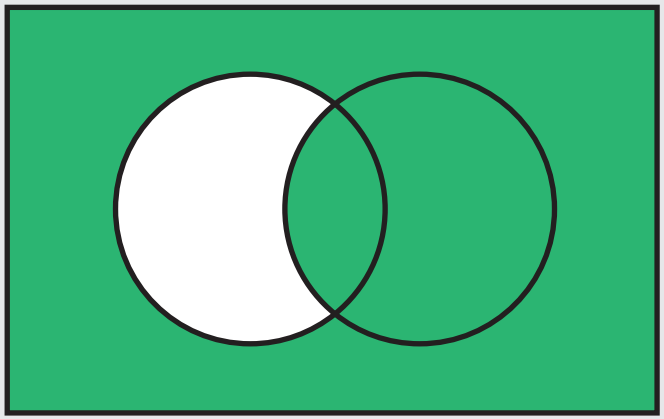
# Computer Architecture

Paul Mellies

Lecture 9 : Boolean Logic & Karnaugh Maps

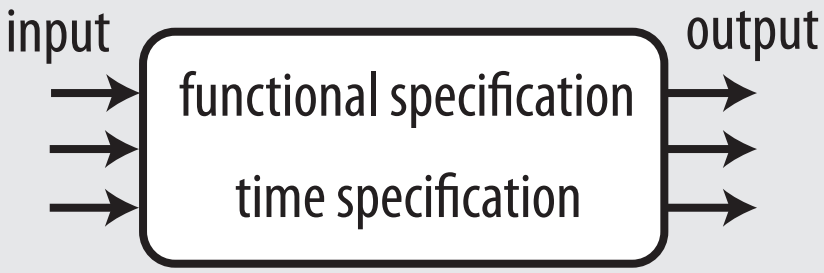


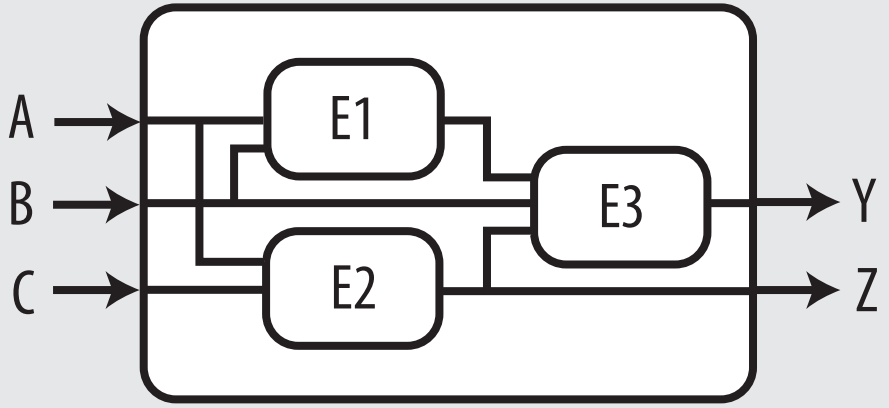
# Boolean Logic

# Circuit ( in digital electronics )

A circuit can be viewed as a black box with

- one or more discrete-valued **input terminals**
- one or more discrete-valued **output terminals**
- a **functional specification** describing the relationship between input and output
- a **timing specification** describing the delay between inputs changing and outputs responding




Peering inside the black box, it is composed of

- nodes — wires classified as input, output and internal
- elements — themselves circuits with inputs, outputs and specifications

# Boolean algebra

Boolean algebra is based on three operations :

- conjunction : $x \wedge y$
- disjunction : $x \vee y$
- negation : $\neg x$

and two constants :

- true : $\mathbf{1}$
- false : $\mathbf{0}$

These operations and constants are required to satisfy a series of equations.

def

# monoid

(algebra)

(& computer science)

a set $M$ equipped with an associative binary operation $M \times M \xrightarrow{\cdot} M$ with a neutral element $e \in M$

$$a, b \in M \longmapsto a \cdot b \in M$$

(1) associativity means that for all $a, b, c \in M$

$$(a \cdot b) \cdot c = a \cdot (b \cdot c)$$

② an element $e$ is <u>neutral</u> precisely when for all elements $a \in M$

$$a \cdot e = a = e \cdot a$$

<u>Examples</u>:

① the set of integers $\mathbb{N}$ equipped with addition

$$a \cdot b = a + b$$

with neutral element $0$

$$0 + a = a = a + 0$$

② the set of integers $\mathbb{N}$ equipped with multiplication

$$a \cdot b = a \times b$$

with neutral element $1$

$$1 \times a = a = a \times 1$$

(3) given a set $A$ of letters ('called an alphabet')

the set $M = A^*$

of finite words on the alphabet $A$

with $\cdot$ defined as concatenation:

$$A = \{a, b\}$$

$$aabbab \cdot abaa$$

$= aabbababaa$

the neutral element
in that case
is the empty word.

# Boolean algebra

Associativity :

$$(x \wedge y) \wedge z = x \wedge (y \wedge z)$$
$$(x \vee y) \vee z = x \vee (y \vee z)$$

Commutativity :

$$x \wedge y = y \wedge x$$
$$x \vee y = y \vee x$$

Distributivity :

and or and or and

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$
$$x \times (y + z) = (x \times y) + (x \times z)$$

times sum

Identities :

$$x \wedge 1 = x$$
$$x \vee 0 = x$$

Annihilation :

$$x \wedge 0 = 0$$

$\wedge \rightleftharpoons$ multiplication

$\vee \rightleftharpoons$ addition

Boole

# Boolean algebra

Idempotence :

$$x \wedge x = x$$
$$x \vee x = x$$

Absorption :

$$x \wedge (x \vee y) = x$$
$$x \vee (x \wedge y) = x$$

Distributivity :

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$

Annihilation :

$$x \vee 1 = 1$$

# Boolean algebra

Complementation :

$$x \wedge \neg x = 0$$
$$x \vee \neg x = 1$$

Double negation :

$$\neg \neg x = x$$

De Morgan duality :

$$\neg x \wedge \neg y = \neg (x \vee y)$$
$$\neg x \vee \neg y = \neg (x \wedge y)$$

$$\neg 1 = 0$$
$$\neg 0 = 1$$

# Boolean algebra

**Definition**
A boolean algebra is defined as a set **A** equipped with the operations and constants $\wedge$ $\vee$ $\neg$ $\mathbf{0}$ $\mathbf{1}$ satisfying the mentioned equations.

**Observation :** every boolean algebra is ordered by the order below :

$$x \leq y \quad \Leftrightarrow \quad x = x \wedge y$$

In this ordered set **A** :

- the constant $\mathbf{0}$ is the least element
- the constant $\mathbf{1}$ is the greatest element

Recall that an order is a transitive, reflexive and antisymmetric relation.

# Boolean algebra

Moreover, the two operations $\wedge$ $\vee$ are monotone in the sense that :

$\forall x, x'; y, y' \in$ **A** x **A** x **A** x **A**

$$x \leq x' \text{ and } y \leq y' \quad \Rightarrow \quad x \wedge y \leq x' \wedge y'$$

$$x \leq x' \text{ and } y \leq y' \quad \Rightarrow \quad x \vee y \leq x' \vee y'$$

Accordingly, the negation $\neg$ is anti-monotone in the sense that :

$\forall x, y \in$ **A** x **A**

$$x \leq y \quad \Rightarrow \quad \neg y \leq \neg x$$

# Example of boolean algebra

The set $\mathbf{A} = \{ \mathbf{true}, \mathbf{false} \}$ defines a boolean algebra with :

$$\mathbf{true} \wedge \mathbf{true} = \mathbf{true}$$
$$\mathbf{true} \wedge \mathbf{false} = \mathbf{false}$$
$$\mathbf{false} \wedge \mathbf{false} = \mathbf{false}$$

$$\mathbf{true} \vee \mathbf{true} = \mathbf{true}$$
$$\mathbf{true} \vee \mathbf{false} = \mathbf{true}$$
$$\mathbf{false} \vee \mathbf{false} = \mathbf{false}$$

$$\neg\ \mathbf{true} = \mathbf{false}$$

$$\neg\ \mathbf{false} = \mathbf{true}$$

$$1 = \mathbf{true}$$

$$0 = \mathbf{false}$$

Observe that in this boolean algebra :

$$\mathbf{false} = \mathbf{false} \wedge \mathbf{true}$$

and thus :

$$\mathbf{false} \leq \mathbf{true}$$

# Example of boolean algebra

Here, we suppose given a set **U**.

The set **A** whose elements are the subsets of **U** is a boolean algebra with conjunction, disjunction and negation defined as follows :

$$X \wedge Y = X \cap Y \qquad X \vee Y = X \cup Y$$

$$\neg X = U \setminus X$$

$$1 = U \qquad 0 = \emptyset$$

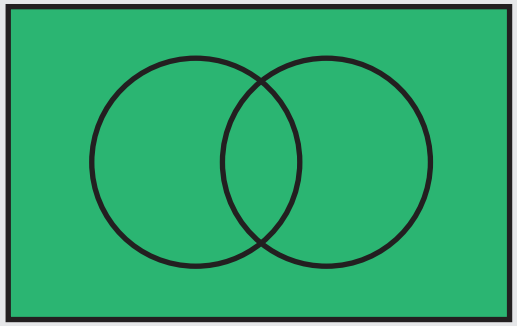
Observe that one recovers the boolean algebra { true , false } when **U** is defined as the singleton set $U = \{*\}$.

$$\textbf{true} = \{*\} \qquad \textbf{false} = \emptyset$$

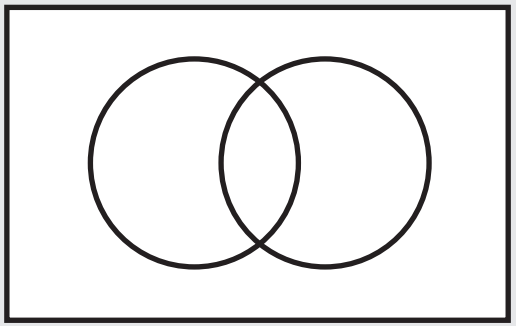
syntax $\wedge \vee \neg 0 1$
boolean expressions

# Venn diagrams

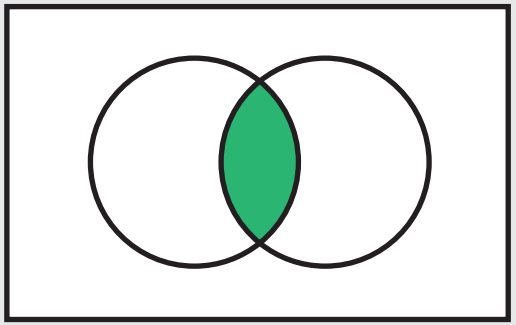
semantics
(= meaning)



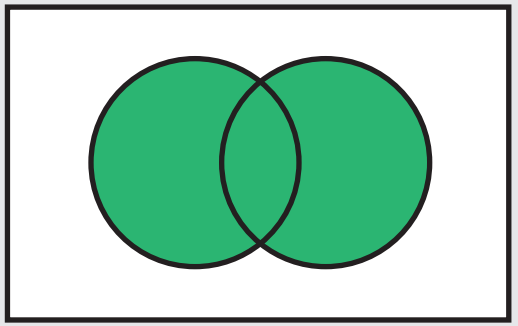
$1 = U$



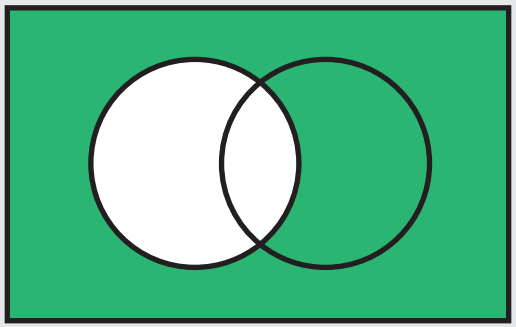
$0 = \emptyset$



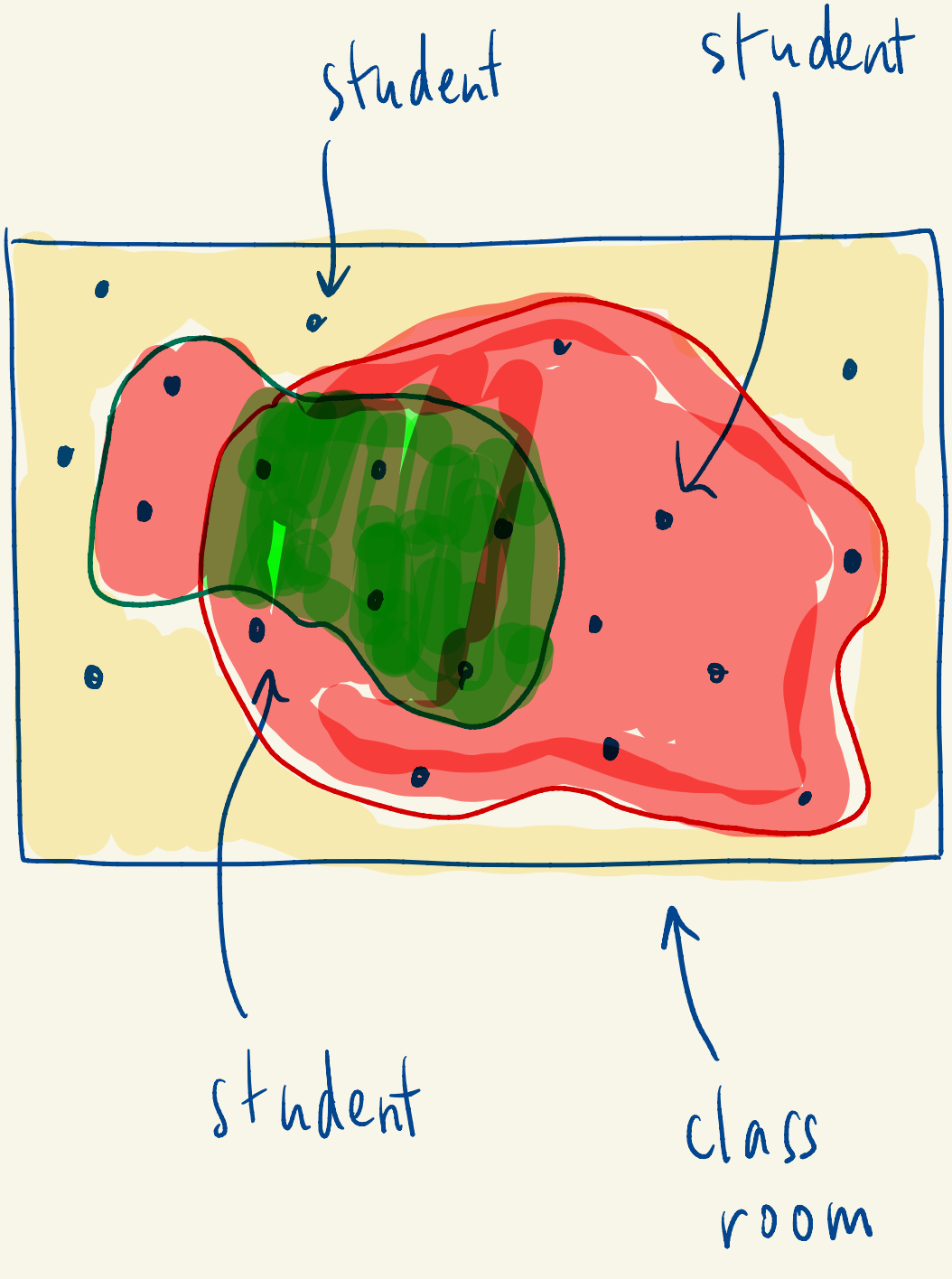
$X \wedge Y = X \cap Y$



$X \vee Y = X \cup Y$



$\neg X = U \setminus X$



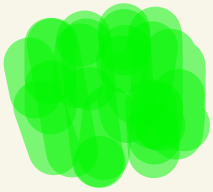

$U$

$X$ = "love ice-cream"

$U$ = the set of students

$Y$ = "love candies"

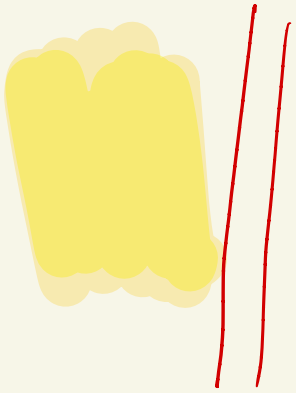
$X \wedge Y$ = "love ice-cream and love candies"



$= X \cap Y$

$X \vee Y$ = "love ice-cream or love candies"

$\neg (X \vee Y)$ = "not (love ice-cream or love candies)"

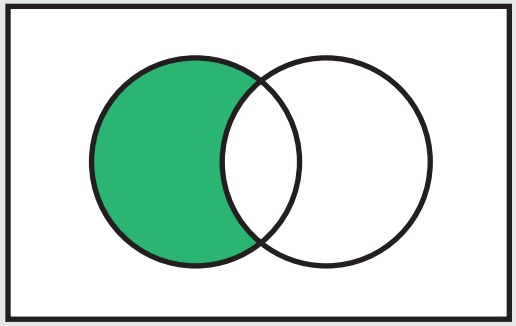


= "not love ice-cream and not love candies"

$\neg X \wedge \neg Y$

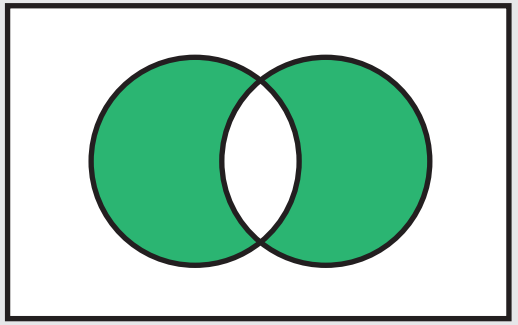
# Venn diagrams
# ( derived constructions )

Difference



$$X \setminus Y = X \wedge \neg Y$$

Symmetric difference
= exclusive or



$$X \triangle Y = (X \vee Y) \setminus (X \wedge Y)$$
$$= (X \vee Y) \wedge \neg (X \wedge Y)$$

$\neg(A \wedge B) = \neg A \vee \neg B$

# Venn diagrams
# ( derived constructions )

the set of students who love icecream but do not love candies

$X \Rightarrow Y =$

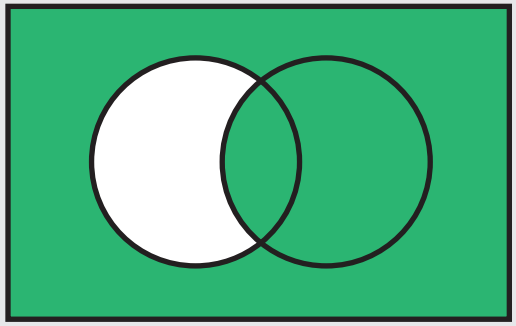
$\neg(X \wedge \neg Y) = \neg X \vee \neg\neg Y$

$= \neg X \vee Y$

$X \wedge \neg Y = \emptyset$

is empty

Implication



$X$ = love icecream

$Y$ = love candies

$$X \to Y = \neg(X \wedge \neg Y)$$

$$= \neg X \vee Y$$

$X \Longrightarrow Y$ = whenever a student in the classroom loves icecream, she/he loves candies

Key property of implication :

$$X \leq Y \to Z \quad \Leftrightarrow \quad X \wedge Y \leq Z$$

$$X \Rightarrow Y \;=\; \neg(X \wedge \neg Y)$$

$$= \neg X \vee \neg\neg Y$$

$$= \neg X \vee Y$$

the fact that $X \Rightarrow Y$ is equivalent to the fact that every student satisfies $\neg X$ or $Y$

$X \Rightarrow Y$: every student who loves icecream loves candies

$\neg X$: does not love ice cream

$Y$: love candies

# Product-of-sums form

Every boolean expression may be transformed modulo the equations of boolean algebras into a conjunction of disjunctions :

$$\begin{aligned} \textbf{prodofsum} &= 1 \mid \textbf{sum} \mid \textbf{prodofsum} \wedge \textbf{prodofsum} \\ \textbf{sum} &= 0 \mid \textbf{literal} \mid \textbf{sum} \vee \textbf{sum} \\ \textbf{literal} &= \textbf{atom} \mid \neg\, \textbf{atom} \end{aligned}$$

Same grammar formulated this time with the arithmetic notation :

$$\begin{aligned} \textbf{prodofsum} &= 1 \mid \textbf{sum} \mid \textbf{prodofsum} \times \textbf{prodofsum} \\ \textbf{sum} &= 0 \mid \textbf{literal} \mid \textbf{sum} + \textbf{sum} \\ \textbf{literal} &= \textbf{atom} \mid \overline{\textbf{atom}} \end{aligned}$$

# Sum-of-products form

Every boolean expression may be transformed modulo the equations of boolean algebras into a disjunction of conjunctions :

$$\begin{aligned} \textbf{sumofprod} &= 0 \mid \textbf{prod} \mid \textbf{sumofprod} \vee \textbf{sumofprod} \\ \textbf{prod} &= 1 \mid \textbf{literal} \mid \textbf{prod} \wedge \textbf{prod} \\ \textbf{literal} &= \textbf{atom} \mid \neg\, \textbf{atom} \end{aligned}$$

BNF = Backus-Naur form

Same grammar formulated this time with the arithmetic notation :

$$\begin{aligned} \textbf{sumofprod} &= 0 \mid \textbf{prod} \mid \textbf{sumofprod} + \textbf{sumofprod} \\ \textbf{prod} &= 1 \mid \textbf{literal} \mid \textbf{prod}\ \textbf{x}\ \textbf{prod} \\ \textbf{literal} &= \textbf{atom} \mid \overline{\textbf{atom}} \end{aligned}$$

- an atom = another name for a variable

  $X, Y, Z$

- a literal is either an atom $X$ or the negation of an atom

  $\neg X$

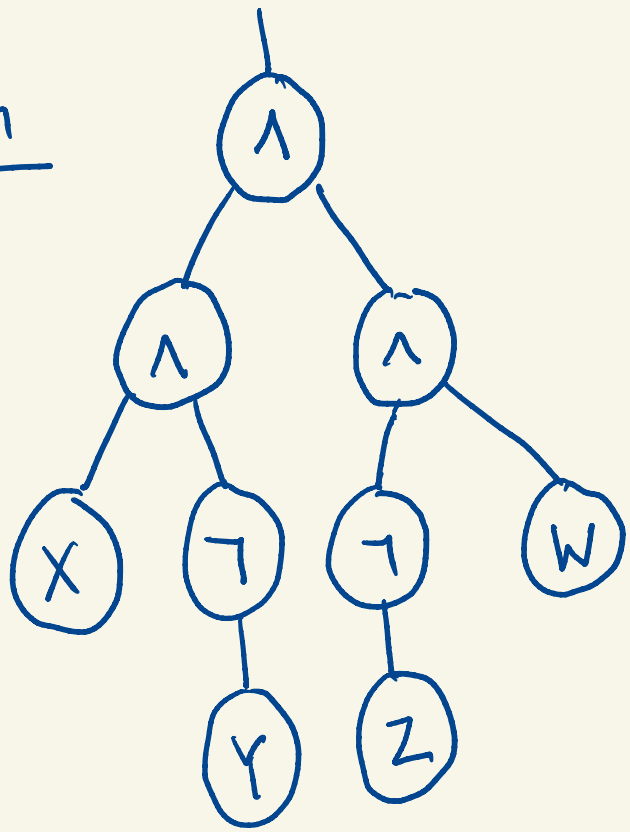
$\neg X =$ tree

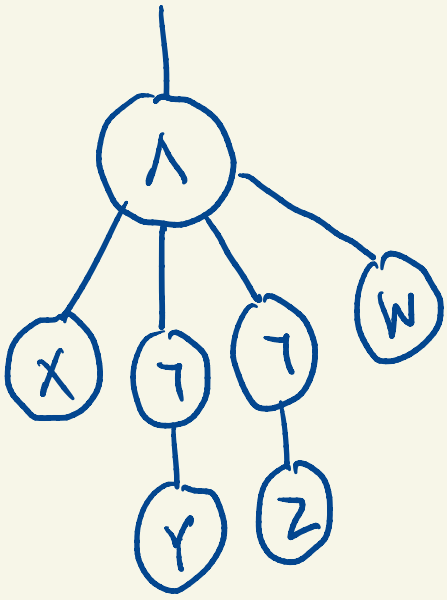


- a product is either $1$
  or a variable $X$
  or the negation of variable $X$
  or a product of variables $X_1, \dots, X_p$ and negations of variables $\neg Y_1, \dots, \neg Y_q$

a boolean expression in product form

$X \wedge \neg Y \wedge \neg Z \wedge W =$



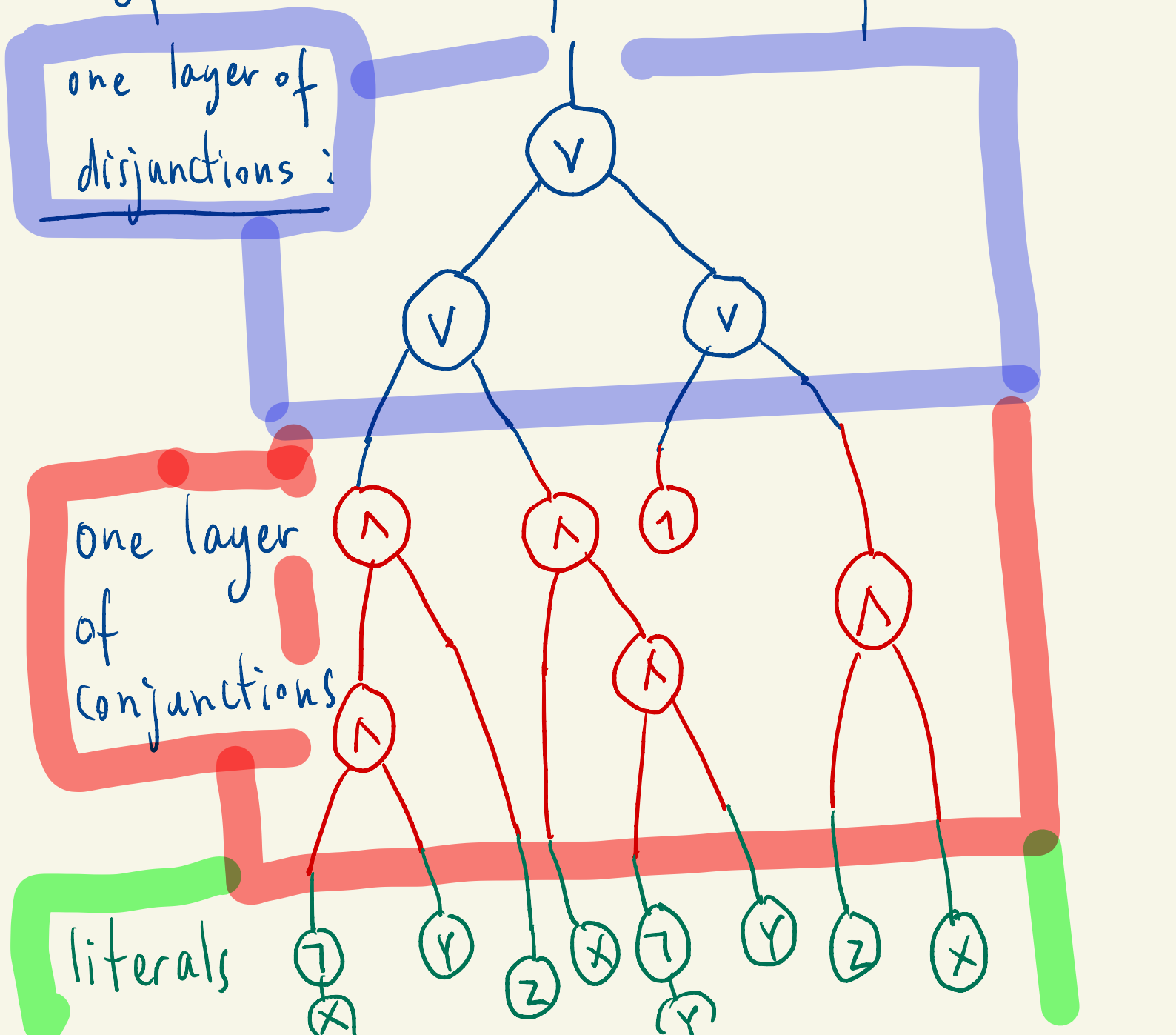



conjunction

a boolean expression
in a <u>sum-of-product form</u>

is defined by a series of disjunctions

of boolean expression in product form

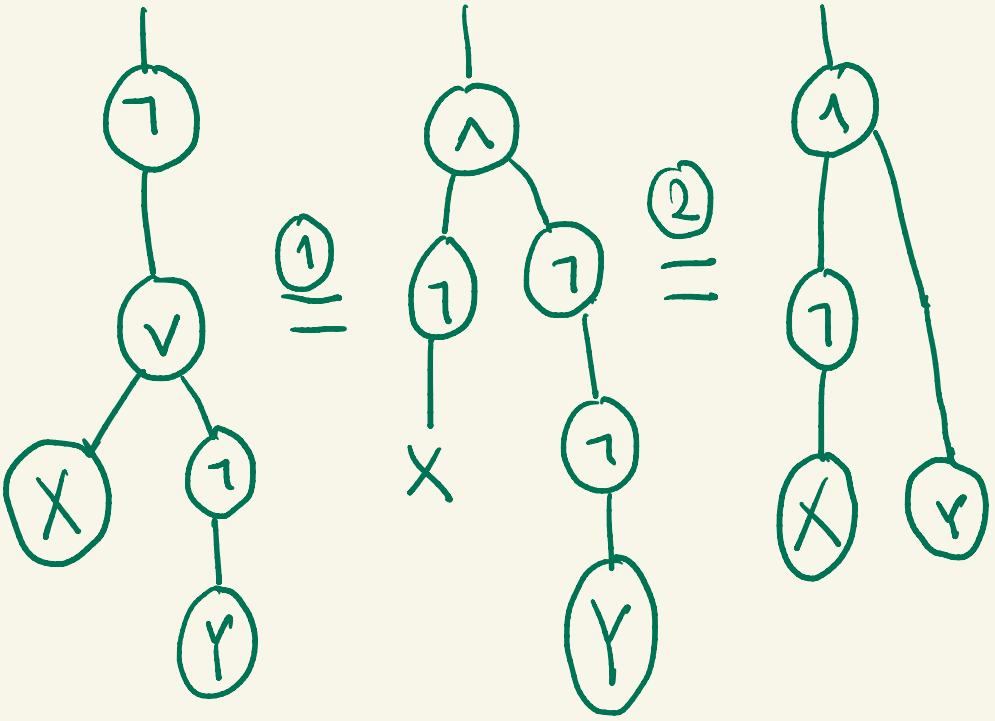


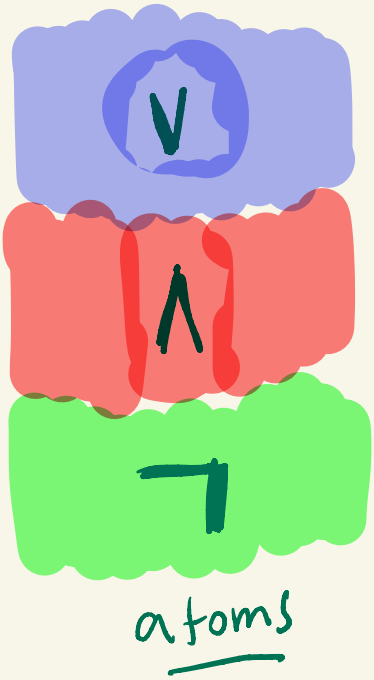
Question:

consider the boolean expression

$$\neg(X \vee \neg Y)$$

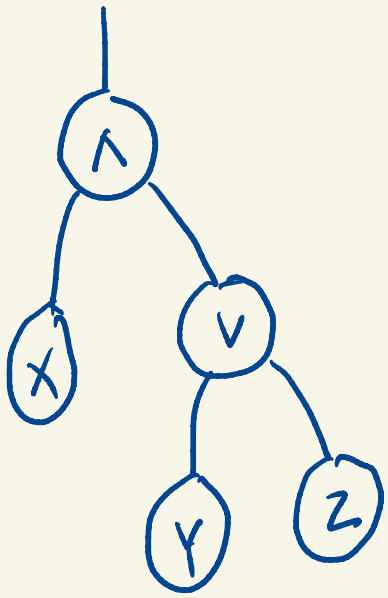
what is its sum of prod form?



