

---

# Unidad 4. Circuitos combinacionales

Circuitos Electrónicos Digitales  
E.T.S.I. Informática  
Universidad de Sevilla

Jorge Juan <jjchico@dte.us.es> 2010-2019

Esta obra esta sujeta a la Licencia Reconocimiento-CompartirIgual 4.0 Internacional de Creative Commons. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-sa/4.0/> o envíe una carta Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

# Contenido

---

- **Funciones lógicas**
- Álgebra de Boole
- Diseño combinacional
- Análisis funcional
- Análisis temporal

# Competencias

---

- Competencias principales

- 1) Transformar expresiones booleanas a sus formas canónicas y normalizadas (SOP y POS)
- 2) Representar cualquier función combinacional mediante tablas de verdad, formas normalizadas y circuito equivalente con puertas AND, OR e inversores
- 3) Reconocer la naturaleza combinacional de problemas reales simples y plantear una solución mediante una función lógica (F)
- 4) Obtener la expresión mínima en forma de SOP o POS de una función lógica, con o sin inespecificaciones, empleando el método del mapa de Karhaugh (F)
- 5) Implementar expresiones SOP/POS mediante circuitos mínimos en dos niveles usando puertas AND, OR e inversores, o bien sólo puertas NAND/NOR (F)
- 6) Obtener la expresión booleana correspondiente a un circuito combinacional construido mediante puertas lógicas (análisis combinacional) (F)
- 7) Representar la evolución temporal de las señales en un circuito combinacional dado el valor y/o formas de onda de sus entradas principales, empleando modelos de retraso sencillos (análisis temporal)

# Competencias

---

- Otras competencias
  - Conocer y saber aplicar los teoremas del Álgebra de Boole
  - Comprender la naturaleza de los azares y saber reconocerlos en el análisis temporal de circuitos

# Bibliografía

---

- LaMeres, capítulo 4.
- Floyd, capítulos 4 y 5.

# Funciones lógicas. Ejemplo

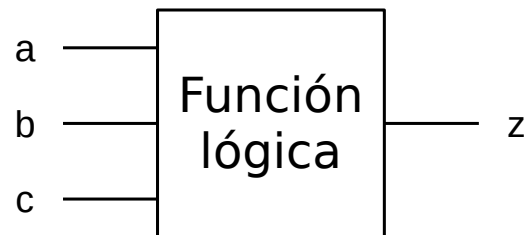
Descripción verbal

## Ejemplo 1

Un sistema de alarma digital puede estar activado o desactivado y dispone de un sensor de presencia y otro de apertura de puerta. Cuando el sistema está desactivado, se activará una señal de alarma si se detecta presencia y la puerta está abierta (se ha olvidado cerrar la puerta). Cuando el sistema está activado, se activará la señal de alarma si se detecta presencia o se abre la puerta.

Descripción formal

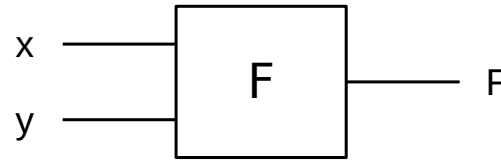
a: 0-desactivado, 1-activado  
b: 0-no presencia, 1-presencia.  
c: 0-p. cerrada, 1-p. abierta



a	b	c	z
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

- Tabla de verdad: especificación formal. Valor de la función para cada valor de las variables (n variables:  $2^n$  valores).
- Los circuitos digitales combinacionales implementan funciones lógicas

# Funciones lógicas de dos variables



x y	F <sub>0</sub>	F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>	F <sub>4</sub>	F <sub>5</sub>	F <sub>6</sub>	F <sub>7</sub>	F <sub>8</sub>	F <sub>9</sub>	F <sub>10</sub>	F <sub>11</sub>	F <sub>12</sub>	F <sub>13</sub>	F <sub>14</sub>	F <sub>15</sub>
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		AND					XOR	OR	NOR	XNOR					NAND	

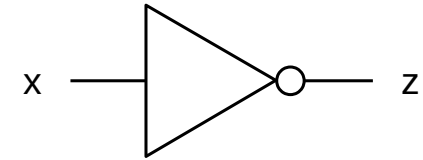
En general, hay  $2^{(2^n)}$  funciones lógicas de n variables

# Operadores lógicos básicos

NOT

x	z
0	1
1	0

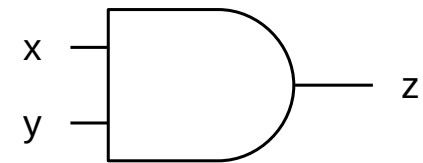
$$z = \bar{x}$$



AND

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

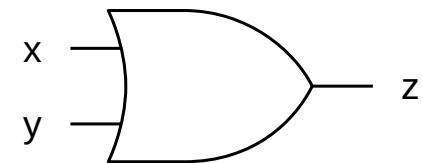
$$z = x \cdot y$$



OR

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

$$z = x + y$$



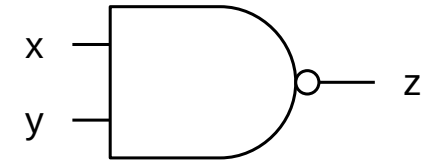


# Otros operadores lógicos

NAND

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

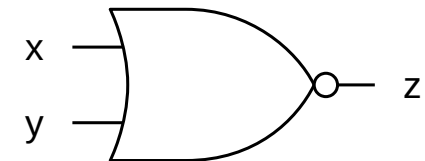
$$z = \overline{x \cdot y}$$



NOR

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

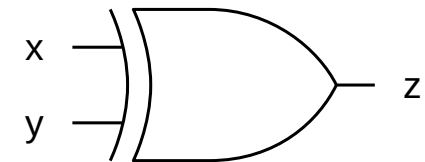
$$z = \overline{x + y}$$



XOR

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

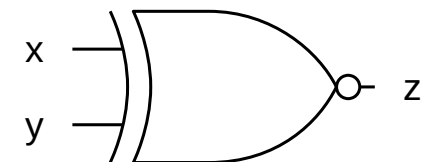
$$z = x \oplus y = \bar{x}y + x\bar{y}$$



