03_sintesis_combinacional_karnaugh

June 19, 2026

1 Síntesis de Circuitos Combinacionales y Mapas de Karnaugh

Electrónica Digital — 2º GIERM (Mecatrónica) Apuntes propios · Tema 3 · Enfoque examen (ejemplo resuelto primero)

1.1 0. Mapa mental del tema

Un **circuito combinacional** es aquel cuya salida depende **únicamente de la combinación de entradas en ese instante** (no tiene memoria). El flujo de diseño (síntesis) es siempre el mismo:

```
Especificación   Tabla de verdad   Mapa de Karnaugh   Expresión mínima   Realización (puerta
```

El **mapa de Karnaugh (K-map)** es un método analítico-geométrico para obtener la expresión algebraica **más sencilla** de una función booleana dada por su tabla de verdad. Es la herramienta estrella para minimizar a mano funciones de hasta 4-5 variables.

En este notebook cubrimos:

1. Formas canónicas: suma de minterms (SOP) y producto de maxterms (POS).
2. Minimización algebraica vs. minimización con K-map.
3. Mapas de Karnaugh de 2, 3 y 4 variables (con código de Gray).
4. Implicantes, implicantes primos y agrupaciones.
5. Condiciones indiferentes (*don't care*, X).
6. Ejemplos resueltos completos con **figuras** (mapas dibujados + circuito).

1.2 1. Formas canónicas

Dada una tabla de verdad, cada fila se identifica por el valor decimal del vector de entrada (orden de bits: la variable más a la izquierda es la de mayor peso).

1.2.1 Minterms (m_i) → Suma de productos (SOP / forma canónica de unos)

- Un **minterm** m_i es un producto (AND) de **todas** las variables, donde cada variable aparece negada si vale 0 y sin negar si vale 1 **en esa fila**.
- La función es la **suma (OR) de los minterms donde** $f = 1$:

$$f(a,b,c) = \sum m(i)$$

Ejemplo con 3 variables, fila $abc = 101$ (decimal 5): $m_5 = a\,\bar{b}\,c$.

1.2.2 Maxterms (M_i) → Producto de sumas (POS / forma canónica de ceros)

- Un **maxterm** M_i es una suma (OR) de **todas** las variables, donde cada variable aparece **negada si vale 1** y sin negar si vale 0 (¡al revés que el minterm!).
- La función es el **producto (AND) de los maxterms donde** $f = 0$:

$$f(a, b, c) = \prod M(i)$$

Ejemplo, fila $abc = 101$: $M_5 = (\bar{a} + b + \bar{c})$.

> **Dualidad clave:** $\sum m(\text{unos})$ y $\prod M(\text{ceros})$ describen la **misma** función. Se elige la forma que dé menos puertas. Si hay pocos 1's conviene SOP; si hay pocos 0's conviene POS.

```
Funcion f(a,b,c) = Sigma m(1, 2, 4, 5)\n
a b c | f | minterm/maxterm
-------------------------------------
0  0  0 | 0 | (a+b+c)
0  0  1 | 1 | a'b'c
0  1  0 | 1 | a'bc'
0  1  1 | 0 | (a+b'+c')
1  0  0 | 1 | ab'c'
1  0  1 | 1 | ab'c
1  1  0 | 0 | (a'+b'+c)
1  1  1 | 0 | (a'+b'+c')
\nSOP canonica (Sigma m):  f = a'b'c + a'bc' + ab'c' + ab'c
POS canonica (Pi  M):   f = (a+b+c) . (a+b'+c') . (a'+b'+c) . (a'+b'+c')
```

1.3 2. Minimización: por qué el K-map

La forma canónica es correcta pero **cara** (muchas puertas). Se puede simplificar con álgebra de Boole aplicando, sobre todo, el teorema de **adyacencia / combinación**:

$$X\,Y + X\,\bar{Y} = X \qquad (X + Y)(X + \bar{Y}) = X$$

Es decir: dos términos que difieren en **una sola variable** se funden y esa variable desaparece. Hacerlo a mano es propenso a errores; el **mapa de Karnaugh** automatiza visualmente esa búsqueda de adyacencias.