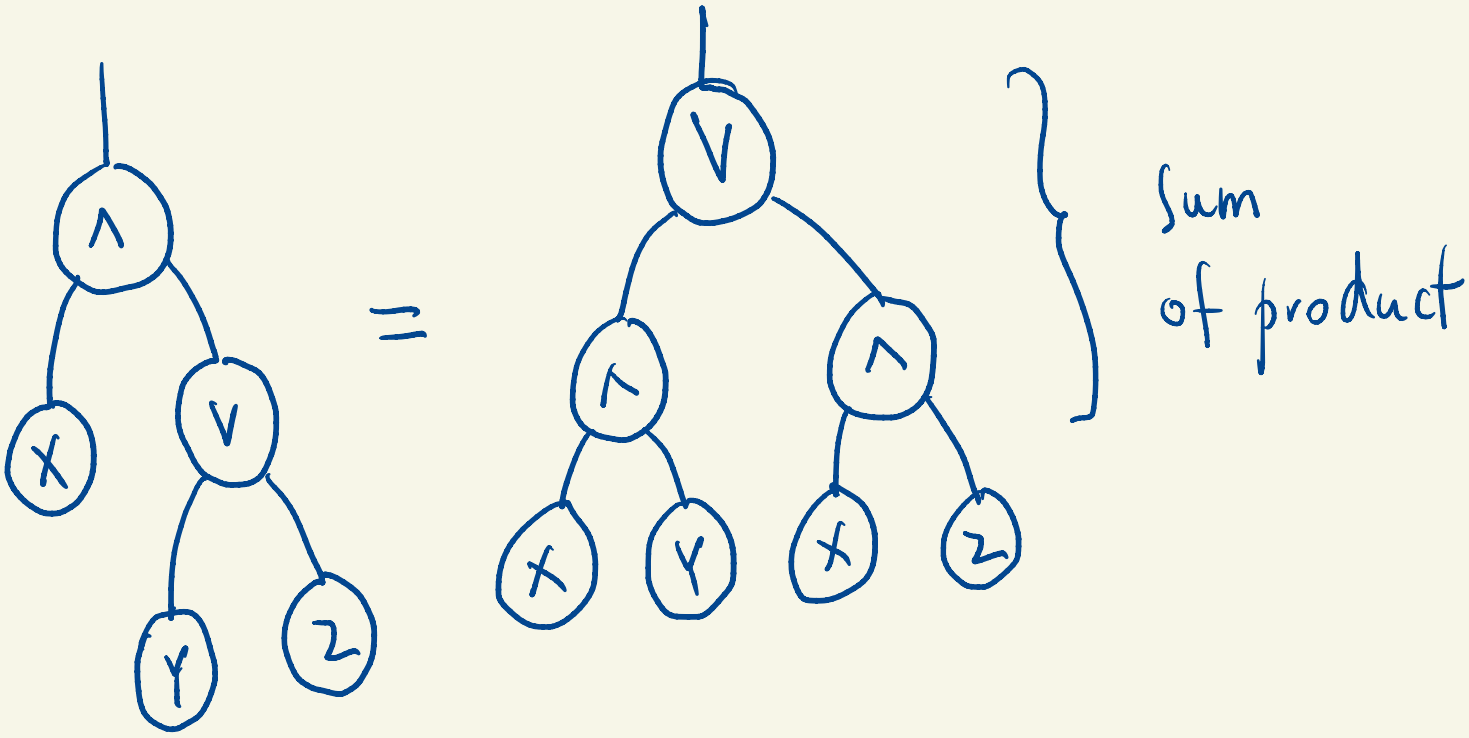




① and ② are equations of boolean algebra

$$X \wedge (Y \vee Z)$$




not in sum-of-product form!

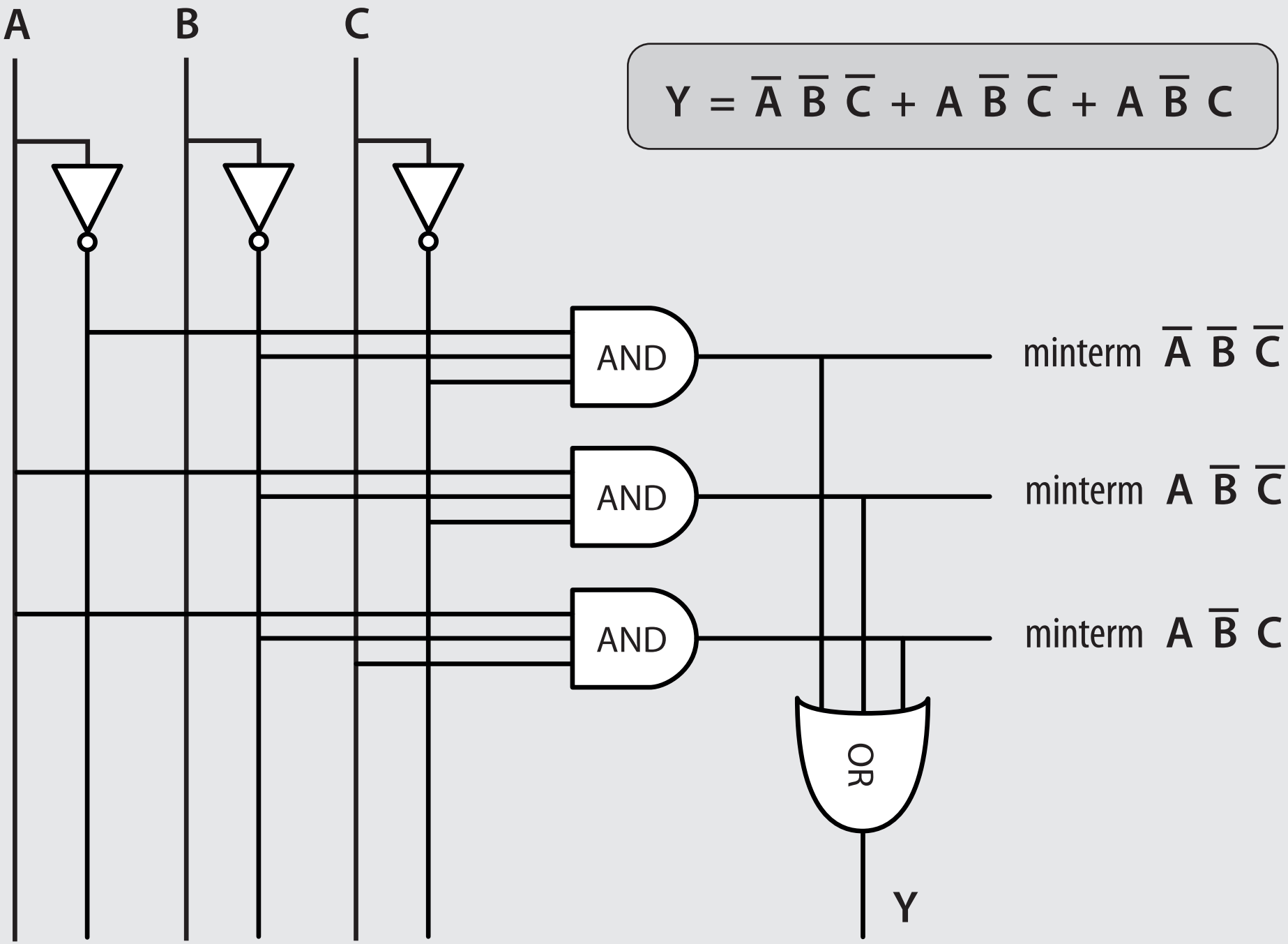



$$X \wedge (Y \vee Z) = (X \wedge Y) \vee (X \wedge Z)$$

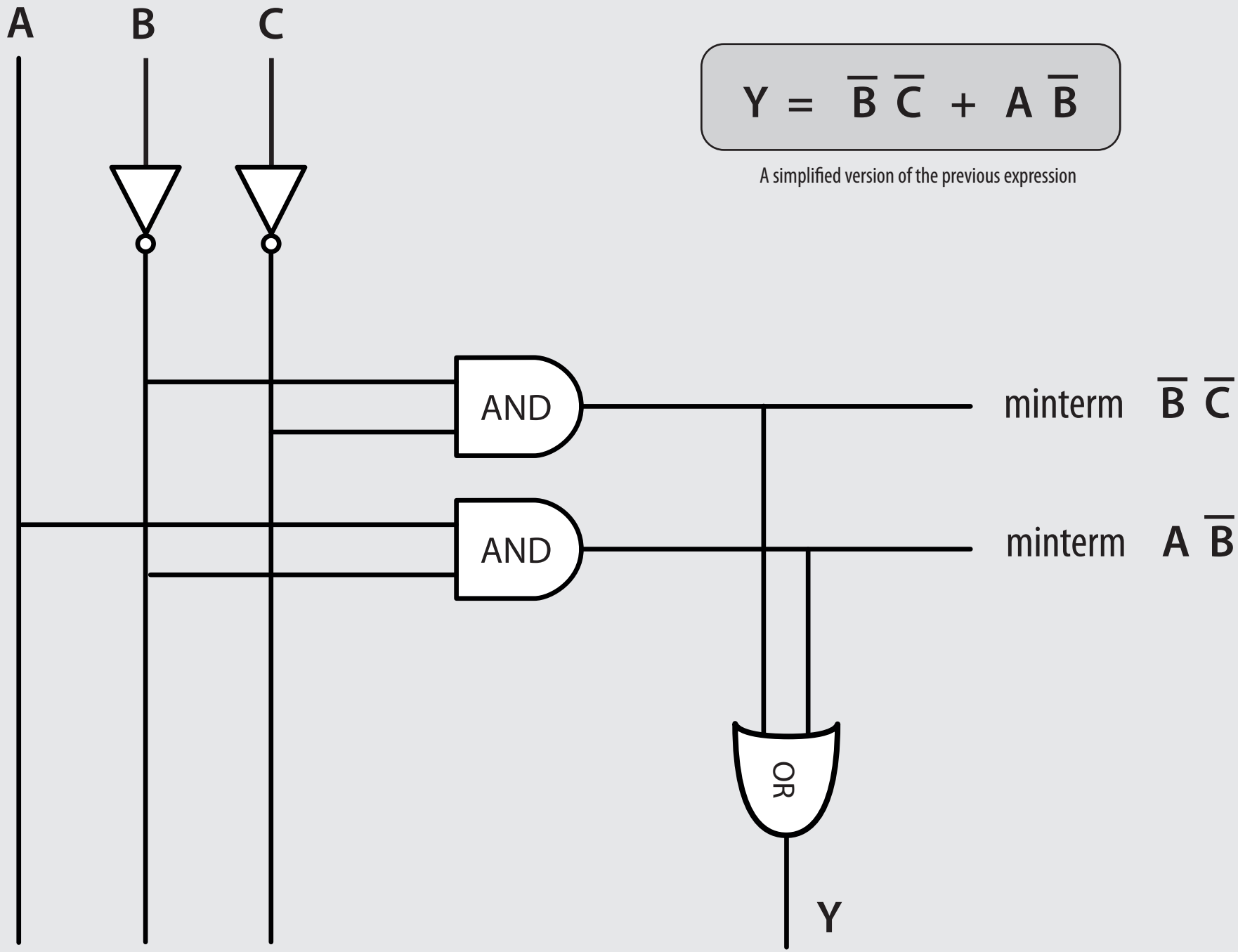
compare with

$$X \times (Y + Z) = XY + XZ$$ in algebra.

# Sum of products in schematics



# Sum of products in schematics




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

A simplified version of the previous expression

# Exercise

Show that the equality

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

follows from the equations of boolean algebra.

$$\overline{A}\,\overline{B}\,\overline{C} + A\,\overline{B}\,\overline{C} \overset{①}{=} (\overline{A} + A)\,\overline{B}\,\overline{C} = 1\,\overline{B}\,\overline{C} \overset{②}{=} \overline{B}\,\overline{C}$$

$$(X+Y)Z \overset{①}{=} XZ + YZ$$

$$1 \wedge X \overset{②}{=} X$$

$$\overline{A} + A \quad (\text{not } A) \text{ or } A = \text{true}$$

$$\| \\ 1$$

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

# Solution

Show that the equality

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

follows from the equations of boolean algebra.

$$\begin{aligned}
\overline{A}\,\overline{B}\,\overline{C} + A\,\overline{B}\,\overline{C} &= (A + \overline{A})\,\overline{B}\,\overline{C} && \text{distributivity}\\
&= 1\,\overline{B}\,\overline{C} && \text{complementation}\\
&= \overline{B}\,\overline{C} && \text{identity}
\end{aligned}$$

$$\begin{aligned}
A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C &= A\,\overline{B}\,(C + \overline{C}) && \text{distributivity}\\
&= A\,\overline{B}\,1 && \text{complementation}\\
&= A\,\overline{B} && \text{identity}
\end{aligned}$$

# Solution

idempotence $X = X \vee X$

Show that the equality

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

follows from the equations of boolean algebra.

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

by idempotence and associativity

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

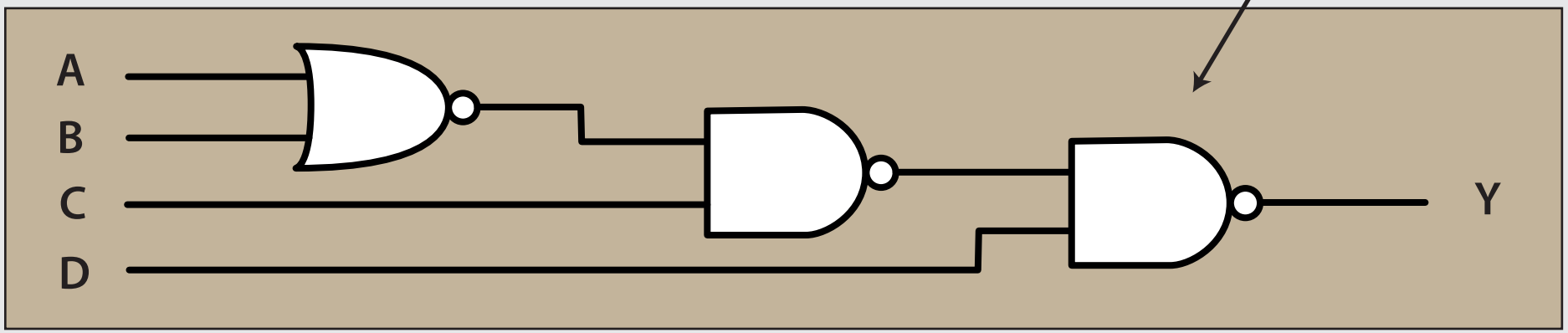
by the two equations of the previous page

This establishes the expected equation.

# Bubble Pushing

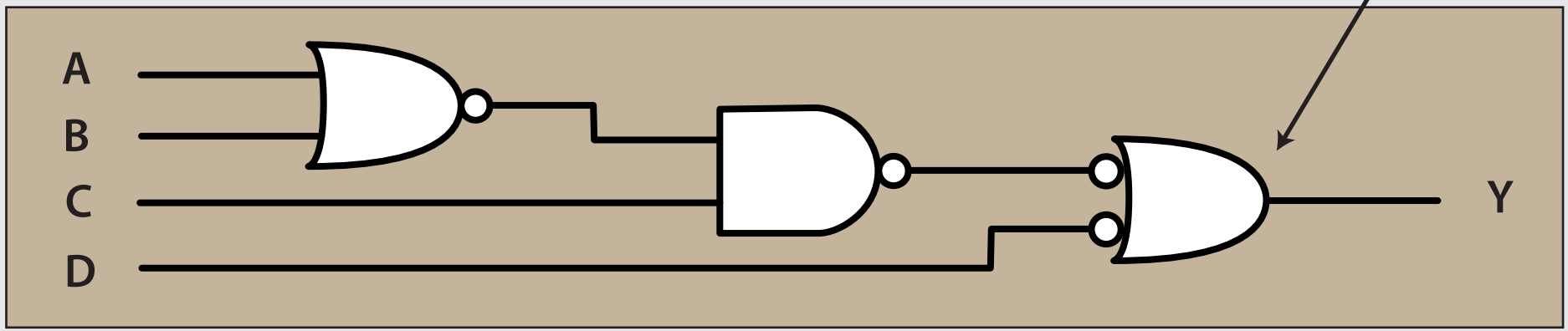
The digital circuit

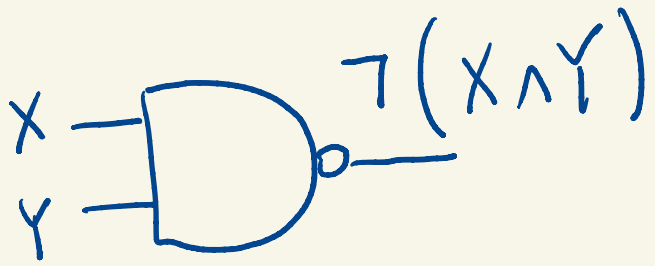
Suppose that one wants
to remove the NAND gate




is functionally equal to

No bubble
on the outpout






$$\neg(X \wedge Y) \equiv \neg X \vee \neg Y$$

de Morgan equation

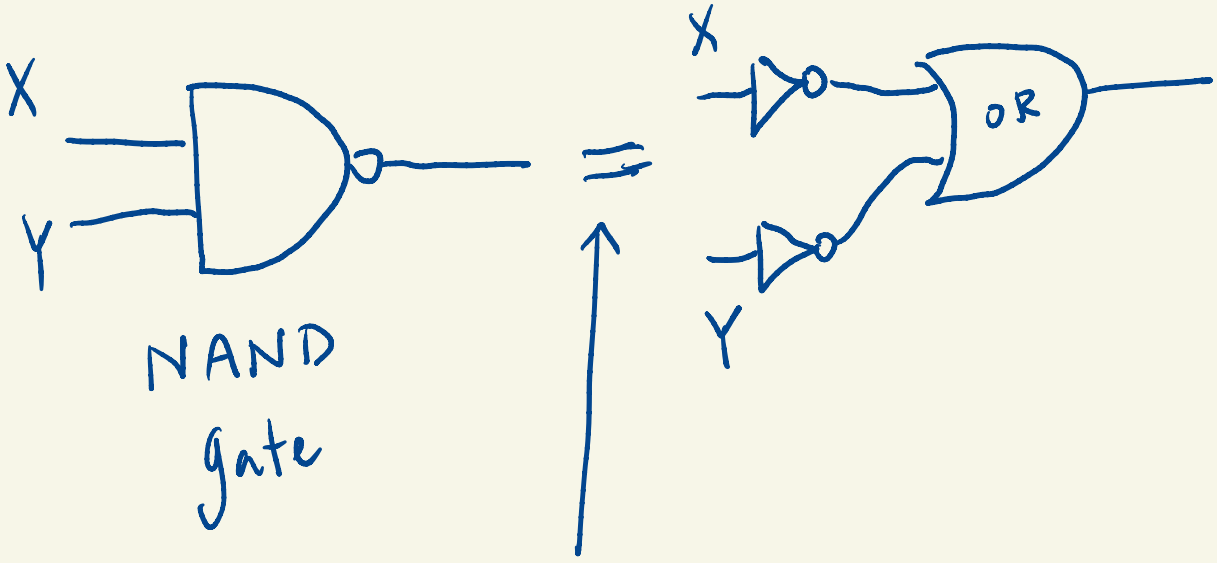
not (love ice cream and love candies)

≡

not love icecream

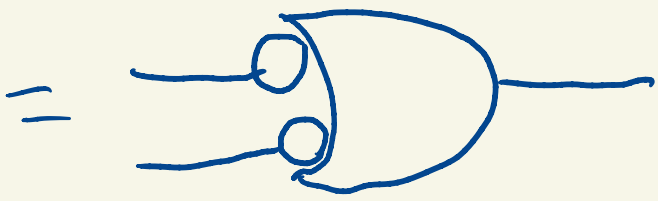
or

not love candies




NAND gate

electronic version of de Morgan equation

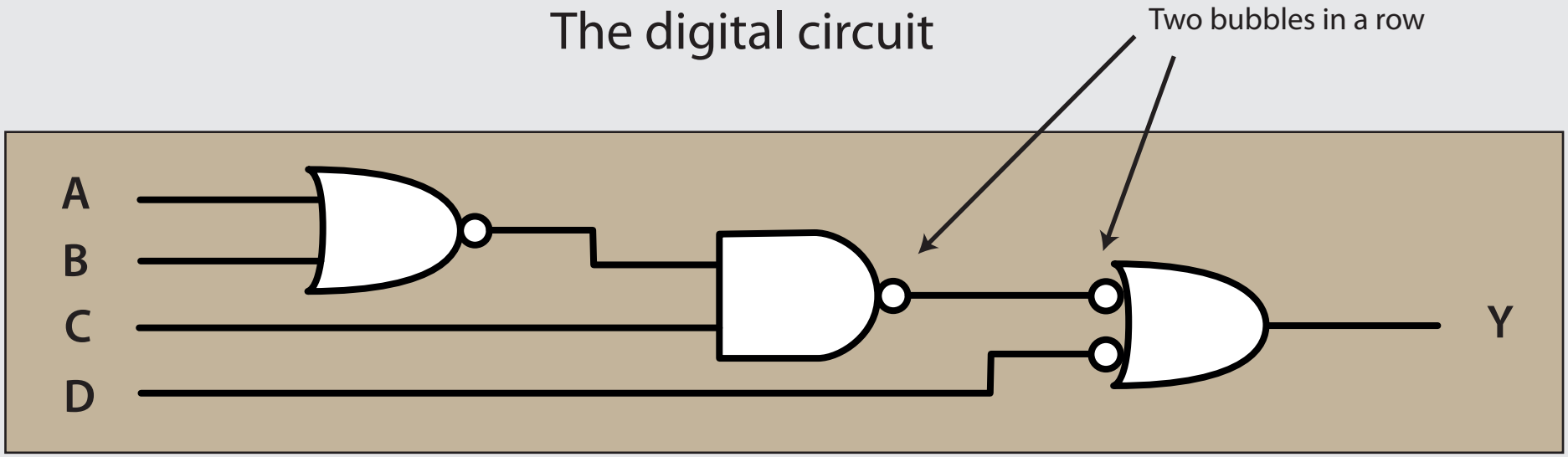


BUBBLE NOTATION

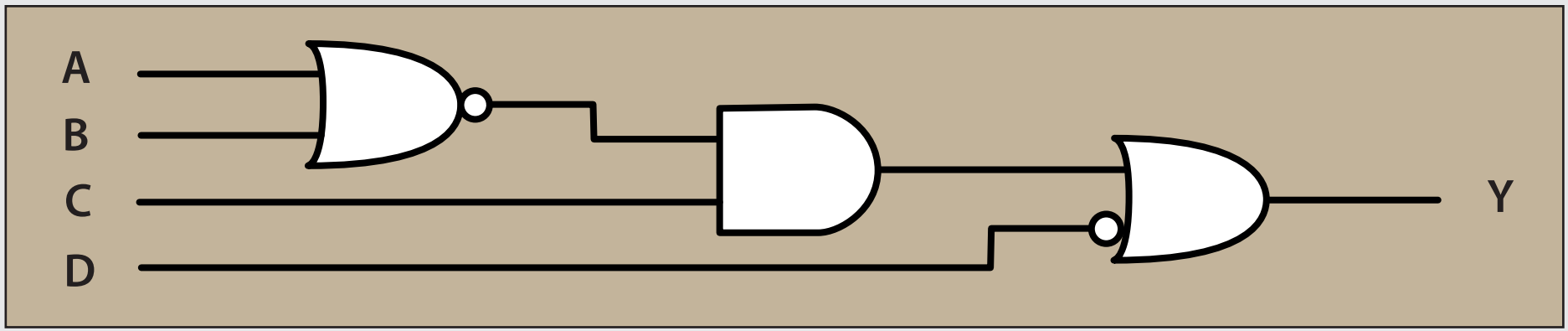
# Bubble Pushing

$\neg\neg X = X$

The digital circuit

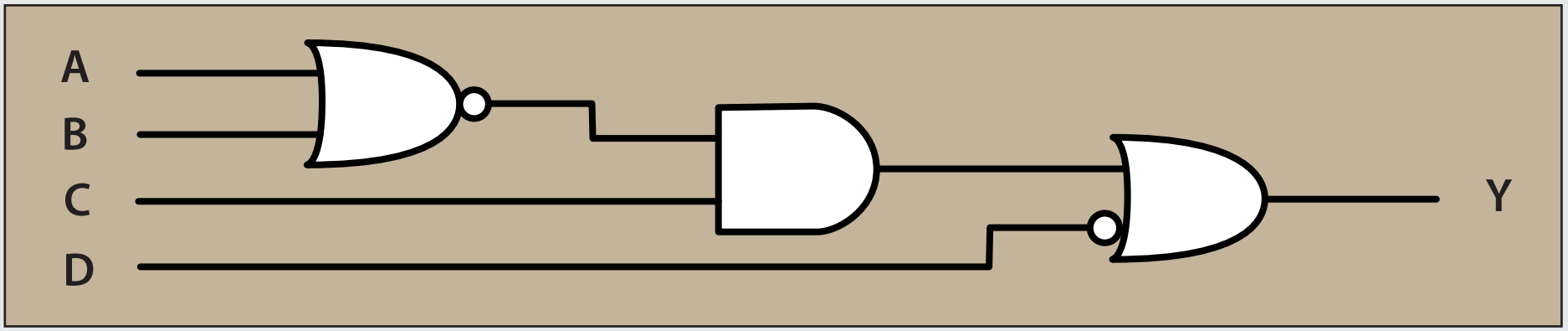



is functionally equal to

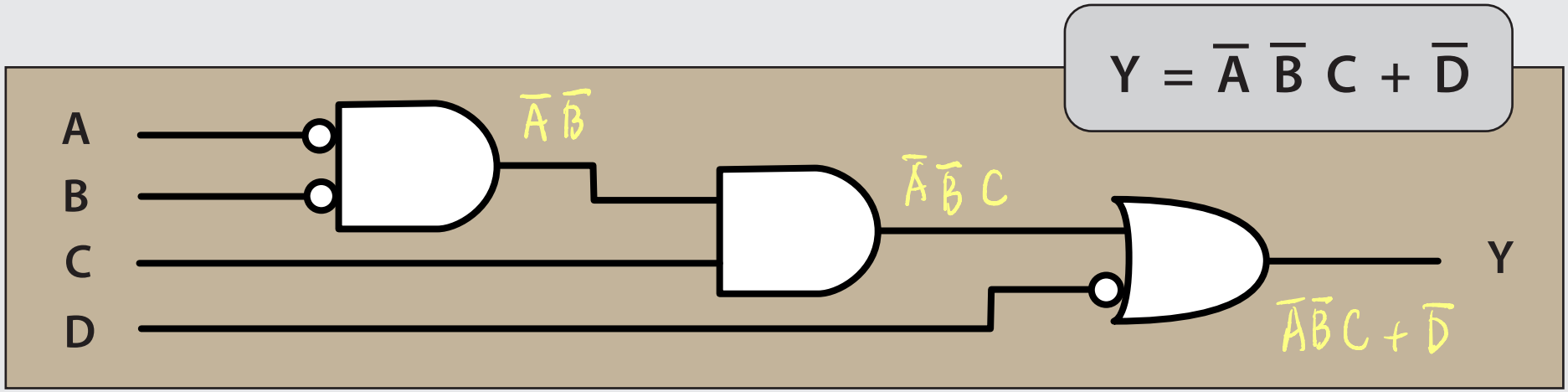


# Bubble Pushing

The digital circuit

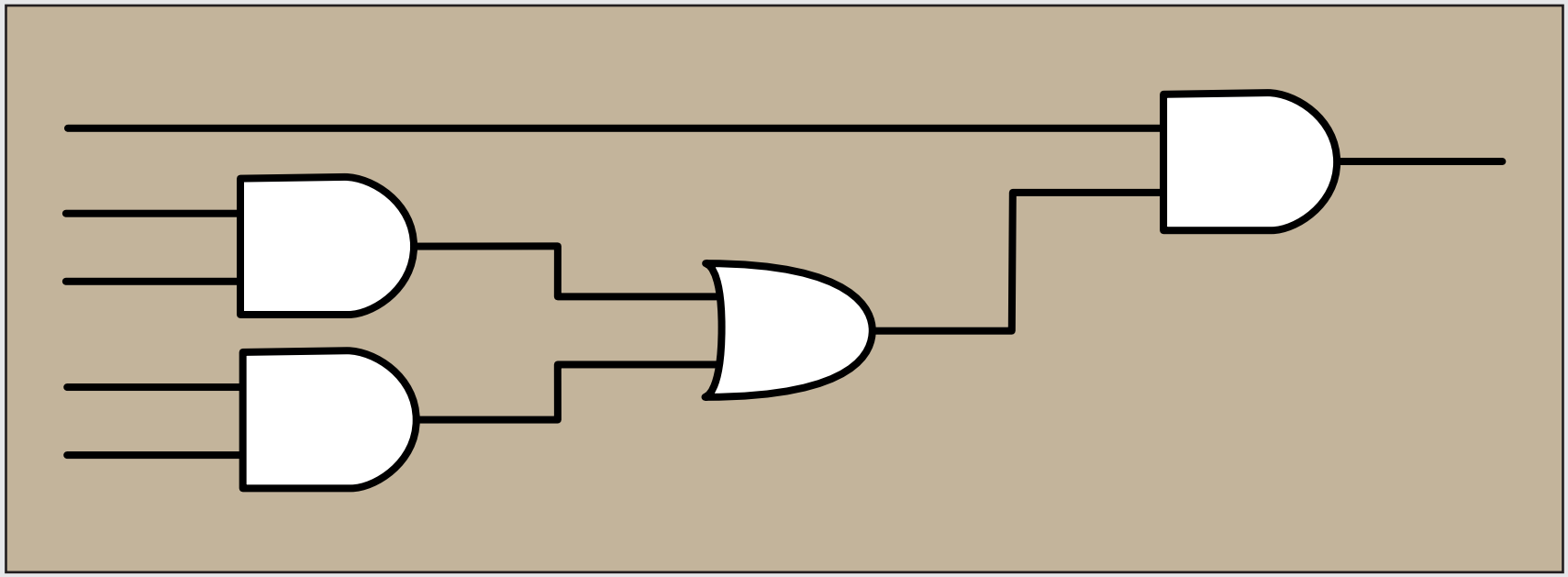



is functionally equal to



# Bubble Pushing for CMOS logic

In some situations, one would like to transform
a logical circuit expressed with AND and OR gates
into an equivalent logical circuit constructed
with NAND gates, NOR gates and inverters.



