TECNOLOGÍA ELECTRÓNICA DE COMPUTADORES

2º Curso – GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

Tema 6. Circuitos electrónicos combinacionales. Puertas y bloques MSI combinacionales

Lección 9. Formas canónicas y tablas de verdad. Minimización de funciones lógicas

Lección 9. Formas canónicas y tablas de verdad. Minimización de funciones lógicas

9.1. Representación de funciones lógicas

- **Tabla de verdad**
- **Formas canónicas de una función lógica**

9.2. Simplificación de funciones lógicas: generalidades

9.3. Minimización de funciones lógicas: método del mapa de Karnaugh

Bibliografía de la lección

Lectura clave:

Thomas L.Floyd. Fundamentos de sistemas digitales.
Ed. Prentice Hall – Pearson Education.
Tema 4. Álgebra de Boole y simplificación lógica. Apartados 4.1. a 4.11.

9.1. Representación de funciones lógicas

Posibles representaciones de una función lógica:

- **Expresión lógica** (combinación de variables relacionadas por las tres operaciones básicas): Infinitas posibilidades. De ellas, destacar:
  - 1ª Forma canónica: Suma de productos de las variables de la función.
  - 2ª Forma canónica: Producto de sumas de las variables de la función.

- **Expresión gráfica** (esquema con puertas lógicas): Infinitas

- **Tabla de verdad** (se construye colocando tantas columnas como variables de entrada, disponiendo en las columnas todas las combinaciones posibles y obteniendo el resultado final): ÚNICA

Ejercicio:

Comprobar que la tabla de verdad es la misma para todas estas expresiones

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

1ª Forma canónica (suma de productos)

¿ Cuándo vale "1" la función ?

A	B	C	f(A,B,C)		Decimal
0	0	0	1	☑	0
0	0	1	0	☐	1
0	1	0	1	☑	2
0	1	1	0	☐	3
1	0	0	1	☑	4
1	0	1	1	☑	5
1	1	0	0	☐	6
1	1	1	1	☑	7

Este término vale "1" si A=0 y B=0 y C=0

$\overline{A} \cdot \overline{B} \cdot \overline{C} = 1$

La función vale "1" si el término 0 ó el 2 ó el 4 ó el 5 ó el término 7 valen "1"

$f(A,B,C) = m_0+m_2+m_4+m_5+m_7$

(m_i = miniterm)

$$f(A,B,C)= \overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot B\cdot\overline{C} + A\cdot\overline{B}\cdot\overline{C} + A\cdot\overline{B}\cdot C + A\cdot B\cdot C$$

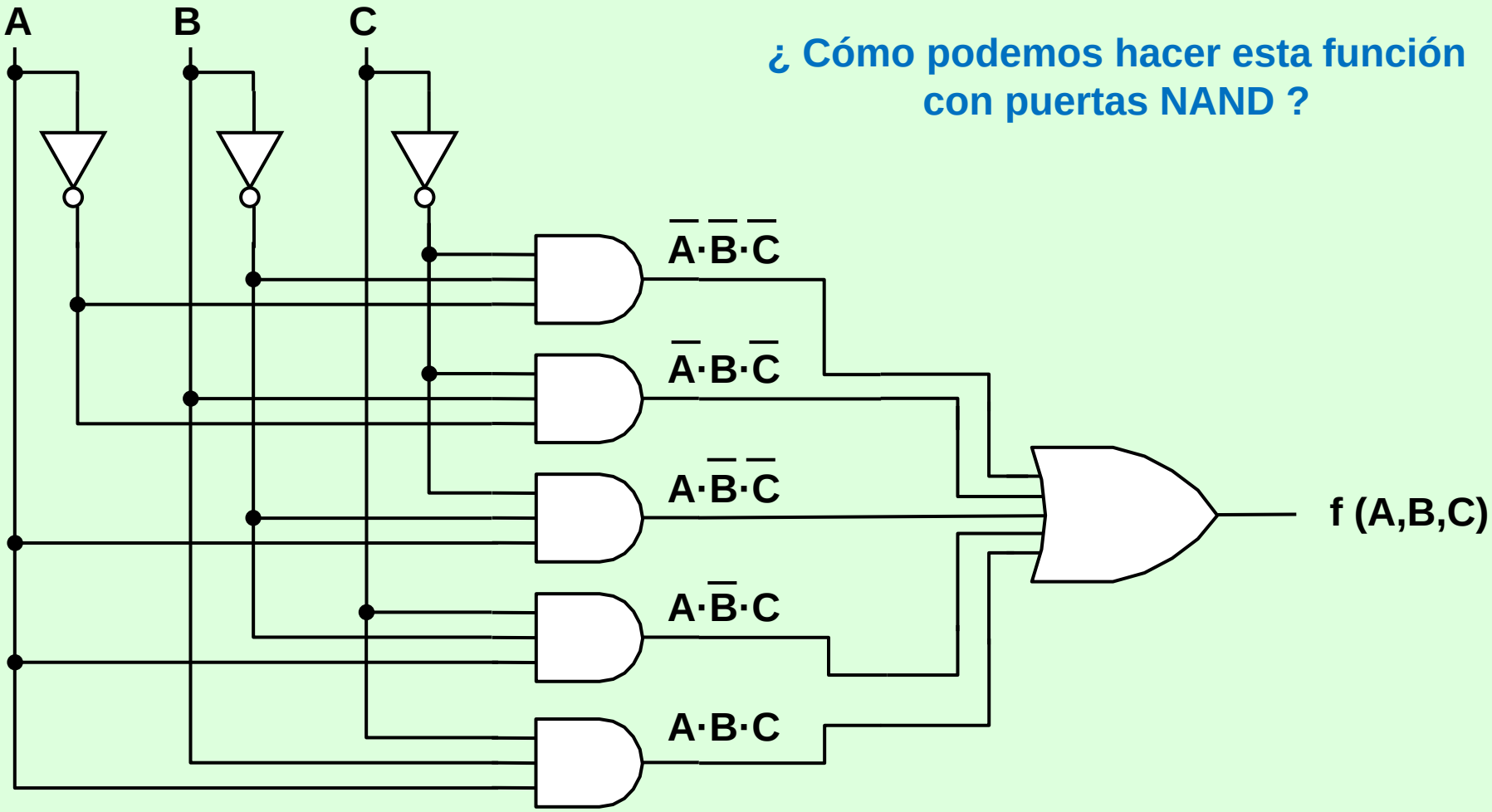
A	B	C	f(A,B,C)	Producto=1
0	0	0	1	$\overline{A}\cdot\overline{B}\cdot\overline{C}$
0	0	1	0	
0	1	0	1	$\overline{A}\cdot B\cdot\overline{C}$
0	1	1	0	
1	0	0	1	$A\cdot\overline{B}\cdot\overline{C}$
1	0	1	1	$A\cdot\overline{B}\cdot C$
1	1	0	0	
1	1	1	1	$A\cdot B\cdot C$

Cada 1 de la tabla de verdad da lugar a un término (un producto)

El estado de la variable depende de si vale 1 o 0 la combinación de la tabla de verdad:

