Lógica de circuitos. Circuitos combinacionales y secuenciales.

Tema 9

Profesores de Educación Secundaria (PES)

OPOSICIONES 2023

ABACUSNT

Tema de muestra. Esta Página está en blanco a propósito.

Tema de muestra. Esta Página está en blanco a propósito.

- **Sistema sin memoria**: La salida depende únicamente de los valores de la entrada H(x(t)) =y(t)
- **Sistema con memoria**: La salida también depende del estado en el que se encuentra el sistema, H(s(t), x(t)) =y(t).

Adicionalmente, dependiendo de cuándo puede cambiar sus señales, los sistemas se clasifican en:

- **Asíncronos:** Pueden variar sus señales en función de la entrada en cualquier momento.
- **Síncrono:** Se basan en una señal de reloj para regular su funcionamiento.

2.3. Lógica binaria

La lógica binaria establece relaciones funcionales entre variables binarias. Cada función puede representarse por una **tabla de verdad**. Las principales son:

Variables booleanas		**Funciones combinacionales**			
x	y	x AND y	x OR y	x XOR y	NOT x
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

3. Álgebra de Boole

Definición

El álgebra de Boole es un conjunto cualquiera A en el que se han definido dos operaciones binarias denominadas **suma lógica** (+), también conocida como OR, y **producto lógico** (•), también conocido como AND, y una operación unitaria denominada **complemento**, también conocida como NOT.

Se dice que A es un álgebra de Boole si cumple las siguientes propiedades:

Propiedad de cierre.

El conjunto A es cerrado para las dos operaciones, es decir, ∀ a, b ∈ A:

$a+b \in A$ $a \bullet b \in A$

Ley asociativa.

$a \bullet b \bullet c = a \bullet (b \bullet c)$,$\forall\ a,b,c \in A$

Ley conmutativa.

$a \bullet b = b \bullet a \quad ,\forall\ a,b \in A$

Ley distributiva.

∀ a,b,c ϵ A, se verifica:

$a \bullet (b+c) = a \bullet b + a \bullet c$ $\qquad$ $a+(b \bullet c) = (a+b) \bullet (a+c)$

Complementario

∀ a ϵ A, ∃ $\overline{a}$, tal que:

$a+\overline{a} = 1$ $\qquad$ $a \bullet \overline{a} = 0$

3.1. Teoremas del álgebra de Boole

Idempotencia

∀ a ϵ A, se verifica:

$a+a = a$ $\qquad$ $a \bullet a = a$

Elemento identidad

∀ a ϵ A, se verifica:

$a + 0 = a$ $\qquad$ $a \bullet 1 = a$

Absorción

∀ a ϵ A, se verifica:

$a+a \bullet b = a$ $\qquad$ $a \bullet (a+b) = a$

Involución

∀ a ϵ A, se verifica:

$\overline{\overline{a}} = a$ (el complemento del complemento es el elemento original)

Teoremas De Morgan (o Leyes de Morgan)

Es **muy importante**, ya que nos permite transformar funciones lógicas que afectan a varios términos:

∀ a ϵ A, se verifica:

$\overline{(a \bullet b)} = \overline{a} + \overline{b}$ $\qquad$ $\overline{(a+b)} = \overline{a} \bullet \overline{b}$

3.2. Funciones lógicas

Las funciones lógicas son expresiones que contienen sumas y restas de variables binarias (sólo 2 valores lógicos) que pueden estar complementadas o no. El resultado de una función, que será un valor lógico, depende de sus variables.

