Tema 2. Funciones Lógicas

- Algebra de Conmutación.
- Representación de circuitos digitales.
- Minimización de funciones lógicas.

Minimización de Funciones Lógicas

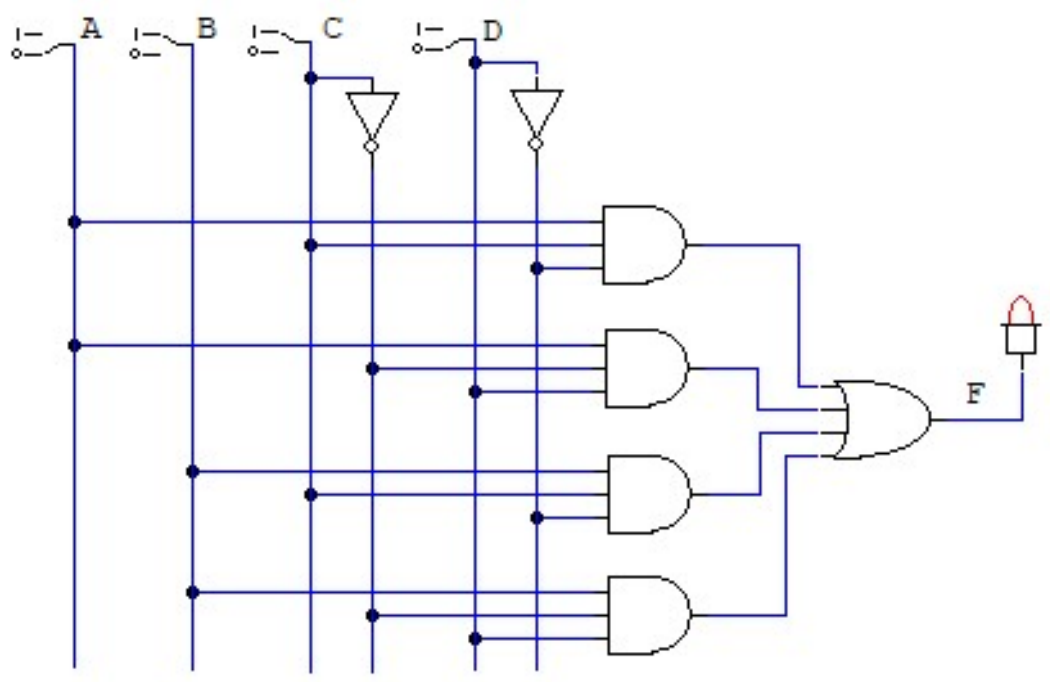
- Minimización en dos niveles. Mapas de Karnaugh de 3 y 4 variables. K-cubos. Definición de una función mínima en dos niveles. Mapas de Karnaugh de 5 y 6 variables.
- Implicantes primos. Implicantes primos esenciales. Minimización en dos niveles mediante el mapa de Karnaugh en problemas lógicos completa e incompletamente especificados. Minimización multifunción.
- Minimización algorítmica en dos niveles (una y varias salidas) y multinivel.

- Minimización en dos niveles de una función lógica. Encontrar una forma SOP (o POS) mínima.

  Extensión a problemas de varias funciones

  Objetivo básico: encontrar formas lógicas con el menor número de términos productos (o sumas) y el menor número de literales por término producto (o suma).

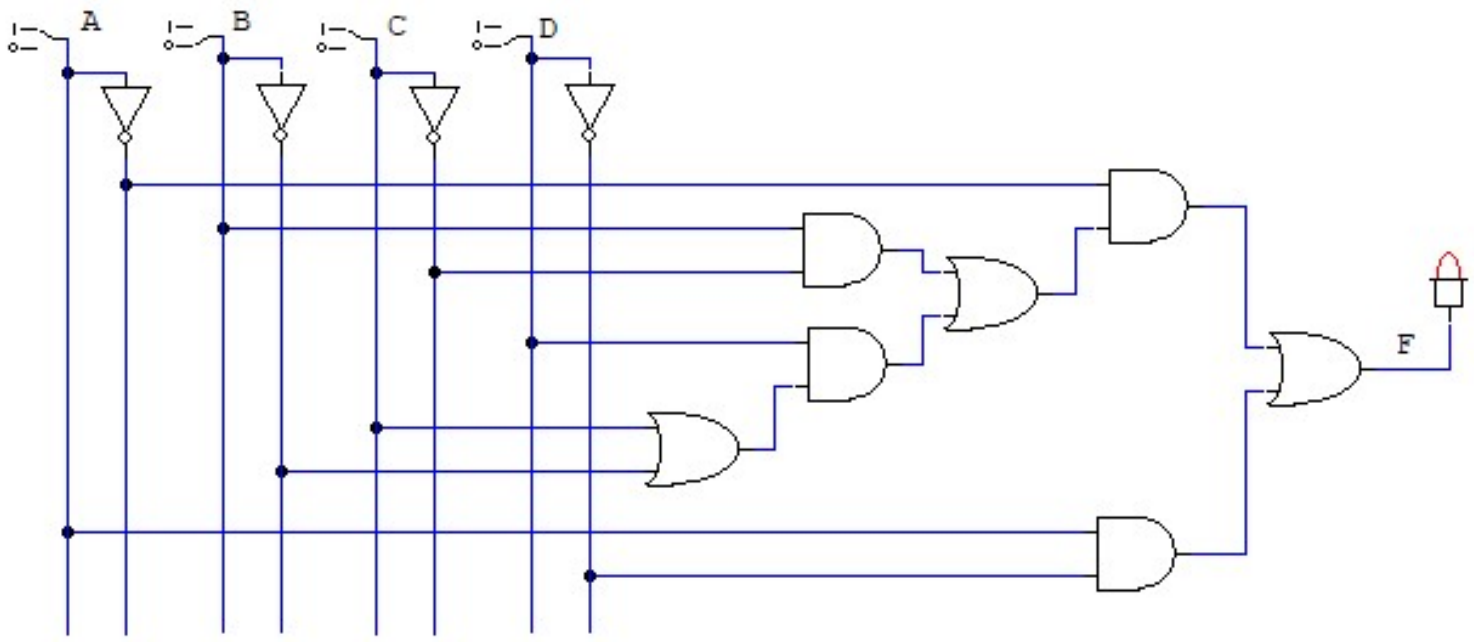
  $F(A, B, C, D) = A\,C\,\overline{D} + A\,\overline{C}\,D + B\,C\,\overline{D} + B\,\overline{C}\,D$



- Síntesis multinivel. Realizar una serie de operaciones sobre funciones lógicas que encuentren una buena forma factorizada (varios niveles AND/OR/AND/OR…).

  Objetivo básico: reducir el número de literales de la expresión lógica.

  $F(A, B, C, D) = \overline{A} [B \overline{C} + D (C + \overline{B})] + A \overline{D}$



Minimización en dos niveles

- El paso de funciones canónicas a funciones estándar mediante álgebra de conmutación no garantiza encontrar una solución mínima si no se usa un método algorítmico.

$$F(x,y,z) = x\,\overline{z} + x\,\overline{y}\,z + \overline{x}\,z + \overline{x}\,y\,\overline{z} = x\,\overline{z} + x\,\overline{y} + \overline{x}\,z + \overline{x}\,y$$

Sin embargo, una solución mínima es:

$$F(x,y,z) = y\,\overline{z} + x\,\overline{y} + \overline{x}\,z$$

La aplicación de los teoremas "a mano" no permite ver relaciones de simplificación "ocultas". A veces sería necesario expandir la función para luego simplificarla. Para ver bien las relaciones se usan métodos gráficos.

Mapa de Karnaugh

- El Mapa de Karnaugh es un método para observar una tabla de verdad de forma gráfica y observar la relaciones de adyacencia entre los 1s (o 0s) de la tabla.

Cada grupo de 1s (o 0s) de 1, 2, 4, 8, 16, …, casillas que formen un cuadrado o un rectángulo en el Mapa son un **cubo** (o implicante) de la función y corresponden a un término producto (o término suma).

Cada casilla está marcada en notación decimal.

A \ B	0	1
0	0	1
1	2	3

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

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

- Los valores en las entradas de los Mapas de Karnaugh se sitúan de forma que entre cada casilla adyacente del Mapa de Karnaugh (izquierda, derecha, arriba, abajo) haya distancia de Hamming 1.

  Hay que considerar que los bordes están unidos y que hay adyacencia entre las filas de abajo y arriba, y las columnas derecha e izquierda.

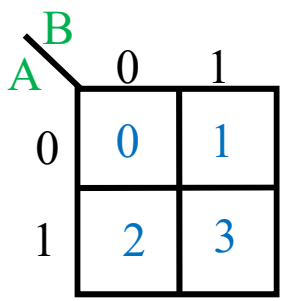
- Cada casilla es adyacente a tantas casillas como entradas haya en la función.

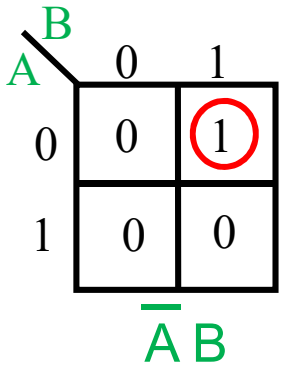
- Los cubos o agrupaciones de 1s o 0s de la función son de un orden determinado (k-cubos):

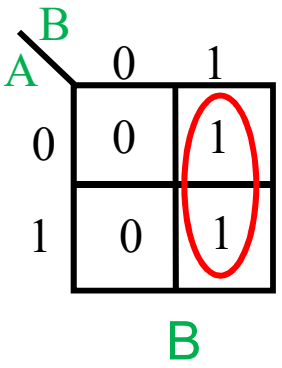
  1 casilla: 0-cubo; 2 casillas: 1-cubo; 4 casillas: 2-cubo; etc.

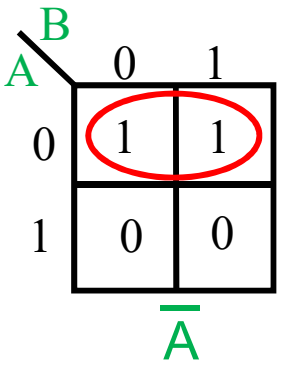
  El número de literales de un k-cubo de una función de N entradas es N-k.

Siguiendo la notación de las formas canónicas, al agrupar los 1s se forman términos productos:
Si X está siempre a 1 => literal X; si X está siempre a 0 => literal $\overline{X}$
Si X toma valores 0 y 1 => no se utiliza X

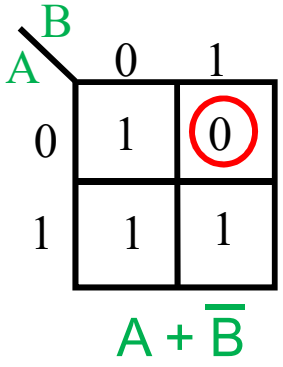


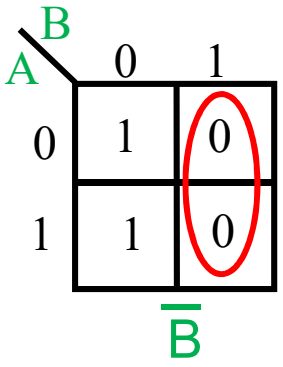


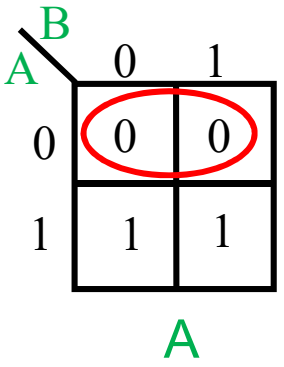




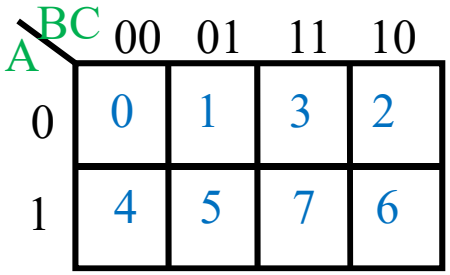
Siguiendo la notación de las formas canónicas, al agrupar los 0s se forman términos sumas:
Si X está siempre a 1 => literal $\overline{X}$; si X está siempre a 0 => literal X
Si X toma valores 0 y 1 => no se utiliza X

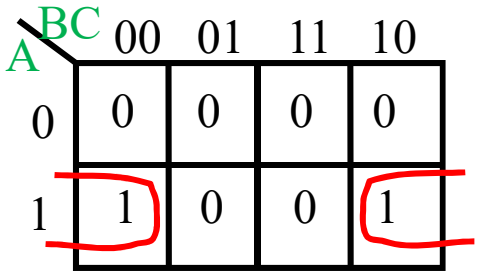




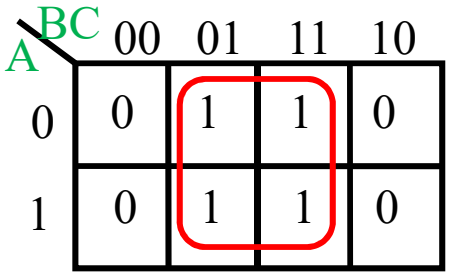


En mapas de 3 o 4 entradas los bordes están unidos y se pueden formar cubos entre las filas de abajo y arriba, y las columnas derecha e izquierda