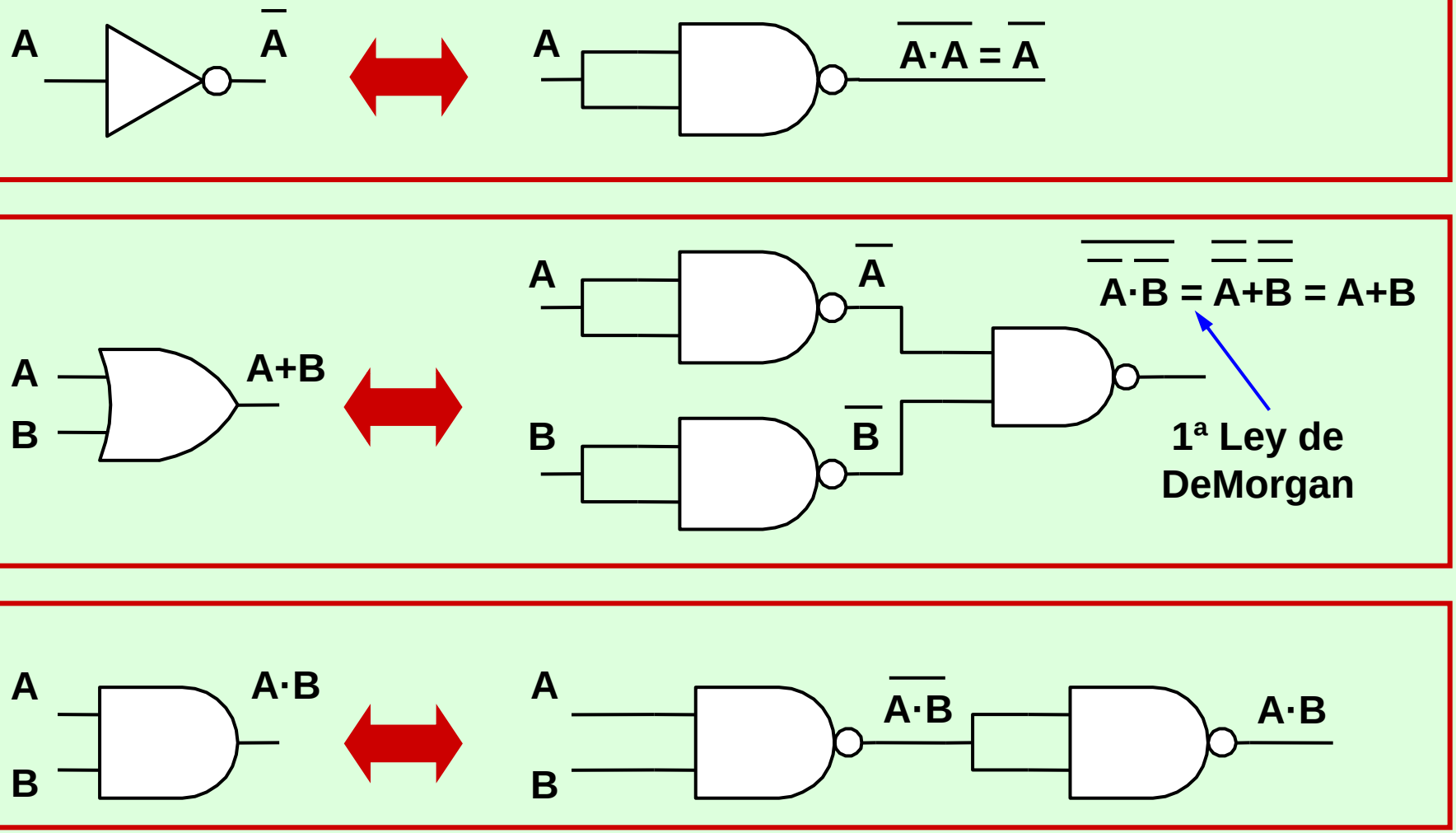
1= Variable normal
0= Variable negada

1.- No es la forma más simple de expresar la función.
2.- El orden (A,B,C) ó (C,B,A) debe tenerse en cuenta: usual C mayor peso
3.- Permite el paso a puertas de forma automática.



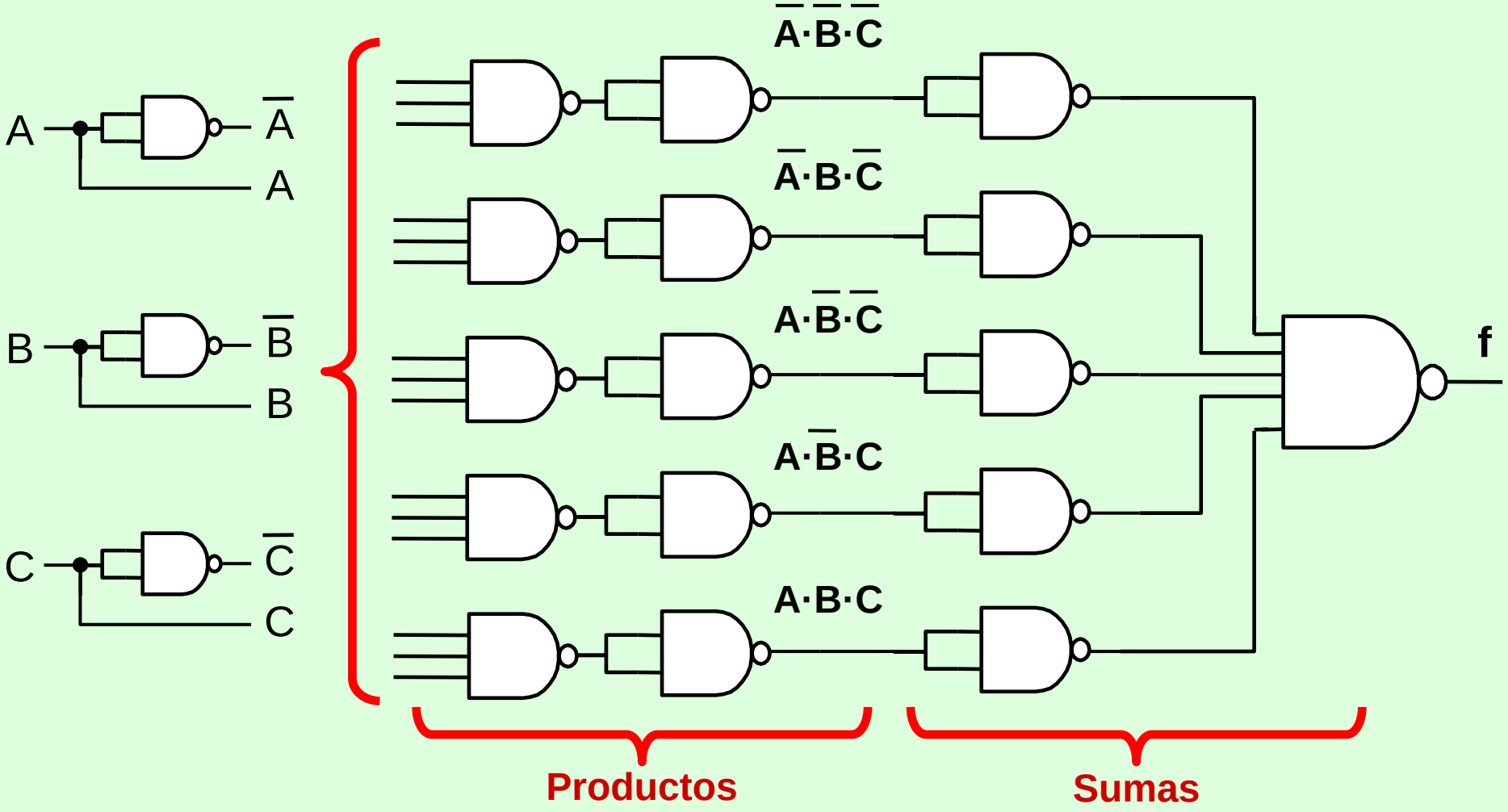
$$f(A,B,C)= \overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot B\cdot\overline{C} + A\cdot\overline{B}\cdot\overline{C} + A\cdot\overline{B}\cdot C + A\cdot B\cdot C$$

¡¡ Todas las puertas básicas se pueden hacer SÓLO con puertas NAND !!

