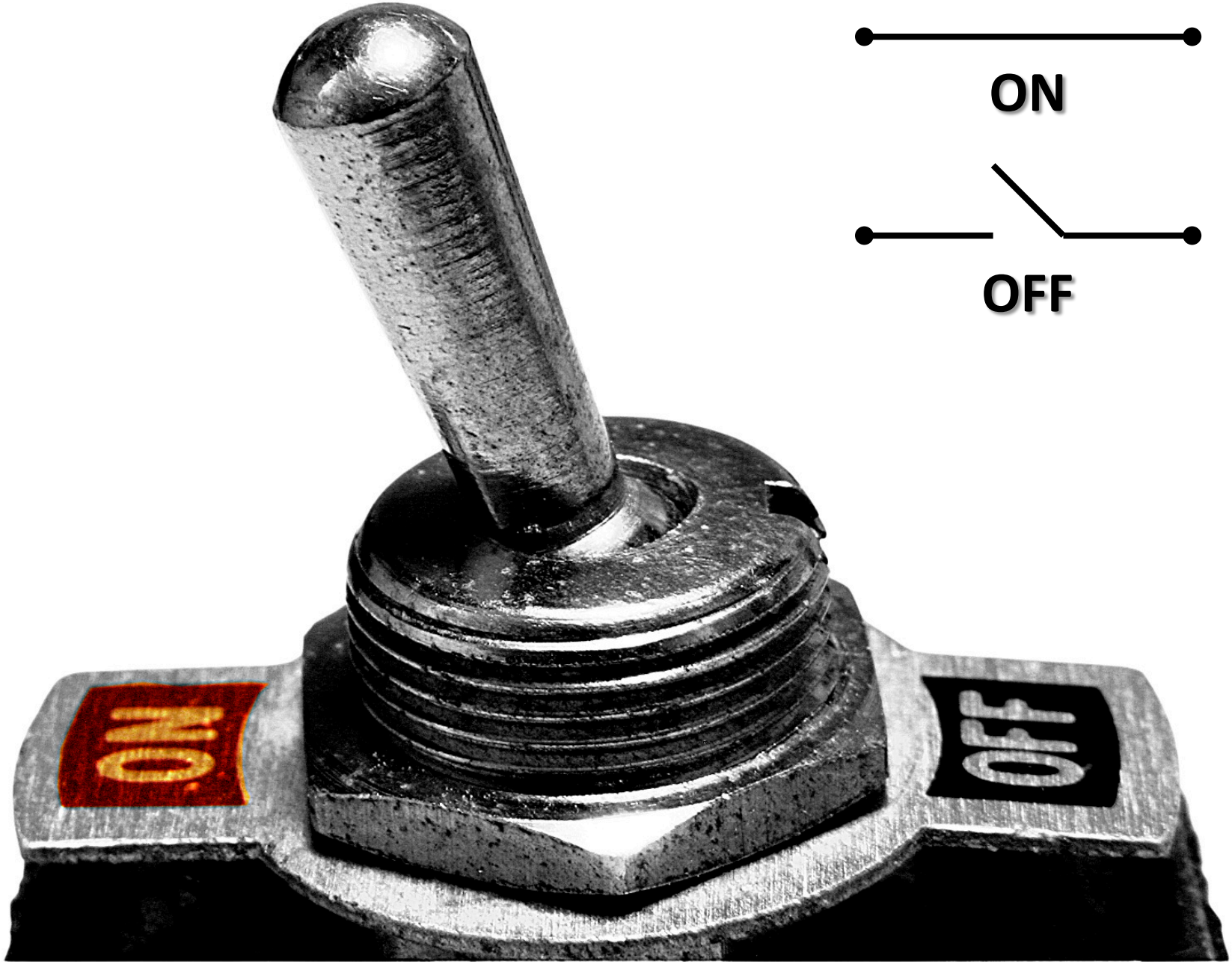
Algebraic Modelling of Switching Networks with Uncertainty

Andrey Mokhov
Newcastle University, UK

Part I: The Switch

"The first light switch employing *quick-break technology* was invented by *John Henry Holmes* in 1884 in the Shieldfield district of Newcastle upon Tyne."

Wikipedia


ON
OFF
ON
OFF

Relay: electrically operated switch

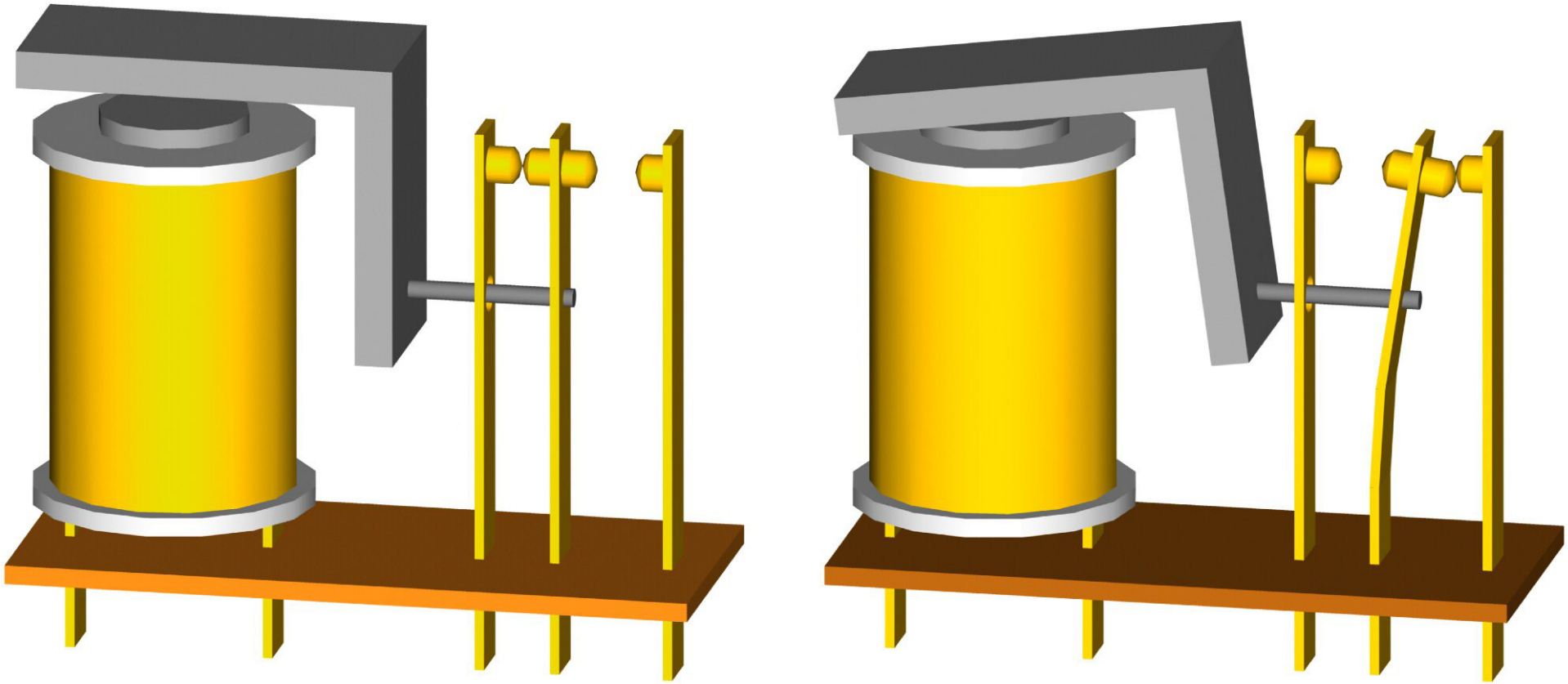
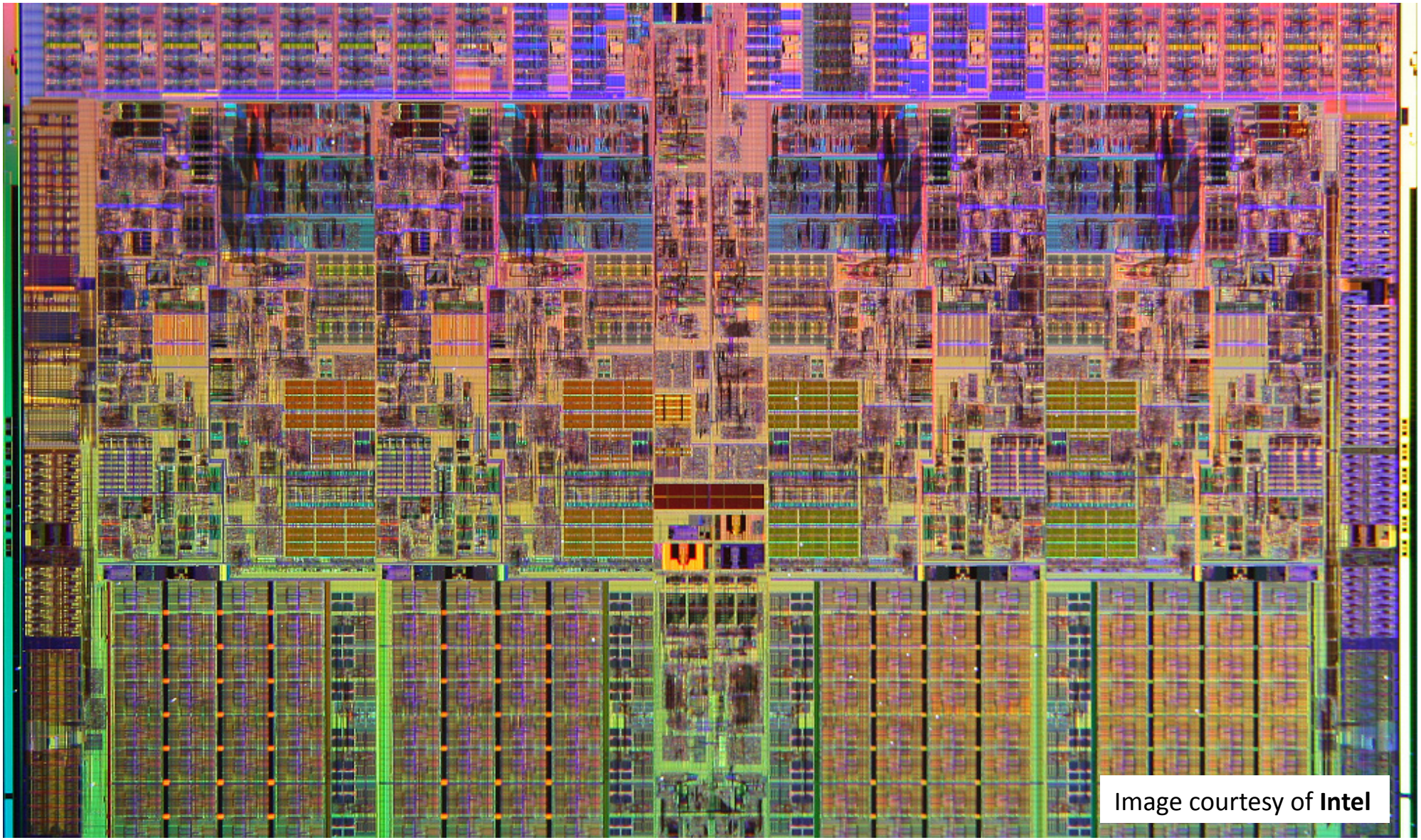
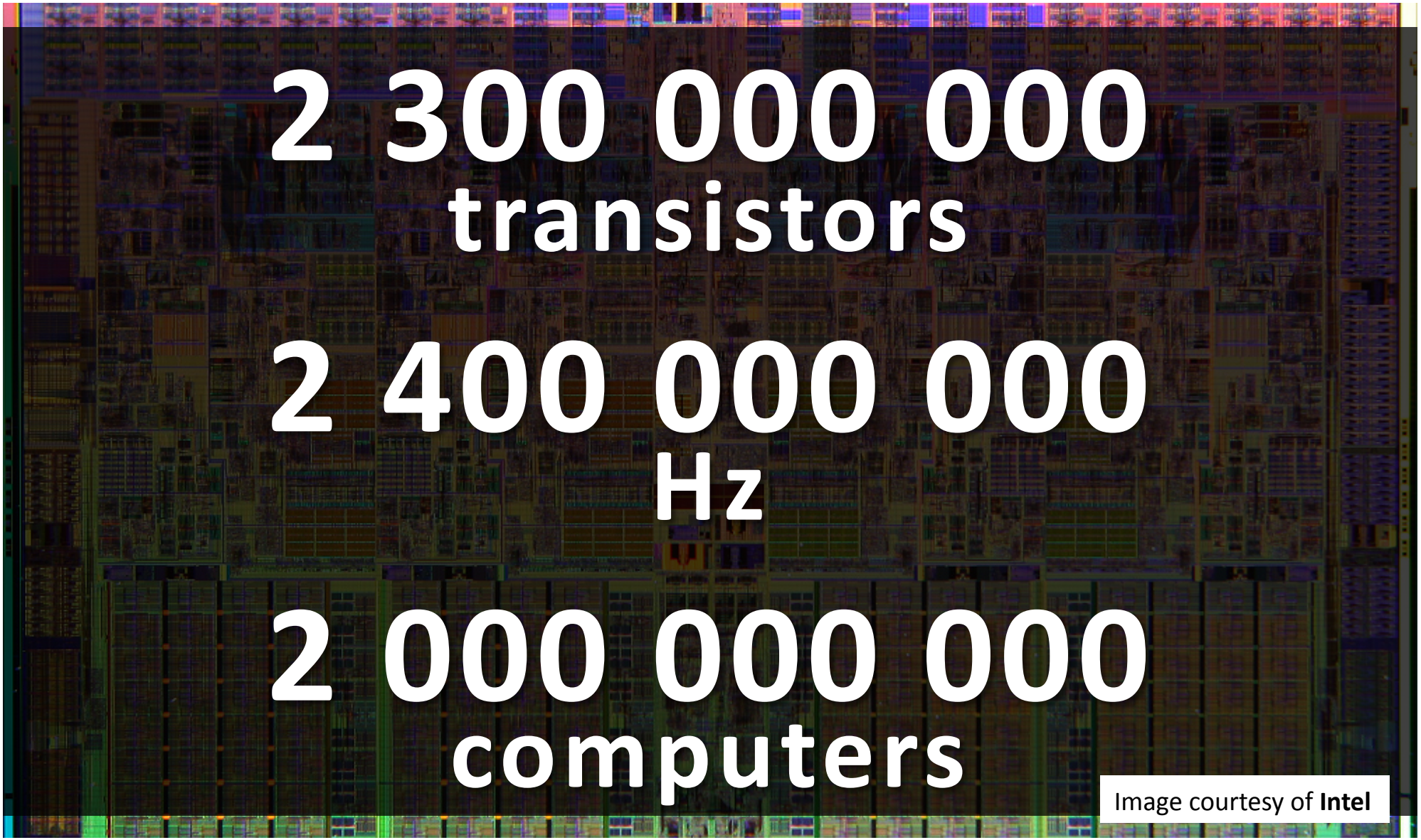


