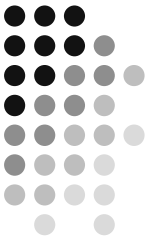
Diseño de Sistemas Digitales

Martín Vázquez,
E. Todorovich y M. Tosini



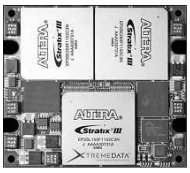
Materialización de Sistemas Lógicos

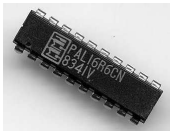
- Circuitos integrados específicos (ASIC - (Appication Specific Integrated Circuits).
- Circuitos Integrados programables por el usuario, por ejemplo PAL, CPLD, FPGAs, etc.
- Circuitos impresos que permiten ensamblar circuitos integrados (estándar, específicos o programables) y componentes discretos (resistencias, capacitores, transistores, etc)

Materialización de Sistemas Lógicos. Ejemplos:

Circuito aeroespacial ASIC Atmel. Mucho conocimiento de electrónica.

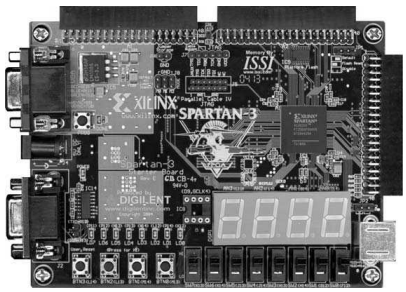
Dispositivos programables por el usuario: FPGA stratix III Altera y PAL. Conocimiento de diseño lógico de sistemas digitales.



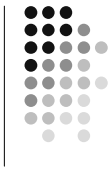


Materialización de Sistemas Lógicos. Ejemplos:

Placa de *Digilent* con componentes discretos y circuitos integrados. Conocimiento de electrónica, sobre todo cuando se diseña en varias capas y altas frecuencias

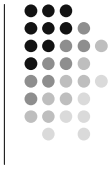


Diseño Lógico



- Independientemente de la implementación final del diseño. Las primeras etapas del diseño lógico son las mismas:
  - Descripción del diseño digital: esquemáticos, lenguajes de descripción de hardware (HDLs), o mediante lenguajes de alto nivel (HLL) tales como C, Handel-C, etc.
  - Simulación funcional y/o comportamental del diseño. Observar el comportamiento y, de ser necesario, corregir la descripción…

Síntesis Lógica



- La síntesis lógica es un proceso por el que se obtiene un circuito en términos de puertas lógicas (pertenecientes a bibliotecas genéricas o específicas) a partir de una descripción abstracta.
  - Tabla de verdad, ecuación lógica, etc.
  - Descripción en VHDL, Verilog, etc.

Generación de una expresión a partir de una tabla de verdad

- Una expresión se dice canónica si cada uno de sus términos hace referencia a todas las variables que intervienen en la expresión, ya sea en forma directa o negada.
- Las expresiones canónicas *conjuntivas* son sumas de términos conjuntivos (también llamados minterms).

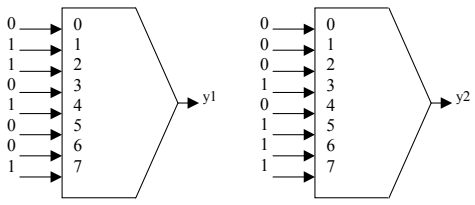
$$Y = \neg X2 \cdot \neg X1 \cdot X0 + \neg X2 \cdot X1 \cdot \neg X0 + X2 \cdot \neg X1 \cdot \neg X0$$

cada uno es un minterm.

x1	x2	x3	f(x1, x2, x3)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

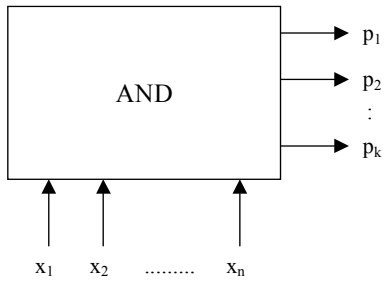
Síntesis mediante multiplexores

- Se emplea un multiplexor con tantas entradas de control como variables de entrada tenga el circuito.
- Cada una de las entradas del MUX (2^n) se conectan a ‘0’ ó a ‘1’ según corresponda al valor de la salida.



