

SISTEMAS DIGITALES

Tema 3- Álgebra de Conmutación

José Luis Ávila Jiménez

Objetivos.

- Indicar los postulados o axiomas del Álgebra de Boole.
- Definir el Álgebra de Conmutación e indicar sus teoremas.
- Definir el concepto de función lógica, mostrando sus distintas formas de representación.
- Mostrar las funciones lógicas de dos variables, indicando sus características básicas.
- Definir el concepto de puerta lógica e indicar las puertas lógicas de las funciones lógicas de dos variables.
- Demostrar la universalidad de las puertas NAND y NOR.
- Comprender los conceptos de Mínterm y Máxterm.
- Analizar el desarrollo de Shannon.
- Comprender las dos formas lógicas y abreviadas para expresar una función lógica: suma de minterms y producto de máxterms.
- Mostrar los fundamentos en los que se basan los métodos de simplificación de funciones lógicas.
- Comprender la metodología de simplificación de funciones lógicas mediante los mapas de Karnaugh.
- Asimilar el concepto de función lógica incompletamente especificada y su forma de simplificación.

TEMA 3: Álgebra de Conmutación

1. – **Álgebra de Boole. Postulados y teoremas.**
2. – Funciones de conmutación:
 - 3.2.1.– Definición.
 - 3.2.2.– Formas de representación: tabla de verdad, expresión lógica y diagrama lógico.
3. – Funciones lógicas básicas: AND, OR, NOT, NAND, NOR, XOR Y XNOR.
 1. – Introducción a las puertas lógicas básicas.
 2. – Conjuntos funcionalmente completos. Suficiencia de las funciones NAND y NOR.
- 4.– Formas canónicas: concepto de minterm y máxterm.
- 3.5.– Desarrollo de Shannon: Primera y segunda forma.
- 3.6.– Fundamentos de la simplificación de funciones. Adyacencias.
- 3.7.– Funciones incompletamente especificadas.
- 3.8.– Método de simplificación de Karnaugh.

3.1.– Álgebra de Boole.

Un Álgebra es una estructura matemática que comprende un conjunto de elementos y un conjunto de operaciones u operadores, que actúan sobre dichos elementos.

Los postulados o axiomas determinan cómo se realizan dichas operaciones. Los postulados no se demuestran y permiten deducir los teoremas y propiedades de dicha estructura.

En 1854 George Boole presentó un tratamiento sistemático de la lógica binaria en su libro “Investigación sobre las Leyes del Pensamiento”, que ahora se denomina **álgebra de Boole**.

En 1938, Claude E. **Shannon** aplicó este álgebra particular para demostrar que las propiedades de los circuitos de conmutación eléctricos se podían representar con un álgebra booleana bivaluada, que se llamó **álgebra de conmutación**.

3.1.– Álgebra de Boole.

Los postulados son la hipótesis de partida para deducir teoremas y propiedades de un álgebra. Para el caso concreto del álgebra de Boole, se pueden utilizar diferentes conjuntos de postulados; uno de los más utilizados es el propuesto por Edward Vermilye **Huntington** en 1904.

Se parte de la existencia de un conjunto de elementos, B , en el que se puede establecer una relación de equivalencia, $=$, para la que se cumple el **principio de sustitución**:

$\forall x, y \in B$, **si x es equivalente a y , ($x = y$)**, se puede sustituir x por y , y viceversa.

3.1.– Álgebra de Boole. Postulados de Huntington.

P1: Leyes de composición interna: Se definen dos leyes de composición interna: $+$ (operador O, OR o suma) y \cdot (operador Y, AND o producto), siendo B cerrado para ambas operaciones:

$$\forall x, y \in B \quad a) x + y \in B$$

$$b) x \cdot y \in B$$

P2: Elementos neutros (o identidad): Existen elementos neutros para ambas operaciones, que son 0 para la suma, y 1 para el producto:

$$a) \exists 0 \in B / \forall x \in B, x + 0 = 0 + x = x$$

$$b) \exists 1 \in B / \forall x \in B, x \cdot 1 = 1 \cdot x = x$$

3.1.– Álgebra de Boole. Postulados de Huntington.

P3: Propiedad conmutativa: $\forall x, y \in B$ a) $x + y = y + x$

$$b) x \cdot y = y \cdot x$$

P4: Propiedad distributiva: $\forall x, y, z \in B$ a) $x + (y \cdot z) = (x + y) \cdot (x + z)$

$$b) x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

P5: Elemento complementario o complemento: $\forall x \in B, \exists y \in B / x + y = 1$

$$b) x \cdot y = 0$$

P6: Cardinalidad acotada: En el conjunto B existen al menos dos elementos diferentes.

$$\exists x, y \in B / x \neq y$$

3.1.– Álgebra de Boole.

Diferencias entre el Álgebra de Boole y el Álgebra definida en el campo de los números reales respecto a la suma y la resta:

- En un álgebra "corriente", la suma (+) no es distributiva respecto al producto (\cdot).
- El álgebra booleana no tiene inverso respecto a las dos operaciones, por tanto no tiene operaciones de resta ni división.
- Existen complementos en el álgebra booleana, pero no en el álgebra corriente.
- El álgebra de Boole se aplica a un conjunto finito de elementos, mientras que el álgebra corriente se aplica a un conjunto infinito de números reales.
- En estos postulados de Huntington no se incluye la propiedad asociativa, ya que se puede demostrar a partir de los postulados (teorema).

3.1.– Álgebra de Conmutación.

Álgebra de Conmutación: El Álgebra de Conmutación es un Álgebra de Boole que emplea solamente dos elementos. $B = \{0,1\}$. Lo representamos por B_2 .

Este álgebra booleana bivaluada constituye la base de la lógica matemática, en la que los dos elementos son verdadero y falso, y del diseño lógico, en el que los elementos son 0 y 1.

Los elementos 0 y 1 de B_2 corresponden a los dos valores binarios usados en los sistemas digitales.

Principio de dualidad: Si en una igualdad se intercambian los operadores "+" y ".", y los elementos 0 y 1, se obtiene otra igualdad válida.

3.1.– Álgebra de Boole. Teoremas.

T1: Ley de unicidad del complemento:

Los elementos neutros para la suma (0) y para el producto (1) son únicos

T2: Ley de idempotencia: $\forall x \in B_2,$ a) $x + x = x$

$$\text{b) } x \cdot x = x$$

T3: Elementos nulos: $\forall x \in B_2,$ a) $x + 1 = 1$

$$\text{b) } x \cdot 0 = 0$$

T4: Teorema de absorción: $\forall x, y \in B_2$ a) $x + (x \cdot y) = x$

$$\text{b) } x \cdot (x + y) = x$$

3.1.– Álgebra de Boole. Teoremas.

T5: Teorema de involución: El complemento del complemento de cada elemento, es el propio elemento. $\forall x \in B_2 \quad \overline{\overline{x}} = x$

T6: Propiedad asociativa de la suma y el producto: $\forall x, y, z \in B_2 \quad x + (y + z) = (x + y) + z$
 $x \cdot (y \cdot z) = (x \cdot y) \cdot z$

T7: Leyes de De Morgan: En general estas leyes dicen lo siguiente:

- El complemento de la suma es igual al producto de complementos.
- El complemento del producto es igual a la suma de complementos.

Para dos variables

$$\overline{x + y} = \overline{x} \cdot \overline{y}$$

$$\overline{x \cdot y} = \overline{x} + \overline{y}$$

Para n variables

$$\overline{x + y + z + \dots} = \overline{x} \cdot \overline{y} \cdot \overline{z} + \dots$$

$$\overline{x \cdot y \cdot z \cdot \dots} = \overline{x} + \overline{y} + \overline{z} + \dots$$

TEMA 3: Álgebra de Conmutación

1. – Álgebra de Boole. Postulados y teoremas.
2. – **Funciones de conmutación:**
 - 3.2.1.– **Definición.**
 - 3.2.2.– Formas de representación: tabla de verdad, expresión lógica y diagrama lógico.
3. – Funciones lógicas básicas: AND, OR, NOT, NAND, NOR, XOR Y XNOR.
 1. – Introducción a las puertas lógicas básicas.
 2. – Conjuntos funcionalmente completos. Suficiencia de las funciones NAND y NOR.
- 4.– Formas canónicas: concepto de minterm y máxterm.
- 3.5.– Desarrollo de Shannon: Primera y segunda forma.
- 3.6.– Fundamentos de la simplificación de funciones. Adyacencias.
- 3.7.– Funciones incompletamente especificadas.
- 3.8.– Método de simplificación de Karnaugh.

