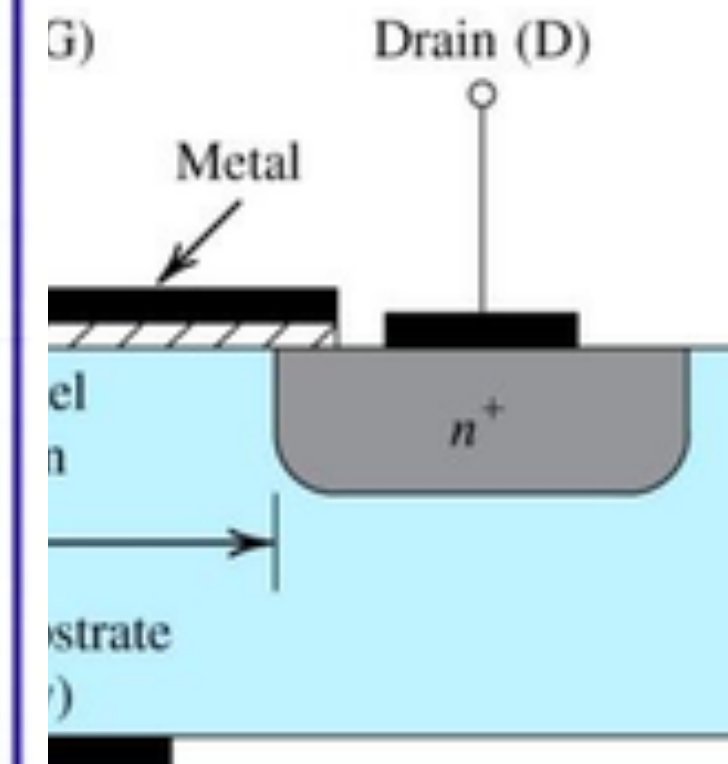






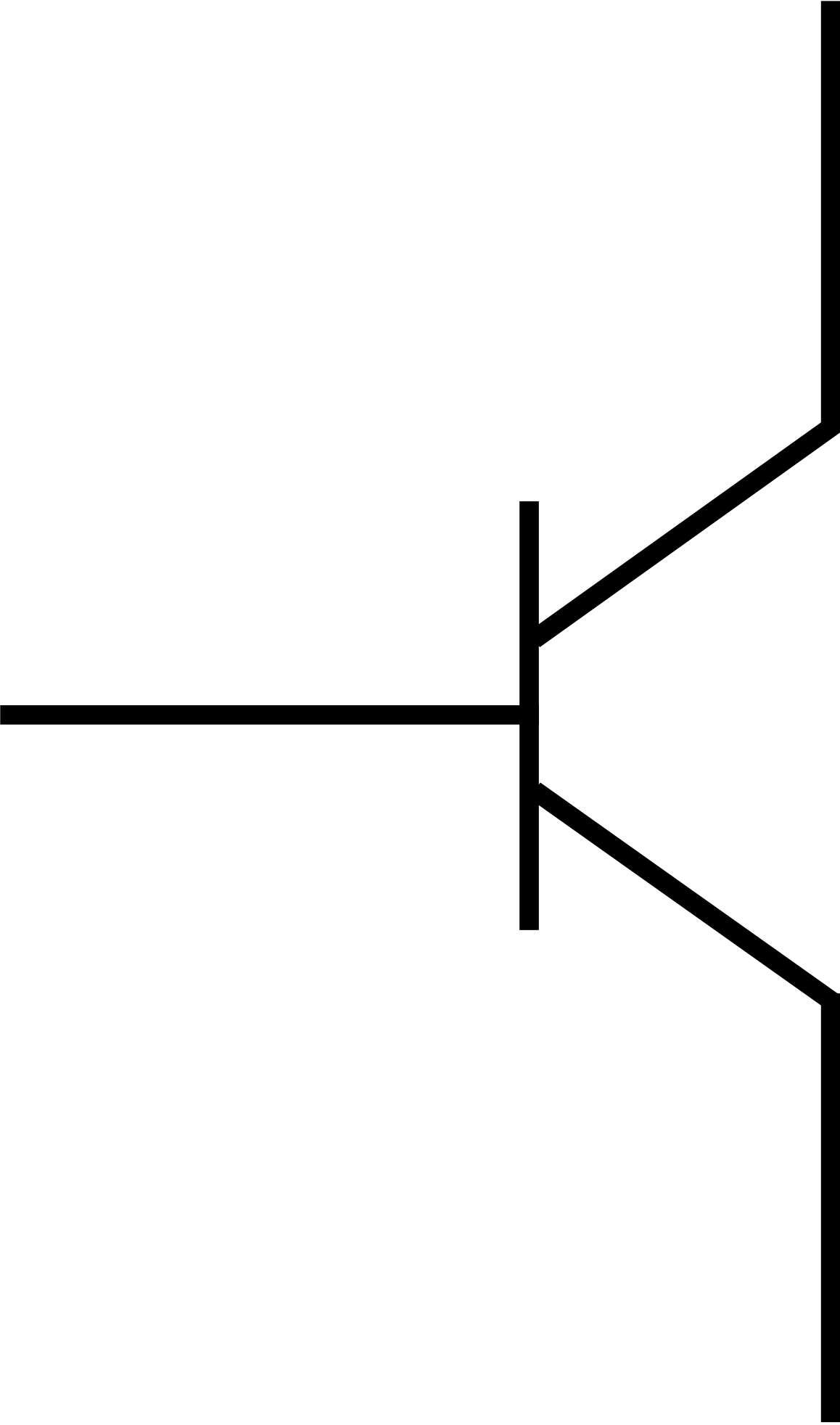
Average Gate height:
 Average Fin width: 18 nm (at bottom)
 Average Fin height: 35 nm
 Average Fin pitch: 60 nm

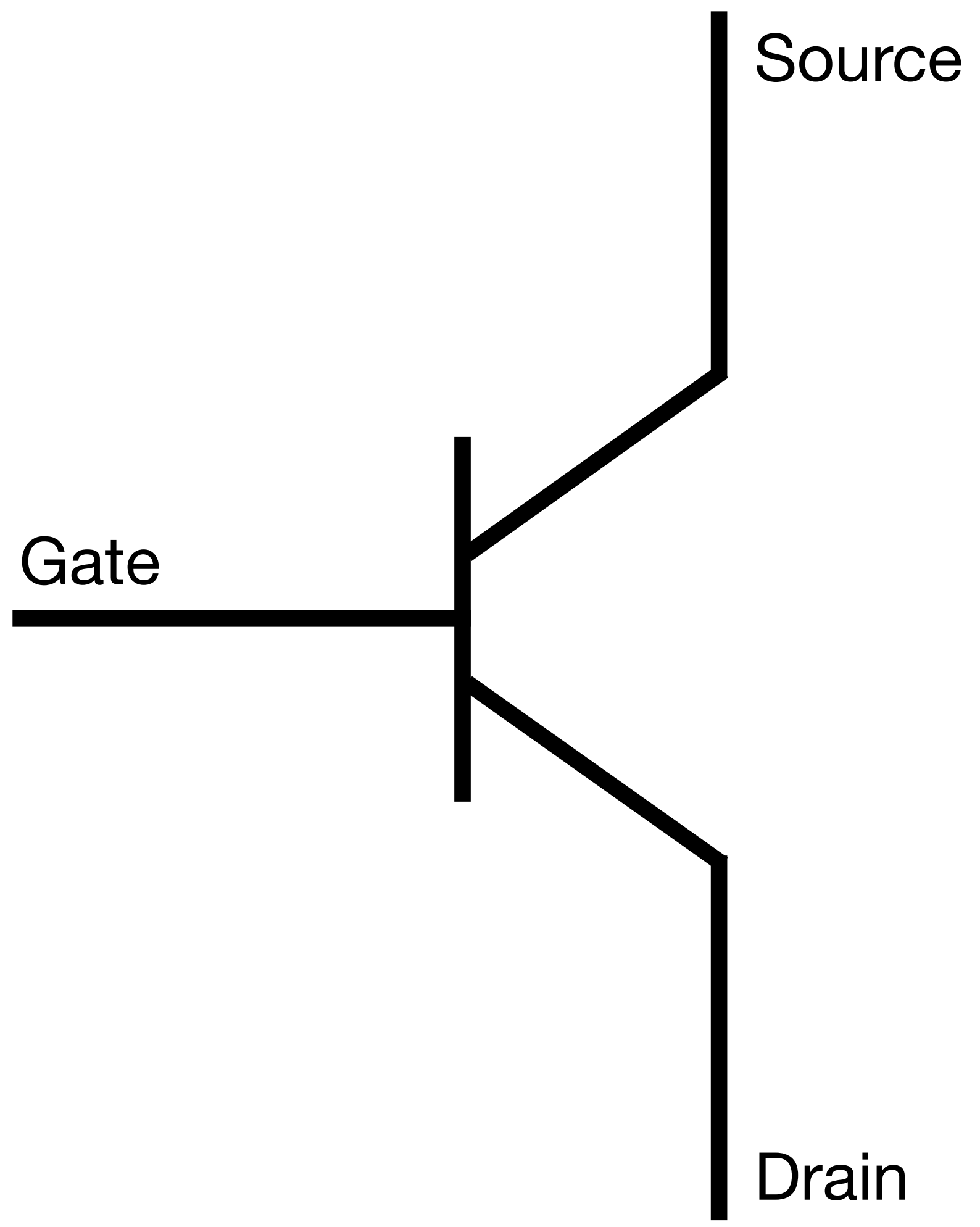
Average Tri-Gate length: ~ 30 nm
 Average Tri-Gate width: ~ 80 nm

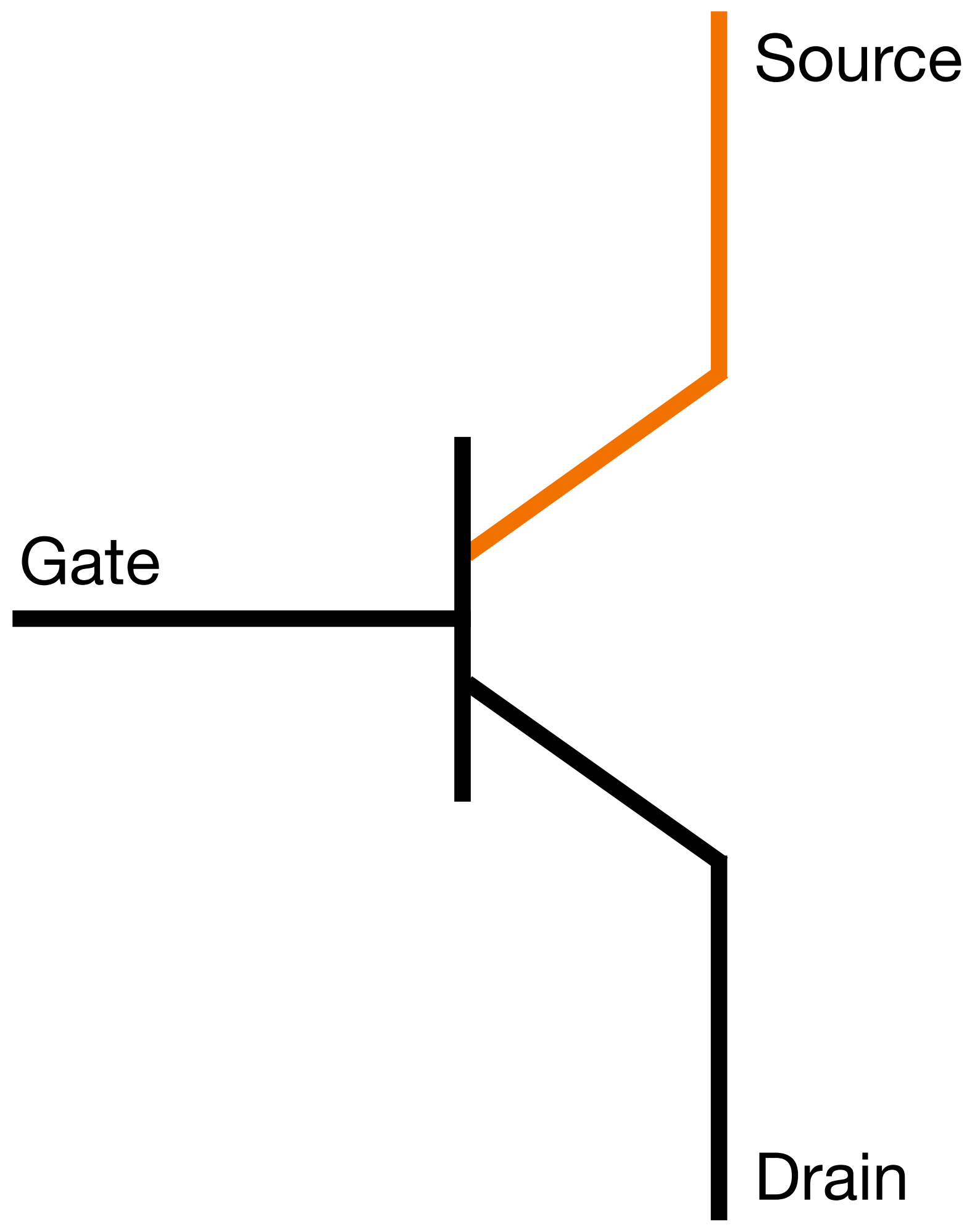


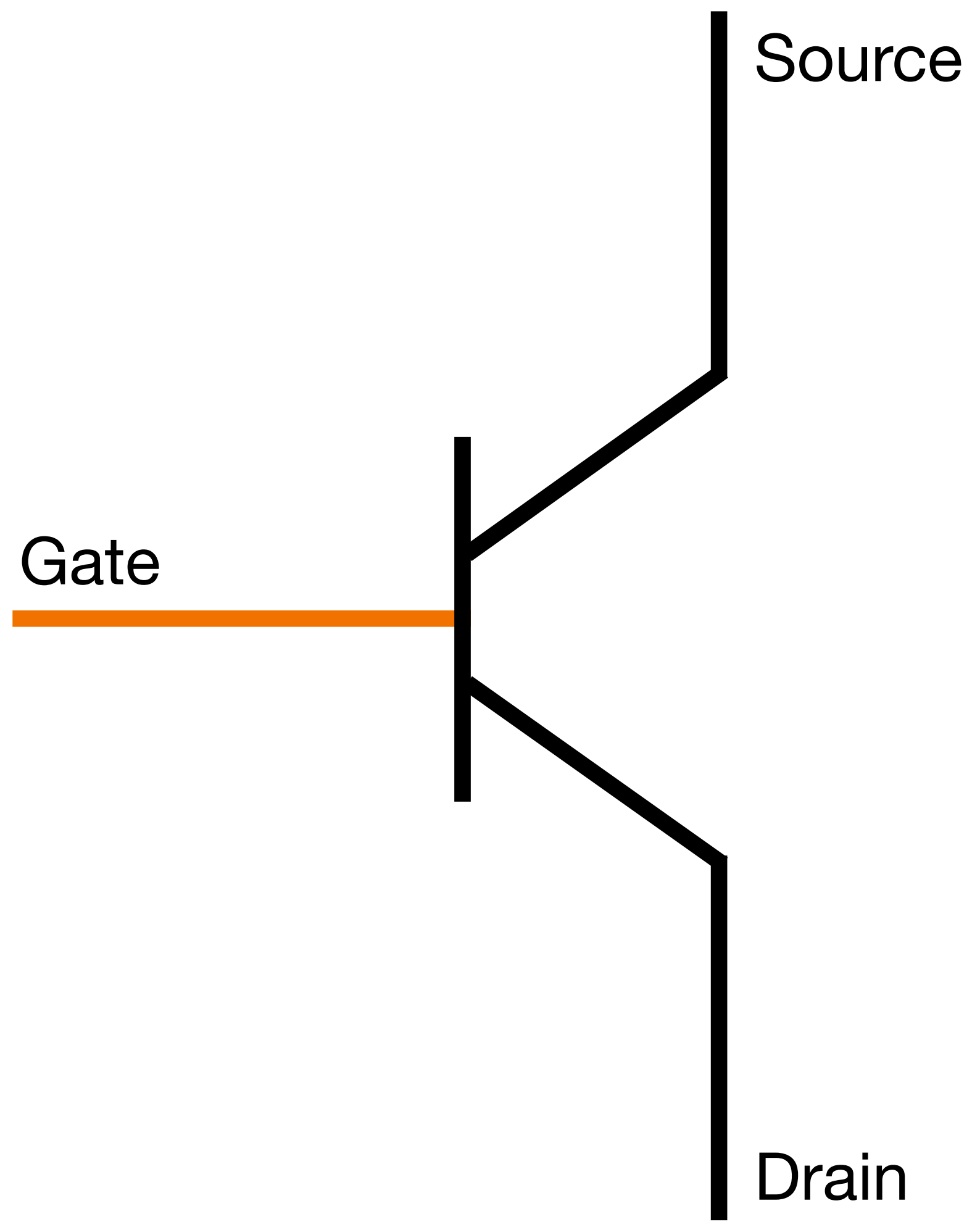
Metal Oxide Semiconductor Field Effect Transistor

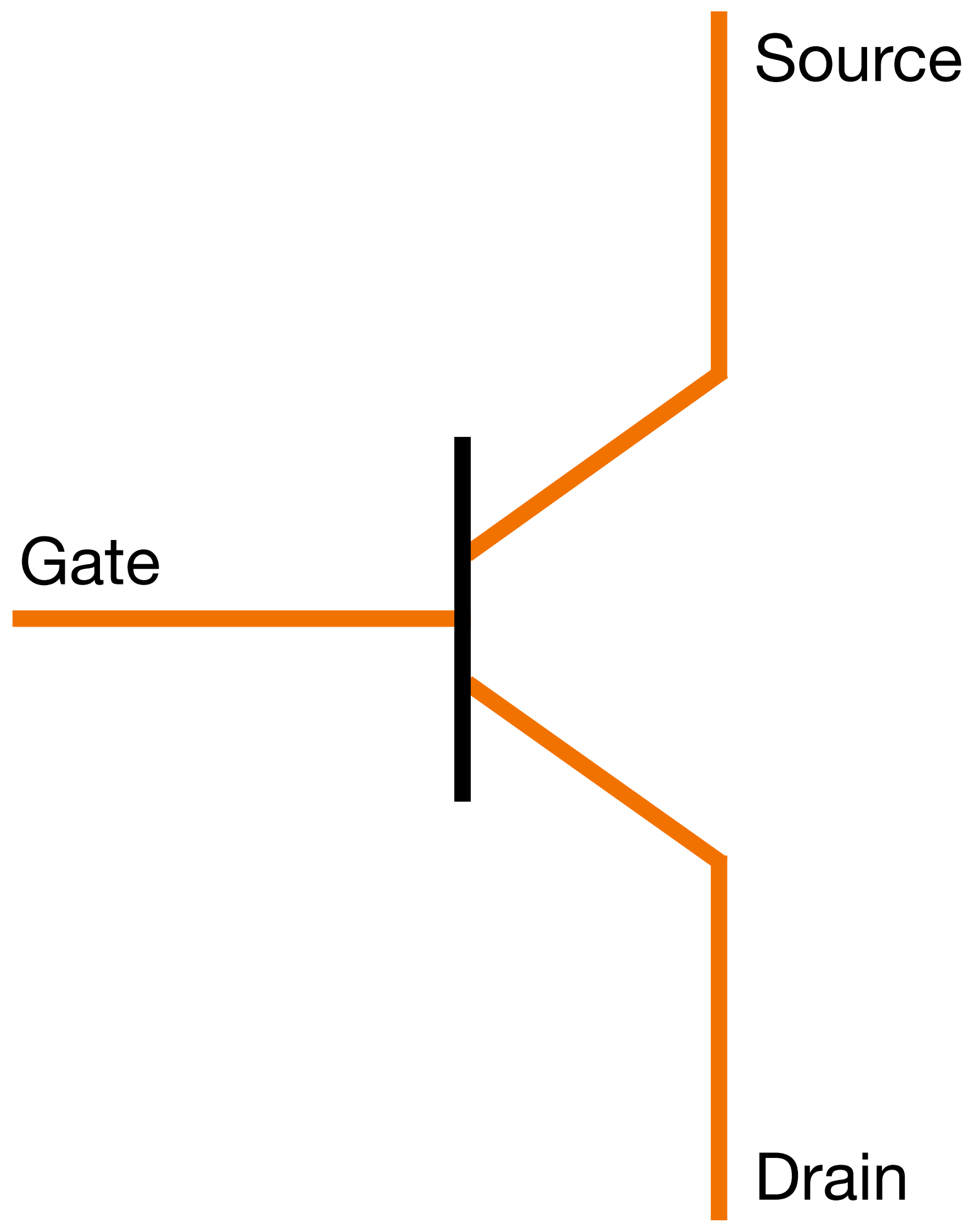
- Invented by John Bardeen, Walter Brattain, William Shockley in 1947
- Solid-state component
- Tiny -- <50 nanometers
- Really, really fast: MHz; really, really reliable











Transistors

- Relay switches -> Vacuum tubes -> Solid-state transistors
- We can turn electricity on/off with electricity
- We can do it really really fast
- How does it do computation?

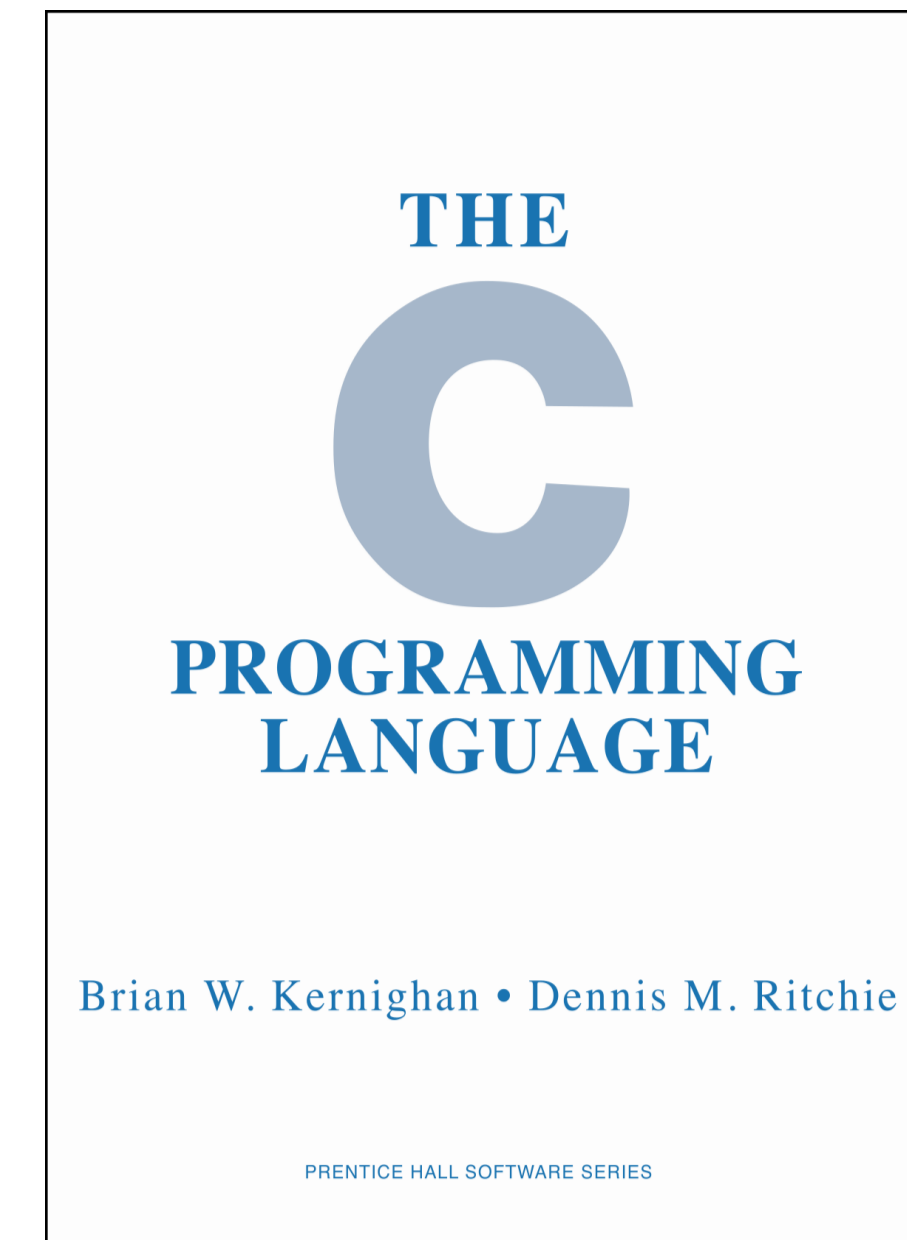
Information

Which one contains more information?

