Circuitos lógicos combinacionales

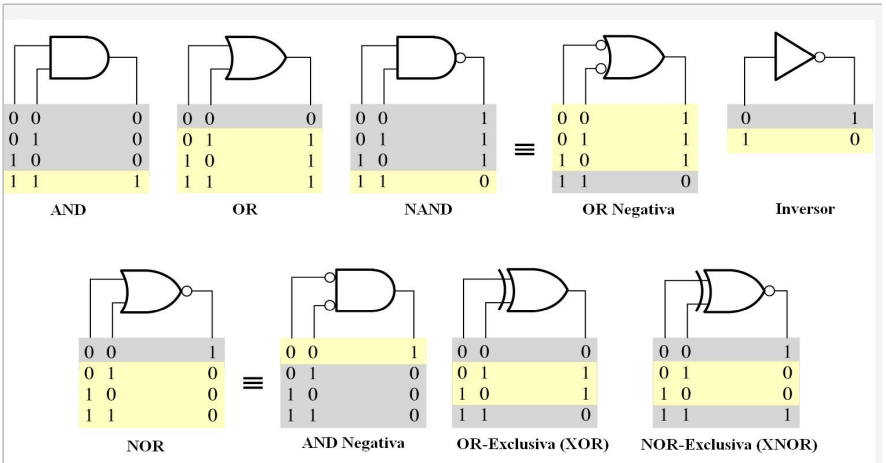
Tema 2

https://youtu.be/r7YNXYGCx7s

¿Qué sabrás al final del capítulo?

- Conocer las formas canónicas de una función
- Implementar funciones con dos niveles de puertas lógicas
  - AND/OR
  - OR/AND
  - NAND
  - NOR
- Analizar sistemas combinacionales, obteniendo la función lógica de salida
- Implementar sistemas combinacionales a partir de su especificación en forma de enunciado con distintos tipos de puertas
- Nociones básicas de memorias

Resumen puertas lógicas



3

Minterm y Maxterm

- Se llama término canónico a aquél que contiene a TODAS las variables de una función dada.
  - Minitérmino o minterm es un producto canónico
  - Maxitérmino o maxterm es una suma canónica
- Ejemplo: Sea F(a,b,c,d). Entonces los términos $a' \cdot b' \cdot c \cdot d$ y $a \cdot b \cdot c \cdot d'$ son minterms y los términos $a'+b+c'+d'$ y $a+b'+c+d'$ son maxterms.
- Para n variables, se tienen 2^n minterms y 2^n maxterms
- Cualquier función booleana pueden expresarse en forma de:
  - Suma de minterms : 1ª Forma canónica
  - Producto de maxterms: 2ª Forma canónica
- Numeración:
  - Minterm: variable → 1, variable → 0

    Ejemplo: $a \cdot b \cdot c \cdot d$ es el minterm 3 (0011). Se representa como m_3
  - Maxterm: variable → 0, variable → 1

    Ejemplo: $a+b+c+d$ es el maxterm 9 (1001). Se representa como M_9

4

Implementación de funciones con minterm y maxterm

1ª Forma canónica: suma de minterms

$$F=\Sigma\, m_i\ (\{i / F(i)=1\})$$

2ª Forma canónica: producto de maxterms

$$F=\Pi\, M_i\ (\{i / F(i)=0\})$$

Notar que $\overline{m_i} = M_i$ y que $\overline{M_i} = m_i$

Ejemplo:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

$F(A,B,C) = \Sigma\ m(2,3,4,6)$ toma los 1's
$F(A,B,C) = \Pi\ M(0,1,5,7)$ toma los 0's

Cálculo de la función negada:

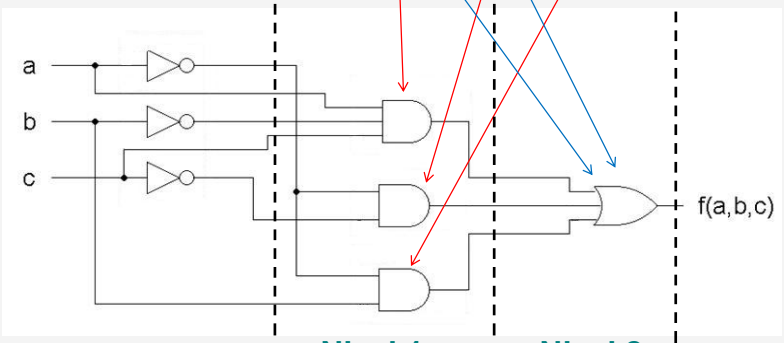
$$\overline{F(A,B,C)} = \overline{\Pi\ M(0,1,5,7)} = \overline{M_0.M_1.M_5.M_7} =$$
$$= m_0+m_1+m_5+m_7 = \Sigma\ m(0,1,5,7)$$

