from
Assembly Language and Computer Architecture
Using C and C++

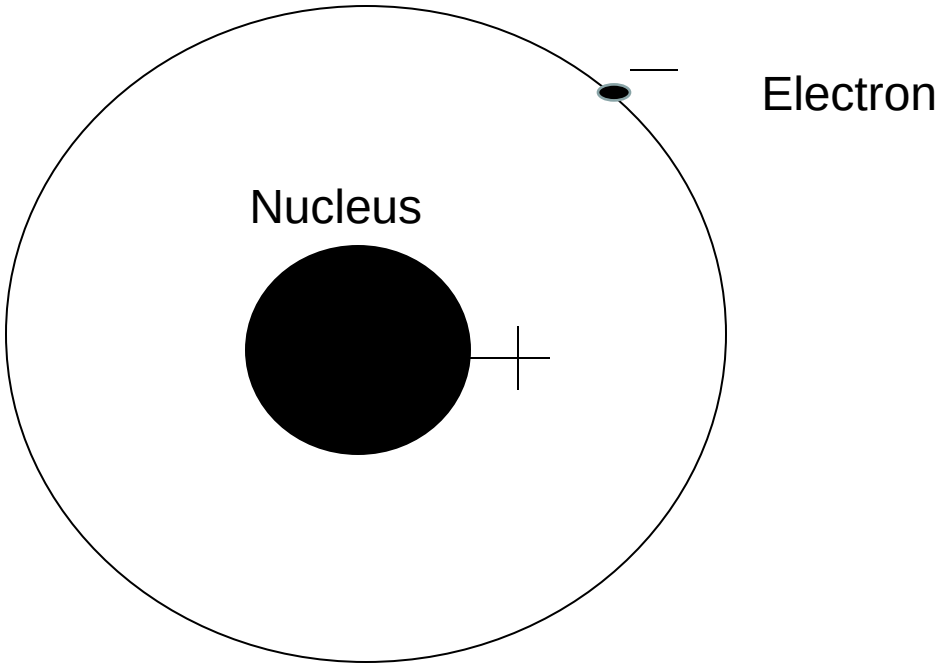
Constructing a Microprogrammed Computer

Basic Electronics and Digital Logic

Digital Circuits

- *Combinational*: a circuit whose output depends only on its present inputs
- *Sequential*: a circuit whose output depends on its past as well as present inputs. That is, it depends on the *sequence* of inputs from the past up to the present.

Atom has a positively-charged nucleus and negatively-charged orbiting electrons



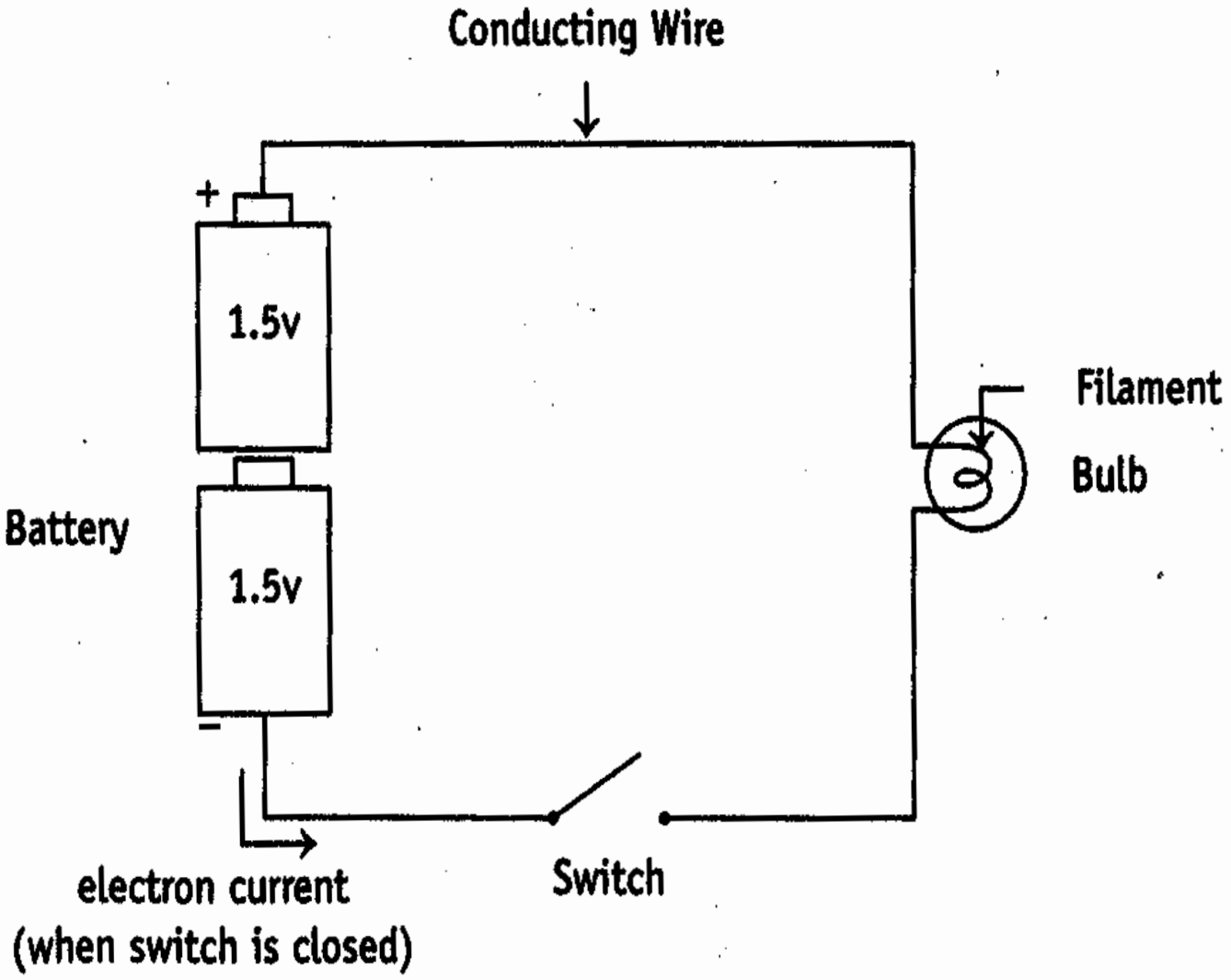
Like charges repel.
Unlike charges attract

A *conductor* has electrons that are not “stuck” to the nucleus.

An *insulator* has no “unstuck” electrons.

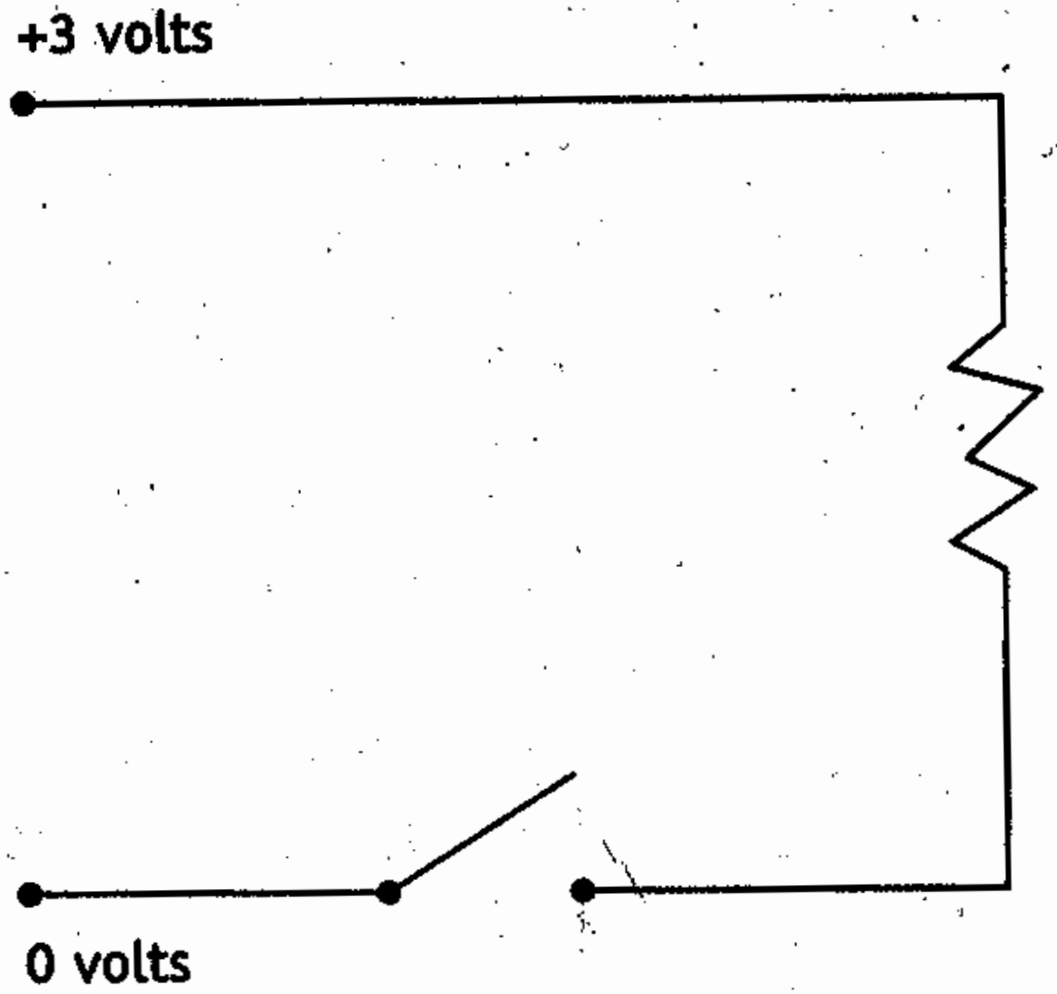
A flashlight is a simple electrical circuit

FIGURE 5.1 Flashlight



Electron current – flow of electrons
Voltage – provides “push” of electrons
Resistance – opposition to current

