TEMA 8
SISTEMAS DIGITALES

Parte 1. ESTADOS LÓGICOS Y PUERTAS LÓGICAS

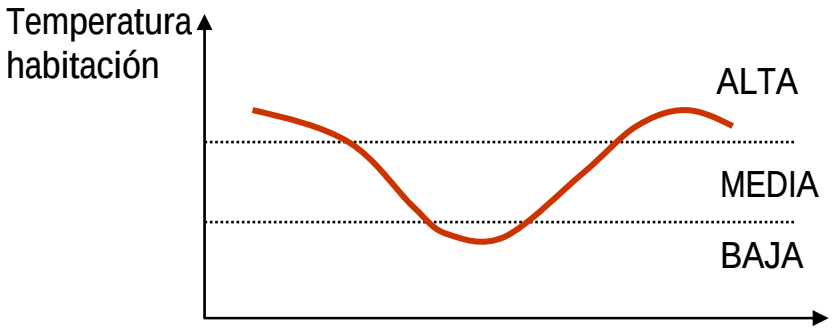
Parte 2. FUNCIONES LÓGICAS Y MAPAS DE KARNAUGH

Parte 3. SISTEMAS DE NUMERACIÓN

Parte 4. BLOQUES FUNCIONALES

Introducción

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



Discretización de una señal analógica

- Mundo real es un mundo analógico. ¿Por qué usar SISTEMAS DIGITALES? Porque nos dan
  - **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**

PARTE 1
ESTADOS LÓGICOS Y PUERTAS LÓGICAS

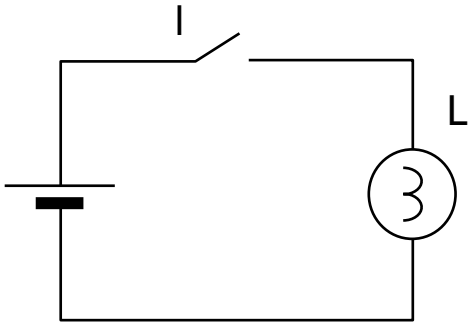
Estados lógicos

Puertas lógicas

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**



I	L
ABIERTO	APAGADA
CERRADO	ENCENDIDA

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

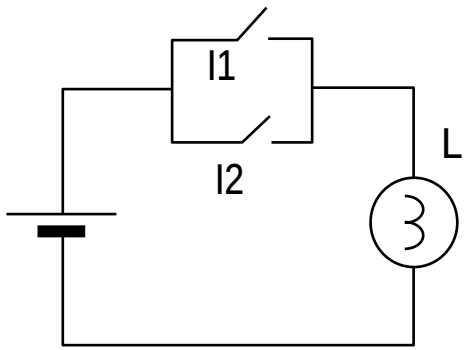
CERRADO = '1'

ABIERTO = '0'

TABLA DE VERDAD: Tabla en la que se relacionan todas las variables del sistema

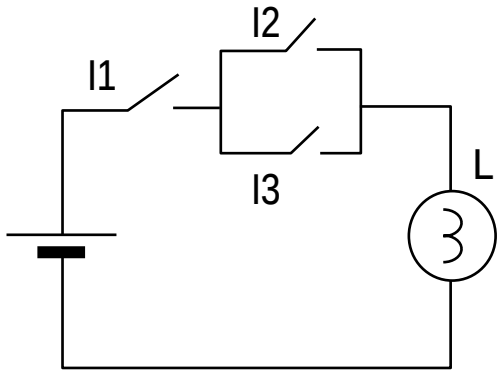
I	L
0	0
1	1

Ejemplos



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

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



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

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

Álgebra de Boole

- El Álgebra de Boole 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

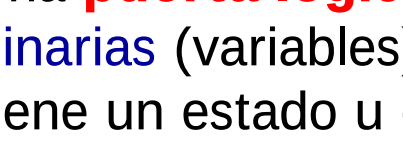
Variables Booleanas

- A, B, C, I1, I2, I3, L
- Tienen 2 estados posibles: '0' ó '1'

Puertas lógicas

- Una **puerta lógica** es una elemento que recibe varias entradas binarias (variables) y, dependiendo del estado de las entradas, su salida tiene un estado u otro.

Puerta AND



$C = A \cdot B$

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

AND

$0 \cdot 0 = 0$

$0 \cdot 1 = 0$

$1 \cdot 0 = 0$

$1 \cdot 1 = 1$

$A \cdot 0 = 0$

$0 \cdot A = 0$

$A \cdot 1 = A$

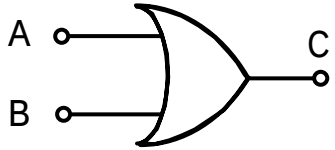
$1 \cdot A = A$

$A \cdot A = A$ Idempotencia

$A \cdot \overline{A} = 0$

Puertas lógicas

Puerta OR



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

C = A + B

OR

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 1

A + 0 = A

0 + A = A

A + 1 = 1

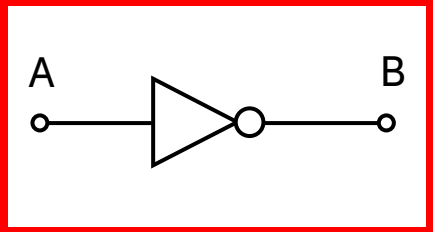
1 + A = 1

A + A = A Idempotencia

$A + \overline{A} = 1$

Puertas lógicas

Puerta NOT (inversor)



