

4. Implementación optimizada de funciones lógicas

angel@uc.cl

IEE2713 Sistemas Digitales



4.1 – 4.5

Introducción a la optimización lógica

angel@uc.cl

IEE2713 Sistemas Digitales

4.1 Mapas de Karnaugh

Row number	x_1	x_2	x_3	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Figure 2.48 The function $f(x_1, x_2, x_3) = \Sigma\, m(0, 2, 4, 5, 6)$.

4.1 Mapas de Karnaugh

- Forma alternativa para representar una tabla de verdad
- Minterm adyacentes pueden ser combinados (bajo ciertas reglas)
- Cada una de las celdas del mapa corresponde a una fila de la tabla de verdad
- Si hay N variables, habrá 2^N celdas
  - OK para N≤5, luego se vuelve difícil de manejar

x_1	x_2	
0	0	m_0
0	1	m_1
1	0	m_2
1	1	m_3

(a) Truth table

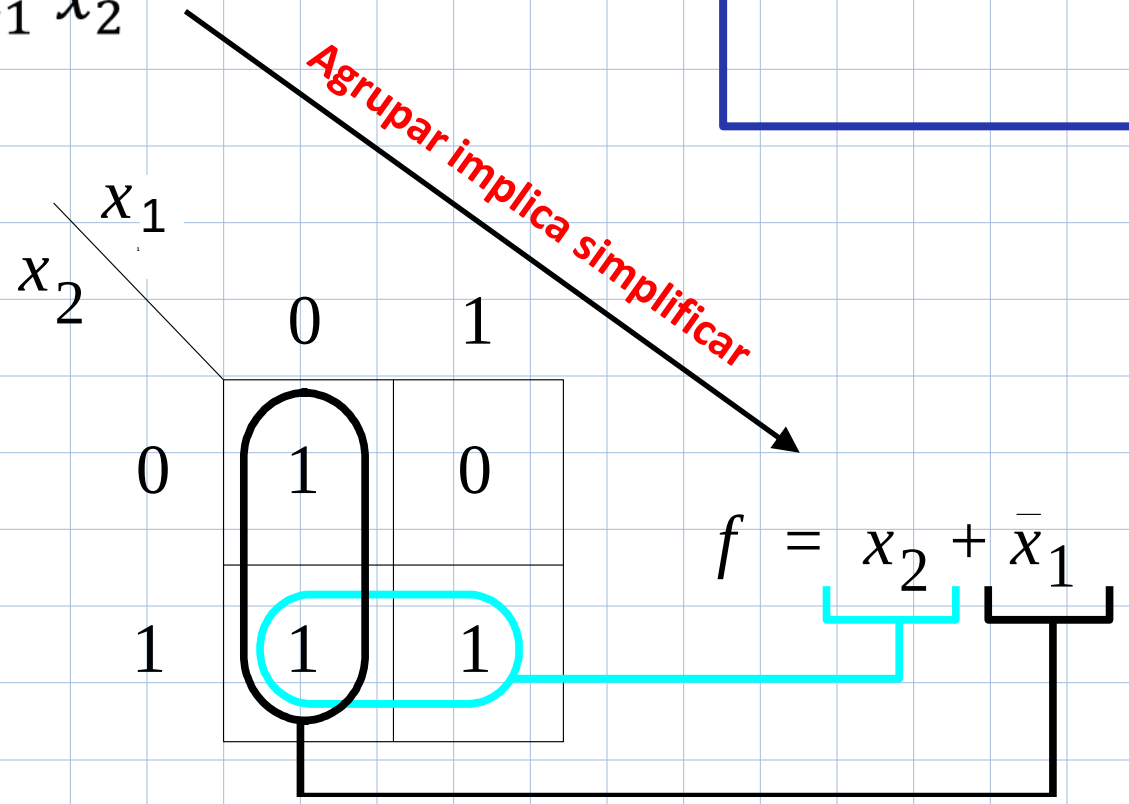
x_2 \ x_1	0	1
0	m_0	m_2
1	m_1	m_3

(b) Karnaugh map

Ejemplo 1 – Dos variables

- $f(x_1, x_2) = x_1 x_2 + \overline{x_1} x_2 + \overline{x_1}\ \overline{x_2}$

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	1
1	0	0
1	1	1



Para simplificar las funciones, intentamos crear grupos grandes de celdas adyacentes de manera de lograr expresiones más compactas

(Paréntesis: Distancia Hamming)

- Más adelante veremos que podemos expresar números en base binaria usando dígitos binarios o bits
  - Ejemplos: 00101110 o 11101010
- La “distancia Hamming” entre dos números binarios es el número de dígitos diferentes entre dos palabras digitales
  - Ejemplos:
    - ¿Distancia Hamming entre 00101111 y 00001111?
    - ¿Distancia Hamming entre 00101101 y 00101101?
- Existen formas de escribir (codificar) números binarios de manera que la distancia Hamming entre dos números **consecutivos** es siempre 1
  - Se llaman códigos Gray y tienen muchas aplicaciones

Ejemplo 2 – Minterms para K-maps de tres variables

x_1	x_2	x_3	
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

(a) Truth table

x_3 \ x_1x_2	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

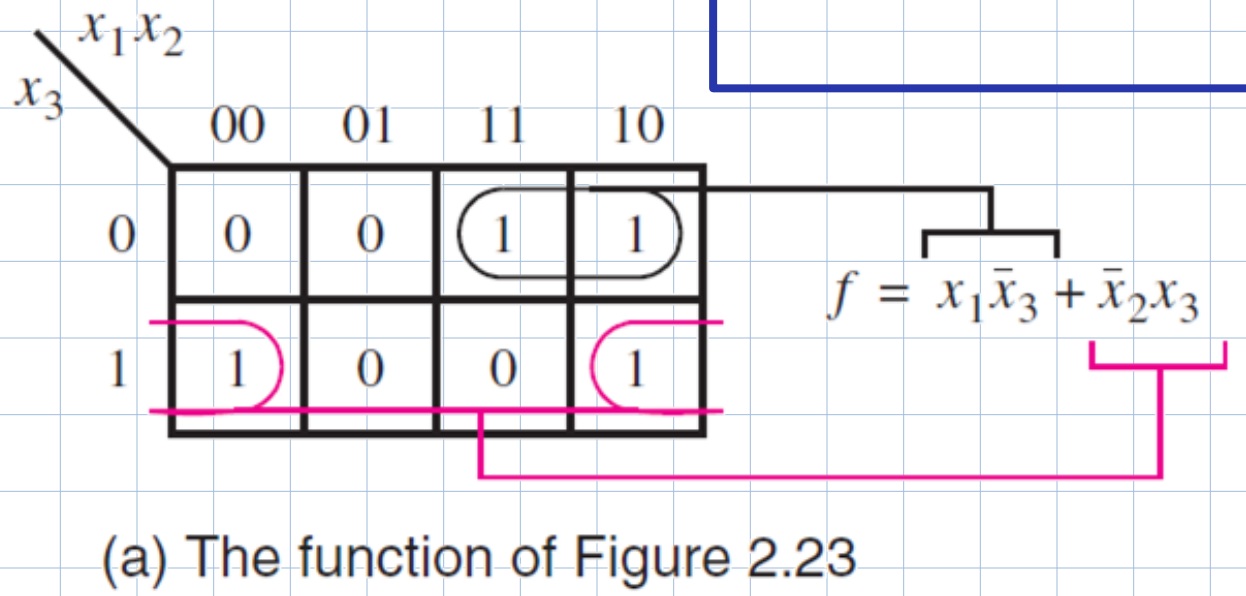
(b) Karnaugh map

El orden de los minterms es poco intuitivo…

¿Por qué?

Ejemplo 2.1 – Tres variables

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
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	0

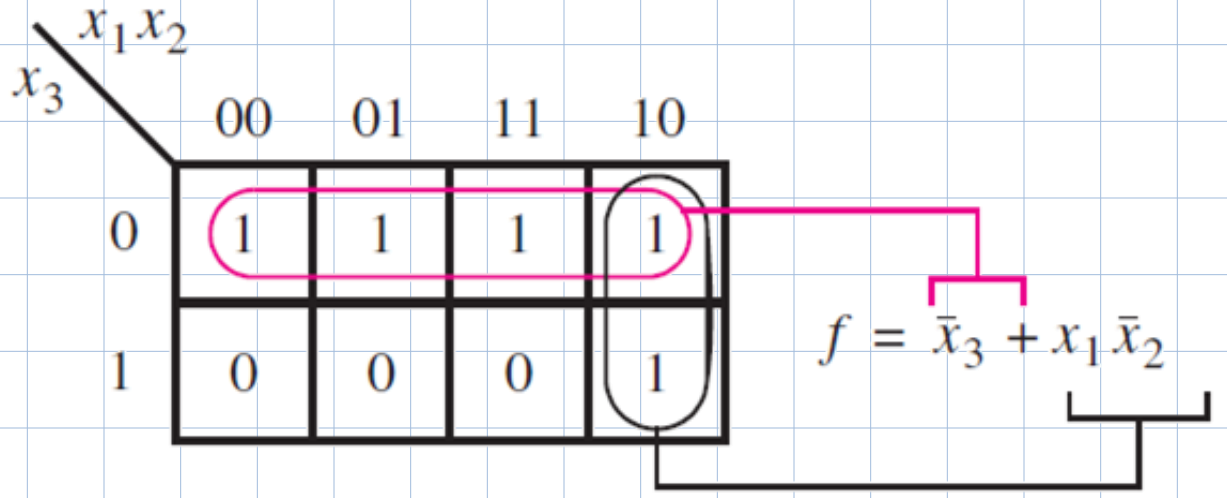


(a) The function of Figure 2.23

Observamos que la distancia hamming entre las celdas del extremo derecho y el izquierdo es 1
Agrupamos variables aparentemente lejanas (pero son adyacentes)