XNOR

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

$$z = \overline{x \oplus y} = \bar{x}\bar{y} + xy$$



# Expresiones lógicas

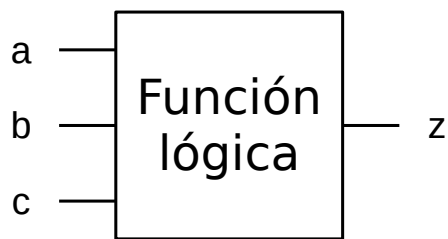
---

- Combinación de operadores AND ( $\cdot$ ), OR ( $+$ ) y NOT ( $\bar{\quad}$ )
- Precedencia de  $\cdot$  sobre  $+$ 
  - $x+(y\cdot z) = x+y\cdot z$
- "." puede ser omitido
  - $x+y\cdot z = x+yz$
- Son una forma de representación de funciones lógicas

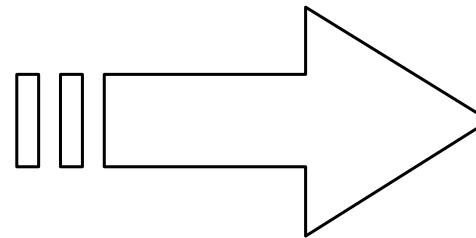
$$f(a,b,c) = (a+b+c) (a + \bar{b} c) + cd \overline{(a+c)}$$

# Funciones lógicas. Resumen

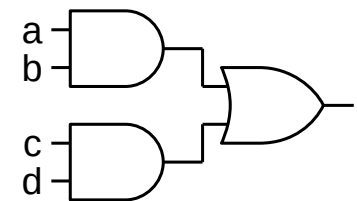
- Es posible construir circuitos elementales que realizan los operadores lógicos básicos (puertas lógicas)
- El objetivo del diseño combinacional es describir las funciones lógicas mediante operadores básicos que permitan construir un circuito digital que realice la función deseada



a	b	c	z
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



Circuito combinacional



# Contenido

---

- Funciones lógicas
- **Álgebra de Boole**
- Diseño combinacional
- Análisis funcional
- Análisis temporal

# Formalización: Álgebra de Boole

- $B = \{0, 1\}$
- $\{B, \text{NOT}, \text{AND}, \text{OR}\}$  forman un Álgebra de Boole
  - Caso particular: Álgebra de Conmutación
- Un Álgebra de Boole cumple los siguiente axiomas



George Boole (1815-1864)

Identidad	$x + 0 = x$	$x \cdot 1 = x$
Conmutativa	$x + y = y + x$	$x \cdot y = y \cdot x$
Distributiva	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	$x + (y \cdot z) = (x + y) \cdot (x + z)$
Complemento	$x + \bar{x} = 1$	$x \cdot \bar{x} = 0$

**Dualidad:** si una expresión se cumple, la expresión que resulta de intercambiar  $+$  con  $\cdot$  y  $0$  con  $1$  también se cumple

# Álgebra de Boole. Teoremas

Idempotencia	$x+x = x$	$x \cdot x = x$
Unicidad del complemento	$\bar{x}$ es único	
Elementos dominantes	$x+1 = 1$	$x \cdot 0 = 0$
Involución	$\overline{(\bar{x})} = x$	
Absorción	$x+xy = x$	$x \cdot (x+y) = x$
Consenso	$x+\bar{x}y = x+y$	$x \cdot (\bar{x}+y) = x \cdot y$
Asociativa	$x+(y+z) = (x+y)+z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
De Morgan	$\overline{x \cdot y} = \bar{x} + \bar{y}$	$\overline{x+y} = \bar{x} \cdot \bar{y}$
Reducción	$xy+x\bar{y} = x$	$(x+y)(x+\bar{y}) = x$

# Formas normalizadas

---

- Suma de productos (SOP)
  - Suma de términos producto
  - Término producto: producto de variables (complementadas o no), cada variable aparece una sola vez
- Producto de sumas (POS)
  - Producto de términos suma
  - Término suma: suma de variables (complementadas o no), cada variable aparece una sola vez
- Obtención
  - Aplicación reiterada de Teorema de De Morgan, Propiedad distributiva y teoremas básicos

# Formas normalizadas. Ejemplo

---

$$(a+b+c) (a + \bar{b} c) + cd \overline{(a+c)} =$$

$$aa + a\bar{b}c + ba + b\bar{b}c + ca + c\bar{b}c + c d \bar{a} \bar{c} =$$

$$a + a\bar{b}c + ab + ac + \bar{b}c$$

$$(a+b+c) (a + \bar{b} c) + cd \overline{(a+c)} =$$

$$(a+b+c)(a+\bar{b})(a+c) + c d \bar{a} \bar{c} =$$

$$(a+b+c)(a+\bar{b})(a+c)$$



# Formas canónicas

---

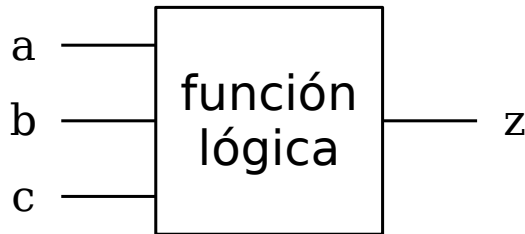
- Dado un conjunto de  $n$  variables:
- Mintérmino
  - Término producto que contiene todas las variables
- Maxtérmino
  - Término suma que contiene todas las variables
- Forma canónica de mintérminos
  - SOP donde todos los términos producto son mintérminos
- Forma canónica de maxtérminos
  - POS donde todos los términos suma son maxtérminos

$$\{a, b, c\}$$

$$\bar{a} b c + a \bar{b} c + a b \bar{c} + a b c$$

$$(a+b+c)(a+b+\bar{c})(a+\bar{b}+c)(\bar{a}+b+c)$$

# Formas canónicas. Mintérminos



Siempre es posible obtener una expresión para una función combinando NOT, AND y OR.