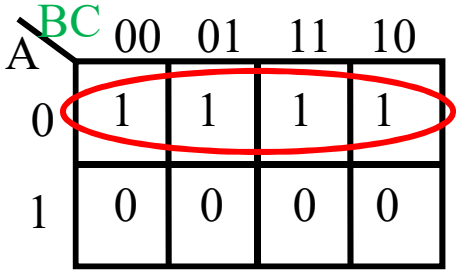




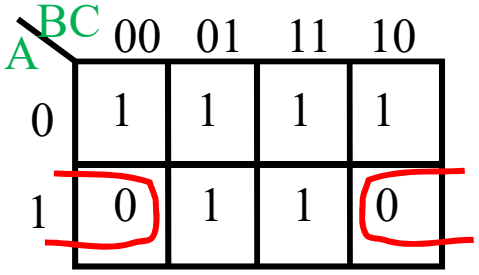
$A\,\overline{C}$



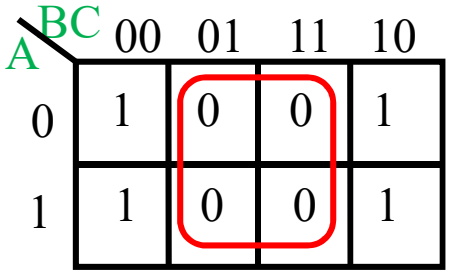
C



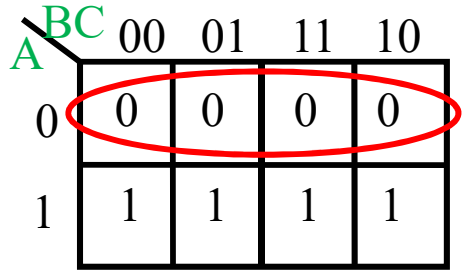
$\overline{A}$



$\overline{A} + C$

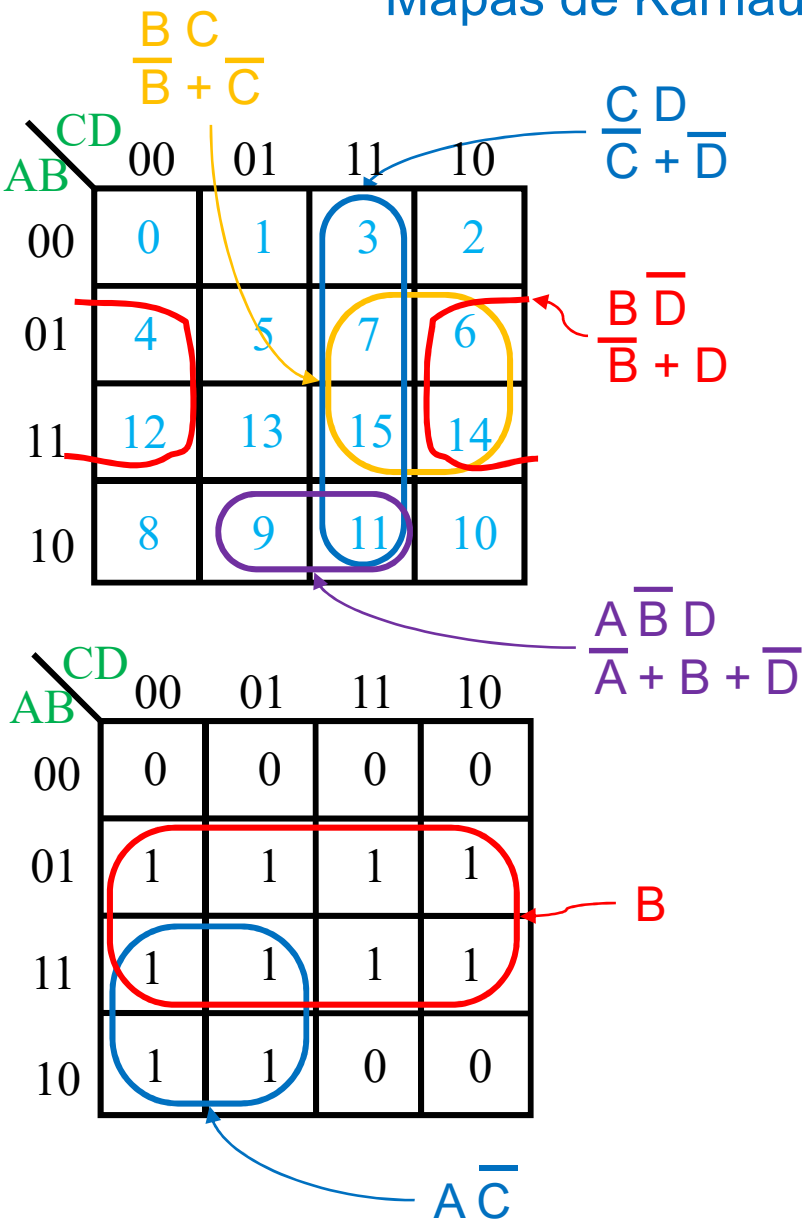


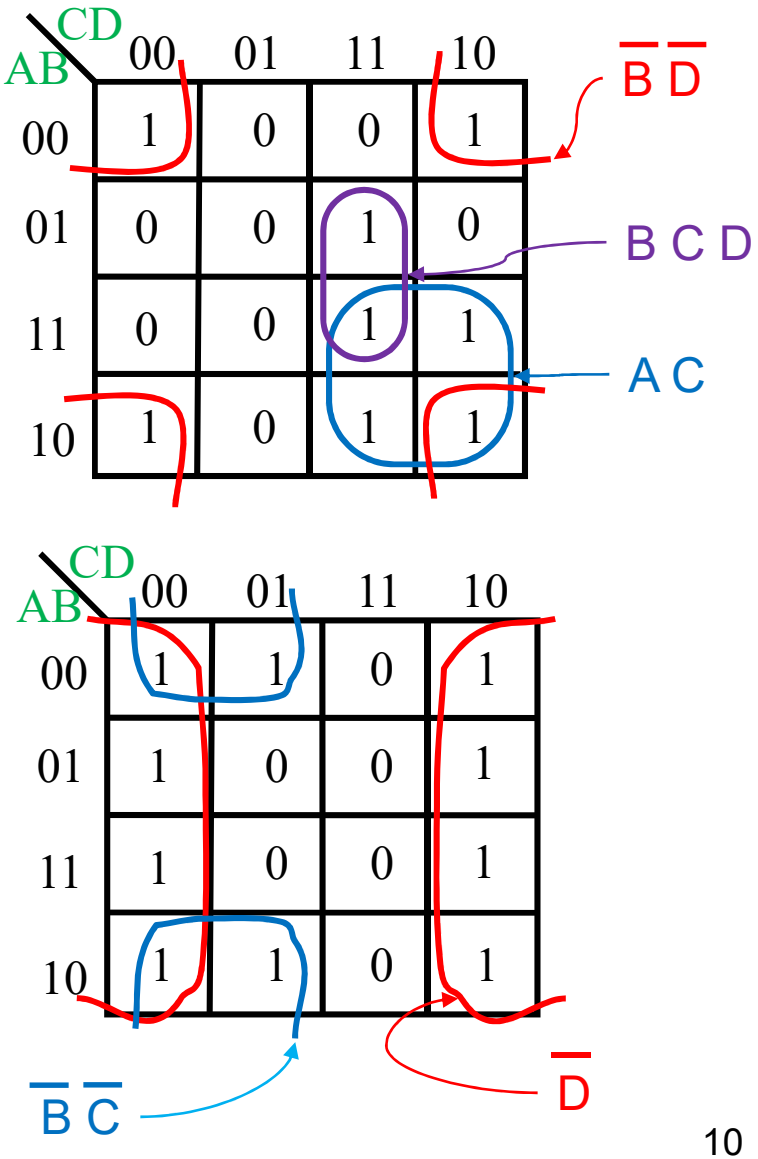
$\overline{C}$



A

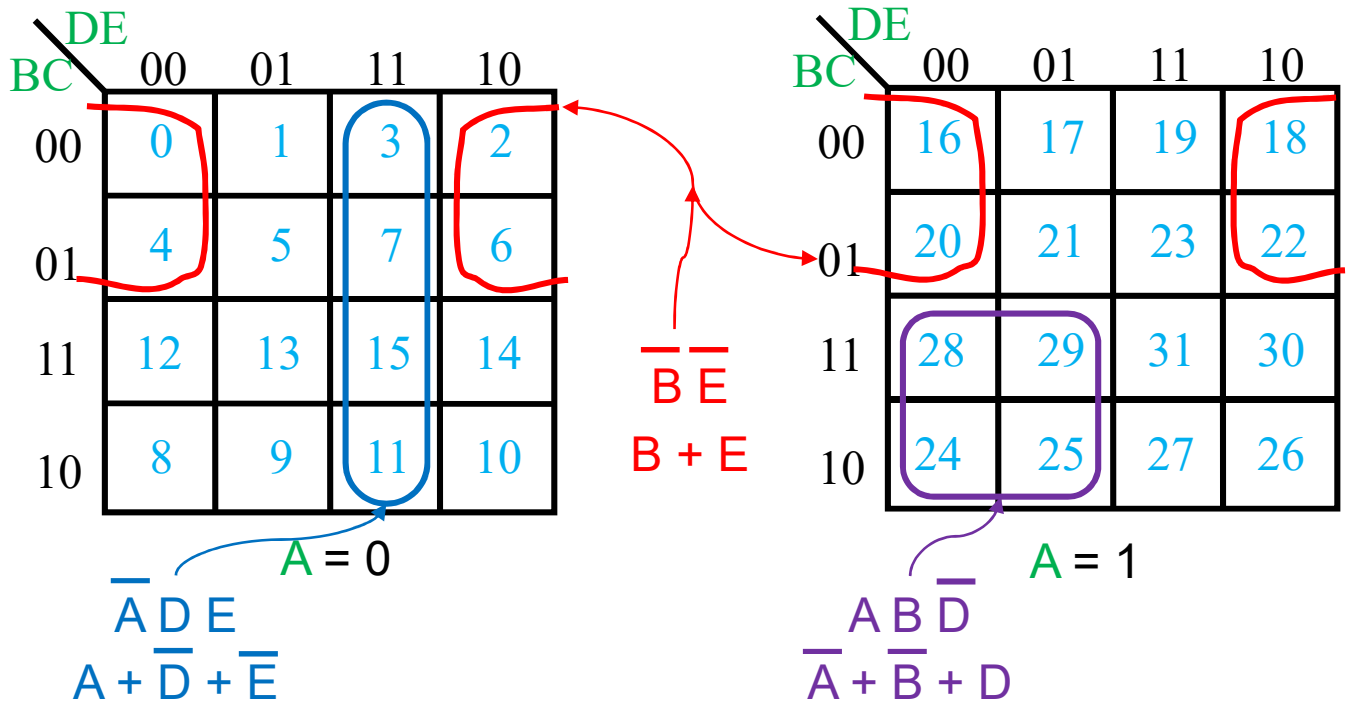
Mapas de Karnaugh de 4 variables F(A,B,C,D)





Mapas de Karnaugh de 5 variables F(A,B,C,D,E)

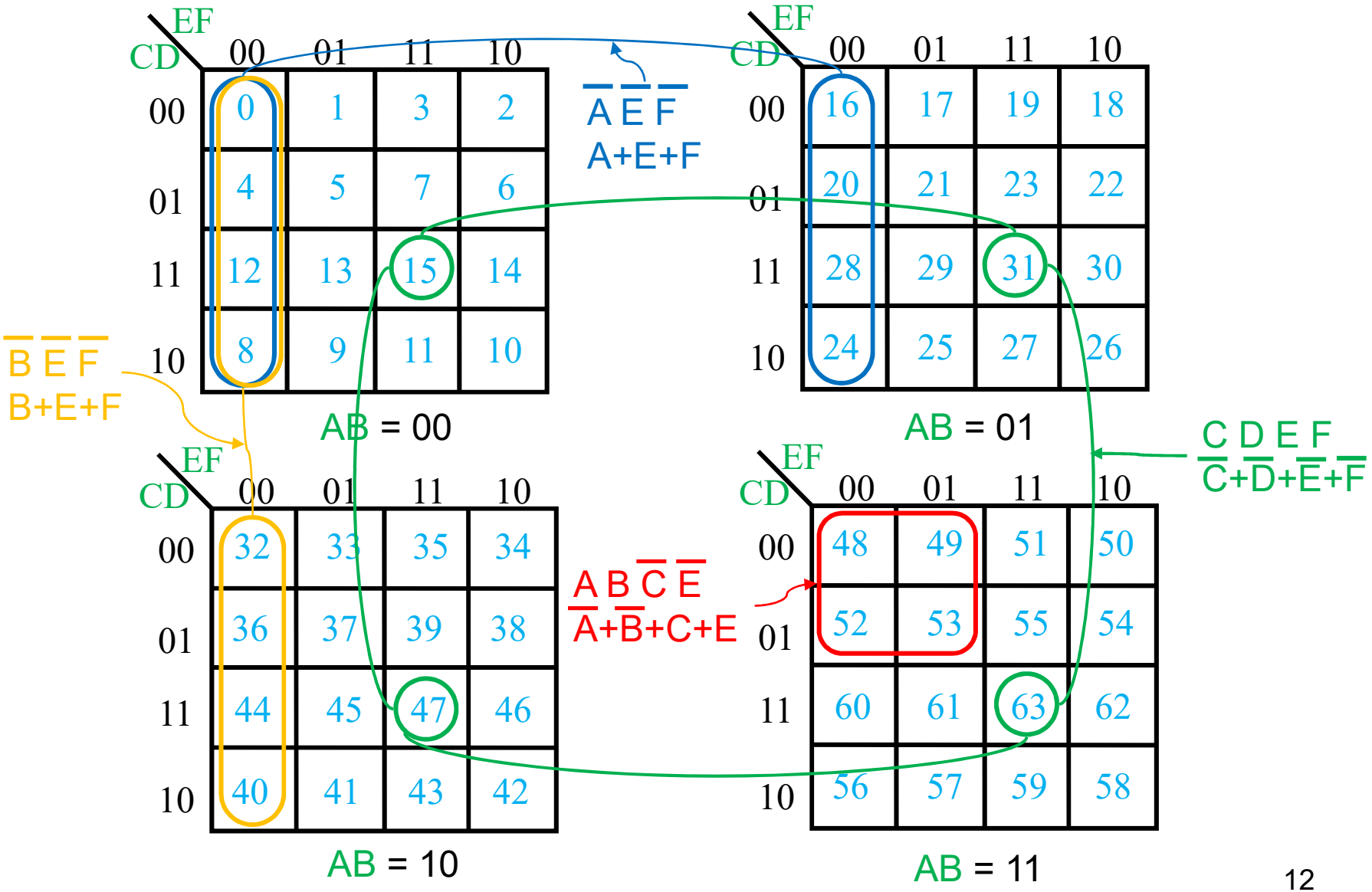
Se forma con 2 mapas de Karnaugh de 4 variables con una variable externa a valor 0 y 1 en cada mapa



Tres tipos de k-cubos (o implicantes):

- Solo en el mapa A = 0
- Solo en el mapa A = 1
- En los dos mapas. Mismas casillas de ambos mapas
  La expresión lógica no depende de A.