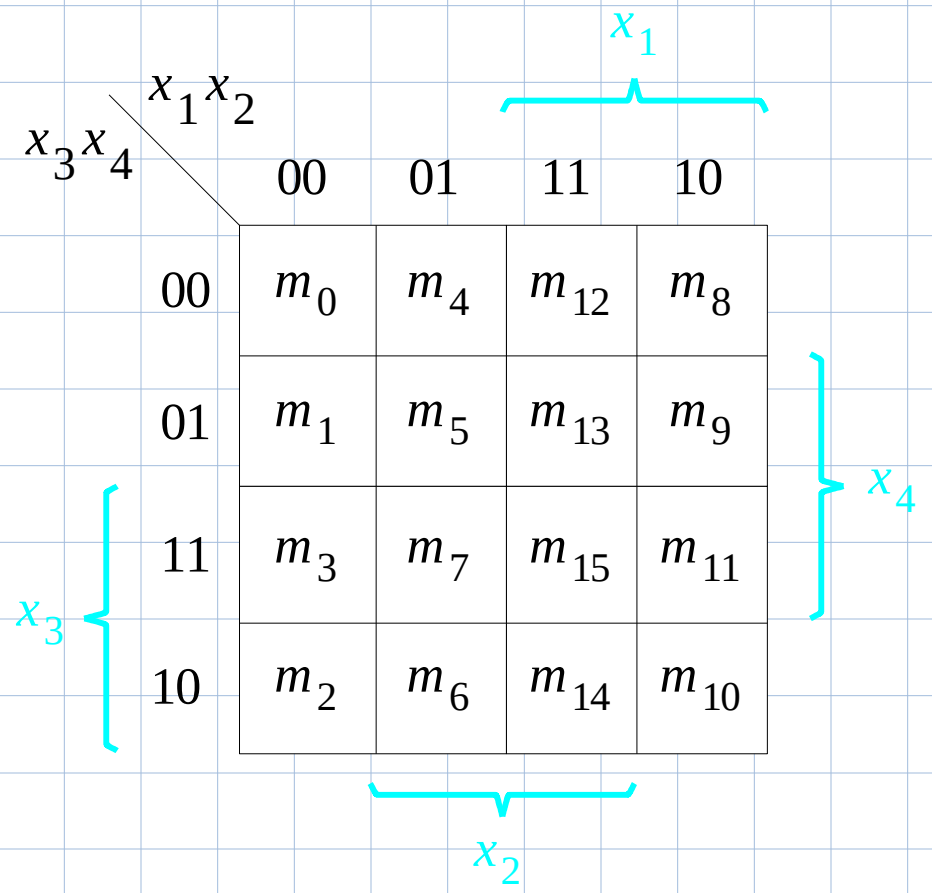
Ejemplo 2.2 – Tres variables

Row number	x_1	x_2	x_3	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0



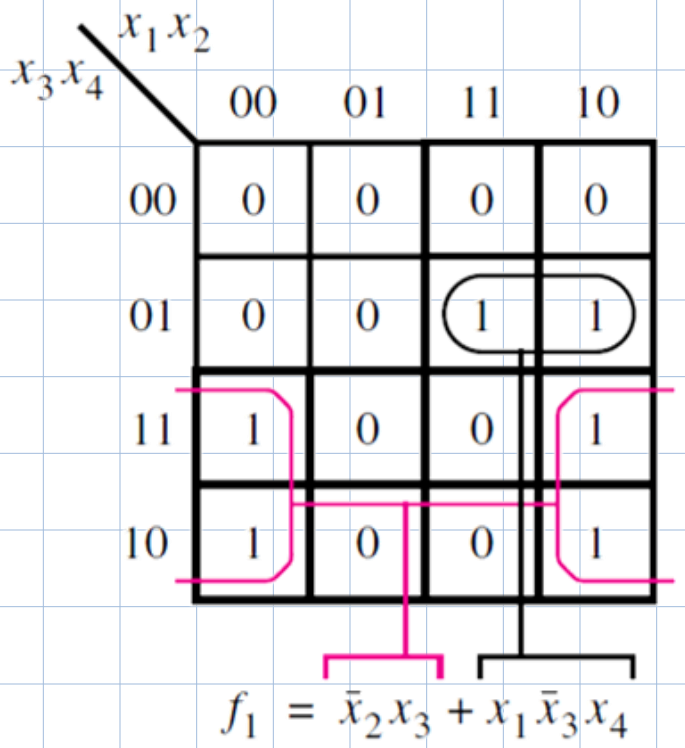
(b) The function of Figure 2.48

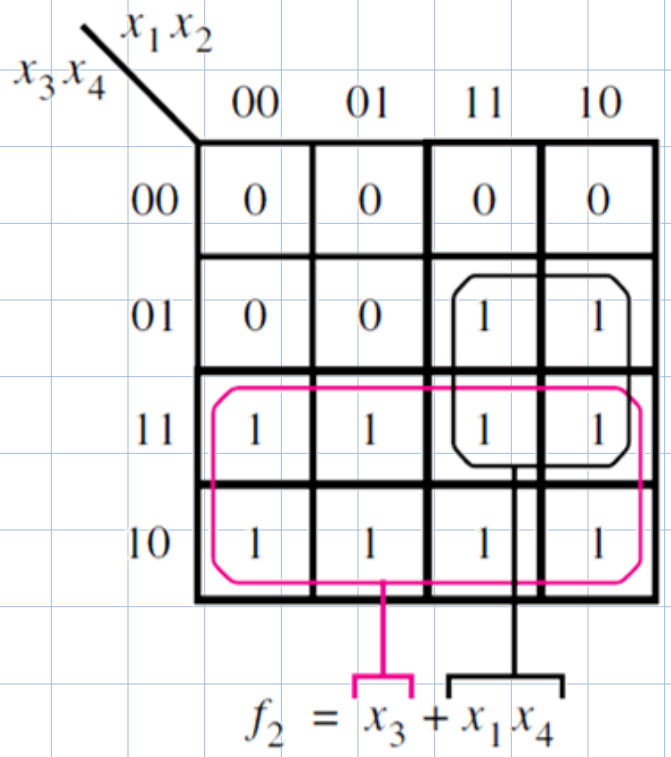
4.1 Mapas de Karnaugh – Cuatro variables



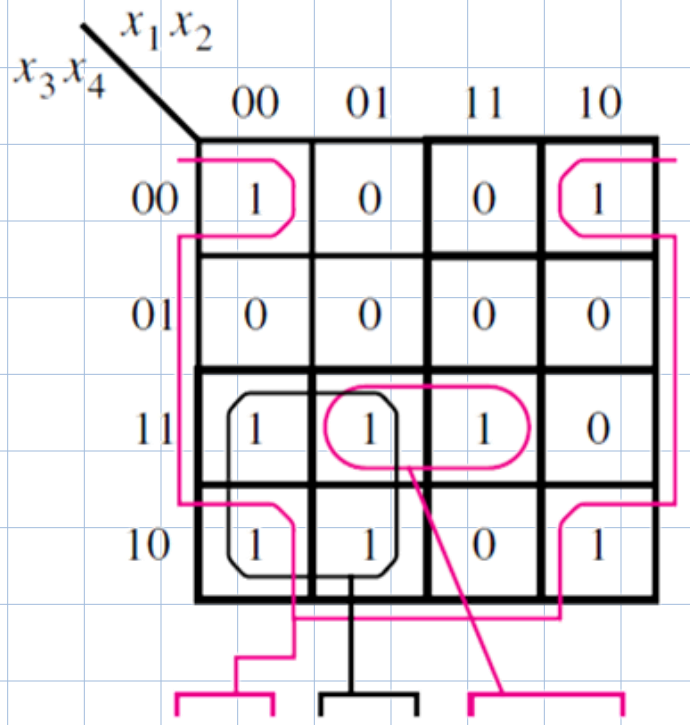
Observemos la ubicación de los minterms, celdas adyacentes mantienen distancia Hamming 1