Idea geométrica: colocamos los 1's de la función en una rejilla ordenada en **código Gray** (filas y columnas adyacentes difieren en un solo bit). Así, **casillas físicamente contiguas** corresponden a minterms que difieren en una variable → se pueden agrupar.

Reglas de agrupación:

- Se agrupan **unos** (para SOP) o **ceros** (para POS).
- Los grupos deben ser **rectángulos** de tamaño potencia de 2: 1, 2, 4, 8, 16 casillas.
- **Cuanto más grandes**, mejor (cada duplicación elimina una variable).
- Los grupos pueden **solaparse** y **dar la vuelta por los bordes** (el mapa es un toroide).
- Hay que **cubrir todos los 1's** con el **menor número de grupos lo más grandes posible**.

1.4 3. Cómo se etiqueta un K-map (código Gray)

Variables	Forma del mapa	Orden Gray de filas/cols
2 (a, b)	2×2	`0, 1`
3 (a, b, c)	2×4	filas `a`=0,1 · cols `bc`= **00, 01, 11, 10**
4 (a, b, c, d)	4×4	filas `ab` y cols `cd` = **00, 01, 11, 10**

La secuencia **00 → 01 → 11 → 10** es Gray: cada paso cambia un solo bit (y 10 vuelve a 00 al cerrar el toroide). Por eso casillas contiguas son adyacentes lógicamente.

A continuación una función que **dibuja** cualquier K-map de 2, 3 o 4 variables a partir de la lista de minterms (1's) y, opcionalmente, de *don't cares* (X).

`Funciones de dibujo de K-map listas.`

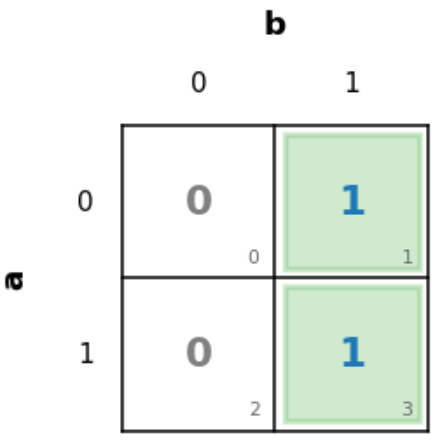
1.5 4. Mapa de 2 variables — ejemplo resuelto

Función: $f(a, b) = \sum m(1, 3)$ → vale 1 en $ab = 01$ y $ab = 11$.

Tabla de verdad:

a	b	f
0	0	0
0	1	1
1	0	0
1	1	1

En el mapa, los dos 1's están en la columna $b = 1$, así que se agrupan en un grupo de 2 → la variable a desaparece. **Resultado:** $f = b$**.** Lo vemos dibujado.



```
Grupo de 2 casillas (columna b=1) → desaparece a → f = b
```

1.6 5. Mapa de 3 variables — ejemplo resuelto

Función: $f(a,b,c) = \sum m(1,2,4,5)$.

Tabla de verdad (orden abc):

