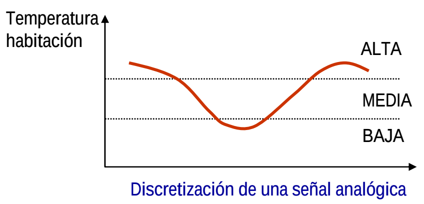
LECCIÓN 11
SISTEMAS DIGITALES

Introducción

- Una señal digital puede tomar un valor entre un número finito (dos)
- de valores o estados. Una señal analógica puede tomar un valor entre un número infinito de valores.



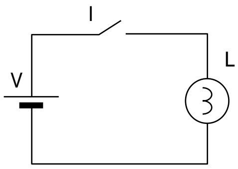
Discretización de una señal analógica

- Mundo real es un mundo analógico. ¿Por qué usar SISTEMAS DIGITALES? Las razones son:
  - **Capacidad para manejar gran cantidad de información**
    - **Fácil de transmitir**
    - **Muy inmune al ruido**
  - **Gran desarrollo de la tecnología: CI (integran millones de transistores)**
  - **Microprocesadores: Gran capacidad de cálculo**

Estados lógicos

Se consideran variables que únicamente tienen dos posibles estados (valores BINARIOS)

Ejemplo: En el circuito existen **2 variables binarias (INTERRUPTOR y LÁMPARA)** que tiene cada una **2** posibles **estados lógicos.**



I	L
ABIERTO	APAGADA
CERRADO	ENCENDIDA

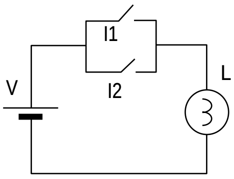
En vez de usar términos para los estados, podemos usar símbolos como:

CERRADO = '1'
ABIERTO = '0'

TABLA DE VERDAD: Tabla en la que se relacionan las variables del sistema y sus posibles estados

I	L
0	0
1	1

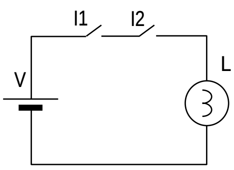
Operadores lógicos



I1	I2	L
0	0	0
0	1	1
1	0	1
1	1	1

Operador lógico OR

L = I1 **OR** I2
L = I1 + I2

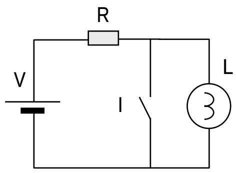


I1	I2	L
0	0	0
0	1	0
1	0	0
1	1	1

Operador lógico AND

L = I1 **AND** I2
L = I1• I2

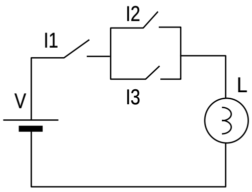
Operadores lógicos



I	L
0	1
1	0

Operador lógico NOT

L = **NOT** I



I1	I2	I3	L
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Combinación de operadores lógicos

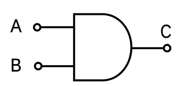
L = I1 **AND** (I2 **OR** I3)
L = I1·(I2+I3)

Puertas lógicas

- Una puerta lógica es un elemento que tiene varias entradas binarias (variables) y una salida, cuyo estado lógico depende del estado de las entradas y del operador lógico que represente.

PUERTAS LÓGICAS BÁSICAS

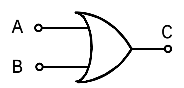
Puerta AND



C = A · B

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

Puerta OR



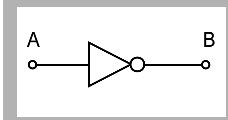
C = A + B

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

Puertas lógicas

PUERTAS LÓGICAS BÁSICAS

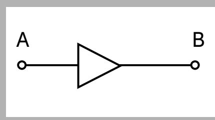
Puerta NOT (inversor)



A	B
0	1
1	0

$B = \overline{A}$

BUFFER



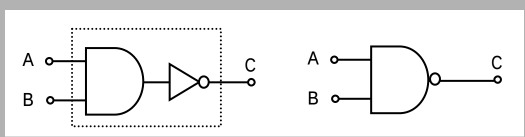
A	B
0	0
1	1

$B = A$

Cambia propiedades ELÉCTRICAS pero NO LÓGICAS
“Refuerza” la energía de la señal lógica (información)

