



Tema 2:
Sistemas combinacionales

Fundamentos de computadores

José Manuel Mendías Cuadros
*Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid*



Contenidos

- ✓ Sistema combinacional
- ✓ Expresiones de conmutación.
- ✓ Forma canónica. Suma de productos.
- ✓ Simplificación por mapas de Karnaugh
- ✓ Puertas lógicas: Síntesis con redes AND-OR
- ✓ Análisis de redes de puertas.
- ✓ Módulos combinacionales básicos: Decodificador, mux...
- ✓ ROM
- ✓ Módulos aritméticos

Bibliografía:

- R. Hermida, F. Sánchez y E. del Corral. *Fundamentos de computadores*.
- S.L. Harris y D. M. Harris. *Digital Design and Computer Architecture*.
- D. Gajsky. *Principios de diseño digital*.

Sistemas combinacionales



- La salida en cada instante depende exclusivamente del valor de la entrada en ese instante.
 - En todo momento, a misma entrada, misma salida.



$$z(t_i) = F(x(t_i)), \text{ con } x(t_i) \in E, z(t_i) \in S$$

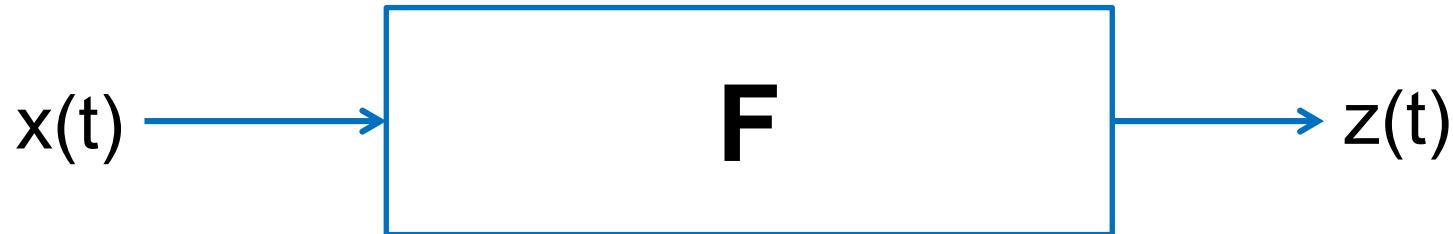
- Para especificar su comportamiento deberán definirse:
 - Los conjuntos discretos de valores de entrada/salida: E, S
 - La función F: E → S



Sistemas combinacionales

versión 2021

tema 2:
Sistemas combinacionales



$$x(t) \in E = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$$

$$z(t) \in S = \{ 0, 1, 2 \}$$

$$F: E \rightarrow S / z(t) = f(x(t)) = x(t) \bmod 3$$

Simulación de su comportamiento:

$x(t)$	0	1	5	1	1	2	8	1	9	0
$z(t)$	0	1	2	1	1	2	2	1	0	0

→ tiempo



Especificación de alto nivel

- **Especificación del dominio:**
 - Conjunto discreto de valores que puede tomar la entrada.
- **Especificación del codominio:**
 - Conjunto discreto de valores que puede tomar la salida.
- **Función de entrada/salida:**
 - Definición del comportamiento del sistema: qué valor toma la salida para cada posible valor de la entrada.
 - Mediante tabla, expresión aritmética, condicional, lógica... o una composición de todas ellas.

Sin embargo, la información debe estar codificada en binario para que sea implementable en un sistema digital

Especificación binaria



- La entrada es un vector de n bits
 - $\underline{x} \in \{0, 1\}^n$ es decir, $\underline{x} = (x_{n-1} \dots x_0)$ con $x_i \in \{0, 1\}$
- La salida es un vector de m bits
 - $\underline{z} \in \{0, 1\}^m$ es decir, $\underline{z} = (z_{m-1} \dots z_0)$ con $z_i \in \{0, 1\}$
- Función de entrada/salida
 - m funciones de conmutación de n variables definiendo cada una el comportamiento de un bit de la salida
 - $F = \{f_i : \{0, 1\}^n \rightarrow \{0, 1\} / z_i = f_i(\underline{x}), \text{ con } 0 \leq i \leq m-1\}$



Descripción binaria



- Proceso de obtener una **especificación binaria** partiendo de una **especificación de alto nivel**:
 1. Codificar el dominio (elegir una representación binaria de cada elemento).
 2. Codificar el codominio.
 3. Traducir la función de E/S.
- Para una misma especificación de alto nivel existen **infinidad** de especificaciones binarias válidas.
 - Cada una **con distinta codificación** del dominio/codominio
- La cardinalidad del dominio/codominio determina la longitud mínima del vector de bits \underline{x} / \underline{z}
 - Para que todos los puntos del dominio/codominio puedan estar representados por una cadena de bits distinta:
 - $n \geq \log_2(|E|)$ y $m \geq \log_2(|S|)$
 - casi siempre quedarán codificaciones sin usar



Descripción binaria

versión 2021

tema 2:
Sistemas combinacionales

FC

8

- Codificación dominio: BCD (4 bits) – usando solo 10 códigos
- Codificación codominio: one-hot (3 bits)
 - $\{ 0 \rightarrow (001), 1 \rightarrow (010), 2 \rightarrow (100) \}$
- Traducción de la función de E/S
 - $F = \{ (0000) \rightarrow (001), (0001) \rightarrow (010), (0010) \rightarrow (100), (0011) \rightarrow (001), (0100) \rightarrow (010), (0101) \rightarrow (100), (0110) \rightarrow (001), (0111) \rightarrow (010), (1000) \rightarrow (100), (1001) \rightarrow (001) \}$

Simulación de su comportamiento:

<u>$x(t)$</u>	0000	0001	0101	0001	0001	0010	1000	0001	1001	0000
<u>$z(t)$</u>	001	010	100	010	010	100	100	010	001	001

→ tiempo

Funciones de conmutación (FC)



- Una **función de conmutación** de n variables es una aplicación
$$f: \{0, 1\}^n \rightarrow \{0, 1\}$$
- Cuando es **total** (todo punto del dominio está asociado a uno del codominio) se dice que está **completamente especificada**
- Se suele definir mediante una **tabla de verdad** que indica el valor que toma la función en cada punto del dominio.

	x_2	x_1	x_0	$f(x_2, x_1, x_0)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1



Funciones de conmutación (FC)

- El número de funciones de conmutación distintas de n variables es finito: 2^{2^n}
 - Para 2 variables existen únicamente 16 distintas

x_1	x_0	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
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

nula and x_1 x_0 XOR or nor nxor $\text{not } x_0$ $\text{not } x_1$ nand unidad



Funciones de conmutación (FC)

- A veces las funciones de conmutación son **parciales** (no están definidas para ciertos puntos del dominio).
 - Típicamente porque existen códigos que no representan ningún valor.
- Una **función de conmutación incompletamente especificada** de n variables es una aplicación:
$$f : \{ 0, 1 \}^n \rightarrow \{ 0, 1, - \}$$
 - Donde '-' (don't care) denota **indiferencia**: da igual que la función valga 0 ó 1 en aquellos puntos del dominio asociados a este valor, dado que esa configuración de entrada nunca se va a producir.

Funciones de conmutación (FC)



	x_3	x_2	x_1	x_0	z_2	z_1	z_0
0	0	0	0	0	0	0	1
1	0	0	0	1	0	1	0
2	0	0	1	0	1	0	0
3	0	0	1	1	0	0	1
4	0	1	0	0	0	1	0
5	0	1	0	1	1	0	0
6	0	1	1	0	0	0	1
7	0	1	1	1	0	1	0
8	1	0	0	0	1	0	0
9	1	0	0	1	0	0	1
10	1	0	1	0	-	-	-
11	1	0	1	1	-	-	-
12	1	1	0	0	-	-	-
13	1	1	0	1	-	-	-
14	1	1	1	0	-	-	-
15	1	1	1	1	-	-	-

}

$E = \{ 0, \dots, 9 \}$
la codificación es BCD

nunca aparecerán estos
códigos



Contenidos

- ✓ Sistema combinacional
- ✓ Expresiones de conmutación.
- ✓ Forma canónica. Suma de productos.
- ✓ Simplificación por mapas de Karnaugh
- ✓ Puertas lógicas: Síntesis con redes AND-OR
- ✓ Análisis de redes de puertas.
- ✓ Módulos combinacionales básicos: Decodificador, mux...
- ✓ ROM
- ✓ Módulos aritméticos



Expresiones de conmutación (EC)

- Forma alternativa de definir FC completamente especificadas
 - Compacta, manipulable y **directamente sintetizable**.
- **Alfabeto:** { x_i , 0, 1, +, ·, $\bar{}$, (,) }
- Variables lógicas: x_i (*puede usarse cualquier letra con o sin subíndice*)
- Constantes: 0, 1
- Operadores : +, ·, $\bar{}$
- Símbolos auxiliares: (,)
- **Reglas de generación:**
 1. Toda variable lógica es una EC válida.
 2. 0 y 1 son EC válidas.
 3. Si A es una EC válida, \bar{A} también lo es.
 4. Si A y B son EC válida, (A), A+B y A·B también lo son.
 5. Solo son EC válidas las generadas usando las reglas 1 a 4.



Expresiones de conmutación (EC)

versión 2021

tema 2:
Sistemas combinacionales

FC

15

- **Semántica:** el álgebra de conmutación { {0,1}, and, or, not}

operador **and**

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

operador **or**

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

operador **not**

x	\bar{x}
0	1
1	0

- **Valor de una EC, E, para una asignación, \underline{a} :** $v(E, \underline{a})$
 - Resultado de sustituir las variables de E por los valores indicados en \underline{a} y realizar las operaciones de acuerdo con el álgebra de conmutación.

$$v(x_2 + \bar{x}_2 \cdot x_1 + x_1 \cdot x_0, (0,1,0)) = 0 + \bar{0} \cdot 1 + 1 \cdot 0 = 0 + 1 \cdot 1 + 1 \cdot 0 = 1$$



Expresiones de conmutación (EC)

- Para una expresión de conmutación dada, el conjunto de todos los pares
$$f = \{ (\underline{a}, v(E, \underline{a})) / \underline{a} \in \{0,1\}^n \}$$
es una función de conmutación.
- En ese caso diremos que *E* representa a *f*
- Dos EC son equivalentes si representan a la misma función de conmutación.
 - Toda FC tiene infinitas EC equivalentes que la representan.
 - Habrá unas más convenientes que otras, en particular las más simples.

Expresiones de conmutación (EC)



$$\overline{x_1} + x_1 \cdot x_0$$

$$v(\overline{x_1} + x_1 \cdot x_0, (0,0)) = \overline{0} + 0 \cdot 0 = 1$$

$$v(\overline{x_1} + x_1 \cdot x_0, (0,1)) = \overline{0} + 0 \cdot 1 = 1$$

$$v(\overline{x_1} + x_1 \cdot x_0, (1,0)) = \overline{1} + 1 \cdot 0 = 0$$

$$v(\overline{x_1} + x_1 \cdot x_0, (1,1)) = \overline{1} + 1 \cdot 1 = 1$$

	x_1	x_0	$f(x_1, x_0)$
0	0	0	1
1	0	1	1
2	1	0	0
3	1	1	1

SON EQUIVALENTES

$$\overline{x_1} + x_0$$

$$v(\overline{x_1} + x_0, (0,0)) = \overline{0} + 0 = 1$$

$$v(\overline{x_1} + x_0, (0,1)) = \overline{0} + 1 = 1$$

$$v(\overline{x_1} + x_0, (1,0)) = \overline{1} + 0 = 0$$

$$v(\overline{x_1} + x_0, (1,1)) = \overline{1} + 1 = 1$$

	x_1	x_0	$f(x_1, x_0)$
0	0	0	1
1	0	1	1
2	1	0	0
3	1	1	1

Sistemas combinacionales



- Estos sistemas se rigen por el álgebra de Boole que tiene las siguientes propiedades:

Propiedad	Versión “+”	Versión “·”
Comutativa	$A + B = B + A$	$A \cdot B = B \cdot A$
Distributiva	$A + (B \cdot C) = (A + B) \cdot (A + C)$	$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$
Elemento neutro	$0 + A = A$	$1 \cdot A = A$
Elem. complementario	$A + \bar{A} = 1$	$A \cdot \bar{A} = 0$
Idempotencia	$A + A = A$	$A \cdot A = A$
Asociativa	$A + (B + C) = (A + B) + C$	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$
Elemento dominante	$1 + A = 1$	$0 \cdot A = 0$
Involución		$\bar{\bar{A}} = A$
Absorción	$A + (A \cdot B) = A$	$A \cdot (A + B) = A$
Leyes de De Morgan	$\overline{A + B} = \bar{A} \cdot \bar{B}$	$\overline{A \cdot B} = \bar{A} + \bar{B}$



Expresiones de conmutación (EC)

- Las anteriores propiedades permiten transformar algebraicamente una EC en otras equivalentes.

$$x_2 \cdot x_1 + x_2 \cdot \overline{x_1} + x_2 \cdot x_0$$



Expresiones de conmutación (EC)

- Las anteriores propiedades permiten transformar algebraicamente una EC en otra equivalente.

$$\begin{aligned} & x_2 \cdot x_1 + x_2 \cdot \overline{x_1} + x_2 \cdot x_0 && \text{distributiva} \\ &= x_2 \cdot (x_1 + \overline{x_1}) + x_2 \cdot x_0 \end{aligned}$$



Expresiones de conmutación (EC)

- Las anteriores propiedades permiten transformar algebraicamente una EC en otra equivalente.

$$\begin{aligned} & x_2 \cdot x_1 + x_2 \cdot \overline{x_1} + x_2 \cdot x_0 && \textit{distributiva} \\ &= x_2 \cdot (x_1 + \overline{x_1}) + x_2 \cdot x_0 && \textit{elem. complementario} \\ &= x_2 \cdot 1 + x_2 \cdot x_0 \end{aligned}$$



Expresiones de conmutación (EC)

- Las anteriores propiedades permiten transformar algebraicamente una EC en otra equivalente.

$$\begin{aligned} & x_2 \cdot x_1 + x_2 \cdot \overline{x_1} + x_2 \cdot x_0 && \textit{distributiva} \\ &= x_2 \cdot (x_1 + \overline{x_1}) + x_2 \cdot x_0 && \textit{elem. complementario} \\ &= x_2 \cdot 1 + x_2 \cdot x_0 && \textit{distributiva} \\ &= x_2 \cdot (1 + x_0) \end{aligned}$$



Expresiones de conmutación (EC)

- Las anteriores propiedades permiten transformar algebraicamente una EC en otra equivalente.

$$\begin{aligned} & x_2 \cdot x_1 + x_2 \cdot \overline{x_1} + x_2 \cdot x_0 && \textit{distributiva} \\ &= x_2 \cdot (x_1 + \overline{x_1}) + x_2 \cdot x_0 && \textit{elem. complementario} \\ &= x_2 \cdot 1 + x_2 \cdot x_0 && \textit{distributiva} \\ &= x_2 \cdot (1 + x_0) && \textit{elem. dominante} \\ & \qquad\qquad\qquad = x_2 \cdot 1 \end{aligned}$$



Expresiones de conmutación (EC)

- Las anteriores propiedades permiten transformar algebraicamente una EC en otra equivalente.

$$x_2 \cdot x_1 + x_2 \cdot \overline{x_1} + x_2 \cdot x_0 \quad \textit{distributiva}$$

$$= x_2 \cdot (x_1 + \overline{x_1}) + x_2 \cdot x_0 \quad \textit{elem. complementario}$$

$$= x_2 \cdot 1 + x_2 \cdot x_0 \quad \textit{distributiva}$$

$$= x_2 \cdot (1 + x_0) \quad \textit{elem. dominante}$$

$$= x_2 \cdot 1 \quad \textit{elem. neutro}$$

$$= x_2$$



Otras operaciones lógicas

versión 2021

tema 2:
Sistemas combinacionales

FC

25

operador **nand**

x	y	$\frac{x \uparrow y}{(x \cdot y)}$
0	0	1
0	1	1
1	0	1
1	1	0

operador **nor**

x	y	$\frac{x \downarrow x}{(x + y)}$
0	0	1
0	1	0
1	0	0
1	1	0

operador **xor**

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

operador **xnor**

x	y	$\frac{(x \oplus y)}{x \cdot y + \bar{x} \cdot \bar{y}}$
0	0	1
0	1	0
1	0	0
1	1	1



Expresiones de conmutación vs. lenguaje natural

- En muchos casos es posible obtener directamente una EC desde un enunciado en lenguaje natural

La barrera debe abrirse si es de día y hay un coche esperando o si el vigilante presiona un pulsador



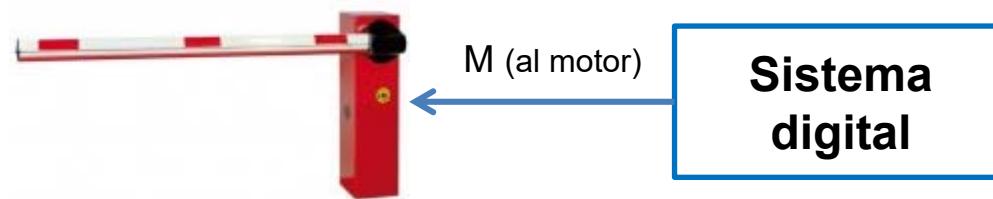
Sistema
digital



EC vs. lenguaje natural

- En muchos casos es posible obtener directamente una EC desde un enunciado en lenguaje natural

La barrera debe abrirse si es de día y hay un coche esperando o si el vigilante presiona un pulsador

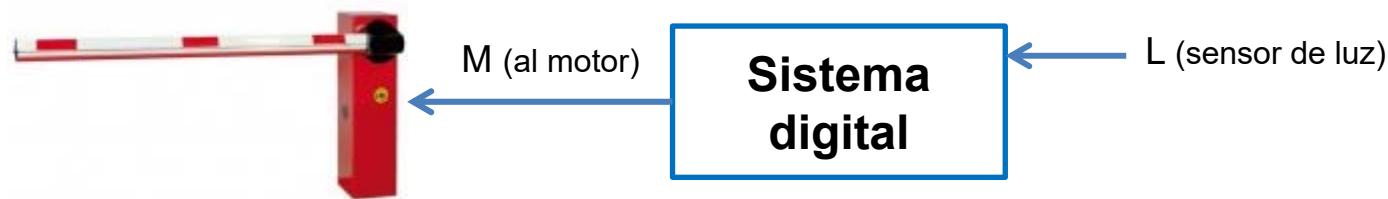


EC vs. lenguaje natural



- En muchos casos es posible obtener directamente una EC desde un enunciado en lenguaje natural

La barrera debe abrirse si es de día y hay un coche esperando o si el vigilante presiona un pulsador



EC vs. lenguaje natural



- En muchos casos es posible obtener directamente una EC desde un enunciado en lenguaje natural

La barrera debe abrirse si es de día y hay un coche esperando o si el vigilante presiona un pulsador



EC vs. lenguaje natural



- En muchos casos es posible obtener directamente una EC desde un enunciado en lenguaje natural

La barrera debe abrirse si es de día y hay un coche esperando o si el vigilante presiona un pulsador





EC vs. lenguaje natural

- En muchos casos es posible obtener directamente una EC desde un enunciado en lenguaje natural

La barrera debe abrirse si es de día y hay un coche esperando o si el vigilante presiona un pulsador



- Codificando los sucesos en "lógica directa"
 - $L=1 \Leftrightarrow$ Se detecta luz (es de día)
 - $P=1 \Leftrightarrow$ Se detecta coche
 - $A=1 \Leftrightarrow$ Se ha presionado el pulsador
 - $M=1 \Leftrightarrow$ Se activa el motor que abre la barrera
- La formulación del enunciado queda:



EC vs. lenguaje natural

- En muchos casos es posible obtener directamente una EC desde un enunciado en lenguaje natural

La barrera debe abrirse si es de día y hay un coche esperando o si el vigilante presiona un pulsador



- Codificando los sucesos en "lógica directa"
 - $L=1 \Leftrightarrow$ Se detecta luz (es de día)
 - $P=1 \Leftrightarrow$ Se detecta coche
 - $A=1 \Leftrightarrow$ Se ha presionado el pulsador
 - $M=1 \Leftrightarrow$ Se activa el motor que abre la barrera
- La formulación del enunciado queda:

$$M = \begin{cases} 1 & \text{si } L=1 \text{ y } P=1 \text{ o } A=1 \\ 0 & \text{en caso contrario} \end{cases}$$

EC vs. lenguaje natural



- En muchos casos es posible obtener directamente una EC desde un enunciado en lenguaje natural

La barrera debe abrirse si es de día y hay un coche esperando o si el vigilante presiona un pulsador



- Codificando los sucesos en "lógica directa"
 - $L=1 \Leftrightarrow$ Se detecta luz (es de día)
 - $P=1 \Leftrightarrow$ Se detecta coche
 - $A=1 \Leftrightarrow$ Se ha presionado el pulsador
 - $M=1 \Leftrightarrow$ Se activa el motor que abre la barrera
- La formulación del enunciado queda:

$$M = \begin{cases} 1 & \text{si } L=1 \text{ y } P=1 \text{ o } A=1 \\ 0 & \text{en caso contrario} \end{cases} \quad \Leftrightarrow \quad M = L \cdot P + A$$



Contenidos

- ✓ Sistema combinacional
- ✓ Expresiones de conmutación.
- ✓ **Forma canónica. Suma de productos.**
- ✓ Simplificación por mapas de Karnaugh
- ✓ Puertas lógicas: Síntesis con redes AND-OR
- ✓ Análisis de redes de puertas.
- ✓ Módulos combinacionales básicos: Decodificador, mux...
- ✓ ROM
- ✓ Módulos aritméticos

Suma de productos canónica



- **Literal:** EC compuesta por una única variable natural o complementada.

$$\overline{x_0} \quad x_1$$

- **Término producto:** EC compuesta únicamente por un producto de literales.

$$x_1 \cdot x_0 \quad \overline{x_1} \cdot x_0 \cdot x_1 \cdot x_2$$

- **Mintérmino de n variables:** término producto de n literales, en donde cada variable aparece una y solo una vez.

$$\overline{x_1} \cdot x_0 \quad \overline{x_2} \cdot x_1 \cdot x_0$$

- **Suma de productos (SP):** EC compuesta únicamente por sumas de términos producto.

$$x_1 \cdot \overline{x_0} \quad x_2 \cdot \overline{x_1} + x_2 \cdot \overline{x_1} \cdot \overline{x_0}$$



Suma de productos canónica

- Notación: Un **mintérmino de n variables** se representará por m_i o $m(i)$, siendo i el número cuya representación binaria se obtiene sustituyendo en el mintérmino ordenado (variables de mayor a menor peso):
 - Cada variable complementada por un 0.
 - Cada variable sin complementar por un 1

$$e(x_3, x_2, x_1, x_0) = \overline{x_3} \cdot x_2 \cdot \overline{x_1} \cdot x_0 = m_5 = m(5)$$
$$(0 \quad 1 \quad 0 \quad 1)_2 = 5_{10}$$

$$e(x_3, x_2, x_1, x_0) = \overline{x_3} \cdot x_2 \cdot x_1 \cdot x_0 = m_7 = m(7)$$
$$(0 \quad 1 \quad 1 \quad 1)_2 = 7_{10}$$

- **Suma de productos canónica (SPC):** EC compuesta únicamente por sumas de mintérminos en la que no hay mintérminos repetidos.
- Cada FC tiene una única SPC

Suma de productos canónica



$$e(x_2, x_1, x_0) = \sum m(7, 3, 1)$$

$$x_2 \cdot x_1 \cdot x_0 + \overline{x_2} \cdot x_1 \cdot x_0 + \overline{x_2} \cdot \overline{x_1} \cdot x_0$$

	x_2	x_1	x_0	m_7	m_3	m_1	$m_7 + m_3 + m_1$
0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1
2	0	1	0	0	0	0	0
3	0	1	1	0	1	0	1
4	1	0	0	0	0	0	0
5	1	0	1	0	0	0	0
6	1	1	0	0	0	0	0
7	1	1	1	1	0	0	1



Equivalencia entre EC

- Método 1:
 - Evaluando la EC punto a punto hasta obtener su tabla de verdad.
- Método 2:
 - Transformando la EC en una suma de productos canónica:
 - Aplicando ley de De Morgan
 - Aplicando la distributividad
 - Multiplicando cada término producto que no contenga una cierta variable x_1 por x_1 más su complemento y aplicando distributividad
 - Eliminando los minterminos repetidos



Conversión de una EC a su SPC

	x_2	x_1	x_0	x_1x_0	$\overline{(x_1x_0)}$	$x_2\overline{(x_1x_0)}$	$x_2\overline{(x_1x_0)} + x_1x_0$
0	0	0	0	0	1	0	0
1	0	0	1	0	1	0	0
2	0	1	0	0	1	0	0
3	0	1	1	1	0	0	1
4	1	0	0	0	1	1	1
5	1	0	1	0	1	1	1
6	1	1	0	0	1	1	1
7	1	1	1	1	0	0	1



Conversión de una EC a su SPC

$$e(x_2, x_1, x_0) = x_2 \overline{(x_1 x_0)} + x_1 x_0$$

	x_2	x_1	x_0	$x_1 x_0$	$\overline{(x_1 x_0)}$	$x_2 \overline{(x_1 x_0)}$	$x_2 \overline{(x_1 x_0)} + x_1 x_0$
0	0	0	0	0	1	0	0
1	0	0	1	0	1	0	0
2	0	1	0	0	1	0	0
3	0	1	1	1	0	0	1
4	1	0	0	0	1	1	1
5	1	0	1	0	1	1	1
6	1	1	0	0	1	1	1
7	1	1	1	1	0	0	1

$$\sum m(3, 4, 5, 6, 7)$$



Contenidos

- ✓ Sistema combinacional
- ✓ Expresiones de conmutación.
- ✓ Forma canónica. Suma de productos.
- ✓ Simplificación por mapas de Karnaugh
- ✓ Puertas lógicas: Síntesis con redes AND-OR
- ✓ Análisis de redes de puertas.
- ✓ Módulos combinacionales básicos: Decodificador, mux...
- ✓ ROM
- ✓ Módulos aritméticos



Mapas de Karnaugh

- Mapa de Karnaugh: tabla de verdad de doble entrada que permite obtener de manera gráfica una EC mínima en forma de suma de productos que la represente.
 - EC mínima que tenga el menor número de términos producto y éstos el menor número de literales.
- Un mapa de Karnaugh de n variables tiene las siguientes propiedades:
 - Como la tabla de verdad que es, tiene 2^n casillas cada una de ellas asociada a un mintérmino.
 - Los mintérminos asociados a casillas adyacentes solo se diferencian en la polaridad de una de las variables.
 - Dos mintérminos adyacentes pueden representarse por un término producto en donde no aparece la variable con diferente polaridad.



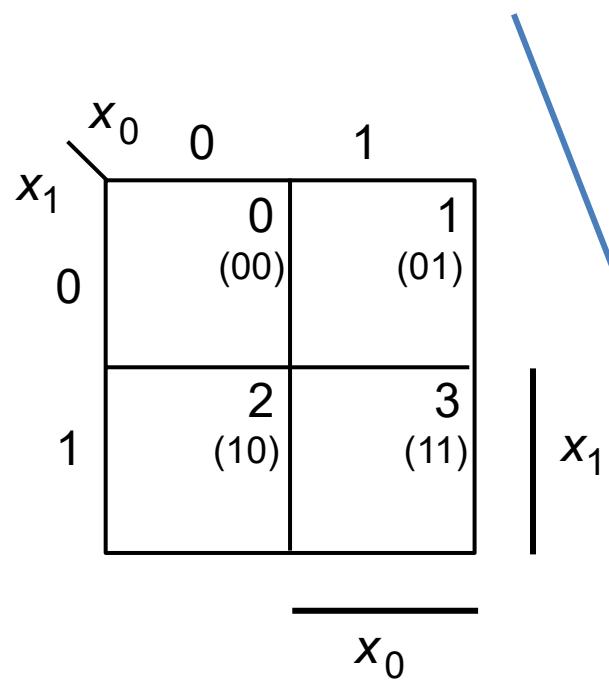
Mapas de Karnaugh

versión 2021

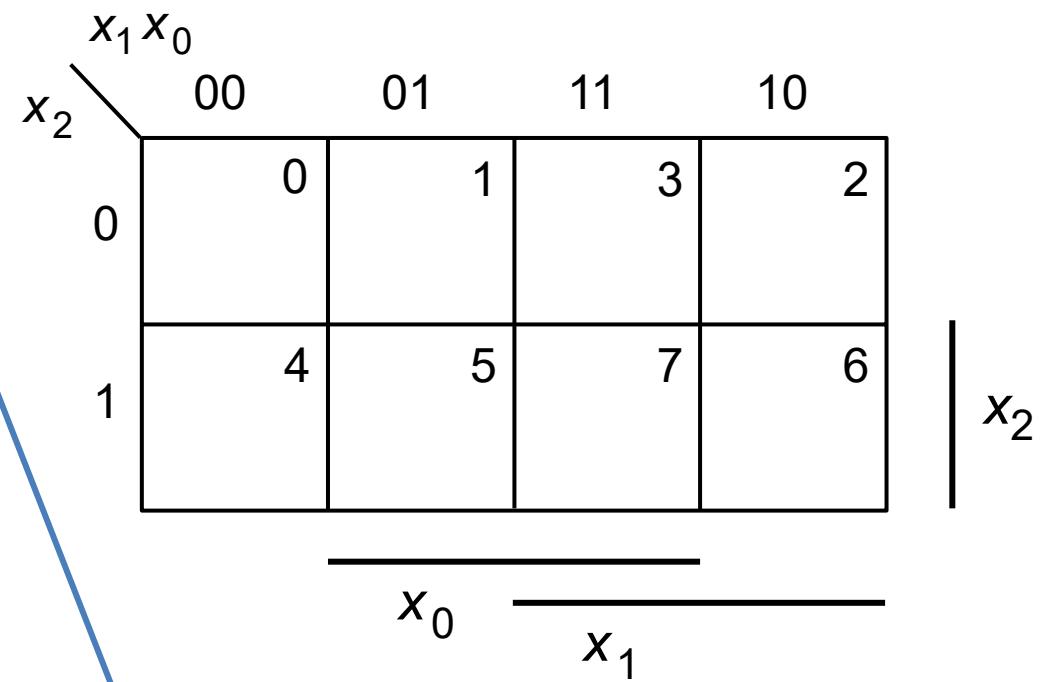
tema 2:
Sistemas combinacionales

FC

43



2 variables

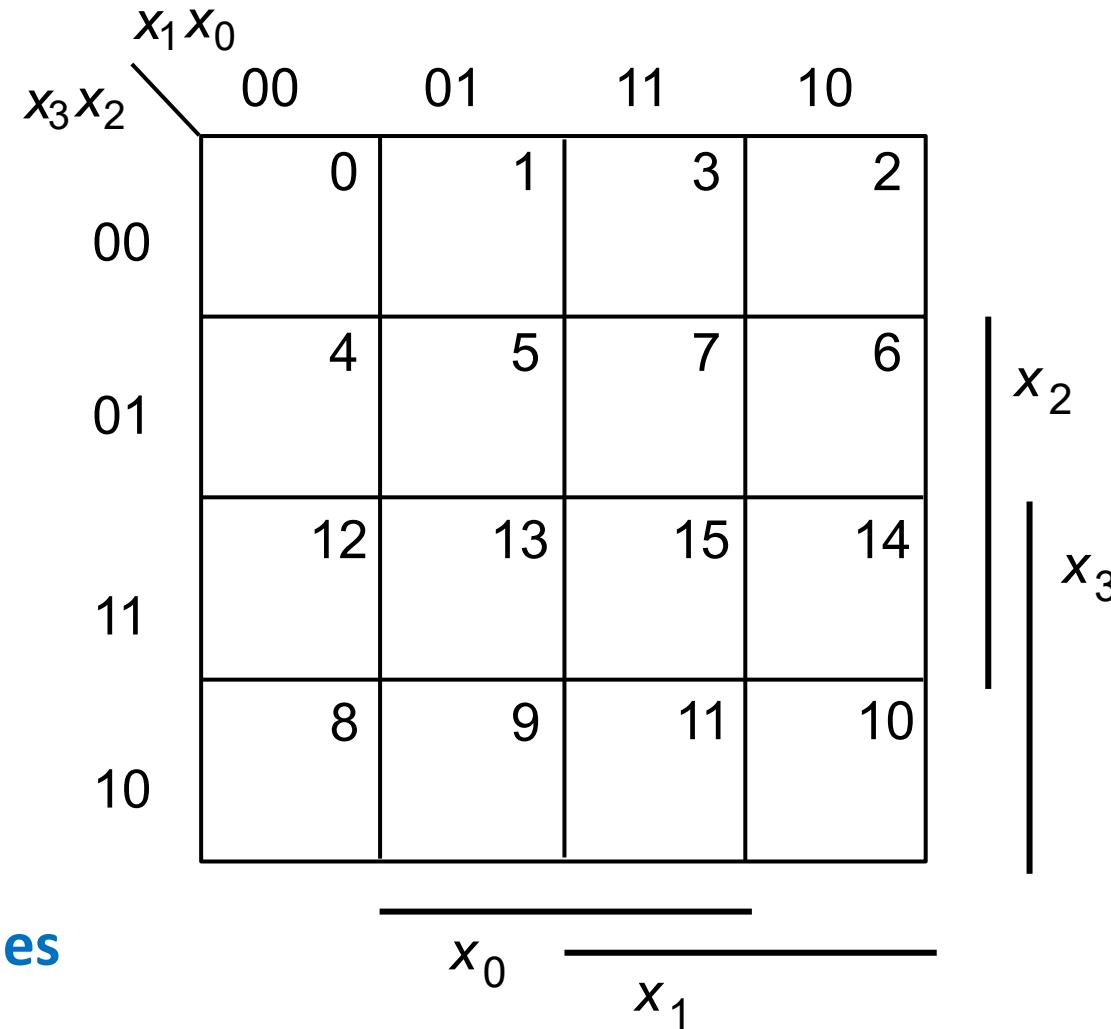


3 variables



Mapas de Karnaugh

4 variables





Mapas de Karnaugh

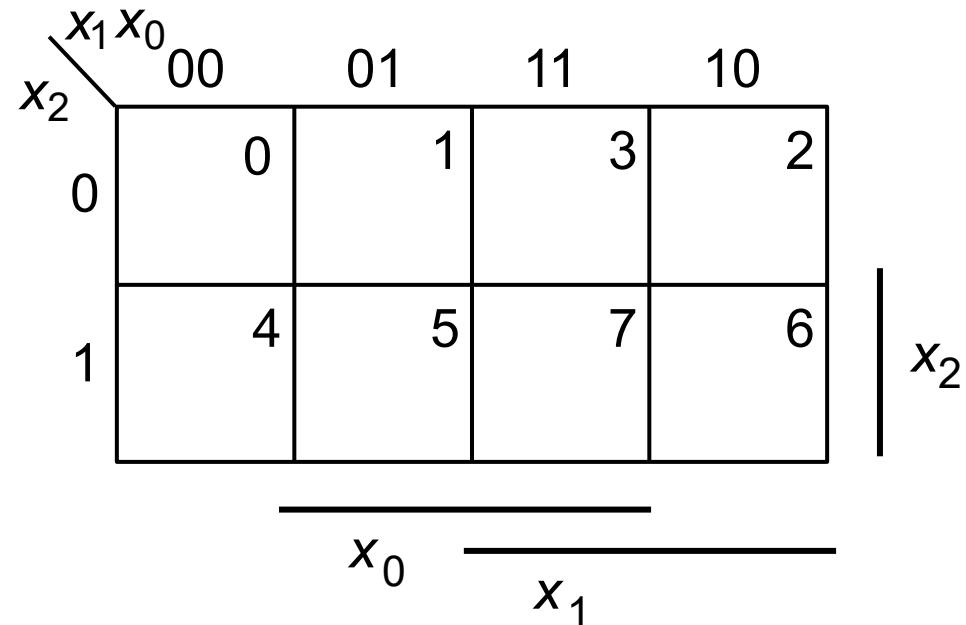
- Para obtener el mapa de Karnaugh de una SPC basta con marcar los mintérminos que la forman.