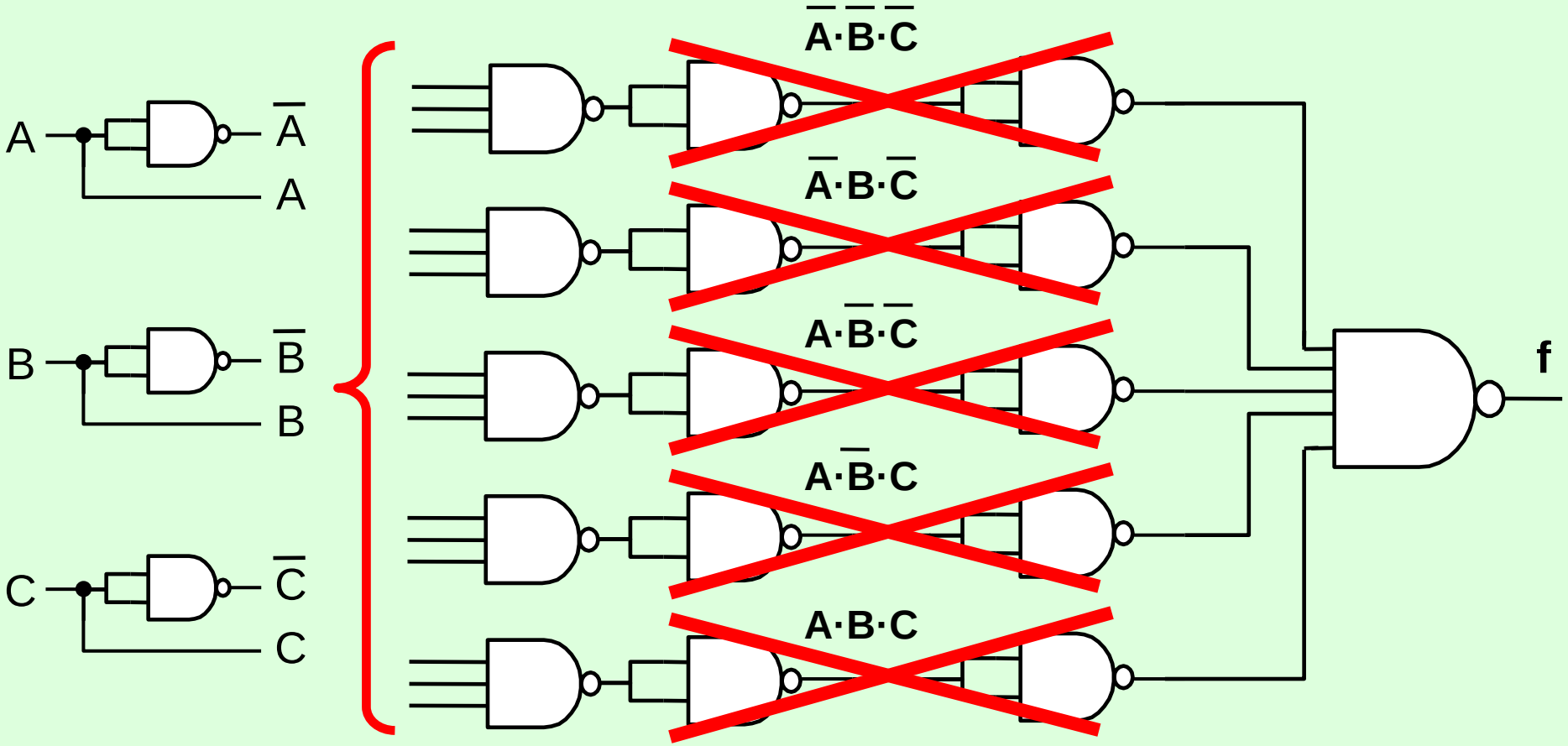


$$f(A,B,C)= \overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot B\cdot\overline{C} + A\cdot\overline{B}\cdot\overline{C} + A\cdot\overline{B}\cdot C + A\cdot B\cdot C$$

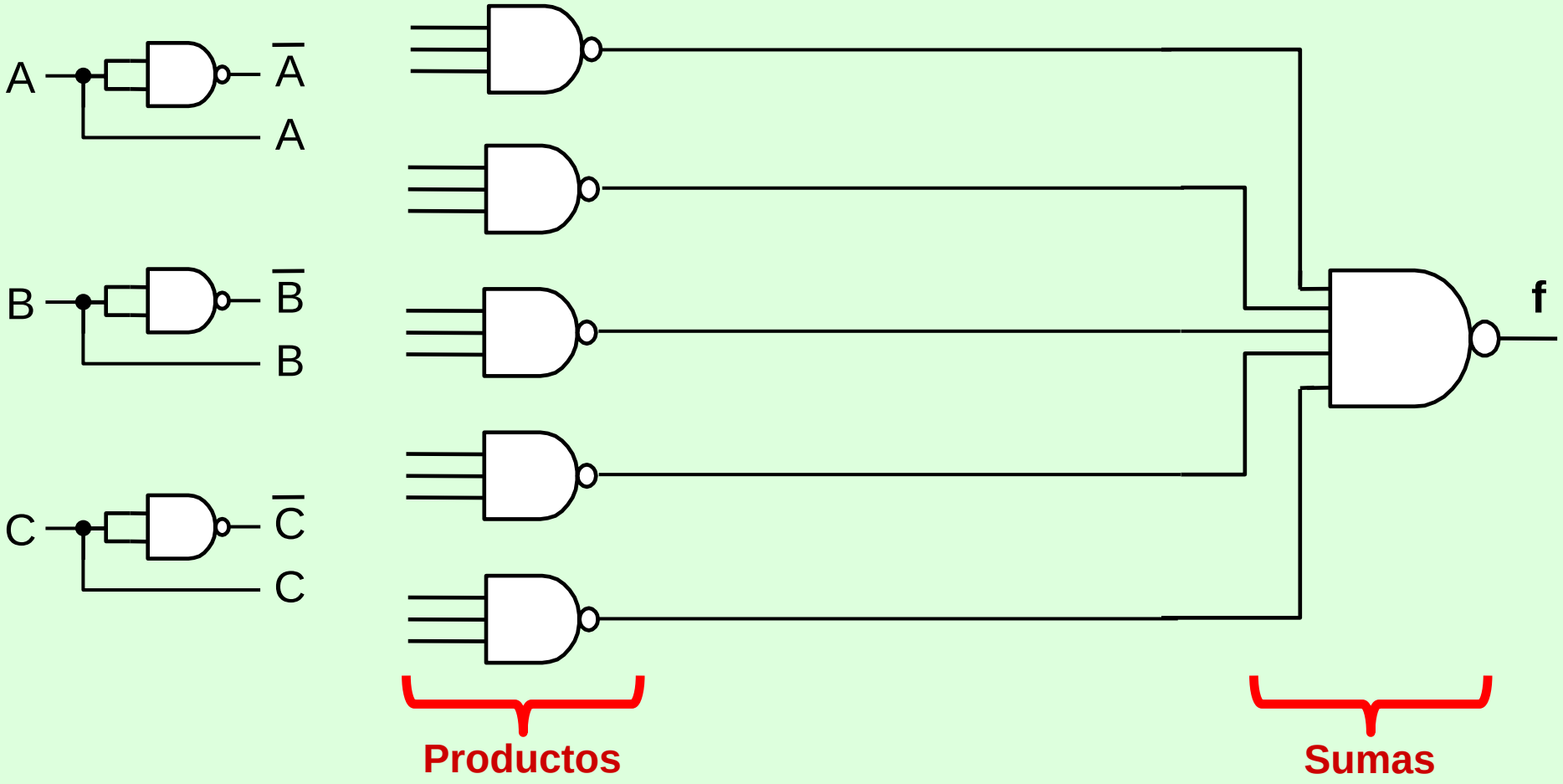
Al sustituir en el circuito:



¡¡ Las puertas intermedias se eliminan !!