Puertas lógicas combinadas

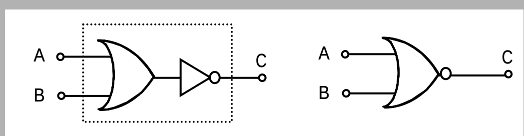
Puerta NAND



A	B	AB	C
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

$C = \overline{A \cdot B}$

Puerta NOR

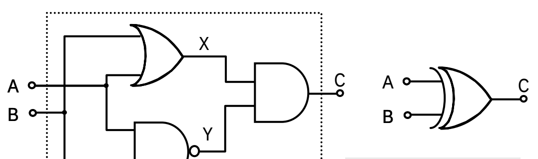


A	B	A+B	C
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

$C = \overline{A + B}$

Puertas lógicas combinadas

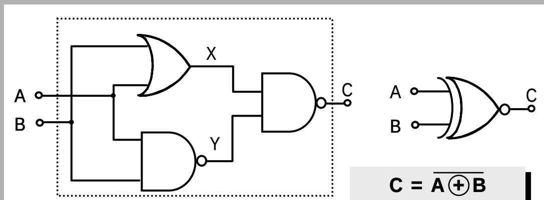
Puerta OR EXCLUSIVO



$C = A \oplus B$

A	B	X	Y	C
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

Puerta NOR EXCLUSIVO



$C = \overline{A \oplus B}$

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

PUERTA DE IGUALDAD

Algebra de Boole. Mapas de Karnaugh

Funciones Booleanas y teoremas

Formas canónicas

Ley de De Morgan: Simplificaciones algebraicas

Mapas de Karnaugh

Condiciones indiferentes

Álgebra de Boole (George Boole, siglo XIX)

- El Álgebra de Boole proporciona una notación para describir funciones lógicas y define:
  - **CONSTANTES**, **VARIABLES** y **FUNCIONES** para describir sistemas binarios.
  - **TEOREMAS** para manipular expresiones lógicas

HERRAMIENTA PARA SIMPLIFICAR FUNCIONES LÓGICAS Y DISEÑAR CIRCUITOS LÓGICOS

Constantes Booleanas	‘0’ ⇨ FALSO ‘1’ ⇨ VERDADERO
Variables Booleanas	A, B, C, I1, I2, I3, L Magnitudes que pueden tomar 2 estados posibles: ‘0’ ó ‘1’

Funciones Booleanas

FUNCIÓN	SÍMBOLO	EXPRESIÓN BOOLEANA
AND	$\cdot$	$C=A\cdot B$
OR	$+$	$C=A+B$
NOT	$\overline{}$	$A=\overline{B}$
NAND		$C=\overline{A\cdot B}$
NOR		$C=\overline{A+B}$
OR exclusivo	$\oplus$	$C=A\oplus B=(A+B)\cdot\overline{(A\cdot B)}$
NOR exclusivo	$\odot$	$C=\overline{A\oplus B}=A\odot B$

UPM DIE

Teoremas (I): Conjunto de Identidades

OPERADORES	AND	OR	NOT
	$0 \cdot 0 = 0$	$0 + 0 = 0$	$\overline{0} = 1$
	$0 \cdot 1 = 0$	$0 + 1 = 1$	$\overline{1} = 0$
	$1 \cdot 0 = 0$	$1 + 0 = 1$	$\overline{\overline{A}} = A$
	$1 \cdot 1 = 1$	$1 + 1 = 1$	
	$A \cdot 0 = 0$	$A + 0 = A$	
	$0 \cdot A = 0$	$0 + A = A$	
	$A \cdot 1 = A$	$A + 1 = 1$	
	$1 \cdot A = A$	$1 + A = 1$	
	$A \cdot A = A$	$A + A = A$	IDEMPOTENCIA
	$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$	

UPM DIE

Teoremas (II): Conjunto de Leyes

Ley conmutativa

$A \cdot B = B \cdot A$

$A + B = B + A$

Ley distributiva

$A (B + C) = AB + AC$

$A + BC = (A + B)(A+C)$

Ley asociativa

$A (BC) = (AB) C$

$A + (B + C) = (A + B) + C$

¡¡¡$(AB) + C \neq A (B + C)$!!!

Ley de absorción

$A + AB = A$

$A (A+B) = A$

Ley de De Morgan

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

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

Ley de De Morgan

- Ley de De Morgan generalizada: El complemento de una función lógica se obtiene complementando todas las variables que intervienen en la función e intercambiando las operaciones lógicas.