Las funciones lógicas pueden representarse de distintas formas:

Forma algebraica

Función de ejemplo: F(a,b,c)= a+a•c+a•b
(Utilizaremos esta misma función en las próximas representaciones)

Tabla de Verdad

Se representan todos los valores que puede tomar una función:

a	b	c		F(a,b,c)
0	0	0		0
0	0	1		0
0	1	0		1
0	1	1		0
1	0	0		0
1	0	1		1
1	1	0		0
1	1	1		1

Forma canónica

Se define como **término canónico de una función lógica**, a todo producto o suma (minterm y maxterm respectivamente), en el que aparecen todas las variables de la función en su forma directa o complementada. **Notamos como a' al complementario de a.**

Minterm:

Se toman las salidas de la tabla de verdad **que son 1** y se expresa como suma de términos producto (para la tabla de verdad anterior tenemos):

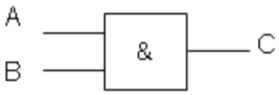
F(a,b,c)= b+a•c+a•b•c =a'•b•c' + a•b'•c + a•b•c

Maxterm:

Se toman las salidas de la tabla de verdad **que son 0** y se expresa como producto de suma de términos (para la tabla de verdad anterior tenemos):

Tema de muestra. Esta Página está en blanco a propósito.

a)



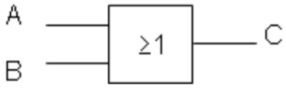
b)

a	b	s
0	0	0
0	1	0
1	0	0
1	1	1

c)

Puerta OR: Salida a 1 si alguna entrada a 1

a)



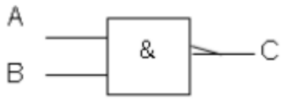
b)

a	b	s
0	0	0
0	1	1
1	0	1
1	1	1

c)

Puerta NAND: Operación NOT AND

a)



b)

a	b	s
0	0	1
0	1	1
1	0	1
1	1	0

c)

4. Circuitos combinacionales

Especificaciones

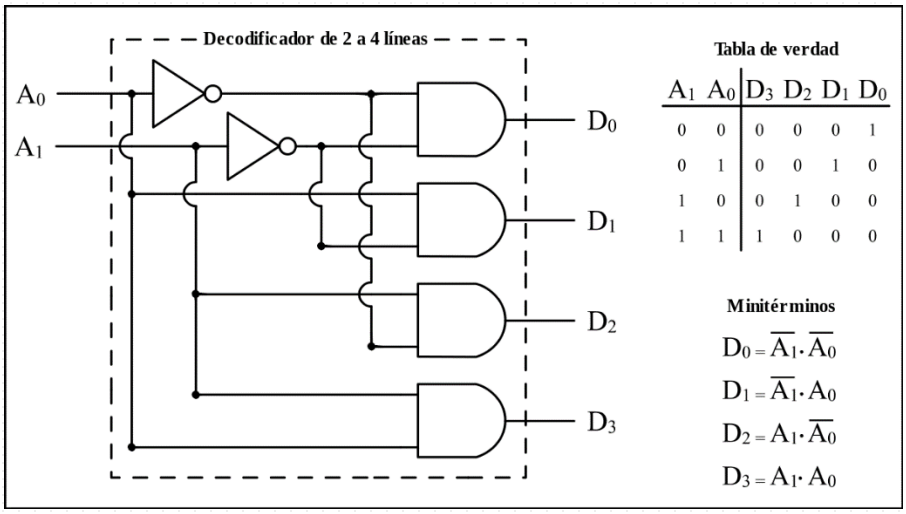
Los circuitos combinacionales dan solución a problemas que pueden expresarse como funciones de conjuntos finitos, las cuales suelen representarse con tablas o expresiones.

