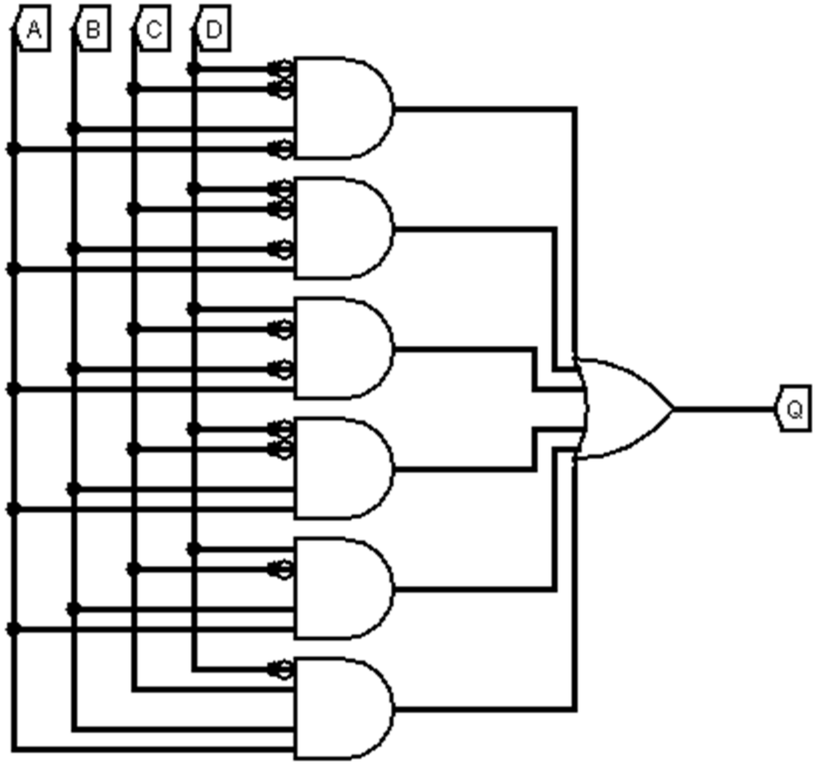
66.70 Estructura del Computador

Diseño de circuitos combinacionales

¿Qué es **ANALIZAR** un circuito?

Dado un circuito lógico determinar el comportamiento de dicho circuito



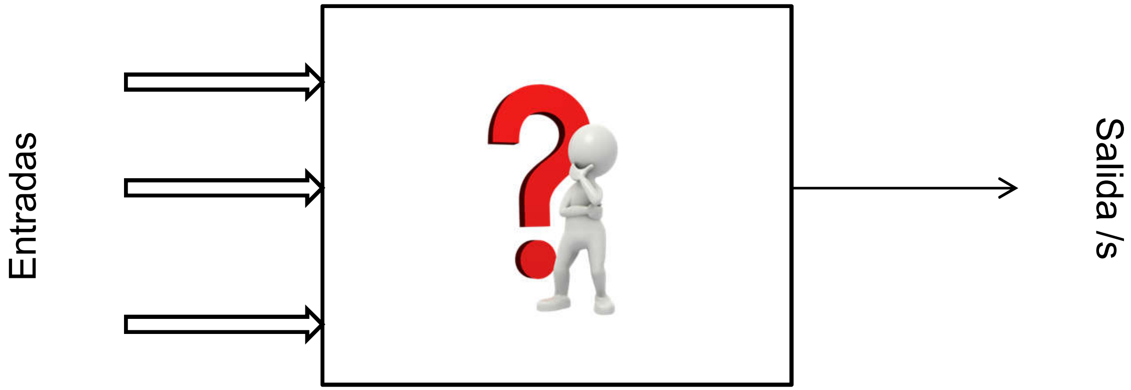
$$Q = f(A, B, C, D) = ?$$

¿Qué hace este circuito?

¿Qué es **DISEÑAR** un Circuito?

Dado un problema encontrar un circuito lógico que resuelva dicho problema

Ejemplo: Diseñar un circuito que controle el encendido de una alarma sonora en función de varios sensores de temperatura, presencia, apertura de puertas y ventanas, etc.



