

Tema 3

ANÁLISIS Y DISEÑO DE SISTEMAS COMBINACIONALES

*TECNOLOGÍA Y ORGANIZACIÓN DE
COMPUTADORES*

1º Grado en Ingeniería Informática.

Tema 3. Análisis y diseño de sistemas combinacionales

RESUMEN:

En este tema se va a tratar lo qué es un sistema combinacional y cómo se diseñan y analizan circuitos combinacionales sencillos. También se analizarán algunos bloques combinacionales que realizan funciones más complejas y que no se pueden analizar a nivel de puertas lógicas.

OBJETIVOS:

Comprender lo qué es un sistema combinacional. Saber diseñar y analizar sistemas combinacionales sencillos. Entender el funcionamiento de los principales componentes combinacionales estándar.

Tema 3. Análisis y diseño de sistemas combinacionales

CONTENIDOS:

3.1 CONCEPTO DE SISTEMA COMBINACIONAL

3.2 ANÁLISIS DE CIRCUITOS COMBINACIONALES

3.3 DISEÑO DE CIRCUITOS COMBINACIONALES

3.4 COMPONENTES COMBINACIONALES ESTÁNDAR

BIBLIOGRAFÍA:

[LLOR,03] Capítulo 4, 5, 6

[GAJS,97] Capítulo 5

[FLOY,00] Capítulo 6, 7

Tema 3. Análisis y diseño de sistemas combinacionales

CONTENIDOS:

3.1 CONCEPTO DE SISTEMA COMBINACIONAL

3.2 ANÁLISIS DE CIRCUITOS COMBINACIONALES

3.3 DISEÑO DE CIRCUITOS COMBINACIONALES

3.4 COMPONENTES COMBINACIONALES ESTÁNDAR

- 3.4.1 Circuitos aritméticos (sumador/restador, comparador)

- 3.4.2 ALU

- 3.4.3 Codificadores/ Decodificadores

- 3.4.4 Multiplexores/ Demultiplexores

- 3.4.5 Dispositivos lógicos programables

3.1 CONCEPTO DE SISTEMA COMBINACIONAL

- **Sistema combinacional:** sistema en el que las salidas en cualquier instante dependen sólo de los valores de las entradas en ese mismo instante.

$$z_i(t) = f_i(x_1(t), x_2(t), \dots, x_n(t))$$

- En realidad, el valor de las salidas en un instante dado depende de la combinación de valor de las entradas presente un cierto tiempo antes, que corresponde al retardo de propagación del sistema

$$Z_i(t) = f_i(x_1(t - t_{pd}), x_2(t - t_{pd}), \dots, x_n(t - t_{pd}))$$

Tema 3. Análisis y diseño de sistemas combinacionales

CONTENIDOS:

3.1 CONCEPTO DE SISTEMA COMBINACIONAL

3.2 ANÁLISIS DE CIRCUITOS COMBINACIONALES

3.3 DISEÑO DE CIRCUITOS COMBINACIONALES

3.4 COMPONENTES COMBINACIONALES ESTÁNDAR

3.4.1 Circuitos aritméticos (sumador/restador, comparador)

3.4.2 ALU

3.4.3 Codificadores/ Decodificadores

3.4.4 Multiplexores/ Demultiplexores

3.4.5 Dispositivos lógicos programables

3.2 Análisis de circuitos combinacionales

Análisis de sistemas combinacionales:

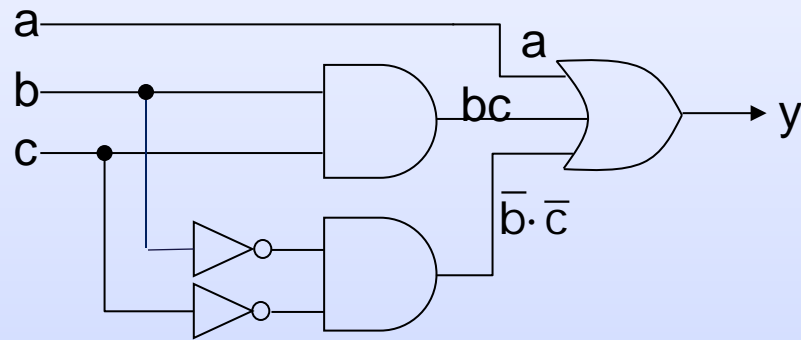
- **Funcional:** Deducir la función lógica que realiza el circuito. Se representa con tablas de verdad, expresiones booleanas o mapas de Karnaugh.
- **Temporal:** Conocer el retardo de propagación y el valor de cada nodo en respuesta a una secuencia de entradas. El comportamiento dinámico del circuito se representa mediante cronogramas.

3.2 Análisis de circuitos combinacionales

- **Ejemplo:**

Análisis funcional:

Tabla verdad			
a	b	c	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



Expresión algebraica:

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

3.2 Análisis de circuitos combinacionales

- **Ejemplo:**

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

Análisis temporal:

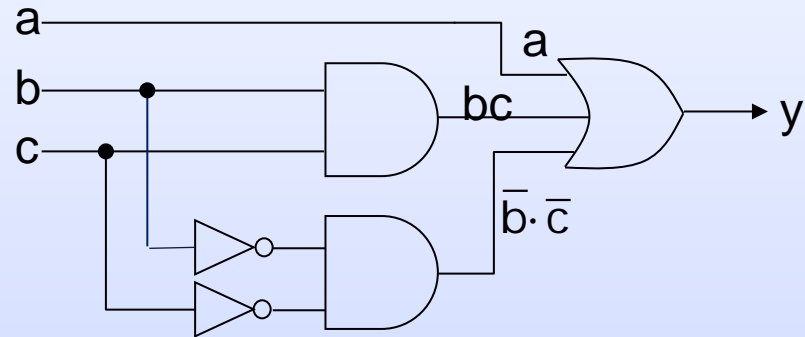
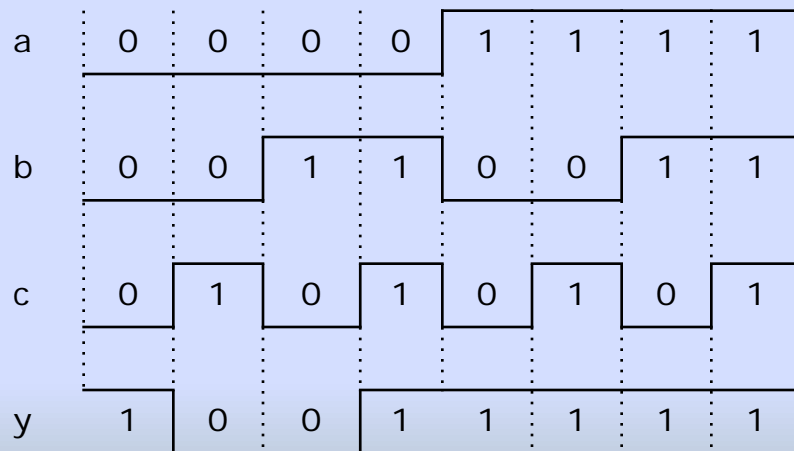
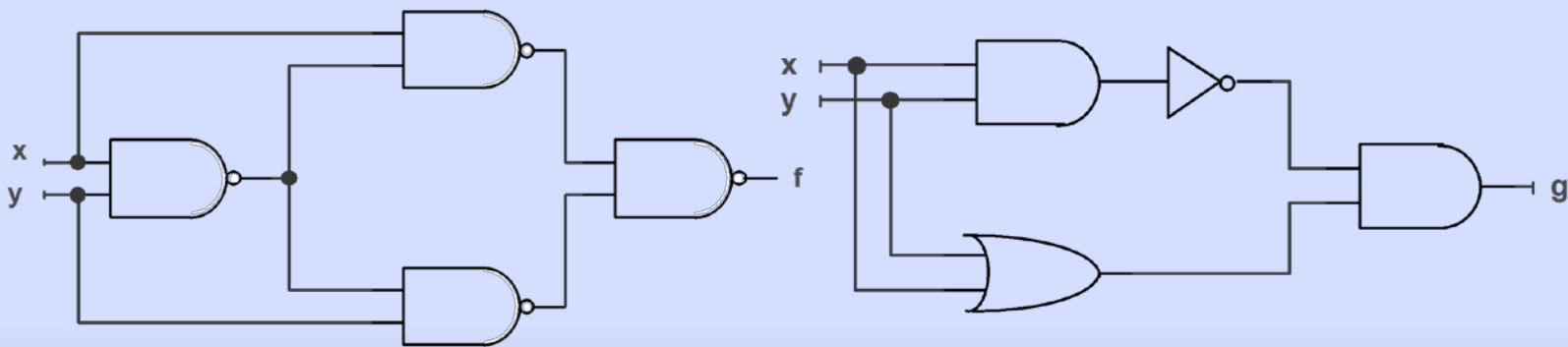


Tabla verdad			
a	b	c	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

3.2 Análisis de circuitos combinacionales

• PRÁCTICA 1

Ejercicio 1: Analice los circuitos de la figura. Para ello obtenga las tablas de verdad y las expresiones algebraicas de las funciones de conmutación f y g resultantes, minimícelas y obtenga la expresión algebraica mínima de las funciones en un circuito combinacional equivalente mínimo en forma AND/OR y OR/AND.



Tema 3. Análisis y diseño de sistemas combinacionales

CONTENIDOS:

3.1 CONCEPTO DE SISTEMA COMBINACIONAL

3.2 ANÁLISIS DE CIRCUITOS COMBINACIONALES

3.3 DISEÑO DE CIRCUITOS COMBINACIONALES

3.4 COMPONENTES COMBINACIONALES ESTÁNDAR

3.4.1 Circuitos aritméticos (sumador/restador, comparador)

3.4.2 ALU

3.4.3 Codificadores/ Decodificadores

3.4.4 Multiplexores/ Demultiplexores

3.4.5 Dispositivos lógicos programables

3.3 Diseño de sistemas combinacionales

- **Síntesis o diseño:** diseñar un circuito a partir de la descripción verbal de su comportamiento:
 1. Realización de la tabla verdad
 2. Obtención de la expresión analítica (teorema de Shannon)
 3. Minimización y expresar la función en términos del tipo de puertas a utilizar en la implementación
 4. Implementación del circuito de menor coste hardware y/o menor retardo a partir de la expresión booleana

3.3 Diseño de sistemas combinacionales

- La implementación del circuito lógico, a partir de una expresión booleana, dependerá de las puertas lógicas disponibles:
 - Con cualquier tipo de puerta lógica
 - A partir de una expresión de suma de productos:
 - puertas AND/OR
 - NAND/NAND
 - A partir de una expresión de producto de sumas:
 - Puertas OR/AND
 - NOR/NOR

3.3 Diseño de sistemas combinacionales

Ejemplo:

- **Descripción:** Diseñar un circuito lógico con tres entradas (a, b, c) y una salida (y) de forma que dicha salida es 1 si y sólo si las señales a y b son 1 ó la señal c es 0.
- Ejemplo:

1. Tabla verdad

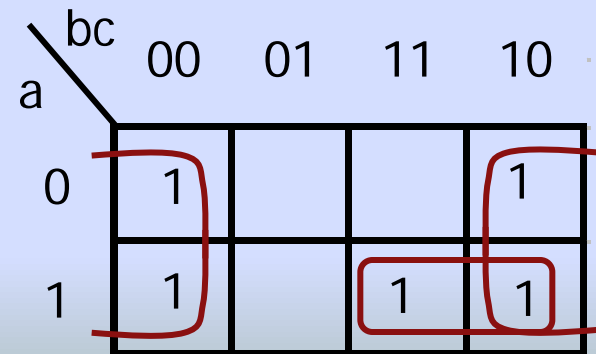
a	b	c	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

2. Expresión analítica (Teorema de Shannon)

$$y(a,b,c) = \sum m(0, 2, 4, 6, 7) = \bar{a} \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b} \cdot \bar{c} + a \cdot b \cdot \bar{c} + a \cdot b \cdot c$$

3. Minimización:

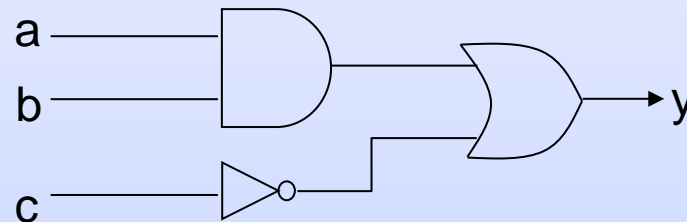
$$y(a,b,c) = \bar{c} + a \cdot b$$



3.3 Diseño de sistemas combinacionales

4. Implementación del circuito: $y(a,b,c) = \bar{c} + a \cdot b$

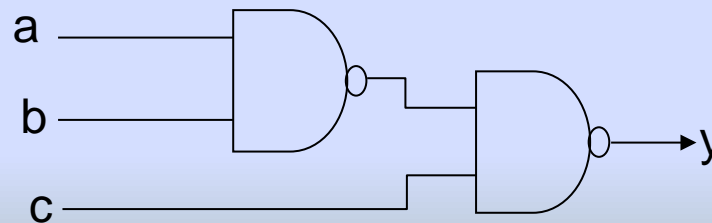
– AND/OR/INVERSORES



– NAND

Aplicando la ley de De Morgan:

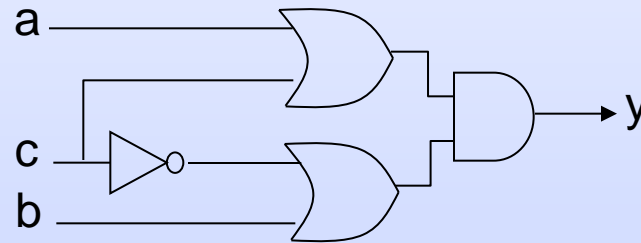
$$y(a,b,c) = \bar{c} + a \cdot b = \overline{\overline{\bar{c} + a \cdot b}} = \overline{\overline{\bar{c}} \cdot \overline{a \cdot b}} = \overline{\overline{\bar{c}} \cdot \overline{a} \cdot \overline{b}} = c \cdot a \cdot b$$



3.3 Diseño de sistemas combinacionales

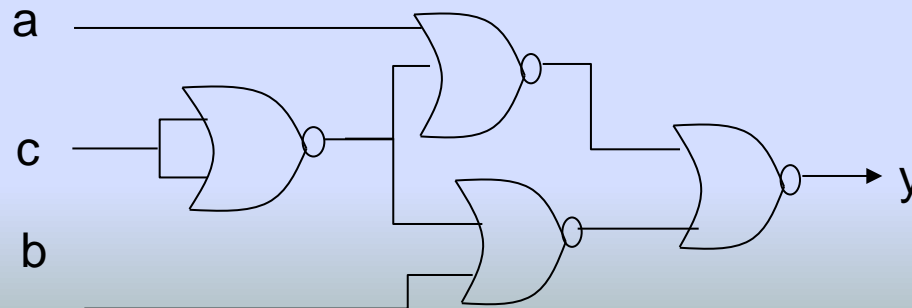
- OR/AND/INVERSORES:

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



- NOR/NOR: aplicando la ley de De Morgan:

$$y(a,b,c) = (a + \bar{c}) \cdot (b + \bar{c}) = \overline{\overline{(a + \bar{c}) \cdot (b + \bar{c})}} = \overline{\overline{(a + \bar{c})} + \overline{(b + \bar{c})}}$$



3.3 Diseño de sistemas combinacionales

- **PRÁCTICA 1**

Ejercicio 2. Diseñe un circuito lógico combinacional partiendo del siguiente enunciado:

"Un jurado consta de cuatro miembros que deben evaluar el examen de un candidato. El candidato aprobará el examen si y sólo si recibe dos o más votos favorables del jurado. Para votar los miembros del jurado disponen cada uno de ellos de un interruptor (A, B, C y D) de manera tal que pulsándolo (interruptor = 1) dan su voto favorable al candidato y no pulsándolo (interruptor = 0) dan su voto negativo al candidato."

3.3 Diseño de sistemas combinacionales

Implemente un circuito lógico mínimo que genere la función que permita determinar si aprueba o suspende un candidato tomando como entradas los cuatro pulsadores A, B, C y D de que dispone el tribunal. Realice la tabla de verdad de la función, minimice la misma e implemente dicha función empleando:

- a) Síntesis de Suma de Productos (AND-OR).
- b) Síntesis de Producto de Sumas (OR-AND).
- c) Síntesis con dos niveles de puertas NAND (NAND-NAND).
- d) Síntesis con dos niveles de puertas NOR (NOR-NOR).

Tema 3. Análisis y diseño de sistemas combinacionales

CONTENIDOS:

3.1 CONCEPTO DE SISTEMA COMBINACIONAL

3.2 ANÁLISIS DE CIRCUITOS COMBINACIONALES

3.3 DISEÑO DE CIRCUITOS COMBINACIONALES

3.4 COMPONENTES COMBINACIONALES ESTÁNDAR

3.4.1 Circuitos aritméticos (sumador/restador, comparador)

3.4.2 ALU

3.4.3 Codificadores/ Decodificadores

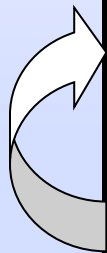
3.4.4 Multiplexores/ Demultiplexores

3.4.5 Dispositivos lógicos programables

3.4 Componentes combinacionales estándar

- Cuando la complejidad de un circuito aumenta, la descripción con tablas de verdad, como suma de términos producto (o producto de términos suma) utilizado en el **nivel de puertas lógicas**, resulta inmanejable.
- Es necesario describir el comportamiento mediante subsistemas, que son funciones más complejas:
 - Procesamiento de datos
 - Enrutamiento de datos
 - Almacenamiento de datos
- Los bloques funcionales que realizan estas funciones se consideran primitivas en el **nivel de registro**.

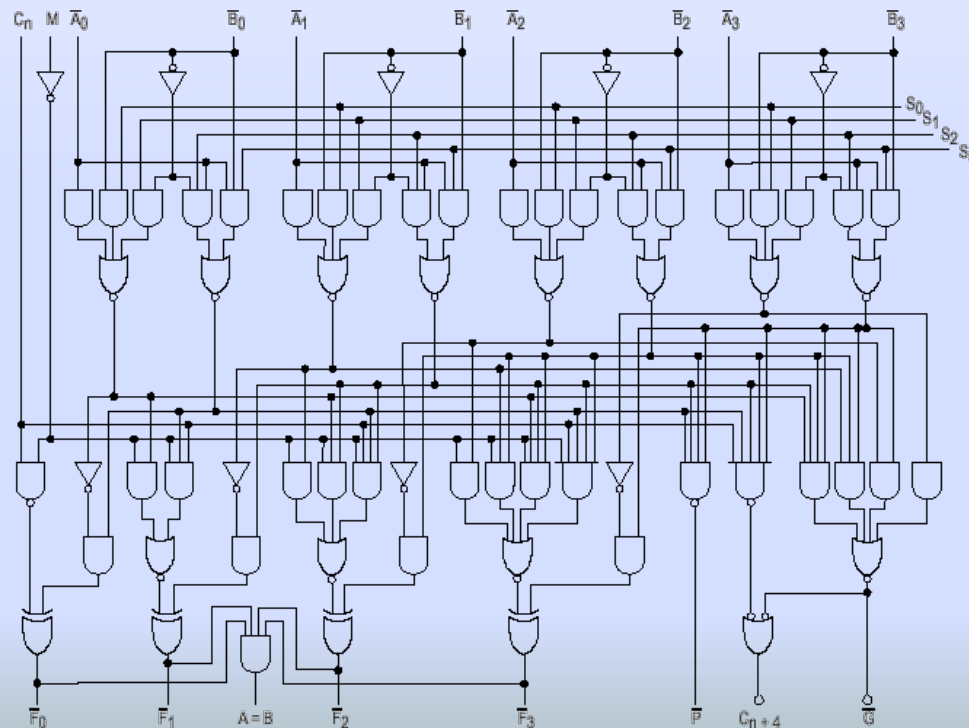
3.4 Componentes combinacionales estándar



NIVEL	COMPORTAMIENTO	COMPONENTES ESTRUCTURALES
Procesador	Instrucciones máquina	Procesadores, controladores, memorias, ASIC
Registro	Algoritmos Diagramas de flujo Cartas ASM	ALUs, MUXs, DEMUXs, registros, contadores, memorias
Puertas lógicas	Ecuaciones booleanas Diagramas Tablas de estado	Puertas lógicas y biestables
Electrónico	Ecuaciones diferenciales Diagramas corriente-tensión	Transistores, resistencias, condensadores
Físico	Layout y modelos	Difusiones P,N, pistas de metal, polisilicio

3.4 Componentes combinacionales estándar

- **Por ejemplo:** la ALU 74X181 tiene una Tabla de verdad con 16.384 filas y 8 ecuaciones booleanas, algunas dependientes de 14 variables.



