Tema 4

Circuitos Combinacionales

Indice

1. Representación binaria:
    - Representación posicional de magnitudes
    - Códigos binarios
2. Funciones combinacionales
3. Análisis de circuitos combinacionales
4. Diseño de circuitos combinacionales

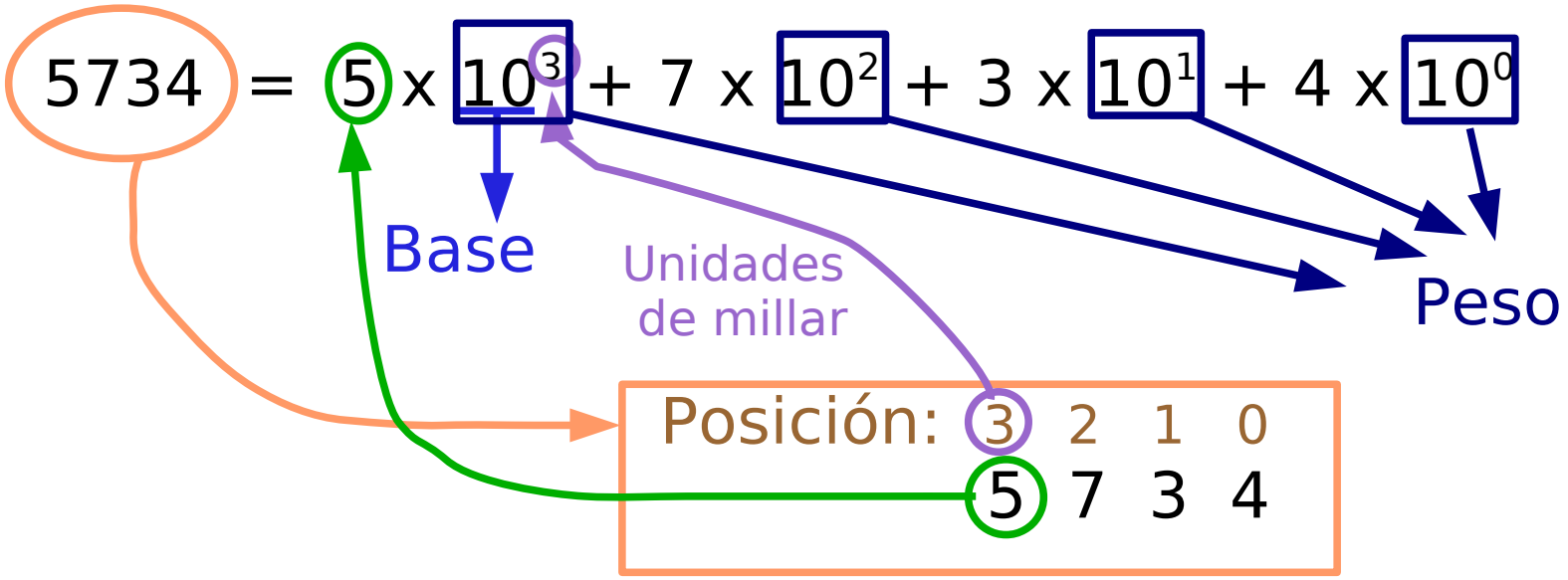
Indice

1. Representación binaria:
    - Representación posicional de magnitudes
    - Códigos binarios
2. Funciones combinacionales
3. Análisis de circuitos combinacionales
4. Diseño de circuitos combinacionales

Representación Posicional de Magnitudes

Base: Número de dígitos distintos que pueden emplearse para representar una magnitud con el sistema utilizado (base 10: 0,1,...,8,9)

La posición ocupada por cada dígito lleva asociada un peso que es una potencia de la base:



Representación Posicional de Magnitudes

Bases interesantes

Base 10: Decimal, dígitos → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

$42_{(10}$

Base 2: Binario, dígitos → 0, 1

$101010_{(2} = 42_{(10}$

Base 8: Octal, dígitos → 0, 1, 2, 3, 4, 5, 6, 7

$52_{(8} = 42_{(10}$

Base 16: Hexadecimal, dígitos → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
A, B, C, D, E, F

$2A_{(16} = 42_{(10}$

Representación Posicional de Magnitudes

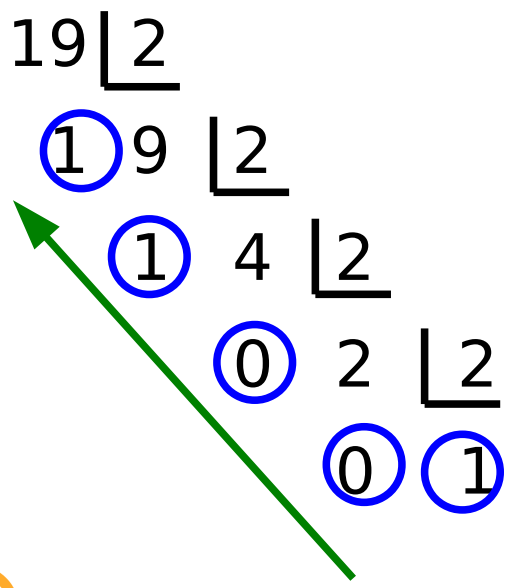
Transformaciones entre bases

Base 2 a Base 10:

$$010011_{(2} = 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 +$$

$$+ \ 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 19_{(10}$$

Base 10 a Base 2: $19_{(10} = 10011_{(2}$



Representación Posicional de Magnitudes

Transformaciones especiales

Base 2 a Base 16:

$16 = 2^4$

$010011_{(2} = 0001\ 0011_{(2} = 13_{(16}$

Base 16 a Base 2:

$A7_{(16} = 1010\ 0111_{(2}$

Base 2 a Base 8:

$8 = 2^3$

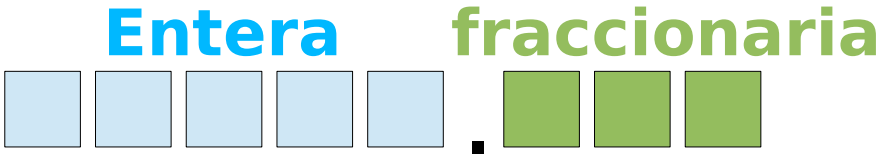
$010011_{(2} = 010\ 011_{(2} = 23_{(8}$

Base 8 a Base 2:

$37_{(8} = 011\ 111_{(2}$

Representación Posicional de Magnitudes

Representación parte fraccionaria

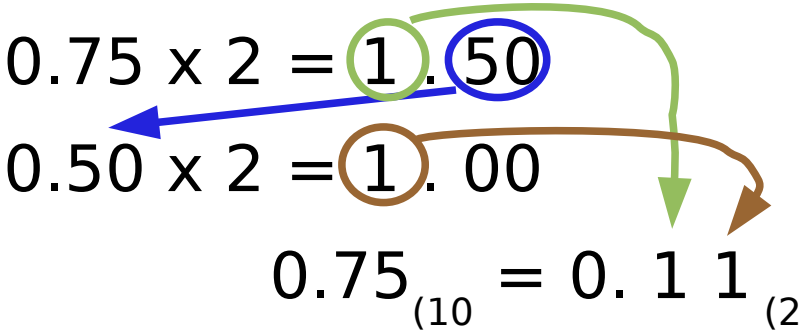


Ejemplo: 0 1 0 1 1 . 1 1 0

Base 2 a Base 10:

$$0.110_{(2} = 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} = 0.75_{(10}$$

Base 10 a Base 2:



Códigos Binarios

BCD

7 Segmentos

Gray

Detectores de Errores

Códigos alfanuméricos: ASCII, ASCII extendido

Códigos Binarios

BCD: Binary Coded Decimal

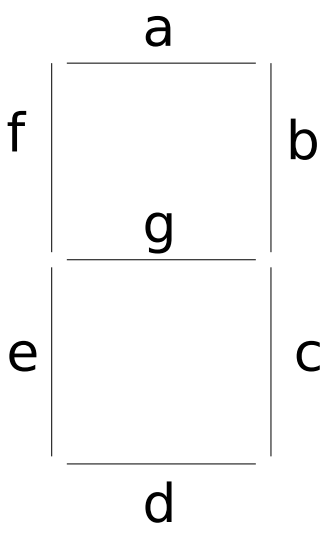
Dígito	Código BCD
0	0000
1	0001
2	0010
3	0011
4	0100

Dígito	Código BCD
5	0101
6	0110
7	0111
8	1000
9	1001

Ejemplo: $16_{(10} = 0001\ 0110\ _{(BCD}$

Códigos Binarios

7 Segmentos



Dígito	Código 7-Seg abcdefg
0	1111110
1	0110000
2	1101101
3	1111001
4	0110011

Dígito	Código 7-Seg abcdefg
5	1011011
6	0011111
7	1110000
8	1111111
9	1110011

Ejemplo: $16_{(10}$ = 0110000 0011111 $_{(7\text{-seg}}$

Códigos Binarios

Gray

Dígito	Código Gray 1 bit
0	0
1	1

Dígito	Código Gray 2 bits
0	0 0
1	0 1
2	1 1
3	1 0

Espejo

Ejercicio: Construir el código Gray de 3 bits

Códigos Binarios

Detectores de errores

- Bit de Paridad: Se añade un bit denominado **bit de paridad** al código binario. Puede hacerse de dos formas:

  1. Paridad par: El número total de 1 debe ser par.
  2. Paridad impar: El número total de 1 debe ser impar.

Código	Bit Paridad Par	Código con Paridad Par	Bit Paridad Impar	Código con Paridad Impar
0000	0	0 0000	1	1 0000
1011	1	1 1011	0	0 1011

Códigos Binarios

- Códigos alfanuméricos:

  - Codifican números, letras, signos de puntuación y otros caracteres

    - Código **ASCII**(American Standard Code for Information Interchange)
      - es un código de 7 bits
      - permite codificar 128 caracteres
      - incluye el alfabeto inglés
    - Código **ASCII extendido**
      - es un código de 8 bits
      - permite hasta 256 caracteres distintos

  - La tabla de códigos ASCII se puede representar en binario, decimal, octal o hexadecimal

Códigos Binarios

ASCII-bin

		B6 B5 B4							
		0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	0 0 0 0	NULL	DEL		0	@	P	`	p
	0 0 0 1	SOH	DC1	!	1	A	Q	a	q
	0 0 1 0	STX	DC2	"	2	B	R	b	r
	0 0 1 1	ETX	DC3	#	3	C	S	c	s
	0 1 0 0	EOT	DC4	$	4	D	T	d	t
	0 1 0 1	ENQ	NAK	%	5	E	U	e	u
B3	0 1 1 0	ACK	SYN	&	6	F	V	f	v
B2	0 1 1 1	BEL	ETB	'	7	G	W	g	w
B1	1 0 0 0	BS	CAN	(	8	H	X	h	x
B0	1 0 0 1	HT	EM	)	9	I	Y	i	y
	1 0 1 0	LF	SUB	*	:	J	Z	j	z
	1 0 1 1	VT	ESC	+	;	K	[	k	{
	1 1 0 0	FF	FS	,	<	L	\	l	|
	1 1 0 1	CR	GS	-	=	M	]	m	}
	1 1 1 0	SO	RS	.	>	N	^	n	~
	1 1 1 1	SI	US	/	?	O	_	o	DEL