5

Implementación de funciones booleanas con AND/OR

- **Funciones expresadas como suma de productos (AND/OR)**

F(a,b,c) = ab'c + a'c' + a'b



6

Implementación de funciones booleanas con AND/OR
https://youtu.be/zOmJycGpgt0

- **Ejemplo:**

f(x,y,z) =∑(1,3,6,7)

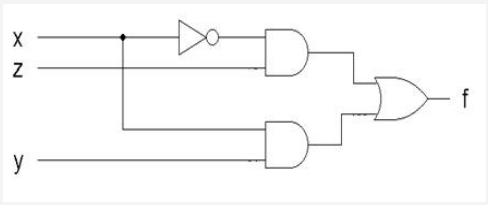
X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

- Esta notación significa la suma de los minitérmino s 1, 3 6 y 7

x \ yz	00	01	11	10
0	0	1	1	0
1	0	0	1	1

Agrupar los 1's de F

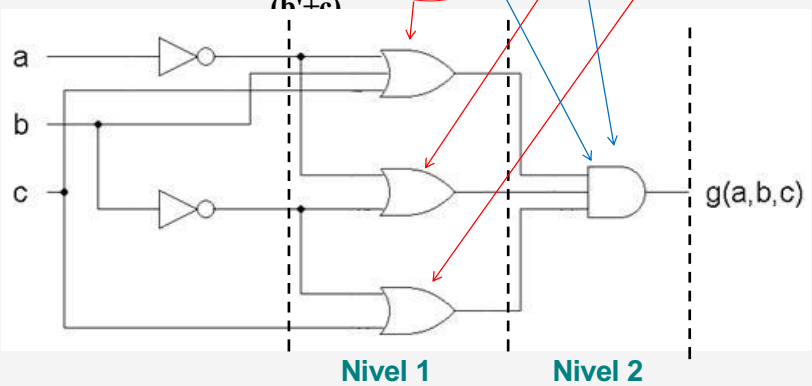
f(x,y,z) = x'z + xy



Implementación de funciones booleanas con OR/AND

- **Funciones expresadas como producto de sumas (OR/AND)**

g(a,b,c) = (a'+b+c) · (a'+b') ·



Implementación de funciones booleanas con OR/AND

Ejemplo

f(x,y,z) =∑(1,3,6,7)

x	y	z	F	$\overline{F}$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	0

F

x \ yz	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$\overline{F}$

x \ yz	00	01	11	10
0	1	0	0	1
1	1	1	0	0

$$\overline{F} = \bar{x}\cdot\bar{z} + x\cdot\bar{y}$$

Agrupar los 1's en $\overline{F}$ y …

… realizar la negación de la función negada

$$\overline{F} = \bar{x}\cdot\bar{z} + x\cdot\bar{y}$$

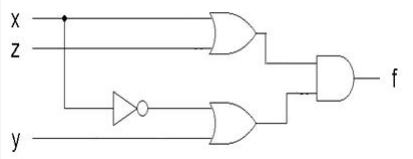
$$F = \overline{\overline{F}} = \overline{\bar{x}\cdot\bar{z} + x\cdot\bar{y}}$$

$$F = \overline{\bar{x}\cdot\bar{z}}\cdot\overline{x\cdot\bar{y}}$$

$$F = (x+z)\cdot(\bar{x}+y)$$

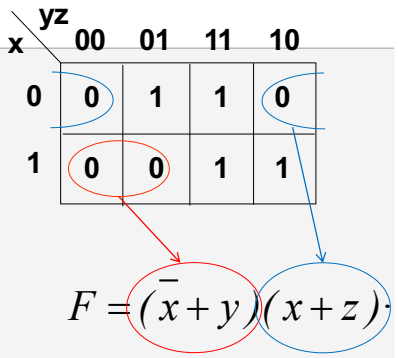
F

x \ yz	00	01	11	10
0	0	1	1	0
1	0	0	1	1



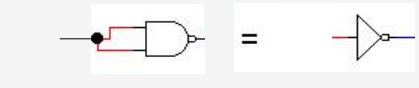
Forma directa:

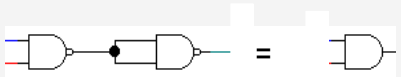
- También se habría llegado a esa expresión agrupando directamente los 0`s de F pero:
  - Cada agrupación de ceros es una suma de variables donde
  - las variables que siempre valen 1 aparecen NEGADAS
  - las variables que siempre valen 0 aparecen AFIRMADAS
  - F es el el producto de todas las sumas

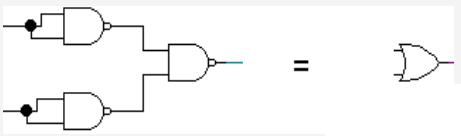


Implementación con puertas NAND

- Las puertas NAND son universales
  - NOT con NANDs
  - AND con NANDs
  - OR con NANDs





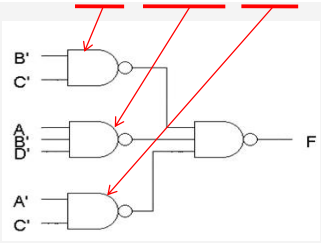


Implementación con puertas NAND en dos niveles

- **Expresar la función como Suma de Productos, aplicar doble negación y De Morgan**

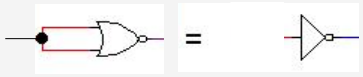
$$F(A,B,C,D) = \overline{B}\cdot\overline{C} + A\cdot\overline{B}\cdot\overline{D} + \overline{A}\cdot\overline{C} = \overline{\left(\overline{\overline{B}\cdot\overline{C} + A\cdot\overline{B}\cdot\overline{D} + \overline{A}\cdot\overline{C}}\right)} =$$

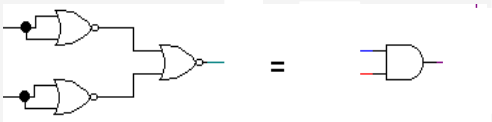
$$= \overline{\left(\overline{\overline{B}\cdot\overline{C}}\right)\cdot\left(\overline{A\cdot\overline{B}\cdot\overline{D}}\right)\cdot\left(\overline{\overline{A}\cdot\overline{C}}\right)}$$



Implementación con puertas NOR

- **Las puertas NOR son universales**
  - **NOT con NORs**
  - **AND con NORs**
  - **OR con NORs**



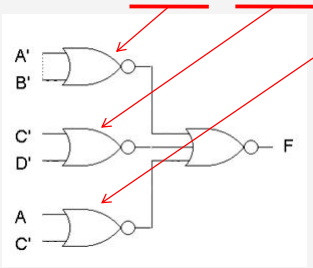


Implementación con puertas NOR en dos niveles

- **Expresar la función como Producto de Sumas, aplicar doble negación y De Morgan**

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

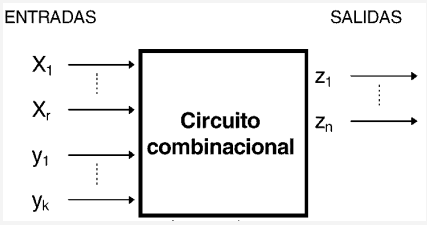
$$= \overline{\left(\overline{(\overline{A}+\overline{B})} + \overline{(\overline{C}+\overline{D})} + \overline{(A+\overline{C})}\right)}$$



Análisis e implementación de sistemas combinacionales

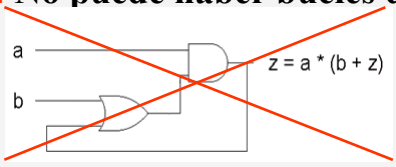
¿Qué es un Circuito Combinacional?

- **Dos tipos de circuitos digitales**
  - Combinacionales**: la salida depende sólo de las entradas**
  - Secuenciales**: la salida depende de las entradas y del estado actual del circuito** ***(entrada + memoria)***

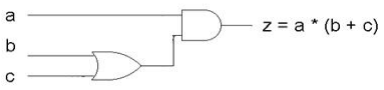


¿Qué es un Circuito Combinacional?