Mapas de Karnaugh de 6 variables F(A,B,C,D,E,F): 4 mapas de 4 variables controlados por dos variables externas



Minimización mediante el Mapa de Karnaugh

- Objetivo: generar una expresión mínima en dos niveles para una función lógica, que incluya a todos los minterms (maxterms) de la función.

- Una expresión de tipo SOP (POS) es mínima cuando:

1. No existe ninguna otra expresión equivalente que incluya menos términos productos (sumas) => utilizar el menor número de cubos en un mapa de Karnaugh.

2. No hay otra expresión equivalente que conste del mismo número de términos productos (sumas) y que tenga un menor número de literales => utilizar los cubos de mayor tamaño en un mapa de Karnaugh.

Minimización mediante el Mapa de Karnaugh

- Implicante primo de una función F es un cubo de F que no está incluido en ningún cubo de mayor orden. El uso de implicantes primos en lugar de cubos garantiza que se puede cumplir la condición 2 (menor número de literales).

- Un proceso de minimización en dos niveles incluye estos dos procesos:

  1. Determinación de todos los implicantes primos (no es absolutamente necesario encontrar todos).

  2. Selección del menor número de implicantes primos que cubre al menos una vez a todos los 1s (o 0s) de la función F. Si hay varias posibilidades tomar los que sean cubos de mayor orden.

Minimización mediante el Mapa de Karnaugh

- Implicante primo esencial: único implicante primo IP que cubre a un minterm (o maxterm) determinado de la función.

  Por ello, la expresión de F en forma SOP (POS) debe incluir obligatoriamente a IP como término producto (suma).

  1ª regla para minimización: Localizar las casillas cubiertas por un único implicante primo IP => IP es esencial, forma parte de la función mínima, y todas las casillas de IP quedan cubiertas, por lo que no hay que preocuparse más de ellas.

Minimización mediante el Mapa de Karnaugh

2ª regla para minimización: Cuando una casilla puede ser cubierta por varios implicantes primos, escoger un implicante primo de orden k máximo entre ellos, que cubra las mismas casillas que quedan sin cubrir (y si se puede alguna más), que todos los otros implicantes. Si hay dos implicantes del mismo tamaño que cumplan esta condición da igual escoger uno u otro.

3ª regla para minimización: Buscar la casilla cubierta por menos implicantes primos y probar con cada uno de ellos aplicando de nuevo las tres reglas. Quedarse con la solución más pequeña generada.

Minimización mediante el Mapa de Karnaugh

- Funciones incompletamente especificadas

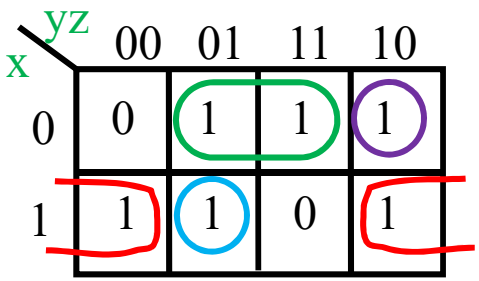
  Generación de implicantes primos: utilizar los "don't care" como si fuesen 1's en la forma SOP y como si fuesen 0's en la forma POS para encontrar implicantes primos.

  Selección de implicantes primos: cubrir todos los 1's (SOP) o los 0's (POS) de la función. No hace falta cubrir los "don't care".

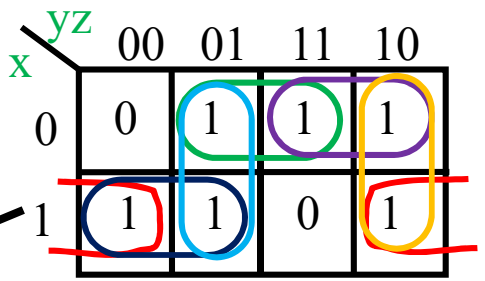
Minimización de

$F(x,y,z) = x\,\overline{z} + x\,\overline{y}\,z + \overline{x}\,z + \overline{x}\,y\,\overline{z}$

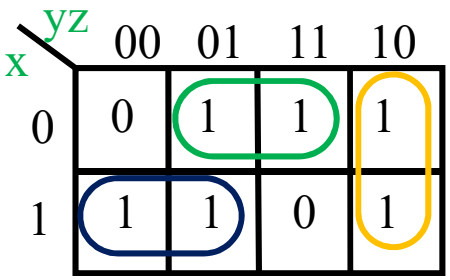
4 Cubos



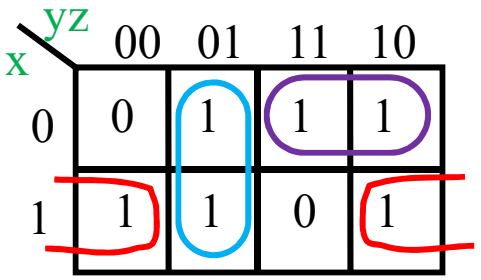
6 Implicantes primos



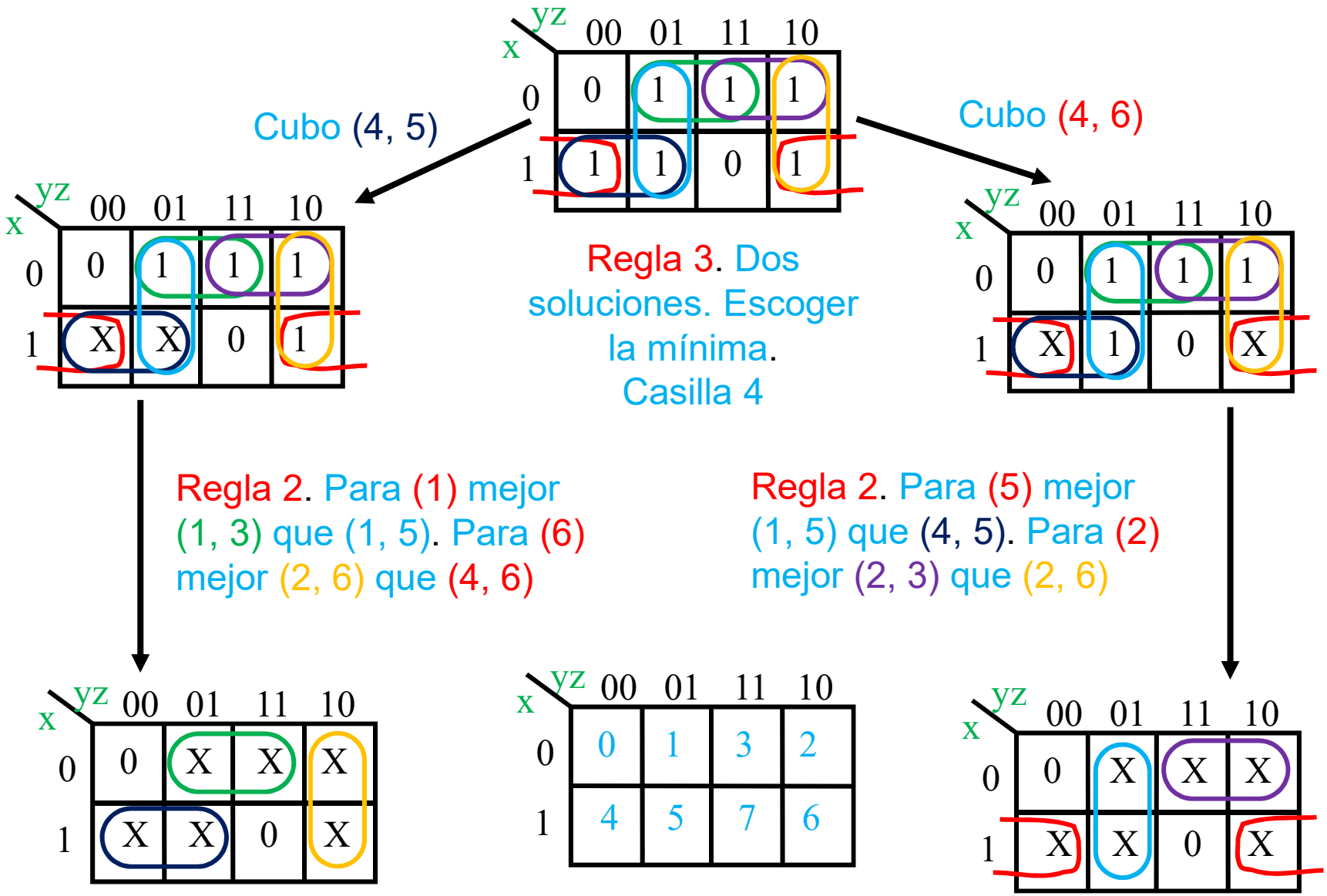
Regla 3. Dos soluciones mínimas



$F(x,y,z) = x\,\overline{y} + \overline{x}\,z + y\,\overline{z}$



$F(x,y,z) = x\,\overline{z} + \overline{y}\,z + \overline{x}\,y$


Cubo (4, 5)
Cubo (4, 6)
Regla 3. Dos soluciones. Escoger la mínima. Casilla 4
Regla 2. Para (1) mejor (1, 3) que (1, 5). Para (6) mejor (2, 6) que (4, 6)
Regla 2. Para (5) mejor (1, 5) que (4, 5). Para (2) mejor (2, 3) que (2, 6)
F(x,y,z) = x ȳ + x̄ z + y z̄
F(x,y,z) = x z̄ + ȳ z + x̄ y

Minimización mediante el Mapa de Karnaugh

- Resolución de un problema lógico:

A la vez si fuese posible

- Encontrar la tabla de verdad del problema lógico.
- Plantear el Mapa de Karnaugh.
- Encontrar la función lógica minimizada con el mapa de Karnaugh.
- Construir el circuito.

Minimización mediante el Mapa de Karnaugh

Encontrar la forma mínima SOP que define en la salida F si un número A de tres bits (A2, A1, A0) está dentro de unas especificaciones determinadas. Hay varias especificaciones, por lo que se dispone de dos entradas de control S1 y S0, que determinan la especificación que se está comprobando. De esta forma, si S1S0 son:

- 00: F es verdadera si A toma valor entre [0, 2], o [4], y falsa en cualquier otro caso.
- 01: F es verdadera si A toma valor entre [1, 3] o [6, 7], y falsa si toma el valor entre [4, 5].
- 10: F es verdadera si A es una potencia de 2, y falsa si es múltiplo de 3 (0 no es potencia ni múltiplo).
- 11: En este caso se sabe que A no puede ser potencia de 2, y que F es verdadera si el número es impar, y falsa en el resto de los casos.