Ejemplo 2.3 – Cuatro variables

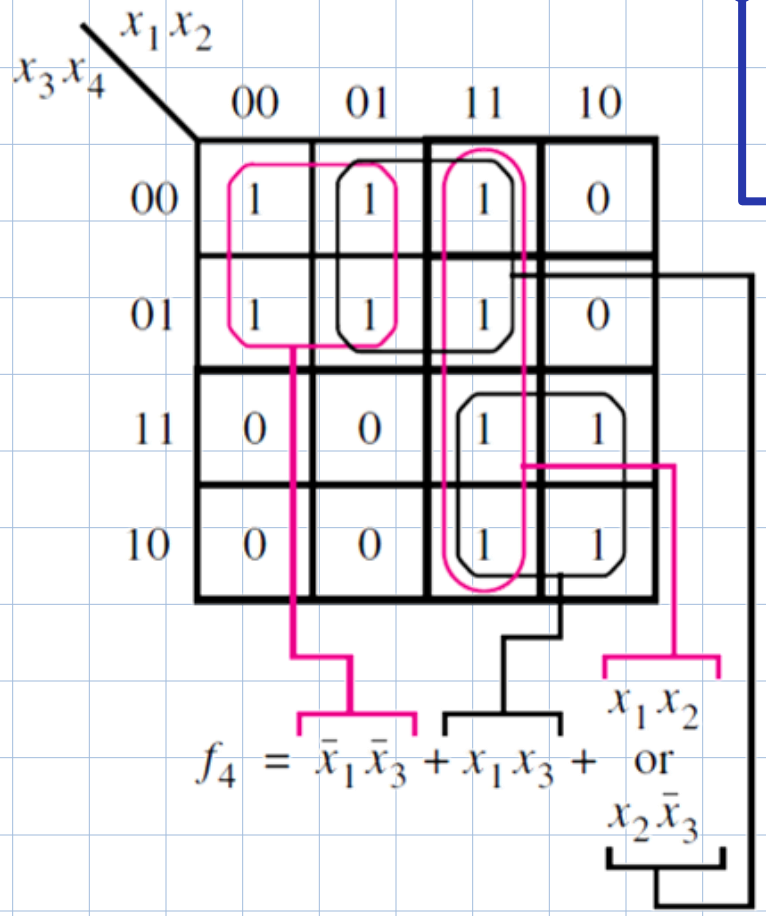




Ejemplo 2.4 – Cuatro variables



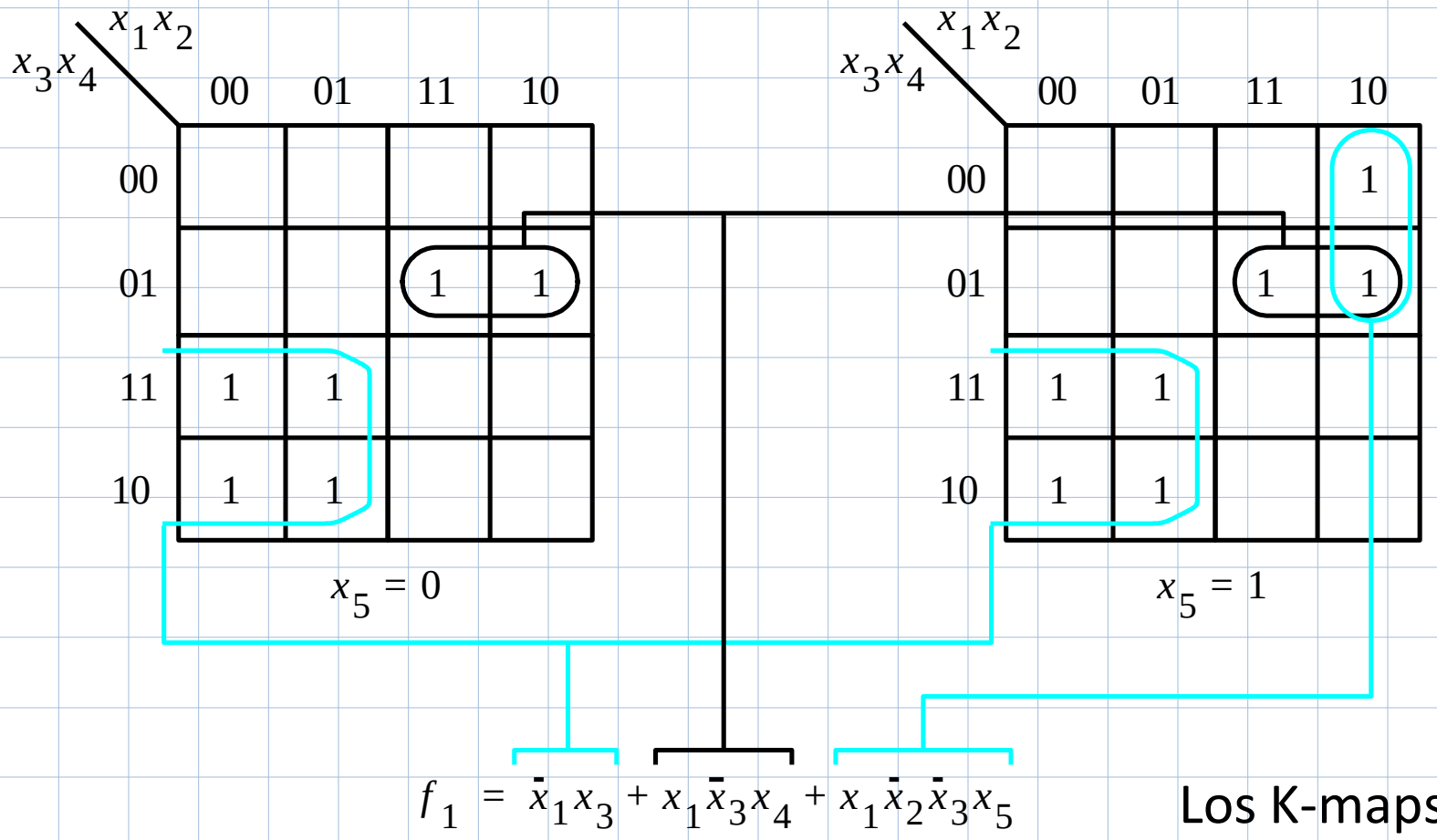
$$f_3 = \bar{x}_2\bar{x}_4 + \bar{x}_1x_3 + x_2x_3x_4$$



$$f_4 = \bar{x}_1\bar{x}_3 + x_1x_3 + \begin{matrix} x_1x_2 \\ \text{or} \\ x_2\bar{x}_3 \end{matrix}$$

Observar que las 4 esquinas son adyacentes!

Ejemplo 2.5 – Cinco variables



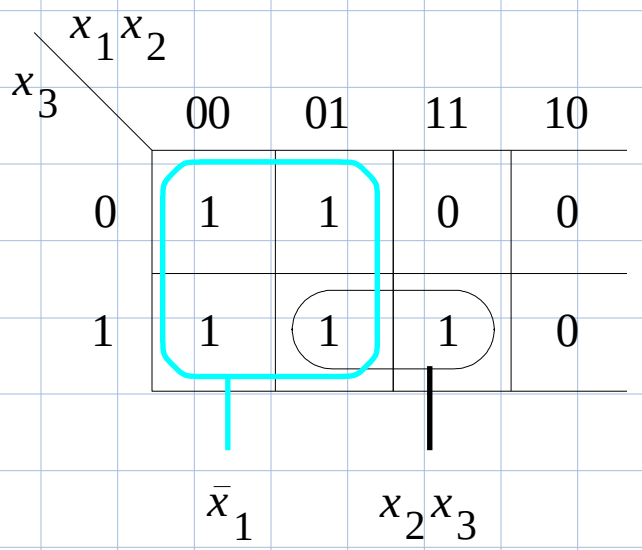
Los K-maps para 5 variables están separados en dos planos

4.2 Estrategia de minimización

- Usar mapa de Karnaugh para un número de variables N > 5 es complicado
- El siguiente es un método organizado para encontrar la implementación de menor costo
- Similar a las técnicas automatizadas para herramientas CAD (que no veremos en el curso)

4.2.1 Terminología

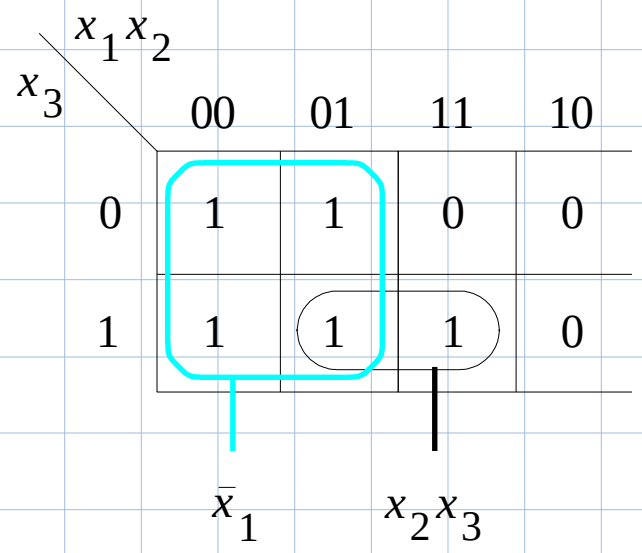
- **Literal**: es el nombre que tiene cada aparición de cada variable en un producto, directa o negada
  - Ej: el implicante $x_5\bar{x}_2x_3$ tiene 3 literales
- **Implicantes**: son productos para los cuales una función dada es igual a 1. En un mapa de Karnaugh, los implicantes serían todos los posibles grupos de 1s.
  - Ej: Esta función tiene **11** implicantes
    - Cada minterm correspondiente a una celda (grupos de uno, 5)
    - Cada par de minterm adyacentes que se puede agrupar en parejas (grupos de dos, 5)
    - Cada grupo de cuatro minterm (1)



4.2.1 Terminología

- **Implicante primo:** Implicante al que no es posible eliminar un literal y obtener otro implicante válido. En un mapa de Karnaugh representan los grupos más grande de 1s posibles
  - Ej: $\bar{x_1}$ y x_2x_3
- **Cover**: conjunto de implicantes que contiene todas las valoraciones para las que una función sea igual a 1
  - Una función puede ser representada por más de un cover
  - Ejemplo: todos estos covers son válidos