3.2.– Funciones de conmutación: Definición.

Una función lógica f , que representaremos $f(x_1, x_2, \dots, x_n)$, donde x_1, x_2, \dots, x_n son las variables de entrada y f la de salida, se define como toda función cuyos valores de entrada y salida solamente pueden ser los elementos del álgebra de conmutación, es decir 0 y 1, y están relacionados mediante los operadores del álgebra de conmutación $\{+, \bullet, ^-\}$.

Las variables de entrada y salida se denominan variables lógicas.

Las señales de entrada y salida de un sistema digital solamente pueden tomar los valores 0 y 1, por lo que:

- Se podrán representar mediante variables lógicas.
- Las señales de salida se podrán expresar matemáticamente a partir de las de entrada mediante una función lógica.
- El álgebra de conmutación permitirá el análisis y diseño de los sistemas digitales.



TEMA 3: Álgebra de Conmutación

- 3.1.– Álgebra de Boole. Postulados y teoremas.
- 3.2.– Funciones de conmutación:
 - 1. – Definición.
 - 2. – **Formas de representación: expresión lógica, tabla de verdad y diagrama lógico.**
- 3. – Funciones lógicas básicas: AND, OR, NOT, NAND, NOR, XOR Y XNOR.
 - 1. – Introducción a las puertas lógicas básicas.
 - 2. – Conjuntos funcionalmente completos. Suficiencia de las funciones NAND y NOR.
- 4.– Formas canónicas: concepto de minterm y máxterm.
- 3.5.– Desarrollo de Shannon: Primera y segunda forma.
- 3.6.– Fundamentos de la simplificación de funciones. Adyacencias.
- 3.7.– Funciones incompletamente especificadas.
- 3.8.– Método de simplificación de Karnaugh.

3.2.2.– Formas de representación: Expresión lógica.

Es una expresión algebraica que relaciona la variable lógica de salida con las de entrada mediante los operadores del Álgebra de Conmutación o Álgebra de Boole $\{+, \cdot, ^-\}$.

- Ejemplo:

$$Z_1 = \overline{x_1 \cdot x_2} + x_3$$

$$Z_2 = x_1 + (x_2 \overline{x_3})$$

Es decir, una función de conmutación puede expresarse como una combinación lineal de las entradas o de sus complementos. $f = f(x_1, x_2, \dots, x_n)$.

3.2.2.– Formas de representación: Tabla de verdad.

Indica el valor de la función lógica para cada combinación de valores de sus variables de entrada.

Para **m** funciones de **n** variables, consta de un encabezamiento, **2ⁿ** filas y de **n+m** columnas.

Ejemplo:

x_1	x_2	z_1	z_2	z_3
0	0	1	0	1
0	1	0	1	1
1	0	0	0	0
1	1	1	0	1

3.2.2.– Formas de representación: Diagrama lógico.

Implementación física de una función lógica mediante puertas lógicas.

Puerta lógica es un circuito digital que implementa un operador del álgebra de conmutación o una función lógica sencilla.

Se implementan mediante Circuitos Integrados Digitales.

Ejemplo:



TEMA 3: Álgebra de Conmutación

3.1.– Álgebra de Boole. Postulados y teoremas.

3.2.– Funciones de conmutación:

1. – Definición.
2. – Formas de representación: tabla de verdad, expresión lógica y diagrama lógico.

3. – Funciones lógicas básicas: AND, OR, NOT, NAND, NOR, XOR Y XNOR.

- 1. – Introducción a las puertas lógicas básicas.**
2. – Conjuntos funcionalmente completos. Suficiencia de las funciones NAND y NOR.

4.– Formas canónicas: concepto de minterm y máxterm.

3.5.– Desarrollo de Shannon: Primera y segunda forma.

3.6.– Fundamentos de la simplificación de funciones. Adyacencias.

3.7.– Funciones incompletamente especificadas.

3.8.– Método de simplificación de Karnaugh.

3.3.– Funciones lógicas básicas: Introducción.

Para implementar físicamente las funciones de conmutación se construye un circuito lógico en el que se usan las variables de la expresión como entrada al circuito lógico que contiene una o más puertas lógicas.

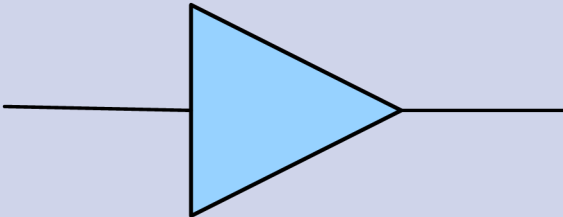
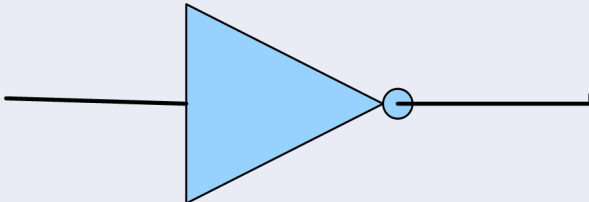
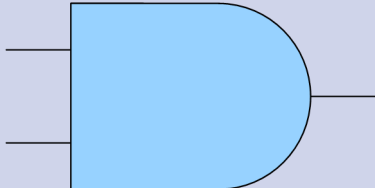
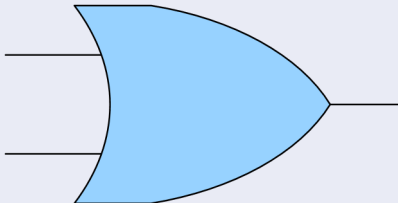
La colección de puertas lógicas que se usan para construir un circuito lógico se denomina biblioteca de puertas, y las puertas de biblioteca se denominan puertas normalizadas.

Una puerta lógica es un circuito digital que implementa un operador del álgebra de conmutación o una función lógica sencilla.

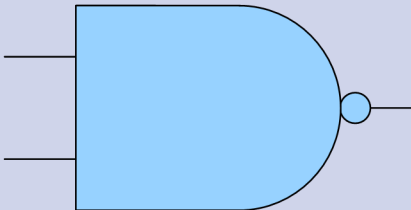
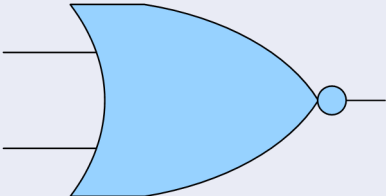
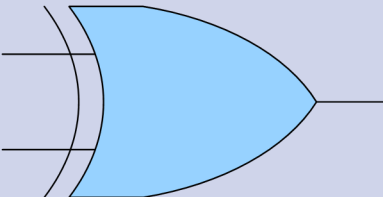
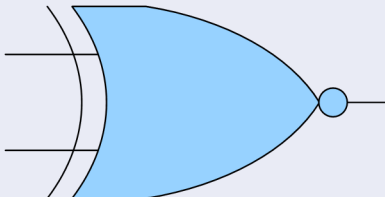
Generalmente, se seleccionan los ocho operadores siguientes:

- Las derivadas de los operadores del Álgebra de Conmutación:
 - Función lógica buffer.
 - Función lógica NOT.
 - Función lógica AND.
 - Función lógica OR.
- Las obtenidas por combinación de varios operadores del Álgebra de Conmutación:
 - Función lógica NAND.
 - Función lógica NOR.
 - Función lógica XOR.
 - Función lógica XNOR.

3.3.– Funciones lógicas básicas.

