

Contents 1..... **¡Error! Marcador no definido.**

Tema 1: Sistemas de numeración2

Sistema decimal2

Sistema binario2

BIN → DEC2

DEC → BIN2

Sistema hexadecimal3

HEX → BIN4

BIN → HEX4

HEX → DEC4

DEC→HEX4

Tema 2: Sistemas electrónicos digitales5

Fundamentos del algebra de Boole5

Variables:5

Operaciones:5

Propiedades7

Puertas lógicas7

Circuitos combinacionales9

Introducción9

Implementaciones de las funciones booleanas9

Multiplexores (MUX)/Demultiplexor (DMUX)16

Decodificadores/Codificadores (DEC)17

Arrays lógicos programables (PLA)19

Memorias de solo lectura (ROM)19

Sumadores20

Circuitos secuenciales20

Tema 1: Sistemas de numeración

Sistema decimal

Usa 10 dígitos, 0,1,.....,9

Sistema de base 10, lo que implica que cada dígito se multiplica por un peso que es una potencia de la base: la que corresponde a la posición de ese dígito.

-Ejemplo:

$$478 = 4 \cdot 10^2 + 7 \cdot 10^1 + 8 \cdot 10^0$$

210 la posición y el exponente.

$$24.37 = 2 \cdot 10^1 + 4 \cdot 10^0 + 3 \cdot 10^{-1} + 7 \cdot 10^{-2}$$

10-1-2 la posición respectivamente

Sistema binario

Usa dos dígitos: 0,1. Dígitos binarios se llaman "BITS" (Binary digiT)

Base 2 → El peso asignado a cada dígito será la potencia de 2 correspondiente a su posición de ese dígito.

Posición: 7 6 5 4 3 2 1 0, -1 -2 -3.....

Peso 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0 2^{-1} 2^{-2} 2^{-3}

-Ejemplos:

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

$$= 8 + 4 + 0 + 1$$

$$= 13_d$$

$$11001,101_b =$$

$$= 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-3}$$

$$= 25.625_d$$

BIN → DEC

- Sumar los pesos de las posiciones en las que aparecen "1"
- Como en los ejemplos anteriores

DEC → BIN

Procedimiento rápido

- Averiguar los pesos que deben sumarse (todos ellos potencias de 2) para obtener el nº decimal a convertir

• **Ejemplo:**

$$45_d = _ _ 1 _ 1 1 _ 1_b$$

$$= 2^5 + 2^3 + 2^2 + 2^0$$

$$23.75_d = _ _ _ 1 _ 1 1 1, 1 1 _ _ _b$$

Procedimiento Sistemático (el de toda la vida)

- Se separa la parte entera de la parte fraccionaria:
 - Parte entera:
Vamos dividiendo entre 2 repetidamente y nos quedamos con los restos en orden inverso al de aparición
 - Parte fraccionaria:
Vamos multiplicando repetidamente por 2, y nos vamos quedando con las partes enteras de los resultados.
En caso de que sean necesarios un umero infinito de dígitos binarios para expresar una parte fraccionaria que en decimal puede expresarse con un número finito de dígitos. En caso de que pase eso tomaríamos solo el numero de dígitos binarios necesarios para obtener la precisión requerida.

-Ejemplo: 23.75

Parte entera 23:

$$23/2=11 \text{ resto } 1, 11/2=5 \text{ resto } 1, 5/2=2 \text{ resto } 1, 2/1=2 \text{ resto } 0, 1/2=0 \text{ resto } 1$$

$$23_d = 10111_b$$

Parte fraccionaria 0.75:

$$0.75 * 2 = 1.5$$

$$0.5 * 2 = 1.0$$

$$0.75_d = 0.11_b$$

$$23.75_d = 10111.11_b$$

-Ejemplo: 0.81 d

$$0.81 * 2 = 1.62$$

$$0.62 * 2 = 1.24$$

$$0.24 * 2 = 0.48$$

$$0.48 * 2 = 0.96$$

$$0.96 * 2 = 1.92$$

$$0.92 * 2 = 1.84$$

$$0.110011 \dots_b$$

Sistema hexadecimal

Se emplea como método compacto de expresar cantidades binarias, puesto que la conversión hexadecimal y binario es inmediata.

16 dígitos 0 → 9 y A B C D E F

Base 16

HEX → BIN

Sustituir cada dígito hexadecimal por su correspondiente combinación de 4 bits

F2A,1C h = [1111][0011][1010],[0001][1100] b

BIN → HEX

Agrupar los bits de 4 en 4 comenzando por la coma.