Ley de De Morgan

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

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

$$\overline{\overline{S}} = S = \overline{\overline{(A+B+C)[\overline{A}(\overline{B}+\overline{C})]}} = \overline{\overline{(A+B+C)}+\overline{[\overline{A}(\overline{B}+\overline{C})]}} =$$

$$= \overline{(\overline{A}\,\overline{B}\,\overline{C})+[A+\overline{(\overline{B}+\overline{C})}]} = \overline{(\overline{A}\,\overline{B}\,\overline{C})+(A+BC)}$$

Formas canónicas

Toda expresión lógica puede descomponerse:

1	f (A, B, C,...) = A f (1, B, C, ...) + $\overline{A}$ f (0, B, C, ...)
2	f (A, B, C,...) = (A+ f(0, B, C,...))($\overline{A}$+ f(1, B, C,...))

Toda expresión lógica puede representarse por una forma canónica:

1ª forma canónica	Suma de productos fundamentales en los que intervienen **todas y cada una** de las variables de entrada f (AB) = AB f (1,1) + $\overline{A}$B f (0,1) + A$\overline{B}$ f (1,0) + $\overline{A}\,\overline{B}$ f (0,0)
2ª forma canónica	Producto de sumas fundamentales en las que intervienen **todas y cada una** de las variables de entrada f (AB) = (A+ B+ f(0,0)) · ($\overline{A}$+ B+ f(1,0)) · (A+ $\overline{B}$+ f(0,1)) ·($\overline{A}$+ $\overline{B}$+ f(1,1))

Expresión Booleana

➢ Extracción de la expresión booleana a partir de una tabla de verdad:

A	B	C	D	
0	0	0	0	
0	0	1	1	$\overline{A}\,\overline{B}C$
0	1	0	0	
0	1	1	0	
1	0	0	1	$A\overline{B}\,\overline{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

$$D = \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + ABC$$

PRODUCTO FUNDAMENTAL (Minterms)

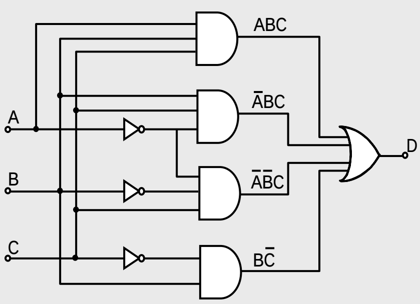
1 Se forma un producto fundamental (minterms) en cada fila de la tabla de verdad en que aparezca un '1' en la columna de la salida.

2 El producto fundamental (minterms) contiene todas y cada una de las variables de entrada. Cada variable aparece de la siguiente forma:
- ***Normal***: si aparece un '1' en la tabla
- ***Complementada***: si aparece un '0'

3 La expresión global para la función lógica es la suma de minterms

Simplificaciones algebraicas

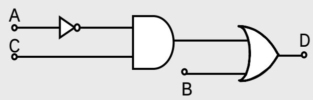
$$D = \overline{A}\,\overline{B}C + B\overline{C} + \overline{A}BC + ABC$$



$$D = \overline{A}\,\overline{B}C + B\overline{C} + \overline{A}BC + ABC = \quad (1)$$
$$= \overline{A}\,\overline{B}C + B\overline{C} + BC(\overline{A} + A) =$$
$$= \overline{A}\,\overline{B}C + B\overline{C} + BC = \quad (1)$$
$$= \overline{A}\,\overline{B}C + B(\overline{C} + C) = \overline{A}\,\overline{B}C + B = \quad (2)$$
$$= (\overline{A}C + B)(\overline{B} + B) = \overline{A}C + B$$

(1) Asociativa

(2) Distributiva



Una expresión lógica se puede simplificar haciendo uso de los teoremas del álgebra de Boole

UPM DIE

Mapas de Karnaugh

Método gráfico de representar la información que contiene la Tabla de Verdad. Se usan para simplificar una expresión de forma sistemática

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

C: B \ A	0	1
0	0	1
1	0	0

A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

D: C \ AB	00	01	11	10
0	0	1	0	0
1	0	1	1	1

Sólo puede variar un dígito entre dos casillas adyacentes

UPM DIE

Simplificación con mapas de Karnaugh

$F = \overline{A}B\overline{C}D + AB\overline{C}D = B\overline{C}D\,(\overline{A} + A) = B\overline{C}D$

F: CD \ AB	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	0	0	0
10	0	0	0	0

$F = B\overline{C}D$

$$
\begin{aligned}
F &= \overline{A}B\overline{C}D + AB\overline{C}D + \overline{A}BCD + ABCD \\
&= B\overline{C}D\,(\overline{A} + A) + BCD\,(\overline{A} + A) \\
&= BD\,(\overline{C} + C) = BD
\end{aligned}
$$

F: CD \ AB	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

$F = BD$

Minimización con mapas de Karnaugh

REGLAS

1ª Construir celdas (rectangulares o cuadradas) con el mayor número posible de '1s' adyacentes, siempre y cuando la celda contenga 2^n '1s'