- **Las salidas tienen que estar completamente determinadas a partir de las entradas en cualquier instante**
- **No puede haber bucles de realimentación**



NO **es combinacional**

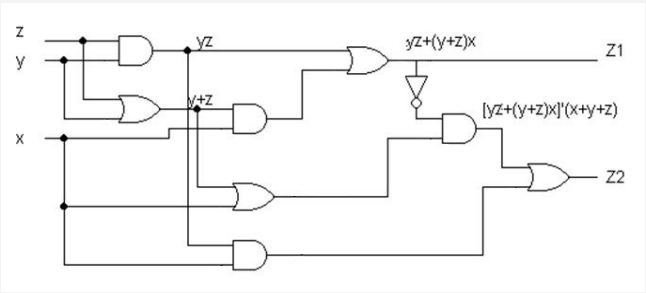


SÍ **es combinacional**

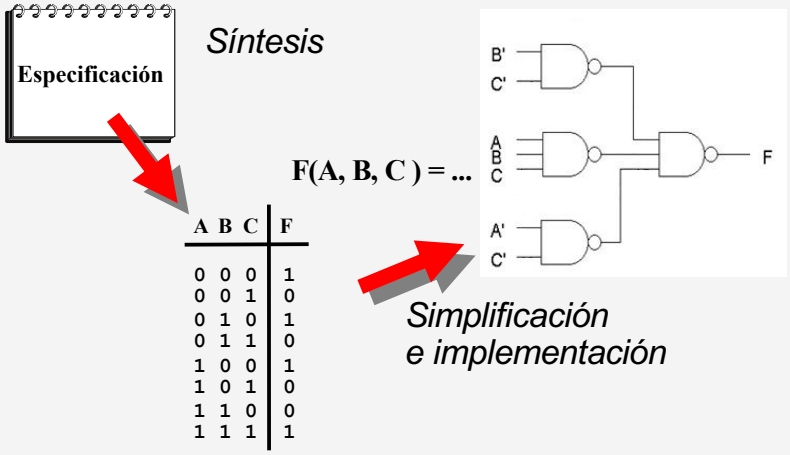
Análisis de circuitos combinacionales

- **Consiste en determinar la expresión algebraica de la función implementada por el circuito**

Se evalúan las expresiones generadas por cada puerta desde su entradas hasta su salida



Síntesis o Diseño de Circuitos Combinacionales



Síntesis o Diseño de Circuitos Combinacionales

Ejemplo

Una máquina expendedora automática proporciona productos con diversos precios: botella de agua 0,50 €, lata de refresco 1,00 €, paquete de galletas 1,50 € y caja de bombones 2,00 €. Sólo admite una moneda de 0,50 €, 1,00 € ó 2,00 € para adquirir el producto y sólo devuelve cambio de 1 moneda, caso de que tuviera que devolver cambio. Habrá casos en los que, al no poder proporcionar el cambio correcto, devolverá la moneda introducida, sin proporcionar el producto.

ENTRADAS		SALIDAS	
Moneda	Producto	¿Suministra?	Cambio
0,00 €	Agua	No	0,00 €
0,00 €	Lata	No	0,00 €
0,00 €	Galletas	No	0,00 €
0,00 €	Bombones	No	0,00 €
0,50 €	Agua	Sí	0,00 €
0,50 €	Lata	No	0,50 €
0,50 €	Galletas	No	0,50 €
0,50 €	Bombones	No	0,50 €
1,00 €	Agua	Sí	0,50 €
1,00 €	Lata	Sí	0,00 €
1,00 €	Galletas	No	1,00 €
1,00 €	Bombones	No	1,00 €
2,00 €	Agua	No	2,00 €
2,00 €	Lata	Sí	1,00 €
2,00 €	Galletas	Sí	0,50 €
2,00 €	Bombones	Sí	0,00 €

Síntesis o Diseño de Circuitos Combinacionales

Codificación

Entradas

Monedas entradas (me_1, me_2)
00: moneda de 0 € (ninguna moneda)
01: moneda de 0,50 €
10: moneda de 1,00 €
11: moneda de 2,00 €

Codificación del producto (t_1, t_2)
00: botella de agua
01: lata de refresco
10: paquete de galletas
11: caja de bombones

Salidas

Monedas retornadas (ms_1, ms_2)
00: moneda de 0 € (ninguna moneda)
01: moneda de 0,50 €
10: moneda de 1,00 €
11: moneda de 2,00 €