FIGURE 5.2 Schematic Diagram



Ohm's law

I is current in amps
E is voltage in volts
R is resistance in ohms

$$I = E/R$$

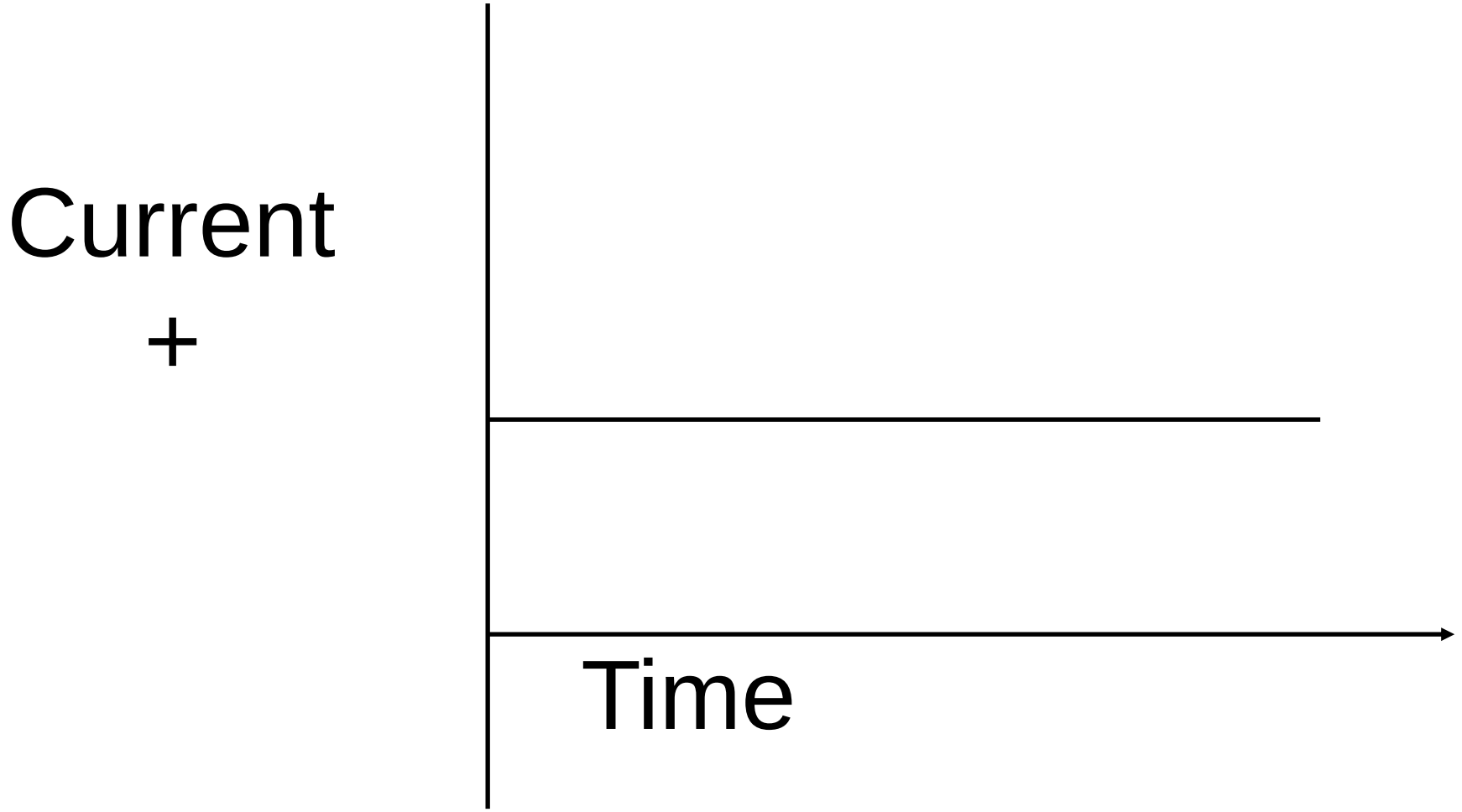
Solving for E in Ohm's law

$$E = IR$$

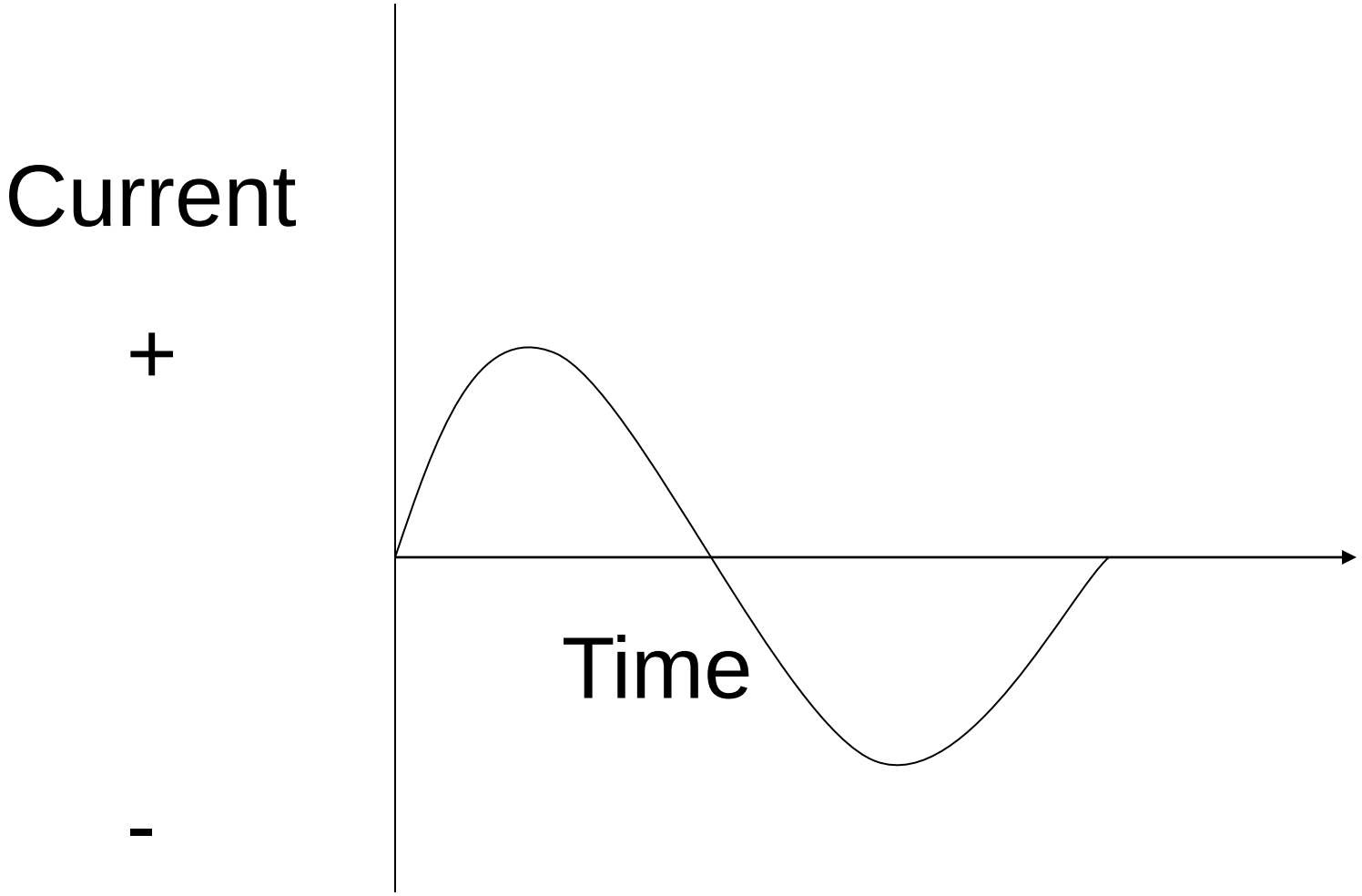
Important observation:

The voltage across a resistor (the *voltage drop*) is zero if I (the current) is zero.

Direct current (DC) flows in only one direction

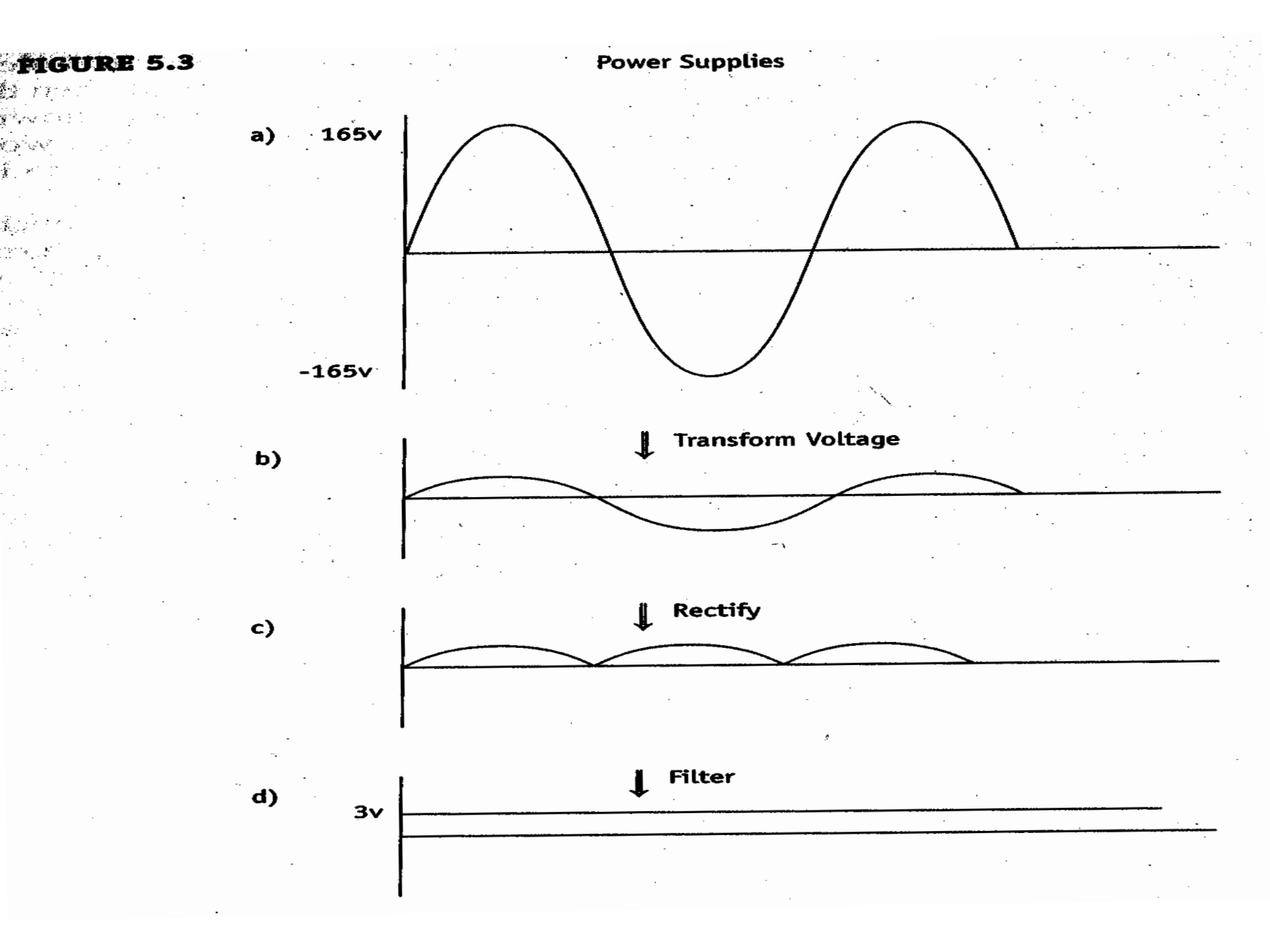


Alternating current (AC) repeatedly changes it direction of flow.



Computers circuits cannot use the 117 volt AC *line voltage* available by household electrical outlets. A computer *power supply* converts line voltage to the DC voltage required by computer circuits.

FIGURE 5.3


Power Supplies
a)
165v
-165v
b)
Transform Voltage
c)
Rectify
d)
Filter
3v

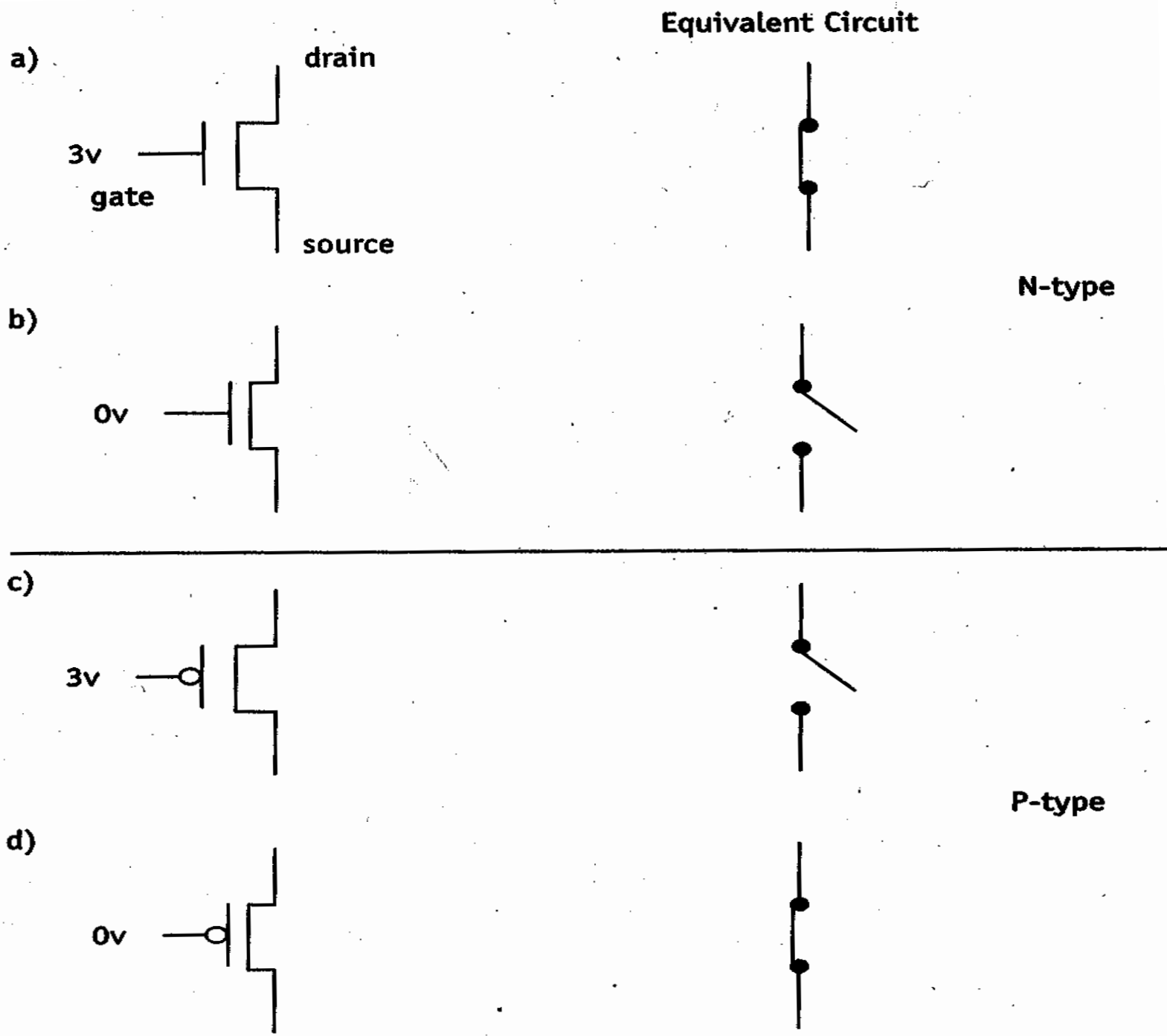
Safety rules with electricity

- Never touch any electrical equipment when any part of your body is wet.
- Never open electrical equipment unless you are qualified to do so.
- Keep one hand in your pocket while examining any electrical circuit.
- Never use any electrical equipment with frayed wires.