Expr. informal -> Expresión formal -> Expresión mínima -> Circuito -> implementación

Lógica combinacional

- Lógica de dos niveles
  - Suma de productos
  - Producto de sumas
- Lógica multinivel

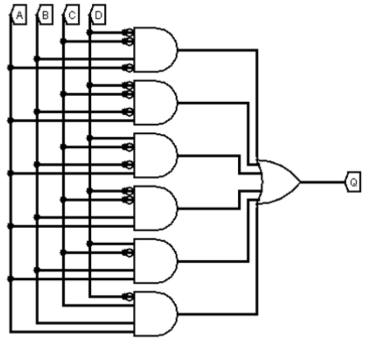
Ejemplo: $x'y + xy' + xz = x'y + x(y' + z)$

Complejidad de una solución

- ¿Cómo podemos medir la complejidad de una solución?

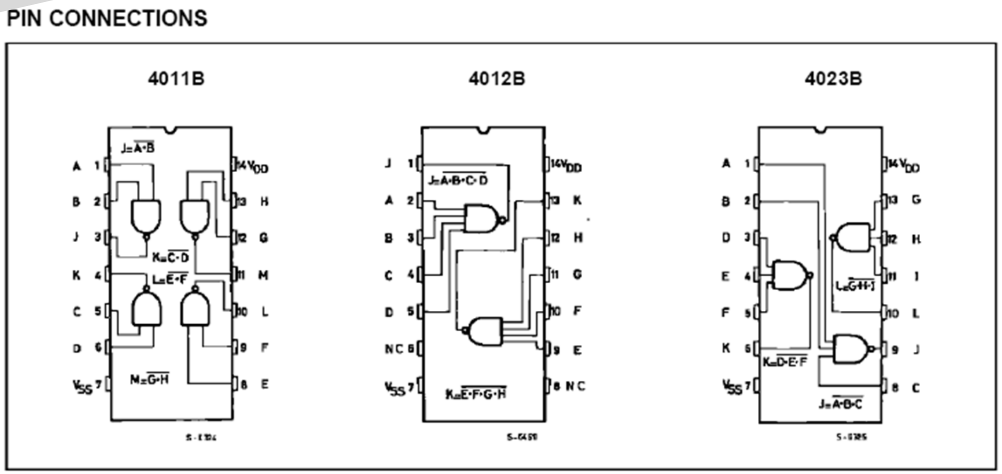
- ¿Cómo podemos comparar dos expresiones equivalentes?

Complejidad de una solución



- $Comp = \sum Tc.Ce$
  - Tc: Tipo de compuerta
  - Ce: Cantidad de entradas
  - Ej: $Comp = 6_{AND}.4_{Entradas} + 1_{OR}.6_{Entradas} = 26$

Las compuertas tienen una cantidad limitada de entradas



Métodos de Simplificación

- **Algebraico**
  - Se trabaja directamente sobre la expresión algebraica (prueba y error)
  - Se basa en eliminar términos y literales aplicando los postulados y los teoremas del Algebra de Boole
- **Gráfico**
  - Mapa de Karnaugh
- **Tabular**
  - Algoritmo de Quine-McCluskey

Simplificación por método algebraico

Ejemplo:

$$F = A'C' + ABC + BC' + A'B'C + A'BC$$

$$F = A'C' + BC' + BC(A + A') + A'C(B + B')$$

$$F = A'C' + BC' + BC + A'C$$

$$F = A'(C' + C) + B(C' + C)$$

$$F = A' + B$$

Método algebraico

Características

- No incluye un procedimiento formal que asegure llegar a una expresión mínima
- Proclive a que se comentan errores de copia en los literales
- Se torna difícil con más de 4 o 5 variables

Método algebraico

ABCD	Q
0000	0
0001	0
0010	0
0011	0
0100	1
0101	0
0110	0
0111	0
1000	1
1001	1
1010	0
1011	0
1100	1
1101	1
1110	1
1111	0

$$\ldots + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}\,D + \ldots$$

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

$$A\overline{B}\,\overline{C}$$

Para la combinación
A=1; B=0; C=0
la función vale 1
independientemente del
valor de D

¡¡SON ADYACENTES!!

