

Introducción a los sistemas digitales

Luis Cucala García

¿Os suena esto? Motivos para trabajar en formato digital

- Ya lo comentamos en el Tema 1. Hay varios motivos para trabajar en formato digital:
 - Cualquier distorsión o ruido añadido a una señal analógica es irrecuperable, pero si que es regenerable una señal digital, mientras no se superen ciertos umbrales

Esto es un "1"



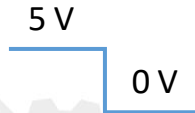
Y esto siguesiendo un "1"



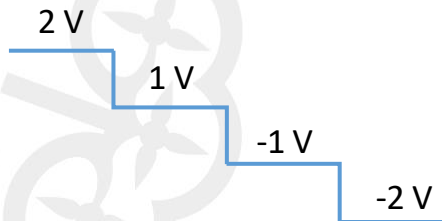
- Otro motivo, probablemente más importante, es que un circuito digital, si tiene elementos de memoria, puede ser programable. Un amplificador analógico solo puede hacer una cosa, amplificar, pero un circuito digital puede hacer muchas cosas con solo cambiar su programa (como en el caso de los microcontroladores)

Circuitos digitales. Lo básico

- Un sistema digital maneja un número discreto de tensiones. Si las tensiones posibles son dos, que es lo normal, hablamos de un sistema digital binario.
 - Es un circuito binario, cada nivel de tensión representa un bit de información

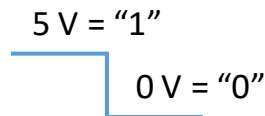


- Se pueden diseñar un sistema multinivel, donde haya N niveles posibles de tensión, de forma que cada nivel representa más de n bits de información (pues $2^n = N$). Son menos comunes por su dificultad de diseño y son menos inmunes al ruido (para vuestra culturilla, el móvil que tenéis en el bolsillo, en su etapa final de radiofrecuencia si maneja multiniveles)

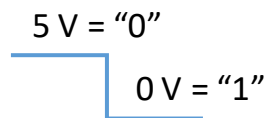


Circuitos digitales. Lo básico

- En un sistema binario, cada uno de los dos niveles de tensión se asocia a un estado lógico
 - Típicamente 0 V es un “0” lógico, y 5 V es un “1” lógico. Esto se conoce como lógica positiva



- También hay una lógica negativa, menos habitual, donde 0 V es un “1” lógico, y 5 V es un “0” lógico



- En lógica positiva, el “0” lógico casi siempre son 0 V, pero el “1” no siempre corresponde a 5 V. Depende de lo que se conoce como “familia lógica”, con esto nos referimos a la tecnología electrónica con la que está fabricado el circuito.
- En un sistema digital binario, las operaciones entre estados lógicos (ceros y unos para entendernos) se realizan mediante un álgebra conocida como **Álgebra de Boole**

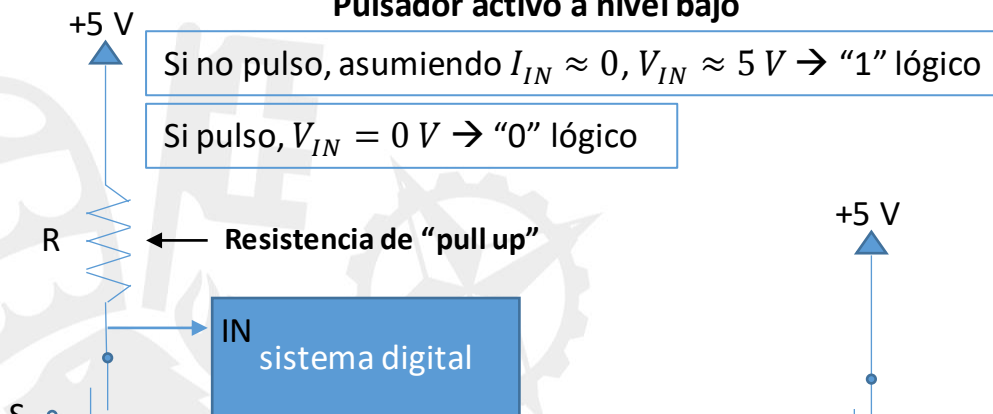
AND (símbolo: \cdot)



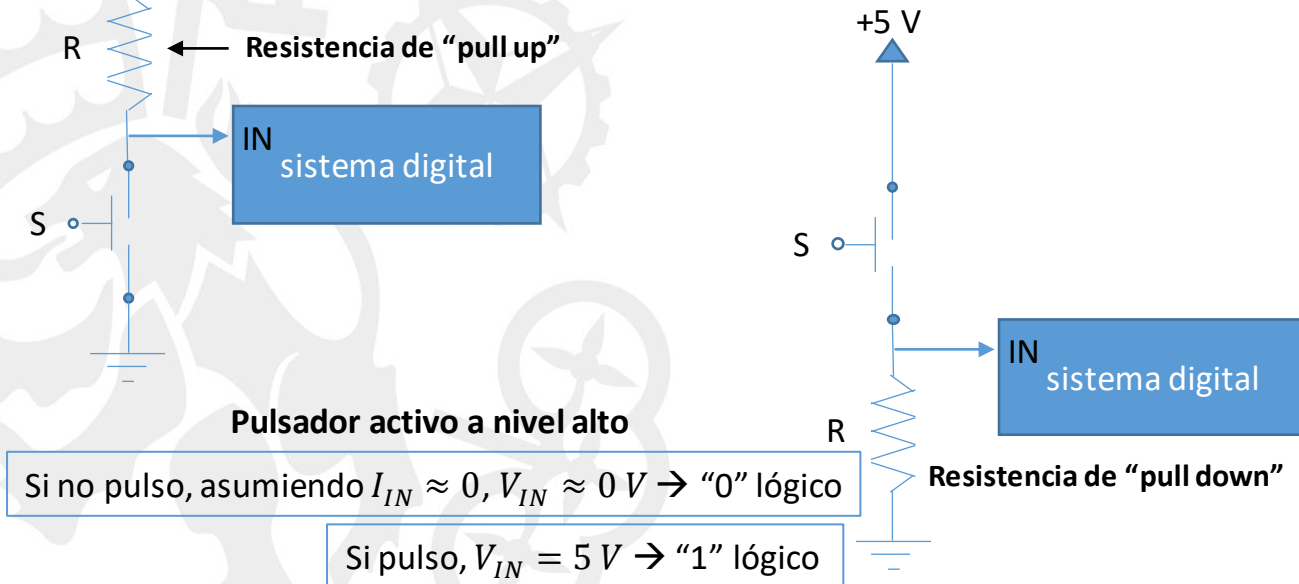
Circuitos digitales. Lo básico

- Formas sencillas de generar un 0 y un 1 lógicos mediante pulsador (asumimos lógica positiva, 0 V corresponde a "0", y 5 V corresponde a "1")

Pulsador activo a nivel bajo



Pulsador activo a nivel alto

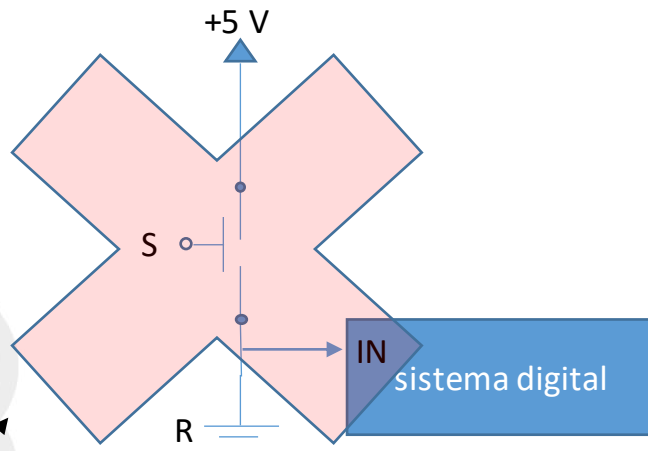
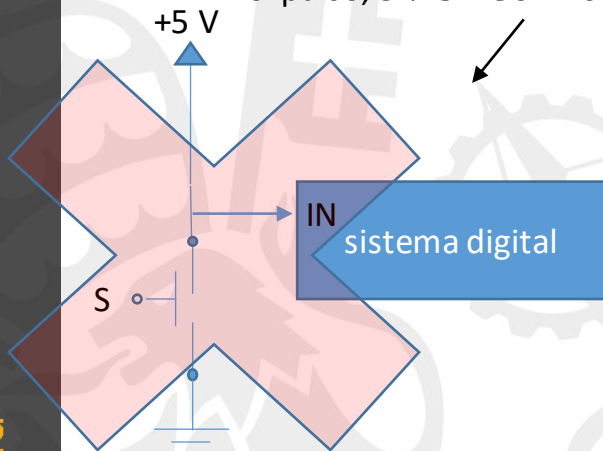


Circuitos digitales. Lo básico

¡¡NO PONER LAS RESISTENCIAS DE PULL UP O PULL DOWN ES UN ERROR MUY GRAVE!!

