Tema 3: Sistemas Digitales: Circuitos Combinacionales

Fundamentos de Computadores

Curso 2020/21

Índice

Agenda

Introducción

- **Ordenador**: Dispositivo digital $\rightarrow$ Información representada de forma discreta, en lugar de continua.
- **Electrónica digital**: Dos niveles de tensión (alta/baja = 1/0) $\rightarrow$ Uso del sistema de numeración binario como abstracción de dichos estados.
- **Circuitos combinacionales (sin memoria)**: Las salidas dependen sólo de las entradas actuales. (Objeto de estudio en este tema)
- **Circuitos secuenciales (con memoria)**: Las salidas dependen de las entradas y del valor almacenado en su memoria (estado). (Objeto de estudio en asignatura ETC de segundo cuatrimestre)

Agenda

Álgebra de Boole

- El Álgebra de Boole es muy adecuada para expresar y analizar circuitos lógicos. Utilizaremos la Álgebra de Boole formada por $\{\{0,1\},+,\cdot\}$; donde los operadores son la suma lógica (OR) y el producto lógico (AND), y el operator unitario NOT (negación, $\overline{A}$, ‘A)
- Sobre ella se pueden definir funciones de n variables $F : \{0,1\}^n \rightarrow \{0,1\}$ que puede describirse mediante:
  - Ecuación lógica:
    $F(A,B,C) = \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot C + A \cdot B \cdot \overline{C} + A \cdot B \cdot C$
  - Tabla de verdad:

ENTRADAS			SALIDA
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Formas Canónicas. Minitérminos y Maxitérminos

- Una misma función lógica puede expresarse mediante infinitas ecuaciones lógicas. Nos centraremos en las formas normalizadas:
  - Suma de productos (minitérminos):

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

$$F(A,B,C) = m_2 + m_3 + m_6 + m_7 = \sum m(2,3,6,7)$$

  - Producto de sumas (maxitérminos):

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

$$F(A,B,C) = M_0 \cdot M_1 \cdot M_4 \cdot M_5 = \prod M(0,1,4,5)$$

Formas Canónicas. Minitérminos y Maxitérminos (ii)

- Demostración de que las ecuaciones anteriores son equivalentes: supongamos que A = 0, B = 1 y C = 0, entonces
  - Suma de productos (minitérminos):

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

$$F(0,1,0) = 1 \cdot 1 \cdot 1 + 1 \cdot 1 \cdot 0 + 0 \cdot 1 \cdot 1 + 0 \cdot 1 \cdot 0 = 1 + 0 + 0 + 0 = 1$$

  - Producto de sumas (maxitérminos):

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

$$F(0,1,0) = (0+1+0) \cdot (0+1+1) \cdot (1+1+0) \cdot (1+1+1) = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Agenda

Mapas de Karnaugh

- Método sencillo para minimizar funciones lógicas, limitado en la práctica hasta 5 ó 6 variables.
- Mapa de Karnaugh = Representación gráfica de una tabla de verdad. Una celda por cada fila de la tabla, minitérminos adyacentes ocupan celdas adyacentes (incluidas adyacencias en los extremos).

B \ A	0	1
0	0	2
1	1	3

C \ AB	00	01	11	10
0	0	2	6	4
1	1	3	7	5

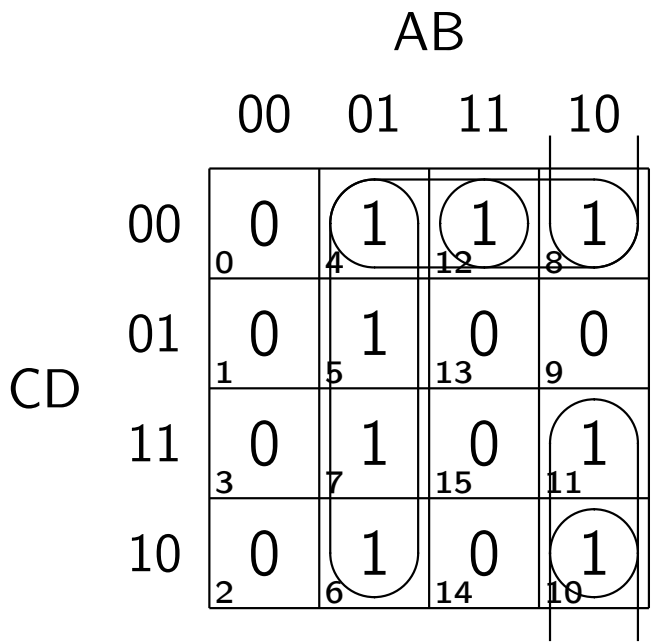
CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

Criterios de simplificación con Mapas de Karnaugh

1. 1 cuadrado tiene n cuadrados adyacentes (1 por variable).
2. Los cuadrados se combinan en grupos de potencias de 2. Al agrupar 2^k celdas, se eliminan k variables.
3. A mayor grupo, menor número de variables en el producto obtenido (puertas AND resultantes con menos entradas).
4. Hay que intentar cubrir todos los unos con el menor número de grupos posibles (puerta OR resultante con menos entradas).
5. Conviene comenzar por los unos más aislados en el mapa (puesto que los otros ofrecen más posibilidades de combinación).

Simplificación con Mapas de Karnaugh

- Simplificar la función
  $F(A, B, C, D) = \sum m(4, 5, 6, 7, 8, 10, 11, 12)$:

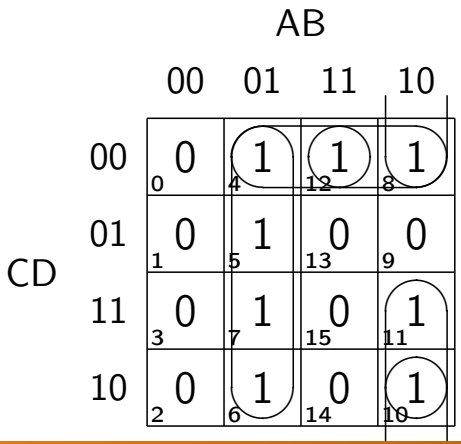


- La expresión simplificada de la función queda:

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

Terminología en la simplificación con Karnaugh

- **Implicante**: Producto de variables cualquiera. $\overline{A}B\overline{C} = \{4, 5\}$.
- **Implicante primo**: Implicante no contenido en otro. $B\overline{C}\,\overline{D} = \{4, 12\}$.
- **Implicante primo esencial**: Implicante primo con al menos un 1 sólo cubierto por él. $\overline{A}B = \{4, 5, 6, 7\}$.
- **Cubierta**: Conjunto de implicantes primos que cubren todos los unos. (Debe incluir, al menos, todos los IP esenciales). $\overline{A}B + A\overline{C}\,\overline{D} + A\overline{B}C = \{\{4, 5, 6, 7\}, \{12, 8\}, \{11, 10\}\}$.



Algoritmo de Minimización con Karnaugh

1. Identificar los implicantes primos. Para esto se busca obtener los grupos con mayor cantidad de unos adyacentes. Los grupos deben contener un número de unos que son potencias de 2.
2. Identificar todos los implicantes primos esenciales.
3. La expresión mínima se obtiene seleccionando todos los implicantes primos esenciales y el menor número de implicantes primos para cubrir los minitérminos no incluidos en los implicantes primos esenciales.

   $\Rightarrow$ Es en forma de suma de productos.