Códigos Binarios

ASCII-oct/hex/dec

DEC	OCT	HEX	Symbol
0	0	0	NUL
1	1	1	SOH
2	2	2	STX
3	3	3	ETX
4	4	4	EOT
5	5	5	ENQ
6	6	6	ACK
7	7	7	BEL
8	10	8	BS
9	11	9	HT
10	12	0A	LF
11	13	0B	VT
12	14	0C	FF
13	15	0D	CR
14	16	0E	SO
15	17	0F	SI
16	20	10	DLE
17	21	11	DC1
18	22	12	DC2
19	23	13	DC3
20	24	14	DC4
21	25	15	NAK
22	26	16	SYN
23	27	17	ETB
24	30	18	CAN
25	31	19	EM
26	32	1A	SUB
27	33	1B	ESC
28	34	1C	FS
29	35	1D	GS
30	36	1E	RS
31	37	1F	US

DEC	OCT	HEX	Symbol
32	40	20	
33	41	21	!
34	42	22	"
35	43	23	#
36	44	24	$
37	45	25	%
38	46	26	&
39	47	27	'
40	50	28	(
41	51	29	)
42	52	2A	*
43	53	2B	+
44	54	2C	,
45	55	2D	-
46	56	2E	.
47	57	2F	/
48	60	30	0
49	61	31	1
50	62	32	2
51	63	33	3
52	64	34	4
53	65	35	5
54	66	36	6
55	67	37	7
56	70	38	8
57	71	39	9
58	72	3A	:
59	73	3B	;
60	74	3C	<
61	75	3D	=
62	76	3E	>
63	77	3F	?

DEC	OCT	HEX	Symbol
64	100	40	@
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	^
95	137	5F	_

DEC	OCT	HEX	Symbol
96	140	60	`
97	141	61	a
98	142	62	b
99	143	63	c
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6F	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	|
125	175	7D	}
126	176	7E	~
127	177	7F	DEL

Códigos Binarios

- Código **ASCII extendido** (8 bits-256 combinaciones)
  - Existen diferentes variaciones de este código:
  - ISO-8859-1 (Latin 1)
    - extensión del ASCII para otros idiomas europeos
  - ISO-8859-15 (Latin 9)
    - extensión del Latin 1, incluye el símbolo del Euro
  - ISO-8859-2 a ISO-8859-14
    - otras extensiones del ASCII para otros alfabetos

Códigos Binarios

ASCII Extendido Latin 1

DEC	OCT	HEX	Symbol
160	240	A0	
161	241	A1	¡
162	242	A2	¢
163	243	A3	£
164	244	A4	¤
165	245	A5	¥
166	246	A6	¦
167	247	A7	§
168	250	A8	¨
169	251	A9	©
170	252	AA	ª
171	253	AB	«
172	254	AC	¬
173	255	AD	
174	256	AE	®
175	257	AF	¯
176	260	B0	°
177	261	B1	±
178	262	B2	²
179	263	B3	³
180	264	B4	´
181	265	B5	µ
182	266	B6	¶
183	267	B7	·

DEC	OCT	HEX	Symbol
184	270	B8	¸
185	271	B9	¹
186	272	BA	º
187	273	BB	»
188	274	BC	¼
189	275	BD	½
190	276	BE	¾
191	277	BF	¿
192	300	C0	À
193	301	C1	Á
194	302	C2	Â
195	303	C3	Ã
196	304	C4	Ä
197	305	C5	Å
198	306	C6	Æ
199	307	C7	Ç
200	310	C8	È
201	311	C9	É
202	312	CA	Ê
203	313	CB	Ë
204	314	CC	Ì
205	315	CD	Í
206	316	CE	Î
207	317	CF	Ï

DEC	OCT	HEX	Symbol
208	320	D0	Ð
209	321	D1	Ñ
210	322	D2	Ò
211	323	D3	Ó
212	324	D4	Ô
213	325	D5	Õ
214	326	D6	Ö
215	327	D7	×
216	330	D8	Ø
217	331	D9	Ù
218	332	DA	Ú
219	333	DB	Û
220	334	DC	Ü
221	335	DD	Ý
222	336	DE	Þ
223	337	DF	ß
224	340	E0	à
225	341	E1	á
226	342	E2	â
227	343	E3	ã
228	344	E4	ä
229	345	E5	å
230	346	E6	æ
231	347	E7	ç

DEC	OCT	X	Symbol
232	350	E8	è
233	351	E9	é
234	352	EA	ê
235	353	EB	ë
236	354	EC	ì
237	355	ED	í
238	356	EE	î
239	357	EF	ï
240	360	F0	ð
241	361	F1	ñ
242	362	F2	ò
243	363	F3	ó
244	364	F4	ô
245	365	F5	õ
246	366	F6	ö
247	367	F7	÷
248	370	F8	ø
249	371	F9	ù
250	372	FA	ú
251	373	FB	û
252	374	FC	ü
253	375	FD	ý
254	376	FE	þ
255	377	FF	ÿ

Códigos binarios

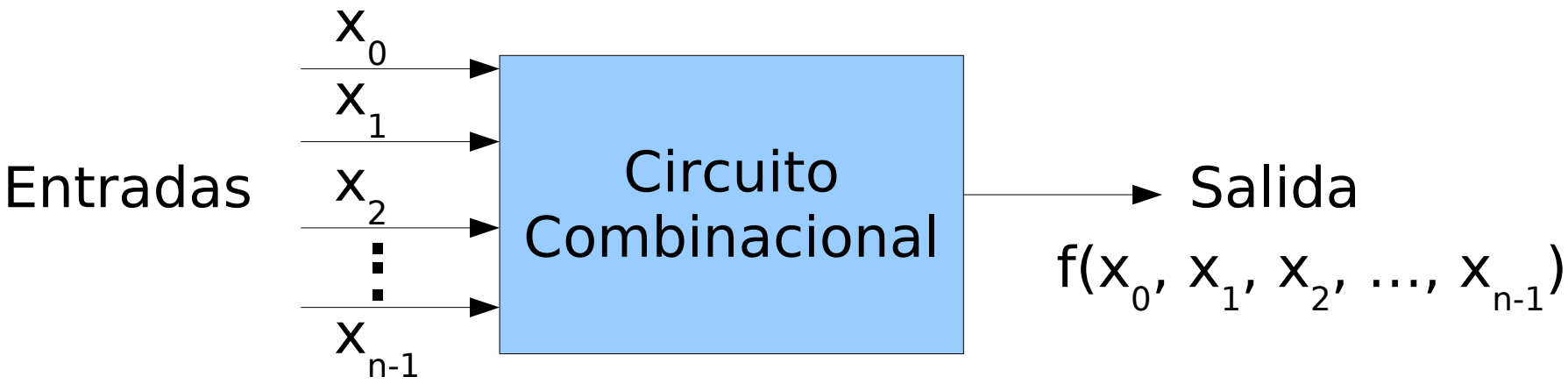
- Imágenes digitales en color
  - Cada imagen digital se compone de píxeles
  - Cada píxel logra el color combinando colores básicos, hay diferentes formatos:
    - CMYK: Cyan Magente Yellow blacK
    - HLS: Hue Saturation Ligthness
    - RGB: Red Green Blue
  - Los monitores utilizan RGB, en alta resolución usan 8 bits para R,8 para G y 8 para B, (2^{24} combinaciones) lo que proporciona 16 millones de colores diferentes.
    - Ejemplos: Negro hex 00 00 00, Blanco hex FF FF FF

      Rojo hex FF 00 00

Indice

1. Representación binaria:
    - Representación posicional de magnitudes
    - Códigos binarios
2. Funciones combinacionales
3. Análisis de circuitos combinacionales
4. Diseño de circuitos combinacionales

Funciones Combinacionales



Definición: Una función de conmutación es una aplicación $f: B^n \rightarrow B$. $f(x_0, x_1, x_2, \ldots, x_{n-1})$

x_i son **variables binarias**.

Una función de conmutación es **completamente especificada** cuando asigna un valor (0 o 1) a todos los posibles valores de sus variables. En otro caso, la función es **incompletamente especificada**.

Funciones Combinacionales

Ejemplos:

Función compl. especificada

$X_2\ X_1\ X_o$	F
0 0 0	1
0 0 1	0
0 1 0	1
0 1 1	1
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	1

Función incompl. especificada

$X_2\ X_1\ X_o$	F
0 0 0	0
0 0 1	0
0 1 0	--
0 1 1	1
1 0 0	1
1 0 1	--
1 1 0	1
1 1 1	--

Funciones Combinacionales

Representación:

Existen varias formas de representar una función de conmutación:

- Expresión
- Tabla de verdad
- Mapa de Karnaugh
- Circuito
- Lenguajes de descripción de hardware (HDL): Verilog.

Funciones Combinacionales

Ejemplo: función de tres variables

Expresión: f(x,y,z) = xy + xz + yz

Tabla de verdad

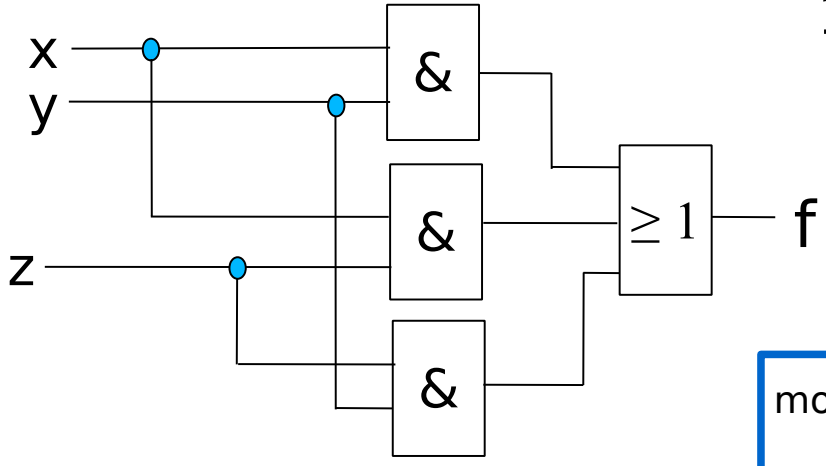
xyz	f(x,y,z)
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

Mapa de Karnaugh

x y / z	00	01	11	10
0	0	0	1	0
1	0	1	1	1

f

Circuito



Código Verilog

```
module ejemplo(
    output f,
    input x, y,z );

  assign   f = x & y | x & z | y & z;
endmodule
```

Funciones Combinacionales

Ejemplo: función de 4 variables

Expresión: $g(x,y,z,u) = xyu + xz\bar{u} + yz$

zu \ xy	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	1	1	0
10	0	1	1	1

g

Código Verilog

```
module g(
        input  x, y, z ,u
        output g
    );

assign g = (x & y & u) | (x & z & ~u)| (y & z);

endmodule
```