A	B
0	1
1	0

$B = \overline{A}$

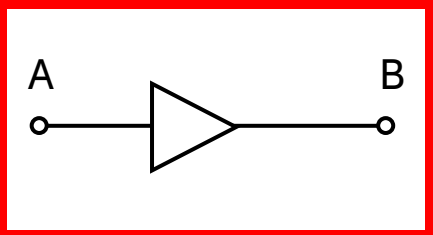
NOT

$\overline{0} = 1$

$\overline{1} = 0$

$\overline{\overline{A}} = A$

BUFFER



A	B
0	0
1	1

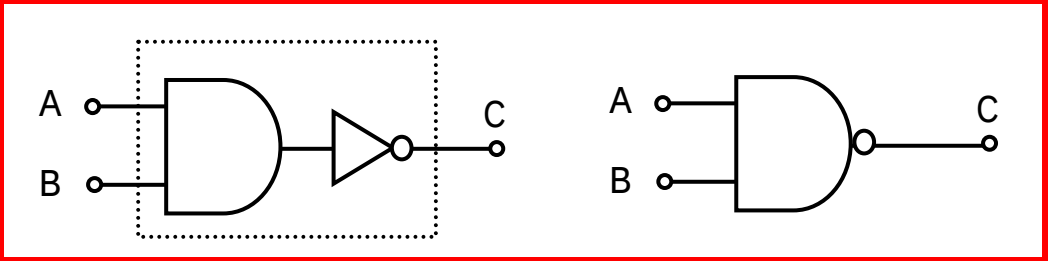
$B = A$

Cambia propiedades ELÉCTRICAS pero NO LÓGICAS

"Refuerza" la energía de la señal lógica (información)

Puertas lógicas

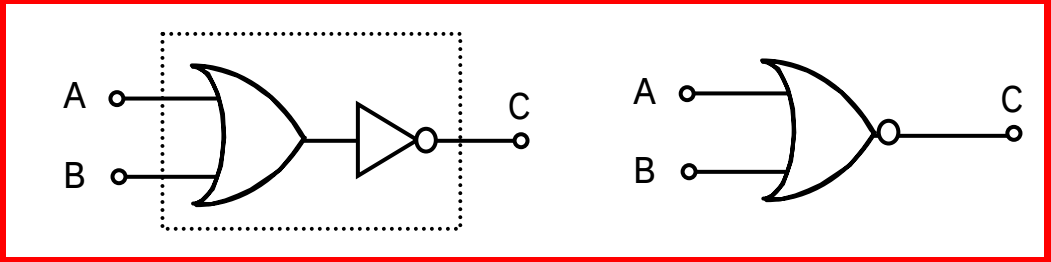
Puerta NAND



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

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

Puerta NOR

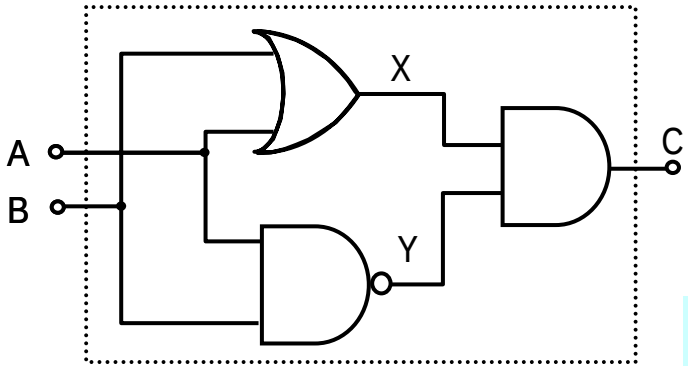


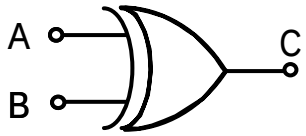
$$C = \overline{A + B}$$

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

Puertas lógicas

Puerta OR EXCLUSIVO

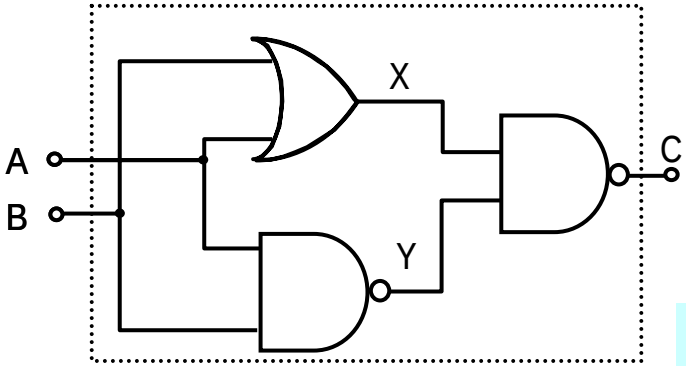


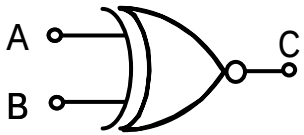


$C = A \oplus B$

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

PARTE 2
FUNCIONES LÓGICAS Y MAPAS DE KARNAUGH

Función lógica. Primera forma canónica

Ley de "De Morgan"

Segunda forma canónica

Mapas de Karnaugh

Función Lógica. Primera forma canónica

- Extracción de la función lógica de una tabla de verdad: 1ª forma canónica

A	B	C	D	
0	0	0	0	
0	0	1	1	$\bar{A}\bar{B}C$
0	1	0	0	
0	1	1	0	
1	0	0	1	$A\bar{B}\bar{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

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

PRODUCTO FUNDAMENTAL (Minterms)

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

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

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”

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

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

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

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

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 interviene en cada producto **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 los que interviene en cada suma **todas y cada una** de las variables de entrada

$f(AB) = (A + B + f(0,0)) \cdot (\overline{A} + B + f(1,0)) \cdot (A + \overline{B} + f(0,1)) \cdot (\overline{A} + \overline{B} + f(1,1))$

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 \ A	0	1
0	0	1
1	0	0

(B)

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

(C)

Sólo puede variar un dígito entre dos casillas adyacentes

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

$F = B\overline{C}D$

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

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 usando mapas de Karnaugh

REGLAS

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

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

3ª Cualquier grupo redundante debe eliminarse

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

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

Minimización con mapas de Karnaugh

Posibles asociaciones | El diagrama es esférico

$F = \overline{C}\,\overline{D} + AC$

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

$F = AB + \overline{B}\,\overline{D}$

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

$F = \overline{B}$

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

Función lógica

$$F = \overline{A}B\overline{C}\overline{D} + AB\overline{C}\overline{D} + A\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}B\overline{C}D + AB\overline{C}D + A\overline{B}\overline{C}D + \overline{A}\overline{B}CD + ABCD + \overline{A}BC\overline{D} + ABC\overline{D}$$