Ejemplo de simplificación por unos

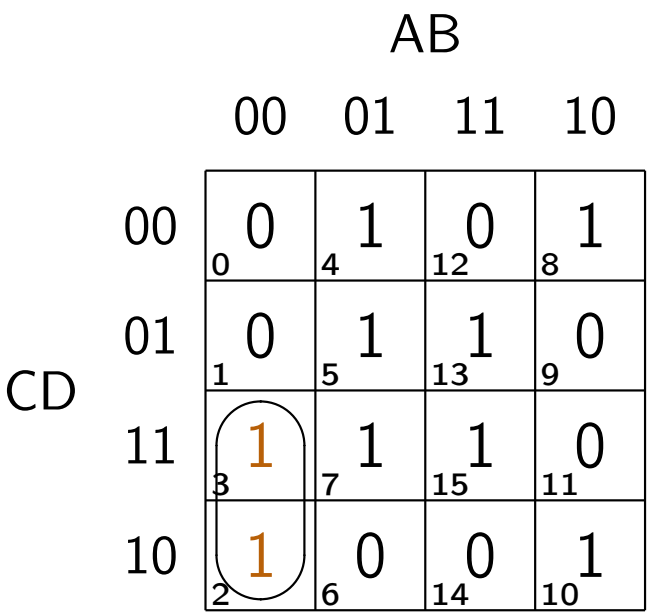
$$F(A,B,C,D) = \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}D + ABCD$$
$$= \sum m(2,3,4,5,7,8,10,13,15)$$

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

Implicantes primos:

Ejemplo de simplificación por unos

$$F(A,B,C,D) = \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD$$
$$+ A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}D + ABCD$$
$$= \sum m(2,3,4,5,7,8,10,13,15)$$

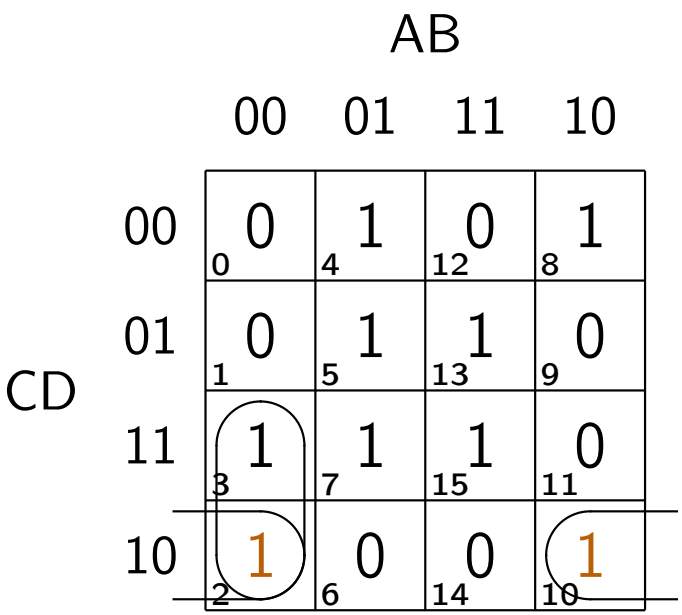


Implicantes primos:

$\bar{A}\bar{B}C$

Ejemplo de simplificación por unos

$$F(A,B,C,D) = \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD$$
$$+ A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}D + ABCD$$
$$= \sum m(2,3,4,5,7,8,10,13,15)$$



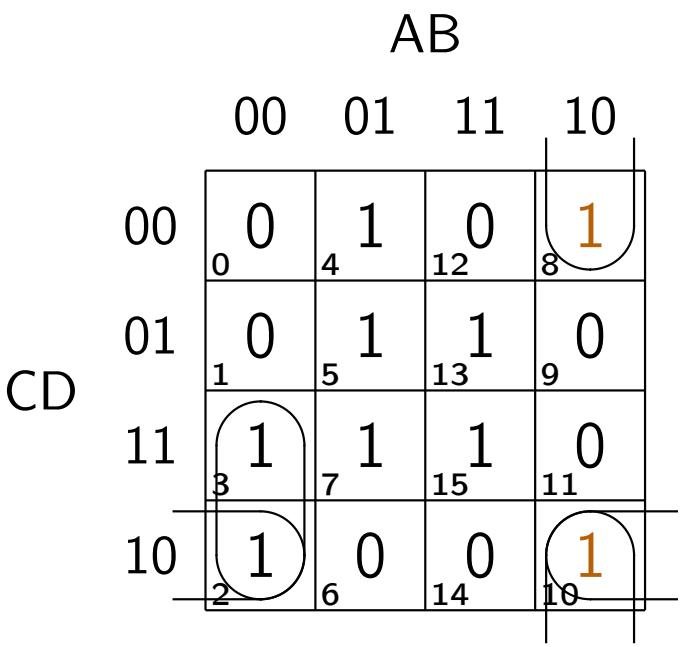
Implicantes primos:

$\bar{A}\bar{B}C$

$\bar{B}C\bar{D}$

Ejemplo de simplificación por unos

$$F(A,B,C,D) = \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD$$
$$+ A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}D + ABCD$$
$$= \sum m(2,3,4,5,7,8,10,13,15)$$



Implicantes primos:

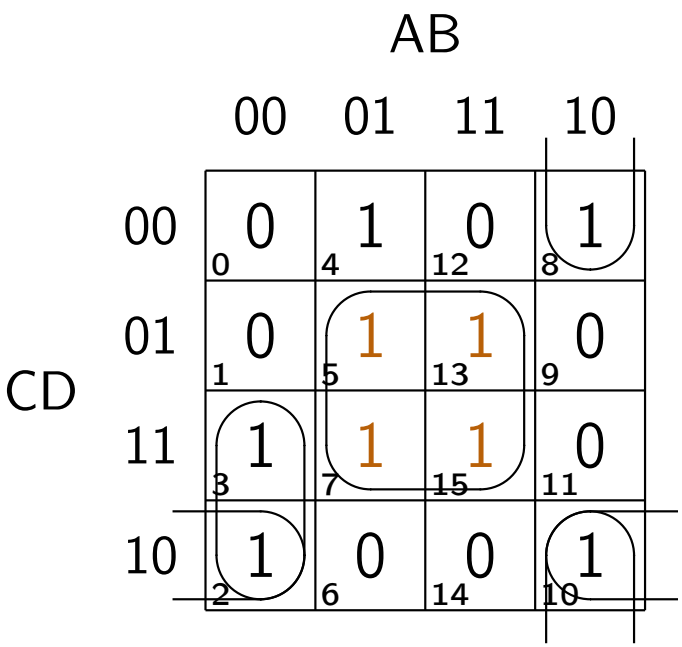
$\bar{A}\bar{B}C$

$\bar{B}C\bar{D}$

$A\bar{B}\bar{D}$

Ejemplo de simplificación por unos

$$F(A,B,C,D) = \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD$$
$$+ A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}D + ABCD$$
$$= \sum m(2,3,4,5,7,8,10,13,15)$$



Implicantes primos:

$\bar{A}\bar{B}C$

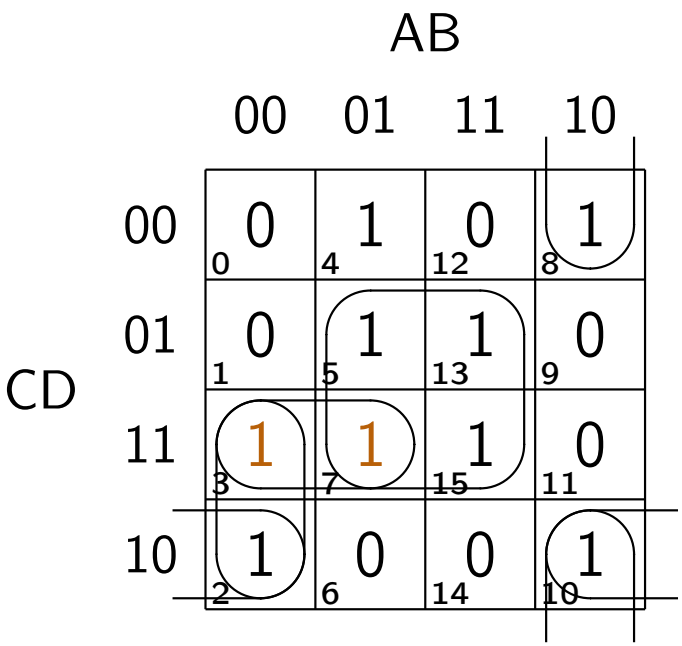
$\bar{B}C\bar{D}$

$A\bar{B}\bar{D}$

BD

Ejemplo de simplificación por unos

$$F(A,B,C,D) = \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}D + ABCD$$
$$= \sum m(2,3,4,5,7,8,10,13,15)$$



Implicantes primos:

$\bar{A}\bar{B}C$

$\bar{B}C\bar{D}$

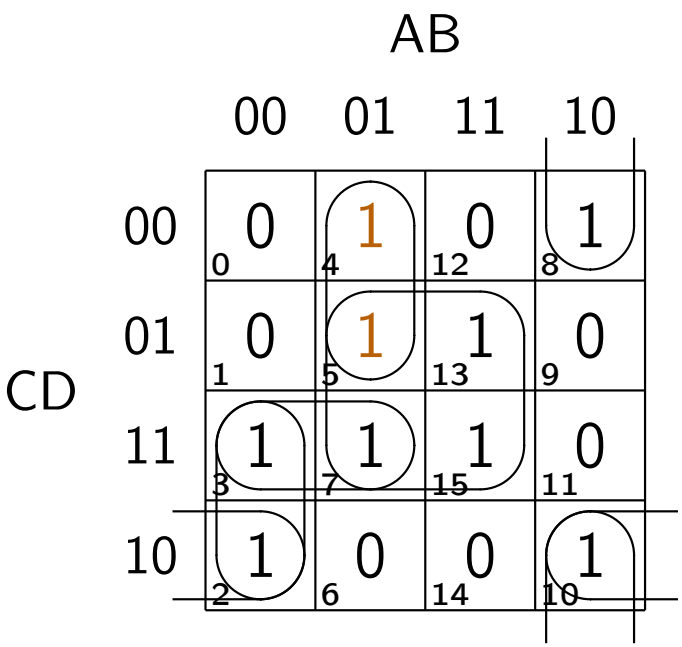
$A\bar{B}\bar{D}$

BD

$\bar{A}CD$

Ejemplo de simplificación por unos

$$F(A,B,C,D) = \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}D + ABCD$$
$$= \sum m(2,3,4,5,7,8,10,13,15)$$



Implicantes primos:

$\bar{A}\bar{B}C$

$\bar{B}C\bar{D}$

$A\bar{B}\bar{D}$

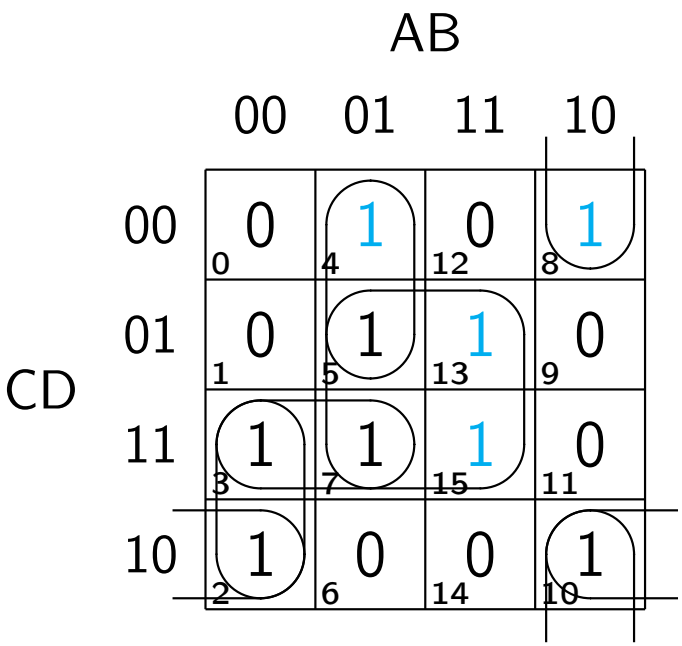
BD

$\bar{A}CD$

$\bar{A}B\bar{C}$

Ejemplo de simplificación por unos

$$F(A,B,C,D) = \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}D + ABCD = \sum m(2,3,4,5,7,8,10,13,15)$$



Implicantes primos:

$\bar{A}\bar{B}C$

$\bar{B}C\bar{D}$

$A\bar{B}\bar{D} \leftarrow$ Esencial

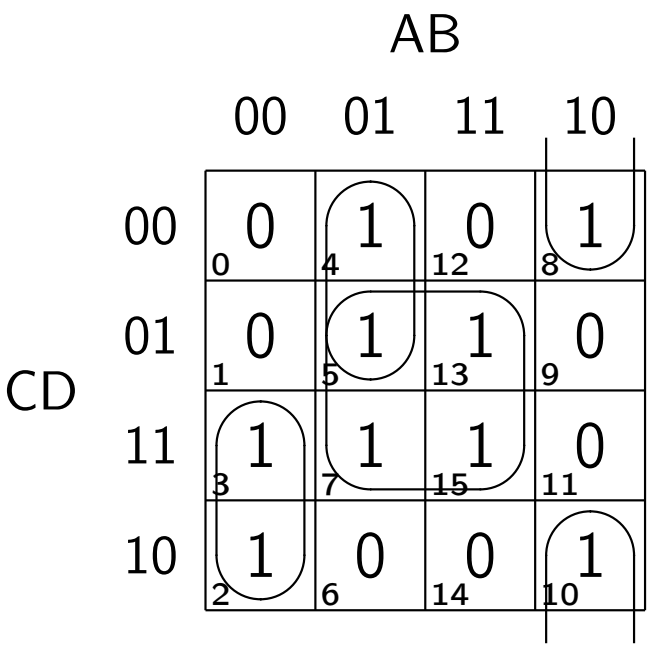
$BD \leftarrow$ Esencial

$\bar{A}CD$

$\bar{A}B\bar{C} \leftarrow$ Esencial

Ejemplo de simplificación por unos

$$F(A,B,C,D) = \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}D + ABCD = \sum m(2,3,4,5,7,8,10,13,15)$$



Implicantes primos:

$\bar{A}\bar{B}C$

$\bar{B}C\bar{D}$

$A\bar{B}\bar{D}$ ← Esencial

BD ← Esencial

$\bar{A}CD$

$\bar{A}B\bar{C}$ ← Esencial

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

Ejemplo de simplificación por ceros

Simplificar *por ceros* la función $\sum m(2,3,4,5,7,8,10,13,15)$

F: CD / AB

	00	01	11	10
00	0 (0)	1 (4)	0 (12)	1 (8)
01	0 (1)	1 (5)	1 (13)	0 (9)
11	1 (3)	1 (7)	1 (15)	0 (11)
10	1 (2)	0 (6)	0 (14)	1 (10)

$\Rightarrow$ $\overline{F}$: CD / AB

	00	01	11	10
00	1 (0)	0 (4)	1 (12)	0 (8)
01	1 (1)	0 (5)	0 (13)	1 (9)
11	0 (3)	0 (7)	0 (15)	1 (11)
10	0 (2)	1 (6)	1 (14)	0 (10)

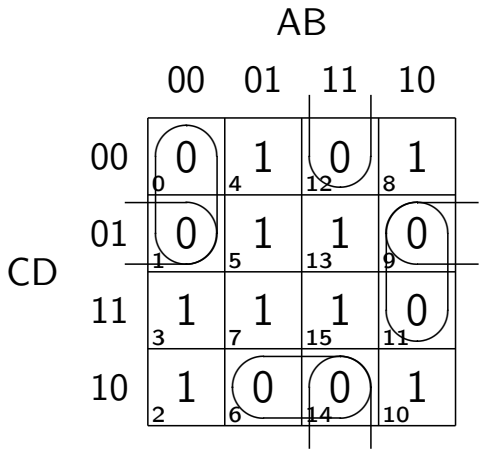
Se muestran los implicantes primos, y una cubierta de la función $\overline{F}$.
El resultado es: $\overline{F} = \overline{A} \cdot \overline{B} \cdot \overline{C} + B \cdot C \cdot \overline{D} + A \cdot B \cdot \overline{D} + A \cdot \overline{B} \cdot D$

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

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

Ejemplo de simplificación por ceros

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



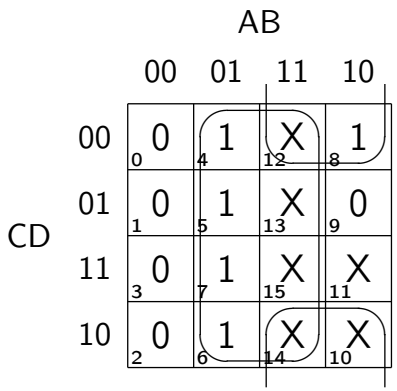
Observaciones:

- La variable que se elimina sigue siendo la que cambia
- De las variables que no cambian, aparecen negadas cuando en las casillas correspondientes aparezca a 1 y sin negar cuando aparezca a 0
- La expresión obtenida es de producto de sumas

Salidas no determinadas

F(A,B,C,D), que valga uno cuando el dígito decimal a la entrada (4 bits), interpretada en binario natural, esté entre 4 y 8 (ambos inclusive):

ENTRADAS				SALIDA
A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X



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

Agenda

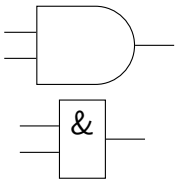
Circuitos Combinacionales Comunes

- **Circuito Combinacional (sin memoria)**: Las salidas dependen sólo de las entradas actuales.
  Ejemplos: multiplexor, memoria de sólo lectura, sumador ...
  Veremos algunos de ellos en este tema, y ampliaremos con otros en ETC (segundo cuatrimestre).
- **Circuito Secuencial (con memoria)**: Las salidas dependen de las entradas y del valor almacenado en su memoria (estado).
  Ejemplos: banco de registros, memoria de datos ...
  Serán objeto de estudio en ETC (segundo cuatrimestre).