Si no pulso, $V_{IN} = 5V \rightarrow$ "1" lógico

Si pulso, 5 V SE CORTOCIRCUITA A MASA Y HUELE A HUMO ...

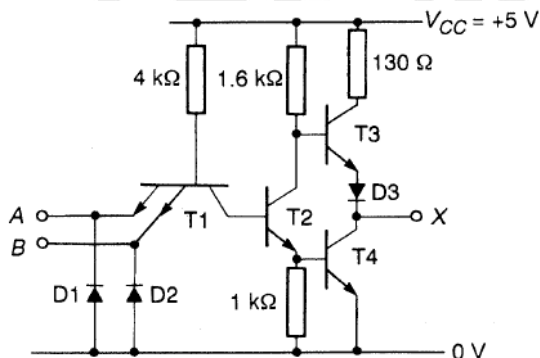


Si no pulso, $V_{IN} = 0V \rightarrow$ "0" lógico

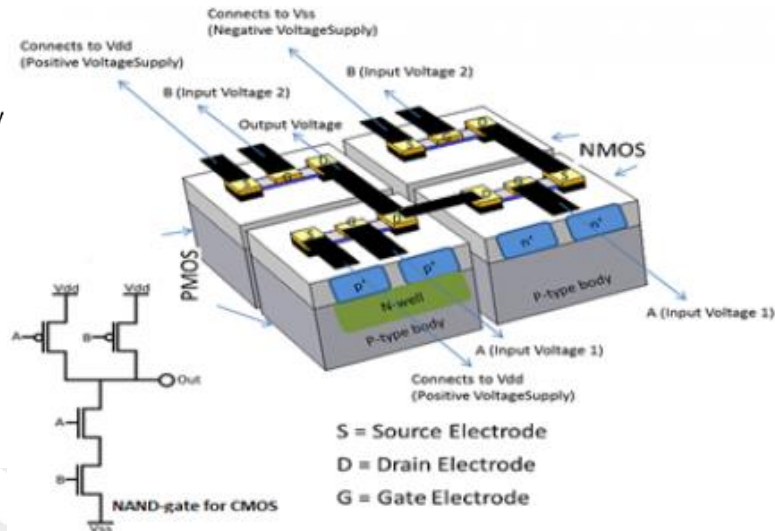
Si pulso, 5 V SE CORTOCIRCUITA A MASA Y HUELE A HUMO ...

Circuitos digitales. Lo básico

- No olvidéis nunca que todas las puertas lógicas que veréis más adelante están hechos con transistores, de modo que en el fondo son circuitos analógicos con limitaciones dictadas por cada familia lógica, y hay rangos admitidos de tensiones que son reconocidos como “0” y “1”, nº máximo de puertas que se pueden conectar unas a otras, incompatibilidad entre familias lógicas, etc.




Puerta NAND familia TTL



Puerta NAND familia CMOS

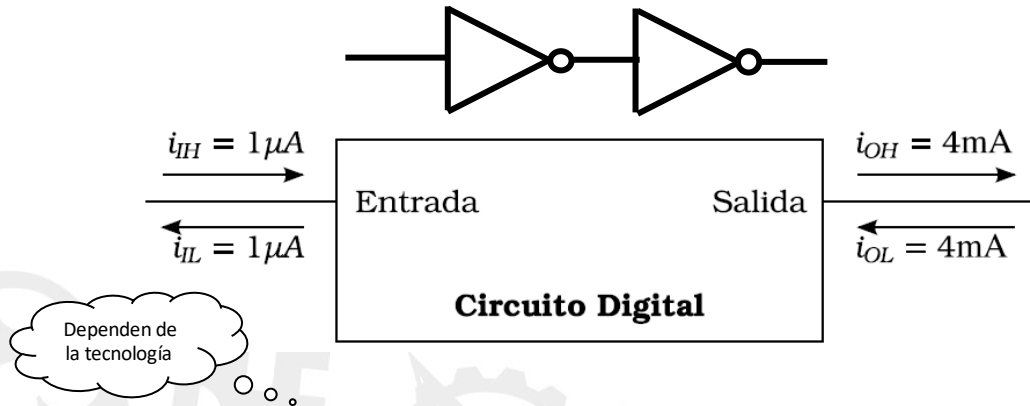
Familias lógicas

nº de puertas lógicas que se pueden conectar a la salida de otra puerta

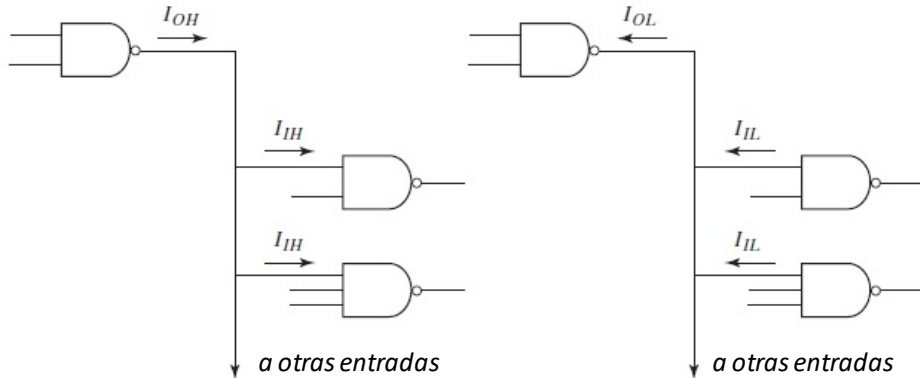


Familia	Fan Out	Potencia disipada por compuerta	Tiempo por compuerta	Alimentación
CMOS	50	0.01 mWatts estática 0.1 mWatts a 1 MHz	70 nanosegundos	3 - 18 Volts
HTL (Diodo Zener-Transistor)	10	55 mWatts	150 nanosegundos	14 - 16 Volts
TTL alta velocidad	10	22 mWatts	6 nanosegundos	5 Volts
TTL diodo Schottky	10	19 mWatts	3 nanosegundos	5 Volts
TTL bajo consumo	10	1 mWatt	33 nanosegundos	5 Volts
TTL bajo consumo con diodo Schottky	10	2 mWatts	9.5 nanosegundos	5 Volts
TTL estándar	10	12mWatts	10 nanosegundos	5 Volts

Corrientes en circuitos digitales. Fan-out

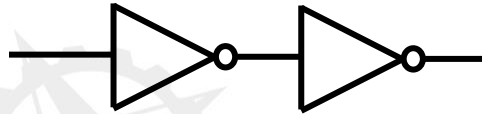


Corrientes de entrada y salida para la familia 74HC alimentada a 5 V.

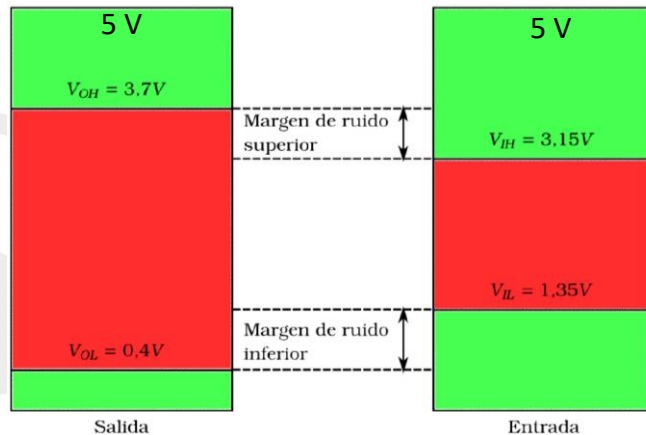


Tensiones en circuitos digitales

- A los estados verdadero/falso, 1/0, se les asigna un voltaje determinado (5/0V, 3/0V, etc), con cierto rango de variación
- Estos voltajes y rangos dependen de la familia lógica
- Por este motivo, no siempre es posible conectar dispositivos de una familia lógica con dispositivos de otra familia