- $f(x_1, x_2, x_3) = x_1x_2x_3 + \overline{x_1}x_2x_3 + \overline{x_1}\,x_2\overline{x_3} + \overline{x_1}\,\overline{x_2}x_3 + \overline{x_1}\,\overline{x_2}\,\overline{x_3}$
- $f(x_1, x_2, x_3) = x_2x_3 + \overline{x_1}x_2 + \overline{x_1}\,\overline{x_2}$
- $f(x_1, x_2, x_3) = x_2x_3 + \overline{x_1}$



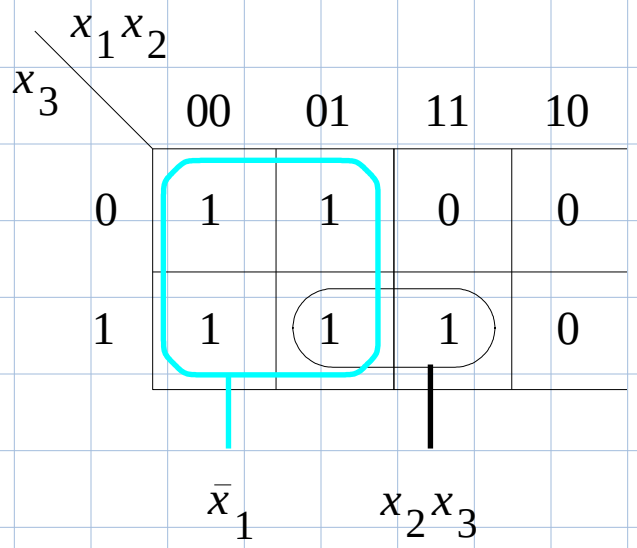
4.2.1 Terminología

- **Costo:** Número de compuertas lógicas sumado con el número de entradas total a las compuertas
  - Se asume que todos los literales están disponibles en sus formas natural y negada (complementada)
  - Ej: costos de las siguientes funciones:
    - $f = x_4x_3 + \overline{x_1}x_2$ tiene un costo de 9
    - $g = \overline{(x_1\overline{x_4} + x_2)}(x_3 + \overline{x_5})$ tiene un costo de 14

4.2.2 Procedimiento de minimización

- Para circuitos más grandes minimizar el costo se vuelve se suma relevancia
- El mínimo costo se obtiene cuando el cover está formado con **implicantes primos esenciales**
- Los implicantes primos esenciales son los implicantes que contienen un minterm (para f=1) que **no está incluído dentro de otro implicante primo**
  - Si no incluyo un implicante primo esencial, no puedo realizar la función

Ejemplo – implicantes primos esenciales



Implicante primo Esencial para:

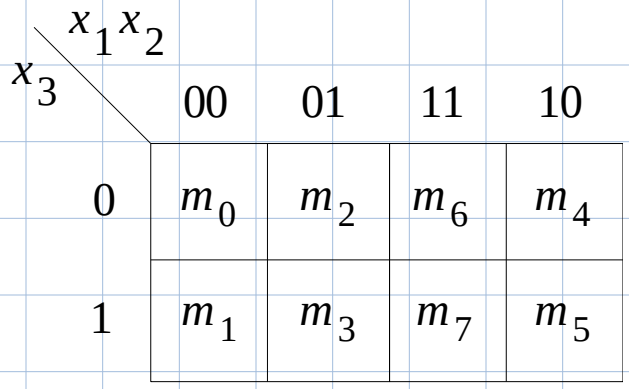
m_7

m_0, m_1, m_2

x_1	x_2	x_3		
0	0	0	m_0	→ 1
0	0	1	m_1	→ 1
0	1	0	m_2	→ 1
0	1	1	m_3	→ 1
1	0	0	m_4	
1	0	1	m_5	
1	1	0	m_6	
1	1	1	m_7	→ 1

(a) Truth table

Recordatorio:



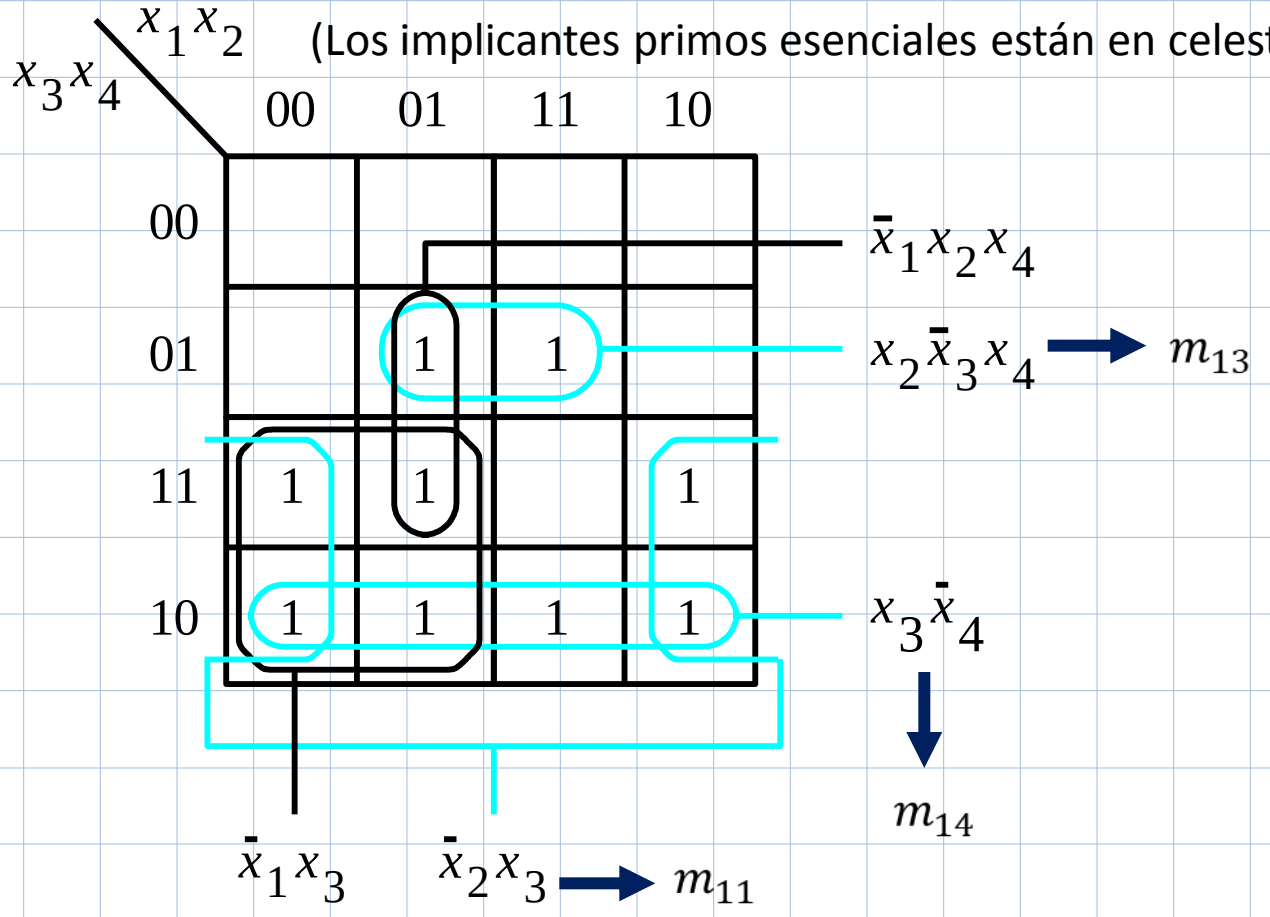
(b) Kamaugh map

Procedimiento de minimización -- Receta

1. Obtener todos los implicantes primos para la función dada
2. Reconocer el conjunto de implicantes primos esenciales
3. Si este conjunto contiene todos los minterms, entonces ése es el cover de mínimo costo.
   - En caso de que no incluya todos los minterm, elija implicantes primos no esenciales que incluyan a los minterm faltantes, discriminando de acuerdo al cover de mínimo costo