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

Forma mínima

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

Sólo puertas NAND

Aplicando "De Morgan" a forma mínima:

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

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

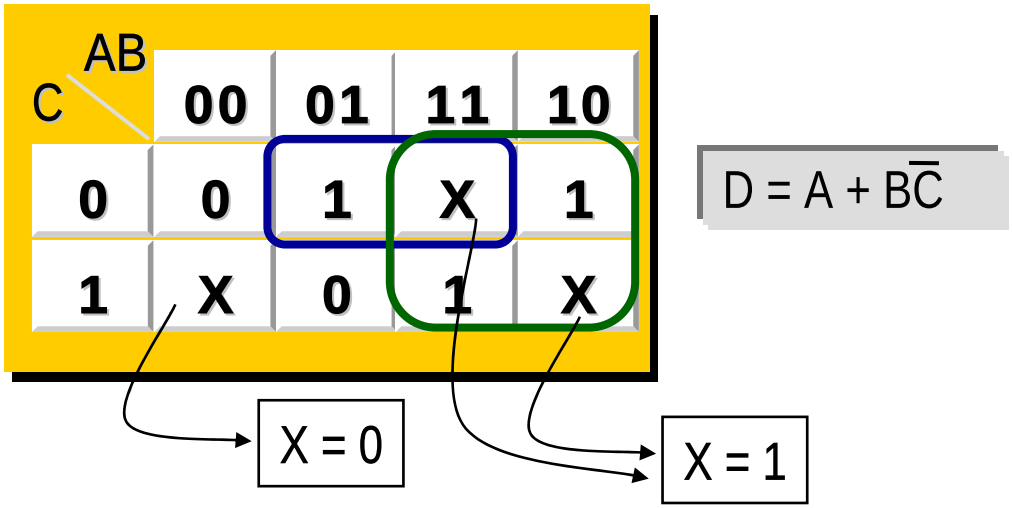
CONDICIÓN INDIFERENTE DE SALIDA

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

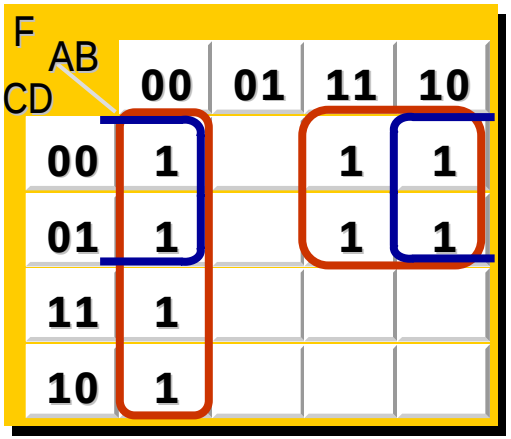
Si estas combinaciones nunca ocurren, la salida D es indiferente
$\Rightarrow D = X$

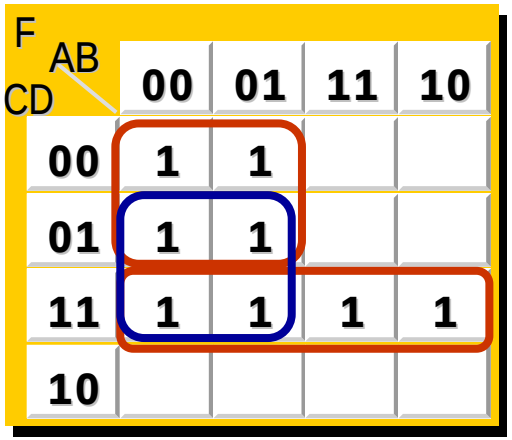
A	B	C	D
0	0	0	0
0	0	1	X
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	X
1	1	0	X
1	1	1	1