Minimización mediante el Mapa de Karnaugh

Obtención de la tabla de verdad a partir de las especificaciones en función del valor de A para cada valor de S1S0:

- 00: F es 1 si A es 0, 1, 2 o 4
  F es 0 si A es 3, 5, 6 o 7 (los que no son 1)

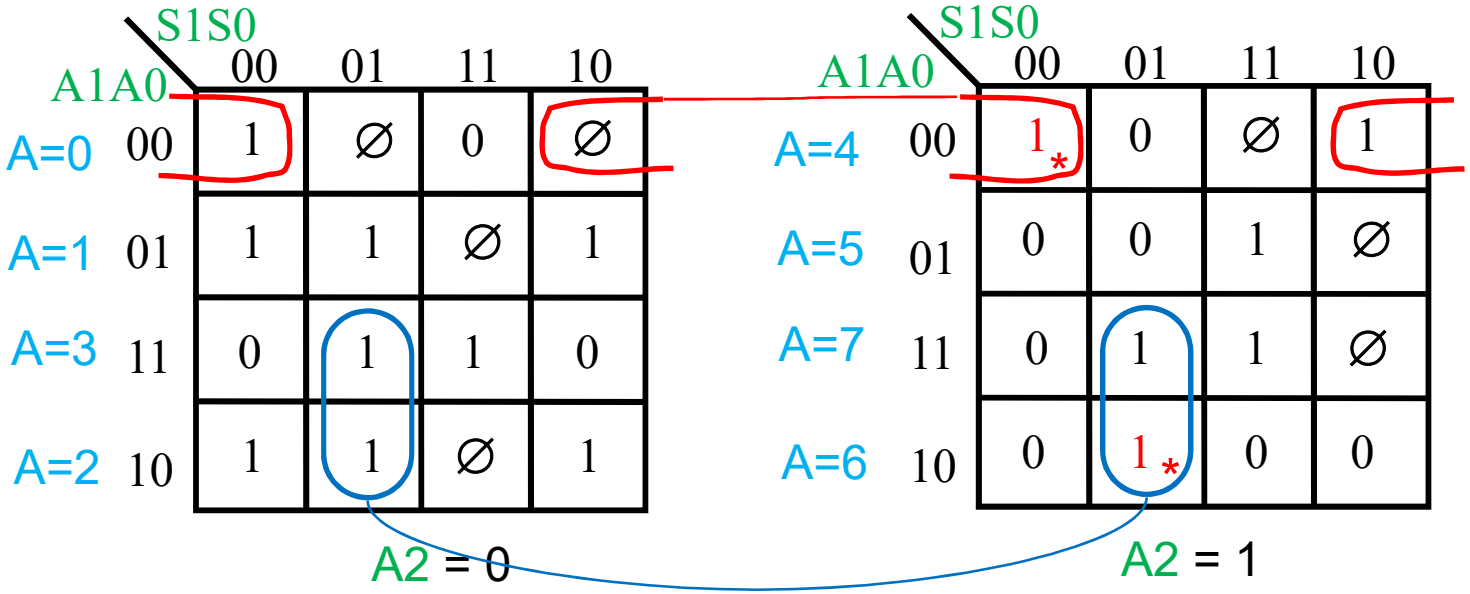
- 01: F es 1 si A es 1, 2, 3, 6 o 7 (los 1s)
  F es 0 si A es 4 o 5 (los 0s)
  F es ∅ si A es 0 (no definido)

- 10: F es 1 si A es 1, 2 o 4 (los 1s)
  F es 0 si A es 3 o 6 (los 0s)
  F es ∅ si A es 0, 5 o 7 (no definidos)

- 11: F es ∅ si A es 1, 2 o 4 (entradas no validas)
  F es 1 si A es 3, 5, 7 (impares, salvo 1 que ya es ∅)
  F es 0 si A es 0 o 6 (resto de valores)

Minimización mediante el Mapa de Karnaugh

Mapa de Karnaugh de 5 variables F(A2,A1,A0,S1,S0)

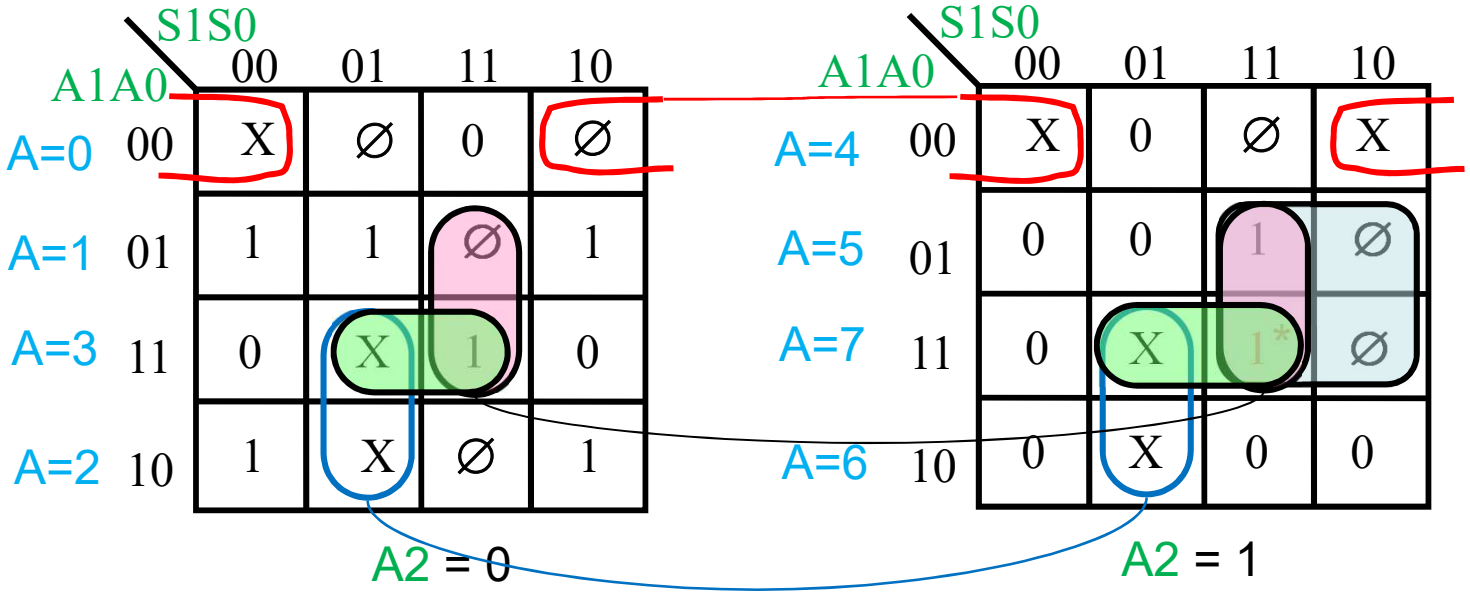


* Casillas que producen implicantes primos esenciales

$F = A1\ \overline{S1}\ S0 + \overline{A1}\ \overline{A0}\ \overline{S0} + \ldots..$

Minimización mediante el Mapa de Karnaugh

Mapa de Karnaugh de 5 variables F(A2,A1,A0,S1,S0)



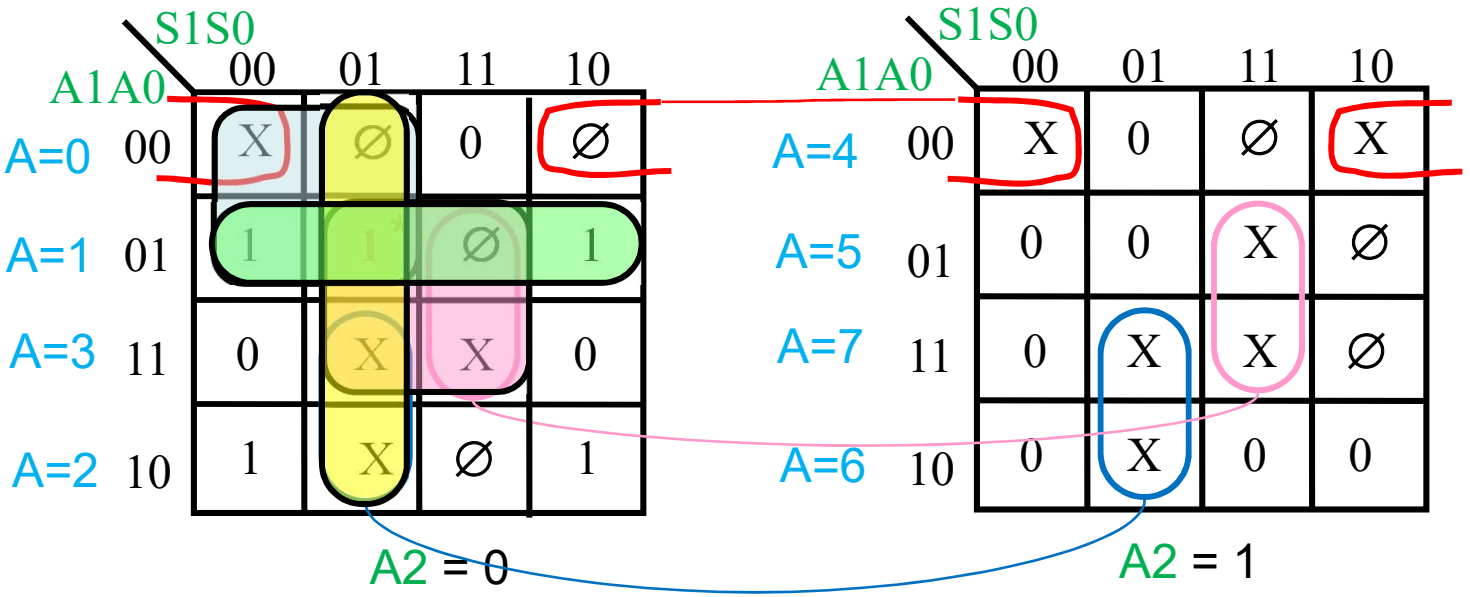
Regla 2. * Casillas que tienen un mejor implicante primo:

- Para la casilla 31 se usa (7, 15, 23, 31) que cubre también a 15 y a 23 (las otras opciones solo cubren a 15 o a 23)

$$F = A1\ \overline{S1}\ S0 + \overline{A1}\ \overline{A0}\ \overline{S0} + A0\ S1\ S0 + \ldots..$$

Minimización mediante el Mapa de Karnaugh

Mapa de Karnaugh de 5 variables F(A2,A1,A0,S1,S0)



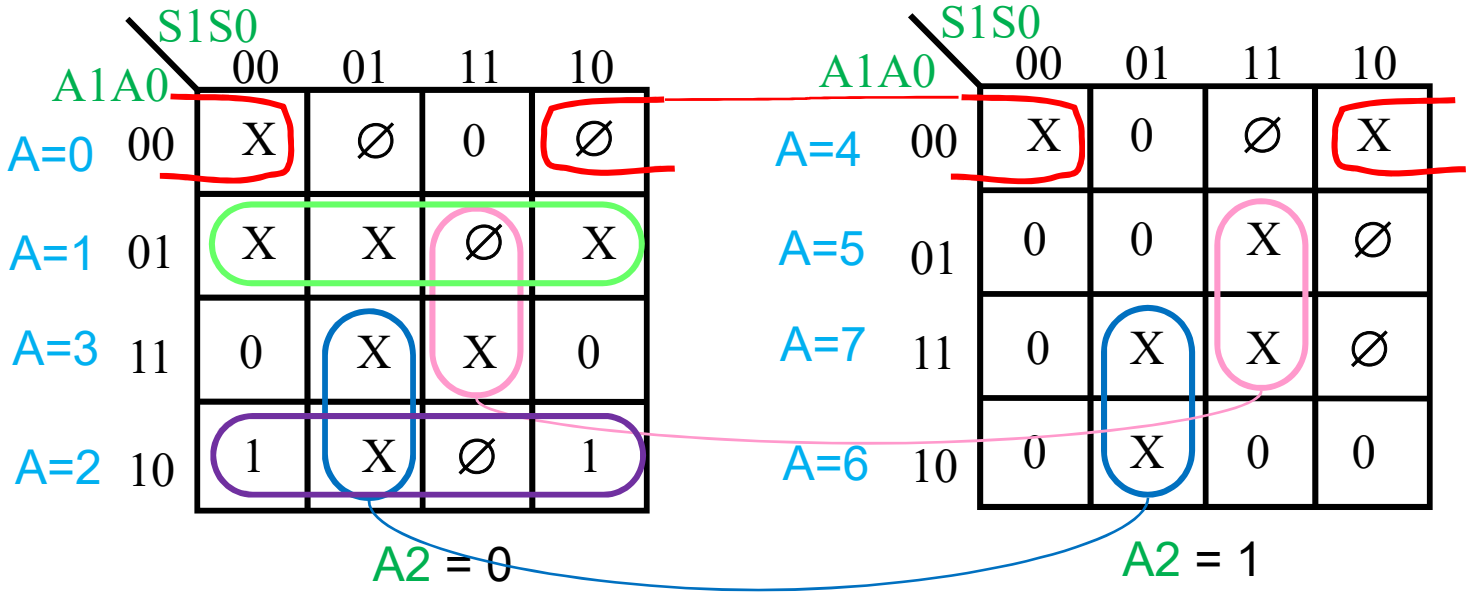
Regla 2. * Casillas que tienen un mejor implicante primo

- Para la casilla 5 se usa (4, 5, 6, 7) que cubre también a 4 y 6 (las otras opciones no cubren nada más o solo a 4)

$$F = A1\ \overline{S1}\ S0 + \overline{A1}\ \overline{A0}\ \overline{S0} + A0\ S1\ S0 + \overline{A2}\ \overline{A1}\ A0 + \ldots..$$

Minimización mediante el Mapa de Karnaugh

Mapa de Karnaugh de 5 variables F(A2,A1,A0,S1,S0)