Planos de Puertas (Repaso)

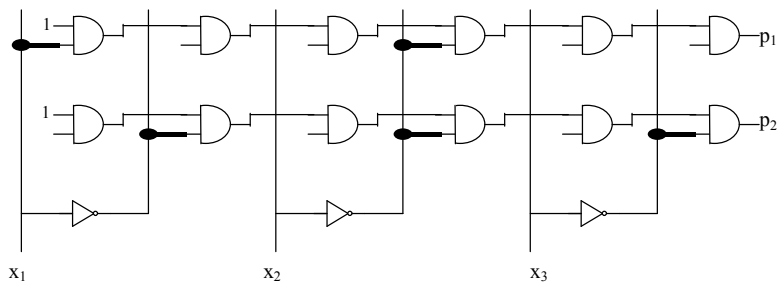
- Para el plano AND, la matriz de compuertas puede tener hasta Xn entradas y hasta pk salidas, donde cada salida pi es el producto de entradas X pudiendo estas estar en forma normal (Xi) o negada (¬ Xi).



Planos de Puertas (Repaso)

- Ejemplo con n = 3 y k = 2:
- Queremos que

  p1 = x1 · ¬ x2 p2 = ¬ x1 · ¬ x2 · ¬ x3



Planos de Puertas (Repaso)

- La programación del plano AND se realiza a través de una cadena de bits de tamaño

  2 * #x *#p.

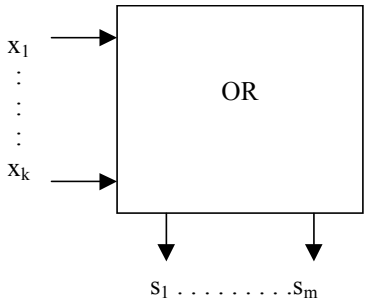
- En donde se representa con un ‘1’ cada posición del plano que debe ser conectada.
- Para el ejemplo:

  1 0 0 1 0 0 -> x1 · ¬ x2

  0 1 0 1 0 1 -> ¬ x1 · ¬ x2 · ¬ x3

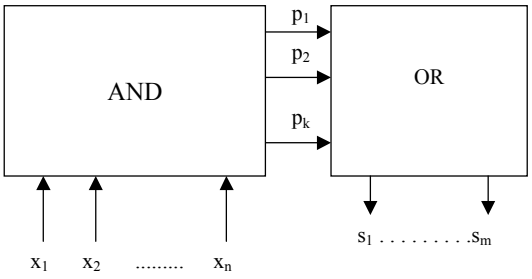
Planos de puertas (Repaso)

- En el plano OR, que es una matriz de compuertas OR interconectadas con k entradas –x1, x2, ..., xk- y m salidas –s1, s2, ..., sm-, cada s_i es la suma (booleana) de algunas variables de entrada, *siempre en forma normal*.



Planos de puertas (Repaso)

- El plano OR en conjunción con un plano AND permite sintetizar funciones booleanas completas mediante componentes denominados PLA (Programmable Logic Array's).



Síntesis mediante Planos de Puertas

- Ejemplo de dos funciones y1 e y2:

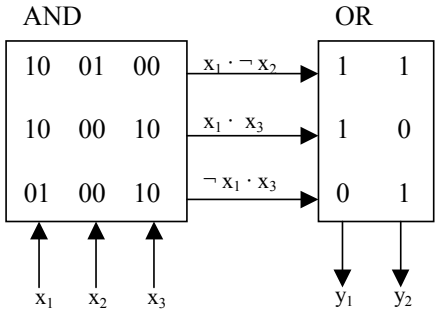
  $y1 = x1 \cdot \neg x2 + x1 \cdot x3$

  $y2 = x1 \cdot \neg x2 + \neg x1 \cdot x3$
- En el plano AND deben implementarse los términos:

  $x1 \cdot \neg x2$

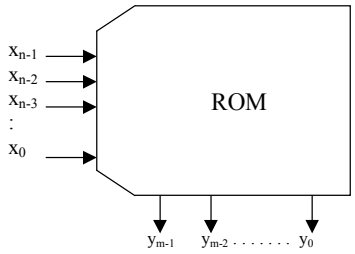
  $x1 \cdot x3$

  $\neg x1 \cdot x3$



Memorias ROM

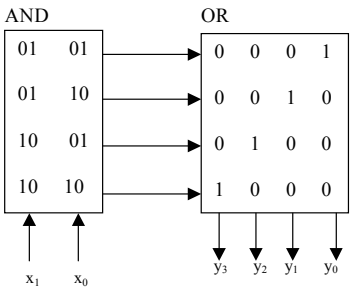
- Una memoria ROM es un dispositivo totalmente cableado. Ante cada combinación de valores de los xi da como resultado un determinado valor en cada salida yi.
- Las ROM son compatible con un plano AND-OR con tantas salidas (yi) como ancho de la celda de datos, y con *n* señales de entrada (xi).



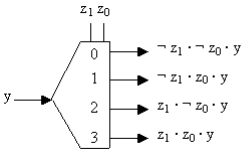
2^n palabras de m bits de ancho

Memorias ROM

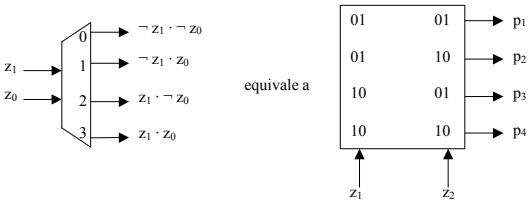
- La figura siguiente muestra un ejemplo de implementación de una memoria ROM pasiva de cuatro celds (n = 2) de 4 bits cada una (m = 4).



Decodificador, demultiplexor y plano AND



Demultiplexor de 1 a 4 con entrada, y = 1, implementa un Decodificador de 2 bits.



Decodificador de 2 bits, equivale a plano AND completo.

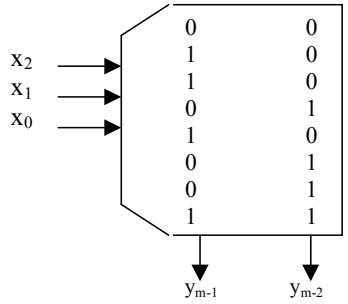
Memorias ROM

- El proceso de síntesis consiste en compilar (en un compilador de circuitos digitales) la secuencia de bits del plano OR.
- La secuencia de bits del plano AND es fija: secuencia de valores de las señales xi desde todos ‘0’ hasta todos ‘1’ (decodificador de direcciones).
- Por ejemplo, para una ROM de 4 palabras de 8 bits cada una con los valores:

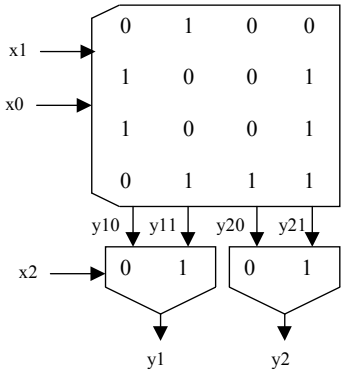
Dirección	Valor
0	AAh
1	FFh
2	03h
3	54h

- Plano AND :
  01 01
  01 10
  10 01
  10 10
- Plano OR :
  10101010
  11111111
  00000011
  01010100

Síntesis mediante memoria ROM



Síntesis mediante ROM+MUX



y11 = y1(x2 = 1)
y10 = y1(x2 = 0)
y21 = y2(x2 = 1)
y20 = y2(x2 = 0)