3.4 Componentes combinacionales estándar

- Existen bloques funcionales que realizan estas tareas y que son primitivas en el **nivel de registro**:
 - **Componentes combinacionales:** Circuitos aritméticos, ALU, Codificadores, Decodificadores, Multiplexores, Demultiplexores, PLDs combinacionales etc.
 - **Componentes secuenciales:** registros, contadores, memorias, bancos de registros.

Tema 3. Análisis y diseño de sistemas combinacionales

CONTENIDOS:

3.1 CONCEPTO DE SISTEMA COMBINACIONAL

3.2 ANÁLISIS DE CIRCUITOS COMBINACIONALES

3.3 DISEÑO DE CIRCUITOS COMBINACIONALES

3.4 COMPONENTES COMBINACIONALES ESTÁNDAR

3.4.1 Circuitos aritméticos (sumador/restador, comparador)

3.4.2 ALU

3.4.3 Codificadores/ Decodificadores

3.4.4 Multiplexores/ Demultiplexores

3.4.5 Dispositivos lógicos programables

3.4.1 Circuitos aritméticos

- SEMI SUMADOR:**

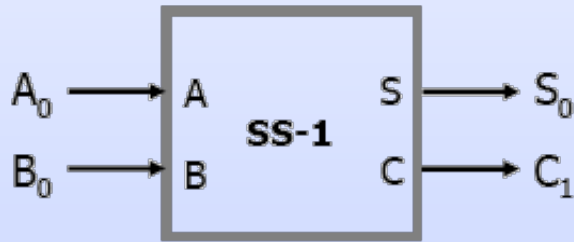
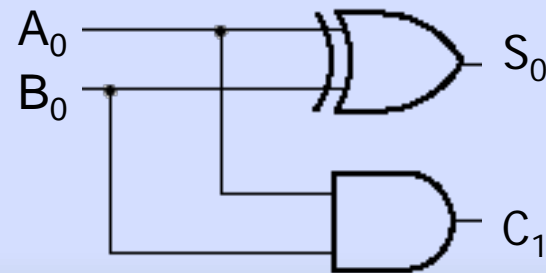


Tabla verdad

A_0	B_0	S_0	C_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S_0 = \sum m(1, 2) = \bar{A}_0 \cdot B_0 + A_0 \cdot \bar{B}_0 = A_0 \oplus B_0$$

$$C_1 = \sum m(3) = A_0 \cdot B_0$$



3.4.1 Circuitos aritméticos

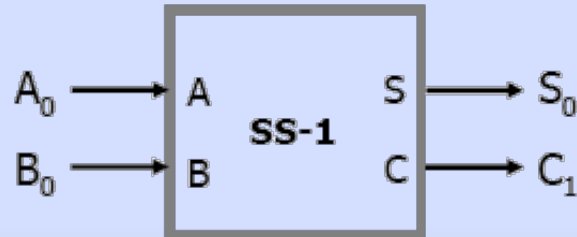
- **PRÁCTICA 2:**

2.1 CIRCUITO SEMISUMADOR.

Utilizando el simulador lógico, realice y compruebe el funcionamiento de un semisumador binario cuya tabla de verdad se representa en la Tabla. Cree un símbolo para el semisumador como el que se representa en la Figura.

Tabla verdad

A_0	B_0	S_0	C_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



3.4.1 Circuitos aritméticos

- SUMADOR COMPLETO DE 1 bit:**

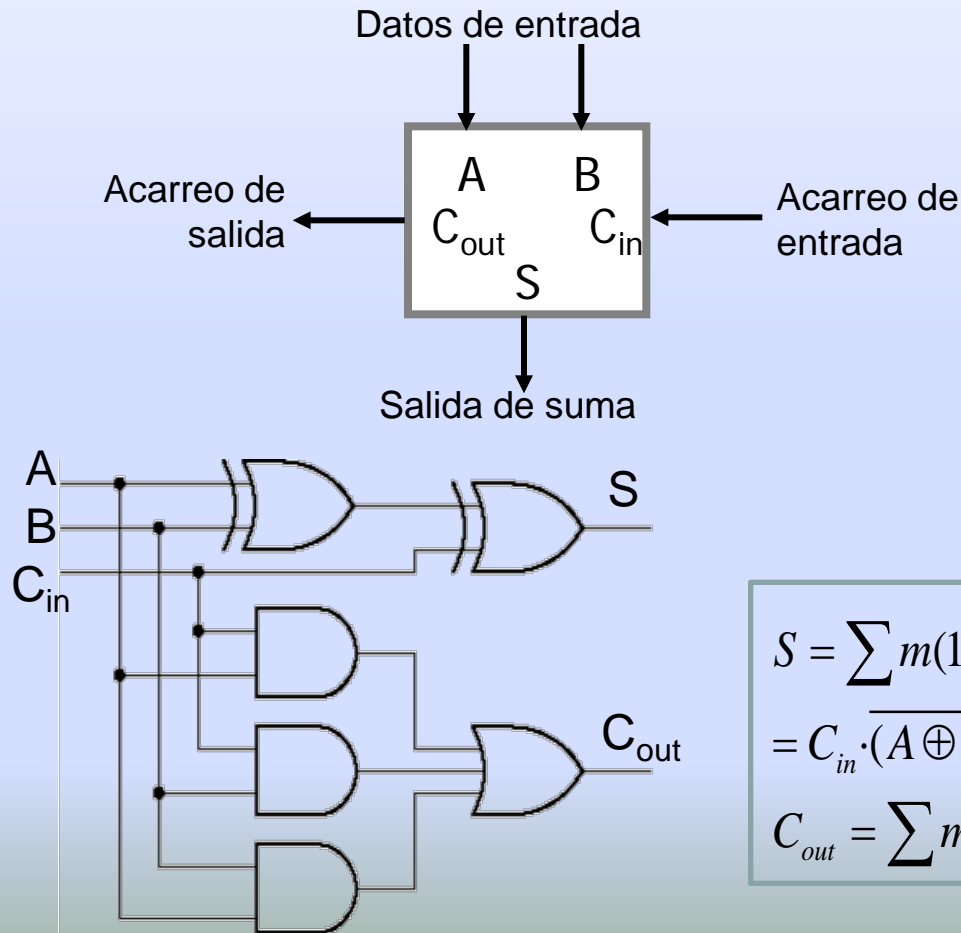


Tabla verdad

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

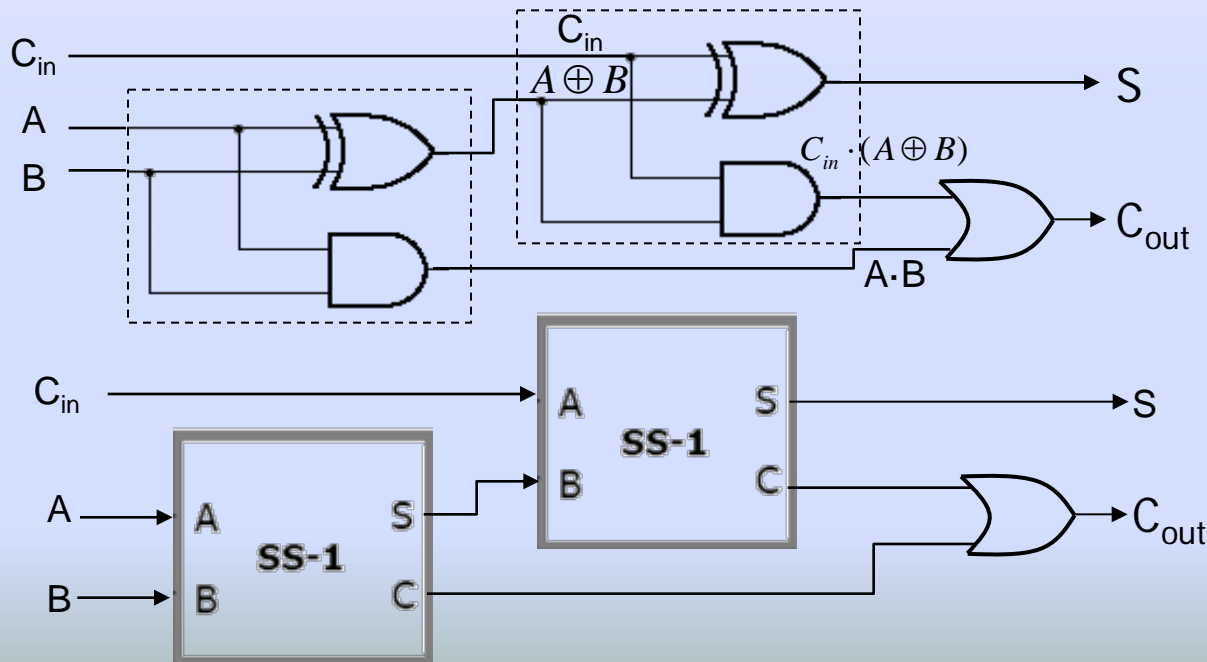
$$\begin{aligned}
 S &= \sum m(1, 2, 4, 7) = \bar{A} \cdot \bar{B} \cdot C_{in} + A \cdot B \cdot C_{in} + \bar{A} \cdot B \cdot \bar{C}_{in} + A \cdot \bar{B} \cdot \bar{C}_{in} = \\
 &= C_{in} \cdot (A \oplus B) + \bar{C}_{in} \cdot (A \oplus B) = C_{in} \oplus (A \oplus B) = C_{in} \oplus A \oplus B \\
 C_{out} &= \sum m(3, 5, 6, 7) = A \cdot B + C_{in} \cdot B + C_{in} \cdot A
 \end{aligned}$$

3.4.1 Circuitos aritméticos

- SUMADOR COMPLETO DE 1 bit a partir de dos semisumadores:**

$$S = C_{in} \oplus A \oplus B$$

$$C_{out} = A \cdot B + C_{in} \cdot \bar{A} \cdot B + C_{in} \cdot A \cdot \bar{B} = A \cdot B + C_{in} \cdot (A \oplus B)$$



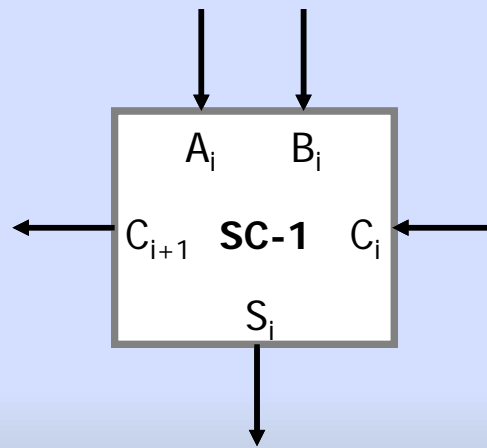
AB Cin	00	01	11	10
0			1	
1		1	1	1

3.4.1 Circuitos aritméticos

- **PRÁCTICA 2:**

2.2 CIRCUITO SUMADOR COMPLETO DE 1 BIT.

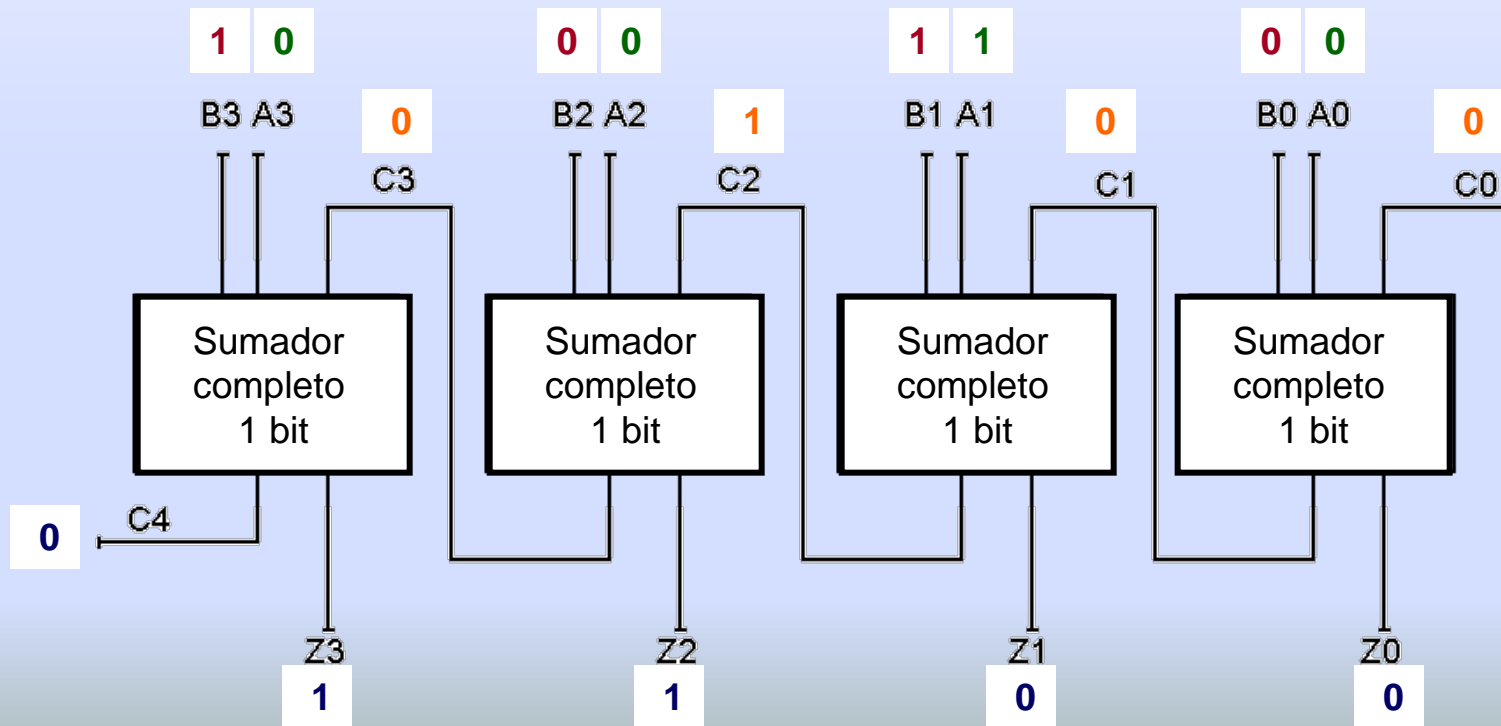
A partir de dos semisumadores y las puertas lógicas que considere oportunas, construya un sumador completo de 1 bit.



3.4.1 Circuitos aritméticos

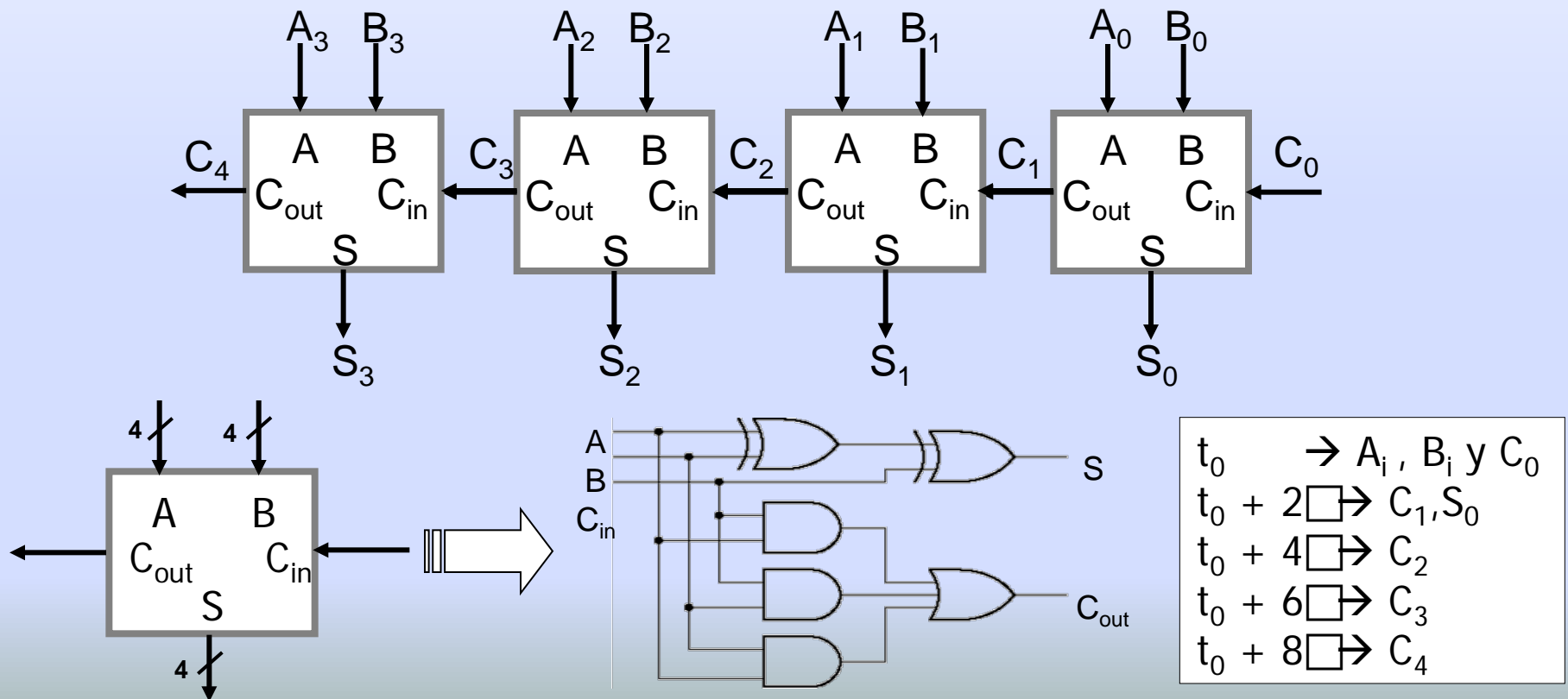
- ¿Cómo se suma?

$$\begin{array}{r} \text{010} \quad \text{Acarreo} \\ + \text{0010} \quad \text{A} \\ \text{1010} \quad \text{B} \\ \hline \text{1100} \quad \text{Z} \end{array}$$



3.4.1 Circuitos aritméticos

- Sumador binario de 4 bits con propagación del acarreo en cascada:



3.4.1 Circuitos aritméticos

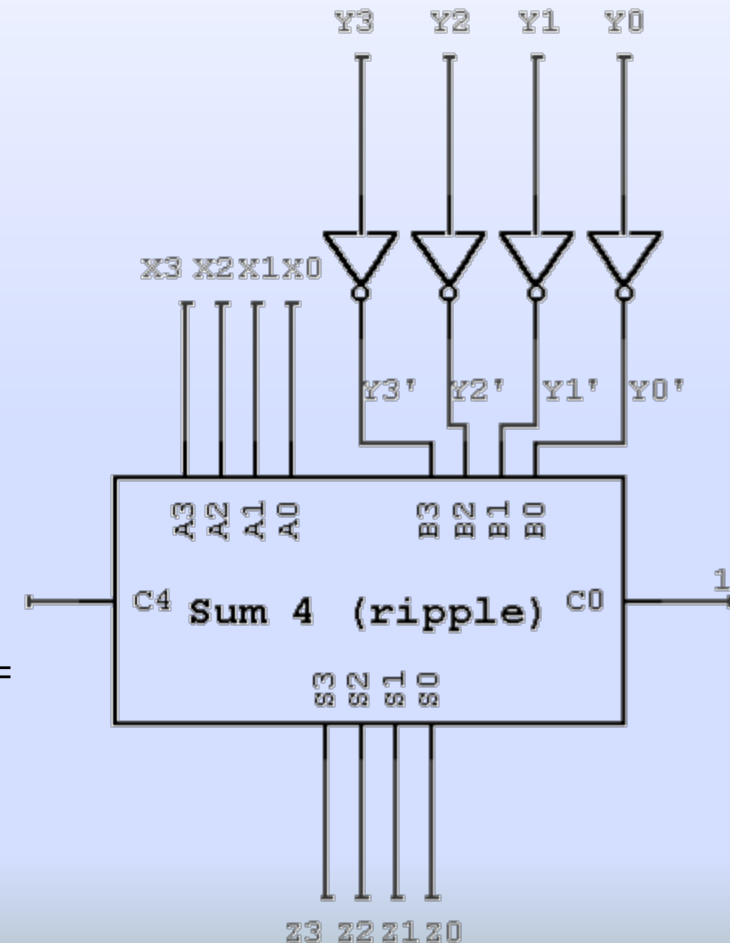
- RESTADOR:**

Resta binaria: $A - B = A + \bar{B} + 1$

$$\begin{array}{r}
 \begin{array}{r}
 1010 \quad A \\
 - 0011 \quad B \\
 \hline
 0111 \quad Z
 \end{array}
 \qquad
 \begin{array}{r}
 1010 \quad A \\
 + 1100 \quad B' \\
 \hline
 1 \quad 0110 \\
 \rightarrow + 1 \\
 \hline
 0111 \quad Z
 \end{array}
 \end{array}$$

