

# FUNDAMENTOS Y ESTRUCTURAS DE COMPUTADORES

Alejandro Gómez Amaro

Profesor: Manuel Agustín Ortiz López

10% Prácticas

10% Parcial de resolución de problemas (no  
eliminatorio)

80% Examen (Teoria + problemas) (Mínimo  
un 5)

# Introducción a los sistemas digitales

## Variables analógicas y digitales

- **Variables analógicas**: Puede tomar  **cualquier valor**. Su **conjunto de valores no es finito**.
- **Variables digitales**: Puede tomar **valores discretas**. Su **conjunto de valores es finito**.

## Sistemas analógicos y digitales

- **Sistema analógico** procesa variables **analógicas**
- **Sistema digital** procesa variables **digitales**

Ej: Termómetro analógico/digital

## Sistemas digitales

- **Sistema**: colección de **objetos** que **interactúan entre sí**
- **Objetos**: componentes.
- **Interacción** por enlaces
- **Estructura**: a través de un **diagrama de bloques**
- **Comportamiento** estableciendo **relaciones entrada - salida**



- $Z_i = f_{zi}(x_1, x_2, \dots, x_n)$
- **Diseño o síntesis de un sistema**: encontrar las **estructuras** del sistema a partir de la descripción de comportamiento.
- **Análisis de un sistema**: describir su comportamiento a partir de su estructura
- **Método lógico de diseño**: Nivel lógico y nivel físico

## Clasificación de sistemas digitales

- **Sistemas combinacionales:** Las salidas son función de las entradas que en ese momento se aplican. Cambio entrada → cambio salida. Se puede especificar con tablas de verdad

Entradas	Salidas

- **Sistemas secuenciales:** Sistemas con memoria. La salida no solo depende de las entradas del circuito en ese momento, también en anterioridad

# Sistemas de numeración posicional

- Un número se define mediante una cadena de dígitos, el valor de cada dígito depende de la posición que ocupe en la cadena, "tiene un peso".
- El peso se expresa en función de la potencia de la base ( $b$ )
- Considerando un sistema con  $n$  dígitos en la parte entera y  $m$  en la fraccionaria:

Ej:  $X = (a_{n-1} \cdot a_{n-2} \dots a_2 \cdot a_1 \cdot a_0, a_{-1}, a_{-2} \dots a_{-m})_b$

- Valor de pesos:

Ej:  $p = b^{n-1} b^{n-2} \dots b^2 b^1 b^0, b^{-1} b^{-2} \dots b^{-m}$

- Valor de los dígitos

Ej:  $0 \leq a_i < b$

- Valor del número  $X$

$$V(X) = a_{n-1} b^{n-1} + a_{n-2} b^{n-2} + \dots + a_2 b^2 + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + \dots + a_{-m} b^{-m}$$

## Sistemas de numeración posicionales

Nombre	Base	Dígitos
Binario	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- En los sistemas digitales se trabaja con un número infinito de " $n$ " dígitos llamado tamaño de la palabra

### Conversion binario → decimal

Ej:  $110010,011_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}$   
 $= 32 + 16 + 8 + 0,25 + 0,125 = 50,375_{10}$

### Conversion octal → decimal

Ej:  $567,45_8 = 5 \cdot 8^2 + 6 \cdot 8 + 7 \cdot 8^0 + 4 \cdot 8^{-1} + 5 \cdot 8^{-2}$   
 $= 5 \cdot 64 + 6 \cdot 8 + 7 \cdot 1 + 4 \cdot 0,125 + 5 \cdot 0,015625$   
 $= 375,578125_{10}$

### Conversion hexadecimal → decimal

Ej:  $C8E,AB_{16} = C \cdot 16^2 + 8 \cdot 16 + E \cdot 16^0 + A \cdot 16^{-1} + B \cdot 16^{-2}$   
 $= 12 \cdot 256 + 8 \cdot 16 + 14 \cdot 1 + 10 \cdot 0,0625 + 11 \cdot 0,00390625$   
 $= 3216,66796875_{10}$

### Conversion decimal → binario

Ej:  $325_{10}$

→  $101000101_2$

Conversion parte decimal decimal → binario

Ej:  $325,625_{10}$

$$\begin{array}{r} 0,625 \cdot 2 = 1,25 \rightarrow 1 \\ 0,25 \cdot 2 = 0,5 \rightarrow 0 \\ 0,5 \cdot 2 = 1 \end{array} \left. \begin{array}{l} \\ \\ \end{array} \right\} 101000101,101_2$$

Conversion decimal → octal

Ej:  $478_{10}$

$$\begin{array}{r} 8 \\ 59 \\ \hline 6 \quad 7 \\ \hline 3 \quad 7 \\ \hline 1 \quad 0 \\ \hline 2 \end{array} \rightarrow 736_8$$

Conversion parte decimal decimal → octal

Ej:  $478,58_{10}$

$$\begin{array}{r} 0,58 \cdot 8 = 4,64 \rightarrow 4 \\ 0,64 \cdot 8 = 5,12 \rightarrow 5 \\ 0,12 \cdot 8 = 0,96 \rightarrow 0 \\ 0,96 \cdot 8 = 7,68 \rightarrow 7 \end{array} \left. \begin{array}{l} \\ \\ \\ \end{array} \right\} 736,4507_8$$

Conversion decimal → hexadecimal

Ej:  $478_{10}$

$$\begin{array}{r} 16 \\ 29 \\ \hline 14 \quad 16 \\ 13 \quad 1 \\ \hline 1 \quad 0 \\ \hline 2 \end{array} \rightarrow 1DE_{16}$$

### Conversión parte decimal decimal $\rightarrow$ hexadecimal

Ej:  $478,58_{10}$

$$\left. \begin{array}{l} 0,58 \cdot 16 = 9,28 \rightarrow 9 \\ 0,28 \cdot 16 = 4,48 \rightarrow 4 \\ 0,48 \cdot 16 = 7,68 \rightarrow 7 \\ 0,68 \cdot 16 = 10,88 \rightarrow A \end{array} \right\} 100,947A_{16}$$

### Conversión binario $\leftrightarrow$ octal

Ej:  $1100101,1101_2$

$001$	$100$	$101$	$,$	$110$	$100$
↑	↑	↓		↑	↑
$1$	$4$	$5$	$,$	$6$	$4$

$145,64_8$

### Conversión binario $\leftrightarrow$ hexadecimal

Ej:  $1101111,11011_2$

$0110$	$1111$	$,$	$1101$	$1000$
↓	↓		↓	↓
$6$	$F$	$,$	$D$	$8$

$6F,D8_{16}$

### BCD (Decimal Codificado Binario)

Código BCD 8421	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
Decimal	0	1	2	3	4	5	6	7	8	9

Ej:  $32,84_{10}$

$3$	$2$	$,$	$8$	$4$
↑	↑		↑	↑
$0011$	$0010$	$,$	$1000$	$0100$

$0011\ 0010,\ 1000\ 0100_{BCD}$

## Código continuo y cíclico

**Continuo:** Código binario en el que las combinaciones binarias correspondientes a los números decimales consecutivos son adyacentes (solo cambia un bit):

Ej:  $00 \rightarrow 01$  es adyacente  
 $01 \rightarrow 10$  no es adyacente

**Cíclico:** Código binario continuo donde la última combinación es adyacente a las primeras

## Código Gray

1 bit	2 bits	3 bits
0	00	000
1	<u>01</u>	001
	11	011
	<u>10</u>	<u>010</u>
	110	
	111	
	101	
	100	

- Código continuo, cíclico sin peso
- Capacidad de codificación del binario natural ( $C = 2^n$ )
- Código reflejado
- Formado a partir de  $n-1$  bits, reflexión espejada, añadiendo en la izquierda un bit de valor 0 en las primeras  $2^{n-1}$  combinaciones y un 1 en el resto.