Un **ejemplo** podría ser el siguiente, en el que tenemos un sensor, y en función del valor de salida del sensor queremos que la luz de un semáforo se comporte de la siguiente forma:

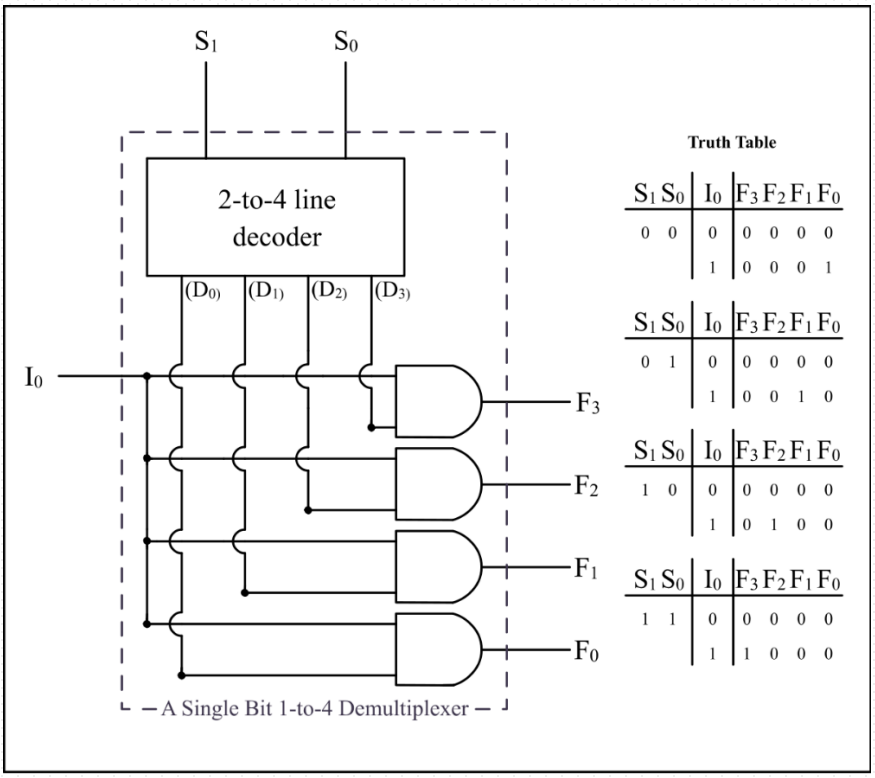
Salidas del Sensor (entrada del circuito)	**X_1**	**Luz del semáforo (Salida del circuito)**	**y_1**
Cerca	1	Rojo	0
Lejos	0	Verde	1

Alguno de los principales módulos estándares de circuitos combinaciones son:

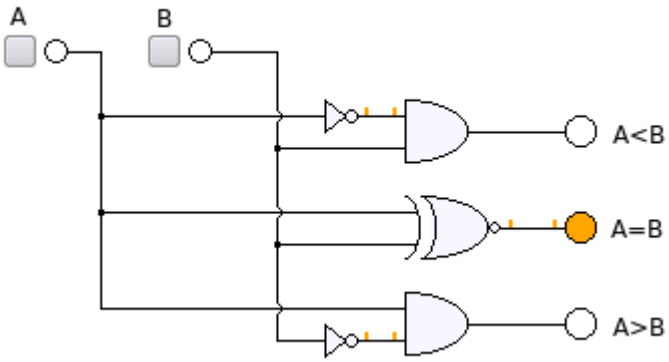
- **Decodificador binario**: Este módulo tiene n entradas, y 2^n salidas. En función de la representación binaria del conjunto de entradas, cuando ésta represente el número i, todas las salidas estarán desactivadas, a excepción de la salida x_i . **Su inverso es el codificador**.



- **Multiplexor**: Se trata de un módulo con n señales de selección, 2^n entradas, y una salida. La salida será igual a la entrada seleccionada por la representación en binario de las señales de selección. Su inverso es el demultiplexor.