Agenda

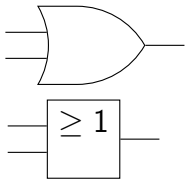
Puertas lógicas básicas

A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1



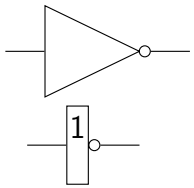
(a) Puerta AND de 2 entradas.

A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1



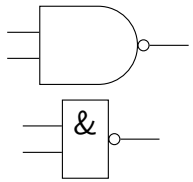
(b) Puerta OR de 2 entradas.

A	$\overline{A}$
0	1
1	0



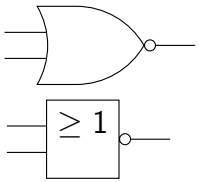
(c) Puerta NOT.

A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0



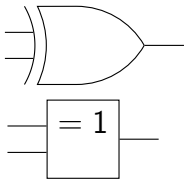
(d) Puerta NAND de 2 entradas.

A	B	$\overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0



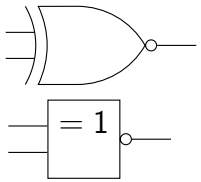
(e) Puerta NOR de 2 entradas.

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



(f) Puerta XOR de 2 entradas.

A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

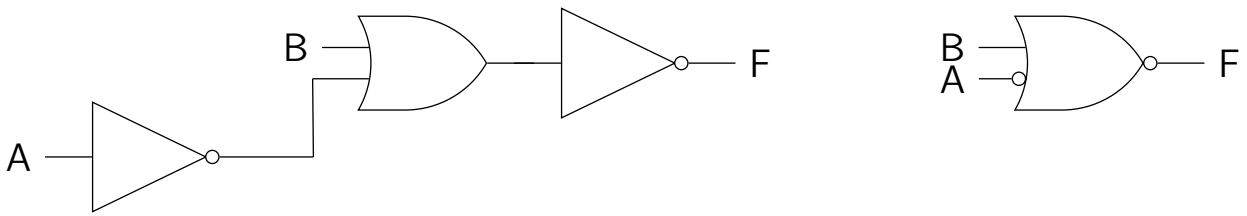


(g) Puerta XNOR de 2 entradas.

- Las puertas lógicas básicas se implementan mediante transistores (de 4 a 6 por puerta).

Implementación de funciones lógicas

Implementación con puertas lógicas básicas de $F = \overline{(\overline{A} + B)}$:

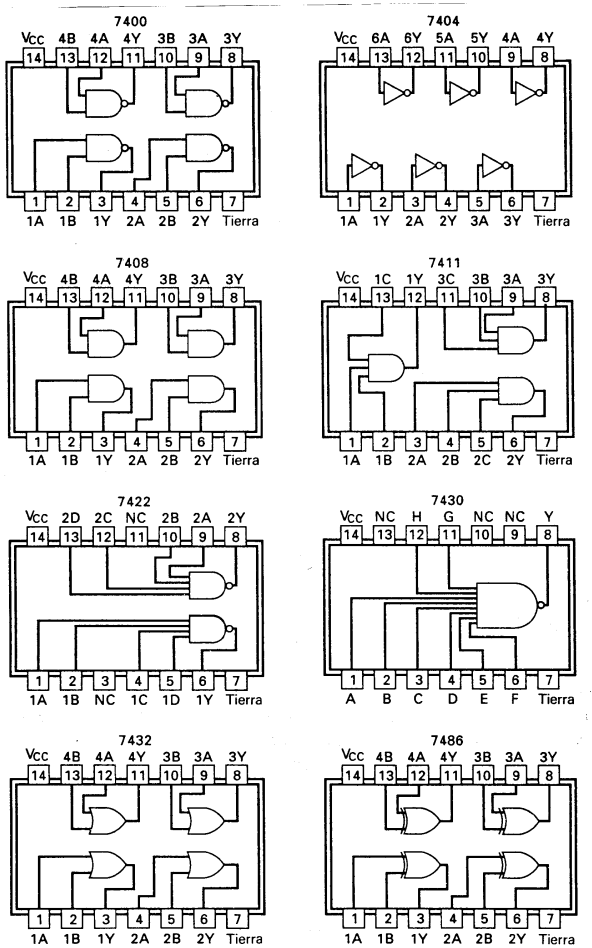


Dibujando explícitamente los inversores (izquierda) o utilizando entradas y salidas con burbujas (derecha)

Implementación física de los circuitos

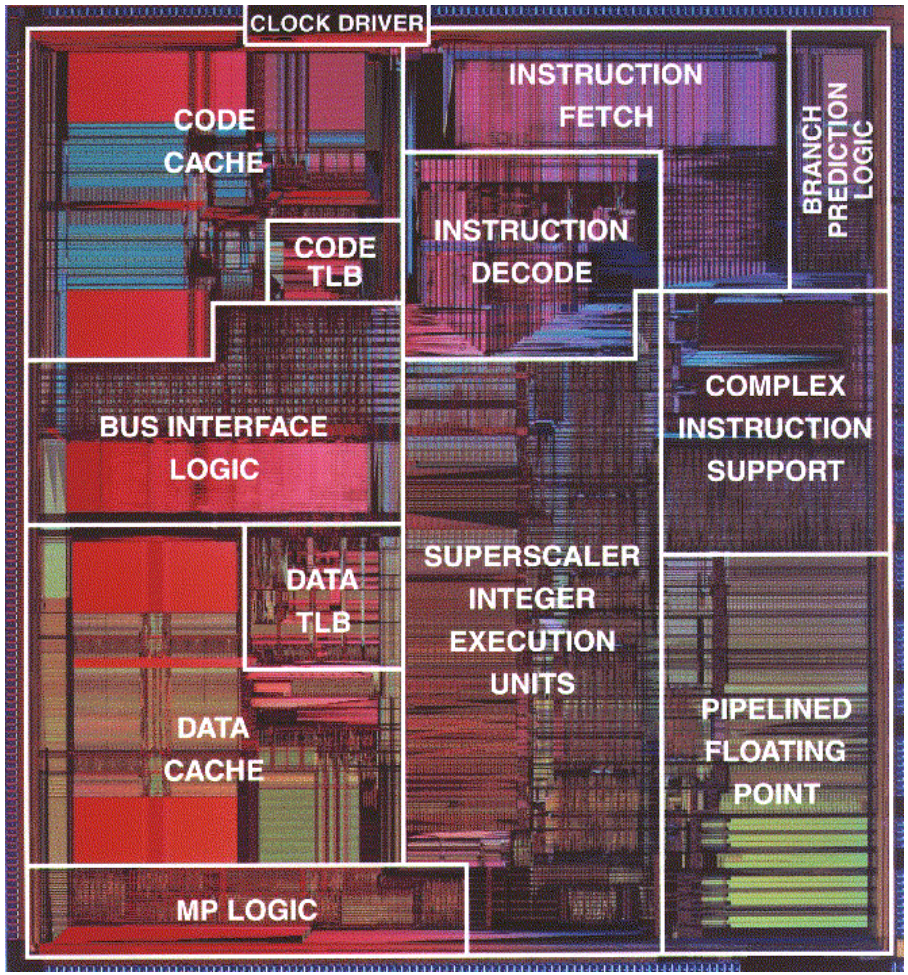
- La mayoría de los circuitos de un sistema electrónico actual se encuentran dentro de chips (circuitos integrados).
- Según la cantidad de puertas lógicas, los podemos clasificar:
  - SSI (c.i. a escala pequeña): 1 a 10 puertas
  - MSI (c.i. a escala media): 10 a 100 puertas
  - LSI (c.i. a escala grande): 100 a 100 000 puertas
  - **VLSI** (c.i. a escala muy grande): Más de 100 000 puertas
- Se construyen mediante técnicas de fotolitografía.
- Los avances tecnológicos en VLSI es lo que ha permitido crear procesadores cada vez más complejos, con más de 1 000 000 000 de transistores.