Mapas de Karnaugh



- Para obtener el mapa de Karnaugh de una SPC basta con marcar los mintérminos que la forman.

$$f(x_2, x_1, x_0) = \Sigma m(0, 3, 7)$$

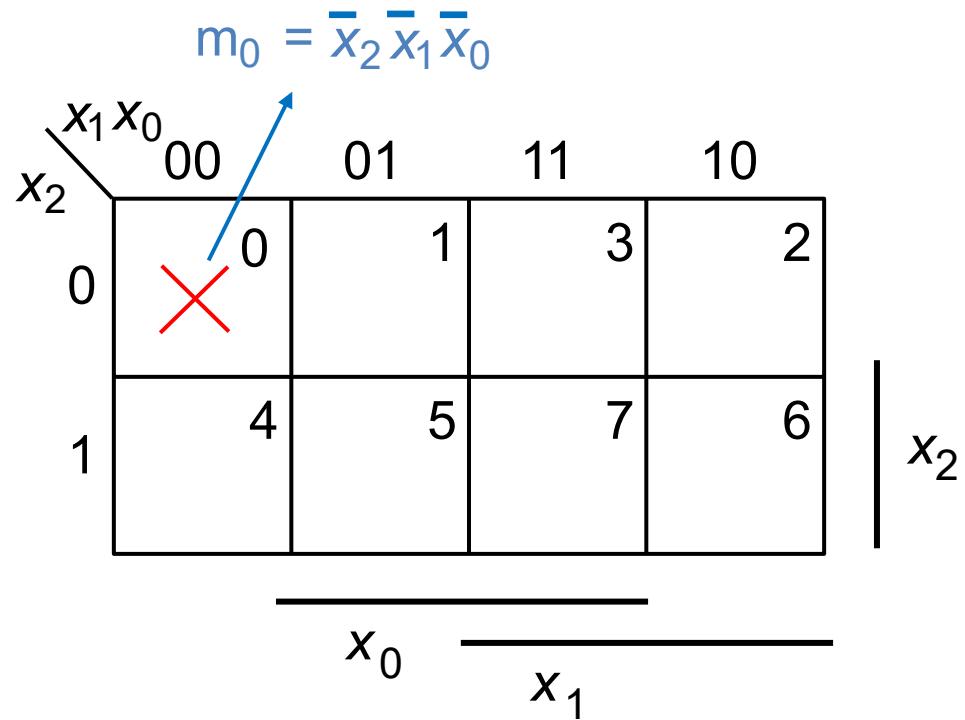


Mapas de Karnaugh



- Para obtener el mapa de Karnaugh de una SPC basta con marcar los mintérminos que la forman.

$$f(x_2, x_1, x_0) = \sum m(0, 3, 7)$$

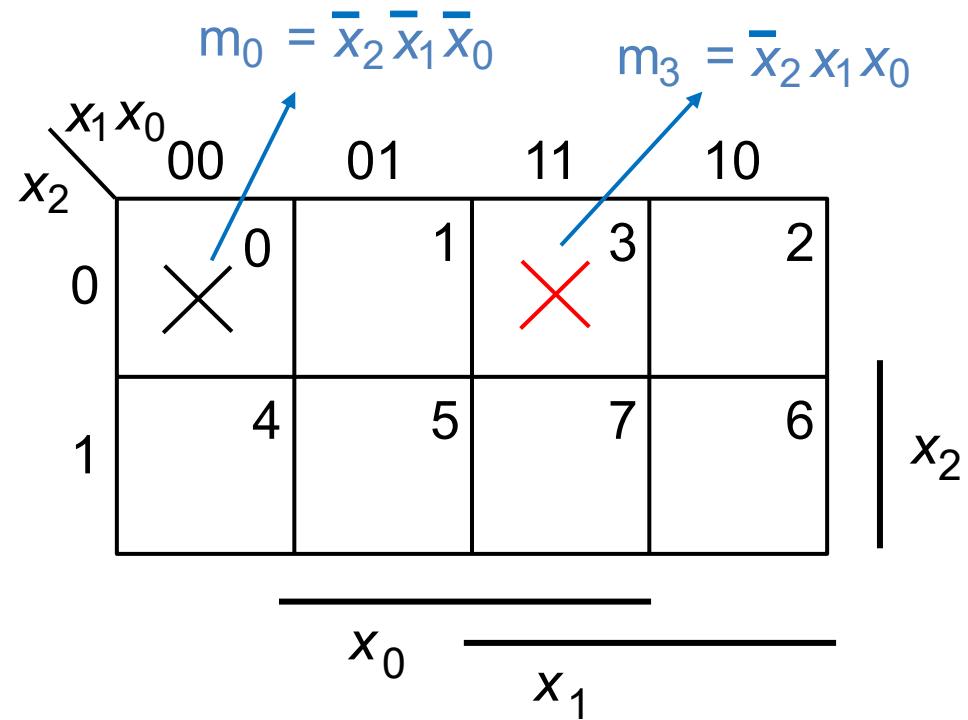


Mapas de Karnaugh



- Para obtener el mapa de Karnaugh de una SPC basta con marcar los mintérminos que la forman.

$$f(x_2, x_1, x_0) = \sum m(0, 3, 7)$$

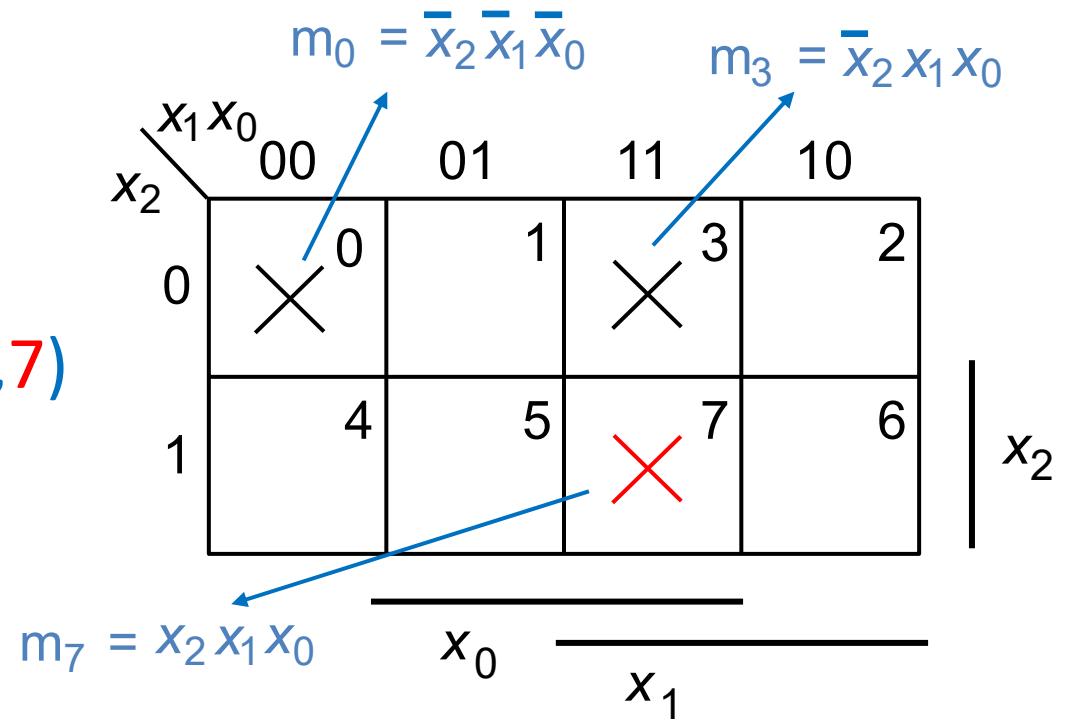




Mapas de Karnaugh

- Para obtener el mapa de Karnaugh de una SPC basta con marcar los mintérminos que la forman.

$$f(x_2, x_1, x_0) = \sum m(0, 3, 7)$$

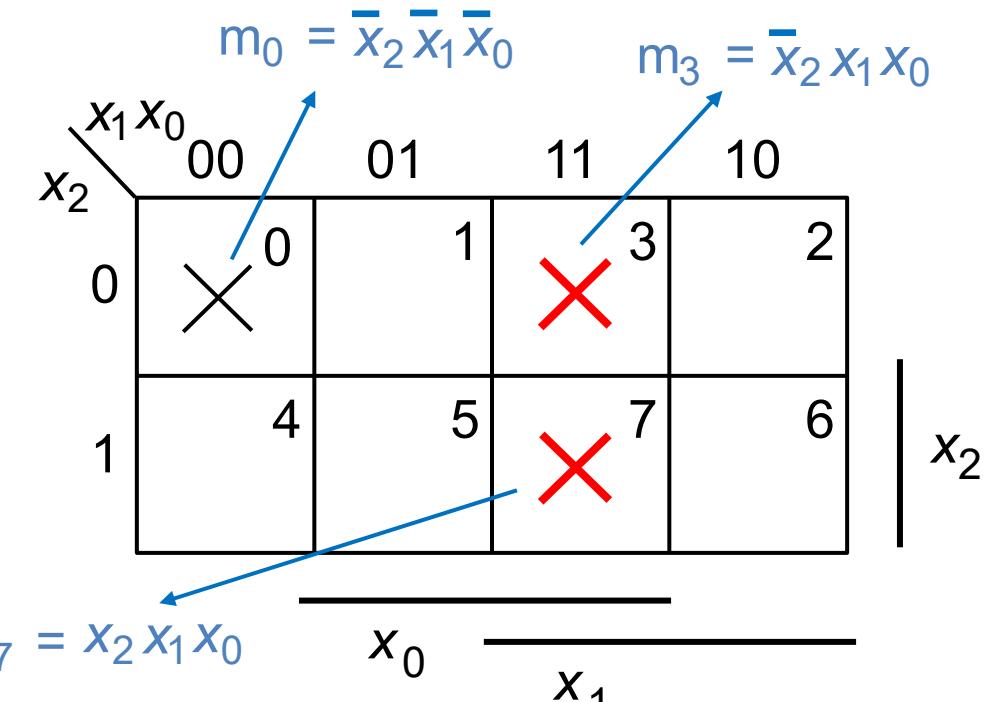


Mapas de Karnaugh



- Para obtener el mapa de Karnaugh de una SPC basta con marcar los mintérminos que la forman.

$$f(x_2, x_1, x_0) = \sum m(0, 3, 7)$$



- m_3 y m_7 son adyacentes luego:

$$m_3 + m_7 = \bar{x}_2 x_1 \bar{x}_0 + x_2 x_1 x_0 = (\bar{x}_2 + x_2)x_1 \bar{x}_0 = x_1 \bar{x}_0$$



Simplificación por MK

- Procedimiento de simplificación:
 - Construir el mapa de Karnaugh de la FC
 - Cubrir todos los mintérminos con el menor número posible de rectángulos de tamaño en casillas múltiplo de 2 (1, 2, 4, 8, 16...)
 - Cada rectángulo se corresponde con un término producto, más simple conforme mayor es el rectángulo.
 - La EC simplificada será la suma de los términos producto obtenidos.
 - Si hay *don't cares*, pueden tomarse como 0 ó 1 según convenga



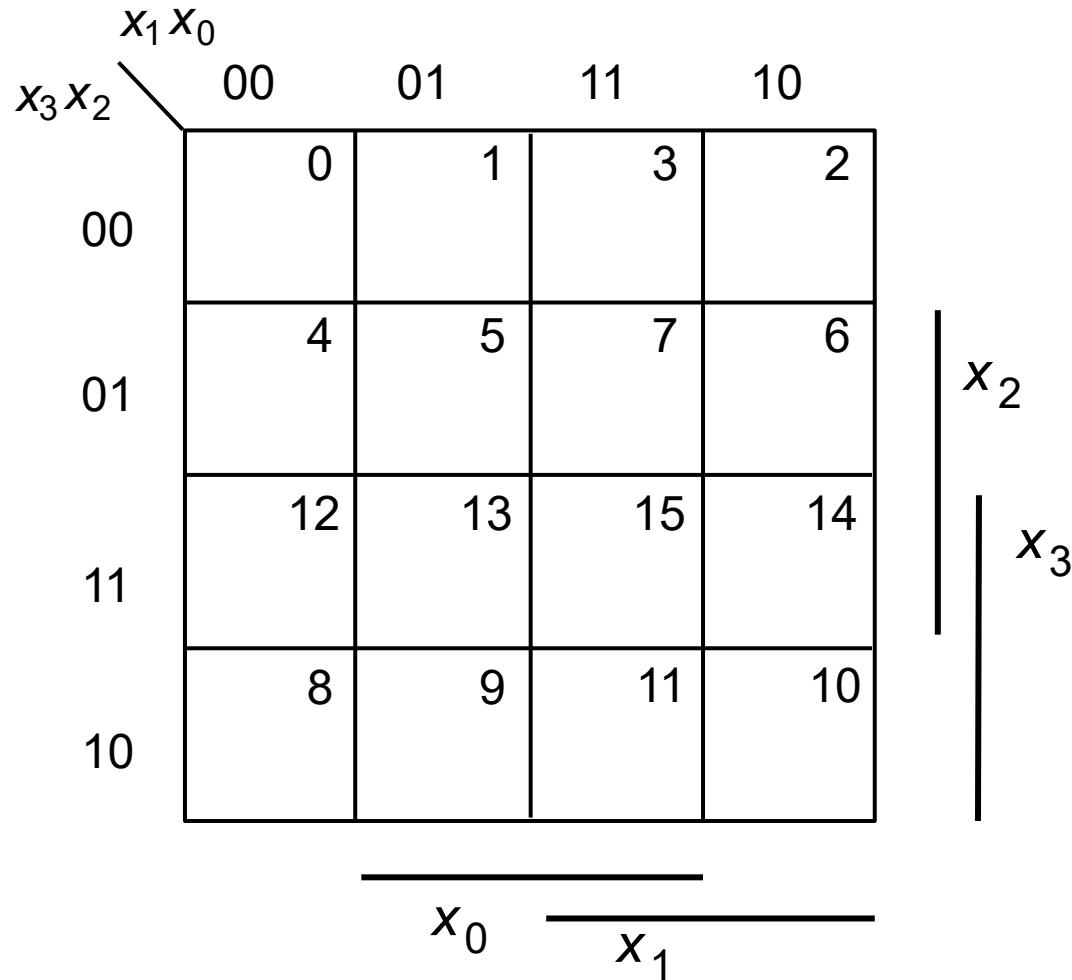
Simplificación por MK

■ Estrategias:

- Los rectángulos deberán ser lo mayor posible, así los términos producto tendrán un menor número de literales.
- Si es necesario, una misma casilla puede ser cubierta varias veces por distintos rectángulos (para que éstos puedan ser más grandes).
- Si una casilla puede cubrirse de distintos modos, empezar cubriendo aquellas que solo puedan hacerlo de una manera.
- Las casillas frontera pueden cubrirse junto con las del otro extremo.
- Las casillas de las esquinas pueden cubrirse todas juntas.



Simplificación por MK





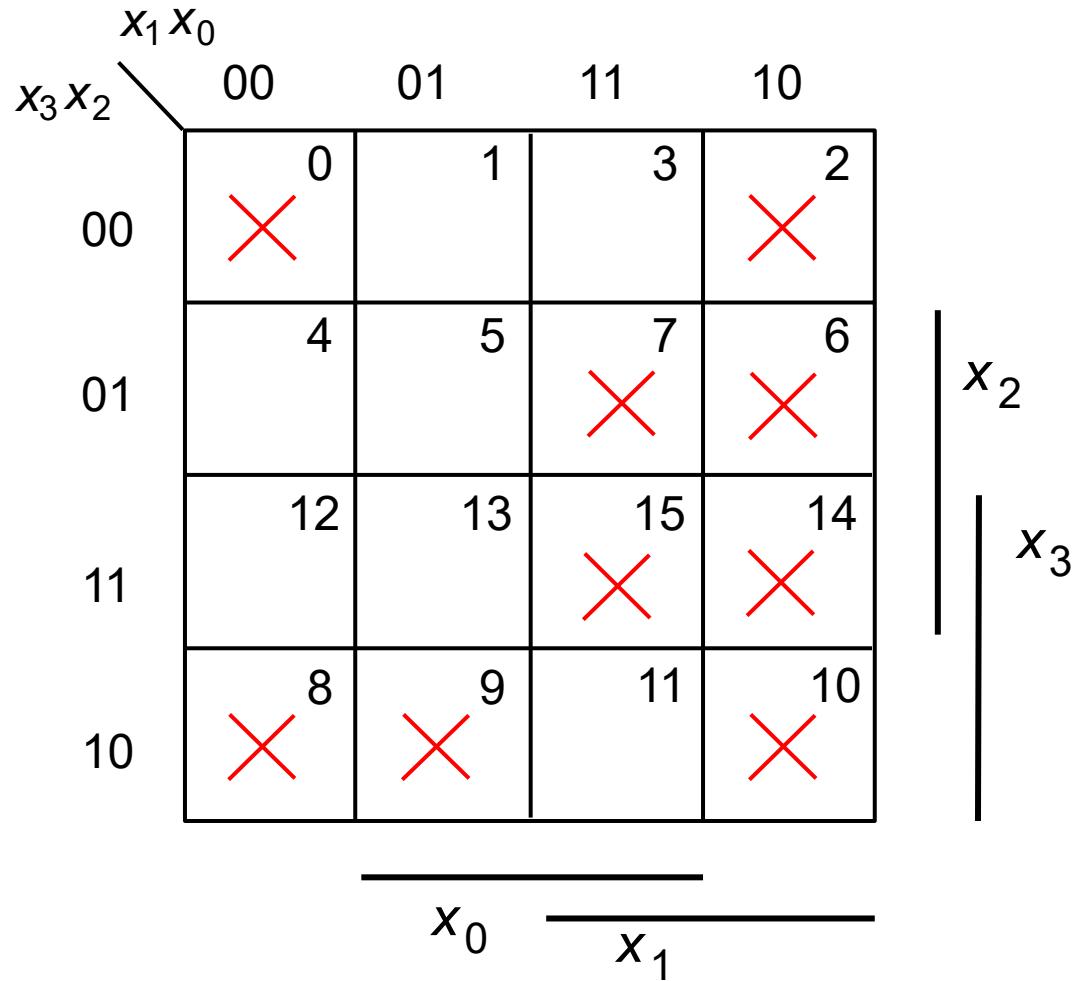
Simplificación por MK

versión 2021

tema 2:
Sistemas combinacionales

FC

54





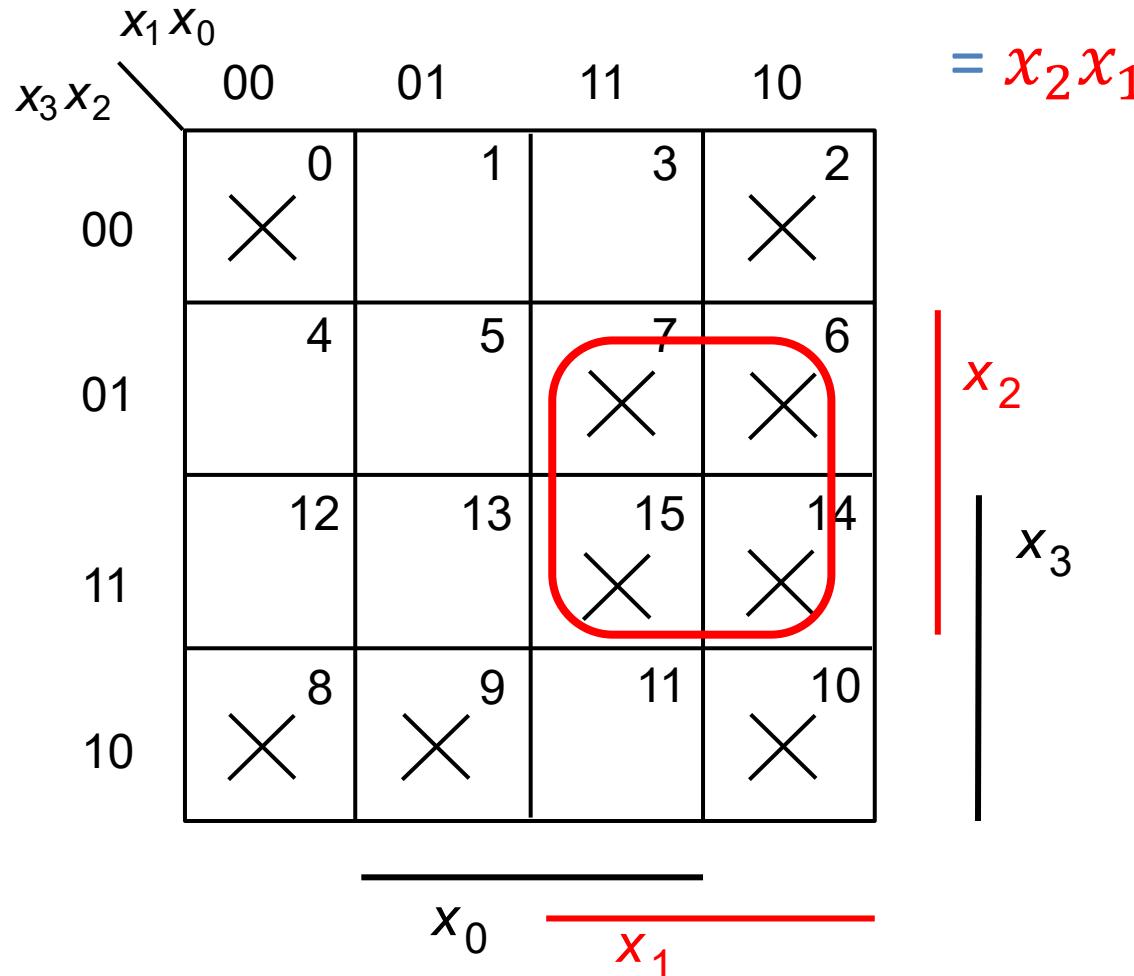
Simplificación por MK

versión 2021

tema 2:
Sistemas combinacionales

FC

55





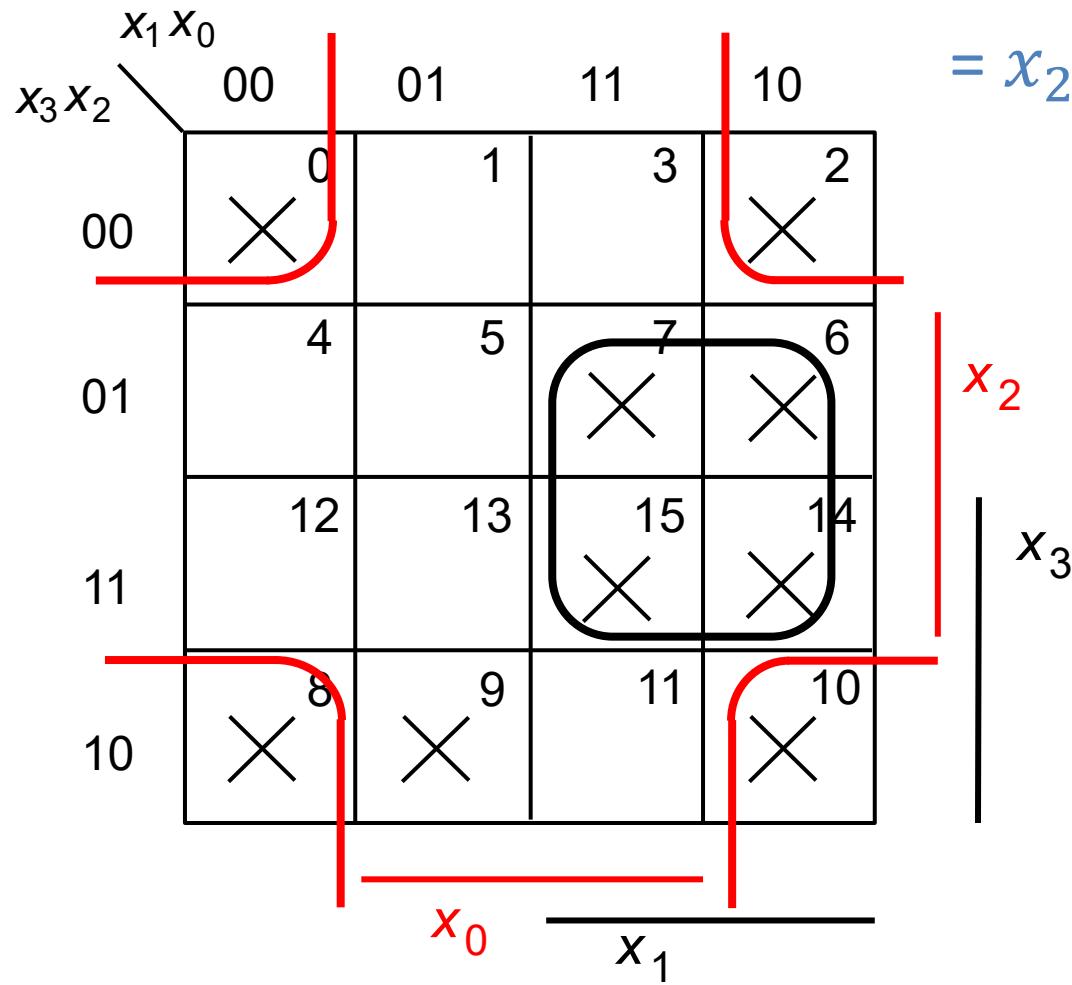
Simplificación por MK

versión 2021

tema 2:
Sistemas combinacionales

FC

56



$$f(x_3, x_2, x_1, x_0) = \sum m(0, 2, 6, 7, 8, 9, 10, 14, 15)$$

$$= x_2 x_1 + \bar{x}_2 \bar{x}_0$$



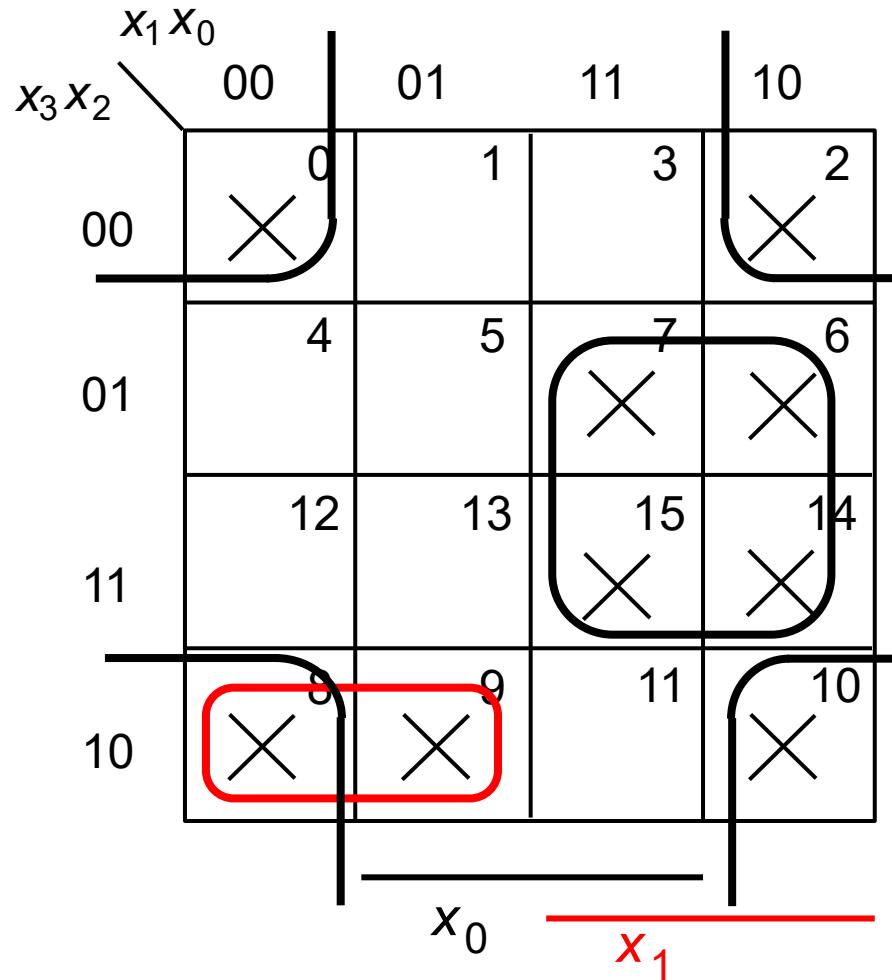
Simplificación por MK

versión 2021

tema 2:
Sistemas combinacionales

FC

57



$$f(x_3, x_2, x_1, x_0) = \sum m(0, 2, 6, 7, 8, 9, 10, 14, 15)$$

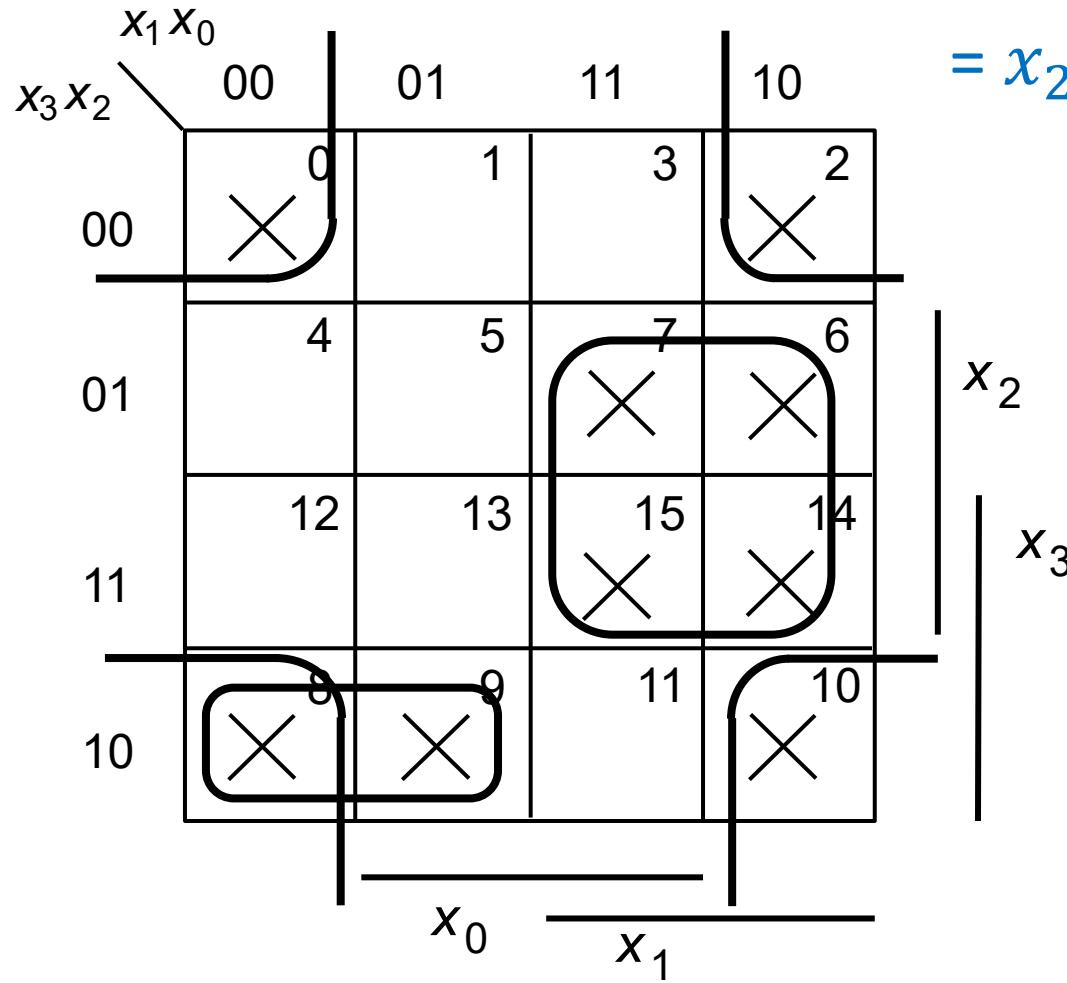
$$= x_2 x_1 + \overline{x_2} \overline{x_0} + x_3 \overline{x_2} \overline{x_1}$$