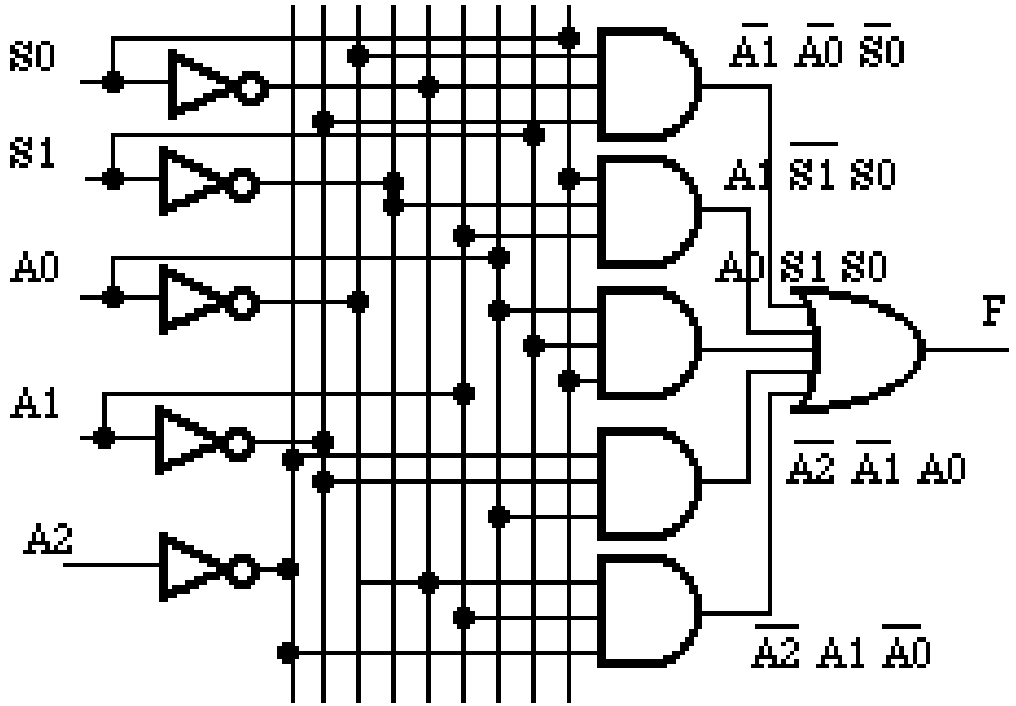
Se cubren los 1s restantes con el menor número posible de implicantes primos. Hay dos opciones de un solo implicante primo (fila inferior o 4 esquinas en A2 = 0). Ambas son del mismo tamaño, eligiendo cada una de ellas dan dos soluciones mínimas distintas

$$F = A1\,\overline{S1}\,S0 + \overline{A1}\,\overline{A0}\,\overline{S0} + A0\,S1\,S0 + \overline{A2}\,\overline{A1}\,A0 + \overline{A2}\,A1\,\overline{A0}$$

$$(+\ \overline{A2}\,\overline{A0}\,\overline{S0})$$

Minimización mediante el Mapa de Karnaugh

$$F = A1\,\overline{S1}\,S0 + \overline{A1}\,\overline{A0}\,\overline{S0} + A0\,S1\,S0 + \overline{A2}\,\overline{A1}\,A0 + \overline{A2}\,A1\,\overline{A0}$$



Minimización mediante el Mapa de Karnaugh

Minimización de varias funciones lógicas

F1(A, B, C) = ∑ (2, 3, 6)
F2(A, B, C) = ∑ (3, 5, 7)

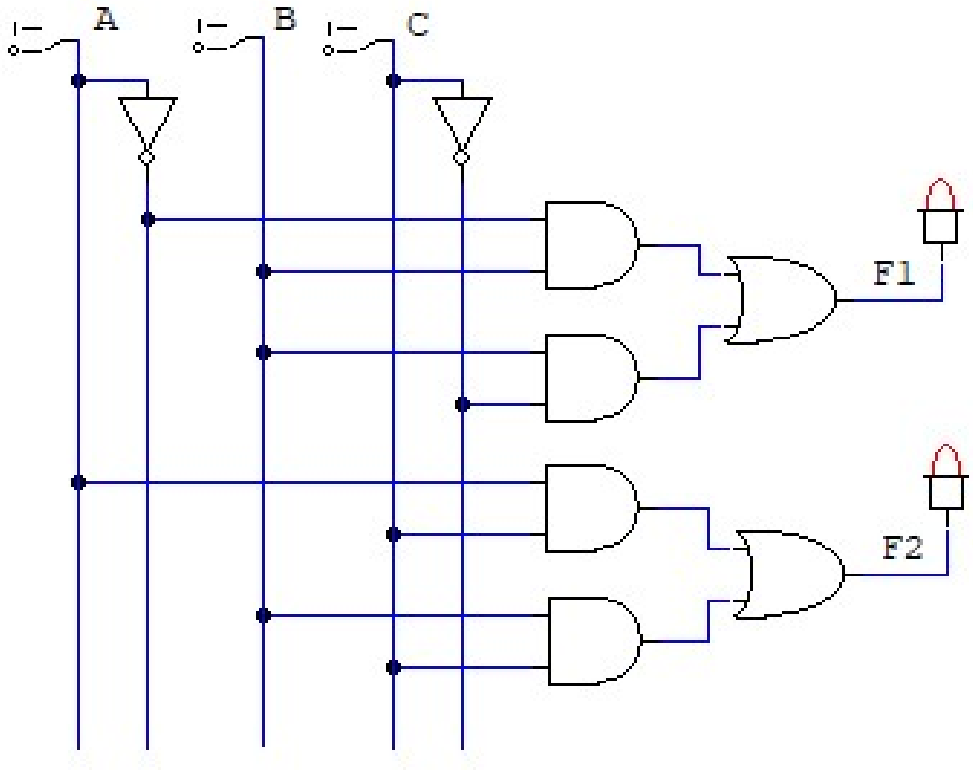
Minimización por separado

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

$F1 = \overline{A}\,B + B\,\overline{C}$

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

F2 = A C + B C



Minimización mediante el Mapa de Karnaugh

Minimización de varias funciones lógicas

F1(A, B, C) = ∑ (2, 3, 6)
F2(A, B, C) = ∑ (3, 5, 7)

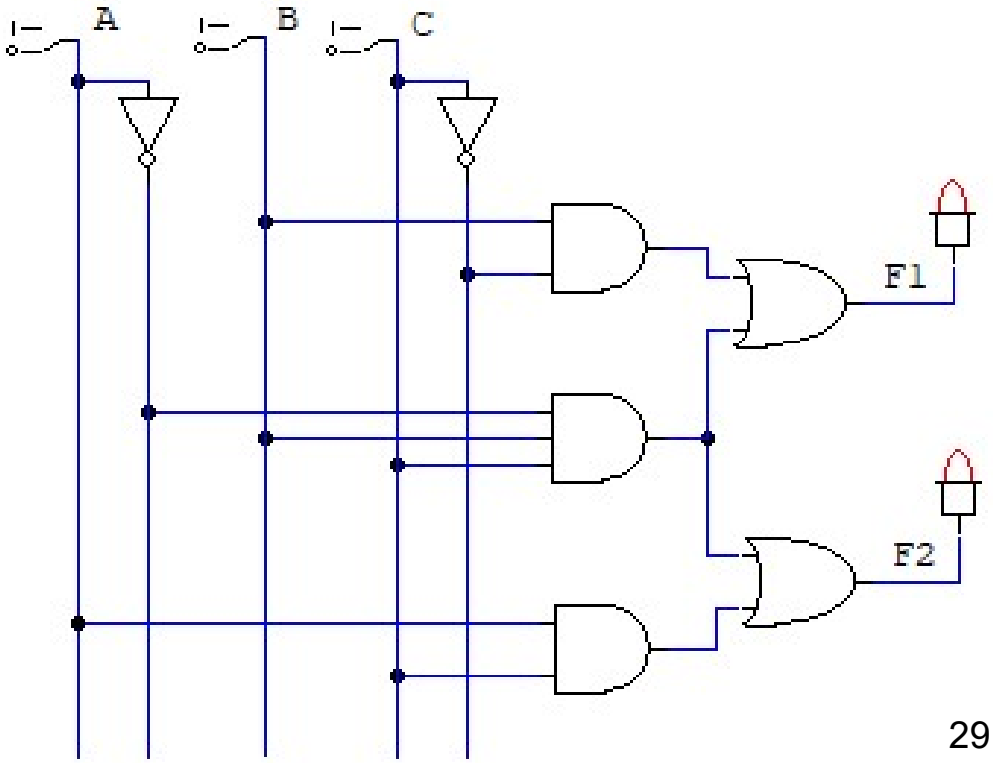
Minimización conjunta. El circuito se reduce. Hay que reducir el número de implicantes de todas las funciones a la vez.

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

$F1 = \overline{A}\,B\,C + B\,\overline{C}$

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

$F2 = \overline{A}\,B\,C + A\,C$

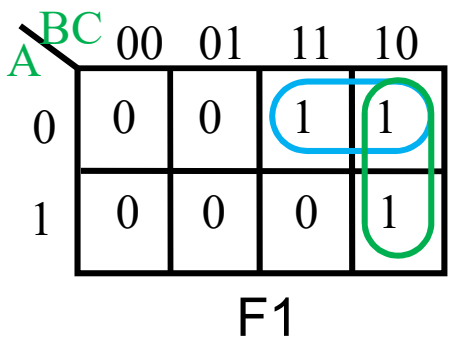


Minimización mediante el Mapa de Karnaugh

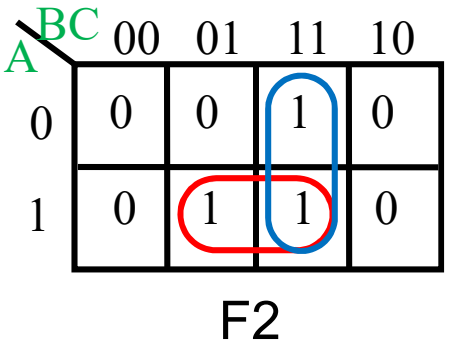
Minimización de varias funciones lógicas

F1(A, B, C) = ∑ (2, 3, 6)
F2(A, B, C) = ∑ (3, 5, 7)

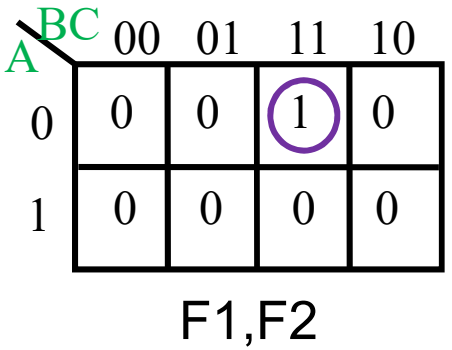
Hay que encontrar los implicantes primos de las intersecciones entre todas las combinaciones de funciones.



F1

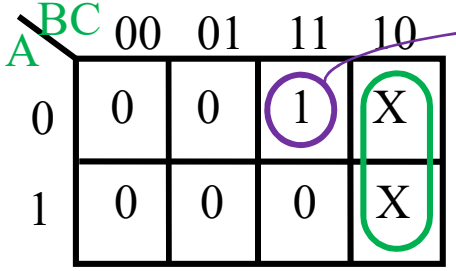


F2

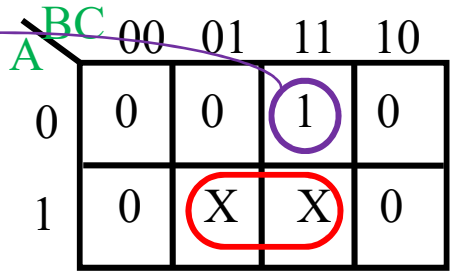


F1,F2

Se obtienen los implicantes primos esenciales:
6 F1 => (2, 6) F1; 5 F2 => (5, 7) F2



F1



F2

Se cubre el resto con el menor número de implicantes primos: 3 F1, 3 F2 => 3 F1,F2

Métodos algorítmicos para síntesis lógica

- Cuando se plantean problemas lógicos muy complejos no solo la minimización sino casi la resolución a mano se hace inviable. Se requieren herramientas de ayuda al diseño (CAD) que generen automáticamente los resultados de síntesis de las funciones lógicas.

- Existen algoritmos aplicados a la síntesis de funciones en dos niveles de una y varias salidas, y a la síntesis multinivel, la mayoría de ellos demasiado complejos para ser explicados en un tiempo razonable.

Método Quine-McCluskey

- Algoritmo de minimización en dos niveles. Las variantes de este algoritmo tiene un límite de entradas y/o salidas para encontrar la solución en un tiempo de cómputo razonable.

- Se divide en dos partes: generación de implicantes primos y selección de implicantes primos. La descripción inicial corresponde a una función de N entradas y una salida aunque puede adaptarse para varias salidas.

- Generación de implicantes primos: parte de los 0-cubos (1s en SOP o 0s en POS), y '–' o "don't cares" representados en binario y ordenados por el número i de 1s en su codificación binaria G[0][i]. Dentro del procedimiento se van generando k-cubos que se guardan en los G[k][i]. Los k-cubos que no generan ningún (k+1)-cubo son implicantes primos.