Suministro (S)
0: NO proporciona producto
1: SÍ proporciona producto

Tabla de verdad

Entradas		Salidas	
me_1 me_2	t_1 t_2	S	ms_1 ms_2
0 0	0 0	0	0 0
0 0	0 1	0	0 0
0 0	1 0	0	0 0
0 0	1 1	0	0 0
0 1	0 0	1	0 0
0 1	0 1	0	0 1
0 1	1 0	0	0 1
0 1	1 1	0	0 1
1 0	0 0	1	0 1
1 0	0 1	1	0 0
1 0	1 0	0	1 0
1 0	1 1	0	1 0
1 1	0 0	0	1 1
1 1	0 1	1	1 0
1 1	1 0	1	0 1
1 1	1 1	1	0 0

Síntesis o Diseño de Circuitos Combinacionales

Simplificación e implementación de algunas funciones

$me_1\ me_2$ \ $t_1\ t_2$	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	0	1	1	1
10	1	1	0	0

$$S = \overline{me_1} \cdot me_2 \cdot \overline{t_1} \cdot \overline{t_2} + me_1 \cdot me_2 \cdot t_2 + me_1 \cdot me_2 \cdot t_1 + + me_1 \cdot \overline{me_2} \cdot \overline{t_1}$$

$me_1\ me_2$ \ $t_1\ t_2$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	0	0
10	0	0	1	1

$$ms_1 = me_1 \cdot me_2 \cdot \overline{t_1} + me_1 \cdot \overline{me_2} \cdot t_1$$

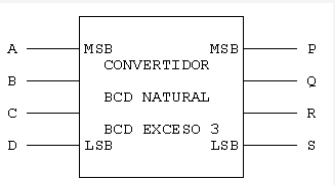
Condiciones “no importa”

- En ocasiones ciertas combinaciones de entradas no tienen sentido o no se pueden dar en el sistema que estamos implementando
- En la tabla de verdad, las variables de salida en las condiciones “no importa” se marcan con (X) o (-)
- A la hora de simplificar, a estos casos “no importa” se les darán los valores que nos convengan para conseguir las simplificaciones más sencillas

Condiciones “no importa”

- **Ejemplo: conversor BCD natural a BCD**

A	B	C	D	P	Q	R	S
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X



P

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$$P = A + B \cdot C + B \cdot D$$

25

Código BCD. Decimal Codificado en Binario

1 Código BCD. Decimal codificado en Binario 2 Código BCD Aiken – Código BCD Exceso 3
3 Código Gray 4 Conversión Binario – GRAY y GRAY – Binario

El Código BCD

Para poder compartir información, que está en formato digital, es común utilizar las representaciones binaria y hexadecimal. Hay otros métodos de representar información y una de ellas es el **código BCD o Decimal codificado en binario**. Con ayuda de este código es más fácil ver la relación que hay entre un número decimal (base 10) y el número correspondiente en binario (base 2)

Este código utiliza 4 dígitos binarios para representar un dígito decimal (0 al 9) (ver en los dos ejemplos que siguen), sin embargo cuando se hace conversión de binario a decimal típica no hay una directa relación entre el dígito decimal y el dígito binario.

26

La representación del mismo número decimal en BCD se muestra en la siguiente imagen.

Ejemplo 2: La conversión directa típica del número decimal 568 a binario es: 568_{10} = 1000111000_2.

La representación del mismo número decimal en BCD se muestra a la siguiente imagen.

27

Para poder obtener el equivalente BCD de cada cifra de los números anteriores, se asigna un "peso" o "valor" según la posición que ocupa. Este "peso" o "valor" sigue el siguiente orden: 8 – 4 – 2 – 1. (Es un código ponderado). Del último ejemplo se observa que el número 5 se representa como: 0 1 0 1.

DECIMAL	BCD
	8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

El primer "0" corresponde al 8, el primer "1" corresponde a 4, el segundo "0" corresponde a 2, y... el segundo "1" corresponde a 1. De lo anterior: 0 x 8 + 1 x 4 + 0 x 2 + 1 x 1 = 5

28

Implementación puertas lógicas

https://youtu.be/cJuHFHGOMcs

29

6. Implementación de circuitos con puertas NAND y NOR

Para la implementación de circuitos lógicos se pueden utilizar cualquier tipo de puertas. Sin embargo la tendencia más común es implementar un circuito empleando solamente un tipo de puertas. De este modo se abaratan costes.