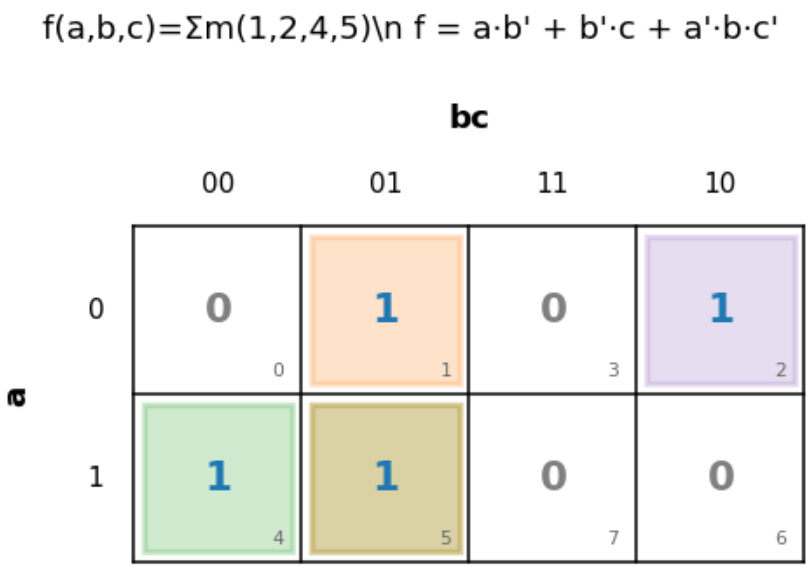
idx	a	b	c	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Agrupaciones (sobre el mapa 2×4 con columnas bc en Gray 00 · 01 · 11 · 10):

- **Grupo A** $= \{m_4, m_5\} \rightarrow$ fila $a = 1$, columnas $bc = 00$ y 01 (cambia c) $\rightarrow$ término $a\,\bar{b}$.
- **Grupo B** $= \{m_1, m_5\} \rightarrow$ $m_1 = 001$ y $m_5 = 101$ son adyacentes en la columna $bc = 01$ (cambia a) $\rightarrow$ término $\bar{b}\,c$.
- **Grupo C** $= \{m_2\}$: $m_2 = 010$. Sus vecinos $m_6 = 110$ y $m_3 = 011$ valen 0, así que m_2 **no se puede agrupar** $\rightarrow$ queda solo $\rightarrow$ término $\bar{a}\,b\,\bar{c}$.

Resultado mínimo: $f = a\bar{b} + \bar{b}c + \bar{a}b\bar{c}$.

> Nota didáctica: m_2 ilustra un **implicante primo esencial de tamaño 1** (casilla solitaria). No siempre se puede agrupar todo en grupos grandes.




```
Grupo a.b'      cubre minterms [4, 5]
Grupo b'.c      cubre minterms [1, 5]
Grupo a'.b.c'   cubre minterms [2]
```

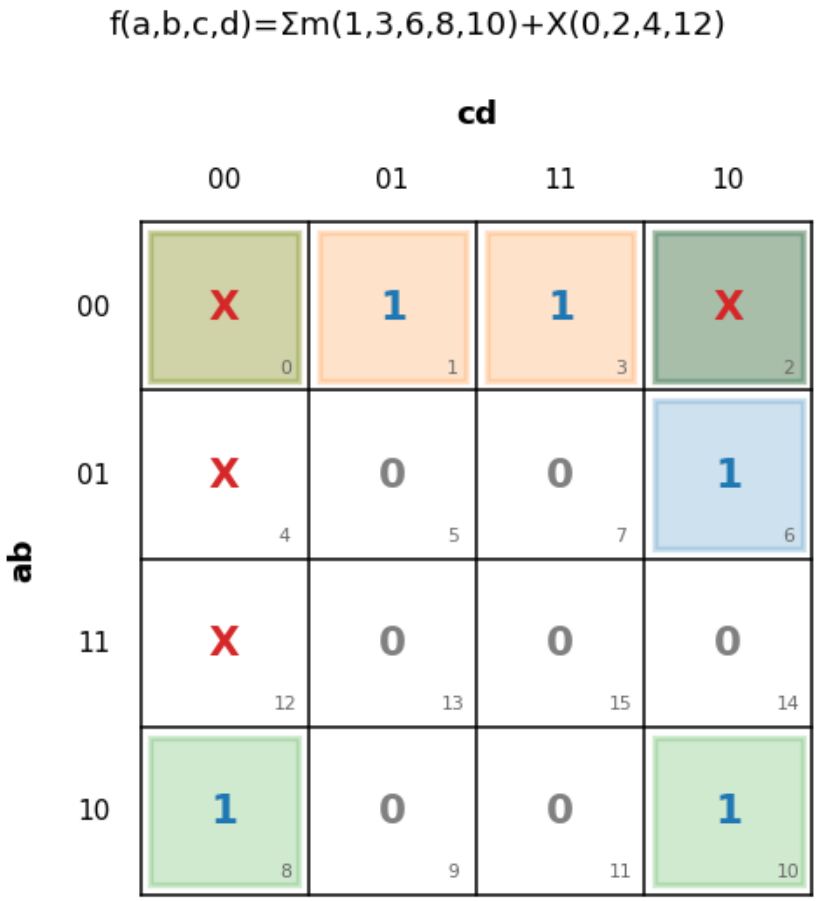
1.7 6. Mapa de 4 variables con *don't care* — ejemplo del PDF