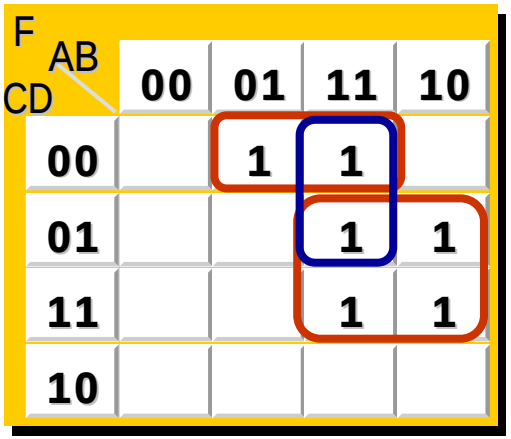


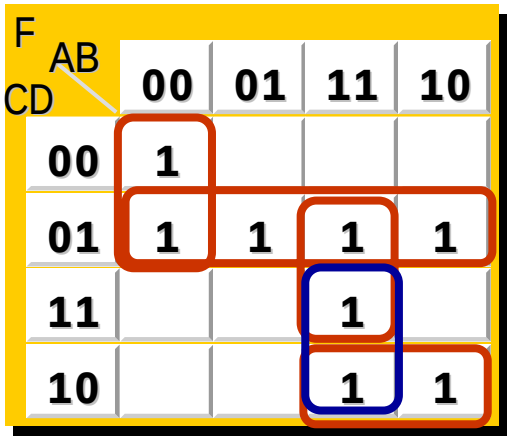
$D = A + B\overline{C}$

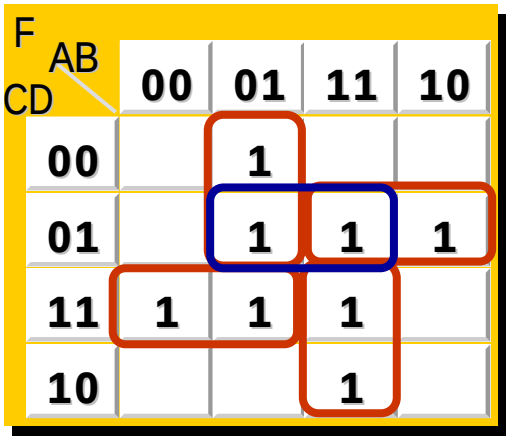
Grupos redundantes







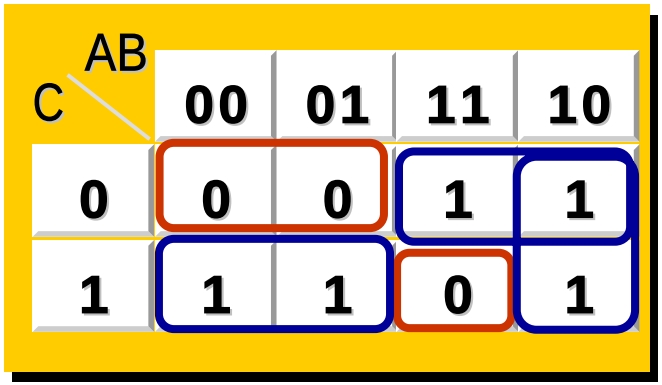




Karnaugh. Mínima expresión

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



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

Mínima expresión como suma de productos

Producto de sumas

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

Mínima expresión como producto de sumas

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

PARTE 3
SISTEMAS DE NUMERACIÓN

Sistemas de numeración

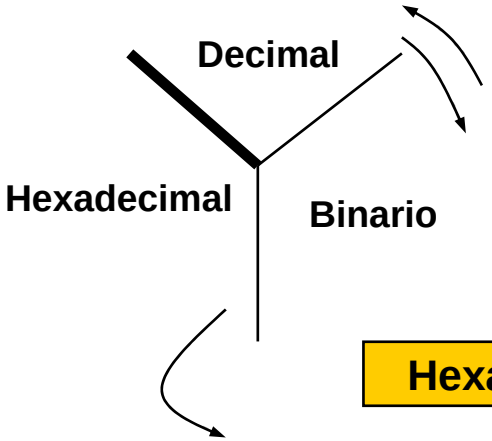
Base 10: $1327 = 1 \times 10^3 + 3 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$

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

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

Cambio de sistema de numeración

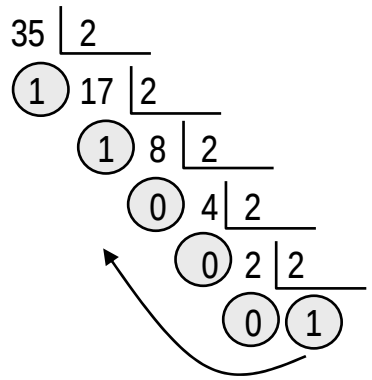
Cambios de base



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

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

PARTE 4
BLOQUES FUNCIONALES

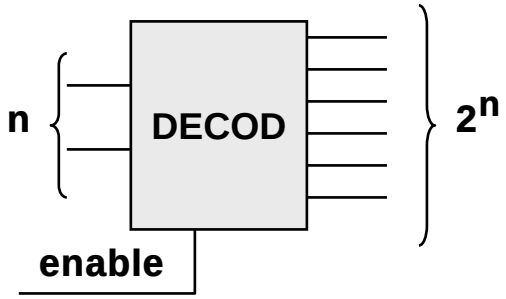
Decodificadores y codificadores

Multiplexores y demultiplexores

Funciones lógicas mediante decod/multiplexores

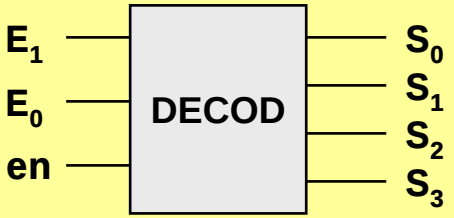
Decodificadores y codificadores

DECODIFICADOR



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

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



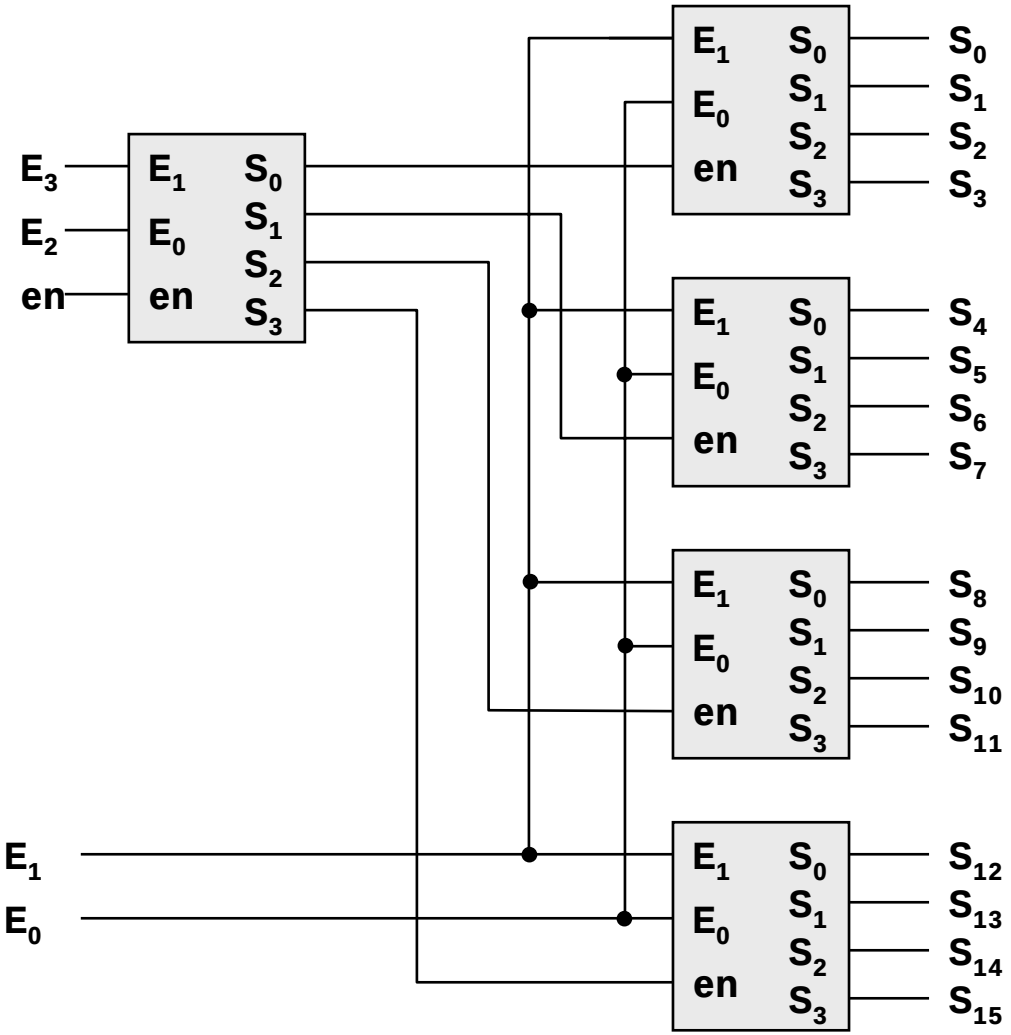
en	E_1	E_0	S_0	S_1	S_2	S_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Funciones lógicas