Este método de implementación solo se puede realizar con puertas NAND o NOR, ya que solo estas dos puertas lógicas son **universales**, es decir se puede realizar cualquier circuito lógico y sustituir cualquier puerta empleando únicamente este tipo de puertas, para ello debemos seguir un cierto protocolo aprovechando que una doble negación es igual a una afirmación ($A = \overline{\overline{A}}$).

Siempre podemos negar una expresión lógica dos veces, tantas veces como necesitemos y ésta quedará inmutable, si luego aplicamos el teorema de DeMorgan a una de las dos negaciones anteriores, conseguimos que un producto negado se convierta en una suma de variables negadas, o bien que una suma negada se convierta en un producto de variables negadas.

Ejemplo de ejecución de cualquier puerta empleando solamente puertas NOR.

30

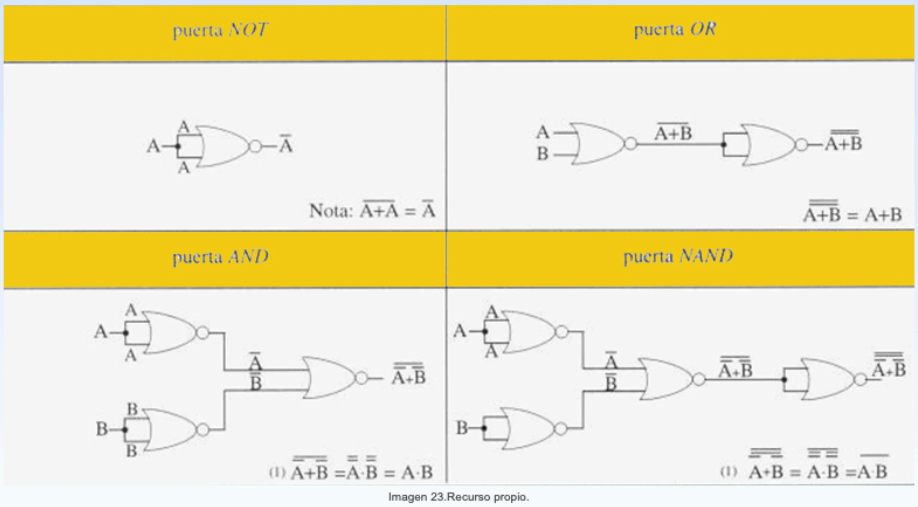


Imagen 23.Recurso propio.

31

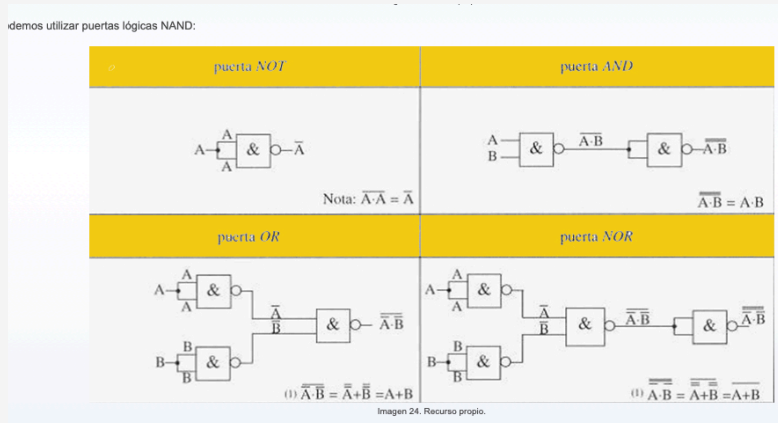


Imagen 24. Recurso propio.

32

La expresión adjunta corresponde a una determinada función lógica:

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

Se desea expresar esta función lógica en forma de productos negados (puertas NAND) o en forma de sumas negadas (Puertas NOR)

Ocultar Información

La función permanece inmutable si la negamos dos veces:

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

Aplicando el teorema de Morgan a la negación:

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

Así habremos conseguido expresar la función en forma de productos negados, es decir de modo que se pueda implementar con puertas NAND.