Este es el **ejemplo 9 de las diapositivas** (función de 4 bits con indeterminaciones):

$$f(a,b,c,d) = \sum m(1,3,6,8,10) + \sum X(0,2,4,12)$$

donde X son **condiciones indiferentes** (*don't care*): combinaciones de entrada que **nunca ocurren** o cuya salida **no importa**. Podemos asignarles 0 **o** 1 según convenga para agrandar grupos (¡pero nunca crear un grupo formado solo por X!).

El propio PDF da la solución $f = a'b' + a'd...$ usando algunas X como 1. Más abajo **derivamos y verificamos** la expresión mínima con un minimizador Quine–McCluskey incluido en el notebook, así que no hay que fiarse del dibujo a ojo.




```
1 = azul C0 | X = rojo C3 (don't care) | 0 = gris
```

1.7.1 6.1 Verificación automática de la minimización

Para no fiarnos solo del dibujo, implementamos un **minimizador SOP** sencillo (Quine–McCluskey: genera implicantes primos y busca los esenciales). Lo usamos para **comprobar** que la expresión candidata es correcta sobre los minterms obligatorios y libre en los *don't cares*.

Un **implicante** es un grupo de 1's (potencia de 2) que la función "cubre". Un **implicante primo** es un implicante que **no puede agrandarse** más sin incluir un 0. Un **implicante primo esencial** es el único que cubre algún minterm concreto → **debe** estar en la solución.

```
Implicantes primos encontrados:
  --00  ->  c'd'
  -0-0  ->  b'd'
  0--0  ->  a'd'
  00--  ->  a'b'
\nBuscando implicantes primos esenciales:
  minterm  1: solo lo cubre 00-- (a'b') -> ESENCIAL
```

```
  minterm  6: solo lo cubre 0--0 (a'd') -> ESENCIAL
  minterm 10: solo lo cubre -0-0 (b'd') -> ESENCIAL
\nExpresion minima:  f = a'b' + a'd' + b'd'

idx  abcd  f_objetivo  f_sintetizada  coincide?
  0  0000          X              1        (libre)
  1  0001          1              1        si
  2  0010          X              1        (libre)
  3  0011          1              1        si
  4  0100          X              1        (libre)
  5  0101          0              0        si
  6  0110          1              1        si
  7  0111          0              0        si
  8  1000          1              1        si
  9  1001          0              0        si
 10  1010          1              1        si
 11  1011          0              0        si
 12  1100          X              0        (libre)
 13  1101          0              0        si
 14  1110          0              0        si
 15  1111          0              0        si
\nMinimizacion CORRECTA
```