2ª Añadir celdas progresivamente con menor número de '1s'

3ª Cualquier grupo redundante debe eliminarse

F: CD \ AB	00	01	11	10
00		1	1	
01			1	1
11			1	1
10	1			

$F = \boxed{\bar{A}\bar{B}C\bar{D}} + \boxed{B\bar{C}\bar{D}} + \boxed{AD}$

Minimización con mapas de Karnaugh

Posibles asociaciones | El diagrama es esférico

$F = \bar{C}\bar{D} + AC$

F: CD \ AB	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	1	1
10	0	0	1	1

$F = AB + \bar{B}\bar{D}$

F: CD \ AB	00	01	11	10
00	1	0	1	1
01	0	0	1	0
11	0	0	1	0
10	1	0	1	1

$F = D$

F: CD \ AB	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

$F = \bar{B}$

F: CD \ AB	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

Mapas de Karnaugh: Ejemplo

$$F = \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D$$
$$+ \bar{A}B\bar{C}D + AB\bar{C}D + A\bar{B}\bar{C}D + \bar{A}BCD +$$
$$ABCD + \bar{A}BC\bar{D} + ABC\bar{D}$$

F: CD \ AB	00	01	11	10
00	0	1	1	1
01	1	1	1	1
11	0	1	1	0
10	0	1	1	0

Forma mínima

$$F = B + \bar{C}D + A\bar{C}$$

Sólo puertas NAND

Aplicando De Morgan a forma mínima:

$$F = \overline{\overline{B + \bar{C}D + A\bar{C}}} = \overline{\bar{B} \cdot \overline{\bar{C}D} \cdot \overline{A\bar{C}}}$$

Condiciones indiferentes

CONDICIÓN INDIFERENTE DE ENTRADA

- La salida será la misma con un '1' o un '0' en la entrada
- Se representa por una X

A	B	C	D
0	0	0	0
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

A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	X	X	1

Condiciones indiferentes

CONDICIÓN INDIFERENTE DE SALIDA: La variable de salida, para una determinada combinación de entrada, es indiferente

Si en nuestro sistema nunca se va a dar una determinada combinación de entrada, quizá se pueda aprovechar para simplificar el diagrama de Karnaugh

Si estas combinaciones nunca ocurren, la salida D es indiferente ⇒ D = X

A	B	C	D
0	0	0	0
0	0	1	X
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	X
1	1	0	X
1	1	1	1

C \ AB	00	01	11	10
0	0	1	X	1
1	X	0	1	X

X = 0

X = 1

$D = A + B\overline{C}$

Grupos redundantes

F: CD \ AB	00	01	11	10
00	1		1	1
01	1		1	1
11	1			
10	1			

F: CD \ AB	00	01	11	10
00	1	1		
01	1	1		
11	1	1	1	1
10				

F: CD \ AB	00	01	11	10
00		1	1	
01			1	1
11			1	1
10				

F: CD \ AB	00	01	11	10
00	1			
01	1	1	1	1
11			1	
10			1	1

F: CD \ AB	00	01	11	10
00		1		
01		1	1	1
11	1	1	1	
10			1	

Karnaugh. Mínima expresión

Suma de productos

A	B	C	D
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

C \ AB	00	01	11	10
0	0	0	1	1
1	1	1	0	1

$D = \overline{A}C + A\overline{C} + A\overline{B}$

Mínima expresión como suma de productos

Producto de sumas

$D = (A + C)\,(\overline{A} + \overline{B} + \overline{C})$

Mínima expresión como producto de sumas

Se agrupan los '0' en vez de los '1'

Sistemas de numeración y Códigos

Sistemas de numeración. Aritmética binaria

Códigos binarios

Códigos BCD

Códigos progresivos

Detección de errores

UPM DIE

Sistemas de numeración

$$1327 = 1 \times 10^3 + 3 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$$

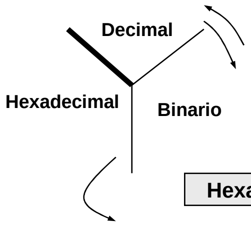
$$N = p_{n-1} \cdot b^{n-1} + p_{n-2} \cdot b^{n-2} + \ldots + p_1 \cdot b^1 + p_0 \cdot b^0$$

b = 10	Sistema decimal	Dígitos	0, 1, 2, ..., 9
b = 2	Sistema binario	Dígitos	0, 1 BIT
b = 16	Sistema hexadecimal	Dígitos	0, 1, ..., 9, A, ... F
b = 8	Sistema octal	Dígitos	0, ..., 7