Nombre	Tabla de verdad	Expresión lógica	Símbolo lógico															
Adaptador o transferencia (buffer)	<table><tr><th>x</th><th>f=x</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	x	f=x	0	0	1	1	$f = x$										
x	f=x																	
0	0																	
1	1																	
NOT (Inversor)	<table><tr><th>x</th><th>f=¬x</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	x	f=¬x	0	1	1	0	$f = \overline{x}$										
x	f=¬x																	
0	1																	
1	0																	
AND (producto)	<table><tr><th>x</th><th>y</th><th>f=x · y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	f=x · y	0	0	0	0	1	0	1	0	0	1	1	1	$f = x \cdot y$	
x	y	f=x · y																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR (suma)	<table><tr><th>x</th><th>y</th><th>f=x + y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	f=x + y	0	0	0	0	1	1	1	0	1	1	1	1	$f = x + y$	
x	y	f=x + y																
0	0	0																
0	1	1																
1	0	1																
1	1	1																

3.3.– Funciones lógicas básicas.

Nombre	Tabla de verdad	Expresión lógica	Símbolo lógico															
NAND	<table><tr><td>x</td><td>y</td><td>$f=\overline{x \cdot y}$</td></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	$f=\overline{x \cdot y}$	0	0	1	0	1	1	1	0	1	1	1	0	$f = \overline{x \cdot y}$	
x	y	$f=\overline{x \cdot y}$																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR	<table><tr><td>x</td><td>y</td><td>$f=\overline{x + y}$</td></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	$f=\overline{x + y}$	0	0	1	0	1	0	1	0	0	1	1	0	$f = \overline{x + y}$	
x	y	$f=\overline{x + y}$																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
XOR (OR exclusiva)	<table><tr><td>x</td><td>y</td><td>$f=x \oplus y$</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	$f=x \oplus y$	0	0	0	0	1	1	1	0	1	1	1	0	$f = x \oplus y = x \cdot \overline{y} + \overline{x} \cdot y$	
x	y	$f=x \oplus y$																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
XNOR (equivalencia o comparación)	<table><tr><td>x</td><td>y</td><td>$f=\overline{x \oplus y}$</td></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	$f=\overline{x \oplus y}$	0	0	1	0	1	0	1	0	0	1	1	1	$f = \overline{x \oplus y} = \overline{x \cdot \overline{y} + \overline{x} \cdot y} = x \cdot y + \overline{x} \cdot \overline{y}$	
x	y	$f=\overline{x \oplus y}$																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

TEMA 3: Álgebra de Conmutación

3.1.– Álgebra de Boole. Postulados y teoremas.

3.2.– Funciones de conmutación:

1. – Definición.
2. – Formas de representación: tabla de verdad, expresión lógica y diagrama lógico.
3. – Funciones lógicas básicas: AND, OR, NOT, NAND, NOR, XOR Y XNOR.
 1. – Introducción a las puertas lógicas básicas.
 2. – **Conjuntos funcionalmente completos. Suficiencia de las funciones NAND y NOR.**
- 4.– Formas canónicas: concepto de minterm y máxterm.
- 3.5.– Desarrollo de Shannon: Primera y segunda forma.
- 3.6.– Fundamentos de la simplificación de funciones. Adyacencias.
- 3.7.– Funciones incompletamente especificadas.
- 3.8.– Método de simplificación de Karnaugh.

3.3.2.– Conjuntos funcionalmente completos.

Se dice que un conjunto de puertas lógicas es funcionalmente completo si en función de ellas se puede expresar cualquier función de conmutación.

Vamos a estudiar tres conjuntos funcionalmente completos: puertas NOT, AND y OR, puertas NAND y puertas NOR.

Las puertas básicas NOT, AND y OR constituyen un conjunto funcionalmente completo ya que coinciden con los operadores con que se expresa una función lógica mediante álgebra de Boole.

Vamos a demostrar que podemos construir NOT, AND y OR sólo con puertas NAND, y a continuación lo demostraremos con puertas NOR.

3.3.2.– Universalidad de las puertas NAND.

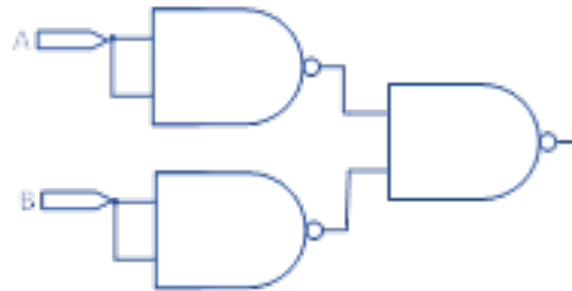
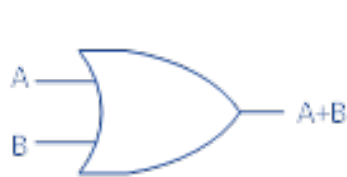
NOT: Aplicando la ley de idempotencia del producto se obtiene que uniendo las dos entradas de una puerta NAND actúa como una puerta NOT.



AND: Aplicamos la ley de involución y necesitamos dos puertas NAND de dos entradas.



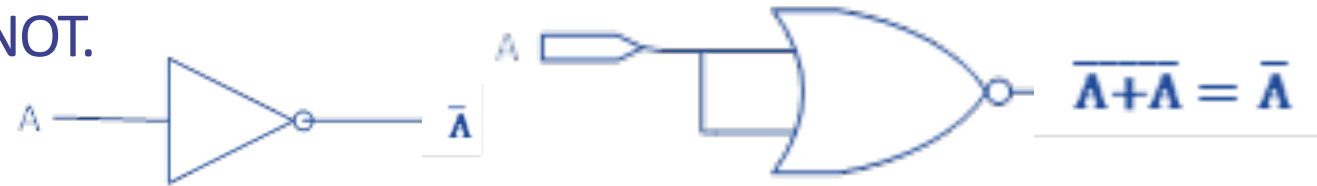
OR: Aplicamos la ley de involución y el teorema de De Morgan. Necesitamos tres puertas NAND de dos entradas.



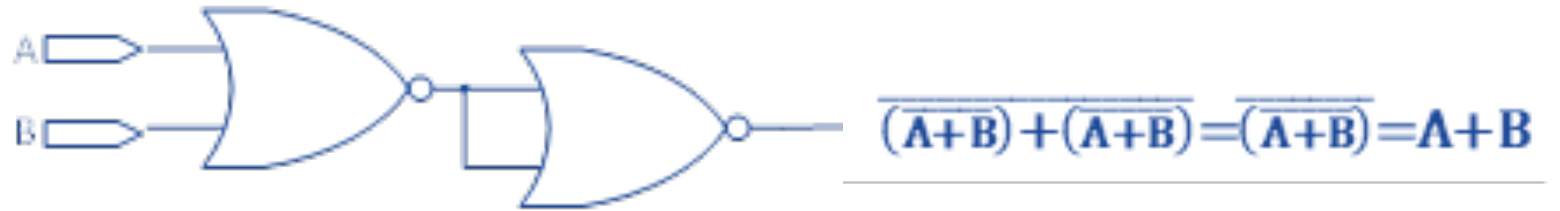
$$\overline{(\overline{A \cdot A}) \cdot (\overline{B \cdot B})} = \overline{\overline{A} \cdot \overline{B}} = \overline{\overline{A} + \overline{B}} = A + B$$

3.3.2.– Universalidad de las puertas NOR.

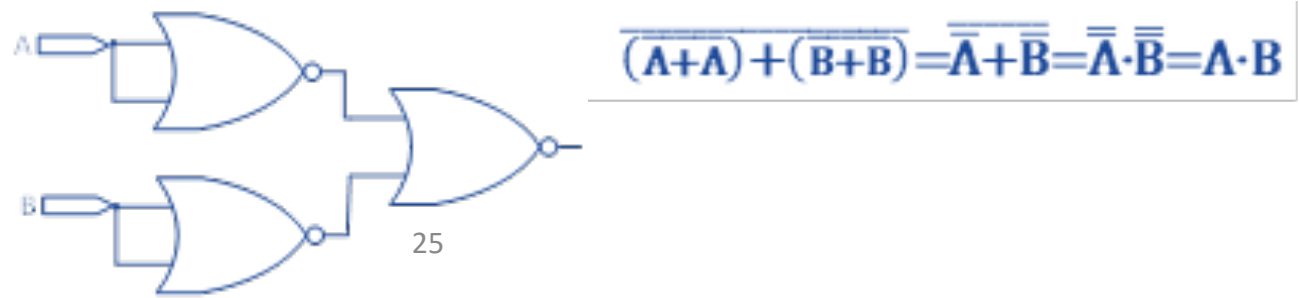
NOT: Aplicando la ley de idempotencia del producto se obtiene que uniendo las dos entradas de una puerta NOR actúa como una puerta NOT.