Ejemplo – Procedimiento de minimización

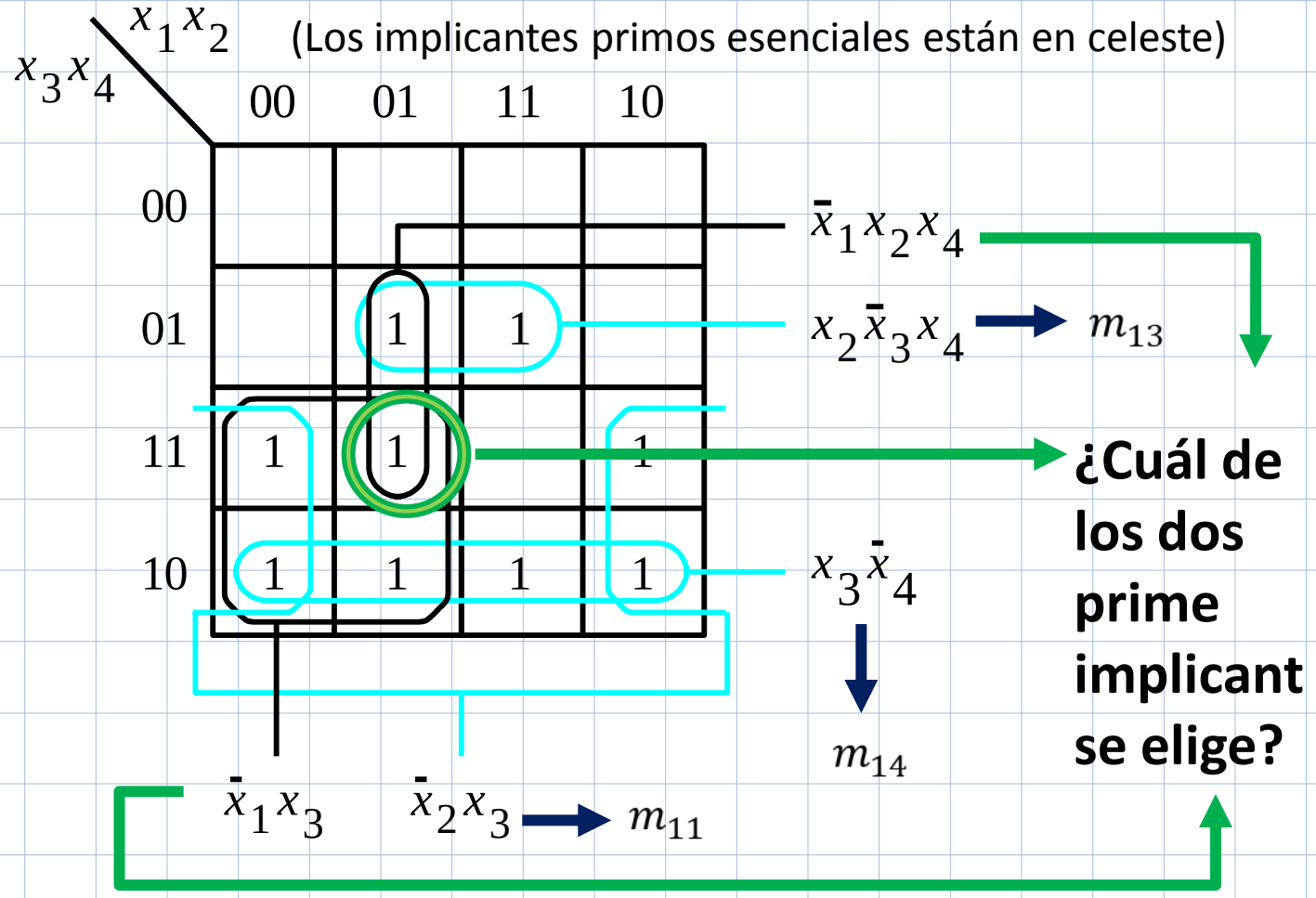
(Los implicantes primos esenciales están en celeste)



Recordatorio:

x_3x_4 \ x_1x_2	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

Ejemplo – Procedimiento de minimización

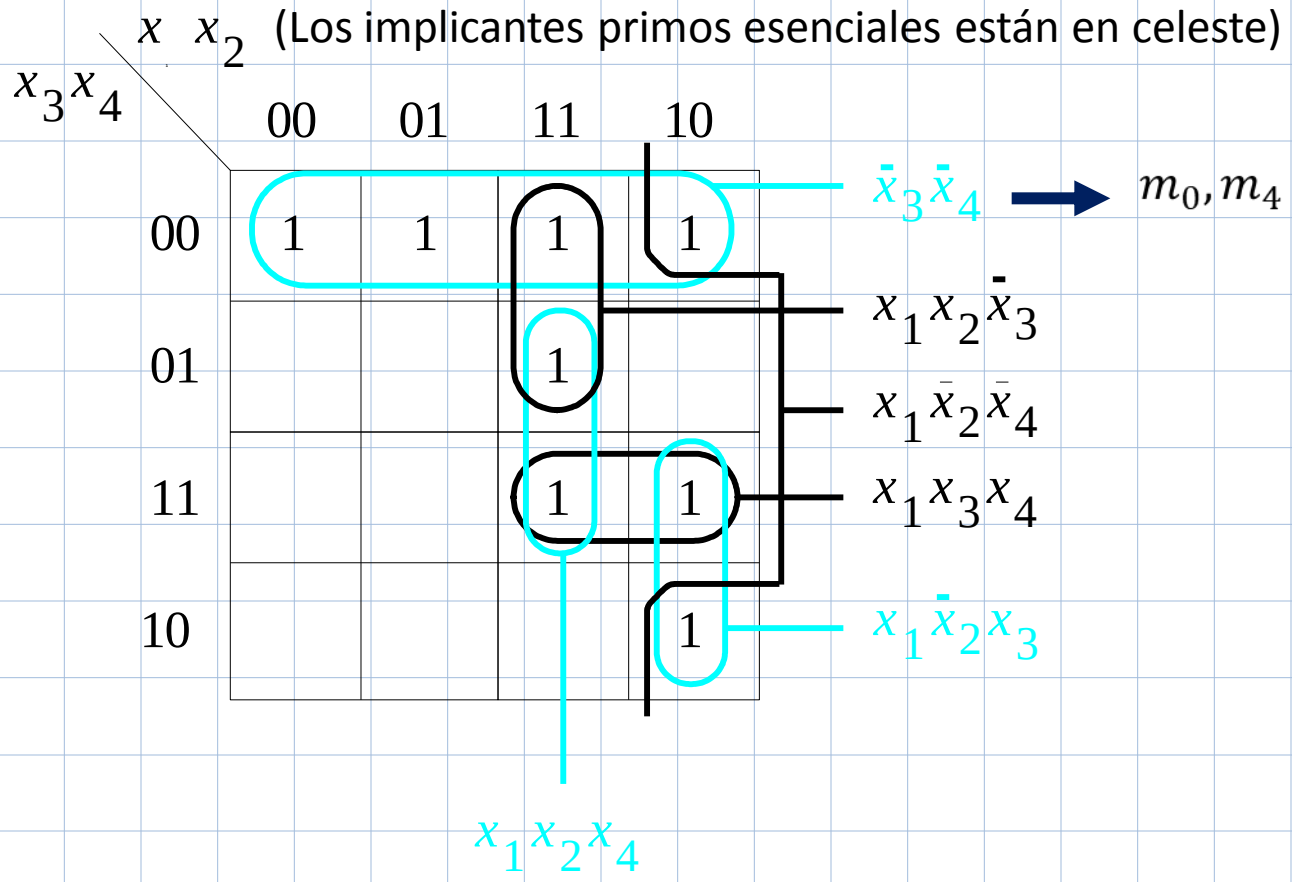


Recordatorio:

x_3x_4 \ x_1x_2	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

Ejemplo – Procedimiento de minimización

(Los implicantes primos esenciales están en celeste)

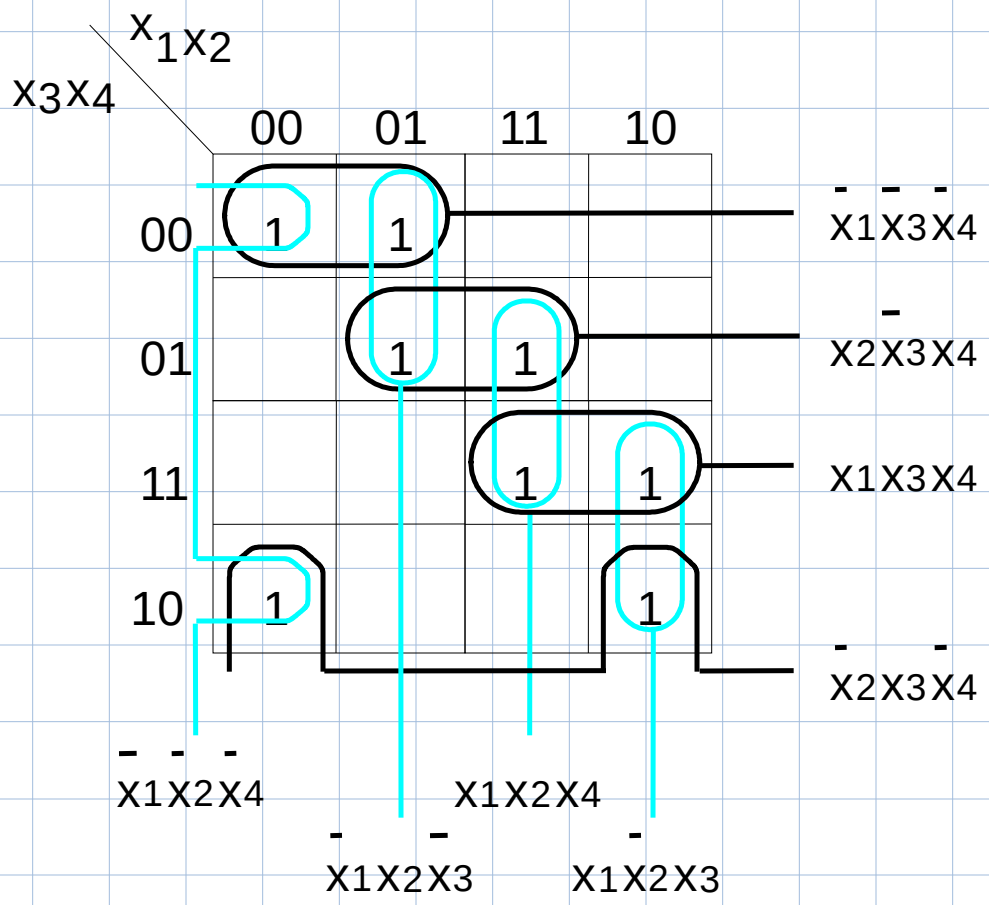


Recordatorio:

x_3x_4 \ x_1x_2	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

No siempre resulta obvio qué cover elegir para lograr mínimo costo

Ejemplo – Procedimiento de minimización



En algunos casos ni siquiera tenemos que incluir implicantes primos esenciales

4.3 Minimización de productos de sumas (POS)

- POS y SOP de una función no siempre tienen el mismo costo
- Una forma de encontrar el POS es encontrar las sumas más largas que cubran todos los maxterm para f = 0
- Otra forma de encontrar el POS de una función es encontrando la SOP **para el complemento de esa función** y luego aplicar el Teorema de DeMorgan (y entendiendo que f = ~~f)