- **Comparador**: Tiene 2 entradas de n bits, y 3 salidas cuyas funciones de salida son:
  - Mayor Si x>y
  - Igual Si x=y
  - Menor Si x<y



5. Diseño de circuitos combinacionales

Los pasos para diseñar un circuito combinacional, obteniendo su función canónica son:

1. Codificar las especificaciones de alto nivel en especificaciones **binarias.**
2. Representar las especificaciones en mapas de **Karnaugh** (matriz donde se representa en función de los valores de las entradas, el valor de la salida).
3. Obtener los **minterms**, formando para ello rectángulos de tamaño 2^n de celdas, del mayor tamaño posible, que agrupen a todos los 1 en el mapa de Karnaugh. También se podría hacer agrupando los maxterms, en ese caso sería agrupando ceros.
4. La función lógica de salida es la **suma de los productos** conformado por los minterms (valores de las variables que no variaban en los rectángulos conformados en el paso anterior), obteniéndose la representación canónica
5. Finalmente diseñamos el circuito en función de la función lógica obtenida, que será una **red de tres niveles**: NOT, AND y OR. Para maxterms sería: NOT, OR, AND.

5.1. Simplificación de funciones

En electrónica digital, resulta muy interesante **optimizar un circuito, es decir, simplificarlo**. Se trata de obtener la representación de la función con el menor número de elementos, lo que nos llevará a la hora de la implementación física de dicha función, a obtener un circuito más sencillo y barato.

Existen varios métodos para simplificar funciones lógicas.

Tema de muestra. Esta Página está en blanco a propósito.

Con estas **configuraciones existe una adyacencia gráfica igual a la adyacencia algebraica**. Definimos los términos adyacentes desde el punto de vista lógico como dos términos del mismo tipo mini o maxi) que difieren sólo en una variable.

Por ejemplo, a • b • c' • d' y a • b • c' • d son minterms de cuatro variables adyacentes lógicamente ya que sólo difieren en la variable d. Ambos términos pueden combinarse eliminando una variable (a • b • c' • d' y a • b • c' • d = y a • b • c' • (d + d').

En el mapa de Karnaugh indicamos los términos que se pueden combinar rodeándolos con un trazo. Estos términos al combinarse nos darán una expresión con menos variables.

Además, se interpretará que la fila adyacente a la primera por arriba es ¡la última por abajo y la adyacente a la última columna por la derecha es la primera por la izquierda.

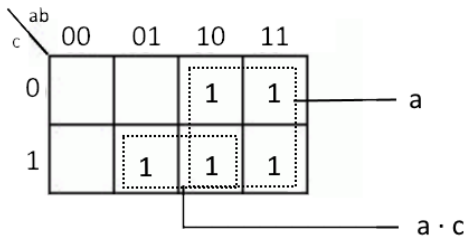
Veamos un ejemplo utilizando minterms:

F = (a + b) • (a + c)

en primer lugar, obtenemos la notación especial abreviada correspondiente a la primera forma canónica

$F(a, b, c) = 011_{(2} + 100_{(2} + 101_{(2} + 110_{(2} + 111_{(2} = m3 + m4 + m5 + m6 + m7 = \sum m\ (3, 4, 5, 6, 7)$

a continuación, colocamos un 1 en cada una de las celdas del mapa de Karnaugh correspondientes a las combinaciones obtenidas. Seguidamente, se agrupan unos adyacentes en conjuntos de potencias de 2 (a partir de 21), es decir, 2, 4, 8, etc., pudiendo haber intersecciones entre los grupos. Cuanto mayor sea el agrupamiento, mayor será la simplificación. A cada grupo de unos le corresponde un ***minterm***, donde se eliminan aquellas variables que aparezcan con valor 0 y 1 dentro del propio grupo y se mantienen, efectuándose entre ellas el producto lógico, aquellas que sólo tengan un valor.



Finalmente, se suman los términos obtenidos lográndose función simplificada.