-Ejemplo 101101,10001

Parte anterior a la coma: [0010] [1101], [1000] [1000] → 2D,88 h

HEX → DEC

Multiplicar cada dígito por su peso (potencia de 16 correspondiente a su posición)

-Ejemplo A4,D1 h

$=10 \cdot 16^1 + 4 \cdot 16^0 + 13 \cdot 16^{-1} + 1 \cdot 16^{-2} = 164.81640625$ d

DEC → HEX

Parte entera: divisiones por 16

Parte fraccionaria: multiplica por 16

Ejemplo:

164.81640625 d =

Parte entera: $164/16 = 10$ resto 4, $10/16 = 0$ resto 10 → A4

Parte fraccionaria: $.81640625 \cdot 16 = 13.0625$

$.0625 \cdot 16 = 1.0$

13=D, por tanto 0, D1

A4,D1 h

Tema 2: Sistemas electrónicos digitales

Fundamentos del algebra de Boole

El algebra de Boole es la herramienta que nos va a permitir el análisis y el diseño de los circuitos electrónicos digitales.

Veamos las variables, operaciones y las propiedades del algebra de Boole.

Variables:

Son variables binarias: pueden tomar 2 valores ,0,1 aunque también se puede llamarse verdadero y falso siendo 0 falso y 1 verdaderos. También puede considerarse el 0 como el nivel bajo de voltaje (0V) y el 1 como el nivel alto(5V,3.3V) cuando estamos hablando en el ámbito de electrónica.

Operaciones:

- Operaciones básicas:

1) **Complemento, Inversion: NOT**, se representa como A'

$A=0$

$A'=1$

2) **Suma Logica, OR, +**

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

3) **Producto lógico, AND, ·**

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

-NOTAS-

1) ¿Cómo se interpreta una expresión que incluye estas operaciones?

EJ: $F=A+(B'·C)$

$1=1+ \underline{\quad\quad}$ ($A=1$)

$1= \underline{\quad\quad} +0'·1$ ($B=0,C=1$)

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

$$2) A+(B' \cdot C) =$$

$$A+B' \cdot C =$$

$$A+B' \cdot C$$

4) Otras operaciones:

OR Exclusiva, XOR (exclusive OR) \oplus

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

La operación XOR puede emplearse como:

1) Comparador:

Si A, B iguales, XOR=0

Si A, B distintos, XOR=1

2) Inversor programable:

$$A \oplus 0 \rightarrow A$$

$$1 \rightarrow A'$$

Es como una puerta NOT que solo invierte cuando en el segundo operando ponemos un 1.

3) Generador de paridad par:

$$1 \oplus 0 \oplus 1 \oplus 1 = 1$$

Ya que entre los operandos y el resultado siempre habrá un numero par de unos, por lo tanto, tiene paridad par.

4) Generador de paridad impar:

$$1 \oplus 0 \oplus 1 \oplus 1 = 1$$

Si entre los operandos detecta un numero impar de unos activa la alarma

NOR (NOT OR): $A' + B'$

A	B	A+B	$(A+B)'$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

NAND (NOT AND) $(A \cdot B)'$

A	B	$A \cdot B$	$(A \cdot B)'$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

XNOR (NOT XOR) $(A \oplus B)'$

A	B	$A \oplus B$	$(A \oplus B)'$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

-NOTA:

$$(A+B)' \neq A'+B'$$

$$(A \cdot B)' \neq A' \cdot B'$$

Propiedades

- Postulados básicos:

Ley conmutativa:

$$A+B=B+A$$

$$A \cdot B=B \cdot A$$

Ley distributiva:

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

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

Elemento neutro:

$$A+0=A$$

$$A \cdot 1=A$$

Elemento complementario:

$$A+A'=1$$

$$A \cdot A'=0$$

- Otras identidades:

$$A+1=1$$

$$A \cdot 0=0$$

$$A+A=A$$

$$A \cdot A=A$$

Ley asociativa:

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

$$A \cdot (B \cdot C)=(A \cdot B) \cdot C$$

Teorema de Morgan:

$$(A+B)'=A' \cdot B'$$

$$(A \cdot B)'=A'+B'$$

Puertas lógicas

Son los circuitos electrónicos que implementan las operaciones del algebra de Boole:

NOT:



OR:



AND:



NOR:



NAND:



XNOR:



Salvo para la puerta NOT también existen puertas de más de 2 entradas.

Llamamos *Retardo de puerta* al tiempo que transcurre desde que se presentan los valores de las entradas hasta que se actualiza el valor de la salida. Este valor es muy pequeño para una sola puerta, pero cuando se interconectan varios niveles de puertas, pero cuando se suman los retardos, puede llegar a causar problemas.