Generación de una expresión a partir de una tabla de verdad

- Una expresión canónica conjuntiva

$$Y = \neg X2 \cdot \neg X1 \cdot X0 + \neg X2 \cdot X1 \cdot \neg X0 + X2 \cdot \neg X1 \cdot X0$$

¿Se puede minimizar haciendo operaciones permitidas?

x2	X1	X0	f(x0, x1, x2)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Síntesis Lógica

- Técnicas y herramientas de minimización
  - Para tablas de verdad (circuitos combinacionales)
    - Mapas de Karnaugh (1953).
    - Algoritmo de Quine–McCluskey (1956).
    - Espresso heuristic logic minimizer (Brayton et al., 1984).

Mapas de Karnaugh

- Las simplificaciones que permiten los diagramas de Karnaugh se basan en la siguiente identidad:

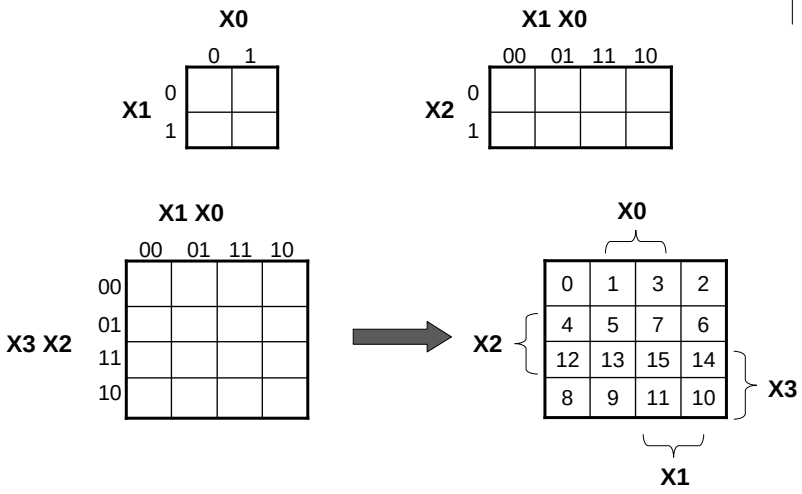
$$A \cdot B \cdot C + A \cdot B \cdot \neg C = A \cdot B \cdot (C + \neg C) = A \cdot B$$

- La ecuación anterior indica que si una variable (la C) aparece negada en un término y no negada en otro que tiene el resto de las variables iguales, puede eliminarse por completo. Los diagramas de Karnaugh ayudan mucho a la localización de estas variables que se pueden suprimir.

Mapas de Karnaugh

- Método manual, gráfico, útil hasta 5 o 6 variables.
- Las variables se ordenan de acuerdo al código de Gray (una sola variable cambia de polaridad entre cuadros adyacentes).
- Para obtener la expresión minimizada:
  - Agrupar los ‘1’ en rectángulos que contengan 1, 2, 4, 8 unos.
  - Cada casilla con un ‘1’ se debe cubrir al menos una vez.
  - Los valores indefinidos (don’t care) pueden aprovecharse para hacer grupos de ‘1’ mas grandes.

Mapas de Karnaugh



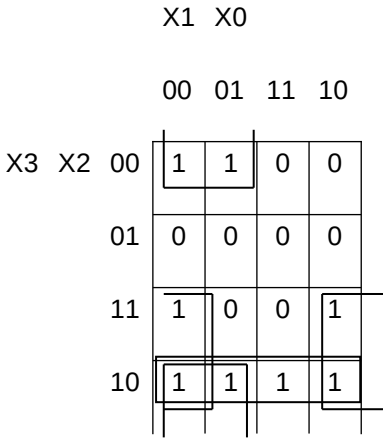
Mapas de Karnaugh

x3	x2	x1	x0	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0

x3	x2	x1	x0	F
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	0
1	1	1	0	1
1	1	1	1	0