Ejemplo 3.1 - POS

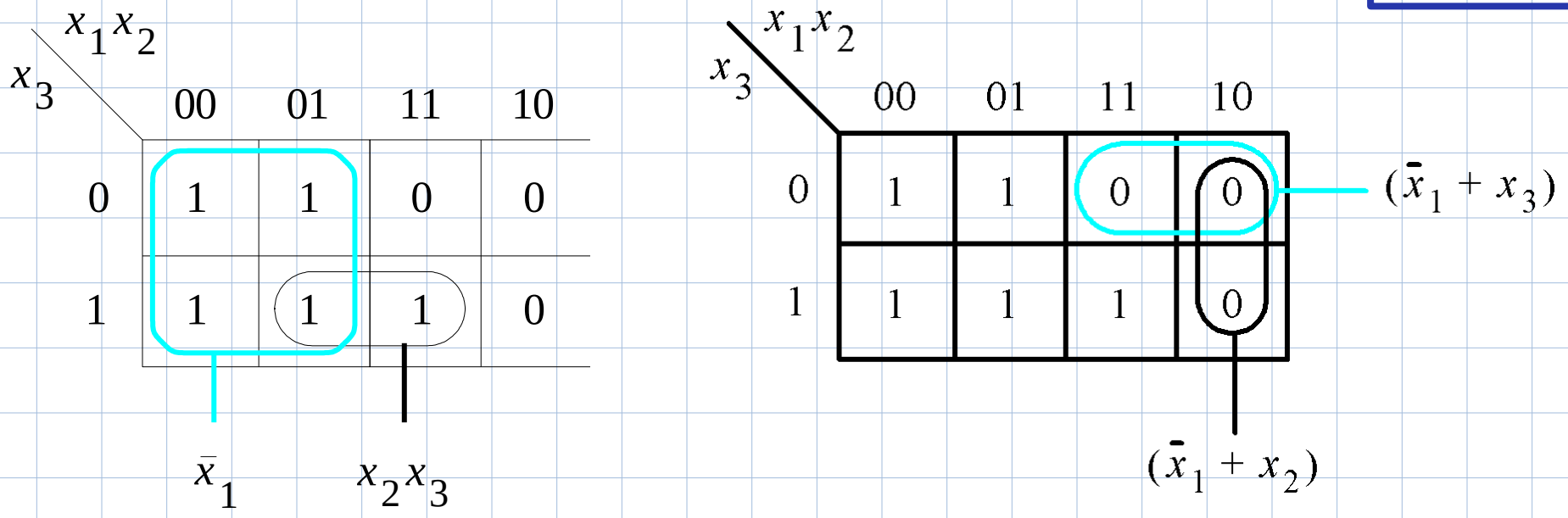
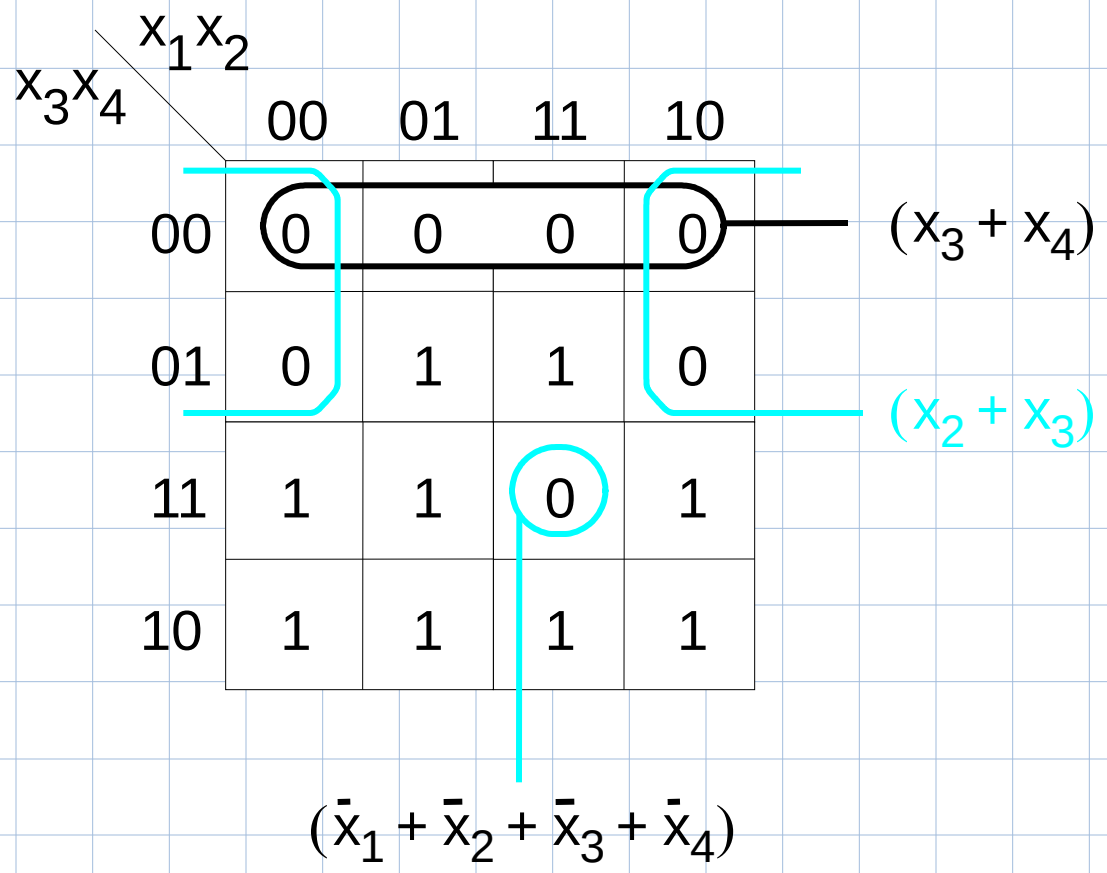


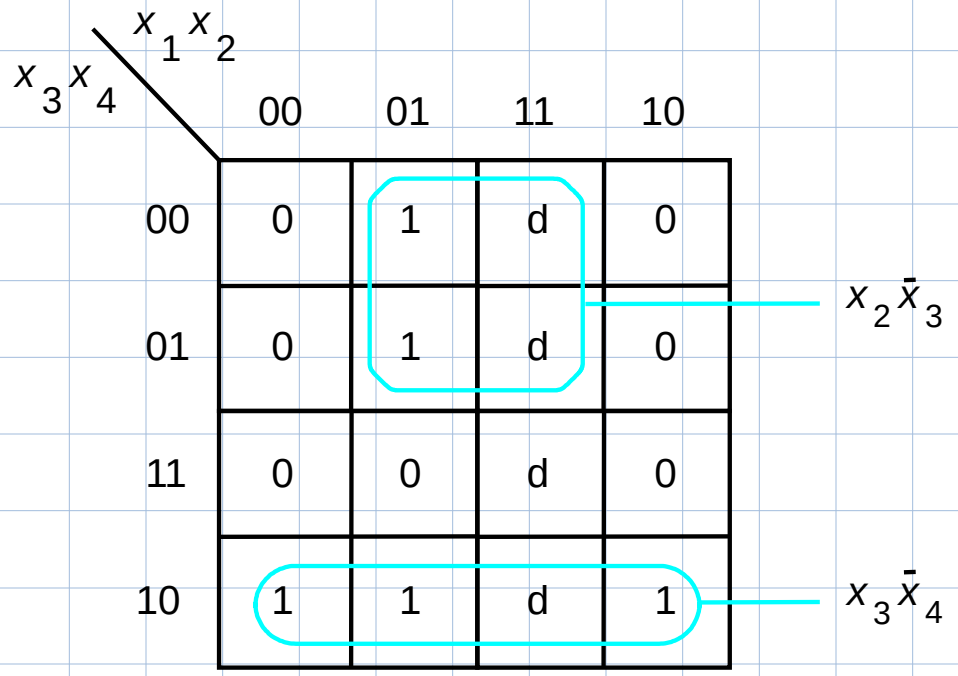
Figure 2.60. POS minimization of $f(x_1, x_2, x_3) = \Pi M(4, 5, 6)$.