## Código Johnson

Decimal	Johnson
0	00000
1	00001
2	00011
3	00111
4	01111
5	11111
6	11110
7	11100
8	11000
9	10000

- Caducidad de codificación de  $2^n$  números diferentes para códigos de  $n$  bits
- Representar dígitos decimales
  - 1. Se calcula el número de bits  
 $\text{Nº valores} = 2^n$
  - Ej:  $10 = 2 \cdot 5 \quad n = 5$
  - 2. La combinación correspondiente al 0  
 $0 = 00000$
  - 3. El resto de números se obtienen con la operación de rotación a la izquierda

## Sistema de representación exceso a M

- Se suma M a los números y se representa en binario natural

Ej: BCD exceso a 3

Decimal	BCD	BCD exces
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

## Código ASCII

$b_3 b_2 b_1 b_0$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	'	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	,	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	-	o	DEL

## Códigos detectores y correctores de errores

- Los sistemas digitales intercambian información mediante algunos códigos (BCD, ASCII, etc...)
- El sistema que envía la información se denomina transmisor
- El sistema que recibe la información se denomina receptor
- Generalmente los sistemas pueden ser transmisores y receptores
- Se pueden producir errores de comunicación por interferencia o averías.
- Se han generado códigos que permiten saber si la información recibida es correcta e incluso algunos que permiten corregir el error



## Códigos detectores de errores

- Requisitos para que un código pueda detectar errores
  - Un código es un código de detección de errores si puede detectar combinaciones binarias que no son el código.
  - La condición para ello es su distancia mínima
  - Distancia
    - La distancia entre dos combinaciones binarias es el número de bits que hay que cambiar para obtener la otra
- $D(0000, 0001) = 1$
- $D(0000, 1111) = 4$
- Distancia mínima ( $D_m$ )
  - menor distancia de todas las posibles parejas de ese código
  - Para que un código pueda detectar errores en un bit su  $D_m$  debe de ser mayor que 1

Ej: Código de 3 bits con posibles 000, 011, 100, 110

No están en el código 001, 010, 101, 111

Si enviamos 100 podríamos recibir por error 110 y 000, pero no 101.  $D_m = 1$

## Códigos de paridad

- Se forma añadiendo al código Dm un bit 'paridad'
- Hay dos tipos
  - Par: El bit paridad toma el valor necesario para que un número de bits con valor 1 en las posiciones del nuevo código sea par. O se considera par
  - Impar: El bit paridad toma el valor necesario para que un número de bits con valor 1 en las posiciones del nuevo código sea impar
- Código de paridad a partir del código binario natural de 3 bits
  - Paridad par:  $b_2, b_1, b_0, P$ 
    - P es 1 si el número de 1 en los bits de Atas es impar
    - P es 0 si el número de 1 en los bits de Atas es par
  - Paridad impar:  $b_2, b_1, b_0, I$ 
    - I es 1 si el número de 1 en los bits de Atas es par
    - I es 0 si el número de 1 en los bits de Atas es impar

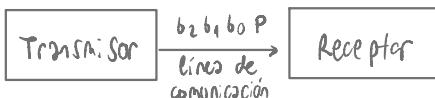
Binario de 3 bits		
$b_2$	$b_1$	$b_0$
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Paridad Par			
$b_2$	$b_1$	$b_0$	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Paridad Impar			
$b_2$	$b_1$	$b_0$	I
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

$$Dm = 2$$

- Comprobación de paridad
  - El transmisor y receptor wan el mismo código de paridad
  - Supongamos el código binario de 3 bits con paridad par



- El receptor comprueba la paridad generando el bit de la paridad par de los palabras recibidas ( $b_2, b_1, b_0, P$ )
  - Si el bit es 0 es correcto
  - Si el bit es 1 es incorrecto
- El transmisor envía el 2 (0101)
  - El receptor lo recibe correctamente:  
 $PP(0101) = 0$
  - El receptor lo recibe incorrectamente:  
 $PP(1101) = 1$
  - Un código de paridad puede detectar los cambios en un número impar de bits pero no par.

# Álgebra de Comutación

## Postulados del Álgebra de Boole

- Un álgebra es una estructura matemática que comprende un conjunto de elementos y operaciones que actúan sobre ellos.
- Los postulados determinan como se realizan dichas operaciones
- Los postulados no se demuestran y permiten deducir los teoremas y propiedades de dicha estructura.
- Se suelen usar los axiomas de Huntington de 1904 para definir el Álgebra de Boole.
- Sea un conjunto de elementos  $B$ , en el que podemos establecer una relación de equivalencia y donde se verifica el principio de sustitución
  - $\exists x, y \in B \mid x = y$
  - En elquier expresión donde aparezca  $x$  o  $y$  se podrá sustituir por la otra

## • Postulados

### 1º Leyes de Composición interna

+ (operación OR) • (operación AND)

$$\forall x, y \in B \rightarrow x+y \in B \quad x \cdot y \in B$$

### 2º Comutatividad de las leyes de composición interna

$$\forall x, y \in B \rightarrow x+y = y+x \quad x \cdot y = y \cdot x$$

### 3º Elementos neutros

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

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

### 4º Distributividad de las leyes de composición interna

$$\forall x, y, z \in B, x+(y \cdot z) = (x \cdot y) + (x \cdot z)$$

$$\forall x, y, z \in B, x \cdot (y+z) = (x \cdot y) + (x \cdot z)$$

### 5º Elemento opuesto

$$\forall x \in B \quad \exists \bar{x} \in B / x + \bar{x} = 1 \quad x \cdot \bar{x} = 0$$

$\bar{x}$  se denomina complemento de  $x$

### 6º Número de elementos

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

En el conjunto  $B$  existen por lo menos 2 elementos

## • Teoremas

- Para cada teorema existen dos enunciados
- No es necesario demostrar ambos enunciados, si demuestra uno y el otro queda aprobado por el principio de dualidad

Teorema 1.  $\forall x \in B_2 \quad x+1=1 \quad x \cdot 0=0$

Teorema 2. Idempotencia.  $\forall x \in B_2 \quad x+x=x \quad x \cdot x=x$

Teorema 3. Invención.  $\forall x \in B_2 \quad x=x$

Teorema 4. Absorción.  $\forall x \in B_2 \quad x+x \cdot y=x \quad x \cdot (x+y)=x$

Teorema 5.  $\forall x,y,z \in B_2 \quad x+[(x \cdot y) \cdot z]=x \quad x \cdot [(x+y)+z]=x$

Teorema 6. Asociativa  $\forall x,y,z \in B_2 \quad x+(y+z)=(x+y)+z$   
 $x \cdot (y \cdot z)=(x \cdot y) \cdot z$

Teorema 7.  $\forall x,y \in B_2 \quad x+\bar{x} \cdot y=x+y \quad x(\bar{x}+y)=x \cdot y$

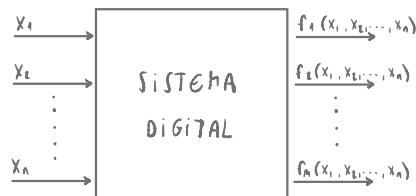
Teorema 8. Leyes de Morgan.  $\forall x,y \in B_2 \quad \overline{x+y}=\bar{x} \cdot \bar{y} \quad \overline{x \cdot y}=\bar{x}+\bar{y}$

## Álgebra de Comutación.

- El álgebra de comutación (A.C) es un álgebra de Boole que emplea únicamente  $B=\{0,1\}$  ( $B_2$ )
- Los operadores cumplen los postulados de Huntington
- Los elementos 0 y 1 corresponden a los valores binarios
- Principio de dualidad
  - En una igualdad si se sustituye 0 por 1, + por \* y viceversa se obtiene otra igualdad
  - Si se parte del enunciado a) y se aplica el principio de dualidad se obtiene b) y viceversa

## funciones lógicas y conmutación

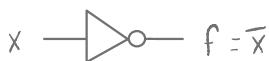
- Una función lógica  $f$ , representada como  $f(x_1, x_2, \dots, x_n)$  donde  $x_1, x_2, \dots, x_n$  son las variables de entrada y  $f$  es la salida
- Las señales de entrada y salida digital solo pueden tomar valores 0 y 1:
  - se pueden representar mediante variables lógicas
  - las señales de salida se pueden expresar matemáticamente mediante una función lógica
  - El álgebra de conmutación permite el análisis y diseño de sistemas digitales