F = ¬X3·¬X2·¬X1·¬X0 + ¬X3·¬X2·¬X1·X0 + X3·¬X2·¬X1·¬X0 + X3·¬X2·¬X1·X0 + X3·¬X2·X1·¬X0 + X3·¬X2·X1·X0 + X3·X2·¬X1·¬X0 + X3·X2·X1·¬X0

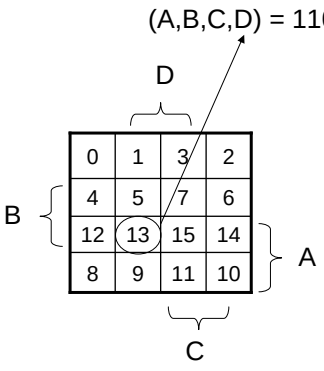
Mapas de Karnaugh

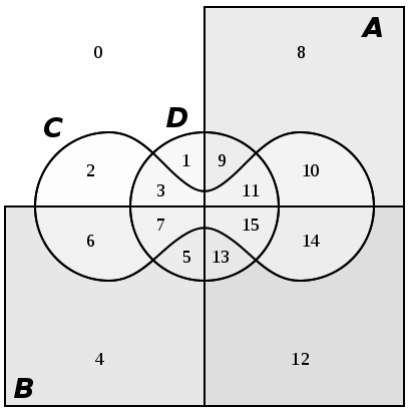


$F = \neg X2 \cdot \neg X1 + X3 \cdot \neg X0 + X3 \cdot \neg X2$

Mapas de Karnaugh

- Un mapa de Karnaugh se puede ver como un diagrama de Venn especialmente dibujado:





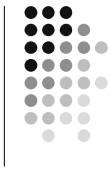
Mapas de Karnaugh

- Éste es un método gráfico que posee limitaciones muy fuertes (casi imposible de hacer) cuando se desea simplificar funciones de más de 6 variables.
- No es determinístico, hay varias maneras de agrupar.
- No necesariamente se encuentra la expresión mínima.

Método tabular de Quine–McCluskey

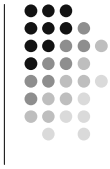
- Mismo funcionamiento que método de Karnaugh.
- Se puede implementar en un programa de computación. Posibilita la síntesis automática.
- Permite encontrar de forma determinística la expresión mínima de una función booleana.

Método tabular de Quine–McCluskey



- Primero encuentra los implicantes primos.
  - Se parte de la función canónica conjuntiva.
  - Se efectúa un proceso iterativo en el que se van combinando términos.
- Una vez encontrado los implicantes primos, obtiene las expresiones mínimas candidatas a partir de la cobertura mínima de los mintérminos involucrados en la función.
- Por último, escoge la/s función/es con mínimo costo según algún criterio: área, profundidad lógica, etc.

Implicantes primos



- Dada una función expresada como suma de conjunciones. Los implicantes primos son las expresiones que no están contenidas en ninguna otra.

Ejemplo, sea la función booleana $F = X2 \cdot \neg X1 \cdot \neg X0 + \neg X2 \cdot X0 + X2 \cdot \neg X0$,

Implicantes primos son: $\neg X2 \cdot X0$ y $X2 \cdot \neg X0$

$X2 \cdot \neg X1 \cdot \neg X0$ está contenida en $X2 \cdot \neg X0$, ya que

$X2 \cdot \neg X0 = X2 \cdot \neg X0 \cdot X1 + X2 \cdot \neg X0 \cdot \neg X1$

Método tabular de Quine–McCluskey

Ejemplo: F= X2·X1·X0 + ¬X3·¬X0 + ¬X3·X1·X0 + X3·¬X2·X0 + X3·¬X1·X0

Para encontrar los implicantes primos, primero se deben encontrar los mintérminos involurados en la función.