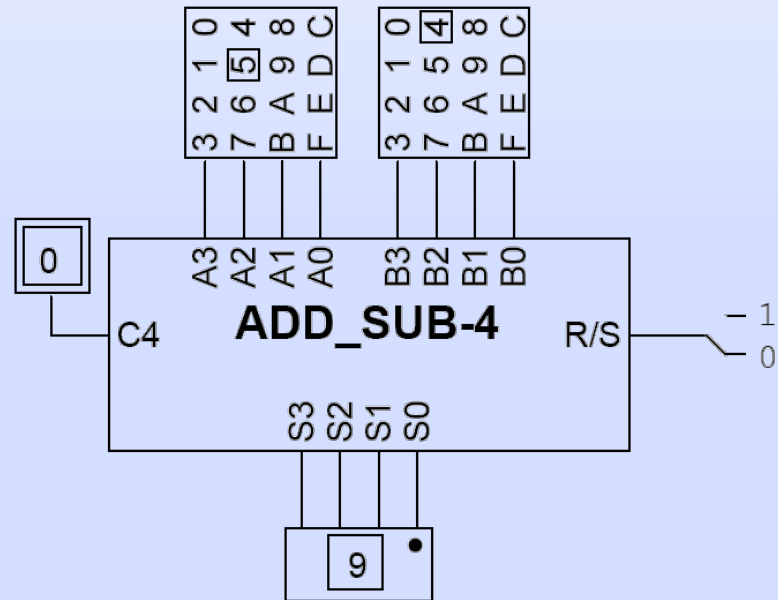
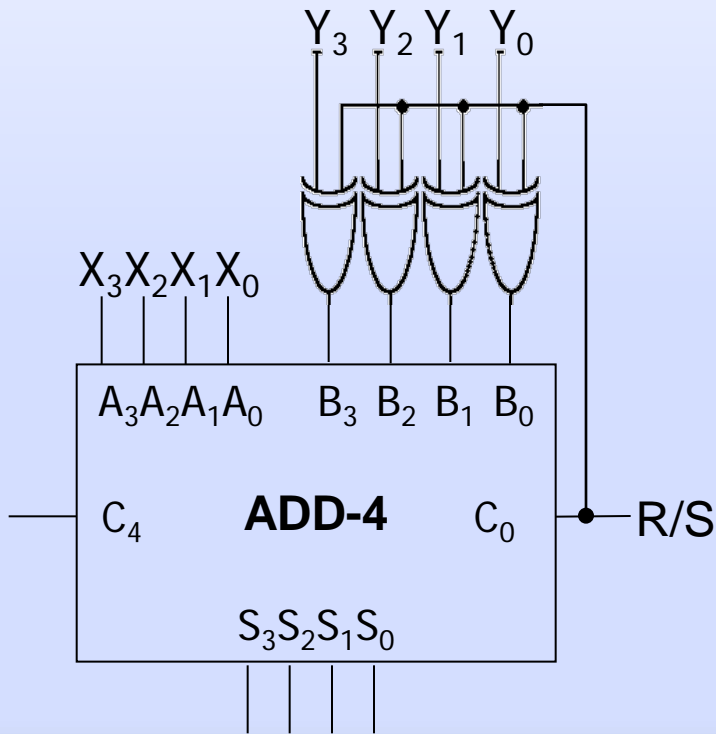
- Restador binario de 4 bits:

$$\begin{aligned}
 Z_3Z_2Z_1Z_0 &= X_3X_2X_1X_0 - Y_3Y_2Y_1Y_0 = \\
 &= X_3X_2X_1X_0 + \bar{Y}_3\bar{Y}_2\bar{Y}_1\bar{Y}_0 + 1
 \end{aligned}$$



3.4.1 Circuitos aritméticos

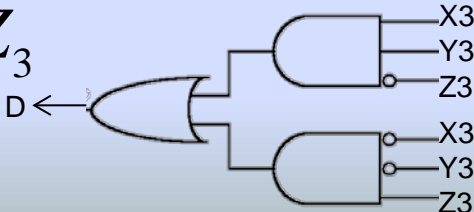
- Sumador/Restador de 4 bits (Práctica 2)



3.4.1 Circuitos aritméticos

- ¿Qué ocurre si obtenemos un resultado no representable?
 - Hay que diseñar la **lógica de desbordamiento** que dependerá del tipo de representación.
 - En complemento a 2 con signo (C2) hay desbordamiento si tras sumar dos números positivos el resultado sale negativo, o si al sumar dos números negativos el resultado sale positivo. No ocurre desbordamiento cuando se suman números de diferente signo.

Ejemplo: Si sumamos dos números de 4 bits representados en C2, donde X_3 , Y_3 y Z_3 son bits de signo, $X_3X_2X_1X_0 + Y_3Y_2Y_1Y_0 = Z_3Z_2Z_1Z_0$

$$D = X_3Y_3\bar{Z}_3 + \bar{X}_3\bar{Y}_3Z_3$$


X3	Y3	Z3	D
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

3.4.1 Circuitos aritméticos

- **SUMADOR/RESTADOR de 4 bits:**

– Si $S/R = 0$, entonces

$$A = X; \quad B = Y; \quad CO = 0$$

luego, $Z = X + Y \rightarrow$ Suma

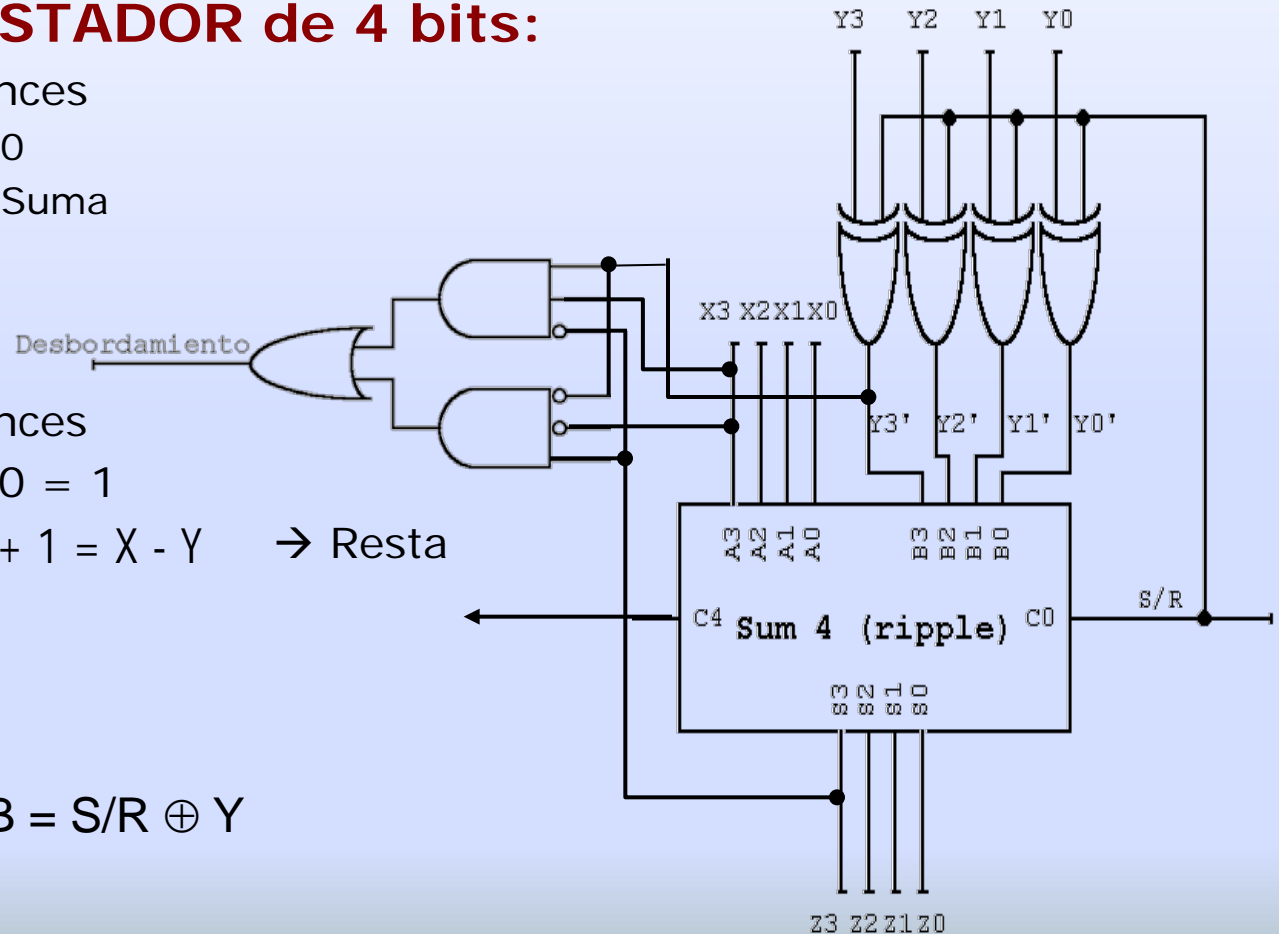
- Si $S/R = 1$, entonces

$$A = X; \quad B = \bar{Y}; \quad CO = 1$$

luego, $Z = X + \bar{Y} + 1 = X - Y \rightarrow \text{Resta}$

S/R	Y	B
0	0	0
0	1	1
1	0	1
1	1	0

$$\left. \begin{array}{l} \\ \\ \end{array} \right\} B = S/R \oplus Y$$



3.4.1 Circuitos aritméticos

- **PRÁCTICA 2:**

2.3 CIRCUITO SUMADOR COMPLETO DE 4 BITS:

Utilizando cuatro sumadores completos de 1 bit como el diseñado en el apartado 2.2, de la Práctica 2, realice un sumador para datos de 4 bits.

2.4 CIRCUITO SUMADOR/RESTADOR DE 4 BITS:

Realice un sumador/restador de 4 bits, añadiendo al sumador binario de 4 bits realizado en el apartado 2.3 las puertas lógicas que considere necesarias.

3.4.1 Circuitos aritméticos

- **COMPARADORES**

Un comparador se puede implementar de varias formas:

- Comparador con 1 salida
- Comparador con 3 salidas
- Comparador con salidas codificadas

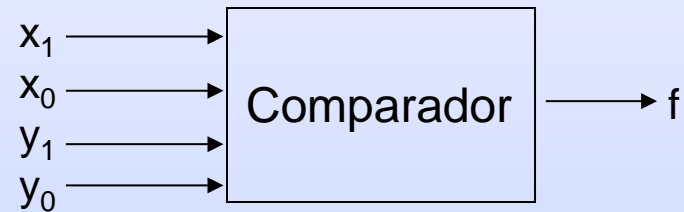
3.4.1 Circuitos aritméticos

- **Comparador con 1 salida**

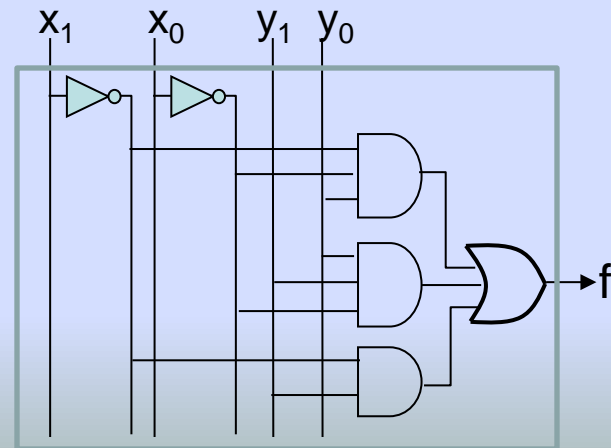
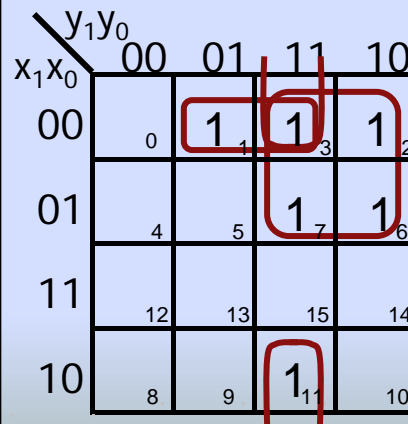
Ejemplo: comparador binario de números de 2 bits

x_1	x_0	y_1	y_0	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

$f = 1$ si $x < y$
 $f = 0$ si $x \geq y$



$$f = \bar{x}_1 \cdot \bar{x}_0 \cdot y_0 + \bar{x}_0 \cdot y_1 \cdot y_0 + \bar{x}_1 \cdot y_1$$



3.4.1 Circuitos aritméticos

- **Comparador con 3 salidas:**

Ejemplo: comparador binario de 1 bit

$G = 1$ si $x > y$

$E = 1$ si $x = y$

$L = 1$ si $x < y$

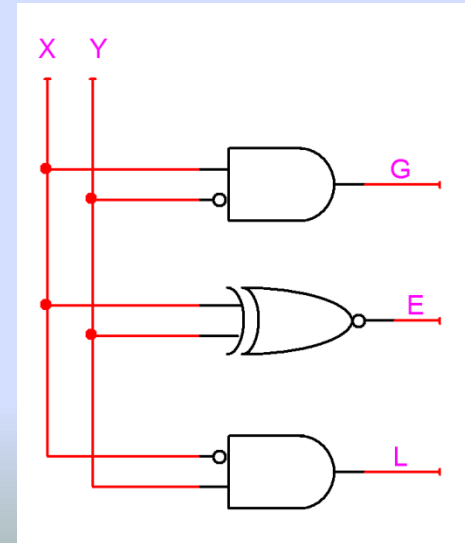


x	y	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$$G = x \cdot \bar{y}$$

$$E = \bar{x} \cdot \bar{y} + x \cdot y = \overline{x \oplus y}$$

$$L = \bar{x} \cdot y$$



3.4.1 Circuitos aritméticos

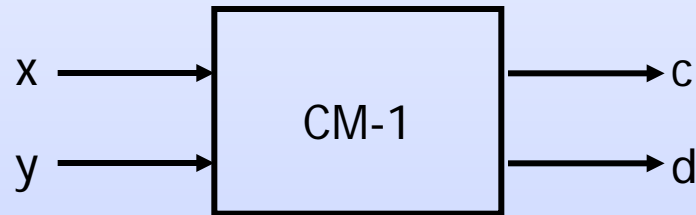
- **Comparador con salidas codificadas:**

Ejemplo: comparador de 1 bit

$cd = 00$ si $x = y$

$cd = 01$ si $x < y$

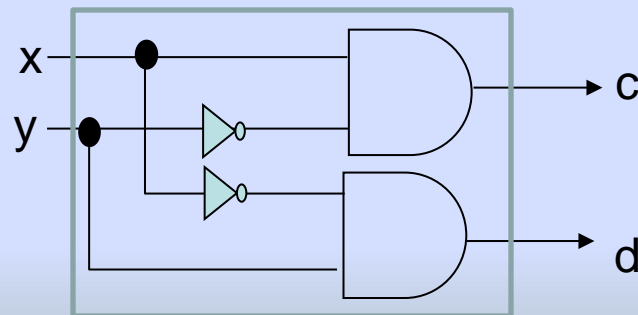
$cd = 10$ si $x > y$



x	y	c	d
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

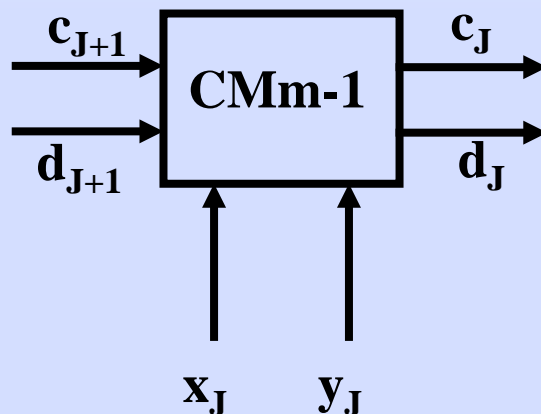
$$c = x \cdot \bar{y}$$

$$d = \bar{x} \cdot y$$



3.4.1 Circuitos aritméticos

- Para construir un comparador modular de n bits, hay que hacer un módulo CMm-1 modificado, de modo que incluya entradas adicionales para poder ampliar a cualquier número de bits.

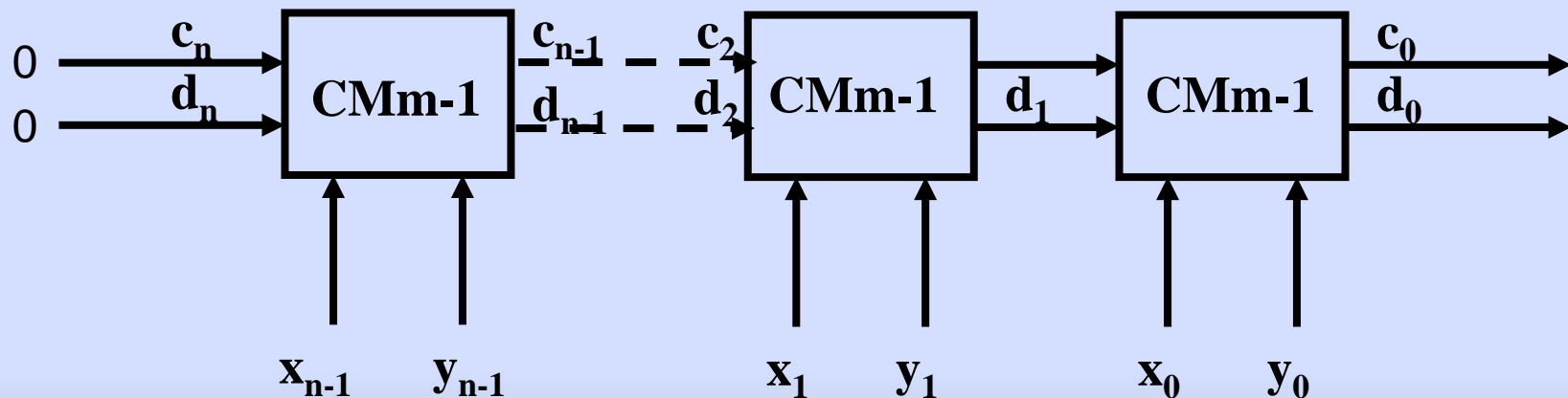


Ejercicio: Obtener la tabla de verdad de las funciones

c_J y **d_J** , minimizarlas y dibujar el circuito con estructura AND/OR.

3.4.1 Circuitos aritméticos

- Para poder comparar números de varios bits podemos construir un **comparador modular**, a partir varios comparadores de 1 bit modificados adecuadamente.
- Ejemplo: comparador modular de n bits.