x_2

x_3

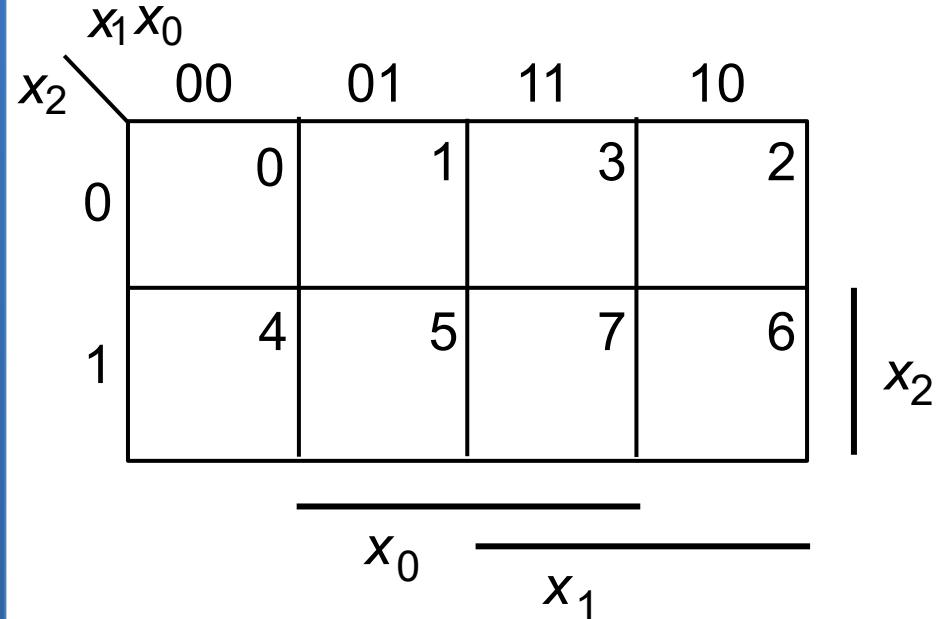


Simplificación por MK



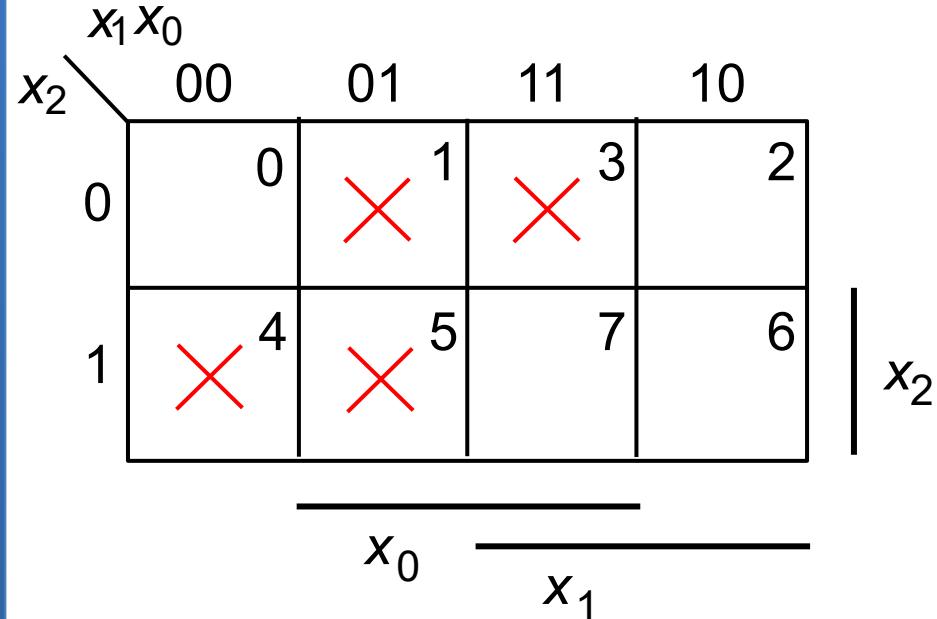


Simplificación por MK





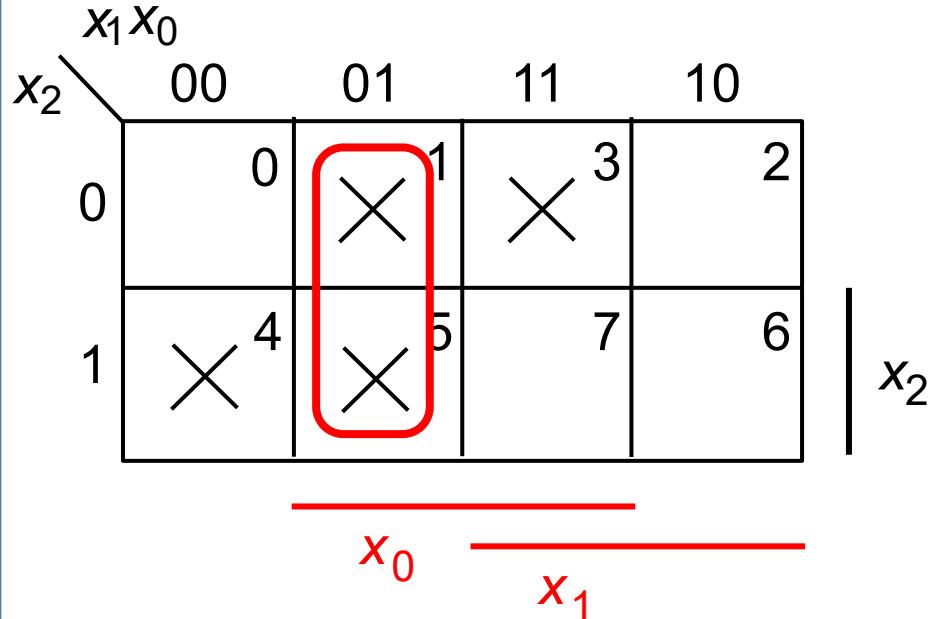
Simplificación por MK



$$f(x_2, x_1, x_0) = \sum m(1, 3, 4, 5)$$



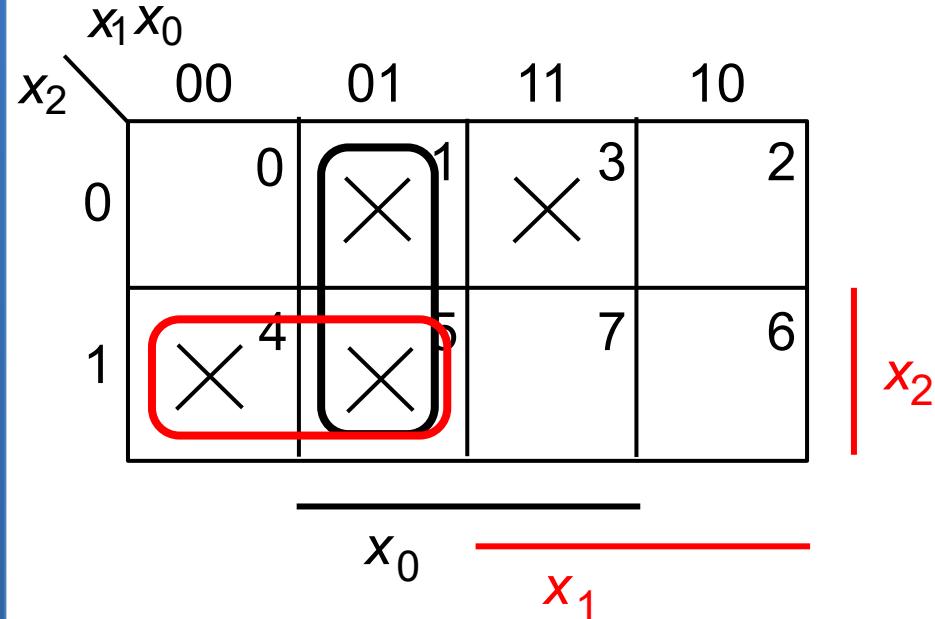
Simplificación por MK



$$= \overline{x_1}x_0$$



Simplificación por MK

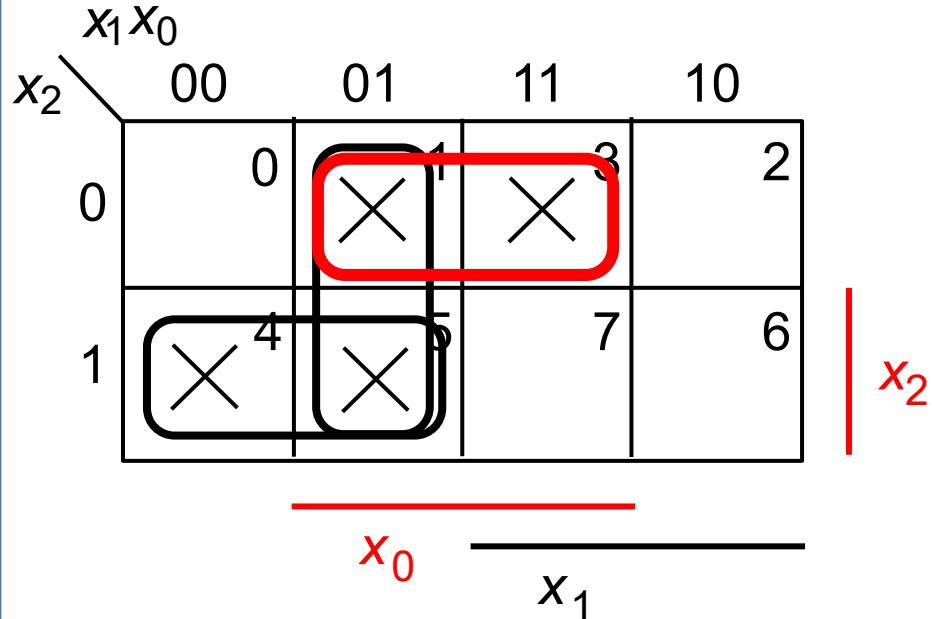


$$f(x_2, x_1, x_0) = \sum m(1, 3, 4, 5)$$

$$= \overline{x_1}x_0 + x_2\overline{x_1}$$



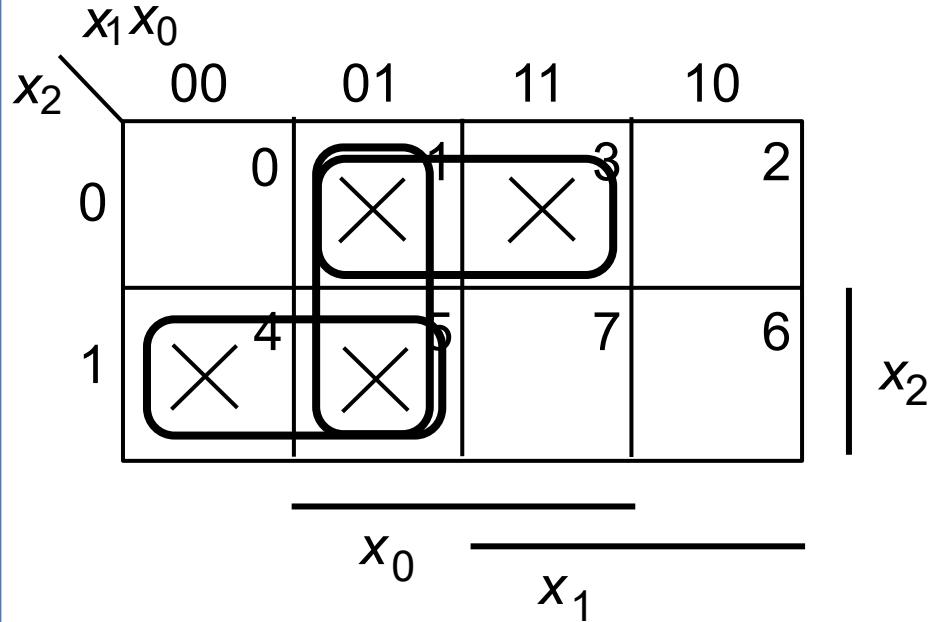
Simplificación por MK



$$= \overline{x_1}x_0 + x_2\overline{x_1} + \overline{x_2}x_0$$



Simplificación por MK

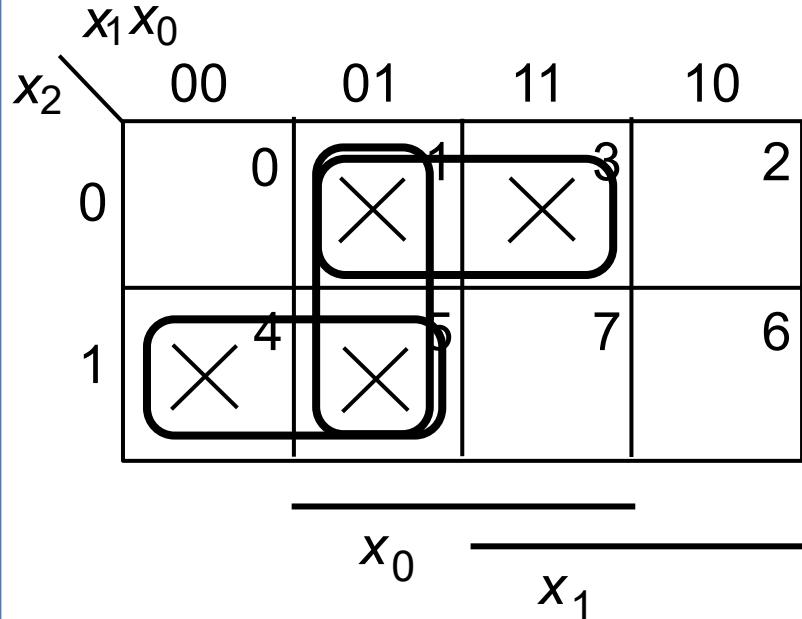


$$f(x_2, x_1, x_0) = \sum m(1, 3, 4, 5)$$

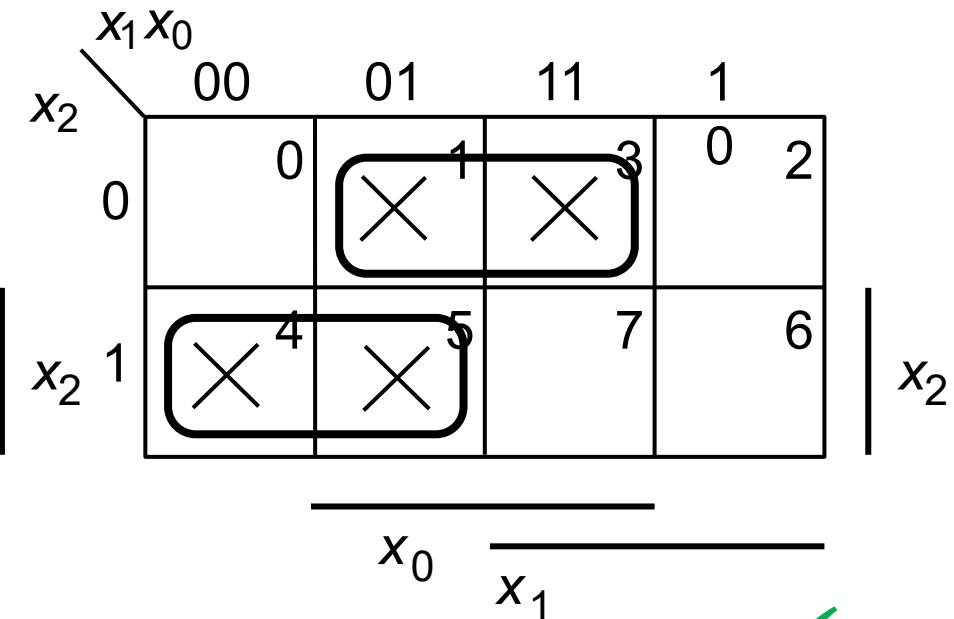
$$= \overline{x_1}x_0 + x_2\overline{x_1} + \overline{x_2}x_0$$



Simplificación por MK



$$= \overline{x}_1x_0 + x_2\overline{x}_1 + \overline{x}_2x_0 \quad \text{X}$$



$$= x_2\overline{x}_1 + \overline{x}_2x_0 \quad \checkmark$$



Contenidos

- ✓ Sistema combinacional
- ✓ Expresiones de conmutación.
- ✓ Forma canónica. Suma de productos.
- ✓ Simplificación por mapas de Karnaugh
- ✓ **Puertas lógicas: Síntesis con redes AND-OR**
- ✓ Análisis de redes de puertas.
- ✓ Módulos combinacionales básicos: Decodificador, mux...
- ✓ ROM
- ✓ Módulos aritméticos

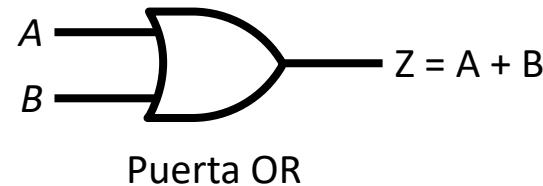
Puertas lógicas



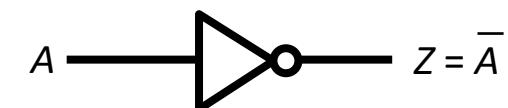
- Dispositivo que realiza **físicamente** una función de commutación **sencilla**.



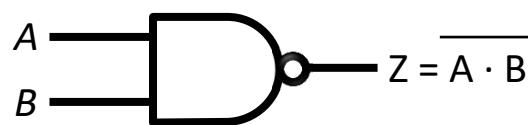
Puerta AND



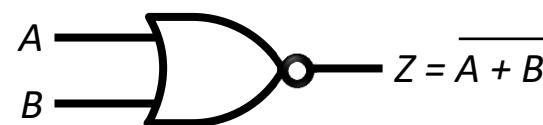
Puerta OR



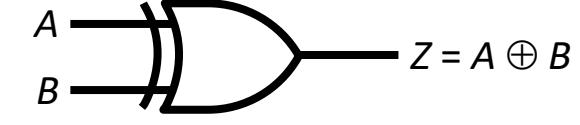
Puerta NOT (Inversor)



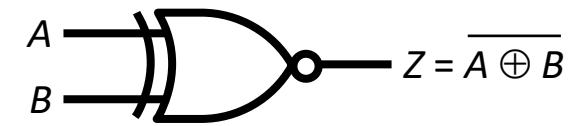
Puerta NAND



Puerta NOR



Puerta XOR

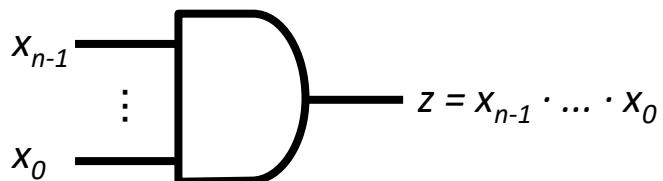


Puerta XNOR

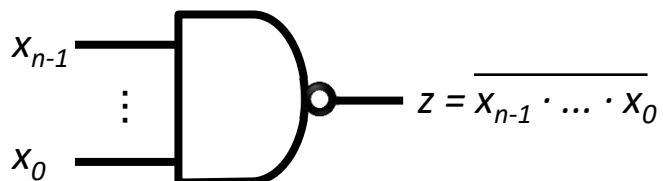
Puertas lógicas



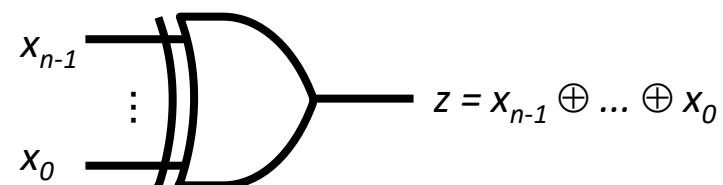
- Existen puertas con mayor número de entradas:



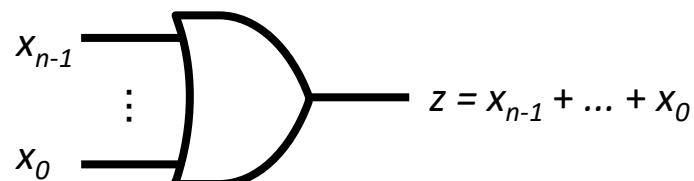
Puerta AND de n entradas



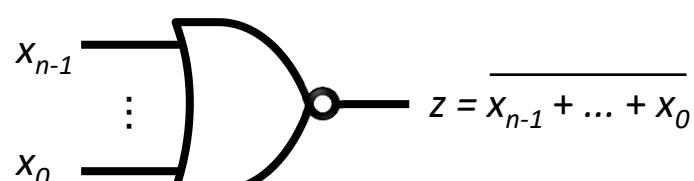
Puerta NAND de n entradas



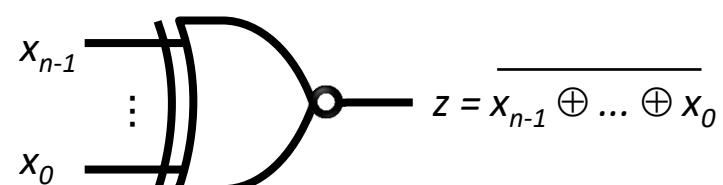
Puerta XOR de n entradas
($z=1$ si el número de $x_i=1$ es impar)



Puerta OR de n entradas



Puerta NOR de n entradas

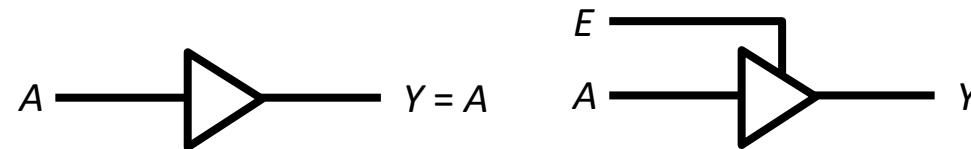


Puerta XNOR de n entradas
($z=1$ si el número de $x_i=1$ es par)



Buffers

- Existen otros dispositivos sin funcionalidad lógica:
 - **Buffer no inversor**: permite compensar la atenuación eléctrica de una señal.
 - **Buffer triestado**: permite desconectar selectivamente una señal.



A	Y
0	0
1	1

E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

Alta impedancia
(desconecta Y de A)

Algunas definiciones



- **Módulo:** dispositivo que realiza físicamente una función conocida de cualquier complejidad.
 - Los hay combinacionales y secuenciales
- **Puerto:** cada una de las líneas de entrada/salida que comunica un módulo con el exterior.
- **Interconexión:** unión de 2 o más puertos entre sí.
- **Red:** colección de módulos interconectados de manera que **toda entrada solo está conectada a una salida** (una salida sí puede estar conectada a varias entradas).
 - Las interconexiones 1:1 y 1:n están permitidas.
 - Las interconexiones n:1 están prohibidas (a menos que se utilicen buffers triestado).



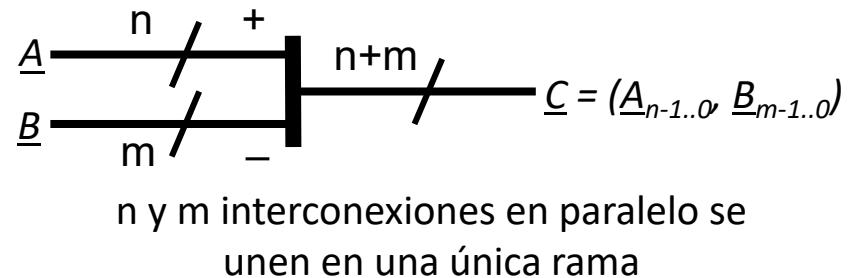
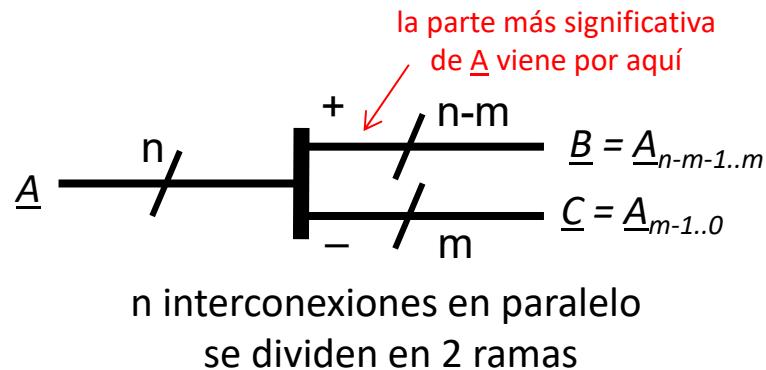
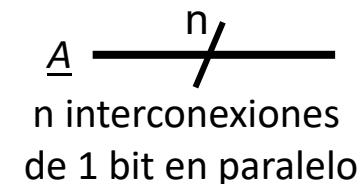
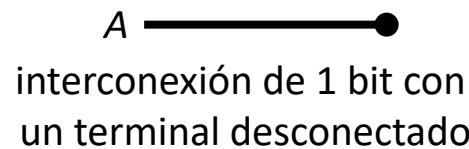
Algunas definiciones

- **Red combinacional:** red de módulos combinacionales en la que **no existen realimentaciones**.
 - no hay ningún camino dentro de la red que pase 2 veces por el mismo punto.
 - toda red combinacional es un módulo combinacional.
- **Nivel de una red:** número máximo de módulos que atraviesa cualquier camino que conecte una entrada con una salida
 - cuando la red es de puertas no se suelen contar los inversores.

Interconexiones



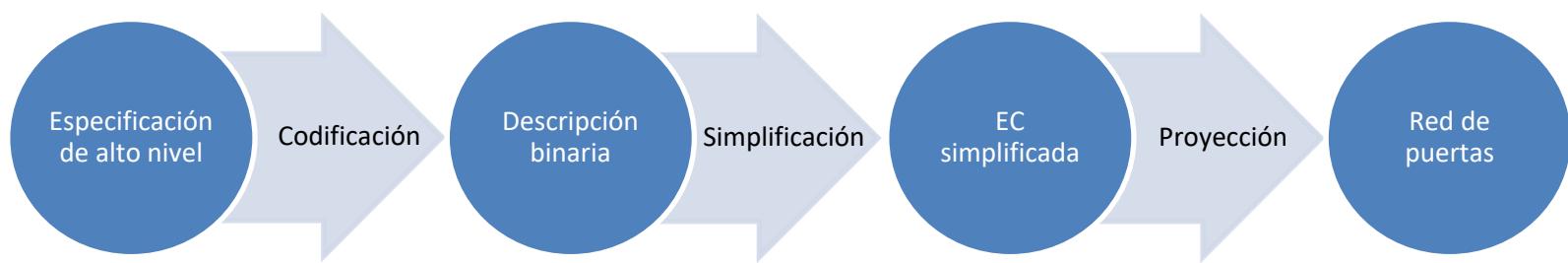
- Al dibujar el **esquema** de un circuito usaremos alguna notación adicional para las interconexiones:





Síntesis de redes de puertas

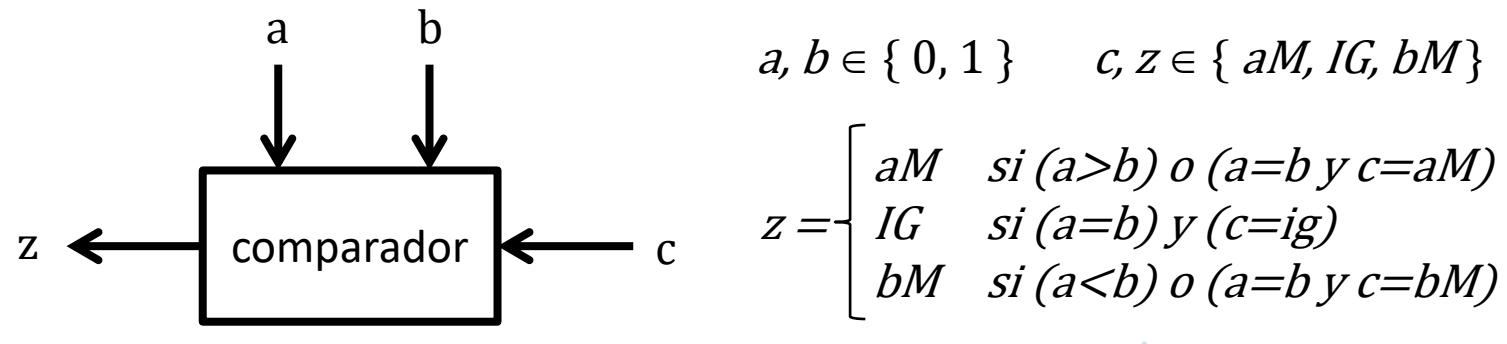
- Dada una especificación de una conducta combinacional implementarla usando puertas.



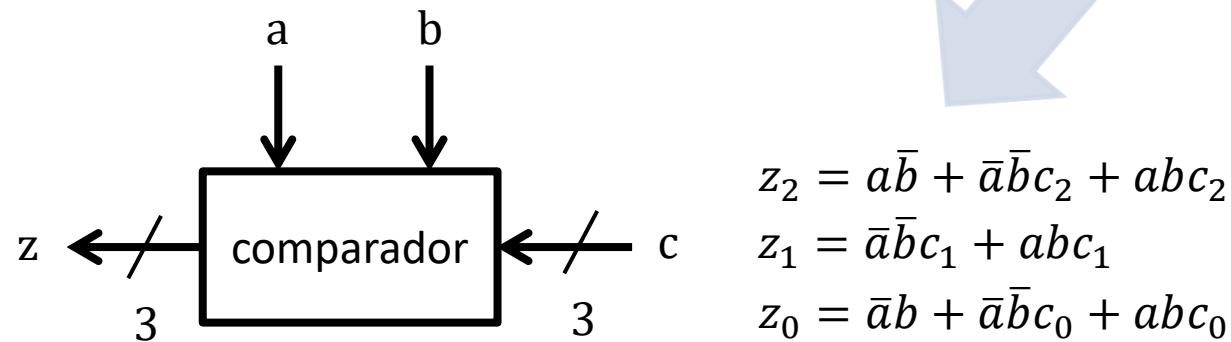
- Implementaciones a 2 niveles
 - Implementación canónica: implementa la SPC con 2 niveles AND-OR.
 - Implementación mínima: implementa una EC_{min} con 2 niveles AND-OR.
 - La red resultante tiene un número mínimo de puertas y éstas tienen un número mínimo de entradas.
- Implementaciones multnivel
 - Tienen un número arbitrario de niveles y se reutilizan cálculos intermedios



Síntesis de redes AND-OR



Codificación: $aM = (100)$, $IG = (010)$, $bM = (001)$





Síntesis de redes AND-OR

Implementación
a 2 niveles

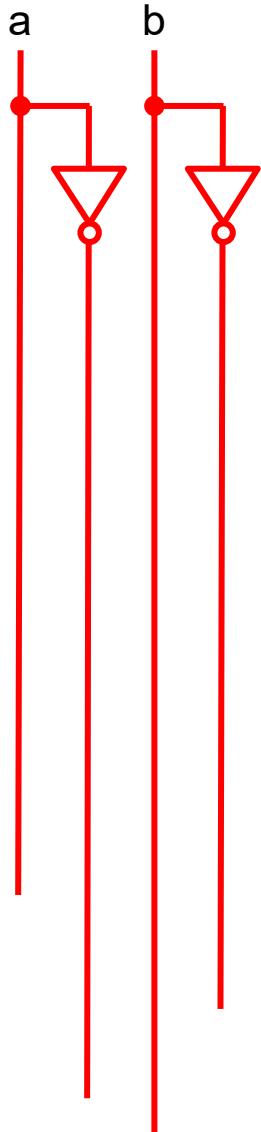
$$z_2 = a\bar{b} + \bar{a}\bar{b}c_2 + abc_2$$

$$z_1 = \bar{a}\bar{b}c_1 + abc_1$$

$$z_0 = \bar{a}b + \bar{a}\bar{b}c_0 + abc_0$$



Síntesis de redes AND-OR



Implementación
a 2 niveles

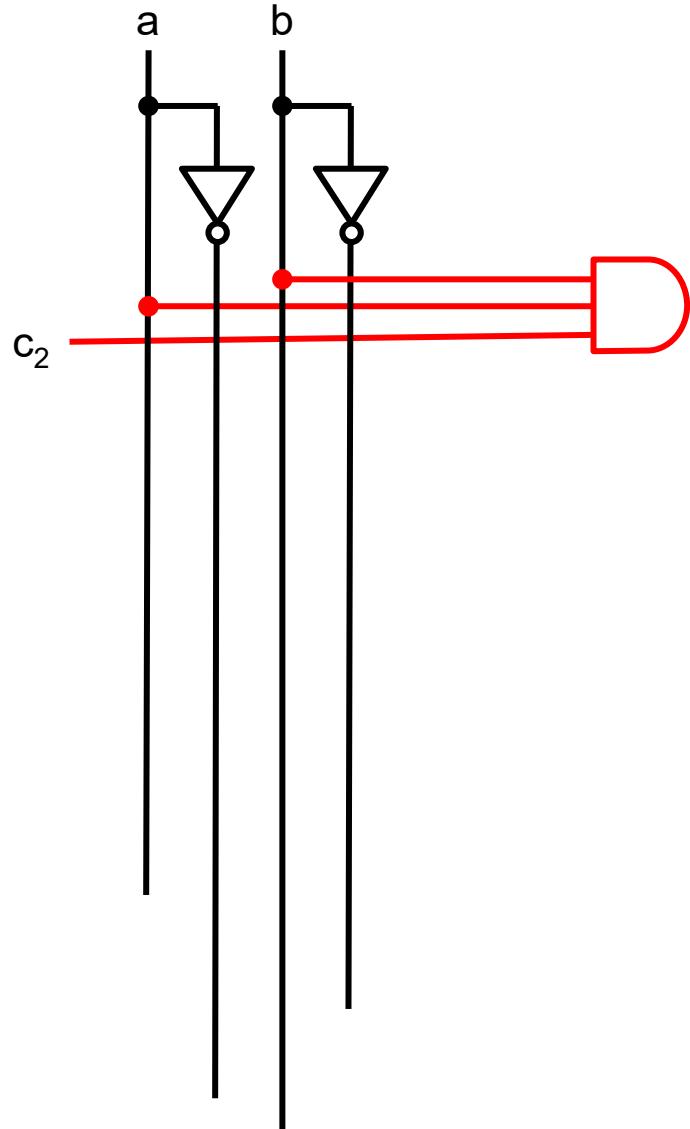
$$z_2 = a\bar{b} + \bar{a}\bar{b}c_2 + abc_2$$

$$z_1 = \bar{a}\bar{b}c_1 + abc_1$$

$$z_0 = \bar{a}b + \bar{a}\bar{b}c_0 + abc_0$$



Síntesis de redes AND-OR



Implementación
a 2 niveles

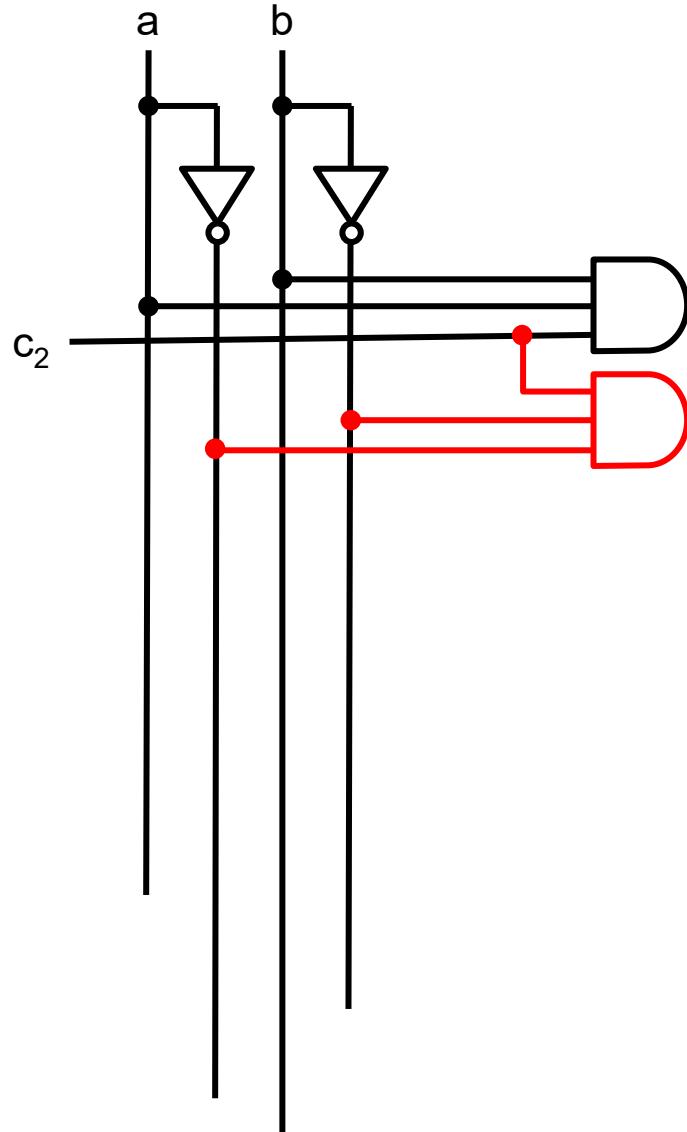
$$z_2 = a\bar{b} + \bar{a}\bar{b}c_2 + abc_2$$

$$z_1 = \bar{a}\bar{b}c_1 + abc_1$$

$$z_0 = \bar{a}b + \bar{a}\bar{b}c_0 + abc_0$$



Síntesis de redes AND-OR



Implementación
a 2 niveles

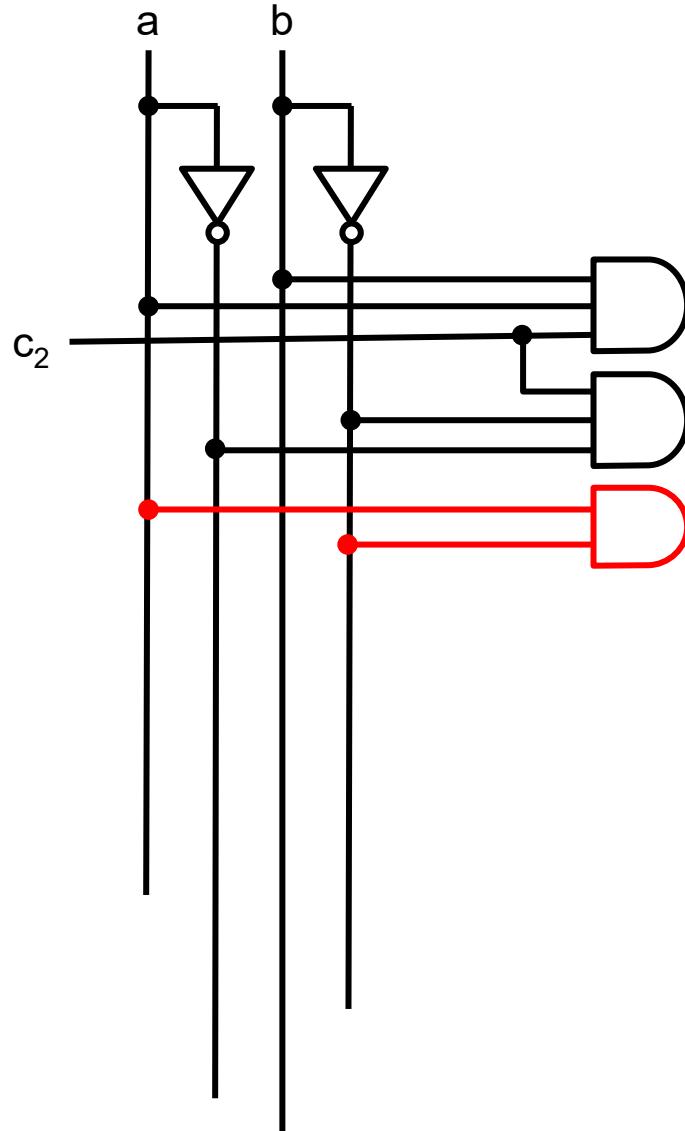
$$z_2 = a\bar{b} + \bar{a}\bar{b}c_2 + abc_2$$