Ejemplo 3.2 - POS

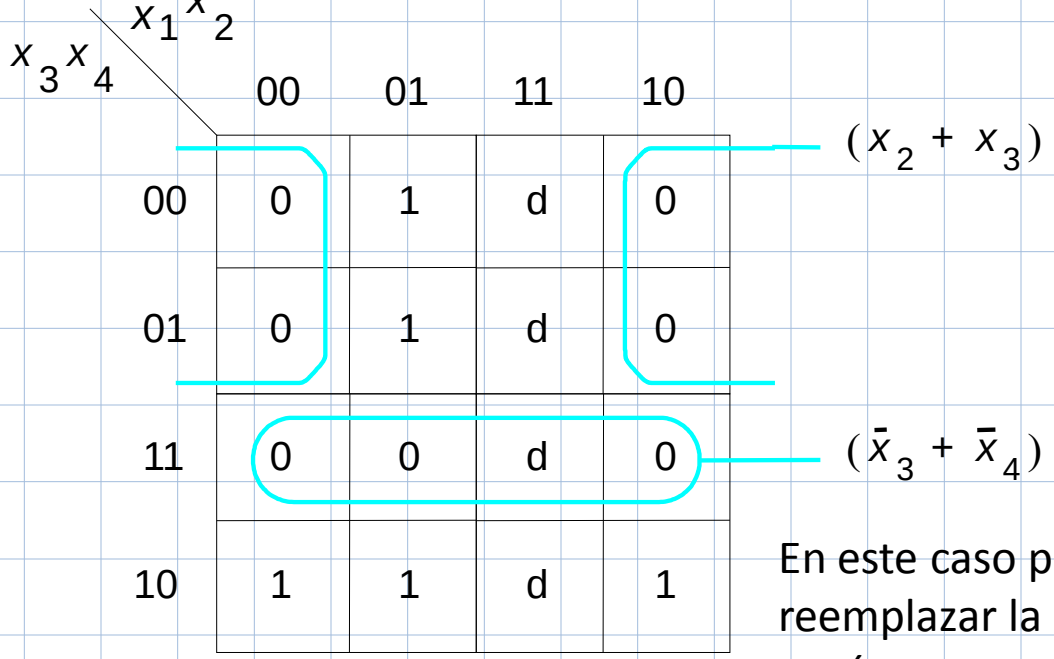


4.4 Funciones especificadas en forma incompleta

- La función tiene salidas no definidas, donde no importa ("don't care", *d*) la salida



(a) SOP implementation



(b) POS implementation

En este caso podemos reemplazar la d por 1 o 0 según nos convenga para lograr la minimización

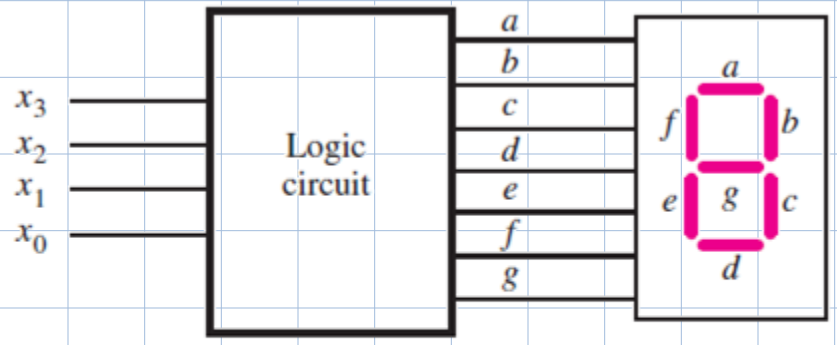
4.5 Circuitos con múltiples salidas

- A veces podemos compartir algunas de las compuertas en diferentes salidas

	x_3	x_2	x_1	x_0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

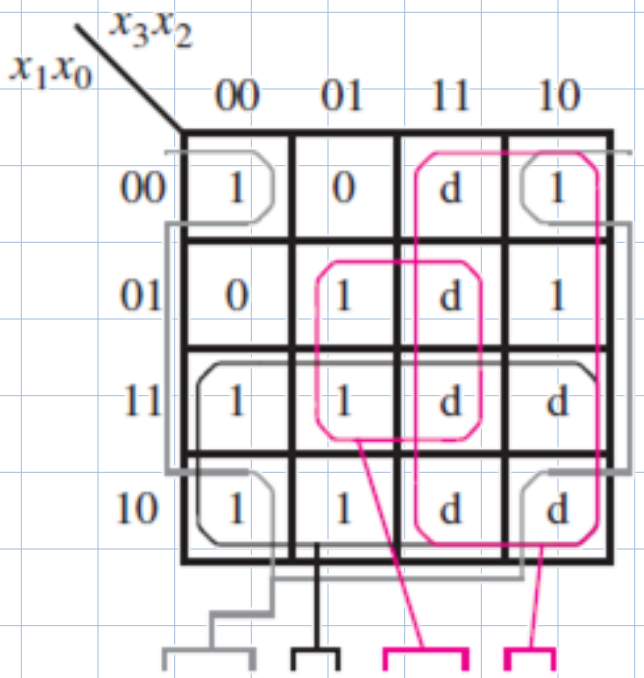
(b) Truth table

Ejemplo – BCD a 7 segmentos

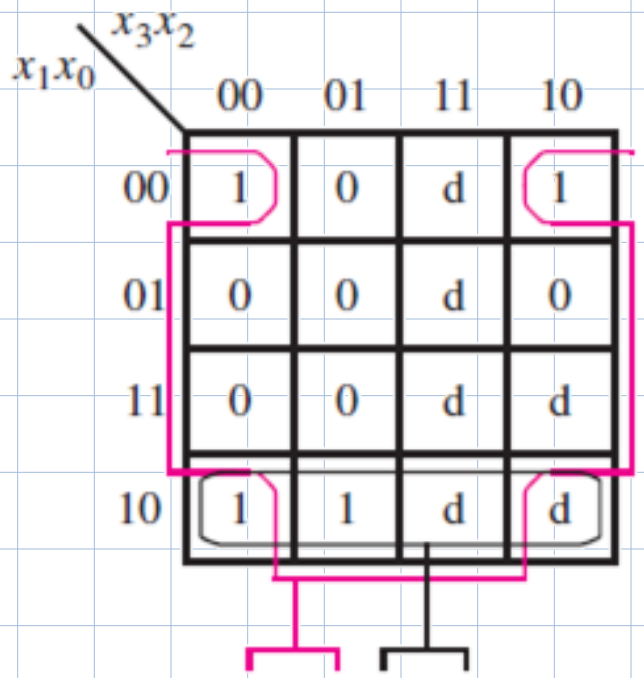


(a) Logic circuit and 7-segment display

4.5 Circuitos con múltiples salidas



$a = \bar{x}_2\bar{x}_0 + x_1 + x_2x_0 + x_3$



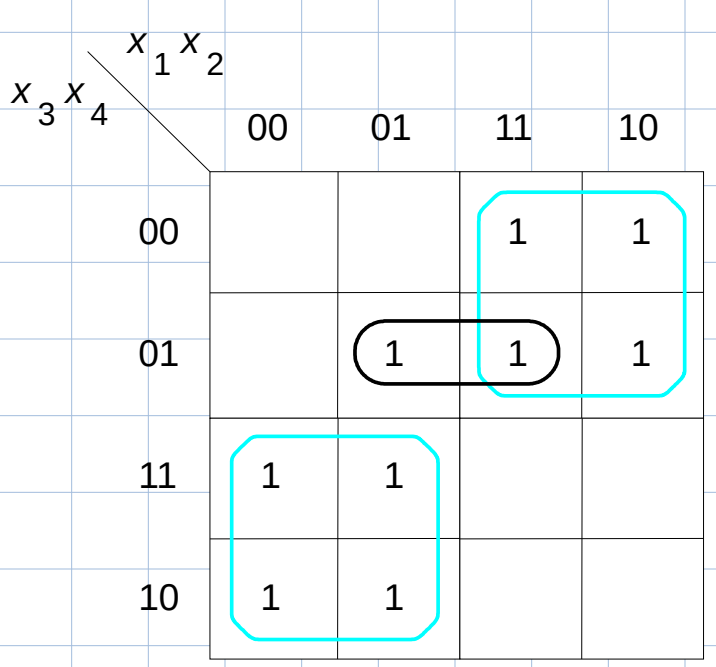
$e = \bar{x}_2\bar{x}_0 + x_1\bar{x}_0$

(c) The Karnaugh maps for outputs a and e.