Dependen de la tecnología

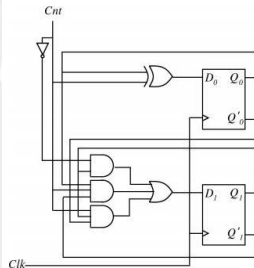
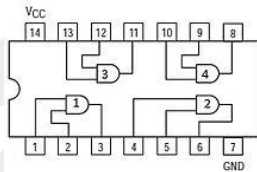
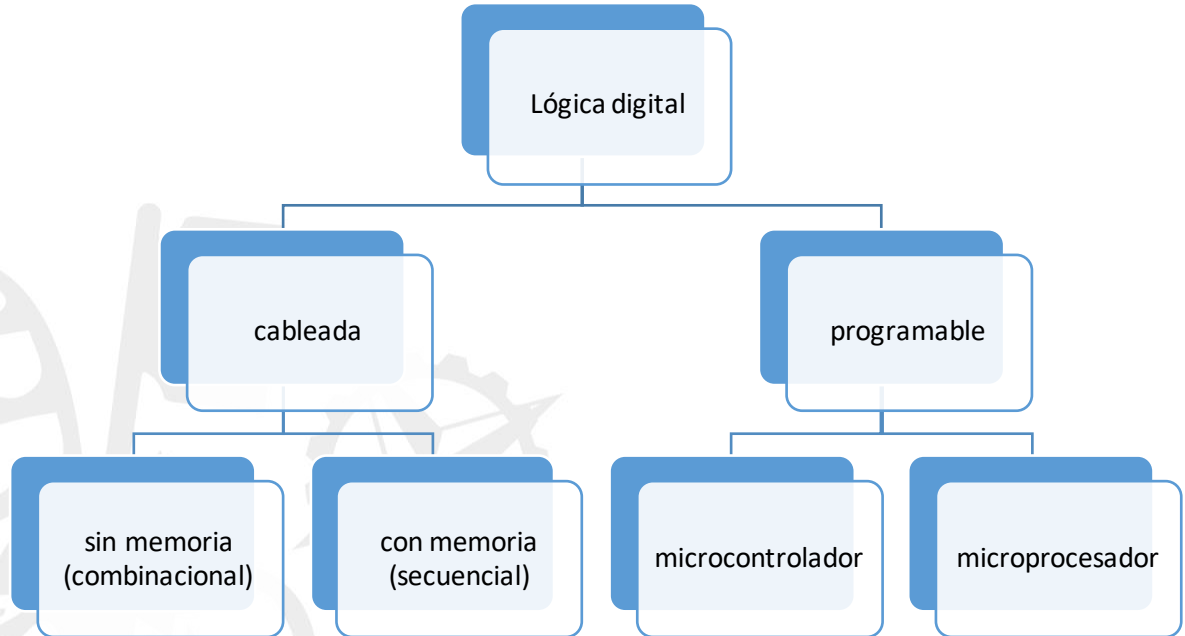


Tensiones y niveles lógicos para la familia 74HC alimentada a 5 V.

Circuitos digitales. Lo básico

- Las operaciones entre estados lógicos (Álgebra de Boole) a realizar en un circuito pueden ser fijas y sin posibilidad de alteración porque dependen de como es el circuito (circuito **cableado**), o se pueden alterar mediante el cambio del estado de sus elementos de memoria (circuito **programable**)
- Los circuitos **cableados**, pueden tener o no elementos de memoria
 - Si no tienen memoria, se llaman **combinacionales**. Hacen lo que tienen que hacer en función de sus entradas actuales, no guardan memoria de lo que sucedió en el pasado
 - Si tienen memoria se llaman **secuenciales**, lo que hacen depende de sus entradas actuales y de lo que sucedió en el pasado. En un circuito cableado secuencial no es posible cambiar su configuración (no le puedes meter un programa...)
- Como os podéis imaginar, todo circuito **programable** tiene elementos de memoria (si no ¿dónde guardas el programa?) y por tanto es secuencial (aunque no siempre)
 - Si el circuito programable tiene relativamente poca memoria interna, poca capacidad de procesamiento, baja velocidad de reloj... lo llamamos **microcontrolador** y se dedica a hacer funciones de control de poca o moderada complejidad.
 - En cambio, si tiene gigas de memoria, frecuencia de reloj del GHz para arriba, etc. lo llamamos **microprocesador**, lo tenemos en cada uno de nuestros móviles, PC's, etc.

Circuitos digitales. Lo básico



Circuitos digitales. Lo básico

- En un microcontrolador o microprocesador los bits nunca se manejan individualmente, se organizan en **palabras** de 8, 16, 32 y hasta 64 bits.
- La palabra mínima y más común es el **byte**, consistente en 8 bits
- Un byte nos permite $2^8 = 256$ combinaciones, de modo que
 - El nº más pequeño que se puede representar es el 0 (en binario 0000 0000)
 - Y el más grande el 255 (en binario 1111 1111)
- Los múltiplos del byte crecen en potencias de $2^{10} = 1024$ (de modo que un Gbyte no son 1.000 Mbyte, sino 1024 Mbyte)

Sistemas de numeración. Decimal y binario

- **Decimal**, el dígito en la posición i tiene un peso 10^i

p.e. $254 = 2 \cdot 10^2 + 5 \cdot 10^1 + 4 \cdot 10^0$

- **Binario**, el dígito en la posición i tiene un peso 2^i . Los circuitos digitales manejan números binarios

bit más significativo (MSB)

bit menos significativo (LSB)

p.e. $11111110(\text{binario}) = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 254 \text{ (decimal)}$

Decimal	Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Sistemas de numeración. Hexadecimal

- **Hexadecimal**, el dígito en la posición i tiene un peso 16^i
- Recordemos que los circuitos programables manejan palabras de como mínimo 1 byte (8 bits)
- Un byte consta de 8 bits. Podemos agruparlos en dos bloques de 4 bits, y recordando que 4 bits pueden representar 16 combinaciones ($2^4 = 16$), podemos representar cada uno de estos bloques de 4 bits con un número hexadecimal
- Las primeras 10 combinaciones del número hexadecimal (de 0000 a 1001) se representan con los símbolos (números) que van del 0 al 9
- Las siguientes usan las letras de la A a la F
- Por ejemplo, el byte **00111111** lo podemos representar como **3F** (y para resaltar que es un nº hexadecimal, se suele preceder con 0x, de modo que quedaría 0x3F)
- Esto es cómodo para nosotros los humanos, permite representar números binarios de forma compacta y se usa mucho en programación (el frió microprocesador siempre usa números binarios)

Decimal	Hexadecimal	binario
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Sistemas de numeración. Hexadecimal

Decimal	Hexadecimal	Binario
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

- Sistema de numeración de 16 dígitos (base 16):

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

- Ejemplo: B5 (en C sería 0xB5)

En decimal su valor es 181:

$$B \cdot 16^1 + 5 \cdot 16^0 = 11 \cdot 16^1 + 5 \cdot 16^0 = 181$$

Y podemos pasarlo a binario así (en C sería 0b10110101)

$$10110101 = 1 \cdot 2^7 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^2 + 1 \cdot 2^0 = 181$$

El truco para pasar de hexadecimal a binario es pasar cada dígito hexadecimal a binario

B en hexadecimal es 1011 binario (11 en decimal)

5 en hexadecimal es 0101 binario (5 en decimal)

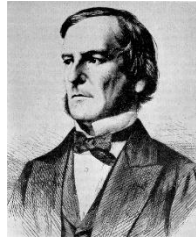
10110101

Sistemas de numeración. BCD

- **BCD** (binary-coded digit). Se representa cada dígito de un número decimal mediante un su equivalente binario, mediante 4 bits
 - P.e. 17 → 00010111
- Este sistema no es óptimo para representar números, pues quedan 6 combinaciones que no se usan
- BCD puede ser útil en situaciones especiales, como cuando queremos representar cada dígito por separado en un “display” de 7 segmentos

Dec	BCD natural			
	b_1	b_2	b_3	b_4
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
sin usar	1	0	1	0
	⋮			⋮
	1	1	1	1

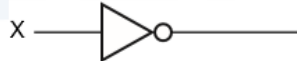
Álgebra de Boole aplicada a circuitos digitales



- Las operaciones entre estados lógicos (entre bits) se realizan mediante una aritmética llamada Álgebra de Boole
- Sean $X, Y...$ variables lógicas que pueden tomar el valor “0” o “1” (hablando con propiedad, “falso” (F) o “verdadero” (T), pues el álgebra de Boole se desarrolló originalmente en el campo de la lógica matemática), entonces tenemos:

Operadores:

NOT (símbolo: $X!$, X' , \bar{X} , $NOT\ X$)



$$Y = X!, X', \bar{X}, NOT\ X$$

AND (símbolo: $.$)



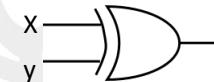
$$Z = X . Y$$

OR (símbolo: $+$)



$$Z = X + Y$$

XOR (símbolo: \oplus)



$$Z = X \oplus Y = A\bar{B} + \bar{A}B$$

Álgebra de Boole aplicada a circuitos digitales

Tablas de verdad de los operadores

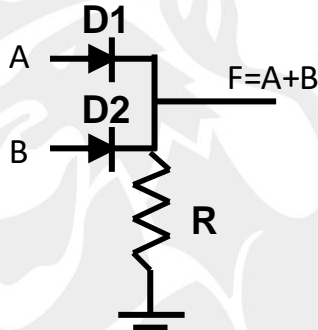
(estas tablas nos indican el valor de la salida en función de las entradas)