¬X3·¬X0 = ¬X3·¬X2·¬X1·¬X0 + ¬X3·X2·¬X1·¬X0 + ¬X3·¬X2·X1¬X0 + ¬X3·X2·X1·¬X0

¬X3·¬X0 genera los mintérminos: m0, m2, m4, m6

X2·X1·X0 ⇒ m7, m15

¬X3·X1·X0 ⇒ m3, m7

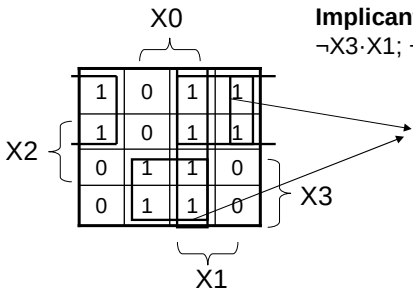
X3·¬X2·X0 ⇒ m9 y m11

X3·¬X1·X0 ⇒ m9 y m13

Método tabular de Quine–McCluskey

Entonces F = (0, 2, 3, 4, 6, 7, 9, 11, 13, 15)

Implicantes primos y esenciales identificados con Karnaugh



Implicantes Primos
¬X3·X1; ¬X3·¬X0; X3·X0; X1·X0

Implicantes esenciales
¬X3·¬X0 y X3·X0

Generación de implicantes primos

- A partir de la expresión canónica conjuntiva, se determina el índice de cada término. El índice es el número de componentes que contengan uno.

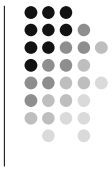
minterms	código	índice	valor decimal
m0	0000	0	0
m2	0010	1	2
m4	0100	1	4
m3	0011	2	3
m6	0110	2	6
m9	1001	2	9
m7	0111	3	7
m11	1011	3	11
m13	1101	3	13
m15	1111	4	15

Generación de implicantes primos

- Se itera combinando términos mediante la aplicación de la siguiente regla:

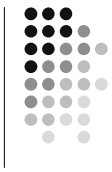
Los términos a combinar no deben diferir entre si, más que en el estado de una de las variables, la cuál será sustituida por un guión

Generación de implicantes primos



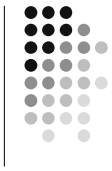
minterms	Implicantes de tam. 2		Implicantes de tam. 4		
m0	**m(0,2)**	00_0	m(0,2,4,6)	0__0	
m2	**m(0,4)**	0_00	m(0,4,2,6)	0__0	**elimina**
m4	**m(2,3)**	001_	m(2,3,6,7)	0_1_	
m3	**m(2,6)**	0_10			
m6	**m(4,6)**	01_0			
m9	m(3,7)	0_11			
m7	m(3,11)	_011			
m11	**m(6,7)**	011_			
m13	m(9,11)	10_1			
m15	m(9,13)	1_01			
	m(7,15)	_111			
	m(11,15)	1_11			
	m(13,15)	11_1			

Generación de implicantes primos



minterms	Implicantes de tam. 2		Implicantes de tam. 4		
m0	**m(0,2)**	00_0	m(0,2,4,6)	0__0	
m2	**m(0,4)**	0_00	m(0,4,2,6)	0__0	**elimina**
m4	**m(2,3)**	001_	m(2,3,6,7)	0_1_	
m3	**m(2,6)**	0_10	m(3,7,11,15)	__11	
m6	**m(4,6)**	01_0	m(3,11,7,15)	__11	**elimina**
m9	**m(3,7)**	0_11			
m7	**m(3,11)**	_011			
m11	**m(6,7)**	011_			
m13	m(9,11)	10_1			
m15	m(9,13)	1_01			
	m(7,15)	_111			
	m(11,15)	1_11			
	m(13,15)	11_1			

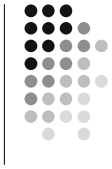
Generación de implicantes primos



minterms	Implicantes de tam. 2		Implicantes de tam. 4		
m0	**m(0,2)**	00_0	m(0,2,4,6)	0__0	
m2	**m(0,4)**	0_00	m(0,4,2,6)	0__0	**elimina**
m4	**m(2,3)**	001_	m(2,3,6,7)	0_1_	
m3	**m(2,6)**	0_10	m(3,7,11,15)	__11	
m6	**m(4,6)**	01_0	m(3,11,7,15)	__11	**elimina**
m9	**m(3,7)**	0_11	m(9,11,13,15)	1__1	
m7	**m(3,11)**	_011	m(9,13,11,15)	1__1	**elimina**
m11	**m(6,7)**	011_			
m13	**m(9,11)**	10_1			
m15	**m(9,13)**	1_01			
	m(7,15)	_111			
	m(11,15)	1_11			
	m(13,15)	11_1			