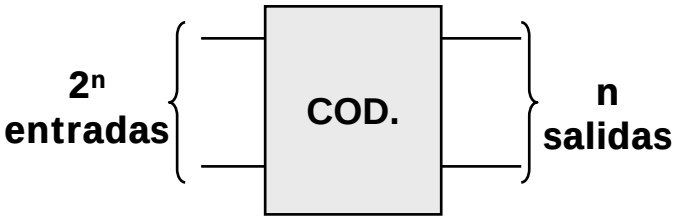
$$S_0 = e_n \cdot \overline{E}_1 \overline{E}_0$$
$$S_1 = e_n \cdot \overline{E}_1 E_0$$
$$S_2 = e_n \cdot E_1 \overline{E}_0$$
$$S_3 = e_n \cdot E_1 E_0$$

EJEMPLO

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



CODIFICADOR



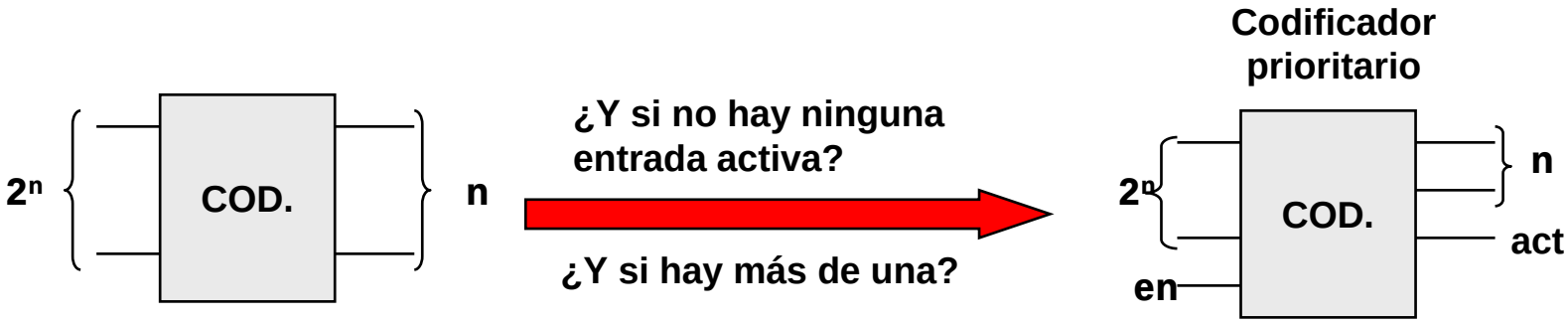
Se codifica en binario sobre la salida el número de entrada que esté activa

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

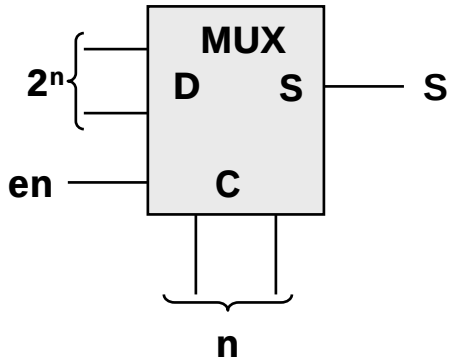


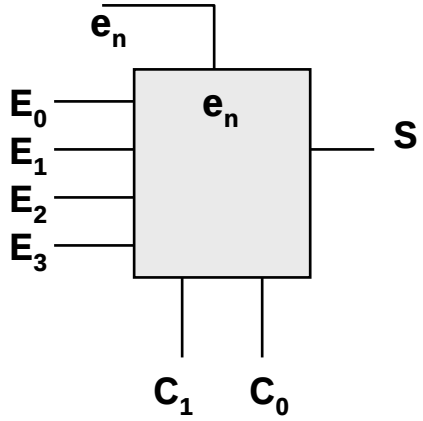
Codificador prioritario

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	
1	x	1	0	0	0	1	1	
1	1	0	0	0	0	0	1	

Multiplexores y demultiplexores

MULTIPLEXOR (MUX)