## función NOT ( $\neg$ )

- $f(x) = \bar{x}$

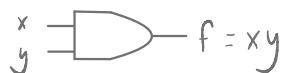
$x$	$f$
0	1
1	0



## función AND ( $\bullet$ )

- $f(x, y) = x \bullet y$

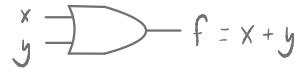
x	y	f
0	0	0
0	1	0
1	0	0
1	1	1



## función OR (+)

- $f(x, y) = x + y$

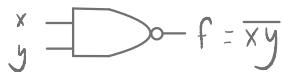
x	y	f
0	0	0
0	1	1
1	0	1
1	1	1



## función NAND

- $f(x, y) = \overline{xy}$

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



## función NOR

- $f(x, y) = \overline{x+y}$

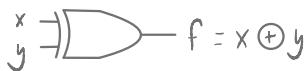
x	y	f
0	0	1
0	1	0
1	0	0
1	1	0



## función XOR ( $\oplus$ )

- $f(x, y) = x \oplus y = \bar{x}y + x\bar{y}$

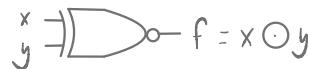
x	y	f
0	0	0
0	1	1
1	0	1
1	1	0



## función XNOR ( $\circ$ )

- $f(x, y) = x \circ y = \bar{x}\bar{y} + xy$

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



# Funciones lógicas

Concepto de minterm y maxterm

- Término producto

- Serie de variables conectadas mediante operador AND

Ej:  $xyz \quad \bar{x}\bar{y}\bar{z}$

- Dado un conjunto de variables lógicas,  $x, \dots, x_n$ , un minterm (Término mínimo) o Producto Fundamental de esas variables es cualquier término producto que aparece en todas las variables solo una vez

Ej:  $(x, y, z, v)$ :

$xyzu \quad \bar{x}\bar{y}\bar{z}v \quad \bar{x}\bar{y}\bar{z}\bar{v}$  son minterms

$xzu \quad \bar{x}\bar{y}\bar{z} \quad \bar{x}\bar{v}$  son términos producto

- Si se consideran los minterms como funciones lógicas de  $n$  variables de entrada cada minterm es 1 para una única combinación de valores de entrada

Ej: Dadas variables  $(x, y, z)$ , el minterm  $\bar{x}\bar{y}\bar{z}$

Es 1 solo si  $x=0, y=1$  y  $z=0$ , para el resto es 0

- Para asignar un único 1 se denomina término mínimo
- Para un caso de  $n$  variables se pueden formar  $2^n$  minterms
- Los minterms pueden nombrarse de manera simplificada por el equivalente en decimal de la combinación de variables que hacen que valga 1

Ej: Valen 1 para  $x=y=z=1$  (111) que equivale a un decimal 7 ( $m_7$ )

$\bar{x}\bar{y}\bar{z}$  vale 1 para (010) ( $m_2$ )

- Para  $n = 3$  las tablas de minterms

x	y	z	Minterms	
0	0	0	$\bar{x}\bar{y}\bar{z}$	$m_0$
0	0	1	$\bar{x}\bar{y}z$	$m_1$
0	1	0	$\bar{x}yz$	$m_2$
0	1	1	$\bar{x}yz$	$m_3$
1	0	0	$xy\bar{z}$	$m_4$
1	0	1	$xyz$	$m_5$
1	1	0	$xyz$	$m_6$
1	1	1	$xyz$	$m_7$

## Concepto de maxterm

- Término suma

Serie de variables conectadas mediante operador OR

Ej:  $x+y+z \quad \bar{x}+y+\bar{z}$

Dado un conjunto de variables lógicas,  $x, \dots, x_n$ , un maxterm

(término máximo) o suma fundamental de esas variables u cualquier término

que aparece en todas las variables solo una vez

Ej:  $(x, y, z, v)$ :

$x+y+z+v \quad \bar{x}+y+\bar{z}+v \quad \bar{x}+\bar{y}+z+\bar{v}$  son maxterminos

$x+z+v \quad \bar{x}+y+\bar{z} \quad \bar{x}+\bar{v}$  son términos suma

- Si se consideran los maxterminos como funciones lógicas de  $n$  variables de entrada cada maxterm es 0 para una única combinación de valores de entrada

Ej: Dadas variables  $(x, y, z)$ , el maxterm  $\bar{x}y\bar{z}$

Es 0 solo si  $x=1, y=0$  y  $z=1$ , para el resto es 1

- Por asignar un único 0 se denominan términos máximos

- Para un caso de  $n$  variables se pueden formar  $2^n$  maxterms

- Los minterms pueden nombrarse de manera simplificada por el equivalente en decimal de la combinación de variables que hacen que valga 0

Ej: Valore 0 para  $x=y=z=0$  (000) que equivale a un decimal 0 ( $m_0$ )

- Para  $n = 3$  las tablas de minterms

X	y	z	Minterms
0	0	0	$x+y+z$ $m_0$
0	0	1	$x+y+\bar{z}$ $m_1$
0	1	0	$\bar{x}+y+z$ $m_2$
0	1	1	$\bar{x}+y+\bar{z}$ $m_3$
1	0	0	$\bar{x}+y+z$ $m_4$
1	0	1	$\bar{x}+y+\bar{z}$ $m_5$
1	1	0	$\bar{x}+\bar{y}+z$ $m_6$
1	1	1	$\bar{x}+\bar{y}+\bar{z}$ $m_7$

## Teorema de Shannon

### Primer teorema

- "Toda función lógica o de conmutación se puede expresar como una suma única de minterms"

Ej: El desarrollo de Shannon para n variables en su primera forma:

$$F(x_1, \dots, x_n) = \bar{x}_1 \dots \bar{x}_n F(0, \dots, 0) + \bar{x}_1 \dots \bar{x}_{n-1} \cdot x_n F(0, \dots, 0, 1) \\ + \bar{x}_1 \dots x_{n-1} \cdot \bar{x}_n F(0, \dots, 1, 0) + \dots x_1 \dots x_n F(1, \dots, 1)$$

- Se obtienen  $2^n$  sumandos, con el producto de uno de los  $2^n$  minterms por una constante 0 o 1 (valor de la función para minterm)
- "Toda función lógica o de conmutación se puede expresar como una suma única de minterms, en concreto mediante la suma de los minterms para los que la función vale 1"
- El teorema de Shannon permite pasar de tablas de verdad a expresión lógica
- Una función lógica de n variables tendrá  $2^n$  minterms correspondientes a los valores de entrada de los filas

Ej:

$m_i$	x	y	z	f	
0	0	0	0	0	
1	0	0	1	1	$\bar{x}\bar{y}z$
2	0	1	0	1	$\bar{x}yz$
3	0	1	1	0	
4	1	0	0	1	$x\bar{y}\bar{z}$
5	1	0	1	0	
6	1	1	0	0	
7	1	1	1	1	$xyz$

$$f(x, y, z) = \bar{x}\bar{y}z + \bar{x}yz + x\bar{y}\bar{z} + xyz$$

## Segundo teorema

- "Toda función lógica de conmutación se puede expresar como un producto único de maxterms"

Ej: El desarrollo de Shannon para n variables en su segunda forma:

$$f(x_1, \dots, x_n) = (x_1 + \dots + x_n + F(0, \dots, 0))(x_1 + \dots + x_{n-1} + \bar{x}_n + F(0, \dots, 0, 1)) \\ (x_1 + \dots + \bar{x}_{n-1} + x_n + F(0, \dots, 1, 0))(\dots)(\bar{x}_1 + \dots + \bar{x}_n + F(1, \dots, 1))$$

- Se obtienen  $2^n$  productos, con los  $n$  ms de uno de los  $2^n$  maxterms por ms constante 0 o 1 (valor de la función para maxterm)
- "Toda función lógica o de conmutación se puede expresar como un producto único de maxterms, en concreto mediante el producto de los maxterms para los que la función vale 0"
- El teorema de Shannon permite pasar de tablas de verdad a expresión lógica
- Una función lógica de  $n$  variables tendrá  $2^n$  maxterms correspondientes a los valores de entrada de los filos

Ej:

M <sub>i</sub>	x	y	z	f	
0	0	0	0	0	x+y+z
1	0	0	1	1	
2	0	1	0	1	
3	0	1	1	0	x+y+ $\bar{z}$
4	1	0	0	1	
5	1	0	1	0	$\bar{x}+y+\bar{z}$
6	1	1	0	0	$\bar{x}+y+z$
7	1	1	1	1	

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

## funciones lógicas incompletamente especificadas

- las funciones lógicas analizadas asignan el valor 0 o 1 a las salidas
- se determinan funciones lógicas completamente especificadas
- funciones lógicas incompletamente especificadas
  - Aquellas en las que en algunas combinación de las variables de entrada es indiferente.