Tema 3. Análisis y diseño de sistemas combinacionales

CONTENIDOS:

3.1 CONCEPTO DE SISTEMA COMBINACIONAL

3.2 ANÁLISIS DE CIRCUITOS COMBINACIONALES

3.3 DISEÑO DE CIRCUITOS COMBINACIONALES

3.4 COMPONENTES COMBINACIONALES ESTÁNDAR

3.4.1 Circuitos aritméticos (sumador/restador, comparador)

3.4.2 ALU

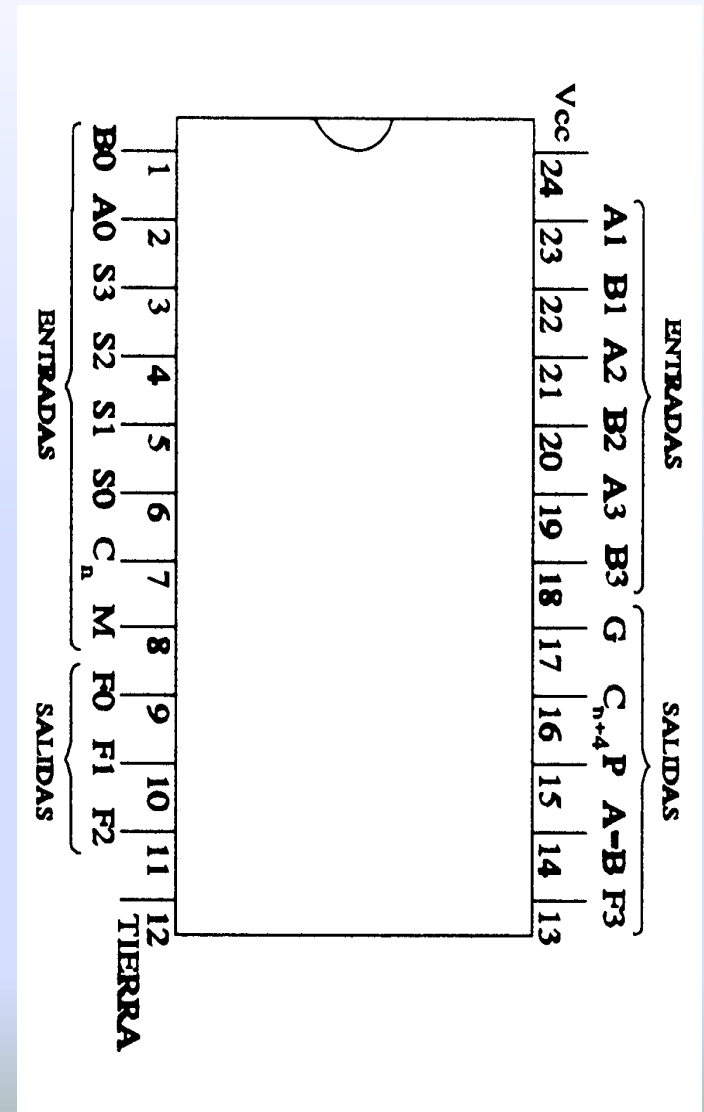
3.4.3 Codificadores/ Decodificadores

3.4.4 Multiplexores/ Demultiplexores

3.4.5 Dispositivos lógicos programables

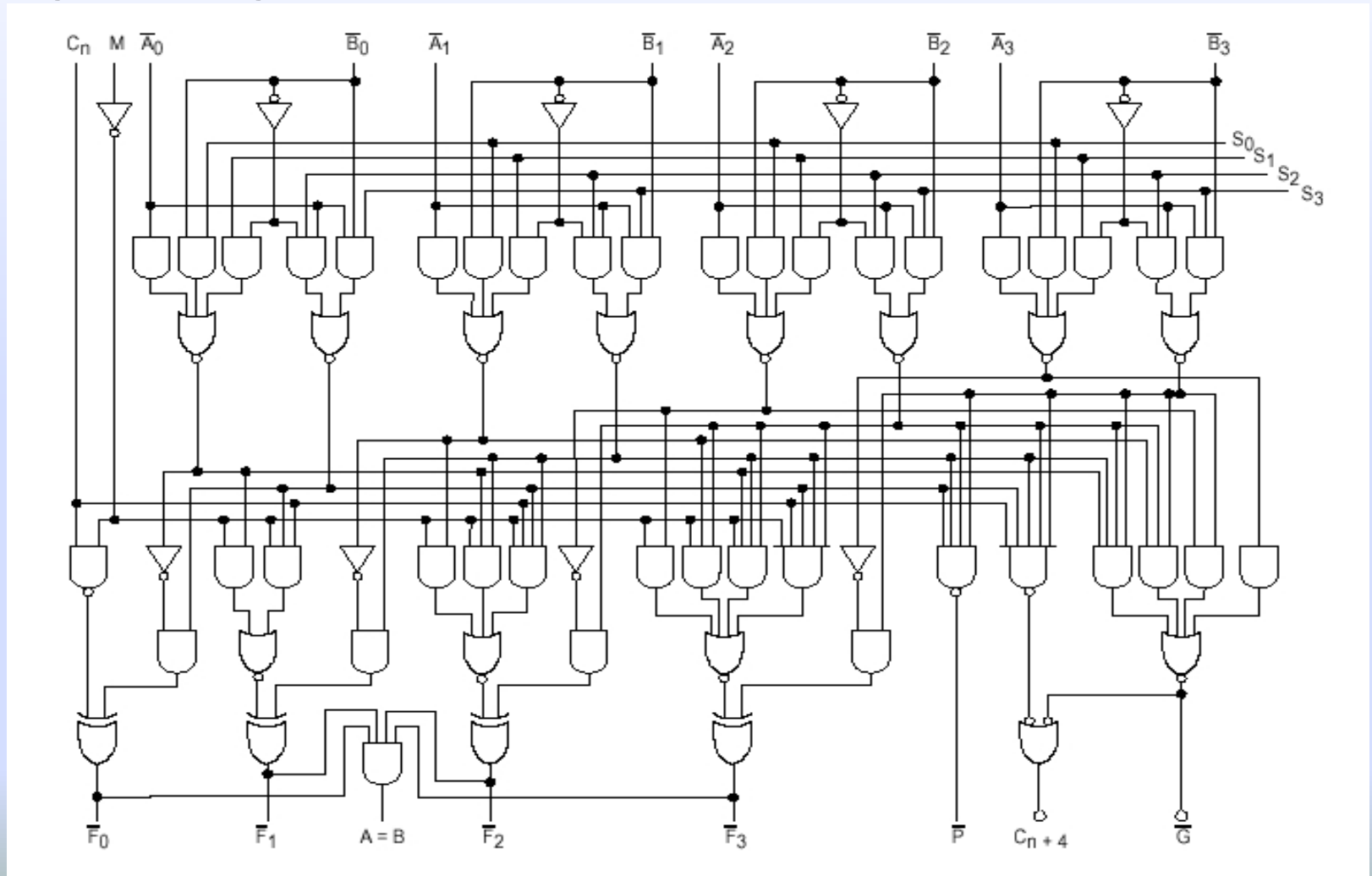
3.4.2 ALU

- Unidad Aritmético Lógica de 4 bits: 74181
 - Entrada de modo de control (M=1 lógicas, M=0 aritméticas)
 - 4 entradas de selección de función ($S_3 - S_0$)
 - Datos de 4 bits ($A_3A_2A_1A_0$, $B_3B_2B_1B_0$)
 - 16 operaciones lógicas
 - 16 operaciones aritméticas



3.4.2 ALU

- Diagrama lógico de la ALU-74181



3.4.2 ALU

- Unidad Aritmético Lógica de 4 bits: 74181

16 operaciones lógicas y 16 aritméticas

Selection				M = 1	M = 0, Arithmetic Functions	
S3	S2	S1	S0	Logic Function	Cn = 0	Cn = 1
0	0	0	0	$F = \overline{A}$	$F = A \text{ menos } 1$	$F = A$
0	0	0	1	$F = A \text{ nand } B$	$F = A B \text{ menos } 1$	$F = A B$
0	0	1	0	$F = \overline{A} + B$	$F = A \overline{B} \text{ menos } 1$	$F = A \overline{B}$
0	0	1	1	$F = 1$	$F = \text{menos } 1$	$F = \text{zero}$
0	1	0	0	$F = A \text{ nor } B$	$F = A \text{ más } (A + \overline{B})$	$F = A \text{ más } (A + \overline{B}) \text{ más } 1$
0	1	0	1	$F = \overline{B}$	$F = A B \text{ más } (A + \overline{B})$	$F = A B \text{ más } (A + \overline{B}) \text{ más } 1$
0	1	1	0	$F = A \text{ xnor } B$	$F = A \text{ menos } B \text{ menos } 1$	$F = A \text{ menos } B$
0	1	1	1	$F = A + \overline{B}$	$F = A + \overline{B}$	$F = (A + \overline{B}) \text{ más } 1$
1	0	0	0	$F = \overline{A} B$	$F = A \text{ más } (A + B)$	$F = A \text{ más } (A + B) \text{ más } 1$
1	0	0	1	$F = A \text{ xor } B$	$F = A \text{ más } B$	$F = A \text{ más } B \text{ más } 1$
1	0	1	0	$F = B$	$F = A \overline{B} \text{ más } (A + B)$	$F = A \overline{B} \text{ más } (A + B) \text{ más } 1$
1	0	1	1	$F = A + B$	$F = (A + B)$	$F = (A + B) \text{ más } 1$
1	1	0	0	$F = 0$	$F = A \text{ más } A$	$F = A \text{ más } A \text{ más } 1$
1	1	0	1	$F = A \overline{B}$	$F = A B \text{ más } A$	$F = A B \text{ más } A \text{ más } 1$
1	1	1	0	$F = A B$	$F = A \overline{B} \text{ más } A$	$F = A \overline{B} \text{ más } A \text{ más } 1$
1	1	1	1	$F = A$	$F = A$	$F = A \text{ más } 1$

3.4.2 ALU

Símbolo lógico:

Opera con los datos: $A_3A_2A_1A_0$ y $B_3B_2B_1B_0$
El resultado es el dato $F_3F_2F_1F_0$

$M = H \rightarrow$ operaciones lógicas

$M = L \rightarrow$ operaciones aritméticas

Con $s_3s_2s_1s_0$ se seleccionan las 16 operaciones lógicas o aritméticas.

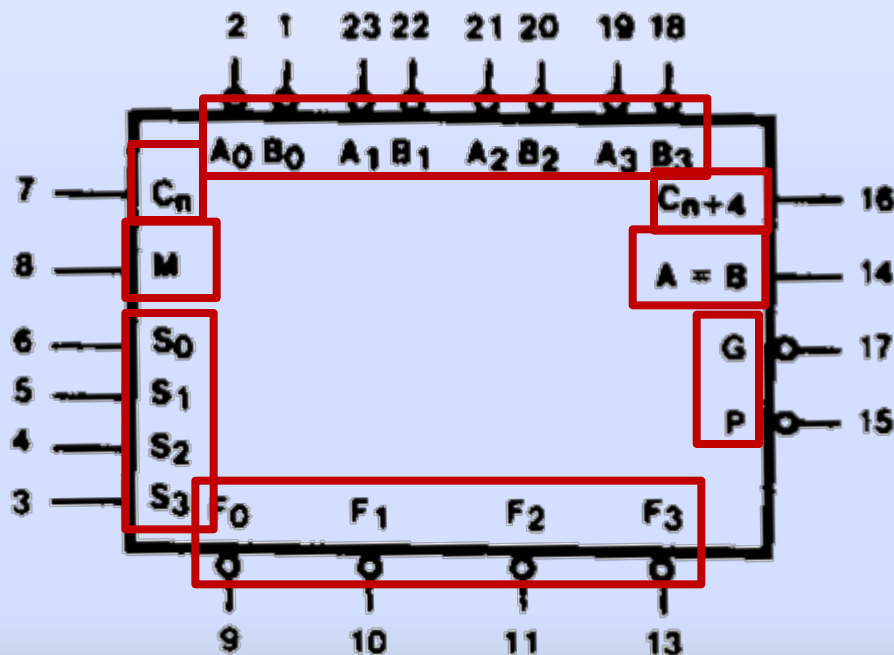
$C_n \rightarrow$ Acarreo de entrada

$C_{n+4} \rightarrow$ Acarreo de salida

$A=B \rightarrow H$ cuando $F'=1$ (resta)

$G' \rightarrow$ Acarreo generado

$P' \rightarrow$ Propagación de acarreo



3.4.2 ALU

- Todas las operaciones aritmético-lógicas se basan en la suma así que se puede diseñar una ALU modificando las entradas de un sumador.
 - **Un ampliador aritmético (AE) o etapa aritmética** es la lógica de modificación utilizada en las operaciones aritméticas.
 - **Un ampliador lógico (LE) o etapa lógica** es la lógica utilizada para las operaciones lógicas.

3.4.2 ALU

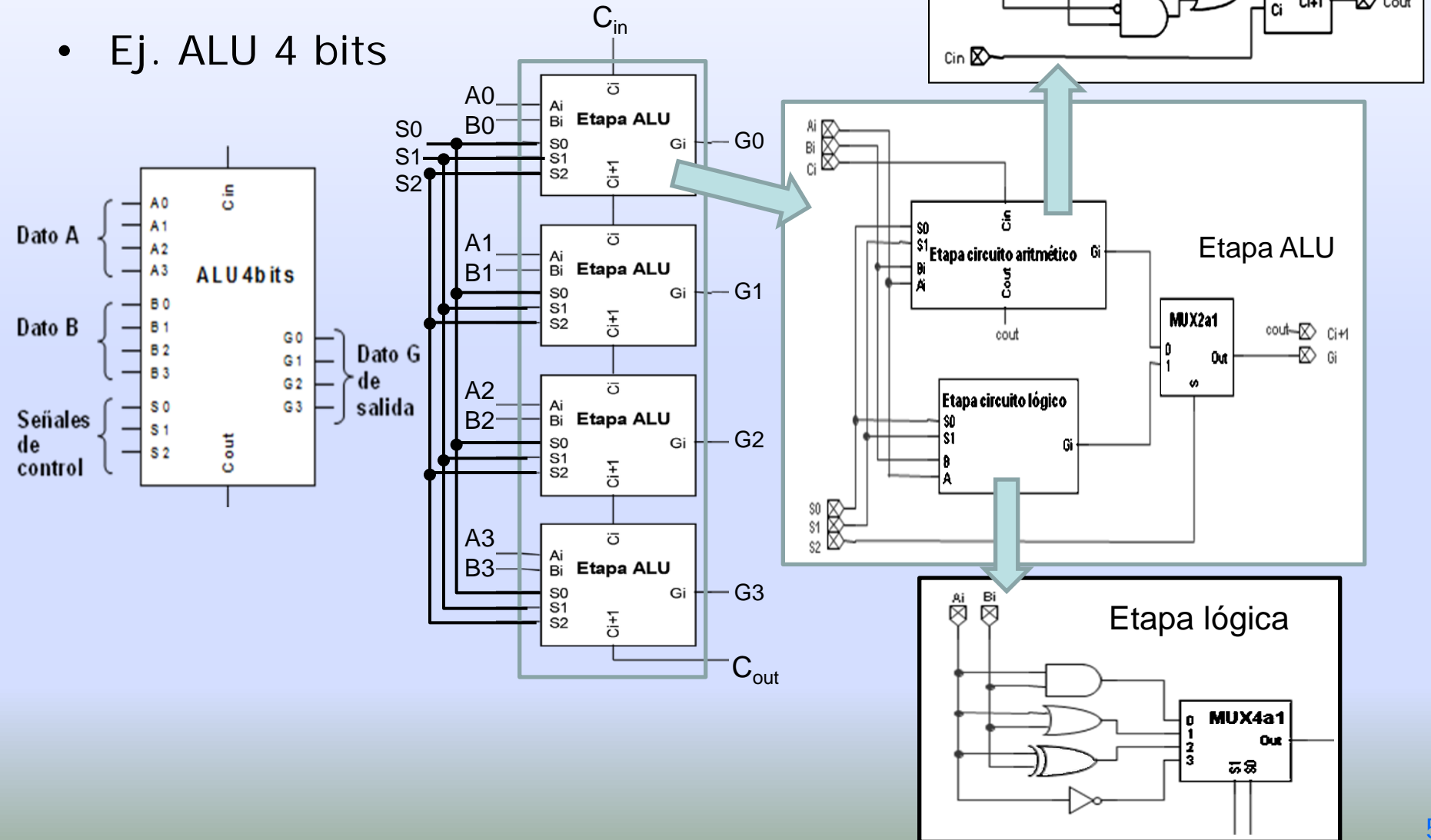
- **Práctica 3: Análisis de una unidad aritmético-lógica (ALU) de 4 bits.**

Analizar teóricamente una Unidad Aritmético-Lógica de 4 bits a partir de los esquemáticos de los circuitos que la implementa. Montar y verificar su funcionamiento con LogicWorks (ver guión de prácticas).

3.4.2 ALU

Práctica 3

- Ej. ALU 4 bits



Tema 3. Análisis y diseño de sistemas combinacionales

CONTENIDOS:

3.1 CONCEPTO DE SISTEMA COMBINACIONAL

3.2 ANÁLISIS DE CIRCUITOS COMBINACIONALES

3.3 DISEÑO DE CIRCUITOS COMBINACIONALES

3.4 COMPONENTES COMBINACIONALES ESTÁNDAR

3.4.1 Circuitos aritméticos (sumador/restador, comparador)

3.4.2 ALU

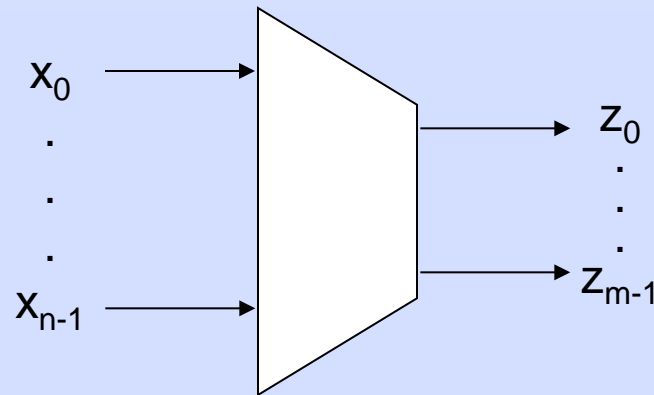
3.4.3 Codificadores/ Decodificadores

3.4.4 Multiplexores/ Demultiplexores

3.4.5 Dispositivos lógicos programables

3.4.3 Codificadores/ Decodificadores

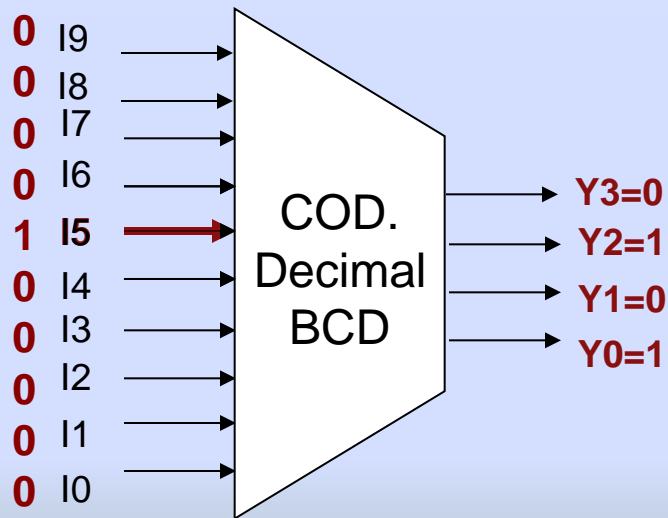
- Un **codificador** es un circuito combinacional con n entradas y m salidas de forma que en un instante sólo una de las entradas puede tomar el valor 1, generando a la salida una combinación de m bits que es única para esa entrada.



3.4.3 Codificadores/ Decodificadores

- **Ejemplo:** Codificador decimal-BCD

Presenta a la salida el código BCD del valor decimal correspondiente a la entrada activa



Entradas										Salidas			
I9	I8	I7	I6	I5	I4	I3	I2	I1	I0	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

3.4.3 Codificadores/ Decodificadores

- Un **codificador binario** proporciona a la salida el valor binario correspondiente a la entrada activa.
- Existen 2 tipos fundamentales de codificadores binarios:
 - **Codificadores sin prioridad:** solo admiten una entrada activada, codificando en la salida el valor binario de la misma y cero cuando no existe ninguna activa
 - **Codificadores con prioridad:** puede haber más de una entrada activada, existiendo prioridad en aquella cuyo valor decimal es más alto.

3.4.3 Codificadores/ Decodificadores

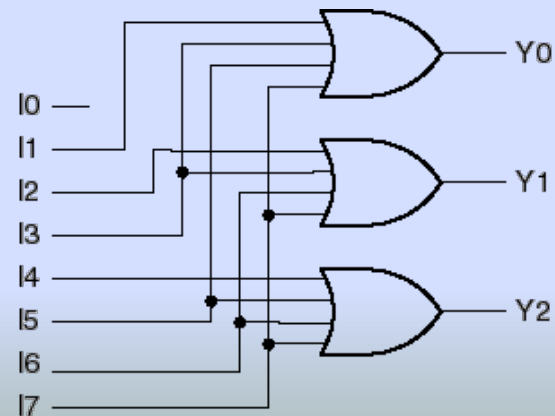
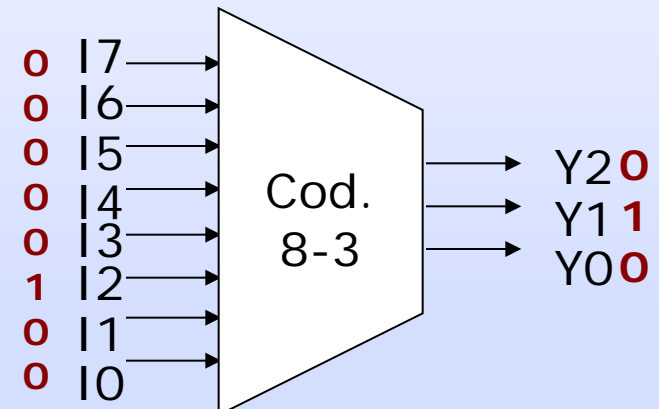
- Ejemplo: Codificador binario 8-a-3 sin prioridad:

Entradas								Salidas		
I7	I6	I5	I4	I3	I2	I1	I0	Y2	Y1	Y0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$Y2 = I4 + I5 + I6 + I7$$

$$Y1 = I2 + I3 + I6 + I7$$

$$Y0 = I1 + I3 + I5 + I7$$



3.4.3 Codificadores/ Decodificadores

- Codificador binario 4 a 2 con prioridad (orden de prioridad 3-2-1-0):

X3	X2	X1	X0	Z1	Z0
0	0	0	0	-	-
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

Z1

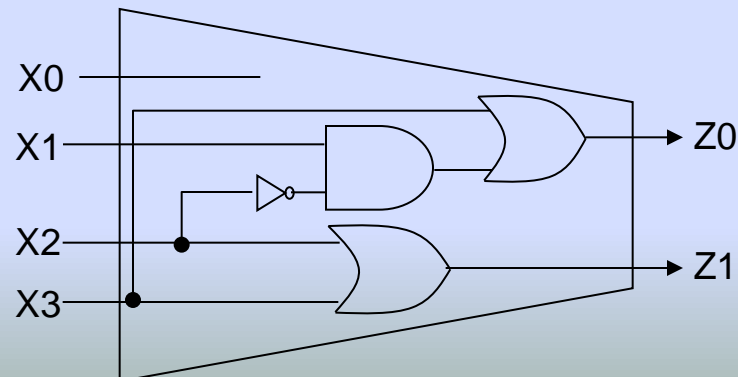
x_1x_0 x_3x_2	00	01	11	10
00	— 0	1 1	3 3	2 2
01	1 4	1 5	1 7	1 6
11	1 12	1 13	1 15	1 14
10	1 8	1 9	1 11	1 10

$$Z1 = X2 + X3$$

Z0

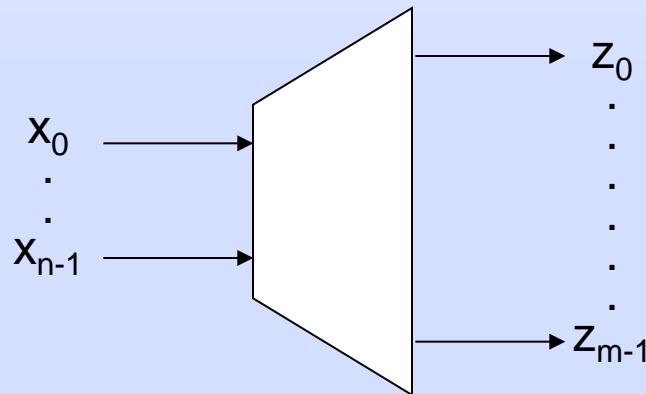
x_1x_0 x_3x_2	00	01	11	10
00	— 0	1 1	1 3	1 2
01	1 4	1 5	1 7	1 6
11	1 12	1 13	1 15	1 14
10	1 8	1 9	1 11	1 10

$$Z0 = X3 + \bar{X2} \cdot X1$$



3.4.3 Codificadores/ Decodificadores

- Un **decodificador** es un circuito combinacional con n entradas y m salidas, donde n es el número de bits que se utilicen en el código y m el número de caracteres que se están decodificando. Cada combinación de entradas pone una salida a 1 (o 0) mientras las demás permanecen a 0 (o 1).