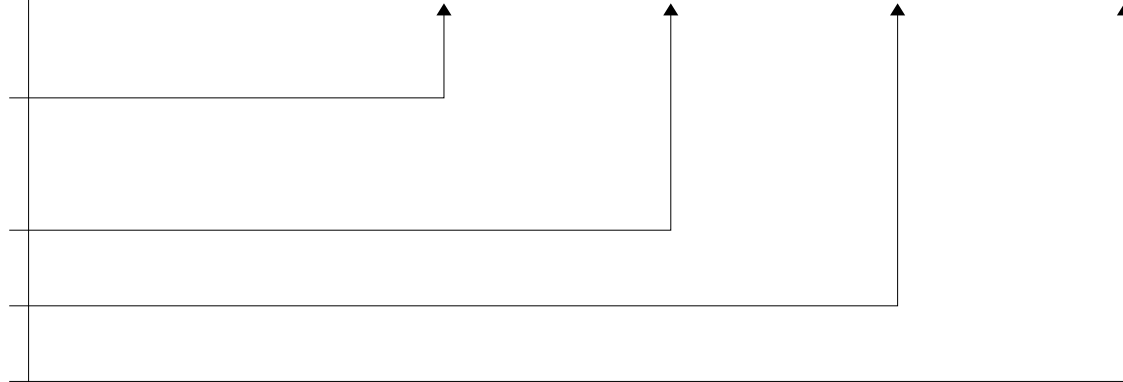
Método:

1. Para cada valor "1" de la función se construye un término producto que sea "1" sólo para esa combinación de valores de entrada.
2. Se suman (OR) todos los términos.

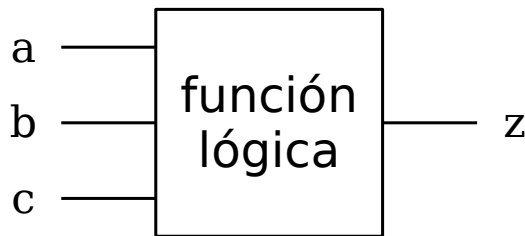
El resultado es una expresión de la función en **forma canónica de mintérminos**.

a b c	z
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

$$z = \bar{a} b c + a \bar{b} c + a b \bar{c} + a b c$$



# Formas canónicas. Maxtérminos



Siempre es posible obtener una expresión para una función combinando NOT, AND y OR.

Método:

1. Para cada valor "0" de la función se construye un término suma que sea "0" sólo para esa combinación de valores de entrada.
2. Se multiplican (AND) todos los términos.

El resultado es una expresión de la función en **forma canónica de maxtérminos**.

a	b	c	z
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

$$z = (a+b+c) \cdot (a+b+\bar{c}) \cdot (a+\bar{b}+c) \cdot (\bar{a}+b+c)$$

# Formas canónicas. Notación

a b c	mintérmino	notación-m
0 0 0	$\bar{a} \bar{b} \bar{c}$	m0
0 0 1	$\bar{a} \bar{b} c$	m1
0 1 0	$\bar{a} b \bar{c}$	m2
0 1 1	$\bar{a} b c$	m3
1 0 0	$a \bar{b} \bar{c}$	m4
1 0 1	$a \bar{b} c$	m5
1 1 0	$a b \bar{c}$	m6
1 1 1	$a b c$	m7

$$z = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc$$

$$z = m3 + m5 + m6 + m7$$

$$z = \Sigma(3,5,6,7)$$

mintérmino: combinación de entradas que hace la función igual a "1"

a b c	maxtérmino	notación-M
0 0 0	$a+b+c$	M0
0 0 1	$a+b+\bar{c}$	M1
0 1 0	$a+\bar{b}+c$	M2
0 1 1	$a+\bar{b}+\bar{c}$	M3
1 0 0	$\bar{a}+b+c$	M4
1 0 1	$\bar{a}+b+\bar{c}$	M5
1 1 0	$\bar{a}+\bar{b}+c$	M6
1 1 1	$\bar{a}+\bar{b}+\bar{c}$	M7

$$z = (a+b+c)(a+b+\bar{c})(a+\bar{b}+c)(\bar{a}+b+c)$$

$$z = M0 M1 M2 M4$$

$$z = \Pi(0,1,2,4)$$

maxtérmino: combinación de entrada que hace la función igual a "0"

# Conversión

---

- Formas canónicas (mint./maxt.)  $\leftrightarrow$  tabla de verdad
  - Relación directa
- SOP/POS  $\rightarrow$  formas canónicas
  - Ampliando cada término producto/suma con las variables que faltan.
  - SOP: multiplicando por 1. Ej:  $(a + \bar{a})$
  - POS: sumando 0. Ej:  $(a\bar{a})$
  - Simplificando mintérminos/maxtérminos iguales.
- SOP/POS  $\rightarrow$  tabla de verdad
  - Método 1: pasando previamente a forma canónica.
  - Método 2: identificando términos con posiciones de la tabla de verdad.
    - Término producto: unos de la función
    - Término suma: ceros de la función

# Conversión. SOP/POS a forma canónica. Ejemplo

---

$$a + a \bar{b} c + ac + \bar{b}c =$$

$$a(b+\bar{b})(c+\bar{c})+a\bar{b}c+(a+\bar{a})\bar{b}c =$$

$$a b c+a b \bar{c}+a \bar{b} c+a \bar{b} \bar{c}+a \bar{b} c+a \bar{b} c+\bar{a} \bar{b} c =$$