$$z_1 = \bar{a}\bar{b}c_1 + abc_1$$

$$z_0 = \bar{a}b + \bar{a}\bar{b}c_0 + abc_0$$



Síntesis de redes AND-OR



Implementación
a 2 niveles

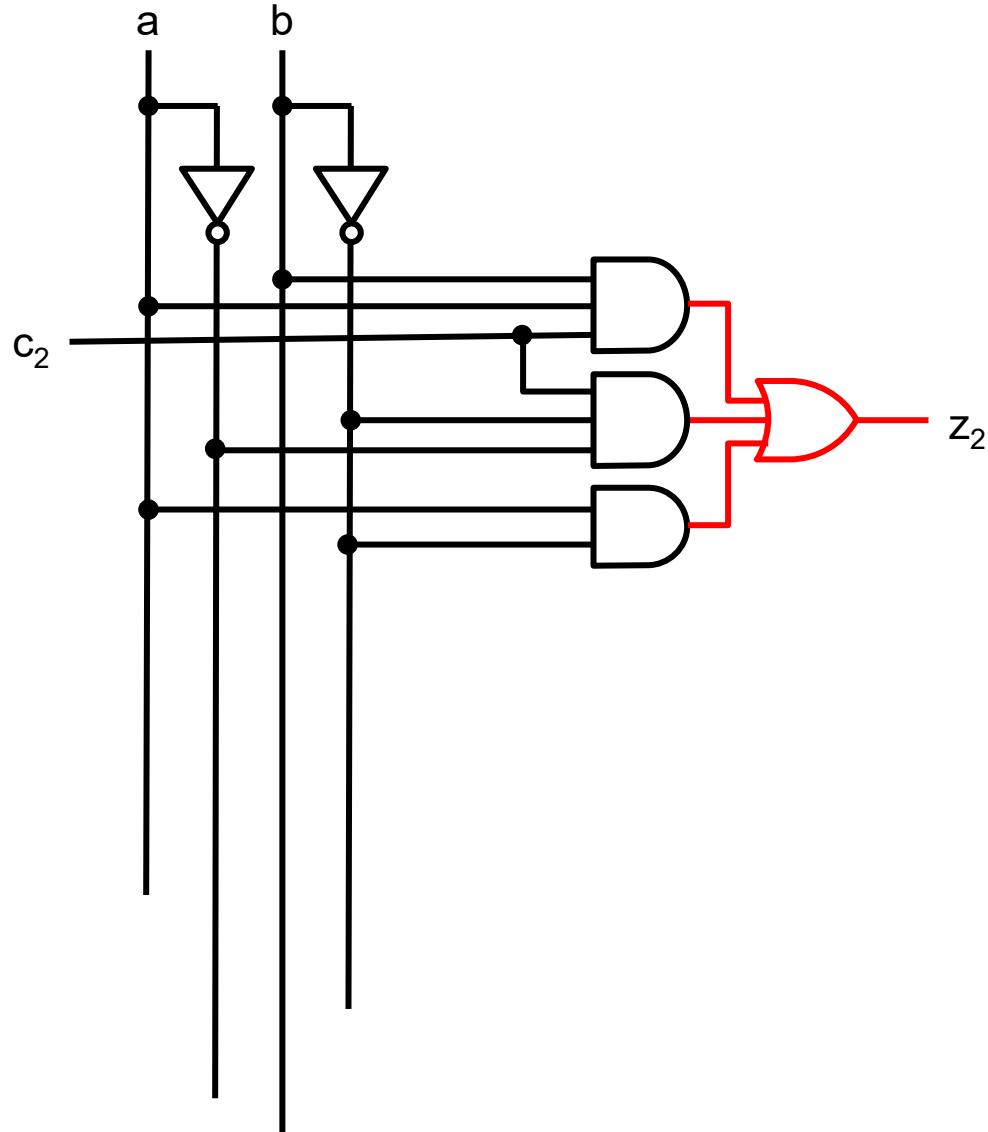
$$z_2 = a\bar{b} + \bar{a}\bar{b}c_2 + abc_2$$

$$z_1 = \bar{a}\bar{b}c_1 + abc_1$$

$$z_0 = \bar{a}b + \bar{a}\bar{b}c_0 + abc_0$$



Síntesis de redes AND-OR



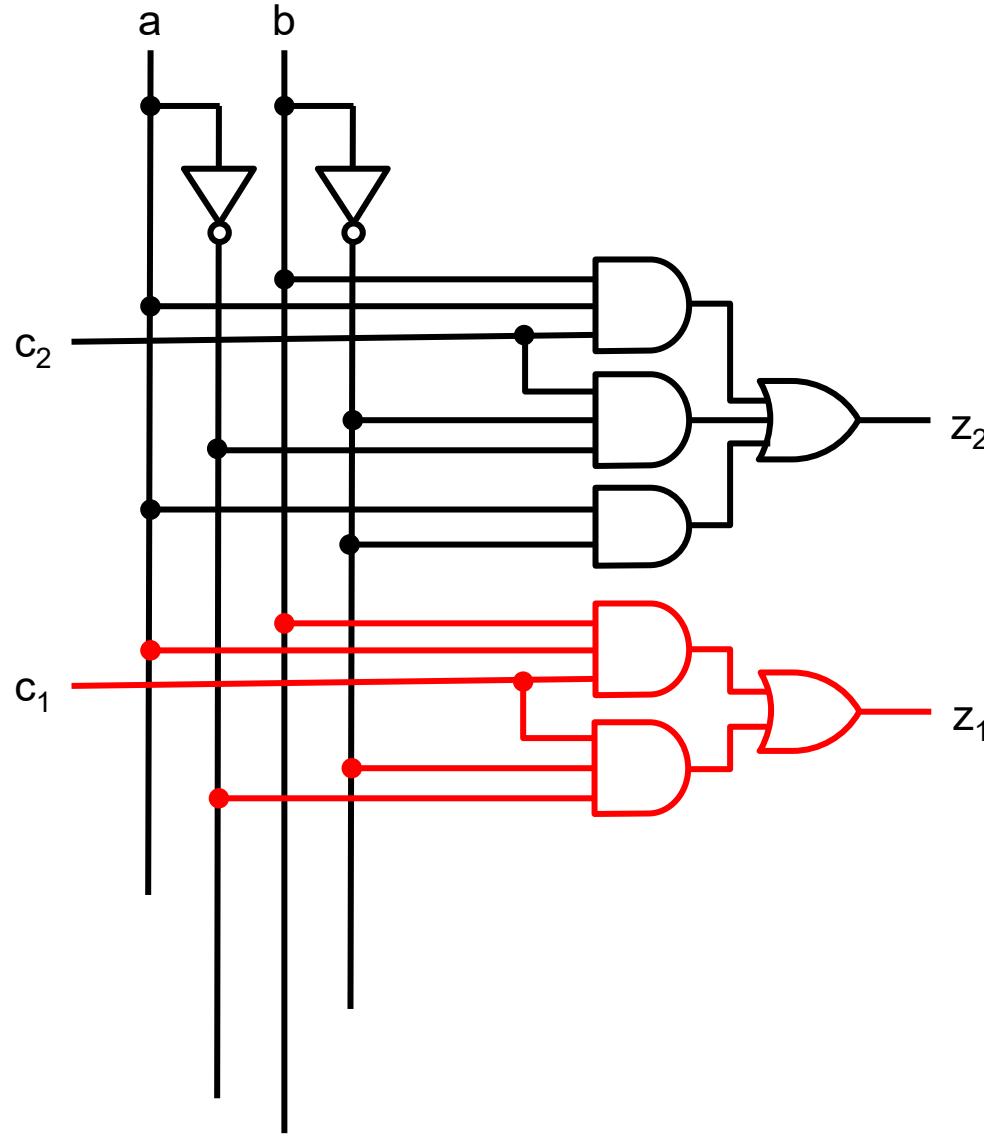
Implementación
a 2 niveles

$$z_2 = a\bar{b} + \bar{a}\bar{b}c_2 + abc_2$$

$$z_1 = \bar{a}\bar{b}c_1 + abc_1$$

$$z_0 = \bar{a}b + \bar{a}\bar{b}c_0 + abc_0$$

Síntesis de redes AND-OR



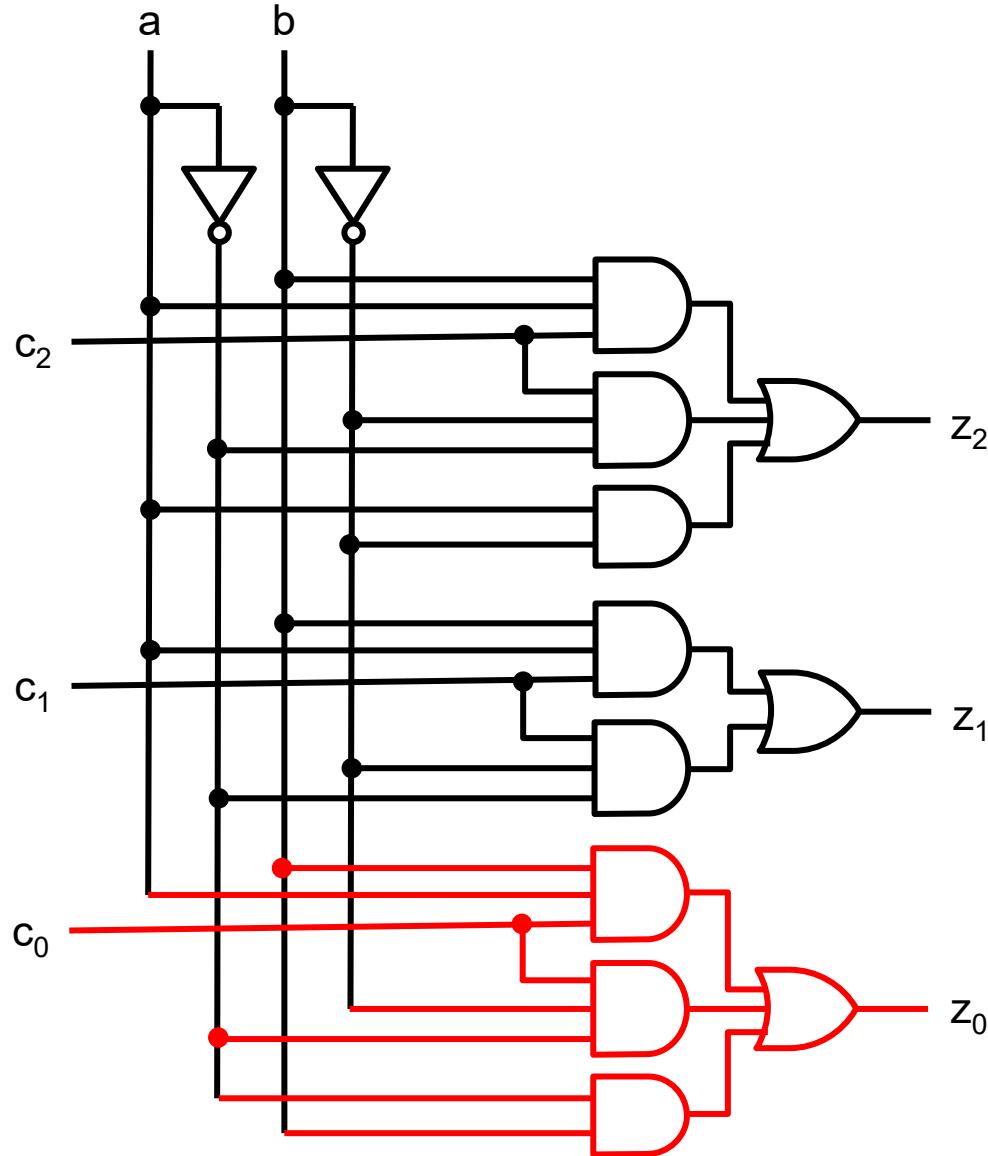
Implementación
a 2 niveles

$$z_2 = a\bar{b} + \bar{a}\bar{b}c_2 + abc_2$$

$$z_1 = \bar{a}\bar{b}c_1 + abc_1$$

$$z_0 = \bar{a}b + \bar{a}\bar{b}c_0 + abc_0$$

Síntesis de redes AND-OR



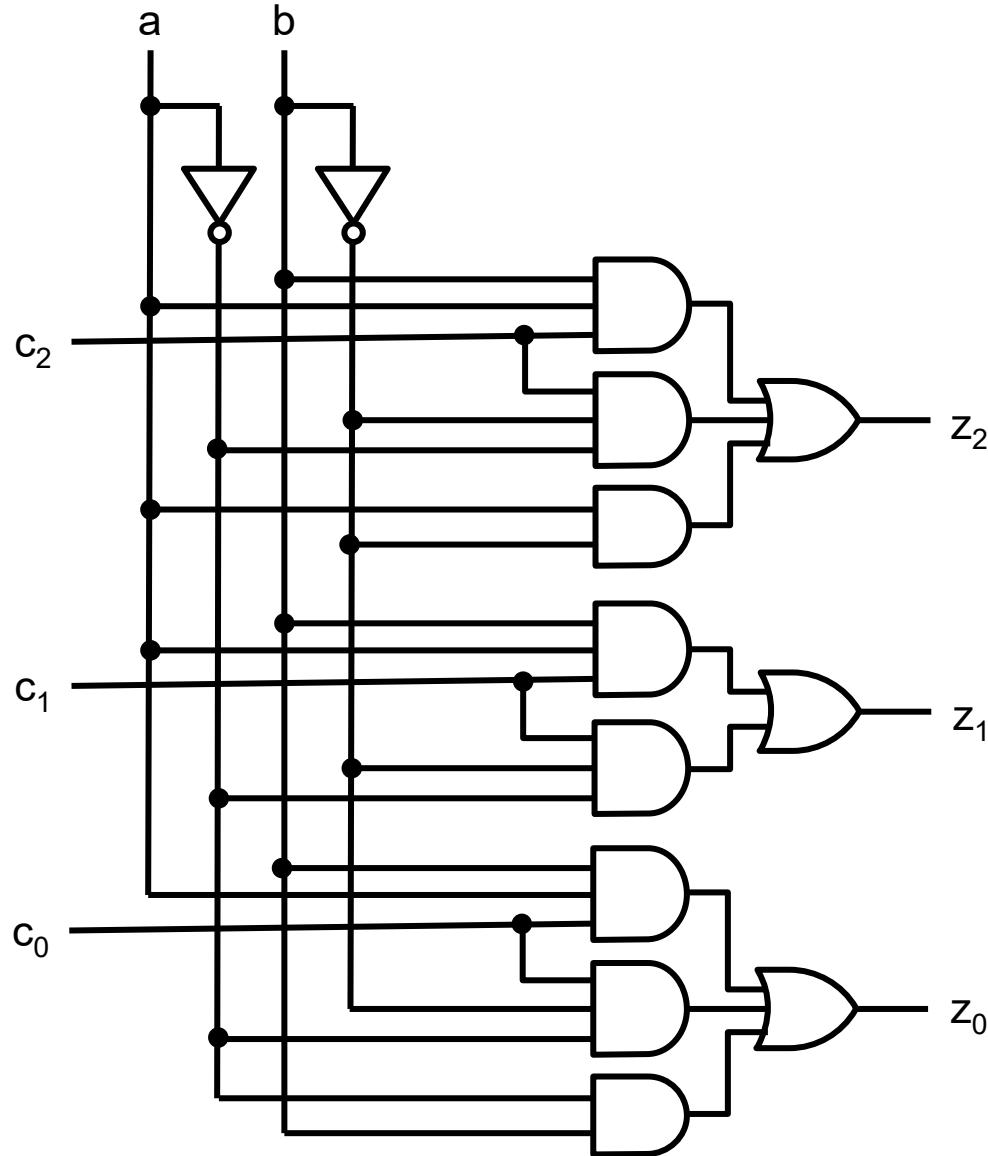
Implementación
a 2 niveles

$$z_2 = a\bar{b} + \bar{a}\bar{b}c_2 + abc_2$$

$$z_1 = \bar{a}\bar{b}c_1 + abc_1$$

$$z_0 = \bar{a}b + \bar{a}\bar{b}c_0 + abc_0$$

Síntesis de redes AND-OR



Implementación
a 2 niveles

$$z_2 = a\bar{b} + \bar{a}\bar{b}c_2 + abc_2$$

$$z_1 = \bar{a}\bar{b}c_1 + abc_1$$

$$z_0 = \bar{a}b + \bar{a}\bar{b}c_0 + abc_0$$



Síntesis de redes AND-OR

tema 2:
Sistemas combinacionales

FC

84

versión 2021

Implementación
multinivel

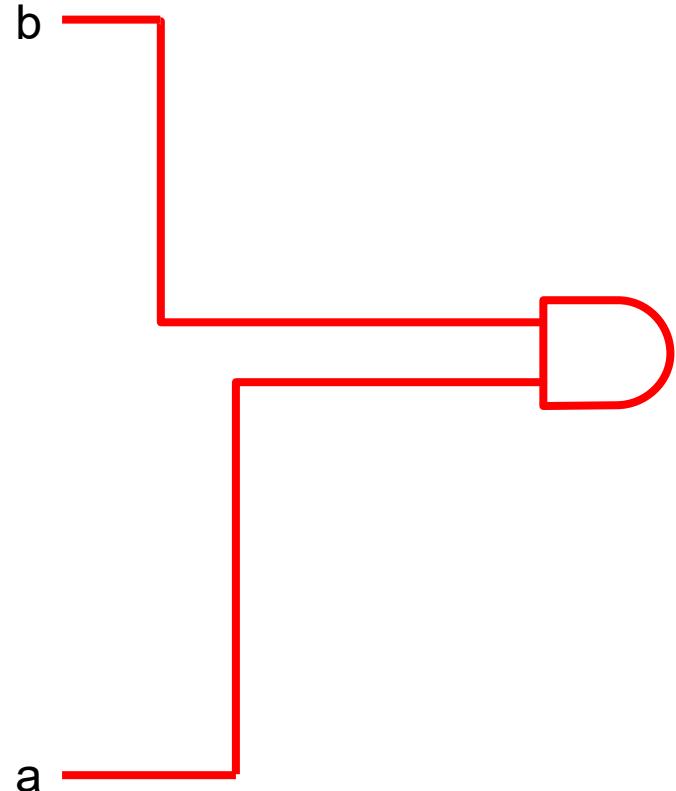
$$\begin{aligned}z_2 &= a\bar{b} + \bar{a}\bar{b}c_2 + abc_2 \\z_1 &= \bar{a}\bar{b}c_1 + abc_1 \\z_0 &= \bar{a}b + \bar{a}\bar{b}c_0 + abc_0\end{aligned}$$

factorizando
 $(\bar{a}\bar{b} + ab)$

$$\begin{aligned}z_2 &= a\bar{b} + (\bar{a}\bar{b} + ab)c_2 \\z_1 &= (\bar{a}\bar{b} + ab)c_1 \\z_0 &= \bar{a}b + (\bar{a}\bar{b} + ab)c_0\end{aligned}$$



Síntesis de redes AND-OR

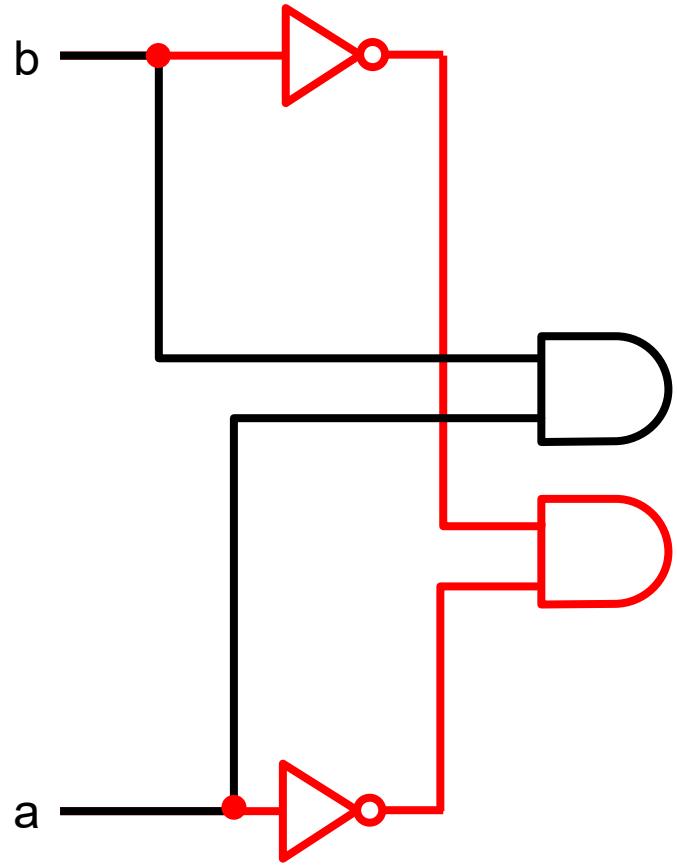


Implementación
multinivel

$$\begin{aligned} z_2 &= a\bar{b} + (\bar{a}\bar{b} + ab)c_2 \\ z_1 &= (\bar{a}\bar{b} + ab)c_1 \\ z_0 &= \bar{a}b + (\bar{a}\bar{b} + ab)c_0 \end{aligned}$$



Síntesis de redes AND-OR

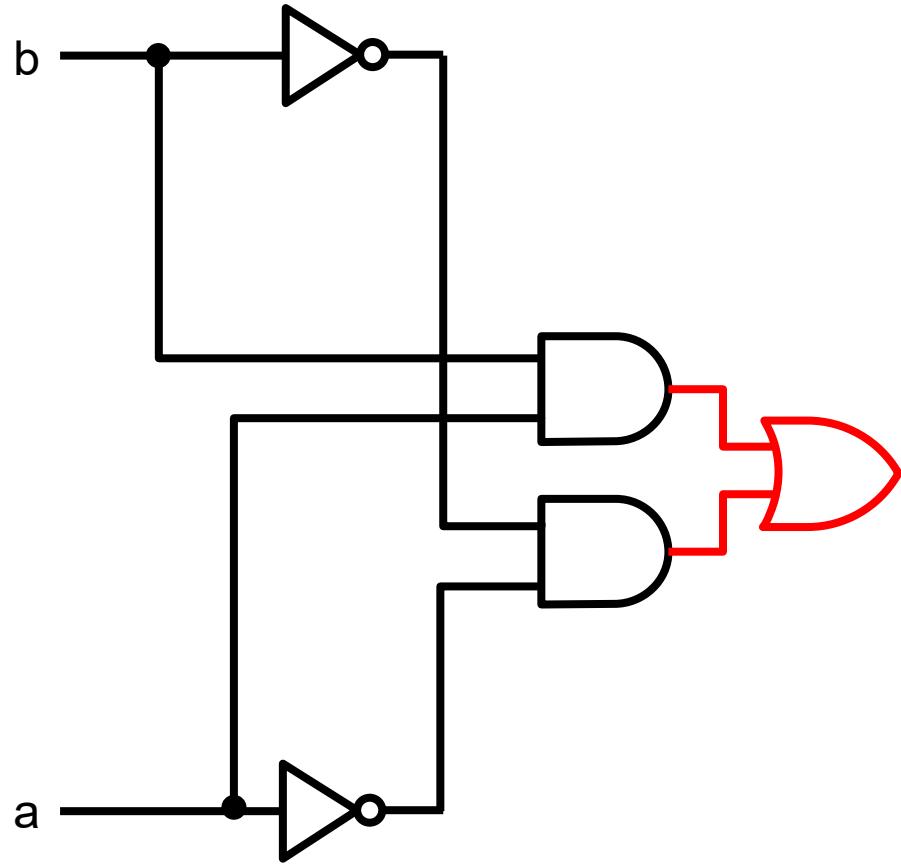


Implementación
multinivel

$$\begin{aligned} z_2 &= a\bar{b} + (\bar{a}\bar{b} + ab)c_2 \\ z_1 &= (\bar{a}\bar{b} + ab)c_1 \\ z_0 &= \bar{a}b + (\bar{a}\bar{b} + ab)c_0 \end{aligned}$$



Síntesis de redes AND-OR

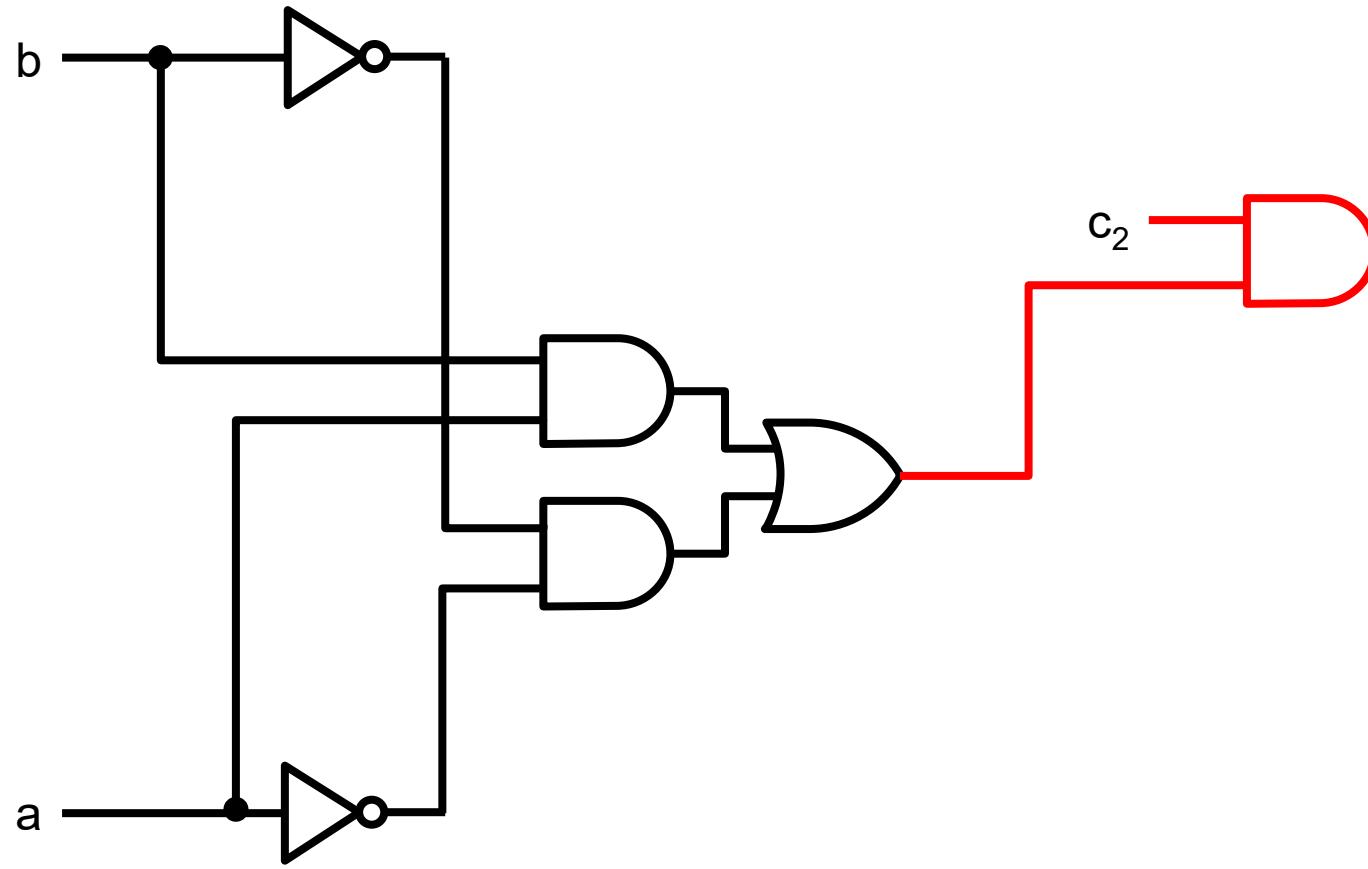


Implementación
multinivel

$$\begin{aligned}z_2 &= a\bar{b} + (\bar{a}\bar{b} + ab)c_2 \\z_1 &= (\bar{a}\bar{b} + ab)c_1 \\z_0 &= \bar{a}b + (\bar{a}\bar{b} + ab)c_0\end{aligned}$$



Síntesis de redes AND-OR

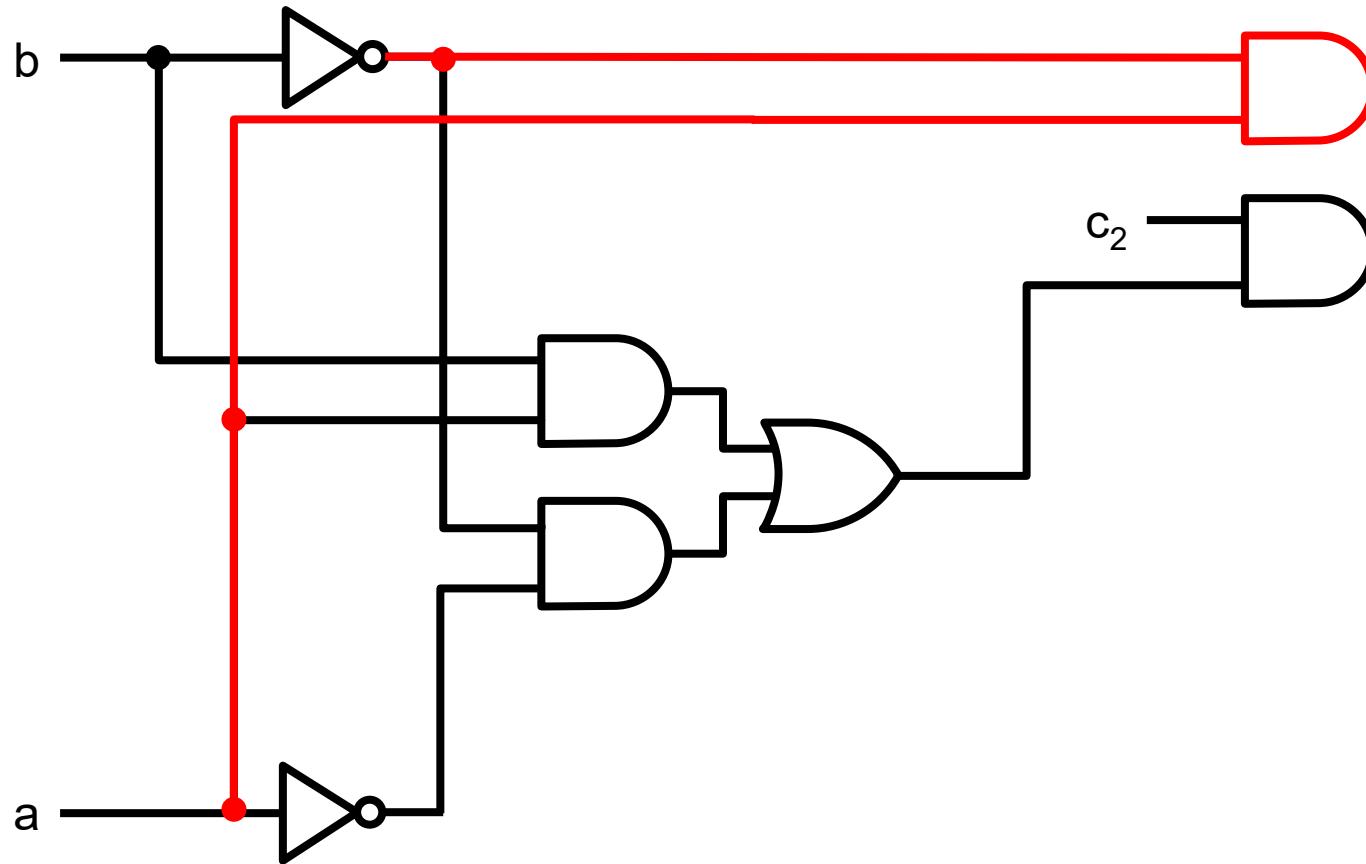


Implementación
multinivel

$$\begin{aligned}z_2 &= a\bar{b} + (\bar{a}\bar{b} + ab)c_2 \\z_1 &= (\bar{a}\bar{b} + ab)c_1 \\z_0 &= \bar{a}b + (\bar{a}\bar{b} + ab)c_0\end{aligned}$$



Síntesis de redes AND-OR

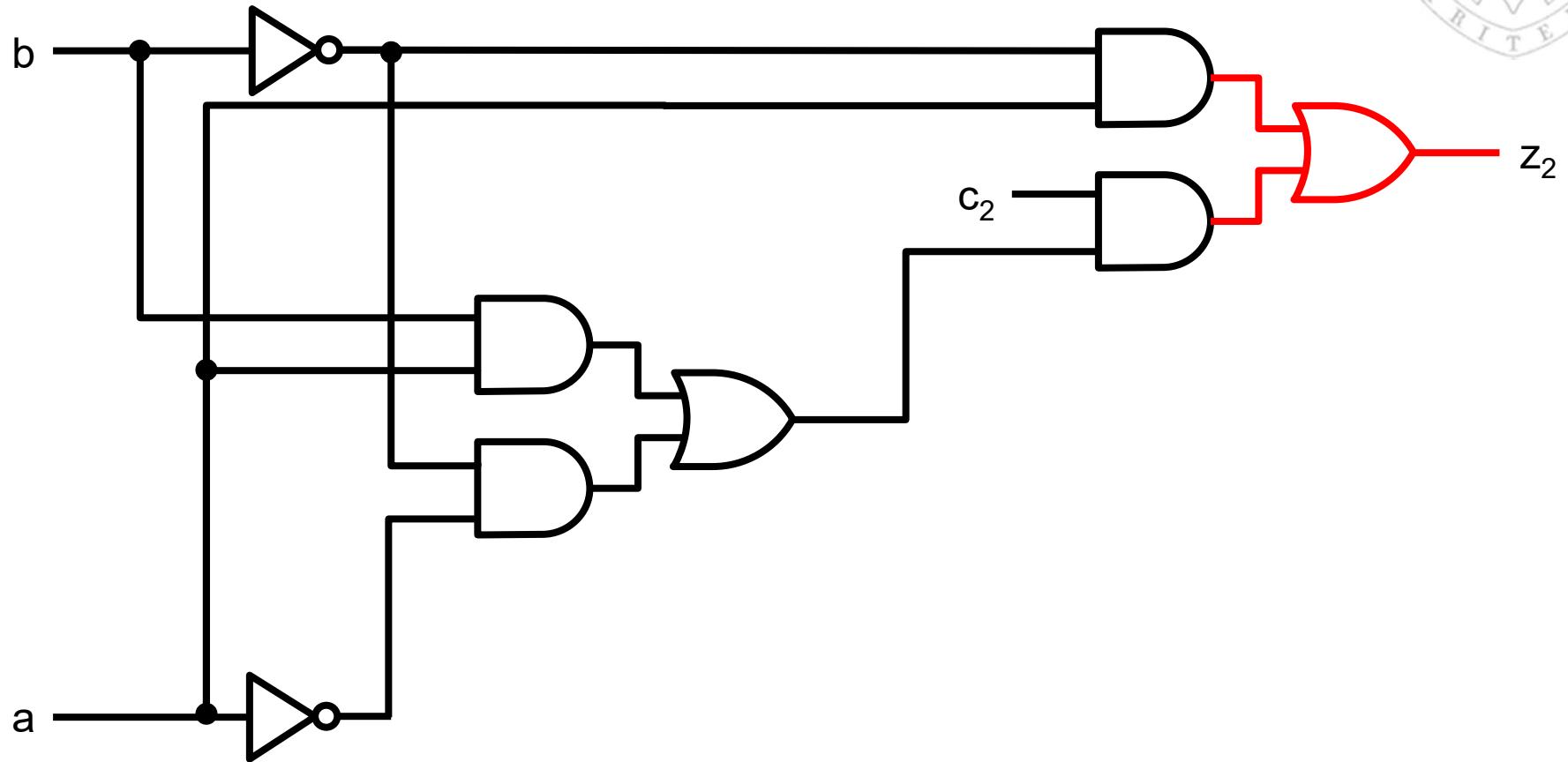


Implementación
multinivel

$$\begin{aligned}z_2 &= \bar{a}\bar{b} + (\bar{a}\bar{b} + ab)c_2 \\z_1 &= (\bar{a}\bar{b} + ab)c_1 \\z_0 &= \bar{a}b + (\bar{a}\bar{b} + ab)c_0\end{aligned}$$