UPM DIE

Cambio de sistema de numeración

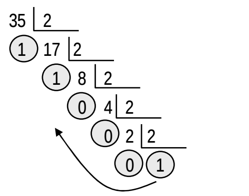
Cambios de base



Binario a Decimal (Suma de potencias):

$1101_b = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13_d$

Decimal a Binario (divisiones sucesivas por dos):



$100011_b = 35_d$

Hexadecimal a binario

$C3A5_h = \underbrace{1100}_{C}\ \underbrace{0011}_{3}\ \underbrace{1010}_{A}\ \underbrace{0101}_{5}{}_b$

Hexadecimal a decimal

$C3A5_h = C \cdot 16^3 + 3 \cdot 16^2 + A \cdot 16^1 + 5 \cdot 16^0 = 50085_d$

Aritmética Binaria

SUMA BINARIA

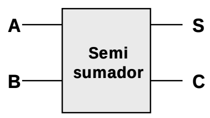
0 + 0 = 0

0 + 1 = 1

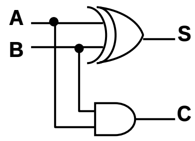
1 + 0 = 1

1 + 1 = 10 (acarreo 1)

Semisumador binario



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



¿Cómo sumar números binarios de 4 bits, 8 bits,...?
Se necesita sumar el acarreo

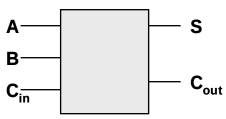
UPM DIE

Aritmética Binaria

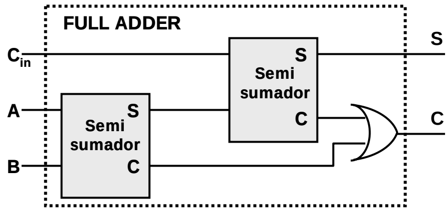
SUMA DE DOS NÚMEROS BINARIOS

C		1	0	1	1	0	
		1	0	1	1	0	22_d
+		1	0	0	1	1	19_d
	1	0	1	0	0	1	41_d

Sumador binario

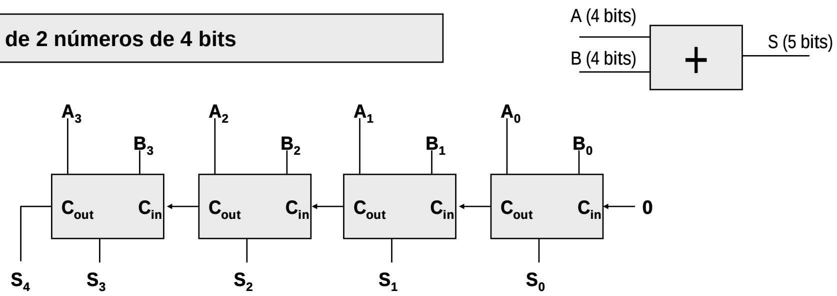


A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Sumador Serie

Suma de 2 números de 4 bits



Características

- **Número de puertas bajo**
- **Retardo proporcional al número de bits**

Códigos binarios

Números Naturales

BINARIO NATURAL — **3 → 011**

Números Enteros

POSITIVOS

Bit de signo + magnitud | **SMMM** | **S=0 → positivo** | **+3 → 0011** (Bit de signo)

NEGATIVOS

N1. Bit de signo +magnitud | **SMMM** | **S=1 → negativo** | **-3 → 1011** (Bit de signo)

N2. Complemento a 1
- **1º) Cambiar el bit de signo**
- **2º) Intercambiar 0s y 1s**

-3 → 1011
0011
C1 de -3 → 1100

N3. Complemento a 2
- **1º) Complementar a 1**
- **2º) Sumar 1**

C1 de -3 → 1100
+1
C2 de -3 → 1101

Representaciones de números binarios negativos utilizando 4 bits

Decimal	Binario signo y magnitud	Binario compl. a 1	Binario compl. a 2
-7	1 111	1000	1001
-6	1 110	1001	1010
-5	1 101	1010	1011
-4	1 100	1011	1100
-3	1 011	1100	1101
-2	1 010	1101	1110
-1	1 001	1110	1111

Usando complemento a 2, las restas se convierten en sumas.

- 4	1100
+2	0010
- 2	1110

- 6	1010
+7	0111
+1	$\boxed{1}$0001

Se elimina el posible acarreo

Si el resultado de la resta es negativo, éste viene dado en complemento a 2.