Ej: En una función lógica de código BCD. Para los valores de entradas 10 al 15 es indiferente el valor que se asigne a las salidas

- Representación en tablas de verdad
  - Se pone un guión en las posiciones de las columnas de las variables de salida correspondiente a minterms o maxterms para los que la función tiene un valor indiferente
  - Este guión sera 0 o 1, según interese en la simplificación
  - Abreviadamente las indiferencias se nombran d

Ej: do y d4

m	x	y	z	f
0	0	0	0	-
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	-
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

## Adyacencias

- Se analizan una función formada por minterms (válida en maxterms)

Ej:  $\left. \begin{array}{l} x\bar{y}\bar{z} \\ x\bar{y}z \\ x\bar{y}\bar{z} \\ xy\bar{z} \end{array} \right\} x\bar{y}$  } adyacencias de orden 1

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

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

- Dos adyacencias de 1º orden forman una de 2º si les falta la misma variable y se diferencian por una única variable, que en una está complementada y en otra no

Ej:  $\left. \begin{array}{l} x\bar{y} \\ xy \end{array} \right\} x$  adyacencia de orden 2

- Una adyacencia de orden 2 representa de forma algebraica o lógica por un único producto de las n-2 variables que coinciden y re elimina la variable que diferencia

Ej:  $x\bar{y} + xy = x(\bar{y} + y) = x \cdot 1 = x$

• Una adyacencia de 2º orden cubre a dos de 1º y a cuatro minterms

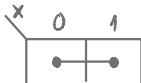
• Una adyacencia de orden n cubre a 2 de orden n-1 y a 2^n minterms

Ej:  $f(x,y,z) = x$  es una función lógica simplificada o realización mínima

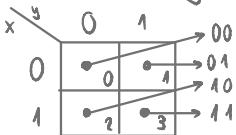
## Simplificación mediante mapas de Karnaugh

- El mapa de Karnaugh de  $n$  variables está formado por  $2^n$  casillas organizadas en filas y columnas coordinadas.
- Para representar las funciones lógicas se dividen las entradas en filas y columnas.

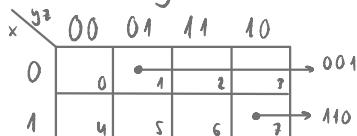
Ej: 1 variable.  $f(x)$



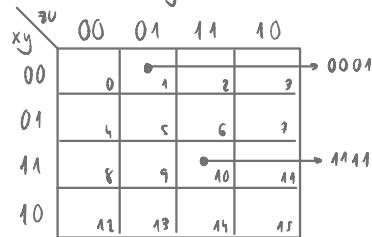
Ej: 2 variables.  $f(x,y)$



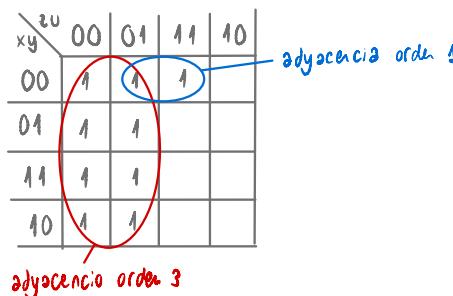
Ej: 3 variables.  $f(x,y,z)$



Ej: 4 variables.  $f(x,y,z,u)$



- **Características de los mapas de Karnaugh**
  - Las coordenadas de los casillar que comparten un lado son adyacentes
  - Las filas superior e inferior son adyacentes
    - Los cellos de la misma columna son adyacentes
  - Las columnas de izquierdo y derecho son adyacentes
    - Los cellos de la misma fila son adyacentes
- **Representación de una función lógica en un mapa de Karnaugh**
  - Para representar una función lógica de n variables se usa un mapa de n variables
  - A cada casilla le corresponde un minterm o maxterm de la función
  - En cada casilla se escribe 0, 1 o -.
  - Si se representa una función con minterms se representan solo 1 y -, si es de maxterms 0 y - (donde - son indiferentes)
  - Mediante mapas de Karnaugh se determinan **adyacencias**
    - Orden 0 : 1 casilla
    - Orden 1 : 2 casillas vecinas
    - Orden 2 : 4 casillas vecinas
    - Orden n :  $2^n$  casillas vecinas



- Procedimiento de simplificación de mapas de Karnaugh
  - Se plantea una función lógica representada como suma de minterms o como producto de maxterms
  - Función representada como suma de minterms
    1. Se representa en mapa de Karnaugh
    2. Obtener el menor número de adyacencias del mayor nivel posible (usando minterms 1 y indiferentes - , según las siguientes condiciones:
      - Las adyacencias constan de  $2^k$  casillas vecinas con valor 1 o -.
      - Las minterms deben quedar cubiertas, los indiferentes no tienen porque (solo para formar adyacencias).
  - Las minterms o adyacencias pueden warse en varias adyacencias
    - Se comienza con los minterms que solo pueden estar cubiertos por una única adyacencia. Se forman de mayor orden parible, adyacencias encerrables
    - Se forma el mínimo número de adyacencia de mayor orden posible para cubrir el resto de minterms

Ej:  $F(x,y,z,u) = \sum m(2,3,4,6,8,12,14)$

$x\bar{y}$	00	01	11	10
$\bar{y}z$	00	01	11	10
00			1 1	
01	1			1
11	1			1
10	1			

$$F(x,y,z,u) = y\bar{u} + \bar{x}\bar{y}z + x\bar{z}\bar{u}$$

$$\text{Ej: } F(x,y,z,u) = \sum m(1, 3, 5, 7, 10, 11, 14, 15)$$

$x\bar{y}$	00	01	11	10
$\bar{y}u$	00	1 1	1	1
$\bar{x}y$	01	1 1	1	1
$xz$	11	1 1	1	1
$xy$	10		1 1	1

$$F(x,y,z,u) = \bar{x}u + xz$$

$$\text{Ej: } G(x,y,z,u) = \sum m(1, 5, 10, 12, 13, 14, 15)$$

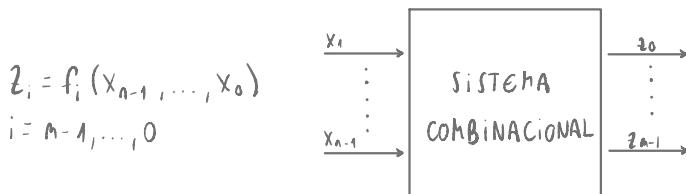
$x\bar{y}$	00	01	11	10
$\bar{y}u$	00	1 1		
$\bar{x}y$	01	1 1		
$xz$	11	1 1	1 1	1
$xy$	10			1

$$F(x,y,z,u) = xy + \bar{x}\bar{z}u + xz\bar{u}$$

# Diseño de circuitos combinacionales

## Sistema Combinacional

- Son sistemas digitales en los que en cualquier instante, los valores de sus señales dependen únicamente del de sus entradas (sin retraso)