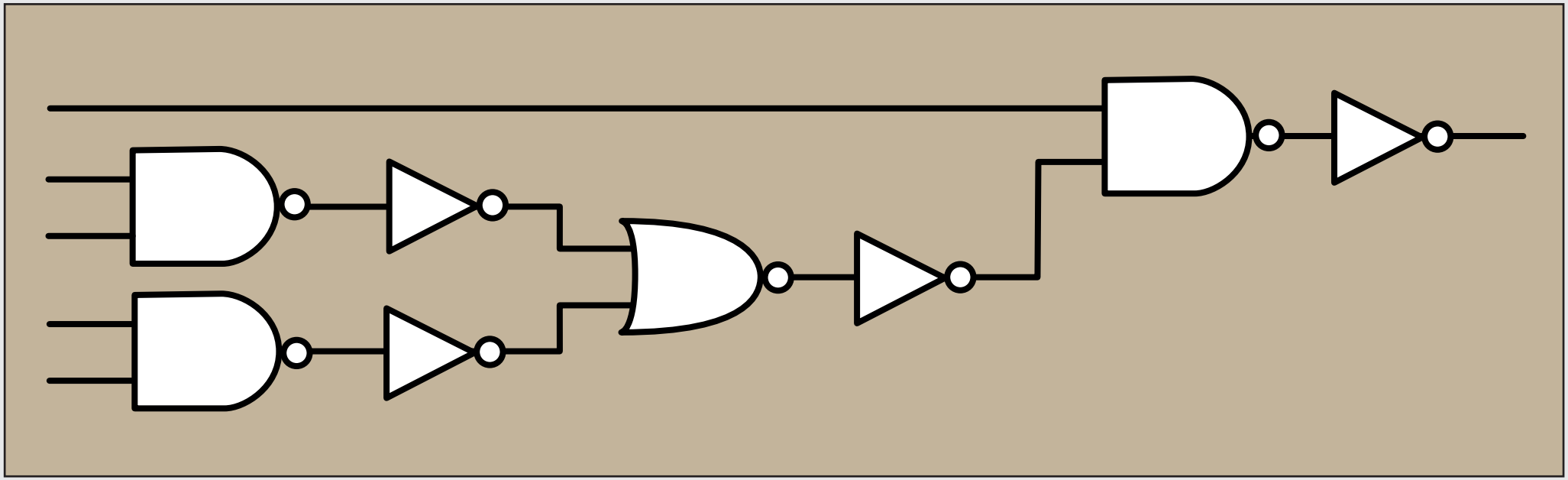
One typical reason is that NAND gates, NOR gates
and inverters are easier to construct in CMOS technology.

# Bubble Pushing for CMOS logic

One brutal way to achieve the translation is to replace

- every AND gate by a NAND gate followed by an inverter
- every OR gate by a NOR gate followed by an inverter

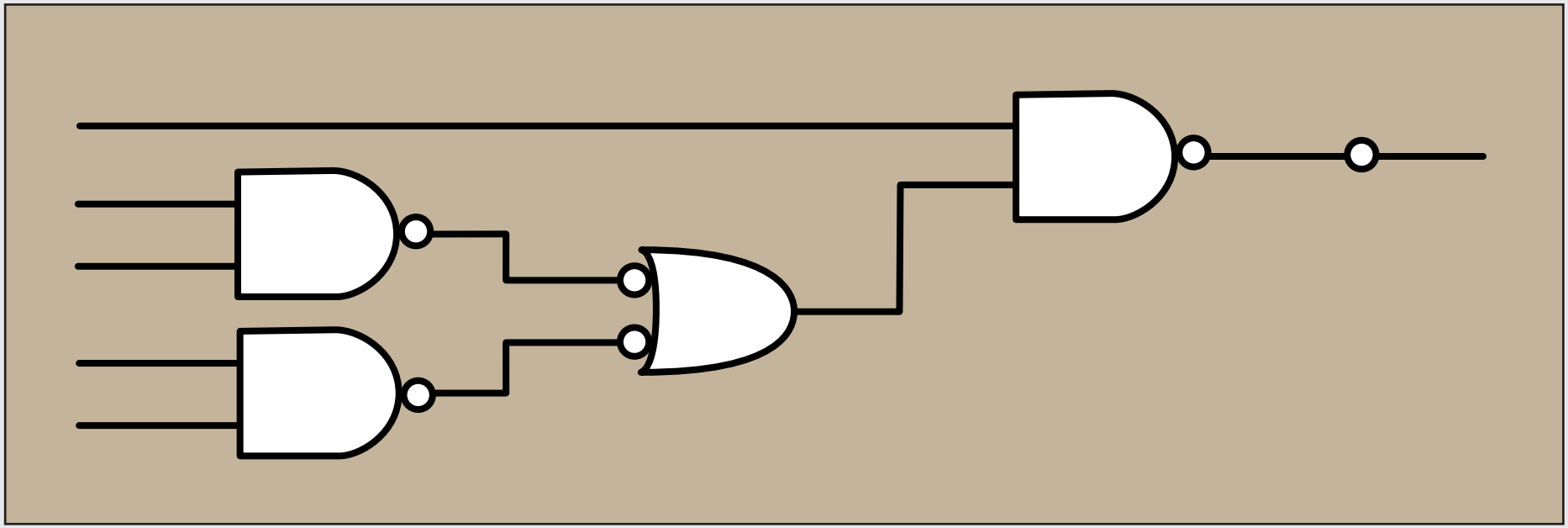
in the way done below :



This procedure works but is far from optimal in general in the number of logical gates in the final CMOS circuit.

# Bubble Pushing for CMOS logic

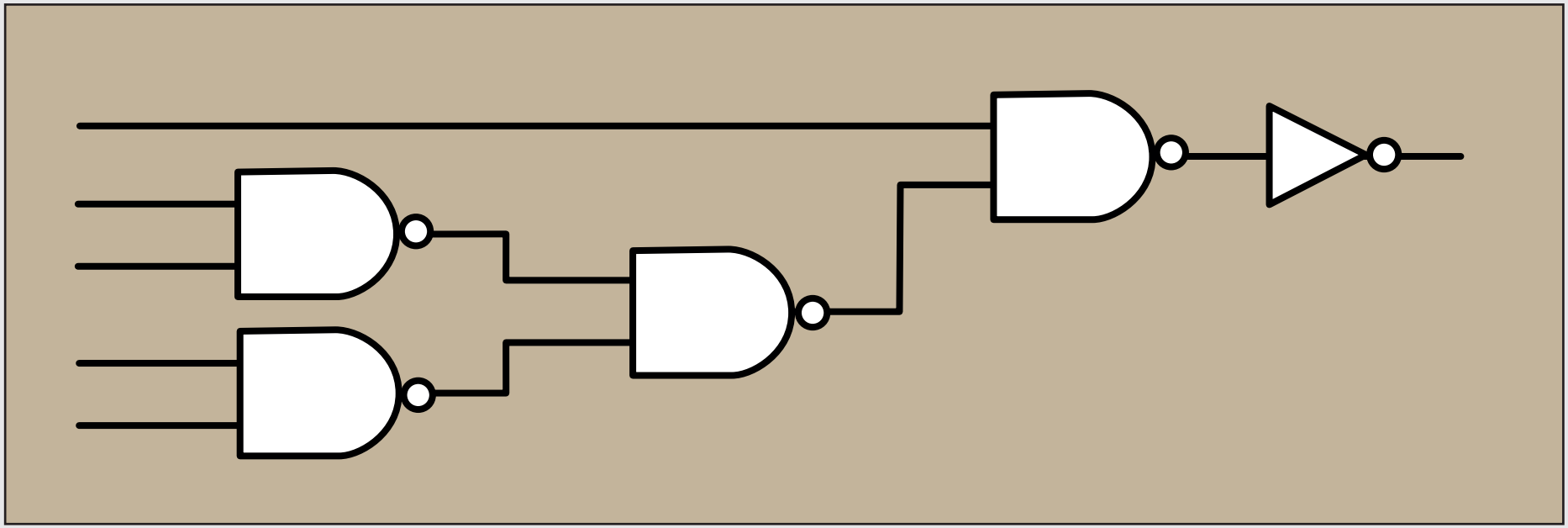
A more sophisticated way to achieve the translation is
to add two negations on some of the wires
of the original logical circuit :



Note that each negation is represented here by a bubble.

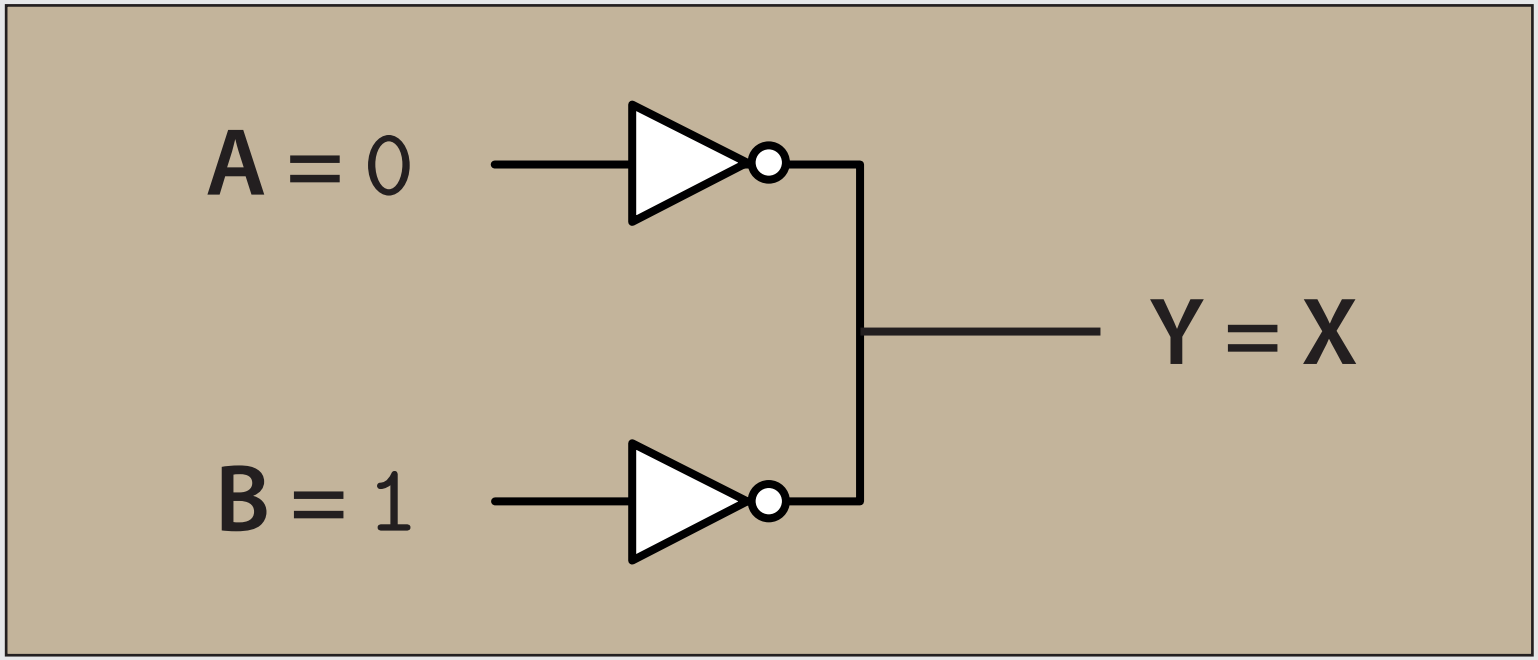
# Bubble Pushing for CMOS logic

A more sophisticated way to achieve the translation is
to add two negations on some of the wires
of the original logical circuit :



One obtains in this way a CMOS circuit with five logical gates
( instead of six gates as in the previous translation )

# The illegal value X

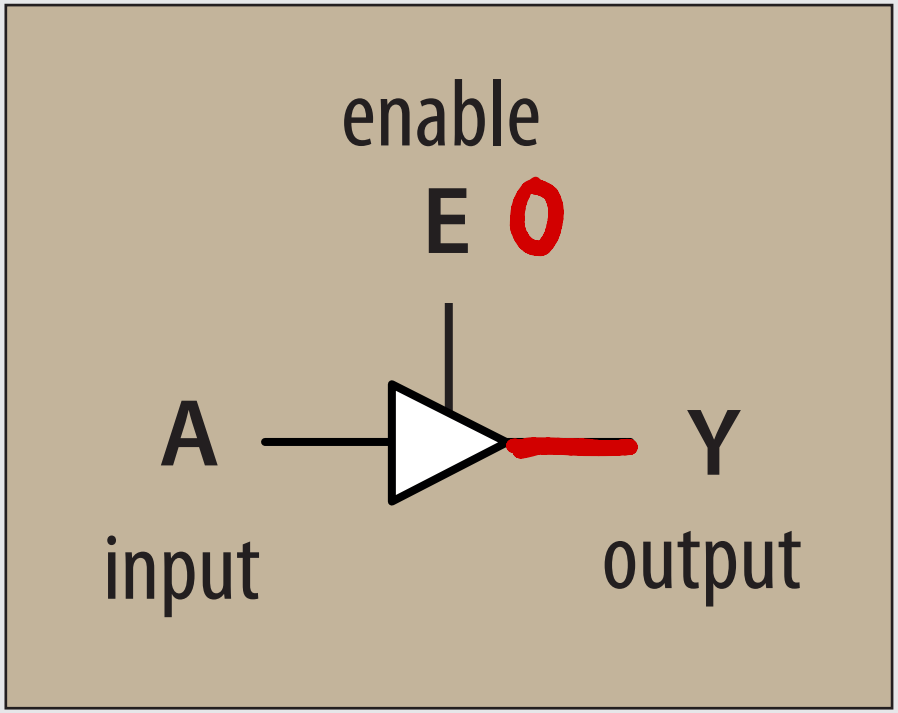



Here, the output **Y** has an unknown or illegal value.

The situation is called a « contention » and is considered as an error which should be avoided.

The reason is that contention may cause large amounts of power to be dissipated between the logical gates, resulting in the circuit getting hot and possibly damaged.

# The floating value Z




| E | A | Y |
|---|---|---|
| 0 | 0 | Z |
| 0 | 1 | Z |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

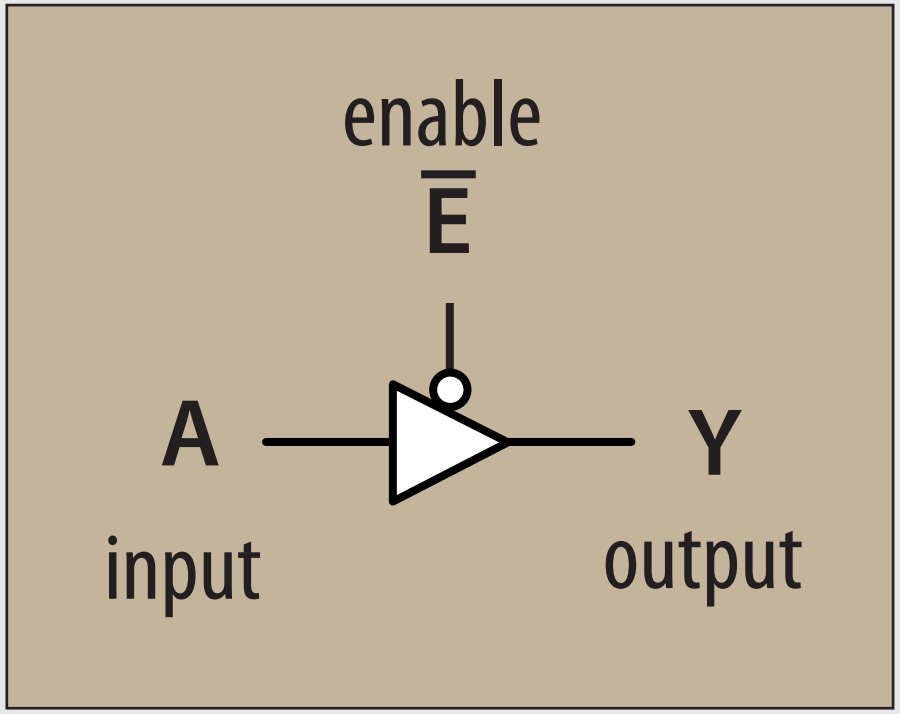
active high enable

The tristate buffer has three possible output values :

- HIGH ( 1 )
- LOW ( 0 )
- FLOATING ( Z )

The output has a floating value means that the wire **Y** may receive any voltage depending on the context.

# The floating value Z




| $\overline{E}$ | A | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | Z |
| 1 | 1 | Z |

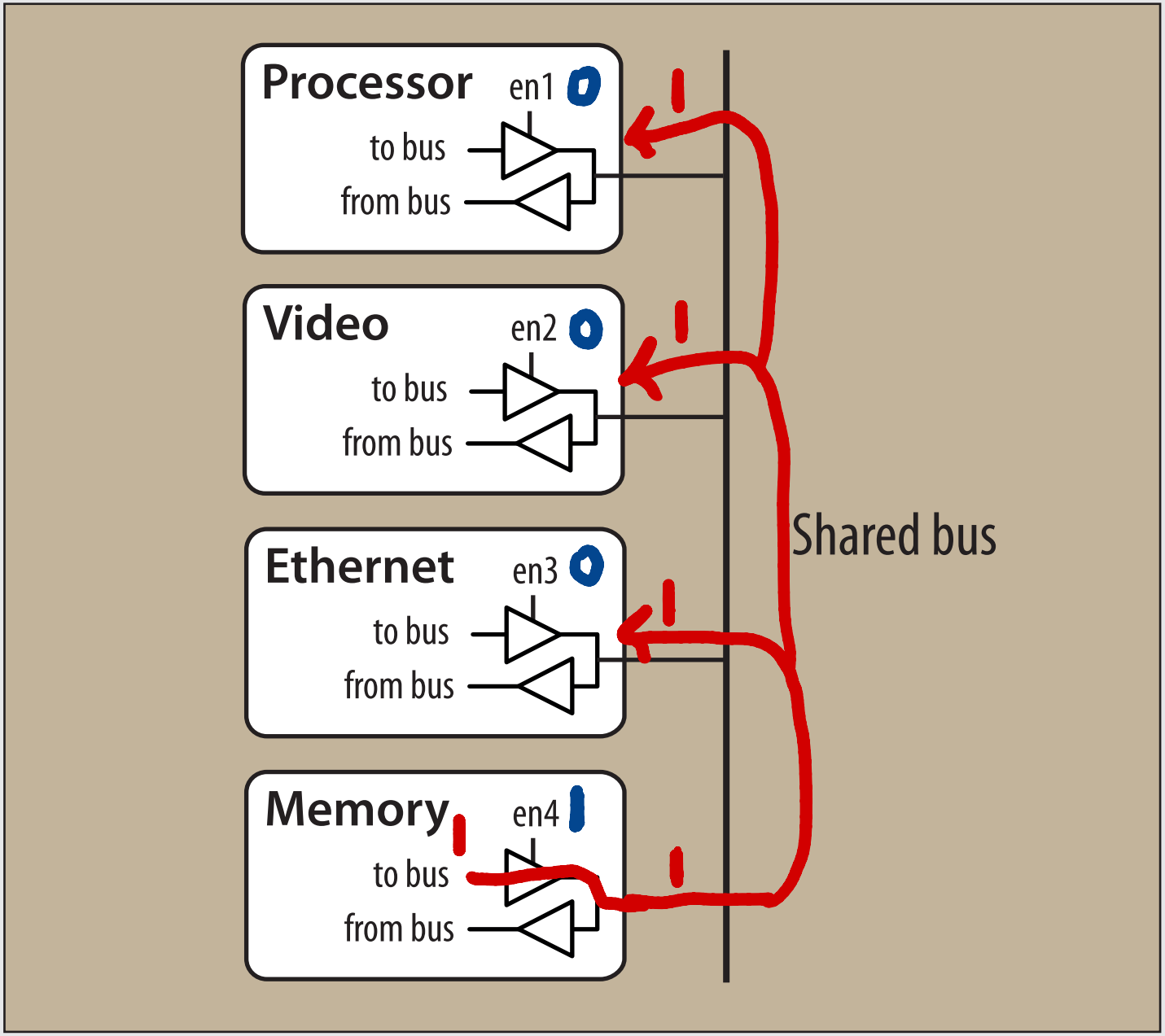
active low enable

The tristate buffer has three possible output values :

- HIGH ( 1 )
- LOW ( 0 )
- FLOATING ( Z )

The output has a floating value means that the wire **Y** may receive any voltage depending on the context.

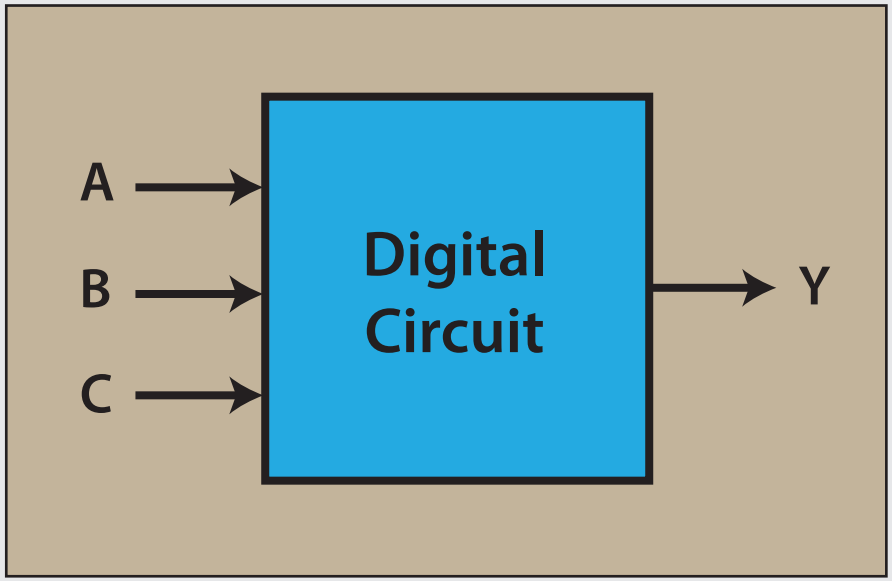
# Tristate bus connecting multiple chips




Today, higher speeds are achieved by point-to-point links between chips

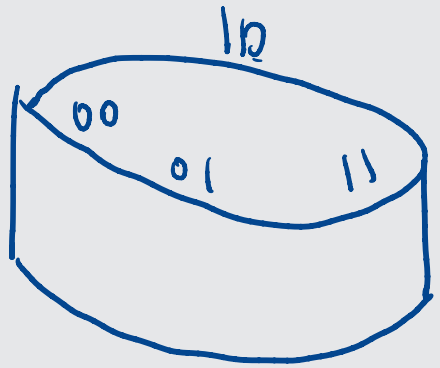
# Karnaugh Maps

# A typical truth value table




| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

# From the truth table to the K-map




| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

| Y / C \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |

Note the clever use of Gray codes here

# Gray codes

Claim: it is possible to enumerate all words of 0's and 1's of length $n$ in such a way that:

(1) all the words are enumerated exactly once

(2) when one moves from one word to the next one changes exactly one bit (including from the last word to the first)

$n = 2$

0 0

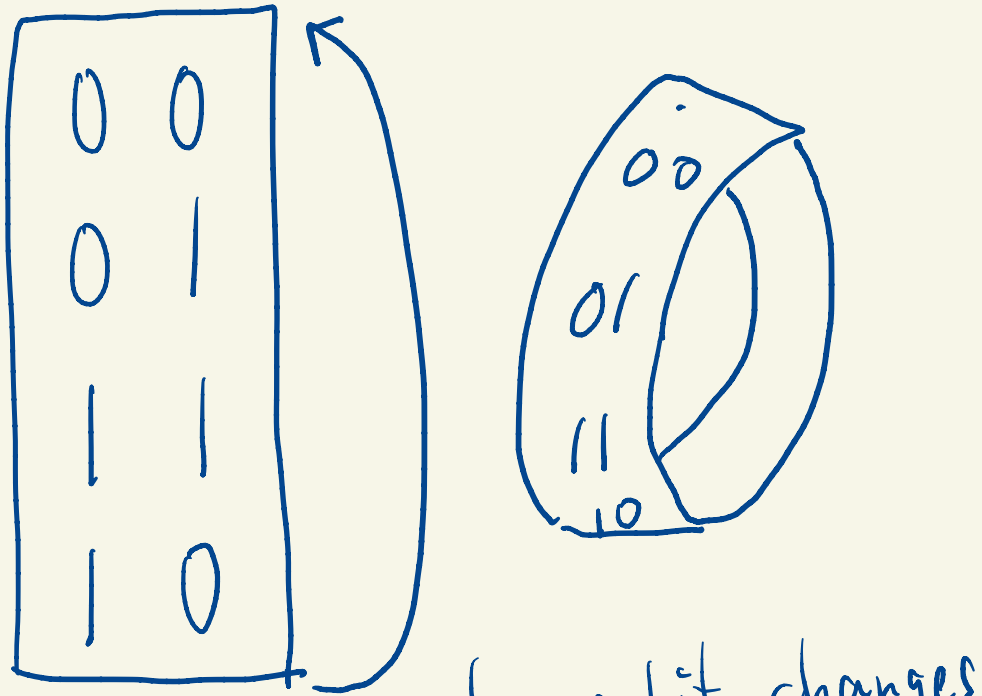
0 1

1 0

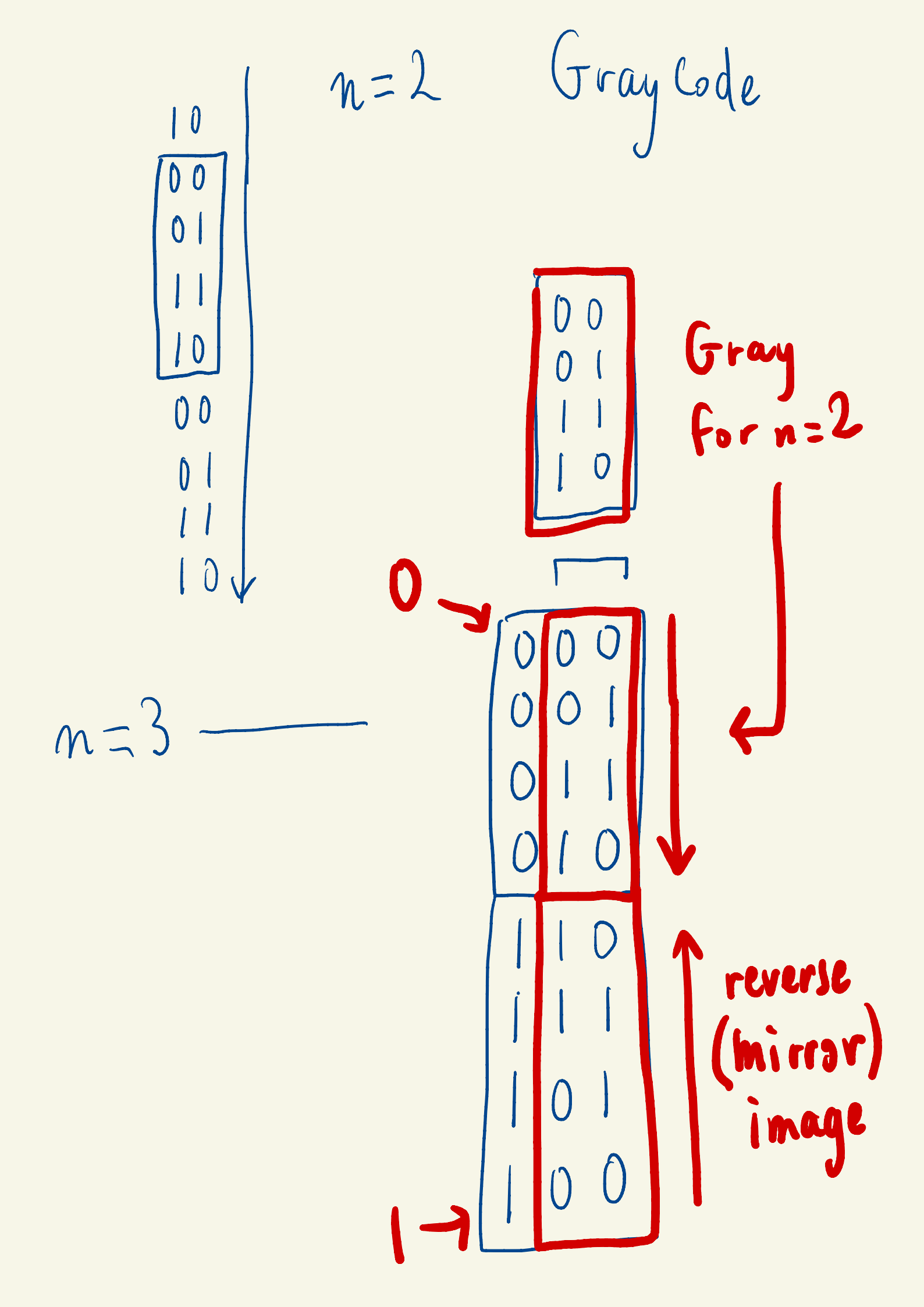
two bits changed

Wrong!