¡¡ Las puertas intermedias se eliminan !!



Ejercicio:

a) Obtener la primera forma canónica de la función bit de paridad par del código BCD

b) Realizar la función con puertas NAND

Pasos:

1º) Tabla de verdad

2º) Primera forma canónica

3º) Implementar con puertas NAND

D	C	B	A	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0

NOTA: Observar que podemos prescindir de los valores fuera del código BCD, ya que no nos importa el valor que proporcione la función

Términos indiferentes: Son aquellas combinaciones de las variables de la función que no se van a presentar nunca como entrada a la misma “. Por tanto, se les puede dar el valor “0” o el “1” en la función.

Ejemplo:

Todas las combinaciones a partir de 1001 no forman parte del código BCD; sin embargo, son combinaciones de las variables DCBA que pueden resultar útiles en la simplificación posterior

D	C	B	A	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

2ª Forma canónica (producto de sumas)

¿ Cuándo vale “0” la función ?

A	B	C	f(A,B,C)		Decimal
0	0	0	1	☐	0
0	0	1	0	☑	1
0	1	0	1	☐	2
0	1	1	0	☑	3
1	0	0	1	☐	4
1	0	1	1	☐	5
1	1	0	0	☑	6
1	1	1	1	☐	7

Este término vale “0” si A=0 y B=0 y C=1

$A + B + \overline{C} = 0$

La función vale “0” si el término 1 y el 3 y el 6 valen “0”

$f(A,B,C) = M_1 \cdot M_3 \cdot M_6$

M_i=maxiterm

$$f(A,B,C)= (A+B+\overline{C})\cdot(A+\overline{B}+\overline{C})\cdot(\overline{A}+\overline{B}+C) = M_1\cdot M_3\cdot M_6$$

A	B	C	f(A,B,C)	Producto=1
0	0	0	1	
0	0	1	0	$A+B+\overline{C}$
0	1	0	1	
0	1	1	0	$A+\overline{B}+\overline{C}$
1	0	0	1	
1	0	1	1	
1	1	0	0	$\overline{A}+\overline{B}+C$
1	1	1	1	

Cada 0 de la tabla de verdad da lugar a un término (una suma)

El estado de la variable depende de si vale 1 o 0 la combinación de la tabla de verdad:

0= Variable normal
1= Variable negada

¡ ATENCIÓN !

El criterio cambia con respecto a los miniterms:

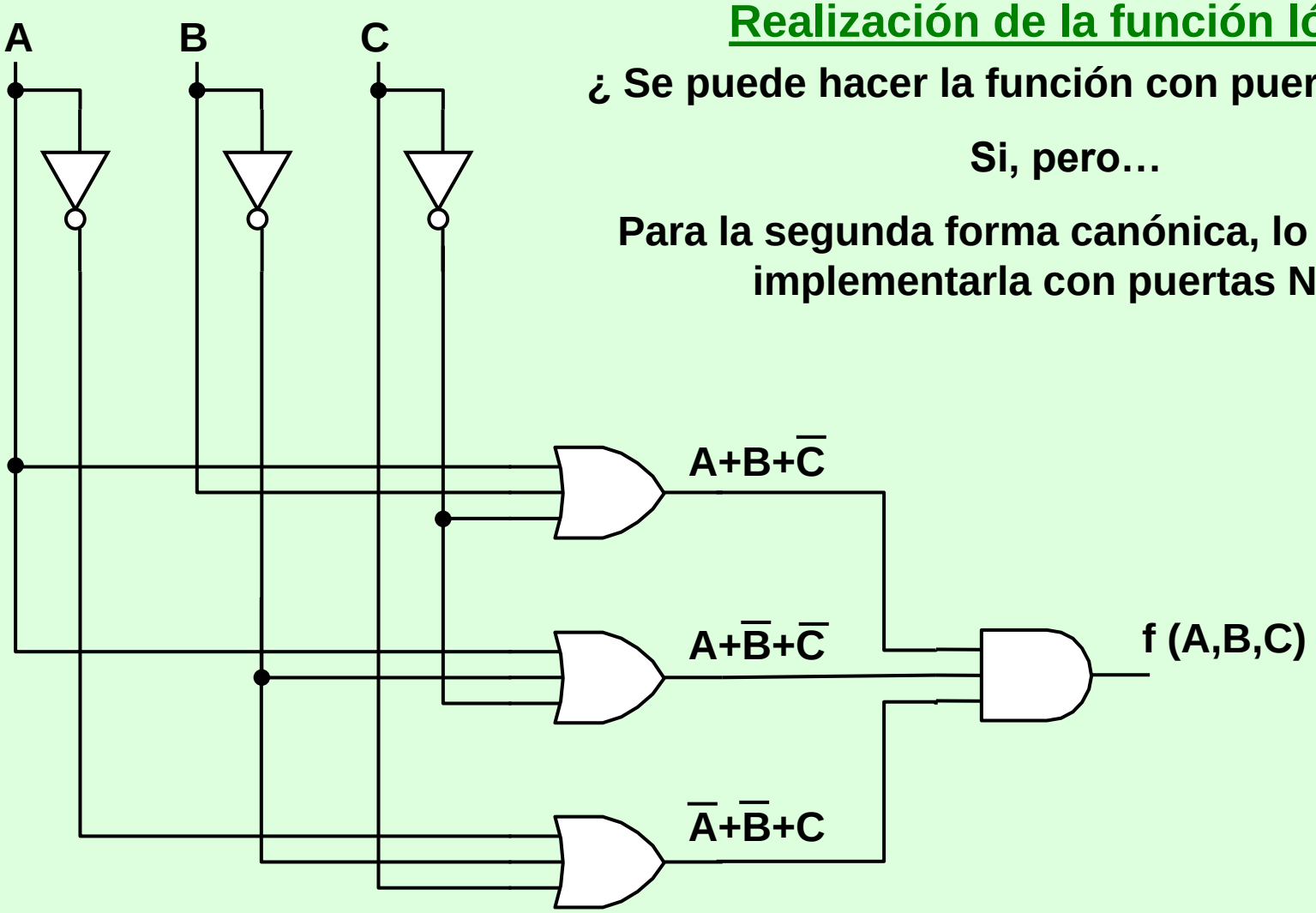
VARIABLE = 0 APARECE SIN COMPLEMENTAR (ESTADO NATURAL)

Realización de la función lógica

¿ Se puede hacer la función con puertas NAND ?