OR: Aplicamos el teorema de involución y necesitamos dos puertas NOR de dos entradas.



AND: Aplicamos la ley de involución y el teorema de De Morgan a uno de los complementos. Necesitamos tres puertas NOR de dos entradas.



TEMA 3: Álgebra de Conmutación

3.1.– Álgebra de Boole. Postulados y teoremas.

3.2.– Funciones de conmutación:

1. – Definición.

2. – Formas de representación: tabla de verdad, expresión lógica y diagrama lógico.

3. – Funciones lógicas básicas: AND, OR, NOT, NAND, NOR, XOR Y XNOR.

1. – Introducción a las puertas lógicas básicas.

2. – Conjuntos funcionalmente completos. Suficiencia de las funciones NAND y NOR.

4. – **Formas canónicas: concepto de minterm y máxterm.**

5. – Desarrollo de Shannon: Primera y segunda forma.

6.– Fundamentos de la simplificación de funciones. Adyacencias.

3.7.– Funciones incompletamente especificadas.

3.8.– Método de simplificación de Karnaugh.

3.4.– Formas canónicas: concepto de minterm y maxterm.

Vamos a ver la notación para poder convertir las tablas de verdad en expresiones algebraicas.

Los diferentes procedimientos de síntesis parten normalmente de expresiones canónicas de las funciones de conmutación.

Definición de conceptos previos:

- Un **literal** es una variable de conmutación o su complemento: Ej: x , y , \bar{x}
- Un **término producto** es una serie de literales conectados por el operador AND. Ej: $\bar{x} \cdot \bar{y} \cdot z$
- Un **término suma** es una serie de literales conectados por el operador OR. Ej: $\bar{x} + \bar{y} + z$
- Se dice que un término producto o un término suma es **normal** si en el mismo no aparece dos veces la misma variable o su complemento.

Normal $\bar{x} + \bar{y} + z$

No normal $\bar{x} + \bar{y} + z + \bar{x}$

3.4.– Formas canónicas: concepto de minterm.

Dado un conjunto de variables x, y, z, \dots ; se define una función lógica llamada **minterm** o término mínimo, o también producto fundamental o canónico de estas variables; como cualquier término producto normal en el que aparecen todas las variables.

Para todos los minterm de n variables, se cumple que asignan el valor lógico 1 a uno y sólo a uno de los caracteres del alfabeto de entrada. Por ejemplo, para tres variables (x, y, z) el minterm $\bar{x} \cdot \bar{y} \cdot z$ le asigna el valor 1 sólo a la combinación de entrada 001.

En general para n variables existirán 2^n minterms, que se numerarán como m_i , siendo $i = 0, 1, 2, \dots, 2^n - 1$. Con el valor lógico 0 indica que la variable está complementada y con el valor lógico 1 que aparece sin complementar.

3.4.– Formas canónicas: concepto de minterm.

Para una variable:

x	mínterm	notación
0	\bar{x}	m_0
1	x	m_1

Para dos variables:

x	y	mínterm	notación
0	0	$\bar{x} \cdot \bar{y}$	m_0
0	1	$\bar{x} \cdot y$	m_1
1	0	$x \cdot \bar{y}$	m_2
1	1	$x \cdot y$	m_3

Para tres variables:

x	y	z	mínterm	notación	m_1	m_5
0	0	0	$\bar{x} \cdot \bar{y} \cdot \bar{z}$	m_0	0	0
0	0	1	$\bar{x} \cdot \bar{y} \cdot z$	m_1	1	0
0	1	0	$\bar{x} \cdot y \cdot \bar{z}$	m_2	0	0
0	1	1	$\bar{x} \cdot y \cdot z$	m_3	0	0
1	0	0	$x \cdot \bar{y} \cdot \bar{z}$	m_4	0	0
1	0	1	$x \cdot \bar{y} \cdot z$	m_5	0	1
1	1	0	$x \cdot y \cdot \bar{z}$	m_6	0	0
1	1	1	$x \cdot y \cdot z$	m_7	0	0

$$x = 0 \Rightarrow \bar{x}$$

$$x = 1 \Rightarrow x$$

3.4.– Formas canónicas: concepto de máxterm.

Dado un conjunto de variables x, y, z, \dots ; se define una función lógica llamada **máxterm** o término máximo, o también suma fundamental o canónica de estas variables; como cualquier término suma normal en el que aparecen todas las variables.

Para todos los máxterm de n variables, se cumple que asignan el valor lógico 0 a uno y sólo a uno de los caracteres del alfabeto de entrada. Por ejemplo, para tres variables (x, y, z) el máxterm $\bar{x} + \bar{y} + z$ le asigna el valor 0 sólo al carácter de entrada 001.

Para n variables existen 2^n máxterm que pueden expresarse como una serie de ceros y unos, indicando con el valor 0 que la variable aparece sin complementar, y con el valor lógico 1 que aparece complementada (justo al contrario que los minterm).

Los máxterm de n variables son 2^n , que se numeran como M_i , siendo $i = 0, 1, \dots, 2^n - 1$.

3.4.– Formas canónicas: concepto de máxterm.

Para una variable:

x	máxterm	notación
0	x	M_0
1	\bar{x}	M_1

Para dos variables:

x	y	máxterm	notación
0	0	$x + y$	M_0
0	1	$x + \bar{y}$	M_1
1	0	$\bar{x} + y$	M_2
1	1	$\bar{x} + \bar{y}$	M_3

Para tres variables:

x	y	z	máxterm	notación	M_2	M_6
0	0	0	$x + y + z$	M_0	1	1
0	0	1	$x + y + \bar{z}$	M_1	1	1
0	1	0	$x + \bar{y} + z$	M_2	0	1
0	1	1	$x + \bar{y} + \bar{z}$	M_3	1	1
1	0	0	$\bar{x} + y + z$	M_4	1	1
1	0	1	$\bar{x} + y + \bar{z}$	M_5	1	1
1	1	0	$\bar{x} + \bar{y} + z$	M_6	1	0
1	1	1	$\bar{x} + \bar{y} + \bar{z}$	M_7	1	1

$$x = 0 \Rightarrow \bar{x}$$

$$x = 1 \Rightarrow x$$

3.4.– Formas canónicas: concepto de minterm y maxterm.

Las representaciones algebraicas de los minterms y los maxterms son el complemento, una de la otra:

x	y	z	mínterm		Máxterm	
0	0	0	$\bar{x} \cdot \bar{y} \cdot \bar{z}$	m_0	$x + y + z$	M_0
0	0	1	$\bar{x} \cdot \bar{y} \cdot z$	m_1	$x + y + \bar{z}$	M_1
0	1	0	$\bar{x} \cdot y \cdot \bar{z}$	m_2	$x + \bar{y} + z$	M_2
0	1	1	$\bar{x} \cdot y \cdot z$	m_3	$x + \bar{y} + \bar{z}$	M_3
1	0	0	$x \cdot \bar{y} \cdot \bar{z}$	m_4	$\bar{x} + y + z$	M_4
1	0	1	$x \cdot \bar{y} \cdot z$	m_5	$\bar{x} + y + \bar{z}$	M_5
1	1	0	$x \cdot y \cdot \bar{z}$	m_6	$\bar{x} + \bar{y} + z$	M_6
1	1	1	$x \cdot y \cdot z$	m_7	$\bar{x} + \bar{y} + \bar{z}$	M_7

TEMA 3: Álgebra de Conmutación

3.1.– Álgebra de Boole. Postulados y teoremas.