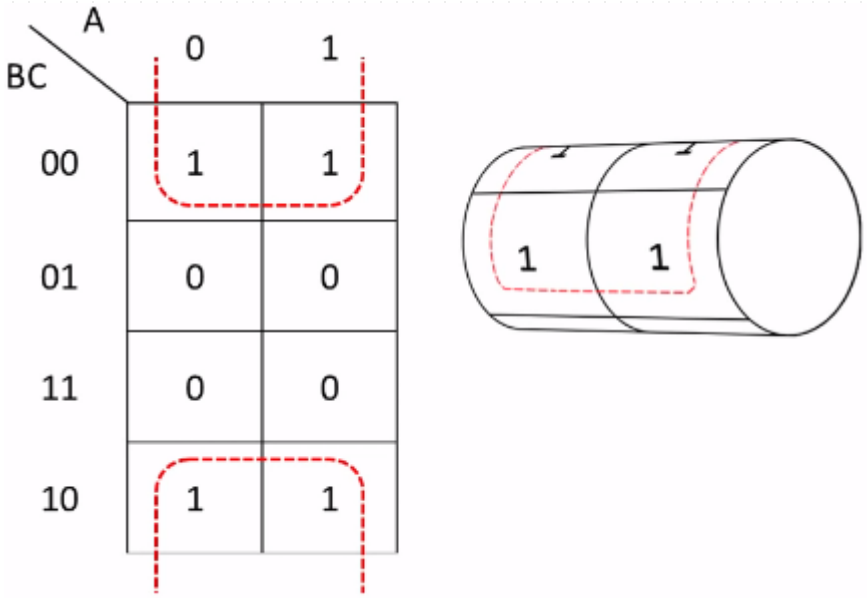
Como se ha indicado, a esta técnica de resolución se la conoce como trabajo en minterms (simplificación utilizando los unos). Existe otra técnica paralela que trabaja con maxterms (simplificación utilizando los ceros) y que no se suele utilizar.

Seguidamente, recogemos con más detalle las reglas a aplicar para la **simplificación por minterms**.

Tema de muestra. Esta Página está en blanco a propósito.

Debes tener en cuenta:

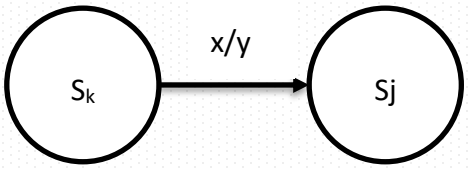
Las agrupaciones deben ser en rectángulos, no vale agrupar términos en diagonales.
Además, si hay términos que pueden agruparse al principio al final de la tabla, lo haremos, porque podemos considerar que están pegados.



6. Circuitos secuenciales

Los circuitos secuenciales tienen como propiedad fundamental que los valores de salida van a estar determinados por los valores de entrada en un momento dado.

El comportamiento de los circuitos secuenciales se representa mediante diagramas de estados, en donde los estados son círculos, y las transiciones flechas, éstas se activan en función de las variables de entrada, por ejemplo:



6.1. Biestables

Los circuitos secuenciales se van a diseñar a partir de los biestables, y se pueden clasificar según la lógica que utilizan en:

Tema de muestra. Esta Página está en blanco a propósito.

Biestable D		
Entrada	Q_{antes}	Q después
D		
0	Q	0
1	Q	1

Biestable T

Si la entrada es cero, la salida no varía, pero si es 1, la salida varía cuando hay algún cambio en la señal de reloj CK.

Biestable T		
Entrada	Q_{antes}	Q después
T		
0	Q	Q
1	Q	$\overline{Q}$

6.2. Ejemplo de Uso de los Mapas de Karnaugh

Dada la siguiente función, descrita por su tabla, vamos a crear el circuito completo:

ENTRADA	ENTRADA	ENTRADA	SALIDA
C	B	A	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Si nos piden obtener la función lógica utilizando la primera forma canónica, tenemos que fijarnos en las filas para las cuales la salida vale “1”. Por la tanto tenemos que la función se puede expresar como suma de minterms:

F=C’B’A + C’BA’+ C’BA+ CB’A + CBA’

El siguiente paso es hacer el mapa de Karnaugh de tres variables:

C \ BA	00	01	11	10
0		1	1	1
1		1		1

El siguiente paso es realizar agrupaciones de tal forma que queden en grupos. En este caso, las agrupaciones que se pueden realizar son:

C \ BA	00	01	11	10
0		1	1	1
1		1		1

En cada agrupación nos fijaremos en las variables que **no cambian** de valor de una celda a otra, dentro de la misma agrupación.