Funciones Combinacionales

Formas normalizadas:

- **suma de productos (sp)**

  $\bar{x}yz + x\bar{w} + \bar{z}w$

  $\bar{x}y$

  $xy + \bar{y} + yz$

- **producto de sumas (ps)**

  $(\bar{y}+w)(x+\bar{z})(\bar{x}+y+w)$

  $x\,(y+z)$

  $(x+\bar{z}+w)$

No normalizadas:

$(abc+b'ad+(a+b+c)'+de)'$ $\qquad$ $xy + z(y+w)$

Funciones Combinacionales

Formas normalizadas:

Para obtener una forma normalizada basta aplicar las leyes de DeMorgan y la propiedad distributiva

- Si una función se expresa en **suma de productos** es fácil obtener sus **unos**

  si un producto es 1 la función vale 1 (elemento dominante de la suma)

- Si una función se expresa en **producto de sumas** es fácil obtener sus **ceros**

  si una suma es 0 la función vale 0 (elemento dominante del producto)

Ejercicio:
obtenga las tablas de verdad de $f = \bar{x}yz + x\bar{w} + \bar{z}w$ / $g=(\bar{y}+w)(x+\bar{z})(\bar{x}+y+w)$

Funciones Combinacionales

Formas canónicas:

- **suma de mintérminos**

el **mintérmino** es un producto que contiene todas las variables

$\bar{x}yzw + xyz\bar{w} + xy\bar{z}w$ (4 variables)

$\bar{x}yz + x\bar{y}z$ (3 variables)

$xy + \bar{x}\,\bar{y} + x\bar{y}$ (2 variables)

- **producto de maxtérminos**

el **maxtérmino** es una suma que contiene todas las variables

$(x+\bar{y}+z+w)(x+y+\bar{z}+w)(\bar{x}+y+z+w)$ (4 variables)

$(x+\bar{z}+y)\,(x+z+\bar{y})$ (3 variables)

$(\bar{y}+\bar{z})\,(y+z)$ (2 variables)

Funciones Combinacionales

Siempre es posible obtener una expresión de la función mediante **suma de mintérminos** y esta es **única:**

- para cada combinación de entradas en la que la función vale 1 se elige el mintérmino que vale 1 para esa combinación

- se suman todos los mintérminos generados

a b c	z
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	1

$$z = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc = m3 + m5 + m6 + m7$$

notación m

$$z = \Sigma(3,5,6,7)$$

Funciones Combinacionales

Siempre es posible obtener una expresión de la función mediante **producto de maxtérminos** y esta es **única:**

- para cada combinación de entradas en la que la función vale 0 se elige el maxtérmino que vale 0 para esa combinación

- se multiplican todos los maxtérminos generados

a b c	z
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	1

$z = \Pi(0,1,2,4)$

notación M

$z = (a+b+c)(a+b+\overline{c})(a+\overline{b}+c)(\overline{a}+b+c) = M0\ M1\ M2\ M4$

Indice

1. Representación binaria:
    - Representación posicional de magnitudes
    - Códigos binarios
2. Funciones combinacionales
3. Análisis de circuitos combinacionales
4. Diseño de circuitos combinacionales

Análisis de Circuitos Combinacionales

Análisis Lógico:

Dado un circuito, consiste en encontrar:

- la expresión algebraica que implementa,
- su tabla de verdad y/o el k-mapa,
- explicación verbal de su función.

Análisis Temporal:

Dado un circuito, consiste en representar la evolución en el tiempo de las entradas y salidas del circuito (cronograma**)**

- **análisis ideal:** Suponiendo que las puertas no tienen retrasos.
- **análisis con retrasos:** un modelo simple es suponer que todas las puertas introducen el mismo retraso.

Análisis de Circuitos Combinacionales

Análisis Lógico: Procedimiento

1. Se obtiene la función lógica realizada por las puertas cuyas entradas corresponden a las entradas primarias del circuito.

2. Se obtiene la función lógica realizada en puertas con entradas conocidas (entradas primarias o salidas de puertas ya calculadas).

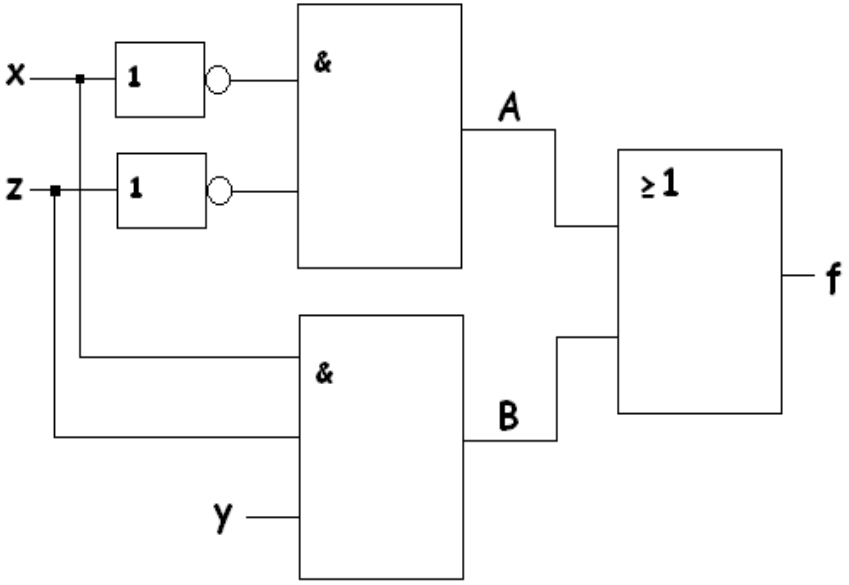
3. Se repite el paso anterior hasta obtener la función de salida

4. Se simplifica la expresión obtenida y/o se representa en un mapa o tabla

Análisis de Circuitos Combinacionales

Análisis Lógico: Ejemplo

Circuito:



Expresión:

f(x,y,z) = A + B

A = x'z'

B = xyz

f(x,y,z) = xyz + x'z'

Tabla:

xyz	f(x,y,z)
000	1
001	0
010	1
011	0
100	0
101	0
110	0
111	1

f(x,y,z) = 1 si

- xyz=1 si x=y=z=1 (111)
- ó
- x'z'=1 si x=z=0 (0-0)

Análisis de Circuitos Combinacionales

Análisis temporal: Procedimiento

- **análisis ideal:** podemos extraer la información de la tabla de verdad o la expresión lógica

- **análisis con retrasos:** debemos obtener todas las ondas que se propagan por el circuito e ir retrasando la onda cada vez que se pasa por una puerta

Análisis de Circuitos Combinacionales

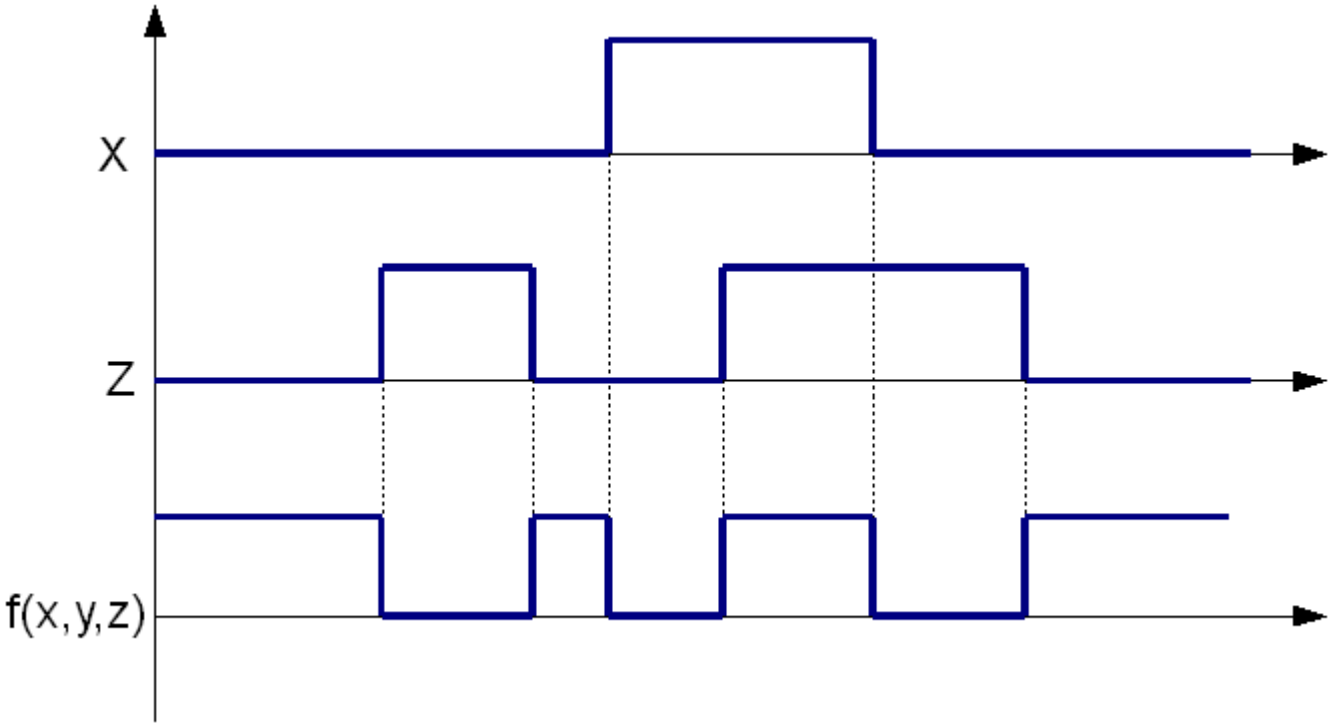
Análisis Temporal (ideal): Ejemplo

Tabla:

xyz	f(x,y,z)
000	1
001	0
010	1
011	0
100	0
101	0
110	0
111	1

Cronograma (con y=1):