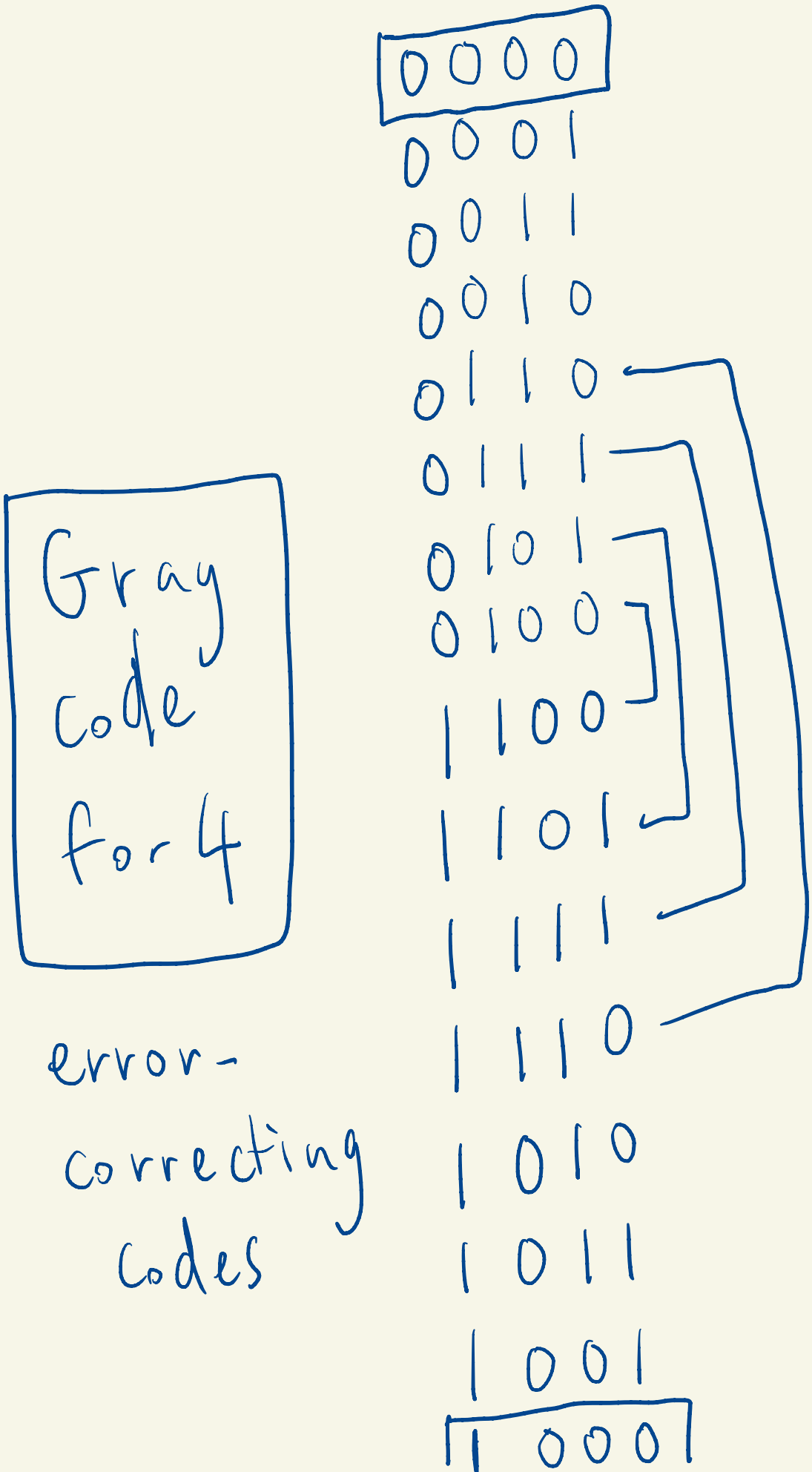
the solution is:




only one bit changes


n=2 Gray Code
10
00
01
11
10
00
01
11
10
n=3
00
01
11
10
Gray for n=2
0
000
001
011
010
110
111
101
100
1
reverse (mirror) image

# Gray for n = 4


0000
0001
0011
0010
0110
0111
0101
0100
1100
1101
1111
1110
1010
1011
1001
1000
Gray code for 4
error-correcting codes

The same construction

can be performed

by induction

for every natural number $n$.

# Same K-map with associated minterms

| Y, C \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |

| Y, C \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | $\bar{A}\bar{B}\bar{C}$ | $\bar{A}\bar{B}\bar{C}$ | $A\bar{B}\bar{C}$ | $A\bar{B}\bar{C}$ |
| 1 | $\bar{A}\bar{B}C$ | $\bar{A}BC$ | $ABC$ | $A\bar{B}C$ |

$\bar{A}B\bar{C}$

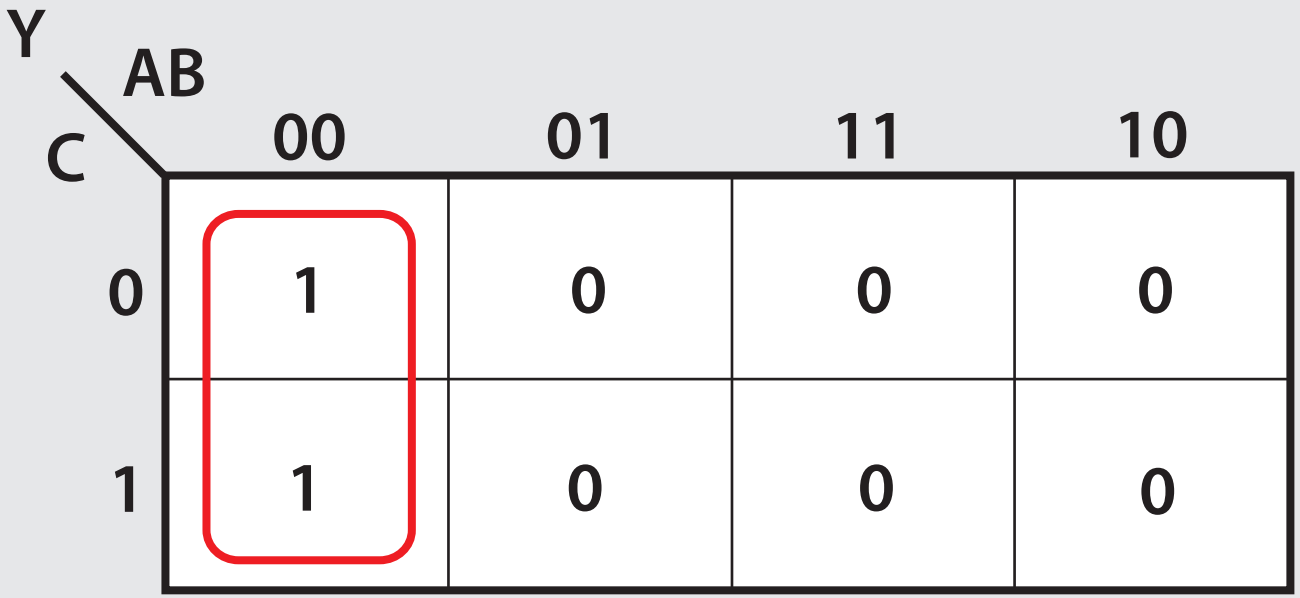
$A = 0$ and $B = 1$ and $C = 0$

$\bar{A}\bar{B}\bar{C}$ means $A = 0$ and $B = 0$ and $C = 0$

From this, one deduces that :

$$Y = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C$$

# K-map minimisation



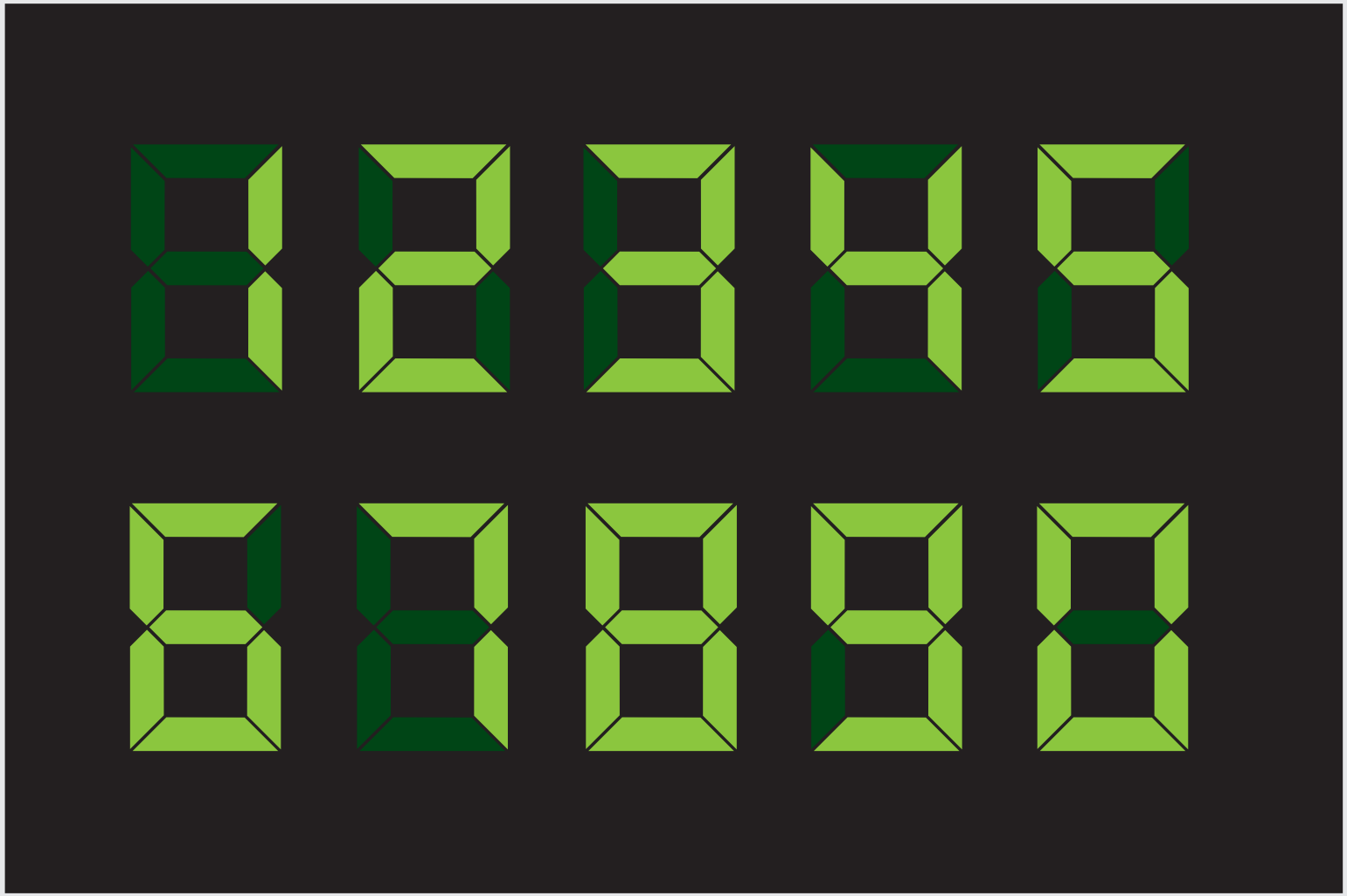

$\overline{A}\,\overline{B}$

A = 0 and B = 0

By circling the 1's in adjacent squares, one deduces that :

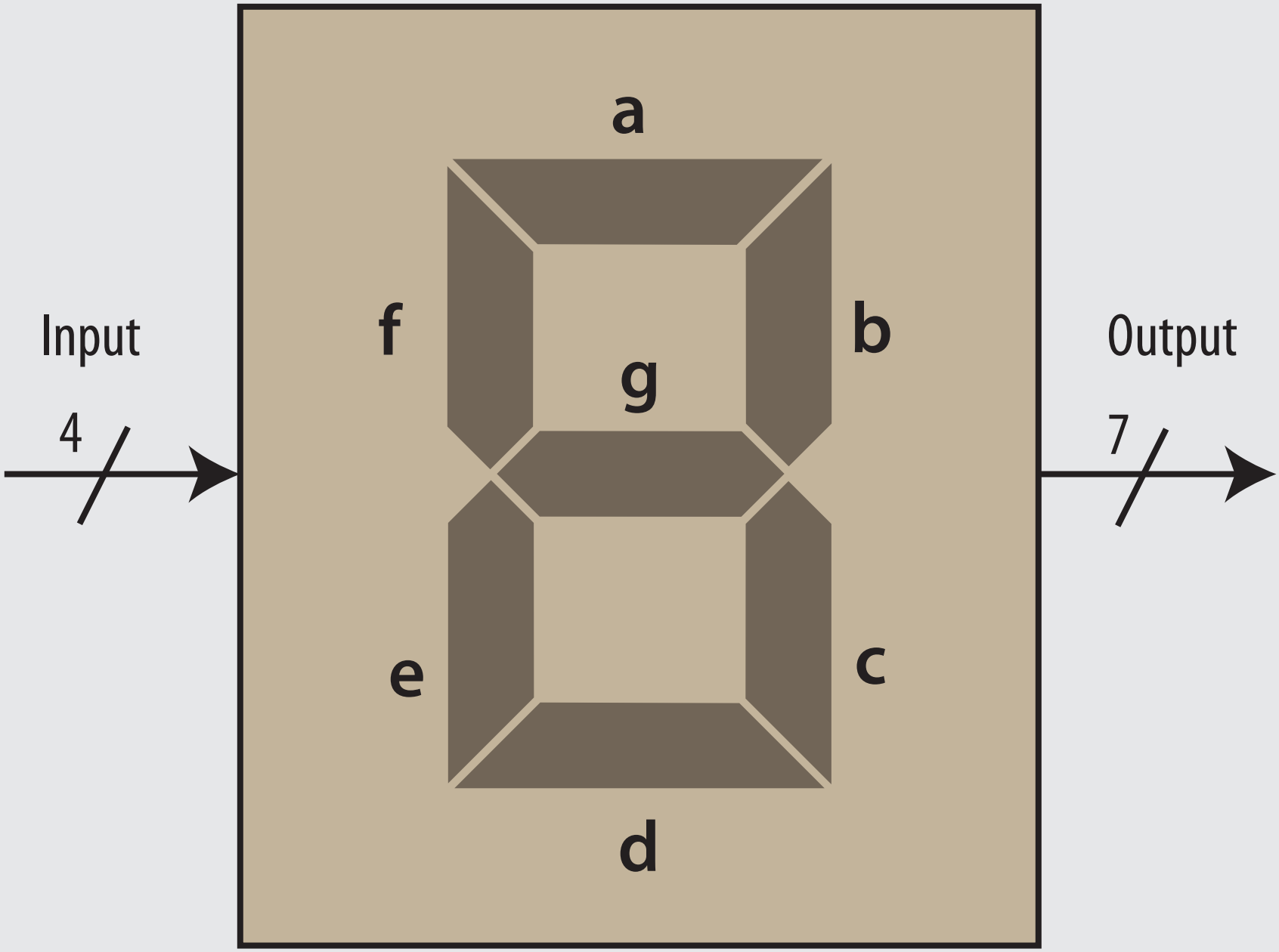
$$Y = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C = \overline{A}\,\overline{B}$$

# Seven-segment display digits



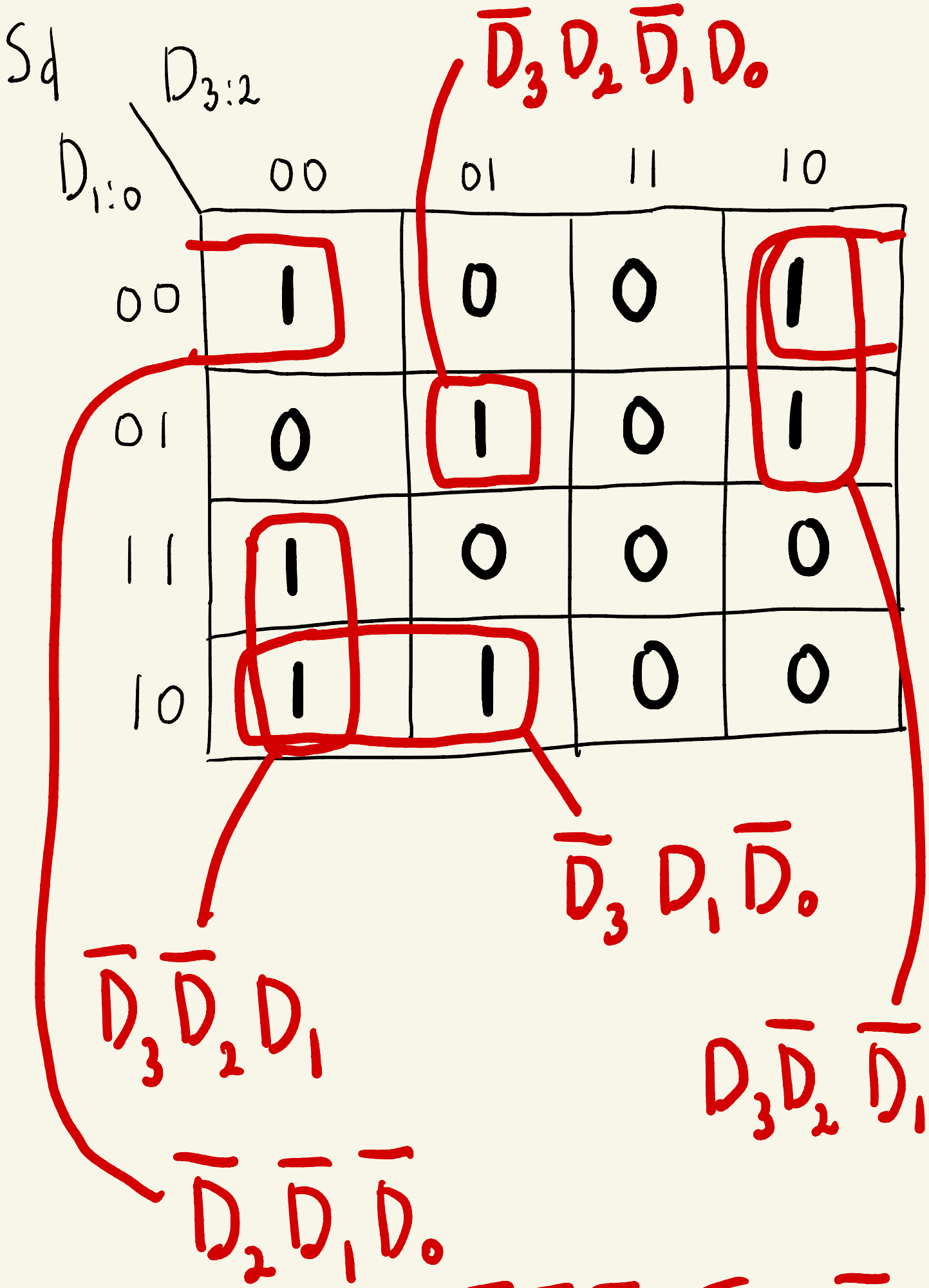
Seven-segment display digits were invented at the beginning of the 20th century.
They became very popular in the 1970's with the advent of LED.

# Seven-segment display decoder



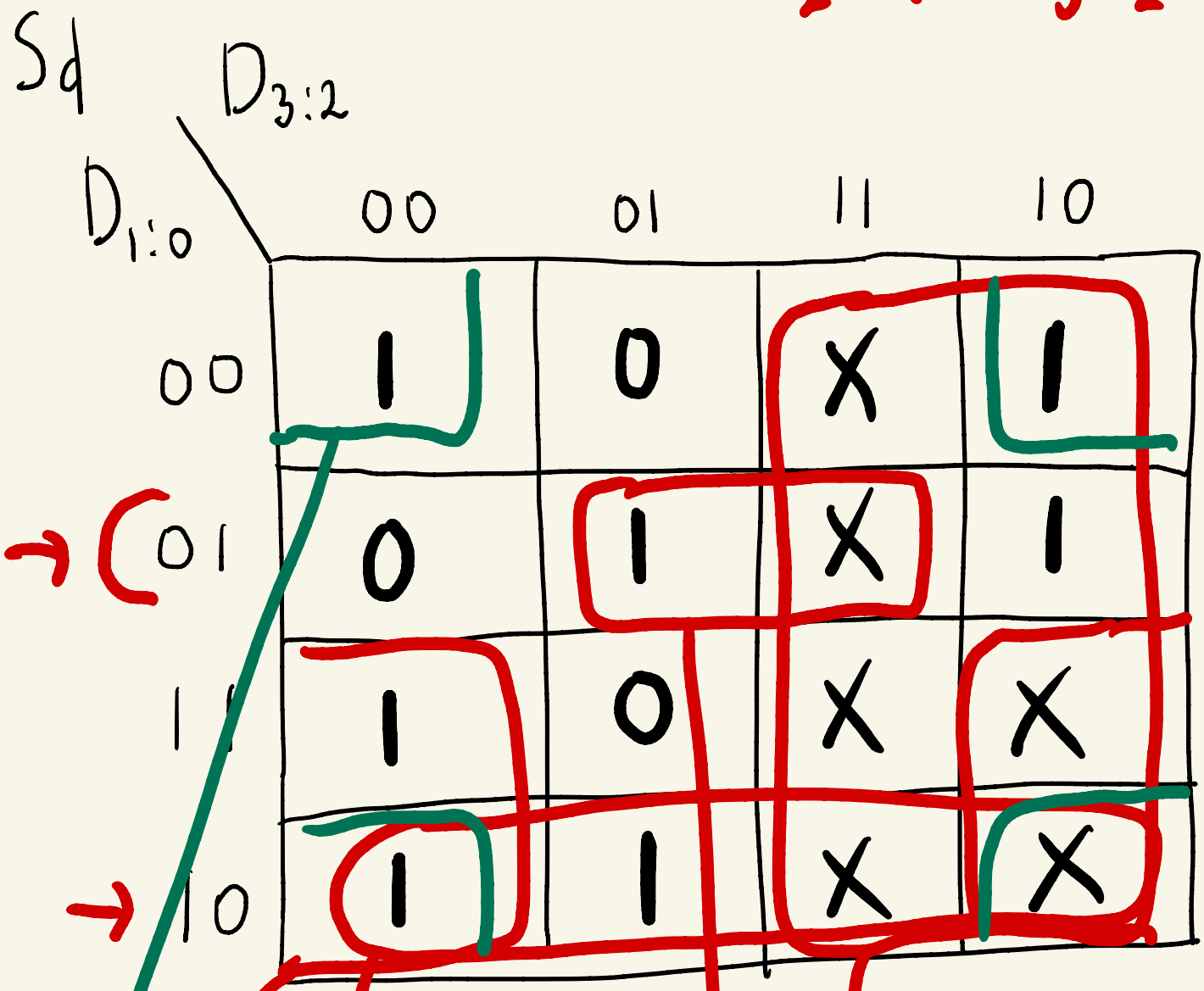
# Seven-segment display decoder truth table

| $D_{3:0}$ | Sa | Sb | Sc | Sd | Se | Sf | Sg |
|---|---|---|---|---|---|---|---|
| 0000 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0001 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0010 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0011 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0100 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0101 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0110 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0111 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1000 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1001 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| others | 0 | 0 | 0 | 0 | 0 | 0 | 0 |




$$Sd = \bar{D}_3\bar{D}_2D_1 + \bar{D}_2\bar{D}_1\bar{D}_0 + \bar{D}_3D_1\bar{D}_0$$

$+ D_3\bar{D}_2\bar{D}_1 + \bar{D}_3 D_2 \bar{D}_1 D_0$




$D_1\bar{D}_0$

$D_3$

$\bar{D}_2 D_1$

$D_2\bar{D}_1 D_0$

$\bar{D}_2\bar{D}_0$

$$S_d = D_3 + D_1\bar{D}_0 + \bar{D}_2 D_1 + \bar{D}_2\bar{D}_0$$

# Decoder K-maps minimisation

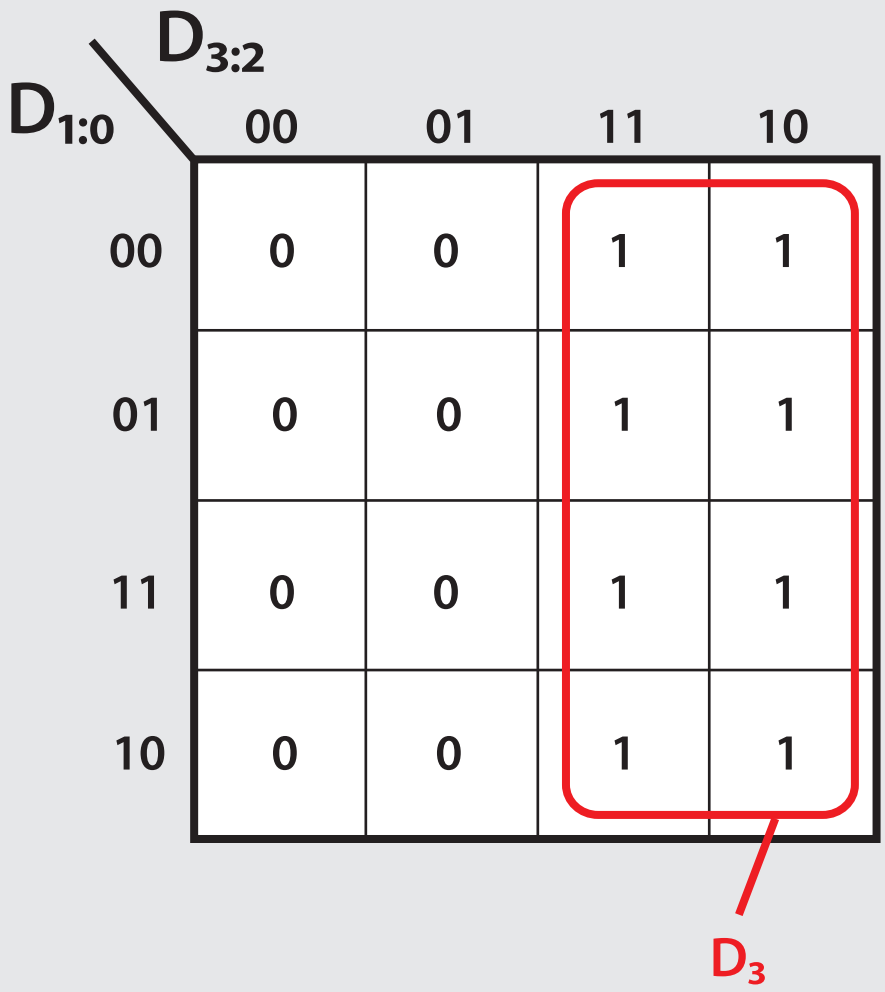
Sa

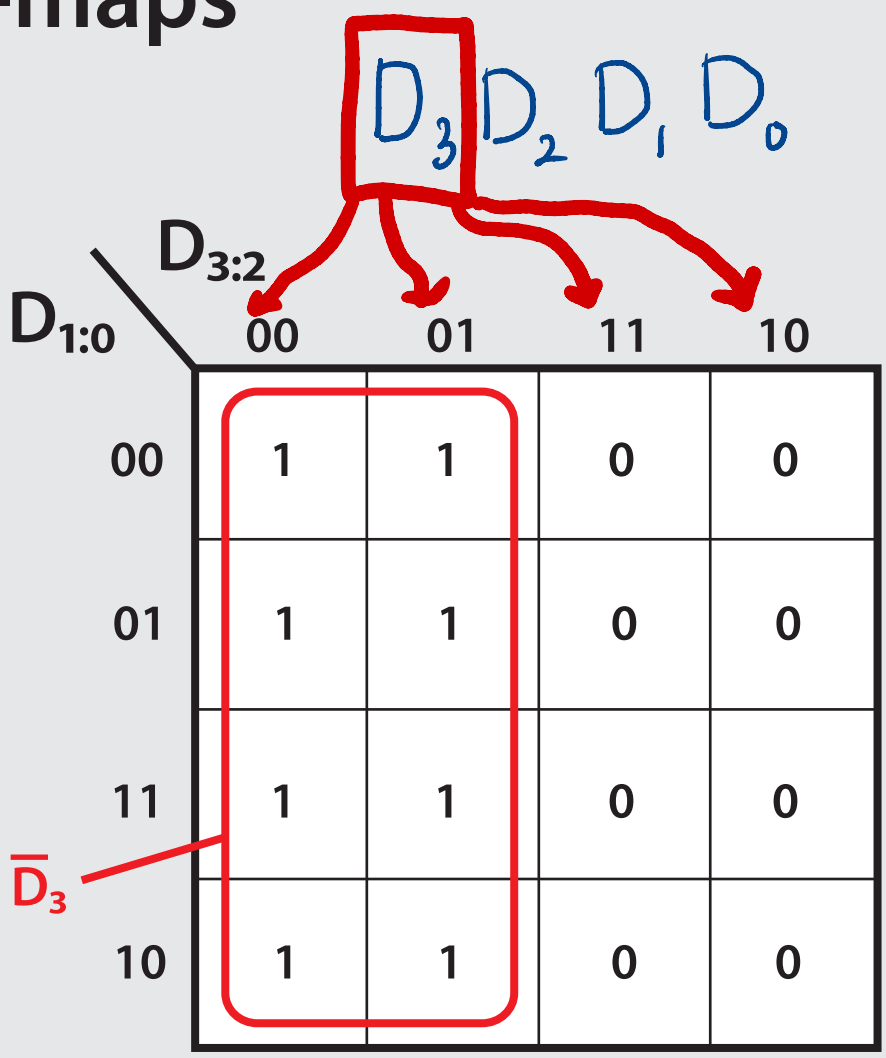
| $D_{1:0}$ \ $D_{3:2}$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 1 |
| 01 | 0 | 1 | 0 | 1 |
| 11 | 1 | 1 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |

Sb

| $D_{1:0}$ \ $D_{3:2}$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 1 | 0 | 0 |
| 10 | 1 | 0 | 0 | 0 |

# Decoder K-maps



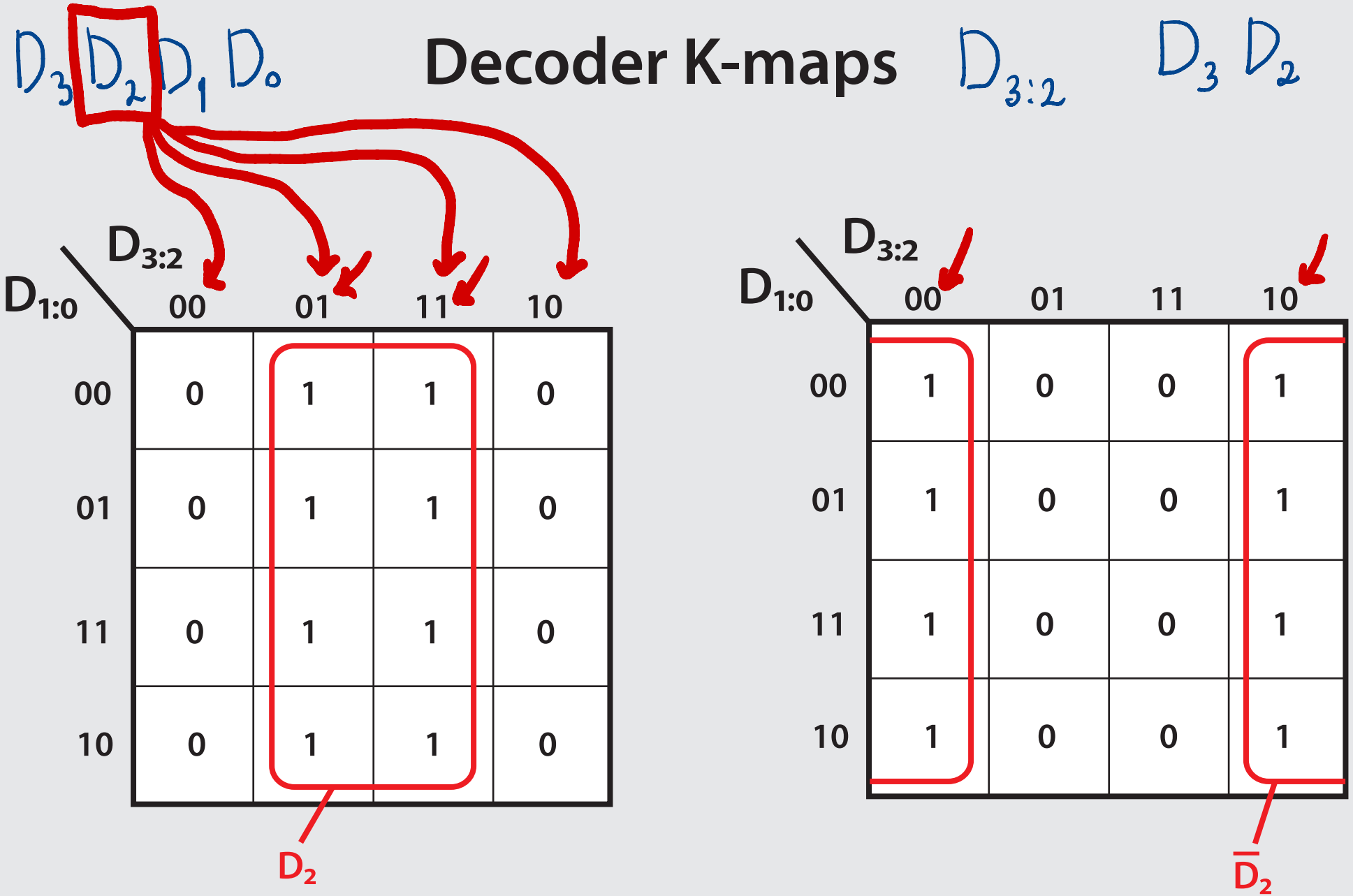




$D_3$ denotes the set of inputs with input bit $D_3$ equal to 1

$\overline{D}_3$ denotes the set of inputs with input bit $D_3$ equal to 0

# Decoder K-maps

$D_3\ D_2\ D_1\ D_0$ $D_{3:2}$ $D_3\ D_2$

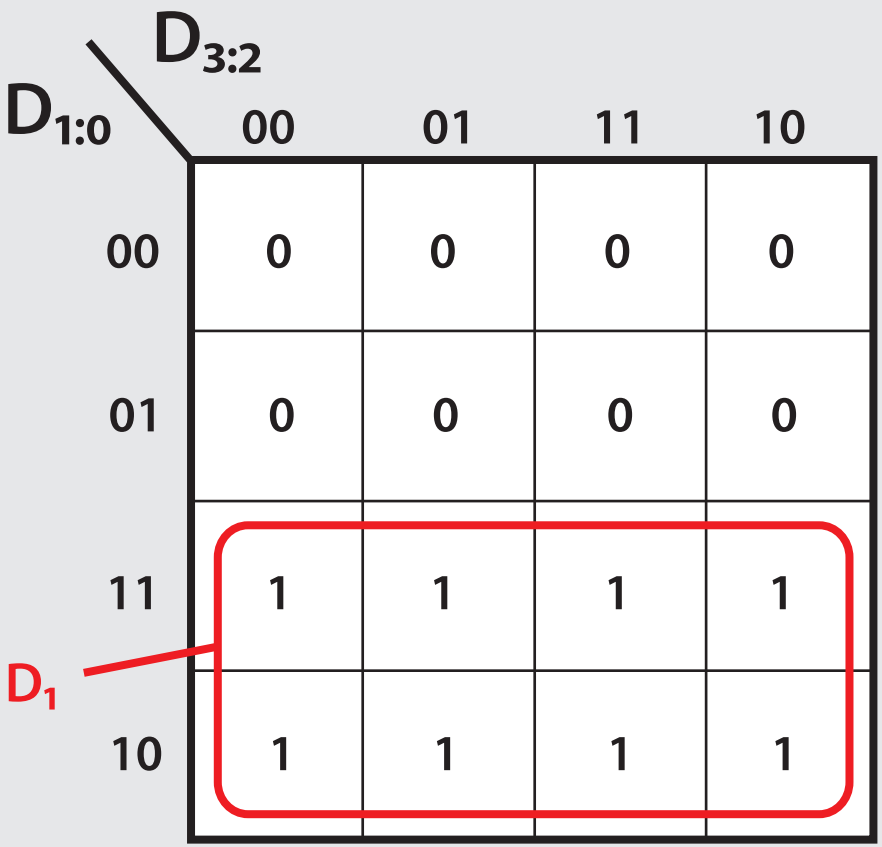


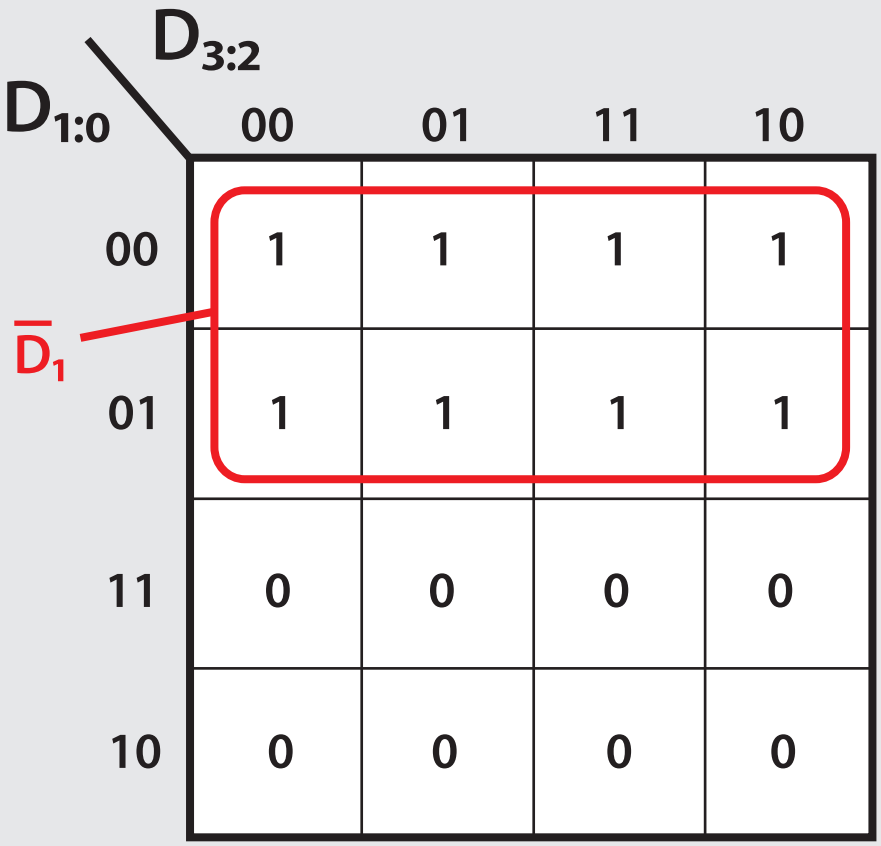

$D_2$ denotes the set of inputs with input bit $D_2$ equal to 1

$\overline{D}_2$ denotes the set of inputs with input bit $D_2$ equal to 0

# Decoder K-maps

$D_{1:0} = D_1 D_0$

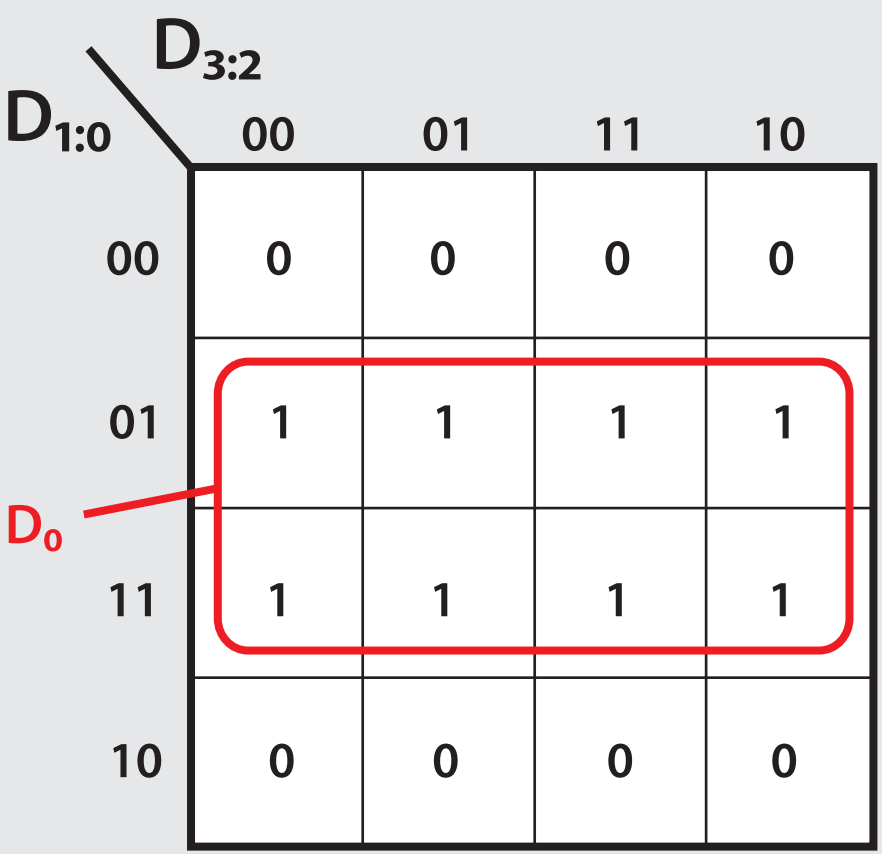


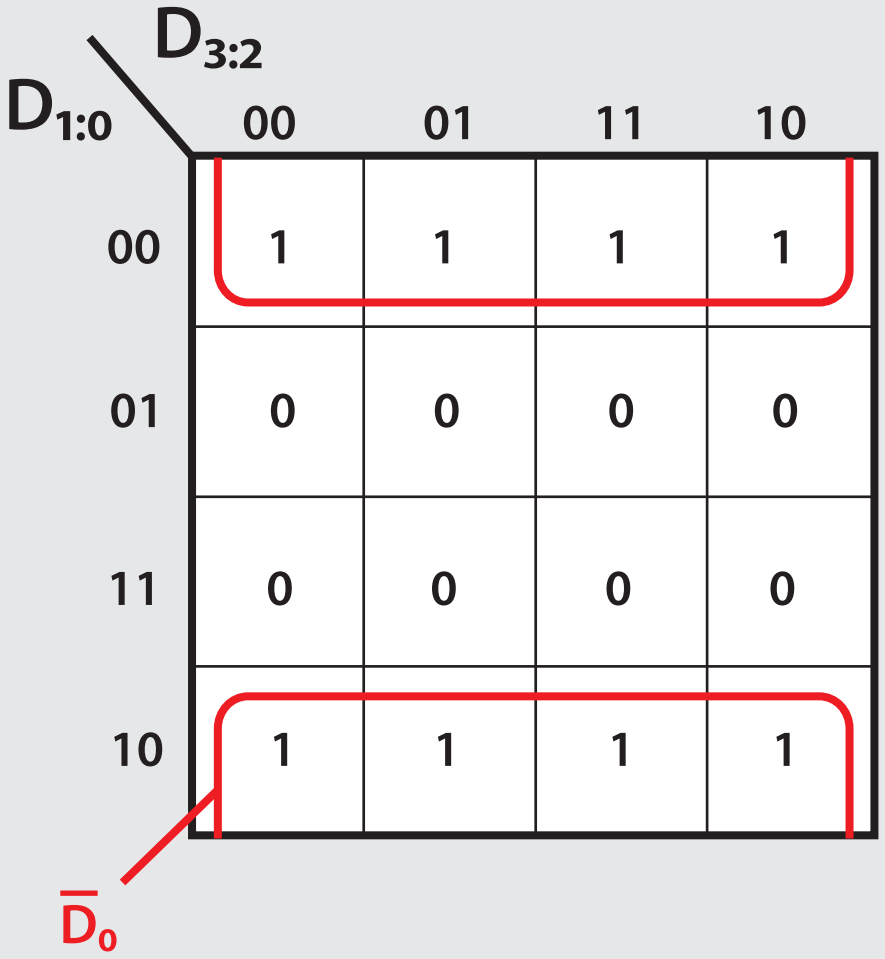




$D_1$ denotes the set of inputs with input bit $D_1$ equal to 1

$\overline{D}_1$ denotes the set of inputs with input bit $D_1$ equal to 0

# Decoder K-maps

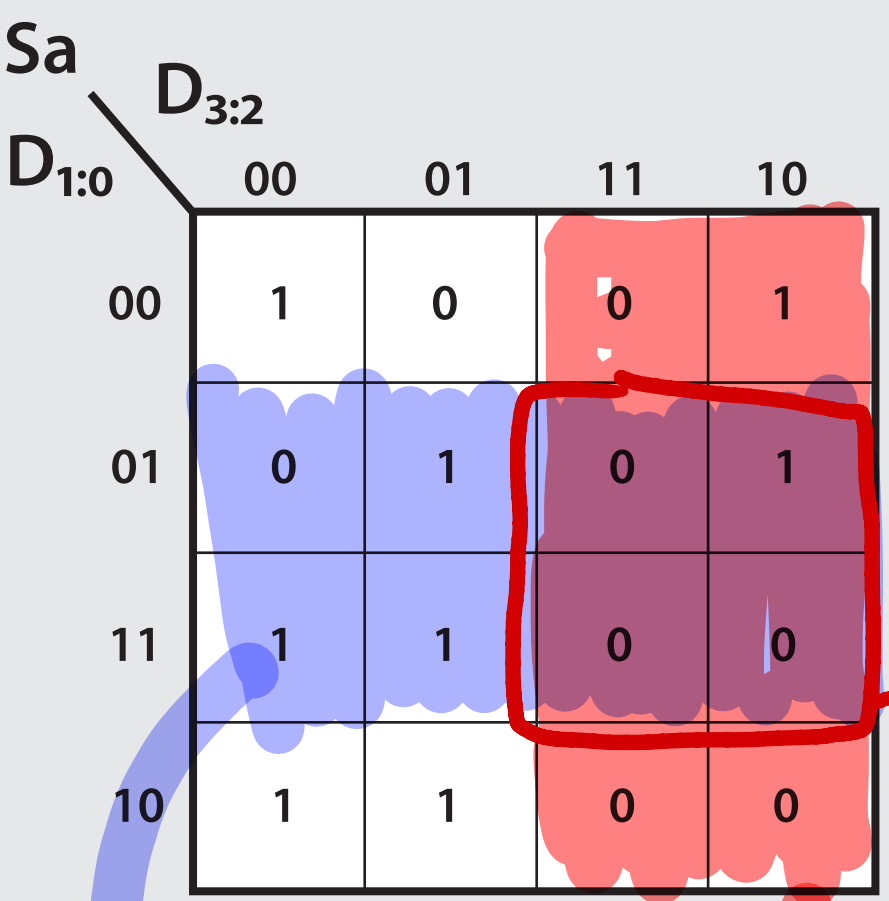


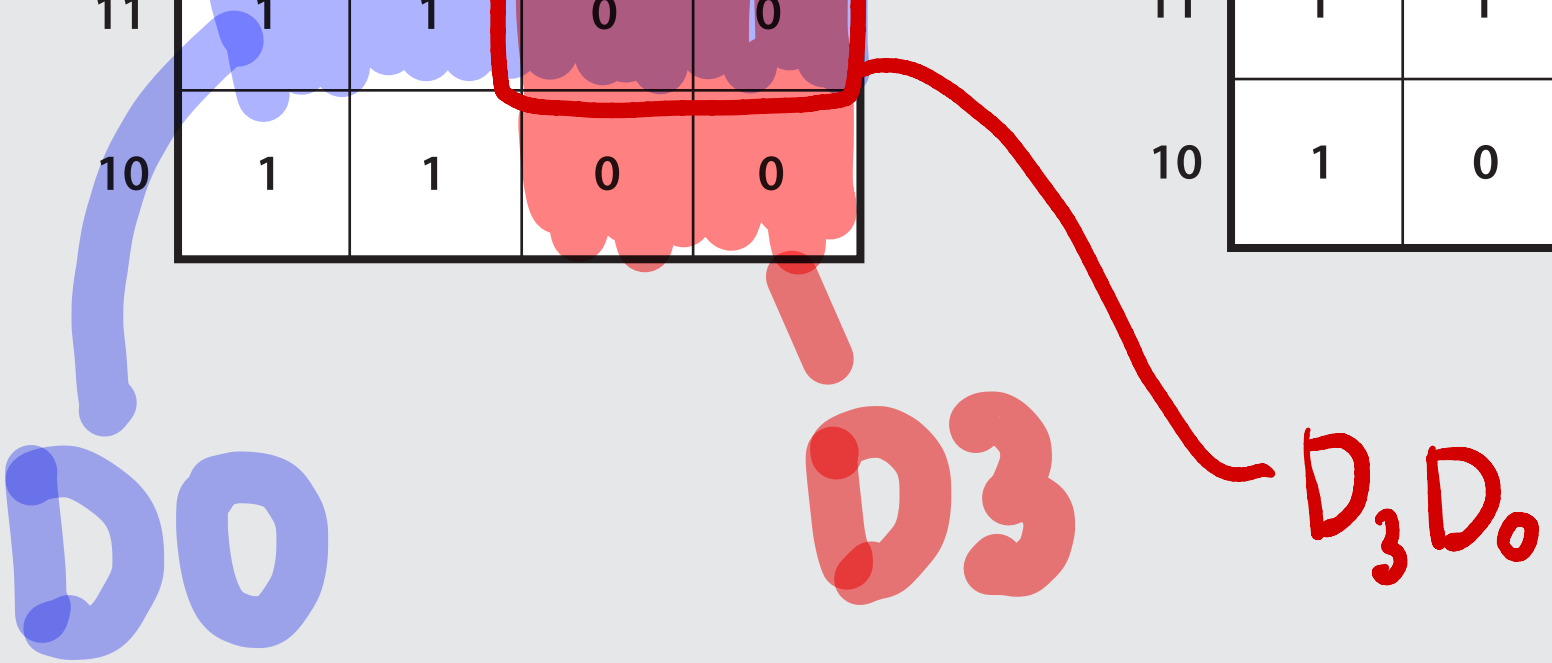

$D_0$ denotes the set of inputs with input bit $D_0$ equal to 1

$\overline{D}_0$ denotes the set of inputs with input bit $D_0$ equal to 0

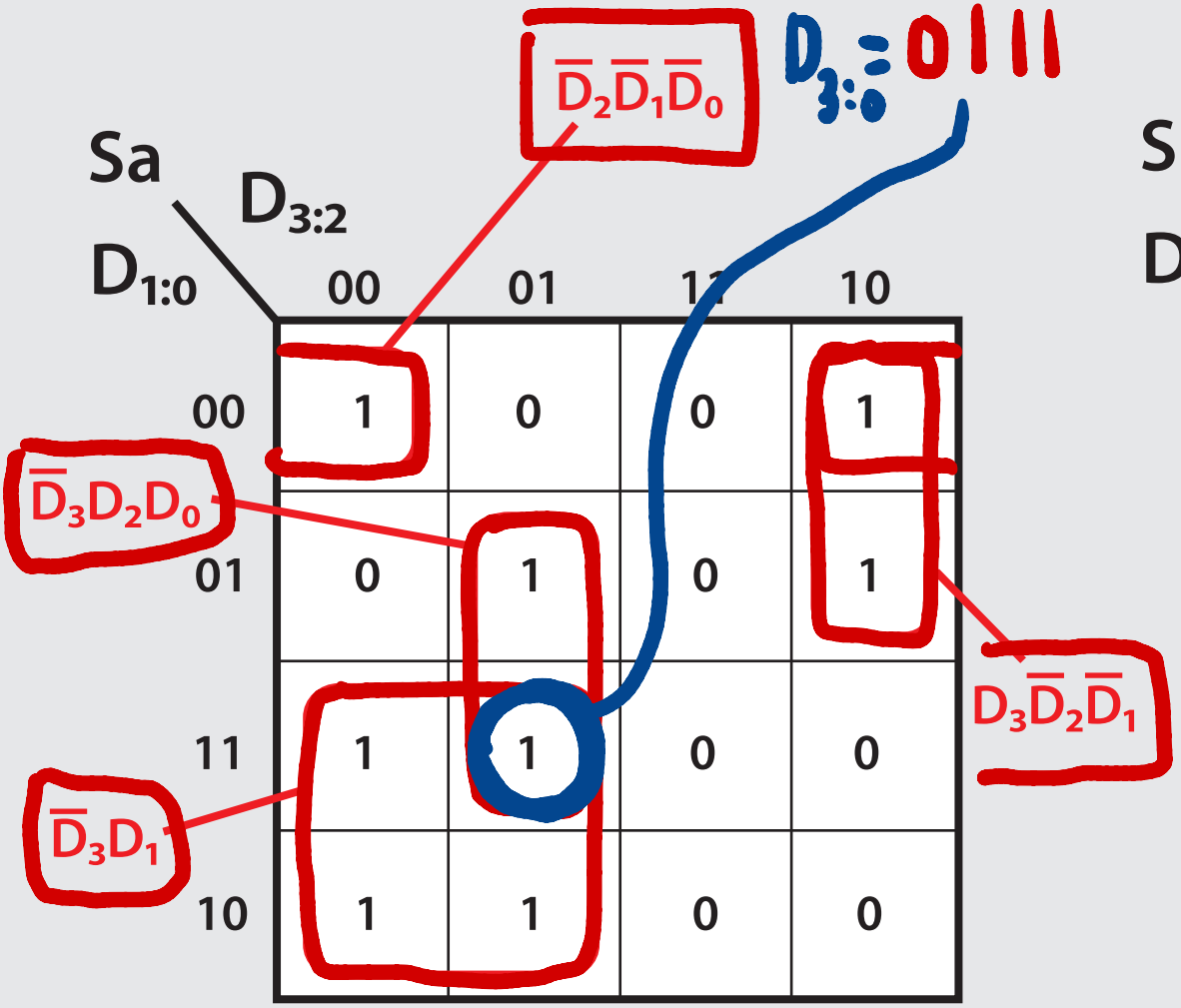
# Decoder K-maps minimisation

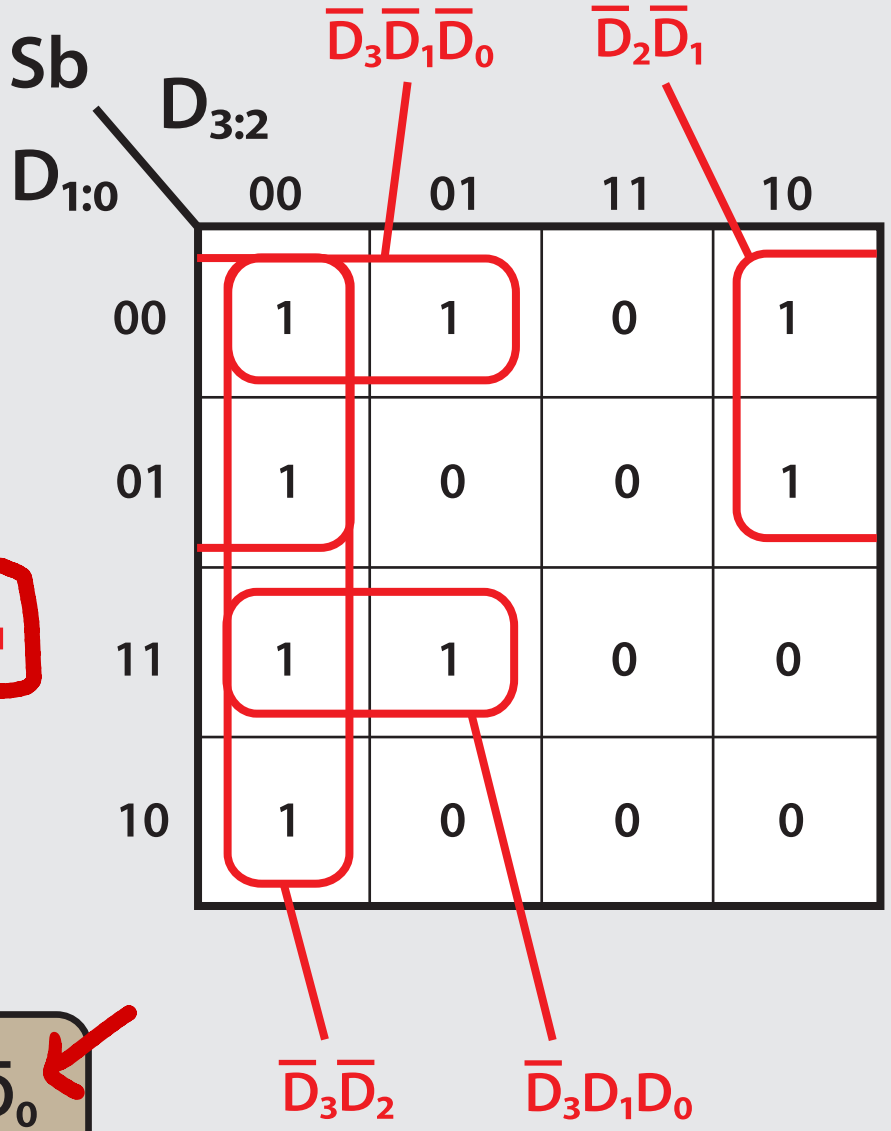



| Sb $D_{1:0}$ \ $D_{3:2}$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | 0 | 1 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | 1 | 1 | 0 | 0 |
| 10 | 1 | 0 | 0 | 0 |