X	NOT
0	1
1	0

X	Y	AND
0	0	0
0	1	0
1	0	0
1	1	1

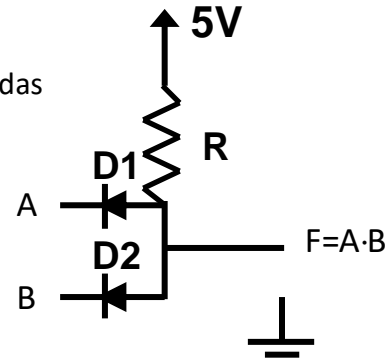
X	Y	OR
0	0	0
0	1	1
1	0	1
1	1	1

X	Y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

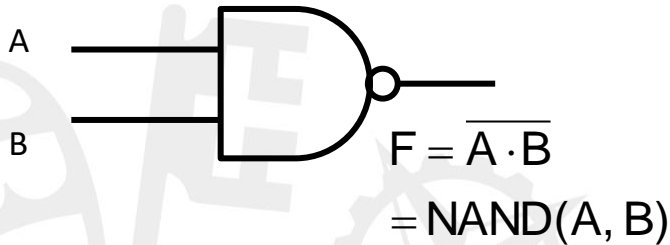


Ej. OR con diodos

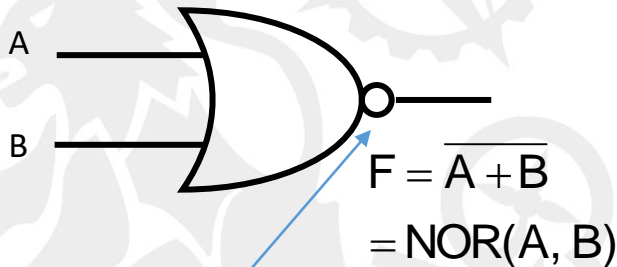
Ej. AND con diodos



Álgebra de Boole aplicada a circuitos digitales. Funciones negadas



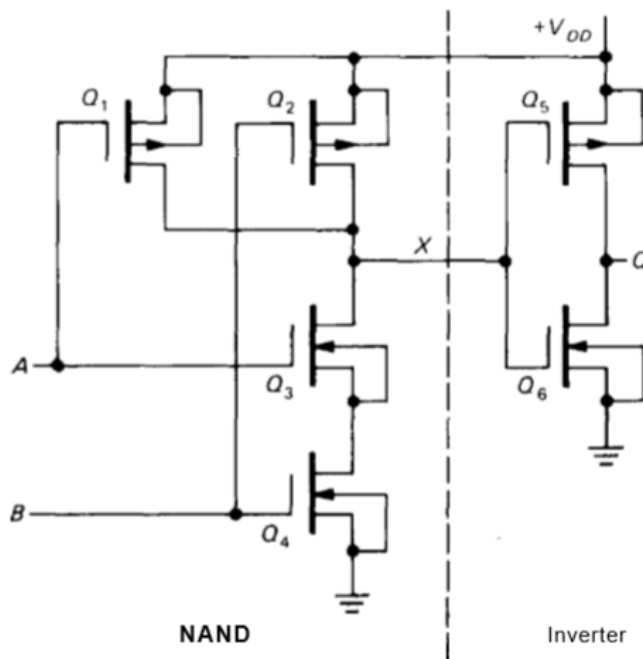
X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0



X	Y	NOR
0	0	1
0	1	0
1	0	0
1	1	0

El circulito a la salida indica que la salida está negada (p.e. antes del círculo es una OR, luego la salida es NOR)

Motivo para usar puertas NAND, NOR y NOT

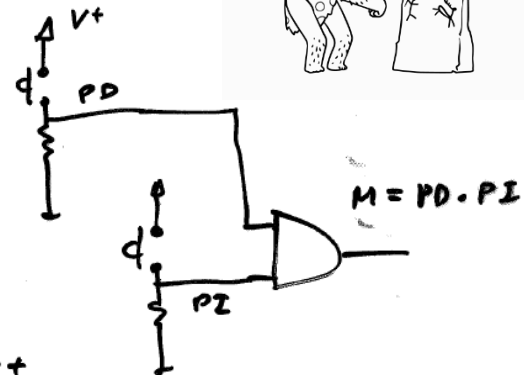
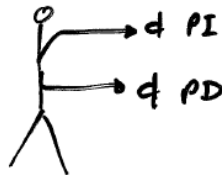
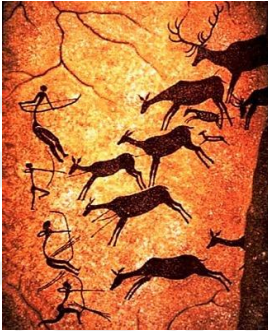


La puerta básica, hecha con el nº mínimo de transistores, es la NAND o la NOR. Para obtener AND u OR hay que añadir transistores:

- Mayor de ocupación de área en el circuito de silicio
- Mayor consumo de energía
- Mayor retardo de propagación

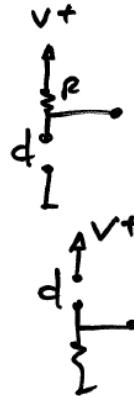
Ejemplos de funciones booleanas

- Maquina herramienta, con 2 pulsadores de seguridad (PI y PD) para su puesta en marcha:



Mb:
pulsador de 1 a 0

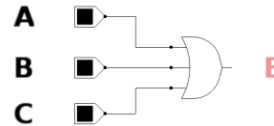
pulsador de 0 a 1



Ejemplos de funciones booleanas

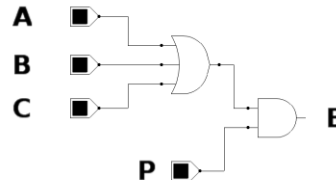
- Sistema de voto electrónico: el candidato es elegido ($E=1$) si cualquiera de 3 electores (A, B y C) le apoya:

$$E=A+B+C$$

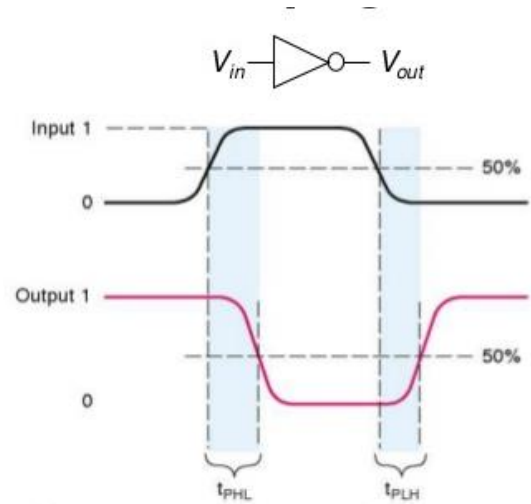
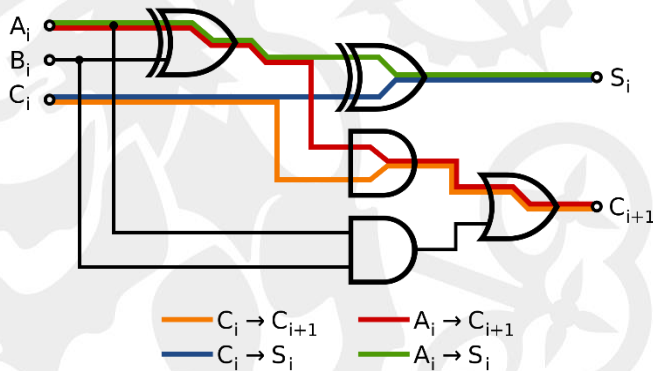


- El candidato es elegido ($E=1$) solo si además cuenta con el apoyo expreso del presidente (P)

$$E=P \cdot (A+B+C)$$

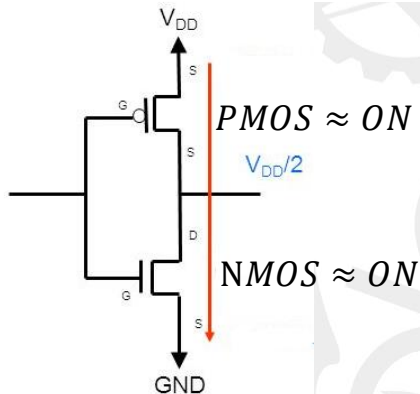
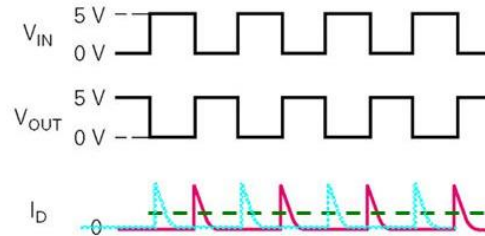
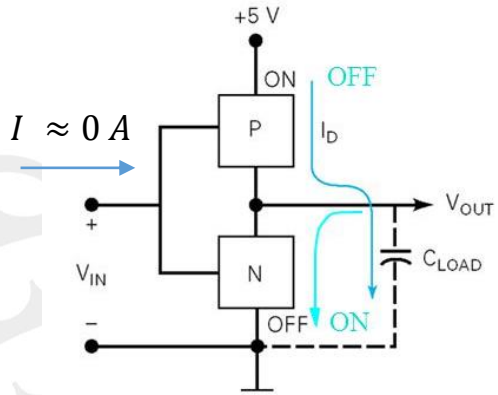


Retardo de propagación de una puerta



$$t_p = \frac{t_{pHL} + t_{pLH}}{2}$$

Disipación de potencia en la familia lógica CMOS



Circuitos integrados puertas NAND, NOR y NOT

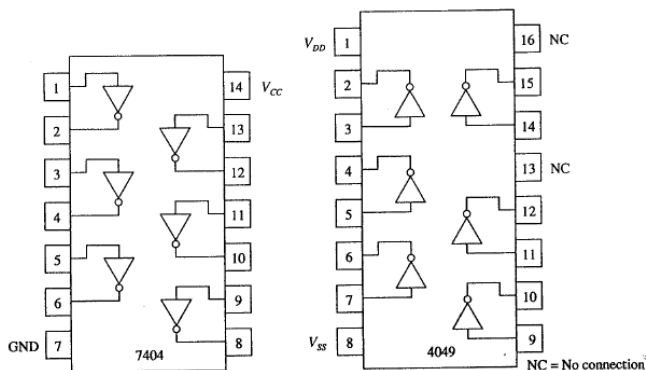


Figure 4-34 7404 TTL and 4049 CMOS inverter pin configurations.