MOS transistors

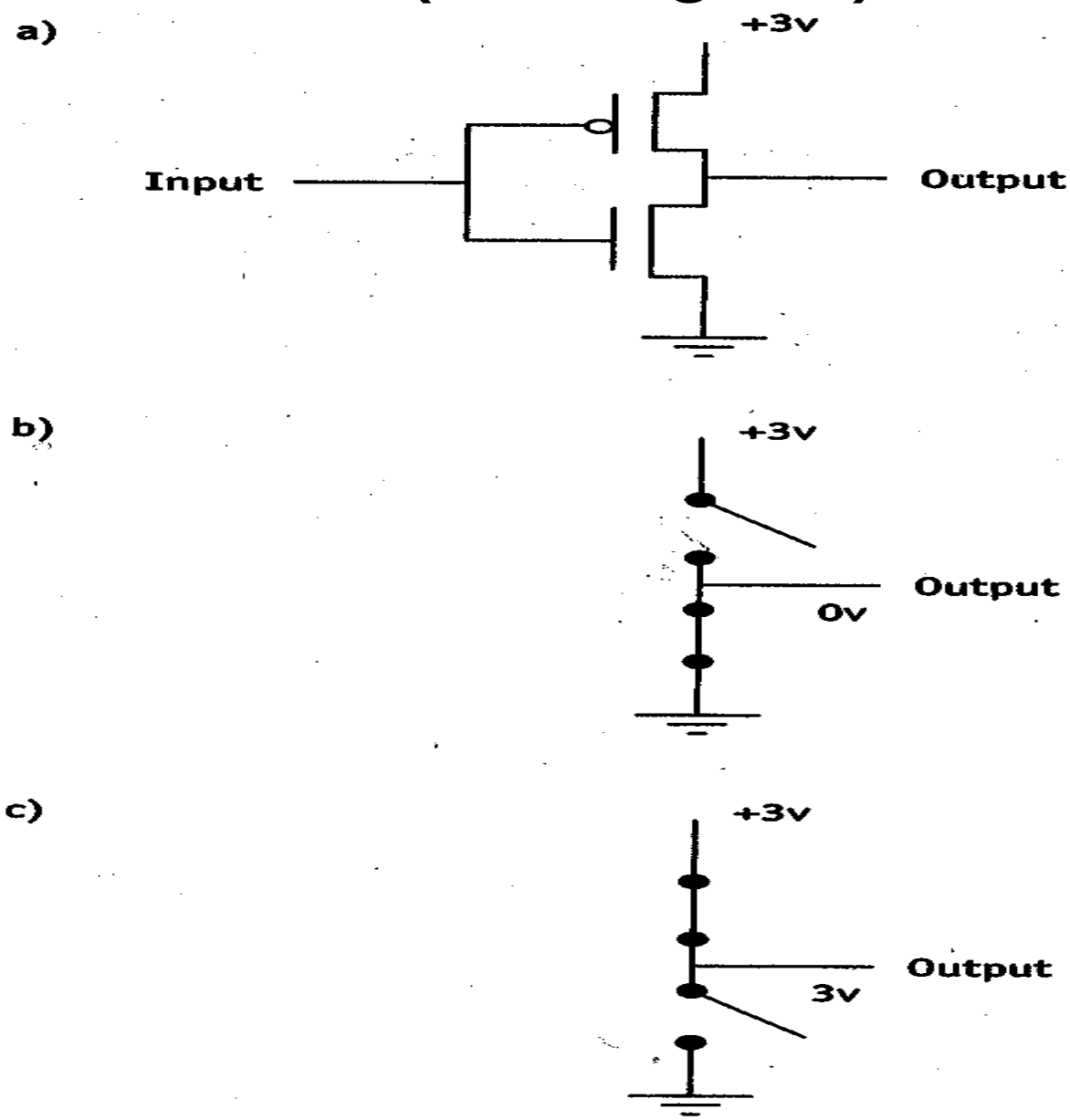
- Low power consumption
- High noise immunity
- High fan out
- Three leads: source, drain, gate
- Two types: PMOS (P-type), NMOS (N-type)
- CMOS technology: PMOS, NMOS used in pairs to minimize current requirements

FIGURE 5.4


Equivalent Circuit
a)
drain
3v
gate
source
N-type
b)
0v
c)
3v
P-type
d)
0v

Inverter (NOT gate)

FIGURE 5.5



Truth table for NOT

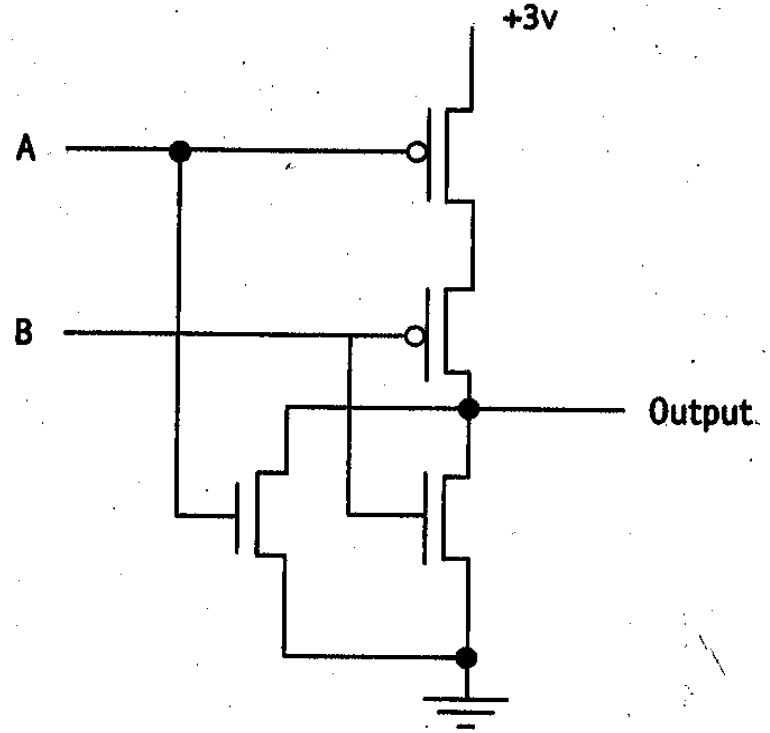
Input/output relationship:

in	out
0	3
3	1

If we let 0 and 3 volts represent the logic values 0 and 1, respectively, we get the truth table

in	out
0	1
1	0

FIGURE 5.6 a) NOR/NAND Function



b)

A	B	Output
0	0	3
0	3	0
3	0	0
3	3	0

c)

A	B	Output	
0	0	1	NOR
0	1	0	
1	0	0	
1	1	0	

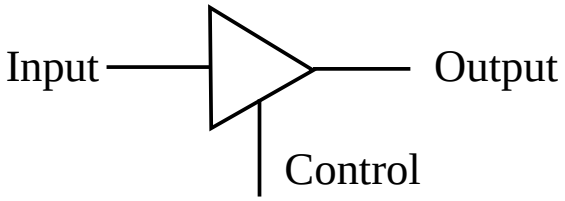
Positive logic

d)

A	B	Output	
1	1	0	NAND
1	0	1	
0	1	1	
0	0	1	

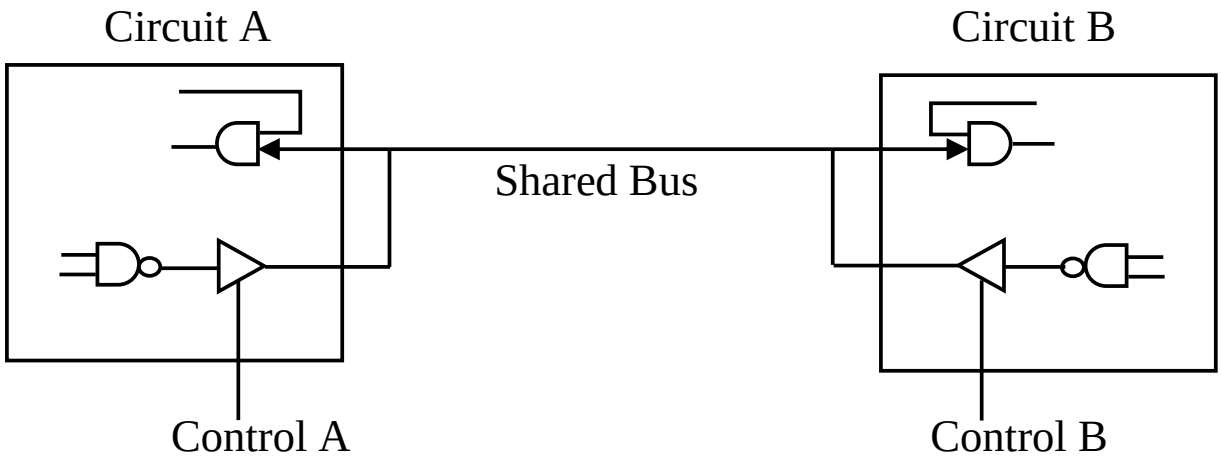
Negative logic

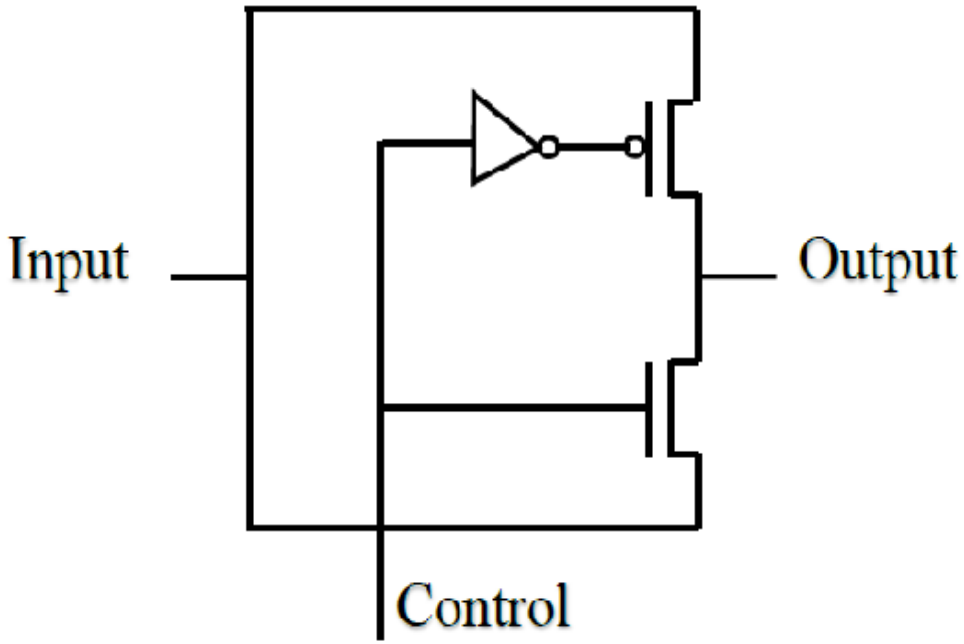
Tri-State Buffer

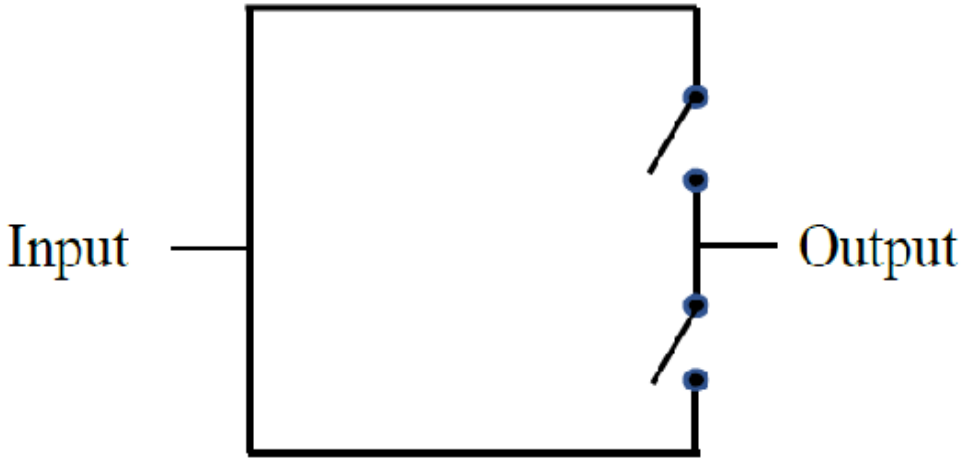


Input	Control	Output
0	0	High-Z
1	0	High-Z
0	1	0
1	1	1

Tri-State Buffer




Input
Output
Control


Input
Output

Positive logic – the higher voltage value represents 1
Negative logic – the lower voltage value represents 1

A *capacitor* stores an electrical charge. When a capacitor is charging or discharging, current is flowing (and heat is generated).