3.2.– Funciones de conmutación:

1. – Definición.
2. – Formas de representación: tabla de verdad, expresión lógica y diagrama lógico.
3. – Funciones lógicas básicas: AND, OR, NOT, NAND, NOR, XOR Y XNOR.
 1. – Introducción a las puertas lógicas básicas.
 2. – Conjuntos funcionalmente completos. Suficiencia de las funciones NAND y NOR.
4. – Formas canónicas: concepto de minterm y máxterm.
5. – **Desarrollo de Shannon: Primera y segunda forma.**
- 6.– Fundamentos de la simplificación de funciones. Adyacencias.
- 3.7.– Funciones incompletamente especificadas.
- 3.8.– Método de simplificación de Karnaugh.

3.5.— Desarrollo de Shannon.

- 📖 Para cualquier función de n variables vamos a obtener desarrollos únicos como suma de productos (mínterms) o como producto de sumas (máxterms). Estos desarrollos constituyen el teorema de Shannon en sus dos formas.
- 📖 Shannon demostró que cualquier función lógica puede expresarse como una suma única de mínterms o como un producto único de máxterms.

3.5.1.– Primera forma del Teorema de Shannon.

📖 “Toda función de conmutación se puede expresar como una suma única de minterms”.

📖 Para **una** variable:

$$f(x) = \bar{x} \cdot f(0) + x \cdot f(1)$$

$$\text{Si } x = 0 (\bar{x} = 1) \Rightarrow f(0) = 1 \cdot f(0) + 0 \cdot f(1) = f(0)$$

$$\text{Si } x = 1 (\bar{x} = 0) \Rightarrow f(1) = 0 \cdot f(0) + 1 \cdot f(1) = f(1)$$

Para **tres** variables:

$$f(x,y,z) = \bar{x}\bar{y}\bar{z}f(0,0,0) + \bar{x}\bar{y}z f(0,0,1) + \bar{x}y\bar{z} f(0,1,0) + \bar{x}y z f(0,1,1) +$$

$$+ x\bar{y}\bar{z} f(1,0,0) + x\bar{y}z f(1,0,1) + xy\bar{z} f(1,1,0) + xyz f(1,1,1)$$

Para **dos** variables:

$$f(x,y) = \bar{x}\bar{y}f(0,0) + \bar{x}y f(0,1) + x\bar{y} f(1,0) + xy f(1,1)$$

x	f(x)
0	f(x=0)
1	f(x=1)

x	y	f(x,y)
0	0	f(0,0)
0	1	f(0,1)
1	0	f(1,0)
1	1	f(1,1)

x	y	z	f(x,y,z)
0	0	0	f(0,0,0)
0	0	1	f(0,0,1)
0	1	0	f(0,1,0)
0	1	1	f(0,1,1)
1	0	0	f(1,0,0)
1	0	1	f(1,0,1)
1	1	0	f(1,1,0)
1	1	1	f(1,1,1)

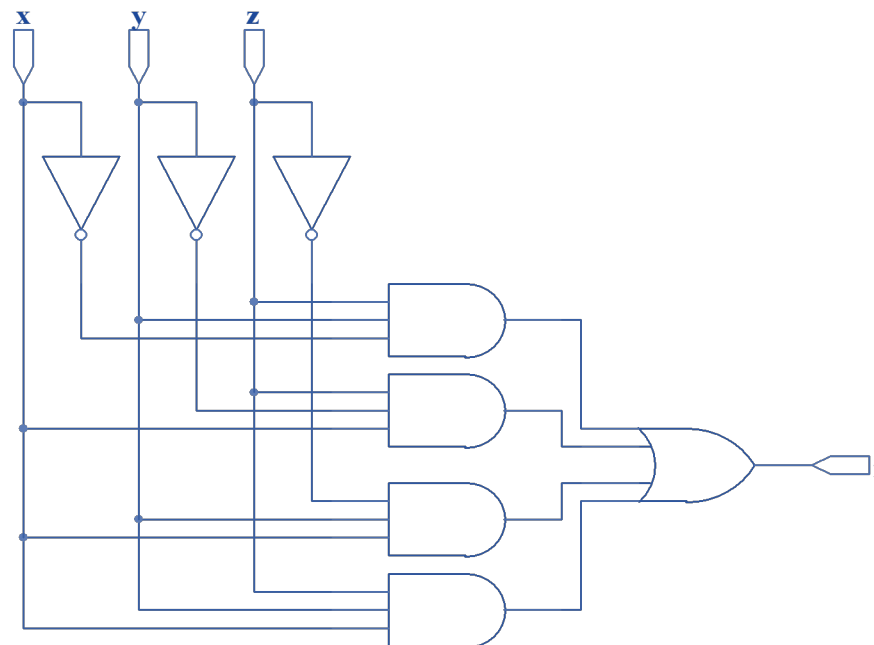
3.5.2.– Primera forma del Teorema de Shannon.

📖 **Conclusión:** cualquier función de n variables puede desarrollarse como suma de los 2^n minterm multiplicado por los respectivos valores de la función $f(x_i)$. Como es una suma, sólo intervendrán aquellos en que la función valga 1.

📖 **Ejemplo:** Sea la función dada por la tabla de verdad siguiente, obtenemos su expresión y circuito lógicos.

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$f = \sum m(3, 5, 6, 7) = m_3 + m_5 + m_6 + m_7 = \bar{x} \cdot y \cdot z + x \cdot \bar{y} \cdot z + x \cdot y \cdot \bar{z} + x \cdot y \cdot z$$



3.5.2.– Segunda forma del Teorema de Shannon.

📖 “Toda función de conmutación se puede expresar como un producto único de máxterms”.

📖 Para **una** variable:

Para **dos** variables:

$$f(x) = [x + f(0)] \cdot [\bar{x} + f(1)]$$

$$f(x,y) = [x+y+f(0,0)] \cdot [x+y\bar{z}+f(0,1)] \cdot [x\bar{y}+y+f(1,0)] \cdot [x\bar{y}+y\bar{z}+f(1,1)]$$

$$\text{Si } x = 0 (\bar{x} = 1) \Rightarrow f(0) = [0 + f(0)] \cdot [1 + f(1)] = f(0) \cdot 1 = f(0)$$

$$\text{Si } x = 1 (\bar{x} = 0) \Rightarrow f(1) = [1 + f(0)] \cdot [0 + f(1)] = 1 \cdot f(1) = f(1)$$

Para **tres** variables:

$$f(x,y,z) = [x+y+z+f(0,0,0)] \cdot [x+y+\bar{z}+f(0,0,1)] \cdot [x+y\bar{z}+z+f(0,1,0)] \cdot [x+y\bar{z}+\bar{z}+f(0,1,1)] \cdot [x\bar{y}+y+z+f(1,0,0)] \cdot [x\bar{y}+y+\bar{z}+f(1,0,1)] \cdot [x\bar{y}\bar{z}+z+f(1,1,0)] \cdot [x\bar{y}\bar{z}+\bar{z}+f(1,1,1)]$$

x	f(x)
0	f(x=0)
1	f(x=1)

x	y	f(x,y)
0	0	f(0,0)
0	1	f(0,1)
1	0	f(1,0)
1	1	f(1,1)

x	y	z	f(x,y,z)
0	0	0	f(0,0,0)
0	0	1	f(0,0,1)
0	1	0	f(0,1,0)
0	1	1	f(0,1,1)
1	0	0	f(1,0,0)
1	0	1	f(1,0,1)
1	1	0	f(1,1,0)
1	1	1	f(1,1,1)

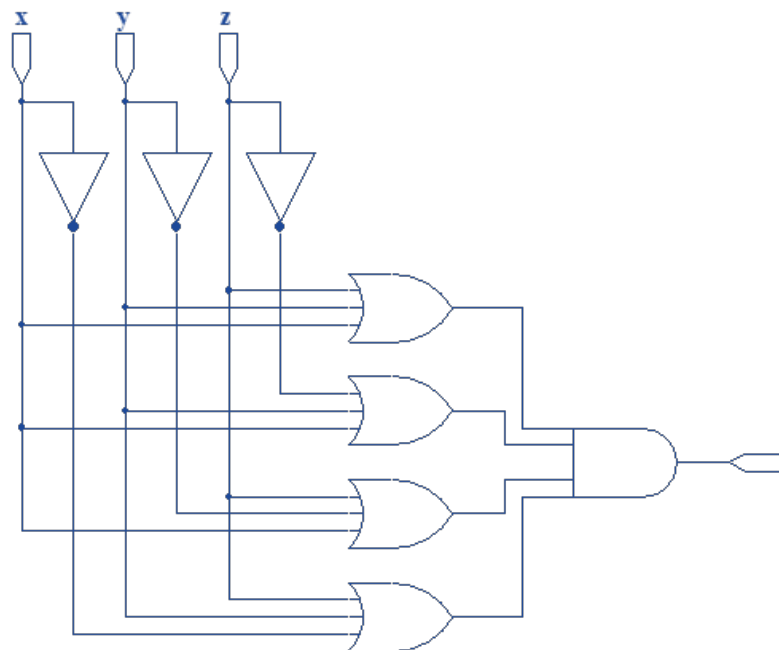
3.5.2.– Segunda forma del Teorema de Shannon.