Implementación física de los circuitos



Algunas pastillas SSI. Disposiciones de patas extraídas de *The TTL Data Book for Design Engineers* (© D.R. por Texas Instruments Incorporated, 1976).

Chips SSI



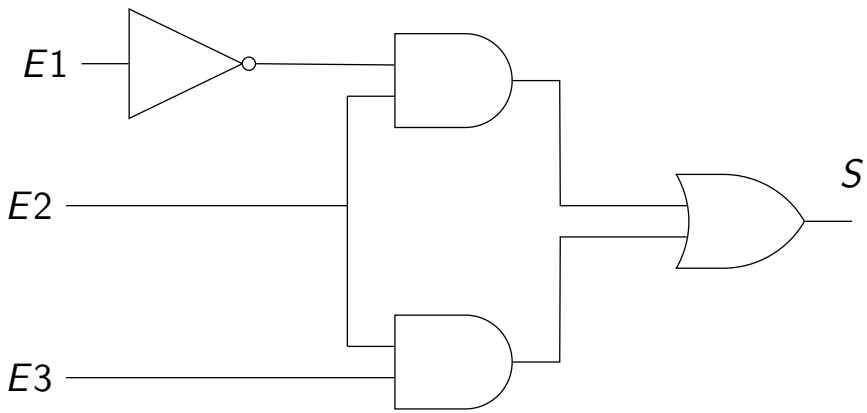
Chip VLSI: Intel Pentium

(más de 3 millones de transistores)

Agenda

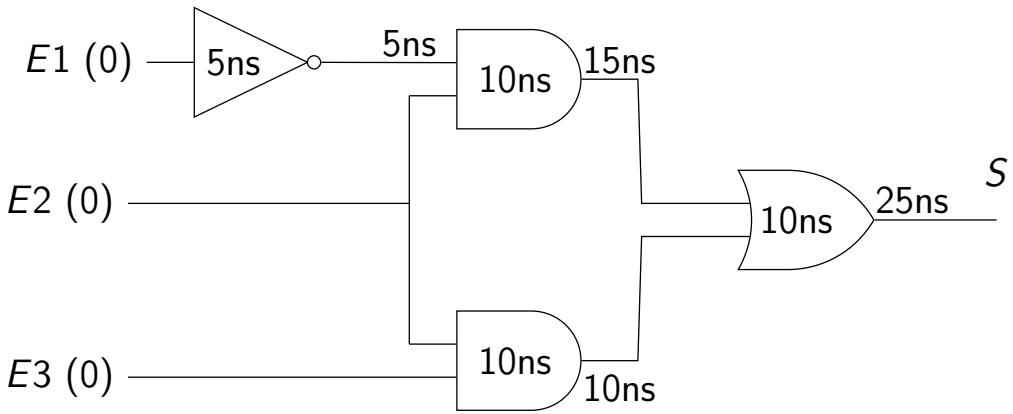
Retardos (i)

- Tiempo que transcurre entre el instante en que un circuito tiene disponibles los valores de señal deseados a la entrada y el instante en que la señal de salida se estabiliza al valor deseado
- Ejemplo: NOT 5 ns, AND 10 ns, OR 10 ns:
  - Tiempo total del circuito de ejemplo = 5+10+10=25ns
  - El AND inferior trabaja en paralelo con la parte superior del circuito.



Retardos (ii)

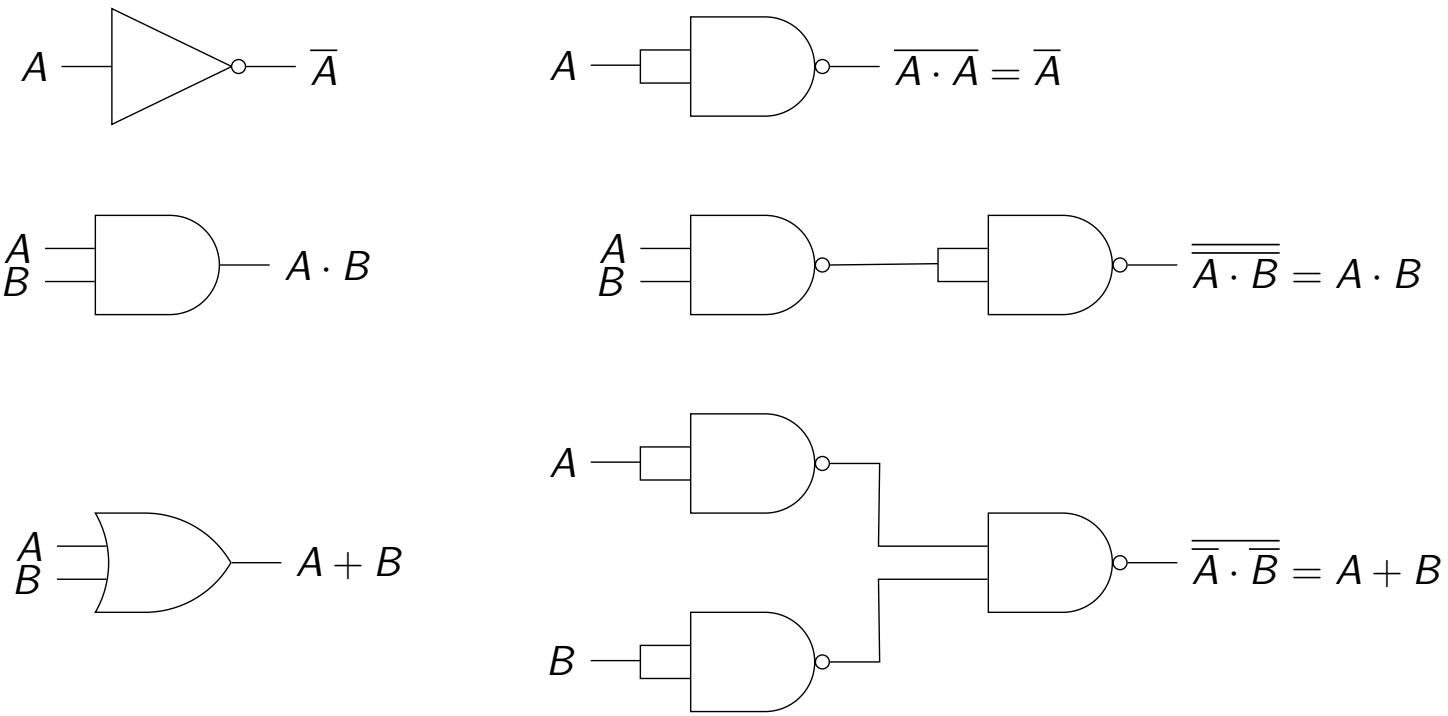
- Tiempo que transcurre entre el instante en que un circuito tiene disponibles los valores de señal deseados a la entrada y el instante en que la señal de salida se estabiliza al valor deseado
- Ejemplo: NOT 5 ns, AND 10 ns, OR 10 ns:
  - Tiempo total del circuito de ejemplo = 5+10+10=25ns
  - El AND inferior trabaja en paralelo con la parte superior del circuito.



Agenda

Implementación con puertas NAND/NOR (i)

- Las puertas NAND/NOR son puertas *universales* $\rightarrow$ es posible implementar cualquier función lógica utilizando únicamente puertas NAND o NOR:



Implementación con puertas NAND/NOR (ii)

- Para implementar con puertas NAND, simplificamos por unos. Sea $F = \sum m(2, 3, 4, 5, 7, 8, 10, 13, 15)$, entonces

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

si a continuación negamos dos veces y aplicamos De Morgan obtenemos:

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

$$= \overline{\overline{(\overline{A} \cdot \overline{B} \cdot C)} \cdot \overline{(\overline{A} \cdot B \cdot \overline{C})} \cdot \overline{(B \cdot D)} \cdot \overline{(A \cdot \overline{B} \cdot \overline{D})}}$$

que puede implementarse directamente mediante puertas NAND.