Si lo que quisiéramos es expresar la función en forma de sumas negadas, (puertas NOR), actuaríamos del siguiente modo:

Se niega toda la función dos veces, y se niegan dos veces todos los productos que aparecen en la expresión lógica:

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

Y aplicando el teorema de DeMorgan a una de las negaciones de los productos:

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

Y así obtenemos la expresión lógica en función de sumas negadas, que es lo se pretendía.

33

Ejemplo o ejercicio resuelto

Implementar la función lógica que se adjunta empleando únicamente puertas NAND. Repetir el ejercicio empleando únicamente puertas NOR.

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

En primer lugar obtendremos la solución **utilizando puertas NAND**.

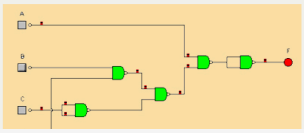
Se aplica una doble negación a toda la expresión y además se niega dos veces la suma:

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

Aplicamos ahora el teorema de Morgan a una de las negaciones de la suma y obtenemos:

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

Que al implementarse quedará:



34

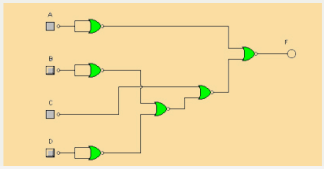
Negamos dos veces toda la expresión y el producto interior del paréntesis:

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

Aplicando el teorema de DeMorgan por un lado a una de las negaciones que abarca a toda la expresión y por otro al producto interior, tendremos:

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

Que al implementarse quedará:



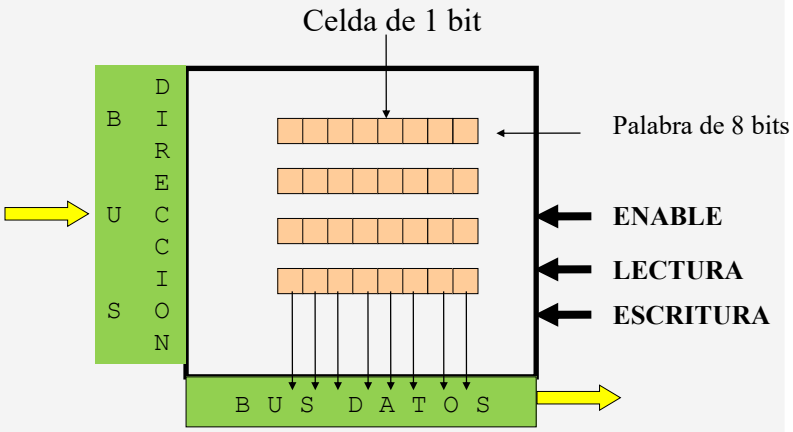
35

Memorias

36

Esquema de una memoria

- Memoria de 4x8 (4 palabras de 8 bits)



Diseño de memorias

- Si se desea una memoria de palabras de *n* bits y se parte de circuitos con ancho de palabra de *t* bits, se necesitarán *n/t* circuitos para alcanzar el ancho de palabra deseado. El valor típico de *t* es 1, 4, 8 o 16.
- Si la capacidad pretendida es *r* palabras y se emplean circuitos de *z* palabras, se necesitarán *r/z* filas de circuitos para lograr dicha capacidad.

Ejemplos de diseño

- Memoria deseada 128 Kpalabras.
- Palabra de 16 bits.
- Circuitos de memoria de 64Kx1.
- Solución:
  - Se necesitan 16/1 = 16 circuitos por fila para formar el ancho de palabra deseado.
  - Se necesitan 128/64 = 2 filas de circuitos.

Otro ejemplo

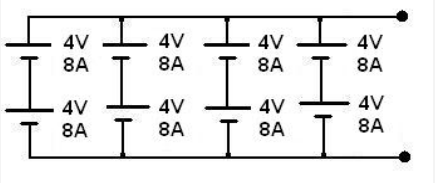
- Memoria deseada 32 Kpalabras.
- Palabra de 8 bits.
- Circuitos de memoria de 8Kx8.
- Solución:
  - Se necesitan 8/8 = 1 circuito por fila para formar el ancho de palabra deseado.
  - Se necesitan 32/8=4 filas de circuitos.