FIGURE 5.7

Capacitance

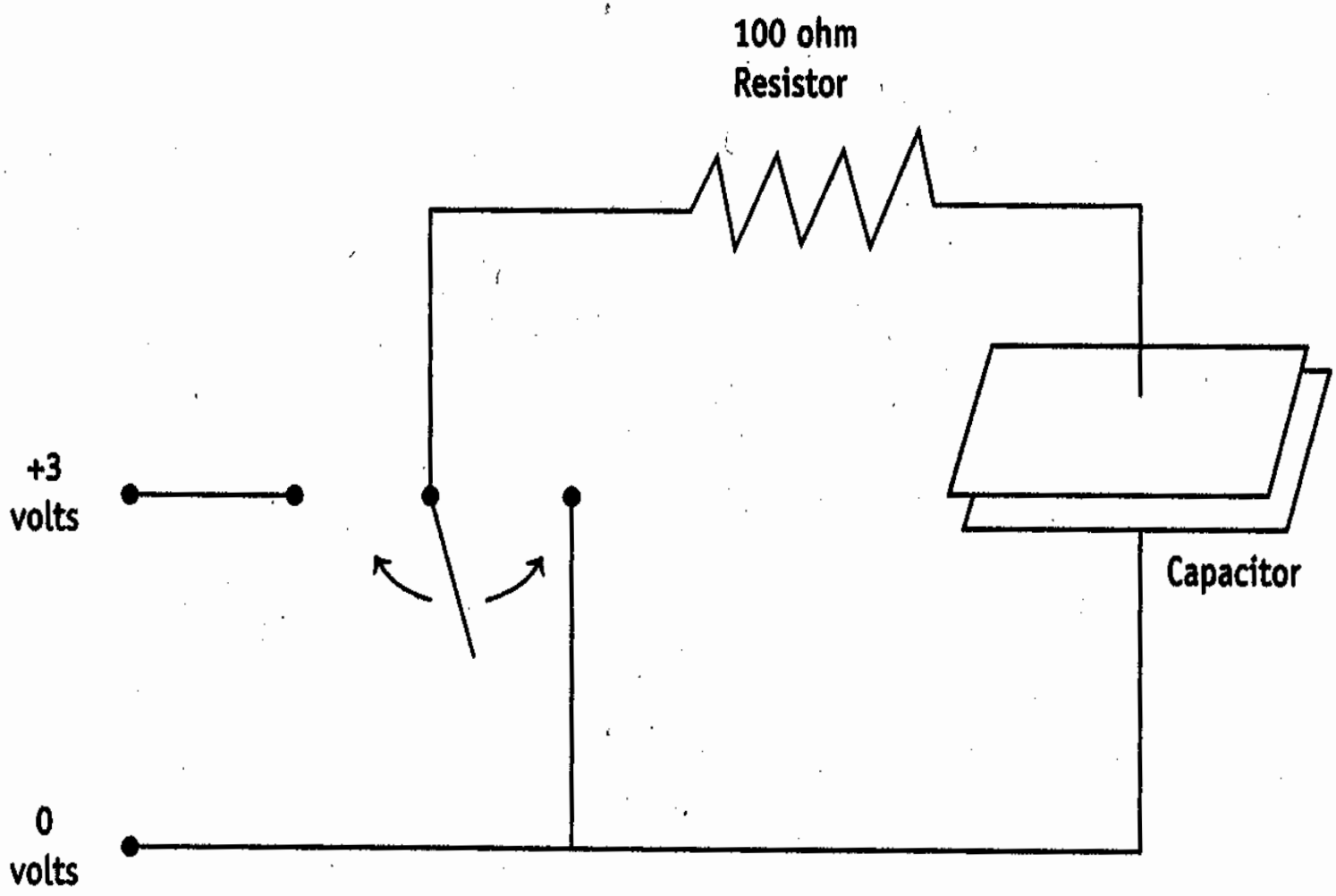
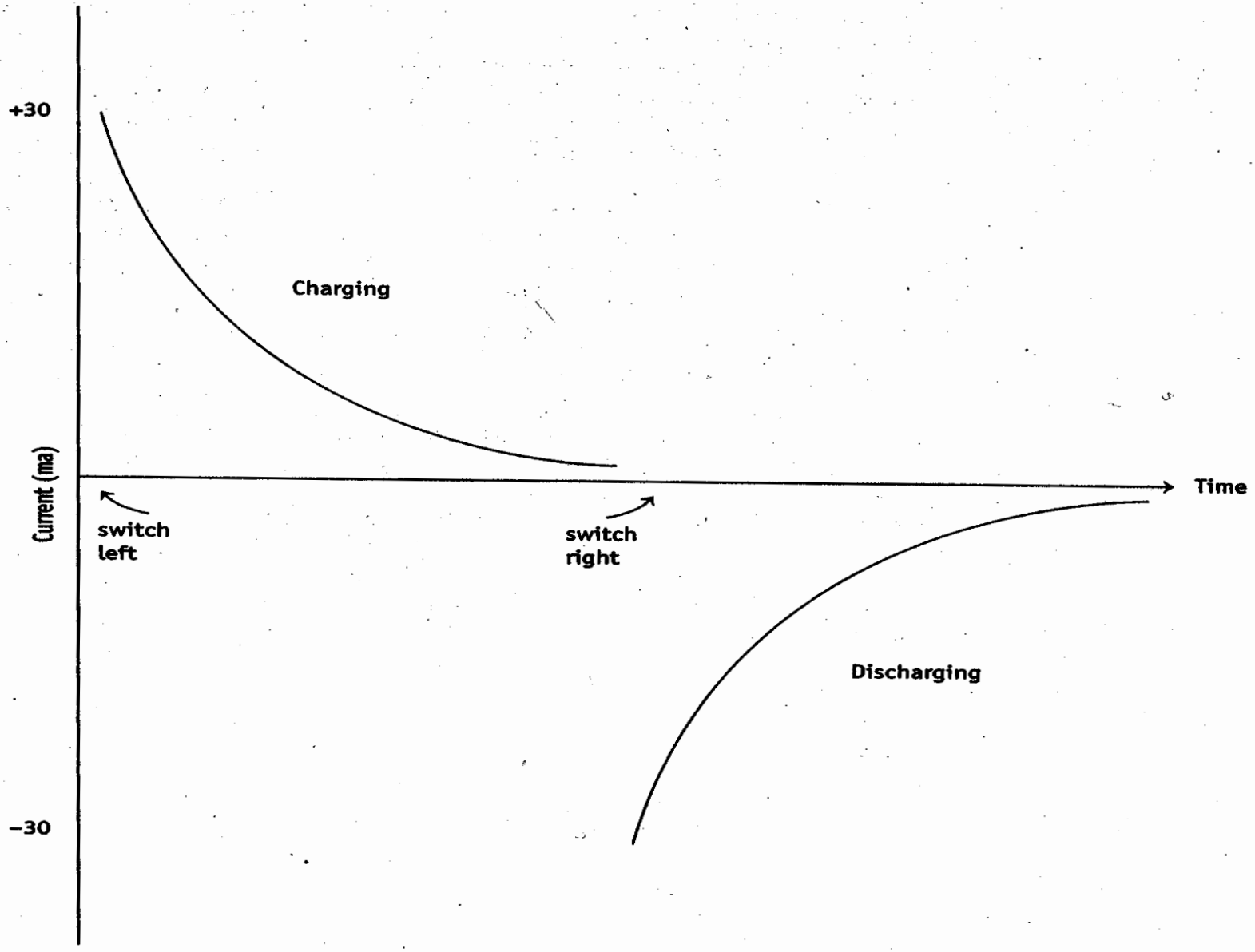


FIGURE 5.8 Resistor - Capacitor Circuit



The heat generated in our capacitor-resistor circuit is proportional to the frequency of voltage changes.

The heat generated is also proportional to the *square* of the magnitude of the voltage change.

To Make Computer Go Faster

- Increase clock speed
- Reduce size to decrease transit time from one subcircuit to another

Problem:

All circuits, including computer circuits, have capacitance.

Increasing clock frequency increases temperature.

Decreasing size increases temperature.

Computer circuits can be damaged by excessive temperature.

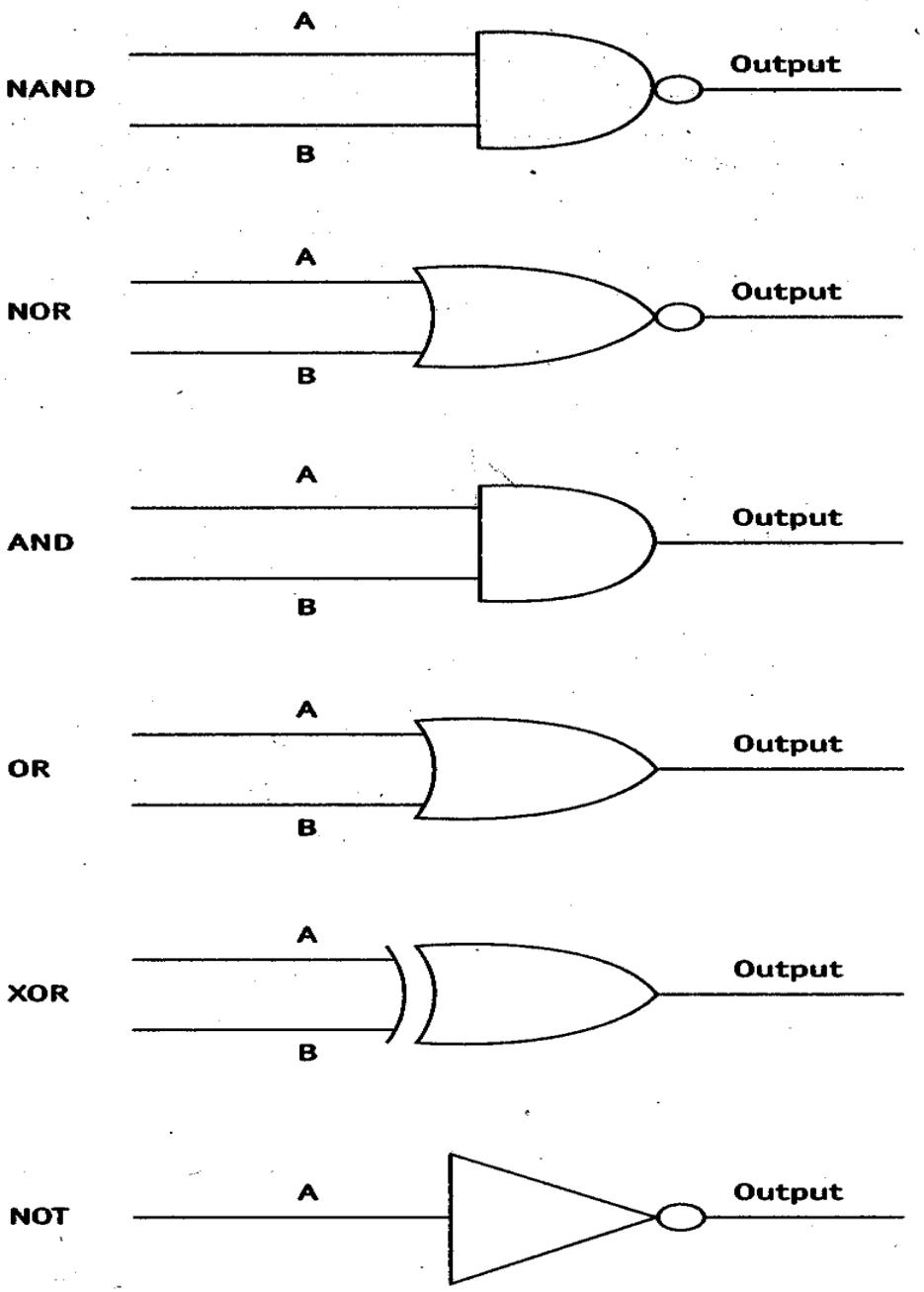
Solutions:

Reduce voltage to compensate for increasing frequency.
But must not make voltage differential too small.

Attach cooling devices.