Image courtesy of **Wikipedia**

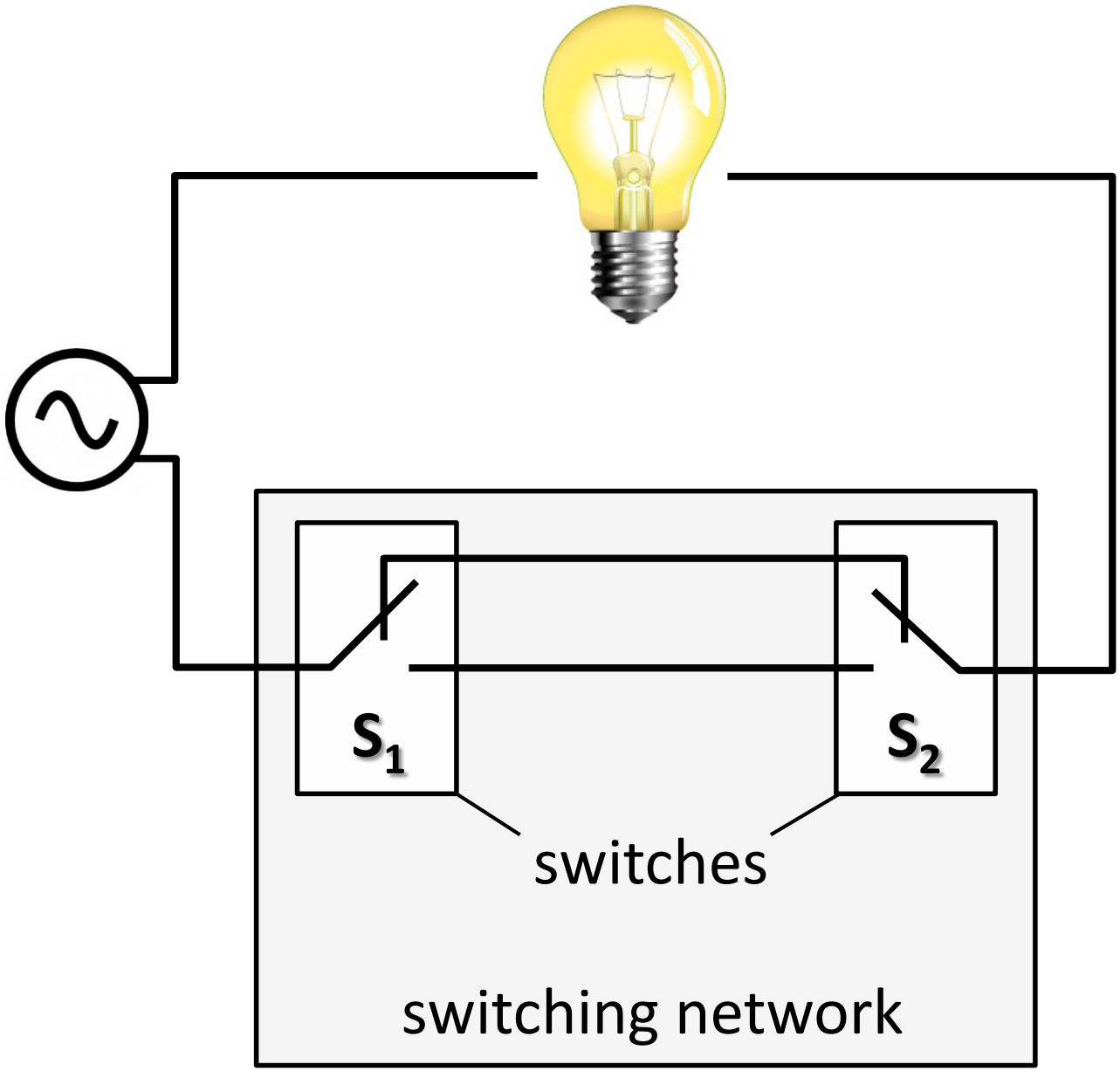
Smaller, faster, cheaper

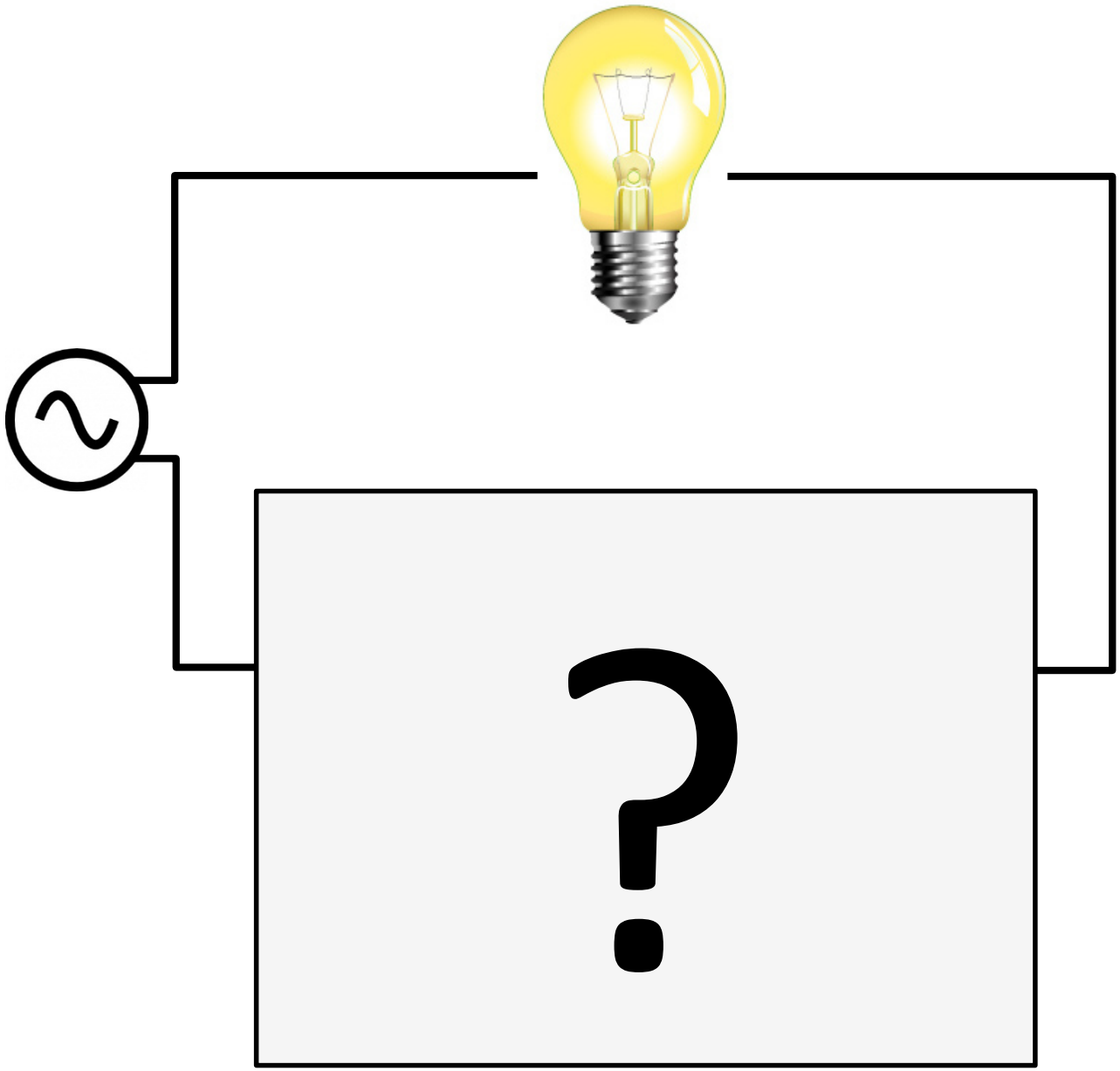


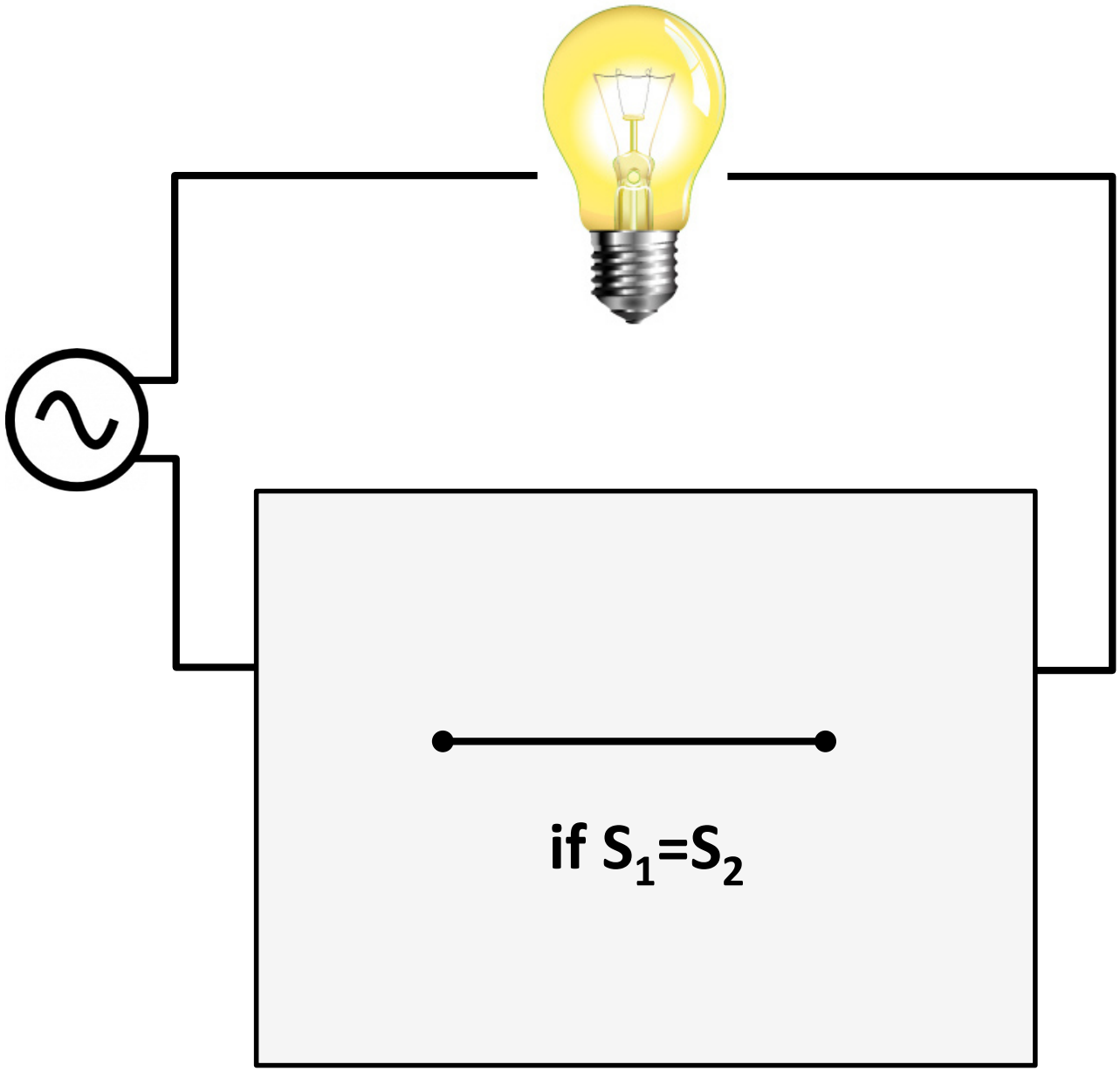
Smaller, faster, cheaper

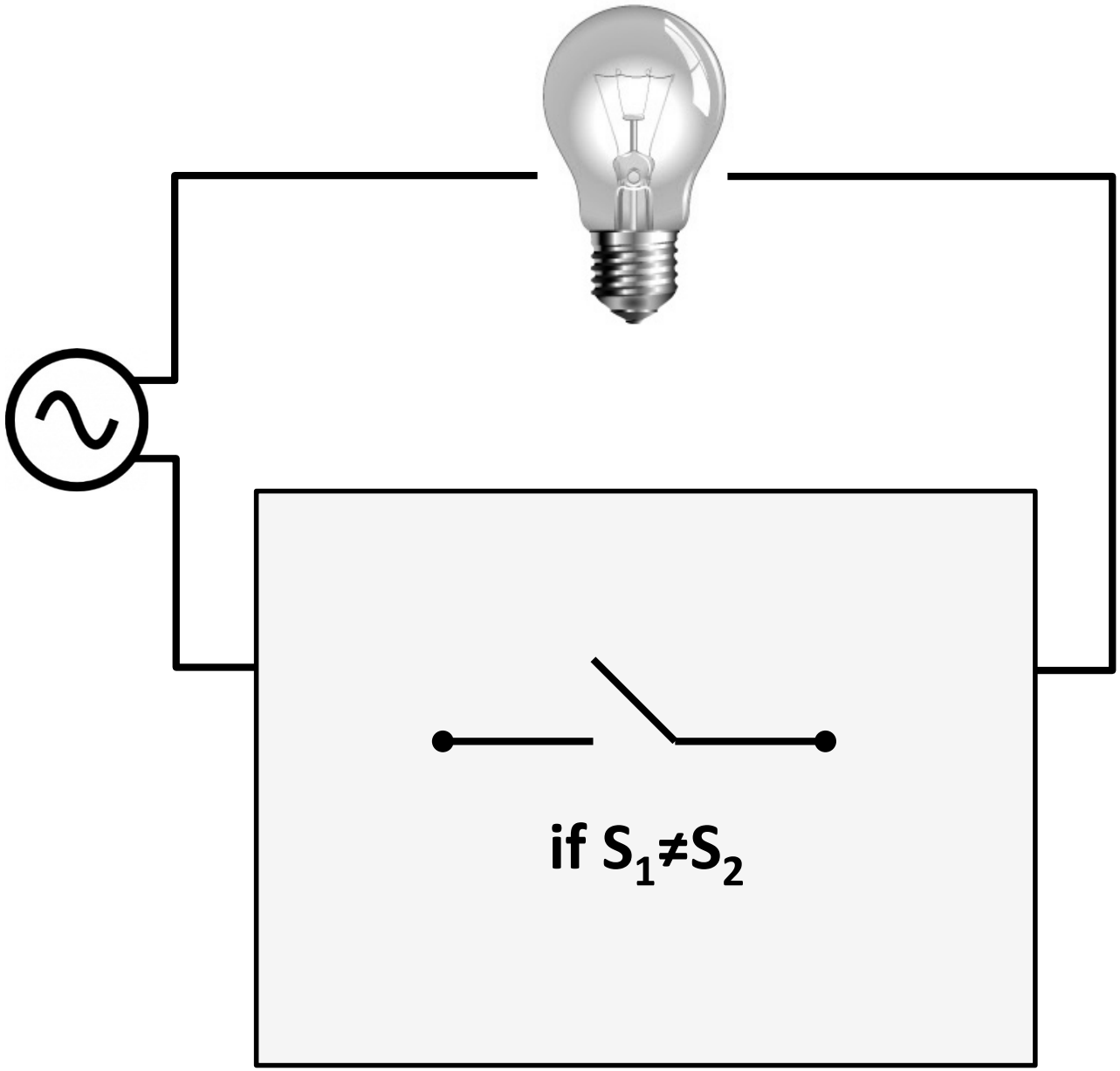


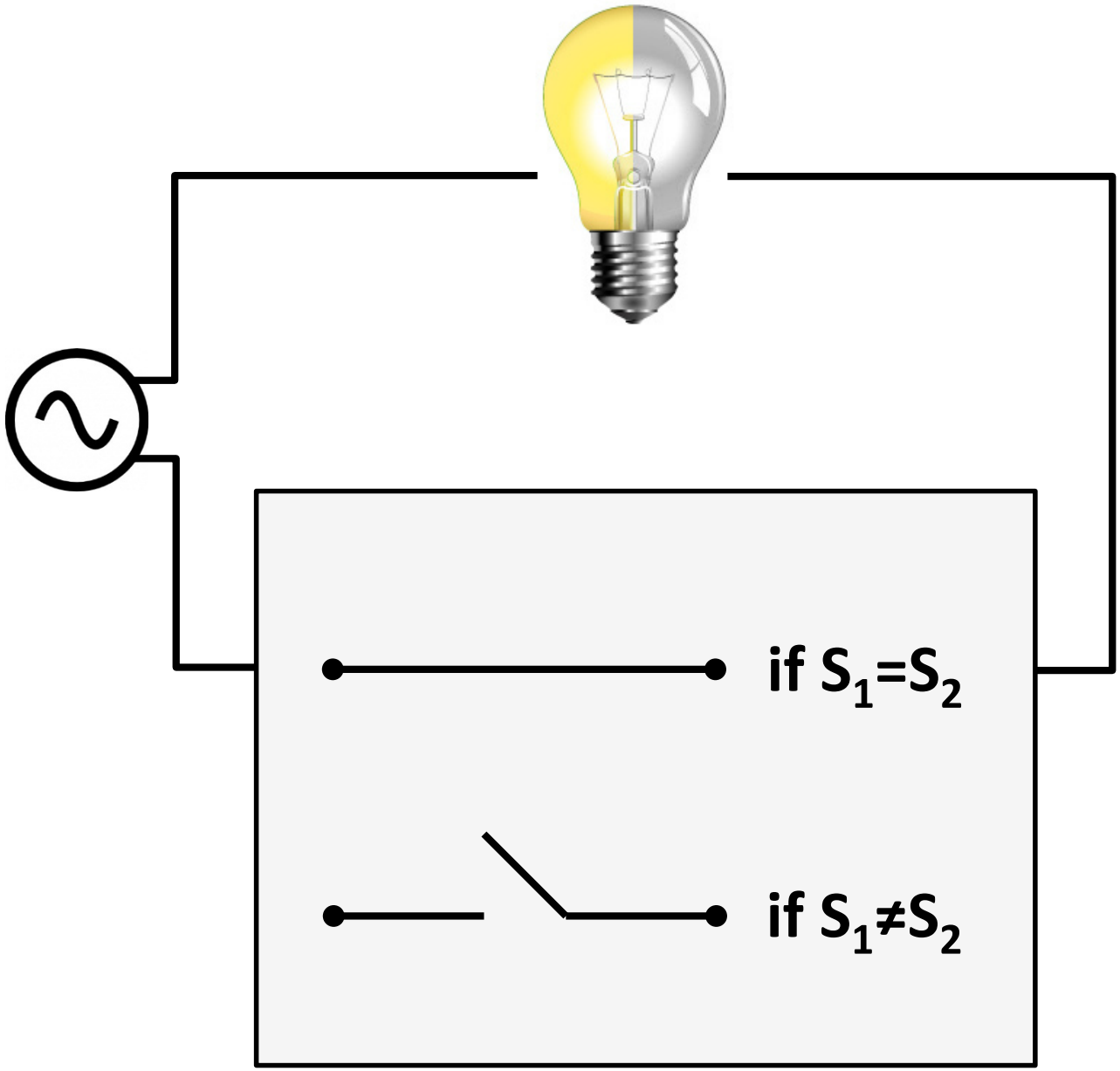
Part II:
Switching Networks

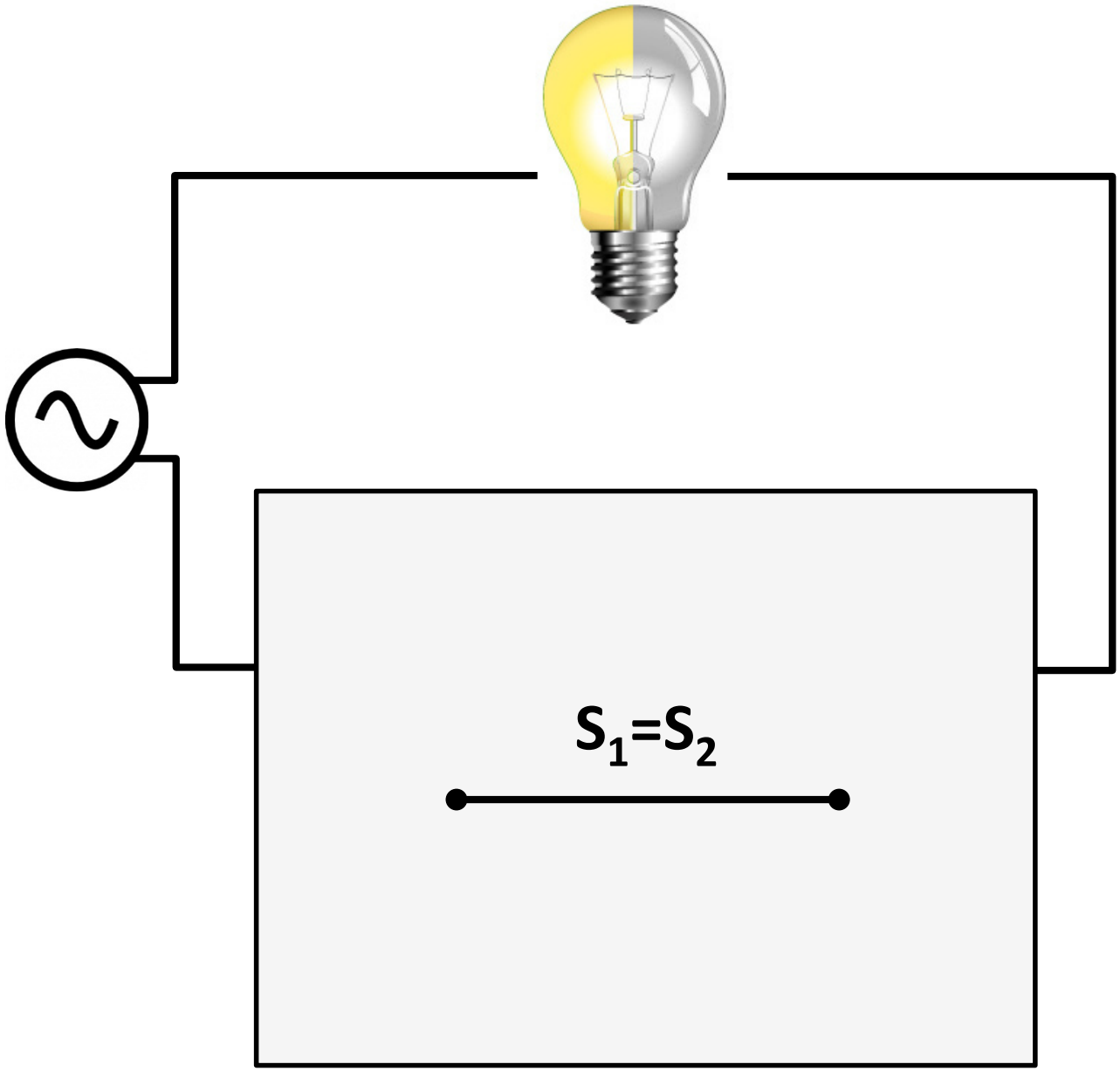