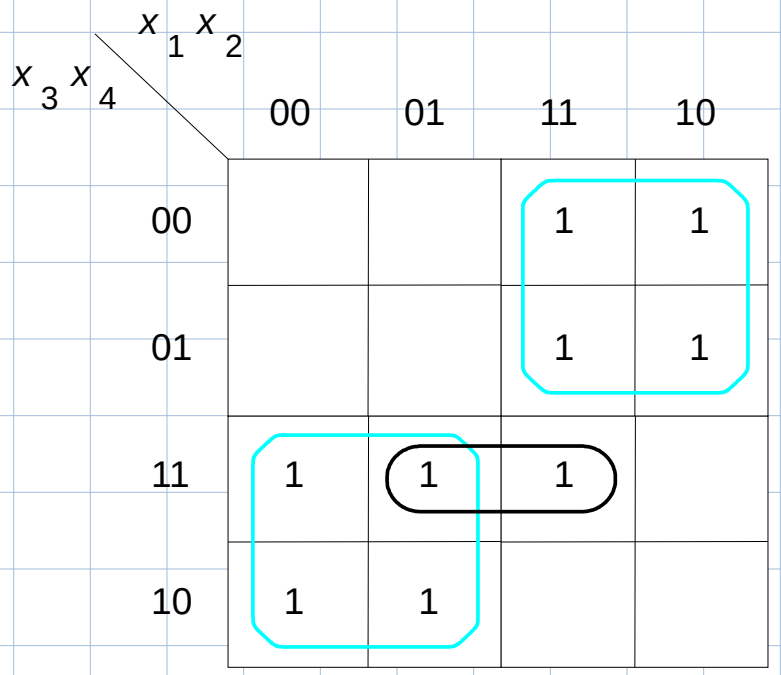
En este caso podemos reutilizar el minterm ~x_2 ~x_0

Ejemplo 4.1 – Múltiples salidas

Las funciones f1 y f2 comparten dos minterms



(a) Function f_1

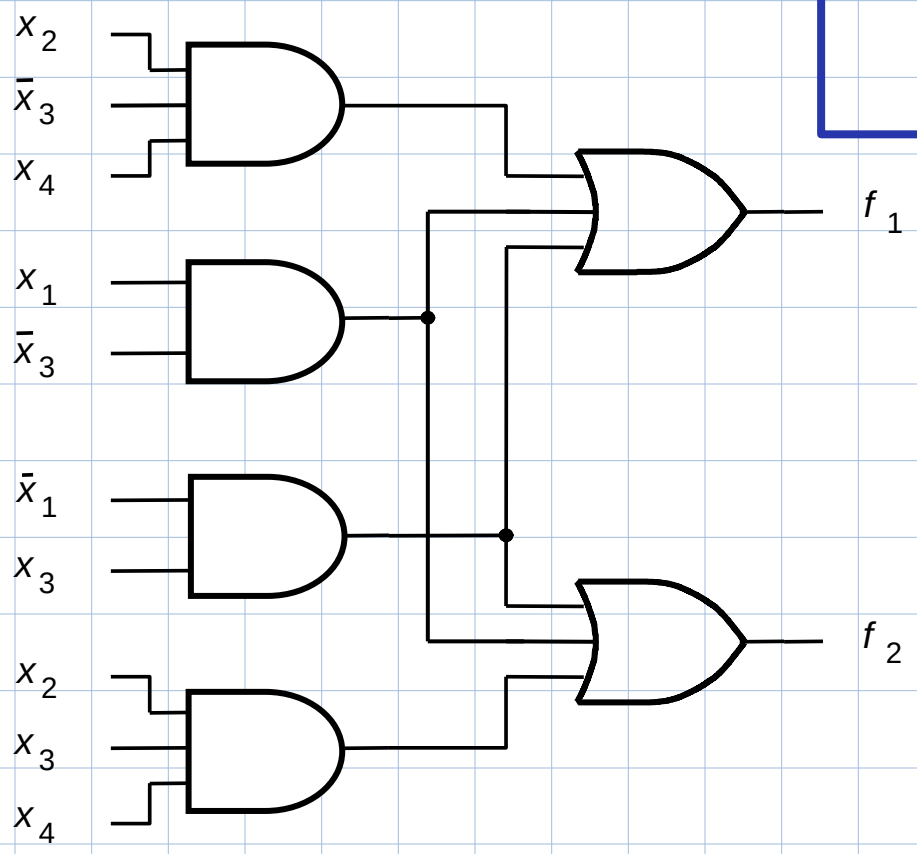


(b) Function f_2

- El costo de f1 implementada en forma independiente es 14
- El costo de f2 implementada en forma independiente también es 14

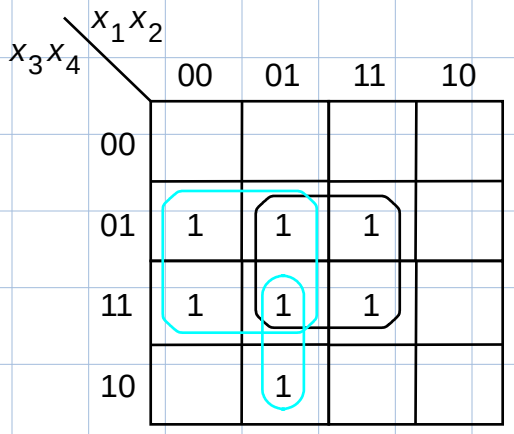
Ejemplo 4.1 – Múltiples salidas

El costo de la implementación combinada de f1 y f2 es 22, mucho menor que el caso de funciones independientes

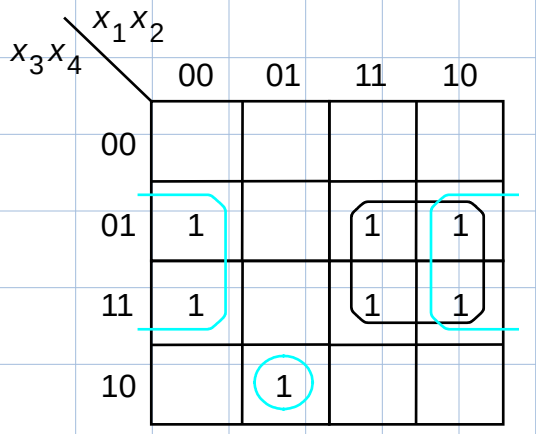


(c) Combined circuit for f_1 and f_2

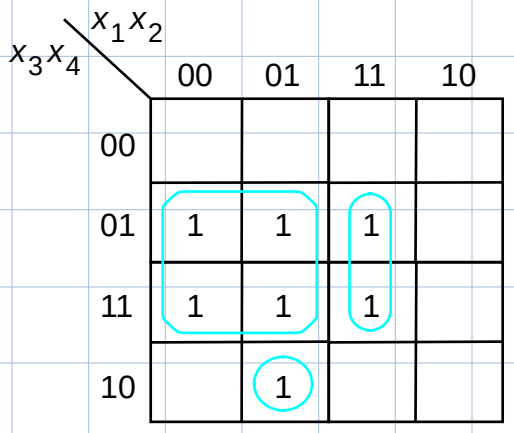
Ejemplo 4.2 – Múltiples salidas



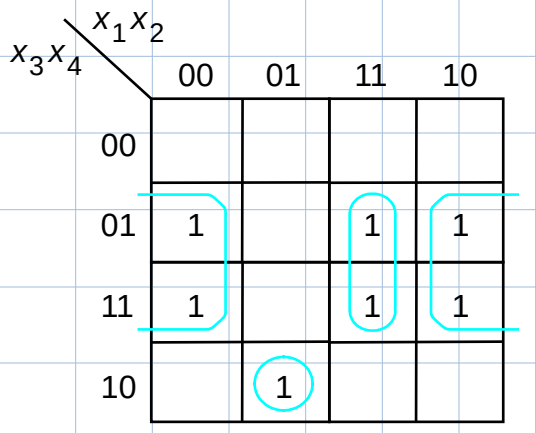
(a) Optimal realization of f_3



(b) Optimal realization of f_4

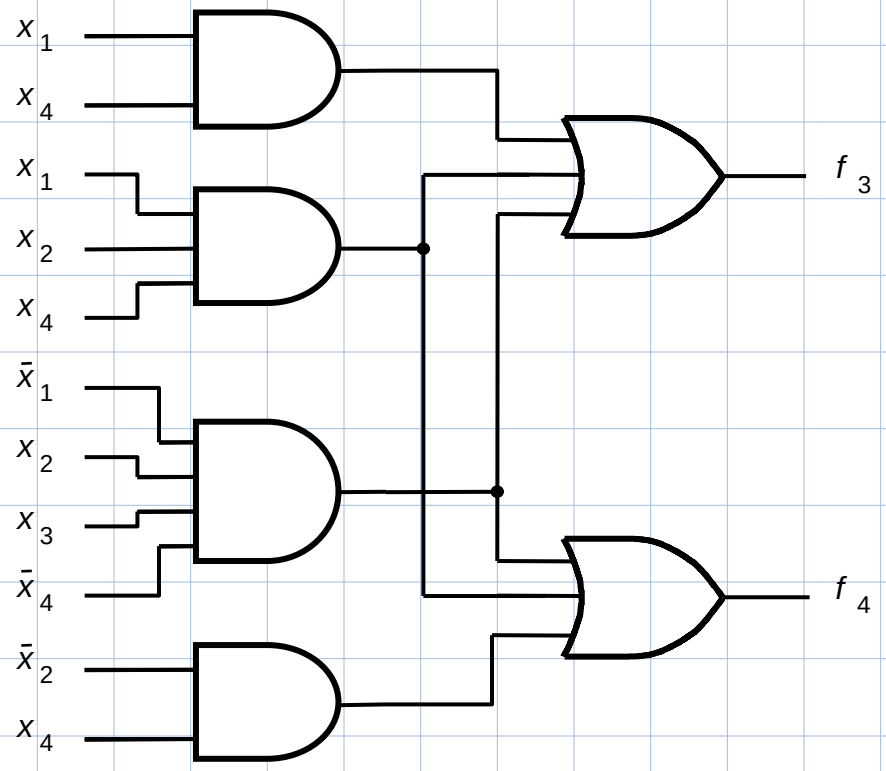


(c) Optimal realization of f_3 and f_4 together



- Podemos encontrar realizaciones óptimas para f3 y f4 individualmente, pero…
- La realización de f3 y f4, combinando compuertas, requiere implementaciones subóptimas de cada una

Ejemplo 4.2 – Múltiples salidas



(d) Combined circuit for f_3 and f_4