Implicantes primos encontrados
¬X3·¬X0; ¬X3·X1; X1·X0; X3·X0

Generación de cobertura mínima



- Se construye tabla de modo que:
  - filas son los implicantes primos encontrados.
  - columnas los mintérminos de la función.
  - identifica los mintérminos cubiertos por los implicantes primos

		m0	m2	m3	m4	m6	m7	m9	m11	m13	m15
a	**m(0,2,4,6)**	x	x		x	x					
b	**m(2,3,6,7)**		x	x		x	x				
c	**m(3,7,11,15)**			x			x		x		x
d	**m(9,11,13,15)**							x	x	x	x

Generación de cobertura mínima

- Identificar implicantes esenciales y generar tabla reducida

		m0	m2	m3	m4	m6	m7	m9	m11	m13	m15
a	m(0,2,4,6)	x	x		x	x					
b	m(2,3,6,7)		x	x		x	x				
c	m(3,7,11,15)			x			x		x		x
d	m(9,11,13,15)							x	x	x	x

Generación de cobertura mínima

- Identificar implicantes esenciales y generar tabla reducida

		m0	m2	m3	m4	m6	m7	m9	m11	m13	m15
a	m(0,2,4,6)	x	x		x	x					
b	m(2,3,6,7)		x	x		x	x				
c	m(3,7,11,15)			x			x		x		x
d	m(9,11,13,15)							x	x	x	x

Tabla reducida

		m3	m7
b	m(2,3,6,7)	x	x
c	m(3,7,11,15)	x	x

Generación de cobertura mínima

- Reglas para tabla reducida:
  - Dos implicantes primos son intercambiables si poseen el mismo costo.
  - Se dice que un implicante primo ***y*** domina a otro ***z***, si tiene todas las marcas de ***z*** y alguna adicional.
  - Tras eliminar las filas se busca implicantes esenciales: el único que cubre una columna
  - Si no quedan cubiertos todos los mintérminos se repite el proceso.

Generación de cobertura mínima

En nuestro ejemplo son intercambiables

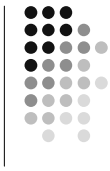
		m3	m7
b	m(2,3,6,7)	x	x
c	m(3,7,11,15)	x	x

Solución es a·d·(b+c), **entonces**

F = ¬X3·¬X0 + X3·X0 + ¬X3·X1 o bien

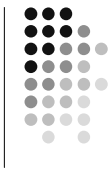
F = ¬X3·¬X0 + X3·X0 + X1·X0

Algoritmo de Quine–McCluskey



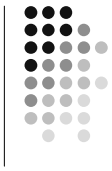
- Si bien el algoritmo de Quine–McCluskey se puede implementar en un programa, su complejidad temporal y espacial no son buenas (exponencial). Es un método exhaustivo, es decir que agota todas las posibilidades.
- Para resolver problemas de muchas variables se deben usar métodos heurísticos sub-óptimos.

Espresso



- Espresso es el algoritmo estándar de síntesis automática de funciones booleanas.
  - Es un algoritmo *greedy* que aplica operaciones iterativamente
  - Se parte de diferentes puntos y se aplican operaciones hasta que no haya mejoras
  - Se puede caer en mínimos locales

Espresso



- Espresso esta disponible en la web como código open-source:
  - http://diamond.gem.valpo.edu/~dhart/ece110/espresso/tutorial.html Programa para bajar más instrucciones sobre cómo operarlo.
  - http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/index.htm El código fuente original se puede bajar de la página de Berkeley.