S_1
S_2
switches
switching network




if $S_1=S_2$

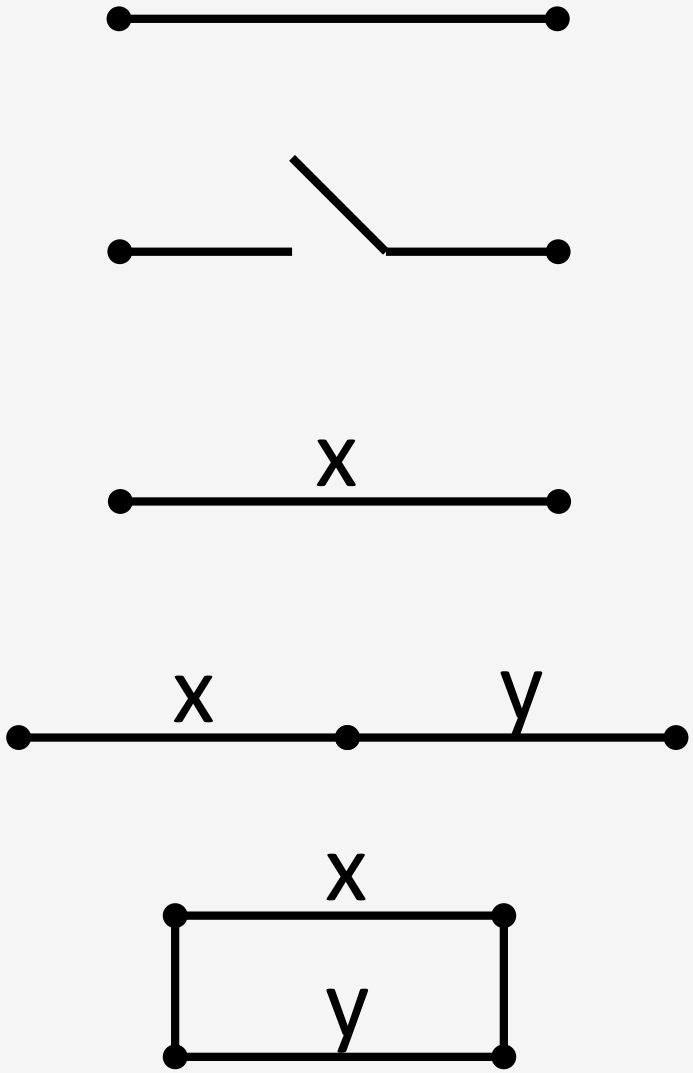

if $S_1 \neq S_2$


if $S_1=S_2$
if $S_1 \neq S_2$


$S_1=S_2$

Shannon's Boolean analysis (1937)

Graph theory



Boolean algebra

$f = 1$

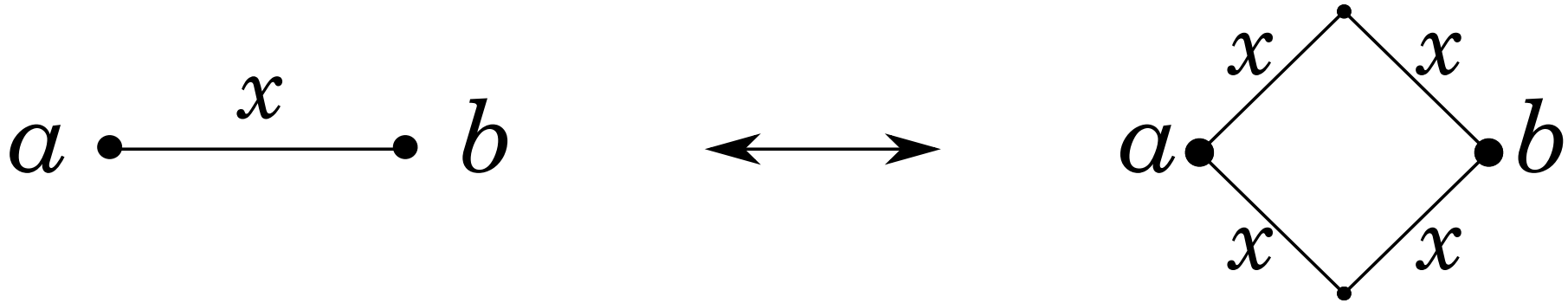
$f = 0$

$f = x$

$f = x \wedge y$

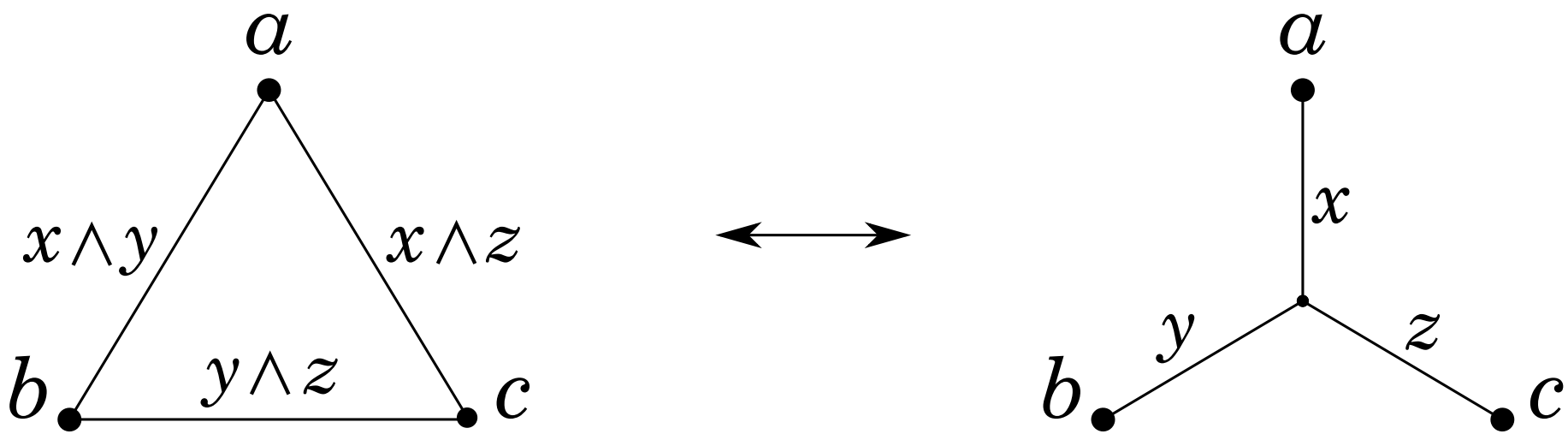
$f = x \vee y$

Equivalent transformations



$f_{ab} = x$ $\qquad$ $f_{ab} = (x \wedge x) \vee (x \wedge x) = x$

Equivalent transformations



$$f_{ab} = x \wedge y$$
$$f_{ac} = x \wedge z$$
$$f_{bc} = y \wedge z$$

Part III: Algebra

"Algebra is arithmetic for lazy people."

Author unknown

Algebra is arithmetic for lazy people

1 + 2 + 3 + … + 100 = ?