1.8 7. Minimización por ceros (POS) — ejemplo del PDF

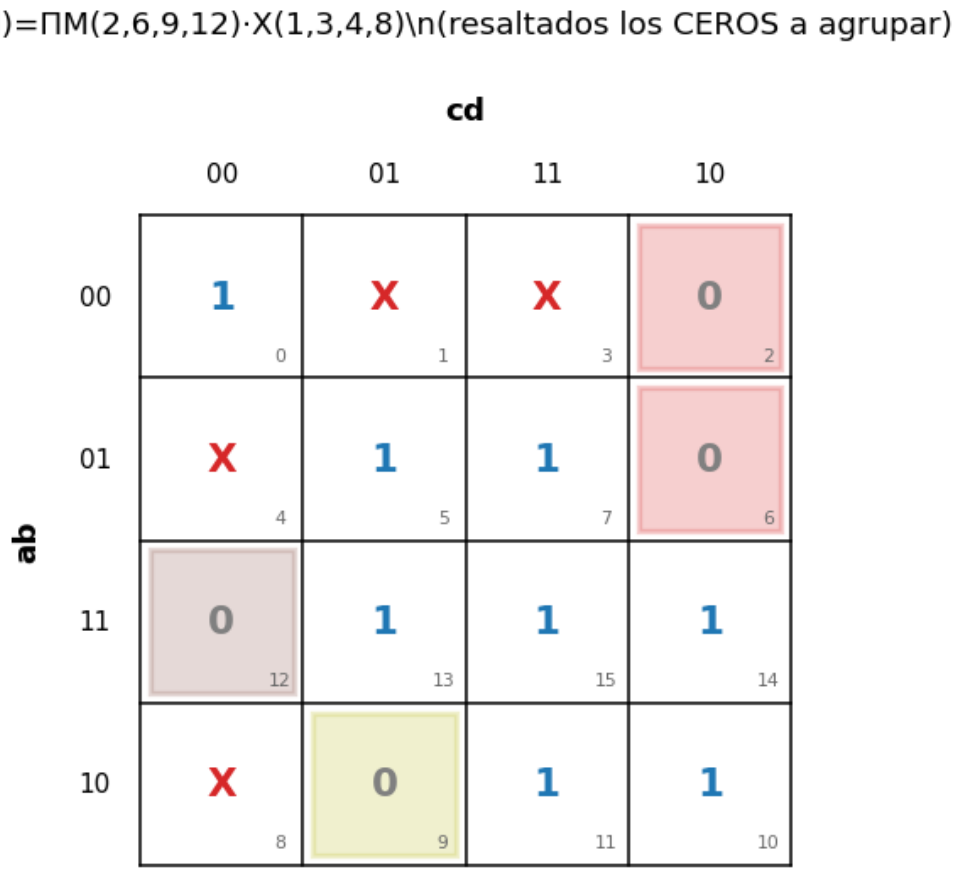
A veces conviene agrupar **ceros**. El ejemplo 2 del PDF:

$$g(a,b,c,d) = \prod M(2,6,9,12) \cdot X(1,3,4,8)$$

Agrupando ceros se obtiene la forma **producto de sumas**. Reglas para POS:

- Para **POS** agrupas **ceros**; cada grupo de ceros da un **maxterm** simplificado (suma OR).
- En el maxterm, una variable aparece **negada si en el grupo vale 1**, sin negar si vale 0 (regla inversa a SOP).

Dibujamos el mapa marcando los ceros agrupados (las casillas grises resaltadas).




```
Para POS se agrupan los 0. Cada grupo de 0 -> un maxterm (suma).
Variable negada si en el grupo vale 1; sin negar si vale 0.
```

1.9 8. Aplicación: realización del circuito (schemdraw)

Una vez minimizada la función, la **realización** es directa: por cada término producto una puerta AND (con las inversiones necesarias) y una OR final que suma los términos.

Dibujamos la realización de la función de 3 variables de la sección 5:

$$f = a\bar{b} + \bar{b}c + \bar{a}b\bar{c}$$

Tres términos AND → una OR de 3 entradas (estructura SOP de 2 niveles AND-OR).

```
Circuito: 3 AND (terminos producto) + 1 OR (suma) -> SOP de 2 niveles.
```

1.10 9. Transitorios en circuitos combinacionales (glitches)

Tema final del PDF: aunque el circuito sea correcto en régimen permanente, **mientras las señales se propagan** por las puertas pueden aparecer **transiciones espurias** llamadas **glitches** (picos

no deseados) por las diferencias de retardo entre caminos.