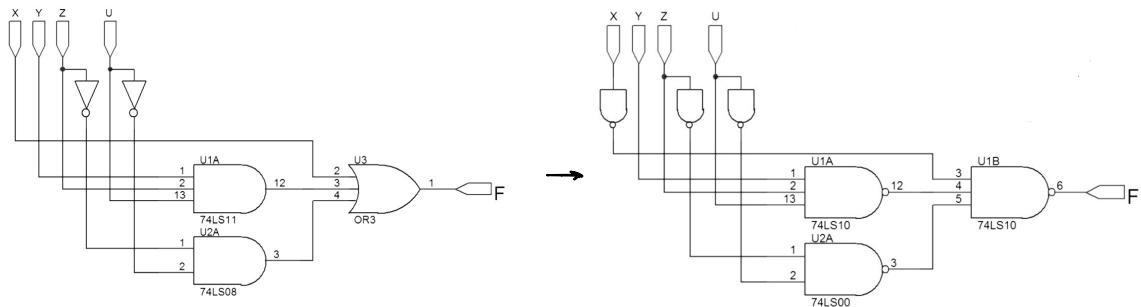
- Los señales de entrada y de salida se representan mediante variables lógicas
- Para  $n$  variables de entrada, hay  $2^n$  combinaciones de salida
- El comportamiento de un sistema combinacional se puede describir mediante  $n$  funciones lógicas (1 para cada señal de salida)
- Cada función lógica de salida se expresa en función de  $n$  variables de entrada
- Etapas de diseño
  1. Enunciar el problema
  2. Determinar el número de variables de entrada y de salida necesarias
  3. Asignar nombres a las variables de entrada y salida
  4. Formar las tablas de verdad
  5. Simplificar una de las funciones lógicas de salida
  6. Dibujar el diagrama lógico de las funciones simplificadas

## Síntesis NAND-NOR

- Cualquier circuito se puede implementar solo con NAND y NOR.
- Si tenemos AND, OR y NOT se puede sintetizar solo NAND o solo NOR siempre y cuando se cumpla:

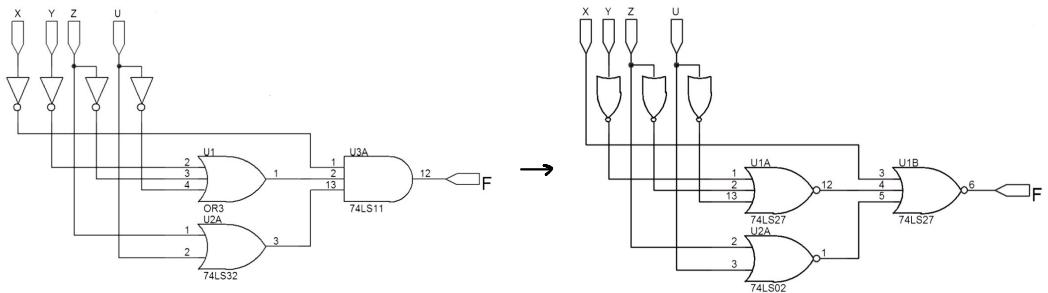
### ▫ Síntesis NAND

- Teorema NAND: Dado un circuito de dos niveles con una puerta OR y otra AND se puede conseguir un circuito que realice la misma función cambiandolas por puestas NAND.
  - Con un circuito donde el retraso un entrada es de nivel OR, para que llegue al primer nivel AND, se cumple si ponemos un AND extra, donde las entradas o sus complementadas se conecten a las dos entradas AND.



### ▫ Síntesis NOR

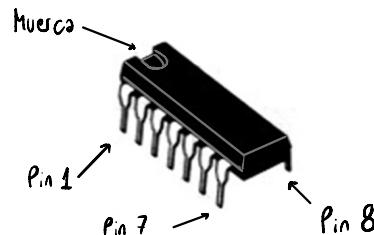
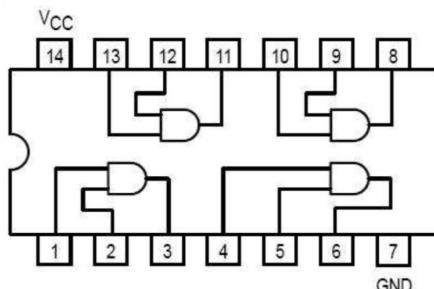
- Teorema NOR: Dado un circuito de dos niveles con una puerta OR y otra AND se puede conseguir un circuito que realice la misma función cambiandolas por puestas NOR.
  - Con un circuito donde el retraso un entrada es de nivel AND, para que llegue al primer nivel OR, se cumple si ponemos un OR extra, donde las entradas o sus complementadas se conecten a las dos entradas OR.



### Encapsulado de las puertas físicas

- Al fabricar las puertas se elige el tipo de encapsulado. Cuando se elige, se ven cuantas puertas van a poner en el chip.
- Si se decide una puerta AND de dos entradas se buscas un circuito que contenga AND.

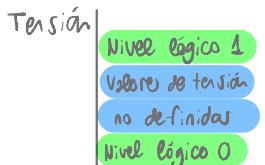
Ej: 74LS08



$$\overline{(\overline{X} + \overline{Y})} = \overline{\overline{X}} = X$$

## Tensión y corrientes de las puertas lógicas

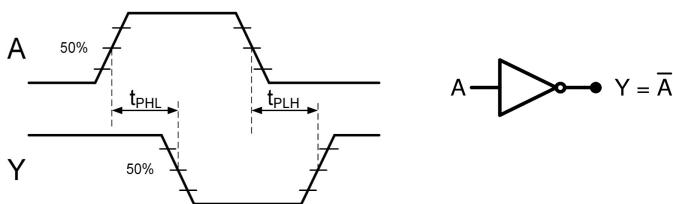
- Los niveles lógicos se relacionan a los rangos de tensión.  
Además varían según transistores para construcción de puertas



- Las salidas de las puertas no puede manejar corriente tan grande, el número de entradas que se pueden conectar a las salidas es limitado (fanout)

## Tiempo de propagación de las puertas lógicas

- Cuando cambia una entrada de una puerta, produce un cambio en las salidas con un retraso de propagación
  - Se generan dos retardos, uno al pasar de alto a bajo ( $t_{PHL}$ ) o de bajo a alto ( $t_{PLH}$ )



## Riesgos en circuitos combinacionales

- Riesgos: situaciones anómalas que se dan en los sistemas generando pequeñas pulsos espurias en los salidas al cambiar entradas
- Pulsos espurias: cambio de valor de corta duración en una señal que debería ser constante
- Debido al retardo de propagación, una señal se propaga por distintas caminos a una misma puerta produciendo pulsos espurias a su salida
- Tipos de riesgos según número de entradas
  - Riesgos lógicos
    - Solo cambia el valor de una entrada
  - Riesgos funcionales
    - Cambia el valor de más de una entrada
- Tipos de riesgos de como cambian las señales de salida
  - Riesgo estático
    - Si cambia el valor de entradas, las salidas permanece en valor  $V$  pero momentáneamente pasa a  $\bar{V}$ .
      - Si  $V=1$  es un riesgo estático si una
      - Si  $V=0$  es un riesgo estático a cero
  - Riesgo dinámico
    - Si al cambiar el valor de entradas las salidas deben cambiar pero momentáneamente vuelve a su valor inicial

## Riego lógico estatico al 1

- Se produce al simplificar a partir de los una de minterms, si hay dos minterms-1 que difieren el valor de una variable de entrada y se simplificar no quedan abiertas.
- Solución: añadir una adyacencia redundante que incluya los dos minterms adyacentes.