Combinational Circuits

FIGURE 5.9



A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

A	Output
0	1
1	0

Building AND and OR gates

FIGURE 5.10

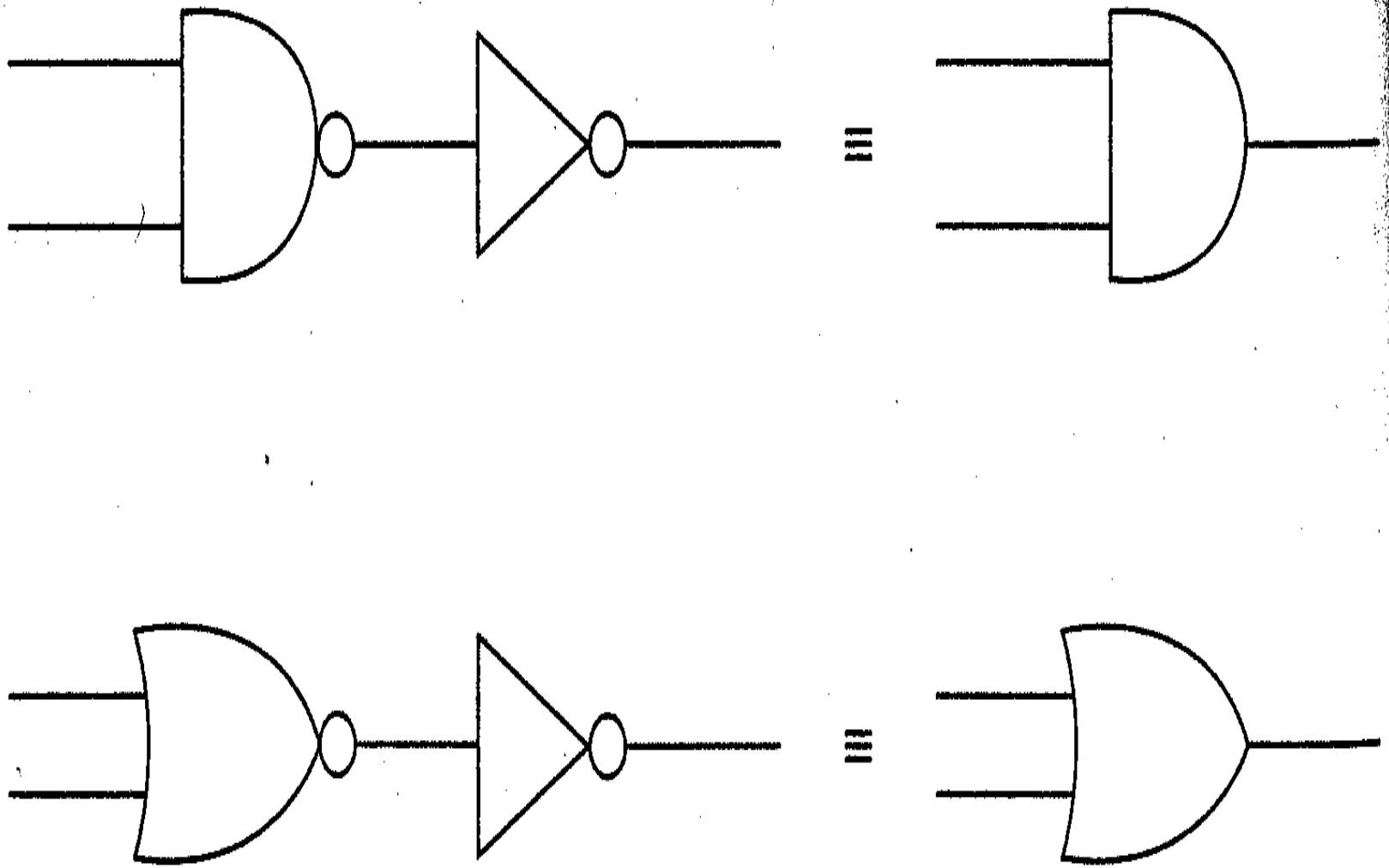
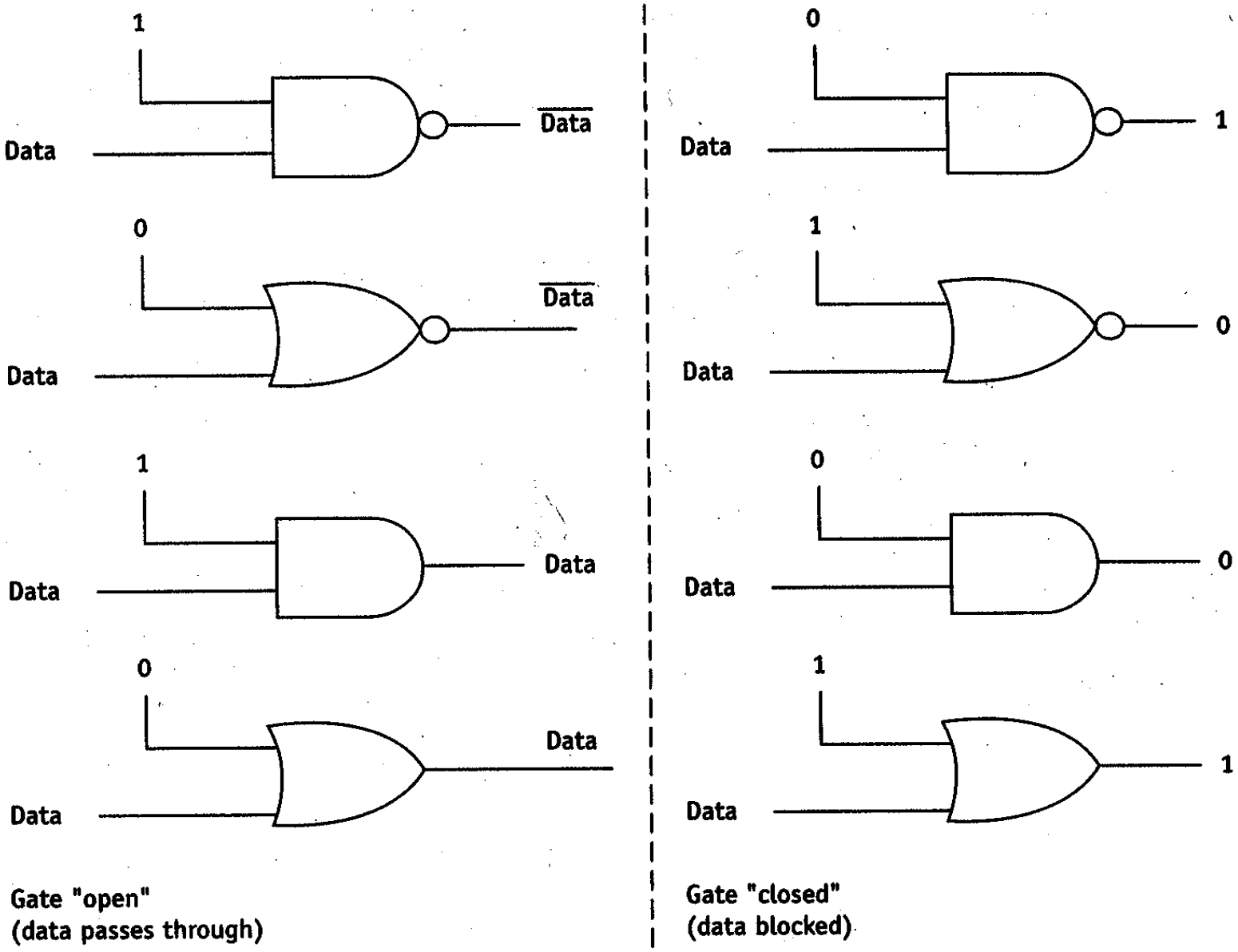


FIGURE 5.11 Gating Action



It is easy to construct the circuit corresponding to any truth table.

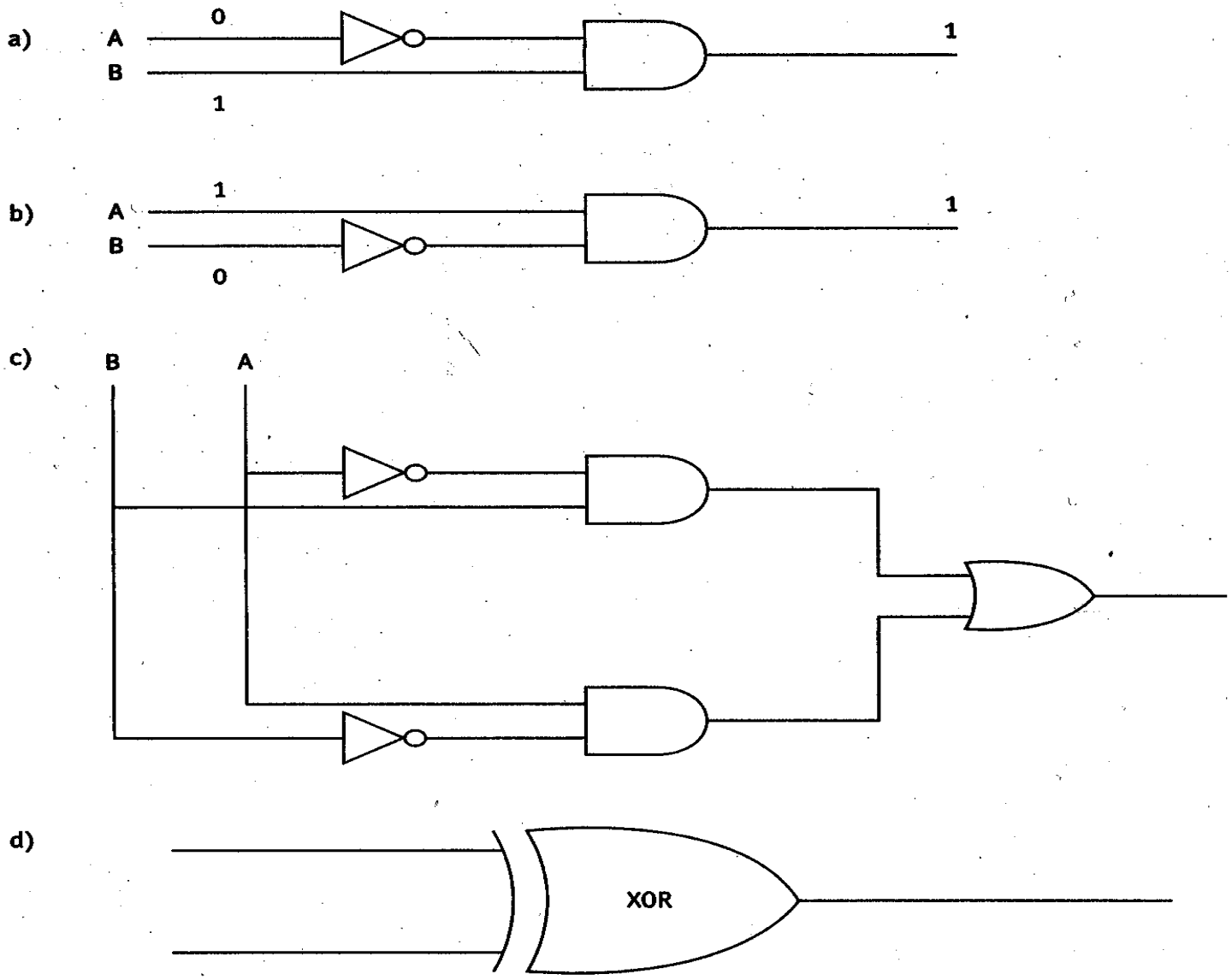
Implementing circuit from truth table

FIGURE 5.12

A	B	C	
0	0	0	
0	1	1	← select this row
1	0	1	← select this row
1	1	0	

Build a circuit for each row with a 1 output, and then connect using an OR gate.

FIGURE 5.13 Implementing a Function



Note the resulting circuit contains AND gates driving an OR gate.

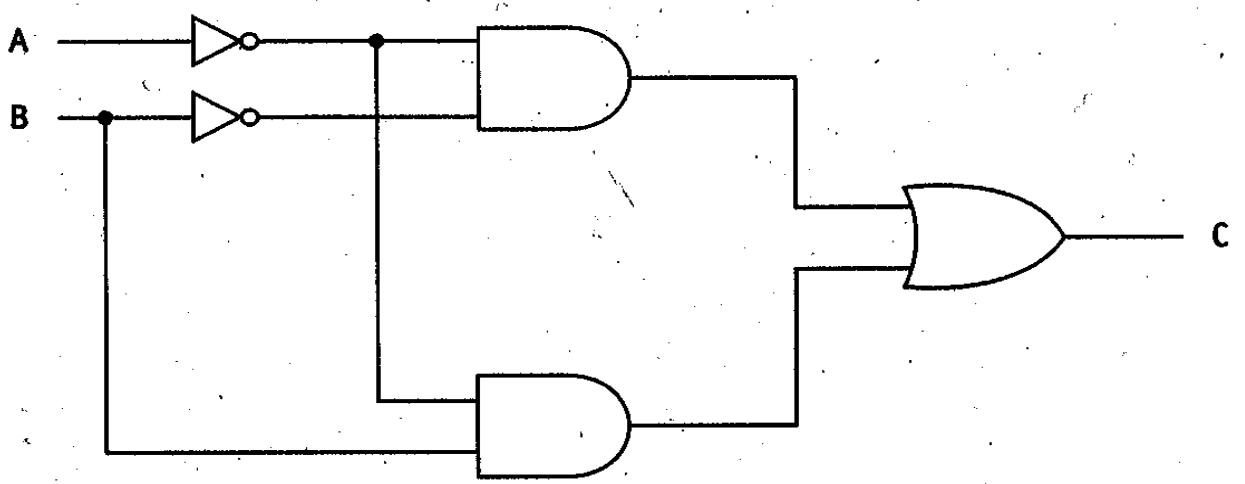
There is more than one way to implement a truth table. The circuit with the least number of gates is called the *minimal* circuit.

FIGURE 5.14 Minimization

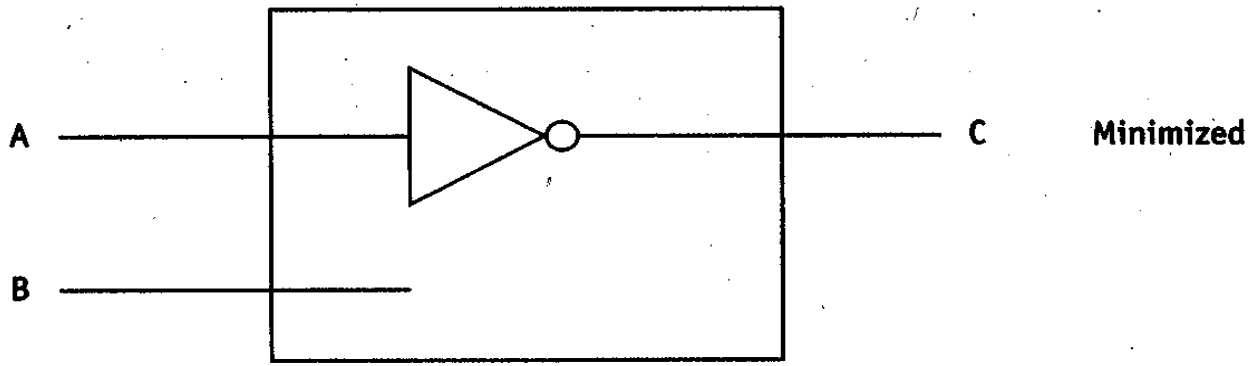
a)

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

b)



c)



Circuits can be minimized using any of several techniques. We will study two such techniques: Boolean algebra and Karnaugh maps.

Boolean expressions

- Describe operation of computer circuits
- ‘+’ represents OR
- Concatenation represents AND
- Superscript bar represents NOT
- Two values: 1 and 0

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

Describes of circuit on the next slide.

You can use Boolean algebra to simplify this expression (hint: factor out “A bar”).