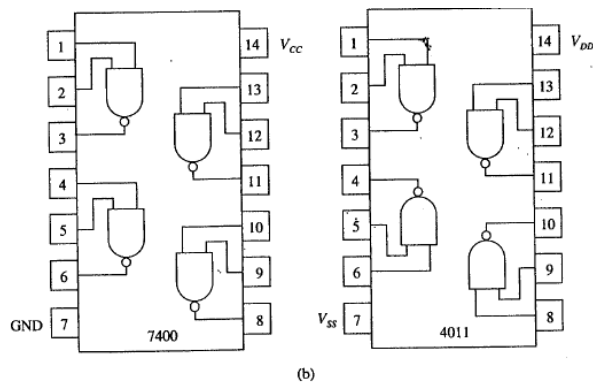
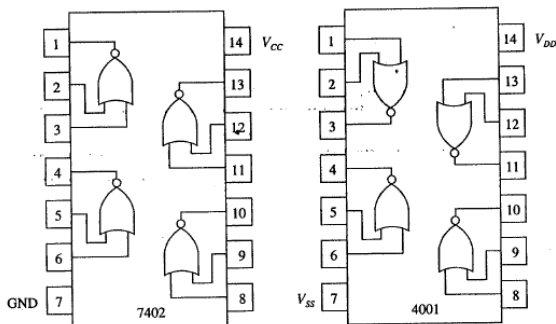
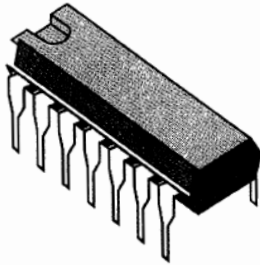
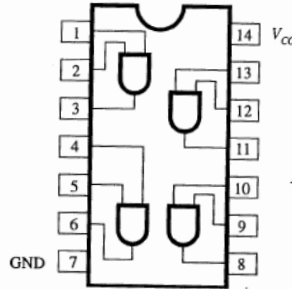


Figure 4-35 (a) 7402 TTL NOR and 4001 CMOS NOR pin configurations; (b) 7400 TTL NAND and 4011 CMOS NAND pin configurations.

Tipos de circuito según montaje

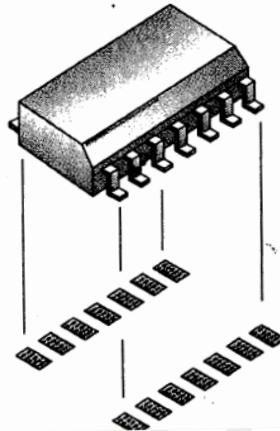
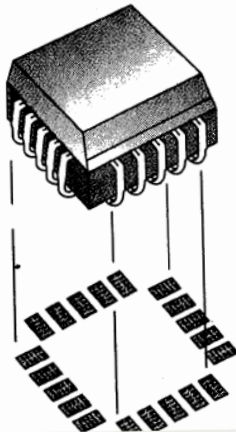


DIP



The 7408 quad two-input AND gate IC pin configuration.

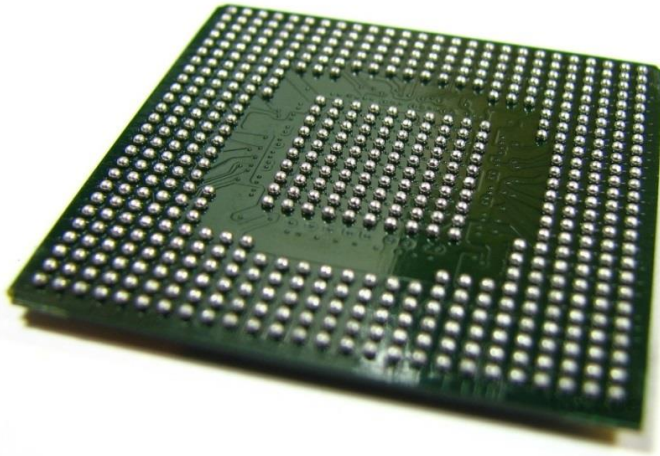
DIP (dual in-line packages)
THT: through-hole technology



SMD (Surface Mounted Device)
SMT: surface mount technology

- más pequeños
- más fáciles de montar por máquinas automáticas
- menos espacio
- diseño más complejo (tamaño, interferencias)
- requieren mayor automatización para ensamblado
- más difíciles de verificar por su menor tamaño

Tipos de circuito según montaje



BALL GRID ARRAY

<https://www.youtube.com/watch?v=tn0EKtLOVx4>

<https://www.youtube.com/watch?v=8G7pUqRZjU8>



Teoremas de álgebra de Boole

(ver "Digital design principles and practice") John F. Wakerly

(T1)	$X + 0 = X$	(T1')	$X \cdot 1 = X$	(Identities)
(T2)	$X + 1 = 1$	(T2')	$X \cdot 0 = 0$	(Null elements)
(T3)	$X + X = X$	(T3')	$X \cdot X = X$	(Idempotency)
(T4)	$(X')' = X$			(Involution)
(T5)	$X + X' = 1$	(T5')	$X \cdot X' = 0$	(Complements)
(T6)	$X + Y = Y + X$	(T6')	$X \cdot Y = Y \cdot X$	(Commutativity)
(T7)	$(X + Y) + Z = X + (Y + Z)$	(T7')	$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$	(Associativity)
(T8)	$X \cdot Y + X \cdot Z = X \cdot (Y + Z)$	(T8')	$(X + Y) \cdot (X + Z) = X + Y \cdot Z$	(Distributivity)
(T9)	$X + X \cdot Y = X$	(T9')	$X \cdot (X + Y) = X$	(Covering)
(T10)	$X \cdot Y + X \cdot Y' = X$	(T10')	$(X + Y) \cdot (X + Y') = X$	(Combining)
(T11)	$X \cdot Y + X' \cdot Z + Y \cdot Z = X \cdot Y + X' \cdot Z$			(Consensus)
(T11')	$(X + Y) \cdot (X' + Z) \cdot (Y + Z) = (X + Y) \cdot (X' + Z)$			
(T12)	$X + X + \dots + X = X$			(Generalized idempotency)
(T12')	$X \cdot X \cdot \dots \cdot X = X$			
(T13)	$(X_1 \cdot X_2 \cdot \dots \cdot X_n)' = X_1' + X_2' + \dots + X_n'$			(DeMorgan's theorems)
(T13')	$(X_1 + X_2 + \dots + X_n)' = X_1' \cdot X_2' \cdot \dots \cdot X_n'$			
(T14)	$[F(X_1, X_2, \dots, X_n, +, \cdot)]' = F(X_1', X_2', \dots, X_n', \cdot, +)$			(Generalized DeMorgan's theorem)
(T15)	$F(X_1, X_2, \dots, X_n) = X_1 \cdot F(1, X_2, \dots, X_n) + X_1' \cdot F(0, X_2, \dots, X_n)$			(Shannon's expansion theorems)
(T15')	$F(X_1, X_2, \dots, X_n) = [X_1 + F(0, X_2, \dots, X_n)] \cdot [X_1' + F(1, X_2, \dots, X_n)]$			

Esta la vamos a usar mucho en los diagramas de Karnaugh

Algebra de Boole

Teoremas y reglas
para manipular
expresiones
lógicas

Leyes

comutatividad

$$\begin{cases} A+B=B+A \\ A \cdot B=B \cdot A \end{cases}$$

asociatividad

$$\begin{cases} A+(B+C)=(A+B)+C \\ A \cdot (B \cdot C)=(A \cdot B) \cdot C \end{cases}$$

distributividad

$$\begin{aligned} (A+B) \cdot (C+D) \\ = AC+AD+BC+BD. \end{aligned}$$

Reglas

AND

$$\begin{aligned} A \cdot 0 &= 0 \\ A \cdot 1 &= A \\ A \cdot A &= A \\ A \cdot \bar{A} &= 0 \end{aligned}$$

OR

$$\begin{aligned} A+0 &= A \\ A+1 &= 1 \\ A+A &= A \end{aligned}$$

NOT

$$\bar{\bar{A}} = A$$

$$A+\bar{A} = 1 \Rightarrow \text{Karnaugh}$$

$$(A+B)(A+\bar{B}) = A$$

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

$$A+AB = A$$

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

$$A(\bar{A}+B) = AB$$

$$A+\bar{A}B = A+B$$

Leyes de De Morgan

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

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



Lógica combinacional

- Circuito combinacional: en cualquier instante la salida del circuito depende solamente de los valores de las entradas en ese instante: circuito sin memoria.
- Procedimiento de diseño sistemático de un circuito con lógica combinacional:
 - Escribir la tabla de verdad
 - Obtener la forma canónica (en rigor innecesaria)
 - Simplificación de la función lógica (muy importante)
 - Implantación con puertas lógicas (tipo único o múltiple)
- Ejemplo: diseñar un circuito de 3 entradas (A, B y C) cuya salida F sea 1 si dos de las entradas, pero no las tres, valen 1.