Mapa de Karnaugh

- Relación entre el mapa de K y la tabla de verdad
- Definiciones:
  - Adyacencias
  - Implicante primo
  - Implicante primo esencial
- Permite encontrar las expresiones mínimas en forma de Suma de Productos o Producto de Sumas
  (mínima cantidad de términos y mínima cantidad de entradas)
- Se basa en: (1) encontrar todos los implicantes primos
  (2) seleccionar un conjunto mínimo de implicantes que cubra la función

Mapa de Karnaugh

Por los 1's de la función

F(A, B, C, D)

Tabla de verdad

1		1	1
	1	1	
1	1		
1			1

1. Marcar implicantes primos
2. Marcar Imp. primos esenciales
3. Construir expr.algebr c/ IPE
4. Agregar 1's hasta completar F

EXPRESION/es MÍNIMA/s

"Suma de Productos"

Resolver:

F = A'C' + ABC + BC' + A'B'C + A'BC

Mapa de Karnaugh

Por los 1's de la función

Ejemplo 1

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

Ejemplo 2

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

Ejemplo 3

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

Mapa de Karnaugh

Por los 0’s de la función

F(A, B, C, D)

Tabla de verdad

	0		
0			0
		0	0
	0	0	

1. Marcar implicantes primos
2. Marcar Imp. primos esenciales
3. Construir expr.algebr c/ IPE
4. Agregar 0’s hasta completar F

EXPRESION/es MÍNIMA/s

“Producto de Sumas”

Resolver:

F = A’C’ + ABC + BC’ + A’B’C + A’BC

Aplicar Mapas de Karnaugh

Para un proceso que se produce en una fábrica, nos pidien diseñar un circuito que encienda una alarma según tres sensores T1, T2 y P

- T1 sensa la temperatura del proceso
  - T1 = 1 si T > 50º
  - T1 = 0 caso contrario
- T2 también sensa la temperatura del proceso
  - T2 = 1 si T < 10º
  - T2 = 0 caso contrario
- P sensa la presión del proceso
  - P = 1 si Presión > 10atm
  - P = 0 caso contrario

La alarma deberá sonar siempre que T sea menor a 10º y P sea mayor a 10atm o siempre que la temperatura supere los 50º

Diseñar <u>el circuito más simple</u> que cumpla con esa función.

Redundancias

o “Funciones incompletamente especificadas”

- Significado
- Como manejarlas en los Mapas de Karnaugh
  - ✓ Cuando conviene incluirlas en un implicante
  - ✓ Relación entre las redundancias de distintos implicantes
  - ✓ Redundancias e implicantes primos esenciales

Simplificación por mapas de Karnaugh

Con redundancias

F(A, B, C, D)

Tabla de verdad

1	x	1	1
	x	x	
	1	1	

“Aprovechar” redundancias

Sólo sin son útiles para simplificar

1. Marcar implicantes primos
2. Marcar Imp. primos esenciales
3. Construir expr.algebr c/ IPE
4. Agregar 1’s hasta completar F

EXPRESION/es MÍNIMA/s

“Producto de Sumas”

Mapas de Karnaugh de 5 variables

Vecindades?

E=0

1			1
	X	X	
X			1

E=1

1			1
1	1	1	1
X			X

Mapas de Karnaugh de 6 variables

Vecindades?

F=0

A=0

1			X
	1	X	
	1	1	
1			1

A=1

1			1
	X	X	
X			1

F=1

A=0

X			1
1	X	X	1
	X	X	
1			1

A=1

1			1
1	1	1	1
X			X

Simplificación por mapa de Karnaugh

- Ventajas
  - Da un procedimiento formal hacia la expresión mínima
  - Aplicable para S de P y para P de S
  - Fácil de aplicar (con pocas variables)