Método Quine-McCluskey

- Los k-cubos se representan en una notación en la que cada entrada puede tomar 3 valores para términos producto (o términos suma):

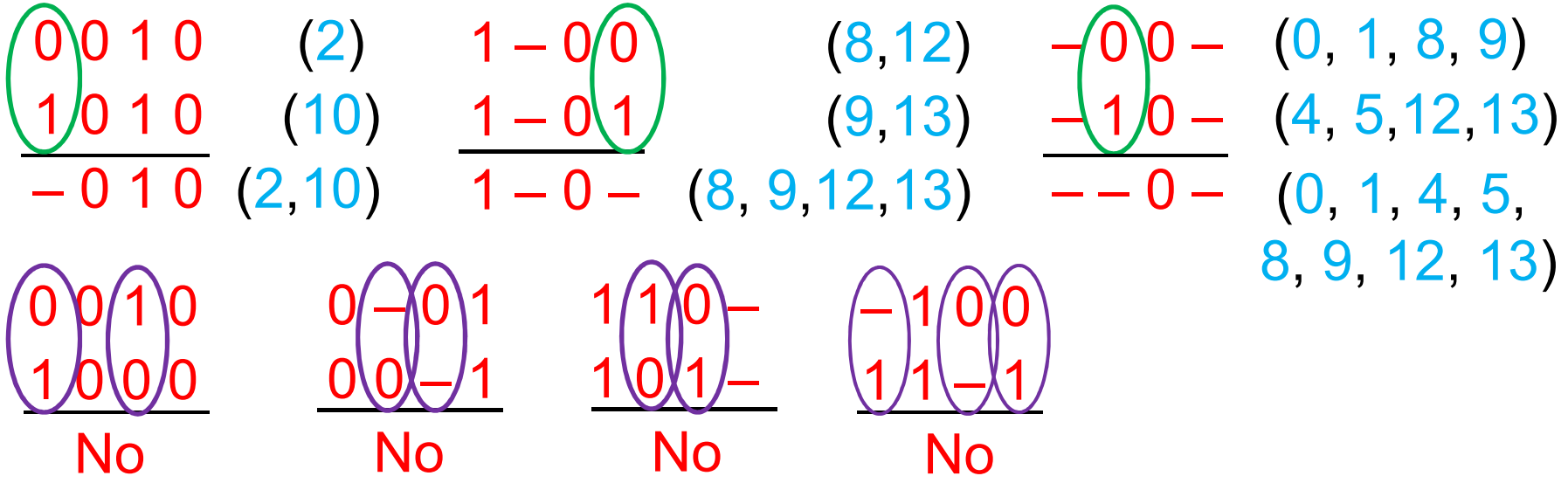
  0 => entrada complementada (sin complementar)
  1 => entrada sin complementar (complementada)
  – => no depende de la entrada

- Ejemplos de términos producto de 4 entradas A B C D:

  A B C D
  0 0 1 0 => $\overline{A}\,\overline{B}\,C\,\overline{D}$, 0-cubo (2)
  1 – 0 1 => $A\,\overline{C}\,D$, 1-cubo (9, 13)
  0 1 – 0 => $\overline{A}\,B\,\overline{D}$, 1-cubo (4, 6)
  – 0 1 – => $\overline{B}\,C$, 2-cubo (2, 3, 10, 11)
  – – 0 – => $\overline{C}$, 3-cubo (0, 1, 4, 5, 8, 9, 12, 13)

Generación de implicantes primos

- Dos k-cubos pueden formar un (k+1)-cubo si sólo difieren en una entrada y los valores distintos son 0 y 1. En el (k+1)-cubo la posición distinta toma el valor '–'. Ejemplos:



- Solo los k-cubos que tienen una diferencia de 1 en el números de entradas a 1 en la codificación binaria podrían formar un (k+1)-cubo.

- Existen otros métodos capaces de generar cubos a partir de cubos de distinto k (operación "star", por ejemplo).

Generación de implicantes primos

1. Inicio. k = 0. Agrupar los 0-cubos en grupos G[0][i], siendo i el número de 1s en la codificación binaria del minterm. i = 0.

2. Comparar cada k-cubo A del grupo G[k][i] con todos los k-cubos B del grupo G[k][i+1], generando los (k+1)-cubos C del grupo G[k+1][i]. Para k mayor que 0, cada C puede generarse varias veces pero no debe aparecer repetido en G[k+1][i]. Los k-cubos A y B se marcan (X), para indicar que no son implicantes primos, al estar incluidos en el (k+1)-cubo C.

3. Si i ≠ N-k-1, incrementar i y volver a 2;
   Si i = N-k-1, incrementar k; ahora:
   si k < N y hay k-cubos en G[k], hacer i = 0 y volver a 2;
   si no ir a 4.

4. Los cubos no marcados son implicantes primos. Fin del algoritmo.

$$F(A, B, C, D) = \sum(1, 2, 3, 5, 11, 12, 15) + \sum_{\varnothing}(6, 10, 13)$$

	Nº 1's	0-cubos	A	B	C	D	
G[0][1]	1	1	0	0	0	1	X
		2	0	0	1	0	X
G[0][2]	2	3	0	0	1	1	X
		5	0	1	0	1	X
		6	0	1	1	0	X
		10	1	0	1	0	X
		12	1	1	0	0	X
G[0][3]	3	11	1	0	1	1	X
		13	1	1	0	1	X
G[0][4]	4	15	1	1	1	1	X

	Nº 1's	1-cubos	A	B	C	D		
G[1][1]	1	(1, 3)	0	0	-	1		$\bar{A}\bar{B}D$ (a)
		(1, 5)	0	-	0	1		$\bar{A}\bar{C}D$ (b)
		(2, 3)	0	0	1	-	X	
		(2, 6)	0	-	1	0		$\bar{A}C\bar{D}$ (c)
		(2, 10)	-	0	1	0	X	
G[1][2]	2	(3, 11)	-	0	1	1	X	
		(5, 13)	-	1	0	1		$B\bar{C}D$ (d)
		(10, 11)	1	0	1	-	X	
		(12, 13)	1	1	0	-		$AB\bar{C}$ (e)
G[1][3]	3	(11, 15)	1	-	1	1		ACD (f)
		(13, 15)	1	1	-	1		ABD (g)

	Nº 1's	2-cubos	A	B	C	D	
G[2][1]	1	(2, 3, 10, 11)	-	0	1	-	$\bar{B}C$ (h)

Selección de implicantes primos

1. Inicio. Formar una tabla de implicantes primos (filas) frente a 0-cubos (columnas): 1s (SOP) o 0s (POS). No usar los "don't cares". Marcar los 0-cubos cubiertos por cada implicante primo.
2. Buscar columnas con una única marca. Incluir el implicante primo correspondiente en la función y eliminar de la tabla todas las columnas cubiertas por el implicante.
3. Si una fila A incluye a, o es igual que, una fila B, y el coste de A es menor o igual que el coste de B, eliminar la fila B.
4. Si una columna A incluye o es igual que una columna B, eliminar la columna A.
5. Si la tabla está vacía ir a 7; si quedan columnas: si los pasos 3 o 4 han modificado la tabla, volver a 2 y si no ir a 6.
6. La tabla es cíclica. Seleccionar un implicante primo IP y generar dos soluciones:
   - F1: incluir IP en la función, eliminando las columnas cubiertas por IP de la tabla.
   - F2: no incluir IP en la función, eliminar la fila IP.
7. Si se han probado todas las soluciones (F1, F2, …) ir al paso 8. Si no, tomar una de las soluciones que quedan por probar y volver a 2.
8. Seleccionar la función de menor coste (si hubiese varias) y generar la función lógica: suma (producto) de los términos producto (suma) seleccionados en el paso 2. Finalizar.

Selección de implicantes primos

- Criterios de coste. El coste de una función es la suma del coste de cada implicante seleccionado para la función. El objetivo de la minimización sería encontrar la función de menor coste.

  Ejemplos de criterios de coste:

  a) Menor número de implicantes. Todos los implicantes tienen coste 1. Típico de circuitos programables (PLA, PAL).

  b) Menor número de implicantes y menor número de literales. Los implicantes tienen como coste el número de literales del término lógico equivalente. Este criterio de coste corresponde a la minimización de una función lógica (primero el menor número de implicantes, y segundo el menor número de literales).

  c) Menor número de puertas lógicas (implicantes). Similar a a), pero con la diferencia de que un implicante de un único literal tiene coste 0.

  d) Menor número de puertas lógicas (implicantes) y de líneas (literales). Similar a b), pero con la diferencia de que un implicante de un único literal tiene coste 0 en puertas y 1 en líneas.

Selección de implicantes primos

$F(A, B, C, D) = \sum(1, 2, 3, 5, 11, 12, 15) + \sum_{\varnothing}(6, 10, 13)$

		1	2	3	5	11	12	15	C
$\bar{A}\bar{B}D$ (1,3)	(a)	X		X					3
$\bar{A}\bar{C}D$ (1,5)	(b)	X			X				3
$\bar{A}C\bar{D}$ (2,6)	(c)		X						3
$B\bar{C}D$ (5,13)	(d)				X				3
$AB\bar{C}$ (12,13)	(e)						X		3
ACD (11,15)	(f)					X		X	3
ABD (13,15)	(g)							X	3
$\bar{B}C$ (2,6,10,14)	(h)		X	X		X			2

1. Tabla Implicantes (filas) frente a 0-cubos (columnas)
   Uso el criterio de coste: b). Número de literales de cada implicante.
   Paso a 2.

Selección de implicantes primos

F(A, B, C, D) = ∑(1, 2, 3, 5, 11, 12, 15) + ∑ ∅ (6, 10, 13)

		1	2	3	5	11	12	15	C
$\bar{A}\bar{B}D$ (1,3)	(a)	X		X					3
$\bar{A}\bar{C}D$ (1,5)	(b)	X			X				3
$\bar{A}C\bar{D}$ (2,6)	(c)		X						3
$B\bar{C}D$ (5,13)	(d)				X				3
$AB\bar{C}$ (12,13)	(e)						X		3
ACD (11,15)	(f)					X		X	3
ABD (13,15)	(g)							X	3
$\bar{B}C$ (2,6,10,14)	(h)		X	X		X			2

2. Columnas con una marca.

(12) => F = **(e)** + ….

Se cubre la columna 12 de la tabla. No hay más columnas cubiertas, ya que (e) no cubre más columnas.

Paso a 3.

Selección de implicantes primos

$F(A, B, C, D) = \sum(1, 2, 3, 5, 11, 12, 15) + \sum_{\varnothing}(6, 10, 13)$

		1	2	3	5	11	15	C
$\bar{A}\bar{B}D$ (1,3)	(a)	X		X				3
$\bar{A}\bar{C}D$ (1,5)	(b)	X			X			3
$\bar{A}C\bar{D}$ (2,6)	(c)		X					3
$B\bar{C}D$ (5,13)	(d)				X			3
ACD (11,15)	(f)					X	X	3
ABD (13,15)	(g)						X	3
$\bar{B}C$ (2,6,10,14)	(h)		X	X		X		2

3. Eliminación de filas:
   (h) incluye a (c); (Coste(h) = 2) ≤ (Coste(c) = 3) => Eliminar (c).
   (b) incluye a (d); (Coste(b) = 3) ≤ (Coste(d) = 3) => Eliminar (d).
   (f) incluye a (g); (Coste(f) = 3) ≤ (Coste(g) = 3) => Eliminar (g).
   Paso a 4.

Selección de implicantes primos

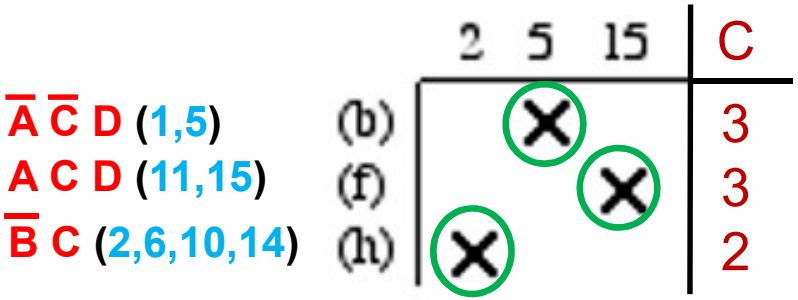
$F(A, B, C, D) = \sum(1, 2, 3, 5, 11, 12, 15) + \sum_{\varnothing}(6, 10, 13)$

		1	2	3	5	11	15	C
$\overline{A}\,\overline{B}\,D$ (1,3)	(a)	X		X				3
$\overline{A}\,\overline{C}\,D$ (1,5)	(b)	X			X			3
$A\,C\,D$ (11,15)	(f)					X	X	3
$\overline{B}\,C$ (2,6,10,14)	(h)		X	X		X		2

4. Eliminación de columnas:
   (3) incluye a (2) => Eliminar (3).
   (1) incluye a (5) => Eliminar (1).
   (11) incluye a (15) => Eliminar (11).
   La fila (b) queda vacía y se elimina. Paso a 5.

Selección de implicantes primos

$F(A, B, C, D) = \sum(1, 2, 3, 5, 11, 12, 15) + \sum_{\varnothing}(6, 10, 13)$



5. Tabla no vacía y modificada. Paso a 2.

2. Columnas con una marca.
   (2) => F = (e) + **(h)** + ….
   (5) => F = **(b)** + (e) + (h) + ….
   (15) => F = (b) + (e) + **(f)** + (h) + ….
   Se cubren las columna 2, 5 y 12 de la tabla. La tabla queda vacía.
   Paso a 3, 4, 5, 7 y 8.