# Decoder K-maps minimisation

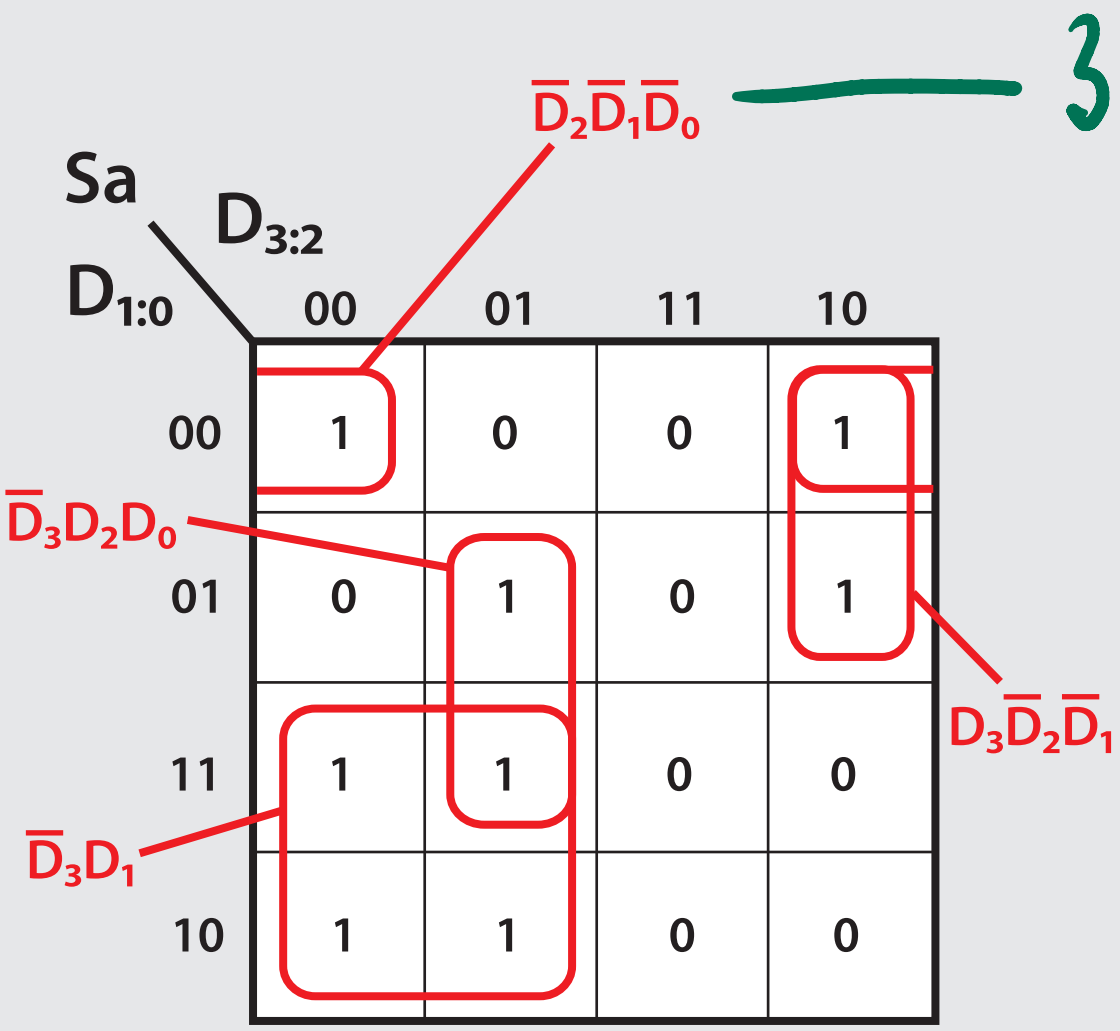



$$Sa = \overline{D}_3D_1 + \overline{D}_3D_2D_0 + D_3\overline{D}_2\overline{D}_1 + \overline{D}_2\overline{D}_1\overline{D}_0$$

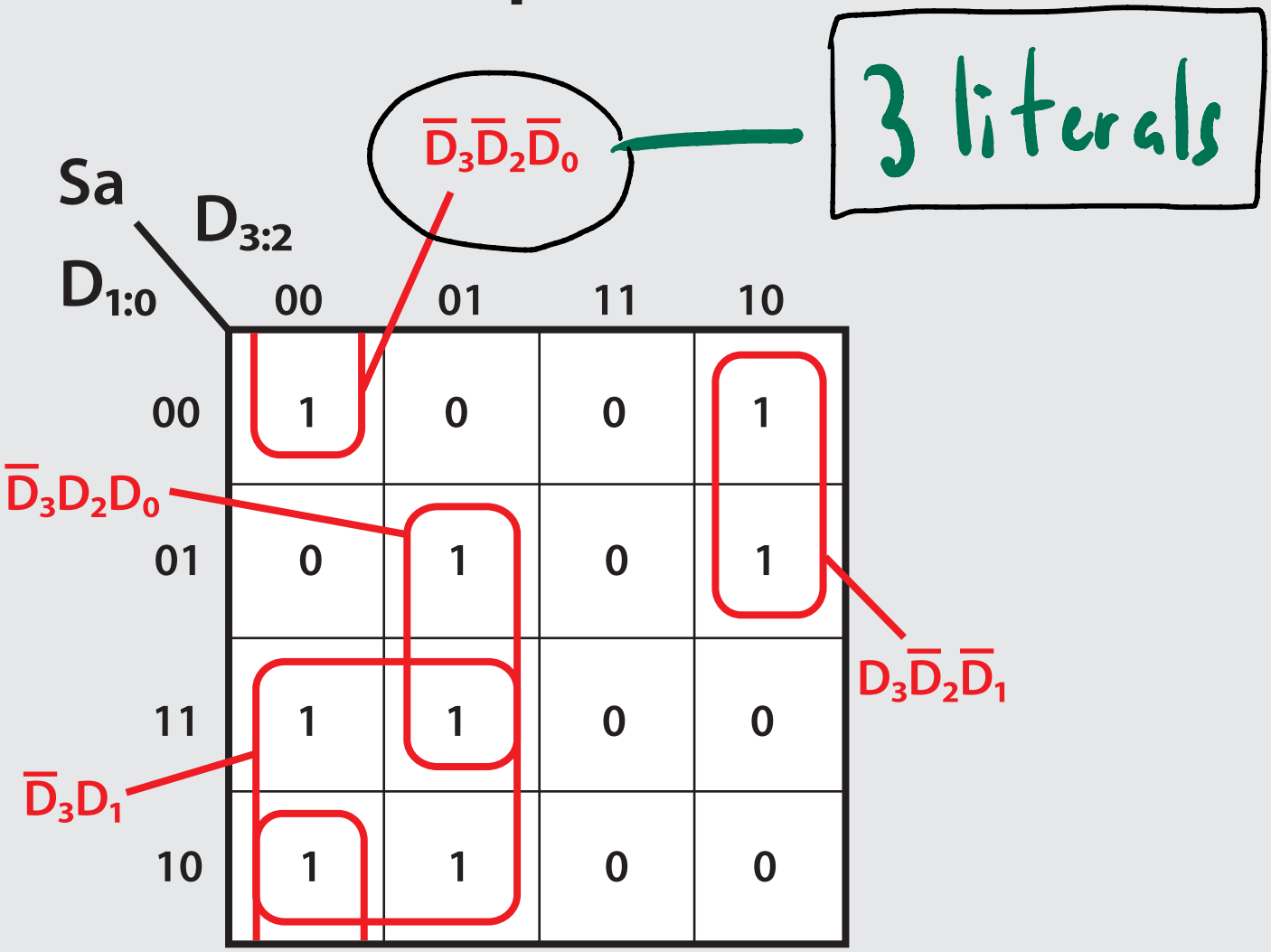
$$Sb = \overline{D}_3\overline{D}_2 + \overline{D}_2\overline{D}_1 + \overline{D}_3D_1D_0 + \overline{D}_3\overline{D}_1\overline{D}_0$$

# Decoder K-maps minimisation




$$Sa = \overline{D}_3D_1 + \overline{D}_3D_2D_0 + D_3\overline{D}_2\overline{D}_1 + \overline{D}_2\overline{D}_1\overline{D}_0$$

# Decoder K-maps minimisation



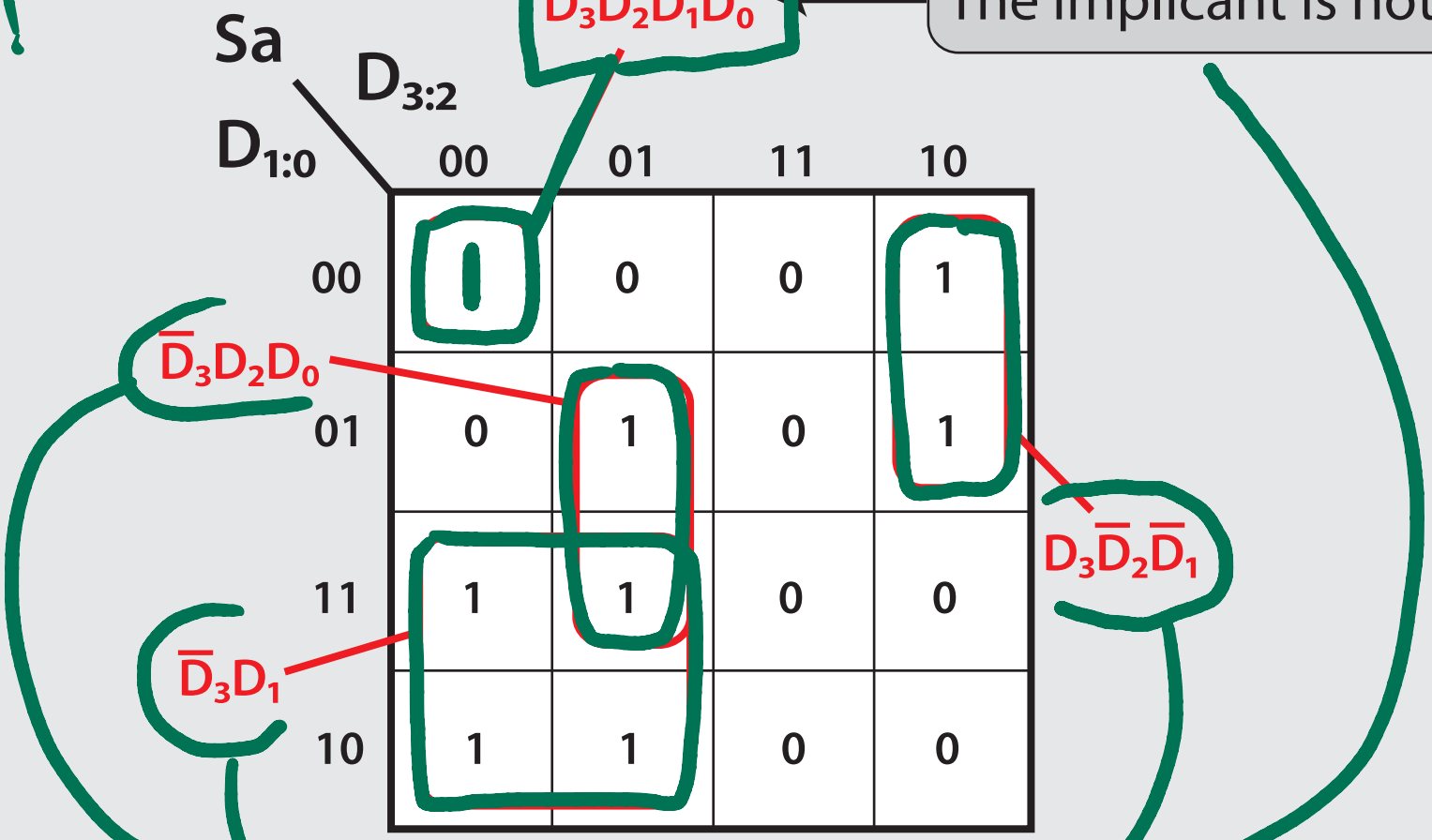

$$Sa = \overline{D}_3D_1 + \overline{D}_3D_2D_0 + D_3\overline{D}_2\overline{D}_1 + \overline{D}_3\overline{D}_2\overline{D}_0$$

# A common mistake

4 literals!

$\overline{D}_3\overline{D}_2\overline{D}_1\overline{D}_0$

The implicant is not prime



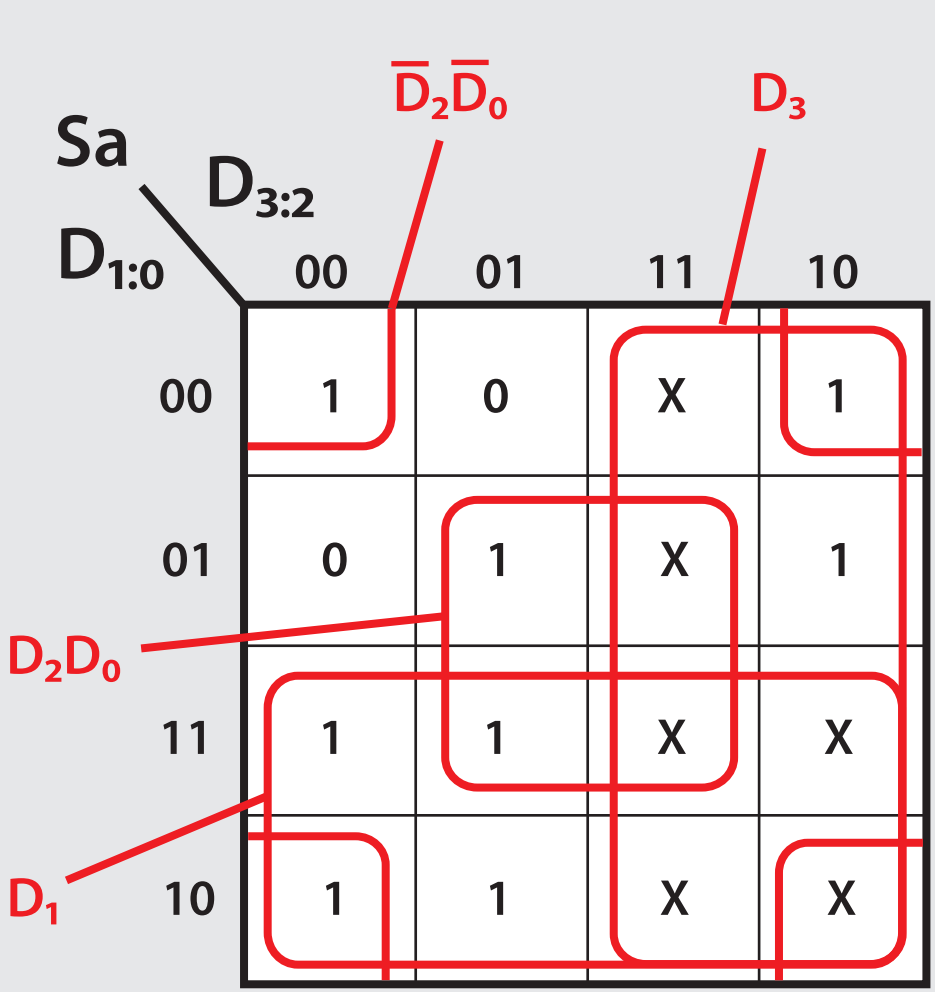

$$Sa = \overline{D}_3D_1 + \overline{D}_3D_2D_0 + D_3\overline{D}_2\overline{D}_1 + \overline{D}_3\overline{D}_2\overline{D}_1\overline{D}_0$$

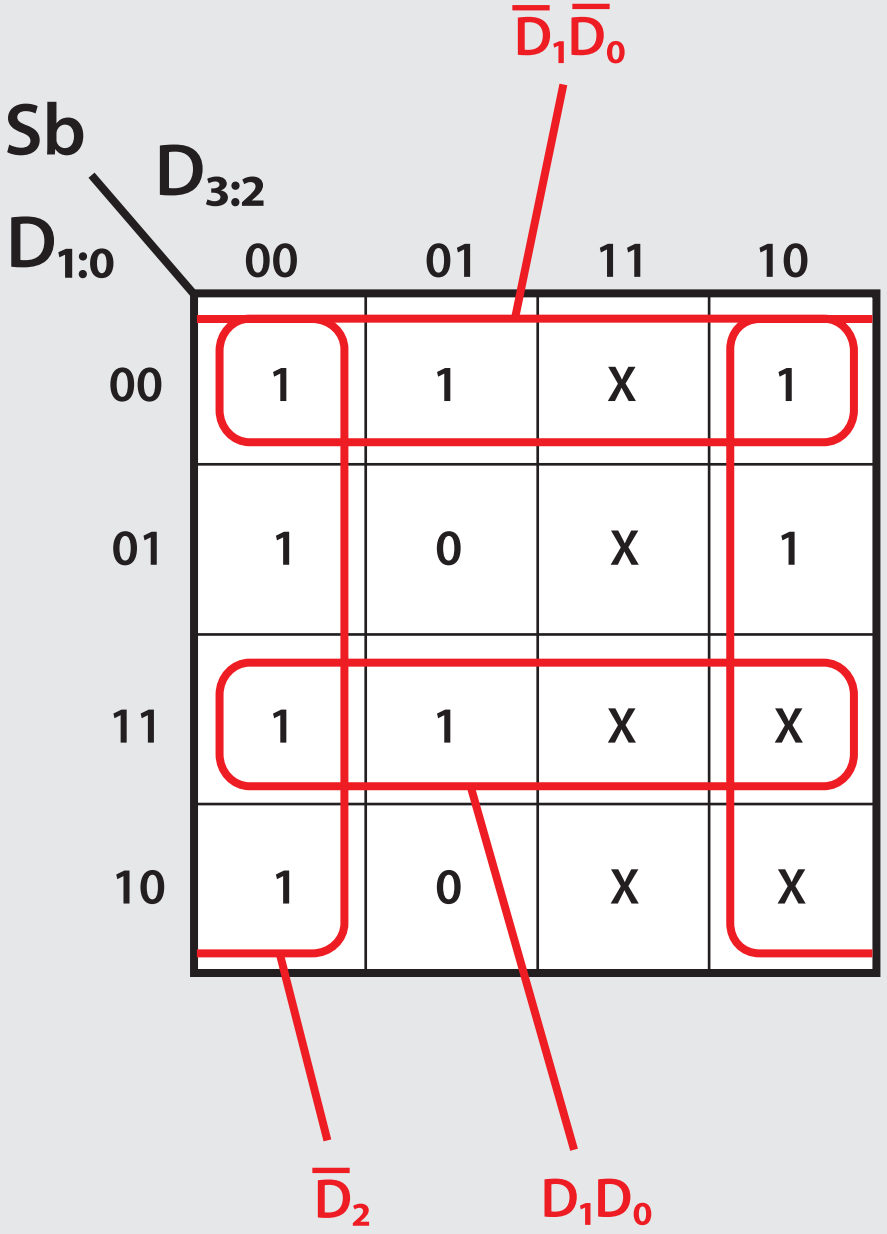
# Seven-segment display decoder truth table with X's

| $D_{3:0}$ | Sa | Sb | Sc | Sd | Se | Sf | Sg |
|---|---|---|---|---|---|---|---|
| 0000 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0001 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0010 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0011 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0100 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0101 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0110 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0111 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1000 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1001 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| others | X | X | X | X | X | X | X |

Here, the value **X** means that the value is undetermined : it can be 0 or 1.

# Decoder K-maps minimisation with X's




$$Sa = D_3 + D_2D_0 + \overline{D}_2\overline{D}_0 + D_1$$

$$Sb = \overline{D}_2 + D_1D_0 + \overline{D}_1\overline{D}_0$$

# Exercise 1

Complete the design of the seven-segment decoder
by designing boolean equations for the segments Sc and Sd :

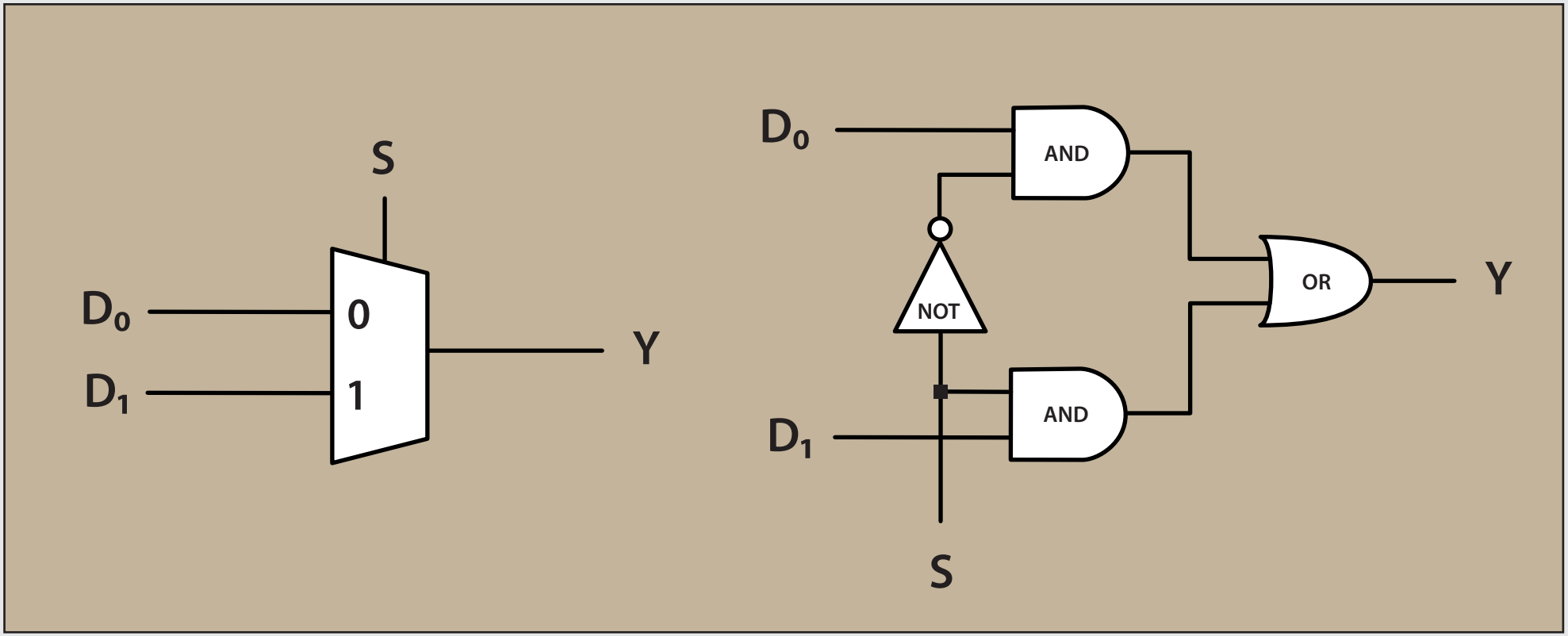
a. assuming that inputs greater than 9
must produce blank ( 0 ) outputs

b. assuming that inputs greater than 9
are don't cares

Then, sketch a reasonably simple gate-level implementation
in the case b. and simulate the resulting circuits on CircuitLab.

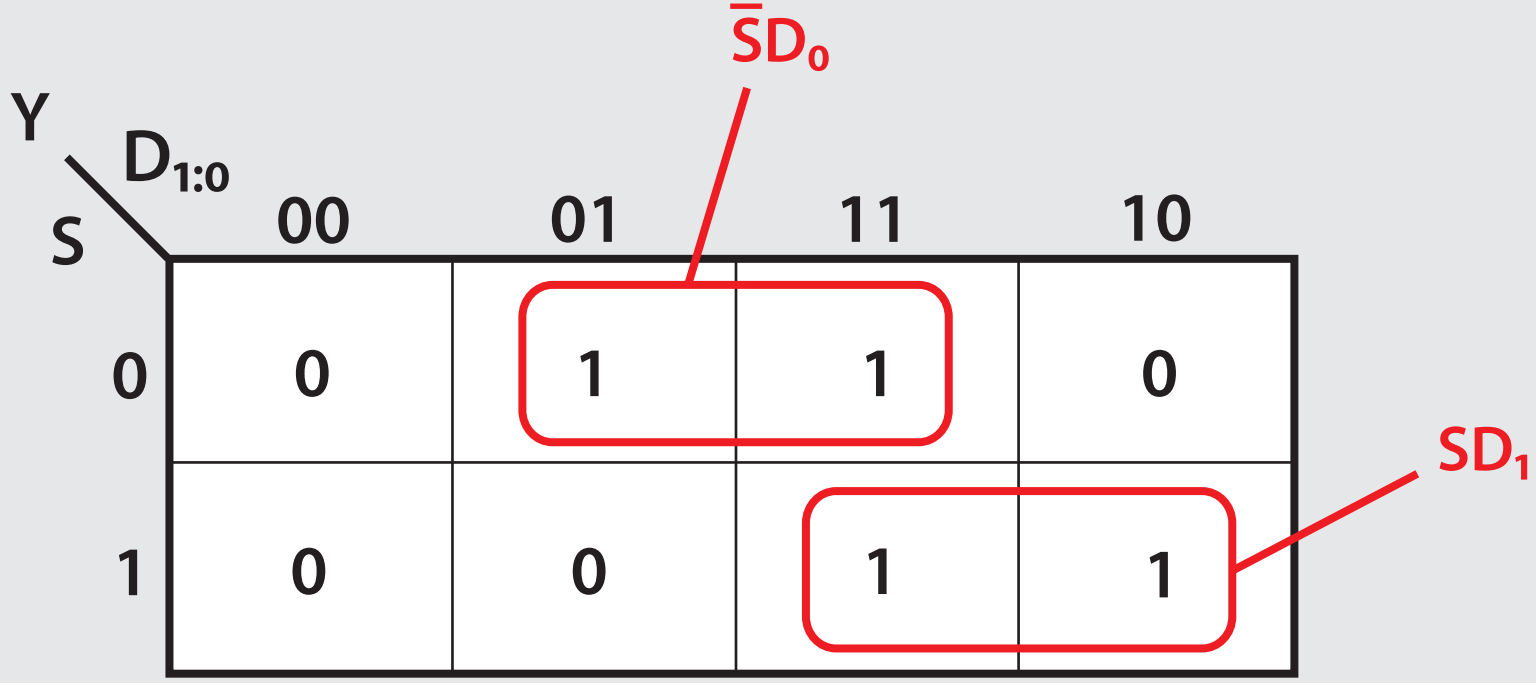
# Combinational Building Blocks

# Multiplexor ( mux )




| S | Y |
|---|---|
| 0 | $D_0$ |
| 1 | $D_1$ |

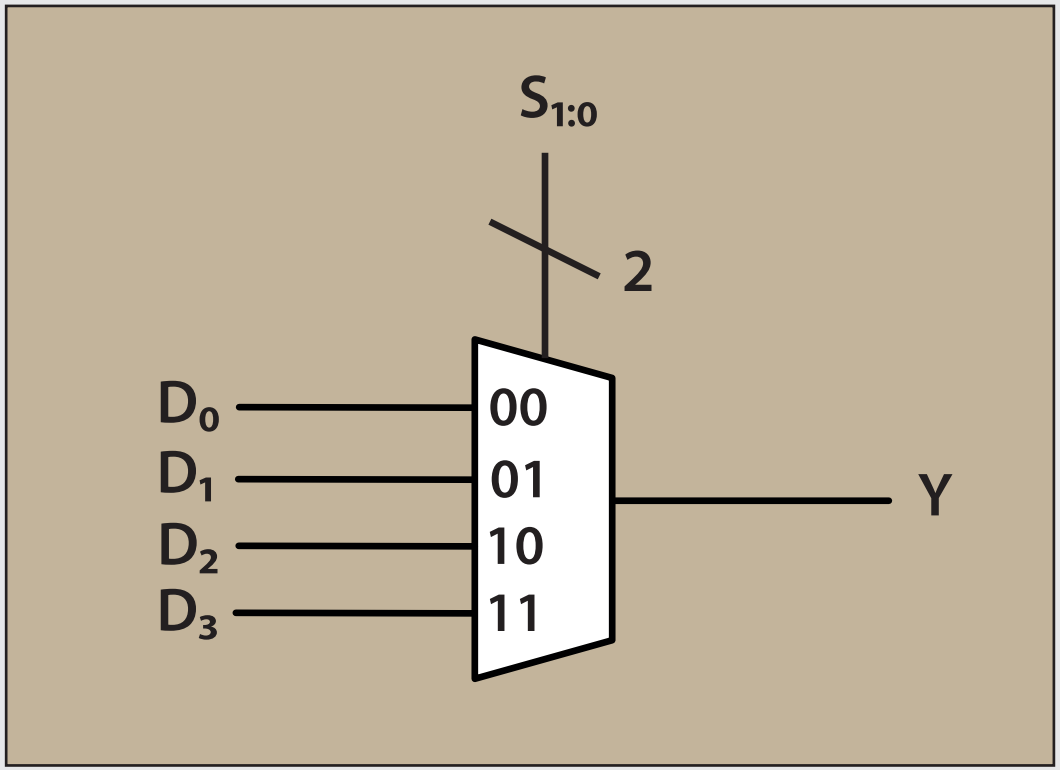
# K-map minimisation



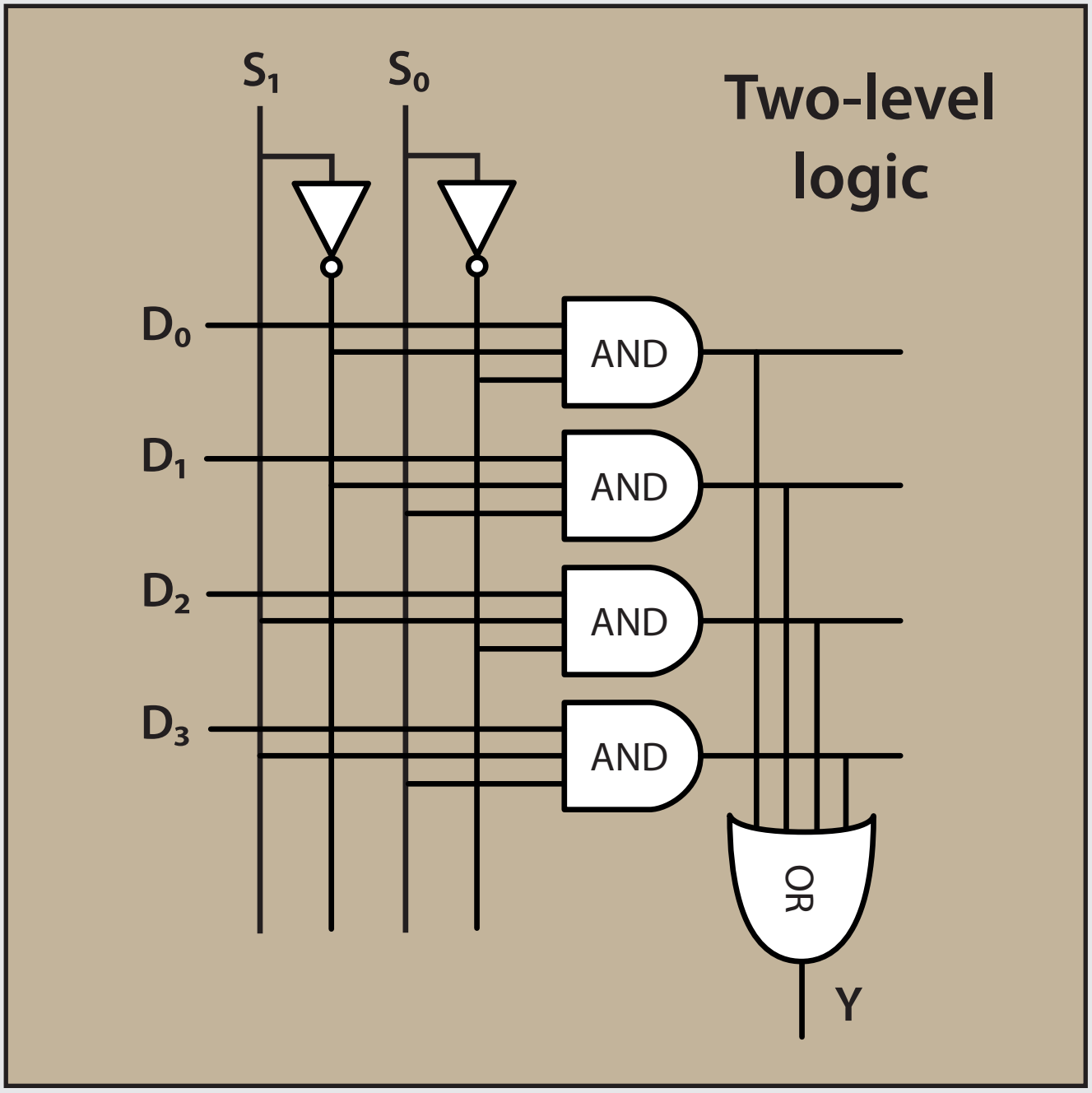

By circling the 1's in adjacent squares, one deduces that :