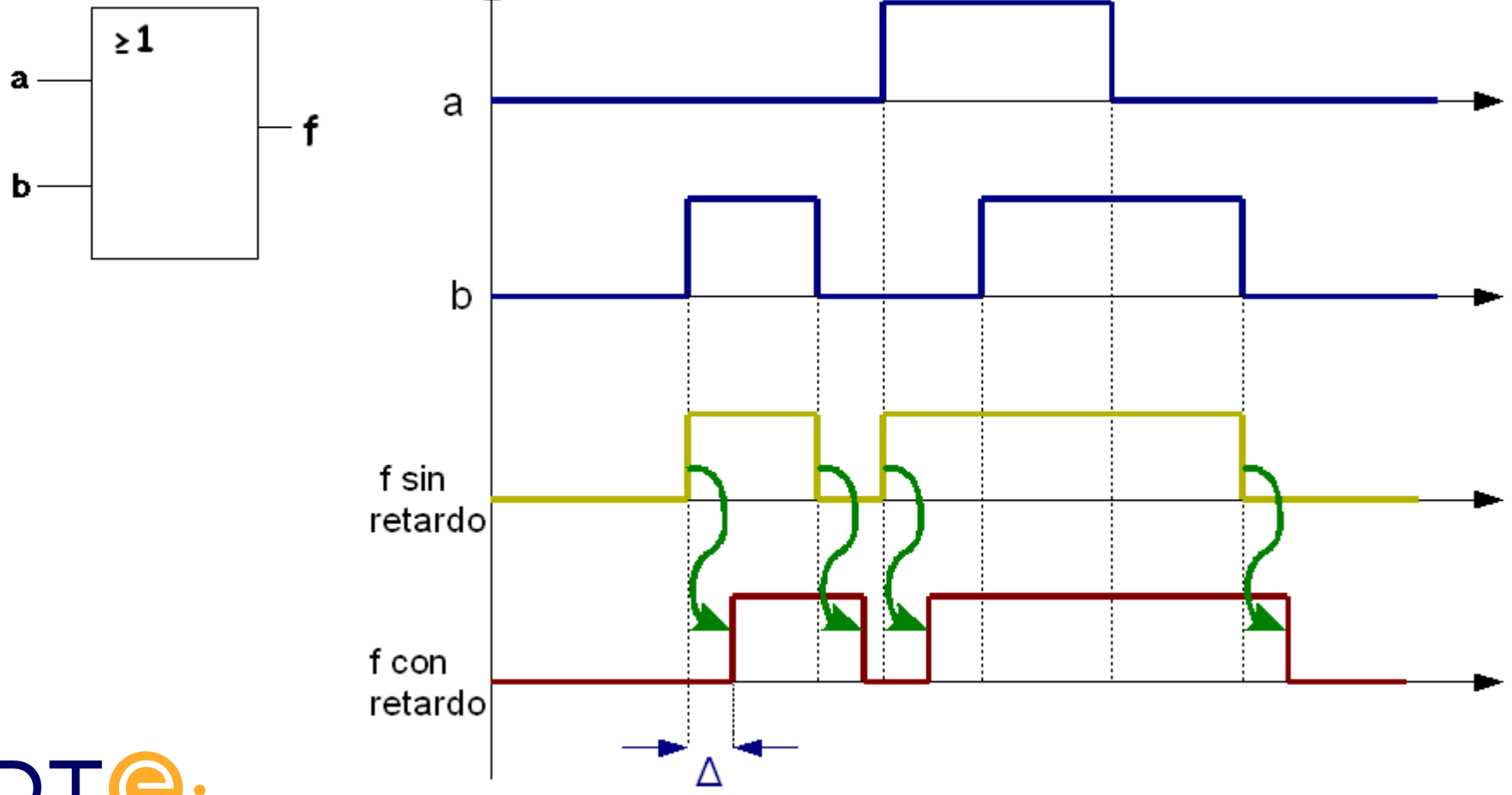
(sin considerar retrasos)



Análisis de Circuitos Combinacionales

Análisis Temporal (con retrasos): Ejemplo

Para cada puerta hay que tener en cuenta el retraso, consideremos una OR que introduce un retraso Δ

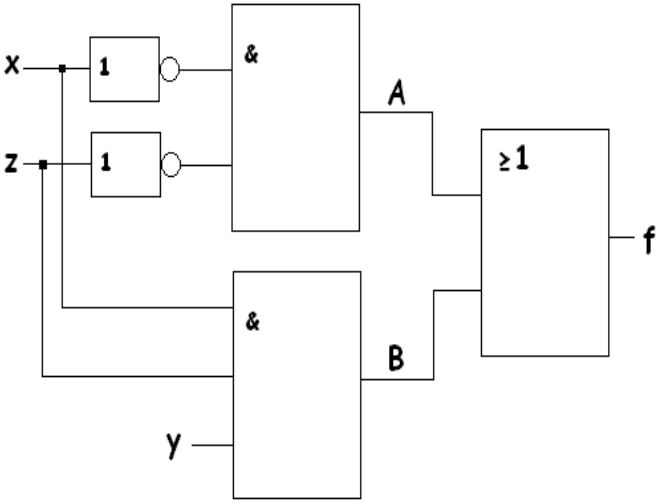


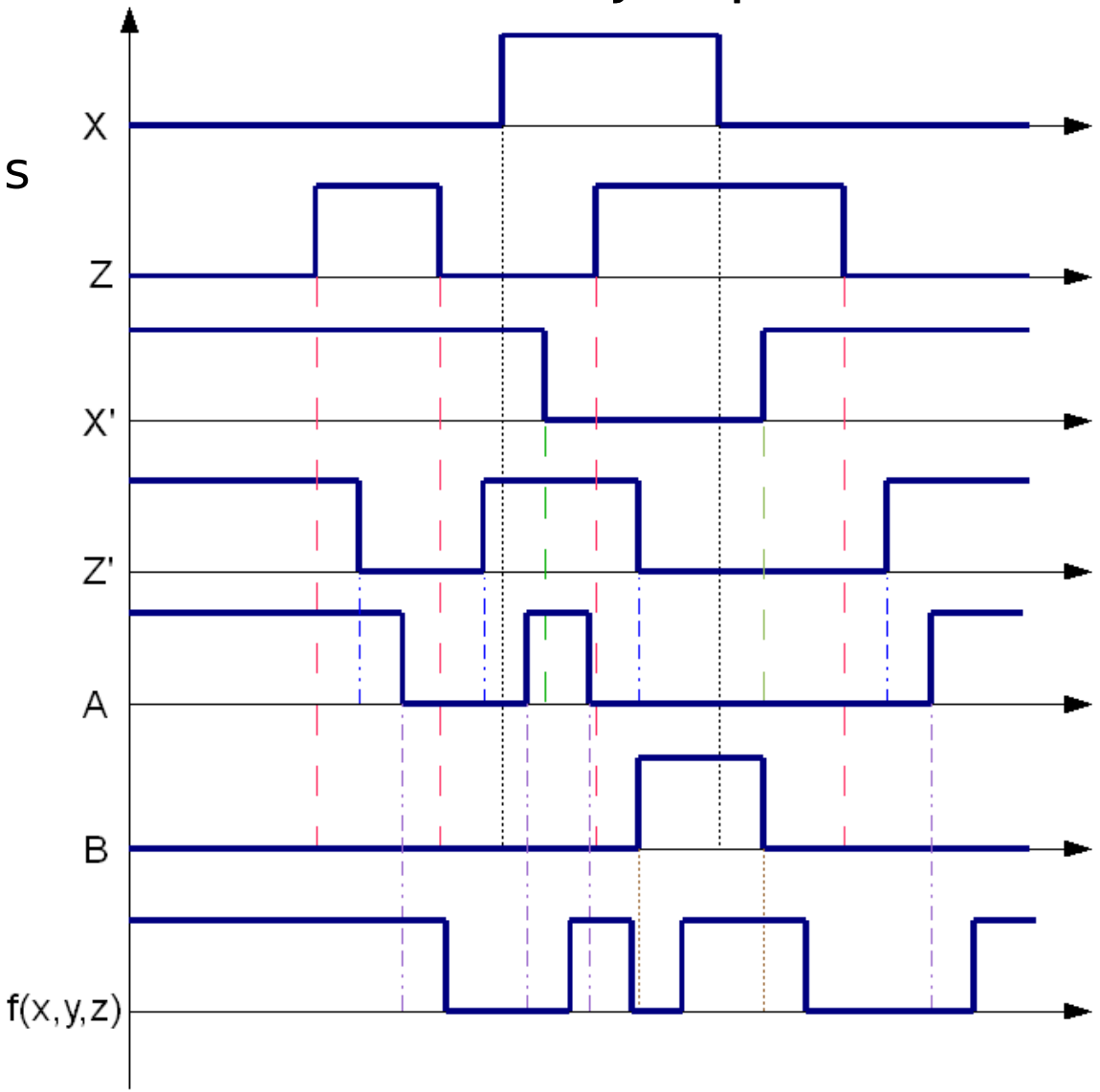
Análisis de Circuitos Combinacionales

Análisis Temporal (con retrasos): Ejemplo

Cronograma (con y=1)

(retraso Δ en todas las puertas)





Análisis de Circuitos Combinacionales

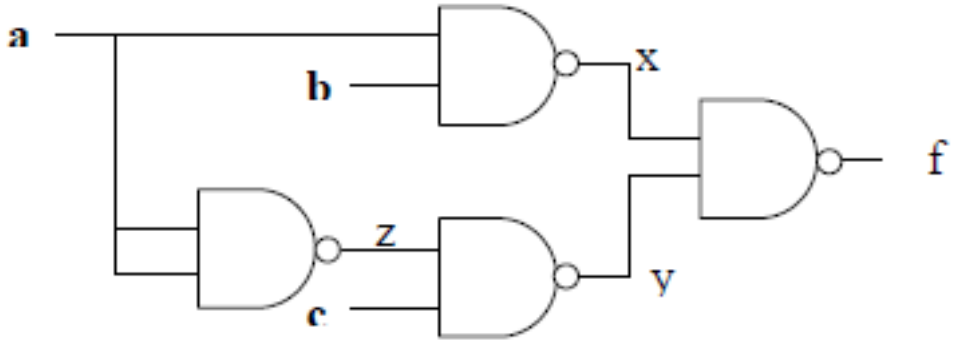
Análisis Temporal (con retrasos): Beneficios

- Permite estimar el retraso de propagación de un circuito que es el tiempo que tarda en proporcionar un valor de salida correcto

- Permite analizar comportamientos no esperados debido a las características temporales: retraso excesivo, **azares**,...

Ejercicio: Analizar temporalmente el circuito de la figura en los casos: 1) análisis ideal y 2) análisis con retrasos considerando Δ=1ns Comparar los resultados obtenidos.

b=c=1,
a → señal periódica (T=10ns)



Indice

1. Representación binaria:
    - Representación posicional de magnitudes
    - Códigos binarios
2. Funciones combinacionales
3. Análisis de circuitos combinacionales
4. Diseño de circuitos combinacionales

Diseño de Circuitos Combinacionales

Objetivos y conceptos básicos:

El diseño (o síntesis) de un circuito es el proceso inverso al análisis:

- se parte de unas especificaciones (descripción de la tarea que ha de realizar el circuito) y se obtiene un circuito que cumple con los requerimientos planteados

- las especificaciones pueden ser de varios tipos, por ejemplo:

    - funcionales

    - temporales (velocidad, azares, ...)