📖 **Conclusión:** una función de conmutación tiene un desarrollo único en forma de producto de máxterm, interviniendo en dicho desarrollo los máxterms correspondientes a aquellos caracteres del alfabeto de entrada a los que la función asigne el valor 0.

📖 **Ejemplo:** Sea la función dada por la tabla de verdad siguiente, obtenemos su expresión y circuito lógicos.

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$f = \prod M(0,1,2,4) = M_0 \cdot M_1 \cdot M_2 \cdot M_4 = (x+y+z) \cdot (x+y+z') \cdot (x+y'+z) \cdot (x'+y+z)$$



TEMA 3: Álgebra de Conmutación

3.1.– Álgebra de Boole. Postulados y teoremas.

3.2.– Funciones de conmutación:

1. – Definición.

2. – Formas de representación: tabla de verdad, expresión lógica y diagrama lógico.

3. – Funciones lógicas básicas: AND, OR, NOT, NAND, NOR, XOR Y XNOR.

1. – Introducción a las puertas lógicas básicas.

2. – Conjuntos funcionalmente completos. Suficiencia de las funciones NAND y NOR.

4.– Formas canónicas: concepto de minterm y máxterm.

3.5.– Desarrollo de Shannon: Primera y segunda forma.

6. – Fundamentos de la simplificación de funciones. Adyacencias.

7.– Funciones incompletamente especificadas.


3.8.– Método de simplificación de Karnaugh.

3.6.— Fundamentos de la simplificación de funciones. Adyacencias.



- La mayoría de los métodos de simplificación de funciones lógicas, como los Mapas de Karnaugh, se basan en encontrar el mínimo número de adyacencias del mayor orden posible que cubran la función.
- Se analizará para una función lógica expresada en la forma de suma de minterms, pero es igualmente válido para el producto de maxterms.
- Dados dos minterms de las mismas variables, se dice que forman una **adyacencia de primer orden** si sus expresiones literales difieren solamente en un bit.
- Por ejemplo, dadas las cuatro variables x, y, z, u ; los minterms $x\bar{y}zu$ y $x\bar{y}\bar{z}u$ forman una adyacencia de primer orden. Aplicando la propiedad distributiva y el elemento complementario, podemos simplificar la función de la forma siguiente:

$$x\bar{y} \cdot zu + x\bar{y} \cdot \bar{z}u = x\bar{y} \cdot u(z + \bar{z}) = x\bar{y} \cdot u \cdot 1 = x\bar{y} \cdot u$$

3.6.— Fundamentos de la simplificación de funciones. Adyacencias.

-  Análogamente dos adyacencias de primer orden forman una adyacencia de segundo orden si difieren solamente en un bit. Por ejemplo para las cuatro variables x, y, z, u ; las adyacencias de primer orden $x\bar{y}u$ y $x\bar{y}\bar{u}$ forman una de segundo orden:

$$x\bar{y}u + x\bar{y}\bar{u} = x\bar{y}(u + \bar{u}) = x\bar{y}1 = x\bar{y}$$

-  Dos adyacencias de segundo orden formarían una adyacencia de tercer orden, y así sucesivamente.
-  Para sistematizar esta agrupación de adyacencias se identifican los minterms con las adyacencias de orden 0; una adyacencia de orden 1 incluye dos de orden 0; una adyacencia de orden 2 tiene dos de orden 1 y 4 de orden 0; una adyacencia de orden 3 incluye dos de orden 2, cuatro de orden 1 y ocho de orden 0; y, en general, una adyacencia de orden k incluirá a 2^k minterm.

3.6.— Fundamentos de la simplificación de funciones. Adyacencias.

Supongamos la función siguiente:

$$f = \Sigma m(0, 1, 2, 3, 7) = m_0 + m_1 + m_2 + m_3 + m_7 = \overline{x}\overline{y}\overline{z} + \overline{x}yz + \overline{x}y\overline{z} + \overline{x}yz + x \cdot y \cdot z$$

Teniendo en cuenta que cada término suma podemos **repetirlo** cuantas veces se quiera en virtud de la ley de idempotencia, podemos ir obteniendo las adyacencias como sigue:

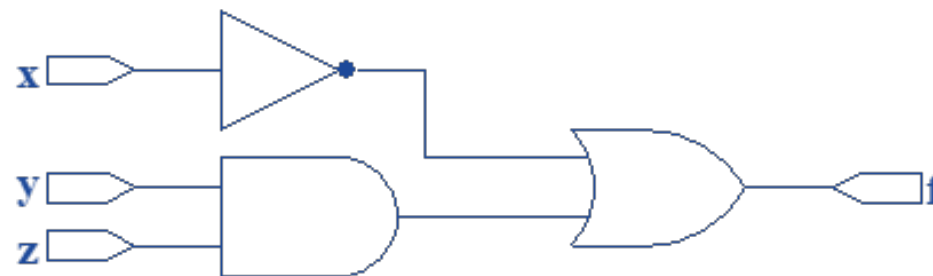
$$\left\{ \begin{array}{l} m_0 + m_1 = \overline{x}\overline{y}\overline{z} + \overline{x}yz = \overline{x}\overline{y} \\ m_2 + m_3 = \overline{x}y\overline{z} + \overline{x}yz = \overline{x}y \end{array} \right.$$

$$m_0 + m_1 + m_2 + m_3 = \overline{x}\overline{y} + \overline{x}y = \overline{x}$$

$$f = \Sigma m(0, 1, 2, 3, 7) = \overline{x} + y \cdot z$$

$$m_3 + m_7 = \overline{x}yz + x \cdot y \cdot z = y \cdot z$$

El circuito final sería mucho más simple:



TEMA 3: Álgebra de Conmutación

3.1.– Álgebra de Boole. Postulados y teoremas.

3.2.– Funciones de conmutación:

1. – Definición.
2. – Formas de representación: tabla de verdad, expresión lógica y diagrama lógico.
3. – Funciones lógicas básicas: AND, OR, NOT, NAND, NOR, XOR Y XNOR.
 1. – Introducción a las puertas lógicas básicas.
 2. – Conjuntos funcionalmente completos. Suficiencia de las funciones NAND y NOR.

4.– Formas canónicas: concepto de minterm y máxterm.

3.5.– Desarrollo de Shannon: Primera y segunda forma.

6. – Fundamentos de la simplificación de funciones. Adyacencias.

7. – Funciones incompletamente especificadas.

8. – Método de simplificación de Karnaugh.

3.7.– Funciones incompletamente especificadas.

- 📖 Son aquellas para las que en una o algunas combinaciones de valores de las variables de entrada no se conoce el valor 0 ó 1 correspondiente de la función, o es **indiferente** su valor.
- 📖 Por ejemplo, puede ocurrir que al especificar una función, se sepa de antemano que ciertas combinaciones de entrada nunca se van a presentar, con lo cual no importa el valor que se le asigne a las mismas.
- 📖 Cuando estas funciones se expresan en una tabla de verdad, las indiferencias se representan con guiones (–), cuando se representan como suma de minterm o producto de máxterm, se añade a la lista de los unos (o ceros) de la función, la lista de indiferencias.
- 📖 La existencia de indiferencias se traduce en una simplificación en la realización de las funciones de conmutación. Vamos a verlo en el siguiente ejemplo:

3.7.— Funciones incompletamente especificadas.