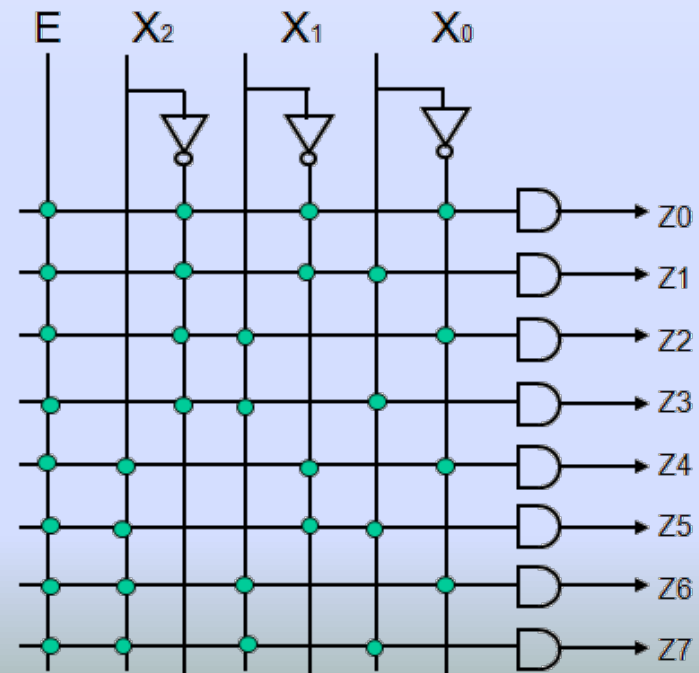
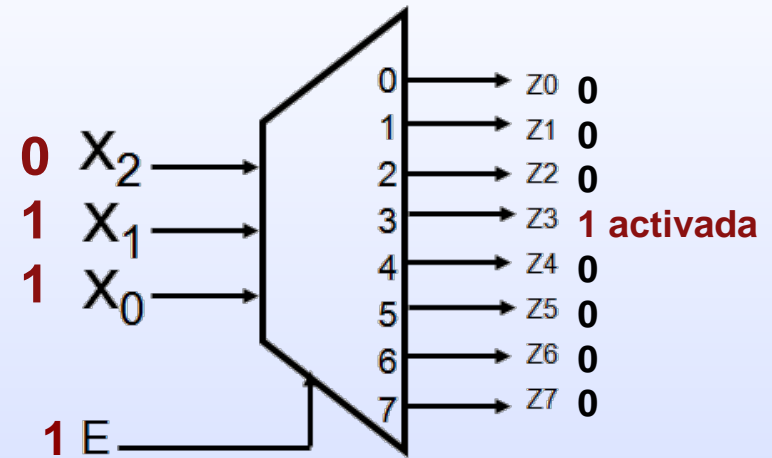


- En un **decodificador binario**, $m = 2^n$, cada combinación de entrada determina la salida cuyo número de orden coincida con el valor binario de las entradas.

3.4.3 Codificadores/ Decodificadores

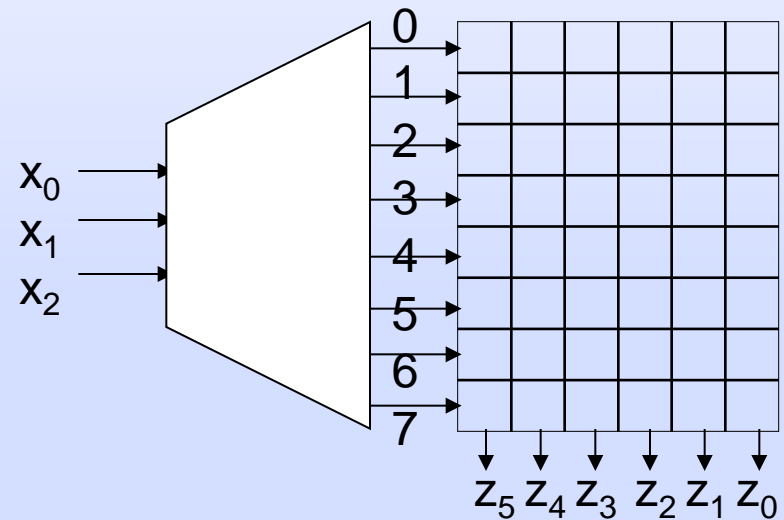
- Decodificador binario de 3 a 8:

$X_2X_1X_0$	Z_7	Z_6	Z_5	Z_4	Z_3	Z_2	Z_1	Z_0
0 0 0	0	0	0	0	0	0	0	E
0 0 1	0	0	0	0	0	0	E	0
0 1 0	0	0	0	0	0	E	0	0
0 1 1	0	0	0	0	E	0	0	0
1 0 0	0	0	0	E	0	0	0	0
1 0 1	0	0	E	0	0	0	0	0
1 1 0	0	E	0	0	0	0	0	0
1 1 1	E	0	0	0	0	0	0	0



3.4.3 Codificadores/ Decodificadores

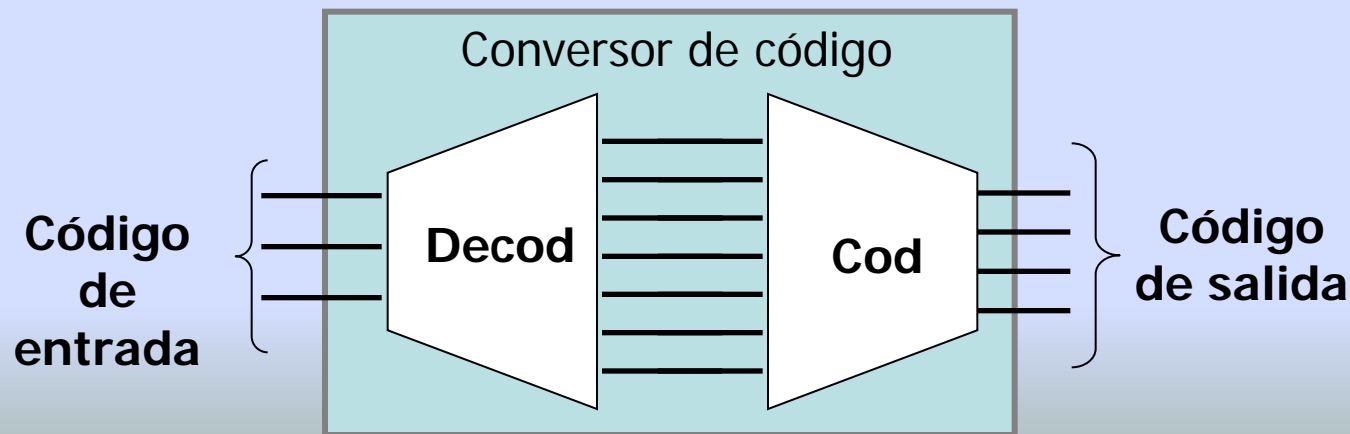
- **Aplicaciones de los decodificadores:**
 - **Memoria** del computador
 - Seleccionar diferentes bancos de memoria
 - Seleccionar palabras dentro de un chip de memoria
 - **Sistema de Entrada/Salida** en un computador: seleccionar diferentes dispositivos
 - **Decodificar los códigos de operación** que identifican cada instrucción.



3.4.3 Codificadores/ Decodificadores

- **Conversores de código:**

- permiten pasar de un código a otro.
- por ejemplo: de Binario natural a 7 segmentos, de Gray a Binario, de Gray a BCD, etc.
- Alternativas de implementación:
 - Simplificación multifuncional
 - Decodificador + Codificador



3.4.3 Codificadores/ Decodificadores

- **Práctica 4: Funcionamiento de codificadores, decodificadores y multiplexores, demultiplexores**

4.1 Realice y simule en LogicWorks los siguientes circuitos:

- Un decodificador binario de 3 entradas y 8 salidas con entrada de habilitación CE.
- Un codificador binario con prioridad de 4 entradas y 2 salidas.

4.2 Conversor de siete segmentos.

- Realice, utilizando Logic Works un conversor de código para un visualizador de 7 segmentos.

Tema 3. Análisis y diseño de sistemas combinacionales

CONTENIDOS:

3.1 CONCEPTO DE SISTEMA COMBINACIONAL

3.2 ANÁLISIS DE CIRCUITOS COMBINACIONALES

3.3 DISEÑO DE CIRCUITOS COMBINACIONALES

3.4 COMPONENTES COMBINACIONALES ESTÁNDAR

3.4.1 Circuitos aritméticos (sumador/restador, comparador)

3.4.2 ALU

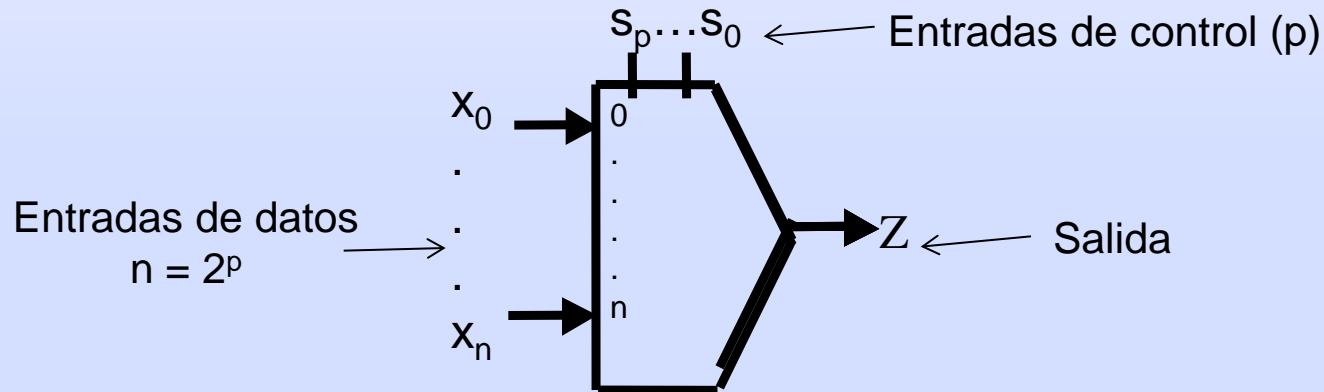
3.4.3 Codificadores/ Decodificadores.

3.4.4 Multiplexores/ Demultiplexores

3.4.5 Dispositivos lógicos programables

3.4.4 Multiplexores/ Demultiplexores

- Un **multiplexor (MUX)** es un bloque combinacional con p entradas de control, 2^p entradas de datos y una salida, de forma que conecta una de las entradas con la salida según el valor de las entradas de control.



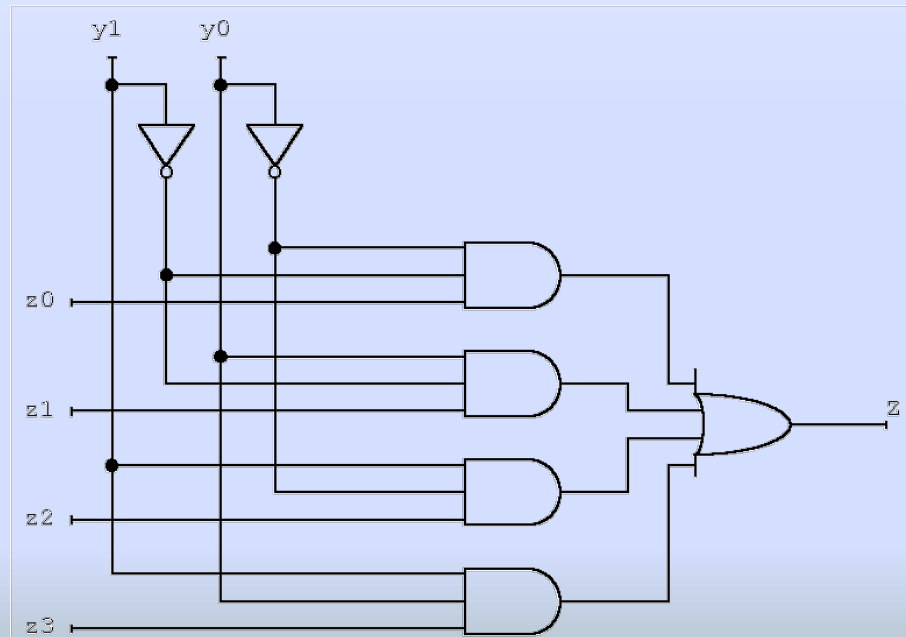
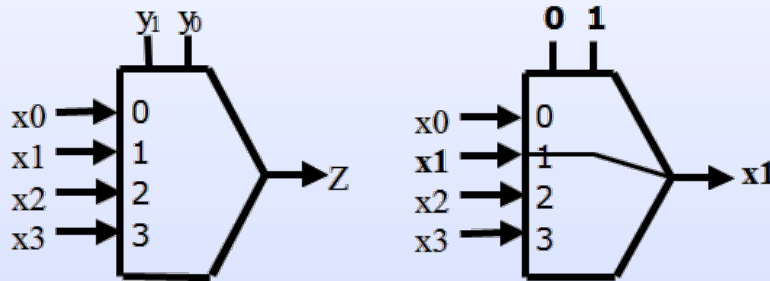
- Si las entradas de datos están numeradas, se conecta con la salida aquella cuyo número de orden coincide con el valor en binario de las entradas de control.
- El multiplexor también se denomina **selector de datos**

3.4.4 Multiplexores/ Demultiplexores

- Multiplexor 4 a 1:**

Entradas de control		Salida
y1	y0	z
0	0	x0
0	1	x1
1	0	x2
1	1	x3

$$z = \bar{y}_1 \cdot \bar{y}_0 \cdot x_0 + \bar{y}_1 \cdot y_0 \cdot x_1 + y_1 \cdot \bar{y}_0 \cdot x_2 + y_1 \cdot y_0 \cdot x_3$$



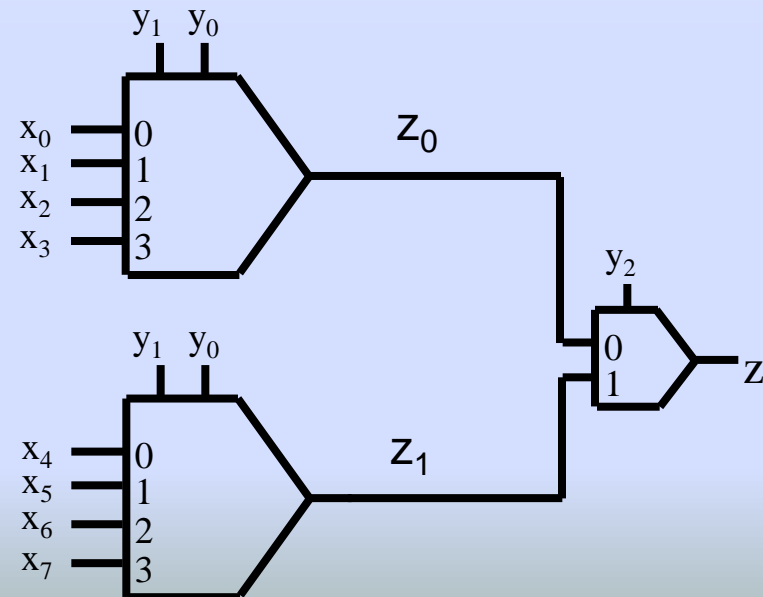
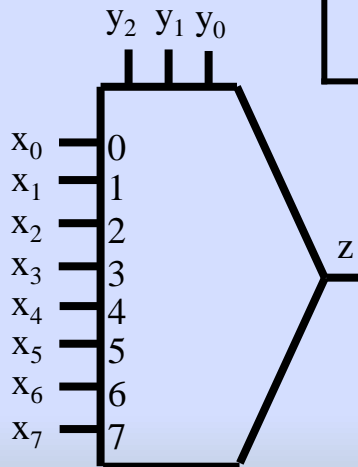
3.4.4 Multiplexores/ Demultiplexores

- Ejemplo:** construir un multiplexor 8 a 1 a partir de dos multiplexores de 4 a 1.

y2	y1	y0	z
0	0	0	x0
0	0	1	x1
0	1	0	x2
0	1	1	x3
1	0	0	x4
1	0	1	x5
1	1	0	x6
1	1	1	x7

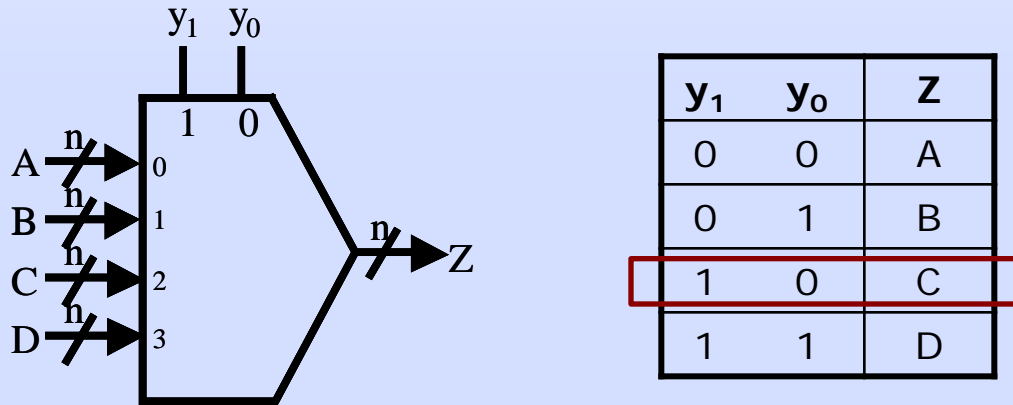
y1	y0	z1	z0
0	0	x0	x4
0	1	x1	x5
1	0	x2	x6
1	1	x3	x7

y2	z
0	z0
1	z1



3.4.4 Multiplexores/ Demultiplexores

- **Aplicaciones:**
 - **Transmitir información** con multiplexores de palabras de n bits. Por ej.: MUX de 4 (palabras de n bits) a 1 (palabra de n bits).