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

Using these laws, we can manipulate Boolean expressions into equivalent simpler expressions. For example, let's use Boolean algebra to simplify

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

We get

$$\begin{aligned} C &= \overline{A}\,\overline{B} + \overline{A}\,B \\ &= \overline{A}(\overline{B} + B) && \text{by the distributive law} \\ &= \overline{A}(1) && \text{by the inverse law} \\ &= \overline{A} && \text{by the identity law} \end{aligned}$$

The laws of Boolean algebra are not identical to the laws of real number algebra.

FIGURE 5.15

Distributive Law	$A(B + C) = AB + AC$
	$A + BC = (A + B)(A + C)$
Commutative Law	$AB = BA$
	$A + B = B + A$
Absorption Law	$A + AB = A$
	$A(A + B) = A$
Identity Law	$A1 = A$
	$A + 0 = A$
Null Law	$A0 = 0$
	$A + 1 = 1$
Idempotent Law	$A + A = A$
	$AA = A$
Inverse Law	$A\overline{A} = 0$
	$A + \overline{A} = 1$
Associative Law	$(A + B) + C = A + (B + C)$
	$(AB)C = A(BC)$
DeMorgan's Laws	$\overline{A + B} = \overline{A}\,\overline{B}$
	$\overline{AB} = \overline{A} + \overline{B}$

Bye, bye black sheep rule:
If two terms differ in only one variable, may combine terms by eliminating “black sheep” variable.

$$ABCD + AB\overline{C}D$$

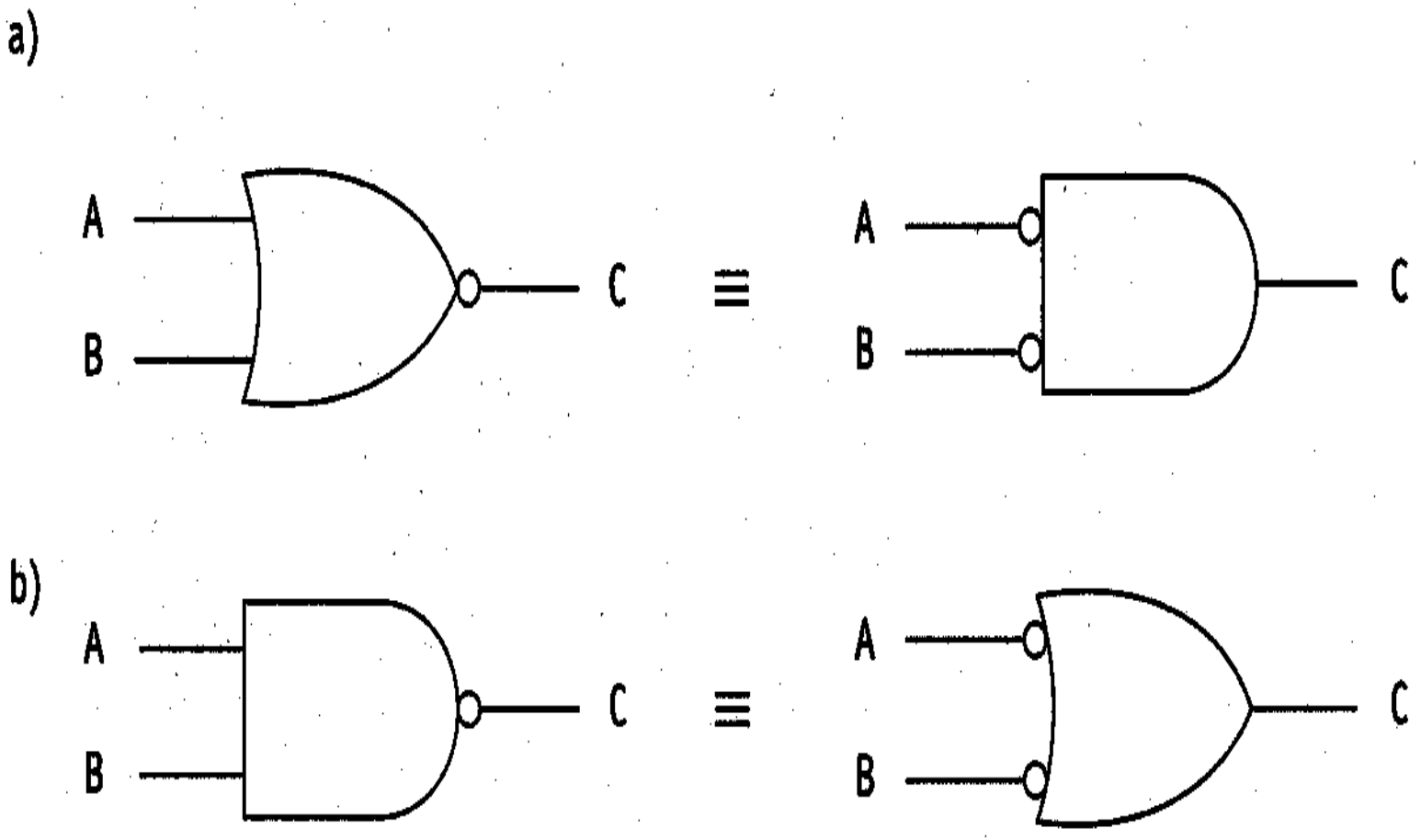
Black sheep rules yields single term **ABD**

DeMorgan's Laws

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

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

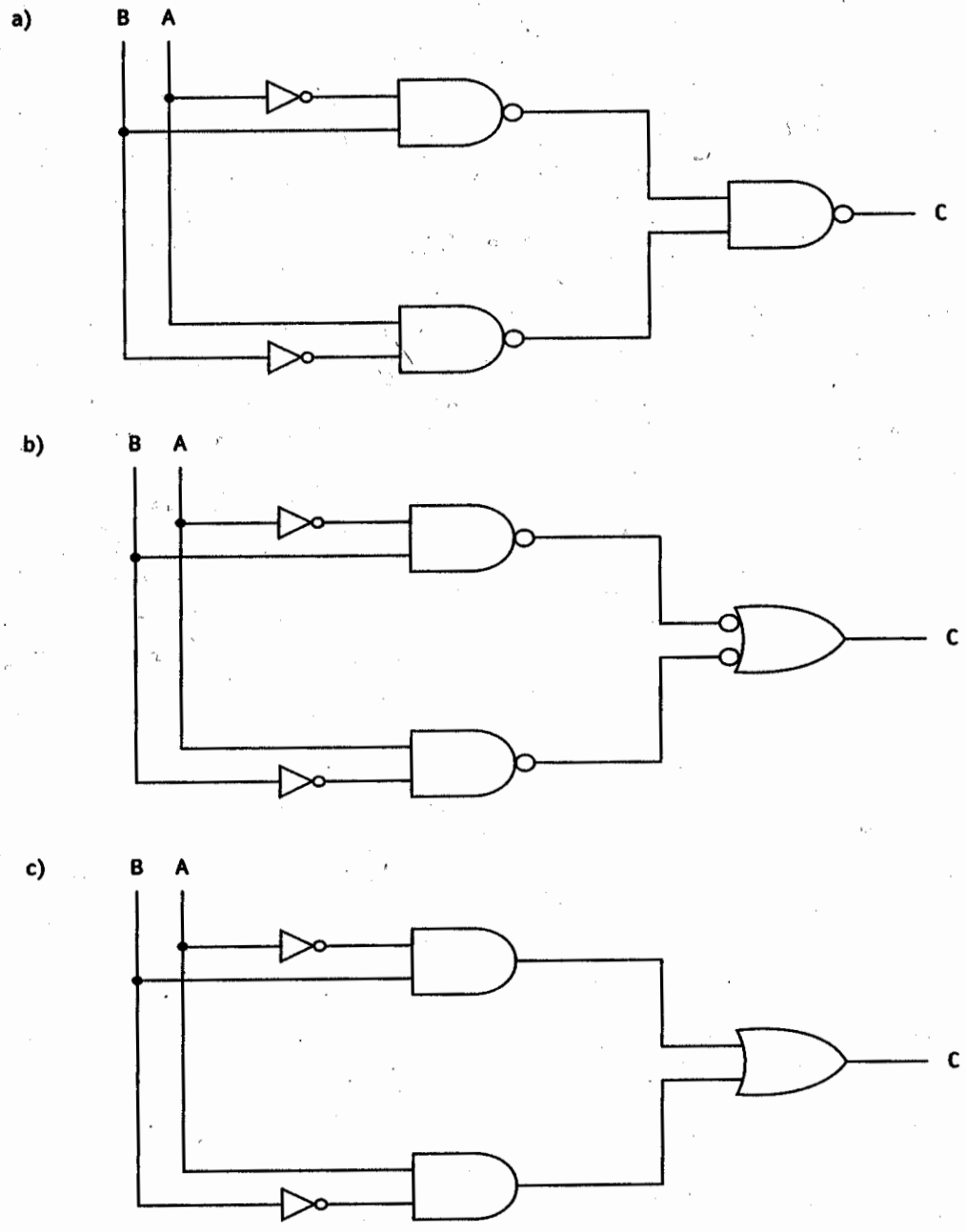
FIGURE 5.16 De Morgan's Laws



Move bubble back: gate changes

In a circuit consisting of AND gates driving a single OR gate, all AND and OR gates can be replaced by NAND gates. The resulting circuit will realize the same function.

FIGURE 5.17 NAND Gate Substitution

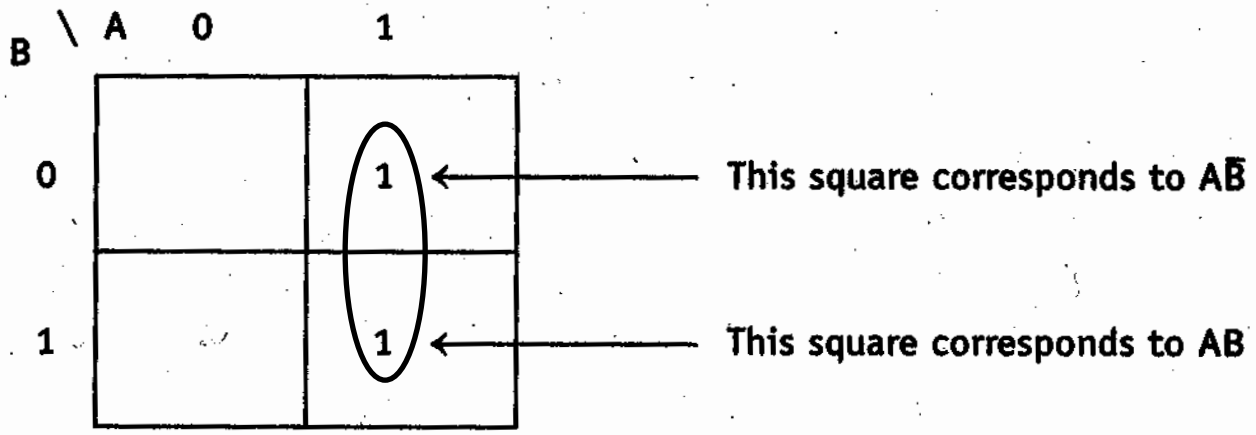


Karnaugh map

- Matrix representation of a Boolean function
- Useful for circuit minimization
- Horizontally and vertically adjacent squares differ in only one variable. Thus, horizontal and vertical groups correspond to terms simplified by the black sheep rule.

FIGURE 5.18

a)



b)

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

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

Black sheep rule yields

$$C = A$$

When the function below is simplified, you have to use AB twice.

$$
\begin{aligned}
C &= A\overline{B} + AB + \overline{A}B \\
&= A\overline{B} + AB + AB + \overline{A}B && \text{by the idempotent law} \\
&= (A\overline{B} + AB) + (AB + \overline{A}B) && \text{by the associative law} \\
&= A + (AB + \overline{A}B) && \text{by the black sheep rule} \\
&= A + B && \text{by the black sheep rule}
\end{aligned}
$$

Now simplify same function using Karnaugh map. Note: using bottom right term (AB) twice