The Joseph Regenstein Library



This book:



Information

How about...

Eckhart Library



The John Crerar Library

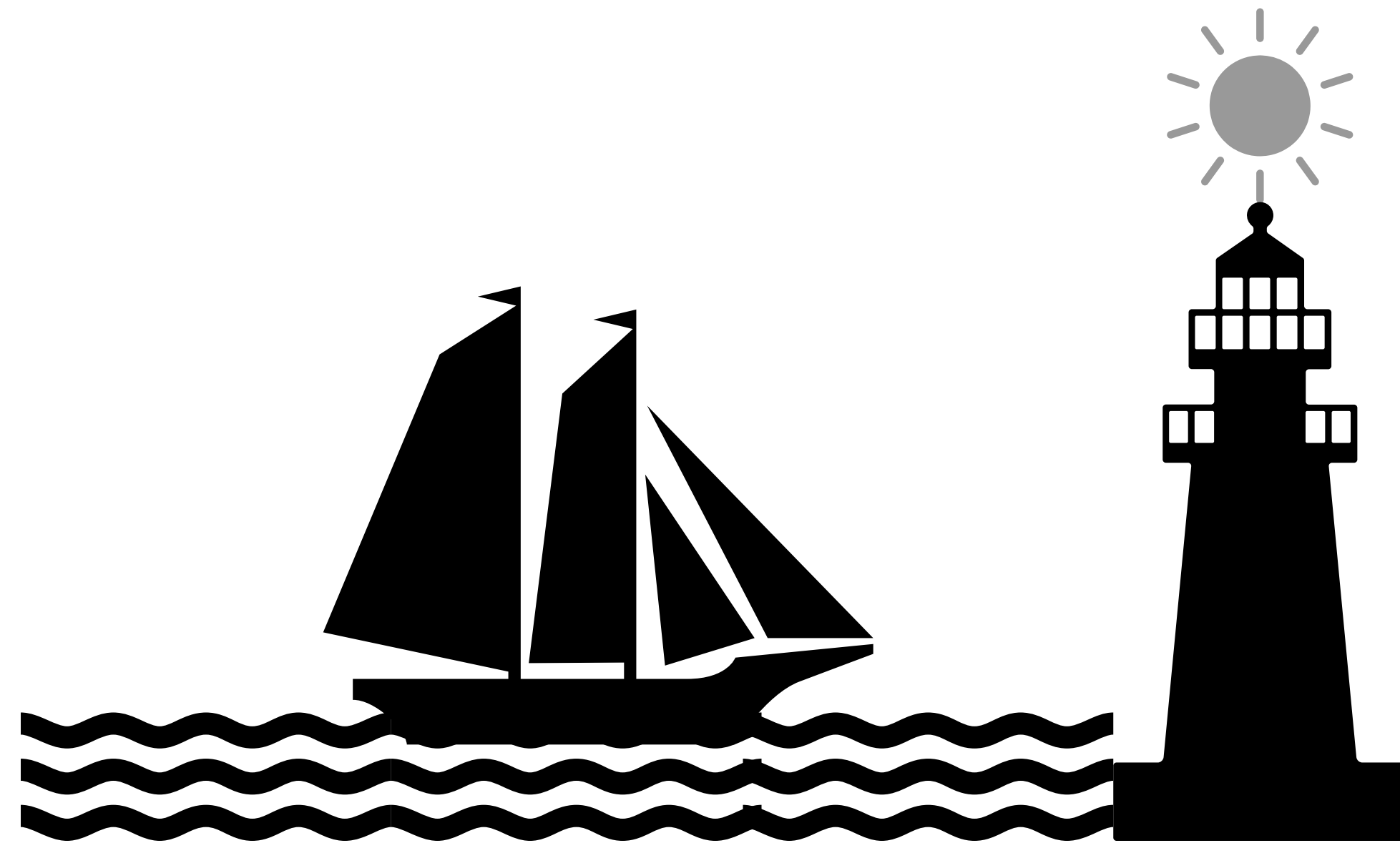


Information

- We can weigh objects
- We can determine the volume of solid objects or liquids
- We can measure the height of walls in this room
- Can we measure information?
- Can we distinguish more information from less?
- What is the unit of information?

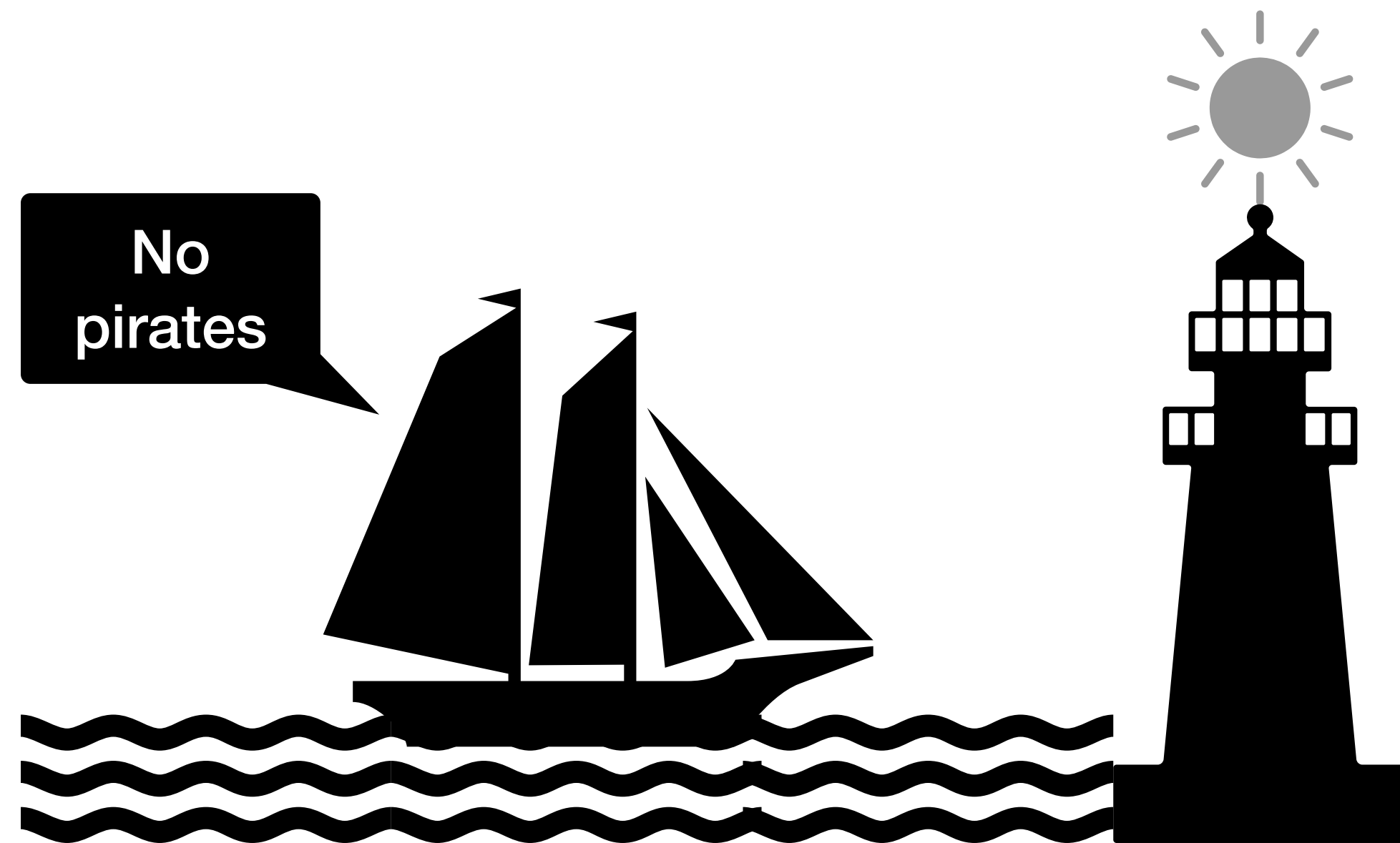
Information

- Example: a lighthouse operator and the captain wanted to communicate if there are pirates ahead. What is the shortest message the lighthouse can send?



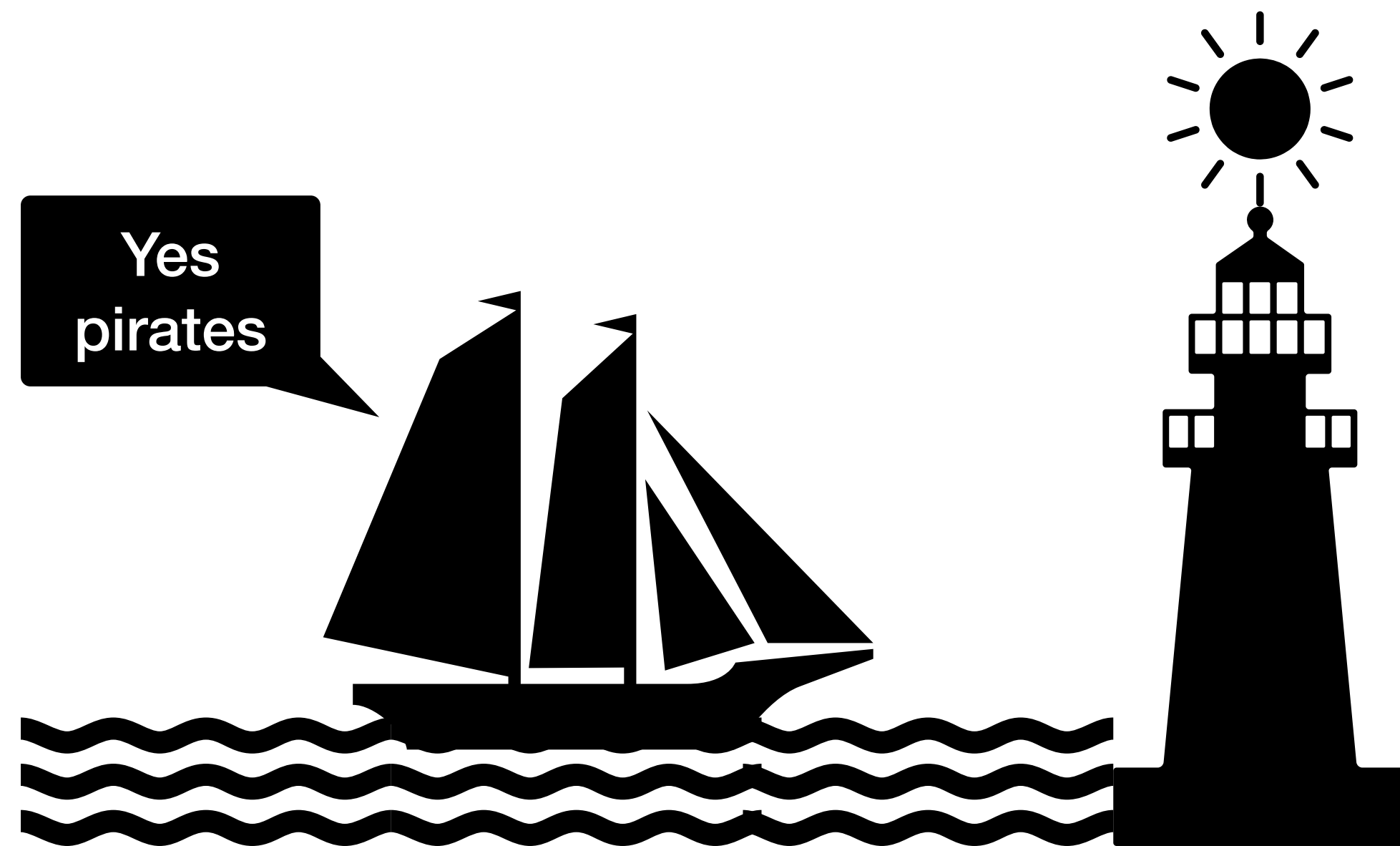
Information

- Example: a lighthouse operator and the captain wanted to communicate if there are pirates ahead. What is the shortest message the lighthouse can send?
- They agree ahead of time:
 - light off -> no pirates



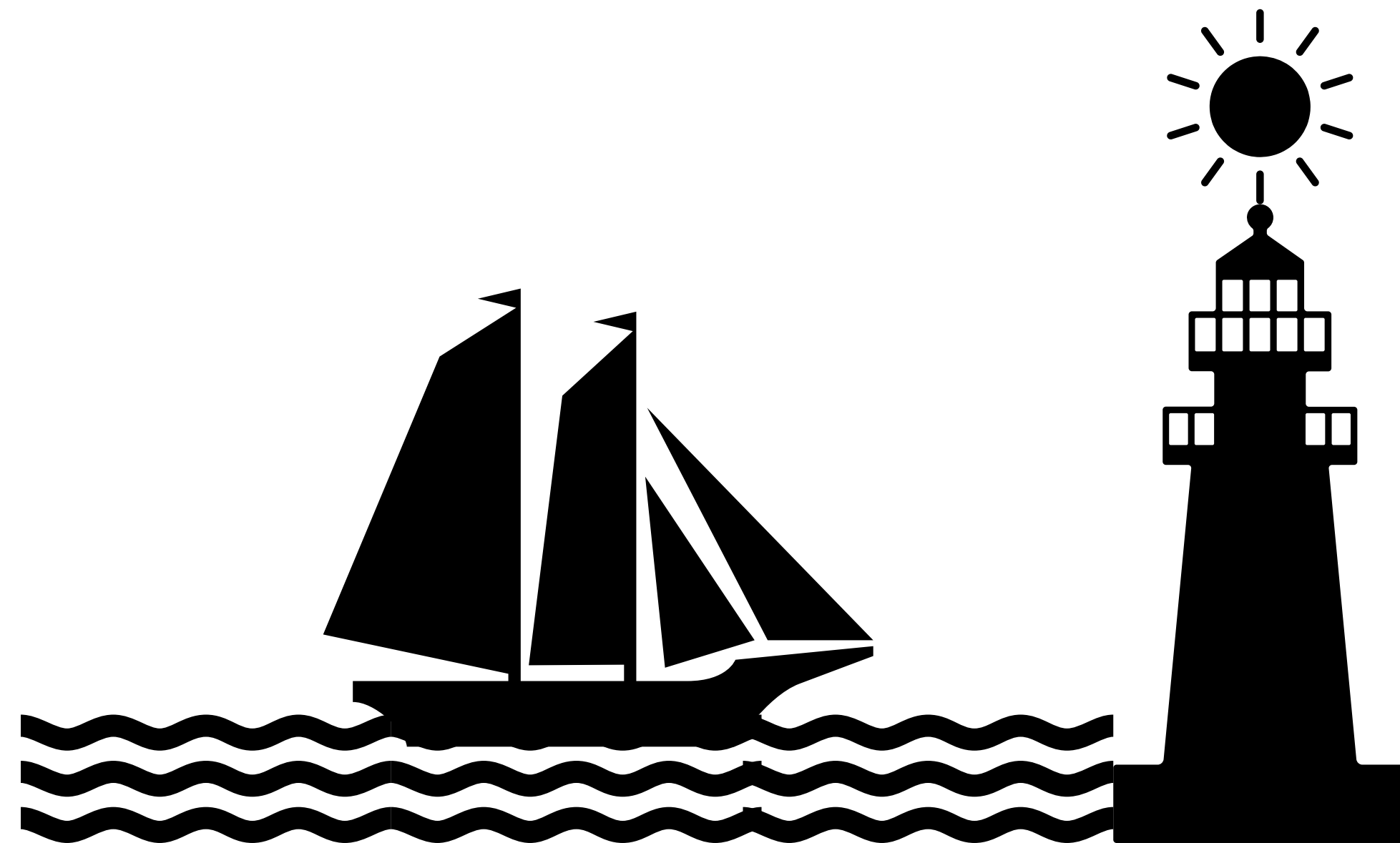
Information

- Example: a lighthouse operator and the captain wanted to communicate if there are pirates ahead. What is the shortest message the lighthouse can send?
- They agree ahead of time:
 - light off -> no pirates
 - light on -> yes pirates



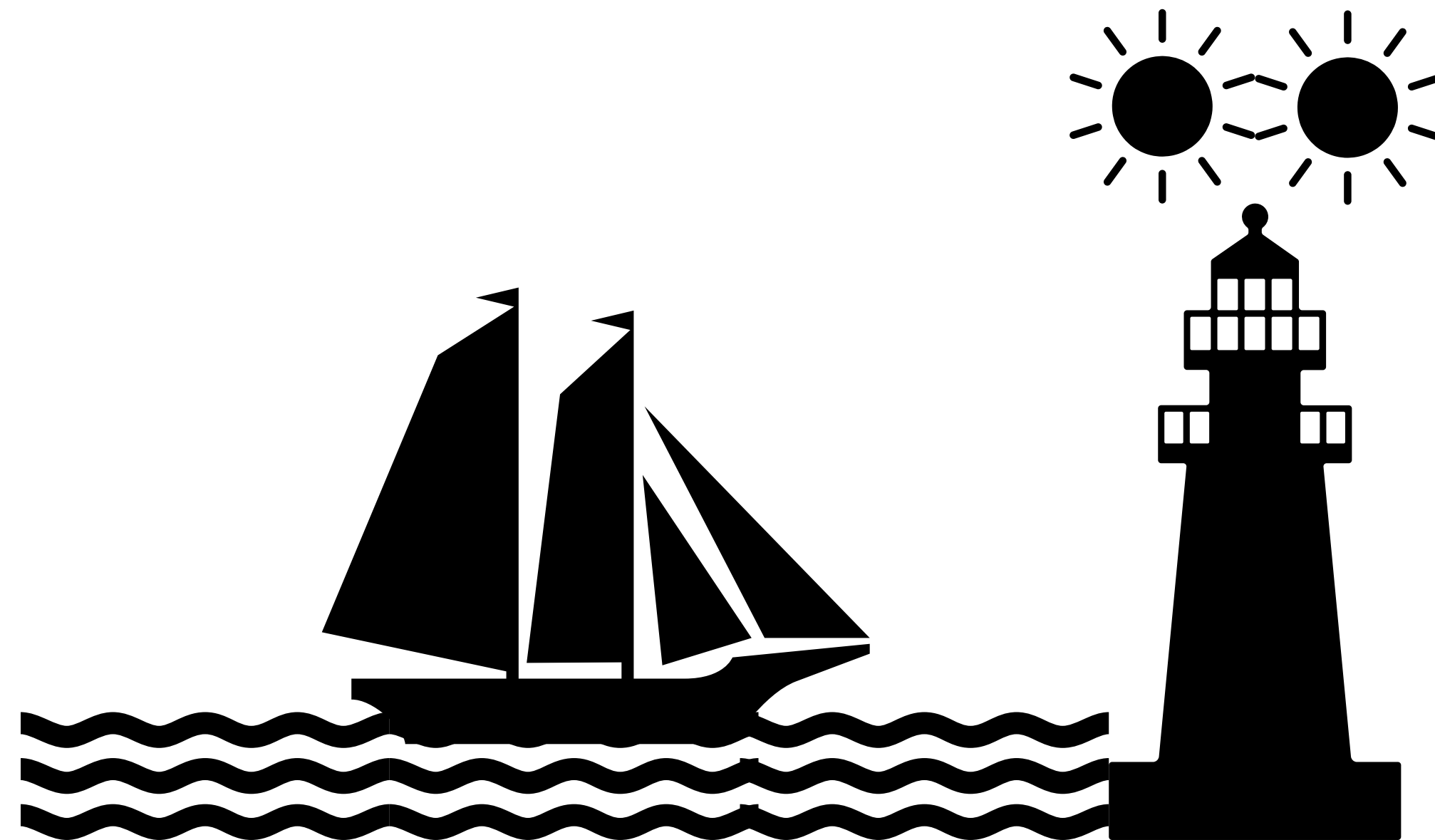
Information

- Example: a lighthouse operator and the captain wanted to communicate if there are pirates ahead *and if there is an iceberg.*