Supongamos la siguiente función:

$$f = \sum m(2,3,4,7,8,12,14) + d(0,13,15) = x'y'zu + x'yzu + x'yz'u + x'yzu + xy'z'u + xyz'u + xyzu$$

Buscamos las adyacencias posibles sin tener en cuenta las indiferencias:

$$m_2 + m_3 = x'y'zu + x'yzu = x'y'z m_3 + m_7$$

$$= x'y'zu + x'yzu = x'zu \quad m_4 + m_{12} =$$

$$x'yz'u + xyz'u = yz'u \quad m_8 + m_{12} =$$

$$x'yz'u + xyz'u = x'u \quad m_{12} + m_{14} =$$

$$xyz'u + xyzu = xyu$$

La función simplificada sin tener en cuenta las indiferencias quedaría así:

$$f = x'y'z + x'zu + yz'u + x'u + xyu$$

3.7.— Funciones incompletamente especificadas.

Supongamos la siguiente función:

$$f = \sum m(2,3,4,7,8,12,14) + d(0,13,15) = \overline{x}\overline{y}\overline{z}u + \overline{x}\overline{y}z\overline{u} + \overline{x}y\overline{z}\overline{u} + \overline{x}y\overline{z}u + \overline{x}y\overline{z}u + \overline{x}y\overline{z}u + \overline{x}y\overline{z}u + \overline{x}y\overline{z}u$$

Buscamos las adyacencias posibles teniendo en cuenta las indiferencias:

$$m_0(-) + m_4 + m_8 + m_{12} = \overline{x}\overline{y}\overline{z}\overline{u} + \overline{x}\overline{y}z\overline{u} + \overline{x}y\overline{z}\overline{u} + \overline{x}y\overline{z}u = \overline{x}\overline{y}\overline{z}(\overline{u} + u) + \overline{x}y\overline{z}(\overline{u} + u) = \overline{x}\overline{y}\overline{z} + \overline{x}y\overline{z}$$

$$)+ m_{14} + m_{15}(-) = \overline{x}y\overline{z}\overline{u} + \overline{x}y\overline{z}u + \overline{x}y\overline{z}u + \overline{x}y\overline{z}u = \overline{x}y\overline{z}(\overline{u} + u) + \overline{x}y\overline{z}(\overline{u} + u) = \overline{x}y\overline{z} + \overline{x}y\overline{z} = \overline{x}y\overline{z}$$

$$\overline{x}\overline{y}\overline{z} + \overline{x}y\overline{z} = \overline{x}\overline{z}$$

$$m_3 + m_7 = \overline{x}\overline{y}z\overline{u} + \overline{x}\overline{y}zu = \overline{x}\overline{y}z(\overline{u} + u) = \overline{x}\overline{y}z$$

La función simplificada **teniendo en cuenta las indiferencias** quedaría así:

$$f = \overline{x}\overline{z} + \overline{x}y\overline{z} + \overline{x}\overline{y}z$$

TEMA 3: Álgebra de Conmutación

3.1.– Álgebra de Boole. Postulados y teoremas.

3.2.– Funciones de conmutación:

1. – Definición.

2. – Formas de representación: tabla de verdad, expresión lógica y diagrama lógico.

3. – Funciones lógicas básicas: AND, OR, NOT, NAND, NOR, XOR Y XNOR.

1. – Introducción a las puertas lógicas básicas.

2. – Conjuntos funcionalmente completos. Suficiencia de las funciones NAND y NOR.

4.– Formas canónicas: concepto de minterm y máxterm.

3.5.– Desarrollo de Shannon: Primera y segunda forma.

3.6.– Fundamentos de la simplificación de funciones. Adyacencias.

3.7.– Funciones incompletamente especificadas.

3.8.– Método de simplificación de Karnaugh.

3.8.– Método de simplificación de Karnaugh.

- 📖 Es un método gráfico de simplificación de funciones lógicas.
- 📖 Un **mapa de Karnaugh** es una forma de representar cualquier función de conmutación en una tabla de forma compacta, de manera que cada celda con otra cualquiera que está junto a ella, difiere solamente en un bit.
- 📖 Un mapa de Karnaugh de **n** variables (o de dimensión **n**) está formado por **2ⁿ** celdas dispuestas en filas y columnas, de forma que cada celda está unívocamente identificada por **n** coordenadas, y pueden tomar los valores lógicos 0 ó 1.
- 📖 Los mapas de Karnaugh de dimensión **n** se van obteniendo a partir de los de dimensión inferior (**n–1**) en código reflejado (código Gray).
- 📖 Dejan de ser útiles a partir de 6 variables. En esta asignatura utilizaremos sólo hasta 4 variables

3.8.– Método de simplificación de Karnaugh.

 Para una variable:

x	0	1
	m_0	m_1

Para dos variables:

x\y	0	1
0	m_0	m_1
1	m_2	m_3

Para tres variables:

x\yz	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

Para cuatro variables

xy\zu	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

Para cinco variables (código reflejado, código Gray):

xy\zuv	000	001	011	010	110	111	101	100
00	m_0	m_1	m_3	m_2	m_6	m_7	m_5	m_4
01	m_8	m_9	m_{11}	m_{10}	m_{14}	m_{15}	m_{13}	m_{12}
11	m_{24}	m_{25}	m_{27}	m_{26}	m_{30}	m_{31}	m_{29}	m_{28}
10	m_{16}	m_{17}	m_{19}	m_{18}	m_{22}	m_{23}	m_{21}	m_{20}

3.8.– Método de simplificación de Karnaugh.

Para seis variables (código reflejado, código Gray):

xyz\uvw	000	001	011	010	110	111	101	100
000	m ₀	m ₁	m ₃	m ₂	m ₆	m ₇	m ₅	m ₄
001	m ₈	m ₉	m ₁₁	m ₁₀	m ₁₄	m ₁₅	m ₁₃	m ₁₂
011	m ₂₄	m ₂₅	m ₂₇	m ₂₆	m ₃₀	m ₃₁	m ₂₉	m ₂₈
010	m ₁₆	m ₁₇	m ₁₉	m ₁₈	m ₂₂	m ₂₃	m ₂₁	m ₂₀
110	m ₄₈	m ₄₉	m ₅₁	m ₅₀	m ₅₄	m ₅₅	m ₅₃	m ₅₂
111	m ₅₆	m ₅₇	m ₅₉	m ₅₈	m ₆₂	m ₆₃	m ₆₁	m ₆₀
101	m ₄₀	m ₄₁	m ₄₃	m ₄₂	m ₄₆	m ₄₇	m ₄₅	m ₄₄
100	m ₃₂	m ₃₃	m ₃₅	m ₃₄	m ₃₈	m ₃₉	m ₃₇	m ₃₆

3.8.– Método de simplificación de Karnaugh.

- Para **representar** una función de conmutación de n variables se usa un mapa de Karnaugh de n variables o de dimensión n. A cada celda del mapa de Karnaugh le corresponde un minterms o máxterms de la función.
- La coordenada de cada casilla y la representación vectorial de cada minterms o máxterms. En cada casilla se escribe el valor 0, 1, –, que tenga la función para el minterm correspondiente.
- Supongamos la misma función del apartado anterior:

$$f = \sum m(2, 3, 4, 7, 8, 12, 14) + d(0, 13, 15) = \bar{x}\bar{y}z\bar{u} + \bar{x}y\bar{z}\bar{u} + \bar{x}yz\bar{u} + \bar{x}yzu + x\bar{y}\bar{z}\bar{u} + x\bar{y}z\bar{u} + x\bar{y}zu + x\bar{y}z\bar{u}$$

xy\zu	00	01	11	10
00	m ₀	m ₁	m ₃	m ₂
01	m ₄	m ₅	m ₇	m ₆
11	m ₁₂	m ₁₃	m ₁₅	m ₁₄
10	m ₈	m ₉	m ₁₁	m ₁₀



xy\zu	00	01	11	10
00	–	0	1	1
01	1	0	1	0
11	1	–	–	1
10	1	0	0	0

3.8.– Método de simplificación de Karnaugh.