8. Solo hay una función F = (b) + (e) + (f) + (h). La escojo.

   $F(A, B, C, D) = A\,B\,\overline{C} + \overline{A}\,\overline{C}\,D + A\,C\,D + \overline{B}\,C$

Selección de implicantes primos

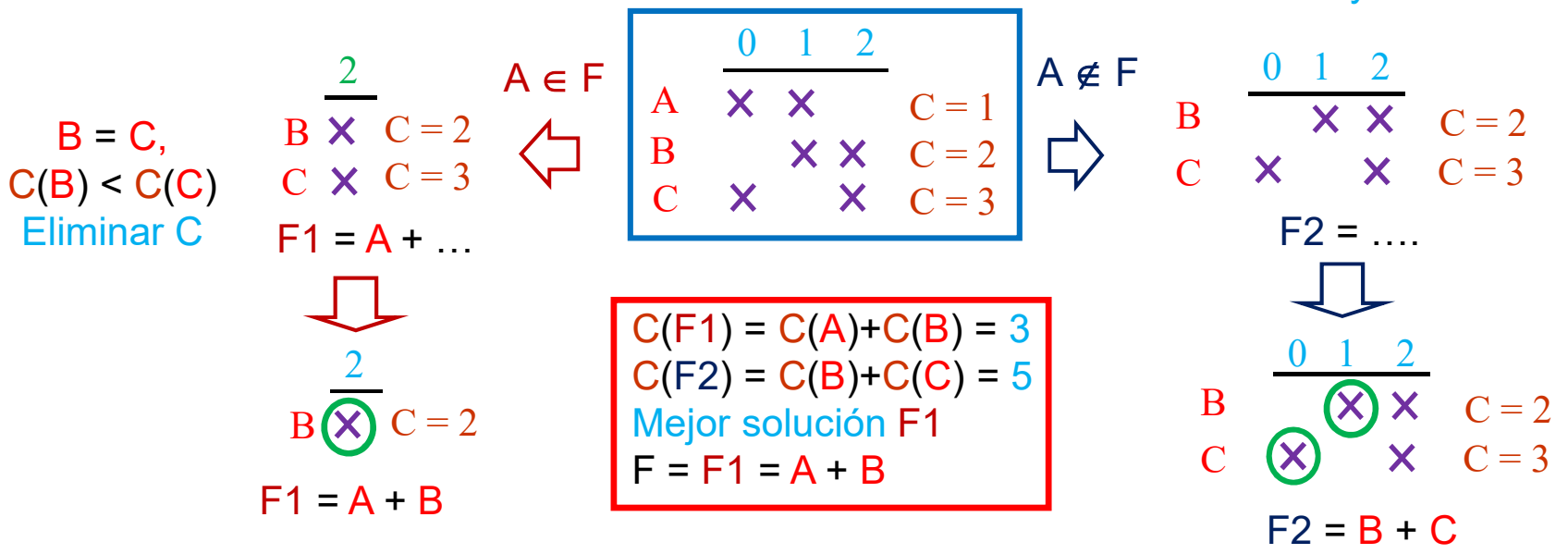
- Resolución de una tabla cíclica: tabla no vacía sin columnas con una única marca, ni eliminación de filas ni eliminación de columnas.

  Seleccionar un implicante IP de la tabla. Probar dos casos:

  a) F1: El implicante IP pertenece a la función => añadir IP a la función F1, incrementar su coste y eliminar las columnas cubiertas por IP. Continuar con el algoritmo.

  b) F2: El implicante IP no pertenece a la función => eliminar la fila IP de la tabla. Continuar con el algoritmo.

  F = F1 si Coste(F1) < Coste(F2); si no F = F2.



Método Quine-McCluskey

- El algoritmo puede adaptarse a minimización de varias funciones.

- Generación de implicantes primos en minimización de varias funciones:

  - Se añade a cada k-cubo una etiqueta por cada función que indica si pertenece (1) o no (0) a la función.

  - Al formar un (k+1)-cubo C a partir de dos k-cubos A y B, las etiquetas de C son el AND lógico de las etiquetas de A y B. Si todas las etiquetas de C son 0, C se elimina.

  - El (k+1)-cubo C cubre a los k-cubos A (o B) solo si las etiquetas de C son iguales a las etiquetas de A (o de B).

$F1(A, B, C) = \sum(0, 3, 4, 6)$

$F2(A, B, C) = \sum(0, 4, 6, 7)$

$F3(A, B, C) = \sum(0, 3, 5, 7)$

Paso 1

	Nº 1's	0-cubos	A	B	C	F1	F2	F3	
G[0][0]	0	0	0	0	0	1	1	1	$\bar{A}\bar{B}\bar{C}$ (a)
G[0][1]	1	4	1	0	0	1	1	0	✗
G[0][2]	2	3	0	1	1	1	0	1	$A\bar{B}\bar{C}$ (b)
		5	1	0	1	0	0	1	✗
		6	1	1	0	1	1	0	✗
G[0][3]	3	7	1	1	1	0	1	1	A B C (c)

Paso 2 ⇩

	Nº 1's	1-cubos	A	B	C	F1	F2	F3	
G[1][0]	0	(0,4)	-	0	0	1	1	0	$\bar{B}\bar{C}$ (d)
G[1][1]	1	~~(4,5)~~	~~1~~	~~0~~		~~0~~	~~0~~	~~0~~	
		(4,6)	1	-	0	1	1	0	$A\bar{C}$ (e)
G[1][2]	2	(3,7)	-	1	1	0	0	1	B C (f)
		(5,7)	1	-	1	0	0	1	A C (g)
		(6,7)	1	1	-	0	1	0	A B (h)

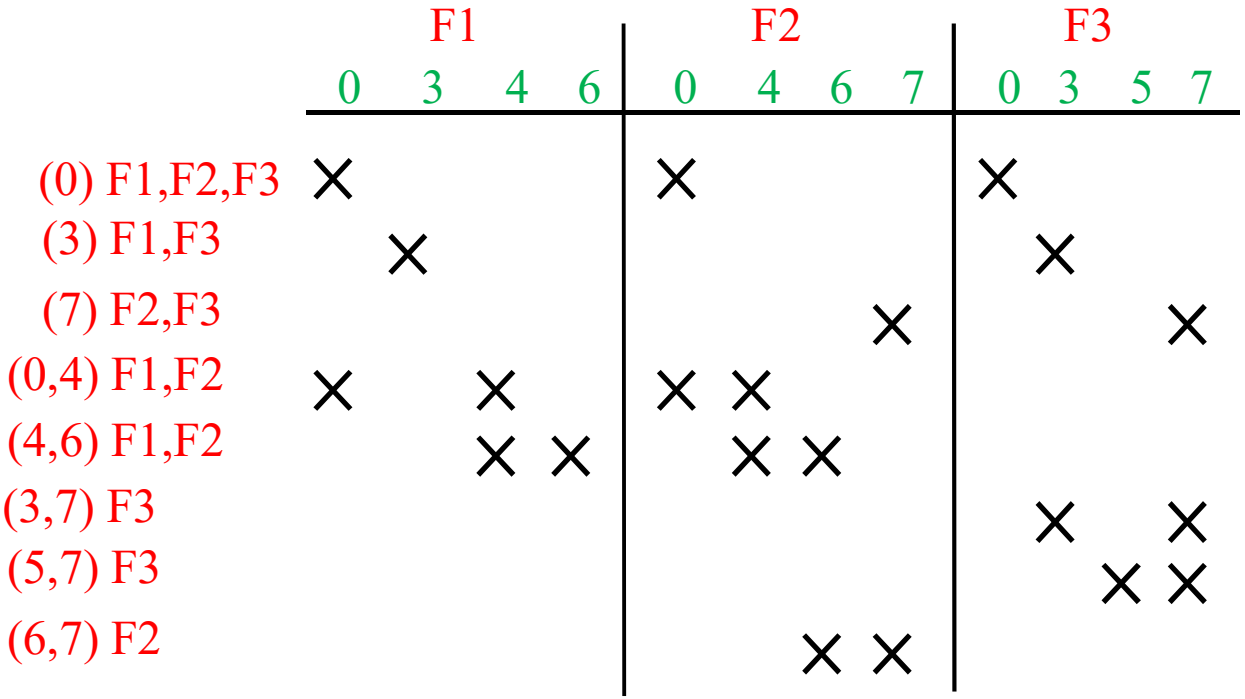
⇦ Paso 2

Nº 1's	2-cubos	A	B	C	F1	F2	F3
0							
1	~~(4,5,6,7)~~	~~1~~			~~0~~	~~0~~	~~0~~

Método Quine-McCluskey

- Selección de implicantes primos en minimización de varias funciones:

1. Se genera una tabla de implicantes primos frente a minterms de cada función. Los implicantes primos solo cubren minterms en las funciones a las que pertenecen



Criterio de coste a): número de implicantes. Todos los implicantes tienen coste 1

Método Quine-McCluskey

2. Se buscan columnas con una marca por función. Al incluir un implicante en una función se actualiza (se reduce) su coste para el resto de las funciones.

3. Realizar eliminación de filas en toda la tabla.

4. Realizar eliminación de columnas entre las columnas de cada función.

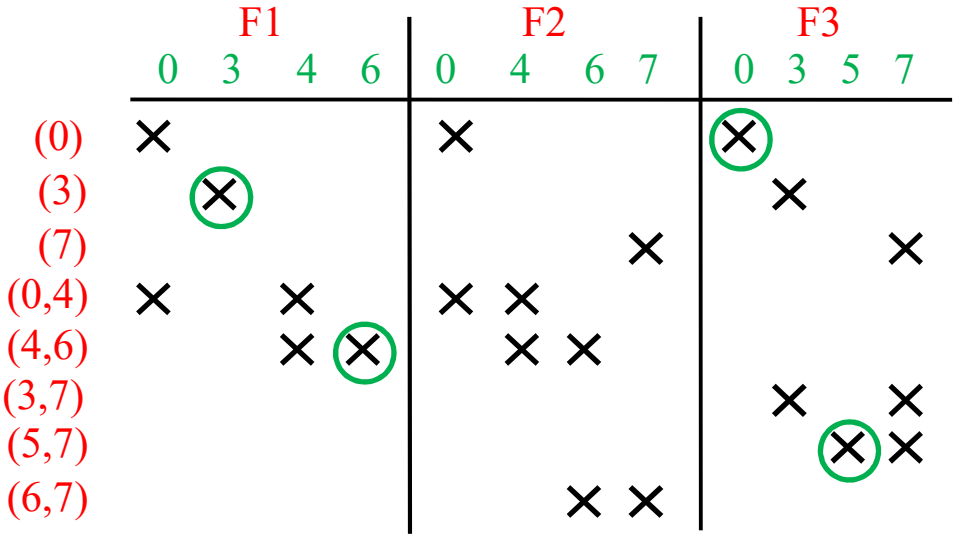
5. Si la tabla está vacía ir a 7; si quedan columnas: si los pasos 3 o 4 han modificado la tabla, volver a 2 y si no ir a 6.

6. Tabla cíclica. Resolverla como en la selección de implicantes de una función: buscar dos soluciones una incluyendo un implicante en una función (Fs1), y otra eliminándola de ella (Fs2).

7. Tomar una de las soluciones que quedan por probar (Fs1, Fs2, …) y volver a 2. Si se han probado todas las soluciones ir al paso 8.

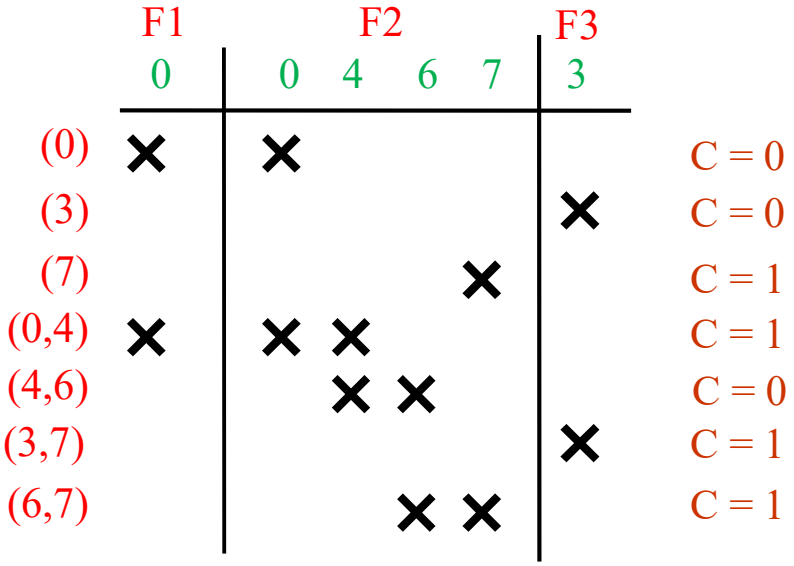
8. Seleccionar la solución de menor coste (si hubiese varias) y generar las funciones lógicas. Finalizar.

Método Quine-McCluskey



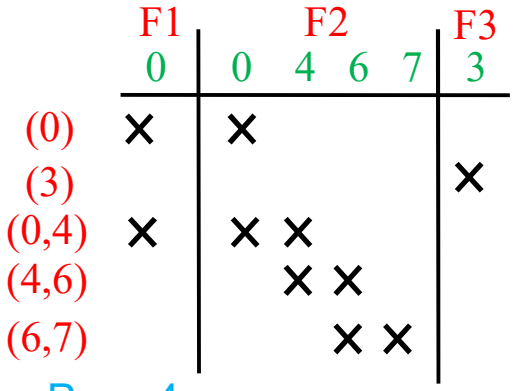
Criterio de Coste a). Menor número de implicantes.
Todas las filas tienen coste 1.