    - de consumo

    - de reusabilidad

    - tecnológicas

Diseño de Circuitos Combinacionales

Objetivos y conceptos básicos

En este tema nos centraremos en el diseño de circuitos:

- que tengan dos niveles (tres para simple raíl)

- que realicen funciones en suma de productos o producto de sumas

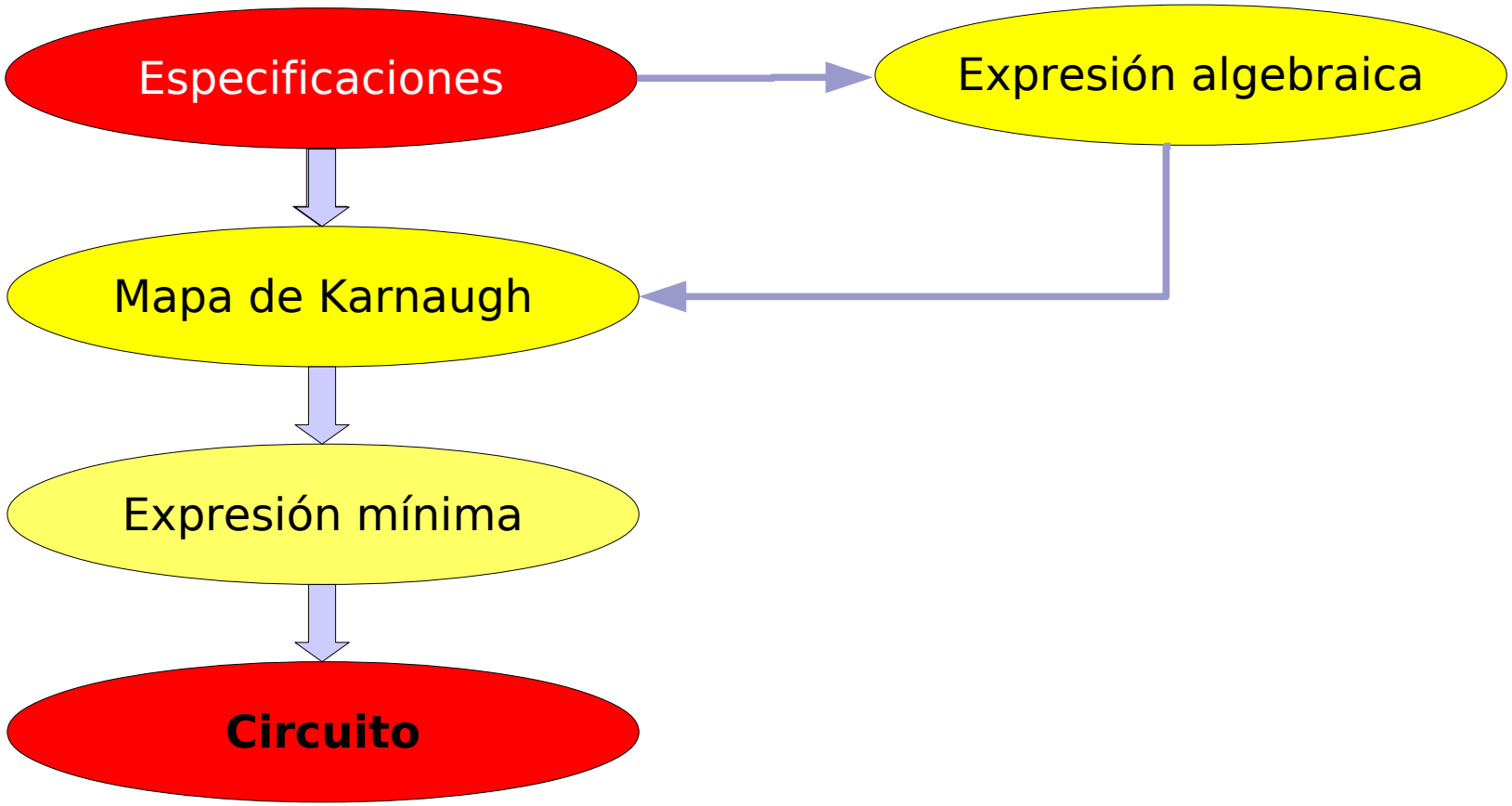
- que sean óptimos: la expresión en sp o ps debe ser lo más simple posible

Por tanto, los criterios de diseño son:

- reducir el número de puertas
- reducir el número de conexiones
- uso de puertas AND, OR, NAND y NOR
  (e inversores en simple raíl)

Diseño de Circuitos Combinacionales

Pasos del proceso:



Diseño de Circuitos Combinacionales

Pasos del proceso:

Paso 1: Descripción textual -> Descripción matemática

- determinar variables de entrada y salida

- si procede, asignar significado a los valores 0 y 1.

- obtener una descripción matemática: expresión, tabla o mapa de Karnaugh

Paso 2: Obtener el mapa de Karnaugh

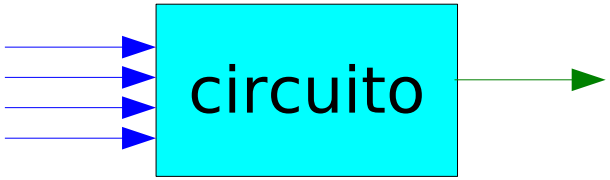
Si en el paso anterior no se obtuvo directamente el mapa sino una expresión algebraica, se tendrá que generar el mapa de Karnaugh a partir de la expresión obtenida

Diseño de Circuitos Combinacionales

Pasos del proceso

Ejemplo1 (pasos 1 y 2):

Suponga que los números entre 0 y 15 están representados en binario con cuatro bits: X3-X0, donde X3 es el bit más significativo. Diseñe un circuito que de salida Z = 1 si y sólo si el número X3-X0 es primo.



X3X2X1X0	z
0 0 0 0	0
0 0 0 1	0
0 0 1 0	1
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

X3X2X1X0	z
1 0 0 0	0
1 0 0 1	0
1 0 1 0	0
1 0 1 1	1
1 1 0 0	0
1 1 0 1	1
1 1 1 0	0
1 1 1 1	0

Diseño de Circuitos Combinacionales

Pasos del proceso

Ejemplo2 (pasos 1 y 2):

Se desea diseñar un circuito combinacional que recibe información del estado de tres bombillas (encendida o apagada) y del estado de un único interruptor (on - off). El circuito debe generar una alarma que se active cuando alguna de las bombillas no esté encendida cuando el interruptor está on, o cuando alguna bombilla esté encendida y el interruptor esté off.

Entradas: tres bombillas, interruptor Salida: Alarma

(b_1, b_2, b_3, i) $\qquad$ a

$b_i = \begin{cases} 0 & \text{apagada} \\ 1 & \text{encendida} \end{cases}$ $\qquad$ $i = \begin{cases} 0 & \text{off} \\ 1 & \text{on} \end{cases}$ $\qquad$ $a = \begin{cases} 0 & \text{inactiva} \\ 1 & \text{activa} \end{cases}$

i b_1 b_2 b_3	a
0 0 0 0	0
0 0 0 1	1
0 0 1 0	1
0 0 1 1	1
0 1 0 0	1
0 1 0 1	1
0 1 1 0	1
0 1 1 1	1

i b_1 b_2 b_3	a
1 0 0 0	1
1 0 0 1	1
1 0 1 0	1
1 0 1 1	1
1 1 0 0	1
1 1 0 1	1
1 1 1 0	1
1 1 1 1	0

Diseño de Circuitos Combinacionales

Ejemplo3 (pasos 1 y 2):

Se desea diseñar un circuito de alarma de coche de dos puertas de tal forma que suene la alarma cuando:

- Las puertas estén cerradas, el motor apagado y se abra el maletero.
- El motor esté encendido, las puertas cerradas y el maletero abierto.
- El freno de mano quitado, el motor encendido y alguna de las puertas abiertas.

Entradas: tres sensores, (p, m, f, M)

p: 0 puertas cerradas, 1 alguna abierta

m: 0 motor apagado, 1 motor encendido

f: 0 freno quitado, 1 freno puesto

M: 0 maletero cerrado, maletero abierto

Salida: A: 0 alarma inactiva, 1 alarma activa

$$A = \bar{p}\,\bar{m}\,M + m\,\bar{p}\,M + \bar{f}\,m\,p$$

Piense como añadir una entrada que permita desactivar la alarma

Diseño con K-mapa

Paso 3: Obtener la expresión mínima en dos niveles

- Nos basaremos en el método del K-mapa

- Expresión mínima como suma de productos

  - Nos fijamos en los 1 del K-mapa, o mintérminos, que son términos producto.
  - Agrupamos los mintérminos para conseguir términos productos con menor número de variables (**implicantes**).

- Expresión mínima como producto de sumas

  - Nos fijamos en los 0 del K-mapa, o maxtérminos, que son términos suma.
  - Agrupamos los maxtérminos para conseguir términos sumas con menor número de variables (**implicadas**).

Diseño con K-mapa

Paso 3: Obtener la expresión mínima en dos niveles (cont)

El agrupamiento de **1** para construir términos productos con menor número de variables es posible gracias a:

1) La adyacencia entre las celdas de un K-mapa (sólo cambian un bit, por efecto del código Gray)

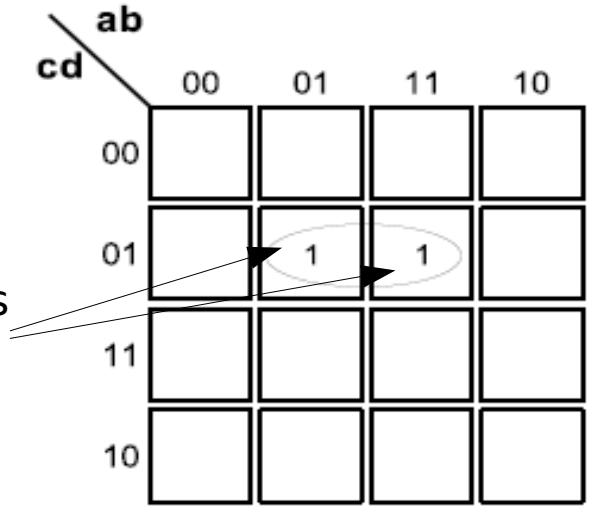
2) Si un término producto se expresa como **p q**, otro adyacente a él, que varíe un bit, sería **p q'**, por tanto la suma de los dos:

$$pq + pq' = p(q + q') = p$$

Es decir, se elimina la variable que aparece complementada y sin complementar en ambos términos.

Ejemplo: f = a' b c' d + a b c' d =

= b c' d (a'+a) = b c' d



Diseño con K-mapa

Implicante

Es un 1 o grupo de 1 representado en el K-mapa. Los grupos tienen 2^n elementos, y estos deben ser vecinos.

Los grupos se van formando a partir de grupos de tamaño inmediatamente inferior. Por ejemplo, agrupamos dos 1 vecinos para formar un grupo. Luego, este grupo podemos agruparlo con otro vecino para obtener un grupo de 4.

El número de 1 del grupo determina el orden de la implicante.

El orden de la implicante está relacionado con el número de variables que posee la expresión del término producto que lo representa.