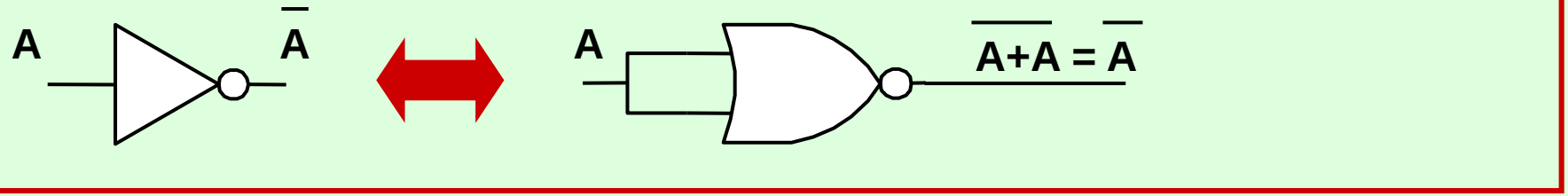
Si, pero…

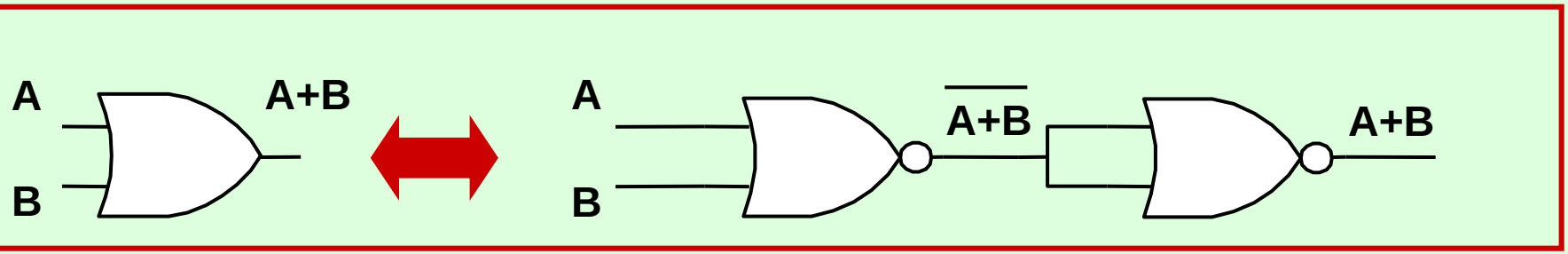
Para la segunda forma canónica, lo óptimo es implementarla con puertas NOR

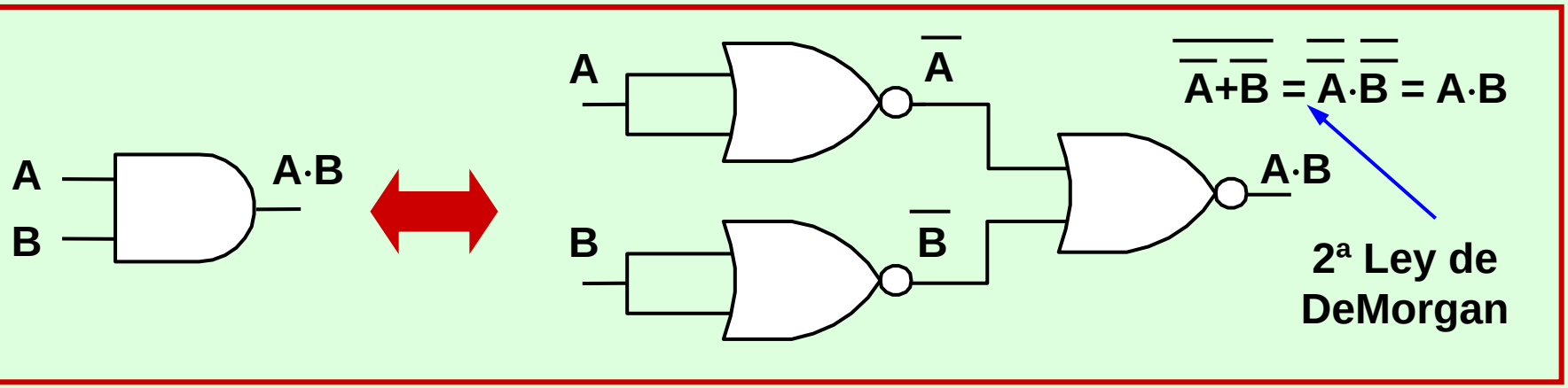


$$f(A,B,C)= (A+B+\overline{C})\cdot(A+\overline{B}+\overline{C})\cdot(\overline{A}+\overline{B}+C)$$

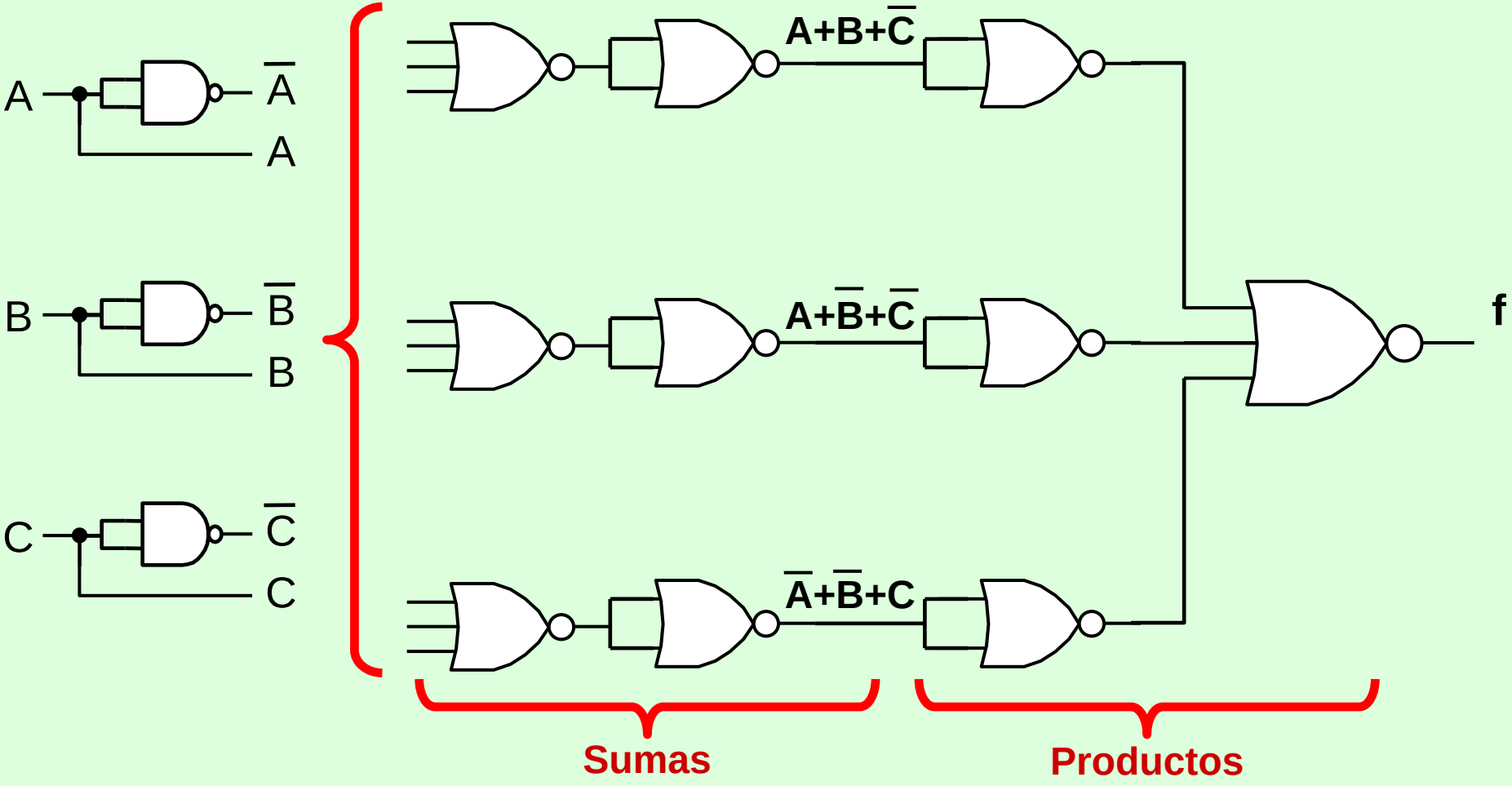
¡¡¡ Todas las puertas básicas se pueden hacer SÓLO con puertas NOR !!!