Ej:

	$x \cdot y^2$	00	01	11	10
0					
1		1	1	1	

→

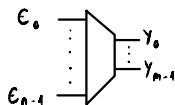
	$x \cdot y^2$	00	01	11	10
0					
1		1	1	1	1

$$F(x,y,z) = x\bar{y} + xz$$
$$F(x,y,z) = x\bar{y} + xz + yz$$

# Circuitos combinacionales lógicos

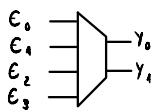
## Codificadores

- Es un bloque combinacional con  $n$  entradas ( $E_{n-1}, \dots, E_1$ ) y  $m$  salidas ( $Y_{m-1}, \dots, Y_0$ ) siendo  $n$  el nº de caracteres a codificar y  $m$  el nº de bits del código binario



### Codificadores sin prioridad

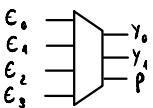
- En cada instante debe de haber solo 1 entrada activa.
- La salida representa esa única entrada



$E_3$	$E_2$	$E_1$	$E_0$	$Y_1$	$Y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

### Codificadores con prioridad

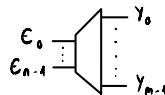
- Esquema de prioridad:  $E_3 (+) \rightarrow E_2 \rightarrow E_1 \rightarrow E_0 (-)$
- La salida P se activa si no hay ninguna entrada



$E_3$	$E_2$	$E_1$	$E_0$	$Y_1$	$Y_0$	$P$
1	-	-	-	1	1	0
0	1	-	-	1	0	0
0	0	1	-	0	1	0
0	0	0	1	0	0	0
0	0	0	0	0	0	1

## Decodificadores

- Realiza la función contraria al codificador
- Es un bloque combinatorial con  $n$  entradas ( $E_{n-1}, \dots, E_1$ ) y  $m$  salidas ( $Y_{m-1}, \dots, Y_0$ ) siendo  $n$  el nº de bits del código binario y  $m$  el nº de caracteres a codificar
- En cada instante se activa solo la salida correspondiente al carácter codificado en sus entradas
- Se indica el tamaño según el formato de  $n$  a  $m$

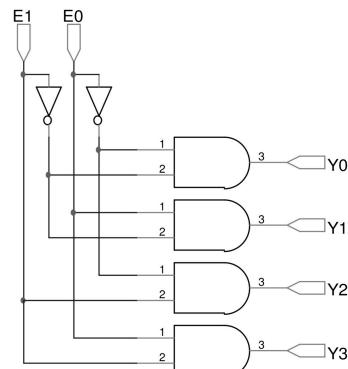


### Decodificadores binarios

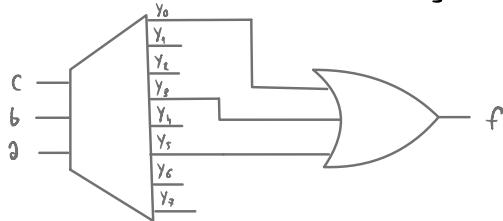
- Palabras del código binario natural
- $m = 2^n$ ,  $n \geq 2^k$
- Las salidas corresponden a los números decimales que representan  $Y_i$ , donde  $i$  es el número decimal
- Las salidas cuya subíndice coincide con el equivalente decimal del número binario que hay en sus entradas

Diagrama de un decodificador binario de 3 entradas (E<sub>2</sub>, E<sub>1</sub>, E<sub>0</sub>) y 4 salidas (Y<sub>3</sub>, Y<sub>2</sub>, Y<sub>1</sub>, Y<sub>0</sub>):

$E_2, E_1, E_0$	$Y_3$	$Y_2$	$Y_1$	$Y_0$
0 0	0	0	0	1
0 1	0	0	1	0
1 0	0	1	0	0
1 1	1	0	0	0

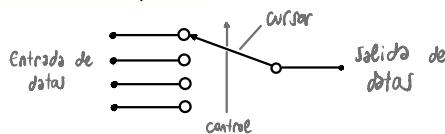


$$\text{Ej: } f(a, b, c) = \sum m(0, 3, 5) + d(2, 6) = m_0 + m_3 + m_5 = Y_0 + Y_3 + Y_5$$

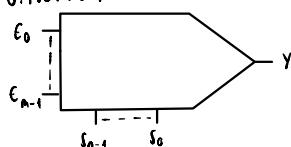


## Multiplexores

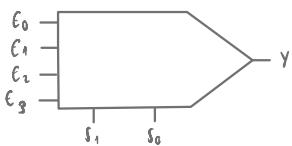
- Realiza una función de comutador de múltiples posiciones
- Varias entradas y una sola salida



- Un multiplexor (MUX) es un bloque combinacional con  $n$  entradas de control de selección ( $s_{n-1}, \dots, s_0$ ),  $m$  entradas de datos ( $e_0, \dots, e_{m-1}$ ) y una salida  $y$
- $m = 2^n$  (multiplexor binario)



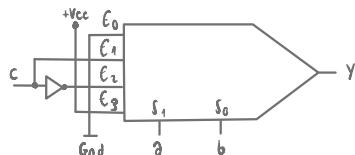
Ej: Multiplexor de 4 a 1



$s_1$	$s_0$	$y$
0	0	$e_0$
0	1	$e_1$
1	0	$e_2$
1	1	$e_3$

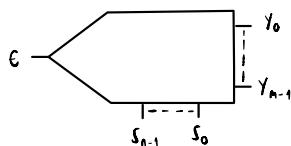
$$\text{Ej: } f(a, b, c) = \sum m(3, 4, 6, 7)$$

a	b	c	f	$\epsilon_i$
0	0	0	0	
0	0	1	0	$\epsilon_0 = 0$
0	1	0	0	
0	1	1	1	$\epsilon_1 = c$
1	0	0	1	
1	0	1	0	$\epsilon_2 = \bar{c}$
1	1	0	1	
1	1	1	1	$\epsilon_3 = 1$



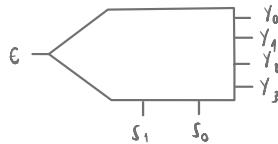
## De multiplexores

- Realiza la función contraria a un multiplexor
- Bloque combinacional con  $n$  entradas de control de selección ( $s_{n-1}, \dots, s_0$ ),  $m$  entradas de datos ( $y$ ) y  $m$  salidas ( $y_0, \dots, y_{m-1}$ )
- $M = 2^n$  (Demultiplexor binario)
- Un decodificador con entrada de habilitación funciona como multiplexor



Ej: Decodificador 2 a 4 con entrada de habilitación

$E$	$S_1$	$S_0$	$y_3$	$y_2$	$y_1$	$y_0$
$E_N$	$E_1$	$E_0$				
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	1	0
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



$S_1$	$S_0$	$y_3$	$y_2$	$y_1$	$y_0$
0	0	0	0	0	$E$
0	1	0	0	$E$	0
1	0	0	$E$	0	0
1	1	$E$	0	0	0

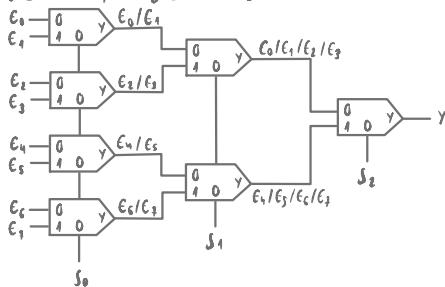
Ampliación de tamaño MUX multiplexores del mismo tamaño

- Si  $n_1$  es el número de entradas de control de selección de los MUX que se van a usar y  $n_2$  los que se van a implementar:
- $p = n_2/n_1$  (número entero)

Ej: MUX de 8 a 1 mediante MUX de 2 a 1

$$\begin{aligned} 2 \text{ a } 1 &= 2^1 \text{ a } 1 \rightarrow n = 1 \\ 8 \text{ a } 1 &= 2^3 \text{ a } 1 \rightarrow n = 3 \end{aligned} \left\{ p = \frac{n_2}{n_1} = \frac{3}{1} = 3 \right.$$

3 niveles de MUX de 2 a 1



$S_2$	$S_1$	$S_0$	$y$
0	0	0	$s_0$
0	0	1	$s_1$
0	1	0	$s_2$
0	1	1	$s_3$
1	0	0	$s_4$
1	0	1	$s_5$
1	1	0	$s_6$
1	1	1	$s_7$

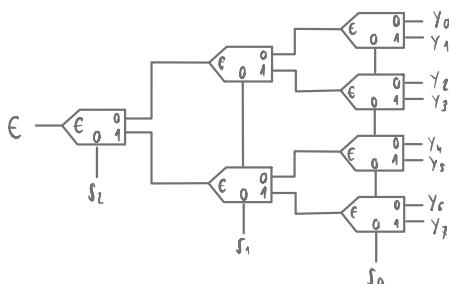
## Ampliación de tamaño de multiplexores del mismo tamaño