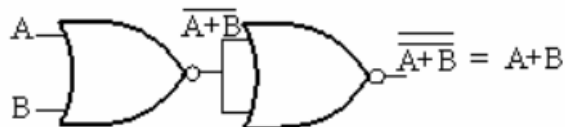
Conjuntos de puertas funcionalmente completos: Se puede implementar cualquier función del Algebra de Boole empleando exclusivamente puertas de un conjunto que sea funcionalmente completo. Los conjuntos funcionalmente completos son los formados por puertas:

- 1) NOT, OR, AND (operaciones basicas)
- 2) NOT, OR Para implementar la operación AND usando solo NOT y OR:
 $(A \cdot B)'' = (A' + B')'$
- 3) NOT, AND Para implementar la operación OR usando solo NOT y AND:
 $(A + B)'' = (A' \cdot B')'$
- 4) NOR Universalidad de las puertas NOR, se puede usar la puerta NOR para implementar las puertas NOT, AND y OR.

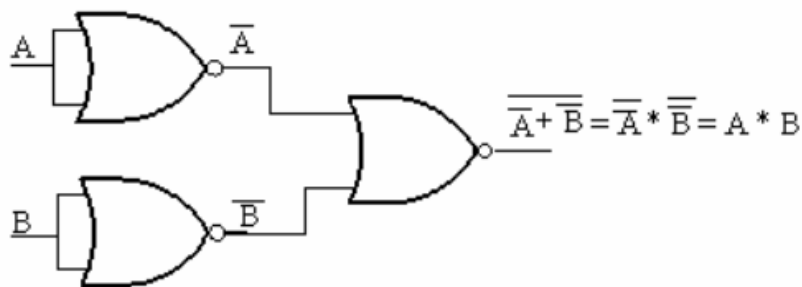
4.1) NOT: $A' = (A + A)'$



4.2) OR: $A + B = (A + B)''$

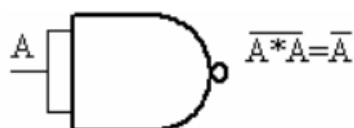


4.3) AND: $(A' + B')' = A'' \cdot B'' = A \cdot B$

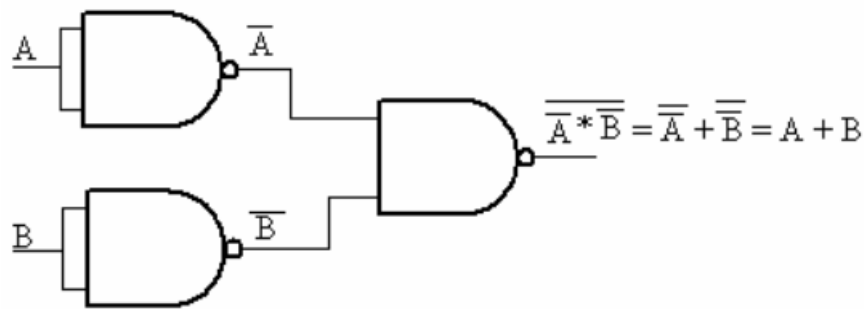


- 5) NAND: Universalidad de las puertas NAND

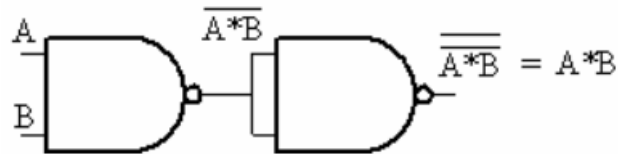
5.1) NOT:



5.2) OR:



5.3) AND:



-NOTA: Cuando tengamos que expresar una función booleana cualquiera solo con puertas NAND o NOR, el procedimiento a seguir será:

- 1) Transformar las operaciones no permitidas (+ en NAND, · en NOR) en la otra, mediante la aplicación del Tº de DeMorgan. Si las operaciones a transformar no están negadas (como requiere DeMorgan) se niegan 2 veces.
- 2) Comprobar que todas las operaciones han quedado negadas (NOR: +'; NAND ·') Si alguna no lo esta → negar 2 veces.

Estos 2 pasos se hacen sobre la expresión y luego se dibuja el circuito a partir de la expresión transformada.

Circuitos combinacionales

Introducción

- Diferencias circuitos combinaciones/secuenciales (Transparencia)
- 3 formas de definición de los circuitos combinacionales

N entradas (I_0, \dots, I_{n-1}) → Circuito combinacional → F_0, \dots, F_{n-1}

- 1) Tabla de verdad

$I_{n-1} \dots I_0$	$F_{m-1} \dots F_0$
0 0
0 1
.....
1 1

- 2) Circuito

Interconexion de puertas logicas

- 3) Expresiones, ecuaciones, funciones | Algebraicas, Booleanas

Implementaciones de las funciones booleanas

Expresiones SOP y POS

Vamos a hacer lo siguiente:

Tabla de verdad → Expresiones algebraicas → Circuito

Suma de productos (SOP)

Producto de sumas (POS)

Con este ejemplo:

Para identificar cómodamente cada combinación de entradas decimal	Variables de entrada			Variables de Salida
	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

SOP: F será cierta siempre que se de alguna de las combinaciones que la hace cierta.