Por ejemplo: si $y_1y_0 = 10$, $Z = C$

siendo $Z = Z_0Z_1 \dots Z_{n-1}$ y $C = C_0C_1 \dots C_{n-1}$

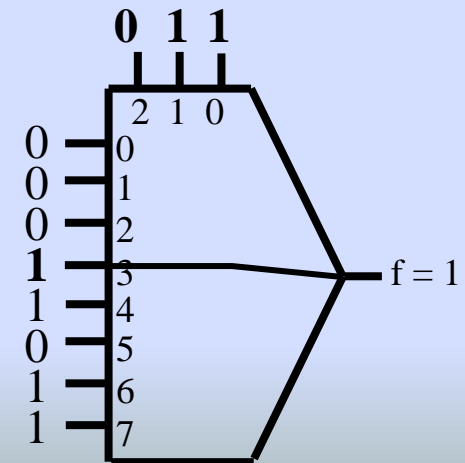
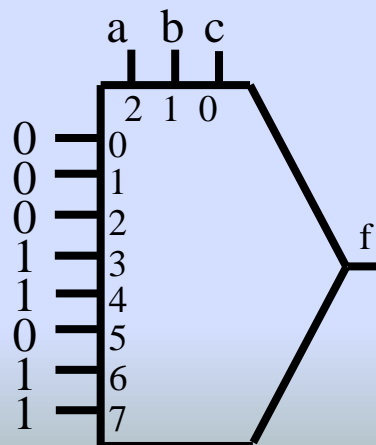
3.4.4 Multiplexores/ Demultiplexores

–Módulos lógicos universales para sintetizar funciones de conmutación:

- Las **entradas de control** son las **variables de conmutación** a sintetizar y
- las **variables de entrada** son los **valores 0 o 1** que la función deba producir.
- Con un MUX de ***n* entradas de control** se puede implementar cualquier función de conmutación de ***n* variables**.

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$f(a,b,c) = \sum m(3,4,6,7)$$



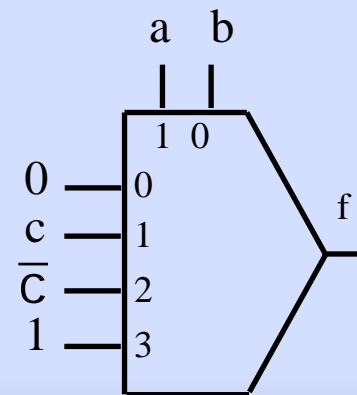
3.4.4 Multiplexores/ Demultiplexores

- Con un multiplexor de **n entradas de control** se pueden sintetizar **funciones de n+1 variables**.
- Por ejemplo, con un multiplexor de 2 entradas de control se puede sintetizar una función de 3 variables:
 $f(a,b,c) = \sum m(3,4,6,7)$

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

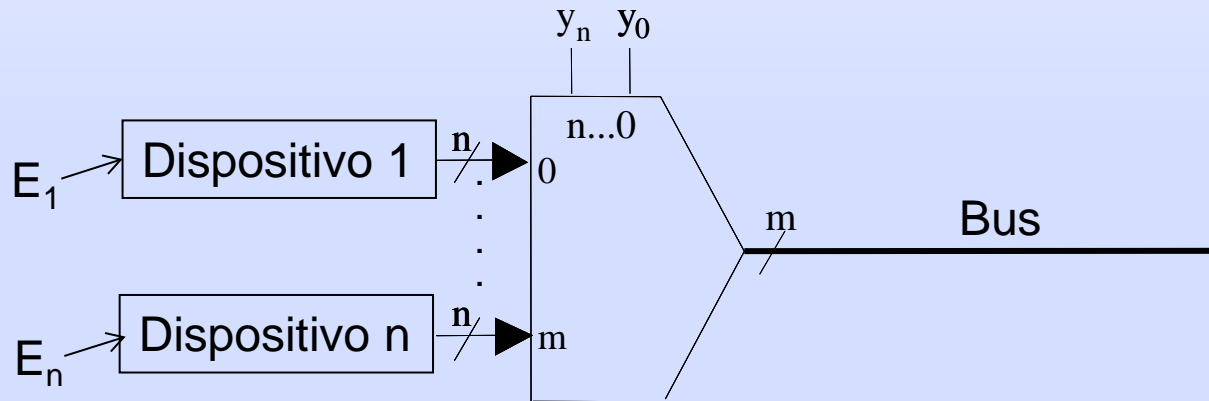
≡

a	b	f
0	0	0
0	1	c
1	0	\overline{C}
1	1	1



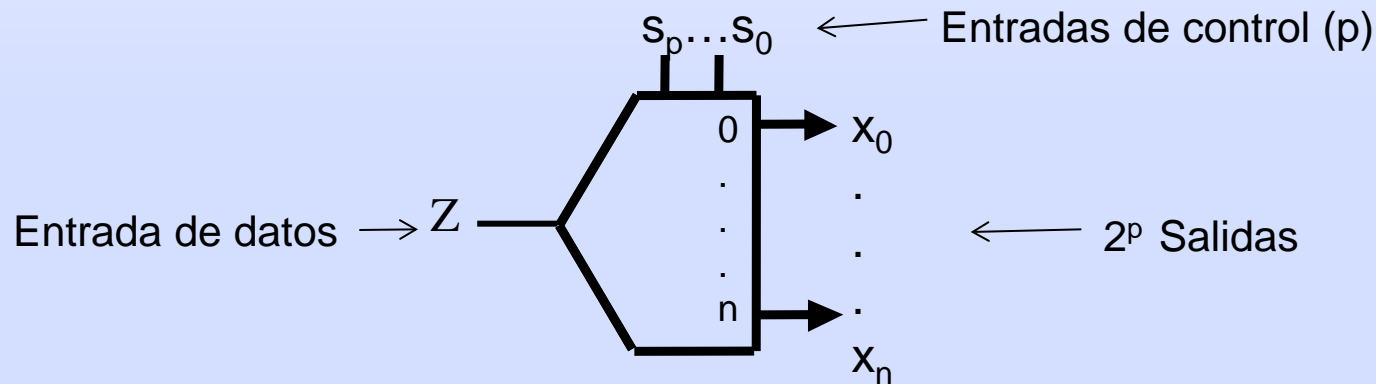
3.4.4 Multiplexores/ Demultiplexores

- **Acceder a un bus** para enviar información desde varias fuentes.



3.4.4 Multiplexores/ Demultiplexores

- Un **demultiplexor** es un bloque combinacional que con p entradas de control, una entrada de datos y 2^p salidas de datos. Conecta la entrada de datos con una de las salidas según el valor de las entradas de control.

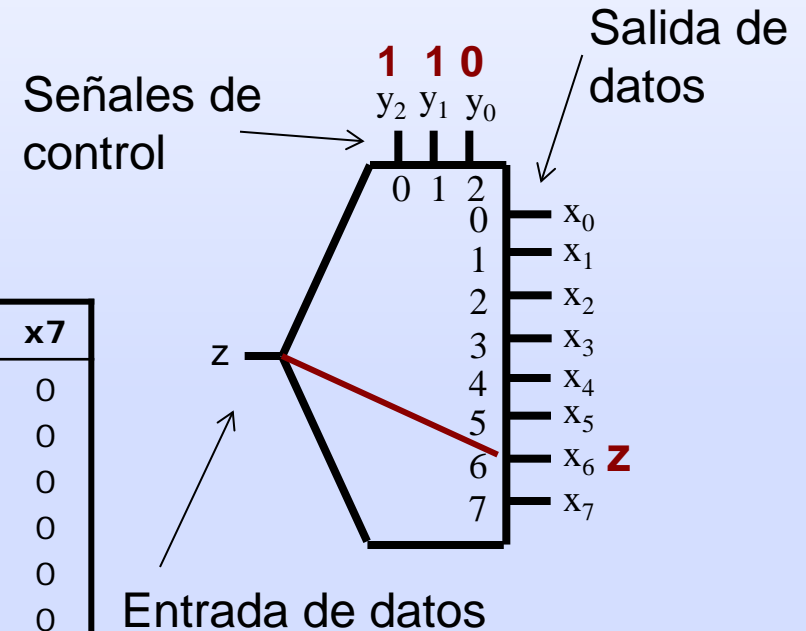


- Si las salidas están numeradas, la entrada se conecta a la salida cuyo número de orden coincide con el valor binario de las entradas de control.

3.4.4 Multiplexores/ Demultiplexores

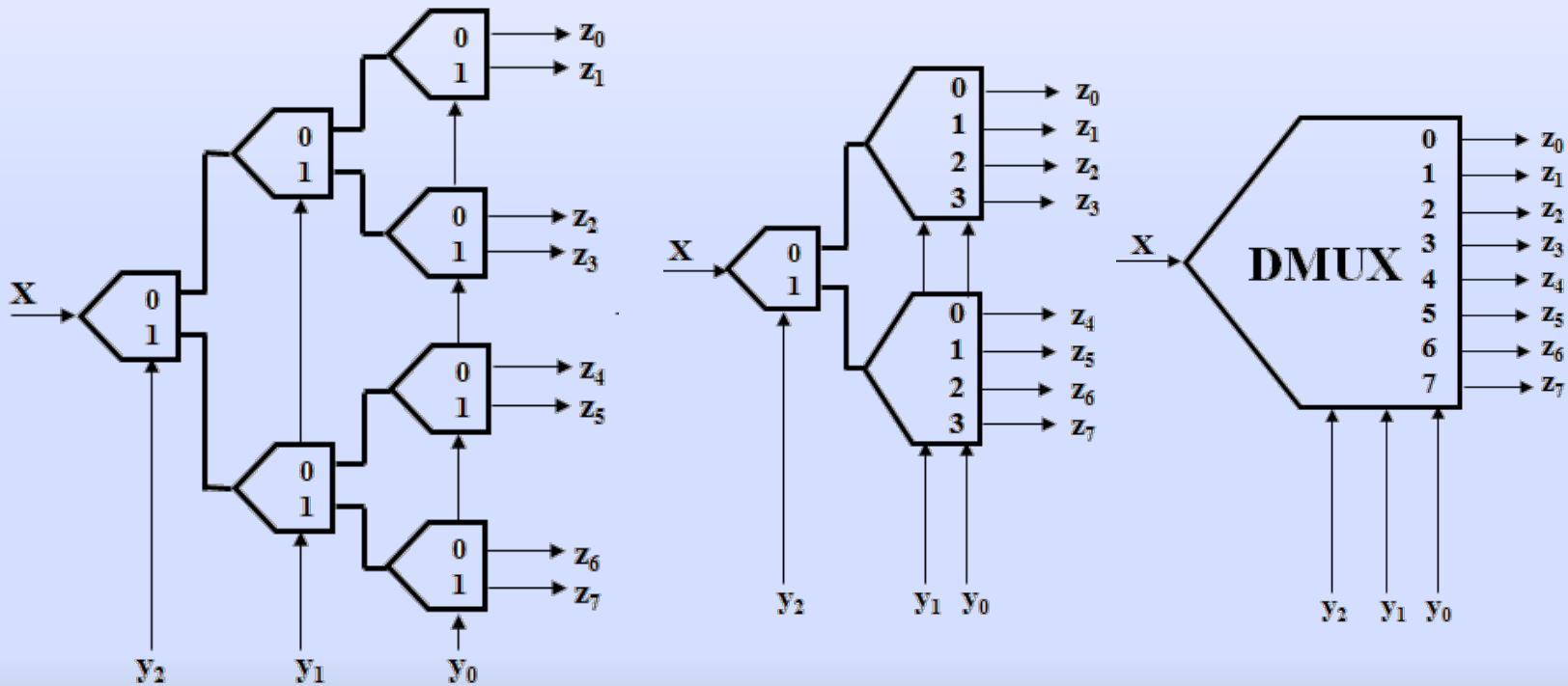
- **Ejemplo:** demultiplexor 1 a 8

y2	y1	y0	x0	x1	x2	x3	x4	x5	x6	x7
0	0	0	Z	0	0	0	0	0	0	0
0	0	1	0	Z	0	0	0	0	0	0
0	1	0	0	0	Z	0	0	0	0	0
0	1	1	0	0	0	Z	0	0	0	0
1	0	0	0	0	0	0	Z	0	0	0
1	0	1	0	0	0	0	0	Z	0	0
1	1	0	0	0	0	0	0	0	Z	0
1	1	1	0	0	0	0	0	0	0	Z



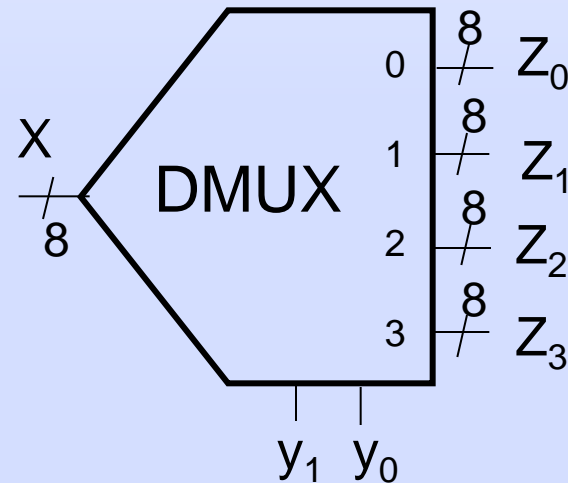
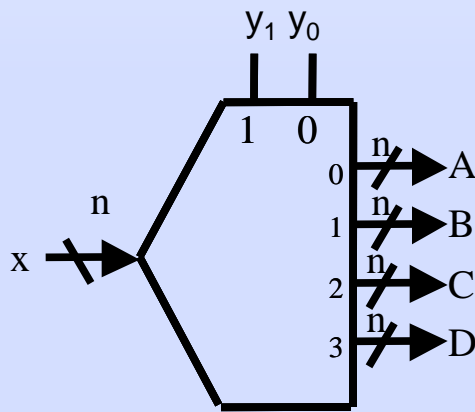
3.4.4 Multiplexores/ Demultiplexores

- Ampliación del número de líneas de control:



3.4.4 Multiplexores/ Demultiplexores

- También hay **demultiplexores de palabras de n bits**. Los demultiplexores se pueden utilizar para ampliación el número de bits de los datos:



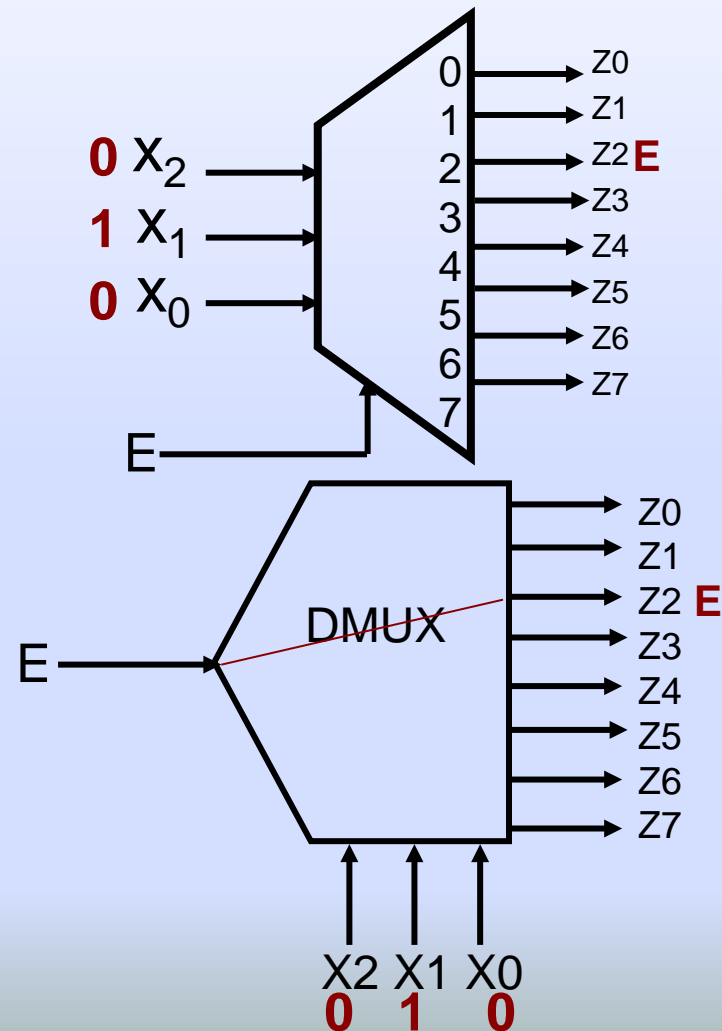
$$X = \{x_7, x_6, \dots, x_0\}$$

$$Z_i = \{z_{i7}, z_{i6}, \dots, z_{i0}\}$$

3.4.4 Multiplexores/ Demultiplexores

Decodificador/Demultiplexor:

- un **demultiplexor** de “n” a “m” equivale a un **decodificador** binario de “n” a “m” con entrada E de habilitación.
- Su denominación indica un contexto diferente de utilización:
 - un **demultiplexor** es un circuito que permite conectar una fuente de datos de entrada a múltiples destinos en función de unas señales de control.
 - Un **decodificador** binario activa la salida cuya número de orden coincide con el valor binario de las entradas.



3.4.4 Multiplexores/ Demultiplexores

- **Práctica 4: Funcionamiento de codificadores, decodificadores y multiplexores, demultiplexores**

4.3 Síntesis de funciones lógicas con multiplexores.

Implemente la función de tres variables

$f(A, B, C)$ cuya tabla de verdad se presenta

en la tabla, utilizando multiplexores de 2 a 1.

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

4.4 Realización de demultiplexores.

Realice un demultiplexor de 1 a 8. Compare este circuito con el decodificador binario de 3 entradas y 8 salidas con entrada de habilitación de chip (CE) implementado en el apartado 1.1 de esta práctica. ¿Mantienen alguna similitud dichos circuitos?

Tema 3. Análisis y diseño de sistemas combinacionales

CONTENIDOS:

3.1 CONCEPTO DE SISTEMA COMBINACIONAL

3.2 ANÁLISIS DE CIRCUITOS COMBINACIONALES

3.3 DISEÑO DE CIRCUITOS COMBINACIONALES

3.4 COMPONENTES COMBINACIONALES ESTÁNDAR

3.4.1 Circuitos aritméticos (sumador/restador, comparador)

3.4.2 ALU

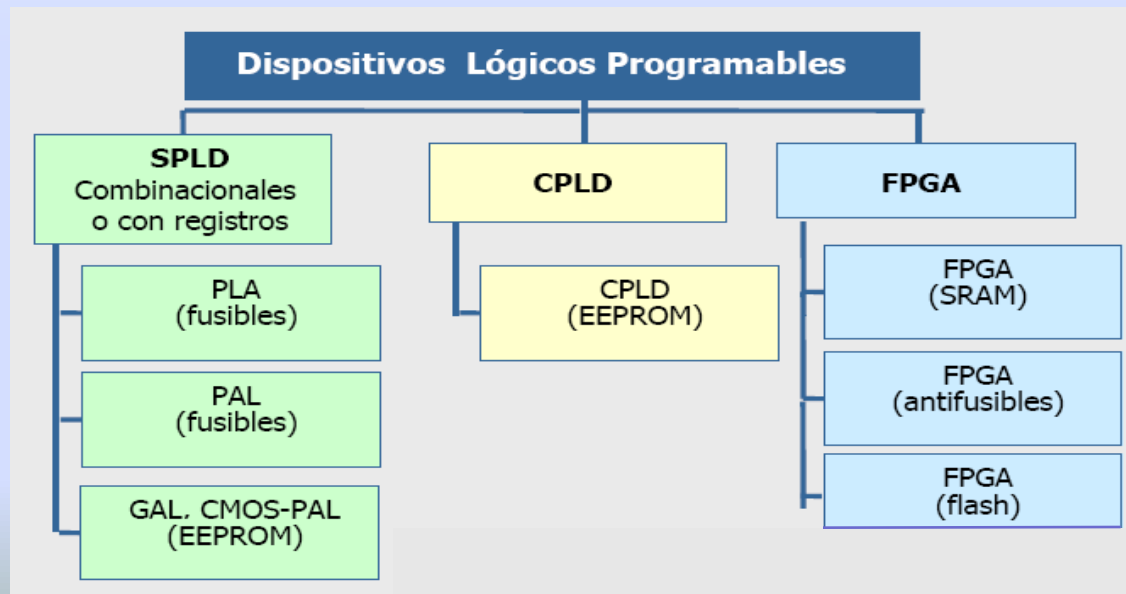
3.4.3 Codificadores/ Decodificadores

3.4.4 Multiplexores/ Demultiplexores

3.4.5 Dispositivos lógicos programables

3.4.5 Dispositivos lógicos programables

- Los **dispositivos lógicos programables** sustituyen en algunas aplicaciones a los circuitos SSI y MSI ya que ocupan menos, se necesitan menos unidades y su coste es inferior.
- Están formados por una **matriz de puertas AND y OR** que se puede programar.