Ejemplo 1

A	B	C	F
0	0	0	0
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

Tabla de Verdad

Términos con salida 1

→ $\bar{A}BC$

→ $A\bar{B}C$

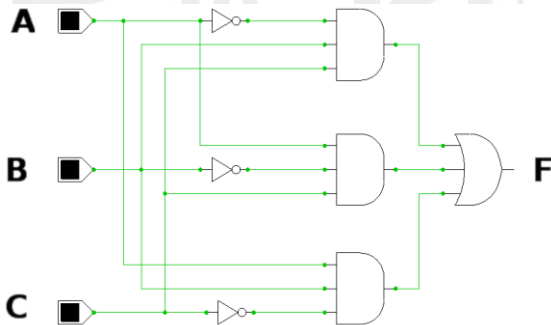
→ $AB\bar{C}$

$$F = \bar{A}BC + A\bar{B}C + AB\bar{C}$$

Forma canónica como suma de productos

salida F sea 1 si dos de las entradas, pero no las tres, valen 1

Forma canónica (contiene todos los términos sin simplificar nada)
Esta forma canónica se llama **Suma de Productos** (o minitérminos)

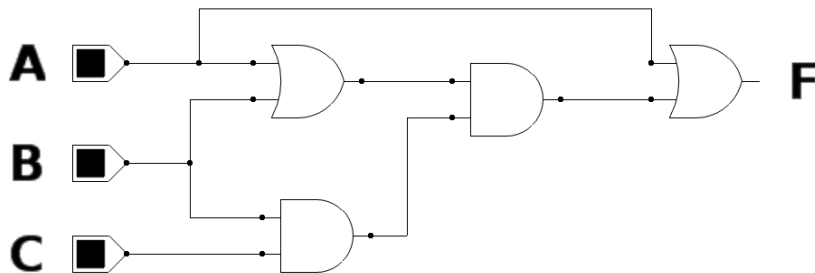
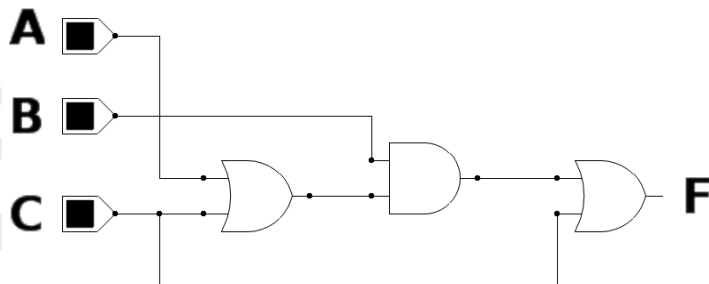


Esta implementación está sin simplificar.
Siempre intentaremos simplificar la forma canónica para ahorrar puertas

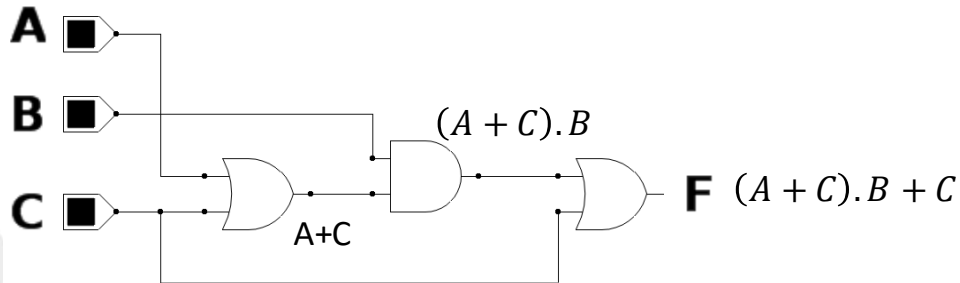
Hay otra forma canónica, llamada Producto de Sumas (o maxitérminos)
$$F = (A + B + C). (A + B + \bar{C}). (A + \bar{B} + C). (\bar{A} + B + C). (\bar{A} + \bar{B} + \bar{C})$$

Problema 1

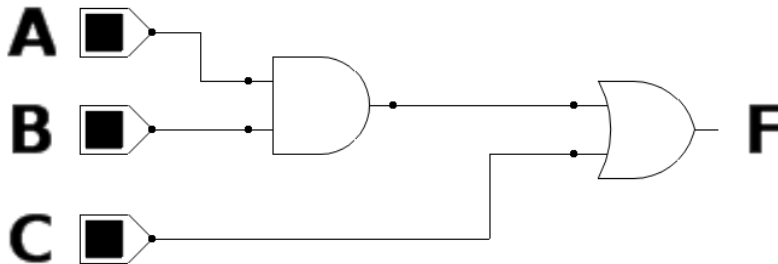
- Simplificar los siguientes circuitos



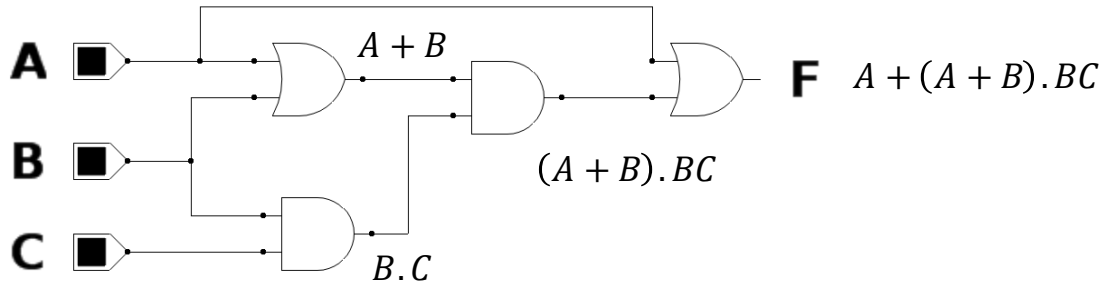
Problema 1.1



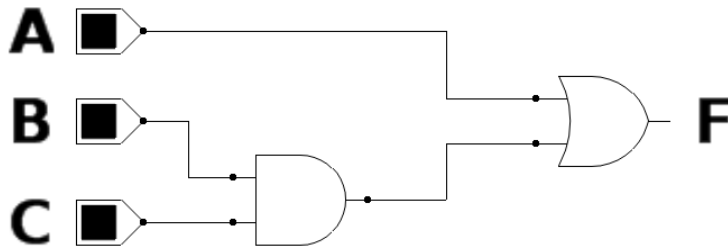
$$F = (A + C).B + C = A.B + C.B + C = A.B + C(B + 1) = A.B + C$$



Problema 1.2



$$F = A + (A + B).BC = A + A.B.C + B.C = A + B.C(A + 1) = A + B.C$$



Diagramas de Karnaugh



- Se basan en hacer explícita la existencia de términos del tipo:

$$A \cdot B + A \cdot \bar{B} = A(B + \bar{B}) = A \cdot 1 = A$$

- Para ello se usa una hábil representación de la función lógica a simplificar conocida como Diagrama de Karnaugh
- Con 2 variables A y B**

A		
	0	1
B	0	1
	1	1

Diagrama de Karnaugh para la función $F = AB + A\bar{B}$. La tabla muestra los valores de la función para las combinaciones de variables A y B. Las casillas con valor 1 están circunscritas en rojo. Las flechas indican las combinaciones de variables que se eliminan al simplificar: $A\bar{B}$ (para A=1, B=0) y AB (para A=1, B=1).

$$F = AB + A\bar{B} = A(B + \bar{B}) = A$$

(F nos indica que nos da igual el valor que tome B, solo importa que A =1)

- Se combinan casillas **adyacentes** de valor 1 y se elimina la variable que varía de una a otra
- Hay que tomar todos los "1" de la tabla al menos una vez



Problema 2

- Compruébese si se pueden considerar adyacentes los unos de la diagonal del siguiente diagrama, con objeto de simplificar ambos términos en uno único.

		A	
		0	1
B	0	1	0
	1	0	1

Diagrama de Karnaugh para la función $F = AB + \bar{A}\bar{B}$. El diagrama muestra una tabla de verdad con variables A y B. Los unos están en las celdas (0,1) y (1,0). Una línea roja dashed rodea los dos unos, con una flecha roja que apunta a un signo de interrogación rojo, indicando que no son adyacentes. Las expresiones $\bar{A}\bar{B}$ y AB están etiquetadas con flechas que apuntan a los unos correspondientes.