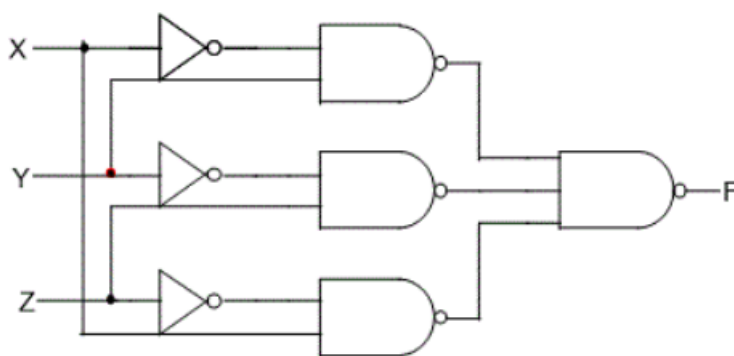
$$F = (A' \cdot B \cdot C') + (A' \cdot B \cdot C) + (A \cdot B \cdot C')$$

En notación compacta:

$$F = \sum_3(2,3,6) \text{ siendo 3, el número de variables de entrada (A, B, C) en este caso} + \sum_0()$$

Términos

indiferentes



POS: F será cierta si no se da ninguna de las combinaciones que la hacen falsa:

$$F = (A' \cdot B' \cdot C')' \cdot (A' \cdot B \cdot C)' \cdot (A \cdot B' \cdot C')' \cdot (A \cdot B \cdot C)' \cdot (A \cdot B \cdot C)' = \text{Aplicando DeMorgan:}$$

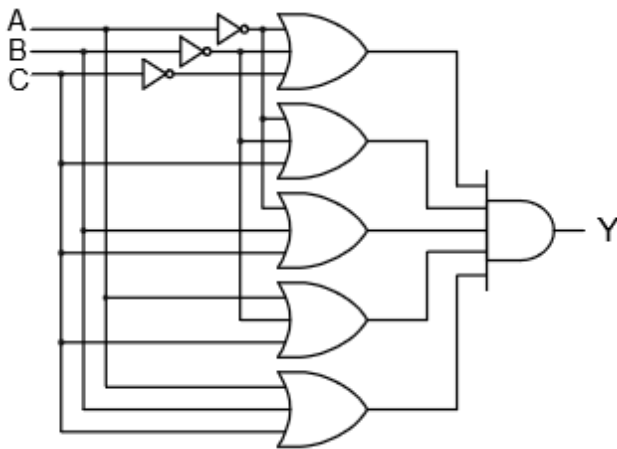
$$= (A + B + C) \cdot (A + B + C') \cdot (A' + B + C) \cdot (A' + B + C') \cdot (A' + B' + C')$$

Formato compacto:

$$F = \pi_3(0', 1', 4', 5', 7') \cdot \prod_0()$$

Los números que contienen un 0 en la salida de la tabla de verdad.

Términos indiferentes



Conclusiones:

1) Expresión SOP:

- a. Un término por cada "1" en la tabla de verdad
- b. En cada termino: las variables a 0 son negadas y las variables a 1 sin negar.

2) Expresión POS:

- a. Un termino por cada "0" en la tabla de verdad
- b. En cada termino: las variables a 0 son sin negar y las variables a 1 son negadas.

Simplificación

Mapas Karnaugh

Procedimiento sistemático que permite simplificar cómodamente función de hasta 4 variables de entrada

- A) Construcción del Mapa:
2 variables de entrada

AB

00	01	11	10

3 variables de entrada

A\BC	00	01	11	10
0				
1				

4 variables de entrada (2^4 combinaciones)

AB\BC	00	01	11	10
00				
01				
11				
10				

- B) Relleno del mapa:

2 formas habituales:

- 1) Tabla de verdad → Mapa Karnaugh → Expresión algebraica simplificada
- 2) Expresión algebraica no simplificada → forma canónica (Aunque este paso acabara siendo prescindible) → Mapa Karnaugh → Expresión algebraica simplificada

Ejemplo:

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

A\BC	00	01	11	10
0			1	1
1				1

Ejemplo 2: $F=A'B+BC'$

Se pasa a forma canónica (todas las variables aparecen en todos los términos)

$$=A'B \cdot (C+C') + (A+A') \cdot BC'$$

$$=A'BC + A'BC' + ABC' + A'BC'$$

$$=A'BC + A'BC' + ABC'$$

A\BC	00	01	11	10
0			1	1
1				1

C) Numeración de las casillas

Vamos a numerar las casillas del mapa de Karnaugh para que nos resulta más ágil rellenarlo.