Diseño de Circuitos Combinacionales

Diseño con K-mapa

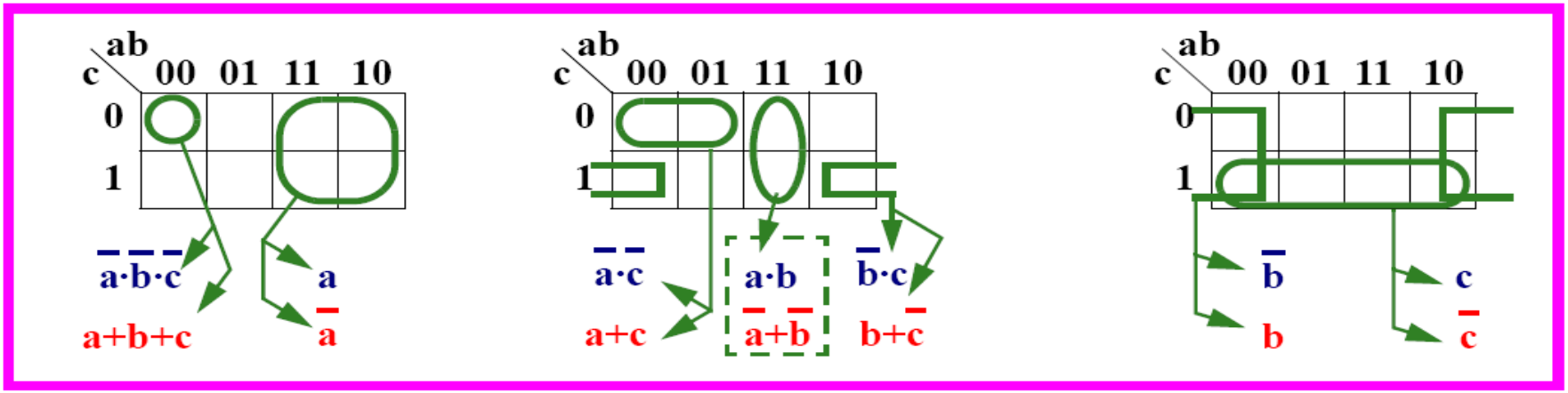
Implicante

Orden	Nº de 1's	Nº variables	Ejemplo 5 var.	
			Implicante	Cuantas
0	$1=2^0$	n	ab'cd'e	32
1	$2=2^1$	n - 1	ab'd'e	80
2	$4=2^2$	n - 2	ab'e	80
3	$8=2^3$	n - 3	b'e	40
4	$16=2^4$	n - 4	b'	10
5	$32=2^5$	n - 5	1	1
k	$m=2^k$	n - k		

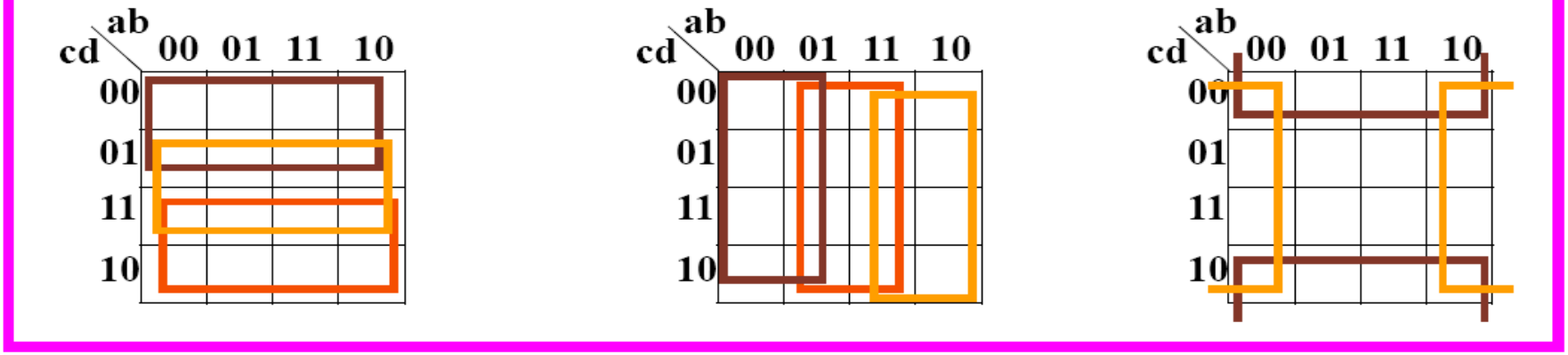
Diseño de Circuitos Combinacionales

Diseño con K-mapa

Agrupaciones posibles



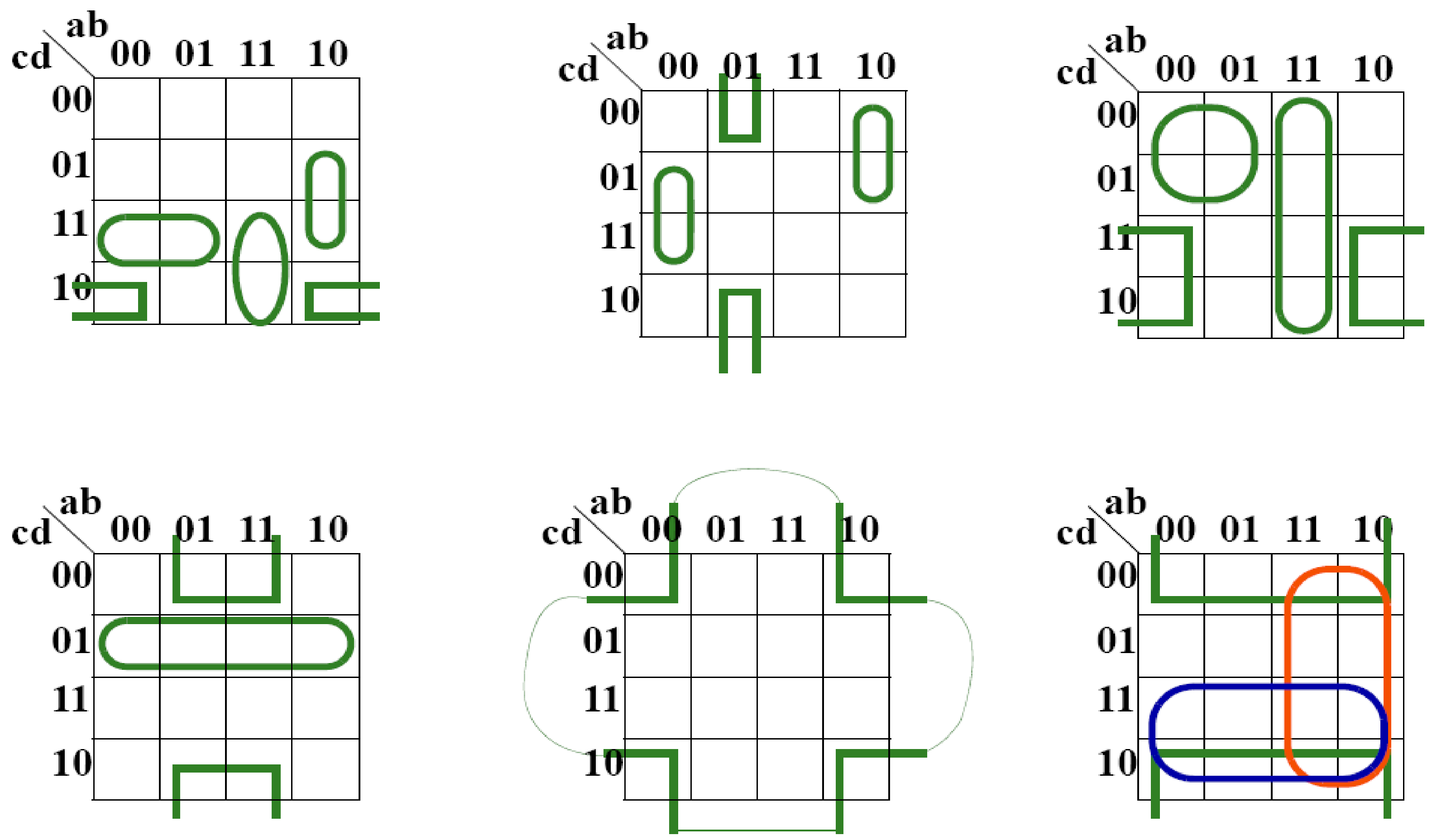
Los mapas de 4 variables contienen varios mapas de 3