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

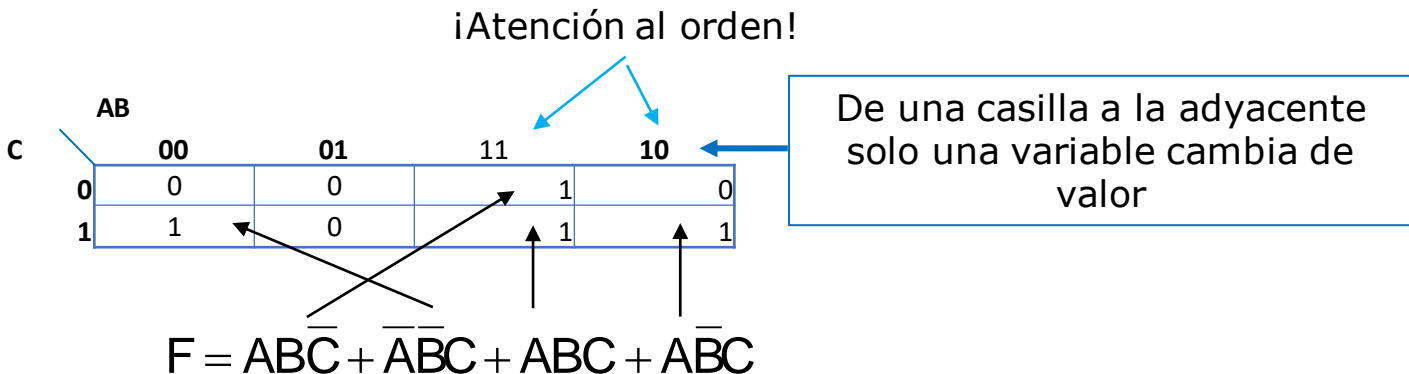
Aquí no hay ningún término común, luego no se puede simplificar

Estos términos, desde el punto de vista de Karnaugh, NO SON ADYACENTES, no se pueden agrupar



Diagramas de Karnaugh

- Con 3 variables:



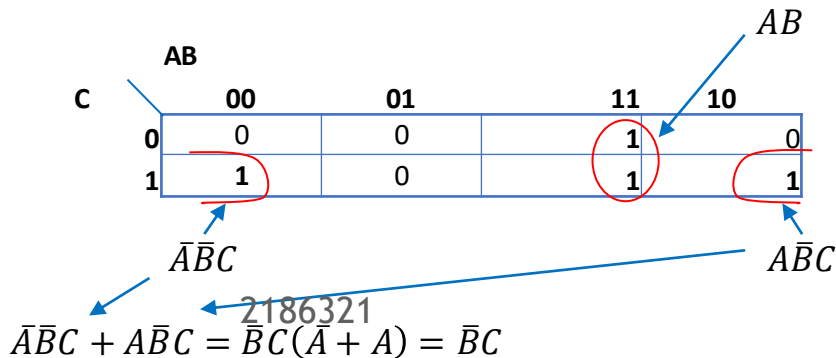
- Se combinan casillas **adyacentes** de valor 1 y se eliminan las variables cuyo valor cambie de unas a otras casillas.
- Los "1" se agrupan en potencias de 2 (**agrupaciones de 2, 4, 8, ...**).
- Por comodidad los ceros normalmente no se escriben.



Diagramas de Karnaugh

- Con 3 variables:

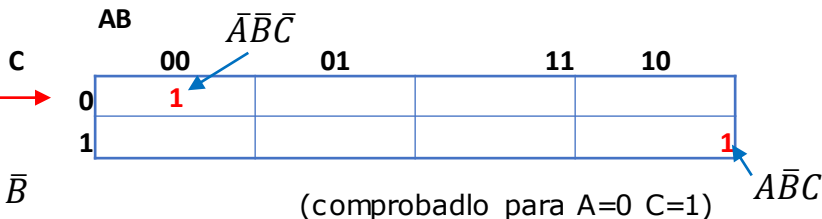
Estos ya sabemos que son adyacentes



Hay un factor común, $\bar{B}C$, y puedo simplificar → son adyacentes

(OJO). Estos **NO** son adyacentes, hay algo común pero no puedo simplificar

$$F' = \bar{A}\bar{B}\bar{C} + A\bar{B}C = \bar{B}(\bar{A}\bar{C} + AC) \neq \bar{B}$$



pues $\bar{A}\bar{C} + AC = \bar{A} XOR C$ la confusión suele venir de aquí → $\overline{\bar{A}C} = \bar{A} + \bar{C} \neq \bar{A}\bar{C}$

(Teorema de De Morgan)



Problema 3

¿Qué funciones son?
Simplificarlas

	PQ			
	00	01	11	10
R				
0	0	1	1	0
1	0	0	1	0

Ej. 3.1

	PQ			
	00	01	11	10
R				
0	1	1	1	1
1	0	0	0	0

Ej. 3.2

	PQ			
	00	01	11	10
R				
0	0	1	1	1
1	0	1	1	0

Ej. 3.3

	PQ			
	00	01	11	10
R				
0	1	0	0	1
1	1	0	0	1

Ej. 3.4

	PQ			
	00	01	11	10
R				
0	0	1	1	0
1	0	1	1	1

Ej. 3.5

	PQ			
	00	01	11	10
R				
0	1	0	0	1
1	1	0	0	0

Ej. 3.6



Problema 3

Este 1 lo cojo 2 veces, es correcto, me permite simplificar al máximo (hay que coger todos los 1 al menos una vez)

	PQ			
	00	01	11	10
R				
0	0	1	1	0
1	0	0	1	0

$$F_{3.1} = \overline{P}Q\overline{R} + P\overline{Q}\overline{R} + PQR$$

Ej. 3.1

simplifico



$$F = Q \cdot \overline{R} + P \cdot Q$$

	PQ			
	00	01	11	10
R				
0	1	1	1	1
1	0	0	0	0

$$F_{3.2} = \overline{A}BC + \overline{A}B\overline{C} + A\overline{B}C + A\overline{B}\overline{C}$$

Ej. 3.2



$$F = \overline{R}$$

	PQ			
	00	01	11	10
R				
0	0	1	1	1
1	0	1	1	0

$$F_{3.3} = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + A\overline{B}C + ABC$$

Ej. 3.3



$$F = Q + P \cdot \overline{R}$$

Este 1 lo cojo 2 veces, es correcto, me permite simplificar al máximo (hay que coger todos los 1 al menos una vez)



Problema 3

Hay un factor común y puedo simplificar → las 4 son adyacentes

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C = \bar{A}\bar{B}(\bar{C} + C) + A\bar{B}(\bar{C} + C) = \bar{B}(\bar{A} + A) = \bar{B}$$

PQ

R	00	01	11	10
0	1	0	0	1
1	1	0	0	1

simplifico

$$F = \bar{Q}$$

Ej. 3.4

$$F_{3.4} = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C$$

PQ

R	00	01	11	10
0	0	1	1	0
1	0	1	1	1



$$F = Q + P.R$$

Ej. 3.5

$$F_{3.5} = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C$$

PQ

R	00	01	11	10
0	1	0	0	1
1	1	0	0	0



$$F = \bar{Q}.\bar{R} + \bar{P}.\bar{Q}$$

Ej. 3.6

$$F_{3.6} = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C}$$



Diagramas de Karnaugh

- Con 4 variables:

$$F = B.C + \bar{B}.\bar{C}$$

	AB			
CD	00	01	11	10
00	1			1
01	1			1
11		1	1	
10		1	1	



	AB			
CD	00	01	11	10
00	1			1
01	1			1
11		1	1	
10		1	1	

Para más variables el método se hace más incómodo -> algoritmo de Quine-McCluskey



Problema 4

Simplificar las funciones

AB

CD	00	01	11	10
00	1	1	1	
01			1	
11		1	1	
10	1		1	

Ej. 4.1

AB

CD	00	01	11	10
00	1			1
01				
11		1		
10	1			1

Ej. 4.2



Problema 4

AB

CD	00	01	11	10
00	1	1	1	
01			1	
11		1	1	
10	1		1	

Ej. 4.1

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

AB

CD	00	01	11	10
00	1			1
01				
11		1		
10	1			1

Ej. 4.2

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

Hay un factor común en las esquinas y puedo simplificar → las 4 son adyacentes

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

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



Resumen de lo que se puede agrupar en Karnaugh

AB

CD	00	01	11	10
00	1	1	1	1
01			1	
11		1		
10		1		1

Grupos de 2.

Horizontales, verticales, compartiendo o no 1's, entre los lados de la tabla, también en horizontal o en vertical

Grupos de 4.

Horizontales, verticales, compartiendo o no 1's, entre los lados de la tabla, también en horizontal o en vertical

AB

CD	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11		1	1	
10		1	1	

Grupos de 8.

Horizontales, verticales, compartiendo o no 1's, entre los lados de la tabla, también en horizontal o en vertical

AB

CD	00	01	11	10
00	1	1	1	1
01	1			1
11	1			1
10	1	1	1	1

Y de 16...