$$a b c + a b \bar{c} + a \bar{b} c + a \bar{b} c + \bar{a} \bar{b} c$$

$$(a+\bar{b}+c)(a+b)(a+\bar{c}) =$$

$$(a+\bar{b}+c)(a+b+c\bar{c})(a+\bar{c}+b\bar{b}) =$$

$$(a+\bar{b}+c)(a+b+c)(a+b+\bar{c})(a+b+\bar{c})(a+\bar{b}+\bar{c}) =$$

$$(a+\bar{b}+c)(a+b+c)(a+b+\bar{c})(a+\bar{b}+\bar{c})$$

# Conversión. SOP/POS a tabla de verdad

---

$$z(a,b,c) = a + a \bar{b} c + ac + \bar{b}c$$

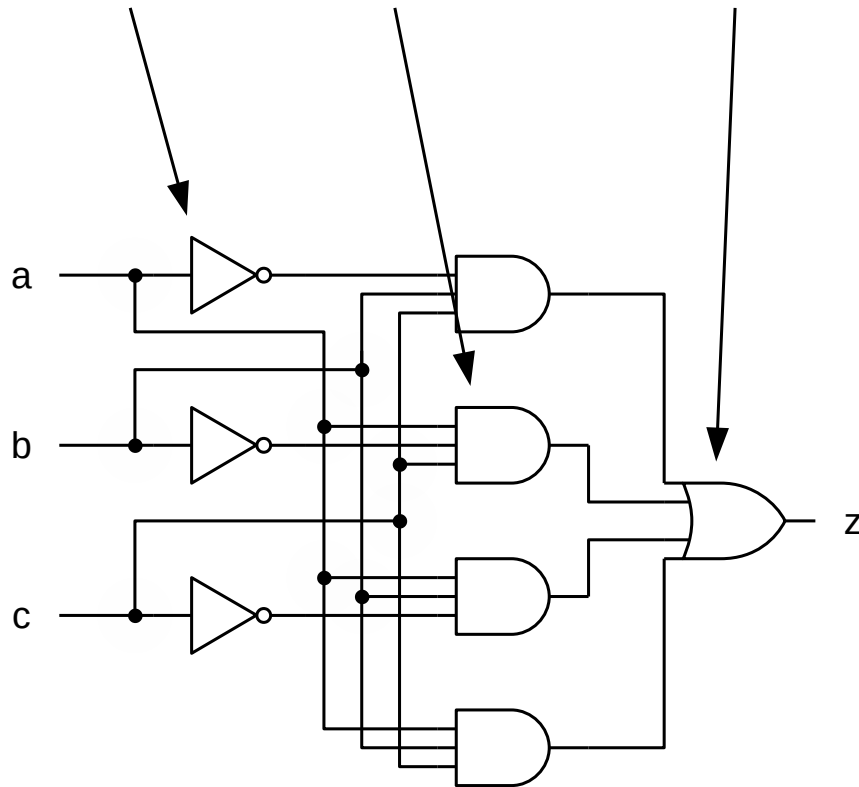
- $z = 1$  si y sólo si
  - $a=1$ , o bien
  - $a=1$  y  $b=0$  y  $c=1$ , o bien
  - $a=1$  y  $c=1$ , o bien
  - $b=0$  y  $c=1$

$$z(a,b,c) = (a+b+c)(a+\bar{b})(a+c)$$

- $z = 0$  si y sólo si
  - $a=0$  y  $b=0$  y  $c=0$ , o bien
  - $a=0$  y  $b=1$ , o bien
  - $a=0$  y  $c=0$

# Aproximación al diseño lógico

$$z = \bar{a} b c + a \bar{b} c + a b \bar{c} + a b c$$



Cualquier SOP (POS) puede ser trasladada directamente a un circuito que implemente la función usando puertas lógicas básicas:

- Inversores (complementos)
- AND (productos)
- OR (sumas)

¿Es mínimo el resultado?



# Contenido

---

- Funciones lógicas
- Álgebra de Boole
- **Diseño combinacional**
- Análisis funcional
- Análisis temporal

# Diseño de circuitos combinatoriales

---

- Basado en la síntesis de circuitos a partir de formas normalizadas (SOP o POS)
- Una expresión más simple dará un circuito más simple
- Las formas normalizadas pueden ser simplificadas de forma sistemática (automatizable)
- Criterios de simplificación básicos
  - Expresiones equivalentes: mismos valores para las mismas entradas.
  - Mínimo número de términos → mínimo número de puertas
  - Mínimo número de literales en cada término → mínimo número de entradas en cada puerta
- Otros criterios
  - Tipos de puertas disponibles
  - Facilidad para hacer complementos (inversores)
  - Consumo de potencia, velocidad de operación, etc.

# Simplificación de formas normalizadas

$$xy + x\bar{y} = x$$

"x" puede ser cualquier expresión

$$z = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc \longrightarrow \text{términos orden 0}$$
  
$$z = bc + ac + ab \longrightarrow \text{términos orden 1}$$

Cada término puede ser usado más de una vez  
( $x+x=x$ )

Implicante de una función

- Término producto que puede ser parte de una expresión en forma de SOP de la función.
- Contiene o "cubre" varios mintérminos de la función.

# Algoritmo de minimización Quine-McCluskey (simp.)

---

1. Comenzar con la lista de los mintérminos de la función
2. Buscar implicantes de primer orden (TO-1)  
Aplicando el teorema de reducción a mintérminos adyacentes
3. Eliminar mintérminos cubiertos de la expresión final  
No hay que preocuparse de los mintérminos cubiertos porque están incluidos en los TO-1
4. Buscar implicantes de segundo orden (TO-2)  
Aplicando el teorema de reducción a los TO-1 adyacentes
5. Eliminar los TO-1 cubiertos de la expresión final
6. Continuar hasta obtener implicantes del mayor orden posible (implicantes primas)  
Las implicantes primas son los términos más simples que contienen todos los mintérminos de la función.  
Las implicantes primas esenciales son los que contienen mintérminos no contenidos en ninguna otra implicante prima.
7. Selecciona todas las implicantes primas esenciales y/o un número mínimo de ellas que cubra todos los mintérminos de la función.

# Algoritmo de minimización Quine-McCluskey (simp.)

---

$$F(a,b,c,d) = \Sigma(0,1,4,9,11,13,15)$$

$$F(a,b,c,d) = \bar{a} \bar{b} \bar{c} \bar{d} + \bar{a} \bar{b} \bar{c} d + \bar{a} b \bar{c} \bar{d} + a \bar{b} \bar{c} d + a \bar{b} c d + a b \bar{c} d + a b c d$$

$$\bar{a} \bar{b} \bar{c} \bar{d}$$

$$\bar{a} \bar{b} \bar{c} d$$

$$\bar{a} b \bar{c} \bar{d}$$

$$a \bar{b} \bar{c} d$$

$$a \bar{b} c d$$

$$a b \bar{c} d$$

$$a b c d$$

# Mapas de Karnaugh (K-mapa)

- K-mapas:
  - Tabla bidimensional
  - Valores de variables ordenados como en código gray
  - Cada celda corresponde a un mintérmino/maxtérmino
  - Localización fácil de mintérminos y términos de orden superior
  - ¡Son cíclicos!
  - Localización fácil de implicantes primos
  - Facilitan la simplificación de expresiones normalizadas mediante el método de Quine-McCluskey

	ab			
cd	00	01	11	10
00	0 <sup>0</sup>	4 <sup>4</sup>	12 <sup>12</sup>	8 <sup>8</sup>
01	1 <sup>1</sup>	5 <sup>5</sup>	13 <sup>13</sup>	9 <sup>9</sup>
11	3 <sup>3</sup>	7 <sup>7</sup>	15 <sup>15</sup>	11 <sup>11</sup>
10	2 <sup>2</sup>	6 <sup>6</sup>	14 <sup>14</sup>	10 <sup>10</sup>