Código BCD (Binary Code Decimal)

El código BCD representa números decimales codificados en binario dígito a dígito

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

BCD Natural

Ejemplo: BCD natural

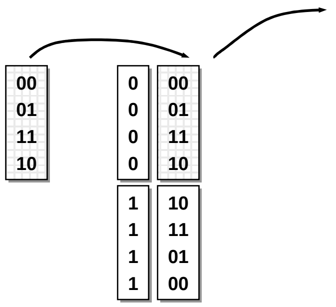
37_d = 0011 0111

Conversión BCD a decimal inmediata

Códigos progresivos

Un código es progresivo si entre dos combinaciones adyacentes hay una diferencia de un solo bit. Si las combinaciones primera y última son progresivas, se dice que el código, además, es cíclico.

El código Gray



Autocomplementarios

Decimal	Gray
0	0 000
1	0 001
2	0 011
3	0 010
4	0 110
5	0 111
6	0 101
7	0 100
8	1 100
9	1 101
10	1 111
11	1 110
12	1 010
13	1 011
14	1 001
15	1 000

Eje de simetría

Detección de errores

Códigos de detección

Paridad par o impar

Dato	Impar	Par
0000	1	0
0001	0	1
0010	0	1
0011	1	0
0100	0	1
0101	1	0
0110	1	0
0111	0	1
1000	0	1
1001	1	0

Se añade un bit para que la cantidad de “1s” sea par (o impar)

$\boxed{1}$ 0 0 1 0 0 1 1
$\boxed{0}$ 1 0 1 0 0 1 1
Bit de paridad (par)

Lógica Combinacional

Ejemplos de Lógica Combinacional

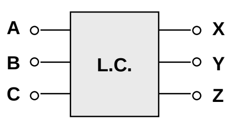
Lógica combinacional

- La salida viene determinada por la COMBINACIÓN de las señales de entrada

Ejemplo I

Diseñar un circuito que convierta un número binario de 3 bits en un número en código Gray

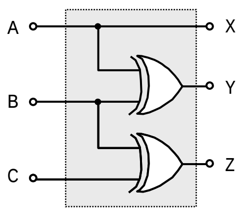
	A	B	C	X	Y	Z
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0



X

C \ AB	00	01	11	10
0	0	0	1	1
1	0	0	1	1

X = A



Y

C \ AB	00	01	11	10
0	0	1	0	1
1	0	1	0	1

$$Y = \overline{A}B + A\overline{B} = A \oplus B$$

Z

C \ AB	00	01	11	10
0	0	1	1	0
1	1	0	0	1

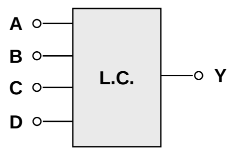
$$Z = B\overline{C} + C\overline{B} = B \oplus C$$

UPM DIE

Ejemplo II

Diseñar un circuito cuya entrada sea un número de 4 dígitos y la salida sea 1 cuando el número de entrada sea primo

	A	B	C	D	Y
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0



Y CD \ AB	00	01	11	10
00	1	0	0	0
01	1	1	1	0
11	1	1	0	1
10	1	0	0	0

$$Y = \overline{A}\cdot\overline{B} + \overline{A}\cdot D + B\cdot\overline{C}\cdot D + \overline{B}\cdot C\cdot D$$

UPM DIE

Ejemplo III

Diseñar un circuito cuya entrada sea un número en BCD de 4 dígitos y la salida sea 1 cuando la entrada valga 1, 2, 5, 6 ó 9.

	A	B	C	D	Y
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
	1	0	1	0	X
	1	0	1	1	X
	1	1	0	0	X
	1	1	0	1	X
	1	1	1	0	X
	1	1	1	1	X

Con 4 dígitos sólo se puede codificar del 0 al 9 con el código BCD ⇒ Hay posibles combinaciones a la entrada que nunca se van a dar ⇒ Salida IRRELEVANTE que podemos aprovechar para minimizar la lógica combinacinal necesaria.