Definición de retardo del circuito: el **camino de mayor retardo** entre una entrada y una salida.

Dos soluciones:

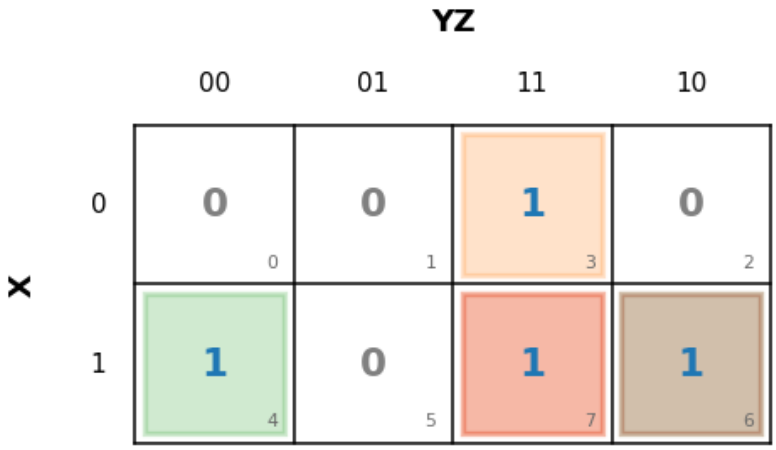
1. **Añadir puertas redundantes** (un implicante de consenso que tape la transición). En el ejemplo del multiplexor $S = X\bar{Z} + YZ$ aparece un glitch al pasar $Z : 1 \to 0$ con $X = Y = 1$; se corrige añadiendo el término de consenso:

$$S = X\bar{Z} + YZ + XY$$

2. **Esperar** un tiempo antes de leer la salida (sincronizar con un reloj → circuitos secuenciales).

El K-map del consenso (XY "puentea" los dos grupos adyacentes que no se solapaban):

Mux S = X·Z' + Y·Z + X·Y\n(grupo rojo X·Y redundante: tapa el glitch)




```
  X.Z'
  Y.Z
  X.Y (consenso, evita glitch)
\nEl termino de consenso X.Y solapa los dos grupos adyacentes -> sin
transitorio.
```

1.11 10. Resumen para el examen

Concepto	Regla rápida
Minterm m_i	AND de todas las variables; negada si bit=0. Suma de minterms donde $f = 1 \to$ SOP
Maxterm M_i	OR de todas las variables; negada si bit=1. Producto de maxterms donde $f = 0 \to$ POS

Concepto	Regla rápida
K-map	Rejilla en código Gray ($00 \cdot 01 \cdot 11 \cdot 10$); casillas contiguas = adyacentes lógicas
Agrupar	Rectángulos de 2^k casillas, lo más grandes posible, pueden solaparse y dar la vuelta
SOP	Agrupa **1's**; cada grupo → AND; variable que cambia dentro del grupo desaparece
POS	Agrupa **0's**; cada grupo → OR (maxterm); variable negada si vale 1
Implicante primo	Grupo de 1's que no se puede agrandar más
Esencial	Único implicante primo que cubre un minterm concreto → obligatorio
Don't care (X)	Vale 0 o 1 a conveniencia; úsalo para agrandar grupos, **no** para crear grupos solo de X
Glitch	Transición espuria por retardos; se evita con término de consenso o reloj

Flujo de síntesis: Especificación → Tabla de verdad → K-map → Expresión mínima (SOP/POS) → Realización con puertas (2 niveles AND-OR / OR-AND).

Apuntes generados a partir de `ED3 Síntesis de Circuitos Combinacionales.pdf` (Universidad de Sevilla). Todas las figuras se dibujan con matplotlib/schemdraw y la minimización se verifica con un minimizador Quine–McCluskey incluido en el propio notebook.