Similitud con la asociación de baterías (I)

- Se desea obtener una batería de 12V y 1A a partir de baterías de 6V y 0,5A
- Solución:
  - Se colocan en serie dos baterías de 6 V, obteniéndose una de 12 V y 0,5 A
  - Se colocan en paralelo dos conjuntos idénticos a los citados en el punto anterior, obteniéndose el resultado deseado

Similitud con la asociación de baterías (II)

- Si todas las baterías son de 4V y 8A, el conjunto resultante será de 8V y 32A



- La tensión equivale a la longitud de palabra y la intensidad de corriente a la capacidad de memoria
- Si 1V=1 bit y 1A=1 K palabra, el circuito anterior se podría ver como una memoria de 32 Kpalabras de 8 bits

Conexionado (I)

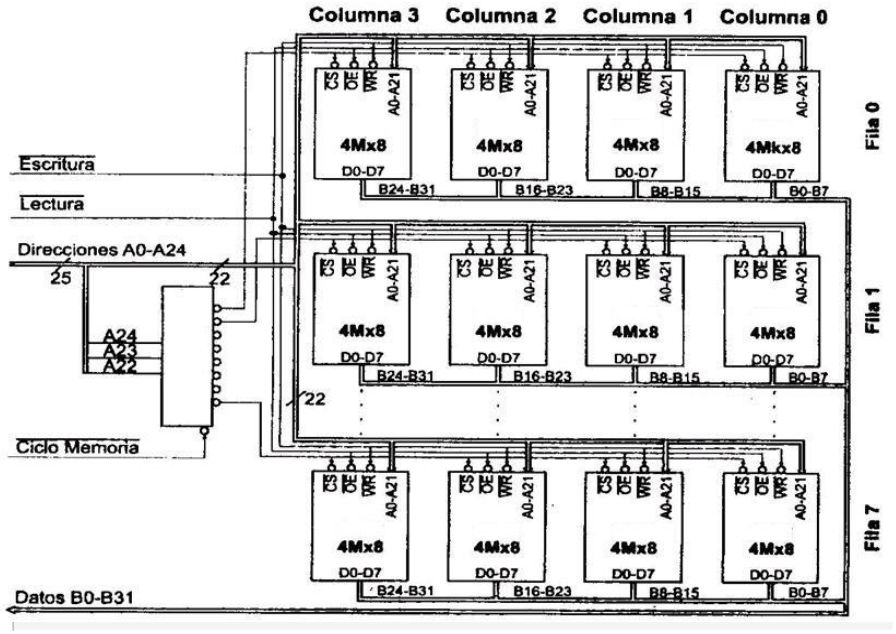
- Bus de datos, bus de dirección, CS, OE y W/R.
- Triestado, se puede conectar sin problema entre sí distintas salidas a un mismo punto eléctrico siempre que no se activen dos o más a la vez.
- Los pines del bus de datos de los circuitos se conectan directamente al bus de datos.
- De esta manera, pueden compartir el bus de datos varios dispositivos simultáneamente

Conexionado (II)

- Para aumentar la longitud de palabra, se usan tantos circuitos como sean necesarios y la asociación de todas las líneas supone una palabra de tamaño mayor.
- Para direccionar en el conjunto, se selecciona la fila de circuitos que contiene la palabra de interés.
- Si los buses no son bidireccionales, se separa la parte de entrada de la parte de salida, pero se siguen aplicando las ideas anteriores.

Memoria de 32Mx32 a partir de módulos de 4Mx8

- Para pasar de palabras de 8 a 32 bits se necesitan 4 módulos, con lo que se tendrán 4Mx32.
- Dicha fila de 4 módulos deberá repetirse otras 7 veces más para completar los 32M.
- Para direccionar cada una de las 8 filas de 4 módulos se utiliza un DEC 3x8, cuyas salidas se conectan con las entradas CS de los módulos de cada fila y en cuya entrada (la del decodificador) se introducen los 3 bits más significativos de los 25 necesarios. Los restantes 22 bits se colocan en paralelo en TODOS los módulos de 4Mx8.



Fuente: Fundamentos de Computadores 9ª Edición. Pedro de Miguel Anasagasti.

Conclusiones

- Es posible implementar una función lógica con cualquiera de estos conjuntos de puertas
  - AND / OR / NOT
  - NAND
  - NOR
- Analizar un circuito combinacional consiste en obtener la función de salida a partir de las entradas y las puertas a las que se encuentran conectadas
- Implementar un circuito combinacional
  - especificación en forma de enunciado
  - síntesis del enunciado en una tabla de verdad
  - simplificación e implementación con un tipo de puertas (p.e. NAND)

Final del Tema 6