- Si  $n_2$  es el número de entradas de control de selección de los demux que se van a usar y  $n_1$  las que se van a implementar:
- $p = n_2/n_1$  (número entero)

Ej: Demux de 1 a 8 mediante demux de 1 a 2

$$\left. \begin{array}{l} 1 \otimes 2 = 1 \otimes 2^1 \rightarrow n=1 \\ 1 \otimes 8 = 1 \otimes 2^3 \rightarrow n=3 \end{array} \right\} p = n_2/n_1 = 3/1 = 3$$

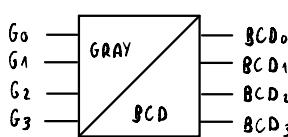
3 niveles de demux de 1 a 2



	$S_2$	$S_1$	$S_0$
$y_0$	0	0	0
$y_1$	0	0	1
$y_2$	0	1	0
$y_3$	0	1	1
$y_4$	1	0	0
$y_5$	1	0	1
$y_6$	1	1	0
$y_7$	1	1	1

## Conversores de código

- Traduce una información representada en un código a otro



	$G_3$	$G_2$	$G_1$	$G_0$	$BCD_3$	$BCD_2$	$BCD_1$	$BCD_0$	Decimal
	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	0	0	1	1
	0	0	1	1	0	0	1	0	2
	0	0	1	0	0	0	1	1	3
	0	1	1	0	0	1	0	0	4
	0	1	1	1	0	1	0	1	5
	0	1	0	1	0	1	1	0	6
	0	1	0	0	0	1	1	1	7
	1	1	0	0	1	0	0	0	8
	1	1	0	1	1	0	0	1	9

$G_1 G_0$	00	01	11	10
$G_2 G_1$	00			
01				
11	1	1	-	-
10	-	-	-	-

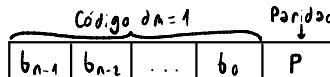
$$BCD_3 = G_3$$

$G_1 G_0$	00	01	11	10
$G_2 G_1$	00	1	1	1
01	1		1	
11	1	-	-	-
10	-	-	-	-

$$BCD_0 = G_3 G_0 + G_2 G_1 G_0 + \bar{G}_1 \bar{G}_1 G_0 + \bar{G}_2 \bar{G}_1 G_0 + \bar{G}_3 G_2 \bar{G}_1 \bar{G}_0$$

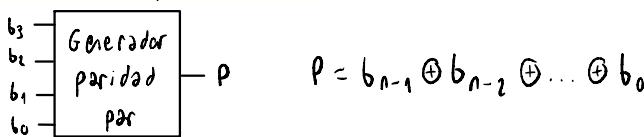
### Generadores y comprobadores de paridad

- Los códigos de paridad se forman a partir de un código de distancia mínima, más un bit adicional (bit de paridad)
- Existen dos tipos de paridad (Par e impar)



## Generador de paridad par

- Las entradas son bits del código de distancia mínima:  $b_{n-1}, b_{n-2}, \dots, b_0$  y de salida el bit de paridad par  $P$
- La salida  $P$  indica el bit de paridad par correspondiente al valor de los bits de entrada
- El bit par  $P$  vale 1 para que el número total de bits del conjunto sea par. Si lo es vale 0
- Utiliza una función XOR

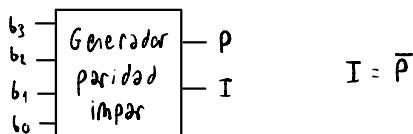


## Comprobador de paridad par

- Se genera la paridad de todos los bits del código de paridad
  - Si  $C=1$ . Paridad incorrecta
  - Si  $C=0$ . Paridad correcta

## Generador de paridad impar

- El bit impar  $I$  vale 1 para que el número total de bits del conjunto sea impar. Si lo es vale 0
- Es complementaria de la paridad par



## Comprobador de paridad impar

- Consiste en un generador de paridad impar
- Comprobador de paridad par complementado

# Aritmética básica

Operaciones aritméticas con números binarios enteros sin signo

- Suma de dos bits

- La suma de dos dígitos binarios ( $x_i, y_i$ ) genera un bit suma ( $s_i$ ) y otro de acarreo ( $c_{i+1}$ )
- Si la suma es mayor de 1 produce 2 bits

$x_i + y_i + z_i$	$c_{i+1}$	$s_i$
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

- Suma de dos números binarios de  $n$  bits

- Empezando por los bits de menor peso se suman los de igual peso junto con el acarreo de la suma anterior
- La suma puede generar acarreo final
  - El resultado tiene más bits que los sumandos
  - No se puede representar con los  $n$  bits del sistema
  - Se ha producido desbordamiento o overflow

Ej:

$$\begin{array}{r} & \begin{matrix} & 1 & 1 & c_i \\ 1 & 0 & 1 & 1 & X_i \\ + & 0 & 0 & 1 & 1 & Y_i \\ \hline 1 & 1 & 1 & 0 & S_i \\ 0 & 0 & 1 & 1 & C_{i+1} \end{matrix} \\ & + \begin{matrix} & 1 & 1 \\ & 3 \\ \hline 1 & 4 \end{matrix} \end{array}$$

- Resta de dos dígitos binarios

- Genera dos bits de resultado, la resta ( $R_i$ ) y el préstamo
  - o borrow ( $B_{i+1}$ )
- Se opera  $X_i - (Y_i + B_i)$
- Se produce préstamo ( $Y_i + B_i > X_i$ )
- Si sumamos ( $Y_i + B_i$ ) se produce acarreo, habrá préstamo

$X_i - Y_i - B_i$	$B_{i+1}$	$R_i$
0 0 0	0	0
0 0 1	1	1
0 1 0	1	1
0 1 1	1	0
1 0 0	0	1
1 0 1	0	0
1 1 0	0	0
1 1 1	1	1

- Se comprueba  $X \geq Y$ , si se cumple se resta  $X - Y$  si no  $-(Y - X)$ . No habrá desbordamiento

$$\begin{array}{cccccc}
 & & & & B_i & \\
 & 1 & 1 & 1 & \swarrow & \\
 & 1 & 1 & 0 & 0 & 1 & X_i \\
 - & \underline{0} & 1 & 1 & 1 & 0 & Y_i \\
 & 0 & 1 & 0 & 1 & 1 & R_i \\
 & 0 & 1 & 1 & 1 & 0 & B_{i+1}
 \end{array}
 \quad
 \begin{array}{r}
 2 \quad 5 \\
 - \quad \underline{1} \quad 4 \\
 1 \quad 1
 \end{array}$$

- Multiplicación de dos números binarios
    - Coincide con la multiplicación decimal
    - Equivale a AND

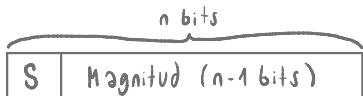
$x_i$	$y_i$	$p_i$
0	0	0
0	1	0
1	0	0
1	1	1

- División de dos números binarios
    - Coincide con la división decimal
    - La división entre 0 es indeterminada

$$\begin{array}{r}
 \text{fj:} \quad 101001 \quad | 110 \\
 -110 \\
 \hline
 010001 \\
 -110 \\
 \hline
 00\underbrace{101}
 \end{array}
 \qquad
 \begin{array}{r}
 41 \quad | \quad 6 \\
 \underline{5} \quad 6
 \end{array}$$

## Sistema de representación signo-magnitud

- Modificación del binario. Se añade un bit de signo



- Magnitud: Indica el binario natural
- S es el signo
  - 0 positivo
  - 1 negativo

Ej: 

0	0	1	1	0	0
---	---	---	---	---	---

 + 12

1	0	1	1	0	0
---	---	---	---	---	---

 - 12

Ej: 

0	0	0	0	0	0
---	---	---	---	---	---

 + 0

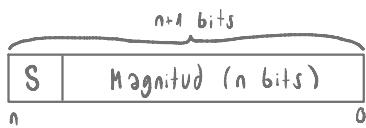
1	0	0	0	0	0
---	---	---	---	---	---

 - 0

### Operaciones de suma y resta

- En suma, si ambos operandos tienen el mismo signo el resultado tambien. Si tienen distintas se resta el mayor al menor y manteniendo el signo mayor
- En resta, si tienen el mismo signo se resta el mayor al menor. Si tienen distintas signos se suman dejando el signo del minuendo