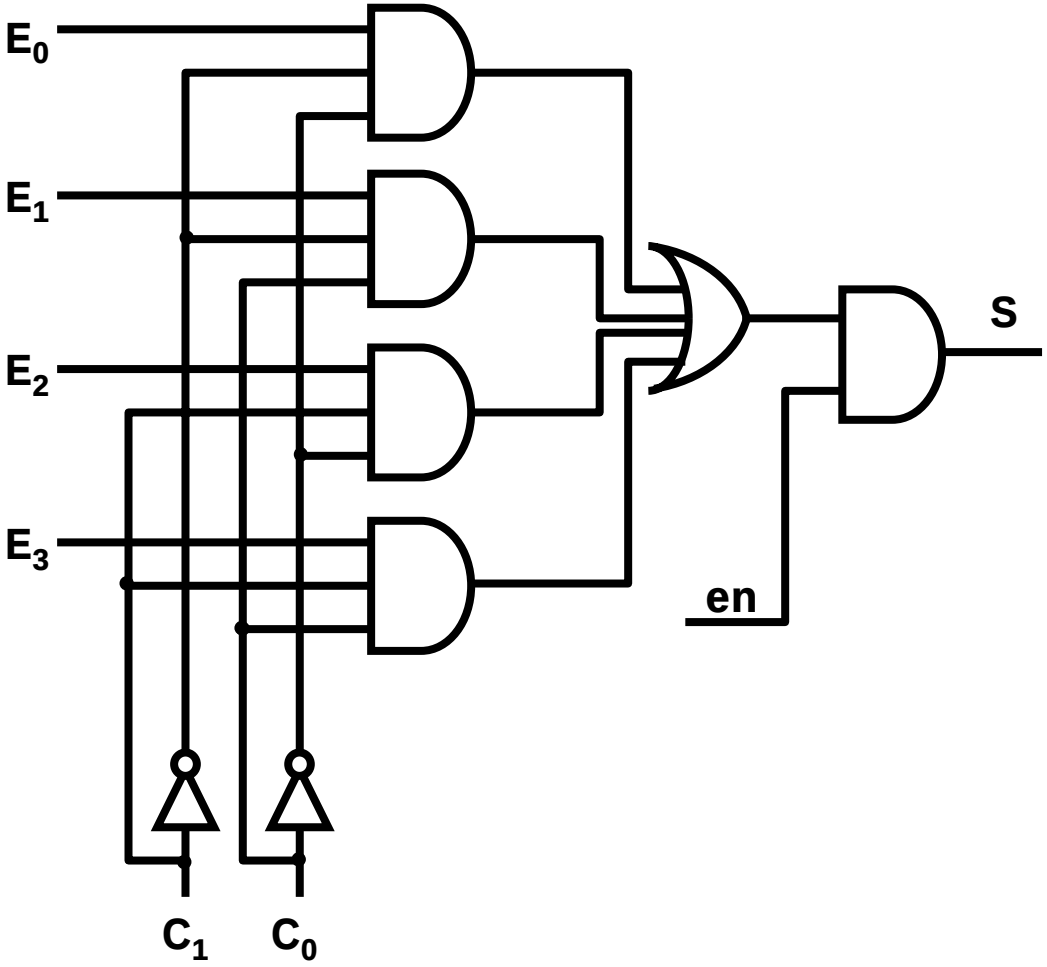


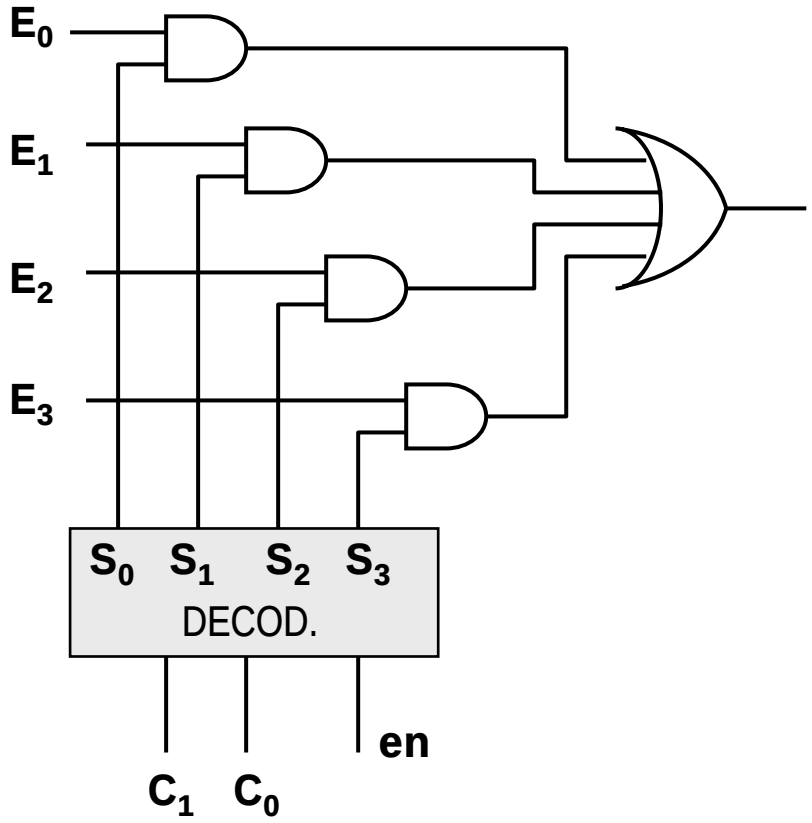


La entrada de datos correspondiente al número codificado en binario en las señales de control se conecta a la salida

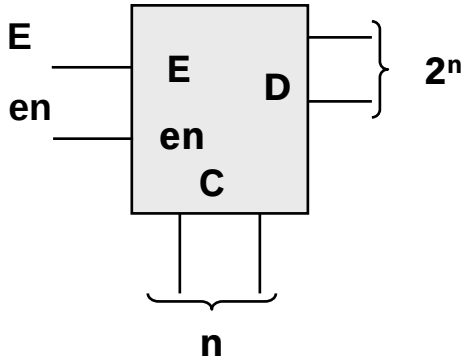
en	E_0	E_1	E_2	E_3	C_1	C_0	S
0	X	X	X	X	X	X	0
1	D	X	X	X	0	0	D
1	X	D	X	X	0	1	D
1	X	X	D	X	1	0	D
1	X	X	X	D	1	1	D