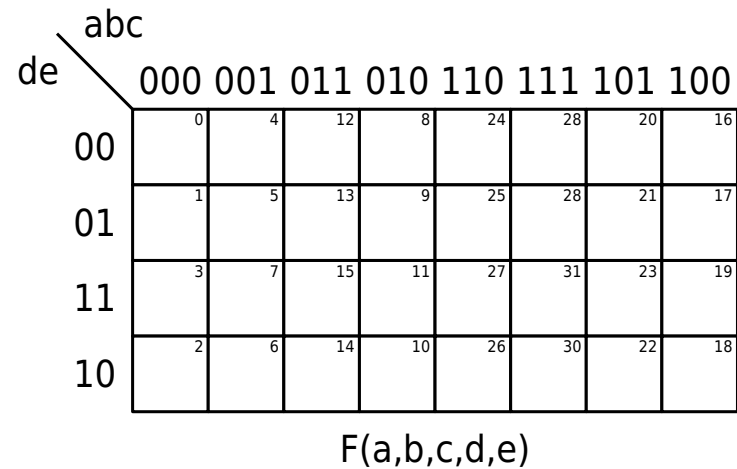
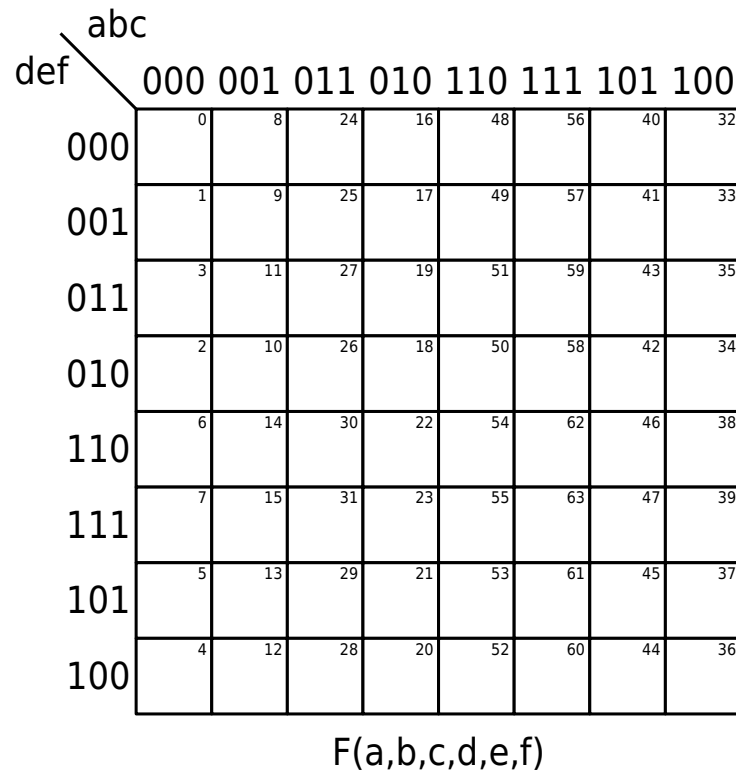
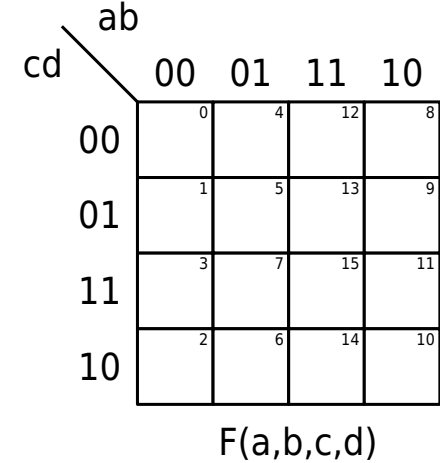
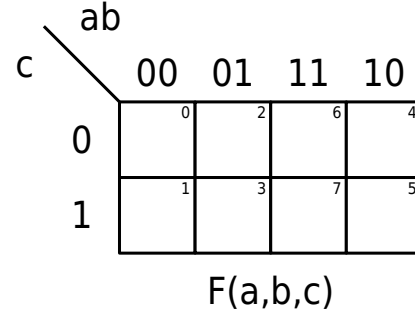
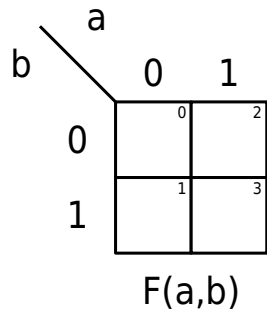
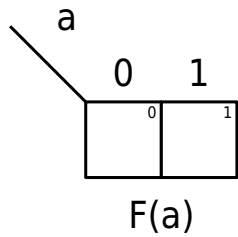
F(a,b,c,d)

$$F(a,b,c,d) = \Sigma(0,1,4,9,11,13,15)$$

	ab			
cd	00	01	11	10
00	1 <sup>0</sup>	1 <sup>4</sup>	0 <sup>12</sup>	0 <sup>8</sup>
01	1 <sup>1</sup>	0 <sup>5</sup>	1 <sup>13</sup>	1 <sup>9</sup>
11	0 <sup>3</sup>	0 <sup>7</sup>	1 <sup>15</sup>	1 <sup>11</sup>
10	0 <sup>2</sup>	0 <sup>6</sup>	0 <sup>14</sup>	0 <sup>10</sup>

F(a,b,c,d)




# K-mapas de 1 a 6 variables



# K-mapas. Ejemplos

	ab			
c	00	01	11	10
0			1	
1	1	1	1	

$f(a,b,c) = ab + \bar{a}c$

-  Implicante primo esencial
-  Implicante primo seleccionado
-  Otros implicantes primos

]



# Minimización SOP/POS (resumen)

---

- Sistemático
- Obtiene expresiones mínimas en dos niveles (+inversores)
- Dos niveles: retraso homogéneo
- Sólo se necesitan algunas puertas básicas: AND, OR, INV
- K-mapa: cálculo manual para funciones de hasta 5 o 6 entradas.
- Quine-McCluskey
  - Sistemático (programable).
  - No es adecuado para funciones de muchas entradas.
  - El tiempo de computación incrementa exponencialmente con el número de entradas (32 entradas  $\rightarrow \sim 10^{15}$  implicantes primos).
  - Alternativa: minimizadores lógicos heurísticos.