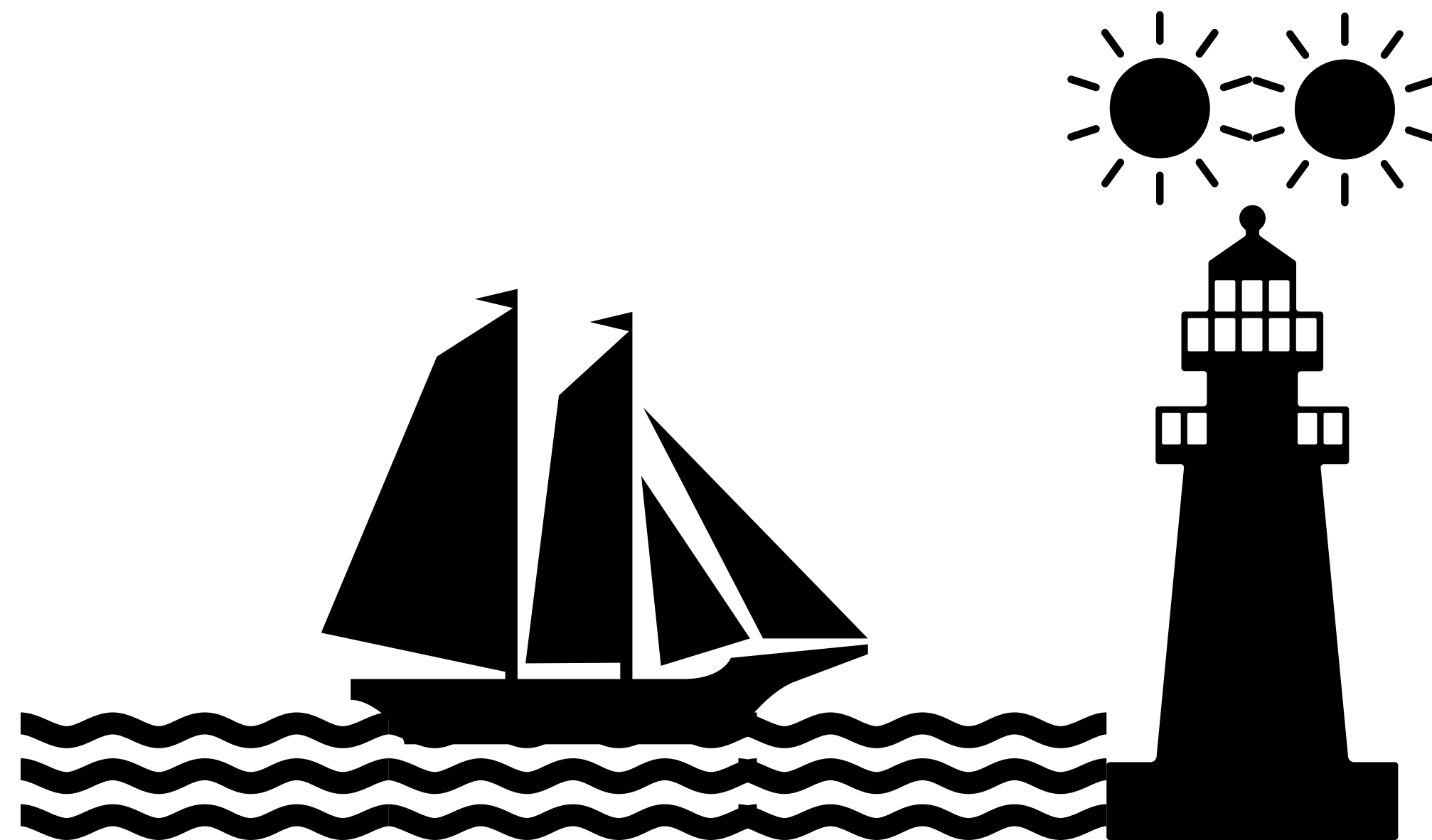
Information

- Example: a lighthouse operator and the captain wanted to communicate if there are pirates ahead *and if there is an iceberg.*



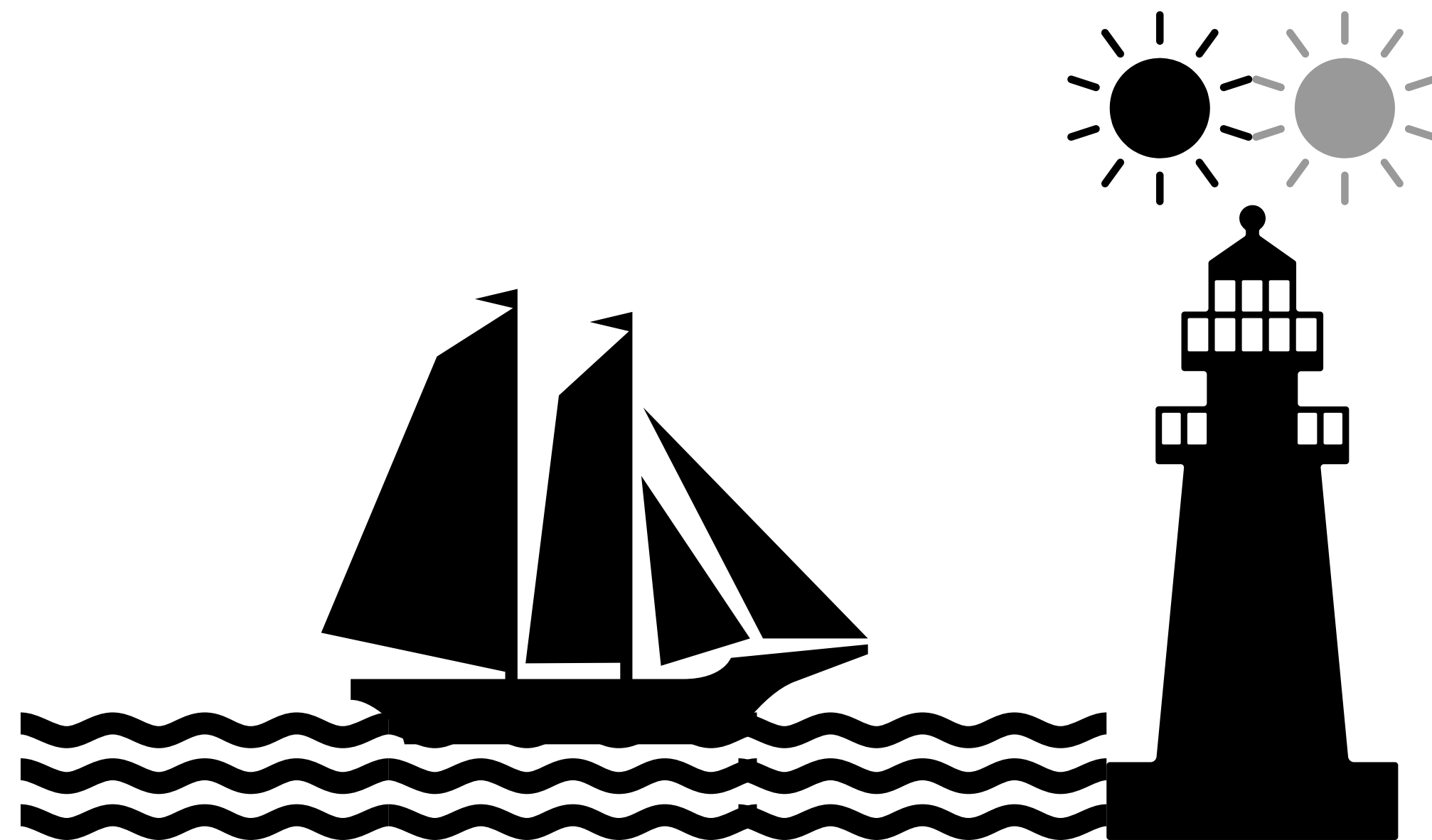
Information

- Example: a lighthouse operator and the captain wanted to communicate if there are pirates ahead *and if there is an iceberg*.
- Agree in advance:
 - On, On -> pirates and icebergs



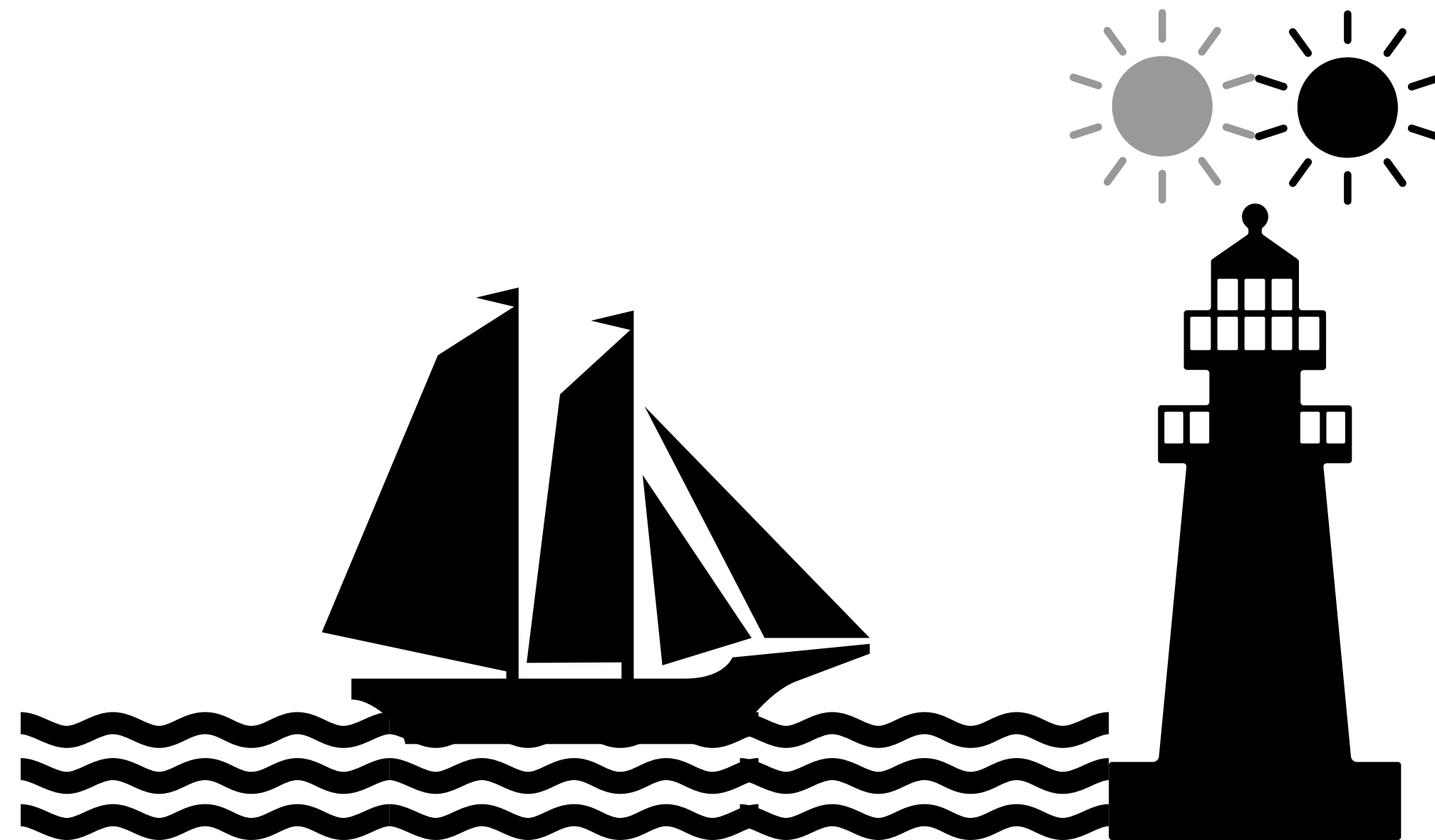
Information

- Example: a lighthouse operator and the captain wanted to communicate if there are pirates ahead *and if there is an iceberg*.
- Agree in advance:
 - On, On -> pirates and icebergs
 - On, Off -> only pirates



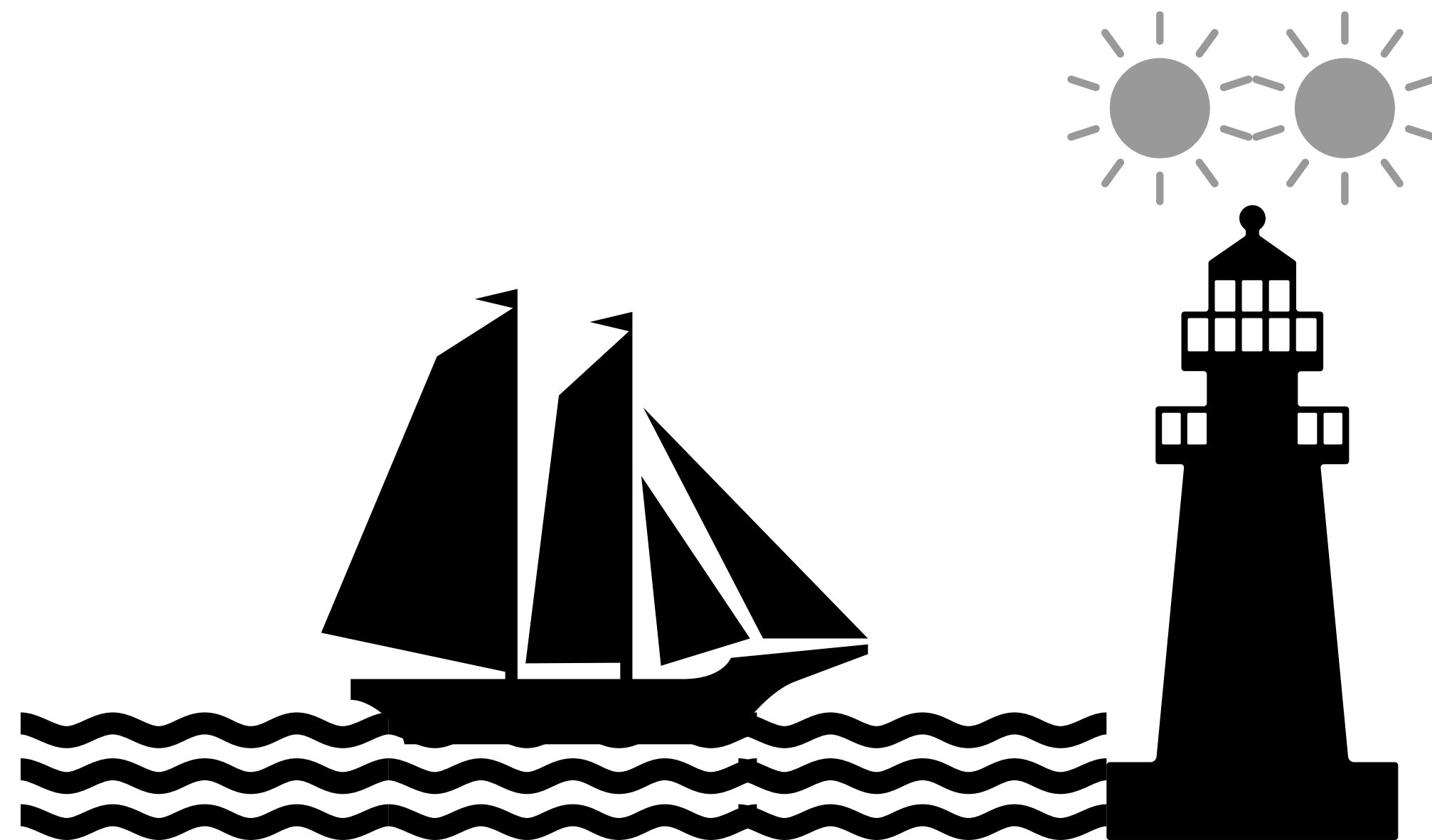
Information

- Example: a lighthouse operator and the captain wanted to communicate if there are pirates ahead *and if there is an iceberg*.
- Agree in advance:
 - On, On -> pirates and icebergs
 - On, Off -> only pirates
 - Off, On -> only iceberg

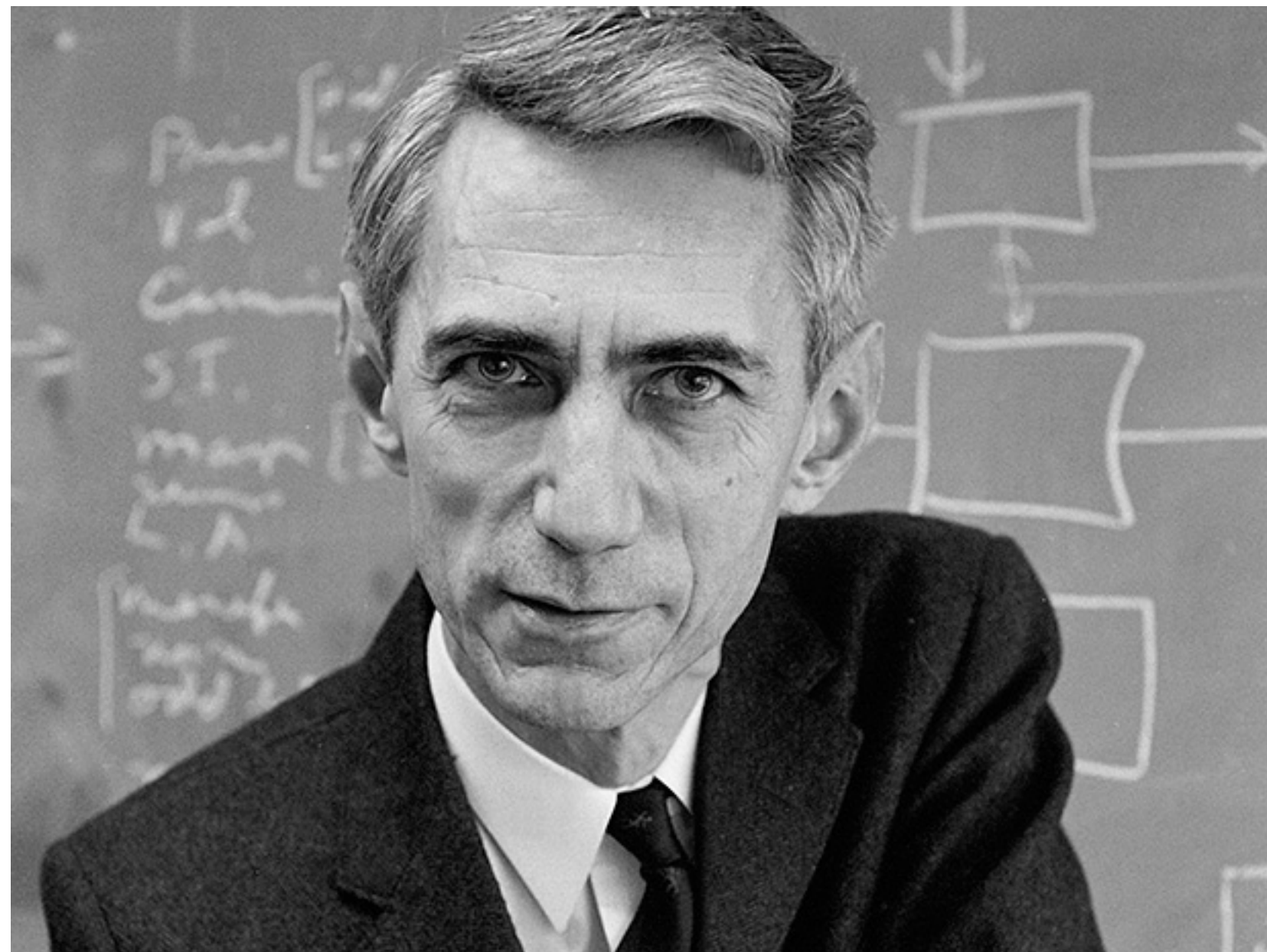


Information

- Example: a lighthouse operator and the captain wanted to communicate if there are pirates ahead *and if there is an iceberg*.
- Agree in advance:
 - On, On -> pirates and icebergs
 - On, Off -> only pirates
 - Off, On -> only iceberg
 - Off, Off -> neither



Information



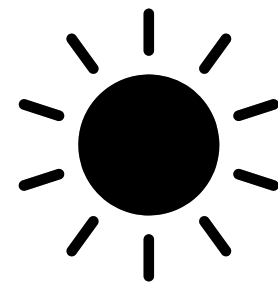
- Claude Shannon (1916 - 2001):
- *A Mathematical Theory of Communication* (1948)

Information

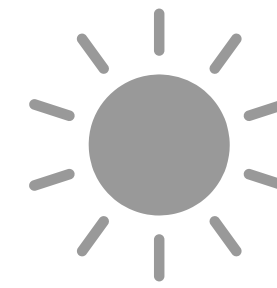
- Insight: whenever two parties communicate, each message is answering one or more *yes-or-no* questions.
- We call each answer *a bit*.
- Information can be measured in bits.
- "A Symbolic Analysis of Relay and Switching Circuits." Shannon 1938.
- In computers, "on," when the voltage is *high*, represents "yes" or "true;" "off," when the voltage is *low*, represents "no" or "false."
- This is called binary.

Information

Why "yes" or "no"?



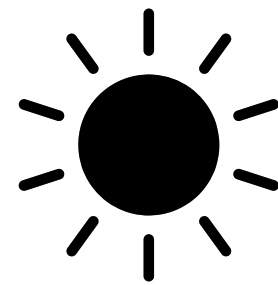
Yes



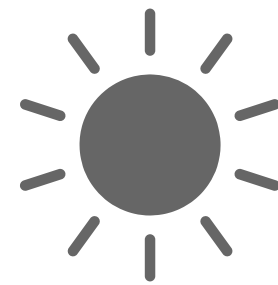
No

Information

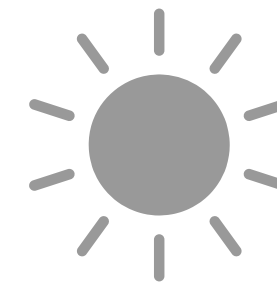
Why "yes" or "no"?



Yes



Maybe?

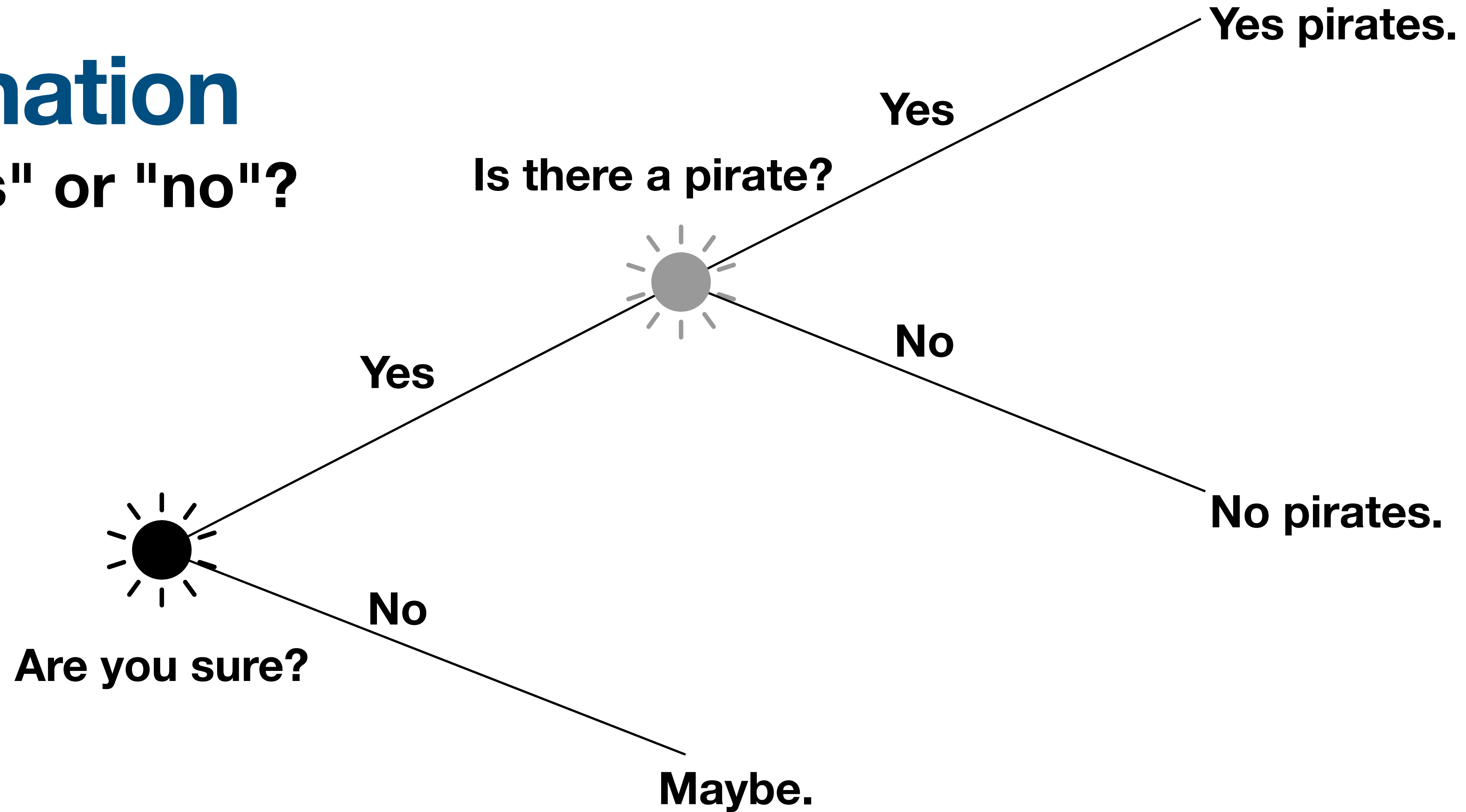


No

- CS answer: "yes" or "no" is the most reduced form.