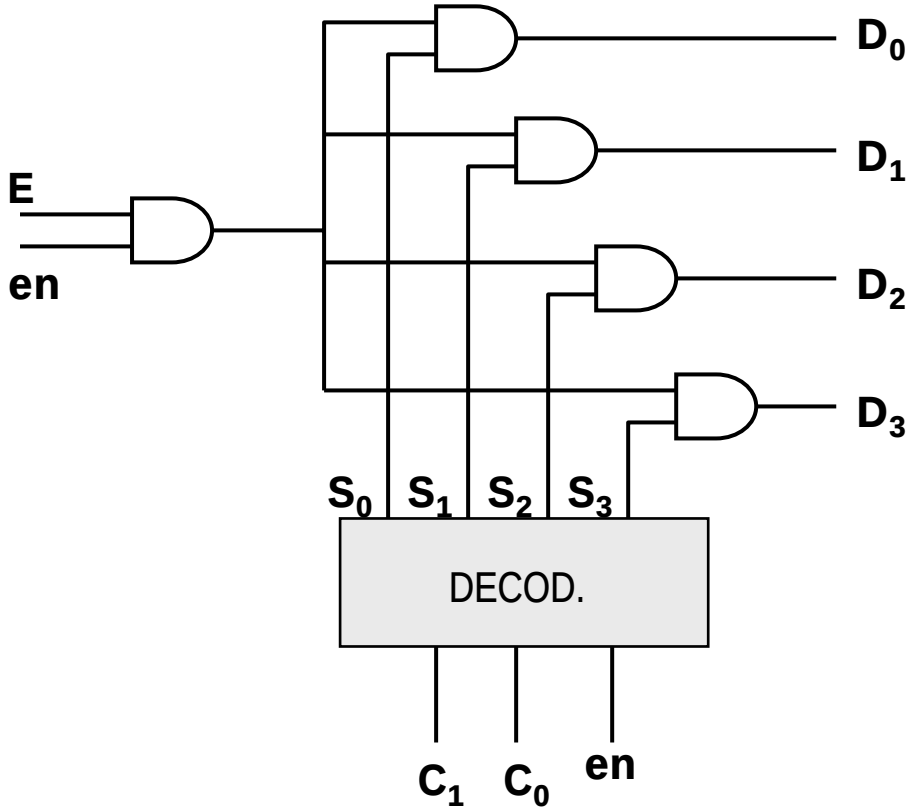
MUX mediante puertas lógicas





DEMULTIPLEXOR





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

en	C_1	C_0	S_0	S_1	S_2	S_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

$$S_2 = en\, C_1 \overline{C_0} E$$

Funciones lógicas mediante decodificadores/mux

Ejemplo

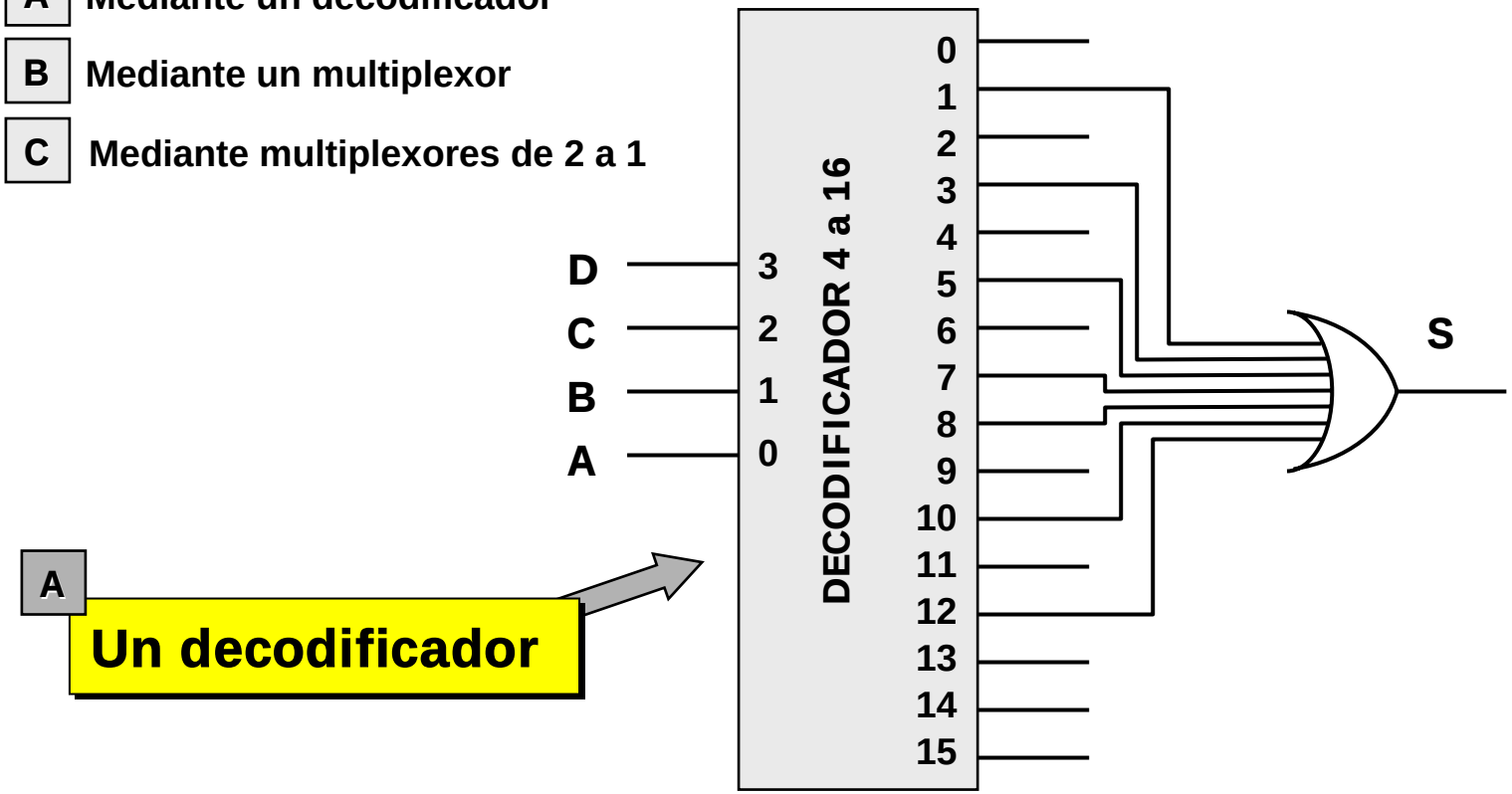
Diseñar un circuito que tiene como entrada el mes del año codificado en binario y como salida un '1' si el mes es de 31 días o un '0' si es de menos de 31 días