(En BCD, el número 10 necesita 8 dígitos ⇒ no se puede representar con 4)

X CD \ AB	00	01	11	10
00	0	0	X	0
01	1	1	X	1
11	0	0	X	X
10	1	1	X	X

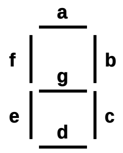
$$Y = \overline{C}D + C\overline{D} = C \oplus D$$

UPM DIE

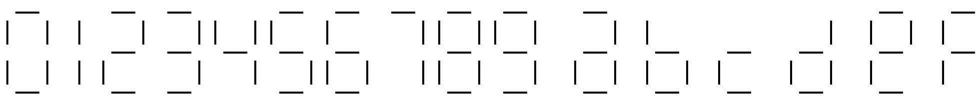
Convertidores de código

A) Convertidor BCD a 7 segmentos

E_3	E_2	E_1	E_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
resto				x	x	x	x	x	x	x



B) Hexadecimal a 7 segmentos



Más complejo

Mayor número de puertas

UPM DIE

Circuitos combinacionales. Bloques funcionales

Decodificadores y codificadores

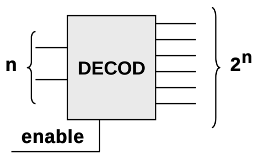
Multiplexores y demultiplexores

Funciones lógicas mediante decod/multiplexores

UPM DIE

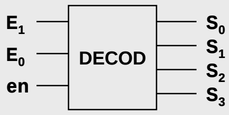
Decodificadores y codificadores

DECODIFICADOR



Se activa la salida correspondiente al número binario codificado en la entrada

***Ejemplo*:**
Decod 2 entradas con enable



en	E_1	E_0	S_0	S_1	S_2	S_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Funciones lógicas

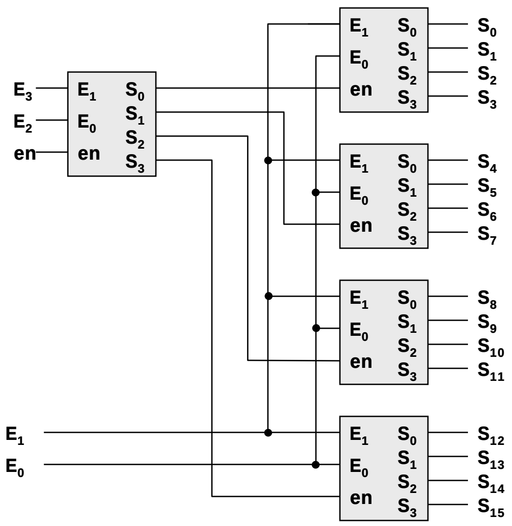
$$S_0 = en \cdot \overline{E}_1 \overline{E}_0$$
$$S_1 = en \cdot \overline{E}_1 E_0$$
$$S_2 = en \cdot E_1 \overline{E}_0$$
$$S_3 = en \cdot E_1 E_0$$

Si en=1, cada una de las salidas del decodificador representa a un término de la 1ª forma canónica de la función $f(E_1,E_0)$. Por tanto, puede representarse cualquier función mediante la adecuada combinación de las salidas de un decodificador.

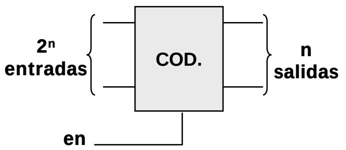
UPM DIE

EJEMPLO

A partir de decodificadores de 2 entradas, construir un decodificador de 4 entradas



CODIFICADOR



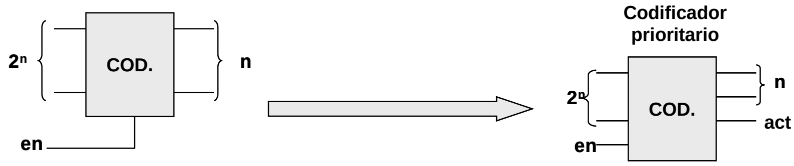
Se codifica en binario sobre la salida el número de entrada que esté activa

en	E_0	E_1	E_2	E_3	A_1	A_0
0	x	x	x	x	0	0
1	0	0	0	1	1	1
1	0	0	1	0	1	0
1	0	1	0	0	0	1
1	1	0	0	0	0	0

¿Cómo distinguir estos dos casos?

Señal de salida adicional

CODIFICADOR

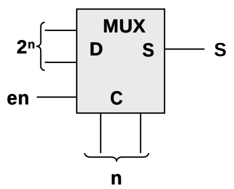


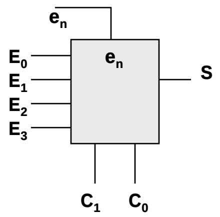
Codificador prioritario

en	E_0	E_1	E_2	E_3	A_1	A_0	act	
0	x	x	x	x	0	0	0	deshabilitado
1	0	0	0	0	0	0	0	inactivo
1	x	x	x	1	1	1	1	activo
1	x	x	1	0	1	0	1	activo
1	x	1	0	0	0	1	1	activo
1	1	0	0	0	0	0	1	activo

Multiplexores y demultiplexores

MULTIPLEXOR (MUX)