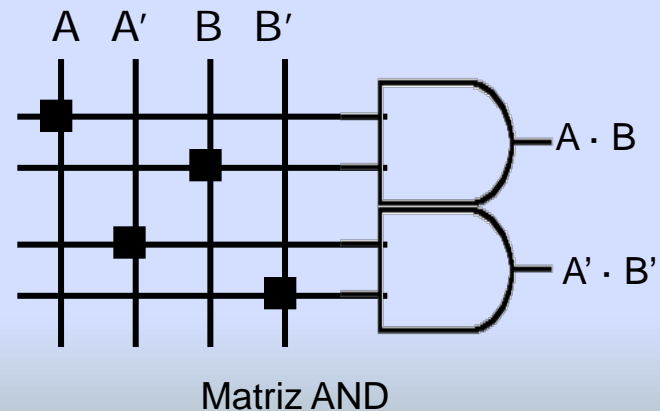
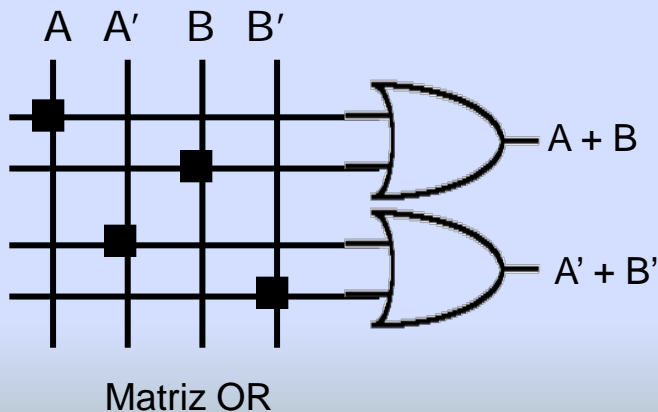
3.4.5 Dispositivos lógicos programables

- Los **PLD** tienen una estructura regular y se particularizan para realizar una aplicación concreta.
- Como salen de fábrica no realizan ninguna función. El diseñador lo programa o configura para que lleve a cabo una tarea determinada.
- Los PLD son **reconfigurables**, es decir, la programación se puede cambiar total o parcialmente y en algunos casos en tiempo real.
- Se **programan** creando o eliminando conexiones entre los distintos dispositivos electrónicos que lo forman.



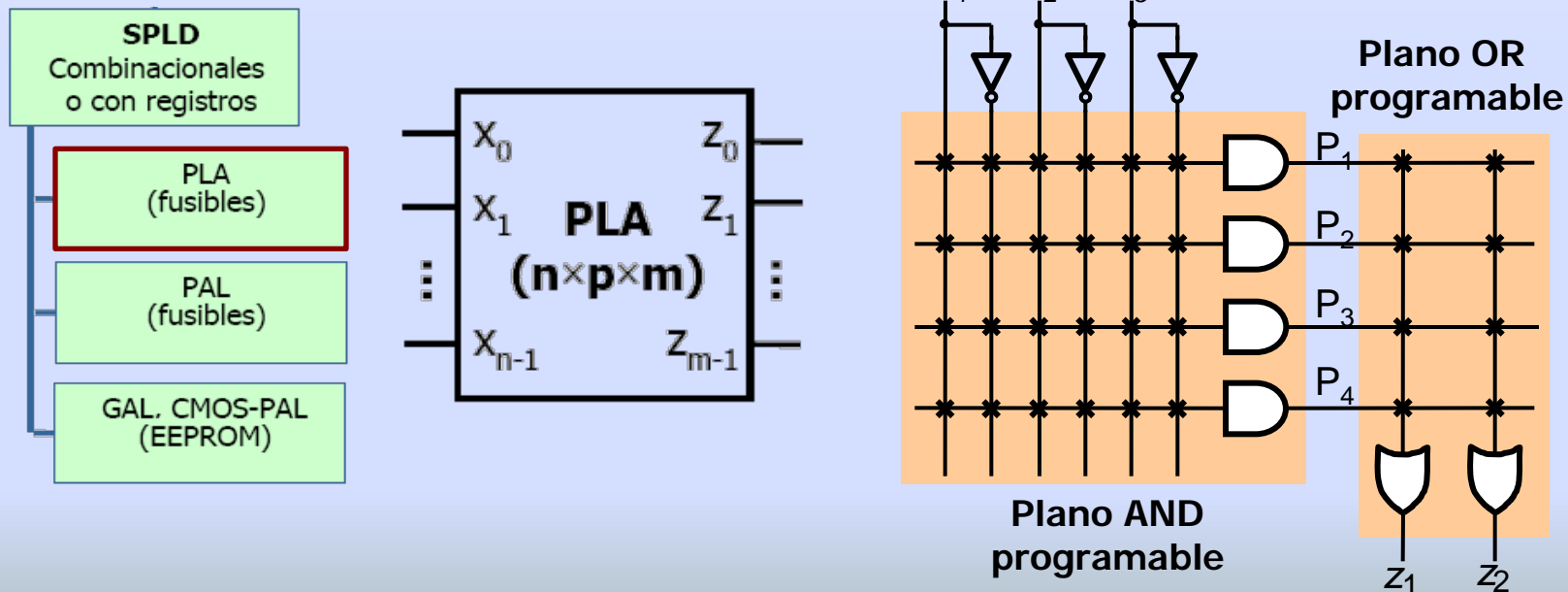
3.4.5 Dispositivos lógicos programables

- Un **plano** o **matriz programable** es una red de conductores en filas y columnas con un elemento electrónico en cada intersección.
 - Una **matriz o plano OR** está formada por puertas OR conectadas a una matriz programable.
 - Una **matriz o plano AND** está formada por puertas AND conectadas a una matriz programable.



3.4.5 Dispositivos lógicos programables

- **PLA** (Programmable Logic Array): Una PLA $n \times p \times m$ es un circuito combinacional con **n entradas**, formado por un plano AND programable con **p términos producto** y un plano OR también programable con **m salidas**.



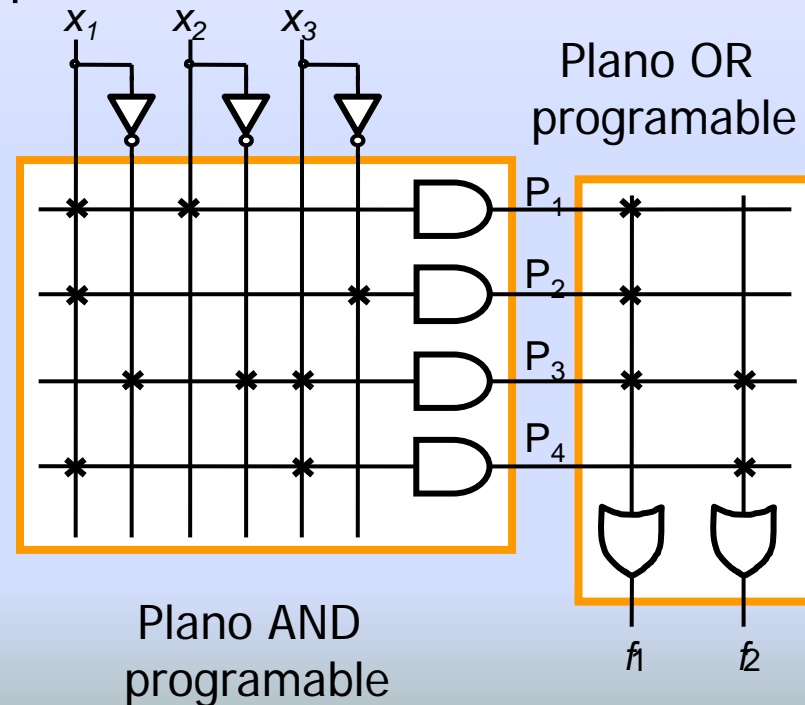
3.4.5 Dispositivos lógicos programables

- **Ejemplo:** Implementar con una PLA las siguientes funciones:

$$f_1 = x_1 \cdot x_2 + x_1 \cdot \bar{x}_3 + \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 = P1 + P2 + P3$$

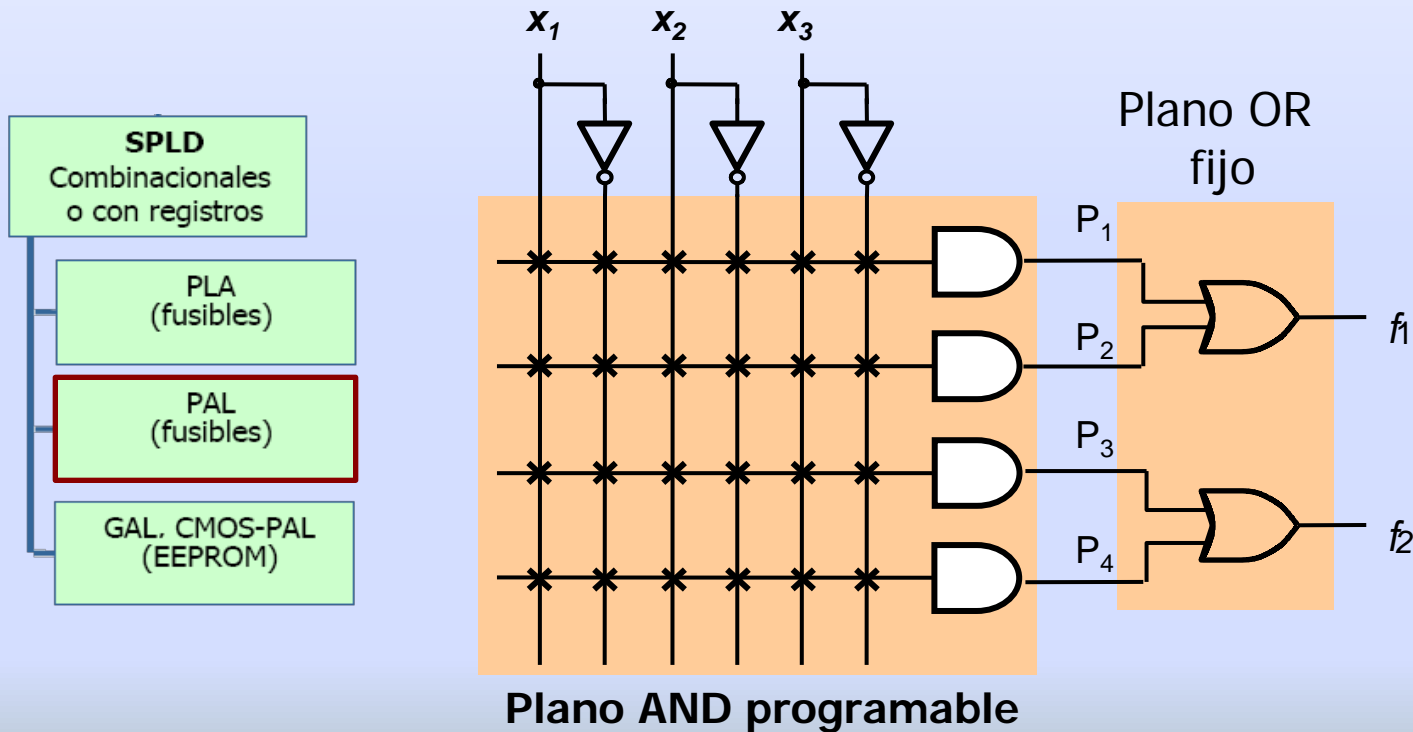
$$f_2 = \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 + x_1 \cdot x_3 = P3 + P4$$

- Términos producto de cualquier orden
- Se pueden compartir términos producto



3.4.5 Dispositivos lógicos programables

- **PAL** (Programmable Array Logic): esta estructura permite implementar cualquier suma de productos.



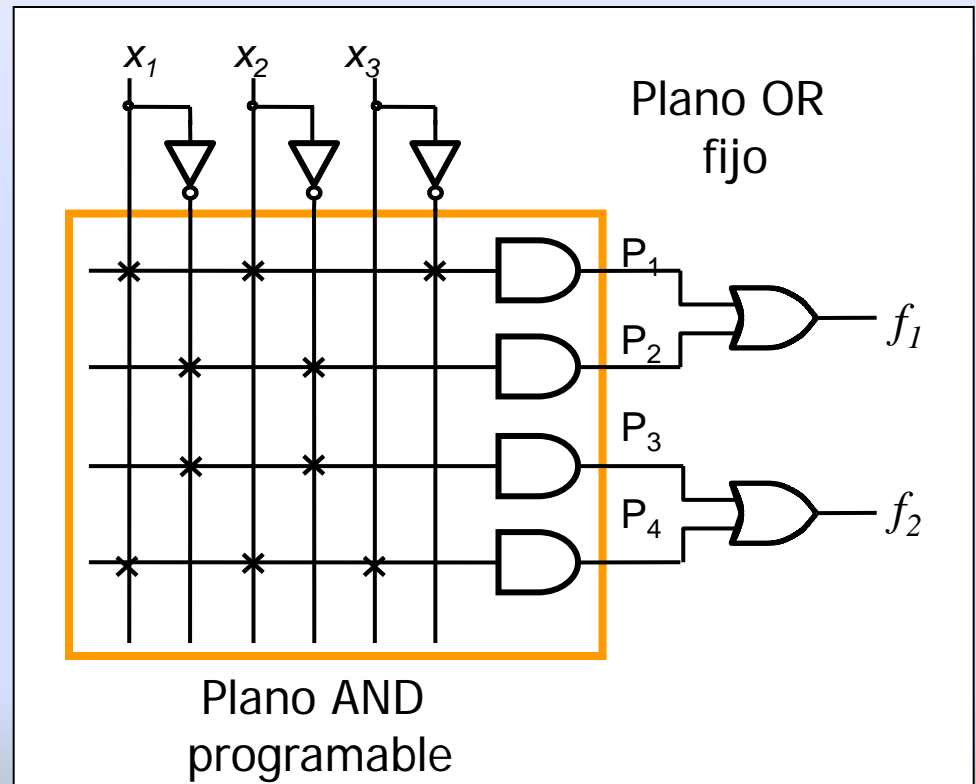
3.4.5 Dispositivos lógicos programables

- **Ejemplo:** implementar con una PAL las siguientes funciones:

$$f_1 = P1 + P2 = x_1 \cdot x_2 \cdot \bar{x}_3 + \bar{x}_1 \cdot \bar{x}_2$$

$$f_2 = P3 + P4 = \bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot x_2 \cdot x_3$$

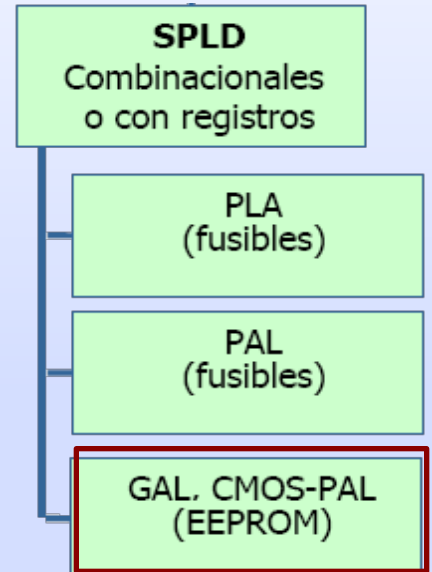
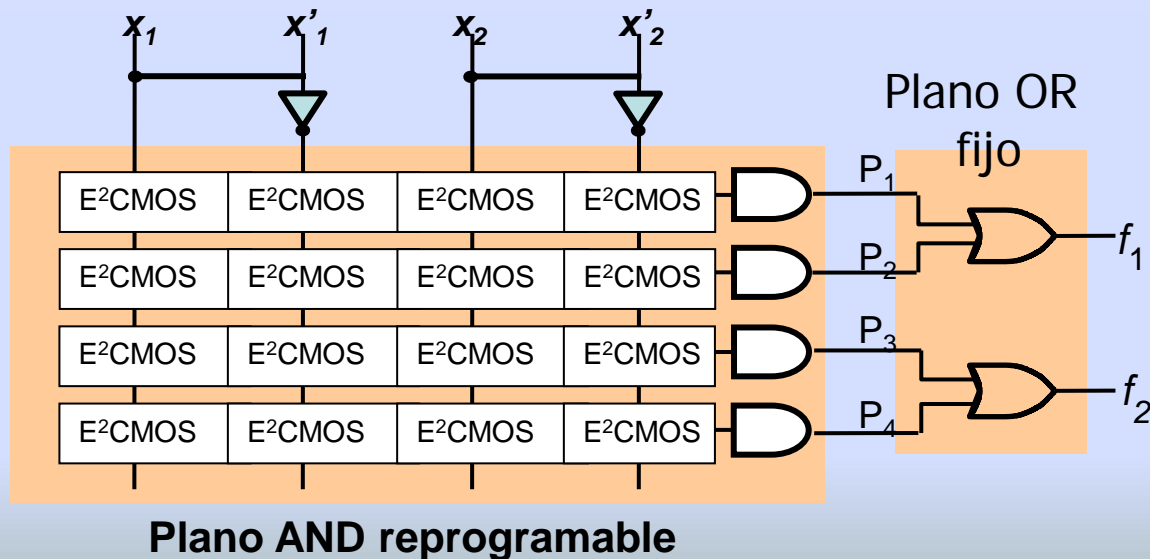
- Términos producto de cualquier orden
- No se pueden compartir términos producto
- Minimización independiente de las funciones
- ✓ Mayor densidad y menor retardo que PLAs



3.4.5 Dispositivos lógicos programables

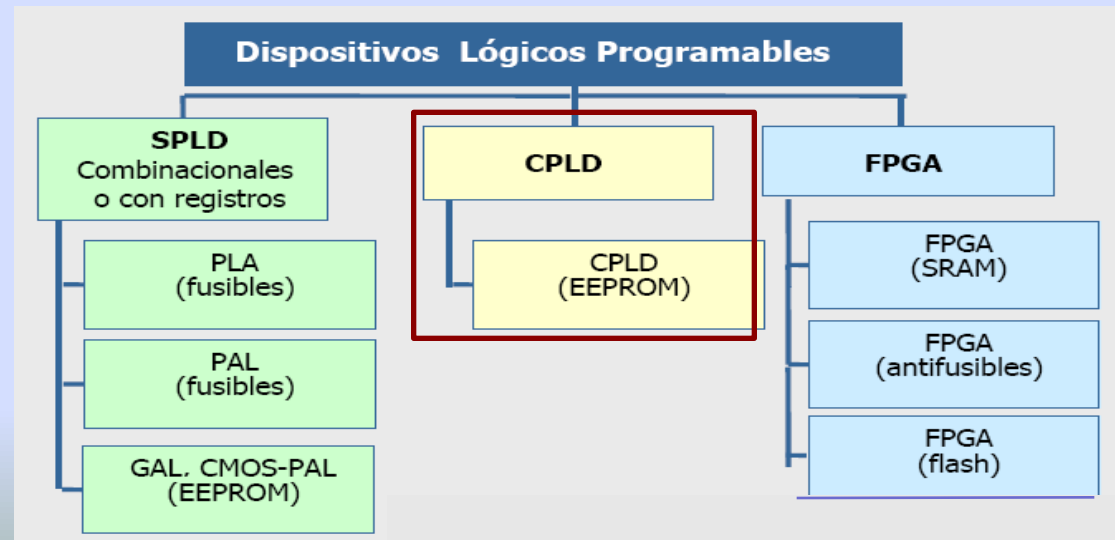
GAL (Generic Array Logic):

- Una GAL básica está formada por un plano AND programable y un plano OR fijo.
- La matriz programable está formada por CMOS borrables (E^2 CMOS) en lugar de fusibles (PAL).



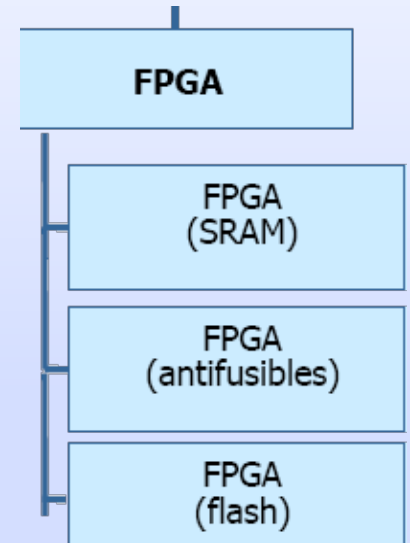
3.4.5 Dispositivos lógicos programables

- **CPLD** (Complex Programmable Logic Device): son una evolución de los SPLD.
 - Suelen estar formados por varios bloques PAL conectados mediante líneas centralizadas programables.
 - Todavía se siguen utilizando. Por ejemplo, los puertos PCI-Express están controlados mediante un CPLD.



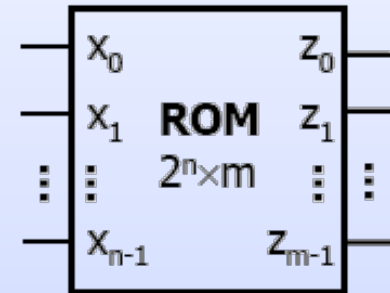
3.4.5 Dispositivos lógicos programables

- **FPGA** (Field Programmable Gate Array): son dispositivos lógicos programables de gran versatilidad.
 - Pueden implementar circuitos combinacionales y secuenciales de gran complejidad.
 - Los más básicos son homogéneos y los más complejos son heterogéneos.
 - **FPGA-SRAM**: volátiles, se puede reprogramar entre 10 y 100 veces, los que más se usan.
 - **FPGA antifusibles**: no volátil, muy estables, Se usan en aplicaciones espaciales
 - **FPGA flash**: Parecidos a los antifusibles pero con tecnología de transistores.