Algebra is arithmetic for lazy people

$$1 + 2 + 3 + \ldots + 100 = ?$$

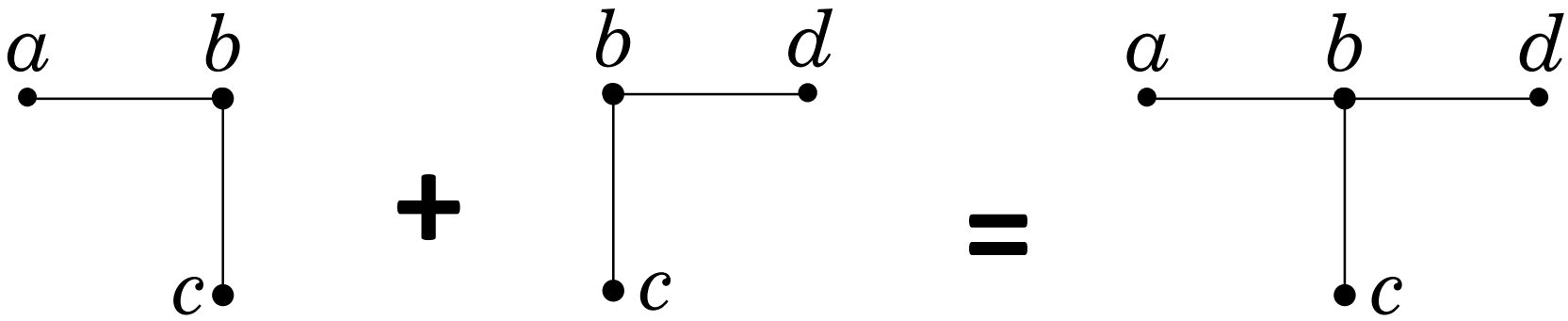
$$= (1 + 100) + (2 + 99) + \ldots$$

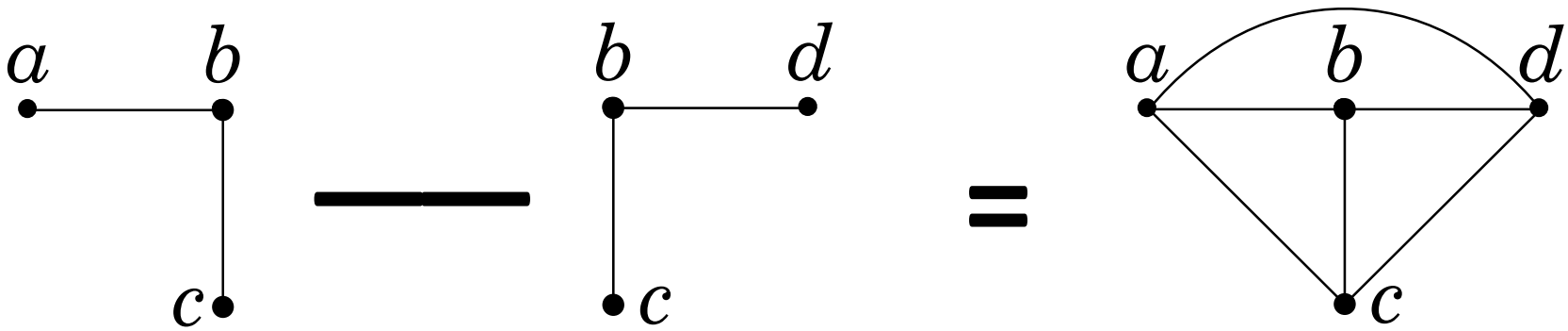
$$= 101 + 101 + \ldots$$

$$= 101 * 50$$

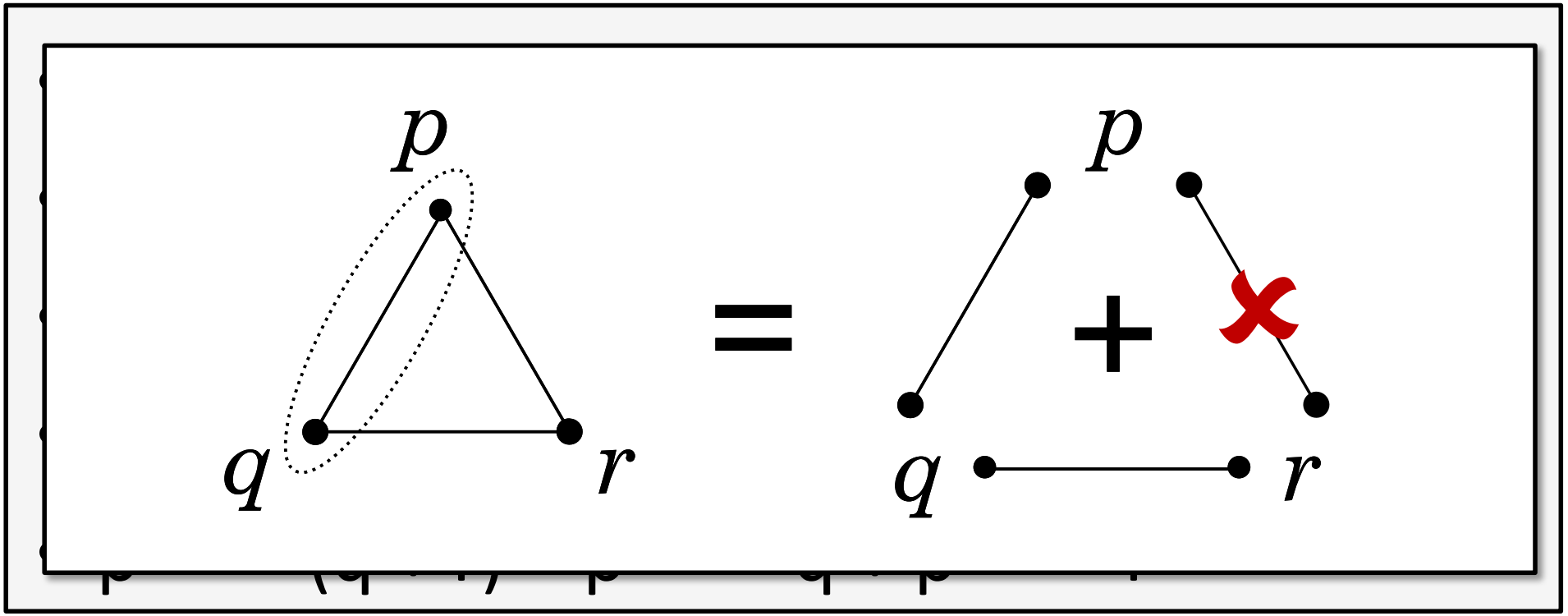
$$= 5050$$

Algebra of Switching Networks





Axioms



- (p —— q) —— r = p —— q + p —— r + q —— r
- (q ≠ ε) => (p —— q) —— r = p —— q + q —— r

Theorems

- $(p \text{ —— } q) \text{ —— } r = p \text{ —— } (q \text{ —— } r)$
- $p + \varepsilon = p$
- $p + p = p$
- $p + p \text{ —— } q = p \text{ —— } q$

Very different from arithmetic!

From graphs to families of graphs

- $[0]p = \varepsilon$
- $[1]p = p$

- $[x]p$ is a family of two graphs: $\{\varepsilon, p\}$

- Switch is a family $\{p + q, p \text{ —— } q\}$

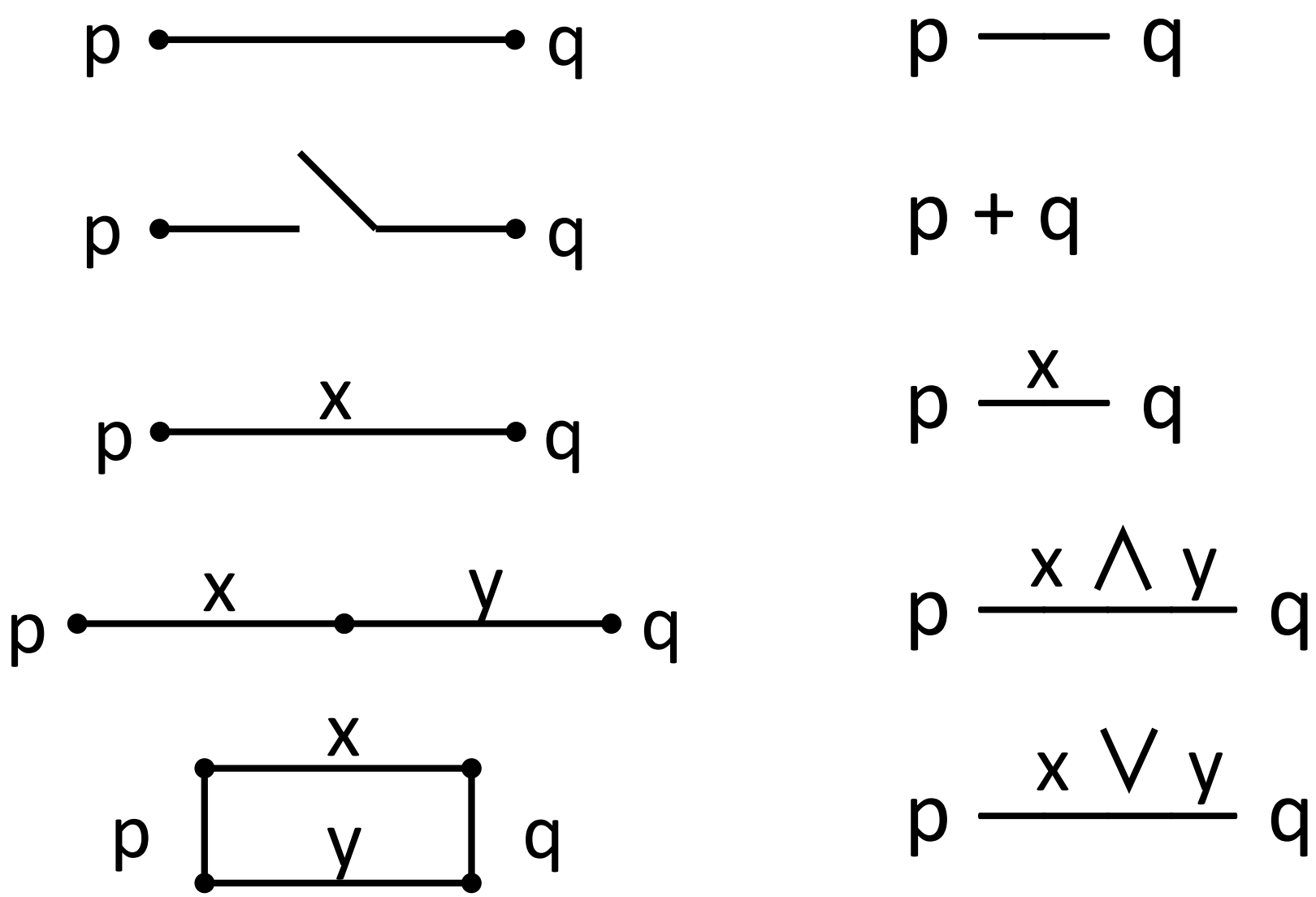
- $p \overset{x}{\text{——}} q = p + q + [x](p \text{ —— } q)$