Síntesis de redes AND-OR

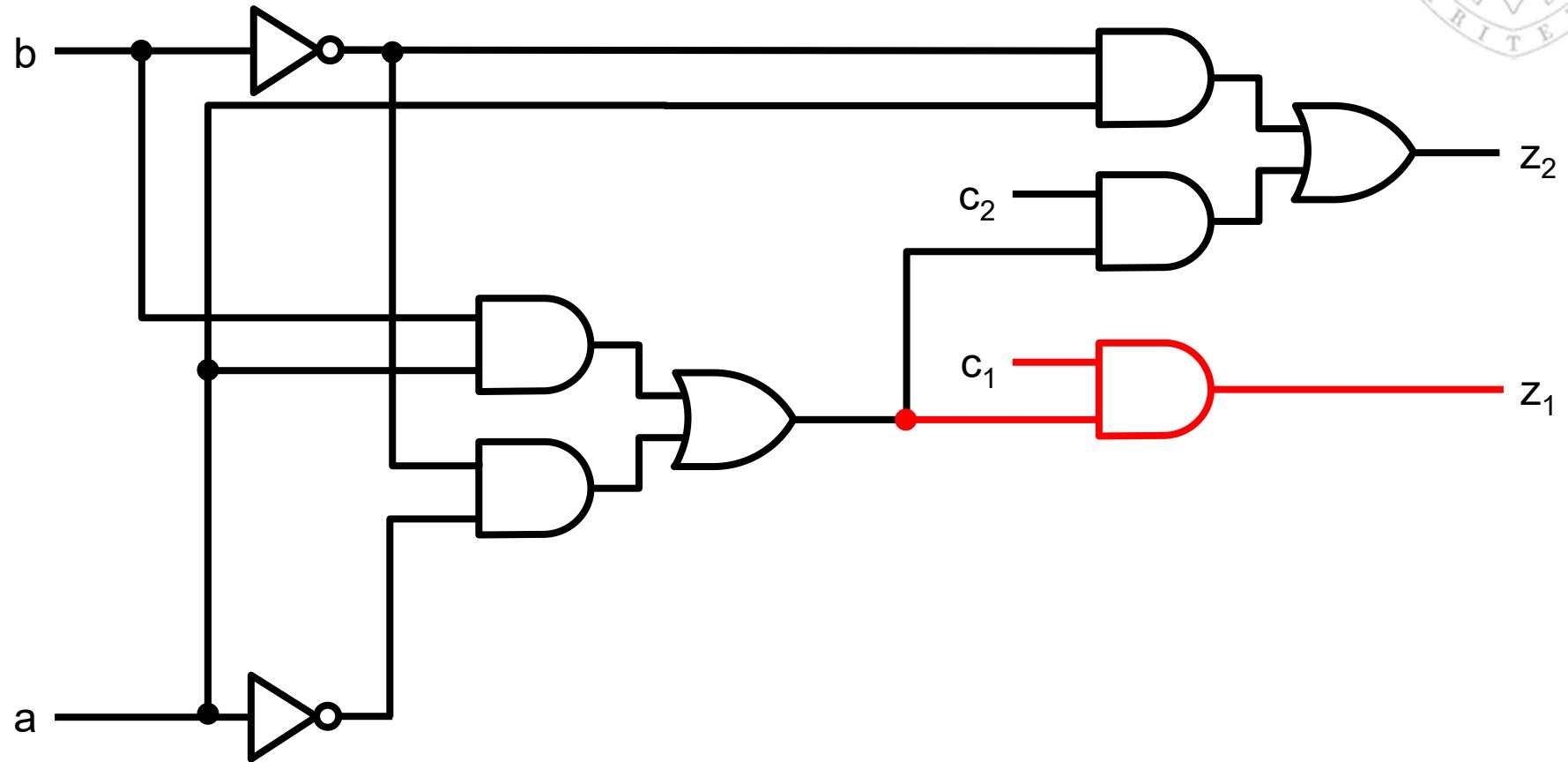


Implementación
multinivel

$$\begin{aligned}z_2 &= ab + (\bar{a}\bar{b} + ab)c_2 \\z_1 &= (\bar{a}\bar{b} + ab)c_1 \\z_0 &= \bar{a}b + (\bar{a}\bar{b} + ab)c_0\end{aligned}$$



Síntesis de redes AND-OR



Implementación
multinivel

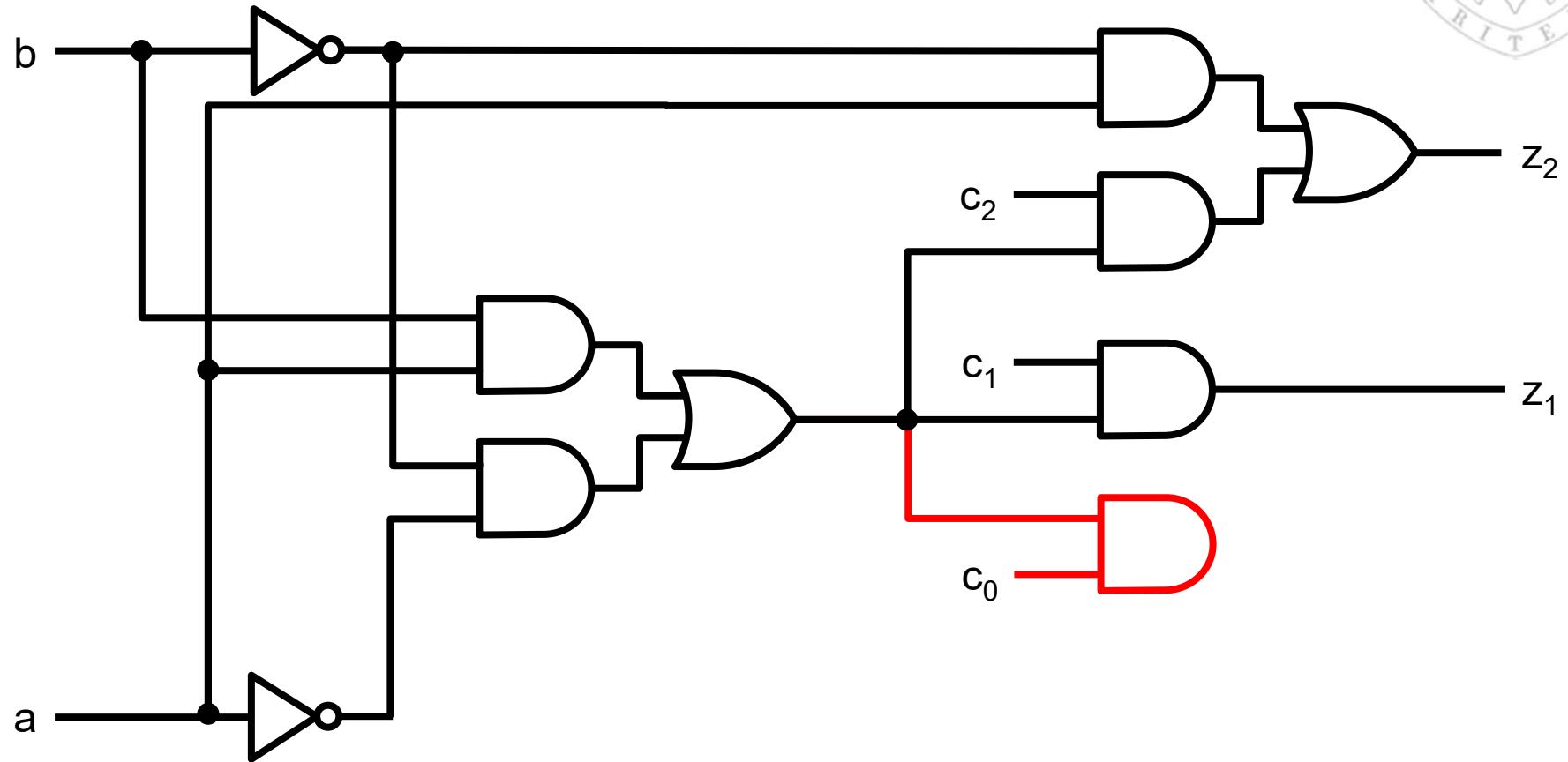
$$z_2 = a\bar{b} + (\bar{a}\bar{b} + ab)c_2$$

$$z_1 = (\bar{a}\bar{b} + ab)c_1$$

$$z_0 = \bar{a}b + (\bar{a}\bar{b} + ab)c_0$$



Síntesis de redes AND-OR



Implementación
multinivel

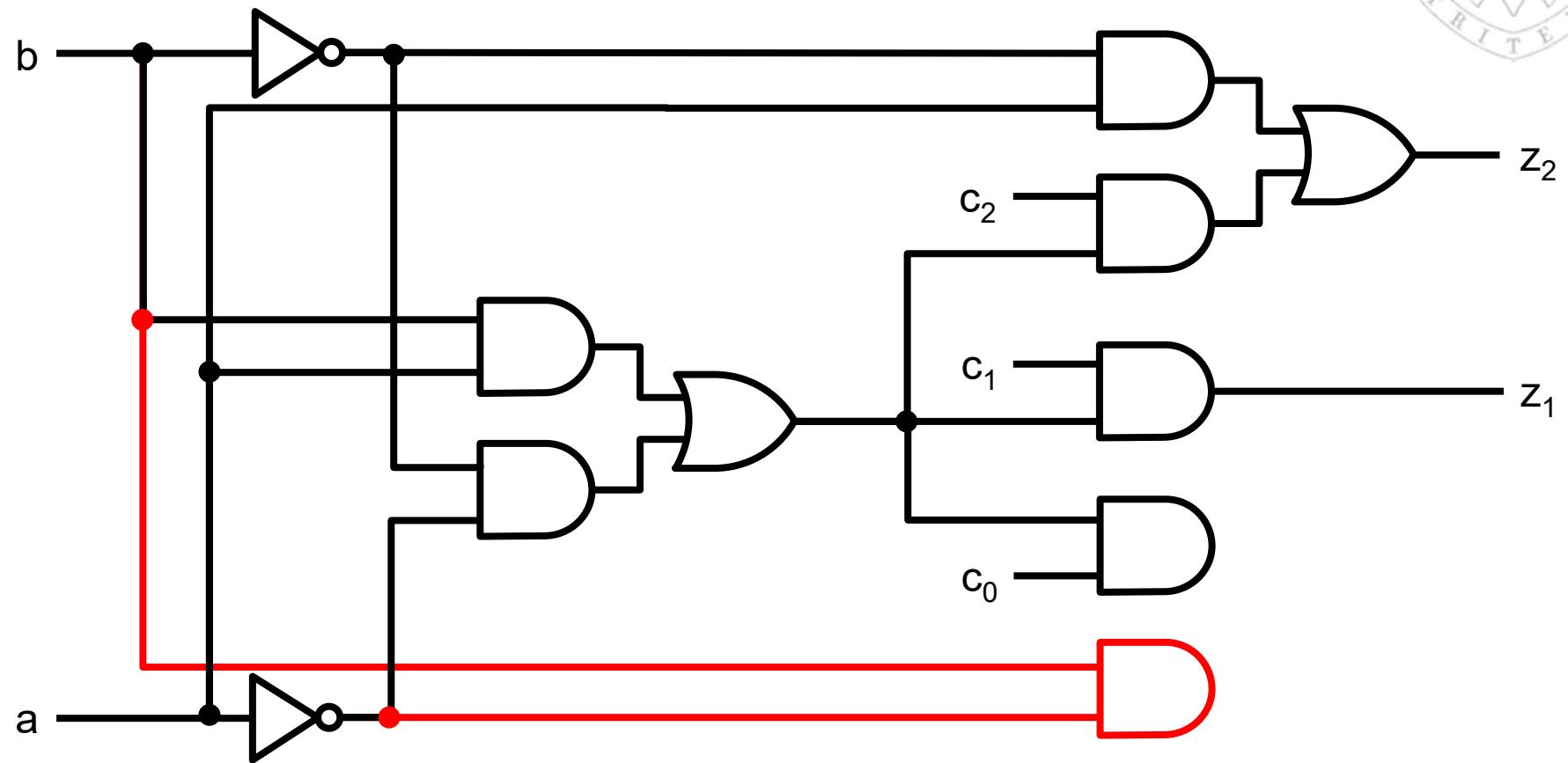
$$z_2 = a\bar{b} + (\bar{a}\bar{b} + ab)c_2$$

$$z_1 = (\bar{a}\bar{b} + ab)c_1$$

$$z_0 = \bar{a}b + (\bar{a}\bar{b} + ab)c_0$$



Síntesis de redes AND-OR

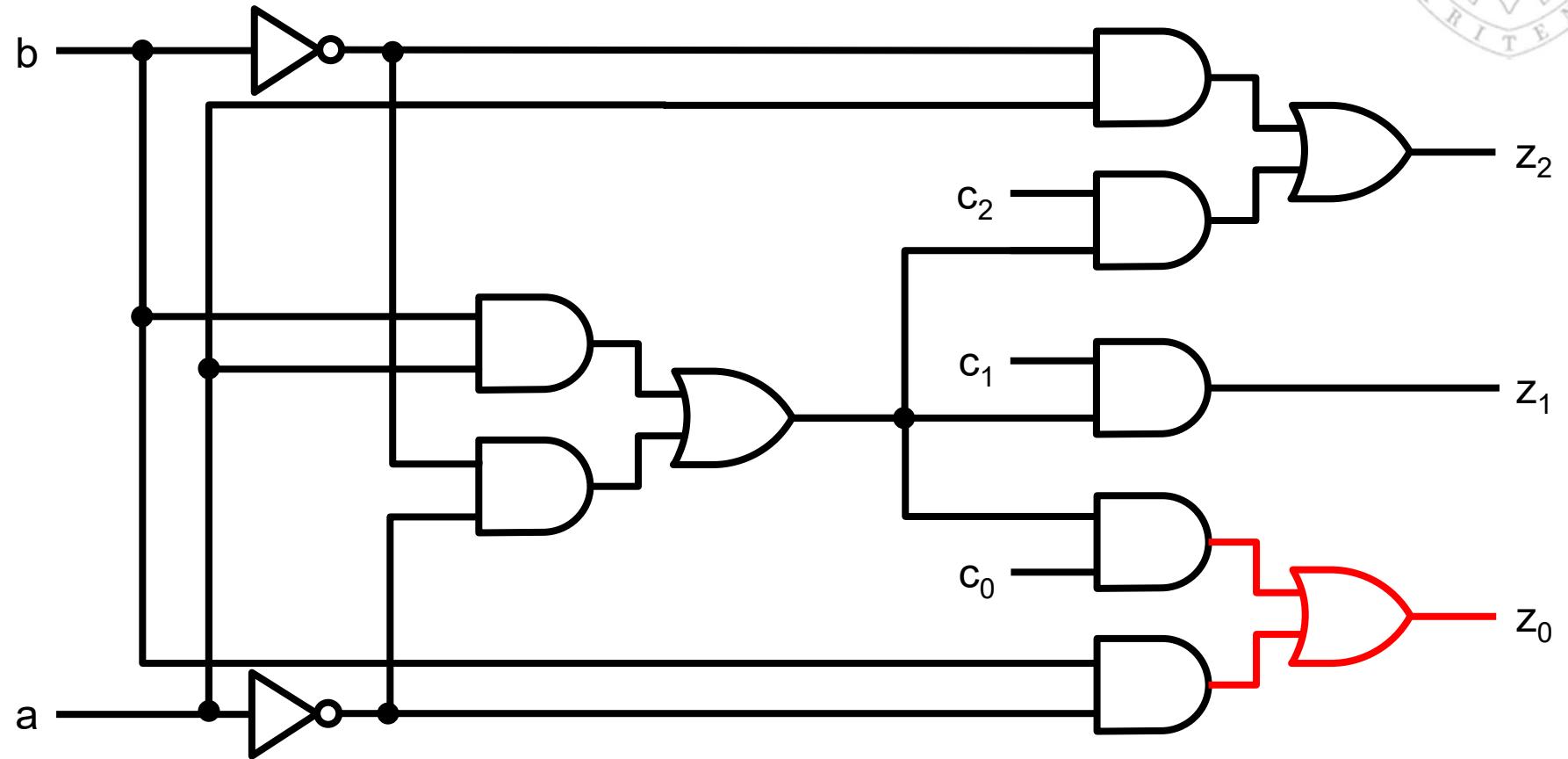


Implementación
multinivel

$$\begin{aligned} z_2 &= a\bar{b} + (\bar{a}\bar{b} + ab)c_2 \\ z_1 &= (\bar{a}\bar{b} + ab)c_1 \\ z_0 &= \bar{a}b + (\bar{a}\bar{b} + ab)c_0 \end{aligned}$$



Síntesis de redes AND-OR



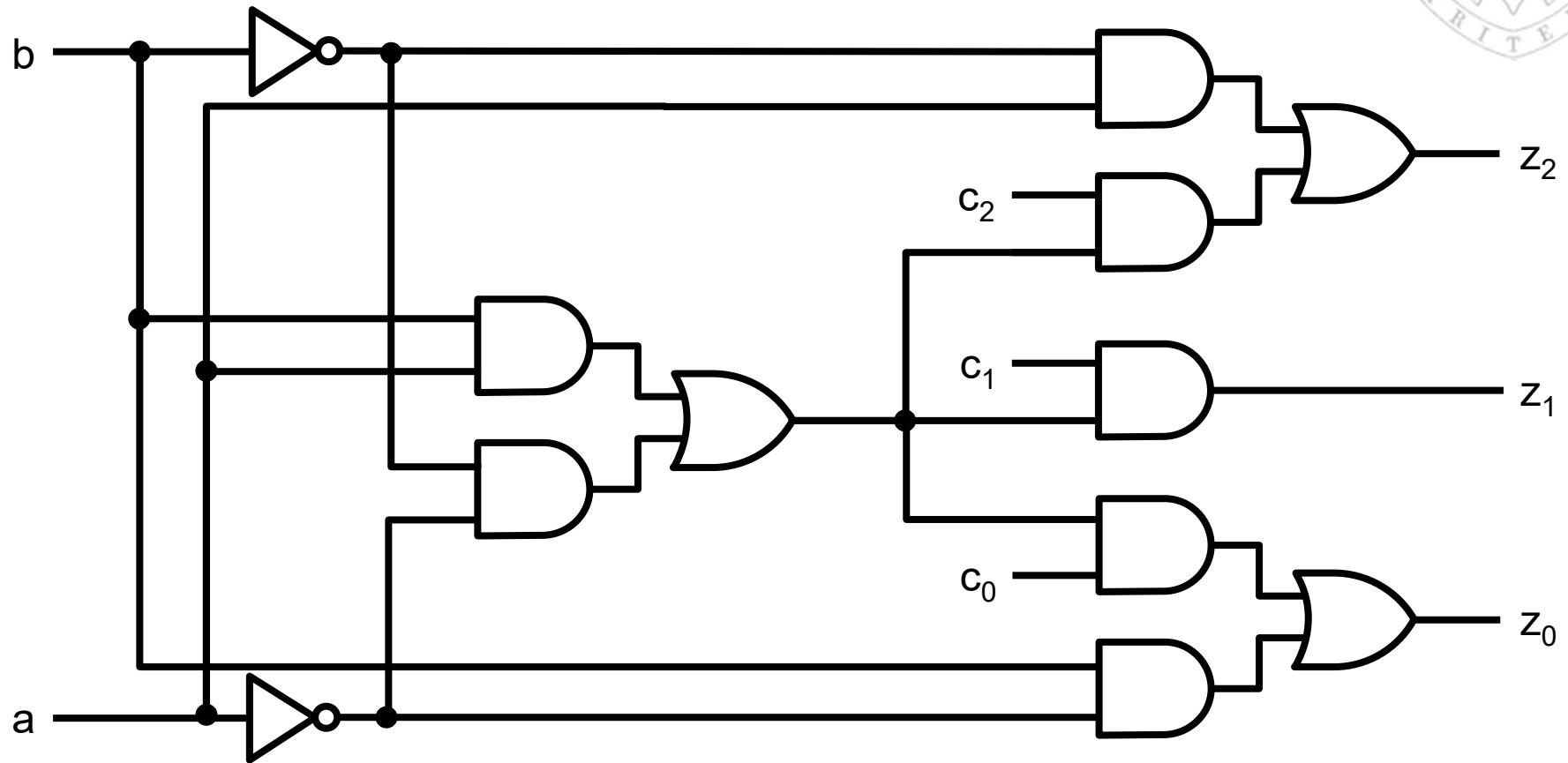
Implementación
multinivel

$$z_2 = a\bar{b} + (\bar{a}\bar{b} + ab)c_2$$

$$z_1 = (\bar{a}\bar{b} + ab)c_1$$

$$z_0 = \bar{a}\bar{b} + (\bar{a}\bar{b} + ab)c_0$$

Síntesis de redes AND-OR



Implementación multinivel

$$\begin{aligned}z_2 &= a\bar{b} + (\bar{a}\bar{b} + ab)c_2 \\z_1 &= (\bar{a}\bar{b} + ab)c_1 \\z_0 &= \bar{a}b + (\bar{a}\bar{b} + ab)c_0\end{aligned}$$



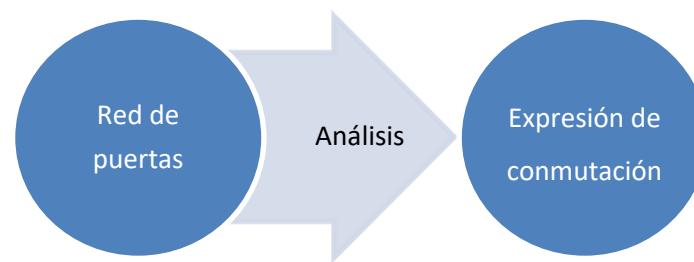
Contenidos

- ✓ Sistema combinacional
- ✓ Expresiones de conmutación.
- ✓ Forma canónica. Suma de productos.
- ✓ Simplificación por mapas de Karnaugh
- ✓ Puertas lógicas: Síntesis con redes AND-OR
- ✓ **Análisis de redes de puertas.**
- ✓ Módulos combinacionales básicos: Decodificador, mux...
- ✓ ROM
- ✓ Módulos aritméticos

Análisis de redes de puertas

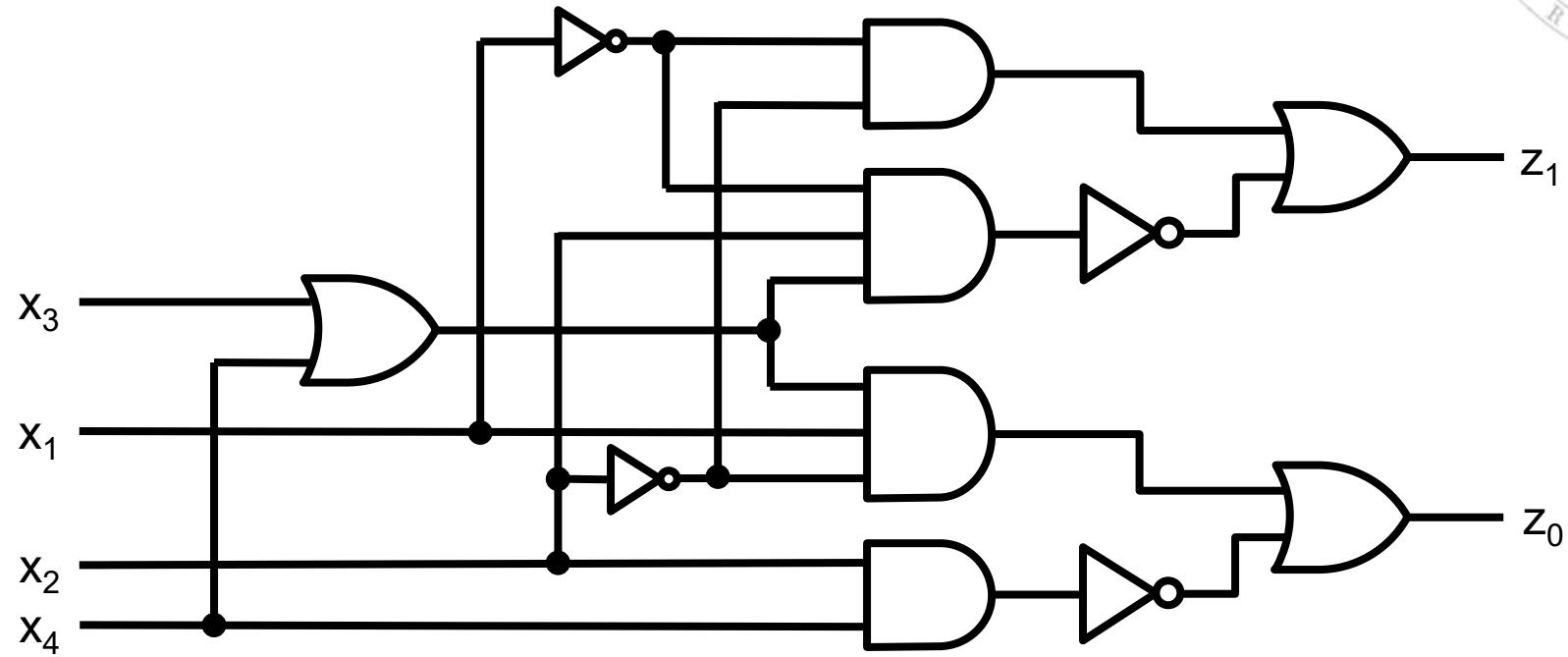


- Dada una red de puertas obtener una descripción de su conducta



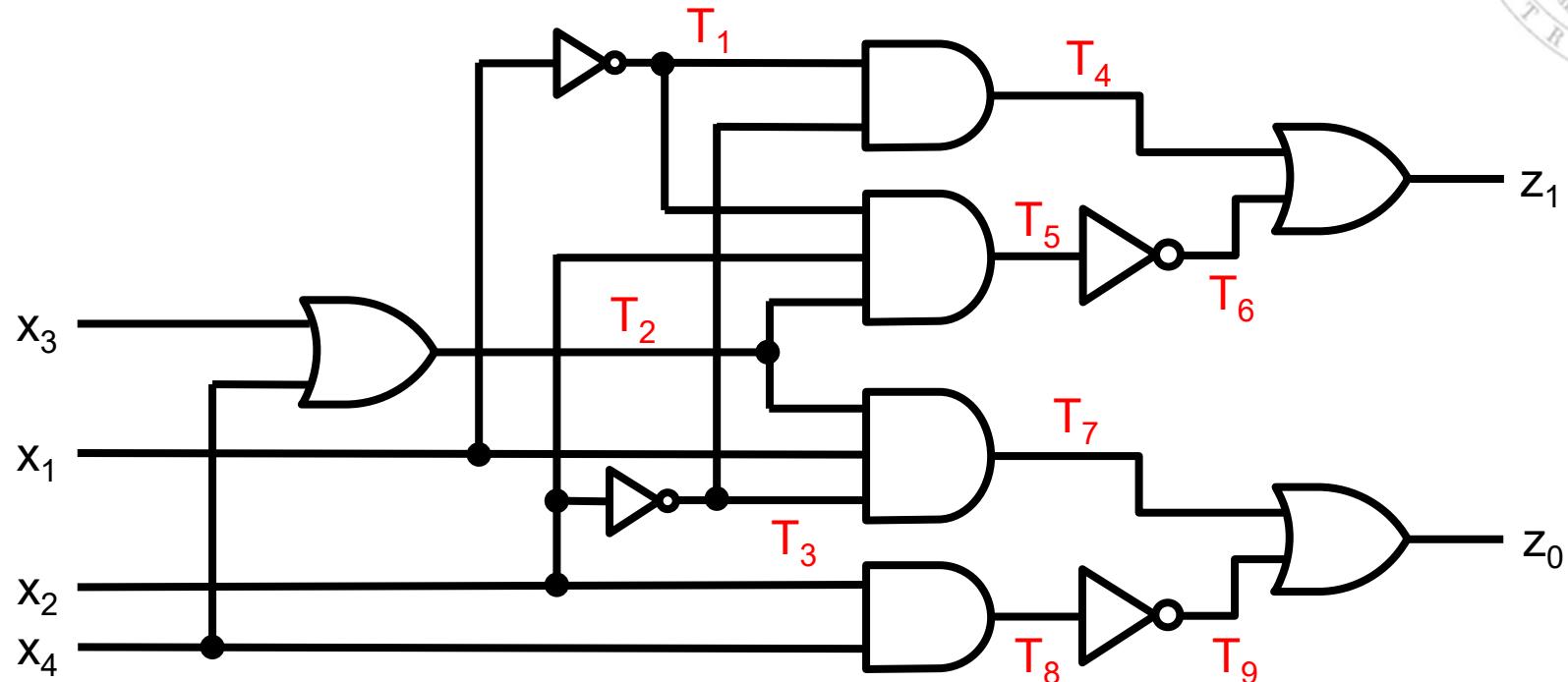
- **Método:**
 - Dar nombre a cada una de las interconexiones intermedias.
 - En dirección de entradas a salidas, obtener una EC de cada una de dichas interconexiones como función de las entradas.
 - Simplificar las expresiones obtenidas.