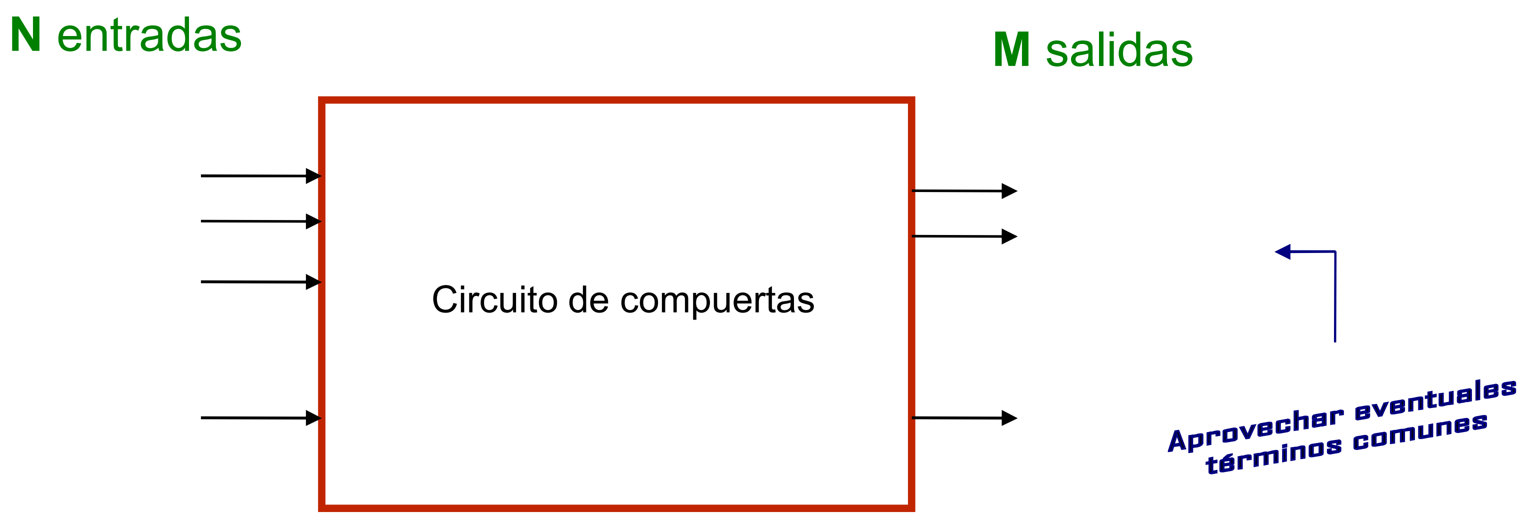
- Desventajas
  - No es aplicable a más de 5 o ¿6? variables
  - Depende de la habilidad visual y experiencia
  - No es apropiado para implementar en software

Bajando más los costos

- ✓ Reducir número de compuertas y mínimo número de entradas
- Elegir entre solución por suma de productos o por producta de sumas
- Reducir el número de inversores
- Reducir el número de circuitos integrados

  (los CI comerciales incluyen varias compuertas en el mismo chip dependiendo del número de entradas)
- Utilizar sólo compuertas NAND
  - Ventajas:
    - Menor costo que AND OR
    - Unificar el tipo de compuertas utilizadas en la implementación
  - Como?
- Compuertas NOR: idem NAND

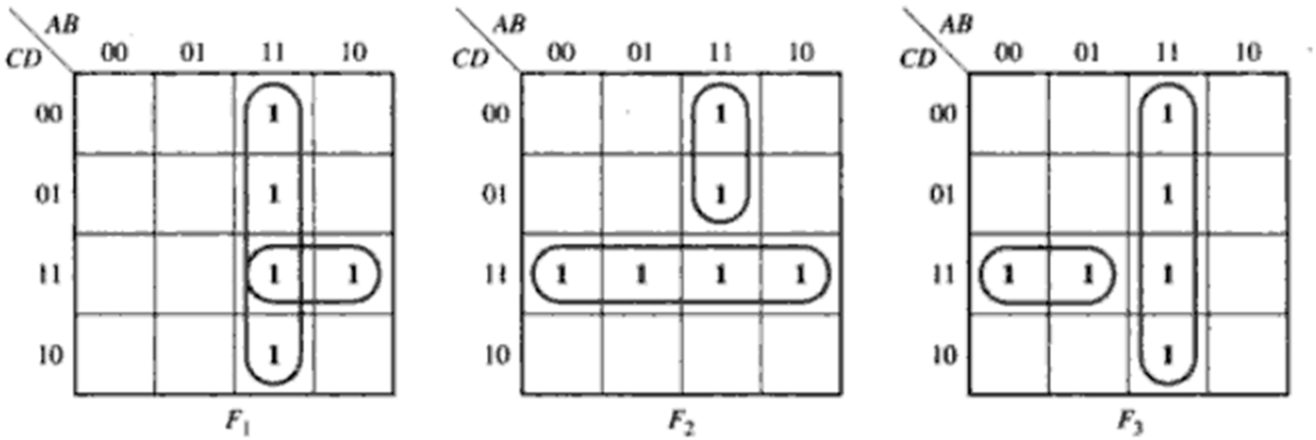
Problemas de salida múltiple



Ejemplo: *columna de 8 leds encendida en correspondencia con datos de tres bits a la entrada*