Se muestra a continuación como numerar el mapa para 4 variables de entrada. Para 2 o 3 variables de entrada bastaría con coger solo una parte del Mapa de 4 variables de entrada.

MSB: Most significant bit, bit mas significativo

LSB: least significant bit, bit menos significativo

Tabla de verdad:

Dec	A (MSB)	B	C	D	F
0	0	0	0	0	
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	
.....
12	1	1	0	0	1
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	1

Mapa de Karnaugh:

AB\CD	00	01	11	10
00	0	1 1	3	2 1
01	4	5	7	6
11	12 1	13	15 1	14
10	8	9	11	10

D) Principio de simplificación

-Ejemplo:

$$ABCD + ABC'D$$

$$ABD(C+C')$$

$$ABD$$

$$ABCD' + ABC'D'$$

$$ABD'(C+C')$$

$$ABD'$$

Y por eso:

$$ABD + ABD'$$

$$AB(D+D')$$

$$ABD$$

Cuando entre 2 términos solo cambia el estado de 1 variable, se simplifica esa variable (queda un término simplificado compuesto por las variables que tenían el mismo estado en ambos términos).

Esa situación en un mapa de Karnaugh corresponde a dos casillas adyacentes

Por lo tanto, la simplificación consistirá en localizar grupos de unos adyacentes, y generar los términos simplificados correspondientes.

Un grupo de dos casillas → simplifica 1 variable.

Un grupo de cuatro casillas → simplifica 2 variables

Un grupo de ocho casillas → simplifica 3 variables

Un grupo de dieciséis casillas → simplifica 4 variables → $F=1$

E) Procedimiento de simplificación

Hacer grupos de unos según las siguientes directrices

- 1) Grupos que sean lo más grandes posibles (así se simplifican más variables)
- 2) Cuantos menos grupos mejor (menos términos en la expresión simplificada)
- 3) Grupos de 2 casillas (2,4,8,16)
- 4) Todos los unos deben quedar englobados (un uno aislado constituye un grupo de una casilla, en la que no se puede simplificar nada)
- 5) Los unos pueden englobarse en varios grupos, si con ello se logran grupos más grandes (directriz 1)
- 6) Comenzar por los unos que solo pueden englobarse de una forma. De este modo, irán determinando como englobar los unos que ofrecen distintas alternativas.

Cuando una función no esta definida para algunas combinaciones de las variables de entrada, realmente nos da igual considerar que la función va a ser 0 o 1 en esos casos, porque no deben suceder nunca.

A esta combinación los llamamos **Términos indiferentes**, las rellenamos con “x” en el mapa de Karnaugh y las consideramos como comodines: se pueden englobar junto con los unos (si logramos con ello grupos mayores), pero también pueden quedarse sin coger.

Tablas Quine McCluskey

Es otro método sistemático de simplificación de funciones booleanas

Es valido para cualquier nº de variables de entrada.

No resulta como de aplicar manualmente. Esta pensado fundamentalmente para ser implementado como programa informático.

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

Veámoslo con un ejemplo:

$$F = ABCD + ABC'D + ABC'D' + AB'CD + A'BCD + A'BC'D + A'B'CD$$

Para entender lo que estamos haciendo dibujamos el mapa de Karnaugh:

FASE 1: Buscar todas las agrupaciones posibles.

Primero ordenamos los términos en función del número de variables sin negar.

Numero de variables sin negar (nº unos)	-
0	$A'B'C'D$
1	$ABC'D'$ $A'BCD'$ $A'BC'D$
2	$ABC'D$ $AB'CD$ $A'BCD$
3	$ABC'D$ $AB'CD$ $A'BCD$
4	$ABCD$

A continuación, vamos comparando todos los términos de un apartado con todos los del siguiente, generando términos simplificados cuando solo cambie el estado de 1 variable. Los términos originales que se combinan para dar lugar a términos simplificados se van dejando marcados.

$A'C'D$
ABC'
$A'BC$
$BC'D$
$A'CD$
ABD
ACD
BCD

Volvemos a simplificar con el mismo método:

BD

FASE 2: Detectar las agrupaciones que son imprescindibles.