Análisis de redes AND-OR



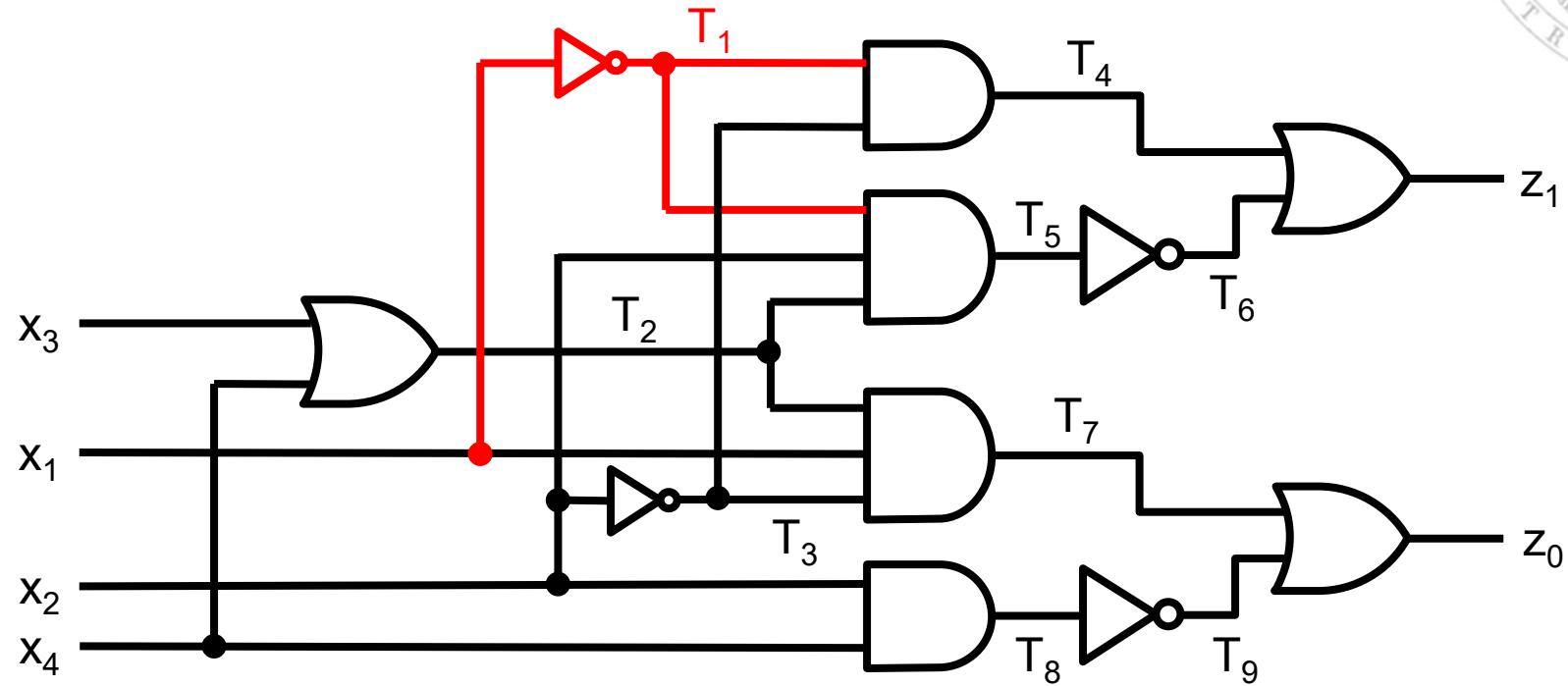


Análisis de redes AND-OR





Análisis de redes AND-OR



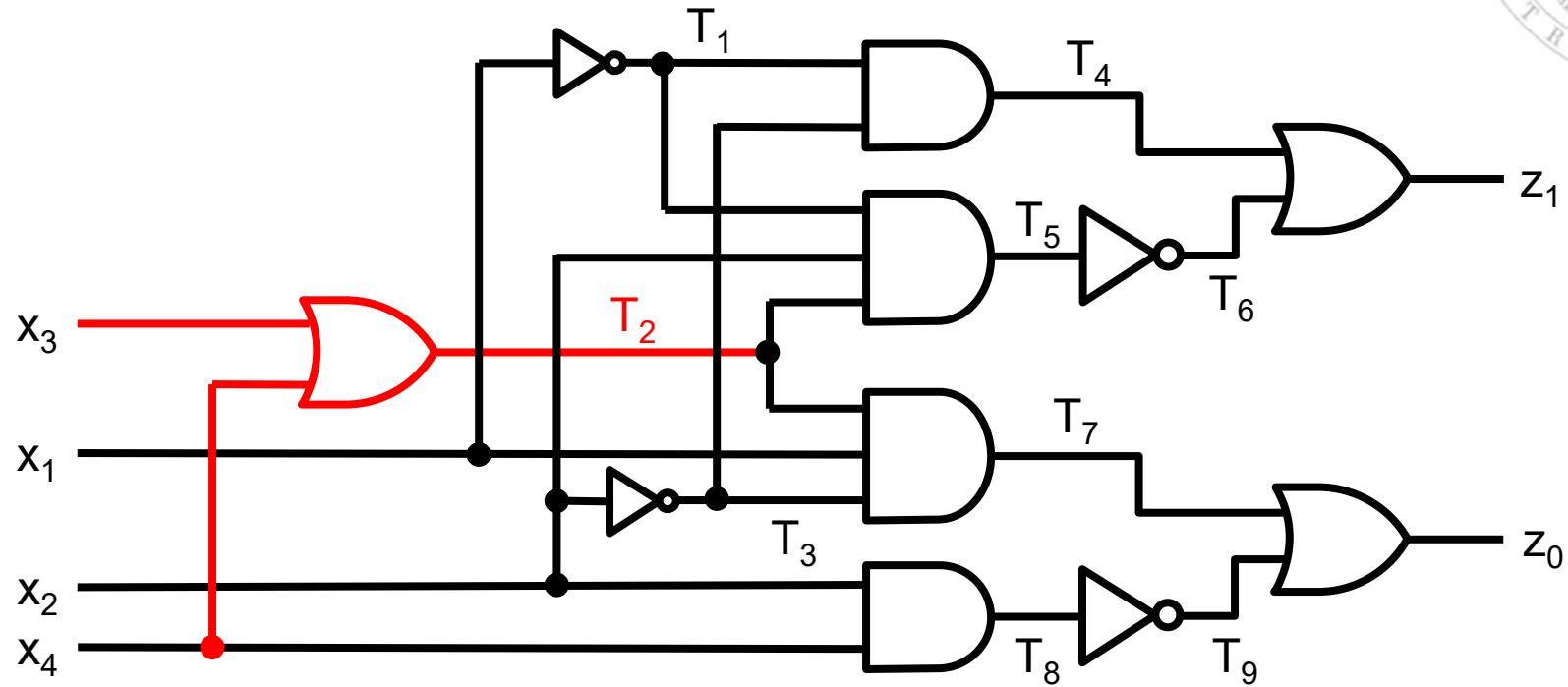
$$T_1 = \bar{x}_1$$



Análisis de redes AND-OR

versión 2021

tema 2:
Sistemas combinacionales

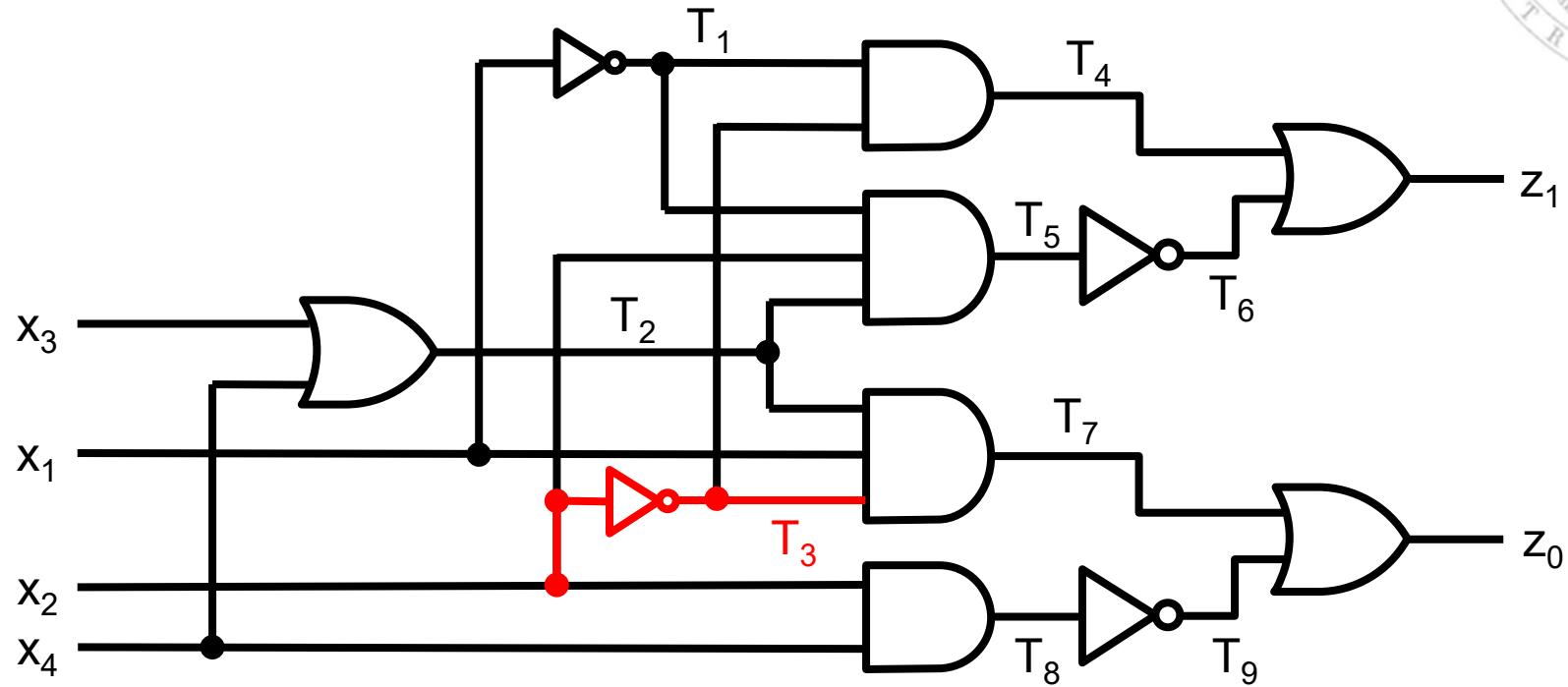


$$T_1 = \overline{x_1}$$

$$T_2 = x_3 + x_4$$



Análisis de redes AND-OR



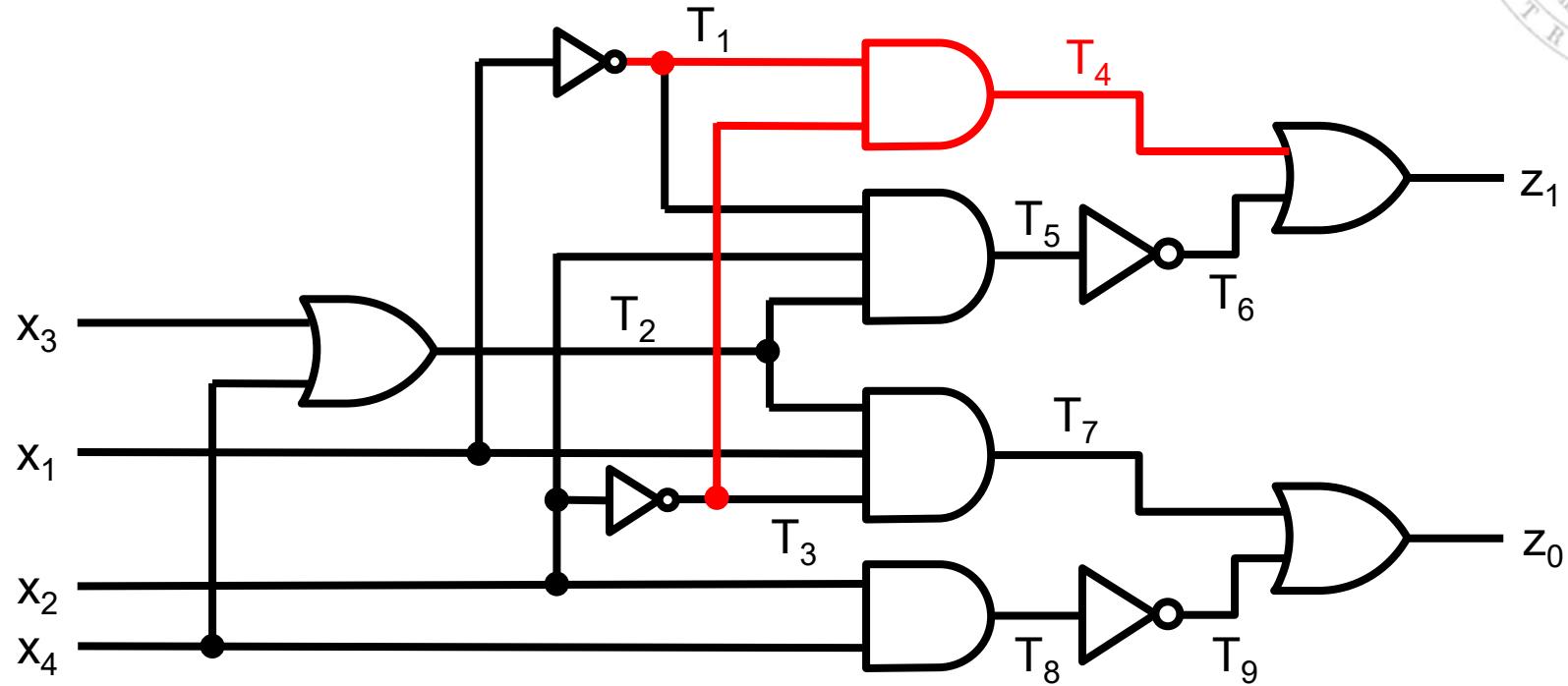
$$T_1 = \overline{x_1}$$

$$T_2 = x_3 + x_4$$

$$T_3 = \overline{x_2}$$



Análisis de redes AND-OR



$$T_1 = \overline{x_1}$$

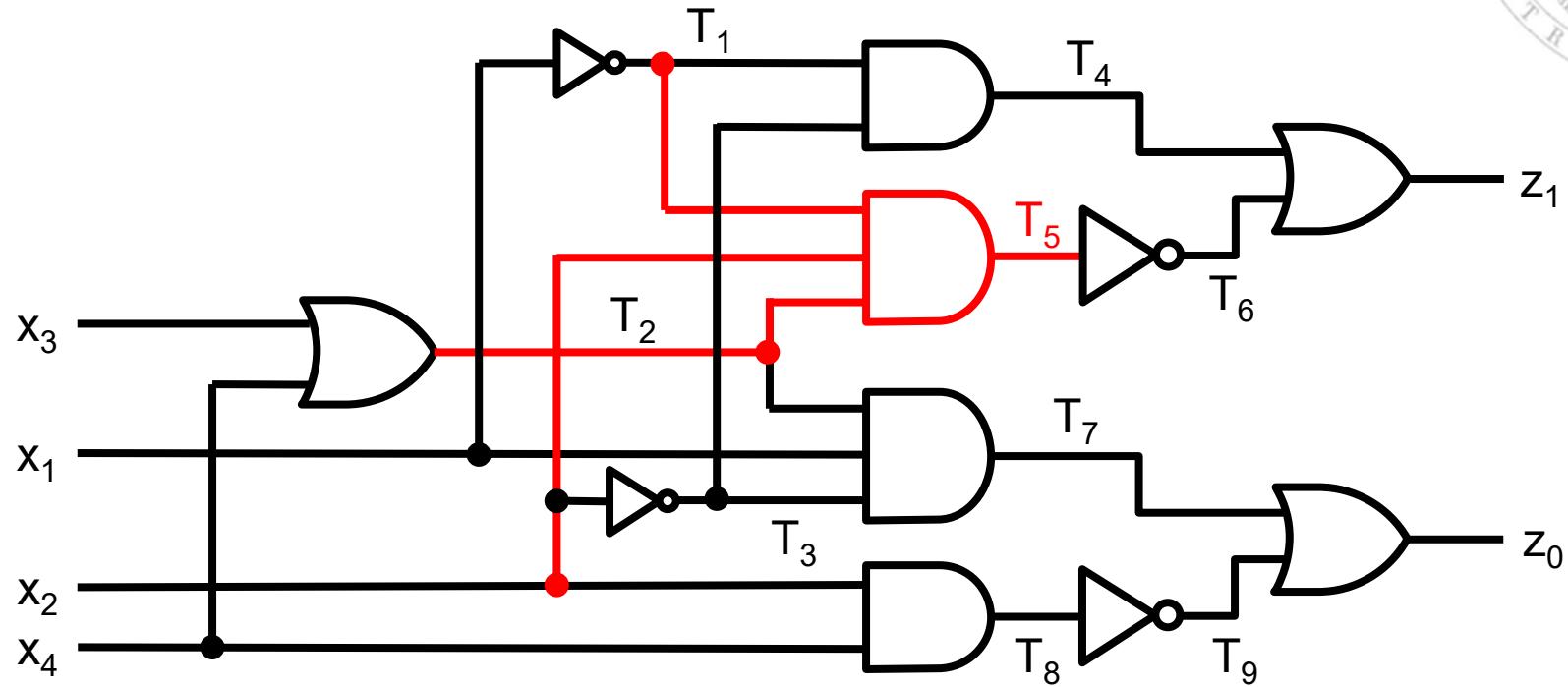
$$T_4 = T_1 T_3 = \overline{x_1} \overline{x_2}$$

$$T_2 = x_3 + x_4$$

$$T_3 = \overline{x_2}$$



Análisis de redes AND-OR



$$T_1 = \overline{x_1}$$

$$T_4 = T_1 T_3 = \overline{x_1} \overline{x_2}$$

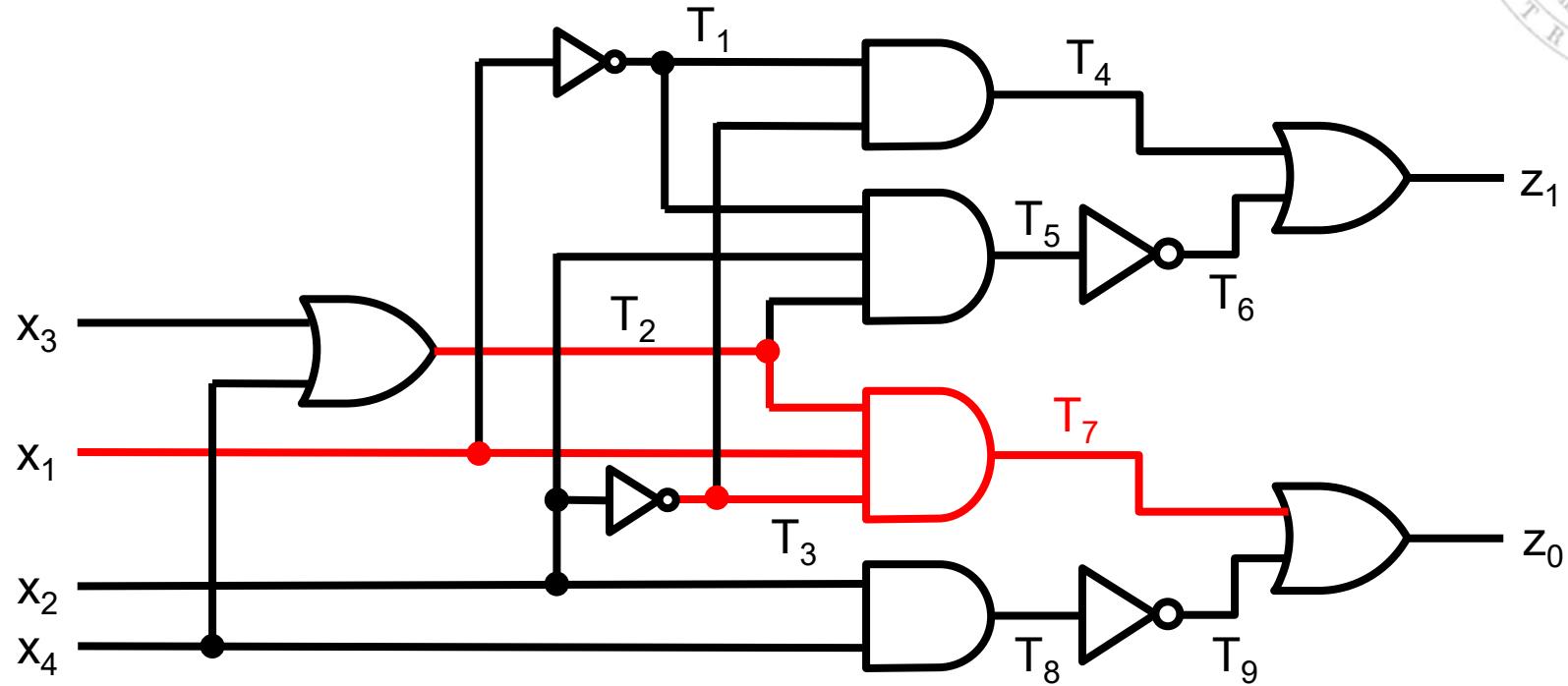
$$T_2 = x_3 + x_4$$

$$T_5 = T_1 x_2 T_2 = \overline{x_1} x_2 (x_3 + x_4)$$

$$T_3 = \overline{x_2}$$



Análisis de redes AND-OR



$$T_1 = \overline{x_1}$$

$$T_4 = T_1 T_3 = \overline{x_1} \overline{x_2}$$

$$T_2 = x_3 + x_4$$

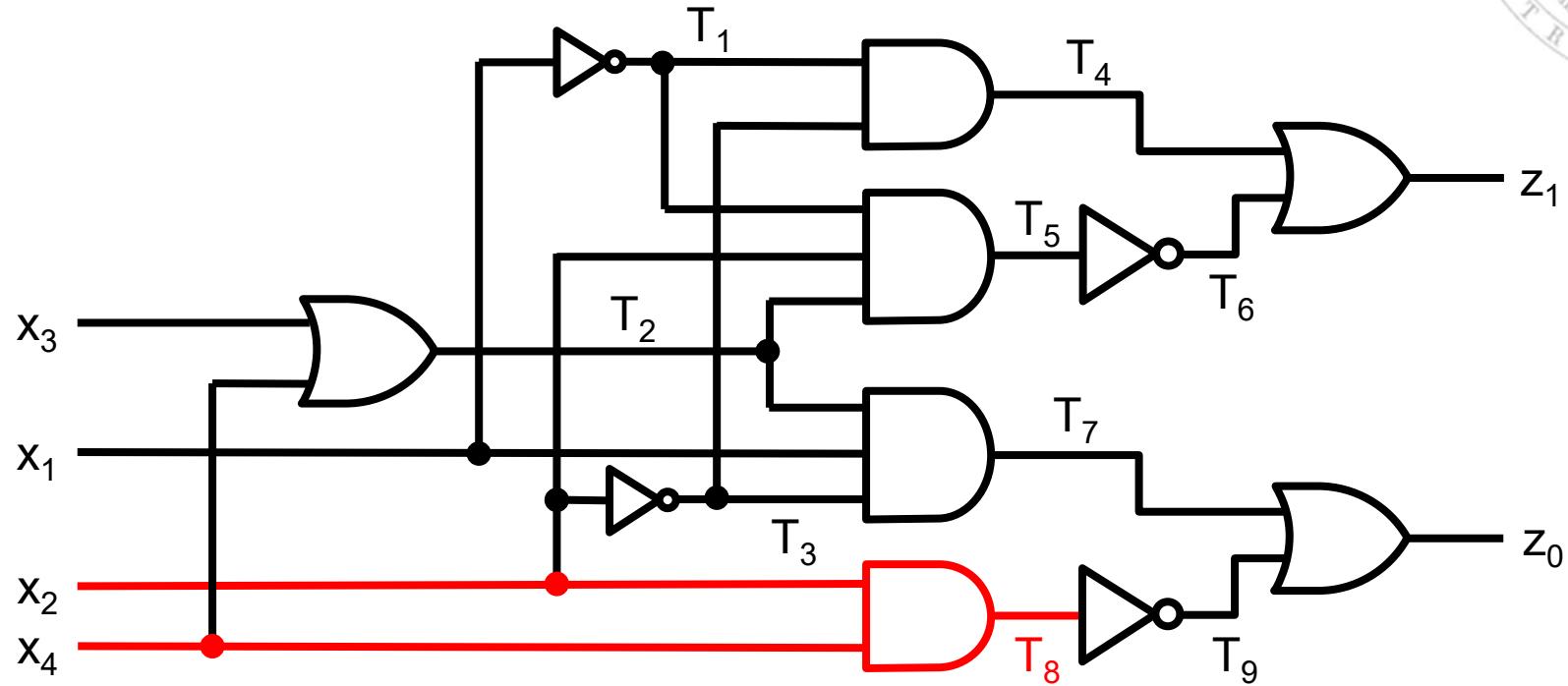
$$T_5 = T_1 x_2 T_2 = \overline{x_1} x_2 (x_3 + x_4)$$

$$T_3 = \overline{x_2}$$

$$T_7 = T_2 x_1 T_3 = (x_3 + x_4) x_1 \overline{x_2}$$



Análisis de redes AND-OR



$$T_1 = \overline{x_1}$$

$$T_4 = T_1 T_3 = \overline{x_1} \overline{x_2}$$

$$T_2 = x_3 + x_4$$

$$T_5 = T_1 x_2 T_2 = \overline{x_1} x_2 (x_3 + x_4)$$

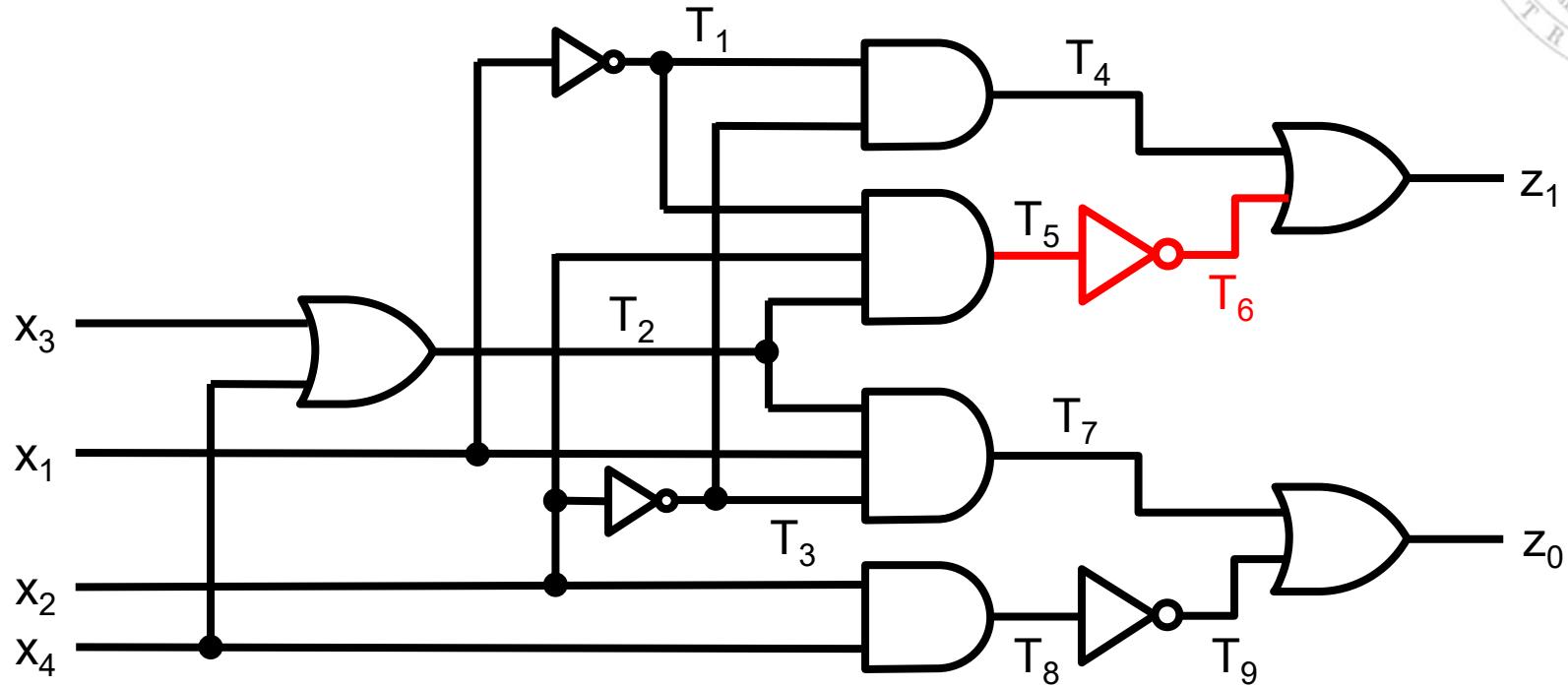
$$T_3 = \overline{x_2}$$

$$T_7 = T_2 x_1 T_3 = (x_3 + x_4) x_1 \overline{x_2}$$

$$T_8 = x_2 x_4$$



Análisis de redes AND-OR



$$T_1 = \overline{x_1}$$

$$T_4 = T_1 T_3 = \overline{x_1} \overline{x_2}$$

$$T_6 = \overline{T_5} = \overline{\overline{x_1}x_2(x_3 + x_4)}$$

$$T_2 = x_3 + x_4$$

$$T_5 = T_1 x_2 T_2 = \overline{x_1} x_2 (x_3 + x_4)$$

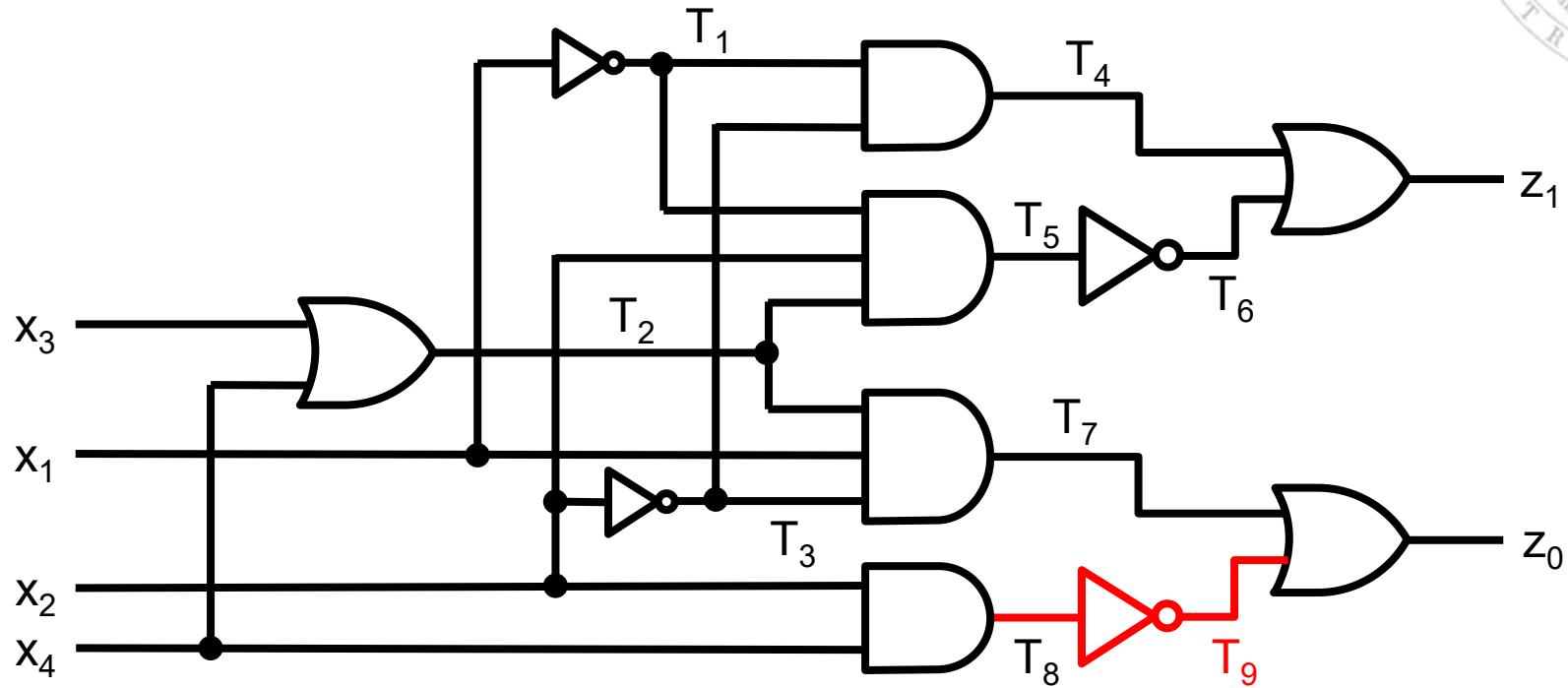
$$T_3 = \overline{x_2}$$

$$T_7 = T_2 x_1 T_3 = (x_3 + x_4) x_1 \overline{x_2}$$

$$T_8 = x_2 x_4$$



Análisis de redes AND-OR



$$T_1 = \overline{x_1}$$

$$T_2 = x_3 + x_4$$

$$T_3 = \overline{x_2}$$

$$T_8 = x_2x_4$$

$$T_4 = T_1T_3 = \overline{x_1}\overline{x_2}$$

$$T_5 = T_1x_2T_2 = \overline{x_1}x_2(x_3 + x_4)$$

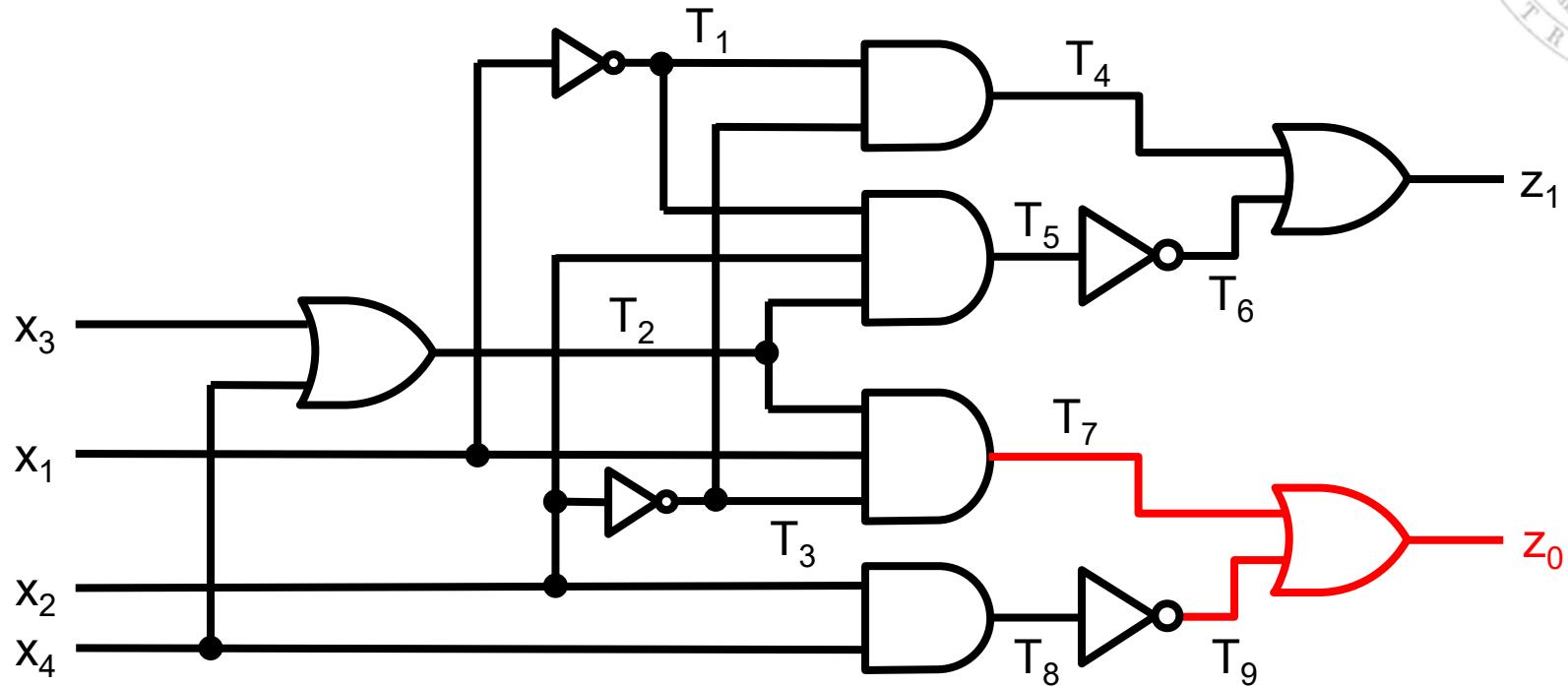
$$T_7 = T_2x_1T_3 = (x_3 + x_4)x_1\overline{x_2}$$

$$T_6 = \overline{T_5} = \overline{\overline{x_1}x_2(x_3 + x_4)}$$

$$T_9 = \overline{T_8} = \overline{x_2x_4}$$



Análisis de redes AND-OR



$$T_1 = \overline{x_1}$$

$$T_2 = x_3 + x_4$$

$$T_3 = \overline{x_2}$$

$$T_4 = T_1 T_3 = \overline{x_1} \overline{x_2}$$

$$T_5 = T_1 x_2 T_2 = \overline{x_1} x_2 (x_3 + x_4)$$

$$T_7 = T_2 x_1 T_3 = (x_3 + x_4) x_1 \overline{x_2}$$

$$T_8 = x_2 x_4$$

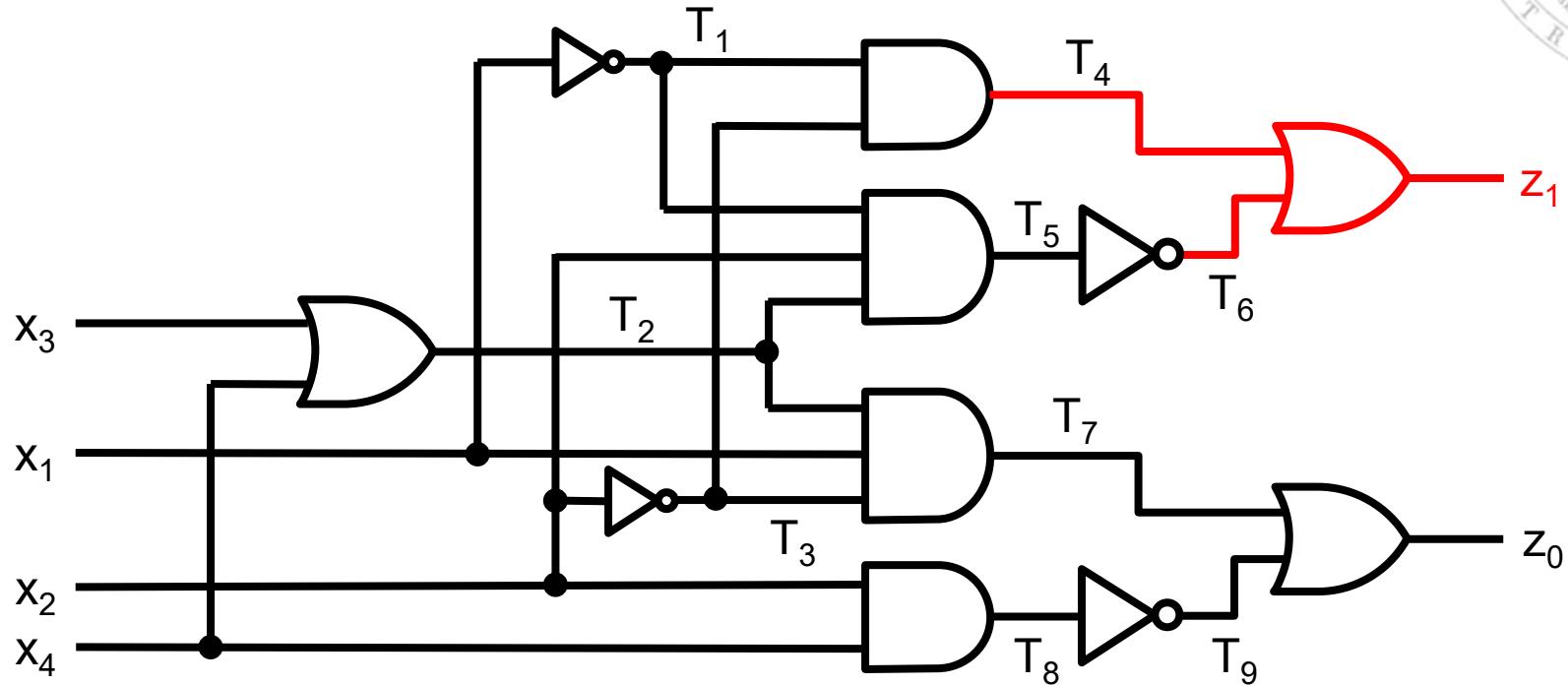
$$T_6 = \overline{T_5} = \overline{\overline{x_1} x_2 (x_3 + x_4)}$$

$$T_9 = \overline{T_8} = \overline{x_2 x_4}$$

$$z_0 = T_7 + T_9 = (x_3 + x_4) x_1 \overline{x_2} + \overline{x_2 x_4}$$



Análisis de redes AND-OR



$$T_1 = \overline{x_1}$$

$$T_2 = x_3 + x_4$$

$$T_3 = \overline{x_2}$$

$$T_4 = T_1 T_3 = \overline{x_1} \overline{x_2}$$

$$T_5 = T_1 x_2 T_2 = \overline{x_1} x_2 (x_3 + x_4)$$

$$T_7 = T_2 x_1 T_3 = (x_3 + x_4) x_1 \overline{x_2}$$

$$T_8 = x_2 x_4$$

$$T_6 = \overline{T_5} = \overline{\overline{x_1} x_2 (x_3 + x_4)}$$

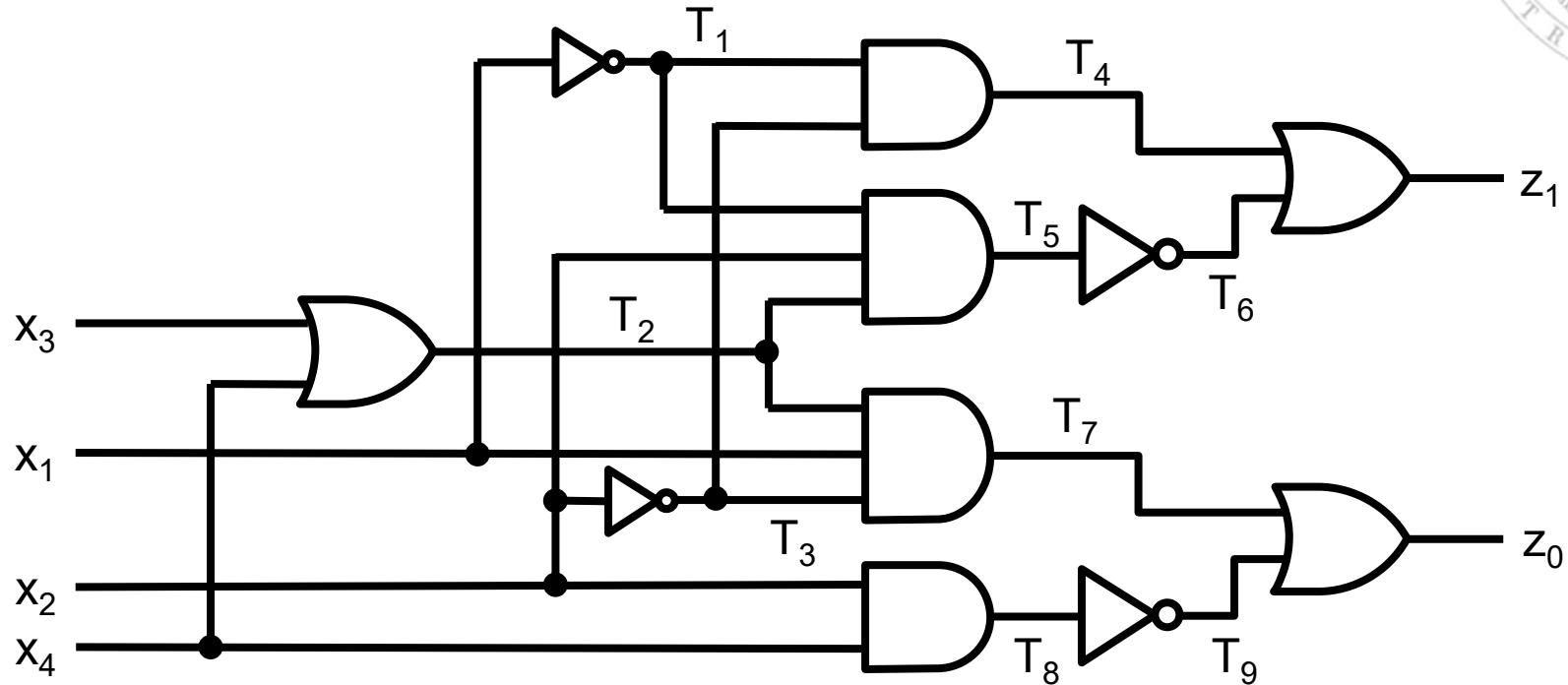
$$T_9 = \overline{T_8} = \overline{x_2 x_4}$$

$$z_0 = T_7 + T_9 = (x_3 + x_4) x_1 \overline{x_2} + \overline{x_2 x_4}$$

$$z_1 = T_4 + T_6 = \overline{x_1} \overline{x_2} + \overline{x_1} x_2 (x_3 + x_4)$$