De las 3 agrupaciones realizadas, se obtienen los siguientes valores:

F= BA' + C'B +BA'

Que sería la función simplificada.

Podríamos **llegar a una solución idéntica utilizando las propiedades y teoremas del álgebra de Boole** (método algebraico), pero sin tener claro cual es el objetivo y hasta cuanto podemos minimizar la función, es bastante más complicado de llevar a cabo.

6.3. Técnicas de diseño actuales

El presente tema ha pretendido sentar las bases del diseño de circuitos basándose en la teoría y metodología tradicional. Sin embargo, actualmente no se pueden utilizar mapas K para simplificar circuitos a mano, dada su tremenda complejidad. En su lugar, se utiliza el **algoritmo de Quine &McCluskey**.

El Algoritmo Quine & McCluskey Es un método de simplificación de funciones booleanas desarrollado por Willard Van Orman Quine y Edward J. McCluskey. Es funcionalmente idéntico a la utilización del mapa de Karnaugh, pero su forma tabular lo hace más eficiente para su **implementación en lenguajes computacionales**, y provee un método determinista de conseguir la mínima expresión de una función booleana.

Actualmente, tampoco se puede abordar el diseño de un sistema complejo sin el uso de un HDL (lenguaje de descripción de hardware), para automatizar las tareas de diseño indicando

especificaciones a alto nivel. Los HDL más utilizados son **VHDL, System Verilog** (estandarizado IEEE 1800) **y ABEL**.

7. Conclusión.

A modo de síntesis, se podría indicar que, una vez comprendida la técnica de los mapas de Karnaugh, la base para diseñar circuitos combinacionales y secuenciales es sencilla. Sin embargo, los circuitos de las maquinas comerciales actuales son de una enorme complejidad, es por ello que se hace necesario el uso de técnicas avanzadas de diseño asistidas por ordenador, tales como **Verilog** basadas en los algoritmos de **Quine y McCluskey**.

Actualmente con la aparición en el mercado tecnológico de dispositivos reprogramables como con son los **FPGA** y **CPLD** y el de los lenguajes de descripción de hardware VHDL y Verilog, también se podría decantar el estudio de este tema desde el punto de vista del diseño software de los sistemas digitales tanto secuenciales como e incluso para el montaje de pequeños circuitos utilizando dispositivos microcontroladores programables (PIC) como **Arduino** o **Rasberry Pi Zero**.

7.1. Sistema educativo

Este tema no tiene una aplicación directa en ningún módulo profesional, aunque puede servir de ampliación de contenidos en:

ESO:

- Introducción al Pensamiento Computacional y Robótica (PES)

Grado Medio

- Montaje y Mantenimiento de Equipos (SMR) (PES/SAI)

Grado Superior

- Sistemas informáticos (DAM / DAW) (PES/SAI)
- Fundamentos de hardware (ASIR) (PES/SAI)

8. Bibliografía

- De Anasagasti, Miguel. “Fundamentos de la Computadora” 9ªed 2004 Edt. Paraninfo
- Patterson D.A. y Hennessy JL. “Estructura y diseño de computadoras: la interfaz hardware/software” 4ª Ed. (2005) Edt McGraw-Hill
- Prieto A, Lloris A, Torres JC. “Introducción a la Informática” 4ªed. (2006) Edt. McGraw-Hill
- Jiménez Cumbreras, Isabel Mª “Sistemas Informáticos” 2ªEd (2018) Edt. Garceta
- Morris Mano, Diseño Digital, (2017) Edt Pearson