Information

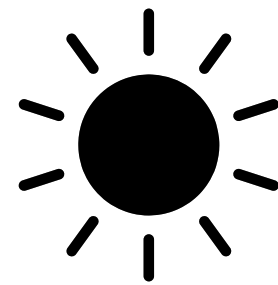
Why "yes" or "no"?



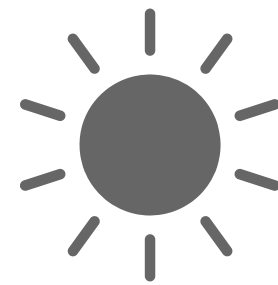
- CS answer: "yes" or "no" is the most reduced form.

Information

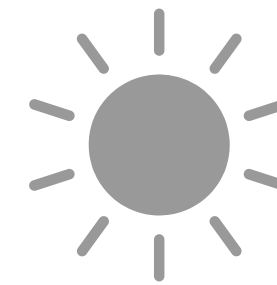
Why "yes" or "no"?



Yes



Maybe?

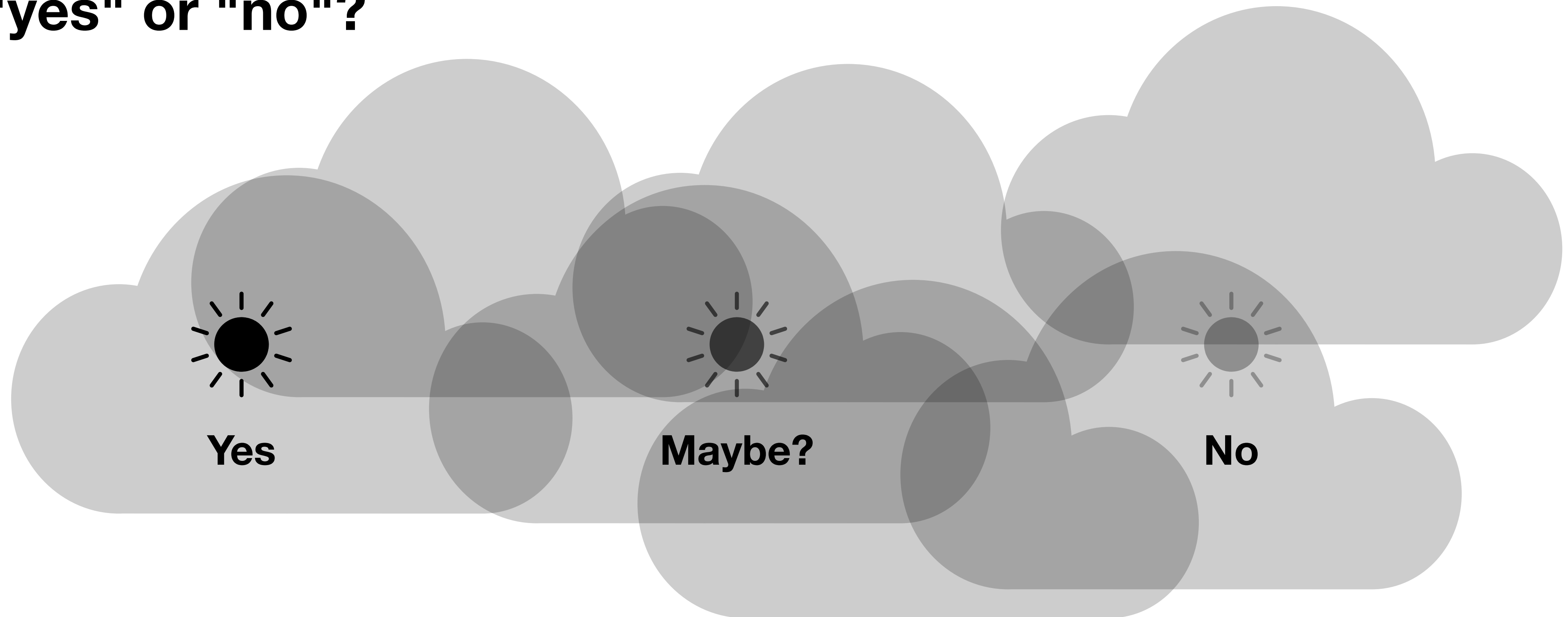


No

- EE answer: the two options can survive a lot of noise.

Information

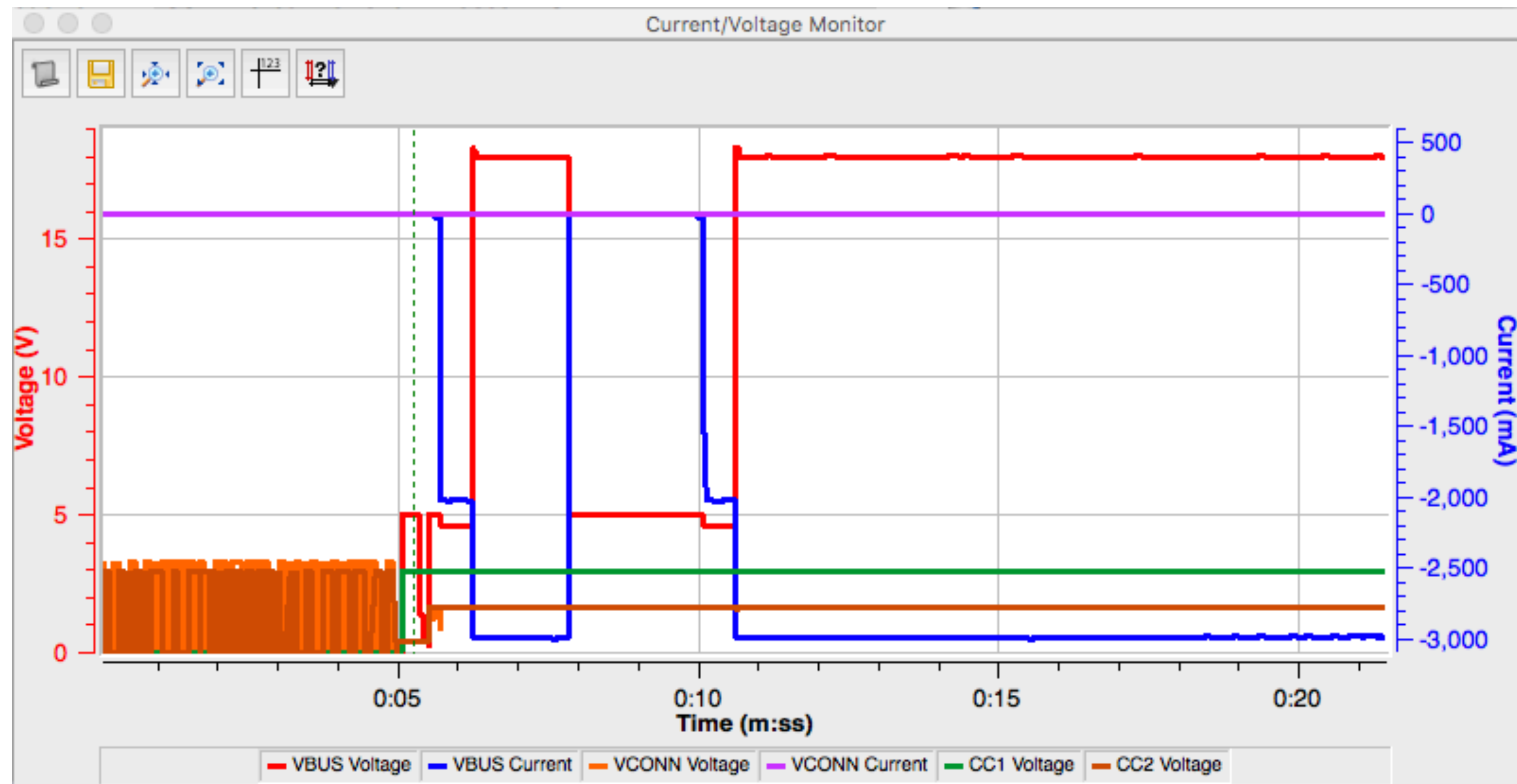
Why "yes" or "no"?



- EE answer: the two options can survive a lot of noise.

Information

Why "yes" or "no"?



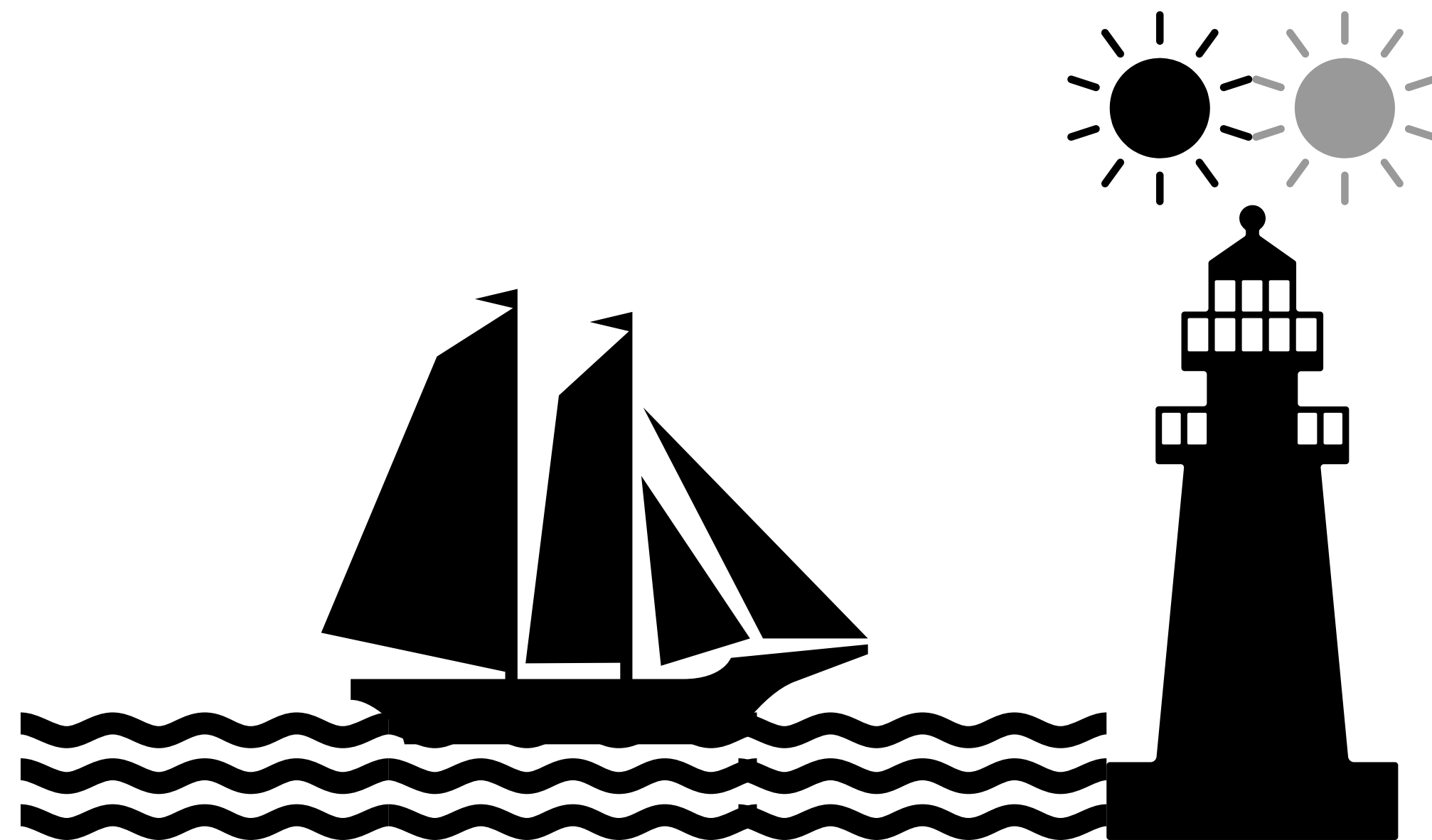
- EE answer: the two options can survive a lot of noise.

Information

- Just saying "yes" or "no" isn't enough
 - We don't know what *questions* they answer
- We have to agree ahead of time what the choices are.
- The more choices we have to make, the more answers we'll have to communicate.

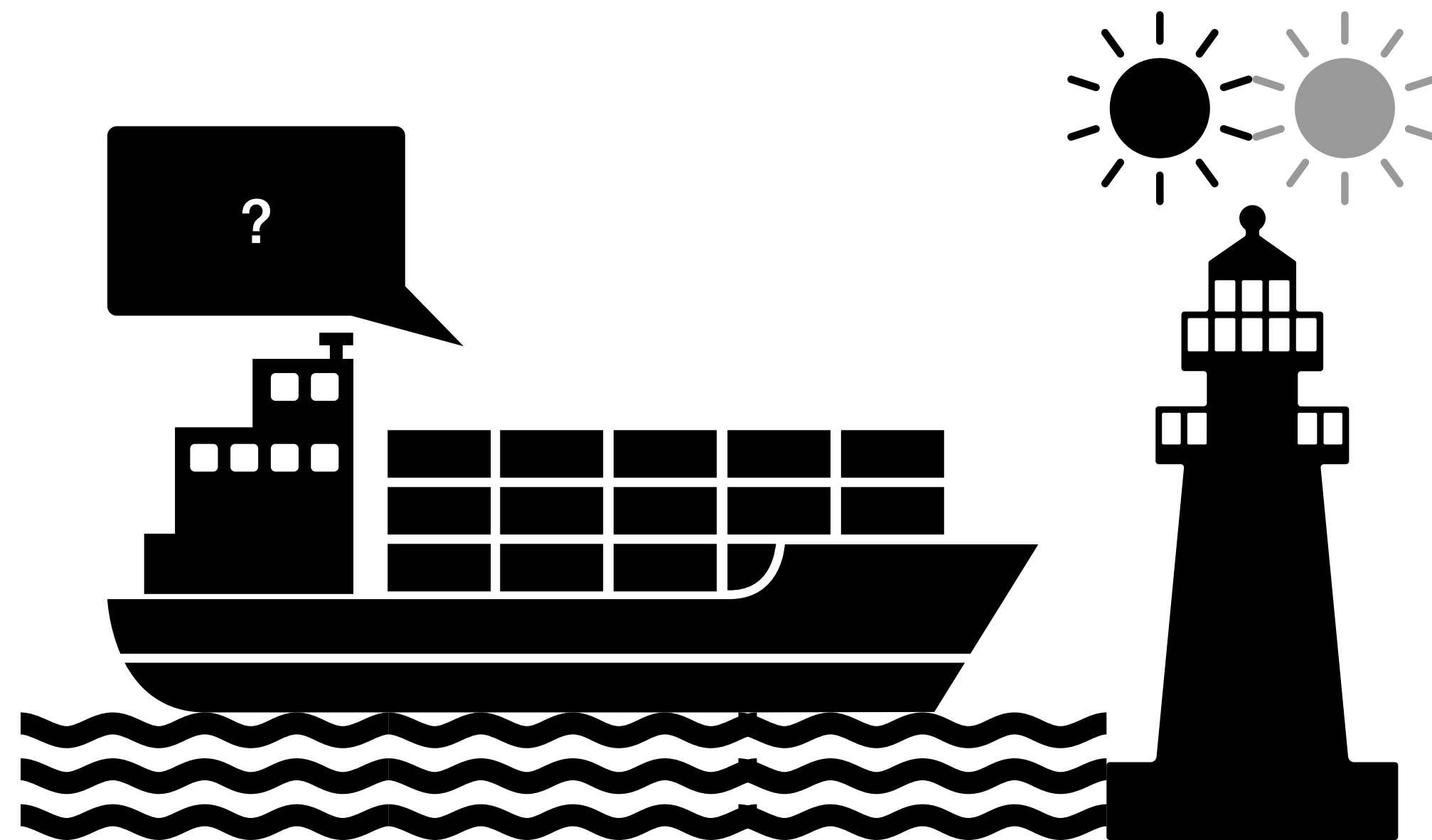
Information Protocol

- The captain and the lighthouse *must* agree on the choices in advance.
 - How many choices?
 - Which light answers which?



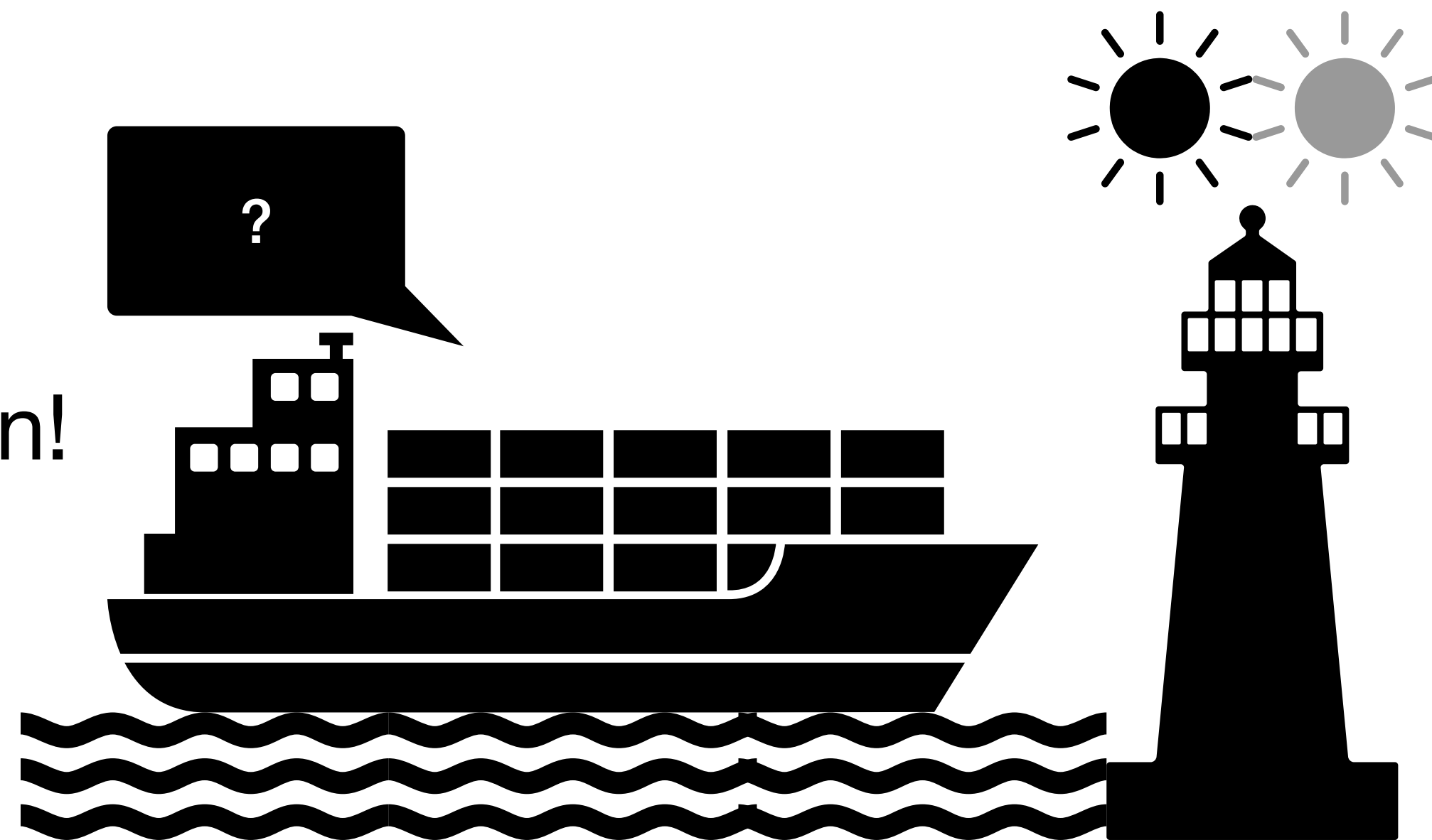
Information Protocol

- The captain and the lighthouse *must* agree on the choices in advance.
 - How many choices?
 - Which light answers which?



Information Protocol

- The captain and the lighthouse *must* agree on the choices in advance.
 - How many choices?
 - Which light answers which?
- If you don't know the context, you don't know what the bits mean!
- This is why we need file format, types, protocols, ...