Análisis de redes AND-OR



$$T_1 = \overline{x_1}$$

$$T_2 = x_3 + x_4$$

$$T_3 = \overline{x_2}$$

$$T_4 = T_1 T_3 = \overline{x_1} \overline{x_2}$$

$$T_5 = T_1 x_2 T_2 = \overline{x_1} x_2 (x_3 + x_4)$$

$$T_7 = T_2 x_1 T_3 = (x_3 + x_4) x_1 \overline{x_2}$$

$$T_8 = x_2 x_4$$

$$T_6 = \overline{T_5} = \overline{\overline{x_1} x_2 (x_3 + x_4)}$$

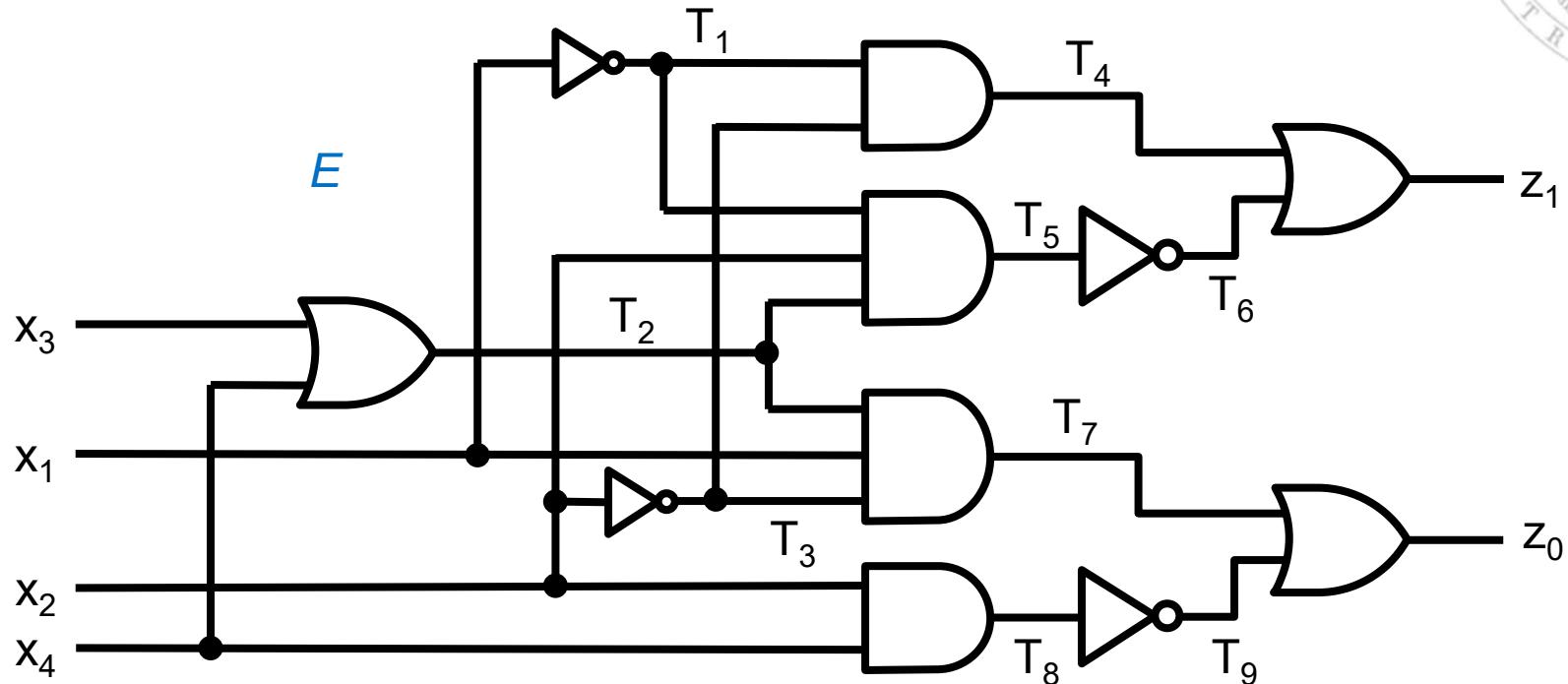
$$T_9 = \overline{T_8} = \overline{x_2 x_4}$$

$$z_0 = T_7 + T_9 = (x_3 + x_4) x_1 \overline{x_2} + \overline{x_2 x_4}$$

$$z_1 = T_4 + T_6 = \overline{x_1} \overline{x_2} + \overline{x_1} x_2 (x_3 + x_4)$$



Análisis de redes AND-OR



$$z_0 = \overline{x_2} + \overline{x_4}$$

$$z_1 = x_1 + \overline{x_2} + \overline{x_3} \overline{x_4}$$

Por absorción



Contenidos

- ✓ Sistema combinacional
- ✓ Expresiones de conmutación.
- ✓ Forma canónica. Suma de productos.
- ✓ Simplificación por mapas de Karnaugh
- ✓ Puertas lógicas: Síntesis con redes AND-OR
- ✓ Análisis de redes de puertas.
- ✓ Módulos combinacionales básicos: Decodificador, mux...
- ✓ ROM
- ✓ Módulos aritméticos



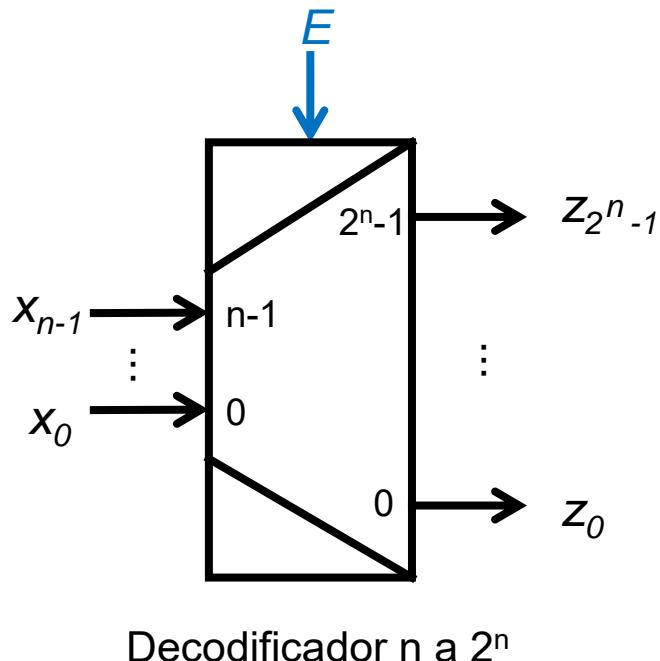
Decodificador

versión 2021

tema 2:
Sistemas combinacionales

FC

114



\underline{x} n entradas de datos

z 2^n salidas de datos

E 1 entrada de capacitación (op)

si la entrada toma la configuración binaria p , la salida $(p)_{10}$ -ésima se activa

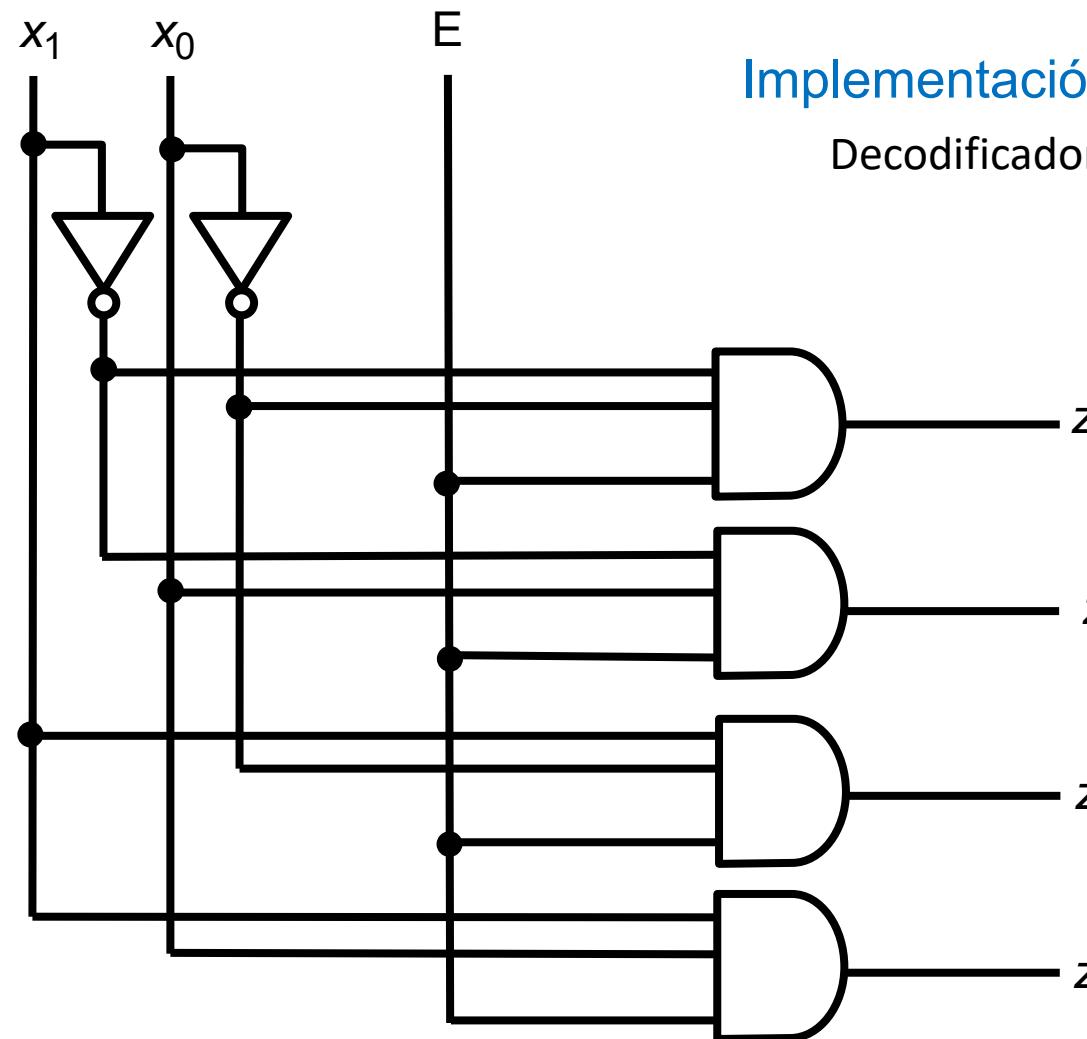
$$z_i = \begin{cases} 1 & \text{si } E=1 \text{ y } (\underline{x})_{10} = i \\ 0 & \text{en caso contrario} \end{cases}$$

$$z_i = E \cdot m_i(\underline{x})$$

Mintérmino



Decodificador

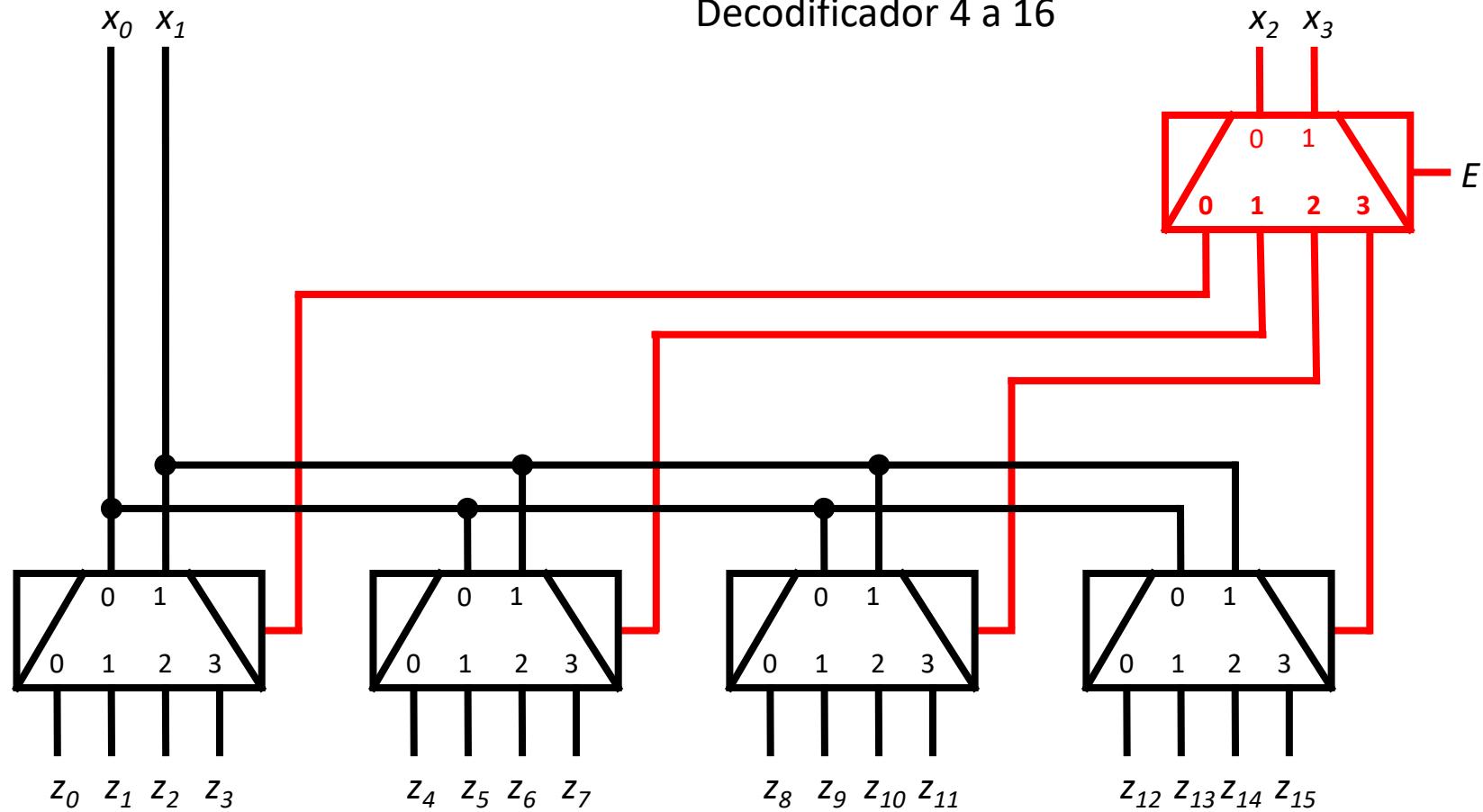




Decodificador

Implementación en árbol

Decodificador 4 a 16

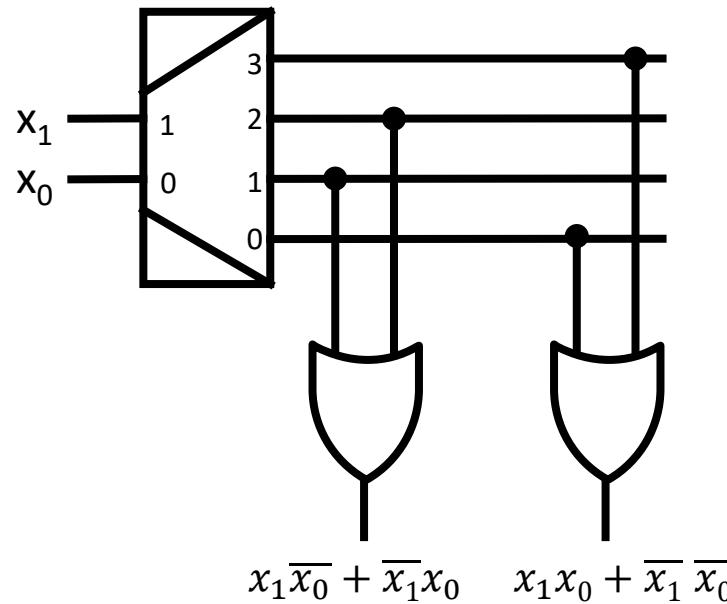


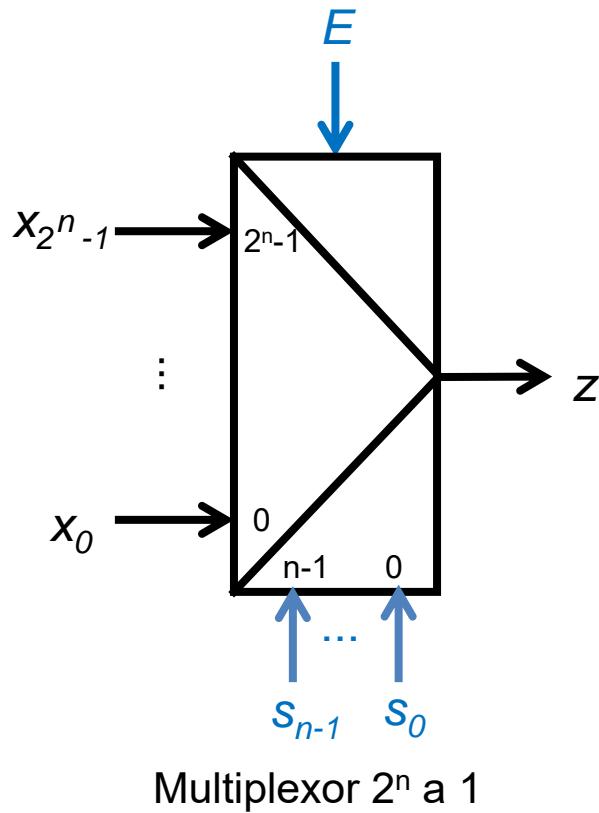


Decodificador

■ Aplicaciones al diseño:

1. Habilitar selectivamente **1 de n** subcomponentes cada uno asociado a un índice (dirección) binaria.
2. Implementar directamente SPC usando puertas OR adicionales (que sumen cada unos de los mintérminos de la FC).





Multiplexor



x 2ⁿ entradas de datos

s n entradas de control

E 1 entrada de capacitación (opcional)

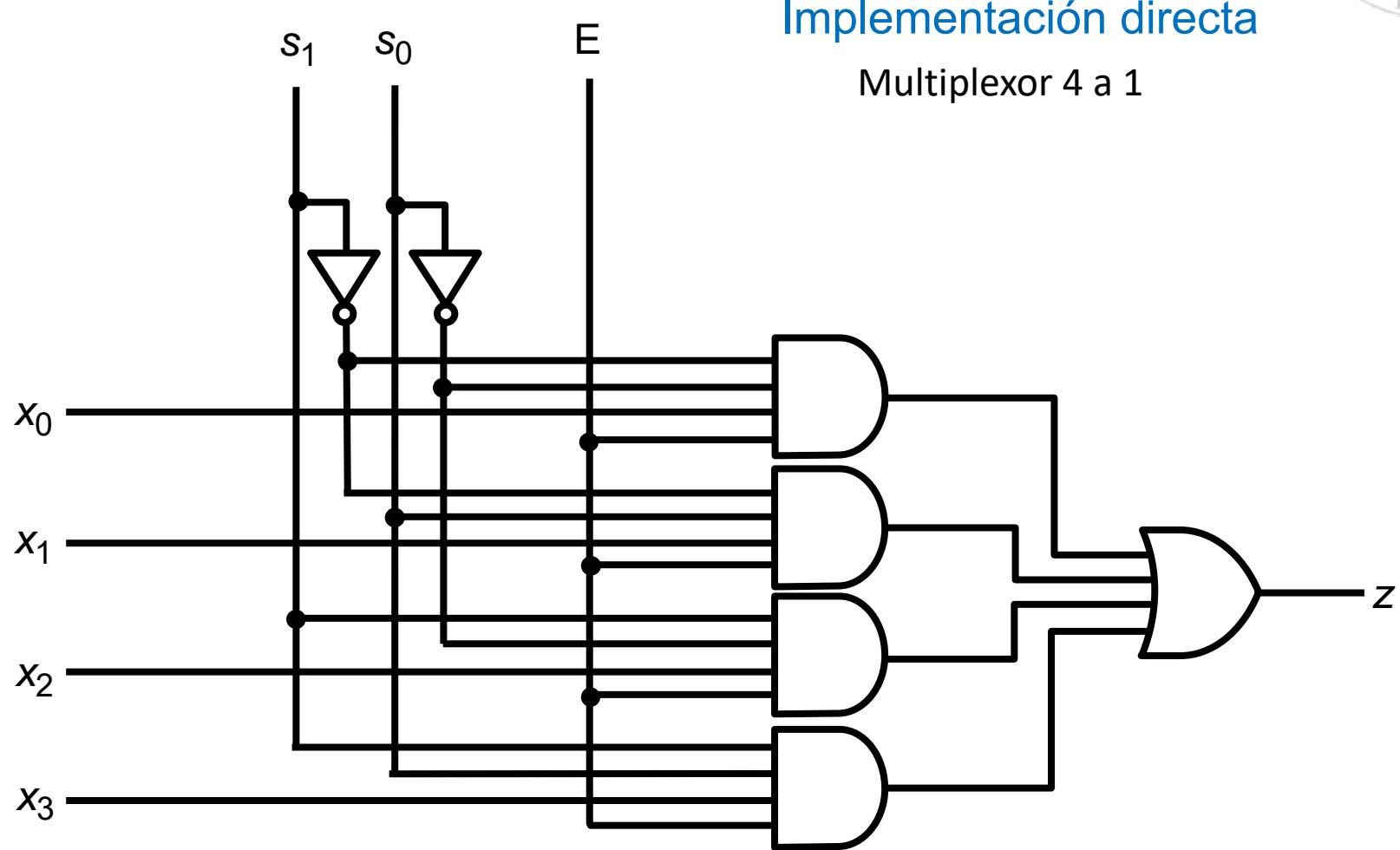
z 1 salida de datos

si la entrada de control toma la configuración binaria p, la salida equivale a la entrada (p)₁₀-ésima

$$z = \begin{cases} x_i & \text{si } E=1 \text{ y } (s)_{10} = i \\ 0 & \text{en caso contrario} \end{cases}$$



Multiplexor

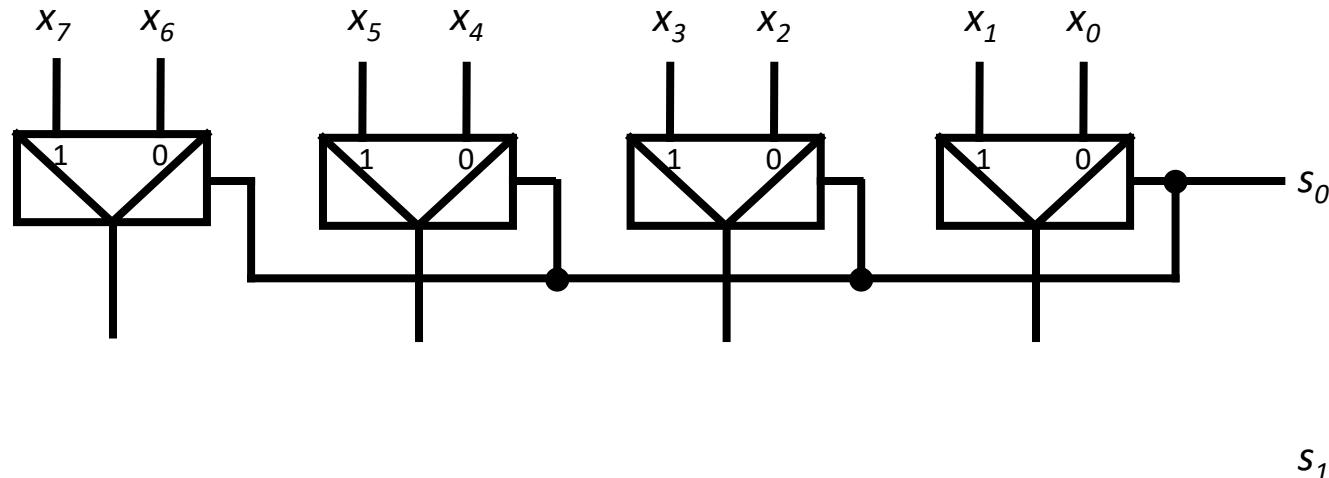




Multiplexor

Implementación en árbol

Multiplexor 8 a 1

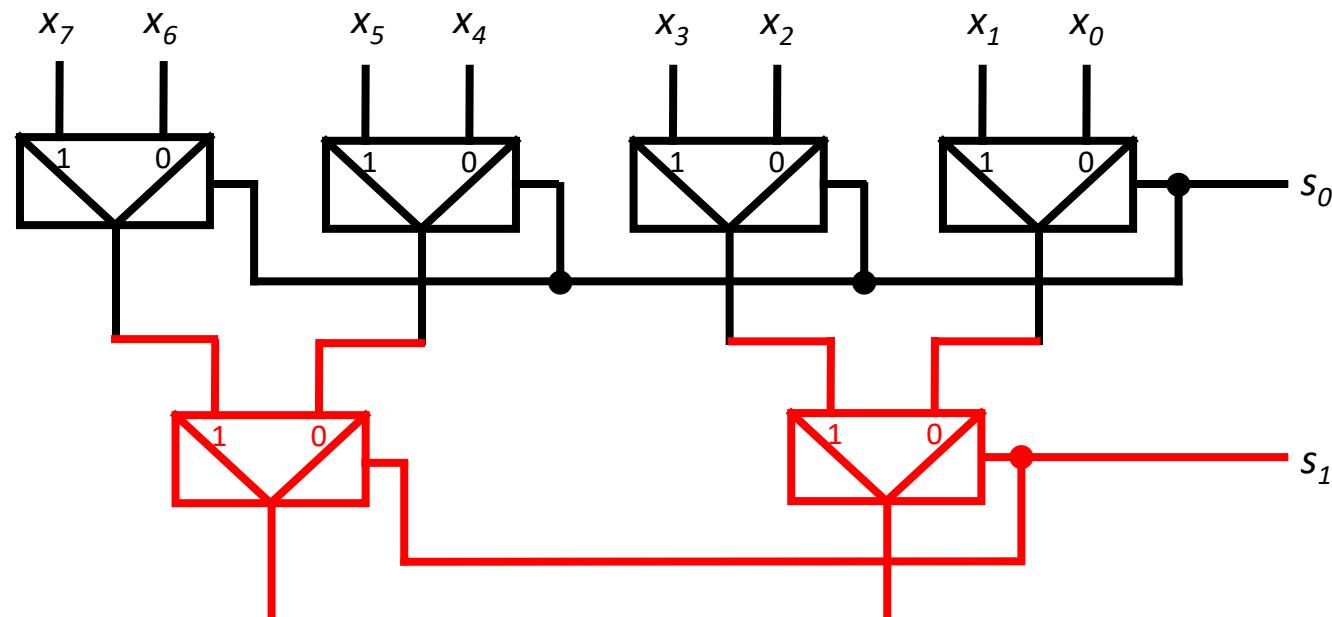




Multiplexor

Implementación en árbol

Multiplexor 8 a 1

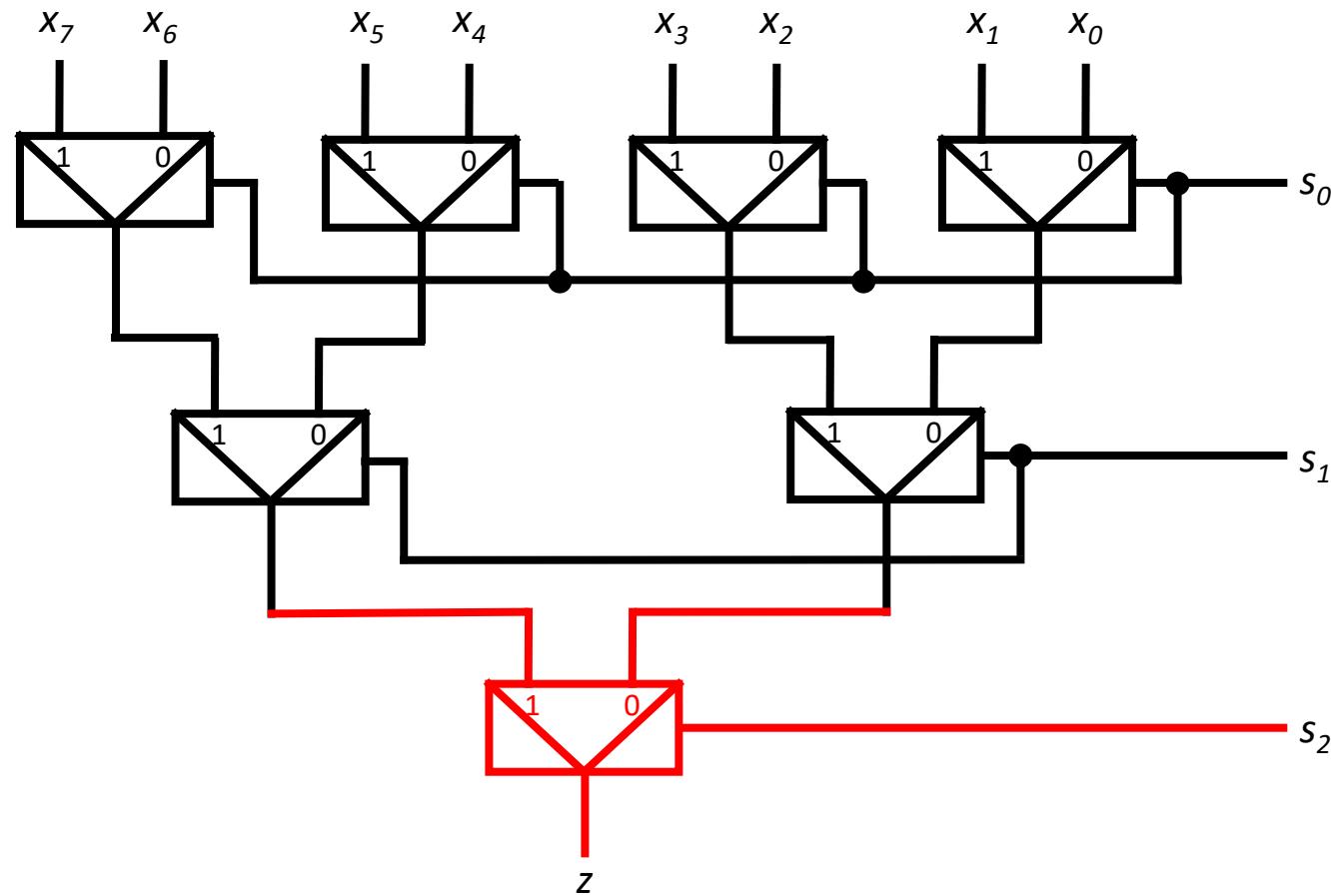




Multiplexor

Implementación en árbol

Multiplexor 8 a 1

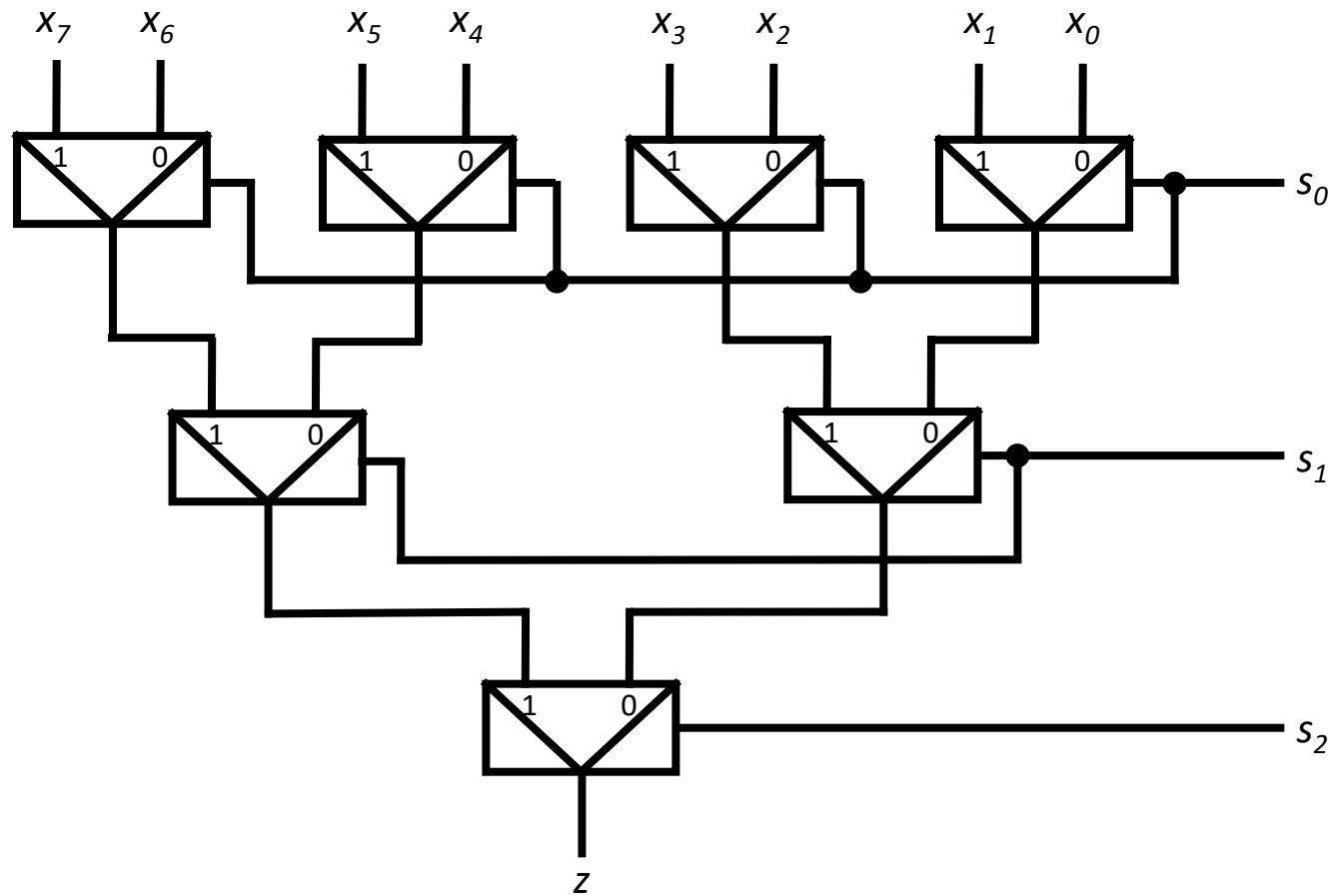




Multiplexor

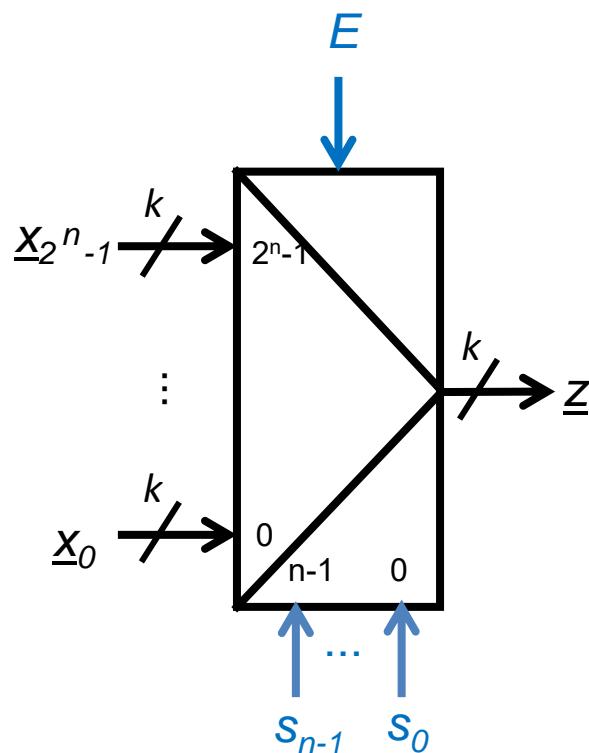
Implementación en árbol

Multiplexor 8 a 1





Multiplexor vectorial



Multiplexor 2^n a 1 de k bits

-
- \underline{x} 2^n entradas de datos de k bits
 - \underline{s} n entradas de control
 - E 1 entrada de capacitación (op)
 - \underline{z} 1 salida de datos de k bits
-

si la entrada de control toma la configuración binaria p , la salida equivale a la entrada $(p)_{10}$ -ésima

$$\underline{z} = \begin{cases} \underline{x}_i & \text{si } E=1 \text{ y } (\underline{s})_{10} = i \\ 0 & \text{en caso contrario} \end{cases}$$