Hacemos la siguiente tabla:

	ABCD	ABC'D	ABC'D'	AB'CD	A'BCD	A'BCD'	A'BC'D	A'B'CD
A'C'D							X □	X O
ABC'		X □	X O					
A'BC					X □	X O		
ACD	X □			X O				
BD	X	X			X		X	

Expresión simplificada:

$$F=A'C'D+A'BC+ACD+ABC'$$

Si nos siguieran quedando columnas sin marcar, tendríamos que seguir añadiendo términos simplificados de modo que todas las columnas quedan finalmente cogidas, y hacerlo mediante los términos simplificados que sean menos y más sencillos.

- Términos indiferentes:

FASE 1: Se tienen en cuenta, para generar todas las agrupaciones que sean posibles.

FASE 2: No se consideran (no se ponen en las columnas), porque la información que contienen no es imprescindible en la expresión final.

[Simplificación Algebraica](#)

-Ver ejercicios (8, hoja resuelta)

[Implementaciones NAND/NOR](#)

-Ver ejercicio 7

Bloques funcionales combinacionales (Resto del apartado de circuitos combinacionales)

[Multiplexores \(MUX\)/Demultiplexor \(DMUX\)](#)

-Funcion: [HOJA APARTE]

-Aplicación:

-Los MUX se emplean para permitir encaminar hacia un solo destino varios posibles fuentes de datos:

[DIAGRAMA]

-Si la información consta de n bits, habrá que usar n MUX.

Ejemplo:

-C |

-IR | → Registro de 8 bits → Necesitamos 8 MUX de 4 a 1

-ALU |

[DIAGRAMA HOJA]

Implementación de cualquier función booleana mediante MUX:

- Caso 1: Nº de variables de entrada de la función a implementar = Nº de entradas de selección del MUX

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

- Caso 2: Nº de variables de entrada de la función a implementar > Nº entradas de selección del MUX

Tomamos 2 de las variables de entrada (por Ej., A y B) y las conectamos a las entradas de selección de un MUX 4 a 1:

A	B	C	F
0	0	0	1
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	0

Decodificadores/Codificadores (DEC)
[esquema]

-Modo de funcionamiento:

A(msb)	B	C	
0	0	0	D0=1, resto=0
0	0	1	D1=1, resto=0
0	1	0	D2=1, resto=0

0	1	1	D3=1, resto=0
1	0	0	D4=1, resto=0
1	0	1	D5=1, resto=0
1	1	0	D6=1, resto=0
1	1	1	D7=1, resto=0

Siempre habrá una salida (y solo una) activa: la que corresponde al nº codificado en binario presente en la entrada.

-Equaciones:

$$D0=A'B'C'$$

$$D1=A'B'C$$

$$D2=A'BC'$$

$$D3=A'BC$$

$$D4=AB'C'$$

$$D5=AB'C$$

$$D6=ABC'$$

$$D7=ABC$$

Circuito: [hoja]

-Aplicación de los decodificadores:

Asociación de dispositivos pequeños para formar dispositivos más grandes.

-Ejemplo: Crear una memoria de 1kbyte mediante la asociación de 4 memorias de 256 bytes.

Diagrama de la memoria [hoja]

(También hay 8 líneas de datos(D7,...,D0), para poder leer o escribir el dato en la posición especificada mediante las líneas de dirección)

-Memoria de $1024 = 2^{10} = 1K$ Bytes

-Mapa de direcciones: [esquema]

-Implementación: [esquema]

-En los dispositivos electrónicos puede aparecer una entrada HABILITACION:

*Si es 0: dispositivo inhabilitado (no se permite su funcionamiento)

*Si es 1: dispositivo habilitado

- A esa entrada se la identifica como:

E: Enable

CE: Chip Enable

CS: Chip select

Importante:

En las entradas del decodificador hemos puesto las líneas de dirección MAS SIGNIFICATIVAS.

Codificador: [Diagrama]

Arrays lógicos programables (PLA)

Dispositivo Lógica programable (PLD)

Una vez diseñado un circuito combinacional tenemos varias alternativas para construirlo:

- 1) Interconectado puertas
Inconvenientes:
 - Tenemos que utilizar varios chips (uno o más por cada tipo de puerta)
 - Mayor tamaño
 - Menor velocidad (conexiones cableadas)
- 2) Chip a medida
 - Solo justificaría su elevado coste y tiempo de diseño el hecho de fabricar muchas unidades
- 3) Chip de uso general: PLA o PLD
 - Es un chip que incorpora una matriz regular de puertas , y en el que las conexiones están sin realizar.
 - El usuario, con la ayuda de un programador, fija las conexiones necesarias de modo que el circuito queda implementado.
 - Por ejemplo: ver Fig.
 - A.19: matriz regular de puertas NOT, AND y OR, que permite implementar 2 funciones como SOP.
 - El chip de uso general constituye una solución intermedia a la interconexión con puertas y al chip a medida:
 - Un solo chip
 - Coste y tiempo de diseño moderados
 - Menos tamaño
 - Mas velocidad (no hay conexiones soldadas)