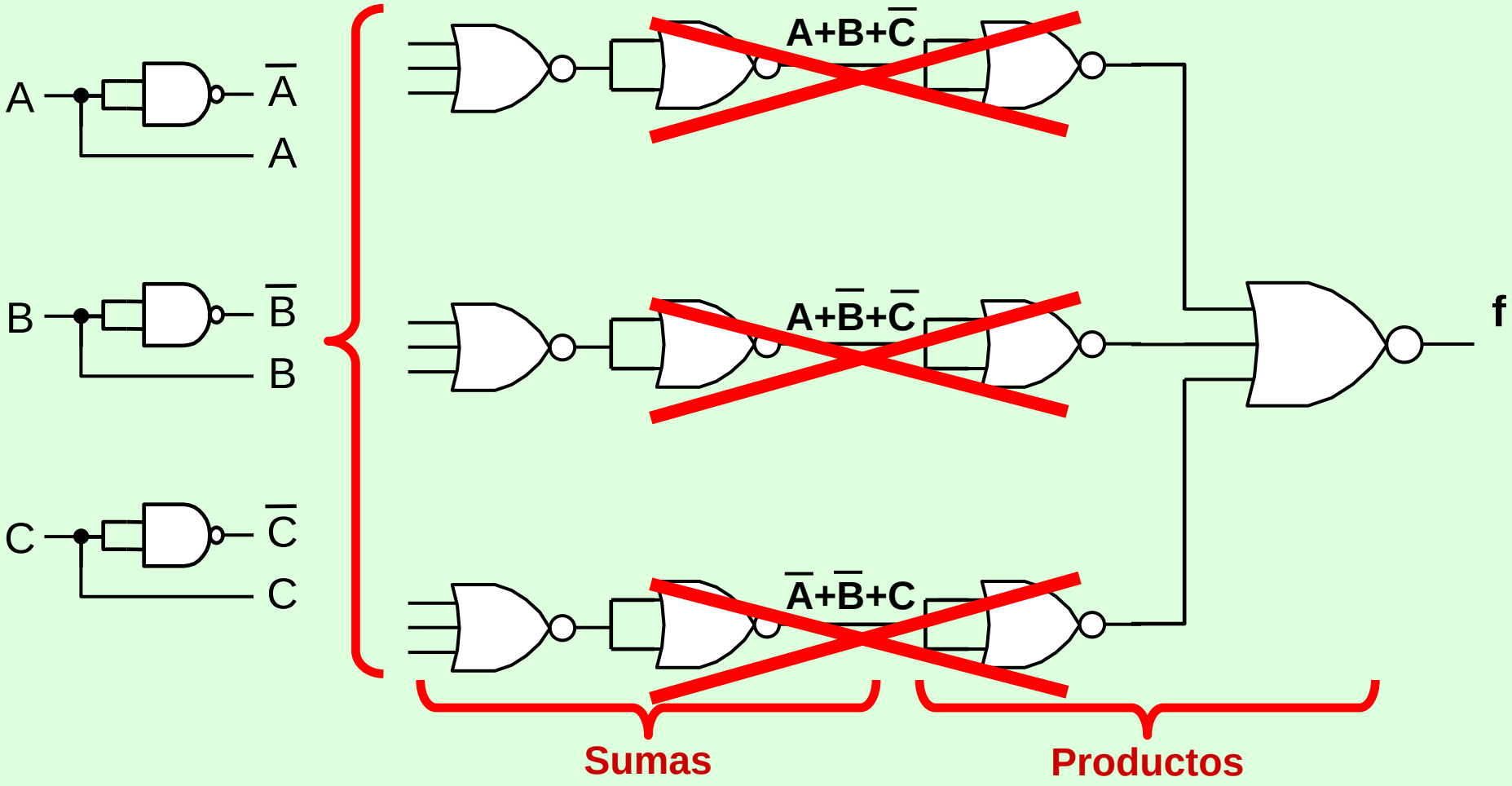




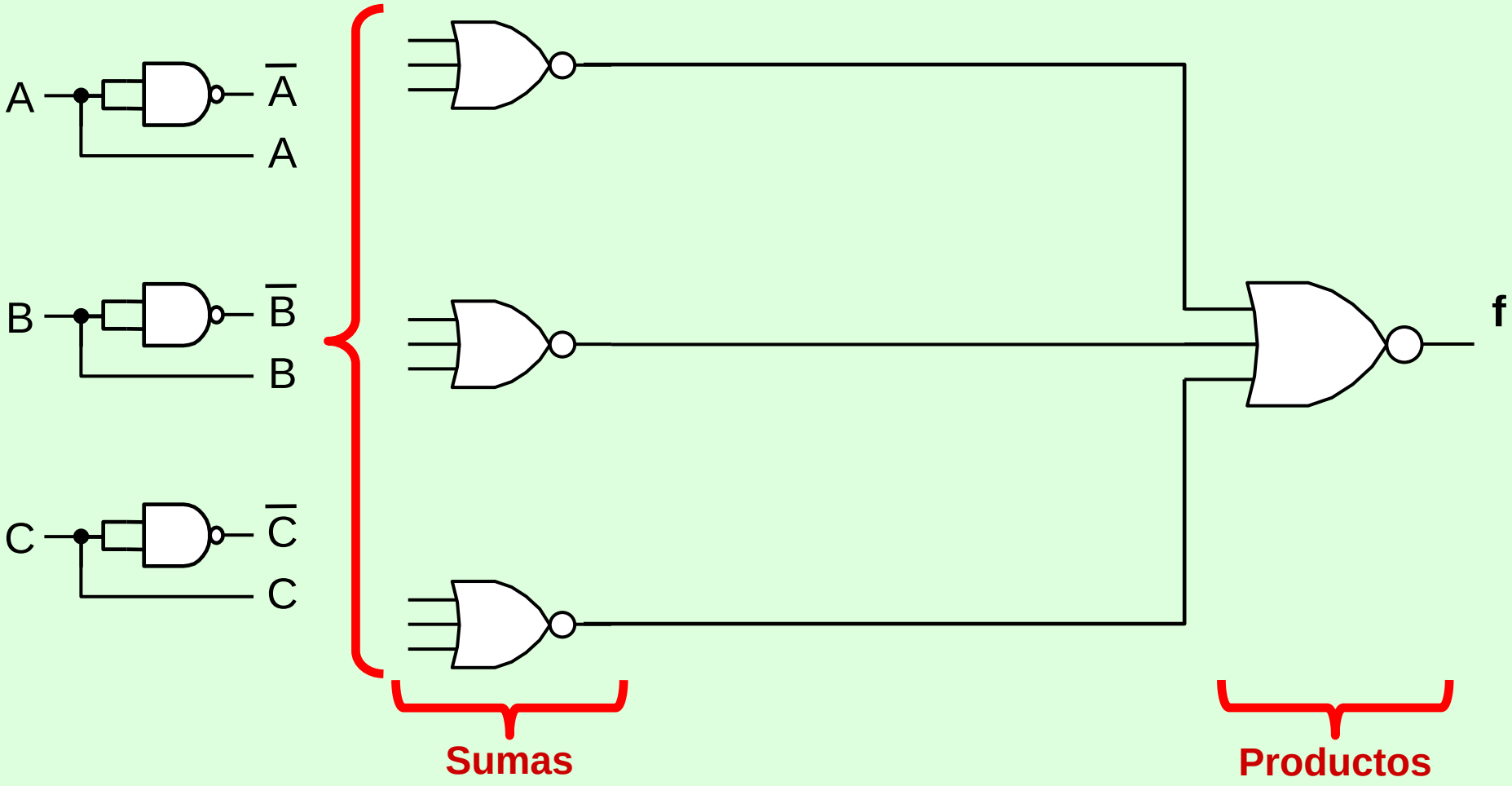
Al sustituir en el circuito:



¡¡ Las puertas intermedias se eliminan !!



¡¡ Las puertas intermedias se eliminan !!



Conversión entre formas canónicas

$$f(A,B,C)= A\cdot B\cdot C + \overline{A}\cdot B\cdot \overline{C} + A\cdot \overline{B}\cdot \overline{C} + A\cdot \overline{B}\cdot C + A\cdot B\cdot C =$$

$$=(A + B + \overline{C})\cdot(A + \overline{B} + \overline{C})\cdot(\overline{A} + \overline{B} + C) =$$

$$= m_0 + m_2 + m_4 + m_5 + m_7 = M_1\cdot M_3\cdot M_6$$

1ª opción:

Obtener la tabla de verdad y proceder según los pasos anteriores

2ª opción:

Conversión directa; los términos que faltan en la primera (segunda) forma son los que componen la segunda (primera)

En el ejemplo: Los términos 1, 3 y 6 son los que faltan en la primera forma canónica y los que componen la segunda forma canónica

Conclusiones sobre las formas canónicas:

1. Son formas estándar, de fácil obtención.

2. Es sencillo elegir la más simple.

3. Son estructuras regulares, aptas para la implementación de algoritmos.

4. Utilizan un solo tipo de puertas. Cualquier función lógica puede realizarse a partir de puertas NAND solamente o de puertas NOR sólamente

5. NO SON LA FORMA MAS SIMPLE DE EXPRESAR UNA FUNCIÓN.

Por ello, nos plantearemos:

¿CÚAL ES LA FORMA MÁS SIMPLIFICADA DE UNA FUNCIÓN LÓGICA?

9.2. Simplificación de funciones lógicas: generalidades

Simplificación de funciones lógicas: posibilidades

1. Métodos algebraicos: Aplicación de teoremas, etc..

Ejemplo:

$$f(A,B,C)= \overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot B\cdot\overline{C} + A\cdot\overline{B}\cdot\overline{C} + A\cdot\overline{B}\cdot C + A\cdot B\cdot C =$$

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

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

Esta expresión es la más simple, pero no tiene porque ser única:

$f(A,B,C) = \overline{A}\cdot\overline{C} + \overline{C}\cdot\overline{B} + A\cdot C$ es también una expresión válida

9.3. Método del mapa de Karnaugh

Simplificación de funciones lógicas: posibilidades

2. Métodos gráficos: EL MAPA DE KARNOUGH

Resulta efectivo cuando el número de variables de la función es inferior o igual a cuatro, se complica para cinco y seis variables y no se emplea para más de seis. Resulta útil para funciones múltiples.

Se basa en los términos adyacentes lógicos: "Son aquellos términos de una función que sólo se diferencian en el ESTADO DE UNA VARIABLE"

En el mapa de Karnaugh los términos adyacentes físicamente también lo son desde el punto de vista lógico, lo que permite visualizarlos.

Los términos adyacentes lógicos se pueden simplificar:

LA VARIABLE QUE CAMBIA DE ESTADO DESAPARECE

Ejemplo: $A \cdot B \cdot C$ y $A \cdot B \cdot \overline{C}$

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

Pasos a realizar (para la primera forma canónica):

1º Partimos de la tabla de verdad.

2º Formamos dos grupos con las variables de entrada, lo más homogéneos posibles en cuanto al número de variables (1-2, 2-2, 2-3, …)

3º Trazamos el mapa con todas las combinaciones de las variables de entrada, de forma que todas las casillas adyacentes físicas deben ser adyacentes lógicas.

4º Trasladamos todos los términos que valen “1” al mapa de Karnough.

5º Trasladamos los términos indiferentes, si los hay.

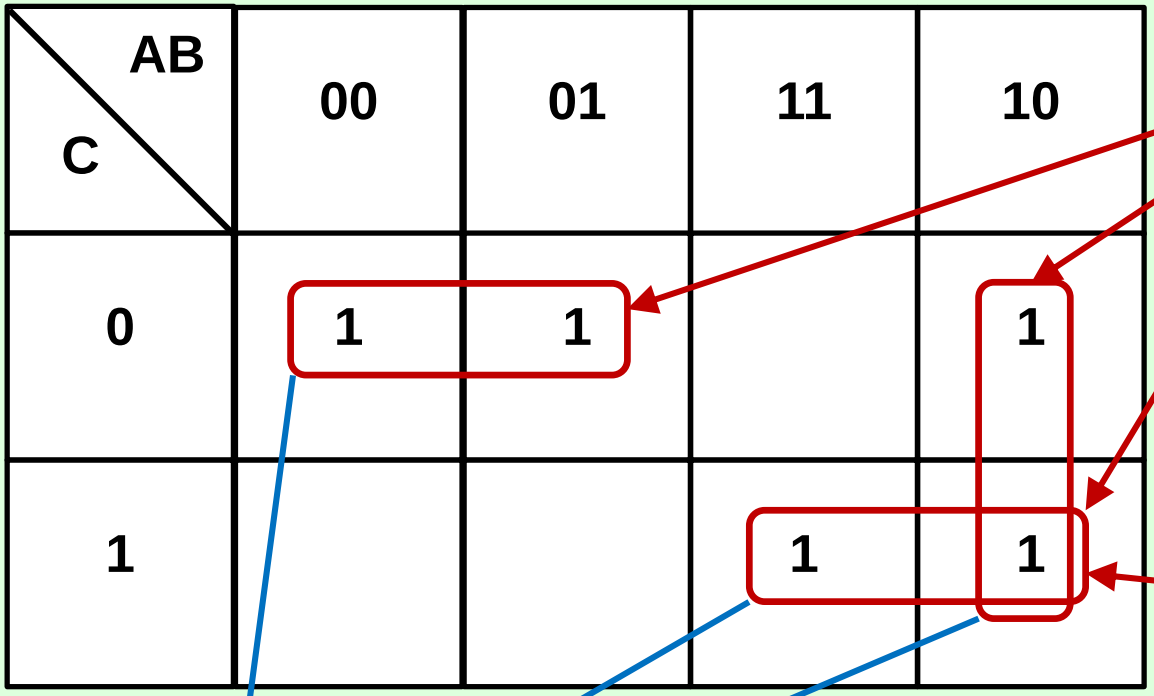
6º Realizamos agrupamientos de 2^n variables adyacentes físicas.