FIGURE 5.19

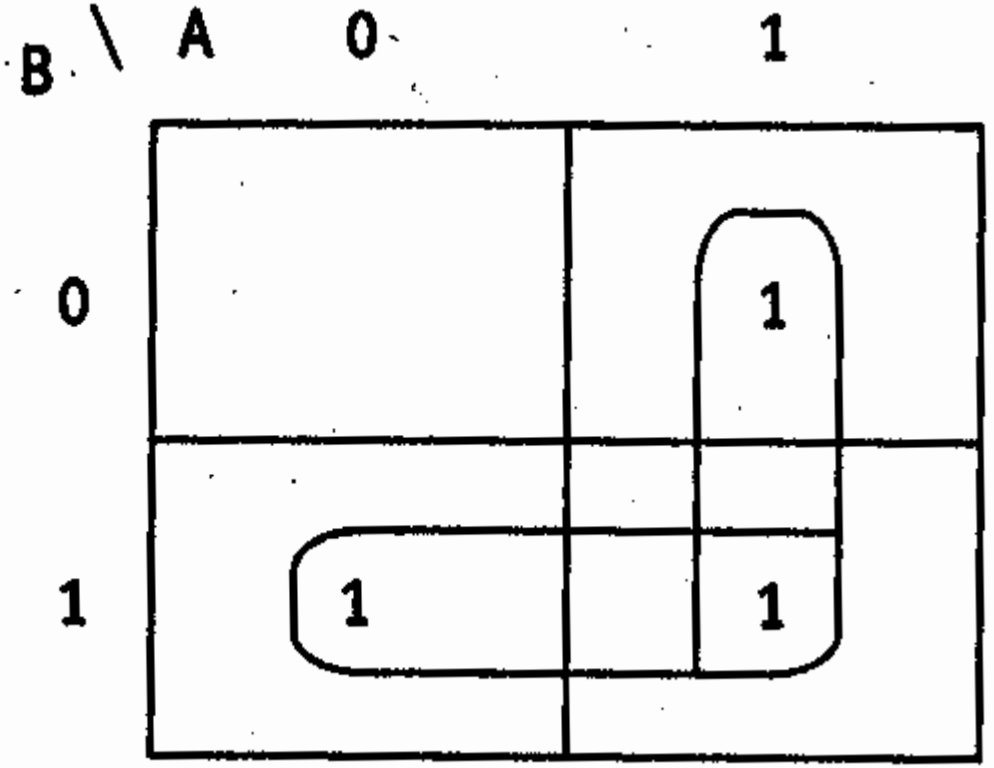
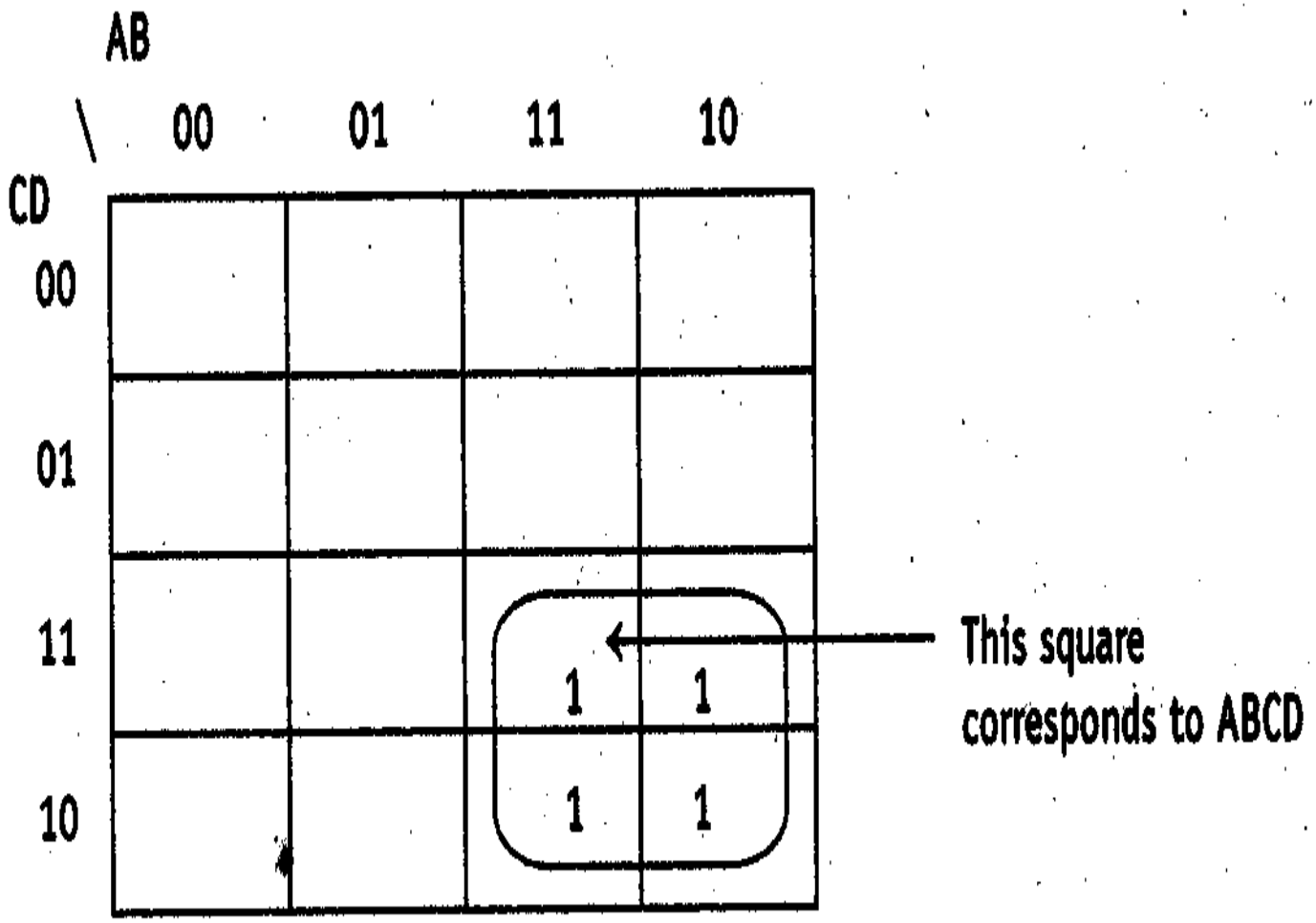
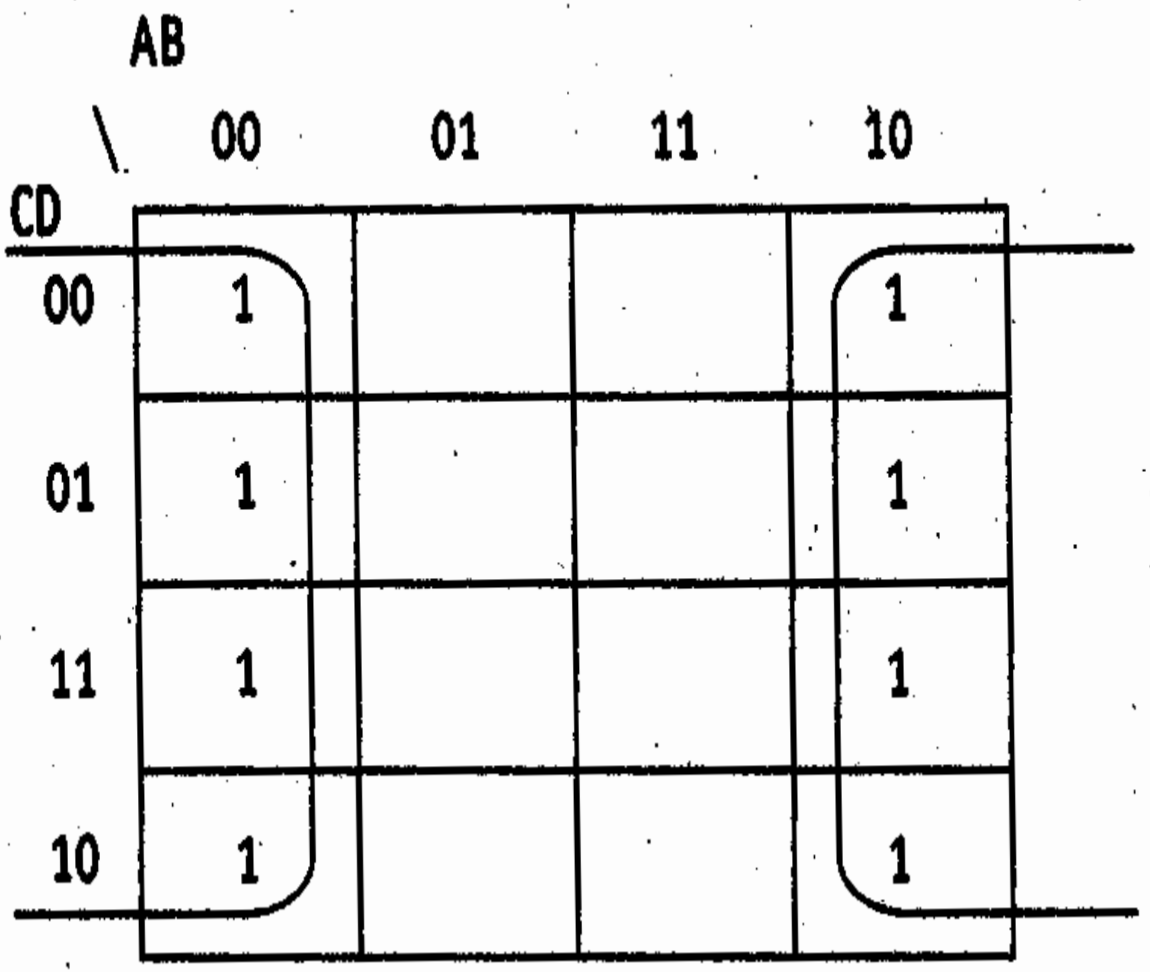