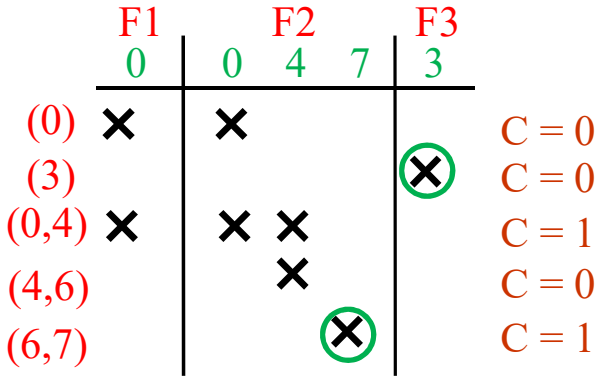
Paso 2:
F1 = (3) + (4,6) + …
F2 = …
F3 = (0) + (5,7) + …



Paso 3:
(6,7) incluye a (7)
(3,7) igual a (3)
con menor o igual coste.
Se eliminan (7) y (3,7)

Método Quine-McCluskey

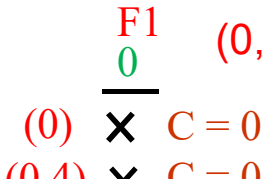




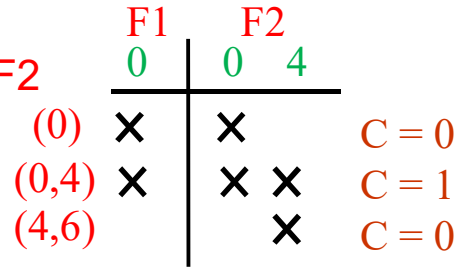
Paso 2.
F1 = (3) + (4,6) + …
F2 = (6,7) + …
F3 = (0) + (5,7) + (3)

Paso 4:
6F2 incluye a 7F2
Se elimina 6F2

(0) es igual a (0,4)
Se elimina (0,4)
Se cubre 0F1 con (0)
F1 = (3) + (4,6) + (0)
F2 = (6,7) + (0,4)
F3 = (0) + (5,7) + (3)
Coste Fs1: 6 implicantes

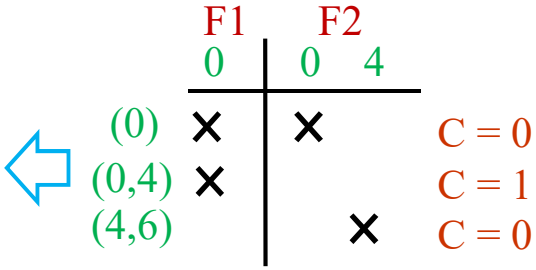


(0, 4) ∈ F2



Paso 5.
Tabla cíclica.
Prueba con (0,4): se incluye o no en F2

(0, 4) ∉ F2



(0,4) se incluye en (0)
Se elimina (0,4)
Se cubre 0F1 y 0F2 con (0), y 4F2 con (4,6)

F1 = (3) + (4,6) + (0)
F2 = (6,7) + (0) + (4,6)
F3 = (0) + (5,7) + (3)
Coste Fs2: 5 Implicantes
Mejor solución

Espresso

- Algoritmo estándar de minimización en dos niveles. Opera con problemas de más de 100 variables de entrada y/o salida de los que no se conoce cuál es la solución óptima.

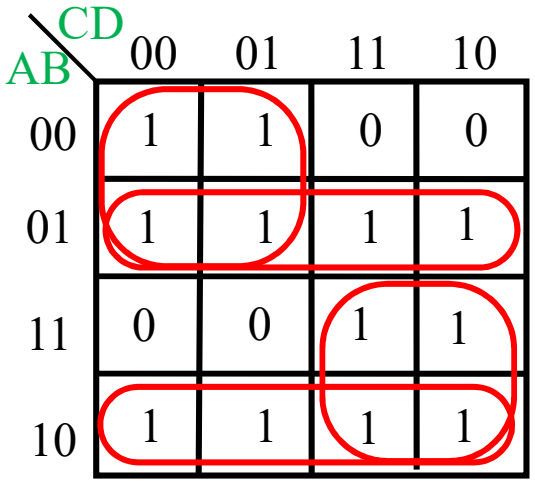
- Utiliza un método de aproximaciones sucesivas basado en tres procedimientos.

  1. Generación de implicantes primos (**Expand**): se parte de un conjunto de cubos que se expanden en determinadas variables para configurar un subconjunto de todos los implicantes primos que pueden cubrir las funciones.
  2. Se selecciona un conjunto mínimo de implicantes primos del conjunto anterior que cubra a todas las funciones (**Irredundant Cover**). En las iteraciones que se realizan se comprueba que esta solución encontrada es mejor que la solución anterior. Si no mejora la solución finalizar.
  3. El conjunto anterior se rehace reduciendo algunos implicantes primos a cubos en las zonas cubiertas por más de 1 implicante (**Reduce**). Se vuelve al paso 1.

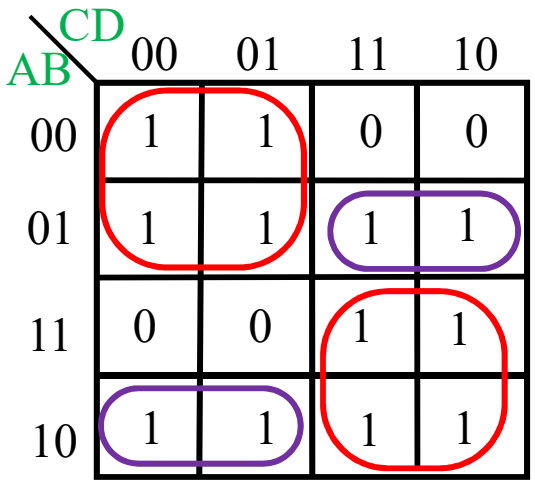
- Incluye además procedimientos extra como el cálculo de implicantes primos esenciales.

Espresso

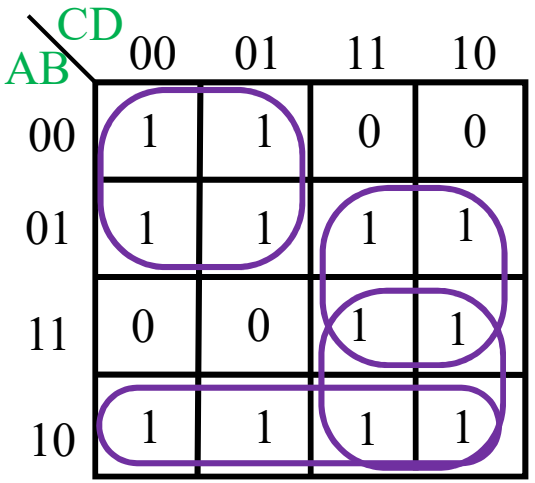
Expand + Irredundant Cover



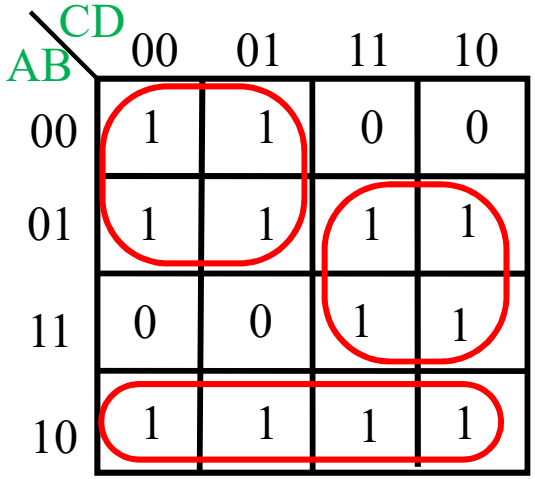
Reduce



Expand



Irredundant Cover



mi_fich

```
.i 3
.o 3
.ilb a b c
.ob f1 f2 f3
.type fd
.p 6
-00 101
001 00-
-10 110
011 1-1
101 -1-
111 0--
.e
```

número de entradas

número de salidas

nombre de cada entrada

nombre de cada salida

Indica el tipo de cubos activos f (ON-SET, 1s), d (DC-SET), r (OFF-SET, 0s) Por defecto fd. No se leen los valores no activos (0s, por defecto)

número de cubos

fin de cubos

Descripción por cubos: -10 110
Para cada entrada:
0 Literal complementado
1 Literal sin complementar
- No depende de la entrada
Para cada salida activada por .type:
1 El cubo pertenece al ON-set
0 El cubo pertenece al OFF-set
- El cubo pertenece al DC-set

espresso *-Dopt* mi_fich

Sin *–Dopt*: minimización iterativa conjunta.
-Dexact: minimización exacta conjunta.
-Dopo: minimización conjunta pero elige F o $\overline{F}$ para cada salida.
-Dso: minimización de cada salida por separado.
-Dso_both: minimización de cada salida por separado, elige F o $\overline{F}$.

```
.i 3
.o 3
.ilb a b c
.ob f1 f2 f3
.phase 111
.p 4
1-1 010
011 101
-10 110
-00 101
.e
```

Indica las salidas en las que se obtiene F (1) y en las que se obtiene $\overline{F}$ (0).

Descripción por cubos: -10 110
Para las entradas como en el fichero de entrada
Para cada salida:
1 El cubo pertenece a la salida
0 El cubo no pertenece a la salida

Síntesis multinivel

- En síntesis multinivel no se aplica el término minimización ya que no existe ningún método que garantice unas funciones lógicas mínimas. Se realizan una operaciones que mejoran de alguna manera el coste final de las expresiones, en principio reduciendo el número de literales del conjunto de las expresiones.

- La síntesis multinivel suele comenzar con una minimización en dos niveles. Al resultado de la minimización se le aplican operaciones típicas de síntesis multinivel y algoritmos complejos para llevarlas a cabo. Entre estas operaciones:

  - Factorización: pasar de dos niveles a forma factorizada.
  - Descomposición: reemplazar una expresión lógica por un conjunto de expresiones lógicas.
  - Substitución: expresar una función en términos de otra función.
  - Colapsado: elimina en una función la dependencia de otras funciones.
  - Extracción: identifica términos comunes en varias expresiones.

Factorización

- Una operación típica de la síntesis multinivel es la factorización: el paso de una expresión en dos niveles a una expresión factorizada.

- Esta operación es un punto de partida importante de la síntesis multinivel y es interesante encontrar una buena (si no la mínima) factorización. Un algoritmo recursivo sencillo (no óptimo) para conseguir una factorización dada una expresión lógica F sigue este procedimiento:

  - Localizar el literal X que más veces aparece en la expresión en dos niveles F y generar dos subexpresiones F1 y F2 de forma que F = X F1 + F2.

  - Repetir recursivamente el procedimiento para F1 y para F2, hasta que en las subexpresiones generadas en cada paso no aparezca ningún literal más de 1 vez.

Factorización

$F = A\,B\,\overline{D} + A\,D\,\overline{E} + \overline{A}\,\overline{B}\,C\,\overline{E} + \overline{A}\,\overline{B}\,D\,E + \overline{A}\,\overline{B}\,\overline{D}\,\overline{E} + B\,C\,E$

$\overline{A}$, $\overline{B}$ y $\overline{E}$ aparecen 3 veces. Elegimos $\overline{A}$ (habría que probar todos)

F1a F2a

$F = \overline{A}\,(\overline{B}\,C\,\overline{E} + \overline{B}\,D\,E + \overline{B}\,\overline{D}\,\overline{E}) + A\,B\,\overline{D} + A\,D\,\overline{E} + B\,C\,E$

$\overline{B}$ aparece 3 veces

A y B aparecen 2 veces.

Elegimos A

F1b F1c F2c

$F = \overline{A}\,(\overline{B}\,(C\,\overline{E} + D\,E + \overline{D}\,\overline{E})) + A\,(B\,\overline{D} + D\,\overline{E}) + B\,C\,E$

$\overline{E}$ aparece 2 veces

No se repiten literales: Fin

F1d F2d

$F = \overline{A}\,(\overline{B}\,(\overline{E}\,(C + \overline{D}) + D\,E)) + A\,(B\,\overline{D} + D\,\overline{E}) + B\,C\,E$

No se repiten literales: Fin

$F = \overline{A}\,\overline{B}\,(\overline{E}\,(C + \overline{D}) + D\,E) + A\,(B\,\overline{D} + D\,\overline{E}) + B\,C\,E$

Síntesis multinivel

- Descomposición: reemplazar una expresión lógica por un conjunto de expresiones lógicas.

$$F = A\,B\,C + A\,B\,D + \overline{A}\,\overline{C}\,\overline{D} + \overline{B}\,\overline{C}\,\overline{D} =$$
$$= A\,B\,(C + D) + (\overline{A} + \overline{B})\,\overline{C}\,\overline{D}$$

$$F = X\,Y + \overline{X}\,\overline{Y} \qquad X = A\,B \qquad Y = C + D$$

- Substitución: expresa una función en términos de otra función.

$$F = A + BC \text{ y } G = A + B \Rightarrow F = G\,(A + C)$$

- Colapsado: elimina en una función la dependencia en otras funciones.

$$F = G\,(A + C) \text{ y } G = A + B \Rightarrow F = A + BC$$

Extracción

- Identifica términos comunes en varias expresiones lógicas. Exige encontrar los factores de las expresiones.

- Se basa en operaciones de división algebraica (o booleana, más compleja) que reexpresa una expresión lógica en términos de divisores (cokernels) y de cocientes (kernels)

$F1 = ABC\ (1) + A\overline{C}G\ (2) + \overline{B}DF\ (3) + CDE\ (4)$
$F2 = \overline{A}B\overline{D}\ (5) + BC\overline{E}\ (6) + \overline{B}DE\ (7) + \overline{B}\,\overline{G}\ (8) + \overline{C}E\overline{G}\ (9)$

Función	Co-kernel	Kernel
F1	A	$BC + \overline{C}G$
F1	C	$AB + DE$
F1	D	$\overline{B}F + CE$
F2	B	$\overline{A}\,\overline{D} + C\overline{E}$
F2	$\overline{B}$	$DE + \overline{G}$
F2	$\overline{E}$	$BC + \overline{C}G$

$F1 = AX + DY$
$F2 = \overline{E}X + \overline{B}Z + \overline{A}\,\overline{B}\overline{D}$
$X = BC + \overline{C}G$
$Y = \overline{B}F + CE$
$Z = DE + \overline{G}$

Extracción

- La extracción se obtiene tomando el menor número de rectángulos (filas-columnas aunque no sean consecutivas) en una tabla entre los co-kernels y los cubos de los kernels, que cubran a todos los términos de las funciones.

$F1 = ABC\ (1) + A\overline{C}G\ (2) + \overline{B}DF\ (3) + CDE\ (4)$
$F2 = \overline{A}B\overline{D}\ (5) + BC\overline{E}\ (6) + \overline{B}DE\ (7) + \overline{B}\,\overline{G}\ (8) + \overline{C}\,\overline{E}G\ (9)$