# Diseño de circuitos combinatoriales

---

- Comprender la descripción funcional (verbal)
- Definir entradas y salidas digitales: número, significado, etc.
- Descripción formal del problema: tabla de verdad, K-mapa, etc.
- Obtener expresión de la función.
- Simplificar expresión
  - SOP o POS, etc.
- Convertir expresión en circuito:
  - SOP: términos con ADN, suma con OR
  - POS: términos con OR, suma con AND
- Complementos
  - Raíl doble: se dispone de los complementos como entradas
  - Raíl simple: no se dispone de los complementos. Inversores.

# Diseño de circuitos combinacionales

---

## Ejemplo 2

Diseñe un circuito combinacional con cuatro entradas ( $x_3, x_2, x_1, x_0$ ) que representen los bits de una cifra BCD  $X$ , y dos salidas ( $c_1, c_0$ ) que representen los bits de una magnitud  $C$ , donde  $C$  es el cociente de la división  $X/3$ .

Por ejemplo, si  $X=7 \rightarrow C=2$ , esto es, si  $(x_3, x_2, x_1, x_0) = (0, 1, 1, 1) \rightarrow (c_1, c_0) = (1, 0)$

Diseñe un circuito mínimo en dos niveles de puertas, salvo inversores, con entradas en raíl simple.

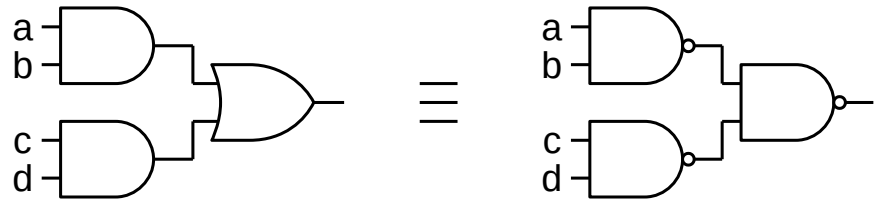
## Ejemplo 3

Diseñe un circuito combinacional para el Ejemplo 1 (Introducción).

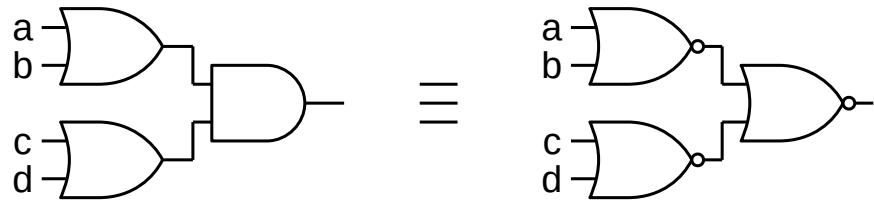
# Alternativas de diseño

- En tecnología CMOS es más fácil hacer NANDs y NORs
- SOP: AND-OR es equivalente a NAND-NAND
- POS: OR-AND es equivalente a NOR-NOR

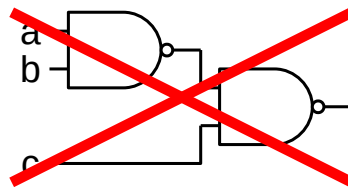
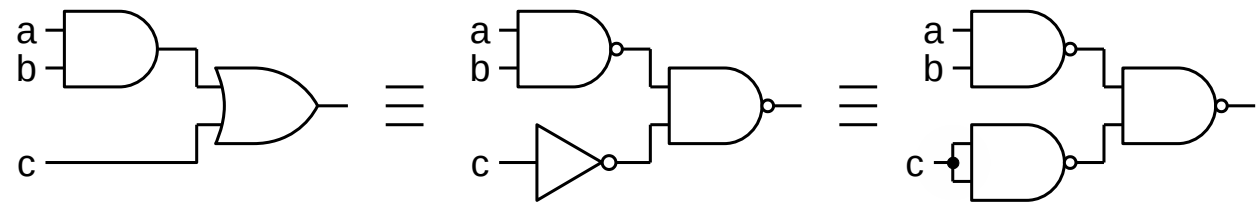
$$ab+cd = \overline{(\overline{ab})(\overline{cd})}$$



$$(a+b)(c+d) = \overline{(\overline{a+b})(\overline{c+d})}$$



$$ab+c = \overline{(\overline{ab})(\overline{c})}$$



# Ejemplo de diseño

---

## Ejemplo 4

Diseñe un circuito combinacional con cuatro entradas ( $x_3, x_2, x_1, x_0$ ) que representan los bits de una cifra BCD  $X$ , y dos salidas ( $c_1, c_0$ ) que representan los bits de una magnitud  $C$ , donde  $C$  es el cociente de la división  $X/3$ .

Por ejemplo, si  $X=7 \rightarrow C=2$ , esto es, si  $(x_3, x_2, x_1, x_0) = (0, 1, 1, 1) \rightarrow (c_1, c_0) = (1, 0)$

- a) Diseñe un circuito mínimo en dos niveles de puertas empleando únicamente puertas NAND.
- b) Diseñe un circuito mínimo en dos niveles de puertas empleando únicamente puertas NOR.

# Inespecificaciones

$$F(a,b,c,d) = \Sigma(0,1,4,9,11,13,15)+d(2,3,5)$$

		ab			
		00	01	11	10
cd	00	1 <sup>0</sup>	1 <sup>4</sup>	0 <sup>12</sup>	0 <sup>8</sup>
	01	1 <sup>1</sup>	- <sup>5</sup>	1 <sup>13</sup>	1 <sup>9</sup>
	11	- <sup>3</sup>	0 <sup>7</sup>	1 <sup>15</sup>	1 <sup>11</sup>
	10	- <sup>2</sup>	0 <sup>6</sup>	0 <sup>14</sup>	0 <sup>10</sup>

F(a,b,c,d)

- Las inespecificaciones son valores de las variables para los que la función no define ningún valor.
- Las inespecificaciones permiten obtener expresiones y circuitos más simples y deben ser identificadas en un problema dado.
- Método:
  - Pueden considerarse “1” para obtener implicantes primos más simples.
  - No es necesario que queden cubiertas por la expresión final.