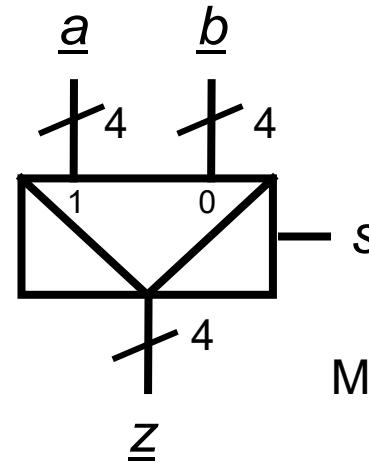
Multiplexor vectorial

versión 2021

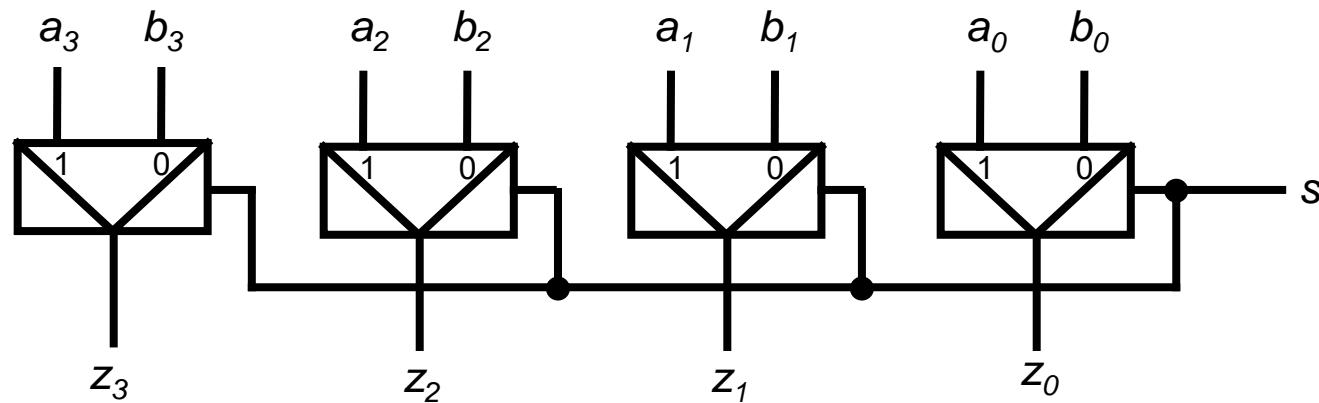
tema 2:
Sistemas combinacionales

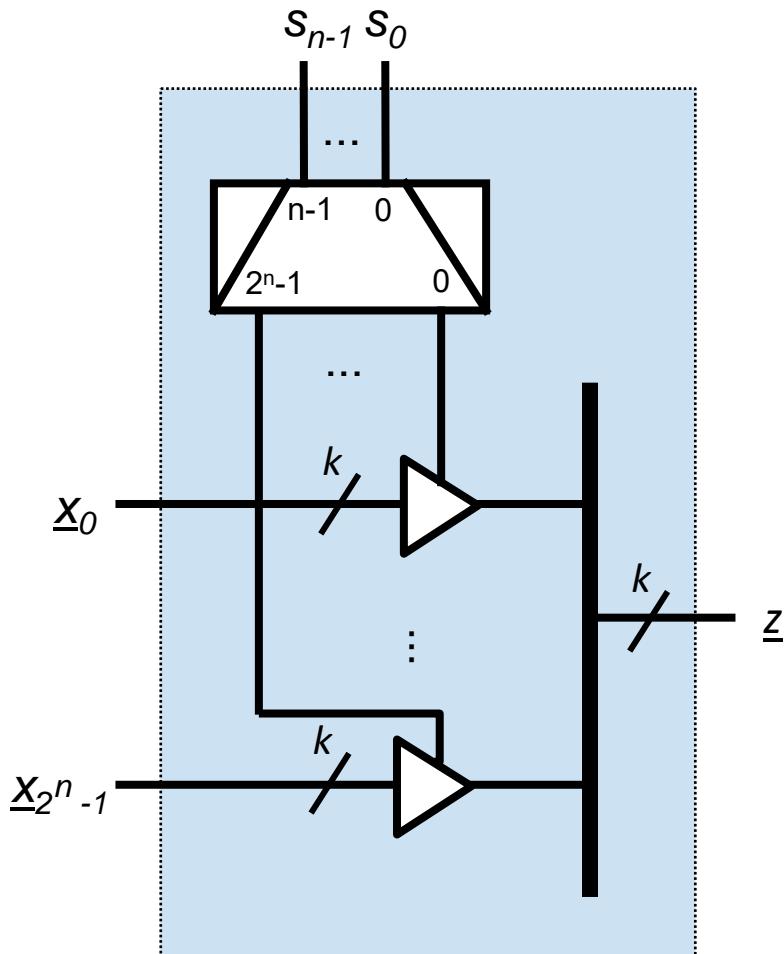
FC

125



Multiplexor 2 a 1 de 4 bits





Bus



-
- x 2^n entradas de datos de k bits
 - s n entradas de control
 - z 1 salida de datos de k bits
-

si la entrada de control toma la configuración binaria p , la salida equivale a la entrada $(p)_{10}$ -ésima



Contenidos

- ✓ Sistema combinacional
- ✓ Expresiones de conmutación.
- ✓ Forma canónica. Suma de productos.
- ✓ Simplificación por mapas de Karnaugh
- ✓ Puertas lógicas: Síntesis con redes AND-OR
- ✓ Análisis de redes de puertas.
- ✓ Módulos combinacionales básicos: Decodificador, mux...
- ✓ ROM
- ✓ Módulos aritméticos



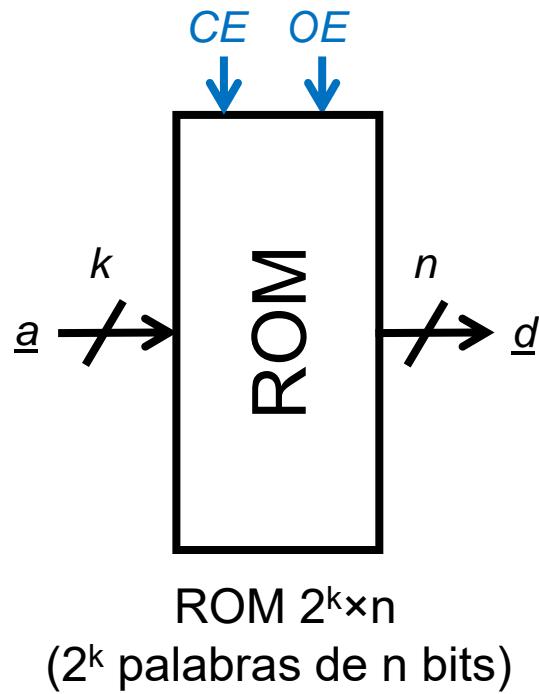
ROM (Read Only Memory)

versión 2021

tema 2:
Sistemas combinacionales

FC

128



-
- a 1 entrada de dirección de k bits
 - d 1 salida de datos de n bits
 - CE 1 entrada de capacitación (op)
 - OE 1 entrada de capacitación de lectura (op)
-

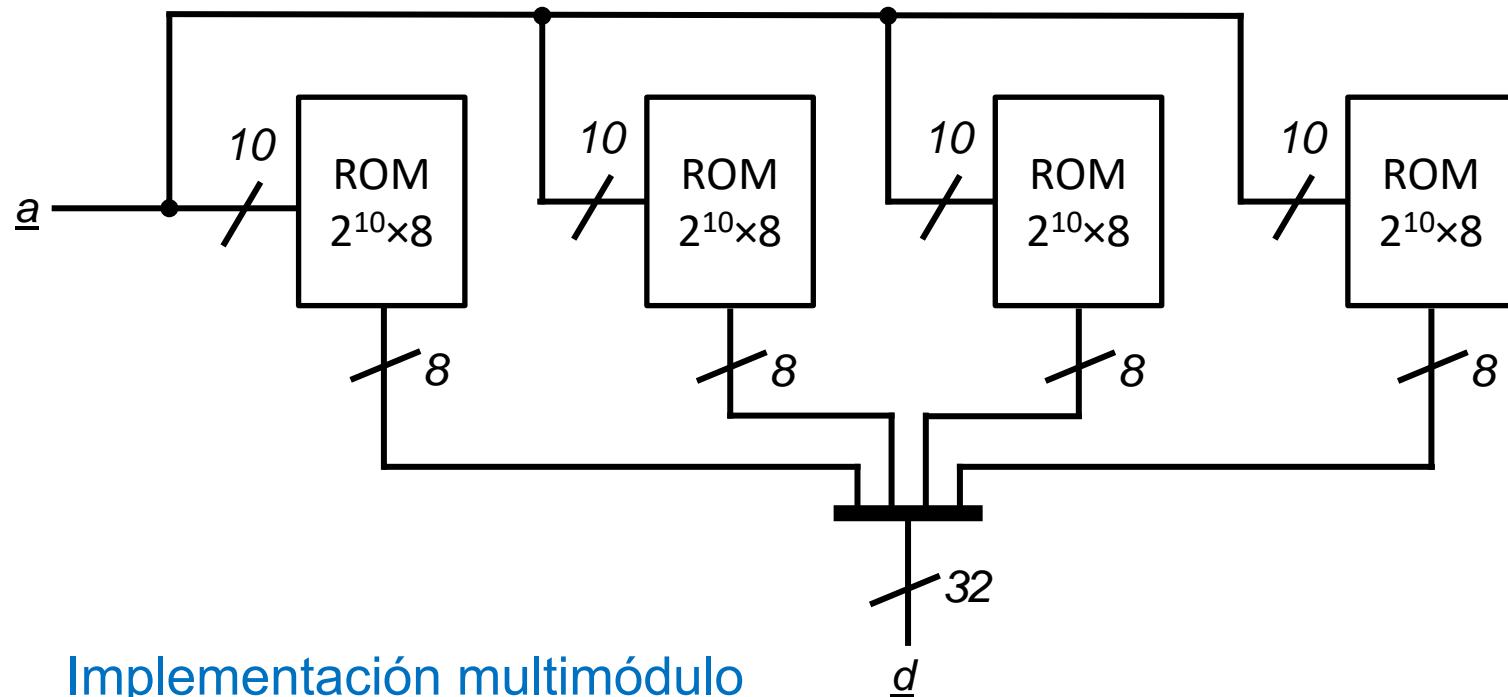
dispositivo programable capaz de implementar n FC de k variables almacenando sus tablas de verdad

memoria no volátil capaz de almacenar 2^k palabras de n bits cada una



ROM (Read Only Memory)

- Varias ROM se pueden componer para comportarse como una ROM de mayor anchura de palabra.



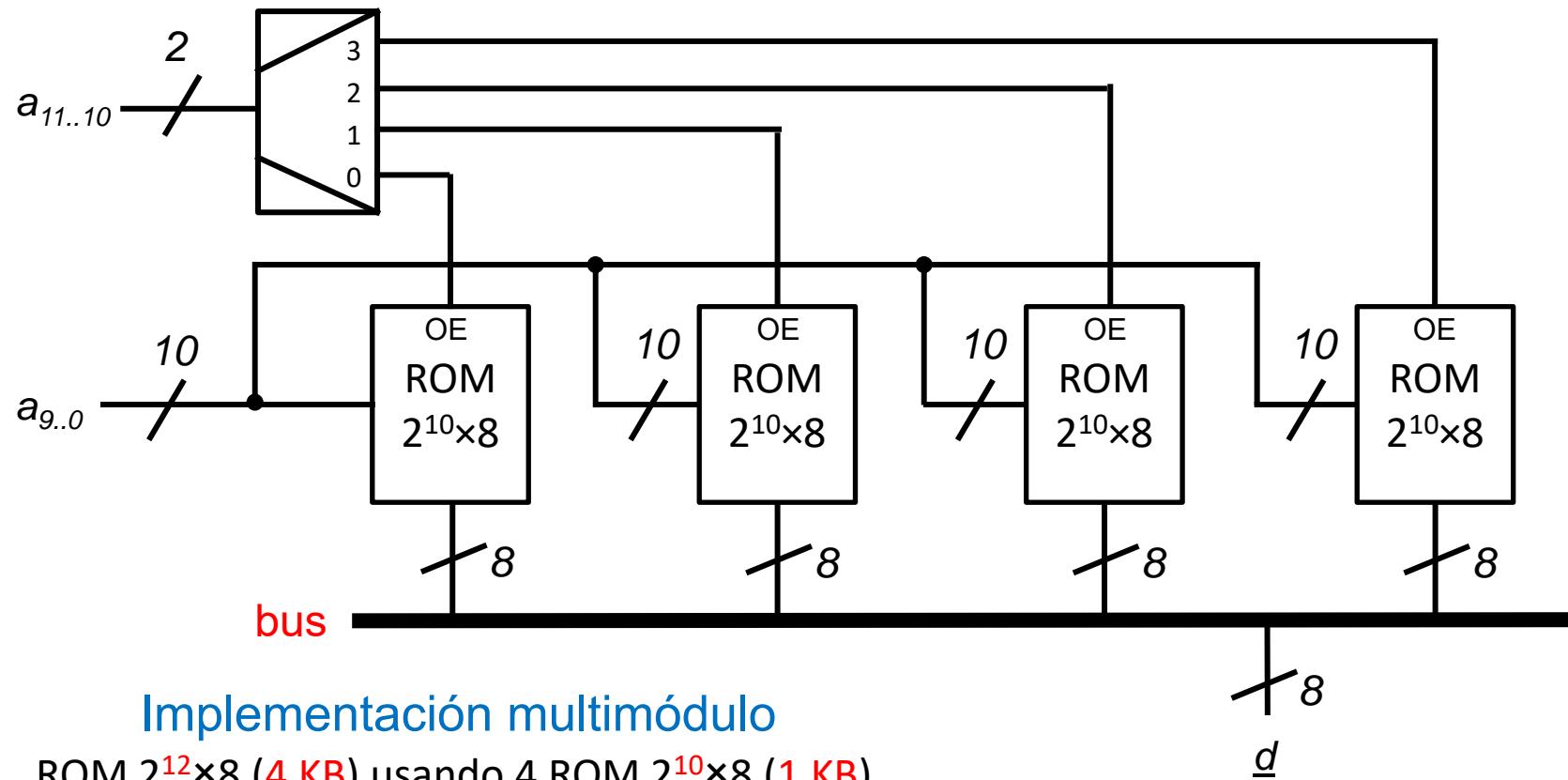
Implementación multimódulo

ROM $2^{10} \times 32$ (4 KB) usando 4 ROM $2^{10} \times 8$ (1 KB)

ROM (Read Only Memory)



- Varias ROM se pueden componer para comportarse como una ROM de mayor profundidad.



Implementación multimódulo

ROM $2^{12} \times 8$ (4 KB) usando 4 ROM $2^{10} \times 8$ (1 KB)

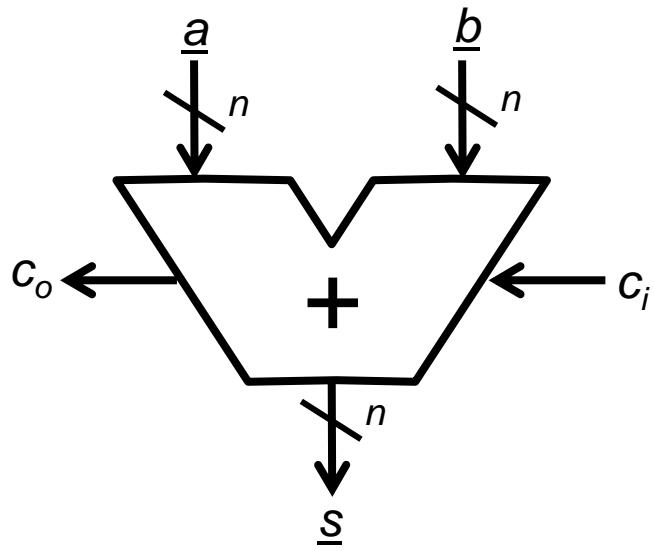


Contenidos

- ✓ Sistema combinacional
- ✓ Expresiones de conmutación.
- ✓ Forma canónica. Suma de productos.
- ✓ Simplificación por mapas de Karnaugh
- ✓ Puertas lógicas: Síntesis con redes AND-OR
- ✓ Análisis de redes de puertas.
- ✓ Módulos combinacionales básicos: Decodificador, mux...
- ✓ ROM
- ✓ **Módulos aritméticos**



Sumador



a, b 2 entradas de datos de n bits

c_i 1 entrada de acarreo

s 1 salida de datos de n bits

c_o 1 salida de acarreo

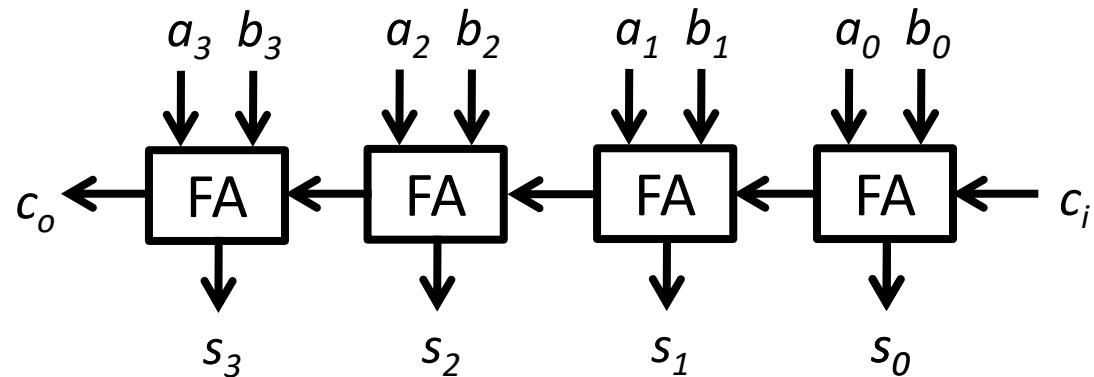
realiza la suma binaria de a + b + c_i

$$s = (a + b + c_i) \bmod 2^n$$

$$c_o = \begin{cases} 1 & (a + b + c_i) \geq 2^n \\ 0 & \text{en caso contrario} \end{cases}$$



Sumador



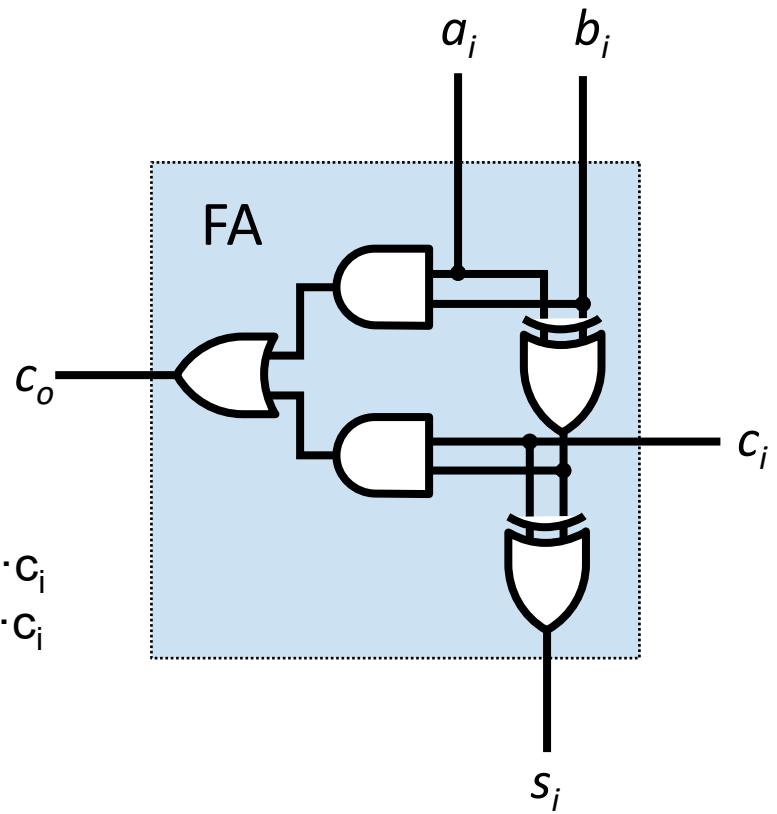
ci	ai	bi	co	si
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$s_i = (a_i \oplus b_i) \oplus c_i$$

$$\begin{aligned} c_o &= a_i \cdot b_i + a_i \cdot c_i + b_i \cdot c_i \\ &= a_i \cdot b_i + (a_i \oplus b_i) \cdot c_i \end{aligned}$$

Implementación con
propagación de acarreos

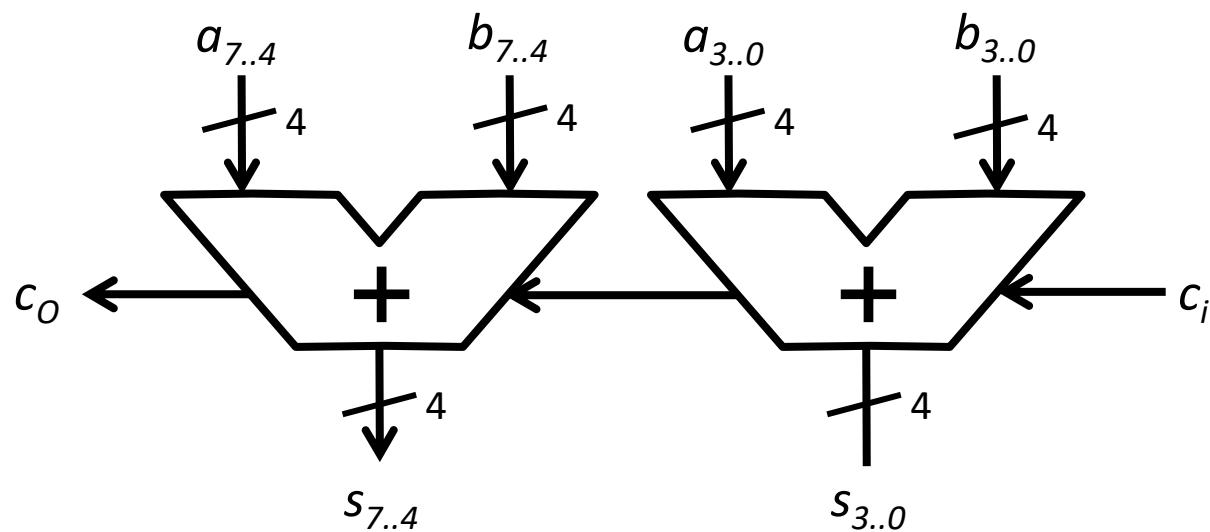
Sumador de 4 bits





Sumador

- Varios sumadores se pueden componer en serie para comportarse como un sumador de **mayor anchura**.

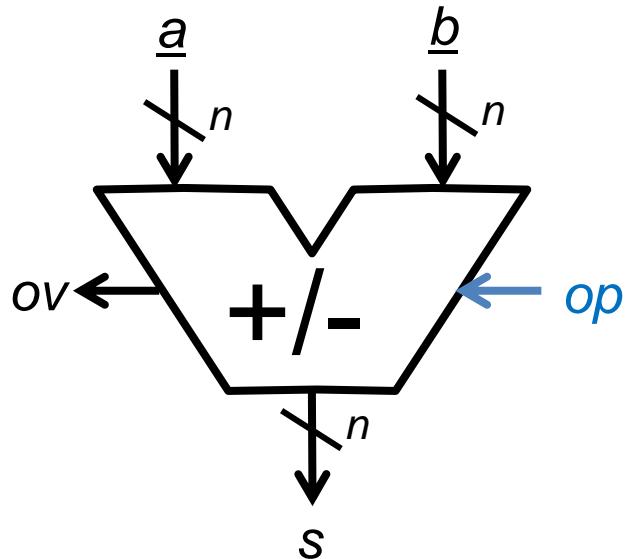


Implementación serie

Sumador de 8 bits



Sumador/restador



-
- $\underline{a}, \underline{b}$ 2 entradas de datos de n bits
op 1 entrada de selección de operación
s 1 salida de datos de n bits
ov 1 salida de overflow
-

realiza la suma/resta en \underline{a} y \underline{b}
(interpretados en C2)

$$\underline{a} - \underline{b} = \underline{a} + (-\underline{b}) = \underline{a} + C2(\underline{b}) = \underline{a} + \overline{\underline{b}} + 1$$

$$s = \begin{cases} \underline{a} + \underline{b} & \text{si } op = 0 \\ \underline{a} + \overline{\underline{b}} + 1 & \text{si } op = 1 \end{cases} = \underline{a} + (\underline{b} \oplus op) + op$$

$$ov = \begin{cases} 1 & (b_{n-1} \oplus op) = 0 \text{ y } a_{n-1} = 0 \text{ y } s_{n-1} = 1 \\ & \text{ó} \\ & (b_{n-1} \oplus op) = 1 \text{ y } a_{n-1} = 1 \text{ y } s_{n-1} = 0 \\ 0 & \text{en caso contrario} \end{cases}$$



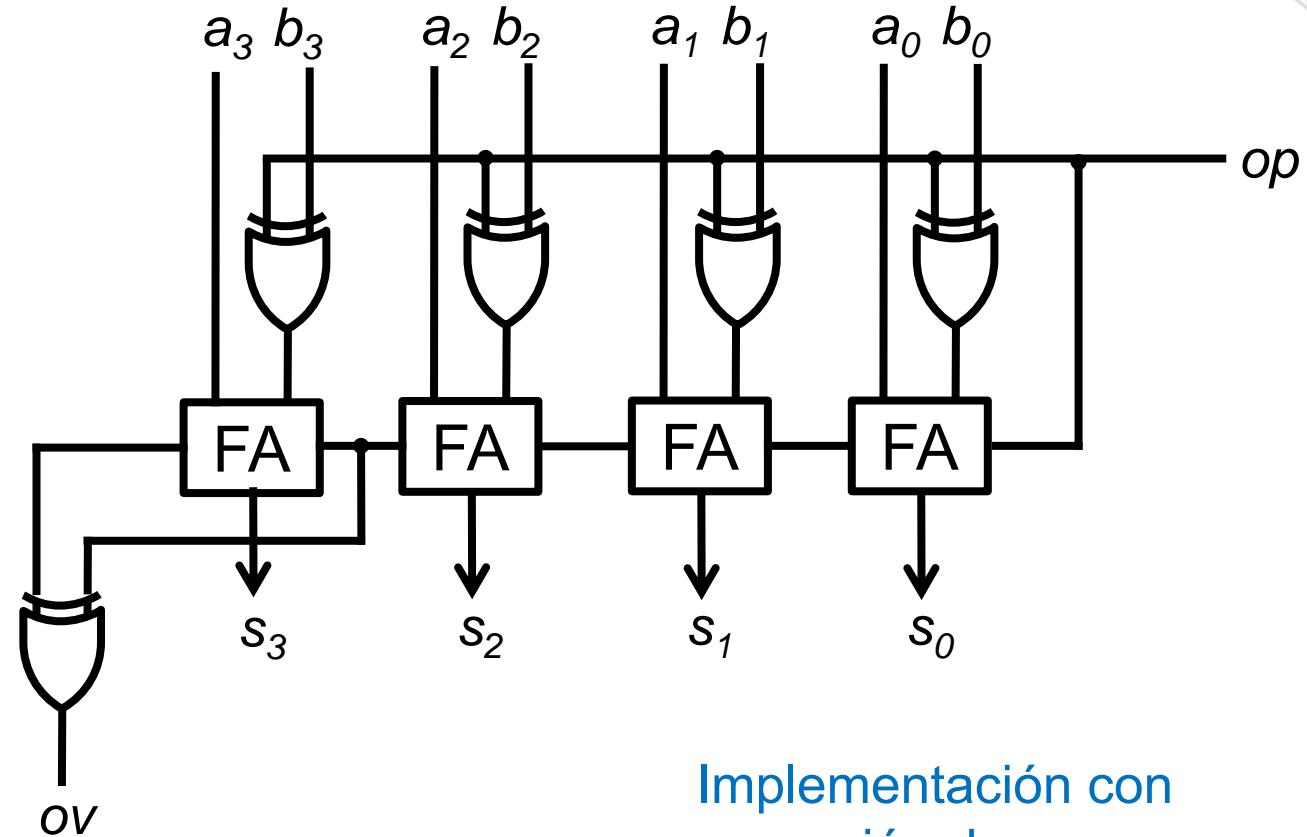
Sumador/restador

versión 2021

tema 2:
Sistemas combinacionales

FC

136



Implementación con
propagación de acarreos
Sumador/restador de 4 bits



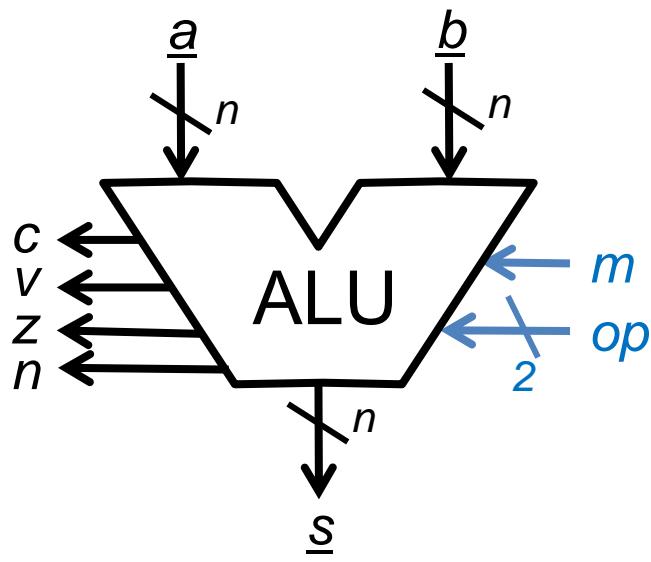
ALU (Arithmetic-Logic Unit)

versión 2021

tema 2:
Sistemas combinacionales

FC

137



a, b 2 entradas de datos de n bits

m 1 entrada de selección de modo

op 1 entrada de selección de operación

s 1 salida de datos de n bits

c 1 salida de acarreo

v 1 salida de overflow

z 1 salida de detección de cero

n 1 salida de detección de negativo

operaciones lógicas

m	op ₁	op ₀	z
0	0	0	not(<u>a</u>)
0	0	1	and(<u>a</u> , <u>b</u>)
0	1	0	<u>a</u>
0	1	1	or(<u>a</u> , <u>b</u>)

operaciones aritméticas

m	op ₁	op ₀	z
1	0	0	<u>a</u> + <u>b</u>
1	0	1	<u>a</u> - 1 = <u>a</u> + (-1) = _{C2} <u>a</u> + <u>1</u>
1	1	0	<u>a</u> - <u>b</u>
1	1	1	<u>a</u> + 1 = <u>a</u> - (-1) = _{C2} <u>a</u> - <u>1</u>



ALU (Arithmetic-Logic Unit)