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

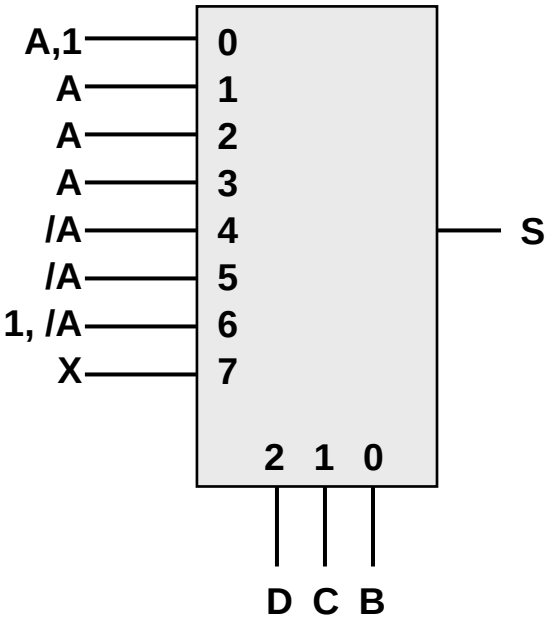
B Mediante un multiplexor

C Mediante multiplexores de 2 a 1

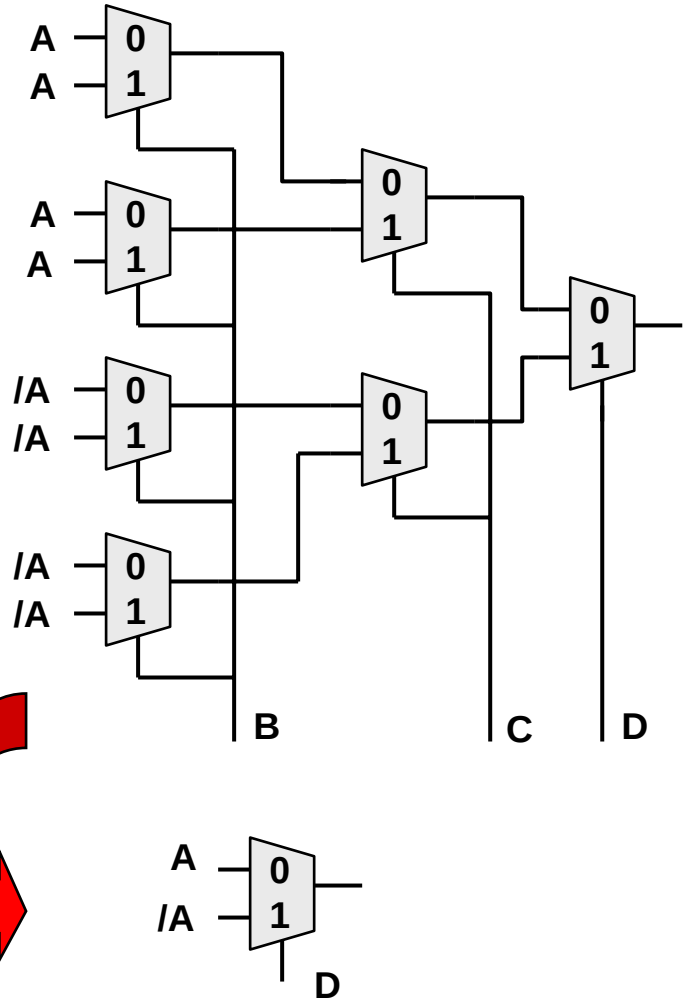


B Un MUX

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

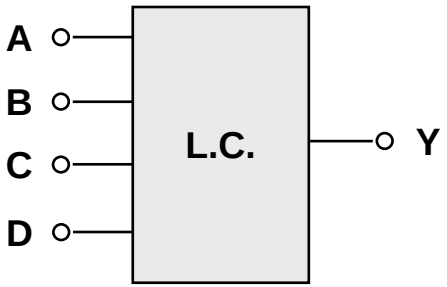


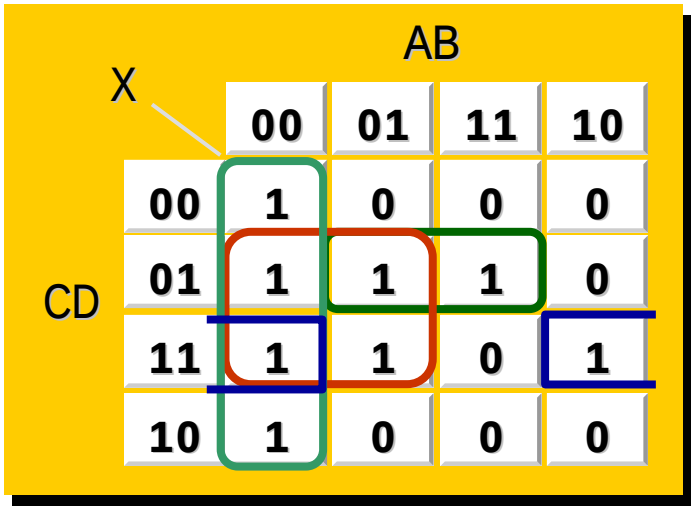
EJEMPLO
LÓGICA COMBINACIONAL

Ejemplo

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





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