Problemas de salida múltiple

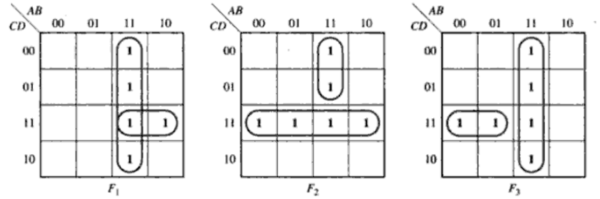
$F_1(A, B, C, D) = \Sigma\, m(11, 12, 13, 14, 15)$
$F_2(A, B, C, D) = \Sigma\, m(3, 7, 11, 12, 13, 15)$
$F_3(A, B, C, D) = \Sigma\, m(3, 7, 12, 13, 14, 15)$



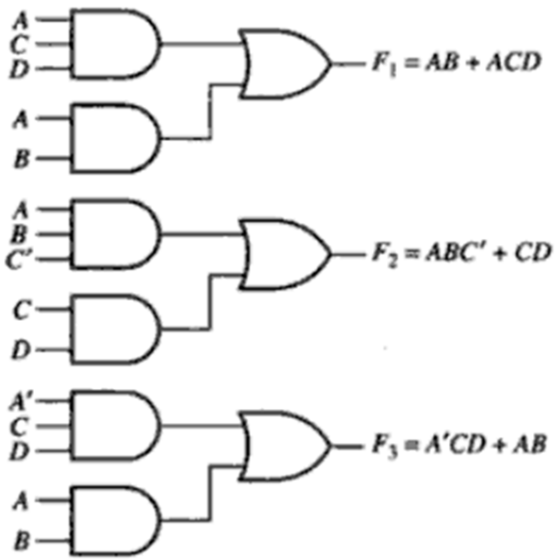
¿Qué compuertas puedo ahorrar respecto del problema de salida única?

Problemas de salida múltiple

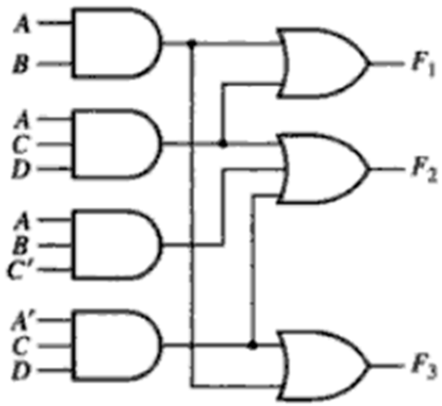
$$F_1(A, B, C, D) = \Sigma\, m(11, 12, 13, 14, 15)$$
$$F_2(A, B, C, D) = \Sigma\, m(3, 7, 11, 12, 13, 15)$$
$$F_3(A, B, C, D) = \Sigma\, m(3, 7, 12, 13, 14, 15)$$



Implementación directa de F_1 F_2 y F_3



Consideradas como salida múltiple



Pasos para diseñar un circuito lógico combinacional

1. *Planteo informal del problema*
2. *Identificación de variables dependientes e independientes*
3. *Formalizar las salidas como funciones lógicas*
4. *Encontrar todas las expresiones mínimas posibles (por 1's y por 0's)*

1. *Diagrama circuital de una de esas expr. mínimas (cuál?)*
2. *Elegir circuitos integrados (un único tipo de compuerta?)*
3. *Implementación física*

Ver tranparencia: "Bajando más los costos"

Algoritmo de Quine-McCluskey

- Resulta apropiado para implementarlo en **software**
- Se organiza en forma **tabular**
- No impone límites, en principio, sobre el **número de variables**

Algoritmo de Quine-McCluskey

Básicamente consiste en:

1. **Eliminar** tanto literales como sea posible **aplicando sistemáticamente** $XY + XY' = X$

2. Usar una **tabla de implicantes primos** para seleccionar un conjunto mínimo de implicantes primos que combinados por medio de OR producen la función a simplificar

Algoritmo de Quine-McCluskey

Pueden combinarse

AB'CD' + AB'CD = AB'C
1 0 1 0 + 1 0 1 1 = 1 0 1 -

No pueden combinarse

A'BC'D + A'BCD'
0 1 0 1 + 0 1 1 0

1. Encontrar todos los implicantes primos
   1. Agrupar minitérminos según la cantidad de 1's
   2. Comparar grupos adyacentes solamente
   3. Combinar minitérminos -> implicantes
   4. Combinar implicantes en pasos sucesivos (tildar cada implicante usado en cada combinación)
   5. Eliminar implicantes duplicados

2. Elegir un conjunto mínimo de implicantes primos
   1. Construir la tabla de implicantes con:
      a. Los implicantes de menor orden que no fueron tildados
      b. Los implicantes de mayor orden
   2. Elegir los implicantes primos esenciales
   1. Completar por medio de otros implicantes primos todos los minitérminos de la función