From graphs to families of graphs

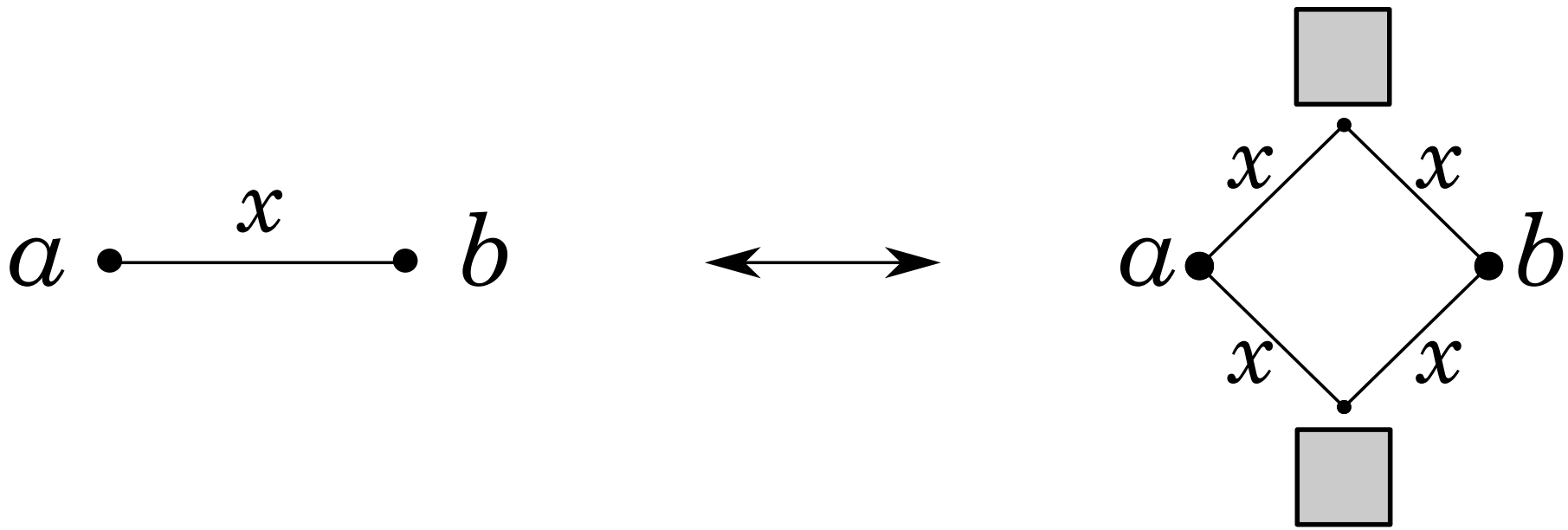
- $[0]p = \varepsilon$
- $[1]p = p$

- $[x]\varepsilon = \varepsilon$
- $[x](p + q) = [x]p + [x]q$
- $[x](p \text{ —— } q) = [x]p \text{ —— } [x]q$
- $[x \wedge y]p = [x][y]p$
- $[x \vee y]p = [x]p + [y]p$

Algebraic representation



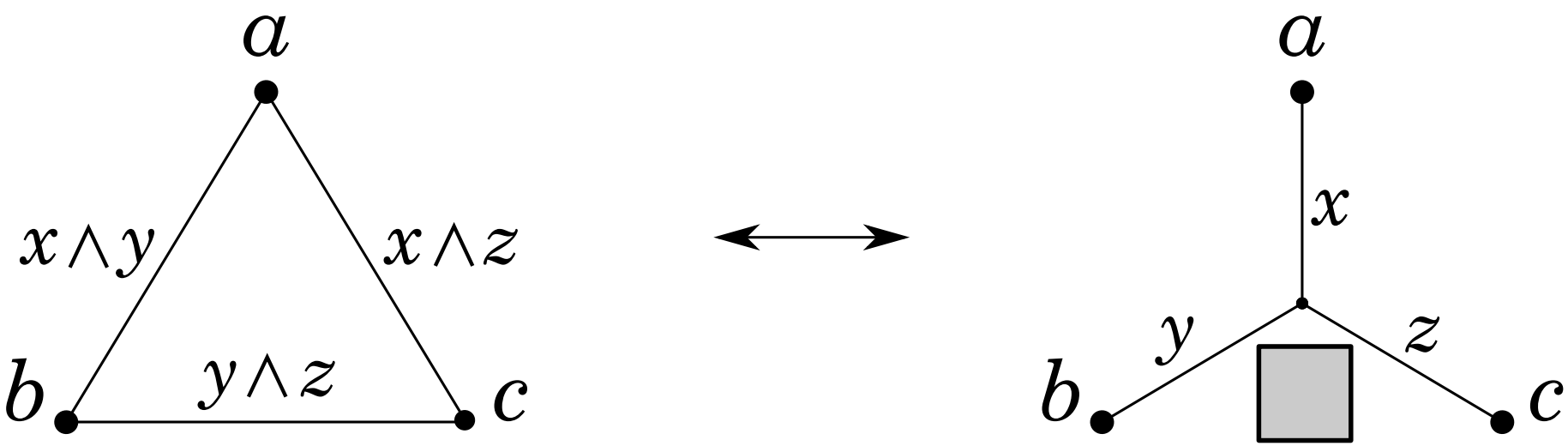
Equivalent transformations



$$a \overset{x}{\text{———}} b \quad = \quad (a + b) \overset{x}{\text{———}} (t_1 + t_2)$$

$$\backslash\{t_1, t_2\}$$

Equivalent transformations

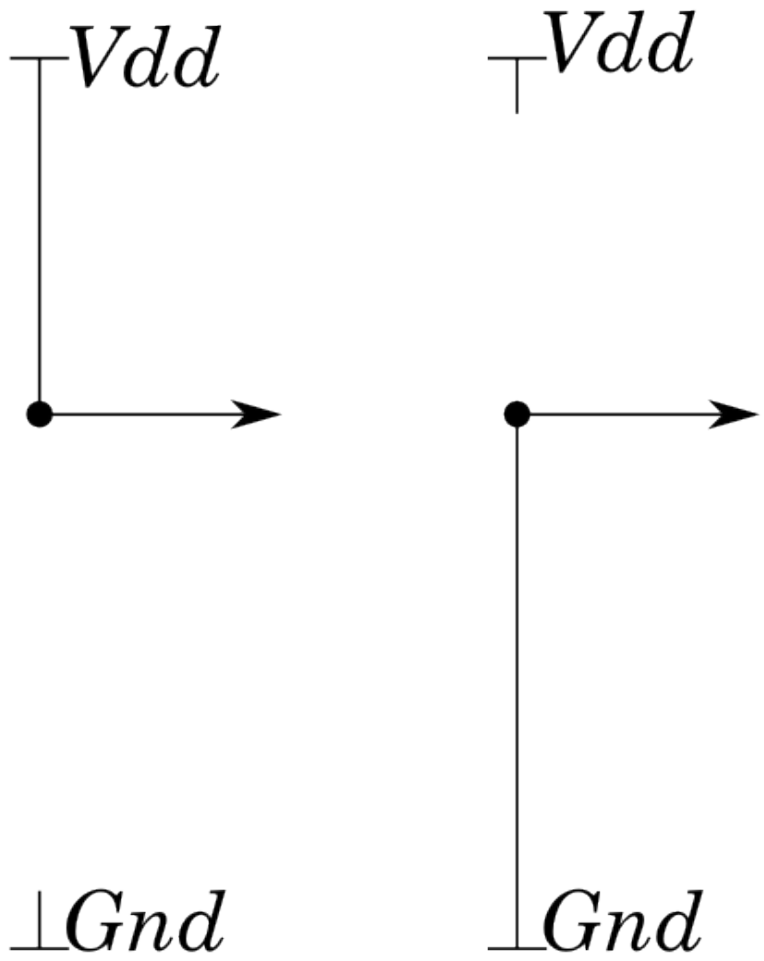


$$a \xrightarrow{x \wedge y} b\,+ \qquad a \xrightarrow{x} t\,+$$

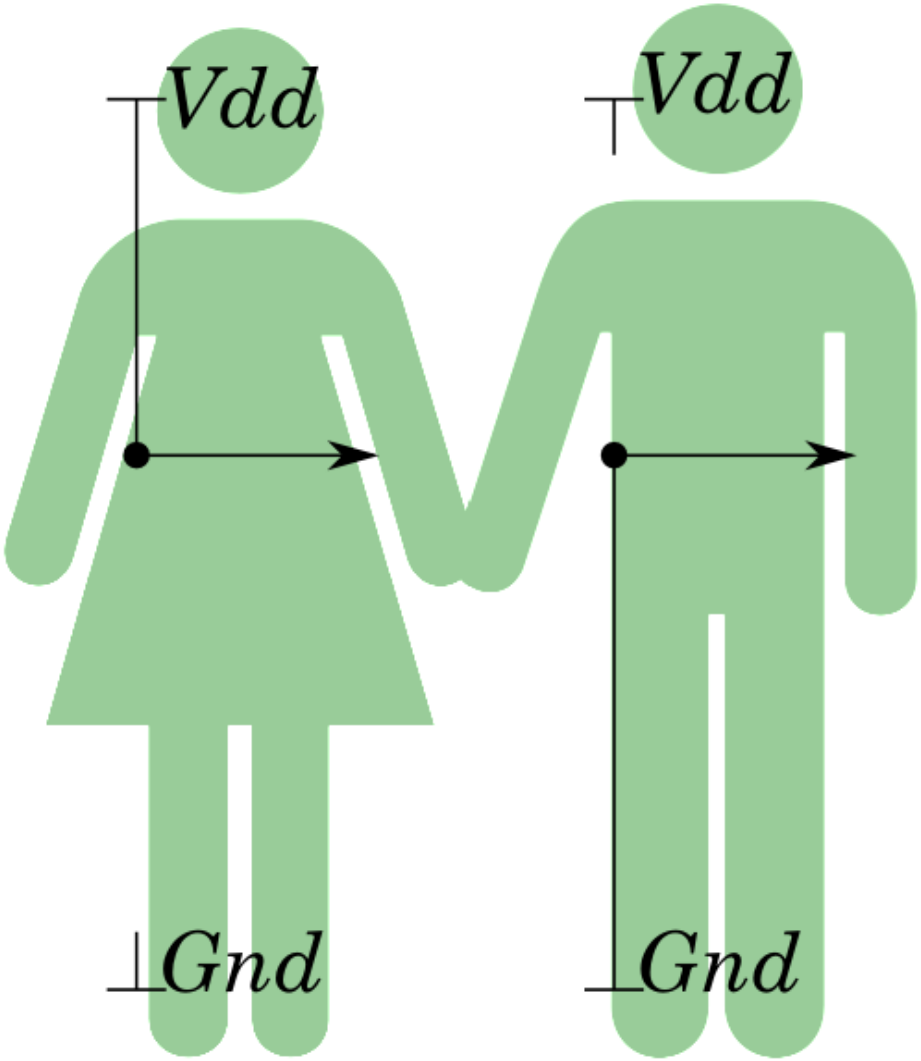
$$a \xrightarrow{x \wedge z} c\,+ \quad = \quad b \xrightarrow{y} t\,+ \qquad \backslash t$$

$$b \xrightarrow{y \wedge z} c \qquad c \xrightarrow{z} t$$

CMOS family of networks

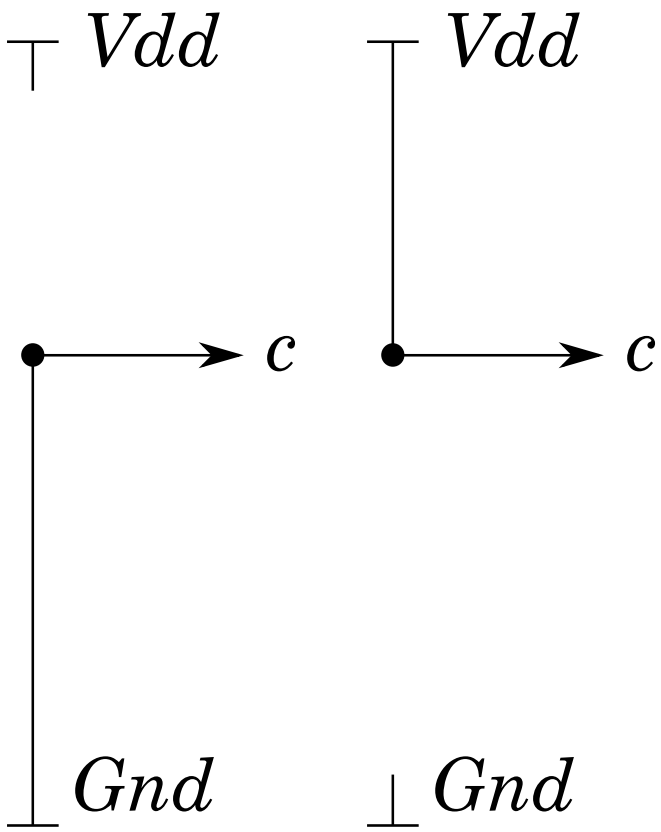


CMOS family of networks



NAND gate synthesis

$$\begin{cases} X = \top + c \text{ —— } \bot, \text{ if } a \wedge b \\ X = \top \text{ —— } c + \bot, \text{ if } \overline{a \wedge b} \end{cases}$$