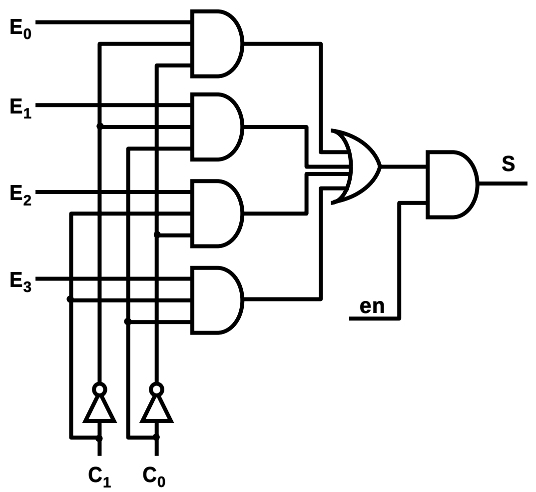


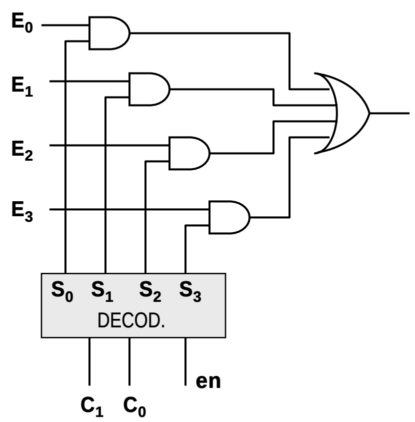


La entrada de datos correspondiente al número codificado en binario en las señales de control se conecta a la salida

en	E_0	E_1	E_2	E_3	C_1	C_0	S
0	X	X	X	X	X	X	0
1	D	X	X	X	0	0	D
1	X	D	X	X	0	1	D
1	X	X	D	X	1	0	D
1	X	X	X	D	1	1	D

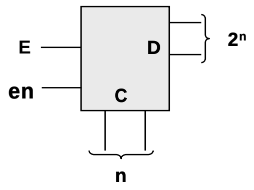
MUX mediante puertas lógicas



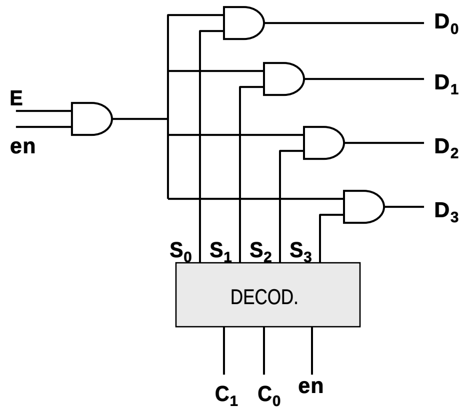


UPM DIE

DEMULTIPLEXOR



Saca la entrada por aquella salida correspondiente al número codificado en las señales de control



en	C_1	C_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	E	0	0	0
1	0	1	0	E	0	0
1	1	0	0	0	E	0
1	1	1	0	0	0	E

$D_2 = en\, C_1 \overline{C_0} E$

UPM DIE

Funciones lógicas mediante decodificadores/mux

Ejemplo

Diseñar un circuito que tiene como entrada el mes del año codificado en binario y como salida un '1' si el mes es de 31 días o un '0' si es de menos de 31 días

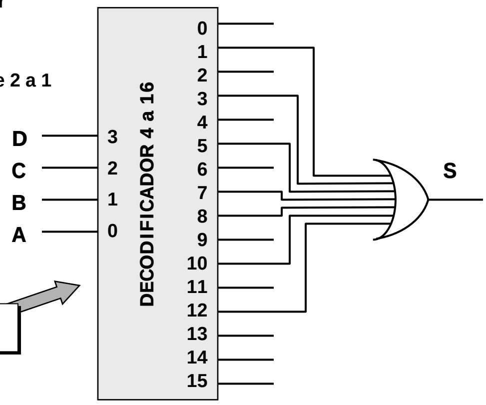
D	C	B	A	S
0	0	0	0	x
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x

A Mediante un decodificador

B Mediante un multiplexor

C Mediante multiplexores de 2 a 1

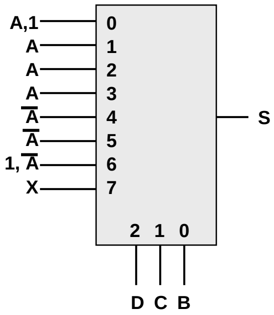
A **Un decodificador**



B

Un MUX

D	C	B	A	S
0	0	0	0	x
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x



C

MUX 2 a 1