Ejemplo: **$F(A,B,C,D) = \sum m(0,1,2,5,6,7,8,9,10,14)$**

Algoritmo de Quine-McCluskey

$$f(a,b,c,d)=\sum m(0,1,2,5,6,7,8,9,10,14)$$

Paso 1 *Agrupar minitérminos según la cantidad de 1's*

grupo 0	0	0000
grupo 1	1	0001
	2	0010
	8	1000
grupo 2	5	0101
	6	0110
	9	1001
	10	1010
grupo 3	7	0111
	14	1110

Sólo debemos comparar grupos adyacentes

Algoritmo de Quine-McCluskey

$$f(a,b,c,d)=\sum m(0,1,2,5,6,7,8,9,10,14)$$

Paso 2 (Combinar implicantes de grupos vecinos)

Columna I

grupo 0	0	0000
grupo 1	1	0001
	2	0010
	8	1000
grupo 2	5	0101
	6	0110
	9	1001
	10	1010
grupo 3	7	0111
	14	1110

Columna II

0,1	000-
0,2	00-0
0,8	-000
1,5	0-01
1,9	-001
2,6	0-10
2,10	-010
8,9	100-
8,10	10-0
5,7	01-1
6,7	011-
6,14	-110
10,14	1-10

Algoritmo de Quine-McCluskey

$$f(a,b,c,d)=\sum m(0,1,2,5,6,7,8,9,10,14)$$

Paso 3 (Agrupar la columna 2 y combinar implic. de grupos vecinos)

Columna I

grupo 0	0	0000	✓
grupo 1	1	0001	✓
	2	0010	✓
	8	1000	✓
grupo 2	5	0101	✓
	6	0110	✓
	9	1001	✓
	10	1010	✓
grupo 3	7	0111	✓
	14	1110	✓

Columna II

0,1	000-	✓
0,2	00-0	✓
0,8	-000	✓
1,5	0-01	
1,9	-001	✓
2,6	0-10	✓
2,10	-010	✓
8,9	100-	✓
8,10	10-0	✓
5,7	01-1	
6,7	011-	
6,14	-110	✓
10,14	1-10	✓

Columna III

0,1,8,9	-00-
0,2,8,10	-0-0
0,8,1,9	-00-
0,8,2,10	-0-0
2,6,10,14	--10
2,10,6,14	--10

Algoritmo de Quine-McCluskey

$$f(a,b,c,d)=\sum m(0,1,2,5,6,7,8,9,10,14)$$

Paso 4 (Eliminar combinaciones repetidas)

Columna I

Grupo	Minterm	Binario	
grupo 0	0	0000	✓
grupo 1	1	0001	✓
	2	0010	✓
	8	1000	✓
grupo 2	5	0101	✓
	6	0110	✓
	9	1001	✓
	10	1010	✓
grupo 3	7	0111	✓
	14	1110	✓

Columna II

Minterms	Binario	
0,1	000-	✓
0,2	00-0	✓
0,8	-000	✓
1,5	0-01	
1,9	-001	✓
2,6	0-10	✓
2,10	-010	✓
8,9	100-	✓
8,10	10-0	✓
5,7	01-1	
6,7	011-	
6,14	-110	✓
10,14	1-10	✓

Columna III

Minterms	Binario
0,1,8,9	-00-
0,2,8,10	-0-0
~~0,8,1,9~~	~~-00-~~
~~0,8,2,10~~	~~-0-0~~
2,6,10,14	--10
~~2,10,6,14~~	~~--10~~

Algoritmo de Quine-McCluskey

$$f(a,b,c,d)=\sum m(0,1,2,5,6,7,8,9,10,14)$$

Paso 5 (Formar F con los términos no tildados)

Columna I

grupo 0	0	0000	✓
grupo 1	1	0001	✓
	2	0010	✓
	8	1000	✓
grupo 2	5	0101	✓
	6	0110	✓
	9	1001	✓
	10	1010	✓
grupo 3	7	0111	✓
	14	1110	✓

Columna II

0,1	000-	✓
0,2	00-0	✓
0,8	-000	✓
1,5	0-01	
1,9	-001	✓
2,6	0-10	✓
2,10	-010	✓
8,9	100-	✓
8,10	10-0	✓
5,7	01-1	
6,7	011-	
6,14	-110	✓
10,14	1-10	✓