7º Se simplifica la función, teniendo en cuenta que los términos que se van son los que cambian en un mismo agrupamiento. En cada grupo de 2^n variables desparecen “n” variables.

Ejercicio: Minimizar la función f mediante el método del mapa de Karnaugh

A	B	C	f(A,B,C)
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

C \ AB	00	01	11	10
0	1	1		1
1			1	1



Agrupaciones de dos términos

Cada agrupación está formada por términos adyacentes física y algebraicamente

Ejemplo: Representa los términos:

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

$f(A,B,C) = \overline{A} \cdot \overline{C} + A \cdot C + A \cdot \overline{B}$

- En cada agrupación, la(s) variable(s) que se modifica(n) se elimina(n)
- La expresión simplificada puede escribirse directamente

Criterios para realizar agrupaciones:

1º Todos los unos deben pertenecer a alguna agrupación (en el peor caso una agrupación sólo tendrá un 1, dando lugar a un término con todas las variables)

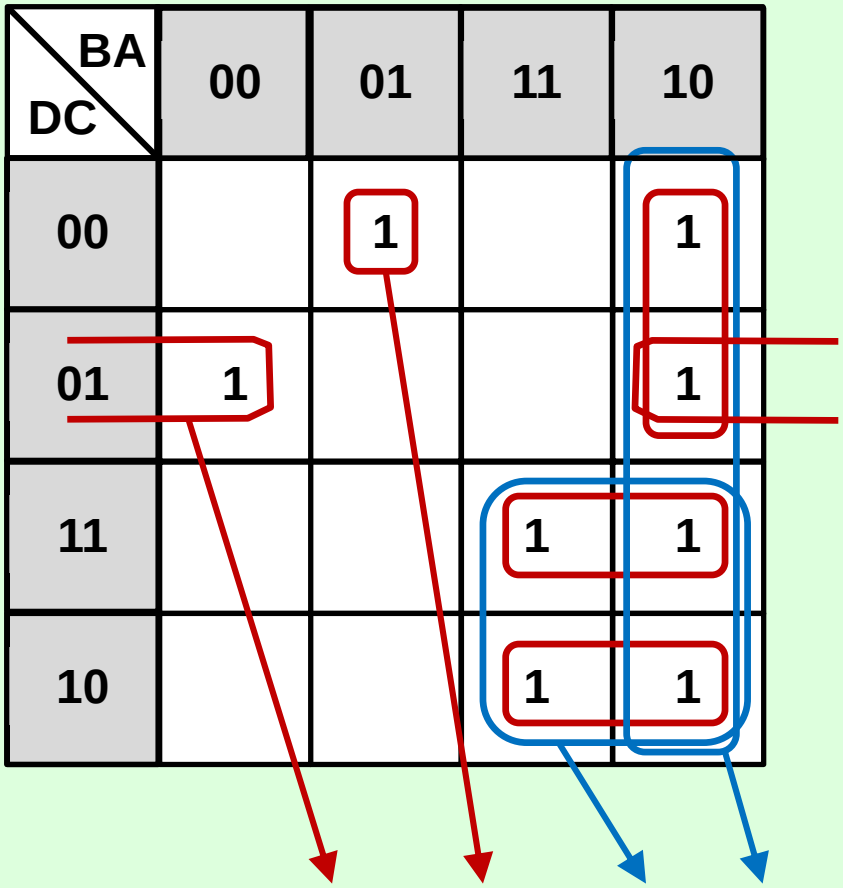
2º Un 1 puede pertenecer a varias agrupaciones, si con ello se consiguen grupos más grandes.

3º Pueden agruparse entre sí agrupaciones más pequeñas para eliminar más variables, si entre las agrupaciones sólo cambia el estado de una variable

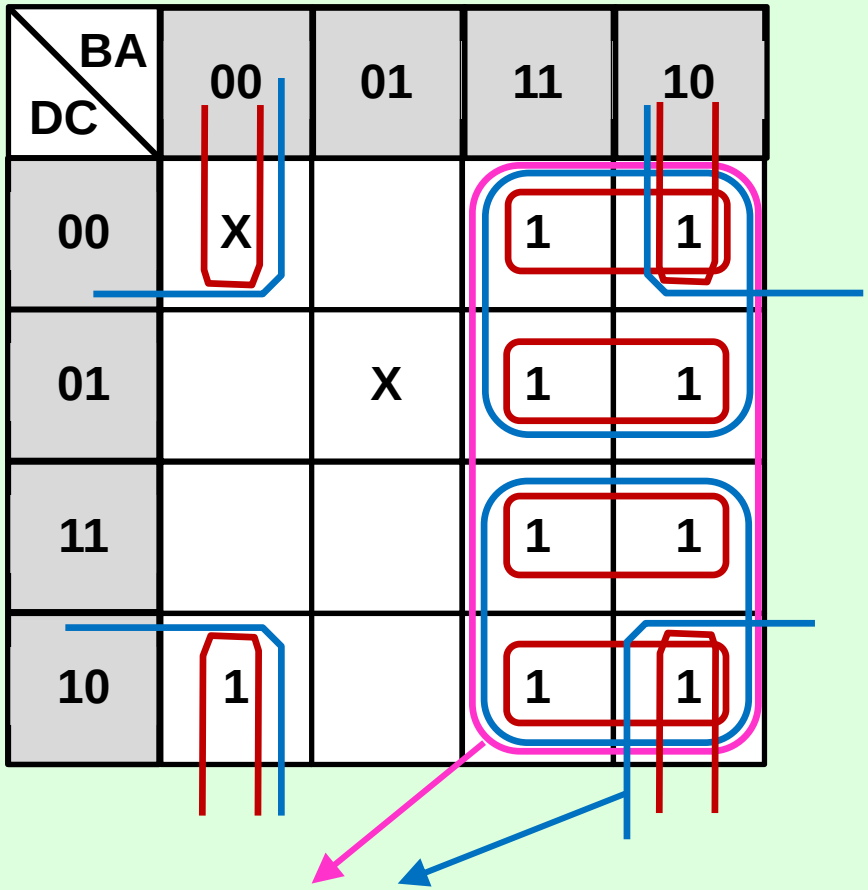
4º La tabla se considera cerrada sobre sí misma por sus bordes, ya que los términos extremos son también adyacentes

5º Los términos indiferentes pueden considerarse como unos o ceros discrecionalmente, para simplificar lo máximo posible (consideraremos como unos si permiten generar agrupaciones mayores, como cero si sólo sirven para generar nuevas agrupaciones sólo con ellos)

Algunos ejemplos:



$f(A,B,C,D)=\overline{A}\cdot C\cdot\overline{D}+A\cdot\overline{B}\cdot\overline{C}\cdot\overline{D}+B\cdot D+\overline{A}\cdot B$



$f(A,B,C,D)=B+\overline{A}\cdot\overline{C}$

Ejercicio:

Implementar de la forma más sencilla posible la función bit de paridad par para un código BCD