Implementación con puertas NAND/NOR (iii)

- Para implementar con puertas NOR, simplificamos por ceros. Sea $F = \sum m(2,3,4,5,7,8,10,13,15)$, entonces

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

si a continuación negamos dos veces obtenemos y aplicamos De Morgan obtenemos:

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

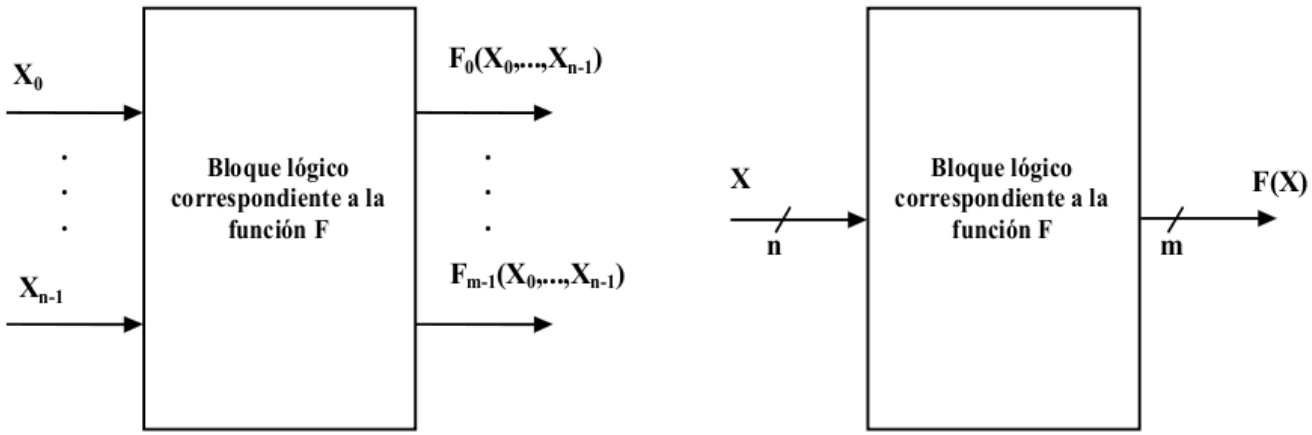
$$= \overline{\overline{(A+B+C)} + \overline{(\overline{B}+\overline{C}+D)} + \overline{(\overline{A}+\overline{B}+D)} + \overline{(\overline{A}+B+\overline{D})}}$$

que puede implementarse directamente mediante puertas NOR.

Agenda

Bloques lógicos

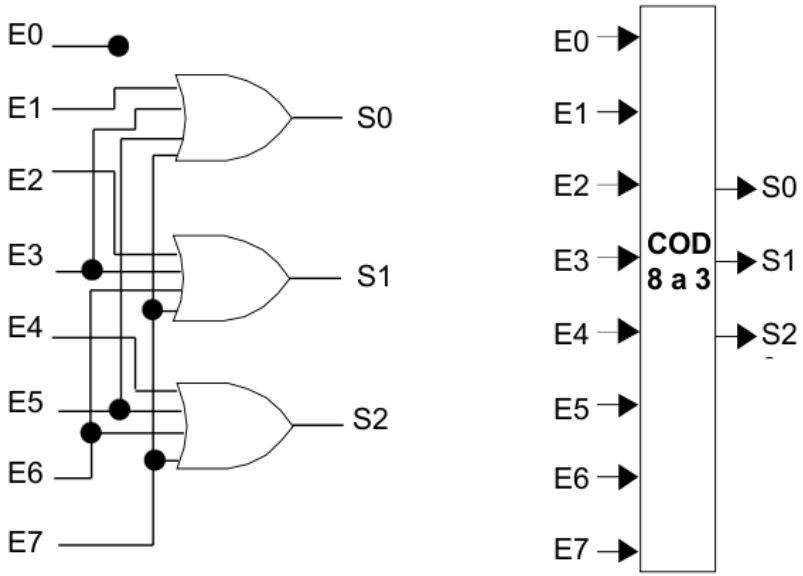
- Conforme construimos funciones lógicas cada vez más complejas se hace inviable representar gráficamente el diagrama de conexión completo con todas las puertas resultantes.
- Utilizaremos bloques lógicos para encapsular y ocultar la complejidad de igual manera que utilizamos funciones y procedimientos al programar:



Agenda

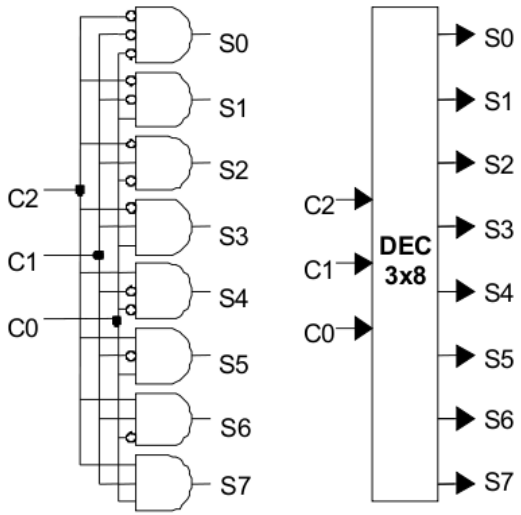
Codificadores y decodificadores

- **Codificador**: Circuito con 2^n líneas de entrada y n líneas de salida.
- Una y sólo una línea de entrada se activa en cada momento. En la salida aparece, codificado en binario, el número de salida activada.
- Es sencillo generalizar para cualquier número de entradas.



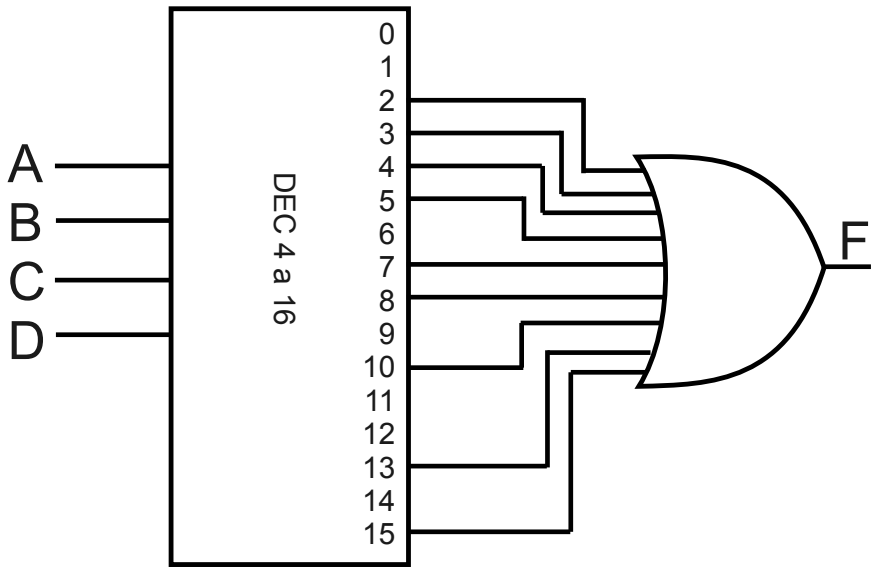
Codificadores y decodificadores

- **Decodificador**: Circuito con n líneas de entrada y 2^n líneas de salida.
- Una y sólo una línea de salida se activa en cada momento. La salida activada es la correspondiente al número binario codificado en la entrada (es un generador de minitérminos).
- También puede generalizarse para cualquier número de entradas.
- Empleado para direccionar posiciones de memoria.