Columna III

0,1,8,9	-00-
0,2,8,10	-0-0
~~0,8,1,9~~	~~-00-~~
~~0,8,2,10~~	~~-0-0~~
2,6,10,14	--10
~~2,10,6,14~~	~~--10~~

$\boxed{f = a'c'd + a'bd + a'bc + b'c' + b'd' + cd'}$

Algoritmo de Quine-McCluskey

$$f(a,b,c,d) = \sum m(0,1,2,5,6,7,8,9,10,14)$$

Resultado obtenido:

$\boxed{\textbf{f = a'c'd + a'bd + a'bc + b'c' + b'd' + cd'}}$

Coincide con el que podríamos obtener por medio . del mapa de Karnaugh?

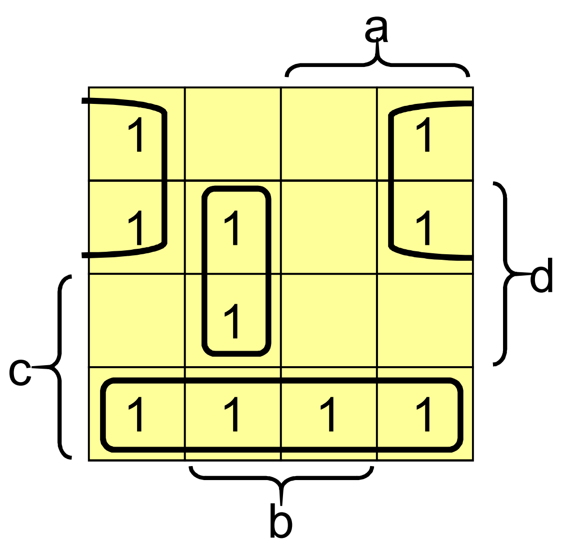
Algoritmo de Quine-McCluskey

$$f(a,b,c,d) = \sum m(0,1,2,5,6,7,8,9,10,14)$$

Resultado obtenido:

$\boxed{\mathbf{f = a'c'd + a'bd + a'bc + b'c' + b'd' + cd'}}$ ← **Quine-McCluskey**

$\boxed{F = a'bd + cd' + b'c'}$ ←



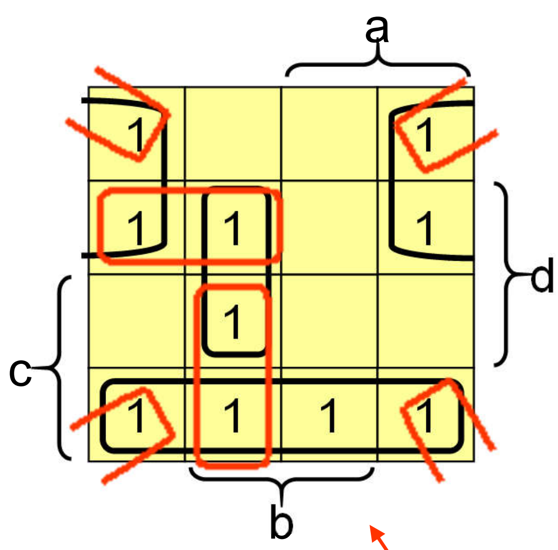
Algoritmo de Quine-McCluskey

$$f(a,b,c,d) = \sum m(0,1,2,5,6,7,8,9,10,14)$$

Resultado obtenido:

f = a'c'd + a'bd + a'bc + b'c' + b'd' + cd' ← **Quine-McCluskey**

F = a'bd + cd' + b'c' ←



Necesitamos un método para eliminar los términos redundantes

"Tabla de implicantes primos"

Tabla de implicantes primos

(Segunda parte del algoritmo de Quine-McCluskey)

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

minitérminos

Implicantes primos

		0	1	2	5	6	7	8	9	10	14
(0,1,8,9)	b'c'	**X**	**X**					**X**	**X**		
(0,2,8,10)	b'd'	**X**		**X**				**X**		**X**	
(2,6,10,14)	cd'			**X**		**X**				**X**	**X**
(1,5)	a'c'd		**X**		**X**						
(5,7)	a'bd				**X**		**X**				
(6,7)	a'bc					**X**	**X**				

¿Cómo sabemos cuáles son los ímplicantes primos esenciales?