$$Y = SD_1 + \bar{S}D_0$$

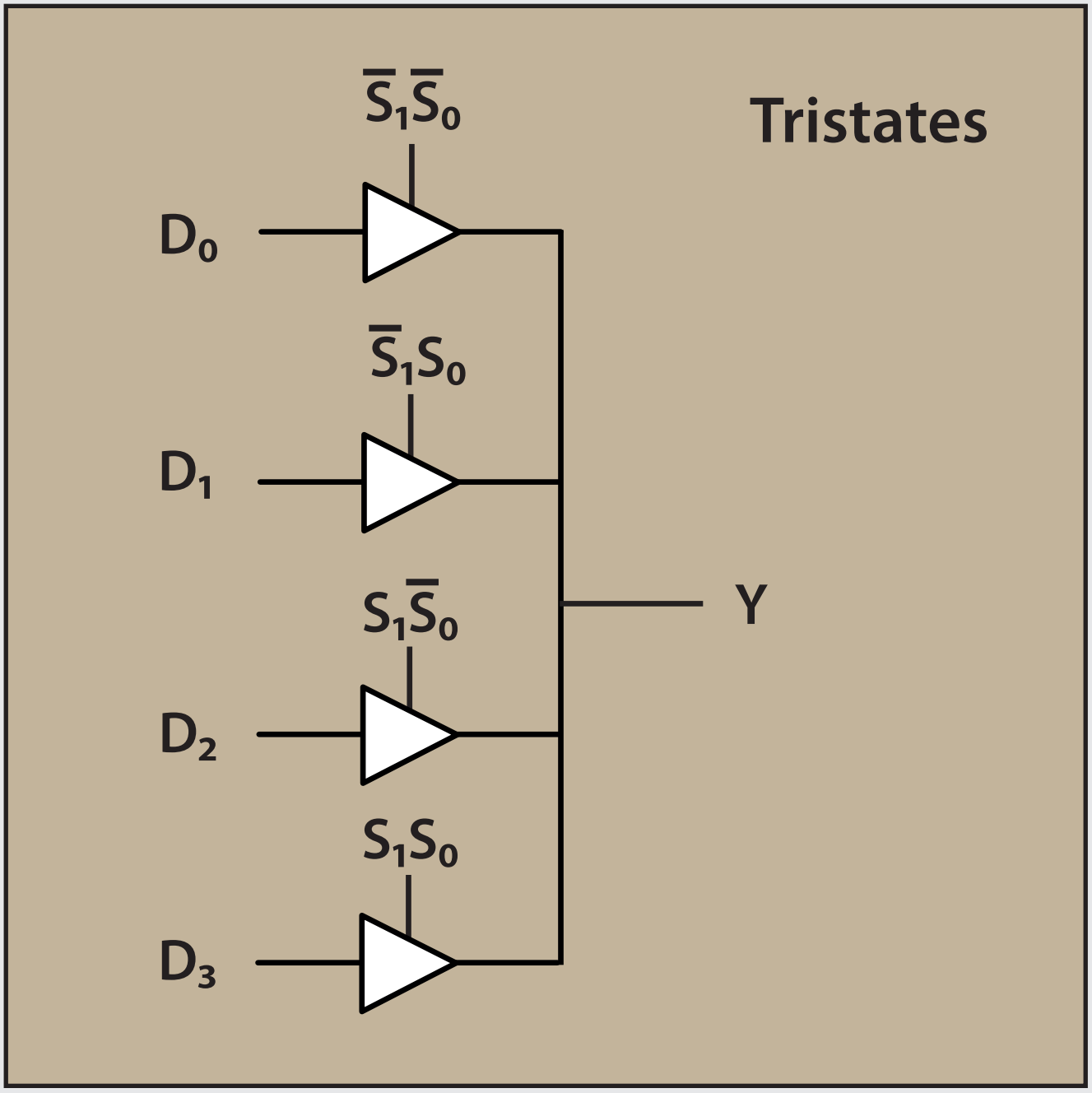
# 4 : 1 multiplexers



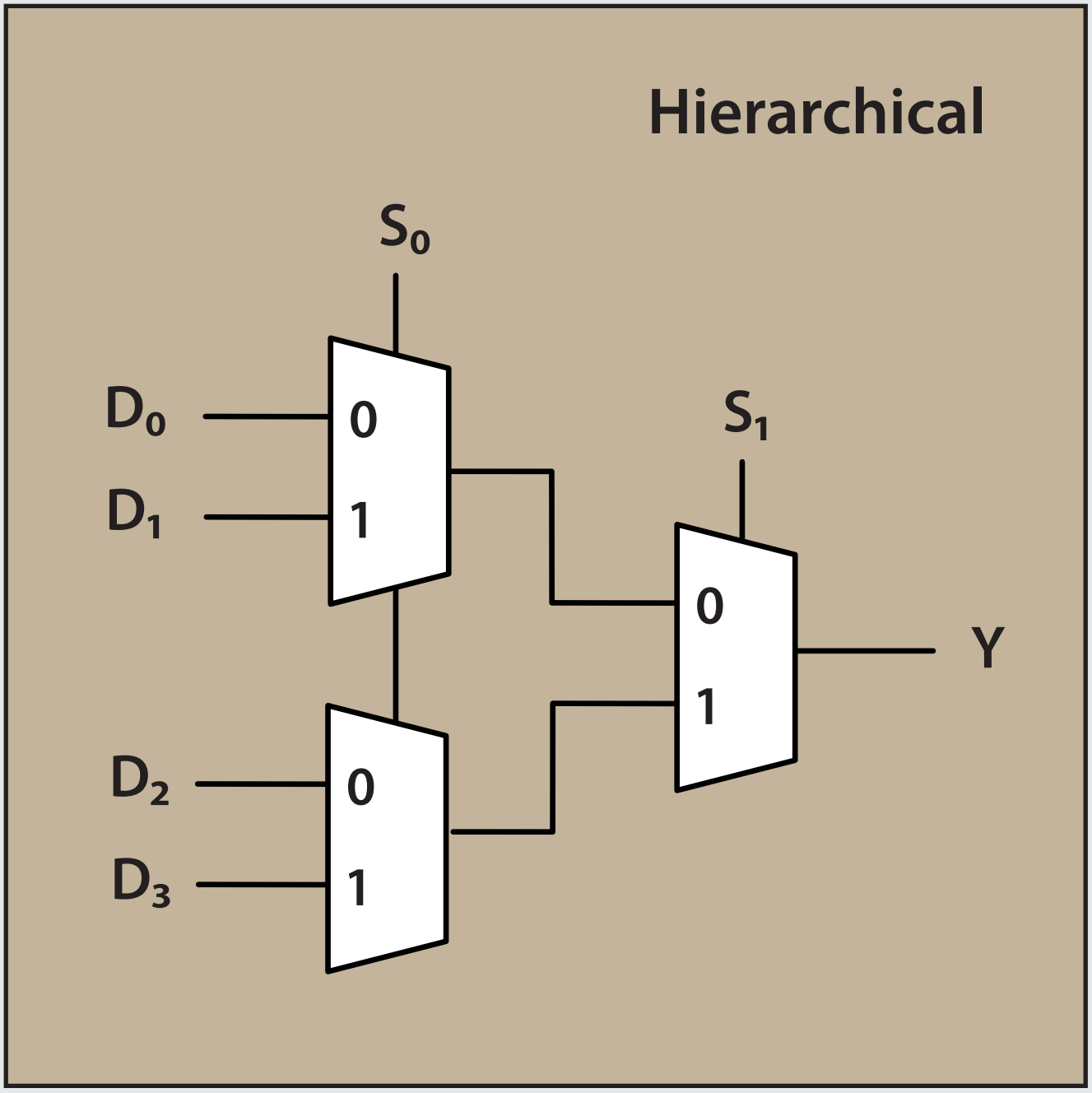
# Implementation of the 4:1 multiplexer



# Implementation of the 4:1 multiplexer

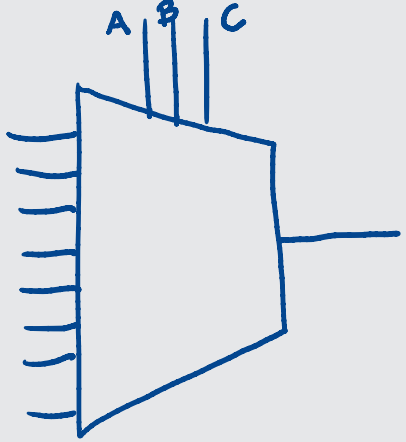


# Implementation of the 4:1 multiplexer



# Exercise

( Example 2.12 in Harris and Harris )




Alyssa P. Hacker needs to implement the function

$$Y = A\bar{B} + \bar{B}\bar{C} + \bar{A}BC$$

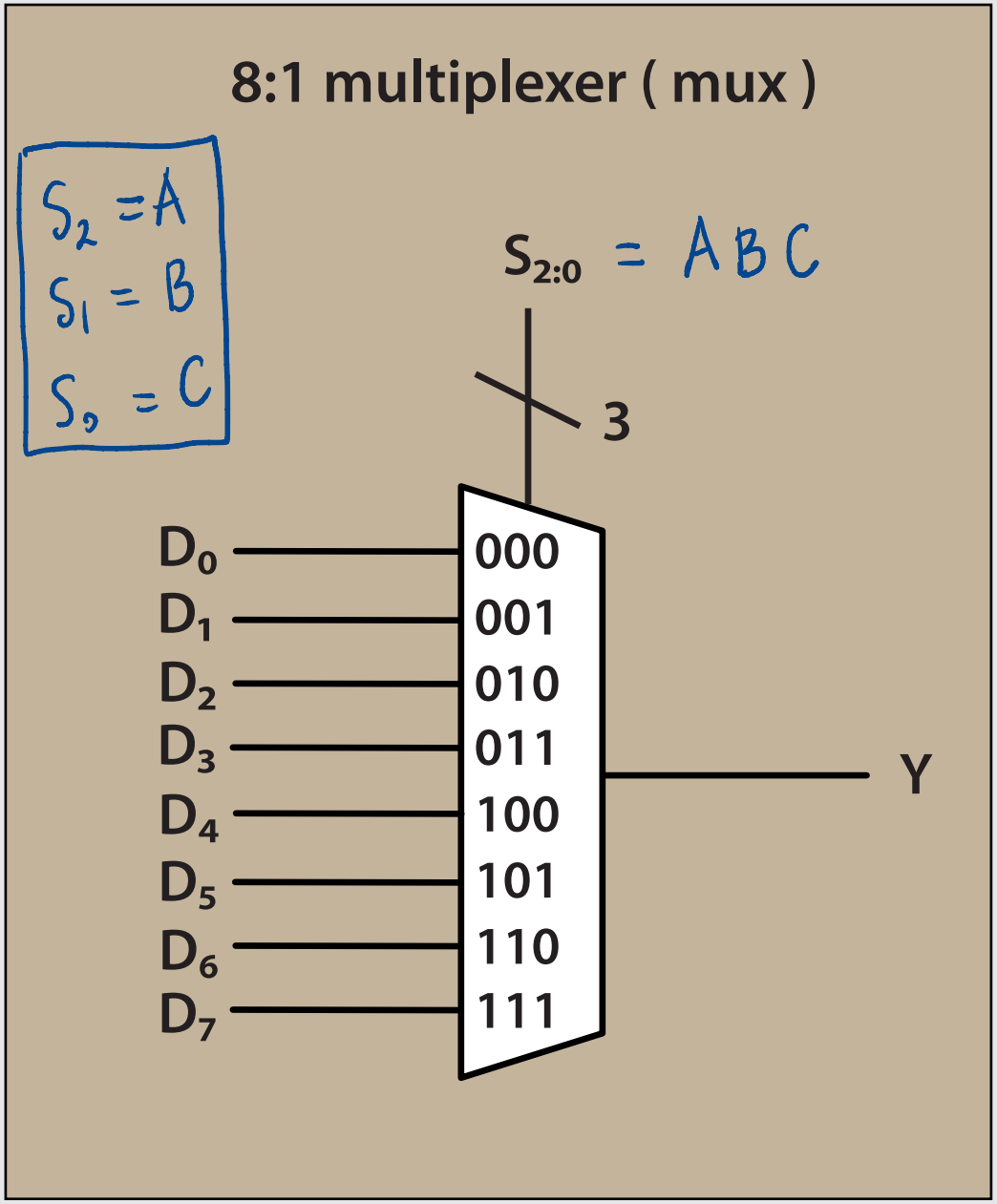
to finish her senior project, but when she looks in her lab kit, the only part she has left is an 8:1 multiplexer.

How does she implement the function?

# Solution

$$Y = A\overline{B} + \overline{B}\overline{C} + \overline{A}BC$$

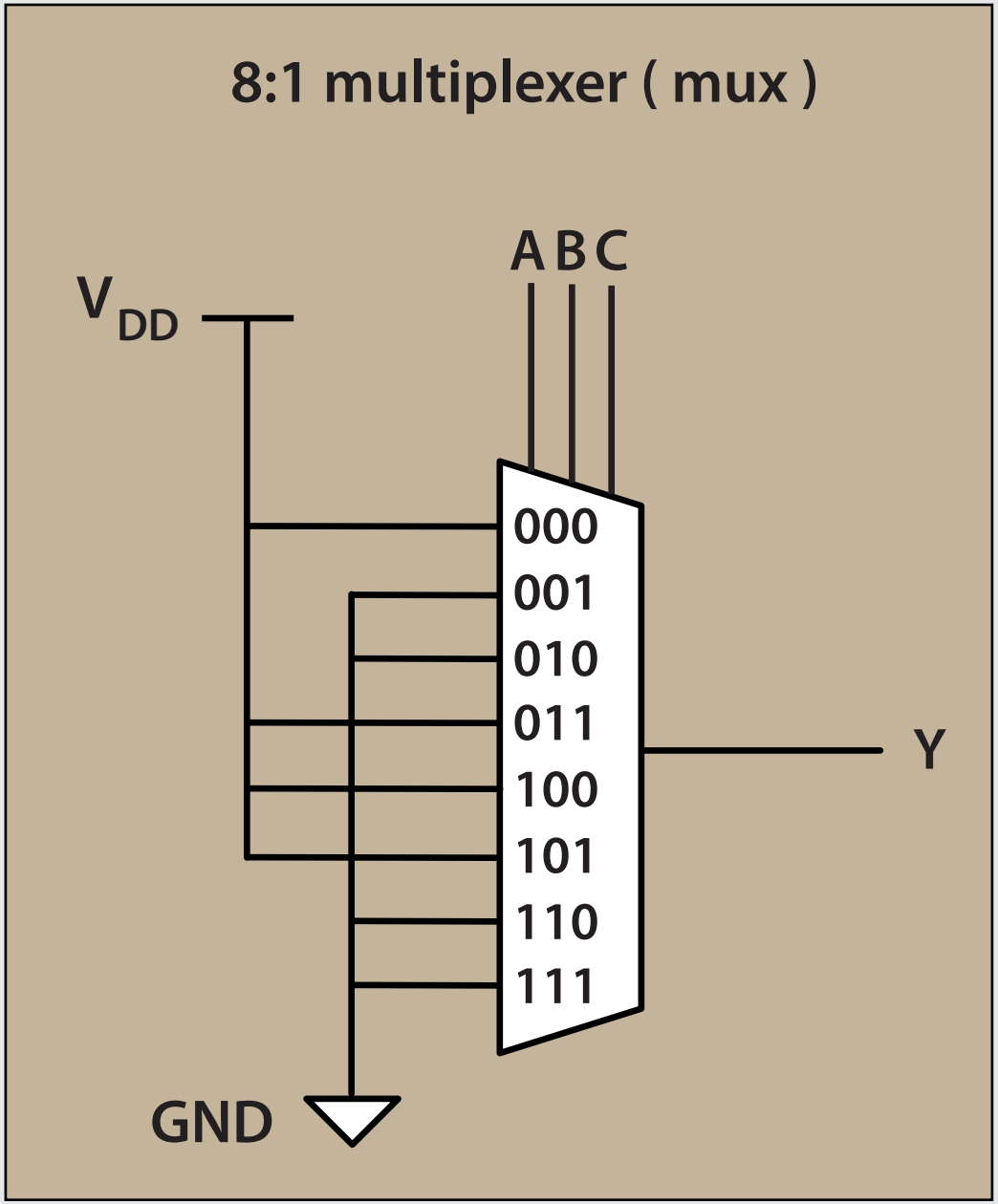
| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |



# Solution

$$Y = A\overline{B} + \overline{B}\overline{C} + \overline{A}BC$$

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

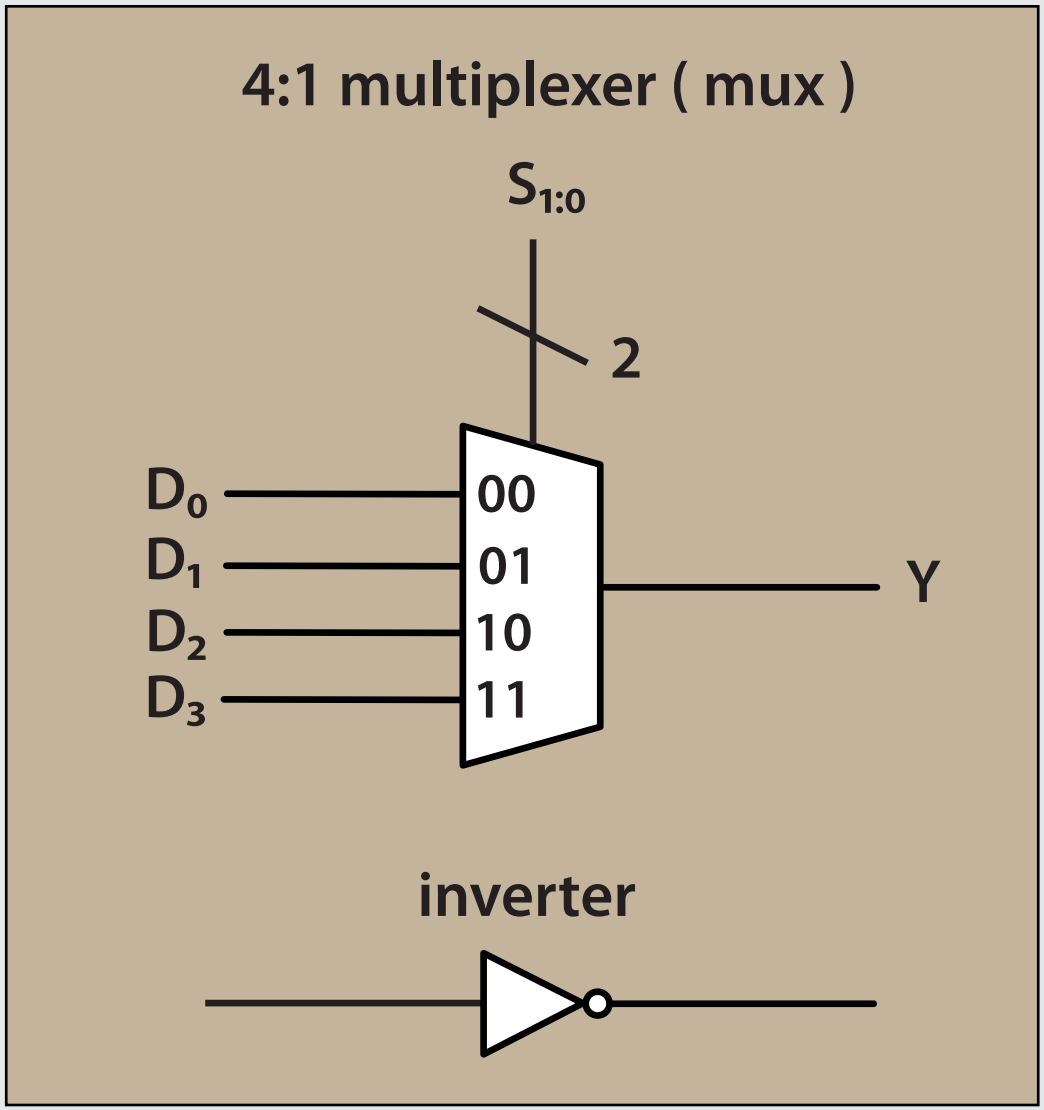


# Exercise

Can you construct the same boolean function with a 4:1 multiplexer and an inverter ?

$$Y = A\overline{B} + \overline{B}\overline{C} + \overline{A}BC$$

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

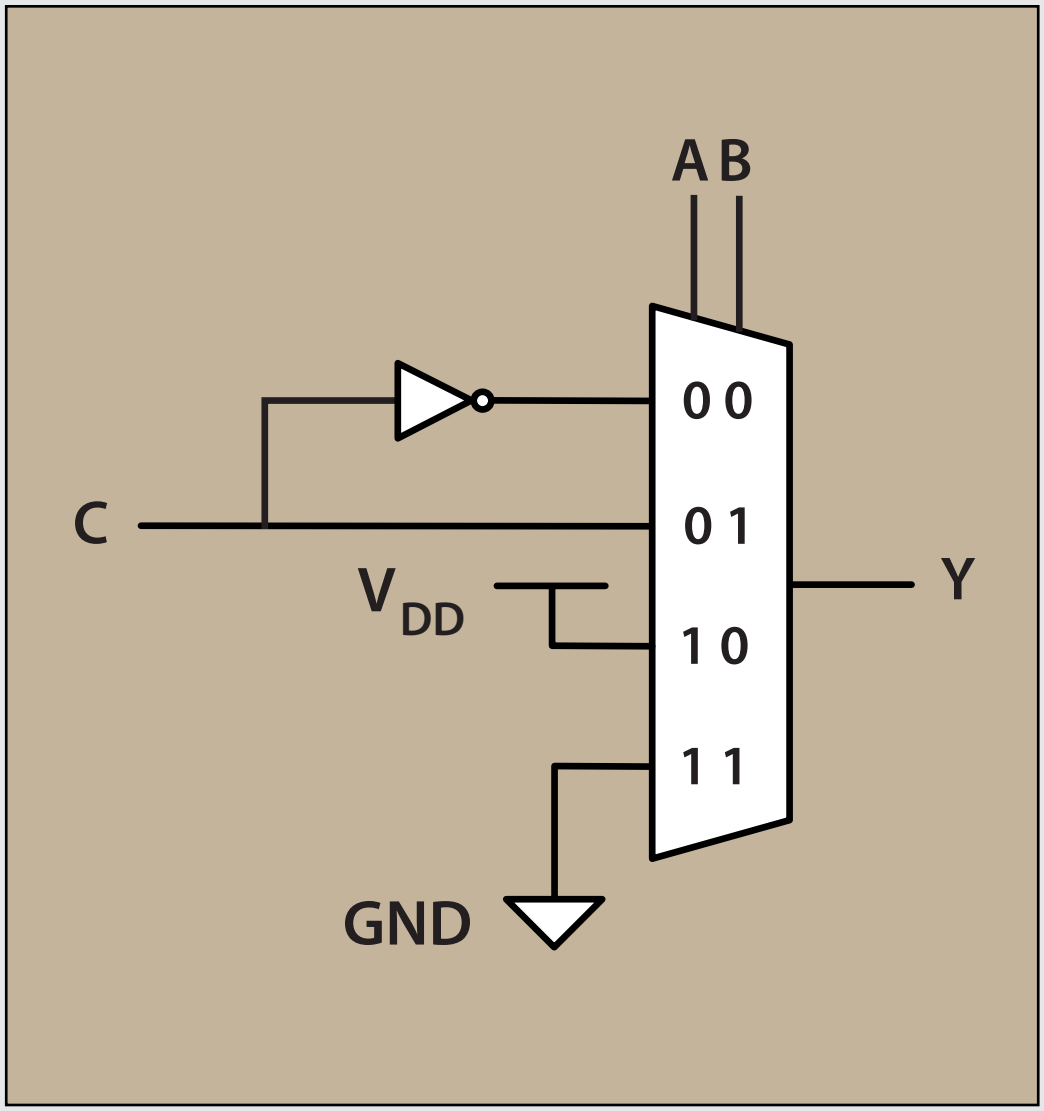


# Solution

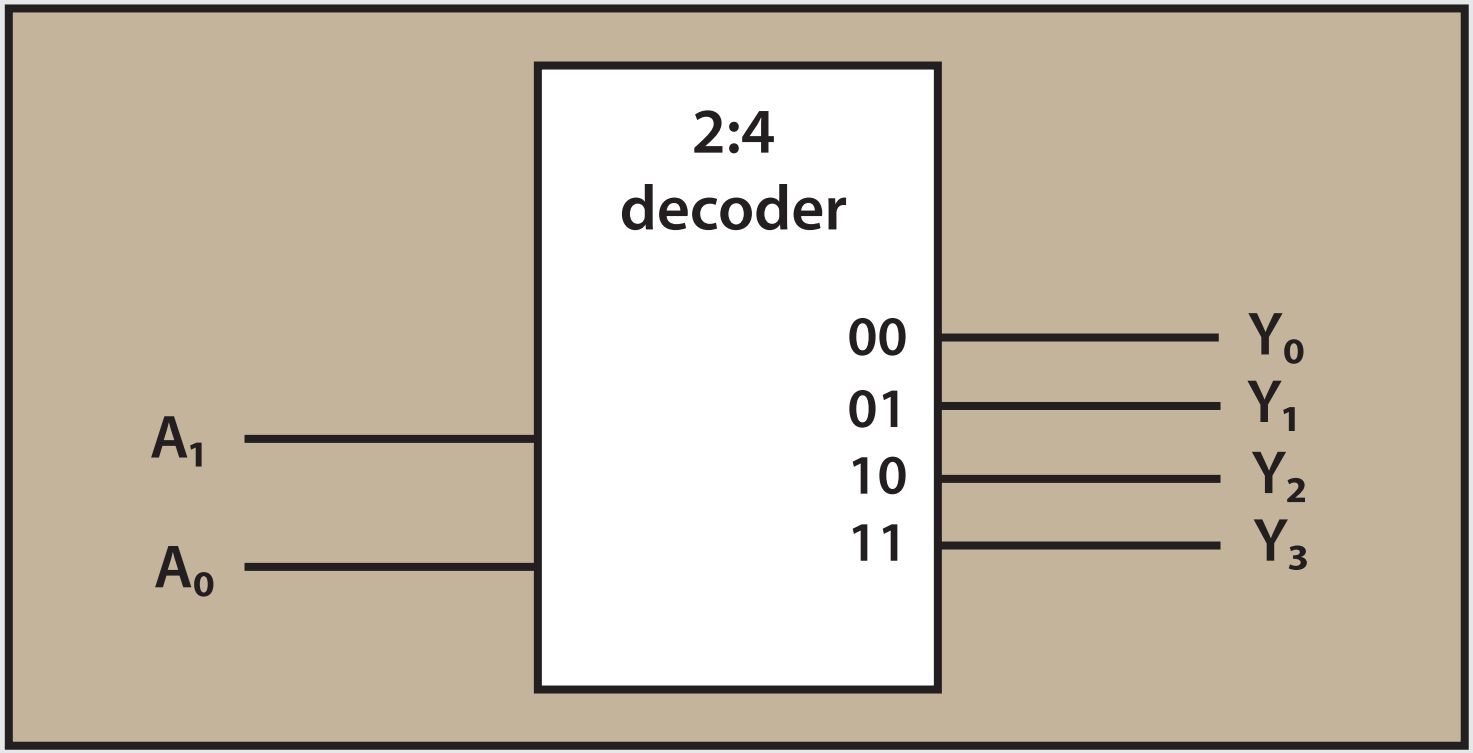
Can you construct the same boolean function with a 4:1 multiplexer and an inverter ?