# Ejemplo de diseño

---

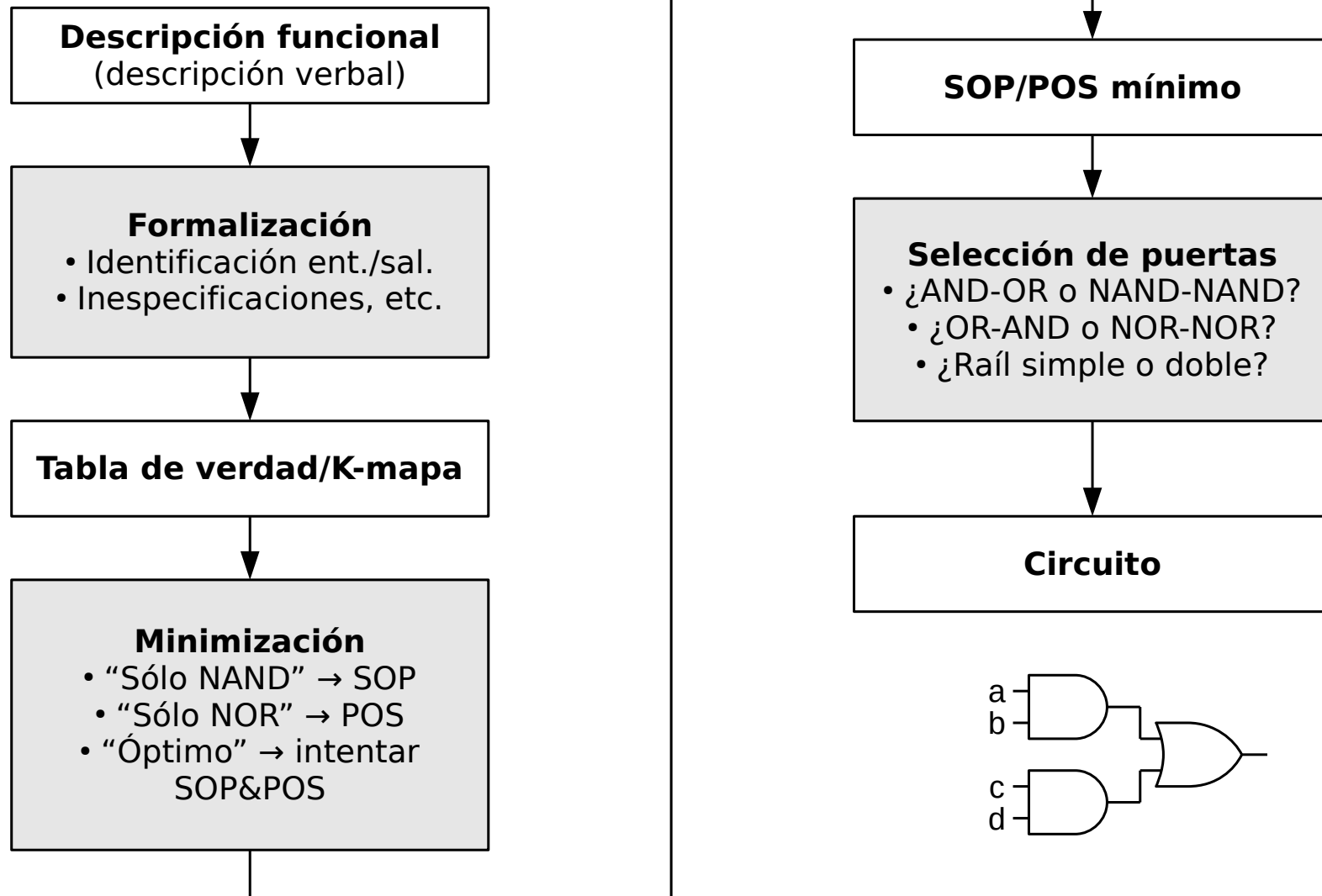
## Ejemplo 5

Diseñe un circuito combinacional con cuatro entradas ( $x_3, x_2, x_1, x_0$ ) que representen los bits de una cifra BCD  $X$ , y dos salidas ( $c_1, c_0$ ) que representen los bits de una magnitud  $C$ , donde  $C$  es el cociente de la división  $X/3$ .

Por ejemplo, si  $X=7 \rightarrow C=2$ , esto es, si  $(x_3, x_2, x_1, x_0) = (0, 1, 1, 1) \rightarrow (c_1, c_0) = (1, 0)$

Diseñe un circuito mínimo en dos niveles de puertas empleando únicamente puertas NAND, considerando posibles inespecificaciones.

# Diseño “manual” de CC (resumen)





# Contenido

---

- Funciones lógicas
- Álgebra de Boole
- Diseño combinacional
- **Análisis funcional**
- Análisis temporal

# Análisis funcional

---

- ¿Qué es?
  - Obtener la función lógica realizada por un circuito combinacional.
- Aplicaciones:
  - Interpretar la operación y/o utilidad del circuito.
  - Rediseñar un circuito equivalente con distintos componentes.
- Método:
  - Identificar entradas y salidas.
  - Para cada puerta con entradas conocidas, calcular la expresión lógica de salida de la puerta.
  - Repetir hasta que se conocen las expresiones de todas las salidas del circuito.
  - Convertir a una representación más útil: tabla de verdad, K-mapa, expresión mínima, etc.
  - Dar una descripción verbal de la operación del circuito.

# Análisis funcional. Ejemplo

## Ejemplo 6

El circuito de la figura corresponde a un sistema de alarma con cuatro entradas y una salida. Las entradas corresponden a:

a: activación del sistema (0 - desactivado, 1 - activado)

b: sensor de fuego (0 - no hay fuego, 1 - sí hay fuego)

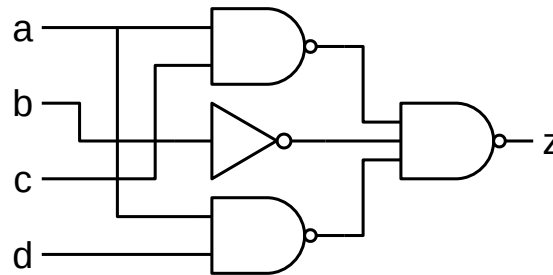
c: sensor de puerta de entrada (0 - puerta cerrada, 1 - puerta abierta)

d: sensor de presencia (0 - no hay presencia, 1 - sí hay presencia)

Cuando la salida z se activa ( $z=1$ ) hace sonar la sirena de la alarma.

a) Analice el circuito y describa su operación con palabras: casos en los que se activará la alarma, etc.

b) Rediseñe el circuito empleando únicamente puertas NOR.