Information

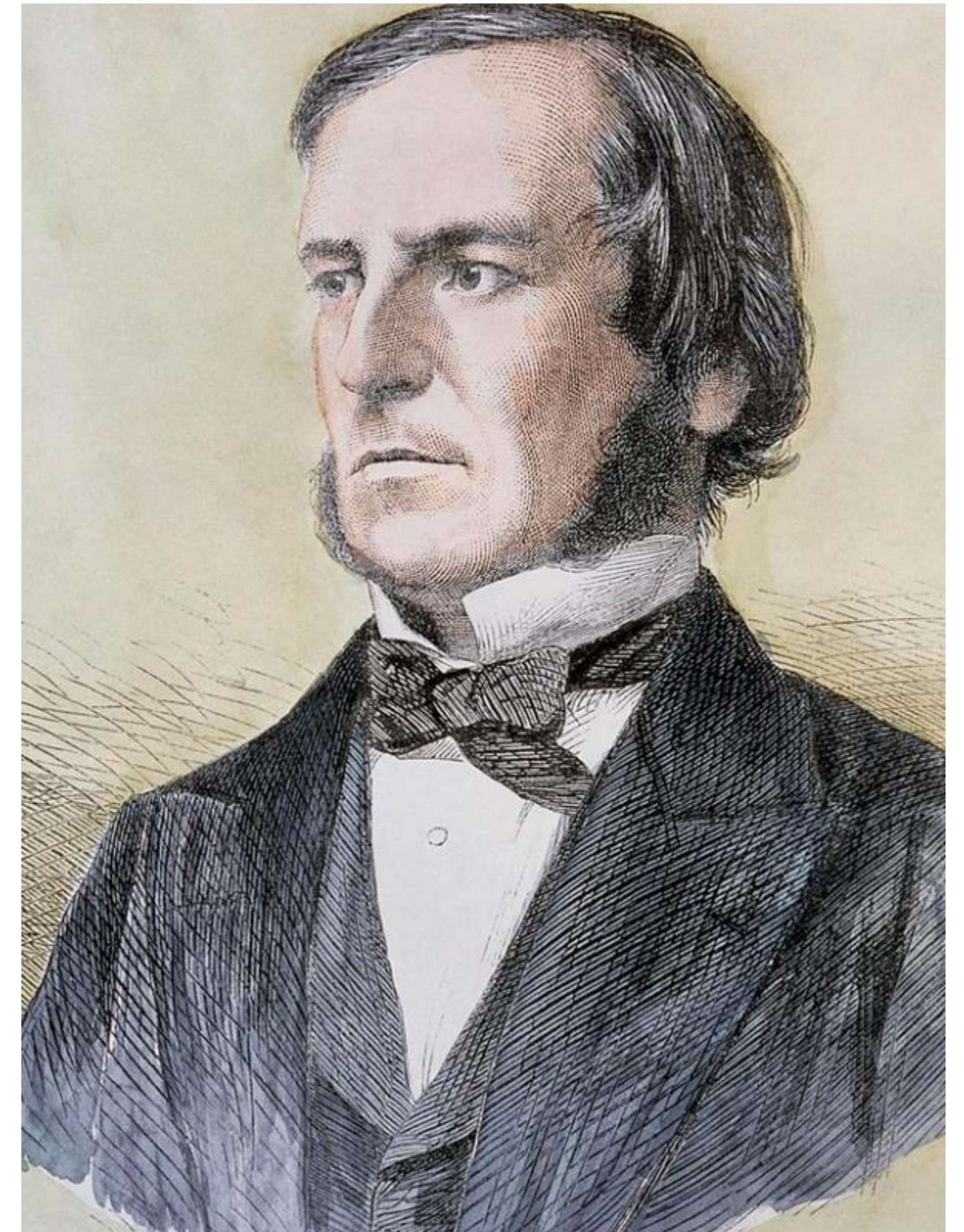
Bits

- A bit is an answer to a yes-no question
- We can model information as a series of yes-no questions
- In order to communicate in bits, we must know:
 - what the questions are;
 - which bit answers which question.
- If we have n bits, we have 2^n choices to communicate.
- If we have n choices to communicate, we need $\lceil \log_2 n \rceil$ bits.

Information

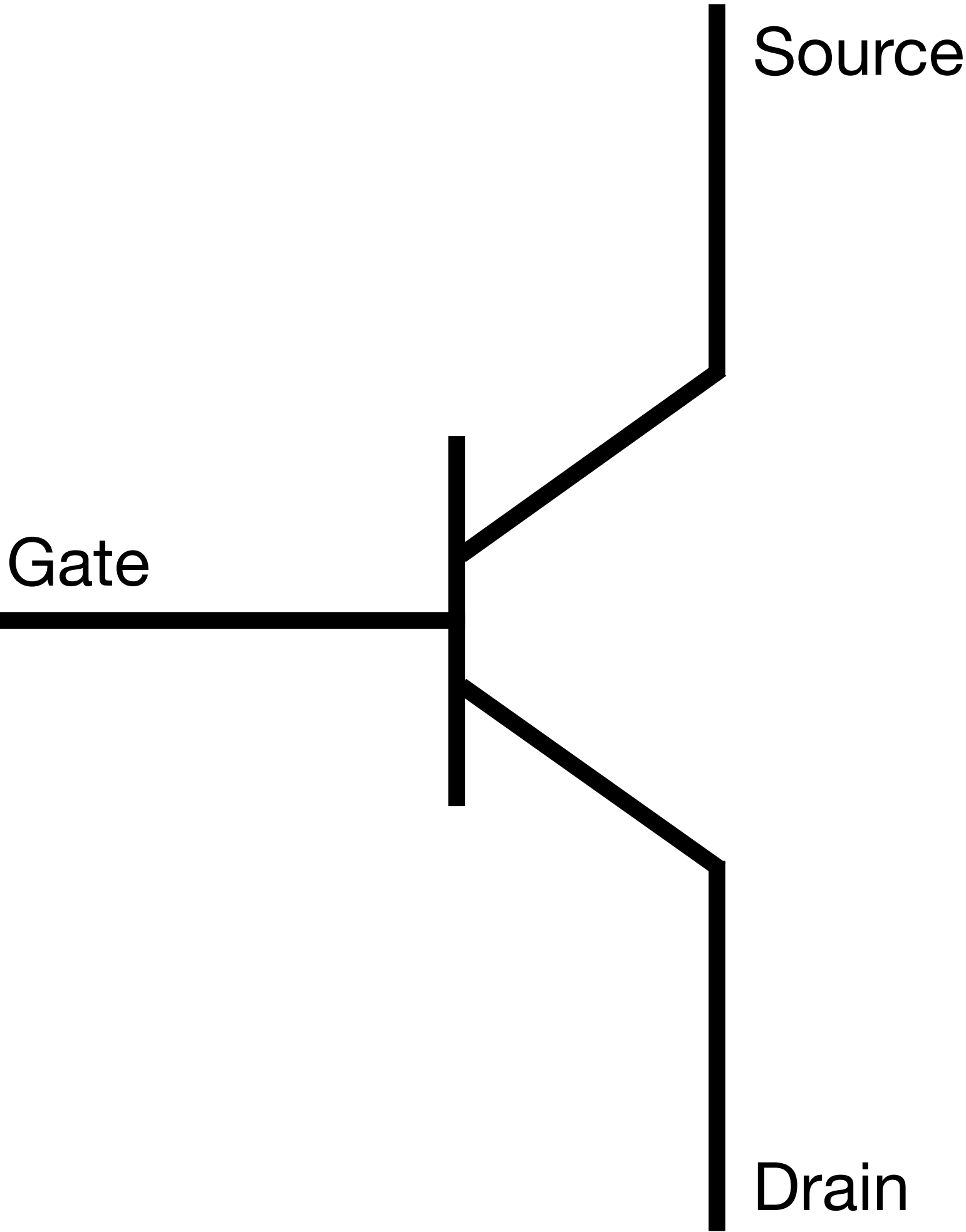
Bits

- George Boole (1814-1864) (49 years)
- "The Mathematical Analysis of Logic." (1847)
- A systematic processing of truth values: Boolean algebra:
 - True and False
 - And, Or, Not
- Transistors can be used to implement Boolean algebra!