Diseño de Circuitos Combinacionales

Diseño con K-mapa

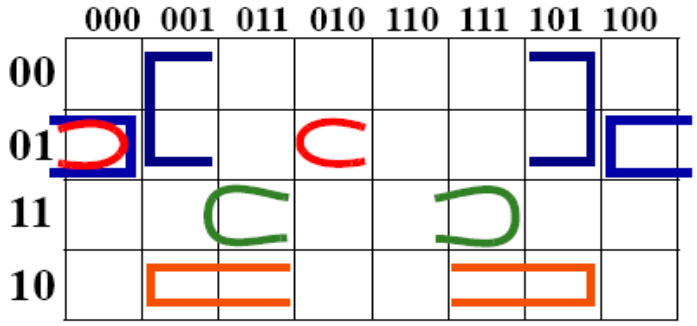
Agrupaciones posibles

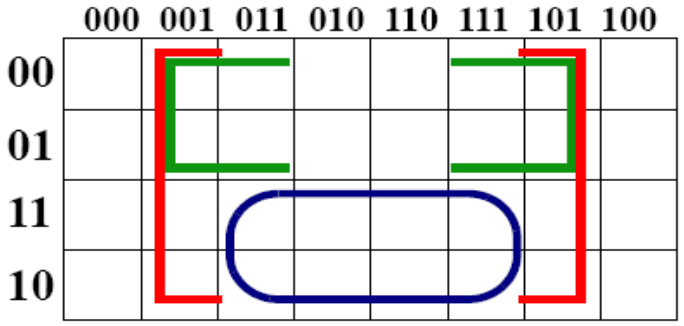


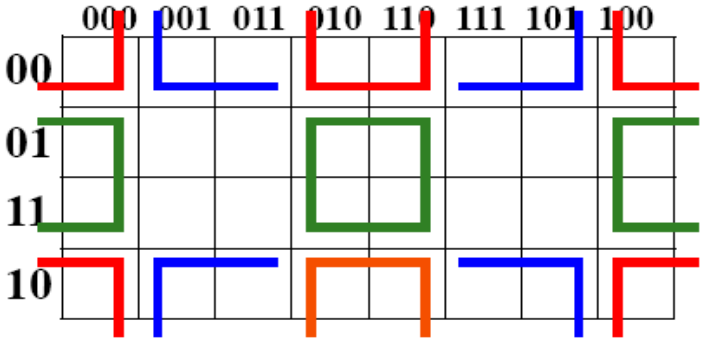
Diseño de Circuitos Combinacionales

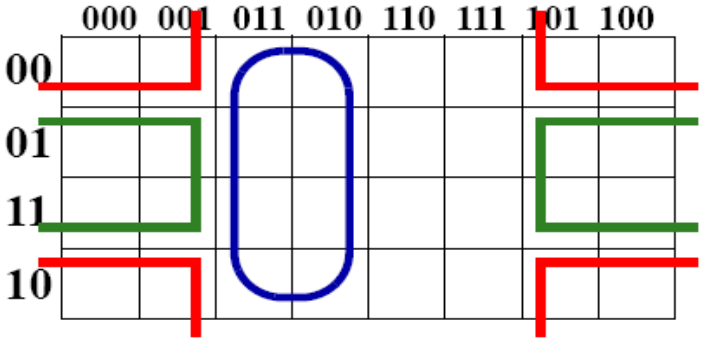
Diseño con K-mapa

Agrupaciones posibles





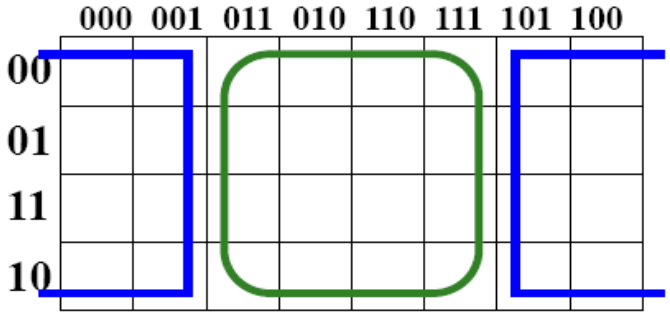


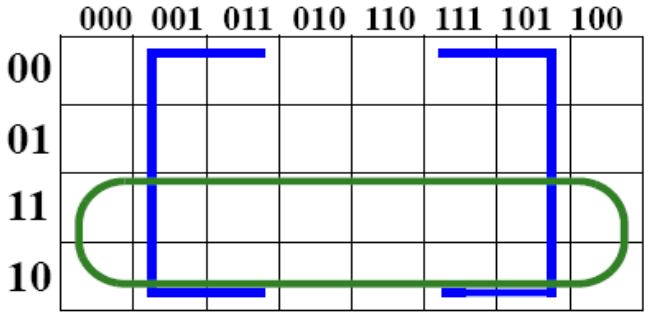


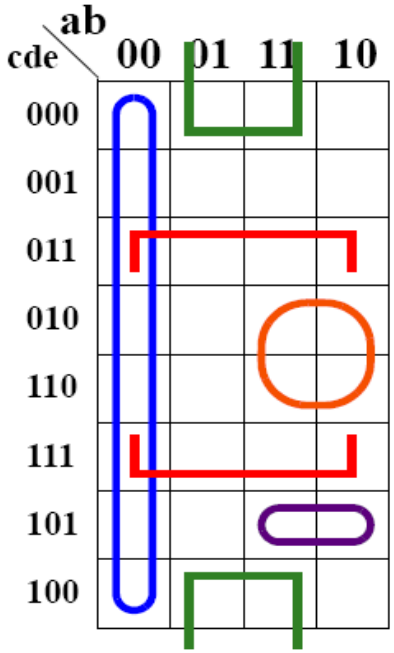
Diseño de Circuitos Combinacionales

Diseño con K-mapa

Agrupaciones posibles







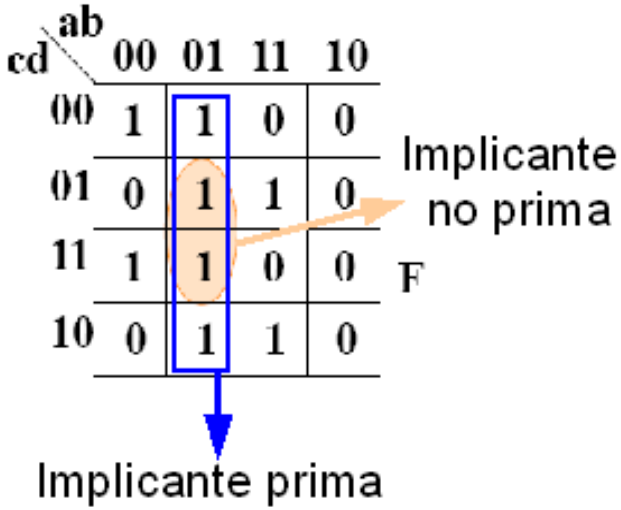
Diseño de Circuitos Combinacionales

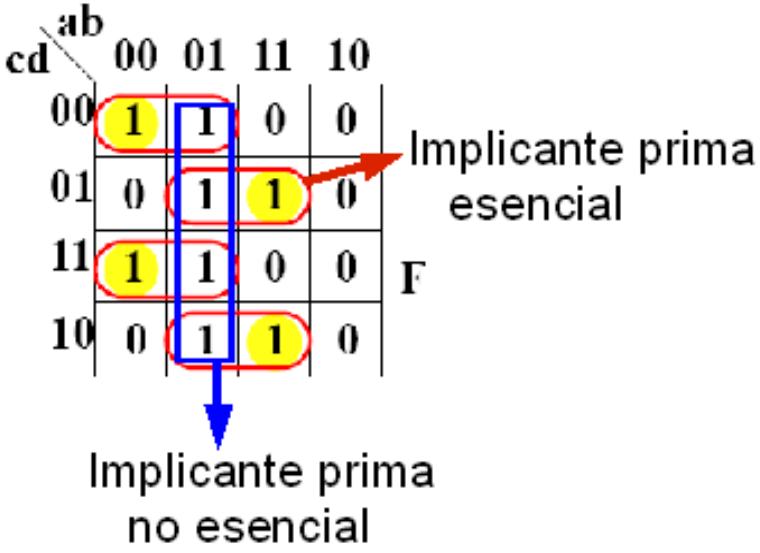
Diseño con K-mapa

Definiciones

Una implicante es **prima** si no está cubierta por ninguna otra implicante de la función.

Una implicante prima es **esencial** si cubre algún mintérmino no incluido en ninguna otra implicante prima. Al mintérmino se le denomina **distinguido**.





Diseño de Circuitos Combinacionales

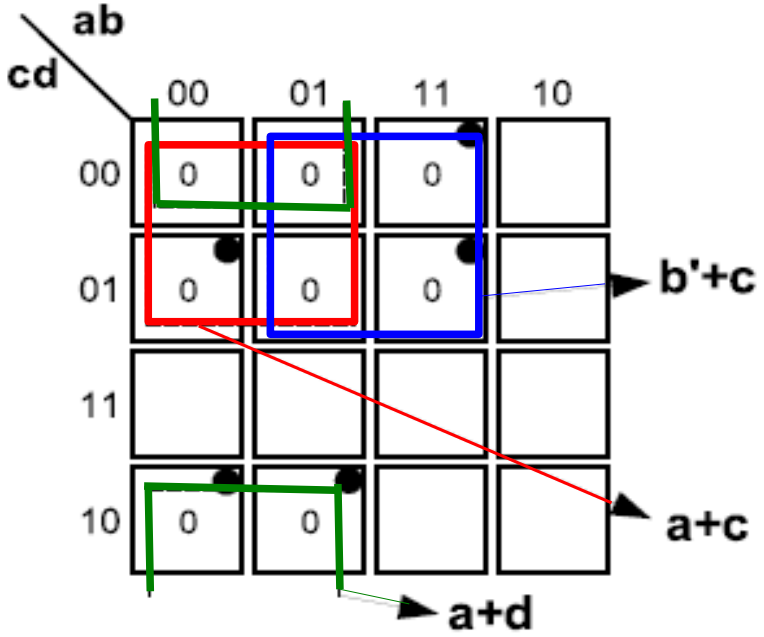
Diseño con K-mapa

La expresión mínima en producto de sumas se obtiene trabajando con las implicadas y los maxtérminos de la misma forma.

Una implicada es **prima** si no está cubierta por ninguna otra implicada de la función.

Una implicada prima es **esencial** si cubre algún maxtérmino no incluido en ninguna otra implicada prima. Al maxtérmino se le denomina **distinguido**.

$$F = \Pi(0,1,2,4,5,6,12,13)$$



Diseño de Circuitos Combinacionales

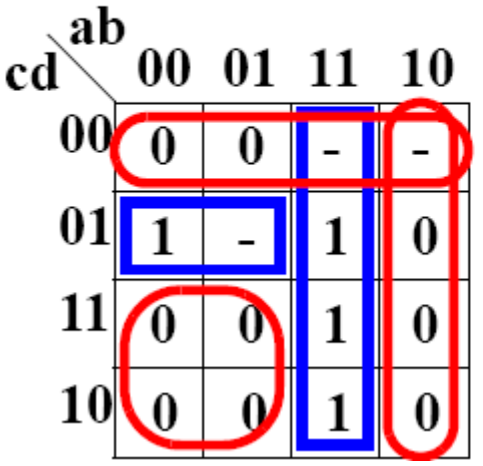
Diseño con K-mapa

Funciones incompletamente especificadas

Las casillas con inespecificación se usan como mejor nos convenga:

- Se pueden incluir para formar grupos mayores.
- No es necesario cubrirlas todas.

Ejemplo: $F = \Sigma(1, 13, 14, 15) + d(5, 8, 12)$



$F_{sp} = a \cdot b + \bar{a} \cdot \bar{c} \cdot d$ ⇒ 5 y 12 se hacen 1

$F_{ps} = (a + \bar{c}) \cdot (c + d) \cdot (\bar{a} + b)$ ⇒ 8 y 12 se hacen 0

Diseño con K-mapa

Expresión mínima en s.p.

La suma mínima se obtiene usando el menor número de implicantes primas obtenidas del K-mapa y que permitan cubrir todos los mintérminos del mismo.

Directrices para la búsqueda de la expresión mínima:

1) Buscar implicantes primas esenciales. Éstas deben aparecer obligatoriamente en la expresión mínma en s.p.

2) Para los mintérminos sin cubrir, procederemos uno por uno, a analizar cuáles son las implicantes primas que permiten su cubrimiento y como regla general se escogerá aquella que, a igualdad de número de literales, tiene un cubrimiento adicional de mintérminos mayor.

3) Repetir el punto 2 hasta que se cubra todo el K-mapa

Consideraciones finales:

a) Las inespecificaciones no se cubren

b) Si no se pudiese aplicar los puntos 1 y 2, se deberán tomar suposiciones de cubrimiento y evaluar, al final, cuál de todas ellas se traduce en un menor coste.

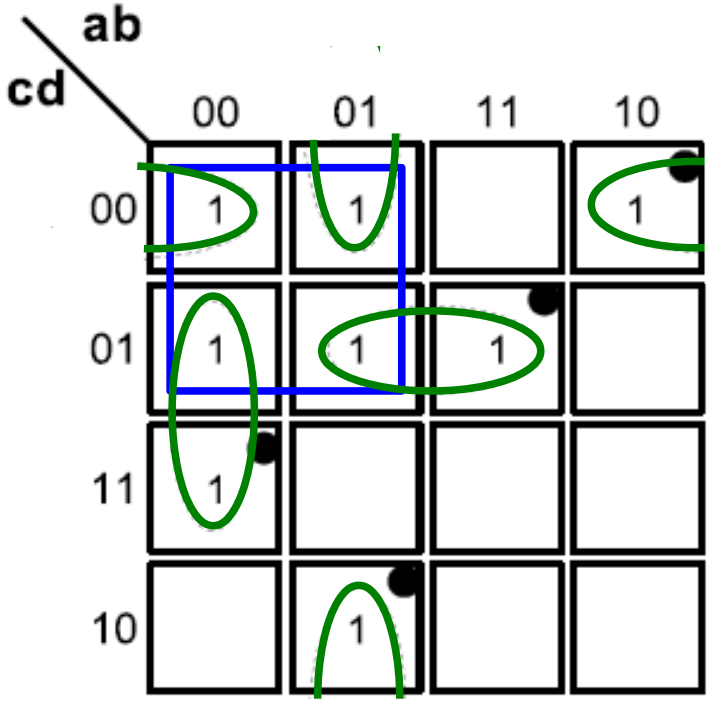
Expresión mínima en p.s.