Tabla de implicantes primos

(Segunda parte del algoritmo de Quine-McCluskey)

f = a’c’d + a’bd + a’bc + b’c’ + b’d’ + cd’

minitérminos

Implicantes primos

		0	1	2	5	6	7	8	9	10	14
(0,1,8,9)	b’c’	X	X					X	X		
(0,2,8,10)	b’d’	X		X				X		X	
(2,6,10,14)	cd’			X		X				X	X
(1,5)	a’c’d		X		X						
(5,7)	a’bd				X		X				
(6,7)	a’bc					X	X				

Una vez que un implicante fue incluido en F, todos los minitérminos que este abarca dejan de ser tenidos en cuenta para formar F

Tabla de implicantes primos

(Segunda parte del algoritmo de Quine-McCluskey)

f = a’c’d + a’bd + a’bc + b’c’ + b’d’ + cd’

minitérminos

Implicantes primos

		0	1	2	5	6	7	8	9	10	14
(0,1,8,9)	b’c’	X	X					X	X		
(0,2,8,10)	b’d’	X		X				X		X	
(2,6,10,14)	cd’			X		X				X	X
(1,5)	a’c’d		X		X						
(5,7)	a’bd				X		X				
(6,7)	a’bc					X	X				

(0,1,8,9) y (2,6,10,14) fueron incluidos

Una vez que un implicante fue incluido en F, todos los minitérminos que este abarca dejan de ser tenidos en cuenta.

Tabla de implicantes primos

(Segunda parte del algoritmo de Quine-McCluskey)

f = a’c’d + a’bd + a’bc + b’c’ + b’d’ + cd’

minitérminos

Implicantes primos

		0	1	2	5	6	7	8	9	10	14
(0,1,8,9)	b’c’	X	X					X	X		
(0,2,8,10)	b’d’	X		X				X		X	
(2,6,10,14)	cd’			X		X				X	X
(1,5)	a’c’d		X		X						
(5,7)	a’bd				X		X				
(6,7)	a’bc					X	X				

(0,1,8,9) y (2,6,10,14) fueron incluidos

Con los Implicantes primos esenciales no cubrimos toda la función.

Con qué criterio elegimos la cantidad mínima de IP- NE?

Tabla de implicantes primos

(Segunda parte del algoritmo de Quine-McCluskey)

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

Implicantes primos

minitérminos

		0	1	2	5	6	7	8	9	10	14
(0,1,8,9)	b'c'	X	X					X	X		
(0,2,8,10)	b'd'	X		X				X		X	
(2,6,10,14)	cd'			X		X				X	X
(1,5)	a'c'd		X		X						
(5,7)	a'bd				X		X				
(6,7)	a'bc					X	X				

(0,1,8,9) y (2,6,10,14) fueron incluidos

Con los Implicantes primos esenciales no cubrimos toda la función.

Con qué criterio elegimos la cantidad mínima de IP- NE?

Elegimos los implicantes que incluyen mayor cantidad de minitérminos

Bibliografía

-------------------- SISTEMAS NUMERICOS Y ARITMETICA BINARIA -----------------------

- Teoría de Conmutación y Diseño Lógico – Hill F., Peterson G. - Ed. Limusa Capitulo 2

- La PC por dentro – GINZBURG Mario - Ed. Biblioteca Técnica Superior - 3º Edición Apéndice 1
:
- La PC por dentro – GINZBURG Mario - Ed. Biblioteca Técnica Superior - 3º Edición
  Complemento a la unidad 1 (Aritmética binaria, al final del libro)

- MURDOCCA M.J., HEURING V. P."Principios de Arquitectura de Computadoras", Prentice Hall, 2002

- JOHN F. WAKERLY, "Diseño digital: Principios y prácticas", Pearson Educación, 2001

Adicionalmente una referencia a la norma IEEE 754 puede consultarse en:
http://es.wikipedia.org/wiki/IEEE_punto_flotante

-------------- ALGEBRA DE BOOLE Y DISEÑO DE CIRCUITOS COMBINACIONALES ----------

- Introducción a las Técnicas Digitales con CI – Mario GINZBURG - Ed Biblioteca Técnica Superior - 8º Ed.
  Capítulos 4, 5 y 6

- Teoría de Conmutación y Diseño Lógico – HILL F., PETERSON G. - Ed. Limusa Capítulos 3, 4, 6 y 7

- MURDOCCA M.J., HEURING V. P."Principios de Arquitectura de Computadoras", Prentice Hall, 2002 (Apéndice)

- JOHN F. WAKERLY, "Diseño digital: Principios y prácticas", Pearson Educación, 2001