Implementación de funciones con decodificadores

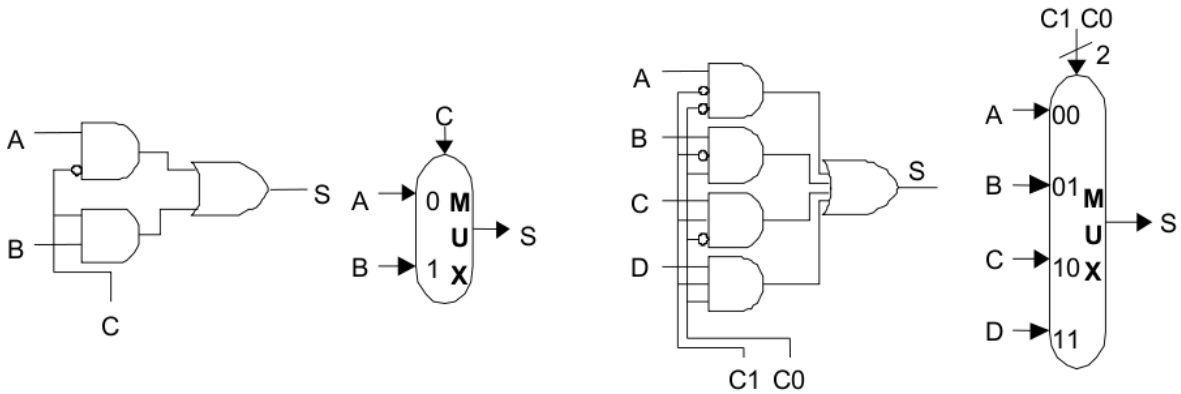
- Sea la función $F(A, B, C, D) = \sum m(2, 3, 4, 5, 7, 8, 10, 13, 15)$, utilizamos un decodificador de 4 a 16, conectando las 4 variables a las 4 entradas del decodificador y conectando a la puerta OR las salidas 2, 3, 4, 5, 7, 8, 10, 13 y 15 del decodificador:



Agenda

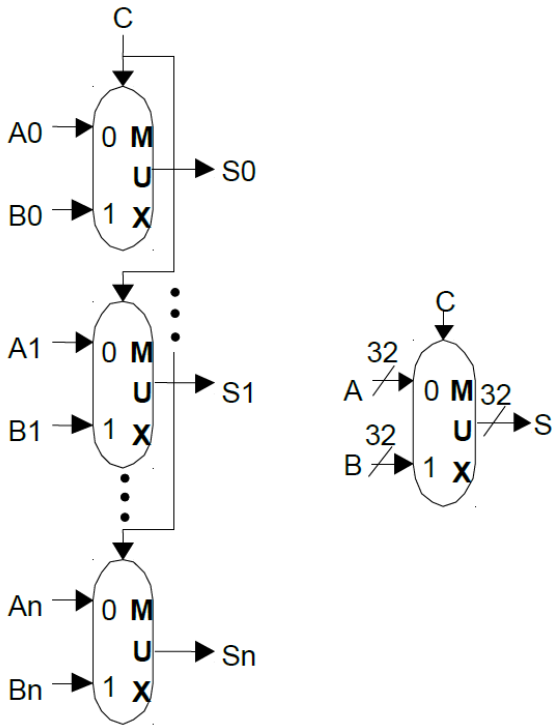
Multiplexores

- **Multiplexores**: 2^n líneas de entrada de datos, n líneas de entrada de control, una sola salida.
- Funciona como un selector de datos: Las n líneas de control seleccionan aquella entrada de datos que se deja pasar hasta la salida.
- También puede generalizarse para cualquier número de entradas. Ejemplos: MUX 2x1 y MUX 4x1.



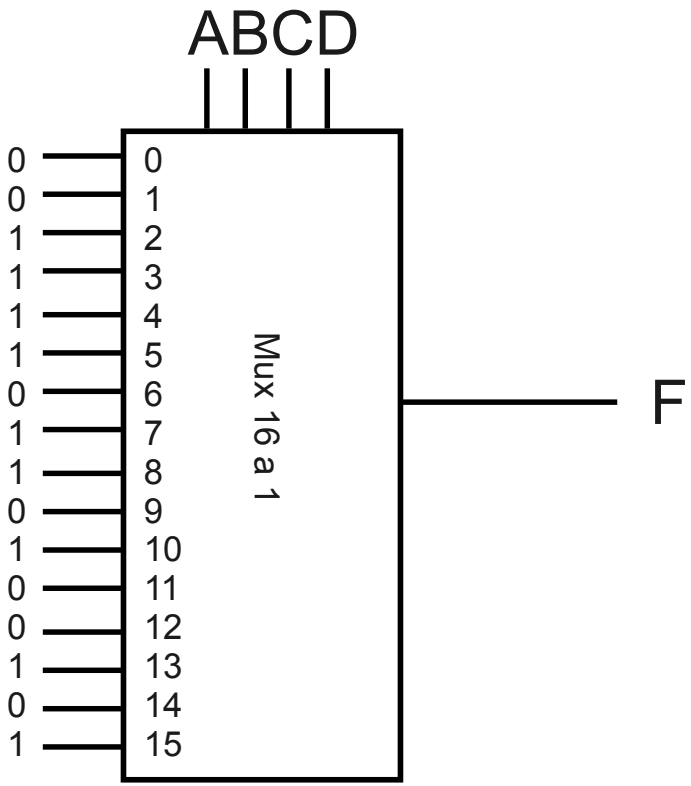
Multiplexores

- Otra posible extensión de los multiplexores es en el ancho de la palabra seleccionada (ancho de entradas de datos y de la salida).
- Ejemplo: MUX 2x1 de 32 bits de ancho, usando 32 MUX 2x1 de 1 bit:



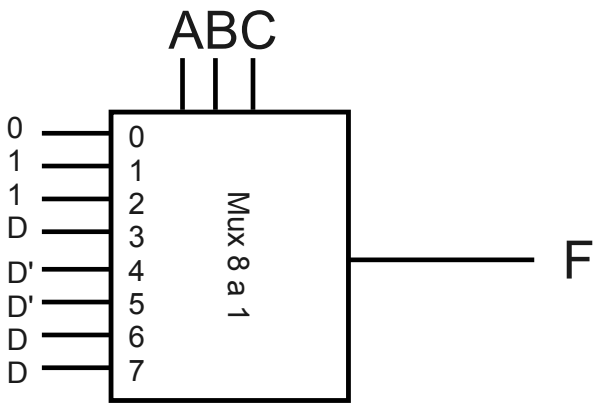
Implementación de Funciones con Multiplexores

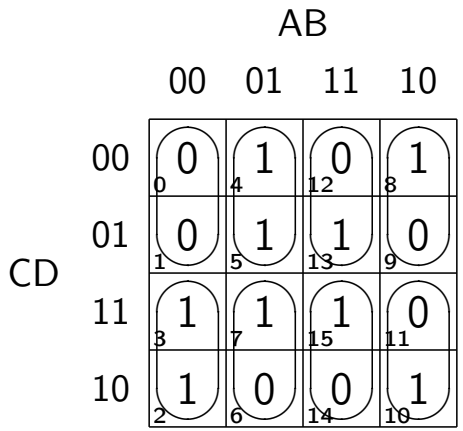
- Sea la función $F(A, B, C, D) = \sum m(2, 3, 4, 5, 7, 8, 10, 13, 15)$, utilizamos un multiplexor 16 a 1, conectando las 4 variables a las 4 entradas de control y poniendo en las entradas los 16 valores de la tabla de verdad:



Implementación de Funciones con Multiplexores

- Si utilizamos un multiplexor de 8 a 1, a cada entrada le corresponde dos entradas de la tabla de verdad con cuatro posibles valores: (0,0), (0,1), (1,0) y (1,1).
- Conectaremos 3 de las variables a las entradas de control y pondremos en las entradas el valor 0, 1, la variable excluida o su negación dependiendo de esas cuatro posibilidades:





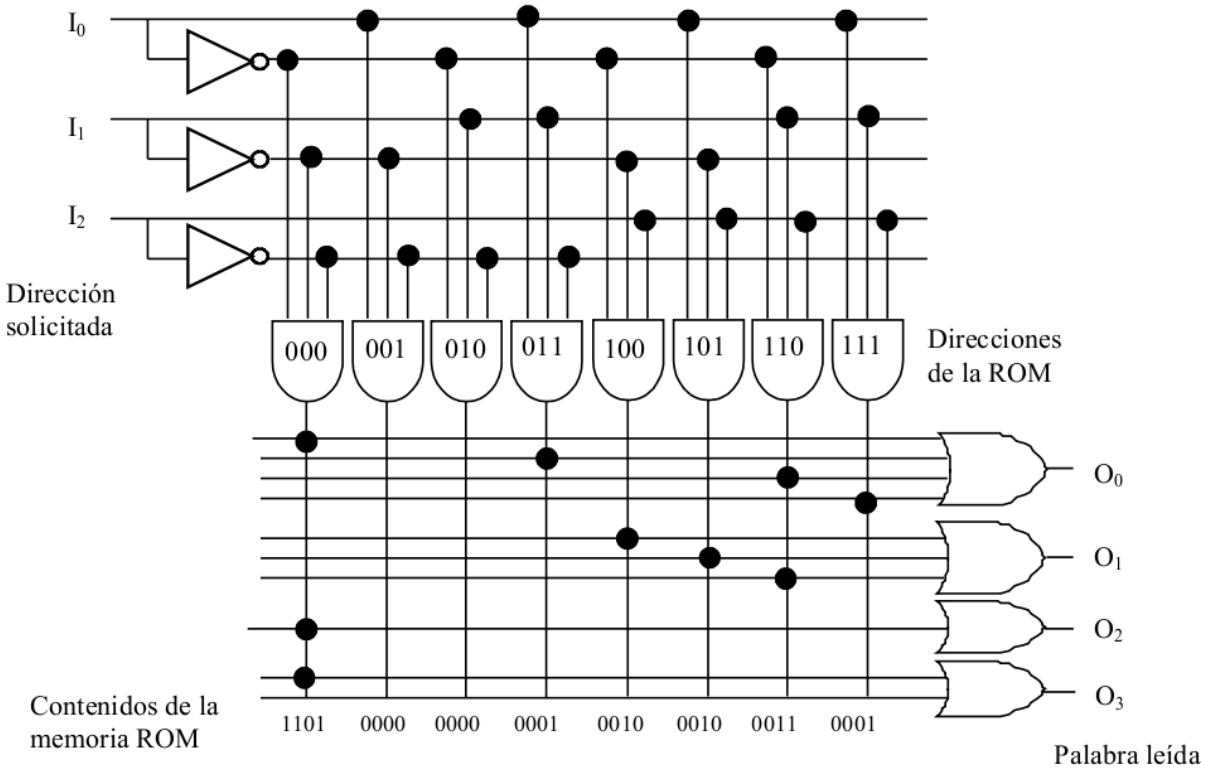
Agenda

Memorias ROM

- Memorias ROM (*Read Only Memory*, memoria de sólo lectura).
- Aunque se llame memoria, es un circuito combinacional.
- m entradas (2^m elementos direccionables, o altura de la ROM).
- n salidas (Cada posición contiene un dato de n bits, anchura de la ROM).
- Forma de la ROM = altura x anchura
- Puede usarse para implementar n funciones binarias distintas dependientes de las mismas m variables de entrada.
- Se implementa usando dos niveles de puertas (aparte de las negaciones de las entradas):
  - Un plano AND, con 2^m puertas de m entradas cada una.
  - Un plano OR, con n puertas de salida.

Memorias ROM

- Esquema de una memoria ROM con 8 posiciones de 4 bits cada una (3 bits de dirección, y anchura de datos 4).



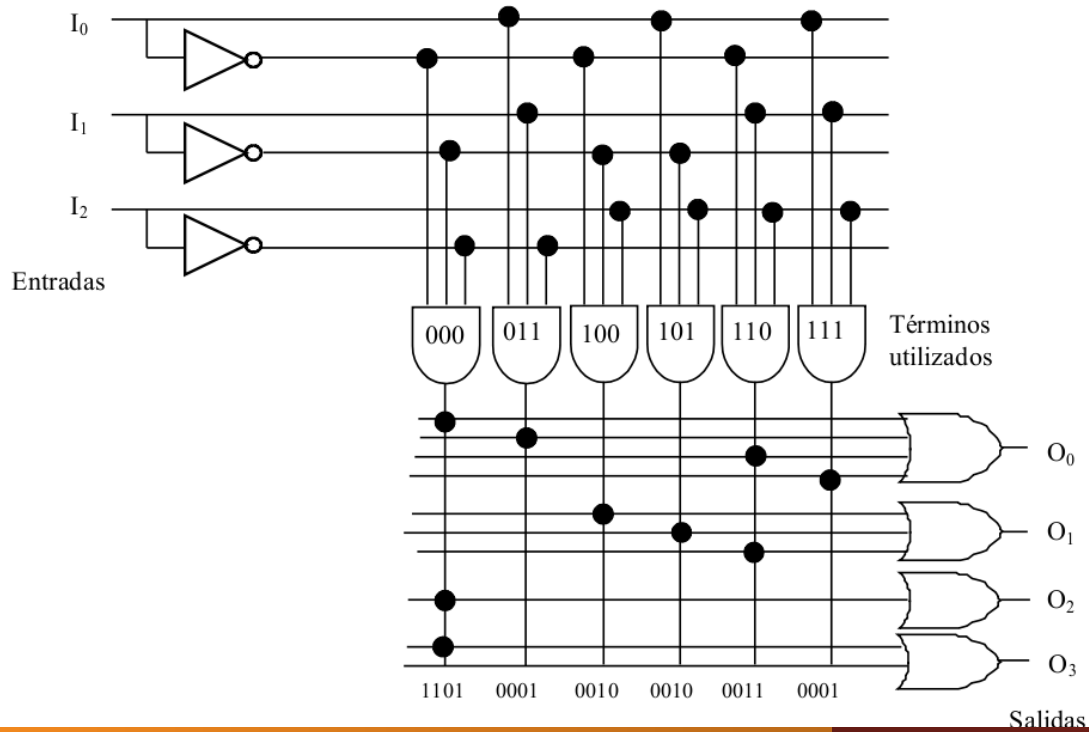
I2	I1	I0	O3	O2	O1	O0
0	0	0	1	1	0	1
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	1
1	0	0	0	0	1	0
1	0	1	0	0	1	0
1	1	0	0	0	1	1
1	1	1	0	0	0	1

Variantes de Memorias ROM

- PROM (*Programmable ROM*, ROM programables).
- EPROM (*Erasable PROM*, PROM borrables).
- EEPROM (*Electronically EPROM*, PROM borrables electrónicamente).
- Memorias Flash (permiten el borrado y reescritura selectivos por bloques, miles de veces).

Arrays Lógicos Programables

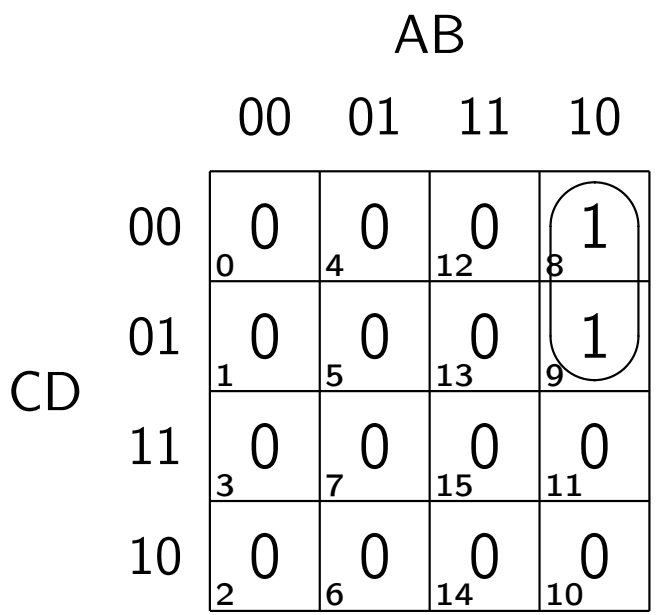
- PLA (*Programmable Logic Array*, array lógico programable).
- Como una ROM, pero sólo se implementan los productos necesarios. Útiles cuando hay muchas entradas, pero sólo unas pocas combinaciones se utilizan realmente. Ejemplo Forma 3x6x4:



Agenda

Propiedad distributiva

- La clave de la simplificación usando mapas de Karnaguh está en usar la propiedad distributiva directamente sobre el álgebra de Boole:



Si unimos las celdas 1001 y la 1000 =>

$$A \cdot \overline{B} \cdot \overline{C} \cdot D + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} =$$
$$A \cdot \overline{B} \cdot \overline{C} \cdot (D + \overline{D}) =$$
$$A \cdot \overline{B} \cdot \overline{C} \cdot (1) = A \cdot \overline{B} \cdot \overline{C}$$

Leyes de Morgan

- Primera ley de Morgan: El complemento de un producto de "n" variables es igual a la suma de los complementos de "n" variables.

$$\overline{(A \cdot B \cdot C \cdot D \cdot \cdots \cdot Z)} = \overline{A} + \overline{B} + \overline{C} + \overline{D} + \cdots + \overline{Z}$$

- Segunda ley de Morgan: El complemento de una suma de "n" variables es igual al producto de los complementos de "n" variables.

$$\overline{(A + B + C + D + \cdots + Z)} = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} \cdot \cdots \cdot \overline{Z}$$

- Estudiadas en la asignatura de Fundamentos Lógicos de la Informática.