- De la misma manera que se habla de minterm adyacentes, se dice que dos celdas son adyacentes si sus coordenadas sólo se diferencian en un bit.
- Análogamente, si dos parejas de celdas forman una adyacencia de primer orden, pueden formar una adyacencia de segundo orden, si difieren solamente en un bit; y así sucesivamente.
- En un mapa de Karnaugh, una adyacencia de orden cero (minterm) corresponde a una celda del mapa; una adyacencia de orden uno corresponde a dos celdas adyacentes; una adyacencia de orden dos corresponde a cuatro celdas; una adyacencia de orden tres a ocho celdas, y así sucesivamente. En general, una adyacencia de orden k corresponde a 2^k celdas adyacentes.
- Si una función contiene indiferencias, se puede utilizar para construir adyacencias del mayor orden posible, aunque no es necesario cubrirlas.
- Cada celda puede utilizarse cuantas veces se desee, en virtud del teorema de idempotencia.

3.8.– Método de simplificación de Karnaugh.

Pasos a seguir para simplificar una función mediante mapas de Karnaugh:

- Se representa la función lógica correspondiente en el Mapa de Karnaugh.
- Identificar todas las adyacencias del mayor orden posible, de forma que cada una incluya algún “1” de la función, y cada “1” esté en alguna adyacencia.
- No debe aparecer ninguna adyacencia incluida en otra de mayor orden, y se deben cubrir todos los “unos” (mínterm) de la función.
- No es necesario cubrir las indiferencias de la función, sólo se utilizan si interesan para construir las adyacencias del mayor orden posible.
- Se determina el término producto (o suma) generado por cada adyacencia.
- Utilizando la lista de adyacencias representar la suma final.

3.8.– Método de simplificación de Karnaugh.

📖 **Ejemplo 1:** $f(x,y,z,u) = \sum m(0, 2, 3, 4, 9, 11) + d(1, 13)$

xy\zu	00	01	11	10
00	1	–	1	1
01	1			
11		–		
10		1	1	

0, 1, 2, 3: 00-- : ~~$\bar{x}\bar{y}$~~

1, 3, 9, 11: -0-1 : $y\bar{u}$

0, 4: 0-00 : ~~$\bar{x}\bar{z}\bar{u}$~~

= ~~$\bar{x}\bar{y}$~~ + $y\bar{u}$ + ~~$\bar{x}\bar{z}\bar{u}$~~

📖 Simplificación como suma de productos (“los unos”): Cuando la variable que permanece invariable es cero la variable se coloca complementada y cuando es uno la variable en la expresión está sin complementar.

3.8.– Método de simplificación de Karnaugh.

📖 **Ejemplo 2:** $f(x,y,z,u) = \sum m(3, 5, 6, 7, 9, 13)$

xy\zu	00	01	11	10
00			1	
01		1	1	1
11		1		
10		1		

$$3, 7 : 0-11 : \bar{x}\bar{z} \cdot u$$

$$6, 7 : 011- : x\bar{y} \cdot z$$

$$5, 13 : -101 : yz\bar{u} \quad (\text{Podría ser } 5,7 : 01-1 : x\bar{y} \cdot u)$$


$$9, 13 : 1-01 : xz\bar{u}$$

$$f = \bar{x}\bar{z} \cdot u + x\bar{y} \cdot z + yz\bar{u} + xz\bar{u}$$

📖 Admite dos simplificaciones igualmente válidas y con el mismo coste en puertas.

📖 Simplificación como suma de productos (“los unos”): Cuando la variable que permanece invariable es cero la variable se coloca complementada y cuando es uno la variable en la expresión está sin complementar.

3.8.– Método de simplificación de Karnaugh.

 **Ejemplo 3:** Puede ocurrir que una función admita varias simplificaciones distintas. En este caso debemos tomar la que necesite menos puertas lógicas y de menor número de entradas. $f(x,y,z) = \sum m(0, 1, 3, 4, 6, 7)$

x\yz	00	01	11	10
0	1	1	1	
1	1		1	1

$$f = \overline{y}z + \overline{x}z + xy$$

x\yz	00	01	11	10
0	1	1	1	
1	1		1	1

$$f = x \cdot \overline{z} + \overline{x}z + y \cdot z$$

 Admite dos simplificaciones igualmente válidas y con el mismo coste en puertas.

3.8.– Método de simplificación de Karnaugh.

- Se puede simplificar una función como **producto de máxterms**, para lo cual debemos tomar los ceros de la función. Se obtiene como **producto de sumas**.

Ejemplo 4: $f(x, y, z, u) = \prod M(1, 5, 7, 12) \cdot d(13)$

xy\zu	00	01	11	10
00		0		
01		0	0	
11	0	—		
10				

$$1, 5 : 0-01 : x+z+u$$

$$5, 7 : 01-1 : x+y+u$$

$$12, 13 : 110- : x+y+z$$

$$f = (x+z+u)(x+y+u)(x+y+z)$$

- Simplificación como producto de sumas (“los ceros”): Cuando la variable que permanece invariable es cero, la variable se coloca sin complementar, y cuando es uno la variable en la expresión está complementada.

- Se debe simplificar con los unos y los ceros y ver cuál es más rentable en puertas y número de entradas a las puertas.

3.8.– Método de simplificación de Karnaugh: Simplificación multifuncional.

- 📖 En los circuitos reales puede darse el caso de tener que implementar varias funciones de las mismas variables.
- 📖 Parece lógico plantearse la minimización conjunta, ya que estas salidas pueden compartir algunas puertas y así disminuir el coste final del circuito.
- 📖 El método de minimización multifuncional con mapas de Karnaugh consiste en tratar de encontrar todas las adyacencias del mayor orden posible que sean comunes a todas las funciones, o al mayor número de ellas.
- 📖 Con estas adyacencias se cubren el mayor número de unos posible, y los que queden sin cubrir se le buscan las adyacencias de igual forma que en los ejemplos anteriores.
- 📖 Veamos un ejemplo, donde tiene ventaja realizar la simplificación multifuncional. Sean las dos funciones siguientes.

3.8.– Simplificación multifuncional. Mapas de Karnaugh.

📖 **Ejemplo 5:** $f(x, y, z, u) = \sum m(1, 3, 5, 7, 10, 11, 14, 15)$

$$g(x, y, z, u) = \sum m(1, 5, 10, 12, 13, 14, 15)$$

xy\zu	00	01	11	10
00		1	1	
01		1	1	
11			1	1
10			1	1

$$f = x \cdot u + x \cdot z$$

xy\zu	00	01	11	10
00		1		
01		1		
11	1	1	1	1
10				1

$$g = xy + x \cdot z \cdot u + x \cdot z \cdot u$$

📖 Simplificadas por separado necesitaremos 5 puertas AND (3 de dos entradas y 2 de tres entradas) y dos puertas OR (1 de dos entradas y otra de tres entradas).

3.8.– Simplificación multifuncional. Mapas de Karnaugh.

📖 **Ejemplo 5:** $f(x, y, z, u) = \sum m(1, 3, 5, 7, 10, 11, 14, 15)$

$$g(x, y, z, u) = \sum m(1, 5, 10, 12, 13, 14, 15)$$

xy\zu	00	01	11	10
00		1	1	
01		1	1	
11			1	1
10			1	1

$$f = z \cdot u + x \cdot z \cdot u + x \cdot z \cdot u$$

xy\zu	00	01	11	10
00		1		
01		1		
11	1	1	1	1
10				1

$$g = x \cdot y + x \cdot z \cdot u + x \cdot z \cdot u$$

📖 Si compartimos las adyacencias $x \cdot z \cdot u$ y $x \cdot z \cdot u$ necesitaremos 4 puertas AND (2 de dos entradas y 2 de tres entradas) y dos puertas OR de tres entradas.



Bibliografía consultada.

- Antonio Lloris y Alberto Prieto. Sistemas Digitales 2º ed. Mc Graw-Hill. 2003.
- Daniel D. Gajski. Principios de Diseño Digital. Ed. Prentice Hall. 1997.

Sistemas Digitales

4º de graduado en Ingeniería Eléctrica

• ¡Muchas gracias por su atención!

• Carlos Diego Moreno Moreno

Área de Arquitectura y Tecnología de Computadores
Departamento de Ingeniería Electrónica y de Computadores.
Escuela Politécnica Superior. Universidad de Córdoba