Memorias de solo lectura (ROM)

Read Only Memory

- Circuitos combinacionales: Por contraposición con los circuitos secuenciales, nos referimos a ellos como “circuitos sin memoria”.
- PERO: las memorias ROM (solo lectura) son circuitos combinacionales, porque sus contenidos no varían: siempre que ponemos una determinada dirección →obtenemos el mismo dato.
- Contenidos de una ROM
[diagrama]
 - Permite implementar un conjunto cualquiera de funciones booleanas sin más que grabar los datos necesarios en cada dirección.

-Implementación de una memoria ROM:

-Como circuito combinación que es, puede implementarse por cualquiera de las técnicas que hemos visto: POS, SOP, NAND, NOR, MUX, DECOD ...

-Por ejemplo: mediante DECOD

-Ver FIG A.20, que implementa la ROM definida en la figura A.8

Sumadores

- No confundir SUMADOR ARITMETICO (lo que vamos a ver en este apartado) con SUMADOR LOGICO (o del algebra de Boole), que es la puerta lógica.
- A) [hoja]
- B) [hoja]
- Sumador de 1 bit [diagrama]
- Sumador de Varios bits [diagrama]
- Problema: acumulación de retardos
- Solución:

Técnica de acarreo anticipado:

Para evitar la acumulación de retardos, es necesarios que cada sumador de 1 bit disponga de todas sus entradas desde un principio:

* A_i

* B_i Sin problema (bits de los 2 n°s a sumar)

* $C_i \rightarrow$ ¿Podemos generarlo desde el principio?

$$C_0 = A_0 B_0 + A_0 C_{in} + B_0 C_{in}$$

$$C_1 = A_1 B_1 + A_1 C_0 + B_1 C_0$$

$$C_2 = A_2 B_2 + A_2 C_1 + B_2 C_1$$

$$C_{out} = A_3 B_3 + A_3 C_2 + B_3 C_2$$

-Conclusión: Los C_i pueden calcularse desde el principio a partir de los A_i , B_i , $C_i \rightarrow$ No será necesario que cada sumador tenga que esperar al sumador anterior \rightarrow NO HAY

ACUMULACION DE RETARDOS

El calculo de los C_i solo conlleva 2 niveles de retardo de puerta:

- Operación AND
- Operaciones OR
- [diagrama]

Las expresiones de los C_i cada vez tienen más términos \rightarrow llega un momento en el que no merece la pena aplicar la técnica.

Se suele utilizar acarreo anticipado hasta 4 o 8 bits.

Para sumar números mas largos se encadenan sumadores (como lo visto en el apartado C), pero son sumadores de hasta 8 bits que internamente no acumulan retardos.

Circuitos secuenciales

Biestables

-Características de todos los biestables:

1) 2 estados estables: en ausencia de entrada, mantienen el valor en la salida, sea 0 o 1.

Constituyen una MEMORIA de 1 BIT.

2) Proporcionan en la salida Q (Salida), Q' (salida negada)

S-R asíncrono

-S: set

-R: reset

[diagrama]

Tabla característica: [hoja]

S-R síncrono

-*Comportamiento Asíncrono*: Cuando cambian las entradas, las salidas se actualizan de forma inmediata (salvo los pequeños retardos de puerta)

-*Comportamiento síncrono*: Existe una señal de RELOJ (CLOCK,CLK) que hace que las salidas de distintos circuitos cambien a la vez (de forma sincronizada), en base a los valores que tengan presentes en sus entradas (con antelación).

Biestable S-R síncrono: [Diagrama hoja]

Introducimos una señal CLOCK:

S-R Síncrono:

- Cuando CLOCK=0 (- - -)
 - S Asíncrono = 0 y R Asíncrono=0 → función memoria: $Q_{n+1}=Q_n$ (No cambia la salida)
- Cuando CLOCK=1(- - -)
 - S asíncrono = S síncrono
 - R asíncrono= R síncrono →Evoluciona la salida del biestable

D síncrono

Tabla característica:

D	Q_{n+1}
0	0
1	1

- Cuando CLOCK=0 (- - -)
 - Grabamos un 0 o un 1 en el biestable (poniendo un 0 o un 1 en la entrada D)
- Cuando CLOCK=1(- - -)
 - Se conserva el dato grabado en la salida del biestable aunque se produzcan cambios en el valor de la entrada D.