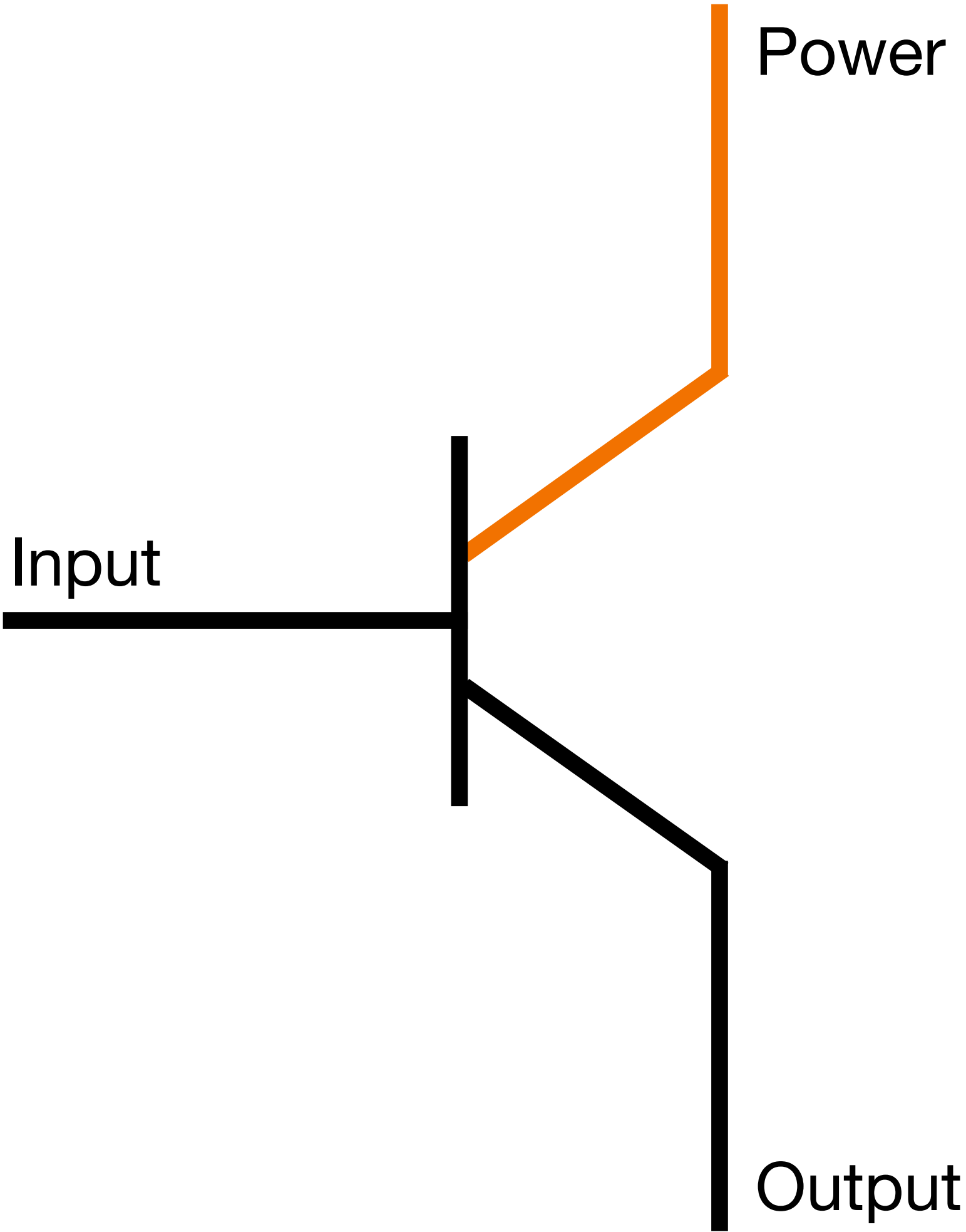
Not

Input	Output
TRUE	FALSE
FALSE	TRUE



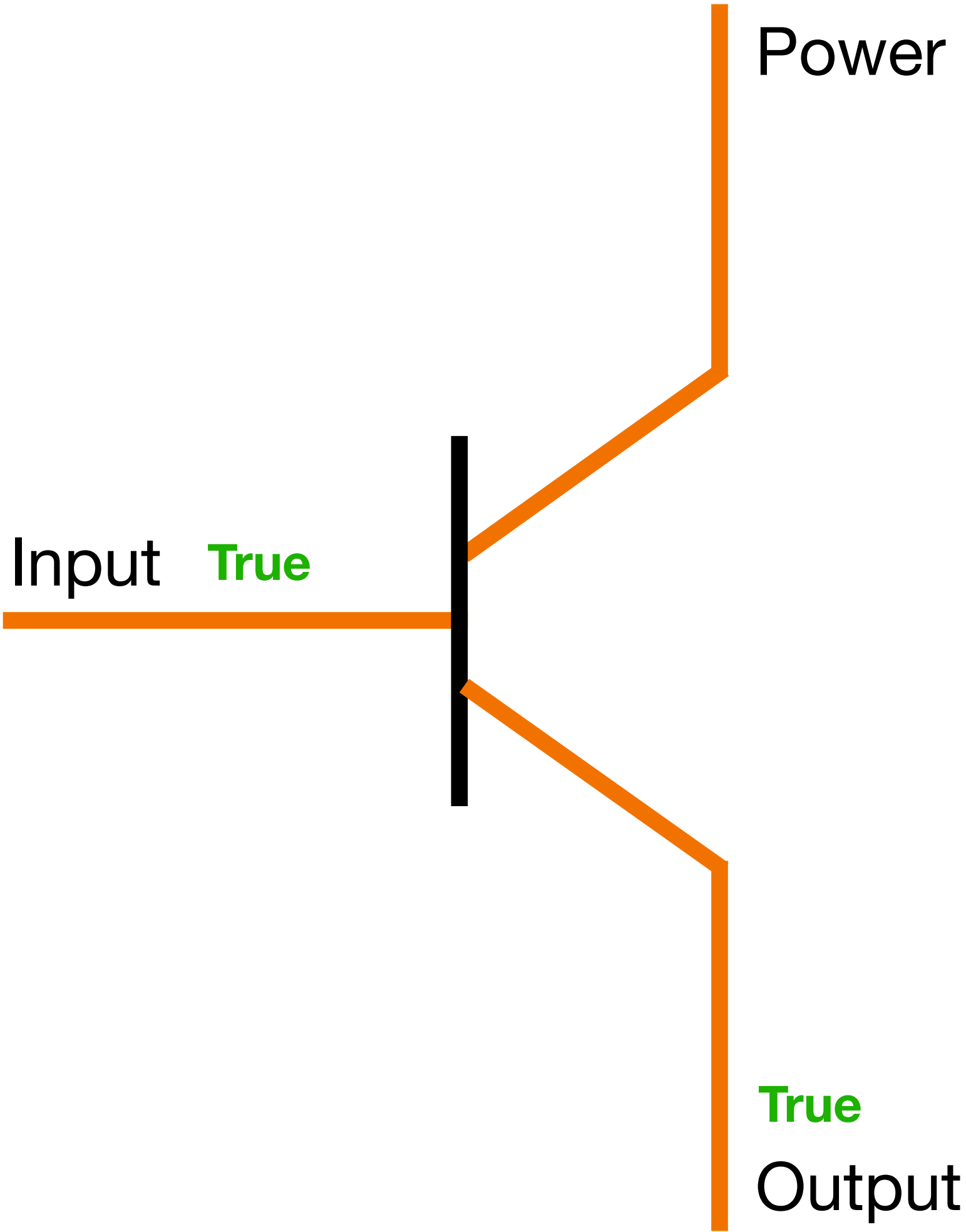
Not

Input	Output
TRUE	FALSE
FALSE	TRUE



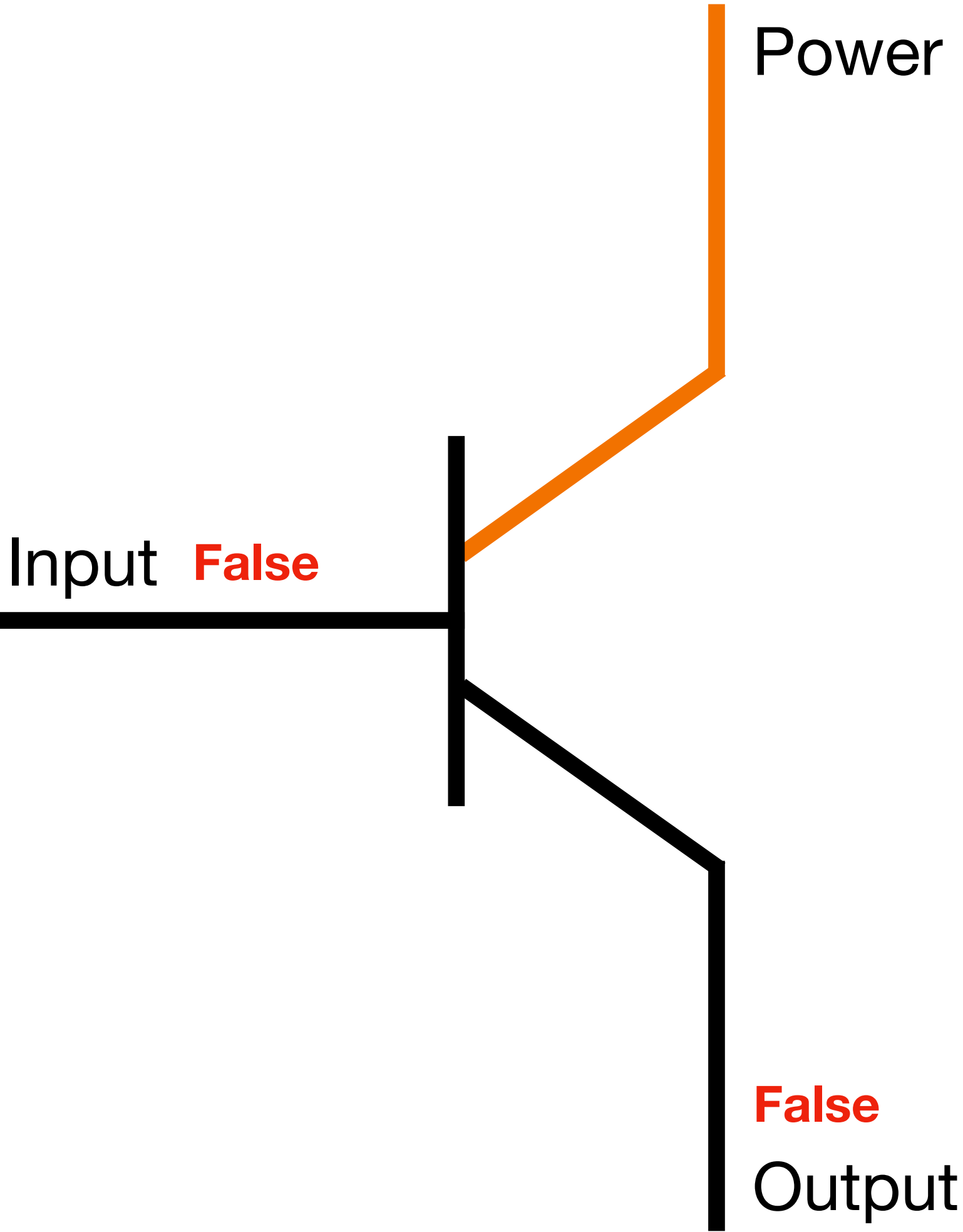
Not

Input	Output
TRUE	FALSE
FALSE	TRUE



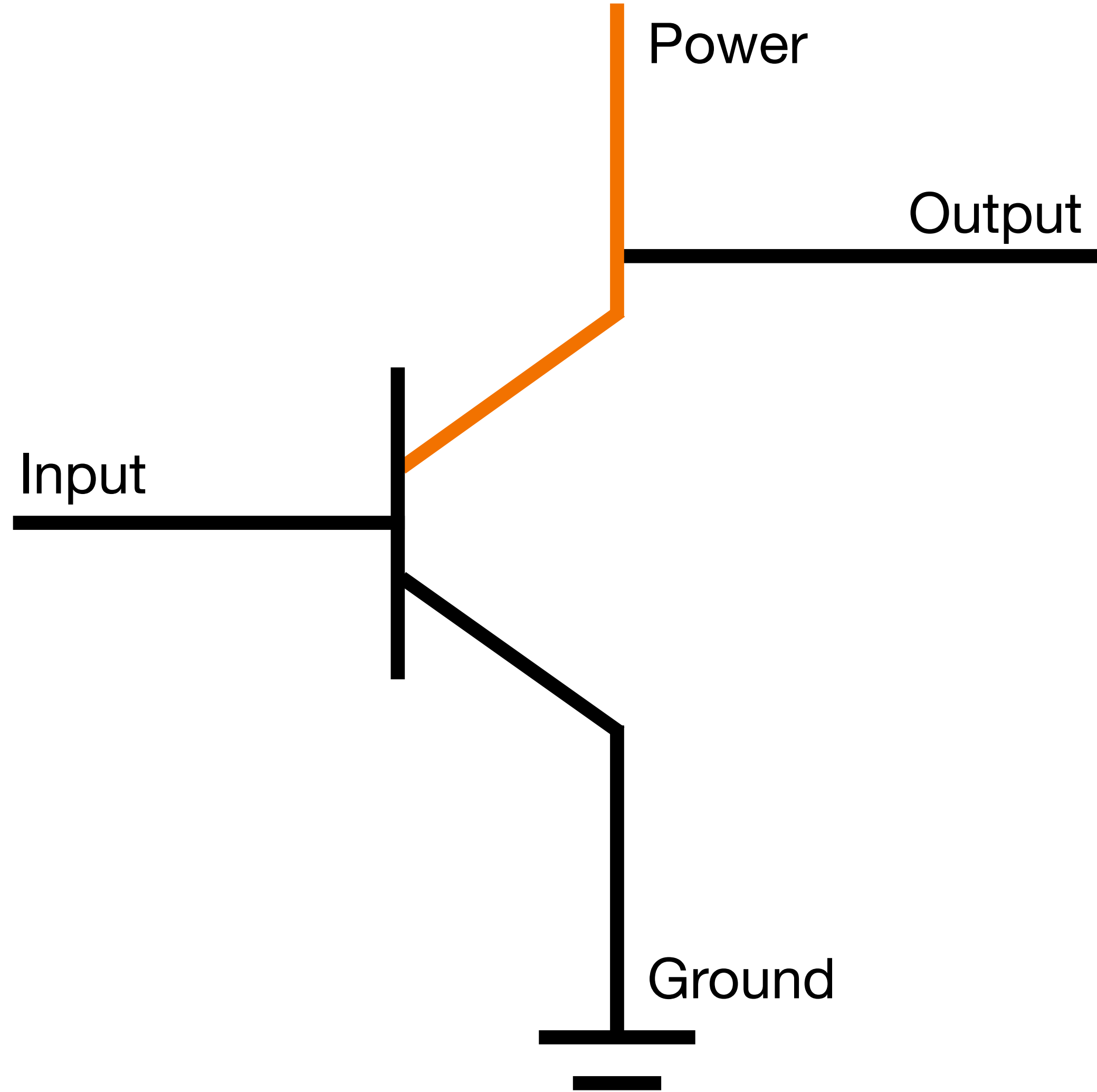
Not

Input	Output
TRUE	FALSE
FALSE	TRUE



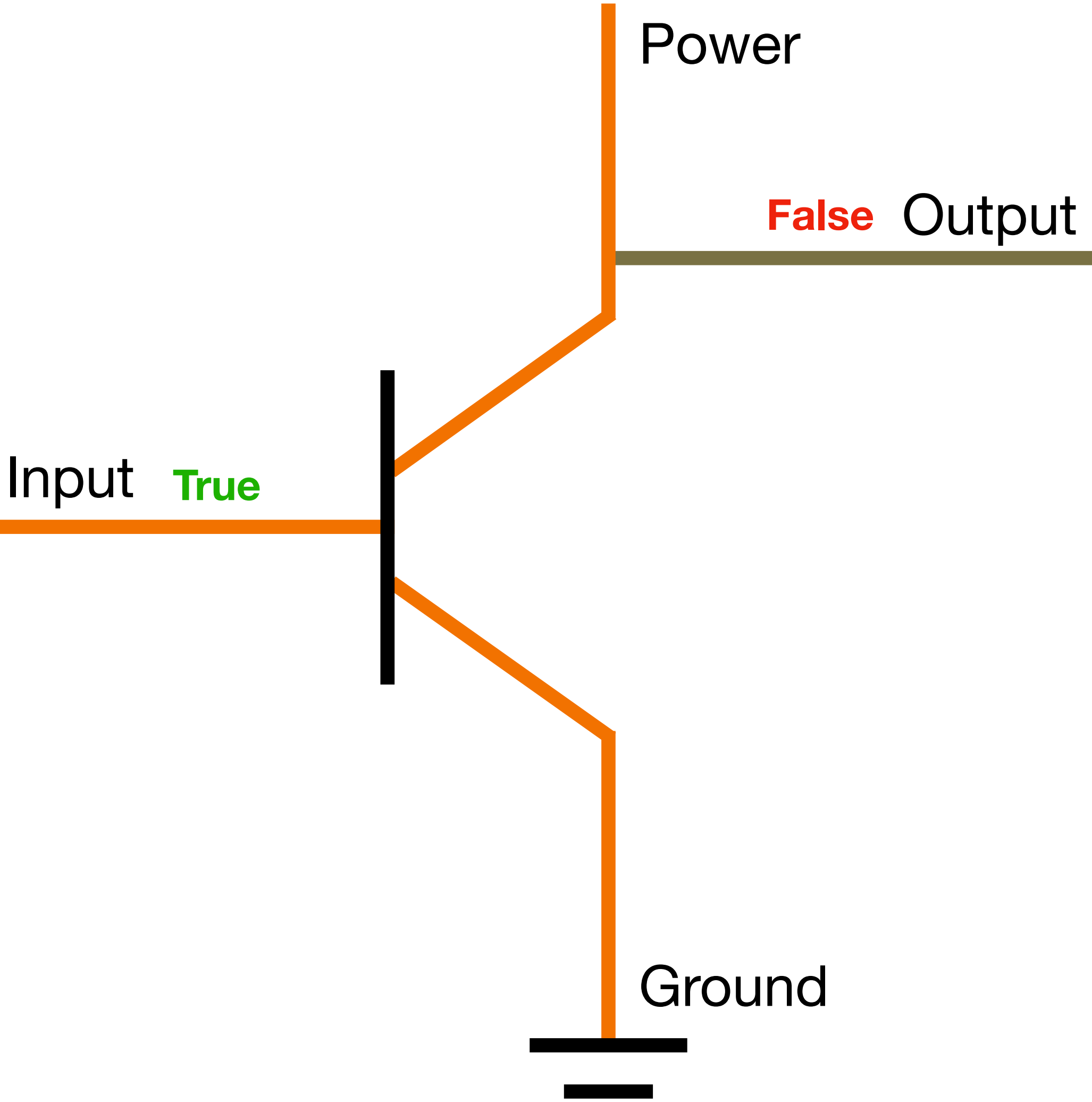
Not

Input	Output
TRUE	FALSE
FALSE	TRUE



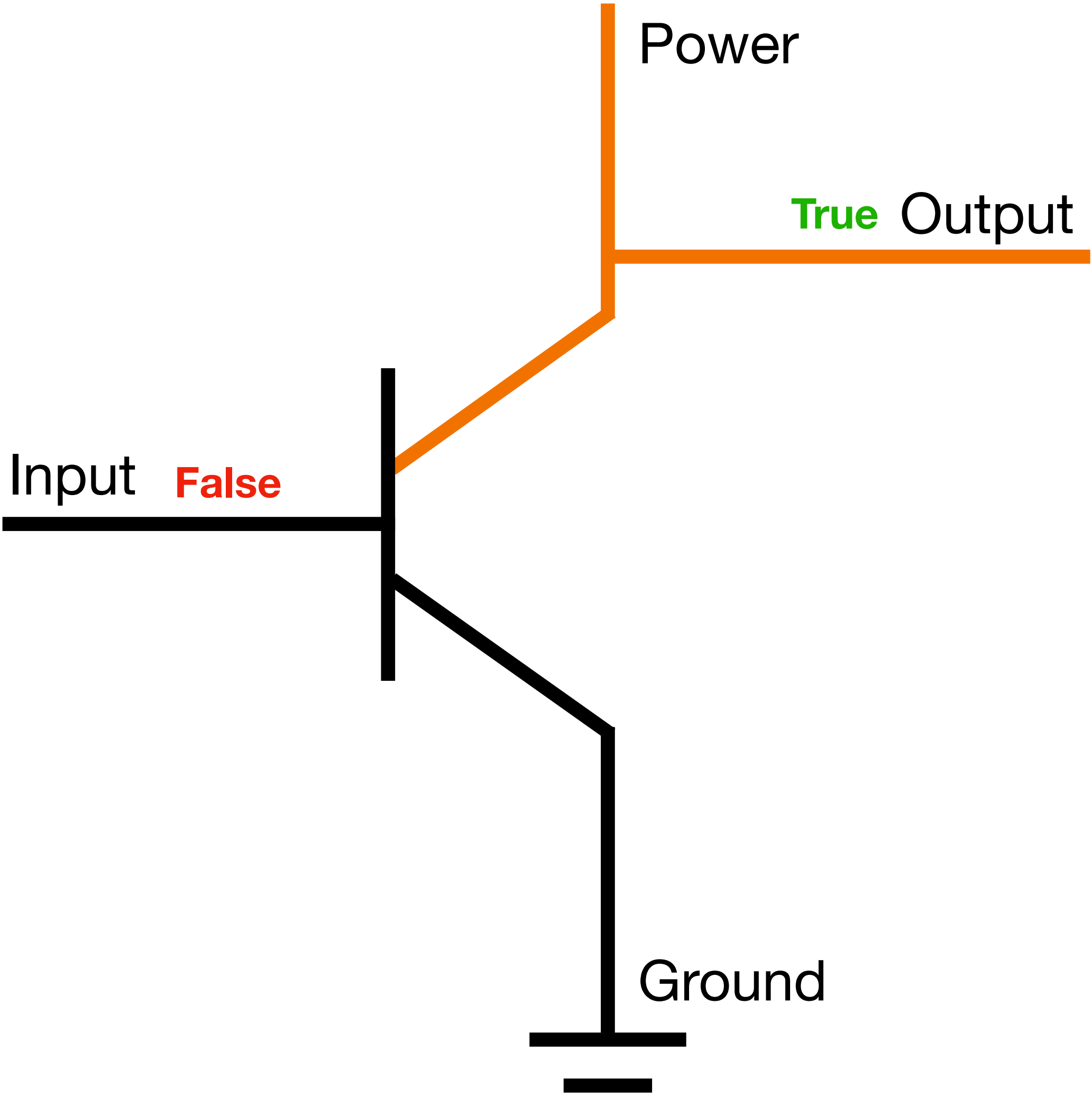
Not

Input	Output
TRUE	FALSE
FALSE	TRUE



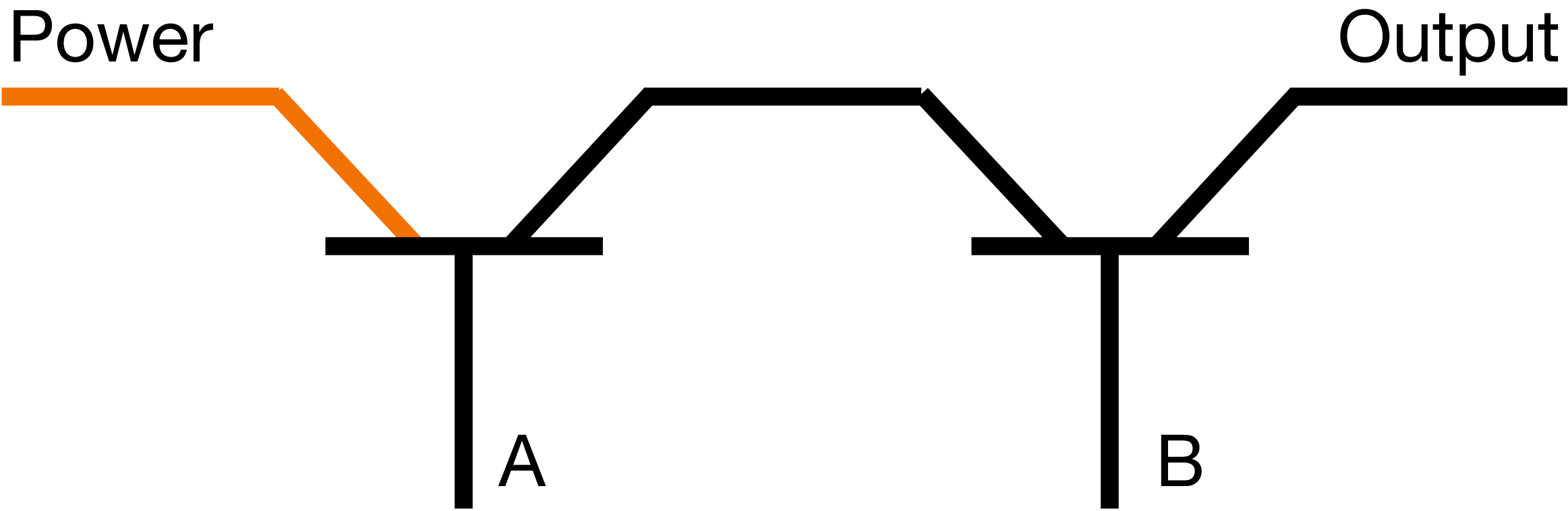
Not

Input	Output
TRUE	FALSE
FALSE	TRUE



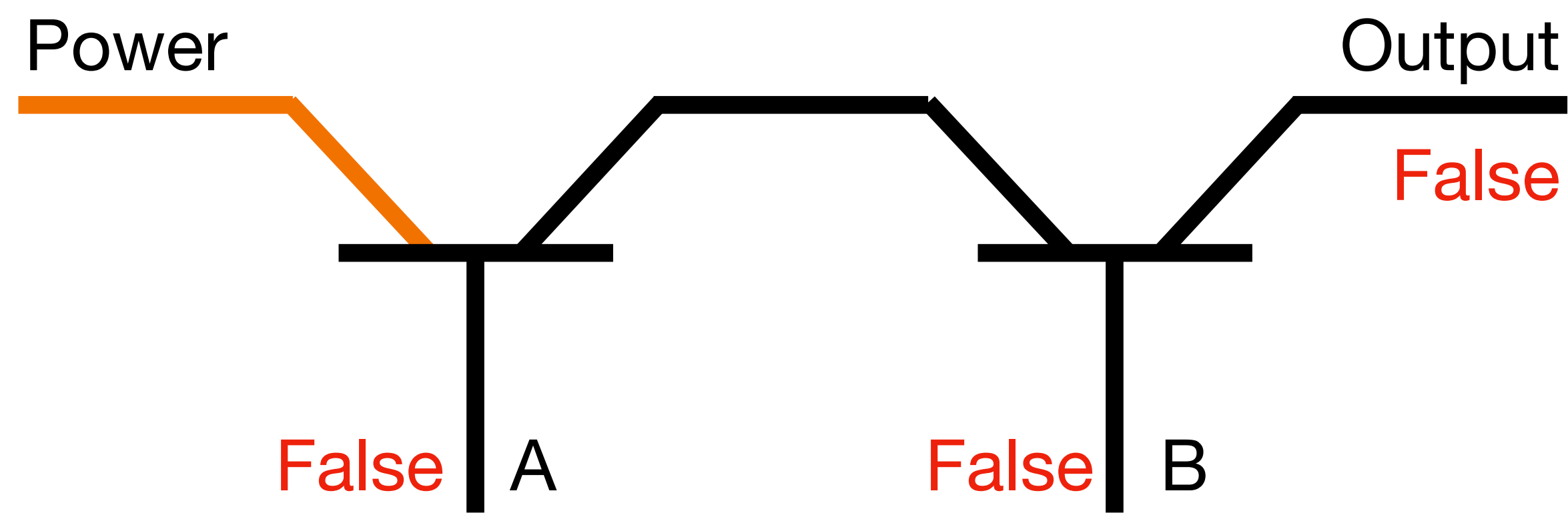
And

A	B	Output
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
FALSE	FALSE	FALSE



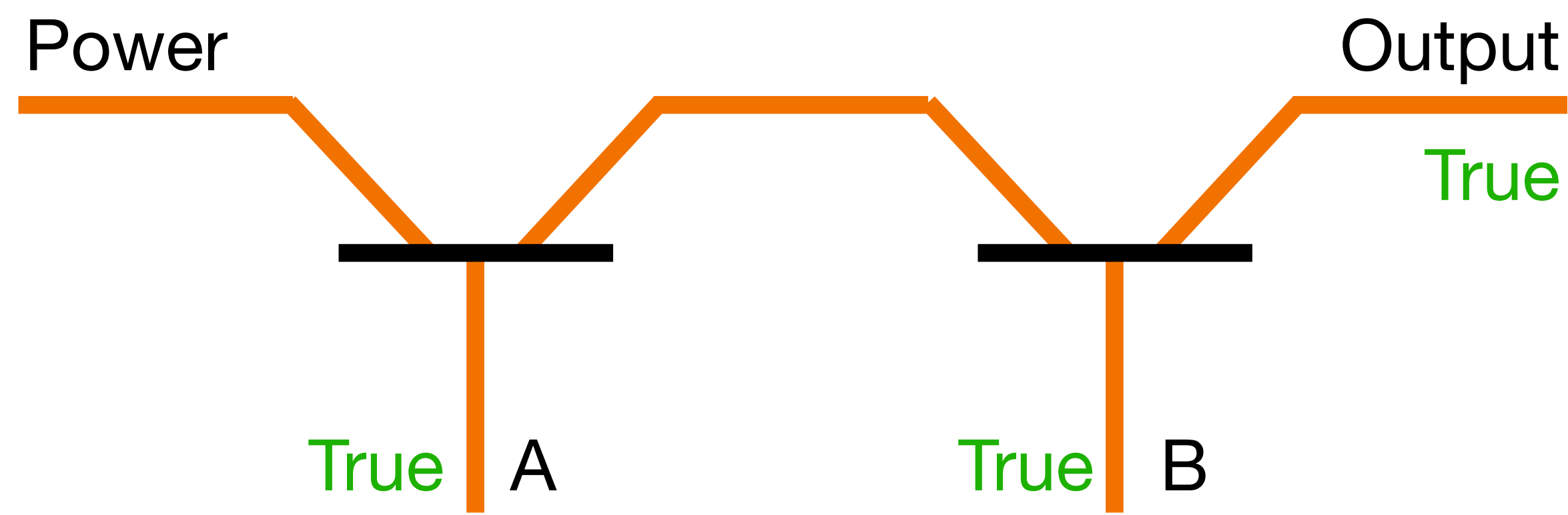
And

A	B	Output
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
FALSE	FALSE	FALSE



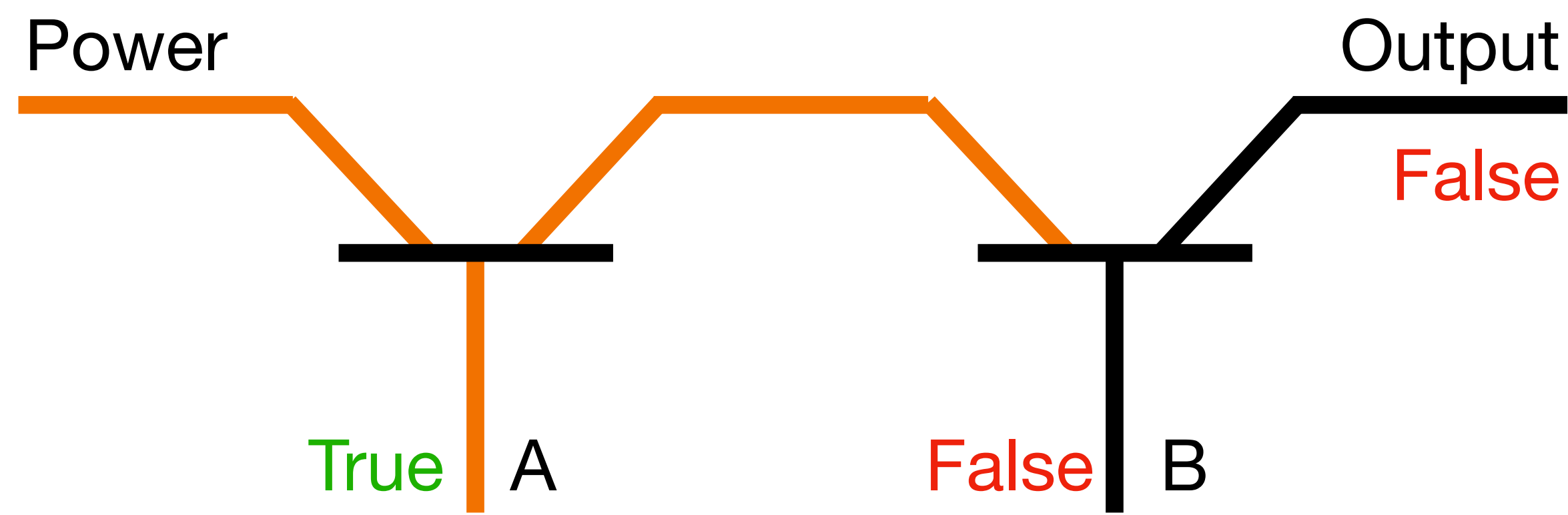
And

A	B	Output
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
FALSE	FALSE	FALSE



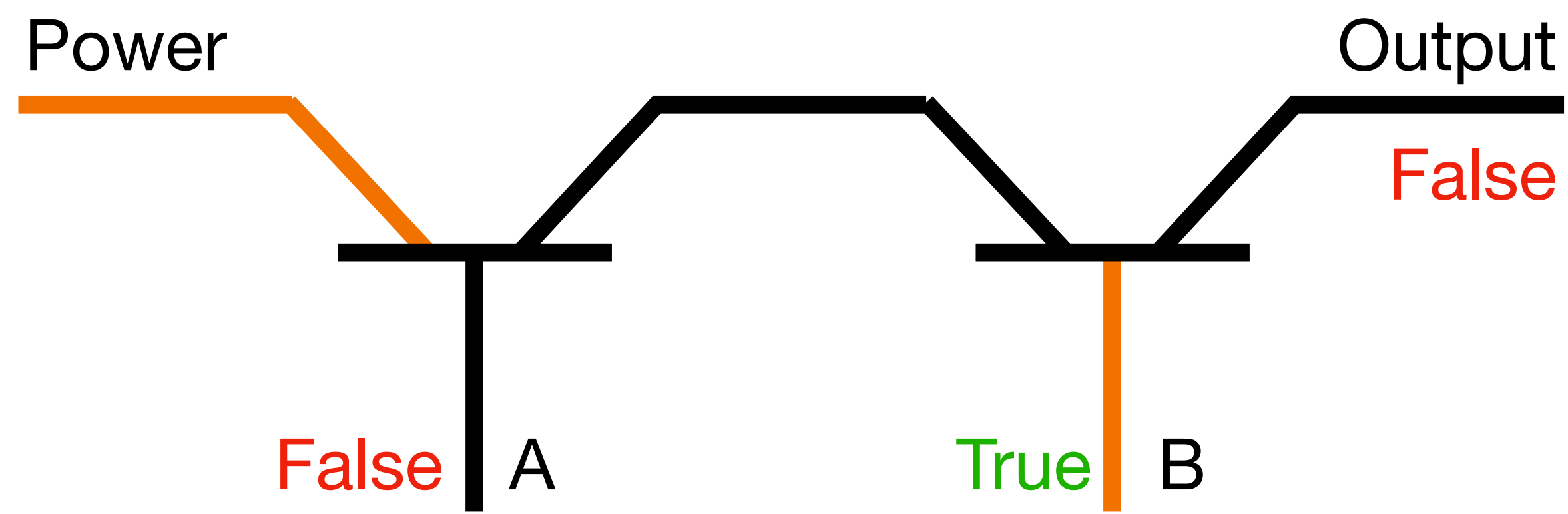
And

A	B	Output
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
FALSE	FALSE	FALSE



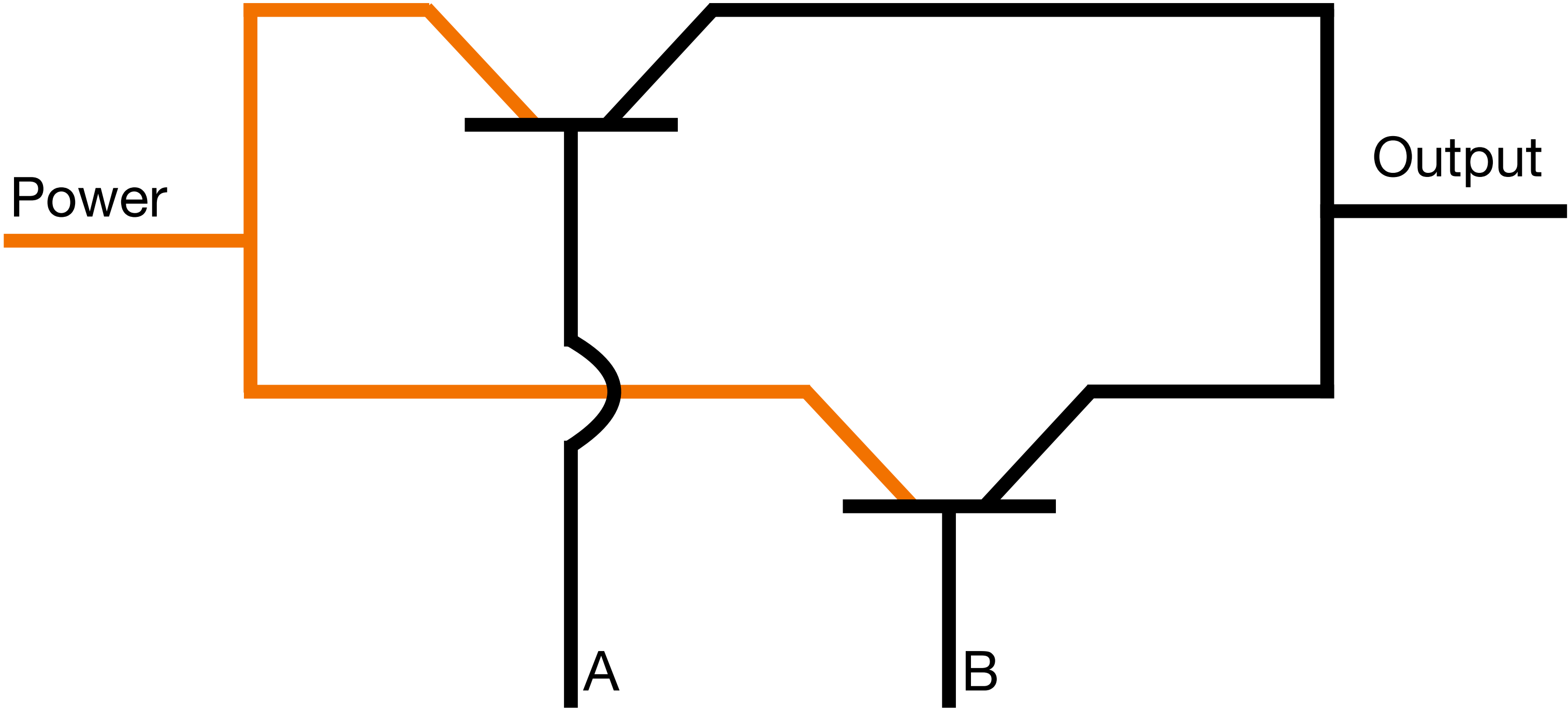
And

A	B	Output
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
FALSE	FALSE	FALSE