FIGURE 5.20



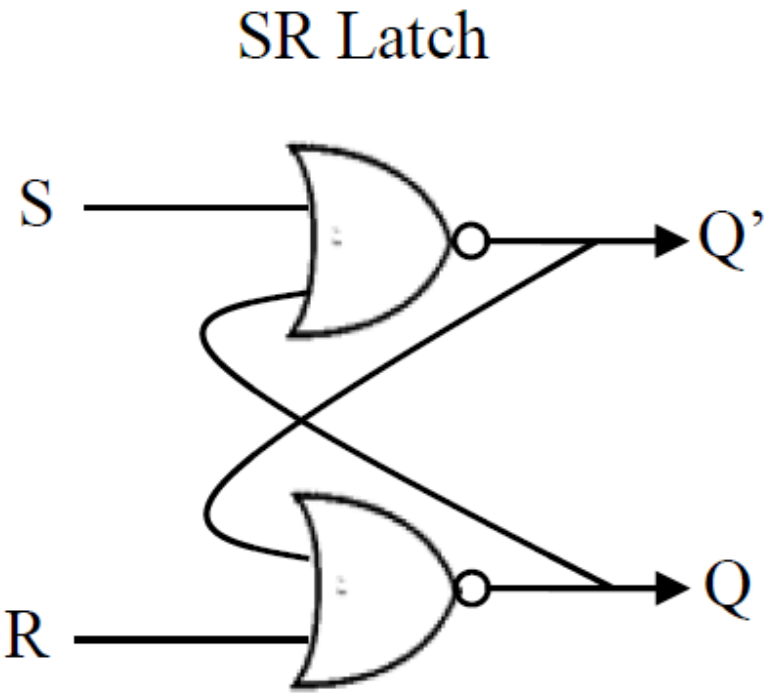
Group corresponds to AC

FIGURE 5.21

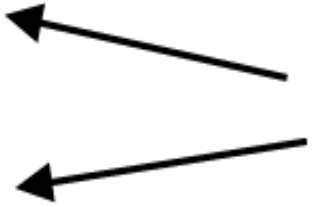


Group corresponds to $\overline{B}$

Sequential Circuits

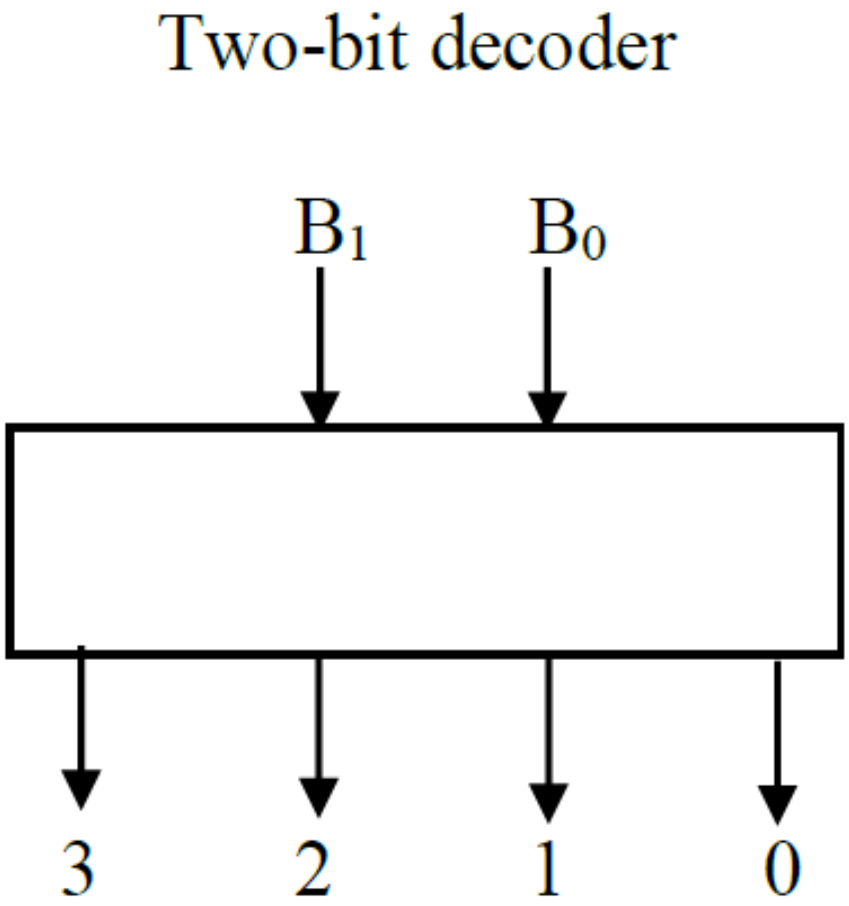


S	R	Q	Q’
0	1	0	1
0	0	0	1
1	0	1	0
0	0	1	0
1	1	0	0

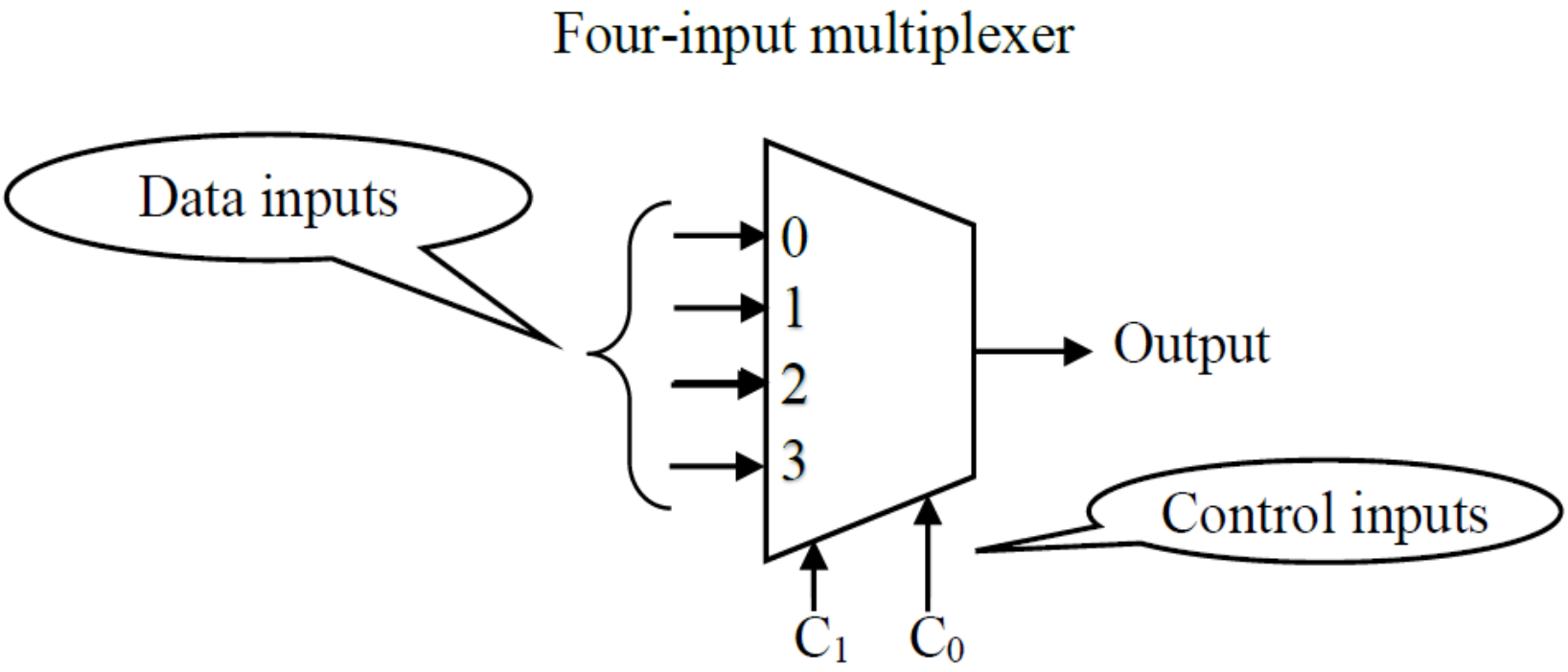


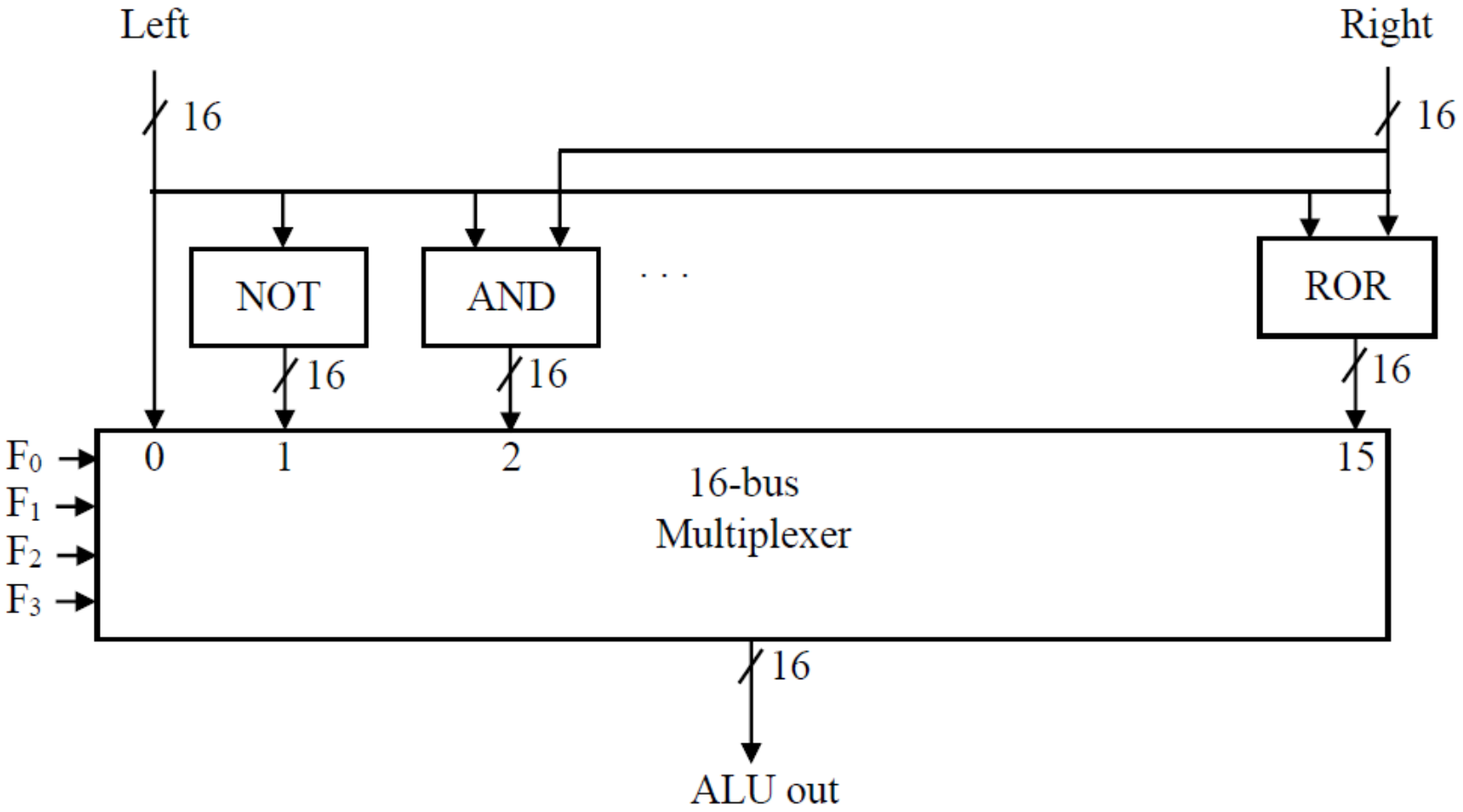
Two rows for S = R = 0

Decoder

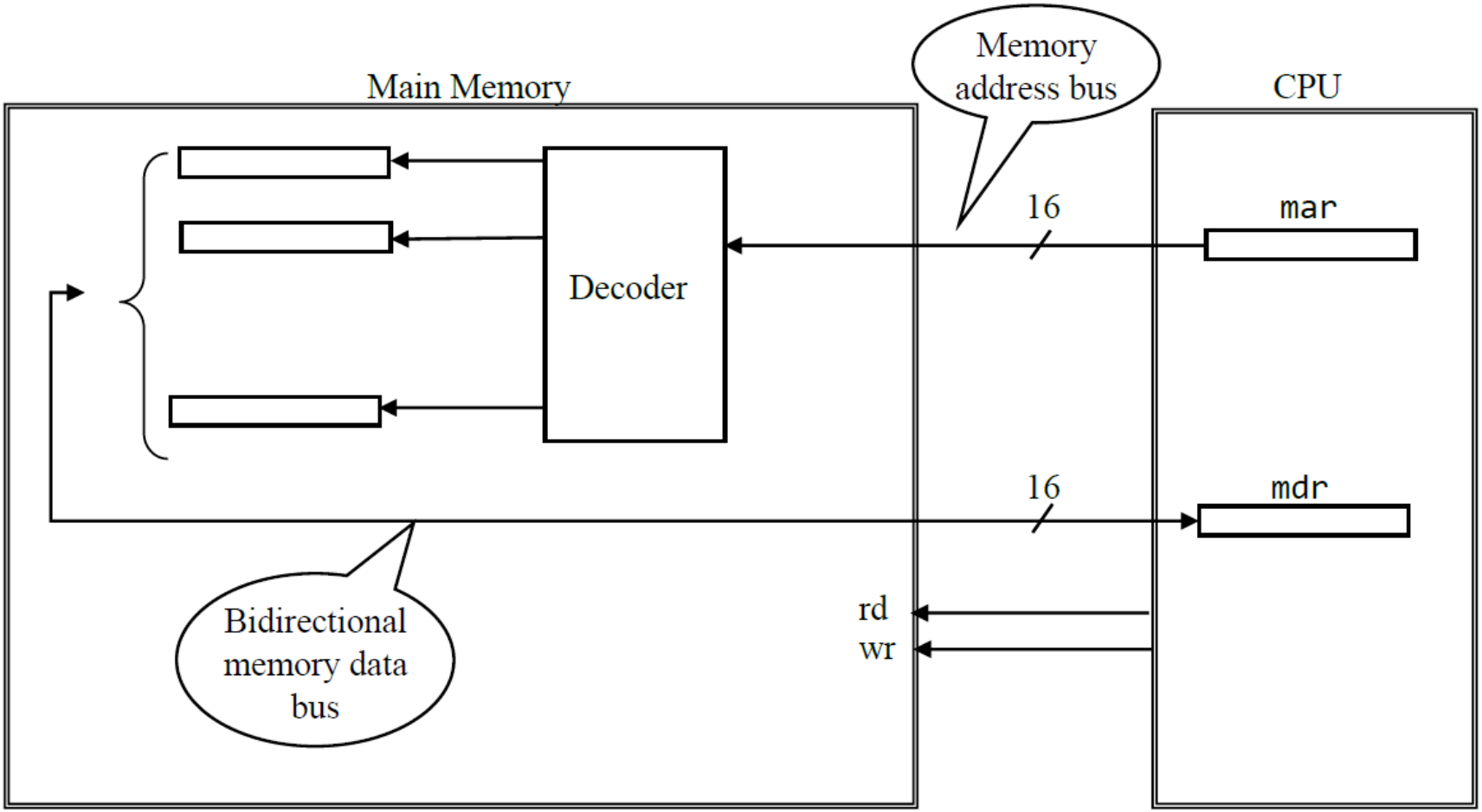


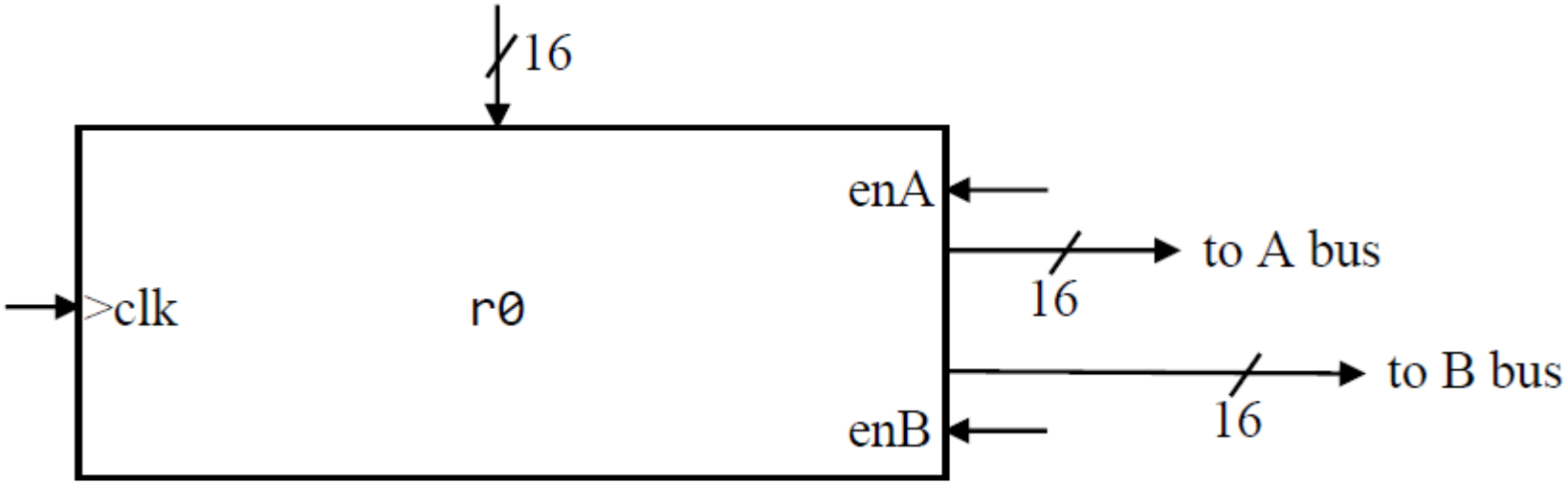
Multiplexer




Left
Right
16
16
NOT
AND
. . .
ROR
16
16
16
F0
F1
F2
F3
0
1
2
15
16-bus
Multiplexer
16
ALU out

Random-Access and Read-Only Memory




16
enA
to A bus
16
>clk
r0
to B bus
16
enB