$$Y = A\overline{B} + \overline{B}\overline{C} + \overline{A}BC$$

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

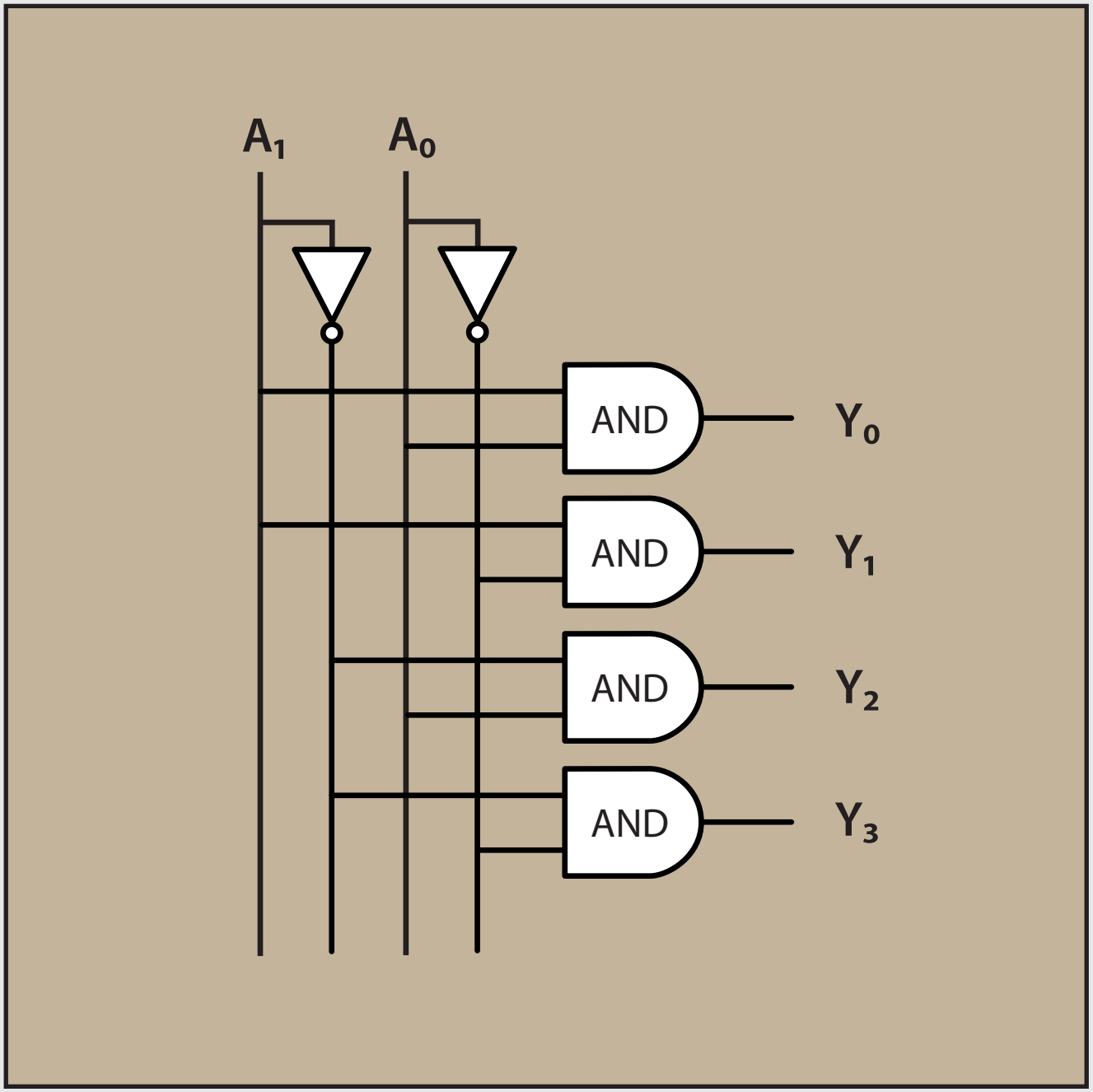


# Decoders



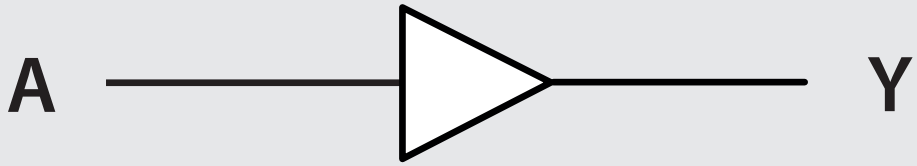

| $A_1$ | $A_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

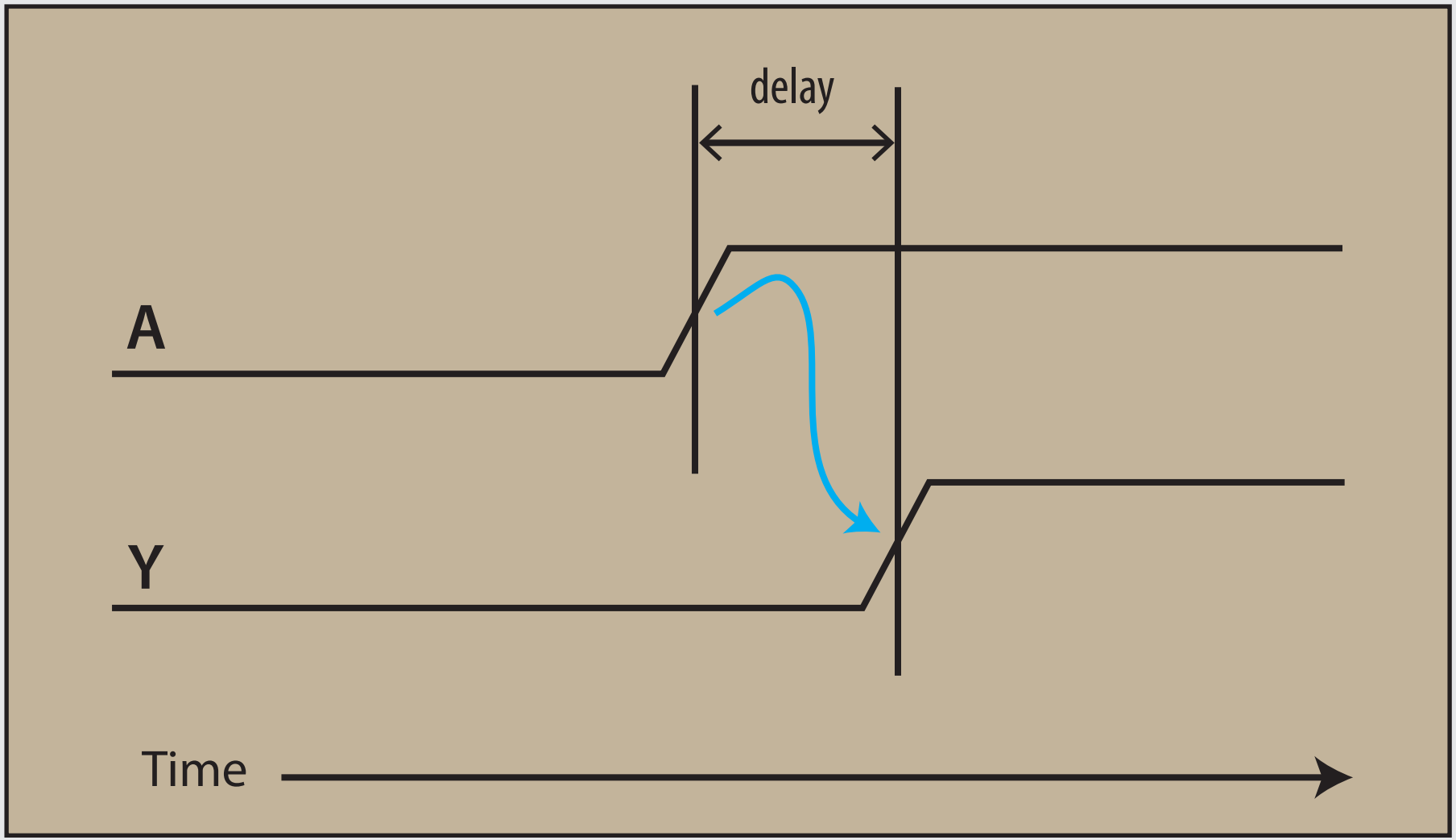
# Implementation of the 2:4 decoder



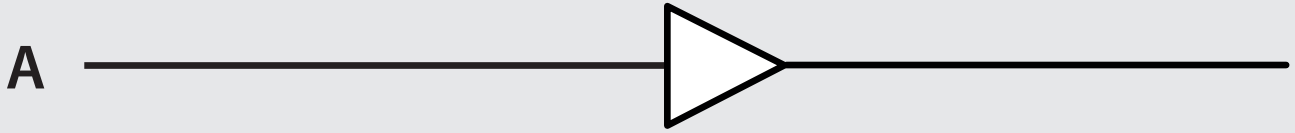
# Timing

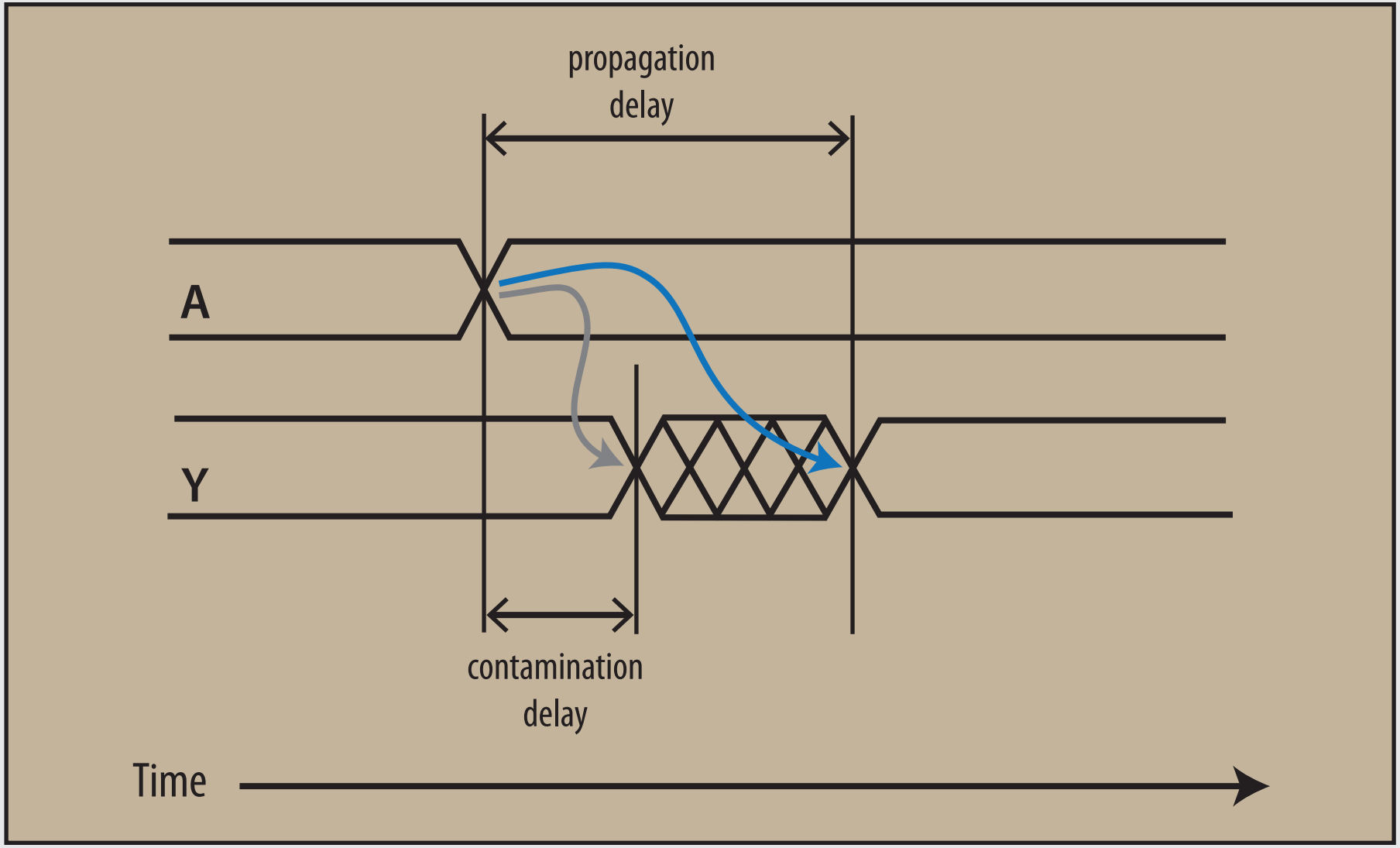
# Time delay



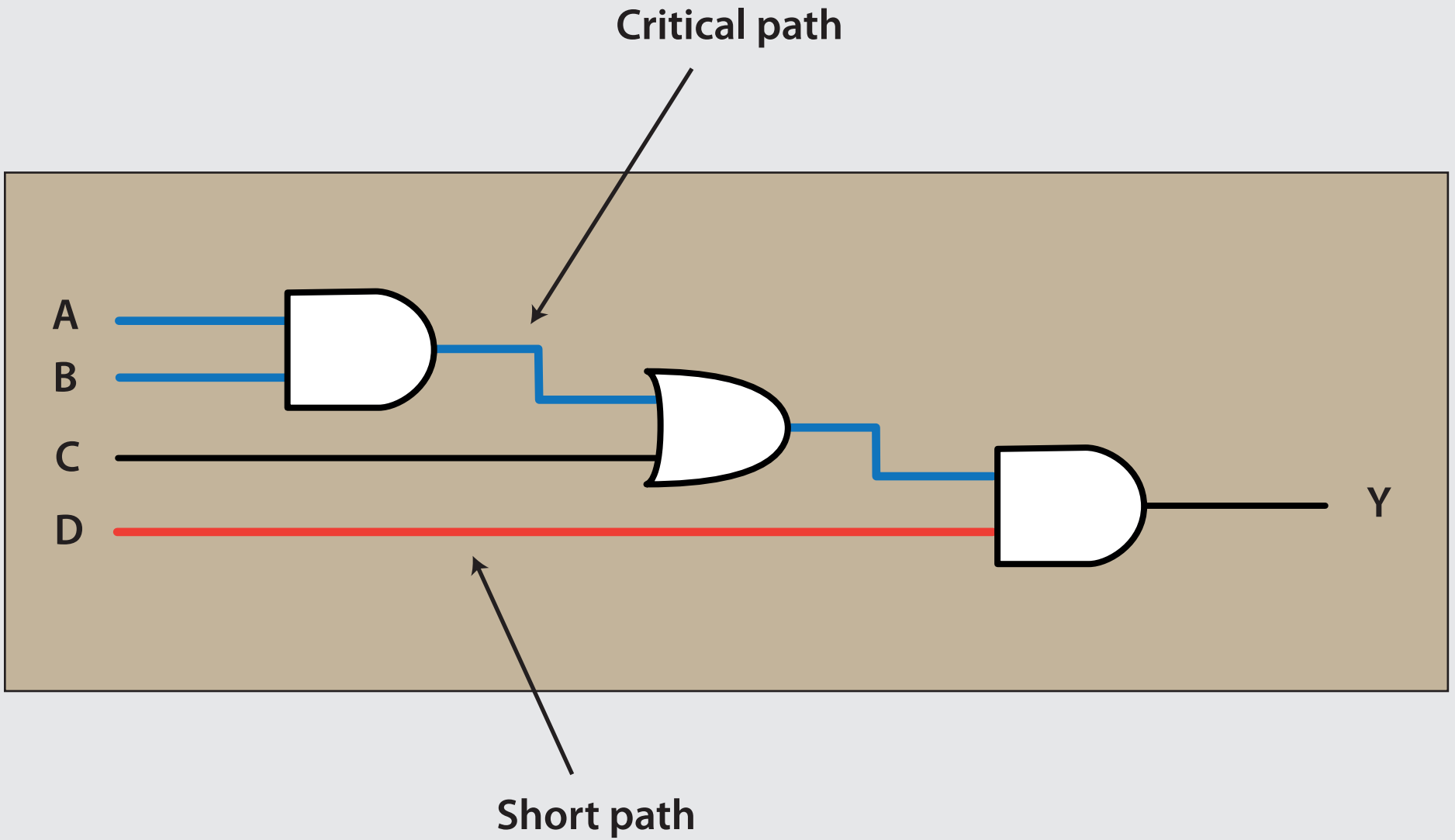



# Propagation and contamination delay

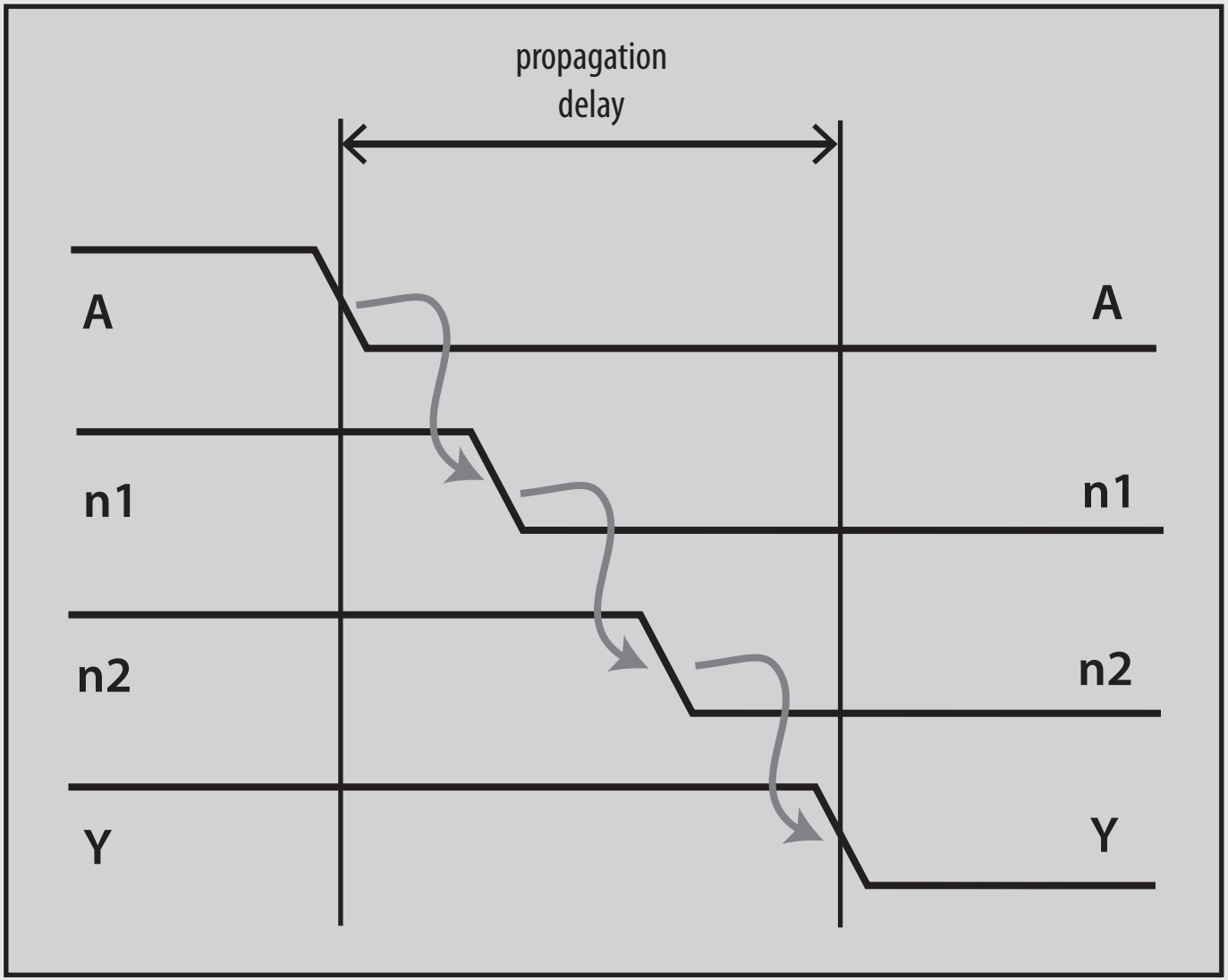


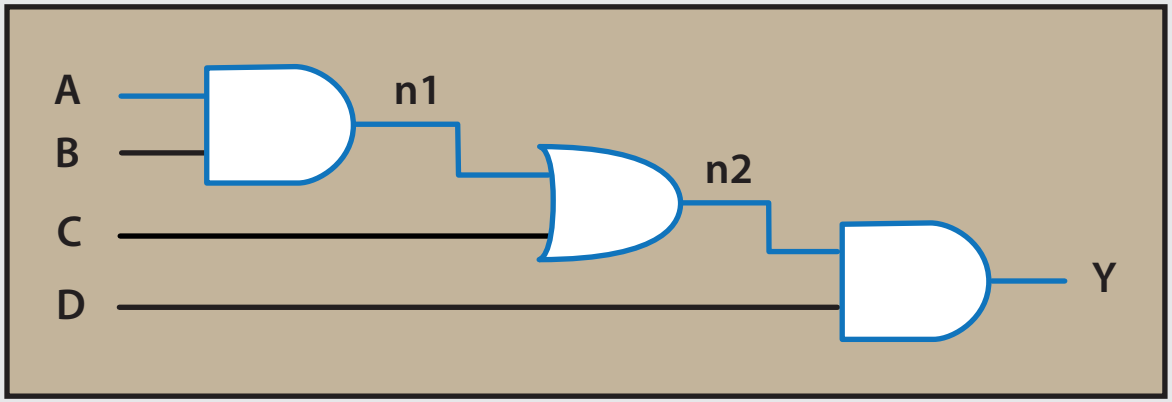



# Critical vs. short path



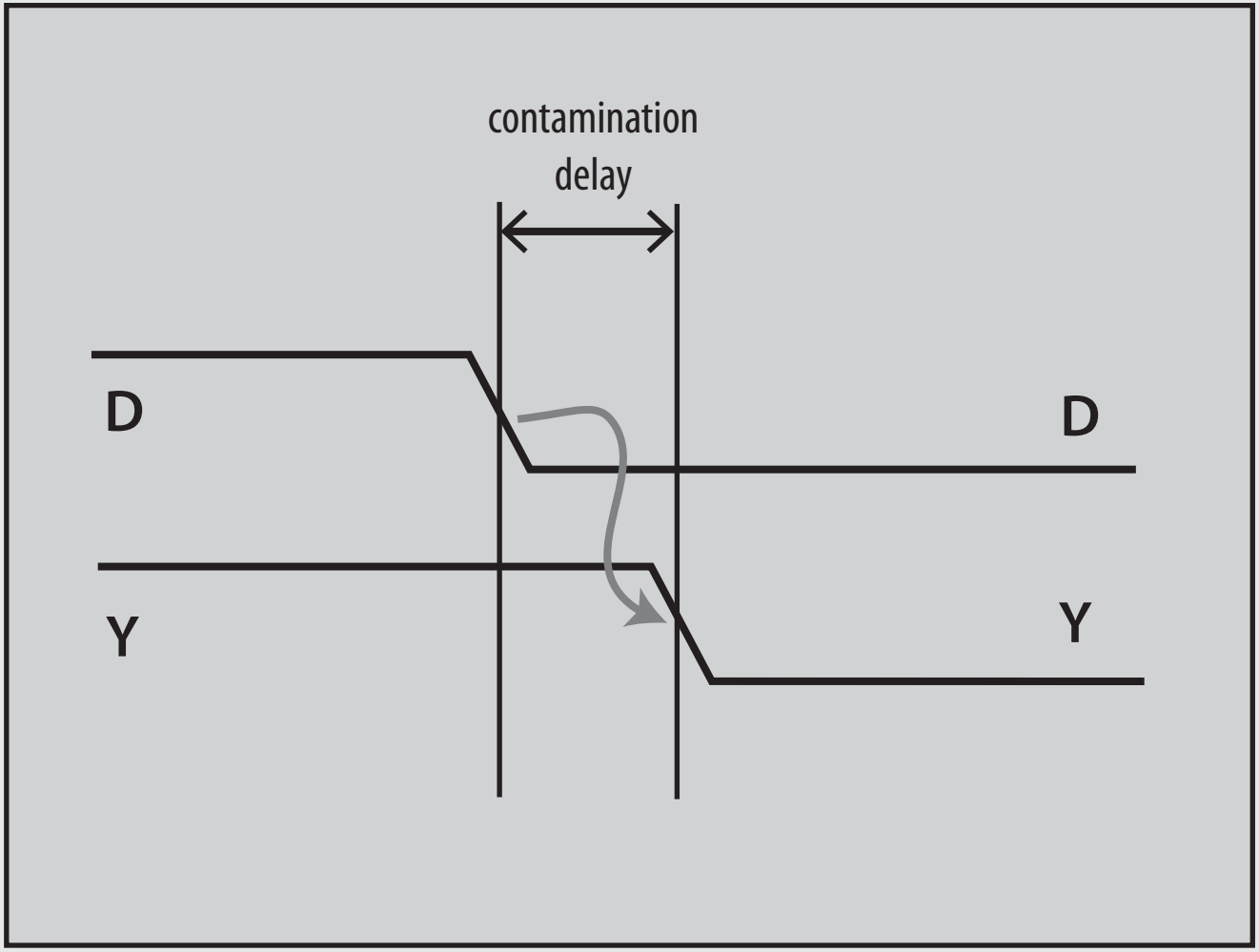
# Critical path waveform

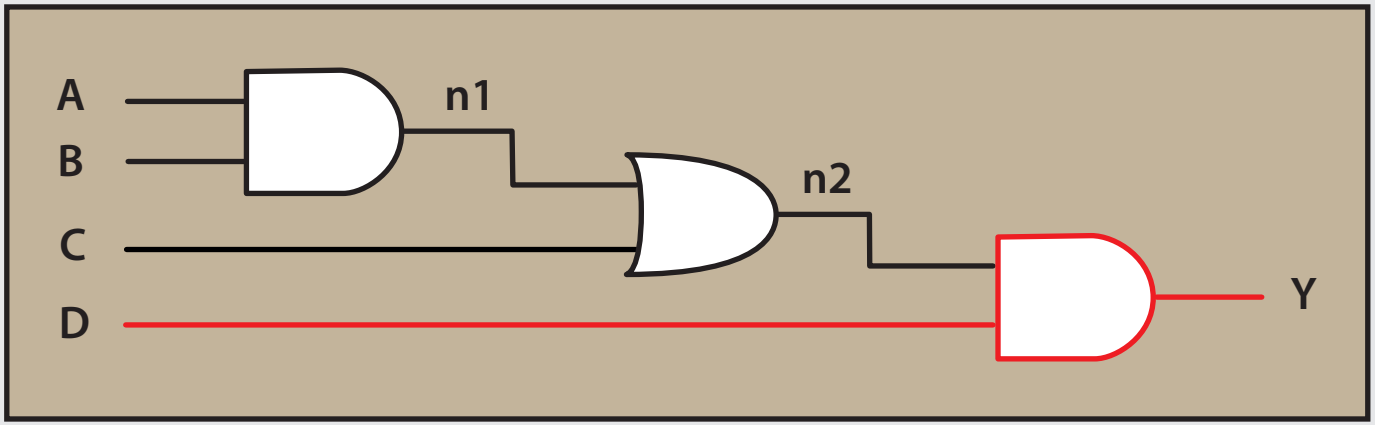



$$t_{pd} = 2\,t_{pd\_and} + t_{pd\_or}$$

# Short path waveform




$$t_{cd} = t_{cd\_and}$$

# Exercise 2

**H & H Example 2.16**

Given the propagation delays for the components given below,
compare the worst-case timing of the three four-input multiplexer designs.

What is the critical path for each design?

Given your timing analysis, why might you choose one design over the other ?

| Gate | $t_{pd}$ ( ps ) |
|---|---|
| NOT | 30 |
| 2-input AND | 60 |
| 3-input AND | 80 |
| 4-input OR | 90 |
| tristate ( A to Y ) | 50 |
| tristate ( enable to Y ) | 35 |