NAND gate synthesis

$$A = B, \text{ if } F \quad \Leftrightarrow \quad [F]A = [F]B$$

NAND gate synthesis

$$\begin{cases} X = \top + c \text{ —— } \bot, \text{ if } a \wedge b \\ X = \top \text{ —— } c + \bot, \text{ if } \overline{a \wedge b} \end{cases}$$

$$\Downarrow$$

$$\begin{cases} [a \wedge b]X = [a \wedge b](\top + c \text{ —— } \bot) \\ [\overline{a \wedge b}]X = [\overline{a \wedge b}](\top \text{ —— } c + \bot) \end{cases}$$

NAND gate synthesis

$$\begin{cases} [a \wedge b]\mathbf{X} = [a \wedge b](\top + c \text{ —— } \bot) \\ [\overline{a \wedge b}]\mathbf{X} = [\overline{a \wedge b}](\top \text{ —— } c + \bot) \end{cases} \quad +$$

$\Downarrow$ by congruence

$$[a \wedge b]\mathbf{X} + [\overline{a \wedge b}]\mathbf{X} = [a \wedge b](\top + c \text{ —— } \bot) + [\overline{a \wedge b}](\top \text{ —— } c + \bot)$$

NAND gate synthesis

$$[a \wedge b]X + [a \overline{\wedge b}]X = [a \wedge b](\top + c \text{ — } \bot) + \overline{[a \wedge b]}(\top \text{ — } c + \bot)$$

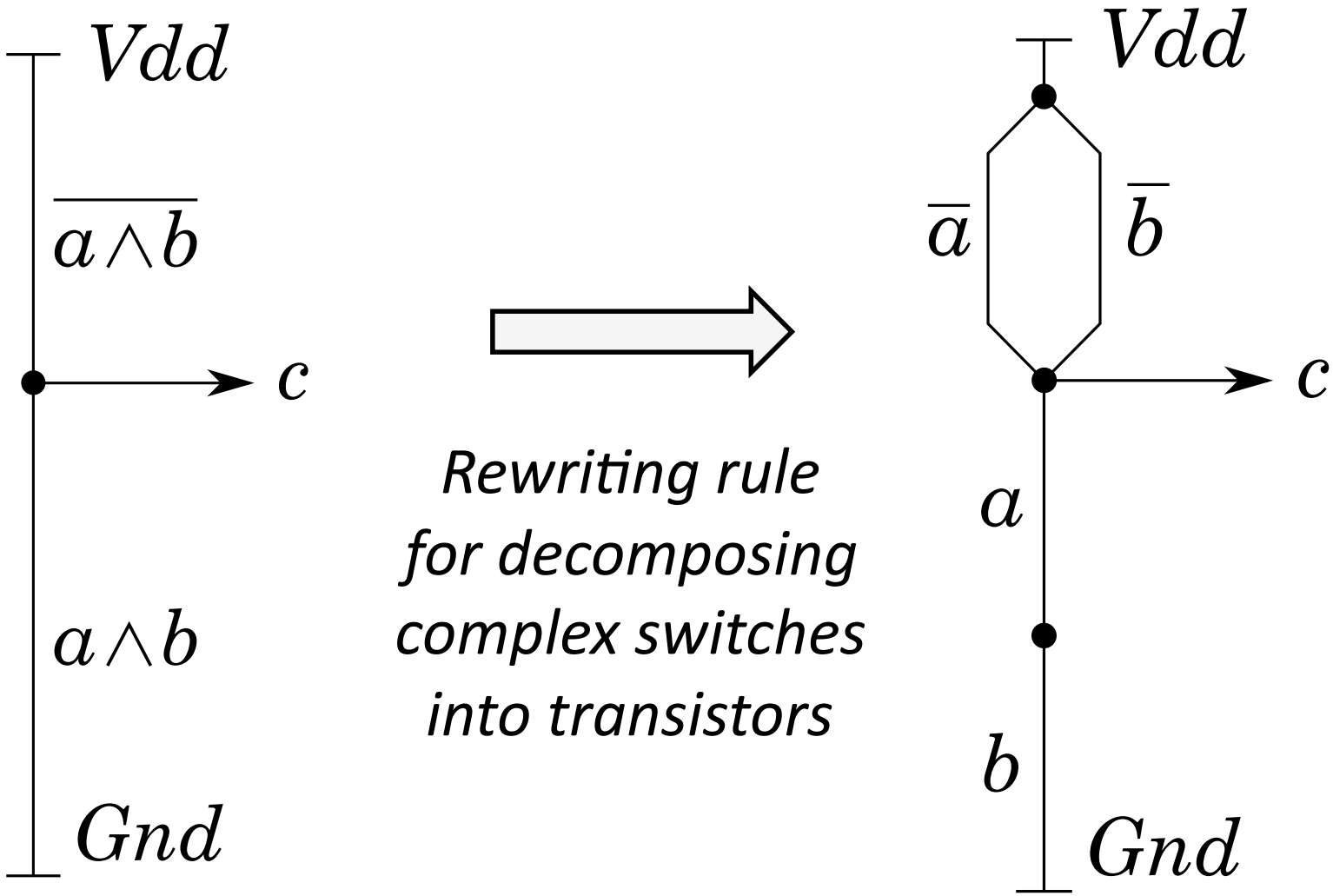
$\Downarrow$ factor

$$[(a \wedge b) \vee (a \overline{\wedge b})]X = [a \wedge b](\top + c \text{ — } \bot) + \overline{[a \wedge b]}(\top \text{ — } c + \bot)$$

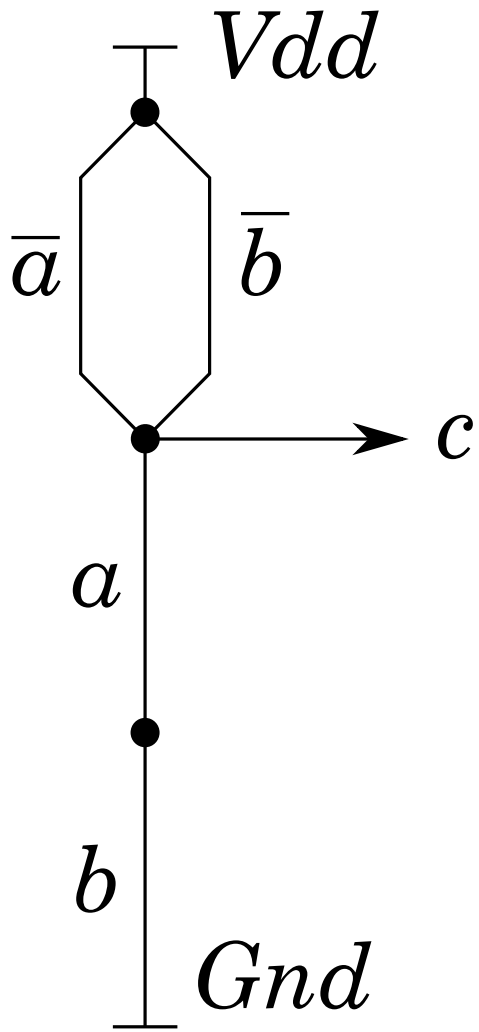
$\Downarrow$ simplify

$$X = [a \wedge b](\top + c \text{ — } \bot) + \overline{[a \wedge b]}(\top \text{ — } c + \bot)$$