AB

CD	00	01	11	10
00	1	1	1	1
01		1	1	
11		1	1	
10	1			1

AB

CD	00	01	11	10
00	1	1	1	1
01				
11				
10	1			1

AB

CD	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

Resumen de lo que NO se puede agrupar en Karnaugh

AB

CD	00	01	11	10
00	1			1
01	NO		1	NO
11				
10				NO 1

Grupos de 2, 4

En diagonal, entre esquinas a diferente altura

AB

CD	00	01	11	10
00	1			
01	NO	1		
11			1	
10				1

Grupos de 3, 5, 6, 7, 9...

Horizontales, verticales, compartiendo o no 1's, entre los lados de la tabla, también en horizontal o en vertical

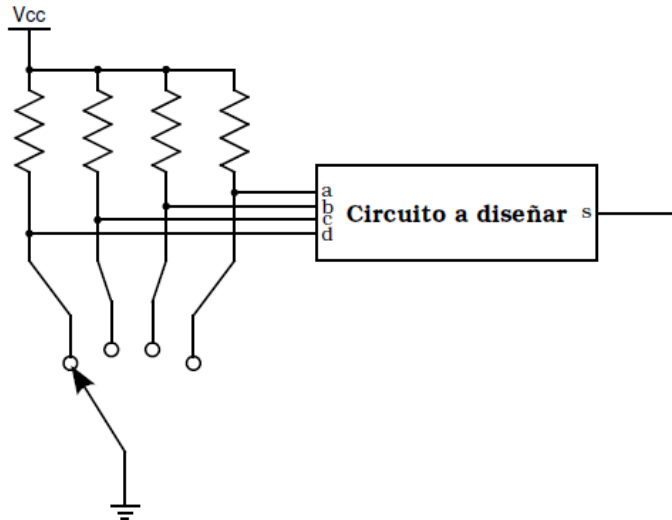
AB

CD	00	01	11	10
00		1	1	1
01		1	1	1
11	NO	1	1	1
10		1	1	1

AB

CD	00	01	11	10
00		1	1	
01		1	1	
11		1	1	
10				

Problema 5: condición “don’t care”



- ¿Qué valores pueden tomar cada una de las entradas a, b, c y d del circuito anterior?
solo una de las variables puede ser 0
- ¿Y qué valores puede tomar la entrada abcd?
- ¿Qué pasa si la función $s=1$ para $abcd=0001$? ¿y si $s=0$?

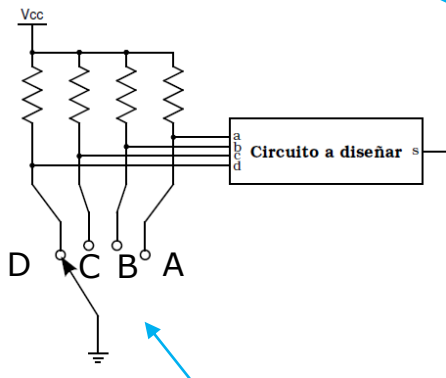
$abcd=0001$ nunca puede pasar, así que nos da lo mismo poner $s=1$ o $s=0$



Problema 5: condición "don't care"

La condición "don't care" es una que nunca puede pasar, de modo que en la tabla podemos decidir si la salida que le corresponde es 1 o 0, según nos convenga para simplificar al máximo

Esta combinaciones nunca pueden pasar, luego la salida es X (lo que yo quiera)



Solo una de las variables puede ser 0 en cada momento, el resto deben de ser 1

(a) Tabla de verdad

a	b	c	d	S
0	0	0	0	X
0	0	0	1	X
0	0	1	0	X
0	0	1	1	X
0	1	0	0	X
0	1	0	1	X
0	1	1	0	X
0	1	1	1	1
1	0	0	0	X
1	0	0	1	X
1	0	1	0	X
1	0	1	1	1
1	1	0	0	X
1	1	0	1	0
1	1	1	0	0
1	1	1	1	X

$$s = \bar{a} \cdot b \cdot c \cdot d + a \cdot \bar{b} \cdot c \cdot d$$

(b) Diagrama de Karnaugh

cd \ ab	00	01	11	10
00	X ₀	X ₁	X ₃	X ₂
01	X ₄	X ₅	1 ₇	X ₆
11	X ₁₂	0 ₁₃	X ₁₅	0 ₁₄
10	X ₈	X ₉	1 ₁₁	X ₁₀

$$s = c \cdot d$$

Las X que están en esta columna decido que son 1, para simplificar los dos 1's "verdaderos" que tengo



Condición “don’t care”

- Todas las casillas con 1 deben tomarse al menos una vez, y pueden/deben tomarse tantas veces cuanto sea necesario para simplificar lo más posible ($A + \bar{A} = 1$).
- Pero como hemos visto, con frecuencia hay combinaciones de entradas que no puedan llegar a producirse.
- En esos casos da igual el valor que tome la función.
- Se toma el que permita una mayor simplificación de la función final (**pero una vez escogido es único**, o bien 0 o bien 1, con objeto de coger al menos una vez todos los 1).



Problema 6

- 3 pulsadores de entrada, A, B y C.
- Solamente 2 entradas pueden estar pulsadas simultáneamente.
- Diseñar un circuito para detectar A=OFF, B=ON y C=ON

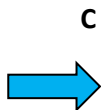


Problema 6

NOTA: no nos lo dicen con claridad...→ se supone que al pulsar ponemos un "1"

- 3 pulsadores de entrada, A, B y C.
- Solamente 2 entradas pueden estar pulsadas simultáneamente.
- Diseñar un circuito para detectar A=OFF, B=ON y C=ON

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	X



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

$$F = B.C \text{ (en vez de } \bar{A}.B.C \text{)}$$



Ejemplo de simplificación mediante Karnaugh (problema 7)

- Diseñar un circuito cuya entrada sea un número del 0 al 9 codificado en BCD, y su salida sea 1 si la entrada es divisible por 3, y 0 en caso contrario (se supondrá en este caso que 0 no es divisible por 3)

Primero planteamos la tabla de verdad

entrada, código BCD					
	A	B	C	D	S
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
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

Después el diagrama de Karnaugh

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

Y finalmente la función simplificada

$$S = AD + \bar{B}CD + BC\bar{D}$$

Estas combinaciones no son posibles en BCD, luego la salida es X, "don't care"

Otro ejemplo de simplificación (problema 8)

- Diseñar un comparador de dos números binarios A y B de 2 bits, con 3 salidas X1, X2 y X3, tales que:

- X1=1 si A>B
- X2=1 si A<B
- X3=1 si A=B

Primero planteamos la tabla de verdad para X1=1 si A>B

nº A, 2 bits nº B, 2 bits

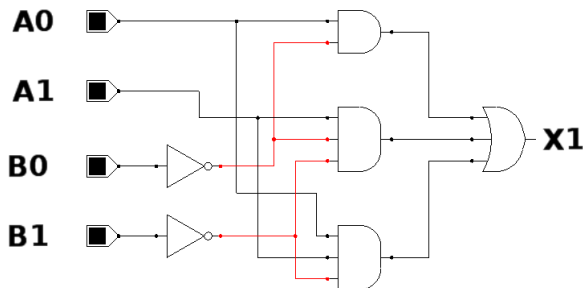
A0	A1	B0	B1	X1
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Después el diagrama de Karnaugh

		B0 B1			
X1	A0 A1	00	01	11	10
	00	0	0	0	0
	01	1	0	0	0
	11	1	1	0	1
	10	1	1	0	0

Y finalmente la función simplificada y el circuito

$$X1 = A0\overline{B0} + A1\overline{B0}\overline{B1} + A0A1\overline{B1}$$



Simplificación mediante Karnaugh (problema 8)

Podemos repetir todo el proceso, poniendo las tablas completas (en la siguiente página se muestran), pero es un rollo, es fácil rellenar directamente el diagrama de Karnaugh

$X2=1$ si $A<B$

X2		B0 B1			
A0	A1	00	01	11	10
00		0	1	1	1
01		0	0	1	1
11		0	0	0	0
10		0	0	1	0

$$X2 = \overline{A}0B0 + \overline{A}0A1B1 + \overline{A}1B0B1$$

Esta no hay quien la simplifique por Karnaugh

$X3=1$ si $A=B$

A0 A1		B0 B1			
		00	01	11	10
00		1	0	0	0
01		0	1	0	0
11		0	0	1	0
10		0	0	0	1

$$X3 = \overline{A}0A1B0B1 + \overline{A}0A1\overline{B}0B1 + A0A1B0B1 + A0\overline{A}1B0\overline{B}1$$

Pero hay un truco para simplificar, haciendo una función de una función, ved la siguiente hoja

Simplificación mediante Karnaugh (problema 8)

Tablas completas

				A>B	A<B	A=B
A0	A1	B0	B1	X1	X2	X3
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	1
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	0	1

obtengo X3 como función de X1 y X2
($X3 = \overline{X1} \cdot \overline{X2}$)

		X1	$\overline{X1}$	$\overline{X2}$
X2	0	0	1	1
	1	1	0	X