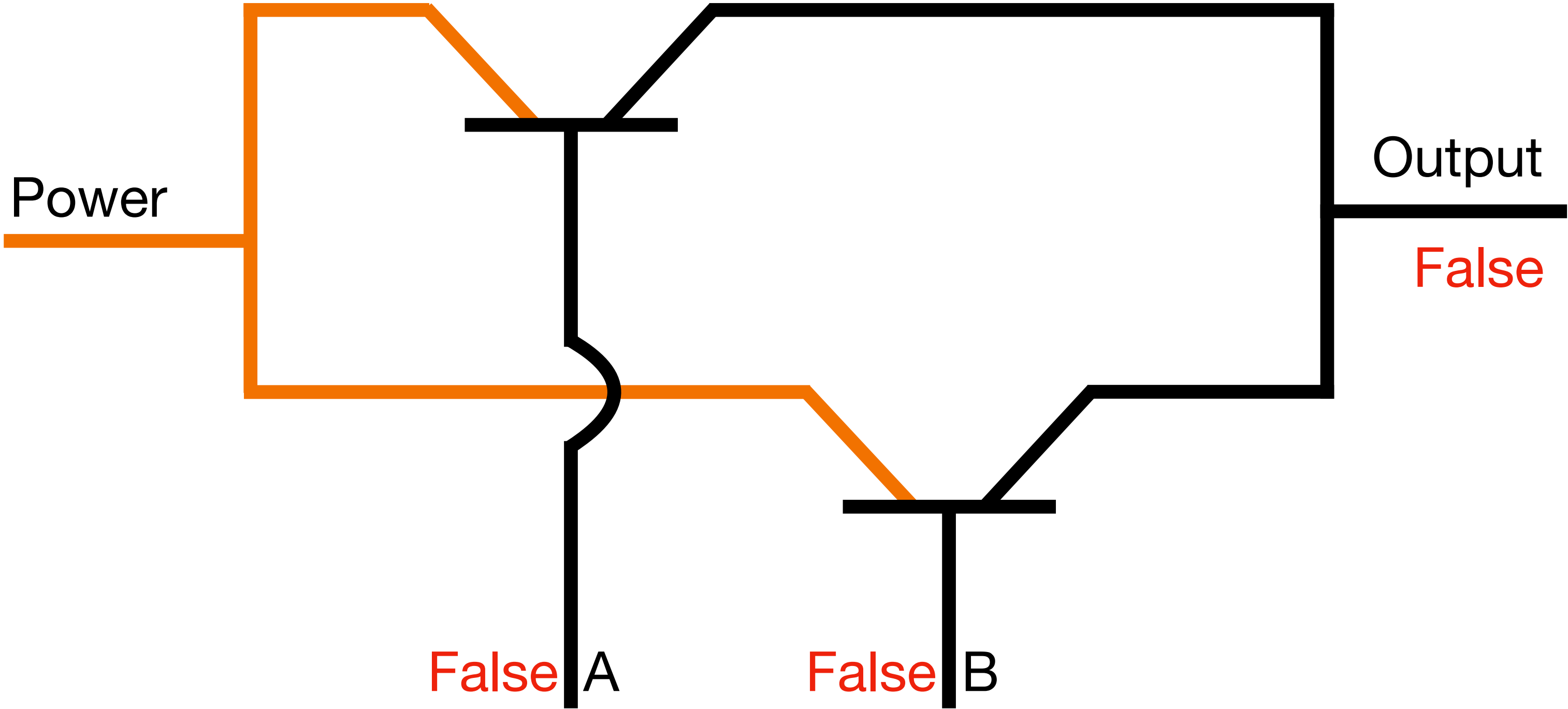
Or

A	B	Output
TRUE	TRUE	TRUE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	FALSE	FALSE



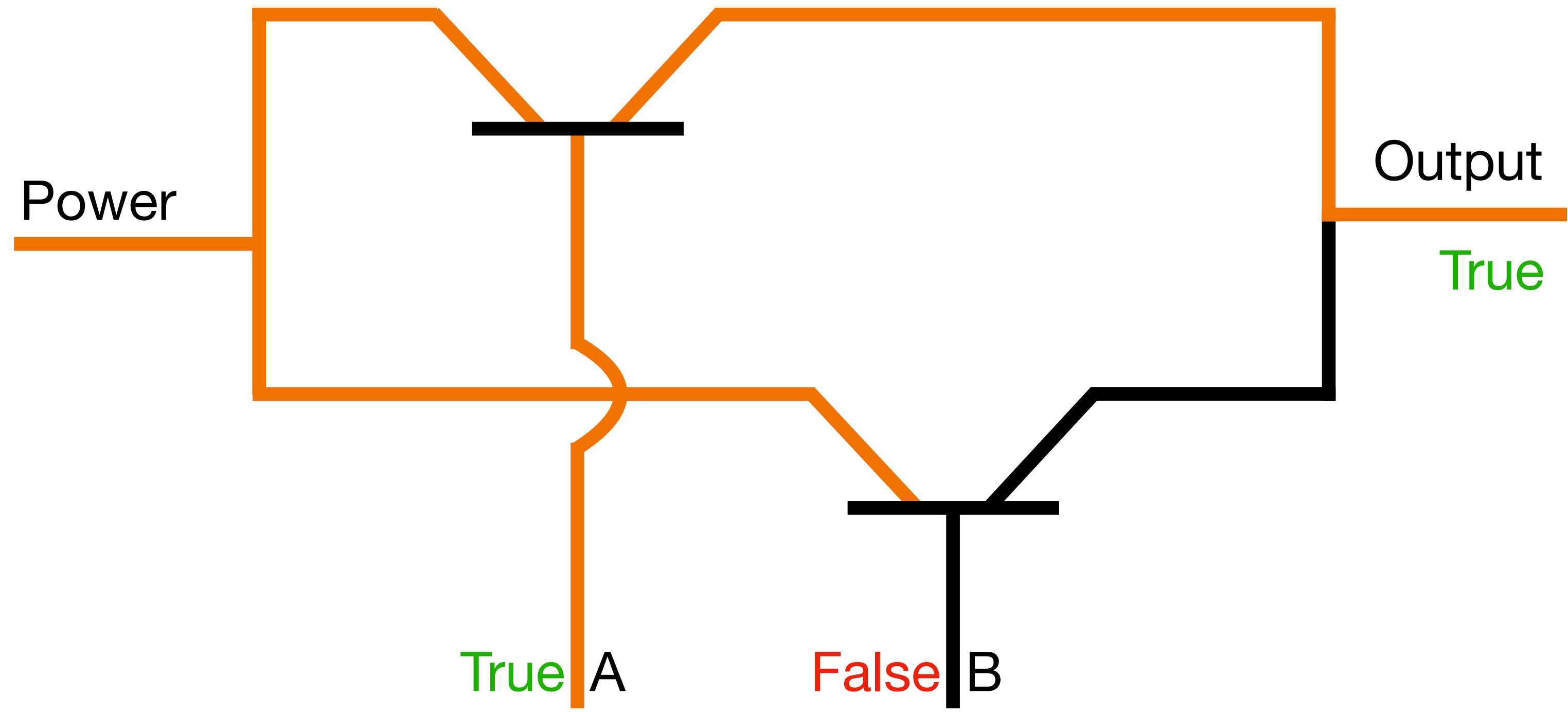
Or

A	B	Output
TRUE	TRUE	TRUE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	FALSE	FALSE



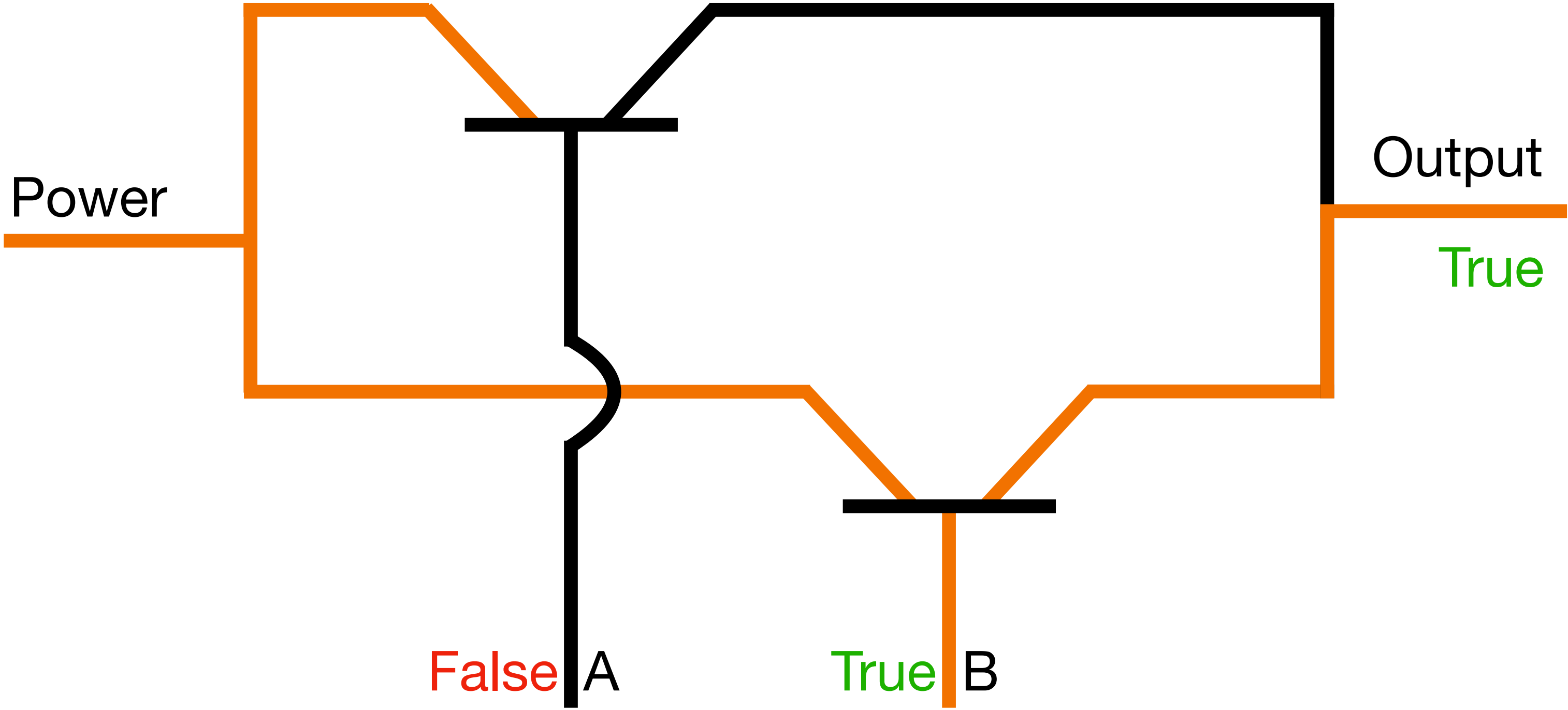
Or

A	B	Output
TRUE	TRUE	TRUE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	FALSE	FALSE



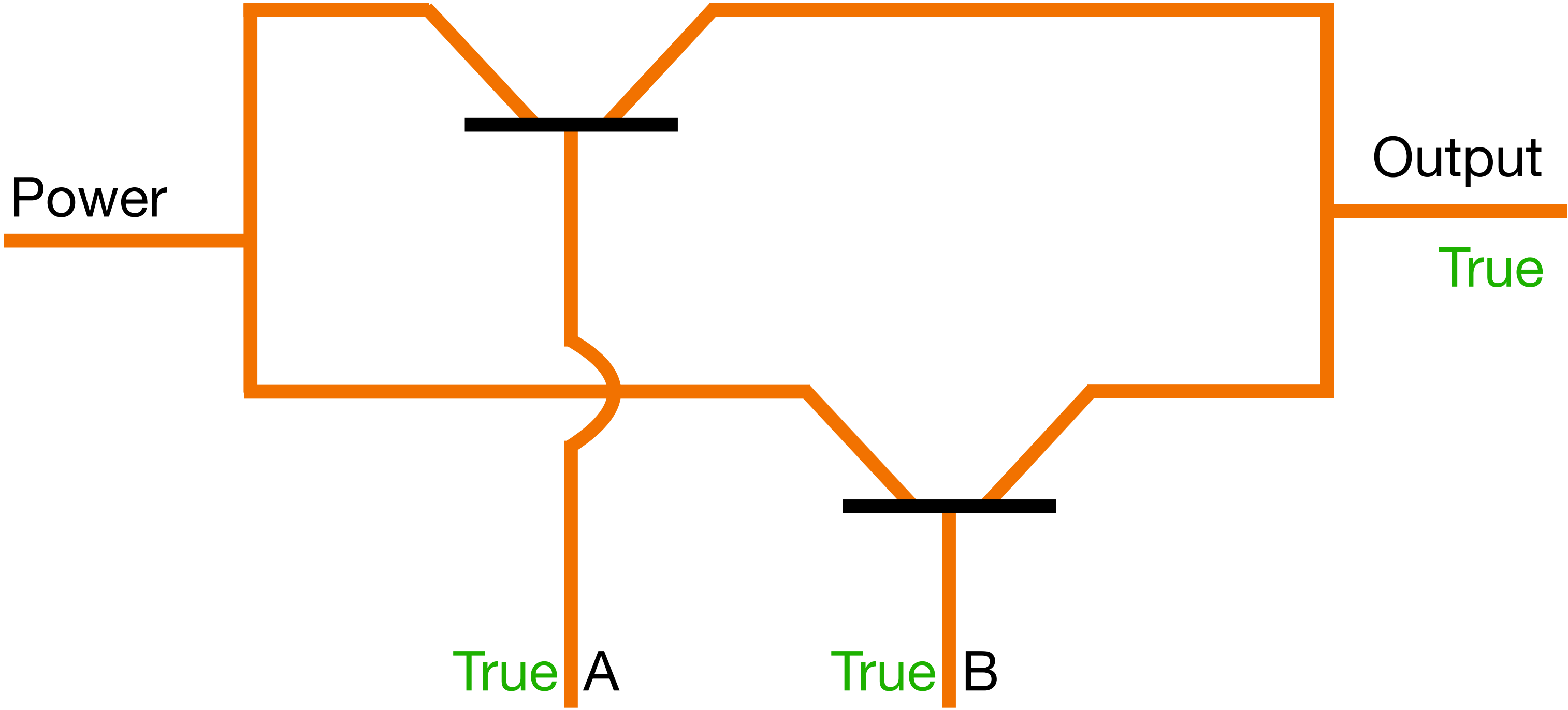
Or

A	B	Output
TRUE	TRUE	TRUE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	FALSE	FALSE

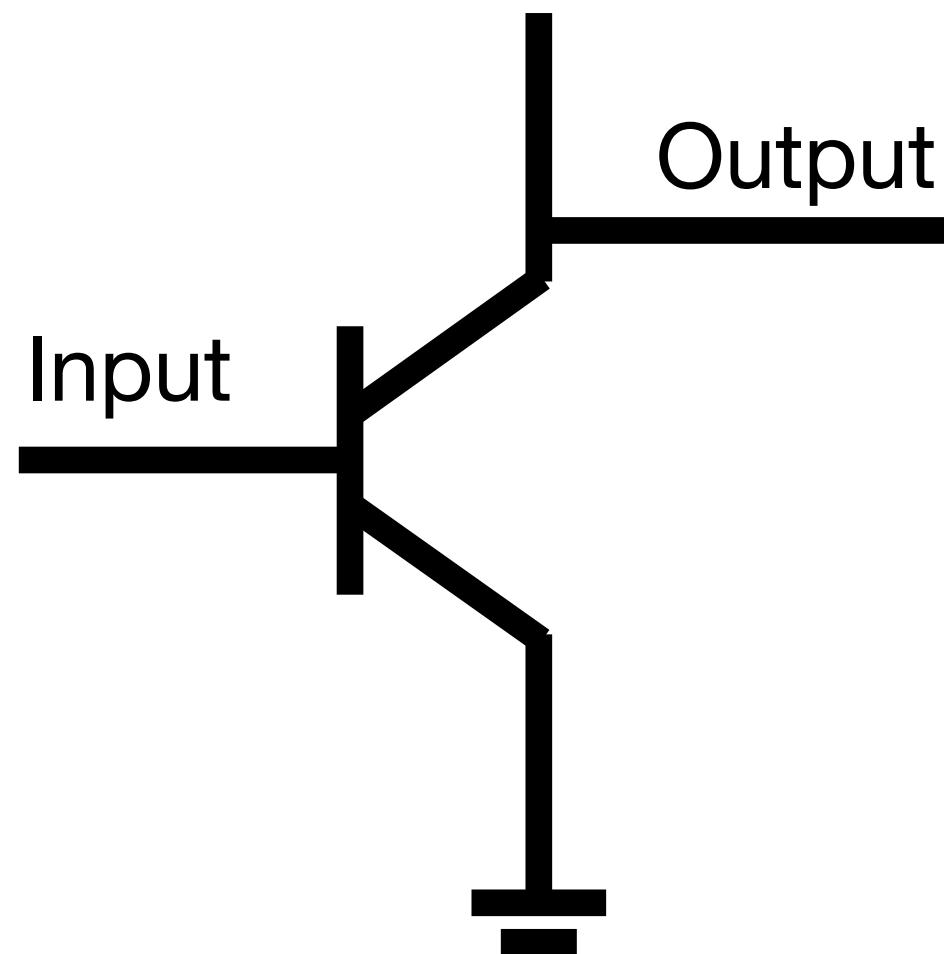


Or

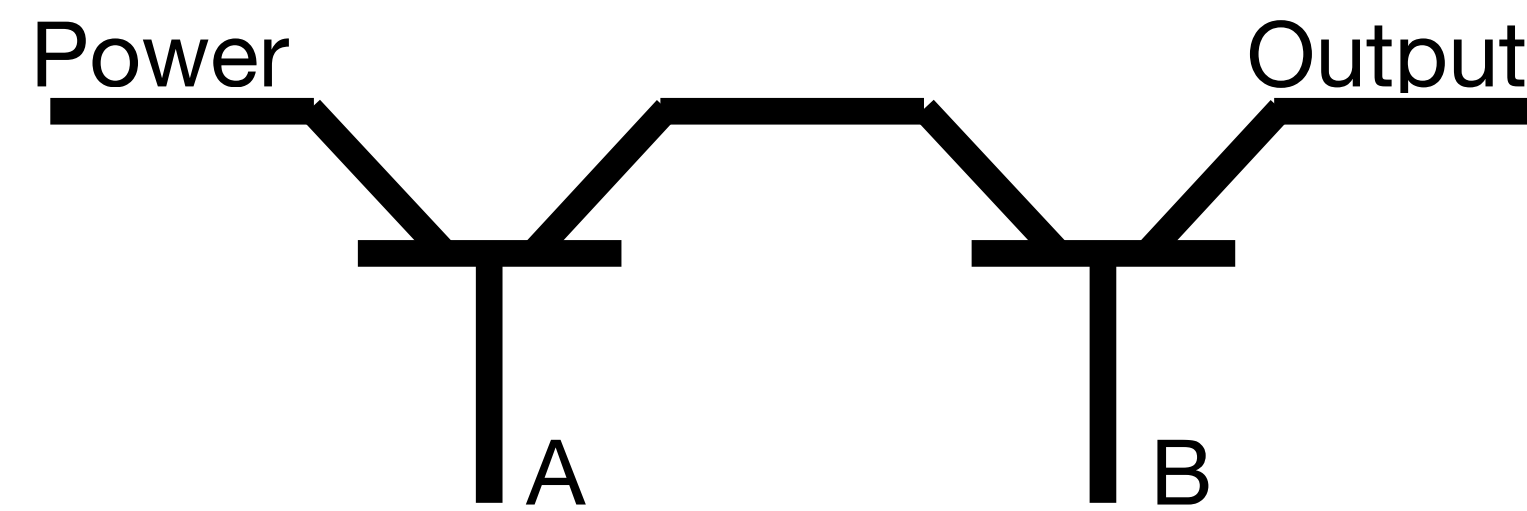
A	B	Output
TRUE	TRUE	TRUE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	FALSE	FALSE