3.4.5 Dispositivos lógicos programables

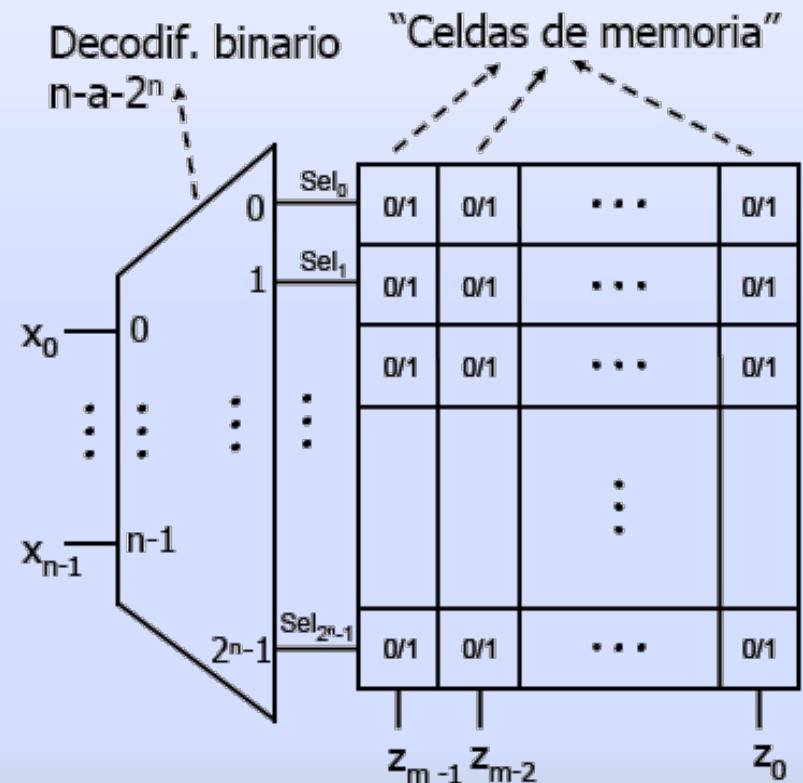
- Una **memoria** es una estructura lógica con n entradas de dirección, m salidas de datos y $m \times 2^n$ celdas de memoria.
- Además, puede tener m entradas de datos y p señales de control
- Almacena de forma permanente 2^n palabras de m bits.
- Si la información se puede cambiar (escribir) la memoria es de **lectura y escritura**, y tendrá además m entradas de datos.
- Si la información no se puede cambiar, es decir, es permanente, la memoria es de **solo lectura (ROM)**.



3.4.5 Dispositivos lógicos programables

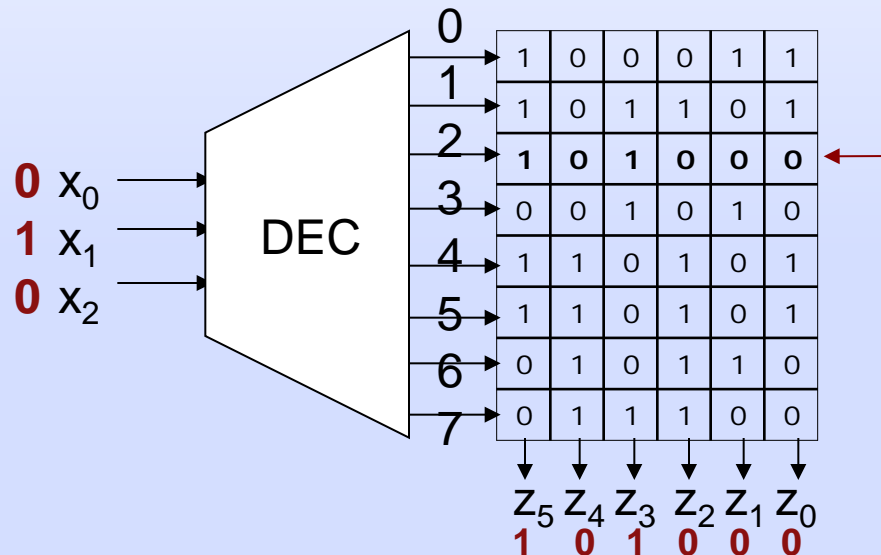
Memorias ROM:

- n entradas de dirección:
 x_{n-1}, \dots, x_0
- 2^n palabras de m bits:
 $(z_{m-1}, \dots, z_0) = M(x_{n-1}, \dots, x_0)$
- Formado por:
 - Decodificador binario de n a 2^n
 - $m \times 2^n$ celdas de memoria En cada celda se almacena 1 bit
- En la salida se obtiene la palabra de m bits (z_{m-1}, \dots, z_0) almacenada en la dirección que especifique su entrada de dirección (x_{n-1}, \dots, x_0)



3.4.5 Dispositivos lógicos programables

- Ejemplo:** memoria ROM de 8 palabras de 6 bits



A la salida se obtiene la palabra almacenada (101000) en la dirección indicada (010).

3.4.5 Dispositivos lógicos programables

- Las ROM son módulos lógicos universales:

$$D_7(x_2x_1x_0) = \sum m_i(0,2,3,5)$$

$$D_5(x_2x_1x_0) = \sum m_i(0,3,5,7)$$

$$D_3(x_2x_1x_0) = \sum m_i(1,4,6)$$

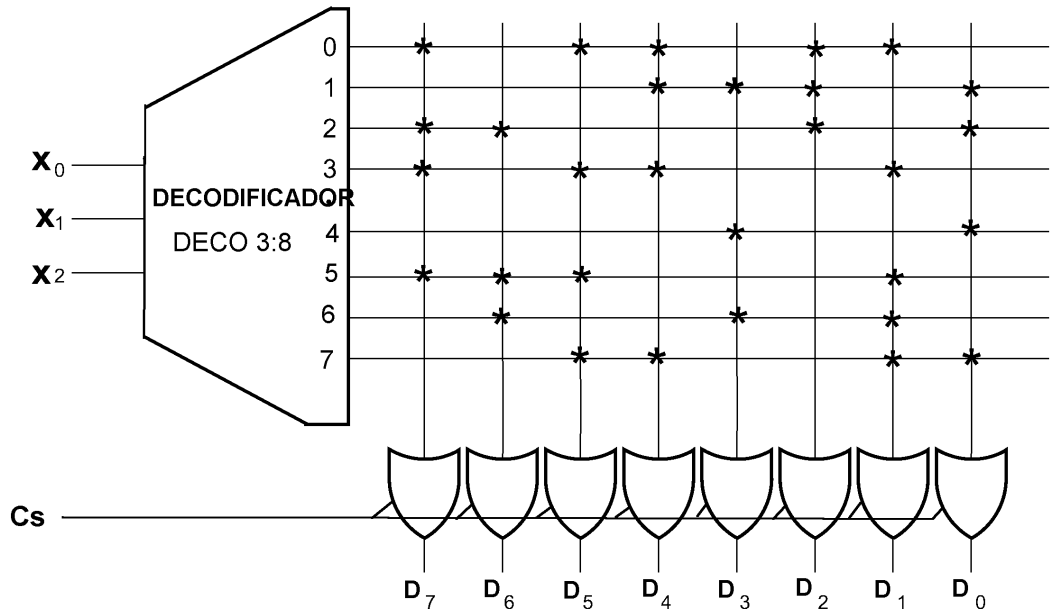
$$D_1(x_2x_1x_0) = \sum m_i(0,3,5,6,7)$$

$$D_6(x_2x_1x_0) = \sum m_i(2,5,6)$$

$$D_4(x_2x_1x_0) = \sum m_i(0,1,3,7)$$

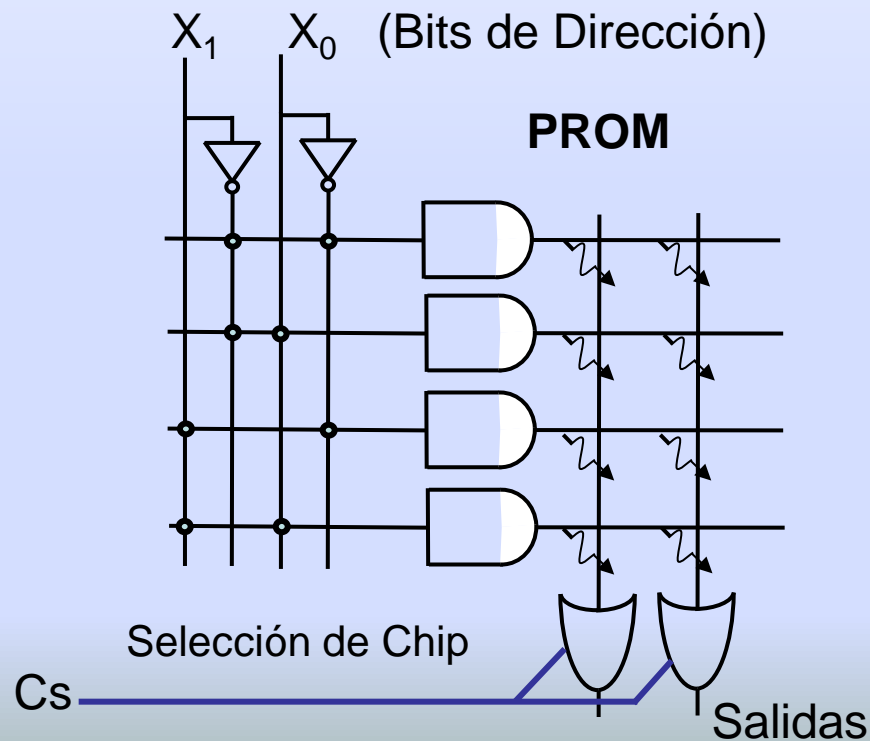
$$D_2(x_2x_1x_0) = \sum m_i(0,1,2)$$

$$D_0(x_2x_1x_0) = \sum m_i(1,2,4,7)$$



3.4.5 Dispositivos lógicos programables

- **PROM:** vienen con todas las conexiones hechas y la programación consiste en deshacer conexiones. Sólo se pueden programar una vez.
 - Ejemplo: PROM $2^2 \times 2$
- **EPROM:** ROM borrable, programable y reprogramable con luz ultravioleta.
- **EEPROM:** ROM borrable, programable y reprogramable eléctricamente (por ejemplo: memorias Flash)



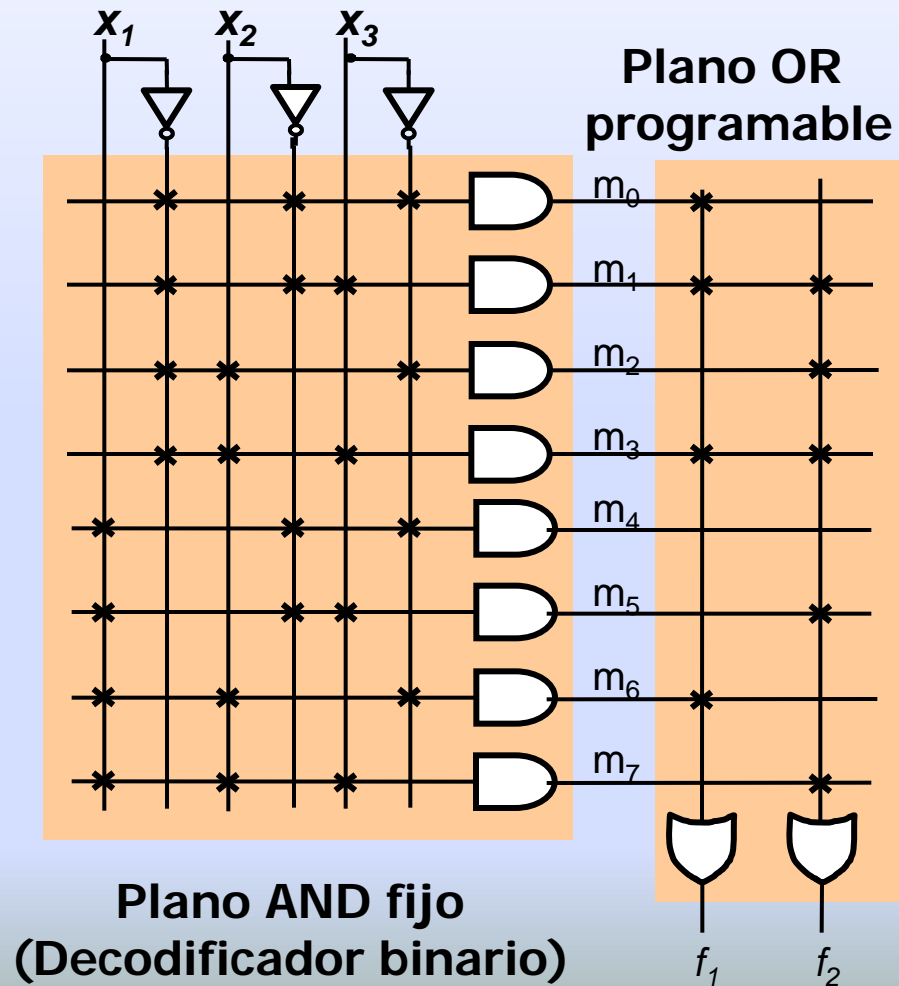
3.4.5 Dispositivos lógicos programables

ROM programable:

- Generación de funciones lógicas a partir de expresión canónica
- Cada nueva variable duplica el tamaño de la memoria
- Ejemplo:

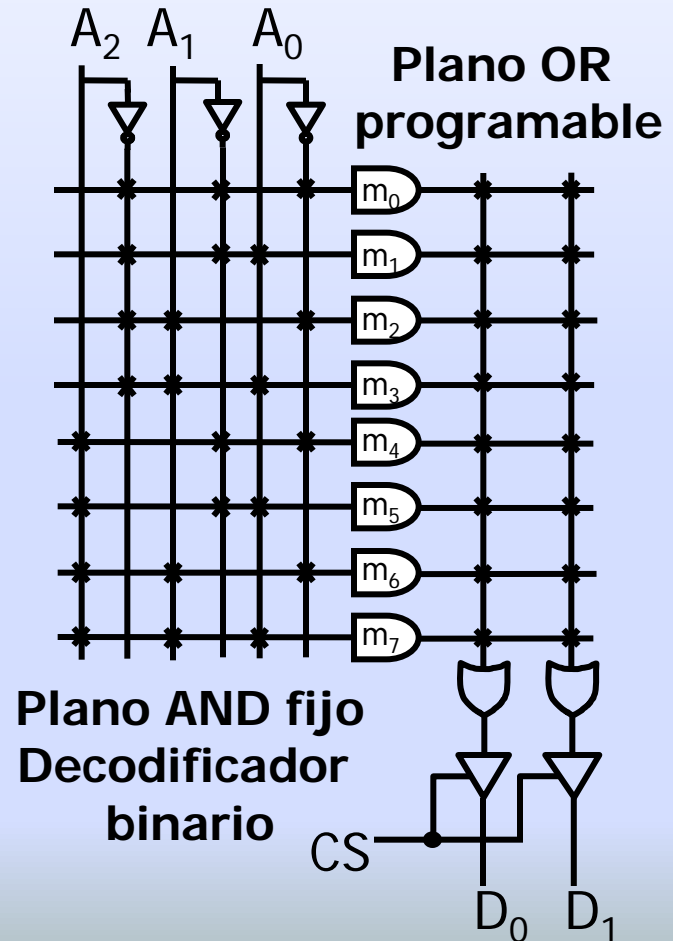
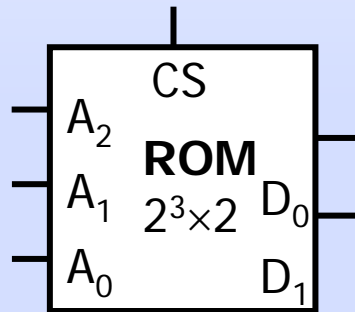
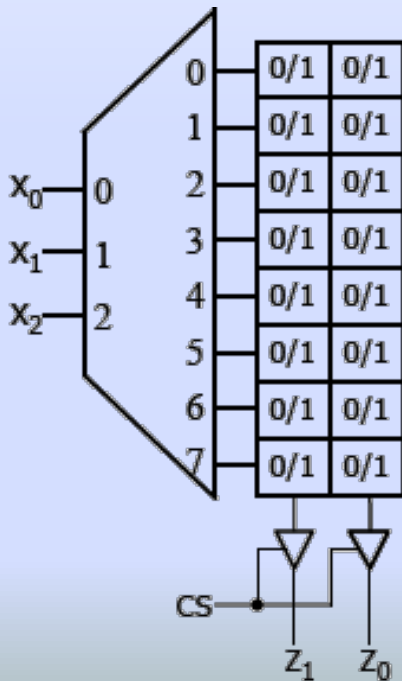
$$f_1(x_1, x_2, x_3) = \sum m(0, 1, 3, 6)$$

$$f_2(x_1, x_2, x_3) = \sum m(1, 2, 3, 5, 7)$$



3.4.5 Dispositivos lógicos programables

- ROM $2^3 \times 2$ con entrada de selección:
CS = 1: Funcionamiento normal
CS = 0: Todas las salidas en alta impedancia



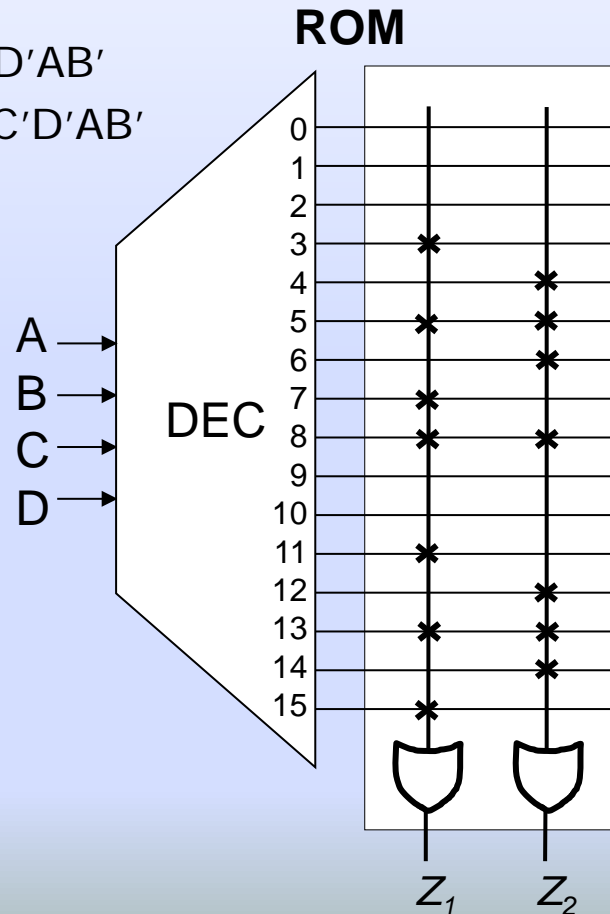
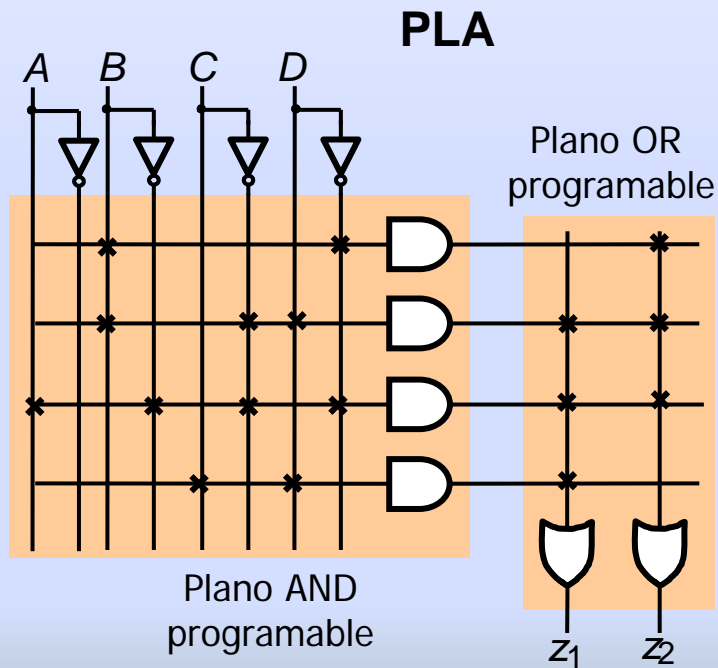
3.4.5 Dispositivos lógicos programables

- Implementación de funciones de conmutación con PLA y ROM.

Por ejemplo:

$$Z1(A,B,C,D) = D'B + C'DB + C'D'AB'$$

$$Z2(A,B,C,D) = CD + C'DB + C'D'AB'$$



A	B	C	D	Z2	Z1
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	1	1
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	1	1
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	1	1
1	1	1	0	1	0
1	1	1	1	0	1

Tema 3

ANÁLISIS Y DISEÑO DE SISTEMAS COMBINACIONALES

*TECNOLOGÍA Y ORGANIZACIÓN DE
COMPUTADORES*

1º Grado en Ingeniería Informática.