# Contenido

---

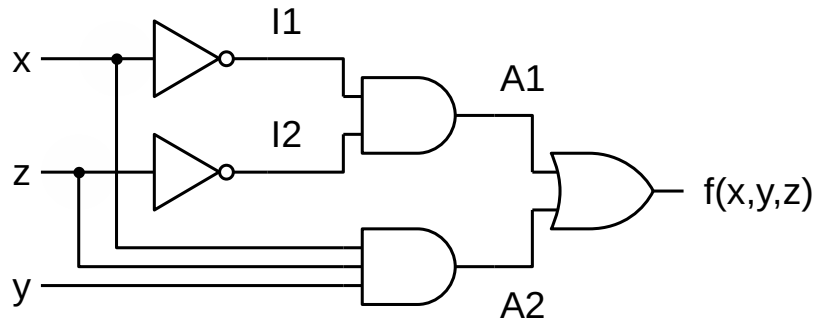
- Funciones lógicas
- Álgebra de Boole
- Diseño combinacional
- Análisis funcional
- **Análisis temporal**

# Análisis temporal

---

- ¿Qué es?
  - Estudiar la evolución con el tiempo de las señales internas de un circuito y de sus salidas para señales de entrada dadas.
  - Forma de onda: representación temporal del valor de una señal.
- Aplicaciones
  - Calcular o estimar el retraso de propagación de un circuito: tiempo que tarda en proporcionar un valor de salida correcto.
  - Analizar posibles fallos o comportamientos no esperados debido a características temporales: retraso excesivo, azares, etc.
- Método “manual”
  - Para cada puerta lógica, obtener la ecuación de salida en función de las entradas de la puerta.
  - Sustituir los valores de las señales de entrada que son constantes (NO SUSTITUIR NADA MÁS)
  - Dibujar las formas de onda desde las entradas primarias hasta las salidas en función de la operación de cada puerta. Aplicar posibles retrasos de cada puerta. Modelo simple: mismo retraso para todas las puertas.

# Análisis temporal. Ejemplo

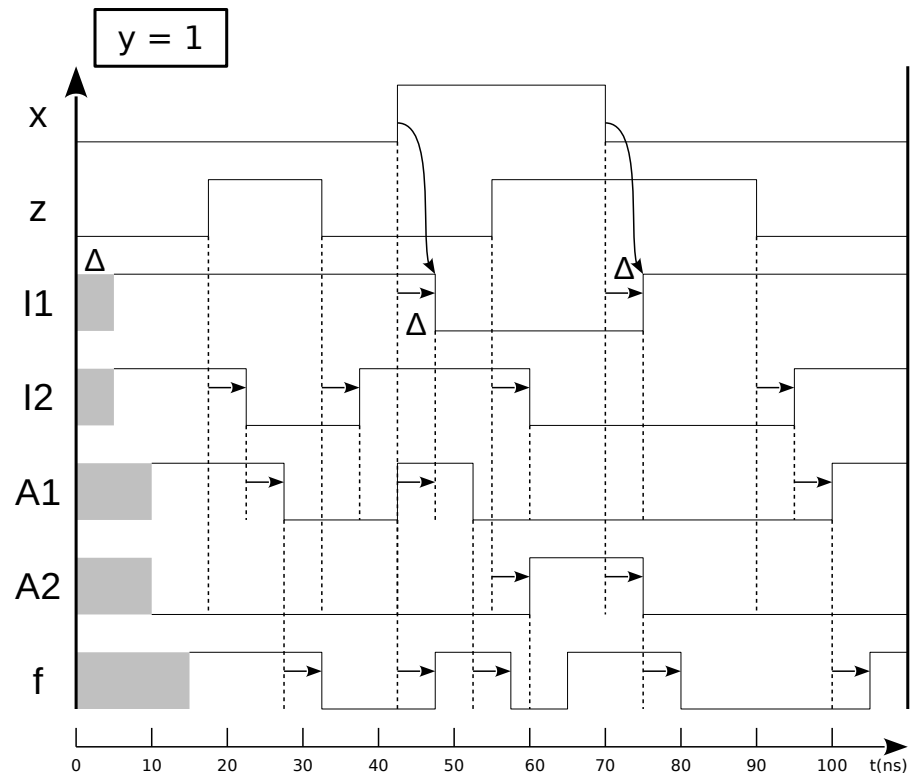


$$\begin{aligned}
 I1 &= \bar{x} \\
 I2 &= \bar{z} \\
 A1 &= I1 I2 \\
 A2 &= x y z \\
 f &= A1 + A2
 \end{aligned}$$

y = 1



$$\begin{aligned}
 I1 &= \bar{x} \\
 I2 &= \bar{z} \\
 A1 &= I1 I2 \\
 A2 &= x z \\
 f &= A1 + A2
 \end{aligned}$$



# Análisis temporal. Azares

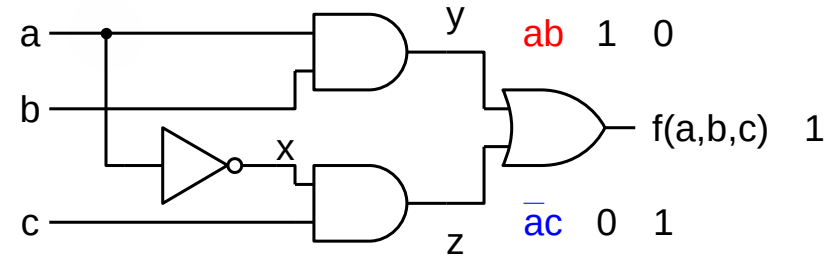
---

- ¿Qué son?
  - Valores **incorrectos** observado de forma **transitoria** a la salida de un circuito combinacional tras un cambio en las entradas.
- Causado por los retraso de propagación de los dispositivos lógicos del circuito.
- Su presencia depende de la estructura interna del circuito.
- No suponen un funcionamiento incorrecto del circuito.
  - Pero pueden causar errores y/o resultados inesperados si no se han tenido en cuenta.
- Son evitables.

# Análisis temporal. Azares

$$f(a,b,c) = ab + \bar{a}c$$

	ab			
c	00	01	11	10
0			1	
1	1	1	1	
	f(a,b,c)			



$$b=c=1; f(a,b,c) = a + \bar{a} = 1$$