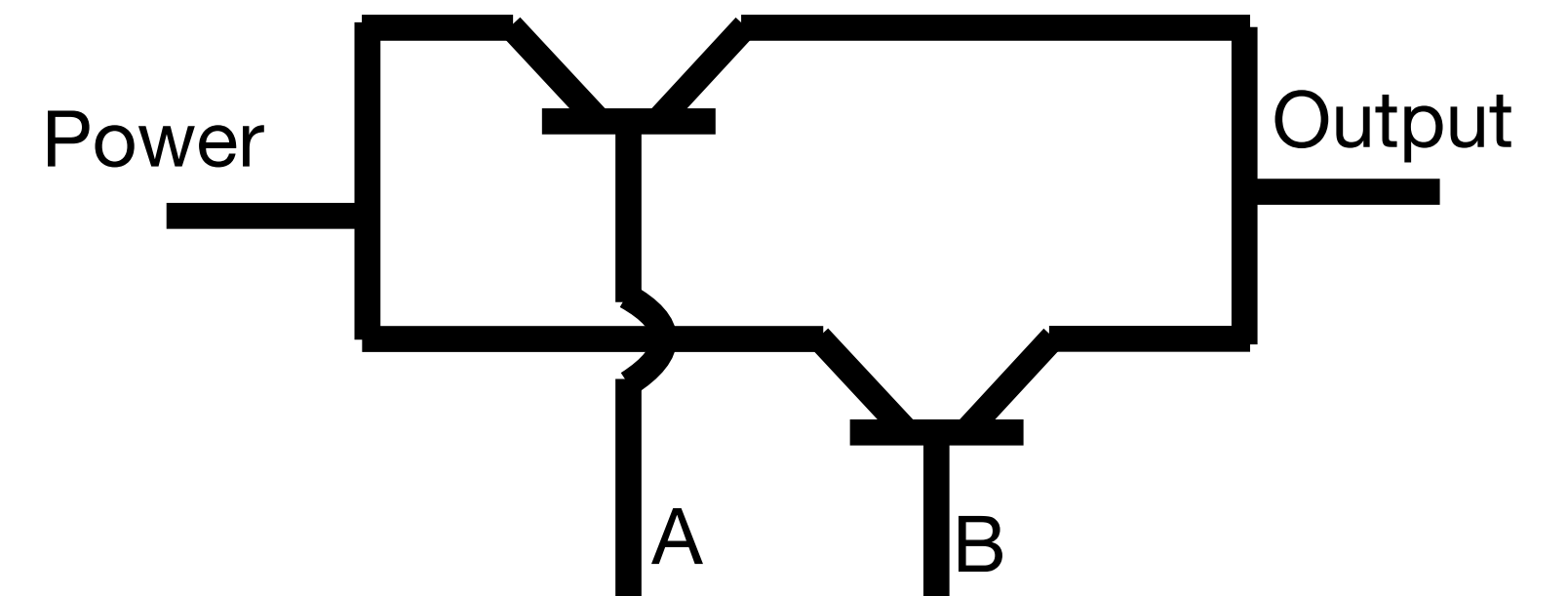
Not



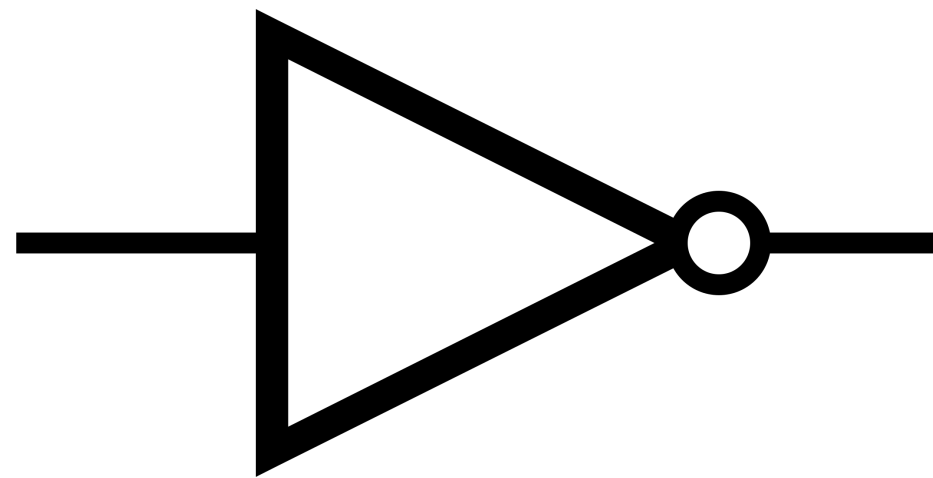
And



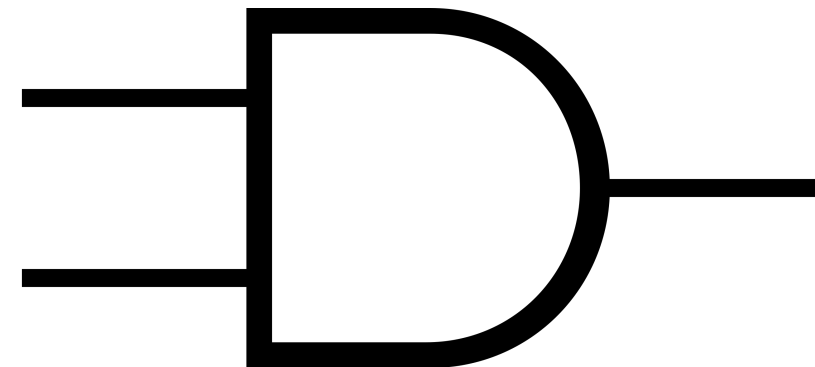
Or



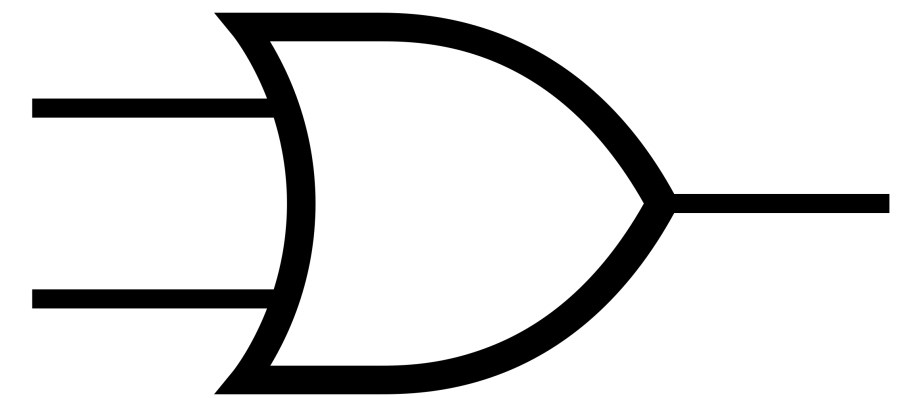
Not



And

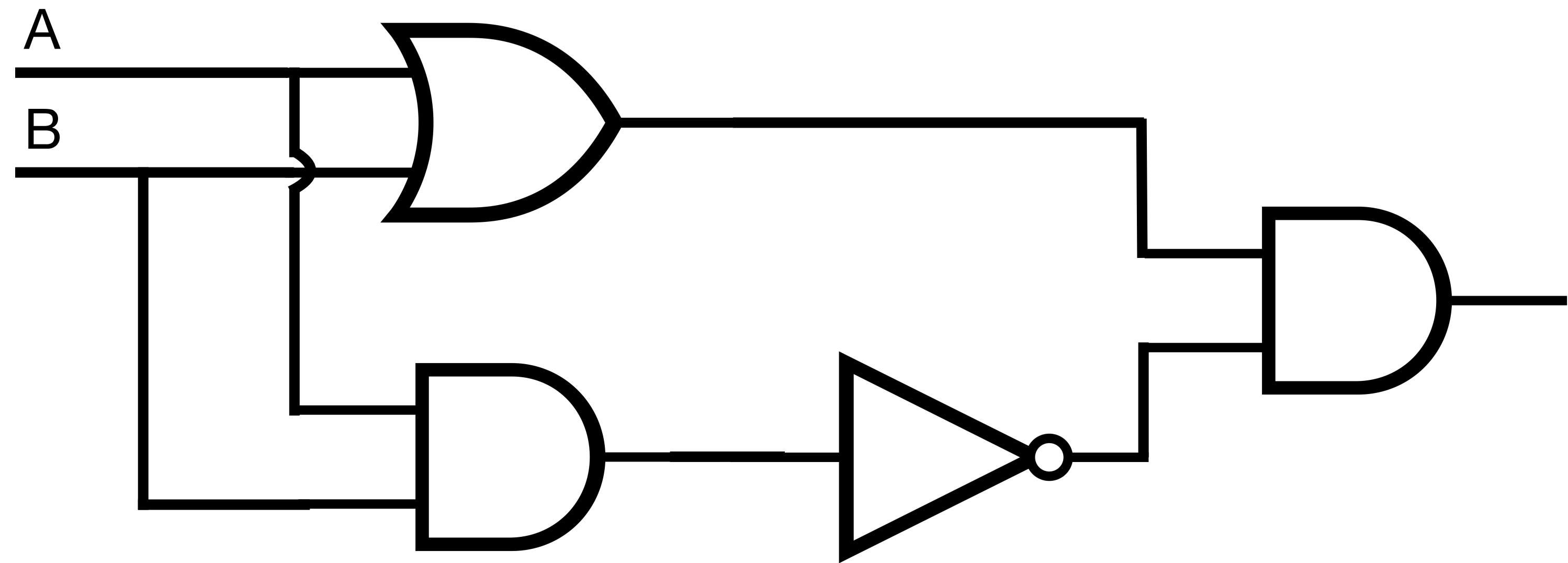


Or



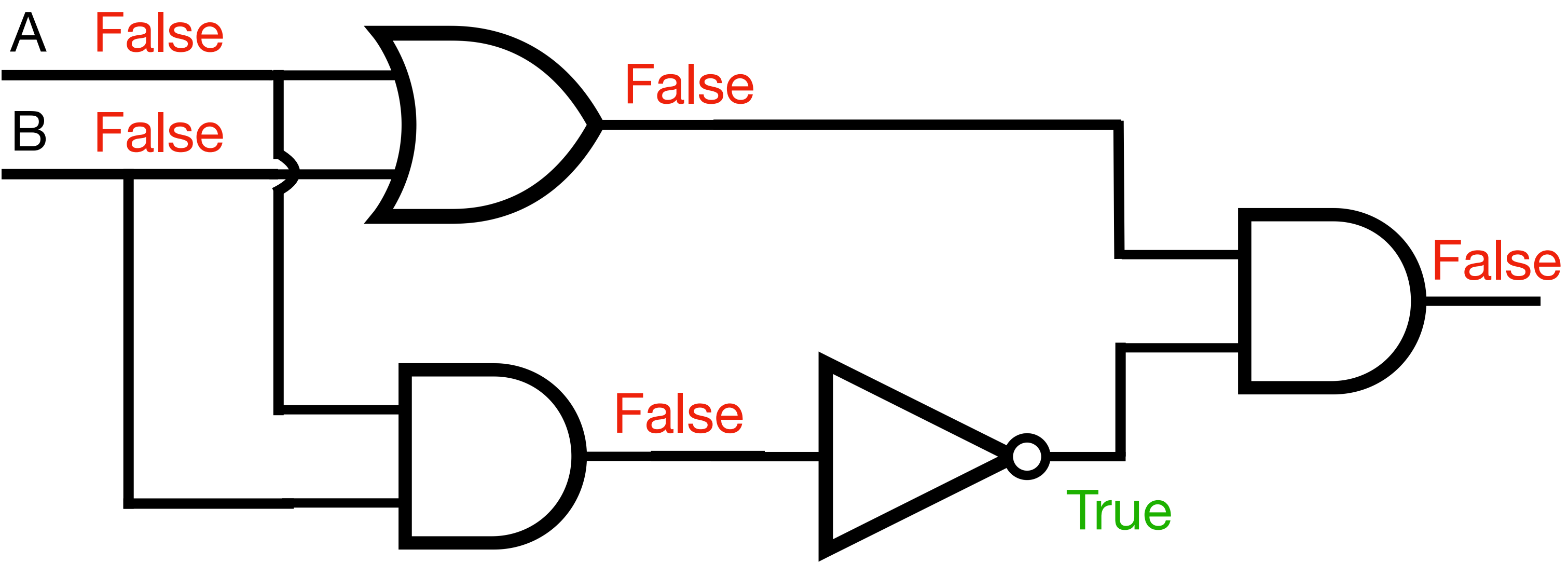
Xor

A	B	Output
TRUE	TRUE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	FALSE	FALSE



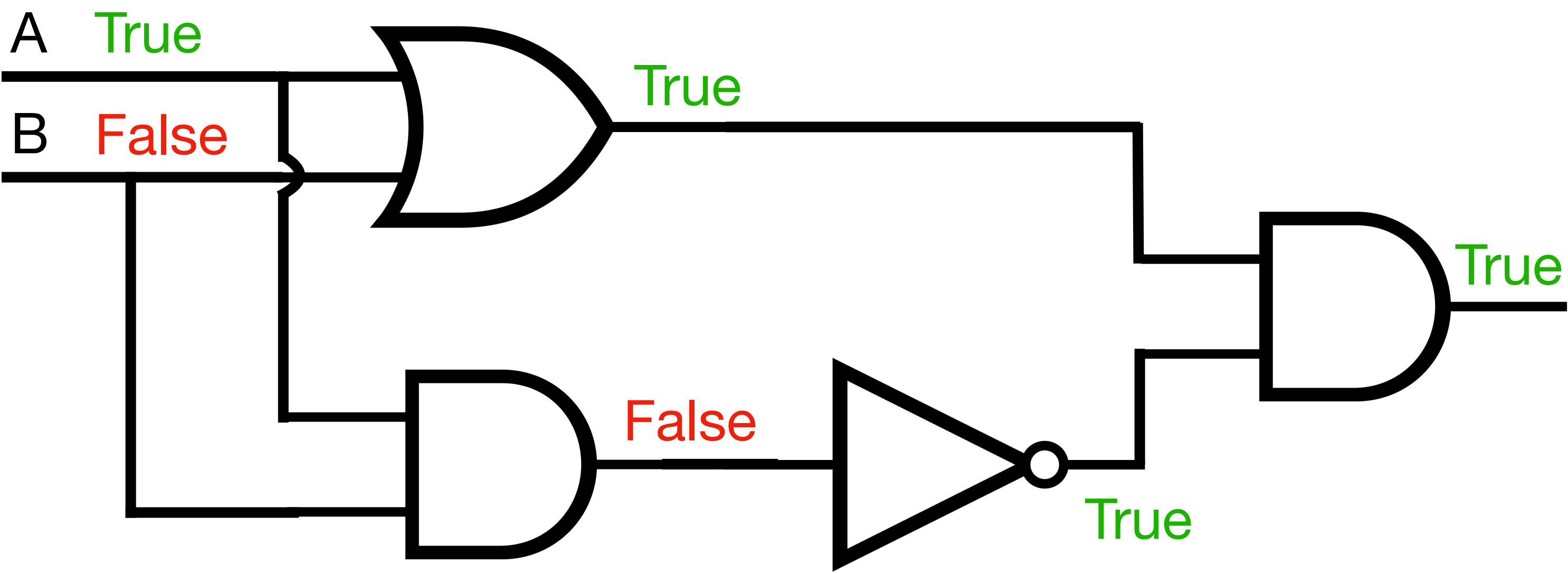
Xor

A	B	Output
TRUE	TRUE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	FALSE	FALSE



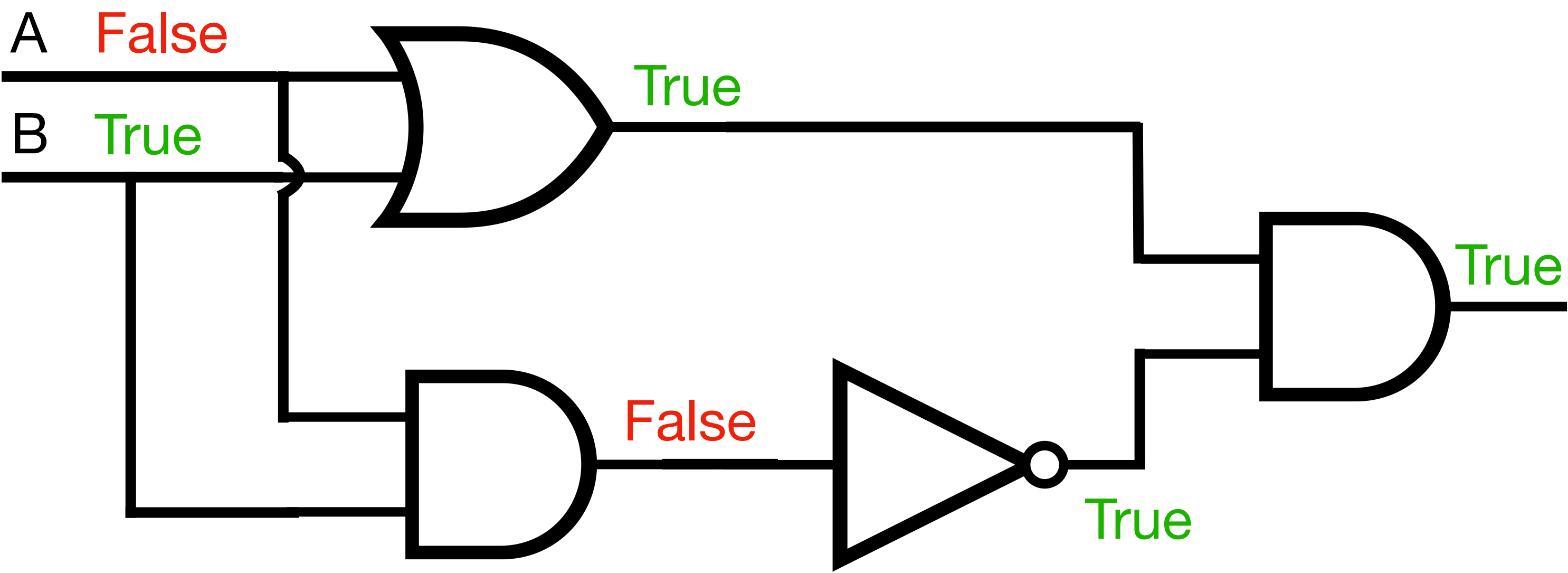
Xor

A	B	Output
TRUE	TRUE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	FALSE	FALSE



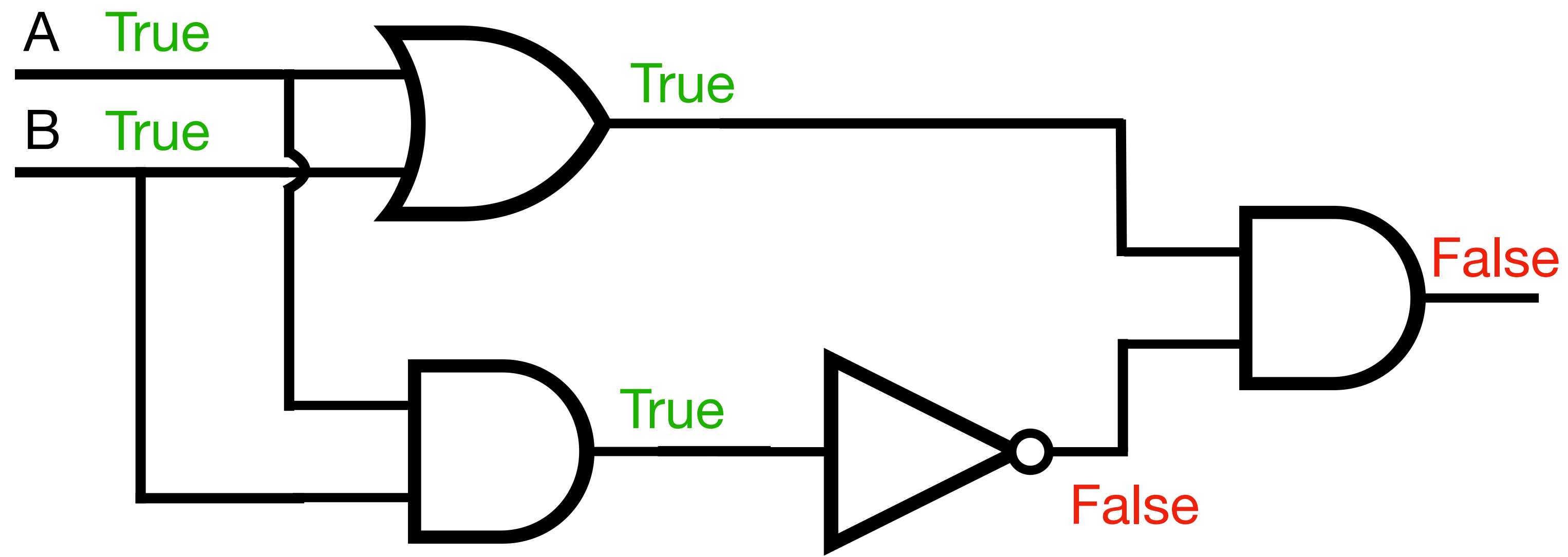
Xor

A	B	Output
TRUE	TRUE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	FALSE	FALSE

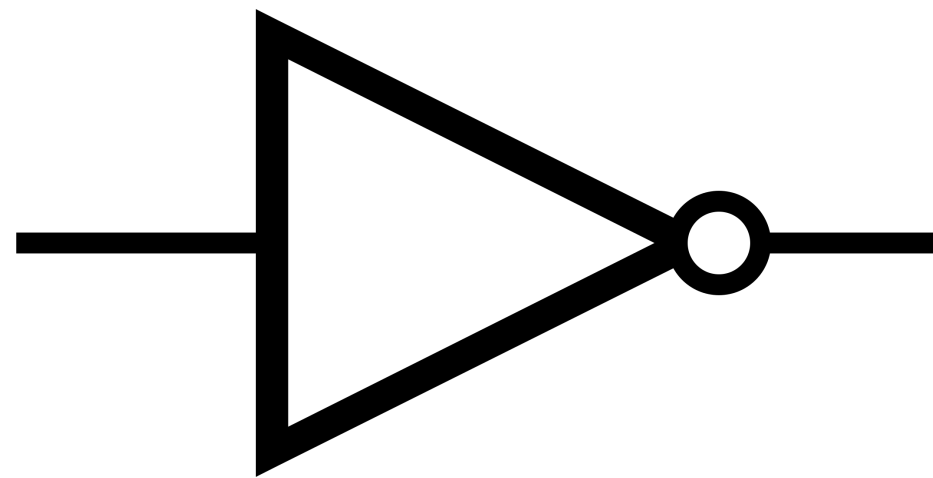


Xor

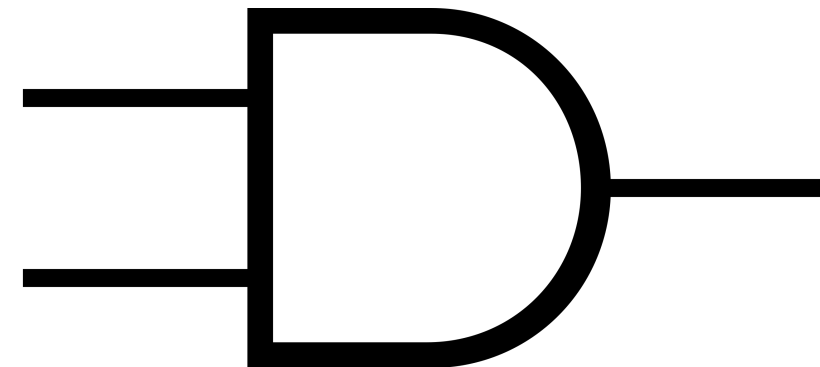
A	B	Output
TRUE	TRUE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	FALSE	FALSE



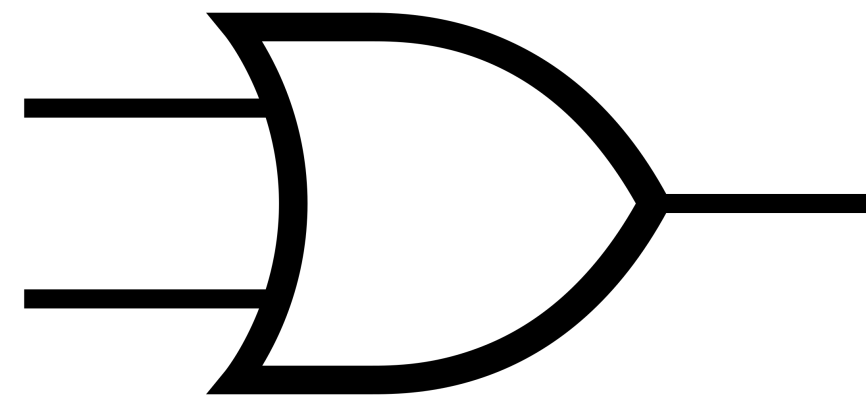
Not



And



Or



Xor