Por esa razón, se suelen denominar LATCH (cerrojo) a los biestables, y muy particularmente al biestable D

J-K síncrono

Tabla característica:


J	K	Q _{n+1}
0	0	Q _n (MEMORIA)
0	1	0
1	0	1
1	1	Q _n ' (Complementario)

Resumen de Biestables:


1) Formas de sincronización con la señal de reloj

a. Sincronización por nivel

i. Por nivel alto


 → CLK

ii. Por nivel bajo

 → CLK o

b. Sincronización por flanco

i. Por flanco ascendente

 → CLK →

ii. Por flanco descendente

 → CLK → o

2) Símbolos: [hoja]

Registros

Un registro es una memoria de n bits.

Se construye interconectando n biestables

2 tipos:

Registros paralelos:

Los n bits se pueden leer todos a la vez. También se pueden escribir todos a la vez.

Registro de desplazamiento:

La transmisión de los bits es vía SERIE: solo se pueden escribir o leer de uno en uno, y en cada pulso de reloj todos los bits se desplazan una posición.

Registros paralelos

Ejemplo: registro paralelo de 4 bits con biestables D:

[Diagrama]

Cuando LOAD = 0 → las entradas de reloj de todos los biestables están a 0 → no se graba dato

Cuando LOAD=1 → Cuando CLOCK \uparrow entonces los 4 biestables cargan un nuevo dato

Registros de desplazamiento

Por ejemplo: registro de desplazamiento de 4 bits con biestables D:

[hoja]

En cada ciclo de reloj, los bits se desplazan una posición a la derecha.

Contadores

Un contador es un registro, cuyo valor va contando los pulsos de la señal de reloj.

Un contador formado por n biestables es un contador de n bits, que puede contar desde 0 hasta $2^n - 1$.

Existen dos tipos de contadores:

1) Contador asíncrono:

La entrada de reloj de cada biestable es la salida (Q o Q') del biestable anterior → acumulación de retardos (funcionalmente "asíncrono").

2) Contador síncrono:

Las entradas de reloj de todos los biestables están conectadas a una misma señal de reloj → todos ellos cambian de estado exactamente a la vez (funcionamiento "síncrono").

Contador asíncrono

[Figura A.32]: Contador asíncrono de 4 bits

4 biestables

Cuenta desde 0 hasta $2^4 - 1 = 15$ (16 números)

En la realidad, los cambios de estado de los 4 biestables no quedarían perfectamente alineados (acumulación de retardos)

Ampliación nº de bits:

Añadiendo más biestables, según el mismo patrón de conexiones.

Contador síncrono:

Ejemplo: contador síncrono de 3 bits mediante biestables J-K:

- 3 biestables
- Cuenta entre 0 y $2^3 - 1 = 7$

Evolucion de estados:

Estado								
A	B	C	Ja	KA	JB	KB	JC	KC
0	0	0	0	X	0	X	1	X
0	0	1	0	X	1	X	X	1
0	1	0	0	X	X	0	1	X
0	1	1	1	X	X	1	X	1
1	0	0	X	0	0	X	1	X
1	0	1	X	0	1	X	X	1
1	1	0	X	0	X	0	1	X
1	1	1	X	1	X	1	X	1

Circuito: [hoja]

Tabla auxiliar: [hoja]

JA=BC

A\BC	00	01	11	10
0			1	
1	X	X	X	X

KA=BC

A\BC	00	01	11	10
0	X	X	X	X
1			1	

JB=C

A\BC	00	01	11	10
0		1	X	X
1		1	X	X

KB=C

A\BC	00	01	11	10
0	X	X	1	
1	X	X	1	

JC=1

A\BC	00	01	11	10
0	1	X	X	1
1	1	X	X	1

KC=1

A\BC	00	01	11	10
0	X	1	1	X
1	X	1	1	X

Diseño de circuitos secuenciales

[Aspectos generales](#)

[Transparencias]

Ejemplo: Automata de Moore:

Detector de secuencia 1-0-1 (se permiten los solapamientos: un bit que ha formado parte de una secuencia puede formar parte también de la siguiente)

[Circuitos]

Pasos:

- 1) Diagrama de estados
 - 2) Tabla de transición
 - 3) Ecuaciones de excitación: J, K, D
 - 4) Ecuaciones de salida: Z
 - 5) Circuito
- 1) Diagrama de estados: [hoja]
 - 2) Tabla de transición:
 - 3) Ecuaciones de excitación:
 Biestables D:
 Karnaugh [Hoja]
 Biestables J-K:
 Karnaugh [Hoja]

[Ejemplo MOORE](#)

[Ejemplo MEALY](#)