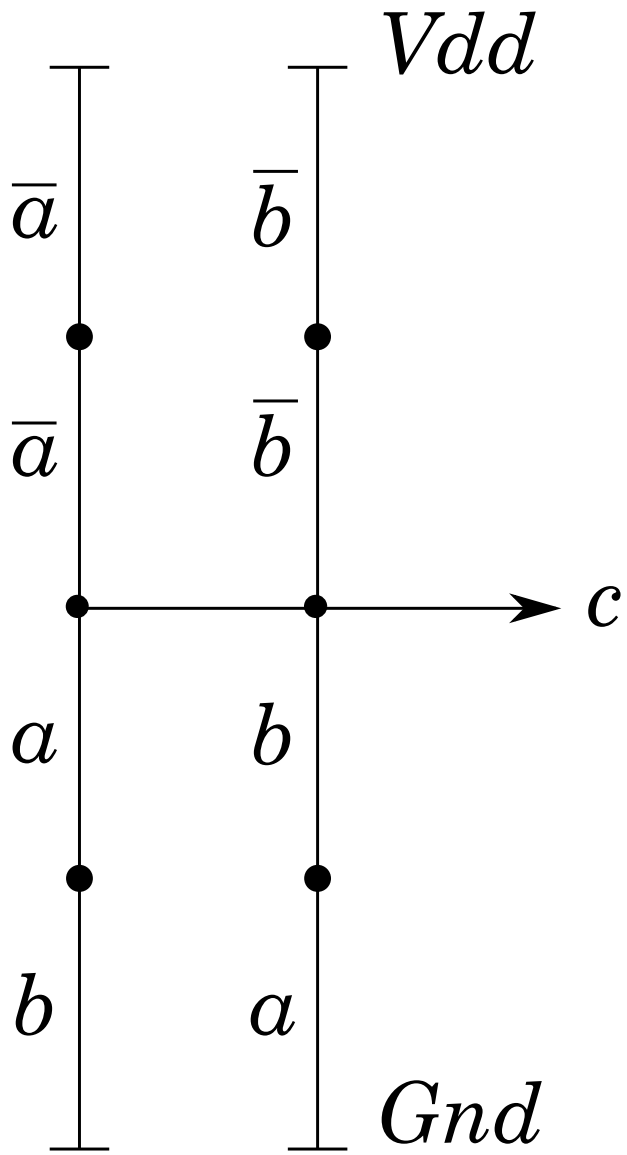
NAND gate synthesis



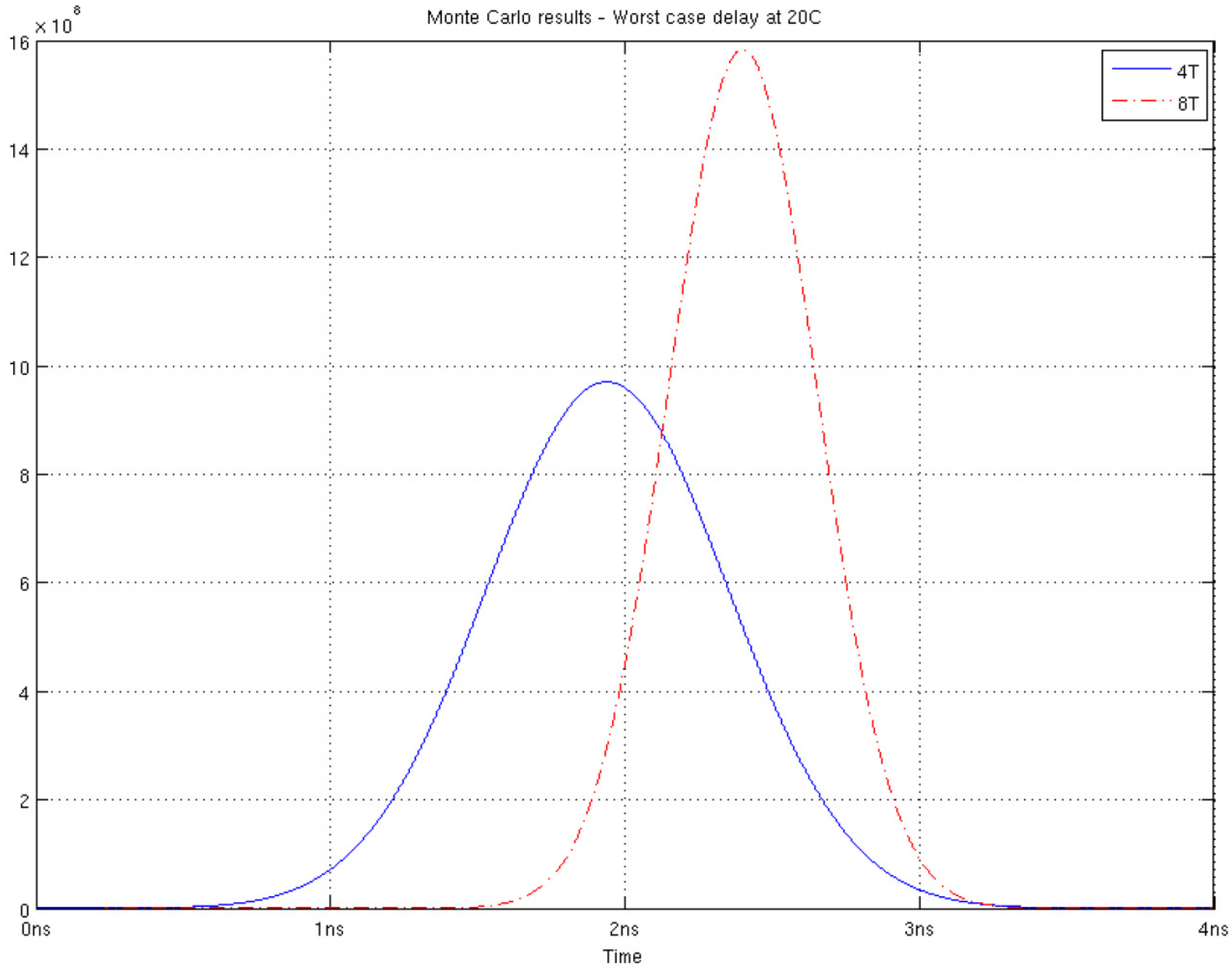
Decreasing uncertainty



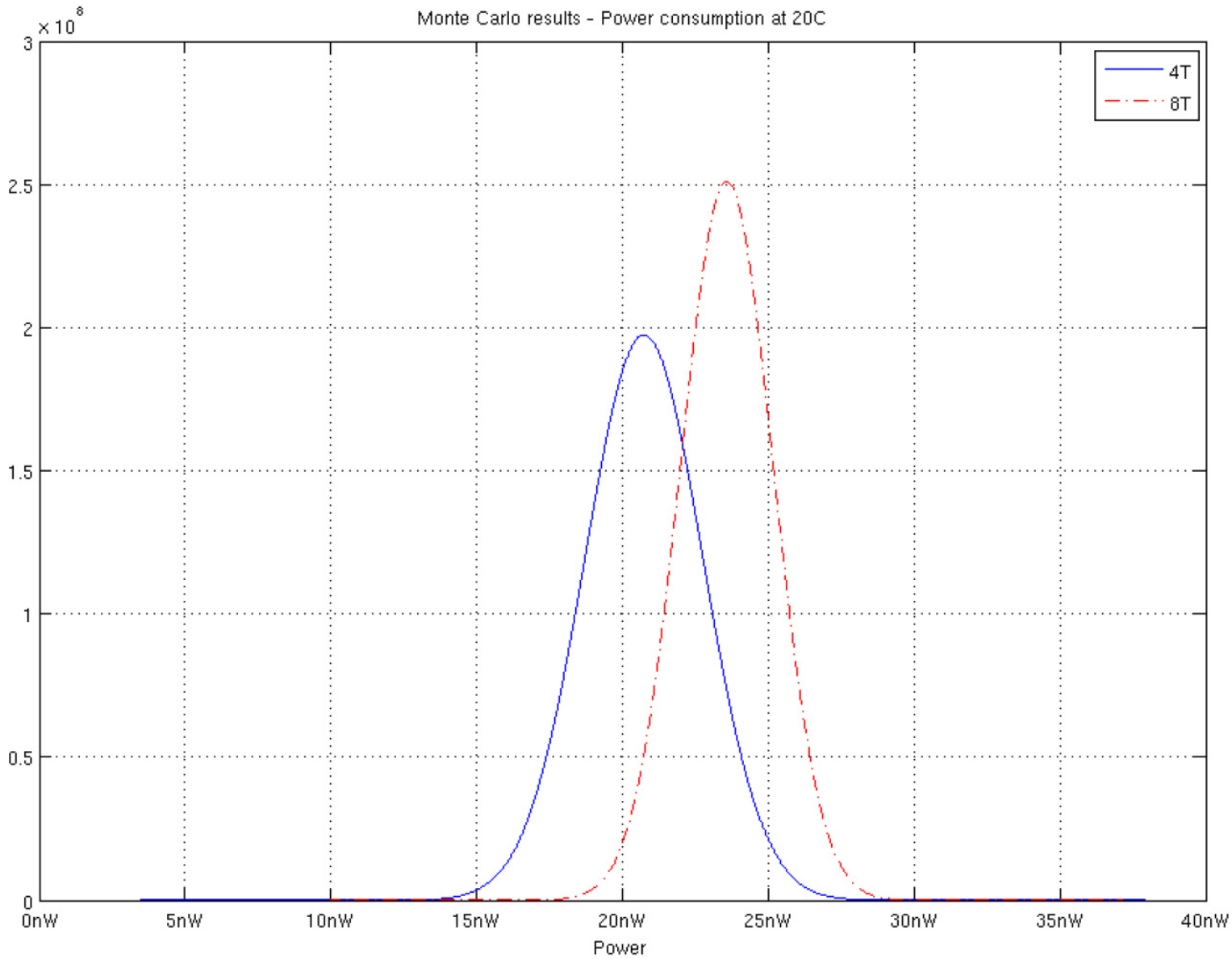
Rewriting rule for symmetric networks



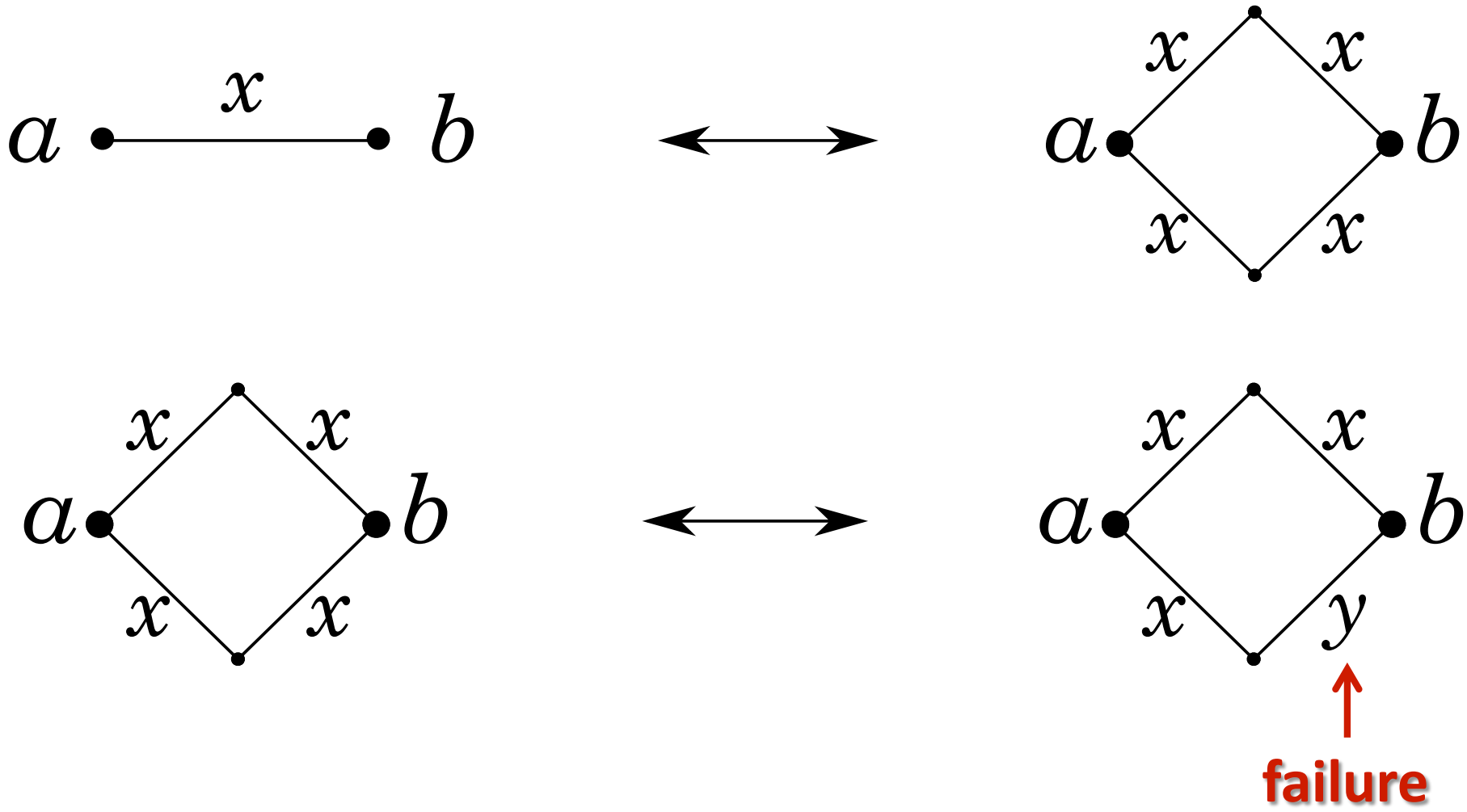
Decreasing uncertainty (delay)



Decreasing uncertainty (power)



Checking non-functional properties



Thank you!