## Representación en complemento a 2



- S determina el signo
  - 0 positivo
  - 1 negativo
- Si A es positivo, S=0 y en el resto de bits se pone A en binario
- Si A es negativo, S=1 y en el resto de bits se pone  $2^n - A$ 
  - $-A = 2^n - A$

Ej:  $-12 = 12_{C2} = 2^6 - 12 = 32 - 12 = 20$

0	0	1	1	0	0

1	1	0	1	0	0

- Suma binaria
  - De dos números positivos

Ej:

0 1 0 1 0	1 0
+ 0 0 1 0 1	+ 5
0 0 1 1 1 1	1 5

Cf      S=0

## 0 Das números negativos

$\epsilon_j$ :

$$\begin{array}{r}
 & 1 & 1 & 1 \\
 & 1 & 0 & 1 & 1 & 0 \\
 + & 1 & 1 & 0 & 1 & 1 \\
 \hline
 & 1 & 1 & 0 & 0 & 1
 \end{array}
 \quad
 \begin{array}{r}
 - 10 \\
 + - 05 \\
 \hline
 - 15
 \end{array}$$

#### □ Los números positivos (con desbordamiento)

$\epsilon_j$ :

$$\begin{array}{r}
 & 1 \\
 & \text{---} \\
 \begin{matrix} 0 & 1 & 0 & 1 & 0 \\ + & 0 & 1 & 1 & 0 & 1 \end{matrix} & \xrightarrow{\quad\text{Cf}\quad} & 1 & 0 \\
 & \underline{\text{---}} & & + & \underline{1 & 3} \\
 & 0 & 1 & 0 & 1 & 1 & & 2 & 3 \\
 & \swarrow & \searrow & & & & & & \\
 \text{Cf} & \text{S negative} & & & & & & & 
 \end{array}$$

#### ▪ Los números negativos (con desbordamiento)

$e_j$ :

$$\begin{array}{r}
 \text{Cj:} \\
 \begin{array}{r}
 \begin{array}{r}
 1 & 0 & 1 & 1 & 0 \\
 + & 1 & 0 & 0 & 1 & 1 \\
 \hline
 1 & 0 & 1 & 0 & 0 & 1
 \end{array} \\
 \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
 \text{Cf} \quad \text{S positive}
 \end{array}
 \end{array}
 \quad
 \begin{array}{r}
 -10 \\
 + -13 \\
 \hline
 -23
 \end{array}$$

□ Los números de distinto signo

- Si se produce escaneo final se descarta

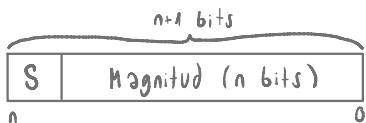
- No pude tener desordenamiento

6

$$\begin{array}{r}
 & & 1 & \\
 & 0 & 1 & 0 & 1 & 0 \\
 + & 1 & 1 & 0 & 1 & 1 \\
 \hline
 & 1 & 0 & 0 & 1 & 0 & 1 \\
 \end{array}
 \quad
 \begin{array}{r}
 + 10 \\
 + - 05 \\
 \hline
 5
 \end{array}$$

Cf S paritivo

## Representación en complemento a 1



- **S determina el signo**
  - 0 positivo
  - 1 negativo
- Si A es positivo, S = 0 y en el resto de bits se pone A en binario
- Si A es negativo, S = 1 y en el resto de bits se pone  $2^n - 1 - A$ 
  - $-A = 2^n - 1 - A$
  - $C_2 = C_1 + 1$

Ej: 

0	0	1	1	0	0
---	---	---	---	---	---

 + 12

1	1	0	0	1	0
---	---	---	---	---	---

 - 12

Ej: 

0	0	0	0	0	0
---	---	---	---	---	---

 + 0

1	1	1	1	1	1
---	---	---	---	---	---

 - 0

Ej: 

1	1	0	0	1	0	- 13
+ 0	1	0	1	0	1	+ 21
<hr/>						$8-1 = 7$

$\downarrow$  acarreo

Ej: Demostración de 0 en C2

0	0	0	0	0	0
---	---	---	---	---	---

Suma de dos números BCD 8421

- Código binario ponderado de 4 bits
- Usa solo combinaciones del 0 (0000) al 9 (1001)
- Si el resultado es mayor a 9 hay que sumarle 6

Ej:

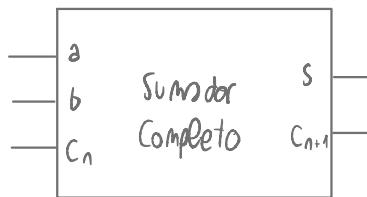
$$\begin{array}{r} 1001 & 9 \\ + 1001 & + \frac{9}{18} \\ \hline 10010 & \longrightarrow \\ + 00110 & \\ \hline 11000 & = \text{resultado corregido } (+6) \end{array}$$

# Circuitos combinacionales aritméticos

## Sumador completo

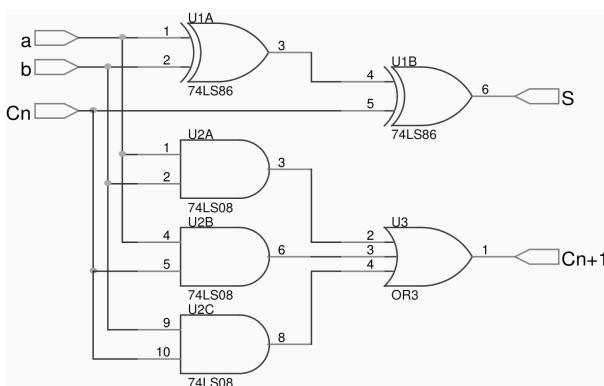
- Suma dos números binarios de 1 bit ( $a, b$ ) y un acarreo ( $C_n$ ) (3 bits)
- Dos salidas ( $S, C_{n+1}$ ), suma y acarreo

$a$	$b$	$C_n$	$C_{n+1}$	$S$
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



$a$	$b$	$C_n$	00	01	11	10
0			0	1		1
1			1	1	1	1

$a$	$b$	$C_n$	00	01	11	10
0			0	1		1
1			1	1	1	1



## Sumador binario de $n$ bits

### • Entradas

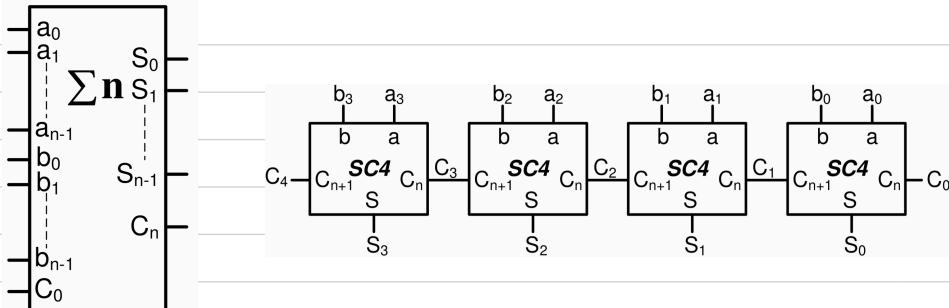
- $n$  entradas para el primer operando  $A (a_{n-1}, \dots, a_0)$
- $n$  entradas para el segundo operando  $B (b_{n-1}, \dots, b_0)$
- Acarreo de entrada ( $C_0$ )

### • Salidas

- $n$  salidas de resultado  $S (S_{n-1}, \dots, S_0)$
- Acarreo de salida ( $C_n$ )

### • Implementación

- $n$  sumadores completas
- Generando una función lógica en el plano AND-OR
- Mediante técnicas se pueden generar acarreos rápidamente  
(Solución intermedia a las anteriores)



- Circuitos S/R mediante  $C_2$  para números binarios sin signo