Idénticos criterios que para s.p, pero usando Implicadas primas.

Diseño de Circuitos Combinacionales

Ejemplos de obtención de la expresión mínima

Ejercicio 1.- f = ∑(0,1,3,4,5,6,8,13)



Buscamos las Ip esenciales

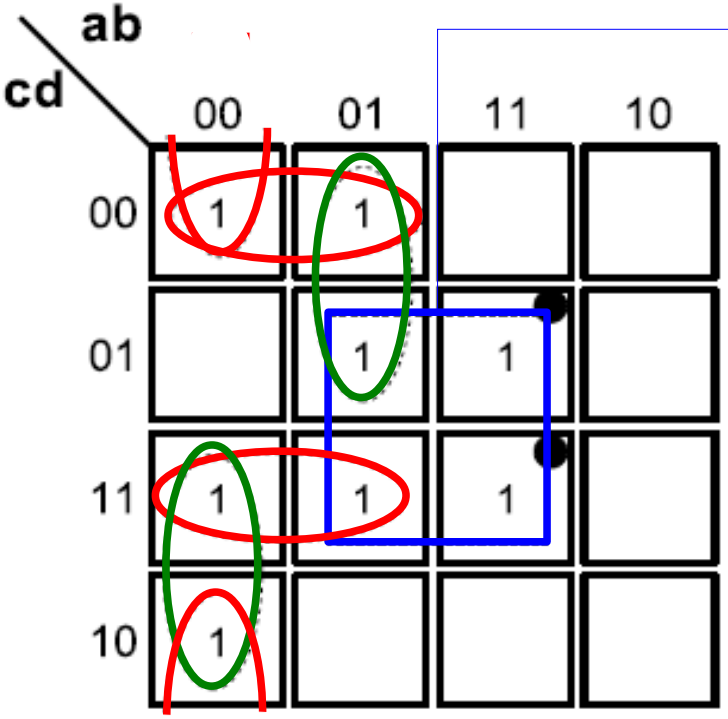
f = a'b'd + b'c'd'+a'bd'+bc'd

Fin. Se han cubierto todos los mintérminos sólo con las IP esenciales

Diseño de Circuitos Combinacionales

Ejemplos de obtención de la expresión mínima

Ejercicio 2.- f = ∑(0,2,3,4,5,7,13,15)



1)Buscamos las Ip esenciales : **bd** con la que no se cubren todos los mintérminos

2)Nos fijamos en el mintérmino 4. Está cubierto por las Ip **a'c'd'** y **a'bd**. Ambas del mismo coste pero la primera cubre también al mintérmino 0 que no estaba cubierto por **bd**. Escogemos **a'c'd'**

3) Ahora sólo quedan por cubrir los mintérminos 2 y3. Obviamente escogemos **a'b'c**

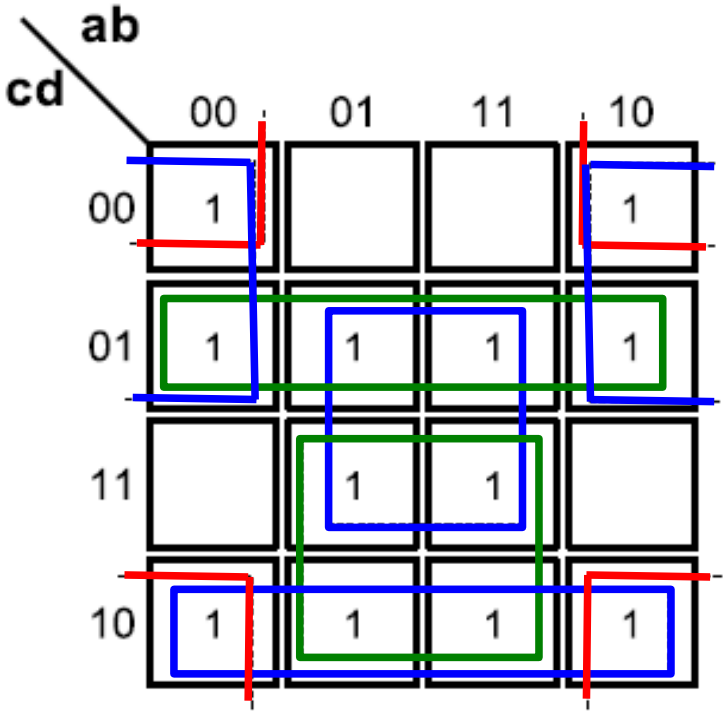
f = bd + a'c'd' + a'b'c

Fin. Se han cubierto todos los mintérminos

Diseño de Circuitos Combinacionales

Ejemplos de obtención de la expresión mínima

Ejercicio 3.- f = ∑(0,1,2,5,6,7,8,9,10,13,14,15)



1)Buscamos las Ip esenciales : ¡NO HAY!

2)Nos fijamos en el mintérmino 0. Está cubierto por las Ip **b'd'** y **b'c'**, ambas del mismo coste. **Suponemos** que la expresión mínima está formada por **b'c'**.

f = b'c' + bd + cd'

3) Repetimos el proceso en el caso en que el mintérmino 0 hubiera sido cubierto por b'd'

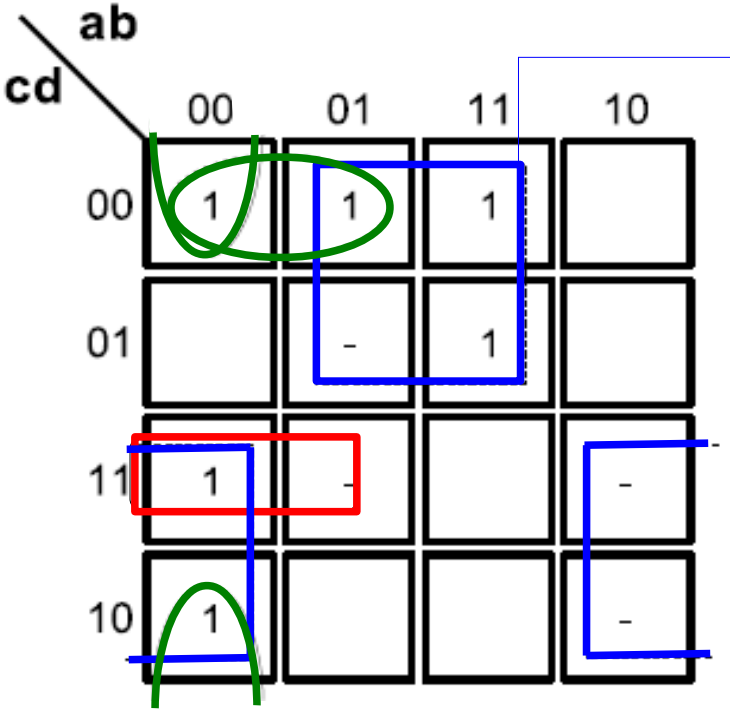
f = b'd' + c'd + bd

Fin. Se han cubierto todos los mintérminos y ambas expresiones tienen el mismo coste. Cualquiera de las dos representa la solución mínima.

Diseño de Circuitos Combinacionales

Ejemplos de obtención de la expresión mínima

Ejercicio 4.- f = ∑(0,2,3,4,12,13) + d(5,7,10,11)



1) Buscamos las Ip esenciales usando inespecificaciones como mintérminos y rechazando aquellas formadas sólo por inespecificaciones: **bc'**

2) Nos fijamos en el mintérmino 3, cubierto por **a'cd** y **b'c**. La mejor opción es **b'c.**

3) Sólo queda por cubrir el mintérmino 0. Hay dos posibilidades: las implicantes **a'b'd'** y **a'c'd'**. Cualquier opción es válida.

4) No se busca cubrimiento de inespecificaciones

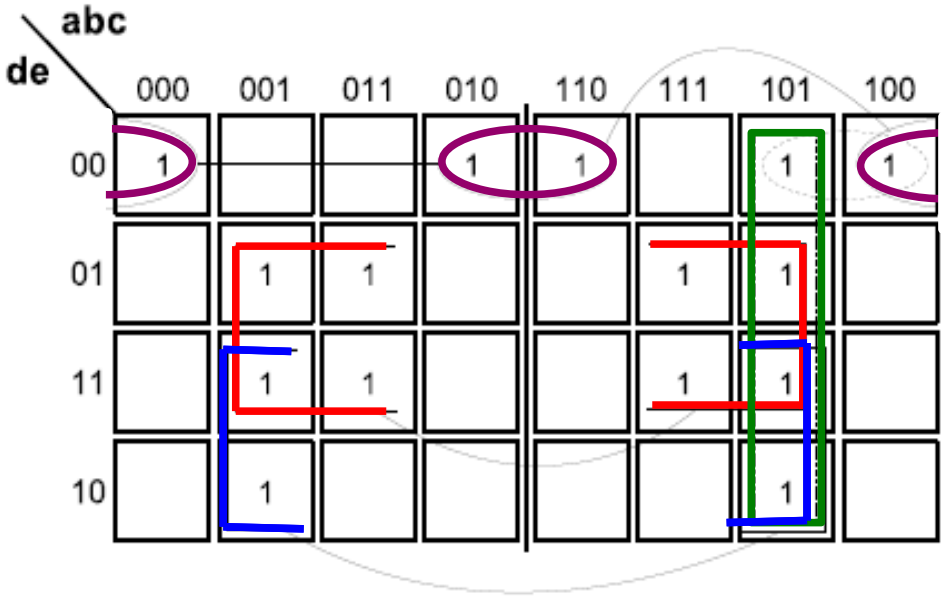
f = bc' + a'b'd' + b'c

Fin. Se han cubierto todos los mintérminos

Diseño de Circuitos Combinacionales

Ejemplos de obtención de la expresión mínima

Ejercicio 5.- f = ∑(0,5,6,7,8,13,15,16,20,21,22,23,24,29,31)



1) La obtención de las implicantes primas en un K-mapa de 5 variables requiere analizar las simetrías entre los sub K-mapas de 4 para cuando la variable más significativa (en este ejemplo es **a**) vale 0 y cuando vale 1.

2) Siga los pasos presentados en las transparencias anteriores.

f = c'd'e' + c e + b' c d + a b'c